Лекція 2. ПРОТОКОЛ ПЕРЕДАЧІ ГІПЕРТЕКСТУ (HTTP)

Організація URL адреси

Створення гіпертекстових посилань в Інтернет є особливістю, яка вперше зробила веб популярним, посилання надають можливість перейти з однієї сторінки в іншу. Користувачам, які публікують сторінки по всьому світу, можливість переглядати сторінку на комп'ютері, яка з'явилася у будь-якому куточку світу, одним кліком було неймовірно. Перехід відбувається за введення адреси URL (*Uniform Resource Locator*). URL дозволяє веб-браузеру вказувати точний файл на веб-сервері або комп'ютері в Інтернет. Концепція дійсно досить проста.

http://www.cengage.com/webpagefolder/anotherfolder/afile.html

Коли ви додаєте URL адресу на свою веб-сторінку HTML, ви вказуєте шлях до певного HTML-файлу. Цей файл може бути на локальному комп'ютері або на комп'ютері в Інтернет.

Часто можна побачити назву файлу в кінці URL. Зверніть увагу на кінець адреси, вказаної у прикладі. Ім’я файла *afile.html* є назвою файла (afile) та розширення *.html*, яке ідентифікує файл як HTML-документ, який може відображати веб-браузер.

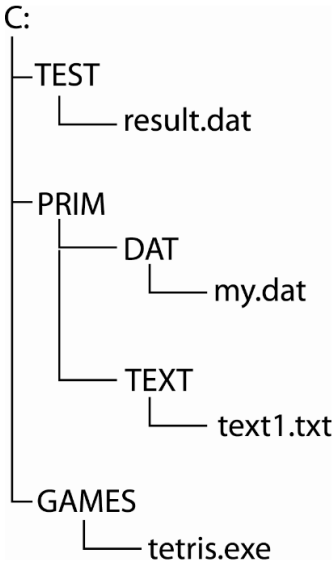
Однак, перш ніж відкривати і переглядати файл *afile.html*, необхідно знати місце розташування файла або шлях до цього файла. Ключ до пошуку місця розташування міститься в URL адресі або веб- адресі. Розіб’ємо URL адресу приклада на окремі частини:

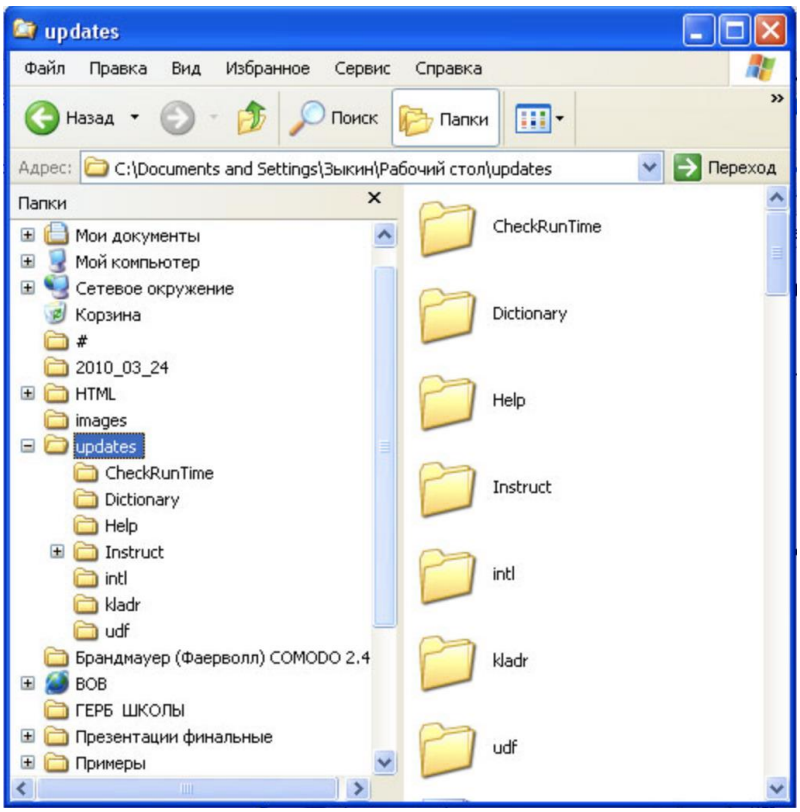
http, за якими йде двокрапка і дві риски. За допомогою http:// можна вказати комп’ютерній мережі, як переносити або переміщувати файли, які ви запитували. HTTP відповідає протоколу *Hypertext Transfer Protocol*. (Ви також можете бачити https://. Літера S – означає безпечний.) Протокол – система зв'язку, яка використовується для передачі даних по комп'ютерних мережах. Це цифрова мова, яку веб-сервери використовують для зв'язку з веб-браузерами.

www.cengage.com – фактиче ім'я веб-сервера (або комп'ютера в мережі), який приймає запит на шукану веб-сторінку (місце розміщення або публікації веб-сторінки). WWW означає *World Wide Web* і спрямовує пошук на конкретний веб-сервер. Цей префікс *www* можна відкидати зовсім, якщо так спроектовано власником сайту. Наприклад, *google.com* і *www.google.com* направляють вас на те саме місце. Проте, зазначення іншого префікса може приймати користувача на різні сервіси, такі як *sites.google.com*, *gmail.google.com* або *docs.google.com*.

.cengage – частина URL-адреси, яка містить назву компанії, яка підтримує веб-сервер. В даному прикладі ця частина свідчить також про те, що це комерційний або бізнес-сайт. Існують також інші адреси з доменами, які позначаються як *.edu* для освіти, *.gov* для урядових веб-сайтів, *.org* для некомерційних або кооперативних організації, або *.biz* для бізнес-сайтів.

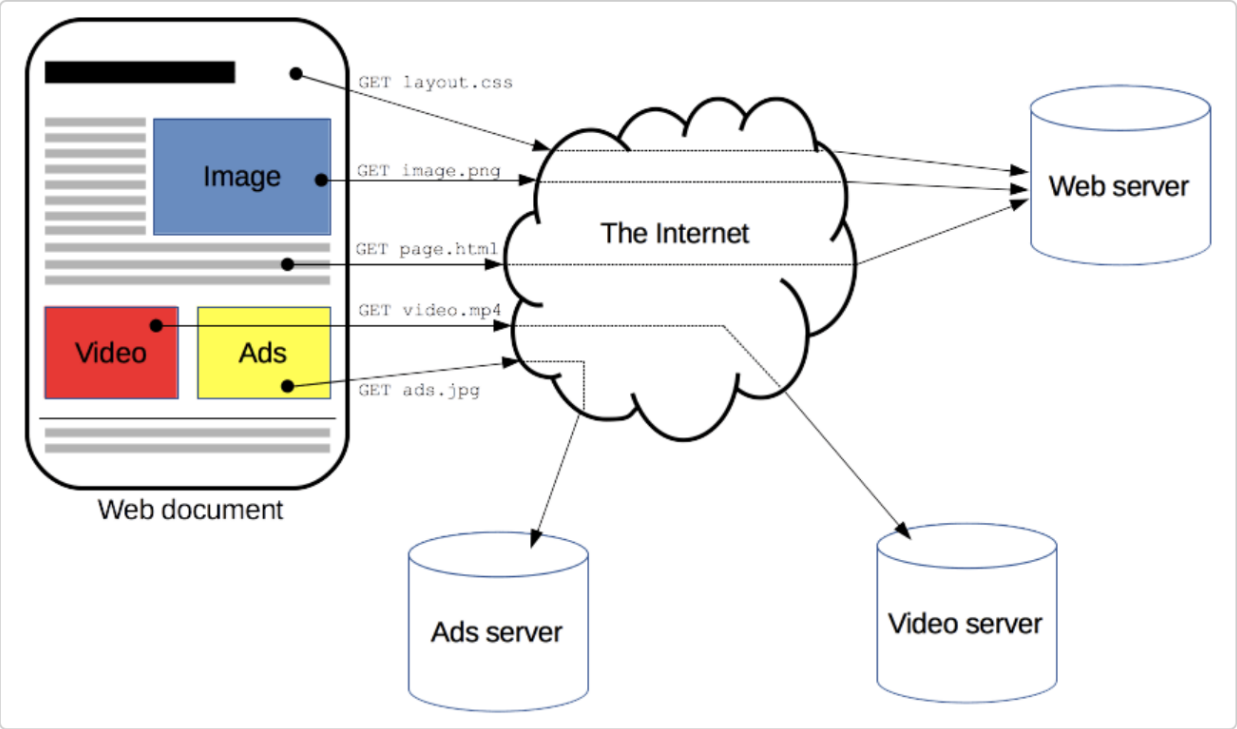
Символи слеш та імена в наступній частині URL адреси (*/webpagefolder/anotherfolder/*) представляють папки на веб-сервері. Їх також називають підкаталогами. На комп'ютері є підкаталоги. На рисунку 2–10 показано, як теки організовані на комп’ютері під керуванням Windows. Всі комп'ютери використовують якусь папку для організації файлів. Якщо ви бажаєте знайти файл на комп’ютері, вам слід знати шлях до всіх підкаталогів, у яких зберігається файл.



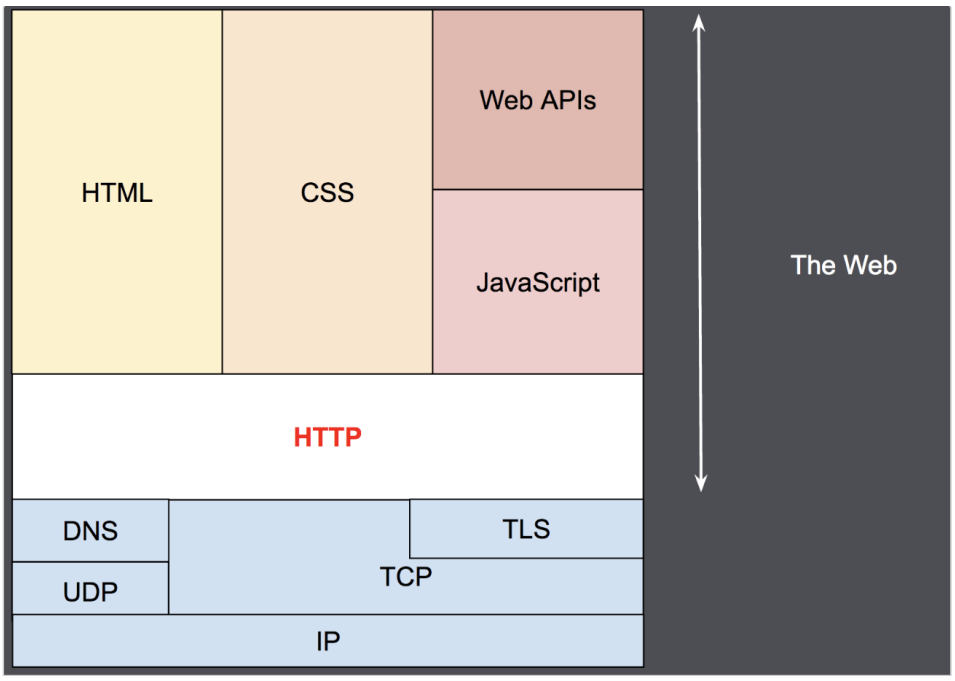


2.2 Огляд HTTP

HTTP – протокол для отримання ресурсів, таких як HTML-документи. Він є основою будь-якого обміну даними в Інтернет і є клієнт-серверним протоколом, що означає запити ініціюються одержувачем, зазвичай веб-браузером. Повний документ відтворюється з різних піддокументів, отриманих, наприклад, з тексту, опису компонування, зображень, відео, сценарії тощо.



Клієнти та сервери спілкуються шляхом обміну окремими повідомленнями (на відміну від потоку даних). Повідомлення, що надсилаються клієнтом, зазвичай веб-браузером, називаються *запитами* і повідомлення, що надсилаються сервером як результати виконання цих запиттів, називаються *відповідями*.



Розроблений на початку 1990-х років, HTTP є розширюваним протоколом, який з часом розвивався. Це протокол прикладного рівня, який надсилається через TCP, або через TCP-з'єднання, зашифроване TLS, хоча теоретично може бути використаний будь-який надійний транспортний протокол. Завдяки своїй розширюваності, він використовується не тільки для отримання гіпертекстових документів, але і зображень, відео або для опису вмісту серверам, як результат HTML форми. HTTP також може використовуватися для отримання частин документів для оновлення веб-сторінок за запитом.

TCP (Transmission Control Protocol) – поширений мережевий протокол, що дозволяє двом хостам реалізовувати з'єднання та обмін потоками даних. TCP гарантує доставку даних і пакетів в тому ж порядку, в якому вони були відправлені. Вінт Серф (Vint Cerf) і Боб Кан (Bob Kahn), які були вченими DARPA в той час, розробили TCP в 1970-х роках.

Роль TCP полягає в тому, щоб забезпечити надійну доставку пакетів, без помилок. TCP має узгоджену контрольну дію, що означає, що початкові запити починаються невеликими, збільшуючи розмір до рівня пропускної здатності комп'ютерів, серверів і мережі, яку вони можуть підтримувати.

Transport Layer Security (TLS), раніше відомий як Secure Sockets Layer (SSL) – протокол, який використовується програмами для безпечного спілкування у мережі, запобігає втручанню та прослуховуванню під час користування електронною поштою, веб-переглядачами, обміну повідомленнями та за використання інших протоколів. Як SSL, так і TLS є протоколами клієнт / сервер, які забезпечують конфіденційність зв'язку за допомогою криптографічних протоколів для забезпечення безпеки обміну через мережу. Коли сервер і клієнт спілкуються за допомогою TLS, це гарантує, що жодна третя сторона не може прослуховувати або переглядати повідомлення відіслані будь-яким чином.

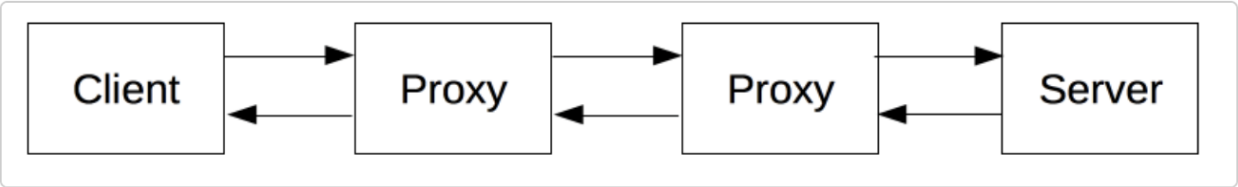
Secure Sockets Layer, або SSL, був старою стандартною технологією безпеки для створення зашифрованого мережевого з'єднання між сервером і клієнтом, забезпечуючи приватність та безпеку всіх переданих даних. Поточна

версія SSL — версія 3.0, випущена Netscape в 1996 році і замінена на Transport Layer Security (TLS).

2.3 Компоненти систем основаних на HTTP

HTTP – клієнт-серверний протокол: запити надсилаються одним суб'єктом, користувачем-агентом (або проксі від його імені). Більшість часу користувач-агент є веб-браузером, але це може бути й інша програма, наприклад, робот, який призначений для того, щоб Web заповнювати і підтримувати індекс пошукової системи.

Кожен індивідуальний запит відправляється на сервер, який обробляє його і надає результат, яка називається відповіддю. Між клієнтом і сервером існують численні сутності, об’єднані під назвою «проксі» (proxy), які виконують різні операції і виступають як шлюзи або кеші (cache), наприклад.



Проксі-сервер – проміжна програма (програма-посередник) або комп'ютер, що використовується під час навігації по різних мережах Інтернет. Вони полегшують доступ до контенту в World Wide Web. Проксі перехоплює запити і обслуговує зворотні відповіді; він може пересилати запити, або ні (наприклад, у випадку кешу), і він може модифікувати його (наприклад, змінюючи його заголовки, на межі між двома мережами).

Проксі може бути на локальному комп'ютері користувача, або будь-де між комп'ютером користувача і сервером призначення в Інтернет. Загалом існує два основних типи проксі-серверів:

Прямий проксі-сервер, який обробляє запити від і до будь-якого місця в Інтернет.

Зворотний проксі, що приймає запити з Інтернет та перенаправляє їх на сервери у внутрішній мережі.

Кеш (веб-кеш або HTTP-кеш) – компонент, який тимчасово зберігає HTTP-відповіді, щоб вони могли використовуватися для подальших HTTP-запитів, поки відповідають певним умовам.

В реальності між браузером користувача та сервером, що обробляє запит, більше комп'ютерів: маршрутизатори, модеми і багато іншого. Завдяки багатошаровій структурі веб-мережі, вони приховані в мережі та транспортних шарах. HTTP знаходиться зверху, на рівні застосунків. Основні шари здебільшого не мають значення для опису HTTP, хоча є важливими для діагностики проблем мережі.

Агент користувача

Агент користувача (User-Agent) – це будь-який інструмент, який діє від імені користувача. Цю роль в першу чергу виконує веб-браузер, але також може виконувати програми, які використовуються інженерами та веб-розробниками для налагодження своїх застосунків.

Браузер завжди є об'єктом, який ініціює запит. Він ніколи не є сервером (хоча деякі механізми були додані для імітації ініційованих сервером повідомлень).

Для відображення веб-сторінки браузер посилає оригінальний запит на отримання HTML-документа, який представляє сторінку. Потім він обробляє цей файл, роблячи додаткові запити, відповідними скриптами виконання, інформацією про компонування (CSS) відображення контенту, та додаткові ресурси, які містяться в межах сторінки (зазвичай зображення і відео). Потім веб-браузер об'єднує ці ресурси, щоб представити повний документ або веб-сторінку. Скрипти, що виконуються браузером, можуть отримувати більше ресурсів в пізніших фазах візуалізації сторінки і браузер відповідно оновлює веб-сторінку.

Веб-сторінка є гіпертекстовим документом. Це означає, що деякі частини відображуваного контенту є посиланнями, які можуть бути активовані (зазвичай, клацанням мишею) для отримання нової веб-сторінки, що дозволяє користувачеві направляти свого агента і здійснювати навігацію через Web. Браузер переводить ці посилання в HTTP-запити, і далі інтерпретує

HTTP-відповіді, щоб представити користувачеві відповідну візуалізацію контенту.

Веб-сервер

На протилежному боці каналу зв'язку знаходиться сервер, що обслуговує документ, який запитав клієнт. З боку користувача сервер виглядає практично як єдина машина, але насправді це може бути набір серверів, що мають спільне збалансоване навантаження, або складне програмне забезпечення, що збирає відповідні дані по інших комп'ютерах (наприклад, кеш, сервер БД або сервери електронної комерції), повністю або частково генеруючи документ за запитом агента користувача

Сервер не обов'язково є однією машиною, але кілька екземплярів серверного програмного забезпечення можуть бути розміщені на одній машині. За допомогою HTTP/1.1 і заголовка Host вони можуть навіть бути за однією і тією ж IP-адресою.

Посередники

Між веб-браузером і сервером численні комп'ютери та програми ретранслюють HTTP-повідомлення. Згідно з багаторівневою структурою веб-стека більшість з них працюють на транспортному, мережевому або фізичному рівнях, їх дія стає очевидною на HTTP-рівні і має значний вплив на продуктивність. Ті, що працюють на прикладному рівні, як правило, називаються посередниками або проксі (proxies).

Proxy Server (від англ. Proxy Proxy – представник, уповноважений; часто просто проксі, проксі-сервер-посередник) – проміжний сервер (набір програм) в комп'ютерних мережах, що виконує роль посередника між користувачем і цільовим сервером (однак, обидві сторони можуть знати або не знати про посередництво), дозволяючи клієнтам виконувати непрямі запити на інші мережеві служби (шляхом прийому та пересилання їх через проксі-сервер) та отримувати відповіді. Клієнт спочатку з'єднується з проксі-сервером і запитує якийсь ресурс (наприклад, електронну пошту), що знаходиться на іншому сервері. Потім проксі-сервер або з'єднується з вказаним сервером і отримує з нього ресурс, або повертає ресурс з власного кешу (якщо проксі має власний кеш). У деяких випадках запит клієнта або відповідь сервера можуть бути

модифіковані проксі-сервером для конкретних цілей. Проксі-сервер дозволяє клієнтському комп'ютеру захищатися від деяких мережевих атак і допомагає зберігати анонімність клієнта, але також може використовуватися зловмисниками для приховування адреси сайту, визначеного як зловмисний, зміни вмісту цільового сайту (spoofing), а також перехоплення запитів користувача.

Проксі-сервери найчастіше використовуються для наступних цілей:

– доступ комп'ютерів локальної мережі до мережі Інтернет;

– кешування даних: якщо до одних і тих же зовнішніх ресурсів часто надходять запити, то можна знизити пропускне навантаження на зовнішню мережу і прискорити отримання запитаної інформації клієнтом;

– стиснення даних: проксі-сервер вивантажує інформацію з Інтернету і передає інформацію кінцевому користувачеві в стисненому вигляді для збереження зовнішнього мережевого трафіку клієнта або внутрішнього трафіку організації, в якій встановлено проксі-сервер;

– захист локальної мережі від зовнішнього доступу: наприклад, можна налаштувати проксі-сервер так, щоб місцеві комп'ютери мали доступ до зовнішніх ресурсів тільки через нього, а зовнішні комп'ютери взагалі не зможуть отримати доступ до локальних ресурсів (вони «бачать» тільки проксі-сервер);

– обмеження доступу з локальної мережі до зовнішньої: наприклад, можна відмовити в доступі до певних веб-сайтів, обмежити використання Інтернету деяким місцевим користувачам, встановити квоти на трафік або пропускну здатність, фільтрувати рекламу і віруси;

– анонімізація доступу до різних ресурсів: проксі-сервер може приховувати інформацію про джерело запиту або користувача; у цьому випадку цільовий сервер бачить тільки інформацію про проксі-сервер, наприклад IP-адресу, але не може визначити справжнє джерело запиту; існують також викривлення проксі-серверів, які передають хибну інформацію про справжнього користувача на цільовий сервер;

– ухилення від обмеження доступу: використовується, наприклад, користувачами країн, де доступ до деяких ресурсів обмежений законом і фільтрується.

Проксі-сервер, до якого може отримати доступ будь-який користувач Інтернету, називається *відкритим* сервером.

Існують наступні види проксі-серверів.

Прозорий проксі – схема зв'язку, за якої трафік або його частина маршрутизується неявно на проксі-сервер (засобами маршрутизатора). У цьому випадку клієнт може використовувати всі переваги проксі-сервера без додаткових налаштувань браузера (або іншої програми для роботи з Інтернетом).

Зворотний проксі – це проксі-сервер, який, на відміну від прямого проксі, пересилає запити клієнта від зовнішньої мережі до одного або декількох серверів, які логічно розташовані у внутрішній мережі. Він часто використовується для збалансування мережевого навантаження між декількома веб-серверами і підвищення їх безпеки, при цьому грає роль міжмережевого екрана (брандмауера або файєрвола) на прикладному рівні.

Веб-проксі – це широкий клас проксі-серверів, які виготовляються у вигляді веб-застосунку.

2.4 Основні аспекти HTTP

HTTP дозволяє з’єднання між системами з різною архітектурою та конфігурацією мережі. Це можливо завдяки тому, що цей протокол висуває найбільш загальні вимоги до систем та не зберігає стан між обмінами різними повідомленнями.

Через це HTTP вважається протоколом без запам’ятовування стану. Для транспортування повідомлень зазвичай використовується протокол TCP (Transmission Control Protocol; протокол управління передачею, протокол TCP – транспортного рівня з набору TCP/IP. Такий протокол гарантує доставку пакетів даних в потрібній послідовності, проте трафік може бути неравномірним, через можливі затримки пакетів), однак може використовуватися будь-який інший механізм, який можна використовувати для транспортування повідомлень, наприклад, QUIC (Quick UDP Internet Connections) – експериментальний інтернет-протокол, розроблений Google в кінці 2012 року. За умовчанням портом для HTTP є порт 80, але можуть використовуватися й інші порти.

Обмін повідомленнями між клієнтом та сервером відбувається за схемою «запит-відповідь». Клієнт починає спілкування шляхом відправлення запита HTTP, у відповідь на який сервер відсилає повідомлення відповіді HTTP.

Поточна версія протокола – HTTP/1.1, в якій порівняно з попередньою (HTTP/1.0) додані додаткові можливості:

– довгострокові з'єднання (persistent connections; передача в одному TCP-з'єднанні кількох об'єктів);

– кодування передачі даних типа "chunked" (chunked transfer-coding; механізм передачі даних в протоколі передачі гіпертексту HTTP, який дозволяє надійно доставляти дані від сервера клієнту без необхідності попереднього зазначення точного розміру всього тіла HTTP-повідомлення; це досягається розбиттям повідомлення на невеличкі частини (chunks), з подальшою передачею кожної частини з зазначенням лише її розміру, завершення передачі повідомлення визначається останньою частиною нулевої довжини.

2.5 Що контролює HTTP

Постійна модифікація та розширення повноважень HTTP дозволила збільшити частку елементів, що контролюються та управляються за протоколом HTTP у мережі.

До списку можливостей, управління якими відбувається за допомогою HTTP належать:

1 Кешування документів: сервер може інструктувати проксі і клієнтів про те, що саме кешувати і на скільки довго.

2 Уникнення прослуховування та інших порушень конфіденціальності: веб-браузери забезпечують чітке розподілення веб-сайтів. Лише сторінки одного походження можуть отримати доступ до всієї інформації веб-сторінки. Такий розподіл є додатковим навантаженням для сервера, тому заголовки HTTP послаблюють таке розподілення на серверному боці, та надають документу змогу стати патчем інформації отриманої з різних доменів.

3 Деякі сторінки могут бути захищені, таким чином лише певні користувачі могут отримати доступ до них. Базова аутентифікація може бути забезпечена за допомогою HTTP, або за допомогою WWW-Authenticate та

аналогічних заголовків, або шляхом встановлення визначеного сеанса за допомогою HTTP-файлів.

4 Проксі-сервери або клієнти часто розташовуються в інтрамережах та приховують свою істинну IP-адресу від інших комп'ютерів. Потім запити HTTP проходять через проксі, для нівелювання такого мережевого бар'єру. Проте не всі проксі є саме HTTP проксі.

2.6 робота з HTTP з'єднаннями

Якщо клієнт бажає зв'язатися з сервером (кінцевим або проксі-сервером) він має виконати наступні дії:

1. Відкрити TCP з'єднання: TCP з'єднання використовується для відправлення запиту або кількох запитів для отримання відповіді. Клієнт може відкривати нове з'єднання, вдруге використовувати вже відкрите з'єднання або відкрити кілька TCP з'єднань з серверами.

2. Відправити HTTP-повідомлення: HTTP-повідомлення (до HTTP/2) є читабельним для людини. В HTTP/2 ці прості повідомлення інкапсулюються у фрейми, що робить їх непридатними для прямого читання, проте принцип залишається тим самим. Наприклад:

```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

3. Прочитати відповідь, відправлену сервером, наприклад:

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!DOCTYPE html>… (here come the 29769 bytes of the
requested web page)
```

4. Закрити з'єднання або використати його для подальших запитів.

Якщо HTTP конвеєр активовано, декілька запитів можуть бути відправлені без очікування повного отримання відповіді на перший запит. Через труднощі поєднання роботи старого ПЗ з сучасними версіями конвеєр HTTP було замінено на HTTP/2.

Протокол HTTP/2 базується на SPDY – протоколі прикладного рівня для передачі веб-контента. Протокол розроблено корпорацією Google. За замислом розробників, даний протокол має замінити деякі частини протокола HTTP такі, як управління з'єднаннями та форматами передачі даних.

Основною метою SPDY є зменшення часу на отримання відповіді у мережевому з'єднанні за рахунок створення в початковому канале зв'язку кількох підканалів з меншою пропускною спроможністю.

2.7 HTTP повідомлення

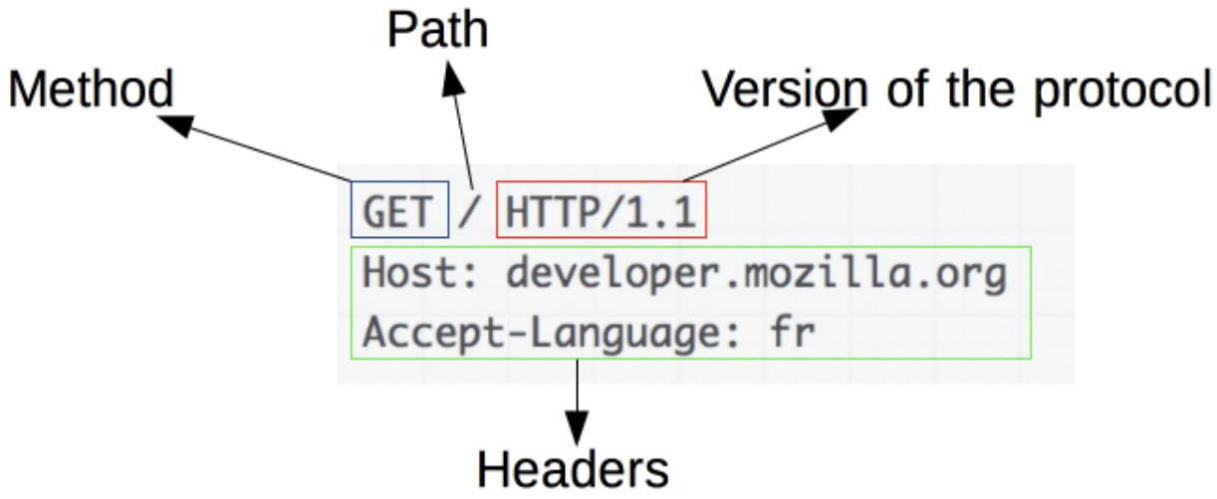
Повідомлення HTTP, визначені в HTTP/1.1 та більш ранніх версіях протоколу є прийнятними для читання. В HTTP/2 ці повідомлення вбудовані в двійкову структуру, рамку або фрейм, яка дозволяє оптимізувати передачу, наприклад, стиснення заголовків та мультиплексування (використання замість

одного каналу підканалів). Навіть якщо лише частина оригінального повідомлення HTTP відправляється за допомогою цієї версії HTTP, семантика кожного повідомлення залишається незмінною, та клієнт відтворює (фактично) початковий запит HTTP/1.1.

Існує два типи HTTP повідомлень: запити та відповіді, кожен з яких має власний формат.

Запити

Приклад запита HTTP:



Запит HTTP складається з наступних елементів:

– Метод HTTP, що є зазвичай дієсловом на кшталт GET, POST або іменник на кшталт OPTIONS або HEAD, що визначає операцію, яку клієнт бажає виконати. Як правило, клієнт прагне отримати ресурс (за допомогою GET) або опублікувати значення форми HTML (за допомогою POST), проте в інших випадках може бути потрібним використання більшої кількості операцій.

– Шлях отримання ресурса; URL ресурса, позбавлений елементів, які є зрозумілими з контекста, наприклад, без протокола (http:/), домена (здесь, developer.mozilla.org) або TCP порта (здесь, 80).

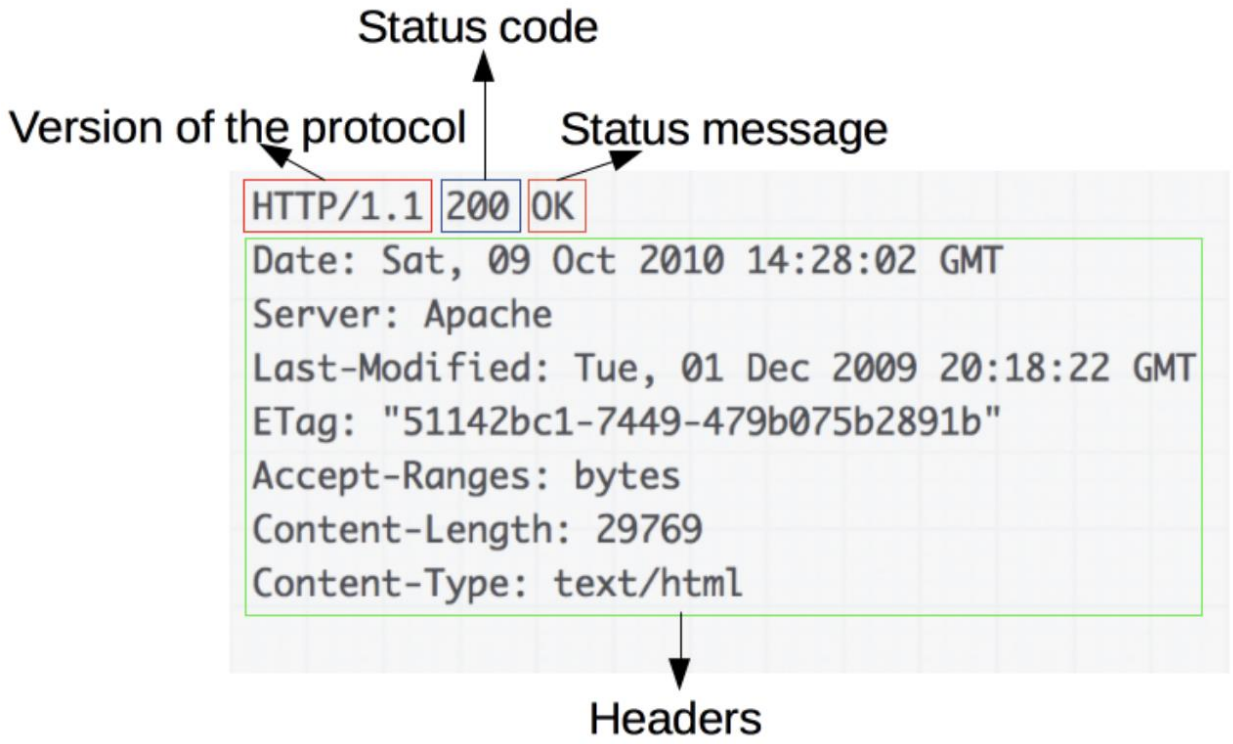
– Версія протокола HTTP.

– Додаткові заголовки, які надають додаткові відомості для серверів.

– Тіло, для деяких методів, таких як POST, так само як і для тих, що розташовані у відповідях, які містять ресурс відправлення.

Відповіді

Приклад відповіді:



Відповідь складається з наступних елементів:

– Версія протокола HTTP яка використовується для обміну.
– Код статуса, який вказує чи був виконаний запит чині, і якщо ні то чому.
– Повідомлення про стан (стислий опис кода стану).
– Заголовки HTTP, наприклад, для запитів.
– Тіло, що містить сам ресурс (цей елемент є опціональним).

2.8 API, що базуються на HTTP протоколі

API, який використовується найчастіше та базується на HTTP, є XMLHttpRequest API. Цей API може застосовуватися для обміну даними між агентом користувача та сервером. Сучасний Fetch API має ті ж самі функції з більш потужним та гнучким набором функцій.

Інший API – це однобічний сервіс, який дозволяє серверу відправляти події, з використанням HTTP в якості транспортного механізма. За використання інтерфейсу EventSource, клієнт відкриває з'єднання та встановлює механізми обробки подій. Браузер клієнта автоматично перетворює повідомлення, що потрапляють до потоку HTTP, у відповідні Event-об'єкти. Після цього ці об'єкти потрапляють до механізмів обробки подій, які були зареєстровані для даного типа подій або до обробника подій onmessage, якщо не було встановлено тип механізму обробки подій.

API (Application Programming Interface) – опис способів взаємодії однієї комп'ютерної програми з іншими. Зазвичай входить до опису певного інтернет-протокола, програмного каркаса (фреймворку) або стандарту викликів функцій ОС. Часто реалізується окремою програмною бібліотекою або сервісом ОС. Використовується програмістами під час написання різних застосунків. Тобто API – це набір компонентів, за допомогою яких комп'ютерна програма (бот або сайт) може використовувати іншу програму.