МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Б.Ю. Жураковський, І.О. Зенів

РОЗРОБКА ТА РЕАЛІЗАЦІЯ МЕРЕЖНИХ ПРОТОКОЛІВ

НАВЧАЛЬНИЙ ПОСІБНИК

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник з дисципліни для студентів,
які навчаються за спеціальністю 121 «Інженерія програмного забезпечення» та
126 «Інформаційні системи та технології»
спеціалізацією «Інженерія програмного забезпечення інформаційно управляючих
систем» та «Інформаційне забезпечення робототехнічних систем»

Київ
КПІ ім. Ігоря Сікорського
2020

Рецензенти *Манько О.О.,* д.т.н., професор, професор кафедри телекомунікацій Одеської академії зв'язку ім. О.С. Попова
Марков С.Ю., д.т.н., доцент, доцент кафедри телекомунікаційних технологій Державного університету телекомунікацій

Відповідальний редактор *Батрак Є.В.,* канд. техн. наук, доц.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 10 від 18.06.2020 р.) за поданням Вченої ради Факультету інформатики та обчислювальної техніки (протокол № 10 від 18.05.2020 р.)

Електронне мережне навчальне видання

Жураковський Богдан Юрійович, доктор техн. наук, проф.
Зенів Ірина Онуфріївна, канд. техн. наук, доц.

РОЗРОБКА ТА РЕАЛІЗАЦІЯ МЕРЕЖНИХ ПРОТОКОЛІВ
НАВЧАЛЬНИЙ ПОСІБНИК

Розробка та реалізація мережних протоколів: Навчальний посібник [Електронний ресурс]: навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» та 126 «Інформаційні системи та технології», спеціалізації «Інженерія програмного забезпечення інформаційно управляючих систем» та «Інформаційне забезпечення робототехнічних систем»/ Б. Ю. Жураківський, І. О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 11,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2020. – 462 с.

Розробка та реалізація мережних протоколів важлива частина сучасної галузі знань, що необхідна для актуального забезпечення взаємозв’язку рівнів та різних технологій будь-якої локальної і глобальної мереж. Мережеві протоколи базуються на міжнародних стандартах, що забезпечують якісну взаємодію різних інноваційних технологій та різних елементів мережі. Вони складають семирівневу структуру, яка здійснює забезпечення вирішення інженерно-технічних питань та потребує постійно оновлювати, вдосконалювати та розроблки нових протоколів, як правила взаємодії всіх складових глобальної мережі. Розробка та реалізація мережних протоколів потребує постійного розвитку та вдосконалення для надання абонентам високонадійних видів послуг з високошвидкісною передачею даних.

ЗМІСТ

ВСТУП

Навчальний посібник «Розробка та реалізація мережних протоколів» розроблений для студентів, які навчаються за напрямком підготовки 121 «Інженерія програмного забезпечення» та 126 «Інформаційні системи та технології» з навчальної дисципліни «Розробка та реалізація мережних протоколів» і грає важливу роль у підготовці фахівців. Він необхідний для професійно-практичної підготовки бакалаврів, що у майбутньому мають можливість використовувати знання та інженерно-фахові вміння в інноваційних галузях широкого та спеціалізованого профілю всесвітньої глобальної мережі.

Знання бакалавра базується на таких дисциплінах: інтеграція інформаційних систем, розподілені системи обробки інформації, архітектура комп'ютера, організація комп'ютерних мереж.

Даний посібник буде фундаментом знань для переліку дисциплін: корпоративні інформаційні системи та технології, програмування Інтернет, методологія і технології побудови інформаційних систем, структурно-функціональний аналіз складних ієрархічних систем.

Основним призначенням даного посібника є: поглиблення теоретичних знань, набуття практичних навичок з застосування міжнародної нормативної бази мережевих протоколів та їх застосування у роботі мережевих пристроїв та інших елементів мережі; набуття навичок дослідження процесів передачі даних у незахищених та захищених локальних мережах при використанні різних мережевих технологій та протоколів, а також набуття навичок усунення вразливостей та протидії мережним атакам на вузли та комунікаційні пристрої мереж. Такий підхід забезпечує набуття студентами комплексних знань та практичних навичок у сфері професійної діяльності.

Цей матеріал є теоретична підготовка майбутніх спеціалістів до рішення

задач у системному просторі та формування у студентів здатностей формалізації задачі, вибору методів її рішення та аналізу результатів. Завдання матеріалу, забезпечити студенту вміння: розробляти мережеві протоколи; проводити класифікацію протоколів мереж по визначеним параметрам; реалізовувати мережеві протоколи; застосовувати мережні протоколи при побудові різноманітних комп'ютерних мереж; проектувати протоколи локальних, транспортних мереж та мереж доступу на базі міжнародних стандартів електрозв'язку. Це забезпечить досвід вибору та використання мережних протоколів та інші навички.

Комп'ютерні практикуми та самостійна робота мають на меті закріпити теоретичні знання студентів, допомогти їм оволодіти прийомами розробки та реалізації мережних протоколів, використовуючи конкретні масиви даних в умовах комп'ютеризованого робочого місця; застосовувати відповідні способи та засоби обробки даних, таких як кодування та стиснення даних, виконувати чисельну оцінку параметрів систем, мереж за допомогою відомих методів з використанням наведеного масиву даних.

Обговорюються питання стандартизації і викладається загальна характеристика моделі ISO/OSI та всі міжмережеві протоколи, що забезпечує якісну роботу. Аналізуються основні технології локальних мереж, устаткування мереж, що працює на фізичному і канальному рівнях, – структуровані кабельні системи, мережеві адаптери, повторювачі і концентратори різних технологій, а також мости і комутатори. Послідовно розглядаються принципи і механізми об'єднання мереж на основі протоколів мережевого рівня.

Внаслідок вивчення дисципліни студент використовувати знання: архітектури комп'ютерних мереж; принципи структурування мереж; методи передачі на різних рівнях; характеристики ліній зв'язку; принципи стандартизації в комп'ютерних мережах; технології Ethernet, Token Ring, FDDI локальних мереж; етапи діагностики мережі; структуру та основні протоколи; типи адресації в IP-мережах.

1. ОСНОВНІ ПОНЯТТЯ

1.1. Узагальнені поняття мережевих протоколів

У світі все підпорядковано правилам або законам, що визначають якісну роботу та функціональний взаємозв'язок елементів системи, в нашому випадку, локальної та глобальної мережі людства. Тому необхідно щоб при обміні даними між комп'ютерами мережі передбачалося, що дані без спотворення та втрати будуть доставлені від відправника адресату. Для цього необхідно, щоб різнотипні комп'ютери, комунікаційні пристрої, мережне обладнання та програмне забезпечення виконували передавання даних за однаковими чітко визначеними правилами. Такі правила називаються мережними протоколами.

Більшість сучасних комп'ютерних мереж здійснюють передавання даних на основі набору протоколів, що має назву TCP/IP (англ. Transmission Control Protocol / Internet Protocol – протокол управління передаванням / міжмережний протокол) [1.1].

Дані, що передаються мережею, розбивають на невеликі пакети та доповнюють даними, що стосуються процесу передавання: адресами комп'ютерів одержувача та відправника, номером та довжиною пакета то що. Кожний пакет передається окремо каналом зв'язку. Маршрут передавання визначають маршрутизатори, вони також слідкують за доставкою пакетів. Якщо пакет з якоїсь причини не потрапив до адресата, він буде повторно відправлений. Після досягнення пункту призначення всі пакети з'єднуються, і дані набувають початкового вигляду. Пакети, у яких виникають спотворення даних під час передавання, передаються повторно [1.2].

Правила розбиття даних на пакети, їх доставки до адресата й об'єднання пакетів в єдине ціле визначає протокол TCP. Пересилання пакетів між комп'ютерами, які можуть мати різну архітектуру, використовувати різні операційні системи та входити до різних мереж, здійснюється на основі

протоколу IP.

Завдяки розбиттю даних на окремі пакети передавання їх мережею відбувається швидко та надійно і стає можливим навіть у випадку, коли частина мережі пошкоджена. У випадку виходу з ладу частини мережі маршрутизаторами буде зроблена спроба визначити новий маршрут для проходження пакета в обхід пошкодженої ділянки.

Мережевий протокол в комп'ютерних мережах — заснований на стандартах набору правил, що визначає принципи взаємодії комп'ютерів в мережі [1.3]. Протокол також задає загальні правила взаємодії різноманітних програм, мережевих вузлів чи систем і створює таким чином єдиний простір передачі. Хости (будь-який вузол мережі що відправляє або приймає дані через мережу називають хостом (host)) взаємодіють між собою. Для того, щоб прийняти і обробити відповідним чином повідомлення, їм необхідно знати як сформовані повідомлення і що вони означають. Прикладами використання різних форматів повідомлень в різних протоколах можуть бути встановлення з'єднання з віддаленою машиною, відправка повідомлень електронною поштою, передача файлів. Зрозуміло, що різні служби використовують різні формати повідомлень.

Мережевий протоко́л — набір правил, що визначає елементи у мережі. Протокол також задає загальні правила взаємодії різноманітних програм, мережевих вузлів чи систем і створює таким чином єдиний простір передачі [1.3]. Хости (будь-який вузол мережі що відправляє або приймає дані через мережу називають хостом (host)) взаємодіють між собою. Для того, щоб прийняти і обробити відповідним чином повідомлення, їм необхідно знати як сформовані повідомлення і що вони означають. Прикладами використання різних форматів повідомлень в різних протоколах можуть бути встановлення з'єднання з віддаленою машиною, відправка повідомлень електронною поштою, передача файлів. Зрозуміло, що різні служби використовують різні формати повідомлень.

Протокол описує:

1. Формат повідомлення, якому застосунки зобов'язані слідувати;

2. Спосіб обміну повідомленнями між комп'ютерами в контексті визначеної дії, як, наприклад, пересилка повідомлення по мережі.

Різні протоколи найчастіше описують лише різні сторони одного типу зв'язку й, узяті разом, утворюють стек протоколів. Назви «протокол» і «стек протоколів» також вказують на програмне забезпечення, яке реалізує протоколи.

Нові протоколи для Інтернету визначаються IETF, інші протоколи — IEEE або ISO. ITU-T займається телекомунікаційними протоколами та форматами.

Також дуже важливо розрізняти два схожі за назвою, але діаметрально протилежні за властивостями, терміни — маршрутизований протокол та протокол маршрутизації. Ще більша плутанина виникає з оригінальною назвою — routed&routing protocols [1.3].

Маршрутизований протокол — це будь-який мережний протокол, адреса мережевого рівня якого надає достатньо інформації для доставки пакету від одного вузла мережі до іншого на основі використовуваної схеми адресації. Такий протокол задає формати полів всередині пакету. Пакети зазвичай передаються від однієї кінцевої системи до іншої. Маршрутизований протокол використовує таблицю маршрутизації для пересилки пакетів [1.1-1.3].

Приклади маршрутизованих протоколів — Internet-протокол (IP), протокол міжмережевого пакетного обміну IPX тощо. Легше всього зрозуміти що таке маршрутизовані протоколи, якщо пам'ятати, що це протоколи передачі даних.

Протокол маршрутизації — такий протокол, який підтримує маршрутизовані протоколи і надає механізми обміну маршрутною інформацією. Повідомлення протоколу маршрутизації передаються між маршрутизаторами (роутерами). Протокол маршрутизації дозволяє роутерам

обмінюватись інформацією між собою для оновлення записів і підтримки таблиці маршрутизації.

Приклади протоколів маршрутизації: RIP, IGRP, EIGRP, OSPF. Легше зрозуміти, що таке протоколи маршрутизації, якщо пам'ятати, що це протоколи обміну маршрутною інформацією.

Для того, щоб протокол був маршрутизованим, він має включати механізми призначення як номера мережі, так і номера вузла для кожного пристрою в мережі. В деяких протоколах, як, наприклад, IPX необхідно визначати лише адресу мережі, оскільки як адресу пристрою ця технологія використовує фізичну адресу (MAC-адресу) пристрою. Інші протоколи, як IP-протокол, вимагають явного задання повної адреси і маски підмережі.

1.2 Загальний опис моделі OSI

Найпоширенішою системою класифікації мережних протоколів (і способів мережного зв'язку загалом) є, так звана, модель OSI (таб. 1.1), відповідно до якої протоколи поділяються на 7 рівнів за своїм призначенням — від фізичного (формування й розпізнавання електричних або інших сигналів) до прикладного (API застосунків для передачі інформації) [1.3].

Таблиця 1.1. Список мережевих протоколів

Модель OSI	
Дані	Рівень
Дані	**7. Прикладний** ***доступ до мережевих служб***
Дані	**6. Представлення** ***представлення і кодування даних***
Дані	**5. Сеансовий** ***керування сеансом зв'язку***
Блоки	**4. Транспортний** ***безпечне та надійне з'єднання «точка - точка»***
Пакети	**3. Мережевий** ***визначення маршруту та логічних адрес***
Кадри	**2. Канальний** ***MAC та LLC (фізична адресація)***
Біти	**1. Фізичний** ***кабель, сигнали, бінарна передача***

Розглянемо рівні та протоколи, що до них належать [1.3].

Рівень 1-ий (фізичний)

- ISDN (Integrated Services Digital Network) - Цифрова мережа Integrated Services
- PDH (Plesiochronous Digital Hierarchy) - Plesiochronous цифрова ієрархія
- RS-232, послідовний лінійний інтерфейс, який оригінально був розроблений для сполучення модемів та комп'ютерних терміналів.
- SDH (Synchronous Digital Hierarchy) - Синхронна цифрова ієрархія
- SONET (Synchronous Optical NETworking) - Синхронна оптична мережа
- Стандартні модемні протоколи / Протоколи серії ITU V, використовувані в з'єднаннях між аналоговими модемами по телефонній лінії.

Рівень 2-ий (канальний)

- ARP (Address Resolution Protocol)
- CDP (Cisco Discovery Protocol)
- DCAP (Data Link Switching Client Access Protocol)
- Ethernet
- FDDI (Fiber Distributed Data Interface)
- HDLC (High Level Data Link Control)
- L2F (Layer 2 Forwarding Protocol)
- L2TP (Layer 2 Tunneling Protocol)
- PPP (Point-to-Point Protocol) - Протокол Точка-точка
- PPTP (Point-to-Point Tunneling Protocol) - Тунельний протокол Точка-точка
- Fibre Channel (FC- fibre channel)
- Token ring
- VTP VLAN Trunking Protocol

Рівень 3 (мережний)

- XNS (Xerox network services)
- IP (Internet Protocol)
- IPv4
- IPv6
- ICMP (Internet Control Message Protocol) — міжмережевий протокол керуючих повідомлень
- ICMPv6
- IGMP (Internet Group Management Protocol) — протокол керування групами Інтернету
- Ipsec (скорочення від IP Security) — набір протоколів для забезпечення захисту даних

Рівень 4-ий (транспортний)

- SPX (Sequenced Packet Exchange) - Послідовність обміну пакетами
- TCP Протокол Керування Передачею
- UDP Протокол датаграм користувача
- SCTP (Stream Control Transmission Protocol) - Протокол Управління Потоковою Передачею
- RTP (Real-time Transport Protocol) - Транспортний протокол в реальному часі
- IL (Internet Link) Спочатку розроблявся як транспортний протокол для 9P

Рівень 5-ий (сеансовий)

- NFS (Network File System) - Мережева файлова система
- *ADSP* (AppleTalk Data Stream Protocol) - Протокол потоків даних AppleTalk
- *ASP* (AppleTalk Session Protocol) - Сеансовий протокол AppleTalk
- *H.245*, Call Control Protocol for Multimedia Communication
- *iSNS* (Internet Storage Name Service)

- *L2TP* (Layer 2 Tunneling Protocol)
- *NetBIOS* (Network Basic Input Output System)
- *PAP* (Password Authentication Protocol)
- *PPTP* (Point-to-Point Tunneling Protocol)
- *RPC* (Remote Procedure Call Protocol)
- *RTCP* (Real-time Transport Control Protocol)
- *SMPP* (Short Message Peer-to-Peer)
- *SCP* (Session Control Protocol)
- *SOCKS* - мережевий протокол SOCKS,
- *ZIP* (Zone Information Protocol)
- *SDP* (Sockets Direct Protocol)

Рівень 6-ий (представницький)

- Apple Filing Protocol
- ASTERIX
- DTLS
- External Data Representation
- MIME
- NetWare Core Protocol
- XML

Рівень 7-ий

- NTP - Мережевий протокол часу
- Gopher, a precursor of web search engines
- Finger, який надає інформацію про користувача
- NNTP - Мережевий протокол новин
- LDAP (Lightweight Directory Access Protocol)
- DHCP (Dynamic Host Configuration Protocol)
- IRC (Internet Relay Chat)
- Jabber - протокол миттєвих повідомлень
- WebDAV(Web Distributed Authoring and Versioning)

- DICT - Словниковий протокол
- BACnet Building Automation and Control Network protocol
- та захищені версії вищезазначеного (такі як HTTPS та інші.)

У комп'ютерних мережах використовують такі стеки протоколів:

- TCP/IP
- IPX/SPX
- NetBIOS/SMB

1.3. Стандарт фізичного рівня для асинхронного інтерфейсу RS-232

RS-232 (Recommended Standard 232, інша назва EIA232 [1.4]) - стандарт фізичного рівня для асинхронного інтерфейсу (UART). Пристрій, що підтримує цей стандарт, широко відоме як послідовний порт персональних комп'ютерів. Історично стандарт мав широке поширення в телекомунікаційному обладнанні. В даний час використовується для підключення до комп'ютерів широкого спектру обладнання, невибагливого до швидкості обміну, особливо при значній відстані його від комп'ютера і відхиленні умов застосування від стандартних. У комп'ютерах, зайнятих офісними і розважальними програмами, практично витіснений інтерфейсом USB.

RS-232 — стандарт інтерфейсу обміну даними між двома пристроями шляхом послідовної передачі даних (асинхронний зв'язок або синхронний зв'язок), знаходить використання у послідовних портах комп'ютерів та інших пристроях.

Офіційна назва: «Інтерфейс між кінцевим обладнанням обробки інформації і кінцевим обладнанням каналу передачі даних, що використовує послідовний обмін двійковими даними» *(Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange).*

Цей стандарт з'єднання устаткування було розроблено в 1969 році низкою

великих промислових корпорацій і опубліковано *Асоціацією електронної промисловості США* (*Electronic Industries Association — EIA*). Міжнародний союз електрозв'язку *ITU-T* використовує аналогічні рекомендації під назвами *V.24 й V.28*. В СРСР стандарт на подібний інтерфейс було прийнято під назвою *«Стык С2»* і описано в документі ГОСТ 18145-81.

Стандарт визначає номенклатуру ланцюгів та технічні вимоги до них (типи з'єднувачів, призначення їх контактів, робочі напруги і т. д.), що використовуються для з'єднання двох класів пристроїв:

Кінцеве обладнання обробки інформації (рос. оконечное оборудование данных, ООД; англ. data terminal equipment, DTE), наприклад, термінал чи послідовний порт COM1 персонального комп'ютера;

Кінцеве устаткування лінії зв'язку (рос. аппаратура передачи данных, АПД; англ. data circuit-terminating equipment, DCE), наприклад, модем.

Стандарт не визначає швидкості передачі. Загальноприйнята швидкість передачі для RS-232 — 9600 біт/сек на відстань до 15 м.

Широко використовується послідовний інтерфейс синхронної і асинхронної передачі даних, який визначається стандартом EIA RS-232-C і рекомендаціями V.24 CCITT. Спочатку створювався для зв'язку комп'ютера з терміналом. В даний час використовується в самих різних застосуваннях.

Розглянемо більш детально на прикладі інтерфейсу RS-232-C. Інтерфейс RS-232-C був розроблений для простого застосування, однозначно визначається по його назві "Інтерфейс між термінальним обладнанням і зв'язковим обладнанням з обміном по послідовному двійкового коду". Кожне слово в назві значиме, воно визначає інтерфейс між терміналом (DTE) і модемом (DCE) з передачі послідовних даних [1.5].

Асоціація електронної промисловості (EIA) розвиває стандарти з передачі даних. Стандарти EIA мають префікс "RS". "RS" означає рекомендований стандарт, але зараз стандарти просто позначаються як "EIA" стандарти. RS-232 був введений в 1962. Стандарт розвивався і в 1969 представлена третя редакція

(RS-232C). Четверта редакція була в 1987 (RS-232D, відома також під EIA-232D). RS-232 ідентичний стандартам МККТТ (CCITT) V.24 / V.28, X.20bis / X.21bis і ISO IS2110.

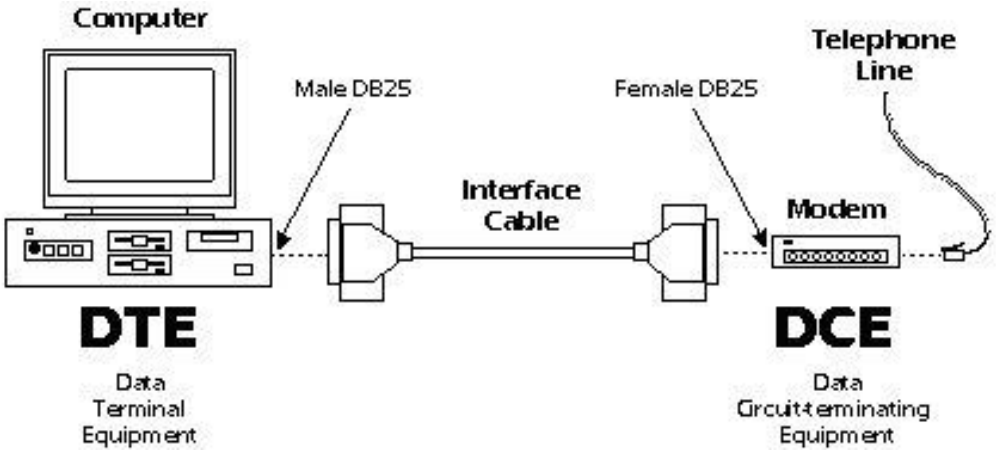


Рис. 1.1. Інтерфейс RS-232-C в елементарній мережі

У цій схемі модем DCE (Data Communication Equipment) є пристроєм - тобто обладнанням провайдера, яке визначає швидкість каналу, перетворює і передає дані від устаткування клієнта. А з боку клієнта передає ці дані DTE (Data Terminal Equipment) пристрій, який зазвичай є маршрутизатором або комп'ютером [1.5].

В RS-232 використовуються два рівні сигналів: логічні 1 і 0. Логічний 1 іноді позначають MARK, логічний 0 - SPACE. Логічної 1 відповідають негативні рівні напруги, а логічному 0 - позитивні. Відповідні значення напруг представлені в таблиці 1.2 і 1.3 та рис 1.2.

Таблиця 1.2. Рівні сигналів даних

Рівень	Передатчик	Приймач
Логический 0	От +5 В до +15 В	От +3 В до +25 В
Логический 1	от-5 В до -15 В	От -3 В до -25 В
Не определен	От -3 В до +3 В	

Таблиця 1.3. Рівні управляючих сигналів

Сигнал	На виході прист (Driver)	На вході пристр (Terminator)
"Off"	От -5 В до -15 В	от -3 В до -25 В
"On"	От 5 В до 15 В	от 3 В до 25 В

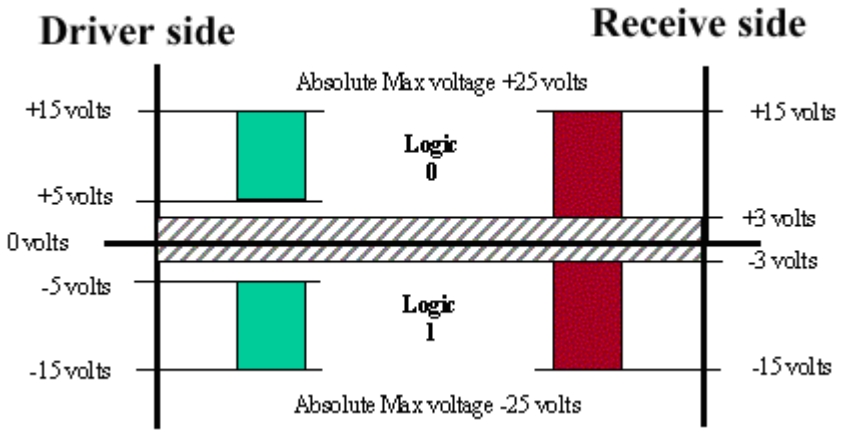


Рис. 1.2. Відповідність рівнів

Сигнали після проходження по кабелю ослабляються і спотворюються. Ослаблення зростає зі збільшенням довжини кабелю. Цей ефект сильно пов'язаний з електричною ємністю кабелю. За стандартом максимальна навантажувальна ємність становить 2500 пФ. Типова погонне ємність кабелю становить 130 пФ, тому максимальна довжина кабелю обмежена приблизно 17 м.

Перед з'єднанням двох комп'ютерів через RS-232, кожен з яких живиться від різних джерел рекомендується вирівняти напруги між їх сигнальними землями перед підключенням.

Інтерфейс RS-232-C з'єднує два пристрої. Лінія передачі першого пристрою з'єднується з лінією прийому другого і навпаки (повний дуплекс) Для управління з'єднаними пристроями використовується програмне підтвердження (введення в потік переданих даних відповідних керуючих символів). Можлива організація апаратного підтвердження шляхом організації додаткових RS-232 ліній для забезпечення функцій визначення статусу і управління.

Сучасні пристрої підтримують швидкість 115 кбіт/сек та більше (таб. 1.4). Існує в 8-, 9-, 25- і 31-контактних варіантах роз'ємів.

Таблиця 1.4. Характеристики RS-232-C

Стандарт	EIA RS-232-C, CCITT V.24
Швидкість передачіи	115 Кбіт/с (максимум)
Відстань передачі	15 м (максимум)
Характер сигнала	несиметричний по напрузі
Колькість драйверів	1
Колькість приймачів	1
Схема зєднання	повний дуплекс, від точки до точки

Також інтерфейс RS-232C призначений для підключення до комп'ютера стандартних зовнішніх пристроїв (принтера, сканера, модему, миші та ін.), А також для зв'язку комп'ютерів між собою. Основними перевагами використання RS-232C в порівнянні з Centronics є можливість передачі на значно більші відстані і набагато простіший з'єднувальний кабель. У той же час працювати з ним трохи складніше. Дані в RS-232C передаються в послідовному коді побайтно. Кожен байт обрамляється стартовим і степових битами. Дані можуть передаватися як в одну, так і в іншу сторону (двобічний режим) [1.5]. В таблиці 1.5 видно порядок обміну по інтерфейсу.

Комп'ютер має 25-контактний (DB25P) або 9-контактний (DB9P) роз'єм для підключення RS-232C. Призначення контактів роз'єму приведене на рис 1.3.

Призначення сигналів наступне:

FG - захисне заземлення (екран).

TxD - дані, що передаються комп'ютером в послідовному коді (логіка негативна).

RxD - дані, що приймаються комп'ютером в послідовному коді (логіка негативна).

RTS - сигнал запиту передачі. Активний в усі час передачі.

CTS - сигнал скидання (очищення) для передачі. Активний в усі час передачі. Каже про готовність приймача.

DSR - готовність даних. Використовується для завдання режиму модему.

SG - сигнальне заземлення, нульовий провід.

DCD - виявлення несучої даних (детектування сигналу).

DTR - готовність вихідних даних.

RI - індикатор виклику. Каже про прийом модемом сигналу виклику по телефонній мережі.

Таблиця 1.5. Порядок обміну по інтерфейсу RS-232C

Назва	Направленння	Опис	Контакт (25-контактний раз'єм)	Контакт (9-контактный раз'єм)
DCD	IN	Carrie Detect (Определение несущей)	8	1
RXD	IN	Receive Data (Принимаемые данные)	3	2
TXD	OUT	Transmit Data (Передаваемые данные)	2	3
DTR	OUT	Data Terminal Ready (Готовность терминала)	20	4
GND	-	System Ground (Корпус системы)	7	5
DSR	IN	Data Set Ready (Готовность данных)	6	6
RTS	OUT	Request to Send (Запрос на отправку)	4	7
CTS	IN	Clear to Send (Готовность приема)	5	8
RI	IN	Ring Indicator (Индикатор)	22	9

Найбільш часто використовуються трьох- або чотирьох-провідний зв'язок (для двонапрвленної передачі). Схема з'єднання для чотирьох лінії зв'язку показана на рисунку 1.3.

Для двохпровідної лінії зв'язку в разі тільки передачі з комп'ютера під зовнішній пристрій використовуються сигнали SG і TxD. Всі 10 сигналів інтерфейсу задіються тільки при з'єднанні комп'ютера з модемом [1.5].

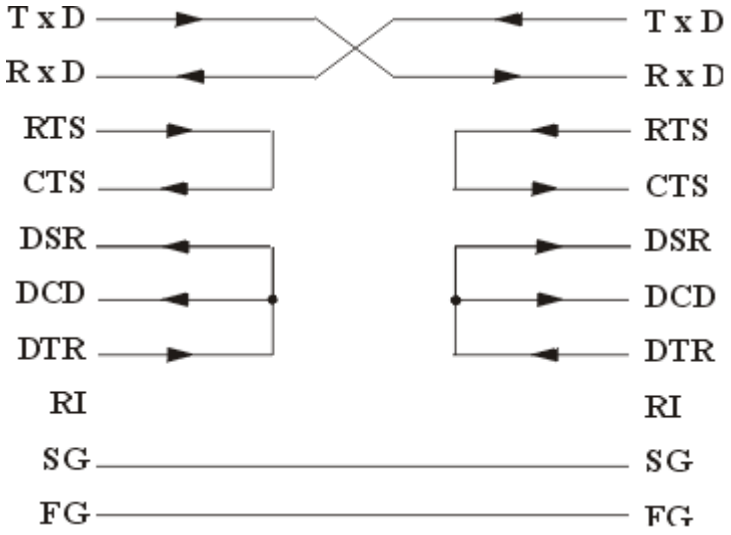


Рис. 1.3 Схема 4-провідної лінії зв'язку для RS-232C

У загальному випадку описує чотири інтерфейсні функції:

- визначення керуючих сигналів через інтерфейс;
- визначення формату даних користувача, переданих через інтерфейс;
- передачу тактових сигналів для синхронізації потоку даних;
- формування електричних характеристик інтерфейсу.

Пов'язані стандарти. Інші послідовні інтерфейси, схожі на RS-232:

- EIA-422 (RS-422)
- RS-423
- RS-449
- RS-485
- MIL-STD-188
- EIA-530
- EIA/TIA-561
- EIA/TIA-562
- TIA-574

Їх студент може вивчити на самостійній роботі.

1.4 Загальні поняття цифрової мережі інтегрального обслуговування

Цифрова мережа інтегрального обслуговування ЦМІО - (Integrated Services Digital Network - ISDN). Така мережа на основі уніфікованих засобів передачі, розподілу, обробки, зберігання та доставки інформації надасть абонентам (користувачам) широкий спектр інформаційного обслуговування [1.6]. Мережа ISDN будується, як правило, на основі телефонної цифрової мережі, та забезпечує передачу інформації між кінцевими пристроями в цифровому вигляді.

Доступ до послуг ISDN відбувається через визначений набір стандартизованих інтерфейсів.

В наш час отримали найбідьше розповсюдження два вида абонентського доступа до ресурсів мережі ISDN:

- базовий (Basic Rate Interface - BRI) со структурою 2B+D, де B=64 кбіт/с, D=16 кбіт/с, групова швидкість при цьому буде 144 кбіт/с, при наявності каналу синхронізації швидкість передачі в линії може бути рівною 160 кбіт/с або 192 кбіт/с,

- первинний-(Primary Rate Interface - PRI) со структурою 30B+D, де B=64 кбіт/с, D=64 кбіт/с, при цьому швидкість передачі з урахуванням сигналів синхронізації буде – 2048 кбіт/с.

Канали *B* є незалежними і можуть бути використані одночасно для різних з'єднань і представлень різних послуг. Канал *D*, в основному, призначені для передачі службової (управляючої) інформації між користувачами і комутаційною станцією. Крім того, по ньому можливо передавати пакети даних і сигнали телеметрії.

Абонентський доступ до ISDN вдбувається в точках з стандартизованими електричними і логічними характеристиками. Основними є інтерфейси R, S, T, U і V, які стандартизовані (крім точки R). На рис. 1.4 Функціональна схема організації абонентського доступу до ISDN

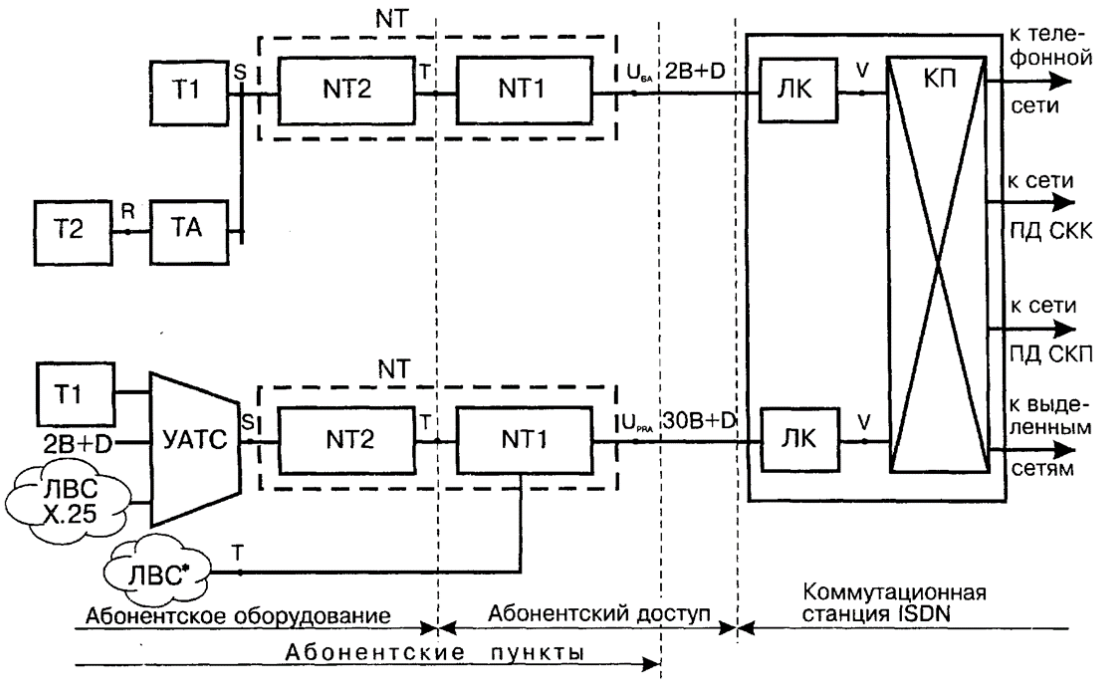


Рис. 1.4. Функціональна схема організації абонентського доступу до ISDN

ЛВС* - маршрутизатор цієї мережі має порт ISDN.

В зв'язку з тим, що в кореспондуючих АП можуть бути використані різні форми представлення інформації, на шостому, рівні відбувається представлення інформації, що передається в мережі на основі єдиної форми, яка на приймальному боці переводиться в ту форму, яка прийнята в даному АП. На прикладному рівні виконуються функції по взаємодії прикладних процесів [1.6].

Взаємодія процесів, що виконуються на одноіменних рівнях ВВС для ISDN (Рекомендація ITU-T I.320) відбувається шляхом пересилок повідомлень між відповідними рівнями. Повідомлення, передане з будь-якого рівня л об'єкта (АП) А, сприймається тільки на рівні п об'єкта (АП) Б. Нижче розташовані рівні майже до першого рівня ці повідомлення не зрозуміють, нижче розташовані рівні повинні бути для цього повідомлення "прозорими".

Правила взаємодії одноіменних рівнів прийнято називати протоколами.

Обладнання NT2 зазвичай виконує функції другого і третього рівнів моделі ВВС, однак може бути й "прозорим".

Інтерфейс U забезпечує взаємодію з абонентським лінійним комплектом, при цьому інтерфейс може бути як двохпровідним (у випадку базового

доступу), так й чотирьохпровідним (іноді у випадку первинного доступу) з використанням лінійних кодів 4B3T або 2B1Q.

В АП до однієї АЛ допускається підключення до 16 різних абонентських терміналів (реально до 8), включаючи аналоговий телефонний апарат (Т2), уніфікований термінал ISDN, персональну ЕОМ, локальну мережу з пакетною комутацією, для чого передбачений інтерфейс X.25, цифрову установчу АТС, яка надає послуги ISDN та ін [1.7].

В систему абонентського доступу до мережі ISDN входить, крім АЛ, мережне обладнання NT1. Наведена таблиця 1.6 типів каналів в інтерфейсі "користувач мережа".

Таблиця 1.6. Типи каналів в інтерфейсі "користувач мережа"

Визначення канала	B	H0	H11	H12	D
Швидкість цифрового потоку, Кбіт/с	64	384	1536 (стандарт ANSI)	1920 (стандарт ETSI)	16/64

Закріплення канала типу В за потоками користувачам може буим різним:

- за одним потоком зі швидкістю 64 Кбіт/с;

- за декількома потоками з одночасною або паралельною передачею по одному каналу.

Функції NT1:

- управління передачею потоку бітів по лінії;

- хронування (синхронізація);

- мультиплексування потоків бітів каналів B1, B2, D;

- контроль робочих характеристик і технічне обслуговування об'єктів в інтерфейсі "користувач мережа".

Функції NT2:

- обробка сигнальних повідомлень по протоколам рівнів ланцюга даних (2-го) та мережного (3-го);

- мультиплексування потоків на рівнях ланцюга даних та мережном;

- комутація каналів типу B;

- концентрація інформаційних потоків від групи терміналів для передачі по одній абонентської лінії;

- технічне обслуговування об'єктів в інтерфейсі "користувач мережа".

Функції терминалів типу TE1

Терминали TE1 повинні забезпечити реалізацію наступних функцій:

- доступа до сигнального каналу D;

- обміну з ISDN при базовому та первинному доступі;

- надання послуг зв'язку.

Доступ до сигнального канала D забезпечує можливість високошвидкісного та надійного обміну всіма необхідними повідомлень для отримання основних та додаткових послуг ISDN [1.3, 1.7]. Обмін користувача з ISDN при базовому доступі реалізується за допомогою протоколів фізичного (таблиця 1.1), ланцюга даних (таблиця 1.7) і мережного (таблиця 1.8) рівнів DSS1.

Таблиця 1.7. Протокол фізичного рівня DSS1

Функції	Опис (Рекомендації I.430, I.412)
Конфігурація проводки	Фізичне з'єднання TE з NT
Лінійний код	AMI (інверсія): TE - NT; 2B1Q: NT - LT
Структура цикла	Синхронізація бітів, октетів, циклів
Управління конфліктами в каналі D	Управління доступом до каналу D
Идентифікація каналів	Идентифікація каналів B і D
Технічна експлуатація	Дії по TE доступу «користувач-мережа» та TE
Електричні характеристики	Реалізація стика в з'єднаннях з пасивною шиною
Физичні характеристики	Розподіл контактів розетки для TE1

Таблиця 1.8. Протокол рівня ланцюга даних DSS1

Функції	Опис (Рекомендация Q.921)
Виключення неправдивих прапорів	Забезпечення інформаційної прозорості при переносі через мережу
Ідентифікація циклу	Розпізнавання і перевірка на достовірність усіх 48-бітових циклів
Встановлення режиму передачі	Посилання виклику в мережу для ініціалізації послуги
Знаходження помилок	Знаходження помилок в прийнятому повідомленні та в логіці обміну з мережею
Програмне управління	Забезпеення безперервної послідовності кадрів в з'єднанні
Відновлення	Виправлення помилок засобами рівня ланцюга даних (знаходження помилок в приймаємих кадрах і їх виправлення з використанням рішаючого зворотнього зв'язку) і інформування мережного протоколу про помилки, які не піддаються виправленню засобами ланцюга даних.
Можливості мовлення	Представлення ланцюгів даних мовлення, які можуть бути ідентифіковані за допомогою глобального (группового) ідентифікатора TEI.

Таблиця 1.9. Протокол мережного рівня DSS1

Функції	Опис (Рекомендация Q.931)
Повідомлення ідентифікації та обробки	Розпізнавання і перевірка правильності форматів повідомлень
Мітка виклику	Ідентифікація запросу виклика в інтерфейсі "користувач – мережа"
Інформація для обміну з мережею	Специфікація типів повідомлень (в фазах встановлення, передачі, роз'єднання)
Повідомлення управління викликом	Формування обов'язкових повідомлень для процедур управління основним викликом

1.5. Загальний огляд протоколів модемного зв'язку

Розглянемо поняття протоколів модемного зв'язку, що мають свої правила. Для того щоб модеми різних виробників могли працювати один з одним, існують протоколи зв'язку (точніше, протоколи модуляції), які є

правилами встановлення, підтримки і закінчення сеансу зв'язку між двома модемами [1.6-1.7].

Ці правила однозначно визначають метод перетворення цифрового сигналу в аналоговий (спосіб модуляції), швидкість передачі даних, послідовність передачі службових даних та корисної інформації. Всі етапи сеансу зв'язку жорстко регламентуються.

Всі протоколи, що регламентують ті чи інші аспекти функціонування модемів, можуть бути віднесені до двох великих груп:

- міжнародні;
- фірмові.

Протоколи міжнародного рівня розробляються під егідою Сектора стандартизації Міжнародного союзу електрозв'язку (ITU-T - International Telecommunications Union - Telecommunications) і приймаються їм в якості рекомендацій [1.1, 1.7].

Всі рекомендації ITU-T стосовно модемів відносяться до серії V.

Фірмові протоколи розробляються Окремими компаніями - виробниками модемів, з метою досягти успіху в конкурентній боротьбі.

При установці зв'язку між двома комп'ютерами модеми повинні надати один одному інформацію про швидкість зв'язку, коригування помилок і стиску даних. Для цього й існують протоколи - стандартизовані алгоритми роботи модему [1.3].

З функціональної точки зору модемні протоколи можуть бути розділені на наступні групи:

- протоколи, що визначають норми взаємодії модему з каналом зв'язку (V. 2, V. 25);
- протоколи, які регламентують з'єднання і алгоритми взаємодії модему і DTE (V. 10, V. 11, V. 24, V. 25, V. 25bis, V. 28);
- протоколи модуляції, що визначають основні характеристики модемів, призначених для комутованих і виділених телефонних каналів. До них

відносяться такі протоколи, як V. 17, V. 22, V. 32, V. 34, HST, ZyX і велика кількість інших;

- протоколи захисту від помилок (V. 41, V. 42, MNP1-MNP4);
- протоколи стиснення даних, що передаються, такі як MNP5, MNP7, V. 42bis;
- протоколи, що визначають процедури діагностики модемів, випробування і вимірювання параметрів каналів зв'язку (V. 51, V. 52, V. 53. V. 54, V. 56).
- протоколи узгодження параметрів зв'язку на етапі її встановлення, наприклад V.8.

Далі наведемо список стандартних, нестандартизованих протоколів та протоколи передачі факсимільних повідомлень [1.3-1.8].

1. Список стандартних протоколів

1.1. V.21

1.2. V.22

1.3. V.22bis

1.4. V.23

1.5. V.29

1.6. V.32

1.7. V.32bis

1.8. V.32terbo

1.9. V.34

1.10. V.34bis

1.11. V.42

1.12. V.42bis

1.13. V.44

1.14. V.70

1.1. 5V.80

1.16. V.90

1.17.V.92

1.18. V.110

1.19. V.120

1.20. V.150.1

2. Нестандартизовані протоколи

3. Протоколи передачі факсимільних повідомлень

3.1. V.17ter

3.2. V.17

3.3. V.27ter

3.4. V.29

Розглянемо список стандартних протоколів. Стандартні протоколи затверджені Міжнародним союзом електрозв'язку (ITU) [1.6-1.10].

V.21 - забезпечує швидкість передачі даних 300 біт / с в дуплексному режимі. Допускає також передачу факсимільних повідомлень.

V.22 - швидкість складає 1200 біт / с в напівдуплексному режимі.

V.22bis - друга редакція протоколу V.22, відрізняється збільшеною швидкістю 2400 біт / с і допускає дуплексний режим.

V.23 - асиметричний протокол 75 біт / с у висхідному (від користувача) каналі і 600 або 1200 біт / с - в низхідному. В кінці 1980-х - початку 1990-х років випускалися безліч нестандартних модемів, які використовували нестандартну, як правило - реалізовану програмно модуляцію та маркувалися як відповідні стандарту «V.23 mode 2». На практиці вони не були сумісні між собою, а реальна швидкість роботи швидшого каналу могла коливатися від 300 до 5600 біт / с. Найбільш відомим представником такого типу модемів були модеми Лександ. Модифікація протоколу V.23, що дозволяє змінювати висхідний і спадний канал місцями в процесі роботи, використовується у французькій комп'ютерної мережі Мінітель (Minitel). Також існує версія протоколу V.27, яка використовується як чотирипровідна версія з фазовою модуляцією 1800 МГц.

V.29 - асиметричний протокол 2400 / 2400-4800-7200-9600, що дозволяє

перемикати напрямок, в якому працює більше швидкісний канал, в процесі роботи. Є стандартним для факсів, але в модемах великого поширення не отримав у зв'язку з більш низькою завадостійкістю, ніж V.32, і низкою проблем з патентами.

V.32 - двобічний режим, швидкість 4800 і 9600 біт / с, допускає автоматичне налаштування швидкості передачі.

V.32 bis - розширення V.32 до швидкості 14400 біт / с.

V.32 terbo - розширення V.32 до швидкості 19200 біт / с, а у USR Courier - до 21600 біт / с.

V.34 - двобічний протокол, максимальна швидкість 28800 біт / с. Може також підтримувати 24000 і 19200 біт / с.

V.34 bis - інша назва - V.34 +. Максимальна швидкість 33600 біт / с. Знижені швидкості: 31200, 24000 і 19200 біт / с.

V.42 - протокол виявлення і корекції помилок для передачі даних з високими швидкостями.

V.42 bis - протокол стиснення даних. Допускає перемикання з режиму стиснення в прозорий режим і назад, причому, незалежно для кожного напрямку.

V.44 - протокол стиснення даних.

V.70 - забезпечує одночасну передачу голосу і даних.

V.80 - протокол відеозв'язку. Забезпечує швидкість передачі відео до 10-15 кадрів в секунду.

V.90 - двобічний асиметричний високошвидкісний протокол передачі. Швидкість в прямому напрямку досягає 56000 біт / с, а в зворотному – 33600 біт / с.

V.92 - найсучасніший протокол. Швидкість в прямому напрямку 56000 біт / с, а в зворотному - 48600 біт / с.

V.110 - протокол для мереж ISDN, що дозволяє також встановлювати з'єднання з модемами на аналогових лініях. Швидкість в синхронному режимі

може досягати 64000 біт / с.

V.120 - альтернатива протоколу V.110 з використанням LAPD.

V.150.1 - протокол модемного зв'язку поверх IP-телефонії.

Крім протоколів, затверджених ITU, існує безліч інших, які були розроблені виробниками обладнання або ж прийняті в якійсь країні [1.10, 1.11].

Bell 103 і Bell 212A - застосовувалися в США до прийняття стандартів ITU-T V.21 і V.22, відповідно.

Крім протоколів, затверджених ITU, існує безліч інших, які були розроблені виробниками обладнання або ж прийняті в якійсь країні.

Bell 103 і Bell 212A - застосовувалися в США до прийняття стандартів ITU-T V.21 і V.22, відповідно

Bell 202 - ще один ранній протокол модемного зв'язку, схожий на V.23

HST - фірмовий протокол компанії U.S.Robotics, доступний тільки в модемах сімейства Courier і серверах доступу. Особливістю протоколу є адаптивний дуплекс - детектування потоку даних з подальшим погодженням більшої швидкості від сторони, яка передає до приймаючої. Сучасна швидкість - 450/16800 біт / с, в більш ранніх версіях - 300/14400 біт / с і - 300/9600 біт / с. Видатна стійкість - одна з переваг протоколу, що дозволяє передавати дані на високій швидкості через такі лінії, де використання протоколів сімейства ITU-T аналогічної швидкості практично позбавлене сенсу.

V.FC або V.fast - протокол, розроблений компанією Rockwell. Підтримує швидкість зв'язку від 14400 до 28800 біт / с. Крім модемів Rockwell, зустрічався і в старих модемах USRobotics нарівні з V.34.

K56flex і x2 - одночасно розроблені компаніями Lucent і Rockwell, з одного боку, і U.S. Robotics, з іншого. Підтримують швидкість зв'язку до 56000 біт / с, але взаємно несумісні. Використання було актуально до прийняття стандарту ITU-T V.90, в даний час практичного сенсу в застосуванні не мають.

PEP (Packet Ensemble Protocol) - фірмовий протокол компанії Telebit з адаптивним дуплексом, швидкість до 18000 біт / с в ідеальних умовах, і 23000

біт / с для TurboPEP, більш нової версії. Найвища стійкість досягається розподілом частотного спектра ТЧ на 512 каналів, з передачею даних тільки по тим з них, які дозволяють приймати 6-бітові пакети даних без втрат. Незважаючи на банкрутство Telebit і поява високошвидкісних протоколів V.34 і V.90, модеми Telebit досі можуть бути затребуваними завдяки наявності PEP / TurboPEP для передачі даних при з'єднанні точка-точка.

ZYX - фірмовий протокол компанії ZyXEL, вперше застосований в модемах сімейства U-1496. Двобічний, швидкість передачі даних 16800 біт / с і 19200 біт / с (тільки для моделей з індексом «+»). Після прийняття стандарту V.34 практичне застосування позбавлене сенсу.

ATMT (Адаптивна ТрансМультіплексорная Технологія) - фірмовий протокол компанії Proxyma Communications для модемів серії CyBear T34 (також відомих під маркою Zelax T34). Напівдуплексний, швидкість передачі - до 30000 біт / с (адаптивно з кроком 75 біт / с) в режимі комутованого каналу; на стандартному діапазоні ТЧ 300 ... 3400 гц - 24000 біт / с. У разі асиметрії потоків даних пропускна здатність модему перерозподіляється між напрямками. Тип модуляції - багатоканальна (до 60 каналів по 75 Гц) АФМ з нелінійним кодуванням і числом точок сигнального простору до 256. Призначений для роботи на лініях з високим рівнем шумів і імпульсних перешкод, каналах з обмеженим спектром частот.

Наостаннє в підрозділі розглянемо список протоколів передачі факсимільних повідомлень.

V.17ter - швидкість до 19200 біт / с, зустрічається не у всіх модемах.

V.17 - швидкість до 14400 біт / с, використовується в сучасних модемах, але, оскільки більшість факсимільних апаратів розраховано на 9600 біт / с, на практиці переваг не дає.

V.27ter - швидкість 2400-4800 біт / с, зустрічається в старих модемах, але його підтримують і багато сучасних модеми.

V.29 - швидкість 7200-9600 біт / с, підтримують всі сучасні модеми.

ЛІТЕРАТУРА

1.1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципи, технологии, протоколы: Учебник для вузов. – СПб.: Питер, 2001. – 672 с.

1.2. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципи, технологии, протоколы: Учебник для вузов. 3-е изд. – СПб.: Питер, 2006.– 958 с.

1.3. Компьютерные сети. 4-е изд./ И. Таненбаум. – СПб.: Питер, 2003. – 992 с.

1.4. Knuth, Donald (2008), Negafibonacci Numbers and the Hyperbolic Plane, Paper presented at the annual meeting of the Mathematical Association of America, San Jose, California.

1.5. Knuth, Donald (2009), The Art of Computer Programming, Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams, ISBN 0-321-58050-8. In the pre-publication draft of section 7.1.3 see in particular pp. 36–39.

1.6. Margenstern, Maurice (2008), Cellular Automata in Hyperbolic Spaces, vol. 2, Advances in unconventional computing and cellular automata, Archives contemporaines, c. 79, ISBN 9782914610834.

1.7. Digital Signal Compression: Principles and Practice (By William A. Pearlman, Amir Said, 2011, ISBN 9780521899826), Chapter 4 "Entropy coding techniques" pp41-76

1.8. Storer, James A. Data Compression via Textual Substitution (неопр.) // Journal of the ACM. — 1982. — October (т. 29, № 4). — С. 928—951. — DOI:10.1145/322344.322346.

1.9. Ryabko, B. Ya. Data compression by means of a «book stack», Problems of Information Transmission, 1980, v. 16: (4), pp. 265–269.

1.10. Ryabko, B. Ya.; Horspool, R. Nigel; Cormack, Gordon V. Comments to: «A locally adaptive data compression scheme» by J. L. Bentley, D. D. Sleator, R. E. Tarjan and V. K. Wei. Comm. ACM 30 (1987), no. 9, 792—794.

1.11. Ilya Grigorik. Google Fonts recently switched to using new Zopfli

compression algorithm. Google+

1.12. J.G. Cleary, and I.H. Witten, Data compression using adaptive coding and partial string matching (недоступная ссылка), IEEE Transactions on Communications, Vol. 32 (4), pp. 396–402, April 1984.

1.13. A. Moffat, Implementing the PPM data compression scheme, IEEE Transactions on Communications, Vol. 38 (11), pp. 1917–1921, November 1990.

1.14. J.G. Cleary, W.J. Teahan, and I.H. Witten, Unbounded length contexts for PPM, In J.A. Storer and M. Cohn, editors, Proceedings DCC-95, IEEE Computer Society Press, 1995.

1.15. Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео Диалог-МИФИ, 2002 г., 384 с. ISBN 5-86404-170-X. 3000 экз.

1.16. G. Nigel N. Martin, Range encoding: An algorithm for removing redundancy from a digitized message, Video & Data Recording Conference, Southampton, UK, July 24-27, 1979.

1.17. «Source coding algorithms for fast data compression» Richard Clark Pasco, Stanford, CA 1976

2 ПРОТОКОЛИ КАНАЛЬНОГО РІВНЯ

Канальний рівень відповідає за доставку кадрів (frame) між пристроями, підключеними до одного мережевого сегменту. Кадри канального рівня не перетинають кордонів мережевого сегмента.Кадри передаються послідовно з обробкою кадрів підтвердження, які надсилаються назад отримувачем. Функції міжмережевий маршрутизації і глобальної адресації здійснюються на більш високих рівнях моделі OSI, що дозволяє протоколам канального рівня зосередитися на локальній доставці і адресації [2.1].

Заголовок кадру містить апаратні адреси відправника і одержувача, що дозволяє визначити, який пристрій відправив кадр і який пристрій має отримати і обробити його. На відміну від ієрархічних і маршрутизованих адрес, апаратні адреси однорівневі. Це означає, що ніяка частина адреси не може вказувати на належність до будь-якої логічної чи фізичної групі.

Коли пристрої намагаються використовувати середу одночасно, виникають колізії кадрів. Протоколи канального рівня виявляють такі випадки і забезпечують механізми для зменшення їх кількості або ж їх запобігання.

Багато протоколи канального рівня не мають підтвердження про прийом кадру, деякі протоколи навіть не мають контрольної суми для перевірки цілісності кадру. У таких випадках протоколи більш високого рівня повинні забезпечувати управління потоком даних, контроль помилок, підтвердження доставки та ретрансляції втрачених даних.

На цьому рівні працюють комутатори, мости.

Канальний рівень (datalinklayer) забезпечує прозорість з'єднання для мережевого рівня. Для цього він пропонує йому такі послуги:

• встановлення логічного з'єднання між взаємодіючими вузлами;

• узгодження в рамках з'єднання швидкостей передавача і приймача інформації;

• забезпечення надійної передачі, виявлення і корекція помилок.

Для вирішення цих завдань канальний рівень формує з пакетів власні протокольні одиниці даних - кадри, що складаються з поля даних і заголовка. Канальний рівень поміщає пакет в поле даних одного або декількох кадрів і заповнює власної службовою інформацією заголовок кадру.

У мережах, побудованих на основі розділяється середовища, фізичний рівень виконує ще одну функцію - перевіряє доступність середовища, що розділяється. Цю функцію іноді виділяють в окремий підрівень управління доступом до середовища (Medium Access Control, MAC).

Розглянемо більш докладно роботу канального рівня, починаючи з моменту, коли мережевий рівень відправника передає канального рівня пакет, а також вказівку, якого вузлу його передати. Для вирішення цього завдання канальний рівень створює кадр, який має поле даних і заголовок. Канальний рівень поміщає (інкапсулює) пакет в поле даних кадру і заповнює відповідної службової інформацією заголовок кадру.

Найважливішою інформацією заголовка кадру є адреса призначення, на підставі якого комутатори мережі будуть просувати пакет.

Одним із завдань канального рівня є виявлення і корекція помилок. Канальний рівень може забезпечити надійність передачі, наприклад, шляхом фіксування кордонів кадру, вміщуючи спеціальну послідовність бітів в його початок і кінець, а потім додаючи до кадру контрольну суму. Контрольна сума обчислюється за певним алгоритмом як функція від всіх байтів кадру. На стороні одержувача канальний рівень групує біти, що поступають з фізичного рівня, в кадри, знову обчислює контрольну суму отриманих даних і порівнює результат з контрольною сумою, переданої в кадрі [2.2.].

Якщо вони збігаються, кадр вважається правильним. Якщо ж контрольні суми не збігаються, фіксується помилка.

У функції канального рівня входить не тільки виявлення помилок, але і їх виправлення за рахунок повторної передачі пошкоджених кадрів. Однак ця

функція не є обов'язковою і в деяких реалізаціях канального рівня вона відсутня, наприклад, в Ethernet.

Перш ніж переправити кадр фізичного рівня для безпосередньої передачі даних в мережу, канального рівня може знадобитися вирішити ще одну важливу задачу. Якщо в мережі використовується поділюване середовище, то перш ніж фізичний рівень начнетпередавать дані, канальний рівень повинен перевірити доступність середовища. Як уже зазначалося, функції перевірки доступності середовища, що розділяється іноді виділяють в окремий підрівень управління доступом до середовища (підрівень MAC).

Якщо поділюване середовище звільнилася (коли вона не використовується, то така перевірка, звичайно, пропускається), кадр передається засобами фізичного рівня в мережу, проходить по каналу зв'язку і надходить у вигляді послідовності бітів в розпорядження фізичного рівня вузла призначення. Цей рівень в свою чергу передає отримані біти «наверх» канального рівня свого вузла.

Протокол канального рівня зазвичай працює в межах мережі, що є однією зі складових більшої складовою мережі, об'єднаної протоколами мережевого рівня. Адреси, з якими працює протокол канального рівня, використовуються для доставки кадрів тільки в межах цієї мережі, а для переміщення пакетів між мережами застосовуються вже адреси наступного, мережевого, рівня.

У локальних мережах канальний рівень підтримує вельми могутній і закінчений набір функцій з пересилання повідомлень між вузлами мережі. У деяких випадках протоколи канального рівня локальних мереж виявляються самодостатніми транспортними засобами і можуть допускати роботу безпосередньо поверх себе протоколів прикладного рівня або додатків без залучення коштів мережевого і транспортного рівнів. Проте для якісної передачі повідомлень в мережах з довільною топологією функцій канального рівня виявляється недостатньо [2.1].

2.1 ARP - протокол визначення адрес

ARP (Address Resolution Protocol — протокол визначення адрес) — комунікаційний протокол, призначений для перетворення IP-адрес (адрес мережевого рівня) в MAC-адреси (адреси канального рівня) в мережах TCP/IP. Він визначений в RFC 826 [2.3].

Перетворення виконується лише для тих IP-пакетів, які відправляються, оскільки лише в момент відправлення створюються заголовки IP та Ethernet.

Перетворення адрес виконується шляхом пошуку за таблицею.

Ця таблиця називається ARP-таблицею, зберігається у пам'яті й містить рядки для кожного вузла мережі.

В двох стовпчиках містяться IP- та Ethernet-адреси.

Якщо потрібно перетворити IP-адресу в Ethernet-адресу, то відбувається пошук запису з відповідною IP-адресою (таб. 2.1).

Таблиця 2.1. Приклад ARP-таблиці

IP-адресу	**Ethernet-адресу**
223.1.2.1	08:00:39:00:2F:C3
223.1.2.2	08:00:5A:21:A7:22
223.1.2.3	08:00:10:99:AC:54

ARP-таблиця необхідна через те, що IP-адреси та Ethernet-адреси вибираються незалежно, і немає жодного алгоритму для перетворення однієї в іншу. IP-адресу вибирає менеджер мережі з урахуванням розташування машини у мережі Інтернет [2.2].

Якщо машину переміщують до іншої частини мережі Інтернет, то її IP-адреса повинна бути змінена. Ethernet-адресу вибирає виробник мережного інтерфейсного обладнання з виділеного для нього згідно з ліцензією адресного простору.

Якщо в машини змінюється мережний адаптер, то змінюється й Ethernet-адреса.

Порядок перетворення адрес відбувається наступним чином.

У ході звичайної роботи мережева програма відправляє прикладне повідомлення, користуючись транспортними послугами TCP [2.1].

Модуль TCP надсилає відповідне транспортне повідомлення через модуль IP. У результаті складається IP-пакет, який має передаватись драйверові Ethernet. IP-адреса місця призначення відома прикладній програмі, модулеві TCP та модулеві IP. Необхідно на її основі знайти Ethernet-адресу місця призначення.

Для пошуку відповідної Ethernet-адреси використовується ARP-таблиця.

Запити та відповіді протоколу ARP описано нижче.

ARP-таблиця заповнюється автоматично модулем ARP по мірі необхідності. Коли за допомогою існуючої ARP-таблиці не вдається перетворити IP-адресу, то відбувається таке:

По мережі передається широкомовний ARP-запит.

Вихідний IP-пакет ставиться в чергу.

Кожний мережний адаптер приймає широкомовні передачі. Усі драйвери Ethernet перевіряють поле типу в прийнятому Ethernet-кадрі й передають ARP-пакети модулю ARP. ARP-запит можна інтерпретувати так: «Якщо ваша IP-адреса збігається із зазначеною, то повідомте мені вашу Ethernet-адресу». Пакет ARP-запиту виглядає приблизно так, як показано в таб.2.2.

Таблиця 2.2. Приклад ARP-запиту

IP-адреса відправника	223.1.2.1
Ethernet-адреса відправника	08:00:5A:21:A7:22
Шукана IP-адреса	223.1.2.3
Шукана Ethernet-адреса	порожньо<>

Кожний модуль ARP перевіряє поле шуканої IP-адреси в отриманому ARP-пакеті і, якщо адреса збігається з його власною IP-адресою, то посилає відповідь прямо за Ethernet-адресою відправника запиту. Пакет з ARP-

відповіддю виглядає приблизно так (таб. 2.3):

Таблиця 2.3. Приклад ARP-відповіді

IP-адреса відправника	223.1.2.3
Ethernet-адреса відправника	08:01:2A:2B:A7:21
IP-адресаавтора запиту	223.1.2.1
Ethernet-адреса автора запиту	08:00:5A:21:A7:22

Цю відповідь одержує машина, що зробила ARP-запит [2.3].

Драйвер цієї машини перевіряє поле типу в Ethernet-кадрі й передає ARP-пакет модулю ARP.

Модуль ARP аналізує ARP-пакет і додає запис у свою ARP-таблицю.

Якщо в мережі немає машини із шуканою IP-адресою, то ARP-відповіді не буде й не буде запису в ARP-таблиці.

Протокол IP буде знищувати IP-пакети, що направляються по цій адресі.

Протоколи верхнього рівня не можуть відрізнити випадок пошкодження мережі Ethernet від випадку відсутності машини із шуканою IP-адресою.

2.2 Мережевий протокол управління каналом передачі даних HDLC

High-Level Data Link Control (HDLC) — біт-орієнтований кодопрозорий мережевий протокол управління каналом передачі даних канального рівня мережевої моделі OSI, розроблений ISO [2.4].

Поточним стандартом для HDLC є ISO 13239.

HDLC може бути використаний у з'єднаннях точка-багатоточка, але в наш час в основному використовується у з'єднаннях точка-точка з використанням асинхронного збалансованого режиму (ABM).

HDLC був розроблений на основі протоколу SDLC фірми IBM. Його несильно змінені дочірні протоколи — LAPB, LAPM, LAPF, LAPD були вбудовані ITU відповідно в стеки протоколів X.25, V.42, Frame Relay, ISDN.

Також HDLC був базою при розробці кадрових механізмів у протоколі PPP, який широко використовується в Інтернеті.

Є декілька типів станцій. Первинна (ведуча) станція (Primary terminal) відповідальна за управління каналом і відновлення його працездатності. Вона виробляє кадри команд. У з'єднаннях точка-багатоточка підтримує окремі зв'язку з кожною з вторинних станцій.

Вторинна (ведена) станція (Secondary terminal) працює під контролем ведучої, відповідаючи на її команди. Підтримує тільки 1 сеанс зв'язку.

Комбінована станція (Combined terminal) поєднує в собі функції як провідної, так і веденої станцій. Виробляє і команди і відповіді. Тільки з'єднання точка-точка [2.2, 2.5].

Кожна із станцій в кожен момент часу знаходиться в одному з 3 логічних станів:

Стан логічного завершення (LDS - Logical Disconnect State)

Якщо вторинна станція знаходиться в режимі нормального закінчення (NDM), то вона може приймати кадри тільки після отримання явного дозволу від первинної. Якщо ж в асинхронному режимі роз'єднання (ADM), то вторинна станція може самовільно ініціювати передачу.

Стан ініціалізації (IS - Initialization State). Використовується для передачі управління на віддалену комбіновану станцію і для обміну параметрами між віддаленими станціями.

Стан передачі інформації (ITS - Information Transfer State). Всім станціям дозволено вести передачу і приймати інформацію. Станції можуть перебувати в режимах NRM, ARM, ABM.

HDLC підтримує три режими логічного з'єднання, що відрізняються ролями взаємодіючих пристроїв.

Режим нормального відповіді (NRM - Normal Response Mode) вимагає ініціації передачі у вигляді явного дозволу на передачу від первинної станції. Після використання каналу вторинної станцією (відповіді на команду

первинної), для продовження передачі вона зобов'язана чекати іншого дозволу. Для вибору права на передачу первинна станція проводить кругової опитування вторинних. В основному використовується в з'єднаннях точка-багатоточка.

Режим асинхронного відповіді (ARM - Asynchronous Response Mode) дає можливість вторинної станції самої ініціювати передачу. В основному використовується в сполуках типу кільце і багатоточкових з незмінною ланцюжком опитування, так як в цих з'єднаннях одна вторинна станція може отримати дозвіл від передачі від іншої вторинної і у відповідь почати передачу. Тобто дозвіл на передачу передається за типом маркера (token). За первинною станцією зберігаються обов'язки з ініціалізації лінії, визначення помилок передачі і логічному роз'єднання. Дозволяє зменшити накладні витрати, пов'язані з початком передачі.

Асинхронний збалансований режим (ABM - Asynchronous Balanced Mode) використовується комбінованими станціями [2.3]. Передача може бути ініційована з будь-якого боку, може відбуватися в повному дуплексі.

У режимі ABM обидва пристрої рівноправні і обмінюються кадрами, які діляться на кадри-команди і кадри-відповіді. Для забезпечення сумісності між станціями, які можуть змінювати свій статус (тип), в протоколі HDLC передбачено 3 конфігурації каналу:

Незбалансована конфігурація (UN - Unbalanced Normal) забезпечує роботу 1 первинної і однієї або декількох вторинних станцій в (симплексному) напівдуплексномуі повнодуплексному режима, з комутованим або некомутованими каналом.

Симетрична конфігурація (UA - Unbalanced Asynchronous) забезпечує взаємодію двох двоточкових незбалансованих станцій. Використовується 1 канал передачі, в який мультиплексуються і команди і відповіді. У наш час не використовується.

Збалансована конфігурація (BA - Balanced Asynchronous) складається з 2 комбінованих станцій. Передача в (симплексному) напівдуплексному і

повнодуплексному режимах, з комутованим або некомутованими каналом. Кожна станція несе однакову відповідальність за управління каналом [2.3].

Кадри HDLC можна передавати, використовуючи синхронні і асинхронні з'єднання. У самих з'єднаннях немає механізмів визначення початку і кінця кадру, для цих цілей використовується унікальна в межах протоколу послідовність прапорців (FD - Frame Delimiter) '01111110 '(0x7E в шістнадцятковому представленні), що поміщається в початок і кінець кожного кадру. Унікальність прапора гарантується використанням бітстафінга у синхронних з'єднаннях і байтстафінга в асинхронних. Бітстафінг - вставка бітів, тут - біту 0 після 5 поспіль йдуть бітів 1. Бітстафінг працює тільки під час передачі інформаційного поля (поля даних) кадру. Якщо передавач виявляє, що передано підряд п'ять одиниць, то він автоматично вставляє додатковий нуль в послідовність переданих бітів (навіть якщо після цих п'яти одиниць і так йде нуль). Тому послідовність 01111110 ніколи не з'явиться на полі даних кадру.

Аналогічна схема працює в приймальнику і виконує зворотну функцію. Коли після п'яти одиниць виявляється нуль, він автоматично видаляється з поля даних кадру. У байтстафінгу використовується escape-послідовність, тут - '01111101 '(0x7D в шістнадцятковому представленні), тобто байт FD (0x7E) в середині кадру замінюється послідовністю байтів (0x7D, 0x5E), а байт (0x7D) - послідовністю байтів (0x7D, 0x5D).

Під час простою середовища передачі при синхронному з'єднанні FD постійно передається по каналу для підтримки бітової синхронізації. Може мати місце поєднання останнього біта 0 одного прапора та початкової біта 0 наступного. Час простою також називається міжкадрових тимчасовим заповненням. Структура кадру HDLC, включаючи прапори FD (таб. 2.4):

Таблиця 2.4. Структура кадру HDLC

Прапор	Адреса	Управляюче поле	Інформаційне поле	FCS	Прапор
8 біт	8 біт	8 або 16 біт	0 або більше біт, кратно 8	16 біт	8 біт

Прапори FD — відкривальні і закривальні прапори, що являють собою коди 01111110, обрамляють HDLC-кадр, дозволяючи приймачу визначити початок і кінець кадру. Завдяки цим прапори в межах HDLC-кадрі відсутнє поле довжини кадру. Іноді прапор кінця одного кадру може (але не обов'язково) бути початковим прапором наступного кадру [2.3, 2.4].

Адреса виконує свою звичайну функцію ідентифікації одного з декількох можливих пристроїв тільки в конфігураціях точка-багатоточка. У двохточечній конфігурації адресу HDLC використовується для позначення напрямку передачі - з мережі до пристрою користувача (10000000) або навпаки (11 млн).

Поле, що управляє займає 1 або 2 байти. Його структура залежить від типу переданого кадру. Тип кадру визначається першими бітами керувального поля: 0 - інформаційний, 01 - керувальний, 11 - ненумерований тип. У структуру керувального поля кадрів усіх типів входить біт P / F, він по-різному використовується в кадрах-командах і кадрах-відповідях. Наприклад, станція-приймач при отриманні від станції-передавача кадру-команди з встановленим бітом P негайно повинна відповісти керувальним кадром-відповіддю, встановивши біт F.

Інформаційне поле призначене для передачі по мережі пакетів протоколу вищого рівня - мережевих протоколів IP, IPX, AppleTalk, DECnet, в окремих випадках - прикладних протоколів, коли ті викладають свої повідомлення безпосередньо в кадри канального рівня . Інформаційне поле може бути відсутнім в керувальних кадрах і деяких ненумерованих кадрах.

Поле FCS (Frame Check Sequence) — контрольна послідовність, необхідна для виявлення помилок передачі [2.3, 2.5]. Її обчислення в основному проводиться методом циклічного кодування з виробляють поліномом X16 + X12 + X5 +1 (CRC-16), відповідно до Рекомендації CCITT V.41. Це дозволяє виявляти всілякі кортежі помилок довжиною до 16 біт викликаються одиночної помилкою, а також 99,9984% всіляких більш довгих кортежів помилок. FCS складається по полях: Адреса, Поле, що управляє, Інформаційне поле. У

рідкісних випадках використовуються інші методи циклічного кодування. Після прорахунку FCS на стороні приймача він відповідає позитивною або негативною квитанцією. Повтор кадру передавальною стороною виконується по приходу негативною квитанції або після закінчення тайм-ауту.

Є різні типи кадрів. I-кадри (інформаційні кадри, кадри даних).

Призначені для передачі даних користувача [2.5]. У процесі передачі інформаційних блоків здійснюється їх нумерація відповідно до алгоритму ковзного вікна. Після встановлення з'єднання дані і позитивні квитанції починають передаватися в інформаційних кадрах. Логічний канал HDLC є дуплексним, так що інформаційні кадри, а значить, і позитивні квитанції можуть передаватися в обох напрямках. Якщо ж потоку інформаційних кадрів у зворотному напрямку немає або ж потрібно передати негативну квитанцію, то використовуються кадри, що управляють. При роботі HDLC для забезпечення надійності передачі використовується ковзне вікно розміром в 7 кадрів (при розмірі керувального поля 1 байт) або 127 (при розмірі керувального поля 2 байти). Для підтримки алгоритму вікна в інформаційних кадрах станції-відправника відводиться 2 поля:

N (S) — номер відправляється кадру;

N (R) — номер кадру, який станція очікує отримати від свого партнера по діалогу.

Припустимо для визначеності, що станція А відправила станції В інформаційний кадр з деякими значеннями NA (S) і NA (R). Якщо у відповідь на цей кадр приходить кадр від станції В, в якому номер посланого цією станцією кадру NB (S) збігається з номером очікуваного станцією А кадру NA (R), то передача вважається коректною. Якщо станція А приймає кадр-відповідь, в якому номер відправленого кадру NB (S) не дорівнює номером очікуваного NA (R), то станція А цей кадр відкидає і посилає негативну квитанцію REJ (відмова) з номером NA (R). Прийнявши негативну квитанцію, станція У зобов'язана повторити передачу кадру з номером NA (R), а також всіх

кадрів з великими номерами, які вона вже встигла відіслати, користуючись механізмом ковзаючого вікна.

Інші послідовні інтерфейси, схожі на RS-232:

I-кадри також містять біт опитування / відповідь P / F (poll / final). У режимі NRM провідний термінал використовує біт P для опитування, ведений - біт F в останньому I-кадрі відповіді. У режимах ARM і ABM біти P / F використовуються для форсування відповіді. В таблиці 2.5 приклад виду кадру.

Таблиця 2.5. Вид I-кадра

Команда / Відповідь	**Опис**	**Формат упр. поля 8 ... 7 ... 6 ... 5 ... 4 ... 3 ... 2 ... 1 ... ..**
C / R	Дані користувача	.- N (R) - ... P / F ... ..- N (S) -.. 0

S-кадри (керувальні) використовуються для контролю потоку помилок передачі. У керувальних кадрах передаються команди і відповіді в контексті встановленого логічного з'єднання, в тому числі запити на повторну передачу перекручених інформаційних блоків [2.3]:

Готовий до Прийому(RR) - використовується як позитивна квитанція (до N (r) -1).

Провідна станція може зробити опитування, встановивши біт P.

Ведена станція на опитування може відповісти кадром з встановленим F бітом, якщо у неї немає даних для передачі. Не готовий до Прийому (RNR) - використовується як позитивна квитанція і запит зупинити передачу I-кадрів до отримання наступного кадру RR.

Провідна або Комбінована станції можуть встановити біт P для уточнення статусу прийому веденої / комбінованої станції.

Ведена / комбінована станції можуть відповісти установкою біта P як індикації зайнятості станції.

Неприйняття (REJ).

Часто використовується як негативна квитанція приймача.

Неприйняття кадрів останнього вікна (повтор передачі з кадру N (r)).

Вибіркове неприйняття (SREJ).

Неприйняття конкретного кадру (повтор передачі одного кадру).

Таблиця 2.6. Види імен, команд, формат управління поля та ін.

Ім'я	**Команда / Відповідь**	**Опис**	**Інформація**	**Формат упр. поля 8 ... 7 ... 6 ... 5 ... 4 ... 3 ... 2 ... 1 ... ..**
Готовий до Прийому *(RR)*	C / R	Позитивна квитанція	Готовий до прийому *I*-кадру	.- N (R) - ... P / F ... 0 ... 0 ... 0 ... 1
Не готовий до Прийому *(RNR)*	C / R	Позитивна квитанція	Не готовий до Прийому	.- N (R) - ... P / F ... 0 ... 1 ... 0 ... 1
Неперервним *(REJ)*	C / R	Негативна квитанція	Повтор *N* кадрів	.- N (R) - ... P / F ... 1 ... 0 ... 1 ... 0
Вибірковий неперервним *(SREJ)*	C / R	Негативна квитанція	Повтор 1 кадру	.- N (R) - ... P / F ... 1 ... 1 ... 0 ... 1

U-кадри (ненумеровані) призначені для встановлення і розриву логічного з’єднання, а також інформування про помилки.

Поле M ненумерованих кадрів містить коди, що визначають тип команд, якими користуються два вузли на етапі встановлення з'єднання (наприклад, SABME, UA, REST).

Встановлення режиму (SNRM, SNRME, SARM, SARME, SABM, SABME, UA, DM, RIM, SIM, RD, DISC)

Ненумерована інформація (UP, UI)

Відновлення (FRMR, RSET)

Невірне поле управління

Перевищена довжина поля даних

Невірна довжина для даного типу кадрів

Невірний номер кадру

Інші (XID, TEST)

Таблиця 2.7. U-кадри (ненумеровані)

Ім'я	Команда / Відповідь	Опис	Інформація	Формат упр. поля 8 ... 7 ... 6 ... 5 ... 4 ... 3 ... 2 ... 1 .....
Встановити режим нормальної відповіді **SNRM**	C	Встановити режим		.. 1 ... 0 ... 0 ... P ... 1 ... 1 ... 0 ... 1
Встановити розширений режим нормальної відповіді **SNRME**	C	Встановити режим		.. 1 ... 1 ... 0 ... P ... 1 ... 1 ... 1 ... 1
Встановити режим асинхронної відповіді **SARM**	C	Встановити режим		.. 0 ... 0 ... 0..P/F..1 ... 1 ... 0 ... 1
Встановити розширений режим асинхронної відповіді **SARME**	C	Встановити режим		.. 0 ... 1 ... 0 ... P. .1 ... 1 ... 1 ... 1
Встановити асинхронний збалансований режим **SABM**	C	Встановити режим		.. 0 ... 0 ... 1..P/F..1 ... 1 ... 1 ... 1
Встановити розширений асинхронний збалансований режим **SABME**	C	Встановити режим		.. 0 ... 1 ... 1 ... P ... 1 ... 1 ... 1 ... 1
Встановити режим ініціалізації **SIM**	C	Ініціювати функцію контролю за лінією в адресуємій станції		.. 0 ... 0 ... 0..P/F..0 ... 1 ... 1 ... 1
Розрив з'єднання **DISC**	C	Розірвати логічне з'єднання		.. 0 ... 1 ... 0..P/F..0 ... 0 ... 1 ... 1
Ненумеровані підтвердження **UA**	R	Підтвердження прийому однієї з команд встановлення режимів		.. 0 ... 1 ... 0 .... F. .0 ... 0 ... 1 ... 1
Режим роз'єднання **DM**	R	Індикація режиму лог. роз'єднання		
Запит роз'єднання **RD**	R	Відповідь на команду DISC		.. 0 ... 1 ... 0..P/F..0 ... 0 ... 1 ... 1
Запит ініціалізації **RIM**	R	Необхідна ініціалізація	Запит команди SIM	
Ненумерована інформація **UI**	C / R	Використовується для обміну інформацією управління		.. 0 ... 0 ... 0..P/F..0 ... 0 ... 1 ... 1
Ненумероване опитування **UP**	C	Використовується для запиту керуючої інформації		.. 0 ... 0 ... 1 .. P ... .0 ... 0 ... 1 ... 1
Перезапуск лічильників **RSET**	C	Вик. для відновлення	Обнуляє N (R), N (S)	.. 1 ... 0 ... 0 .. P ... .1 ... 1 ... 1 ... 1
Обмін статусом **XID**	C / R	Вик. для запиту / передачі статусу		.. 1 ... 0 ... 1..P/F..1 ... 1 ... 1 ... 1
Тест **TEST**	C / R	Обмін ідентичними інф. полями для тесту		.. 1 ... 1 ... 1..P/F..0 ... 0 ... 1 ... 1
Не прийняття кадру **FRMR**	C / R	Повідомлення про невірний кадр		

2.3. L2TP – протокол підтримки віртуальних приватних мереж

L2TP (Layer 2 Tunneling Protocol — протокол тунелювання другого рівня) — в комп'ютерних мережах тунельний протокол, використовується для підтримки віртуальних приватних мереж. L2TP не забезпечує шифрування та конфіденційність сам по собі, він спирається на інкапсульований протокол для забезпечення конфіденційності [2.4].

Попри те, що L2TP діє подібно протоколу Канального рівня моделі OSI, в дійсності він є протоколом Сеансового рівня і використовує зареєстрований UDP-порт 1701.

Тепер звернемося до історії. 1996-1997 - конкуренція між протоколами L2F (Cisco) і PPTP (Microsoft) 1997 - угода між розробниками про спільну розробку протоколу L2TP 1999 - опублікований стандарт RFC 2661, що описує протокол L2TP Вважається, що протокол L2TP увібрав у себе найкращі риси

PPTP і L2F. Опублікований у 1999 році як пропонований стандарт RFC 2661, L2TP базувався головним чином на двох більш ранніх тунельних протоколах для PPP: протоколі естафетної передачі на другому рівні (L2F) від Cisco та тунельному протоколі точка-точка (PPTP) від Microsoft. По загальній думці, протокол L2TP увібрав в себе всі переваги PPTP і L2F. Нова версія цього протоколу, L2TPv3, була опублікована як запропонований у 2005 році стандарт RFC 3931. Головною перевагою L2TP є те, що цей протокол дозволяє створювати тунель не лише в мережах IP, але і в таких, як ATM, X.25 та frame relay.

Схема роботи може виглядати наступним чином: дистанційна система ініціює PPP-з'єднання з LAC через телефонну мережу PSTN. LAC потім прокладає тунель для PPP-з'єднання через Інтернет, Frame Relay або ATM до LNS, і таким чином здійснюється доступ до початкової LAN. Адреси віддаленій системі надаються вихідною LAN через узгодження з PPP NCP. Аутентифікація, авторизація та аккаунтинг можуть бути надані областю управління LAN, як якщо б користувач був безпосередньо з'єднаний з сервером мережевого доступу NAS. LAC-клієнт (EOM, яка виконує програму L2TP) може також брати участь в тунелюванні до вихідної LAN без використання окремого LAC, якщо EOM, що містить програму LAC-клієнта, вже має з'єднання з Інтернет. Створюється "віртуальне" PPP-з'єднання, і локальна програма L2TP LAC формує тунель до LNS. Як і у вищеописаному випадку, адресація, аутентифікація, авторизація та аккаунтинг будуть забезпечені областю управління вихідної LAN [2.5].

Проведемо огляд протоколу L2TP. L2TP використовує два види пакетів: керуючі та інформаційні повідомлення.

Керуючі повідомлення використовуються при встановленні, підтримці та анулювання тунелів і викликів.

Інформаційні повідомлення використовуються для інкапсуляції PPP-кадрів, що пересилаються по тунелю.

Керуючі повідомлення використовують надійний контрольний канал в межах L2TP (таб. 2.8), щоб гарантувати доставку. Інформаційні повідомлення при втраті не пересилаються повторно.

Таблиця 2.8. Структура протоколу

PPP кадри	
L2TP інформаційні повідомлення	L2TP управляючі повідомлення
L2TP інформаційний канал (ненадійний)	L2TP канал управління (надійний)
Транспортування пакетів (UDP, FR, ATM тощо)	

Керуюче повідомлення має порядковий номер, який використовується в керуючому каналі для забезпечення надійної доставки.

Інформаційні повідомлення можуть використовувати порядкові номери, щоб відновити порядок пакетів і детектувати втрату кадрів.

Всі коди надсилаються в порядку, прийнятому для мереж.

L2TP застосовує як транспортний протокол UDP і використовує однаковий формат повідомлень як для управління тунелем, так і для пересилання даних. L2TP в реалізації Microsoft використовує як контрольних повідомлень пакети UDP, що містять шифровані пакети PPP [2.3, 2.6].

L2TP часто використовують для створення тунелю через VPN.

Для забезпечення безпеки L2TP-пакетів зазвичай використовують протокол IPsec, який надає конфіденційність, аутентифікацію та цілісність. Комбінація цих двох протоколів відома як L2TP/IPsec.

Кінцеві вузли L2TP тунелю називаються LAC (L2TP концентратора доступу) і LNS (L2TP Server Network). LAC є ініціатором тунелю, тоді LNS - сервер, який очікує нових тунелів.

Коли тунель встановлено, мережевий трафік між вузлами є двонаправленим. Потім, протоколи більш високих рівнів запускаються всередині тунелю L2TP. Для цього, L2TP сесія встановлюється всередині тунелю для кожного протоколу вищого рівня (наприклад, для ПКС). Як LAC,

так і LNS можуть ініціювати сесії.

Трафік для кожної сесії ізолюється за допомогою L2TP, тому можливо налаштувати кілька віртуальних мереж через один тунель.

Пакети L2TP для контрольного та інформаційного каналів використовують один і той же формат заголовка (табл. 2.9.):

Таблиця 2.9. Формат заголовка

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16						31
T	L	x	x	S	x	O	P	x	x	x	x	Версія				Довжина(опц)						
ID тунелю																ID сесії						
Ns (опц)																Nr (опц)						
Offset Size (опц)																Offset Pad (опц).....						
Payload data																						

Біт тип (T) характеризує різновид пакета.

Він встановлюється рівним 0 для інформаційних повідомлень і 1 - для управляючих [2.6, 2.7].

Якщо біт довжини (L) дорівнює 1, поле довжина присутнє.

Для керуючих повідомлень цей біт повинен бути рівний 1.

Біти x зарезервовані для майбутніх застосувань.

Всі зарезервовані біти повинні бути встановлені рівними 0 для вихідних повідомлень і ігноруватися для вхідних.

Якщо біт послідовності (S) дорівнює 1, присутні поля Ns і Nr.

Біт S для керуючих повідомлень повинен бути рівний 1.

Якщо біт зсуву (O) дорівнює 1, поле величини зміщення присутнє.

Біт O для керуючих повідомлень повинен бути рівний 0.

Біт пріоритету (P) повинен бути рівний 0 для всіх керуючих повідомлень. Для інформаційних повідомлень - якщо цей біт дорівнює 1, це інформаційне повідомлення має пріоритет в черзі.

Поле Ver вказує версію заголовка інформаційного повідомлення L2TP.

Значення 1 зарезервовано для детектування пакетів L2F в умовах, коли

вони приходять упереміш з L2TP-пакетами. Пакети, отримані з невідомим значенням поля Ver, відкидаються.

Поле Довжина (опціонально) вказує загальну довжину повідомлення в октетах.

ID-тунелю містить ідентифікатор керуючого з'єднання. Ідентифікатори тунелю L2TP мають тільки локальне значення. Тобто, різні кінці одного тунелю повинні мати різні ID. ID-тунелю в кожному повідомленні має бути тим, яке очікує отримувач. ID-тунелю вибираються при формуванні тунелю [2.3, 2.7].

ID-сесії визначає ідентифікатор для сесії даного тунелю. Сесії L2TP іменуються за допомогою ідентифікаторів, які мають тільки локальне значення. ID-сесії в кожному повідомленні має бути тим, яке очікує отримувач. ID-сесії вибираються при формуванні сесії.

Поле Ns визначає порядковий номер інформаційного або керуючого повідомлення, починаючи з нуля і збільшуючись на 1 (по модулю 216) для кожного посланого повідомлення.

Поле Nr містить порядковий номер, який очікується для наступного повідомлення. Таким чином, Nr робиться рівним Ns останнього по порядку отриманого повідомлення плюс 1 (по модулю 216). В інформаційних повідомленнях, Nr зарезервовано і, якщо присутній (це визначається S-бітом), повинно ігноруватися при отриманні.

Поле величина зсуву (Offset Size), якщо є, специфікує число октетів після заголовка L2TP, де має починатися поле даних. Вміст заповнювача зсуву не визначено. Якщо поле зміщення присутній, заголовок L2TP завершується після завершального октету заповнювача зсуву.

Типи управляючих повідомлень приведено нижче.

Тип повідомлення AVP визначає специфічний тип керуюючого повідомлення, що відправляється.

Управління контрольним з'єднанням:

0 (зарезервовано)

1 (SCCRQ) Start-Control-Connection-Request

2 (SCCRP) Start-Control-Connection-Reply

3 (SCCCN) Start-Control-Connection-Connected

4 (StopCCN) Stop-Control-Connection-Notification

5 (зарезервовано)

6 (HELLO) Hello

Управління викликами (Call Management)

7 (OCRQ) Outgoing-Call-Request

8 (OCRP) Outgoing-Call-Reply

9 (OCCN) Outgoing-Call-Connected

10 (ICRQ) Incoming-Call-Request

11 (ICRP) Incoming-Call-Reply

12 (ICCN) Incoming-Call-Connected

13 (зарезервовано)

14 (CDN) Call-Disconnect-Notify

Повідомлення про помилки

15 (WEN) WAN-Error-Notify

Управління сесією PPP

16 (SLI) Set-Link-Info

Протокольні операції мають певні правила. Необхідна процедура встановлення PPP-сесії тунелювання L2TP включає в себе два етапи [2.3, 2.4]:

Встановлення керуючого каналу для тунелю. Формування сесії відповідно до запиту вхідного або вихідного виклику.

Тунель і відповідний керуючий канал повинні бути сформовані до ініціалізації вхідного або вихідного виклику. L2TP-сесія повинна бути реалізована до того, як L2TP зможе передавати PPP-кадри через тунель. В одному тунелі можуть існувати кілька сесій між одними і тими ж LAC і LNS.

Керуюче з'єднання є первинним, яке має бути реалізовано між LAC і LNS перед запуском сесії. Встановлення керуючого з'єднання включає в себе

безпечну ідентифікацію партнера, а також визначення версії L2TP, можливостей каналу, кадрового обміну і т.д.

L2TP включає в себе просту, опціонну, CHAP-подібну систему аутентифікації тунелю в процесі встановлення керуючого з'єднання.

Після успішного встановлення керуючого з'єднання можуть формуватися індивідуальні сесії. Кожна сесія відповідає одному PPP інформаційного потоку між LAC і LNS. На відміну від встановлення керуючого з'єднання, встановлення сесії є асиметричним щодо LAC і LNS. LAC запитує LNS доступ до сесії для вхідних запитів, а LNS запитує LAC запустити сесію для роботи з вихідними запитами.

Коли тунель сформований, PPP-кадри від віддаленої системи, одержувані LAC, звільняються від CRC, канальних заголовків і т. ін., інкапсульованих в L2TP, і переадресовуються через відповідний тунель. LNS отримує L2TP-пакет і обробляє інкапсульований PPP-кадр, як якщо б він був отриманий через локальний інтерфейс PPP.

Відправник повідомлення, асоційований з певною сесією і тунелем, поміщає ID сесії і тунелю (специфіковані партнером) у відповідні поля заголовка всіх вихідних повідомлень.

Використання порядкових номерів в каналі даних Порядкові номери, визначені в заголовку L2TP, використовуються для організації надійного транспортування керуючих повідомлень. Кожен партнер підтримує окрему нумерацію для керуючого з'єднання і для кожної інформаційної сесії в межах тунелю.

На відміну від каналу управління L2TP, інформаційний канал L2TP використовує нумерацію повідомлень не для повторної пересилки, а для детектування втрат пакетів і / або відновлення вихідної послідовності пакетів, перемішаних при транспортуванні [2.3].

LNS може ініціювати заборону нумерації повідомлень в будь-який час в ході сесії (включаючи перше інформаційне повідомлення).

Переривання сесії може бути ініційовано LAC або LNS і виконується шляхом посилки керуючого повідомлення CDN. Після того як остання сесія перервана, керуюче з'єднання може бути також розірвано.

Розрив контрольного з'єднання може бути ініційований LAC або LNS і виконується шляхом посилки одного керуючого повідомлення StopCCN.

Інкапсуляція пакетів L2TP/IPSec виконується в два етапи [2.3., 2.6].

1. Інкапсуляція L2TP. Кадр PPP (IP-датаграма або IPX-датаграма) полягає в оболонку з заголовком L2TP і заголовком UDP.

2. Інкапсуляція IPSec. Потім отримане L2TP-повідомлення полягає в оболонку з заголовком і трейлером IPSec ESP (Інкапсуляція Security Payload), трейлером перевірки автентичності IPSec, що забезпечує цілісність повідомлення та перевірку справжності, і заголовком IP. У заголовку IP-адреси джерела і приймача відповідають VPN-клієнта і VPN-сервера.

На завершення L2TP виконує другий PPP-інкапсуляцію для підготовки даних до передачі. L2TP пакет складається з 31 біта (табл. 2.10):

Таблиця 2.10. Структура L2TP пакета

Біти 0-15	**Біти 16-31**
Flags and Version Info	Length (опціонально)
Tunnel ID	Session ID
Ns (опціонально)	Nr (опціонально)
Offset Size (опціонально)	Offset Pad (опціонально).....
Payload data	

Значення полів [2.3, 2.6]:

Flags and version

Прапори, вказують тип пакету (керуючий або дані) і наявність полів з довжиною, номером послідовності і зрушенням.

Length (опціонально)

Повна довжина повідомлення в байтах, є лише якщо встановлено прапор довжини.

Tunnel ID

Відображає ідентифікатор керуючого з'єднання.

Session ID

Відображає ідентифікатор сесії всередині тунелю.

- ***Ns (опціонально)***

Послідовний номер для даних або керуючого повідомлення, починаючи з 0 і збільшуючись на одиницю (за модулем 2^{16}) для кожного відправленого повідомлення. Існує, коли встановлений відповідний прапор.

- ***Nr (опціонально)***

Послідовний номер повідомлення, очікуваного для прийняття. *Nr* встановлюється як Ns останнього отриманого повідомлення плюс один (за модулем 2). У повідомленнях з даними, *Nr* зарезервований, і, якщо існує, зобов'язаний бути проігноровано.

- ***Offset Size (опціонально)***

Визначає, де знаходяться дані після *L2TP* заголовка. Якщо це поле існує, *L2TP* заголовок закінчується після останнього байта *offset padding*. Існує, коли встановлений відповідний прапор.

- ***Offset Pad (опціонально)***

Змінної довжини, вказується *offset size*. Вміст поля невизначено.

- ***Payload data***

Самі передані дані. Змінної довжини (Максимальний розмір = Максимальному розміром *UDP* пакету - розмір заголовка *L2TP*) [2.2, 2.5].

Реалізація L2TP через специфічне середовище.

- Протокол *L2TP* є самодокументованим, що працюють поверх рівня, який служить для транспортування.

- Однак, необхідні деякі деталі підключення до середовища, для того щоб забезпечити сумісність різних реалізацій.

- ***L2TP і Ipsec***

■ При роботі поверх *IP, IPsec (безпечний IP)* надає безпеку на пакетному рівні. Всі керуючі та інформаційні пакети *L2TP* в конкретному тунелі виглядають для системи *IPsec*, як звичайні інформаційні *UDP / IP-пакети*. Крім транспортної безпеки *IP, IPsec* визначає режим роботи, який дозволяє тунелювати *IP*-пакети, а так само засоби контролю доступу, які необхідні для додатків, що підтримують IPsec. Ці засоби дозволяють фільтрувати пакети на основі характеристик мережевого і транспортного рівнів. У моделі *L2TP*-тунелю аналогічна фільтрація виконується на *PPP*-рівні або мережевому рівні поверх *L2TP*.

Міркування безпеки

■ Протокол L2TP стикається при своїй роботі з кількома проблемами безпеки. Нижче розглянуті деякі підходи для вирішення цих проблем.

■ **Безпека на кінці тунелю**

■ Кінці тунелю можуть опційно виконувати процедуру аутентифікації один одного при встановленні тунелю. Ця аутентифікація має ті ж атрибути безпеки, що і *CHAP*, і володіє розумним захистом проти атак відтворення і спотворення в процесі встановлення тунелю. Для реалізації аутентифікації *LAC* і *LNS* повинні використовувати загальний секретний ключ.

■ **Безпека пакетного рівня**

■ Забезпечення безпеки *L2TP* вимагає, щоб транспортна середу могла забезпечити шифрування переданих даних, цілісність повідомлень і аутентифікацію послуг для всього *L2TP*-трафіку. Сам же *L2TP* відповідальний за конфіденційність, цілісність і аутентифікованість *L2TP*-пакетів всередині тунелю.

■ ***Layer 2 Forwarding Protocol (L2F)*** *(Протокол естафетної передачі на другому рівні)* - один з протоколів тунелювання, розроблений

компанією *Cisco Systems* для створення віртуальних приватних мереж зв'язку через Інтернет.

■ *L2F* не забезпечує шифрування і конфіденційності сам по собі, він спирається на інкапсулюємий протокол для забезпечення конфіденційності. *L2F* був спеціально розроблений для тунелювання трафіку протоколу *PPP*.

Формат пакетов L2F

■ Протокол конкурував з іншим популярним тунельним протоколом *PPTP* з 1996 року. Підсумком конкуренції стало підписання в 1997 році угоди між розробниками цих протоколів (компаніями Cisco і Microsoft) про створення нового протоколу, на основі як *L2F*, так і *PPTP*.

Структура протоколу L2F виглядає наступним чином (табл. 2.11):

Табоиця 2.11. Структура протоколу *L2F*

+	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**16-23**	24-31
0	F	K	P	S	0	0	0	0	0	0	0	0	C	Версія			Протокол	Порядковий номер
32	Ідентифікатор мультипл																Ідентификатор клиєнта	
64	Розмір																зсув	
96	Ключ пакета																	
128	Тіло повідомлення																	
...	...																	
...	Контрольна сума																	

Додатковий матеріал залишено на самостійне вивчення студентів [2.2].

2.4 PPP - протокол точка-точка

PPP (Point-to-Point Protocol) — протокол точка-точка, протокол канального рівня мережевої моделі OSI [2.1-2.7].

PPP — це механізм для створення й запуску IP (Internet Protocol) й інших мережевих протоколів на послідовних лініях зв'язку — прямий послідовний зв'язок (по нуль-модемному кабелю), зв'язок поверх Ethernet або модемний зв'язок по телефонних лініях.

Використовуючи PPP, можна під'єднати комп'ютер до PPP-сервера й одержати доступ до ресурсів мережі, до якої під'єднаний сервер так, начебто ви підключені безпосередньо до цієї мережі.

Протокол PPP є основою для всіх протоколів другого рівня. Зв'язок по протоколу PPP складається із чотирьох стадій встановлення зв'язку (здійснюється вибір протоколів аутентифікації, шифрування, стиснення і встановлюються параметри з'єднання), визначення аутентичності користувача

(реалізуються алгоритми аутентифікації, на основі протоколів PAP, CHAP або MS-CHAP), контроль повторного виклику PPP (необов'язкова стадія, у якій підтверджується справжність віддаленого клієнта), виклик протоколу мережевого рівня (реалізація протоколів установлених на першій стадії). PPP включає IP, IPX й NetBEUI пакети всередині PPP кадрів.

Зазвичай використовується для встановлення прямих з'єднань між двома вузлами. Широко застосовується для з'єднання комп'ютерів за допомогою телефонної лінії. Також використається поверх широкосмугових з'єднань. Багато хто з провайдерів використають PPP для надання комутованого доступу в Інтернет.

PPP протокол був розроблений на основі HDLC і доповнений деякими можливостями, які до цього зустрічалися тільки в пропрієтарних протоколах.

Пропрієтарний протокол [2.2-2.7].

Link Control Protocol (LCP) забезпечує автоматичне налаштування

інтерфейсів на кожному кінці (наприклад, установка розміру пакетів) і опціонально проводить аутентифікацію. Протокол LCP працює поверх PPP, тобто до роботи LCP повинен бути PPP зв'язок.

RFC 1994 описує Challenge-handshake authentication protocol (CHAP), який є найкращим для з'єднань з провайдерами. Вже застарілий Password authentication protocol (PAP) все ще іноді використовується.

Іншим варіантом аутентифікації через PPP є розширений протокол аутентифікації (EAP - Extensible Authentication Protocol).

Після того, як з'єднання було встановлено, поверх нього може бути налаштована додаткова мережа.

Зазвичай використовується Internet Protocol Control Protocol (IPCP), хоча Internetwork Packet Exchange Control Protocol (IPXCP) і AppleTalk Control Protocol (ATCP) були колись популярні.

Internet Protocol Version 6 Control Protocol (IPV6CP) отримає більше поширення в майбутньому, коли IPv6 замінить IPv4 як основний протокол мережного рівня.

PPP дозволяє працювати декільком протоколам мережного рівня на одному каналі зв'язку. Іншими словами, всередині одного PPP-з'єднання можуть передаватися потоки даних різних мережевих протоколів (IP, IPX Novell і т. д.), а також дані протоколів канального рівня локальної мережі. Для кожного мережевого протоколу використовується протокол управління мережею (NCP) який його конфігурує (погоджує деякі параметри протоколу).

Виявлення закільцьованих зв'язків [2.1-2.7].

PPP виявляє закільцьовані зв'язки, використовуючи особливість, що включає магічні числа (magic numbers).

Коли вузол відправляє повідомлення PPP LCP, вони можуть включати в себе магічне число.

Якщо лінія закольцована, вузол отримує повідомлення LCP з його власним магічним числом замість отримання повідомлення з магічним числом

клієнта.

Найбільш важливі особливості

- *Link Control Protocol* встановлює і завершує з'єднання, дозволяючи вузлам визначати налаштування з'єднання. Також він підтримує і байт-, і біт-орієнтовані кодування.
- *Протокол управління мережею (Network Control Protocol)* використовується для визначення налаштувань мережевого рівня, таких як мережева адреса або налаштування стиснення, після того як з'єднання було встановлено.

2.5. PPTP - протокол захищення з'єднання з сервером

PPTP (Point-to-Point Tunneling Protocol) — тунельний протокол типу точка-точка, що дозволяє комп'ютеру встановлювати захищене з'єднання з сервером за рахунок створення спеціального тунелю в стандартній, незахищеній мережі [2.1-2.4].

PPTP поміщає (інкапсулює) кадри PPP в IP-пакети для передачі по глобальній IP-мережі, наприклад інтернет. PPTP може також використовуватися для організації тунелю між двома локальними мережами.

PPTP використовує додаткове TCP- з'єднання для обслуговування тунелю.

Протоколи туннелювання потрібні тому, що деякі протоколи не можуть маршрутизуватися через певні мережі.

Кадри PPP, які не маршрутизуються через мережу Інтернет, не можуть самостійно знайти правильний маршрут через Інтертет до одержувача, для цього їм потрібна підтримка.

Саме для цього і потрібні протоколи тунелювання. Вони дозволяють проводити інкапсуляцію пакетів PPP, завдяки чому кадри правильно доходять до одержувача.

Протокол PPTP (Point-to-Point-Tunneling Protocol) розроблений компанією Microsoft спільно з компаніями Ascend Communications, 3Com/Primary Access, ECI-Telematics і US Robotics.

Цей протокол був представлений в робочу групу «PPP Extentions» IETF як претендент на стандартний протокол створення захищеного каналу при доступі віддалених користувачів через публічні мережі (в першу чергу Internet) до корпоративних мереж.

Специфікація протокола була опублікована RFC 2637 в 1999 році. Вона не була ратифікована IETF. Протокол вважається менш захищеним, ніж IPSec.

PPTP працює, встановлюючи звичайний PPP сеанс протилежною стороною за допомогою протоколу Generic Routing Encapsulation. Друге з'єднання на TCP-порту 1723 використовується для ініціювання та управління GRE-з'єднанням. PPTP складно перенаправляти за мережевий екран, оскільки він вимагає одночасного встановлення двох мережевих сеансів.

PPTP-трафік може бути зашифрованим за допомогою MPPE. Для аутентифікації клієнтів можуть використовуватися різноманітні механізми, найбезпечніші з них — MS-CHAPv2 и EAP-TLS.

Багатопротокольність — основна перевага інкапсулюючих протоколів канального рівня, до яких належить протокол PPTP.

PPTP працює шляхом інкапсуляції «рідних» пакетів локальної мережі — наприклад, пакетів IPX — всередину пакетів TCP / IP. Весь пакет IPX, включаючи його керуючу інформацію, стає корисним навантаженням для пакета TCP / IP, який потім можна передавати по Internet. Програмні засоби на іншому кінці лінії зв'язку витягують пакет IPX і направляють його для нормальної обробки згідно з його власним протоколом. Цей процес називається тунелюванням — ймовірно, тому, що створюється коридор в Internet, що з'єднує два вузли.

Процес зв'язку по протоколу PPTP [2.1, 2.6-2.8].

Введення IP-адреси ініціює передачу запиту серверу на початок сеансу.

Клієнт чекає від сервера підтвердження імені користувача і пароля і відповіді повідомленням, що з'єднання встановлено.

У цей момент починає свою роботу канал PPTP, і клієнт може приступити до тунелювання пакетів серверу. Оскільки вони можуть бути пакетами IPX і NetBEUI, сервер може виконувати з ними свої звичайні процедури забезпечення захисту.

В основі обміну даними по протоколу PPTP лежить керуюче з'єднання PPTP — послідовність керуючих повідомлень, які встановлюють і обслуговують тунель. Повне з'єднання PPTP складається тільки з одного з'єднання TCP / IP, яке вимагає передачі ехо-команд для підтримки його відкритим, поки виконуються транзакції.

У протоколі PPTP визначено дві схеми його застосування. Перша схема розрахована на підтримку захищеного каналу між сервером віддаленого доступу ISP (Internet Service Provider — провайдера Internet) і прикордонним маршрутизатором корпоративної мережі. Друга схема показує, як користувач двічі встановлює віддалене з'єднання за допомогою утиліти Dial-Up Networking, що являє собою клієнтську частину сервісу віддаленого доступу Windows.

Реалізація PPTP. Cisco першою реалізувала PPTP і пізніше ліцензувала цю технологію корпорації Microsoft [2.1-2.5].

PPTP вдалося домогтися популярності завдяки тому, що це перший протокол тунелювання, який був підтриманий корпорацією Microsoft. Всі версії Microsoft Windows, починаючи з Windows 95 OSR2, мають у своєму складі PPTP-клієнт, однак існує обмеження на два одночасних вихідних з'єднання. А сервіс віддаленого доступу для Microsoft Windows містить у себі PPTP сервер.

До недавнього часу в Linux- дистрибутивах була відсутня повна підтримка PPTP через побоювання патентних претензій з приводу протоколу MPPE. Вперше повна підтримка MPPE з'явилася в Linux 2.6.13. Офіційно підтримка PPTP була розпочата з версії ядра Linux 2.6.14. Тим не менше, сам факт застосування MPPE в PPTP фактично не забезпечує безпеку протоколу

PPTP.

- Операційна система *FreeBSD* підтримує *PPTP* протокол, використовуючи як сервер *PPTP* порт *mpd* (/ usr / ports / net / mpd), використовуючи підсистему *netgraph*. Як клієнт *PPTP* в системі *FreeBSD* може виступати або порт pptpclient (/ usr / ports / net / pptpclient), або порт mpd, що працює в режимі клієнта.

- *Mac OS X* поставляється з вбудованим PPTP клієнтом. Cisco і Efficient Networks продають PPTP клієнти для старіших версій Mac OS. *КПК Palm*, що підтримують технологію *Wi-Fi*, поставляються з *PPTP* клієнтом Mergic.

- Microsoft Windows Mobile 2003 і новіші версії також підтримують протокол *PPTP*.

Безпека протоколу PPTP [2.1-2.5].

Протокол PPTP дозволяє створювати захищені канали для обміну даними по різних мережевих протоколах — IP, IPX або NetBEUI. Дані цих протоколів інкапсулюються за допомогою протоколу PPTP в пакети протоколу IP, за допомогою якого переносяться в зашифрованому вигляді через мережу TCP/IP.

Інкапсулюється вихідний кадр PPP, тому протокол PPTP можна віднести до класу протоколів інкапсуляції канального рівня в мережевий.

PPTP був об'єктом численних аналізів безпеки, в ньому були виявлені серйозні проблеми.

Відомі проблеми стосуються аутентифікації PPP протоколів, влаштування протоколу MPPE та інтеграції між аутентифікаціями MPPE і PPP для створення ключа сеансу. Короткий огляд даних проблем:

MSCHAP-v1 абсолютно ненадійний. Існують утиліти для легкого вилучення хешів паролів з перехопленого обміну MSCHAP-v1.

MSCHAP-v2 вразливий до словникової атаки на перехоплені challenge response пакетів. Існують програми, що виконують даний процес.

При використанні MSCHAP-v1, MPPE використовує однаковий RC4

сеансовий ключ для шифрування інформаційного потоку в обох напрямках. Тому стандартним методом є виконання XOR'а потоків з різних напрямків разом, тому криптоаналітик може дізнатися про ключ.

MPPE використовує RC4 потік для шифрування. Не існує методу для аутентифікації цифробуквенного потоку, і тому даний потік вразливий до атаки, що виконує підмін бітів. Зловмисник легко може змінити потік при передачі й змінити деякі біти, щоб змінити вихідний потік без небезпеки свого виявлення. Цей підмін біт може бути виявлений за допомогою протоколів, які підраховують контрольні суми.

2.6 FC - сімейство протоколів для високошвидкісного передавання даних

Fibre Channel (FC) (fibre channel) — сімейство протоколів для високошвидкісного передавання даних. Стандартизацією протоколів займається Технічний комітет T11, що входить до складу Міжнародного комітету із стандартів у сфері IT (InterNational Committee for Information Technology Standards — INCITS), акредитованого Американським національним інститутом стандартів (ANSI) [2.1-2.6].

Початкове застосування FC в області суперкомп'ютерів згодом практично повністю перейшло в сферу мереж зберігання даних, де FC використовується як стандартний спосіб підключення до систем зберігання даних рівня підприємства.

Fibre Channel Protocol (FCP) транспортний протокол (як TCP в IP-мережах), що інкапсулює протокол SCSI по мережах Fibre Channel. Є основою побудови мереж зберігання даних.

Коли технологія тільки була розроблена, вона підтримувала лише оптоволокно (fiber); підтримку мідних кабелів було додано пізніше. Незважаючи на це, комітет розробки вирішив зберегти ім’я, але перейти на

британський правопис fibre в назві стандарту. Американське написання fiber стосується лише оптичних кабелів.

Мережу Fibre Channel можна прокладати як по мідних кабелях, так і по оптоволокні.

Версії Fibre Channel (таб. 1.12). Історія Fibre Channel почалася в 1985 році, а в 1994 році був затверджений ANSI як стандарт, що спрощував інтерфейс HIPPI, для якого застосовувався масивний 50-парний кабель з громіздкими коннекторами. Спочатку інтерфейс Fibre Channel повинен був підвищити дальність і спростити підключення ліній передачі, а не збільшити швидкість [6-15].

Таблиця 2.12. Версії Fibre channel

Версії Fibre Channel			
Назва	Пропускна здатність (Gbps)	Продуктивність (MBps)	Рік
1GFC	1.0625	100	1997
2GFC	2.125	200	2001
4GFC	4.25	400	2005
8GFC	8.5	800	2008
10GFC Послідовний	10.51875	1000	2004
10GFC Паралельний	12.75		
16GFC	14.025	3200	2011
20GFC	21.04	2000	2008
64GFC	28.05	6400	2014

Топології Fibre Channel [2.2, 2.4]. Топології FC визначають взаємне підімкнення пристроїв, а саме передавачів (трансміттерів) і приймачів (ресіверів). Існує три типи топології FC:

Точка — точка (point-to-point). Пристрої сполучені безпосередньо, трансміттер одного пристрою сполучений з ресівером іншого і навпаки. Всі відправлені одним пристроєм кадри призначені для другого пристрою [2.3].

Керована петля (arbitrated loop) Пристрої об'єднані в петлю трансміттер кожного пристрою сполучений з ресівером наступного. Перед тим, як петля зможе служити для передачі даних, пристрої домовляються про адреси. Для передачі даних по петлі пристрій повинен оволодіти «естафетою» (token).

Додавання пристрою в петлю приводить до припинення передачі даних і збору заново петлі. Для побудови керованої петлі використовують концентратори, які здатні розмикати або замикати петлю при додаванні нового пристрою або виході пристрою з петлі.

Комутована зв'язна архітектура (switched fabric). Заснована на застосуванні комутаторів. Дозволяє підключати більшу кількість пристроїв, ніж в керованій петлі, при цьому додавання нових пристроїв не впливає на передачу даних між вже підключеними пристроями. Оскільки на основі комутаторів можна будувати складні мережі, на комутаторах підтримуються розподілені служби управління мережею (fabric services), передачі даних, що відповідають за маршрути, реєстрацію в мережі і привласнення мережевих адрес і інші.

Іноді під топологією FC помилково мають на увазі топологію мережі зберігання даних, тобто, взаємне підключення устаткування інфраструктури і крайових пристроїв.

Рівні Fibre Channel. Fibre Channel складається з п'яти рівнів [2.8]:

- FC-0 Фізичний. Описує середовище передачі, трансивери, коннектори і типи використовуваних кабелів. Включає визначення електричних і оптичних характеристик, швидкостей передачі даних і інших фізичних компонентів. Підтримується як оптичне, так і електричне середовище (вита пара, коаксіальний або твінаксіальний кабелі, а також багатомодове або одномодове волокно), із швидкістю передачі даних від 133 мегабіт/с до 10 гігабит/с на відстані до 50 кілометрів.
- FC-1 Кодування. Описує процес 8b/10b Кодування (кожні 8 біт даних кодуються в 10-бітовий символ (Transmission Character)), спеціальні символи і контроль помилок. Для 10GFC використовується кодування 64b/66b, внаслідок цього 10GFC несумісний з 1/2/4/8GFC.
- FC-2 Кадрування і сигналізація. Описує сигнальні протоколи. На цьому рівні відбувається визначення слів, розбиття потоку даних на кадри.

Визначає правила передачі даних між двома портами, класи служб).

- FC-3 Загальних для вузла служб. Визначає базові і розширені служби для транспортного рівня, а також такі особливості, як: розщеплювання потоку даних (striping) (Можливість передачі потоку даних через декілька з'єднань (маршрутів), відображення безлічі портів на один пристрій.
- FC-4 Відображення протоколів. Надає можливість інкапсуляції інших протоколів (SCSI, ATM, IP, HIPPI , AV, VI, IBM SBCCS і багато інших.)

Порти Fibre Channel [2.2, 2.9-2.13].

Залежно від підтримуваної топології і типу пристрою порти розділяються на декілька типів:

Порти вузлів:

- **N_Port (Node port)**, порт пристрою з підтримкою топології FC-P2P («точка-точка») або FC-SW (з комутатором).
- **NL_Port (Node Loop port)**, порт пристрою з підтримкою топології FC-AL (arbitrated loop - керована петля).

Порти комутатора/маршрутизатора (тільки для топології FC-SW):

- **F_Port (Fabric port)**, порт «тканини». Використовується для підключення портів типу **N_Port** до комутатора. Не підтримує топологію петлі.
- **FL_Port (Fabric Loop port)**, порт «тканини» з підтримкою петлі. Використовується для підключення портів типу NL_Port до комутатора.
- **E_Port (Expansion port)**, порт розширення. Використовується для з'єднання комутаторів. Може бути сполучений тільки з портом типу **E_Port**.
- **EX_port** порт для з'єднання FC-маршрутизатора і FC-коммутатора. З боку комутатора він виглядає як звичайний **E_port**, а з боку маршрутизатора це **EX_port**.
- **TE_port (Trunking Expansion port (E_port))** внесений до Fibre Channel компанією CISCO, зараз прийнятий як стандарт. Це розширений ISL або EISL. TE_port надає крім стандартних можливостей **E_port**

маршрутизацію множинних VSANs (Virtual SANs). Це реалізовано застосуванням нестандартного кадру Fibre Channel (vsan тегування).

Загальний випадок:

- **L_Port (Loop port)**, будь-який порт пристрою з підтримкою топології «Петля» - **NL_port** або **FL_port**.
- **G_port (Generic port)**, порт з автовизначенням. Автоматично може визначатися як порт типу **E_Port**, **N_Port**, **NL_Port**.

Розглянемо варіанти оптичного середовища передачі даних та їх характеристики в таб. 20. Оптичний кабель на кінцях має коннектори Fibre channel (рис. 2.1)

Рис. 2.1. Коннектори **Fibre channel** - LC (зліва) і SC (справа)

Таблиця 2.13. Типи середовища та їх характеристики

Тип середовища	Швидкість (Mbyte/s)	Передавач	Модифікація	Відстань
Одномодове волокно	400	1300 нм Довгохвильовий лазер	400-SM-LL-I	2 м - 2 км
	100	1550 нм Довгохвильовий лазер	100-SM-LL-V	2 м - >50 км
		1300 нм Довгохвильовий лазер	100-SM-LL-I	2 м - 2 км
	200	1550 нм Довгохвильовий	200-SM-LL-V	2 м - >50 км
		1300 нм Довгохвильовий лазер	200-SM-LL-L	2 м - 10 км
		1300 нм Довгохвильовий	200-SM-LL-I	2 м - 2 км
Багатомодове волокно (50µм)	400	850 nm Короткохвильовий лазер	400-M5-SN-I	0.5 м - 150 м
	200		200-M5-SN-I	0.5 м - 300 м
	100		100-M6-SN-I	0.5 м - 300 м
			100-M6-SL-I	2 м - 175 м

Устаткування Fibre Channel [2.2-2.5, 2.9].

Устаткування для інфраструктури Fibre Channel підрозділяється на декілька класів.

- *Директори* — багатопортові модульні комутатори з високим ступенем доступності.
- *Виділені комутатори (standalone switches)* — комутатори з фіксованою кількістю портів.
- *Комутатори зі стеком (stackable switches)* — комутатори, що мають додаткові високопродуктивні порти для зв'язку незалежних шасі між собою.
- *Вбудовувані комутатори (embedded switches)* — комутатори, що вбудовуються в блейд-корзину (blade enclosure), де є розділення портів за функцією (порти, призначені для підключення серверів, не можуть бути використані для міжкомутаторних з'єднань).
- *Концентратори (hubs)* — пристрої, що забезпечують зв'язок в керованій петлі (Arbitrated Loop).
- *Концентратори-комутатори (loop-switches)* — комутатори, що забезпечують зв'язок в керованій петлі (*Arbitrated Loop*). Концентратори і концентратори-комутатори практично не використовуються для підключення крайових пристроїв; використовуються для підключення дисків до контроллерів в дискових масивах.

Для збільшення дальності з'єднання використовують додаткове трансмісійне устаткування, таке як мультиплексори на основі WDM і ін. Основні виробники устаткування для інфраструктури Fibre Channel: Brocade, Cisco, QLogic, Emulex.

Логічні елементи потоку даних [2.1-2.5, 2.10].

При передачі даних виділяють наступні логічні послідовності:

- ***Впорядковані набори***

- Чотирибайтні слова (*Transmission Words*), що містять дані і спеціальні символи. Розбиття потоку даних на впорядковані набори дозволяє зберігати синхронізацію між передавачем і ресівером на рівні бітів і слів. Впорядковані набори завжди починаються з символу K28.5. Основні типи наборів визначаються сигнальним протоколом.

■ ***Роздільники кадрів***

■ Роздільники кадрів використовуються для відділення одного кадру від іншого. Існує два таких набори:

- *Початок кадру (Start Of Frame, SOF)*
- *Кінець кадру (End Of Frame, EOF)*

■ ***Базові сигнали***

- *Сигнал бездіяльності (Idle).* Передається для позначення готовності приймати і відправляти кадрів.
- Сигнал готовності ресівера (*Receiver Ready, R_RDY).* Використовується при управлінні потоком даних (див. Класи Обслуговування) для індикації наявності місця в буфері ресівера.
- *Базові послідовності.* Передаються для сповіщення про нестандартний стан порту. При отриманні такої послідовності у відповідь посилається відповідна послідовність або сигнал бездіяльності.

Стандарт підтримує чотири послідовності:

- *Offline (OLS)*
- *Not Operational (NOS)*
- *Link Reset (LR)*
- *Link Reset Response (LRR)*

Класи служб [6-20].

■ Fibre Channel підтримує наступні класи служб (Classes of service, COS).

■ Стандарт FC-PH визначає Класи 1-3, Клас 4 визначений в стандарті FC-PH-2 (у FC-FS-2 встановлений застарілим), Клас 5 запропонований для

ізохронного режиму, але недостатньо стандартизований, Клас 6 визначений в стандарті FC-PH-3, Клас F — в стандартах FC-SW і FC-SW2 [2.1-2.5, 2.11].

■ **Клас 1 — Acknowledged Connection Service (виділені канали з підтвердженням)**. Між двома пристроями через комутатор або фабрику встановлюється виділене з'єднання. Приймаючий пристрій відправляє на передавальний пристрій підтвердження прийому кожного кадру. З'єднання залишається відкритим до тих пір, поки передача даних не буде завершена. Час встановлення з'єднання становить декілька мікросекунд. Канал, що надається, зазвичай дуплексний, хоча з потреби можлива організація сімплексного (наприклад, якщо необхідно одночасно передавати дані одному вузлу і приймати від іншого).

■ Пристроям доступна вся його пропускна спроможність. Використовується крізне управління потоком. Гарантується висока швидкість обміну і правильний порядок прийому кадрів. Ідеально підходить для додатків, що працюють з великими об'ємами даних, — наприклад, системи моделювання або обробки відео. Якщо пропускна спроможність не використовується повністю даним застосуванням, вона все одно недоступна для інших застосувань, поки з'єднання не буде закрито, оскільки спроби з'єднання з таким портом відкидатимуться з видачею сигналу «зайнято». Тому, в стандарті рекомендується закривати з'єднання даних для передачі. В цьому випадку доступна максимальна пропускна спроможність. Основний недолік - неможливість роботи між собою портів з різною швидкістю роботи. Стандартізованниє в FC-PH-2 однонаправлена передача, буферизація класу 1 і Camp on, починаючи з FC-FS, вважаються застарілими.

■ **Клас 2 — Acknowledged Connectionless Service (передачі без організації з'єднання з підтвердженням)**.

■ Кожен кадр комутується незалежно від останніх, кінцевий порт може одночасно передавати і приймати дані від декількох вузлів, при цьому канал між двома взаємодіють не виділяється (по суті, відбувається

мультиплексування комутатором трафіку).

- Кожен кадр підтверджується приймаючим пристроєм.
- Кадри можуть доставлятися по різних маршрутах, тобто впорядкована доставка кадрів в даному класі не гарантована, впорядковування послідовності кадрів здійснюється на рівні FC-2.
- Утилізація доступної смуги пропускання нижча, ніж в Класі 1, оскільки включаються механізми регулювання потоку на покадровій основі.

Клас 3 — Unacknowledged Connectionless Service, іноді називається Datagram Connectionless Service (передачі без організації з'єднання і без підтвердження).

Аналогічний класу 2 за винятком того, що немає підтвердження доставки. Пропускна спроможність у відсутність помилок, через відсутність підтверджень, трохи (від 0% в більшості випадків до 3% в гіршому для класу 2 випадку) збільшується в порівнянні з класом 2, але гарантій доставки немає, впорядкована доставка кадрів не гарантована. Впорядковування послідовності кадрів здійснюється на рівні FC-2, а запит на повторну передачу втрачених кадрів здійснюється протоколами верхніх рівнів. Відповідно, у разі помилок передачі, а також якщо кадр відхиляється або ресурс зайнятий, то кадр втрачається, і підключаються протоколи верхніх рівнів. Пропускна спроможність падає, оскільки у протоколів верхніх рівнів час реакції і тайм-аути істотно вищі, ніж на рівні FC-2. При цьому, для протоколів реального часу, затримка з повтором може бути такій, що передавана інформація вже застаріла. Використовується для організації багатоадресних і широкомовних розсилок, застосовується також в системах масової пам'яті. Найпоширеніший клас комутованих FC-сетей, оскільки найпростіший в реалізації і оскільки найпоширеніші протоколи верхніх рівнів SCSI і IP працюють саме в цьому класі.

- **Клас 4 — Fractional Bandwidth Connection-oriented Service (з'єднання з дробовою смугою пропускання) між N_Ports**. Схожий з Класом

1, оскільки теж припускає встановлення з'єднання, підтвердження доставки, фіксовану затримку, дотримання порядку кадрів. З'єднання між портами встановлюється у вигляді віртуального каналу із смугою пропускання, достатньою для надання послуг з передбаченою якістю (QOS, що включає гарантовані смугу пропускання і максимальну затримку). Такий віртуальний двонаправлений канал полягає двох однонаправлених віртуальних з'єднань (Virtual Circuit, VC), причому на кожному VC можуть забезпечуватися різні QOS. Кожен N_port може встановлювати декілька таких з'єднань (до 254). Використовується для критичних до часу доставки даних — наприклад, видео- і аудіопотоків.

- **Клас 5 — Isochronous Service (ізохронне з'єднання)**. Не стандартизований. Призначений для додатків, що вимагають негайної доставки даних без проміжної буферизації.

- **Клас 6 — Unidirectional Connection Service (однонаправлене з'єднання)**. Аналогічний Класу 1, але є виключно однонаправленим. Використовується для широкомовних і багатоадресних розсилок через відповідний сервер. *N_port* може зажадати з'єднання Класу 6 на одне або декілька пристроїв (портів). Встановлене з'єднання існує, поки ініціатор в явному вигляді не закриє його. Розроблений для доставки трафіку реального часу (наприклад, аудіо і відео).

- **Змішаний клас — Intermix** — є підвидом класу 1. Дозволяє передавати кадри класу 2 або 3 в ті моменти, коли додаток першого класу не займає канал, причому кадри класів 2 або 3 необов'язково повинні бути адресовані тому ж одержувачеві, що і у класу 1. Був спеціально розроблений з метою частково усунути блокування фабрики передачами першого класу.

- **Клас F** — використовується комутаторами для управління і передачі службовій інформації, передача йде без встановлення з'єднання по Inter Switch Links (ISL) між E_ports.

Сфери застосування Fibre Channel [2.1-2.5, 2.12].

SAN-Світч Qlogic з підключеними до нього *FC* (оптичними) конекторами (рис. 2.2).

Рис. 2.2. SAN-Світч Qlogic з підключеними до нього FC (оптичними) конекторами

Fibre Channel широко застосовується для створення мереж зберігання даних. Завдяки високій швидкості передачі даних, малій затримці і розширюваності практично не має аналогів в цій області. Проте, останніми роками, область його застосування поступово переміщається в сегмент високопродуктивних систем і рішень, а бюджетний сегмент з успіхом освоюється недорогими рішеннями SCSI на базі *Gigabit Ethernet і 10G Ethernet*. Намітилася також тенденція до перенесення транспортного рівня протоколу FC в той же *Gigabit Ethernet і 10G Ethernet* за допомогою протоколів *FCoE і FCIP*.

2.7 NDP протокол виявлення сусідів

Протокол виявлення сусідів (*Neighbor Discovery Protocol, NDP*) - протокол з набору протоколів *TCP / IP*, який використовується спільно з *IPv6* .

Він працює на мережевому рівні Моделі Інтернету (*RFC 1122*) і відповідальний за автонастройку адреси кінцевих і проміжних точок мережі, виявлення інших вузлів на лінії, визначення адреси інших вузлів канального рівня, виявлення конфлікту адрес, пошук доступних маршрутизаторів і *DNS*-серверів, визначення префікса адреси і підтримки доступності інформації про шляхи до інших активних сусідніх вузлів (*RFC 4861*).

Цей протокол встановлює п'ять різних типів пакета *ICMPv6* для виконання функцій *IPv6*, схожих з *ARP*, *ICMP,IRDP* і *R outer Redirect* протоколів для *IPv4* [2.1-2.5, 2.13]. Проте, він надає багато поліпшень щодо *IPv4* аналогів (*RFC 4861*, секція 3.1). Наприклад, він включає *NUD*, який підвищує надійність доставки пакетів в присутності проблемних маршрутизаторів або підключень, або мобільних пристроїв.

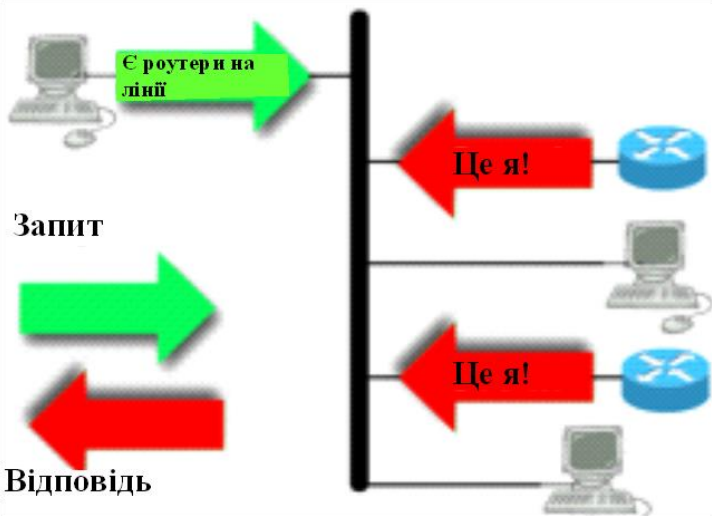


Рис. 2.3. Процес виявлення маршрутизатора

NDP встановлює наступні п'ять типів пакета ICMPv6:

- Запит на доступність маршрутизаторів
- Відповідь маршрутизатора
- Запит доступних сусідів
- Відповідь сусіда
- Перенаправлення

Ці повідомлення використовуються для забезпечення наступної функціональності:

■ *Виявлення маршрутизатора*: вузол може розмістити маршрутизатор, що знаходиться на підключеній лінії.

■ *Виявлення підмережі*: вузли можуть виявляти працюючі підмережі для підключених ліній.

■ *Виявлення параметрів:* вузли можуть запитувати параметри лінії (наприклад, розмір MTU).

- *Автоматична настройка адреси:* конфігурація адрес мережевих інтерфейсів.
- *Дозвіл адреси:* робота між IP-адресою і адресами рівня каналу зв'язку.
- *Виявлення наступного переходу:* вузли можуть знаходити наступний на шляху пакета маршрутизатор.
- *Виявлення недоступності сусіда (NUD):* визначення того, що сусід більш недоступний на лінії.
- *Виявлення конфлікту адрес (DAD):* вузли самі можуть визначати, чи зайнятий адрес.
- *Перенаправлення:* маршрутизатор може інформувати вузол про інших найкращих маршрутизаторів для початку шляху пакета.
- *Рекурсивний DNS-сервер (RDNSS) і список пошуку DNS (DNSSL)* призначається через параметри відгуку маршрутизатора (RA). Це нова функція, і підтримується не усім програмним забезпеченням.

Є деякі уразливості, а саме:

- Деякі маршрутизатори уразливі при роботі з протоколом NDP.
- Найчастіше, маршрутизатори мають менше доступних адрес для NDP, ніж є в підмережі IPv6 (зазвичай 2 ^ 64 або більше, для підтримки SLAAC).

ЛІТЕРАТУРА

2.1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципи, технологии, протоколы: Учебник для вузов. – СПб.: Питер, 2001. – 672 с.

2.2. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципи, технологии, протоколы: Учебник для вузов. 3-е изд. – СПб.: Питер, 2006.– 958 с.

2.3. Компьютерные сети. 4-е изд./ И. Таненбаум. – СПб.: Питер, 2003. – 992 с.

2.4. Кулаков Ю.О., Луцький Г.М. Комп'ютерні мережі. Підручник. – Київ.: «Юніор», 2005. – 396 с.

2.5. Галкин В. А., Григорьев Ю. А. Телекоммуникации и Сети. — М.: МГТУ им. Н. Э. Баумана, 2003. С. 608. ISBN 5-7038-1961-X.

2.6. Friend, George E.; John L. Fike, H. Charles Baker, John C. Bellamy (1988). Understanding Data Communications (вид. 2nd). Indianapolis: Howard W. Sams & Company. ISBN 0-672-27270-9.

2.7. Stallings, William (2004). Data and Computer Communications (вид. 7th). Upper Saddle River: Pearson/Prentice Hall. ISBN 978-013100-6812.

2.8. S. Tanenbaum, Andrew (2005). Computer Networks (вид. 4th). 482,F.I.E., Patparganj, Delhi 110 092: Dorling Kindersley(India)Pvt. Ltd.,licenses of Pearson Education in South Asia. ISBN 81-7758-165-1.

2.9. RFC 4861, Протокол обнаружения соседей для протокола IP версии 6 (IPv6), T. Narten et al. (сентябрь 2007)

2.10. RFC 2461, Протокол обнаружения соседей для протокола IP версии 6 (IPv6), T. Narten, декабрь 1998

2.11. RFC 6106, IPv6 Параметры ответа маршрутизатора для настройки DNS, J. Jeong (Ed.), S. Park, L. Beloeil, S. Madanapalli (ноябрь 2010)

2.12. http://inconcepts.biz/~jsw/IPv6_NDP_Exhaustion.pdf

2.13. Operational Neighbor Discovery Problems and Enhancements. draft-gashinsky-v6nd-enhance-00

3 ПРОТОКОЛИ МЕРЕЖЕВОГО РІВНЯ

3.1. IP-протокол адресації пакетів

IP-протокол (Internet Protocol) — найпоширеніша реалізація ієрархічної схеми мережевої адресації. Використовуваний в мережі Інтернет, протокол відповідає за адресацію пакетів, але не відповідає за встановлення з'єднань, не є надійним і дозволяє реалізувати тільки негарантовану доставку даних. Термін «протокол без встановлення з'єднань» (connectionless) означає, що протокол для взаємодії не потребує виділеного каналу, як це відбувається під час телефонної розмови і не існує процедури виклику перед початком передачі даних між мережевими вузлами [3.1].

Протокол IP вибирає найефективніший шлях з числа доступних на основі рішень прийнятих протоколом маршрутизації. Відсутність надійності і негарантована доставка не означає, що система працює погано або ненадійно, а вказує лиш на те, що протокол IP не докладає ніяких зусиль, щоб перевірити чи був пакет доставлений за призначенням. Ці функції делеговані протоколам транспортного та вищих рівнів. Транспортний рівень також відповідає за збірку пакетів у повідомлення в потрібній послідовності.

Інформація, проходячи вниз по рівням моделі OSI, на кожному рівні певним чином обробляється протоколами цього рівня. На малюнку можна побачити, як на мережному рівні дані інкапсулюються всередину пакетів, іноді названих дейтаграммами (датаграмами).

Протокол IP розпізнає формат заголовка пакета (адресну частину та іншу службову інформацію включно), але ніяким чином не аналізує і не піклується про фактичні дані. Він приймає і передає будь-які дані, передані протоколами верхніх рівнів [3.2].

Пересилка пакетів і комутація всередині маршрутизатора нижче.

Заголовок і хвіст фрейму відкидаються і замінюються новими щоразу при проходженні пакетом маршрутизатора.

Причина в тому, що блоки інформації другого рівня (фрейми) використовуються для локальної доставки пакетів, у той час як блоки третього рівня (пакети) призначені для наскрізної передачі даних згідно з схемою маршрутизації.

Ethernet-фрейми другого рівня призначені для роботи всередині широкомовних доменів з призначеними кожному мережному пристрою MAC-адресами [3.1].

Фрейми другого рівня інших типів, як наприклад послідовні двохточкові з'єднання або Frame relay розподілених мереж (мереж WAN), використовують свою власну схему адресації другого рівня. Принциповим є те, що незалежно від використовуваної схеми адресації другого рівня, всі вони розроблені для використання всередині одного широкомовного домену другого рівня. При проходженні крізь пристрій третього рівня інформація другого рівня змінюється.

Із фрейму, що приходить на інтерфейс роутера, витягається MAC-адреса і перевіряється, адресований цей пакет безпосередньо якомусь вузлу чи інтерфейсу, чи він є широкомовним (ця процедура виконується всіма пристроями всередині домену колізій).

В будь-якому з цих випадків пакет буде оброблено, в іншому — відкинуто, оскільки він адресований іншому вузлу в домені колізій.

Таким чином, домен колізій — розподілене середовище передачі даних, в якому пристрої працюють в режимі конкуренції.

На основі значення, що зберігається в полі контрольної суми, за допомогою циклічного збиткового коду (Cyclical Redundancy Check — CRC), що був вилучений з хвоста отриманого фрейму, перевіряється чи були дані пошкоджені [3.1-3.3].

Якщо перевірка дає позитивний результат — фрейм відкидається.

У випадку негативного результату, заголовок і хвіст фрейму відкидаються і пакет передається на третій рівень.

Далі виконується перевірка, чи було пакет адресовано маршрутизатору, чи потрібна подальша маршрутизація на шляху до місця призначення.

Пакети, адресовані роутеру як IP-адреса отримувача, мають адресу одного з його інтерфейсів. У таких пакетів видаляється заголовок і вони передаються на четвертий, транспортний рівень.

Якщо пакет потребує маршрутизації, IP-адреса пункту призначення пакету порівнюється з записами в таблиці маршрутизації.

Якщо знайдено точну відповідність або існує стандартний маршрут, — пакет відправляється на інтерфейс, що вказано в таблиці маршрутизації.

Коли пакет комутується на вихідний інтерфейс, нове значення CRC додається у хвіст фрейму і, в залежності від типу інтерфейсу (Ethernet, Frame relay або послідовний), пакету додається відповідний заголовок. Після чого фрейм пересилається в інший широкомовний домен на шляху до кінцевого пункту призначення.

Структура IP-пакету. IP-пакети складаються з даних верхнього рівня та IP-заголовку. За специфікацією протоколу, пакет має бути не більший за 65535 бітів (з заголовком та даними включно) [3.1, 3.4].

Версія (Version) — 4-бітове поле, що описує використовувану версію протоколу IP. Всі пристрої зобов'язані використовувати протокол IP однієї версії, пристрій що використовує іншу версію буде відкидати пакети.

Довжина IP-заголовку (IP header Length — HLEN) — 4-бітове поле, що описує довжину заголовку пакету в 32-бітових блоках. Це значення — це повна довжина заголовку з врахуванням двох полів змінної довжини.

Тип обслуговування (Type of Service — TOS) — 8-бітове поле, що вказує на ступінь важливості інформації, що привласнена протоколом верхнього рівня.

Загальна довжина (Total Length) — 16-бітове поле, що описує довжину пакету в байтах, із заголовком та даними включно. Для того щоб вирахувати довжину блока даних, потрібно від повної довжини відняти значення поля HLEN.

Ідентифікація (Identification) — шістнадцятибітове поле, що зберігає ціле число, яке описує даний пакет. Це число являє собою послідовний номер.

Прапорці (Flags) — 3-бітове поле, в якому два молодших біта контролюють фрагментацію пакетів. Перший біт визначає чи було пакет фрагментовано, а другий чи є цей пакет останнім фрагментом в серії фрагментів.

Зміщення фрагментації (Fragment Offset) — 13-бітове поле, що допомагає зібрати разом фрагменти пакетів. Це поле дозволяє використовувати 16 бітів в сумі для прапорів фрагментації.

Час життя (Time-to-Live — TTL) — 8-бітове поле — лічильник, в якому зберігаються послідовно зменшуване значення кількості пройдених вузлів (роутерів, що їх ще іноді в цьому випадку називають хопами (hops)) на шляху до місця призначення. У випадку коли лічильник пройдених хопів дорівнюватиме нулю — пакет буде відкинуто, таким чином попереджується нескінченна циклічна пересилка пакетів [3.1, 3.4, 3.5].

Протокол (Protocol) — 8-бітове поле, що вказує на те, який протокол верхнього рівня отримає пакет, після завершення обробки IP-протоколом. Наприклад TCP або UDP.

Контрольна сума заголовку (Header Checksum) — 16-бітове поле, що допомагає перевірити цілісність заголовку пакету.

IP-адреса відправника (Source IP address) (адресант, сорс, відправник) — 32-бітове поле, що зберігає IP-адресу вузла-відправника.

IP-адреса отримувача (Destination IP adress) (адресат, дест, отримувач) — 32-бітове поле, що зберігає адресу вузла призначення (отримувача).

Опції (Options) — поле змінної довжини, що дозволяє протоколу IP реалізувати підтримку різних опцій, зокрема засобів безпеки.

Підкладка (Padding) — поле, що використовується для вставки додаткових нулів, для гарантування кратності IP-заголовку 32 бітам (таб. 3.1).

Дані (Data) — поле змінної довжини (64 Кбіт макс.), що зберігає інформації для верхніх рівнів

Таблиця. 3.1. Структура IP-пакету в бітових блоках

<table>
<tr><th>Біти 0-3</th><th>4-7</th><th>8-15</th><th>16-18</th><th>19-23</th><th>24-31</th></tr>
<tr><td>Версія</td><td>HLEN</td><td>Тип обслуговування</td><td colspan="3">Загальна довжина</td></tr>
<tr><td colspan="3">Ідентифікація</td><td>Прапорці</td><td colspan="2">Зміщення фрагментації</td></tr>
<tr><td colspan="2">Час життя</td><td>Протокол</td><td colspan="3">Контрольна сума заголовку</td></tr>
<tr><td colspan="6">IP-адреса відправника</td></tr>
<tr><td colspan="6">IP-адреса отримувача</td></tr>
<tr><td colspan="5">Опції</td><td>Додаток</td></tr>
<tr><td colspan="6">Дані (65535 мінус заголовок)</td></tr>
<tr><td colspan="6">...</td></tr>
</table>

IP-пакет складається з даних протоколу верхнього рівня і заголовку, що має описану вище структуру. Хоча основною частиною заголовку є адреси відправника і призначення, саме інші частини заголовку роблять протокол таким надійним і гнучким. Інформація, що зберігається в полях заголовку задає дані пакету і призначена для протоколів верхніх рівнів.

3.2. IPv4 – четверта версія IP протоколу

IPv4 - це четверта версія протоколу IP (Internet Protocol), який на сьогоднішній день є основним та обслуговує більшу частину мережі Інтернет. IPv4 протокол встановлює правила функціонування комп'ютерних мереж за принципом обміну пакетами. Це протокол низького рівня, який відповідає за встановлення з'єднання між вузлами мережі на основі IP-адрес[3.1-3.6].

Адреси вузлів в мережі, згідно з протоколом IPv4 мають довжину 32 біт, що дає в сукупності 232 = 4 294 967 296 можливих адрес. Але не всі адреси використовуються для глобального простору (Інтернет), частину адрес виділено для спеціальних потреб, наприклад, для організації локальних (приватних) мереж, віртуальних мережевих інтерфейсів, використовуються в тестових цілях, є спеціальними адресами і так далі.

Представлення IPv4 адрес. IPv4 адреси, як правило, записуються у вигляді чотирьох десяткових чисел від 0 до 255 розділених символом "." (крапка), наприклад, мінімальна можлива адреса - 0.0.0.0, максимальна - 255.255.255.255. Число від 0 до 255, як правило, в комп'ютерних системах вимагає для зберігання 1 байт або 8 біт інформації, таким чином 8 * 4 = 32 біта або 4 байти, що відповідає заявленій довжині адреси.

- Хоча можуть бути використані і інші представлення адрес, залежно від необхідності (на прикладі адреси 123.45.67.89):
- *З крапкою:*
  - **Десяткове:** 123.45.67.89
  - **Двійкове:** 01111011.00101101.01000011.01011001
  - **Шістнадцяткове:** 0x7B.0x2D.0x43.0x59
  - **Вісімкове:** 0173.0055.0103.0131
- *Без крапки:*
  - **Десяткове:** 2066563929

- **Двійкове:** 01111011001011010100001101011001
- **Шістнадцяткове:** 0x7B2D4359
- **Вісімкове:** 017355103131

Безкласова адресація (CIDR). Спочатку адресація в IP-мережах здійснювалася за класовим принципом (існували класи, які розподіляли адресний простір на великі блоки). Проте дана схема виявилася непрактичною і сьогодні в Інтернет використовується безкласова адресація, відома як Classless Inter-Domain Routing, або скорочено CIDR [3.1-3.5].

У цілому, CIDR дозволяє описувати блоки IP-адрес для Інтернет-підмереж. Так, стандартним вважається запис CIDR у вигляді IP-адреси, наступного за нею символу "/" та число, що означає бітову маску підмережі, наприклад, 12.13.14.0/24.

Число 24 в даному випадку буде означати кількість старших бітів у масці підмережі. Так як IP-адреса складається з 32 біт, але маскою є старші 24, це означає, що для всіх можливих адрес в мережі залишається 32 - 24 = 8 біт. Тобто 28 = 256 можливих. Або, якщо наша маска була б 23 біта а не 24, то для адрес залишилося б 9 біт = 29 = 512 можливих, і навпаки, якщо маска буде 25 біт, то для адрес залишиться 232-25 = 27 = 128 можливих. Таким чином, ми можемо описувати мережі, що складаються з різної кількості доступних адрес. Крім того, одна велика мережа може бути всередині знову роздрібнена на кілька менших підмереж, ті в свою чергу можуть бути також розбиті на підмережі і так далі.

Кількість можливих вузлів (хостів) в підмережі завжди мінімум на 2 менше кількості всіх можливих адрес (за деякими винятками, коли адресують мережі /32 та /31). Обумовлено це тим, що перша адреса резервується, як ідентифікатор мережі, а остання є широкомовною.

Згідно з характеристиками, визначеними різними стандартами, що відносяться до протоколу IPv4, існують такі спеціальні адреси (таб. 3.2):

Таблиця 3.2. Спеціальні адреси протоколу IPv4

Мережа (адреса)	Опис	Стандарт
0.0.0.0/8	Джерело адрес поточної мережі	RFC 5735
10.0.0.0/8	Для організації приватних мереж	RFC 1918
100.64.0.0/10	Для використання в мережі провайдера	RFC 6598
127.0.0.0/8	Інтерфейс комутації всередині хоста	RFC 5735
169.254.0.0/16	Для автоматичного конфігурування (наприклад, за відсутності DHCP)	RFC 3927
172.16.0.0/12	Для організації приватних мереж	RFC 1918
192.0.0.0/24	Для спеціального призначення (зарезервовано IETF)	RFC 5735
192.0.2.0/24	Тестова мережа 1, для використання в якості прикладів в документації	RFC 5735
192.88.99.0/24	Для трансляцій з IPv6 в IPv4	RFC 3068
192.168.0.0/16	Для організації приватних мереж	RFC 1918
198.18.0.0/15	Для тестування продуктивності	RFC 2544
198.51.100.0/24	Тестова мережа 2, для використання в якості прикладів в документації	RFC 5737
203.0.113.0/24	Тестова мережа 3, для використання в якості прикладів в документації	RFC 5737
224.0.0.0/4	Для багатоадресної розсилки	RFC 5771
240.0.0.0/4	Зарезервовано для можливих потреб у майбутньому	RFC 1700
255.255.255.255	Широкомовна адреса	RFC 919

Тобто, як видно, з усього адресного простору IPv4 частина адрес використовується для спеціальних потреб, а це означає, що для потреб реальних вузлів мережі вільних адрес залишається навіть менше, ніж теоретично визначено IPv4 протоколом. На сьогоднішній день адресний простір IPv4 практично повністю вичерпано, всі вільні адреси використані для спеціальних потреб або роздані різним організаціям для потреб їх мереж.

Тому останнім часом здійснюється поступовий перехід на новий протокол IPv6.

3.3. IPv6 – шоста версія IP протоколу

IPv6 (Internet Protocol version 6) — нова версія IP-протоколу — IP версії 6. Розробка протоколу IPv6 почалася 1992 року, а з 2003 р. його підтримку забезпечують виробники більшості телекомунікаційного устаткування (корпоративного рівня). IPv6 — новий крок у

розвитку Інтернету. Цей протокол розроблено з урахуванням вимог до Глобальної мережі, що постійно зростають. 3 лютого 2011 року IANA виділила останні п'ять блоків IP-адрес /8 (IPv4) [3.1-3.6].

Найбільш суттєва різниця між IPv4 та IPv6 полягає в тому, що раніше на інтернет-адресу виділяли 4 байти (32 біти), що відповідає стандартній на сьогодні чотириблоковій адресі IP, а протокол IPv6 виділяє на адресу 16 байтів (128 бітів). Це відповідає 340 секстильйонам адрес (3,4x1038) або по 5x1028 адрес на кожну людину.

У квітні 2009 у мережі UA-IX запущено процес перевірки протоколу IPv6. Серед перших компаній, що ухвалили рішення про участь в тестуванні, — «ТопНЕТ» і «Датагруп». Вони встановили IPv6 BGP-з'єднання з маршрутизатором UA-IX і здійснили обмін маршрутною інформацією між ними. У квітні 2011 розпочалось масове впровадження IPv6 серед домашніх користувачів інтернет [3.1, 3.4].

Наприкінці 1980-х стала очевидною нестача адресного простору Інтернет. На початку 1990-х, навіть після введення безкласової адресації, виявилось, що однієї економії та використання NAT'у буде замало, щоб запобігти вичерпання адресного простору, і необхідна зміна адресації. Крім того, накопичилась певна кількість пропозицій щодо усунення недоліків наявної моделі Інтернет. Наприкінці 1992 року IETF оголосила конкурс на створення протоколу Інтернет наступного покоління (IP Next Generation — IPng).

25 липня 1994 року IETF ствердила модель IPng з утворенням кількох робочих груп IPng. У 1996 було створено серію RFC, що визначали новий протокол Інтернет. Оскільки версія 5 вже була раніше призначена експериментальному протоколу передачі мультимедійних потоків, новий протокол отримав версію 6.

Переведення на IPv6 почало виконуватись всередині Google з 2008 року. У специфікації стандарту мобільних мереж LTE вказана обов'язкова

підтримка IPv6. Проблемою для впровадження IPv6 є те, що не всі операційні системи повністю підтримують протокол IPv6.

Розширення адресного простору скасовує необхідність використання NAT, оскільки на кожну людину припадає близько 3*108 унікальних адрес [3.4].

Принцип призначення хосту IPv6 адреси є ієрархічним. Мінімальний розмір підмережі — /64 (264). Молодша частина адреси (64 біти) використовується як унікальний ідентифікатор користувача, наступна частина визначає підмережу всередині оператора зв'язку, далі йде ідентифікатор самого оператора. Такий підхід значно спрощує маршрутизацію.

З IPv6 вилучено кілька функцій, що ускладнюють роботу маршрутизаторів:

Маршрутизатори більше не розбивають (фрагментують) пакет на частини (розбиття пакета можливо тільки на боці передавача). Відповідно, оптимальний MTU має визначатися за допомогою Path MTU discovery. Для покращення роботи протоколів, що потребують низького рівня втрати пакетів, мінімальний MTU збільшено до 1280 байт. Інформацію про фрагментацію пакетів перенесено з основного заголовка в розширені;

Зникла контрольна сума. Оскільки канальні (Ethernet) та транспортні (TCP) протоколи також перевіряють коректність пакета, контрольна сума на рівні IP вважається зайвою. Крім того, кожен маршрутизатор зменшує hop limit на одиницю, що призводить до потреби у перерахуванні суми в IPv4.

Незважаючи на суттєве збільшення розміру адреси IPv6, завдяки цим покращенням основний заголовок пакета збільшився лише у 2 рази: з 20 до 40 байтів [3.1, 3.6].

Покращення IPv6 у порівнянні з IPv4:

В надшвидкісних мережах можлива підтримка надвеликих пакетів (джамбограм) — до 4 гігабайт;

Time to Live перейменовано в Hop limit;

З'явились відмітки потоків та класи трафіку;

З'явилась багатоадресна передача;

Протокол IPsec з рекомендованого перетворився на обов'язковий.

У момент ініціалізації мережевого інтерфейсу йому призначується локальна IPv6-адреса, з префіксом fe80::/10, у молодшій частині адреси розміщується ідентифікатор інтерфейсу. Ідентифікатором інтерфейсу часто слугує 64-бітний розширений унікальний ідентифікатор EUI-64[en], що найчастіше формується з MAC адреси. Локальна адреса дійсна тільки в межах мережевого сегмента канального рівня, і використовується, в основному, для обміну інформаційними ICMPv6 пакетами.

Для більшого адміністративного контролю може бути використаний DHCPv6[en], що дозволяє адміністратору маршрутизатора призначати вузлам конкретні адреси [3.2, 3.8, 3.9].

Для отримання інших адрес вузол може запросити інформацію про налаштування мережі у маршрутизаторів за допомогою ICMPv6 повідомлення «Router Solicitation». Цей запит відсилається на групову (multicast) адресу маршрутизаторів. У відповідь маршрутизатори відсилають ICMPv6 повідомлення «Router Advertisement», що може містити інформацію про префікс мережі, адресу шлюзу, адреси рекурсивних серверів DNS, MTU та багато інших параметрів. Поєднуючи мережевий префікс та ідентифікатор інтерфейсу, вузол отримує нову адресу. Для захисту персональних даних ідентифікатор інтерфейсу може бути замінений на псевдовипадкове число.

Адреси IPv6 мають 128 бітів. Дизайн адресного простору IPv6 реалізує зовсім іншу філософію дизайну, ніж в IPv4, в якій підмережа використовувалася для підвищення ефективності використання малого адресного простору [3.1-3.6].

У IPv6 адресний простір вважається досить великим у передбачуваному майбутньому, а локальна підмережа завжди використовує 64 біти для частини що приймає адрес, позначеної як ідентифікатор інтерфейсу, тоді як найважливіші 64 біти використовуються як префікс для маршрутизаторів.

Ідентифікатор унікальний лише в підмережі, до якої підключений хост. IPv6 має механізм автоматичного виявлення адреси, так що адресу автоконфігурації завжди створює унікальну передачу.

Введення поля «Відмітка потоку» в протоколі IPv6 дозволяє значно спростити процедуру маршрутизації однорідного потоку пакетів. Потік — це послідовність пакетів, що надсилаються відправником певному адресату. При цьому припускається, що всі пакети даного потоку мають бути оброблені певним чином. Характер даної обробки задається додатковими заголовками [3.2-3.7].

Припускається існування декількох потоків між відправником та отримувачем. Відмітка потоку призначається вузлом-відправником шляхом генерації псевдовипадкового 20-бітного числа. Всі пакети одного потоку мають містити однакові заголовки, що оброблюються маршрутизатором.

При отриманні першого пакета з відміткою потоку маршрутизатор аналізує додаткові заголовки, виконує певні операції відповідно до цих заголовків та запам'ятовує результати обробки (адресу наступного вузла, опції заголовку переходів, переміщення адрес у заголовку маршрутизації тощо) в локальному кеші. Ключем для такого запису є комбінація адреси відправника та відмітки потоку. Наступні пакети з тією самою комбінацією адреси відправника та відмітки потоку обробляються з урахуванням інформації кешу без детального аналізу усіх полів заголовка [3.3, 3.9].

Час життя запису у кеші становить не більше 6 секунд, навіть якщо пакети цього потоку продовжують надходити. Після видалення запису з кешу при отриманні наступного пакета потоку, пакет обробляється у

звичайному режимі і для нього відбувається формування нового запису в кеші. Слід зауважити, що вказаний час життя потоку може бути явно заданий вузлом відправником за допомогою протоколу керування або опцій заголовку переходів, і може перевищувати 6 секунд.

Пріоритезація пакетів забезпечується маршрутизаторами на основі перших шести бітів поля Traffic Class. Перші три біти визначають клас трафіку, решта бітів визначають пріоритет видалення. Чим більше значення пріоритету, тим вище пріоритет пакета.

В залежності від задач розробники IPv6 рекомендують використовувати наступні коди класу трафіку (таб. 3.3):

Таблиця 3.3. Коди класу трафіку для IPv6

Клас трафіку	Призначення
0	Нехарактеризований трафік
1	Наповнювальний трафік (мережеві новини)
2	Автономний інформаційний трафік (електрона пошта)
3	Резерв
4	Неавтономний масовий трафік (FTP, HTTP, NFS)
5	Резерв
6	Інтерактивний трафік (Telnet, X terminal, SSH)
7	Керівний трафік (BGP, SNMP)

IPv6 адреси показуються як вісім груп по чотири шістнадцяткові цифри, розділених двокрапками [3.8]. Приклад адреси:

2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d

Якщо одна чи більше груп підряд дорівнюють 0000, то вони можуть скорочено записуватись як подвійна двокрапка (::).

Наприклад, 2001:0db8:0000:0000:0000:0000:ae21:ad12 може бути скорочена до 2001:db8::ae21:ad12,

0000:0000:0000:0000:0000:0000:ae21:ad12 — до ::ae21:ad12. Скорочення не дозволяється у випадку, коли адреса містить 2 окремі нульові групи через виникнення невизначеності.

При використанні IPv6-адреси в URL необхідно брати адресу в квадратні дужки:

http://[2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d]/

Якщо потрібно вказати порт, то він пишеться після дужок:

http://[2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d]:8080/

Опис полів:

- *Version*: версія протоколу; для IPv6 це значення дорівнює 6 (значення в бітах — 0110).
- *Traffic class*: пріоритет пакета (8 бітів). Це поле містить два параметри. Старші 6 бітів використовуються DSCP для класифікації пакетів. Решта два біти використовуються ECN для контролю перевантаження.
- *Flow label*: відмітка потоку (див. відмітки потоків).
- *Payload length*: на відміну від поля *Total length* протоколу IPv4 дане поле не включає заголовок пакета (16 бітів). Максимальний розмір, що визначається розміром поля, — 64 Кбайти. Для пакетів більшого розміру використовується *Jumbo payload*.
- *Next header*: вказує тип розширеного заголовка (*IPv6 extension*), що розміщений одразу за основним. В останньому розширеному заголовку поле *Next header* вказує тип транспортного протоколу (TCP, UDP і т. д.)
- *Hop limit*: аналог поля *time to live* в IPv4 (8 бітів).
- *Source Address* і *Destination Address*: адреси відправника та отримувача відповідно; по 128 бітів.

3.4. Порівняння IPv4 та IPv6

Проведено аналіз параметрів IPv4 та IPv6 в таблиці 3.4 [3.1-3.7]

Таблиця 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Адреси	Довжина - 32 біта (4 байта). Адреса складається з адреси мережі і адреси хоста. Довжина цих компонентів залежить від класу адреси. Адреси діляться на класи A, B, C, D і E. Клас адреси визначається декількома початковими бітами адреси. Загальна кількість адрес IPv4 складає 4 294 967 296. В текстовому вигляді адреса IPv4 записується як nnn.nnn.nnn.nnn, де 0 <= nnn <= 255, а кожна буква nпредставляет десяткову цифру. Незначущі нулі можна не вказувати. Максимальна довжина адреси становить 15 символів, без урахування маски.	Довжина - 128 біт (16 байт). Зазвичай перші 64 біта задають номер мережі, а другі 64 біта - номер хоста. Часто в якості номера хоста або його компонента в адресі IPv6 виходить на основі MAC-адреси або іншого ідентифікатора інтерфейсу. У подсетях з деякими префіксами архітектура IPv6 складніше архітектури IPv4. Кількість адрес IPv6 в 1028 (79 228 162 514 264 337 593 543 950 336) разів більше числа адрес IPv4. В текстовому вигляді адреса IPv6 записується як xxxx: xxxx: xxxx: xxxx: xxxx: xxxx: xxxx: xxxx, де кожна буква x - це шістнадцяткова цифра, що представляє 4 біта. Незначущі нулі можна не вказувати. В текстовому форматі замість будь-якого числа нулів в адресі можна вказати подвійна двокрапка (: :). Наприклад, адреса :: ffff: 10.120.78.40 представляє собою адресу IPv6, перетворений в IPv4.

Продовження таблиці 3.4 Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Розташування адрес	Спочатку адреси розподілялися по класах мереж. Коли число вільних адрес почала стрімко зменшуватися, адреси були розбиті на більш дрібні групи за допомогою протоколу безкласову міждоменну маршрутизацію (CIDR). Адреси не були рівномірно розподілені між різними організаціями і країнами.	Розподіл адрес поки знаходиться на початковому етапі. Робоча група Internet (IETF) і група, відповідальна за розробку архітектури Internet (IAB), рекомендували надати кожній організації, домашнього комп'ютера або пристрою префікс підмережі розміром / 48 біт. У цьому випадку ще 16 біт префікса залишиться для ідентифікатора підмережі. Простір адрес досить велике для того, щоб надати кожному жителю планети власний префікс підмережі довжиною / 48біт.
	Зазвичай цей атрибут задається тільки для адрес IPv4, призначених службою DHCP.	Для адрес IPv6 задається два терміни дії: бажаний і допустимий, причому бажаний термін дії завжди <= допустимого. Після закінчення пріоритетного терміну дії адреса перестає бути вказані в якості IP-адреси відправника для нових з'єднань, якщо доступний настільки ж хороший бажаний адресу. Після закінчення допустимого терміну дії адреса перестає застосовуватися (розпізнаватися) в якості IP-адреси одержувача при прийомі пакетів, або в якості IP-адреси відправника. Для деяких адрес IPv6, наприклад, адрес рівня лінії зв'язку, за замовчуванням встановлений необмежений бажаний і допустимий термін дії

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Маска адреси	Застосовується для відділення адреси мережі від адреси хоста.	Не застосовується
Префікс адреси	Іноді застосовується для відділення адреси мережі від адреси хоста. У деяких випадках вказується в адресу у вигляді суфікса / nn.	Застосовується для визначення префікса підмережі в адресі. Вказується у вигляді суфікса / nnn (максимум 3 десяткові цифри, 0 <= nnn <= 128). Прикладом може служити адреса fe80 :: 982: 2a5c / 10, в якому перші 10 біт представляють префікс підмережі.
Протокол перетворення адрес (ARP)	ARP застосовується в протоколі IPv4 для визначення фізичної адреси, наприклад, адреси MAC або адреси каналу зв'язку, пов'язаного з адресою IPv4.	В IPv6 ці функції є вбудованими. Вони реалізовані в алгоритмах автоматичної настройки адрес і пошуку сусідів, в яких застосовується протокол ICMPv6. У зв'язку з цим протокол ARP6 ні розроблений.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Простір адрес	До звичайних адресами цей термін непридатний. Вважається, що існують діапазони приватних адрес і циклічні адреси. Всі інші адреси розглядаються як глобальні.	В *IPv6* поняття простору адрес вбудовано в архітектуру. Існує два простору звичайних адрес, в тому числі адреси рівня лінії зв'язку і глобальні адреси. Групові адреси належать до 14 різних просторів. Простір, до якого належить адресу, враховується при виборі адреси відправника і одержувача за замовчуванням. Зоною називається екземпляр простору адрес в окремій мережі. Іноді адреси *IPv6* необхідно вказувати разом з ідентифікатором зони. Цей ідентифікатор задається в форматі %***zid***, де ***zid*** - це номер (зазвичай короткий) або ім'я. Ідентифікатор зони вказується після адреси, але до префікса. Наприклад, 2ba :: 1: 2: 14e: 9a9b: c% 3/48.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Типи адрес	Адреси IPv4 діляться на три основних типи: звичайні адреси, групові адреси і широкомовні адреси	Адреси IPv6 поділяються на три основних типи: звичайні адреси, групові адреси і нечіткі адреси
ARPING	ARPING - це основна утиліта TCP / IP для перевірки досяжності систем в локальній мережі.	Аналогічна підтримка для IPv6 доступна за допомогою утиліти NDPING.
Трасування з'єднань	Трасування з'єднань є засобом для збору докладної інформації про пакети TCP / IP і інших пакетах, які приймаються і відправляються системою.	Та ж сама підтримка для IPv6.
Налаштування	Перед тим як нова система зможе встановлювати з'єднання з іншими системами, в ній необхідно виконати настройку, тобто визначити IP-адреси і маршрути.	Налаштування потрібно виконувати тільки для застосування деяких функцій. IPv6 може застосовуватися з будь-яким адаптером Ethernet, а також виконуватися в будь-якому циклічному інтерфейсі. Інтерфейси IPv6 налаштовують самі себе шляхом автоматичної настройки IPv6 без збереження стану. Крім того, інтерфейс IPv6 можна налаштувати вручну. В результаті система зможе підключатися до інших локальних або віддалених систем IPv6, в залежності від типу мережі та маршрутизатора IPv6.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Система імен доменів (DNS)	Додатки застосовують DNS для перетворення імен хостів в IP-адреси за допомогою API сокетів *gethostbyname* (). Крім того, за допомогою DNS додатки можуть перетворити IP-адреси в імена хостів. Для цього застосовується API gethostbyaddr (). У *IPv4* для зворотного перетворення застосовується домен in-addr.arpa.	Та ж сама підтримка для *IPv6*. Для підтримки *IPv6* застосовується тип запису *AAAA* (чотири літери A) і функція зворотного перетворення (перетворення IP-адреси в ім'я). Додаток може вибрати, чи слід приймати адреси *IPv6* від *DNS* та з'єднуватися за допомогою цих адрес. API сокетів gethostbyname () підтримує тільки IPv4. В *IPv6* застосовується новий API getaddrinfo (), за допомогою якого додаток за власним вибором отримує інформацію або тільки для адрес *IPv6*, або для адрес *IPv4 і IPv6*. Для зворотного перетворення в *IPv6* застосовується домен *ip6.arpa.* Якщо з його допомогою перетворення виконати не вдається, то застосовується домен *ip6.int*. (За детальною інформацією зверніться до розділу *API getnameinfo ()* - Отримати інформацію про ім'я для адреси сокета.)

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Протокол динамічної настройки хостів (DHCP)	DHCP застосовується для динамічного отримання IP-адреси та іншої інформації про конфігурацію. IBM і підтримує сервер DHCP для IPv4.	Реалізація DHCP IBM і не підтримує IPv6. Однак, може бути використана реалізація Сервера DHCP ISC.
Протокол передачі файлів (FTP)	FTP служить для прийому і відправлення файлів по мережі.	Та ж сама підтримка для IPv6.
Фрагменти	Якщо пакет занадто великий для його передачі по каналу зв'язку, відправник (хост або маршрутизатор) може розбити його на кілька фрагментів.	В IPv6 пакет можна розбити на пакети тільки на вузлі відправника. Збірка пакета може виконуватися тільки на вузлі одержувача. Застосовується заголовок розширення фрагментації.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Таблиця хостів	Настроюється таблиця, яка пов'язує IP-адреса з ім'ям хоста (наприклад, 127.0.0.1, циклічний адреса). Ця таблиця застосовується програмою перетворення імен сокетов. Ця програма викликається перед зверненням до DNS, або після звернення до DNS, якщо перетворення виконати не вдалося (порядок звернення залежить від пріоритету пошуку імені хоста).	Та ж сама підтримка для IPv6.
Підтримка IBM Navigator for i	IBM Navigator for i - це рішення, яке дозволяє повністю налаштувати TCP / IP.	Та ж сама підтримка для IPv6.
Інтерфейс	Логічний об'єкт, який застосовується в TCP / IP для передачі пакетів. У IPv4 це поняття завжди тісно пов'язане з адресою, а іноді еквівалентно йому. Іноді інтерфейс називається логічним інтерфейсом. Інтерфейси IPv4 запускаються і завершують роботу незалежно один від одного і від TCP / IP. Для запуску і завершення роботи інтерфейсу можна скористатися командами STRTCPIFC і ENDTCPIFC, а також IBM Navigator for i.	Та ж сама підтримка для IPv6.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Протокол керуючих повідомлень *Internet (ICMP)*	Застосовується в протоколі *IPv4* для обміну інформацією про мережу.	У протоколі *IPv6* застосовується для тих же цілей. Однак Протокол керуючих повідомлень Internet версії 6 *(ICMPv6)* підтримує ряд нових атрибутів. Основні типи повідомлень залишилися колишніми, наприклад, цільової вузол недосяжний, луна-запит і відповідь. Нові типи і коди були додані для підтримки функції пошуку сусідів та інших пов'язаних з нею функцій.
Протокол Internet для управління групами (IGMP)	GMP застосовується маршрутизаторами *IPv4* для пошуку хостів, яким повинні доставлятися дані багатоцільовий розсилки. Крім того, він застосовується хостами *IPv4* для сповіщення маршрутизаторів *IPv4* про наявність на хості одержувачів багатоцільовий розсилки.	*IGMP* замінений на протокол *MLD* для *IPv6*. MLD протокол виконує ті ж функції, що і протокол *IGMP* в *IPv4*. Він застосовує протокол *ICMPv6*, в якому передбачено кілька нових типів, призначених для *MLD*.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Заголовок IP	Довжина становить від 20 до 60 байт в залежності від числа додаткових параметрів IP.	Довжина становить рівно 40 байт. У заголовку IP ніякі додаткові параметри не вказуються. Як правило, структура заголовка IPv6 простіше, ніж в IPv4.
Розширені можливості пошуку заголовка IP	Різні додаткові параметри, які можна вказати в заголовку IP (перед заголовком транспортного рівня).	У заголовку IPv6 додаткові параметри не вказуються. Замість них IPv6 додає додаткові заголовки. Такі заголовки можуть містити інформацію AH і ESP (як і в IPv4), а також інформацію про проходження транзитних ділянок, маршруті, фрагменті і одержувача. В даний час IPv6 підтримує кілька заголовків розширення.
Байт протоколу в заголовку IP	Код протоколу транспортного рівня. Прикладом значення може служити ICMP.	Заголовок, який вказується відразу після заголовка IPv6. У ньому задаються ті ж значення, що і в поле протоколу заголовка IPv4. Після цього заголовка може бути вказаний ще ряд додаткових заголовків, формат яких може бути розширений. Наступним може бути вказаний заголовок транспортного протоколу, один з додаткових заголовків або заголовок ICMPv6.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Байт Тип сервісу в заголовку IP	Застосовується протоколом QoS і диференційованими службами для визначення класу потоку даних.	Використовує різні коди для позначення класу потоку даних IPv6. В даний час протокол IPv6 не підтримує поле TOS.
З'єднання LAN	З'єднання LAN застосовується інтерфейсом IP для підключення до фізичної мережі. Існує кілька типів, наприклад, Ethernet. Іноді називається фізичним інтерфейсом, каналом зв'язку або лінією зв'язку.	IPv6 може застосовуватися з будь-яким адаптером Ethernet, крім того, цей протокол підтримується у віртуальній мережі Ethernet між логічними розділами.
Протокол L2TP	Протокол L2TP можна розглядати як віртуальний протокол PPP. Він може застосовуватися при роботі з будь-яку підтримує лінією зв'язку.	Та ж сама підтримка для IPv6.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Циклічна адреса	Циклічний адреса - це інтерфейс з адресою виду 127. *. *. * (Як правило, 127.0.0.1), який може застосовуватися вузлом тільки для відправки пакета самому собі. Відповідний фізичний інтерфейс (опис лінії) називається * LOOPBACK.	Такий же принцип, як і в IPv4. Передбачено єдиний циклічний адреса - 0000: 0000: 0000: 0000: 0000: 0000: 0000: 0001 у яких :: 1 (скорочений варіант). Відповідний віртуальний фізичний інтерфейс називається * LOOPBACK.
Максимальний блок передачі (MTU)	Максимальний блок передачі - це максимальне число байт, яке можна передати по лінії зв'язку певного типу, наприклад, лінії зв'язку Ethernet або модемного лінії. Зазвичай в IPv4 максимальний блок передачі дорівнює 576.	В IPv6 мінімальний розмір MTU становить 1280 байт. Отже, пакети IPv6, розмір яких менше цього обмеження не розбиваються на фрагменти. Для передачі пакетів IPv6 по лінії зв'язку з розміром MTU менше 1280 байт ці пакети повинні розбиватися і збиратися на рівні каналу зв'язку.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
NDPING	Аналогічна підтримка для IPv4 доступна за допомогою утиліти ARPING.	NDPING - це утиліта TCP / IP для перевірки досяжності сусідніх систем для інтерфейсів IPv6.
Netstat	Netstat - це утиліта, що надає інформацію про стан з'єднань, інтерфейсів і маршрутів TCP / IP. Викликається за допомогою IBM Navigator for i і текстового інтерфейсу.	Та ж сама підтримка для IPv6.
Перетворення мережних адрес (NAT)	Одна з основних функцій брандмауера, вбудована в стек протоколів TCP / IP. Для її налаштування використовується IBM Navigator for i.	На даний момент функція NAT не підтримує протокол IPv6. Точніше, в IPv6 функція NAT не потрібна. У зв'язку зі значним розширенням простору адрес в IPv6 не виникає проблема нестачі адрес. Крім того, в цьому протоколі передбачені більш прості засоби зміни адреси.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Таблиця мереж	У IBM Navigator for i - таблиця, яка містить інформацію про імена і IP-адреси мереж. Маска мережі не вказується. Наприклад, хост Network 14 і IP-адреса 1.2.3.4.	Ця таблиця не змінилася в IPv6.
Протокол найкоротшого шляху (OSPF)	OSPF - це протокол маршрутизатора, який у великих мережах автономних систем більш кращий, ніж RIP.	Та ж сама підтримка для IPv6.
Запит на отримання інформації про вузол	Не підтримується.	Зручна мережева утиліта, схожа на утиліту ping. Вона дозволяє запросить у іншого вузла IPv6 його ім'я хоста, звичайну адресу IPv6 або адреса IPv4. В даний час ця утиліта не підтримується.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Фільтрація пакетів	Фільтрація пакетів - це одна з основних функцій брандмауера, вбудована в стек протоколів TCP / IP. Для її налаштування використовується IBM Navigator for i.	Фільтрація пакетів не підтримує IPv6.
Пересилання пакетів	Стек *TCP / IP IBM* і можна налаштувати для пересилання пакетів IP, призначені для віддалених адрес мережі. Зазвичай вхідний і вихідний інтерфейси підключені до різних локальних мереж.	Пересилання пакетів підтримує IPv6 з обмеженнями. Стек *TCP / IP IBM* і не підтримує пошук сусідів як маршрутизатор.
PING	*PING* - це основний засіб TCP / *IP* для перевірки досяжності хоста. Викликається за допомогою *IBM Navigator for i* і текстового інтерфейсу.	Та ж сама підтримка для *IPv6.*

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Двоточковий протокол (PPP)	PPP дозволяє встановлювати комутовані з'єднання за допомогою різних модемів і ліній зв'язку.	Та ж сама підтримка для IPv6.
Обмеження на використання портів	У IBM Navigator for i користувач може вибрати номери портів або діапазони номерів портів TCP або протоколу призначених для користувача дейтаграм (UDP), які дозволено використовувати тільки певного профайлу.	Заборони на порти для протоколу IPv6 збігаються з заборонами для IPv4.
Порти	В TCP і UDP застосовуються різні набори портів, номера яких знаходяться в діапазоні від 1 до 65535.	В IPv6 застосовуються аналогічні порти. Оскільки в цьому протоколі передбачено нове сімейство адрес, число наборів портів збільшилася до чотирьох. Наприклад, передбачено два порти TCP з номером 80, до яких можуть підключатися програми: один з них знаходиться в AF_INET, а другий - в AF_INET6.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Внутрішні і зовнішні адреси	Всі адреси IPv4 є зовнішніми. Виняток становлять три діапазону внутрішніх адрес, що їх організація визначила IETF в документі RFC 1918: 10. *. *. * (10/8), 172.16.0.0 - 172.31.255.255 (172.16 / 12) і 192.168. *. * (192.168 / 16). Внутрішні адреси зазвичай застосовуються в різних організаціях. Такі адреси не розпізнаються в Internet.	В IPv6 застосовується аналогічна структура адрес, але з деякими істотними відмінностями. Адреси діляться на зовнішні і тимчасові (тимчасові адреси раніше називалися анонімними). Додаткова інформація наведена в RFC 3041. На відміну від внутрішніх адрес IPv4, тимчасові адреси розпізнаються в глобальній мережі. Вони застосовуються для іншої мети. Тимчасова адреса приховує ідентифікатор клієнта, який встановлює з'єднання (з міркувань захисту). Термін дії тимчасової адреси обмежений. Така адреса не містить ідентифікатор інтерфейсу, тобто адресу каналу зв'язку (MAC). Як правило, тимчасову адресу не можна відрізнити від звичайного зовнішнього адреси. В IPv6 також є поняття обмеженого адресного простору, пов'язане з передбаченим розподілом адрес

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Таблиця протоколів	У IBM Navigator for i - таблиця, яка містить імена протоколів і пов'язані з ними номера портів. Наприклад: UDP, 17. За замовчуванням в таблиці є записи для наступних протоколів: IP, TCP, UDP, ICMP.	Ця таблиця може застосовуватися в IPv6 без змін.
Quality of service (QoS)	Quality of service дозволяє задати пріоритет пакетів і пропускну здатність для додатків TCP / IP.	Та ж сама підтримка для IPv6.
Зміна адреси	Зміна адреси виконується вручну або за допомогою DHCP. Зміна адрес комп'ютерів в мережі організації являє собою досить трудомісткий процес, який рекомендується виконувати лише в разі крайньої необхідності	Зміна адреси - це важлива вбудована функція протоколу IPv6, яка в значній мірі виконується автоматично, особливо з префіксом / 48.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Маршрут	Один або кілька IP-адрес, пов'язаних з парою значень, яка включає в себе ім'я фізичної інтерфейсу і IP-адреса наступного транзитного вузла. Якщо адреса одержувача пакета IP входить в зазначену групу адрес, то цей пакет пересилається вказаною транзитному вузлу по заданій лінії зв'язку. Маршрути IPv4 пов'язані з інтерфейсом IPv4, а значить, і з адресою IPv4. Маршрут за замовчуванням називається * DFTROUTE.	Принципово аналогічно IPv4. Є одна істотна відмінність: маршрути IPv6 пов'язані з фізичним інтерфейсом (каналом зв'язку, наприклад, ETH03), а не з логічним інтерфейсом. Одна з причин зв'язку маршруту з фізичним інтерфейсом полягає в тому, що в IPv6 і в IPv4 застосовуються різні алгоритми вибору адреси відправника.
Протокол інформації про маршрутизації (RIP)	RIP - протокол маршрутизації, який підтримується демоном routed.	В даний час протокол RIP не підтримує IPv6.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Таблиця служб	У IBM і - таблиця, яка містить імена служб і пов'язані з ними номера портів і імена протоколів. Наприклад, ім'я служби FTP, порт 21, TCP і протокол призначених для користувача дейтаграм (UDP). У таблиці служб зазначено велике число стандартних служб. Ця таблиця застосовується багатьма додатками для визначення порту служби.	В IPv6 ця таблиця застосовується без змін.
Простий протокол керування мережею (SNMP)	Протокол SNMP служить для управління системами.	Та ж сама підтримка для IPv6.
Віртуальна приватна мережа (VPN)	Віртуальна приватна мережа спільно з функцією IPsec дозволяє розширити захищену внутрішню мережу за рахунок зовнішньої мережі.	Та ж сама підтримка для IPv6.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
API сокетів	Ці API можуть застосовуватися в додатках для роботи з TCP / IP. Зміни, внесені в сокети в протоколі IPv6, не впливають на роботу додатків, які не планують застосовувати IPv6.	В IPv6 додатки з використанням сокетів можуть застосовувати нове сімейство адрес: AF_INET6. Зміни, внесені в API в протоколі IPv6, не впливають на роботу існуючих додатків, що використовують протокол IPv4. Додатки, які повинні підтримувати потоки даних IPv4 і IPv6, або тільки потік даних IPv6, можна легко адаптувати шляхом перетворення адрес IPv4 в адреси IPv6 формату :: ffff: a.b.c.d, де a.b.c.d - вихідний адресу IPv4 клієнта. Нові API підтримують перетворення адрес IPv6 з текстового формату в двійковий, і навпаки.
Telnet	Telnet дозволяє працювати з віддаленої системою так само, як з системою, з якою встановлено пряме з'єднання.	Та ж сама підтримка для IPv6.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Вибір адреси відправника	Додаток може призначити IP-адреса відправника (зазвичай для цього застосовується API сокетів bind ()). Якщо зв'язування буде встановлено з INADDR_ANY, то адреса відправника вибирається виходячи з маршруту.	Як і при роботі з IPv4, додаток може призначити адресу відправника в форматі IPv6 за допомогою функції bind (). Крім того, воно може дозволити системі вибрати адресу IPv6 відправника за допомогою in6addr_any. Однак оскільки з лінією зв'язку IPv6 може бути пов'язано декілька адрес IPv6, буде застосовуватися інший внутрішній алгоритм вибору IP-адреси відправника.
Запуск і завершення роботи	Для запуску і завершення роботи IPv4 служать команди STRTCP і ENDTCP. IPv4 зазвичай запускається при виконанні команди STRTCP для запуску TCP / IP.	Для запуску або завершення роботи IPv6 використовуйте параметр STRIP6 або команди STRTCP і ENDTCP. IPv6 може не запускатися під час запуску TCP / IP. Запустити IPv6 можна пізніше. Всі інтерфейси IPv6 запускаються автоматично, якщо параметр AUTOSTART дорівнює * YES (це стандартне значення). IPv6 неможливо застосовувати або налаштовувати без IPv4. Циклічний інтерфейс IPv6 :: 1, буде визначено і активований автоматично при запуску IPv6.

Продовження таблиці 3.4. Порівняння IPv6 та IPv4

Опис	IPv4	IPv6
Трасування маршруту	Трасування маршруту - це одна з основних функцій TCP / IP, яка застосовується для визначення маршруту. Викликається за допомогою IBM Navigator for i і текстового інтерфейсу.	Та ж сама підтримка для IPv6.
Транспортні рівні	TCP, UDP, RAW.	Подібні транспортні протоколи існують і в IPv6.
Невизначена адреса	Такий тип адреси відсутня. У програмуванні для гнізд 0.0.0.0 використовується як INADDR_ANY.	Дорівнює :: / 128 (128 нульових бітів). Вказується в якості IP-адреси відправника в деяких пакетах при пошуку сусідів, а також в інших випадках, наприклад, при роботі з сокетами. У додатках з API сокетів адресу :: / 128 використовується в якості in6addr_any.

3.5. ICMP — міжмережевий протокол керуючих повідомлень

ICMP (Internet Control Message Protocol) — міжмережевий протокол керуючих повідомлень, що входить в стек протоколів TCP/IP. В основному ICMP використовується для передачі повідомлень про помилки й інші виняткові ситуації, що виникли при передачі даних [3.1- 3.3]. Також на ICMP покладаються деякі сервісні функції, зокрема на основі цього протоколу заснована дія таких загальновідомих утиліт як ping та traceroute.

Протокол ICMP описаний в RFC 792 (з доповненнями в RFC 950) і є стандартом Інтернету (входить в стандарт STD 5 разом з протоколом IP). Хоча формально ICMP використовує IP (ICMP пакети інкапсулюються в IP пакети), він є невіддільною частиною IP й обов'язковий при реалізації стека TCP/IP. Поточна версія ICMP для IPv4 називається ICMPv4. В IPv6 існує аналогічний протокол ICMPv6.

Протокол ICMP не є протоколом орієнтованим на з'єднання (як наприклад TCP), тобто при втраті пакету ICMP не буде робити ніяких спроб по його відновленню. ICMP повідомлення (тип 12) генеруються при знаходженні

помилок у заголовку IP пакета (за винятком самих ICMP пакетів, щоб не призвести до нескінченного потоку ICMP повідомлень про ICMP повідомлення).

ICMP повідомлення (тип 3) генеруються маршрутизатором при відсутності маршруту до адресата (таб. 3.5).

Швидкість джерела може просити відправника зменшити швидкість повідомлення, яке було відправлено до маршрутизатора або хосту. Це повідомлення може бути створене, якщо маршрутизатор або хост не має достатнього буферного простору для обробки запиту, або може виникнути ситуація, що маршрутизатор або хост-буфер майже заповнений. Дані передаються на дуже високій швидкості від хоста або з декількох хостів до конкретного маршрутизатора в мережі. Можливості маршрутизації буферизації обмежені в межах вказаного діапазону [3.2-3.6].

Таблиця 3.5 Типи ICMP пакетів

Тип	Код	Назва	Запит	Помилка
0	0	Ехо-відповідь	x	
1 та 2		Зарезервовано		
3	0	Мережа недоступна		x
3	1	Хост недоступний		x
3	2	Протокол недоступний		x
3	3	Порт недоступний		x
3	4	Потрібна фрагментація, але DF (прапорець «не фрагментувати») встановлено		x
3	5	Маршрутизація джерелом не вдалося		x
3	6	Мережа призначення невідома		x
3	7	Хост призначення невідомий		x
3	8	Хост призначення ізольовано (застарів)		x
3	9	Мережа призначення адміністративно заборонена		x

Продовження таблиці 3.5 Типи ICMP пакетів

Тип	Код	Назва	Запит	Помилка
3	10	Хост призначення адміністративно заборонено		x
3	11	Мережа недоступна для TOS		x
3	12	Хост недоступний для TOS		x
3	13	Зв'язок адміністративно заборонено (фільтрування запобігає маршрутизації)		x
3	14	Порушення передування хосту (бажане передування не дозволено для комбінації хоста або мережі та порту)		x
3	15	Порушення передування в дії (передування датаграми нижче рівня встановленого адміністраторами)		x
4	0	Вгамовування джерела (управління заторами)		
5	0	Переадресація для мережі		
5	1	Переадресація для хосту		
5	2	Переадресація для TOS та мережі		
5	3	Переадресація для TOS та хосту		
6		Змінна адреса хосту		
7		Зарезервовано		
8	0	Ехо-запит	x	

Продовження таблиці 3.5 Типи ICMP пакетів

Тип	Код	Назва	Запит	Помилка
9	0	Оголошення маршрутизатора		
10	0	Клопотання маршрутизації		
11	0	Час життя пакету став рівний 0 під час транзиту		x
11	1	Час збірки фрагментів пакету закінчився		x
12	0	Помилка заголовку IP-пакету		x
12	1	Відсутня необхідна опція заголовку		x
12	2	Погана довжина заголовку		x
13	0	Запит мітки часу	x	
14	0	Мітка часу	x	
15	0	Запит інформації	x	
16	0	Відповідь інформацією	x	
17	0	Запит маски адреси	x	
18	0	Маска адреси	x	

Маршрутизатор не може стояти в черзі більше ніж обсяг обмеженого

простору буферизації. Таким чином, якщо в черзі з'являються нові вхідні дані, то вони відкидаються, поки черга не заповнилась. Але так як немає, в механізмі, підтвердження присутності на мережевому рівні, клієнт не знає, чи дані досягли мети успішно.

Тому деякі заходи щодо виправлення становища повинні бути прийняті на мережевому рівні, щоб уникнути такого роду ситуацій. Ці заходи розглядаються як джерело гарантії. У механізмі швидкості джерела, маршрутизатор бачить, що вхідна швидкість передачі даних значно вища, ніж вихідна швидкість передачі даних, і відправляє ICMP-повідомлення клієнтам, повідомивши їх, що вони повинні зменшити швидкість передачі даних або зачекати певний час, перш ніж відправити більше даних. Коли клієнт отримує це повідомлення, воно автоматично виконує вищезазначені дії, що дозволяють маршрутизатору очистити чергу.

Таким чином, джерело швидкості ICMP з допомогою повідомлень можуть управляти потоком даних на мережевому рівні.

При переадресації запитів пакети даних спрямовуються на альтернативний маршрут.

ICMP переадресація є механізмом маршрутизаторів для передачі інформації про маршрутизацію до хосту. Повідомлення інформує господаря про потребу оновити свою маршрутну інформацію (для посилання пакетів на альтернативний маршрут) [3.3, 3.8].

Якщо хост намагається надіслати дані через маршрутизатор (R1) і R1 відправляє дані на інший маршрутизатор (R2) і прямий шлях від хоста до R2 доступний (тобто, хост і R2 знаходяться на тому ж сегменті Ethernet), то R1 буде посилати повідомлення про перенаправлення інформуючи хост, що найкращий шлях для призначення через R2.

Хост повинен посилати пакети для призначення безпосередньо до R2. Маршрутизатор буде як і раніше відправити вихідні дейтаграми до місця призначення.

Однак, якщо дейтаграма містить інформацію про маршрутизації, це повідомлення не буде відправлено, навіть якщо найкращий маршрут доступний. RFC1122 стверджує, що переадресація повинна бути послана тільки на шлюз і не повинна бути відправлені хосту через інтернет [3.5]. Код вказує причину для переадресації, може бути одна з наступних (таб. 3.6)

Таблиця 3.6 Коди та види переадресації

Код	Опис
0	Переадресація мережі
1	Переадресація хост
2	Переадресація для типу обслуговування та мережі
3	Переадресація для типу обслуговування та хосту

IP-адреса 32-розрядний адрес шлюзу, до якого переадресація повинна бути надіслана.

Напрямок недоступності існує, щоб повідомляти клієнта про те, що пункт призначення недоступний з яких-небудь причин (таб. 3.7). Повідомлення Destination Unreachable (Напрямок недоступності) може бути отримане в результаті TCP або ICMP іншої передачі.

Шлюз може відправити повідомлення про недоступність пункту призначення на хост інтернету джерела дейтаграми. Помилка не буде створена, якщо вихідна дейтаграма має групову адресу призначення. Причини цього повідомлення: фізичне підключення до хосту не існує, вказаний протокол або порт не активний, дані не фрагментовані.

Таблиця. 3.7. Повідомлення про напрямок недоступності

де:

Тип поля (біти 0-7) повинні бути встановлені до 3;

Код поля (біти 8-15) використовується для визначення типу помилки.

Максимальний блок передачі обмеження посилання на максимальну кількість байт даних в одній передачі (тобто кадр, клітка, пакет).

Найменший MTU будь-якої ланки на поточному шляху знаходиться між двома хостами.

Це може змінюватися з плином часу, так як маршрут між двома хостами, особливо в Інтернеті, може змінюватися з плином часу.

3.6. IGMP — протокол керування групами Інтернету

IGMP (Internet Group Management Protocol — протокол керування групами Інтернету) — протокол керування груповою (multicast) передачею даних в мережах, базованих на протоколі IP [3.2, 3.7].

IGMP використовується маршрутизаторами і IP-точками для об'єднання мережевих пристроїв в групи. Цей протокол є частиною специфікації групової передачі пакетів в IP-мережах. IGMP розташований вище мережевого рівня, хоча, насправді, функціонує не як транспортний протокол. Він в багато чому

аналогічний ICMP для односторонньої передачі. IGMP може використовуватись для підтримки потокового відео і онлайн-ігор, для таких типів програм він дозволяє використовувати ресурси мережі більш ефективно.

Брандмауери, зазвичай, дозволяють користувачу відключити цей протокол, якщо в ньому нема потреби.

IGMP використовується лише в мережах IPv4, оскільки в IPv6 групова передача пакетів реалізована інакше.

Протокол групового управління в Інтернеті (Internet Group Management Protocol, IGMP) був розроблений в 1989 році для забезпечення більш ефективної розсилки інформації по IP-адресами, ніж традиційні методи одноадресної і широкомовної передачі. Існує три версії IGMP: IGMPvl (RFC 1112), IGMPv2 (RFC 2236) hIGMPv3 (RFC 3376).

Протокол IGMP використовується виключно при взаємодії безпосередньо пов'язаних один з одним маршрутизатора і хоста, коли останній виступає (або бажає виступати) в ролі одержувача трафіку групового мовлення.

Мережа, що надає послуги групової передача даних (наприклад, відео) із використанням IGMP, може мати наступну базову архітектуру: IGMP використовується клієнтським комп'ютером і сусідніми комутаторами для з'єднання клієнта і локального маршрутизатора, що здійснює групову передачу. Далі між локальним і віддаленим маршрутизаторами використовується протокол Protocol Independent Multicast (PIM), з його допомогою груповий трафік прямує від відеосервера до численних клієнтів групової передачі.

Згідно з Request for Comments (RFC), документом спільноти Internet Engineering Task Force (IETF), існує три версії IGMP. IGMPv1 визначений в RFC 1112, IGMPv2 - в RFC 2236 і IGMPv3 - в RFC 3376. Основною перевагою IGMPv3 над IGMPv2 є підтримка фільтрації IP-адрес [3.1-3.5, 3.9]. За допомогою цього механізму вузол може повідомити, з яких адрес він ,буде отримувати пакети, а з яких ні.

Протокол IGMP реалізований у вигляді серверної та клієнтської частин,

перша з яких виконується на маршрутизаторі, друга - у вузлі мережі, що отримує груповий трафік. Клієнт надсилає повідомлення про належність до якої-небудь групи локального маршрутизатора, в цей час маршрутизатор знаходиться в очікуванні повідомлень і періодично розсилає клієнтам запити.

Операційні системи FreeBSD, Linux і Windows підтримують клієнтську частину протоколу. В системі Linux IGMPv3 був доданий у версії ядра 2.5. Для FreeBSD IGMPv3 був доданий у версії 8.0. Для реалізації серверної частини IGMP в Linux використовуються демони, наприклад, mrouted може діяти як IGMP маршрутизатор. Існують також цілі програмні комплекси (такі, як XORP), що дозволяють перетворити звичайний комп'ютер у повнофункціональний маршрутизатор групової передачі.

Локальна мережа може мати декілька хостів, зацікавлених в отриманні трафіку однієї і тієї ж групи, але маршрутизатора досить підтвердження тільки від одного хоста для того, щоб продовжити передавати трафік в мережу для цієї групи. При використанні протоколу IGMPvl або IGMPv2 для обмеження числа відповідей хостів на запит маршрутизатора будь-який хост, що складається в групі, замість того щоб негайно відповісти на запит, спочатку чекає протягом деякого інтервалу часу, чи не з'явиться в мережі відповідь якогось іншого хоста. Якщо після закінчення цього часу він так і не зміг дочекатися появи в мережі відповіді іншого хоста, то він посилає маршрутизатора власний звіт про членство. Якщо ж використовується протокол IGMPv3, то ніяких пауз не встановлюється, і хости відразу генерують повідомлення про членство [3.1, 3.5].

Грунтуючись на інформації, отриманої за допомогою IGMP, маршрутизатори можуть визначати, в які підключені до них мережі необхідно передавати груповий трафік.

Всі типи IGMP-повідомлень мають довжину 8 байт і складаються з чотирьох полів. Залежно від версії протоколу IGMP призначення полів може дещо змінюватися. Нижче таблиця 3.8 показана структура повідомлення для версії IGMPv2.

Таблиця 3.8. Структура пакетів IGMP

1-4 байти	**Тип повідомлення**	**Максимальний час відповіді**	**Контрольна сума**
5-8 байти	**Адреса групового мовлення (Multicast group address)**		

Поле максимального часу відповіді використовується хостами для обчислення часу затримки відповіді. Час затримки вибирається випадковим чином з інтервалу від нуля до значення, заданого в цьому полі.

Поле адреси групового мовлення в IGMP-повідомленні не містить адресу призначення, воно несе в собі інформацію, по-різному використовується в різних типах повідомлень. Наприклад, маршрутизатор, посилаючи запит про членство, поміщає в цьому полі нулі, а хост в повідомленнях «Звіт про членство» і «Покинути групу» поміщає в це поле адресу групи, в яку він хоче вступити або яку він хоче покинути відповідно.

Щоб хост зміг отримувати трафік групового мовлення, недостатньо встановити на ньому протокол IGMP, за допомогою якого хост може відправити повідомлення свого маршрутизатора про бажання приєднатися до груп. Крім цього, треба конфігурувати мережевий інтерфейс хоста так, щоб він став захоплювати з локальної мережі кадри, що несуть в собі пакети групового мовлення для тієї групи, до якої приєднався хост. Для цього необхідно налаштувати інтерфейс на прослуховування певної групової адреси канального рівня, відповідного груповій IP-адреси [3.5].

На жаль, адресний простір групових IP-адрес в 32 рази об'ємніше простору групових MAC-адрес. Тобто відображення цих двох адресних просторів виявляється далеко неоднозначним - на один і той же груповий MAC-адреса відображається цілий блок з 32 різних групових IP-адрес.

Отже, коли мережевий адаптер захоплює кадр, що містить пакет

групового мовлення, існує значна ймовірність того, що цей пакет був спрямований зовсім іншій групі. Однак ця помилка скоро виявляється.

Коли кадр передається вгору по стеку, протокол IP перевіряє, чи збігається груповий IP-адреса в поле адреси призначення інкапсульованого пакета з груповим IP-адресою даного інтерфейсу. (Ні групові IP-адреси, ні групові MAC-адреси ніколи не використовуються в якості адрес відправника).

Структура пакетів IGMPv3. Визначена стандартом RFC 3376. Запит приналежності (Membership Query Message). Запити приналежності розсилаються маршрутизаторами для того, щоб для кожного вузла визначити його приналежність до яких-небудь груп (group membership state) і список джерел інформації, від яких даний вузол хоче отримувати повідомлення (reception state) [3.1, 3.4].

Існує три типи таких запитів:

Загальні запити (General Queries) - дозволяють отримати повну інформацію для кожного з вузлів. Маршрутизатор періодично розсилає ці запити всім системам, підключеним до його мережі.

Запити із зазначенням групи (Group-Specific Queries) - використовуються для визначення стану підписки для заданої групи вузлів. Такі запити розсилаються по відповідній груповій адресі.

Запити із зазначенням групи та джерела (Group-and-Source-Specific Queries) - дозволяє для кожного вузла заданої групи визначити, які повідомлення з усіх, що посилаються заданими джерелами, цей вузол хоче отримувати.

Код макс. відповіді (Max Resp Code) - максимальний час (в 1/10 секунди) очікування відповіді, відповідного даному запиту. Якщо значенням є число, менше 128, воно використовується безпосередньо. Якщо ж значення більше або дорівнює 128, воно інтерпретується як експонента з мантиси [3.1-3.4].

Контрольна сума (Checksum) - 16-бітова контрольна сума для всього IGMP-повідомлення.

Групова адреса (Group Address) - групова адреса, яка використовується в запитах із зазначенням групи. При загальному запиті це поле встановлюється рівним нулю. Resv - поле зарезервовано, його слід обнуляти при посилці і ігнорувати при отриманні. Прапор S (Припинити серверну обробку, Suppress Router-side Processing) - установка цього прапора вказує всім маршрутизаторам, що отримали це повідомлення, припинити оновлення за таймером.

QRV (Змінна надійності запитувача, Querier's Robustness Variable) - містить змінну надійності (Robustness Variable), значення якої використовується надсилаючим пристроєм. Маршрутизатори повинні оновлювати їх змінні надійності відповідно до останнього отриманого запиту, поки це поле не стане нульовим.

QQIC (Код інтервалу запиту, Querier's Query Interval Code) - значення поля вказує інтервал між запитами (Query Interval), використовуваний запитуючою. Якщо значенням є число, менше 128, воно використовується безпосередньо. Якщо ж значення більше або дорівнює 128, воно інтерпретується як експонента з мантиси.

Кількість джерел (Number of Sources, N) - визначає число адрес джерел, присутніх в цьому запиті. Для загальних запитів і запитів із зазначенням групи це значення дорівнює нулю. Для запитів із зазначенням групи та джерела це поле ненульове, воно обмежене значенням MTU мережі. Адреса джерела - масив індивідуальних (не групових) IP-адрес джерел даних.

Звіт про приналежність (Membership Report Message) [3.1- 3.6].

Reserved - повинно встановлюватися в нуль при передачі і ігноруватися при прийомі; Number of Group Record - кількість полів Group Record в повідомленні; Group Record - блок полів, що містить інформацію про членство відправника в групі:

Record Type - тип запису: Поточний стан - надсилається у відповідь на запит, повідомляє про поточний режим фільтрації, щодо зазначеного групової адреси, приймає значення MODE_IS_INCLUDE і MODE_IS_EXCLUDE; Зміна

режиму - надсилається при зміні режиму фільтрації, приймає значення CHANGE_TO_INCLUDE_MODE і CHANGE_TO_EXCLUDE_MODE; Зміна списку джерел - надсилається при зміні списку джерел без зміни режиму фільтрації: ALLOW_NEW_SOURCES - в режимі INCLUDE адреси додаються до списку, в режимі EXCLUDE - видаляються зі списку; BLOCK_OLD_ SOURCES - в режимі EXCLUDE адреси додаються до списку, в режимі INCLUDE - видаляються зі списку. Aux Data Len - довжина додаткових даних в 32-бітових словах. Number of Sources - кількість адрес джерел даних; Multicast Address - групова адреса, до якої відноситься інформація в запису; Source Address - масив індивідуальних IP-адрес джерел даних; Auxilary Data - додаткова інформація,яка не повинна використовуватися в поточній версії даного протоколу, оскільки це є недоцільно.

3.7. IPsec — набір протоколів для захисту даних

IPsec (IP Security) — набір протоколів для забезпечення захисту даних, що передаються за допомогою протоколу IP, дозволяє здійснювати підтвердження справжності та/або шифрування IP-пакетів. IPsec також містить в собі протоколи для захищеного обміну ключами в мережі Інтернет [3.1, 3.8].

Стандарти:

RFC 2401 (Security Architecture for the Internet Protocol) — Архітектура захисту для протоколу IP.

RFC 2402 (IP Authentication header) — аутентифікаційні заголовок IP.

RFC 2403 (The Use of HMAC-MD5-96 within ESP and AH) — Використання алгоритму хешування MD-5 для створення аутентифікаційні заголовка.

RFC 2404 (The Use of HMAC-SHA-1-96 within ESP and AH) — Використання алгоритму хешування SHA-1 для створення аутентифікаційні заголовка.

RFC 2405 (The ESP DES-CBC Cipher Algorithm With Explicit IV) — Використання алгоритму шифрування DES.

RFC 2406 (IP Encapsulating Security Payload (ESP)) — Шифрування даних.

RFC 2407 (The Internet IP Security Domain of Interpretation for ISAKMP) — Область застосування протоколу управління ключами.

RFC 2408 (Internet Security Association and Key Management Protocol (ISAKMP)) — Управління ключами і аутентифікатором захищених з'єднань.

RFC 2409 (The Internet Key Exchange (IKE)) — Обмін ключами.

RFC 2410 (The NULL Encryption Algorithm and Its Use With IPsec) — Нульовий алгоритм шифрування і його використання.

RFC 2411 (IP Security Document Roadmap) — Подальший розвиток стандарту.

RFC 2412 (The OAKLEY Key Determination Protocol) — Перевірка відповідності ключа.

Протоколи IPsec, на відміну від інших добре відомих протоколів SSL та TLS, працюють на мережевому рівні (рівень 3 моделі OSI). Це робить IPsec гнучкішим, так що він може використовуватися для захисту будь-яких протоколів, що базуються на TCP та UDP. IPsec може використовуватися для забезпечення безпеки між двома IP-вузлами, між двома шлюзами безпеки або між IP-вузлом і шлюзом безпеки. Протокол є «надбудовою» над IP-протоколом, і обробляє сформовані IP-пакети описаним нижче способом. IPsec може забезпечувати цілісність та / або конфіденційність даних переданих по мережі [3.9].

IPsec використовує наступні протоколи для виконання різних функцій:

- *Authentication Header (AH)* забезпечує цілісність віртуального з'єднання (переданих даних), аутентифікацію джерела інформації та додаткову функцію із запобігання повторної передачі пакетів
- *Encapsulating Security Payload (ESP)* може забезпечити конфіденційність (шифрування) переданої інформації, обмеження потоку

конфіденційного трафіку. Крім цього, він може забезпечити цілісність віртуального з'єднання (переданих даних), аутентифікацію джерела інформації та додаткову функцію із запобігання повторної передачі пакетів (Всякий раз, коли застосовується ESP, в обов'язковому порядку повинен використовуватися той чи інший набір даних послуг із забезпечення безпеки).

- *Security Association (SA)* забезпечують зв'язку алгоритмів і даних, які надають параметри, необхідні для роботи AH і / або ESP. Internet security association and key management protocol (ISAKMP) забезпечує основу для аутентифікації і обміну ключами, перевірки автентичності ключів.

Наприкінці 1980-х стала очевидною нестача адресного простору Інтернет. На початку 1990-х, навіть після введення безкласової адресації, виявилось, що однієї економії та використання NAT'у буде замало, щоб запобігти вичерпання адресного простору, і необхідна зміна адресації. Крім того, накопичилась певна кількість пропозицій щодо усунення недоліків наявної моделі Інтернет. Наприкінці 1992 року IETF оголосила конкурс на створення протоколу Інтернет наступного покоління (IP Next Generation — IPng).

25 липня 1994 року IETF ствердила модель IPng з утворенням кількох робочих груп IPng. У 1996 було створено серію RFC, що визначали новий протокол Інтернет. Оскільки версія 5 вже була раніше призначена експериментальному протоколу передачі мультимедійних потоків, новий протокол отримав версію 6.

Концепція *«захищеного віртуального з'єднання» (SA, «Security Association»)* є фундаментальною в архітектурі *IPsec*. *SA* це симплексне з'єднання, яке формується для транспортування по ньому відповідного трафіку. При реалізації послуг безпеки формується *SA* на основі використання протоколів *AH* або *ESP* (або обох одночасно). *SA* визначений відповідно до концепції міжтермінального з'єднання (point-to-point) і може функціонувати в двох режимах: транспортний режим (*PTP*) і режим тунелювання (*РТУ*). Транспортний режим реалізується при *SA* між двома *IP*-вузлами. В режимі

тунелювання *SA* формує *IP*-тунель [3.1-3.5, 3.9].

Всі *SA* зберігаються в базі даних *SADB (Security Associations Database) IPsec*-модуля. Кожне *SA* має унікальний маркер, що складається з трьох елементів:

- *Індексу параметра безпеки (SPI)*
- *IP-адреси призначення*
- *Ідентифікатора протоколу безпеки (ESP або AH)*.

IPsec-модуль, маючи ці три параметри, може відшукати в *SADB* запис по конкретному *SA*. У список компонентів *SA* входять:

- ***Послідовний номер***

32-бітове значення, яке використовується для формування поля *Sequence Number* в заголовках *AH і ESP*.

- ***Переповнення лічильника порядкового номера***
- Прапор, який сигналізує про переповнення лічильника послідовного номера.
- ***Вікно для придушення атак відтворення***
- Використовується для визначення повторної передачі пакетів. Якщо значення в полі *Sequence Number* не потрапляє в заданий діапазон, то пакет знищується.
- ***Інформація AH***
- використовуваний алгоритм аутентифікації, необхідні ключі, час життя ключів та інші параметри.
- ***Інформація ESP***
- алгоритми шифрування і аутентифікації, необхідні ключі, параметри ініціалізації (наприклад, IV), час життя ключів та інші параметри
- ***Режим роботи IPsec***
- тунельний або транспортний
- ***MTU***
- Максимальний розмір пакета, який можна передати по віртуальному

каналу без фрагментації.

Так як захищені віртуальні з'єднання є симплексними, то для організації дуплексного каналу, як мінімум, потрібні два SA. Крім цього, кожен протокол (ESP / AH) повинен мати свою власну SA для кожного напрямку, тобто, зв'язка AH + ESP вимагає наявності чотирьох SA. Всі ці дані розташовуються в SADB.

В SADB містяться:

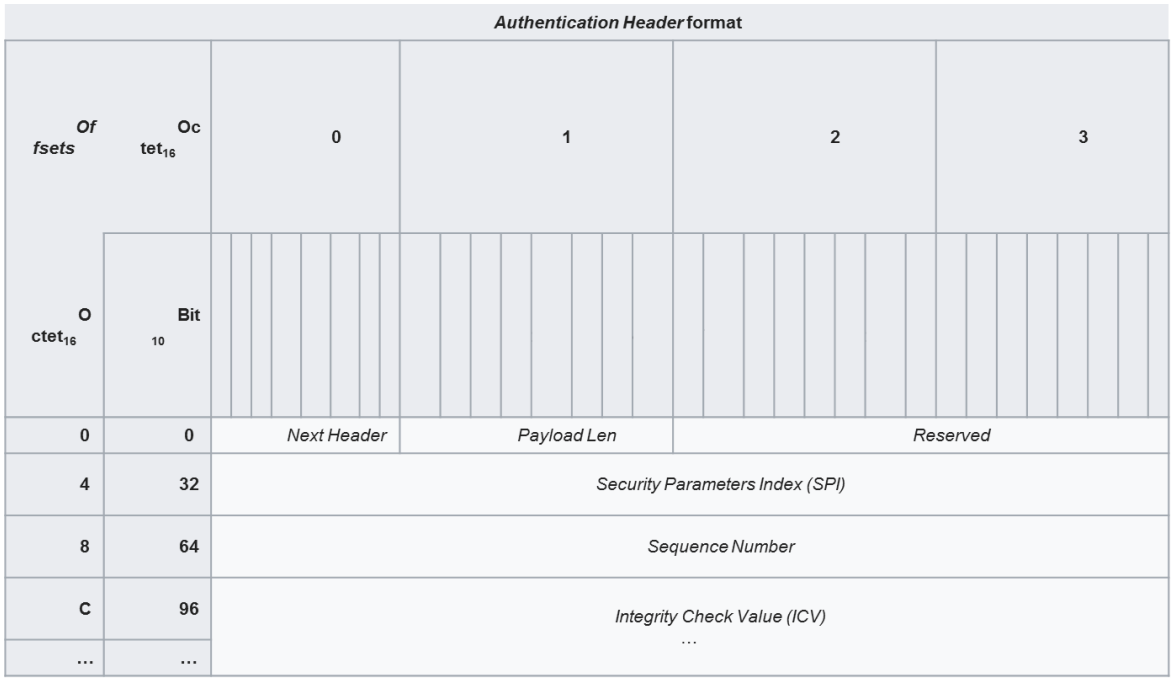
- AH: алгоритм аутентифікації.
- AH: секретний ключ для аутентифікації.
- ESP: алгоритм шифрування.
- ESP: секретний ключ шифрування.
- ESP: використання аутентифікації (так / ні).
- Параметри для обміну ключами.
- Обмеження маршрутизації.
- IP політика фільтрації.

Крім бази даних SADB, реалізації IPsec підтримують базу даних SPD (Security Policy Database-База даних політик безпеки). Запис в SPD складається з набору значень полів IP-заголовка і полів заголовка протоколу верхнього рівня. Ці поля називаються селекторами. Селектори використовуються для фільтрації вихідних пакетів, з метою поставити кожен пакет у відповідність з певним SA. Коли формується пакет, порівнюються значення відповідних полів у пакеті (селекторні поля) з тими, які містяться SPD. Знаходяться відповідні SA. Потім визначається SA (у випадку, якщо воно є) для пакета і пов'язаний з нею індекс параметрів безпеки (SPI). Після чого виконуються операції IPsec (операції протоколу AH або ESP) [3.9].

Приклади селекторів, які містяться в SPD:

- IP-адреса місця призначення
- IP-адреса відправника
- Протокол IPsec (AH, ESP або AH + ESP)
- Порти відправника та одержувача/

Таблиця 3.9. Формат заголовка аутентифікації



Authentication Header format					
Offsets Octet16		0	1	2	3
Octet16	Bit10				
0	0	Next Header	Payload Len	Reserved	
4	32	Security Parameters Index (SPI)			
8	64	Sequence Number			
C	96	Integrity Check Value (ICV) …			
…	…				

***Next Header* (8 bits) -** Тип заголовка протоколу, що йде після заголовка *AH*. По цьому полю приймальний *IP-sec* модуль дізнається про захищається протоколі верхнього рівня. Значення цього поля для різних протоколів можна подивитися в RFC 1700.

***Payload Len* (8 bits) -** Це поле визначає загальний розмір *AH*-заголовка в 32-бітових словах, мінус 2. Незважаючи на це, при використанні *IPv6* довжина заголовка повинна бути кратна 8 байтам.

***Reserved* (16 bits) -** Зарезервовано. Заповнюється нулями.

***Security Parameters Index* (32 bits) -** Індекс параметрів безпеки. Значення цього поля разом з *IP*-адресою одержувача і протоколом безпеки (*AH-протокол*), однозначно визначає захищене віртуальне з'єднання (*SA*) для даного пакета. Діапазон значень SPI 1 … 255 зарезервований IANA.

***Sequence Number* (32 bits) -** Послідовний номер. Служить для захисту від повторної передачі. Поле містить монотонно зростаюче значення параметра. Незважаючи на те, що одержувач може відмовитися від послуги із захисту від повторної передачі пакетів, воно є обов'язковим і завжди присутній в AH-заголовку. Передавальний IPsec-модуль завжди використовує це поле, але

одержувач може його і не обробляти.

Integrity Check Value - Контрольна сума. Повинна бути кратна 8-байтам для IPv6, і 4-байтам для IPv4.

Протокол ***AH*** використовується для аутентифікації, тобто для підтвердження того, що ми зв'язуємося саме з тим, з ким припускаємо, і що дані, які ми отримуємо, не спотворені при передачі.

Обробка вихідних IP-пакетів відбувається наступним чином [3.1-3.6].

Якщо передавальний IPsec-модуль визначає, що пакет пов'язаний з SA, яке передбачає AH-обробку, то він починає обробку. В залежності від режиму (транспортний або режим тунелювання) він по-різному вставляє AH-заголовок в IP-пакет. У транспортному режимі AH-заголовок розташовується після заголовка протоколу IP і перед заголовками протоколів верхнього рівня (Зазвичай, TCP або UDP).

В режимі тунелювання весь вихідний IP-пакет обрамляється спочатку заголовком AH, потім заголовком IP-протоколу. Такий заголовок називається зовнішнім, а заголовок вихідного IP-пакета-внутрішнім. Після цього передавальний IPsec-модуль повинен згенерувати послідовний номер і записати його в поле Sequence Number. При встановленні SA послідовний номер встановлюється в 0, і перед відправкою кожного IPsec-пакета збільшується на одиницю. Крім того, відбувається перевірка — чи не зациклився лічильник. Якщо він досяг свого максимального значення, то він знову встановлюється в 0. Якщо використовується послуга щодо запобігання повторної передачі, то при досягненні лічильника свого максимального значення, що передає IPsec-модуль переустановлює SA. Таким чином забезпечується захист від повторної посилки пакета — приймальний IPsec-модуль буде перевіряти поле Sequence Number, і ігнорувати пакети, які приходять повторно. Далі відбувається обчислення контрольної суми ICV.

Контрольна сума обчислюється із застосуванням секретного ключа, без якого зловмисник зможе заново обчислити хеш, але не знаючи ключа, не зможе

сформувати правильну контрольну суму. Конкретні алгоритми, що використовуються для обчислення ICV, можна дізнатися з RFC 4305. В даний час можуть застосовуватися, наприклад, алгоритми HMAC-SHA1-96 або AES-XCBC-MAC-96.

Протокол AH обчислює контрольну суму (ICV) за наступними полям IPsec-пакета:

Поля IP-заголовка, що не були схильні до змін в процесі транслювання, або визначені як найважливіші;

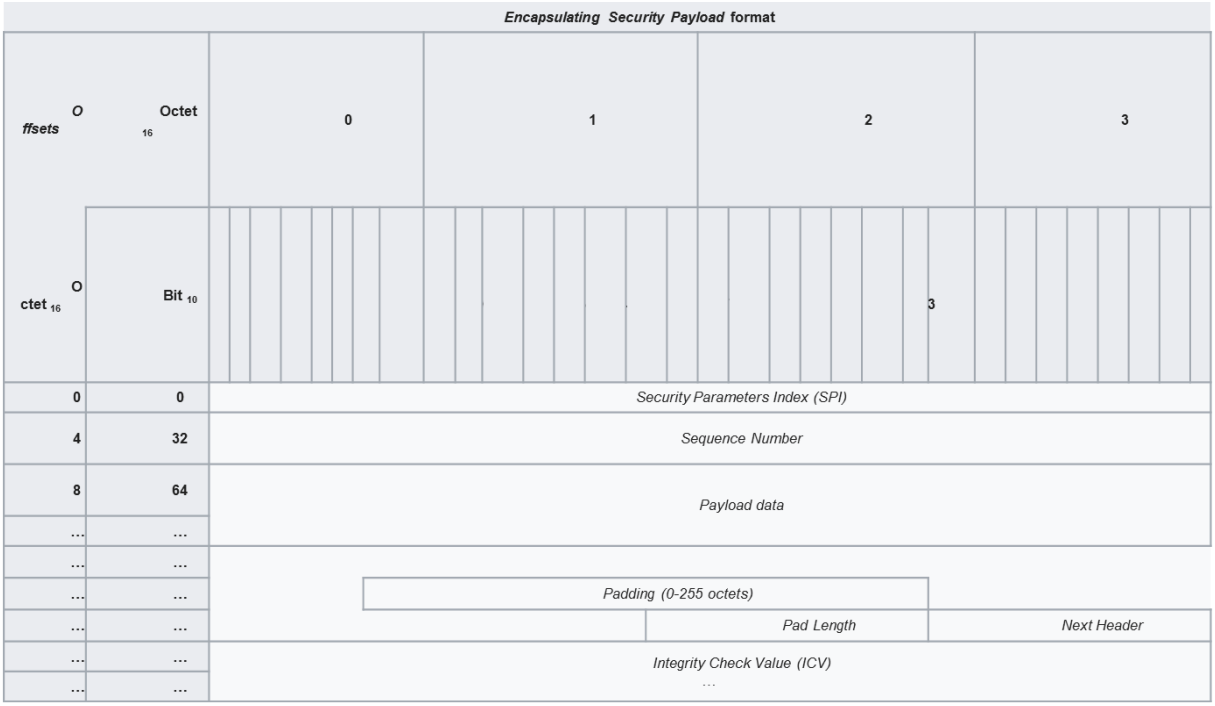
AH-заголовок (Поля: «Next Header», "Payload Len, " Reserved ", " SPI ", " Sequence Number ", " Integrity Check Value «. Поле» Integrity Check Value "встановлюється в 0 при обчисленні ICV.

Дані протоколу верхнього рівня: Якщо поле може змінюватися в процесі транспортування, то його значення встановлюється в 0 перед обчисленням ICV. Винятки становлять поля, які можуть змінюватися, але значення яких можна передбачити при прийомі. При обчисленні ICV вони не заповнюються нулями. Прикладом змінюваного поля може служити поле контрольної суми, прикладом змінюваного, але зумовленого може бути IP-адресу одержувача. Докладніший опис того, які поля як враховуються при обчисленні ICV, можна знайти в стандарті RFC 2402.

Після отримання пакета, що містить повідомлення AH-протоколу, приймальний IPsec-модуль шукає відповідне захищене віртуальне з'єднання (SA) SADB (Security Associations Database), використовуючи IP-адресу одержувача, протокол безпеки (AH) і індекс SPI. Якщо відповідне SA не знайдено, пакет знищується. Знайдене захищене віртуальне з'єднання (SA) вказує на те, чи використовується послуга щодо запобігання повторної передачі пакетів, тобто на необхідність перевірки поля Sequence Number. Якщо послуга використовується, то поле перевіряється. Для цього використовується метод ковзаючого вікна. Приймальний IPsec-модуль формує вікно з шириною W. Лівий край вікна відповідає мінімальному послідовному номеру (Sequence

Number) N правильно прийнятого пакета. Пакет з полем Sequence Number, в якому міститься значення, починаючи від N +1 і закінчуючи N + W, приймається коректно [3.2, 3.7].

Таблиця 3.10. Обробка вихідних IP-пакетів



Якщо отриманий пакет виявляється по ліву межу вікна - він знищується.

Потім приймальний IPsec-модуль обчислює ICV за відповідними полям прийнятого пакета, використовуючи алгоритм аутентифікації, який він дізнається з запису про SA, і порівнює отриманий результат із значенням ICV, розташованим в поле «Integrity Check Value».

Якщо обчислене значення ICV збіглося з прийнятим, то прийшов пакет вважається дійсним і приймається для подальшої IP-обробки.

Якщо перевірка дала негативний результат, то пакет який прийшов - знищується.

Encapsulating Security Payload:

- ***Security Parameters Index* (32 bits)**

Індекс параметрів безпеки. Значення цього поля разом з IP-адресою одержувача і протоколом безпеки (AH-протокол), однозначно визначає захищене віртуальне з'єднання (SA) для даного пакета. Діапазон значень SPI 1

… 255 зарезервований IANA для подальшого використання.

- ***Sequence Number*** **(32 bits)**

Послідовний номер. Служить для захисту від повторної передачі. Поле містить монотонно зростаюче значення параметра. Незважаючи на те, що одержувач може і відмовитися від послуги із захисту від повторної передачі пакетів, воно завжди присутній в AH-заголовку. Відправник (передавальний IPsec-модуль) повинен завжди використовувати це поле, але одержувач може і не мати потребу в його обробці.

Обробка вихідних IPsec-пакетів розглянемо далі [3.1, 3.3-3.6].

Якщо передавальний IPsec-модуль визначає, що пакет пов'язаний з SA, яке передбачає ESP-обробку, то він починає обробку. В залежності від режиму (транспортний або режим тунелювання) вихідний IP-пакет обробляється по-різному. У транспортному режимі передавальний IPsec-модуль здійснює процедуру обрамлення (інкапсуляції) протоколу верхнього рівня (наприклад, TCP або UDP), використовуючи для цього ESP-заголовок і ESP-закінчення, не зачіпаючи при цьому заголовок вихідного IP -пакета. В режимі тунелювання IP-пакет обрамляється ESP-заголовком і ESP-закінченням, після чого обрамляється зовнішнім IP-заголовком. Далі проводиться шифрування-в транспортному режимі шифрується тільки повідомлення протоколу вище лежачого рівня (тобто все, що знаходилося після IP-заголовка у вихідному пакеті), в режимі тунелювання-весь вихідний IP-пакет. Передавальний IPsec-модуль із запису про SA визначає алгоритм шифрування і секретний ключ. Стандарти IPsec дозволяють використання алгоритмів шифрування triple-DES, AES і Blowfish.

Так як розмір відкритого тексту повинен бути кратний певному числу байт, наприклад, розміром блоку для блокових алгоритмів, перед шифруванням проводиться ще й необхідне доповнення повідомлення що шифрується.

Зашифроване повідомлення поміщається в поле Payload Data.

В поле Pad Length поміщається довжина доповнення. Потім, як і в AH,

обчислюється Sequence Number. Після чого вважається контрольна сума (ICV).

Контрольна сума, на відміну від протоколу AH, де при її обчисленні враховуються також і деякі поля IP-заголовка, в ESP обчислюється тільки по полях ESP-пакета за вирахуванням поля ICV. Перед обчисленням контрольної суми воно заповнюється нулями. Алгоритм обчислення ICV, як і в протоколі AH, що передає IPsec-модуль визначається з запису про SA, з яким пов'язаний оброблюваний пакет [3.2, 3.4-3.6].

Після отримання пакета, що містить повідомлення ESP-протоколу, приймальний IPsec-модуль шукає відповідне захищене віртуальне з'єднання (SA) в SADB (Security Associations Database), використовуючи IP-адресу одержувача, протокол безпеки (ESP) і індекс SPI. Якщо відповідне SA не знайдено, пакет знищується. Знайдене захищене віртуальне з'єднання (SA) вказує на те, чи використовується послуга щодо запобігання повторної передачі пакетів, тобто на необхідність перевірки поля Sequence Number. Якщо послуга використовується, то поле перевіряється. Для цього, так само як і в AH, використовується метод ковзаючого вікна. Приймальний IPsec-модуль формує вікно з шириною W. Лівий край вікна відповідає мінімальному послідовному номеру (Sequence Number) N правильно прийнятого пакета. Пакет з полем Sequence Number, в якому міститься значення, починаючи від N +1 і закінчуючи N + W, приймається коректно.

Якщо отриманий пакет виявляється по ліву межу вікна - він знищується. Потім, якщо використовується послуга аутентифікації, приймальний IPsec-модуль обчислює ICV за відповідними полям прийнятого пакета, використовуючи алгоритм аутентифікації, який він дізнається з запису про SA, і порівнює отриманий результат із значенням ICV, розташованим в поле «Integrity Check Value» [3.2, 3.6-3.9].

Якщо обчислене значення ICV збіглося з прийнятим, то прийшов пакет вважається дійсним. Якщо перевірка дала негативний результат, то пакет який прийшов - знищується. Далі проводиться розшифрування пакета. Приймальний

IPsec-модуль дізнається з запису про SA, який алгоритм шифрування використовується, і секретний ключ.

Перевірка контрольної суми і процедура розшифрування можуть проводитися не тільки послідовно, а й паралельно. В останньому випадку процедура перевірки контрольної суми повинна закінчитися раніше процедури розшифрування, і якщо перевірка ICV провалилася, процедура розшифрування також повинна припинитися. Це дозволяє швидше виявляти зіпсовані пакети, що, в свою чергу, підвищує рівень захисту від атак типу «відмова в обслуговуванні» (DOS-атаки) [3.1, 3.8, 3.9].

Далі розшифроване повідомлення відповідно до полем Next Header передається для подальшої обробки.

Протокол IPsec використовується, в основному, для організації VPN-тунелів. В цьому випадку протоколи ESP і AH працюють в режимі тунелювання. Крім того, налаштовуючи політики безпеки певним чином, протокол можна використовувати для створення міжмережевого екрану.

Сенс міжмережевого екрану полягає в тому, що він контролює і фільтрує пакети, що проходять через нього, відповідно до заданих правил. Встановлюється набір правил, і екран переглядає всі пакети які проходять через нього. Якщо передані пакети потрапляють під дію цих правил, міжмережевий екран обробляє їх відповідним чином. Наприклад, він може відхиляти певні пакети, тим самим припиняючи небезпечні з'єднання [3.2, 3.8].

Налаштувавши політику безпеки відповідним чином, можна, наприклад, заборонити інтернет-трафік. Для цього достатньо заборонити відсилання пакетів, в які вкладаються повідомлення протоколів HTTP та HTTPS. IPsec можна застосовувати і для захисту серверів — для цього відкидаються всі пакети, окрім пакетів, необхідних для коректного виконання функцій сервера. Наприклад, для Web-сервера можна блокувати весь трафік, за винятком з'єднань через 80-й порт протоколу TCP, або через порт TCP 443 у випадках, коли застосовується HTTPS.

ЛІТЕРАТУРА

3.1 Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципи, технологии, протоколы: Учебник для вузов. – СПб.: Питер, 2001. – 672 с.

3.2 Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципи, технологии, протоколы: Учебник для вузов. 3-е изд. – СПб.: Питер, 2006.– 958 с.

3.3 Компьютерные сети. 4-е изд./ И. Таненбаум. – СПб.: Питер, 2003. – 992 с.

3.4 Проектирование и техническая эксплуатация сетей передачи дискретных сообщений. М., “Радио с связь”, 1988.

3.5 Кулаков Ю.О., Луцький Г.М. Комп’ютерні мережі. Підручник. – Київ.: «Юніор», 2005. – 396 с.

3.6 Галкин В. А., Григорьев Ю. А. Телекоммуникации и Сети. — М.: МГТУ им. Н. Э. Баумана, 2003. С. 608. ISBN 5-7038-1961-X

3.7 Friend, George E.; John L. Fike, H. Charles Baker, John C. Bellamy (1988). Understanding Data Communications (вид. 2nd). Indianapolis: Howard W. Sams & Company. ISBN 0-672-27270-9.

3.8 Stallings, William (2004). Data and Computer Communications (вид. 7th). Upper Saddle River: Pearson/Prentice Hall. ISBN 978-013100-6812.

3.9 S. Tanenbaum, Andrew (2005). Computer Networks (вид. 4th). 482,F.I.E., Patparganj, Delhi 110 092: Dorling Kindersley(India)Pvt. Ltd.,licenses of Pearson Education in South Asia. ISBN 81-7758-165-1.

4. ПРОТОКОЛИ ТРАНСПОРТНОГО РІВНЯ

4.1. UDP-протокол датаграм користувача

User Datagram Protocol, UDP (Протокол датаграм користувача) — один із протоколів в стеку TCP/IP. Від протоколу TCP він відрізняється тим, що працює без встановлення з'єднання [4.1].

UDP — це один з найпростіших протоколів транспортного рівня моделі OSI, котрий виконує обмін повідомленнями (датаграмами — datagram) без підтвердження та гарантії доставки. При використанні протоколу UDP відповідальність за обробку помилок і повторну передачу даних покладена на протокол рівнем вище. Але попри всі недоліки, протокол UDP є ефективним для серверів, що надсилають невеликі відповіді великій кількості клієнтів [4.1].

- Протокол UDP використовують такі сервіси та протоколи вищого рівня:
- TFTP (*Trivial File Transfer Protocol*, найпростіший протокол передачі файлів),
- SNMP (*Simple Network Management Protocol*, простий протокол управління мережею),
- DHCP (*Dynamic Host Configuration Protocol*, протокол динамічної конфігурації вузла),
- DNS (*Domain Name System*, служба доменних імен).
- Також цей протокол може використовуватися для різноманітних мережевих ігор реального часу, потокового відео та аудіо, інших типів даних.

UDP є одним з найпростіших протоколів транспортного рівня моделі OSI. Його детальний опис можна знайти в IETF RFC 768. UDP забезпечує дуже простий інтерфейс між мережним та програмним рівнями. UDP не гарантує доставку повідомлень, та відправник не запам'ятовує стан вже відісланих повідомлень. З цієї причини протокол UDP іноді розшифровують як «Unreliable

Datagram Protocol» (протокол ненадійних датаграм). Якщо на базі UDP треба організувати надійну передачу даних, то для цього необхідно залучити протоколи більш високого рівня.

Заголовок UDP-конверту складається з 4 полів, з яких 2 є опціональними.

«Порт відправника» та «контрольна сума» — це 16-бітні поля, котрі ідентифікують відправляючий та одержуючий процеси.

«Порт відправника» є необов'язковим, оскільки UDP працює без встановлення з'єднання та відправник може не потребувати відповіді. В такій ситуації «порт відправника» повинен дорівнювати нулю. Поле «Розмір» є обов'язковим, воно визначає довжину усієї UDP-дейтаграми в байтах, з полем «Дані» включно. Мінімальне значення цього поля дорівнює 8 байт.

Останнє поле заголовка довжиною 16 біт містить у собі контрольну суму заголовка і поля даних. «Контрольна сума» теж є необов'язковим полем, але на практиці воно майже завжди використовується.

Програми, що використовують UDP як транспортний протокол, мають бути готові до помилок, втрати деяких конвертів та повторної передачі даних. Деякі програми, такі як TFTP, можуть використовувати додаткові програмні механізми для підвищення надійності передачі. Але у більшості випадків для таких програм надійність не є необхідною і може навіть завадити уповільненням зв'язку. Потокове відео, ігри реального часу та VoIP (голос поверх IP) є прикладами програм, що дуже часто використовують UDP. Якщо ж програма потребує високого рівня надійності, то може використовуватися такий протокол як TCP або надлишковість коду, за допомогою якої можна знаходити помилки при передачі даних.

Оскільки у протоколі UDP відсутній будь-який контрольний механізм запобігання перевантаженням, мережні механізми повинні мати засоби для зменшення ефекту потенційних перевантажень від великого, неконтрольованого потоку UDP-трафіку. Кажучи інакше, оскільки UDP-відправники не спроможні виявляти перевантаженість, єдиним інструментом

для призупинення надмірного UDP-трафіку залишаються мережні елементи, такі як роутери, що використовують «черги конвертів» та «відкидання конвертів». DCCP (Datagram Congestion Control Protocol, протокол контролю навантаженості датаграм) був створений як часткове рішення цієї проблеми. Він контролює навантаження на кінцевих вузлах високошвидкісних потоків UDP-трафіку, наприклад, потокового відео[4.1].

Хоча кількість *UDP*-трафіку в типовій мережі сягає лишень кількох відсотків, проте багато важливих програм використовують *UDP*. Серед них:

- DNS (Domain Name System, служба доменних імен),
- SNMP (*Simple Network Management Protocol*, простий протокол управління мережею),
- DHCP (*Dynamic Host Configuration Protocol*, протокол динамічної конфігурації вузла),
- RIP (*Routing Information Protocol*, протокол маршрутизації інформації) та багато інших.

4.2. SCTP-протокол передачі з керуванням потоком

SCTP (Stream Control Transmission Protocol — «протокол передачі з керуванням потоком») — протокол транспортного рівня в комп'ютерних мережах, створений в 2000 році в IETF. RFC 3286 містить технічний вступ до нього [4.1].

Як і будь-який інший протокол передачі даних транспортного рівня, SCTP працює аналогічно TCP або UDP. TCP і UDP працюють настільки по-різному, що проводити аналогію до них обох некоректно. Вся аналогія — в тому, що SCTP, TCP і UDP відносяться до одного і того ж рівня моделі OSI. Але насправді SCTP має в арсеналі широкий спектр нововведень, таких як багатопоточність, захист від SYN-flood, а також синхронне з'єднання між двома хостами на двох і більше незалежних фізичних каналах (multi-homing).

Необхідне безпечне встановлення підключення в SCTP. Створення нового підключення в протоколах TCP і SCTP відбувається за допомогою механізму підтвердження пакетів.

У протоколі TCP ця процедура отримала назву триетапне підтвердження (three-way handshake). Клієнт посилає пакет SYN (Synchronize). Сервер відповідає пакетом SYN-ACK (Synchronize-Acknowledge). Клієнт підтверджує прийом пакета SYN-ACK пакетом ACK. На цьому процедура встановлення з'єднання завершується [4.2].

Протокол TCP має потенційну вразливість, обумовлену тим, що порушник, встановивши фальшиву IP-адресу відправника, може послати серверу безліч пакетів SYN. При отриманні пакету SYN сервер виділяє частину своїх ресурсів для встановлення нового з'єднання. Обробка безлічі пакетів SYN рано чи пізно використає всі ресурси сервера і зробить неможливим обробку нових запитів. Така атака отримала назву «відмова в обслуговуванні» (Denial of Service, або скорочено DoS).

Протокол SCTP захищений від подібних атак за допомогою механізму чотирьохетапного підтвердження (four-way handshake) і введенням маркера (cookie). За протоколом SCTP клієнт починає процедуру встановлення з'єднання посилкою пакета INIT. У відповідь сервер посилає пакет INIT-ACK, який містить маркер (унікальний ключ, що ідентифікує нове з'єднання). Потім клієнт відповідає посилкою пакета COOKIE-ECHO, в якому міститься маркер, посланий сервером. Тільки після цього сервер виділяє свої ресурси новому підключенню і підтверджує це відправленням клієнту пакету COOKIE-ACK.

Для вирішення проблеми затримки пересилання даних при виконанні процедури чотирьохетапного підтвердження в протоколі SCTP допускається включення даних в пакети COOKIE-ECHO і COOKIE-ACK.

При передачі даних в SCTP є відмінності між процедурою закриття сокетів протоколу SCTP і процедурою часткового закриття (half-close) протоколу TCP [4.3].

У протоколі TCP можлива ситуація, коли вузол закриває у себе сокет (виконуючи посилку пакету FIN), але продовжує приймати дані. Пакет FIN вказує кореспонденту на відсутність даних для передачі, проте до тих пір, поки кореспондент не закриє свій сокет, він може продовжувати передавати дані. Стан часткового закриття використовується додатками вкрай рідко, тому розробники протоколу SCTP вважали за потрібне замінити його послідовністю повідомлень для розриву існуючої асоціації. Коли вузол закриває свій сокет (надсилає повідомлення SHUTDOWN), обидва кореспонденти повинні припинити передачу даних, при цьому дозволяється лише обмін пакетами, що підтверджують прийом раніше відправлених даних.

TCP керує послідовністю байт: дані, послані застосунком-відправником, мають надходити до застосунку-одержувача виключно в тому ж порядку (у той час як протокол IP здатний змінити послідовність пакетів; крім того, зниклі пакети надсилаються повторно і зазвичай прибувають до одержувача з порушенням послідовності; для боротьби з цими явищами, дані накопичуються в буфері). SCTP може транспортувати дані між двома точками одночасно за кількома потоками повідомлень.

На противагу до TCP, SCTP обробляє цілі повідомлення, а не звичайні байти інформації. Це означає, що якщо відправник відсилає серверу повідомлення, що складається зі 100 байт за перший крок, а за ним ще 50 байт, то одержувач за перший крок отримає саме перші 100 байт в першому повідомленні, а тільки потім і тільки 50 байт на другий операції читання з сокета [4.3].

Термін «багатопоточність» (multi-streaming) означає здатність SCTP паралельно передавати по декільком незалежним потокам повідомлень. Наприклад, ми передаємо декілька фотографій через HTTP-застосунок (наприклад браузер). Можна використовувати для цього зв'язок з кількох TCP-з'єднань, однак також є допустимим SCTP-асоціація (SCTP-association), що керує декількома потоками повідомлень для цієї мети.

TCP досягає правильного порядку байт в потоці, абстрактно призначаючи порядковий номер кожного відісланої одиниці, а впорядковуючи прийняті байти, використовуючи призначені порядкові номери, у міру їхнього прибування. З іншого боку, SCTP присвоює різні порядкові номери повідомленням, відісланим в конкретному потоці. Це дозволяє незалежне упорядкування повідомлень з різних потоків. Так чи інакше, багатопоточність є опцією в SCTP. В залежності від бажань користувацького застосунку, повідомлення можуть бути оброблені не в порядку їхнього відправлення, а в порядку їхнього надходження.

Переваги використання *SCTP* включають в себе [4.1, 4.4]:

- *Використання множинних інтерфейсів (Multihoming)*

Припустимо, у нас є два хоста і хоча б один з них має декілька мережевих інтерфейсів і, відповідно, кілька *IP*-адрес. У *TCP*, поняття «з'єднання» означає обмін даними між двома точками, в той час, як в *SCTP* має місце концепція «асоціації» (*association*), що позначає все, що відбувається між двома хостами.

- *Потік*

Дані приходять в точку з незалежних потоків. Це дозволяє усунути феномен *en: Head-of-line blocking*, яким так страждає *TCP*.

Пошук шляху з моніторингом.

Протоколом вибирається первинний маршрут передачі даних, а також проводиться перевірка і моніторинг зв'язності шляху.

Механізми перевірки дійсності.

Захист адресата від flood-атак (технологія 4-way handshake), і повідомлення про втрачені пакети і порушені ланцюжки.

Покращена система контролю помилок, що підходить для jumbo-пакетів в Ethernet.

Частина переваг випливає з того факту, що спочатку розробники SCTP проектували протокол під потреби передачі телефонії (SS7) за протоколом IP.

Протокол TCP надає основні засоби для передачі даних по

мережі Internet по надійному шляху [4.1, 4.3]. Однак TCP накладає деякі обмеження на транспорт даних:

1. TCP надає надійну передачу даних у суворій послідовності. Проте одні додатки вимагають передачу без управління і контролю послідовності, а інші будуть цілком задоволені частковою впорядкованістю даних. Обидва цих випадки страждають через непотрібність затримок, пов'язаних з відновленням і впорядкуванням порушених послідовностей TCP.

2. Природа TCP орієнтована на потік байт, що викликає незручності. Програми змушені самостійно додавати власні маркери в пакети, щоб розпаралелити передачу власних повідомлень, а також використовувати додаткові хитрування, щоб переконатися в тому, що ціле повідомлення було доставлено за певний час.

3. Обмежені рамки можливостей TCP- сокетів ще більше ускладнюють завдання надання можливості паралельної передачі інформації до хоста по декількох каналах зв'язку (див.multi-homingвище).

TCP щодо уразлививості до атак класу «Відмова в обслуговуванні» (DoS), таким як SYN-flood.

Всі ці обмеження наносять шкоду продуктивності роботи телефонних мереж через IP.

SCTP був розроблений з деякими функціями дозволяють підвищити безпеку, такими як "4-х кратне рукостискання" (у порівнянні з "триразовим рукостисканням" в TCP), щоб запобігти SYN-flood атаки, і великих Cookie для перевірки автентичності асоціації.

Надійність була одним з ключових аспектів розробки безпеки протоколу SCTP. Multi-homing дозволяє асоціації залишатися відкритою, навіть якщо деякі використовувані маршрути і інтерфейси стали недоступні. Це має особливе значення, який використовуючи SCTP, передає повідомлення та сервіси протоколів ОКС-7 поверх IP мережі, що вимагає сильної стійкості під час відключень лінків для підтримки телекомунікаційних послуг, навіть при

серйозних аномаліях в мережі [4.1-4.4].

Шифрування не є частиною оригінального дизайну SCTP. SCTP іноді є гарним кандидатом для перевірки на міцність стека TCP / IP. Деякі операційні системи поширюються з підтримкою протоколу SCTP, але зважаючи на його слабку популярності (в порівнянні з TCP або UDP), іноді забувають налаштувати в брандмауері виявлення вторгнень, що дає можливості для сканування трафіку.

Таблиця 4.1. Порівняння можливостей протоколів транспортного рівня

Параметр	UDP	TCP	SCTP
Встановлення зв'язку	Ні	Так	Так
Надійна передача	Ні	Так	Так
Збереження меж повідомлення	Так	Ні	Так
Впорядкована доставка	Ні	Так	Так
Невпорядкована доставка	Так	Ні	Так
Контрольні суми даних	Так	Так	Так
Розмір контрольної суми (біт)	16	16	32
Шлях MTU	Ні	Так	Так
Управління нагромадженням	Ні	Так	Так
Багатонитевість	Ні	Ні	Так
Підтримка декількох інтерфейсів	Ні	Ні	Так
Поєднання потоків	Ні	Так	Так

При формуванні кадрів повідомлення забезпечується збереження кордонів повідомлення в тому вигляді, в якому воно передається сокетами; це означає, що якщо клієнт посилає серверу 100 байт, за якими йдуть 50 байт, то сервер сприймає 100 байт і 50 байт за дві операції читання. Точно так же функціонує протокол UDP, це є особливістю протоколів, орієнтованих на роботу з повідомленнями.

На противагу їм протокол TCP обробляє неструктурований потік байт. Якщо не використовувати процедуру формування кадрів повідомлення, то вузол мережі може отримувати дані за розміром більше або менше відправлених. Такий режим функціонування вимагає, щоб для протоколів,

орієнтованих на роботу з повідомленнями і функціонуючих поверх протоколу TCP, на прикладному рівні було надано спеціальний буфер даних і виконувалася процедура формування кадрів повідомлень (що потенційно є складним завданням). Протокол SCTP забезпечує формування кадрів при передачі даних. Коли вузол виконує запис у сокет, його кореспондент гарантовано отримує блок даних того ж розміру [4.1-4.5].

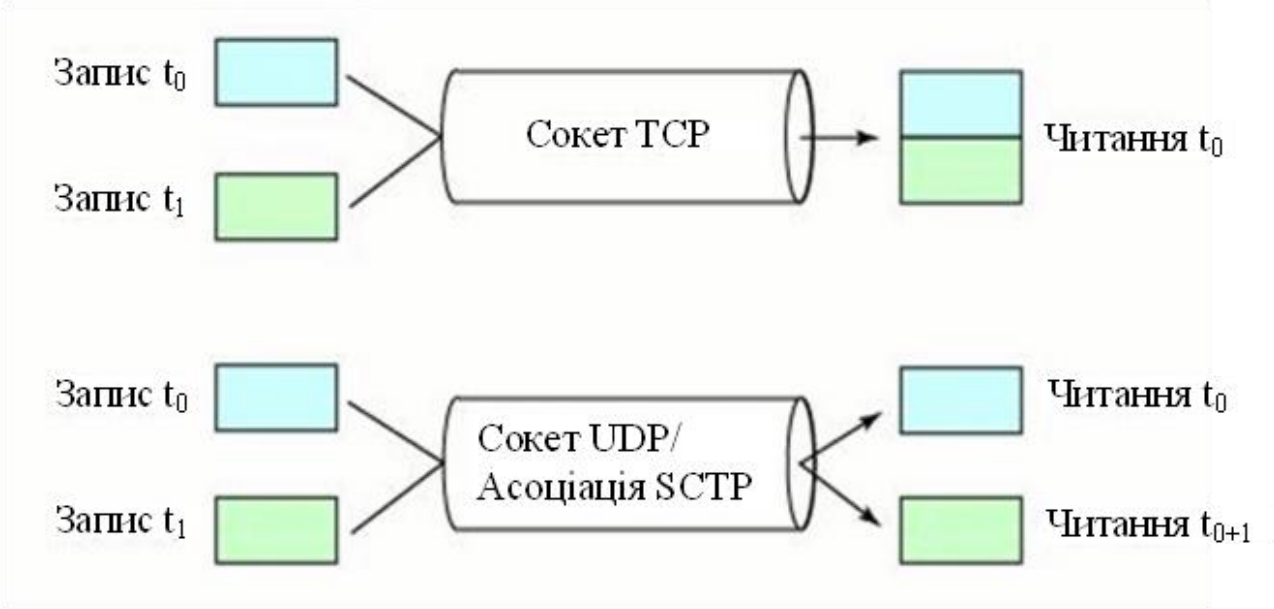


Рис. 4.1. Формування кадрів при передачі даних, що забезпечує протокол SCTP

Біти	Біти 0 - 7	8 - 15	6 - 23	24 - 31
+0	Порт джерела		Порт призначення	
32	Тег перевірки			
64	Контрольна сума			
96	Тип 1 блока	Прапорці 1 блока	Довжина 1 блока	
128	Дані блока			
…	…			
…	Тип N блока	Прапорці N блока	Довжина N блока	
…	Дані N блока			

Рис. 4.2. Структура пакета SCTP

SCTP пакети мають простішу структуру, ніж пакети TCP. Кожен пакет складається з двох основних розділів:

Загальний заголовок, який займає перші 12 байт (виділені синім кольором). Блоки даних, які займають решту пакету.

Перший блок відзначений зеленим кольором, і останній з блоків N (N блок) виділено червоним.

Кожен блок має ідентифікатор типу довжиною 1 байт, що дозволяє визначити 255 різних типів блоків. RFC 4960 визначає список видів блоків, всього наразі визначено 15 типів. Інша частина блоку складається з двох байт (максимальний розмір 65535 байт) і даних. Якщо блок не утворює 4 байта, то вона неявно заповнюється нулями, які не включені в довжину блоку.

Повторна передача блоків DATA може бути обумовлена тайм-аутом, визначеним таймером повтору (retransmission timer) або отриманням SACK, що показують що блок DATA не був отриманий адресатом. Для зниження ймовірності насичення повтор передачі блоків DATA обмежується. Значення тайм-ауту для повтору (RTO) встановлюється на основі оцінки часу кругового обходу і зменшується експоненціально з ростом частоти втрати повідомлень. Для активних асоціацій з майже постійним рівнем трафіку DATA причиною повтору скоріше за все будуть повідомлення SACK, а не тайм-аут. Для зниження ймовірності непотрібних повторів використовується правило 4 SACK, відповідно до якого повтор передачі відбувається тільки по четвертому SACK, яка вказує на пропуск блоку даних. Це дозволяє запобігти повтору передачі, викликані порушенням порядку доставки.

Безумовно відбувається збій в дорозі.

Підтримується лічильник для числа повторів передачі за конкретною адресою одержувача без підтвердження успішної доставки.

Коли значення цього лічильника досягає заданого порогу (конфігураційний параметр), адреса оголошується неактивною і протокол SCTP починає використовувати іншу адресу для передачі блоків DATA.

Крім того, по всіх невживаним (додатковим) адресами періодично передаються спеціальні блоки Heartbeat і підтримується лічильник числа блоків Heartbeat, переданих без повернення відповідного Heartbeat Ack.

- Протокол SCTP реалізований в наступних операційних системах:
- Linux 2.4 і вище
- Sun Solaris 10
- Cisco IOS 12 +
- DragonFly BSD починаючи з версії 1.4
- QNX Neutrino,
- BSD UNIX (з зовнішнім доповненням від проекту KAME)
- FreeBSD починаючи з версії 7
- HP-UX
- AIX 5

4.3. TCP - протокол керування передачею

TCP (Transmission Control Protocol) - протоко́л керува́ння переда́чею — разом із протоколом IP є стрижневим протоколом Інтернету, який дав назву моделі TCP/IP. Протокол призначений для управління передачею даних у комп'ютерних мережах, працює на транспортному рівні моделі OSI [4.1-4.6].

На відміну від іншого поширеного протоколу транспортного рівня UDP, TCP забезпечує надійне доправляння даних від хоста-відправника до хоста-отримувача, для цього встановлюється логічний зв'язок між хостами. Таким чином TCP належить до класу протоколів зі встановленим з'єднанням.

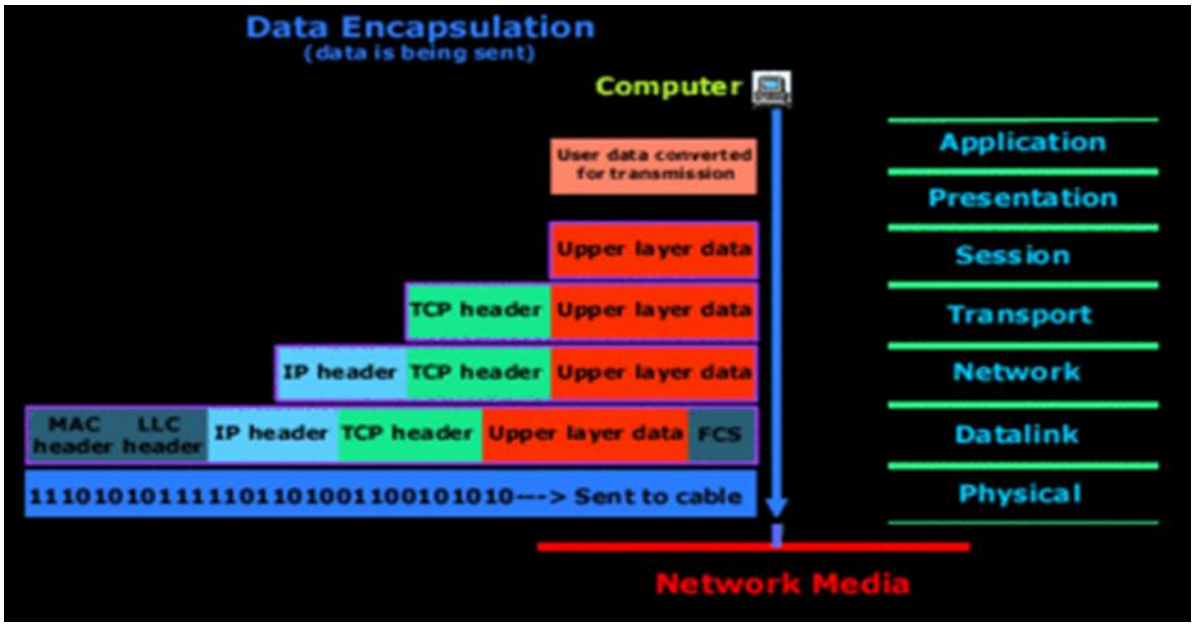


Рис. 4.3. Інкапсуляція TCP в моделі OSI

TCP отримує потоки даних від протоколів верхніх рівнів OSI-моделі, початковим джерелом яких є протоколи прикладного рівня, такі як HTTP, FTP та інші. Кожний протокол верхнього рівня має свій визначений TCP-порт [4.1, 4.5].

TCP розбиває конкретний потік даних на порції, та додає до кожної з них заголовок з номером послідовності. Отримані таким чином порції даних традиційно називаються TCP-сегментами. Далі кожний сегмент інкапсулюється в IP-пакет і передається через IP-протокол до хоста-отримувача.

Після надходження IP-пакету до хоста-отримувача перевіряється коректність отриманих даних у TCP-сегменті, методом перерахування контрольної суми, та переконується, що попередні сегменти даних також були успішно отримані. Після чого хост-отримувач надсилає запит до хоста-відправника про нову, або повторну передачу порції даних, що одночасно є підтвердженням того, що всі сегменти з номерами послідовності, меншими ніж номер нового запиту, були успішно отримані.

У свою чергу TCP-сегменти деінкапсулюються з IP-пакетів, розміщуються в правильному порядку та з них вилучаються TCP-заголовки. Отриманий таким чином потік даних передається до того протоколу верхнього рівня, з якого первісно надійшли дані на стороні хоста-відправника.

Трохи історії. Піонером у розробці сучасних комп'ютерних мереж вважається Агентство передових оборонних дослідницьких проектів США, (DARPA, Defense Advanced Research Projects Agency) зі своєю розробкою - мережею ARPANET запущеною у 1969 році.

Появою протоколу вважають 1974 рік, коли інститут інженерів з електротехніки та електроніки опублікував роботу «Протокол для пакетної мережевої комунікації (A Protocol for Packet Network Intercommunication)», авторами якої були Вінтон Серф та Роберт Елліот Кан. У цій праці TCP було абревіатурою від Transmission Control Program (програма керування передачею).

Блок даних (PDU, Protocol data unit) TCP називається сегментом, хоча часто також використовують слово пакет, але таке вживання може вносити плутанину з IP-пакетом.

TCP-сегмент складається із TCP-заголовка і поля Дані (Data), яке називають сегментом даних або пейлодом або SDU.

Стандартний розмір TCP-заголовка — 20 байт, але з використанням опцій розмір може зростати до 60 байт. Як правило, опціями хости обмінюються на етапі встановлення з'єднання.

Розмір сегменту даних (поля даних) визначається опцією MSS (Максимальний розмір сегменту, Maximum segment size) на етапі встановлення з'єднання. Якщо обміну опціями не відбулося, то розмір сегменту даних встановлюється за замовчуванням 536 байт [4.1, 4.4-4.7].

Розмір сегменту даних тісно пов'язаний з MTU (Максимальний блок передачі). Фактично MSS дорівнює MTU з відніманням розміру IP- і TCP-заголовків. Наприклад у сучасній мережі Ethernet MTU дорівнює 1500 байт; тоді оптимальний розмір MSS буде 1460 байтів (1500 мінус 20 байт заголовка IP і 20 байт заголовка TCP).

Таблиця 4.2. Формат TCP сегменту

Формат TCP сегменту																															
Октет		0								1								2								3							
	Біт	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Порт джерела (*Source port*)																Порт призначення (*Destination port*)														
4	32	Номер послідовності (*Sequence number*)																															
8	64	Номер підтвердження (*Acknowledgment number*)																															
12	96	Зміщення даних (*Data offset*)				Зарезервовано (*Reserved*) **0 0 0**				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Розмір вікна (*Window Size*)													
16	128	Контрольна сума (*Checksum*)																	Покажчик важливості (*Urgent pointer*)													
20 ... 56	160 ... 448	Опції (*Options*) *необов'язкове, розмір залежно від значення поля «Зміщення даних»*																															
20 + ...	160 +	**Дані (*Data*)**																															

- **Порт джерела (**0 — 15 біти).

Порт джерела (Source port) ідентифікує номер *TCP*-порту, з якого відправляється сегмент

- **Порт призначення (**16 — 31 біти).

Порт призначення (Destination port) ідентифікує номер *TCP*-порту, на який відправляється сегмент.

- **Номер послідовності (**32 — 63 біти).

Номер послідовності (Sequence number) є числом, що відображає номер першого байту в сегменті надісланих даних від хоста-відправника до хоста-отримувача. Це число є акумулювальним, тобто поточний *номер послідовності* є сумою *номеру послідовності* попереднього сегменту і кількості даних (в байтах) відправлених у ньому. Використовується для відстежування кількості та правильної послідовності отриманих сегментів даних.

- **Номер підтвердження (**64 — 95 біти).

Номер підтвердження (Acknowledgment number) фактично є запитом від хоста отримувача на надіслання нового сегменту даних починаючи зі

вказаного номера. З іншого боку, коли хост відправник отримує це повідомлення, він переконується, що всі сегменти даних з номерами послідовності меншими за *номер підтвердження* були успішно прийняті отримувачем

- **Зміщення даних (**96 — 99 біти).

Зміщення даних (Data offset) 4-бітний номер, який визначає розмір TCP-заголовка в 32-бітових словах. Мінімальний розмір становить 5 (*0101*) слів, а максимальний — 15 (*1111*), що є відповідно 20 і 60 байт. Фактично визначає розмір поля **Опції** (*Options*) від 0 до 40 байт.

- Зарезервовано (100—102 біти), зарезервовані для майбутнього використання і повинні містити нулі (000).

Прапорці (керуючі біти). Це поле містить бітові прапорці, з яких шість основних описані в RFC 793 з 106 по 111 біт, два прапорці додані до заголовка в RFC 3168, розміщуються в 104 і 105 бітах заголовка, в 103 біті знаходиться експериментальний прапорець згідно з RFC 3540. Прапорці вважається встановленими, якщо їх бітове значення є 1.

- 103 NS — Одноразова сума (Nonce Sum), використовується з метою покращення роботи механізму явного повідомлення про перевантаження (Explicit Congestion Notification, ECN).
- 104 CWR — Вікно перевантаження зменшено (Congestion Window Reduced), прапорець встановлюється, щоб показати що TCP-сегмент був отриманий зі встановленим полем ECE, іншими словами це є підтвердженням отримання сегменту даних з прапорцем ECE від хоста партнера.
- 105 ECE — ECN-Exo (ECN-Echo), поле показує, що відправник підтримує ECN.

Основні поля заголовка:

- 106 URG — Важливість (Urgent), вказує, що TCP-сегмент містить важливі дані. Коли до хоста-отримувача надходить сегмент зі встановленим прапорцем URG, TCP відправляє важливі дані з цього сегменту,

які знаходяться завдяки полю покажчик важливості до відповідного протоколу верхнього рівня минаючи чергу і без перевірки успішності надходження попередніх сегментів.[4.1, 4.8];

- 107 ACK — Підтвердження (Acknowledge) успішності отримання TCP-сегменту;
- 108 PSH — Просування (Push), також як і прапорець URG, вказує, на пріоритетність TCP-сегменту. Хост-відправник позачергово надсилає цей сегмент даних через IP-мережу. За аналогією з прапорцем URG, PSH інструктує хост-отримувач, що сегмент даних має бути негайно переданий до прикладного рівня (кінцевого споживача даних).
- 109 RST — Обривання (Reset) вказує, хосту-отримувачу негайно скинути з'єднання без подальшої взаємодії. Така ситуація наступає у разі, якщо сервер (хост-відправник) не надає послуги визначеного сервісу. Наприклад клієнт (хост-отримувач) запросив у веб-сервера послуги у форматі протоколу HTTPS (TCP-порт 443), але веб-сервер надає послуги лише у форматі HTTP (TCP-порт 80). Ця властивість TCP часто використовується хакерами для сканування портів мережі жертви.
- 110 SYN — Синхронізація (Synchronize) використовується для встановлення з'єднання між хостами при так званому триходовому рукостисканні
- 111 FIN — Фініш (Finish) вказує на завершення з'єднання.

- **Розмір вікна (**112—127 біти).
  - *Розмір вікна* (*Window Size*) визначає кількість байтів даних, які відправник може надіслати до того, як отримає підтвердження (запит на новий сегмент) від хоста-отримувача. На практиці це означає, що хост-відправник може надсилати певну кількість сегментів даних без отримання підтвердження від хоста-отримувача.
  - *Розмір вікна TCP* вираховує на основі максимальної пропускної здатності (*bandwidth*) лінії зв'язку між хостами (фактично це є *пропускна*

здатність відрізка шляху з її найгіршим значенням) та загальній затримці (*latency*) (часу потрібному на доставку сегмента) на всьому шляху.

- **Контрольна сума (**128—143 біти). *Контрольна сума (Checksum)* розраховується на основі усього *TCP*-сегменту включно із заголовком та важливих полів IP-пакету: IP-адрес хостів відправника та отримувача, номеру протоколу (*TCP* має номер 6) та загального розміру IP-пакету. Контрольна сума забезпечує можливість перевірки цілісності надісланих даних

- **Покажчик важливості (**144—159 біти).

- *Покажчик важливості (Urgent pointer)*. Поле береться до уваги тільки в разі встановленого прапорця *URG*, та містить значення зміщення відносно номеру послідовності сегменту. Фактично це число вказує на позицію в *TCP*-сегменті де закінчуються *важливі дані*. Тобто *важливі дані* знаходяться одразу після *TCP*-заголовка і закінчуються перед місцем на яке вказує *покажчик важливості*.

Опції (160—479 біти). Опції (Options) необов'язкове поле, розмір якого визначається в залежності від значення поля зміщення даних та є кратним 8 (одному байту).

Кожна опція в свою чергу складається з 3-х полів: Номер (kind) — 1 байт, Довжина (length, вказує на загальний розмір опції в байтах) — 1 байт, Дані (data) в залежності від поля довжина. Опції використовується для обміну додаткових параметрів між хостами з метою покращення функціонування протоколу TCP. Частіше за все це поле включає наступні опції:

MSS (Максимальний розмір сегменту, Maximum segment size), RFC 793, номер — 2, довжина — 4. Опція максимальний розмір сегменту визначає максимальний розмір поля Дані в TCP-сегменті тобто кількість даних які можуть бути поміщені в один сегмент при їх передачі між хостами [4.1, 4.8].

Масштабування вікна (Window scale), RFC 7323, номер — 3, довжина — 3, слугує для збільшення значення TCP-вікна, максимальне значення цієї опції є

14. Новий розмір TCP-вікна вираховується по формулі: розмір вікна * 2n, де n є значення опції масштабування вікна. Стандартний максимальний розмір вікна відображається 16-ти бітовим числом, тобто може мати максимальне значення — 65535 (64 Кб), використовуючи опцію масштабування вікна з максимально допустимим значенням 14, отримуємо 65535 * 214 = 65535 * 16384 = 1073725440 (1 Гб). Великі значення TCP-вікна використовуються коли на шляху пакетів із TCP-сегментами зустрічаються WAN-лінки зі значними максимальними пропускними здатностями (bandwidth) та великими затримками (latency).

Вибіркові підтвердження (Selective Acknowledgments, SACK), RFC 2018, номер — 2, довжина — від 4 байт — верхня межа варіюється, як правило містить у собі два 2-х байтних поля даних. Мета її введення є покращення ефективності роботи TCP, як відомо TCP для передачі сегментів даних використовує протокол IP, який є протоколом без встановлення з'єднання, тобто доправлення пакетів з TCP-сегментами не є гарантованим, допускається, що частина IP-пакетів може бути втрачена. В свою чергу TCP забезпечує надійне доправляння даних, що базується на механізмі надсилання номерів підтвердження (acknowledgment number). Якщо якийсь сегмент від відправника до отримувача не надійшов у встановлений час то ініціюється повторна передача починаючи зі втраченого сегменту, навіть якщо TCP-сегменти з номерами послідовності більшими за номер втраченого сегменту були успішно отримані [4.1-4.4, 4.9]. Механізм вибіркового підтвердження дозволяє ретранслювати лише втрачені сегменти даних, чим суттєво покращує ефективність роботи TCP.

Мітки часу (Timestamps), RFC 7323, номер — 8, довжина — 10, містить у собі два 4-х байтних поля Значення мітки часу (Timestamp Value) та Exo-відповідь мітки часу (Timestamp Echo Reply) [4.1-24.6, 4.10].

Як правило хости обмінюються значеннями міток часу на етапі встановлення з'єднання. За допомогою міток часу TCP визначає скільки

потрібно часу на доправлення сегментів між хостами. На основі цих значень встановлюються TCP-таймери відповідальні на стороні хоста-відправника за повторну передачу даних, якщо підтвердження отримання не надійшло у встановлений час, а у разі використання опції вибіркового підтвердження хост-отримувач самостійно ініціює запит на повторну передачу конкретного сегменту даних.

Якщо деякий простір поля Опції лишається незаповненим то він заповнюється спеціальною опцією NOP (No-Operation, нічого не робити), RFC 793, номер — 1, довжина — відсутня.[4.10].

Поняття порт (port) є ключовим у протоколі TCP. Порт з точки зору операційних систем називається сокетом (socket) процесу. Іншими словами це програмний інтерфейс для забезпечення обміну даними між процесами.

Для розуміння поняття порту з точки зору комп'ютерних мереж легко провести аналогію з роботою звичайної пошти. Коли ви відправляєте листа, то заповнюєте поля Куди: адреса будинку і Кому: людина, яка проживає у цьому будинку. У мережі Інтернет на питання Куди: відповідає IP-протокол, тобто це є IP-адреса хоста, а на питання Кому: відповідає TCP-протокол (або інший протокол транспортного рівня), тобто це номер протоколу прикладного рівня (процес, що відповідає за цей протокол з точки зору операційної системи), який «мешкає» за вказаною IP-адресою хоста.

За аналогією зі звичайною поштою ви можете направити 2-а листа 2-м різним людям, які живуть в одному будинку, в Інтернеті ви можете направити TCP-сегменти 2-м різним протоколам прикладного рівня за однією і тією ж самою IP-адресою. Звичайно треба заповнити і зворотню адресу.

В Інтернеті широко вживається форма запису : ip-адреса: порт.

Команда netstat -n може показати наступне:

192.168.1.31:54132 198.35.26.96:443

192.168.1.31:54138 198.35.26.96:22

Тобто якийсь хост з локальної мережі (використовує приватну IP-адресу)

має одночасно з'єднання з сервером в глобальній мережі Інтернет з ip-адресою 198.35.26.96 по портам 443 (протокол HTTPS) та 22 (протокол SSH).

За надання і використання *портів* відповідальна IANA, хоча на практиці на відміну від використання IP-адрес це правило не завжди дотримується. Наприклад нічого не заважає адміністраторам 2-х хостів домовитися передавати, якийсь тип IP-трафіку по *порту* 80, який є закріплений за протоколом http і є відкритим на більшості файрволів [4.1,4.3].

TCP-заголовок має 16-бітові поля порт джерела та порт призначення, які можуть приймати значення від 0 до 65535.

- Порти поділяються на категорії:
- **Загальновідомі** (*well-known*),
- **Зареєстровані** (*registered*),
- **Динамічні** (*приватні, ефемерні, dynamic, private, ephemeral*),

Загальновідомі (well-known), діапазон номерів 0 — 1023. Як правило це порти, які використовують протоколи прикладного рівня TCP/IP стеку затверджені IETF, згідно з принципами відкритого стандарту. Наприклад: HTTP порт −80, HTTPS — 443, SMTP — 25, Telnet — 23, SSH — 22.

Зареєстровані (registered), діапазон номерів 1024 — 49151. За задумом, ці порти призначаються для протоколів різних виробників програмного забезпечення, які мають їх зареєструвати в IANA. Наприклад ігровий сервіс Xbox Live використовує зареєстрований порт 3074. На практиці багато портів у цьому діапазоні використовуються без офіційної реєстрації [4.1, 4.11].

Динамічні (приватні, ефемерні, dynamic, private, ephemeral), діапазон номерів 49152-65535. IANA не реєструє ці порти. Використовуються, як правило хостами-клієнтами, для встановлення TCP-з'єднань з хостами-серверами. Як у прикладі вище, коли клієнт з IP-адресою 192.168.1.31 встановив 2 з'єднання зі сервером 198.35.26.96, на стороні клієнта використовуються динамічні порти 54132 та 54138, а на стороні сервера загальновідомі порти 443

(протокол HTTPS) та 22 (протокол SSH).

Задачею протоколу, як випливає з його назви: «протокол керування передачею», є контроль надійної передачі даних між хостами, для забезпечення цього між хостами встановлюється логічне з'єднання, яке зветься TCP-сесією.

Протокол TCP працює у форматі архітектури клієнт-сервер [4.4-4.8].

Хост який надсилає запит на отримання сервісу є клієнтом, той хто відповідає на запит зветься сервером. Хости зв'язуються один з одним за TCP-портами, на стороні клієнта це, як правило, динамічний порт, а на стороні сервера це загальновідомий або зареєстрований порт, номер якого відповідає протоколу прикладного рівня. Номери портів та значення інших параметрів заносяться до заголовка TCP-сегменту.

Тут під терміном сервер (server) розуміють комп'ютер під управлінням операційної системи, який має доступ до IP-мережі та надає послуги одного чи декількох сервісів прикладного рівня. В свою чергу на сервері-комп'ютері встановлені спеціальні сервер-програми, які забезпечують роботу протоколів прикладного рівня. Таким чином, якщо це веб-сервер, то на ньому мусить бути встановлена одна із таких програм, як наприклад: Apache HTTP Server.

На практиці сервер-комп'ютер надає послуги відразу декількох сервісів, тобто на ньому може бути встановлено більше однієї сервер-програми, що забезпечують роботу декількох протоколів прикладного рівня. Наприклад сервер-комп'ютер може бути одночасно веб-сервером і FTP-сервером та забезпечувати роботу протоколів HTTP, HTTPS та FTP [4.1, 4.8].

Важливо розуміти, що протягом TCP-сесії дані надсилаються в обох напрямках, як від сервера до клієнта так і від клієнта до сервера, тобто створюються два потоки даних. Причому не завжди більший потік даних прямує від сервера до клієнта.

Кожна окрема сесія роботи протоколу *TCP* може бути поділена на три фази:

- Встановлення з'єднання
- Передача даних
- Закінчення з'єднання

Встановлення з'єднання:

Для встановлення з'єднання протокол TCP використовує триходове рукостискання (handshaking), назване так за кількістю повідомлень між хостами:

Клієнт формує TCP-заголовок: у поле порт джерела заносить свій номер TCP-порта, як правило динамічний, у поле порт призначення номер порту протоколу прикладного рівня, послуги якого хоче отримати, в поле номер послідовності сегменту довільне значення та встановлює прапорець SYN. Сформований таким чином TCP-сегмент відправляється серверу. TCP-сесія на стороні клієнта переходить у стан SYN-SENT.

Сервер отримує TCP-сегмент від клієнта зі встановленим прапорцем SYN та у відповідь формує TCP-заголовок: у поле порт джерела заносить свій номер TCP-порта, у поле порт призначення номер порту клієнта. Додає до отриманого від клієнта номера послідовності сегменту 1 і поміщає отримане число до номеру підтвердження, вносить свій власний початковий номер послідовності сегменту та відправляє сегмент до клієнта з прапорцями SYN та ACK. TCP-сесія на стороні сервера переходить у стан SYN-RECEIVED.

Після отримання TCP-сегмента зі встановленими прапорцями ACK та SYN клієнт переходить у стан ESTABLISHED та відповідає на запит сервера про синхронізацію шляхом додавання 1 до номера отриманої послідовності сегменту та поміщає це число до номера підтвердження. Далі клієнт надсилає таким чином сформований сегмент до сервера з прапорцем ACK

Після отримання сервером TCP-сегмента з прапорцем ACK стан TCP-сесії на його стороні стає також ESTABLISHED, разом з чим розпочинається передача даних [4.1-4.4, 4.13].

Також на етапі встановлення з'єднання між хостами, як правило відбувається обмін опціями, тобто TCP-параметрами, які впливають на ефективність передачі даних.

Закінчення з'єднання:

Ініціатором закінчення з'єднання може бути, як клієнт так і сервер. Для закінчення TCP-сесії використовується так зване чотириходове рукостискання (four-way handshake).

Ініціатор розірвання з'єднання направляє своєму партнеру TCP-сегмент зі встановленим прапорцем FIN. TCP-сесія ініціатора переходить зі стану ESTABLISHED у стан FIN-WAIT-1

Хост-отримувач приймає FIN від ініціатора та посилає у відповідь TCP-сегмент зі встановленим прапорцем ACK. TCP-сесія отримувача переходить зі стану ESTABLISHED у стан CLOSE-WAIT. З набуттям хостом цього стану TCP припиняє отримувати нові запити, на передачу даних, від відповідного протоколу верхнього рівня та встановлює таймер на завершення попередніх запитів. Ініціатор отримує ACK та переходить у стан FIN-WAIT-2.

Після закінчення оброблення всіх запитів протоколів верхнього рівня хост-отримувач переходить у стан LAST-ACK та відправляє ініціатору TCP-сегмент зі встановленим прапорцем FIN.

Ініціатор приймає FIN від отримувача та посилає у відповідь TCP-сегмент зі встановленим прапорцем ACK. Отримувач приймає ACK та переходить у стан CLOSED.

Ініціатор розірвання з'єднання чекає протягом подвійного часу від MSL (максимального життя сегмента, maximum segment lifetime), щоб переконатися, що посланий ACK був отриманий та також переходить у стан CLOSED.

4.4. RTCP протокол управління передачею в реальному часі

RTCP (RTP Control Protocol) - протокол управління передачею в реальному часі - протокол, який використовується спільно з RTP. Протокол описаний в RFC 3550, механізм реагування на зміни в мережі [4.1, 4.13].

Наприклад, одержавши інформацію про підвищення інтенсивності трафіка в мережі й зменшенні виділеної цьому застосунку смуги пропущення, застосунок може вжити заходів і стримати свої вимоги до смуги пропущення за рахунок деякої втрати якості. Після зниження навантаження в мережі застосунок може відновити вихідну смугу пропущення й продовжити роботу з тією якістю, що воно надавало спочатку.

Під час сеансу RTP- необхідною є інформація зворотного зв'язку, оскільки дуже часто одержувачі не можуть приймати інформацію з визначеними якісними параметрами або ж у деяких ситуаціях виникає потреба динамічно зменшити швидкість чи інші параметри передавання.

Для одержання такої зворотної інформації, а також для виконання деяких функцій керування і призначено протокол RTCP (RFC 1889).

Він, як і RTP, використовує сервіс протоколу UDP і має свій номер порту.

Основні функції протоколу RTCP [4.1, 4.9, 4.13]:

- *надання інформації від одержувачів даних.* Протокол *RTCP* багатоадресний і всі одержувачі періодично розсилають усім учасникам сеансу інформацію про якість приймання інформації, нерівномірність потоку тощо. Це дає змогу діагностувати стан передавання й ухвалювати рішення щодо зміни параметрів системи;
- *одержання повнішої інформації про сеанс*. Пакети протоколу *RTP* містять тільки ідентифікатор джерела синхронізації, Протокол *RTCP* доповнює протокол *RTP* функціями передавання текстового опису відправника, стану та структури потоків сеансу, видів та форматів даних, які передаються;
- *масштабування сеансу.* Без масштабування зі збільшенням

кількості учасників сеансу кількість *RTCP-пакетів* могла б зайняти всю перепускну здатність каналів зв'язку і блокувати передавання первинної інформації. Тому важливою функцією протоколу *RTCP* є налагодження інтенсивності генерування пакетів залежно від кількості учасників сеансу, В цілому частка *RTCP-пакетів* не повинна перевищувати 5% від загального потоку сеансу.

Типи повідомлень протоколу RTCP [4.1]:

- *RTCP* виділяє кілька типів пакетів: доповідь відправника, звіт приймача, опис джерела, завершення сеансу і спеціальні прикладні повідомлення. Крім того, протокол є розширюваним і дозволяє певному додатку розширити *RTCP* пакет. Основним стандартом для розширення *RTCP* є RFC 3611.
- *Sender report (SR). Доповідь відправника*.

Доповідь відправника відправляється періодично за допомогою відправників конференції з метою звітування статистики про передавання ти приймання усіх *RTP*-пакетів, надісланих в певний проміжок часу. У доповіді відправник включає абсолютну позначку , яка відповідає кількості секунд , що пройшли з півночі 1 січня 1900. Абсолютна відмітка дозволяє приймачу синхронізувати *RTP*-пакети. Це особливо важливо, коли аудіо і відео передаються одночасно, тому що потоки аудіо і відео використовують незалежні відносні тимчасові мітки.

- *Receiver report (RR). Звіт приймача.*

Звіт приймача призначений для пасивних учасників, тим, котрі не надсилають RTP-пакети. Звіт інформує відправника та інших одержувачів про якість сервісу.

- *Source description (SDES) Опис джерела.*

Ці повідомлення використовуються для надсилання імен учасникам конференції. Вони також можуть використовуватись для надання додаткової інформації (ім'я, телефонний номер, e-mail власника джерела).

■ *End of participation (BYE). Завершення сеансу.*

Джерело надсилає *BYE* повідомлення для того, щоб закрити потік. Це дозволяє йому повідомити приймачів (кінцеві точки) про те, що він покидає конференцію. Хоча інші джерела можуть виявити відсутність джерела, дане повідомлення є прямим оголошення . Це також корисно для медіа- змішувачів.

■ *Application-specific message (APP). Спеціальні прикладні повідомлення.*

Повідомлення від конкретного додатка забезпечує механізм для розробки конкретних програм розширення протоколу *RTCP*.

4.5. SPX –протокол гарантує доставку пакета

SPX (Sequence Packet eXchange) і його вдосконалена модифікація SPX II являють собою транспортні протоколи 7-рівневої моделі OSI [4.2, 4.14].

Це протокол гарантує доставку пакета і використовує техніку ковзного вікна (віддалений аналог протоколу TCP). У разі втрати або помилки пакет даних пересилається повторно, число повторень задається програмно.

У протоколі SPX не передбачена широкомовна або мультикастинг-адресація. У SPX індукується ситуація, коли партнер несподівано перериває з'єднання, наприклад через обрив зв'язку.

Пакети SPX вкладаються в пакети IPX. При цьому в полі тип пакету IPX записується код 5. Заголовок пакета SPX завжди містить 42 байти, включаючи 30 байт заголовка IPX-пакета, куди він вкладений.

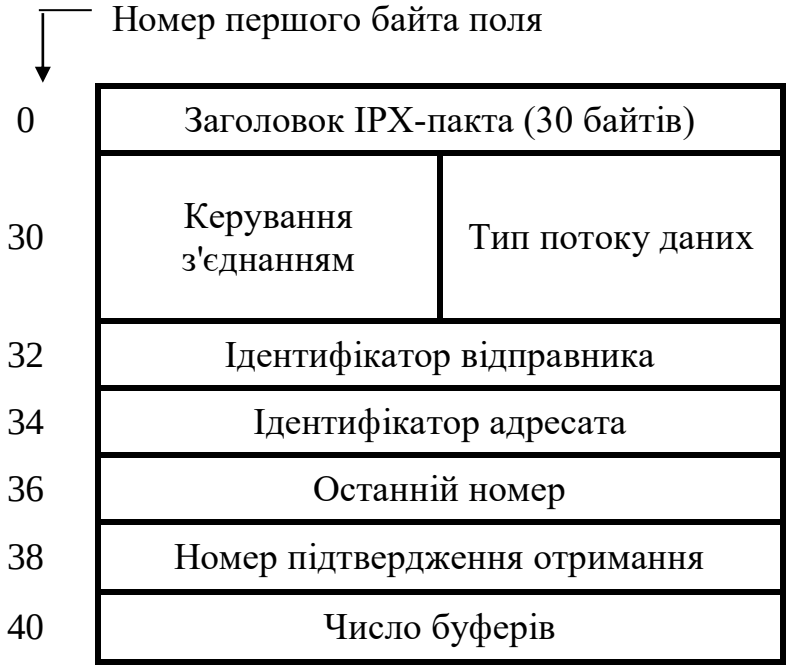


Рис. 4.4 Формат заголовка SPX

Поле *управління з'єднанням* визначає, чи є даний пакет системним чи прикладним [4.2, 4.14]. Це поле містить однобітові прапори, використовувані *SPX* і *SPX II* для керування потоком даних у віртуальному каналі:

- *0x01 XHD* Зарезервовано *SPX II* для розширення заголовків;
- *0x02 RES1* Призначення поля не визначено, має дорівнювати нулю;
- *0x04 NEG SPX II (SIZ)* узгоджує розмір запиту / відгуку, для spx має дорівнювати нулю;
- *0x08 SPX2* Тип пакету *SPX II*, для *SPX* має дорівнювати нулю;
- *0x10 EOM* Встановлюється клієнтом spx для індикації кінця повідомлення (*end-of-message*);
- *0x20 ATN* (attention) зарезервовано для спеціальних запитів (не підтримується *SPX*);
- *0x40 ACK* Встановлюється для запиту підтвердження отримання даного пакету. Запити і відгуки обробляються на рівні *SPX* (додаток не повинен змінювати цей код);
- *0x80 SYS* Встановлюється, якщо цей пакет є системним і служить

для підтвердження. Додатки не використовують пакети цього типу.

Поле *тип потоку даних* характеризує тип даних, поміщених в пакет. Значення цього поля перераховані нижче:

- *0x00-0x07* визначається клієнтом і може використовуватися в додатках;
- *0x80-0xfb* зарезервовані на майбутнє;
- *0xfc SPX II*, впорядковане звільнення запиту;
- *0xfd SPX II*, впорядковане звільнення підтвердження;
- *0xfe* вказує на закінчення зв'язку (*end-of-connection*). При закритті каналу spx-драйвер посилає клієнту пакет, де в полі тип потоку записаний даний код;
- *0xff* підтвердження отримання повідомлення про закінчення зв'язку (*end-of-connection-acknowledgment).* Цим кодом позначається пакет, що підтверджує закриття каналу, в прикладну програму такий пакет не передається

Є й інші поля. Поля ідентифікатора відправника і одержувача містять коди, що визначають учасників інформаційного обміну, присвоюються SPX-драйвером в момент встановлення зв'язку [4.1, 4.13-4.15]. У запитах на з'єднання це поле містить код 0xffff. Дане поле служить для забезпечення демультиплексування пакетів, що надходять на один і той же з'єднувач (socket).

Поле послідовний номер визначає число пакетів пересланих в одному напрямку. Кожен з партнерів обміну має свій лічильник, який скидається в нуль після досягнення 0xffff, після чого рахунок може продовжуватися. Для програми це поле, також як і наступні два, недоторкане. spx-пакети підтвердження містять в цьому полі порядковий номер останнього посланого пакета.

Поле номер підтвердження характеризує послідовний номер наступного пакета, який SPX очікує отримати. Будь-який пакет з порядковим номером менше, ніж задано в полі номера підтвердження, доставлений благополучно і не вимагає ретрансміссіі.

Поле число буферів служить для вказання числа доступних на станції буферів (буфери нумеруються, починаючи з 0, один буфер здатний прийняти один пакет) і використовується для організації управління потоком даних між додатками. Код цього поля інформує партнера про найбільший порядковий номер пакету, який може бути посланий. Протокол SPX посилає пакети до тих пір, доки "послідовний номер не стане рівним числу-вказівнику на віддаленій ЕОМ.

Алгоритм надсилання пакетів SPX є наступним. SPX-протокол не посилає наступний пакет до тих пір, поки не отримає підтвердження одержання попереднього. Хоча в протоколі SPX передбачений алгоритм ковзних вікон (як і в TCP), практично він в наш час не використовується, що цілком виправдано для локальних мереж. Для досить великих LAN, де пакет проходить через кілька перемикачів або маршрутизаторів, нехтування технікою вікон стає недозволеною розкішшю.

На випадок непередбачених обривів зв'язку в SPX є алгоритм "сторожова собака". Цей алгоритм реалізується спеціальною програмою, яка активується лише у разі, коли протягом певного часу в каналі відсутній трафік в будь-якому з напрямів (машина все зробила, а оператор заснув). У цьому випадку програма посилає спеціальні пакети і, якщо певна кількість спроб "достукатися" до партнера не увінчається успіхом, сесія переривається [4.1, 2.9, 4.15].

Якщо партнер не надсилає відгук за певний відрізок часу (RTT), проводиться повторна посилка пакету, при цьому RTT збільшується на 50%. Значення RTT не повинно перевищити величини max_retry_delay, яка за замовчуванням дорівнює 5 секундам.

Якщо зв'язок не відновилася, відправник намагається знайти інший маршрут до адресата. Якщо маршрут знайдений, лічильник спроб скидається в нуль і процедура відправки запускається знову. Допустима кількість спроб може лежати в діапазоні 1-255 (за замовчуванням - 10). При відсутності успіху сесія переривається.

Значення RTT визначається за часом відгуку найближчого маршрутизатора, який подвоюється, а до отриманої величини додається деяка константа.

Для кожної з сесій RTT визначається незалежно. Багато тимчасових констант задаються адміністратором мережі.

Протокол spx дозволяє здійснити від 100 до 2000 з'єднань одночасно (за замовчуванням це число дорівнює 1000).

Конфігураційні параметри SPX-протоколу (і мережі) зберігаються у файлах shell.cfg і net.cfg.

У 1992 році була розроблена нова версія SPX - SPX II. Головне удосконалення протоколу пов'язано із застосуванням пакетів більшого розміру. Раніше довгі SPX-пакети фрагментувалися і пересилалися по частинах, враховуючи, що черговий пакет може бути посланий лише після отримання підтвердження, неважко зрозуміти крайню неефективність такої схеми [4.1, 4.16]. Стандарт SPX дозволяє обмін пакетами з розміром, обмеженим тільки використовуваної мережевим середовищем.

Так в Ethernet пакет SPX II може мати довжину 1518 байт. Крім того, SPX II допускає використання технології вікон, тобто можна послати кілька кадрів, не чекаючи отримання підтвердження на кожен з вже посланих. Розмір вікна встановлюється відповідно до коду, що містяться в полі число-покажчик (число буферів / пакетів). При необхідності адміністратор може задати розмір вікна раз і назавжди.

Формат пакетів SPX II дещо відрізняється від SPX.

У SPX II збільшено число допустимих кодів для поля управління з'єднанням, введено додаткове поле в заголовок підтвердження (два байти, ім'я поля "розширене підтвердження").

Нове поле додано після поля число буферів.

Алгоритм встановлення зв'язку в SPX II відрізняться від варіанту SPX тим, що необхідно узгодити розмір пакетів, що пересилаються.

Рис. 4.5. Формат заголовка SPX-II

Використання протоколу SPX-II з Socket додатками.

Для програми з інтерфейсом Sockets API протокол SPX використовується автоматично в момент встановлення сімейство AF_NS. При підключенні AS / 400 SPX-II завжди просить партнера по з'єднанню використовувати протокол SPX-II. Якщо партнер підтримує тільки SPX або не хоче використовувати SPX-II, встановлюються тільки з'єднання SPX.

SPX-II містить два поліпшення, пов'язаних з продуктивністю, в порівнянні з SPX. Це можливість узгоджувати максимальні розміри пакетів від одного кінця до іншого і можливість відправляти кілька пакетів без явного підтвердження від віддаленої системи. Кількість пакетів, які можуть бути відправлені без підтвердження, називається розміром вікна для конкретного з'єднання [4.1, 4.17].

Існує непродуктивні витрати часу з'єднання, пов'язані з максимальним узгодженням розміру пакета, тому що обидві системи повинні обмінюватися цією інформацією в серії пакетів, пов'язаних з протоколом SPX-II.

Додаток може прийняти на себе накладні витрати на узгодження розміру пакета, вказавши опцію сокетов, або за замовчуванням він може обійти це узгодження.

Не погоджуючи максимальний розмір пакета, найбільший пакет, який можна відправити і отримати по з'єднанню SPX-II, становить 576 байт, що також є максимальним розміром пакета для з'єднань SPX.

Розмір вікна для конкретного з'єднання SPX-II не обмовляється. Він заснований на параметрі розміру вікна прийому SPX-II в описі IPX. Ні опцій сокетов, пов'язаних з розміром вікна.

Щоб дозволити узгодження розміру пакета, для вибору сокетов SO_MTU необхідно встановити значення ZERO, перш ніж видається виклик connect. Після того, як з'єднання встановлено, узгоджений максимальний розмір пакета буде повернений, якщо програма витягує значення SO_MTU через виклик функції getsockopt.

Якщо сервер AS / 400 прослуховує і очікує запит на підключення, з'єднання SPX-II буде встановлено, якщо клієнт запросить використання протоколу SPX-II. Узгодження розміру пакета також буде включено, якщо клієнт його запросить (в цьому випадку сервера не потрібно встановлювати SO_MTU в нуль).

Якщо ваші програми використовують SPX-II, опція сокетов SO_KEEPALIVE розглядається підтримкою SPX-II як ON, навіть якщо програма встановила її в OFF, тому що протокол SPX-II вимагає, щоб функція сторожового таймера була активна для всіх з'єднань SPX-II.

Використання маршрутизаторів Cisco в мережах Novell NetWare [4.18].

ОС NetWare включає в себе спеціальний контрольний протокол IPX Watchdog), що виробляє періодичний опитування неактивних сполук з робочими станціями і передає сервера повідомлення про збої цих сполук або про доступність тієї чи іншої робочої станції. Якщо звіт протоколу IPX Watchdog на якомусь із з'єднань не доступний, то сервер закриває цю сполуку.

Деякі програми, що входять до складу ОС NetWare, що вимагають гарантованого з'єднання і використовують для цього систему підтверджень правильності передачі пакетів (наприклад, Remote Console RCONSOLE, Remote

Printer RPRINTER і NetWare for SAA (System Application Architecture)), працюють з протоколом SPX. Пристрої, що знаходяться на обох кінцях з'єднання SPX періодично посилають один одному діагностичні запити, що зберігають активне з'єднання навіть в тому випадку, якщо передачі корисних даних не відбувається.

Як і пакети оновлень RIP і SAP, пакети IPX Watchdog і SPX можуть викликати постійну активність комутованих каналів WAN, роблячи їх використання невиправдано дорогим.

Інтервали у передачі цих пакетів можуть бути збільшені - це кілька знизить рівень використання каналу. В цьому плані Cisco IOS має можливість емулювати обидва цих протоколи, знижуючи таким чином загальну кількість трафіку, що передається через канали WAN і запобігаючи потрапляння цих пакетів в сполуки, які використовують DDR.

Маршрутизатор і сервери доступу, що працюють під управлінням Cisco IOS, можуть безпосередньо відповідати серверам NetWare на розсилаються ними запити IPX Watchdog. Ця функціональна особливість, також відома як IPX spoofing або NCP spoofing, дозволяє виробляти локальне дозвіл запитів. Що стосується функції SPX spoofing, то пристрої, що працюють під Cisco IOS, можуть відповідати на запити keepalive, розсилаючи відповідні пакети, як клієнтам, так і серверів, забезпечуючи стійке з'єднання між ними.

Використання цих функціональних можливостей забезпечує відсутність передачі непотрібного трафіку по дорогим каналам WAN, що дозволяє істотно знизити вартість їх експлуатації в умовах великих розподілених мереж NetWare.

4.6. IL протокол для передачі повідомлень 9P через IP

Протокол IL (Internet Link) служить для передачі повідомлень 9P (протоколу файлової системи з ОС Plan9) через IP.

Це протокол з встановленням логічних з'єднань, який забезпечує надійну

передачу упорядкованих повідомлень між машинами. Оскільки процес може мати тільки єдиний запит 9P, який очікує виконання, немає необхідності для управління потоком в IL [4.2, 4.19].

Подібно TCP, IL має адаптивні затримки: він встановлює масштаб час підтвердження прийому і час повторної передачі для відповідності мережевий швидкості. Це дозволяє протоколу добре працювати як в мережі Internet з великими затримками, так і в локальних мережах Ethernet з великими пропускними здатностями. Також IL робить ніяких сліпих повторних передач, щоб уникнути збільшення перевантаження зайнятих мереж.

В Plan 9 реалізація IL менше і швидше ніж TCP. IL є основним транспортним протоколом в Plan 9 при роботі через Internet.

Станом на 4 редакцію ОС Plan 9 (2003), від IL відмовилися на користь TCP, через погану роботи на великих відстанях.

4.7. DCCP протокол механізму для відстеження перевантажень у мережі

DCCP (Datagram Congestion Control Protocol) — протокол транспортного рівня моделі OSI, розроблений IETF. Прийнятий як стандарт в березні 2006 року. Він надає механізми для відстеження перевантажень у мережі, уникаючи можливості використання механізмів прикладного рівня. Цей протокол не гарантує доставку інформації в потрібному порядку [4.1, 4.19].

DCCP дуже ефективний для застосунків, в яких дані, що прийшли не вчасно, стають непотрібними. Наприклад: потокове медіа-мовлення, онлайн ігри і інтернет-телефонія. Головна особливість цих застосунків полягає в тому, що старі повідомлення дуже швидко стають непотрібними, тому краще отримати нове повідомлення, ніж намагатися переслати старе. Але на даний момент більшість таких застосунків самостійно реалізують відстеження перевантажень, а як протоколи передачі використовуються TCP або UDP.

Протокол DCCP доступний в ядрі Linux з версії 2.6.14 і поліпшується з кожним випуском.

4.8. ECN - розширення протоколу IP

ECN (Explicit Congestion Notification) — (явне повідомлення про перевантаження) — розширення протоколу IP, описане в RFC 3168. ECN дозволяє обом сторонам в мережі дізнаватися про виникнення затору на маршруті до заданого хосту або мережі без відкидання пакетів [4.1, 4.19].

Це додаткова функція, яка використовується тільки в тому випадку, коли обидві кінцеві точки обміну інформацією повідомляють, що вони хочуть її використовувати.

ECN використовує два біти в DiffServ області в заголовку IP, для IPv4 в байті TOS, а в IPv6 в октеті класу передачі пакета. Ці два біти можуть використовуватися для установки в одне з наступних значень:

- потік підтримує ECN : ECN-Capable Transport (ECT);
- потік не підтримує ECN : Not-ECN-Capable Transport (Not-ECT);
- підтверджене перевантаження : Congestion Experienced (CE).

Деяке застаріле або тестове мережеве обладнання відкидає пакети з встановленими бітами ECN, а не ігнорує їх.

На додаток до двох ECN-бітам в заголовку IP, TCP використовує два прапора заголовка TCP для сигналізації відправнику про заторі і скорочення обсягу інформації, яку він посилає.

Використання ECN в з'єднаннях TCP не є обов'язковим.

Розширення ECN також визначено для інших протоколів транспортного рівня, які виконують контроль заторів в мережі, зокрема DCCP і SCTP. Загальний принцип використання схожий на TCP, хоча деталі кодування відрізняються.

В принципі можна використовувати розширення ECN і з протоколами,

що лежать на рівнях над UDP. Однак, UDP вимагає, щоб контроль перевантажень здійснювався на рівні додатків, а поточної можливості додатків для мереж не дають їм доступу до ECN бітам.

На додаток до двох ECN-бітам в заголовку IP, TCP використовує два прапора заголовка TCP для сигналізації відправнику про заторі і скорочення обсягу інформації, яку він посилає [4.1, 4.19].

Використання ECN ефективно тільки в поєднанні з політикою активного управління чергою (AQM), і користь від ECN залежить від правильності використання AQM.

Дослідним шляхом було встановлено, що ECN погано впливає на продуктивність сильно перевантаженої мережі, якщо використовуються AQM алгоритми, ніколи не відкидають пакети. Сучасні реалізації AQM дозволяють уникнути подібної проблеми, замінюючи маркування пакетів на їх відкидання, в разі критичних перевантажень.

Багато сучасні реалізації протоколу TCP / IP мають підтримку ECN, проте вони зазвичай поставляються з вимкненим ECN.

Windows 7 і Windows 8 підтримують розширення ECN, але воно відключена за замовчуванням. Підтримка ECN може бути включена за допомогою наступної команди:

netsh interface tcp set global ecncapability = enabled.

Mac OS X 10.5 за замовчуванням підтримує ECN. Управління проводиться за допомогою інтерфейсу sysctl:

net.inet.tcp.ecn_negotiate_in net.inet.tcp.ecn_initiate_out

Ядро Linux підтримує розширення ECN вже протягом деякого часу, проте воно за замовчуванням вимкнено. У більшості версій ядра, воно може бути активована через інтерфейс Sysctl:

sysctl net.ipv4.tcp_ecn = 1.

Маркування ECN полів маршрутизаторами залежить в тій чи іншій формі від управління активними чергами. Маршрутизатор повинні бути відповідним

чином налаштовані для маркування полів ECN.

Маршрутизатор Cisco IOS виконують маркування ECN, якщо налаштований WRED, починаючи з версії 12.2 (8).

4.9. RSVP - протокол резервування мережевих ресурсів

RSVP (Resource ReSerVation Protocol) — протокол резервування мережевих ресурсів [4.1, 4.20].

З метою повідомлення маршрутизаторам мережі потреб кінцевих вузлів як обслуговування потоків використовується додатковий протокол — RSVP.

Працює він таким чином: вузол-джерело до передачі даних, що вимагають певної нестандартної якості обслуговування (наприклад, постійної смуги пропускання для передачі відеоінформації), посилає по мережі спеціальне повідомлення у форматі протоколу RSVP. Це повідомлення про шлях (path message) містить дані про тип переданої інформації і необхідні пропускні спроможності. Воно передається маршрутизаторам по всьому шляху від вузла-відправника до адреси призначення, при цьому визначається послідовність маршрутизаторів, в яких необхідно зарезервувати певну смугу пропускання [4.1, 4.21].

Маршрутизатор, одержавши таке повідомлення, перевіряє свої ресурси з метою визначення можливості виділення необхідної пропускної спроможності. При її відсутності маршрутизатор запит відкидає. Якщо необхідна пропускна здатність досяжна, то маршрутизатор налаштовує алгоритм обробки пакетів таким чином, щоб вказаному потоку завжди надавалась необхідна пропускна здатність, а потім передає повідомлення наступному маршрутизатору вздовж шляху. В результаті, по всьому шляху від вузла-відправника до адреси призначення резервується необхідна пропускна здатність з метою забезпечення запитуваної якості обслуговування.

Протокол RSVP, крім використання для сигналізації вимог до якості

обслуговування (архітектура QoS IntServ), використовується також для сигналізації MPLS TE LSP (MultiProtocol Label Switching Traffic Engineering Label-Switched Path). Для сигналізації MPLS TE LSP використовується модифікована версія протоколу — RSVP-TE (RFC 3209, RFC 5420).

4.10. Стек протоколів IPX / SPX

Стек протоколів IPX / SPX (Internetwork Packet Exchange / Sequenced Packet Exchange - міжмережевий обмін пакетами / послідовний обмін пакетами) є власністю компанії Novell.

Він був розроблений на початку 80-х років для мережевої операційної системи NetWare, яка ще донедавна займала одну з лідируючих позицій серед серверних операційних систем [4.1, 4.22].

Протоколи IPX (Internetwork Packet Exchange) і SPX, які дали ім'я стека, є прямою адаптацією протоколів XNS фірми Xerox, поширених в набагато менше ступеня, ніж IPX / SPX.

За кількістю установок протоколи IPX / SPX лідирують, і це обумовлено тим, що сама ОС NetWare займає лідируюче положення з часткою установок в світовому масштабі приблизно в 65%.

Стек протоколів IPX / SPX (Internetwork Packet Exchange / Sequenced Packet Exchange - міжмережевий обмін пакетами / послідовний обмін пакетами) є власністю компанії Novell.

Він був розроблений на початку 80-х років для мережевої операційної системи NetWare, яка ще донедавна займала одну з лідируючих позицій серед серверних операційних систем.

Протоколи IPX (Internetwork Packet Exchange) і SPX, які дали ім'я стека, є прямою адаптацією протоколів XNS фірми Xerox, поширених в набагато менше ступеня, ніж IPX / SPX.

За кількістю установок протоколи IPX / SPX лідирують, і це обумовлено

тим, що сама ОС NetWare займає лідируюче положення з часткою установок в світовому масштабі приблизно в 65%.

Сімейство протоколів фірми Novell і їх відповідність моделі ISO / OSI.

IPX і SPX отримані на підставі IDP і SPP протоколів відповідно компанії Xerox Network Systems [4.23]. IPX - протокол мережевого рівня (рівень 3 мережевий моделі OSI), в той час як SPX - протокол транспортного рівня (рівень 4 мережевий моделі OSI). SPX рівень знаходиться на вершині IPX рівня і встановлює з'єднання між двома мережевими вузлами і здійснює передачу повідомлень між ними. SPX використовується в основному клієнт-серверними додатками.

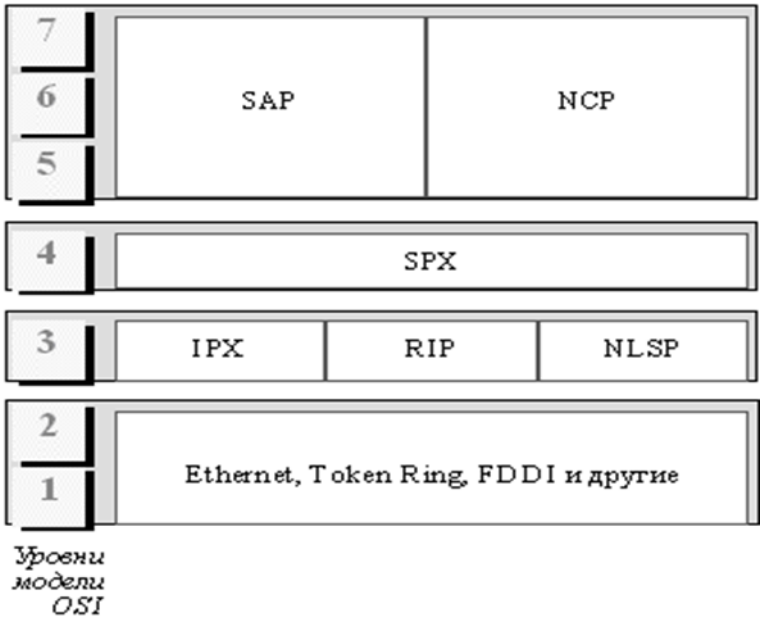


Рис. 4.6. Вид клієнт-серверних додатків

IPX є протоколом, який займається питаннями адресації та маршрутизації пакетів в мережах Novell. Маршрутні рішення IPX засновані на адресних полях в заголовку його пакета, а також на інформації, що надходить від протоколів обміну маршрутною інформацією. Наприклад, IPX використовує інформацію, що поставляється або протоколом RIP, або протоколом NLSP (NetWare Link State Protocol) для передачі пакетів комп'ютера призначення або наступного маршрутизатора.

Протокол IPX підтримує тільки дейтаграмний спосіб обміну

повідомленнями, за рахунок чого економно споживає обчислювальні ресурси.

Отже, протокол IPX забезпечує виконання трьох функцій: завдання адреси, встановлення маршруту і розсилку дейтаграм.

IPX і SPX забезпечують з'єднання, подібне TCP / IP (Transmission Control Protocol / Internet Protocol), з IPX протоколом, який має схожі риси з IPsec (IP Security), і з SPX, який схожий на TCP. IPX / SPX були головним чином розроблені для локальних мереж (LANs - LAN (Local Area Network)), і ефективно справляються з цим завданням (зазвичай їх продуктивність перевищує TCP / IP в локальній мережі).

Звичайно, TCP / IP став на ділі стандартним протоколом. Це не дивно, враховуючи його перевершує продуктивність по глобальних мережах і Інтернету (який використовує виключно TCP / IP), а також внаслідок того, що TCP / IP розвиненіший протокол, створений спеціально для цих цілей [4.1, 4.21].

Незважаючи на зв'язок протоколів з NetWare, вони не потрібні для забезпечення зв'язку з серверами NetWare (станом на NetWare 5.x) і не використовуються виключно в мережах NetWare. NetWare з'єднання вимагає NCP реалізацію, яка може використовувати IPX / SPX, TCP / IP, або обидва відразу для транспортування [4.3, 4.22].

Саме компанія Novell відповідальна за використання IPX в якості популярного комп'ютерного мережевого протоколу через їх переважання на ринку програмного забезпечення мережевої операційної системи (з NetWare) з кінця 1980-х і до середини 1990-х. Хотя останнім часом її популярність набагато знизилася, і за темпами зростання вона помітно відстає від Microsoft Windows.

Багато особливостей стека IPX / SPX обумовлені орієнтацією ранніх версій ОС NetWare (до версії 4.0) на роботу в локальних мережах невеликих розмірів, що складаються з персональних комп'ютерів зі скромними ресурсами. Зрозуміло, що для таких комп'ютерів компанії Novell потрібні були протоколи, на реалізацію яких була б потрібна мінімальна кількість оперативної

пам'яті (обмеженою в IBM-сумісних комп'ютерах під управлінням DOS (Disk Operating System) обсягом 640 Кбайт) і які швидко працювали б на процесорах невеликої обчислювальної потужності.

В результаті протоколи стека IPX / SPX донедавна добре працювали в локальних мережах і не дуже - в великих корпоративних мережах, так як вони занадто перевантажували повільні глобальні зв'язки широкомовними пакетами, які інтенсивно використовуються декількома протоколами цього стека (наприклад, для встановлення зв'язку між клієнтами і серверами). Ця обставина, а також той факт, що стек IPX / SPX є власністю фірми Novell, і на його реалізацію потрібно отримувати ліцензію (тобто відкриті специфікації не підтримувалися), довгий час обмежували його поле діяльності тільки мережами NetWare. Однак з моменту випуску версії NetWare 4.0 фахівці Novell внесли і продовжують вносити в протоколи серйозні зміни, спрямовані на їх адаптацію для роботи в корпоративних мережах. Зараз стек IPX / SPX реалізований не тільки в NetWare, але і в декількох інших популярних мережевих ОС, наприклад SCO UNIX, Sun Solaris, Microsoft Windows [4.1, 4.23].

Спочатку NetWare клієнт був написаний компанією Novell для DOS. Перші версії вимагали жорстко пов'язаний стек протоколів, де виконуваний файл створювався б адміністратором для кожної мережевої карти окремо.

Цей виконуваний файл був би завантажений протягом встановленого часу і залишався б в пам'яті до тих пір, поки система не завершить програму. Пізніше реалізації дозволили мережевого стеку завантажуватися і розвантажуватися динамічно за допомогою модулів, що виконуються. Це значно спростило процедуру обслуговування клієнтських терміналів в мережі.

Протокол IPX / SPX на ділі був зразком для багатокористувацьких мережевих ігор ери DOS. Життєвий цикл багатьох ігор був збільшений за рахунок тунельних програм таких, як Kali Linux і Kahn, які дозволили грати в

них через Інтернет [4.9, 4.22].

Внаслідок поширеності IPX / SPX в локальних мережах в 1990-х, Microsoft Corporation додав підтримку для протоколів в мережеві стеки Microsoft Windows, починаючи з Windows for Workgroups і Microsoft Windows NT. Microsoft навіть назвав свої реалізації "NWLink", маючи на увазі, що включення транспортів рівня 3/4 забезпечить зв'язок з серверами NetWare.

Насправді, протоколи підтримувалися як вихідний транспорт для Microsoft Windows SMB (Server Message Block) / NetBIOS, і зв'язок з NetWare потребувала додаткове встановлення NCP клієнта (Microsoft Corporation надала Microsoft Windows 95 базового клієнту NetWare, але вона не встановлювалася автоматично, і спочатку підтримувався тільки режим системної бази даних NetWare).

NWLink був все ще забезпечений Microsoft Windows (ранніми версіями по Windows Server 2003 включно), але не підтримувався Microsoft Windows Vista. Його використання строго перешкоджали, так як він не міг бути використаний для організації мережі Microsoft Windows за винятком транспорту для NetBIOS, який застарів.

Головним чином, 32-бітове клієнтське програмне забезпечення Microsoft Windows компанії Novell уникнуло NWLink завдяки альтернативі, розробленої Novell, хоча деякі версії допускають використання реалізації IPX / SPX Microsoft Corporation (з попередженням про потенційну несумісності).

Протягом декількох років компанія Novell поставляла вихідний клієнт NetWare для OS / 2. За структурою він нагадував клієнт для DOS.

Novell також випускала IPX клієнт для Classic Mac OS X, званий MacIPX. Він використовувався не тільки Mac NetWare клієнтом, а також іграми такими, як Doom і Warcraft III для багатокористувацької гри [4.23].

Реалізації були написані для різновидів Unix / Linux, обидві компанією Novell і іншими розробниками. Зокрема, Novell's UnixWare спочатку підтримувала IPX / SPX. Звичайно, поки UnixWare могла виступати в якості

клієнта NetWareслужби, і додатки могли додатково підтримувати IPX / SPX як транспорт, UnixWare не надавала можливості спільно використовувати файли або принтери в NetWare мережі без додаткового програмного пакета. Open Enterprise Server - Linux не підтримує IPX / SPX.

Відкрита вихідна операційна система FreeBSD включає IPX / SPX стек, щоб підтримати файловий системний клієнт NetWare і службу NetWare, що використовує Mars NWE (забезпечуючи тим самим деяку функціональність). OpenBSD відмовилася від підтримки з версії 4.2 і 4.1 знадобилося трохи доопрацювати, щоб скомпілювати з IPX [4.1, 4.23].

Використання IPX різко знизилося за останні роки в зв'язку з тим, що розвиток Інтернету зробило TCP / IP всюдисущим.

Перша спроба Novell підтримати TCP / IP як клієнтський протокол, названий NetWare / IP, просто туннелізірована IPX в IP пакети, тим самим дозволяючи клієнтам і службам NetWare обмінюватися даними по чистій TCP / IP мережі. Однак через складну реалізацію і значні втрати в продуктивності в зв'язку з туннелюванням нагору, NetWare / IP був в основному проігнорований за винятком механізму, призначеного для того, щоб направити IPX через маршрутизатори TCP / IP-only і канали WAN.

NetWare 5.x розробила вихідну підтримку для NCP по протоколу TCP / IP, який зараз є найкращою конфігурацією. Наступником NetWare і Open Enterprise Server є OES-NetWare, яка забезпечує підтримку застарілих IPX / SPX, і OES-Linux, яка підтримує тільки TCP / IP.

Microsoft Corporation і Novell забезпечили підтримку (через Proxy Server / ISA Server і BorderManager, відповідно) IPX / SPX як протоколу внутрішньої мережі, щоб зв'язатися через міжмережевий екран. Це дозволяє машині, використовуючи клієнтське програмне забезпечення, отримати доступ до Інтернету без локального встановлення TCP / IP; клієнтське програмне забезпечення емулює вихідний TCP / IP стек і забезпечує WinSock підтримку для локальних додатків (наприклад, веб-браузери), але насправді зв'язується з

фаєрволом по IPX / SPX.

У додаванні до спрощення міграції для застарілих IPX LAN , це забезпечує заходи безпеки, оскільки використання IPX протоколу у внутрішній мережі передбачає природний бар'єр на випадок, якщо зловмисники спробують піддати небезпеці міжмережевий екран [4.2].

На жаль, стек протоколів IPX / SPX від початку орієнтований на обслуговування мереж невеликого розміру, тому у великих мережах його використання малоефективно: зайве використання широкомовного мовлення на низькошвидкісних лініях зв'язку неприпустимо.

Єдина область, де IPX залишається корисним, це обхід з'єднань VPN, політика безпеки яких забороняє зв'язок з іншими пристроями LAN (такими як принтери та мережеве сховище даних) через TCP / IP.

ЛІТЕРАТУРА

4.1 Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципи, технологии, протоколы: Учебник для вузов. 3-е изд. – СПб.: Питер, 2006.– 958 с.

4.2 Douglas E. Comer. Internetworking with TCP/IP, Vol. 1: Principles, Protocols and Architecture.

4.3 Дуглас Камер. Сети TCP/IP, том 1. Принципы, протоколы и структура. М. «Вильямс» 2003, ISBN 0-13-018380-6

4.4 Комп'ютерні мережі. Частина 1. Навчальний посібник [Електронний ресурс]: навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» та 126 «Інформаційні системи та технології», спеціалізації «Інженерія програмного забезпечення інформаційно управляючих систем» та «Інформаційне забезпечення робототехнічних систем»/ КПІ ім. Ігоря Сікорського; уклад. Б. Ю. Жураковський, І.О. Зенів;. – Електронні текстові дані (1 файл: 4,3 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 336 с. – Назва з екрана. URL: https://classroom.google.com/u/0/c/MTQ1MDk5NzA3OTQ1?hl=ru

4.5 TCP Flag Options — Section 4.

4.6 TCP, Transmission Control Protocol.

4.7 TCP Window Size, Checksum & Urgent Pointer — Section 5.

4.8 Transmission Control Protocol (TCP) Parameters.

4.9 Галкин В. А., Григорьев Ю. А. Телекоммуникации и Сети. — М.: МГТУ им. Н. Э. Баумана, 2003. С. 608. ISBN 5-7038-1961-X.

4.10 Friend, George E.; John L. Fike, H. Charles Baker, John C. Bellamy (1988). Understanding Data Communications (вид. 2nd). Indianapolis: Howard W. Sams & Company. ISBN 0-672-27270-9.

4.11 Stallings, William (2004). Data and Computer Communications (вид. 7th). Upper Saddle River: Pearson/Prentice Hall. ISBN 978-013100-6812.

4.12 S. Tanenbaum, Andrew (2005). Computer Networks (вид. 4th).

482,F.I.E., Patparganj, Delhi 110 092: Dorling Kindersley(India)Pvt. Ltd.,licenses of Pearson Education in South Asia. ISBN 81-7758-165-1.

4.13 Douglas E. Comer. Internetworking with TCP/IP, Vol. 1: Principles, Protocols and Architecture. Дуглас Камер. Сети TCP/IP, том 1. Принципы, протоколы и структура. М. «Вильямс» 2003, ISBN 0-13-018380-6

4.14 IPX/SPX // Wikipedia. URL: https://en.wikipedia.org/wiki/IPX/SPX.

4.15 Guide to Cisco Router Configuration // OpenNET — веб-сайт, русскоязычный интернет-проект, посвящённый открытым и свободным компьютерным технологиям. URL: http://www.opennet.ru/soft/cisco-configuration.html

4.16 Measuring Interactions Between Transport Protocols and Middleboxes. Alberto Medina, Mark Allman, and Sally Floyd. Internet Measurement Conference 2004, August 2004.

4.17 Aleksandar Kuzmanovic. The power of explicit congestion notification. In Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications. 2005

4.18 "New Networking Features in Windows Server 2008 and Windows Vista“.

4.19 "ECN (Explicit Congestion Notification) in TCP/IP".

4.20 IPX Packet Structure.

4.21 Novell's NetWare 4.1.

4.22 Interoperating with OS/2 Named Pipes.

4.23 WinSock 2 Developer Components for NetWare.

5. ПРОТОКОЛИ СЕАНСОВОГО РІВНЯ

Сеансовий рівень (Session layer) - 5-й рівень мережної моделі OSI, відповідає за підтримання сеансу зв'язку, дозволяючи додаткам взаємодіяти між собою тривалий час.

Рівень управляє створенням / завершенням сеансу, обміном інформацією, синхронізацією завдань, визначенням права на передачу даних і підтримкою сеансу в періоди неактивності додатків. Синхронізація передачі забезпечується приміщенням в потік даних контрольних точок, починаючи з яких поновлюється процес при порушенні взаємодії.

Сеанси передачі складаються із запитів і відповідей, які здійснюються між додатками. Служби сеансового рівня зазвичай використовуються в середовищах додатків, в яких потрібне використання віддаленого виклику процедур.

Прикладом протоколу сеансового рівня є X.225, або ISO 8327. В разі втрати або тривалої з'єднання цей протокол може спробувати його відновити. Якщо з'єднання не використовується тривалий час, то протокол сеансового рівня може його закрити і відкрити заново. Він дозволяє проводити передачу в дуплексному або в напівдуплексному режимах і забезпечує наявність контрольних точок в потоці обміну повідомленнями [5.1].

Іншими прикладами реалізації сеансового рівня є Zone Information Protocol (ZIP) - протокол AppleTalk, що забезпечує узгодженість процесу зв'язування по імені, а також протокол управління сеансом (Session Control Protocol (SCP)) - протокол рівня сеансу IV стадії проекту розробки стека протоколів DECnet.

В рамках семантичних конструкцій сеансового рівня мережевої архітектури OSI цей рівень відповідає на запити на послуги з представницького вирівняти здійснює службові запити до транспортному рівню.

Права доступу. Відновлення сеансу (встановлення контрольних точок та

відновлення).

Сеансовий рівень моделі OSI відповідає за встановлення контрольних точок і відновлення. Він дозволяє відповідним чином поєднувати і синхронізувати інформацію декількох потоків, можливо від різних джерел.

Прикладом застосування є організація відеоконференцій в мережі, коли звуковий і відео потоки повинні бути синхронізовані для уникнення проблем з синхронізацією руху губ з промовою. Управління правами на участь в розмові гарантує, що той, хто показується на екрані, дійсно є співрозмовником, який в даний момент говорить.

Ще одним застосуванням є передачі в прямому ефірі, в яких необхідно без різких переходів накладати звуковий і відео потоки і переходити від одного потоку до іншого, щоб уникнути перерв в ефірі або зайвих накладень [5.1].

Протоколи:

ADSP, Протокол потоків даних AppleTalk (AppleTalk Data Stream Protocol)

- ASP, Сеансовий протокол AppleTalk (AppleTalk Session Protocol)
- H.245, Call Control Protocol for Multimedia Communication
- ISO-SP, OSI session-layer protocol (X.225, ISO 8327)
- iSNS, Internet Storage Name Service
- L2F, Layer 2 Forwarding Protocol
- L2TP, Layer 2 Tunneling Protocol
- NetBIOS, Network Basic Input Output System
- PAP, Password Authentication Protocol
- PPTP, Point-to-Point Tunneling Protocol
- RPC, Remote Procedure Call Protocol
- RTCP, Real-time Transport Control Protocol
- SMPP, Short Message Peer-to-Peer
- SCP, Session Control Protocol
- SOCKS, мережевий протокол SOCKS, см. Сокет

• ZIP, Zone Information Protocol

• SDP, Sockets Direct Protocol

Порівняння з моделлю DOD. У еталонної моделі DOD (TCP / IP) відсутній розгляд порушених у моделі OSI питань застосування семантики транспортного протоколу, і тому сеансовий рівень не розглядається.

Управління сеансами OSI в з'єднанні з типовими транспортними протоколами (TCP, SCTP) міститься в протоколах транспортного рівня або ж в іншому випадку зачіпає область протоколів прикладного рівня.

Шари моделі DOD є описом рамок функціонування (додаток, з'єднання хост-хост, мережа, зв'язок), але не докладними приписами щодо способу функціонування або семантики даних.

5.1. AppleTalk стек протоколів для комп'ютерної мережі

AppleTalk (AppleTalk Data Stream Protocol) - стек протоколів, розроблених Apple Computer для комп'ютерної мережі. Він був спочатку включений в Macintosh (1984), але потім компанія відмовилася від нього на користь TCP/IP.

Відповідна сеансовому рівню моделі OSI версія AppleTalk складається з п'яти протоколів, що підтримують повністю дуплексну передачу даних, перетворення логічних назв в адреси, доступ до принтера, переупорядочение пакетів і т. д [5.1].

Перший протокол сеансового рівня називається протоколом потоків даних (AppleTalk Data Stream Protocol - ADSP). Протокол ADSP надає повністю дуплексні послуги, орієнтовані на встановлення з'єднання і характеризуються високим ступенем надійності. Така надійність досягається шляхом встановлення логічного з'єднання (сеансу) між двома взаємодіючими процесами на клієнтських машинах.

Протокол ADSP дозволяє управляти цим з'єднанням, забезпечуючи контроль потоку даних, переупорядочение пакетів і розсилку підтверджень про

прийом пакетів. Для встановлення логічного з'єднання між процесами використовуються номери сокетів. Після встановлення з'єднання дві системи можуть почати обмін даними.

Наступним протоколом сеансового рівня AppleTalk є власне сеансовий протокол (AppleTalk Session Protocol - ASP).

Протокол ASP забезпечує надійну доставку даних, використовуючи для цього орієнтоване на коректність прийнятих послідовностей управління сеансом (sequence-oriented session management), і надає доступ до транспортних послуг протоколу транспортного рівня AppleTalk Transport Protocol (ATP).

Протокол маршрутизації з оновленням середовища AppleTalk (AppleTalk Update-Based Routing Protocol - AURP) використовується в великих мережах AppleTalk і застосовується в основному для маршрутизації і підтримки обміну інформацією між маршрутизирующий пристроями, зокрема, між маршрутизаторами Exterior Gateway.

Крім того, до складу сеансового рівня AppleTalk входить протокол доступу до принтера (Printer Access Protocol - PAP). Незважаючи на те що спочатку протокол PAP був розроблений для управління доступом до мережевих принтерів, він може використовуватися для забезпечення обміну даними між різними пристроями. Між пристроями встановлюється двонаправлене з'єднання і одночасно здійснюється управління потоком даних і контроль послідовності пакетів [5.1, 5.2].

І, нарешті, останній протокол сеансового рівня AppleTalk, - протокол зонної інформації (Zone Information Protocol - ZIP). Протокол ZIP надає механізм логічного групування окремих мережевих пристроїв за допомогою «дружніх» імен. Такі логічні групи називаються зонами (zones). У розширеній мережі комп'ютери можуть охоплювати кілька мереж, але залишатися при цьому логічно згрупованими в одну зону. Однак в невеликих, нерозширена мережах може бути визначена єдина зона.

П'ять протоколів сеансового рівня AppleTalk надають клієнтам

можливість встановлювати логічне з'єднання і обмінюватися даними між комп'ютерами незалежно від відстані між ними.

Для перетворення назви зон в номери мереж і вузлів ZIP використовує протокол зв'язування імен (Name Binding Protocol - NBP), що належить транспортному рівню.

Для розсилки даних про зміну конфігурації зони використовується протокол ATP.

5.2. PAP протокол простої перевірки автентичності

PAP (Password Authentication Protocol) - протокол простої перевірки автентичності, який передбачає відправку імені користувача і пароля на сервер віддаленого доступу відкритим текстом (без шифрування) [5.1].

Протокол аутентифікації PAP використовується в протоколі PPP (Point-to-Point Protocol), для надання користувачам доступу до серверних ресурсів. Майже всі мережеві операційні системи підтримують протокол PAP.

PAP передає незашифровані ASCII коди по мережі і тому вкрай небезпечний, оскільки пересилаються паролі можна легко читати в пакетах, якими обмінюються сторони в ході перевірки автентичності. Зазвичай PAP використовується тільки при підключенні до старих серверів віддаленого доступу на базі UNIX, які не підтримують жодні інші протоколи перевірки автентичності.

5.3. H.245 - протокол узгодження параметрів з'єднання

H.245 - протокол узгодження параметрів з'єднання. Використовується, наприклад, в H.323 або H.324 сеансі зв'язку.

Канал контролю, що працює за стандартом H.245, служить для виявлення розуміється обома сторонами набору функцій, для управління роботою

логічних каналів і деяких загальних повідомлень [5.1]. У кожному сеансі зв'язку існує один і тільки один канал H.245.

Рис. 5.1. Повідомлення протоколів H.245 і H.225 в рамках найпростішого (базового виклику) в H.323

Стандарт H.245 описує процедуру встановлення загального набору мультимедійних можливостей. Для цього використовуються повідомлення TerminalCapabilitySet (TCS) і TerminalCapabilitySetAck (TCSA), що направляються кожною стороною, яка обслуговує виклик. Таким чином, в рамках процедури встановлення виклику між обладнанням (програмами) сторони обмінюються інформацією про кодеках і домовляються про те, на якому з них буде проходити розмова (обмін мультимедійними даними). Якщо одна сторона не має якогось кодека, то іншій стороні не можна використовувати

цей кодек [5.1, 5.3].

Після процедури TCS відкривається логічний канал, що закривається на етапі завершення виклику.

В рамках H.323 / H.225.0 сигналізації в IP-телефонії для сесії протоколу H.245 передбачено відкриття окремих гнізд (порт). Для зменшення проблем з кількістю портів, зокрема, для подолання NAT, існує опциональная функція тунелювання (Tunnelling), коли H.245 сесія встановлюється в рамках H.225 сесії (на тому ж сокеті).

Існує також кілька варіантів процедури FastStart, які дозволяють скоротити кількість повідомлень до встановлення з'єднання, при цьому повідомлення протоколу H.245 передаються одночасно з повідомленнями H.225 і відкриття логічного каналу пропонується раніше. Однак процедури FastStart підтримуються не всіма пристроями H.323.

5.4. PPTP тунельний протокол типу точка-точка

PPTP (Point-to-Point Tunneling Protocol) - тунельний протокол типу точка-точка, що дозволяє комп'ютеру встановлювати захищене з'єднання з сервером за рахунок створення спеціального тунелю в стандартній, незахищеною мережі.

PPTP поміщає (інкапсулює) кадри PPP в IP-пакети для передачі по глобальної IP-мережі, наприклад Інтернет [5.1, 5.4].

PPTP може також використовуватися для організації тунелю між двома локальними мережами. PPTP використовує додаткове TCP-з'єднання для обслуговування тунелю [5.17].

Специфікація протоколу була опублікована як «інформаційна» RFC 2637 в 1999 році. Вона не була ратифікована IETF. Протокол вважається менш безпечним, ніж IPSec.

PPTP працює, встановлюючи звичайну PPP сесію з протилежною стороною за допомогою протоколу Generic Routing Encapsulation.

Друге з'єднання на TCP-порту тисячі сімсот двадцять три використовується для ініціації і управління GRE-з'єднанням. PPTP складно перенаправляти за мережевий екран, так як він вимагає одночасного встановлення двох мережевих сесій [5.17].

PPTP-трафік може бути зашифрований за допомогою MPPE. Для аутентифікації клієнтів можуть використовуватися різні механізми, найбільш безпечні з них - MS-CHAPv2 і EAP-TLS.

Cisco першої реалізувала PPTP і пізніше ліцензувала цю технологію корпорації Microsoft.

PPTP вдалося домогтися популярності завдяки тому, що це перший протокол тунелювання, який був підтриманий корпорацією Microsoft. Всі версії Microsoft Windows, починаючи з Windows 95 OSR2, включають в свій склад PPTP-клієнт, однак існує обмеження на два одночасних вихідних сполуки. А сервіс віддаленого доступу для Microsoft Windows включає в себе PPTP сервер.

Деякий час в Linux-дистрибутивах була відсутня повна підтримка PPTP через побоювання патентних претензій з приводу протоколу MPPE. Вперше повна підтримка MPPE з'явилася в Linux 2.6.13 (2005 рік). Офіційно підтримка PPTP була розпочата з версії ядра Linux 2.6.14. Проте, сам факт застосування MPPE в PPTP фактично не забезпечує безпеку протоколу PPTP [5.1, 5.4-5.9].

PPTP був об'єктом безлічі аналізів безпеки, в ньому були виявлені різні серйозні уразливості. Відомі відносяться до використовуваних протоколах аутентифікації PPP, влаштуванню протоколу MPPE і інтеграції між аутентифікації MPPE і PPP для установки сесійного ключа. Короткий огляд даних вразливостей:

• MSCHAP-v1 абсолютно ненадійний. Існують утиліти для легкого вилучення хешів паролів з перехопленого обміну MSCHAP-v1.

• MSCHAP-v2 вразливий для словникової атаки на перехоплені challenge response пакети. Існують програми, які виконують даний процес.

• У 2012 році було показано, що складність підбору ключа MSCHAP-v2

еквівалентна підбору ключа до шифрування DES, і був представлений онлайн-сервіс, який здатний відновити ключ за 23 години

• При використанні MSCHAP-v1, MPPE використовує однаковий RC4 сесійний ключ для шифрування інформаційного потоку в обох напрямках. Тому стандартним методом є виконання XOR'a потоків з різних напрямків разом, завдяки чому криптоаналитик може дізнатися ключ.

• MPPE використовує RC4 потік для шифрування. Не існує методу для аутентифікації цифробуквене потоку і тому даний потік вразливий для атаки, що робить підміну бітів. Зловмисник легко може змінити потік при передачі і замінити деякі біти, щоб змінити вихідний потік без небезпеки свого виявлення. Дана підміна бітів може бути виявлена за допомогою протоколів, які вважають контрольні суми.

5.5. RTCP - протокол управління передачею в реальному часі

RTCP (Real-Time Transport Control Protocol - протокол управління передачею в реальному часі) - протокол, який використовується спільно з RTP. Протокол описаний в RFC 3550. RTCP базується на періодичній передачі керуючих пакетів всім учасникам сесії, використовуючи той же механізм розсилки, що і для пакетів даних [5.1, 5.4].

Протокол RTCP використовується для передачі інформації про затримки і втрати медіа-пакетів, джиттер-буфері, рівні звукового сигналу. Також передаються метрика якості сигналу (Call Quality Metrics) і Echo Return Loss.

Визначено такі типи повідомлень RTCP:

• SR - Sender Report - звіт відправника по відправленим медіа-пакетів RTP

• RR - Receiver Report - звіт одержувача за отриманими медіа-пакетів RTP

• SDES - елементи опису джерела, включаючи cname

• BYE - Зазначає припинення участі в групі

• APP - Специфічні функції програми.

В рекомендації RFC 3611 певного також повідомлення XR - Extended Report, яке дозволяє відправляти більше число параметрів, в порівнянні зі стандартними звітами, а саме:

- Час отримання пакета;
- Порядкові номери втрачених пакетів;
- Порядкові номери повторюваних пакетів;
- Очікуваний час доставки;
- Затримка з моменту прийому останнього звіту RTCP Receiver Report;
- Загальна статистика медіа-пакетів.

Оцінка VoIP - напрямки (MOS і R Factor - параметр характеризує якість сигналу).

Існує варіант протоколу RTP с шифруванням Secure Real-time Transport Protocol (SRTP) для забезпечення безпечної передачі даних. І так як RTP тісно пов'язаний з RTCP (Real-Time Control Protocol), який може використовуватися, щоб управляти сесією RTP, у SRTP також є споріднений протокол, названий Secure RTCP (або SRTCP). SRTCP забезпечує ті ж самі функції, пов'язані з безпекою в RTCP, для тієї ж функціональності SRTP в RTP [5.1, 5.4-5.6].

Протокол SRTCP описаний в RFC 3711 про SRTP.

SRTCP додає 3 нових обов'язкових поля "SRTCP index", "encrypt-flag" і "authentication tag" і опциональное поле MKI в опис пакета RTCP.

Використання SRTP або SRTCP є необов'язковим при використанні RTP або RTCP, але навіть якщо SRTP / SRTCP використовуються, всі додаткові можливості (такі як шифрування і встановлення автентичності) опційні і можуть бути включені або виключені. Єдиний виняток - функція аутентифікації повідомлень, яка обов'язкова при використанні SRTCP.

Для шифрування медіа потоку (в цілях конфіденційності голосового з'єднання), SRTP разом з SRTCP стандартизують використання тільки єдиного шифру, AES, який може використовуватися в двох режимах, що перетворюють спочатку блоковий шифр AES в потоковий шифр.

5.6. NFS - протокол мережевого доступу до файлових систем

Network File System (NFS) - протокол мережевого доступу до файлових систем, спочатку розроблений Sun Microsystems в 1984.

Заснований на протоколі виклику віддалених процедур (ONC RPC).

Мережева файлова система дозволяє монтувати файлову систему з віддаленого комп'ютера так, як ніби вона локальна і знаходиться у вашій системі. Після такого монтування користувач може безпосередньо звертатися до файлів цієї віддаленої файлової системи [5.1-5.4, 5.8].

Перевага полягає в тому, що різні комп'ютери мережі можуть отримувати прямий доступ до одних і тих же файлів без необхідності створення їх копій. Існує тільки один екземпляр файлу, що знаходиться в віддаленій файловій системі, і до нього може звертатися будь-який комп'ютер.

Система NFS працює в мережі TCP/IP.

Віддалений комп'ютер, на якому знаходиться файлова система, надає її іншим машинами мережі.

Це досягається за рахунок експорту файлової системи. Щоб це здійснити необхідно на віддаленому комп'ютері додати відповідні рядки у файл конфігурації NFS (його ім'я - / etc / exports), а також запустити два процеси-демони, що забезпечують доступ до віддаленого комп'ютера (це програми rpc.mountd і rpc.nfsd).

У кожному рядку файлу / etc / exports вказується експортована система і мережеві комп'ютери, які мають право доступу до неї.Дл я файлової системи вказується точка монтування - каталог, в який вона монтується. Потім, список комп'ютерів, що мають до неї доступ. За кожним ім'ям може слідувати розділений комами список опцій монтування, взятий в круглі дужки.

Наприклад, для одного комп'ютера можна надати доступ лише для читання, для іншого - для читання і запису і т.д.

Якщо зазначені тільки опції, вони поширюються на всіх. До файлової системи, змонтованої в каталозі / pub (це ім'я зазвичай використовується для загальнодоступного каталогу), всім комп'ютерам надається доступ тільки для читання без перевірки прав доступу.

Існує безліч реалізацій NFS-серверів і клієнтів для різних операційних систем і апаратних архітектур. Найновіша версія NFS - v.4, яка підтримує різні засоби аутентифікації (зокрема, Kerberos і LIPKEY з використанням протоколу RPCSEC GSS) і списків контролю доступу (як POSIX , чи Windows -типів).

NFS надає клієнтам прозорий доступ до файлів і файлової системи сервера. На відміну від FTP, протокол NFS здійснює доступ тільки до тих частин файлу, до яких звернувся процес, і основна перевага його в тому, що він робить цей доступ прозорим. Це означає, що будь-який додаток клієнта, який може працювати з локальним файлом, з таким же успіхом може працювати і з NFS файлом, без будь-яких модифікацій самої програми [5.1-5.4, 5.9].

NFS-клієнти отримують доступ до файлів на NFS-сервері шляхом відправки RPC -запитів на сервер. Це може бути реалізовано з використанням звичайних користувальницьких процесів - а саме, NFS-клієнт може бути для користувача процесом, який здійснює конкретні RPC-виклики на сервер, який так само може бути для користувача процесом.

Важливою частиною останньої версії стандарту NFS (v4.1) стала специфікація pNFS, націлена на забезпечення розпаралелених реалізації спільного доступу до файлів, що збільшує швидкість передачі даних пропорційно розмірам і ступеня паралелізму системи [5.1-5.7].

При неправильному налаштуванні NFS також створює потенційну небезпеку несанкціонованого доступу до диску через мережу, і, як наслідок, розголошення і псування інформації. Тому при використанні мережевих файлових систем потрібно приділяти багато уваги безпеці та контролю доступу.

Існують також інші мережеві файлові системи:

1. Файлова система Samba для оффтопікових (TM) клієнтів.

2. Файлова система Andrew для IBM, надає спільне використання файлів з додатковою безпекою та покращенню продуктивністю.

3. Файлова система Coda розроблена для роботи з від'єднаними клієнтами.

Більшість властивостей файлових систем Andrew і Coda реалізовані в NFS4. Основною перевагою NFS є готовність, стандартність, зрозумілість та широка підтримка на багатьох платформах.

Remote Procedure Calls (RPC) - віддалений виклик процедур. Багато клієнт-серверних програм замість того, щоб слухати якийсь окремий мережевий порт, користуються технологією RPC. Коли сервіс ініціалізується, він під’єднується до випадкового порта і тоді реєструє цей порт з допомогою утиліти Portmapper. NFS відноситься саме до сервісів RPC, тому вимагає працюючого Portmapper перед її запуском [5.1-5.4, 5.8].

Встановлення утиліт NFS. Для роботи NFS додатково потрібні утиліти з http://nfs.sourceforge.net/ (http://www.nfsv4.org), які для Gentoo можна встановити командою: # emerge net-fs/nfs-utils.

Цей пакет встановить portmapper та серверні і клієнтські утиліти nfs та автоматично вирішить всі залежності.

Щоб встановити сервер NFS потрібно:

- включити підтримку NFS сервера в ядро,

- налаштувати файл /etc/exports та файли безпеки: /etc/hosts.allow і /etc/hosts.deny.

Для забезпечення надійної роботи сервісу NFS в мережі його потрібно встановлювати на drbd.

Компіляція ядра Лінукс з підтримкою сервера NFS[5.1-5.4, 5.10].

Для роботи NFS потрібна підтримка зі сторони ядра Лінукс. Якщо ваше ядро не має такої підтримки, потрібно його перезібрати. Додатково потрібно ввімкнути:

Networking --->

[*] Networking support

Networking options ---->

<*> Packet socket

[*] Packet socket: mmapped IO

<*> Unix domain sockets

<*> IPsec user configuration interface

<*> PF_KEY sockets

[*] TCP/IP networking

.....

File systems ----> ..... Network File Systems ---->

<*> NFS file system support

[ ] Allow direct I/O on NFS files (EXPERIMENTAL)

<*> NFS server support

[*] Provide NFSv3 server support

[*] Provide server support for the NFSv3 ACL protocol extension

[*] Provide NFSv4 server support (EXPERIMENTAL)

--- Provide NFS server over TCP support

--- Secure RPC: Kerberos V mechanism (EXPERIMENTAL)

Спочатку потрібно відредагувати конфігураційний файл */etc/exports*.

Файл /etc/exports вказує як, що і до кого експортувати через NFS. Типовий */etc/exports* для сервера, що обслуговує бездискові станції, виглядає так:

$ cat /etc/exports

 # one line like this for each slave

/diskless/10.0.0.101 10.0.0.101(sync,rw,no_root_squash,no_all_squash)

 # common to all slaves

/opt 10.0.0.0/24(sync,ro,no_root_squash,no_all_squash)

/usr 10.0.0.0/24(sync,ro,no_root_squash,no_all_squash)

/home 10.0.0.0/24(sync,rw,no_root_squash,no_all_squash)

 # if you want to have a shared log

/var/log 10.0.0.0/24(sync,rw,no_root_squash,no_all_squash)

Перше поле визначає директорію яка експортується, друге — кому і як. Останнє поле розділене на дві частини, перша — кому саме дозволено монтувати дану директорію, а друга — що він може з цією файловою системою робити:

- *ro* : дозволяється тільки читати;
- *rw* : дозволяється читати і писати;
- *noaccess* : всі файли і директорії нижче вказаної будуть не доступні;
- *link_relative* : якщо зустрічається абсолютне символічне посилання його змінювати на відносне (../../);
- *link_absolute* : не змінювати символічні посилання;
- *no_root_squash* : дозволяє доступ до сервера з uid root — типово клієнтський користувач root прив'язується до anonymous. Важливо для бездискових станцій, що пишуть на диск, щоб їх не "squash"ило;
- *no_all_squash* : Важливо для бездискових станцій, щоб їх не "squash"ило;
- *all_squash* : прив'язує всіх віддалених користувачів до anonymous. Тобто вони матимуть доступ тільки до файлів з публічними правами, а файли що вони створюватимуть, будуть доступні всім користувачам anonymous. Тобто всі користувачі є nobody, оскільки 2 не присвоюється для *UID*, а *GID* використовується 65534. Щоб змінити *UID/GID* потрібно явно вказати anonuid=1111 та anongid=1111.
- *map_static* : типово, права доступу в *NFS*, для ідентичних користувачів, що мають однакові *UID* на сервері та клієнтській машині будуть прив'язані автоматично. Тобто користувачі будуть мати такі ж допуски до файлів на віддаленому сервері, якби вони знаходились локально.

Є декілька рівнів безпеки NFS.

Рівень безпеки при монтуванні дозволяє вказати котрим комп'ютерам дозволено монтувати систему і з якими правами (читання/запис).

Користувацький рівень безпеки дозволяє зв'язати користувача локальної системи з користувачем на NFS сервері, тоді користувач буде мати стандартні права і допуски в UNIX [5.1, 5.12].

Також можна використовувати політики SELinux(8).

І нарешті останній файл, який прийдеться відредагувати для сервера: */etc/conf.d/nfs* котрий визначає декілька ініціалізаційних опцій, ось його вигляд у *Gentoo*:

$ cat /etc/conf.d/nfs

 # Config file for /etc/init.d/nfs

 # Number of servers to be started up by default

RPCNFSDCOUNT=16

 # Options to pass to rpc.mountd

RPCMOUNTDOPTS="“

RPCNFSDCOUNT - рівне кількості клієнтських машин, можливо потрібно змінити. Далі, як звичайно:

 # /etc/init.d/nfs start

І щоб цей скрипт стартував при включенні системи виконуємо:

 # rc-update add nfs default

Для інших потрібно пам'ятати, що nfs запускається після portmap чи rpc.portmap. Тобто порядок запуску в ручну такий:

rpc.portmap

rpc.mountd

rpc.nfsd

rpc.statd

rpc.lockd (якщо потрібно)

rpc.rquotad.

Можна перевірити чи встановлене ядро підтримує NFS, для цього виконуємо: $ cat /proc/filesystems

якщо є стрічка: nodev nfs

Значить ядро зібране з підтримкою NFS і наступний пункт можна пропустити.

Детальна документація про збирання ядра Лінукс. Додатково потрібно ввімкнути:

```
Networking  --->
  [*] Networking support
        Networking options  --->
            <*> Packet socket
            [*]   Packet socket: mmapped IO
            <*> Unix domain sockets
            <*> IPsec user configuration interface
            <*> PF_KEY sockets
            [*] TCP/IP networking
            .........
File systems  --->
.........
    Network File Systems  --->
        <*> NFS file system support
        [*]   Provide NFSv3 client support
        [*]    Provide client support for the NFSv3 ACL protocol extension
        [*]   Provide NFSv4 client support (EXPERIMENTAL)
        [ ]   Allow direct I/O on NFS files (EXPERIMENTAL)
        ............
        [*]   Root file system on NFS                    Для бездискових станцій
        --- Secure RPC: Kerberos V mechanism (EXPERIMENTAL)
```

Файл /etc/fstab на клієнтських комп'ютерах вказує, що експортує сервер і що клієнти можуть змонтувати в себе. Клієнтський fstab файл, для бездискових станцій знаходиться в /diskless/10.0.0.101/etc/fstab та повинен виглядати приблизно так:

```
# these entries are essential
    10.0.0.10:/diskless/10.0.0.101    /          nfs
    sync,hard,intr,rw,nolock,rsize=8192,wsize=8192  0 0
    10.0.0.10:/opt                    /opt       nfs
     sync,hard,intr,ro,nolock,rsize=8192,wsize=8192  0 0
    10.0.0.10:/usr                    /usr       nfs
    sync,hard,intr,ro,nolock,rsize=8192,wsize=8192  0 0
    10.0.0.10:/home                   /home      nfs
    sync,hard,intr,rw,nolock,rsize=8192,wsize=8192  0 0
    none                         /proc      proc      defaults          0 0
    10.0.0.10:/var/log      /var/log    nfs       hard,intr,rw      0 0
    # для підтримки SELinux
    none                 /selinux    selinuxfs  defaults                0 0
    #Тільки для кластерів openMosix
    none                 /mnt/mfs    mfs        dfsa=1                  0 0
```

В цьому прикладі 10.0.0.10 - адреса IP сервера NFS. Перше поле визначає директорію, що експортується (на сервері), друге — точку монтування на бездисковій станції, третє — файлову систему (для NFS — відповідно, nfs). В четвертому полі визначаються додаткові параметри, які використовуються при монтуванні розділу, список параметрів наведено у mount(8). Для простоти у прикладі використано жорсткі точки монтування, але за допомогою різних параметрів у /etc/fstab можна зробити кластер більш ефективним.

Можливі проблеми та шляхи їх розв'язання.

- Після зміни файлу */etc/exports* варто виконати команду:

  # exportfs -a -v

- Вона перевірить його на помилки та зробить доступними усі публічні ресурси (-a), а якщо щось не так — докладно про це повідомить (-v).
- Для перевірки запущених сервісів portmapper виконуємо:

  # rpcinfo -p

- Щоб переконатись, що portmapper та обгортання TCP працює вірно, запускаємо:

  # strings /sbin/portmap | grep hosts

- Уважно читаємо та аналізуємо логи.

Перша версія застосовувалася тільки для внутрішнього використання в Sun в експериментальних цілях [5.1].

Версія 2 (NFSv2) випущена в березні 1989 року, спочатку повністю працювала по протоколу UDP [5.4, 5.15-5.20]. Розробники вирішили не зберігати даних про внутрішній стан всередині протоколу, як приклад, блокування, реалізована поза базового протоколу. Люди, залучені в створення NFS версії 2 - Рости Сендберг (Rusty Sandberg,) Боб Лайон (Bob Lyon), Білл Джой і Стів Клейман (Steve Kleiman).

NFSv3 вийшла в червні 1995 року, в ній додана підтримка дескрипторів файлів змінного розміру до 64 байт (у версії 2 - масив фіксованого розміру 32 байта), знято обмеження на 8192 байт в RPC-виклики читання і запису (тим самим, розмір переданого блоку у викликах обмежений тільки межею для UDP-датаграми - 65535 байт), реалізована підтримка файлів великих розмірів, підтримані асинхронні виклики операцій записи, до процедур READ і WRITE додані виклики ACCESS (перевірка прав доступу до файлу), MKNOD (створення спеціального файлу Unix), READDIRPLUS (повертає име а файлів в каталозі разом з їх атрибутами), FSINFO (повертає статистичну інформацію про файлову систему), FSSTAT (повертає динамічну інформацію про файлову систему), PATHCONF (повертає POSIX.1-інформацію про фото) і COMMIT (передає раніше зроблені асинхронні записи на постійне зберігання) [5.1, 5.3].

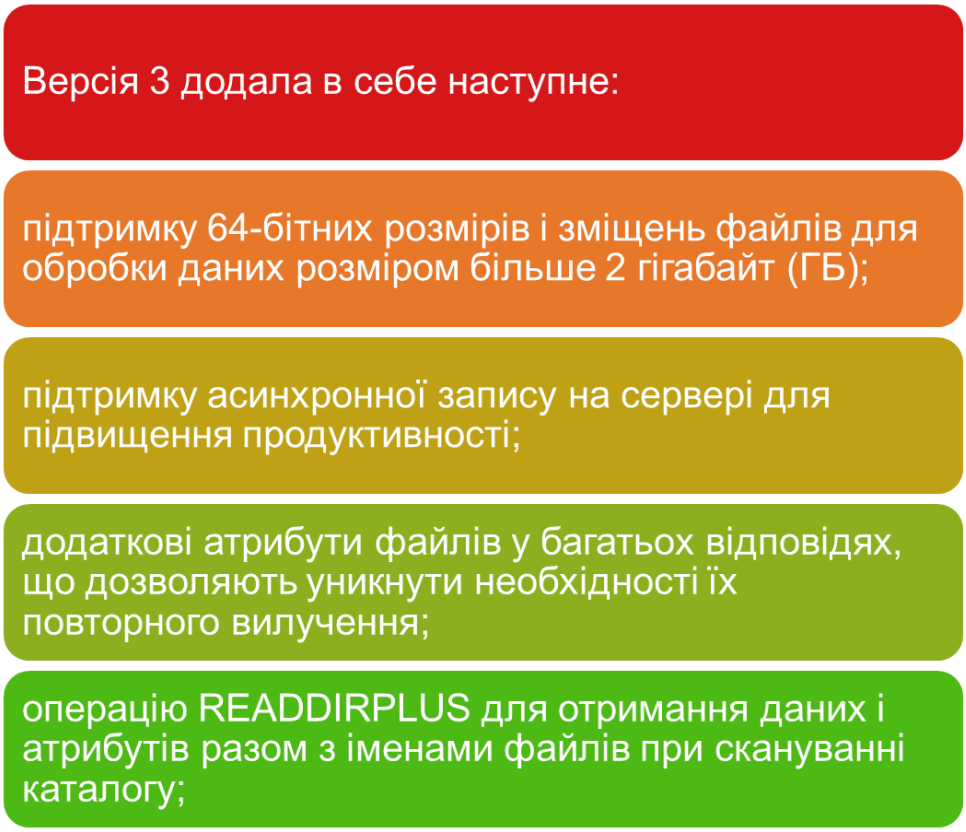


Рис. 5.2. Особливості NFSv3

На момент введення версії 3 відзначено зростання популярності в середовищі розробників протоколу TCP. Деякі незалежні розробники самостійно додали підтримку протоколу TCP для NFS версії 2 в якості транспортного, Sun Microsystems додали підтримку TCP в NFS в одному з додатків до версії 3. З підтримкою TCP з'явилася можливість використання NFS в глобальних мережах.

WebNFS - це розширення для NFS версій 2 і 3, яке дозволяють легше інтегруватися в веб-браузери і дає можливість роботи через брандмауер. Різні сторонні протоколи стали асоціюватися з NFS, в тому числі [5.19-5.26]:

1. Менеджер блокувань мережі (NLM - Network Lock Manager) і монітор стану мережі (NSM - Network Status Monitor) разом забезпечують засоби для блокування файлів в мережі. Ці кошти, хоча формально не пов'язані з NFS, можна знайти в більшості реалізацій NFS. Вони забезпечують сервіси, неможливі в базовому протоколі. NLM і NSM реалізують функціонування сервера за допомогою демонів lockd і statd, відповідно.

2. Протокол видаленої інформації про квоти (RQUOTAD) (NFS дозволяє користувачам переглядати дискову квоту на віддаленому NFS сервері) [5.1, 5.20-5.24].

Хоча NFS найчастіше використовують в Unix-подібних системах, даний протокол можна також використовувати і на інших операційних системах, таких, як Mac OS Classic, OpenVMS, Microsoft Windows, Novell NetWare, і IBM.

У число альтернативних протоколів віддаленого доступу до файлів входять Server Message Block (SMB, також відомий як CIFS) протокол, Apple Filing Protocol (AFP), NetWare Core Protocol (NCP). Під операційною системою Microsoft Windows SMB і NetWare Core Protocol (NCP) використовується частіше, ніж NFS. У системах Macintosh AFP зустрічається частіше, ніж NFS.

NFSv4 випущена в грудні 2000 року під впливом AFS та CIFS, в неї включені поліпшення продуктивності і безпеки [5.1-5.5, 5.23]. Версія 4 стала першою версією, розробленої спільно з Internet Engineering Task Force (IETF). NFS версії v4.1 була схвалена IESG в січні 2010 року (нова специфікація, об'ємом 612 сторінок, стала відома як найдовший документ, схвалений IETF). Важливим нововведенням версії 4.1 є специфікація pNFS - Parallel NFS, механізму паралельного доступу NFS-клієнта до даних безлічі розподілених NFS-серверів. Наявність такого механізму в стандарті мережевої файлової системи допоможе будувати розподілені хмарні сховища та інформаційні системи.

Версія 4 розроблена під впливом Эндрской файлової системи (AFS) і блоки повідомлень сервера SMB, також звана CIFS), включає в себе підвищення продуктивності, забезпечує кращу безпеку і вводить протокол з дотриманням встановлених умов.

Версія 4 стала першим дистрибутивом, розробленим в Цільовій групі Internet Engineering Task Force (IETF) після того, як Sun Microsystems передала розробку протоколів стороннім фахівцям. NFS версія 4.1 спрямована на надання підтримки протоколу для використання кластерних розгортання серверів,

включаючи можливість надання масштабованого паралельного доступу до файлів, розподіленим між декількома серверами (розширення pNFS). Новітній протокол файлової системи NFS 4.2 (RFC 7862) – був офіційно випущений в листопаді 2016 року.

5.7. L2TP протокол тунелювання другого рівня

L2TP (Layer 2 Tunneling Protocol - протокол тунелювання другого рівня) - в комп'ютерних мережах тунельний протокол, що використовується для підтримки віртуальних приватних мереж.

Головна перевага L2TP полягає в тому, що цей протокол дозволяє створювати тунель не тільки в мережах IP, але і в таких, як ATM, X.25 і Frame Relay [5.1, 5.14, 5.30-5.34].

Незважаючи на те, що L2TP діє на зразок протоколу канального рівня моделі OSI, насправді він є протоколом сеансового рівня і використовує зареєстрований UDP-порт 1701 [5.15].

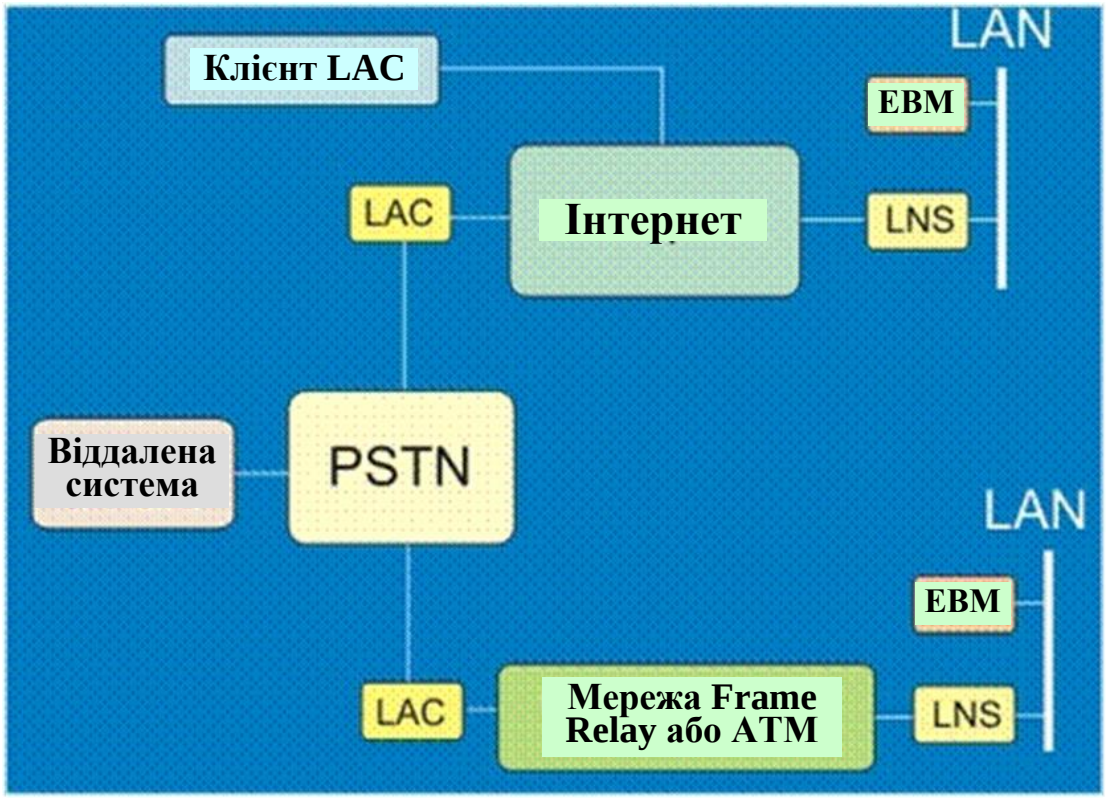


Рис. 5.3. Схема роботи протокола L2TP

- *LAN* - локальні мережі (Local Area Network), до яких підключаються через L2TP;
- *EOM* - комп'ютер (и), підключені до локальної мережі безпосередньо;
- *LNS - L2TP Network Server,* сервер доступу до локальної мережі по L2TP;
- *LAC - L2TP Access Concentrator,* пристрій для прозорого підключення своїх користувачів до LNS через мережу тієї чи іншої архітектури;
- *Віддалена система* - система, яка бажає підключитися до LAN через L2TP;
- *Клієнт LAC* - EOM, яка сама для себе виконує роль LAC для підключення до LNS;
- *PSTN* - коммутируемая телефонна мережа (Public Switched Telephone Network);
- *Інтернет, Мережа Frame Relay або ATM* - мережі різних архітектур.

Метою тут є тунелювання кадрів PPP між вилученою системою або клієнтом LAC і LNS, розміщеному в LAN [5.16, 5.23-5.25].

Дистанційна система ініціює PPP-з'єднання з LAC через телефонну мережу PSTN (Public Switched Telephone Network). LAC потім прокладає тунель для PPP-з'єднання через Інтернет, Frame Relay або ATM до LNS, і таким чином здійснюється доступ до вихідної LAN. Адреси віддаленій системі надаються вихідної LAN через узгодження з PPP NCP. Аутентифікація, авторизація та аккаунтинга можуть бути надані областю управління LAN, як якщо б користувач був безпосередньо з'єднаний з сервером мережевого доступу NAS.

LAC-клієнт (EOM, яка виконує програму L2TP) може також брати участь в тунелюванні до вихідної LAN без використання окремого LAC, якщо EOM, що містить програму LAC-клієнта, вже має з'єднання з Інтернет. Створюється «віртуальне» PPP-з'єднання, і локальна програма L2TP LAC формує тунель до LNS. Як і в описаному вище випадку, адресація, аутентифікація, авторизація та аккаунтинга будуть забезпечені областю управління вихідної LAN.

L2TP використовує два види пакетів:

- керуючі та інформаційні повідомлення.
- Керуючі повідомлення використовуються при встановленні, підтримці і анулювання тунелів і викликів.
- Інформаційні повідомлення використовуються для інкапсуляції PPP-кадрів, що пересилаються по тунелю.
- Керуючі повідомлення використовують надійний контрольний канал в межах L2TP, щоб гарантувати доставку.
- Інформаційні повідомлення при втраті не відсилаються повторно.

<table>
<tr><td colspan="2">PPP кадри</td></tr>
<tr><td>L2TP інформаційні повідомлення</td><td>L2TP управлячі повідомлення</td></tr>
<tr><td>L2TP інформаційний канал (ненадійний)</td><td>L2TP канал управління (надійний)</td></tr>
<tr><td colspan="2">Транспортування пакетів (UDP, FR, ATM і т.д.)</td></tr>
</table>

Рис. 5.4. Структура протокола

Керуючий повідомлення має порядковий номер, який використовується в керуючому каналі для забезпечення надійної доставки.

Інформаційні повідомлення можуть використовувати порядкові номери, щоб відновити порядок пакетів і детектувати втрату кадрів. Всі коди надсилаються в порядку, прийнятому для мереж [5.1, 5.14-5.17].

Пакети L2TP для контрольного та інформаційного каналів використовують один і той же формат заголовка (рис. 5.5).

<table>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th></th><th></th><th></th><th>31</th></tr>
<tr><td>T</td><td>L</td><td>x</td><td>x</td><td>S</td><td>x</td><td>O</td><td>P</td><td>x</td><td>x</td><td>x</td><td>x</td><td colspan="4">Версія</td><td colspan="5">Довжина (опц)</td></tr>
<tr><td colspan="16">ID тунеля</td><td colspan="5">ID сесії</td></tr>
<tr><td colspan="16">Ns (опц)</td><td colspan="5">Nr (опц)</td></tr>
<tr><td colspan="16">Offset Size (опц)</td><td colspan="5">Offset Pad (опц).....</td></tr>
<tr><td colspan="21">Payload data</td></tr>
</table>

Рис. 5.5. Формат заголовка

• Біт тип (T) характеризує різновид пакету.

Він встановлюється рівним 0 для інформаційних повідомлень і 1 - для керуючих.

• Якщо біт довжини (L) дорівнює 1, поле довжина присутнє.

Для керуючих повідомлень цей біт повинен дорівнювати 1.

• Біти x зарезервовані для майбутніх застосувань.

Всі зарезервовані біти повинні бути встановлені рівними 0 для вихідних повідомлень і ігноруватися для вхідних.

• Якщо біт послідовності (S) дорівнює 1, присутні поля Ns і Nr.

Біт S для керуючих повідомлень має дорівнювати 1.

• Якщо біт зсуву (O) дорівнює 1, поле величини зсуву присутній.

Біт O для керуючих повідомлень має дорівнювати 0.

• Біт пріоритету (P) має дорівнювати 0 для всіх керуючих повідомлень. Для інформаційних повідомлень - якщо цей біт дорівнює 1, це інформаційне повідомлення має пріоритет в черзі.

• Поле Ver вказує версію заголовка інформаційного повідомлення L2TP.

Значення 1 зарезервовано для детектування пакетів L2F в умовах, коли вони приходять упереміш з L2TP-пакетами. Пакети, отримані з невідомим значенням поля Ver, відкидаються [5.23-5.25]..

• Поле Довжина (опціонально) вказує загальну довжину повідомлення в октетах.

• ID-тунелю містить ідентифікатор керуючого з'єднання. Ідентифікатори тунелю L2TP мають тільки локальне значення. Тобто, різні кінці одного тунелю повинні мати різні ID. ID-тунелю в кожному повідомленні повинно бути тим, яке очікує отримувач. ID-тунелю вибираються при формуванні тунелю.

• ID-сесії визначає ідентифікатор для сесії даного тунелю. Сесії L2TP іменуються за допомогою ідентифікаторів, які мають тільки локальне значення. ID-сесії в кожному повідомленні повинно бути тим, яке очікує отримувач. ID-сесії вибираються при формуванні сесії.

• Поле Ns визначає порядковий номер інформаційного або керуючого повідомлення, починаючи з нуля і збільшуючись на 1 (по модулю 216) для кожного посланого повідомлення.

• Поле Nr містить порядковий номер, який очікується для наступного повідомлення. Таким чином, Nr робиться рівним Ns останнього за ліком отриманого повідомлення плюс 1 (по модулю 216). В інформаційних повідомленнях, Nr зарезервовано і, якщо присутній (це визначається S- бітом), має ігноруватися при отриманні.

• Поле величина зміщення (Offset Size), якщо є, специфікує число октетів після заголовка L2TP, де має починатися поле даних. Вміст заповнювач зміщення не визначене. Якщо поле зміщення присутній, заголовок L2TP завершується після завершального октету заповнювач зміщення.

Тип повідомлення AVP визначає специфічний тип посилається керуючого повідомлення [5.1, 5.14-5.23]. Управління контрольним з'єднанням:

• 0 (зарезервовано)

• 1 (SCCRQ) Start-Control-Connection-Request

• 2 (SCCRP) Start-Control-Connection-Reply

• 3 (SCCCN) Start-Control-Connection-Connected

• 4 (StopCCN) Stop-Control-Connection-Notification

• 5 (зарезервовано)

• 6 (HELLO) Hello

Управління викликами (Call Management)

• 7 (OCRQ) Outgoing-Call-Request

• 8 (OCRP) Outgoing-Call-Reply

• 9 (OCCN) Outgoing-Call-Connected

• 10 (ICRQ) Incoming-Call-Request

• 11 (ICRP) Incoming-Call-Reply

• 12 (ICCN) Incoming-Call-Connected

• 13 (зарезервовано)

• 14 (CDN) Call-Disconnect-Notify

Повідомлення про помилки

• 15 (WEN) WAN-Error-Notify

Управління сесією PPP

• 16 (SLI) Set-Link-Info

Необхідна процедура встановлення PPP-сесії тунелювання L2TP включає в себе два етапи:

• встановлення керуючого каналу для тунелю

• формування сесії відповідно до запиту вхідного або вихідного дзвінка.

Тунель і відповідний керуючий канал повинні бути сформовані до ініціалізації вхідного або вихідного дзвінка. L2TP-сесія повинна бути реалізована до того, як L2TP зможе передавати PPP-кадри через тунель. В одному тунелі можуть існувати кілька сесій між одними і тими ж LAC і LNS.

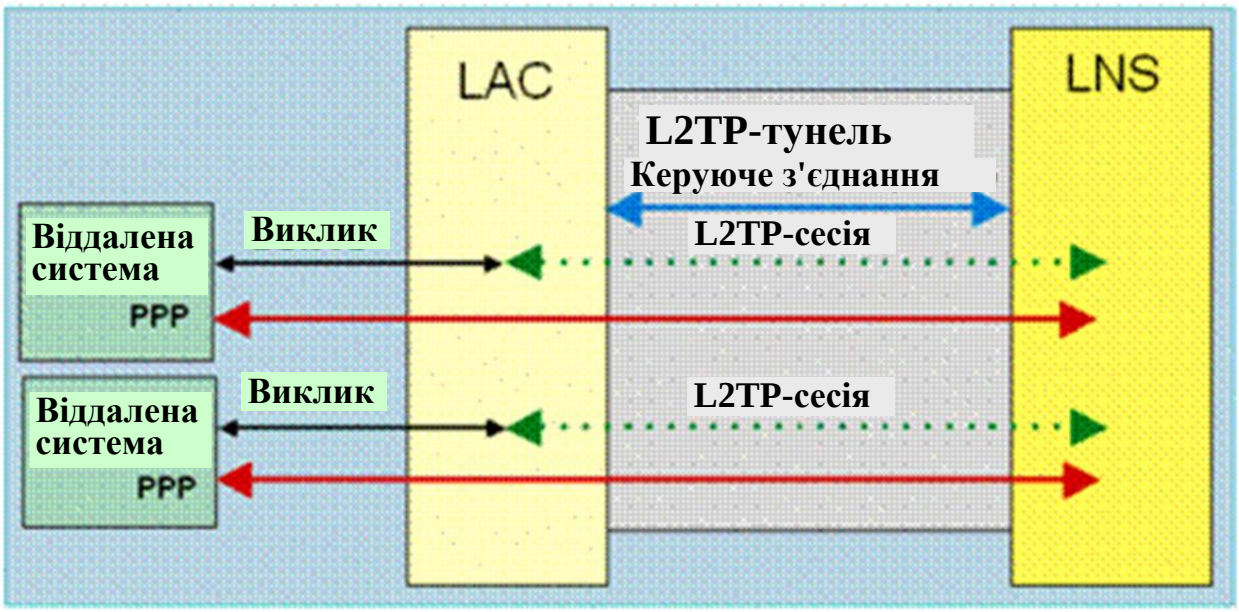


Рис. 5.6. Тунел кількох сесій між одними і тими ж LAC і LNS

Є первинним, яке повинно бути реалізовано між LAC і LNS перед запуском сесії. Встановлення керуючого з'єднання включає в себе безпечну ідентифікацію партнера, а також визначення версії L2TP, можливостей каналу, кадрового обміну і т. д. L2TP включає в себе просту, опціонну, CHAP-подібну систему аутентифікації тунелю в процесі встановлення керуючого з'єднання.

Після успішного встановлення керуючого з'єднання можуть формуватися

індивідуальні сесії. Кожна сесія відповідає одному PPP інформаційного потоку між LAC і LNS. На відміну від встановлення керуючого з'єднання, встановлення сесії є асиметричним щодо LAC і LNS. LAC запитує LNS доступ до сесії для вхідних запитів, а LNS запитує LAC запустити сесію для роботи з вихідними запитами [5.1, 5.17, 5.30-5.33].

Коли тунель сформований, PPP-кадри від віддаленої системи, одержані LAC, звільняються від CRC, канальних заголовків і т. п., інкапсульованих в L2TP, і переадресовуються через відповідний тунель. LNS отримує L2TP-пакет і обробляє інкапсульований PPP-кадр, як якщо б він був отриманий через локальний інтерфейс PPP.

Відправник повідомлення, асоційований з певною сесією і тунелем, поміщає ID сесії і тунелю (специфіковані партнером) у відповідних полях заголовка Ваших повідомлень.

Порядкові номери, певні в заголовку L2TP, використовуються для організації надійного транспортування керуючих повідомлень. Кожен партнер підтримує окрему нумерацію для керуючого з'єднання і для кожної інформаційної сесії в межах тунелю.

На відміну від каналу управління L2TP, інформаційний канал L2TP використовує нумерацію повідомлень не для повторної пересилки, а для детектування втрат пакетів і / або відновлення початкової послідовності пакетів, перемішаних при транспортуванні.

LNS може ініціювати заборону нумерації повідомлень в будь-який час в ході сесії (включаючи перше інформаційне повідомлення) [5.1, 5.20].

Механізм keepalive використовується L2TP для того, щоб розрізняти простої тунелю і тривалі періоди відсутності управління або інформаційної активності в тунелі.

Це робиться за допомогою керуючих повідомлень Hello після заданого періоду часу, який минув з моменту останнього одержання керуючого повідомлення через тунель.

При недоставці повідомлення Hello тунель оголошується неробочим, і система повертається в початковий стан.

Механізм переведення транспортної середовища в початковий стан шляхом введення повідомлень Hello гарантує, що розрив з'єднання між LNS і LAC буде детектувати на обох кінцях тунелю [5.1, 5.14-5.20].

Переривання сесії може бути ініційовано LAC або LNS і виконується шляхом посилки керуючого повідомлення CDN. Після того як остання сесія перервана, керуюче з'єднання може бути також розірвано.

Розрив контрольного з'єднання може бути ініційований LAC або LNS і виконується шляхом посилки одного керуючого повідомлення StopCCN.

Протокол L2TP є самодокументуємим, який працює поверх рівня, який служить для транспортування. Однак, необхідні деякі деталі підключення до середовища, для того щоб забезпечити сумісність різних реалізацій. Протокол L2TP стикається при своїй роботі з декількома проблемами безпеки. Нижче розглянуті деякі підходи для вирішення цих проблем.

Кінці тунелю можуть опціонно виконувати процедуру аутентифікації один одного при встановленні тунелю.

Ця аутентифікація має ті ж атрибути безпеки, що і CHAP, і володіє розумним захистом проти атак відтворення і спотворення в процесі встановлення тунелю.

Для реалізації аутентифікації LAC і LNS повинні використовувати загальний секретний ключ.

Забезпечення безпеки L2TP вимагає, щоб транспортне середовище могла забезпечити шифрування переданих даних, цілісність повідомлень і аутентифікацію послуг для всього L2TP-трафіку. Сам же L2TP відповідальний за конфіденційність, цілісність і аутентіфікацію L2TP-пакетів всередині тунелю.

При роботі поверх IP, IPsec (безпечний IP) надає безпеку на пакетному рівні [5.1, 5.20-5.26]. Всі керуючі та інформаційні пакети L2TP в конкретному тунелі виглядають для системи IPsec як звичайні інформаційні UDP / IP-пакети.

Крім транспортної безпеки IP, IPsec визначає режим роботи, який дозволяє туннелировать IP-пакети, а також засоби контролю доступу, які необхідні для додатків, що підтримують IPsec. Ці засоби дозволяють фільтрувати пакети на основі характеристик мережевого і транспортного рівнів. У моделі L2TP-тунелю аналогічна фільтрація виконується на PPP-рівні або мережевому рівні поверх L2TP.

Література, що використовувалась в цьому мадеріалі також рекомендована для ретельного самостійного вивчення.

5.8. iSNS мережевий протокол автоматизації в TCP / IP мережах

Internet Storage Name Service (iSNS) (Служба імен сховищ Інтернету) - мережевий протокол, що дозволяє автоматизувати відкриття, управління і конфігурація ISCSI і Fibre Channel пристроїв (з використанням iFCP шлюзів) в TCP / IP мережах. Протокол iSNS описується в RFC 4171.

iSNS надає управління сервісами аналогічно тим, які є в сімействі Fibre Channel, що дозволяє для стандартної IP мережі працювати багато в чому таким же чином, як працює в Fibre Channel мережу зберігання даних. Через те, що iSNS здатний емулювати фабрику сервісів Fibre Channel, і управляти як ISCSI, так і Fibre Channel пристроями, iSNS сервер може бути використаний в якості об'єднуючого пункту для всієї мережі зберігання. Хоча слід зазначити, що стандартами iSNS є обов'язковим підтримка iFCP протоколу, що підтримує ISCSI [5.1, 5.34-5.35].

Стандарт iSNS визначає чотири компоненти (RFC 4171 — Internet Storage Name Service (iSNS)).

Протокол iSNSP це такий протокол, який визначає, яким чином клієнти і сервери iSNS спілкуються між собою. Він призначений для використання на різних платформах, включаючи комутатори та кінцеві об'єкти. iSNSP ґрунтується на повідомленнях запитів і відповідей, які разом формують повну

транзакцію [5.1, 5.35]:

iSNS-клієнтами можуть бути як пристрої зберігання даних, що надають свої послуги через мережу, так і комп'ютери, які користуються (опосередковано через мережі передачі даних) послугами мережевих пристроїв зберігання даних. iSNS-клієнти взаємодіють з iSNS-серверами по протоколу iSNSP.

Пристрої зберігання даних (наприклад iSCSI Target або FC Storage), використовуючи протокол iSNSP, реєструються на iSNS-сервері, надаючи йому інформацію про атрибути пристрою, в результаті чого:

а) стають членами Домену Виявлення (Discovery Domain, DD);

б) можуть завантажити з iSNS-сервера інформацію про інших зареєстрованих пристроях зберігання;

в) можуть отримувати від iSNS-сервера асинхронні повідомлення про події, що сталися в їх домені Виявлення (RFC4171);

г) можуть зобов'язатися повідомляти iSNS-сервер про зміни свого стану (доступності).

Комп'ютери, що користуються через мережу послугами пристроїв зберігання даних (iSCSI-ініціатори), - отримують від iSNS-сервера інформацію про доступні в їх домені Виявлення пристроях зберігання даних, а також оповіщення про появу нових і видаленні (недоступності) існували пристроїв зберігання даних.

iSNS сервери реагують на запити iSNS протоколу, а також на запити, зроблені iSNS клієнтами, які використовують iSNSP. iSNS сервери ініціюють повідомлення про зміни в iSNSP і зберігають інформацію, належним чином минулий перевірку справжності і представляє собою заявку на реєстрацію в базі даних iSNS [7, RFC 4171].

Бази даних iSNS це інформаційні сховища (репозиторії) для iSNS серверів. Вони зберігають інформацію про атрибути клієнтів iSNS, причому каталоги зберігання можуть відрізнятися в залежності від реалізації iSNS, наприклад, вони можуть зберігати атрибути клієнта в каталозі LDAP.

iSNS забезпечує чотири основних сервіси:

1. Реєстрація імен та пошук ресурсів зберігання даних.

2. Сервіс реєстрації імен надає всім об'єктам в мережі можливість зареєструватися і опитувати бази даних для пошуку ресурсів зберігання даних. Наприклад, клієнти-ініціатори можуть отримати від iSNS сервера інформацію про інших ініціаторів (наприклад, iSCSI-ініціаторів) і кінцевих об'єктах (наприклад, iSCSI-цілях (iSCSI Target)).

3. Дослідження домену та авторизації. Адміністратори можуть використовувати домени виявлення для поділу пристроїв зберігання даних на керовані групи. Для цього угруповання адміністратори можуть обмежувати авторизацію кожного вузла в найбільш підходящою підмережі, зареєстрованої в iSNS, що дозволяє розширити мережу сховищ даних за рахунок скорочення числа непотрібних запитів на авторизацію шляхом обмеження часу, який кожен вузол витрачає на встановлення входу в мережу.

Кожен вузол може використовувати Login Control для делегування свого управління доступом і політики авторизації iSNS сервера. Таке делегування покликане сприяти централізації управління доступом.

4. Сервіс повідомлення про зміни (SCN) дозволяє серверам iSNS видавати повідомлення про кожну подію, яке зачіпає вузли зберігання даних керованої ними мережі.

Кожен клієнт iSNS може зареєструватися для отримання повідомлень від імені своїх вузлів зберігання, і кожен клієнт буде реагувати на це відповідно до своїх власних вимог і реалізацією.

Двонаправлені відображення між Fibre Channel і ISCSI пристроями.

Через те, що в базах даних iSNS зберігаються імена і пошукова інформація про Fibre Channel і ISCSI пристроях, iSNS сервери здатні зберігати відображення Fibre Channel пристроїв до проксі-пристроїв ISCSI в мережі IP.

Ці відображення може бути також зроблені і в протилежному напрямку, що дозволяє iSNS серверів зберігати відображення ISCSI пристроїв до проксі-

WWNs.

5.9. SMPP – короткі повідомлення тимчасової мережі

SMPP - (Short Message Peer-to-Peer) короткі повідомлення тимчасової мережі. Є відкритим стандартом в телекомунікаційній галузі, який розроблений спеціально, щоб забезпечити гнучкий інтерфейс для передачі коротких повідомлень між зовнішніми сутностями (пристрої, додатки) коротких повідомлень (ESME), маршрутизаторами (RE) і центрами повідомлень (SMSC).

SMPP часто використовують треті особи, такі як постачальники додаткових послуг, агентства новин, для передачі повідомлень - часто масово, передачі SMS повідомлень [5.1, 5.32].

За даним протоколом можна передавати SMS, EMS, голосову пошту, повідомлення, стільникове радіомовлення, WAP-повідомлення, USSD повідомлення та ін.

Через свою універсальність і підтримкою SMS-протоколів без GSM, як UMTS, IS-95 (CDMA), CDMA2000 , ANSI-136 (TDMA) і подібних, SMPP є найбільш широко використовуваним протоколом для короткого обміну повідомленнями за межами мереж ОКС7 (SS7).

SMPP використовує модель роботи клієнт-сервер. Центр повідомлень (SMSC), як правило, виступає в якості сервера, очікуючи підключення від клієнта - ESME. Коли SMPP використовується для SMS пірінга, що відправляє МС зазвичай виступає в якості клієнта.

Протокол заснований на парах запит-відповідь PDU (блоків даних протоколу або пакетах) обмінюються через OSI шар 4 (TCP сесії або X25 SVC3) з'єднань. Загальновідомий порт, призначений IANA для SMPP при роботі над TCP є 2775, але часто використовуються довільні номери портів.

Перед обміном повідомлень, повинна бути відправлена і підтверджена команда прив'язки. Команда прив'язки визначає, в якому напрямку можна буде

відправляти повідомлення; bind_transmitter дозволяє клієнту тільки відправляти повідомлення на сервер, bind_receiver означає, що клієнт буде тільки отримувати повідомлення, і bind_transceiver (введений в SMPP 3.4) дозволяє передавати повідомлення в обох напрямках [5.1, 5.21-5.25].

При використанні команди прив'язки, ESME повинен себе ідентифікувати за допомогою параметрів system_id, system_type і password; address_range призначений для вказівки адреси ESME, але зазвичай, передається порожнім. Так само, в команді зв'язки є interface_version, в якому вказують версію протоколу, який буде використовуватися під час сесії.

Обмін повідомленнями може бути синхронним, де кожен вузол очікує відповіді на кожен PDU або асинхронний, де множинні запити можуть бути відправлені без очікування відповіді; кількість непідтверджених запитів називається вікно; для найкращого взаємодії в зв'язці, обидві сторони повинні мати ідентичний настройки розміру вікна.

У SMPP блоки PDU є бінарними і кодуються для ефективності. Вони починаються з заголовка, який відповідають тілу (рис. 5.7):

<table>
<tr><th colspan="5">SMPP PDU</th></tr>
<tr><td colspan="4">PDU Header (обов’язково)</td><td>PDU Body (опціонально)</td></tr>
<tr><td>Command length</td><td>Command Id</td><td>Command Status</td><td>Sequence Id</td><td>PDU Body</td></tr>
<tr><td>4 octets</td><td colspan="4">Length = (Command Length value — 4) octets</td></tr>
</table>

Рис. 5.7. Заголовок відповідного тіла SMPP і блоки PDU

Кожен PDU починається з заголовка. Тема складається з 4-х полів, довжина кожного з яких дорівнює 4 октетам: command_length

Загальна довжина PDU в октетах (включаючи саму команду довжини поля); повинна становити не менше 16 октетів, так як кожен PDU повинен

містити заголовок 16 октету: command_id;

Вказує операцію SMPP (або команди): command_status;

Завжди значення 0 в запитах; у відповіді він несе в собі інформацію про результат операції: sequence_number.

Використовується для кореляції запитів і відповідей в рамках сесії SMPP; забезпечує асинхронну зв'язок (за допомогою методу «ковзного вікна»).

Всі числові поля в SMPP користуються великим протилежне, а це значить, що перший октет є старшим байтом (MSB).

Це приклад двійкового кодування 60 октетів submit_sm PDU. Дані представлені в значеннях октетне Hex як єдиного дампа і потім заголовка і тіла розбивкою цього PDU [5.1, 5.32].

Обсяг значень показаний в десятковому вигляді в дужках і значеннями Hex після них. Якщо ви бачите один або кілька шістнадцятирічних октету, то це тому, що даний розмір поля використовує 1 або більше октетів кодування.

- **Заголовок PDU**

'command_length', (60) ... 00 00 00 3C

'command_id', (4) ... 00 00 00 04

'command_status', (0) ... 00 00 00 00

'sequence_number', (5) ... 00 00 00 05

Зміст PDU

'service_type', () ... 00

'source_addr_ton', (2) ... 02

'source_addr_npi', (8) ... 08

'source_addr', (555) ... 35 35 35 00

'dest_addr_ton', (1) ... 01

'dest_addr_npi', (1) ... 01

'dest_addr', (555555555) ... 35 35 35 35 35 35 35 35 35 00

'esm_class', (0) ... 00

'protocol_id', (0) ... 00

'priority_flag', (0) ... 00

'schedule_delivery_time', (0) ... 00

'validity_period', (0) ... 00

'registered_delivery', (0) ... 00

'replace_if_present_flag', (0) ... 00

'data_coding', (3) ... 03

'sm_default_msg_id', (0) ... 00

'sm_length', (15) ... 0F

'short_message', (Hello Wikipedia) ... 48 65 6C 6C 6F 20 77 69 6B 69 70 65 64 69 61'

Зверніть увагу, що текст в поле short_message повинен відповідати data_coding. Коли data_coding 8 (UCS2), текст повинен бути в UCS-2BE (або його розширення, UTF-16BE). Коли data_coding вказує на 7-бітну систему кодування, кожен септет зберігається в окремому октеті в поле short_message (з найбільш старшим бітом, встановленим у 0).

SMPP 3.3 data_coding точно скопійовані значення TP-DCS GSM 03.38, що робить його придатним тільки для GSM 7-бітного алфавіту за замовчуванням, UCS2 або бінарних повідомлень; SMPP 3.4 введений новий список значень data_coding (таб. 28).

Опис data_coding = 4 або 8 є таким же, як в SMPP 3.3. Інші значення в діапазоні 1-15 зарезервовані в SMPP 3.3. На жаль, на відміну від SMPP 3.3, де data_coding = 0 був однозначно GSM 7-бітний алфавіт, для SMPP 3.4 і вище GSM 7-бітний алфавіт за замовчуванням відсутній в цьому списку, і data_coding = 0 можуть відрізнятися для різних СМСЦ - це може бути ISO-8859-1, ASCII, GSM 7-бітний алфавіт, UTF-8 або будь-яка інша налаштована для кожного ESME [5.1, 5.25-5.29].

Таблиця 5.1. SMPP 3.4 введен новый список значений data_coding:

data_coding	Значення	data_coding	Значення
0	SMSC Default Alphabet (SMPP 3.4) / MC Specific (SMPP 5.0)	10	ISO-2022-JP (Music Codes)
1	IA5 (CCITT T.50)/ASCII (ANSI X3.4)	11	Зарезервований
2	Octet unspecified (8-bit binary)	12	Зарезервований
3	Latin 1 (ISO-8859-1)	13	Extended Kanji JIS (X 0212-1990)
4	Octet unspecified (8-bit binary)	14	KS C 5601
5	JIS (X 0208-1990)	15-191	Зарезервований
6	Cyrillic (ISO-8859-5)	192—207	GSM MWI control — see GSM 03.38
7	Latin/Hebrew (ISO-8859-8)	208—223	GSM MWI control — see GSM 03.38
8	UCS2 (ISO/IEC-10646)	224—239	Зарезервований
9	Pictogram Encoding	240—255	GSM message class control — see GSM 03.38

При використанні data_coding = 0, обидві сторони (ESME і SMSC) повинні бути впевнені, що вони вважають це покажчиком до однієї і тієї ж кодуванні. В іншому ж випадку, краще не використовувати data_coding = 0. Це може ускладнити використання GSM 7-бітного алфавіту, так як деякі СМСЦ вимагає data_coding = 0, інші, наприклад, data_coding = 241.

SMPP був реалізований на Java в проект jSMPP. Він використовується в Apache Camel і різних інших популярних безкоштовних програмних проектах для SMS-повідомлень. Альтернативна реалізація Java nmote-smpp. Проект python-SMPP надає SMPP для користувачів Python. Проект PHP-SMPP представляє SMPP для користувачів PHP.

Незважаючи на широке поширення, SMPP має ряд проблемних особливостей:

- Ні data_coding для GSM 7 bit default alphabet
- Ні стандартного значення data_coding = 0
- Нечітка підтримка кодування Shift-JIS

• Несумісність submit_sm_resp між версіями SMPP

• Особливості ID повідомлення при отриманні звіту про доставку, при використанні SMPP 3.3.

Немає data_coding для GSM 7 bit default alphabet.

У SMPP 3.3 всі значення data_coding трактуються відповідно до GSM 03.38, але починаючи з SMPP 3.4 відсутній значення data_coding для GSM 7 bit default alphabet.

Немає стандартного значення data_coding = 0

Згідно SMPP 3.4 і 5.0 data_coding = 0 означає "SMSC Default Alphabet". Яка кодування це насправді, залежить від типу SMSC і його конфігурації.

Нечітка підтримка кодування Shift-JIS.

Одна з кодувань в CDMA стандартної C.R1001 є Shift-JIS використовується для японської мови. SMPP 3.4 і 5.0 визначає три кодування для Японії (JIS, ISO-2022-JP і Розширена кандзі JIS), але жоден з них не збігається з CDMA MSG_ENCODING 00101.

5.10. SDP- транспортно-агностичний протокол

Sockets Direct Protocol (SDP) - це транспортно-агностичний протокол для підтримки потокових сокетів Видаленого доступу до пам'яті (RDMA) в "фабричній мережі" (Fabric Network). SDP спочатку визначалася Software Working Group (SWG) з Торгової асоціації InfiniBand. Спочатку вона розроблялася для InfiniBand [5.1, 5.33]. Зараз SDP підтримується Альянсом OpenFabric.

SDP позначає стандартний провідний протокол (Wire protocol) на фабриці RDMA для підтримки потокових сокетів (SOCK_STREAM). SDP використовує різні функції RDMA мережі для передачі даних без копіювання (Zero-copy). SDP це чистий провідний протокол на рівні спеціалізації і не йде ні в один сокет API або його специфічних реалізацій.

Мета SDP - надавати RDMA-прискорену альтернативу до TCP на IP. Метою є зробити це в такій формі, яка була б прозора для додатка.

Solaris 10 і Solaris 11 Express включають підтримку SDP. Деякі інші UNIX подібні операційні системи планують включити підтримку SDP. Windows пропонує підсистему Winsock Direct, яка може використовуватися і для підтримки SDP.

Підтримка SDP також представлена у Випуску JDK7 платформи Java для додатків, випущених для операційних систем Solaris і Linux. [5.1, 5.33] База даних Oracle 11g підтримує з'єднання через SDP.

SDP оперує тільки з потоковими сокетами і, якщо він встановлений в систему, обходить стек TCP / IP для потокового з'єднання між будь-кінцевій точці в структурі RDMA. Всі інші типи сокетів підтримуються стеком IP в Linux системах і оперують через стандартні інтерфейси IP. Стек IP не має залежності на стек SDP, а стек SDP залежимо від IP драйверів для локальних призначень IP і для дозволів IP адрес в ідентифікації кінчений точки.

SDP використовується в Telstra на її 3G платформі Next G для доставки потокового мобільного телебачення [5.1, 5.34].

Проти підтримки цього протоколу в останніх випусках виступає OFED і шукає альтернативи. Ймовірними пропозиціями виступлять RSOCKET, WINSOCK і т.д.

5.11 SCP - протокол управління сеансом

Протокол управління сеансом (SCP) - це метод створення декількох з'єднань з малим навантаженням з одного з'єднання TCP (Transmission Control Protocol) [5.1, 5.28, 5.35].

Кілька таких легких з'єднань можуть бути активними одночасно. SCP - це протокол рівня сеансу в моделі взаємодії відкритих систем (OSI). На рівні сеансу (на п'ятому рівні в семишарові телекомунікаційної моделі) протоколи

рівня сеансу надають послуги для координації зв'язку між локальними і віддаленими програмами, встановлення, управління та завершення з'єднань.

SCP працює по протоколу TCP, в залежності від нього для забезпечення з'єднань і надійного обслуговування. Хоча SCP є байтовими-орієнтованим протоколом, він також підтримує маркери кордону повідомлень. TCP - це набір правил, використовуваних разом з Інтернет-протоколом (IP) для передачі даних у вигляді блоків повідомлень між комп'ютерами через Інтернет.

5.12. Стек протоколів NetBIOS

Досить популярний стек протоколів, розробкою якого займалися компанії IBM і Microsoft, відповідно, орієнтований на використання в продуктах цих компаній [5.1, 5.25]. Як і у TCP / IP, на фізичному і канальному рівні стека NetBIOS / SMB працюють стандартні протоколи, такі як Ethernet, Token Ring і інші, що робить можливим його використання в парі з будь-яким активним мережевим обладнанням. На верхніх же рівнях працюють протоколи NetBIOS (Network Basic Input / Output System) і SMB (Server Message Block)].

Протокол NetBIOS був розроблений в середині 80-х років минулого століття, але незабаром був замінений на більш функціональний протокол NetBEUI (NetBIOS Extended User Interface), що дозволяє організувати дуже ефективний обмін інформацією в мережах, що складаються не більше ніж з 200 комп'ютерів.

Щоб обмін між комп'ютерами був можливий, кожен з них повинен мати логічним ім'ям. Для обміну даними між комп'ютерами використовуються логічні імена, що привласнюються комп'ютерам динамічно при їх підключенні до мережі. При цьому таблиця імен поширюється на кожен комп'ютер мережі. Підтримується також робота з груповими іменами, що дозволяє передавати дані відразу декільком адресатам.

Головні плюси протоколу NetBEUI - швидкість роботи і дуже малі

вимоги до ресурсів. Якщо потрібно організувати швидкий обмін даними в невеликої мережі, що складається з одного сегмента, кращого протоколу для цього не знайти. Крім того, для доставки повідомлень встановлене з'єднання не є обов'язковою вимогою: в разі відсутності з'єднання протокол використовує Датаграмний метод, коли повідомлення забезпечується адресою одержувача та відправника і "пускається в дорогу", переходячи від одного комп'ютера до іншого [5.1, 5.18].

Однак NetBEUI володіє й істотним недоліком: він повністю позбавлений поняття про маршрутизації пакетів, тому його використання в складних складових мережах не має сенсу.

Що стосується протоколу SMB (Server Message Block), то з його допомогою організовується робота мережі на трьох найвищих рівнях - сеансовому, рівні уявлення і прикладному рівні. Саме при його використанні стає можливим доступ до файлів, принтерів та інших ресурсів мережі. Даний протокол кілька разів був удосконалений (вийшло три його версії), що дозволило застосовувати його навіть в таких сучасних операційних системах, як Windows 7 та Windows 10. Протокол SMB універсальний і може працювати в парі практично з будь-яким транспортним протоколом, наприклад TCP / IP і SPX.

5.13. RPC - клас технологій, віддалений виклик процедур

Віддалений виклик процедур, рідше Виклик віддалених процедур (Remote Procedure Call, RPC) - клас технологій, що дозволяють комп'ютерним програмам викликати функції або процедури в іншому адресному просторі (на віддалених комп'ютерах, або в незалежній сторонньої системі на тому самому пристрої) [5.1, 5.32-34].

Зазвичай реалізація RPC-технології включає в себе два компоненти: мережевий протокол для обміну в режимі клієнт-сервер і мова серіалізації

об'єктів (або структур, для необ'єктних RPC). Різні реалізації RPC мають дуже відрізняється один від одного архітектуру і різняться в своїх можливостях: одні реалізують архітектуру SOA, інші - CORBA або DCOM. На транспортному рівні RPC використовують в основному протоколи TCP і UDP, однак, деякі побудовані на основі HTTP (що порушує архітектуру ISO / OSI, так як HTTP - спочатку не транспортний протокол).

Існує безліч технологій, що забезпечують RPC:

• DCE / RPC - Distributed Computing Environment / Remote Procedure Calls (бінарний протокол на базі різних транспортних протоколів, в тому числі TCP / IP і Named Pipes з протоколу SMB / CIFS)

• DCOM - Distributed Component Object Model, відомий як MSRPC Microsoft Remote Procedure Call або «Network OLE» (об'єктно-орієнтоване розширення DCE RPC, що дозволяє передавати посилання на об'єкти і викликати методи об'єктів через такі посилання)

• ZeroC ICE

• JSON-RPC- JavaScript Object Notation Remote Procedure Calls (текстовий протокол на базі HTTP) см. Специфікацію: RFC-4627

• .NET Remoting (бінарний протокол на базі TCP, UDP, HTTP)

• Java RMI - Java Remote Method Invocation - див. Специфікацію: http://java.sun.com/j2se/1.5.0/docs/guide/rmi/index.html

• SOAP - Simple Object Access Protocol (текстовий протокол на базі HTTP) (Специфікація: RFC-4227)

• Sun RPC (бінарний протокол на базі TCP і UDP і XDR) RFC -1831 друга назва - ONC RPC RFC -1833

• XML RPC (текстовий протокол на базі HTTP) див. Специфікацію: RFC-3529

• Routix.RPC

5.14. DCE / RPC - система віддаленого виклику процедур

DCE / RPC (Distributed Computing Environment / Remote Procedure Calls - розподілена обчислювальна середовище / видалені виклики процедур) - система віддаленого виклику процедур, розроблена для Distributed Computing Environment (DCE) [5.1, 5.34].

Ця система дозволяє програмістам займатися розробкою розподіленого програмного забезпечення, як ніби це все працює на тому ж комп'ютері, без необхідності хвилюватися за код, який відповідає за роботу з мережею.

DCE / RPC був описаний Open Software Foundation в "Request for Technology". Однією з ключових компаній, які внесли вклад в систему, була Apollo Computer, яка привнесла "Network Computing Architecture", що стала Network Computing System (NCS), а потім великою частиною самої DCE / RPC.

Раніше джерело DCE був доступний тільки під власною ліцензією. Станом на 12 січня 2005 року, він був доступний згідно з визнаною ліцензією з відкритим вихідним кодом, яка дозволяє більш широкому колу людей працювати над джерелом для розширення своїх можливостей і зберігати його в актуальному стані. Джерело може бути завантажений через Інтернет. Видання містить приблизно 100 файлів ".tar.gz", які займають 170 мегабайт [5.35].

The Open Group заявила, що буде працювати з спільнотою DCE для того, щоб зробити DCE доступним для спільноти розробників з відкритим вихідним кодом, а також продовжуючи пропонувати джерело через веб-сайт The Open Group.

Еталонна реалізація DCE / RPC (версія 1.1) раніше була доступна під BSD-сумісної (Free Software) ліцензією OSF / 1.0 і як і раніше доступна принаймні в Solaris, AIX і VMS.

DCE також як і раніше доступний в колишніх умовах без ліцензії з відкритим вихідним кодом на сайті The Open Group.

DCE / RPC використовувалася в Національній системі страхового

забезпечення в Великобританії. В даний час використовується

• інформаційним студентським порталом Університету штату Пенсільванія (США);

• старої версій HP OpenView Operations для Unix / Windows агентів;

• Microsoft Exchange / Outlook (MAPI / RPC).

Альтернативні версії і реалізації

• FreeDCE - реалізація DCE 1.1 перенесена на Linux, підтримує 64-бітові платформи і використовує autoconf для спрощення портування в інші платформи. Порт на Win32 розробляється.

Entegrity Solutions ліцензували в OSF весь код DCE 1.2.2 і перенесли його на Win32, створивши продукт названий PC/DCE – (див. http://support.entegrity.com/private/pcdce32.asp).

• Версії DCE / RPC від Microsoft, яка називається "MSRPC", інтегрована в Windows NT. MSRPC запозичена з реалізації DCE 1.1.

• Samba містить реалізацію MSRPC, яка повинна бути мережного-сумісної і IDL-сумісної з MSRPC. Вона не є бінарному-сумісної з MSRPC.

• Wine містить реалізацію MSRPC що має намір бути бінарному і IDL-сумісної з MSRPC, але не є мережевий сумісної з MSRPC.

• J-Interop - робоча реалізація MSRPC на Java.

• Jar Apac - DCE / RPC на Java.

Розподілені додатки надають абсолютно нові рішення в проектуванні і розвитку.

Деякі додатки є розподіленими спочатку: багато користувачів ігри, додатки для обміну думками, для телеконференцій - все це приклади подібних додатків. Для них переваги стійкої інфраструктури очевидні.

Багато інших додатків також є розподіленими в тому сенсі, що вони мають як мінімум два компоненти, що працюють на різних машинах. Але, оскільки ці програми не були створені для використання в розподіленої середовищі, вони досить обмежені в масштабованості і в гнучкості

перерозподілу.

Будь-який тип поточних або групових додатків, більшість додатків клієнт/сервер і навіть деякі desktop-додатки обов'язково керують способом комунікацій і кооперації своїх користувачів.

Розгляд таких додатків як розподілених і робота з необхідним компонентом в потрібному місці надають користувачеві переваги і оптимізують використання мережевих і комп'ютерних ресурсів.

Додатки, які розроблялися як розподілені, можуть поєднувати різних клієнтів з різними потужностями за допомогою роботи компонента з боку клієнта, якщо це можливо, і - з боку сервера, коли це необхідно.

Розробка розподілених додатків дає системному менеджеру велика перевага у вигляді гнучкості в перерозподілі.

До того ж, розподілені додатки є набагато більш масштабованими, ніж їх монолітні побратими. Якщо вся логіка комплексного додатка зосереджена в єдиному модулі, є єдиний спосіб прискорити роботу без установки програми: більш швидкісний апаратне забезпечення.

5.15. Розподілені додатки DCOM

Сьогоднішні сервери і операційні системи легко модифікуються, однак частіше буває дешевше придбати ще одну таку ж машину, ніж зробити upgrade, щоб прискорити сервер вдвічі [5.1, 5.20].

При правильно сконструйованому розподіленому додатку на початку роботи все компоненти можуть запуститися з одного сервера. При збільшенні завантаження деякі компоненти можуть перерозподілятися на додаткові, менш дорогі машини.

Розподілені додатки надають вам абсолютно нові рішення в проектуванні і розвитку. Для того, щоб отримати ці додаткові можливості, необхідно зробити значні вкладення.

Деякі додатки є розподіленими спочатку: багато користувачів ігри, додатки для обміну DCOM (Distributed Component Object Model) є розвитком багатокомпонентної моделі (COM). COM визначає, яким чином компоненти взаємодіють зі своїми клієнтами. Ця взаємодія здійснюється таким чином, щоб клієнт і компонент могли з'єднуватися без необхідності використовувати певний проміжний компонент системи і клієнт міг викликати методи компонента. Рис. 1 ілюструє це з точки зору багатокомпонентної моделі (рис. 5.8).

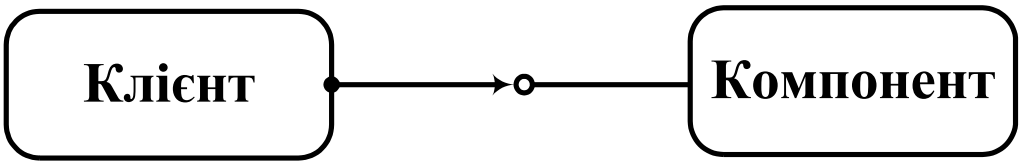


Рис. 5.8. COM-компоненти в одному процесі

В сьогоднішніх операційних системах процеси ізольовані один від одного. Клієнт, якому потрібно зв'язатися з компонентом іншого об'єкта, не може викликати компонент безпосередньо, а повинен використовувати деяку форму зв'язку між процесами, передбачену операційною системою. COM організовує подібне з'єднання в повністю прозорою манері: він перехоплює виклики з боку клієнта і адресує їх компоненту іншого процесу. Рис. 5.9 показує, як run-time бібліотеки COM / DCOM організовують з'єднання між клієнтом і компонентом [5.1, 5.20].

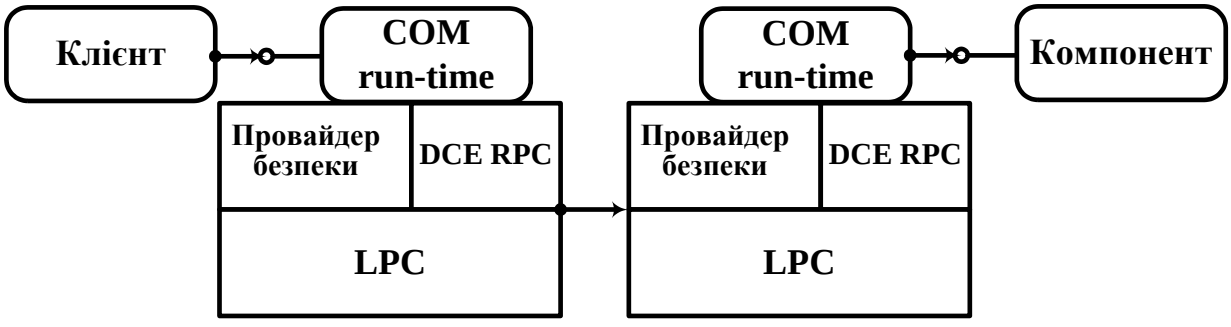


Рис. 5.9. COM-компоненти в різних процесах

Коли клієнт і компонент знаходяться на різних машинах, DCOM просто замінює локальне з'єднання між процесами мережевим протоколом. При цьому ні клієнт, ні компонент і не підозрюють, що дроти, що сполучають їх, стали трохи довше.

Рис. 5.10 показує архітектуру DCOM в загальному: COM run-time пропонує клієнтам і компонентів об'єктно-орієнтовані сервіси і використовує RPC і провайдер безпеки для генерації стандартних мережевих пакетів, що відповідають стандарту протоколу DCOM.

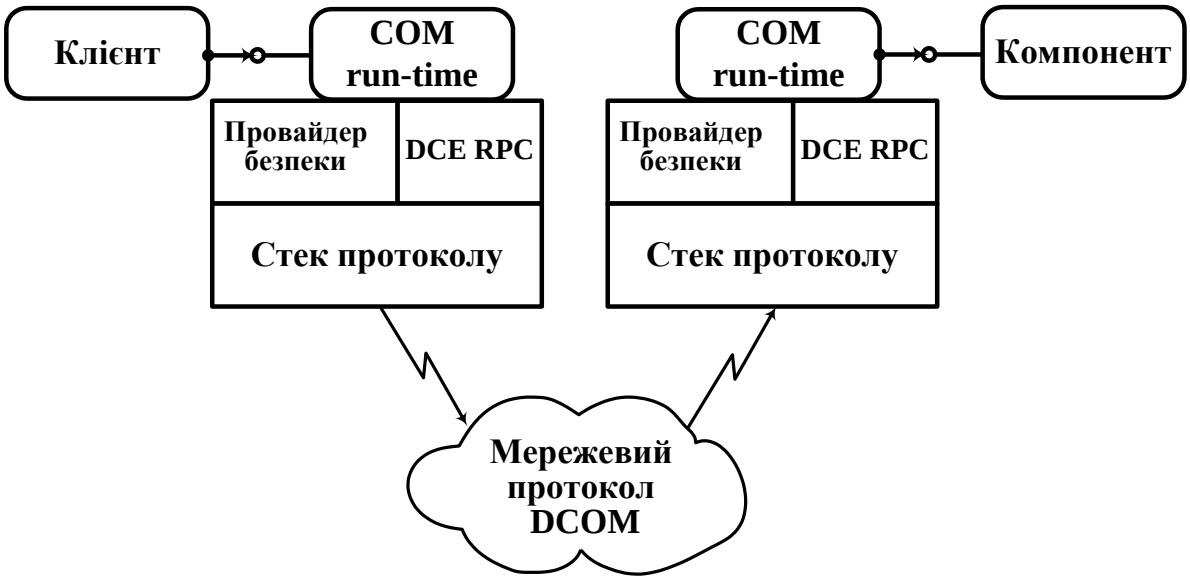


Рис. 5.10. DCOM: COM-компоненти на різних машинах

Велика частина розподілених додатків розробляється не наспіх і не в вакуумі.

Існуюча інфраструктура апаратного забезпечення, існуюче програмне забезпечення, існуючі компоненти і інструменти повинні бути інтегровані і покликані зменшити час і вартість розробки і перерозподілу.

У сенсі інвестицій DCOM має безсумнівні переваги перед будь-якими COM компонентами і інструментами.

Величезний ринок компонентів дозволяє зменшити час розробки за допомогою інтегрування стандартизованих рішень в користувальницький додаток.

Багато розробників знайомі з COM і можуть легко застосувати свої знання в створенні розподілених DCOM-додатків [5.1, 5.20].

Коли ви починаєте використовувати розподілене додаток в реальному мережі, виявляється кілька особливостей:

• Компоненти, які взаємодіють частіше, повинні бути "ближче" один до

одного.

• Деякі компоненти можуть працювати на певних машинах або в певних місцях.

• Використання менших за розміром компонентів збільшує гнучкість в перерозподілі, але при цьому збільшується мережевий трафік.

• Компоненти більшого розміру зменшують мережевий трафік, але зате зменшується і гнучкість в перерозподілі.

За допомогою DCOM ці критичні моменти обходяться досить легко, оскільки в вихідному коді не визначені деталі перерозподілу. DCOM повністю приховує місце розташування компонента, будь він у тому ж процесі, що і клієнт, або на іншому кінці світу. У будь-якому випадку способи, якими клієнт з'єднується з компонентом і викликає методи компонента, ідентичні. DCOM не потребує не тільки в змінах вихідного коду, але навіть і в рекомпіляції програми. Проста реконфігурація змінює спосіб, яким компоненти з'єднуються один з одним.

Незалежність DCOM від місця розташування значно спрощує завдання розподілу компонентів додатка для отримання оптимального швидкодії в цілому. Уявімо, наприклад, що певні компоненти повинні перебувати на певних машинах або в певних місцях. Якщо додаток має велике число маленьких компонентів, ви можете зменшити завантаження мережі переміщенням їх в потрібний сегмент ЛОМ, на потрібну машину або навіть в потрібний процес. Якщо програма містить меншого числа великих компонентів, завантаження мережі - менша з проблем, так як ви можете розмістити компоненти на більш швидкі з доступних машин.

Рис. 5.11 демонструє, як один і той же компонент може бути перенесений на машину клієнта при задовільною смузі пропускання між машиною "клієнт" і машиною "середнього шару", і на машині сервера, якщо клієнт працює з додатком по повільному мережевому з'єднанні.

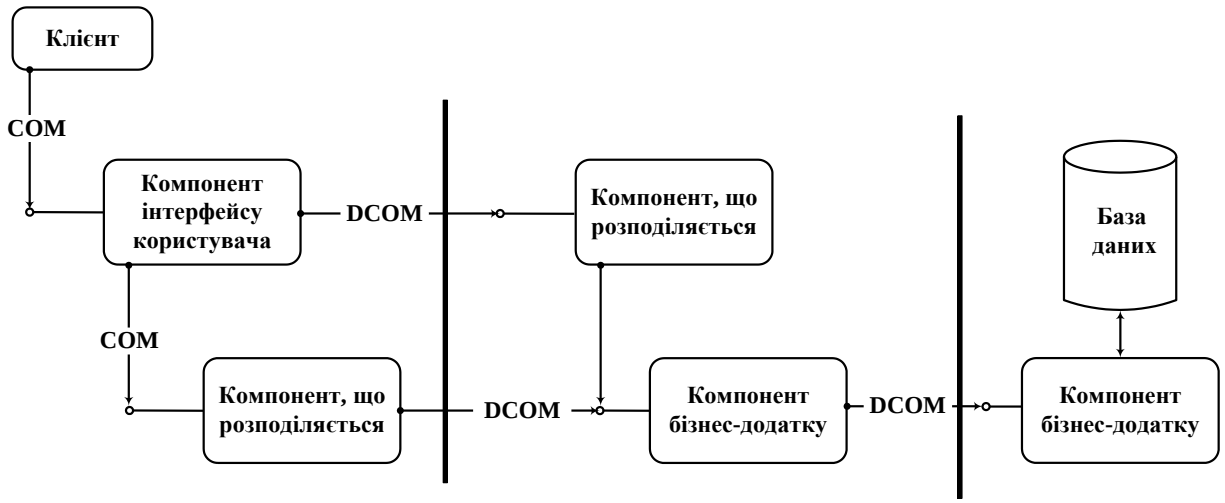


Рис. 5.11. Розподіл компонентів додатка для отримання оптимального швидкодії

Завдяки незалежності DCOM від місця розташування, додаток може переміщати взаємодіючі компоненти з "довколишніх" машин на одну і ту ж машину або навіть в один процес.

Навіть якщо функціональність великого логічного модуля організовується великим числом маленьких компонентів, вони можуть як і раніше ефективно взаємодіяти один з одним.

Компоненти можуть працювати на машині, на якій це найбільш доцільно: призначений для користувача інтерфейс знаходиться на машині клієнта або "поруч" з нею. Компонент, що працює з базою даних, - на сервері "близько" до бази даних.

Мережеві з'єднання не так стійкі, як з'єднання всередині машини. Компоненти розподіленого додатка повинні знати, що клієнт більш не активний, навіть - чи особливо - в разі мережевої або апаратної аварії.

DCOM управляє з'єднанням компонентів, призначених для одного клієнта, так само, як і компонентів, які обслуговують кілька клієнтів, використовуючи лічильник посилань для кожного компонента. Коли клієнт з'єднується з компонентом, DCOM збільшує значення лічильника посилань компонента. Коли клієнт розриває з'єднання, DCOM зменшує значення лічильника посилань компонента. Якщо значення лічильника досягає нуля - компонент вільний [5.1, 5.20].

Для визначення активності клієнта DCOM використовує ефективний протокол відстеження. Машина клієнта посилає періодичне повідомлення. При порушенні з'єднання DCOM зменшує лічильник і звільняє компонент, якщо значення лічильника стане рівним нулю. З точки зору компонента, випадок відключення клієнта, випадок аварії мережі і випадок поломки машини клієнта реєструються одним і тим же механізмом підрахунку. Додатки можуть використовувати такий механізм підрахунку з'єднань для свого вивільнення.

У багатьох випадках потік інформації між компонентом і його клієнти не однонаправлені: компоненту потрібно ініціювати деякі дії з боку клієнта, наприклад, повідомлення про те, що тривалий процес завершився, оновлюються дані, які він переглядав користувачем (огляд новин), або надійшло наступне повідомлення в телеконференції або розрахованої на багато користувачів грі.

Багато протоколи ускладнюють процес здійснення такого типу симетричною зв'язку. В DCOM кожен компонент може бути одночасно провайдером і споживачем функціональності. Один і той же механізм, одні і ті ж можливості керують зв'язком в обох напрямках, полегшуючи організацію зв'язку і взаємодії клієнт / сервер.

DCOM пропонує стійкий розподілений механізм реєстрації з'єднань, який повністю прозорий для програми. DCOM - це одночасно симетричний мережевий протокол і модель програмування, що пропонують не тільки традиційне взаємодія клієнт / сервер, але і повноцінну зв'язок між клієнтами і серверами.

Багато протоколи ускладнюють процес здійснення такого типу симетричною зв'язку. В DCOM кожен компонент може бути одночасно провайдером і споживачем функціональності. Один і той же механізм, одні і ті ж можливості керують зв'язком в обох напрямках, полегшуючи організацію зв'язку і взаємодії клієнт / сервер.

DCOM використовує можливість Windows підтримувати симетричну багатопроцесорну обробку. Для додатків, що використовують модель вільних

потоків, DCOM організовує об'єднання потоків вхідних запитів. На багатопроцесорних машинах це об'єднання потоків оптимізується відповідно до числа доступних процесорів: надмірно великі потоки призводять до занадто частого перемикання між вмістом, в той час як занадто маленькі потоки можуть привести до простою деяких процесорів. DCOM захищає розробника від деталей управління потоками і забезпечує оптимальне швидкодію, що може забезпечити лише кодування управління об'єднанням потоків вручну, що вимагає великих витрат.

Використовуючи підтримку симетричною багатопроцесорної обробки Windows, DCOM-додатки можуть безболісно масштабуватися від рівня однопроцесорних машин до рівня величезних багатопроцесорних систем.

Перерозподіл найпростіше виконується для компонентів, які не повинні знаходитися в певному місці або ділити своє місцезнаходження з іншими компонентами. В цьому випадку є можливість роботи численних копій компонента на різних машинах.

Найчастіше навантаження може бути розділена між машинами з урахуванням таких критеріїв, як ємність машини або навіть поточне завантаження.

DCOM полегшує зміну способу з'єднання клієнтів з компонентами і компонентів між собою. Одні і ті ж компоненти можуть бути динамічно перерозподілені без виконання повторної роботи або навіть рекомпіляції. Все, що необхідно - це оновити реєстрацію, файлову систему або базу даних, в яких занесені місця розташування кожного компонента (рис. 5.12).

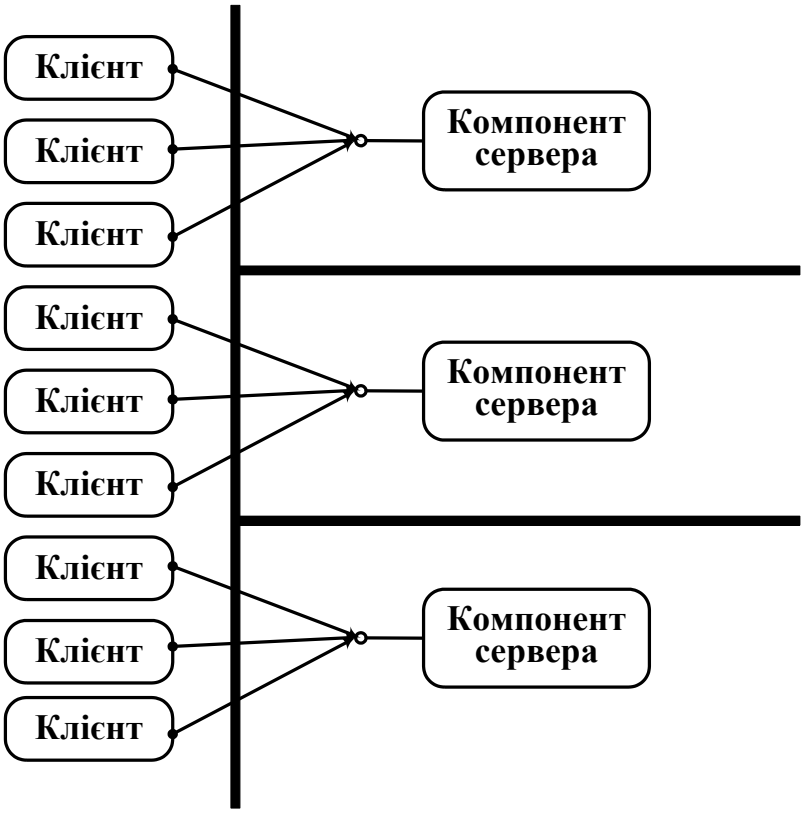


Рис. 5.12. Паралельний перерозподіл

Велика частина реальних розподілених додатків мають один або більше критичний компонент, який бере участь в більшості операцій.

Це можуть бути компоненти бази даних або компоненти, пов'язані з бізнесом, доступ до яких має здійснюватися постійно для реалізації політики "першим прийшов - першим обслужений" [5.1, 5.20].

Компоненти такого типу не можуть дублюватися, оскільки їхнє призначення полягає в організації єдиної точки синхронізації серед всіх користувачів програми. Для збільшення швидкодії розподіленого додатка в цілому такі компоненти, що створюють "вузькі місця", повинні перерозподілятися на спеціально виділений потужний сервер.

DCOM допомагає вам ізолювати такі критичні компоненти на ранніх етапах проектування, розміщувати спочатку компоненти на одній машині, а пізніше - переносити критичні компоненти на окремі машини (рис. 5.13).

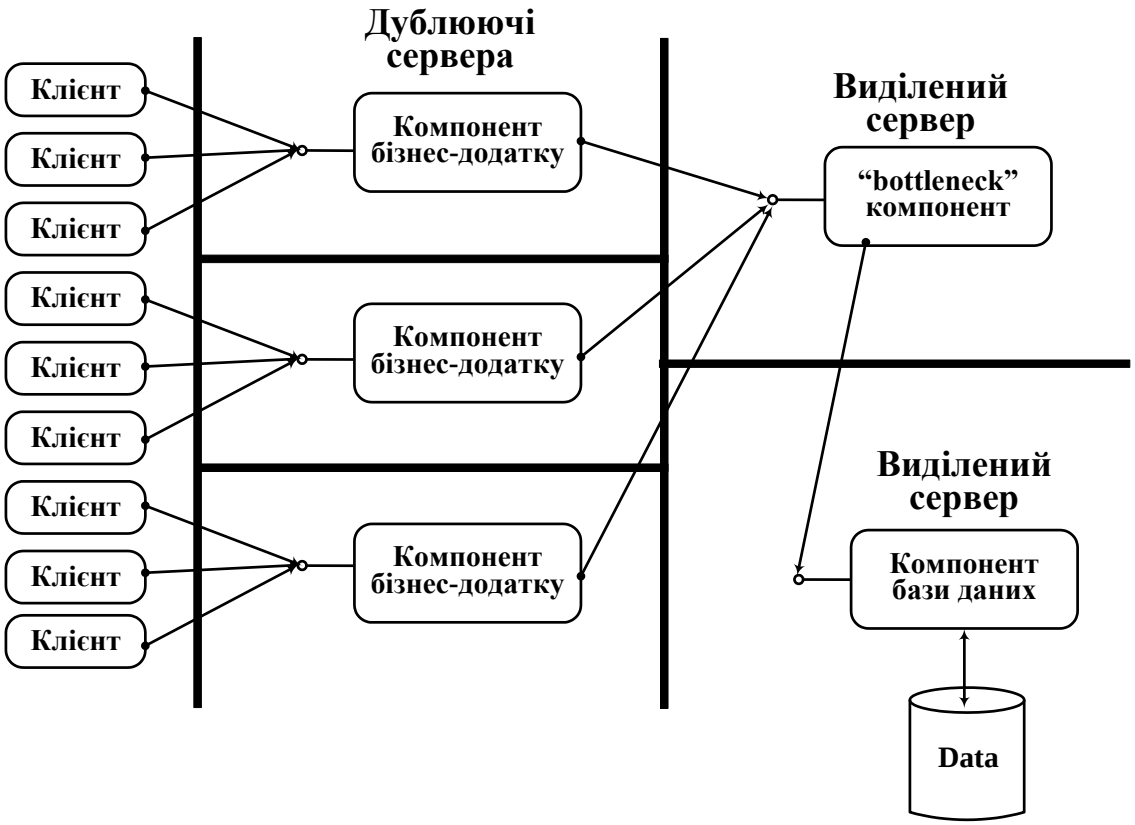


Рис. 5.13. Ізоляція критичних компонентів

Для таких критичних компонентів DCOM в цілому може організувати більш швидке виконання завдання.

Подібні компоненти - це зазвичай частина послідовності процесу організації в електронній торговій системі замовлень на покупку або продаж: запити повинні оброблятися в міру надходження (першим прийшов - першим обслужений). Одне з рішень полягає в поділі завдання на менші компоненти з перерозподілом кожного компонента на окрему машину.

Результат цього подібний поточному методу (pipelining), використовуваному в сучасних мікропроцесорах: перший запит обробляється першим компонентом (наприклад, виконують перевірку вмісту) і передається на наступний компонент (який, наприклад, виконує оновлення бази даних).

Як тільки перший компонент передав запит на наступний, він готовий обробити такий запит.

Фактично, дві машини паралельно обробляють безліч запитів в певному порядку їх надходження.

Те ж саме можна виконувати і на одній машині (при використанні DCOM): численні компоненти можуть виконуватися в різних потоках або процесах.

Надалі, коли потоки зможуть бути розподілені на одній многопроцессорной машині або на різні машини, цей підхід полегшить масштабованість.

Модель програмування DCOM з її незалежністю від місця розташування полегшує схеми перерозподілу в міру розростання додатки: спочатку машина сервера може містити всі компоненти, з'єднуючись з ними як з дуже ефективними серверами в процесі.

Фактично додаток виглядає як добре налагоджене монолітне додаток.

У міру збільшення потреб можна додати інші машини з перерозподілом компонентів на ці машини без всякого зміни коду.

Оскільки функціональність в DCOM-моделі програмування згрупована в інтерфейси, ви можете проектувати нові клієнтські програми для роботи зі старими серверами, нові сервери - для роботи зі старими клієнтами або комбінувати ці можливості для задоволення ваших запитів [5.1, 5.20].

У звичайних моделях об'єктів навіть найменша зміна методу серйозно позначається на взаємодії між клієнтом і компонентом (рис. 5.14).

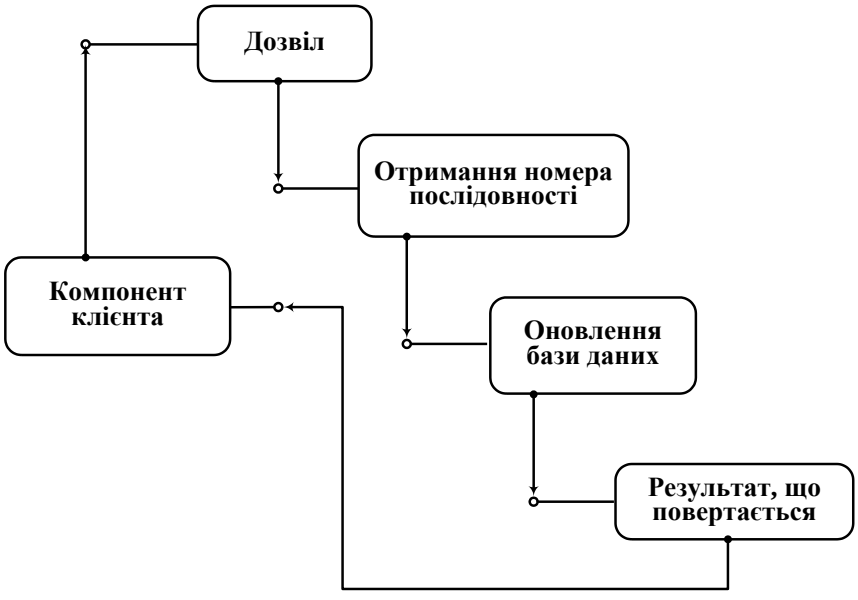


Рис. 5.14. Поточний метод

Деякі моделі дозволяють додавати нові методи в кінець списку методів, але при цьому немає можливості безпечної перевірки взаємодії нових методів зі старими компонентами. Розгляд цього питання в мережевому аспекті вносить ще більші ускладнення, адже зазвичай кодування і наявність провідного зв'язку впливає на набір методів і параметрів. Додавання або зміна методів і параметрів значно змінює і мережевий протокол. DCOM усуває всі ці проблеми єдиним, витонченим, уніфікованим рішенням і для об'єктної моделі, і для мережевого протоколу (рис. 5.15.).

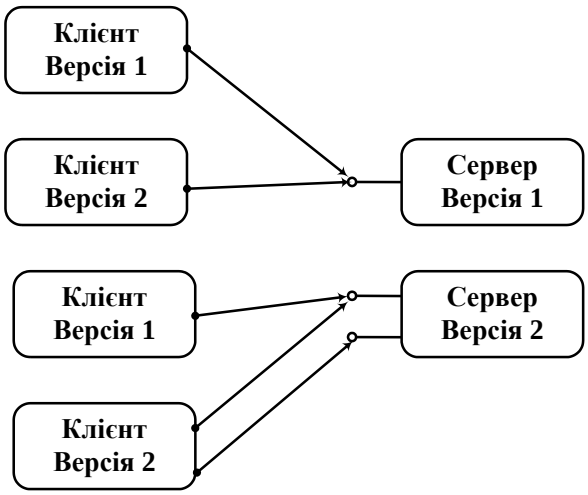


Рис. 5.15. Стійкий контроль версій

Наскільки швидкий цей механізм RPC? Для визначення цього є різні параметри швидкодії:

• Наскільки швидкий виклик порожнього методу?

• Наскільки швидкі реальні виклики методів, що посилають і повертають дані?

• Наскільки швидка робота по мережі?

Переваги DCOM в масштабованості і швидкодії в цілому можуть бути отримані тільки при використанні добре організованого управління об'єднанням потоків і тестових протоколів (pinging protocols). Велика частина розподілених додатків навіть при значних вкладеннях не отримають приблизно такого ж збільшення швидкодії, як при використанні стандартизованих DCOM-протоколу і моделі програмування.

Багато протоколи потребують постійного управлінні. Компонентів

потрібно знати, що на машині клієнта виникла серйозна аварія апаратного забезпечення або мережеве з'єднання між клієнтом і компонентом перервалося на тривалий час. Спільним рішенням цієї проблеми є постійна пересилання повідомлень з певною періодичністю (pinging). Якщо сервер не отримує повідомлення протягом певного часу, це означає, що зв'язки з клієнтом немає.

DCOM використовує такі повідомлення для кожної з машин. Навіть якщо машина клієнта використовує 100 компонентів з машини сервера, єдине тестове повідомлення підтримує зв'язок всіх клієнтів.

DCOM дозволяє різним програмам (навіть з різних джерел) використовувати єдине, оптимізоване, постійне управління і протокол визначення аварій мережі, незначно зменшуючи смугу пропускання (рис. 5.16).

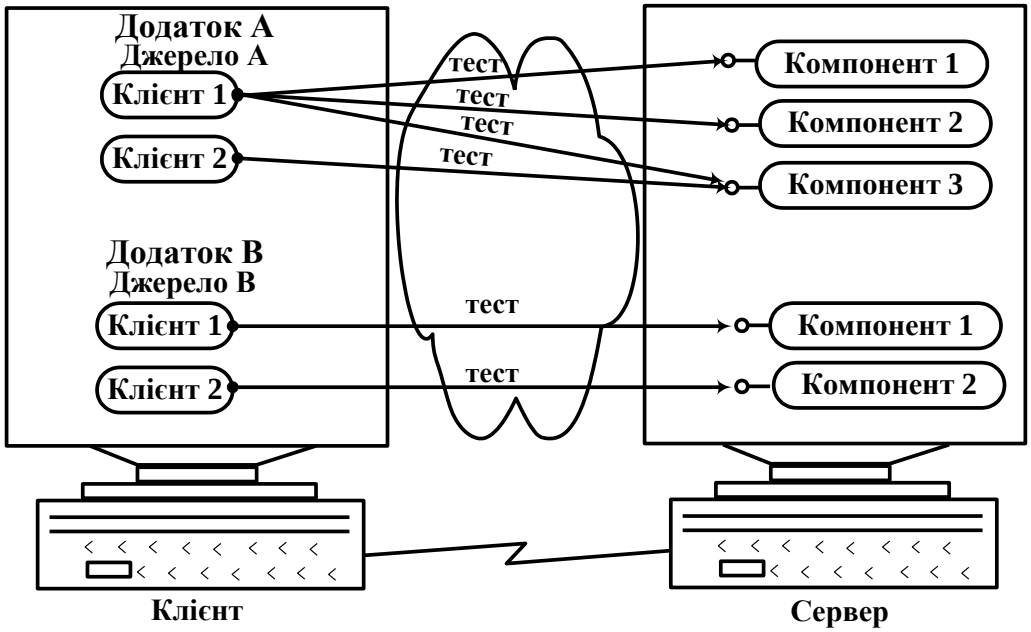


Рис. 5.65. Поєднане постійне управління

DCOM полегшує дизайнерам компонентів виконання подібного пакетування без того, щоб клієнтам потрібно було використовувати API пакетування. Механізм маршалинга DCOM дозволяє компоненту передавати код, званий "заступником об'єкта" (proxy object) і дозволяє перехоплювати численні виклики методів і упаковувати їх в єдиний виклик віддаленої процедури клієнту:

Приклад. Розробник з попереднього прикладу продовжує

перераховувати методи один за іншим, оскільки це спосіб, який необхідний логіці програми. (Це потрібно списку API). Однак, перший виклик для початку перерахування пересилається в заступник об'єкта, визначений додатком, який викликає все рядки (або набір рядків) і кеширует їх в об'єкт-заступник. Потім наступні виклики виходять з цього кеша без додаткових мережевих обмінів. Розробник продовжує працювати з простою моделлю програмування, хоча в цілому додаток вже оптимізовано (рис. 5.17).

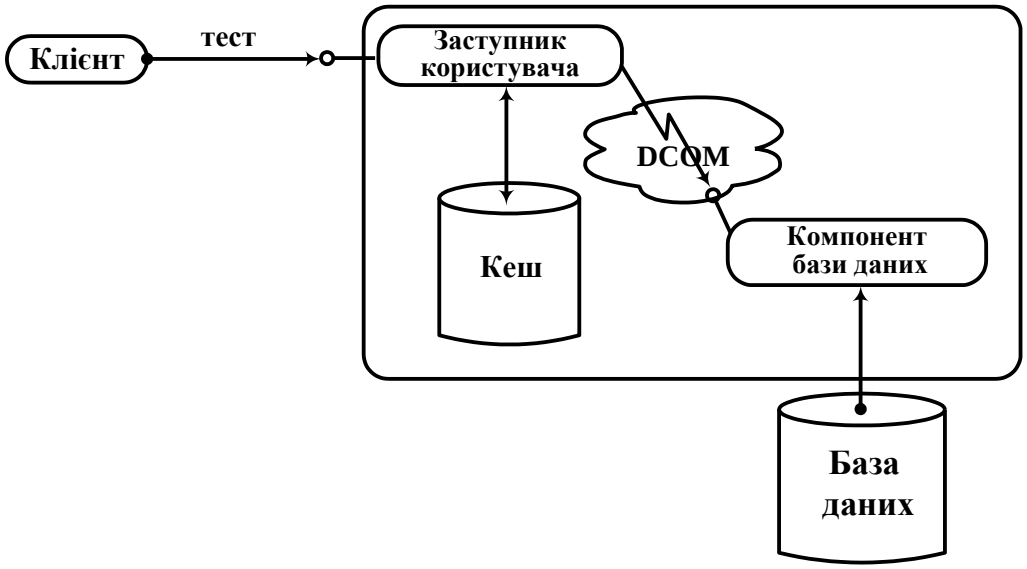


Рис. 5.17. Компонентна модель: кешування з боку клієнта

Крім того, DCOM дозволяє виконувати ефективну переадресацію з одного компонента на інший. Якщо компонент містить посилання на інший компонент другої машини, він може передати це посилання клієнту, який працює на третій машині (посилання клієнта на інший компонент, який працює на іншій машині). Коли клієнт використовує це посилання, він безпосередньо з'єднується з другим компонентом. DCOM стикується ці посилання і дозволяє оригіналу компонента і машині вийти з цієї схеми взаємодії. Це надає користувачеві сервіси директорій, що дозволяють повертати посилання на широкий діапазон віддалених компонентів.

Приклад 1. Додаток "шахи" дозволяє гравцям, які шукають партнера, реєструватися в сервісі каталогу шахів. Інші гравці можуть запросити цей список або встати в чергу. При виборі гравцем партнера сервіс каталогу шахів

повертає посилання на компонент клієнта партнера. DCOM автоматично з'єднує двох гравців; в подальших транзакціях сервіс каталогу більш не бере.

Приклад 2. Компонент "Брокер" має з'єднання з 20 машинами з серверами, на яких працюють ідентичні компоненти. Відбувається постійна перевірка завантаження серверів і визначення того, чи змінилася кількість серверів. Коли клієнт потребує компоненті, він підключається до компоненту "брокер", який повертає посилання на компонент сервера з мінімальним завантаженням. DCOM автоматично з'єднує клієнта з сервером; після цього компонент "брокер" випадає зі схеми взаємодії [5.1, 5.20, 5.35].

Якщо необхідно, DCOM навіть дозволяє компонентам включатися в довільні призначені для користувача протоколи, використання яких має на увазі роботу поза механізму DCOM. Компонент може використовувати призначений для користувача маршалинга для пересилання заступника об'єкта в процес клієнта, який в подальшому може використовувати будь-який довільний протокол для спілкування з компонентом (рис. 5.18).

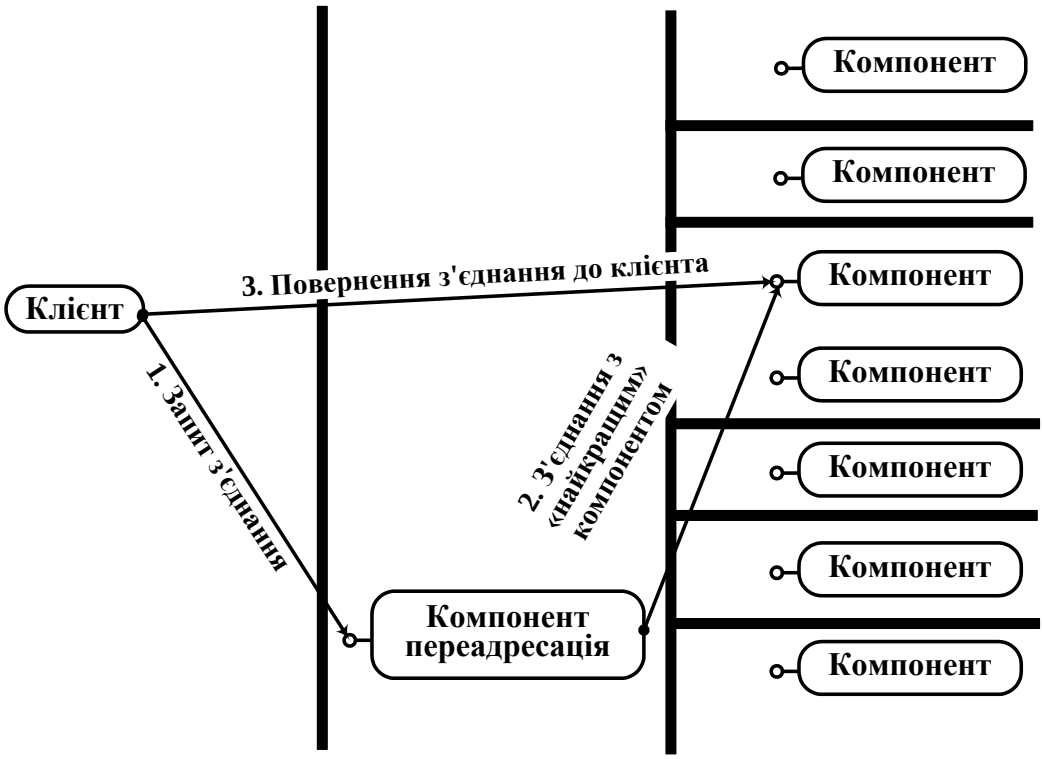


Рис. 5.18. Включення в довільні призначені для користувача протоколи

DCOM надає безліч способів поєднати реальний мережевий протокол з

мережевим трафіком без зміни взаємодії клієнта з компонентом: кешування з боку клієнта, переадресація посилань і заміна мережевого транспорту при необхідності - лише частина можливих способів (рис. 5.19).

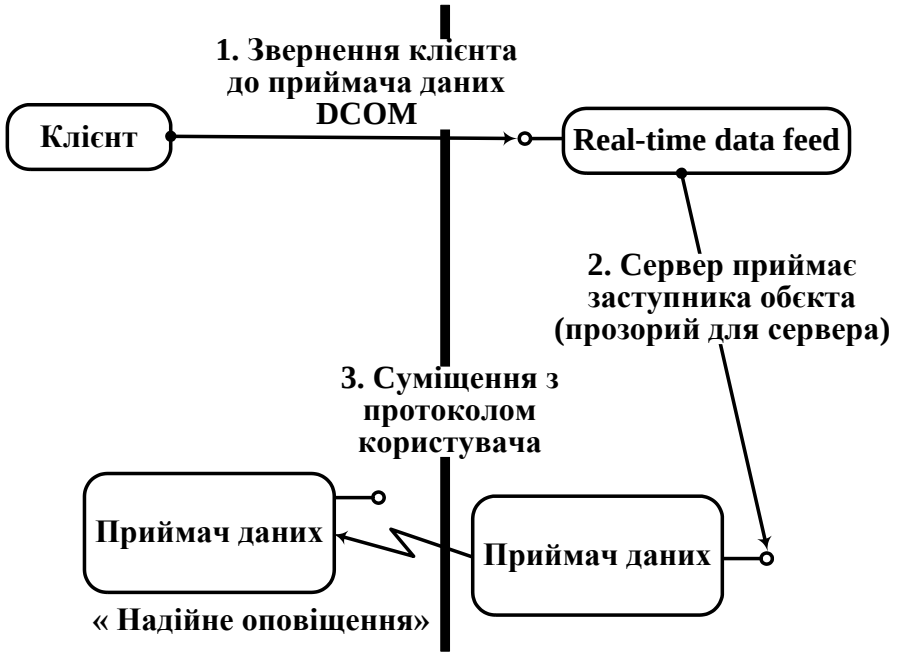


Рис. 5.19. Спосібв поєднати реальний мережевий протокол з мережевим трафіком

DCOM може забезпечити безпеку роздільного додатком без вбудовування спеціалізованого кодування безпеки в клієнт або компонент. Модель програмування DCOM приховує потребу в безпеці з боку компонента точно так же, як приховує і його місцезнаходження. Той же двійковий код, який працює на одній машині, де безпека - не головне вимога, може використовуватися і в розподіленому додатку із забезпеченням захищеності.

DCOM досягає подібної прозорості в безпеці, дозволяючи розробникам і адміністраторам конфігурувати установки захищеності для кожного компонента. Точно так же, як файлова система Windows дозволяє адміністратору встановити списки управління доступом (ACL) для файлів і каталогів, DCOM містить списки управління доступом для компонентів. Ці списки показують, які користувачі або групи користувачів мають право доступу до компонентів певного класу. Ці списки можуть легко конфігуруватися інструментом конфігурації DCOM (DCOMCNFG) або програмуватися за допомогою реєстру Windows

Коли б клієнт ні викликав метод або створив приклад компонента, DCOM отримує поточний призначене для користувача ім'я клієнта, пов'язаного з поточним процесом (насправді - поточний виконуваний потік). Windows гарантує, що даний користувальницький ідентифікатор автентичний.

Після цього DCOM передає ім'я користувача машині або процесу, де працює цей компонент. На машині компонента DCOM знову перевіряє ім'я користувача, використовуючи аутентифікаційний механізм, і перевіряє список управління доступом для даного компонента (насправді - для першого компонента, що працює в процесі, що містить цікавить компонент.

DCOM пропонує надзвичайно ефективний механізм забезпечення безпеки, який дозволяє розробнику створювати розподілені додатки, не побоюючись за його захищеність. Будь-провайдер безпеки, підтримуваний Windows, може використовуватися механізмом безпеки DCOM (рис. 5.20).

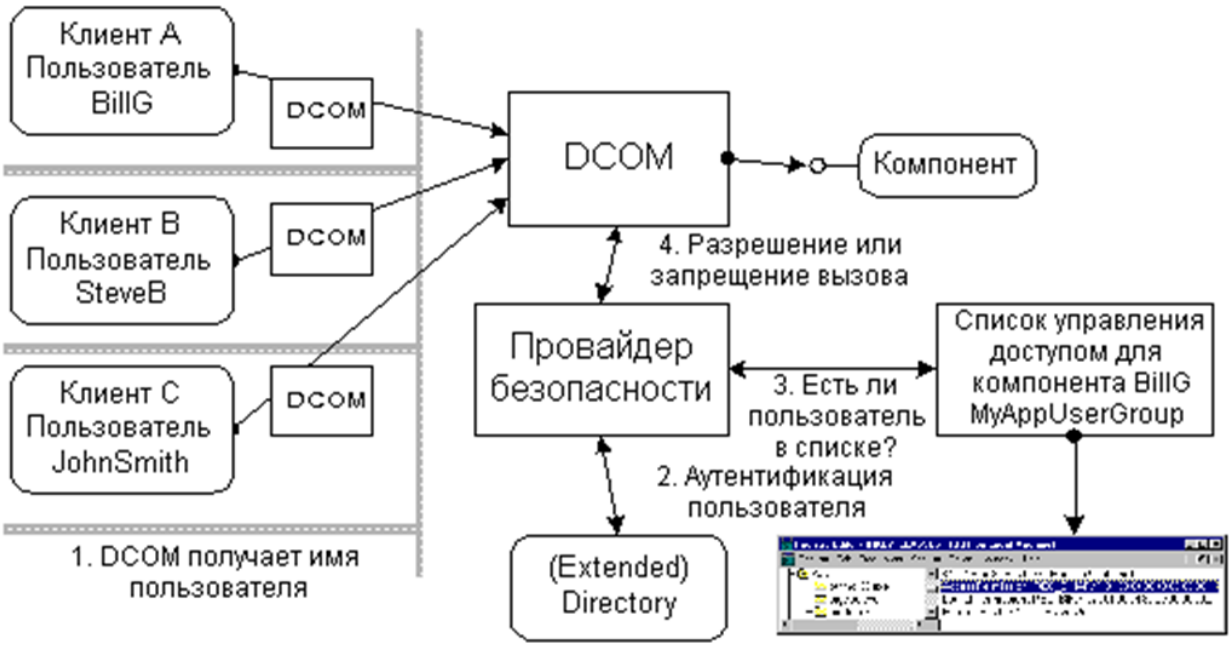


Рис. 5.20. Конфігурування безпеки

Більш детально це описано в Білих Сторінках "Архітектура DCOM". Якщо ім'я клієнта не включено в цей список (прямо або побічно - в якості члена групи користувачів), DCOM просто відхиляє цей виклик ще до того, як компонент залучився в роботу. Цей звичайний механізм безпеки повністю прозорий як для клієнта, так і для компонента і оптимізований. Він базується на

framework безпеки Windows, який є найбільш інтенсивно використовуваної (і оптимізованої). Частиною операційної системи Windows: при кожному доступі до файлу або навіть до такого примітиву синхронізації потоків, як сигналізація або подія, Windows виконує ідентичну перевірку доступу. Той факт, що Windows як і раніше може непогано конкурувати в швидкодії з іншими операційними системами і мережевими операційними системами, показує, наскільки ефективний цей механізм безпеки [5.1, 5.20].

• Транспортна підсистема:

- управління вихідними і вхідними з'єднаннями.

- підтримка поняття «межа повідомлення» для транспортних протоколів, які не підтримують його безпосередньо (TCP).

- підтримка гарантованої доставки для транспортних протоколів, які не підтримують її безпосередньо (UDP).

• Пул потоків (тільки для викликається сторони). Надає контекст виконання для викликаного по мережі коду.

• маршалінга (аналог «серіалізації»). Упаковка параметрів викликів в потік байт стандартним чином, не залежних від архітектури (зокрема, від порядку байт в слові). Зокрема, йому можуть піддаватися масиви, рядки і структури, на які вказують параметри-покажчики.

• Шифрування пакетів і накладення на них цифрового підпису.

• Аутентифікація і авторизація. Передача по мережі інформації, що ідентифікує суб'єкт, який здійснює виклик.

У деяких реалізаціях RPC (.NET Remoting) кордону підсистем є відкритими поліморфними інтерфейсами, і можливо написати свою реалізацію майже всіх перерахованих підсистем. В інших реалізаціях (DCE RPC в Windows) це не так.

5.16. SOCKS Secure- протокол з використанням сервісів за

міжмережевими екранами

SOCKS («SOCKet Secure») - мережевий протокол, який дозволяє пересилати пакети від клієнта до сервера через проксі-сервер прозоро (непомітно для них) і таким чином використовувати сервіси за міжмережевими екранами (фаєрвол) [5.1, 5.2].

Пізніша версія SOCKS5 передбачає аутентифікацію, так що тільки авторизовані користувачі отримують доступ до сервера.

Клієнти за фаєрволом, які потребують доступі до зовнішніх серверів, замість цього можуть бути з'єднані з SOCKS-проксі-сервером. Такий проксі-сервер контролює права клієнта на доступ до зовнішніх ресурсів і передає клієнтський запит зовнішнього сервера. SOCKS може використовуватися і протилежним способом, здійснюючи контроль прав зовнішніх клієнтів з'єднуватися з внутрішніми серверами, що знаходяться за фаєрволом (брандмауером).

На відміну від HTTP-проксі-серверів, SOCKS передає всі дані від клієнта, нічого не додаючи від себе, тобто з точки зору кінцевого сервера, дані, отримані ним від SOCKS-проксі, ідентичні даними, які клієнт передав би безпосередньо, без проксінг.

SOCKS більш універсальний, він не залежить від конкретних протоколів програм (7-го рівня моделі OSI) і оперує на рівні TCP-з'єднань (4 OSI рівня). Зате HTTP-проксі кеширує дані і може більш ретельно фільтрувати вміст переданих даних. Протокол був розроблений системним адміністратором компанії MIPS Девідом Кобласом (David Koblas). Після того, як в 1992 році MIPS увійшла до складу корпорації Silicon Graphics, Коблас зробив доповідь про SOCKS на Симпозіумі з безпеки Usenix, і SOCKS став публічно доступним. Четверту версію протоколу розробив Ін-Да Лі (Ying-Da Lee) з NEC.

SOCKS-сервери зазвичай використовують порт 1080.

SOCKS 4 призначений для роботи через міжмережевий екран без

аутентифікації для додатків типу клієнт-сервер, що працюють по протоколу TCP, таких, як Telnet, FTP і таких популярних протоколів обміну інформацією, як HTTP, WAIS та Gopher. По суті, SOCKS-сервер можна розглядати як міжмережевий екран, що підтримує протокол SOCKS. Типовий запит SOCKS 4 виглядає наступним чином (таб. 5.1, 5.2):

Таблиця 5.1. Запит клієнта до SOCKS-Серверу

Розмір	Опис
1 байт	Номер версії SOCKS, 1 байт (для цієї версії повинен бути 0x04)
1 байт	Код команди: • 0x01 = установка TCP/IP з'єднання • 0x02 = призначення TCP/IP-порта (binding)
2 байта	Номер порту
4 байта	IP-адрес
n+1 байт	ID користувача. Строка змінної довжини, завершується NUL-байтом (0x00). Поле призначено для ідентифікації користувача

Таблиця 5.2. Відповідь сервера SOCKS-Клієнту

Розмір	Опис
1 байт	NUL-байт
1 байт	Код відповіді: • 0x5a = запит представлений • 0x5b = запит відхилений або помилковий • 0x5c = запит не вдався, бо не запущений *identd* (або він перебуває з сервера) • 0x5d = запит не вдався, оскільки клієнтський *identd* не може підтвердити ідентифікатор користувача в запиті
2 байта	Довільні дані, повинні бути проігноровані
4 байта	Довільні дані, повинні бути проігноровані

SOCKS 5 розширює модель SOCKS 4, додаючи до неї підтримку UDP, забезпечення універсальних схем строгої аутентифікації і розширює методи адресації, додаючи підтримку доменних імен і адрес IPv6. Початкова установка зв'язку тепер складається з наступного:

• Клієнт підключається, і посилає вітання, яке включає перелік підтримуваних методів аутентифікації

• Сервер вибирає з них один (або посилає відповідь про невдачу запиту, якщо жоден із запропонованих методів неприйнятний)

• В залежності від обраного методу, між клієнтом і сервером може пройти кілька повідомлень

• Клієнт посилає запит на з'єднання, аналогічно SOCKS 4

• Сервер відповідає, аналогічно SOCKS 4.

Таблиця 5.3. Методи аутентифікації пронумеровані в такий спосіб:

0x00	Аутентифікація не потребується
0x01	GSSAPI
0x02	Ім'я користувача/ Пароль
0x03-0x7F	Зарезервовано IANA
0x80-0xFE	Зарезервовано для методів приватного використання

Таблиця 5.4. Початкове вітання від клієнта:

Розмір	Опис
1 байт	Номер версії *SOCKS* (повинен бути 0x05 для цієї версії)
1 байт	Кількість підтримуваних методів аутентифікації
n байт	Номери методів аутентифікації, змінна довжина, 1 байт для кожного підтримуваного методу

5.17. GSSAPI - загальний програмний інтерфейс сервісів безпеки

GSSAPI (GSS, GSSAPI, Generic Security Services API, загальний програмний інтерфейс сервісів безпеки) - API для доступу до сервісів безпеки. Описано в стандарті IETF. Призначений для вирішення проблеми несумісності схожих сервісів безпеки [5.1, 5.28].

GSSAPI сам по собі не забезпечує сервісів безпеки, замість цього він забезпечує інтерфейс між додатками і реалізаціями GSSAPI (зазвичай бібліотеками). Ці бібліотеки забезпечують сумісний з GSS-API інтерфейс, дозволяючи створювати додатки, здатні працювати з різними бібліотеками

безпеки; дозволяючи замінювати бібліотеки без необхідності переписувати додатки. Відмінною особливістю додатків, реалізованих з використанням GSSAPI є використання закритих повідомлень (токенів), які приховують подробиці реалізації від вищестоящих додатків. Серверна і клієнтська частина додатків створюються таким чином, щоб взаємодіяти з допомогою токенів GSSAPI. Токени зазвичай можуть передаватися через незахищену (публічну) мережу. Після обміну сторонами (клієнтом і сервером) деякою кількістю повідомлень, бібліотека GSSAPI інформує обидві сторони взаємодії щодо встановлення безпечного контексту.

Після встановлення безпечного контексту захищаються повідомлення додатки можуть бути «загорнуті» (зашифровані) за допомогою GSSAPI для безпечної передачі між сервером і клієнтом. Типові аспекти безпеки, що забезпечуються бібліотеками, що реалізують GSSAPI:

• конфіденційність

• цілісність

• справжність обох сторін інформаційного обміну.

GSSAPI описує приблизно 45 викликів. Основні:

• GSS_Acquire_cred - отримання призначеного для користувача докази ідентичності (найчастіше закритий ключ, пароль)

• GSS_Import_name - конвертація імені користувача, зберігає в форму, що дозволяє визначити об'єкт безпеки

• GSS_Init_sec_context - створює клієнтський токен для відсилання на сервер (зазвичай виклик, в рамках моделі Виклик-відповідь (аутентифікація))

• GSS_Accept_sec_context - обробляє токен, створений за допомогою GSS_Init_sec_context і, можливо, повертає токен відповіді

• GSS_Wrap - конвертує дані додатки в форму захищеного повідомлення (зазвичай шифрация)

• GSS_Unwrap - витягує із захищеного повідомлення дані додатки (зазвичай розшифровка).

GSSAPI був стандартизований для мов C (RFC 2744) і Java (JSR-072).

До обмежень GSSAPI можна віднести те, що він стандартизує тільки аутентифікацію, але не авторизацію, і що він передбачає архітектуру клієнт-сервер.

Припускаючи поява нових механізмів безпеки, GSSAPI включає в себе спеціальний псевдо-механізм, SPNEGO, який дозволяє виявляти і використовувати механізми не існували на момент, коли додаток було зібрано.

GSSAPI часто застосовується в зв'язці з Kerberos. На відміну від GSSAPI, API Kerberos не стандартизований (і існують несумісні API). GSSAPI дозволяє використовувати різні реалізації Kerberos без зміни коду програми.

Близькі технології: RADIUS, SASL, TLS, SSPI, SPNEGO [5.1, 5.28].

Name (ім'я) - двійковий рядок для позначення ідентифікатора (ім'я користувача, додатки і т. Д.) Наприклад, Kerberos використовує формат 'user @ REALM для користувачів і service / hostname @ REALM для додатків.

• Credential (посвідчення) - інформація, яка доводить справжність об'єкта (зазвичай пароль або закритий ключ).

• Context (контекст) - стан каналу зв'язку

• Token (токен) - непрозоре (для додатка) повідомлення, яке надсилається на етапі встановлення з'єднання або в ході передачі захищеного повідомлення

• Mechanism (механізм) - реалізація нижчого рівня GSSAPI, що забезпечує фактичні ім'я, посвідчення і маркери. Типові механізми: Kerberos, NTLM, DCE, SESAME, SPKM, LIPKEY.

• Initiator / acceptor (ініціатор / одержувач) - сторона, що відправляє перший токен є ініціатором; протилежна сторона - одержувач. Зазвичай одержувачем є сервер, а ініціатором клієнт.

5.18. Kerberos - мережевий протокол аутентифікації

Kerberos - мережевий протокол аутентифікації, який пропонує механізм взаємної аутентифікації клієнта і сервера перед встановленням зв'язку між ними, причому в протоколі врахований той факт, що початковий обмін інформацією між клієнтом і сервером відбувається в незахищеною середовищі, а передані пакети можуть бути перехоплені і модифіковані [5.1, 5.30-5.35].

Перша версія протоколу Kerberos була створена в 1983 році в Массачусетському технологічному інституті (MIT) в рамках проекту «Афіна».

Основною метою проекту була розробка плану по впровадженню комп'ютерів в навчальний процес MIT і супутнього цьому ПО. Проект був освітнім, але в кінцевому підсумку результат включив кілька програмних продуктів, які широко використовуються і сьогодні (наприклад, X Window System). Загальнодоступним цей протокол став, починаючи з версії 4.

Припустимо, що існує дві людини, які знають один і той же секрет, відомий тільки цим двом. Тоді будь-який з них зможе легко переконатися, що має справу зі своїм напарником. Для цього йому всього лише доведеться перевірити, чи знає його співрозмовник загальний секрет.

Приклад.

Пункт 1. Домовленість про пароль. Нехай Аліса спілкується з Борисом. При цьому Борис використовує інформацію тільки тоді, коли впевнений, що інформація отримана від Аліси. Щоб уникнути підробки - вони домовилися між собою про пароль, який знають тільки вони удвох. При отриманні повідомлення Борис може укласти з листа - чи знає його співрозмовник пароль. Якщо співрозмовнику Бориса пароль відомий, то можна стверджувати, що його співрозмовником є Аліса.

Пункт 2. Виникнення проблеми передачі пароля. Тепер визначимо - яким же чином Алісі і Борису показувати знання пароля. Звичайно, можна просто включити пароль в текст листа. Наприклад: «Привіт, Борис. Наш пароль. ». Якби тільки Борис і Аліса були впевнені, що їхні листи ніхто не читає - тоді

можна було б скористатися цим способом. Але, на жаль, пошта передається по мережі, в якій працюють інші користувачі, серед яких є цікава Наталя. Наталя поставила собі завдання - з'ясувати пароль, відомий тільки Борису і Алісі, за допомогою якого вони обмінюються повідомленнями один з одним. Тепер абсолютно ясно, що не можна вказувати пароль просто в тексті листа. Значить, про паролі можна говорити відкрито, але при цьому треба дати знати співрозмовнику, що пароль Вам відомий.

Пункт 3. Рішення проблеми передачі пароля. Протокол Kerberos вирішує проблему передачі пароля засобами криптографії з секретним ключем. Замість того, щоб повідомляти один одному пароль, учасники сеансу зв'язку обмінюються криптографічним ключем, знання якої підтверджує особистість співрозмовника. Але для того, щоб така технологія була можлива, необхідно, щоб загальний ключ був симетричним, тобто, ключ повинен забезпечувати і шифрування, і розшифровку повідомлення.

Пункт 4. Проблема обміну паролями. При використанні простих протоколів, типу описаного вище, виникає одна важлива проблема. У випадку з Борисом і Алісою треба зрозуміти, як вони домовляються про секретний ключі для листування один з одним. Звичайно, люди можуть зустрітися в парку і домовитися, але ж в мережевих переговорах беруть участь і машини. Якщо під Алісою розуміти клієнтську програму, а під Борисом - службу на мережевому сервері, то зустрітися вони ніяк не можуть. Проблема полягає ще в тому, що коли клієнту треба посилати повідомлення на кілька серверів, в цьому випадку для кожного сервера їй доведеться обзавестися окремим ключем. А сервера тоді буде потрібно стільки секретних ключів, скільки у нього клієнтів. Необхідність зберігати на кожному комп'ютері таку кількість ключів створює ризик для всієї системи безпеки.

Пункт 5. Рішення проблеми обміну паролями. Для вирішення цих проблем проектом «Афіна» і був розроблений спеціальний протокол - Kerberos. За аналогією з давньогрецькою міфологією, цей протокол був названий на честь

триголового пса, який захищав вихід з царства Аїда, - Цербера, або більш точно - Кербера. Трьом головам Цербера в протоколі відповідають три учасники безпечної зв'язку: клієнт, сервер і довірений посередник між ними. Роль посередника тут грає центр розподілу ключів «Key distribution center», KDC.

KDC - це служба, яка працює на фізично захищеному сервері. KDC зберігає базу даних з інформацією про облікові записи всіх клієнтів мережі. Разом з інформацією про кожного абонента в базі KDC зберігається криптографічний ключ, відомий тільки цього абоненту і службі KDC. Цей ключ служить для зв'язку клієнта з центром [5.1, 5.30-5.35].

Звернення клієнта до сервера через KDC. Якщо клієнт хоче звернутися до сервера - він посилає повідомлення KDC. KDC надсилає кожному учаснику сеансу копії сеансового ключа, що діють протягом невеликого проміжку часу.

Призначення цих ключів - проведення аутентифікації клієнта і сервера.

Копія сеансового ключа, що пересилається на сервер, шифрується за допомогою довгострокового ключа цього сервера, а спрямовується клієнту - довготривалого ключа даного клієнта.

Теоретично, для виконання функцій довіреної посередника центру KDC досить направити сеансові ключі безпосередньо абонентам безпеки. Однак на практиці реалізувати таку схему надзвичайно складно. Тому на практиці застосовується інша схема керування паролями, яка робить протокол Kerberos набагато більш ефективним.

Сеансові мандати. У відповідь на запит клієнта, який має намір підключитися до сервера, служба KDC направляє обидві копії сеансового ключа клієнта. Повідомлення, призначене клієнту, шифрується за допомогою довгострокового ключа, загального для даного клієнта і KDC, а сеансовий ключ для сервера разом з інформацією про клієнта вкладається в блок даних, який отримав назву сеансового мандата («session ticket»).

Потім сеансовий мандат цілком шифрується за допомогою довгострокового ключа, який знають тільки служба KDC і даний сервер.

Після цього вся відповідальність за обробку мандата, який несе в собі зашифровані сеансовий ключ, покладається на клієнта, який повинен доставити його на сервер.

Отримавши відповідь KDC, клієнт отримує з нього мандат і свою копію сеансового ключа, які поміщає в безпечне сховище (воно розташовується не на диску, а в оперативній пам'яті).

Коли виникає необхідність зв'язатися з сервером, клієнт посилає йому повідомлення, що складається з мандата, який як і раніше зашифрований із застосуванням довготривалого ключа цього сервера, і власного аутентифікатора, зашифрованого за допомогою сеансового ключа.

Цей мандат в комбінації з аутентифікатором якраз і становить посвідчення, за яким сервер визначає «особистість» клієнта.

Сервер, отримавши «посвідчення особи» клієнта, перш за все за допомогою свого секретного ключа розшифровує сеансовий мандат і витягує з нього сеансовий ключ, який потім використовує для дешифрування аутентифікатора клієнта.

Якщо все проходить нормально - робиться висновок, що посвідчення клієнта видано довіреною посередником, тобто - службою KDC.

Клієнт може зажадати у сервера проведення взаємної аутентифікації. У цьому випадку сервер за допомогою своєї копії сеансового ключа шифрує мітку часу з аутентифікатора клієнта і в такому вигляді пересилає її клієнту в якості власного аутентифікатора.

Одна з переваг застосування сеансових мандатів полягає в тому, що сервера не потрібно зберігати сеансові ключі для зв'язку з клієнтами. Вони зберігаються в кеш-пам'яті посвідчень («credentials cache») клієнта, який направляє мандат на сервер кожен раз, коли хоче зв'язатися з ним.

Сервер, зі свого боку, отримавши від клієнта мандат, дешифрує його і витягує сеансовий ключ. Коли потреба в цьому ключі зникає, сервер може просто стерти його зі своєї пам'яті.

Такий метод дає і ще одну перевагу: у клієнта зникає необхідність звертатися до центру KDC перед кожним сеансом зв'язку з конкретним сервером [5.1, 5.30-5.35].

Сеансові мандати можна використовувати багаторазово. На випадок же їх розкрадання встановлюється термін придатності мандата, який KDC вказує в самій структурі даних.

Це час визначається політикою Kerberos для конкретної області. Зазвичай термін придатності мандатів не перевищує восьми годин, тобто - стандартної тривалості одного сеансу роботи в мережі. Коли користувач відключається від неї, кеш-пам'ять посвідчень обнуляється, і все сеансу мандати разом з сеансовими ключами знищуються.

Ранні версії Kerberos (с 1 по 3) були створені всередині MIT і використовувалися в цілях тестування. Ці реалізації містили істотні обмеження і були корисні тільки для вивчення нових ідей і виявлення проблем, які могли виникнути під час розробки.

Kerberos 4 вперше була опублікована 24 січня 1989 року. Вона стала першою версією, поширюваної за межами MIT, підготовленої для декількох виробників, які включили її в свої операційні системи. Крім того, інші великі проекти по розподілених системам (наприклад, Andrew File System) використовували ідеї Kerberos 4 для своїх систем аутентифікації. Основними розробниками даної версії були Стів Міллер (Steve Miller) і Кліффорд Ньюман (Clifford Neuman).

Основи того, що повинно було стати Kerberos 4, були описані в технічному плані «Афіна», а остаточний варіант був закріплений в вихідному коді еталонної реалізації, опублікованій MIT.

Однак через обмеження на експорт програмного забезпечення, що використовує шифрування, накладених американським урядом, Kerberos 4 не міг бути поширений за межами Сполучених Штатів. Так як Kerberos 4 при шифруванні використовував алгоритм DES, організації за межами США не

могли за законом використовувати дане програмне забезпечення. У відповідь на це команда розробників з MIT створила спеціальну версію Kerberos 4, виключивши з неї весь код, що стосується шифрування. Дані заходи дозволили обійти обмеження на експорт.

Пізніше Еррол Янг (Errol Young) в Університеті зв'язку Австралії (Bond University of Australia) додав в цю версію власну реалізацію DES. Вона називалася «E-Bones» (скорочення від «encrypted bones» [40]) і могла вільно поширюватися за межами США.

У 2006 році було оголошено про припинення підтримки Kerberos 4 [41].

З метою подолання проблем безпеки попередньої версії Джоном Колем (John Kohl) і Клиффордом Ньюманом (Clifford Neuman) була розроблена 5 версія протоколу, яка в 1993 році була опублікована в RFC 1510. З плином часу, в 2005 специфікацією почала займатися IETF Kerberos work group. Опубліковані ними документи включають в себе:

• характеристики шифрування і контрольної суми (RFC 3961);

• стандарт шифрування Advanced Encryption Standard (AES) (RFC 3962);

• Kerberos 5 «The Kerberos Network Authentication Service (V5)» (RFC 4120), уточнює деякі аспекти RFC 1510, і має намір використовуватися для більш докладного і чіткого опису;

• нове видання GSS-API специфікації «The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2.» (RFC 4121).

У червні 2006 року був представлений RFC 4556 описує розширення для 5-ї версії під назвою PKINIT (Public key cryptography for initial authentication in Kerberos).

Даний RFC описував, як використовувати асиметричне шифрування на етапі аутентифікації клієнта.

На наступний рік (2007) MIT сформували Kerberos Консорціум (Kerberos Consortium) щодо сприяння подальшому розвитку.

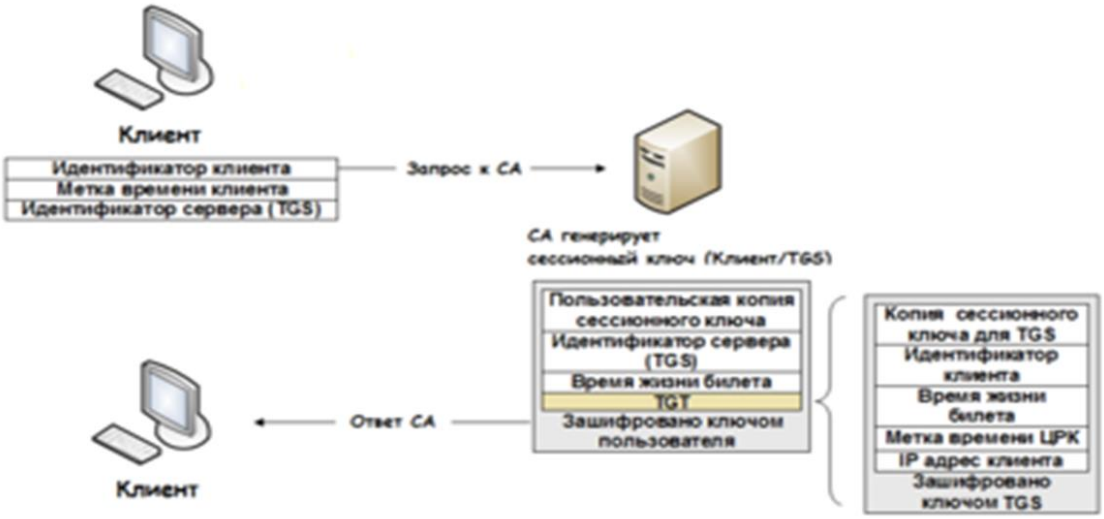


Рис. 5.21. Етап аутентифікації кліента

Kerberos 4 в значній мірі заснований на протоколі Нідхема-Шредера, але з двома істотними змінами.

• Перша зміна протоколу зменшувало кількість повідомлень, що пересилаються між клієнтом і сервером аутентифікації.

• Друге, більш суттєва зміна базового протоколу, полягає у введенні TGT (Ticket granting ticket - мандат для отримання мандата) концепції, що дозволяє користувачам аутентифікуватися в декількох сервісах, використовуючи свої довірливі дані тільки один раз.

Як результат, протокол Kerberos 4 містить два логічних компонента:

• Сервер аутентифікації (CA, Authentication Server, AS)

• Сервер видачі мандатів або дозволів (Ticket Granting Server, TGS).

Зазвичай ці компоненти поставляються як єдина програма, яка запускається на центрі розподілу ключів (KDC - містить базу даних логінів / паролів для користувачів і сервісів використовують Kerberos).

Сервер аутентифікації виконує одну функцію: отримує запит, який містить ім'я клієнта, що запитує аутентифікацію, і повертає йому зашифрований TGT. Потім користувач може використовувати цей TGT для запиту подальших мандатів на інші сервіси. У більшості реалізацій Kerberos час життя TGT 8-10 годин. Після цього клієнт знову повинен запросити його у CA.

Перше повідомлення, що відправляється центру розподілу ключів - запит до CA, так само відомий як AS_REQ. Це повідомлення відправляється відкритим текстом і містить ідентифікаційні дані клієнта, мітку часу клієнта і ідентифікатор сервера, який надає мандат (TGS).

Коли KDC отримує AS_REQ повідомлення - він перевіряє, що клієнт, від якого прийшов запит, існує, і його мітка часу близька до локального часу KDC (зазвичай ± 5 хвилин). Дана перевірка проводиться не для захисту від повторів (повідомлення надсилається відкритим текстом), а для перевірки відповідності часу. Якщо хоча б одна з перевірок не проходить - клієнтові відправляється повідомлення про помилку, і він не аутентифікується [5.1, 5.30-5.35].

У разі вдалої перевірки CA генерує випадковий сеансовий ключ, який буде спільно використовуватися клієнтом і TGS (даний ключ захищає подальші запити мандатів у TGS на інші сервіси). KDC створює 2 копії сесійного ключа: одну для клієнта і одну для TGS.

Потім KDC відповідає клієнту повідомленням сервера аутентифікації (AS_REP), зашифрованим довгостроковим ключем клієнта. Це повідомлення включає TGT, зашифрований TGS ключем, копію сесійного ключа для клієнта, час життя мандата і ідентифікатор TGS (TGT містить: копію сесійного ключа для TGS, ідентифікатор клієнта, час життя мандата, мітку часу KDC, IP адреса клієнта).

Коли користувач захоче отримати доступ до сервісу - він підготує повідомлення для TGS (TGS_REQ), що містить 3 частини: ідентифікатор сервісу, копію TGT, отриману раніше, і аутентифікатор (аутентифікатор складається з мітки часу, зашифрованою сесійним ключем, отриманим від CA, і служить для захисту від повторів).

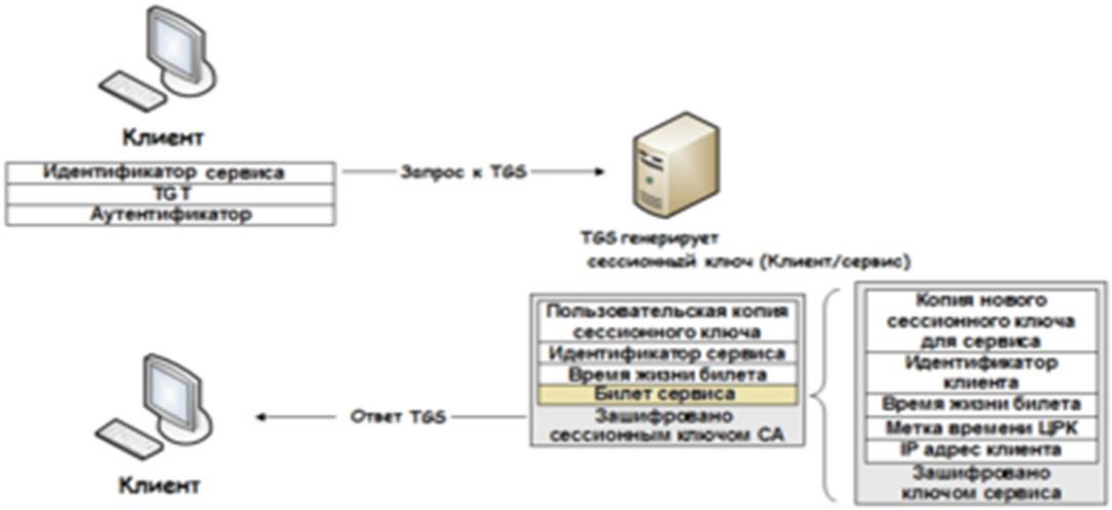


Рис. 5.22. Етап авторизації клієнта на TGS

При отриманні запиту мандата від клієнта KDC формує новий сесійний ключ для взаємодії клієнт / сервіс.

Потім відправляє у відповідь повідомлення (TGS_REP), зашифроване сесійним ключем, отриманим від CA.

Це повідомлення містить новий сеансовий ключ, мандат сервісу, зашифрований довготривалим ключем сервісу, ідентифікатор сервісу і час життя мандата (Service ticket містить: копію нового сесійного ключа, ідентифікатор клієнта, час життя мандата, локальне час KDC, IP клієнта).

Деталі останнього кроку - відправки мандата служби сервера додатків - не стандартизовані Kerberos 4, тому його реалізація повністю залежить від додатка.

Kerberos 5 є розвитком четвертої версії, включає всю попередню функціональність і містить безліч розширень. Однак з точки зору реалізації Kerberos 5 є абсолютно новим протоколом.

Основною причиною появи п'ятої версії була неможливість розширення. Згодом, атака повним перебором на DES використовується в Kerberos 4 стала актуальна, але використовувані поля в повідомленнях мали фіксований розмір і використовувати більш стійкий алгоритм шифрування не представлялося можливим.

Для вирішення даної проблеми було вирішено створити розширюваний

протокол з можливістю використання на різних платформах на основі технології ASN.1. Це дозволило використовувати в транзакціях різні типи шифрування. Завдяки цьому була реалізована сумісність з попередньою версією. Крім того у KDC з'являється можливість вибирати найбільш безпечний протокол шифрування, підтримуваний сторонами [5.1, 5.33-5.35].

Крім того оригінальний протокол Kerberos 4 схильний перебору по словнику. Дана уразливість пов'язана з тим, що KDC видає на вимогу зашифрований TGT будь-якому клієнту. Важливість даної проблеми також наголошує на тому, що користувачі зазвичай вибирають прості паролі.

Щоб ускладнити проведення даної атаки, в Kerberos 5 було введено попереднє встановлення автентичності. На даному етапі KDC вимагає, щоб користувач засвідчив свою особистість перш, ніж йому буде виданий мандат.

За попередню аутентифікацію відповідає політика KDC, якщо вона потрібна, то користувач при першому запиті до CA отримає повідомлення KRB_ERROR. Це повідомлення скаже клієнтові, що необхідно відправляти AS_REQ запит зі своїми даними для встановлення автентичності. Якщо KDC не впізнав їх, то користувач отримає ще одне повідомлення KRB_ERROR, що повідомляє про помилку, і TGT не буде виданий.69 61'.

- Протокол використовує тільки симетричне шифрування і передбачає, що у кожного кореспондента (Аліси і Бориса) є загальний секретний ключ з третьої довіреної стороною (Трент).
- Аліса направляє довіреної стороні (Трент) свій ідентифікатор і Боба:

{\displaystyle Alice\to \left\{A,B\right\}\to Trent}

Трент генерує два повідомлення. Перше включає мітку часу, *{\displaystyle T_{T}}*, час життя ключа ключа *{\displaystyle L},* новий сеансовий ключ для Аліси і Бориса *{\displaystyle K}* і ідентифікатор Бориса *{\displaystyle B}.* Це повідомлення шифрується загальним ключем Аліси і Трента. Друге повідомлення містить те ж саме, крім ідентифікатора - він замінений на ідентифікатор Аліси *{\displaystyle A}*. Саме повідомлення шифрується загальним

ключем Трента і Бориса: *{\displaystyle Trent\to \left\{E_{A}\left(T_{T},L,K,B\right),E_{B}\left(T_{T},L,K,A\right)\right\}\to Alice}*

■ Аліса генерує повідомлення з власного ідентифікатора *{\displaystyle A}* і мітки часу *{\displaystyle T_{T}}* , після чого шифрує повідомлення сеансовим ключем *{\displaystyle K}* і посилає Бобу разом з другим повідомленням від Трента: *{\displaystyle Alice\to \left\{E_{K}\left(A,T_{T}\right),E_{B}\left(T_{T},L,K,A\right)\right\}\to Boris}*

З метою власної аутентифікації Боб шифрує модифіковану мітку часу *{\displaystyle T_{T}+1}* загальним сеансовим ключем *{\displaystyle K}* і посилає її Алісі: *{\displaystyle Bob\to \left\{E_{K}\left(T_{T} +1\right) \right\} \to Alice}*

Важливим припущенням є синхронізація годин всіх учасників протоколу. Однак на практиці використовується синхронізація з точністю до декількох хвилин із запам'ятовуванням історії передач (з метою виявлення повтору) протягом деякого часу.

Схема роботи Kerberos 5 (рис. 5.22) в даний час відбувається наступним чином:

Вхід користувача в систему:

1. Користувач вводить ім'я і пароль на клієнтській машині.

2. Клієнтська машина виконує над паролем односторонню функцію (зазвичай хеш), і результат стає секретним ключем клієнта / користувача.

Аутентифікація клієнта:

1. Клієнт відсилає запит (AS_REQ) на CA для отримання аутентифікаційних вірчих даних і подальшого їх надання TGS сервера (згодом він буде їх використовувати для отримання мандатів без додаткових запитів на застосування секретного ключа користувача.) Даний запит містить:

Ідентифікатор клієнта, його мітка часу і ідентифікатор сервера.

2. Якщо політика KDC вимагає попередньої аутентифікації, то користувач отримує повідомлення KRB_ERROR, у відповідь на яке посилає повторний запит, але вже с даними для встановлення автентичності.

3. CA перевіряє, чи є такий клієнт в базі. Якщо є, то назад CA відправляє повідомлення (AS_REP), що включає:

o Сесійна ключ Клієнт / TGS, ідентифікатор TGS та час життя мандата, зашифровані секретним ключем клієнта.

o TGT (який включає ідентифікатор і мережеву адресу клієнта, мітку часу KDC, період дії мандата і сесійний ключ Клієнт / TGS), зашифрований секретним ключем TGS.

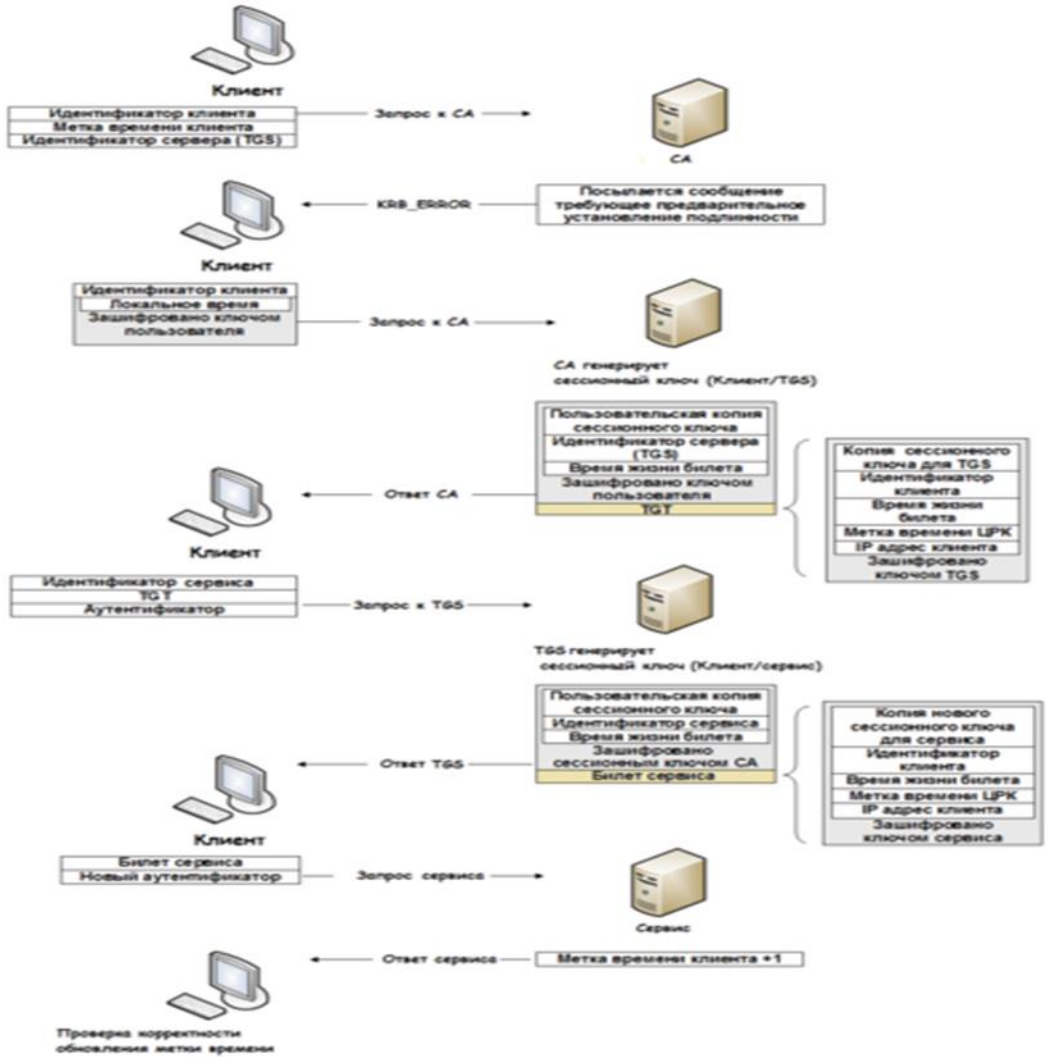


Рис. 5.23. Схема роботи Kerberos 5

Якщо ж ні, то клієнт отримує нове повідомлення, яке говорить про виникнення помилки [5.1, 5.30-5.35].

1. Отримавши повідомлення, клієнт розшифровує свою частину для отримання сесійної Ключа Клієнт / TGS. Цей сесійний ключ використовується

для подальшого обміну з сервером TGS. (Важливо: Клієнт не може розшифрувати TGT, так як воно зашифровано секретним ключем TGS) У цей момент у користувача достатньо даних, щоб авторизуватися на TGS.

Авторизація клієнта на TGS:

1. Для запиту сервісу клієнт формує запит на TGS (TGS_REQ) містить наступні дані:

TGT, отриманий раніше і ідентифікатор сервісу.

Аутентифікатор (складений з ID клієнта і тимчасового штампа), зашифрований на Сесійній Ключі Клієнт / TGS.

2. Після отримання TGS_REQ, TGS витягує з нього TGT і розшифровує його використовуючи секретний ключ TGS. Це дає йому Сесійна Ключ Клієнт / TGS. Їм він розшифровує аутентифікатор. Потім він генерує сесійний ключ клієнт / сервіс і посилає відповідь (TGS_REP) включає:

мандат сервісу (який містить ID клієнта, мережеву адресу клієнта, мітку часу KDC, час дії мандата і Сесійна Ключ клієнт / сервіс) зашифрований секретним ключем сервісу.

Сесійна ключ клієнт / сервіс, ідентифікатор сервісу і час життя мандата, зашифровані на Сесійній Ключі Client / TGS.

Запит сервісу клієнтом:

1. Після отримання TGS_REP, у клієнта достатньо інформації для авторизації на сервісі. Клієнт з'єднується з ним і посилає повідомлення містить:

Зашифрований мандат сервісу отриманий раніше.

Новий аутентифікатор, зашифрований на сесійному ключі клієнт / сервіс, і включає ID клієнта і мітку часу.

2. Сервіс розшифровує мандат використовуючи свій секретний ключ і отримує сесійний ключ клієнт / сервіс. Використовуючи новий ключ, він розшифровує аутентифікатор і посилає клієнтові наступне повідомлення для підтвердження готовності обслужити клієнта і показати, що сервер дійсно є тим, за кого себе видає мітку часу, зазначену клієнтом + 1, зашифровану на

сесійному ключі клієнт / сервіс.

3. Клієнт розшифровує підтвердження, використовуючи сесійний ключ клієнт / сервіс і перевіряє, чи дійсно мітка часу коректно оновлена. Якщо це так, то клієнт може довіряти сервера і може почати посилати запити на сервер.

4. Сервер надає клієнту необхідний сервіс.

Розширення PKINIT торкнулося етапу попередньої аутентифікації клієнта. Після чого вона стала відбуватися таким чином:

1. Користувач ідентифікується в системі і пред'являє свій закритий ключ.

2. Клієнтська машина формує запит на CA (AS_REQ), в якому вказує, що буде використовуватися асиметричне шифрування. Відмінність даного запиту полягає в тому, що він підписується (за допомогою закритого ключа клієнта) і крім стандартної інформації містить сертифікат відкритого ключа користувача.

3. Отримавши запит, KDC спочатку перевіряє достовірність сертифіката користувача, а потім електронний підпис (використовуючи отриманий відкритий ключ користувача). Після цього KDC перевіряє локальне час, прислане в запиті (для захисту від повторів).

4. Упевнившись в автентичності клієнта, KDC формує відповідь (AS_REP), в якому на відміну від стандартної версії, сеансовий ключ зашифрована відкритим ключем користувача. Крім того відповідь містить сертифікат KDC і підписується його закритим ключем (аналогічно запиту клієнта).

5. Отримавши відповідь, користувач перевіряє підпис KDC і розшифровує свій сеансовий ключ (використовуючи свій закритий).

Подальші етапи відбуваються згідно стандартному опису Kerberos V5.

В табл. 5.5 - 5.7 показано формат заголовка.

Таблиця 5.5. Сервер повідомляє про свій вибір

Розмір	Опис
1 байт	Номер версії SOCKS (повинен бути 0x05 для цієї версії)
1 байт	Обраний метод аутентифікації або 0xFF, якщо не було запропоновано прийнятного методу

Подальша ідентифікація залежить від обраного методу.

Таблиця 5.6. Запит клієнта

Розмір	Опис
1 байт	Номер версии SOCKS (должен быть 0x05 для этой версии)
1 байт	Код команды: • 0x01 = установка TCP/IP соединения • 0x02 = назначение TCP/IP порта (binding) • 0x03 = ассоциирование UDP-порта
1 байт	Зарезервированный байт, должен быть 0x00
1 байт	Тип адреса: • 0x01 = адрес IPv4 • 0x03 = имя домена • 0x04 = адрес IPv6
Зависит от типа адреса	Назначение адреса: • 4 байта для адреса IPv4 • Первый байт — длина имени, затем следует имя домена без завершающего нуля на конце • 16 байт для адреса IPv6
2 байта	Номер порта, в порядке от старшего к младшему (big-endian)

Таблиця 5.7 Відповідь сервера

Розмір	Опис
1 байт	Номер версії SOCKS (0x05 для цієї версії)
1 байт	Код відповідіответа: • 0x00 = запрос предоставлений • 0x01 = помилка SOCKS-сервера • 0x02 = з'єднання заборонено набором правил • 0x03 = мережу недоступна • 0x04 = хост недоступний • 0x05 = відмова в з'єднанні • 0x06 = закінчення TTL • 0x07 = команда не підтримується / помилка протоколу • 0x08 = тип адреси не підтримується
1 байт	Байт зарезервований, повинен бути 0x00
1 байт	Тип подальшого адреси: 0x01 = адреса IPv4 • 0x03 = ім’я домена • 0x04 = адресf IPv6
Зависит от типа адреса	Призначення адреси: 4 байта для адреса IPv4 • Перший байт - довжина імені, потім слідує ім'я домена без завершального нуля на кінці • 16 байт для адресb IPv6
2 байта	Номер порту, в порядку від старшого до молодшого (big-endian)

5.19. ZIP - протокол стека AppleTalk

Zone Information Protocol (ZIP) - (протокол інформації про зону) являє собою протокол сеансового рівня стека AppleTalk, який підтримує відповідність між номером мережі і ім'ям зони в маршрутизаторах мереж AppleTalk.

Протокол ZIP використовується переважно маршрутизаторами AppleTalk. Однак і інші новостворені вузли мережі використовують служби протоколу ZIP для вибору зони. У кожному маршрутизаторі ZIP веде таблицю інформації про зону (zone information table - ZIT). [5.1, 5.13, 5.35].

Таблиці ZIT представляють собою списки, де кожному номеру мережі відповідає одне або кілька імен зон. Кожна таблиця ZIT містить карту відповідностей між номерами мереж і іменами зон для кожної мережі в об'єднаній мережі.

Протокол інформації про зону (ZIP) - це протокол рівня сеансу в наборі протоколів AppleTalk, який підтримує зіставлення імен номерів між номерами в маршрутизаторах AppleTalk. ZIP використовується в основному маршрутизаторами AppleTalk. Однак інші мережеві вузли використовують служби ZIP при запуску, щоб вибрати свою зону. ZIP підтримує таблицю інформації про зону (ZIT) в кожному маршрутизаторі.

ZIT - це списки, підтримувані ZIP, які відображають певні мережеві номера для одного або декількох імен зон. Кожен ZIT містить мережеве зіставлення імен номерів для кожної мережі в міжмережевий мережі. ZIP надає додаткам і процесам доступ до імен зон. Зона являє собою логічну угруповання вузлів в мережі AppleTalk, і кожна зона ідентифікується по імені. Ім'я зони зазвичай використовується для ідентифікації приналежності між групою вузлів, наприклад групою вузлів, що належать до певного відділу всередині організації.

Блок схема (рис. 63) показує ZIP і його базові протоколи. Частина ZIP, яка реалізована на некінцевих вузлах, таких як робочі станції, використовує служби ATP.

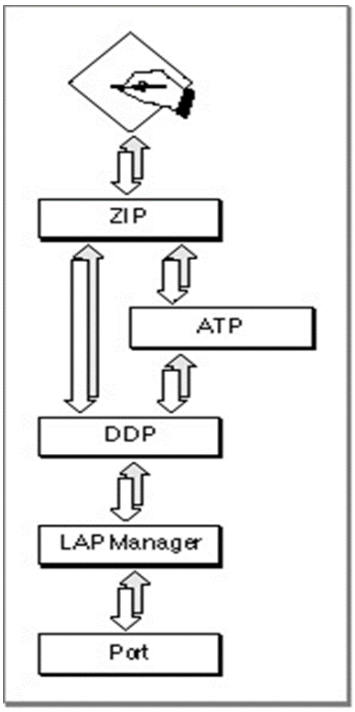


Рис. 5.24. Блок схема показує взаємозв'язок ZIP з іншими протоколами стека

ZIP підтримує відображення мереж і зон, які вони включають для всіх мереж, що належать до мережі AppleTalk:

• Кожен вузол мережі належить зоні; вузол може належати тільки одній зоні одночасно;

• Необмежена мережа містить лише одну зону, і всі вузли в цій мережі належать до однієї і тієї ж зоні.

• Одна розширена мережа може містити вузли, які відносяться до 255 різних зон. Одна зона може включати в себе вузли, що належать різним розширеним мереж. Кожна розширена мережа AppleTalk асоціювала з нею список зон, до яких можуть належати його вузли. Вузол, що з'єднує мережу, може вибрати свою зону з цього списку.

На кожному вузлі маршрутизатора в мережі ZIP створює таблицю інформації про зону, яка включає в себе номер кожної мережі (розширені мережі мають діапазони номерів мережі) в поєднанні з списком зон мережі.

Вузли, які не є маршрутизаторами, такі як системи кінцевих користувачів, не містять таблицю інформації про зону. Проте, частина ZIP реалізована на кожному непереадресованому вузлі, щоб додатки і процеси могли отримати

доступ до імені зони свого власного вузла, іменах всіх зон в їх локальної мережі або іменам всіх зон в мережі.

Драйвер .XPP реалізує частину ZIP, яка знаходиться на некінцевих вузлах, і надає інтерфейс, який дозволяє додатку або процесу запитувати інформацію про назву зони в діалозі на основі транзакцій.

ZIP використовує служби ATP на основі транзакцій для транспортування запитів з вузлів робочої станції на вузли маршрутизатора.

Інформаційний протокол зони (ZIP) використовується для підтримки зіставлення між мережами і іменами зон, щоб полегшити процес пошуку імені, що виконується протоколом прив'язки імені.

Деякі змінні, що відносяться до протоколу інформації про зону (ZIP), застосовні тільки до маршрутизаторів AppleTalk. Такі змінні включені в цю групу.Група ZIP End Node управляє змінними, що відносяться до протоколу інформації про зону (ZIP), які можуть застосовуватися як до маршрутизаторів, так і до кінцевих вузлів.

ZIP надає три функції, які можна використовувати для отримання імен зареєстрованих зон. Можна використовувати ці функції для отримання:

• ім'я зони, до якої належить додаток і його вузол

• імена зон в локальній мережі або імена всіх зон, які існують у всій мережі AppleTalk, до якої належить локальна мережа

Додатки, запущені на вузлах, підключених як до розширених, так і до необмежених мереж, можуть використовувати ZIP для отримання імені зони свого вузла. Додаток, запущене на вузлі, що належить розширеної мережі, може викликати ZIP, щоб отримати список всіх імен зон, пов'язаних з цією мережею. Наприклад, додаток мережевого адміністрування може використовувати ZIP для надання адміністратору списку зон для певної мережі, щоб адміністратор міг вибрати правильну зону для вузла при додаванні вузлів в мережу.

Можна використовувати ZIP в поєднанні з NBP. Наприклад, можна використовувати ZIP для пошуку зон в мережі, а потім використовувати NBP

для пошуку імен в кожній зоні.

ZIP відправляє функції GetMyZone, GetLocalZones і GetZoneList як запитів протоколу транзакцій AppleTalk (ATP). Ці запити завжди запитують один відповідь [5.1, 5.30-5.33].

Наприклад, коли викликається ZIP для запиту інформації про назву зони, частина ZIP, реалізована на вузлі, на якому запущено додаток, відправляє запит з використанням транзакційних служб ATP на частину ZIP, реалізовану на локальному маршрутизаторі, який містить таблицю ZIP; використовуючи ATP, ZIP на вузлі маршрутизатора передає відповідь на запит.

Коли викликається GetMyZone для отримання імені зони вузла, ZIP повертає повне ім'я зони в одній відповіді ATP і записує це ім'я зони в наданий буфер. Однак, коли потрібно отримати список імен зон, що належать або локальної мережі, або всієї мережі, що утворює Інтернет, ZIP не завжди має можливість повертати повний список імен в одній відповіді ATP. В цьому випадку потрібно повторно викликати функцію ZIP в циклі, щоб отримати всі імена зон.

Кожен з функцій GetMyZone, GetLocalZones і GetZoneList використовує блок параметрів типу XPPParamBlock для введення вхідних і вихідних значень для виклику. Використовується запис варіанту xCallParam для блоку параметрів XPP для функцій ZIP. Цей блок параметрів містить поле ioRefNum, яке інтерфейс MPW встановлює на номер документу драйвера XPP.

Блок параметрів для кожної з трьох функцій ZIP включає поле csCode і поле xppSubCode. Немає необхідності встановлювати ці значення полів перед викликом функції; інтерфейс MPW заповнює значення для кожного з цих полів.

Значення поля csCode завжди одно xCall.

Значення поля xppSubCode визначає конкретну функцію ZIP, і вона відрізняється для кожної з трьох функцій.

Для цих трьох функцій ZIP вказується значення тайм-ауту і повтору, які визначають поведінку транзакції ATP, на яку розраховується виклик. Перед

викликом функції ZIP необхідно встановити значення для цих полів. Поле xppTimeout блоку параметрів використовується для встановлення значення тайм-ауту і поля xppRetry для встановлення значення повтору. Тайм-аут повідомляє ATP, скільки часу в секундах чекати між кожною спробою, а значення повтору вказує, як він може повторювати спробу.

Для кожної функції надається буфер для зберігання даних імені повернутої зони і буфер, який потрібно ZIP для його власного використання. Ці два буфера і блок параметрів XPPParamBlock, який призначається для цієї функції, належать ZIP для життя виклику; не можна маніпулювати ними або змінювати їх вміст під час операції. Пам'ять для цих буферів і блоку параметрів належить функції до тих пір, поки функція не завершить виконання.

Якщо для Boolean параметра async встановиться значення TRUE, або потрібно надати процедуру завершення, інакше програма має опитати поле ioResult блоку параметрів, щоб визначити, коли функція завершила операцію.

ЛІТЕРАТУРА

5.1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципи, технологии, протоколы: Учебник для вузов. 3-е изд. – СПб.: Питер, 2006.– 958 с.

5.2. RFC 3530 - NFS версії 4

5.3. RFC 1813 - NFS версії 3

5.4. RFC 1813 - NFS версії 3RFC 1094 - NFS версії 2

5.5. 5.5. Open Network Computing Remote Procedure Call, RFC 1057, RFC 1831

5.6. RFC 3010, RFC 3530

5.7. RFC 3010, пересмотренная версия — RFC 3530, апрель 2003

5.8. RFC 5661

5.9. Measuring Interactions Between Transport Protocols and Middleboxes. Alberto Medina, Mark Allman, and Sally Floyd. Internet Measurement Conference 2004, August 2004.

5.10. Aleksandar Kuzmanovic. The power of explicit congestion notification. In Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications. 2005

5.11. Network File System Version 4 // IETF

5.12. Протокол L2TP (Layer Two Tunneling Protocol) в Microsoft Technet

5.13. Настройка L2TP VPN в Windows

5.14. RFC 2661 Layer Two Tunneling Protocol «L2TP».

5.15. RFC 2341 Cisco Layer Two Forwarding (Protocol) «L2F». (Предшественник L2TP.)

5.16. RFC 2637 Point-to-Point Tunneling Protocol (PPTP). (Предшественник L2TP.)

5.17. RFC 2809 Implementation of L2TP Compulsory Tunneling via RADIUS.

5.18. RFC 2888 Secure Remote Access с помощью L2TP.

5.19. RFC 3070 Layer Two Tunneling Protocol (L2TP) через Frame Relay.

5.20. RFC 3145 L2TP Disconnect Cause Information.

5.21. RFC 3193 Защита L2TP используя IPsec.

5.22. RFC 3301 Layer Two Tunnelling Protocol (L2TP): ATM access network.

5.23. RFC 3308 Layer Two Tunneling Protocol (L2TP) Differentiated Services.

5.24. RFC 3355 Layer Two Tunnelling Protocol (L2TP) Over ATM Adaptation Layer 5 (AAL5).

5.25. RFC 3371 Layer Two Tunneling Protocol «L2TP» Management Information Base.

5.26. RFC 3437 Layer Two Tunneling Protocol Extensions for PPP Link Control Protocol Negotiation.

5.27. RFC 3438 Layer Two Tunneling Protocol (L2TP) Internet Assigned Numbers: Internet Assigned Numbers Authority (IANA) Considerations Update.

5.28. RFC 3573 Signaling of Modem-On-Hold status in Layer 2 Tunneling Protocol (L2TP).

5.29. RFC 3817 Layer 2 Tunneling Protocol (L2TP) Active Discovery Relay for PPP over Ethernet (PPPoE).

5.30. RFC 3931 Layer Two Tunneling Protocol — Version 3 (L2TPv3).

5.31. RFC 4045 Extensions to Support Efficient Carrying of Multicast Traffic in Layer-2 Tunneling Protocol (L2TP).

5.32. Протокол туннелей на сетевом уровне L2 (L2TP) / Семёнов Ю. А.. Telecommunication technologies — телекоммуникационные технологии — v. 3.5 — М.: ГНЦ ИТЭФ, 2010

5.33. RFC 4171 — Internet Storage Name Service (iSNS)

5.34. Luke Kenneth Casson Leighton. DCE/RPC over SMB: Samba and Windows NT Domain Internals. — Sams, 1999. — ISBN 1-57870-150-3.

5.35. RFC 4171 — Internet Storage Name Service (iSNS).

6. ПРОТОКОЛИ ПРЕДСТАВНИЦЬКОГО РІВНЯ

6.1. AFP - мережевий протокол представницького і прикладного рівнів

Apple Filing Protocol (AFP), «AppleShare», частина підсистеми Apple File Service, AFS) - мережевий протокол представницького і прикладного рівнів [6.1, 6.2] мережевий моделі OSI, що надає доступ до файлів в Mac OS X. Він підтримує Юнікод- сумісні імена файлів, обмеження файлів POSIX і ACL, розширену блокування файлів. До Mac OS 9 протокол був основним протоколом передачі файлів під Mac OS.

Сторонні реалізації протоколу AFP (AFS) доступні для операційних систем Windows, Novell NetWare, Linux і FreeNAS. [6.1- 6.4].

Починаючи з OS X версії 10.9 "Mavericks" (2013 рік), Apple використовує SMB2 замість AFP в якості основного протоколу віддаленого доступу до файлів [6.4].

Ранні версії сервера AFP були доступні в Mac OS починаючи з версії System 6 в AppleShare і AppleShare IP, також в Mac OS X Server 1.x. У клієнтських ОС AFP називається «Personal File Sharing» і підтримує до десяти підключень. [6.5]

Ця AFP-реалізація спирається на протокол версії 1.x або 2.x. AppleShare IP-5.x, 6.x, а реліз Mac OS X Server «1.x» представив протокол версії 2.2.

Це була перша версія, яка надавала транспортні зв'язки по стеку протоколів TCP / IP, також збільшено максимальний розмір «расшаренних» файлів з 2 ГБ до 4 ТБ, але максимальний розмір файлу, який може зберігатися, залишився рівним 2 гинув у зв'язку з обмеженням в Mac OS. [6.6]

Зміни, внесені в AFP починаючи з версії 3.0, являють собою значний прогрес в протоколі, надаючи функції, розроблені спеціально для клієнтів Mac OS X.

AFP 3.1 був введений в Mac OS X Server версії 10.2. Включена підтримка аутентифікації Kerberos і безпечні з'єднання AFP через Secure Shell (SSH). Максимальний розмір «расшаренних файлів» і розмір файлу збільшений до 8 Тіб починаючи з Mac OS X Server 10.2 [6.7], а потім до 16 Тіб в Mac OS X Server 10.3. [6.8].

AFP 3.2 додана підтримка списків контролю доступу та розширені атрибути в Mac OS X Server 10.4. Максимальний розмір «расшаренних» файлів становить не менше 16 Тіб, хоча компанія Apple не повідомляла про межах в Mac OS X Server 10.4. Також AFP 3.2 була введена в Mac OS X Leopard і виправлена підтримка Time Machine (синхронізація, методи захисту інформації та повідомлення в режимі сну).

AFP 3.3 додана підтримка відтворення кеш (потрібно для Time Machine).

AFP версій 3 і пізніші використовують виключно TCP / IP (номера портів 548 або 427) для передачі даних і підтримують AppleTalk тільки в якості протоколу виявлення.

Версії AFP 2.x підтримують роботу як через TCP / IP (використовуючи Data Stream Interface) так і через AppleTalk.

Більш ранні версії протоколу можуть використовувати тільки AppleTalk.

6.2. ASTERIX - протокол відповідальний за визначення та збір даних

ASTERIX (All Purpose Structured Eurocontrol Surveillance Information Exchange, багатоцільовій структурованій обмін інформацією спостереження Євроконтролю) - протокол прикладного/представніцького рівня, відповідальний за визначення та збір даних, розроблення для забезпечення трансляції та обміну даними спостереження [6.1-6.5].

Специфікація *ASTERIX* розбита на частини.

У першій частині описуються основні принципи, а решта частини відносяться до різних областей застосування. Кожній області застосування

повинна відповідати одна і тільки одна частина специфікації. Частина може мати різні версії, які можуть співіснувати одночасно і підтримуватися незалежно один від одного.

Правила дозволяють визначити 32 області застосування. Кожній області повинна бути присвоєна початкова категорія від 0 до 31 включно. Щоб уникнути неправильного тлумачення даних, будь-якої нової версії частини повинна бути присвоєна категорія, чий номер відрізняється від попереднього.

У специфікації ASTERIX редакції 2.1 (квітень 2013) і більш ранніх редакціях структура даних наступна (таб. 6.1-6.3).

Таблиця 6.1. Структура блоку даних

Байт	0	1...2	3...			
Позначення	CAT	LEN	RECORD 1	RECORD 2	..	RECORD K
Найменування	Категорія даних	Розмір блока даних	Запис 1	Запис 2	..	Запис K
Розмір, байт	1	2	1+			

Таблиця 6.2. Структура блоку запису

Байт	0...m	(m+1)...			
Позначення	FSPEC	DATA FIELD 1	DATA FIELD 2	...	DATA FIELD N
Найменування	Специфікація полей	Поле даних 1	Поле даних 2	...	Поле даних N
Розмір, байт	1+	0+	0+	...	0+

Таблиця 6.3. Специфікація полів (FSPEC)

Байт	0								1								...(L-1)	L							
Біт											0	1	2	3	4	5	6...(M-8)	-7	-6	-5	-4	-3	-2	-1	
Позначення	1	2	3	4	5	6	7	X	8	9	10	11	12	13	14	X	15...F[N-7]	[N-6]	[N-5]	[N-4]	[N-3]	[N-2]	[N-1]	[N]	X=0

де:

• L + 1 - розмір специфікації полів в байтах;

• M + 1 - розмір специфікації полів в бітах;

• N - кількість полів;

• F [Y] - індикатор наявності поля Y;

• FX - індикатор розширення полів.

Значення в біті F [Y] визначає чи присутній поле Y в запису. Якщо F [Y] дорівнює 0, то поле відсутнє, якщо F [Y] дорівнює 1, то поле присутній.

Якщо FX дорівнює 1, то наступний байт - це продовження специфікації полів. Якщо FX дорівнює 0, то це означає, що досягнуто кінець специфікації полів [6.1-6.5].

ASTERIX. Редакція 2.2 і пізніші специфікації ASTERIX редакції 2.2 (жовтень 2014 року) структура даних була змінена. Стало неможливим об'єднуватися записи в блоці даних. Дані представляються одним записом. Нова структура динячих використовується тільки для категорій 015 і 238. Для інших категорій старого зразка структура даних (таб. 6.4).

Таблиця 6.4. ASTERIX. Структура блоку запису

Байт	0	1...2	3...m	(m+1)...			
Позначення	CAT	LEN	FSPEC	Data field 1	Data field 2	.. .	Data field N
Найменування	Категорія даних	Розмір блока даних	Специфікація полів	Поле даних 1	Поле даних 2	.. .	Поле даних N
Розмір, байт	1	2	1+	0+	0+	.. .	0+

де: N — кількість полів

Класифікація полів даних наступна.

• *Стандартне поле даних.*

- *Поле фіксованої довжини.*
- *Поле довжини, що розширюється.*
- *Поле явно заданої довжини.*
- *Поле з повторенням.*
- *Складений поле.*

• *Поле даних спеціального призначення.*

• *Зарезервоване поле даних.*

- Стандартне поле даних
- Поле фіксованої довжини

  Повинно містити фіксовану кількість байт.
- Поле розширюється довжини

Має змінну довжину. Починається з основної частини, заздалегідь визначеної довжини, за якою слідує певна кількість додаткових частин заздалегідь певної довжини. Всі додаткові частини рівної довжини. Наявність такого додаткового поля визначається молодшим значущим бітом в останньому байті попередній частині (основної або додаткової). Такий біт називається індикатором розширення поля і позначається FX.

Таблиця 6.5. Структура поля довжини, що розширюється

Частина	Основна		Додаткові				
Розмір, байт	n		i		...	i	
Найменування/позначення	Дані	FX=1	Дані	FX=1	...	Дані	FX=0

Поле явно заданої довжини починається з байта, який містить довжину поля в байтах, включаючі і сам цей байт.

Таблиця 6.6. Структура поля явно заданої довжини

Байт	0	1...n
Найменування	Розмір поля	Данні
Розмір, байт	1	n

Поле з повторенням має змінну довжину. Починається з байта, значення якого дорівнює кількості підполів. Підполя однакового розміру, слідують один за одним.

Таблиця 6.7. Структура поля с повторенням

Байт	0	1...i	(i+1)...	
Позначення	REP=N	Data Subfield 1	...	Data Subfield N
Наіменування	Кількість повторів	Підполе 1	...	Підполе N
Розмір, байт	1	n	...	n

Складне поле має змінну довжину. Складається з основного підполя, за яким слідують підполя даних. Основне поле визначає наявність підполів і може розширюватися за допомогою механізму розширення поля (FX). Підполя можуть бути фіксованої довжини, довжини, що розширюється, явно заданої довжини і з повтореннями, але не можуть бути складовими.

Таблиця 6.8. Структура складного поля

Байт	0...m	(m+1)...			
Позначення	Primary Subfield	Data Subfield 1	Data Subfield 2	...	Data Subfield N
Найменування	Основне підполе	Підполе даних 1	Підполе даних 2	...	Підполе даних N
Розмір, байт	1+	0+	0+	...	0+

Основна концепція протоколу Kerberos.

Таблиця 6.9.Структура основного підполя

Байт	0								1								...(L-1)	L							
Бит											0	1	2	3	4	5	6...(M-8)	-7	-6	-5	-4	-3	-2	-1	
Позначення	F1	F2	F3	F4	F5	F6	F7	X	F8	F9	F10	F11	F12	F13	F14	X	F15...SF[N-7]	F[N-6]	F[N-5]	F[N-4]	F[N-3]	F[N-2]	F[N-1]	F[N]	X=0

де:• L + 1 - розмір специфікації підполів в байтах;

• M + 1 - розмір специфікації підполів в бітах;

• N - кількість підполів;

• SF [Y] - індикатор наявності підполя Y;

• FX - індикатор розширення поля.

Поле даних спеціального призначення використовується для передачі довільних даних. Поле прозоро для незацікавлених користувачів, тобто вміст може бути проігноровано.

Коли поле використовується, для нього виділяється біт в FSPEC. Позначення такого біта: SP.

Перший байт поля містить розмір поля в байтах, включаючи цей байт. Наступні байти містять довільні дані.

Таблиця 6.10. Структкра поля даних спеціального призначення

Байт	0	1...n
Найменування	Розмір поля	Данні
Розмір, байт	1	n

Зарезервоване поле данніх використовується тільки в специфікації ASTERIX редакції 2.1 і раніших. Призначено для надання механізму внесення проміжних змін в задану категорію. Поле прозоро, як і поле спеціального призначення. Перший байт поля містить розмір поля в байтах, включаючи цей байт. Наступні байти містять довільні дані.

Таблиця 6.11. Структура зарезервованого поля даних

Байт	0	1...n
Найменування	Розмір поля	Дані
Розмір, байт	1	n

Починаючи з специфікації ASTERIX редакції 2.2 поле не використовується.

6.3. Протокол датаграм безпеки транспортного рівня DTLS

Протокол датаграм безпеки транспортного рівня - Datagram Transport Layer Security (DTLS**)** забезпечує захищеність з'єднань для протоколів, що використовують датаграми [6.9-6.11].

DTLS дозволяє програмам, основаним на комунікаціях за допомогою датаграмм, повідомлятися безпечним способом, що запобігає перехоплення, прослуховування, втручання, не порушуючи захисту цілісності даних або підробку вмісту повідомлення.

Протокол DTLS заснований на потоковому протоколі Transport Layer Security (TLS) і забезпечує, таким чином, необхідні гарантії безпеки. Дейтаграммна семантика основного транспортного протоколу успадковується протоколом DTLS - його застосування не буде страждати від потокових затримок, але має враховувати витрати переупорядочивания пакетів, втратою датаграмм, а також надлишкового розміру даних, більших, ніж Датаграммним розмір пакета.

Протокол датаграм безпеки транспортного рівня - Datagram Transport Layer Security (DTLS) забезпечує захищеність з'єднань для протоколів, що використовують датаграми.

DTLS дозволяє програмам, основаним на комунікаціях за допомогою датаграмм, повідомлятися безпечним способом, що запобігає перехоплення, прослуховування, втручання, не порушуючи захисту цілісності даних або підробку вмісту повідомлення [6.13-6.18].

Протокол DTLS заснований на потоковому протоколі Transport Layer Security (TLS) і забезпечує, таким чином, необхідні гарантії безпеки. Дейтаграммна семантика основного транспортного протоколу успадковується протоколом DTLS - його застосування не буде страждати від потокових затримок, але має враховувати витрати переупорядочивания пакетів, втратою датаграмм, а також надлишкового розміру даних, більших, ніж Датаграммним

розмір пакета.

Версії протоколу DTLS 1.0 грунтуються на TLS 1.1, і версія DTLS 1.2 заснована на TLS 1.2.

Таблиця 6.12. Бібліотеки, які підтримують протокол DTLS'

Програме забезпечення	DTLS 1.0	DTLS 1.2
OpenSSL	Так	Так
GnuTLS	Так	Так
MatrixSSL	Так	Так
NSS	(Бета)	Ні
SChannel	Так	Ні
Secure Transport	Так	Ні
CyaSSL	Так	Так
Libsystools	Так	Ні
Python	Так	Ні
@nodertc/dtls	Ні	Так
java-dtls	Так	Так
pion/dtls(Go)	Ні	Так
californium/scandium (Java)	Ні	Так

6.4. XDR зовнішнє уявлення даних

External Data Representation (XDR) - міжнародний стандарт передачі даних в Інтернеті, який використовується в різних RFC для опису типів. дозволяє організувати не залежну від платформи передачу даних між комп'ютерами в гетерогенних мережах. [6.13-6.15].

External Data Representation - це стандарт IETF з 1995 року. Він дозволяє даними бути упакованими не залежних від архітектури способом, таким чином, дані можуть передаватися між гетерогенними комп'ютерними системами.

• Перетворення з локального уявлення в XDR називається кодуванням.

• Перетворення з XDR в локальне уявлення називається декодуванням.

• XDR виконаний як портативна (переносна) бібліотека функцій між різними операційними системами і так само не залежить від транспортного рівня.

XDR (External Data Representation - зовнішнє уявлення даних) - міжнародний стандарт передачі даних в Інтернеті, який використовується в різних RFC для опису типів. XDR дозволяє організувати не залежну від платформи передачу даних між комп'ютерами в гетерогенних мережах.

External Data Representation (XDR) - це стандарт IETF з 1995 року. Він дозволяє даними бути упакованими не залежних від архітектури способом, таким чином, дані можуть передаватися між гетерогенними комп'ютерними системами.

• Перетворення з локального уявлення в XDR називається кодуванням.

• Перетворення з XDR в локальне уявлення називається декодуванням.

• XDR виконаний як портативна (переносна) бібліотека функцій між різними операційними системами і так само не залежить від транспортного рівня.

Різні комп'ютери можуть мати різний внутрішнє подання інформації. Наприклад, 32-бітний Integer має 2 можливі форми представлення:

• Порядок байтів від старшого до молодшого (Motorola 68000)

• Прямий порядок байтів (Intel 80x86)

Для деяких функцій WinSock їхні аргументи (тобто, параметри функцій) повинні зберігатися в зворотному порядку.

• Сервер і клієнт можуть обмінюватися різними типами даних.

• Якщо сервер і клієнт виконуються на двох відповідних машинах, використовуючи різний внутрішнє представлення даних, то вони повинні узгоджувати точно представлення всіх даних, що передаються між ними.

• Sun Microsystems розробила external data representation (XDR), який визначає уявлення для різних типів даних (integer, enumeration ....)

• XDR став стандартом де-факто для більшості клієнт-серверних додатків:

- Програма перетворює повідомлення зі свого внутрішнього уявлення в XDR для подальшої передачі. Це називається кодуванням.

- Одержувач перетворює отримане повідомлення з XDR в власне уявлення. Це називається декодуванням.

XDR вказує уявлення для більшості типів даних в C:

Закодована інформація містить лише дані, вона не містить інформації про тип даних.

Наприклад, після кодування 32-бітного integer результатом буде 32-бітний integer в XDR. Чи не буде інформації про те, що це integer. Клієнти і сервери, що використовують XDR, повинні погоджувати тип даних повідомлень, якими вони обмінюються [6.13-6.18].

XDR вказує уявлення для більшості типів даних в C:

Закодована інформація містить лише дані, вона не містить інформації про тип даних.

Наприклад, після кодування 32-бітного integer результатом буде 32-бітний integer в XDR. Чи не буде інформації про те, що це integer. Клієнти і сервери, що використовують XDR, повинні погоджувати тип даних повідомлень, якими вони обмінюються.

Посилка повідомлення в XDR:

• посилати повідомлення може складатися з декількох пунктів даних (items).

• Наприклад, повідомлення містить інформацію про студента. Воно складається з трьох пунктів:

- ім'я (рядок символів) -ID (ціле) сукупність GPA (floating-point number)

• Перед посилкою повідомлення програма (клієнт або сервер) конвертує всю інформацію пунктів з внутрішнього представлення в XDR.

• Кроки конвертації:

1. Надання буфера для зберігання всієї інформації повідомлення, яка повинна бути надіслана.

2. Виклик xdrmem_create () для ініціалізації потоку XDR.

Наприклад: xdrmem_create () повертає покажчик на порожній потік.

1. Виклик стандартної програми в XDR для перетворення кожного пункту інформації. Вона буде дописувати закодовану інформацію в кінець потоку наступним чином:

* # * Поміщати закодовану інформацію в наступне доступне місце в буфері

* # * Оновлювати внутрішній покажчик на потік, поміщаючи його на нове доступне вільне місце.

• Коли програма отримує повідомлення в XDR, вона конвертує кожен пункт даних в повідомленні з XDR в своє внутрішнє уявлення.

• Кроки:

1. Виклик *xdrmem_create ()* для ініціалізації потоку XDR, вказавши *XDR_DECODE,* як четвертий аргумент.

2. Приміщення отриманого повідомлення в буфер.

3. Виклик підходящої стандартної програми перетворення для декодування кожного пункту даних отриманого повідомлення.

6.5. MIME - стандарт, що описує передачу різних типів даних по електронній пошті

Multipurpose Internet Mail Extensions **(**MIME) - стандарт, що описує передачу різних типів даних по електронній пошті, а також, в загальному випадку, специфікація для кодування інформації і форматування повідомлень таким чином, щоб їх можна було пересилати через Інтернет. [6.13-6.18].

MIME визначає механізми для передачі різного роду інформації всередині текстових даних (зокрема, за допомогою електронної пошти), а саме:

текст на мовах, для яких використовуються кодування, відмінні від ASCII, і нетекстові дані, такі, як картинки, музика, фільми і програми. MIME є також фундаментальним компонентом комунікаційних протоколів, таких як HTTP, яким потрібно, щоб дані передавалися в контексті повідомлень, подібних e-mail, навіть якщо дані реально не є e-mail.

Основний формат електронних повідомлень визначено в RFC 5322, який є оновленою версією RFC 2822 (який, в свою чергу, є оновленою версією RFC 822). Ці стандарти визначають схожі формати для текстових e-mail-заголовків і вмісту і правил, що відносяться до полів, які загально доступні, таким як To :, Subject :, From: і Date.

MIME визначає набір e-mail-заголовків для визначення додаткових атрибутів повідомлення, включаючи тип контенту, і визначає безліч кодувань, які можуть бути використані для подання 8-бітних бінарних даних за допомогою символів з 7-бітного ASCII. MIME також визначає правила для кодування символів з Extended ASCII (з кодами 128-255) в заголовках e-mail-повідомлення, таких як Subject :.

MIME розширюємо для нових типів - його визначення включає метод для реєстрації нових типів контенту та інших атрибутів.

Організація даних MIME. Формат MIME підтримує передачу кількох сутностей в межах одного повідомлення.

Причому суті можуть передаватися не тільки у вигляді однорівневої послідовності, але і у вигляді ієрархії з вкладенням елементів один в одного.

Для позначення множини вмісту використовуються медіатипи multipart/*.

Робота з такими типами здійснюється за загальними правилами, описаним в RFC 2046 (якщо інше не визначено конкретним медіа-типом). Якщо одержувачу невідомо, як працювати з типом, то він обробляє його так само, як multipart / mixed.

Для передачі множинного повідомлення в заголовок Content-Type додається параметр boundary (межа), який позначає послідовність символів, які

поділяють частини повідомлення. Межа може складатися з цифр, букв і символів « '() + _, -. /: =?». При використанні спеціальних символів (не цифри і букв) значення параметра boundary слід укладати в подвійні лапки ". Максимальна довжина кордону - 70 символів [6.16-6.18].

Початок кожної частини повідомлення позначається рядком --boundary. Кінець останнього повідомлення позначається рядком --boundary--. Найперші символи розриву рядків CRLF (коди 13 і 10), якими починаються і закінчуються прикордонні рядки, які не входять в вміст самої частини. Якщо за ними слідують ще переноси рядків, то вони вже належать включається частини.

Перед першою частиною і після останньої може бути додатковий текст. Він називається преамбулою і епілогом, відповідно. У протоколі HTTP ці елементи ігноруються.

В електронному листі преамбула може містити текст, виведений клієнтами електронної пошти, які не розуміють формату MIME.

На самому початку включається частини розташовуються заголовки, що описують її вміст (Content-Type, Content-Length і т. п.).

Перед безпосередньо тілом частини обов'язково повинна бути порожній рядок, навіть якщо заголовки відсутні.

Якщо не визначений Content-Type, то він береться за замовчуванням – text / plain.

Таблиця 6.13. Перелік стандартів

RFC	Дата	Тема	Обновлён в (Updated by)	Обновляет (Updates)	Заменён (Obsoleted by)	Заменяет (Obsoletes)
RFC 1556	Грудень 1993	Handling of Bi-directional Texts in MIME (Обрабка двонаправлених текстів в MIME)	—	—	—	—
RFC 2045	Листопад 1996	MIME Part One: Format of Internet Message Bodies (MIME Частина перша: Формат тіла повідомлень)	2184, 2231, 5335, 6532	—	—	1521, 1522, 1590
RFC 2046	листопад 1996	MIME Part Two: Media Types (MIME Частина друга: типи вмісту)	2646, 3798, 5147	—	—	1521, 1522, 1590
RFC 2047	листопад 1996	MIME Part Three: Message Header Extensions for Non-ASCII Text (MIME Частина третя: Розширення заголовка для не-ASCII-тексту)	2184, 2231	—	—	1521, 1522, 1590
RFC 2049	листопад 1996	MIME Part Five: Conformance Criteria and Examples (MIME Часть п'ята: Відповідність критеріям й приклади)	—	—	—	1521, 1522, 1590
RFC 4288	грудень 2005	Media Type Specifications and Registration Procedures	—	—	—	2048
RFC 4289	грудень 2005	MIME Part Four: Registration Procedures	—	—	—	2048
RFC 4855	лютий 2007	Media Type Registration of RTP Payload Formats	—	—	—	

6.6. NCP протокол функцій мережевих послуг для організації обміну

NetWare Core Protocol (NCP) - це мережевий протокол, який використовується в деяких продуктах від Novell, є надбудовою над протоколом IPX або TCP / IP і використовується для організації обміну між робочою станцією і файловим сервером [6.18-6.21].

В основному NCP пов'язаний і використовується в операційній системі NetWare, але його частини були реалізовані на інші платформи, такі як Linux, Windows NT і Unix.

Протокол використовується для доступу до файлів, службі друку, службі каталогу, синхронізації годин, обміну повідомленнями, віддаленого виконання команд та інших функцій мережевих послуг для організації обміну між робочою станцією і файловим сервером. Novell eDirectory використовує NCP для синхронізації змін даних між серверами в дереві служби каталогів.

Протокол NCP реалізований в NetWare 3.x на системному рівні. У

NetWare 4.x пропонується API-інтерфейс NCP Extension для звернення до протоколу NCP з прикладних програм на робочих станціях і в розробляється NLM-модулів. Для обміну даними між програмами по протоколу NCP використовуються пакети IPX з номером сокета 0x0451 і типом пакета 17.

Зв'язок між робочою станцією і файловим сервером, які використовують API-інтерфейс до протоколу NCP, зазвичай організовується за наступною схемою:

• NLM-модуль реєструє якусь свою функцію як розширення NCP;

• програма на робочій станції або файловому сервері зв'язується з NetWare і отримує необхідний ідентифікатор розширення NCP;

• програма на робочій станції або файловому сервері використовує зареєстровану функцію NLM-модуля як віддалену процедуру, передаючи їй вихідні дані і отримуючи результати обробки.

6.7. XML - розширена мова розмітки

XML (EXtensible Markup Language) - розширена мова розмітки. Рекомендований Консорціумом Всесвітньої павутини (W3C). Специфікація XML описує XML-документи і частково описує поведінку XML-процесорів (програм, які читають XML-документи і забезпечують доступ до їх вмісту).

XML розроблялася як мова з простим формальним синтаксисом, зручна для створення і обробки документів програмами і одночасно зручний для читання і створення документів людиною, з підкресленням націленості на використання в Інтернеті. Мова називається розширеною, оскільки ній не фіксується розмітка, яка використовується в документах: розробник вільний створити розмітку відповідно до потреб конкретної області, будучи обмеженим лише синтаксичними правилами мови [6.18-6.21].

XML (EXtensible Markup Language) - розширена мова розмітки.

Рекомендований Консорціумом Всесвітньої павутини (W3C).

Специфікація XML описує XML-документи і частково описує поведінку XML-процесорів (програм, які читають XML-документи і забезпечують доступ до їх вмісту).

XML розроблялася як мова з простим формальним синтаксисом, зручна для створення і обробки документів програмами і одночасно зручний для читання і створення документів людиною, з підкресленням націленості на використання в Інтернеті. Мова називається розширеною, оскільки ній не фіксується розмітка, яка використовується в документах: розробник вільний створити розмітку відповідно до потреб конкретної області, будучи обмеженим лише синтаксичними правилами мови.

Специфікація XML описує мову і ряд питань, що стосуються кодування та обробки документів. Матеріал цієї секції є скорочений виклад опису мови в Специфікації XML, адаптоване для цієї статті.

Нормативним вважається англійський варіант документа, тому основні терміни наводяться з їх англійськими оригіналами.

Переведення основних термінів в основному слід доступному в інтернеті перекладу специфікації на російську мову, виняток становлять терміни tag і declaration. Для терміна tag тут використовується переклад тег. Для терміна declaration віддано перевагу поширеній перекладу оголошення (проти також поширеною кальки декларація).

У літературі та інтернеті можуть зустрічатися і інші переклади основних термінів. З фізичної точки зору документ складається з сутностей (Entities), з яких кожна може відсилати на іншу сутність. Єдиний кореневий елемент - документная сутність. Зміст сутностей - символи.

З логічної точки зору документ складається з коментарів (Comments), оголошень (Declarations), елементів (Elements), посилань на сутності (Character references) та інструкцій обробки (Processing instructions). Все це в документі структурували розміткою (Markup).

Сутність - дрібна частина в документі. Всі сутності що-небудь містять, і у

всіх них є ім'я (існують винятки, напр. Документна сутність). Простіше кажучи, термін «сутність» описує «сущу річ», «щось».

Документ складається з сутностей, зміст яких - символи. Всі вони розділені на два типи: символьні дані (Character data) і розмітки.

До розмітці належать: теги (Tags), що позначають межі елементів, оголошення та інструкції обробки, включаючи їх атрибути (Attributes), посилання на сутності, коментарі, а також послідовності символів, що обрамляють секції «CDATA».

Частина документа, яка не належить розмітці, становить символьні дані документа. Всі складові частини документа узагальнюються в пролог і кореневий елемент.

Кореневий елемент - обов'язкова частина документа, складова всю його суть (пролог, взагалі кажучи, може бути відсутнім). Може включати (а може не включати) вкладені в нього елементи і символьні дані, а також коментарі.

Вкладені в кореневий елемент елементи, в свою чергу, можуть включати вкладені в них елементи, символьні дані і коментарі, і так далі.

Пролог може включати оголошення, інструкції обробки, коментарі. Його слід починати з оголошення XML, хоча в певній ситуації допускається відсутність цього оголошення.

Елементи документа повинні бути правильно вкладені: будь-який елемент, що починається всередині іншого елемента (тобто будь-який елемент документа, крім кореневого), повинен закінчуватися усередині елемента, в якому він почався.

Символьні дані можуть зустрічатися всередині елементів як безпосередньо так і в спеціальних секціях «CDATA».

Оголошення, інструкції обробки і елементи можуть мати пов'язані з ними атрибути. Атрибути використовуються для зв'язування з логічною одиницею тексту пар ім'я-значення.

Розмітка завжди починається символом <і закінчується символом >.

Поряд з символами <i>, спеціальну роль для розмітки грає також символ &. Кутові дужки позначають межі елементів, інструкцій обробки і деяких інших послідовностей. Амперсанд дозволяє виконати заміну тексту за допомогою сутностей (Entities).

Рішення проблеми неоднозначності розмітки. Вживання розмічальних символів в символьних даних ускладнює розпізнавання конструкцій розмітки і може створити проблему неоднозначності структури. У XML ця проблема вирішується таким чином: <i & не можуть бути присутніми в символьних даних і в значеннях атрибутів в їх безпосередньому вигляді, для їх подання в цих випадках зарезервовані спеціальні суті:

Таблиця 6.14. Правило заміни символів

Символ	Заміна
<	<
>	>
&	&

Крім того, для вживання апострофів і лапок усередині значень атрибутів використовуються наступні сутності: '--' "----"

Правило заміни символів, використовуваних в розмітці, на ними позначаються суті не поширюється на символьні дані в секціях «CDATA», зате виконується у всіх інших місцях документа [6.18-6.21].

Числові посилання на символи вказують кодову позицію символу в наборі символів документа. Числові посилання на символи можуть приймати дві форми:

1. синтаксис «**&#D;**», где D — десяткове число;
2. синтаксис «**&#xH;**» или «**&#XH;**», де H — шістнадцяткове число (шістнадцяткові числа в числових символьних посиланнях не чутливі до регістру).

Приклади числових посилань на символи:

- **å** — (в десятковій формі) являє букву «а» з маленьким

кружком над нею (використовується, наприклад, в норвезькому мовою);

- **å** — (в шістнадцятковому) являє собою той же символ;
- **å** — (в шістнадцятковому) також представляє той же символ;
- **И** — (в десяткового формі) являє велику літеру кирилиці «І»;
- **水** — (в шістнадцятковому) представляє китайський символ для води;

У мові XML все імена повинні починатися з літери, символу підкреслення (_) або двокрапки (:) і тривати лише допустимими для імен символами, а саме вони можуть містити тільки букви, що входять в секцію букв кодування Unicode, арабські цифри, дефіси, знаки підкреслення , точки і двокрапки.

Однак імена не можуть починатися з рядка xml в будь-якому регістрі. Імена, що починаються з цих символів, зарезервовані для використання консорціумом W3C. Потрібно пам'ятати, що так як букви не обмежені виключно символами ASCII, то в іменах можна використовувати слова з рідної мови. Оголошення *XML* вказує версію мови, на якій написаний документ. Оскільки інтерпретація вмісту документа залежить від версії мови, то Специфікація наказує починати документ з оголошення *XML*.

У першій (1,0) версії мови використання оголошення не було обов'язковим, в наступних версіях воно обов'язково. Таким чином, версія мови визначається з оголошення, і якщо оголошення відсутня, то приймається версія 1.0. Крім версії *XML*, оголошення може також містити інформацію про кодування документа і «залишатися документом зі своїм власним *DTD*, або з підключеним».

Приклад:

<?xml version="1.1" encoding="UTF-8" ?>

або:

<?xml version="1.0" encoding="windows-1251"?>

Во всех цих прикладах відсутній атрибут «standalone», який як раз і визначає, підключити чи документу опис розмітки зовні. По замовчуванню він дорівнює «no»:

<?xml version="1.0" encoding="windows-1251" standalone="no"?>

якщо XML-документ посилається на інші DTD-файли, які описують, що документ може вміщувати, ви повинні вказати standalone="no"

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

якщо XML-документ не посилається на інші файли і буде користуватися своїм DTD, ви повинні вказати standalone="yes"

Для об’яви типа документа існує спеціальна інструкція !DOCTYPE. Вона дозволяє задавати за допомогою мови DTD, які в документ входять елементи, які їх атрибути, які сутності можуть використовуватися й будь-які ще.

Наприклад, вот коректний документ:

<?xml version="1.0"?>

<greeting>Hello, world!</greeting>

В ньому є кореневий елемент <greeting>Hello, world!</greeting>, і з логічної точки зору документ існує. Однак він недійсний (not valid).

За допомогою об’яви типа документа (DTD) мозливо описати його зміст й логічну структуру [6.18-6.21]., а також зв’язувати з визначеним елементом пару «ім’я — значення». Вот як виглядає пролог в запису Бекуса — Наура:

Об’ява типа документа:

```
prolog          ::= XMLDecl? Misc* (doctypedecl Misc*)?
XMLDecl         ::= '<?xml' VersionInfo EncodingDecl? SDDecl? S? '?>'
VersionInfo     ::= S 'version' Eq ("'" VersionNum "'" | '"' VersionNum '"')
Eq              ::= S? '=' S?
VersionNum      ::= '1.' [0-9]+
Misc            ::= Comment | PI | S
doctypedecl     ::= '<!DOCTYPE' S Name (S ExternalID)? S? ('[' intSubset ']' S?)? '>'
DeclSep         ::= PEReference | S
intSubset       ::= (markupdecl | DeclSep)*
markupdecl      ::= elementdecl | AttlistDecl | EntityDecl | NotationDecl | PI | Comment
extSubset       ::= TextDecl? extSubsetDecl
extSubsetDecl   ::= ( markupdecl | conditionalSect | DeclSep)*
```

Після XML-оголошення можуть слідувати коментарі, інструкції обробки

або ж порожні простору, але потім йде Оголошення типу документа, де «Name» - ім'я кореневого тега, «ExternalID» - зовнішній ідентифікатор, а «intSubset» - оголошення розмітки або ж посилання на сутність. Як свідчить специфікація, якщо зовнішній ідентифікатор оголошується разом з внутрішнім оголошенням, то останнім йде перед першим.

Наприклад:

<?xml version="1.0"?>

<!DOCTYPE greeting SYSTEM "hello.dtd">

<greeting>Hello, world!</greeting>

Тут «SYSTEM" hello.dtd "» - зовнішній ідентифікатор: адреса «hello.dtd» дозволяє задіяти дані в документі «hello.dtd» як оголошення розмітки.

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE greeting [

  <!ELEMENT greeting (#PCDATA)>

]>

<greeting>Hello, world!</greeting>

Тут же розмітка була оголошена місцево.

нструкції обробки (Processing instruction, PI), дозволяють розміщувати в документі інструкції для додатків. У наступному прикладі показана інструкція обробки, передає xml-stylesheet-додатком (наприклад, браузеру) інструкції в файлі mystyle.css за допомогою атрибута href: <? Xml-stylesheet type = "text / css" href = "mystyle.css"? >

<?xml-stylesheet href="my-style.css"?>

Коментарі (Comment) не належать до символьним даним документа. Коментар починається послідовністю «<! -» і закінчується послідовністю «->», всередині не може зустрічатися комбінація символів «-». Символ & не використовується всередині коментаря в якості розмітки.

Елемент (Element) є поняттям логічної структури документа. Кожен документ містить один або кілька елементів. Межі елементів представлені

початковим і кінцевим тегами. Ім'я елемента в початковому і кінцевому тегах елемента має збігатися. Елемент може бути також представлений тегом порожнього, тобто не включає в себе інші елементи і символьні дані, елемента.

Тег (Tag) - конструкція розмітки, яка містить ім'я елемента.

Початковий тег: <element1>

Кінцевий тег: </ element1>

Тег порожнього елемента: <empty_element1 />

В елементі атрибути можуть використовуватися тільки в початковому тегу і тезі порожнього елемента.

Використання:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE recipe>
<recipe name="хлеб" preptime="5min" cooktime="180min">
  <title>
    Простой хлеб
  </title>
  <composition>
    <ingredient amount="3" unit="стакан">Мука</ingredient>
    <ingredient amount="0.25" unit="грамм">Дрожжи</ingredient>
    <ingredient amount="1.5" unit="стакан">Тёплая вода</ingredient>
  </composition>
  <instructions>
   <step>
     Смешать все ингредиенты и тщательно замесить.
   </step>
   <step>
     Закрыть тканью и оставить на один час в тёплом помещении.
   </step>
   <!--
```

```
      <step>
        Почитать вчерашнюю газету.
      </step>
       - это сомнительный шаг...
     -->
    <step>
      Замесить ещё раз, положить на противень и поставить в духовку.
    </step>
  </instructions>
</recipe>
```

Секція CDATA не є логічною одиницею тексту.

Секція може зустрічатися в будь-якому місці документа, де синтаксис дозволяє розміщувати символьні дані.

Секція починається <! [CDATA [і завершується]]>. Між цією розміткою знаходяться символьні дані, символьні дані при цьому включають символи <> & в їхній безпосередній формі.

Коректний (Well-formed) документ відповідає всім загальним правилам синтаксису XML, які можуть застосовуватися до будь-якого XML-документу: правильна структура документа, збіг імен в початковому і кінцевому тезі елемента і т. п.

Документ, який неправильно побудований, не може вважатися документом XML.

Кодування документів. Специфікація вимагає, щоб обробні програми підтримували принаймні два кодування Юнікод: UTF-8 і UTF-16.

XML-процесор і додаток. Специфікація XML визначає поняття XML-процесор і додаток. XML-процесор (парсер) - програма, що аналізує розмітку і передає інформацію про структуру документа іншій програмі - з додатком.

Специфікація XML накладає певні вимоги на процесор, не торкаючись вимог до додатка.

Регламентація роботи з документами: правила, мови, програмні інтерфейси.

Дійсний документ. Перевіряючі і неперевіряючі процесори. Документ є дійсним, якщо з ним пов'язано оголошення типу документа і якщо цей документ відповідає представленим в оголошенні типу обмеженням.

XML-процесори діляться на два класи: перевіряючі і неперевіряючі. Перевіряючі процесори перевіряють дійсність документа і повинні повідомляти (за вибором користувача) про порушення обмежень, сформульованих в оголошенні типу документа [6.20-6.21].

Неперевіряючі процесори не перевіряють дійсність документа, але обов'язки по попередній обробці документа, згадані вище, залишаються за ними.

Опис типів: мови схем. Для опису типів документів використовуються мови схем (Schema language). Оскільки XML є підмножиною мови SGML, то він успадкував розроблений для SGML мову Document Type Definition (DTD). Пізніше були розроблені і інші мови схем, найбільш відомі з яких XML Schema, RELAX NG.

Перетворення документа XML. Для вирішення завдання перетворення документа XML в іншу схему або інший формат призначений мову XSLT.

Формат для візуалізації документа. Для форматованого документа (документа, підготовленого до візуалізації) призначений формат XSL-FO.

Мови запитів. XPath - синтаксис для адресації вмісту документа, представленого у формі дерева. Вирази XPath використовуються в мові XQuery. Вирази XPath, взагалі кажучи, можуть використовуватися в будь-якому контексті, де доречно використовувати формальні посилання на елементи дерева, зокрема, в якості параметрів для методів інтерфейсів доступу до документа. [6.1, 6.19 - 6.21].

XQuery - мова програмування, орієнтований на роботу з документами.

Читання XML: три варіанти API. Подієвий API (event-driven API, push-

style API) - XML-процесор читає XML; при певному подію (появі відкриває або закриває тега, текстового рядка, атрибута) викликається callback-функція.

• + Витрачає мало пам'яті

• + При обробці величезного XML є стандартною точкою, що дозволяє миттєво зупинити обробник.

• - Вкрай складний для прикладного програміста: доводиться тримати в пам'яті інформацію, в якому місці документа ми знаходимося.

• + Бібліотека проста в програмуванні.

• - Утруднена підтримка перехресних посилань: треба організувати тимчасове зберігання строкових посилань, а коли документ буде лічений - перетворити ідентифікатори в покажчики.

• - При помилку в XML в пам'яті залишається полусозданная структура предметної галузі; програміст повинен своїми руками коректно знищити її.

• - API тільки для читання, для запису потрібно інший API.

• ± Природний вибір, коли з величезного XML треба витягти трохи даних

• ± Природний вибір, коли XML треба перетворити в структуру предметної галузі.

• Приклади бібліотек: SAX, Expat.

Об'єктний API (Document Object Model, DOM, «об'єктна модель документа») - зчитує XML і відтворює його в пам'яті у вигляді об'єктної структури.

• - Витрачає багато пам'яті - набагато більше, ніж сам XML займає на диску. На pugixml витрата пам'яті втричі і більше перевищує довжину XML.

• + Простий для прикладного програміста.

• + Бібліотека проста в програмуванні.

• + Найчастіше вдається розпізнати «майже вірні» XML з переплутаним порядком тегів.

• + Дозволяє довільний доступ до XML. Це, наприклад, спрощує роботу з перехресними посиланнями.

• + При помилку в XML в пам'яті залишається полусозданная структура XML, яка буде автоматично знищена самою бібліотекою.

• + Загальний API для читання і запису.

• ± Природний вибір, коли об'єктом предметної області є сам XML: наприклад, в веб-браузері, XML-редакторі, в імпортері до програми-локалізатор, який витягує рядки з XML довільної структури.

• ± Природний вибір, коли потрібно завантажити XML, злегка переробити і зберегти. Ті частини, які чіпати не потрібно, не вимагають ніякого коду.

• Приклади бібліотек: JDOM, TinyXML, pugixml

Бувають і гібридні API: зовнішні і незначні частини читаються потоковим методом, а внутрішні і важливі - об'єктним.

API прямого запису записує XML тег за тегом, атрибут за атрибутом.

• + Швидкий, немає проміжних об'єктів.

• - Примітивна бібліотека може робити неоптимальний XML (наприклад, <tag> </ tag> замість <tag />). Працююча оптимально набагато складніше в програмуванні.

• - Непридатний для окремих специфічних завдань.

• - Якщо структури предметної області працюють ненадійно, без спеціальних заходів (записати в пам'ять або в інший файл, потім перейменувати) можна залишитися з «впала» програмою і втраченим файлом.

• - При помилку програміста може вийти синтаксично некоректне XML.

• - API тільки для запису, для читання потрібно інший API.

Об'єктний API, він же Document Object Model.

• - Створює об'єктну структуру для XML, що може відняти пам'яті більше, ніж структура предметної галузі.

• ± Універсальний (втім, в більшості завдань переваги над добре опрацьованим API прямого запису немає - на відміну від читання).

• + Навіть якщо структури предметної області працюють ненадійно, а

програміст не передбачив жодної «захисту», єдиний сценарій, коли файл перезаписується на неповний - нестача місця на диску.

• + Загальний API для запису і читання.

• Приклади бібліотек: ті ж, що і для читання XML методом DOM.

XML - мова розмітки, іншими словами, засіб опису документа. Саме в ніші документів, текстів, де частка різнотипних символьних даних велика, а частка розмітки мала - XML успішний.

З іншого боку, обмін даними у відкритих системах не зводиться до обміну документами.

Надмірність розмітки XML (а в цілях розробки мови прямо вказано, що лаконічність не є пріоритетом проекту) позначається в ситуаціях, коли дані не вписуються в традиційну модель документи.

Стрічка новин, наприклад, що оформляється з використанням синтаксису XML (формати RSS, Atom), являє собою не документ в традиційному розумінні, а потік однотипних міні-документів - багатослівна і надлишкова розмітка в цьому випадку становить істотну частину переданих даних.

W3C стурбований ефективністю застосування XML, і відповідні робочі групи займаються цією проблемою

Інша ситуація, коли формати XML можуть виявитися не кращим рішенням - робота з даними з простою структурою і невеликим за обсягом змістом полів даних.

У цьому випадку частка розмітки в загальному обсязі велика, а програмна обробка XML може виявитися невиправдано витратною, в порівнянні з роботою з даними більш простої структури. У цій області розробники розглядають засоби, спочатку орієнтовані на дані, такі як INI, YAML, JSON.

ЛІТЕРАТУРА

6.1. http://docwiki.cisco.com/wiki/AppleTalk#AppleTalk_Filing_Protocol " AFP performs functions at the presentation and application layers of the AppleTalk protocol suite.

6.2. Network Protocols Handbook, 2005, ISBN 0-9740945-2-8, page 300-301, Figure 2-26

6.3. Service Name and Transport Protocol Port Number Registry — IANA.

6.4. New OS X uses Windows file sharing by default. AFP is out, SMB2 is in—but whose version of SMB2? And what about Time Machine? 2013

6.5. AppleShare & AppleShare IP File Sharing: Chart of All Limitations.

6.6. Mac OS 8, 9: Mac OS Extended Format - Volume and File Limits.

6.7. Mac OS X Server 10.2: Tested and theoretical maximums (limits).

6.8. Mac OS X Server 10.3: Tested and theoretical maximums (limits).

6.9. Apple's Developer documentation on AFP Version Differences.

6.10. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципи, технологии, протоколы: Учебник для вузов. 3-е изд.– СПб.: Питер, 2006. – 958 с.

6.11. http://tools.ietf.org/html/draft-peck-suiteb-dtls-srtp-02 Suite B Profile for Datagram Transport Layer Security / Secure Real-time Transport Protocol (DTLS-SRTP)

6.12. As of version 1.0.2. *The OpenSSL Project*. The OpenSSL Project (22 января 2015).

6.13. OpenSSL: News, ChangeLog

6.14. NSS 3.14 release notes. *Mozilla Developer Network*. Mozilla.

6.15. An update is available that adds support for DTLS in Windows 7 SP1 and Windows Server 2008 R2 SP1. Microsoft.

6.16. libsystools – a TLS/DTLS open source library for Windows/Linux using OpenSSL

6.17. *Дмитрий Цветцих*. Secure UDP communications using DTLS in pure

js.GitHub.

6.18. *Дмитрий Цветцих*. DTLS in pure js. npm.

6.19. *Mobius Software LTD*. Non blocking Java DTLS Implementation based on BouncyCastle and Netty. Mobius Software LTD.

6.20. *Sean DuBois*. pion/dtls: DTLS 1.2 Server/Client implementation for Go. GitHub.

6.21. californium/scandium: DTLS 1.2 Server/Client implementation for java and coap. Includes connection id extension.. Eclipse Foundation.

7. АЛГОРИТМИ СТИСНЕННЯ ПРЕДСТАВНИЦЬКОГО РІВНЯ

7.1. Алгоритм створення черезстрокового зображення - Adam7

Adam7 - алгоритм створення черезстрокового зображення, специфічний для формату PNG [7.1]. Черезрядкові зображення розбиваються на сім менших, накладаючи на кожну ділянку 8x8 наступну маску (рис. 7.1):

```
1 6 4 6 2 6 4 6
7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6
7 7 7 7 7 7 7 7
3 6 4 6 3 6 4 6
7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6
7 7 7 7 7 7 7 7
```

Рис. 7.1 Маска для алгоритму стиснення Adam7

Числа відповідають номеру маленького зображення, куди потрапить піксель.

У цьому алгоритмі використовується сім проходів в двох вимірах, на відміну від подібних алгоритмів, використовуваних в GIF і використовують тільки чотири проходу по вертикалі. Це дає більш високу швидкість розгортки за менше число операцій, ніж в алгоритмі бикубической інтерполяції.

Adam7 отримав свою назву на честь Адама Костелло (Adam M. Costello), який запропонував цей метод 30 січня 1995 року, розширивши існував алгоритм з п'ятьма проходами Лі Даніеля Крокера (Lee Daniel Crocker). [7.2 - 7.4].

Asymmetric numeral systems (ANS, від «асиметричні системи числення») - сімейство методів ентропійного кодування, винайдених Ярославом Дудою в 2006 на основі введеної ним концепції асиметричних систем числення.

З 2014 року використовується для стиснення даних в ряді програм, так як ці методи за ступенем стиснення дають приблизно настільки ж гарне акуратне наближення до оптимального ентропійне кодування, як і арифметичне кодування, але мають вищу швидкодію, не уступаючи по швидкості

розпакування алгоритмам кодування Хаффмана; крім того, суттєвим є те, що ці методи не захищені патентами і вільні до використання, так як створення і поширення вільної альтернативи арифметичному кодування було метою автора.

Асиметричні системи числення є узагальненням позиційних систем числення, при яких різні символи можуть кодуватися різною кількістю цифр в залежності від попередніх цифр (символів).

В обчислювальній техніці прийнято представляти інформацію у вигляді потоку бітів, і додавання нової інформації - символу - вдає із себе приписування до числа в кінці відповідних коду символу цифр - нових молодших розрядів. При підході зі звичайними позиційними системами числення, будь-якому символу відповідає однакову кількість цифр. Це добре підходить в разі, коли ймовірність зустріти різні символи однакова.

Коли ймовірності зустріти різні символи розрізняються, для більш компактного запису інформації використовується ентропійне кодування. Так, в кодуванні Хафмана різні символи можуть записуватися різною кількістю бітів. Однак при цьому символи кодуються цілим кількістю біт - що, зокрема, означає, що як би часто не зустрічався б символ, для його кодування буде потрібно не менше одного біта.

В асиметричних системах числення кодування символу залежить не тільки від того, що це за символ, але і від попереднього контексту, що відбивається станом. Кількість потрібних цифр залишається цілим, але воно змінюється і може бути навіть нульовим.

Brotli - алгоритм стиснення даних з відкритим вихідним кодом, розроблений Юрки Алакуйяла (Jyrki Alakuijala) і Золтаном Сабадка.

Метод стиснення brotli заснований на сучасному варіанті алгоритму LZ77, Ентропійно кодуванні Хаффмана і моделюванні контексту 2-го порядку.

Призначений для прискорення завантаження веб-сторінок, підтримується в браузерах Chrome, заснованих на Chromium і в Firefox.

Як і zopfli, інший алгоритм стиснення від Google, brotli був названий на

честь швейцарського хлібобулочного виробу, brötli.

Алгоритм brotli був вперше представлений в 2015 році як спеціалізований алгоритму стиснення веб-шрифтів. [7.9]

У вересні 2015 року інженери Google представили версію brotli, що містила поліпшення для універсального стиснення даних без втрат, особливо для використання при стисканні інтернет-трафіку. Алгоритм і реалізація були перероблені для поліпшення ступеня стиснення і прискорення операцій стиснення і розпаковування. Був доопрацьований API роботи з потоками, з'явилися більш високі рівні стиснення, зменшилося споживання пам'яті. [7.8]

На відміну від більшості універсальних алгоритмів стиснення, brotli поставляється з вбудованим 120-кілобайтний словником. Цей словник містить близько 13 тисяч рядків, фраз і інших послідовностей, часто зустрічалися у великому корпусі текстових і HTML-документів [7.10, 7.11]. Подібна особливість дозволяє збільшити ступінь стиснення для деяких коротких файлів.

У порівнянні з класичним алгоритмом deflate (середина 1990-х, ZIP, gzip), brotli, як правило, досягає на 20% вищий ступінь стиснення для текстових файлів, зберігаючи подібну швидкість стиснення і розпаковування. Стислі за допомогою brotli потоки отримали тип кодування br.

На відміну від zopfli, алгоритм brotli не є назад сумісним з zlib і deflate.

7.2. bzip2 - безкоштовна вільна утиліта командного рядка з відкритим вихідним кодом для стиснення даних

bzip2 - безкоштовна вільна утиліта командного рядка з відкритим вихідним кодом для стиснення даних, реалізація алгоритму Барроуза - Уилера.

Розроблено і вперше опублікована Джуліаном Сюард (Julian Seward) в липні 1996 року (версія 0.15). Стабільність і популярність компресора росли протягом кількох років, і версія 1.0 була опублікована в кінці 2000 року.

Ефективність. Відповідно до традицій UNIX, bzip2 одноразово може

виконувати тільки одну операцію: або стиснення, або розпакування і тільки для одного файлу. При стисненні bzip2 додає до імені файлу розширення «.bz2». Для упаковки декількох файлів їх спершу архівують в один файл утилітою tar, а потім стискають за допомогою bzip2. Такі архіви зазвичай мають розширення «.tar.bz2». bzip2 стискає більшість файлів ефективніше, але повільніше, ніж більш традиційні утиліти gzipірілі zip. В цьому відношенні він схожий на інші сучасні алгоритми стиснення. [7.1-7.3].

bzip2 виконує стиснення даних з суттєвим навантаженням на CPU (що обумовлено його математичним апаратом). bzip2 застосовують, якщо немає обмежень на час стиснення і на навантаження на CPU, наприклад, для разової упаковки великого обсягу даних.

У деяких випадках bzip2 поступається за ефективністю стиснення архіваторам 7-Zip (метод стиснення LZMA) і rar. Згідно з даними автора програми від 2005 року, метод стиснення bzip2 поступається за ефективністю стиснення на 10 15% [3] найкращим методам, відомим на той момент (PPM) [4], але при цьому в 2 рази швидше при стисненні і в 6 разів швидше при розпакуванні.

Метод стиснення *bzip2* працює наступним чином:

• *стиснені дані діляться на блоки фіксованого розміру;*

• *виконується перетворення Барроуза - Уилера для перетворення послідовностей багаторазово чергуються символів в рядки однакових символів;*

• *застосовує перетворення MTF;*

• *використовується кодування Хаффмана.*

Приблизний розмір блоку можна вибрати за допомогою аргументів командного рядка («-1» для 100 кілобайт, «-2» для 200 КБ, ..., «-9» для 900 КБ). Кожен блок стискається незалежно, стислі блоки записуються послідовно один за одним, на початку кожного використовується 48-бітна послідовність - магічне число 0x314159265359 (в кодуванні ASCII при вирівнюванні на кордон байта відображається як «1AY & SY»), тобто запис перших десяткових цифр

числа π в форматі BCD [7.5]. Кінець файлу позначається 48-бітної константою 0x177245385090, що представляє собою корінь з числа Пі. На початку файлів формату bzip2 використовується наступний заголовок: двухбайтовая сигнатура «BZ», потім вказівка на метод ентропійного стиснення - «h» (Хаффман) і розмір блоку (десяткове число від 0 до 9).

За рахунок використання незалежного стиснення окремих блоків можливі реалізації формату з паралельним стисненням або розпакуванням (для розпакування може знадобитися індекс зсувів для кожного блоку) [7.6].

Приклади використання bzip2

- *# Команда для сжатия файла «file»*
- bzip2 file
- *# или*
- bzip2 --compress file
-
- *# Команда для распаковки файла «file.bz2»*
- bzip2 -d file.bz2
- *# или*
- bzip2 --decompress file.bz2
- *# или*
- bunzip2 file.bz2
- *# bunzip2 - копия bzip2 или ссылка на bzip2*

Версія *GNU tar* підтримує прапор *«-j» («--bzip2»),* який дозволяє створювати і розпаковувати файли *«tar.bz2»* без використання перенаправлень вводу-виводу (*Pipeline*). приклад:

 # Упаковка данных в архив tar и сжатие bzip2 при помощи GNU tar

tar -cvjf file.tar.bz2 list_of_files

 # или

tar --create --verbose --bzip2 --file file.tar.bz2 list_of_files

 # Распаковка архива tar, сжатого bzip2 при помощи GNU tar

tar -xvjf file.tar.bz2

 # или

tar --extract --verbose --bzip2 --file file.tar.bz2

Сучасні версії *GNU* tar можуть автоматично визначити метод стиснення даних, і тому прапор *«-j» («--bzip2»)* можна не використовувати. приклад:

tar -xvf file.tar.bz2

 # або

tar --extract --verbose --file file.tar.bz2

Крім того, існує набір утиліт для виконання пошуку, виведення, відновлення і порівняння даних в форматі *bzip2*:

• *bzcat* - розпакування даних і висновок на термінал;

• *bzmore, bzless* - розпакування даних і посторінковий вивід на термінал;

• *bzcmp* - розпакування двох файлів, порівняння вмісту і повідомлення результату: «дорівнює» або «не дорівнює»;

• *bzdiff* - розпакування двох файлів, порівняння вмісту і висновок відмінностей;

• *bzgrep, bzegrep, bzfgrep* - розпакування даних і пошук в розпакованому;

• *bzip2recover* - розпакування будь-яких блоків, які тільки можна розпакувати.

Архів «*.bz2*» містить потік (*Stream*) стислих даних. Слово «потік» вживається, так як дані можна розділити логічно і блоки даних стискуються незалежно один від одного. Стислі дані складаються з наступних полів:

• заголовок розміром 4 байта;

• нуль або більше блоків стислих даних різного розміру;

• маркер, що позначає кінець стислих даних і контрольна сума (*CRC*) розміром 32 біта, обчислена для всього потоку;

• кілька невикористовуваних біт для доповнення розміру потоку до цілого кількості байт.

7.3. Алгоритм стиснення CABAC, CTW, Deflate, DMC, FLIF, gzip

Контекстно-адаптивне двійкове арифметичне кодування (Кадак; **CABAC** від Context-adaptive binary arithmetic coding) - форма ентропійного (статистичного) кодування, яке використовується в відеокодек стандарту *H.264 / MPEG-4 AVC*. Використовується техніка стиснення без потерьдля отримання більш високого ступеня стиснення, ніж більшість алгоритмів, які доступні в кодуванні відео [7.1-7.3].

Є одним з основних переваг кодека *H.264 / AVC. CABAC* підтримується тільки в основному (*Main*) і більш високих профілях кодека, а також вимагає витрачати чималу кількість робочих циклів процесора в чисто програмної реалізації, як з точки зору циклів, так і з точки зору потужності системи для декодування (перегляду) відео, закодованого з використанням цієї технології. Також, важкий в векторизації і розпаралелюванні. Існує контекстно-адаптивне нерівномірне кодування (*Context-adaptive variable-length coding, CAVLC*), більш низько ефективні схема статистичного кодування, яка використовується для підвищення продуктивності на більш слабких системах декодування.

Кадак має кілька режимів передбачення для різного контексту. Спочатку конвертуються все небінарние символи в бінарні; далі, для кожного біта кодек вибирає, яку модель передбачення використовувати; після цього він використовує отриману від найближчих елементів інформацію для оптимізації ступеня можливості передбачень.

Арифметичне кодування є фінальним кроком стиснення даних.

CTW (Context Tree Weighting - Зважування контекстного дерева) - алгоритм передбачення і стиснення без втрат, створений Willems, Shtarkov, and Tjalkens (1995).

CTW є одним з небагатьох алгоритмів, які забезпечують як хороші теоретичні показники, так і добре показують себе на практиці.

При оцінці ймовірності символу алгоритм CTW з певною вагою змішує

статистику передбачень багатьох моделей Маркова різного порядку, кожна з яких створюється на основі умовних імовірнісних оцінок нульового порядку.

Deflate - це алгоритм стиснення без втрат, що використовує комбінацію алгоритмів LZ77 і Хаффмана.

Спочатку був описаний Філом Кацем для другої версії його архіватора PKZIP, який згодом був визначений в RFC 1951 (1996 рік).

Deflate вважається вільним від всіх існуючих патентів, і поки залишався в силі патент на LZW (він застосовується в форматі GIF), це призвело до використання Deflate не тільки в форматі ZIP, для якого Кац спочатку його спроектував, але також в компресорі / декомпресор gzip і в PNG-зображеннях.

Deflate-потік містить серії блоків. Перед кожним блоком знаходиться трёхбітовий заголовок:

• Один біт: прапор останнього блоку.

• 1: блок останній.

• 0: блок не останній.

• Два біта: метод, за допомогою якого були закодовані дані.

• 00: дані не закодовані (в блоці знаходяться безпосередньо вихідні дані).

• 01: дані закодовані за методом статичного Хаффмана.

• 10: дані закодовані за методом динамічного Хаффмана.

• 11: зарезервоване значення (помилка).

Велика частина блоків кодується за допомогою методу 10 (динамічний Хаффман), який надає оптимізоване дерево кодів Хаффмана для кожного нового блоку. Інструкції для створення дерева кодів Хаффмана йдуть безпосередньо за заголовком блоку.

Компресія виконується в два етапи:

• заміна повторюваних рядків покажчиками (алгоритм LZ77);

• заміна символів новими символами, грунтуючись на частоті їх використання (алгоритм Хаффмана).

DMC *(Dynamic Markov compression, динамічне Марківське стиснення*

[7.1]) - алгоритм стиснення даних без втрат, розроблений *Горданом Кормаком (Gordon Cormack) і Найджелом Хокспулом (Nigel Horspool)* [7.2].

Метод побудований аналогічно методу *PPM*: сам алгоритм є предиктором (розраховує ймовірності символів), а безпосереднє стиск проводиться ентропійним кодувальником. На відміну від *PPM*, метод *DMC* як правило працює на рівні бітів, тоді як *PPM* - на рівні байтів. *DMC* забезпечує зіставні з *PPM* рівні стиснення і швидкість обробки, але вимагає більше пам'яті і не так поширений, як *PPM*. Деякими з сучасних реалізацій є: компресор hook від Наніі Франческо Антоніо (*Nania Francesco Antonio*), компресор o*camyd* від Франка Швеллінгера (*Frank Schwellinger*), також *DMC* використовується в якості однієї з моделей в компресорі *Мета Матони (Matt Mahoney) paq8l*.

Всі перераховані компресори засновані на оригінальній реалізації 1993 року в мові *C* від Гордона Кормака. DMC пророкує і кодує один біт за один логічний крок роботи. Він відрізняється від PPM тим, що працює на рівні бітів, а не байтів. Відмінність від методів змішування контекстів (наприклад, PAQ) полягає в тому, що DMC будує і використовує одну єдину модель джерела. Безпосереднє стиснення здійснюється за допомогою арифметичного кодування.

Модель джерела пророкує наступний біт на підставі поточного контексту. Модель може бути представлена у вигляді графа, кожен вузол якого містить в собі два лічильника: n0 і n1, де n0 - лічильник бітів зі значенням 0, і n1 - лічильник бітів зі значенням 1. Один з вузлів завжди є поточним. Один з лічильників в поточному вузлі збільшується, коли у вихідних даних зустрічається біт з відповідним значенням. Також вузол має два ребра, що зв'язують його з іншими вузлами або з самим собою. Одне з ребер використовується, коли у вихідних даних зустрічається 0, інше - коли 1.

У найпростішому випадку модель складається з одного вузла, що посилається на самого себе. У даній конфігурації модель може тільки вважати співвідношення кількості бітів зі значенням 0 до кількості бітів зі значенням 1 у вихідних даних. Коли ж в моделі стає більше одного вузла, то при обробці

чергового біта може відбутися перехід до іншого вузла, а також утворення нового вузла.

Передбачення наступного біта здійснюється розрахунком ймовірностей для 0 за формулою p0 = n0 / n = n0 / (n0 + n1) і, відповідно, для 1 по формулі p1 = 1 - p0 = n1 / n. Таким чином, кожен вузол втілює окремий стан моделі, в якому вона робить різні передбачення. Контекст в моделі DMC не запам'ятовується явно, але він враховується моделлю в результаті переходів між вузлами графа станів. Моделювання здійснюється однаково і при стисненні і при декомпресії.

Free Lossless Image Format (FLIF) - вільний формат стиснення зображень без втрати якості, який, по завіреннях розробників, за ступенем стиснення перевершує PNG, lossless WebP, lossless BPG і lossless JPEG 2000 [7.2]. В даний час стандартизований і документований. FLIF16 Він доступний за ліцензією LGPL. FLIF використовує один з варіантів арифметичного кодування (MANIAC, Meta-Adaptive Near-zero Integer Arithmetic Coding) в якості алгоритму ентропійного стиснення.

Файл містить 4 частини: два заголовка, метадані та піксельні дані в моделі RGB з глибиною до 16 біт. Підтримується анімація. За замовчуванням формат несе черезстрокову запис зображення, алгорітмічно схожу з Adam7.

gzip (скорочення від GNU Zip) - утиліта стиснення і відновлення (декомпресії) файлів, що використовує алгоритм Deflate.

Застосовується в основному в UNIX-системах, в ряді яких є стандартом де-факто для стиснення даних.

Була створена Жан-Лу Галлі (Jean-Loup Gailly) і Марком Адлером (Mark Adler). Версія 0.1 була випущена 31 жовтня 1992 року, а версія 1.0 - в лютому 1993 року.

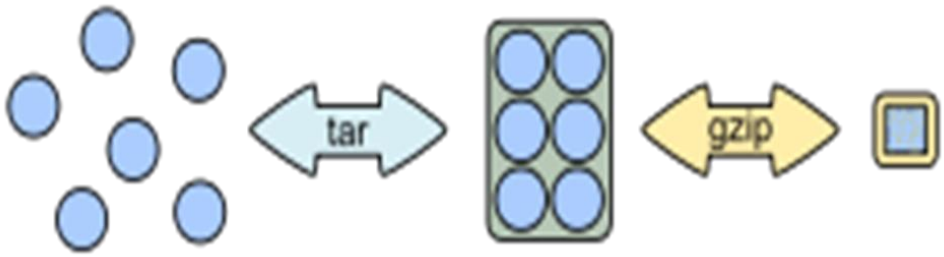


Рис. 7.2 Схема роботи з архівом .tar.gz з декількома файлами

Відповідно до традицій UNIX-програмування, gzip виконує тільки дві функції: стиснення і розпаковування одного файлу; упаковка декількох файлів в один архів неможлива. При стисненні до оригінального розширення файлу додається суфікс .gz. Для упаковки декількох файлів зазвичай їх спочатку архівують (об'єднують) в один файл утилітою tar, а потім цей файл стискають за допомогою gzip. Таким чином, стислі архіви зазвичай мають подвійне розширення .tar.gz, або скорочене .tgz.

З іншого боку, зазначена особливість дає gzip можливість працювати з безперервним потоком даних, упаковуючи / розпаковуючи їх «на льоту». Це широко застосовується в UNIX-системах: за допомогою перенаправлення потоків можна працювати з упакованими файлами так само легко, як і з розпакованими (розпаковуючи їх в пам'яті при читанні і упаковуючи при запису); багато UNIX-утиліти мають вбудовану підтримку цього механізму. Останнім часом gzip активно застосовується для стиснення інтернет-трафіку. Зараз gzip підтримують більшість сучасних браузерів [7.2].

Крім того, існує набір утиліт для пошуку, виведення і порівняння даних в форматі gzip: zcat, zdiff, zfgrep, zless, zcmp, zegrep, zgrep, zmore.

7.4. Алгоритм стиснення LZ4, LZ77, LZ78, LZJB, LZMA, 7-Zip, LZMA2, LZSS

LZ4 - алгоритм стиснення даних без втрат, орієнтований на високу швидкість стиснення і розпаковування. Він відноситься до сімейства методів стиснення LZ77, які працюють з байтовими потоками. Відрізняється компактним кодом для розпакування.Алгоритм LZ4 має трохи менший ступінь стиснення, ніж більш ранній метод стиснення LZO.

LZO, в свою чергу, стискає з меншим ступенем, ніж класичні gzip і DEFLATE. Однак, LZ4 за швидкістю стиснення близький до LZO і в кілька разів швидше gzip'a, а швидкість розпакування у LZ4 значно вище, ніж у LZO.

Стислі дані в методі LZ4 представляються у вигляді послідовності записів. Кожен запис починається з токена - одного байта, розбитого на два 4-бітних поля. Перше поле визначає кількість байтів літеральної послідовності - тобто рядки, яка при розпакуванні буде скопійована в вихідний потік. Друге поле визначає довжину рядка, копируемой з уже розпакованого буфера (зі словника) [7.1-7.3].

Значення 0 в поле відповідає мінімальній довжині збіги в 4 байта. Значення 15 в поле є ознакою використання додаткового байта, значення якого буде додано до довжини. Якщо додатковий байт довжини дорівнює 255, то до поля довжини додається значення ще одного байта, що дозволяє вказувати довільні довжини через серію байтів із значенням 255 (0xff).

Рядок літерала в стислій послідовності слід за токеном і додатковими байтами довжин литерала. Потім записується зміщення збіги у вихідному буфері та додаткові байти довжини збігу.

Додатково можуть використовуватися фрейми, що вказують на розмір даних і містять контрольні суми. Ентропійне кодування (таке, як Код Хаффмана) не застосовується.

Стиснення може здійснюватися над потоком байтів або над

послідовністю блоків. Досягається ступінь стиснення залежить від обсягу роботи, виконуваної для пошуку збігів. Якщо витратити більше часу на стиснення, то буде отримано більше компактний стислий файл, а швидкість його розпакування виросте.

Оригінальна реалізація LZ4 написана на мові програмування Сі Яном Колле (Yann Collet) і поширюється на умовах ліцензії BSD. Існують портовані версії і інтерфейси для безлічі мов, в тому числі Java, C #, Python і т. д. [7.1, 7.6].

Деякі бази даних, наприклад Hadoop, використовують LZ4 завдяки його високій швидкості стиснення. LZ4 також реалізований в складі ядра Linux починаючи з версії 3.11, може застосовуватися для прискорення завантаження. Файлова система ZFS в складі реалізацій FreeBSD, Illumos, «ZFS on Linux» і ZFS-OSX підтримує метод LZ4 для стиснення даних. Ядро Linux підтримує LZ4 для стислих образів ФС SquashFS починаючи з версії 3.19. LZ4 також реалізований в складі архіватора Zstd від Яна Колле.

LZ77 і LZ78 - алгоритми стиснення без втрат, опубліковані в статтях ізраїльських математиків Авраама Лемпеля і Яакова Зіва в 1977 і 1978 роках. Ці алгоритми - найбільш відомі варіанти в сімействі LZ *, яке включає в себе також LZW, LZSS, LZMA і інші алгоритми. [7.6-7.13].

Обидва алгоритми відносяться до словниковим методам, на відміну від інших методів зменшення надмірності, таких як RLE і арифметичне стиснення. LZ77 є алгоритмом з «ковзним вікном», що еквівалентно неявному використання словникового підходу, вперше запропонованого в LZ78.

Для розуміння даного алгоритму необхідно розібратися з двома його складовими: принципом ковзаючого вікна і механізмом кодування збігів.

Принцип ковзного вікна. Метод кодування, відповідно до принципу ковзного вікна, враховує вже раніше зустрічалося інформацію, тобто інформацію, яка вже відома для кодувальника і декодувальник (друге і подальші входження деякої рядка символів в повідомленні замінюються посиланнями на її перше входження).

Завдяки цьому принципу алгоритми LZ * іноді називаються методами стиснення на безперервній основі вікна. Ковзне вікно можна представити у вигляді буфера (або більш складної динамічної структури даних), який організований так, щоб запам'ятовувати «сказану» раніше інформацію і надавати до неї доступ.

Таким чином, сам процес стискає кодування згідно LZ77 нагадує написання програми, команди якої дозволяють звертатися до елементів «ковзного вікна», і замість значень сжимаемой послідовності вставляти посилання на ці значення в «ковзному вікні». Розмір ковзного вікна може динамічно змінюватися і складати 2, 4 або 32 кілобайт. Слід також зазначити, що розмір вікна кодувальника може бути менше або дорівнює розміру вікна декодувальник, але не навпаки.

Наведене вище порівняння процесу кодування з «програмуванням» може наштовхнути на передчасний висновок про те, що алгоритм LZ77 відноситься до методів контекстного моделювання. Тому слід зазначити, що алгоритм LZ77 прийнято класифікувати як метод словникового сжатіяданних, коли замість поняття «ковзного вікна» використовується термін «динамічного словника».

Механізм кодування збігів. Перед тим, як перейти до розгляду механізму кодування, уточнимо поняття збіги (Match). Розглянемо послідовність з N елементів. Якщо всі елементи послідовності унікальні, то така послідовність не міститимуть жодного повторюваного елемента, або, інакше кажучи, в послідовності бракуватиме хоча б двох рівних один одному або співпадаючих елементів. [7.8 - 7.10].

У стандартному алгоритмі LZ77 збігу кодуються парою:

• довжина збігу (match length)

• зміщення (offset) або дистанція (distance)

Пара, яка кодується трактується саме як команда копіювання символів з ковзаючого вікна з певної позиції, або дослівно як: «Повернутися в буфері символів на значення зсуву і скопіювати значення довжини символів ,

починаючи з поточної позиції ».

Хоча така інтерпретація може здатися інтуїтивно зрозумілою, вона мало говорить про сутність алгоритму LZ77 як методу стиснення. Особливість даного алгоритму стиснення полягає в тому, що використання кодуємої пари довжина-зміщення є не тільки прийнятним, але і ефективним в тих випадках, коли значення довжини перевищує значення зсуву.

Приклад з командою копіювання не зовсім очевидний: «Повернутися на 1 символ назад в буфері і скопіювати 7 символів, починаючи з поточної позиції». Яким чином можна скопіювати 7 символів з буфера, коли зараз в буфері знаходиться тільки 1 символ? Однак наступна інтерпретація кодуючої пари могла б пояснити ситуацію: кожні 7 наступних символів збігаються (еквівалентні) з 1 символом перед ними.

Це означає, що кожен символ можна однозначно визначити, перемістившись назад в буфері - навіть якщо цей символ ще відсутня в буфері на момент декодування поточної пари довжина-зміщення. Така кодується пара представлятиме собою багаторазове (визначається значенням зміщення) повторення послідовності (яка визначається значенням довжини) символів, що являє собою більш загальну форму RLE.

Недоліки:

• неможливість кодування підстрок, віддалених один від одного на відстані, більшій довжини словника

• довжина підрядка, яку можна закодувати, обмежена розміром буфера

• мала ефективність при кодуванні незначного обсягу даних

На відміну від LZ77, що працює з уже отриманими даними, LZ78, запропонований в 1978 році, орієнтується на дані, які тільки будуть отримані (LZ78 не використовує «ковзне» вікно, він зберігає словник з вже переглянутих фраз).

Алгоритм зчитує символи повідомлення до тих пір, поки підстрока, що накопичується, входить цілком в одну з фраз словника. Як тільки цей рядок

перестане відповідати хоча б одній фразі словника, алгоритм генерує код, що складається з індексу рядка в словнику, яка до останнього введеного символу містила вхідні рядок, і символу, який порушив збіг.

Потім в словник додається введена підстрока. Якщо словник вже заповнений, то з нього попередньо видаляють менш всіх використовувану в порівняннях фразу.

LZJB - алгоритм стиснення даних без втрат, винайдений Джефом Бонвіком в 1998 році для стиснення аварійних дампів програм і даних в файлової системі ZFS. Заснований на методі стиснення з використанням словника. [7.1, 7.14].

Цей алгоритм включає безліч виправлень до алгоритму LZRW1, який в свою чергу є варіантом LZRW, що є членом сімейства алгоритмів стиснення Lempel-Ziv. Цей алгоритм націлений на збільшення швидкості стиснення.

У 2012-2013 роках в ZFS розглядалася заміна LZJB на байт-орієнтований LZ4, як більш швидкий метод з трохи кращим стисненням.

Стислий потік в LZJB є байтовим. Для розмітки потоку використовується керуючий байт, що описує типи наступних 8 послідовностей. Кожен біт керуючого байта визначає тип одного елемента. Біт зі значенням 0 відповідає літерально байту: один байт стисненого потоку копіюється у вихідний потік. Біт зі значенням 1 означає посилання в словник («збіг»). Два наступних за ним байта містять 6-бітове поле довжини і 10-бітове поле зміщення (LLLLLLdd dddddddd). Біти довжини декодируются в довжину від 3 до 66 байтів (довжина дорівнює L + 3), потім з вихідного буфера вибирається рядок, що відстоїть на «зсув» (d) байтів назад від поточної позиції.

LZMA (Lempel-Ziv-Markov chain-Algorithm) - алгоритм стиснення даних, що розробляється з 1996 або 1998 року Ігорем Павловим. Використовується в архіваторі 7-Zip того ж автора для створення стислих архівів у форматі 7z [7.15].

Алгоритм заснований на схемі стиснення даних по словнику, подібна до використаної в LZ77, і забезпечує високий коефіцієнт стиснення (зазвичай

перевищує коефіцієнт, що отримується при стисненні з використанням bzip2), а також дозволяє використовувати словники різного розміру (до 4 Гб).

Також lzma - утиліта командного рядка з відкритим кодом для стиснення даних з LZMA SDK, яка працює з файлами, що мають формат і розширення .lzma. Комплект засобів розробки з відкритим вихідним кодом LZMA, написаний на мові C ++, використовує покращений алгоритм стиснення LZ77, доповнений алгоритмом інтервального кодування, а також спеціальними процедурами для обробки двійкових файлів.

LZMA підтримує різні варіанти хеш-ланцюжків, довічних і префіксних дерев в якості основи алгоритмів пошуку по словнику.

LZMA SDK містить також алгоритм *BCJ / BCJ2*, реалізований для процесорів архітектури *x86, ARM, PowerPC, IA-64 і ARM Thumb*. У ньому точки переходу перед стисненням нормалізуються - тобто, наприклад, для x86 це означає, що інструкції ближніх і умовних переходів і виклики функцій перетворюються з форми з відносним зсувом «перейти на тисячу шістсот шістьдесят п'ять байт назад» в форму з абсолютним адресою «перейти до адресою 5554 ».

Алгоритм *BCJ2*, реалізований в *7-Zip*, використовує 32-бітну адресацію. У пакувальників виконуваних файлів UPX адресація залежить від типу архітектури (наприклад, для виконуваних файлів *DOS* використовується 16-бітна адресація) [7.1, 7.16].

Реалізація, яка, починаючи з версії 4.61 beta, переведена з ліцензії *CPL* в категорію суспільного надбання, має такі властивості:

• *Швидкість стиснення: приблизно 1 Мб / с на процесорі x86 з частотою 2 ГГц.*

• *Швидкість вилучення: близько 10-20 Мб / с на процесорі x86 з частотою 2 ГГц.*

• *Підтримка багатопоточності.*

Розмір коду розпакування LZMA становить близько 5 Кб; витрата

динамічної пам'яті залежить від розміру словників. Ці можливості дозволяють реалізувати розпакування на вбудованих системах.

Використання особливостей Microsoft Windows в вихідному коді ускладнює створення версій програми для Unix. Проте, існує дві працездатні портованої версії: в p7zip більш-менш портіровані версії утиліт командного рядка 7z і 7za для POSIX-систем (GNU / Linux, Solaris, OpenBSD, FreeBSD, Cygwin і інших), Mac OS X і BeOS.

Також є офіційна портіруємість реалізації - LZMA Utils, призначена для створення потокових компресорів, подібних gzip. З 2008 року вона починає все частіше використовуватися в системах управління пакетами - зокрема, dpkg та RPM.

7-Zip використовує досить гнучкий формат архіву, його підтримують і деякі сторонні утиліти (наприклад, читання 7z підтримує WinRAR). [7.1, 7.17].

Також існує порт 7-Zip для Mac OS X, який називається Compress, в даний час являє собою досить недопрацьований інструмент. Для Mac OS X існують ще збірки p7zip і 7zX.

Для роботи з LZMA автор надає свій кроссплаформенний SDK, що володіє перерахованими вище властивостями. Основна частина SDK написана на C ++ і спочатку поширювалася на умовах GNU LGPL. Варто відзначити кілька моментів:

• З версії 4.57 LZMA SDK надає також ANSI C-продажу як алгоритму розпакування, так і алгоритму компресії, що розширює сферу застосування SDK і спрощує використання у вбудованих системах і інших обмежених середовищах (наприклад, в ядрах операційних систем).

• З версії 4.62 LZMA SDK став доступний на умовах Public Domain, тобто допускається його використання для будь-яких цілей без будь-яких обмежень.

Деякі мережеві пристрої (на зразок US Robotics 9105 і 9106) в якості вбудованого використовують модифікований Linux, що завантажується зі стислою файлової системи. В якості алгоритму стиснення файлової системи

замість Zlib використовується алгоритм LZMA. Як правило, такий файлової системою є squashfs з LZMA-патчем.

Крім цього, LZMA використовується в реалізаціях UEFI як один з алгоритмів стиснення.

LZMA2 - нова версія алгоритму LZMA. Даний алгоритм має наступні переваги перед алгоритмом LZMA:

• вихідний потік може містити одночасно стиснені та стислі дані (нестискувані дані записуються як є, що економить біти).

• найкраща підтримка багатопоточності при компресії і декомпресії.

Lzop - це вільне програмне забезпечення для стиснення файлів, яке використовує алгоритм LZO. Програма є вільним програмним забезпеченням і розповсюджується під ліцензією GNU General Public License [7.1, 7.18].

Lzop був розроблений для наступних цілей:

• швидка швидкість роботи (як за компресії, так і декомпресії);

• можливість використовувати замість gzip;

•портативність.

Відмінності від GZIP:

Основні відмінності між Lzop і GZIP полягають в наступному:

• Не видаляє вихідні файли. Ця опція встановлена за замовчуванням;

• файли, стиснуті за допомогою lzop, матимуть розширення '.lzo';

• швидкість стиснення і розпаковування набагато вище, ніж у gzip, але ступінь стиснення гірше.

LZSS (Lempel-Ziv-Storer-Szymanski, Лемпеля-Зів-Сторер-Шиманський) - алгоритм стиснення даних без втрат, похідний від методу LZ77. Створено в 1982 році Джеймс Сторер і Томасом Шиманським. LZSS був описаний в статті «Data compression via textual substitution» (стиснення даних через текстові підстановки) [7.1, 7.19].

LZSS є словниковим методом стиснення. Він намагається замінити повторно зустрінуті послідовності символів на посилання в словник.

Основна різниця між вихідним LZ77 і LZSS полягає в тому, що в методі LZ77 запис посилання на словник може бути довшим, ніж рядок, яку вона заміщає (тобто запис такого посилання робить стислий фрагмент довше, ніж нестислий). У методі LZSS подібні посилання опускаються, в разі, якщо довжина рядка менше деякої настройки («break even»). Більш того, LZSS застосовує однобітний прапор для позначення того, чи є наступний фрагмент стисненого потоку літералом (байтом) або посиланням в словник (парою значень довжина і зміщення). [7.1, 7.19].

Перехід на початок (Move-to-front, MTF) - перетворення для кодування даних (зазвичай потоку байтів), розроблене для поліпшення продуктивності ентропійного кодування.

При гарній реалізації воно досить швидко для включення як додатковий крок в алгорітмахсжатія даних. Також може використовуватися спільно з BWT, перетворенням Барроуза – Уилера.

Основною ідеєю перетворення є заміна кожного вхідного символу його номером в спеціальному стеці недавно використаних символів. Послідовності ідентичних символів, наприклад, будуть замінені (починаючи з другого символу) на послідовність нулів. Якщо ж символ довго не з'являвся у вхідній послідовності, він буде замінений великим числом. Перетворення замінює послідовність вхідних символів на послідовність цілих чисел, якщо у вхідних даних було багато локальних кореляцій, то серед цих чисел будуть переважати невеликі, краще стискувані ентропійним кодуванням, ніж вихідні дані.

Спочатку алгоритм називався «стопка книг» («book stack»).

Часто використовується при перетворенні байтів. Спочатку кожне можливе значення байта записується в список, в клітинку з номером, рівним значенню байта, тобто (0, 1, 2, 3, ..., 255). В процесі обробки даних цей список змінюється.

Перший оброблений символ замінюється самим собою, після чого елемент, що відповідає цьому символу, переміщається в голову списку

(зрушуючи елементи з 0 по своє становище на 1 вправо).

Наступні символи кодуються номером елемента, що містить їх значення.

Після кодування кожного символу ці елементи також просуваються до голови списку.

Microsoft Point-to-Point Compression (MPPC) - протокол стиснення даних, спочатку розроблений для використання поверх з'єднань PPP. Використовує алгоритм Lempel-Ziv зі змінним вікном буфера історії розмір блоку 8192 байт.

7.5. PAQ - серія вільних архіваторів з текстовим інтерфейсом та версії

PAQ - серія вільних архіваторів з текстовим інтерфейсом, які спільними зусиллями розробників піднялися в перші місця рейтингів багатьох тестів стиснення даних (хоча і ціною процесорного часу і обсягу пам'яті). Кращий результат в цій серії на більшості тестів було отримано архіватором PAQ8JD, створеним спільними зусиллями Метта Махоні (Matt Mahoney), Олександра Ратушняка, Сергія Оснача, Пшемислава Скібіньского (Przemysław Skibiński) і Білла Петтіс (Bill Pettis), і випущеним 30 грудня 2006 року. Однак в деяких тестах він відстає від WinRK (створеного Малькома Тейлором в січні 2005 року) в режимі PWCM. PWCM (PAQ weighted context mixing, «PAQ зважене змішування контекстів») - стороння пропріетарна реалізація алгоритму PAQ. Спеціально налаштовані версії алгоритму PAQ виграли призи в Приз Хаттер і Калгарі Корпус Челлендж [7.1-7.3, 7.20].

В основі алгоритму лежить ідея контекстного моделювання. Контекст – це історія появи символу, тобто інформація про символи, що передують поточному в стисливому потоці.

При цьому процес компресії розбивається на дві фази: моделювання і кодування. PAQ використовує алгоритм змішування контекстів.

Змішування контекстів - це техніка, тісно пов'язана з алгоритмом PPM, але відмінність полягає в тому, що ймовірність появи наступного символу

обчислюється на основі зваженої комбінації великого числа моделей, що залежать від різних контекстів, не обов'язково слідують один за одним.

У PAQ сімействі для збору статистики і передбачення ймовірності наступного символу використовуються в основному такі моделі:

• n-грами - контекст; попередні n байт (як в PPM).

• словникові n-грами, що ігнорують регістр і неалфавітні символи (корисні в текстових даних).

• «розріджені» контексти, наприклад, другий і четвертий символи перед кодуються (корисні в деяких бінарних форматах).

• «аналогові» контексти, що складаються з верхньої половини двійкового представлення 8- або 16-бітових слів (корисні в мультимедійних форматах даних).

• двовимірні контексти (корисні для зображень, табличних даних). Довжина ряду визначається знаходженням повторюваних патернів байт.

• спеціалізовані моделі, такі, як x86-виконувані файли або Windows Bitmap, TIFF, JPEG-зображення. Ці моделі активуються, коли даний тип файлу визначається.

Всі версії PAQ пророкують і стискають за раз один біт, але розрізняються в деталях реалізації того, як передбачення комбінуються і обробляються після. Як тільки самий корінь була визначена ймовірність появи наступного біта, біт стискається арифметичним кодером. Існує три способи для комбінування прогнозів моделей в залежності від версії PAQ [7.1, 7.20].

• від PAQ1 до PAQ3 кожне пророцтво представлено парою бітових лічильників. Ці лічильники комбінувалися зваженим підсумовуванням, таким, що більшу вагу визначається довшим контекстом.

• в PAQ4 до PAQ6 передбачення комбінувалися, як і в першому випадку, але ваги, що належать кожній моделі, призначалися так, щоб більш точні моделі отримували перевагу.

• в PAQ7 і пізніших версіях вихід кожної моделі є ймовірність, на відміну

від пари лічильників, ймовірності підсумовуються за допомогою штучних нейронних мереж.

PAQ1SSE і пізніші версії використовували пост обробку передбачення методом вторинної оцінки символу (SSE - Secondary Symbol Estimation), тобто комбіноване пророкування і невеликий контекст використовувалися для вибору наступного передбачення по таблиці. Після того, як символ був закодований, дані в таблиці коректувалися для зменшення помилки передбачення. Вторинна оцінка символу може бути об'єднана в один процес з різними контекстами або може виконуватися паралельно, усереднити з виходами моделей. [7.1, 7.21].

Рядок s стискається в байтовую рядок, що представляє число x в 256-значної системі числення big endian в інтервалі [0; 1] таке, що P (r <s) ≤ x <P (r ≤ s), де P (r < s) - ймовірність, що випадкова рядок r такої ж довжини, як s, лексикографічно менше, ніж s. Завжди можна знайти такий рядок x, що довжина її хоча б на один байт перевершує ліміт Шеннона: -log2 P (r = s) біт. Довжина s зберігається в заголовку архіву.

Арифметичний кодер в PAQ реалізований шляхом зберігання верхньої та нижньої кордонів x для кожного кроку передбачення, початково [0; 1]. Після кожного передбачення поточний інтервал ділиться пропорційно можливостям того, що наступний біт буде 0 і наступний біт буде 1, на підставі попередніх бітів s. Наступний біт кодується, вибираючи відповідний підінтервал як новий інтервал.

Число x декомпресивні в рядок s ідентичною серією бітових пророкувань (так як попередні біти s відомі). Інтервал ділиться як при стисненні. Частина, що містить x, стає новим інтервалом, і відповідний біт додається до s.

У PAQ верхня і нижня межі інтервалу представляються трьома частинами. Найбільш значущі розряди по підставі 256 ідентичні, так вони можуть бути записані як старші байти x. Наступні 4 байта зберігаються в пам'яті так, що старший байт різниться. Молодші біти маються на увазі всі нулями для нижньої межі і все одиницями для верхньої межі. Кодування завершується

записом ще одного байта з нижньої межі [7.1, 7.21].

Кожна модель розділяє вхідний потік біт s на безліч контекстів і відображає кожен контекст на стан історії бітів, представлене 8 бітами. У версіях до PAQ6 стан було представлено парою лічильників (n0, n1). У PAQ7 і більш пізніх стан містило при певних умовах також останній біт або цілу послідовність. Значення станів відображаються в ймовірності, використовуючи 256-значну таблицю. Після табличного передбачення значення в таблиці трохи вирівнювалося (зазвичай до 0,4%) для зменшення похибки передбачення.

У всіх PAQ8-версіях стану історії бітів містять наступну інформацію:

• Точна послідовність до 4 бітів.

• Пара лічильників і індикатор для останнього біта для послідовностей від 5 до 15 біт.

• Пара лічильників для послідовностей від 16 до 41 біта.

Щоб зберегти кількість станів рівним 256, такі обмеження були накладені на лічильники: (41, 0), (40, 1), (12, 2), (5, 3), (4, 4), (3, 5), (2, 12), (1, 40), (0, 41). Якщо лічильник перевищує ліміт, наступний стан вибирається з подібним співвідношенням n0 / n1. Таким чином, якщо поточний стан (n0 = 4, n1 = 4, останній біт = 0) і 1 отримана на вході, тоді новий стан ні (n0 = 4, n1 = 5, останній біт = 1), а (n0 = 3 , n1 = 4, останній біт = 1).

Більшість контекстних моделей реалізовані як хеш-таблиці. Деякі невеликі контексти реалізовані як індексні масиви.

Попередня обробка тексту. Деякі версії PAQ, особливо PAsQDAa, PAQAR (обидві походять від PAQ6), і від PAQ8HP1 до PAQ8HP8 (нащадки PAQ8 і здобули верх в Призі Хаттер) обробляють текст, переглядаючи його і замінюючи слова з тексту, що містяться в зовнішньому словнику, одно- і трёхбайтнимі кодами. Додатково слова в верхньому регістрі кодуються спеціальним символом і перекладом слова в нижній регістр. У PAQ8HP-серії словник організований угрупованням синтаксично і семантично схожих слів разом. Це дозволяє використовувати моделі, які використовують тільки верхні

біти словникових кодів як контекстів.

Найбільш значущі зміни до алгоритму PAQ. [7.1, 7.21-7.27].

■ • **PAQ1** був випущений 6 січня 2002 року *Меттом Махоні*. Він використовував фіксовані ваги і не включав розріджені і аналогові моделі. Всього використовувалося 5 моделей.

■ • **PAQ1SSE / PAQ2** був випущений 11 травня 2003 року *Сергієм Оснач*. Він значно поліпшив стиснення додаванням вторинної оцінки символу між провісником і кодувальником. Вторинна оцінка символу подавала на вхід невеликої контекст і поточний прогноз, і на виході виходило нове пророцтво з таблиці. Табличне значення потім оновлювалося для відображення поточного біта.

■ • **PAQ3N** був випущений 9 жовтня 2003 року Було додано розріджена модель.

■ • **PAQ4**, випущений 15 листопада 2003 *Меттом Махоні*, використовував адаптивне зважування. *PAQ5 (18 грудня 2003) і PAQ6 (30 грудня 2003)* були незначними поліпшеннями, що включають аналогову модель. До цього часу *PAQ* конкурував з кращими PPM-компресорами і привернув увагу спільноти людей, що займаються стисненням даних, що призвело до численних поліпшень до квітня 2004 року *Берто Дестасіо* доводив моделі і поправив порядок пересування між лічильників. *Йохан де Бок* (*Johan de Bock*) вніс поліпшення в інтерфейс користувача. *Девід А. Скотт* поліпшив арифметичний кодер. *Фабіо Буффон* прискорив програму.

• У період з 20 травня 2004 по 27 липня 2004 Олександр Ратушняк випустив сім версій архиватора PAQAR, в якому ступінь стиснення була значно підвищена шляхом додавання багатьох нових моделей, численних міксерів з вибором ваги по контексту, додаванням вторинної оцінки сімволана вихід кожного міксера і, нарешті , додаванням попередньої обробки виконуваних файлів архітектури процесорів Intel. PAQAR залишався на вершині програм стиснення даних без втрат до кінця 2004 року, але був набагато повільніше

своїх попередників.

• З 18 січня по 7 лютого 2005 року Пшемислав Скібіньскій випустив чотири версії PAsQDa, що базувалися на PAQ6 і PAQAR і доповнені англійською словниковим препроцесором. Він досяг найкращого результату на Калгарі Корпусі, але не на більшості інших тестів.

• Модифікована Меттом Махоні версія PAQ6 взяла приз на Калгарі Корпус Челлендж 10 січня 2004 року. Ця подія перекрилося десятьма послідовними версіями PAQAR Олександра Ратушняка. Найбільш пізня побачила світ 5 червня 2006 року, вона складалася з стислих разом даних і тексту програми і займала 589 862 байти.

• *PAQ7* був випущений в грудні 2005 року *Меттом Махоні*. *PAQ7* - це повністю переписаний *PAQ6* і його варіанти (*PAQAR, PAsQDa*). Ступінь стиснення була схожа з *PAQAR,* але час виконання - в 3 рази менше. Але йому не вистачало x86 і словника, тому він був не такий гарний для стиснення виконуваних модулів *Microsoft Windows* і англійських текстів, як *PAsQDa.* Хоча він включав моделі для кольорових *BMP, TIFF* і *JPEG*-файлів, тому стискав їх краще. Головна відмінність *PAQ7* було в тому, що він використовував нейронну мережу для комбінування моделей, на відміну від зменшує градієнт міксера. Інший рисою *PAQ7* була можливість стискати вбудовані у файли *Excel, Word і PDF* зображення *Bitmap і JPEG*.

• *PAQ8A* був випущений 27 січня 2006 і *PAQ8C* 13 лютого. Це був експериментальний предвипуск очікуваного *PAQ8.* Він виправляв деякі компромісні рішення в *PAQ7,* зокрема, слабке стиснення в деяких випадках. *PAQ8A* також включав в себе моделі для x86-виконуваних файлів.

• PAQ8F був випущений 28 лютого 2006 року. PAQ8F містив три поліпшення в порівнянні з PAQ8A: більш ефективне використання пам'яті в контекстної моделі, нову непряму контекстну модель і новий інтерфейс користувача для підтримки технології drag-n-drop під Windows. Він не містив англійського словника, як PAQ8B / C / D / E варіанти.

• PAQ8G був випущений 3 березня 2006 року Пшемиславом Скібіньскім. PAQ8G - це PAQ8F, але зі словниками та переробленої моделлю препроцесора текстових даних, що не покращувала стиснення на нетекстових файлах.

• PAQ8H з'явився 22 березня 2006 року завдяки Олександру Ратушняку і був оновлений 24 березня 2006 року. PAQ8H був поліпшенням PAQ8G в деяких моделях.

• Павло Л. Голобородько випустив PAQ8I 18 серпня 2006 року, з виправленням помилок 24 серпня, 4 вересня, і 13 вересня. Він містив додавання моделі напівтонових чорно-білих зображень для PGM-файлів.

• Білл Петтіс випустив PAQ8J 13 листопада 2006 року. Програма базувалася на PAQ8F з деякими поліпшеннями текстової моделі, запозиченими з PAQ8HP5. Хоча вона не включала в себе словники з PAQ8G або PGM-моделі з PAQ8I.

• Серж Оснач випустив серію поліпшень моделі: PAQ8JA - 16 листопада 2006 року, PAQ8JB - 21 листопада і PAQ8JC - 28 листопада.

• PAQ8JD побачив світ 30 грудня 2006 року стараннями Білла Петтіс. Програма була перенесена на Win32 і 32- і 64-бітну платформу Linux.

• Білл Петтіс справив PAQ8K 13 лютого 2007 року. У нього були додані додаткові моделі для бінарних файлів.

• PAQ8L народився 13 березня 2007 року. Модель для динамічного марковского стиснення була додана до існуючому.

Великий текстовий тест (Large Text Compression Benchmark, LTCB) Метта Махоні ранжує програми по стиснутому розміру доступного публічно файлу розміром 109 байт англійського тексту Вікіпедії. На відміну від інших тестів, він включає в розмір стисненого файлу розмір декомпресора і будь-яких необхідних для стиснення файлів в якості zip-архіву. Станом на 9 лютого 2007 року, PAQ8HP8 був першим з 62 програм.

PAQ дуже вимогливий до ресурсів пам'яті і процесорного часу. Наступна таблиця 44 порівнює час стиснення та розпакування на машині Athlon-64 2,2

гігагерца, а також витрата пам'яті в Мегабайтах для деяких популярних програм з цього тесту.

Таблиця 7.1. Великий текстовий тест

Ранг	Програма	Розмір стисненрго файла	Час стиснення	Пам'ять
1	PAQ8HP8	133 423 109	64 639 секунд	1849 МБ
18	PPMd	183 976 014	880 секунд	256 МБ
44	Bzip2	254 007 875	379 секунд	8 МБ
49	InfoZIP	322 649 703	104 секунди	0,1 МБ

PAQ - це вільне програмне забезпечення і поширюється на умовах GNU General Public License. Це дозволяє іншим авторам зробити форк PAQ і вносити такі зміни, як графічний інтерфейс користувача, або поліпшити швидкість стиснення за рахунок коефіцієнта компресії. Найбільш відомі PAQ-клони:

• WinUDA 0.291, базується на PAQ6, але швидше;

• UDA 0.301 грунтується на алгоритмі PAQ8I;

• KGB Archiver в основному PAQ6v2 з графічним інтерфейсом (бета-версія підтримує PAQ7-алгоритм стиснення);

• Emilcont на основі PAQ6;

• PeaZip - графічна оболонка (для Microsoft Windows і GNU / Linux) для PAQ8F, PAQ8JD і PAQ8L.

7.6. PNG - растровий формат зберігання графічної інформації

PNG (portable network graphics) - растровий формат зберігання графічної інформації, що використовує стиснення без втрат за алгоритмом Deflate.

PNG був створений як вільний формат для заміни GIF, тому в Інтернеті з'явився рекурсивний акронім «PNG is Not GIF» (PNG - НЕ GIF). [7.1, 7.27].

Формат PNG спроектований для заміни застарілого і більш простого

формату GIF, а також, в деякій мірі, для заміни значно складнішого формату TIFF. Формат PNG позиціонується передусім для використання в Інтернеті і редагування графіки.

PNG підтримує три основних типи растрових зображень:

• Півтонове зображення (з глибиною кольору 16 біт)

• Кольорове індексовані зображення (палітра 8 біт для кольору глибиною 24 біт)

• Повнокольорове зображення (з глибиною кольору 48 біт)

Формат PNG зберігає графічну інформацію в стислому вигляді. Причому це стиснення проводиться без втрат, на відміну, наприклад, від JPEG з втратами.

Він має наступні основні переваги перед GIF:

• практично необмежену кількість квітів в зображенні (GIF використовує в кращому разі 8-бітний колір);

• опціональна підтримка альфа-каналу;

• можливість гамма-корекції;

• двовимірна черезстрокова розгортка;

• можливість розширення формату для користувача блоками (на цьому заснований, зокрема, APNG).

Формат GIF був розроблений фірмою CompuServe в 1987 році і спочатку був недоступний для вільного використання. До закінчення в 2004 році дії патентів на алгоритм стиснення LZW, що належали Unisys і використовуваних в GIF, його застосування у вільному програмному забезпеченні було утруднено. На даний момент такі труднощі зняті. PNG ж з самого початку використовує відкритий, непатентований алгоритм стиснення Deflate, безкоштовні реалізації якого доступні в Інтернеті.

Цей же алгоритм використовують багато програм компресії даних, в тому числі PKZIP і gzip (GNU zip).

Формат PNG має більш високим ступенем стиснення для файлів з великою кількістю квітів, ніж GIF, але різниця складає близько 5-25%, що

недостатньо для абсолютне панування формату, так як невеликі 2-16-кольорові файли формат GIF стискає з не меншою ефективністю [7.1, 7.28].

PNG є хорошим форматом для редагування зображень, навіть для зберігання проміжних стадій редагування, так як відновлення і перезберігання зображення проходять без втрат в якості. Також, на відміну, наприклад, від TIFF, специфікація PNG не дозволяє авторам реалізацій вибирати, які можливості вони збираються реалізувати. Тому будь-яке збережене зображення PNG може бути прочитано в будь-якому іншому додатку, що підтримує PNG.

Різні реалізації алгоритму Deflate дають різну ступінь стиснення, тому були створені програми для стискання зображень з декількома варіантами з метою отримання найкращого стиснення - наприклад, OptiPNG і advpng з комплекту AdvanceCOMP (використовує 7-Zip).

7.7. Словникові алгоритми стиснення даних та версії *ROLZ*

ROLZ *(Reduced Offset LZ - алгоритм Лемпела-Зива з скороченими зсувами)* - словниковий алгоритм стиснення даних, близький до *LZ77,* але використовує деякі контекстні прийоми для зменшення числа активних зсувів. Саме поняття *ROLZ* вперше ввів *Malcolm Taylor* в своєму архіваторі *RK* в 1999 році і цей алгоритм є одним з найбільш сучасних підходів до побудови швидких ефективних алгоритмів стиснення [7.1, 7.26-7.29].

У стандартному *LZ77* збігу кодуються парою:

• довжина збігу (*Match length*)

• зміщення (*Offset*) або дистанція (*Distance*)

Проблема цієї схеми в тому, що при кодуванні є надмірність. Наприклад, при розмірі словника в 4 Кбайт є 4096 варіантів зміщення. Зрозуміло, що більшість зсувів не буде використана, і якщо з 4096 варіантів використовується, наприклад, тільки 512, то для кодування зміщення досить 9 біт замість 12 (скорочення на 25%).

Багатьма авторами робилася спроба скоротити можливі значення зсувів, серед них можна відзначити:

LZFG-C2 Автори: *Edward R. Fiala, Daniel H. Greene, 1989 рік.*

Збіги кодуються *HE* парою [довжина, зміщення], а спеціальним індексом, відповідним певної рядку в словнику. Іншими словами, однакові фрази мають однаковий індекс, за рахунок чого і забезпечується економне кодування збігів.

LZRW4 Автор: *Ross Williams, 1991 рік.*

По суті *LZRW4* аналогічний *ROLZ.* Хоча автором і не робилося створення повноцінної програми, в наведеному демонстраційному компресорі можна бачити схему *ROLZ* в її чорновому варіанті.

LZP1-LZP4 Автор: *Charles Bloom, 1995 рік.*

LZP - алгоритм словникового стиснення, який при кодуванні збіги обходиться без зсувів зовсім. Це можливо завдяки тому, що зміщення щодо поточного контексту запам'ятовуються в спеціальній таблиці і компресор з декомпресор оперують цією таблицею однаковим чином.

LZ77-PM Автори: *Dzung T. Hoang, Philip M. Long, Jeffrey Scott Vitter, 1995 рік.*

Цей алгоритм схожий на *ROLZ,* з тією відмінністю, що для скорочення активних зсувів використовуються контексти змінної довжини замість контекстів фіксованого порядку.

За описом автора цей алгоритм являє собою швидкий алгоритм *Лемпела-Зива* з великим словником, який здатний охоплювати великі дистанції в словнику одночасно швидко і ефективно.

Слід зазначити, що *RK* є комерційним архіватором з закритим вихідним кодом і багато деталей використаних алгоритмів не були розкриті. Але завдяки окремим людям таємниця були прочинені і навіть було написано кілька безкоштовних програм на даному алгоритмі стиснення.

Отже, для скорочення активних зсувів використовується контекст фіксованого порядку. В оригіналі це контекст першого порядку (тобто один

символ, що передує поточному символу), також можливе використання інших контекстів - скажімо, контексту другого порядку (два символи, що передують поточному) і т. д.

Замість того, щоб шукати збіги для всіх зсувів в словнику, обмежимося пошуком тільки від тих зсувів, перед якими був присутній поточний контекст. У найпростішому випадку можна використовувати якусь таблицю зсувів:

```
// найдём самое длинное совпадение

context = buf[position - 1]; // текущий контекст первого порядка

max_match = 0; // длина совпадения для кодирования
index = 0; // индекс совпадения (match index)

for (i = 0; i < TAB_SIZE; i++) { // для каждого сохранённого смещения в таблице для данного контекста
  offset = tab[context][i]; // найдем смещение

  match_length = get_match(offset); // найдем длину совпадения

  if (match_length > max_match) { // найдено более длинное совпадение
    max_match = match_length;
    index = i; // сохраним индекс
  }
 }
// обновим таблицу

for (i = TAB_SIZE - 1; i > 0; i--) // сначала сдвинем все смещения вверх, удалив самое старое
```

```
  tab[context][i] = tab[context][i - 1];

tab[context][0] = position; // затем добавим текущее смещение

// закодируем литерал или совпадение

if (max_match >= MIN_MATCH) {
  encode_match_length(max_match); // закодируем длину совпадения
  encode_match_index(index); // закодируем индекс совпадения
  position += max_match;
}
else { // совпадения нужной длины не найдено
  encode_literal(buf[position]); // закодируем литерал
  position++;
}
```

Це найпростіший спосіб. На практиці перебір, скажімо, 1024 зсувів на кожному кроці може зайняти занадто багато часу. Для прискорення пошуку може бути використано хешування і різні структури для швидкого пошуку, що застосовуються в широко поширених реалізаціях алгоритмів сімейства LZ77.

В оригінальному ROLZ літерали кодуються з використанням контекстної моделі першого порядку і це не випадково. Справа в тому, що дана схема кодує більше число літералів, якщо порівнювати зі стандартним LZ77, так як дуже короткі збіги будуть просто не займані ROLZ схемою. Наприклад, при використанні контексту першого порядку і при мінімальній довжині збіги в три символи актуальна довжина мінімального збігу буде дорівнює чотирьом (1 (контекст) + 3 (мінімальна довжина збігу) = 4). Контекстна модель першого порядку або більш складна модель PPM 1-0 при кодуванні літералів здатна компенсувати цей недолік алгоритму. Автор: *Malcolm Taylor, 2005 рік.*

Ці алгоритми являють собою подальший розвиток *ROLZ* [7.3-7.6]:

ROLZ2 - був розроблений з метою забезпечення максимальної швидкості розпакування. ROLZ3 - для максимального стиснення, при незначній втраті в швидкості розпакування [7.1, 7.27-7.30]

Обидва алгоритми для скорочення активних зсувів використовують контекст першого порядку, також вони здатні використовувати таблицю розміром до 32 000 зсувів для кожного контексту.

• ***ROLZ2*** для кодування літералов використовує просту і швидку модель першого порядку.

• ***ROLZ3*** використовує більш складну *CM (Context Mixing)* модель другого порядку.

Але головною відмінною рисою цих алгоритмів є використання оптимального розбору при кодуванні. ***Динамічне програмування*** (*Dynamic Programming*) - прийом, застосовуваний тут, здатний прораховувати оптимальну послідовність літералів / збігів на багато кроків вперед, враховуючи при виборі реальну вартість кодування літерала або збігу.

Zopfli - програмне забезпечення для стиснення даних, що кодує дані в формати DEFLATE, gzip і zlib.

Воно стискає дані з великим коефіцієнтом стиснення, ніж інші реалізації DEFLATE і zlib, але вимагає значно більше часу для створення архіву.

Програмне забезпечення було випущено компанією Google в лютому 2013 року в якості вільної бібліотеки під ліцензією Apache версії 2.0. Назва Zöpfli є швейцарським диминутивом від слова zopf - назви швейцарського хліба.

Zopfli може створювати як чистий потік даних DEFLATE, так і дані DEFLATE, поміщені в формати gzip або zlib. За замовчуванням програма стискає в 15 ітерацій, але за потреби можна на здійснення меншого або більшого числа ітерацій для дотримання балансу між часом і коефіцієнтом стиснення.

При налаштуваннях за замовчуванням, результат роботи Zopfli зазвичай

на 3-8% менше, ніж результат роботи zlib з максимальним доступним коефіцієнтом стиснення, однак стиснення вимагає приблизно в 80 разів більше часу. Час розпакування архівів, створених Zopfli і zlib, практично не відрізняється.

В силу значно більш повільного стиску, zopfli менш придатний для використання для стиснення на ходу і зазвичай використовується для одноразового стиснення статичних даних. Це зокрема істинно для веб-вмісту, що передається за допомогою стискання HTTP на основі DEFLATE, і веб-вмісту в форматах, заснованих на DEFLATE, таких як PNG або WOFF[7.6].

Велика щільність інформації досягається за рахунок більш повних технік стиснення. Метод грунтується на ітеративному моделюванні ентропії і алгоритмах пошуку найкоротшого шляху для пошуку шляху мінімальної бітової вартості в графі всіх можливих DEFLATE-уявлень нестиснених даних.

7.8. Алгоритми стиснення даних без втрат

Zstandard - алгоритм стиснення даних без втрат, що розробляється з 2015 року Яном Колле (Yann Collet) за підтримки Facebook'a; zstd - еталонна реалізація алгоритму Zstandard на мові програмування Сі под вільною ліцензією BSD. Версія 1.0 алгоритму і реалізації були представлені в кінці серпня 2016 года [7.1, 7.25-7.28]. Поєднує словниковий алгоритм стиснення даних типу LZ77 і ефективне ентропійне кодування типу tANS (FSE - Finite State Entropy), алгоритм, схожий з кодом Хаффмана, який реалізує нецілим кількість біт для зберігання символів.

Метою розробки є досягнення коефіцієнтів стиснення, порівнянних або перевершують класичний алгоритм deflate (розроблений в середині 1990-х, реалізований в Zip, gzip і інших) при більш високій швидкості як стиснення, так і розпакування. Подібні цілі вирішують алгоритми Brotli (Google) і LZFSE (Apple, також використовує ANS FSE).

За даними LTCB, zstd 0.6 при максимальному ступені стиснення показує рівень стиснення, близький до архіваторам boz, yxz, tornado; вищий, ніж у lza, brotli, bzip2, забезпечуючи при цьому дуже швидку розпакування (2,2 нс / байт на Core i7-3930K при 4,5 ГГц).

Алгоритм реалізує 22 рівня стиснення, що розрізняються швидкістю і ефективністю (рівень «-1» - найшвидший, рівень «-22» - більш ефективний). Опціонально алгоритм може проаналізувати заданий набір даних для складання спеціалізованих зовнішніх словників. Задається користувачем словник покращує ступінь стиснення подібних файлів, але потрібно при розпакуванні. Словники застосовують для стиснення невеликих файлів, що мають спільні набори рядків, наприклад, xml-файли.

Еталонна реалізація алгоритму на Сі поширюється під вільною ліцензією BSD. Код опублікований на сайті Github. Починаючи з версії 1.3.1, з умов було прибрано згадка обмеженою патентної ліцензії, а код був переліцензувати під умовами подвійний BSD + GPLv2 ліцензії.

Метод Zstandard включений до складу ядра Linux з версією 4.14 від листопада 2017 для використання в файлових системах, зокрема в btrfs і squashfs. Також метод тестувався в ОС FreeBSD для інтеграції в файлову систему OpenZFS.

Алгоритм використовується в ряді дата-центрів і системах обробки «великих даних», зокрема в AWS Red Shift. Підтримується в базах даних, наприклад RocksDB.

Алгоритм стиснення підтримується в архіваторі FreeArc.

Опис методу Zstandard і MIME тип «application / zstd» були подані в IETF як Інтернет-чернетки.

Корпорація Canonical в дистрибутиві Ubuntu планує перевести пакетний формат deb на стиск за допомогою zstd, починаючи з версії 18.10 (жовтень 2018) заради прискорення процесу установки приблизно на 10 відсотків. Стиснення пакетів за допомогою Zstd на рівні 19 призводить до трохи більшого розміру

пакета, ніж при раніше використовувався алгоритмі xz (LZMA), але zstd дозволяє швидше розпаковувати.

Повноцінна реалізація алгоритму з вибором рівня стиснення використовується в форматах .NSZ / .XCZ, розроблені homebrew-спільнотою для гібридної ігрової консолі Nintendo Switch. [7.1, 7.27].

PPM (Prediction by Partial Matching - прогноз щодо часткового збігу) - адаптивний статистичний алгоритм стиснення даних без втрат, заснований на контекстному моделюванні і прогнозі. Модель PPM використовує контекст - безліч символів в стислому потоці, що передують даному, щоб пророкувати значення символу на основі статистичних даних.

Сама модель PPM лише пророкує значення символу, безпосереднє стиск здійснюється алгоритмами ентропійного кодування, як наприклад, алгоритм Хаффмана, арифметичне кодування.

Довжина контексту, який використовується при прогнозі, зазвичай сильно обмежена. Ця довжина позначається n і визначає порядок моделі PPM, що позначається як PPM (n). Необмежені моделі також існують і позначаються просто PPM. Якщо передбачення символу по контексту з n символів не може бути вироблено, то відбувається спроба передбачити його за допомогою n-1 символів. Рекурсивний перехід до моделей з меншим порядком відбувається, поки передбачення не відбудеться в одній з моделей, або коли контекст стане нульової довжини (n = 0).

Моделі ступеня 0 і -1 слід описати особливо. Модель нульового порядку еквівалента нагоди контекстно-вільного моделювання, коли ймовірність символу визначається виключно з частоти його появи в стисливому потоці даних. Подібна модель зазвичай застосовується разом з кодуванням по Хаффману [7.1, 7.31-7.33].

Модель порядку -1 представляють собою статичну модель, присвоюють ймовірності символу певне фіксоване значення; зазвичай все символи, які можуть зустрітися в стисливому потоці даних, при цьому вважаються

равновероятнимі.

Для отримання хорошої оцінки ймовірності символу необхідно враховувати контексти разного довжин.

PPM є варіантом стратегії перемішування, коли оцінки ймовірностей, зроблені на підставі контекстів разного довжин, об'єднуються в одну загальну ймовірність. Отримана оцінка кодується будь-яким ентропійних кодером (ЕК), зазвичай це якась разновідность арифметичного кодера.

На етапі ентропійного кодування і відбувається власне стиснення.

Велике значення для алгоритму PPM має проблема обробки нових символів, ще не зустрічалися у вхідному потоці. Це проблема носить назву проблема нульової частоти. Деякі варіанти реалізацій PPM вважають лічильник нового символу рівним фіксованою величиною, наприклад, одиниці. Інші реалізації, як наприклад, PPM-D, збільшують псевдосчётчік нового символу кожен раз, коли, дійсно, в потоці з'являється новий символ (іншими словами, PPM-D оцінює ймовірність появи нового символу як відношення числа унікальних символів до загальної кількості використовуваних символів).

Опубліковані дослідження алгоритмів сімейства PPM з'явилися в середині 1980-х років. Програмні реалізації не були популярні до 1990-х років, тому як моделі PPM вимагають значної кількості оперативної пам'яті. Сучасні реалізації PPM є одними з кращих серед алгоритмів стиснення без втрат для текстів на природній мові.

Варіанти алгоритму PPM на даний момент широко використовуються, головним чином для компресії надлишкової інформації і текстових даних. Наступні архіватори використовують PPM [7.31-7.33]:

• boa, заснований на PPMz (Ian Sutton)

• HA, PPM order 4, оригінальний метод оцінки ймовірності відходу (Harry Hirvola)

• lgha, заснований на коді архиватора ha (Юрій Ляпко)

• ppmpacktc, заснований на коді PPMd, PPMz, PPMVC і коді HA з

реалізацією hsc (Олександр М'ясников)

• arhangel, заснований на алгоритмах ha з додаванням набору фільтрів для різних даних (Юрій Ляпко)

• PPMd - реалізація PPM order-2..16, застосовується успадкування інформації («дурилка» Дмитра Шкаріна)

• ppmz - реалізований метод Z (Charles Bloom)

• rk - реалізація PPMz з набором фільтрів (Malcolm Taylor)

• rkuc - PPM з порядками 16-12-8-5-3-2-1-0 (Malcolm Taylor)

• rkive (Malcolm Taylor)

• x1 - реалізація LZP і PPM (Stig Valentini)

• RAR (версії 3 і вище) - реалізація варіанту PPMd, PPMII

• 7-Zip - реалізація варіанту PPMd

• WinZip (версії 10 і вище) - реалізація варіанту PPMd

7.9. Інкрементне та інтервальне кодування

Інкрементне кодування, також відоме під назвою фронтальне стиснення або тилове стиск, - це тип дельта-кодування (delta encoding), де загальні префікси або суфікси і їх довжини записуються таким чином, щоб уникати дублювання даних.

Цей алгоритм добре підходить для стиснення відсортованих даних, наприклад, списку слів у словнику [7.1, 7.6, 7.33]. (табл. 7.2).

Цей метод використовувався як базовий для утиліти GNU locate в індексі імен файлів і каталогів. Також дельта-кодування (delta encoding) використовується для довжин загального префікса. Це означає додатковий крок, в якому замість довжини загального префікса використовується зміна в довжині загального префікса. Незважаючи на простоту, інкрементне кодування може зберегти багато пам'яті, особливо при використанні перед іншими архіваторами, такими як gzip або bzip2.

Таблиця 7.2. Список слів у словнику

Вхідні дані	Загальний префікс	Стиснений вивід
myxa myxophyta myxopod nab nabbed nabbing nabit nabk nabob nacarat nacelle	начало данных 'myx' 'myxop' нет общего префикса 'nab' 'nabb' 'nab' 'nab' 'nab' 'na' 'nac'	0 myxa 3 ophyta 5 od 0 nab 3 bed 4 ing 3 it 3 k 3 ob 2 carat 3 elle
64 байт		46 байт

Інтервальне кодування (діапазон кодування) - ентропійний метод кодування, запропонований Г. Найджелом і Н. Мартіном в 1979 році. Це різновид арифметичного кодування.

Інтервальне кодування кодує всі символи повідомлення в одне число, на відміну від, наприклад, коду Хаффмана, який присвоює кожному символу послідовність біт і об'єднує всі бітові послідовності разом.

Припустимо, необхідно зашифрувати повідомлення «*AABA <EOM>*», де *<EOM>* - це символ кінця повідомлення (End of message). Для цього прикладу передбачається, що декодувальник знає, що ми маємо намір закодувати рівно п'ять символів в десятковій системі числення (алгоритм в даному випадку підтримує 105 різних комбінацій символів в діапазоні [0, 100000)) використовуючи розподіл ймовірностей {A: 0.60; B: 0.20; <EOM>: 0.20}. Кодувальник ділить діапазон [0, 100000) на три піддіапазони:

A: [0, 60000)

B: [60000, 80000)

<EOM>: [80000, 100000)

Оскільки наш перший символ - A, це знижує наш початковий діапазон до [0, 60000). Другий символ ділить цей діапазон ще на три частини:

AA: [0, 36000)

AB: [36000, 48000)

A<EOM>: [48000, 60000)

З двома закодованими символами наш діапазон стає [0, 36000) і наш третій символ надає наступні варіанти:

AAA: [0, 21600)

AAB: [21600, 28800)

AA<EOM>: [28800, 36000)

На цей раз вибір падає на другий з трьох варіантів, які представляють собою повідомлення, яке ми хочемо закодувати, і наш діапазон стає [21600, 28800). Може здатися, що стало складніше визначити наші піддіапазони в даному випадку, але насправді це не так: ми можемо просто відняти нижню межу з верхньої межі, щоб визначити, що в нашому діапазоні є 7200 чисел; перші 4320 з них представляють 0,60 від загального числа, такі 1440 представляють наступні 0,20, а решта 1440 представляють які 0,20 від загального діапазону [7.1, 7.33]. Надбавка нижньої межі дає нам наші діапазони:

AABA: [21600, 25920)

AABB: [25920, 27360)

AAB<EOM>: [27360, 28800)

Нарешті, наш діапазон звузився до [21600, 25920), у нас залишився тільки один символ для кодування. Використовуючи ту ж техніку, як і раніше, для поділу діапазону між нижньою і верхньою межею ми знаходимо три піддіапазона, що залишилися:

AABAA: [21600, 24192)

AABAB: [24192, 25056)

AABA<EOM>: [25056, 25920)

І так як <EOM> - це наш останній символ - наш кінцевий діапазон - [25056, 25920). Так як всі п'ятизначні числа, що починаються з «251», потрапляють в наш останній ряд, то ми могли б передати один з тризначних префіксів, щоб однозначно висловити вихідне повідомлення (той факт, що

насправді існує вісім таких префіксів, говорить про те, що можна оптимізувати алгоритм. Але вони виникли через використання десяткової системи числення, а не двійковій).

Арифметичне кодування аналогічно інтервальному, використовує дробові числа в діапазоні [0,1). Відповідно, в результаті арифметичний код інтерпретується як початок з неявним «0.», так як це просто різні інтерпретації одних і тих же методів кодування, то будь-який арифметичний кодер - це відповідний інтервальний кодувальник, і навпаки.

На практиці, однак, так зване діапазонні кодировщики мають тенденцію бути реалізованими в значній мірі, як описано в статті Мартіна, в той час як арифметичні кодувальники взагалі не називають діапазонними. Часто різницею є побайтова і побітова ренормалізація. Інтервальні кодувальники схильні використовувати байти, а не біти. Хоча це і знижує рівень стиснення, це швидше, ніж виконання перенормування для кожного біта.

7.10. Алгоритм Шеннона – Фано, метод Хаффмана та коди

Алгоритм Шеннона - Фано - один з перших алгоритмів стиснення, який вперше сформулювали американські вчені Шеннон і Роберт Фано.

Даний метод стиснення має велику схожість з алгоритмом Хаффмана, який з'явився на кілька років пізніше і є логічним продовженням алгоритму Шеннона. Алгоритм використовує коди змінної довжини: часто зустрічається символ кодується кодом меншої довжини, рідко зустрічається - кодом більшої довжини.

Коди Шеннона - Фано - префіксние, тобто ніяке кодове слово не є префіксом будь-якого іншого. Це властивість дозволяє однозначно декодувати будь-яку послідовність кодових слів.

Кодування Шеннона - Фано (Shannon-Fano coding) - алгоритм префіксного неоднорідного кодування. [7.1-7.6, 7.34].

Відноситься до імовірнісних методів стиснення (точніше, методам контекстного моделювання нульового порядку).

Подібно алгоритму Хаффмана, алгоритм Шеннона - Фано використовує надмірність повідомлення, укладену в неоднорідному розподілі частот символів його (первинного) алфавіту, тобто замінює коди більш частих символів короткими двійковими послідовностями, а коди більш рідкісних символів - довшими двійковими послідовностями.

Алгоритм був незалежно один від одного розроблений Шенноном (публікація «Математична теорія зв'язку», 1948 рік) і, пізніше, Фано (опубліковано як технічний звіт).

Приклад побудови кодової схеми (рис. 7.3) для шести символів a1 - a6 і ймовірностей pi.

a_i	p(a)	1	2	3	4	Итого
a_1	0.36	0	00			00
a_2	0.18		01			01
a_3	0.18	1	10			10
a_4	0.12		11	110		110
a_5	0.09			111	1110	1110
a_6	0.07				1111	1111

Рис. 7.3. Побудови кодової схеми

1. Символи первинного алфавіту m1 виписують по спадаючій ймовірностей.

2. Символи отриманого алфавіту ділять на дві частини, сумарні ймовірності символів яких максимально близькі один одному.

3. У префіксному коді для першої частини алфавіту присвоюється двоичная цифра «0», другої частини - «1».

4. Отримані частини рекурсивно діляться і їх частин призначаються відповідні виконавчі цифри в префіксному коді.

Коли розмір підалфавіта стає дорівнює нулю або одиниці, то подальшого подовження префіксного коду для відповідних йому символів первинного алфавіту не відбувається, таким чином, алгоритм привласнює різним символам префіксні коди різної довжини.

На кроці ділення алфавіту існує неоднозначність, так як різниця сумарних ймовірностей може бути однакова для двох варіантів поділу (враховуючи, що всі символи первинного алфавіту мають ймовірність більше нуля).

Іншими словами, оптимальна поведінка на кожному кроці шляху ще не гарантує оптимальності всієї сукупності дій.

Тому код Шеннона - Фано не є оптимальним в загальному сенсі, хоча і дає оптимальні результати при деяких розподілах ймовірностей.

Для одного і того ж розподілу ймовірностей можна побудувати, взагалі кажучи, кілька кодів Шеннона - Фано, і всі вони можуть дати різні результати.

Якщо побудувати всі можливі коди Шеннона - Фано для даного розподілу ймовірностей, то серед них будуть знаходитися і всі коди Хаффмана, тобто оптимальні коди.

Вмхідні символи:

A (частота народження 50);

B (частота народження 39);

C (частота народження 18);

D (частота народження 49)$

E (частота народження 35);

F (частота народження 24).

Приклад кодового дерева на рисунку 7.4.

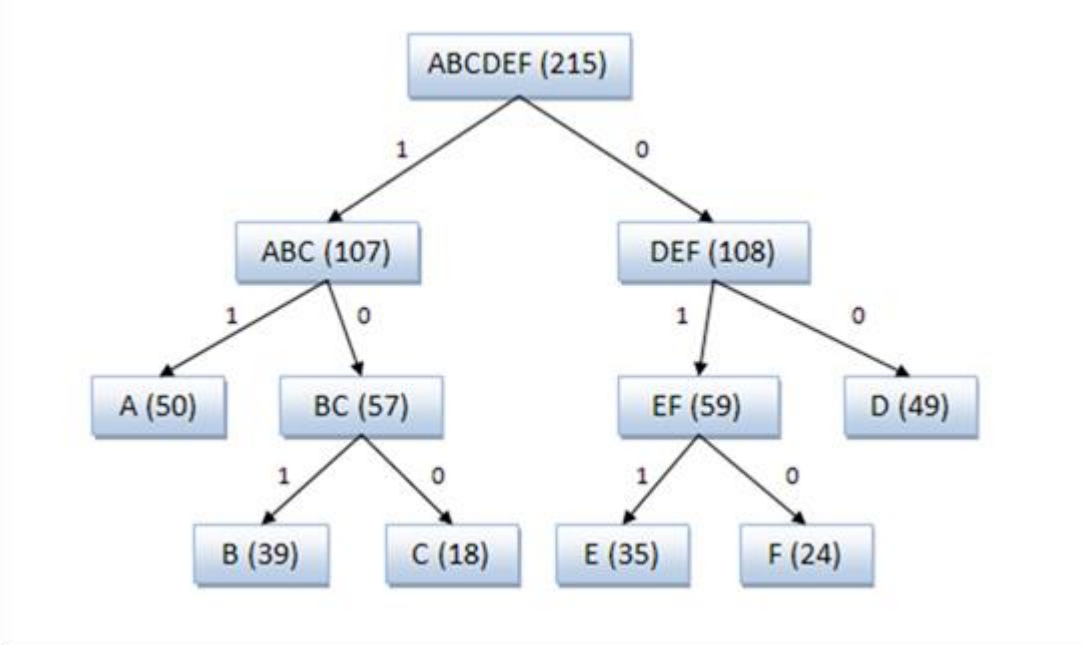


Рис. 7.4. Отриманий код: A - 11, B - 101, C - 100, D - 00, E - 011, F - 010

Кодування Шеннона - Фано є досить старим методом стиснення, і на сьогоднішній день воно не представляє особливого практичного інтересу.

У більшості випадків довжина послідовності, стислій за цим методом, дорівнює довжині стислій послідовності з використанням кодування Хаффмана.

Але на деяких послідовностях можуть сформуватися неоптимальні коди Шеннона - Фано, тому більш ефективним вважається стиснення методом Хаффмана

Кодування довжин **серій (Run-length encoding, RLE)** або кодування повторів - алгоритм стиснення даних, який замінює повторювані символи (серії) на один символ і число його повторів. Серією називається послідовність, що складається з декількох однакових символів. При кодуванні (упаковці, стисканні) рядок однакових символів, що складають серію, замінюється рядком, що містить сам повторюваний символ і кількість його повторів.

Розглянемо зображення, що містить текст чорного кольору на суцільному білому тлі. При порядковому читанні пікселів такого зображення будуть зустрічатися серії білих (фон) і чорних (літери) пікселів. Буквою B позначимо чорний піксель, а буквою W – білий [7.1, 7.35].. Розглянемо якусь довільну рядок зображення довжиною 47 символів:

WWWWWWWWWBBBWWWWWWWWWWWWWWWWWWWWWWWWBWWWWWWWWWWWWWW

Порахуємо кількість символів:

1. 4 символ «B»;

2. 43 символів «W».

Разом знайдено 5 серій. Замінимо серії на число повторів і сам повторюваний символ:

12W3B24W1B14W

Вийшла послідовність з 18 символів. Вихідна послідовність складалася з 47 символів. Дані були стиснуті в 47 / 18≈2.61 рази.

Візьмемо рядок, що складається з великої кількості неповторюваних символів:

ABCABCABCDDDFFFFFF

Після стиснення методом RLE такий рядок буде виглядати так:

1A1B1C1A1B1C1A1B1C3D6F

Вихідна рядок складається з 18 символів, а стиснута - з 22. Розмір даних збільшився в 22 / 18≈1.22 рази.

Щоб після стиснення розмір даних не збільшувався, алфавіт, в якому

записані довжини серій, ділять на дві частини (зазвичай рівні). Наприклад, алфавіт цілих чисел можна розділити на дві частини: позитивні і негативні числа. Позитивні числа використовують для запису кількості повторів одного символу, а негативні - для запису кількості неоднакових символів, що слідують один за одним.

Порахуємо символи з урахуванням вищесказаного:

• спочатку один за одним слідують 9 Не однакових символів: «ABCABCABC»;

• потім записані 3 символу «D»;

• нарешті записані 6 символів «F».

Стисла рядок запишеться у вигляді:

-9ABCABCABC3D6F

Вихідна рядок складається з 18 символів, а стиснута - з 15. Розмір даних зменшився в 18/15 = 1.2 рази.

Припустимо, реалізація методу RLE для запису довжин серій (для підрахунку кількості символів) використовує змінну целочисельного типу зі знаком «signed char». У таку зміну можна записати числа від -128 до 127 включно. Як же бути, якщо довжина серії дорівнює 128 символів і більше? В цьому випадку серію поділяють на частини так, щоб довжина частини не перевищувала 127 символів. Наприклад, серія, що складається з 256 символів «A», буде закодована наступним рядком (256 = 127 + 127 + 2): 127A127A2A.

Запис на деякій мові програмування алгоритму RLE з урахуванням цих обмежень нетривіальна. Звичайно, кодування, яке використовується для зберігання зображень, оперує з двійковими даними, а не з символами ASCII, як в розглянутих прикладах, проте принцип залишається тим же.

Очевидно, що таке кодування ефективно для даних, що містять велику кількість серій, наприклад, для простих графічних зображень, таких як іконки і графічні малюнки. Однак це кодування погано підходить для зображень з плавним переходом тонів, таких як фотографії.

Поширені формати для упаковки даних за допомогою RLE включають в себе PackBits, PCX і ILBM.

Методом кодування довжин серій можуть бути стиснуті довільні файли з двійковими даними, оскільки специфікації на формати файлів часто включають в себе повторювані байти в області вирівнювання даних. Проте, сучасні системи стиснення (наприклад, Deflate) частіше використовують алгоритми на основі LZ77, які є узагальненням методу кодування довжин серій і оперують з послідовностями символів виду «BWWBWWBWWBWW».

Звукові дані, які мають довгі послідовні серії байт (такі як низькоякісні звукові семпли) можуть бути стиснуті за допомогою RLE після того, як до них буде застосовано Дельта-кодування.

Метод стиснення з використанням словника - розбиття даних на слова і заміна їх на індекси в словнику. [7.1, 7.35].

В даний час це найбільш поширений підхід для стиснення даних, він є природним узагальненням RLE.У найбільш поширеному варіанті реалізації словник поступово поповнюється словами з вихідного блоку даних в процесі стиснення.

Основний параметр будь-якого словникового методу - це розмір словника. Чим більше словник, тим вище ефективність. Однак для неоднорідних даних надмірно великий розмір може бути шкідливий, так як при різкій зміні типу даних словник буде заповнений неактуальними словами. Для ефективної роботи цих методів при стисненні потрібна додаткова пам'ять - приблизно на порядок більше, ніж потрібно для вихідних даних словника. Суттєва перевага словникових методів - проста і швидка процедура розпакування. Додаткова пам'ять при цьому не потрібно. Така особливість вкрай важлива, якщо необхідний оперативний доступ до даних.

До методів стиснення з використанням словника відносяться наступні алгоритми: LZ77 / 78, LZW, LZO, Deflate, LZMA, LZX, ROLZ, LZ4, Zstd.

Унарний код - двійковий префіксний код змінної довжини для подання

натуральних чисел. [7.1, 7.36].

Код являє собою послідовність одиничних біт довжиною, рівній кодованих число, завершується нульовим бітом. Таким чином можливо закодувати нуль і все натуральні числа. Код зручний для кодування послідовностей чисел з різко переважаючими малими величинами. Є різновидом коду Голобма з m = 1 або коду Райса з k = 0.

Приклад кодування чисел:

0 - 0

1 - 10

2 - 110

3 - 1110

4 - 11110

5 – 111110

Універсальний код для цілих чисел в стисненні даних - префіксний код, який перетворює позитивні цілі числа в двійкові слова, з додатковим властивістю: при будь-якому дійсному розподіл ймовірностей на цілих числах, поки розподіл - монотонно (тобто для будь-якого), очікувані довжини двійкових слів знаходяться в межах постійного фактора очікуваних довжин, які оптимальний код призначив би для цього розподілу ймовірностей.

Універсальний код асимптотично оптимальний, якщо коефіцієнт між фактичними і оптимальними очікуваними довжинами пов'язує функція інформаційної ентропії коду, яка наближається до 1, так як ентропія наближається до нескінченності.

Більшість префіксних кодів для цілих чисел призначає довші ключові слова великим цілим числам. Такий код може використовуватися, щоб ефективно закодувати повідомлення, яке продовжуватиметься з набору можливих повідомлень, просто впорядковуючи набір повідомлень по зменшенню ймовірності а потім пересилаючи індекс якого товару, призначеного повідомлення. Універсальні коди в загальному не

використовуються для точно відомих розподілів ймовірностей [7.1, 7.34-7.37].

Універсальні коди включають в себе:

• Унарне кодування;

• Гамма-код Еліаса;

• Дельта-код Еліаса;

• Омега-код Еліаса;

• Дельта-код;

• Кодування Фібоначчі;

• Експоненціальний код Голомба.

Деякі неуніверсальні коди:

• Одномісне кодування, використовується в кодах Еліаса;

• Кодування Райса;

• Кодування Голомба.

Їх неуніверсальність проявляється в тому, що якщо будь-які з них використовувати, щоб закодувати розподіл Гаусса-Кузьміна або дзета-розподіл з параметром s = 2, то очікувана довжина ключового слова нескінченний.

Використання коду Хаффмана і арифметичного кодування (коли вони можуть використовуватися разом) дають кращий результат, ніж будь-який інший універсальний код.

Однак, універсальні коди корисні, коли код Хаффмана не може використовуватися - наприклад, коли неможливо визначити точну вірогідність кожного повідомлення, але відомо ранжування їх ймовірностей.

Універсальні коди також корисні, коли код Хаффмана відпрацьовує не зовсім коректно. Наприклад, коли відправник знає ймовірності повідомлень, а одержувач - немає, код Хаффмана вимагає передачі ймовірностей до одержувача. Використання універсального коду позбавляє від таких незручностей. Кожен універсальний код дає власне «мається на увазі розподіл» ймовірностей p (i) = 2-l (i), де l (i) - довжина i-го ключового слова і p (i) - ймовірність символу передачі. Якщо фактичні ймовірності повідомлення - q (i) і

відстань Кульбака - Лейблера DKL (q || p) мінімізує код з l (i), потім оптимальний код Хаффмана для цього безлічі повідомлень буде еквівалентний до цього коду. З тих пір, як універсальні коди стали працювати швидше, щоб кодувати і декодувати, ніж код Хаффмана, універсальний код був би кращий у випадках, де DKL (q || p) досить маленький.

Для будь-якого геометричного розподілу кодування Голомба оптимально. З універсальними кодами, мається на увазі розподіл - приблизно енергетичний закон як наприклад 1 / n2. Для коду Фібоначчі, яке мається на увазі розподіл становить приблизно 1 / nq.

Експонентний код Голомба порядку k - це універсальний код, параметризований цілим числом k. Розроблено Соломоном Голомбом. Для кодування невід'ємного числа в експонентний код Голомба порядку k можна використовувати наступний метод:

1. Взяти число N в двійковому коді, без останніх k цифр. Додати до нього 1 (арифметично): N = N + 1. Записати отримане N.

2. Підрахувати кількість C біт в N.

3. Відняти від C одиницю: C = C-1. Записати З нульових біт перед обраним числом N.

Для порядку k = 0 код виглядає так:

0 => 1 => 1

1 => 10 => 010

2 => 11 => 011

3 => 100 => 00100

4 => 101 => 00101

5 => 110 => 00110

6 => 111 => 00111

7 => 1000 => 0001000

8 => 1001 => 0001001

...

Експонентний код Голомба при k = 0 використовується в стандартах стиснення відео H.264 і MPEG-4 AVC, в яких є також можливість кодування знакових чисел шляхом присвоєння значення 0 ключовим словом "0" в бінарному вигляді і подальше призначення кодових слів до вхідних значень збільшуються амплітуд і змінних знаків.

Експонентний код Голомба також використовується в алгоритмі кодування нестислого відео Dirac.

При k = 0 експоненціальне кодування Голомба збігається з гамма-кодом Еліаса цього ж числа плюс один. Таким чином, він може кодувати нуль, тоді як гамма-код Еліаса може кодувати тільки числа більше нуля.

Незважаючи на близькі назву, експоненціальне кодування Голомба лише трохи аналогічно кодуванню Голомба, яке представляє собою тип ентропійного кодування, але не є універсальним кодом.

В математиці **негафібоначчіева система числення - універсальний код**, який кодує ненульові цілі числа в двійкові кодові слова. Узагальнює фібоначчійовий систему числення на все ненульові цілі числа (а не тільки натуральні). Всі коди закінчуються "11" і не мають "11" в середині або на початку слова. Коди для цілих чисел від -11 до 11 наведені нижче[7.17-7.19].

xx	негафібоначієво представлення	негафібоначієв код
-11	101000	0001011
-10	101001	1001011
-9	100010	0100011
-8	100000	0000011
-7	100001	1000011
-6	100100	0010011
-5	100101	1010011
-4	1010	01011
-3	1000	00011
-2	1001	10011

-1	10	011
0	0	(не кодируется)
1	1	11
2	100	0011
3	101	011
4	10010	010011
5	10000	000011
6	10001	00011
7	10100	001011
8	10101	101011
9	1001010	01010011
10	1001000	00010011
11	1001001	10010011

Код Фібоначчі тісно пов'язаний з негафібоначчієвим поданням, іноді використовується математиками позиційною системою числення.

Код негафібоначчі для кожного ненульового цілого числа - це в точності його негафібоначчієво уявлення (представлення через числа Фібоначчі з негативними номерами) зі зворотним порядком цифр і додатковоюодиницею в кінці. Всі негативні числа мають непарну кількість розрядів, все позитивні - парну. Для кодування ненульового цілого числа X слід [7.1, 7.24, 7.37].

1. Розрахувати кількість необхідних розрядів N шляхом підсумовування непарних (або парних) чисел негафібоначчі з номерами від 1 до N.

2. Після знаходження N - кількості бітів для кодування X - відняти N-е число негафібоначчі з X, запам'ятати отриману різницю і поставити одиницю на місце N-го розряду шуканого кодового слова.

3. Спускаючись від N-го біта до першого, порівнювати кожне відповідне число негафібоначчі з отриманою різницею. Віднімати його з різниці, якщо модуль нової різниці буде менше, і, крім того, попередній старший розряд не одиниця. Розглядати нову різницю і т.д. У відповідний розряд ставиться

одиниця, якщо віднімання здійснювалося, і нуль в іншому випадку.

4. На N + 1-е місце поміщається одиниця, щоб закінчити кодування і показати, що біт, щоб закінчити.

Для декодування слід видалити останню одиницю, іншим бітам привласнити значення 1, -1, 2, -3, 5, -8, 13 ... (числа негафібоначчі), і скласти значення в ненульових розрядах.

Ентропійне кодування - кодування послідовності значень з можливістю однозначного відновлення з метою зменшення обсягу даних (довжини послідовності) за допомогою усереднення ймовірностей появи елементів в закодованої послідовності.

Передбачається, що до кодування окремі елементи послідовності мають різну ймовірність появи. Після кодування в результуючої послідовності ймовірності появи окремих символів практично однакові (ентропія на символ максимальна). Розрізняють декілька варіантів кодів:

• Зіставлення кожному елементу вихідної послідовності різного числа елементів результуючої послідовності. Чим більша ймовірність появи вихідного елемента, тим коротше відповідна результуюча послідовність. Прикладом можуть служити код Шеннона - Фано, код Хаффмана,

• Зіставлення кількох елементів вихідної послідовності фіксованого числа елементів кінцевої послідовності. Прикладом є код Танстола.

• Інші структурні коди, засновані на операціях з послідовністю символів. Прикладом є кодування довжин серій.

• Якщо приблизні характеристики ентропії потоку даних попередньо відомі, може бути корисний більш простий статичний код, такий як унарное кодування, гамма-код Еліаса, код Фібоначчі, код Голомба або кодування Райса.

Згідно з теоремою Шеннона, існує межа стиснення без втрат, що залежить від ентропії джерела. Чим більш передбачувані одержувані дані, тим краще їх можна стиснути. Випадкова незалежна рівноімовірнісна послідовність стиску без втрат не піддається.

ЛІТЕРАТУРА

7.1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципи, технологии, протоколы: Учебник для вузов. 3-е изд. – СПб.: Питер, 2006.– 958 с.

7.2. Duda, Jarek. «Optimal encoding on discrete lattice with translational invariant constrains using statistical algorithms.» arXiv preprint arXiv:0710.3861 (2007).

7.3. Duda, Jarek. «Asymmetric numeral systems» arXiv preprint arXiv:0902.0271 (2009).

7.4. Duda, Jarek, Khalid Tahboub, Neeraj J. Gadgil, and Edward J. Delp. «The use of asymmetric numeral systems as an accurate replacement for huffman coding.»In Picture Coding Symposium (PCS), 2015, pp. 65-69. IEEE, 2015.

7.5. Duda, Jarek. «Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding.» arXiv preprint arXiv:1311.2540 (2013).

7.6. Najmabadi, Seyyed Mahdi, Zhe Wang, Yousef Baroud, and Sven Simon. «High throughput hardware architectures for asymmetric numeral systems entropy coding.» In Image and Signal Processing and Analysis (ISPA), 2015 9th International Symposium on, pp. 256—259. IEEE, 2015.

7.7. Richard Chirgwin. Google's new squeeze: Brotli compression open-sourced. The Register (23 September 2015).

7.8. Alakuijala, Jyrki Comparison of Brotli, Deflate, Zopfli, LZMA, LZHAM and Bzip2 Compression Algorithms.

7.9. Slater-Robbins, Max Chrome and Firefox are about to get a lot faster thanks to Google's new data compression algorithm.

7.10. Hakbeom Jang; Channoh Kim, Jae W. Lee. Practical Speculative Parallelization of Variable-Length Decompression Algorithms (англ.). Конференция Languages, Compilers and Tools for Embedded Systems 2013 (June 20–21, 2013). — «The bzip2 file format defines a 48-bit pattern called magic header

(0x314159265359), which signals the beginning of a new compressed block».

7.11. Ватолин Д. Методы сжатия данных. Устройство архиваторов, сжатие изображения и видео.. — М.: Диалог-МИФИ, 1993. — С. 9. — ISBN 5-86404-170-x.

7.12. Phillip Lougher. Squashfs: Add LZ4 compression configuration option.

7.13. Владимир Лидовский, Лекция 7: Подстановочные или словарно-ориентированные алгоритмы сжатия информации. Методы Лемпела-Зива (LZ78) в курсе лекций «Основы теории информации и криптографии» // Интуит.ру, 11.04.2007.

7.14. M. A. Basir, M. H. Yousaf. Transparent Compression Scheme for Linux File System (англ.) // The Nucleus. — 2012. — Vol. 49, no. 2. — P. 133.

7.15. Storer, James A. Data Compression via Textual Substitution (неопр.) // Journal of the ACM. — 1982. — October (т. 29, № 4). — С. 928—951. — DOI:10.1145/322344.322346.

7.16. Ryabko, B. Ya. Data compression by means of a «book stack», Problems of Information Transmission, 1980, v. 16: (4), pp. 265–269.

7.17. Ryabko, B. Ya.; Horspool, R. Nigel; Cormack, Gordon V. Comments to: «A locally adaptive data compression scheme» by J. L. Bentley, D. D. Sleator, R. E. Tarjan and V. K. Wei. Comm. ACM 30 (1987), no. 9, 792—794.

7.18. Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео Диалог-МИФИ, 2002 г., 384 с. ISBN 5-86404-170-X. 3000 экз.

7.19. Y. Rathore, M. Ahirwar, R. Pandey. A Brief Study of Data Compression Algorithms (англ.) // Journal of Computer Science IJCSIS. — 2013. — October (vol. 11, no. 10). — P. 90.

7.20. Knuth, Donald (2008), *Negafibonacci Numbers and the Hyperbolic Plane*, Paper presented at the annual meeting of the Mathematical Association of America, San Jose, California.

7.21. Knuth, Donald (2009), *The Art of Computer Programming, Volume 4,*

Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams, ISBN 0-321-58050-8. In the pre-publication draft of section 7.1.3 see in particular pp. 36–39.

7.22. Margenstern, Maurice (2008), *Cellular Automata in Hyperbolic Spaces*, vol. 2, Advances in unconventional computing and cellular automata, Archives contemporaines, c. 79, ISBN 9782914610834.

7.23. Digital Signal Compression: Principles and Practice (By William A. Pearlman, Amir Said, 2011, ISBN 9780521899826), Chapter 4 "Entropy coding techniques" pp41-76

7.24. *Alexander Neumann.* Zopfli: Neue Kompressionsbibliothek von Google | heise Developer (нем.).

7.25. *Alakuijala, Jyrki* Data compression using Zopfli .

7.26. *Dean Hume.* Improved Compression Ratios Using Zopfli (2015).

7.27. *Sharwood, Simon* Google open sources very slow compression algorithm. *The Register* (2013).

7.28. *Ilya Grigorik.* Google Fonts recently switched to using new Zopfli compression algorithm. Google+

7.29. J.G. Cleary, and I.H. Witten, Data compression using adaptive coding and partial string matching (недоступная ссылка), *IEEE Transactions on Communications*, Vol. **32** (4), pp. 396–402, April 1984.

7.30. A. Moffat, Implementing the PPM data compression scheme, *IEEE Transactions on Communications*, Vol. **38** (11), pp. 1917–1921, November 1990.

7.31. J.G. Cleary, W.J. Teahan, and I.H. Witten, Unbounded length contexts for PPM, In J.A. Storer and M. Cohn, editors, *Proceedings DCC-95*, IEEE Computer Society Press, 1995.

7.32. C. Bloom, Solving the problems of context modeling.

7.33. W.J. Teahan, Probability estimation for PPM.

7.34. T. Schürmann and P. Grassberger, Entropy estimation of symbol sequences (недоступная ссылка), *Chaos*, Vol. **6**, pp. 414–427, September 1996.

7.35. Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин *Методы сжатия*

данных. Устройство архиваторов, сжатие изображений и видео Диалог-МИФИ, 2002 г., 384 с. ISBN 5-86404-170-X. 3000 экз.

7.36. G. Nigel N. Martin, *Range encoding: An algorithm for removing redundancy from a digitized message*, Video & Data Recording Conference, Southampton, UK, July 24-27, 1979.

7.37. «Source coding algorithms for fast data compression» Richard Clark Pasco, Stanford, CA 1976

8. ПРОТОКОЛИ ПРИКЛАДНОГО РІВНЯ

Протокол прикладного рівня (Application layer) - протокол верхнього (7-го) рівня мережевої моделі OSI, забезпечує взаємодію мережі й користувача.

Рівень дозволяє додаткам користувача мати доступ до мережевих служб, таким, як обробник запитів до баз даних, доступ до файлів, пересилання електронних повідомлень.

Також відповідає за передачу службової інформації, надає додаткам інформацію про помилки і формує запити до рівня уявлення. Приклад: HTTP, POP3, SMTP.

8.1. DHCP - протокол динамічної конфігурації вузла

DHCP (Dynamic Host Configuration Protocol — протокол динамічної конфігурації вузла) — це стандартний протокол прикладного рівня, який дозволяє комп'ютерам автоматично отримувати IP-адресу та інші параметри, необхідні для роботи в мережі. Для цього комп'ютер звертається відповідно — до DHCP-сервера. Мережевий адміністратор може задати діапазон адрес, які будуть розподілені між комп'ютерами. Це дозволяє уникнути ручного налаштування комп'ютерів мережі й зменшує кількість помилок. Протокол DHCP використовується в більшості великих мереж TCP/IP [8.1].

DHCP є розширенням протоколу BOOTP, що використовувався раніше для забезпечення бездискових робочих станцій IP-адресами при їхньому завантаженні. DHCP зберігає зворотну сумісність з BOOTP.

Стандарт протоколу DHCP був прийнятий у жовтні 1993 року. Остання версія протоколу (березень 1997 року) описана в RFC 2131. Нова версія DHCP, призначена для використання в середовищі IPv6, зветься DHCPv6 і визначена в RFC 3315 (липень 2003 року).

Крім IP-адреси, DHCP також може повідомляти клієнтові додаткові

параметри, необхідні для нормальної роботи в мережі. Ці параметри називаються опціями DHCP. Список стандартних опцій можна знайти в RFC 2132. Деякими з найбільш часто використовуваних опцій є:

- IP-адреса маршрутизатора за замовчуванням;
- маска підмережі;
- адреси серверів DNS;
- ім'я домену DNS.

Деякі постачальники програмного забезпечення можуть визначати власні, додаткові опції DHCP.

Протокол DHCP працює за схемою клієнт-сервер. Під час запуску системи комп'ютер, який є DHCP-клієнтом, відправляє в мережу запит на отримання IP-адреси. DHCP-сервер відповідає і відправляє повідомлення-відповідь, яка містить IP-адресу і деякі інші конфігураційні параметри. При цьому сервер DHCP може працювати в різних режимах, включаючи [8.1]:

1. Динамічний розподіл - адміністратор присвоює IP-діапазон адрес на сервері DHCP. Кожен клієнтський комп'ютер в мережі повинен запросити IP-адресу від DHCP-сервера, коли мережа ініціалізується за концепцією "оренди". Коли закінчується термін оренди, якщо вона не буде продовжена, DHCP-сервер має право повернути адресу і призначити її на інші комп'ютери.

2. Автоматичне виділення - сервер DHCP буде постійно призначати вільний IP-адрес з діапазону, встановленого адміністратором, запитуючому комп'ютеру. Основна відмінність з динамічним розподілом в тому, що сервер зберігає записи минулих завдань IP і намагається привласнити ту ж адресу тому ж комп'ютеру для майбутніх мережних підключень.

3. Статичний розподіл - сервер DHCP робить призначення IP-адрес виключно на основі таблиці MAC-адрес, які зазвичай заповнені вручну адміністратором мережі. Якщо MAC-адреса комп'ютера не зазначена в таблиці, йому не буде призначена мережева адреса.

Протокол DHCP є клієнт-серверним, тобто в його роботі беруть участь

клієнт DHCP і сервер DHCP. Передача даних здійснюється за допомогою протоколу UDP, при цьому сервер приймає повідомлення від клієнтів на порт 67 і відправляє повідомлення клієнтам на порт 68.

Протокол DHCP побудований так, що клієнт може звертатися із запитом відразу до декількох серверів. Клієнт DHCP, що потребує адресу, посилає широкомовний пакет DHCPDISCOVER в пошуках сервера. Пакет містить апаратну адресу запитувача клієнта. Потім один або кілька серверів DHCP розглядають запит і посилають у відповідь пакет DHCPOFFER, що містить пропоновану IP-адресу і "час оренди".

Клієнт вибирає адресу з отриманих пакетів DHCPOFFER. Вибір клієнта залежить від його призначення - наприклад, він може вибрати адресу з найбільшим часом оренди. Слідом за тим клієнт посилає пакет DHCPREQUEST з адресою вибраного сервера.

Таблиця 8.1. Структура протоколу DHCP

Поле	Опис	Довжина(в байтах)
op	Тип повідомлення. Наприклад може приймати значення: BOOTREQUEST (1, запит від клієнта до сервера) і BOOTREPLY (2, відповідь від сервера до клієнта).	1
htype	Тип апаратної адреси. Допустимі значення цього поля визначені в RFC1700 «Assigned Numbers». Наприклад, для MAC-адресиEthernet 10 Мбіт/с це поле приймає значення 1.	1
hlen	Довжина апаратної адреси в байтах. Для MAC-адреси Ethernet - 6	1
hops	Кількість проміжних маршрутизаторів (так званих агентів ретрансляції DHCP), через які пройшло повідомлення. Клієнт встановлює це поле в 0.	1
xid	Унікальний ідентифікатор транзакції, що генерується клієнтом на початку процесу отримання адреси.	4
secs	Час в секундах з моменту початку процесу отримання адреси. Може не використовуватися (в цьому випадку воно встановлюється в 0).	2

Продовження таблиці 8.1. Структура протоколу DHCP

Поле	Опис	Довжина(в байтах)
flags	Поле для прапорів - спеціальних параметрів протоколу DHCP	2
ciaddr	IP-адреса клієнта. Заповнюється тільки в тому випадку, якщо клієнт вже має власну IP-адресу і здатний відповідати на запити ARP (це можливо, якщо клієнт виконує процедуру поновлення адреси після закінчення терміну оренди).	4
yiaddr	Нова IP-адреса клієнта, запропонована сервером	4
siaddr	IP-адреса сервера. Повертається в реченні DHCP (див. нижче).	4
giaddr	IP-адреса агента ретрансляції, якщо такий брав участь в процесі доставки повідомлення DHCP до сервера.	4
chaddr	Апаратна адреса (зазвичай MAC-адреса) клієнта.	16
sname	Необов'язкове ім'я сервера у вигляді нуль-термінованого рядка.	64
file	Необов'язкове ім'я файлу на сервері, що використовується бездисковими робочими станціями при віддаленому завантаженні. Як і sname, представлено у вигляді нуль-термінованого рядка.	128
options	Поле опцій DHCP. Тут вказуються різні додаткові параметри конфігурації. На початку цього поля вказуються чотири особливих байта зі значеннями 99, 130, 83, 99 («чарівні числа»), що дозволяють серверу визначити наявність цього поля. Поле має змінну довжину, проте DHCP-клієнт повинен бути готовий прийняти DHCP-повідомлення завдовжки 576 байт (в цьому повідомленні поле options має довжину 340 байт).	змінна

Обраний сервер посилає підтвердження (DHCPACK) і процес узгодження завершується. Пакет DHCPACK містить обумовлені адресу та час оренди. Сервер позначає виділену адресу як зайняту - до закінчення терміну оренди цю адресу не можна буде присвоїти іншому клієнту. Клієнту залишилося тільки сконфігурувати себе відповідно до надісланих даних і можна приступати до роботи в мережі.

Отже, на запит DHCPDISCOVER може відповісти кілька серверів. Клієнт повинен вибрати одну з пропозицій і послати у відповідь пакет DHCPREQUEST з ідентифікатором вибраного сервера. Інші сервери переглядають пакет DHCPREQUEST і укладають на основі ідентифікатора сервера, що їх пропозиція була відкинута. Таким чином, вони знають, що запропоновані ними IP-адреси вільні для призначення іншим клієнтам. У разі якщо сервер не може прийняти конфігурацію, він посилає пакет DHCPNAK (відмова в підтвердженні), що змушує клієнта почати процес узгодження заново. Виходячи з цього, якщо в мережі два DHCP-сервера з різними конфігураціями, немає ніякої гарантії, що клієнт вибере саме ваш сервер.

8.2. BOOTP - ротокол автоматичного отримання клієнтом IP-адреси

BOOTP (Bootstrap Protocol) — мережевий протокол, що використовується для автоматичного отримання клієнтом IP-адреси. Це зазвичай відбувається під час завантаження комп'ютера. BOOTP визначений в RFC 951.

BOOTP дозволяє бездисковим робочим станціям отримувати IP-адресу перш, ніж буде завантажена повноцінна операційна система. Історично це використовувалося для Unix-подібних бездискових станцій, які, зокрема, могли отримувати інформацію про місцеположення завантажувального диску за допомогою цього протоколу. А також великими корпораціями для установки заздалегідь налаштованого програмного забезпечення (наприклад Windows) на новопридбані комп'ютери. Обслуговуючий персонал зіткнувся з проблемами постійного підключення та переміщення нових пристроїв, а також з необхідністю зміни конфігурації мережі для відповідності сучасним вимогам до мереж. Все це призвело до необхідності створення механізму для автоматизації конфігурації мережевих вузлів, розподілених операційних систем і мережевого програмного забезпечення.

Найбільш ефективним способом реалізації такого механізму може бути збереження конфігураційних параметрів і образів програмного забезпечення на одному або декількох серверах завантаження (boot server). Під час запуску система взаємодіє з таким сервером, отримує від нього початкові параметри конфігурації і при необхідності завантажує з нього потрібне програмне забезпечення.

BOOTP був введений в RFC 951 як заміна застарілого RARP. Спочатку BOOTP розроблявся для бездискових робочих станцій. Сучасні умови привели до необхідності автоматизації завантаження систем, що мають в ПЗУ тільки базові засоби для IP, UDP і TFTP. Вихідний сценарій завантаження виглядав наступним чином [8.1, 8.2]:

1. Клієнт відправляє в широкомовної розсилки повідомлення UDP на

завантажувальний інформацію.

2. Сервер повертає клієнту його IP-адресу і, при необхідності, місце розташування файлу завантаження.

3. За допомогою найпростішого протоколу пересилки файлів TFTP (Trivial File Transfer Protocol) клієнт завантажує в власну пам'ять необхідне програмне забезпечення і починає роботу.

Код операції (opcode) вказує на тип повідомлення:

1 - для запиту (BOOTREQUEST);

2 - для відгуку (BOOTREPLY).

Таблиця 8.2. Формат повідомлень BOOTP

Зміщення сігмента	Довжина, октет	Опис
0	1	**Op** Код операції
1	1	**HType** Тип обладнання
2	1	**HLen** Довжина апаратної адреси
3	1	**Hops** Кількість пересилань
4	4	**XID** Ідентифікатор транзакції
8	4	**Secs** Лічильник секунд від моменту відправки клієнтом першого запроса
10	2	Не использовалось в RFC 951 **Flags** — поле флагов в RFC 1542
12	4	**CIAddr** IP-адреса клієнта
16	4	**YIAddr** IP-адреса, надана клієнту сервером

Продовження таблиці 8.2. Формат повідомлень BOOTP

Зміщення сігмента	Довжина, октет	Опис
20	4	**SIAddr** IP-адреса сервера
24	4	**GIAddr** IP-адреса проміжного маршрутизатора
28	16	**CHAddr** Апаратна адреса клієнта
44	64	**SName** Ім'я хоста сервера
108	128	**File** Ім'я загрузочного файла
236	64	**Vend** Область для розробників і Додаткові параметри

Визначає тип використовуваного мережевого обладнання, використовуючи значення аналогічні полю Hardware Type (HType, HRD) в специфікації протоколу ARP [8.3, 8.4].

Таблиця 8.3. Деякі часто використовувані значення:

HType	Опис
1	Ethernet (10Mb)
6	IEEE 802 Networks
7	ARCNET
15	Frame Relay
16	Asynchronous Transfer Mode (ATM)
18	Fibre Channel
20	Serial Line

Визначає довжину апаратного адреси в повідомленні. Для мереж *Ethernet* і інших, що використовують *IEEE 802*, значення цього параметра дорівнює 6. Аналогічне поле в *ARP*-пакеті - *HLN*.

Даний сегмент використовується ретрансляторами для контролю пересилання повідомлення. Значення поля встановлюється в 0 перед відправкою і збільшується на 1 при проходженні через кожен ретранслятор.

Ідентифікатор транзакції (*transaction ID*) - 32-бітове ціле число, яке встановлюється клієнтом і повертається сервером. Воно дозволяє клієнту зіставити відгук із запитом. Клієнт встановлює в це поле випадкове число для кожного запиту [8.1, 8.2].

Коли клієнт відсилає перший запит на завантаження даних, поле лічильника секунд має нульове значення. Якщо на запит не спадає відповіді, по завершенні тайм-ауту клієнт знову відправляє запит, змінюючи значення в полі лічильника секунд. Для тайм-ауту клієнт використовує випадковий інтервал, що збільшується до значення 60 с.

Дане поле не має спеціального призначення. Його вміст може перевіряти сервер або мережевий монітор для визначення тривалості очікування клієнтом завантаження по мережі. Сервер може використовувати значення з поля лічильника секунд для ранжирування запитів по пріоритетам, проте в даний час в більшості реалізацій це поле ігнорується.

В оригінальному стандарті RFC 951 це двухбайтовое полі не заповнювався. Згідно RFC 1542 воно використовується для встановлення прапорів.

Таблиця 8.4. Данні прапора

Ім'я прапора	Розмір, біт	Опис
B	1	Широкомовлення: при відправці оригінального повідомлення клієнту невідомий власний IP-адреса, і цей прапор виставляється в значення "1". Такий стан вказує отримав пакет BOOTP-серверів і ретрансляторів, що запит від цього клієнта повинен бути розісланий як широкомовний.
Reserved	15	Зарезервовані і не використовуються, значення виставлені в 0.

Якщо клієнт вже знає свій IP адреса, він заповнює поле IP адреса клієнта (client IP address). Якщо немає - клієнт встановлює це значення в 0. У останньому випадку сервер вставляє в поле ваш IP адреса (your IP address) IP

адреса клієнта. Поле IP адреса сервера (server IP address) заповнюється сервером. Якщо використовується уповноважений сервер, він заповнює IP адреса шлюзу (gateway IP address).

Клієнт повинен встановити свій апаратний адресу клієнта (client hardware address). Це те ж значення, яке знаходиться в заголовку Ethernet і в поле UDP датаграми, завдяки чому воно стає доступним будь-якому призначеному для користувача процесу (наприклад, сервера BOOTP), який отримав датаграмму. Зазвичай процесу, який працює з UDP датаграми, складно або практично неможливо визначити значення, що знаходиться в полі заголовка Ethernet датаграми, в якій передається UDP датаграма.

Ім'я хоста сервера (server hostname) це рядок, яка заповнюється сервером (не обов'язково). Сервер також може заповнити поле імені завантажувального файлу (boot filename). У це поле заноситься повний шлях до файлу, який використовується при завантаженні.

Спочатку область для розробників (vendor specific area) використовувалася в повідомленнях для пересилання відомостей, специфічних для конкретної реалізації. Однак на початку застосування BOOTP ця область залишалася вільною, хоча великий обсяг інформації (наприклад, маска підмережі або адреса маршрутизатора за замовчуванням) формально не включався в повідомлення. Область для розробників служила для розміщення додаткових конфігураційних параметрів, а також відомостей, специфічних для розробника. У цій області визначено досить багато різних полів.

Для BOOTP виділено два заздалегідь відомих порта: 67 для сервера і 68 для клієнта. Це означає, що клієнт не вибирає невикористаний динамічно призначається порт, а використовує порт номер 68. Причина, по якій були обрані два номери портів, замість того щоб використовувати тільки один для сервера, полягає в тому, що сервер може відправити відгук (хоча зазвичай він цього не робить) широкомовною чином [8.1, 8.2]. Якщо відгук від сервера поширювався б широкомовною чином, і якщо клієнту було б необхідно вибрати

динамічно призначається номер порту, ці широкомовні пакети також були б отримані іншими додатками на інших хостах, які використовують той же самий динамічно призначається номер порту. Таким чином, можна зробити висновок, що відправляти циркулярний запит на випадковий (динамічно призначається) номер порту не раціонально.

Якщо клієнт скористається заздалегідь відомим портом сервера (67), всі сервери в мережі будуть змушені переглядати кожен широкомовний відгук. (Якщо все сервера були «розбуджені», їм доведеться перевірити код операції, визначити, що це відгук, а не запит, і знову «заснути».) Тому вибір був зупинений на тому, як все зроблено зараз, тобто клієнт має свій власний єдиний заздалегідь відомий порт, який відрізняється від заздалегідь відомого порту сервера. Якщо кілька клієнтів завантажуються в один і той же час, і якщо відгуки від сервера поширюються широкомовними запитами, кожен клієнт переглядає відгуки, які призначені для клієнтів. Клієнти використовують поле ідентифікатора транзакції в BOOTP заголовку, щоб зіставити відгук із запитом, або ж сервери переглядають повернутий апаратну адресу клієнта.

8.3. BACnet – протокол в системах автоматизації будівель і мережах управління

BACnet (Building Automation and Control network) - мережевий протокол, який застосовується в системах автоматизації будівель і мережах управління.

BACnet-пристрій - це пристрій системи автоматизації (контролер, датчик, виконавчий механізм), що підтримує протокол BACnet.

Мережа BACnet - промислова мережа, що складається з BACnet-пристроїв [8.1-8.3]. BACnet гарантує можливість взаємодії між пристроями різних виробників, якщо алгоритми цих пристроїв реалізовані на основі стандартних функціональних блоків BIBB (BACnet Interoperability Building Block). Блоки BIBB використовуються для обміну даними між пристроями.

Вони розроблені для спрощення роботи інженерів, яким достатньо написати короткі специфікації, що описують вимоги до взаємодії різних пристроїв, що входять в систему BACnet. Підтримувані блоки BIBB для кожного пристрою BACnet перераховані в PICS (Protocol Implementation Conformance Statement). PICS це документ, детально описує тип даного пристрою BACnet, і його можливості до взаємодії з іншими пристроями.

Кожен пристрій в мережі BACnet описується набором стандартних об'єктів. Кількість однакових об'єктів, що становлять пристрій, не обмежена. Стандарт визначає наступні типи об'єктів:

- Аналоговий вхід (AI)
- Аналоговий вихід (AO)
- Аналогове значення (AV)
- Двійковий вхід (BI)
- Двійковий вихід (BO)
- Двійкове значення (BV)
- Вхід з багатьма станами (Multi-State Input)
- Вихід з багатьма станами (Multi-State Output)
- Календар (Calendar)
- Реєстрація події (Event Enrollment)
- Файл (File)

Клас повідомлення (Notification Class)

- Група (Group)
- Цикл (Loop)
- Програма (Program)
- Розклад (Schedule)
- Команда (Command)
- Пристрій (Device)
- HVAC (Heating Ventilating Air-Conditioning).

В процесі розвитку стандарту можуть з'явитися нові стандартні об'єкти.

Кожен об'єкт в мережі BACnet характеризується набором властивостей, які описують його поведінку або управляють його роботою.

Стандарт визначає класи прикладних задач, які виконують пристрої:

• Тривоги і події

• Доступ до файлів

• Доступ до об'єктів

• Управління віддаленим пристроєм

• Віртуальний термінал.

Класи прикладних задач описуються набором служб (сервісів), які використовуються для спілкування між пристроями.

Наприклад, клас управління віддаленим пристроєм включає наступні служби:

• DeviceCommunicationControl

• ConfirmedPrivateTransfer

• UnconfirmedPrivateTransfer

• ReinitializeDevice

• ConfirmedTextMessage

• UnconfirmedTextMessage

• TimeSynchronization (синхронізація часу)

• Who-Is (пошуку пристрою)

• I-Am (відповідь пристрою)

• Who-Has (пошуку об'єкта)

• I-Have (відповідь пристрою, що містить об'єкт).

Для класу доступу до об'єктів задані служби:

• CreateObject (створити об'єкт)

• DeleteObject (видалити об'єкт)

• ReadProperty (прочитати властивість)

• ReadPropertyConditional (прочитати властивість - за умовою)

• ReadPropertyMultiple (прочитати групу властивостей)

• WriteProperty (записати властивість)

• WritePropertyMultiple (записати групу властивостей)

• AddListElement (внести елемент в список)

• RemoveListElement (видалити елемент зі списку)

В якості канального / фізичного рівнів BACnet використовує такі технології:

• ARCNET

• Ethernet

• BACnet / IP

• PTP (Point-To-Point) через RS-232

• MS / TP (Master-Slave / Token-Passing) через RS-485

• LonTalk.

8.4. DICT - мережевий протокол для доступу до словників

DICT - мережевий протокол для доступу до словників, створений DICT Development Group.

Протокол опісується в RFC 2229. Створений з метою перевершити пропрієтарній протокол Webster. Протокол дозволяє Клієнтам одночасний пошук в декількох Словниках, Різні алгоритми пошуку, використання мультимедіа. Dict-сервери и клієнти використовують TCP-порт 2628.

Протягом багатьох років, Інтернет-спільнота покладалась на "Вебстер“ - протокол для доступу до визначень мовою оригіналу. Вебстер протокол підтримував доступ до одного словника і (опціонально) одного тезауруса [8.4]. В останні роки кількість загальнодоступних Вебстер серверів в Інтернеті різко зменшилася. DICT протокол призначений для забезпечення доступу до декількох баз даних.

Посилається запит на визначення слова, йде пошук індекса слова (за допомогою легко розширюваного набору алгоритмів), можуть бути надані

інформація про сервер (наприклад, які стратегії пошукового індексу підтримуються, або які бази даних є), а також інформація про базу даних (наприклад, авторське право, цитата, або розподіл інформації).

Крім того, DICT протокол має опції, які можуть бути використані для обмеження доступу до деяких або всіх баз даних.

Протокол DICT дозволяє клієнту отримувати від сервера переклад переданого серверу слова. Існуючі DICT-клієнти можуть використовувати як локальний сервер (при цьому словник знаходиться на локальній машині), так і віддалений (при цьому не потрібно тримати у себе на комп'ютері словник і запускати сервер, потрібні лише клієнт і підключення до Інтернету).

8.5. Finger - протокол надання інформації про користувачів віддаленого комп'ютера

Finger - мережевий протокол, призначений для надання інформації про користувачів віддаленого комп'ютера. Протокол Finger є простим протоколом (описаний в RFC 1288), який служить для отримання інформації про користувачів вузлів Інтернету.

Програма Finger може надати дані про список користувачів, які працюють в даний момент на сюжеті комп'ютері, про конкретного користувача (дата останнього сеансу входу в систему і т. д.), про список завантажених завдань, про типах інтерфейсів (наприклад, терміналів).

Первісна версія програми була написана Les Earnest в 1971 році. Кінцева редакція протоколу була підготовлена Earl Killian з Массачусетського Технологічного Інституту та Brian Harvey (SAIL).

Finger базується на Transmission Control Protocol, використовуючи TCP-порт 79. Даний протокол забезпечує інтерфейс для віддаленої інформаційної програми користувача (RUIP - Remote User Information Program). Локальна ЕОМ здійснює TCP-з'єднання з віддаленим вузлом через вказаний порт [8.1,

8.5]. Після цього стає доступною програма RUIP і користувач може посилати їй свої запити. Кожен запит являє собою рядок тексту. RUIP, отримавши запит, аналізує його і надсилає відповідь, після чого з'єднання закривається. Зазвичай з боку сервера протокол реалізований програмою «fingerd», а з боку клієнта - програмою «finger», яка надає інтуїтивний призначений для користувача інтерфейс. Наприклад, в системах Unix, команда finger @ foo.bar.net теоретично повертає список користувачів комп'ютера за адресою «foo.bar.net» (звичайно, тільки якщо на ньому запущена «fingerd»), а команда finger boris@foo.bar. net повертає повну інформацію про користувача «boris», включаючи ім'я, прізвище, телефон і зміст файлу «.plan» в його домашньому каталозі.

Будь-які пересилаються дані повинні мати формат ASCII, не мати контролю по парності і кожен рядок повинна завершуватися послідовністю CRLF (ASCII 13, за яким слід ASCII 10).

Програма *RUIP* повинна приймати будь-які запити *Finger*. Такі запити можуть мати такий вигляд:

finger [опція] [login1 [login2 ...]]

За замовчуванням команда *finger* виводить в список для кожного користувача системи *Unix* на даний момент:

• ім'я реєстрації в систему

• повне ім'я

• ім'я терміналу

• статус записи (при відсутності дозволу на запис перед термінальним ім'ям вказується символ «*»)

• час простою

• час реєстрації

• знаходження місця роботи і телефонний номер (якщо вони відомі)

Час простою обчислюється як час, що минув з моменту виконання будь-яких дій на даному терміналі.

Воно включає попередні виклики команди finger, яка, можливо,

модифікувала файл пристрою, який відповідає цьому терміналу.

Воно відображається в хвилинах, якщо воно виведено одним цілим числом, в годинах і хвилинах, якщо в його відображенні присутній двокрапка (:), або в днях і годинах, якщо у висновку присутній символ «d».

Таблиця 8.5. Опції - Формат запитів Finger

-b	Скорочений довгий формат виведення користувачів
-f	Пригнічує друк заголовка рядка (короткий формат)
-i	Швидкий список користувачів із зазначенням часу простоїв
-l	Викликає висновок в довгому форматі
-p	Пригнічує друк файлів .plan
-q	Швидкий список користувачів
-s	Викликає висновок в короткому форматі
-w	Викликає вузький форматний список зазначених користувачів

Крім того, існує більш довгий формат виведення і він використовується командою finger в тому випадку, якщо заданий список імен користувачів. (Допускаються поряд з першим і останнім іменами користувачів також і облікові імена.)

Цей формат складається з декількох рядків; він включає всю інформацію, зазначену вище, і, додатково, призначені для користувача вхідний каталог і інтерпретатор shell реєстрації, будь-який план, який користувач розмістив у файлі .plan в своєму вхідному каталозі, і проект, відповідно до якого задані користувачі працюють в даний момент, взятий з файлу .project, який також знаходиться у вхідному каталозі. Якщо в домашньому каталозі зазначеного користувача знаходиться файл .nofinger, то по команді finger інформація про цього користувача; не повертається.

8.6. FTP - протокол передачі файлів

Протокол передачі файлів (File Transfer Protocol, FTP) — дає можливість абоненту обмінюватися двійковими і текстовими файлами з будь-яким комп'ютером мережі, що підтримує протокол FTP. Установивши зв'язок з віддаленим комп'ютером, користувач може скопіювати файл з віддаленого комп'ютера на свій, або скопіювати файл зі свого комп'ютера на віддалений.

При розгляді FTP як сервісу Інтернет мають на увазі не просто протокол, а саме сервіс — доступ до файлів, які знаходяться у файлових архівах. FTP — стандартна програма, яка працює за протоколом TCP, яка завжди поставляється з операційною системою [8.1, 8.6].

Її початкове призначення — передача файлів між різними комп'ютерами, які працюють у мережах TCP/IP: на одному з комп'ютерів працює програма-сервер, на іншому — програма-клієнт, запущена користувачем, яка з'єднується з сервером і передає або отримує файли через FTP-сервіс.

Все це розглядається з припущенням, що користувач зареєстрований на сервері та використовує логін та пароль на цьому комп'ютері.

Ця риса послужила причиною того, що програми FTP стали частиною окремого сервісу Інтернету. Доволі часто сервер FTP налаштовується таким чином, що з'єднатися з ним можна не тільки під своїм ім'ям, але й під умовним іменем anonymous — анонім. У такому випадку для користувача стає доступною не вся файлова система комп'ютера, а лише деякий набір файлів на сервері, які складають вміст серверу anonymous FTP — публічного файлового архіву.

Якщо користувач хоче надати у вільне користування файли з інформацією, програмами і т. і., то йому достатньо організувати на власному комп'ютері, включеному в Інтернет, сервер anonymous FTP. Створення такого серверу — процес доволі простий, програми-клієнти FTP вельми розповсюджені, — тому сьогодні публічні файлові архіви організовані в

основному як сервери anonymous FTP. Перелік інформації, яка міститься на таких серверах, включає всі аспекти життя: від звичайних текстів до мультимедіа. Перша реалізація протоколу (1971 р.) передбачала обмін між клієнтом і сервером повідомленнями, що складаються з заголовка (72 біт) і даних змінної довжини.

Тема повідомлення включала в себе запит до FTP-сервера або відповідь від нього, тип і довжину переданих даних. Як дані передавалися параметри запиту (наприклад, шлях і ім'я файлу), інформація від сервера (наприклад, список файлів у каталозі) і самі файли. Таким чином, команди і дані передавалися по одному і тому ж каналу.

У 1972 р. протокол був повністю змінений, і прийняв вигляд, близький до сучасного. Команди з параметрами від клієнта та відповіді сервера передаються по TELNET-з'єднання (канал управління), для передачі даних створюється окреме з'єднання (канал даних).

У наступних редакціях була додана можливість роботи в пасивному режимі, передачі файлів між FTP-серверами, введені команди отримання інформації, зміни поточного каталогу, створення і видалення каталогів, збереження файлів під унікальним ім'ям. Деякий час існували команди для передачі електронної пошти через FTP, проте згодом вони були виключені з протоколу.

У 1980 р. FTP-протокол став використовувати TCP. Остання редакція протоколу була випущена в 1985 р.

У 1997 р. з'явилося доповнення до протоколу, що дозволяє шифрувати і підписувати інформацію в каналі управління і каналі даних.

У 1999 р. випущено додаток, присвячене інтернаціоналізації протоколу, яке рекомендує використовувати кодування UTF-8 для команд і відповідей сервера і визначає нову команду LANG, що встановлює мову відповідей.

Таблиця 8.6. Відмінність від HTTP

Властивість	FTP	HTTP
Базується на сесіях роботи	Так	Ні
Вбудована аутентифікація користувачів	Так	Ні
Спочатку передбачений для передачі	Великих двійкових файлів	Невеликих текстових файлів
Модель з'єднання	Подвійне підключення	Поодиноке підключення
Підтримує текстовий і двійковий режими передачі	Так	Ні
Підтримує вказівки типів даних, що передаються (MIME заголовки)	Ні	Так
Підтримує операції над файловою системою (mkdir, rm, rename, и т. д.)	Так	Ні

Досить яскрава особливість протоколу FTP в тому, що він використовує множинне (як мінімум - подвійне) підключення. При цьому один канал є керуючим, через який надходять команди сервера і повертаються його відповіді (зазвичай через TCP-порт 21), а через інші відбувається власне передача даних, по одному каналу на кожну передачу. Тому в рамках однієї сесії по протоколу FTP можна передавати одночасно кілька файлів, причому в обох напрямках.

Для кожного каналу даних відкривається свій TCP порт, номер якого вибирається або сервером, або клієнтом, в залежності від режиму передачі.

Протокол FTP (як і HTTP) має двійковий режим передачі, що скорочує накладні витрати трафіку і зменшує час обміну даними при передачі великих файлів.

Починаючи роботу через протокол FTP, клієнт входить в сесію, і всі операції проводяться в рамках цієї сесії (простіше кажучи, сервер пам'ятає поточний стан). Протокол HTTP нічого не «пам'ятає» - його завдання - віддати дані і забути, тому запам'ятовування стану при використанні HTTP здійснюється зовнішніми по відношенню до протоколу методами.

FTP працює на прикладному рівні моделі OSI і використовується для передачі файлів за допомогою TCP / IP. Для цього повинен бути запущений FTP-сервер, що очікує вхідних запитів. Комп'ютер-клієнт може зв'язатися з сервером по порту 21. Це з'єднання (потік управління) залишається відкритим на час сесії.

Друге з'єднання (потік даних), може бути відкритий як сервером з порту 20 до порту відповідного клієнта (активний режим), або ж клієнтом з будь-якого порту до порту відповідного сервера (пасивний режим), що необхідно для передачі файлу даних.

Потік управління використовується для роботи з сесією - наприклад, обмін між клієнтом і сервером командами і паролями за допомогою telnet-подібного протоколу.

Наприклад, «RETR ім'я файлу» передасть вказаний файл від сервера клієнту. Внаслідок цієї двухпортовой структури FTP вважається внешнеполосним протоколом, на відміну від внутрішньсмугового HTTP.

Різниця роботи пасивного режиму та активного:

Активний режим

Дії сервера і клієнта:

1.Клієнт встановлює зв'язок і надсилає запит на 21 порт сервера з порту N (N> 1024);

2.Сервер посилає відповідь на порт N (N> 1024) клієнта;

3.Сервер встановлює зв'язок для передачі даних по порту 20 на порт клієнта N +1.

Пасивний режим

Дії сервера і клієнта:

1.Клієнт встановлює зв'язок і надсилає запит (повідомляє, що треба працювати в пасивному режимі) на 21 порт сервера з порту N (N> 1024);

2.Сервер посилає відповідь і повідомляє номер порту для каналу даних P (P> 1024) на порт N (N> 1024) клієнта;

3.Клієнт встановлює зв'язок для передачі даних по порту N +1 на порт сервера P (P> 1024).

FTP може працювати в активному або пасивному режимі, від вибору якого залежить спосіб установки з'єднання [8.1, 8.5].

В активному режимі клієнт створює керуюче TCP-з'єднання з сервером і відправляє серверу свою IP-адресу і довільний номер клієнтського порту, після чого чекає, поки сервер запустить TCP-з'єднання з цим адресою і номером порту.

У разі, якщо клієнт знаходиться за брандмауером і не може прийняти вхідне TCP-з'єднання, може бути використаний пасивний режим. В цьому режимі клієнт використовує потік управління, щоб послати сервера команду PASV, і потім отримує від сервера його IP-адресу і номер порту, які потім використовуються клієнтом для відкриття потоку даних з довільного клієнтського порту до отриманого адресою і порту.

Обидва режими були оновлені у вересні 1998 р для підтримки IPv6. В цей час були проведені подальші зміни пасивного режиму, що поновили його до розширеного пасивного режиму.

З'єднання і передача даних.

При передачі даних по мережі можуть бути використані чотири представлення даних [8.1, 8.6]:

• ASCII - використовується для тексту. Дані, якщо необхідно, до передачі конвертуються з символьного подання на хості-відправника в «восьмібітних ASCII», і (знову ж таки, якщо необхідно) в символьне уявлення приймає хоста. Зокрема, змінюються символи перекладу рядка (CR / chr (13) /, LF / chr (10) / в Windows на LF / chr (10) / в Unix / Linux. Як наслідок, цей режим не підходить для файлів, що містять не тільки звичайний текст.

• Режим зображення (зазвичай званий бінарним) - пристрій-відправник посилає кожен файл байт за байтом, а одержувач зберігає потік байтів при отриманні. Підтримка даного режиму була рекомендована для всіх реалізацій

FTP.

• EBCDIC - використовується для передачі звичайного тексту між хостами в кодуванні EBCDIC. В іншому цей режим аналогічний ASCII-режиму.

• Локальний режим - дозволяє двом комп'ютерам з ідентичними установками посилати дані у власному форматі без конвертації в ASCII.

Для текстових файлів надані різні формати управління і настройки структури записи. Ці особливості були розроблені для роботи з файлами, що містять Telnet або ASA-форматування.

Передача даних може здійснюватися в будь-якому з трьох режимів:

• Поточний режим - дані надсилаються у вигляді безперервного потоку, звільняючи FTP від виконання будь-якої було обробки. Замість цього вся обробка виконується TCP. Індикатор кінця файлу не потрібен, за винятком поділу даних на записи.

• Блочний режим - FTP розбиває дані на кілька блоків (блок заголовка, кількість байт, поле даних) і потім передає їх TCP.

• Режим стиснення - дані стискаються єдиним алгоритмом (зазвичай кодуванням довжин серій).

FTP не розроблявся як захищений (особливо за нинішніми мірками) протокол і має численні уразливості в захисті. У травні 1999 автори RFC 2577 звели уразливості в наступний список проблем:

• Приховані атаки (bounce attacks)

• Спуф-атаки (spoof attacks)

• Атаки методом грубої сили (brute force attacks)

• Перехоплення пакетів, сніффінг (packet capture, sniffing)

• Захист імені користувача

• Захоплення портів (port stealing)

FTP не може зашифрувати свій трафік, всі передачі - відкритий текст, тому імена користувачів, паролі, команди і дані можуть бути прочитані ким завгодно, здатним перехопити пакет по мережі. Ця проблема характерна для

багатьох специфікацій Інтернет-протоколу (в їх числі SMTP, Telnet, POP, IMAP), розроблених до створення таких механізмів шифрування, як TLS і SSL. Звичайне рішення цієї проблеми - використовувати «безпечні», TLS-захищені версії вразливих протоколів (FTPS для FTP, TelnetS для Telnet і т. д.) Або ж інший, більш захищений протокол, начебто SFTP / SCP, що надається з більшістю реалізацій протоколу Secure Shell.

Явний FTPS - розширення стандарту FTP, що дозволяє клієнтам вимагати того, щоб FTP-сесія була зашифрована. Це реалізується відправкою команди «AUTH TLS». Сервер має можливість дозволити або відхилити з'єднання, які не можуть звертатися TLS. Це розширення протоколу визначено в специфікації RFC 4217.

Неявний FTPS - застарілий стандарт для FTP, що вимагає використання SSL- або TLS-з'єднання. Цей стандарт повинен був використовувати відмінні від звичайного FTP порти.

SFTP, або «SSH File Transfer Protocol», не пов'язаний з FTP, за винятком того, що він теж передає файли і має аналогічний набір команд для користувачів.

SFTP, або безпечний FTP, - це програма, яка використовує SSH (Secure Shell) для передачі файлів. На відміну від стандартного FTP він шифрує і команди, і дані, оберігаючи паролі і конфіденційну інформацію від відкритої передачі через мережу. За функціональністю SFTP схожий на FTP, але так як він використовує інший протокол, клієнти стандартного FTP не можуть зв'язатися з SFTP-сервером і навпаки[8.1, 8.6]. Основні команди:

ABOR - Перервати передачу файлу

• *CDUP* - Змінити каталог на вищій.

• *CWD* - Змінити каталог.

• *DELE* - Видалити файл (DELE filename).

• *EPSV* - Увійти в розширений пасивний режим. Застосовується замість PASV.

• *HELP* - Виводить список команд, які приймаються сервером.

• *LIST* - Повертає список файлів каталогу. Список передається через з'єднання даних.

• *MDTM* - Повертає час модифікації файлу.

• *MKD* - Створити каталог.

• *NLST* - Повертає список файлів каталогу в більш короткому форматі, ніж LIST. Список передається через з'єднання даних.

• *NOOP* - Порожня операція.

• *PASS* - Пароль.

PASV - Увійти в пасивний режим. Сервер поверне адресу і порт, до якого потрібно підключитися, щоб забрати дані. Передача почнеться при введенні наступних команд: RETR, LIST і т. Д.

• *PORT* - Увійти в активний режим. Наприклад PORT 12,34,45,56,78,89. На відміну від пасивного режиму для передачі даних сервер сам підключається до клієнта.

• *PWD* - Повертає поточний каталог.

• *QUIT* - Кінець сеансу.

• *REIN* - Реініціалізіровать підключення.

• *RETR* - Завантажити файл. Перед RETR повинна бути команда PASV або PORT.

• *RMD* - Видалити каталог.

• *RNFR і RNTO* - Перейменувати файл. RNFR - що перейменовувати, RNTO - будь-що.

• *SIZE* - Повертає розмір файлу.

• *STOR* - Завантажити файл. Перед STOR повинна бути команда PASV або PORT.

• *SYST* - Повертає тип системи (UNIX, WIN, ...).

• *TYPE* - Встановити тип передачі файлу (бінарний, текстовий).

• *USER* - Ім'я користувача для входу на сервер.

8.7. FXP - протокол обміну файлами

FXP (File eXchange Protocol — протокол обміну файлами) — спосіб передачі файлів між двома FTP-серверами напряму, не завантажуючи їх на свій комп'ютер. При FXP-сесії клієнт відкриває два FTP-з'єднання до двох різних серверів, запитуючи файл на першому сервері, вказуючи в команді PORT IP-адресу другого сервера [8.4, 8.7].

Безперечною перевагою підтримки стандарту FXP є те, що на кінцевих користувачів, охочих скопіювати файли з одного FTP-сервера на інший, вже не діє обмеження пропускної спроможності їх власного інтернет-з'єднання. Немає необхідності завантажувати собі файл, щоб потім завантажити його на інший FTP-сервер. Таким чином, час передачі файлів буде залежати тільки від швидкості з'єднання між двома віддаленими FTP-серверами, яка в більшості випадків більша ніж у «користувача».

FXP став використовуватися зловмисниками для атак на інші сервери: в команді PORT вказується IP-адреса і порт сервісу що атакується на комп'ютері жертви, і командами RETR / STOR проводиться звернення на цей порт від особи FTP-сервера, а не атакуючої машини, що дозволяло влаштовувати масштабні DDoS-атаки з використанням відразу багатьох FTP-серверів, або обходити систему безпеки комп'ютера жертви, якщо він покладається тільки на перевірку IP клієнта і використовуваний для атаки FTP-сервер знаходиться в довіреної мережі або на шлюзі.

Відтак зараз практично усі сервери перевіряють відповідність IP-адреси, вказаної в команді PORT, IP-адресу FTP-клієнта і за замовчуванням забороняють використання там IP-адрес третіх сторін. Таким чином, використання FXP неможливо при роботі з публічними FTP-серверами.

Очевидно, що таке кодування ефективно для даних, що містять велику кількість серій, наприклад, для простих графічних зображень, таких як іконки і графічні малюнки. Однак це кодування погано підходить для зображень з

плавним переходом тонів, таких як фотографії.

Поширені формати для упаковки даних за допомогою RLE включають в себе PackBits, PCX і ILBM.

Методом кодування довжин серій можуть бути стиснуті довільні файли з двійковими даними, оскільки специфікації на формати файлів часто включають в себе повторювані байти в області вирівнювання даних. Проте, сучасні системи стиснення (наприклад, Deflate) частіше використовують алгоритми на основі LZ77, які є узагальненням методу кодування довжин серій і оперують з послідовностями символів виду «BWWBWWBWWBWW».

Звукові дані, які мають довгі послідовні серії байт (такі як низькоякісні звукові семпли) можуть бути стиснуті за допомогою RLE після того, як до них буде застосовано Дельта-кодування.

8.8. Gopher - протокол розподіленого пошуку і передачі документів

Gopher - мережевий протокол розподіленого пошуку і передачі документів, який був широко поширений в Інтернеті до 1993 року. Протокол призначався для надання доступу до документів в Інтернет, але мав менше можливостей, ніж HTTP, і згодом був ним повністю витіснений. Gopher — засіб пошуку інформації в мережі Інтернет, що дозволяє знаходити інформацію за ключовими словами і фразами. Робота із системою Gopher нагадує перегляд змісту, при цьому користувачу пропонується пройти крізь ряд вкладених меню і вибрати потрібну тему [8.8 - 8.10].

Gopher дозволяє одержати інформацію без вказівки імен і адрес авторів, завдяки чому користувач не витрачає багато часу і нервів. Він просто повідомляє систему Gopher, що саме йому потрібно, і система знаходить відповідні дані. Gopher-серверів понад дві тисячі, тому з їхньою поміччю не завжди просто знайти необхідну інформацію. Якщо виникли ускладнення, можна скористатися службою VERONICA. VERONICA здійснює пошук більш

ніж у 500 системах Gopher, звільнюючи користувача від необхідності переглядати їх вручну.

Хоча нині його розвиток йде набагато повільніше інших сервісів схожого призначення, але, все ж таки, через Gopher можна отримати доступ до великої кількості інформації — в першу чергу, з історичних причин, — був період, коли Gopher був найкращим засобом для передачі інформації і деякі організації та компанії досі продовжують його використовувати.

Сучасні засоби роботи з інформацією в Internet забезпечують доступ і до серверів Gopher, тому немає необхідності вивчення методів роботи із спеціальними програмами-клієнтами Gopher.

Gopher — це розподілена система експорту структурованої інформації. При роботі з Gopher користувач знаходиться у системі вкладених меню, з яких доступні файли різних типів — як правило, прості тексти, але це може бути і графіка і звук і будь-які інші типи файлів. Таким чином, для публічного доступу надаються файли з інформацією не у вигляді файлової структури, як в FTP, а в вигляді анотованої деревоподібної структури.

Gopher — сервіс прямого доступу і потребує, щоб і сервер, і клієнт були повноцінно підключені до Internet.

8.9. IRC - технологія багатокористувацьких конференцій в текстовому режимі через мережу Інтернет

Протокол IRC (Internet Relay Chat) створив у 1988 році фінський вчений і програміст Яркко Ойкарінен (Jarkko Oikarinen).

При підключенні до серверу IRC користувач бачить список доступних каналів, у кожний з яких (або відразу в декілька) він може «увійти» (підключитися). Каналом є віртуальна «кімната», в якій можуть знаходитися декілька користувачів. Всі повідомлення, що видаються в канал, видно всім користувачам, які знаходяться на цьому ж каналі. Кожен канал має свою назву і,

як правило, певну тему для обговорення. Після «входу» на канал користувач може бачити, що пишуть інші учасники каналу, а також може сам писати повідомлення. Тема, що обговорюється на каналі, зазвичай випливає з його назви (наприклад, канал #wikipedia-uk) [8.11 - 8.13].

Різні сервери можуть об'єднуватися (лінкуватися) в мережу з єдиним простором імен користувачів і каналів. Великі світові IRC-мережі налічують у своєму складі сотні серверів. Аварійний тимчасовий розрив IRC-мережі на дві частини називається netsplit'ом (сплітом).

IRC надає можливість як групового, так і приватного спілкування. Для групового чату в IRC призначені канали, на яких користувачі можуть збиратися та вести спілкування.

Оператори IRC-мережі керують роботою серверів та мережі в цілому. Як правило, в IRC-мережах на операторів теж встановлюються обмеження на рівні правил мережі, мережевого етикету або навіть на рівні IRCd/IRC-сервісів.

Можна задавати різні режими каналів за допомогою команди MODE.

У RFC 2811 визначені такі режими:

+o user — позначає оператора каналу.

+v user — дає користувачеві право говорити на модерованих каналах (див. + m).

+a — анонімний канал. Імена всіх користувачів ховаються як anonymous!anonymous@anonymous (відсутня в багатьох реалізаціях).

+m — тільки користувачі з прапорами +o, +h, або +v можуть посилати в нього повідомлення.

+n — тільки що знаходяться на каналі користувачі можуть посилати в нього повідомлення.

+p/+s —- канал ховається в усіх відповідях сервера якщо користувач не знаходиться на цьому каналі.

+t — тему каналу можуть змінювати тільки оператори.

+l limit — обмежує кількість користувачів на каналі числом limit.

+k key — встановлює ключ (пароль) на канал key.

+i — на канал можна увійти тільки за запрошенням (invite).

+b — вивести список банів +b на каналі. Доступний всім користувачам.

+c — оформлення тексту на каналі заборонено (не визначено в RFC).

Режими користувачів

+i — невидимий користувач.

+s — отримувати повідомлення сервера.

+w — отримувати wallops.

+o — оператор сервера. Для отримання повинна використовуватися команда OPER.

8.10. POP3 - протокол отримання вхідних листів з віддаленого сервера

POP3 (Post Office Protocol) - протокол для отримання вхідних листів з віддаленого сервера, його підтримують всі поштові клієнти і поштові сервери.

Якщо по протоколу SMTP листи тільки транспортуються, то за допомогою протоколу POP3 ви можете отримати ці листи з сервера і прочитати їх за допомогою поштового клієнта.

Основним недоліком цього протоколу є те, що для перегляду листів, ці листи завантажуються з сервера і зберігаються локально на комп'ютері користувача. За замовчуванням, при роботі з цим протоколом, викачані листи з сервера видаляються, хоча цю опцію можна відключити в налаштуваннях поштового клієнта [8.14, 8.15]. Робота протоколу відбувається по 110 порту і по 995 порту, при захищеному SSL-з'єднанні.

Робота протоколу виглядає дуже просто:

• Авторизація на POP3 сервері.

• Отримання інформації про стан поштової скриньки і скачування листів.

• Оновлення інформації на сервері (за фактом видалення завантажених

листів) і закриття з'єднання.

Оскільки використання цього протоколу для отримання пошти не дуже зручно, був розроблений альтернативний, більш зручний протокол IMAP.

IMAP (Internet Message Access Protocol) - більш сучасний і зручний ніж POP3 протокол, для керування електронною поштою на віддаленому сервері. Даний протокол використовує для підключення 143 порт. Хоча цей протокол і підтримує можливість відправки пошти, в більшості випадків для відправки використовується більш надійний протокол SMTP. Якщо порівнювати його з POP3, IMAP має цілий ряд переваг:

• На відміну від POP3, який просто викачує вхідні листи і зберігає їх локально, з IMAP ви працюєте з поштою безпосередньо на сервері. Це позбавляє від необхідності завантажувати дані з сервера і передавати їх назад. При використанні поштового клієнта, це виглядає так, як буд то листи знаходяться локально на вашому комп'ютері.

• При використанні POP3, підключення до сервера виробляється на невеликий термін, необхідний для завантаження нових листів. IMAP дозволяє постійно тримати з'єднання відкритим і завантажувати оновлювати інформацію про поштовій скриньці на вимогу.

POP3 вимагає підключення тільки одного клієнта до поштової скриньки. З IMAP можна підключатися і працювати з одним поштових скриньках одночасно декільком користувачам і бачити зміни, внесені іншими користувачами. Це корисно, коли необхідно, що б одним поштовою скринькою користувалися кілька чоловік одночасно [8.14 - 8.16].

• IMAP використовує систему прапорів для листів, що визначають їх стан (прочитано, відповіли, видалено, важливе, чернетка, недавнє). Це дозволяє більш зручно працювати з поштою за допомогою поштового клієнта.

• IMAP дозволяє працювати не тільки з листами, але і з поштовими скриньками. Доступні такі операції як видалення, створення, перейменування поштових скриньок, переміщення листів між ящиками. Також можлива

настройка доступу до поштових скриньок.

Загалом, якщо ви плануєте працювати з поштою через поштовий клієнт, рекомендується використовувати саме цей протокол.

8.11. SMB - протокол розділеного доступу елементів мережі

Server Message Block (SMB) — протокол прикладного рівня (в моделі OSI), зазвичай використовується для надання розділеного доступу до файлів, принтерів, послідовних портів передачі даних, та іншої взаємодії між вузлами в комп'ютерній мережі [8.15, 8.17].

Також надає можливості міжпроцесної взаємодії з аутентифікацією. Зазвичай, використовується на комп'ютерах з Microsoft Windows: в середовищі Microsoft, часто позначається як «Microsoft Windows Network».

Слід розрізняти:

- протокол SMB
- сервіси, що використовують протокол SMB
- NetBIOS
- сервіси DCE/RPC, які використовують SMB як канал міжпроцесної взаємодії з аутентифкацією (через іменовані канали)
- протоколи мережевого оточення (англ. Network Neighborhood), які, в основному (але не виключно) працюють через канали датаграм використовуючи транспортний рівень NetBIOS.

У 1996 році Microsoft стала використовувати нову назву для доповненої версії протоколу, яка використовувалася в Windows NT 4.0 — CIFS (Common Internet File System); нове ім'я прижилося, і SMB та CIFS фактично стали синонімами.

Багато людей вважають, що SMB протокол перевантажує мережу, тому що кожний клієнт передає свою присутність до всієї мережі (broadcasts — широкомовне повідомлення). SMB сам по собі не використовує широкомовних

повідомлень.

Проблем широкомовних повідомлень зазвичай зв'язані з NetBIOS (мережевим біосом). Оскільки за замовчуванням Microsoft Windows server використовує широкомовне повідомлення для публікації та знаходження сервісів. NetBIOS використовує широкомовні повідомлення через певні проміжки часу.

Якщо мережа має не більше 20 комп'ютерів проблем не виникає, але при збільшенні числа комп'ютерів виникають проблеми.

Правильне впровадження NetBIOS Name Server (NBNS) може зменшити цю проблему — наприклад Windows Internet Naming Service (WINS) пропонує прийнятне рішення для систем від Майкрософт [8.14 - 8.17].

Microsoft впровадив нову версію Server Message Block (SMB) протоколу (SMB 2.0 чи SMB2) з Windows Vista у 2006 році.

SMB2 покращує попередні версії SMB2 для Windows додаючи можливість групувати декілька дій в одиничний запит, що суттєво зменшує число повторних звертань (round-trips) до сервера. SMB1 також має сумуючий механізм відомий як AndX щоб компонувати декілька дій, але Microsoft clients рідко використовують AndX. SMB2 підтримує більше розміри буфера, що ефективно при передачі великих файлів.

SMB2 вводить так звані міцні файл хендли (durable file handles) це дозволяє конектам до SMB сервера переживати короткі перерви у роботі мережі, такі які наприклад можуть траплятися у безпровідній мережі, без того щоб починати нову сесію. SMB2 включає підтримку для символічних лінків (symbolic links).

SMB 1 часто використовує 16-bit розміри. SMB2 використовує 32 чи 64 bits у багатьох випадках, і 16 байтові для файлових хендлів.

8.12. SSH - протокол віддаленного управління комп'ютером і тунелювання TCP-з'єднань

Secure Shell, SSH (Secure SHell — «безпечна оболонка») — мережевий протокол рівня застосунків, що дозволяє проводити віддалене управління комп'ютером і тунелювання TCP-з'єднань (наприклад, для передачі файлів). Схожий за функціональністю з протоколом Telnet і rlogin, проте шифрує весь трафік, в тому числі і паролі, що передаються.

Криптографічний захист протоколу SSH не фіксований, можливий вибір різних алгоритмів шифрування. Клієнти і сервери, що підтримують цей протокол, доступні для різних платформ. Крім того, протокол дозволяє не тільки використовувати безпечний віддалений shell на машині, але і туннелювати графічний інтерфейс — X Tunnelling (тільки для Unix-подібних ОС або застосунків, що використовують графічний інтерфейс X Window System). Так само ssh здатний передавати через безпечний канал (Port Forwarding) будь-який інший мережевий протокол, забезпечуючи (при належній конфігурації) можливість безпечної пересилки не тільки X-інтерфейсу, але і, наприклад, звуку [8.18].

Підтримка SSH реалізована у всіх UNIX системах, і на більшості з них в числі стандартних утиліт присутні клієнт і сервер ssh. Існує безліч реалізацій SSH-клієнтів і для не - UNIX ОС. Велику популярність протокол отримав після широкого розвитку сніферів, як альтернативне небезпечному телнету рішення для управління важливими вузлами.

Зараз відомо дві гілки версій — 1 і 2. Проте гілка 1 зупинена, оскільки наприкінці 90-х у ній було знайдено багато вразливостей, деякі з яких досі накладають серйозні обмеження на її використання, тому перспективною (такою, що розвивається) і найбезпечнішою є версія 2.

Перша версія протоколу, SSH-1, була розроблена в 1995 році дослідником Тату Ілоненом з Технологічного університету Хельсінкі,

Фінляндія. SSH-1 був написаний для забезпечення більшої конфіденційності, ніж протоколи rlogin, telnet і rsh. У 1996 році була розроблена безпечніша версія протоколу, SSH-2, несумісна з SSH-1. Протокол набув ще більшої популярності, і до 2000 року у нього було близько двох мільйонів користувачів. В даний час під терміном «SSH» зазвичай мається на увазі саме SSH-2, тому що перша версія протоколу з огляду істотних недоліків зараз практично не застосовується.

У 2006 році протокол був затверджений робочою групою IETF як Інтернет-стандарт.

Проте, в деяких країнах (Франція, Росія, Ірак і Пакистан) до сих пір потрібен спеціальний дозвіл у відповідних структурах для використання певних методів шифрування, включаючи SSH. Див закон Російської Федерації «Про федеральних органах урядового зв'язку і інформації» (закон втратив силу з 1 липня 2003 року у зв'язку з прийняттям федерального закону від 30.06.2003 № 86-ФЗ).

Поширені дві реалізації SSH: пропріетарна комерційна та безкоштовна вільна. Вільна реалізація називається OpenSSH. До 2006 року 80 % комп'ютерів мережі Інтернет використовувало саме OpenSSH. Пропріетарна реалізація розробляється організацією SSH Inc., Вона безкоштовна для некомерційного використання. Ці реалізації містять практично однаковий набір команд.

Протокол SSH-2, на відміну від протоколу telnet, стійкий до атак прослуховування трафіку («сніфінг»). Протокол SSH-2 також стійкий до атак шляхом приєднання посередині (session hijacking) — неможливо включитися у вже встановлену сесію або перехопити її.

Для запобігання атак «людина посередині» при підключенні до хосту, ключ якого ще не відомий клієнту, клієнтське ПО показує користувачеві «зліпок ключа» (key fingerprint). Рекомендується ретельно перевіряти показуваний клієнтським ПО «зліпок ключа» (key fingerprint) зі зліпком ключа сервера, бажано отриманим по надійним каналах зв'язку або особисто [8.18].

Підтримка SSH реалізована у всіх UNIX-подібних системах, і на

більшості з них в числі стандартних утиліт присутні клієнт і сервер ssh. Існує безліч реалізацій SSH-клієнтів і для не-UNIX ОС. Велику популярність протокол отримав після широкого розвитку аналізаторів трафіку і способів порушення роботи локальних мереж, як альтернативне небезпечному протоколу Telnet рішення для управління важливими вузлами.

Для роботи по SSH потрібен SSH-сервер і SSH-клієнт. Сервер прослуховує з'єднання від клієнтських машин і при встановленні зв'язку виробляє аутентифікацію, після чого починає обслуговування клієнта. Клієнт використовується для входу на віддалену машину і виконання команд.

Для з'єднання сервер і клієнт повинні створити пари ключів — відкритих і закритих — і обмінятися відкритими ключами. Зазвичай використовується також і пароль.

Клієнти та оболонки:

- GNU/Linux, *BSD: kdessh, lsh-client, openssh-client, putty, ssh, Vinagre
- MS Windows і Windows NT: PuTTY, SecureCRT, ShellGuard, Axessh, ZOC, SSHWindows, ProSSHD, XShell
- MS Windows Mobile: PocketPuTTy, mToken, sshCE, PocketTTY, OpenSSH, PocketConsole
- Mac OS: NiftyTelnet SSH
- Symbian OS: PuTTY
- Java: MindTerm, AppGate Security Server
- J2ME: MidpSSH
- iOS: i-SSH, ssh (в комплекті з Terminal)
- Android: connectBot
- Blackberry: BBSSH
- Maemo 5: OpenSSH.

SSH це протокол, який може бути використаний для багатьох додатків на різних платформах, включаючи самі Unix варіантів (Linux, BSD автора, включаючи від Apple OS X, і Solaris), а також Microsoft Windows. Деякі

програми можуть зажадати нижче функції, які доступні тільки або сумісним з конкретними клієнтами SSH або серверів. Наприклад, з використанням протоколу SSH для реалізації VPN можна, але в даний час тільки з OpenSSH сервер і клієнт реалізації:

- Для входу в оболонку на віддаленому хості (заміна Telnet і Rlogin)
- Для виконання однієї команди на віддаленому хості (заміна RSH)
- Безпечна передача файлів
- У поєднанні з Rsync для резервного копіювання, скопіюйте та дзеркала файлів ефективно і безпечно
- Для експедирування, доставка або тунельні порт (не плутати з VPN, які маршрутів пакети між різними мережами, або мости два широкомовний домен и в один).
- Для використання як повноцінного зашифрованого VPN. Зверніть увагу, що тільки OpenSSH сервер і клієнт підтримує цю функцію.
- Для переадресації X з віддаленого хости (можливо через кілька проміжних хазяїв)
- Для перегляду веб-сторінок через шифроване з'єднання з проксі SSH клієнти, які підтримують SOCKS протокол.
- Для безпечного монтажу каталогу віддаленого сервера на локальному комп'ютері за допомогою SSHFS.
- Для автоматизованого дистанційного моніторингу та управління серверами через одну або більше механізмів, описаних вище.
- Для розвитку на мобільні або вбудовані пристрої, які підтримують SSH.

Є кілька механізмів передачі файлів за допомогою захищених протоколів Shell:

- Secure Copy (SCP), який відбувся від RCP Протокол по SSH;
- Rsync, повинен бути більш ефективним, ніж SCP;
- SSH File Transfer Protocol (SFTP), безпечна альтернатива FTP (не

плутати з FTP через SSH);

- Файли передаються по протоколу оболонки (так званий FISH), випущений в 1998 році, який перетворився з Unix Shell команди через SSH.

Команда підключення до локального SSH-сервера з командного рядка GNU / Linux або FreeBSD для користувача pacify (сервер прослуховує нестандартний порт 30000): $ ssh -p 30000 pacify@127.0.0.1

Генерація пари ключів (в UNIX-подібних ОС) здійснюється командою $ ssh-keygen

Генерація пари SSH-2 RSA-ключів довжиною 4096 біта програмою puttygen під UNIX-подібними ОС: $ puttygen -t rsa -b 4096 -o sample

Деякі клієнти, наприклад, PuTTY, мають і графічний інтерфейс користувача.

Для використання SSH в Python існують такі модулі, як python-paramiko і python-twisted-conch [8.14 - 8.18].

Поради щодо безпеки використання SSH:

- Заборона віддаленого root-доступу.
- Заборона підключення з порожнім паролем або відключення входу по паролю.
- Вибір нестандартного порту для SSH-сервера.
- Використання довгих SSH2 RSA-ключів (2048 біт і більше). Системи шифрування на основі RSA вважаються надійними, якщо довжина ключа не менше 1024 біт.
- Обмеження списку IP-адрес, з яких дозволений доступ (наприклад, настроюванням файєрвола).
- Заборона доступу з деяких потенційно небезпечних адрес.
- Відмова від використання поширених або широко відомих системних логинів для доступу по SSH.
- Регулярний перегляд повідомлень про помилки аутентифікації.
- Установка систем виявлення вторгнень (IDS — Intrusion Detection

System).

Використання пасток, підроблюють SSH-сервіс (honeypots).

SSH-тунель — це тунель, який створюється за допомогою SSH-з'єднання і використовується для шифрування тунельованих даних. Використовується для того, щоб убезпечити передачу даних в Інтернеті (аналогічне призначення має IPsec). Особливість полягає в тому, що незашифрований трафік будь-якого протоколу шифрується на одному кінці SSH-з'єднання і розшифровується на іншому.

Практична реалізація може виконуватися декількома способами:

- Створенням Socks-проксі для програм, які не вміють працювати через SSH-тунель, але можуть працювати через Socks-проксі
- Використанням додатків, які вміють працювати через SSH-тунель.
- Створенням VPN-тунелю, підходить практично для будь-яких додатків.

Якщо програма працює з одним певним сервером, можна налаштувати SSH-клієнт таким чином, щоб він пропускав через SSH-тунель TCP-з'єднання, що приходять на певний TCP-порт машини, на якій запущений SSH-клієнт. Наприклад, клієнти Jabber підключаються по замовчуванню на порт 443. Тоді, щоб налаштувати підключення до сервера Jabber через SSH-тунель, SSH-клієнт налаштовується на перенаправлення підключень з будь-якого порту локальної машини (наприклад, з порту 4430) на віддалений сервер (наприклад, jabber.example.com і порт 443): $ ssh -L 4430:jabber.example.com:443 somehost.

В даному випадку Jabber-клієнт налаштовується на підключення до порту 4430 сервера localhost (якщо ssh-клієнт запущений на тій же машині що і Jabber-клієнт).

Для створення ssh-тунелю необхідна машина з запущеним ssh-сервером і доступом до jabber.example.com. Така конфігурація може використовуватися у випадку, якщо з локальної машини доступ до jabber.example.com закритий файєрволом, але є доступ до деякого ssh-серверу, у якого обмеження доступу в

Інтернет відсутні.

SSH — це протокол прикладного рівня. SSH-сервер зазвичай прослуховує з'єднання на TCP-порту 22.

Специфікація протоколу SSH-2 міститься в RFC 4251. Для аутентифікації сервера в SSH використовується протокол аутентифікації сторін на основі алгоритмів електронно-цифрового підпису RSA або DSA.

Для аутентифікації клієнта також може використовуватися ЕЦП RSA або DSA, але допускається також аутентифікація за допомогою пароля (режим зворотної сумісності з Telnet) і навіть ip-адреси хоста (режим зворотної сумісності з rlogin). Аутентифікація за паролем найбільш поширена, вона безпечна, тому що пароль передається по зашифрованому віртуального каналу. Аутентифікація по ip-адресою небезпечна, цю можливість найчастіше відключають.

Для створення спільного секрету (сеансового ключа) використовується алгоритм Діффі — Хеллмана (DH). Для шифрування переданих даних використовується симетричне шифрування, алгоритми AES, Blowfish або 3DES. Цілісність переданих даних перевіряється за допомогою CRC32 в SSH1 або HMAC-SHA1/HMAC-MD5 в SSH2.

Для стиснення шифруючих даних може використовуватися алгоритм LempelZiv (LZ77), який забезпечує такий же рівень стиснення, що і архіватор ZIP. Стиснення SSH включається лише за запитом клієнта, і на практиці використовується рідко [8.17 - 8.19].

SSH зазвичай використовується для входу на віддалену машину і виконувати команди, але він також підтримує тунельний протокол, портове експедирування TCP і UDP порт | TCP порти. він може передавати файли за допомогою відповідних SSH File Transfer Protocol | SSH File Transfer (SFTP) або Secure Copy (SCP) протоколів SSH використовує клієнт-сервер модель.

SSH програма зазвичай використовується для встановлення з'єднання з ухвалення віддалених підключень. І те й інше зазвичай присутнє на більшості

сучасних операційні системи, включаючи Mac OS, більшість розподілів GNU / Linux, OpenBSD, FreeBSD, NetBSD, Solaris і OpenVMS. Відомо, що Windows є одним з небагатьох сучасних / серверних операційних систем, які не включають SSH за замовчуванням.

SSH відіграє важливу роль в обчислень для вирішення проблем з підключенням, уникаючи питань безпеки, піддаючи віртуальну машину безпосередньо до Інтернету.

Тунель SSH може забезпечити безпечний шлях через Інтернет, через брандмауер на віртуальній машині.

Версія 1.x

У 1995 році Tatu Ylonen, дослідник Гельсінського технологічного університету, (Фінляндія), розробив першу версію протоколу (зараз він називається 'SSH-1'). Мета SSH повинен був замінити раніші Rlogin, TELNET і RSH протоколи, які не забезпечують надійну аутентифікацію, і не гарантують конфіденційність. Ylonen випустив свій протокол безкоштовно в липні 1995 року, і інструмент швидко завоював популярність. До кінця 1995 року база SSH користувачів виросло до 20.000 користувачів в п'ятдесяти країнах.

У грудні 1995 року заснував Ylonen SSH Communications Security.

Оригінальна версія програмного забезпечення SSH використовували різні частини вільне програмне забезпечення, наприклад, GNU libgmp, але більш пізні версії випущена SSH Secure Communications перетворилася в більш пропрієтарне програмне забезпечення.

У 1998 році вразливість була описана в SSH 1.5, що дозволяло несанкціонованого вставки контенту в зашифрованому потоці SSH через недостатню цілісності даних захист від CRC-32 використовується в даній версії протоколу. У січні 2001 року була виявлена уразливість, яка дозволяє зловмисникові змінювати останній блок IDEA. У тому ж місяці, інша уразливість була виявлена, що дозволяло шкідлививому серверу пересилання

аутентифікації клієнта на інший сервер. Так як SSH-1 притаманні недоліки дизайну, які роблять його вразливим, в даний час він вважається застарілим і його слід уникати, відключивши повернення до SSH-1. Більшість сучасних серверів і клієнтів підтримують SSH-2.

Версія 1.99

У січні 2006 року, після версії 2.1 був створений, RFC 4253, що означало, що сервер SSH, який підтримує як 2.0 так і попередні версії SSH повинні визначити свої версії як 1,99. Це не фактичні версії, але метод ідентифікації зворотна сумісність.

OpenSSH і OSSH

У 1999 році розробники, бажали повернутися до старої 1.2.12 релізовуючи оригінальну SSH програму, яка була останньою випущена під з відкритим вихідним кодом ліцензію. OSSH Björn Grönvall згодом почав розвиватися з цього коду.

Версія 2.x

«Secsh» була офіційним ім'ям робочої групи IETF, відповідальної за версію 2 протоколу SSH.

У 2006 році переглянуто версії протоколу і SSH-2 був прийнятий як стандарт. Ця версія несумісна з SSH-1. SSH-2 порівняно з SSH-1 має кращі функції безпеки, так і інші поліпшення.

Краща захищеність, проявляється у використанні протоколу Діффі-Геллмана і сильнішій цілісності, коли перевірка відбувається за допомогою MAC-підписів. Нові можливості SSH-2 включають в себе можливість запускати будь-яку кількість оболонок сесій на одному SSH з'єднанні. Через переваги SSH-2 над SSH-1 та більшу популярність, такі реалізації SSH-2 як libssh (v0.8.0+), Lsh and Dropbear підтримують тільки протокол SSH-2.

У листопаді 2008 року теоретичну вразливість була виявлена для всіх версій SSH, який дозволив відновлення до 32 біт відкритого тексту з блоку

зашифрованого тексту, зашифрованого за допомогою тодішнього стандартного режиму шифрування за умовчанням.

Пізніше протокол SSH був змінений і розширений в наступних публікаціях:

- RFC 4419, Діффі-Хеллмана Група обмін на Secure Shell (SSH) Transport Layer Protocol (березень 2006 р.)
- RFC 4432, RSA Key Exchange для Secure Shell (SSH) Transport Layer Protocol (березень 2006 р.)
- RFC 4462, Generic Security Service Application Program Interface (GSS-API) аутентифікації і обміну ключами для Secure Shell (SSH) Protocol (травень 2006 р.)
- RFC 4716, Secure Shell (SSH) Public Key Формат файлу (листопад 2006)
- RFC 5656, Еліптичні алгоритм інтегрування кривій в безпеці транспортного рівня Shell (грудень 2009 року).

SSH-2 протокол має внутрішню архітектуру (визначено в RFC 4251) з чітко розділеними шарами. До них відносяться: Транспортний шар (RFC 4253).

Цей шар обробляє початковий обмін ключами, автентифікує сервер, встановлює шифрування, стиснення і перевірку цілісності даних. Він надає у вищий шар інтерфейс для відправки та отримання текстових пакетів розміром до 32768 байт кожен (цей розмір може бути збільшений в певній реалізації). Транспортний рівень також організовує повторний обмін ключами. Як правило, це відбувається після передачі 1 Гб даних або коли пройшла 1 година, що швидше настане.

Шар автентифікації користувача (RFC 4252). Цей шар обробляє перевірку автентичності клієнта і надає ряд методів аутентифікації. Автентифікація є клієнто-орієнтованою: коли користувач отримує запит на введення пароля, то це може бути запит SSH клієнта, а не сервера. Сервер просто відповідає на запити клієнта про автентифікацію. Широко вживані методи автентифікації

користувачів включають наступне:

Пароль: Метод простий пароль аутентифікації, в тому числі засіб, що дозволяє пароль, щоб бути змінені. Цей метод не реалізований всіма програмами;

З відкритим ключем: метод відкритого ключа перевірки автентичності на основі, як правило, підтримують принаймні, DSA або RSA пари ключів, з іншими реалізаціями також підтримує X.509 сертифікати.

Інтерактивний (RFC 4256): універсальний метод, коли сервер відправляє один або декілька запитів вводу інформації і клієнт відображає їх і відправляє назад відповідь введені в користувачем. Використовується для забезпечення одноразовий пароль аутентифікації, такі як S / Key або SecurID. Використовується деяких конфігураціях OpenSSH, коли PAM є основним постачальником хост аутентифікації для забезпечення ефективної аутентифікації по паролю, що іноді призводить до нездатності увійти в систему з клієнтом, який підтримує тільки прості пароль метод перевірки автентичності.

GSSAPI методи аутентифікації, які забезпечують розширюваної схеми для виконання SSH аутентифікації з використанням зовнішніх механізмів, таких як Kerberos 5 або NTLM, забезпечуючи Single Sign On Можливість SSH сесій. Ці методи, як правило, здійснюються комерційними реалізаціями SSH для використання в організаціях, хоча OpenSSH дійсно є робоча реалізації GSSAPI.

• Сполуки шарів (RFC 4254). Цей шар визначає поняття канали, канал запити і глобальний запитів за допомогою якої SSH послуги. Одне з'єднання SSH можна розмістити кілька каналів одночасно, кожна передача даних в обох напрямках. Канал запити використовуються для релейний вихід за смугу каналу конкретні дані, такі, як змінився розмір вікна термінала або код виходу з серверного процесу. SSH клієнт запитує сервер на стороні порту, який направлятиметься з використанням глобальної запитом. Стандартні типи каналів включають в себе:

- Оболонка для терміналу, SFTP і Exec запитів (у тому числі SCP

трансфертів)

- Прямий TCP/IP для клієнт-сервер передаються з'єднання
- Спрямований-TCP/IP для сервер-клієнт направляється з'єднань
- SSHFP DNS-запис (RFC 4255) надає громадськості відбитки пальців ключа хоста для того, щоб допомогти в перевірці достовірності господаря.

Це відкрита архітектура забезпечує значну гнучкість, дозволяючи SSH, який буде використовуватися для різних цілей, крім безпечної оболонки.

Функціональність транспортного рівня тільки порівнянна з Transport Layer Security (TLS); шар аутентифікації користувачів вельми розширюваної за допомогою користувальницьких методів аутентифікації і з'єднання шарів забезпечує можливість мультиплексування багатьох середніх сесій в одне з'єднання SSH, функція порівнянна з BEEP і не доступні в TLS.

8.13. SMTP – протокол пересилання електронної пошти

Simple Mail Transfer Protocol (Простий Протокол Пересилання Пошти) — це протокол, який використовується для пересилання електронної пошти до поштового сервера або з клієнта-комп'ютера, або між поштовими серверами.

В IANA для SMTP зареєстрований порт 25. SMTP з'єднання де застосовується SSL шифрування використовують порт 465. Формально SMTP визначений в RFC 821 (STD 10) та покращений RFC 1123 (STD 3) розділ 5.

Протокол, який використовується зараз, також відомий як ESMTP і визначений в RFC 2821.

SMTP — порівняно простий, текстовий протокол, в якому з'єднання відбувається завжди за ініціативи відправника. SMTP — синхронний протокол і складається із серії команд, що посилаються клієнтом та відповідей сервера [8.17 - 8.19]. Відправником зазвичай є поштовий клієнт кінцевого користувача або поштовий сервер.

SMTP було розроблено як протокол транспортування і доставки, тому системи, що використовують SMTP, завжди повинні бути у робочому стані. Протокол часто використовується для передачі повідомлень клієнтами електронної пошти, які, проте, не мають можливості діяти як сервер (рис. 8.1.).

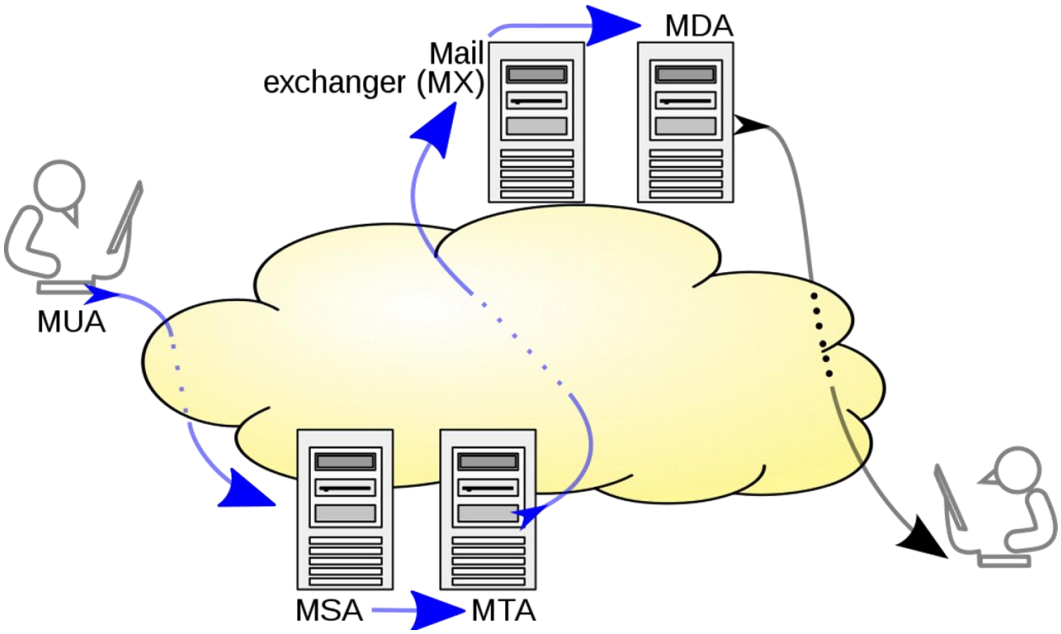


Рис. 8.1. Модель обробки пошти

Сині стрілки можуть бути реалізовані з використанням різних версій SMTP. Електронна пошта представлена поштовим клієнтом (MUA, mail user agent - призначений для користувача поштовий агент) для поштового сервера (MSA, mail submission agent - агент відправки електронної пошти) за допомогою SMTP по TCP-порту 587.

Звідти MSA доставляє пошту своїм агентам передачі повідомлень (MTA, mail transfer agent). Часто ці два агента є просто різними зразками одного і того ж програмного забезпечення, запущеного з різними параметрами на одному пристрої.

Локальна обробка може бути проведена як на окремій машині, так і розділена між різними пристроями; в першому випадку залучені процеси мають загальний доступ до файлів, у другому випадку SMTP використовується для пересилання повідомлення внутрішньо, причому кожен хост налаштований на використання наступного пристрою в якості проміжного хоста.

Кожен процес - сам по собі MTA, тобто - SMTP-сервер.

Граничний MTA повинен знайти цільової хост.

Він використовує систему доменних імен (DNS) для пошуку записів поштового обмінника (mail exchanger - MX) домену одержувача (частина адреси, що знаходиться праворуч від символу @). Повертаємий запис поштового MX містить ім'я цільового хоста. Потім MTA підключається до сервера обміну в якості SMTP-клієнта [8.17 - 8.19].

Як тільки мета MX приймає вхідне повідомлення, вона передає його агенту доставки пошти (mail delivery agent - MDA) для локальної доставки повідомлення. MDA передбачає можливість зберігати повідомлення у відповідному форматі поштової скриньки. Прийом пошти, знову ж таки, може бути проведений як декількома, так і одним комп'ютером - зображення показує два найближчих ящика для кожного випадку. MDA може доставляти повідомлення прямо на зберігання або передавати їх по мережі за допомогою SMTP або будь-яких інших засобів, в тому числі протоколу локальної пересилання пошти (Local Mail Transfer Protocol - LMTP) - похідного від SMTP, призначеного для цієї мети.

Після доставки на локальний поштовий сервер повідомлення зберігається для пакетного пошуку по аутентифікованим поштовим клієнтаім (MUA).

Повідомлення витягується додатками кінцевого користувача (поштовими клієнтами) з використанням протоколу IMAP (Internet Message Access Protocol), який полегшує доступ до повідомлень і управляє зберігається поштою, або за допомогою протоколу POP (Post Office Protocol), який зазвичай використовує традиційний mbox-формат файлів, або фірмовими системами на кшталт Microsoft Exchange / Outlook або Lotus Notes / Domino. Клієнти мережевий пошти можуть використовувати будь-який метод, але протокол пошуку часто не відповідає офіційним стандартам.

SMTP визначає передачу повідомлення, а не його зміст.

Таким чином, він задає оболонку повідомлення і її параметри (такі, як

відправник оболонки), але не заголовок або тіло самого повідомлення. STD 10 і RFC 5321 визначають SMTP (оболонку), в той час як STD 11 і RFC 5322 - повідомлення (заголовок і тіло), офіційно званий форматом поштового повідомлення (Internet Message Format).

Протокол SMTP підтримує передачу повідомлень (електронної пошти) між довільними вузлами мережі internet. Маючи механізми проміжного збереження пошти і механізми підвищення надійності доставки, протокол SMTP допускає використання різних транспортних служб. Він може працювати навіть у мережах, що не використовують протоколи сімейства TCP/IP.

Протокол SMTP забезпечує як групування повідомлень на адресу одного одержувача, так і розмноження декількох копій повідомлення для передачі в різні адреси [8.17 - 8.19]. Над модулем SMTP розташовується поштова служба конкретних обчислювальних систем.

Взаємодія в рамках SMTP будується за принципом двостороннього зв'язку, що встановлюється між відправником і одержувачем поштового повідомлення При цьому відправник ініціює з'єднання і посилає запити на обслуговування, а одержувач - відповідає на ці запити. Фактично відправник виступає в ролі клієнта, а одержувач - сервера.

Канал зв'язку встановлюється безпосередньо між відправником і одержувачем повідомлення. При такій взаємодії пошта досягає абонента протягом декількох секунд після відправлення.

Простий протокол передачі пошти забезпечує двосторонній обмін повідомленнями між локальним клієнтом і сервером. Обмін командами і відповідями на них називається поштовою транзакцією (mail transaction). Команди простого протоколу передачі пошти (SMTP) представлені в таблиці.

У відповідності до специфікації команди, позначені хрестиком (X) у табл., зобов'язані бути присутнім у будь-якій реалізації SMTP. Інші команди SMTP можуть бути реалізовані додатково. Кожна SMTP-команда повинна закінчуватися або пробілом (якщо в неї є аргумент), або комбінацією CRLF. В

описі команд вживалося слово “дані”, а не <повідомлення>. Цим підкреслювалося, що, крім тексту, SMTP дозволяє передавати і двійкову інформацію, наприклад графічні чи звукові файли. Іншими словами, SMTP здатний передавати дані будь-якого змісту, а не тільки текстові повідомлення.

Таблиця 8.7. Опис команд

Команда	Обов'язкова	Опис
HELO	X	Ідентифікує модуль-передавач для модуля-приймача (hello).
MAIL	X	Починає поштову транзакцію, що завершується передачею даних в одну чи кілька поштових скринь (mail).
RCPT	X	Ідентифікує одержувача поштового повідомлення (recipient).
DATA		Рядки, що слідують за цією командою, розглядаються одержувачем як дані поштового повідомлення. У випадку SMTP, поштове повідомлення закінчується комбінацією символів: CRLF-точка-CRLF.
RSET		Перериває поточну поштову транзакцію (reset).
NOOP		Вимагає від одержувача не починати ніяких дій, а тільки видати відповідь OK. Використовується головним чином для тестування.(No operation).
QUIT		Вимагає видати відповідь OK і закрити поточне з'єднання.
VRFY		Вимагає від приймача підтвердження, що її аргумент є дійсним ім'ям користувача. (Див. Примітка.).
SEND		Починає поштову транзакцію, що доставляє дані на один чи кілька терміналів (а не в поштову скриньку).
SOML		Починає транзакцію MAIL чи SEND, що доставляє дані на один чи кілька терміналів або у поштові скриньки.
SAML		Починає транзакцію MAIL і SEND, що доставляють дані на один чи кілька терміналів і в поштові скриньки.
EXPN		Команда вимагає від SMTP-приймача підтвердження, чи дійсно аргумент є адресою поштового розсилання і якщо так, повернути адресу одержувача повідомлення (expand).
HELP		Команда вимагає від SMTP-приймача повернути повідомлення-довідку про його команди.
TURN		Команда вимагає від SMTP-приймача або сказати OK і помінятися ролями, тобто стати STMP- передавачем, або послати повідомлення-відмовлення і залишитися в ролі SMTP-приймача.

У будь-який момент під час транзакції клієнт може використовувати команди NOOP, HELP, EXPN і VRFY. У відповідь на кожну команду сервер висилає клієнту визначену інформацію [8.17 - 8.19].

В залежності від відповіді клієнт може почати конкретні дії, однак специфікація SMTP нічого не говорить з цього приводу. Наприклад, клієнт може передати команду VRFY для того, щоб переконатися, що ім'я користувача дійсно. Якщо сервер відповість, що даного імені не існує, клієнт може не передавати пошту для цього користувача.

У специфікації SMTP, однак, немає ніяких указівок з цього приводу - клієнт може нічого не робити у відповідь на команду VRFY.

Коди відповідей SMTP. У специфікації SMTP потрібно, щоб сервер відповідав на кожну команду SMTP-клієнта. Сервер відповідає тризначною

комбінацією цифр, що називається кодом відповіді. Разом з кодом відповіді, як правило, передається один чи кілька рядків текстової інформації.

Кілька рядків тексту, як правило, супроводжують тільки команди EXPN і HELP.У специфікації SMTP, однак, відповідь на будь-яку команду може складатися з декількох рядків тексту.

Кожна цифра в коді відповіді має визначений сенс. Перша цифра означає, чи було виконання команди успішним (2), неуспішним (5), чи ще не закінчилося (3). Як зазначено в додатку Е документа RFC 821, простий SMTP-клієнт може аналізувати тільки першу цифру у відповіді сервера, і на підставі її продовжувати свої дії. Друга і третя цифри коду відповіді роз'ясняють значення першої.

Проміжні агенти. [8.18 - 8.20]. Термін "маршрут доставки" (forward-path) служить для того, щоб відрізняти поштову скринька (mailbox), ім'я якої абсолютно, від шляху (він може бути різним), по якому слідує пошта.

Припустимо, що ми хочемо доставити два поштових повідомлення за однією адресою. Обидва повідомлення мають один пункт призначення, однак не обов'язково будуть слідувати ідентичному маршруту.

Як правило, конкретний маршрут для пошти вибирається системним адміністратором.

Щоб направити пошту за потрібним шляхом, використовуються значення маршруту доставки і зворотного маршруту, у яких указуються проміжні агенти (relay agents). Проміжний агент доставки – це так званий поштовий хаб (mail hub), який настроєний на передачу транзитної пошти.

У наш час проміжні агенти присутні практично у всіх мережах, що входять у Internet.

Щоб спростити процес конфігурації поштової системи, у локальній мережі встановлюється один комп'ютер, що служить проміжним агентом (relay host). Уся пошта користувачів попадає спочатку на нього. Потім цей комп'ютер розсилає повідомлення по Internet. Крім того, такий комп'ютер може служити

захистом фірми від злодіїв-хакерів.

Обмежуючи спілкування локальної мережі з зовнішнім світом до рівня пошти, організація зводить до мінімуму ризик небажаного вторгнення у свої власні системи.

Крім того, адмініструвати і захищати в цьому випадку приходиться єдиний комп'ютер. SMTP у змозі послати повідомлення безпосередньо з комп'ютера користувача на комп'ютер адресата в тому випадку, якщо між ними існує пряме поштове з'єднання.

Як правило, між двома комп'ютерами знаходиться один чи кілька проміжних агентів. Щоб забезпечити доставку, у поштовому повідомленні потрібно вказати ім'я комп'ютера-одержувача і точне найменування поштової скриньки.

Аргументом команди MAIL є зворотний маршрут, що включає ім'я джерела повідомлення й імена всіх проміжних агентів. Аргумент команди RCPT - маршрут доставки, що містить ім'я одержувача повідомлення. Зворотний маршрут описує шлях, який пройшло повідомлення, тоді як маршрут доставки ідентифікує місце призначення.

Зворотний маршрут використовується SMTP, коли потрібно передати повідомлення про помилку або про неможливість доставки повідомлення, коли воно вже пройшло через проміжний агент. В міру просування повідомлення по Internet записи про його маршрут змінюються.

До обов'язків системних адміністраторів входить правильна побудова місцевих локальних агентів на передачу повідомлень проміжному агенту, і навпаки, проміжні агенти на доставку повідомлень місцевим агентам. Якщо в проміжного агента зміниться ім'я, усе, що потрібно зробити в конфігурації місцевого агента - змінити ім'я комп'ютера в системі DNS. Інші параметри конфігурації не змінюються.

Зворотні маршрути і маршрути доставки будуються агентами передачі пошти в міру проходження повідомлення від одного агента до наступного.

Якщо черговий на шляху сполучення SMTP-агент не вміє обслуговувати проміжну доставку, він має відповісти кодом, який передбачений на випадок відсутності місцевої поштової скриньки.

```
S: (ожидает соединения)
C: (Подключается к порту 25 сервера)
S:220 mail.company.tld ESMTP is glad to see you!
C:HELO
S:250 domain name should be qualified
C:MAIL FROM: <someusername@somecompany.ru>
S:250 someusername@somecompany.ru sender accepted
C:RCPT TO: <user1@company.tld>
S:250 user1@company.tld ok
C:RCPT TO: <user2@company.tld>
S:550 user2@company.tld unknown user account
C:DATA S:354 Enter mail, end with "." on a line by itself
C:From: Some User <someusername@somecompany.ru>
C:To: User1 <user1@company.tld> C:Subject: tema
C:Content-Type: text/plain
C:
C:Hi!
C:.
S:250 769947 message accepted for delivery
C:QUIT
S:221 mail.company.tld CommuniGate Pro SMTP closing connection S: (закрывает
соединение)
```

Рис. 8.2. Приклад найпростішої SMTP-сесії.

де, C: — клиент, S: — сервера

В результаті такої сесії лист буде доставлено адресату user1@company.tld, але не буде доставлено адресату user2@company.tld, тому що такої адреси не існує [8.18 - 8.20].

Багато клієнтів запитують розширення SMTP, підтримувані сервером, за допомогою команди HELO з специфікації розширеного SMTP (RFC 1870). HELO використовується тільки в тому випадку, якщо сервер не відповів на EHLO.

Сучасні клієнти можуть використовувати ключ SIZE розширення ESMTP для запиту максимального розміру повідомлення, яке буде прийнято.

Більш старі клієнти і сервери можуть спробувати передати надмірно великі повідомлення, які будуть відхилені після споживання мережевих ресурсів, включаючи час з'єднання.

Користувачі можуть вручну заздалегідь визначити максимальний розмір,

який приймає ESMTP-серверами. Клієнт замінює команду HELO на EHLO.

```
S: 220 smtp2.example.com ESMTP Postfix
C: EHLO bob.example.org
S: 250-smtp2.example.com Hello bob.example.org [192.0.2.201]
S: 250-SIZE 14680064
S: 250-PIPELINING
S: 250 HELP
```

Рис. 8.3. Клієнт замінює команду HELO на EHLO

smtp2.example.com оголошує, що він прийме повідомлення розміром не більше ніж 14,680,064 октетів (8-бітних байтів).

Залежно від фактичного використання сервера, він може на даний момент не прийняти повідомлення такої величини. У найпростішому випадку, ESMTP-сервер оголосить максимальний SIZE тільки при взаємодії з користувачем через HELO. Початкова специфікація SMTP не включала коштів для аутентифікації відправників. Згодом, в RFC 2554 було введено розширення. Розширення SMTP (ESMTP) надає поштових клієнтів можливості завдання механізму забезпечення безпеки для сервера, аутентифікації і профілю безпеки SASL (Simple Authentication and Security Layer) для наступних передач повідомлень.

Продукти Microsoft реалізують власний протокол - SPA (Secure Password Authentication) за допомогою розширення SMTP-AUTH [8.18 - 8.20].

Однак, непрактичність широкого поширення реалізації і управління SMTP-AUTH означає, що проблема спаму не може бути вирішена за його допомогою.

Широке зміна SMTP, так само як і повна його заміна, вважаються непрактичними через величезну інстальованої бази SMTP. Internet Mail 2000 був одним з претендентів для такої заміни.

Спам функціонує завдяки різним чинникам, в тому числі не відповідають стандартам реалізації MTA, уразливості в захисті операційних систем (зокрема через постійним широкосмуговим підключенням), що дозволяє спамерам віддалено контролювати комп'ютер кінцевого користувача і посилати з нього спам. Існує кілька пропозицій для побічних протоколів, які допомагають роботі

SMTP. Дослідницька група Anti-Spam (The Anti-Spam Research Group - ASRG) - підрозділ Дослідницької групи Інтернет-технологій працює над поштової аутентифікацією і іншими пропозиціями для надання простий аутентифікації, яка буде гнучкою, легковажною і масштабованої. [8.18 8.20 - 8.22].

Недавня діяльність Інженерної ради Інтернету (IETF) включає в себе MARID (2004), який призвів до двох затверджених IETF-експериментів в 2005, і DomainKeys Identified Mail в 2006.

STARTTLS в RFC тисяча вісімсот шістьдесят дев'ять наказує починати сесію не командою HELO, а командою EHLO. У разі, якщо сервер не підтримує розширень, то він відповість на EHLO помилкою, в цьому випадку клієнт повинен послати команду HELO і не використовувати розширення протоколу.

Якщо ж сервер підтримує ESMTP, то крім привітання він повідомить список підтримуваних розширень протоколу SMTP, наприклад:

```
ehlo office.company1.tld
250-mail.company2.tld is pleased to meet you
250-DSN
250-SIZE
250-STARTTLS
250-AUTH LOGIN PLAIN CRAM-MD5 DIGEST-MD5 GSSAPI MSN NTLM
250-ETRN
250-TURN
250-ATRN
250-NO-SOLICITING
250-HELP
250-PIPELINING 250 HELO
```

Рис. 8.4. список підтримуваних розширень протоколу SMTP

8.14. SNMP - простий протокол керування мережею

Протокол SNMP (Simple Network Management Protocol) працює на базі UDP і призначений для використання мережними керуючими станціями. Він дозволяє керуючим станціям збирати інформацію про положення справ у мережі internet. Протокол визначає формат даних, їх обробка й інтерпретація

залишаються на розсуд керуючих станцій або менеджеру мережі [8.18, 8.23].

SNMP - це протокол прикладного рівня, розроблений для стека TCP/IP. Протокол SNMP використовується для одержання від мережних пристроїв інформації про їхній статус, продуктивність та інші характеристики, що зберігаються в базі даних інформації управління MIB (Managmant Information Base). Простота SNMP багато в чому визначається простотою MIB SNMP, особливо їх перших версій MIB I і MIB II. Крім того, протокол SNMP також є не дуже складним.

Існують стандарти, що визначають структуру MIB, у тому числі набір типів її об'єктів, їх імена і припустимі операції над цими об'єктами (наприклад, "читати").

Деревоподібна структура MIB містить обов'язкові (стандартні) піддерева, а також у ній можуть бути приватні (private) піддерева, що дають можливість виробнику інтелектуальних пристроїв управляти якимись специфічними функціями пристрою на основі специфічних об'єктів MIB.

Агент у протоколі SNMP - це обробний елемент, що забезпечує менеджерам, розміщеним на станціях управління мережею, доступ до значень змінних MIB і цим надає їм можливість реалізовувати функції спостереження за пристроєм та управління ним.

Основні операції управління винесені до менеджера, а агент SNMP виконує пасивну роль, передаючи менеджеру за його запитом значення накопичених статистичних змінних. При цьому пристрій працює з мінімальними витратами на підтримку протоколу управління. Він використовує майже усю обчислювальну потужність для виконання своїх основних функцій маршрутизатора, моста (шлюза) або концентратора, а агент збирає статистичні дані та значення змінних про стан пристрою і передає їх менеджеру системи управління.

SNMP - це протокол типу "запит-відповідь", тобто на кожний запит, що надійшов від менеджера, агент повинен передати відповідь [8.18, 8.23].

Особливістю протоколу є його надзвичайна простота. Він містить всього кілька команд:

• команда Get-request використовується менеджером для одержання від агента значення якогось об'єкта згідно з його ім'ям;

• команда GetNext-request використовується менеджером для відкривання значення наступного об'єкта (без указування його імені) при послідовному перегляді таблиці об'єктів;

• за допомогою команди Get-response агент SNMP передає менеджеру відповідь на команди Get-request або GetNext–request;

• команда Set використовується менеджером для зміни значення якогось об'єкта. За допомогою команди Set здійснюється управління пристроєм. Агент повинен розуміти зміст значень об'єкта, що використовується для управління пристроєм, і на підставі цих значень здійснювати реальне управління - відключити порт, приписати порт визначеної VLAN та ін. Команда Set придатна також для встановлення умови, при виконанні якої агент SNMP повинен надіслати менеджеру відповідне повідомлення. Може бути визначена реакція на такі події, як ініціалізація агента, рестарт агента, переривання зв'язку, відновлення зв'язку, помилкова аутентифікація і втрата найближчого маршрутизатора. Якщо відбувається будь-яка з цих подій, агент ініціює переривання;

• команда Trap використовується агентом для повідомлення менеджера про виникнення особливих ситуацій;

• версія SNMP v.2 додає до цього набору команду GetBulk, яка дозволяє менеджеру за один запит одержати інформацію про кілька значень змінних.

Нині існує кілька стандартів на бази даних інформації управління для протоколу SNMP. Основними є стандарти MIB-I і MIB-II, а також версія бази даних для віддаленого управління RMON (Remote Monitoring) MIB. Крім того, існують стандарти для спеціальних пристроїв MIB конкретного типу (наприклад, MIB для концентраторів або MIB для модемів), а також окремі MIB

конкретних фірм-виробників устаткування.

Початкова специфікація MIB-I визначала тільки операції читання значень змінних. Операції переміни або встановлення значень об'єкта є частиною специфікацій MIB-II.

Версія MIB-I (RFC 1156) визначає 114 об'єктів, що підрозділяються на вісім груп:

• System – загальні дані про пристрій (наприклад, ідентифікатор постачальника, час останньої ініціалізації системи);

• Interfaces – параметри мережних інтерфейсів пристрою (наприклад, їх кількість, типи, швидкості обміну, максимальний розмір пакета та ін.);

• Address Translation Table – опис відповідності між мережними і фізичними адресами (наприклад, за протоколом ARP);

• Internet Protocol – дані за протоколом IP (адреси IP- шлюзів та хостів, статистика IP-пакетів);

• ICMP – дані стосовно протоколу обміну повідомленнями управління ICMP;

• TCP – дані стосовно протоколу TCP (наприклад, про TCP-з'єднання);

• UDP – дані стосовно протоколу UDP (кількість переданих, прийнятих і помилкових UDP-дейтаграм);

• EGP – дані стосовно протоколу обміну маршрутною інформацією Exterior Gateway Protocol, що використовується в Internet (кількість повідомлень, прийнятих з помилками та без помилок).

З цього переліку груп змінних зрозуміло, що стандарт MIB-I розроблявся з жорсткою орієнтацією на управління маршрутизаторами, які підтримують протоколи стеку TCP/IP.

У версії MIB-II (RFC 1213), прийнятій у 1992 р., розширено набір стандартних об'єктів (близько 185), а кількість груп збільшилася до 10.

Об'єкт SysUpTime містить значення тривалості часу роботи системи з моменту останнього перевантаження, об'єкт SysObjectID - ідентифікатор

пристрою (наприклад, маршрутизатор).

Об'єкт ifNumber визначає кількість мережних інтерфейсів пристрою, а об'єкт ifEntry є вершиною піддерева, що описує один із конкретних інтерфейсів пристрою. Об'єкти ifType та ifAdminStatus, що входять до цього піддерева, визначають відповідно тип та стан одного з інтерфейсів, у цьому випадку інтерфейсу Ethernet.

До об'єктів, що описують кожний конкретний інтерфейс пристрою, входять такі:

• ifType - тип протоколу, що підтримує інтерфейс. Цей об'єкт набуває значень усіх стандартних протоколів канального рівня;

• ifMTU - максимальний розмір пакету мережного рівня, що можна послати через цей інтерфейс;

• ifSpeed - пропускна спроможність інтерфейсу в бітах за секунду (100 для Fast Ethernet);

• ifPhysAdress - фізична адреса порту; для Fast Ethernet нею буде MAC-адреса;

• ifAdminStatus - бажаний статус порту: up - готовий передавати пакети; down - не готовий передавати пакети; testing - перебуває в тестовому режимі;

• ifOperStatus - фактичний поточний статус порту, має ті самі значення, що й ifAdminStatus

• ifInOctets - загальна кількість байтів, включаючи службові, прийнята портом від моменту останньої ініціалізації SNMP-агента;

• ifInNUcastPkts - кількість пакетів з індивідуальною адресою інтерфейсу, доставлених протоколу верхнього рівня;

• ifInUcastPkts - кількість пакетів із неіндивідуальною адресою інтерфейсу, доставлених протоколу верхнього рівня;

• ifInDiscards - кількість пакетів, що були прийняті інтерфейсом, виявилися коректними, але не були доставлені протоколу верхнього рівня через переповнення буфера пакетів або з іншої причини;

• ifInErrors - кількість надісланих пакетів, що не були передані протоколу верхнього рівня через виявлення в них помилок.

Крім об'єктів, що описують статистику за вхідними пакетами, існують аналогічні об'єкти, що стосуються вихідних пакетів.

З опису об'єктів MIB-II видно, що ця база даних не дає детальної статистики характерних помилок кадрів Ethernet. Крім того, вона не відображає зміни характеристик у часі, що цікавить адміністратора мережі [8.18, 8.24].

Ці обмеження були зняті новим стандартом RMON MIB, що спеціально орієнтований на збір детальної статистики стосовно протоколу Ethernet, до того ж із підтримкою такої важливої функції, як побудова агентом залежностей статистичних характеристик від часу.

Для найменування змінних бази MIB і однозначного визначення їх форматів використовується додаткова специфікація SMI (Structure of Management Information). Наприклад, специфікація SMI включає як стандартне ім'я IPAdress і визначає його формат як рядок із 4 байт. Інший приклад - ім'я Counter, для якого визначено формат у вигляді цілого числа в діапазоні від 0 до 232-1.

При описанні змінних MIB і форматів протоколу SNMP специфікація SMI спирається на формальну мову ASN.1, прийняту ISO як систему позначень (нотації) для опису термінів комунікаційних протоколів (багато комунікаційних протоколів, наприклад, IP, PPP або Ehternet, обходяться без цієї нотації).

Нотація ASN.1 призначена для встановлення однозначної відповідності між термінами, що використовуються людиною-оператором, і тими даними, що передаються в комунікаційних протоколах апаратурою. Однозначність, що досягається, є дуже важливою в гетерогенному середовищі, характерному для корпоративних мереж. Так, замість того щоб зазначити, що деяка змінна протоколу являє собою ціле число, розробник протоколу, що використовує нотацію ASN.1, повинен точно визначити формат і припустимий діапазон змінної. В результаті документація на MIB, написана за допомогою нотації

ASN.1, може точно і механічно транслюватися у форму кодів, характерних для повідомлень протоколів.

Нотація ASN.1 підтримує базовий набір різноманітних типів даних, таких як ціле число, рядок тощо, а також дає можливість конструювати з цих базових типів складові дані - масиви, перерахування, структури.

Існують правила трансляції структур даних, описаних на ASN.1, у структури даних мов програмування, наприклад C++. Відповідно є транслятори, що виконують цю роботу. Приклади описань даних за допомогою ASN.1 наведені при описанні протокольних блоків даних SNMP [8.18, 8.24].

Нотація ASN.1 широко використовується при описанні багатьох стандартів OSI, зокрема, моделей об'єктів управління і структури повідомлень протоколу CMIP.

Імена змінних MIB можуть бути записані як у символьному, так і в числовому форматах. Символьний формат використовується для подання змінних у текстових документах і на екрані дисплея, а числові імена - у повідомленнях протоколу SNMP. Наприклад, символьному імені SysDescr відповідає числове ім'я I, а точніше 1.3.6.1.2.1.1.1.

Складове числове ім'я об'єкта SNMP MIB відповідає повному імені цього об'єкта в дереві реєстрації об'єктів стандартизації ISO. Розробники протоколу SNMP не використовували традиційний для стандартів Internet спосіб фіксації чисельних параметрів протоколу відповідно до документа RFC, який називається “Assigned Numbers” (там описуються, наприклад, чисельні значення, що може набувати поле Protocol пакета IP). Замість цього вони зареєстрували об'єкти баз MIB SNMP у всесвітньому дереві реєстрації стандартів ISO, зображеному на рис. 8.5.

Як і в будь-яких складних системах, простір імен об'єктів ISO має деревоподібну ієрархічну структуру, причому на рисунку показано тільки його верхню частину. Від кореня цього дерева відходять три гілки, що відповідають стандартам, контрольованим ISO, ITU і спільно ISO-ITU.

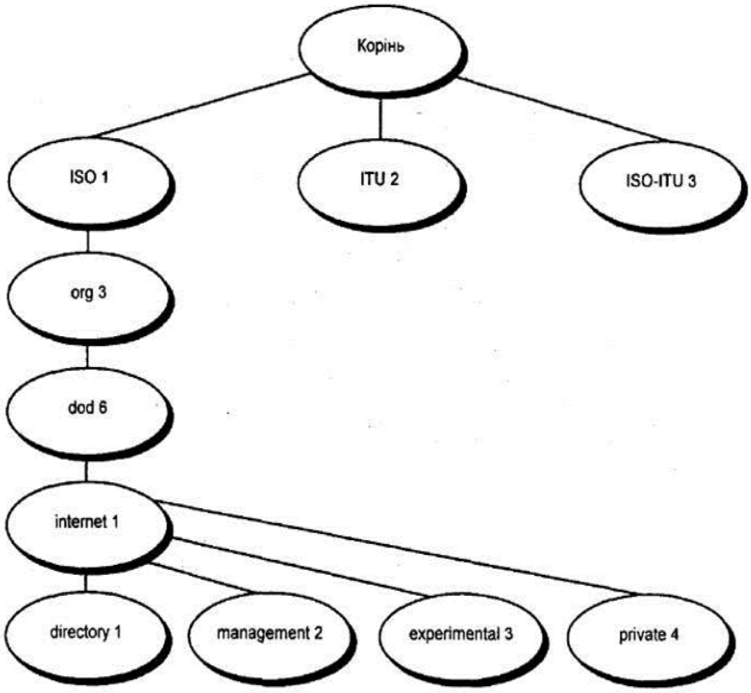


Рис. 8.5. Простір імен об'єктів ISO

У свою чергу, ISO створила гілку для стандартів, утворюваних національними і міжнародними організаціями (гілка org). Стандарти Internet створювалися під егідою Міністерства оборони США (Departament of Defence, DoD), тому стандарти MIB потрапили в піддерево dod-internet, а далі, природно, у групу стандартів управління мережею - гілка mgmt [8.18, 8.24].

Об'єкти будь-яких стандартів, утворюваних під егідою ISO, однозначно ідентифікуються складовими символьними іменами, що починаються від кореня цього дерева.

У повідомленнях протоколів символьні імена не використовуються, а застосовуються однозначно відповідні їм складові числові імена.

Кожна гілка дерева імен об'єктів нумерується в дереві цілими числами зліва направо, починаючи з одиниці, і цими числами заміняються символьні імена. Тому повне символьне ім'я об'єкта MIB має вигляд: iso.org.dod.internet.mgmt.mib, а повне числове ім'я -1.3.6.1.2.1.

Група об'єктів private (4) зарезервована за стандартами, утворюваними приватними компаніями Cisco, Hewlett-Packard та ін. Те саме дерево реєстрації використовується для найменування класів об'єктів CMIP і TMN.

Відповідно кожна група об'єктів MIB-I і MIB-II також має, крім стислих символьних імен, наведених вище, повні символьні імена і відповідні їм числові імена.

Наприклад, стисле символьне ім'я групи System має повну форму iso.org.dod.internet.mgmt.mib.system, а відповідне їй числове ім'я -1.3.6.1.2.1.

Протокол SNMP обслуговує передавання даних між агентами і станцією управління мережею. SNMP використовує дейтаграмний транспортний протокол UDP, що не забезпечує надійної доставки повідомлень. Протокол, що організує надійне передавання дейтаграм на основі з'єднань TCP, дуже завантажує пристрої, якими управляють, що на момент розробки протоколу SNMP були не дуже потужними, тому від послуг протоколу TCP вирішили відмовитися.

Повідомлення SNMP, на відміну від повідомлень багатьох інших комунікаційних протоколів, не мають заголовків із фіксованими полями. Відповідно до нотації ASN.1 повідомлення SNMP складається з довільної кількості полів, і перед кожним полем знаходиться опис його типу та розміру.

Будь-яке повідомлення SNMP складається з трьох основних частин: версії протоколу (version), ідентифікатора спільності (community), що використовується для групування пристроїв, якими управляє визначений менеджер, та зони даних, в якій утримуються описані вище команди протоколу, імена об'єктів та їхні значення. Зона даних поділяється на блоки даних протоколу PDU (Protocol Data Unit) [8.18, 8.24].

Наприклад, вигляд формату повідомлення SNMP у нотації ASN.1 має вигляд:

- SNMP-Message ::=
- SEQUENCE {
- version INNTEGER {
- version-1 (0).
- }

- community
- OCTET STRING.
- SNMP-PDUs
- ANY
- }

Зона даних може містити п'ять різних типів PDU, що відповідають п'ятьом командам протоколу SNMP:

- SNMP-PDUs ::=
- CHOICE {
- get-request
- GetRequest-PDU.
- get-next-request
- GetNextRequest-PDU.
- get-response
- GetResponse-PDU.
- set-request
- SetRequest-PDU.
- trap
- Trap-PDU.
- }

Для кожного типу PDU є визначення його формату. Наприклад, у спосіб, зображений нижче, може бути описаний формат блоку Get-Request-PDU:

- GetRequest-PDU ::=
- IMPLICT SEQUENCE {
- request-id
- RequestID.
- error-status
- ErrorStatus.

- error-index
- ErrorIndex.
- variable-bindings
- VarBindList
- }

Далі стандарт SNMP визначає відповідно формат змінних блоку Get-Request-PDU. Змінна RequestID - це 4-байтове ціле число (використовується для встановлення відповідності відповідей запитам), ErrorStatus та ErrorIndex - це однобайтові цілі числа, що у запиті мають бути встановлені в 0. VarBindList - це список числових імен об'єктів, значеннями яких цікавиться менеджер. У нотації ASN.1 цей список складається з пар "ім'я – значення". При запиті значення змінної має бути встановлене в 0.

Специфікація RMON MIB.

Новітнім застосуванням функціональних можливостей SNMP є специфікація RMON, що забезпечує віддалену взаємодію з базою MIB.

До появи RMON протокол SNMP не міг використовуватися віддалено, він припускав тільки локальне управління пристроями.

База RMON MIB має поліпшений набір властивостей для віддаленого управління, тому що містить агреговану інформацію про пристрій, що не потребує передавання мережею великих обсягів інформації.

Об'єкти RMON MIB містять додаткові лічильники помилок у пакетах, більш гнучкі засоби аналізу трендів і статистики, потужніші засоби фільтрації для захоплення та аналізу окремих пакетів, а також більш складні умови встановлення сигналів попередження.

Агенти RMON MIB інтелектуальніші порівняно з агентами MIB-I або MIB-II і виконують значну частину роботи з опрацювання інформації про пристрій, що раніше виконували менеджери [8.18, 8.24]. Ці агенти можуть розташовуватися всередині різноманітних комунікаційних пристроїв, а також можуть бути виконаними у вигляді окремих програмних модулів, що працюють

на універсальних персональних комп'ютерах і ноутбуках.

Об'єкту RMON привласнений номер 16 у наборі об'єктів MIB, а сам об'єкт RMON об'єднує 10 груп таких об'єктів:

• Statistics - поточні накопичені статистичні дані про характеристики пакетів, кількість колізій тощо;

• History - статистичні дані, збережені через визначені проміжки часу для наступного аналізу тенденцій їх змін;

• Alarms - граничні значення статистичних показників, при перевищенні яких агент RMON посилає повідомлення менеджеру;

• Hosts - дані про хости мережі, у тому числі і про їх MAC-адреси;

• HostTopN - таблиця найбільш завантажених хостів мережі;

• TrafficMatrix - статистика про інтенсивність трафіка між кожною парою хостів мережі, упорядкована у вигляді матриці;

• Filter - умови фільтрації пакетів;

• PacketCapture - умови захоплення пакетів;

• Event - умови реєстрації і генерації подій.

Ці групи пронумеровані в зазначеному порядку, тому, наприклад, група Hosts має числове ім'я 1.3.6.1.2.1.16.4.

Десяту групу складають спеціальні об'єкти протоколу Token Ring.

Усього стандарт RMON MIB визначає близько 200 об'єктів у 10 групах, зафіксованих у двох документах - RFC 1271 для мереж Ethernet і RFC 1513 для мереж Token Ring.

Характерною рисою стандарту RMON MIB є його незалежність від протоколу мережного рівня (на відміну від стандартів MIB-I і MIB-II, орієнтованих на протоколи TCP/IP). Тому він зручний для гетерогенних середовищ, що використовують різноманітні протоколи мережного рівня.

Розглянемо докладніше групу Statistics, що визначає, яку інформацію про кадри (названих у стандарті пакетами) Ethernet може надати агент RMON. Група History заснована на об'єктах групи Statistics, тому що її об'єкти

дозволяють не дуже складно будувати часові ряди для об'єктів групи Statistics.

У групу Statistics входять разом з деякими іншими такі об'єкти:

• etherStatsDropEvents - загальна кількість подій, коли пакети були проігноровані агентом через недостатність його ресурсу. Самі пакети при цьому не обов'язково були втрачені інтерфейсом;

• etherStatsOctets - загальна кількість байтів (включаючи помилкові пакети), прийнятих із мережі (крім преамбули, але включаючи байти контрольної суми);

• etherStatsPkts - загальна кількість отриманих пакетів (включаючи помилкові);

• etherStatsBroadcastPkts - загальна кількість нормальних пакетів, що були послані за широкомовною адресою;

• etherStatsMulticastPkts - загальна кількість нормальних пакетів, отриманих за мультимовною адресою;

• etherStatsCRCAlignErrors - загальна кількість отриманих пакетів, що мали довжину (крім преамбули) між 64 і 1518 байт, не містили цілого числа байтів {alignment error) або мали неправильну контрольну суму (FCS eroor).

• etherStatsUndersizePkts - загальна кількість пакетів, що мали довжину меншу ніж 64 байт, але були правильно сформованими;

• etherStatsOversizePkts - загальна кількість отриманих пакетів, що мали довжину більшу ніж 1518 байт, але були правильно сформованими;

• etherStatsFragments - загальна кількість отриманих пакетів, що не складалися з цілого числа байтів або мали неправильну контрольну суму і мали до того ж довжину, меншу за 64 байт;

• etherStatsJabbers - загальна кількість отриманих пакетів, що не складалися з цілого числа байтів або мали неправильну контрольну суму і довжину більшу ніж 1518 байт;

• etherStatsCollisions - найкраща оцінка кількості колізій на певному сегменті Ethernet;

• etherStatsPkts64Octets - загальна кількість отриманих пакетів (включаючи пошкоджені) ємністю 64 байт;

• etherStatsPkts65to127Octets - загальна кількість отриманих пакетів (включаючи пошкоджені) ємністю від 65 до 127 байт;

• etherStatsPkts128to255Octets - загальна кількість отриманих пакетів (включаючи пошкоджені) ємністю від 128 до 255 байт;

• etherStatsPkts256to511Octets - загальна кількість отриманих пакетів (включаючи пошкоджені) ємністю від 256 до 511 байт;

• etherStatsPkts512to1023Octets - загальна кількість отриманих пакетів (включаючи пошкоджені) ємністю від 512 до 1023 байт;

• etherStatsPkts1024to1518Octets - загальна кількість отриманих пакетів (включаючи пошкоджені) ємністю від 1024 до 1518 байт.

З опису об'єктів зрозуміло, що за допомогою агента RMON, умонтованого в будь-який комунікаційний пристрій мережі, можна провести дуже детальний аналіз роботи сегмента Ethernet або Fast Ethernet. Спочатку можна одержати дані про типи помилок, що зустрічаються в сегменті, у кадрах, а потім доцільно за допомогою групи History зібрати дані щодо залежності інтенсивності цих помилок від часу (у тому числі й з прив'язкою до часу). Після аналізу тимчасових залежностей можна зробити деякі попередні висновки про джерело помилкових кадрів і на цій підставі сформулювати більш тонкі умови захоплення кадрів із специфічними ознаками (задаючи умови в групі Filter), що відповідають висунутій версії. Після цього можна провести ще детальніший аналіз за рахунок вивчення захоплених кадрів, видобуваючи їх з об'єктів групи Packet Capture. Пізніше було прийнято стандарт RMON 2, що поширює ідеї інтелектуальної RMON MIB на протоколи верхніх рівнів, виконуючи частину роботи аналізаторів протоколів.

Протокол SNMP є основою багатьох систем управління, хоча має кілька принципових недоліків:

• відсутність засобів взаємної аутентифікації агентів і менеджерів.

Єдиний засіб, який міг би належати до засобів аутентифікації, є використання в повідомленнях так званого “рядка співтовариства” (“community string”). Цей рядок передається мережею у відкритій формі в повідомленні SNMP і є основою для розподілу агентів і менеджерів на “співтовариства”, а агент взаємодіє тільки з тими менеджерами, що вказують у полі community string той самий символьний рядок, що і рядок, який зберігається в пам'яті агента. Це безумовно, не засіб аутентифікації, засіб структурування агентів і менеджерів. Версія SNMP v.2 мала б ліквідувати цей недолік, але в результаті розбіжностей між розробниками стандарту нові засоби аутентифікації хоча і з'явилися в цій версії, але як необов'язкові;

• робота через ненадійний протокол UDP (саме так працює більшість реалізацій агентів SNMP) призводить до втрат аварійних повідомлень (повідомлень trap) від агентів менеджерам, що може призвести до неякісного управління. Виправлення ситуації переходом на надійний транспортний протокол із встановленням з'єднань може призвести до втрати зв'язків з великою кількістю агентів SNMP, умонтованих в обладнання. (Протокол CMIP працює над надійним транспортом стека OSI і вказаного недоліку не має).

Розробники платформ управління намагаються усунути названі недоліки. Наприклад, у платформі HP 0V Telecom DM TMN, яка є платформою для розробки багаторівневих систем управління відповідно до стандартів TMN ISO, працює нова реалізація SNMP, що організує надійний обмін повідомленнями між агентами і менеджерами за рахунок самостійної організації повторного передавання втрачених повідомлень SNMP.

8.15. IMAP - протокол доступу до інтернет-повідомлень

IMAP (Internet Message Access Protocol — «протокол доступу до інтернет-повідомлень») — мережевий протокол прикладного рівня для доступу до електронної пошти [8.18].

Аналогічно до POP3, служить для роботи з вхідними листами, однак забезпечує додаткові функції, зокрема, можливість пошуку за ключовим словом без збереження пошти в локальній пам'яті.

IMAP надає користувачеві великі можливості для роботи з поштовими скриньками, розташованими на центральному сервері. Поштовий клієнт, що використовує цей протокол, отримує доступ до сховища кореспонденції на сервер так, начебто ця кореспонденція розташована на комп'ютері одержувача. Електронними листами можна маніпулювати з комп'ютера користувача (клієнта) без постійного пересилання з сервера і назад файлів з повним змістом листів. Для відправки листів використовується протокол SMTP [8.18]. IMAP був розроблений для заміни простішого протоколу POP3 і має такі переваги в порівнянні з останнім:

Листи зберігаються на сервері, а не на машині клієнта. Можливий доступ до одної і тої ж поштової скриньки з різних клієнтів. Підтримується також одночасний доступ декількох клієнтів. У протоколі є механізми, за допомогою яких клієнт може бути проінформований про зміни, зроблені іншими клієнтами.

Підтримка декількох поштових скриньок (або тек). Клієнт може створювати, вилучати і перейменовувати поштові скриньки на сервері, а також переміщати листи з одної поштової скриньки в інші.

Можливе створення спільних папок, до яких можуть мати доступ декілька користувачів.

Інформація про стан листів зберігається на сервері і доступна всім клієнтам. Листи можуть бути позначені як прочитані, важливі тощо

Підтримка пошуку на сервері. Немає необхідності завантажувати з

сервера безліч повідомлень для того, щоб знайти одне потрібне.

Підтримка онлайн-роботи. Клієнт може підтримувати з сервером постійне з'єднання, при цьому сервер у реальному часі інформує клієнта про зміни в поштових скриньках, у тому числі про нові листи.

Передбачено механізм розширення можливостей протоколу.

Поточна версія протоколу має позначення IMAP4rev1 (IMAP, версія 4, ревізія 1). Протокол підтримує передачу пароля користувача в зашифрованому вигляді. Крім того, IMAP-трафік можна зашифрувати за допомогою SSL [8.20].

Версії протоколу IMAP:

Original IMAP (1986, специфікація відсутня)

IMAP2 (1988 - RFC 1064, 1990 - RFC 1176)

IMAP3 (1991, RFC 1203)

IMAP2bis (специфікація існує тільки в чорновому варіанті 1993 року)

IMAP4 (перейменований IMAP2bis)

IMAP працює тільки з повідомленнями і не вимагає яких-небудь пакетів зі спеціальними заголовками. Кожне повідомлення має кілька пов'язаних із ним атрибутів. Ці атрибути можуть бути визначені індивідуально або спільно з іншими атрибутами.

Кожному повідомленню ставиться у відповідність 32-бітовий код, який при використанні спільно з унікальним ідентифікатором утворює 64-бітову послідовність, яка гарантуватиме однозначну ідентифікацію повідомлення в поштовій скриньці. Чим пізніше повідомлення прийшло, тим більше його UID.

UID асоціюється з поштовою скринькою і надсилається у вигляді коду uidvalidity відгуку (ok) на фазі вибору поштової скриньки. Якщо UID з попередньої сесії з якоїсь причини не може бути використаний, UID повинен бути інкрементований.

UID повідомлення не повинно змінюватися в межах сесії, його не слід змінювати і від сесії до сесії. Однак якщо неможливо зберегти UID повідомлення в подальшій сесії, кожна наступна сесія повинна мати новий

унікальний код ідентифікатора, який повинен бути більше, ніж будь який UID, використаний раніше.

Цей атрибут являє собою список з нуля або більше іменованих лексем, співвіднесених з даним повідомленням. Прапор встановлюється шляхом його додавання до цього списку і обнуляється шляхом його видалення. В IMAP 4.1 існує два типи прапорів. Прапор може бути постійним або чинним лише на час даної сесії [8.18, 8.20].

Системним прапором є прапор, ім'я якого визначено в специфікації протоколу. Всі системні прапори починаються з символу \ .

В даний час визначені наступні системні прапори:

\Seen — повідомлення прочитане

\Answered — на повідомлення відправлений відповідь

\Flagged — повідомлення відзначене як «важливе»

\Deleted — повідомлення відзначене як вилучене

\Draft — повідомлення відзначене як чернетку

\Recent — нещодавнє повідомлення (вперше з'явилося в ящику в ході поточної сесії).

Дата і час повідомлення залежать від специфіки його доставки:

Протокол SMTP — час доставки кінцевому адресату.

Команда копіювання — внутрішній час відправника.

Команда append — час, заданий параметрами команди.

Інші атрибути:

Розмір повідомлення — число октетів у повідомленні.

Структура конверта повідомлення.

Структура тіла повідомлення.

З'єднання IMAP 4.1 на увазі встановлення зв'язку між клієнтом і сервером. Клієнт посилає серверу команди, сервер клієнтові — дані та повідомлення про статус виконання запиту. Всі повідомлення, як клієнта, так і сервера мають форму рядків, що завершуються спеціальною послідовністю.

Будь-яка процедура починається з команди клієнта. Будь-яка команда клієнта починається з префікса-ідентифікатора (зазвичай короткий літерно-цифровий рядок, наприклад, A0001, A0002 тощо), званого міткою (tag). Для кожної команди клієнт генерує свою мітку. [8.18, 8.21].

Можливі два випадки, коли рядок, відправлений клієнтом, не є закінченою командою. У першому — аргумент команди забезпечується кодом, що визначає число октетів в рядку. У другому — аргументи команди вимагають відгуку з боку сервера. В обох випадках сервер посилає запит продовження команди, що починається з символу +.

Клієнт повинен завершити відправку однієї команди, перш ніж відправити іншу.

Протокольний приймач сервера читає рядок команди, що прийшла від клієнта, здійснює її розбір, виділяє параметри і передає серверу дані. По завершенні команди сервер посилає відгук.

Дані, що передаються сервером клієнтові, а також статусні відгуки, які не вказують на завершення виконання команди, мають префікс * і називаються Непоміченими відгуками.

Дані можуть бути відправлені сервером у відповідь на команду клієнта або за власною ініціативою. Формат даних не залежить від причини відправки.

Відгук вказує на вдале / невдале виконання операції. Він використовує ту ж мітку, що і команда клієнта, запустивши процедуру. Таким чином, якщо здійснюється більш ніж одна команда, мітка сервера вказує на команду, яка викликала даний відгук. Є три види відгуку завершення сервера: ok (успішне виконання), no (невдача), bad (протокольна помилка, наприклад, не розпізнана команда або зафіксована синтаксична помилка) [8.18, 8.23].

Протокольний приймач клієнта IMAP 4.1 читає рядок відгуку від сервера і вживає дії згідно з першим символом * або + .

Клієнт повинен бути готовий прийняти будь-який відгук сервера в будь-який час. Дані сервера повинні бути записані так, щоб клієнт міг їх

безпосередньо використовувати, не посилаючи серверу уточнюючих запитів.

Сервер IMAP 4.1 знаходиться в одному з чотирьох станів. Більшість команд можна використовувати лише в певних станах (рис. 8.6).

У стані без аутентифікації клієнт повинен надати ім'я і пароль, перш ніж йому стане доступна більшість команд. Перехід в цей стан виробляється при встановленні з'єднання без попередньої аутентифікації.

У стані аутентифікації клієнт ідентифікований і повинен вибрати поштову скриньку, після чого йому стануть доступні команди для роботи з повідомленнями. Перехід в цей стан відбувається при встановленні з'єднання з попередньою аутентифікацією, коли видані всі необхідні ідентифікаційні дані або при помилковому виборі поштової скриньки.

У стан вибору система потрапляє, коли успішно здійснений вибір поштової скриньки.

У стан виходу система потрапляє при перериванні з'єднання в результаті запиту клієнта або внаслідок незалежного рішення сервера.

Рис.8.6 Стани роботи сервера IMAP 4.1

1. З'єднання без попередньої аутентифікації
2. З'єднання з попередньою аутентифікацією
3. З'єднання відкинуто
4. Успішне завершення команди LOGIN або AUTHENTICATE
5. Успішне завершення команди SELECT або EXAMINE

6. Виконання команди CLOSE або невдала команда SELECT або EXAMINE
7. Виконання команди LOGOUT, закриття сервера, або переривання з'єднання.

Нижче розглянуті команди протоколу IMAP.

LOGIN

Дозволяє клієнту при реєстрації на сервері IMAP використовувати ідентифікатор користувача та пароль у звичайному текстовому вигляді. Це не найкращий метод, але іноді це єдина можливість підключитися до сервера.

AUTHENTICATE

Дозволяє клієнту використовувати при реєстрації на сервері IMAP альтернативні методи перевірки автентичності. Індивідуальна перевірка справжності користувачів не є обов'язковою і підтримується не всіма серверами IMAP. До того ж реалізації такої перевірки можуть розрізнятися залежно від сервера. Коли клієнт видає команду AUTHENTICATE, сервер відповідає на неї рядком виклику в кодуванні base64. Далі клієнт повинен відправити відповідь на виклик сервера про перевірку справжності, також закодований base64. Якщо на сервері не підтримується метод перевірки автентичності, запропонований клієнтом, він включає в свою відповідь слово NO. Після цього клієнт повинен продовжити переговори з узгодження методу перевірки автентичності. Якщо всі спроби визначити метод перевірки автентичності зазнали невдачі, то клієнт робить спробу зареєструватися на сервері допомогою команди LOGIN.

CLOSE

Закриває поштову скриньку. Коли поштова скринька закритий, то всі повідомлення, помічені прапором \DELETED, фізично видаляються з нього. Не має параметрів.

LOGOUT

Завершує сеанс для поточного ідентифікатора користувача і закриває всі відкриті поштові скриньки. Якщо які-небудь повідомлення були помічені прапором \deleted, то за допомогою цієї команди вони будуть фізично видалені з

поштової скриньки.

CREATE

Створює новий поштову скриньку. Ім'я та місце розташування нових поштових скриньок визначаються відповідно до загальних специфікаціями сервера.

DELETE

Застосовується до поштових скриньок. Сервер IMAP при отриманні цієї команди спробує видалити поштову скриньку з ім'ям, зазначеним як аргумент команди. Повідомлення видаляються разом з ящиками і відновленню не підлягають.

RENAME

Змінює ім'я поштової скриньки. Ця команда має два параметри - ім'я поштової скриньки, який потрібно перейменувати, і нове ім'я поштової скриньки.

SUBSCRIBE

Додає поштову скриньку в список активних ящиків клієнта. В цей команді використовується тільки один параметр - ім'я поштової скриньки, який потрібно внести в список. Поштова скринька не обов'язково повинен існувати, щоб його можна було додати до списку активних ящиків - це дозволяє додавати в список активних ящиків ящики, які ще не створені, або видаляти їх, якщо вони порожні.

UNSUBSCRIBE

Видаляє поштові скриньки зі списку активних. У ній так само використовується один параметр - ім'я поштової скриньки, який видаляється зі списку активних ящиків клієнта. При цьому сам по собі поштову скриньку не видаляється.

STATUS

Формує запит про поточний стан поштової скриньки. Першим параметром для цієї команди є ім'я поштової скриньки, до якого вона

застосовується. Другий параметр - це список критеріїв, за якими клієнт хоче отримати інформацію. Команда STATUS може використовуватися для отримання інформації про стан поштової скриньки без його відкриття за допомогою команд SELECT або EXAMINE.

Користувач може одержати інформацію за критеріями:

* MESSAGES - загальне число повідомлень в поштовій скриньці

* RECENT - число повідомлень з прапором \recent

* UIDNEXT - ідентифікатор UID, який буде призначений новим повідомленням

* UIDVALIDITY - унікальний ідентифікатор поштової скриньки

* UNSEEN - число повідомлень без прапора \seen

LIST

Отримати список всіх поштових скриньок клієнта; має два параметри.

LSUB

На відміну від команди LIST використовується для отримання списку ящиків, активізованих командою SUBSCRIBE . Параметри - такі ж, як у LIST.

APPEND

Додає повідомлення в кінець вказаного поштової скриньки. Як аргументи указуються ім'я ящика, прапори повідомлення (не обов'язково), мітка часу (не обов'язково) і саме повідомлення - заголовок і тіло.

Є наступні прапори повідомлень:

* \Seen - прочитано

* \Answered - написана відповідь

* \Flagged - термінове

* \Deleted - позначено для видалення

* \Draft - чернетка

* \Recent - нове повідомлення, воно надійшло у поштову скриньку після закінчення минулого сеансу

Якщо в команді вказані прапори, то вони встановлюються для доданого

повідомлення. У будь-якому випадку для повідомлення встановлюється прапор \Recent.

Якщо в команді задана мітка часу, то цей час буде встановлено як час створення повідомлення, в іншому випадку за час створення береться поточний час.

Оскільки повідомлення складається не з одного рядка, використовуються літерали.

8.16. SIP - протокол ініціації сеансів

Протокол ініціації сеансів - Session Initiation Protocol (SIP) є протоколом прикладного рівня і призначається для організації, модифікації і завершення сеансів зв'язку: мультимедійних конференцій, телефонних з'єднань і розподілу мультимедійної інформації [8.18].

Користувачі можуть брати участь в існуючих сеансах зв'язку, запрошувати інших користувачів і бути запрошеними ними до нового сеансу зв'язку. Запрошення можуть бути адресовані певному користувачеві, групі користувачів або всім користувачам.

Протокол SIP розроблений групою MMUSIC (Multiparty Multimedia Session Control) комітету IETF (Internet Engineering Task Force), а специфікації протоколу преУ основу протоколу робоча група MMUSIC заклала наступні принципи:

Персональна мобільність користувачів. Користувачі можуть переміщатися без обмежень в межах мережі, тому послуги зв'язку повинні надаватися їм в будь-якому місці цієї мережі. Користувачеві привласнюється унікальний ідентифікатор, а мережа надає йому послуги зв'язку незалежно від того, де він знаходиться. Для цього користувач за допомогою спеціального повідомлення -REGISTER - інформує про свої переміщення сервер визначення місця розташування.

Масштабованість мережі. Вона характеризується, в першу чергу, можливістю збільшення кількості елементів мережі при її розширенні. Серверна структура мережі, побудованої на базі протоколу SIP, повною мірою відповідає цій вимозі.дставлені в документі RFC 2543.

Розширюваність протоколу. Вона характеризується можливістю доповнення протоколу новими функціями при введенні нових послуг і його адаптації до роботи з різними застосуваннями.

Як приклад можна привести ситуацію, коли протокол SIP використовується для встановлення з'єднання між шлюзами, що взаємодіють з ТМЗК за допомогою сигналізації ОКС7 або DSS1. В даний час SIP не підтримує прозору передачу сигнальної інформації телефонних систем сигналізації [8.18, 8.24]. Внаслідок цього додаткові послуги ISDN виявляються недоступними для користувачів IP-сетей.

Розширення функцій протоколу SIP може бути вироблене за рахунок введення нових заголовків повідомлень, які мають бути зареєстровані у вже згадуваній раніше організації IANA. При цьому, якщо SIP-сервер приймає повідомлення з невідомими йому полями, то він просто ігнорує їх і обробляє лише ті поля, які він знає.

Для розширення можливостей протоколу SIP можуть бути також додані і нові типи повідомлень.

Інтеграція в стек існуючих протоколів Інтернет, розроблених IETF. Протокол SIP є частиною глобальної архітектури мультимедіа, розробленої комітетом Internet Engineering Task Force (IETF). Ця архітектура включає також протокол резервування ресурсів (Resource Reservation Protocol - RSVP, RFC 2205), транспортний протокол реального часу (Real-Time Transport Protocol - RTP, RFC 1889), протокол передачі потокової інформації в реальному часі, протокол опису параметрів зв'язку (Session Description Protocol -SDP, RFC 2327).

Проте функції протоколу SIP не залежать ні від одного з цих протоколів.

Взаємодія з іншими протоколами сигналізації. Протокол SIP може бути використаний спільно з протоколом H.323. Можливо також взаємодія протоколу SIP з системами сигналізації ТФОП - DSS1 і ОКС7. Для спрощення такої взаємодії сигнальні повідомлення протоколу SIP можуть переносити не лише специфічну SIP-адрес, але і телефонний номер формату E.164 або будь-якого іншого формату.

Крім того, протокол SIP, нарівні з протоколами H.323 і ISUP/IP, може застосовуватися для синхронізації роботи пристроїв управління шлюзами; в цьому випадку він повинен взаємодіяти з протоколом MGCP.

Іншою важливою особливістю протоколу SIP є те, що він пристосований до організації доступу користувачів мереж IP-телефонии до послуг інтелектуальних мереж, і існує думка, що саме цей протокол стане основним при організації зв'язку між вказаними мережами.

Однією з найважливіших особливостей протоколу SIP є його незалежність від транспортних технологій. Як транспорт можуть використовуватися протоколи X.25, Frame Relay, AAL5/ATM, IPX і ін. Структура повідомлень SIP не залежить від вибраної транспортної технології. Але, в той же час, перевага віддається технології маршрутизації пакетів IP і протоколу UDP. При цьому, правда, необхідно створити додаткові механізми для надійної доставки сигнальній інформації. До таких механізмів відносяться повторна передача інформації при її втраті, підтвердження прийому і ін.

Сигнальні повідомлення можуть переноситися не лише протоколом транспортного рівня UDP, але і протоколом TCP. Протокол UDP дозволяє швидше, ніж TCP, доставляти сигнальну інформацію (навіть з врахуванням повторної передачі непідтверджених повідомлень), а також вести паралельний пошук місця розташування користувачів і передавати запрошення до участі в сеансі зв'язку в режимі багатоадресної розсилки. У свою чергу, протокол TCP спрощує роботу з міжмережевими екранами (firewall), а також гарантує надійну доставку даних. При використанні протоколу TCP різні повідомлення, що

відносяться до одного виклику, або можуть передаватися по одному TCP-з’єднанню, або для кожного запиту і відповіді на нього може відкриватися окреме TCP-з’єднання.

Протокол ініціювання сеансового звязку (SIP)	Прикладний рівень
Протоколи TCP і UDP	Транспортний рівень
Протоколи IP∨4 і IP∨6	Мережевий рівень
PPP, ATM, Ethernet	Рівень ланки даних
UTP5, SDN, DDN, ∨.34 та ін.	Фізичний рівень

Рис. 8.7. Місце протоколу SIP в стеку протоколів TCP/IP

По мережі з маршрутизацією пакетів IP може передаватися призначена для користувача інформація практично будь-якого вигляду: мова, відео і дані, а також будь-яка їх комбінація, звана мультимедійною інформацією. При організації зв'язку між терміналами користувачів необхідно сповістити зустрічну сторону, якого роду інформація може прийматися (передаватися), алгоритм її кодування і адреса, на яку слід передавати інформацію.

Таким чином, однією з обов'язкових умов організації зв'язку за допомогою протоколу SIP є обмін між сторонами даними про їх функціональні можливості. Для цієї мети найчастіше використовується протокол опису сеансів зв'язку - SDP (Session Description Protocol) [8.18, 8.24]. Оскільки протягом сеансу зв'язку може вироблятися його модифікація, передбачена передача повідомлень SIP з новими описами сеансу засобами SDP.

Для передачі мовної інформації комітет IETF пропонує використовувати протокол RTP, але сам протокол SIP не унеможливлює вживання для цих цілей інших протоколів.

У протоколі SIP не реалізовані механізми управління потоками інформації і надання гарантованої якості обслуговування. Крім того, протокол SIP не призначений для передачі призначеної для користувача інформації, в його повідомленнях може переноситися інформація лише обмеженого об'єму. При перенесенні через мережу дуже великого повідомлення SIP не виключена

його фрагментація на рівні IP, що може вплинути на якість передачі інформації.

У глобальній інформаційній мережі Інтернет вже досить давно функціонує експериментальна ділянка Mbone, яка утворена з мережевих вузлів, що підтримують режим багатоадресної розсилки мультимедійній інформації. Найважливішою функцією Mbone є підтримка мультимедійних конференцій, а основним способом запрошення учасників до конференції став протокол SIP.

Протокол SIP передбачає організацію конференцій трьох видів:

• у режимі багатоадресної розсилки (multicasting), коли інформація передається на одну multicast-адрес, а потім доставляється мережею кінцевим адресатам;

• за допомогою пристрою управління конференції (MCU), до якого учасники конференції передають інформацію в режимі крапка-крапка, а воно, у свою чергу, обробляє її (тобто змішує або комутує) і розсилає учасникам конференції;

• шляхом з'єднання кожного користувача з кожним в режимі крапка-крапка.

Протокол SIP дає можливість приєднання нових учасників до вже існуючого сеансу зв'язку, тобто двосторонній сеанс може перейти в конференцію.

Розроблені методи спільної роботи цього протоколу з перетворювачем мережевих адрес - Network Address Translator (NAT).

Для організації взаємодії з існуючими застосуваннями IP-сетей і для забезпечення мобільності користувачів протокол SIP використовує адресу, подібну до адреси електронної пошти. Як адреси робочих станцій використовуються спеціальні універсальні покажчики ресурсів - URL (Universal Resource Locators), так звані SIP URL

SIP-адреса бувають чотирьох типів:

• имя@домен;

• имя@хост

• имя@IP-адрес;

• №телефона@шлюз.

частина - це ім'я користувача, зареєстрованого в домені або на робочій станції. Якщо друга частина адреси ідентифікує який-небудь шлюз, то в першій вказується телефонний номер абонента.

У другій частині адреси вказується ім'я домена, робочої станції або шлюзу. Для визначення IP-адреса пристрою необхідно звернутися до служби доменних імен - Domain Name Service (DNS). Якщо ж в другій частині SIP-адреса розміщується IP-адрес, то з робочою станцією можна зв'язатися безпосередньо.

На початку SIP-адреса ставиться слово «sip:», вказуюче, що це саме SIP-адрес, оскільки бувають та інші (наприклад, «mailto:»). Нижче наводяться приклади SIP-адресов:

sip: als@rts.loniis.ru

sip: user1@192.168.100.152

sip: 294-75-47@gateway.ru

В деякому розумінні прародичем протоколу SIP є протокол перенесення гіпертексту - HTTP (Hypertext Transfer Protocol, RFC 2068). Протокол SIP успадкував від нього синтаксис і архітектуру «клієнт-сервер».

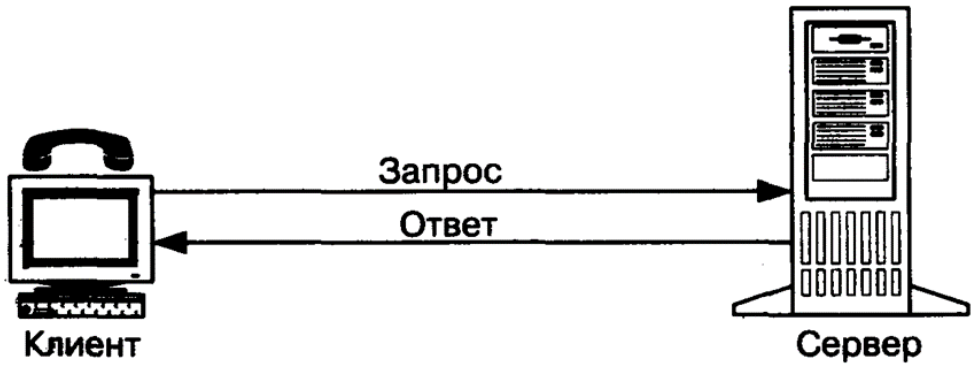


Рис.8.8. Архітектура "клієнт-сервер

Клієнт видає запити, в яких вказує, що він бажає отримати від сервера. Сервер приймає запит, обробляє його і видає відповідь, яка може містити повідомлення про успішне виконання запиту, повідомлення про помилку або

інформацію, що зажадалася клієнтом.

Управління процесом обслуговування виклику розподілене між різними елементами мережі SIP. Основним функціональним елементом, що реалізовує функції управління з'єднанням, є термінал. Останні елементи мережі відповідають за маршрутизацію викликів, а в деяких випадках надають додаткові послуги.

У разі, коли клієнт і сервер взаємодіють безпосередньо з користувачем (тобто реалізовані в крайовому устаткуванні користувача), вони називаються, відповідно, клієнтом агента користувача - User Agent Client (UAC) - і сервером агента користувача - User Agent Server (UAS).

Слід особливо відзначити, що сервер UAS і клієнт UAC можуть (але не зобов'язані) безпосередньо взаємодіяти з користувачем, а інші клієнти і сервери SIP цього робити не можуть. Якщо в пристрої присутні і сервер UAS, і клієнт UAC, то воно називається агентом користувача - User Agent (UA), а за своєю суттю є термінальним устаткуванням SIP.

Окрім терміналів визначено двох основних типів мережевих елементів SIP: проксі-сервер (proxy server) і сервер переадресації (redirect server).

Проксі-сервер (proxy - представник) представляє інтереси користувача в мережі. Він приймає запити, обробляє їх і, залежно від типа запиту, виконує певні дії. Це може бути пошук і виклик користувача, маршрутизація запиту, надання послуг і так далі. Проксі-сервер складається з клієнтської і серверної частин, тому може приймати виклики, ініціювати власні запити і повертати відповіді. Проксі - сервер може бути фізично поєднаний з сервером визначення місця (в цьому випадку він називається registrar) розташування або існувати окремо від цього сервера, але мати можливість взаємодіяти з ним по протоколах LDAP (RFC 1777), rwhois (RFC 2167) і по будь-яких інших протоколах.

Передбачено двох типів проксі-серверів - із збереженням станів (stateful) і без збереження станів (stateless).

Сервер першого типа зберігає в пам'яті вхідний запит, який з'явився

причиною генерації одного або декількох витікаючих запитів. Ці витікаючі запити сервер також запам'ятовує Всі запити зберігаються в пам'яті сервера лише до закінчення транзакції, тобто до здобуття відповідей на запити [8.18].

Сервер першого типа дозволяє надати більшу кількість послуг, але працює повільніше, ніж сервер другого типа. Він може застосовуватися для обслуговування невеликої кількості клієнтів, наприклад, в локальній мережі. Проксі-сервер повинен зберігати інформацію про стани, якщо він:

- використовує протокол TCP для передачі сигнальної інформації;
- працює в режимі багатоадресної розсилки сигнальній інформації;
- розмножує запити.

Останній випадок має місце, коли проксі-сервер веде пошук користувача, що викликається, відразу в декількох напрямах, тобто один запит, який прийшов до проксі-сервера, розмножується і передається одночасно по всіх цих напрямах.

Сервер без збереження станів просто ретранслює запити і відповіді, які отримує. Він працює швидше, ніж сервер першого типа, оскільки ресурс процесора не витрачається на запам'ятовування станів, унаслідок чого сервер цього типа може обслужити більшу кількість користувачів. Недоліком такого сервера є те, що на його базі можна реалізувати лише найбільш прості послуги. Втім, проксі-сервер може функціонувати як сервер із збереженням станів для одних користувачів і як сервер без збереження станів - для інших.

Алгоритм роботи користувачів з проксі-сервером виглядає таким чином. Постачальник послуг IP-телефонии повідомляє адресу проксі-сервера своїм користувачам. Користувач передає до проксі-сервера запит з'єднання [8.18, 8.24]. Сервер обробляє запит, визначає місце розташування користувача, що викликається, і передає запит цьому користувачеві, а потім отримує від нього відповідь, підтверджуючу успішну обробку запиту, і транслює цю відповідь користувачеві, що передав запит. Проксі-сервер може модифікувати деякі заголовки повідомлень, які він транслює, причому кожен сервер, що обробив

запит в процесі його передачі від джерела до приймача, повинен вказати це в SIP-запросе для того, щоб відповідь на запит повернулася по такій же дорозі.

Сервер без збереження станів просто ретранслює запити і відповіді, які отримує. Він працює швидше, ніж сервер першого типа, оскільки ресурс процесора не витрачається на запам'ятовування станів, унаслідок чого сервер цього типа може обслужити більшу кількість користувачів. Недоліком такого сервера є те, що на його базі можна реалізувати лише найбільш прості послуги. Втім, проксі-сервер може функціонувати як сервер із збереженням станів для одних користувачів і як сервер без збереження станів - для інших.

Алгоритм роботи користувачів з проксі-сервером виглядає таким чином. Постачальник послуг IP-телефонии повідомляє адресу проксі-сервера своїм користувачам. Користувач передає до проксі-сервера запит з'єднання. Сервер обробляє запит, визначає місце розташування користувача, що викликається, і передає запит цьому користувачеві, а потім отримує від нього відповідь, підтверджуючу успішну обробку запиту, і транслює цю відповідь користувачеві, що передав запит. Проксі-сервер може модифікувати деякі заголовки повідомлень, які він транслює, причому кожен сервер, що обробив запит в процесі його передачі від джерела до приймача, повинен вказати це в SIP-запросе для того, щоб відповідь на запит повернулася по такій же дорозі.

Сервер переадресації призначений для визначення поточної адреси користувача, що викликається. Користувач передає до сервера повідомлення з відомою йому адресою користувача, що викликається, а сервер забезпечує переадресацію виклику на поточну адресу цього користувача. Для реалізації цієї функції сервер переадресації повинен взаємодіяти з сервером визначення місця розташування [8.18 - 8.24].

Сервер переадресації не термінує виклики як сервер RAS і не ініціює власні запити як проксі-сервер. Він лише повідомляє адресу або користувача, що викликається, або проксі-сервера. За цією адресою ініціатор запиту передає новий запит. Сервер переадресації не містить клієнтську частину програмного

забезпечення.

Але користувачеві не обов'язково зв'язуватися з яким-небудь SIP-сервером. Він може сам викликати іншого користувача за умови, що знає його поточна адреса.

Користувач може переміщатися в межах мережі, тому необхідний механізм визначення його місця розташування у нинішній момент часу.

Наприклад, співробітник підприємства виїжджає у відрядження, і всі виклики, адресовані йому, мають бути направлені в інше місто на його тимчасове місце роботи. Про те, де він знаходиться, користувач інформує спеціальний сервер за допомогою повідомлення REGISTER.

Можливі два режими реєстрації: користувач може повідомити свою нову адресу один раз, а може реєструватися періодично через певні проміжки часу. Перший спосіб личить для випадку, коли термінал, доступний користувачеві, включений постійно, і його не переміщають по мережі, а другий - якщо термінал часто переміщається або вимикається.

Для зберігання поточної адреси користувача служить сервер визначення місця розташування користувачів, що є базою даних адресної інформації. Окрім постійної адреси користувача, в цій базі даних може зберігатися один або декілька поточних адрес.

Цей сервер може бути поєднаний з проксі-сервером (в такому разі він називається registrar) або бути реалізований окремо від проксі-сервера, але мати можливість зв'язуватися з ним.

У RFC 2543 сервер визначення місця розташування представлений як окремий мережевий елемент, але принципи його роботи в цьому документі не регламентовані. Варто звернути увагу на те, що користувач, якому потрібна поточна адреса користувача, що викликається, не зв'язується з сервером визначення місця розташування безпосередньо. Цю функцію виконують SIP-серверы за допомогою протоколів LDAP (RFC 1777), rwhois (RFC 2167), або інших протоколів (рис. 8.9).

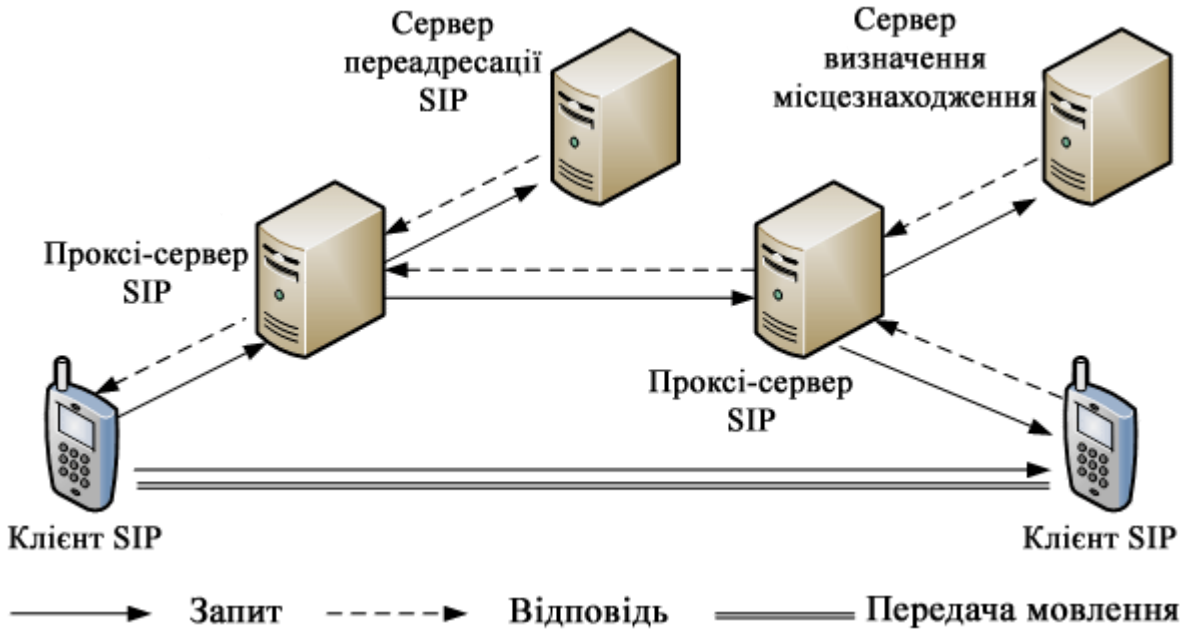


Рис.8.9. Приклад мережі натбазі протоколу SIP

Мережі SIP будуються з елементів трьох основних типів: терміналів, проксі-серверів і серверів переадресації.

SIP-сервери є окремими функціональними мережевими елементами. Фізично вони можуть бути реалізовані на базі серверів локальної мережі, які, окрім виконання своїх основних функцій, також оброблятимуть SIP-сообщения.

Термінали ж можуть бути двох типів: персональний комп'ютер із звуковою платою і програмним забезпеченням SIP-клиента (UA) або SIP-телефон, що підключається безпосередньо до ЛВС Ethernet. Таким чином, користувач локальної обчислювальної мережі передає всі запити до свого SlP-серверу, а той обробляє їх і забезпечує встановлення з'єднань.

Шляхом програмування сервер можна забудувати на різні алгоритми роботи: він може обслуговувати частину користувачів (наприклад, керівництво підприємства або особливо поважних осіб) по одних правилах, а іншу частину - по інших. Можливо також, що сервер враховуватиме категорію і терміновість викликів, а також вести нарахування плати за розмови.

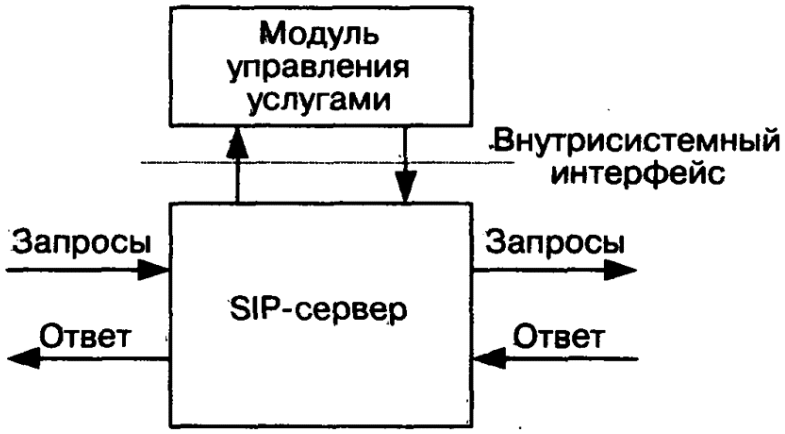


Рис. 8.10. Структурна схема організації послуг SIP-сервера

Модуль управління послугами відповідає за надання послуг і за загальне управління сервером. Прийняті сервером запити і відповіді поступають в модуль управління послугами і обробляються ним, на підставі чого визначається реакція на отримані повідомлення.

Інтерфейс людина-машина дозволяє гнучко міняти налаштування сервера і вести моніторинг мережі.Згідно архітектурі «клієнт-сервер» всі повідомлення діляться на запити, передавані від клієнта до сервера, і на відповіді сервера клієнтові.

Наприклад, аби ініціювати встановлення з'єднання, користувач повинен повідомити серверу ряд параметрів, зокрема, адреса користувача, що викликається, параметри інформаційних каналів і ін. Ці параметри передаються в спеціальному SIP-запросе. Від користувача, що викликається, до того, що викликає передається відповідь на запит, що також містить ряд параметрів.

Всі повідомлення протоколу SIP (запити і відповіді), є послідовності текстових рядків, закодованих відповідно до документа RFC 2279. Структура і синтаксис повідомлень SIP, як вже згадувалося раніше, ідентичні використовуваним в протоколі HTTP.

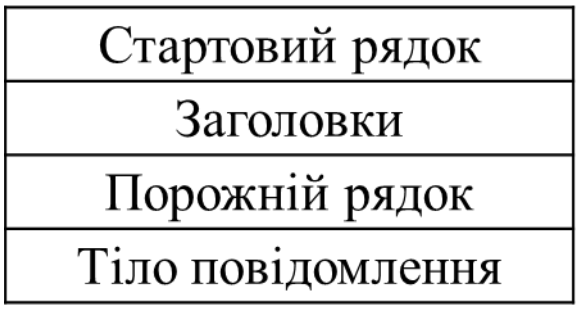


Рис. 8.11. Повідомлення протоколу SIP

Стартовий рядок є початковим рядком будь-якого SIP-сообщения.

Якщо повідомлення є запитом, в цьому рядку вказуються тип запиту, адресат і номер версії протоколу.

Якщо повідомлення є відповіддю на запит, в стартовому рядку вказуються номер версії протоколу, тип відповіді і його коротка розшифровка, призначена лише для користувача.

Заголовки повідомлень містять відомості про відправника, адресата, дорогу дотримання і ін., загалом, переносять інформацію, необхідну для обслуговування даного повідомлення.

Про типа заголовка можна взнати по його імені. Воно не залежить від регістра (тобто букви можуть бути прописні і рядкові), але звичайне ім'я пишуть з великої букви, за якою йдуть рядкові.

Повідомлення протоколу SIP можуть містити так зване тіло повідомлення. У запитах INVITE і OPTIONS тіло повідомлення містить опис сеансів зв'язку, наприклад, у форматі протоколу SDP.

Запит BYE тіла повідомлення не містить, а ситуація із запитом REGISTER підлягає подальшому вивченню.

З відповідями справа йде інакше: будь-які відповіді можуть містити тіло повідомлення, але вміст тіла в них буває різним.

У протоколі SIP визначено чотири види заголовків:

• Загальні заголовки, присутні в запитах і відповідях;

• Заголовки вмісту, переносять інформацію про розмір тіла повідомлення або про джерело запиту (починаються із слова «Content»);

• Заголовки запитів, передавальні додаткову інформацію про запит;

• Заголовки відповідей, передавальні додаткову інформацію про відповідь.

Заголовок містить назву, за якою, відокремлене двокрапкою, слідує значення заголовка. У полі значення містяться передавані дані. Слід зазначити, що якщо сервер приймає повідомлення, заголовки яких йому не відомі, то ці заголовки ігноруються.

зв'язку або всіх реєстрації окремого клієнта, він подібний до мітки з'єднання (call reference) в сигналізації DSS-1. Значення ідентифікатору привласнює сторона, яка ініціює виклик. Заголовок CALL-ID складається з буквено-числового значення і імені робочої станції, яка привласнила значення цьому ідентифікатору. Між ними повинен стояти символ @, наприклад, 2345call@rts.loniis.ru. Можлива наступна ситуація: до однієї мультимедійної конференції відносяться декілька з'єднань, тоді всі вони матимуть різні ідентифікатори CALL-ID.

Якщо необхідне візуальне виведення імені користувача, наприклад, на дисплей, то ім'я користувача також розміщується в полі Te.

Заголовок Te - визначає адресата. Окрім SIP-адреса тут може стояти параметр «tag» для ідентифікації конкретного терміналу користувача (наприклад, домашнього, робочого або стільникового телефону) у тому випадку, коли всі його термінали зареєстровані під однією адресою SIP URL.

Запит може множитися і досягти різних терміналів користувача; аби їх розрізняти, необхідно мати мітку tag. Її вставляє в заголовок термінальне устаткування викликаного користувача при відповіді на прийнятий запит.

Заголовок From - ідентифікує відправника запиту; по структурі аналогічний полю Te [8.18, 8.24]

Таблиця 8.8. Види заголовків повідомлень SIP

Загальні заголовки	Заголовки вмісту	Заголовки запитів	Заголовки відповідей
CALL-ID (ідентифікатор сеансу зв'язку)	Content-Encoding (кодування тіла повідомлення)	Accept (приймається)	Allow (дозвіл)
Contact (контактувати)	Content-Length (розмір тіла повідомлення)	Accent-Encoding (метод кодування підтримується)	Proxy-Authenticate (підтвердження достовірності проксі-сервера)
CSeq (послідовність)	Content-Type (тип вмісту)	Accent-Language (мова підтримується)	Retro-After (повторити через деякий час)
Date (Дата)		Authorization (авторизація)	Server (сервер)
Encryption (шифрування)			Unsupported (не підтримується)
Expires (спрацьовування таймера)		Hide (приховати)	Warning (попередження)
From (джерело запиту)		Max-forwards (максимальна кількість переадресацій)	VWVW-Authenticate (підтвердження достовірності WWW-сервера)
Record-Route (запис маршруту)		Organization (організація)	
Timestamp (мітка часу)		Priority (пріоритет)	
To (Адресат)		Proxy-Authorization (авторизація проксі-сервера)	
Via (через)		Proxy-Require (потрібний проксі-сервер)	
		Route (маршрут)	
		Require (потрібний)	
		Response-Key (ключ кодування відповіді)	
		Subject (тема)	
		User-Agent (агент користувача)	

Заголовок CSeq - унікальний ідентифікатор запиту, що відноситься до одного з'єднання. Він служить для кореляції запиту з відповіддю на нього. Заголовок складається з двох частин: натурального числа з діапазону від 1 до 232 і типа запиту [8.18, 8.24]. Сервер повинен перевіряти значення CSeq в кожному запиті, що приймається, і вважати запит новим, якщо значення CSeq більше попереднього. Приклад заголовка: CSeq: 2 INVITE.

Заголовок Via служить для того, щоб уникнути ситуації, в яких запит піде по замкнутій дорозі, а також для тих випадків, коли необхідно, аби запити і відповіді обов'язково проходили поодинці і тій же дорозі (наприклад, в разі використання міжмережевого екрану - firewall).

Річ у тому, що запит може проходить через декілька проксі-серверів, кожен з яких приймає, обробляє і переправляє запит до наступного проксі-сервера, і так до тих пір, поки запит не досягне адресата. Таким чином, в заголовку Via вказується вся дорога, пройденный запитом: кожен проксі-сервер додає поле зі своєю адресою. При необхідності (наприклад, аби забезпечити

секретність) дійсна адреса може ховатися. Наприклад, запит на своїй дорозі оброблявся двома сі-серверами: спочатку сервером loniis.ru, потім sip.telecom.com. Тоді в запиті з'являться наступні поля:

Via: SIP/2.0/UDP sip.telecom.com:5060; branch=721 e418c4.1 Via: SIP/2.0/UDP loniis.ru: 5060, де параметр «branch» означає, що на сервері sip.telecom.com запит був розмножений і направлений одночасно по різних напрямах, і наш запит був переданий по напряму, який ідентифікується таким чином: 721e418c4.1.

Вміст полів Via копіюється із запитів у відповіді на них, і кожен сервер, через який проходить відповідь, видаляє поле Via зі своїм ім'ям.

Таблиця 8.9. Зв'язок заголовків із запитами і відповідями протоколу SIPv2.Q

Назва заголовка	Місце використання заголовка	ACK	BYE	CAN	INV	OPT	REG
Accept	Заголовок в запитах	F	F	F	0	0	0
Accept	Заголовок відповідає 415	F	F	F	0	0	0
Accent-Encoding	Заголовок в запитах	F	F	F	0	0	0
Accent-Encoding	Заголовок відповідає 415	F	F	F	0	0	0
Accent-Language	Заголовок в запитах	F	0	0	0	0	0
Accent-Language	Заголовок відповідає 415	F	0	0	0	0	0
Allow	Заголовок відповідає 200	F	F	F	F	M	F
Allow	Заголовок відповідає 405	0	0	0	0	0	0
Authorization	Заголовок в запитах	0	0	0	0	0	0
CALL-ID	Загальний заголовок - копіюється із запитів у відповіді	M	M	M	M	M	M
Contact	Заголовок в запитах	0	F	F	0	0	0
Contact	Заголовок у відповідях 1xx	F	F	F	0	0	F
Contact	Заголовок у відповідях 2xx	F	F	F	0	0	0
Contact	Заголовок у відповідях 3xx	F	0	F	0	0	0
Contact	Заголовок відповідає 485	F	0	F	0	0	0
Content-Encoding	Заголовки вмісту	0	F	F	0	0	0
Content-Length	Заголовки вмісту	0	F	F	0	0	0
Content-Type	Заголовки вмісту	*	F	F	*	*	*
Cseq	Загальний заголовок - копіюється із запитів у відповіді	M	M	M	M	M	M
Date	Заголовок у відповідях	0	0	0	0	0	0
Encryption	Заголовок у відповідях	0	0	0	0	0	0
Expires	Заголовок у відповідях	F	F	F	0	F	0
From	Загальний заголовок - копіюється із запитів у відповіді	M	M	M	M	M	M

Продовження таблиці 8.9. Зв'язок заголовків із запитами і відповідями протоколу SIPv2.Q

Назва заголовка	Місце використання заголовка	ACK	BYE	CAN	INV	OPT	REG
Hide	Заголовок в запитах	0	0	0	0	0	0
Max-forwards	Заголовок в запитах	0	0	0	0	0	0
Organization	Загальний заголовок	F	F	F	0	0	0
Proxy-Authenticate	Заголовок відповідає 407	0	0	0	0	0	0
Proxy-Authorization	Заголовок в запитах	0	0	0	0	0	0
Proxy-Require	Заголовок в запитах	0	0	0	0	0	0
Priority	Заголовок в запитах	F	F	F	0	F	F
Require	Заголовок в запитах	0	0	0	0	0	0
Retry-After	Заголовок в запитах	F	F	F	P	F	0
Retry-After	Заголовок у відповідях 404, 480, 486, 503, 600 і 603	0	0	0	0	0	0
Response-Key	Заголовок в запитах	F	0	0	0	0	0
Record-Route	Заголовок в запитах	0	0	0	0	0	0
Record-Route	Заголовок у відповідях 2xx	0	0	0	0	0	0
Route	Заголовок в запитах	0	0	0	0	0	0
Server	Заголовок у відповідях	0	0	0	0	0	0
Subject	Заголовок в запитах	F	F	F	0	F	F
Timestamp	Загальний заголовок	0	0	0	0	0	0
To	Загальний заголовок - копіюється із запитів у відповіді	M	M	M	M	M	M
Unsupported	Заголовок відповідає 420	0	0	0	0	0	0
User-Agent	Загальний заголовок	0	0	0	0	0	0
Via	Загальний заголовок - копіюється із запитів у відповіді	M	M	M	M	M	M
Warning	Заголовок у відповідях	0	0	0	0	0	0
WWW-Authenticate	Заголовок відповідає 401	0	0	0	0	0	0

* Примітка - поле необхідне лише у разі, коли тіло повідомлення містить яку-небудь інформацію, тобто не є порожнім.

У заголовок Record-route проксі-сервер вписує свою адресу - SIP URL, - якщо хоче, аби подальші запити пройшли через нього.

Заголовок Content-Type визначає формат опису сеансу зв'язку. Сам опис сеансу, наприклад, у форматі протоколу SDP, включається в зміст повідомлення.

Заголовок Content-Length вказує розмір тіла повідомлення. Після того, як ми розглянули найбільш заголовки повідомлень протоколу SIP, що часто зустрічаються, слід звернути увагу на те, що запити і відповіді на них можуть включати лише певний набір заголовків (Таблиця). Тут знову буква «M» означає обов'язкову присутність заголовка в повідомленні, буква «O» - не обов'язкова присутність, буква «F» - забороняє присутність заголовка.

У справжній версії протоколу SIP визначено шість типів запитів [8.18, 8.24]. Кожен з них призначений для виконання досить широкого круга завдань, що є явною гідністю протоколу SIP, оскільки завдяки цьому число повідомлень, якими обмінюються термінали і сервери, зведене до мінімуму. За допомогою запитів клієнт повідомляє про поточне місце розташування, запрошує

користувачів взяти участь в сеансах зв'язку, модифікує вже встановлені сеанси, завершує їх і так далі Сервер визначає типа прийнятого запиту по назві, вказаній в стартовому рядку. У тому ж рядку в полі REQUEST-URI вказана SIP-адрес устаткування, якому цей запит адресований. Вміст полів Те і REQUEST-URI може розрізнятися, наприклад, в полі Те може бути вказаний публікована адреса абонента, а в поле REQUEST-URI - поточну адресу користувача.

Запит INVITE запрошує користувача взяти участь в сеансі зв'язку. Він зазвичай містить опис сеансу зв'язку, в якому вказується вигляд інформації, що приймається, і параметри (список можливих варіантів параметрів), необхідні для прийому інформації, а також може вказуватися вигляд інформації, яку користувач, що викликається, бажає передавати. Відповідає на запит типа INVITE вказується вигляд інформації, яка прийматиметься користувачем, що викликається, і, крім того, може вказуватися вигляд інформації, яку користувач, що викликається, збирається передавати (можливі параметри передачі інформації) [8.18 - 8.24].

У цьому повідомленні можуть міститися також дані, необхідні для аутентифікації абонента, і, отже, доступу клієнтів до SIP-серверу. При необхідності змінити характеристики вже організованих каналів передається запит INVITE з новим описом сеансу зв'язку. Для запрошення нового учасника до вже встановленого з'єднання також використовується повідомлення INVITE.

Запит АСЬК підтверджує прийом відповіді на запит INVITE. Слід зазначити, що запит АСЬК використовується лише спільно із запитом INVITE, тобто цим повідомленням устаткування користувача показує, що воно отримало остаточну відповідь на свій запит INVITE. У повідомленні АСЬК може міститися остаточний опис сеансу зв'язку, передаваний користувачем, що запитує.

Запит CANCEL відміняє обробку раніше переданих запитів з тими ж, що і в запиті CANCEL, значеннями полів CALL-ID, To, From і CSeq, але не впливає на ті запити, обробка яких вже завершена. Наприклад, запит CANCEL

застосовується тоді, коли проксі-сервер розмножує запити для пошуку користувача по декількох напрямах і в одному з них його знаходить.

Обробку запитів, розісланих у всіх останніх напрямах, сервер відміняє за допомогою повідомлення CANCEL.

Запитом BYE устаткування користувача, що викликається або викликає, завершує з'єднання. Сторона, що отримала запит BYE, повинна припинити передачу мовної (мультимедійною) інформації і підтвердити його виконання відповіддю 200 ОК.

За допомогою запиту типа REGISTER користувач повідомляє своє поточне місце розташування. У цьому повідомленні містяться наступні поля:

• Поле Те містить адресну інформацію, яку треба зберегти або модифікувати на сервері;

• Поле From містить адресу ініціатора реєстрації. Зареєструвати користувача може або він сам, або інша особа, наприклад, секретар може зареєструвати свого начальника;

• Поле Contact містить нову адресу користувача, по якому повинні передаватися всі подальші запити INVITE. Якщо в запиті REGISTER поле Contact відсутнє, то реєстрація залишається колишньою. В разі відміни реєстрації тут поміщається символ «*»;

• У полі Expires вказується час в секундах, протягом якого реєстрація дійсна. Якщо дане поле відсутнє, то за умовчанням призначається час - 1 година, після чого реєстрація відміняється. Реєстрацію можна також відмінити, передавши повідомлення REGISTER з полем Expires, якому привласнено значення Про, і з відповідним полем Contact.

Запитом OPTIONS користувач, що викликається, запрошує інформацію про функціональні можливості термінального устаткування користувача, що викликається. У відповідь на цей запит устаткування користувача, що викликається, повідомляє необхідні відомості. Вживання запиту OPTIONS обмежене тими випадками, коли необхідно дізнатися про функціональні

можливості устаткування до встановлення з'єднання. Для встановлення з'єднання запит цього типа не використовується.

• У полі Expires вказується час в секундах, протягом якого реєстрація дійсна. Якщо дане поле відсутнє, то за умовчанням призначається час - 1 година, після чого реєстрація відміняється. Реєстрацію можна також відмінити, передавши повідомлення REGISTER з полем Expires, якому привласнено значення Про, і з відповідним полем Contact.

Запитом OPTIONS користувач, що викликається, запрошує інформацію про функціональні можливості термінального устаткування користувача, що викликається. У відповідь на цей запит устаткування користувача, що викликається, повідомляє необхідні відомості. Вживання запиту OPTIONS обмежене тими випадками, коли необхідно дізнатися про функціональні можливості устаткування до встановлення з'єднання. Для встановлення з'єднання запит цього типа не використовується.

Після випробувань протоколу SIP в реальних мережах виявилось, що для вирішення ряду завдань вищезгаданих шести типів запитів недостатньо. Тому можливо, що в протокол будуть введені нові повідомлення. Так, в поточній версії протоколу SIP не передбачений спосіб передачі інформації управління з'єднанням або іншої інформації під час сеансу зв'язку. Для вирішення цього завдання був запропонований новий тип запиту - INFO. Він може використовуватися в наступних випадках:

• для перенесення сигнальних повідомлень ТфОП/isdn/cotobbix мереж між шлюзами протягом розмовної сесії;

• для перенесення сигналів DTMF протягом розмовної сесії;

• для перенесення білінгової інформації.

типовий запит типа INVITE:

INVITE sip: watson@boston.bell-tel.com SIP/2.0 Via: SIP/2.0/UDP kton.bell-tel.com From: A. Bell <sip: a.g.bell@bell-tel.com> To: T. Watson <sip: watson@bell-tel.com> CALL-ID: 3298420296@kton.bell-tel.com Cseq: 1 INVITE

Content-Type: application/sdp Content-Length: ...

v=0

o=bell 53655765 2353687637 IN IP4 12&.3.4.5

C=IN IP4 kton.bell-tel.com

m=audio 3456 RTP/AVP 0345

В даному прикладі користувач Bell (a.g.bell@bell-tel.com) викликає користувача Watson (watson@bell-tel.com). Запит передається до проксі-сервера (boston.bell-tel.com). У полях Te і From перед адресою коштує запис, який користувач бажає вивести на дисплей користувача, що викликається. У змісті повідомлення устаткування користувача вказує у форматі протоколу SDP, що воно може приймати в порту 3456 мовну інформацію, упаковану в пакети RTP і закодовану по одному з наступних алгоритмів кодування: 0 - PCMU, 3 - GSM, 4 - G.723 і 5 - DVI4.

При передачі повідомлень протоколу SIP, упакованих в сигнальні повідомлення протоколу UDP, існує вірогідність того, що розмір запиту або відповіді виявиться більше максимально допустимого для даної мережі, і станеться фрагментація пакету. Аби уникнути цього, використовується стислий формат імен основних заголовків, подібно до того, як це робиться в протоколі SDP:

Таблиця 8.10. Стислий формат імен основних заголовків

Стисла форма імені	Повна форма імені
з	Content-Type
e	Content- Encoding
f	From
i	CALL-ID
m	Contact (від "moved")
1	Content-Length
s	Subject
t	To
v	Via

При написанні імен заголовків в стислому вигляді повідомлення INVITE,

виглядатиме таким чином:

INVITE sip: watson@boston.bell-tel.com SIP/2.0 v: SIP/2.0/UDP kton.bell-tel.com f: A. Bell <sip: a.g.bell@bell-tel.com> t: T. Watson <sip: watson@bell-tel.com> i: 3298420296@kton.bell-tel.com Cseq: 1 INVITE з: application/sdp 1: ...

v=0

o=bell 53655765 2353687637 IN IP4 128.3.4.5

C=IN IP4 kton.bell-tel.com

m=audio 3456 RTP/AVP 0345

Таблиця 8.11. Приклад запиту INVITE із скороченими заголовками

Тип запиту	Опис запиту
INVITE	Запрошує користувача до сеансу зв'язку. Містить SDP-описание сеансу
ACK	Підтверджує прийом остаточної відповіді на запит INVITE
BYE	Завершує сеанс зв'язку. Може бути переданий будь-якій із сторін, що беруть участь в сеансі
CANCEL	Відміняє обробку запитів з тими ж заголовками CALL-ID, To, From і CSeq, що і в самому запиті CANCEL
REGISTER	Переносить адресну інформацію для реєстрації користувача на сервері визначення місця розташування
OPTION	Запрошує інформацію про функціональні можливості терміналу

Після прийому і інтерпретації запиту, адресат (проксі-сервер) передає відповідь на цей запит. Вміст відповідей буває різним:

- підтвердження встановлення з'єднання,

- передача запитаної інформації,

- відомості про несправності і так далі.

Структуру відповідей і їх види протокол SIP успадкував від протоколу HTTP. Визначено шість типів відповідей, що несуть різне функціональне навантаження. Тип відповіді кодується тризначним числом. Найважливішою є перша цифра, яка визначає клас відповіді, останні дві цифри лише доповнюють першу. В деяких випадках устаткування навіть може не знати всі коди відповідей, але воно обов'язково повинне інтерпретувати першу цифру відповіді.

Всі відповіді діляться на дві групи: інформаційні і фінальні.

Інформаційні відповіді показують, що запит знаходиться у стадії обробки. Вони кодуються тризначним числом, що починається з одиниці, - 1xx. Деякі інформаційні відповіді, наприклад, 100 Trying, призначені для установки на нуль таймерів, які запускаються в устаткуванні, що передало запит. Якщо до моменту спрацьовування таймера відповідь на запит не отримана, то вважається, що цей запит втрачений і може (по розсуду виробника) бути переданий повторно. Одна з поширених відповідей -180 Ringing; за призначенням він ідентичний сигналу «Контроль посилки виклику» в ТМЗК і означає, що користувач, що викликається, отримує сигнал про вхідний виклик.

Фінальні відповіді кодуються тризначними числами, що починаються з цифр 2, 3, 4, 5 і 6. Вони означають завершення обробки запиту і містять, коли це потрібно, результат обробки запиту. Призначення фінальних відповідей кожного типа розглядається нижче.

Відповіді 2xx означають, що запит був успішно оброблений. В даний час зі всіх відповідей типа 2xx визначений лише один -200 OK. Його значення залежить від того, на який запит він відповідає:

• відповідь 200 OK на запит INVITE означає, що устаткування, що викликається, згодне на участь в сеансі зв'язку; у телі відповіді вказуються функціональні можливості цього устаткування;

• відповідь 200 OK на запит BYE означає завершення сеансу зв'язку, в телі відповіді жодної інформації не міститься;

• відповідь 200 OK на запит CANCEL означає відміну пошуку, в телі відповіді жодної інформації не міститься;

• відповідь 200 OK на запит REGISTER означає, що реєстрація прошла успішно;

• відповідь 200 OK на запит OPTION служить для передачі відомостей про функціональні можливості устаткування, ці відомості містяться в телі відповіді.

Відповіді 3xx інформують устаткування користувача про нове місце

розташування користувача, що викликається, або переносять іншу інформацію, яка може бути використана для нового виклику:

• відповідає 300 Multiple Choices вказується декілька SIP-адресов, по яких можна знайти користувача, що викликається, і користувачеві пропонується вибрати один з них;

• відповідь 301 Moved Permanently означає, що користувач, що викликається, більше не знаходиться за адресою, вказаною в запиті, і направляти запити потрібно на адресу, вказану в полі Contact;

• відповідь 302 Moved Temporary означає, що користувач тимчасово (проміжок часу може бути вказаний в полі Expires) знаходиться за іншою адресою, яка вказується в полі Contact.

Відповіді 4xx інформують про те, що в запиті виявлена помилка. Після здобуття такої відповіді користувач не повинен передавати той же самий запит без його модифікації:

• відповідь 400 Bad Request означає, що запит не зрозумів із-за наявності в нім синтаксичних помилок;

• відповідь 401 Unauthorized означає, що запит вимагає проведення процедури аутентифікації користувача. Існують різні варіанти аутентифікації, і відповідає може бути вказано, який з них використовувати в даному випадку;

• відповідь 403 Forbidden означає, що сервер зрозумів запит, але відмовився його обслуговувати. Повторний запит посилати не слід. Причини можуть бути різними, наприклад, запити з цієї адреси не обслуговуються і т.д.;

• відповідь 485 Ambiguous означає, що адреса в запиті не визначає користувача, що викликається, однозначно;

• відповідь 486 Busy Here означає, що користувач, що викликається, зараз не може прийняти вхідний виклик за даною адресою. Відповідь не унеможливлює зв'язатися з користувачем за іншою адресою або, наприклад, залишити повідомлення в мовній поштовій скриньці.

Відповіді 5xx інформують про те, що запит не може бути оброблений із-

за відмови сервера:

• відповідь 500 Server Internal Error означає, що сервер не має можливості обслужити запит із-за внутрішньої помилки. Клієнт може спробувати повторно послати запит через деякий час;

• відповідь 501 Not Implemented означає, що в сервері не реалізовані функції, необхідні для обслуговування цього запиту. Відповідь передається, наприклад у тому випадку, коли сервер не може розпізнати типа запиту;

• відповідь 502 Bad Gateway інформує про те, що сервер, що функціонує як шлюз або проксі-сервер, прийняв некоректну відповідь від сервера, до якого він направив запит;

• відповідь 503 Service Unavailable говорить від тому, що сервер не може в даний момент обслужити виклик унаслідок перевантаження або проведення технічного обслуговування.

Відповіді бхх інформують про те, що з'єднання з користувачем, що викликається, встановити неможливо:

• відповідь 600 Busy Everywhere повідомляє, що користувач, що викликається, зайнятий і не може прийняти виклик в даний момент ні по одному з адрес, що є у нього. Відповідь може вказувати час, відповідний для виклику користувача;

• відповідь 603 Decline означає, що користувач, що викликається, не може або не бажає прийняти вхідний виклик. Відповідає може бути вказаний відповідний для виклику час;

• відповідь 604 Does Not Exist Anywhere означає, що користувача, що викликається, не існує.

Приклад відповіді на запит INVITE.

Запити і відповіді на них утворюють *SIP*-транзакцию. Вона здійснюється між клієнтом і сервером і включає всі повідомлення, починаючи з першого запиту і закінчуючи фінальною відповіддю. При використанні як транспорт протоколу *TCP* всі запити і відповіді, що відносяться до однієї транзакції,

передаються по одному *TCP*-соединению.

SIP/2.0 200 OK

Via: SIP/2.0/UDP kton.bell-tel.com

From: A. Bell <sip:a.g.bell@bell-tel.com>

To: <sip:watson@bell-tel .com>;

CALL-ID: 3298420296@kfcon.bell-fcel.com Cseq: 1 INVITE

Content-Type: application/sdp Content-Length: ...

v=0

o=watson 4858949 4858949 IN IP4 192.1.2.3

t=3149329600 0

c=IN IP4 bostcon.bell-tel.com

m=audio 5004 RTP/AVP 0 3

a=rtpmap:0 PCMU/8000

a=rtpmap:3 GSM/8000

В даному прикладі приведена відповідь користувача Watson на запрошення взяти участь в сеансі зв'язку, отримане від користувача Bell.

Сторона, що викликається, інформує ту, що викликає про те, що вона може приймати в порту 5004 мовну інформацію, закодовану відповідно до алгоритмів кодування PCMU, GSM.

Поля From, To, Via, CALL-ID узяті із запиту, наведеному у попередньому прикладі. З прикладу видно, що це відповідь на запит INVITE з полем CSeq:1.

Після того, як було розглянути запити і відповіді на них, можна відзначити, що протокол SIP передбачає різні алгоритми встановлення з'єднання. При цьому варто звернути увагу, що одні і ті ж відповіді можна інтерпретувати по-різному залежно від конкретної ситуації. У таблиці нижче наведені, як приклад деякі відповіді на запити, визначені протоколом SIP.

Таблиця 8.12. Відповіді на запити, визначені протоколом SIP

Код відповіді	Пояснення	Призначення
100	Trying	Запит обробляється, наприклад, сервер звертається до баз даних, але місце розташування користувача, що викликається, зараз не визначене
180	Ringing	Місце розташування користувача, що викликається, визначене. Йому дається сигнал про вхідний виклик
181	Call Is Being Forwarded	Проксі-сервер переадресує виклик до іншого користувача
182	Queued	Користувач, що викликається, тимчасово не доступний, але вхідний виклик поставлений в чергу. Коли користувач, що викликається, стане доступним, він передасть фінальну відповідь
200	OK	Команда успішно виконана
300	Multiple Choices	Користувач, що викликається, доступний по декількох адресах. Користувач може вибрати будь-який з них
301	Moved Permanently	Користувач змінив своє місце розташування, його нова адреса вказана в полі Contact
302	Moved Temporarily	Користувач тимчасово змінив своє місце розташування, його нова адреса вказана в полі Contact
305	Use Proxy	Сторона, що викликається, може прийняти вхідний виклик лише у тому випадку, коли він проходить через проксі-сервер. Користувачеві рекомендується звернутися до проксі-сервера, адреса якого вказана в полі Contact. Відповідь передається лише термінальним устаткуванням (UAS)

Протоколом SIP передбачені 3 основних види сценарію встановлення з'єднання:

- за участю проксі-сервера,

- за участю сервера переадресації,

- безпосередньо між користувачами.

Відмінність між перерахованими сценаріями полягає в тому, що по-різному здійснюється пошук і запрошення користувача, що викликається.

У першому випадку ці функції покладає на себе проксі-сервер, а користувачеві необхідно знати лише постійну SIP-адрес користувача, що викликається.

У другому випадку ця сторона самостійно встановлює з'єднання, а сервер переадресації лише реалізує перетворення постійної адреси абонента, що викликається, в його поточну адресу.

В третьому випадку користувачеві для встановлення з'єднання необхідно знати поточну адресу користувача, що викликається.

Перераховані сценарії є простими. Адже перш ніж виклик досягне адресата, він може пройти через декілька проксі-серверів, або спочатку прямує до сервера переадресації, а потім проходить через один або декілька проксі-серверів.

Крім того, проксі-сервери можуть розмножувати запити і передавати їх

по різних напрямах і так далі. Але, все ж, як вже було вже відмічено на початку параграфа, ці три сценарії є основними.

Розглянемо детально два перші сценарії; третій сценарій в даній главі розглядатися не буде.

Встановлення з'єднання за участю сервера переадресації. Адміністратор мережі повідомляє користувачам адресу сервера переадресації. Користувач передає запит INVITE (1) на відому йому адресу сервера переадресації і порт 5060, використовуваний за умовчанням). У запиті користувач вказує адресу користувача, що викликається. Сервер переадресації запрошує поточну адресу потрібного користувача в сервера визначення місця (2) розташування, який повідомляє йому цю адресу (3).

Сервер переадресації відповідає 302 Moved temporarily передає стороні поточну адресу користувача (4), що викликається, або він може повідомити список зареєстрованих адрес користувача, що викликається, і запропонувати користувачеві самому вибрати один з них. Ця сторона підтверджує прийом відповіді 302 посилкою повідомлення ACK (5).

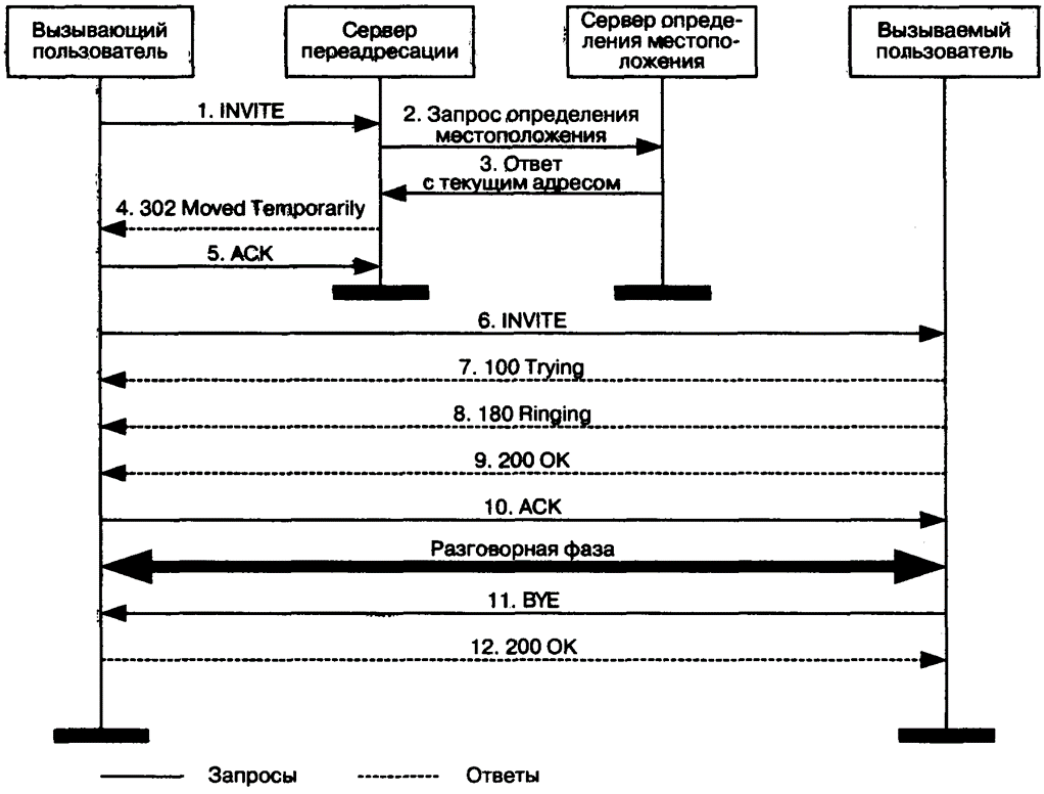


Рис. 8.12 Встановлення з'єднання за участю сервера переадресації

Тепер користувач може зв'язатися безпосередньо із стороною, що викликається. Для цього він передає новий запит INVITE (6) з тим же ідентифікатором CALL-ID, але іншим номером CSeq. У телі повідомлення INVITE вказуються дані про функціональні можливості сторони у форматі протоколу SDP. Сторона, що викликається, приймає запит INVITE і починає його обробку, про що повідомляє відповіддю 100 Trying (7) зустрічному устаткуванню для перезапуску його таймерів. Після завершення обробки запиту, що поступив, устаткування сторони, що викликається, повідомляє свого користувача про вхідний виклик, а зустрічній стороні передає відповідь 180 Ringing (8). Після прийому користувачем, що викликається, вхідного виклику видаленій стороні передається повідомлення 200 OK (9), в якому містяться дані про функціональні можливості терміналу, що викликається, у форматі протоколу SDP. Термінал користувача підтверджує прийом відповіді запитом ACK (10). На цьому фаза встановлення з'єднання закінчена і починається розмовна фаза [8.18, 8.24].

Після закінчення розмовної фази будь-якої із сторін передається запит BYE (11), який підтверджується відповіддю 200 OK (12).

Адміністратор мережі повідомляє адресу цього сервера користувачам. Користувач передає запит INVITE (1) на адресу проксі-сервера і порт 5060, використовуваний за умовчанням. У запиті користувач вказує відому йому адресу користувача, що викликається.

Проксі-сервер запрошує поточну адресу користувача, що викликається, в сервера визначення місця (2) розташування, який і повідомляє йому цю адресу (3). Далі проксі-сервер передає запит INVITE устаткуванню, що безпосередньо викликається (4). Знову в запиті містяться дані про функціональні можливості терміналу, але при цьому в запит додається поле Via з адресою проксі-сервера для того, щоб відповіді по дорозі назад йшли через нього. Після прийому і обробки запиту устаткування, що викликається, повідомляє свого користувача про вхідний виклик, а зустрічній стороні передає відповідь 180 Ringing (5),

копіюючи в нього із запиту поля Te, From, CALL-ID, CSeq і Via. Після прийому виклику користувачем зустрічній стороні передається повідомлення 200 OK (6), що містить дані про функціональні можливості терміналу, що викликається, у форматі протоколу SDP. Термінал користувача підтверджує прийом відповіді запитом АСК (7). На цьому фаза встановлення з'єднання закінчена і починається розмовна фаза.

Після закінчення розмовної фази однієї із сторін передається запит BYE (8), який підтверджується відповіддю 200 OK (9).

Всі повідомлення проходять через проксі-сервер, який може модифікувати в них деякі поля.

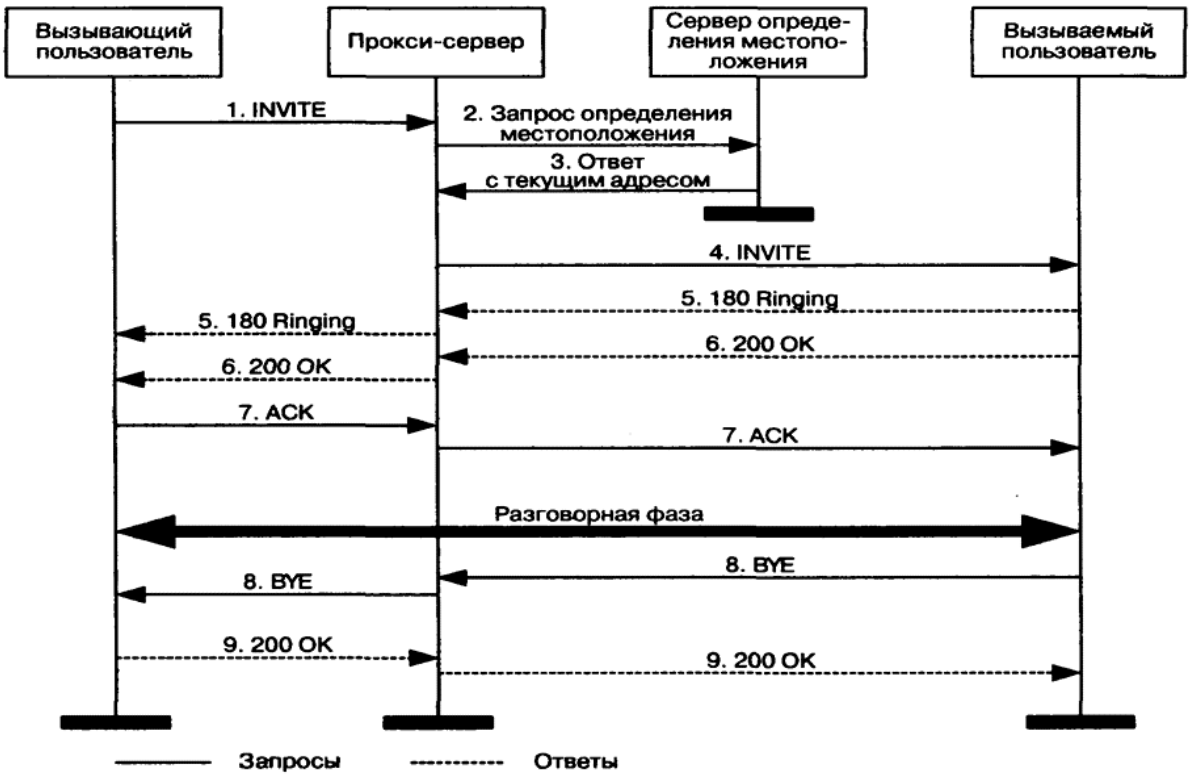


Рис.8.13. Встановлення з'єднання за участю проксі-сервера

Додаткова послуга «Перемикання зв'язку» дозволяє користувачеві перемкнути встановлене з'єднання до третьої сторони. Користувач У встановлює зв'язок з користувачем А, який, переговоривши з В, перемикає цей зв'язок до користувача С, а сам відключається.

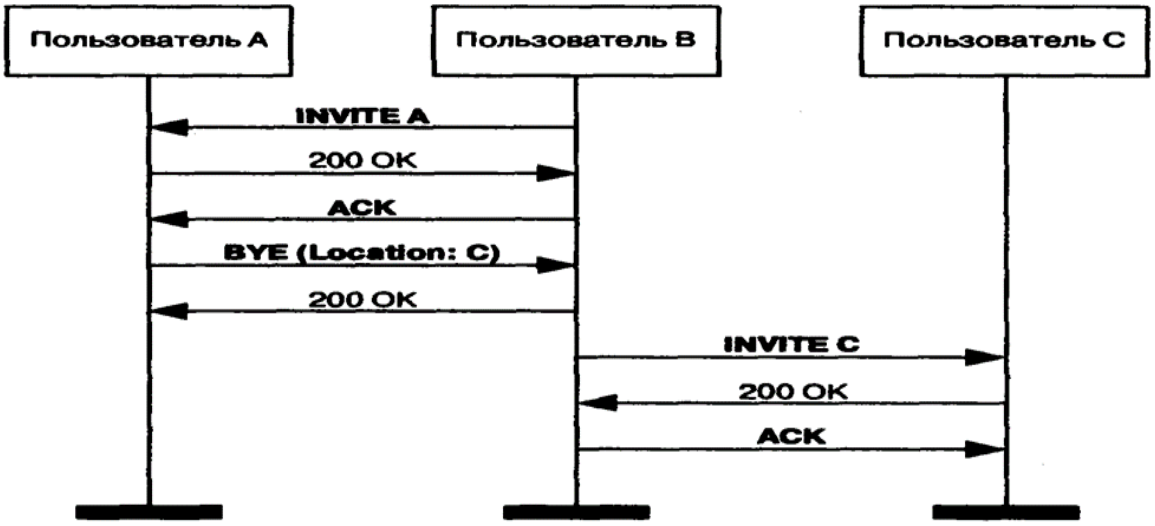


Рис. 8.14. Додаткова послуга «Перемикання зв'язку»

Додаткова послуга «Переадресація виклику» дозволяє користувачеві призначити адресу, на яку, за певних умов, слід направляти вхідні до нього виклики. Такими умовами можуть бути зайнятість користувача, відсутність його відповіді протягом заданого часу або і те, і інше; можлива також безумовна переадресація. Устаткування користувача, що замовив цю послугу, отримавши повідомлення INVITE B, перевіряє умови, в яких воно отримане, і якщо умови вимагають переадресації, передає повідомлення INVITE із заголовком Also, вказуючи в нім адресу користувача, до якого слід направити виклик. Термінал користувача, отримавши повідомлення INVITE з таким заголовком, ініціює новий виклик за адресою, вказаною в полі Also.

Рис. 8.15. Додаткова послуга «Переадресація виклику»

У нашому випадку користувач А викликає користувача В, а термінал останнього переадресує виклик до користувача С.

Додаткова послуга «Повідомлення про виклик під час зв'язку» дозволяє користувачеві, що бере участь в телефонній розмові, отримати повідомлення про те, що до нього поступив вхідний виклик

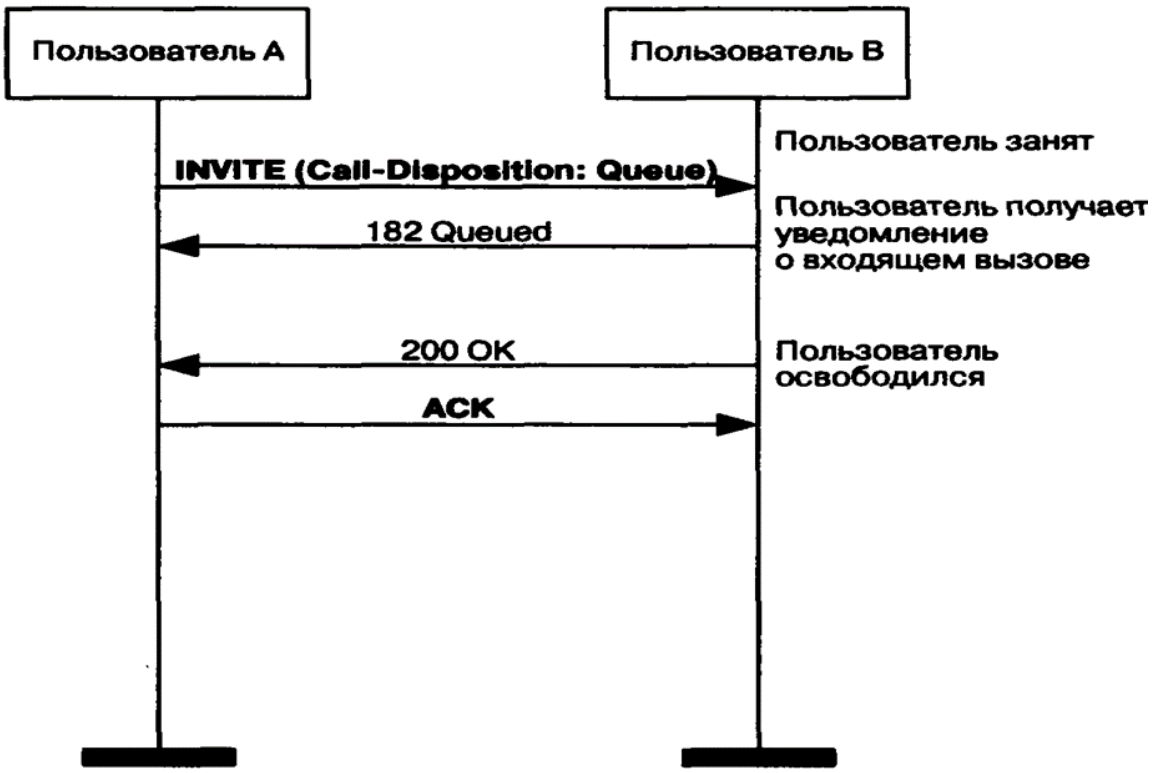


Рис. 8.16. Додаткова послуга «Повідомлення про виклик під час зв'язку»

Послуга реалізується за допомогою заголовка Call-Disposition, в якому міститься інструкція по обслуговуванню виклику.

Користувач передає запит INVITE із заголовком Call-Disposition: Queue, який інтерпретується таким чином: користувач хоче, аби виклик був поставлений в чергу, якщо користувач, що викликається, буде зайнятий. Сторона, що викликається, підтверджує виконання запиту відповіддю 182 Queued, який може передаватися неодноразово протягом періоду чекання.

Користувач, що викликається, отримує повідомлення про вхідний виклик, а коли він звільняється, цій стороні передається фінальна відповідь 200 ОК.

ЛІТЕРАТУРА

8.1. Alexander Neumann. Zopfli: Neue Kompressionsbibliothek von Google | heise Developer (нем.).

8.2. Alakuijala, Jyrki Data compression using Zopfli .

8.3. Dean Hume. Improved Compression Ratios Using Zopfli (2015).

8.4. Sharwood, Simon Google open sources very slow compression algorithm. The Register (2013).

8.5. Ilya Grigorik. Google Fonts recently switched to using new Zopfli compression algorithm. Google+

8.6. J.G. Cleary, and I.H. Witten, Data compression using adaptive coding and partial string matching (недоступная ссылка), IEEE Transactions on Communications, Vol. 32 (4), pp. 396–402, April 1984.

8.7. Moffat, Implementing the PPM data compression scheme, IEEE Transactions on Communications, Vol. 38 (11), pp. 1917–1921, November 1990.

8.8. J.G. Cleary, W.J. Teahan, and I.H. Witten, Unbounded length contexts for PPM, In J.A. Storer and M. Cohn, editors, Proceedings DCC-95, IEEE Computer Society Press, 1995.

8.9. C. Bloom, Solving the problems of context modeling.

8.10. W.J. Teahan, Probability estimation for PPM.

8.11. T. Schürmann and P. Grassberger, Entropy estimation of symbol sequences (недоступная ссылка), Chaos, Vol. 6, pp. 414–427, September 1996.

8.12. Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео Диалог-МИФИ, 2002 г., 384 с. ISBN 5-86404-170-X. 3000 экз.

8.13. G. Nigel N. Martin, Range encoding: An algorithm for removing redundancy from a digitized message, Video & Data Recording Conference, Southampton, UK, July 24-27, 1979.

8.14. «Source coding algorithms for fast data compression» Richard Clark

Pasco, Stanford, CA 1976

8.15. Bill Croft, John Gilmore. Bootstrap Protocol (BOOTP). IETF (September 1985).

8.16. J. Reynolds, J. Postel. Assigned Numbers IETF (October 1994).

8.17. W. Wimer. Clarifications and Extensions for the Bootstrap Protocol IETF (October 1993).

8.18. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципи, технологии, протоколы: Учебник для вузов. 3-е изд.– СПб.: Питер, 2006. – 958 с.

8.19. Hughes, L. Internet e-mail Protocols, Standards and Implementation. — Artech House Publishers, 1998. — ISBN 0-89006-939-5.

8.20. Hunt, C. sendmail Cookbook. — O’Reilly Media, 2003. — ISBN 0-596-00471-0.

8.21. Johnson, K. Internet Email Protocols: A Developer's Guide. — Addison-Wesley Professional, 2000. — ISBN 0-201-43288-9.

8.22. Loshin, P. Essential Email Standards: RFCs and Protocols Made Practical. — John Wiley & Sons, 1999. — ISBN 0-471-34597-0.

8.23. Rhoton, J. Programmer's Guide to Internet Mail: SMTP, POP, IMAP, and LDAP. — Elsevier, 1999. — ISBN 1-55558-212-5.

8.24. Wood, D. Programming Internet Mail. — O'Reilly, 1999. — ISBN 1-56592-479-7.