МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

КОМП'ЮТЕРНІ МЕРЕЖІ

ЛАБОРАТОРНІ РОБОТИ З КРЕДИТНОГО МОДУЛЯ «КОМП'ЮТЕРНІ МЕРЕЖІ 1. ОСНОВНІ ПРИНЦИПИ ПОБУДОВИ КОМП'ЮТЕРНИХ МЕРЕЖ»: виконання, оформлення та захист

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою
«Системне програмування і спеціалізовані комп'ютерні системи»
спеціальності 123 «Комп'ютерна інженерія»

Київ
КПІ ім. Ігоря Сікорського
2021

Посібник з виконання лабораторних робіт з кредитного модуля «Комп'ютерні мережі 1. Основні принципи побудови комп'ютерних мереж» дисципліни «Комп'ютерні мережі» [Електронний ресурс] : навч. посіб. для студ. спеціальності 123 «Комп'ютерна інженерія», освітньої програми «Системне програмування та спеціалізовані комп'ютерні системи» / КПІ ім. Ігоря Сікорського ; уклад.: М. М. Орлова, А. А. Крайносвіт, П. А. Сергієнко. – Електронні текстові дані (1 файл: 6,85 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 165 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 2 від 09.12.2021 р.)
за поданням Вченої ради факультету прикладної математики
(протокол № 3 від 25.10.2021р.)

Електронне мережеве навчальне видання

КОМП'ЮТЕРНІ МЕРЕЖІ

«ЛАБОРАТОРНІ РОБОТИ З КРЕДИТНОГО МОДУЛЯ «КОМП'ЮТЕРНІ МЕРЕЖІ 1. ОСНОВНІ ПРИНЦИПИ ПОБУДОВИ КОМП'ЮТЕРНИХ МЕРЕЖ»»: виконання, оформлення та захист

Укладачі: *Орлова Марія Миколаївна,* канд. техн. наук, доц.
Крайносвіт Аркадій Артемович
Сергієнко Павло Анатолійович

Відповідальний редактор *Тарасенко В.П.*, д-р. техн. наук, проф.

Рецензенти: *Кулаков Ю.О.,* д-р техн. наук, проф.

Навчальний посібник розроблено для ознайомлення студентів з вимогами, правилами виконання, оформлення та оцінювання курсового проєкту з кредитного модуля «Комп'ютерні мережі 1. Основні принципи побудови комп'ютерних мереж» дисципліни «Комп'ютерні мережі». Навчальне видання призначене для студентів спеціальності 123 «Комп`ютерна інженерія», освітньої програми «Системне програмування та спеціалізовані комп'ютерні системи» кафедри системного програмування і спеціалізованих комп'ютерних систем факультету прикладної математики КПІ ім. Ігоря Сікорського.

ЗМІСТ

ВСТУП

Даний посібник призначений для:

- визначення правил виконання та захисту лабораторних робіт з дисципліни «Комп'ютерні мережі» *для здобувачів ступеня бакалавра за освітньою програмою «Системне програмування та спеціалізовані комп'ютерні системи» спеціальності 123 «Комп'ютерна інженерія»* для студентів кафедри системного програмування і спеціалізованих комп'ютерних систем факультету прикладної математики КПІ імені Ігоря Сікорського;
- визначення правил і вимог до виконання лабораторних робіт;
- визначення правил і вимог до оформлення лабораторних робіт:
- правил оцінювання якості виконання лабораторних робіт та їх захисту з кредитного модуля «Комп'ютерні мережі 1. Основні принципи побудови комп'ютерних мереж» дисципліни «Комп'ютерні мережі».

Дисципліна «Комп'ютерні мережі» є базовою дисципліною циклу загальної підготовки бакалаврів за спеціальністю 123 «Комп'ютерна інженерія», яка ознайомлює студентів зі структурою, організацією та принципами функціонування комп'ютерних мереж (КМ). Одним з важливих видів індивідуальних завдань, що передбачені навчальною програмою дисципліни «Комп'ютерні мережі» та виконуються студентами при її вивченні, є лабораторні роботи (ЛР).

Самостійне виконання лабораторних робіт є обов'язковим при вивченні дисципліни «Комп'ютерні мережі».

Виконання лабораторних робіт сприяє розширенню та поглибленню отриманих теоретичних знань щодо організації та функціонування комп'ютерних мереж, а також обробки даних при введенні в мережу та їх передачі через комунікаційне середовище.

Посібник містить матеріали для виконання всіх лабораторних робіт кредитного модуля «Комп'ютерні мережі 1. Основні принципи побудови комп'ютерних мереж» дисципліни «Комп'ютерні мережі», що виконуються в першому семестрі, теоретичні відомості, які необхідні для виконання кожної лабораторної роботи, приклади та додаткові інформаційні матеріали, вимоги до виконання завдання, оформлення протоколу та захисту лабораторної роботи тощо.

1. МЕТА ТА ЗАВДАННЯ ЛАБОРАТОРНИХ РОБІТ

Метою циклу лабораторних робіт з дисципліни «Комп'ютерні мережі» є закріплення та поглиблення знань щодо принципів організації комп'ютерних мереж, принципів обробки даних, що передаються мережею, стеку протоколів мережі Інтернет, інкапсуляції тощо.

Кожна лабораторна робота призначена для вивчення конкретної теми та пов'язана з відповідними питаннями, що стосуються розгляду особливостей організації та функціонування окремих ресурсів комп'ютерної мережі та їх взаємодії. Крім того, необхідно ознайомитись та вивчити аналізатори мережевого трафіку, які надають можливість їх практичного використання в подальшій роботі.

В результаті виконання **лабораторних робіт** з кредитного модуля «Комп'ютерні мережі 1. Основні принципи побудови комп'ютерних мереж» студент повинен:

ЗНАТИ:

- особливості архітектури комп'ютерних мереж, організації взаємодії їх окремих модулів, функціонування окремих ресурсів КМ, процедур передачі повідомлень мережею тощо;
- загальну методику обробки та перетворення даних при їх передачі в мережу та просуванні через комунікаційне середовище;

УМІТИ:

- аналізувати вимоги до параметрів і особливостей передачі даних, аналізувати структури каналів передачі та комп'ютерної мережі в цілому;
- оцінювати особливості передачі даних з використанням різних способів та протоколів передачі, оцінювати локалізацію трафіку в IP-мережі;
- аналізувати і оцінювати результати виконаної роботи;

НАПРАЦЮВАТИ ДОСВІД:

- аналізу мережного (мережевого) трафіку за допомогою різних ресурсів (Wireshark, CISCO Packet Tracer та інших);
- перетворення DNS-імен в IP-адреси;
- роботи з сокетами різних типів.

2. ВИКОНАННЯ ТА ЗАХИСТ ЛАБОРАТОРНИХ РОБІТ

В кредитному модулі «Комп'ютерні мережі 1. Основні принципи побудови комп'ютерних мереж» передбачено виконання 8 лабораторних робіт, які повинні бути виконані та захищені у визначені терміни:

- лабораторні роботи 1-4 виконуються та захищаються студентами не пізніше 8-го тижня (включно) осіннього семестру (до першої атестації);
- лабораторні роботи 5-7 виконуються та захищаються студентами не пізніше 14-го тижня (включно) осіннього семестру (до другої атестації);
- лабораторна робота 8 виконується та захищається студентами не пізніше 16-го тижня (включно) осіннього семестру (до залікової сесії).

При недотриманні даного графіку здачі нараховуються штрафні бали, а саме «- 1 бал» за кожний тиждень затримки.

Не дозволяється одночасно захищати більше двох лабораторних робіт.

Для захисту лабораторної роботи обов'язково оформляється протокол, який повинен містити наступні елементи:

- назва роботи;
- мета роботи;
- стислі теоретичні відомості за темою лабораторної роботи;
- порядок виконання роботи та отримані результати;
- висновки по роботі (обов'язково);
- в разі необхідності до протоколу додаються скріншоти, таблиці результатів тощо.

Перед захистом оформлений протокол надсилається викладачу для перевірки та попереднього аналізу виконаної роботи. В процесі захисту лабораторної роботи студент обов'язково відповідає на поставлені викладачем запитання.

Лабораторна робота може виконуватись та захищатись бригадою, яка складається з двох студентів (не більше). В цьому випадку кожен зі студентів виконує свою частину лабораторної роботи та отримує запитання при захисті роботи.

В разі поглибленого опрацювання теми лабораторної робот, ретельної підготовки до захисту та ґрунтовних відповідей на поставлені викладачем запитання студент може отримати додаткові заохочувальні бали, але не більше «+ 2 бали» за ЛР.

Лабораторна робота 1
Стеки мережевих протоколів.
Аналізатор мережевого трафіку Wireshark

Мета роботи: засвоєння функцій модулів різних рівнів еталонної моделі OSI, процедури інкапсуляції та формування повідомлень для передачі в мережу; ознайомлення та вивчення аналізатора мережевого трафіку Wireshark.

План виконання лабораторної роботи

1. Ознайомитися та засвоїти теоретичні відомості про еталонну модель взаємодії відкритих систем OSI та стек мережевих протоколів TCP/IP.
2. Ознайомитися з можливостями аналізатора мережевого трафіку Wireshark.
3. За допомогою аналізатора Wireshark виконати захоплення та провести аналіз мережевих пакетів.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Еталонна модель взаємодії відкритих систем OSI

Еталонна модель взаємодії відкритих систем OSI (Open System Interconnect Reference Model,) – це універсальний стандарт, який описує взаємодію комп'ютерів через комп'ютерну мережу (КМ). За допомогою цієї моделі складна і багатогранна задача взаємодії віддалених робочих станцій КМ представляється сукупністю простіших підзадач, кожна з яких вирішується своїм модулем (програмним або апаратним ресурсом).

Модель складається з 7-ми рівнів, кожен з яких виконує строго визначені функції, і може взаємодіяти тільки зі своїми сусідами й виконувати відведені тільки йому функції (рис. 1.1). Зауважимо, що це є саме еталонна модель, з якою порівнюються стеки протоколів будь-яких сучасних комп'ютерних мереж, які мають на теперішній момент від 4 до 7 рівнів (модулів), реалізованих програмним або апаратним чином.

Прикладний рівень (Application layer)

Верхній прикладний рівень моделі забезпечує взаємодію мережі і користувача. Рівень дозволяє додаткам користувача, таким як обробник запитів до баз даних, доступ до файлів, пересилку повідомлень електронної пошти, доступ до мережевих служб. Також він відповідає за передачу службової інформації, надає додаткам інформацію про помилки і формує запити до рівня представлення.

Представницький рівень (Presentation layer)

Цей рівень відповідає за перетворення форматів даних і кодування/декодування даних. Запити програм, отримані із прикладного рівня, він перетворює у стандартний формат повідомлення для передачі мережею, а

отримані з мережі дані перетворює у формат, зрозумілий відповідному сервісу програм. На цьому рівні може здійснюватися стиснення/розпакування або кодування/декодування даних, а також перенаправлення запитів іншому мережевому ресурсу, якщо вони не можуть бути оброблені локально.

Рисунок 1.1 – Рівні моделі OSI

Сеансовий рівень (Session layer)

Цей рівень відповідає за підтримку сеансу зв'язку, дозволяючи застосуванням взаємодіяти між собою тривалий час. Рівень керує створенням/завершенням сеансу, обміном інформацією, синхронізацією завдань, визначенням права на передачу даних і підтримкою сеансу в періоди неактивності застосувань. Синхронізація передачі забезпечується розміщенням у потоці даних контрольних точок, починаючи з яких відновлюється процес при порушенні взаємодії.

Транспортний рівень (Transport layer)

Транспортний рівень призначений для доставки даних без помилок, втрат і дублювання в тій послідовності, у якій вони були передані. При цьому немає значення, які дані передаються, звідки й куди, тобто він визначає сам механізм передачі. Блоки даних повідомлення він розділяє на сегменти, розмір яких залежить від протоколу, що використовується, або налаштуваннями каналу. Крім того, при прийомі даних з мережі пакети об'єднуються в єдине повідомлення, яке і передається через порти на вищі рівні. Протоколи цього рівня призначені для взаємодії типу point-to-point.

Мережевий рівень (Network layer)

Мережевий рівень моделі OSI, призначений для визначення маршруту передачі даних через комунікаційне середовище мережі. Відповідає за трансляцію логічних адрес і імен у фізичні, визначення найкоротших маршрутів, комутацію і маршрутизацію пакетів, відстеження проблем і

перевантажень у мережі. На цьому рівні працює такий мережевий пристрій, як маршрутизатор.

Канальний рівень (Data Link layer)

Цей рівень призначений для забезпечення взаємодії мереж на фізичному рівні і контролю за помилками, які можуть виникнути. Отримані із фізичного рівня дані він упаковує в кадри даних, перевіряє на цілісність, якщо потрібно, виправляє помилки і відправляє на мережевий рівень. Канальний рівень може взаємодіяти з одним або декількома фізичними рівнями, контролюючи і керуючи цією взаємодією. Для локальних мереж цей рівень представлено 2 підрівнями - MAC (Media Access Control), який регулює доступ до розподіленого фізичного середовища, та LLC (Logical Link Control), який забезпечує управління передачею між модулями. На цьому рівні працюють комутатори, мости і мережеві адаптери.

MAC-підрівень забезпечує коректне спільне використання загального середовища, надаючи його в розпорядження тієї або іншої станції мережі. Також він додає адресну інформацію до фрейму, позначає початок і кінець фрейму.

Рівень LLC відповідає за достовірну передачу кадрів даних між вузлами, а також реалізує функції інтерфейсу з мережевим рівнем. Також він здійснює ідентифікування протоколу мережевого рівня.

У програмуванні цей рівень представляє драйвер мережевої карти, в операційних системах є програмний інтерфейс взаємодії канального і мережевого рівнів між собою, це не новий рівень, а просто реалізація моделі для конкретної ОС. Приклади таких інтерфейсів: NDIS, ODI.

Фізичний рівень (Physical layer)

Найнижчий рівень моделі призначений безпосередньо для передачі потоку даних. Здійснює передачу електричних або оптичних сигналів у кабель і відповідно їх приймання і перетворення в біти даних відповідно до методів кодування цифрових сигналів, тобто здійснює інтерфейс між мережевим носієм і мережевим пристроєм. На цьому рівні працюють концентратори і повторювачі (ретранслятори) сигналу. Фізичний рівень визначає електричні, механічні, процедурні і функціональні специфікації для середовища передачі даних, у тому числі роз'єми, розпаювання і призначення контактів, рівні напруги, синхронізацію зміни напруги, кодування сигналу.

Цей рівень приймає кадр даних від канального рівня, представляє його в тій послідовності сигналів, які потім передаються у лінію зв'язку. Передача кадру даних через лінію зв'язку вимагає від фізичного рівня визначення таких елементів: тип середовища передачі (проводовий або безпроводовий, мідний кабель або оптичне волокно) і відповідних конекторів; як повинні бути представлені біти даних у середовищі передачі; як кодувати дані; якими повинні бути схеми приймача і передавача.

У сучасних мережах використовуються три основних типи середовища передачі: мідний кабель (copper), оптичне волокно (fiber) та бездротове середовище передачі (wireless). Тип сигналу, за допомогою якого здійснюється

передача даних, залежить від типу середовища передачі. Для мідного кабелю сигнали, що представляють біти даних, є електричними імпульсами, для оптичного волокна - імпульсами світла. У випадку використання безпроводових з'єднань сигнали є радіохвилями (електромагнітними хвилями).

Коли пристрій, що працює на фізичному рівні, кодує біти кадру в сигнали для конкретного середовища передачі, він має розрізняти кадри. Тобто позначати, де закінчується один кадр і починається інший. Інакше мережеві пристрої, що здійснюють прийом сигналів, не зможуть визначити, коли кадр буде отриманий повністю. Відомо, що початок і кінець кадру позначається на канальному рівні, але в багатьох технологіях фізичний рівень також може додати спеціальні сигнали, що використовуються тільки для позначення початку і кінця кадру даних.

Технології фізичного рівня визначаються стандартами, що розробляються такими організаціями: The International Organization for Standardization (ISO), The Institute of Electrical and Electronics Engineers (IEEE), The American National Standards Institute (ANSI), The International Telecommunication Union (ITU), The Electronics Industry Alliance/Telecommunications Industry Association (EIA/TIA), тощо. Дані стандарти охоплюють чотири області, що належать фізичному рівню: фізичні та електричні властивості середовища передачі, механічні властивості (матеріали, розміри, розпайка контактів конекторів), кодування (представлення бітів сигналами), визначення сигналів для керування інформацією. Всі компоненти апаратного забезпечення, такі, як мережеві карти (Network interface card, NIC), інтерфейси і конектори, матеріали кабелів та їх конструкція визначаються стандартами фізичного рівня. Слід зазначити, що функції фізичного рівня вбудовані у мережеве обладнання (hardware).

Основними функціями фізичного рівня є: фізичні компоненти, кодування даних, передача даних. Фізичні компоненти — електронне обладнання, середовище передачі і конектори, через які передаються сигнали, що представляють біти даних.

Кожний з описаних рівнів визначається сервісом, який він надає розташованому вище рівню, і протоколом – набором правил і форматів даних для взаємодії між собою об’єктів одного рівня, що працюють на різних комп’ютерах (рис. 1.2).

Дані, які обробляються на кожному рівні, мають різний формат і називаються протокольними блоками даних PDU (Protocol Data Unit).

При передачі кожний рівень приєднує до PDU, що надходить з верхнього рівня, службову інформацію у вигляді заголовку певної структури. Цей процес називається інкапсуляцією. Процес продовжується до досягнення самого нижнього рівня (фізичного рівня або рівня доступу до мережі), з якого дані передаються на мережевий пристрій. В пристрої одержувача відбувається зворотній процес, деінкапсуляція даних на кожному рівні. Потім додаток, нарешті, використовує дані. Процес продовжується, поки всі дані не будуть передані і отримані.

Вся складна процедура мережевої взаємодії може бути поділена на деяку кількість простіших процедур, що послідовно виконуються об'єктами певних рівнів моделі. Модель побудована таким чином, що об'єкти одного рівня двох взаємодіючих комп'ютерів взаємодіють безпосередньо один з одним за допомогою відповідних протоколів, не знаючи, які рівні розташовані нижче і які функції вони виконують. Задача кожного з рівнів полягає в наданні через стандартизований інтерфейс певного сервісу модулю розташованого вище рівня, скориставшись, в разі необхідності, сервісом, який надає даному об'єкту нижче розташований рівень.

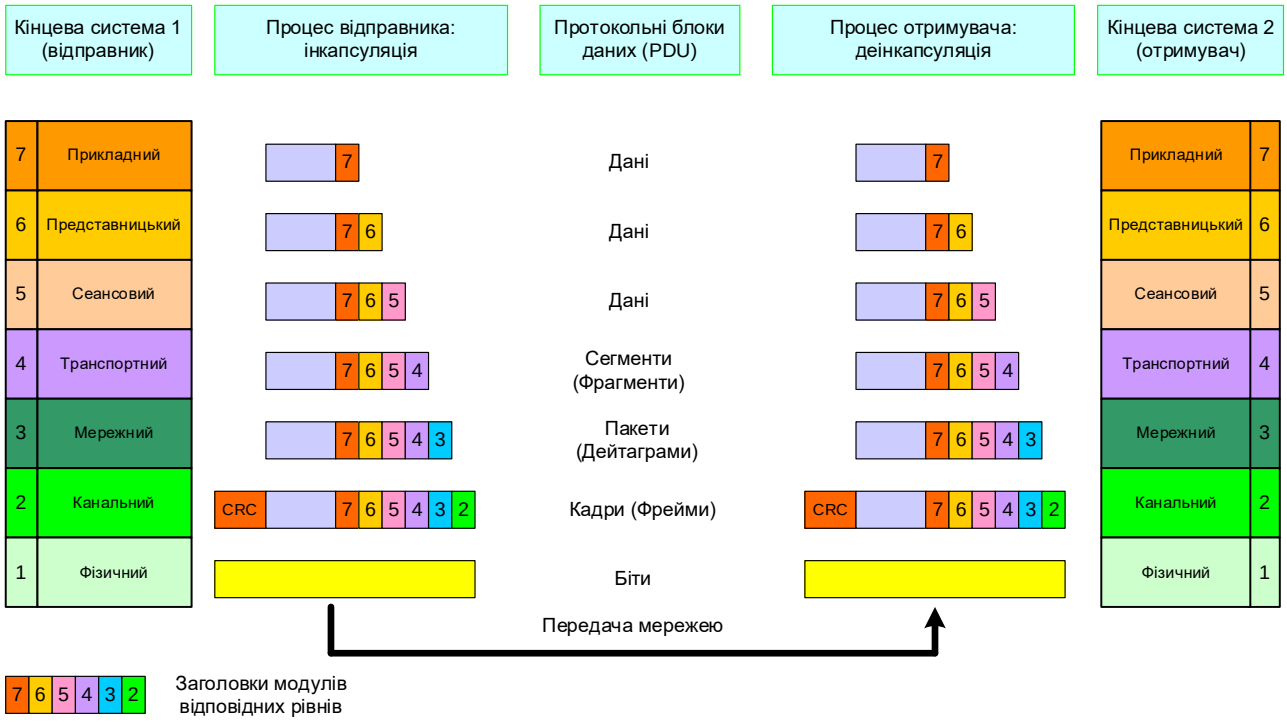


Рисунок 1.2 – Потоки даних від верхніх рівнів до нижніх. Інкапсуляція і деінкапсуляція

Наприклад, деякий процес відправляє дані через мережу процесу, що знаходиться на іншому комп'ютері. Через стандартизований інтерфейс процес-відправник передає дані нижньому рівню, який надає процесу сервіс для пересилки даних, а процес-отримувач через такий же стандартизований інтерфейс отримує ці дані від нижнього рівня. При цьому ні один із процесів не знає, як саме здійснює передачу даних протокол нижнього рівня, скільки ще рівнів знаходиться під ним, яке фізичне середовище передачі даних і яким шляхом дані передаються.

Ці процеси, з іншого боку, можуть знаходитися не на самому верхньому рівні моделі. Припустимо, що вони через стандартний інтерфейс взаємодіють із застосуваннями вище розташованого рівня і їхня задача (сервіс, що надається)– перетворення даних, а саме фрагментація і збирання великих блоків даних, які вище розташовані застосування відправляють один одному. При цьому сутність

цих даних і їх інтерпретація для процесів, що розглядаються, абсолютно не важливі.

Можлива також взаємозаміна об’єктів одного рівня (наприклад, при зміні способу реалізації сервісу) таким чином, що об’єкт вище розташованого рівня не помітить заміни.

Об'єкти, які виконують функції рівнів, можуть бути реалізовані в програмному, програмно-апаратному або апаратному вигляді. Як правило, чим нижче рівень, тим більша частка апаратної частини в його реалізації.

Модулі усіх рівнів, що функціонують в одній технічній системі, взаємодіють за допомогою інтерфейсів, а однойменні модулі, що функціонують в різних технічних системах, взаємодіють за допомогою протоколів(рис. 1.3).

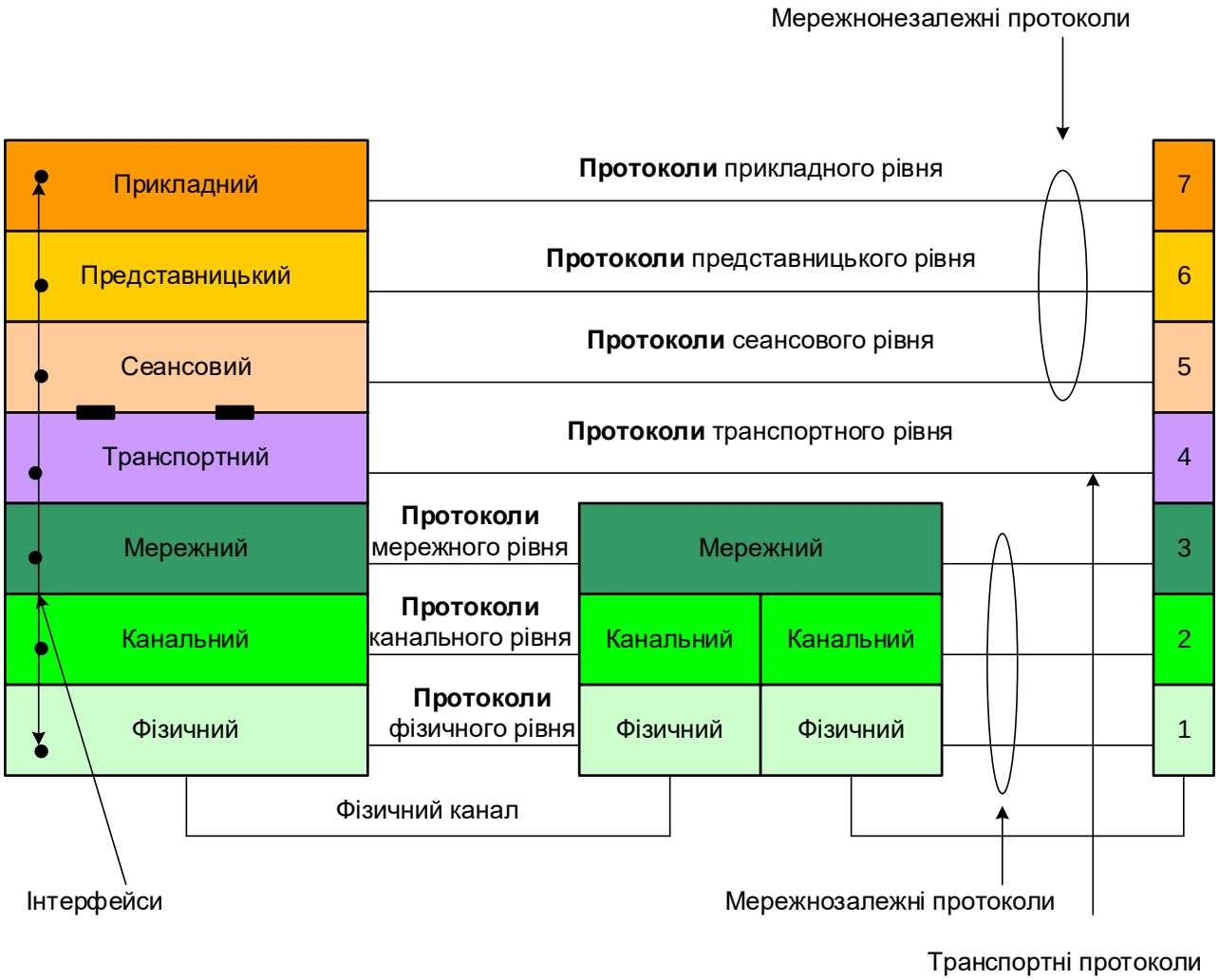


Рисунок 1.3 – Інтерфейси і протоколи

1.2. Інкапсуляція і обробка пакетів

При передачі повідомлення від прикладного рівня до фізичного для введення в мережу модуль кожного рівня додає до блоку даних, отриманого з сусіднього, верхнього рівня, свою службову інформацію у вигляді відповідного

заголовку і, в разі необхідності, кінцевика (CRC), який додається в кінець блоку. Ця операція називається інкапсуляцією даних і відбувається на модулях кожного рівня стеку протоколів. Службова інформація призначається для об’єкту однойменного рівня на віддаленій робочій станції,; її формат і інтерпретація визначаються протоколом даного рівня.

Зрозуміло, що блоки, які надходять з сусіднього верхнього рівня, можуть містити службову інформацію більш високих рівнів, тобто вже бути інкапсульованими.

Процедура інкапсуляції в комп’ютерних мережах — це метод побудови модульних мережевих протоколів, коли логічно незалежні функції мережі абстрагуються від нижче розташованих механізмів шляхом включення цих механізмів у вище розташовані об’єкти.

Загальна процедура інкапсуляції представлена на рис. 1.4.

В ряді випадків, блок даних може не передаватись на самий верхній прикладний рівень, оскільки даний модуль є не модулем кінцевої обробки, а тільки проміжним (комунікаційним вузлом на маршруті між відправником і отримувачем. У цьому випадку протокол відповідного рівня при аналізі службової інформації виявить, що блок даних на цьому рівні адресований не йому (хоча з точки зору нижніх рівнів він був адресований саме цьому комп’ютеру). Тоді протокол виконає необхідні дії для перенаправлення пакету до місця призначення або поверне відправнику з повідомленням про помилку, але в будь-якому випадку він не буде передавати дані на верхній рівень.

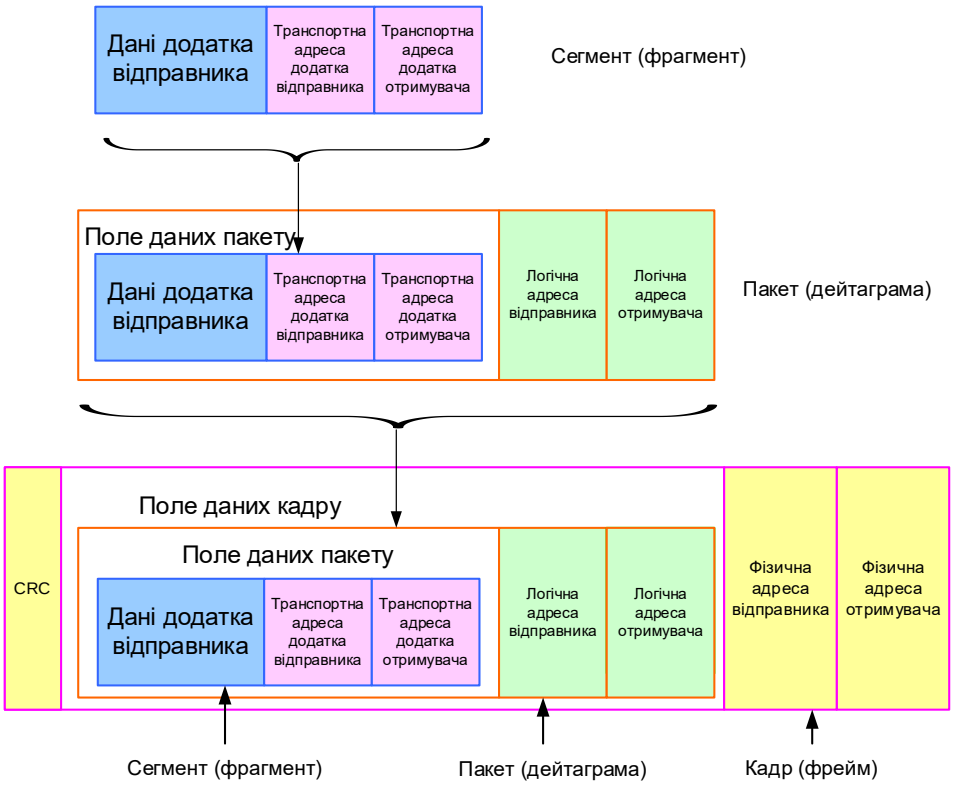


Рисунок 1.4 – Загальна процедура інкапсуляції

1.3. Стек протоколів TCP/IP

На відміну від еталонної моделі OSI, модель TCP/IP або модель DOD (Department of Defense) значною мірою більше орієнтується на забезпечення мережевої взаємодії, ніж на жорстке розділення функціональних рівнів.

Реалізація стеку протоколів TCP/IP фірми Microsoft відповідає чотирирівневій моделі замість семирівневої моделі, як показано на рисунку 1.5. При цьому прикладний рівень, який і реалізує відповідний прикладний сервіс, виконує сукупність функцій, а саме: формування повідомлення, представлення його у необхідному форматі, активізацію або створення необхідних портів процесів та підтримку їх (в разі необхідності) в активному стані протягом всього сеансу зв'язку. Кожен модуль моделі TCP/IP виконує більшу кількість функцій порівняно з моделлю OSI, внаслідок чого і зменшується кількість рівнів до чотирьох.

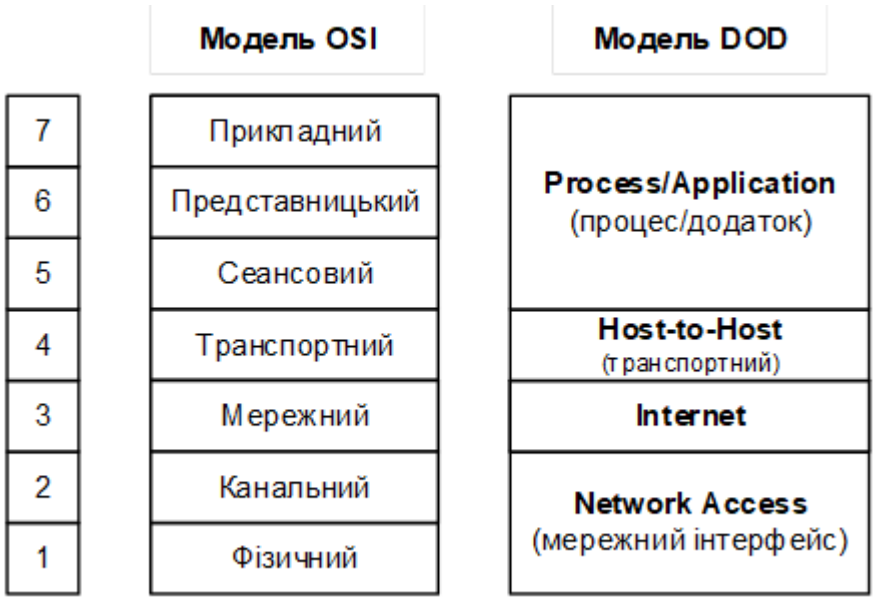


Рисунок 1.5 – Стек протоколів TCP/IP

Особливості TCP/IP:

- відкриті стандарти протоколів, що розробляються незалежно від програмного та апаратного забезпечення;
- незалежність від фізичного середовища передачі;
- унікальна система адресації;
- стандартизовані протоколи високого рівня для реалізації сервісів користувача.

Терміни, що використовуються для визначення блоків даних, які передаються, різні при використанні різних протоколів транспортного рівня - TCP і UDP (рис. 1.6).

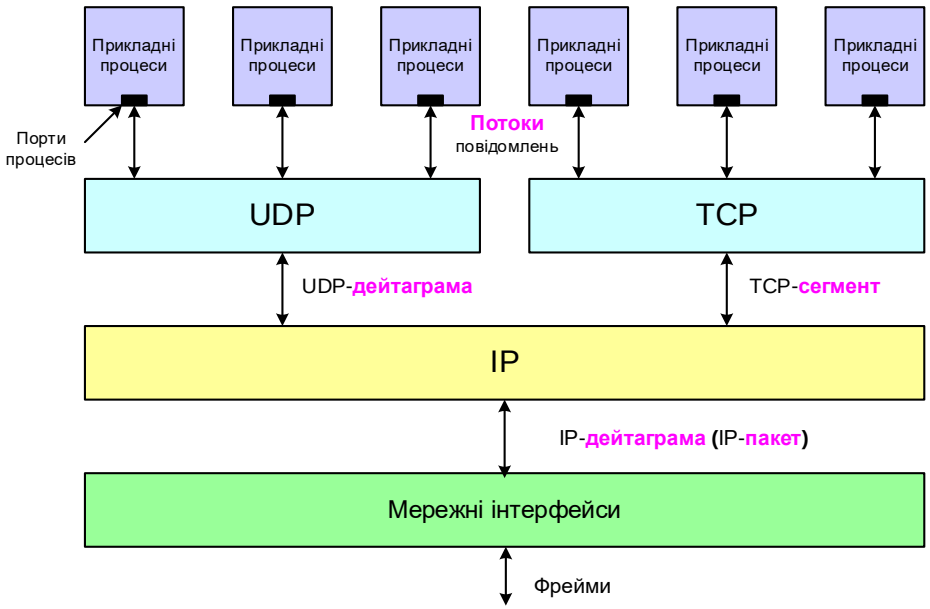


Рисунок 1.6 – Назви блоків даних, що обробляються протоколами різних рівнів

Обробка повідомлення, сформованого прикладним сервісом, при передачі в канал через стек протоколів TCP/IP принципово не відрізняється від такої ж обробки в еталонній моделі OSI. Особливості передачі повідомлення стосуються тільки протоколів, які оброблюють прийняту з більш високого рівня блоку даних та формату службової інформації (заголовку), з якою працює відповідний ресурс.

В мережі Інтернет використовуються три типи адрес, кожна з яких застосовуються відповідними модулями стеку протоколів. Кожна технічна система в мережі TCP/IP визначається адресами (ідентифікаторами) трьох типів, які використовуються модулями різних рівнів.

- **Символьний ідентифікатор-ім'я (символьна адреса)**, що призначається адміністратором і складається з декількох частин: ідентифікатора станції, імені організації, ідентифікатора домену. Це ім'я, яке також називають ***доменним*** іменем або ***DNS-іменем***, використовується на прикладному рівні, наприклад, у протоколах FTP, telnet та інших. Прикладами символьних адрес можуть служити microsoft.com, g.com.ua тощо.
- **Логічна адреса - IP-адреса** призначається адміністратором в момент підключення до мережі та конфігурування комп'ютерів і маршрутизаторів. IP-адреса характеризує не окремий комп'ютер або маршрутизатор, а одне мережеве з’єднання. Логічна адреса є адресою **програмного** забезпечення і використовується для передачі потоку повідомлень мережею.
- **Фізична (локальна) адреса** вузла залежить від технології, за допомогою якої побудована окрема мережа чи її сегмент, в яку входить даний вузол. Для вузлів, що входять у локальні мережі, - це **MAC-адреса** (Medium

Access Control або Media Access Control) мережного адаптера або порту маршрутизатора або іншого технічного вузла, наприклад, 11:A0:17:3D:BC:01. Фізична адреса є адресою **апаратного** забезпечення конкретного вузла, яка використовується при прийомі даних (або їх передачі) з каналу передачі в конкретну технічну систему (або видачі підготовлених даних з вихідного інтерфейсу системи в канал).

Загальна структура інкапсуляції, представлена на рис. 1.4 для стеку протоколів TCP/IP, відрізняється тільки форматами службової частини (заголовком) блоків даних кожного рівня, які залежать від типу протоколу, що використовується в конкретному випадку.

Наприклад, в якості транспортної адреси відправника та отримувача використовуються порти процесів (з додатковими параметрами передачі в разі обробки за допомогою протоколу TCP або без таких параметрів при використанні протоколу UDP). На мережевому рівні при формуванні пакету (дейтаграми) в якості логічної адреси використовується IP-адреса відправника та отримувача (32-бітної в разі застосування протоколу IPv4 або 128-бітної в разі застосування протоколу IPv6). На рівні канальних інтерфейсів в якості фізичної адреси кадру (фрейму) зазвичай використовується 48-бітна MAC-адреса, що застосовується в багатьох протоколах Ethernet (стандарти IEEE 802.2 та IEEE 802.3), Token Ring, FDDI, WI-FI (стандарт IEEE 802.11) та інших, структура якої наведена на рис. 1.7. Унікальний 24-бітний ідентифікатор OUI (Organizationally Unique Identifier) надається реєстраційним підрозділом IEEE (Registration Authority Committee), а 24-бітна адреса NIC визначає ідентифікатор даного модуля у виробника.

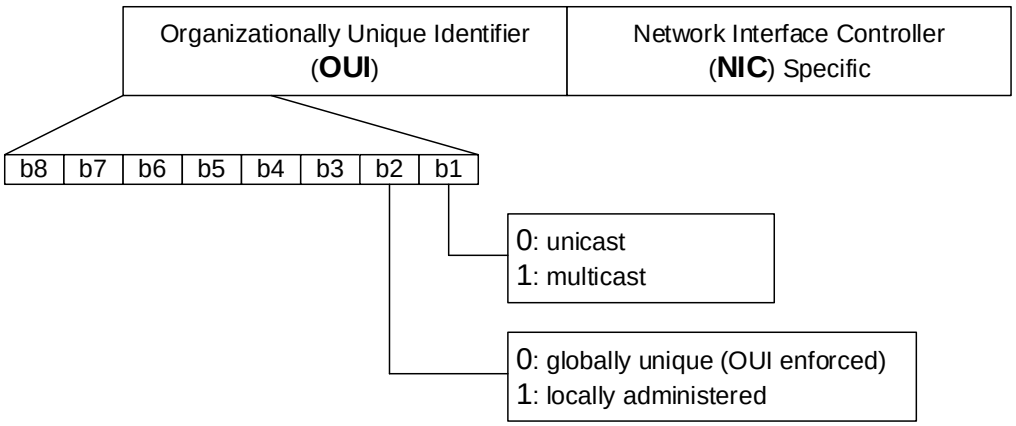


Рисунок 1.7 – Структура MAC-адреси

На сьогодні більшість мережевих протоколів канального рівня використовують один з трьох просторів MAC-адрес, які управляються IEEE: MAC-48, EUI-48 та EUI-64. Розширений ідентифікатор EUI-48 (Extended Unique Identifier) використовується для інших типів апаратного та програмного забезпечення (наприклад, мережевих протоколів). Інститут IEEE вважає термін

MAC-48 застарілим і розглядається як окремий випадок використання ідентифікатора EUI-48 для стандартів IEEE 802.x та інших.

В подальшому виробники та інші організації повинні використовувати визначення EUI-48. Ідентифікатори MAC-48 і EUI-48 ідентичні при самостійному використанні, але є деякі особливості при їх інкапсуляції в EUI-64.

Розширений ідентифікатор EUI-64 використовується в мережах FireWire, а також в протоколі IPv6 в якості молодших 64 біт в мережевій адресі вузла.

Ідентифікатор інтерфейсу в форматі EUI-64 складається з трьох частин:

- 24-бітний OUI на основі MAC-адреси клієнта, в якому сьомий біт є зворотним, тобто якщо 7-й біт має значення 0, він стає 1, і навпаки;
- в середину вставляється 16-бітне значення:
  - FFFE (в шістнадцятковій системі числення) для OUI-48;
  - FFFF для MAC-48;
- 24-бітний ідентифікатор пристрою на основі MAC-адреси клієнта.

2. РОБОТА З АНАЛІЗАТОРОМ ТРАФІКУ Wireshark

При виконанні системного адміністрування комп’ютерних мереж доволі часто доводиться мати справу зі складними ситуаціями, в яких не допомагають ані інструменти збору статистики (наприклад, netstat), ані стандартні утиліти на основі протоколу ICMP (ping, traceroute тощо). В таких випадках часто використовуються спеціалізовані діагностичні утиліти, які дозволяють прослуховувати мережевий трафік і аналізувати його на рівні блоків передачі окремих протоколів. Вони називаються аналізаторами трафіку або сніферами) і дозволяють, по-перше, локалізувати проблеми мережі та точніше їх діагностувати, а по-друге, виявляти в мережі не тільки різні типи трафіків, а й шкідливе програмне забезпечення.

На сьогодні існує значна кількість систем моделювання роботи комп’ютерних мереж і аналізаторів мережевого трафіку. В даній роботі необхідно ознайомитись з особливостями роботи аналізатора мережевого трафіку Wireshark.

2.1. Аналізатор мережевого трафіку Wireshark

Одним з найбільш поширених і популярних аналізаторів трафіку є Wireshark. Існують версії для різних операційних систем: Linux, Windows, MacOS, FreeBSD, Solaris. Wireshark також використовує бібліотеку libpcap для збирання пакетів. Wireshark має зручний графічний інтерфейс.

Для того щоб перехоплювати протокольні одиниці інформації (PDUs), які передаються мережею, потрібно мати фізичне підключення до відповідної мережі, а також запущену копію Wireshark на комп’ютері. Початкове вікно програми виглядає так, як показано на рисунку 1.8.

Щоб розпочати захоплення пакетів потрібно, при необхідності, ввести фільтр захоплення в поле «Enter a capture filter» та вибрати інтерфейс. У вікні, що відкриється, можна спостерігати за процесом захоплення трафіку на вибраному інтерфейсі (рис. 1.9).

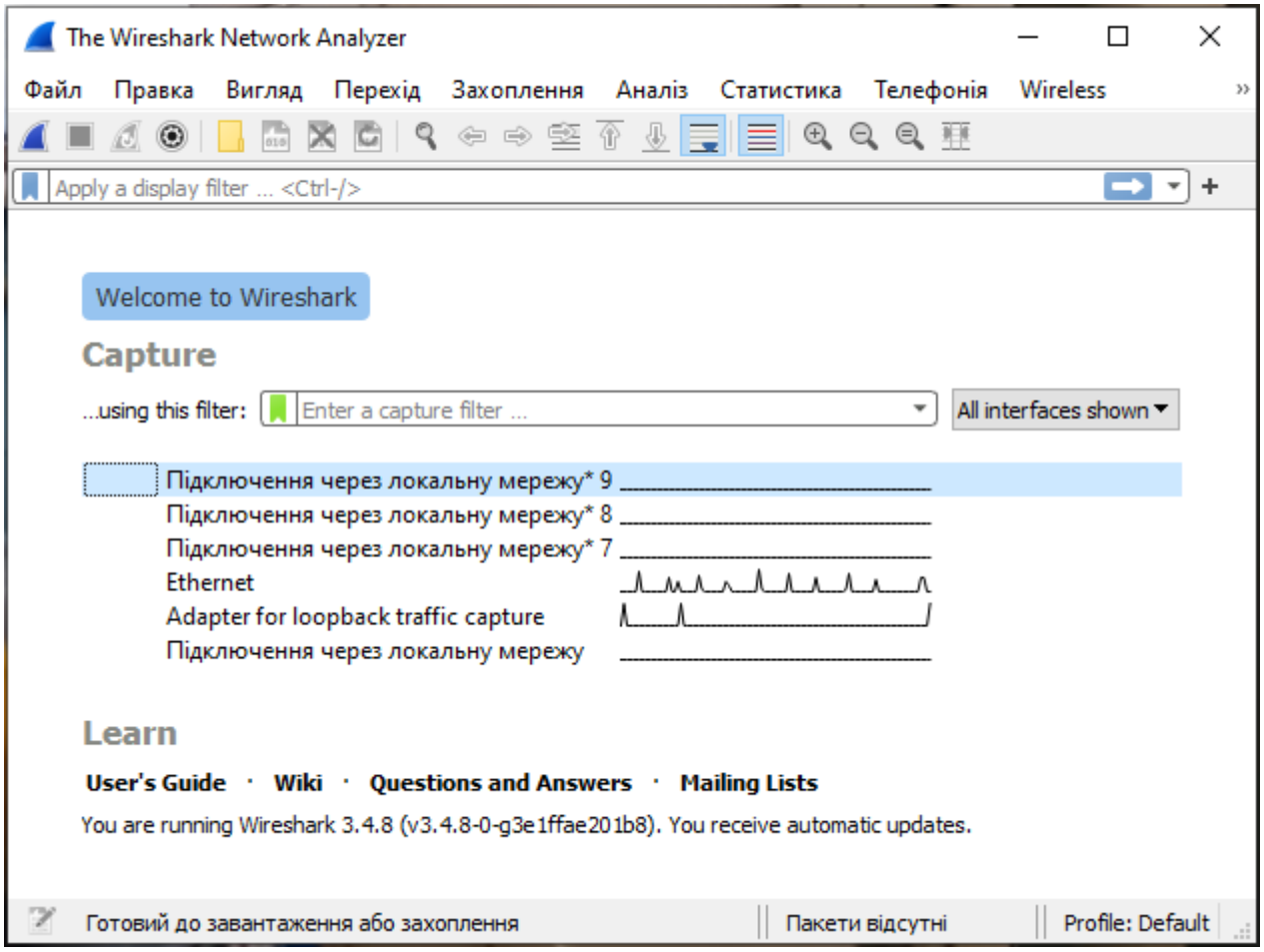


Рисунок 1.8 – Початкове вікно програми Wireshark

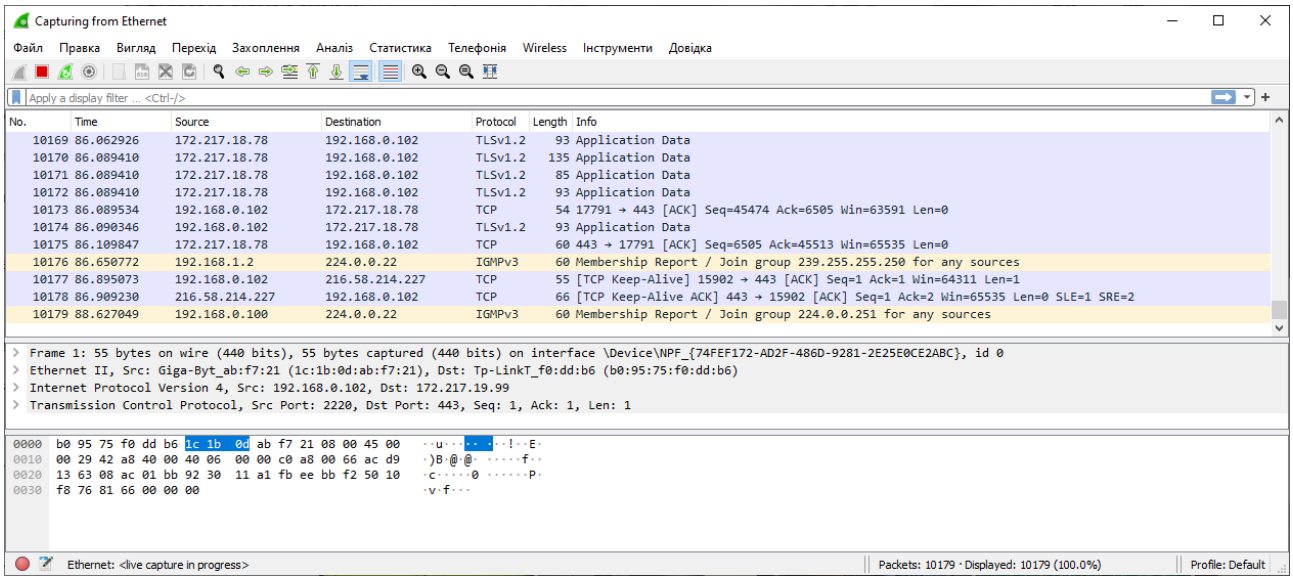


Рисунок 1.9 – Головне вікно програми Wireshark

Потрібно перевірити такі параметри:

- чи працює мережевий адаптер у, так званому, promiscuous mode;. якщо даний режим не обрано, то програма зможе захоплювати лише ті пакети, які були адресовані комп'ютеру, на якому вона встановлена (рис. 1.10);
- чи ввімкнено Resolve MAC addresses; дана опція дозволяє Wireshark транслювати знайдені мережеві адреси в імена (рис. 1.11).

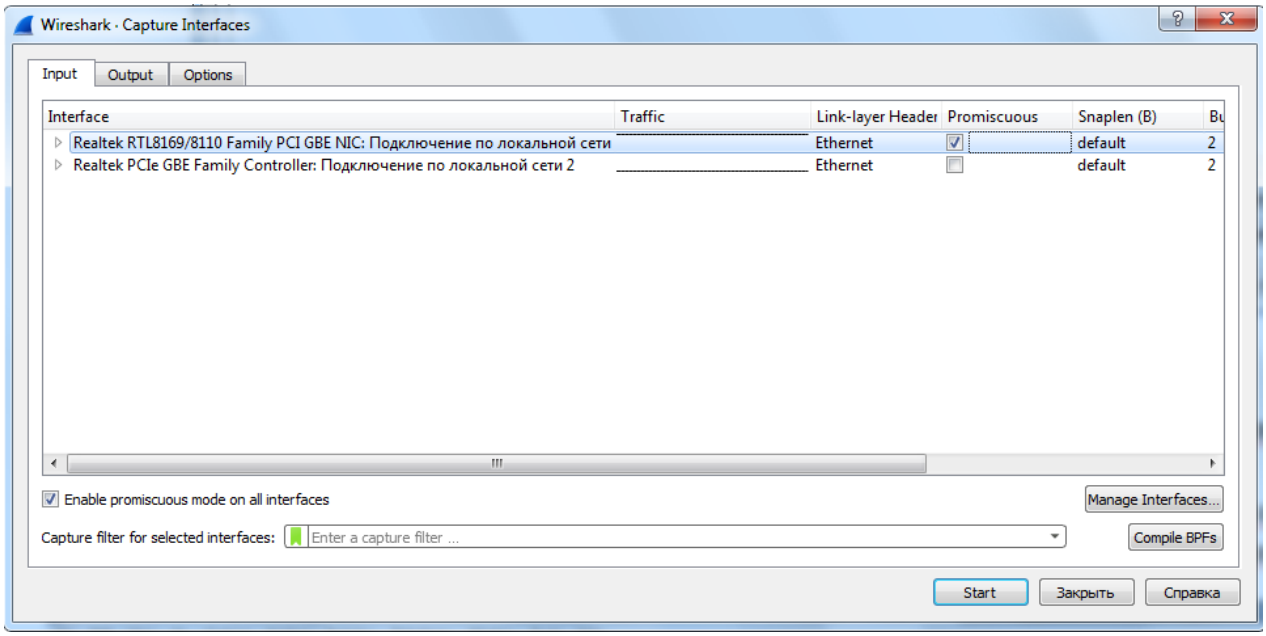


Рисунок 1.10 – Вікно налаштування параметрів захоплення трафіку

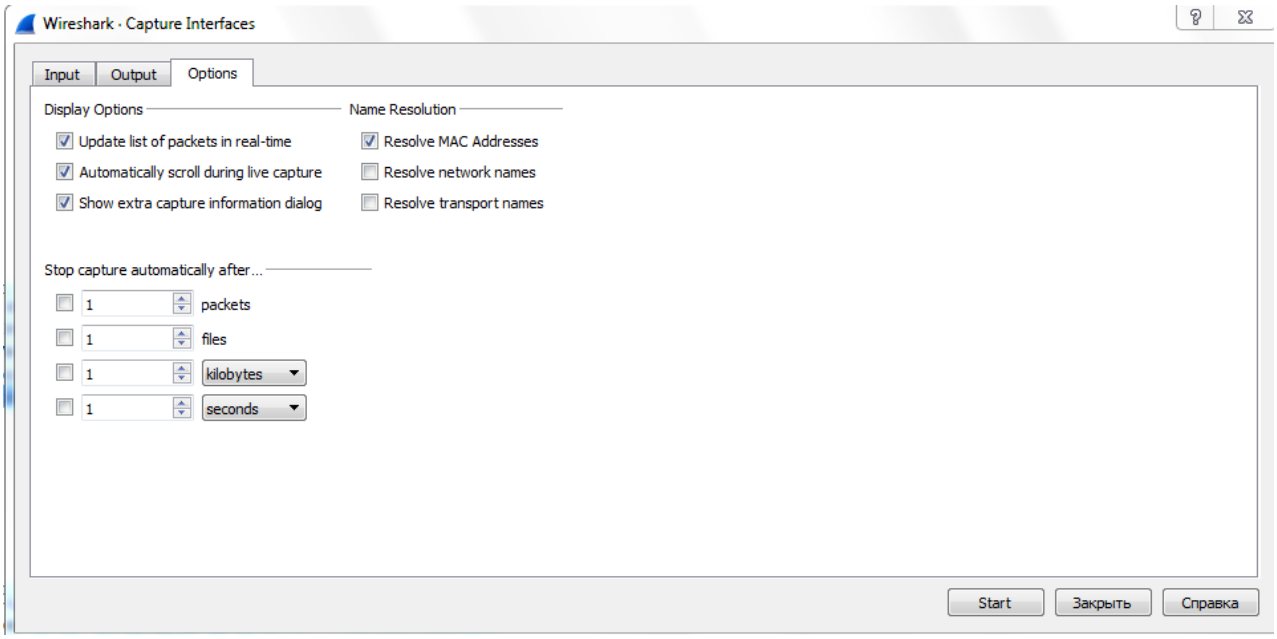


Рисунок 1.11 – Вікно налаштування параметрів перетворення імен

Після того, як налаштування завершені, потрібно натиснути кнопку Start, в результаті чого розпочнеться процес перехоплення пакетів.

Для того щоб перервати процес, потрібно натиснути на кнопку Stop. В результаті захоплення пакетів буде припинено (рис. 1.12). У даному вікні зображено результат виконання команди ping. Застосувати фільтр захоплення можна і після запуску програми. Для цього потрібно зупинити захоплення кнопкою Stop, увійти в меню Capture, потім в меню Options і у вікні, що відкриється, ввести необхідний фільтр.

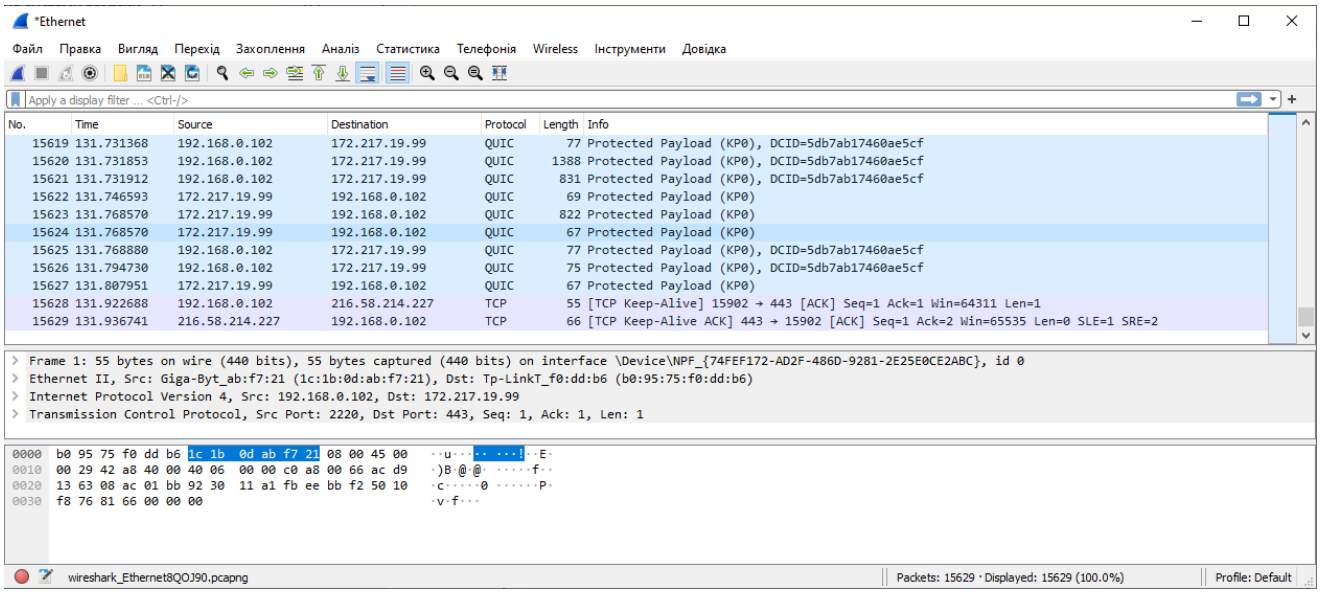


Рисунок 1.12 – Головне вікно програми з результатами захоплення мережевого трафіку

Головне вікно програми складається із трьох частин:

– поле з переліком пакетів, що містить стислу інформацію про кожний захоплений пакет; на цій панелі можна обрати потрібний пакет і його вміст відобразиться у двох інших панелях;
– поле зі структурою пакетів, яке дозволяє вивчити вміст пакету більш детально (інформація про протокол, службові поля пакету);
– поле з байтовою/символьною структурою пакетів показує фактичний вміст обраного пакету у шістнадцятковому/символьному коді.

У Wireshark використовується велика кількість різних фільтрів. Вони поділяються на два види:

– фільтри перехоплення трафіку (capture filters);
– фільтри відображення (display filters).

Перехоплення дозволяє економити оперативну пам’ять і місце на жорсткому диску. Фільтр − це вираз із групи примітивів, об’єднаних, при необхідності, логічними функціями (and, or, not). Записується цей вираз в полі «Capture Filter» діалогового вікна «Capture options». Фільтри, які

використовуються часто, можна зберігати в профілі для повторного використання.

На рисунку 1.13 наведені скріншоти екрану, отримані при роботі програми Wireshark, що ілюструють взаємне розташування Ethernet-, IP- та TCP- заголовків в кадрі.

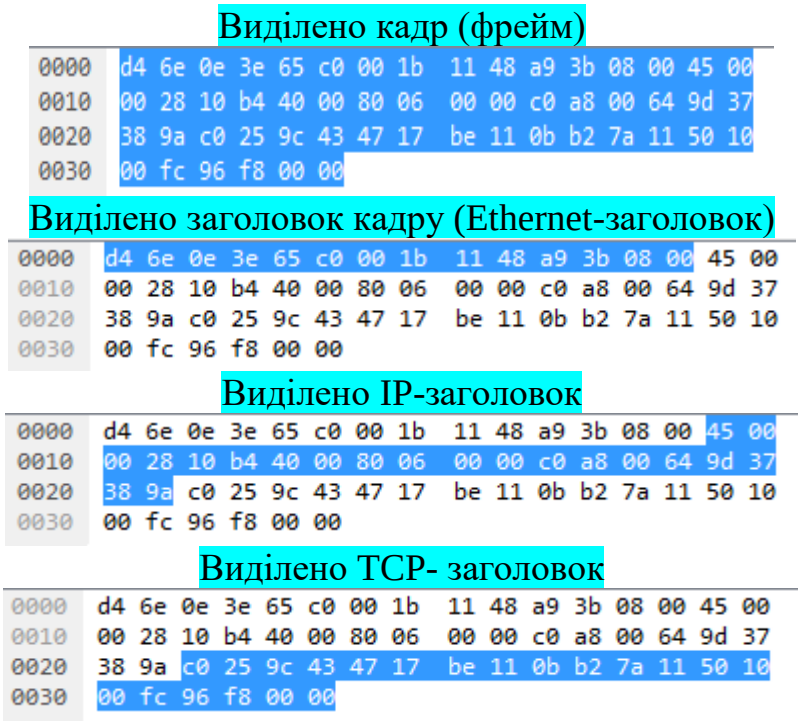


Рисунок 1.13 – Скріншоти екрану, отримані при роботі програми Wireshark

Фільтри відображення працюють з вже перехопленим трафіком. Вписати фільтр відображення можна прямо у відповідне поле (працює випадаючий список-підказка) головного вікна програми після кнопки «Filter» (цією кнопкою можна викликати профіль для фільтрів, які часто використовуються). При натисканні кнопки «Expression…» відкривається конструктор фільтрів.

Захоплені дані можна зберегти у файл. З цими даними можна працювати пізніше. Для цього потрібно натиснути «Файл», потім «Відкрити» і вибрати збережений файл.

Щоб добре розуміти значення фільтрів і що саме вони показують, необхідно добре розуміти роботу мережі.

Фільтри можуть складатись із ключових слів та скорочень, десяткових та шістнадцяткових чисел. Наприклад, можна відобразити тільки ті TCP пакети, які містять рядок **kpi** , якщо використовувати наступний фільтр:

tcp contains kpi

Тут **tcp** застосовується для вибору протоколу, **contains** - ключове слово, яке означає, що знайдені пакети міститимуть шуканий вміст.

У таблиці 1.1 перелічені оператори фільтрів.

Логічні оператори, приклади яких наведені в таблиці 1.2, дозволяють створювати детальні фільтри з використанням відразу декількох умов. Рекомендується додатково використовувати дужки для отримання тих значень, які необхідні.

Таблиця 1.1 - Логічні оператори фільтрів Wireshark

Англомовний синтаксис	С-подібний синтаксис	Опис оператора	Зразок застосування
eq	==	Дорівнює	ip.src==10.0.0.5
ne	!=	Не дорівнює	ip.src!=10.0.0.5
gt	>	Більше ніж	frame.len > 10
lt	<	Менше ніж	frame.len < 128
ge	>=	Більше чи дорівнює	frame.len ge 0x100
le	<=	Менше, чи дорівнює	frame.len <= 0x20
contains		Протокол, поле, або зріз(slice) включає значення	sip.To contains "a1762"
matches	~	Протокол або текстове поле збігається з Perl-сумісним регулярним виразом	http.host matches "acme\\.(org|com|net)"
bitwise_and	&	Симетричне І	tcp.flags & 0x02

Таблиця 1.2 – Приклади логічних операторів

Оператор	Опис
and/&&	Логічне І, дані виводяться, якщо вони відповідають обом частинам фільтру. Наприклад, фільтр ip.src==192.168.1.1 and tcp покаже тільки пакети, які надходять від 192.168.1.1 і які пов’язані з протоколом TCP. Будуть показані тільки дані, що задовольняють обом умовам.
or/||	Логічне АБО, достатньо щоб тільки одна умова була істиною; якщо обидві умови є істиною, то це теж підходить. Наприклад, фільтр tcp.port==80 or tcp.port==8080 покаже TCP пакети, які пов’язані з портом 80 або 8080.
not/!	Логічне Ні використовується, коли потрібно виключити деякі пакети. Тобто будуть показані всі пакети, крім тих, які задовольняють умові, що слідує після Ні. Наприклад, фільтр !dns покаже всі пакети, крім DNS.

Приклади логічних операторів:
показати HTTP або DNS трафік:

http or dns

Показати будь-який трафік, крім ARP, ICMP и DNS:

!(arp or icmp or dns)

Слід зазначити, що в аналізаторі Wireshark існує значна кількість фільтрів, які можуть використовуватись для фільтрації трафіку конкретного протоколу. Основні типи цих фільтрів наведено у Додатку А.

Завдання

1. При виконанні роботи використовується програмне забезпечення для аналізу протоколів комп'ютерних мереж Wireshark. Запустити відповідну програму.
2. Вибрати інтерфейс для захоплення трафіку (меню Capture/Interface) та активізувати режим захоплення.
3. Скопіювати через мережу файл розміром кілька десятків Мбайт.
4. Завершити захоплення трафіку та перейти до режиму аналізу.

Контрольні запитання

1. Поясніть поняття еталонної моделі взаємодії відкритих систем OSI?
2. Яке призначення еталонної моделі взаємодії відкритих систем?
3. Які рівні містить еталонна модель OSI та їх основні функції?
4. На які протокольні одиниці розбивається інформація для передачі даних мережею?
5. Які функції виконуються на транспортному рівні?
6. Які функції виконуються на мережевому рівні?
7. Які функції виконуються на фізичному рівні?
8. Які функції виконуються на канальному рівні?
9. Поясніть різницю між кадром, пакетом, дейтаграмою, сегментом.
10. На які рівні поділяється стек мережевих протоколів TCP/IP?
11. Поясніть процедуру інкапсуляції?
12. Призначення та функції аналізатора мережевого трафіку Wireshark.
13. Які види фільтрів підтримує програма Wireshark?
14. Наведіть приклади застосування різних видів фільтрів Wireshark.

Лабораторна робота 2
Аналіз просування даних по стеку TCP/IP з використанням аналізатора трафіку Wireshark. Транспортний і мережевий рівні

Мета роботи: засвоєння функцій модулів транспортного та мережевого рівнів стеку протоколів TCP/IP, структури заголовків протоколів TCP та UDP, псевдозаголовку, аналіз фрагментів протоколу TCP за допомогою аналізатора мережевого трафіку Wireshark.

План виконання лабораторної роботи

1. Ознайомитися та засвоїти теоретичні відомості, викладені в методичному посібнику до лабораторної роботи.
2. Виконати завдання до лабораторної роботи.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

При просуванні повідомлення з використанням протоколів стеку TCP/IP від прикладного рівня до фізичного рівня воно фрагментується (розбивається на сегменти) залежно від обмежень в каналі передачі. В процесі обробки відповідними протоколами до кожного блоку даних на кожному рівні додається службова інформація у вигляді заголовків певного формату.

На прикладному рівні реалізовані сервіси, які широко використовуються на практиці (рис. 2.1). До них належать: протокол передачі файлів між віддаленими системами FTP (File Transfer Protocol), протокол емуляції віддаленого терміналу telnet, поштові протоколи, протокол визначення імен DNS (Domain Name System) тощо. Кожна прикладна програма вибирає тип транспортування: або безперервний потік байтів, або послідовність окремих повідомлень. Прикладна програма передає дані транспортному рівню через порти, які можуть бути як стандартизованими, так і динамічними, значення яких змінюється від сеансу до сеансу.

Транспортний рівень підтримує два базових типи комунікації: сервіс, орієнтований на встановлення з'єднання, та сервіс без встановлення з'єднання (дейтаграмний). Сервіс комунікації з встановленням з'єднання передбачає процедуру встановлення, підтримки та розриву з'єднання між об'єктами верхніх рівнів. Цей тип комунікації використовують багато протоколів верхніх рівнів і називають надійним. Перевагами цього типу сервісу є: можливість виправлення помилок, управління потоками даних, контроль послідовності передачі блоків даних. Однак в деяких випадках більш ефективним є сервіс передачі даних без встановлення з'єднання. Це, в першу чергу, стосується програм, які потребують мінімальних сукупних витрат, пов'язаних з передачею даних. Прикладами таких користувачів транспортного сервісу є

системи збору даних, системи розповсюдження інформації, системи, що функціонують в діалоговому режимі «запит-відповідь» тощо.

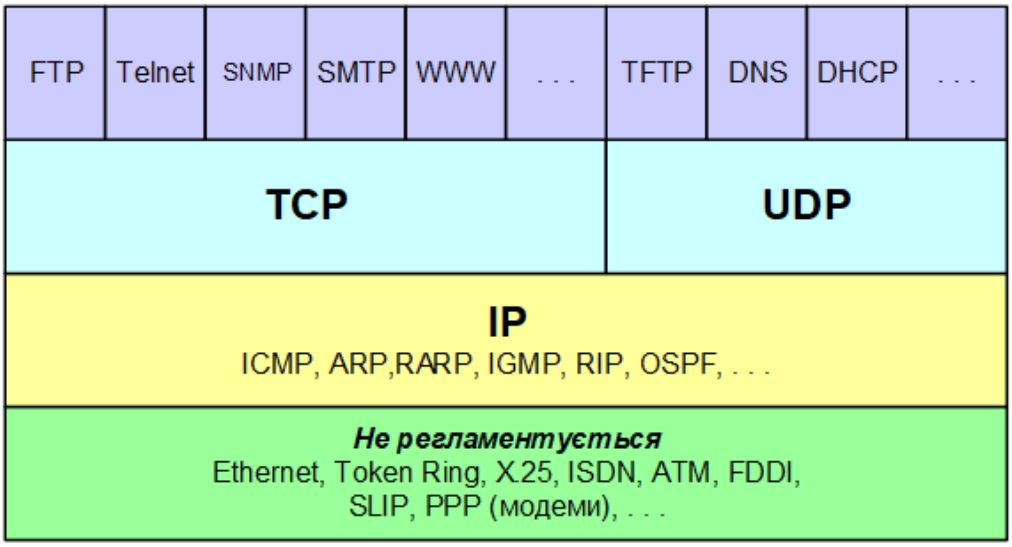


Рисунок 2.1– Стек протоколів TCP/IP

Передача повідомлення через мережу передбачає поділ його на блоки. Розміри цих блоків визначаються максимальним розміром блоку даних канального рівня MTU (Maximum Transfer Unit) або налаштуваннями, встановленими мережевим адміністратором. Ця процедура називається сегментацією. Саме тому блоки даних транспортного рівня називають сегментами. Зазвичай термін «сегмент» застосовують до одиниць передачі даних в межах надійного (гарантованого) транспортного сервісу, наприклад, TCP-сегменти. Блоки даних негарантованого транспортного сервісу називають UDP-дейтаграмами або просто дейтаграмами. На стороні одержувача виникає зворотна задача: формування повідомлення з отриманих сегментів.

Аналізуючи заголовок пакету, який отримано від мережевого рівня, транспортний модуль визначає за номером порту отримувача, якому із процесів-застосувань направлені дані, і пропонує ці дані відповідному процесу (можливо, після перевірки їх на наявність помилок і т.п.). Номери портів отримувача і відправника записуються в заголовок транспортним модулем, що відправляє дані; заголовок транспортного рівня містить також і іншу службову інформацію; формат заголовку залежить від транспортного протоколу, що використовується (UDP чи TCP).

Основні функції протоколів UDP та TCP, які найчастіше використовуються на практиці, представлено на рисунку 2.2. Опрацювання даних прикладного рівня на транспортному рівні відбувається по-різному і залежить від протоколу, який використовується на транспортному рівні: TCP чи UDP.

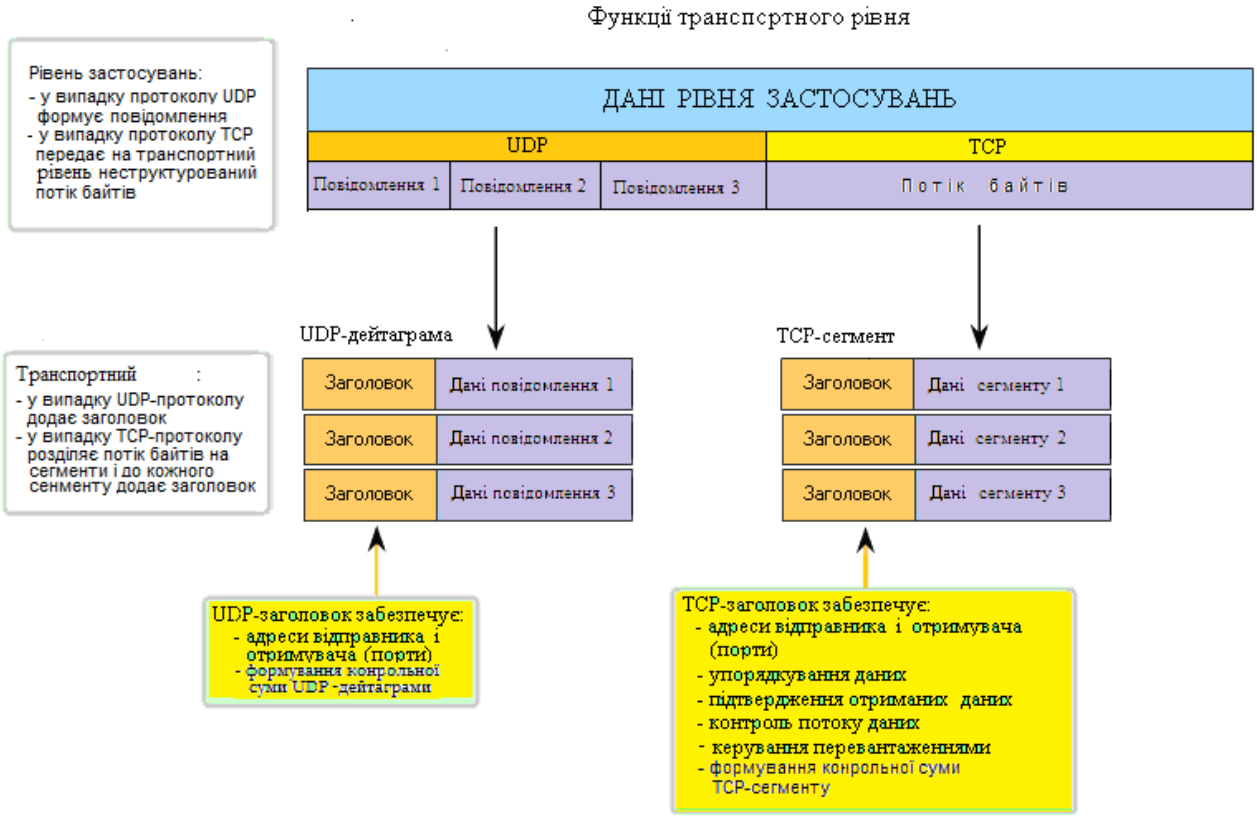


Рисунок 2.2 – Функції транспортного рівня

У випадку використання протоколу TCP дані прикладного рівня (потік байтів) можуть бути розділені на блоки, які після додавання до кожного з них TCP-заголовків називаються сегментами. Поділ потоку байтів на сегменти відбувається без врахування їхньої логічної структури, тобто межі між записами не зберігаються. Поділ або сегментація даних рівня застосувань гарантує по-перше можливість їх передачі в рамках технічних обмежень середовища передачі, а по-друге те, що дані із різних застосувань можуть бути мультиплексовані в каналі передачі. В TCP кожен заголовок сегменту містить порядковий номер. Цей порядковий номер дозволяє протоколам транспортного рівня на кінцевому хост-вузлі збирати сегменти у тому порядку, в якому вони були відправлені.

У випадку використання протоколу UDP опрацювання даних прикладного рівня обмежується приєднанням до повідомлення, яке надійшло із прикладного рівня, UDP-заголовку. Утворений таким чином блок даних називається дейтаграмою.

Хоча служби, які використовують UDP, також відслідковують взаємодію застосувань, вони не слідкують за порядком, в якому інформація була передана. Застосування, яке використовує UDP, повинне враховувати той факт, що дані, можливо, не надійдуть в тому порядку, в якому були відправлені.

1.1. Протокол UDP. Заголовок UDP-сегменту

Протокол передачі дейтаграм користувача UDP (User Datagram Protocol) фактично не виконує ніяких особливих функцій додатково до функцій мережевого рівня (протоколу IP). Протокол UDP використовується або при пересиланні коротких повідомлень, коли витрати на встановлення сеансу і перевірку коректної доставки даних виявляються вищими за витрати на повторну (при невдачі) пересилку повідомлення, або тоді, коли сама організація процесу-додатку забезпечує встановлення з'єднання і перевірку доставки пакетів (наприклад, NFS).

До даних користувача, що надійшли від прикладного рівня, додається UDP-заголовок (рис. 2.3), і сформована таким чином UDP-дейтаграма відправляється на мережевий рівень.

Рисунок 2.3 – Формат UDP-дейтаграми

- Source Port – номер порту процесу-відправника;
- Destination Port – номер порту процесу-отримувача;
- Length - довжина UDP-дейтаграми разом із заголовком, в октетах;
- Checksum – контрольна сума(за наявністю).. Поле Checksum містить контрольну суму пакету UDP, яка визначається для всього пакету UDP з доданим псевдозаголовком.

Після заголовку безпосередньо ідуть дані користувача, передані модулю UDP прикладним рівнем. Протокол UDP розглядає ці дані як одне повідомлення; він ніколи не розбиває повідомлення для передачі на кілька фрагментів і не об'єднує кілька повідомлень для пересилки в одному сегменті.

При отриманні пакету мережевого рівня модуль UDP перевіряє контрольну суму і передає вміст повідомлення прикладному процесу, номер порту якого вказаний в полі «Destination Port».

Якщо перевірка контрольної суми виявила помилку або якщо процесу, що підключений до потрібного порту, не існує, пакет ігнорується. Якщо пакети надходять швидше, ніж модуль UDP встигає їх опрацювати, то ці пакети також ігноруються. Протокол UDP є ненадійним протоколом без встановлення з'єднання, освільки:

- не має ніяких засобів підтвердження безпомилкового прийому даних або повідомлення про помилку;
- не забезпечує надходження повідомлень про порядок відправки;
- не виконує попереднього встановлення сеансу зв'язку між прикладними процесами.

Максимальна довжина UDP-дейтаграми дорівнює максимальній довжині IP-дейтаграми (65535 октетів) зменшеної на мінімальну довжину IP-заголовку (20) і UDP-заголовку (8), тобто 65507 октетів. На практиці зазвичай використовуються блоки даних довжиною 8192 октет.

Прикладами прикладних процесів, які використовують протокол UDP, є мережева файлова система NFS (Network File System), простий протокол передачі файлів TFTP (Trivial File Transfer Protocol), простий протокол керування мережею SNMP (Simple Network Management Protocol), доменна служба імен DNS (Domain Name Service).

1.2. Протокол TCP. Заголовок TCP-сегменту

Протокол TCP (Transmission Control Protocol) орієнтований на встановлення з'єднання між взаємодіючими прикладними процесами, який забезпечує надійну передачу та прийом даних. Метою протоколу є забезпечення надійного обміну даними між прикладними сервісами робочих станцій мережі. Для цього TCP забезпечує встановлення логічного з'єднання між прикладними процесами віддалених модулів мережі, контролює послідовність передачі сегментів повідомлення через комунікаційне середовище мережі, виявляє та обробляє помилки передачі.

При прийомі з мережі дейтаграми, в полі Protocol якої вказано код протоколу TCP (6), модуль IP передає дані цієї дейтаграми модулю TCP. Ці дані являють собою TCP-сегмент, що містить TCP-заголовок і дані користувача (дані прикладного сервісу). Модуль TCP аналізує службову інформацію заголовку, визначає, якому саме процесу потрібно передати дані користувача, перевіряє цілісність і порядок надходження даних і підтверджує їх прийом іншій стороні. В разі відсутності помилок і правильної послідовності даних користувача вони передаються прикладному процесу.

Максимальний розмір сегменту MSS (Maximum Segment Size), який TCP-рівень може надіслати на віддалений кінець з'єднання, вибирається таким чином, щоб при інкапсуляції сегменту в IP-пакет він поміщався туди цілком, тобто MSS не повинен бути більшим за максимальний розмір поля даних IP-дейтаграми, і розраховується за наступною формулою:

$$MSS = MTU - sizeof(TCPHDR) - sizeof(IPHDR),$$

де sizeof(TCPHDR) і sizeof(IPHDR) - відповідно розміри TCP та IP заголовків.

Розмір кожного з цих заголовків може бути від 20 до 60 байтів (при відсутності та наявності додаткових опцій відповідно). Тобто при відсутності додаткових параметрів передачі MSS = 1500 - 20 - 20 = 1460 байтів.

При встановленні з'єднання кожна сторона може оголосити своє значення MSS. І тоді вибирається найменше із двох значень.

Максимальний розмір сегменту TCP за замовчуванням – 536 байт. Якщо хост-вузол бажає встановити інше значення максимального розміру сегменту, то воно вказується як опція TCP в сегменті синхронізації TCP SYN під час

встановлення з’єднання TCP. Це значення не може бути змінене після того, як з’єднання встановлене.

TCP-сегмент, структура якого наведена на рис. 2.4, містить заголовок і дані і завжди кратний 4 байтам. Розмір заголовку – не менше 20 байт і може бути збільшений за рахунок додаткових опцій (парламентів) передачі до 60 байт.

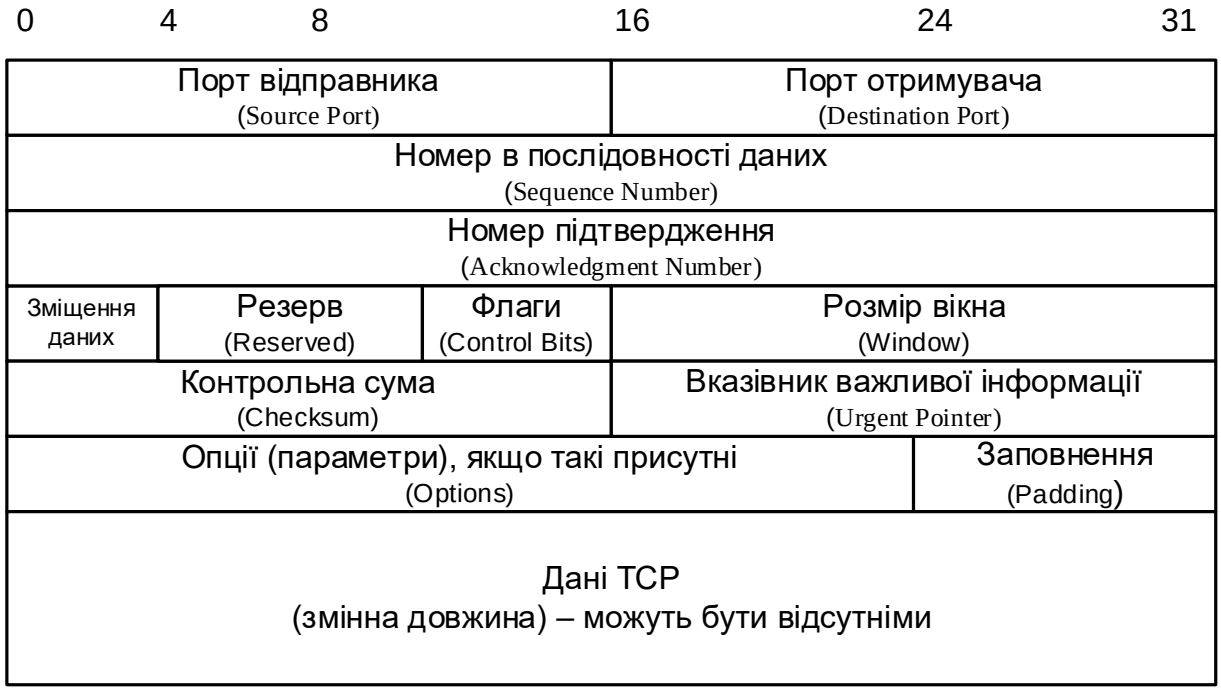


Рисунок 2.4 – Формат TCP-сегменту

- Source Port, Destination Port - номери портів процесу-відправника і процесу-отримувача відповідно;
- Sequence Number– порядковий номер першого октету в полі даних користувача. Значення, що присвоєне сегменту TCP, визначає номер стартового байту пакета, якщо тільки не встановлений флаг SYN; якщо в сегменті встановлено флаг SYN, то поле номера містить значення початкового порядкового номеру послідовності (ISN), а перший октет даних має номер ISN + 1;
- Acknowledgment Number (ACK) – значення, яке відправляється станції-відправнику, яке підтверджує прийом раніше відправленого сегменту(сегментів); задає також наступний порядковий номер байту, який станція очікує отримати; при встановленому з’єднанні сегмент підтвердження відправляється завжди;
- Data Offset (зміщення даних) – визначає довжину заголовку TCP-сегмента (тобто кількість 32-бітних блоків в заголовку TCP) та вказує, де закінчується заголовок та починаються дані (визначає зміщення даних в сегменті);

- Reserved (6 біт) – зарезервовано; заповнюється нулями.
- Control Bits – біти для управління; активним є стан «біт встановлений».

    - URG - **Флаг терміновості**. Індикатор терміновості, що використовується при відправці повідомлення адресату, який очікує на екстрену інформацію. Таке повідомлення може бути передано станції, що отримує дані, якщо вона закрила вікно прийому для відправника. Однак станція отримувач все ще буде приймати сегменти, в яких цей флаг встановлено.
    - ACK - **Флаг підтвердження**. Якщо даний флаг встановлено, то пакет містить підтвердження для дейтаграми, яка була передана раніше.
    - PSH - **Флаг форсованої відправки**. Сегмент з таким встановленим флагом вимагає виконання операції *push,* тобто включена функція виштовхування і необхідна невідкладна відправка даних після зчитування сегмента (даних цього пакета).
    - RST - **Перевстановлення з'єднання**. Обрив з'єднання з метою відмови на запит з'єднання. Переривання зв'язку (перезавантаження даного з'єднання).
    - SYN - **Флаг синхронізації** номерів сегментів в черзі, використовується для ініціалізації та встановлення порядкового значення.
    - FIN - **Флаг закінчення**. Визначає, що в ініціатора з'єднання (відправника даних) більше немає даних для відправлення, тобто передача інформації завершена.

В документі RFC 3168 запропоновано використовувати два молодших біти зарезервованої області для управління перевантаженням в мережі (рис. 2.5). Проте останнім часом згідно з RFC 3168 пропонується використовувати два старших біти TCP-прапорців для зберігання бітів ECN (Explicit Congestion Notification).

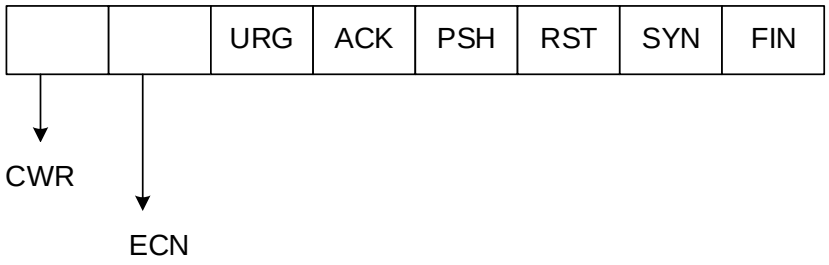


Рисунок 2.5 – Розташування бітів управління

Біт ECN (Explicit Congestion Notification), який називають ехо-бітом, встановлюється, коли в прийнятому сегменті встановлено біт повідомлення про перевантаження в байті «Тип обслуговування» IP-заголовку. Це означає, що

обидві сторони, які взаємодіють з використанням протоколу TCP, підтримують повідомлення про перевантаження, що виявляється в процедурі узгодження параметрів TCP-з'єднання. Exo-біт ECN в TCP-заголовку повідомляє відправника про необхідність знизити швидкість передачі даних через перевантаження в мережі між відправником і отримувачем. Після отримання TCP-сегменту з встановленим exo-бітом відправник в два рази зменшує вікно перевантаження – розмір буферу даних що відправляються. Після цього він встановлює біт CWR (Congestion Window Reduced), який свідчить про зменшення вікна перевантаження для повідомлення іншій стороні про виконані дії для зниження рівня перевантаження. Цей механізм дозволяє зменшити кількість відкинутих пакетів, але встановлення додаткових бітів може викликати повідомлення про тривогу від систем виявлення вторгнень. Тому на сьогодні більшість користувачів використовують ці біти тільки з метою сканування. Крім того, деякі пристрої фільтрації пакетів можуть не пропустити вхідні пакети з цими встановленими прапорцями. Тому доведеться виконати ще багато дій для поступового впровадження бітів ECN і для можливості відрізняти їх від спроб сканування.

- Window - величина вікна в октетах містить кількість байтів, які можуть бути відправлені після байту, отримання якого вже підтверджено. За допомогою значення величини вікна хост-отримувач повідомляє відправника про поточний розмір вхідного буфера для конкретного з'єднання. Значення величини вікна змінюється динамічно під час з'єднання. Воно зменшується на розмір прийнятих даних, але ще не опрацьованих хостом-отримувачем. Якщо вхідний буфер отримувача заповнюється повністю, то розмір вікна зменшується до 0, що повідомляє хосту-відправнику про необхідність тимчасово припинити передачу даних. Опрацювавши частину даних із вхідного буфера, хост-отримувач відправляє відправнику оновлену інформацію про розмір вікна, що вказує на можливість відновлення передачі даних. Очевидно, що хост-отримувач управляє потоком TCP-даних, в основному, за допомогою інформації про розмір вікна, тобто потоком даних в мережі керує переважно отримувач.
- Checksum – контрольна сума, яка крім заголовку сегменту і поля даних, враховує ще і псевдозаголовок, який записується між заголовком TCP (або UDP в разі його використання) та заголовком протоколу IP (рис.2.6). Цей псевдозаголовок містить IP-адресу відправника (4 октети), IP-адресу отримувача (4 октети), нульовий октет, 8-бітне поле «Протокол», яке аналогічне полю в IP-заголовку, і 16 біт довжини TCP сегменту в октетах. Такий підхід забезпечує захист протоколу TCP від сегментів, які «помилилися» в маршруті. Інформація для псевдозаголовку передається через інтерфейс «Протокол TCP/мережевий рівень» як аргументи або як результат обробки запитів від протоколу TCP до протоколу IP.

Метою використання псевдозаголовку є перевірка того факту, що TCP-сегмент **(**UDP-дейтаграмма) переданий в коректне місце призначення, яке в

загальному випадку характеризується сукупністю адреси конкретної робочої станції та порту додатка на ньому.

Рисунок 2.6 – Розташування псевдозаголовку та його структура

- Вказівник важливої інформації (Urgent Pointer) служить для зберігання довжини термінових даних, які розміщуються на початку поля даних сегменту. Вказує зміщення октету, який слідує за терміновими даними, відносно першого октету в сегменті. Наприклад, в сегменті передаються октети з 2001-го по 3000-й, при цьому перші 100 октетів є терміновими даними, тоді поле Urgent Pointer = 100. Протокол TCP не визначає, як саме мають опрацьовуватися термінові дані, але припускає, що прикладний процес буде намагатися швидко їх опрацювати. Поле Urgent Pointer обробляється тільки тоді, коли встановлений прапорець URG.
- Опції (Options) – поле змінної довжини; може бути відсутнім або містити одну опцію або список опцій, які реалізують додаткові послуги протоколу TCP. Опція складається із октету «Тип опції», за яким можуть знаходитися октет «Довжина опції в октетах» і октети з даними для опції.

  Стандарт протоколу TCP визначає три опції (типи 0, 1, 2).

Опції типів 0 і 1 («Кінець списку опцій» і «Немає операції» відповідно) складаються із одного октету, який містить значення типу опції. При виявленні в списку опції «Кінець списку опцій» аналіз опцій припиняється, навіть якщо довжина заголовку сегмента (Data Offset) ще не вичерпана. Опція «Немає операції» може використовуватися для вирівнювання між опціями по межі 32 біти.

Опція типу 2 «Максимальний розмір сегменту» складається із 4 октетів: одного октету типу опції (значення дорівнює 2), одного октету довжини (значення дорівнює 4) і двох октетів, які містять максимальний розмір сегменту MSS, який може отримати TCP-модуль, який відправив сегмент з даною опцією. Опцію можна використовувати тільки в SYN-сегментах на етапі встановлення з’єднання.

- Padding – вирівнювання заголовку по межі 32-бітного слова, якщо список опцій займає не ціле число 32-бітних слів, і заповнюється нулями.

1.3. Мережевий рівень і протокол IP

На мережевому рівні стеку протоколів TCP/IP функціонує протокол IP (будь-якої версії). Згідно документу RFC-791 протокол IP забезпечує передачу блоків даних, які називаються дейтаграмами, від відправника до отримувача, де відправниками і отримувачами є комп'ютери (робочі станції), які ідентифікуються адресами фіксованої довжини (IP-адресами). Протокол IP виконує, за необхідності, також фрагментацію і зборку дейтаграм для передачі даних через мережу з невеликою кількістю пакетів.

Протокол IP доставляє блоки даних, які називаються дейтаграмами, від хост-вузла з певною IP-адресою до іншого хост-вузла. IP-адреса − це унікальний 32-бітний (в разі використання протоколу IP версії 4) ідентифікатор комп'ютера або іншого модуля мережі (його мережевого інтерфейсу). Дані для дейтаграми передаються IP-модулю з транспортного рівня. IP-модуль доповнює ці дані заголовком, в який додається IP-адреса відправника і отримувача та інша службова інформація. Сформована таким чином дейтаграма передається на рівень доступу до середовища передачі (наприклад, одному із фізичних інтерфейсів) для відправки в канал передачі даних.

Коли модуль IP отримує дейтаграму з нижнього рівня, він перевіряє IP-адресу призначення. Якщо дейтаграма адресована даному комп'ютеру, то дані з неї передаються на обробку модулю розташованого вище рівня (якому саме – вказано у заголовку дейтаграми). Якщо ж адреса призначення дейтаграми – інший комп'ютер, то модуль IP може прийняти одне з двох рішень: або знищити дейтаграму, або відправити її до місця призначення, визначивши маршрут проходження (таким чином функціонують проміжні станції – маршрутизатори).

Також може виникнути необхідність на границі мереж з різними характеристиками розділити дейтаграму на фрагменти, а потім зібрати в єдине ціле на комп'ютері-отримувачі.

Якщо модуль IP з будь-якої причини не може доставити дейтаграму, вона знищується. При цьому модуль IP відправляє комп'ютеру-відправнику цієї дейтаграми повідомлення про помилку. Такі повідомлення відправляються за допомогою протоколу ICMP (Internet Control Message Protocol), який є невід'ємною частиною модуля IP. Ніяких інших засобів контролю коректності даних, підтвердження їх доставки, забезпечення коректного порядку проходження дейтаграм, попереднього встановлення з'єднання між комп'ютерами протокол IP не має. Ці задачі виконує транспортний рівень.

Протокол IP є *ненадійним* протоколом *без встановлення з'єднання*, що в свою чергу означає, що протокол IP не підтверджує доставку даних, не контролює цілісність отриманих даних і не виконує операцію квотування (handshaking) – обмін службовими повідомленнями, які підтверджують встановлення з'єднання з вузлом призначення і його готовність до прийому

даних. Протокол IP обробляє кожну дейтаграму як незалежну одиницю, яка не має зв'язку з іншими дейтаграмами в Інтернет. Після того, як дейтаграма відправляється в мережу, її подальша доля ніяк не контролюється відправником (на рівні протоколу IP). Якщо дейтаграма не может бути доставлена, вона знищується. Вузол, який знищує дейтаграму, може відправити за зворотною адресою ICMP-повідомлення про помилку. Гарантію правильності передачі даних надають протоколи вищерозташованого рівня (наприклад, протокол TCP), які мають для цього необхідні механізми.

IP-дейтаграма складається із заголовку і поля даних. Заголовок дейтаграми складається з 32-розрядних слів і має змінну довжину, яка залежить від розміру поля додаткових параметрів «Options», але завжди кратна 32 бітам. Довжина заголовка дейтаграми – від 20 до 60 байтів. За заголовком безпосередньо розташовані дані, які передаються в дейтаграму (рис. 2.7).

Рисунок 2.7 – Формат заголовку IP-дейтаграми

Значення полів заголовку наступні:

- Ver - версія протоколу IP, в лабораторній роботі використовується протокол версії 4 (хоча існують інші версії, які мають номери 5-8).
- IHL (Internet Header Length) - довжина заголовку в 32-бітних словах; діапазон допустимих значень від 5 (мінімальна довжина заголовку, поле «Options» відсутнє) до 15 (тобто, може мати максимум 40 байтів опцій).
- TOS (Type Of Service)– значення поля визначає пріоритет дейтаграми і необхідні параметри передачі (тип маршрутизації). Структура байту TOS представлена на рис. 2.8.

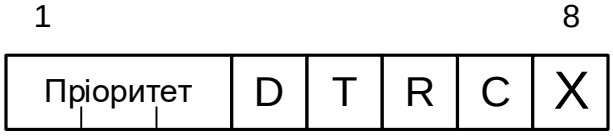


Рисунок 2.8 – Структура байту «Тип сервісу»

Пріоритети ((precedence) на практиці використовується рідко):
0 (000) – звичайний;

1 (001) – пріоритетний;
2 (010) – негайний;
3 (011) – терміновий (миттєвий);
4 (100) – екстрений (більш, ніж миттєвий);
5 (101) – CRITIC/ECP (обробка критичних і екстерних викликів)**;**
6 (110) – міжмережне управління;
7 (111) – мережне управління.

Біти D, T, R, C визначають бажаний тип маршрутизації (вибір маршруту):
D – Date – з мінімальною затримкою;
T – Throughput – з максимальною пропускною спроможністю;
R – Reliability – з максимально. надійністю;
C – Cost – з мінімально. вартістю.
За замовчуванням всі біти встановлені в 0 (звичайний сервіс).

В дейтаграмі може бути встановлений тільки один із бітів D, T, R, C. Якщо необхідно реалізувати максимальну безпеку передачі, то встановлюються всі ці чотири біти.

Реальне урахування пріоритетів і вибір маршруту у відповідності з значенням байту TOS залежить від маршрутизатора, його програмного забезпечення і налаштувань. Маршрутизатор може підтримувати розрахунок маршрутів для усіх типів TOS, для частини або ігнорувати TOS взагалі. Маршрутизатор може враховувати значення пріоритету при обробці всіх дейтаграм або при обробці дейтаграм, які надходять тільки від деякої обмеженої множини вузлів мережі, або зовсім ігнорувати пріоритет.

З моменту появи у складі IP-заголовку байт «Тип обслуговування» TOS (Type of Service) зазнав кількох змін. Однією з них стало передбачене в документі RFC 2481 і більш новому RFC 3168 використання двох молодших бітів цього байту для зберігання явного повідомлення про перевантаження ECN (Explicit Congestion Notification). Це пов'язано з використанням деякими маршрутизаторами методу випадкового раннього виявлення RED (Random Early Detection) або активного управління чергами з вірогідністю втрати пакетів.

При високому навантаженні в мережі маршрутизатор може відкидати деякі пакети. Метод RED призначений для зменшення негативного ефекту втрати пакетів за допомогою обчислення вірогідності перевантаження черги до інтерфейсу маршрутизатора і маркування пакетів, які можуть бути відкинуті при виникненні такого перевантаження.

- Total Length – довжина всієї дейтаграми в октетах, разом із заголовком і даними, максимальне значення 65535, мінімальне – 21 (заголовок без опцій і один октет в полі даних).
- ID (Identification), Flags (3 біти), Fragment Offset (13 біт) використовуються для фрагментації та зборки дейтаграм.

Ідентифікатор дейтаграми містить унікальний код дейтаграми, що визначає порядковий номер дейтаграми в послідовності. Це поле використовується приймаючим вузлом для об'єднання в єдине ціле фрагментів дейтаграми. Значення цього поля встановлюється відправником.

Флаги

Біт 0 – зарезервований.

Біт 1 виконує управління фрагментацією:

DF=0 – можна фрагментувати,

DF=1 – фрагментація заборонена.

Біт 2 показує, чи є даний фрагмент останнім.

MF=0 – останній фрагмент,

MF=1 – є наступний фрагмент.

Зміщення фрагменту показує, де у вихідній дейтаграмі перебуває даний фрагмент. Величина зміщення задається в **64-бітних блоках.** Перший фрагмент має нульове зміщення. Поле загальної довжини вказує довжину вихідного пакету, а поле зміщення показує вузлу, що здійснює збирання, зміщення відносно його початку.

- TTL (Time To Live) – «час життя» дейтаграми. Встановлюється відправником, вимірюється в кількості комунікаційних вузлів, через які може передаватися дейтаграма, або в секундах. Кожен маршрутизатор, через який проходить дейтаграма, переписує значення TTL, попередньо віднімаючи від нього час, витрачений на обробку дейтаграми. Оскільки швидкість обробки даних на маршрутизаторах велика, на одну дейтаграму витрачається зазвичай менше секунди, тому фактично кожен маршрутизатор зменшує TTL на одиницю. При досягненні значення TTL=0 дейтаграма знищується, при цьому відправнику зазвичай відправляється відповідне ICMP-повідомлення. Використання та контроль TTL запобігає зациклюванню дейтаграми при передачі в мережі.
- Protocol (8 біт) – визначає програму (розташований вище протокол стеку), якій мають бути передані дані дейтаграми для подальшої обробки. Коди деяких протоколів, які найчастіше використовуються на практиці, наведені в таблиці 2.1.
- Header Checksum – контрольна сума заголовку, вираховується як доповнення до суми всіх 16-бітових слів заголовку. Перед підрахунком контрольної суми значення поля «Header Checksum» обнуляється. Оскільки маршрутизатори змінюють значення деяких полів заголовку при обробці дейтаграми (як мінімум, поля TTL), контрольна сума кожним маршрутизатором перераховується заново. Якщо при перевірці контрольної суми виявляється помилка, дейтаграма знищується.
- Source Address - IP-адреса відправника.
- Destination Address - IP-адреса отримувача.

Таблиця 2.1 - Коди протоколів

Код	Протокол	Опис
1	ICMP	Протокол контрольних повідомлень
2	IGMP	Протокол керування групою хостів
3	GGP	Протокол «шлюз-шлюз»
4	IP	IP поверх IP (інкапсуляція)
6	TCP	TCP
8	EGP	Протокол зовнішньої маршрутизації (застарів)
9	IGP	Протокол внутрішньої маршрутизації (застарів)
17	UDP	UDP
27	RDP	Протокол надійних даних
28	IRTP	Протокол міжмережевої надійної передачі
46	RSVP	Протокол резервування ресурсів при мультикастингу
88	IGRP	Протокол внутрішньої маршрутизації компанії CISCO
89	OSPF	Протокол внутрішньої маршрутизації

- Options - опції, поле змінної довжини. Опцій може бути одна, декілька або жодної. Опції визначають додаткові послуги модуля IP по обробці дейтаграми, в заголовок якої вони включені.
- Padding – вирівнювання заголовку по границі 32-бітного слова, якщо список опцій займає не ціле число 32-бітних слів. Поле «Padding» заповнюється нулями.

2. Аналіз захоплених пакетів TCP за допомогою програми Wireshark

Завдання 1

За допомогою програми Wireshark необхідно виконати захоплення даних сеансу FTP і визначити значення полів заголовків протоколу TCP при передачі файлів з використанням протоколу FTP між хост-комп’ютером і анонімним FTP-сервером. Під’єднання до анонімного FTP-серверу і завантаження файлу виконується за допомогою браузера.

1. Активізувати режим захоплення даних з використанням програми Wireshark.
2. Завантажити файл довідки README.TXT.
   2.1.Під ’єднатися до FTP-сервера центру FreeBSD: ftp://ftp3.ie.freebsd.org.
   2.2. В розділі pub/FreeBSD знайти і завантажити файл README.TXT (рисунок 2.9).
   2.3.Після завершення завантаження файлу зупинити захоплення даних програмою Wireshark.

Содержание /

Имя	Размер	Дата изменения
favicon.ico	5.3 kB	21.04.2020, 03:00:00
index.html	668 B	21.04.2020, 03:00:00
pub/		21.04.2020, 03:00:00

Имя	Размер	Дата изменения
FreeBSD/		28.10.2020, 13:45:00

Имя	Размер	Дата изменения
README.TXT	4.2 kB	07.05.2015, 03:00:00
TIMESTAMP	35 B	28.10.2020, 13:45:00
development/		28.10.2020, 13:45:00
dir.sizes	2.6 kB	28.10.2020, 12:00:00
doc/		12.11.2017, 02:00:00
ports/		12.11.2017, 02:00:00
releases/		28.10.2020, 13:45:00
snapshots/		09.11.2018, 02:00:00

Рисунок 2.9 – Скріншот екрану

3. Відкрити головне вікно програми Wireshark.

Під час сеансу FTP-підключення до сайту **ftp3.ie.freebsd.org** захоплюється достатньо велика кількість блоків даних. Для того, щоб обмежити кількість цих даних для подальшого аналізу, необхідно застосувати фільтр відображення tcp and ip.addr == 139.178.72.202 в полі Filter і натиснути клавішу Enter (рис. 2.10). Введена IP-адреса 139.178.72.202 є адресою сайту **ftp3.ie.freebsd.org**.

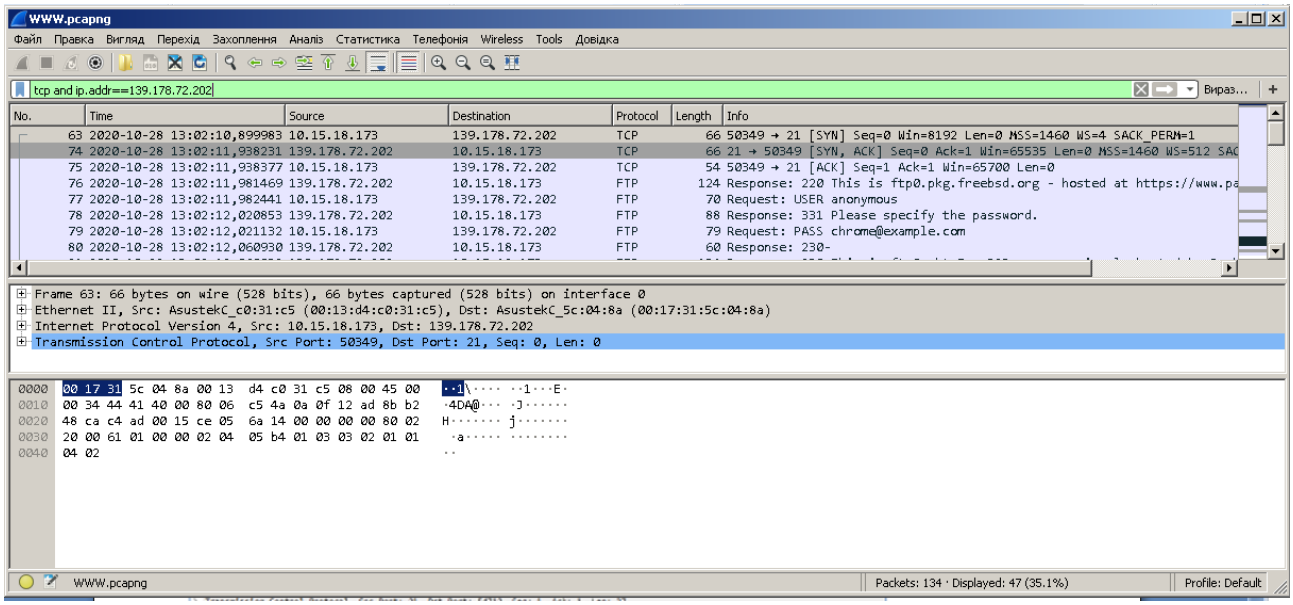


Рисунок 2.10 – Приклад захоплення блоків даних

4. Проаналізувати поля заголовків сегментів TCP.

Після застосування фільтру TCP в перших трьох блоках, показаних в верхньому розділі панелі, відображено створення надійного сеансу зв'язку протоколом транспортного рівня TCP.

Послідовність [SYN], [SYN, ACK] і [ACK] ілюструє тристороннє квотування (рисунок 2.11).

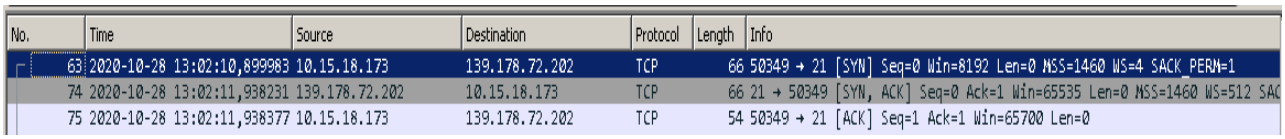


Рисунок 2.11 – Встановлення з'єднання (триразове рукостискання)

Протокол TCP, як правило, використовується під час сеансу зв'язку для управління доставкою IP-дейтаграм, перевірки їх отримання і керування розміром вікна. Для кожного обміну даними між FTP-клієнтом і FTP-сервером запускається новий сеанс TCP. Після завершення передачі даних сеанс TCP закривається. Після завершення сеансу FTP протокол TCP виконує планове відключення і припиняє роботу.

Програма Wireshark відображає докладні дані TCP-пакетів на панелі відомостей про пакети (середній розділ). Виділити перший фрагмент TCP і розгорнути його. Відкриється розгорнутий фрагмент TCP, аналогічний представленому на рисунку 2.12.

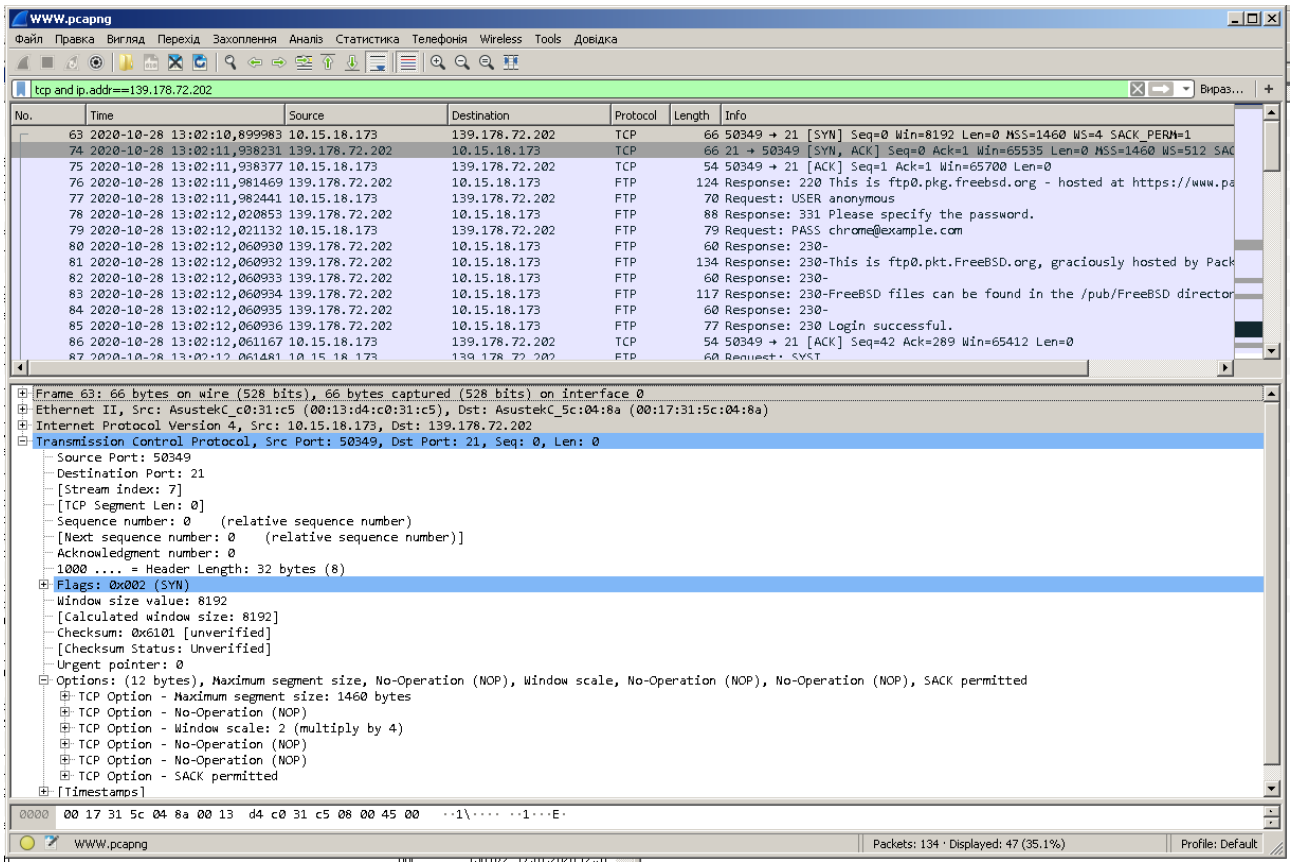


Рисунок 2.12 – Приклад розгорнутого фрагменту TCP

До кожного поля надаються пояснення.

- Поле TCP Source Port Number (номер порту джерела) призначене для хост-вузла, який відкрив з'єднання.
- Поле TCP Destination Port Number (номер порту призначення) використовується для ідентифікації протоколу верхнього рівня або додатку на віддаленому сайті. Значення в діапазоні від 0 до 1023 зв'язані з популярними сервісами і застосуваннями, наприклад Telnet, FTP і HTTP. Комбінація IP-адреси джерела, порту джерела, IP-адреси призначення і порту призначення однозначно визначають сеанс як для відправника, так і для отримувача. В наведених прикладах вказано порт призначення 21, який використовується для FTP. FTP-сервери прослуховують порт 21 для підключення FTP-клієнтів.
- В полі Sequence Number (порядковий номер) вказується номер першого октету в сегменті.
- В полі Acknowledgment Number (номер підтвердження) вказується наступний октет, який очікується отримувачем.
- Значення в полі Flags виконують особливу роль в управлінні сеансами і опрацюванні сегментів. Найчастіше використовуються наступні:
  - ACK — підтвердження отримання сегменту.
  - SYN — синхронізація, встановлюється тільки в першому TCP-сегменті в процесі тристороннього квотування.
  - FIN — завершення, запит про припинення сеансу TCP.
- Window size (розмір вікна) — це значення віна зміщення, яке визначає число октетів, які можуть бути передані до надходження підтвердження.
- Поле Urgent pointer (вказівник важливості) використовується тільки з встановленим прапорцем важливості Urgent (URG), коли користувачу необхідно передати важливі дані отримувачу, на які потрібно звернути особливу увагу.
- Поле Options (параметри) − поле змінної довжини; може бути відсутнім або містити одну опцію або список опцій, які реалізують додаткові послуги протоколу TCP. В лабораторній роботі треба звернути особливу увагу на опцію максимального розміру TCP-сегменту MSS.

Після встановлення сеансу TCP з'являється можливість для передачі FTP-трафіка між робочою станцією та FTP-сервером. FTP-клієнт і сервер взаємодіють один з одним, не помічаючи, що при цьому TCP займається управлінням сеансом.

Після завершення сеансу FTP клієнт FTP відправляє команду quit (завершити). FTP-сервер підтверджує припинення сеансу FTP, відправляючи Response: 221 Goodbye. У відповідь на це сеанс TCP FTP-сервера відправляє сегмент TCP FTP-клієнту, повідомляючи про припинення сеансу TCP.

Сеанс TCP FTP-клієнта підтверджує отримання сегмента припинення сеансу, після чого відправляє власне повідомлення про припинення сеансу TCP. FTP-сервер, який ініціював припинення сеансу TCP, відправляє сегмент з

встановленим бітом ACK з підтвердженням припинення, і сеанс TCP завершується. Послідовність цих дій представлена на рисунку 2.13.

Коли FTP-сервер передав всі дані, відправляється сегмент з встановленим прапорцем FIN. Робоча станція у відповідь відправляє ACK, щоб підтвердити отримання FIN. Станція відправляє сегмент з встановленим бітом FIN FTP-серверу для завершення сеансу TCP. FTP-сервер відправляє відповідь, яка містить встановлений біт ACK для підтвердження отримання FIN від робочої станції. Після цього сеанс TCP між FTP-сервером і робочою станцією завершується.

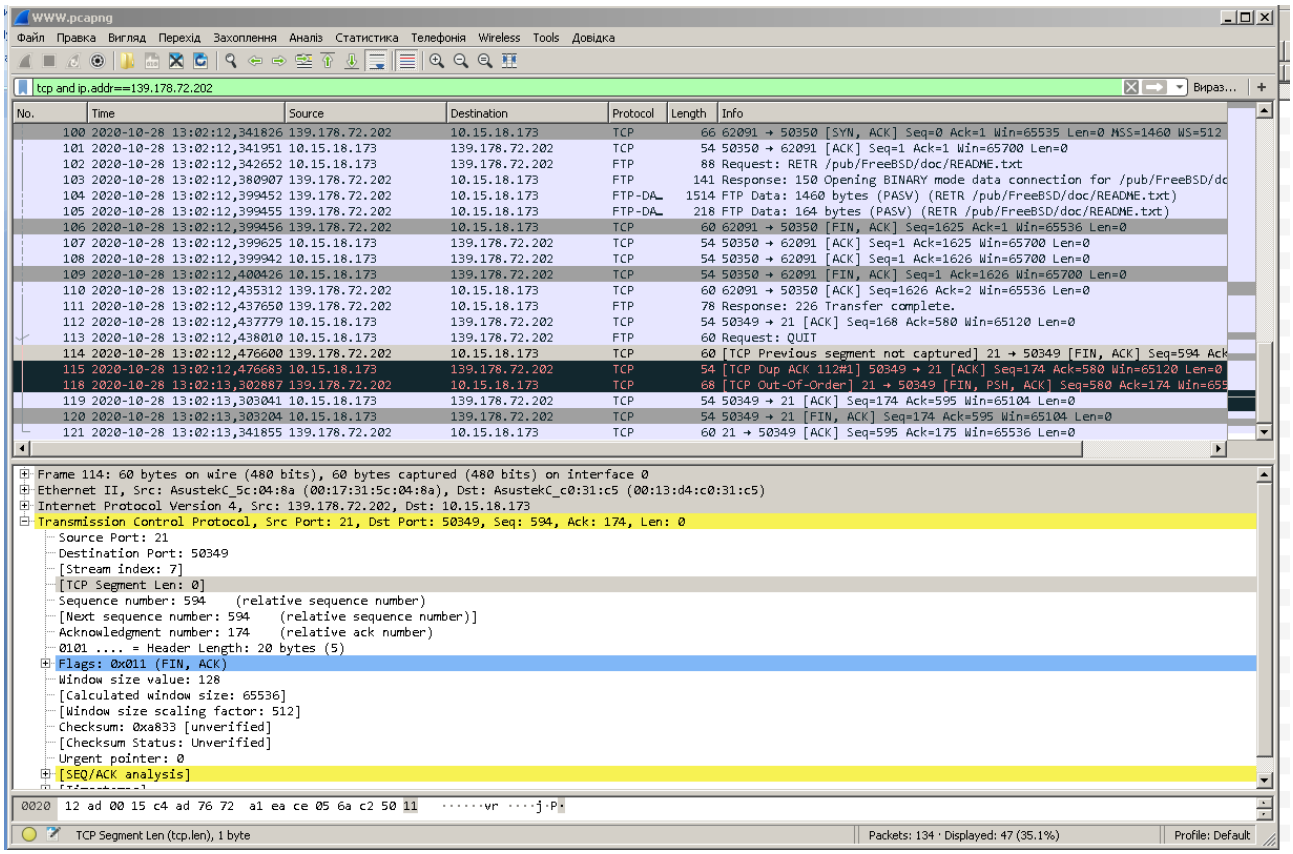


Рисунок 2.13 – Приклад сеансу TCP

Завдання 2

1. Ознайомитись з можливостями фільтрації даних за різними ознаками, зокрема, за MAC-адресою відправника і отримувача. Фільтр створюється за описаною вище методикою. Відповідно до рекомендачій викладача сформувати фільтр за MAC-адресою.
2. Розглянути результат інкапсуляції при передачі даних. В захоплених пакетах виділити службову інформацію (заголовки) всіх блоків даних, а також, за наявністю, кінцевика.
3. Використовуючи фільтр відображення **tcp.flags.syn = = 1** відібрати сегменти-запити, які містять встановлений прапорець **SYN** у заголовку та сегменти-відповіді, які містять встановлені прапорці SYN та ACK. Провести

аналіз поля **Options** заголовку TCP. Яке значення MSS використовується в з'єднанні, що аналізується?

4. За допомогою меню «Statistics» необхідно отримати і додати до звіту таку інформацію:
    - кількість захоплених пакетів та байтів;
    - середня швидкість передачі даних (в бітах за секунду);
    - середній розмір пакета;
    - час, протягом якого здійснювалось захоплення трафіку;
    - вивести таблицю Ethernet Conversations та пояснити вміст її рядків;
    - вивести IO Grafs, за допомогою якого визначити пікову швидкість передачі даних протягом інтервалу, що підлягає аналізу.
5. За результатами роботи зробити висновки.

Контрольні запитання

1. Поясніть процедуру створення TCP-сегментів і UDP-дейтаграм на транспортному рівні стеку протоколів TCP/IP?
2. Яка довжина заголовків протоколів транспортного рівня UDP і TCP)
3. Поясніть назви та призначення полів в заголовках протоколів транспортного рівня?
4. Як вираховується контрольна сума TCP-сегменту та UDP-дейтаграми?
5. Які поля входять до складу псевдозоголовку?
6. Як відбувається встановлення і розірвання TCP-з'єднання?
7. Яка довжина заголовку протоколу мережевого рівня (IP-заголовку)?
8. Які поля містить IP-заголовок та їх призначення?
9. Які види фільтрів підтримує аналізатор трафіку Wireshark?

Лабораторна робота 3
Аналіз просування IP-пакетів в об'єднаній мережі з використанням аналізатора трафіку Wireshark. Рівень мережевих інтерфейсів. Фрагментація IP-дейтаграм

Мета роботи: засвоєння функцій модулів мережевих інтерфейсів, структури заголовку кадру Ethernet, структури мережевого адаптера, процедури фрагментації IP-дейтаграм за допомогою аналізатора мережевого трафіку Wireshark.

План виконання лабораторної роботи

1. Ознайомитися та засвоїти теоретичні відомості, викладені в посібнику до лабораторної роботи.
2. За допомогою аналізатора Wireshark виконати захоплення та провести аналіз фрагментованих мережевих пакетів.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

Розглянемо просування IP-пакету через об'єднану мережу, структура якої представлена на рисунку 3.1. Розглянемо спрощений варіант, вважаючи, що вузли мережі мають IP-адреси без масок.

Проаналізуємо та розглянемо детально як взаємодіє протокол IP з протоколами визначення адрес ARP (Address Resolution Protocol) і системою доменних імен DNS (Domen Name System).

1.1. Формування DNS-запиту

Користувачу робочої станції fmm.lst.org, який знаходиться в мережі з адресою 177.43.0.0, необхідно встановити зв'язок з FTP-сервером. Користувачу відоме символьне ім'я сервера ftp.lst.org, тому він набирає в браузері команду звернення до FTP-сервера:

ftp://ftp.lst.org

Виконання цієї команди ініціює наступні операції:

- DNS-клієнт fmm.lst.org звертається до найближчого DNS-сервера 243.7.18.8 із запитом про IP-адресу сервера ftp.lst.org, з яким йому необхідно зв'язатися;
- DNS-сервер, опрацювавши запит, передає DNS-клієнту відповідь про знайдену IP-адресу сервера ftp.lst.org;
- FTP-клієнт fmm.lst.org, використовуючи отриману IP-адресу сервера ftp.lst.org, формує і передає повідомлення на FTP-сервер.

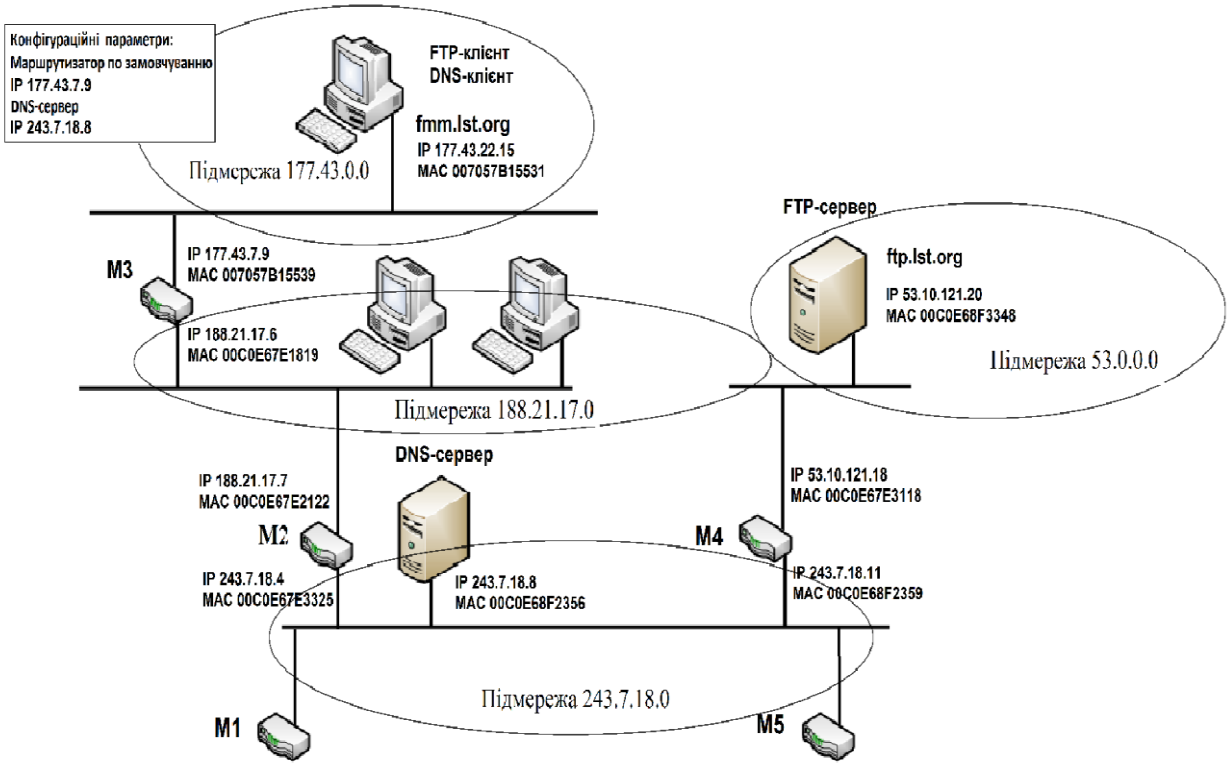


Рисунок 3.1 – Фрагмент об’єднаної мережі

Розглянемо послідовно, як при виконанні цих операцій взаємодіють протоколи DNS, IP, ARP і Ethernet, і що відбувається при цьому з кадрами і пакетами.

Формування IP-пакету з інкапсульованим в нього DNS-запитом

Програмний модуль FTP-клієнта, отримавши команду

ftp:// ftp.lst.org,

передає запит до працюючої на цьому ж комп’ютері клієнтської частини протоколу DNS, яка в свою чергу, формує запит до DNS-сервера на перетворення доменного імені ftp.lst.org в IP-адресу. Запит упаковується в UDP-дейтаграму, потім в IP-пакет. У заголовку пакету вказується IP-адреса 243.7.18.8 DNS-сервера (адреса отримувача). Ця адреса відома програмному забезпеченню клієнтського комп’ютера, оскільки вона входить до складу його конфігураційних параметрів. Сформований IP-пакет буде передаватися через мережу в незмінному вигляді (рис. 3.2), поки не надійде до DNS-сервера.

Рисунок 3.2 – Інкапсуляція дейтаграми UDP з DNS-запитом в IP-пакет

Передача кадру Ethernet з IP-пакетом маршрутизатору М3

Для передачі IP-пакету необхідно упакувати його в кадр Ethernet, вказавши у заголовку MAC-адресу отримувача. Протокол Ethernet може доставляти кадри тільки тим адресатам, які розташовані в межах однієї локальної мережі з відправником. Якщо ж адресат розташований в іншій мережі, то кадр потрібно передати маршрутизатору для подальшого просування пакету. Для цього модуль IP, порівнявши номери мереж в адресах відправника і отримувача, тобто 177.43.22.15 і 243.7.18.8, визначає, що пакет направляється в іншу мережу, тобто, що його потрібно передати маршрутизатору М3. IP-адреса маршрутизатора по замовчуванню також відома клієнтському вузлу – вона входить до складу конфігураційних параметрів.

Але для кадру Ethernet необхідно вказати не IP-адресу, а MAC-адресу отримувача. Таке перетворення адрес виконує протокол ARP за допомогою ARP-таблиці. Оскільки звернення до маршрутизатора відбуваються доволі часто, будемо вважати, що потрібна MAC-адреса в ARP-таблиці є і вона має значення 00:70:57:B1:55:39. Тепер клієнтський комп'ютер fmm.lst.org може відправити маршрутизатору М3 пакет, упакований в кадр Ethernet (рис.3.3).

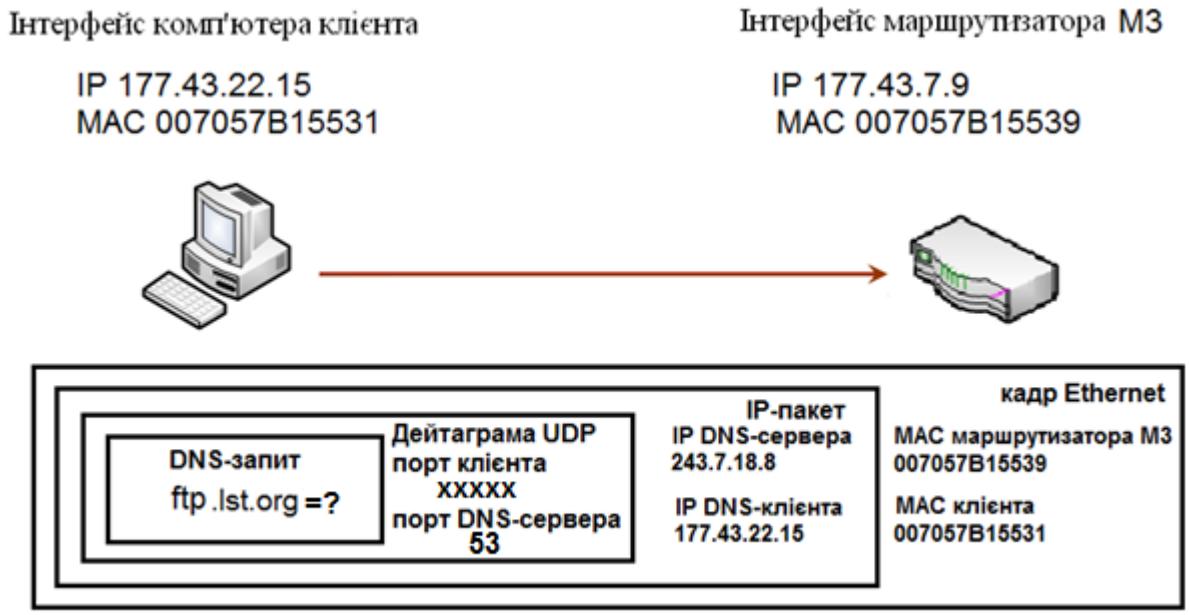


Рисунок 3.3 – Кадр Ethernet з інкапсульованим IP-пакетом, відправлений із клієнтського комп'ютера на маршрутизатор М3

Визначення IP-адреси і MAC-адреси наступного маршрутизатора М2

Кадр Ethernet приймається вхідним інтерфейсом 177.43.7.9 маршрутизатора М3. Протокол Ethernet, який працює на цьому інтерфейсі, перевіряє кінцевик фрейму FCS (зазвичай чотирибайтне значення циклічного коду), за допомогою якого перевіряється наявність помилок в прийнятих даних. Значення FCS вираховується відправником і записується у відповідне поле. Модуль, який приймає фрейм, вираховує дане значення самостійно і порівнює з отриманим.

Якщо виявлені помилки, фрейм знищується. Якщо помилок немає, заголовок фрейму і кінцевик відкидаються, а IP-пакет передається модулю протоколу IP, який знаходить у заголовку пакету адресу 243.7.18.8 і проглядає записи своєї таблиці маршрутизації. Можливо, що маршрутизатор М3 не знаходить маршруту безпосередньо до адреси призначення 243.7.18.8, але знаходить рядок в своїй таблиці маршрутизації, який містить адресу мережі призначення:

243.7.18.0 188.21.17.7 188.21.17.6

Цей рядок вказує, що пакети для мережі 243.7.18.0 маршрутизатор М3 повинен передавати на свій вихідний інтерфейс 188.21.17.6, з якого вони надійдуть на інтерфейс наступного маршрутизатора М2, який має IP-адресу 188.21.17.7. Однак IP-адреси маршрутизатора недостатньо, щоб передати пакет через мережу Ethernet. Потрібна ще MAC-адреса маршрутизатора М2. Протокол IP активує модуль протоколу ARP і, якщо в ARP-таблиці немає запису про MAC-адресу маршрутизатора М2, то формується та надсилається в мережу широкомовний ARP-запит, який надходить на всі інтерфейси модулів мережі 188.21.17.0. Відповідь надає тільки вхідний інтерфейс маршрутизатора М2 (00:C0:E6:7E:21:22). Тепер, знаючи MAC-адресу маршрутизатора М2, маршрутизатор М3 надсилає йому IP-пакет з DNS-запитом (рис. 3.4).

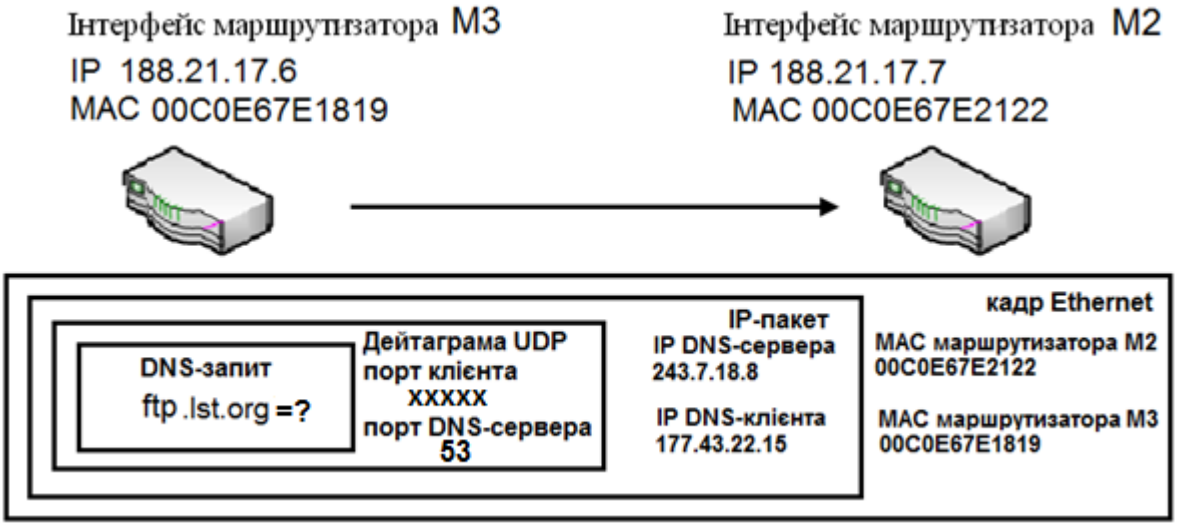


Рисунок 3.4 – Кадр Ethernet з інкапсульованим IP-пакетом, відправлений з маршрутизатора М3 на маршрутизатор М2

IP-пакет знову інкапсулюється в кадр Ethernet з вже відомими MAC-адресами: до нього приєднується новий Ethernet заголовок і новий кінцевик.

Маршрутизатор М2 доставляє пакет DNS-серверу

Модуль Ethernet маршрутизатора М2 перевіряє кінцевик FCS кадру. Якщо виявлені помилки, пакет знищується. Якщо помилки не виявлені, кадр передається модулю IP маршрутизатора М2, який відкидає заголовок кадру Ethernet і кінцевик і вилучає з IP-пакету адресу призначення. Далі модуль IP проглядає таблицю маршрутизації. В ній він знаходить, що мережа призначення 243.7.18.0 безпосередньо під'єднана до його інтерфейсу 243.7.18.4. Це означає, що пакети не потрібно маршрутизувати, однак потрібно визначити MAC-адресу вузла призначення. Протокол ARP, взаємодіючи з протоколом IP, як це описано раніше, визначає MAC-адресу DNS-сервера (00:C0:E6:8F:23:56). Отримавши відповідь про MAC-адресу, маршрутизатор М2 відправляє в мережу призначення кадр Ethernet з DNS-запитом (рис. 3.5).

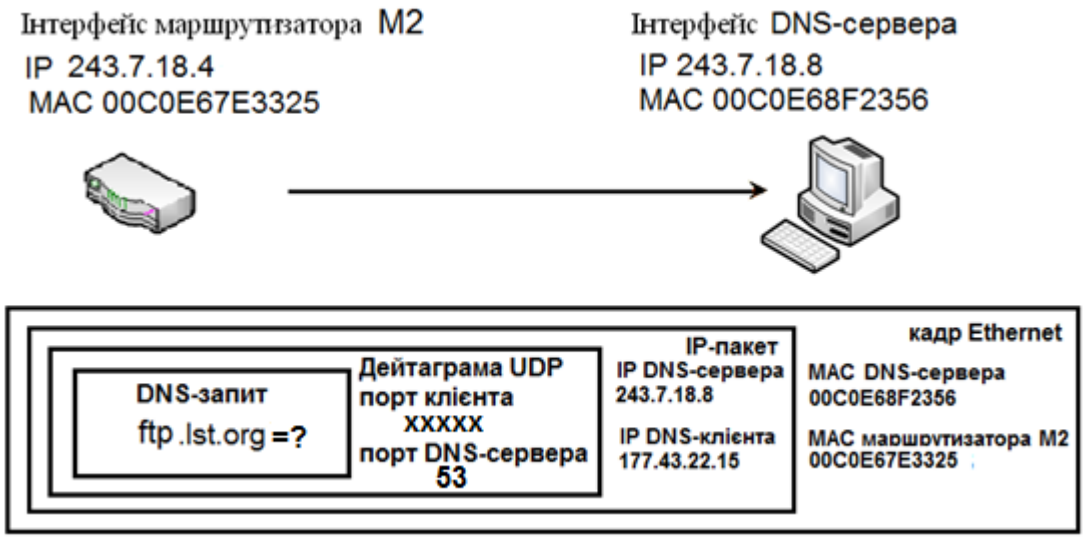


Рисунок 3.5 – Кадр Ethernet з DNS-запитом, відправлений з маршрутизатора М2

Прийом кадру Ethernet мережевим адаптером DNS-сервера

Мережевий адаптер DNS-сервера приймає кадр Ethernet, виявляє збіг MAC-адреси призначення, яка міститься у заголовку, зі своєю власною адресою і направляє його модулю IP. Після аналізу полів заголовку IP-пакету із пакету вилучаються дані вище розташованих протоколів. DNS-запит передається програмному модулю DNS-сервера.

DNS-сервер переглядає свої таблиці, можливо, звертається до інших DNS-серверів і формує відповідь, сенс якої наступний: доменному імені ftp.lst.org відповідає IP-адреса 53.10.121.20. Ця відповідь DNS-сервера передається через інтерфейси протоколів UDP, IP та Ethernet і надходить до модуля канального рівня, де інкапсулюється в кадр протоколу Ethernet.

Передача IP-пакету з відповіддю через об'єднану мережу в зворотному напрямку до робочої станції клієнта, яка ініціювала звернення, відбувається аналогічно вищеописаному: протоколи UDP, IP і Ethernet послідовно обробляють відповідь DNS-сервера інкапсулюючи її у відповідні блоки даних.

Останній етап цієї передачі, від маршрутизатора М3 до робочої станції клієнта наведено на рисунку 3.6.

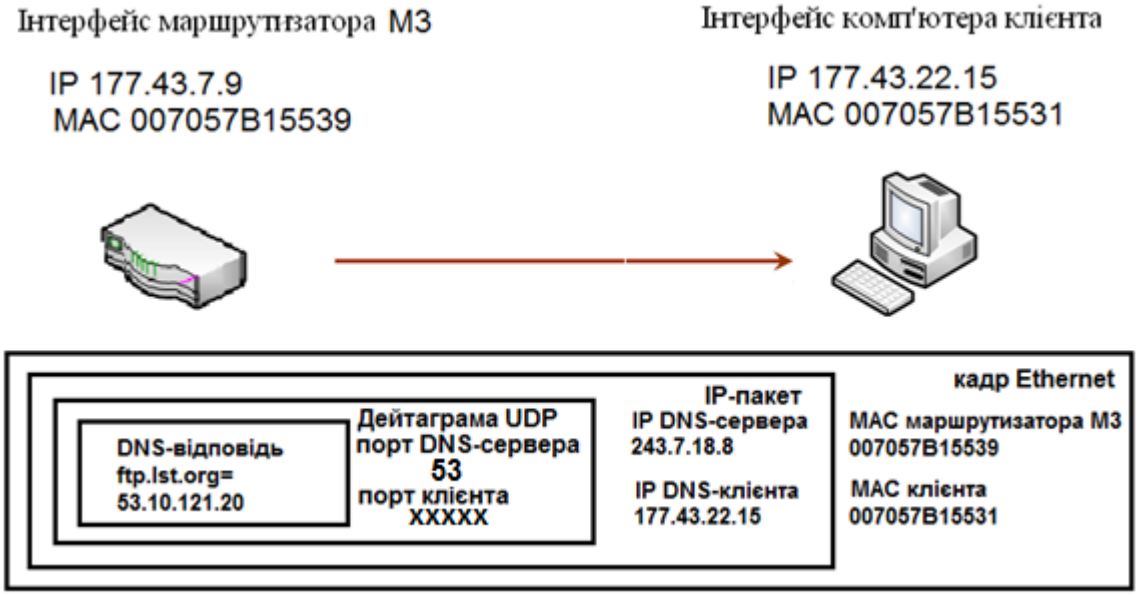


Рисунок 3.6 – Кадр Ethernet з DNS-відповіддю, відправлений з маршрутизатора М3 комп’ютеру клієнта

Передача пакету від FTP-сервера до FTP-клієнта

FTP-клієнт, отримавши IP-адресу FTP-сервера, надає йому своє повідомлення, використовуючи ті ж механізми доставки даних через об’єднану мережу, які розглянуті раніше.

1.2. Фрагментація IP-дейтаграм

Мережевий рівень зазвичай має обмеження максимального розміру кадру, який може бути переданий. Коли IP-рівень отримує IP-дейтаграму, яку необхідно відправити, він визначає, на який локальний інтерфейс відправляється (або маршрутизується) дейтаграма, і направляє запит в цей інтерфейс для визначення максимального розміру блоку даних MTU (Maximum Transmission Unit), який може бути переданий протоколом IP без фрагментації. Протокол IP порівнює MTU з розміром дейтаграми і, якщо необхідно, виконує фрагментацію. Дейтаграма може бути фрагментована хостом-відправником або будь-яким комунікаційним модулем на маршруті передачі (справедливо тільки для протоколу IP версії 4). При фрагментації IP-дейтаграми кожний отриманий фрагмент повинен мати свій заголовок, який збігається із заголовком первинної дейтаграми, і додаткові поля, які ідентифікують або порядковий номер отриманої дейтаграми, або значення її зміщення у нефрагментованій дейтаграмі. Кожний з отриманих блоків дейтаграми маршрутизується незалежно від інших, що призводить до того, що отримані фрагменти дейтаграми надійдуть в кінцевий пункт призначення не в тому порядку, в якому вони були відправлені. Збирається дейтаграмма **тільки** в модулі призначення

(для деяких інших мережевих протоколів повторна зборка виконується на маршрутизаторі наступної пересилки, а не в пункті призначення, що призводить до додаткових затримок в процесі передачі). Фрагменти фрагментованої дейтаграми можуть проходити через різні маршрутизатори, і неможливо ані керувати ними, ані гарантувати маршрут, яким вони передаються. Всі фрагменти, що належать одній і тій же дейтаграмі, повинні бути передані в хост-вузол призначення.

Фрагментація вимагає, щоб розмір утворюваних фрагментів (за винятком IP-заголовку) був кратним 8 байтам для всіх фрагментів за винятком останнього.

Якщо один фрагмент буде втрачено, вся дейтаграма повинна бути передана повторно, оскільки не існує способу повторної передачі тільки одного фрагменту дейтаграми, оскільки, якщо фрагментація була виконана проміжним комунікаційним вузлом, а не відправником, він не зможе визначити як дейтаграма була фрагментована в процесі передачі. Тому саме з цієї причини проміжної фрагментації намагаються уникати.

Протокол TCP на транспортному рівні намагається уникати фрагментації, і для сервісного додатку практично неможливо примусити TCP відправляти сегменти, які потребують фрагментації, тобто відправити сегмент такого розміру, для якого буде потрібна фрагментація, практично неможливо. При використанні протоколу UDP на транспортному рівні досягнути IP-фрагментації доволі легко.

Отже, при передачі дейтаграми із середовища з більшим значенням MTU в середовище з меншим MTU виникає необхідність у фрагментації дейтаграми. Для виконання фрагментації (і дефрагментації) використовуються поля «ID (Identification)», «Flags» і «FragmentOffset» IP-заголовку.

Ідентифікатор дейтаграми містить унікальний код дейтаграми, що визначає порядковий номер дейтаграми в послідовності. Це поле використовується приймаючим вузлом для об’єднання в єдине ціле фрагментів дейтаграми. Значення цього поля встановлюється відправником.

Флаги (рис. 3.7).

Біт 0 – зарезервований (завжди дорівнює 0).

Біт 1 виконує управління фрагментацією:

DF=0 - можна фрагментувати,

DF=1 - фрагментація заборонена.

Біт 2 показує чи є даний фрагмент останнім.

MF=0 - останній фрагмент,

MF=1 - є наступний фрагмент.

Рисунок 3.7 – Розміщення флагів в IP-дейтаграмі

Зміщення фрагменту показує, де у вихідній дейтаграмі знаходиться даний фрагмент. Величина зміщення задається в 64-бітних блоках. Перший фрагмент має нульове зміщення. Поле загальної довжини вказує довжину вихідного пакету, а поле зміщення показує вузлу, що здійснює збирання, зміщення відносно його початку.

Розглянемо механізм фрагментації на маршрутизаторі на прикладі об'єднаної мережі, структура якої наведена на рисунку 3.8.

В одній із підмереж (Frame Relay) значення MTU дорівнює 4080, в іншій (Ethernet) — 1500. Хост-вузол, який належить мережі Frame Relay, передає дані хосту-вузлу в мережі Ethernet. На обох хостах, а також на маршрутизаторі, який зв'язує ці мережі, встановлений стек протоколів TCP/IP.

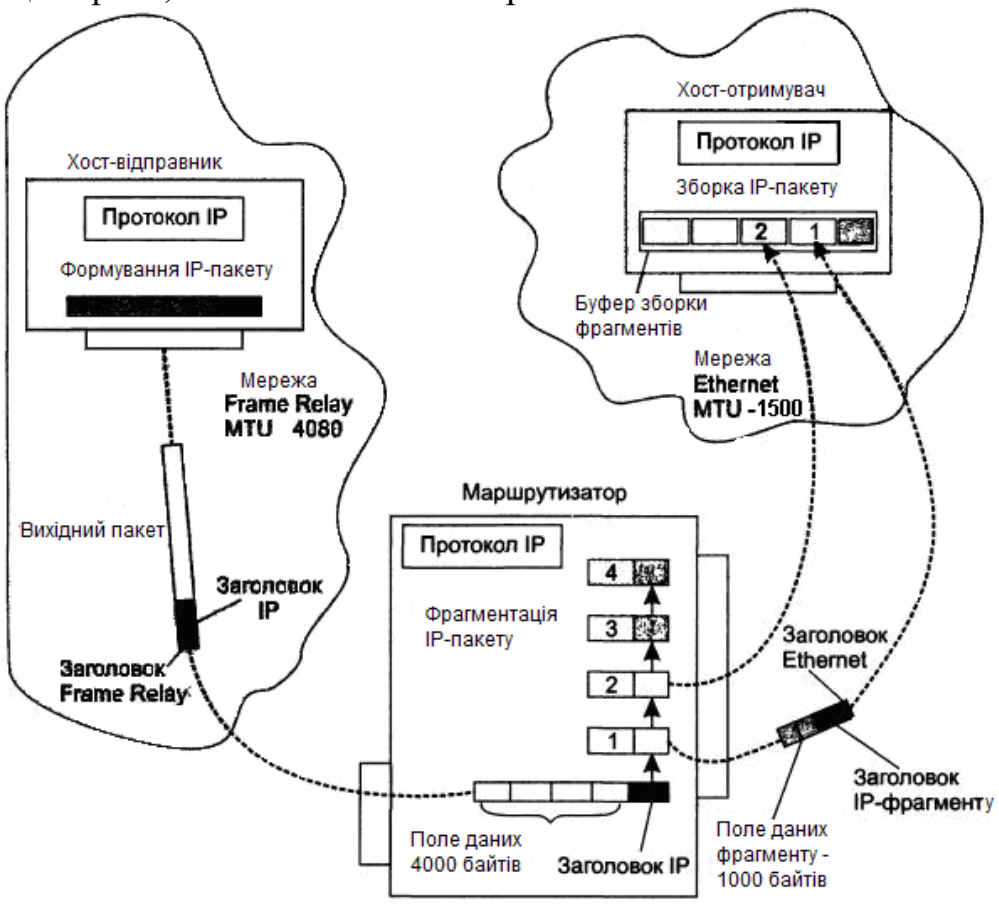


Рисунок 3.8 – Структура об'єднаної мережі

Транспортному рівню хосту-відправника відоме значення MTU розташованої нижче мережі (4080). Враховуючи таке обмеження, модуль TCP розділяє потік даних на сегменти розміром 4000 байтів і передає через інтерфейс протоколу IP, який інкапсулює сегменти в поле даних IP-дейтаграм і генерує для них заголовки. Поля заголовку, які безпосередньо пов'язані з фрагментацією, наступні:

- ідентифікатор: всім фрагментам одного повідомлення присвоюється унікальний ідентифікатор, наприклад, 12456;

- зміщення фрагменту: оскільки пакет поки що не був фрагментований, полю зміщення присвоюється значення 0;
- флаг: поле MF також обнуляється, це показує, що пакет одночасно є і своїм останнім фрагментом;
- флаг: поле DF встановлюється в 0, це означає, що даний пакет можна фрагментувати.

Загальна величина IP-пакету складає 4000 байтів плюс 20 байтів (розмір заголовку IP), тобто 4020 байтів, це менше значення MTU кадру Frame Relay, яке в даному прикладі дорівнює 4080. Далі модуль IP хосту-відправника передає цей кадр своєму мережевому інтерфейсу Frame Relay, який відправляє кадри наступному маршрутизатору.

До кожного із сегментів має бути приєднаний повноцінний IP-заголовок, а також деякі поля (ідентификатор, TTL, флаги DF і MF, зміщення), що безпосередньо призначені для наступної зборки фрагментів в початкове повідомлення.

Модуль IP маршрутизатора, аналізуючи мережеву адресу прийнятої IP-дейтаграми визначає, що пакет потрібно передати в мережу Ethernet, яка має значення MTU, що дорівнює 1500, що значно менше розміру пакета, який надійшов на вхідний інтерфейс. Тому IP-дейтаграму необхідно фрагментувати. Модуль IP вибирає розмір поля даних фрагменту у 1000 байт, так що із одного великого IP-пакету утворюється 4 менших фрагментів-дейтаграм. Для кожного фрагменту і його IP-заголовку в маршрутизаторі створюється окремий буфер (на рисунку фрагменти і відповідні їм буфери пронумеровані від 1 до 4).

В процесі фрагментації можуть змінитися значення деяких полів IP-заголовків в отриманих фрагментах дейтаграм, а саме контрольної суми заголовку, зміщення фрагменту, довжини дейтаграми, значення деяких флагів. Отримані фрагменти дейтаграм мають довжину 1020 байтів (з урахуванням IP-заголовку), тому вони інкапсулюються в поле даних фрейму.

На рисунку 3.8 показані різні стадії переміщення фрагментів по мережі. Фрагмент 2 вже надійшов до хосту-отримувача і зберігається в буфері. Фрагмент 1 ще рухається по мережі Ethernet, решта фрагментів знаходяться в буферах маршрутизатора.

Розглянемо, як відбувається формування первинної дейтаграми з окремих фрагментованих пакетів на хості призначення.

На хості призначення для кожного фрагментованого пакету відводиться окремий буфер, в який записуються всі фрагменти одного повідомлення (потоку), що мають однакові IP-адреси відправника і отримувача та ідентифікатор (для розглянутого прикладу 12345). Зборка полягає в розташуванні даних із кожного фрагменту в тому місці виділеного буфера, який визначений зміщенням фрагменту і вказаний у його заголовку.

Коли перший фрагмент вихідного пакету приходить на хост-отримувач, цей хост запускає таймер, який визначає максимальний час очікування

прибуття решти фрагментів даного пакету. В різних реалізаціях IP застосовуються різні правила вибору максимального часу очікування. Зокрема, таймер може бути встановлений на фіксований період часу (від 60 до 120 секунд), рекомендований RFC. Як правило, цей інтервал достатній для доставки пакета від відправника отримувачу. В інших реалізаціях максимальний час очікування визначається за допомогою адаптивних алгоритмів вимірювання і статистичної обробки часових параметрів мережі, які дозволяють оцінювати очікуваний час прибуття фрагментів. Нарешті, тайм-аут може бути вибраний на основі значень TTL отримуваних фрагментів. Останній підхід базується на тому, що немає сенсу чекати надходження інших фрагментів пакету, якщо час життя одного із отриманих фрагментів вже вичерпано.

1.3. Рівень мережевого інтерфейсу

Рівень мережевого інтерфейсу відповідає фізичному і канальному рівням моделі OSI і відповідає за встановлення мережевого з'єднання в конкретній фізичній мережі. Інтерфейс із мережею може бути реалізований драйвером пристрою або системою (комутатор, маршрутизатор), що використовує свій протокол канального рівня.

У стеку протоколів TCP/IP цей рівень не регламентований, але він підтримує всі популярні стандарти канального і фізичного рівнів: Ethernet, Token Ring, FDDI, Fast Ethernet, Gigabit Ethernet тощо. Для територіально розподілених мереж зазвичай використовуються протоколи PPP (Point to Point Protocol) і SLIP (Simple Line Internet Protocol), а для глобальних мереж — протоколи X.25 і Frame Relay. Зазвичай з появою нової технології локальних або глобальних мереж вона швидко включається в стек TCP/IP за рахунок розробки відповідного стандарту RFC, що визначає метод інкапсуляції IP-дейтаграм у її кадри.

В лабораторній роботі розглядаються складові гібридної моделі TCP/IP: канальний та фізичний рівні.

Протокол IP на мережевому рівні діє незалежно від середовища, яке використовується для передачі даних на нижніх рівнях стеку протоколів. Модуль канального рівня повинен прийняти IP-дейтаграму і підготувати її для передачі в комунікаційному середовищі. При цьому необхідно враховувати максимальний розмір фрейму MTU, який може передаватися в канальному середовищі. Канальний рівень передає значення MTU до мережевого рівня. Використовуючи надане значення MTU, мережевий рівень визначає розмір пакетів

Канальний рівень. Заголовок кадру

Канальний рівень (Data link layer) визначає правила доступу до фізичного середовища та управляє передачею інформації в каналі, здійснюючи формування сигналу про початок передачі й організовуючи початок і власне передачу інформації зі створенням сигналу закінчення передачі і наступним переведенням каналу в пасивний стан. Обов’язковою функцією будь-якого

протоколу канального рівня є перевірка коректності прийнятої інформації та виправлення виникаючих помилок або формування службових повідомлень про повторну передачу блоків даних, які містили помилки.

На фізичному рівні пересилаються біти і при цьому не враховується, що фізичне середовище передачі може бути зайняте. Тому одним із завдань канального рівня є перевірка доступності середовища передачі.

Таким чином, канальний рівень забезпечує створення, передачу та прийом інформаційних блоків, що називаються кадрами даних, обслуговуючи запити мережевого рівня і використовуючи для передачі та прийому кадрів сервіс фізичного рівня. Спочатку цей рівень був створений як функціонально єдиний рівень, що розв'язує завдання:

- при передачі — власне передачу кадру даних з мережевого рівня на фізичний рівень і забезпечення безпомилкової передачі по фізичному рівню кадрів із однієї системи на іншу;
- при прийомі —розподіл незмонтованих бітів з фізичного рівня в кадри для більш високих рівнів.

На сьогодні функції канального рівня, як правило, реалізуються програмно-апаратно.

Формат заголовку кадру

При передачі даних на канальному рівні раніше створена дейтаграма інкапсулюється в кадр, тобто додається ще заголовок кадру та кінцевик.

Існує декілька стандартів формату кадру Ethernet: стандарт Ethernet II, IEEE 802.2 (802.3+802.2), IEEE 802.3 і IEEE SNAP. На практиці в обладнанні Ethernet використовується тільки один формат кадру, а саме кадр Ethernet DIX (рис. 3.9).

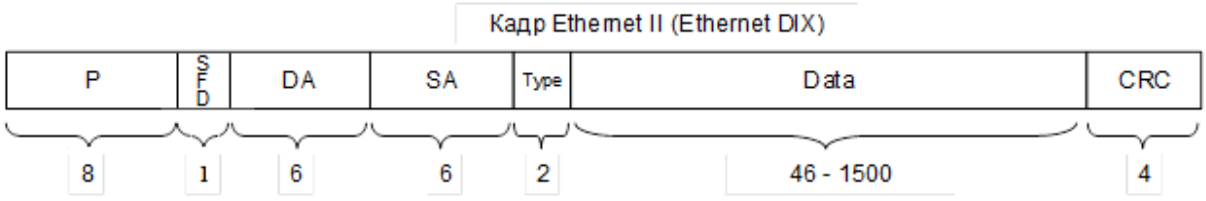


Рисунок 3.9 – Формат кадру Ethernet II

P (*Preamble*) – преамбула, використовується для синхронізації і складається з семи (або восьми) байт, які мають значення 10101010;

SFD (*Start of Frame Delimiter*) – роздільник початку кадру (значення 10101011) вказує на те, що наступний байт є байтом заголовку кадру;

DA (*Destination Address*) – адреса призначення (6-байтна MAC-адреса отримувача);

SA (***Source Address***) –-адреса відправника (6-байтна MAC-адреса відправника,завжди unicast-адреса);

Type – поле визначає який протокол верхнього рівня передає свої дані в кадрі (виконує функції полів DSAP и SSAP з заголовку LLC);

Data – містить дані, що передаються протоколом верхнього (мережевого) рівня;

CRC (*Cyclic Redundancy Check*) – контрольна послідовність кадру містить контрольну суму кадру, яка обчислена за алгоритмом CRC-32.

В полі даних інкапсульовані блоки даних протоколів більш високого рівня, наприклад, TCP/UDP і IP. Приклад фрагменту фрейму з інкапсульованим в полі даних TCP- і IP-заголовками представлено на рисунку 3.10.

Стандарти Ethernet II і IEEE 802.3 визначають мінімальну довжину кадру в 64 байти по стандарту Fast Ethernet 100Base-T і 512 байтів по стандарту Gigabit Ethernet 1000Base-T, а максимальну — 1500 байт. В цю довжину входять всі байти, починаючи з поля «MAC-адреса призначення» і закінчуючи полем «Контрольна послідовність кадру». Поля «Преамбула» и «Початок обмежувач кадру (SFD)» в розмір кадру не включаються.

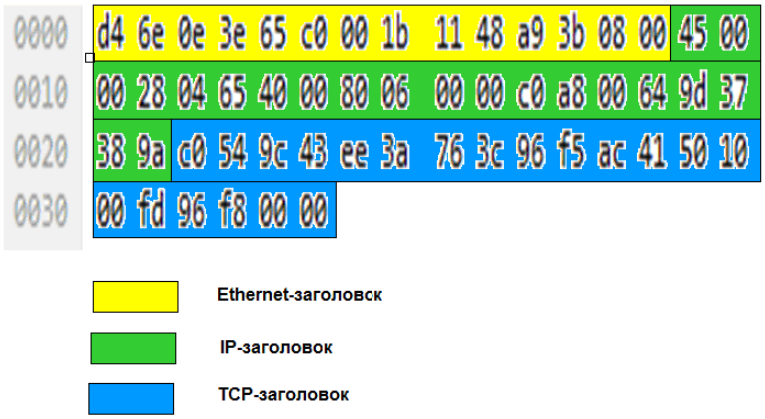


Рисунок 3.10 – Фрагмент кадру з інкапсульованими в полі даних TCP- і IP-заголовками

Якщо величина кадру, що передається, менше мінімального значення або більше максимального значення, пристрій, що отримує кадр, відкидає його.

Фізичний рівень

Фізичний рівень виконує сервісні функції для канального рівня. Його призначенням є забезпечення механічних, електричних, функціональних і процедурних засобів для встановлення, підтримки і роз'єднання фізичних з'єднань з метою передачі послідовностей бітів між об'єктами канального рівня.

Функції цього рівня реалізуються у всіх пристроях, під'єднаних до мережі, незважаючи на те, що інформація, яка передана по фізичному середовищу, на цьому рівні не перетворюється. Фактично даний рівень пов'язаний з фізичним доступом до мережі методом прийому і передачі інформації, здійснюючи, наприклад, прийом даних від розташованого вище канального рівня, перетворення цих даних в електричні, оптичні або радіосигнали і направлення останніх через середовище передачі на прийомний вузол, забезпечуючи:

- кодування інформації й синхронізацію бітових послідовностей, гарантуючи необхідну достовірність прийому/передачі;
- перетворення електричного, оптичного, механічного та функціонального інтерфейсів мережевого кабелю для передачі по ньому (через фізичне середовище) неструктурованих бітових потоків.

Основними функціями, які виконуються фізичним рівнем, є:

- встановлення і роз'єднання фізичних з'єднань між об'єктами мережі;
- передача послідовностей бітів у синхронному або асинхронному режимі;
- управління фізичним рівнем.

На фізичному рівні визначаються специфікації на механічні, електричні й інші властивості середовища передачі, сигнали, обладнання, з'єднання, параметри кабелів і з'єднувачів. Цього рівня стосуються характеристики фізичних середовищ передачі даних, такі як смуга пропускання, завадозахищеність, хвильовий опір та інші. На цьому ж рівні визначаються характеристики електричних сигналів, що передають дискретну інформацію, таку як крутість фронтів імпульсів, рівні напруги або струму сигналу, що передається, тип кодування, швидкість передачі сигналів. Окрім того, тут стандартизуються типи з'єднувачів і призначення кожного контакту.

Дискретний канал зв'язку захищений від впливу завад тільки потенційною завадостійкістю переданих сигналів (безперервних або дискретних). Оскільки на фізичному рівні не розв'язується задача виправлення спотворених бітів, його слід вважати ненадійною системою передачі. Функції фізичного рівня реалізуються, як правило, апаратно через мережевий адаптер або послідовний порт у всіх пристроях, під'єднаних до мережі.

Мережеві адаптери виконують наступні основні операції при прийомі або передачі повідомлення.

1. Гальванічну розв'язку з коаксіальним кабелем або витою парою.
2. Прийом (передачу) даних.
3. Буферизацію.
4. Формування пакету.
5. Доступ до каналу зв'язку.
6. Ідентифікацію своєї адреси в пакеті, який приймається.
7. Перетворення паралельного коду у послідовний код при передачі даних, і із послідовного коду в паралельний при прийомі.
8. Кодування і декодування даних.
9. Передача або прийом імпульсів.

Мережеві адаптери разом із мережевим програмним забезпеченням здатні розпізнавати і опрацьовувати помилки, які можуть виникати із-за електричних перешкод, колізій або некоректної роботи обладнання.

Завдання

1. В лабораторній роботі проводиться дослідження виконання фрагментації на мережевому рівні стеку TCP/IP. При виконанні роботи використовується програмне забезпечення для аналізу протоколів комп'ютерних мереж Wireshark.
2. Визначіть значення максимального розміру пакету MTU, який може бути переданий канальним рівнем без фрагментації на тому інтерфейсі Вашого комп'ютера, на якому буде відбуватися захоплення пакетів програмою Wireshark.

   У Windows для цього можна скористатися командою із командного рядка

   netsh interface ipv4 show subinterfaces,

   а в Unix про значення MTU можна дізнатися за допомогою команди

   ifconfig

   В мережах типу Ethernet значення MTU зазвичай дорівнює 1500 байтів.
3. Запустіть програму Wireshark. Виберіть інтерфейс для захоплення трафіку (меню Capture/Interface) та активізуйте режим захоплення.
4. Перейдіть в командний рядок і виконайте команду **ping,** вказавши цільову IP-адресу, наприклад, Вашого комп'ютера і параметр **-l xxxx,** де значення xxxx має перевищувати значення MTU, щоб була виконана фрагментація (наприклад, 5000).
5. Після захоплення трафіку, який виник в результаті виконання команди **ping,** зупиніть захоплення програмою Wireshark. Проведіть аналіз пакетів, що утворилися. Зверніть увагу на процес фрагментації IP-дейтаграм, що відбувся, та на величину блоку корисного навантаження у фрагментованих пакетах.
6. Результати захоплення фрагментованих пакетів занесіть у звіт.

Контрольні запитання

1. Поясніть процедуру фрагментації IP-дейтаграм? За яких умов фрагментація відбувається?
2. Які поля IP-заголовку застосовуються при виконанні фрагментації та дефрагментації?
3. Чому при плануванні мережевої взаємодії потрібно намагатися уникати фрагментації?
4. Назвіть основні функції рівня мережевих інтерфейсів.
5. Поясність процес інкапсуляції.
6. Назвіть основні функції мережевого адаптера.

Лабораторна робота 4
Адресація в TCP/IP-мережах. Багатоадресне розсилання

Мета роботи: ознайомлення та засвоєння типів адрес, які використовуються для ідентифікації хост-модулів комп'ютерних мереж, структури IP-адреси, особливостей багатоадресного розсилання.

План виконання лабораторної роботи

1. Ознайомитися з теоретичними відомостями, що викладені в методичному посібнику до лабораторної роботи та засвоїти їх.
2. Виконати завдання до лабораторної роботи.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

Стек протоколів TCP/IP призначений для об'єднання окремих підмереж, побудованих за різними технологіями канального та фізичного рівнів (Ethernet, Token Ring, FDDI, ATM, X.25 і т.д.) в одну об'єднану мережу. Кожна з технологій нижнього рівня має свою схему адресації. Тому на міжмережевому рівні потрібний один спосіб адресації, який дозволяє унікально ідентифікувати кожний вузол об'єднаної мережі. Таким способом в TCP/IP-мережах є IP-адресація.

Вузол об'єднаної мережі, який має IP-адресу, називається хостом (host).

В стеку TCP/IP протоколами різних рівнів використовуються наступні три типи адрес:

- символьні (доменні) імена;
- IP-адреси (логічні, мережеві);
- локальні (фізичні, апаратні).

Символьні доменні імена (domain name) застосовуються для зручності представлення IP-адрес. Людині незручно запам'ятовувати числові IP-адреси, тому була розроблена спеціальна служба DNS (Domain Name System), яка встановлює відповідність між IP-адресами і символьними доменними іменами.

IP-адреси (IP address) – це основний тип адрес, користуючись яким мережевий рівень передає повідомлення, які називаються IP-дейтаграмами (IP-пакетами). Ці адреси мають довжину 4 байти (для протоколу IPv4), які записуються в десятковому вигляді і значення байтів розділяються крапками, наприклад, 117.52.9.44. IP-адреса вузла в протоколі IP призначається незалежно від локальної адреси вузла. Маршрутизатор зазвичай входить відразу в кілька мереж. Тому кожен порт маршрутизатора має власну IP-адресу. Кінцевий вузол також може входити в декілька IP-мереж. У цьому випадку робоча станція повинна мати кілька IP-адрес, за кількістю мережевих адаптерів. Таким чином, IP-адреса характеризує не окремий хост-вузол або маршрутизатор, а частину адреси одного мережевого з'єднання.

Локальна (фізична) адреса – це адреса, присвоєна вузлу відповідно до технології підмережі, яка входить в об'єднану мережу. Якщо мережею або

сегментом є локальна мережа Ethernet, Token Ring або FDDI, то локальна адреса – це MAC-адреса (Media Access Control address).

MAC-адреси призначаються мережевим адаптерам і портам маршрутизаторів виробниками обладнання і є унікальними, оскільки розподіляються централізовано. MAC-адреса має, зазвичай, довжину 6 байтів і записується в шістнадцятковому вигляді, наприклад, 00-08-A0-12-5F-72 або 00:08:A0:12:5F:72.

1.1. Структура IP-адреси

IP-адреса – це 32-розрядне двійкове число, розділене на групи по 8 біт, які називаються октетами, наприклад:
00010001 11101111 00101111 01011110

Зазвичай IP-адреси записуються у вигляді чотирьох десяткових октетів, які розділяються крапками. Таким чином, приведену вище IP-адресу можна записати у такій формі: 17.239.47.94.

Слід зауважити, що максимальне значення октету дорівнює (у двійковій системі) 11111111, що відповідає числу 255 в десятковій системі. Тому IP-адреси, в яких хоча б один октет перевищує це число, є недійсними.

Приклад: 172.16.123.1 – допустима адреса, 172.16.123.256 – недійсна адреса, оскільки 256 виходить за межі допустимого діапазону. IP-адреса складається із двох логічних частин – номеру (ідентифікатору) мережі і номеру (ідентифікатору) вузла в цій мережі. При передачі пакету із однієї мережі в іншу використовується тільки та частина IP-адреси, яка містить номер мережі. Коли пакет надходить в мережу призначення, адреса вузла вказує на конкретний вузол в цій мережі.

Щоб записати ID мережі, в поле номера вузла в IP-адресі записують нулі. Щоб записати ID хосту, в поле номера мережі записують нулі. Наприклад, якщо в IP-адресі 172.16.123.1 перші два байти відводяться під номер мережі, останні два байти – під номер вузла, то номери записуються наступним чином:
ID підмережі: 172.16.0.0. ID хосту: 0.0.123.1.

По кількості розрядів, які відводяться для номера вузла (або номера мережі), можна визначити загальну кількість вузлів (або мереж): якщо число розрядів для номера вузла дорівнює N, то загальна кількість дорівнює $2^N - 2$. Два вузли віднімаються внаслідок того, що адреси з усіма розрядами, що дорівнюють нулю або одиниці, є особливими і використовуються зі спеціальною метою.

Наприклад, якщо під номер вузла в деякій підмережі відводиться два байти, то загальна кількість вузлів в такій підмережі дорівнює $2^{16} - 2 = 65534$ вузли.

Для визначення того, яка частина IP-адреси відповідає за ID мережі, а яка за ID хосту, використовуються два способи: за допомогою класів і за допомогою масок. Загальне правило: під ID мережі відводяться перші кілька бітів IP-адреси, біти, які залишилися, визначають ID хосту.

Існує п’ять класів IP-адрес: A, B, C, D і E (рис. 4.1). За приналежність до того чи іншого класу відповідають перші біти IP-адреси, які часто називають ключем, який містить від 1 до 5 бітів. Результатом такого розподілу стало створення невеликої кількості великих мереж (класу A) з дуже великою кількістю хост-вузлів, помірної кількості середніх мереж (клас B) зі значною кількістю кінцевих вузлів, і великої кількості мереж (клас C) з невеликою кількістю хост-вузлів. Три з них (A, B, C) використовуються для адресації мереж, а два (D, E) мають спеціальне призначення.

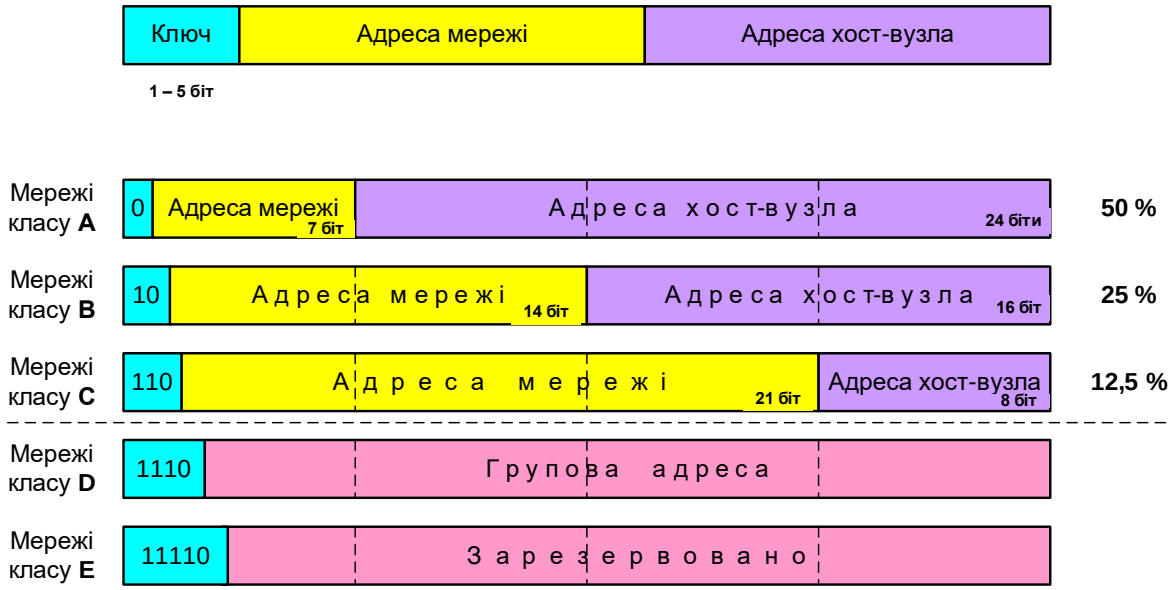


Рисунок 4.1 – Класи IP-адрес

В таблиці 4.1 представлено основні показники мереж всіх класів. Повний діапазон адрес, які можуть мати хост-вузли, що підключаються до мереж відповідного класу, наступні:

клас A 1.0.0.1 – 126.255.255.254,
клас B 128.1.0.1 – 191.254.255.254,
клас C 192.0.1.1 – 223.255.254.254.

Таблиця 4.1 – Характеристика IP-адрес різних класів

Клас	Формат	Шаблон перших біт (ключ) першого октету	Десяткові значення першого байту адреси мережі	Максимальна кількість мереж	Максимальна кількість вузлів в одній мережі
A	Net.Node.Node.Node	0	1 - 126	126 (2^7-2)	16777214 (2^{24}-2)
B	Net.Net.Node.Node	10	128 – 191	16382 (2^{14}-2)	65532 (2^{16}-2)
C	Net.Net.Net.Node	110	192 – 223	2097150 (2^{21}-2)	254 (2^8-2)
D	-	1110	224 – 239	-	-
E	-	11110	240 - 247	-	-

Протокол IP підтримує три способи адресації:

- індивідуальну (unicast);
- групову (multicast);
- широкомовну (broadcast).

При **одиничній адресації** дейтаграми передаються конкретному одиничному пристрою.

При **широкомовній адресації** повідомлення надсилають одну дейтаграму, що доставляється всім пристроям мережі. Якщо широкомовний трафік призначений тільки для вузлів локальної мережі, то його реалізація не викликає особливих ускладнень. Якщо ж повідомлення необхідно передати в іншу мережу, то в цьому випадку глобальна мережа повинна мати значну пропускну спроможність.

При **груповій адресації** дейтаграмми доставляються певній групі пристроїв. При цьому не генерується надлишковий трафік, що особливо важливо при роботі в розподілених мережах. Дейтаграми із груповою та одиничною адресою розрізняються адресою. У заголовку IP-дейтаграми із груповою адресацією замість IP-адрес класів A, B, C записується адреса класу D, тобто групова адреса. **Групова адреса** призначається деяким пристроям-одержувачам (групі пристроїв). Відправник записує дану групову адресу в заголовок IP-дейтаграми. Дейтаграма буде доставлена всім членам групи. Перші чотири біти адреси класу D дорівнюють 1110, іншу частину адреси (28 біт) займає ідентифікатор групи.

Групові адреси лежать у діапазоні від 224.0.0.0 до 239.255.255.255. Наприклад, групові адреси, що використовуються в Інтернет, лежать в діапазоні 224.0.1.0 - 238.255.255.255, локальні групові адреси – в діапазоні 239.0.0.0 -239.255.255.255, а зарезервовані групові адреси лежать в діапазоні 224.0.0.0 - 224.0.0.255. Зокрема, цей останній діапазон відведений протоколам маршрутизації та іншим низькорівневим протоколам. У таблиці 4.2 наведені деякі зарезервовані IP-адреси класу D.

В локальних мережах LAN (Local Area Network) використовуються **приватні IP-адреси** (*private IP address*), які називають також ***внутрішніми, внутрішньомережними, локальними або «сірими».*** Це IP-адреси, які належать до спеціального діапазону, що не використовується в Internet. Такі адреси використовуються в локальних мережах, і їх розподілення ніким не контролюється. До таких адрес відносяться:

10.0.0.1 – 10.255.255.254 (маска підмережі при безкласовій адресації 255.0.0.0 або префікс */8);

172.16.0.1 – 172.31.255.254 (маска підмережі при безкласовій адресації 255.240.0.0 або префікс */12);

192.168.0.1 – 192.168.255.254 (маска підмережі при безкласовій адресації 255.255.0.0 або префікс */16).

На сьогодні класова адресація «в чистому вигляді» не використовується, оскільки призводить до неефективного розподілу адресного простору, суттєвого збільшення розміру таблиць маршрутизації і, як наслідок, збільшення часу доставки повідомлень. Для вирішення цих проблем використовують структуризацію мереж, тобто виділення підмереж. При цьому в деякій мережі будь-якого класу виділяються окремі області, підмережі, які повинні мати свої ідентифікатори (адреси).

Таблиця 4.2 - Зарезервовані адреси класу D

Адреса	Призначення
224.0. 0.0	Базова адреса мультикасту (зарезервована)
224.0. 0.1	Всі пристрої підмережі (даного сегменту мережі)
224.0. 0.2	Всі маршрутизатори підмережі
224.0. 0.4	Всі DVMRP-маршрутизатори
224.0. 0.5	Всі MOSPF-маршрутизатори сегменту мережі
224.0. 0.6	Всі DR-маршрутизатори сегменту мережі
224.0. 0.7	Аудіоновини
224.0. 0.9	RIP версії II
224.0. 0.10	Всі EIGRP-маршрутизатори сегменту мережі
224.0. 0.11	IEFT аудіо
224.0. 0.12	IEFT відео
224.0. 0.13	Всі PIM-маршрутизатори сегменту мережі
224.0. 0.22	Всі IGMP-маршрутизатори сегменту мережі
224.0. 0.251	Мультикастова адреса DNS (mDNS)

1.1. **Виділення підмереж. Поняття маски**

Якщо разом з IP-адресою використовувати маску, то це дозволить виконувати адресацію більш гнучко і відмовитися від класової адресації. При цьому частину мережі називають підмережею. Для виділення підмережі маршрутизатору необхідно «накласти» маску підмережі на IP-адресу, що дозволить виділити в адресі ідентифікатори мережі, підмережі та кінцевого хост-вузла.

Виділення (створення) підмереж – це властивість програмного забезпечення IP створювати декілька нових підмереж з однієї великої мережі. В результаті організація має одну адресу і створює необхідну їй структуру для кожної фізичної мережі (сегменту). Для ідентифікації підмережі використовується старша частина IP-адреси, що призначена для адресації хост-вузлів (рис. 4.2).

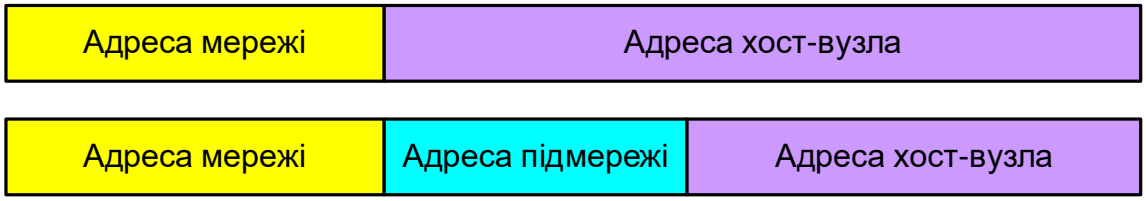


Рисунок 4.2 – Ідентифікація підмереж

Маска мережі та підмережі використовується для визначення того, які біти в IP-адреси визначають ідентифікатор мережі та підмережі, а які – ідентифікатор хост-вузла.

Маска являє собою 32-бітне слово, яке складається з послідовності «1», за якою йде послідовність «0»:

- «1» в масці визначають біти IP-адреси, в яких міститься ідентифікатор мережі та підмережі;
- «0» в масці визначають біти IP-адреси, в яких записаний ідентифікатор хост-вузла.

Наприклад, нехай для IP-адреси 172.16.155.5 задано маску 255.255.192.0. Двійковий вид відповідно такий:

- IP-адреса 10101100.00010000.10011011.00000101;
- маска 11111111.11111111.11000000.00000000.

Якщо ігнорувати маску й інтерпретувати адресу 172.16.155.5 на основі класів, то ідентифікатор мережі є 172.16.0.0, а вузла – 0.0.155.5 (оскільки адреса відноситься до класу B).

Якщо використовувати маску, то 18 послідовних одиниць в масці 255.255.192.0 після «накладання» на IP-адресу 172.16.155.5 ділять її на дві частини:

- ідентифікатор мережі 10101100.00010000.10;
- ідентифікатор вузла 011011.00000101.

У десятковій формі запис номера мережі і вузла після доповнення до 32 біт мають відповідно такий вигляд: 172.16.128.0 і 0.0.27.5.

Виділення з допомогою маски адреси мережі і хоста можна інтерпретувати як виконання логічної операції I (AND).

Для запису масок використовуються також такі формати запису з використанням префіксу, наприклад, 172.16.155.5/18 – 18 означає префікс, що вказує на кількість одиниць на початку маски.

1.2. **Загальні відомості про Multicast**

Існують такі типи мережевого трафіку (зауважимо, що не обов’язково використовуються всі чотири типи трафіку конкретною версією протоколу):

- **Unicast** - одноадресна розсилка - один відправник, один отримувач (наприклад, запит HTTP-сторінки у вебсервера);
- **Broadcast** - широкомовна розсилка - один відправник, отримувачі - всі пристрої в широкомовному сегменті (наприклад, ARP-запит);
- **Multicast** - багатоадресна розсилка - один відправник, багато отримувачів. (наприклад: IPTV);
- **Anycast** - одноадресна розсилка найближчому вузлу - один відправник, взагалі отримувачів багато, але фактично дані відправляються тільки одному (наприклад, Anycast DNS).

Основним принципом мультикастових розсилок є те, що відправник відправляє тільки одну копію трафіку, незалежно від кількості отримувачів, а трафік (потік даних) отримують тільки ті кінцеві системи, які дійсно зацікавлені в ньому.

Основне завдання групових запитів полягає в зменшенні навантаження на хости-вузли, які не приймають участі в роботі певного прикладного сервісу. При багатоадресному розсиланні відправник формує та відправляє один потік даних загальним маршрутом тим отримувачам, які підписані на цю розсилку. Перевага такого підходу полягає в тому, що додавання нових користувачів не вимагає збільшення пропускної здатності мережі загальним маршрутом до користувачів послуги, і відповідно, зменшується навантаження на проміжні модулі. При запуску на сервері застосування (додатку) з підтримкою групового розсилання, воно посилає в мережу повідомлення про те, що відповідна група доступна для приєднання. Клієнт, який бажає приєднатися до розсилки, посилає повідомлення про це. Всі проміжні маршрутизатори запам’ятовують, що за відповідним маршрутом знаходиться клієнт відповідної мультикастової групи.

Хост може належати одній або кільком групам. Якщо це можливо, мережева плата повідомляє, до якої групи належить хост, після чого мережева плата приймає тільки фрейми певної групи

Щоб технологія працювала, вона має підтримуватися сервером, клієнтом і усіма проміжними маршрутизаторами. Щоб комутатори направляли пакети тільки потрібним отримувачам, вони мають підтримувати IGMP snooping, в іншому випадку пакети розсилаються широкомовно. Також потрібно пам’ятати про те, що мультикаст може блокуватися міжмережевими екранами.

Недоліком групової розсилки є неможливість використання на транспортному рівні протоколу TCP. Використання протоколу UDP тягне за собою всі його недоліки: ненадійність доставки, відсутність засобів реагування на затори в мережі і т.д. Крім того, в окремих випадках при зміні маршрутів розсилки групові дейтаграми можуть втрачатися і дублюватися, і це має враховуватися застосуваннями.

Побудова об’єднаної мережі з підтримкою мультикастингу є набагато складнішим завданням, ніж організація групової розсилки в межах однієї мережі.

Якщо члени мультикастової групи знаходяться в різних мережах для мультикастингу потрібна спеціальна власна маршрутизація. Тому для маршрутизації групових дейтаграм були розроблені спеціальні методи і протоколи маршрутизації.

Відзначимо, що не всі провайдери Internet підтримують багатоадресне з'єднання.

Для використання мультикастингу, він має підтримуватися в стеку TCP/IP сервера, клієнтів і проміжних маршрутизаторів. В локальних мережах за управління мультикаст-групами відповідає протокол IGMP (Internet Group Management Protocol), в глобальній мережі – протокол PIM (Protocol Independent Multicast).

Мультикастинг відбувається на канальному і мережевому рівнях. Для мультикастових груп зарезервовані адреси як на канальному, так і на мережевому рівнях.

В IPv4 для багатоадресної розсилки зарезервований діапазон адрес від 224.0.0.0 до 239.255.255.255(224.0.0.0/4), в IPv6 – ff00::/8. Адреси з цього діапазону можуть використовуватися тільки як адреси призначення, вони ніколи не можуть бути використані як адреси відправників.

Для визначення широкомовної MAC-адреси призначення в Ethernet-кадрах необхідно виконати додаткові кроки. При доставці unicast-повідомлень трафік направляється на унікальну MAC-адресу вузла призначення. Маршрутизатор дізнається про його MAC-адресу за допомогою протоколу визначення адрес ARP.

Для multicast-трафіка механізм визначення MAC-адрес модулів призначення інший. MAC-адреса багатоадресної розсилки – це особливе значення, яке в шістнадцятковому форматі починається з 01-00-5E (24 біти), а наступний 25-й біт повинен дорівнювати 0. Решта MAC-адреси багатоадресної розсилки створюється шляхом перетворення молодших 23 бітів IP-адреси групи багатоадресної розсилки в 6 шістнадцяткових (рис. 4.3, на якому показано відповідність IP-адреси 231.205.98.177 на мережевому рівні MAC-адресі 01:00:5E:4D:62:B1 на рівні Ethernet).

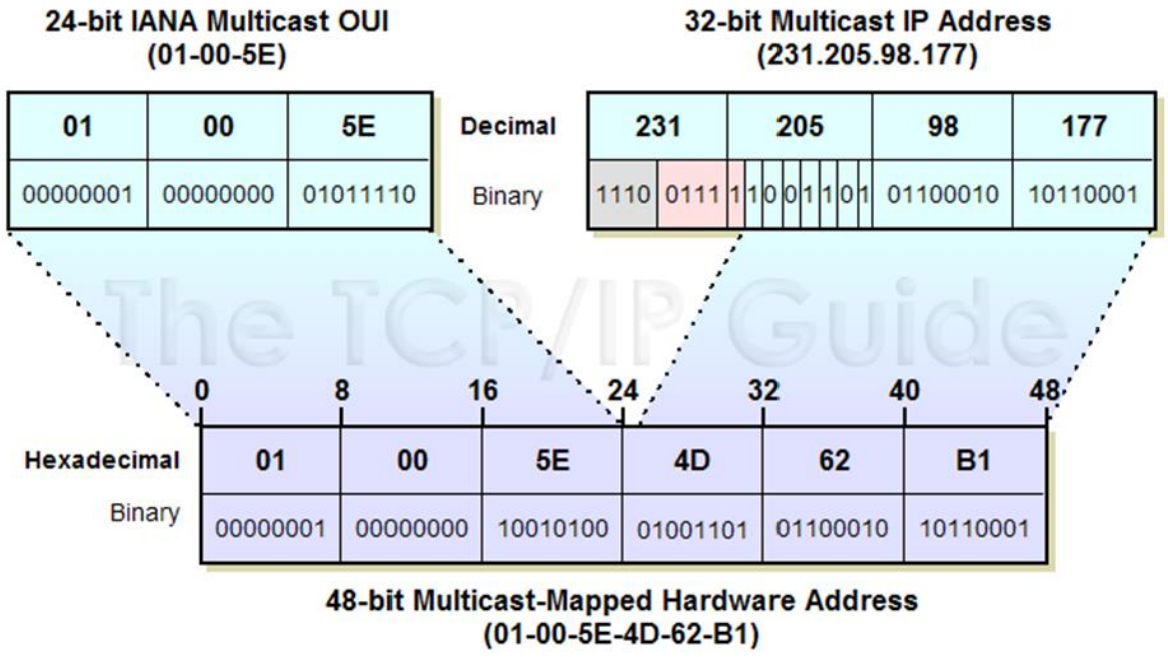


Рисунок 4.3 – Співвідношення мультикастних MAC- і IP-адрес

Оскільки при перетворенні IP-адреси в MAC-адресу втрачаються 5 бітів першого октету IP-адреси, отримана адреса не є унікальною. Кожній MAC-адресі відповідають 32 IP-адреси групової розсилки. Це необхідно враховувати при призначенні IP-адрес багатоадресної розсилки.

В протоколі IPv6 при використанні багатоадресної передачі даних також необхідно, щоб кілька вузлів могли отримувати потік даних із загальною MAC-адресою. MAC-адреса групової передачі протоколу IPv6 починається з префіксу, який складається з 16 бітів – 0x33-33. Наступні 32 біти формуються із останніх 32 бітів ідентифікатора багатоадресної групи (*Group ID*). Наприклад:
FF02::2 =33-33-00-00-00-02;
FF02::1:FF5C:B300= 33-33-FF-5C-B3-00

Групова адреса не ділиться на поля адреси мережі та вузла і опрацьовується маршрутизатором особливим чином.

В Windows переглянути членство в багатоадресних групах можна за допомогою команди: ***netsh interface ipv4 show joins*.**

Загальні відомості про багатоадресну маршрутизацію

Багатоадресна розсилка забезпечує доставку потоку даних групі вузлів на IP-адресу групи багатоадресної розсилки. У цієї групи немає фізичних або географічних обмежень: вузли можуть знаходитися у будь-якому місці. Вузли, які зацікавлені в отриманні даних для певної групи, повинні приєднатися до цієї групи (підписатися на розсилку) за допомогою міжмережевого протоколу управління групами IGMP (Internet Group Management Protocol,). Після цього

пакети багатоадресної розсилки, в полі призначення заголовку якого містяться групові адреси, будуть надходити на цей вузол і опрацьовуватися.

Кожна IP-адреса діапазону багатоадресного пересилання - це мультикастова група. Одна адреса - одна група, наприклад, IP-адресу 224.2.2.4 будуть отримувати тільки ті хости, які під'єднані до групи 224.2.2.4.

Протокол IGMP використовується клієнтським комп'ютером і сусідніми комутаторами для з'єднання клієнтського модуля і локального маршрутизатора, який здійснює групову передачу. Далі між локальним і віддаленими маршрутизаторами використовується протокол PIM (Protocol Independent Multicast), за допомогою якого груповий трафік направляється від сервера до численних клієнтів групової передачі.

Протокол IGMP реалізований у вигляді серверної і клієнтської частин, перша з яких виконується на маршрутизаторі, друга - на вузлі мережі, який отримує груповий трафік. Клієнт відправляє повідомлення про приналежність до якої-небудь групи локальному (найближчому) маршрутизатору, в цей час маршрутизатор перебуває в очікуванні повідомлень і періодично розсилає клієнтам запити (рис. 4.4).

Коли ініціюється багатоадресна передача, програмне забезпечення або сервіс створює багатоадресну групу. Ця адреса багатоадресної групи має IP-адресу з першим октетом в діапазоні 224–239 (клас D) і є адресою призначення для даного трафіку. Хост, який ініціює передачу, відправляє повідомлення (яке називається звітом про приналежність до IGMP) на адресу 224.0.0.2 (усім багатоадресним маршрутизаторам), вказуючи адресу багатоадресної групи. Комутатор отримує це повідомлення, додає багатоадресну групу у свою таблицю і додає порт прийому як учасника цієї групи. Він також пересилає цей звіт усім багатоадресним маршрутизаторам. Потім маршрутизатор додає ці хости у таблицю багатоадресної маршрутизації. Всі хости, які бажають стати учасниками групи, також відправляють повідомлення приєднання. Комутатор перехоплює ці повідомлення і додає порти прийому як учасників групи. Він також пересилає ці повідомлення багатоадресному маршрутизатору. Весь трафік, який відправляється на адресу призначення багатоадресного розсилання, направляється тільки на порти модулів, які є членами вказаної групи. Для того, щоб підтримувати інформацію про приналежність в актуальному стані, ініціатор IGMP-запиту продовжує відправляти запити приналежності. Всі хост-вузли, які бажають залишатися в групі, повинні відповідати на ці запити. Якщо хости, які входять в групу, не відповідають протягом встановленого часу, комутатор видаляє ці порти з таблиці групи. Після того як усі учасники будуть видалені з багатоадресної групи, комутатор видаляє адресу багатоадресної розсилки із своєї таблиці.

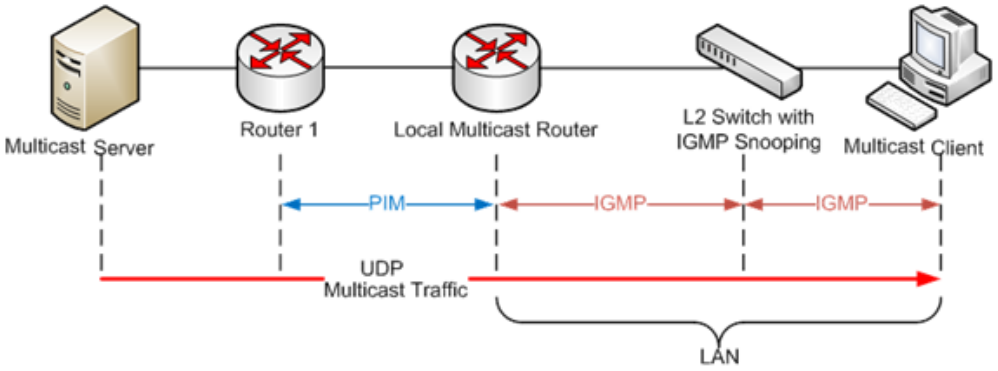


Рисунок 4.4 – Використання протоколів PIM і IGMP на ділянках мережі

Управління багатоадресною розсилкою на 2-му рівні моделі OSI (IGMP Snooping)

За замовчуванням, коли комутатор 2-го рівня отримує багатоадресний трафік, він починає передавати кадри на всі порти крім того, з якого надійшов цей кадр, оскільки в його таблиці комутації відсутній запис про багатоадресну MAC-адресу. Це не тільки недоцільно, але й може викликати проблеми на мережевих пристроях, змушуючи їх обробляти несподіваний для них потік даних.

Управління багатоадресною розсилкою на комутаторі 2-го рівня може бути виконане двома способами.

Перший спосіб полягає у створенні статичних таблиць комутації для портів, до яких під'єднані клієнти багатоадресних груп. Це дозволяє обмежити багатоадресний трафік і передавати його тільки через ті порти, до яких під'єднані вузли-підписники. Однак цей спосіб не дозволяє динамічно відслідковувати приєднання або вихід членів із багатоадресної групи. Іншим способом, який дозволяє вирішити проблему лавинної передачі багатоадресних пакетів і динамічно відслідковувати стан підписки вузлів, є функція IGMP-прослуховування (IGMP-Snooping).

IGMP Snooping - це функція другого рівня моделі OSI, яка дозволяє комутаторам вивчати членів багатоадресних груп, під'єднаних до його портів, прослуховуючи IGMP-повідомлення (запити і відповіді), які передаються між вузлами-підписниками і маршрутизаторами (або комутаторами рівня 3) мережі.

Коли вузол, який під'єднаний до комутатора, бажає ввійти в багатоадресну групу або відповідає на IGMP-запит, отриманий від маршрутизатора (або комутатора рівня 3) багатоадресної розсилки, він відправляє IGMP-відповідь, в якій вказана адреса багатоадресної групи. Комутатор аналізує інформацію в IGMP-відповіді і створює в своїй асоціативній таблиці комутації IGMP-Snooping (OIL) запис для цієї групи (якщо вона не існує). Цей запис зв'язує порт, до якого під'єднаний вузол-підписник, порт, до якого під'єднаний маршрутизатор (комутатор рівня 3) багатоадресної розсилки, і MAC-адресу багатоадресної групи. Якщо комутатор отримує IGMP-відповідь для цієї ж групи від іншого вузла даного сегменту мережі, то він

додає номер порту у вже існуючий запис асоціативної таблиці комутації IGMP Snooping (рис. 4.5).

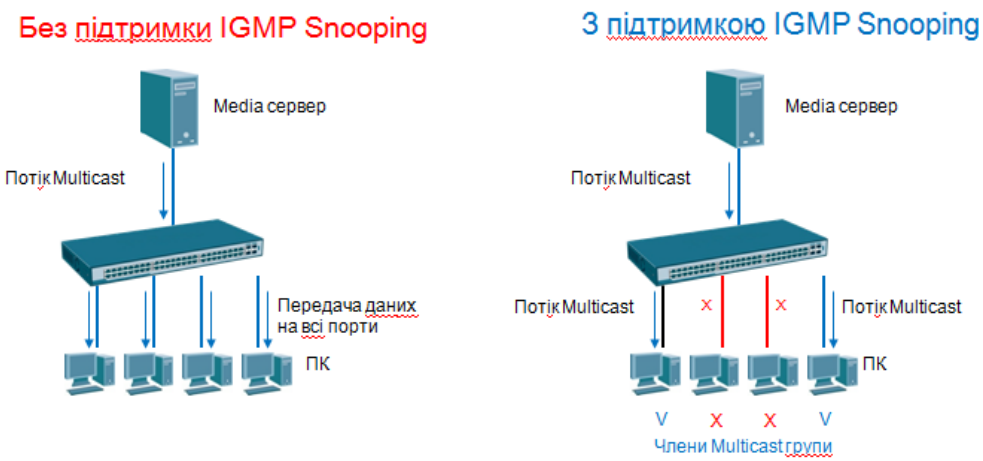


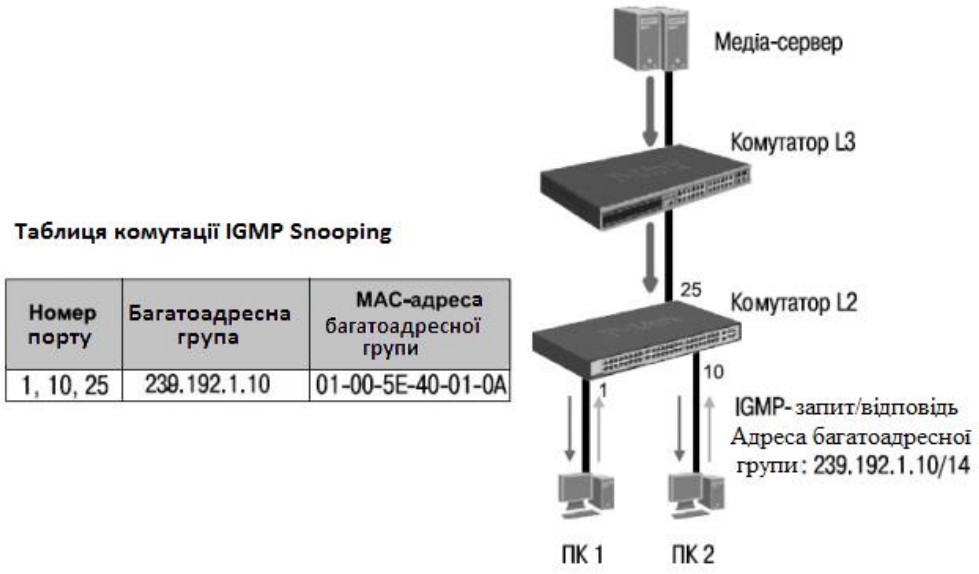
Рисунок 4.5 – Використання IGMP Snooping на локальному комутаторі

Розглянемо приклад роботи функції IGMP Snooping в мережі, представленої на рисунку 4.6.

Комутатор L3 відправляє IGMP-запит про приналежність до групи комутатору L2, який розсилає його через усі порти, за виключенням порту-отримувача. ПК1 бажає приєднатися до багатоадресної групи 239.192.1.10 і відправляє IGMP-відповідь на адресу групи, вказуючи багатоадресну MAC-адресу призначення 0x01-00-5E-40-01-0A. Процесор комутатора L2 аналізує IGMP-відповідь і створює в асоціативній таблиці комутації IGMP Snooping (спочатку вона пуста) запис для MAC-адреси 0x01-00-5E-40-01-0A, що відповідає груповій адресі 239.192.1.10. Також в цей запис заноситься інформація про порти, до яких під'єднані ПК1 і комутатор L3.

ПК2 також бажає вступити в багатоадресну групу 239.192.1.10 і відправляє IGMP-відповідь на адресу групи, не очікуючи отримання чергового IGMP-запиту. Комутатор L2 аналізує IGMP-відповідь і додає *порт* 10, до якого під'єднаний ПК 2, у вже існуючий запис для MAC-адреси 0x01-00-5E-40-01-0A.

В результаті порти 1, 10 і 25 асоційовані з багатоадресною MAC-адресою 0x01-00-5E-40-01-0A. Для підвищення продуктивності при виконанні функції IGMP Snooping в комутаторах використовуються спеціалізовані мікросхеми ASIC, які перевіряють IGMP-повідомлення на апаратному рівні.



Таблиця комутації IGMP Snooping

Номер порту	Багатоадресна група	MAC-адреса багатоадресної групи
1, 10, 25	239.192.1.10	01-00-5E-40-01-0A

Рисунок 4.6 - Процес створення таблиці комутації IGMP Snooping

IGMP має три версії протоколу: v1, v2 і v3. Зазвичай IGMPv1 ідентифікує локальний маршрутизатор, використовуючи протокол багатоадресної маршрутизації. IGMPv2 додає можливість групових запитів, дозволяючи комутаторам відправляти повідомлення хостам в багатоадресній групі. IGMPv3 надає більше зручних можливостей для підтримки фільтрації певних джерел.

Маршрутизація мультикастових повідомлень на відрізку маршрутизатор – клієнт

При одноадресній маршрутизації кожний маршрутизатор перевіряє адресу призначення вхідного пакету і шукає його в таблиці комутації, щоб визначити, який інтерфейс використовувати для передачі пакету до місця призначення. Адреса джерела маршрутизатором не аналізується. Однак при багатоадресній маршрутизації адреса джерела (яка є простою адресою одноадресного розсилання) використовується для визначення напрямку потоку даних. Джерело багатоадресного трафіку вважається вхідним. Маршрутизатор визначає, які вихідні інтерфейси є адресатами для цієї багатоадресної групи (адреси призначення), і відправляє пакет через відповідні інтерфейси. Термін «пересилка зворотного шляху» використовується для опису цієї концепції маршрутизації пакетів від джерела, а не до місця призначення.

Отже, IGMP працює на відрізку маршрутизатор - клієнт. Коли клієнт бажає приєднатися до групи, наприклад, 239.192.1.10, він відправляє IGMP-запит, тобто хост рапортує про те, що він бажає отримати трафік цієї групи. Процедура під’єднання клієнтського модуля до мультикастової групи представлена на рисунку 4.7.

Маршрутизатор R1, отримавши такий запит, формує вихідний список отримувачів. Якщо трафік вже існує, R1 відразу починає передавати його в інтерфейс, з якого отримав IGMP-запит. Повідомлення відправляються на ту ж

мультикастову адресу 239.192.1.10, до якої бажав приєднатися хост. Так працює протокол IGMPv2.

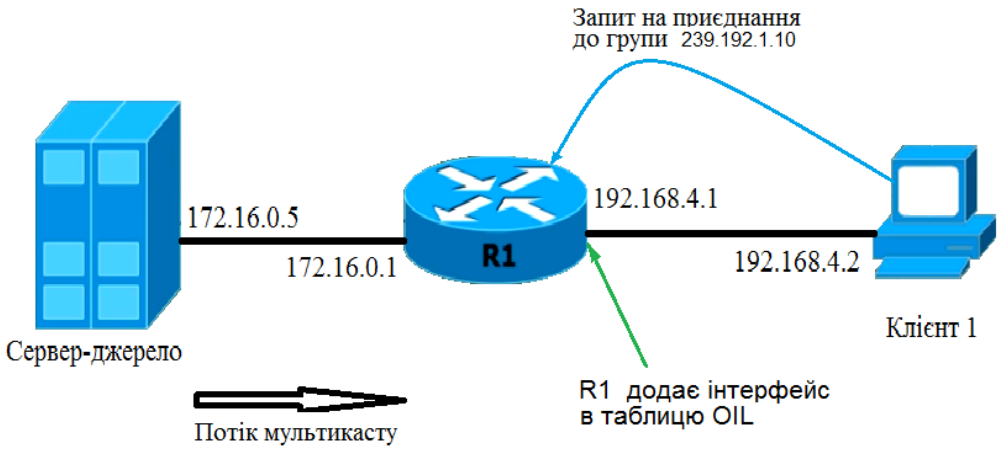


Рисунок 4.7 – Приєднання клієнта до мультикастової групи

Якщо клієнт вирішив від’єднатися, він відправляє пакет IGMP Leave (рис. 4.8). Отримавши цей пакет, маршрутизатор видаляє інтерфейс зі списку отримувачів і після деякої паузи (щоб впевнитися, що інших клієнтів немає) припиняє передавати трафік в цей інтерфейс. Періодично маршрутизатор розсилає спеціальні повідомлення IGMP Query з широкомовною адресою призначення (224.0.0.1). Такий пакет носить назву загального запиту General Query. У відповідь на General Query будь-який клієнт повинен відправити IGMP-повідомлення. Таким чином маршрутизатор перевіряє, в яких групах є отримувачі з конкретними інтерфейсами. Крім того, існують повідомлення Group Specific Query, які надсилаються на адресу конкретної групи.

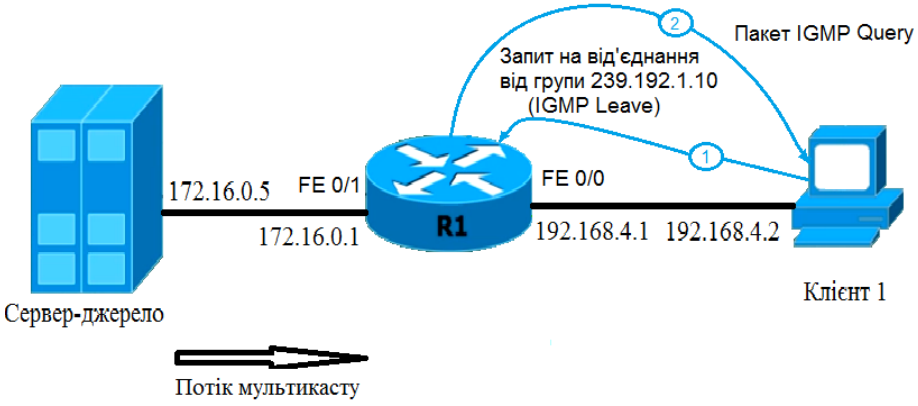


Рисунок 4.8 – Виконання запиту на вихід із мультикастової групи

Коли маршрутизатор отримує IGMP Leave, він не відразу видаляє інтерфейс. Спочатку він відправляє повідомлення Group Specific Query на випадок, якщо там є ще клієнти. Якщо вони є, то надходить IGMP-відповідь, якщо відповідь не надійшла – інтерфейс можна видалити.

Зазвичай протоколи IGMP і RIP включаються окремо, оскільки у них різні задачі, але на маршрутизаторах Cisco вони включаються одночасно.

Маршрутизація мультикастових повідомлень на відрізку сервер – маршрутизатор

Раніше було розглянуто випадок, коли на маршрутизатор мультикастовий трафік вже надходить. Розглянемо виникнення мультикастового трафіку. За допомогою протоколу IGMP маршрутизатор дізнається про клієнта. З іншого боку маршрутизатор отримує потік мультикасту. Тепер виникає задача з'єднати потік мультикасту з клієнтом. Таке з'єднання відбувається завдяки роботі протоколу PIM. Його задача полягає в побудові дерева (в розглянутому прикладі це дерево складається з одного маршрутизатора). Його таблиця маршрутизації містить записи про призначення портів, наприклад:

Incoming Interface: FastEthernet 0/1
Outcoming Interface list:
FastEthernet 0/0
……

Маршрутизатор приймає трафік на інтерфейс FastEthernet 0/1 і передає його на інтерфейс FastEthernet 0/0. Якщо протокол PIM не буде задіяний, то трафік не буде передаватися взагалі.

Розглянемо приклад складнішої мережі, структура якої представлена на рисунку 4.9.

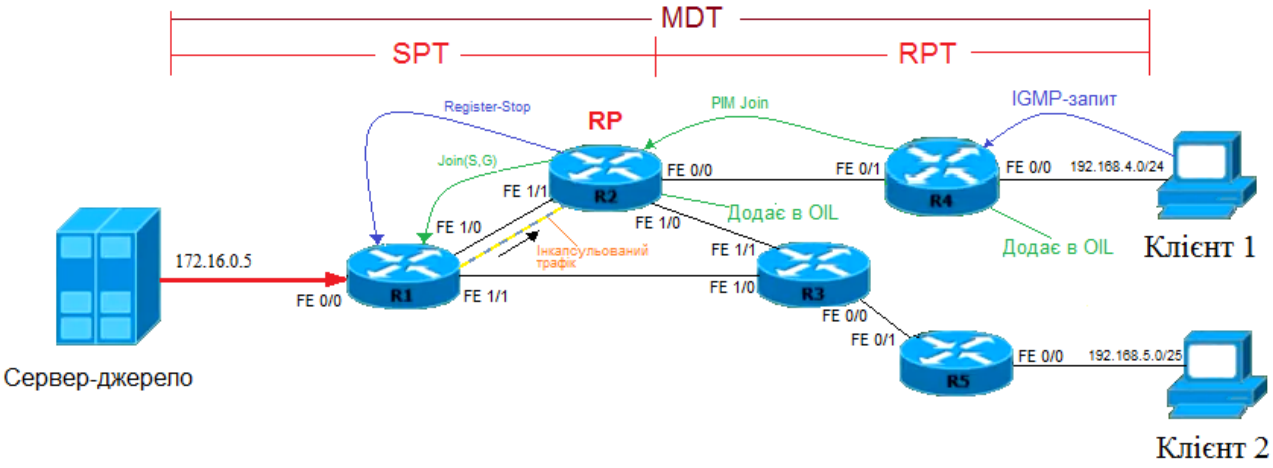


Рисунок 4.9 – Приклад мультикастової маршрутизації

Маршрутизатор R1 отримує потік UDP і знає про джерело мультикасту. Клієнт спілкується з маршрутизатором R4 через протокол IGMP, тому R4 знає про клієнта. Але R1 не знає куди передавати трафік, а R4 нічого не знає про джерело. Тому в мережі завжди є хоча б один маршрутизатор, який має інформацію як про джерело мультикасту, так і про клієнтів. Такий маршрутизатор називається точкою зустрічі RP (Rendezvous Point). До маршрутизатора RP надходить трафік від джерела і до нього ж прагнуть клієнти. Всі маршрутизатори в мережі повинні знати, який маршрутизатор є RP у мережі.

В першу чергу всі маршрутизатори повинні отримати інформацію про сусідні маршрутизатори в мережі. Роблять це вони за допомогою повідомлення **Hello.** Час життя TTL повідомлення **Hello** дорівнює 1.

Отже, клієнт 1 при підключенні відправляє IGMP-запит. Маршрутизатор R4 додає інтерфейс, з якого надійшов цей запит в список вихідних інтерфейсів, а потім відправляє пакет PIM Join в напрямку до RP. Таким чином він демонструє бажання приєднатися до мультикастового дерева. PIM Join надсилається на спеціальну (широкомовну) адресу 224.0.0.13 із TTL, що дорівнює 1. Практично всі пакети PIM використовують саме цю адресу призначення. Кожний маршрутизатор на шляху до RP буде використовувати її індивідуально, тобто маршрутизатор приймає такий PIM Join, опрацьовує його, видаляє старий, формує новий і відправляє його далі, і так до того моменту, поки PIM Join не надійде до RP. У розглянутому прикладі PIM Join надходить до RP відразу.

Маршрутизатор RP, отримавши PIM Join, додає інтерфейс в список вихідних інтерфейсів OIL, формуючи таким чином одну гілку мультикастового дерева.

Нехай тепер клієнт 2 бажає під'єднатися до мультикастової групи. Він також відправляє IGMP-запит на R5. Маршрутизатор R5 формує пакет PIM Join і відправляє його у напрямку маршрутизатора RP. Така відправка відбувається через той інтерфейс, який веде до RP згідно з юнікастовою таблицею маршрутизації.

Маршрутизатор R3 отримує пакет PIM Join і не передає його далі, а опрацьовує і знищує. Якщо у нього ще немає клієнтів із цієї групи, він формує новий пакет PIM Join і відправляє його у напрямку RP. Маршрутизатор RP, отримавши PIM Join, додає відповідний запис у список вихідних інтерфейсів таблиці OIL.

Тепер сформоване повноцінне дерево, яке називається дерево від маршрутизатора RP до клієнта 1 та клієнта 2 (RPT), тобто маршрутизатор знає про наявність клієнтів.

Але поки-що на маршрутизаторі RP немає трафіку. Після запуску сервера-джерела маршрутизатор R1 починає отримувати трафік. Кожний вхідний мультикастовий пакет він упаковує в юнікастовий, додає юнікастову адресу отримувача пакету маршрутизатора RP і відправляє такий пакет як звичайний IP-пакет прямо на RP з TTL, який дорівнює 255. Таке повідомлення називається реєстрацією джерела на RP (PIM Register) і разом з повідомленням Register Stop передаються юнікастом на юнікастові адреси. Далі RP отримує цей юнікастовий пакет, вилучає з нього інкапсульований мультикаст і відразу відправляє його в побудоване дерево RPT, якщо воно є, тобто одна копія передається до інтерфейсу FastEthernet 0/0, а інша – на FastEthernet 1/0.

Кожний наступний маршрутизатор, отримавши мультикастовий трафік, теж розсилає його у всі свої інтерфейси згідно зі списками вихідних інтерфейсів.

Але така інкапсуляція (або тунелювання) не самий кращий варіант. По-перше, кожний пакет потрібно інкапсулювати, потім розпакувати. Крім того, збільшується довжина пакету, пов'язана з тунелюванням. По-друге, якщо на шляху від джерела до RP виявиться клієнт цієї групи (маршрутизатор RX на рис. 4.10), то трафік від сервера повинен спочатку передаватися до RP, а потім повернутися назад. В результаті, лінія зв'язку буде використовуватися двічі.

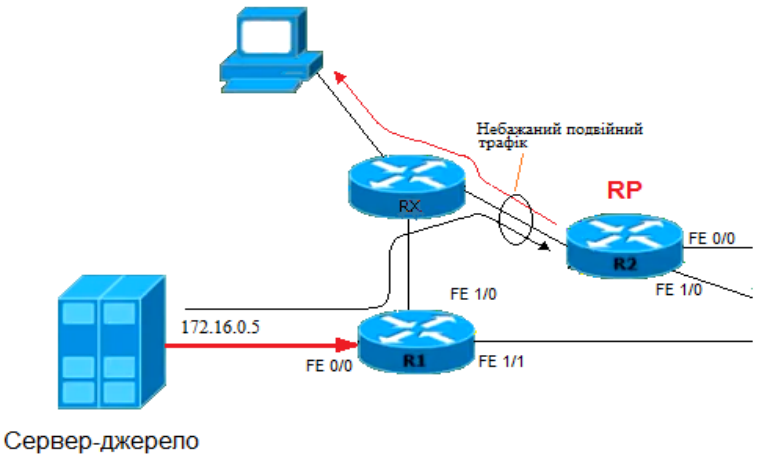


Рисунок 4.10 – Поява небажаного тунелювання

Тому, як тільки маршрутизатор RP отримує перший пакет PIM Register, він ініціює побудову дерева від сервера-джерела до себе. Для цього він відправляє пакет PIM Join (S,G) в напрямку сервера-джерела. Це не такий PIM Join, який направляли клієнтські маршрутизатори, це так званий Source Specific Join. В ньому вказується конкретна адреса сервера-джерела, від якого необхідно побудувати дерево. Тепер важливо, щоб дерево було побудоване саме від цього джерела, адже їх може бути кілька. Передається цей пакет до джерела так само, як і звичайний PIM Join: від вузла до вузла

Коли маршрутизатор R1 отримує такий PIM Join, паралельно з повідомленням PIM Register він починає відправляти чистий мультикаст по дереву, побудованим за допомогою PIM Join.

Коли на маршрутизатор RP починають надходити дві копії трафіку (інкапсульований і чистий), він відправляє пакет PIM Register Stop (теж юнікастовий, як і звичайний пакет PIM Register). Це не означає відмову від мультикасту, це повідомлення для маршрутизатора R1 про те, що потрібно припиняти відправляти інкапсульований трафік.

Побудоване дерево від джерела до RP називається найкоротшим шляхом від джерела до RP SPT (Shortest Path Tree).

Отже, розглянуто процес побудови шляху від джерела мультикасту до сервера RP (SPT) і шлях від клієнтів до сервера RP (RPT). Повний шлях від сервера да клієнтів має загальну назву MDT (Multicast Distribution Tree).

Завдання

1. При виконанні роботи використовується програмне забезпечення для аналізу протоколів комп'ютерних мереж **Wireshark**. Запустити програму Wireshark.
2. Відібрати для аналізу 3 кадри: 1 персональний (unicast), 1 груповий (multicast), 1 широкомовний (broadcast). Всі ці кадри скопіювати у звіт, позначивши в кожному з них MAC-адресу відправника, MAC-адресу отримувача, тип або довжину кадру. Для фільтрації широкомовних кадрів необхідно в меню Analyze/Display Filter вибрати фільтр «Ethernet broadcast». Для фільтрації групових кадрів необхідного створити новий фільтр з назвою «Ethernet multicast». Для цього необхідно в меню Analyze/Display Filter вибрати New, в полі Filter Name набрати «Ethernet multycast», напроти Filter String натиснути кнопку Expression. У полі Field Name знайти рядок «Ethernet». Далі вибрати підрядок eth.multicast, у полі «Relation» вибрати «= =», в полі «Predefined Values» вибрати «This is a multicast frame». Аналогічно створюється фільтр для unicast.

Для спостереження за трафіком multicast почергово використайте:

а) фільтр захоплення,
б) фільтр відображення,
в) фільтр мережевого рівня,
г) фільтр канального рівня.

Контрольні запитання

1. Що таке хост?
2. Які види адрес використовуються в стеку TCP/IP? Наведіть приклади.
3. Із яких частин складається IP-адреса?
4. Що таке маска підмережі?
4. Як визначається номер підмережі в IP-адресі?
6. Визначте номер підмережі на основі маски: 116.98.04.39/27.
7. Які основні особливості протоколу IPv6?
8. Опишіть механізм визначення MAC-адрес призначення при виконанні пересилок багатоадресних повідомлень.
9. За яких двох етапів відбувається утворення каналу передачі даних при мультикастовому з'єднанні клієнта з відправником мультикасту?
10. У яких випадках мультикастинг користується юнікастовим способом адресації?
11. Що таке «точка зустрічі» Renderzvous Point в мультикастовій мережі?

Лабораторна робота 5
Доменна служба імен. Утиліти nslookup та dig

Мета роботи: поглиблене самостійне вивчення спеціальних питань, присвячених організації та конфігуруванню сервера доменних імен.

План виконання лабораторної роботи

1. Ознайомитися та засвоїти теоретичні відомості, викладені в методичному посібнику до лабораторної роботи.
2. Виконати завдання до лабораторної роботи.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Простір імен DNS

Internet представляє собою сукупність сотень тисяч взаємопов'язаних гетерогенних мереж і комп'ютерів (хост-машин). Побудова Internet базується на концепції ієрархії протоколів. Протоколи Internet називаються сімейством протоколів TCP/IP. В стеку TCP/IP комп'ютери взаємодіють один з одним за допомогою різних типів адрес. На фізичному рівні використовуються низькорівневі фізичні адреси, які ідентифікують апаратні пристрої, що використовуються. На мережевому рівні застосовується перша абстракція, яка використовує логічні адреси хост-машин (IP-адреси). На прикладному рівні застосовується абстракція іншого рівня, яка використовує зручні для сприймання користувачами імена хост-машин (hostnames). В сімействі протоколів TCP/IP перетворення адрес мережевого рівня (IP-адрес) в фізичні адреси виконується протоколом ARP. Для перетворення (або зв'язування (binding)) імен хост-машин в IP-адреси використовується служба доменних імен DNS (Domain Name System).

Служба DNS також використовується для маршрутизації електронної пошти.

Схема взаємодії програми користувача з локальним і віддаленими DNS-серверами показана на рисунку 5.1.

Служба DNS базується на двох основних концепціях:

1) розподіленій базі даних, яка зберігає узагальнені записи про ресурси мережі (resource records) з децентралізованим керуванням;
2) схемі іменування, що базується на ієрархічно-структурованих доменних іменах.

Система DNS являє собою розподілену базу даних, яка зберігається не на одному DNS-сервері. Кожен DNS-сервер підтримує власну інформаційну базу даних, а також запускає відповідне застосування, що відправляє запити до інших серверів. Протокол DNS дозволяє клієнтам і серверам спілкуватися між собою, що дозволяє локально контролювати окремі сегменти загальної бази даних. Дані у кожному сегменті доступні через мережу завдяки використанню

технології клієнт-сервер. Висока продуктивність служби DNS досягається за допомогою використання механізмів копіювання (replication) і кешування (caching).

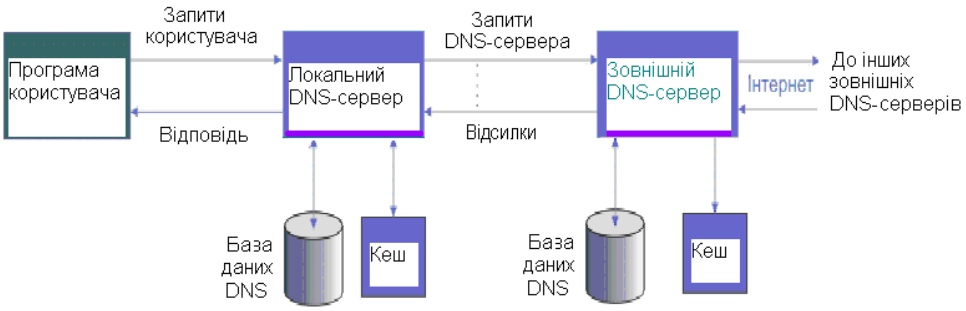


Рисунок 5.1 – Схема взаємодії програми з DNS-серверами

Програми, які реалізують серверну частину DNS, називаються серверами імен (name servers). Сервер імен зберігає інформацію про деякий сегмент загальної бази даних DNS, для якого він є повноважним (авторитетним) сервером і робить її доступною для клієнтів, якими є так звані програми-вирішувачі (resolvers). Резолвери, зазвичай, є бібліотечними функціями які генерують запити і надсилають їх через мережу серверам імен.

Система DNS представляє собою ієрархічну структуру, схожу на дерево, і кожен вузол може або самостійно визначати роботу розташованих нижче вузлів, або делегувати виконання цієї роботи іншим вузлам.

Кожен вузол на рисунку 5.2 має мітку довжиною до 63 символів. Корінь дерева - це спеціальний вузол без мітки. Мітки можуть містити літери верхнього або нижнього регістру. Ім’я домену для будь-якого вузла в дереві - це послідовність міток, що починається з кореневого вузла (при цьому мітки розділяють крапками). Кожен вузол дерева повинен мати унікальне ім’я домену, але однакові мітки можуть використовуватися в різних точках дерева.

Повноваження доменів збільшуються при наближенні до кореня дерева.

Налаштування кореневої зони розташовані на багатьох серверах/дзеркалах, розміщених по всьому світу. Вони зберігають інформацію про всі сервери кореневої зони, а також відповідають за домени першого рівня (ru, net, org та ін).

Сервери кореневої зони відповідають на запити, надаючи інформацію тільки про домени першого рівня.

Кореневе ім’я (root's name) має нульову мітку і позначається одиночною крапкою ("."). Кожному вузлу (node) дерева відповідає розділ бази даних або домен (domain). Кожен домен згодом може ділитися на піддомени, які є нащадками своїх батьківських вузлів (parent nodes). Кожен домен має мітку (label), яка ідентифікує його розташування відносно батьківського домену. Крім того, домен має доменне ім’я (domain name), яке ідентифікує його розташування в базі даних DNS. Повне доменне ім’я являє собою послідовність

міток від кореневого домену, які відокремлюються один від одного символом «.».

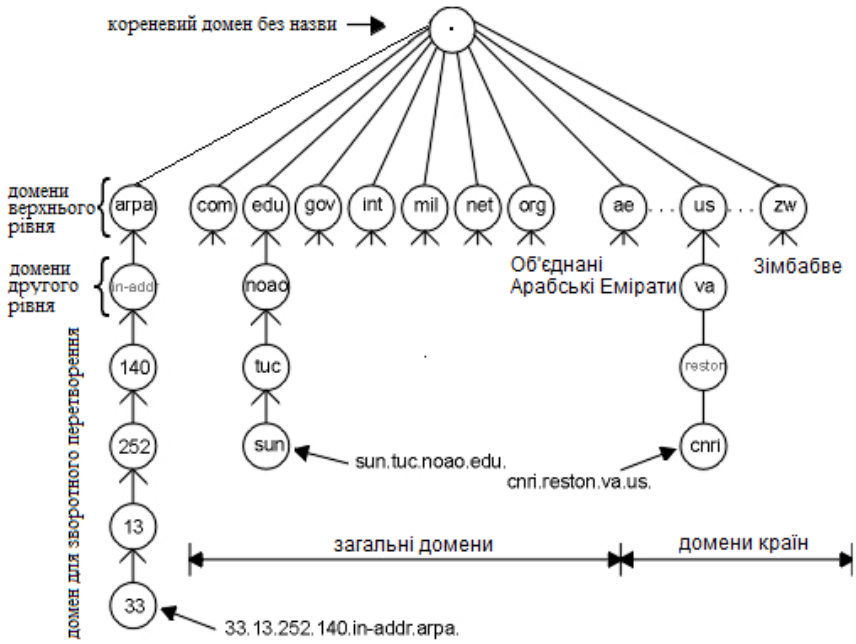


Рисунок 5.2 – Структура бази даних DNS

Як вже зазначалося, ім'я складається з кількох полів (міток). Ці поля мають довжину не більше 63 символів. Поля розділені крапками. Ім'я може складатися не більше ніж з 255 полів (октетів), включаючи байт довжини. Аналіз імені виконується справа наліво.

Ім’я хосту в домені має вигляд, показаний на рисунку 5.3.

```
scs.ntu-kpi.kiev.ua.
 |    |         |   ||  |
 |    |         |   ||  +---кореневий домен
 |    |         |   |+-----домен першого рівня
 |    |         |   +------крапка, що відокремлює домени в імені
 |    |         +----------домен другого рівня
 |    +-------------------домен третього рівня
 +----------------------піддомен/домен четвертого рівня, можливо, ім’я хосту.
```

Рисунок 5.3 – Компоненти імені домену

Доменні імена відображають логічну структуру мережі і, часто, функціональне призначення того або іншого хосту. Крайнє праве поле позначає домен верхнього рівня, далі, справа наліво, поля вказують на піддомени у порядку ієрархічної вкладеності, крайнє ліве поле позначає ім'я хосту. Наприклад, **scs.ntu-kpi.kiev.ua** - хост **scs** в домені **ntu-kpi**, який входить до складу домену **kiev**, який в свою чергу входить до складу домену **ua**.

Дерево доменних імен аналогічне файловій системі Linux. Коренем дерева є домен «.» (крапка). Повне - абсолютне або повністю визначене, fully qualified domain name - доменне ім'я закінчується крапкою, яка позначає корінь доменного дерева, але зазвичай ця завершальна крапка опускається і мало хто її використовує.

Фактично, ім'я вузла в доменному дереві є індексом для даних, пов'язаних з цим ім’ям. Ці дані можуть бути різних типів (IP-адреси, псевдоніми хосту та ін.), вони зберігаються в базі даних серверу імен (DNS-серверу) у вигляді стандартних записів ресурсів (Standard Resourse Record), формат яких буде розглянутий пізніше. Ім'я домену може збігатися також з ім'ям доменного вузла. Наприклад, може існувати як хост з іменем **kiev.ua**, з яким можна встановити з'єднання, так і домен **kiev.ua**, в якому знаходяться хости і, можливо, піддомени.

Приклади прямого і зворотного перетворення будуть розглянуті відповідно в підрозділах 1.4 та 1.5.

Служба DNS є ієрархічною структурою серверів, де кожний сервер відповідає за певну зону, тобто за свою частину дерева доменних імен, зберігає відповідні бази даних і відповідає на запити. При цьому розташовані вище по дереву сервери мають інформацію про адреси розташованих нижче серверів, що забезпечує зв'язність дерева (говорять, що розташований вище сервер делегує розташованому нижче серверу повноваження з обслуговування певної зони).

1.2. Типи серверів імен

У DNS визначено два типи серверів імен: первинні (primary, masters) і вторинні (secondary, slave). Первинний сервер імен отримує дані про зону, для якої він є повноважним, із файлів на хост-машині, де він розміщується. Вторинний сервер імен отримує дані про зону від первинного сервера імен. Коли стартує вторинний сервер, він зв’язується з повноважним (первинним) сервером зони і копіює дані про зону. Таку процедуру називають пересилкою зони (zone transfer). В процесі роботи вторинний сервер періодично опитує первинний сервер. Якщо дані на первинному сервері були змінені, то вторинний сервер оновлює свої дані, копіюючи всі дані зони з первинного сервера.

Зазвичай для зони створюються один первинний сервер імен і один або кілька вторинних серверів. Існує два механізми копіювания зони: повне копіювання (AXFR) та інкрементне (incremental) копіювання зони (IXFR).

Якщо сервер не має запитуваної інформації, він повинен взаємодіяти з кореневими серверами. Загальновідомих кореневих серверів у світі всього 13, їх IP-адреси мають знаходитися в конфігураційних файлах кожного DNS-сервера.

Список кореневих серверів можна отримати за допомогою анонімного FTP-сервера за адресами: nic.ddn.mil або ftp.rs.internic.net. Кореневі сервери зберігають інформацію про імена і адреси усіх серверів доменів першого рівня.

Сервери бувають рекурсивними і нерекурсивними (або ітеративними). **Нерекурсивний сервер** діє таким чином: якщо у нього є запитувана адреса, кешована із попереднього запиту, або якщо він повноважний (або авторитетний) для домену, до якого належить доменне ім’я, то він дає відповідну відповідь. В іншому випадку замість правильної відповіді він дає посилання на авторитетний сервер іншого домену, який повинен знати відповідь. **Рекурсивний сервер** імен повертає тільки реальні відповіді та повідомлення про помилки. Він сам відслідковує посилання, звільняючи від цих обов’язків клієнта. Побічним наслідком такої роботи може бути швидке наповнення кешу рекурсивного серверу і, як наслідок, додаткові витрати часу на обробку рекурсивних запитів, тобто пропускна спроможність серверу зменшується. Тому сервери нижчих рівнів зазвичай є рекурсивними, а сервери вищих рівнів (кореневого і, частково, першого) – нерекурсивними.

1.3. Кореневі сервери імен

Кореневі сервери імен зберігають інформацію про повноважні сервери всіх доменів першого рівня. У відповідь на запит про яке-небудь доменне ім’я кореневий сервер імен може надати імена і адреси повноважних серверів імен домену першого рівня, в який входить ім’я, вказане у запиті. Сервери імен першого рівня видають імена і адреси повноважних серверів імен домену тільки другого рівня. Кожен запитуваний сервер імен видає запитуючій стороні інформацію про те, як наблизитись до відповіді, яку він шукає, або надати саму відповідь.

Оскільки перетворення всіх доменних імен в Internet починається з кореневих серверів, вони є найбільш критичним ресурсом в DNS. Тому в Internet існує кілька кореневих серверів, розташованих в різних мережах різних частин світу.

1.4. Перетворення доменних імен в IP-адреси

Мережеве програмне забезпечення, маршрутизатори та інша мережева апаратура працюють з IP-адресами. Перетворення «доменне ім'я на IP адресу» (прямі перетворення) виконуються службою DNS шляхом пошуку в доменному дереві потрібного імені та витягування пов'язаної з цим ім'ям інформації певного типу (IP-адреси).

Як було зазначено раніше, існує дві схеми запитів, які можуть використовуватися для перетворення доменного імені: рекурсивні та нерекурсивні (ітеративні).

Розглянемо приклад **рекурсивного** (recursive query) запиту (рис. 5.4).

1. Клієнт (браузер, поштова програма) відправляє запит резолверу, який, використовуючи свої файли конфігурації, визначає адресу локального сервера імен.

2**.** Резолвер відправляє запит локальному серверу імен. Resolver, зазвичай, не є якою-небудь програмою чи системною компонентою, а являє собою набір процедур із бібліотеки програмного забезпечення додатків, які дозволяють

програмі, яка відредагована з ними, виконувати запити до системи доменних імен і отримувати відповіді на них. Ці процедури звертаються до сервера доменних імен і, таким чином, обслуговують запити програм-застосувань користувача.

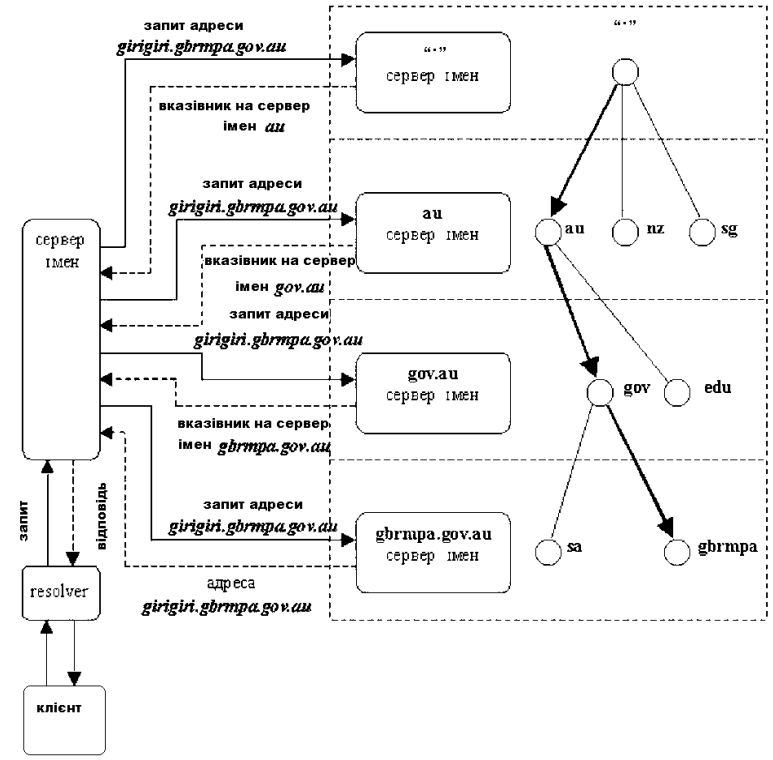


Рисунок 5.4 – Приклад рекурсивного запиту на перетворення доменного імені

3. Сервер імен приймає цей рекурсивний запит і, не маючи інформації ні про домен, ні, можливо, навіть про зону au (тобто необхідні дані відсутні в його базі і в кеші від попередніх запитів), відправляє рекурсивний (або нерекурсивний, в залежності від налаштувань) запит про адресу хост-машини girigiri.gbrmpa.gov.au кореневому серверу.
4. Кореневий сервер не виконує рекурсивні запити, і тому виконує обробку даного запиту як ітеративного і повертає ім’я й адресу авторитетного сервера в зоні au.
5. Локальний сервер звертається до сервера «au» з тим же запитом, і отримує посилання на сервер імен «gov.au».
6. Сервер імен «gov.au» вказує локальному серверу на сервер «gbrmpa.gov.au».
7. І нарешті локальний сервер виконує запит серверу «gbrmpa.gov.au» і отримує від нього потрібну адресу.
8. Сервер локальної мережі повертає резолверу клієнта запрошену адресу.

Зауважимо, що даний приклад розглянуто в припущенні, що перед запитом жодні із запрошених даних не кешувались, за винятком даних серверів домену au.

На практиці кількість кроків зменшена до мінімуму, оскільки на шляху проходження запитів зустрічаються кешуючі сервери, які зберігають необхідну інформацію у кеші.

Якщо сервер DNS не підтримує рекурсивних запитів, то клієнт за замовчуванням буде виконувати нерекурсивні (ітеративні) запити.

Розглянемо приклад нерекурсивного (ітеративного) запиту.

Нехай хост store.khsu.ru має встановити з'єднання з хостом movies.yahoo.com (рис.5.5).

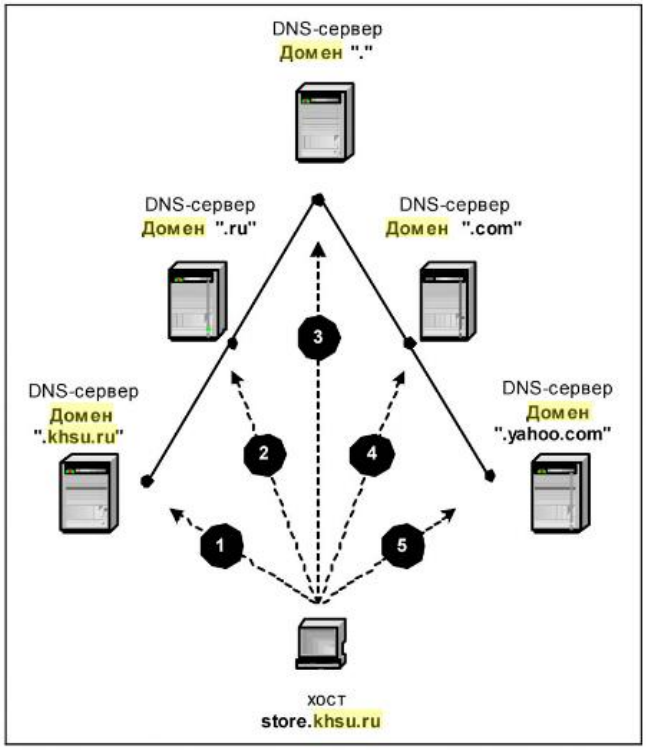


Рисунок 5.5 – Приклад нерекурсивного перетворення доменного імені

Для ітеративного перетворення вказаного доменного імені клієнт виконує наступні дії.

1. Клієнт звертається до серверу імен, який обслуговує домен khsu.ru. Цей сервер імен не має інформації про запитуване ім'я і повертає клієнту посилання на розташований вище сервер імен, який обслуговує домен ru.
2. Клієнт звертається до серверу імен, який обслуговує домен ru. Цей сервер імен не має інформації про запитуване ім'я і повертає клієнту посилання на кореневий сервер імен (домен «.»).
3. Клієнт звертається до кореневого серверу імен. Кореневий сервер імен виявляє, що запитуване доменне ім'я належить до домену com. Клієнту повертається посилання на сервер імен, що обслуговує домен com.
4. Клієнт звертається до серверу імен, який обслуговує домен com. Цей сервер імен знаходить в своїй базі даних запис про сервер імен, який обслуговує домен yahoo.com. Посилання на цей сервер імен повертається клієнту.
5. Клієнт звертається до серверу імен, який обслуговує домен yahoo.com. Цей сервер імен знаходить в своїй базі даних запис про хост movies.yahoo.com. IP-адреса, яка відповідає цьому імені, повертається клієнту.

У цьому сценарії основне навантаження з перетворення доменного імені в IP-адресу лежить на DNS-клієнті.

Рекомендованим методом перетворення імен в IP-адреси є рекурсивний. Деякі адміністратори використовують ітеративний спосіб перетворення адрес з метою збільшення продуктивності сервера DNS. Якщо рекурсія відключена, то сервер DNS виконує ітеративні запити.

Розглянемо, яким чином локальний DNS-сервер, отримавши рекурсивний запит від резолвера, вибирає DNS-сервер із списку авторитетних.

Для вирішення цього питання DNS-сервери використовують метрику, яка називається часом відгуку RTT (RoundTrip Time), і визначає затримку, з якою надходять відповіді на запити від віддаленого сервера. Кожен раз, при передачі запиту віддаленому серверу, локальний DNS-сервер запускає внутрішній таймер. Таймер зупиняється при одержанні відповіді, і метрика фіксується локальним сервером. Якщо доводиться вибирати один із кількох авторитетних серверів, то обирається сервер із найменшим значенням RTT.

До того, як локальний сервер вперше надсилає запит якому-небудь серверу і отримує від нього відповідь, віддаленому серверу присвоюється випадкова величина RTT, яка є меншою, ніж всі інші, отримані шляхом вимірювань. Таким чином, гарантовано буде виконано опитування всіх авторитетних серверів для певної зони, перш ніж почнеться вибір на основі метрики.

1.5. Зворотні перетворення (перетворення IP-адрес в доменні імена)

Служба DNS призначена не тільки для перетворення DNS-імені в IP-адресу, але і для вирішення зворотної задачі – знаходження DNS-імені за відомою IP-адресою, що вирішується шляхом створення зворотних зон.

Зворотна зона − це система таблиць, які зберігають відповідність між IP-адресами і DNS-іменами хостів мережі. Для організації розподіленої служби і використання для пошуку DNS-імен того ж самого програмного забезпечення, що і для пошуку IP-адрес, застосовується оригінальний підхід, який полягає в тому, що IP-адреса подається у вигляді DNS-імені.

Перший етап перетворення полягає в тому, що складові IP-адреси інтерпретуються як складові DNS-імені. Наприклад, адреса 15.16.192.152 розглядається як така, що складається зі старшої частини, що відповідає домену 15, потім слідує домен 16, в який входить домен 192, в який входить домен 152.

Далі, враховуючи, що при записуванні IP-адреси старша частина адреси є самою лівою, а при записуванні DNS-імені − самою правою, то складові в перетвореній адресі вказуються в зворотному порядку, тобто в цьому прикладі − 152.192.16.15.

Для збереження відповідності всіх адрес, які починаються, наприклад, з числа 192, створюється зона 192 зі своїми серверами імен. Для записів про сервери, які підтримують старші в ієрархії зворотні зони, створена спеціальна зона in-addr.arpa, тому повний зворотний запис для адреси в нашому прикладі має такий вигляд: 152.192.16.15.in-addr.arpa.

Сервери для зворотних зон використовують файли баз даних, які не залежать від файлів основних зон, в яких розташовані записи про пряму

відповідність тих же адрес і імен. Така організація даних може привести до неузгодженості, оскільки одна й та ж сама відповідність вводиться в файли двічі.

Структура домену in-addr.arpa показана на рисунку 5.6.

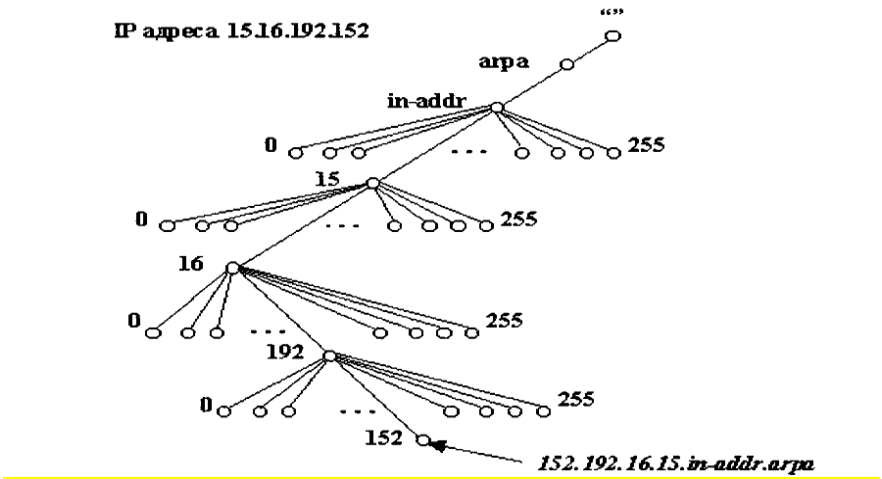


Рисунок 5.6 – Приклад зони in-addr.arpa

Сенс описаної схеми полягає в тому, що зворотне DNS-перетворення «IP-адреса в ім'я» не є якимось особливим перетворенням, а є звичайним прямим перетворенням в одному із піддоменів домену in-addr.arpa, при цьому даними для кожного вузла (тобто, фактично, IP-адресою) в цьому домені виступає власне доменне ім'я хоста. Пошук виконується за звичайним алгоритмом: спочатку опитується кореневий сервер, який, якщо у нього в кеші немає готової відповіді, повертає адресу серверу, що відповідає за in-addr.arpa. Далі сервер in-addr.arpa (за відсутності готової відповіді) повертає адресу серверу, що відповідає за 15.in-addr.arpa, і так далі.

Зустрітися з доменом in-addr.arpa і переставленими байтами в IP-адресі можна тільки тоді, коли ми маємо справу з DNS, використовуючи такі програми як host або переглядаючи пакети з використанням утиліти tcpdump. Під час роботи прикладної програми, функція gethostbyaddr зазвичай приймає IP-адресу і повертає інформацію про хост. Перестановка байтів і додавання домену in-addr.arpa виконується функцією (gethostbyaddr) автоматично.

Підводячи підсумки можна сказати наступне.

В DNS є такі типи запитів: ітеративний (він же нерекурсивний), рекурсивний і зворотній.

Ітеративний запит відправляє доменне ім’я DNS-серверу і просить повернути або IP-адресу цього домену, або ім’я DNS-серверу, авторитетного для цього домену. При цьому сервер DNS не опитує інші сервери для отримання відповіді. Так працюють кореневі та сервери доменів першого рівня TLD-сервери (Top-Level Domain), тобто самого високого рівня після кореневого домена.

Рекурсивний запит відправляє доменне ім’я DNS-серверу і просить повернути IP-адресу цього домену. При цьому цей DNS-сервер може звертатися до інших DNS-серверів.

Зворотній запит відправляє IP-адресу і просить повернути доменне ім’я.

Будь-який DNS-сервер повинен відповідати на ітеративні запити. Можна налаштувати DNS-сервер відповідати і на рекурсивні запити. Якщо DNS-сервер не налаштований відповідати на рекурсивні запити, він їх обробляє як ітеративні.

Як правило, в локальній мережі є DNS-сервер, який обробляє рекурсивні запити, а також, скоріше за все, він налаштований на кешування запитів для економії трафіку і зниження навантаження на мережу.

В DNS є такі відповіді:

- авторитетна відповідь (authoritative response) - надходить від відповідальних за зону DNS-серверів.
- неавторитетна відповідь (non authoritative response) - надходить від серверів, які не відповідають за зону (від кешуючих).

1.6. Передача DNS-повідомлень

В службі DNS повідомлення передаються або UDP-дейтаграмами, або через віртуальний канал протоколу TCP. Дейтаграми є прийнятними для передачі запитів, оскільки вони менше завантажують мережу і мають більш високу продуктивність. Передача зони повинна виконуватись з використанням віртуального каналу, оскільки потрібно забезпечити надійність передачі.

Для доступу до сервера імен використовується порт 53 як для протоколу TCP, так і для протоколу UDP.

Повідомлення, що надсилаються з використанням UDP-протоколу, обмежуються 512 байтами (не враховуючи IP і UDP заголовки). Якщо у відповіді на запит програма отримує відповідь з встановленим бітом прапорців TC=1 (повідомлення обрізане), програма повторює запит, але вже з використанням протоколу TCP. Цей протокол застосовується також для зонних обмінів між первинним і вторинним DNS-серверами.

Повідомлення, відправлені з використанням протоколу UDP, можуть бути втрачені, тому застосовується стратегія повторної передачі. Запити і відповіді на них можуть переупорядковуватись в мережі або при обробці сервером імен, тому програми, які виконують перетворення, не повинні залежати від порядку надходження відповідей і повинні коректно розбирати такі ситуації.

1.6.1. Формат повідомлення DNS

DNS-запит і DNS-відповідь використовують однакову структуру заголовку, яка наведена на рисунку 5.7.

Повідомлення містить фіксований 12-байтовий заголовок, за яким розташовані чотири поля змінної довжини.

Значення в полі ідентифікації (identification) встановлюється клієнтом і повертається сервером. Це поле дозволяє клієнту визначати, на який запит надійшла відповідь.

16-бітне поле прапорців (flags) поділене на кілька частин, як показано на рисунку 5.8.

Рисунок 5.7 – Загальний формат DNS-запиту і DNS-відповіді

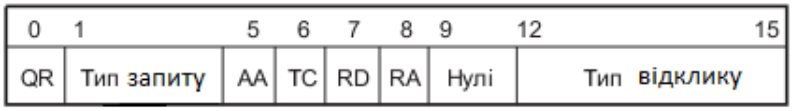


Рисунок 5.8 – Поле прапорців (flags) у заголовку DNS

- QR - (тип повідомлення), 1-бітове поле: 0 означає - запит, 1 означає - відповідь.
- Тип запиту (opcode), 4-бітне поле. Значення 0 - стандартний запит; 1 - інверсний запит і 2 - запит статусу сервера.
- AA - 1-бітний прапорець, який означає «авторитетну відповідь» (authoritative answer). Сервер DNS має повноваження для цього домену.
- TC - 1-бітове поле, яке означає «обрізано» (truncated). В випадку UDP це означає, що повна довжина відповіді перевищила 512 байтів, але були повернуті тільки перші 512 байтів відповіді.
- RD - 1-бітове поле, яке означає «потрібна рекурсія» (recursion desired). Біт може бути встановлений у запиті і потім повернутий у відповіді. Цей прапорець вимагає від DNS-сервера опрацювати цей запит самому (тобто сервер повинен сам визначити потрібну IP адресу, а не повертати адресу іншого DNS-сервера), що називається рекурсивним запитом (recursive query). Якщо цей біт не встановлений і запитуваний сервер DNS не має авторитетної відповіді, запитуваний сервер поверне список інших серверів

DNS, до яких необхідно звернутися, щоб отримати відповідь. Це називається запитом, що повторюється (iterative query).

- RA - 1-бітове поле, яке означає «рекурсія можлива» (recursion available). Цей біт встановлюється в 1 у посиланні, якщо сервер підтримує рекурсію. Більшість серверів DNS підтримують рекурсію, за винятком кореневих серверів, які не можуть обробляти рекурсивні запити з огляду на свою завантаженість.
- нуль - це 3-бітне поле повинно містити 0.
- тип відклику (rcode) - це 4-бітове поле коду повернення. Значення: 0 - немає помилок і 3 - помилка в імені. Значення «помилка в імені» повертається тільки від повноважного сервера DNS і означає, що імені домену, вказаного в запиті, не існує.

Розташовані далі чотири 16-бітові поля вказують на кількість пунктів в чотирьох полях змінної довжини, які завершують запис. В запиті кількість запитів (number of questions) зазвичай дорівнює 1, а решта три лічильника містять 0. У відклику кількість відповідей (number of answers) в крайньому випадку дорівнює 1, а решта два лічильника можуть бути як нульовими, так і ненульовими.

1.7. База даних DNS

Для кожного домену адміністратор веде базу даних DNS. База даних доменної системи імен DNS являє собою набір текстових файлів, які системний адміністратор супроводжує на основному сервері імен цього домену. Елементи бази даних називаються *записами про ресурси;* скорочено RR-записами. Типи і формати записів про ресурси регламентуються документами RFC-882, 1035 і 1183. Вторинний DNS-сервер періодично копіює бази даних DNS з основного сервера.

У файлах конфігурації сервера вказується, в якому саме файлі знаходяться дані, що описують зони, і чи є сервер первинним або вторинним для цієї зони.

Розглянемо деякі типи записів.

Запис SOA

SOA (Start Of Authority Record) - ресурсний запис DNS про сервер, який зберігає еталонну інформацію про домен.

Запис SOA позначає початок *зони* - групи записів про ресурси, розташовані в одній області простору імен DNS. До складу домена DNS зазвичай входить мінімум дві зони; одна служить для перетворення імен машин в IP-адреси, а інша - для зворотного перетворення.

Формат запису SOA:

ім’я [TTL] SOA Дані

Для кожної зони створюється лише один запис SOA; опис зони продовжується, поки не з’явиться наступний запис SOA. Запис SOA містить

ім’я зони, адреси, необхідні для встановлення технічних контактів, значення різних тайм-аутів. Цей запис повинен бути першим записом зони. Наприклад:

```
scs.kpi.ua.   IN SOA scs.kpi.ua. domainmaster.scs.kpi.ua. (
              2016072902; Serial (yymmdd{revision_count})
                    4h     ; Refresh in seconds (every 4 hours)
                    1h     ; Retry in seconds (every 1 hour)
                    7d     ; Expire in seconds (after 7 days)
                    1d )   ; Minimum in seconds (1 day)
```

Поле ім’я може містити символ @ для позначення імені поточної зони. В цьому прикладі можна було замість scs.kpi.ua. використати @.
Поле TTL відсутнє. Визначає «час життя» (time-to-live) для конкретного запису. Необов’язковий параметр. Якщо значення параметру у записі не вказане, то «час життя» визначається параметром default TTL.
Рекомендоване значення: 86400 (1d);
Клас - IN (Internet), тип - SOA, а решта елементів складають поле *дані.*
Сервер scs.kpi.ua- основний сервер імен цієї зони.
Запис domainmaster.scs.kpi.ua. вказує адресу електронної пошти для технічних контактів з адміністратором у форматі *користувач.машина*, (а не *користувач@машина*). Якщо потрібно відправити пошту адміністратору домену, треба замінити першу крапку знаком @ і видалити останню крапку.

Перший числовий параметр serial - порядковий номер блоку інформації про конфігурацію зони. Це може бути будь-яке ціле число, яке повинно збільшуватися при кожній зміні файлу даних зони. Ці порядкові номери не обов’язково мають бути послідовними, але повинні монотонно збільшуватись. Допоміжні сервери будуть копіювати нові дані тільки у тому випадку, якщо порядковий номер запису в SOA основного сервера буде більшим порядкових номерів записів, які зберігаються допоміжними серверами (формальне обмеження на величину порядкового номеру тільки одне: число справа від десяткової крапки повинно бути менше 9999).

Наступні чотири елементи - значення тайм-аутів (в секундах), які керують проміжком часу, на протязі якого дані можна буде кешувати в різних точках всесвітньої бази даних DNS.

Перший елемент - тайм-аут регенерації, **refresh**. Він показує, як часто допоміжні сервери повинні звірятися з основним сервером і перевіряти, чи не змінився порядковий номер конфігурації зони. Якщо порядковий номер змінився, допоміжні сервери повинні створити новий екземпляр даних зони . Загально прийняте значення для даного тайм-ауту - від однієї до шести годин (3600 - 21600 секунд).

Якщо допоміжний сервер намагається перевірити порядковий номер основного, а той не відповідає, допоміжний сервер робить повторну спробу після закінчення періоду часу, заданого тайм-аутом **retry**. Досвід показує, що

прийнятне значення для даного параметру - від 20 до 60 хвилин (1200 - 3600 секунд).

Якщо основний сервер на протязі тривалого часу відключений, то допоміжні сервери будуть багаторазово намагатися оновити свої дані, але ці спроби завжди будуть закінчуватися невдачею. В кінці кінців, допоміжні сервери вирішать, що основний сервер не включиться вже ніколи і що його дані, напевно, застаріли. Параметр **expire** визначає, як довго допоміжні сервери будуть продовжувати обслуговувати цей домен при відсутності основного серверу. Система повинна вижити, навіть якщо основний сервер не буде працювати довгий час, тому параметру **expire** слід надавати досить велике значення (рекомендується вибирати від тижня до місяця).

Параметр **minimum** задає час життя записів про ресурси, яке буде прийматися за замовчуванням. Він кешується разом з записами і використовується для відміни їхньої дії на неавторитетних серверах. Кожен запис у полі *час* містить явно задане значення; якщо таке значення не встановлене, то використовується значення із запису SOA.

Записи NS

Записи NS описують сервери, які авторитетні для даної зони. Зазвичай ці записи розташовані за записом SOA. Запис NS має такий формат:

зона [*час*] [*клас*] NS *ім'я_машини*

Наприклад:

```
scs.kpi.ua.      IN ns ns.scs.kpi.ua.
scs.kpi.ua.      IN ns ns1.scs.kpi.ua.
scs.kpi.ua.      IN ns ns2.scs.kpi.ua.
```

Записи A

Записи A забезпечують перетворення доменних імен хостів в IP-адреси. Раніше таке перетворення виконувалось тільки за допомогою файлу **/etc/hosts**. Для кожного мережевого інтерфейсу комп'ютера має бути зроблений один запис A. Цей запис має такий формат:

ім'я_машини [час] [клас] A ip-адреса

Наприклад:

```
eternity    IN    A    10.15.18.161
```

Записи PTR

Записи PTR виконують зворотне перетворення IP-адрес в імена машин. Як і у випадку із записами A, для кожного мережевого інтерфейсу машини має бути зроблений один запис PTR.

Загальний формат запису PTR такий:

адреса [час] [клас] PTR ім'я_машини

Запис PTR в домені IN-ADDR.ARPA, що відповідає вище наведеному запису A для машини eternity, буде мати такий вигляд:

10.15.18.161 IN PTR eternity.scs.kpi.ua.

Зворотні відповідності в записах PTR в домені IN-ADDR.ARPA, використовуються всіма програмами, які аутентифікують вхідний мережевий трафік. Наприклад, цільова машина отримує по мережі повідомлення від хоста з деякою IP-адресою. Користуючись послугами DNS, вона перетворює цю IP-адресу в ім'я і порівнює його зі списками імен у своїх файлах конфігурації.

Програми netstat, sendmail, X Window, syslog, finger, ftp, rlogind отримують імена машин і IP-адрес за допомогою зворотного перетворення.

Важливо, щоб записи A відповідали записам PTR. Невідповідність і відсутність останніх призводить до тайм-аутів, в результаті чого система починає працювати повільно.

Записи MX

Основний запис в доменній зоні, який вказує на імена поштових серверів цього домену. Без запису *mx* неможливе отримання і відправка пошти. Неможливість відправки пояснюється тим, що більшість поштових серверів перш ніж прийняти повідомлення, з метою захисту від спаму обов'язково перевіряють наявність в DNS-зоні адресата *mx*-записів і їхню відповідність IP-адресі відправника. Якщо такого запису немає або якщо адреса не відповідає вказаній, віддалений поштовий сервер з великою вірогідністю відмовить в прийомі електронної пошти. Запис *mx* підтримує пріоритети. Пріоритет *mx* - ціле число від 0 до 65535, але зазвичай виставляють 10, 20, 30 і т.д. Пріоритети можуть бути однаковими, тоді DNS або балансує ці записи, якщо це налаштовано на DNS-сервері, або відправник буде використовувати перший доступний сервер. Зауважимо, що адресою поштового сервера в *mx*-записі не може бути IP-адреса, тільки доменне ім'я.

В наступному прикладі вищий пріоритет має сервер mx.scs.kpi.ua.
Запис mx має такий формат:
ім'я домену[час][клас] mx пріоритет ім'я_серверу

```
scs.kpi.ua.   IN mx    5    mx.scs.kpi.ua.
scs.kpi.ua.   IN mx   20    mx.kpi.ua.
```

Записи CNAME

Канонічне ім'я - це синонім другого доменного імені. Даний тип доменного запису може бути корисним, наприклад, коли існує кілька доменних імен, які в результаті перетворення посилаються на одну й ту ж IP-адресу. У цьому випадку достатньо лише одного запису, що явно перетворюється на IP-адресу, решту ж можна зробити її синонімами. Це може бути корисним при переході на іншу IP-адресу. В цьому випадку достатньо буде внести зміни тільки в один запис. Всі синоніми будуть вказувати на нову IP-адресу автоматично.

Також мнемоімена корисні у випадку, коли ім'я машини змінилося і потрібно дозволити користувачам, які знають старе ім'я, отримати доступ до машини.

Запис CNAME має такий формат:

мнемоім'я [час] [клас] CNAME ім'я_машини

Наприклад:

```
smtp        IN    CNAME    scs.kpi.ua.
```

Якщо у машини є записи CNAME, які містять її мнемонічні імена, інші записи для даної машини повинні посилатися на її реальне ім’я, а не на мнемонічне. Коли програми DNS зустрічають запис CNAME, вони зупиняють свої запити по мнемонічному імені і переключаються на реальне ім'я.

Записи TXT

Запис TXT використовується для додавання довільного тексту до DNS-записів .

Запис TXT має такий формат:

ім'я [час] [клас] TXT *інформація*

Наприклад, такий запис підтверджує повноваження поштового сервера:

```
scs.kpi.ua  IN  TXT  “v=spf1 mx ip4:77.47.131.42  –all”
```

1.8. Програма nslookup

Ця програма поширюється в комплекті BIND і доступна у багатьох операційних системах.

Переважно *nslookup* використовується для відправлення запитів тим же способом, який використовується DNS-клієнтом. Але іноді *nslookup* може використовуватися для відправки запитів DNS-серверам, як це робить не клієнт, а власне DNS-сервер.

Програма nslookup (зазвичай – /usr/sbin/nslookup в Linux) дозволяє провести DNS-перетворення в явному вигляді. Наприклад:

```
%nslookup www.lib.ru
Server: ns.kpi.ua
Address: 195.245.194.34

Name:www.lib.ru
Address: 204.146.18.33
```

Відповідь програми означає, що був опитаний DNS-сервер ns.kpi.ua (його IP-адреса 195.245.194.34) і отримана IP-адреса www.lib.ru 204.146.18.33.

Приклад зворотного перетворення:

%nslookup 81.176.66.163
Server: ns.kpi.ua
Address: 195.245.194.34

Name:www.lib.ru
Address:204.146.18.33

Програма **nslookup** працює також в режимі командного рядка.

Розглянемо деякі команди:

server [ім'я_сервера, який опитується]

lserver [ім'я_сервера, який опитується]

Перша команда означає змінити опитуваний DNS сервер, наприклад: **server ns.kiev.ua**. Без аргументу - встановити сервер за замовчуванням, поточний сервер.

Всі запити (окрім команди lserver) відправляються до поточного серверу, встановленому в даний момент.

nslookup дозволяє напряму звертатися з запитами до серверів, що безпосередньо відповідають за ту або іншу зону. Якщо ж відповідь надійшла від серверу, що не відповідає за зону, для хоста якого запрошувалася інформація (наприклад, дані були взяті з кеша), така відповідь буде помічена як «non-authoritative answer».

server і **lserver** відрізняються тим, що при зміні серверу командою **server** адреса нового серверу перетвориться за допомогою поточного серверу, а команда **lserver** виконує таке ж перетворення за допомогою серверу, встановленого для nslookup за замовчуванням – «свого» серверу. Це має значення, коли поточний сервер з якоїсь причини не відповідає на запити.

set type=тип_даних

- встановити режим запиту даних певного типу. Наприклад:

>**set type=NS**
>lib.ru

означає запит списку DNS-серверів, що відповідають (authoritative) за домен lib.ru. (Запит в цьому випадку повинен складатися з імені домена, а не окремого хоста.)

Можливі типи:

`set recurse`

відправляти рекурсивні запити (вибрано за замовчуванням).

`set norecurse`

відправляти ітеративні запити.

`set domain=ім'я_домена`

встановити ім'я домена, що додається до неповних доменних імен (за умовчанням береться з `/etc/resolv.conf`).

`set debug`
детально показувати вміст вхідних відповідей.

`set nodebug`
відмінити `set debug` (відмінено за замовчуванням).

`set d2`
детально показувати вміст запитів, що відправляються.

`set nod2`
відмінити `set d2` (відмінено за замовчуванням).

`set all`
показати значення всіх опцій.

`ls ім'я_домена`
вивести перелік хостів вказаного домена, наприклад `ls vvsu.ru`. Заздалегідь слід перейти на опитування сервера, що відповідає `(authoritative)` за даний домен. З причин безпеки деякі сервери не виконують цю команду.

`help`
допомога.

`exit`
вихід.

Будь-які введені дані, що не є командою, сприймаються як запит.

Направлення результатів виконання команди у файл виконується за допомогою символу > .

Виконайте із командного рядка команду
%nslookup -type=ns .
Поясніть отриманий результат.

1.9. Програма dig

dig - це ще одна програма для роботи з даними сервера DNS в домені. При використанні **dig** всі аспекти поведінки і створення запитів визначаються в командному рядку, оскільки діалоговий режим роботи в **dig** не реалізований.

Доменне ім'я, для якого виконується пошук, – це перший аргумент, тип запиту (наприклад, *a* - при пошуку адресних записів, *mx* - при пошуку MX-записів) – це другий аргумент; за замовчуванням виконується пошук адресних записів. DNS-сервер, якому відправляються запити, вказується після символу «@». Тут можна використовувати доменне ім'я або IP-адресу. За

замовчуванням запити направляються DNS-серверам, перерахованим в r*esolv.conf.*

Аргументи можна записувати у будь-якому порядку, а ***dig*** самостійно визначить, що ***mx*** це, скоріше за все, тип запису, а не доменне ім'я, для якого виконується пошук.

Так команда

% **dig scs.kpi.ua**

виконує пошук адресних записів для *scs.kpi.ua;* запити відправляються першому DNS-серверу із перерахованих в *resolv.conf.*

Команда

% **dig scs.kpi.ua mx**

виконує пошук MX-записів для *scs.kpi.ua* через той же DNS-сервер.

dig надає повні повідомлення-відповіді DNS, включаючи спеціально виділені розділи (заголовку, запиту, відповіді, авторитетності і типу), а також надає RR-записи у форматі майстер-файлу. Це зручно, якщо потрібно скористатися виведеними даними діагностики для створення файлу даних зони, або файлу посилань на кореневі сервери.

Наприклад, результат наступної команди наведено на рис. 5.9.

dig google.com.ua

```
[root@scs ~]# dig google.com.ua

; <<>> DiG 9.4.-ESV <<>> google.com.ua
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36646
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 8

;; QUESTION SECTION:
;google.com.ua.                 IN      A

;; ANSWER SECTION:
google.com.ua.          291     IN      A       172.217.20.163

;; AUTHORITY SECTION:
google.com.ua.          72316   IN      NS      ns3.google.com.
google.com.ua.          72316   IN      NS      ns1.google.com.
google.com.ua.          72316   IN      NS      ns4.google.com.
google.com.ua.          72316   IN      NS      ns2.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         85311   IN      A       216.239.32.10
ns1.google.com.         96917   IN      AAAA    2001:4860:4802:32::a
ns3.google.com.         85441   IN      A       216.239.36.10
ns3.google.com.         85473   IN      AAAA    2001:4860:4802:36::a
ns4.google.com.         85441   IN      A       216.239.38.10
ns4.google.com.         88162   IN      AAAA    2001:4860:4802:38::a
ns2.google.com.         85441   IN      A       216.239.34.10
ns2.google.com.         85473   IN      AAAA    2001:4860:4802:34::a

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Feb  7 10:13:05 2018
;; MSG SIZE  rcvd: 305

[root@scs ~]#
```

Рисунок 5.9 – Результат виконання команди **dig google.com.ua**

1.10. Файл /etc/named.conf

Конфігурація DNS-сервера описується в конфігураційному файлі /etc/named.conf (BIND v.8). Конфігураційний файл містить інформацію про всі зони, для яких DNS-сервер є первинним або вторинним. Крім імені зони, в першому випадку вказується файл, в якому міститься база даних DNS для даної зони, у другому - адреса первинного серверу і файл тимчасового зберігання бази даних, отриманої від первинного серверу. До числа зон входять зони (домени), в яких сервер здійснює прямий пошук («ім'я в адресу»), зони реверсивного пошуку («адреса в ім'я», домен *.in-addr.arpa) і зона локальної петлі (0.0.127.in-addr.arpa, яка служить для перетворення адреси 127.0.0.1 в ім'я localhost). Також у конфігураційному файлі вказується файл ініціалізації кешу і каталог, в якому розташовані всі файли з даними.

Рядок конфігураційного файлу /etc/named.conf формується за принципом:

зона <ім'я зони>
тип_серверу
розміщення_даних,

де *тип_серверу*: master чи slave (кожний сервер кешує дані, що проходять через нього), *розміщення_даних:* файл - для первинного серверу; первинний сервер і файл тимчасового зберігання - для вторинного серверу.

Приклад конфігураційного файлу для первинного серверу домену fpm.kpi.ua:

```
options {
        directory      "/etc/namedb";

zone "." {
        type hint;
        file "named.root";
};
zone "0.0.127.IN-ADDR.ARPA" {
        type master;
        file "pri/localhost.rev";
};
zone "fpm.kpi.ua" {
        type master;
        file "pri/fpm.kpi.ua";
        also-notify { 10.7.1.23; 10.7.1.24; };
};
```

Приклад файлу /pri/fpm.kpi.ua бази даних DNS для зони наведено на рис. 5.10.

```
; $Id: fpm.kpi.ua,v 1.1 2009/09/07 17:03:28 fm Exp fm $
;
; zone fpm.kpi.ua
;

$TTL    1d
$ORIGIN fpm.kpi.ua.
@       IN      SOA     fpm.kpi.ua. domainmaster.fpm.kpi.ua. (
                        2016121402      ; Serial (yymmdd(revision_count))
                        4h      ; Refresh in seconds (every 4 hours)
                        1h      ; Retry in seconds (every 1 hour)
                        7d      ; Expire in seconds (after 7 days)
                        1d )    ; Minimum in seconds (1 day)

        IN      NS      fpm.kpi.ua.

        IN      A       10.15.17.1

www     IN      A       10.15.18.122
pmai    IN      A       10.15.18.121
scs     IN      A       10.15.18.111
mis     IN      A       10.7.10.102
```

Рисунок 5.10 – Приклад файлу /pri/fpm.kpi.ua бази даних DNS

1.11. Файл named.root

Файл named.root. містить IP-адреси кореневих серверів ієрархії DNS, з опитування яких починається процедура пошуку інформації (якщо потрібна інформація відсутня в кеші або у власній базі даних локального сервера DNS). Для оновлення файлу named.root потрібно отримати список всіх кореневих серверів, тобто серверів, що відповідають за домен «.» (крапка). Отримати такий список можна за допомогою програми nslookup (результат направити у файл, файл відредагувати по формату named.root).

Завдання

1. Виконати пряме перетворення для вказаного викладачем доменного імені. Звернути увагу на наявність канонічного доменного імені і псевдоніма (alias). Виконати зворотне перетворення для отриманої IP-адреси. Перетворення виконувати шляхом посилки ітеративних запитів, не забудьте вказати абсолютні доменні імена. Виконати пряме і зворотне перетворення для scs.kpi.ua.
2. Налаштувати хост так, щоб він звертався до іншого DNS-серверу своєї зони. Список серверів своєї зони знайти.
3. Для вказаного викладачем хосту знайти зону DNS, до якої він належить; сервери DNS, які її обслуговують; тимчасові характеристики взаємодії первинного і вторинного серверів для цієї зони; можливі псевдоніми даного хосту. Одержувати тільки авторитетні відповіді.

Контрольні запитання

1. Призначення системи DNS.
2. Простір імен DNS.

3. Структури бази даних DNS.
4. Типи серверів DNS, їхнє призначення та особливості функціонування.
5. Перетворення доменних імен в IP-адреси.
6. Зворотні перетворення.
7. Особливості рекурсивного запиту на перетворення доменного імені.
8. Особливості нерекурсивного перетворення доменного імені.
9. Повідомлення DNS, призначення полів повідомлення, формат заголовку.
10. Призначення та особливості функціонування програми nslookup.
11. Записи бази даних DNS, їх типи, призначення та формат.
12. Призначення та особливості функціонування програми dig.
13. Конфігурування DNS-сервера.

Лабораторна робота 6
Засвоєння принципів перетворення DNS-імен в IP-адреси

Мета роботи: використовуючи програму моделювання комп'ютерних мереж засвоїти принципи адресації на канальному та мережевому рівнях моделі OSI, принципи динамічного призначення IP-адрес і принципів перетворення DNS-імен в IP-адреси.

План виконання лабораторної роботи

Завдання №1. Побудова локальної мережі з двома робочими станціями.
Завдання №2. Засвоєння принципів адресації на канальному і мережевому рівнях.
Завдання №3. Засвоєння принципів динамічного призначення IP-адрес.
Завдання №4. Засвоєння принципів перетворення DNS-імен в IP-адреси.
Завдання №5. Ознайомлення з відомостями про структуру перехресного кабеля.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

Для виконання завдань лабораторної роботи використовується програма Cisco Packet Tracer. Cisco Packet Tracer, це багатофункціональна програма моделювання мереж, яка дозволяє експериментувати з поведінкою мережі та оцінювати можливі сценарії. Packet Tracer полегшує вивчення складних технологічних принципів та надає можливість виконувати дії, які розвивають глибоке розуміння мережевих технологій.

Packet Tracer дозволяє студентам створювати мережі з практично необмеженою кількістю пристроїв, а викладачам – легко описати і показати складні технічні принципи та проєкти мережевих систем.

Завдання №1. Побудова локальної мережі з двома робочими станціями

Відкрити програму Cisco Packet Tracer, вибрати End Devices і перетягнути мишкою на робоче поле дві робочі станції Generic (рисунок 6.1).

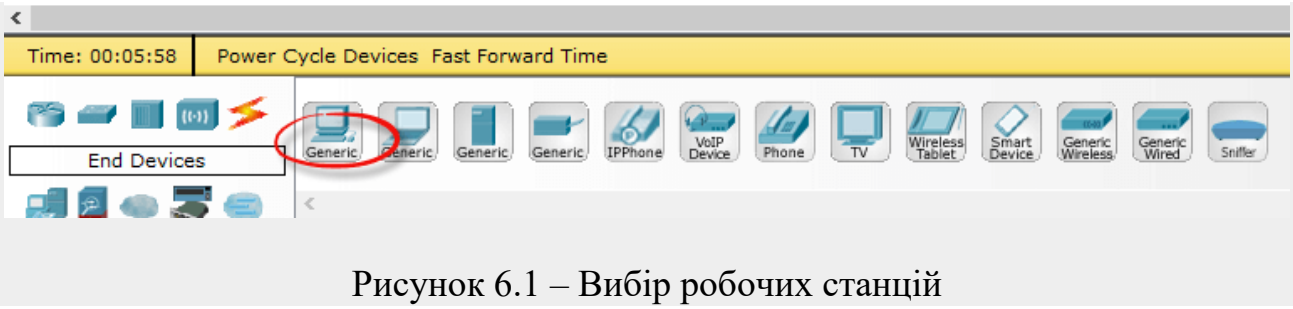


Рисунок 6.1 – Вибір робочих станцій

В результаті отримана наступна конфігурація (рисунок 6.2).

Рисунок 6.2 – Вибрані комп’ютери

Далі потрібно з’єднати дві робочі станції кабелем. Для цього вибрати Connections і перехресний кабель (рисунок 6.3):

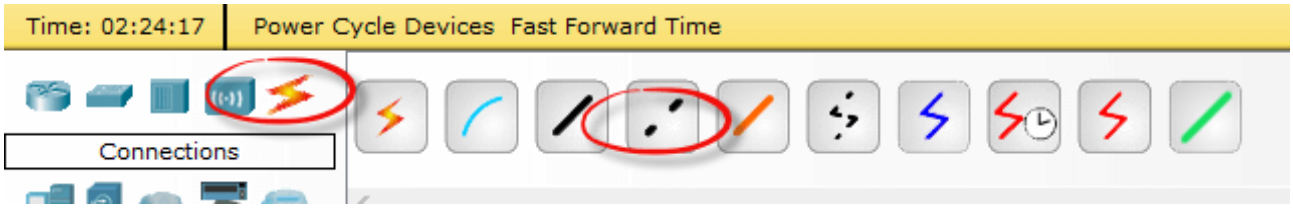


Рисунок 6.3 – Вибір перехресного кабелю

Причини використання перехресного кабелю описані в розділі «Використання перехресного кабелю».

Мишкою вибрати першу робочу станцію і підключити кабель до FastEthernet0 (рисунок 6.4):

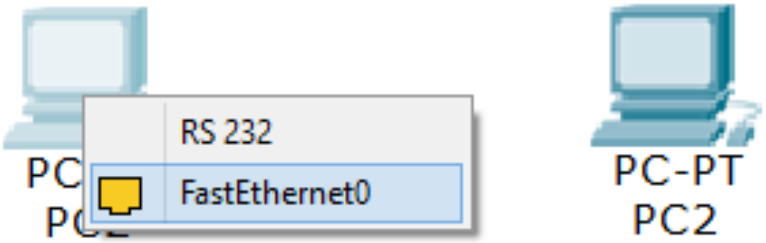


Рисунок 6.4 – Під’єднання кабелю до першої робочої станції

Перетягнути мишкою з’єднання на другу робочу станцію і вибрати також FastEthernet0 (рисунок 6.5):

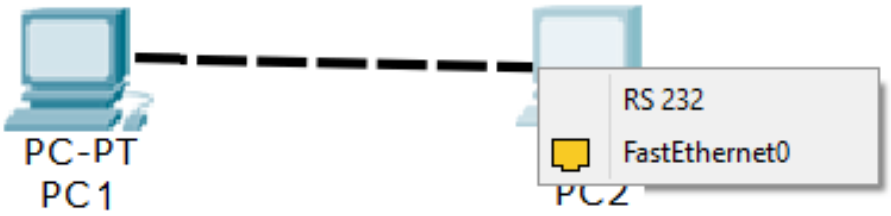


Рисунок 6.5 – Під’єднання кабелю до другої робочої станції

В результаті можна побачити, що з’єднання між робочими станціями відбулося – засвітилися зелені індикатори link (рисунок 6.6):

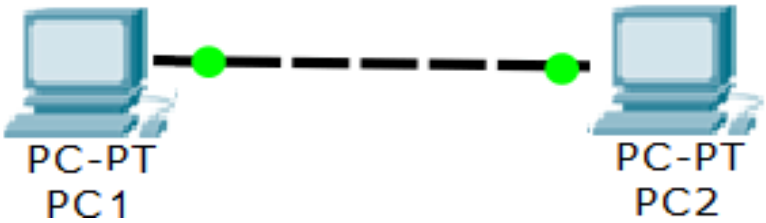


Рисунок 6.6 – Дві робочі станції, що з’єднані кабелем

Тепер потрібно вказати статичну IP-адресу комп'ютера. Для цього зробити подвійний клік по першій робочій станції, перейти в меню Desktop і вибрати режим IP Configuration (рисунок 6.7).

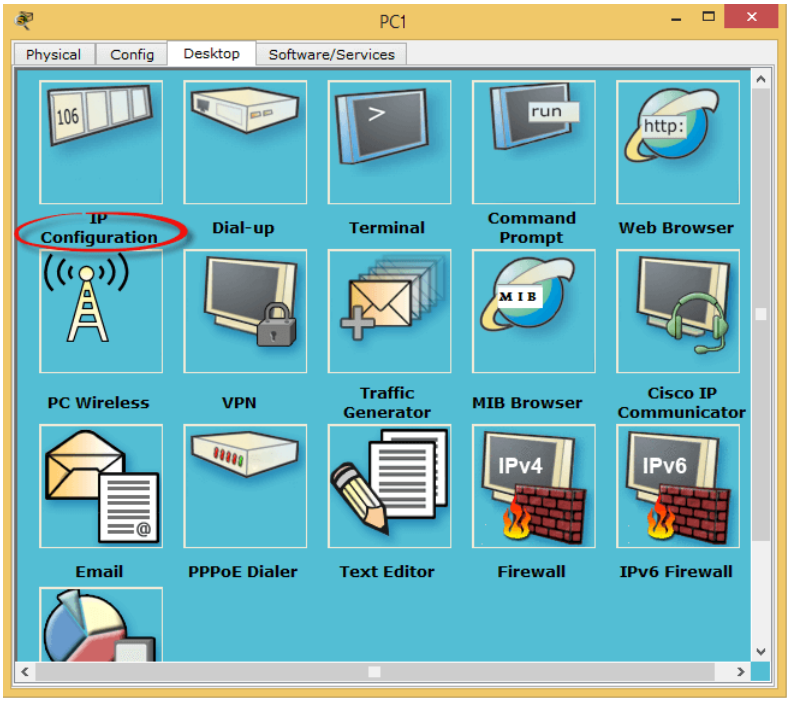


Рисунок 6.7 – Вибір вікна IP Configuration

Після цього задати IP-адресу і маску, як показано на рисунку 6.8.

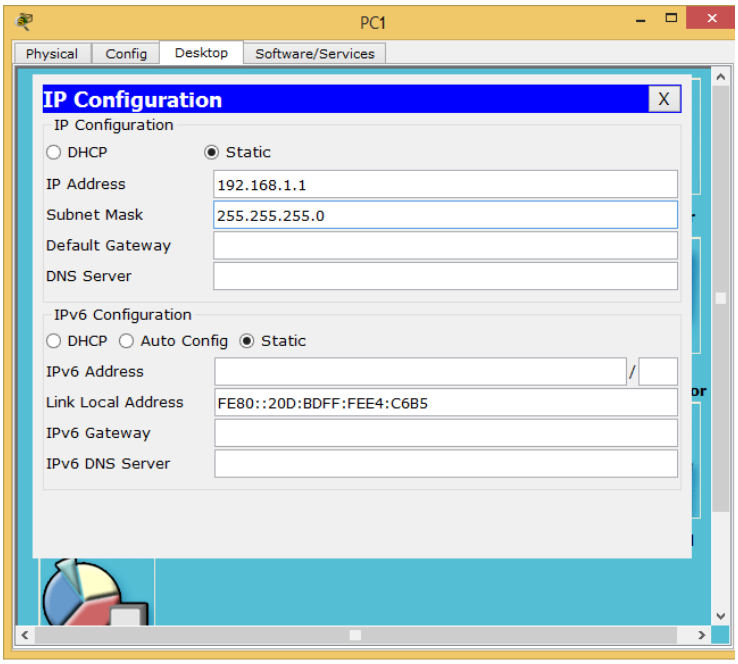


Рисунок 6.8 – Призначення IP-адреси робочій станції 1

На другій робочій станції виконати такі ж дії, але вказати IP-адресу 192.168.1.2 (рисунок 6.9).

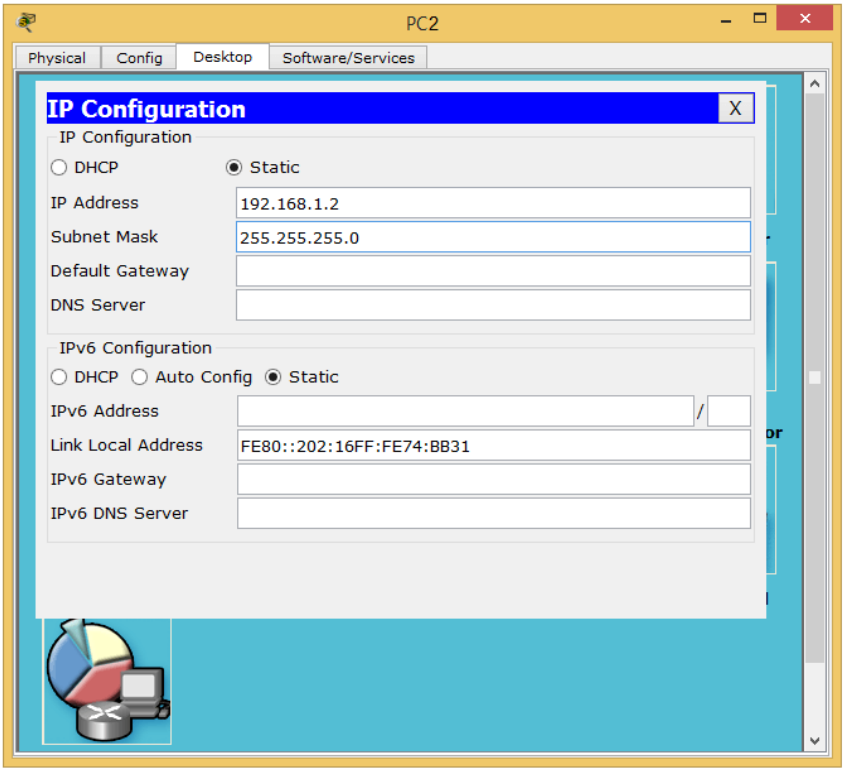


Рисунок 6.9 – Призначення IP-адреси робочій станції 2

Тепер на другій робочій станції вибрати режим Command Promt (рисунок 6.10).

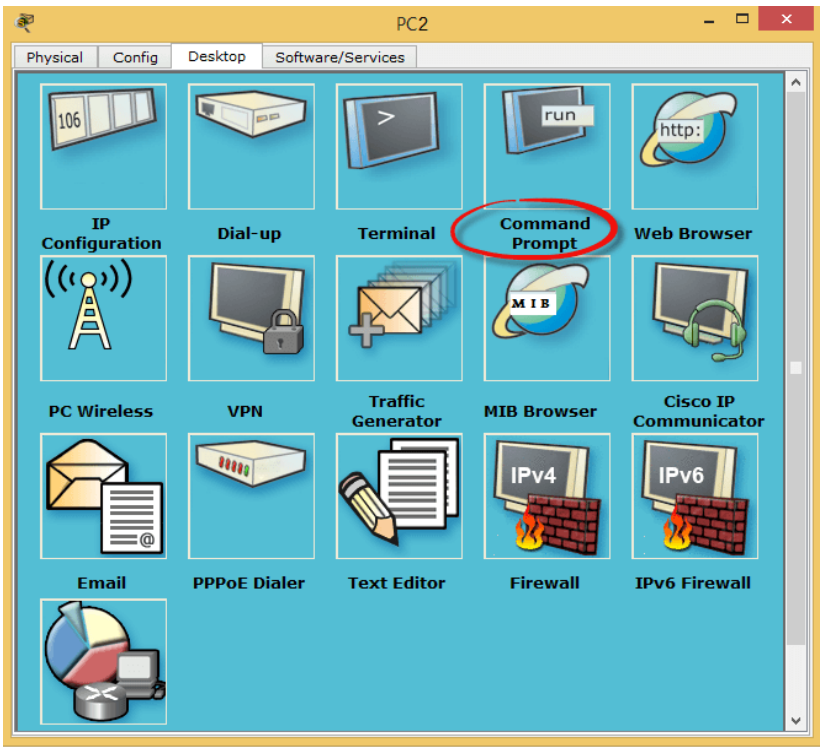


Рисунок 6.10 – Перехід до командного рядка

Відкривається командний рядок. З командного рядка виконати команду **ping 192.168.1.1**, в результаті виконання якої видно, що зв'язок між робочими станціями встановлений.

ping - утиліта для перевірки з'єднань в мережах TCP/IP. Утиліта відправляє запити з використанням протоколу ICMP вказаному вузлу мережі (ICMP Echo-Request) і фіксує відповіді, що повертаються (ICMP Echo-Reply). Час між відправкою запиту і отриманням відповіді RTT дозволяє визначити затримки передачі обраним маршрутом і частоту втрати пакетів, тобто опосередковано визначити завантаженість каналів передачі даних і проміжних пристроїв. Повна відсутність ICMP-відповідей може також означати, що віддалений вузол (або будь-який проміжний маршрутизатор) блокує ICMP Echo-Reply або ігнорує ICMP Echo-Request запити.

На станції PC1 створено пакет (конверт), який чекає початку просування його мережею. Запустити просування пакету покроково можна, натиснувши на кнопку «*Capture/Forward*» (Вперед) у вікні симуляції. Якщо натиснути на кнопку «*Auto Capture/Play*» (відтворення), то можна спостерігати весь цикл проходження пакету мережт.. В закладці «*Event list*» (Список подій) можна бачити успішний результат «пінгування» (рисунок 6.11 та 6.12).

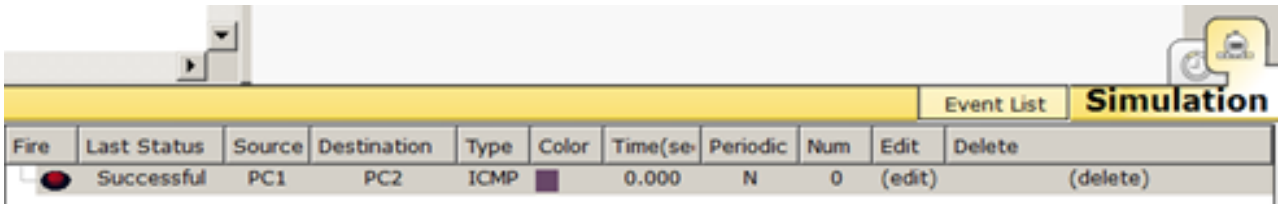


Рисунок 6.11 – З’єднання між робочими станціями PC1 і PC2 встановлене

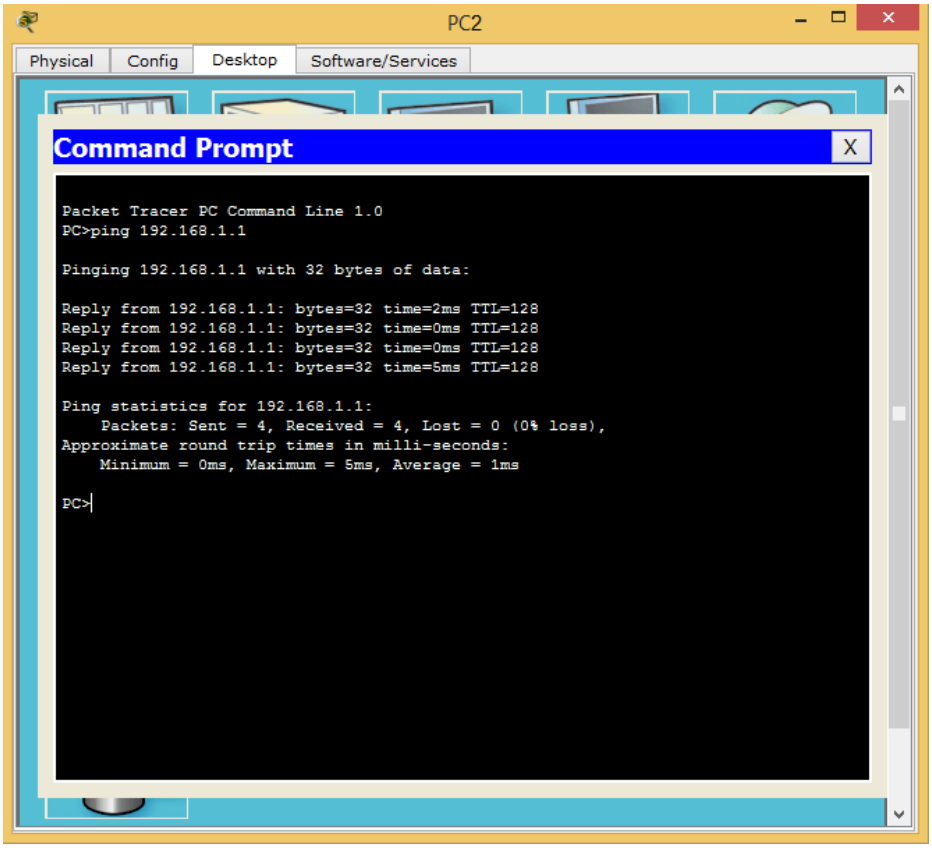


Рисунок 6.12 – Результат виконання утиліти ping при встановленому з’єднанні

1.1. Модель OSI в Cisco Packet Tracer

В режимі симуляції можна не тільки відслідковувати протоколи, що використовуються, але й бачити, на якому з семи рівнів еталонної моделі OSI даний протокол задіяний.

Натискання кнопкою миші на пакеті показує додаткову інформацію про передавання пакету мережею. На вкладці *OSI Model* (Модель OSI) представлена інформація про рівні OSI, на яких працює даний мережевий пристрій (рисунок 6.13).

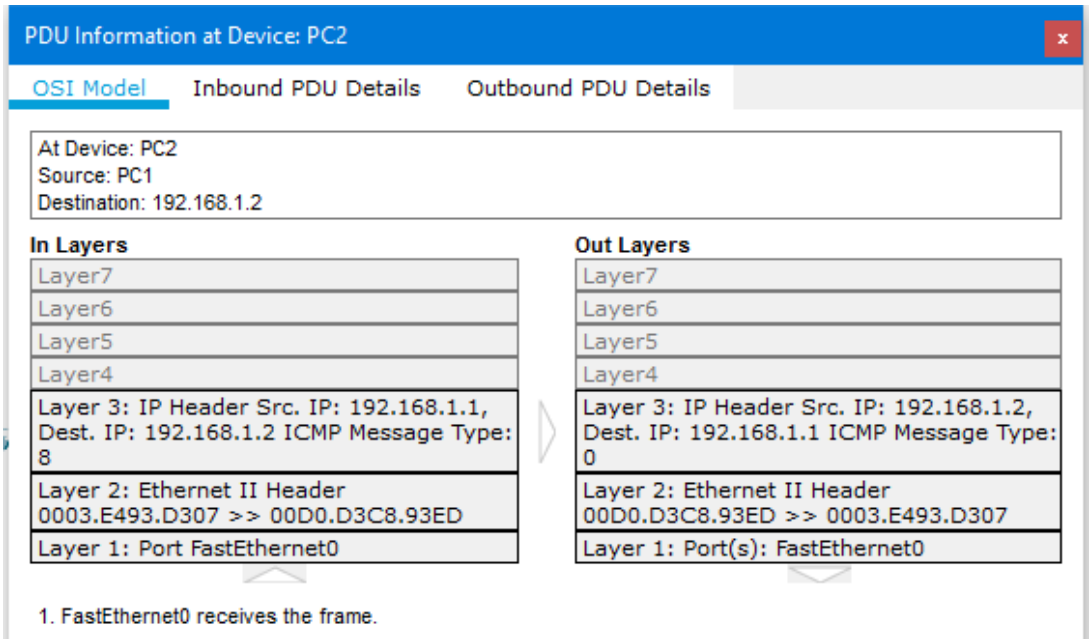


Рисунок 6.13 – Моніторинг руху пакету рівнями моделі OSI

На другій та третій вкладках можна побачити структуру відповідно вхідного та вихідного пакетів (на рис. 6.14 вибрана вкладка зі структурою вхідного пакету).

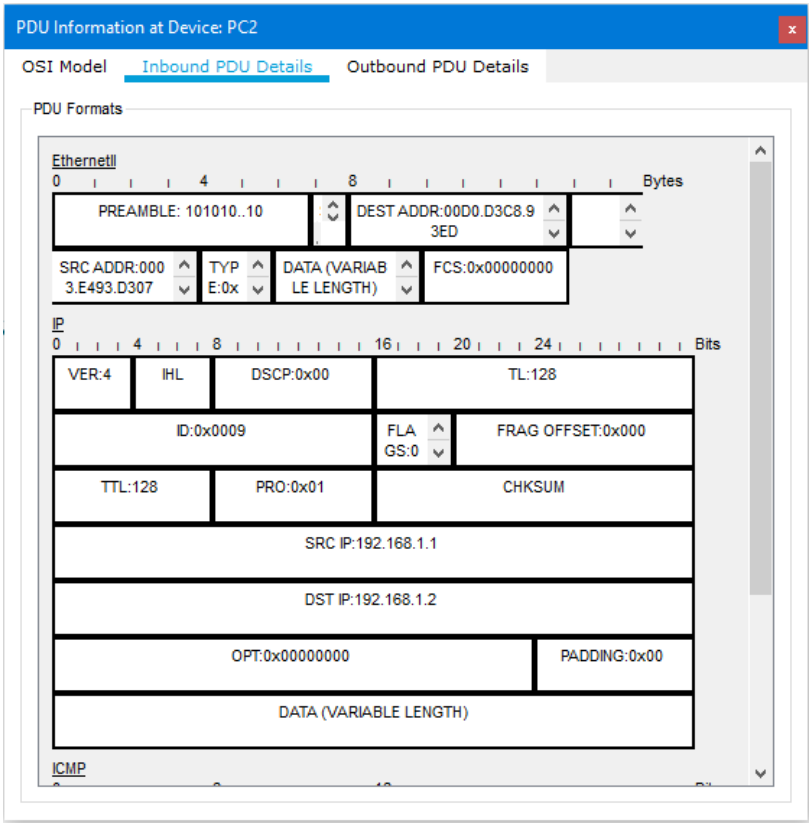


Рисунок 6.14 – Структура вхідного пакету

Завдання №2. Засвоєння принципів адресації на канальному та мережевому рівнях

Засвоєння принципів MAC-адресації та визначення MAC-адрес на основі IP-адрес.

1. За допомогою програми Packet Tracer побудувати мережу як показано на рисунку 6.15.

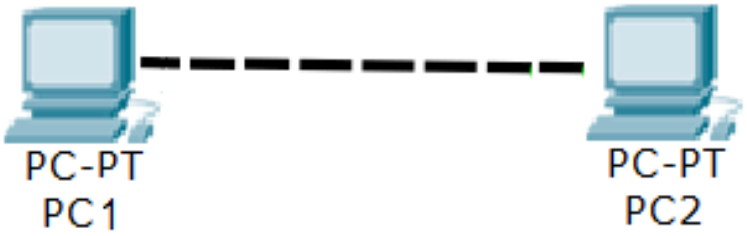


Рисунок 6.15 – Структура мережі для завдання 2

2. Призначити робочим станціям IP-адреси та маску. Перші три байти IP-адрес збігаються з адресою мережі, а останній байт дорівнює відповідно 1 та 2.
3. Визначити MAC-адресу кожної робочої станції за допомогою команди **ipconfig /all** з командного рядка.
4. На робочій станції PC1 з командного рядка виконати команду **arp –a.** Зафіксувати отриманий результат.
5. На робочій чтанції PC1 з командного рядка виконати команду **ping** на адресу счтанції PC2, після чого знову виконати команду **arp –a.** Пояснити отриманий результат.
6. На робочій станції PC1 з командного рядка виконати команду **arp –d.** Пояснити призначення ключа –d.
7. Перейти в режим "Simulation". Із командного рядка робочої станції PC1 виконати команду **ping** на IP-адресу станції PC2. На «Simulation panel» натиснути кнопку «CaptureForward». Пояснити призначення пакетів, що відправляються зі станції PC1. Натиснути кнопкою миші на кожному із створених пакетів і переглянути їх вміст. Звернути увагу на адреси відправника та отримувача. Натискаючи на «CaptureForward» прослідкувати за формуванням та передачею створених пакетів мережею, одночасно спостерігаючи за командним рядком. Пояснити призначення ARP-пакету.
8. Зробити висновки.

Завдання №3. Засвоєння принципів динамічного призначення IP-адрес

Короткі теоретичні відомості

Протокол динамічного конфігурування DHCP (Dynamic Host Configuration Protocol) працює відповідно до технології клієнт-сервер. Процес отримання мережевих налаштувань відбувається в кілька етапів і описується схемою DORA (Discover-Offer-Request-Acknowledge):

Discover (Виявлення)

Клієнт DHCP підключається до мережі і починає пошук DHCP-сервера, для чого відправляє запит DHCPDISCOVER на широкомовну адресу 255.255.255.255. Адреса клієнта в цьому запиті 0.0.0.0, оскільки своєї адреси у клієнта ще немає. Також в запиті клієнт вказує свою MAC-адресу. Запит надходить всім модулям, які знаходяться в даному сегменті мережі, але відповідають на нього тільки DHCP-сервери.

Offer (Пропозиція)

DHCP-сервер, який отримав запит DHCPDISCOVER, аналізує його зміст, вибирає підходящу конфігурацію мережі та відправляє її в повідомленні DHCPOFFER**.** Зазвичай DHCPOFFER відправляється на MAC-адресу клієнта, вказану в DHCPDISCOVER, хоча іноді може використовуватися широкомовне розсилання. Якщо в мережі знаходиться кілька DHCP-серверів, то клієнт отримує кілька відповідей DHCPOFFER і вибирає з них одну, як правило, отриману першою.

Request(Запит)

Отримавши відповідь сервера, клієнт відповідає повідомленням DHCPREQUEST**,** в якому «офіційно» запитує у сервера надані йому налаштування. В повідомленні DHCPREQUEST міститься та ж інформація, що і в DHCPDISCOVER, а також IP-адреса вибраного DHCP-сервера. DHCPREQUEST надсилається на широкомовну адресу і ті DHCP-сервери, адреса яких відсутня в повідомленні, розуміють, що їхня пропозиція відкинута.

Acknowledge (Підтвердження)

DHCP-сервер, адреса якого вказана в DHCPREQUEST, отримує повідомлення і розуміє, що його обрали. Він фіксує прив’язку до клієнта і відповідає повідомленням DHCPACK, підтверджуючи видані клієнту налаштування. DHCPACK надсилається на MAC-адресу клієнта, вказану в DHCPREQUEST. Клієнт отримує повідомлення DHCPACK, перевіряє налаштування і застосовує конфігурацію, яка була отримана в повідомленні DHCPOFFER.

При виконанні пересилок DHCP клієнт використовує протокол UDP, порт 68, DHCP сервер – UDP, порт 67.

Рисунок 6.16 – Приклад взаємодії клієнта з DHCP-сервером

Завдання.

1. За допомогою програми Packet Tracer побудувати мережу, структура якої наведена на рисунку 6.17.

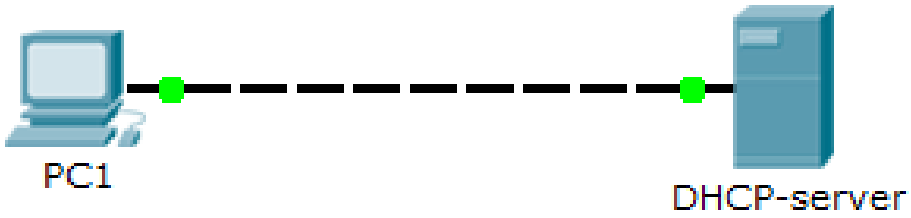


Рисунок 6.17 – Структура мережі для завдання 3

2. Перейти до режиму «Simulation» .

3. Вибрати режим конфігурування сервера. Призначити IP-адресу, перші три байти якої збігаються з адресою мережі, а останній байт дорівнює 100.

4. У режимі конфігурування перейти до вкладки DHCP і налаштувати таким чином: у полях *Default Gateway* та *DNS Server* ввести адреси, перші три байти яких збігаються з адресою мережі, а останній байт відповідно дорівнює 254 та 253. У полі *Start IP Adress* ввести адресу, перші три байти якої збігаються з адресою мережі, а останній дорівнює 10. Значення поля *Maximum number of Users* встановити рівним 15.

5. На PC1 включити динамічний режим конфігурування адрес (DHCP). Впевнитись, що поле IP-адреси, маска підмережі, шлюз за замовчуванням та DNS-сервер порожні. Не закриваючи вікна конфігурування PC1 натиснути кнопку «CaptureForward».

6. Переглянути вміст пакета, що передав PC1, звернути увагу, що вказано в полі адреси отримувача, порівняти вміст пакетів «Inbound PDU» та «Outbound PDU». Визначити, яку інформацію передав DHCP-сервер станції.

7. Послідовно натискаючи на кнопку «CaptureForward» спостерігати за рухом пакетів та аналізувати їх вміст. Дочекатися завершення конфігурування. На робочій станції PC1 переглянути вміст полів: IP-адреса, маска підмережі, шлюз за замовчуванням та DNS- сервер.

8. Зробити висновки.

Завдання №4. Засвоєння принципів перетворення DNS-імен в IP-адреси

1. За допомогою програми Packet Tracer побудувати мережу, структуру якої наведено на рисунку 6.18. Призначити такі IP-адреси: перші три байти всіх адрес відповідають адресі мережі, останній байт PC1 – 1; DNS-сервера – 253; HTTP-сервера – 252.

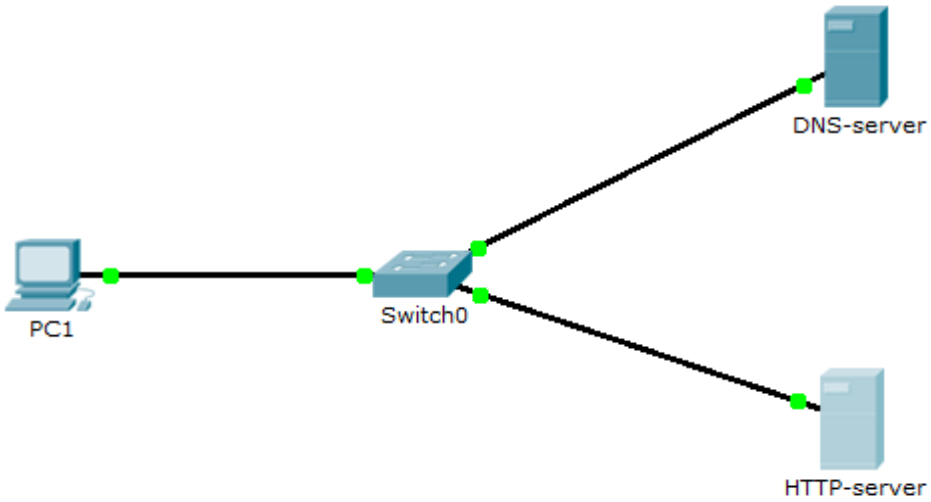


Рисунок 6.18 – Структура мережі для завдання 4.

2. На PC1 в полі DNS вказати відповідну адресу.
3. На DNS-сервері в вкладці DNS додати запис, в якому в полі *Domain Name* вказати ім’я *scs.kpi.ua*, а в полі IP-Address – адресу HTTP-сервера.
4. Перейти до режиму «Simulation».
5. На PC1 в командному рядку виконати команду **ping**, вказавши доменне ім’я *scs.kpi.ua* та натиснути кнопку «Capture/Forward».

6. Натискати кнопку «Capture/Forward», поки на PC1 не буде сформований DNS-запит. Пояснити призначення ARP-пакетів, які були сформовані на попередніх кроках.
7. Коли DNS-запит надійде на сервер, переглянути його вміст (Inbound PDU) та вміст відповіді (Outbound PDU). Пояснити інформацію, що міститься в цих пакетах.
8. Натиснути кнопку «Capture/Play» та дочекатися завершення виконання команди, пояснити причину утворення всіх пакетів.
9. Перейти в режим «Real Time».
10. У налаштуваннях DNS-сервера відключити DNS-сервіс. У полі URL веббраузера на PC1 набрати IP-адресу HTTP-сервера. Зафіксувати результат. Замість IP-адреси HTTP-сервера набрати його DNS-ім'я. Пояснити отриманий результат. Не закриваючи вікна веббраузера включити DNS-сервіс. Зафіксувати зміни, що відбулися.

Завдання №5. Кабель типу «вита пара». Використання перехресного кабелю

Витою парою (twisted pair) називається кабель, в якому ізольована пара провідників скручена з невеликим числом витків на одиницю довжини. Скручування проводів зменшує електричні перешкоди ззовні при поширенні сигналів кабелем, а екрановані виті пари ще більше збільшують завадозахищеність сигналів. Кабель типу «вита пара» (скручена пара, кручена пара) використовується в багатьох мережевих технологіях, включаючи Ethernet, ARCNet і IBM Token Ring. Кабелі на витій парі підрозділяються на: неекрановані UTP (Unshielded Twisted Pair) і екрановані мідні кабелі. Останні підрозділяються на два різновиди: з екрануванням кожної пари і загальним екраном STP (Shielded Twisted Pair) і з одним тільки загальним екраном FTP (Foiled Twisted Pair). Наявність або відсутність екрану в кабелі зовсім не означає наявності або відсутності захисту даних, які передаються в ньому, а говорить лише про різні підходи до зниження впливу завад. Відсутність екрану робить неекрановані кабелі гнучкими і стійкими до зламів. Крім того, вони не вимагають дорогого контуру заземлення для експлуатації в нормальному режимі порівняно з екранованими кабелями. Неекрановані кабелі ідеально підходять для прокладання в приміщеннях усередині офісів, а екрановані краще використовувати для установки в місцях з особливими умовами експлуатації.

Існує два стандарти мережевого кабелю – типи T568A і T568B. Вони відрізняються порядком розташування кольорових проводів в роз'ємі RJ-45 (рисунок 6.19). Схема з'єднання T568B більш поширена, хоч багато пристроїв також підтримують і схему T568A. Якщо обидва кінці кабелю з'єднані згідно з одним стандартом, то це пряме з'єднання, якщо ні, то це перехресне з'єднання. Обидва стандарти можна використовувати для побудови прямого кабеля (рисунок 6.20).

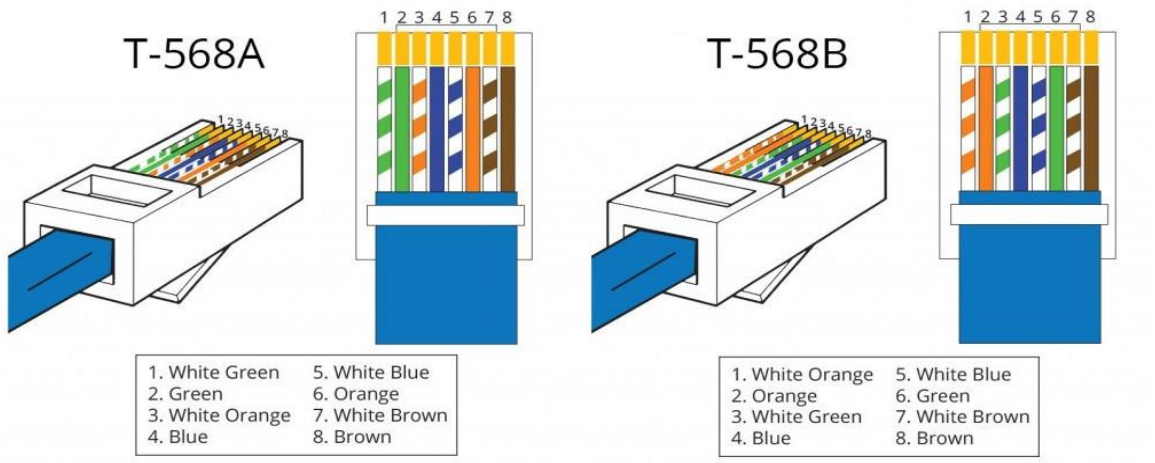


Рисунок 6.19 – Два стандарти мережевого кабеля

В схемі 568A місцями помінялися пара 1-2 (зелена) і пара 3-6 (помаранчева) зі збереженням черговості.

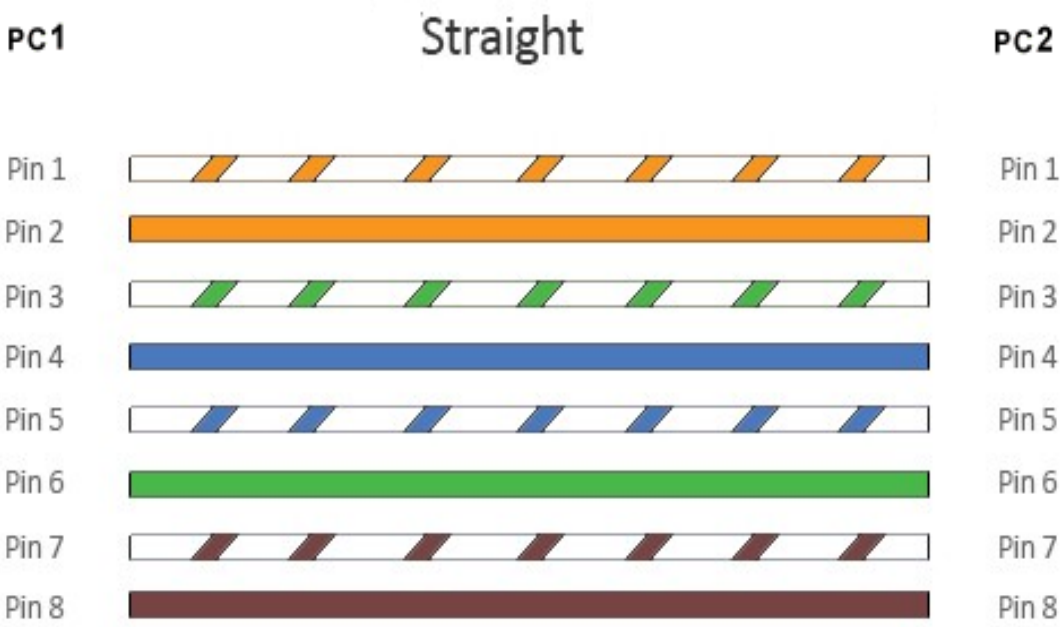


Рисунок 6.20 – Прямий кабель типу «Вита пара»

По восьми проводах мережевого кабелю типу «Вита пара» передаються сигнали «Передача» (TX) та «Прийом» (RX) (рисунок 6.21). Провідники, які під’єднані до контактів 1 і 2, 3 і 6, 4 і 5, 7 і 8, утворюють виту пару (рисунок 6.22). По парам провідників передаються сигнали одного логічного призначення, але в протилежних фазах. Це забезпечує суттєве зменшення впливів одного провідника на інший при передачі високочастотного сигналу та зменшенню завад від сторонніх джерел.

N	Назва	Опис
1	TX_D1+	Tranceive data +
2	TX_D1-	Tranceive data -
3	RX_D2+	Receive data +
4	BI_D3+	Bi-directional Data+
5	BI_D3-	Bi-directional Data-
6	RX_D2-	Receive data -
7	BI_D4+	Bi-directional Data+
8	BI_D4-	Bi-directional Data-

Рисунок 6.21 – Розподілення сигналів на контактах роз’єму RJ-45

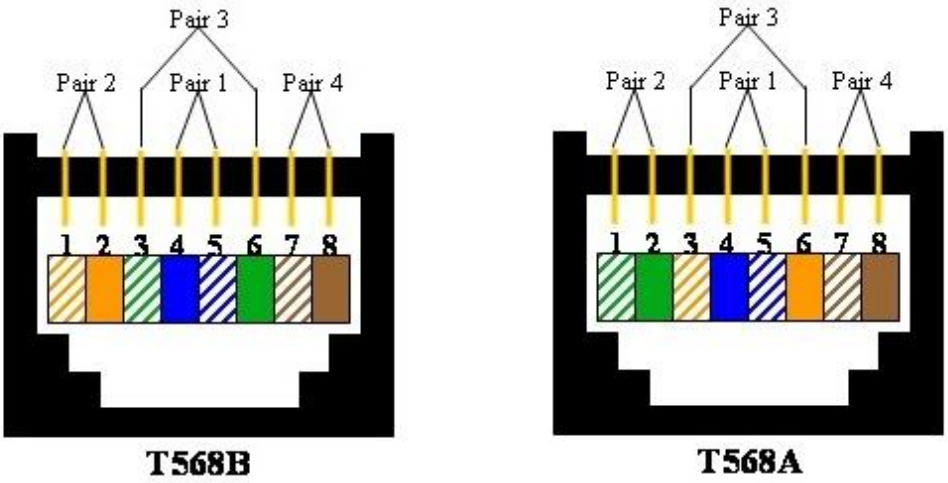


Рисунок 6.22 – Групування по парам провідників у мережевому кабелі

Необхідність застосування перехресного кабелю викликана тим, що коли, наприклад, одна робоча станція передає сигнал по провіднику, під’єднаному до контакту TX роз’єму, інша має приймати цей сигнал на контакт RX роз’єму (рисунок 6.23).

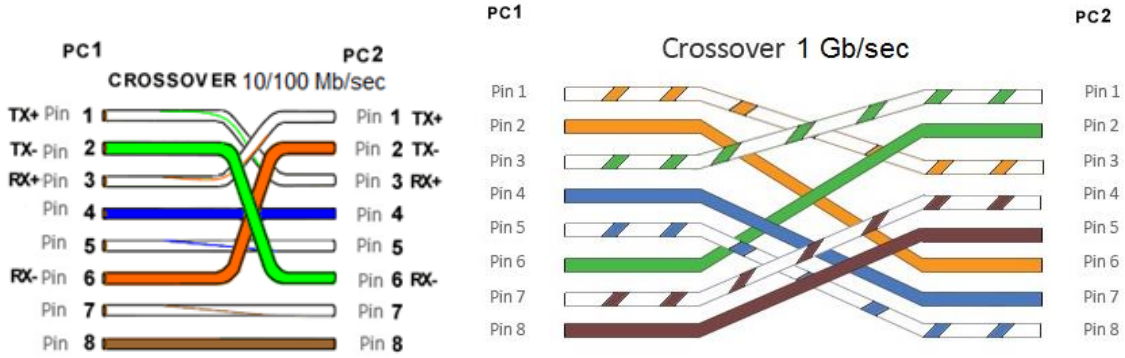


Рисунок 6.23 – Перехресний мережевий кабель

Для виготовлення кросоверного (перехресного) кабеля необхідно один його кінець обтиснути відповідно до схеми 568A, а інший – до схеми 568B.

Спочатку перехресний кабель використовувався для з'єднання однотипного обладнання: робочої станції з робочою станцією, комутатора з комутатором і т.п., тобто для пристроїв, які знаходяться на одному рівні моделі OSI. Однак останнім часом все сучасне мережеве обладнання підтримує технологію автоматичного розпізнавання і налаштування під різний тип обжимання кабелю Auto MDI/MDI-X (рисунок 6.24).

Програма Cisco Packet Tracer зобов'язує нас притримуватись традиції використовувати перехресний кабель для з'єднання пристроїв одного рівня моделі OSI.

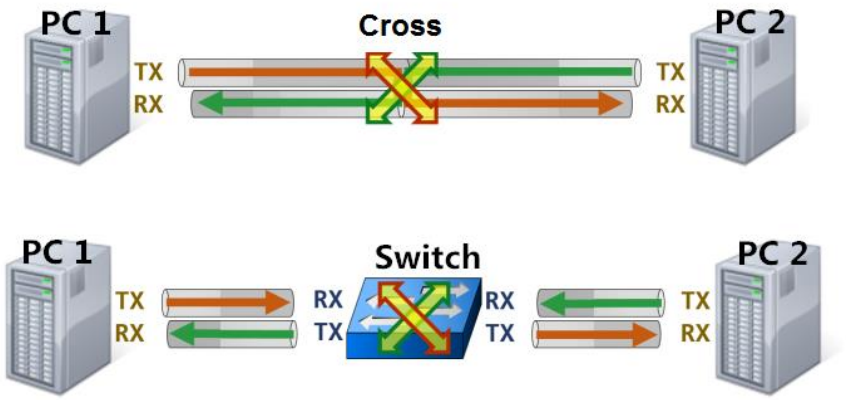


Рисунок 6.24 – Коли комп'ютер під'єднаний до комутатора перехресний кабель не потрібний

Контрольні запитання

1. Чому два комп'ютери потрібно з'єднувати перехресним кабелем?
2. Яка особливість перехресного кабеля?
3. Які поля у вікні мережевих налаштувань комп'ютера необхідно заповнити при виконанні мережевих налаштувань вручну?
4. Які чотири етапи мережевої взаємодії виконуються при централізованому управлінні мережевими налаштуваннями за допомогою протоколу DHCP?

Лабораторна робота 7
Пакетні фільтри

Мета роботи: поглиблене самостійне вивчення спеціальних питань, присвячених організації та конфігуруванню брандмауера, що використовує iptables.

План виконання лабораторної роботи

1. Ознайомитися та засвоїти теоретичні відомості викладені в навчально-методичному посібнику до лабораторної роботи.
2. Виконати завдання до лабораторної роботи. Скласти звіт.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1.Організація простого брандмауера

У комп'ютерній мережі брандмауер (firewall) – це програмно-апаратний засіб, який розміщується на межі мережі і використовується для двосторонньої передачі лише авторизованих певним чином даних. Найчастіше брандмауери захищають внутрішню корпоративну мережу від несанкціонованого проникнення із зовнішньої мережі. Однак їх можна використовувати для фільтрування вихідної інформації, обмеження доступу користувачів внутрішньої мережі назовні тощо. Брандмауери застосовують різні алгоритми фільтрування та мають різний ступінь захисту і вартість. Розрізняють такі типи брандмауерів:

- брандмауери з фільтруванням пакетів (працюють на канальному і мережевому рівнях);
- шлюзи прикладного рівня (фільтрують інформацію по додатках);
- брандмауери рівня з'єднання.

1. 2.Типи брандмауерів

Всі брандмауери можна розділити на три типи:

а) Пакетні фільтри

Пакетні фільтри проглядають IP-адреси, прапори, номери портів в заголовках пакетів, тип пакету залежно від протоколу, що використовується, адреси відправника і отримувача. На підставі цих даних вони приймають рішення про дію, яку необхідно виконати з даним пакетом. Пакети, що приходять на брандмауер, можуть бути пропущені, відкинуті або модифіковані згідно з вказаними в брандмауері правилами. Більшість брандмауерів цього типу мають початкові налаштування, які можна реконфігурувати. Подібні

брандмауери часто використовуються не тільки в серверах, але і, наприклад, в інтелектуальних мережевих комутаторах.

До позитивних характеристик пакетних фільтрів можна віднести такі: гнучкість у визначенні правил фільтрації, незначна затримка при аналізі пакетів, до недоліків – локальну мережу видно з Internet, правила фільтрації потребують хороших знань технології TCP/IP, відсутні ефективні засоби аутентифікації.

б) Сервери прикладного рівня

Сервери прикладного рівня використовують сервери конкретних сервісів: TELNET, FTP, HTTP і т. ін., які запускаються на брандмауері і пропускають через себе весь трафік, що належить до даного сервісу. Таким чином, між клієнтом і сервером утворюються два з'єднання: від клієнта до брандмауера і від брандмауера до місця призначення. Повний набір підтримуваних серверів розрізняється для кожного конкретного брандмауера. Використання серверів прикладного рівня дозволяє вирішити важливу задачу аутентифікації на рівні застосування користувача. При написанні правил доступу використовуються такі параметри як назва сервісу, ім'я користувача, допустимий часовий діапазон використання сервісу, комп'ютери, з яких можна користуватися сервісом, схеми аутентифікації. Сервери протоколів прикладного рівня дозволяють забезпечити найвищий рівень захисту: взаємодія із зовнішнім світом реалізується через невелике число прикладних програм, які повністю контролюють весь вхідний і вихідний трафік.

Проксі-сервер прикладного рівня**,** як це випливає з його назви, вміє «втручатися» в процедуру взаємодії клієнта і сервера одним з прикладних протоколів, наприклад, того ж HTTP, HTTPS, SMTP/POP, FTP або telnet. Щоб виконувати роль посередника на прикладному рівні, проксі-сервер повинен «розуміти» сенс команд, «знати» формати і послідовність повідомлень, якими обмінюються клієнт і сервер відповідної служби. Це дає можливість проксі-серверу проводити аналіз вмісту повідомлень, робити висновки про підозрілий характер того чи іншого сеансу.

До позитивних характеристик серверів прикладного рівня можна віднести такі: локальна мережа невидима з Internet, захист на прикладному рівні дозволяє виконувати додаткові перевірки вхідного трафіку, зменшуючи тим самим можливість несанкціонованого проникнення зловмисників, наявність ефективної системи аутентифікації.

в) Сервери рівня з'єднання

Сервер рівня з'єднання являє собою транслятор TCP-з'єднання. Користувач утворює з'єднання з певним портом на брандмауері, після чого останній проводить з'єднання з місцем призначення з іншої сторони від брандмауера. Під час сеансу цей транслятор копіює байти в обох напрямках. Як правило, пункт призначення задається наперед, тоді як джерел може бути багато. Використовуючи різні порти, можна створювати різні конфігурації. Такий тип серверу дозволяє створювати транслятор для будь-якого заданого

користувачем сервісу, що базується на TCP, здійснювати контроль доступу до цього сервісу, збір статистики про його використання.

Проксі-сервер рівня з'єднань виконує свою посередницьку місію на транспортному рівні, контролюючи TCP-з'єднання. Очевидно, що працюючи на більш низькому рівні, проксі-сервер має набагато менший «інтелект» і має менше можливостей для виявлення та попередження атак. Однак він має одну дуже важливу перевагу перед проксі-сервером прикладного рівня: універсальність, тобто він може бути використаний будь-якими додатками, що працюють по протоколу TCP (а в деяких випадках і UDP).

Прикладом проксі-сервера даного типу є розроблений досить давно, але все ще широко застосовуваний сервер **SOCKS** (від SOCKetS).

Для досягнення більш високого рівня безпеки функції пакетних брандмауерів і серверів прикладного рівня можуть бути об’єднані в одному міжмережевому екрані.

Брандмауери нового покоління **NGFW (**Next-Generation Firewall) на відміну від звичайних брандмауерів надають можливість переглядати вміст кожного пакету даних. Це дозволяє заборонити використання небажаних додатків або обмежити можливисті окремих додатків.

1.3. Міжмережевий екран Netfilter/iptables

Netfilter - міжмережевий **екран** (брандмауэр), вбудований в ядро Linux починаючи з версії 2.4.

iptables - назва утиліти користувача (запускається із командного рядка), для керування системою Netfilter. З її допомогою адміністратори створюють і змінюють правила, які керують фільтрацією і перенаправленням пакетів.

Розглянемо схему роботи Netfilter/iptables (рис. 7.1).

Мережеві пакети надходять на мережевий інтерфейс, налаштований на стек TCP/IP, і після деяких простих перевірок ядром (наприклад, підрахунок контрольної суми) проходять через послідовність **ланцюжків (chain)** (позначені пунктиром). Пакет обов'язково проходить первинний ланцюжок **PREROUTING**. Після ланцюжка **PREROUTING**, у відповідності з таблицею маршрутизації, аналізується адреса призначення пакету і визначається, куди він далі буде направлений (в який ланцюжок). Якщо пакет не адресовано локальній системі (в TCP-пакеті поле «адреса отримувача» - не локальна система), то він направляється в ланцюжок **FORWARD**, якщо пакет адресований локальній системі, то він спрямовується в ланцюжок **INPUT** і після проходження **INPUT** віддається локальним процесам-демонам. Після обробки локальною програмою, при необхідності, формується відповідь. Пакет, який відправляється локальною системою у відповідності з правилами маршрутизації, направляється на відповідний маршрут (хост із локальної мережі або маршрутизатор) і також направляється в ланцюжок **OUTPUT**. Після ланцюжка **OUTPUT** (або **FORWARD**, якщо пакет був адресований іншому

хосту) пакет знову звіряється з правилами маршрутизації і відправляється в ланцюжок **POSTROUTING**.

Кожний ланцюжок, який проходить пакет, складається із набору таблиць (table), позначені овалами. Таблиці в різних ланцюжках мають схожу назву, але ніяким чином між собою не пов'язані. Наприклад, таблиця **nat** в ланцюжку **PREROUTING** ніяк не зв'язана з таблицею nat в ланцюжку **POSTROUTING**. Кожна таблиця складається з впорядкованого набору правил. Кожне правило містить умову, якій повинен відповідати пакет і дії до пакету, коли пакет відповідає цій умові.

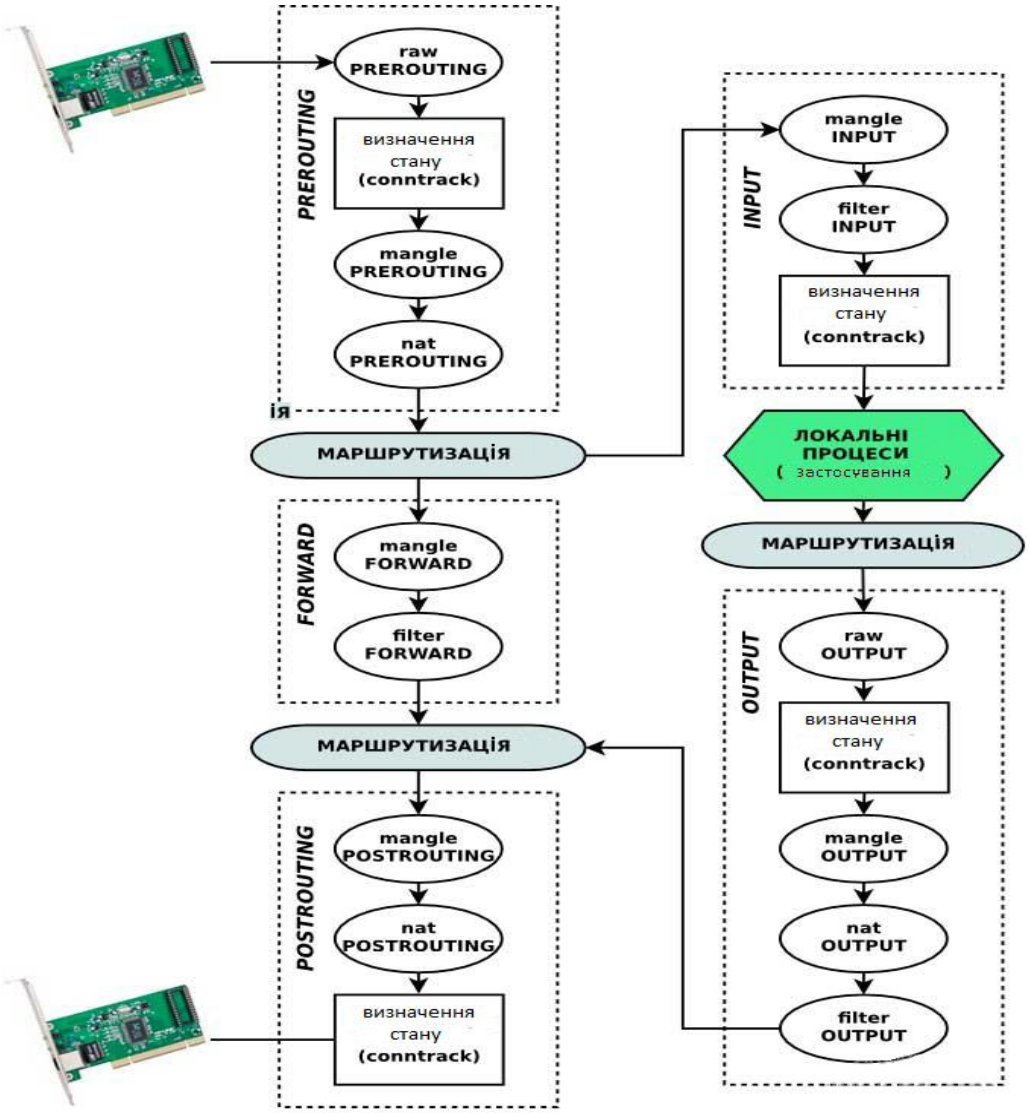


Рисунок 7.1 – Схема роботи Netfilter/iptables

Проходячи через низку ланцюжків пакет послідовно проходить кожну таблицю (у показаному на рисунку порядку) і в кожній таблиці послідовно звіряється з кожним правилом (точніше – з кожним набором умов/критеріїв в правилі), і якщо пакет відповідає якому-небудь критерію, то виконується задана дія над пакетом. При цьому в кожній таблиці (крім таблиці користувача) присутня задана за замовчуванням політика. Ця політика визначає дію над

пакетом, у випадку, якщо пакет не відповідає жодному правилу в таблиці. Частіше за все – це дія **ACCEPT**, щоб прийняти пакет і передати в наступну таблицю або **DROP** – щоб відкинути пакет. У випадку, якщо пакет не буде відкинутий, він завершує свою подорож по ядру системи і відправляється в мережевий інтерфейс, який відповідає правилам маршрутизації.

Ланцюжки Netfilter:
PREROUTING - для первинної обробки вхідних пакетів;
INPUT - для вхідних пакетів, адресованих безпосередньо локальному комп'ютеру;
FORWARD - для пакетів, адресованих іншим комп'ютерам;
OUTPUT - для пакетів, створених локальним комп'ютером;
POSTROUTING - для завершальної обробки пакетів, які відправляються.

За допомогою утиліти iptables також можна створювати і відкидати власні ланцюжки.

Ланцюжки організовані в 4 наступні таблиці.

- **raw** - пакет проходить цю таблицю до передачі системі визначення стану. Використовується рідко, наприклад, для маркування пакетів, які не опрацьовуються системою визначення стану. Для цього в правилі вказується дія NOTRACK. Присутня в ланцюжках PREROUTING і OUTPUT.
- **mangle** - містить правила модифікації IP-пакетів (зазвичай, полів заголовку). Крім того, підтримуються дії TTL, TOS, і MARK (для модифікації полів TTL і TOS, і для модифікації маркерів пакету). Рідко необхідна і може бути небезпечна. Присутня у всіх п’яти стандартних ланцюжках.
- **nat** - призначений для підміни адреси відправника або отримувача. Дану таблицю проходить тільки перший пакет з потоку, трансляція адрес або маскування (підміна адреси відправника або отримувача) застосовується до всіх наступних пакетів в потоці автоматично**.** Підтримує дії DNAT, SNAT, MASQUERADE, REDIRECT. Використовується в ланцюжках PREROUTING, OUTPUT, і POSTROUTING.
- **filter - основна таблиця**, використовується за замовчуванням, якщо назва таблиці не вказана, а також для фільтрації пакетів в ланцюжках INPUT, FORWARD, і OUTPUT.

Безпосередньо для фільтрації пакетів використовуються таблиці **filter**.

1.4. Конфігурування брандмауера

Для забезпечення фільтрації пакетів в брандмауері при використанні ядра, починаючи з версії Linux 2.4, застосовується інструментальний засіб iptables.

Фільтр пакетів – це програма, яка аналізує заголовки пакетів, які надходять, і вирішує подальшу долю всього пакету. Фільтр може відкинути пакет (DROP), прийняти (ACCEPT) пакет або виконати більш складну його обробку.

В Linux фільтр пакетів вбудований в ядро (як модуль або як невід'ємна частина ядра), якщо при конфігуруванні ядра використовувалася опція CONFIG_NETFILTER.

Утиліта **iptables** взаємодіє з ядром і вказує йому, які пакети підлягають фільтруванню. За допомогою саме цієї програми можна керувати фільтром пакетів. Утиліта iptables вставляє та видаляє правила в/із таблиці фільтру пакетів ядра. Це означає, що всі зміни в таблиці фільтру, зроблені за допомогою утиліти iptables, будуть втрачені після перезавантаження. Для збереження виконаних змін після перезавантаження використовуються утиліти **iptables-save и iptables-restore**, які зберігають і відновлюють правила фільтрування.

За допомогою команди iptables можна вводити ланцюжки правил, які дозволяють керувати проходженням пакетів до системи. Програма **iptables** замінила утиліту **ipchains**, що застосовувалася в більш ранніх версіях Linux. Проте можна продовжувати використовувати команди **ipchains** і команди **ipfwadm**, що застосовувалися раніше, завантаживши модулі ipchains.o або ipfwadm.o, передбачені в програмному забезпеченні Netfilter.

Принцип фільтрації такий: коли пакет проходить через ядро, він перевіряється на відповідність одному або кільком правилам. При цьому в залежності від цих правил він може бути пропущений (ACCEPT), відкинутий (DROP) або відхилений (REJECT). Крім цього, він може бути відправлений на перевірку в інший ланцюжок правил. Факт проходження пакету, який підпадає під дію певного правила, може бути відмічений в syslog. Правила можуть виконувати перевірку адреси/порту відправника/отримувача, протоколу, прапорців TCP тощо.

Зауважимо, REJECT відрізняється від DROP тим, що замість простого знищення пакету його відправнику надсилається повідомлення про недоступність комп'ютера отримувача. Це створює додатковий трафік, але в окремих випадках спонукає відправника припинити відправляти пакети (DROP розцінюється відправником як таймаут, тобто пакети просто відправляються, і підтвердження не приходять, а при REJECT відправник отримає повідомлення про те, що комп'ютер отримувача тимчасово недоступний або не існує).

1.5. Правила IP-таблиць

Ядро запускається з трьома списками правил в таблиці фільтру пакетів, які називаються firewall chains або просто chains (ланцюжки). Ці наперед визначені ланцюжки називаються так: INPUT, OUTPUT і FORWARD. Кожний ланцюжок представляє собою список правил, які визначають дію з пакетами, які вони аналізують. Ці дії називаються метою правила. Метою може бути і перехід на інший (визначений користувачем) ланцюжок в цій же таблиці.

Спрощена взаємодія ланцюжків наведена на рисунку 7.2.

Три овали представляють три ланцюжки. Коли пакет надходить в певний ланцюжок, пакет досліджується для того, щоб визначити, що робити з пакетом в подальшому. Якщо ланцюжок «говорить», що пакет треба відкинути (DROP),

пакет знищується, якщо ланцюжок дозволяє проходження пакету (ACCEPT), пакет проходить до наступного кроку за схемою. Ланцюжок - це набір визначених правил. Кожне правило говорить: «якщо заголовок пакету має такий вигляд, то потрібно виконати такі дії». Якщо пакет не відповідає жодному правилу, застосовується правило, передбачене за замовчуванням. Якщо не вказане інше, за замовчуванням застосовується правило «приймати всі пакети»; це правило мається на увазі у всіх прикладах даної лабораторної роботи. Для створення більш чутливих ланцюжків часто застосовується також правило «відкидати всі пакети», що не відповідають жодному правилу ланцюжка.

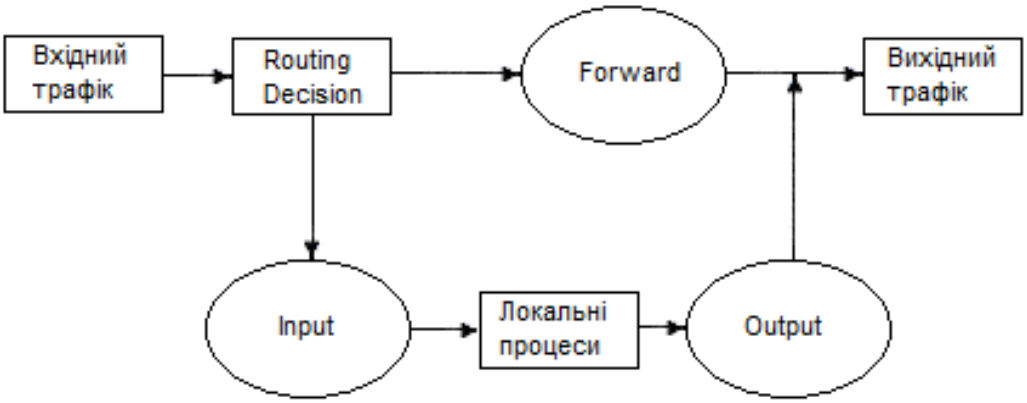


Рисунок 7.2 – Схема роботи Netfilter/iptables

Коли пакет приходить, наприклад, через Ethernet карту, ядро спочатку переглядає адресу призначення пакету: це називається «маршрутизацією».

Якщо пакет адресований цій станції, пакет проходить в ланцюжок **INPUT**. Якщо це так, то будь-який процес, що очікує на цей пакет, отримає його.

Якщо пакет адресований іншому комп’ютеру, і якщо в ядрі відключена можливість маршрутизації, або якщо ядро не знає, як цей пакет маршрутизувати, пакет відкидається. Але якщо маршрутизація дозволена, і пакет адресований іншому мережевому інтерфейсу, пакет прямує прямо в ланцюжок **FORWARD**; якщо він дозволений (ACCEPT), він буде маршрутизований.

Нарешті, програма, яка виконується на самій станції, може сама відправляти пакети. Такі пакети будуть проходити через ланцюжок **OUTPUT**: якщо ланцюжок дозволяє їх (ACCEPT), то пакет продовжує свій шлях до інтерфейсу, якому він адресований.

1.6. Використання iptables

Робота починається з трьома вбудованими ланцюжками INPUT, OUTPUT і FORWARD, які не видаляються. Дії, які можна виконувати з ланцюжками iptables, наступні.

– Створити новий ланцюжок (-N).
– Видалити пустий ланцюжок (-X).
– Змінити правило по замовчуванню для вбудованого ланцюжка (-P).
– Переглянути правила в ланцюжку (-L).
– Очистити всі правила в ланцюжку (-F).
– Обнулити лічильники пакетів і байтів у всіх правилах в ланцюжку (-Z).

До команд, які використовуються в правилах ланцюжків, відносяться наступні.

– Додати нове правило до ланцюжка (-A).
– Вставити нове правило в ланцюжок (-I).
– Замінити правило в ланцюжку (-R).
– Видалити правило в ланцюжку (-D).
– Видалити перше правило в ланцюжку, що відповідає вказаному (-D).

В командах iptables імена ланцюжків мають бути введені літерами верхнього регістру.

Відразу за командою повинна бути вказана назва ланцюжка, до якого застосовується правило: це може бути ланцюжки INPUT, OUTPUT, FORWARD або ланцюжок, визначений користувачем. Далі перераховуються різні опції, які позначають дії, що будуть виконуватись. Одні опції вказують на критерії, які аналізуються в правилі (наприклад, адреси відправника та отримувача пакету **-s, -d**, інші визначають дію, яка повинна бути виконана з пакетом (-j). Деякі основні опції перераховані нижче.

В наступному прикладі користувач додав до ланцюжка INPUT правило: приймати всі пакети з адресою джерела 192.168.1.55. Опція -s визначає адресу відправника, вказану в пакеті, а опція -j визначає мету (дію). Як правило, в простому брандмауері використовуються дії ACCEPT і DROP. Дія ACCEPT дозволяє пропустити пакет, а дія DROP вимагає його не пропускати. Згідно наведеному нижче правилу всі отримані (INPUT) пакети з адресою джерела (-s), що збігається з 192.168.1.55, будуть прийняті та пропущені через брандмауер.

```
iptables -A INPUT -s 192.168.1.55 -j ACCEPT
```

1.7. Критерії iptables

Розглянемо критерії, що дозволяють виділяти пакети. Їх можна розділити на п’ять груп:

1) загальні критерії, які можуть застосовуватися в будь-яких правилах;
2) TCP-критерії, які застосовуються тільки до TCP-пакетів;
3) UDP-критерії, які застосовуються тільки до UDP-пакетів;
4) ICMP-критерії для роботи з ICMP-пакетами;
5) спеціальні критерії, такі як state, owner, limit та ін.

1.7.1. Загальні критерії

Критерій	Призначення
-p [!] [--sport порт] протоколи: ALL	протокол - один з протоколів TCP, UDP, ICMP або всі параметр *порт* дозволяє вказати порт
-s [!] адреса	мережева адреса джерела, що перевіряється.
-d [!] адреса	мережева адреса отримувача, що перевіряється.
-i [!] ім'я	ім'я вхідного мережевого інтерфейсу (наприклад eth0)
-j дія(мета) [порт]	Дія (мета) правила (для мети REDIRECT повинен бути вказаний порт [порт])
-o [!] ім'я	ім'я вихідного мережевого інтерфейсу (наприклад eth1)
-m модуль	використовуваний модуль – наприклад state.

Опції модуля state, наприклад NEW, INVALID, RELATED і ESTABLISHED застосовуються для розпізнавання стану пакету.

Символ **!** використовується для логічної інверсії опції, перед якою він стоїть.

1.7.2. TCP-критерії

Критерій	Призначення
--sport, **--source-port** [!]	порт, з якого був відправлений пакет. Можна вказувати номер порту або назву мережевої служби.
--dport, **--destination-port** [!]	порт або діапазон портів, на який пакет був адресований.
--tcp-flags [!]	визначає маску і прапорці tcp-пакету.
--syn [!]	критерію задовольняють пакети з встановленим прапорцем SYN і скинутими прапорцями ACK и FIN
--tcp-option [!]	перевіряється TCP-параметр пакету.

1.7.3. UDP-критерії

--sport, **--source-port** !] [!]	аналогічно **–sport** в TCP-критерії
--dport, **--destination-port** [!]	аналогічно **–dport** в TCP-критерії

1.7.4. ICMP-критерії

--icmp-type [!] - тип повідомлення ICMP, визначається номером або іменем. Числові значення визначаються в RFC 792. Щоб отримати список імен ICMP-значень треба виконати команду **iptables --protocol icmp –help**

1.7.5. Явні критерії

Перед використанням деяких критеріїв програми, які реалізують їх виконання, мають бути завантажені явно, за допомогою ключа **-m** або **--match**. Так, наприклад, якщо необхідно використати критерий **state**, то треба явно вказати це в рядку правила: **-m state** перед критерієм, що використовується.

Відмінність явних і неявних критеріїв полягає тільки в тому, що перші треба підвантажувати явно, а другі підвантажуються автоматично.

Перелічимо явні критерії:

state − перевіряється стан з'єднання (state). Відомі 4 стани: INVALID, ESTABLISHED, NEW и RELATED;

limit − встановлюється граничне число пакетів в одиницю часу, яке може пропустити правило;

mac − використовується для перевірки MAC-адреси мережевого вузла, яку передав пакет;

mark − певним чином «помічає» пакети. Використовується з різною метою, наприклад, для обмеження трафіка і фільтрування;

multiport − дозволяє вказувати в тексті правила кілька портів і кілька діапазонів портів;

owner − використовується для перевірки «власника» пакетів;

tos − використовується для проведення перевірки бітів TOS (Type Of Service) − 8-ми бітового поля в заголовку IP-пакету. Модуль має бути завантажений явно, з ключем **-m tos**

ttl − time to live - числове поле в IP-заголовку. При проходженні чергового маршрутизатора, це число зменшується на 1. Якщо число стає рівним нулю, то відправнику пакету буде передане ICMP- повідомлення типу 11 з кодом 0 (TTL equals 0 during transit) або з кодом 1 (TTL equals 0 during assembly). Для використання цього критерію необхідно явно завантажити модуль ключем -m ttl

1.7.6. Дії і переходи

Дії і переходи повідомляють правилу, що необхідно виконати, якщо пакет відповідає заданому критерію. Частіше за все застосовуються дії **ACCEPT** и **DROP**.

Коротко розглянемо поняття переходів.

Запис переходів у правилах має такий же вигляд як і запис дій, тобто, ставиться ключ **-j** і вказується назва ланцюжка правил, на який виконується перехід. На перехід накладається низка обмежень, перше - ланцюжок, на який виконується перехід, повинен знаходитися в тій же таблиці, що й ланцюжок, із якого цей перехід відбувається, друге - ланцюжок, який є метою переходу, має бути створений до того, як на нього будуть виконуватись переходи.

Наприклад, створимо ланцюжок **tcp_packets** в таблиці filter за допомогою команди

Iptables -N tcp_packets

Тепер можна виконати перехід на цей ланцюжок :

iptables -A INPUT -p tcp -j tcp_packets

Це правило виконується наступним чином. Зустрівши пакет протоколу TCP, iptables виконає перехід на ланцюжок **tcp_packets** і продовжить рух пакету по цьому ланцюжку. Якщо пакет досягне кінця ланцюжка, то він буде повернутий в ланцюжок **INPUT,** і рух пакету продовжиться з правила, що слідує за правилом, яке викликало перехід. Якщо до пакету у вкладеному ланцюжку буде застосована дія **ACCEPT**, то автоматично пакет буде вважатися прийнятим і в ланцюжку, з якого був виклик, подальший рух буде припинений. Пакет може продовжувати рух іншими ланцюжками в інших таблицях.

До інших дій відносяться наведені нижче.

Дія DNAT використовується для перетворення адреси призначення в IP-заголовку пакета. Якщо пакет підпадає під критерій правила, що виконує **DNAT**, то в цьому пакеті, і у всіх наступних пакетах із цього потоку, адреси призначення будуть перетворені і передані на певні пристрої, хост або мережу.

Дія LOG - дія, яка призначена для журналювання окремих пакетів і подій. В журнал можуть заноситися заголовки IP-пакетів і інша інформація, яка потім може бути прочитана за допомогою **dmesg або syslogd** або за допомогою інших програм.

Дія MARK використовується для установки "міток" для певних пакетів. Ця дія може виконуватись тільки в межах таблиці *mangle*.

Дія MASQUERADE - ця дія в основі своїй аналогічна дії **SNAT,** тільки не має ключа **--to-source;** причиною тому те, що маскарадинг може працювати, наприклад, з dialup-підключенням або з DHCP, тобто в тих випадках, коли IP-адреса призначається динамічно. Якщо підключення динамічне, то потрібно використовувати маскарадинг, якщо ж підключення статичне, то краще використовувати дію **SNAT**.

Дія MIRROR В результаті дії **MIRROR** в пакеті поля **source** і **destination** міняються місцями (invert the source and destination fields) і

	пакет відправляється в мережу. Застосовується тільки для експериментів і демонстрацій.
Дія QUEUE	ставить пакет в чергу на обробку процесом користувача. Дія може бути використана для потреб обліку, проксування або додаткової фільтрації пакетів.
Дія REDIRECT	виконує перенаправлення пакетів і потоків на інший порт тієї ж самої станції. **REDIRECT** може використовуватися тільки в ланцюжках PREROUTING та OUTPUT таблиці **nat.**
Дія REJECT	використовується, як правило, в тих же випадках, що і **DROP**, але на відміну від **DROP**, **REJECT** повідомляє хост, який передав пакет, про помилку.
Дія RETURN	зупиняє рух пакету по поточному ланцюжку правил і виконує повернення в ланцюжок, з якого був перехід в поточний ланцюжок, якщо поточний ланцюжок був вкладений, або, якщо поточний ланцюжок лежить на самому верхньому рівні (наприклад, INPUT), то до пакету буде застосована політика за замовчуванням.
Дія SNAT	використовується для перетворення мережевих адрес (Source Network Address Translation), тобто заміну IP-адреси відправника в заголовку IP-пакету.
Дія TOS	використовується для встановлення бітів в полі Type of Service IP-заголовку. Поле TOS містить 8 біт, які використовуються для маршрутизації пакетів.
Дія TTL	використовується для модифікації поля Time To Live в заголовку IP-пакету.
Дія ULOG	надає можливість журналювання пакетів в просторі користувача. Ця дія замінює традиційну дію **LOG**, що базується на системному журналі

1.8. Сценарії IP-таблиць

Команди iptables можна вводити в командному рядку командного інтерпретатора. Проте після вимкнення системи ці команди будуть втрачені. Для збереження введених команд їх треба помістити в сценарій. Можна створити сценарій /etc/iptables.rules і помістити в нього команди iptables. З часом можна буде застосовувати сценарій iptables-save для виводу правил на стандартний пристрій виводу.

Тепер вже достатньо інформації для створення простого сценарію IP-таблиць, що дозволяє встановити основні засоби захисту окремого комп'ютера, підключеного до Internet. В наступному сценарії використовується простий процес фільтрації за допомогою IP-таблиць для захисту окремого комп'ютера від спроб зламу ззовні. Спочатку встановлюється ціль ACCEPT для вхідних правил IP-таблиць. Це означає, що якщо пакет не відповідає жодному критерію правил,

згідно якому він повинен бути знищений, пакет буде пропущений. Потім відкидаються спроби зламу за методом імітації IP-адреси, а також будь-які спроби ініціювати з'єднання ззовні (пакети SYN). До того ж спроби створити з'єднання ззовні реєструються в журналі.

```
# В комп'ютері з адресою 192.168.1.1 застосовується
# пристрій Ethernet eth0
# Відключити маршрутизацію IP-пакетів
echo 0 > /proc/sys/net/ipv4/ip_forward
# Видалити правила ланцюжка
iptables -F INPUT
# Встановити використовувану за замовчуванням ціль ланцюжка
INPUT iptables -P INPUT ACCEPT
# Імітація локальної IP-адреси; відкинути всі
# зовнішні пакети з внутрішньою адресою
iptables -A INPUT -j DROP -i eth0 –s 192.168.1.1
# Імітація локальної IP-адреси; відкинути всі
# зовнішні пакети з адресою локального хоста
iptables -A INPUT -j DROP -i eth0 –s 127.0.0.0/255.0.0.0
iptables -A INPUT -m state -state NEW -j DROP -i eth0 -p tcp
# Заборонити ініціалізацію з'єднань ззовні і
# дозволити використання раніше встановлених
# з'єднань
iptables -A INPUT -m state -state NEW –i eth0 -j DROP
# Включити перенаправлення IP
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Спочатку в цьому сценарії необхідно очистити поточні IP-таблиці за допомогою команди очищення (опція -F). На той час, поки відбувається ввод в комп'ютер правил і таблиці, повинна бути також відключена маршрутизація IP-пакетів

```
echo 0>/proc/sys/net/ipv4/ip_forward
```

Всі існуючі правила ланцюжків INPUT відміняються, а потім за замовчуванням встановлюється дія ACCEPT. Для установки цілі, яка використовується в таблиці за замовчуванням, служить опція -P. Ціль, вказана за замовчуванням, виконується, якщо пакет не відповідає ніякому іншому правилу. В даному прикладі як загальна ціль для таблиць INPUT застосовується ACCEPT. Будь-які пакети, що не відповідають ніякому іншому правилу таблиці INPUT, будуть прийняті і пропущені в систему.

```
# Видалити правила ланцюжка
```

```
iptables -F INPUT
# Встановити використовувану за замовчуванням мету ланцюжка
iptables -P INPUT ACCEPT
```

Одним зі способів захисту системи від імітації локальних IP-адрес системи в будь-якому пакеті є перевірка всіх адрес джерела на предмет того, чи не вказана в них IP-адреса поточного власного комп'ютера. В даному прикладі буде відкинутий будь-який пакет, джерелом адреси якого є адреса власної системи. Така ж стратегія може застосовуватися і до локального хоста. Будь-який пакет, що надходить ззовні по інтерфейсу eth0, в якому як джерело адреси вказаний локальний хост, буде знехтуваний, тому що це пакет, відправлений сторонньою особою, який маскується під власний пакет локального хоста.

```
# Імітація локальних IP-адрес; відкинути всі зовнішні пакети з
# внутрішньою адресою
iptables -A INPUT -j DROP -i eth0 -s 192.168.1.1
# Імітація локальних IP-адрес; відкинути всі зовнішні пакети
# з адресою локального хоста
iptables -A INPUT -j DROP -i eth0 -s 127.0.0.0/255.0.0.0
```

Щоб унеможливити для сторонніх осіб ініціювати доступ до системи, можна створити правило, що виключає надходження пакетів SYN (для нових з'єднань) ззовні з використанням опції -state з ключовим словом NEW.

Наступна команда дозволяє відкинути всі спроби встановлення нових з'єднань через інтерфейс eth0 (припускається, що комп'ютер підключений до Internet або локальної мережі тільки через інтерфейс eth0).

```
iptables -A INPUT -m state —state NEW –j DROP -i eth0 -p tcp
```

Після встановлення всіх правил знову включається маршрутизація IP-пакетів:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Програма iptables надає можливість розширення її функцій, і в IP-таблиці можна включити ряд опцій з додатковими критеріями відбору.

Однією з найзручніших є функція розширення state, яка дозволяє легко знаходити в пакеті інформацію про пройдений маршрут. Для її використання потрібно спочатку вказати модуль state за допомогою опції -m state. Потім можна застосувати опцію **-state**. Після цього можна вказувати в таблицях будь-який з наступних критеріїв.

Опція	Призначення
NEW	Пакет, який створює нове з'єднання.

ESTABLISHED	Пакет, який належить до існуючого з'єднання.
RELATED	Пакет, який пов'язаний з існуючим з'єднанням, але не належить до нього. Такими є повідомлення про помилки ICMP або пакети протоколу встановлення з'єднання FTP.

В наступному прикладі показано, як можна знищити будь-які пакети, які є спробою створити нове з'єднання через інтерфейс eth0.

iptables –A INPUT –m state –state NEW –i eth0 –j DROP

Для отримання списку правил, що зберігаються в IP-таблицях, може застосовуватися опція **-L**. Після додавання опції –v(verbose) з'явиться докладний лістинг. В наступному прикладі показаний перелік правил, встановлених після виконання приведеного вище сценарію myfilter.

iptables -L
Chain INPUT (policy ACCEPT 2 packets, 206 bytes)
target prot opt in source destination
DROP all – eth0 any turtle.mytrek.com anywhere
DROP all — eth0 any 127.0.0.0/8 anywhere
DROP tcp – eth0 any anywhere anywhere state NEW
Chain FORWARD (policy ACCEPT 773 packets, 318322 bytes) target prot opt in source destination
Chain OUTPUT (policy ACCEPT 88 packets, 11508 bytes) target prot opt in source destination

1.9. Приклади використання iptables

Правила IP-таблиць зазвичай застосовуються до конкретного мережевого інтерфейсу, такого як інтерфейс Ethernet, який використовується для підключення до Internet. У відокремленій системі, що підключена до Internet, зазвичай застосовуються два інтерфейси: з'єднання Internet з інтерфейсом локального хоста і внутрішні з'єднання між процесами на комп'ютері. Посилання на інтерфейс для Internet виконується з використанням імені пристрою для цього інтерфейсу. Наприклад, на карту Ethernet з ім'ям пристрою /dev/eth0 можна посилатися, використовуючи ім'я eth0. Модем, що працює по протоколу PPP, з ім'ям пристрою /dev/ppp0, матиме ім'я ppp0. В правилах IP-таблиць для позначення вхідного інтерфейсу застосовується опція **-i**; цю опцію можна використовувати тільки у вхідних і перенаправляючих ланцюжках. Опція **-o** вказує на вихідний інтерфейс і може застосовуватися тільки для вихідних і перенаправляючих ланцюжків. Тому правила можуть застосовуватися до пакетів, що входять в конкретні мережеві пристрої і виходять з них.

1. В наступному прикладі перше правило містить посилання на пристрій Ethernet (eth0), а друге - на локальний хост.

```
# iptables -A INPUT -j DROP -i eth0 –s 192.168.44.87
# iptables -A INPUT -j ACCEPT -i lo
```

2. Показати статус iptables

```
# iptables -L -n -v
```

Тут:
-L : вивести список правил.
-v : вивести додаткову інформацію. Ця опція показує ім'я інтерфейсу, опції, TOS маски.
-n : вивести IP- адресу і порт числами (не використовуючи DNS- сервери для перетворення імен. Це пришвидшує відтворення).

3. Вивести список правил з номерами рядків

```
# iptables -n -L -v --line-numbers
```

4. Показати INPUT або OUTPUT ланцюжки правил.

```
# iptables -L INPUT -n -v
# iptables -L OUTPUT -n -v --line-numbers
```

5. Видалити правила брандмауера.
Щоб показати номери рядків з існуючими правилами:

```
# iptables -L INPUT -n --line-numbers
# iptables -L OUTPUT -n --line-numbers
# iptables -L OUTPUT -n --line-numbers | less
# iptables -L OUTPUT -n --line-numbers | grep 202.54.1.1
```

Отримаємо список пронумерованих правил . Наприклад, щоб видалити правило в рядку 3, потрібно виконати таку команду:

```
# iptables -D INPUT 3
```

Видалимо правило в таблиці INPUT, яке містить IP- адресу джерела (202.54.1.1) :

```
# iptables -D INPUT -s 202.54.1.1 -j DROP
```

6. Додати правило в брандмауер.
Щоб додати одне або кілька правил в ланцюжок, спочатку отримаємо список правил з номерами рядків:

```
# iptables -L INPUT -n --line-numbers
```

Наприклад, такий:

Chain INPUT (policy DROP)
num target prot opt source destination
1 DROP all -- 202.54.1.1 0.0.0.0/0
2 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state NEW,ESTABLISHED

Щоб вставити правило між 1 і 2 рядком, виконаємо команду:

iptables -I INPUT 2 -s 202.54.1.2 -j DROP

Перевіримо , чи з'явилося правило в ланцюжку:

iptables -L INPUT -n --line-numbers

Отримаємо:
Chain INPUT (policy DROP)
num target prot opt source destination
1 DROP all -- 202.54.1.1 0.0.0.0/0
2 DROP all -- 202.54.1.2 0.0.0.0/0
3 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state NEW,ESTABLISHED

6. Зберігаємо правила брандмауера.
Через iptables-save:

iptables-save > /etc/iptables.rules

7. Відновлюємо правила брандмауера.
Через iptables-restore:

iptables-restore < /etc/iptables.rules

8. Встановлюємо політику за замовчуванням
Щоб заблокувати увесь трафік:

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
iptables -L -v –n

Після виконання цих команд жодний пакет не залишить даний хост.

9. Блокувати тільки вхідні з'єднання.

Щоб заблокувати всі не ініційовані модулем вхідні пакети, але дозволити вихідний трафік:

```
# iptables -P INPUT DROP
# iptables -P FORWARD DROP
# iptables -P OUTPUT ACCEPT
# iptables -A INPUT -m state --state NEW,ESTABLISHED -j ACCEPT
# iptables -L -v –n
```

Вихідні пакети і ті пакети, які були збережені в рамках вже встановлених сесій дозволені.

10. Відкидати адреси ізольованих мереж в публічній мережі:

```
# iptables -A INPUT -i eth1 -s 192.168.0.0/24 -j DROP
# iptables -A INPUT -i eth1 -s 10.0.0.0/8 -j DROP
```

Список IP-адрес для ізольованих мереж:

10.0.0.0/8 (A)
172.16.0.0/12 (B)
192.168.0.0/16 (C)
224.0.0.0/4 (MULTICAST D)
240.0.0.0/5 (E)
127.0.0.0/8 (LOOPBACK)

11. Блокування окремої IP- адреси.
Щоб заблокувати IP-адресу 1.2.3.4:

```
# iptables -A INPUT -s 1.2.3.4 -j DROP
# iptables -A INPUT -s 192.168.0.0/24 -j DROP
```

12. Заблокувати вхідні запити порту.
Щоб заблокувати всі вхідні запити порту 80:

```
# iptables -A INPUT -p tcp --dport 80 -j DROP
# iptables -A INPUT -i eth1 -p tcp --dport 80 -j DROP
```

Щоб заблокувати запит порту 80 з адреси 1.2.3.4:

```
# iptables -A INPUT -p tcp -s 1.2.3.4 --dport 80 -j DROP
# iptables -A INPUT -i eth1 -p tcp -s 192.168.1.0/24 --dport 80 -j DROP
```

13. Заблокувати запити на вихідну IP- адресу.

Щоб заблокувати конкретний домен, визначимо його адресу:

host -t a facebook.com

Відповідь: facebook.com has address 69.171.228.40
Знайдемо CIDR для 69.171.228.40:

whois 69.171.228.40 | grep CIDR

Відповідь: CIDR: 69.171.224.0/19
Заблокуємо доступ на 69.171.224.0/19:

iptables -A OUTPUT -p tcp -d 69.171.224.0/19 -j DROP

Також можна використовувати ім'я домену для блокування:

iptables -A OUTPUT -p tcp -d www.facebook.com -j DROP
iptables -A OUTPUT -p tcp -d facebook.com -j DROP

14. Записати подію і заблокувати.
Щоб записати в журнал рух пакетів перед блокуванням, додамо нове правило:
iptables -A INPUT -i eth1 -s 10.0.0.0/8 -j LOG --log-prefix "IP_SPOOF A: "
iptables -A INPUT -i eth1 -s 10.0.0.0/8 -j DROP

Перевіримо журнал (по замовчуванню /var/log/messages):

tail -f /var/log/messages
grep -i --color 'IP SPOOF' /var/log/messages

15. Заблокувати або дозволити трафік з певних MAC- адрес:

iptables -A INPUT -m mac --mac-source 00:0F:EA:91:04:08 -j DROP
*дозволити тільки для TCP port # 8080 з mac-адреси 00:0F:EA:91:04:07 *
iptables -A INPUT -p tcp --destination-port 22 -m mac --mac-source
00:0F:EA:91:04:07 -j ACCEPT

16. Дозволити або заборонити ICMP Ping запити.
Щоб заборонити:

iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
iptables -A INPUT -i eth1 -p icmp --icmp-type echo-request -j DROP

Дозволити для певних мереж / хостів:

iptables -A INPUT -s 192.168.1.0/24 -p icmp --icmp-type echo-request -j ACCEPT

Дозволити тільки частину ICMP-запитів:

** припускається, що политики по замовчуванню для вхідних пакетів встановлені в DROP **

iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A INPUT -p icmp --icmp-type destination-unreachable -j ACCEPT
iptables -A INPUT -p icmp --icmp-type time-exceeded -j ACCEPT

** дозволимо відповідати на запит **

iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

17. Відкрити діапазон портів.
iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 7000:7010 -j ACCEPT

18. Відкрити діапазон адрес:

дозволити підключення до порту 80 (Apache), якщо адреса в діапазоні від 192.168.1.100 до 192.168.1.200
iptables -A INPUT -p tcp --destination-port 80 -m iprange --src-range 192.168.1.100-192.168.1.200 -j ACCEPT
приклад для nat
iptables -t nat -A POSTROUTING -j SNAT --to-source 192.168.1.20-192.168.1.25

19. Закрити або відкрити стандартні порти.
Замінити ACCEPT на DROP, щоб заблокувати порт:

ssh tcp port 22
iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 22 -j ACCEPT
cups (printing service) udp/tcp port 631 для локальної мережі
iptables -A INPUT -s 192.168.1.0/24 -p udp -m udp --dport 631 -j ACCEPT
iptables -A INPUT -s 192.168.1.0/24 -p tcp -m tcp --dport 631 -j ACCEPT
time sync via NTP для локальної мережі (udp port 123)
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p udp --dport 123 -j ACCEPT
tcp port 25 (smtp)
iptables -A INPUT -m state --state NEW -p tcp --dport 25 -j ACCEPT

```
# dns server ports ##
iptables -A INPUT -m state --state NEW -p udp --dport 53 -j ACCEPT
iptables -A INPUT -m state --state NEW -p tcp --dport 53 -j ACCEPT
## http/https www server port ##
iptables -A INPUT -m state --state NEW -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -m state --state NEW -p tcp --dport 443 -j ACCEPT
## tcp port 110 (pop3) ##
iptables -A INPUT -m state --state NEW -p tcp --dport 110 -j ACCEPT
## tcp port 143 (imap) ##
iptables -A INPUT -m state --state NEW -p tcp --dport 143 -j ACCEPT
## Samba file server для локальної мережі ##
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 137 -j
ACCEPT
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 138 -j
ACCEPT
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 139 -j
ACCEPT
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 445 -j
ACCEPT
## proxy server для локальної мережі ##
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 3128 -j
ACCEPT
## mysql server для локальної мережі ##
iptables -I INPUT -p tcp --dport 3306 -j ACCEPT
```

20. Обмежити кількість паралельних з'єднань з сервером для однієї адреси. Для обмеження використовується **connlimit** модуль. Щоб дозволити тільки 3 ssh з'єднання для одного клієнта:

```
# iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 3 -j
REJECT
```

Встановити кількість запитів HTTP до 20:

```
# iptables -p tcp --syn --dport 80 -m connlimit --connlimit-above 20 --connlimit-mask
24 -j DROP
```

1.Дозволити приймати пакети лише з вибраного хосту, всім іншим заборонити.

```
# iptables -A INPUT -s scs.kiev.ua -j ACCEPT
# iptables -A INPUT -j DROP
```

2. Блокувати всі не ініційовані нами вхідні пакети, але дозволити вихідний трафік:

```
# iptables -P INPUT DROP
# iptables -P FORWARD DROP
# iptables -P OUTPUT ACCEPT
# iptables -A INPUT -m state --state NEW,ESTABLISHED -j ACCEPT
```

3. Заборонити ICMP виклики на нашій машині:

```
# iptables -A OUTPUT -p icmp -j DROP
```

4. Дозволити або заборонити ICMP Ping запити:
Щоб заборонити:

```
# iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
# iptables -A INPUT -i eth1 -p icmp --icmp-type echo-request -j DROP
```

Дозволити для певних мереж / хостів:

```
# iptables -A INPUT -s 192.168.1.0/24 -p icmp --icmp-type echo-request -j ACCEPT
```

Дозволити певні ICMP запити:

```
### ** припускається, що політики за замовчуванням для вхідних пакетів встановлені в DROP ** ###
# iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
# iptables -A INPUT -p icmp --icmp-type destination-unreachable -j ACCEPT
# iptables -A INPUT -p icmp --icmp-type time-exceeded -j ACCEPT
## ** дозволяємо відповідати на запит ** ##
# iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
```

Ланцюжок icmp дозволяє коректні icmp пакети, зокрема тільки ping.

```
echo "in_icmp"
$IPTABLES -A in_icmp -p icmp -m state --state NEW --icmp-type echo-request -j ACCEPT
$IPTABLES -A in_icmp -p icmp -m state --state NEW --icmp-type time-exceeded -j ACCEPT
$IPTABLES -A in_icmp -p icmp -m state --state NEW --icmp-type destination-unreachable -j ACCEPT
$IPTABLES -A in_icmp -p icmp -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A in_icmp -p icmp -s 0/0 --icmp-type 11 -j ACCEPT
```

5.Закрити всі порти, крім 25 (smtp), 80 (http), 443 (https-http з шифруванням)
Записуємо перелік правил на вхідні підключення, які дозволяють потрібні порти, а останнім правилом забороняємо вхідний потік.
Приблизно так:

```
iptables -A INPUT -p TCP --dport 80 -j ACCEPT
iptables -A INPUT -p TCP --dport 443 -j ACCEPT
iptables -A INPUT -p TCP --dport 25 -j ACCEPT
iptables -A INPUT -j DROP
```

6. Закрити або відкрити стандартні порти.
Замінити ACCEPT на DROP, щоб заблокувати порт.

```
## ssh tcp port 22 ##
iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 22 -j
ACCEPT
## cups (printing service) udp/tcp port 631 для локальної мережі ##
iptables -A INPUT -s 192.168.1.0/24 -p udp -m udp --dport 631 -j ACCEPT
iptables -A INPUT -s 192.168.1.0/24 -p tcp -m tcp --dport 631 -j ACCEPT
## time sync via NTP для локальної мережі (udp port 123) ##
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p udp --dport 123 -j
ACCEPT
## tcp port 25 (smtp) ##
iptables -A INPUT -m state --state NEW -p tcp --dport 25 -j ACCEPT
# dns server ports ##
iptables -A INPUT -m state --state NEW -p udp --dport 53 -j ACCEPT
iptables -A INPUT -m state --state NEW -p tcp --dport 53 -j ACCEPT
## http/https www server port ##
iptables -A INPUT -m state --state NEW -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -m state --state NEW -p tcp --dport 443 -j ACCEPT
## tcp port 110 (pop3) ##
iptables -A INPUT -m state --state NEW -p tcp --dport 110 -j ACCEPT
## tcp port 143 (imap) ##
iptables -A INPUT -m state --state NEW -p tcp --dport 143 -j ACCEPT
## Samba file server для локальної мережі ##
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 137 -j
ACCEPT
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 138 -j
ACCEPT
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 139 -j
ACCEPT
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 445 -j
ACCEPT
## proxy server для локальної мережі ##
iptables -A INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 3128 -j
ACCEPT
## mysql server для локальної мережі ##
iptables -I INPUT -p tcp --dport 3306 -j ACCEPT
```

7. Заблокуати доступ на 69.171.224.0/19:

iptables -A OUTPUT -p tcp -d 69.171.224.0/19 -j DROP

8. Всі пакети всіх(ALL) протоколів (udp,tcp,icmp), які надходять на вхідний інтерфейс -i $LAN_IFACE і надійшли від джерела -s $LAN_NET нашого хосту – не приймаються:

#IPTABLES -A INPUT -p ALL -i $LAN_IFACE -s $LAN_NET -j DROP

Завдання

1. Дозволити приймати пакети лише з хоста scs.kpi.ua. Виконайте перевірку дії встановлених правил.
2. Заборонити з'єднання з вашою машиною з комп'ютерів в локальній мережі.
3. Відключити відклик на ICMP запити на вашій машині. Виконайте перевірку дії встановлених правил.
4. Закрити всі порти, крім SMTP. Виконайте перевірку дії встановлених правил.
5. Створити ланцюжок користувача *spammsg*. В цьому ланцюжку записати правило, за яким усі повідомлення від джерела спаму будуть знищуватися. Виконати перехід з ланцюжка INPUT на ланцюжок spammsg.
6. Показати викладачу.

Контрольні запитання

1. Брандмацери: визначення та призначення.
2. Типи брандмауерів. Особливості використання.
3. Приклади міжмережевих екранів та їх функціонування.
4. Конфігурування брандмауера.
5. Призначення IP-таблиць.
6. Призначення iptables. Критерії iptables.
7. Сценарії IP-таблиць.

Лабораторна робота 8
Електронна пошта

Мета роботи: засвоїти основні принципи функціонування електронної пошти, ознайомитися зі структурою сервера електронної пошти та вивчити роботу прикладних протоколів SMTP, POP та IMAP.

План виконання лабораторної роботи

1. Ознайомлення та засвоєння теоретичних відомостей про структуру сервера електронної пошти, взаємодію його складових частин.
2. Ознайомлення із стандартними засобами захисту від спаму.
3. Ознайомлення із основними командами протоколів SMTP і POP3.
4. Виконання завдань до лабораторної роботи.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Організація служби електронної пошти в Інтернеті

Електронна пошта – служба передачі повідомлень між зареєстрованими адресатами. Її робота регламентована RFC-822 Standart of the format of ARPA Internet text messages (Стандарт форматування текстових повідомлень в мережі ARPA Internet) та доповненнями RFC-1123, RFC-1138, RFC-1148, RFC-1327 і RFC-2156.

Основними складовими об'єктами електронної пошти є спеціальні комп'ютери, які називаються поштовими серверами, які відправляють і отримують електронні повідомлення.

Сервер, який отримує повідомлення, працює відповідно до протоколу POP (Post Ofice Protocol). Сервер, який відправляє повідомлення працює відповідно до протоколу SMTP (Simple Mail Transfer Protocol).

Поштовий сервер складається з програмних компонентів, кожен з яких виконує певну функцію. Кожен компонент взаємодіє з іншими компонентами, забезпечуючи повну функціональність поштового сервера.

Поштові сервери взаємодіють за допомогою поштових протоколів, що забезпечують пересилання і розпізнавання переданої мережею інформації. Комп'ютери-клієнти поштових серверів обслуговують користувачів електронної пошти.

Система електронної пошти базується на 3-х типах програм:

- транспортні агенти MTA (Mail Transport Agent);
- агенти доставки MDA (Mail Delivery Agent);
- агенти користувача MUA (Mail User Agent).

MUA - це поштова програма типу mail або elm під Linux або MS Outlook для Windows. За її допомогою користувач створює електронне повідомлення та відправляє його.

MTA - це поштовий сервер, що використовує простий протокол передачі пошти SMTP (Simple Mail Transfer Protocol) для передачі повідомлень від одного поштового сервера до іншого (до речі, сервери, частіше за все, знаходяться в різних мережах (або підмережах)). Для системи Linux часто роль MTA виконує система Sendmail, яка розроблена Еріком Олменом (Eric Allman) в університеті Берклі, або Exim - більш нова програма, але менш функціональна ніж Sendmail.

MDA розподіляє отримані повідомлення між користувачами (це можуть бути як програми, так і користувачі, як локальні, так і віддалені). Більшість Linux-систем використовують як MDA програму **procmail**.

Агент доставки виконує доставку повідомлення адресату. Існує кілька стандартних типів агентів доставки:

- **local** - направляє лист на поштову скриньку, яка розташована на тому ж комп'ютері; доставка виконується, наприклад, додаванням повідомлення до певного файлу (в Unix це файл /var/mail/*поштова_скринька*);
- **SMTP** - направляє лист на поштову скриньку в іншому поштовому домені; доставка виконується шляхом з'єднання з транспортним агентом на віддаленому сервері за допомогою протоколу SMTP;
- **prog** - поштове повідомлення направляється на опрацювання певною програмою; доставка виконується викликом цієї програми, на вхід якої подається прийняте повідомлення.

Взагалі, методи доставки (і, відповідно, агенти) можуть бути різними: наприклад, збереження листа в базі даних; відправлення листа по факсу і т.п. Вибір агента доставки для кожного конкретного повідомлення здійснюється транспортним агентом згідно з заданою конфігурацією транспортного агента і адресою призначення.

Для отримання пошти клієнт встановлює з'єднання з MDA за протоколом POP3 (Post Office Protocol) або IMAP (Internet Message Access Protocol), обов'язково передаючи дані для авторизації. MDA перевіряє наявність користувача в списках і, при успішній перевірці, передає клієнту всі нові повідомлення, які знаходяться в його поштовій скриньці.

Основна різниця між ними полягає в тому, що IMAP дозволяє MUA працювати з поштовою скринькою на поштовому сервері (тобто всі листи знаходяться на сервері), що корисно, якщо користувач не має постійно закріпленого за собою комп'ютера.

Адреса електронної пошти має такий вигляд:

поштова_скринька @поштовий домен,

де поштова_скринька - ім'я (ідентифікатор) отримувача листа, а поштовий_домен - ім'я домену, в якому знаходиться поштова скринька. Наприклад:

somebody@mail.ru, one@scs.ntu-kpi.kiev.ua

Розглянемо дещо спрощену структуру поштового сервера, а також дії, які виконуються, коли відправник намагається отримати або відправити поштове повідомлення.

Розглянемо вхідне повідомлення (червоні стрілки) від somebody@mail.ru до one@scs.ntu-kpi.kiev.ua (рис. 8.1).

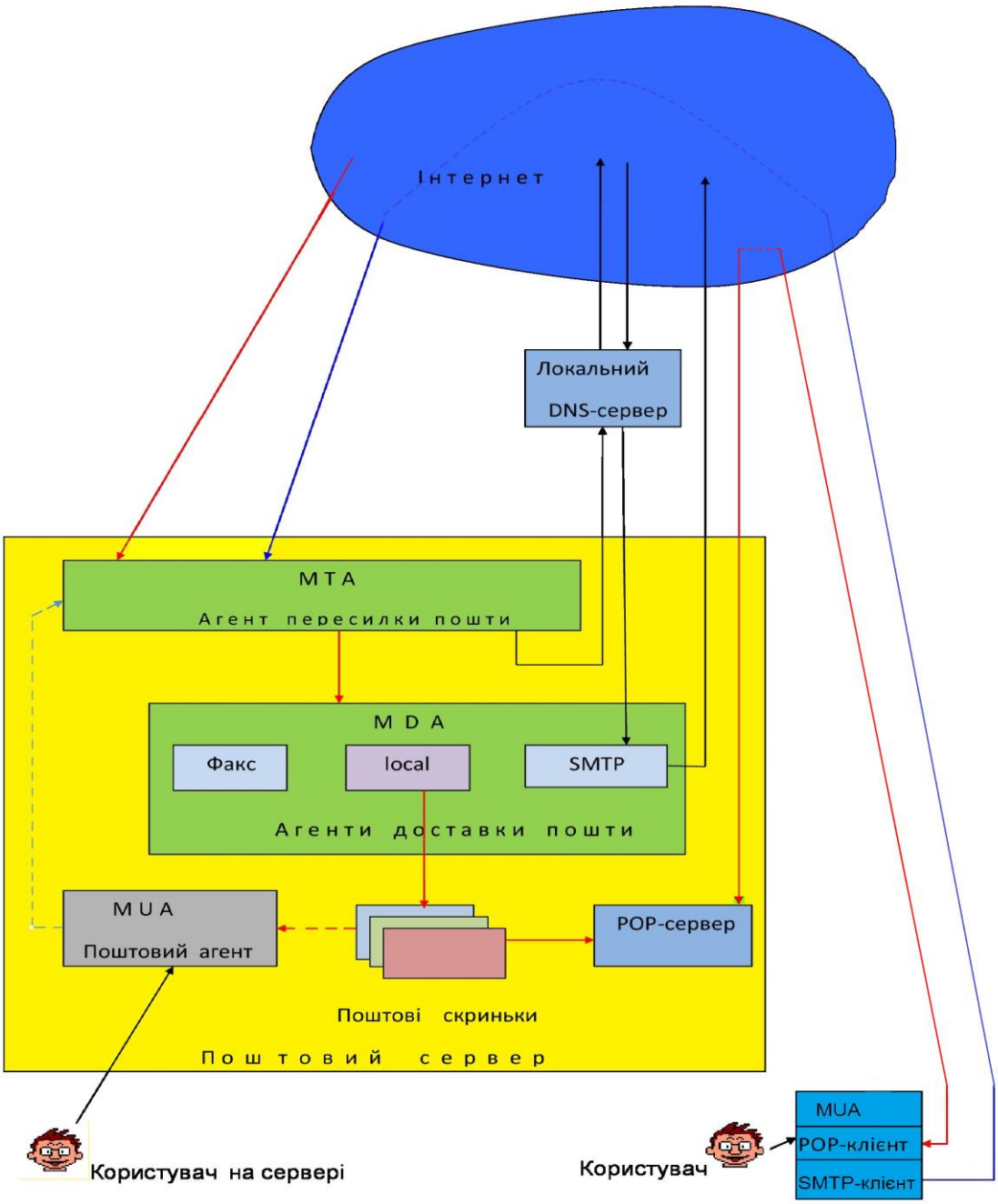


Рисунок 8.1 – Структурна схема сервера електронної пошти

Повідомлення надходить з мережі до транспортного агента MTA (для передачі повідомлень транспортному агенту через мережу використовується

протокол SMTP, який буде розглянуто далі). MTA, проаналізувавши заголовок повідомлення, визначає, що воно адресоване в поштовий домен scs.ntu-kpi.kiev.ua, який він обслуговує. MTA викликає агент доставки local, починає працювати програма цього агента і на її вхід подається текст повідомлення зі всіма заголовками. Агент доставки local виконує доставку повідомлення шляхом додавання повідомлення в форматі поштової скриньки до файлу /var/mail/поштова_скринька, і завершує роботу. Транспортний агент аналізує статус виходу (exit status) програми агента доставки, за яким визначає чи була доставка повідомлення успішною, чи виникла помилка. У випадку помилки MTA формує повідомлення про помилку, яке направляється відправнику листа (і, як правило, адміністратору поштового сервера). При успішному завершенні роботи агента доставки лист вважається доставленим адресату.

Повідомлення, що надійшло на локальний поштовий сервер, зберігається для пакетного пошуку аутентифікованими поштовими клієнтами (MUA). Повідомлення зчитуються клієнтськими поштовими програмами з використанням протоколу POP3, або протоколу IMAP, або поштовими системами сторонніх виробників (наприклад, Microsoft Exchange).

Отримувач (точніше, агент користувача MUA) може отримати доступ до своєї поштової скриньки двома способами.

1. Агент користувача працює на комп’ютері, де знаходиться поштовий сервер. В цьому випадку MUA просто зчитує повідомлення, що надійшло, із файлу /var/mail/поштова_скринька і зберігає його, при необхідності, в своєму каталозі для подальшого опрацювання. Цьому способу відповідає червоний пунктир на рисунку 8.1.
2. Користувач працює на іншому комп’ютері. В цьому випадку для доступу до файлу /var/mail/поштова_скринька через мережу використовується протокол POP3. Протокол POP3 забезпечує можливість користувачеві звернутися до свого поштового сервера і вилучити накопичену для нього пошту. При цьому він повинен запустити спеціальний поштовий агент (UA), що працює відповідно до протоколу POP-3, і налаштувати його для роботи зі своїм поштовим сервером. Повідомлення доставляються клієнтові за протоколом POP3, а надсилаються, як і раніше, за допомогою протоколу SMTP. Тобто на комп'ютері користувача існують два окремих агента-інтерфейси до поштової системи - агент доставки (POP) і агент відправки (SMTP). Цей варіант зображений на рисунку суцільною червоною лінією, що виходить від поштових скриньок. Детально протокол POP3 буде розглянутий в іншому розділі. Оскільки протокол POP3 працює поверх протоколу TCP/IP, немає ніяких обмежень на розташування комп’ютера клієнта (червона лінія на рисунку 8.1, яка виходить від POP-сервера, загалом проходить через Інтернет).

Тепер розглянемо як відбувається відправка вихідного повідомлення від one@scs.ntu-kpi.kiev.ua до somebody@mail.ru (сині стрілки). Повідомлення надходить до транспортного агента двома способами в залежності від того, де працює відправник. Якщо відправник працює на поштовому сервері, то його

MUA безпосередньо звертається до транспортного агента і передає йому повідомлення для somebody@mail.ru (синій пунктир). Якщо ж відправник працює на іншому комп'ютері, то його MUA зв'язується з транспортним агентом через мережу Інтернет по протоколу SMTP. (Знову ж, так як протокол SMTP працює поверх TCP/IP, немає ніяких обмежень на місцезнаходження комп'ютера відправника - синя лінія на рисунку, що виходить з комп'ютера відправника, загалом проходить через Інтернет.)

Отримавши повідомлення, MTA аналізує його заголовок і визначає, що це повідомлення направлене в інший поштовий домен і не підпадає ні під які особливі випадки (наприклад, не повинно бути доставлене через протокол обміну файлами між комп'ютерами UUCP (Unix-to-Unix CoPy) або відправлене по факсу – це все визначається конфігурацією MTA. Як наслідок, для доставки повідомлення вибирається агент SMTP, при цьому MTA виконує запит до DNS-сервера щоб визначити, хто є обробником пошти в домені mail.ru (DNS-сервер має повернути IP-адресу поштового сервера домену mail.ru). Ця адреса разом із текстом повідомлення буде передана агенту доставки, який по протоколу SMTP виконає з'єднання за вказаною адресою і, таким чином, відправить повідомлення транспортному агенту поштового сервера mail.ru. Після того, как всі повідомлення в цей домен будуть прийняті, з'єднання завершиться.

Якщо під час цієї передачі виникне не фатальна помилка (наприклад, віддалений сервер буде тимчасово вимкнений), то агент SMTP завершить роботу зі статусом «Відкладено», і MTA помістить повідомлення в чергу для повторної відправки. Поштова служба буде робити спроби доставити лист адресату кожні 4 години протягом 5 днів, і лише якщо за цей час зв`язок з віддаленим поштовим сервером не буде відновлений, поверне цей лист назад з поміткою «Мережа недоступна».

Розглянемо детально як відбувається пошук IP-адреси поштового сервера отримувача поштового повідомлення (рисунок 8.2).

Поштовий клієнт з'єднується з MTA по виділеному для протоколу SMTP порту 25 і в першу чергу передає йому свої облікові дані. Авторизувавши користувача, MTA приймає повідомлення і намагається передати його далі. Для знаходження мережевої адреси поштового сервера отримувача поштові сервери використовують глобальну систему доменних імен (DNS). Сервер, ім'я якого вказано в записі MX в базі даних DNS поштового домену призначення, є поштовим сервером отримувача. Вся пошта, відправлена на адресу цього поштового домену, надходить на цей сервер, який виконує певні дії по опрацюванню отриманого повідомлення.

Агент передачі SMTP відправляє запити локальному серверу DNS для отримання MX-запису на основі формату електронної пошти, тобто, якщо поштова адреса адресата знаходиться в домені example.com, поштовий агент направить запит для отримання MX-запису DNS-сервера домену example.com, а якщо поштова адреса адресата знаходиться в домені mail.ru, агент SMTP направить запит для отримання MX-запису DNS-сервера домену mail.ru.

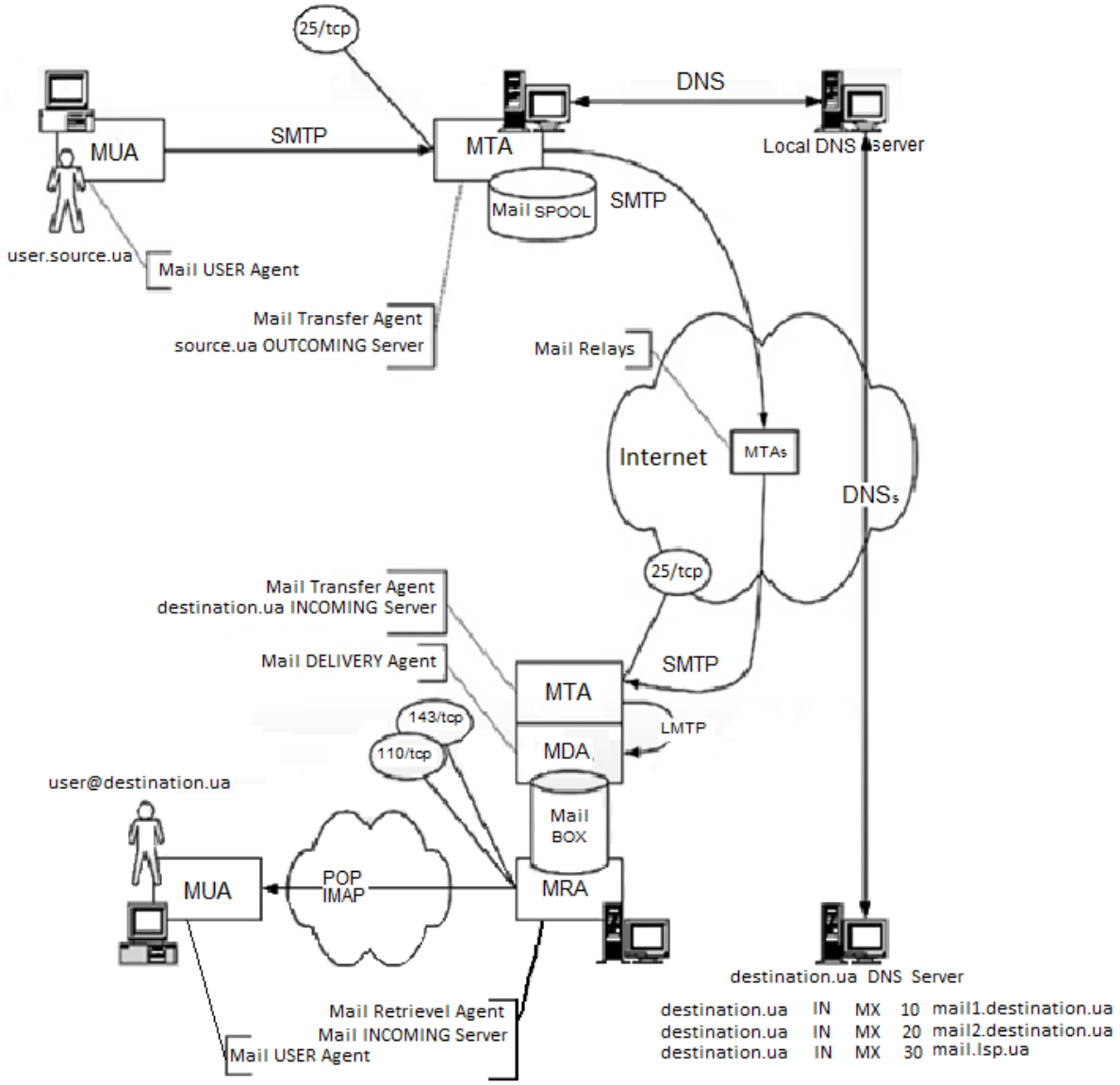


Рисунок 8.2 – Взаємодія поштового сервера і сервера DNS

Ще раз зазначаємо, що MX-запис визначає доменне ім’я системи обміну поштою для поштових повідомлень, що надходять в цей домен. За допомогою записів цього типу вказуються адреси (доменні імена) поштових серверів, які обробляють та перенаправляють пошту для цього домену. Зауважимо, що адресою поштового серверу в MX-записі не може бути IP-адреса, тільки доменне ім’я.

Коли DNS-сервер повертає ім’я поштового сервера отримувача, поштовий сервер відправника звертається до системи DNS повторно. Тепер він використовує DNS для того, щоб перетворити доменне ім’я поштового сервера отримувача на його IP-адресу.

Кожному поштовому серверу також повинен відповідати запис типу A в базі даних DNS цього домену.

Наприклад:

```
destination.ua.  IN      MX    10   mail.destination.ua.
destination.ua.  IN      MX    20   mail2.destination.ua.
destination.ua.  IN      MX    30   mail3.isp.ua.

; the local ( in-zone ) mail server (s)  need an A record

mail        IN      A     192.168.0.3
mail2       IN      A     192.168.0.4
mail3       IN      A     192.168.1.3
```

Для одного домену може використовуватись кілька MX-записів (з різними пріоритетами). Якщо один із поштових серверів (з найвищим пріоритетом) не може отримати пошту, то пошта відправляється на поштові сервери з меншим пріоритетом (так звані запасні поштові сервери). Запасні поштові сервери будуть намагатися передати повідомлення на основний сервер пізніше.

Якщо запис MX для поштового домену отримувача не буде знайдений в DNS, буде виконана спроба знайти запис типу A (IP-адреса) для того ж доменного імені, тобто робиться припущення, що ім'я поштового домену і є іменем реального комп'ютера і на цьому комп'ютері запущений MTA – це справедливо для більшості Unix-систем. Агент доставки буде намагатися доставити повідомлення безпосередньо за цією адресою.

При неможливості відправити повідомлення, воно повертається відправнику (приймається в поштову скриньку відправника) з повідомленням про помилку.

Коли повідомлення має лекілька адрес призначення з одного домену, модуль SMTP може передати поштовому серверу, що обслуговує домен, тільки одну копію повідомлення, і цей сервер самостійно виконає доставку повідомлення всім отримувачам в цьому домені. Але якщо повідомлення розсилається за значною кількістю адрес, модуль SMTP може розділити їх на кілька частин і відправити кілька копій повідомлення, кожна з яких буде містити тільки частину списку адрес.

Якщо для одного домену є кілька повідомлень, то модуль SMTP може відкрити кілька з'єднань з поштовим сервером, що обслуговує цей домен і відправити ці повідомлення одночасно.

1.2. Деякі стандартні засоби захисту від спаму

Розглянемо деякі стандартні засоби захисту від спаму.

1. Отримавши вхідне повідомлення, MTA перевіряє ім'я домена отримувача в адресній частині повідомлення. Якщо домен входить до переліку тих, що обслуговуються даним MTA, обробка повідомлення продовжується, якщо ні – сервер поштове повідомлення не приймає. Після перевірки імені домену перевіряється ім'я отримувача. Якщо він присутній у переліку користувачів, повідомлення надходить в його поштову скриньку, якщо ні, то можливі два

варіанти: повна відмова від прийому повідомлення або прийом повідомлення в загальну поштову скриньку (поштову скриньку адміністратора). З одного боку таке налаштування дещо збільшує кількість спаму, що приймається, але з іншого – дозволяє не загубити повідомлення з помилками в написанні адреси.

2. Ще одним заходом захисту від спаму є запит вказівника в зворотній зоні DNS PTR-запису (PoinTeR), який зв'язує IP-адресу з іменем домену. Аналізуючи PTR-запис, MTA приймає поштове повідомлення тільки у тому випадку, якщо ім'я домену відправника у заголовку повідомлення збігається з іменем домену, де знаходиться сервер-відправник.

Розглянемо цей випадок детальніше (рис. 8.3). Деякий спамерський сервер spam.com намагається розіслати листи з підробленою адресою відправника начебто від відомого нам довіреного сервера example.com. З метою боротьби зі спамом MTA example.org формує PRT-запит для IP-адреси сервера-відправника, який той повідомляє під час SMTP сесії. Для адреси y.y.y.y PTR-запит поверне ім'я домену spam.com, яке не збігається з іменем домена відправника, що буде причиною відмови в прийомі даного повідомлення. В той же час повідомлення від сервера x.x.x.x будуть отримані, оскільки домен із PTR-запису для x.x.x.x (example.com) збігається з іменем домена відправника. Якщо DNS-зона налаштована неправильно, сервер-отримувач, перевіривши PTR-запис, відхилить поштове повідомлення, оскільки ім'я сервера-відправника не збігається з результатом зворотного DNS-запиту.

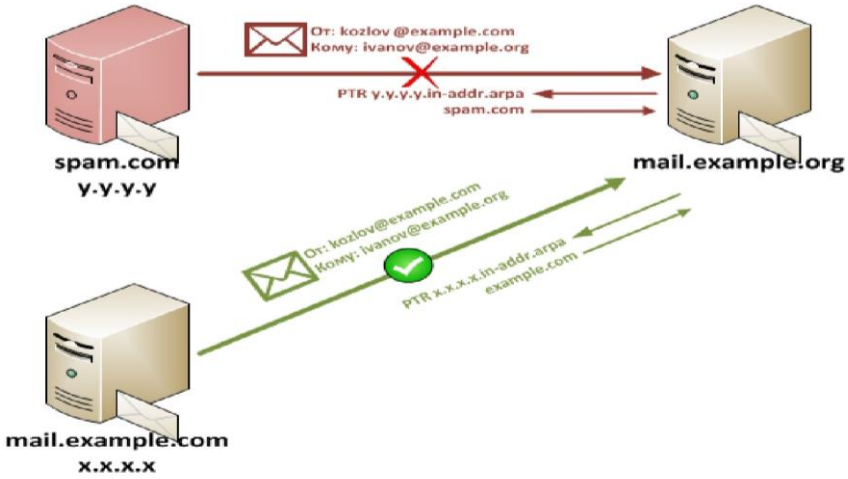


Рисунок 8.3 – Процедура захисту від спаму

Відсутність PTR-запису практично завжди веде до відмови в прийомі повідомлення, оскільки існує мовчазна угода про те, що добропорядний відправник зобов'язаний забезпечити вірно налаштовану зворотну зону.

3. Із метою боротьби зі спамом використовується порівняно нова технологія SPF (Sender Policy Framework). Цей засіб захисту від підміни адреси відправника прийнято як стандарт RFC з 2006 року і використовує дуже мало ресурсів як відправника, так і отримувача.

Ця технологія дозволяє створювати спеціальний запис SPF в DNS домені, який явно вказує список IP-адрес серверів, які мають право відправляти пошту від імені домена-відправника і механізм обробки повідомлень, відправлених від інших серверів.

У звичайному варіанті SPF-запис має такий вигляд:

example.com. IN TXT "v=spf1 +a +mx -all"

Він означає, що від імені домену example.com пошту можуть відправляти вузли, вказані в A-записах (+a) і в MX-записах (+mx), всі інші поштові повідомлення від імені домену example.com мають бути відхилені (-all); v=spf1 – версія, завжди spf1. Запис SPF зберігається в TXT-записі домену.

4. Налаштування електронної пошти, що ідентифікована доменними ключами DKIM (DomainKeys Identified Mail) – це метод e-mail аутентифікації, що базується на перевірці справжності цифрового підпису. Публічний ключ зберігається в TXT-записі домену.

DKIM – це криптографічна технологія, запроваджена як стандарт з 2007 року. Як і SPF, покликана запобігти підробці адрес відправника, унікальним чином ідентифікувати його, а також підтвердити, що в процесі доставки повідомлення електронної пошти не було змінене. Потребує підтримки поштового сервера для підписування кожного повідомлення доменним ключем. Для перевірки використовується відкритий ключ, вказаний в DNS зоні.

DKIM потрібен для того, щоб поштові сервіси могли перевіряти достовірність відправника, тобто, захищає отримувача повідомлення від шахрайських листів (листів, які відправлені з підміною адреси відправника).

Прикладом DKIM-записів може бути такий:

mail._domainkey.tld TXT "v=DKIM1; k=rsa; t=s; p=<публічный ключ>"

5. Ідентифікація повідомлень, створення звітів і визначення відповідності за доменним іменем DMARC-специфікація (Domain-based Message Authentication, Reporting and Conformance) – це технічна специфікація, створена групою організацій для зменшення кількості спамових і фішінгових електронних листів, яка базується на ідентифікації поштових доменів відправника на основі правил і параметрів, заданих на поштовому сервері отримувача. Тобто, поштовий сервер сам вирішує, довіряти повідомленню чи ні, і діє згідно з DMARC-записом.

1.3. Основні команди протоколу SMTP

Для пересилки поштових повідомлень через Інтернет між транспортними агентами та від агентів користувача до транспортних агентів використовується протокол SMTP (Simple Mail Transfer Protocol), порт якого має значення 25.

Після встановлення з'єднання з клієнтом сервер чекає введення команд і даних в текстовому вигляді (рядкові та прописні букви в командах не розрізняються). Реакція серверу полягає у видачі тризначного числового коду з текстовими коментарями.

Числовий код призначений для автоматичної обробки відповідей серверу програмою-клієнтом. Код, що починається з 2, є позитивною відповіддю, з 3 – проміжним позитивним відгуком (тобто після введення команди очікуються додаткові дані), коди виду 5xx сигналізують про помилку.

Основні команди SMTP:

HELO hostname protocol
- перша команда сеансу, hostname - доменне ім'я викликаючого хоста (клієнта).

MAIL FROM: email_адреса_від_кого
- зворотна адреса.

RCPT TO: email_адреса_кому
- адреса одержувача (у разі декількох адресатів команда повторюється для кожного адресата).

DATA
- початок введення тексту повідомлення; сервер посилає проміжний позитивний відгук 354 і розглядає всі подальші рядки як текст (тіло) повідомлення; кінець введення - рядок, що містить одну крапку. Перед текстом повідомлення вводяться поля заголовку. Кожне поле заголовку повинне починатися з нового рядка. Між заголовком і текстом повідомлення має бути *один незаповнений рядок*.

VRFY *email_адреса*
- надходить позитивний відгук (250, 251 або 252), якщо сервер може *спробувати* доставити повідомлення за вказаною адресою; інакше надходить негативний відгук 550. Якщо *email_адреса* - не локальна адреса на сервері, то позитивний відгук не обов'язково означає, що ця адреса існує. Для локальних адрес проводиться підстановка з файлу /etc/aliases (якщо адреса там вказана) і в полі текстового коментаря виводиться результат, часто до нього додається справжнє ім'я користувача.

EXPN email_addr
- якщо email_addr - локальна адреса списку розсилки, то вивести адреси в цьому списку; інакше поведінка команди не визначена. Як правило, якщо email_addr - локальна адреса, то виконується підстановка з файлу /etc/aliases (якщо там ця адреса вказана); інакше просто видається позитивний відгук 250.

RSET

- повернення сеансу до початкового стану (як після введення HELO).

QUIT
- кінець зв'язку.

Команди EXPN і VRFY не обов'язково виконуються сервером; часто їх відключають з міркувань секретності. Для передачі пошти ці команди не потрібні; фактично, вони призначені для людини. Дії цих команд в стандарті чітко не визначені, і їх реалізація не є обов'язковою.

Існують також додаткові команди - так званий розширений SMTP (ESMTP). Не всі сервери підтримують команди ESMTP, і кожний сервер може підтримувати якусь свою підмножину ESMTP-команд, у тому числі нестандартних. Для того, щоб з'ясувати, які додаткові команди підтримуються, необхідно замість HELO виконати команду EHLO.

1.4. Основні команди протоколу POP3

Протокол POP (Post Office Protocol) використовується для пересилки нової пошти користувача із сервера на робочу станцію користувача. В Linux роль POP-серверу виконує програма qpopper, до складу якої входить демон qpopper і програма pop-auth, яка управляє аутентифікаційною базою даних. Зараз використовується версія 3; по складу команд вона несумісна з попередніми версіями.

Номер порту сервера POP – TCP/110. Після встановлення з'єднання із клієнтом сервер чекає введення команд і даних в текстовому вигляді (рядкові та прописні букви в командах не розрізняються). Реакція серверу - рядок починається з мітки "+OK" або "-ERR", за якою слідує текстовий коментар і, якщо команда цього вимагає, з нового рядка виводяться дані (текст повідомлення або лістинг повідомлень). Дані закінчується рядком, у якому записаний тільки один символ "." ("крапка"). Якщо серед даних є такий рядок, то крапка в цьому рядку подвоюється.

Розглянемо основні команди протоколу POP3.

USER і*м'я_користувача*
- перша команда сеансу, вводиться і*м'я* користувача (ідентифікатор поштової скриньки).

PASS *пароль*
- друга команда сеансу, вводиться пароль.

STAT
- після мітки "+OK" виводить два числа: число повідомлень і їх загальний об'єм в байтах.

LIST *n*

- якщо *n* вказане, то після мітки "+OK" виводиться розмір повідомлення номер *n* в байтах. Інакше виводиться список з двох колонок: номер повідомлення, пропуск, розмір повідомлення в байтах; список закінчується рядком, що містить тільки символ "." ("крапка").

RETR *n*
- виводить повідомлення номер *n*. Вивід закінчується рядком, що містить тільки символ "." ("крапка").

DELE *n*
- видаляє повідомлення номер *n* з серверу; при цьому нумерація повідомлень не змінюється, а всі видалені *в даному сеансі* повідомлення можуть бути відновлені командою REST.

LAST
- після мітки "+OK" виводиться номер останнього за часом повідомлення, для якого була виконана команда RETR; ця інформація зберігається між сеансами, що дозволяє не запрошувати вже отримані користувачем, але не видалені з серверу, повідомлення.

TOP n m
- виводить заголовок і *m* перших рядків повідомлення номер *n*. Вивід закінчується рядком, що містить тільки символ "." ("крапка").

RSET
- відміняє видалення всіх повідомлень, видалених в даному сеансі.

NOOP
- немає операції.

QUIT
- кінець зв'язку.

1.5. Структура email-повідомлення

Розглянемо структуру поштового повідомлення. Воно складається з тіла повідомлення (тексту, створеного відправником) і службової інформації.

Службову інформацію розділяють на два типи - конверт (інформація, необхідна для доставки листа) і заголовок.

Нижче наведений перелік основних полів заголовку і їх призначення.

From: one@scs.ntu-kpi.kiev.ua

Поштова адреса відправника (в залежності від типу MUA може мати різний формат)

To: somebody@mail.ru, “Some S” <somebody2@mail.ru>

Містить перелік поштових адрес отримувачів (в залежності від типу MUA може мати різний формат). Адреси в переліку розділені комами.

Cc: somebody3@mail.ru, somebody4@mail.ru

аналогічно **To**

Bcc: oops@mail.ru

Аналогічно Cc. Єдина відмінність, що адреси, перелічені в Bcc, не будуть включені в заголовок повідомлень, що доставляються отримувачу**.**

Subject: to study

Тема листа. Не обов’язкове поле. Може використовуватися MUA для фільтрації пошти.

Date: Sun, 23 Jul 2003 18:29:08 +1100

Дата відправлення листа.

Reply-To: somebody5@mail.ru

Адреса, за якою необхідно відправляти відповідь на лист. Якщо поле не заповнено, відповідь прийде за адресою, вказаною в From.

Message-ID: <>

Рядок, згенерований MTA для ідентифікації повідомлення.

Received: ukrpost.net .

Поле, вбудоване у заголовок кожним вузлом, через які проходить повідомлення (включаючи станції отримувача та відправника). Вказується ім’я вузла, ідентифікатор повідомлення, час і дата отримання, з якого вузла отримано і яка MTA використана.

Розглянемо багатоцільове розширення інтернет-пошти **MIME** (Multipurpose Internet Mail Extensions), яке описує передачу різних типів даних електронною поштою, а також специфікації для кодування інформації та форматування даних таким чином, щоб їх можна було пересилати в мережі Інтернет.

MIME визначає механізми для передачі довільної інформації всередині текстових даних (зокрема, за допомогою електронної пошти), а саме: текст

мовами, для яких використовується кодування, відмінне від ASCII, і не текстовий контент, такий, як рисунки, музика, фільми і програми. MIME є також фундаментальним компонентом комунікаційних протоколів, таких як HTTP, яким потрібно, щоб дані передавались в контексті повідомлень, подібних e-mail, навіть якщо дані реально не є e-mail-подібними.

Основний формат електронних повідомлень визначений в RFC-5322, який є оновленою версією RFC-2822 (який, в свою чергу, є оновленою версією RFC-822). Ці стандарти визначають формати текстових e-mail-заголовків. MIME визначає набір e-mail-заголовків для визначення додаткових атрибутів повідомлення, включаючи тип контенту, і визначає множину методів кодування, які можуть бути використані для представлення 8-бітових бінарних даних за допомогою символів із 7-бітового ASCII. MIME також визначає правила для кодування символів із Extended ASCII (з кодами 128-255) в заголовках e-mail-повідомлення, таких, як Subject.

Специфікація MIME відкрита для новых типів – її визначення включає метод для реєстрації нових типів контенту та інших атрибутів.

MIME - специфікації, які визначають додатки у форматі поштових повідомлень для:

- передачі восьмибітних текстів і повністю двійкових даних (RFC-822 регламентувала пересилку даних в ASCII – тобто, 7-и бітових даних);
- підтримки складних повідомлень, що складаються з декількох розділів (можливо, тих, що містять дані різних типів).

Формування і розбір повідомлень у відповідності до специфікацій MIME проводиться поштовими агентами користувача. Опис MIME знаходиться в RFC2045 ÷ 2049.

Для виконання вказаних задач вводяться додаткові заголовки "Content-Type:" і "Content-Transfer-Encoding:", які визначають відповідно тип даних, що містяться в повідомленні (розділі повідомлення), спосіб представлення цих даних, і заголовок "MIME-Version: 1.0", який вказує версію MIME і використовується для позначення того, що це повідомлення є не звичайним email-повідомленням, згідно RFC-822, а складене згідно специфікації MIME. Заголовок "Content-Type:" має формат:

Content-Type: *тип/підтип* [; *параметр=значення* [;...]]

Параметр (параметри) для деяких типів даних повинні бути присутні, для інших – необов'язкові. Неопізнані параметри ігноруються.

1.6. Програма Sendmail: її функції і складові частини

Програма Sendmail – це сукупність транспортного поштового агента (MTA) і агента доставки SMTP. За отримання самої пошти в Sendmail відповідає інший сервер POP3, який базується на програмі **cucipop(Cubic Circle POP3 daemon).** Sendmail фактично є стандартним MTA для Unix-

подібних систем і надається разом з операційною системою. Програма розповсюджується безкоштовно, як в початкових кодах, так і в rpm. Нижче описуються тільки ключові моменти, необхідні для роботи.

Sendmail складається з програми /usr/lib/sendmail, конфігураційного файлу /etc/mail/sendmail.cf, файлу псевдонімів /etc/mail/aliases і інших допоміжних файлів, а також документів довідника man. Програма використовує каталоги /etc/mail для зберігання конфігураційного і допоміжних файлів і /var/spool/mqueue для черги повідомлень. Шлях до каталогу черги повідомлень може змінюватися від системи до системи. Конфігураційний файл /etc/mail/sendmail.cf напряму змінювати не можна. Редагувати потрібно файл з розширенням .mc, а потім створювати із нього sendmail.cf.

Програма /usr/lib/sendmail виконує різні дії залежно від того, з якими ключами або під яким ім'ям вона запущена, наприклад:

#/usr/lib/sendmail -bd

Запуск в режимі демона (відбувається при старті системи) для прийому SMTP-з'єднань.

#/usr/lib/sendmail -bt

Запуск в тестовому режимі для перевірки конфігурації.

#/usr/bin/newaliases

Перегляд файлу /etc/mail/aliases і оновлення бази даних aliases (/usr/bin/newaliases – посилання на /usr/lib/sendmail).

#/usr/bin/mailq

Перегляд вмісту черги повідомлень (/usr/bin/mailq – посилання на /usr/lib/sendmail).

%/usr/lib/sendmail somebody@scs.ntu-kpi.kiev.ua

Виклик MTA для обробки і відправки повідомлення до **somebody**. Повідомлення з усіма заголовками передається на стандартний вхід. Зазвичай такий виклик проводиться призначеним для користувача агентом.

Основні задачі адміністратора sendmail полягають в наступному:

- визначення псевдонімів і списків розсилки;
- внесення змін і доповнень у файл sendmail.cf.

1.6.1. Псевдоніми, списки розсилки і форвардінг

Файл /etc/aliases (->/etc/mail/aliases) дозволяє визначити альтернативні імена для одержувачів пошти, зв'язати стандартні імена типу MAILER-DAEMON з реальними одержувачами і створити списки розсилки.

Файл складається з текстових рядків формату:

псевдонім: одержувач[,одержувач ...]

Наприклад:

#обов’язкові записи

MAILER-DAEMON: postmaster

postmaster: root

#альтернативні імена для somebody

abba: somebody

abba.somebody: somebody

somebody: usomebody@mail.ru

#список розсилки

friends: somebody, another@scs.ntu-kpi.kiev.ua, hey

#адміністратор списку friends (його ім'я: "owner-ім’я_списку")

owner-friends: bestman

#вставити список розсилки з іншого файлу

staff: :include:/home/manager/mail/staff

#додати повідомлення в кінець файлу (обов'язково вказується повний шлях до файлу)

filer: /var/tmp/add_to_this_file

одержувачем повідомлення є програма, яка запускається

і текст повідомлення подається їй на стандартний вхід

myprog: |"/usr/local/bin/do_something"

Щоб записи в зміненому файлі /etc/aliases були прочитані програмою sendmail, необхідно виконати команду newaliases.

Файл /etc/aliases є системним і редагується адміністратором. За допомогою директиви ":include:" зручно доручати формування списку розсилки іншому (звичайному) користувачу. Адреси у файлі, що підключається директивою ":include:", розташовуються по одиному на рядок. Рядки, що починаються з символу #, ігноруються.

Кожний користувач може створити в своєму домашньому каталозі файл .forward зі списком поштових адрес, поодинці на рядок. В цьому випадку вся пошта, що приходить цьому користувачу, буде передаватися за вказаними адресами, замість доставки користувачу. Щоб вхідна пошта залишалася також і в локальній поштовій скриньці, її адреса має бути також включена в .forward.

Файл .forward обробляється після файлу /etc/aliases. Файл .forward може також містити директиви перенаправлення пошти в програму або додавання до файлу аналогічно тому, як це робиться в /etc/aliases.

1.6.2. Конфігурація sendmail (файл sendmail.cf)

Дії sendmail з обробки заголовків поштових повідомлень і визначення способів і шляхів доставки повідомлень визначаються конфігураційним файлом /etc/mail/sendmail.cf, який має спеціальний синтаксис. У цьому файлі описані службові макроси (підстановки з одного елемента) і класи (набори елементів); правила (правила переписування заголовків і обробки повідомлень), згруповані в набори правил; агенти з доставки повідомлень тощо.

Файл sendmail.cf можна розглядати як програму з обробки заголовку поштового повідомлення (в основному, адреси відправника і отримувача). Центральними елементами цієї програми є правила, що виконують роль операторів розгалуження та циклу, і набори правил, що виконують роль функцій. Макроси і класи можна порівняти зі змінними та масивами. Результатом роботи цієї "програми" є сформовані (можливо, змінені) для кожного поштового повідомлення заголовки (особливо це стосується заголовків "From:" і "To:") і вибір агента доставки.

Крім того, в конфігураційному файлі містяться визначення агентів доставки і опції, що визначають ті або інші аспекти роботи sendmail.

7. Установка сервера POP3

Сервер POP3 під Unix реалізований у вигляді однієї програми-демона, наприклад, програми popper. Для її запуску необхідно внести номер порту для сервісу POP3 (порт 110/tcp) в /etc/services і відповідний рядок в /etc/inetd.conf, наприклад:

```
POP3  stream tcp   nowait root /usr/local/lib/popper  popper -s
```

Агент завершує роботу з наступними станами виходу (exit status):

EX_OK = 0 – повідомлення доставлено
EX_NOUSER = 67 – немає такого користувача (користувач не знайдений в userlist або файл userlist відсутній)
EX_USAGE = 64 – невірний виклик агента (не вказано ім'я користувача в командному рядку)
EX_SOFTWARE = 70 – помилка в програмі агента (наприклад, помилка при відкритті файлу).

(Якщо стан виходу агента не дорівнює нулю, sendmail, що запустив цього агента, поверне відправнику відповідне повідомлення про помилку.)

8. Конфігурування Sendmail

Всі конфігураційні файли Sendmail знаходяться в каталозі */etc/mail*. В більшості випадків, коли сервер виконує прийомом вхідної пошти, з боку

користувача будуть вимагатися всього чотири файли: */etc/mail/access*, */etc/mail/aliases*, */etc/mail/mailertable* и */etc/mail/relay-domains*.

Файл *access* дозволяє управляти доступом до поштового сервера на рівні доменів і окремих хостів.

Файл *aliases* містить карту перенаправлень електронної пошти для локального хосту.

Файл *mailertable* дозволяє перевизначити записи типу MX для даного поштового сервера. Цей файл особливо зручний при використанні системи FreeBSD для забезпечення додаткового рівня захисту в системах з недостатнім рівнем захисту, таких як Microsoft Exchange.

Файл *relay-domains* містить перелік імен доменів і адрес, для яких наш сервер буде виконувати функції ретранслятора електронної пошти. В основному, в цей список будуть включатися клієнти, які використовують поточний комп'ютер як поштовий сервер, однак в цей список можуть потрапити й інші домени, для яких цей сервер буде виконувати роль резервного ретранслятора пошти.

Для кожного з цих файлів (за виключенням *relay-domains*) існує відповідна база даних, файл з розширенням *.db*. Подібно до файлів */etc/passwd.db* і */etc/spwd.db*, вони створюються з текстових файлів для забезпечення програм більш швидким способом доступу до даних. Після внесення змін у текстовий файл потрібно оновити і відповідний файл бази даних, про що більш детально буде сказано при розгляді кожного конкретного файлу.

Розглянемо більш детально кожний з цих файдів.

Файл */etc/mail/access* дозволяє явно визначити, який модуль може передавати пошту через поточну систему. Цей файл зазвичай використовується для обмеження прийому пошти від небажаних кореспондентів. За замовчуванням Sendmail відкидає будь-які поштові повідомлення, не призначені для локальної системи. Цей файл дозволяє перевизначити інші правила Sendmail, зокрема - чорні списки. База даних з правилами доступу, в ідеалі, має використовуватись для опису окремих винятків із інших правил, які описуються параметрами налаштування Sendmail, хоча є і задачі, які можна вирішити тільки за допомогою бази даних **access**.

Файл ***/etc/mail/access*** складається з двох стовпчиків. В лівому стовпчику вказуються ім'я домену, ім'я хосту, IP-адреса або блок IP-адрес, звідки приходить електронна пошта. В правому стовпчику вказуються інструкції, що визначають дії з електронними повідомленнями. Це може бути одна з інструкцій: RELAY, OK, REJECT або DISCARD. Крім того, тут же за допомогою інструкції ERROR можна визначити текст власного повідомлення про помилку для певних хостів або доменів.

Інструкція OK означає, що демон Sendmail має тільки приймати, але не ретранслювати електронні повідомлення від вказаних хостів.

Більш сувора інструкція REJECT, яка категорично наказує відхиляти будь-яке повідомлення, яке надходить від вказаного джерела. Допустимо,

модуль з IP-адресою 192.168.8.83 передає дуже значні об’єми електронної пошти, намагаючись викликати переповнення жорсткого диску на поштовому сервері. В цьому випадку можна відмовитись від прийому пошти за допомогою інструкції REJECT:

192.168.8.83 REJECT

Можна відхиляти пошту, відправляючи у відповідь власне повідомлення про помилку з текстом повідомлення після інструкції ERROR. Зазвичай, відхиляючи приймання пошти, Sendmail генерує код повідомлення 550. Текст повідомлення потрібно обмежувати лапками, щоб Sendmail не видалив пропуски і не змінив його яким-небудь іншим способом.

suckycompany.com ERROR: 550 "Ваш хост постійно надсилає спам"

Нарешті, деякі хости можуть бути настільки недружніми до нас, що ми навіть бачити не хочемо пошту, яка надходить від них. І навіть не бажаємо відхиляти повідомлення від них. В цьому випадку потрібно використовувати інструкцію DISCARD:

absolutefreebsd.com DISCARD

Файл *<u>/etc/mail/aliases</u>* містить шляхи перенаправлення електронної пошти для конкретних облікових записів або користувачів. Кожний рядок в цьому файлі починається з псевдоніму, а після нього через дві крапки вказується перелік активних користувачів системи, яким повинна бути перенаправлена пошта. Почтові псевдоніми враховуються як в конфігурації прийому, так і в конфігурації передачі пошти.

Пересилання пошти від одного користувача до іншого реалізується наступним чином.

Часто адміністратори надають перевагу пересиланню пошти, відправленої користувачу root, на дійсний обліковий запис. Приведений нижче приклад демонструє, як організувати пересилання пошти користувача root на інший обліковий запис:

root: mwlucas@AbsoluteFreeBSD.com

В дійсності багато адрес електронної пошти не мають облікових записів, зв’язаних з ними. Наприклад, обов’язкова адреса *postmaster@* зазвичай не пов’язана з дійсним обліковим записом. Пересилати таку електронну пошту за дійсним обліковим записом можна за допомогою псевдоніму:

postmaster: root

Згідно з цими двома прикладами пошта для postmaster буде пересилатися за адресою root, а з адреси root - на адресу mwlucas@AbsoluteFreeBSD.com,

тобто адресат mwlucas@AbsoluteFreeBSD.com буде отримувати всю пошту, яка адресована адресатам *root@* і *postmaster@* на цьому комп'ютері.

Файл *aliases* вже містить кілька стандартних адрес інтернет-сервісів, а також псевдоніми для всіх облікових записів сервісів FreeBSD, які використовуються за замовчуванням, які виконують переадресацію на обліковий запис root. Визначивши дійсну адресу електронної пошти для псевдоніму root, можна автоматично отримувати всі системні електронні повідомлення.

При визначенні псевдоніму можна також перерахувати кілька користувачів, створюючи таким чином невеликі локальні списки розсилки електронної пошти. Такий прийом не підходить для організації динамічних списків, коли користувачі часто підписуються та відписуються, але він цілком підходить для швидкої організації простих списків:

```
sales:   mwlucas, bpollock, sales@nostarch.com
```

Пересилка пошти в файли реалізується наступним чином.

За допомогою файлу *aliases* можна реалізувати ще більш цікаву можливість — пересилку повідомлень в інші пункти призначення, що не є адресами електронної пошти. Якщо вказати ім'я файлу, Sendmail буде дописувати повідомлення в цей файл. Можна організувати протоколювання всієї пошти, яка надходить на електронну адресу, наприклад так:

```
mwlucas:   /var/log/mwlucas-mail, mwlucas
```

Пересилка пошти програмам реалізується наступним чином.

Можна пересилати пошту програмам для автоматичної обробки. Для цього потрібно просто вказати повний шлях до програми після символу вертикальної риски (|). Наприклад, якщо є сценарій для обробки вхідної пошти, яка надходить на певну адресу, організувати пересилку пошти можна за допомогою такого визначення псевдоніму:

```
orders:   |/usr/local/bin/process-orders.pl
```

Нарешті, в файл *aliases* можна включати вміст інших файлів. Це дозволяє довіреним користувачам змінювати свої псевдоніми самостійно. Файл, вміст якого включається, повинен містити список електронних адрес, по одній адресі в рядку.

```
clientlist:   include:/usr/home/salesdude/clientlist.txt
```

Файл *mailertable* дозволяє вибірково перевизначати записи MX. Така можливість дозволяє перетворити FreeBSD в особливий брандмауер. Файл mailertable особливо зручний при використанні системи FreeBSD для забезпечення додаткового захисту в системах з недостатнім рівнем безпеки,

таких як Microsoft Exchange. Задіявши комп'ютер FreeBSD як загальнодоступний ретранслятор пошти і файл *mailertable* на ній, можна організувати захист системи Exchange без використання дорогого брандмауера електронної пошти. Приклад: нехай в системі присутні хости з назвами *freebsd.absolutefreebsd.com* і *exchange.absolutefreebsd.com*. Тоді запис DNS мав би мати такий вигляд:

```
absolutefreebsd.com IN  MX  10  freebsd.absolutefreebsd.com
```

Тепер всі можуть надсилати електронну пошту на машину з операційною системою FreeBSD.

На цій машині в файл *mailertable* слід додати такий рядок:

```
.absolutefreebsd.com  smtp:[exchange.absolutefreebsd.com]
```

Це дозволяє комп'ютеру з операційною системою FreeBSD ігнорувати запис MX для домену, вказаного в лівому стовпчику, і пересилати всю пошту для цього домену на хост *exchange.absolutefreebsd.com*.

Файл *relay-domains* містить перелік імен доменів і адрес, для яких поточний сервер буде виконувати функції ретранслятора електронної пошти. В основному, в цей перелік будуть включатися клієнти, які використовують цей комп'ютер як поштовий сервер, проте в цей перелік можуть попасти й інші домени, для яких поточний сервер буде виконувати роль резервного ретранслятора пошти. Імена доменів і адрес потрібно вказувати по одному в рядку. Наприклад, запис, який створює сервіс резервного ретранслятора пошти для домену *absolutefreebsd.com має такий вигляд*:

```
absolutefreebsd.com
```

Активізація змін до розглянутих файлів реалізується наступним чином.

Для файлів *access*, *aliases* і *mailertable* існують відповідні файли бази даних (*.db*). Змінивши один з цих файлів, потрібно оновити і відповідний йому файл бази даних. Сучасні версії Sendmail забезпечують управління цими файлами бази даних за допомогою утиліти make(1). Щоб оновити файли *access* і *mailertable*, потрібно перейти в каталог */etc/mail* і виконати команду make maps:

```
# cd /etc/mail
# make maps
```

Утиліта make(1) порівняє час останньої зміни кожного файлу з часом останньої зміни відповідного файлу бази даних і виконає перезборку бази даних, якщо текстовий файл виявиться більш новим.

Щоб перебудувати базу даних *aliases*, потрібно виконати команду **newaliases(8)** або make aliases:

```
# cd /etc/mail
# make aliases
```

Зміни в базі даних починають діяти відразу після її перезборки. Однак інші зміни вимагають перезапус всієї поштової системи. Зміни в файлі *mailertable* належать саме до таких змін. Поштову систему можна перезапустити різними способами, але рекомендується використовувати для цього систему початкового запуску FreeBSD, щоб забезпечити нормальну поведінку поштового сервера після перезавантаження:

```
# /etc/rc.d/sendmail restart
```

Завдання

1. Відправити поштове повідомлення за адресою yuri_gol@mail.ru через безпосередній діалог з SMTP-сервером, заздалегідь встановивши адресу поштового сервера, який приймає пошту для цього адресата.

2. Забезпечити доступ до пошти для користувачів свого комп'ютера по протоколу POP3. Створити POP-користувача (користувача, який може тільки отримувати/відправляти пошту і змінювати пароль).

3. Отримати пошту по протоколу POP3 вручну.

Контрольні запитання

1. Організація служби електронної пошти в Інтернет.
2. Використання служби DNS для маршрутизації електронної пошти.
3. Структура email-повідомлення.
4. Специфікації MIME.
5. Основні команди протоколу SMTP.
6. Основні команди протоколу POP3.

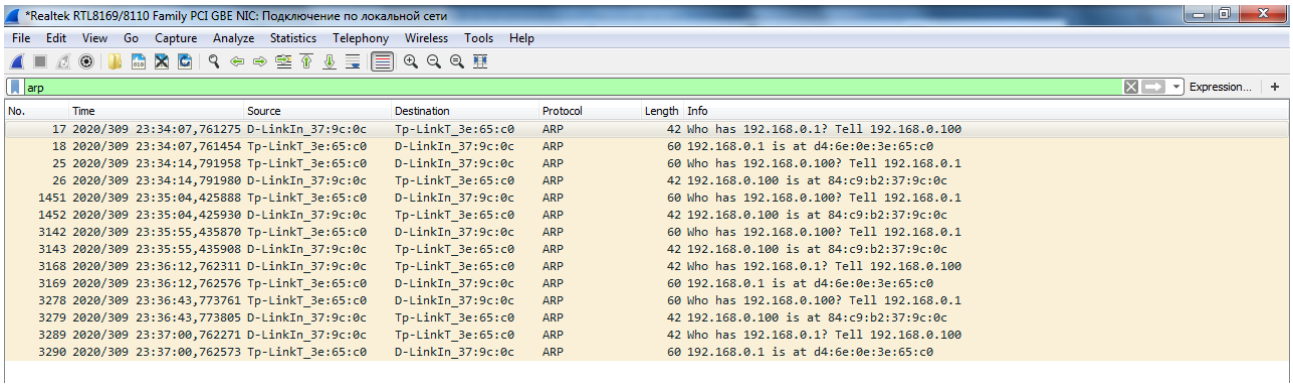
Додаток А

Основні типи фільтрів Wireshark

Фільтр трафіку протоколів канального рівня

Для спостереження за ARP- трафіком:

arp



Показати фрейми ARP протоколу, відправлені з пристрою, що має MAC-адресу 00:c0:ca:96:cf:cb:

arp.src.hw mac == 00:c0:ca:96:cf:cb

Показати фрейми ARP протоколу, відправлені з пристрою , що має IP-адресу 192.168.50.90:

arp.src.proto_ipv4 == 192.168.50.90

Показати фрейми ARP протоколу, відправлені на пристрій, що має IP-адресу 192.168.50.1:

arc.dst.proto_ipv4 == 192.168.50.1

Показати Ethernet трафік:

eth

Показати фрейми, відправлені з пристрою, що має MAC-адресу 00:c0:ca:96:cf:cb:

eth.src == 00:c0:ca:96:cf:cb 00

Показати фрейми, відправлені на пристрій, що має MAC-адресу 78:cd:8e:a6:73:be:

eth.dst == 78:cd:8e:a6:73:be

Фільтр трафіку протоколів мережевого рівня

Фільтрація IPv4 протоколу

Показати IP-трафік (TCP, UDP, а також протоколи рівня додатків DNS, HTTP — тобто практично все, крім протоколів канального рівня, які не

використовують IP-адреси для передачі даних (в локальних мережах Ethernet для доставки пакетів вони використовують MAC-адреси)):

ip

Якщо сказати точніше, мається на увазі трафік протоколу IPv4, який зазвичай називають просто IP (Internet Protocol).

Показати трафік, пов'язаний з певною IP-адресою. Будуть захоплені пакети, в яких ця IP-адреса є джерелом даних або отримувачем даних:

ip.addr == x.x.x.x

Показати трафік, пов'язаний з двома IP-адресами. Єдино можливий випадок, одна із цих адрес буде джерелом, інша — адресою доставки.

ip.addr == x.x.x.x && ip.addr == y.y.y.y

Показати трафік, джерелом якого є хост з IP-адресою 138.201.81.199:

ip.src == 138.201.81.199

Показати трафік, адресатом якого є хост с IP- адресою 138.201.81.199:

ip.dst == 138.201.81.199

Фільтрація підмереж і діапазонів IP-адрес

Можна замість однієї IP-адреси вказати адресу підмережі:

ip.addr == 92.168.1.0/24

Фільтрація трафіку, відправленого з певного діапазону IP-адрес. Якщо потрібно провести фільтрацію трафіку, джерелом якого є підмережа, то можна використати такий фільтр:

ip.src == 192.168.1.0/24

Фільтрація трафіку, призначеного для відправки на певний діапазон IP-адрес. Якщо потрібно провести фільтрацію трафіку, пунктом призначення якого є підмережа, то можна використати такий фільтр:

ip.dst == 192.168.1.0/24

Фільтрація IPv6 протоколу

Показати трафік IPv6 (Internet Protocol шостої версії):

ipv6

Фільтрація по IPv6-адресі. Для фільтрації по IPv6-адресі використовується фільтр:

ipv6.addr == 2604:a880:800:c1::2ae:d001

Фільтрація підмереж і діапазонів адрес IPv6

Замість однієї IPv6-адреси можна вказати підмережу:

ipv6.addr == 2604:a880:800:c1::2ae:d000/64

Фільтрація трафіку, джерелом якого є певна IPv6-адреса:

ipv6.src == 2604:a880:800:c1::2ae:d001

Фільтрація трафіку, відправленого на певну IPv6-адресу:

ipv6.dst == 2604:a880:800:c1::2ae:d001

Фільтрація трафіку, відправленого з певного діапазону адрес IPv6. Якщо потрібно фільтрувати трафік, джерелом якого є підмережа:

ipv6.src == 2604:a880:800:c1::2ae:d000/64

Фільтрація трафіку, призначеного для відправки на певний діапазон адрес IPv6. Якщо потрібно фільтрувати трафік, пунктом призначення якого є підмережа:

ipv6.dst == 2604:a880:800:c1::2ae:d000/64

Фільтрація ICMPv6 (Internet Control Message Protocol — протокол міжмережевих керуючих повідомлень шостої версії) в Wireshark виконується фільтром:

icmpv6

Фільтр трафіку протоколів транспортного рівня

Щоб побачити тільки трафік TCP:

tcp

Показати трафік, джерелом або портом призначення якого є певний порт, наприклад 8080:

tcp.port==8080

Показати трафік, джерелом якого є порт 80:

tcp.srcport == 80

Показати трафік, який відправляється службі, яка прослуховує порт 80:

tcp.dstport == 80

Показати TCP-пакети з прапорцем SYN:

tcp.flags.syn==1

Показати TCP пакети з прапорцем SYN і без прапорця ACK:

tcp.flags.syn==1 && tcp.flags.ack==0

Аналогічно і для інших прапорців:

SYN

tcp.flags.syn==1

ACK

tcp.flags.ack==1

RST

tcp.flags.reset==1

FIN

tcp.flags.fin==1

CWR

tcp.flags.cwr

ECN

tcp.flags.ecn

URG

tcp.flags.urg==1

PSH

tcp.flags.push==1

Також можна використоувати синтаксис виду tcp.flags == 0x0XX, наприклад:

- **FIN** це **tcp.flags == 0x001**
- **SYN** це **tcp.flags == 0x002**
- **RST** це **tcp.flags == 0x004**
- **ACK** це **tcp.flags == 0x010**
- Встановлені одночасно **ACK і FIN** це **tcp.flags == 0x011**
- Встановлені одночасно **ACK і SYN** це **tcp.flags == 0x012**
- Встановлені одночасно **ACK и RST** це **tcp.flags == 0x014**

Фільтрувати потік TCP з номером X:

tcp.stream eq X

Фільтрувати по номеру потоку:

tcp.seq == x

Показати повторні відправки пакетів. Допомагає відслідковувати зменшення продуктивності додатків і втрату пакетів:

tcp.analysis.retransmission

Фільтр, який покаже проблемні пакети (втрачені сегменти, повторну відправку та інші).

tcp.analysis.flags

Фільтри для оцінки якості мережевого з’єднання

А тепер розглянемо такі характеристики фрейму, яких немає у заголовку фрейму. Вони надаються фрейму програмою Wireshark в результаті аналізу. Ці характеристики стосуються TCP-фреймів.

Фільтр виводить інформацію про фрейми з прапорцем ACK, які є дублями. Велика кількість таких фреймів може свідчити про проблеми зв’язку:

tcp.analysis.duplicate_ack_num == 1

Фільтр показує фрейми для яких не захоплений попередній сегмент:

tcp.analysis.ack_lost_segment

Це нормально на початку захоплення – оскільки інформація перехоплюється не з самого початку сесії.

Для перехоплення фреймів, які з'являються в результаті ретрансмісії (відправляються повторно):

tcp.analysis.retransmission

Вивід фреймів, які отримані в невірному порядку:

tcp.analysis.out_of_order

Щоб побачити тільки трафік UDP:

udp

Показати трафік, джерелом якого є порт 53:

udp.srcport == 53

Показати трафік, який відправляється службі, яка прослуховує порт 53:

udp.dstport == 53

Показати UDP пакет, в якому зустрічається певний рядок, наприклад, рядок *hackware*:

udp contains hackware

Показати трафік ICMP:

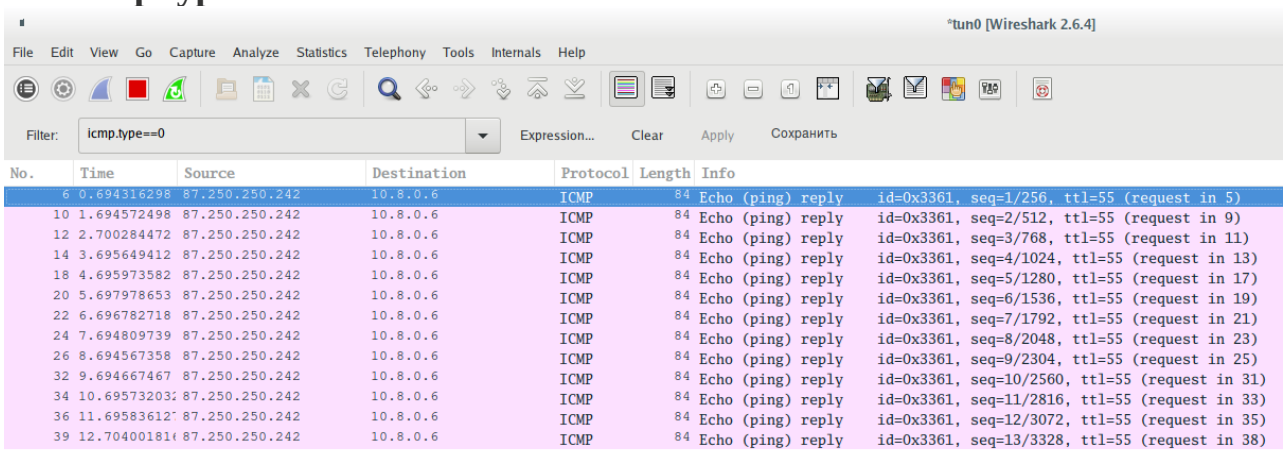
icmp

Показати тільки трафік ICMP v6 (шостої версії)

icmpv6

Показати всі відповіді на *ping*:

icmp.type==0



Показати всі ping-запити:

icmp.type==8

Показати всі помилки недосяжності/заборони хостів і портів:

icmp.type==3

Показати всі спроби перенаправити маршрут з використанням ICMP:

icmp.type==8

Фільтр трафіку протоколів прикладного рівня

Для протоколів прикладного рівня HTTP, DNS, SSH, FTP, SMTP, RDP, SNMP, RTSP, GQUIC, CDP, LLMNR, SSDP є фільтри, які називаються так само як і протоколи, але пишуться маленькими літерами.

Наприклад, щоб побачити HTTP трафік:

http

Щоб побачити трафік нового протоколу HTTP/2**:**

http2

При прийнятті рішення, якому протоколу належать дані, що передаються, програма аналізує номер порту, що використовується. Якщо використовується нестандартный порт, то програма не зможе знайти потрібні дані. Наприклад, якщо було виконано підключення до SSH по порту 1234, то фільтр ssh не знайде SSH трафік.

Фільтр, який показує тільки дані, які передані методом POST:

http.request.method == "POST"

Фільтр, який показує тільки дані, передані методом GET:

http.request.method == "GET"

Пошук запитів до конкретного сайту (хосту):

http.host == "<URL>"

Пошук запитів до конкретного сайту по частині імені:

http.host contains "тут.частина.імені"

Фільтр для виводу HTTP запитів, в яких передавались cookie :

http. cookie

Запити, в яких сервер встановив cookie в браузер користувача:

http.set_cookie

Для пошуку будь-яких переданих зображень:

http.content_type contains "image"

Для пошуку конкретних видів зображень:

http.content_type contains "gif"
http.content_type contains "jpeg"
http.content_type contains "png"

Для пошуку файлів певного типу:

http.content_type contains "text"
http.content_type contains "xml"
http.content_type contains "html"
http.content_type contains "json"
http.content_type contains "javascript"
http.content_type contains "x-www-form-urlencode"
http.content_type contains "compressed"
http.content_type contains "application"

Пошук в Wireshark запитів на отримання файлів певного типу. Наприклад, для пошуку переданих ZIP архівів:

http.request.uri contains "zip"

Пошук файлів в HTTP потоці:

http.file_data

Щоб побачити, які HTTP дані отримані із затримкою, використовується такий фільтр:

http.time>1

Він покаже трафік, отриманий пізніше ніж через 1 секунду.

Щоб побачити всі DNS-запити і відповіді**:**

dns

Щоб побачити, які DNS-запити перевищили час виконання:

dns.time>1

Будуть показані відповіді, які надійшли більш ніж через секунду після відправки запиту.

Цей фільтр показує, які dns запити не можуть бути правильно перетворені:

dns.flags.rcode != 0

Показати тільки DNS запити:

dns.flags.response == 0

Показати тільки DNS відповіді:

dns.flags.response == 1

Показати запити і відповіді на них, в яких відбувається пошук IP для google.com:

dns.qry.name == "google.com"

Показати DNS-запити і відповіді щодо запису A:

dns.qry.type == 1

Показати DNS-запити і відповіді щодо запису AAAA:

dns.qry.type == 28

Показати відповіді, в яких для запису A відправлена IP-адреса 216.58.196.3:

dns.a == 216.58.196.3

Показати відповіді, в яких для запису AAAA відправлена IP-адресан 2a01:4f8:172:1d86::1:

dns.aaaa == 2a01:4f8:172:1d86::1

Показати записи с CNAME apollo.archlinux.org:

dns.cname == "apollo.archlinux.org"

Показати відповіді довжиною більше 30:

dns.resp.len > 30

Показати запити довжиною більше 25:

dns.qry.name.len >25

Показати відповіді DNS-серверів на яких доступна рекурсія:

dns.flags.recavail == 1

Показати відповіді DNS-серверів на яких не доступна рекурсія:

dns.flags.recavail == 0

Чи бажана рекурсія (якщо запитуваний DNS-сервер не має інформації про ім’я хосту, чи має він опитувати інші DNS-сервери в пошуку цієї інформації):

dns.flags.recdesired == 1

Якщо в запиті стоїть 1, значить рекурсія потрібна, якщо 0 — значить вона небажана.

Чи приймати неаутентифіковані дані (0 означає не приймати, 1 означає приймати):

dns.flags.checkdisable == 0

Щоб побачити, як призначаються IP-адреси по протоколу DHCP:

udp.dstport==67 або **bootp.option.dhcp**

Щоб показати DHCP запити:

bootp.option.dhcp == 3

Щоб показати DHCP Discover:

bootp.option.dhcp == 1

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Комп'ютерні мережі: підручник / Азаров О.Д., Захарченко С.М., Кадук О.В., Орлова М.М., Тарасенко В.П. – Вінниця: ВНТУ. – 2020. – 378 с.
2. Комп'ютерні мережі: навчальний посібник / Азаров О.Д., Захарченко С.М., Кадук О.В., Орлова М.М., Тарасенко В.П.. – Вінниця: ВНТУ. – 2013. – 371 с.
3. Таненбаум Э., Уэзерпол Д. Компьютерные сети. – СПб.: Питер. – 2012. – 960 с.
4. В.Г.Олифер, Н.А. Олифер. Компьютерные сети. Принципы, технологии, протоколы. Учебник. ПИТЕР. – 2020. – 467 с.
5. Педагогічний програмний засіб (ППЗ) «Телекомунікаційні системи та мережі. Структура й основні функції. Том 1.
6. В. А. Ганжа, В. В. Шиманский. Компьютерные сети. Минск БГУИР, 2015.
7. Телекомунікаційні та інформаційні мережі : Підручник [для вищих навчальних закладів] / П.П. Воробієнко, Л.А. Нікітюк, П.І. Резніченко. – К.: САММІТ-Книга, 2010. – 708 с.: іл.
8. К. Дуглас. Tcp/ip крупным планом. MirKnig, 2009.
9. Крис Сандерс. Анализ пакетов: практическое руководство по использованию Wireshark и tcpdump для решения реальных проблем в локальных сетях. 3-е издание. Издательский дом Вильямс, 2018.
10. Фильтры Wireshark. https://hackware.ru/?p=7008Фильтры