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.

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

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.

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:

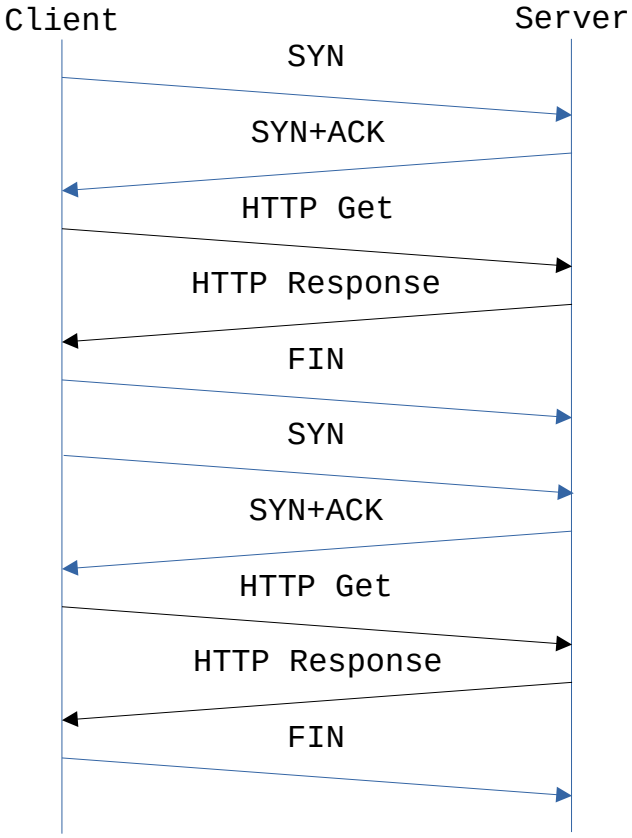
`mailto:admin@comsys.kpi.ua та file:///C:/foo.html.`

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.

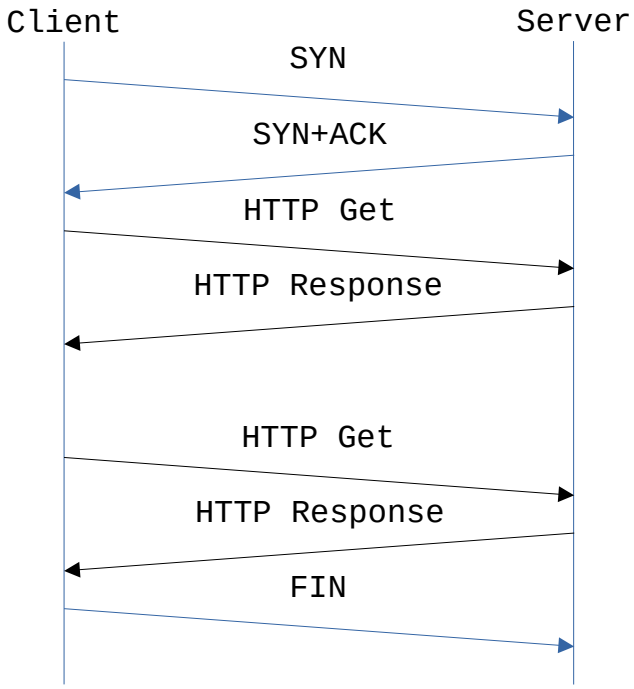
Щоб подолати цю ситуацію, версія HTTP 1.1 запровадила постійні з'єднання - клієнт і сервер можуть обмінюватися кількома повідомленнями запиту/відповіді через одне і те ж TCP-з'єднання. Постійні з'єднання мають багато переваг. По-перше, вони, очевидно, усувають накладні витрати на встановлення з'єднання, 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.

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
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

Варіант	Ім'я Web-серверу	Порт HTTP	Порт HTTPS	Користувачі	Метод балансування
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