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

НАВЧАЛЬНИЙ ПОСІБНИК

з дисципліни

“МЕРЕЖНІ ТЕХНОЛОГІЇ”

для студентів
спеціальності 123 - “Комп'ютерна інженерія”

2021

Навчальний посібник з дисципліни “Мережні технології” для студентів спеціальності 123 – “Комп’ютерна інженерія” /Роковий О.П., Коган А.В. , Алєнін О.І. – Київ: КПІ, 2021. – 63 с.

Автори: **Роковий Олександр Петрович**,
кандидат технічних наук,
Коган Алла Вікторівна
кандидат технічних наук,
Алєнін Олег Ігорович

Рецензент: Кулаков Ю. О., проф. кафедри ОТ ФІОТ, д.т.н, професор

Затверджено
вченою радою
ФІОТ

Протокол № 2
від 27.09.2021 р.

Затверджено
на засіданні кафедри
ОТ

Протокол № 3
від 22.09.2021 р.

ЗАГАЛЬНІ ПОЛОЖЕННЯ

Цей навчальний посібник являє собою керівництво до лабораторного практикуму з курсу «Мережні технології» для вивчення та практичного освоєння студентами напряму підготовки спеціальності 123 - “Комп'ютерна інженерія”.

Лабораторні роботи виконуються в операційній системі GNU/Linux.

Мета цього посібника - познайомити студентів з найбільш поширеними сервісами прикладного рівня стеку протоколів TCP/IP.

Перед кожною роботою наведено короткі теоретичні відомості, що містять достатній обсяг інформації для виконання лабораторних робіт .

Оформлення звіту та порядок його подання

Для позитивної оцінки по кожній роботі студент надає викладачу оформлений звіт.

Звіт має містити:

- титульний аркуш (на ньому вказують назву міністерства, назву університету, назву кафедри, тему роботи, виконавця та особу, що приймає звіт, рік);
- мету, варіант (якщо є необхідність) і завдання роботи;
- короткий огляд теоретичних відомостей;
- змістовний аналіз отриманих результатів та висновки.

Звіт виконують на білому папері формату А4 (210 х 297 мм) та подаються викладачеві у ***електронному форматі***. Текст розміщують тільки з однієї сторони листа. Поля сторінки з усіх боків – 20 мм. Для набору тексту звіту використовують шрифт Times New Roman, 12 пунктів. Міжрядковий інтервал: полуторний – для тексту звіту, одинарний – для лістингів програм, таблиць і роздруківок даних.

Під час захисту роботи студент повинний продемонструвати знання по змісту роботи, по теоретичному матеріалу, аналізувати кожен етап роботи, виконувати завдання сумісні з роботою, що викладач може попросити виконати на місці. Студент повинний вміти правильно аналізувати отримані результати.

Зміст

Лабораторна робота № 1

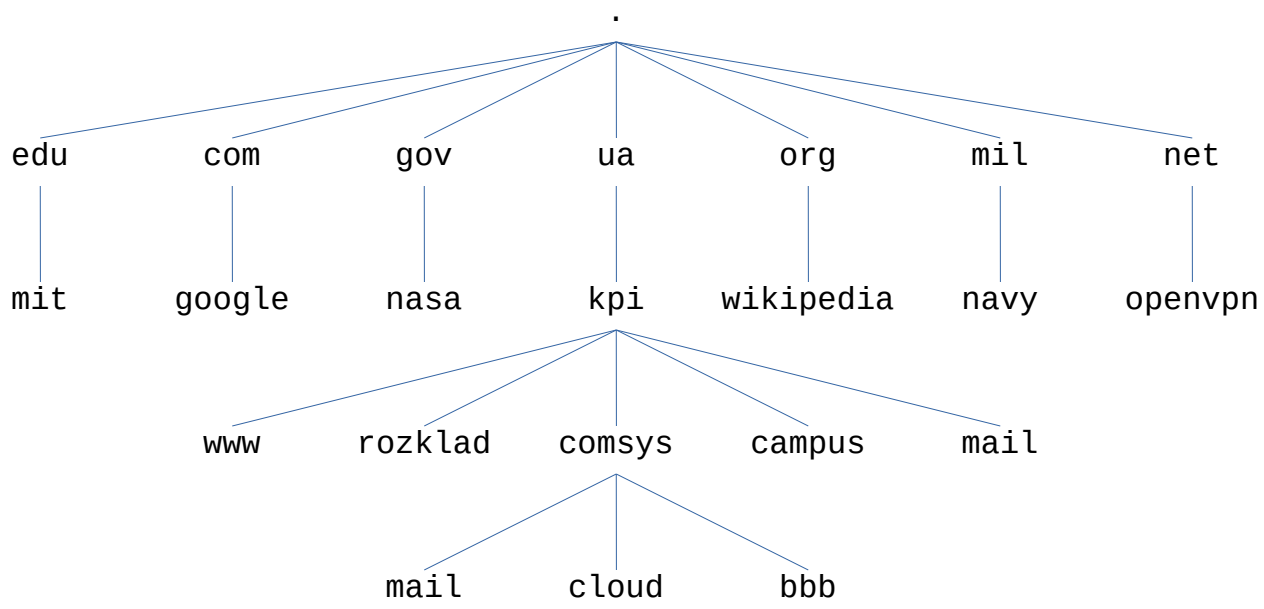
Служба доменних імен (DNS)

1. Короткі теоретичні відомості.

1.1. Ієрархія доменів.

DNS реалізує ієрархічний простір імен для Інтернет-об’єктів. На відміну від імен файлів Unix, які обробляються зліва направо (компоненти імен відокремлюються символом “/”), імена DNS обробляються зправо наліво і використовують символ “.” в якості роздільника. Хоча вони обробляються зправо наліво, користувачі все ще читають доменні імена зліва направо. Прикладом доменного імені для вузла є comsys.kpi.ua.

Доменні імена використовуються для іменування Інтернет-об’єктів. Під цим мається на увазі те, що DNS не використовується строго для перетворення імен вузлів у адреси вузлів. Точніше сказати, що DNS відображає доменні імена у значення. На даний момент ці значення є IP-адресами.



Малюнок 1. Приклад ієрархії доменів.

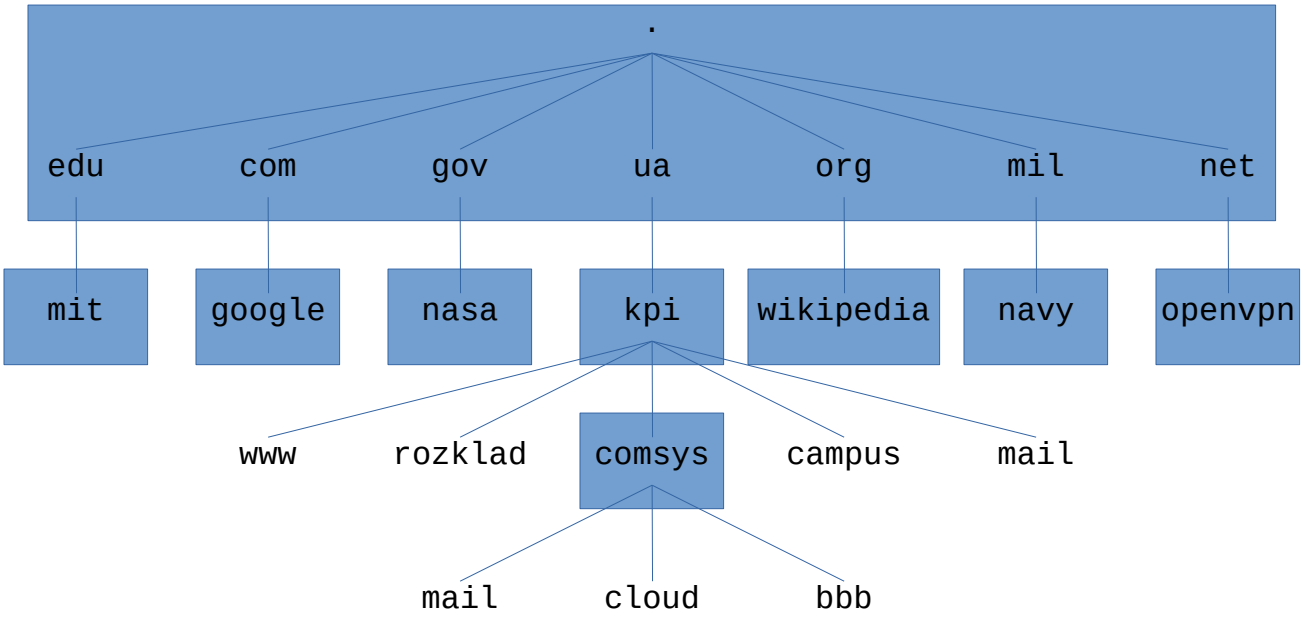
Ієрархія DNS, так само як ієрархія файлів Unix, може бути візуалізована у вигляді дерева, де кожен вузол у дереві відповідає домену, а листя у дереві відповідають іменованим вузлам. На малюнку 1 наведено приклад ієрархії доменів.

Під час розробки ієрархії доменних імен відбулася значна кількість дискусій щодо того, які конвенції регулюватимуть імена, які повинні видаватися вгорі ієрархії. На першому рівні ієрархія не дуже широка. Для кожної країни світу виділене одне доменне ім'я, а також “велика

шістка” доменів: .edu, .com, .gov, .mil, .org та .net. Усі ці шість доменів спочатку базувались у США, наприклад: лише акредитовані в США навчальні заклади можуть зареєструвати доменне ім’я .edu. За останні роки кількість доменів верхнього рівня була розширена для вирішення великого попиту на імена доменів .com. До нових доменів верхнього рівня належать .biz, .coop, .info... Зараз існує понад 1200 доменів верхнього рівня.

1.2. Сервери імен.

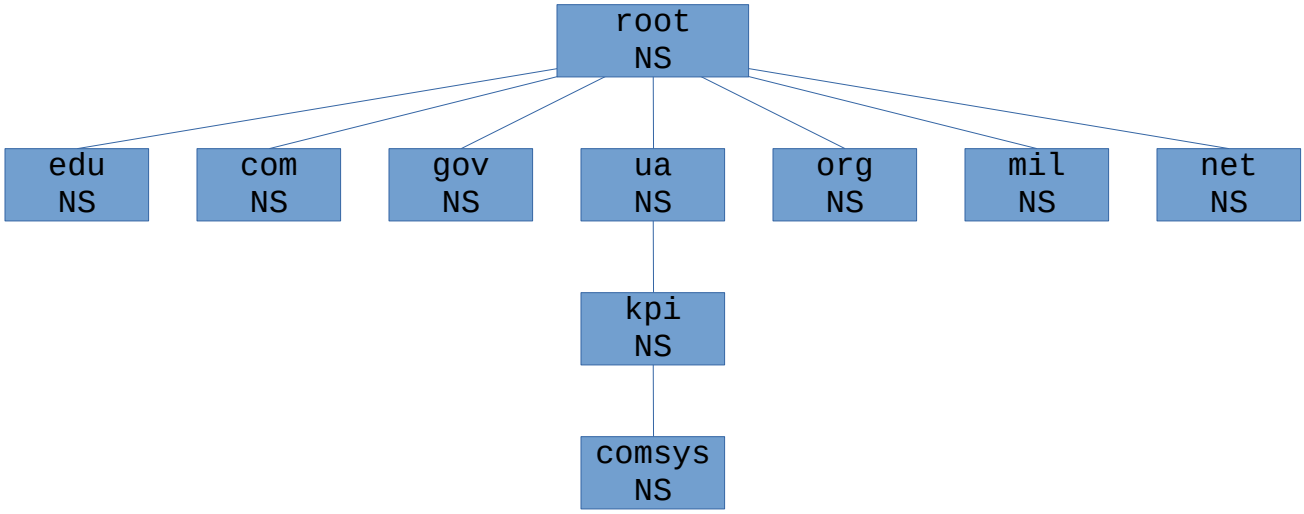
Повна ієрархія доменних імен існує лише абстрактно. Першим кроком є розділення ієрархії на піддерева, які називаються зонами. На малюнку 2 показано, як ієрархію, подану на малюнку 1, можна розділити на зони. Кожна зона може розглядатися як відповідна якомусь адміністративному органу, який відповідає за цю частину ієрархії. Наприклад, верхній рівень ієрархії утворює зону, якою керує Internet Corporation for Assigned Names and Numbers (ICANN). У центрі на малюнку знаходиться зона, яка відповідає Національному технічному університету України “Київський політехнічний інститут імені Ігоря Сікорського”. У межах цієї зони деякі кафедри не хочуть керувати власною частиною ієрархії (і тому вони залишаються в зоні університетського рівня), тоді як інші, як кафедра обчислювальної техніки, керує власною зоною.



Малюнок 2. Ієрархія доменів, розділена на зони.

Важливість зони полягає в тому, що вона відповідає основній одиниці реалізації в DNS - серверу імен. Зокрема, інформація, що міститься в кожній зоні, реалізована на двох або більше серверах імен. Кожен сервер імен, у свою чергу, є програмою, до якої можна отримати доступ через мережу Інтернет. Клієнти надсилають запити на сервери імен, а сервери імен їм відповідають. Іноді відповідь містить повну інформацію, яку бажає клієнт, а іноді відповідь містить вказівник на інший сервер, якому клієнт повинен відправити запит. Таким чином, з точки зору реалізації, точніше вважати, що DNS представлений ієрархією серверів імен, а не ієрархією доменів, як показано на малюнку 3.

Кожна зона реалізована на двох або більше серверах імен для забезпечення відмовостійкості. Тобто інформація з домену буде доступна, навіть якщо один сервер імен виходить з ладу. З іншого боку, сервер імен може обслуговувати більше однієї зони.



Малюнок 3. Ієрархія серверів імен.

Кожен сервер імен містить інформацію про зону як сукупність записів ресурсів. По суті, запис ресурсу - це прив’язка імені до значення, яке складається з п'яти полів:

```
(Name, Value, Type, Class, TTL)
```

Поля Name та Value – містять інформацію, яка цікавить кінцевих користувачів, тоді як поле Type вказує, як значення слід інтерпретувати. Наприклад, Type = A вказує, що Value є IP-адресою. Таким чином, записи A реалізують відображення імені та адреси. Інші типи записів включають:

- `NS` – поле Value містить доменне ім'я для вузла, на якому запущений сервер імен, який знає, як перетворювати імена в межах зазначеного домену;
- `CNAME` – поле Value містить канонічну назву для конкретного вузла; воно використовується для визначення псевдонімів;
- `MX` – поле Value містить доменне ім'я вузла, на якому знаходиться поштовий сервер, який оброблює повідомлення для вказаного домену.

Поле Class було включено, щоб дозволити сутностям, крім NIC, визначати корисні типи записів. На сьогоднішній день єдиним широко використовуваним Class є Internet, його позначають IN. Поле “Час життя” (TTL) показує, як довго цей ресурсний запис є дійсним. Він використовується серверами, які кешують записи ресурсів з інших серверів. Коли час життя TTL закінчується, сервер повинен вилучити запис із кешу.

Приклад.

Кореневий сервер імен містить запис NS для кожного сервера імен домену верхнього рівня (TLD). Це визначає сервер, який може відповідати на запити для цієї частини ієрархії DNS (.ua

та .com у прикладі). Він також має записи A, які перетворюють ці імена у відповідні IP-адреси. У сукупності ці два записи ефективно реалізують покажчик з кореневого сервера імен на один із серверів TLD.

```
(ua, in1.ns.ua, NS, IN)
(in1.ns.ua, 74.123.224.40, A, IN)
(com, a.gtld-servers.net, NS, IN)
(a.gtld-servers.net, 192.5.6.30, A, IN)
...
```

Просуваючись вниз по ієрархії на один рівень, сервер має записи для таких доменів, як цей:

```
(kpi.ua, ns.kpi.ua, NS, IN)
(ns.kpi.ua, 77.47.128.130, A, IN)
...
```

Наведені запис NS та запис A для сервера імен, який відповідає за частину ієрархії kpi.ua. Цей сервер може мати можливість безпосередньо відповідати на деякі запити (наприклад, comsys.kpi.ua), тоді як він перенаправлятиме інші запити на сервер ще на один рівень в ієрархії нижче (наприклад, для запиту про cloud.comsys.kpi.ua).

```
(cloud.comsys.kpi.ua, 10.18.49.190, A, IN)
(comsys.kpi.ua, ns.comsys.kpi.ua, NS, IN)
(ns.comsys.kpi.ua, 10.18.49.2, A, IN)
...
```

Нарешті, сервер імен третього рівня, який керує доменом comsys.kpi.ua, містить записи A для всіх своїх вузлів. Він також може визначити набір псевдонімів (записів CNAME) для кожного з цих вузлів. Псевдоніми іноді є просто зручними (наприклад, коротшими) назвами комп'ютерів. Наприклад, www.comsys.kpi.ua є псевдонімом для вузла з іменем comsys.kpi.ua. Це дозволяє перенести веб-сайту на інший сервер, не змінюючи налаштування у користувачів; вони просто продовжують використовувати псевдонім, не зважаючи на те, на якому сервері зараз знаходиться веб-сайт домену. Записи обміну поштою (MX) служать тій самій меті для програм електронної пошти - вони дозволяють адміністратору змінювати вузол, який отримує пошту домену, не змінюючи електронної адреси.

```
(www.comsys.kpi.ua, comsys.kpi.ua, CNAME, IN)
(comsys.kpi.ua, 77.47.192.42, A, IN)
(comsys.kpi.ua, mail.comsys.kpi.ua, MX, IN)
(mail.comsys.kpi.ua, 77.47.193.180, A, IN)
...
```

Система імен DNS зазвичай використовується для іменування вузлів (включаючи сервери) та сайтів, хоча записи ресурсів можна визначити практично для будь-якого типу об’єкта. DNS не використовується для іменування окремих людей або інших об’єктів, таких як файли або каталоги; інші системи імен зазвичай використовуються для ідентифікації таких об'єктів. Наприклад, X.500 - це система імен ISO, призначена для спрощення ідентифікації людей. Це дозволяє назвати особу, вказавши набір атрибутів: ім’я, прізвище, номер телефону, поштову адресу тощо. X.500 виявився занадто громіздким - і, в якомусь сенсі, був узурпований потужними пошуковими системами, які тепер доступні в Інтернеті, - але з часом він перетворився на Легкий протокол доступу до каталогів (LDAP). LDAP - це підмножина X.500, спочатку розроблена як інтерфейс для персональних комп'ютерів до X.500. Сьогодні широко використовується, переважно на рівні підприємств, як система збереження інформації про користувачів.

1.3. Перетворення імені.

Враховуючи ієрархію серверів імен, необхідно розглянути питання про те, як клієнт використовує ці сервери для перетворення доменного імені в IP-адресу. Для ілюстрації основної ідеї, припустимо, клієнт хоче перетворити ім'я comsys.kpi.ua. Клієнт міг спочатку надіслати запит, що містить це ім’я, на один із кореневих серверів (як ми побачимо нижче, це рідко трапляється на практиці, але на сьогодні достатньо для ілюстрації базової операції). Кореневий сервер, який не може збігатися з цілим іменем, повертає найкращий збіг, який у нього є - запис NS для ua, який вказує на сервер TLD in1.ns.ua. Сервер також повертає всі записи, пов’язані з цим записом, у цьому випадку запис A для in1.ns.ua. Клієнт, не отримавши відповіді, далі відправляє той самий запит на сервер імен на IP-адресу 74.123.224.40. Цей сервер також не може збігатися з цілим ім'ям, тому повертає NS та відповідні записи A для домену kpi.ua. Ще раз клієнт надсилає той самий запит, що й раніше, на сервер на вузлі з IP 77.47.128.130. Цього разу досягнуто сервера, який може повністю обробити запит. Остаточний запит до сервера з адресою 77.47.128.130 видає запис A для comsys.kpi.ua, і клієнт дізнається, що відповідною IP-адресою є 77.47.192.42.

Цей приклад все ще залишає без відповіді кілька питань щодо процесу перетворення. Перше питання полягає в тому, як спочатку клієнт знайшов кореневий сервер, або, по-іншому, як він шукає ім’я сервера, який знає, як перетворити потрібне ім'я? Це фундаментальна проблема будь-якої системи іменування, і відповідь полягає в тому, що систему потрібно якось завантажити. У цьому випадку відображення імені та адреси для одного або декількох кореневих серверів добре відомо; тобто він публікується якимось чином поза самою системою імен.

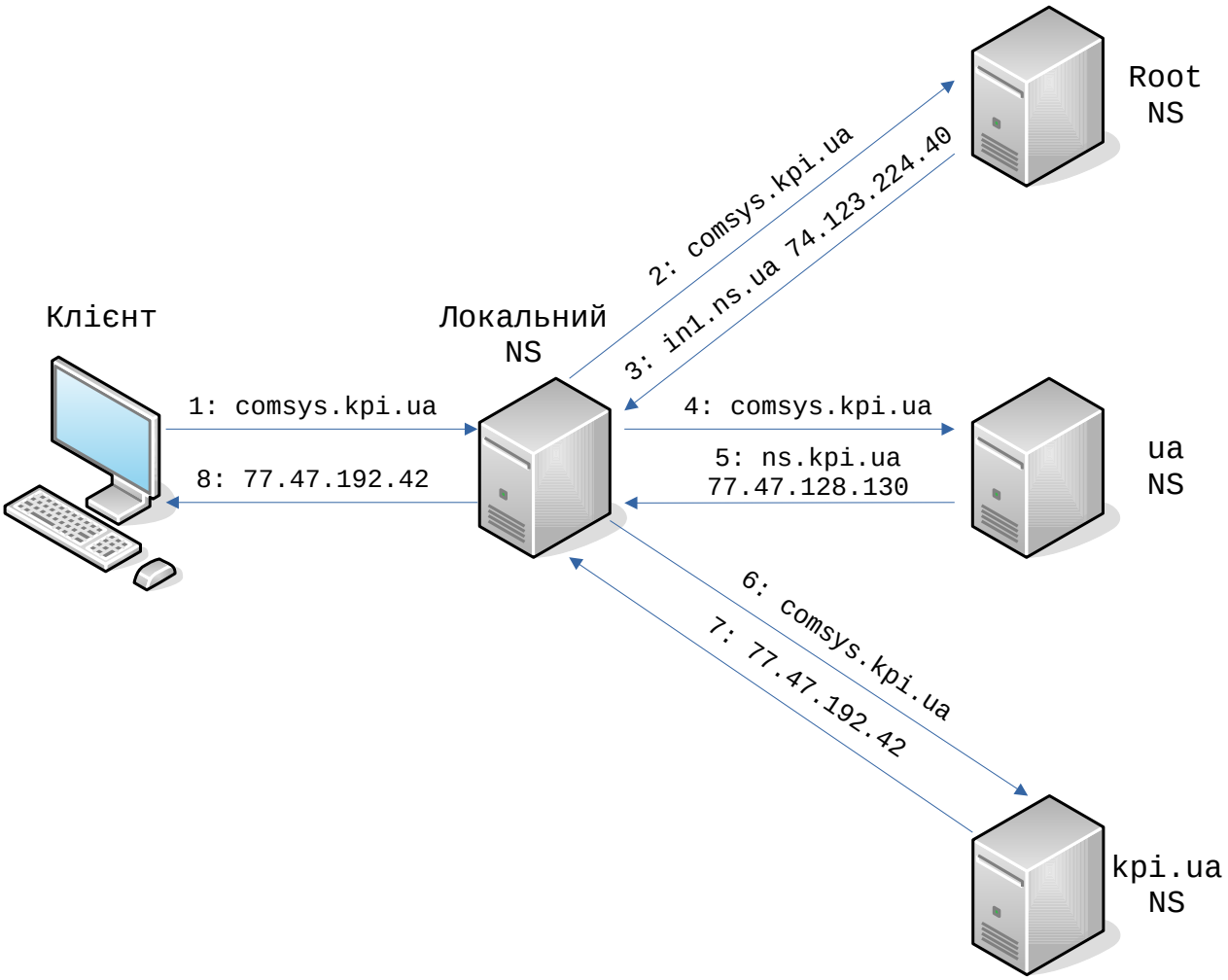
Однак на практиці не всі клієнти знають про кореневі сервери. Натомість клієнтська програма, що працює на кожному вузлі Інтернету, ініціалізується адресою локального сервера імен. Наприклад, усі вузли кафедри обчислювальної техніки знають про сервер ns.comsys.kpi.ua. Цей локальний сервер імен, у свою чергу, має записи ресурсів для одного або декількох кореневих серверів, наприклад:

```
('root', a.root-servers.net, NS, IN)
(a.root-servers.net, 198.41.0.4, A, IN)
```

Таким чином, перетворення імені насправді передбачає запит клієнта на локальний сервер, який, у свою чергу, діє як клієнт, який запитує віддалені сервери від імені клієнта. Це призводить до взаємодії клієнт/сервер, проілюстровану на малюнку 4. Однією з переваг цієї

моделі є те, що всі вузли в Інтернеті не повинні постійно оновлюватися. Де розташовані поточні кореневі сервери повинні знати лише DNS-сервери. Друга перевага полягає в тому, що локальний сервер бачить відповіді, які повертаються із запитів, відправлених усіма локальними клієнтами. Локальний сервер кешує ці відповіді і іноді може перетворювати майбутні запити, не виходячи з мережі. Поле TTL у записах ресурсів, що повертаються віддаленими серверами, вказує, як довго кожен запис можна безпечно кешувати. Цей механізм кешування може бути використаний і далі вгору по ієрархії, зменшуючи навантаження на кореневий і TLD-сервери.

Друге питання полягає в тому, як працює система, коли користувач подає часткове ім’я (наприклад, comsys), а не повне доменне ім’я (наприклад, comsys.kpi.ua). Відповідь полягає в тому, що клієнтська програма налаштована на локальний домен, в якому знаходиться вузол (наприклад, kpi.ua), і вона додає цей рядок до будь-яких простих імен перед надсиланням запиту.



Малюнок 4. Перетворення імен на практиці.

1.4. Доменна термінологія.

1.4.1. Система доменних імен.

Система доменних імен, більш відома як "DNS" - це мережна система, яка дозволяє перетворити зручні для людини (символьні) імена в унікальні IP-адреси.

1.4.2. Доменне ім'я.

Доменне ім’я - це дружнє для людини ім’я, яке ми звикли пов’язувати з Інтернет-ресурсом. Наприклад, "google.com" - це доменне ім'я. Деякі люди скажуть, що частина "google" - це домен, але загалом ми можемо називати комбіновану форму іменем домену.

URL-адреса “google.com” пов’язана із серверами, що належать Google Inc. Система доменних імен дозволяє нам отримувати доступ до серверів Google, коли ми вводимо “google.com” у веббраузері.

1.4.3. IP-адреса.

IP-адреса - це те, що ми називаємо адресою мережі. Кожна IP-адреса повинна бути унікальною в своїй мережі. Коли ми говоримо про вебсайти, ця мережа - це весь Інтернет.

IPv4, найпоширеніша форма адрес, записується як чотири набори чисел, кожен набір має до трьох цифр, причому кожен набір відокремлений крапкою. Наприклад, “111.222.111.222” може бути дійсною IP-адресою IPv4. За допомогою DNS ми прив'язуємо ім'я до цієї адреси, щоб вам не потрібно було запам'ятовувати складний набір чисел для кожного місця, яке ви хочете відвідати в мережі.

1.4.4. Домен верхнього рівня.

Домен верхнього рівня (TLD) є найбільш загальною частиною домену. Домен верхнього рівня - це найвіддаленіша частина праворуч (відокремлена крапкою). Загальними доменами верхнього рівня є "com", "net", "org", "gov", "edu" та "io".

Домени верхнього рівня знаходяться на вершині ієрархії з точки зору доменних імен. ICANN (Інтернет-корпорація з присвоєними іменами та номерами) певним сторонам надає управлінський контроль над доменами верхнього рівня. Потім ці сторони можуть розповсюджувати доменні імена згідно TLD, як правило, через реєстратора доменів.

1.4.5. Вузли.

В межах домену власник домену може визначити окремі вузли, які посилаються на окремі комп’ютери або служби, доступні через домен. Наприклад, більшість власників доменів роблять свої веб-сервери доступними через відкритий домен (example.com), а також через визначення “вузла” “www” (www.example.com).

Ви можете мати інші визначення вузлів під загальним доменом. Ви можете отримати доступ до API через вузол “api” (api.example.com), або можете отримати доступ до ftp, визначивши вузол, який називається “ftp” або “files” (ftp.example.com або files.example.com). Імена вузлів можуть бути довільними, якщо вони унікальні для домену.

1.4.6. Піддомен.

Предметом, що стосується вузлів, є піддомени.

DNS працює в ієрархії. Домени верхнього рівня можуть мати багато доменів. Наприклад, "com" TLD має як "google.com", так і "ubuntu.com" під ним. "піддомен" означає будь-який домен, який є частиною більшого домену. У цьому випадку ”ubuntu.com” можна сказати як піддомен “com”. Зазвичай це просто називається доменом, або частина “ubuntu” називається SLD, що означає домен другого рівня.

Так само кожен домен може контролювати "піддомени", які знаходяться під ним. Зазвичай це те, що ми маємо на увазі під піддоменами. Наприклад, може бути піддомен для історичного відділу школи за адресою “www.history.school.edu”. Частина "history" є піддоменом.

Різниця між іменем вузла та піддоменом полягає в тому, що вузол визначає комп’ютер або ресурс, тоді як піддомен розширює батьківський домен. Це метод поділу самого домену.

Незалежно від того, чи йдеться про піддомени чи вузли, можна побачити, що самі ліві частини домену є найбільш конкретними. Ось як працює DNS: від найбільш до найменш конкретного, коли читаєте зліва направо.

1.4.7. Повне доменне ім'я.

Повне доменне ім’я, яке часто називають FQDN, - це те, що ми називаємо абсолютним доменним ім’ям. Домени в системі DNS можуть бути надані відносно один одного, і як такі, можуть бути дещо неоднозначними. Ідентифікаційне доменне ім’я (FQDN) - це абсолютне ім’я, яке вказує його розташування щодо абсолютного кореня системи доменних імен.

Це означає, що він визначає кожен батьківський домен, включаючи TLD. Правильне повне доменне ім’я закінчується крапкою, що вказує на корінь ієрархії DNS. Прикладом повного доменного імені є "mail.comsys.kpi.ua.". Іноді програмне забезпечення, яке вимагає повного доменного імені, не вимагає кінцевої крапки, але кінцева крапка потрібна, щоб відповідати стандартам ICANN.

1.4.8. Сервер імен.

Сервер імен - це комп'ютер, призначений для перетворення доменних імен в IP-адреси. Ці сервери виконують більшу частину роботи в системі DNS. Оскільки загальна кількість перетворень доменів є занадто великою для будь-якого одного сервера, кожен сервер може перенаправити запит на інші сервери імен або делегувати відповідальність за підмножину піддоменів, за які вони відповідають.

Сервери імен можуть бути "авторитетними", тобто вони дають відповіді на запити про домени, що знаходяться під їх контролем. В іншому випадку вони можуть вказувати на інші сервери або подавати кешовані копії даних інших серверів імен.

1.4.9. Файл зони.

Файл зони - це простий текстовий файл, який містить зіставлення між іменами доменів та IP-адресами. Ось як система DNS нарешті з’ясовує, з якою IP-адресою слід зв’язатися, коли користувач запитує певне доменне ім’я.

Файли зон знаходяться на серверах імен і, як правило, визначають ресурси, доступні під певним доменом, або місце, куди можна звернутись, щоб отримати цю інформацію.

1.4.10. Записи.

У файлі зони зберігаються записи. У своїй найпростішій формі запис в основному являє собою єдине відображення між ресурсом та іменем. Вони можуть зіставити доменне ім’я з IP-адресою, визначити сервери імен для домену, визначити поштові сервери для домену тощо.

1.5. Як працює DNS?

1.5.1. Кореневі сервери.

Як ми вже говорили вище, DNS за своєю суттю є ієрархічною системою. На вершині цієї системи знаходиться так звані «кореневі сервери». Ці сервери контролюються різними організаціями та делегуються повноваженнями ICANN (Інтернет-корпорація з присвоєння імен та номерів).

В даний час працює 13 кореневих серверів. Однак, оскільки існує неймовірна кількість імен, які потрібно перетворювати щохвилини, кожен із цих серверів насправді відображається дзеркально. Цікавим у цій установці є те, що кожне дзеркало для одного кореневого сервера має

однакову IP-адресу. Коли надсилаються запити на певний кореневий сервер, запит буде перенаправлено до найближчого дзеркала цього кореневого сервера.

Що роблять ці кореневі сервери? Кореневі сервери обробляють запити на інформацію про домени верхнього рівня. Отже, якщо надходить запит на щось, що сервер імен нижчого рівня не може вирішити, запит надходить до кореневого сервера домену.

Кореневі сервери насправді не знатимуть, де розміщений домен. Однак вони зможуть направити запитувача на сервери імен, які обробляють спеціально запитаний домен верхнього рівня.

Отже, якщо на кореневий сервер подається запит на “www.wikipedia.org”, кореневий сервер не знайде результат у своїх записах. Він перевірить свої файли зон на наявність списку, який відповідає “www.wikipedia.org”. Не знайде.

Натомість він знайде запис для домену “org” верхнього рівня і дасть клієнту, який запитував, адресу сервера імен, відповідального за “org” адреси.

1.5.2. Сервери TLD.

Потім запитувач надсилає новий запит на IP-адресу (надану йому кореневим сервером), яка відповідає за домен верхнього рівня.

Отже, щоб продовжити наш приклад, він надішле запит на сервер імен, відповідальний за знання про “org” домени, щоб перевірити, чи знає він, де знаходиться “www.wikipedia.org”.

Ще раз запитувач шукатиме “www.wikipdia.org” у своїх файлах зони. Він не знайде цей запис у своїх файлах.

Однак він знайде запис із переліком IP-адреси сервера імен, відповідального за “wikipedia.org”. Це наближає клієнта до потрібної відповіді.

1.5.3. Сервери імен на рівні домену.

На даний момент запитувач має IP-адресу сервера імен, який відповідає за знання фактичної IP-адреси ресурсу. Він надсилає новий запит на сервер імен, ще раз запитуючи, чи може він перетворити “www.wikipedia.org”.

Сервер імен перевіряє свої файли зон і виявляє, що у нього є файл зони, пов'язаний з “wikipedia.org”. Усередині цього файлу є запис для вузла “www”. Цей запис повідомляє IP-адресу, де знаходиться цей вузол. Сервер імен повертає остаточну відповідь запитувачу.

1.5.4. Що таке сервер перетворення імен?

У наведеному вище сценарії ми називали "запитувача".

Майже у всіх випадках запитувачем буде вузол, який ми називаємо "сервером перетворення імен". Сервер перетворення імен налаштований на запитання інших серверів. В основному це посередник для користувача, який кешує попередні результати запитів для збільшення швидкості та знає адреси кореневих серверів, щоб мати змогу перетворити запити, зроблені для вузлів, про які він ще не знає.

В основному, користувач, як правило, має кілька серверів перетворення імен, налаштованих у своїй комп'ютерній системі. Сервери перетворення імен зазвичай надаються провайдером або іншими організаціями. Наприклад, Google пропонує свої сервери перетворення імен, які ви можете запитувати. Вони можуть бути налаштовані на вашому комп'ютері автоматично або вручну.

Коли ви вводите URL-адресу в адресний рядок веббраузера, комп’ютер спочатку перевіряє, чи зможе він локально з’ясувати, де знаходиться ресурс. Він перевіряє файл “hosts” на комп’ютері. Потім він надсилає запит на сервер перетворення імен і чекає відповідь, щоб з'ясувати IP-адресу ресурсу.

Потім сервер перетворення імен перевіряє свій кеш на відповідь. Якщо він не знаходить його, він виконує описані вище дії.

1.6. Файли зони.

Файли зон - це спосіб, яким сервери імен зберігають інформацію про домени, про які вони знають. Кожен домен, про який знає сервер імен, зберігається у файлі зони. Більшість запитів, що надходять на середній сервер імен, не є тим, для чого сервер буде мати файли зон.

Якщо він налаштований на обробку рекурсивних запитів, таких як сервер перетворення імен, він знайде відповідь і поверне її. В іншому випадку він повідомить запитуючу сторону, де шукати далі.

Чим більше файлів зон має сервер імен, тим на більше запитів він зможе авторитетно відповісти.

Файл зони описує "зону" DNS, яка в основному є підмножиною всієї системи імен DNS. Зазвичай він використовується для налаштування лише одного домену. Він може містити кілька записів, які визначають, де знаходяться ресурси для відповідного домену.

$ORIGIN зони - параметр, який за замовчуванням дорівнює найвищому рівню повноважень зони.

Отже, якщо для налаштування «example.com» використовується файл зони домену, для $ORIGIN буде встановлено example.com .

Це або налаштовано у верхній частині файлу зони, або його можна визначити у файлі конфігурації сервера DNS, який посилається на файл зони. У будь-якому випадку, цей параметр описує, для чого зона буде повноважною.

Аналогічно, $TTL налаштовує "час життя" інформації, яку він надає. В основному це таймер. Кешуючий сервер імен може використовувати попередньо запрошені результати для відповіді на запитання, поки значення TTL не закінчиться.

1.7. Типи записів.

1.7.1. SOA запис.

Запис авторизації, або SOA, є обов’язковим записом у всіх файлах зони. Це має бути перший явний запис у файлі (хоча вище можуть бути вказані специфікації $ORIGIN або $TTL). Початок авторитетного запису виглядає приблизно так:

```
domain.com.  IN SOA   ns1.domain.com. admin.domain.com. (
                          12083           ; serial number
                          3h              ; refresh interval
                          30m             ; retry interval
                          3w              ; expiry period
                          1h              ; negative TTL
                          )
```

Пояснимо, для чого призначена кожна частина:

- **domain.com.**: Це корінь зони. Вказує, що файл зони призначений для домену domain.com. Часто ви бачите, що це замінено на @, що є просто заповнювачем, який замінює вміст змінної $ORIGIN.

- **IN SOA**: Частина “IN” означає Інтернет (і вона буде присутня у багатьох записах). SOA є показником того, що це запис Start of Authority.
- **ns1.domain.com.**: визначає основний сервер імен для цього домену. Сервери імен можуть бути як основними, так і вторинними, і якщо налаштовано динамічний DNS, один сервер повинен бути “основним”. Якщо ви не налаштували динамічний DNS, то це лише один із ваших основних серверів імен.
- **admin.domain.com.**: Це електронна адреса адміністратора для цієї зони. "@" Замінено крапкою в електронній адресі. Якщо в іменній частині адреси електронної пошти, як правило, є крапка, вона замінюється на "\" в цій частині (your.name@domain.com стає your\name.domain.com).
- **12083**: Це серійний номер файлу зони. Кожного разу, коли ви редагуєте файл зони, ви повинні збільшувати це число, щоб файл зони розповсюджувався правильно. Вторинні сервери перевірять, чи не є серійний номер первинного сервера для зони більшим, ніж той, який вони мають у своїй системі. Якщо це так, вони запитує новий файл зони, якщо ні, продовжують обслуговувати оригінальний файл.
- **3h**: Інтервал оновлення для зони. Це кількість часу, який вторинний сервер буде чекати перед опитуванням первинного для зміни файлу зони.
- **30m**: Це інтервал повторної спроби для цієї зони. Якщо вторинний сервер не може підключитися до основного, коли закінчився період оновлення, він зачекає стільки часу і спробує опитати основний.
- **3w**: Це термін придатності. Якщо вторинний сервер імен не зміг зв’язатися з основним протягом такого періоду часу, він більше не повертає відповіді як авторитетне джерело для цієї зони.
- **1h**: Стільки часу сервер імен буде кешувати помилку імені, якщо він не може знайти запитане ім’я у цьому файлі.

1.7.2. A та AAAA записи.

Обидва ці записи відображають вузол на IP-адресу. Запис "A" використовується для зіставлення вузла з IP-адресою IPv4, тоді як записи "AAAA" використовуються для зіставлення вузла з адресою IPv6.

Загальний формат цих записів такий:

```
host     IN      A       IPv4_address
host     IN      AAAA    IPv6_address
```

Отож, оскільки наш запис SOA вказав основним сервером вузол “ns1.domain.com”, нам необхідно вказати його адресу, оскільки “ns1.domain.com” знаходиться в зоні “domain.com”, що цей файл є визначальним.

Запис може виглядати приблизно так:

```
ns1     IN  A       111.222.111.222
```

Зверніть увагу, що ми не повинні вказувати повне ім’я. Ми можемо просто вказати вузол без повного доменного імені, а сервер DNS заповнить решту значенням $ORIGIN. Однак ми могли б так само легко використовувати повне доменне ім’я:

```
ns1.domain.com.     IN  A       111.222.111.222
```

У більшості випадків тут ви визначаєте свій вебсервер як “www”:

```
www     IN  A       222.222.222.222
```

Ми також повинні сказати, як перетворюється базовий домен. Ми можемо зробити це так:

```
domain.com.     IN  A       222.222.222.222
```

Замість цього ми могли використати “@” для посилання на базовий домен:

```
@       IN  A       222.222.222.222
```

Ми також маємо можливість перетворити все, що в цьому домені не визначено явно для цього сервера. Ми можемо зробити це за допомогою символу "*":

```
*       IN  A       222.222.222.222
```

Усі вони так само добре працюють із записами AAAA для адрес IPv6.

1.7.3. CNAME записи.

Записи CNAME визначають псевдонім канонічного імені сервера (такий, який визначається записом A або AAAA).

Наприклад, можемо мати запис імені A, що визначає вузол “server1”, а потім використовувати “www” як псевдонім для цього вузла:

```
server1     IN  A       111.111.111.111
www         IN  CNAME   server1
```

Майте на увазі, що ці псевдоніми мають певні втрати продуктивності, оскільки вони потребують додаткового запиту до сервера. Здебільшого того самого результату можна було досягти, використовуючи додаткові записи A або AAAA.

Одним із випадків, коли рекомендується CNAME, є надання псевдоніма для ресурсу поза поточною зоною.

1.7.4. MX записи.

Записи MX використовуються для визначення поштових серверів, які використовуються для домену. Це допомагає електронним повідомленням правильно надходити на ваш поштовий сервер.

На відміну від багатьох інших типів записів, поштові записи зазвичай не відображають вузол до чогось, оскільки вони застосовуються до всієї зони. Вони зазвичай виглядають так:

```
        IN  MX  10   mail.domain.com.
```

Зверніть увагу, що на початку немає імені вузла.

Також зверніть увагу, що там є додатковий номер. Це номер налаштування, який допомагає комп’ютерам вирішити, на який сервер надсилати пошту, якщо визначено кілька поштових серверів. Менші цифри мають вищий пріоритет.

Запис MX, як правило, повинен вказувати на вузол, визначений записом A або AAAA, а не на такий, який визначений CNAME.

Отже, скажімо, у нас є два поштових сервери. Повинні бути записи, які виглядають приблизно так:

```
        IN  MX  10  mail1.domain.com.
        IN  MX  50  mail2.domain.com.
mail1   IN  A       111.111.111.111
mail2   IN  A       222.222.222.222
```

У цьому прикладі вузол “mail1” є найкращим сервером обміну електронною поштою.

Ми також могли б написати це так:

```
        IN  MX  10  mail1
        IN  MX  50  mail2
mail1   IN  A       111.111.111.111
mail2   IN  A       222.222.222.222
```

1.7.5. NS записи.

Цей тип запису визначає сервери імен, які використовуються для цієї зони.

Можливо, вам цікаво, "якщо файл зони знаходиться на сервері імен, чому йому потрібно посилатися на себе?". Частиною того, що робить DNS таким успішним, є його багаторівневе кешування. Однією з причин визначення серверів імен у файлі зони є те, що файл зони може фактично обслуговуватися з кешованої копії на іншому сервері імен.

Як і записи MX, це загальнозонові параметри, тому вони також не приймають вузли. Загалом вони виглядають так:

```
        IN  NS      ns1.domain.com.
        IN  NS      ns2.domain.com.
```

Ви повинні мати принаймні два сервери імен, визначені у кожному файлі зони, для забезпечення відмовостійкості. Більшість програм DNS-сервера вважає файл зони недійсним, якщо існує лише один сервер імен.

Як завжди, включіть відображення для вузлів із записами A або AAAA:

```
        IN  NS      ns1.domain.com.
        IN  NS      ns2.domain.com.
ns1     IN  A       111.222.111.111
ns2     IN  A       123.211.111.233
```

1.7.6. PTR записи.

Використовувані записи PTR визначають ім'я, пов'язане з IP-адресою. Записи PTR є оберненими до записів A або AAAA. Записи PTR унікальні тим, що починаються з кореня .arpa і

делегуються власникам IP-адрес. Регіональні реєстри Інтернету (RIR) керують делегуванням IP-адрес організаціям та постачальникам послуг. До регіональних Інтернет-реєстрів входять APNIC, ARIN, RIPE NCC, LACNIC та AFRINIC.

Ось приклад запису PTR для 11.22.33.44:

```
44.33.22.11.in-addr.arpa.   33692   IN  PTR host.example.com.
```

Цей приклад запису PTR для адреси IPv6 показує формат реверсного запису DNS-сервера Google IPv6 2001: 4860: 4860 :: 8888.

```
8.8.8.8.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.6.8.4.0.6.8.4.1.0.0.2.ip6.arpa. 86400IN PTR google-public-dns-a.google.com.
```

2. Завдання на роботу.

2.1. Встановити та налаштувати первинний та резервний сервери для двох прямих та зворотної зон, які відповідають наступним вимогам:

- в прямих зонах обов'язково мають бути записи типів: SOA, NS, A, CNAME, які відповідають варіанту завдання;
- в зворотній зоні обов'язково мають бути записи типів: SOA, NS, PTR, які відповідають варіанту завдання.

2.2. Перевірити результат передачі (transfer) зон на резервному сервері.

2.3. Перевірити працездатність первинного та резервного серверів для всіх зон за допомогою діагностичних утиліт, наприклад: nslookup, dig, host.

2.4. Рекомендується використовувати наступне програмне забезпечення:

- DNS-сервер: bind9.

Варіанти завдання.

Варіант	**Ім'я зон**	**Ресурси зони**		
		Ім'я	**Адреса**	**Псевдонім**
1	letter.net zone01.com	alpha	172.20.1.10	ws1
		beta	172.20.1.20	ws2
		gamma	172.20.1.30	ws3
		delta	172.20.1.40	ws4
		omega	172.20.1.50	ws5
2	planet.edu zone02.org	mercury	192.168.11.21	srv1
		venus	192.168.11.22	srv2
		earth	192.168.11.23	srv3
		saturn	192.168.11.24	srv4
		jupiter	192.168.11.25	srv5

Варіант	Ім'я зон	Ресурси зони		
		Ім'я	Адреса	Псевдонім
3	cat.com zone03.net	tiger	192.168.1.11	www
		lion	192.168.1.12	ftp
		lynx	192.168.1.13	ssh
		leopard	192.168.1.14	pop3
		jaguar	192.168.1.15	imap
4	flower.org zone04.edu	rose	172.20.1.31	pc1
		gerbera	172.20.2.32	pc2
		tulip	172.20.3.33	pc3
		aster	172.20.4.34	pc4
		peony	172.20.5.35	pc5
5	linux.net zone05.com	ubuntu	172.25.11.10	node1
		debian	172.25.11.20	node2
		centos	172.25.11.30	node3
		gentoo	172.25.11.40	node4
		fedora	172.25.11.50	node5
6	color.edu zone06.net	red	192.168.22.10	ws1
		green	192.168.22.20	ws2
		blue	192.168.22.30	ws3
		black	192.168.22.40	ws4
		white	192.168.22.50	ws5
7	metal.com zone07.org	gold	172.30.10.31	srv1
		silver	172.30.10.32	srv2
		iron	172.30.10.33	srv3
		copper	172.30.10.34	srv4
		zinc	172.30.10.35	srv5
8	capital.org zone08.edu	london	192.168.33.1	www
		tokyo	192.168.33.2	ftp
		paris	192.168.33.3	ssh
		rome	192.168.33.4	pop3
		berlin	192.168.33.5	imap

Варіант	Ім'я зон	Ресурси зони		
		Ім'я	Адреса	Псевдонім
9	currency.net zone09.com	dollar	192.168.55.10	pc1
		dinar	192.168.55.20	pc2
		lira	192.168.55.30	pc3
		peso	192.168.55.40	pc4
		real	192.168.55.50	pc5
10	river.edu zone10.net	nile	172.21.11.10	node1
		amazon	172.21.11.20	node2
		congo	172.21.11.30	node3
		amur	172.21.11.40	node4
		mekong	172.21.11.50	node5
11	fruit.com zone11.org	apple	172.31.50.1	ws1
		orange	172.31.50.2	ws2
		grape	172.31.50.3	ws3
		banana	172.31.50.4	ws4
		lemon	172.31.50.5	ws5
12	digit.org zone12.edu	one	192.168.77.11	srv1
		two	192.168.77.22	srv2
		three	192.168.77.33	srv3
		four	192.168.77.44	srv4
		five	192.168.77.55	srv5
13	month.net zone13.com	march	192.168.99.10	www
		april	192.168.99.20	ftp
		may	192.168.99.30	ssh
		june	192.168.99.40	pop3
		july	192.168.99.50	imap
14	name.edu zone14.org	maria	172.16.70.1	pc1
		tomas	172.16.70.2	pc2
		tereza	172.16.70.3	pc3
		stefan	172.16.70.4	pc4
		sara	172.16.70.5	pc5

Варіант	Ім'я зон	Ресурси зони		
		Ім'я	Адреса	Псевдонім
15	country.com zone15.net	france	192.168.88.11	node1
		china	192.168.88.12	node2
		spain	192.168.88.13	node3
		italy	192.168.88.14	node4
		germany	192.168.88.15	node5

3. Контрольні запитання.

3.1. Призначення та принципи роботи DNS.
3.2. Типи DNS-серверів та їх призначення.
3.3. Типи ресурсних записів в DNS та їх призначення.
3.4. Робота сервера в ітеративному та рекурсивному режимах.
3.5. Загальна процедура перетворення доменного імені в IP-адресу.
3.6. Кешування в DNS.

4. Література.

https://web.archive.org/web/20210412192956/https://book.systemsapproach.org/applications/infrastructure.html

https://web.archive.org/web/20210330175027/https://www.digitalocean.com/community/tutorials/an-introduction-to-dns-terminology-components-and-concepts

RFC1034 https://tools.ietf.org/html/rfc1034

RFC1035 https://tools.ietf.org/html/rfc1035

Лабораторна робота № 2

Сервіс електронної пошти

1. Короткі теоретичні відомості.

Електронна пошта (email або e-mail) - це метод обміну повідомленнями (поштою) між людьми, які використовують електронні пристрої. У 1960-х роках електронна пошта використовувалася обмежено, користувачі могли надсилати її лише користувачам одного комп’ютера. Деякі системи також підтримували форму миттєвих повідомлень, коли відправник і одержувач повинні знаходитися в режимі онлайн одночасно. Рей Томлінсон вважається винахідником електронної пошти; у 1971 році він розробив першу систему, здатну надсилати пошту між користувачами на різних вузлах по ARPANET, використовуючи знак @ для зв’язку імені користувача з сервером призначення. До середини 1970-х років цю форму визнали електронною поштою.

Електронна пошта працює через комп’ютерні мережі, насамперед через Інтернет. Сучасні системи електронної пошти базуються на моделі зберігання та пересилання. Поштові сервери приймають, пересилають, доставляють та зберігають повідомлення. Ні користувачі, ні їхні комп’ютери не зобов’язані перебувати в мережі одночасно; їм потрібно підключитися, як правило до поштового сервера або через вебінтерфейс, щоб надсилати або отримувати повідомлення або завантажувати їх.

Спочатку як текстовий комунікаційний носій ASCII, електронна пошта була розширена за допомогою багатоцільових розширень Інтернет-пошти (MIME) для перенесення тексту в інших наборах символів та вкладеннях мультимедійного вмісту. Міжнародна електронна пошта з інтернаціоналізованими адресами електронної пошти, що використовують UTF-8, стандартизована, але не набула широкого поширення.

1.1. Формат повідомлення.

RFC 822 визначає повідомлення, які мають дві частини: заголовок та тіло. Обидві частини представлені в тексті ASCII. Первинно тіло повідомлення було простим текстом. Це все ще так, хоча RFC 822 був доповнений MIME, щоб дозволити тілу повідомлення нести різноманітні типи даних. Ці дані все ще представлені у вигляді тексту ASCII, але оскільки вони можуть бути кодованою версією, скажімо, зображення JPEG, вони не обов’язково читаються користувачами.

Заголовок повідомлення являє собою ряд рядків, які закінчуються символами <CRLF>. (<CRLF> позначає повернення каретки плюс подачу рядка, які є парою керуючих символів ASCII, які часто використовуються для позначення кінця рядка тексту.) Заголовок відокремлений від тіла повідомлення порожнім рядком. Кожен рядок заголовка містить тип і значення, розділені двокрапкою. Багато з цих рядків заголовків знайомі користувачам, оскільки їх просять заповнити, коли вони складають повідомлення електронної пошти; наприклад, заголовок ідентифікує одержувача повідомлення, заголовок говорить щось про мету повідомлення. Інші заголовки заповнюються базовою системою доставки пошти. Приклади включають (якщо повідомлення передане) ім'я користувача, що надіслав повідомлення та кожен поштовий сервер, який обробляв це повідомлення. Звичайно, є багато інших рядків заголовків описаних в RFC 822.

RFC 822 було розширено в 1993 році (і з тих пір неодноразово оновлювалось), щоб дозволити повідомленням електронної пошти нести багато різних типів даних: аудіо, відео, зображення, документи PDF тощо. MIME складається з трьох основних частин. Перший фрагмент являє собою набір рядків заголовків, які доповнюють вихідний набір, визначений RFC 822. Ці рядки заголовка різними способами описують дані, що переносяться в тілі повідомлення. Вони включають: версію MIME, що використовується, зручний для читання опис того, що є в повідомленні, типи даних, що містяться в повідомленні, і як дані в тілі повідомлення закодовані.

Друга частина - це визначення набору типів вмісту (та підтипів). Наприклад, MIME визначає кілька різних типів зображень, включаючи `image/gif` та `image/jpeg`, кожен із очевидним значенням. В якості іншого прикладу, `text/plane` посилається на простий текст, який ви можете знайти у звичайному повідомленні, тоді як `text/richtext` позначає повідомлення, що містить текст із «розміткою» (текст із використанням спеціальних шрифтів, курсиву тощо). Як третій приклад, MIME визначає тип програми, де підтипи відповідають результатам роботи різних прикладних програм (наприклад, `application/postscript` та `application/msword`).

MIME також визначає тип `multipart`, який говорить про те, як структуровано повідомлення, що містить більше одного типу даних. Це як мова програмування, яка визначає як базові типи (наприклад, цілі числа та числа з плаваючою комою), так і складені типи (наприклад, структури та масиви). `mixed` один із можливих `multipart` підтипів, який говорить, що повідомлення містить набір незалежних фрагментів даних у визначеному порядку. Потім кожен фрагмент має власний рядок заголовка, який описує тип цього фрагмента.

Третя частина - це спосіб кодування різних типів даних, щоб вони могли бути відправлені в електронному повідомленні ASCII. Проблема полягає в тому, що для деяких типів даних (наприклад, зображення JPEG) будь-який 8-бітовий байт зображення може містити одне з 256 різних значень. Лише підмножина цих значень є дійсними символами ASCII. Важливо, щоб повідомлення електронної пошти містили лише ASCII, оскільки вони можуть проходити через низку проміжних систем (шлюзи, як описано нижче), які передбачають, що вся електронна пошта є ASCII, і це може пошкодити повідомлення, якщо воно містить символи, що не є ASCII. Для вирішення цієї проблеми MIME використовує пряме кодування двійкових даних у набір символів ASCII. Кодування називається base64. Ідея полягає в тому, щоб відобразити кожні три байти вихідних двійкових даних у чотири символи ASCII. Це робиться шляхом групування двійкових даних у 24-бітові одиниці та розбиття кожної такої одиниці на чотири 6-бітові частини. Кожен 6-розрядний фрагмент відображається на одному з 64 дійсних символів ASCII; наприклад, 0 відображається на A, 1 на B і так далі. Якщо подивитись на повідомлення, закодоване за допомогою схеми кодування base64, то можна побачити лише 52 великі та малі літери, 10 цифр від 0 до 9 та спеціальні символи + та /. Це перші 64 значення в наборі символів ASCII.

Окрім того, щоб зробити читання пошти якомога безболіснішим для тих, хто все ще наполягає на використанні лише текстових зчитувачів пошти, повідомлення MIME, яке складається лише зі звичайного тексту, може кодуватися за допомогою 7-бітового ASCII. Також є читабельне кодування для даних ASCII.

Поєднавши все це, повідомлення, що містить звичайний текст, зображення JPEG та файл PostScript, виглядатиме приблизно так:

```
MIME-Version: 1.0
Content-Type: multipart/mixed;
boundary="-------417CA6E2DE4ABCAFBC5"
From: Alice Smith <alice@cisco.com>
To: bob@comsys.kpi.ua
Subject: test message
Date: Mon, 06 Sep 2021 19:45:19 +0200

---------417CA6E2DE4ABCAFBC5
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Bob,

Here are the jpeg image and draft report I promised.

--Alice

---------417CA6E2DE4ABCAFBC5
Content-Type: image/jpeg
Content-Transfer-Encoding: base64
... unreadable encoding of a jpeg figure
---------417CA6E2DE4ABCAFBC5
Content-Type: application/postscript; name="draft.ps"
Content-Transfer-Encoding: 7bit
... readable encoding of a PostScript document
```

У цьому прикладі рядок у заголовку повідомлення говорить, що це повідомлення містить різні фрагменти, кожен позначений рядком символів, який не відображається в самих даних. Потім кожен фрагмент має власні рядки Content-Type та Content-Transfer-Encoding.

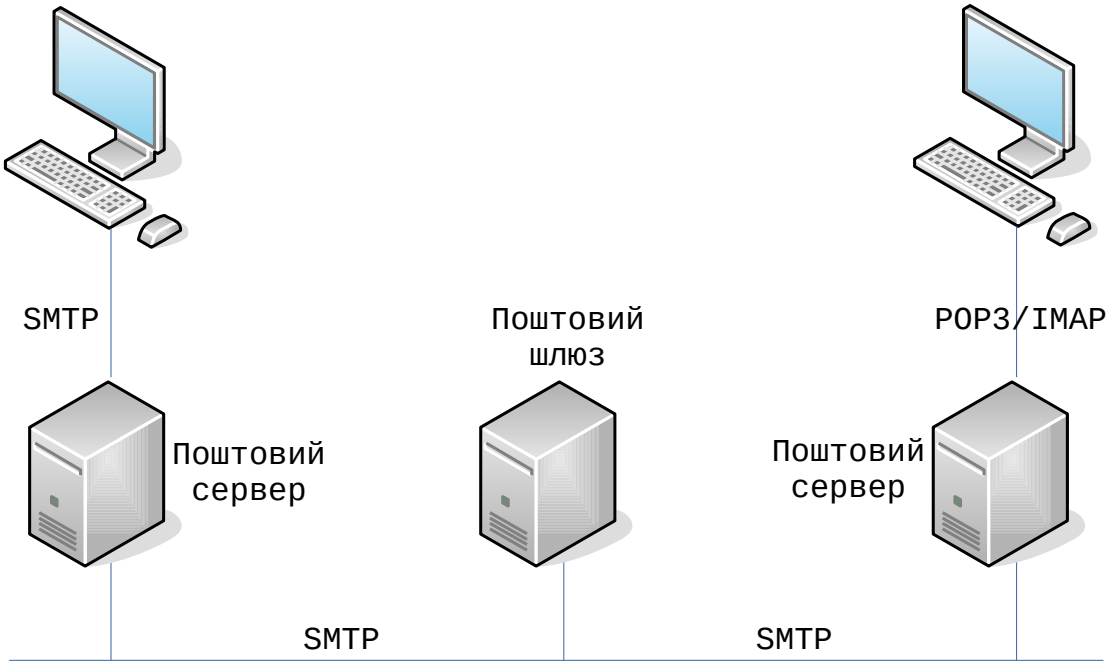
1.2. Передача повідомлень.

Протягом багатьох років більшість електронних листів переміщувались з вузла на вузол, використовуючи лише SMTP. Незважаючи на те, що SMTP продовжує відігравати центральну роль, зараз це лише один із декількох протоколів електронної пошти. Інтернет-протокол доступу до повідомлень (IMAP) та Протокол поштового відділення (POP) є ще двома важливими протоколами отримання поштових повідомлень.

Щоб зрозуміти роль SMTP у правильному контексті, необхідно визначити ключових гравців. По-перше, користувачі взаємодіють із програмою редагування пошти, коли вони складають, зберігають, шукають та читають свої електронні листи. Існує велика кількість редакторів пошти, як і багато веб-браузерів. На початку становлення мережі Інтернет користувачі, як правило, входили на вузол, на якому знаходилась їх поштова скринька, а програма зчитування пошти, яку вони викликали, була локальною програмою, яка завантажувала повідомлення з файлової системи. Сьогодні, звичайно, користувачі віддалено отримують доступ до своєї поштової скриньки зі свого ноутбука чи смартфона; вони не входять на вузол, що зберігає їхню пошту (поштовий сервер). Другий протокол передачі пошти, такий як

POP або IMAP, використовується для віддаленого завантаження електронної пошти з поштового сервера на пристрій користувача.

По-друге, на кожному вузлі, що містить поштову скриньку, працює поштовий демон (або процес). Цей демон називається агентом передачі повідомлень (MTA), який відіграє роль поштового відділення. Користувачі (або їх редактори пошти) передають демону повідомлення, які вони хочуть надіслати іншим користувачам, демон використовує запущений SMTP через TCP для передачі повідомлення демону, що працює на іншій машині, і демон розміщує вхідні повідомлення в поштовій скриньці користувача (де редактор пошти цього користувача може їх пізніше знайти). Оскільки SMTP - це протокол, який може реалізувати кожен, теоретично може існувати безліч різних реалізацій поштового демона. На практиці широко використовується лише декілька популярних реалізацій, причому найпоширенішими є стара програма sendmail від Berkeley Unix, postfix та exim.



Малюнок 1. Пересилання повідомлень електронної пошти.

Хоча, можливо, що MTA на машині відправника встановлює SMTP/TCP-з'єднання з MTA на поштовому сервері одержувача, у багатьох випадках пошта передається через один або кілька поштових шлюзів на своєму шляху від вузла відправника до вузла одержувача. Як і на кінцевих вузлах, ці шлюзи також використовують демон агента передачі повідомлень. Не випадково ці проміжні вузли називаються шлюзами, оскільки їх робота полягає у зберіганні та пересиланні повідомлень електронної пошти, подібно до того, як “шлюз IP” (який називається маршрутизатором) зберігає та пересилає пакети IP. Єдина різниця полягає в тому, що поштовий шлюз зазвичай буферизує повідомлення на диску і готовий спробувати повторно передати їх на наступний вузол протягом декількох днів, тоді як IP-маршрутизатор буферизує пакети в пам'яті і готовий повторити спробу їх передачі лише за долю секунди. Малюнок 1 ілюструє шлях із одним проміжним вузлом на шляху від відправника до одержувача.

Для чого потрібні поштові шлюзи? Чому вузол відправника не може надіслати повідомлення вузлу одержувача? Одна з причин полягає в тому, що одержувач не хоче вказувати конкретний вузол, на якому він читає електронну пошту, що надходить на його адресу. Інша причина - це масштаб: у великих організаціях часто трапляється, що на різних вузлах зберігаються поштові скриньки організації. Наприклад, пошта, що надсилається на bob@comsys.kpi.ua, спочатку може надсилатися на поштовий шлюз mail.comsys.kpi.ua, а потім пересилається - за допомогою другого з'єднання до конкретного вузла, на якій bob має поштову скриньку. Шлюз переадресації підтримує базу даних, яка відображає користувачів у вузли, на яких знаходяться їх поштові скриньки. Відправник не повинен знати цього конкретного імені. (Список заголовків рядків у повідомленні допоможе простежити шлюзи пошти, по яких пройшло певне повідомлення.) Ще однією причиною, особливо на ранніх етапах розвитку електронної пошти, є те, що вузол, на якому розміщується поштова скринька будь-якого користувача, не завжди може бути доступний і в цьому випадку поштовий шлюз зберігає повідомлення до моменту його доставки.

Незалежно від кількості поштових шлюзів на шляху, незалежне з'єднання SMTP використовується між кожним вузлом для переміщення повідомлення ближче до одержувача. Кожен сеанс SMTP включає діалог між двома демонами пошти, причому один виконує роль клієнта, а інший - сервера. Під час одного сеансу між двома вузлами може передаватися кілька повідомлень. Оскільки RFC 822 визначає повідомлення, що використовують ASCII як базове представлення, SMTP також базується на ASCII. Це означає, що користувач може видавати себе клієнтською програмою SMTP.

SMTP найкраще зрозуміти на простому прикладі. Далі наводиться обмін між відправляючим доменом comsys.kpi.ua та приймаючим доменом cisco.com. У цьому випадку користувач bob з comsys.kpi.ua намагається надіслати пошту користувачам alice та tom в cisco.com. Додано додаткові порожні рядки, щоб зробити діалог більш читабельним.

```
HELO pc.comsys.kpi.ua
250 mail.comsys.kpi.ua Hello pc.comsys.kpi.ua [10.18.51.101]

MAIL FROM:<bob@cs.comsys.kpi.ua >
250 OK

RCPT TO:<alice@cisco.com>
250 OK

RCPT TO:<tom@cisco.com>
550 No such user here

DATA
354 Start mail input; end with <CRLF>.<CRLF>
Hello Alice and Tom!
<CRLF>.<CRLF>
250 OK

QUIT
221 Closing connection
```

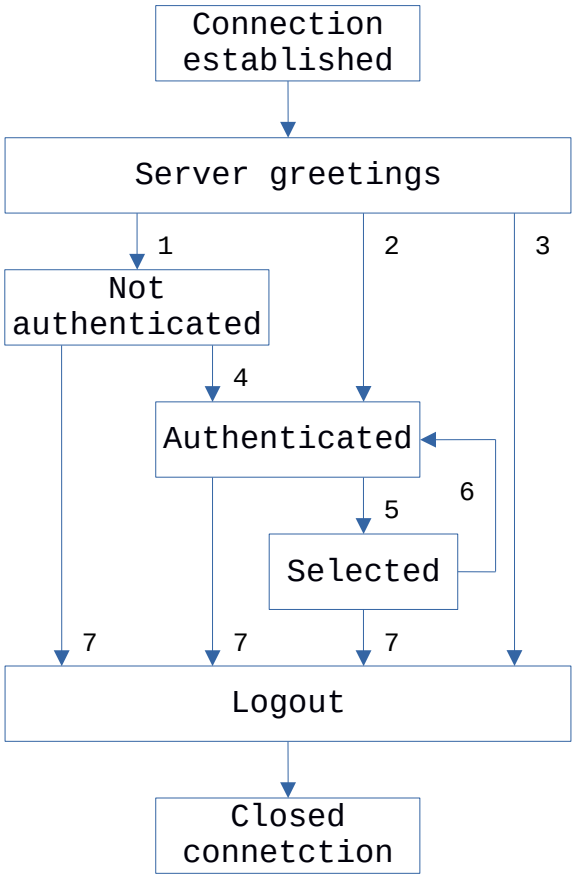
Як видно, SMTP передбачає послідовність обмінів між клієнтом та сервером. У кожному обміні клієнт публікує команду (наприклад, QUIT), а сервер відповідає кодом (наприклад, 250, 550, 354, 221). Сервер також повертає зручне для читання пояснення коду (наприклад, "No such user here"). У цьому конкретному прикладі клієнт спочатку ідентифікує себе на сервері за допомогою команди HELO. В якості аргументу він надає своє доменне ім’я. Сервер перевіряє, що це ім'я відповідає IP-адресі, що використовується TCP-з'єднанням (сервер повідомляє цю IP-адресу клієнту). Потім клієнт запитує у сервера, чи готовий він прийняти пошту для двох різних користувачів; сервер відповідає, кажучи "так" одному, а "ні" іншому. Потім клієнт надсилає повідомлення, яке закінчується рядком із єдиною крапкою ("."). Нарешті, клієнт розриває з'єднання.

Звичайно, існує безліч інших команд і кодів повернення. Наприклад, сервер може відповісти на команду клієнта RCPT кодом 251, який вказує на те, що користувач не має поштової скриньки на цьому вузлі, але що сервер обіцяє переслати повідомлення на інший поштовий демон. Іншими словами, вузол функціонує як поштовий шлюз. В якості іншого прикладу, клієнт може здійснити операцію VRFY для перевірки електронної адреси користувача, але фактично не надсилаючи повідомлення користувачеві.

1.3. Редактор пошти.

Останній крок - це те, що користувач фактично отримує свої повідомлення з поштової скриньки, читає їх, відповідає на них і, можливо, зберігає копію для подальшого використання. Усі ці дії користувач виконує, взаємодіючи з програмою редагування пошти. Як зазначалося раніше, спочатку цей зчитувач був просто програмою, що працює на тій самій машині, що і поштова скринька користувача, і в цьому випадку він міг просто читати та писати файл, який реалізує поштову скриньку. Це було поширеним випадком в епоху до ноутбуків. Сьогодні найчастіше користувач отримує доступ до своєї поштової скриньки з віддаленої машини, використовуючи ще один протокол, такий як POP або IMAP.

IMAP багато в чому схожий на SMTP. Це протокол клієнт/сервер, що працює через TCP, де клієнт (який працює на комп'ютері користувача) видає команди у вигляді <CRLF> -термінованих текстових рядків ASCII та поштового сервера (працює на вузлі, яка підтримує поштову скриньку користувача). Обмін починається з того, що клієнт автентифікує себе та ідентифікує поштову скриньку, до якої він хоче отримати доступ. Це може бути представлено простою діаграмою переходу стану, зображеною на малюнку 2. На цій діаграмі LOGIN і LOGOUT - це приклади команд, які може видавати клієнт, тоді як OK - одна з можливих відповідей сервера. Інші загальні команди включають і EXPUNGE, з очевидними значеннями. Додаткові відповіді сервера включають NO (клієнт не має дозволу виконувати цю операцію) та BAD (команда неправильно сформована).



Малюнок 2. Діаграма переходу стану IMAP.

Коли користувач відправляє команду `FETCH`, сервер повертає повідомлення у форматі MIME, а програму редагування пошти декодує. Окрім самого повідомлення, IMAP також визначає набір атрибутів повідомлення, якими обмінюються як частина інших команд, незалежно від передачі самого повідомлення. Атрибути повідомлення включають таку інформацію, як розмір повідомлення та, що ще цікавіше, різні прапори, пов’язані з повідомленням (наприклад, `Seen`, `Answered`, `Deleted` та `Recent`). Ці прапорці використовуються для синхронізації клієнта та сервера; тобто, коли користувач видаляє повідомлення у програмі зчитування пошти, клієнт повинен повідомити про цей факт поштовий сервер. Пізніше, якщо користувач вирішить видалити всі видалені повідомлення, клієнт відправляє команду `EXPUNGE` серверу, який знає, що насправді видаляє всі раніше помічені на видалення повідомлення з поштової скриньки.

Нарешті, коли користувач відповідає на повідомлення або надсилає нове повідомлення, програма редагування пошти не пересилає повідомлення від клієнта на поштовий сервер за допомогою IMAP, а замість цього використовує SMTP. Це означає, що поштовий сервер користувача є фактично першим поштовим шлюзом, пройденим по шляху від комп'ютера відправника до поштової скриньки одержувача.

2. Завдання на роботу.

2.1. Створити центр сертифікації (CA) issuer і subject якого відповідають варіанту завдання. Сертифікати для поштових серверів мають підписуватись сертифікатом та ключем створеного CA. Сертифікат створеного CA необхідно додати в список довірених на всіх вузлах, які використовуються в роботі.

2.2. Встановити та налаштувати 2 поштових сервери (mail/message transfer agent - MTA) для пересилки повідомлень електронної пошти за допомогою протоколу SMTP, які відповідають наступним вимогам:

- кожний поштовий сервер є кінцевим отримувачем повідомлень для домену, ім'я якого відповідає варіанту завдання;
- доменне ім'я поштового серверу має відповідати формату: mail.<ім'я домену>;
- в кожному домені створені поштові скриньки з іменами, які відповідають варіанту завдання;
- обов'язкова автентифікація користувачів при відправленні повідомлень на зовнішні MTA;
- підтримка TLS для передачі повідомлень;
- вихідні повідомлення містять підпис DKIM;
- перевірка вхідних повідомлень на наявність та коректність підпису DKIM, повідомлення з некоректним підписом відкидаються;
- перевірка домену відправника повідомлення та обробка повідомлення у відповідності з SPF, яка відповідає варіанту завдання;
- перевірка повідомлень на наявність шкідливого програмного забезпечення;
- контекстний аналіз повідомлень для виконання спам-фільтрації.

2.3. Для кожного сервера додати функціонал доставлення повідомлень засобами протоколів POP3 та IMAP з підтримкою TLS.

2.4. Виконати аналіз протокольного обміну між вузлами під час відправлення, пересилки та доставлення поштових повідомлень. Проаналізувати заголовки листів.

2.5. Рекомендується використовувати наступне програмне забезпечення:

- MTA: exim або postfix;
- доставлення повідомлень засобами POP3 та IMAP: dovecot;
- антивірусна фільтрація: ClamAV;
- спам-фільтрація: SpamAssassin.

2.6. Кожний поштовий сервер рекомендується встановлювати та налаштовувати на окремій віртуальній машині. Для перевірки роботи модулів поштового сервера та аналізу протокольного обміну рекомендується використовувати утиліти: telnet, openssl, swaks.

2.7. Додати необхідні записи типів: MX та TXT для забезпечення функціонування поштових серверів згідно з варіантом завдання.

Варіанти завдання.

Варіант	**Issuer і Subject CA**	**Домен 1**	**Домен 2**	**Поштові скриньки**	**Політика SPF**
1	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=letter CA	letter.net	zone01.com	alpha@letter.net beta@letter.net gamma@letter.net delta@zone01.com omega@zone01.com	v=spf1 mx -all

Варіант	Issuer і Subject CA	Домен 1	Домен 2	Поштові скриньки	Політика SPF
2	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=planet CA	planet.edu	zone02.org	mercury@planet.edu venus@planet.edu earth@planet.edu saturn@zone02.org jupiter@zone02.org	v=spf1 mx ~all
3	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=cat CA	cat.com	zone03.net	tiger@cat.com lion@cat.com lynx@cat.com leopard@zone03.net jaguar@zone03.net	v=spf1 a mx -all
4	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=flower CA	flower.org	zone04.edu	rose@flower.org gerbera@flower.org tulip@flower.org aster@zone04.edu peony@zone04.edu	v=spf1 ptr mx -all
5	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=linux CA	linux.net	zone05.com	ubuntu@linux.net debian@linux.net centos@linux.net gentoo@zone05.com fedora@zone05.com	v=spf1 a mx ~all
6	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=color CA	color.edu	zone06.net	red@color.edu green@color.edu blue@color.edu black@zone06.net white@zone06.net	v=spf1 ptr mx ~all
7	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=metal CA	metal.com	zone07.org	gold@metal.com silver@metal.com iron@metal.com copper@zone07.org zinc@zone07.org	v=spf1 mx -all
8	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=capital CA	capital.org	zone08.edu	london@capital.org tokyo@capital.org paris@capital.org rome@zone08.edu berlin@zone08.edu	v=spf1 mx ~all
9	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=currency CA	currency.net	zone09.com	dollar@currency.net dinar@currency.net lira@currency.net peso@zone09.com real@zone09.com	v=spf1 a mx -all

Варіант	Issuer і Subject CA	Домен 1	Домен 2	Поштові скриньки	Політика SPF
10	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=river CA	river.edu	zone10.net	nile@river.edu amazon@river.edu congo@river.edu amur@zone10.net mekong@zone10.net	v=spf1 ptr mx -all
11	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=fruit CA	fruit.com	zone11.org	apple@fruit.com orange@fruit.com grape@fruit.com banana@zone11.org lemon@zone11.org	v=spf1 a mx ~all
12	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=digit CA	digit.org	zone12.edu	one@digit.org two@digit.org three@digit.org four@zone12.edu five@zone12.edu	v=spf1 ptr mx ~all
13	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=month CA	month.net	zone13.com	march@month.net april@month.net may@month.net june@zone13.com july@zone13.com	v=spf1 mx -all
14	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=name CA	name.edu	zone14.org	maria@name.edu tomas@name.edu tereza@name.edu stefan@zone14.org sara@zone14.org	v=spf1 mx ~all
15	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=country CA	country.com	zone15.net	france@country.com china@country.com spain@country.com italy@zone15.net germany@zone15.net	v=spf1 a mx -all

3. Контрольні питання.
3.1. Архітектура електронної пошти.
3.2. Призначення та принципи роботи протоколів SMTP, POP3 та IMAP.
3.3. Формат поштового повідомлення.
3.4. Маршрутизації повідомлень електронної пошти.
3.5. Методи автентифікації поштових повідомлень DKIM та SPF.
3.6. Технологія DMARC.

4. Література.

https://web.archive.org/web/20210412192955/https://book.systemsapproach.org/applications/traditional.html

https://en.wikipedia.org/wiki/Email

RFC5321 https://tools.ietf.org/html/rfc5321

RFC1939 https://tools.ietf.org/html/rfc1939

RFC3501 https://tools.ietf.org/html/rfc3501

RFC6376 https://tools.ietf.org/html/rfc6376

RFC7208 https://tools.ietf.org/html/rfc7208

Лабораторна робота № 3

Сервіс передачі файлів (FTP)

1. Короткі теоретичні відомості.

File Transfer Protocol (FTP) - це стандартний мережний протокол, який використовується для обміну файлами через комп'ютерну мережу, наприклад Інтернет. FTP побудований на архітектурі клієнт-сервер і використовує окремі з'єднання для керування та передачі даних між клієнтом та сервером. Клієнтські програми спочатку були інтерактивними інструментами командного рядка зі стандартизованим синтаксисом команд, але графічні інтерфейси користувача були розроблені для всіх настільних операційних систем, що використовуються сьогодні. FTP також часто використовується як компонент програми для автоматичної передачі файлів для внутрішніх функцій програми. FTP можна використовувати з виконанням автентифікації на основі пароля користувача або анонімно.

1.1. Способи підключення.

FTP працює через протокол управління передачею TCP. Зазвичай сервери FTP прослуховують порт номер 21 (зарезервований IANA) для вхідних з'єднань від клієнтів. Підключення до цього порту з FTP-клієнта формує керуюче з'єднання, за яким передаються на передаються передаються і відповіді. FTP відкриває спеціальні з'єднання для передачі даних на інші номери портів. Параметри потоків даних залежать від конкретно режиму транспортування. Для передачі даних зазвичай використовується порт номер 20.

У активному режимі клієнт FTP відкриває порт, надсилає серверу FTP номер цього порту через з'єднання даних, і чекає з'єднання з сервера FTP. Коли FTP-сервер ініціює підключення даних до FTP-клієнта, він прив'язує вихідний порт до порту 20 на FTP-сервері.

Для того, щоб використовувати активний режим, клієнт надсилає команду PORT з IP та портом як аргументом. Формат IP та порту - "h1, h2, h3, h4, p1, p2". Кожне поле є десятковим поданням 8 бітів IP-адреси вузла, за яким слідує обраний порт даних. Наприклад, клієнт з IP-адресою 192.168.0.1, прослуховуючи порт 49154 для з'єднання даних, надішле команду "`PORT 192,168,0,1,192,2`". Поля порту слід інтерпретувати як p1 × 256 + p2 = порт (у цьому прикладі, 192 × 256 + 2 = 49154).

У пасивному режимі FTP-сервер відкриває порт, надсилає FTP-клієнту IP-адресу сервера, до якого він підключається, та порт, на якому він прослуховує (16-бітове значення, розбите на старший та молодший байти, як показано вище) і чекає з'єднання з FTP-клієнтом. У цьому випадку FTP-клієнт встановлює з'єднання даних на порт, вказаний сервером.

Для використання пасивного режиму клієнт надсилає команду PASV, на яку сервер відповість чимось подібним до "`227 Entering Passive Mode (192,168,0,1,192,52)`". Синтаксис IP-адреси та порту такий самий, як і для аргументу команди PORT.

У розширеному пасивному режимі FTP-сервер працює точно так само, як і пасивний, проте він передає лише номер порту (не розбитий на старший та молодший байти), і клієнт повинен припустити, що він підключається до тієї самої IP-адреси сервера, до якої встановлено керуюче з'єднання. Розширений пасивний режим був доданий RFC 2428 у вересні 1998 року.

Поки дані передаються через потік даних, потік керування не працює. Це може спричинити проблеми з великою передачею даних через брандмауери, які очікують сеанси після тривалих періодів простою. Хоча файл цілком може бути успішно переданий, брандмауер може відключити сеанс керування, що спричинить генерування помилки.

Протокол FTP підтримує відновлення перерваних завантажень за допомогою команди REST. Клієнт передає кількість байтів, яку він вже отримав як аргумент, команді REST і перезапускає передачу. Наприклад, у деяких клієнтах командного рядка існує часто ігнорувана, але цінна команда "reget" (що означає "отримати знову"), яка призведе до продовження перерваної команди "get".

Відновити завантаження не так просто. Хоча протокол FTP підтримує команду APPE для додавання даних до файлу на сервері, клієнт не знає точної позиції, в якій перервана передача. Він повинен отримати розмір файлу іншим способом, наприклад, за допомогою списку каталогів або за допомогою команди SIZE.

У режимі ASCII (див. Нижче) відновлення передачі може бути проблематичним, якщо клієнт і сервер використовують різні символи кінця рядка.

1.2. Проблеми з безпекою.

Оригінальна специфікація FTP є за своєю суттю незахищеним способом передачі файлів, оскільки не існує жодного способу передачі даних і команд в зашифрованому вигляді. Це означає, що в більшості мережних конфігурацій імена користувачів, паролі, команди FTP та передані файли можуть бути захоплені будь-якою людиною за допомогою сніфера пакетів. Це проблема, загальна для багатьох специфікацій протоколів Інтернету, написаних до створення рівня захищених сокетів (SSL), таких як HTTP, SMTP та Telnet. Загальним рішенням цієї проблеми є використання SFTP (протокол передачі файлів SSH) або FTPS (FTP поверх SSL), який додає до FTP шифрування SSL або TLS, як зазначено в RFC 4217.

1.3. Коди повернення FTP.

Коди повернення сервера FTP вказують їх статус цифрами, що знаходяться в них. Коротке пояснення значень різних кодів наведено нижче:

- 1xx: Позитивна попередня відповідь. Запрошена дія ініціюється, але перед її початком буде надіслана інша відповідь.
- 2xx: Позитивна відповідь на завершення. Запитану дію завершено. Клієнт тепер може виконати нову команду.
- 3xx: Позитивна проміжна відповідь. Команда була успішною, але необхідна подальша команда, перш ніж сервер зможе діяти за запитом.
- 4xx: Перехідна відповідь негативного завершення. Команда не була успішною, але клієнт може вільно спробувати команду ще раз, оскільки помилка є лише тимчасовою.
- 5xx: Постійна негативна відповідь. Команда не була успішною, і клієнт не повинен намагатися повторити її ще раз.
- x0x: Помилка сталася через синтаксичну помилку.
- x1x: Ця відповідь є відповіддю на запит на інформацію.
- x2x: Ця відповідь є відповіддю, що стосується інформації про з'єднання.
- x3x: Ця відповідь є відповіддю, що стосується обліку та авторизації.
- x4x: поки не вказано

- x5x: ці відповіді вказують на стан файлової системи сервера щодо запитуваної передачі чи іншої дії файлової системи.

1.4. Анонімний FTP.

Вузол, який надає послугу FTP, може додатково забезпечити анонімний доступ до FTP. Зазвичай користувачі входять до служби за допомогою анонімного облікового запису, коли з'являється запит на ім'я користувача. Хоча користувачів зазвичай просять надіслати свою електронну адресу замість пароля, фактично перевірка наданих даних не виконується.

Оскільки сучасні FTP-клієнти зазвичай приховують анонімний процес входу від користувача, ftp-клієнт надасть фіктивні дані на запит пароля (оскільки адреса електронної пошти користувача може бути невідома застосунку). Наприклад, такі агенти користувача ftp вказують перелічені паролі для анонімних входів:

- Mozilla Firefox (3.5.2) - `mozilla@example.com`
- KDE Konqueror (3.5) - `anonymous@`
- wget (1.10.2) - `wget@`
- lftp (3.4.4) - `lftp@`
- Opera (9.6.4) - `opera@`

Протокол Gopher пропонується як альтернатива анонімному FTP, а також протокол службових файлів.

1.5. Параметри передачі.

Відповідно до стандарту FTP RFC959 передача даних визначається чотирма основними параметрами:

- структура даних: орієнтована на потік, на запис або на сторінку
- тип даних: текстові типи ASCII, EBCDIC, з підтипами для різних варіантів керування кареткою; бінарні типи орієнтовані на байт або довільну довжину слів
- контроль вертикального формату: для текстових типів ASCII та EBCDIC, якщо вказано вертикальне управління форматом
- режим передачі: передача, орієнтована на потік, нестиснута передача, орієнтована на блок, або передача, орієнтована на стиснений блок

До 1990-х років використання FTP зосереджувалося на структурі файлів, орієнтованих на потік, та режимі передачі, орієнтованих на потік; більшість FTP-серверів та клієнтів, починаючи з 1990-х років, не підтримують інші файлові структури або режими передачі.

1.6. Структура даних.

Структура даних визначається за допомогою команди STRU. Наступні файлові структури визначені в розділі 3.1.1 RFC959:

- Структура F або FILE (орієнтована на потік). Файли розглядаються як довільна послідовність байтів, символів або слів. Це звичайна структура файлів у системах Unix та інших системах, таких як CP/M, MSDOS та Microsoft Windows. [Розділ 3.1.1.1]
- R або RECORD структура (орієнтована на запис). Файли розглядаються як розділені на записи, які можуть мати фіксовану або змінну довжину. Ця організація файлів є загальною для систем мейнфреймів та середнього класу, таких як MVS, VM/CMS, OS/400 та VMS.

- P або PAGE структура (орієнтована на сторінку). Файли розділені на сторінки, які можуть містити дані або метадані; кожна сторінка може також мати заголовок із різними атрибутами. Ця файлова структура була спеціально розроблена для систем TENEX і, як правило, не підтримується на інших платформах. RFC1123, розділ 4.1.2.3, рекомендує не застосовувати цю структуру.

1.7. Типи даних.

Тип даних визначається за допомогою команди TYPE. Визначено такі типи даних:

- A (ASCII). Текстові дані, передані через мережу в наборі символів NVT ASCII.
- E (EBCDIC). Текстові дані, передані через мережу в наборі символів EBCDIC.
- I або IMAGE (орієнтований на байти). Двійкові дані передаються у вигляді потоку 8-бітових байтів.
- L або LOCAL (орієнтований на слова). Двійкові дані, передані як потік слів. Кількість бітів у слові вказується як аргумент, наприклад: L32 для 32-розрядних слів, L36 для 36-розрядних слів.

Історично поширеною проблемою були клієнти та сервери FTP, які за замовчуванням мають тип ASCII, але не забезпечують жодного захисту від передачі двійкових файлів. В результаті двійкові файли пошкоджені, наприклад, заміна символів нового рядка. У більшості сучасних клієнтів цього можна уникнути, якщо автоматично встановити тип зображення. Іншим підходом буде вибір FTP TYPE на основі типу файлу, записаного у файловій системі (для тих файлових систем, які роблять це), або евристично.

L8 фактично еквівалентний I, і більшість FTP-серверів або клієнтів не приймають інші розміри слів, крім 36-розрядних платформ. Дані повинні бути передані в упакованому двійковому форматі.

Зверніть увагу, тип даних вказує тип для передачі, а не тип, в якому дані зберігаються в системах відправника або отрмувача. Клієнт та сервер можуть вільно перетворювати дані у форму, яка є найбільш зручною на їх платформі. Наприклад, текстові типи даних A та E можуть піддаватися трансляції набору символів (наприклад, ASCII проти EBCDIC), замін символів нового рядка (наприклад, CRLF проти LF) або трансляція текстових даних між орієнтованими на потік та орієнтованими на запис форматами (тобто один запис на рядок, можливо, заповнений пробілами до максимальної довжини рядка проти орієнтованого на потік символами нового рядка для розділення рядків).

Часто FTP-клієнти використовують слово "MODE" для позначення типу даних, хоча це помилково, оскільки слово "MODE" вже прийнято для позначення режиму передачі.

1.8. Режим передачі.

Режим передачі визначається командою MODE. Визначаються наступні режими:

- S або STREAM MODE: дані представлені у вигляді потоку 8-бітових байтів. Для файлів, орієнтованих на запис, визначено механізм екранування, який чітко вказує межі записів та явний кінець файлу. Для потоково-орієнтованих файлів не визначено жодного механізму екранування, а кінець файлу представлений закриттям з'єднання.
- B або BLOCK MODE: дані представлені у вигляді потоку блоків. Кожен блок має заголовок, який вказує його довжину, а також прапори для позначення кінця запису та кінця файлу. Прапори також можуть використовуватися для позначення підозрілого блоку даних, наприклад блок даних, зчитуваний з магнітної стрічки, який мав помилку

контрольної суми, але все одно передається, хоча він може містити спотворені дані. Також підтримує маркери перезапуску, які дозволяють перезапустити передачу даних з цієї точки.

- C або COMPRESSED MODE: подібний до потокового режиму, але додає підтримку кодування довжини циклу, а також прапори, визначені в блочному режимі.

Зараз більшість FTP-клієнтів та серверів підтримують лише режим STREAM.

1.9. FTP команди.

Команди, які починаються з літери X, зазвичай зарезервовані для експериментальних розширень, хоча замість цього слід використовувати підкоманди SITE.

RFC959 визначає такі команди FTP, які також були присутні в RFC765:

- USER: надає ім’я користувача для входу.
- PASS: надає пароль для входу.
- ACCT: надає облікову інформацію. Наприклад, користувач може працювати над кількома проектами; обліковий запис може бути використаний для того, щоб стягнути плату за зберігання даних за правильним проектом. (Зазвичай не застосовується).
- CWD: змінює робочий каталог на вказаний.
- REIN: видаляє всю інформацію про автентифікацію та параметри; має відбуватися повторна реєстрація через USER.
- QUIT: розриває з'єднання.
- PORT: вказує вузол/порту для передачі даних.
- PASV: увійти в пасивний режим.
- TYPE: вказує тип даних та вертикальний контроль формату.
- STRU: вказує структуру даних.
- MODE: вказує режим передачі.
- RETR: ініціює передачу даних із сервера на клієнт із зазначенням імені файлу для отримання.
- STOR: ініціює передачу даних з клієнта на сервер, вказавши ім'я файлу, який слід зберігати на сервері.
- APPE: подібно до STOR, за винятком того, що файл вже існує, додає отримані дані до кінця, а не створює новий.
- ALLO: виділяє місце для файлу. Необов’язково вказує максимальний розмір кожного запису.
- REST: визначає маркер перезапуску, з якого слід відновити передачу. Спочатку призначений для використання з маркерами перезапуску, надісланими сервером у режимі B або C, але згодом розширений у RFC3659 до зміщення байтів, зазначених у режимі S.
- RNFR: щоб перейменувати файл, вкажіть файл, який буде перейменовано.
- RNTO: щоб перейменувати файл, вказує нову назву файлу та виконує перейменування. Часто також використовується для переміщення файлів.
- DELE: видаляє файл.
- PWD: друкує поточний робочий каталог.
- LIST: відкриває з'єднання даних із типом даних A або E для передачі списку файлів у поточному каталозі. Формат даних специфічний для системи, але призначений для читання людиною.

- NLST: подібний до LIST, але передає без розмітки імена файлів за допомогою CRLF або NL.
- SITE: надає підкоманди для виконання певних системних послуг. Характер цих послуг не визначений.
- STAT: без аргументів, поточний стан з'єднання. З аргументом, еквівалентним LIST, але список передається через контрольне з'єднання, інкапсульоване в повідомленнях.
- HELP: надає довідку, необов’язково аргумент для вказівки конкретної команди, за якою запитується допомога.
- NOOP: нічого не робить.

RFC959 додає такі нові команди, яких не було в RFC765:

- CDUP: змінює робочий каталог на батьківський. В даний час позначення батьківського каталогу різняться залежно від платформи (хоча найчастіше .. в системах, що походять з Unix або MS DOS).
- SMNT: підключіть іншу файлову систему або том. Призначений для систем, таких як DOS або VMS, де існує різниця між томом та каталогом у іменах шляхів; але зазвичай не виконуються навіть на таких системах.
- STOU: унікальний завантаження на сервер - ініціює передачу даних з клієнта на сервер; сервер повинен вибрати унікальне ім'я для файлу, який потрібно отримати.
- RMD: видаляє каталог.
- MKD: створює каталог.
- SYST: визначає операційну систему сервера.

RFC765 описав ряд команд, які були видалені в RFC959. Вони не були частиною реалізацій FTP з початку 1980-х, оскільки їх функціональність пізніше (частково) була замінена SMTP:

- MLFL: використовується для надсилання електронної пошти через з'єднання для передачі даних.
- MAIL: використовується для надсилання електронної пошти через контрольне з'єднання.
- MSND: як MAIL, але надсилає дані безпосередньо на термінал користувача, а не на їх поштову скриньку.
- MSOM: поводиться як MAIL або MSND - надсилати на термінал, якщо це дозволено, інакше на поштову скриньку.
- MSAM: подібний до MSOM - за винятком того, що MSOM надсилає на поштову скриньку лише в тому випадку, якщо доставка на термінал неможлива; але MSAM надсилає на поштову скриньку незалежно від того, чи успішно здійснена спроба доставки на термінал.
- MRSQ: дозволяє передавати одне електронне повідомлення декільком користувачам на одному вузлі.
- MRCP: після MRSQ ідентифікує одного з таких одержувачів; повторюється для кожного одержувача.

RFC2228 додає ряд команд, пов’язаних із шифруванням та автентифікацією повідомлень:

- AUTH: визначає механізм автентифікації/захисту, який буде використовуватися.
- ADAT: визначає дані безпеки, характерні для обраного механізму AUTH.
- PBSZ: використовується для узгодження максимального розміру буфера для зашифрованих даних.
- PROT: визначає рівень захисту каналу даних. Визначені такі рівні:
  - C (Clear) - канал передачі даних не підлягає ні шифруванню, ні захисту цілісності.

    - S (Safe) - захист цілісності, що застосовується до каналу даних.
    - E (конфіденційно) - шифрування, що застосовується до каналу даних.
    - P (Private) - як шифрування, так і захист цілісності, що застосовуються до каналу даних.
- CCC: вимикає захист цілісності для наступних команд в каналі управління.
- MIC: надсилає команду із захистом цілісності.
- CONF: надсилає команду із захистом конфіденційності.
- ENC: надсилає команду із захистом цілісності та конфіденційності.

RFC1639 додає підтримку FTP через довільні транспортні протоколи, такі як IPX/SPX. Для цього він визначає дві нові команди:

- LPRT: схожий на PORT, але підтримує довільні формати адрес та портів.
- LPSV: подібне розширення до PASV.

RFC2389 визначає дві нові команди, що використовуються як загальний механізм розширення для FTP:

- FEAT: отримує список додаткових функцій, що підтримуються FTP-сервером.
- OPTS: загальний механізм для клієнта для вказівки параметрів довільних команд FTP.

RFC2428 додає дві нові команди, подібні в принципі до RFC1639, але різні в деталях:

- EPRT: подібний до PORT, але підтримує довільні сімейства адрес, а не лише IPv4; спеціально призначений для IPv6.
- EPSV: подібне розширення до PASV

LPRT надсилає адреси у вигляді довільного рядка-октету (хоча і в десятковому кодуванні), EPRT - у форматованому рядку, формат рядка залежить від формату адреси. EPRT передбачає використання 16-розрядних номерів портів у стилі TCP, тоді як LPRT є більш гнучким і підтримує транспортні протоколи з номерами портів, що перевищують 16-бітові.

RFC2640 додає одну нову команду:

- LANG: використовується для вибору мови для FTP-повідомлень

RFC3659 визначає кілька нових команд:

- MDTM: отримує час модифікації файлу
- SIZE: отримати розмір файлу
- MLSD: отримання списку файлів у каталозі. На відміну від NLST, повертає не тільки імена файлів, а й атрибути; але на відміну від LIST, він повертає атрибути у розширюваному стандартизованому форматі, а не в довільному, специфічному для платформи.
- MLST: те саме, що MLSD, але отримує список для окремого файлу, а не каталогу. Для каталогів отримує власні атрибути, а не їх перелік. MLST не вимагає з'єднання даних, але повертає один рядок, що містить список для запитуваного шляху.

1.10. FTP і NAT.

Представлення IP-адрес та номерів портів у команді PORT та відповіді PASV створює ще одну проблему для пристроїв трансляції мережних адрес (NAT) при обробці FTP. Пристрій NAT має змінити ці значення, щоб вони містили IP-адресу NAT-клієнта та порт, вибраний NAT-пристроєм для з'єднання даних. Нова адреса та порт, можливо, за своєю десятковою подачею відрізнятимуться від початкової адреси та порту. Це означає, що зміна значень на контрольному підключенні пристроєм NAT повинна виконуватися обережно, змінюючи поля TCP Sequence і

Acknowledgment для всіх наступних пакетів. Така трансляція зазвичай не виконується в більшості пристроїв NAT, але для цього існують спеціальні шлюзи прикладного рівня.

1.11. FTP через SSH.

FTP через SSH іноді називають безпечним FTP; його не слід плутати з іншими методами захисту FTP, такими як SSL/TLS (FTPS). Інші методи передачі файлів за допомогою SSH, які не пов'язані з FTP, включають SFTP та SCP; у кожному з них весь обмін (облікові дані та дані файлів) завжди захищена протоколом SSH.

2. Завдання на роботу.

2.1. Створити закритий ключ та запит на сертифікат для доменного імені, яке відповідає варіанту завдання. Підписати запит на сертифікат, використовуючи сертифікат і закритий ключ центру сертифікації (CA), створеного в лабораторній роботі № 2.

2.2. Встановити та налаштувати файловий сервер, який реалізує протоколи FTP, FTPS та SFTP та відповідає наступним вимогам:

- анонімний доступ (користувач ftp або anonymous);
- в домашньому каталозі анонімного користувача створені 2 каталоги: pub та incoming, каталог pub має права тільки читання, каталог incoming має права на читання та запис;
- створені користувачі з іменами, доступом та правами, які відповідають варіанту завдання;
- для створених користувачів забезпечується доступ по протоколам, які відповідають варіанту завдання.

2.3. Виконати аналіз протокольного обміну між клієнтом та сервером під час автентифікації та пересилки файлів.

2.4. Рекомендується використовувати наступне програмне забезпечення:

- FTP-сервер: vsftpd + openssh або proftpd;
- FTP-клієнт: lftp.

2.5. Для перевірки роботи файлового серверу та аналізу протокольного обміну рекомендується використовувати утиліти: ftp, telnet, netcat, openssl, lftp.

2.6. Додати запис типу A для доменного імені файлового серверу в DNS. Додати сертифікат власного CA у список довірених на вузлі, де запускається FTP-клієнт.

Варіанти завдання.

Варіант	**Ім'я FTP-серверу**	**Користувачі**			
		Ім'я	**Протокол**	**Доступ**	**Права**
1	ftp.letter.net	alpha	FTP,FTPS	до всього дерева каталогів	RW
		beta	SFTP	тільки домашній каталог	R
		gamma	FTP,FTPS,SFTP	тільки домашній каталог	RW
		delta	SFTP	до всього дерева каталогів	RW
		omega	FTP,FTPS	тільки домашній каталог	RW

Варіант	Ім'я FTP-серверу	Користувачі			
		Ім'я	Протокол	Доступ	Права
2	ftp.planet.edu	mercury	FTP,FTPS	тільки домашній каталог	R
		venus	SFTP	до всього дерева каталогів	R
		earth	FTP,FTPS,SFTP	тільки домашній каталог	RW
		saturn	SFTP	тільки домашній каталог	R
		jupiter	FTP,FTPS	до всього дерева каталогів	RW
3	ftp.cat.com	tiger	FTP,FTPS	до всього дерева каталогів	R
		lion	SFTP	тільки домашній каталог	RW
		lynx	FTP,FTPS,SFTP	тільки домашній каталог	RW
		leopard	SFTP	до всього дерева каталогів	RW
		jaguar	FTP,FTPS	тільки домашній каталог	R
4	ftp.flower.org	rose	FTP,FTPS	тільки домашній каталог	RW
		gerbera	SFTP	до всього дерева каталогів	RW
		tulip	FTP,FTPS,SFTP	тільки домашній каталог	R
		aster	SFTP	тільки домашній каталог	RW
		peony	FTP,FTPS	до всього дерева каталогів	R
5	ftp.linux.net	ubuntu	FTP,FTPS	до всього дерева каталогів	RW
		debian	SFTP	тільки домашній каталог	R
		centos	FTP,FTPS,SFTP	тільки домашній каталог	R
		gentoo	SFTP	до всього дерева каталогів	R
		fedora	FTP,FTPS	тільки домашній каталог	RW
6	ftp.color.edu	red	FTP,FTPS	тільки домашній каталог	R
		green	SFTP	до всього дерева каталогів	RW
		blue	FTP,FTPS,SFTP	тільки домашній каталог	RW
		black	SFTP	тільки домашній каталог	R
		white	FTP,FTPS	до всього дерева каталогів	RW
7	ftp.metal.com	gold	FTP,FTPS	до всього дерева каталогів	RW
		silver	SFTP	тільки домашній каталог	RW
		iron	FTP,FTPS,SFTP	тільки домашній каталог	R
		copper	SFTP	до всього дерева каталогів	R
		zinc	FTP,FTPS	тільки домашній каталог	RW

Варіант	Ім'я FTP-серверу	Користувачі			
		Ім'я	Протокол	Доступ	Права
8	ftp.capital.org	london	FTP,FTPS	тільки домашній каталог	RW
		tokyo	SFTP	до всього дерева каталогів	R
		paris	FTP,FTPS,SFTP	тільки домашній каталог	RW
		rome	SFTP	тільки домашній каталог	R
		berlin	FTP,FTPS	до всього дерева каталогів	RW
9	ftp.currency.net	dollar	FTP,FTPS	до всього дерева каталогів	R
		dinar	SFTP	тільки домашній каталог	R
		lira	FTP,FTPS,SFTP	тільки домашній каталог	RW
		peso	SFTP	до всього дерева каталогів	RW
		real	FTP,FTPS	тільки домашній каталог	R
10	ftp.river.edu	nile	FTP,FTPS	тільки домашній каталог	RW
		amazon	SFTP	до всього дерева каталогів	RW
		congo	FTP,FTPS,SFTP	тільки домашній каталог	RW
		amur	SFTP	тільки домашній каталог	R
		mekong	FTP,FTPS	до всього дерева каталогів	R
11	ftp.fruit.com	apple	FTP,FTPS	до всього дерева каталогів	RW
		orange	SFTP	тільки домашній каталог	R
		grape	FTP,FTPS,SFTP	тільки домашній каталог	R
		banana	SFTP	до всього дерева каталогів	R
		lemon	FTP,FTPS	тільки домашній каталог	RW
12	ftp.digit.org	one	FTP,FTPS	тільки домашній каталог	RW
		two	SFTP	до всього дерева каталогів	R
		three	FTP,FTPS,SFTP	тільки домашній каталог	R
		four	SFTP	тільки домашній каталог	RW
		five	FTP,FTPS	до всього дерева каталогів	RW
13	ftp.month.net	march	FTP,FTPS	до всього дерева каталогів	R
		april	SFTP	тільки домашній каталог	RW
		may	FTP,FTPS,SFTP	тільки домашній каталог	RW
		june	SFTP	до всього дерева каталогів	RW
		july	FTP,FTPS	тільки домашній каталог	R

Варіант	Ім'я FTP-серверу	Користувачі			
		Ім'я	Протокол	Доступ	Права
14	ftp.name.edu	maria	FTP,FTPS	тільки домашній каталог	RW
		tomas	SFTP	до всього дерева каталогів	R
		tereza	FTP,FTPS,SFTP	тільки домашній каталог	RW
		stefan	SFTP	тільки домашній каталог	R
		sara	FTP,FTPS	до всього дерева каталогів	RW
15	ftp.country.com	france	FTP,FTPS	до всього дерева каталогів	RW
		china	SFTP	тільки домашній каталог	RW
		spain	FTP,FTPS,SFTP	тільки домашній каталог	R
		italy	SFTP	до всього дерева каталогів	RW
		germany	FTP,FTPS	тільки домашній каталог	R

3. Контрольні питання.

3.1. Протокол FTP призначення та особливості роботи.

3.2. Пасивний та активний режими роботи FTP-сервера.

3.3. Команди протоколу FTP.

3.4. Протокол FTPS (FTP зверху SSL/TLS).

3.5. Протокол SFTP.

4. Література.

https://web.archive.org/web/20210410193501/http://wiki.gis.com/wiki/index.php/File_Transfer_Protocol

RFC959 https://tools.ietf.org/html/rfc959

SSH File Transfer Protocol https://tools.ietf.org/html/draft-ietf-secsh-filexfer-13

Лабораторна робота № 4

Мережна файлова система (NFS)

1. Короткі теоретичні відомості.

1.1. Версії NFS.

Протокол NFS розроблений компанією Sun Microsystems.

NFSv1 була розроблена в 1989 році і була експериментальною, працювала на протоколі UDP. Версія 1 описана в RFC 1094.

NFSv2 була випущена в тому ж 1989 році описувалася тим же RFC 1094 і так само базувалася на протоколі UDP.

NFSv3 доопрацьована в 1995 році і описана в RFC 1813. Основними нововведеннями третьої версії стало підтримка файлів великого розміру, додана підтримка протоколу TCP.

NFSv4 доопрацьована в 2000 році і описана в RFC 3010, в 2003 році переглянута і описана в RFC 3530. Четверта версія включила в себе поліпшення продуктивності, підтримку різних засобів.

NFSv4.1 була схвалена IESG в 2010 році, і отримала номер RFC 5661. Важливим нововведенням версії 4.1, є специфікація pNFS - Parallel NFS.

1.2. Відповідність NFS моделі ISO OSI.

Рівні ISO OSI	**Протоколи NFS**
Прикладний	NFS
Представлення	XDR
Сеансовий	RPC
Транспортний	UDP/TCP
Мережний	IP
Канальний	IEEE 802.3
Фізичний	

1.3. Цілі створення NFS.

Потенційна підтримка різних операційних систем (не тільки UNIX), щоб сервери і клієнти NFS можливо було б реалізувати в різних операційних системах.

Протокол повинен бути незалежним від будь-яких апаратних засобів.

Повинні бути реалізовані прості механізми відновлення у випадку відмов сервера або клієнта.

Додатки повинні мати прозорий доступ до виділених файлів без використання спеціальних імен або бібліотек і без перекомпіляції.

Для UNIX-клієнтів повинна підтримуватися семантика UNIX.

Продуктивність NFS повинна бути порівнянна з продуктивністю локальних дисків.

Реалізація не повинна бути залежною від транспортних засобів.

1.4. Компоненти NFS.

Реалізація NFS складається з декількох компонентів. Деякі з них локалізовані або на сервері, або на клієнті, а деякі використовуються і на обох сторонах з'єднання. Деякі компоненти не потрібні для забезпечення основних функціональних можливостей, але становлять частину розширеного інтерфейсу NFS.

Протокол NFS визначає набір запитів (операцій), які можуть бути спрямовані клієнтом до сервера, а також набір аргументів і значень, які повертаються для кожного з цих запитів.

Протокол віддаленого виклику процедур (RPC) визначає формат всіх взаємодій між клієнтом і сервером. Кожен запит NFS посилається як пакет RPC.

Зовнішнє представлення даних (XDR - External Data Representation) забезпечує машинно-незалежний метод кодування даних для пересилання через мережу. Всі запити RPC використовують кодування XDR для передачі даних. XDR і RPC використовуються для реалізації багатьох інших сервісів, крім NFS.

Програмний код сервера NFS відповідає за обробку всіх запитів клієнта і забезпечує доступ до експортованих файлових систем. Програмний код клієнта NFS реалізує всі звернення клієнтської системи до віддалених файлів шляхом посилки серверу одного або декількох запитів RPC.

Протокол монтування визначає семантику монтування та відключення файлових систем NFS. NFS використовує кілька фонових процесів-демонів. На сервері набір демонів nfsd очікує запити від клієнтів NFS і відповідає на них. Демон mountd обробляє запити монтування. На клієнті набір демонів biod обробляє асинхронний ввод-вивід блоків файлів NFS.

Менеджер блокувань мережі (NLM - Network Lock Manager) і монітор стану мережі (NSM - Network Status Monitor) разом забезпечують засоби для блокування файлів в мережі. Ці сервіси, хоча формально не пов'язані з NFS, можна знайти в більшості реалізацій NFS. Вони забезпечують функції, не можливі в базовому протоколі. NLM і NSM реалізують функціонування сервера за допомогою демонів lockd і statd, відповідно.

1.5. Основні процедури NFS.

lookup (dirfh, name) повертає (fh, attr).
create (dirfh, name, attr) повертає (newfh, attr).
remove (dirfh, name) повертає (status).
read (fh, offset, count) повертає (attr, data).
write (fh, offset, count, data) повертає (attr).

1.6. Налаштування сервера NFS.

/etc/exports - основний конфігураційний файл, який зберігає в собі конфігурацію експортованих каталогів.

/var/lib/nfs/xtab - містить список каталогів, змонтованих віддаленими клієнтами.

/var/lib/nfs/etab - список каталогів, які можуть бути змонтовані віддаленими системами із зазначенням всіх параметрів експортованих каталогів.

/var/lib/nfs/rmtab - список каталогів, що не відключені в даний момент.

/proc/fs/nfsd - спеціальна файлова система (ядро 2.6) для управління NFS сервером.

/proc/net/rpc - містить "сиру" (raw) статистику, яку можна отримати за допомогою nfsstat.
/var/run/portmap_mapping - інформація про зареєстровані в RPC сервіси.

1.7. Файл /etc/exports.

Файл /etc/exports є єдиним файлом, що вимагає редагування для налаштування NFS-сервера, файл управляє наступними параметрами:

- Які клієнти можуть звертатися до файлів на сервері.
- До яких ієрархій каталогів на сервері може звертатися кожен клієнт.
- Як імена користувачів клієнтів будуть відображатися на локальні імена користувачів.

Кожен рядок файлу exports мають такий вигляд:

```
точка_експорта клієнт1 (опції) [клієнт2 (опції) ...]
```

1.8. Опції експорту ієрархій каталогів.

auth_nlm (no_auth_nlm) або secure_locks (insecure_locks) - вказує, що сервер повинен вимагати аутентифікацію запитів на блокування (за допомогою протоколу NFS Lock Manager).

nohide (hide) - якщо сервер експортує дві ієрархії каталогів, при цьому одна примонтована в іншу, то клієнту необхідно явно змонтувати другу (дочірню) ієрархію, інакше точка монтування дочірньої ієрархії буде виглядати як порожній каталог. Опція nohide призводить до появи другої ієрархії каталогів без явного монтування.

ro (rw) - дозволяє тільки запити на читання (запис). В кінцевому рахунку - можливість читати/записувати чи ні визначається на підставі прав файлової системи, при цьому сервер не здатний відрізнити запит на читання файлу від запиту на виконання, тому дозволяє читання, якщо у користувача є права на читання або виконання.

secure (insecure) - вимагає, щоб запити NFS надходили з захищених портів (<1024), щоб програма без прав root не могла монтувати ієрархію каталогів.

subtree_check (no_subtree_check) - Якщо експортується підкаталог файлової системи, але не вся файлова система, сервер перевіряє, чи знаходиться запитаний файл в експортованому. підкаталозі. Вимкнення перевірки зменшує безпеку, але збільшує швидкість передачі даних.

sync (async) - вказує, що сервер повинен відповідати на запити тільки після запису на диск змін, виконаних цими запитами. Опція async вказує серверу не чекати запису інформації на диск, що підвищує продуктивність, але знижує надійність.

wdelay (no_wdelay) - вказує серверу затримувати виконання запитів на запис, якщо очікується подальший запит на запис.

nosuid - забороняє виконувати setuid програми із змонтованого каталогу.

nodev (no device) - забороняє використовувати в якості пристроїв символьні та блочні спеціальні файли.

lock (nolock) - дозволяє блокування NFS (за замовчуванням). nolock відключає блокування NFS (працює програмне забезпечення демон lockd) і зручна при роботі зі старими серверами, які не підтримують блокування NFS.

mounthost = ім'я - ім'я вузла, на якому запущений демон монтування NFS - mountd.

mountport = n - порт, який використовується демоном mountd.

port = n - порт, який використовується для підключення до NFS сервера (за замовчуванням 2049, якщо демон rpc.nfsd не зареєстрований на RPC-сервері). Якщо n = 0 (за замовчуванням), то NFS посилає запит до portmap на сервері, щоб визначити порт.

rsize = n (read block size - розмір блоку читання) - кількість байтів, що читаються за один раз з NFS-сервера (стандартно - 4096).

wsize = n (write block size - розмір блоку записи) - кількість байтів, що записуються за один раз на NFS-сервер (стандартно - 4096).

tcp або udp - для монтування NFS використовувати протокол TCP або UDP відповідно.

bg - при втраті доступу до сервера, повторювати спроби в фоновому режимі, щоб не блокувати процес завантаження системи.

fg - при втраті доступу до сервера, повторювати спроби в пріоритетному режимі. Даний параметр може заблокувати процес завантаження системи повтореннями спроб монтування. З цієї причини параметр fg використовується переважно при налагодженні.

2. Завдання на роботу.

2.1. Встановити та налаштувати файловий сервер, який реалізує мережну файлову систему NFS та відповідає наступним вимогам:

- створені каталоги: /nfs/public, /nfs/private, /nfs/incoming;
- каталог /nfs/public експортується в режимі тільки читання;
- каталог /nfs/private експортується в режимі читання та запису і тільки для вузла, на якому працює NFS-клієнт, запити від всіх користувачів на клієнті відображаються на заданого варіантом користувача на сервері, який має повні права в цьому каталозі;
- каталог /nfs/incoming експортується в режимі читання та запису для вузлів, які знаходяться в тій же мережі, що і NFS-клієнт, запити від користувача root на клієнті відображаються на заданого варіантом користувача на сервері, запити інших користувачів клієнта залишаються без змін на сервері, права на каталог встановлені відповідно варіанту завдання.

2.2. Виконати аналіз протокольного обміну між клієнтом та сервером під час монтування файлової системи та пересилки файлів.

2.3. Рекомендується використовувати наступне програмне забезпечення:

- NFS-сервер: nfs-kernel-server;
- NFS-клієнт: nfs-common.

2.4. Для перевірки роботи файлового серверу та аналізу протокольного обміну рекомендується використовувати утиліти: nfsstat, showmount, mount, tcpdump.

Варіанти завдання.

Варіант	**Ім'я NFS-серверу**	**Відображення користувачів**		**Права на каталог /nfs/incoming**			
		/nfs/private	**/nfs/incoming**	**RWX**	**-**	**RX**	**RWX**
1	nfs.letter.net	alpha	beta	gamma	delta	omega	beta
2	nfs.planet.edu	mercury	venus	earth	saturn	jupiter	venus
3	nfs.cat.com	tiger	lion	lynx	leopard	jaguar	lion
4	nfs.flower.org	rose	gerbera	tulip	aster	peony	gerbera
5	nfs.linux.net	ubuntu	debian	centos	gentoo	fedora	debian
6	nfs.color.edu	red	green	blue	black	white	green

Варіант	Ім'я NFS-серверу	Відображення користувачів		Права на каталог /nfs/incoming			
		/nfs/private	/nfs/incoming	RWX	-	RX	RWX
7	nfs.metal.com	gold	silver	iron	copper	zinc	silver
8	nfs.capital.org	london	tokyo	paris	rome	berlin	tokyo
9	nfs.currency.net	dollar	dinar	lira	peso	real	dinar
10	nfs.river.edu	nile	amazon	congo	amur	mekong	amazon
11	nfs.fruit.com	apple	orange	grape	banana	lemon	orange
12	nfs.digit.org	one	two	three	four	five	two
13	nfs.month.net	march	april	may	june	july	april
14	nfs.name.edu	maria	tomas	tereza	stefan	sara	tomas
15	nfs.country.com	france	china	spain	italy	germany	china

3. Контрольні питання.
3.1. Відповідність NFS моделі ISO OSI.
3.2. Архітектура NFS.
3.3. Призначення та функції компонент NFS: mountd, nfsd, rpcbind (portmapper).
3.4. Процедура монтування файлової системи.

4. Література.
RFC7530 https://tools.ietf.org/html/rfc7530
https://ru.wikipedia.org/wiki/Network_File_System

Лабораторна робота № 5

Сервіс доступу до файлів SMB/CIFS

1. Короткі теоретичні відомості.

1.1. Сервіси SMB/CIFS.

Управління сесіями. Створення, підтримка і розрив логічного каналу між робочою станцією і мережними ресурсами файлового сервера.

Файловий доступ. Робоча станція може звернутися до файл-серверу з запитами на виконання типових файлових операцій (відкриття файлу, читання даних і т.п.).

Сервіс друку. Робоча станція може ставити файли в чергу для друку на сервері і отримувати інформацію про чергу друку.

Сервіс повідомлень. SMB підтримує просту передачу адресних і широкомовних повідомлень по локальній мережі.

1.2. Місце в стеку протоколів TCP/IP.

Рівні ISO OSI	**Протоколи SMB**
Прикладний	SMB
Представлення	
Сеансовий	NetBIOS
Транспортний	UDP/TCP
Мережний	IP
Канальний	IEEE 802.3
Фізичний	

1.3. Версії протоколу SMB.

У 1983 році компанія IBM (Баррі Фейгенбаум) розробила протокол SMB.

IBM розробила API для мережної взаємодії між вузлами в локальній мережі - NetBIOS API.

У поєднанні з транспортним протоколом, який повинен називатися NetBEUI - NetBIOS.

У 1988 році Microsoft і Intel змінили протокол під назвою «Базовий протокол», який використовував NetBIOS API для доставки пакетів протоколу CIFS.

У 1996 році SMB був перейменований в CIFS з новими функціями.

SMB/CIFS - NetBIOS поверх TCP, використовувався Microsoft до Windows 2000.

У Windows Vista з'явилася нова версія протоколу - SMB 2.0. Протокол був значно спрощений (у SMB було понад 100 команд, а в SMB 2 - всього 19).

У Windows 8 з'явилася нова версія протоколу - SMB 3.0.

SMB 3.0.2 був представлений з Windows 8.1 та Windows Server 2012 R2; у цих та пізніших випусках попередню версію SMB 1 можна відключити для підвищення безпеки.

SMB 3.1.1 був представлений з Windows 10 та Windows Server 2016. Ця версія підтримує шифрування AES-128 GCM на додаток до шифрування CCM AES-128, доданого в SMB3, та реалізує перевірку цілісності перед автентифікацією за допомогою хешу SHA-512. SMB 3.1.1 також робить безпечні переговори обов'язковими при підключенні до клієнтів за допомогою SMB 2.x і вище.

1.4. Протокол NetBIOS.

NetBIOS працює поверх безлічі транспортних протоколів, але на сьогодні основні реалізації працюють поверх стеку TCP/IP.

Документи RFC1001 і RFC1002 описують роботу NetBIOS поверх TCP/UDP (NBT). RFC1001 описує концепцію і методи. RFC1002 містить детальну специфікацію. У цих документах описані 3 основних служби:

- імен;
- сесій;
- датаграм.

1.5. Служба імен.

Складається з реєстрації і запитів імен. NetBIOS-ім'я - зрозумілий для людини ідентифікатор комп'ютера. Також як і в системі імен DNS NetBIOS-ім'я повинне бути зареєстроване в базі і має перетворюватися в IP-адресу для транспортування пакетів. DNS-імена і IP-адреси статично закріплені за комп'ютером, тоді як NetBIOS-імена реєструються динамічно під час завантаження операційної системи. Працює з використанням широкомовлення або сервера імен NetBIOS (NBNS або WINS). Комп'ютери для реєстрації та перегляду імен можуть використовувати:

- b-node;
- p-node;
- m-node;
- h-node.

1.6. Типи комп’ютерів NetBIOS.

Роль	Значення
b-node	Використовує лише широкомовну реєстрацію та перегляд імен.
p-node	Використовує тільки реєстрацію точка-точка і перегляд імен.
m-node	Використовує широкомовну реєстрацію. Якщо вона успішна, він повідомляє про це сервер NBNS. Використовує широкомовний метод для перегляду імен; використовує NBNS сервер, якщо метод широкомовних запитів невдалий.
h-node (гібрид)	Використовує NBNS сервер для реєстрації та перегляду імен; використовує широкомовні запити, якщо сервер NBNS не доступний.

1.7. Правила використання NetBIOS-імен.

NetBIOS-ім'я може використовуватися одним комп'ютером (unique name) або декількома комп'ютерами. В останньому випадку ім'я належить до групи комп'ютерів (group name).

NetBIOS імена утворюють плоский простір імен без ієрархії, як в системі DNS. Це означає, що кожен комп'ютер повинен використовувати унікальну послідовність символів, що утворюють його ім'я. На відміну від системи DNS, де 2 і більше імені можуть збігатися, якщо вони знаходяться в різних доменах.

NetBIOS-імена повинні складатися тільки з наступних символів:

```
a-z
A-Z
0-9
! @ # $ % ^ & ( ) - ' { } . ~
```

NetBIOS-ім'я може складатися максимум з 15 символів для ідентифікації ресурсу, 16-й символ показує тип ресурсу.

1.8. Типи ресурсів NetBIOS.

Ім’я ресурсу	**Значення в полі тип**
Standard Workstation Service	00
Messenger Service (WinPopup)	03
RAS Server Service	06
Domain Master Browser Service (associated with primary domain controller)	1B
Master Browser name	1D
Fileserver (including printer server)	20
Network Monitor Agent	BE

1.9. Служба сесій.

Сесія забезпечує відмовостійкий зі збереженням послідовності відправки обмін повідомленнями між парою NetBIOS застосунків. Використовується 139 порт протоколу TCP для емуляції функціонування сесії. SMB використає цю службу для відправки команд, наприклад, для роботи з файлами або принтерами.

1.10. Служба датаграм.

Для роботи SMB досить тільки служб сесій і імен, але для пошуку комп'ютерів в мережі (функція перегляду) необхідна служба датаграм. Функція перегляду мережі не є частиною протоколу SMB. Служба датаграм забезпечує ненадійний, без збереження порядку надходження і без встановлення з'єднання сервіс обміну повідомленнями. Для роботи служби датаграм використовується порт 138 протоколу UDP. SMB можуть працювати поверх TCP без NetBIOS. DNS і доменні імена можуть використовуватися для забезпечення сервісу імен, служба сесій може працювати прямо поверх TCP, служба датаграм - прямо поверх UDP.

2. Завдання на роботу.

2.1. Встановити та налаштувати файловий сервер, який реалізує протоколи SMB/CIFS та відповідає наступним вимогам:

- створені користувачі з іменами, доступом та правами, які відповідають варіанту завдання;
- гостьовий доступ (без проходження автентифікації) до каталогів з правами, які відповідають варіанту завдання;
- доступ до домашнього каталогу для автентифікованого користувача з правами, які відповідають варіанту завдання;
- доступ до прихованого каталогу (невидимий при перегляді списку спільних ресурсів штатними засобами) з правами, які відповідають варіанту завдання.

2.3. Виконати аналіз протокольного обміну між клієнтом та сервером під час автентифікації та пересилки файлів.

2.4. Рекомендується використовувати наступне програмне забезпечення:

- SMB-сервер: samba;
- SMB-клієнт: smbclient.

2.5. Для перевірки роботи файлового серверу та аналізу протокольного обміну рекомендується використовувати утиліти: nbtscan, nmblookup, smbtree, nmap.

Варіанти завдання.

Варіант	**Ім'я користувача**	**Права доступу до каталогів**			
		Прихований	**Домашній**	**public**	**incoming**
1	alpha	R	RW	R	RW
	beta	RW	RW	R	-
	gamma	-	R	R	RW
	гість	R	-	R	RW
2	mercury	-	RW	-	-
	venus	RW	R	R	RW
	saturn	R	RW	R	RW
	гість	RW	-	-	RW
3	tiger	RW	RW	R	RW
	lion	-	RW	-	RW
	lynx	R	-	R	-
	гість	-	-	R	-
4	rose	R	R	R	RW
	gerbera	RW	RW	-	-
	aster	-	R	-	RW
	гість	R	-	R	-

Варіант	Ім'я користувача	Права доступу до каталогів			
		Прихований	Домашній	public	incoming
5	ubuntu	R	RW	R	RW
	debian	RW	RW	R	-
	centos	-	R	R	RW
	гість	R	-	-	RW
6	red	RW	RW	R	RW
	green	-	RW	-	RW
	blue	R	-	R	-
	гість	RW	-	R	RW
7	gold	R	RW	R	RW
	silver	RW	RW	R	-
	iron	-	R	R	RW
	гість	RW	-	-	-
8	london	R	R	R	RW
	tokyo	RW	RW	-	-
	paris	-	R	-	RW
	гість	-	-	-	RW
9	dollar	R	RW	R	RW
	dinar	RW	RW	R	-
	lira	-	R	R	RW
	гість	RW	-	R	-
10	nile	-	RW	-	-
	amazon	RW	R	R	RW
	congo	R	RW	R	RW
	гість	RW	-	R	-
11	apple	RW	RW	R	RW
	orange	-	RW	-	RW
	grape	R	-	R	-
	гість	R	-	-	-
12	one	R	R	R	RW
	two	RW	RW	-	-

Варіант	Ім'я користувача	Права доступу до каталогів			
		Прихований	Домашній	public	incoming
	three	-	R	-	RW
	гість	-	-	R	RW
13	march	R	RW	R	RW
	april	RW	RW	R	-
	may	-	R	R	RW
	гість	RW	-	-	RW
14	maria	-	RW	-	-
	tomas	RW	R	R	RW
	tereza	R	RW	R	RW
	гість	R	-	R	-
15	france	RW	RW	R	RW
	spain	-	RW	-	RW
	italy	R	-	R	-
	гість	RW	-	R	RW

3. Контрольні питання.

3.1. Стек NetBIOS/SMB.

3.2. NetBIOS імена та типи ресурсів.

3.3. Сервіс імен NetBIOS. Реєстрація та перегляд імен NetBIOS. Типи комп'ютерів NetBIOS.

3.4. Вибори головного переглядача.

3.5. Служби датаграм та сесій.

4. Література.

RFC1001 https://tools.ietf.org/html/rfc1001
RFC1002 https://tools.ietf.org/html/rfc1002
[MS-CIFS] http://msdn.microsoft.com/en-us/library/ee442092.aspx
[MS-SMB] http://msdn.microsoft.com/en-us/library/cc246231.aspx
[MS-SMB2] http://msdn.microsoft.com/en-us/library/cc246482.aspx

Лабораторна робота № 6

Web сервіс

1. Короткі теоретичні відомості.

1.1. World Wide Web.

Всесвітня павутина є настільки розповсюдженою і зробила Інтернет доступним для такої кількості людей, що іноді вона здається синонімом Інтернету. Насправді, розробка системи, яка стала Веб, почалася приблизно в 1989 році, задовго після того, як Інтернет став широко розповсюдженою мережею. Початкова мета Інтернету - знайти спосіб організувати та отримати інформацію, спираючись на уявлення про гіпертекстові - взаємопов’язані документи, які існували принаймні з 1960-х років. Основна ідея гіпертексту полягає в тому, що один документ може посилатися на інший документ, а протокол HTTP та мова документів HTML були розроблені для досягнення цієї мети.

Один із корисних способів уявлення про Інтернет - це набір клієнтів та серверів, які співпрацюють, усі вони говорять однією мовою: HTTP. Більшість людей мають доступ до Інтернету через графічну клієнтську програму або веб браузер, наприклад Safari, Chrome, Firefox або Internet Explorer.

Якщо необхідно організувати інформацію у систему пов’язаних документів або об’єктів, потрібно мати можливість отримати один документ, щоб розпочати процес. Отже, будь-який веббраузер має функцію, яка дозволяє користувачеві отримати об’єкт, відкривши URL-адресу. Уніфіковані локатори ресурсів (URL-адреси) настільки знайомі більшості з нас, що легко забути про те, що їх не було вічно. Вони надають інформацію, яка дозволяє розміщувати об’єкти в Інтернеті, і виглядають так:

```
http://comsys.kpi.ua/index.html
```

Якщо відкрити цю URL-адресу, то веббраузер встановить TCP-з'єднання з вебсервером на вузлі з ім'ям comsys.kpi.ua і отримає та відобразить файл з назвою index.html. Більшість файлів у Інтернеті містять зображення та текст, а багато з них мають інші об’єкти, такі як аудіо- та відеоматеріали, фрагменти коду тощо. Вони також часто містять URL-адреси, які вказують на інші файли, які можуть бути розташовані на інших вузлах, що є основою "гіпертексту" в HTTP та HTML. Веббраузер має можливість розпізнавання URL-адрес (часто виділяючи або підкреслюючи певний текст), можна попросити браузер відкрити їх. Ці вбудовані URL-адреси називаються гіпертекстовими посиланнями. Коли необхідно веб-браузеру відкрити одну з таких вбудованих URL-адрес (наприклад, вказуючи та натискаючи на ньому мишею), він відкриє нове з'єднання, а також отримає та відображає новий файл. Це називається переходити за посиланням. Таким чином, стає дуже легко переходити з одного вузла на інший по мережі, переходячи за посиланнями на різну інформацію. Основа гіпертекстової системи - це спосіб вставити посилання в документ і дозволити користувачу перейти за цим посиланням, щоб отримати інший документ.

Якщо попросити браузер переглянути сторінку, то браузер (клієнт) отримує сторінку з сервера за допомогою протоколу HTTP, що працює через TCP. Як і SMTP, HTTP - це

текстоорієнтований протокол. По суті, HTTP - це протокол запиту/відповіді, де кожне повідомлення має загальний формат:

```
START_LINE <CRLF>
MESSAGE_HEADER <CRLF>
<CRLF>
MESSAGE_BODY <CRLF>
```

де, <CRLF> позначає повернення каретки+подачу рядка. Перший рядок (START_LINE) вказує на тип повідомлення: запит або відповідь. По суті, він визначає “віддалену процедуру”, яка підлягає виконанню (у разі повідомлення із запитом), або статус запиту (у разі повідомлення з відповіддю). Наступний набір рядків визначає список параметрів, які відповідають запиту чи відповіді. Ці рядки MESSAGE_HEADER мають нуль або більше рядків. Список закінчується порожнім рядком. Кожен з них виглядає як рядок заголовка в повідомленні електронної пошти. HTTP визначає багато можливих типів заголовків, деякі з яких стосуються запитів повідомлень, інші - повідомлень відповідей, а інші - даних, що містяться в тілі повідомлення. Після порожнього рядка йде вміст запитуваного повідомлення (MESSAGE_BODY); у цій частині повідомлення сервер розміщує запитувану сторінку під час відповіді на запит, і зазвичай вона порожня для повідомлень із запитом.

Чому HTTP працює через TCP? Розробникам не потрібно було цього робити, але TCP дійсно добре відповідає потребам HTTP, зокрема, забезпечуючи надійну доставку (кому потрібна вебсторінка з відсутніми даними?), контроль потоку та контроль перевантажень. Однак є кілька проблем, які можуть виникнути при побудові протоколу запиту/відповіді поверх TCP, особливо якщо ігнорувати тонкощі взаємодії між протоколами застосунку та транспортного рівня.

1.2. HTTP-запит.

Перший рядок повідомлення із запитом HTTP визначає три речі: операцію, яку потрібно виконати, вебсторінку, на якій операція має бути виконана, та версію HTTP, що використовується. Хоча протокол HTTP визначає широкий набір можливих операцій із запитами, включаючи операції запису, які дозволяють розмістити вебсторінку на сервері, дві найпоширеніші операції - це GET (запит вмісту вказаного в URL ресурсу) та HEAD (запит метаданих ресурсу, вказаного в URL). Перший, очевидно, використовується, коли браузер хоче отримати та відобразити вебсторінку. Останнє використовується для перевірки дійсності гіпертекстового посилання або для того, щоб перевірити, чи була змінена певна сторінка з моменту останнього її отримання браузером. Повний набір операцій узагальнено у Таблиці 1.

Наприклад:

```
GET http://comsys.kpi.ua/index.html HTTP/1.1
```

Каже, що клієнт хоче, щоб сервер на вузлі comsys.kpi.ua повернув сторінку з назвою index.html. У цьому прикладі використовується абсолютна URL-адреса. Також можна використовувати відносний ідентифікатор та вказати ім’я вузла в одному з рядків MESSAGE_HEADER, наприклад:

```
GET index.html HTTP/1.1
Host: comsys.kpi.ua
```

Тут Host є одним із можливих полів MESSAGE_HEADER. Одним з найцікавіших з них є If-Modified-Since, що дає клієнту можливість умовно запитувати вебсторінку. Сервер повертає сторінку лише в тому випадку, якщо вона була змінена з часу, зазначеного у цьому рядку заголовка.

Таблиця 1. Методи запитів HTTP.

Метод	**Опис**
OPTIONS	Визначення можливостей веб-сервера
GET	Запит вмісту вказаного в URL ресурсу
HEAD	Запит метаданих ресурсу вказаного в URL
POST	Передачі даних заданому в URL ресурсу
PUT	Завантаження ресурсу на вказаний в запиті URL
DELETE	Видалення вказаного в URL ресурсу
TRACE	Трасування вказаного в URL ресурсу
CONNECT	Підключення до Web-сервера через проксі

1.3. HTTP-відповідь.

Як і повідомлення із запитами, повідомлення-відповіді починаються з того ж рядка START_LINE. У цьому випадку рядок вказує версію HTTP, яка використовується, тризначний код, що вказує на успішність запиту, та текстовий рядок із пояснення відповіді. Наприклад, START_LINE:

```
HTTP/1.1 202 Accepted
```

Вказує на те, що сервер зміг прийняти запит. При цьому:

```
HTTP/1.1 404 Not Found
```

Вказує, що сервер не зміг виконати запит, оскільки сторінку не знайдено. Існує п'ять загальних типів кодів відповідей, перша цифра коду вказує на його тип. Таблиця 2 містить п’ять типів кодів.

Таблиця 2. Коди відповідей HTTP.

Код	**Тип**	**Приклади причин**
1xx	Інформаційний	Запит отримано, процес триває
2xx	Успіх	Запит успішно прийнятий, зрозумілий і оброблений
3xx	Перенаправлення	Для виконання запиту необхідно вжити додаткових заходів
4xx	Помилка клієнта	Запит містить неправильний синтаксис або не може бути виконаний
5xx	Помилка серверу	Серверу не вдалося виконати правильний запит

Також подібно до повідомлень із запитами, повідомлення-відповіді можуть містити один або кілька рядків MESSAGE_HEADER. Ці рядки передають клієнту додаткову інформацію. Наприклад, рядок заголовка Location вказує, що запитувана URL-адреса доступна в іншому місці. Таким чином, якби, наприклад, веб-сторінка кафедри обчислювальної техніки перейшла з http://comsys.kpi.ua/index.html на https://comsys.kpi.ua/new/index.html, тоді сервер за оригінальною адресою може відповісти

```
HTTP/1.1 301 Moved Permanently
Location: https://comsys.kpi.ua/new/index.hml
```

У загальному випадку повідомлення-відповідь також міститиме запитувану сторінку. Ця сторінка є документом HTML, але оскільки вона може містити нетекстові дані (наприклад, зображення GIF), вона кодується за допомогою MIME. Деякі рядки MESSAGE_HEADER містять атрибути вмісту сторінки, включаючи термін дії (час, коли вміст вважається застарілим) та (час, коли вміст востаннє змінювався на сервері).

1.4. Уніфікований ідентифікатор ресурсу.

URL-адреси, які HTTP використовує як адреси, є одним із видів уніфікованого ідентифікатора ресурсу (URI). URI - це рядок символів, який ідентифікує ресурс, де ресурсом може бути все, що має ідентичність, наприклад: документ, зображення або послуга.

Формат URI дозволяє включати різні більш спеціалізовані види ідентифікаторів ресурсів у простір URI ідентифікаторів. Перша частина URI - це схема, яка називає певний спосіб ідентифікації певного виду ресурсу, наприклад mailto для адрес електронної пошти або file для імен файлів. Друга частина URI, відокремлена від першої частини двокрапкою, є частиною, специфічною для схеми. Це ідентифікатор ресурсу, що відповідає схемі в першій частині, як у URI:

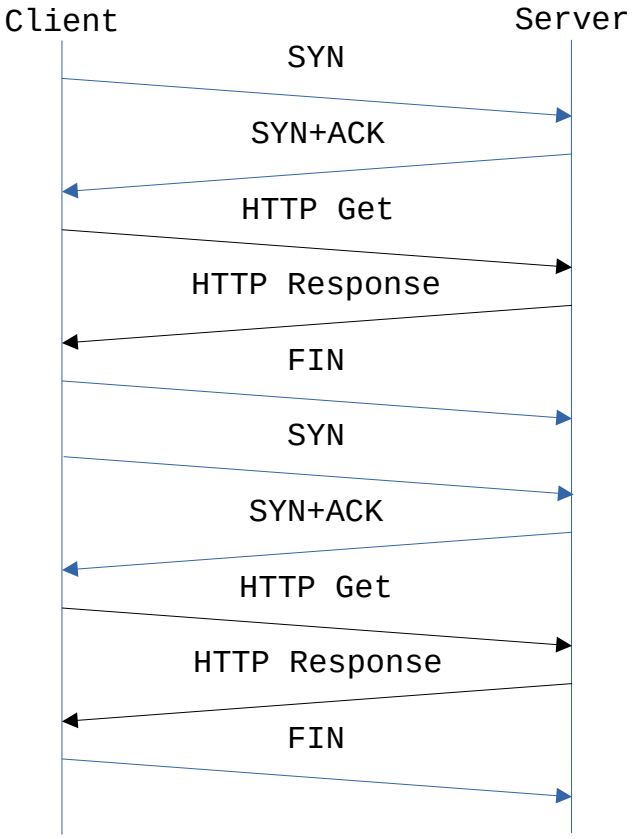
```
mailto:admin@comsys.kpi.ua та file:///C:/foo.html.
```

1.5. TCP з'єднання.

Початкова версія HTTP 1.0 використовувала окреме з'єднання TCP для кожного елемента даних, отриманого з сервера. Неважко зрозуміти, наскільки це був дуже неефективний механізм: необхідно встановлювати та завершувати з'єднання, навіть якщо клієнт хотів лише перевірити наявність останньої копії сторінки. Таким чином, отримання сторінки, що містить текст та десяток картинок або іншу дрібну графіку, призведе до встановлення та закриття 13 окремих TCP-з'єднань. На малюнку 1 показана послідовність подій для отримання сторінки, яка містить лише один вбудований об’єкт. Блакитні лінії вказують на повідомлення TCP, а чорні - на запити та відповіді HTTP. (Деякі з TCP ACK не відображаються, щоб уникнути захаращення зображення.) Ви можете побачити, що два періоди обертання пакету витрачаються на встановлення TCP-з'єднаня, а ще два (принаймні) - на отримання сторінки та зображення. Окрім впливу затримки, на сервері також є затримка обробки запиту для обробки додаткового встановлення та завершення з'єднання TCP.

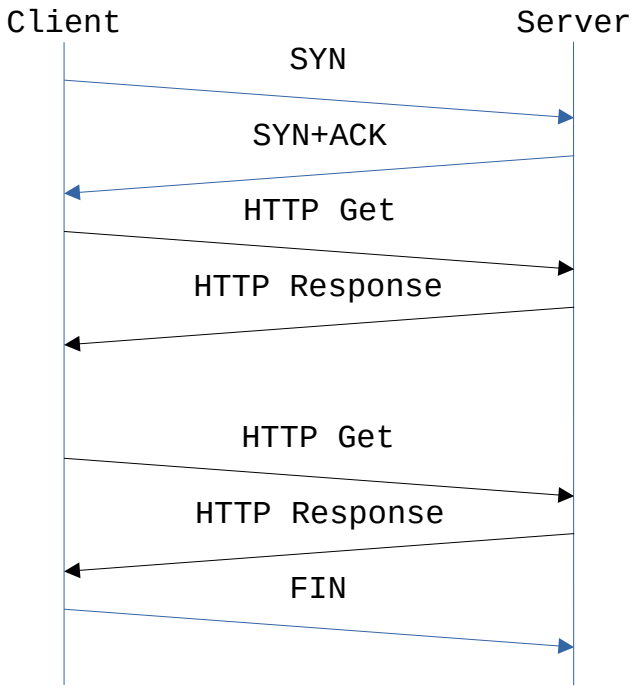
Щоб подолати цю ситуацію, версія HTTP 1.1 запровадила постійні з'єднання - клієнт і сервер можуть обмінюватися кількома повідомленнями запиту/відповіді через одне і те ж TCP-

з'єднання. Постійні з'єднання мають багато переваг. По-перше, вони, очевидно, усувають накладні витрати на встановлення з'єднання, тим самим зменшуючи навантаження на сервер, навантаження на мережу, викликане додатковими сегментами TCP, і затримку, яку бачить користувач. По-друге, оскільки клієнт може надсилати декілька повідомлень із запитами по одному з'єднанню TCP, механізм вікна перевантаження TCP може працювати більш ефективно. Це тому, що необов’язково проходити фазу повільного старту для кожної сторінки. На малюнку 2 показана транзакція з постійним з'єднанням.



Малюнок. HTTP 1.0

Однак постійне з'єднання також має недоліки. Проблема в тому, що ні клієнт, ні сервер не знають, як довго утримувати певне TCP-з'єднання відкритим. Це особливо важливо для сервера, який може попросити тримати з'єднання відкритими для тисяч клієнтів. Рішення полягає в тому, що сервер повинен витримати час і закрити з'єднання, якщо протягом певного періоду часу він не отримував запитів через з'єднання. Крім того, і клієнт, і сервер повинні спостерігати, чи інша сторона вирішила закрити з'єднання, і вони повинні використовувати цю інформацію як сигнал про те, що вони також повинні закрити свою сторону з'єднання. (Нагадаємо, що обидві сторони повинні закрити TCP-з'єднання до його повного припинення.) Занепокоєння з приводу цієї додаткової складності може бути однією з причин того, що постійне з'єднання не використовувалось з самого початку, але сьогодні загальновизнано, що переваги постійного з'єднання більше, ніж недоліки.



Малюнок. Поведінка HTTP 1.1 із постійним з’єднанням.

1.6 HTTP/2.

Хоча HTTP версії 1.1 все ще широко використовується, нова версія 2.0 була офіційно затверджена IETF у 2015 році. Відома як HTTP/2, нова версія зворотно сумісна з 1.1 (тобто вона приймає той самий синтаксис для полів заголовка, статусу коди та URI), але він додає дві нові функції.

По-перше, це зменшити обсяг службових даних, які вебсервер надсилає веббраузерам. Якщо подивитись на склад HTML на типовій вебсторінці, можна знайти безліч посилань на інші фрагменти (наприклад, зображення, сценарії, файли стилів), необхідні браузеру для відображення сторінки. Замість того, щоб змушувати клієнта запитувати ці фрагменти у наступних запитах, протокол HTTP/2 надає серверу можливість об’єднати необхідні ресурси та заздалегідь надіслати їх клієнту, не створюючи додаткову затримку при передачі запитів на ці ресурси. Ця функція поєднується з механізмом стиснення, який зменшує кількість байтів, які потрібно передати. Основна мета - мінімізувати затримку, яку відчуває кінцевий користувач з моменту натискання на гіперпосилання до повної візуалізації вибраної сторінки.

Другою великою перевагою HTTP/2 - є мультиплексування кількох запитів в одному TCP - з'єднанні. Це виходить за межі того, що підтримує версія HTTP 1.1 - дозволяючи послідовності запитів повторно використовувати TCP-з'єднання - дозволяючи цим запитам накладатися один на одний. Те, як це робить HTTP/2, має звучати знайомо: він визначає абстракцію каналу (технічно, канали називаються потоками), дозволяє декілька одночасних потоків бути активними в певний час (кожен з них має унікальний ідентифікатор потоку) і обмежує кожен потік одночасно до одного активного обміну запитами/відповідями.

1.7. Кешування.

Важливою стратегією впровадження, яка робить Інтернет більш корисним, є кешування вебсторінок. Кешування має багато переваг. З точки зору клієнта, сторінка, яку можна отримати з кешу поблизу, може завантажуватись набагато швидше, ніж якщо її потрібно отримати з віддаленого сервера. З точки зору сервера кеш зменшує навантаження на сервер.

Кешування може бути реалізовано в різних місцях. Наприклад, веббраузер користувача може кешувати нещодавно відкриті сторінки та просто відображати кешовану копію, якщо користувач знову відвідує ту саму сторінку. Як інший приклад, сайт може підтримувати єдиний кеш на рівні всього сайту. Це дозволяє користувачам скористатися сторінками, раніше завантаженими іншими користувачами. Ближче до середини мережі Інтернет постачальники послуг Інтернету (ISP) можуть кешувати сторінки. У другому випадку користувачі на вебсайті, швидше за все, знають, яка машина кешує сторінки від імені сайту, і вони налаштовують свої браузери для підключення безпосередньо до вузла кешування. Цей вузол іноді називають проксі. На противагу цьому, сайти, які підключаються до провайдера, ймовірно, не знають, що провайдер кешує сторінки. Буває, що запити HTTP, що надходять з різних сайтів, проходять через загальний маршрутизатор провайдера. Цей маршрутизатор може заглянути всередину повідомлення із запитом і подивитися на URL-адресу потрібної сторінки. Якщо сторінка є у кеші, він повертає її. Якщо ні, він пересилає запит на сервер і стежить за тим, щоб відповідь була передана в іншому напрямку. Після цього маршрутизатор зберігає копію в надії, що він зможе використати її для задоволення майбутнього запиту.

Незалежно від того, де сторінки кешуються, можливість кешування вебсторінок є досить важливою, оскільки HTTP був розроблений для полегшення роботи. Хитрість полягає в тому, що кеш повинен переконатися, що він не відповідає застарілою версією сторінки. Наприклад, сервер встановлює дату закінчення терміну дії (поле заголовка Expires) кожній сторінці, яку він надсилає клієнту (або кешу між сервером і клієнтом). Кеш пам’ятає цю дату і знає, що не потрібно повторно перевіряти сторінку кожного разу, коли її запитують, доки не закінчиться цей термін. Після закінчення цього часу (або якщо це поле заголовка не встановлено) кеш може використовувати операцію HEAD або умовну операцію GET (GET з рядком заголовка), щоб перевірити наявність останньої копії сторінки. Загалом, існує набір директив кешування, яким повинні підкорятися всі механізми кешування по ланцюжку запит/відповідь. Ці директиви визначають, чи можна документ кешувати, як довго він може кешуватися, наскільки свіжим повинен бути документ тощо.

2. Завдання на роботу.

2.1. Створити закритий ключ та запит на сертифікат для доменного імені, яке відповідає варіанту завдання. Підписати запит на сертифікат, використовуючи сертифікат і закритий ключ центру сертифікації (CA), створеного в лабораторній роботі № 2.

2.2. Встановити та налаштувати Web-сервер, який реалізує протоколи HTTP та HTTPS, виконує функції балансування навантаження для двох Web-серверів та відповідає наступним вимогам:

- доменне ім’я сервера балансування навантаження відповідає варіанту завдання;
- сервер балансування навантаження слухає порти для протоколів HTTP та HTTPS, номери портів відповідають варіанту завдання;
- доступ до сервера балансування навантаження мають тільки користувачі, що пройшли автентифікацію, імена яких відповідають варіанту завдання;
- метод балансування навантаження відповідає варіанту завдання;

2.3. Встановити та налаштувати 2 Web-сервери, які реалізують протокол HTTP та виконують роль серверів обробки запитів клієнтів.

2.4. Виконати аналіз протокольного обміну між клієнтом та сервером балансування навантаження під час виконання запитів по протоколам HTTP та HTTPS.

2.5. Рекомендується використовувати наступне програмне забезпечення:

- Web-сервер: nginx або apache;
- Web-клієнт: curl.

2.6. Для перевірки роботи Web-серверу та аналізу протокольного обміну рекомендується використовувати утиліти: telnet, netcat, openssl, curl.

2.7. Додати запис типу A для доменного імені Web-серверу в DNS. Додати сертифікат власного CA у список довірених на вузлі, де запускається Web-клієнт.

Варіанти завдання.

Варіант	**Ім'я Web-серверу**	**Порт HTTP**	**Порт HTTPS**	**Користувачі**	**Метод балансування**
1	www.letter.net	8081	8444	alpha beta gamma	round-robin
2	www.planet.edu	8082	8445	mercury venus earth	least-connected
3	www.cat.com	8083	8446	tiger lion lynx	ip-hash
4	www.flower.org	8084	8447	rose gerbera tulip	round-robin
5	www.linux.net	8085	8448	ubuntu debian centos	least-connected
6	www.color.edu	8086	8449	red greenu blue	ip-hash
7	www.metal.com	8087	8450	gold silver iron	round-robin
8	www.capital.org	8088	8451	london tokyo paris	least-connected

Варіант	Ім'я Web-серверу	Порт HTTP	Порт HTTPS	Користувачі	Метод балансування
9	www.currency.net	8089	8452	dollar dinar lira	ip-hash
10	www.river.edu	8090	8453	nile amazon congo	round-robin
11	www.fruit.com	8091	8454	apple orange grape	least-connected
12	www.digit.org	8092	8455	one two threeg	ip-hash
13	www.month.net	8093	8456	marcht april may	round-robin
14	www.name.edu	8094	8457	maria tomas tereza	least-connected
15	www.country.com	8095	8458	france china spain	ip-hash

3. Контрольні питання.
3.1. Протокол HTTP.
3.2. Покажчики ресурсів URI та URL.
3.3. Структура повідомлень HTTP.
3.4. Методи HTTP.
3.5. Сесії в протоколі HTTP.
3.6. Кешування в протоколі HTTP.
3.7. Постійне з’єднання в протоколі HTTP.

4. Література.

https://web.archive.org/web/20210412192955/https://book.systemsapproach.org/applications/traditional.html
RFC1945 https://tools.ietf.org/html/rfc1945
RFC7230 https://tools.ietf.org/html/rfc7230
RFC7540 https://tools.ietf.org/html/rfc7540
RFC6265 https://tools.ietf.org/html/rfc6265
RFC7234 https://tools.ietf.org/html/rfc7234