**Міністерство освіти і науки України**

Київський національний університет

імені Т.Шевченка

***Реферат***

*на тему*:

«*Порівняння протоколів*

*FTP і HTTP****»***

***Студента другого курсу***

***Групи К-23***

***Арзамасцева Владислава***

***Факультету комп’ютерних наук***

***та кібернетики***

***Київ***

*2020*

***План***

**Вступ…………………………………………………………….3-4**

**Розділ 1: Протокол FTP………………………………………5-12**

**Розділ 2: Протокол HTTP……………………………………13-21**

**Розділ 3: Спільне і відмінне…………………………………22-39**

**Висновки……………………………………………………….30**

**Список використаних джерел………………………………31**

***Спочатку було слово… Потім протокол -***

***для його передачі (HTTP/0.9)***

***Вступ***

Темою мого реферату є «Порівняння протоколів FTP і HTTP».

На фоні швидко змінюваної мережі Інтернет протокол FTP має старий, навіть дещо архаїчний вигляд. Ранні чорнові специфікації протоколу датуються 1971-м роком, створення поточної специфікації розпочато в 1985-му році. Протягом останніх двох десятиріч протокол не змінювався в своїй основі. В той час мережею Інтернет користувались в основному університети та дослідницькі центри. Користувачів було не багато, в основному, вони знали одне одного та працювали заодно. Інтернет був «дружньою родиною», а проблем безпеки не виникало. Зараз багато чого змінилось. Технологічний процес розвивався швидше, ніж можна було уявити, одночасно з’явилось нове покоління користувачів. Інтернет тепер звичайне явище, завдяки якому мільйони людей спілкуються одне з одним різними способами. Однак, в той же час, Інтернет став ворожим. Доступність та відкритість мережі зацікавила і користувачів зі злочинними намірами, які активно почали використовувати помилки та недосвідченість інших. Побічним ефектом такого розвитку подій стали NAT-роутери та персональні файрволи для захисту користувачів від недоопрацювань операційних систем та додатків. В більшості випадків ці явища конфліктують з роботою протоколу. Ситуацію погіршують недоопрацювання в самих роутерах та файрволах. В той же час правильне налаштування FTP пропонує досить надійний та перевірений спосіб передачі файлів.

Протокол передачі гіпертекста (HTTP) – протокол прикладного рівня для розподілених, сумісних, багатосередовищних інформаційних систем. HTTP використовуюється в World Wide Web (WWW) починаючи з 1990 року. Першою версією HTTP, відомою як HTTP/0.9, був простий протокол для передачі необроблених даних через Інтернет. Очевидно, що цей протокол потребував доопрацювання, напевно, і сам розробник Тім Бе́рнерс-Лі  це розумів. Можливо, саме з цієї причини протокол носив не номер 1, який зазвичай присвоювався релізам, а 0,9 (beta), ніби наголошуючи, що це пробна версія і потребує покращення. RFC 1945 HTTP/1.0 був покращенням цього протоколу, допускав MIME-подібний формат повідомлень, що містить метаінформацію про дані, що передаються та мав модифіковану семантику запитів/відповідей. Однак, HTTP/1.0 недостатньо враховував особливості роботи з ієрархічними проксі-серверами (hierarchical proxies), кешуванням, постійними з’єднаннями та віртуальними хостами. Це потребувало введення нової версії протоколу.

Сьогодні вже нікого не здивуєш переглядом фільмів через Інтернет, шопінгом в online-магазині, люди працюють та спілкуються з друзями за допомогою мережі Інтернет.

Метою написання реферату є дослідження історії виникнення та порівняння схем роботи протоколів FTP і HTTP, визначення їх особливостей, аналіз спільних та відмінних рис.

При написанні роботи я користувався наступними методами наукового дослідження: аналізу, порівняння, узагальнення. Матеріали дослідження можуть бути використані при написанні рефератів та курсових робіт на зазначену тематику.

Робота складається зі вступу, трьох розділів, висновків та списку використаних джерел.

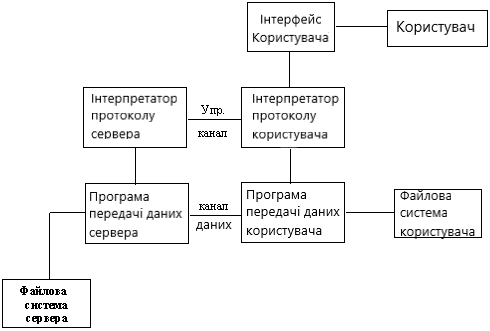
***Розділ 1: Протокол FTP***

**1.1 Визначення**

FTP (File Transfer Protocol) – протокол передачі файлів мережею. Він дозволяє підключатись до серверів цього протоколу і продивлятись вміст каталогів, завантажувати файли із сервера і на сервер. Це один із найстаріших прикладних протоколів, з’явився ще до HTTP і навіть до TCP/IP, в 1971 році. І сьогодні він широко використовується для розповсюдження ПЗ і доступу до віддалених хостів. Користувачі FTP можуть пройти аутентифікацію, передаючи логін і пароль відкритим текстом, або ж зайти «анонімно», якщо це дозволено на сервері – насправді передати „anonymous” в якості імені та адресу електронної пошти в якості пароля.

**1.2 Схема роботи**

Протокол побудований на архітектурі «клієнт-сервер» і використовує різні мережеві з’єднання для передачі команді даних між клієнтом і сервером. На схемі зображено модель протоколу



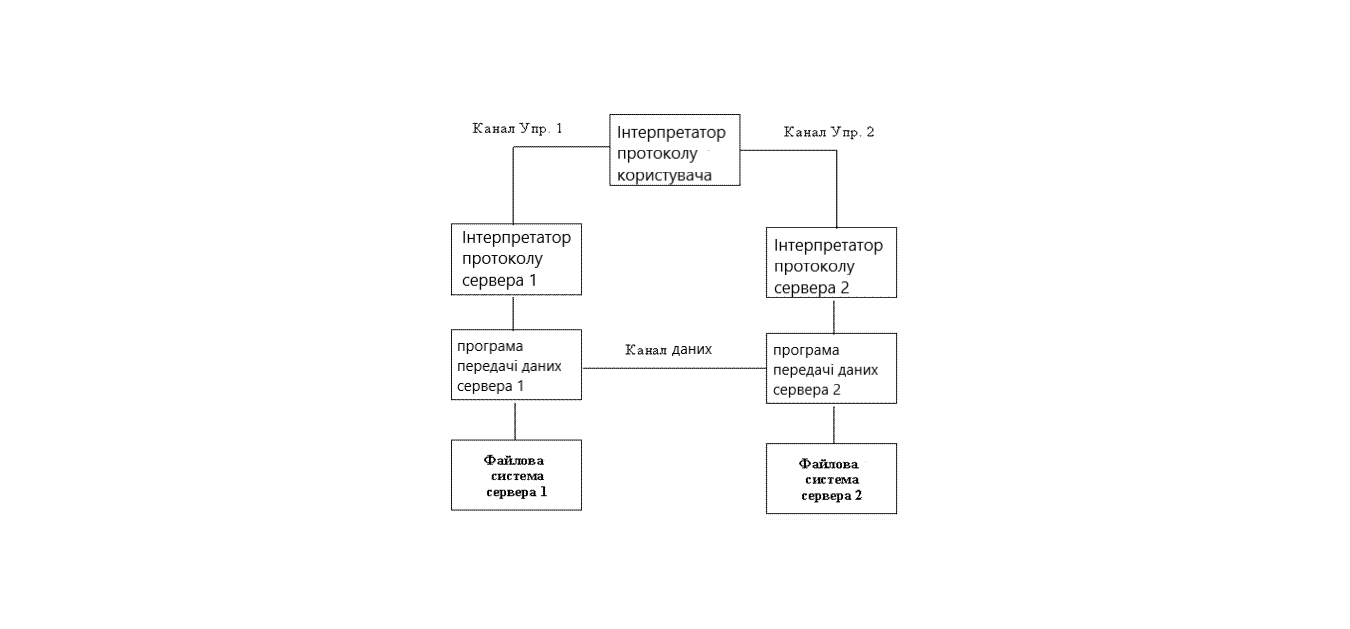
В FTP з’єднання ініціюється інтерпретатором протоколу користувача. Управління обміном реалізується за стандартом протоколу TELNET. Команди FTP генеруються інтерпретатором протоколу користувача і передаються на сервер. Відповіді сервера відправляються клієнту також по каналу управління. В загальному випадку користувач має можливість встановити контакт з інтерпретатором протоколу сервера і відмінними від інтерпретатора користувача засобами.

Команди FTP визначають параметри каналу передачі даних і самого процесу передачі. Вони також визначають і характер роботи з віддаленою і локальною файловими системами. Сесія управління ініціалізує канал передачі даних. При організації каналу передачі даних послідовність дій інша, відмінна від організації каналу управління. В цьому разі сервер ініціює обмін даними згідно з параметрами, узгодженими в сесії управління.

Канал даних встановлюється для того ж хоста, що і канал управління, через який ведеться налаштування каналу даних. Канал даних може бути використаний як для прийому, так і для передачі даних.

Можлива ситуація, коли дані можуть передаватись на третю машину. В такому разі клієнт організовує канал управління з двома серверами і організовує прямий канал даних між ними. Команди управління ідуть через користувача, а дані – напряму між серверами.

Канал управління повинен бути відкритим при передачі даних між машинами. У випадку його закриття передача даних припиняється.



**1.3 З’єднання і передача даних**

Сервер відповідає по потоку управління 3-значними ASCII-кодами стану з необов’язковим текстовим повідомленням. Наприклад, «200» (чи «200 ОК») означає, що остання команда була успішно виконана. Цифри представляють код відповіді, текст – пояснення. Поточна передача по каналу може бути перервана за допомогою спеціального повідомлення, відправленого по потоку управління.

FTP може працювати у активному чи пасивному режимі, від вибору якого залежить спосіб встановлення з’єднання. В активному режимі клієнт сам створює керуюче TCP-з’єднання із сервером і відправляє свою IP-адресу і довільний номер клієнтського порту, після чого чекає, доки сервер запустить TCP-з’єднання з цією адресою і номером порту. Якщо клієнт знаходиться за фаєрволом і не може прийняти вхідне TCP-з’єднання, можна використати пасивний режим. В цьому режимі клієнт використовує потік управління, щоб послати серверу команду PASV, і далі отримує від сервера його IP-адресу і номер порту, котрі згодом використовуються клієнтом для відкриття потоку даних з довільного клієнтського порту в отриману адресу і порт.

При передачі даних мережею, можуть бути використані 4 представлення даних:

* ASCII – використовується для тексту. Дані, якщо необхідно, до передачі конвертуються із символьного представлення на хоста-відправника, у 8-бітний ASCII, і потім, знову ж, якщо треба, у символьне представлення хоста-одержувача.
* Бінарний режим (режим зображення) – хост-відправник посилає кожен файл байт за байтом, а одержувач зберігає потік байтів. Підтримка цього режиму рекомендована для всіх реалізацій FTP.
* EBCDIC – використовується для передачі звичайного тексту між хостами в кодуванні EBCDIC. В іншому він аналогічний до ASCII-режиму.
* Локальний режим – дозволяє 2-м комп’ютерам з ідентичними установками посилати дані у власному форматі без конвертації у ASCII.

Для текстових файлів надано різноманітні формати управління і налаштування структури запису. Ці особливості було розроблено для роботи з файлами, що містять Telnet чи ASA форматування.

Передача може здійснюватись у будь-якому з 3-х режимів:

* Потоковий режим – дані посилаються як неперервний потік, звільняючи FTP від виконання будь-якої обробки. Замість цього вся обробка виконується TCP. Ідентифікатор кінця файлу не потрібен, за винятком розділення даних на записи.
* Блочний режим – FTP розбиває дані на декілька блоків (блок заголовка, кількість байт, поле даних) і далі передає їх TCP.
* Режим стискання – дані стискаються єдиним алгоритмом.

**1.4 Безпека**

FTP не був розроблений як захищений (особливо за сучасними мірками) протокол і уразливий до наступних атак:

* Приховані атаки (bounce attack)
* Спуф-атаки (spoof attacks)
* Атаки методом грубої сили (brute force attacks)
* Перехоплення пакетів, сніффінг (packet capture, sniffing)
* Захоплення портів(port stealing)

FTP не може зашифрувати свій трафік, всі дані передаються у вигляді тексту, тож будь-хто може прочитати імена користувачів, паролі, команди та дані. Ця проблема характерна для багатьох Інтернет-протоколів (SMTP, TELNET, POP, IMAP), розроблених до створення таких механізмів шифрування, як TLS, SSL. Звичне розв’язання цієї проблеми – використання безпечних, TLS-захищених версій уразливих протоколів (FTPS для FTP, TELNETS для TELNET і т.д.), або інший, більш захищений протокол, як SFTP/SCP, які надаються з більшістю реалізацій протоколу Secure Shell.

***Безпечний FTP***

Існує декілька способів безпечної передачі файлів, котрі іноді називають «Безпечним FTP».

**FTPS**

Явний FTPS – розширення стандарту FTP, яке дозволяє клієнтам вимагати зашифровану FTP-сесію. Це реалізується надсиланням команди «AUTH TLS». Сервер може дозволити або відхилити спроби встановити з’єднання, котрі не вимагають TLS. Неявний FTPS – застарілий стандарт для FTP, що вимагає використання SSL- чи TLS-з’єднань і використовувати інші порти.

### SFTP

SFTP (SSH File Transfer Protocol) не пов’язаний з FTP, крім факту, що

він теж передає файли і має аналогічний набір команд для клієнтів. SFTP – це програма, що використовує SSH для передачі файлів. На відміну від стандартного FTP, він шифрує іще й команди, і дані, захищаючи паролі і конфіденційні дані від відкритої передачі мережею. SFTP за функціональністю схожий на FTP, але оскільки він використовує інший протокол, то клієнти стандартного FTP не можуть зв’язатися із SFTP-сервером і навпаки.

### FTP через SSH (не SFTP)

FTP через SSH відноситься до практики тунелювання звичайної FTP-сесії через SSH-з’єднання. Оскільки FTP використовує декілька TCP-з’єднань, тунелювання через SSH особливо складне. Коли багато SSH-клієнтів намагаються встановити тунель для каналу управління (початкове «клієнт-серверне» з’єднання по порту 21), захищеним буде тільки цей канал; при передачі даних ПЗ FTP на будь-якому кінці встановить нові TCP-з’єднання (канали даних), які обійдуть SSH-з’єднання і втратять цілісність захисту.

**1.5 Основні команди**

* ABOR — Перервати передачу файлу
* CDUP — Змінити каталог на вищестоячий
* CWD — Змінити каталог
* DELE — Видалити файл (DELE filename).
* EPSV — Увійти в розширений пасивний режим. Застосовується замість PASV.
* HELP — Виводить список команд, що приймає сервер.
* LIST — Повертає список файлів каталогу. Список передається через з’єднання даних.
* MDTM — Повертає час модифікації файла.
* MKD — Створити каталог.
* NLST — Повертає список файлів каталога в більш короткому форматі, ніж LIST. Список передається через з’єднання данных.
* NOOP — Пуста операція.
* PASS — Пароль.
* PASV — Увійти в пасивний режим. Сервер поверне адресу і порт, до яких треба підключитися, щоб забрати данні. Передача почнеться при введенні наступних команд: RETR, LIST і т. д.
* PORT — Увійти в активний режим. Наприклад PORT 12,34,45,56,78,89. На відміну від пассивного режиму, для передачі даних сервер сам підключається до клієнта.
* PWD — Повертає поточний каталог.
* QUIT — Відключитись.
* REIN — Реініціалізувати підключення.
* RETR — Скачати файл. Перед RETR повинна бути команда PASV чи PORT.
* RMD — Видалити каталог.
* RNFR и RNTO — Перейменувати файл. RNFR — що перейменовувати, RNTO — у що.
* SIZE — Повертає розмір файла.
* STOR — Закачати файл. Перед STOR повинна бути команда PASV чи PORT.
* SYST — Повертає тип системи (UNIX, WIN, …).
* TYPE — Встановити тип передачі файла (бінарний, текстовий).
* USER — Им’я користувача для входу на сервер.

**1.6 Коди відповідей FTP**

Нижче представлено опис відповіді, яку можна отримати від FTP-сервера. Як було сказано раніше, код відповіді – 3-значне число. 1-ша цифра відповідає за один із трьох результатів: успіх, відмова чи вказівку на помилку або неповну відповідь:

* 2хх – Успіх.
* 4хх/5хх – Команда не може бути виконана.
* 1хх/3хх – Помилка або неповна відповідь.

2-га цифра визначає тип помилки:

* x0z – Синтаксична.
* x1z – Інформація. Відповідає інформаційному повідомленню.
* x2z – З’єднання. Повідомлення відноситься до керуючого з’єднання або до з’єднання даних.
* x3z – Відповідає повідомленням про аутентифікацію користувача і його правам.
* x4z – Не визначено.
* x5z – Файлова система. Відповідає повідомленню про стан файлової системи.

3-тя цифра остаточно визначає помилку.

**1.7 Приклад**

У прикладі жирним шрифтом виділено команди, які подає клієнт, звичайним шрифтом – повідомлення сервера. Примітки відділено двома флешами:

220 FTP server ready.

220 Hello World!

**USER anonymous**

331 Anonymous login ok, send your complete email address as your password

**PASS \*\*\*\*\*\*\*\*\*\*\*\***

230 Logged in anonymously.

**PASV**

227 Entering Passive Mode (192,168,254,253,233,92) //Клієнт повинен відкрити з’єднання на переданий IP

**LIST**

150 Here comes the directory listing. // Сервер передає список файлів у каталог

226 Directory send OK.

**CWD incoming**

250 Directory successfully changed.

**PASV**

227 Entering Passive Mode (192,168,254,253,207,56)

**STOR example.avi**

150 Ok to send data. // Клиієнт передає вміст файлу

226 File receive OK.

**QUIT**

221 Goodbye.

Аргумент 192,168, 254,253, 56 означає, що з’єднання із сервером очікується на вузлі з IP-адресою 192.268.254.253 на порту 207 << 8 + 56 = 53048, де << - побітовий зсув, 207 записується в молодший розряд, а потім зсувається в старший, щоб у молодший записати 56.

На багатьох FTP-серверах існує каталог (із назвою incoming, upload, etc.), відкритий на запис і призначений для завантаження файлів на сервер. Це дозволяє клієнтам наповнювати сервер свіжими даними.

***Розділ 2: Протокол HTTP***

**2.1 Визначення і структура**

HTTP (HyperText Transfer Protocol – «протокол передачі гіпертексту») – протокол прикладного рівня передачі даних (спершу у вигляді гіпертекстових документів формату HTML, наразі використовується для передачі будь-яких даних).

В основі протоколу – технологія «клієнт-сервер». Клієнти ініціюють з’єднання і відправляють запити, сервери – очікують з’єднання для отримання запиту, виконують якісь дії та відправляють клієнту повідомлення з результатом. Між цими запитами і відповідями як правило існує багато посередників. Їх називають проксі, вони виконують різноманітні функції і працюють як шлюзи або кеш (приміром).

**2.2 Схема роботи**

Процес відправки запиту можна розбити на декілька частин:

1. В першу чергу здійснюється DNS-запит, котрий повинен перетворити адресу сайту із URI на IP.
2. Після визначення IP встановлюється зв’язок між сервером і HTTP клієнтом.
3. Пересилання запиту.
4. Затримка, в яку входить пересилка інформації на сервер, її обробка і відправлення відповіді на запит.
5. Отримання відповіді на запит.

**Загальна структура HTTP повідомлення від клієнта виглядає так:**

1. Рядок запиту – вказує метод передачі, URL, що вказує на ресурс, який цікавить клієнта і версія протоколу HTTP.
2. Заголовки – спеціальні параметри, що містять якусь службову інформацію про з’єднання. Деякі з них є суто інформаційними для клієнта чи сервера, а інші є командами. Заголовок має вигляд «Параметр: значення». В залежності від розміщення, вони поділяються на:
   1. **General Headers** — повинні бути і в запитах клієнта, і у відповідях сервера.
   2. **Request Headers** — використовуються лише в запитах клієнта.
   3. **Response Headers** — використовуються лише у відповідях сервера.
   4. **Entity Headers** — супроводжують кожну сутність (entity) повідомлення.
3. Порожня лінія (насправді, символ CRLF), що позначає кінець мета-інформації.
4. Необов’язкове тіло повідомлення.

**Загальна структура HTTP повідомлення від сервера:**

1. Рядка стану HTTP відповіді, де сервер вказує версію HTTP і код відповіді (Напр. HTTP/1.1 301 Moved Permanently - means the requested resource was permanently moved and redirecting to some other resource). Коди помилок:

* хх – інформування про процес передачі.
* 2хх – успішне отримання й обробка запиту.
* 3хх – перенаправлення на інший URI.
* 4хх – помилка клієнта.
* 5хх – помилка на стороні сервера.

1. Кількох полів HTTP заголовка, розділених між собою символом CRLF (Напр. Content-Type: html).
2. Пустого рядка (тільки символ CRLF), що означає кінець полів заголовка.
3. Необов’язкове тіло HTTP повідомлення.

**2.3 Методи HTTP**

#### GET

Використовується для запиту контенту вказаного ресурсу. З його допомогою можна також почати якийсь процес. В такому разі в тіло відповіді треба включити інформацію про виконання процесу.

Клієнт може передавати параметри виконання запиту в URL ресурсу після символу «?»:

GET /path/resource?param1=value1&param2=value2 HTTP/1.1

Запити типу GET вважаються ідемпотентними.

#### HEAD

Аналогічний методу GET, за винятком того, що у відповіді сервера відсутнє тіло. Він використовується для отримання метаданих, валідації URL (перевірка існування ресурсу), і щоб з’ясувати, чи не змінився він з моменту останнього звернення.

* POST

Використовується для передачі клієнтських даних заданому ресурсу. Зазвичай його використовують для відправки форм або даних, які потребують більш сек’юрної передачі, адже, на відміну від методу GET, вони включаються в тіло запиту. Також за допомогою методу POST зазвичай завантажуються файли на сервер. Метод POST не є ідемпотентним. При результаті виконання у тіло відповіді треба включити повідомлення про цей результат.

#### PUT

Застосовується для завантаження контенту запиту на вказаний в запиті URL. Якщо ресурс не існує, то сервер його створює, і повертає статус 201(Created). Якщо ресурс було змінено, то сервер повертає код 200 (Ok) чи 204 (No Content). Сервер не повинен ігнорувати некоректні заголовки Content-\*,що передає клієнт із повідомленням. Якщо якийсь із заголовків не розпізнано, то повертається код помилки 501 (Not Implemented).

Фундаментальна відмінність від методу POST полягає в розумінні призначень URI ресурсів. POST припускає, що по вказаному URI буде відбуватись обробка даних. При використанні PUT ми заміняємо контент на даному ресурсі даними повідомлення.

#### PATCH

Аналогічно PUT, тільки застосовується до фрагмента ресурсу.

* DELETE

Видаляє ресурс.

#### TRACE

Повертає отриманий запит так, щоб клієнт міг побачити, яку інформацію додають, змінюють проміжні сервери в запиті.

#### CONNECT

Перетворює з’єднання запиту в прозорий TCP/IP-тунель, зазвичай, щоб сприяти встановленню захищеного SSL-з’єднання через нешифрований проксі.

**2.4 Заголовки HTTP**

|  |  |  |
| --- | --- | --- |
| Заголовок | Група | Опис |
| Allow | Entity | Список методів, що можна застосувати до ресурсу. |
| Content-Encoding | Entity | Застосовується, якщо необхідно перекодувати вміст. |
| Content-Language | Entity | Локалізація вмісту. |
| Content-Length | Entity | Розмір контенту (в октетах) |
| Content-Range | Entity | Діапазон (використовується для підтримання багато поточного завантаження). |
| Content-Type | Entity | Вказує тип вмісту (mime-type, наприклад, text/html). Часто включає посилання на таблицю символів локалі (charset). |
| Expires | Entity | Дата/час, після якої ресурс вважається застарілим. Використовується проксі-серверами. |
| Last-Modified | Entity | Дата/час останньої модифікації сутності. |
| Cache-Control | General | Визначає директиву управління механізмами кешування для проксі-серверів. |
| Connection | General | Задає параметри, необхідні для конкретного з’єднання. |
| Date | General | Дата і час формування повідомлення. |
| Pragma | General | Використовується для спеціальних інструкцій, котрі можуть бути застосовані до будь-якого одержувача по всьому ланцюжку запитів/відповідей (наприклад, pragma: no-cache). |
| Transfer-Encoding | General | Задає тип перетворення, яке можна застосувати до тіла повідомлення. На відміну від Content-Encoding, цей заголовок поширюється на все повідомлення, а не тільки на сутність. |
| Via | General | Використовується шлюзами і проксі для відображення проміжних протоколів і вузлів між клієнтом і веб-сервером. |
| Warning | General | Додаткова інформація про поточний статус, яка не може бути представлена в повідомленні. |
| Accept | Request | Визначає застосовні типи даних, очікувані у відповіді |
| Accept-Charset | Request | Визначає кодування символів (charset) для даних, очікуваних у відповіді. |
| Accept-Encoding | Request | Визначає можливі формати кодування/декодування вмісту (напр, gzip) |
| Accept-Language | Request | Можливі мови. |
| Authorization | Request | Облікові дані клієнта, який робить запит. |
| From | Request | Електронна адресі відправника. |
| Host | Request | Им’я/мережева адреса [і порт] сервера. За замовчуванням - 80. |
| If-Modified-Since | Request | Використовується для виконання методів з умовою (Якщо змінився...). Якщо ресурс змінився, то він передається з сервера, інакше - із кешу. |
| Max-Forwards | Request | Представляє механізм обмеження кількості шлюзів і проксі при використанні методів TRACE і OPTIONS. |
| Proxy-Authorization | Request | Використовується при запитах, що проходять через проксі, і які вимагають авторизації. |
| Referer | Request | Адреса, з якої виконується запит. Цей заголовок відсутній, якщо перехід виконується з адресного рядка чи, наприклад, за посиланням із js-скрипта. |
| User-Agent | Request | Інформація про агента користувача (клієнта). |
| Location | Response | Адрес пере направлення. |
| Proxy-Authenticate | Response | Повідомлення про статус з кодом 407. |
| Server | Response | Інформація про ПЗ сервера, який відповідає на запит (це може бути як веб, так і проксі сервер). |

**2.5 Безпека**

Сам по собі HTTP протокол не використовує шифрування для передачі інформації. Проте для HTTP існує популярне розширення, яке упаковує дані, що передаються, у криптографічний протокол **SSL** чи **TLS**  
 Назва цього розширення — **HTTPS** (*HyperText Transfer Protocol Secure*). Для HTTPS-з’єднань. Зазвичай застосовується порт 443. HTTPS широко застосовується для захисту інформації від перехоплення, а також, як правило,забезпечує від атак типу **man-in-the-middle** – вид компрометації зв’язку, при якому хакер таємно втручається в протокол передачі даних, і змінює або видаляє інформацію.

Наразі HTTPS підтримується всіма популярними веб-браузерами.  
  
**2.6 Додаткові можливості HTTP**

Протокол HTTP легко і зручно розширювати. Так,специфікація HTTP передбачає можливість використання заголовку Upgrade для переключення на обмін даними за іншим протоколом. Запит з таким заголовком відправляється клієнтом. Якщо серверу потрібно змінити протокол, він може повернути клієнту відповідь зі статусом 426 «Upgrade Required», і тоді клієнт зможе надіслати новий запит уже із цим заголовком. Така можливість використовується, наприклад, для організації обміну даними по протоколу WebSocket(протокол, описаний у специфікації RFC 6455, який дозволяє обом сторонам передавати дані у потрібний момент, без відправлення додаткових HTTP запитів). Стандартне рукостискання зводиться до відправлення HTTP запиту з заголовком Upgrade, який має значення «websocket», на що сервер посилає відповідь із кодом «101 Switching Protocols», і далі будь-яка сторона вже може починати передачу даних за протоколом WebSocket.

**2.7 Особливості протоколу HTTP**

Більшість протоколів передбачають встановлення TCP-сесії, в ході якої 1 раз відбувається авторизація, і всі подальші дії відбуваються в контексті цієї авторизації. Часто HTTP встановлює окрему TCP-сесію на кожен запит. В більш пізніх версіях HTTP було дозволено робити декілька запитів у рамках однієї TCP-сесії, але браузери зазвичай запитують тільки сторінку і включені в неї об’єкти (картинки, css та ін), а потім одразу розривають сесію. Для підтримки авторизованого (не анонімного) доступу в HTTP зазвичай використовують cookies – причому такий спосіб авторизації дозволяє зберегти сесію навіть після перезавантаження клієнта і сервера.

При доступі до даних по FTP чи за файловими протоколами, тип даних у файлі визначається за розширенням, що не завжди зручно. HTTP перед тим, як передати самі дані, передає заголовок «Content-type: тип/підтип», що дозволяє клієнту однозначно визначити, яким чином обробляти дані. Це особливо важливо при роботі із CGI-скриптами, коли розширення імені файлу вказує не на тип даних, що відправляються клієнту, а на необхідність запуску даного файлу на сервері і надіслання клієнту результату роботи програми, записаної в цьому файлі (При цьому один і той самий файл може створювати відповіді різних типів, в залежності від аргументів запиту і своїх власних «міркувань»)

Крім того, HTTP дозволяє клієнту присилати на сервер параметрі, які будуть передані скрипту. Для цього ж в HTML було введено форми.

Ці особливості HTTP дозволили створити пошукові машини, форуми і Інтернет-магазини. Це комерціалізувало Інтернет, призвело до створення компаній, що забезпечують доступ у Інтернет (провайдерів), створення сайтів і тд.

**2.8 Приклад**

Звичайний GET-запит

Запит:

[GET](https://ru.wikipedia.org/wiki/HTTP#GET) /wiki/*страница* HTTP/1.1

Host: ru.wikipedia.org

User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5

Accept: text/html

Connection: close

*(пустая строка)*

Відповідь:

HTTP/1.1 200 OK

Date: Wed, 11 Feb 2009 11:20:59 GMT

Server: Apache

X-Powered-By: PHP/5.2.4-2ubuntu5wm1

Last-Modified: Wed, 11 Feb 2009 11:20:59 GMT

Content-Language: ru

Content-Type: text/html; charset=utf-8

Content-Length: 1234

Connection: close

*(пустая строка)*

*(запрошенная страница в* [*HTML*](https://ru.wikipedia.org/wiki/HTML)*)*

***Розділ 3: Порівняння протоколів FTP і HTTP***

***3.1 Швидкість передачі***

* Що робить FTP швидким?

1. В потоці, що передається немає мета-інформації, тільки чисті бінарні дані. Довідкові дані ідуть через окреме з’єднання
2. Немає накладних витрат на перекодування даних.

* Що робить HTTP швидким?

1. Повторне використання існуючих постійних з’єднань підвищує продуктивність TCP, не витрачається час на встановлення новіх з’єднань
2. Конвеєрна обробка дозволяє швидше робити запит на отримання кількох файлів з одного й того ж сервера
3. Автоматичне стискання трафіку зменшує об’єм даних, що передаються. Це може присокорити передачу за наявності швидких клієнта і сервера і повільного каналу зв’язку
4. Немає команд в потоці передачі даних. Це економить час обробки

Врешті-решт, результат залежить від конкретних деталей, проте я б сказав, що для поодиноких файлів не можна помітити якусь різницю.

Для поодиноких файлів невеликого розміру і повільного з’єднання FTP буде краще. При отриманні кількох файлів поспіль (Особливо невеликих) HTTP зазвичай показує кращі результати.

***3.2 Вік протоколів***

FTP (RFC959) з’явився за 10 років до появи HTTP. На той час FTP був єдиним протоколом в Інтернеті. Перші зачатки того, що перетворилося на документ RFC959, можна знайти в далекому 1971 році.

***3.3 Заливка***

Обидва протоколи вміють це робити. У FTP є команда "append", HTTP має підхід «Ось вам дані, а ви самі розбирайтесь, що з ними робити», тобто жодних команд по управлінню файлами, що заливаються, немає. Варто сказати, що існує протокол WebDAV, побудований над HTTP, котрий дозволяє працювати з файлами в традиційній манері, наче вони знаходяться на вашому локальному пристрої.

***3.4 Формати ASCII, EBCDIC чи бінарний***

FTP має уявлення про формат файлу, тому може передавати дані як в ASCII, так і в двійковому форматі (raw bytes). HTTP ж завжди відправляє файли в двійковому вигляді. Таким чином, FTP вміє перетворювати дані «на льоту», якщо вони передаються між системами з різними архітектурами (Windows/Linux/мейнфрейми).

Наприклад, якщо відправник використовує одну схему для кодування кінця рядка, а отримувач – іншу, то FTP зробить так, що вони один одного зрозуміють. Unix використовує тільки символ NL, а MS Windows – 2 симовли підряд, CR і LF. EBCDIC перекодування використовується на старих мейнфреймах. HTTP, на противагу FTP, надає метадані для файлів, “Content-Type”. Таким чином, метадані можуть використовуватись клієнтами для інтерпретації вмісту.

***3.5 Заголовки***

Передача файлів через HTTP завжди включає в себе набір заголовків, в яких знаходяться метадані. FTP ніколи не передає ніяких заголовків. В зв’язку з цим, при передачі великої кількості малих файлів, їх заголовки складають значну частину трафіку. В заголовках HTTP знаходиться інформація про час і дату модифікації файлів, кодуванні символів, імені сервера, його версій і т.д.  
***3.6 Пайплайни або конвейери***

HTTP підтримує конвеєрну обробку даних. Це означає. Що клієнт можу запросити нову передачу файлів до того, як завершиться попередня, що дає можливість прибрати затримки при закачуванні декількох документів поспіль. TCP пакети, таким чином, будуть оптимізовані для максимальної швидкості передачі.

Щось подібне, але не зовсім, є і в FTP. Це підтримка множинних запитів для паралельного отримання файлів в одному керуючому з’єднанні. Звісно, для цього треба використовувати нові ТСР з’єднання для передачі бінарних даних, по одному для кожного файлу, проте, далеко не всі FTP-сервери підтримують це.

***3.7 FTP команди/відповіді***

FTP клієнт можу посилати на сервер багато команд і отримувати на них відповіді від сервера. Навіть передача одного файлу включає в себе цілу серію таких простих команд. Це, звичайно, негативно позначається на швидкості, тому що кожна команда вимагає обробки на двох сторонах: клієнті і сервері. Через це виникають затримки. HTTP передачі даних - це переважно, тільки один запит і одна відповідь (для кожного файлу). Отримання одного файлу через FTP іноді може займати до десятка команд і відповідей між клієнтом і сервером.

***3.8 2 з’єднання***

Одна з найбільших проблем для FTP в реальній роботі - це використання двох з'єднань. Перше - для відправки керуючих команд, а друге - для передачі вмісту файлу. Для цієї мети він щоразу відкриває окремий потік TCP. Якщо ви передаєте 100 файлів, по черзі будуть відкриті і закриті 100 TCP з'єднань.

***3.9 Фаєрволи і NAT***

## FTP використовує два з'єднання: керуюче та для передачі даних. З'єднання для даних може йти в двох напрямках, і використовувати динамічні номера портів. Це додає головного болю адміністраторам і часто вимагає від фаєрволів розуміння специфіки функціонування FTP на рівні мережевого протоколу, щоб забезпечити нормальну роботу.

## Це також означає, що якщо обидві сторони з'єднання знаходяться за NAT, ви, швидше за все, не зможете користуватися FTP.

## Крім того, NAT «вбиває» незайняті з'єднання, через які тривалий час не було передачі даних. Тому, під час довгих передач по FTP на повільних каналах зв'язку ми опиняємося в ситуації, коли з'єднання виявляється розірваним, тому що NAT вирішив, що воно вже неактивно.

## Щоб такого не відбувалося, доводиться час від часу відправляти фіктивні порожні команди, щоб з'єднання підтримувалося в "живому" стані. Результат - невеликий, але зайвий трафік.

***3.10 Активний і пасивний режими***

FTP відкриває друге з'єднання в активному або пасивному режимі. Якщо працює активний режим (з'єднання ініціює сервер) - будуть проблеми із з'єднанням у складних мережах, тому що таке поєднання неможливо через NAT. Тому, в більшості випадків використовується пасивний режим (passive mode), коли з'єднання відбувається тільки з боку клієнта

***3.11 Зашифровані керуючі з’єднання***

## Оскільки брандмауери повинні вміти розбирати керуюче з'єднання FTP, щоб дати можливість коректно відкривати друге з'єднання для передачі бінарних даних, існує величезна проблема з зашифрованими сполуками (FTP-SSL або FTPS). Щойно керуюче з'єднання стає зашифрованим, фаєрвол вже не в змозі інтерпретувати його команди, щоб розуміти, коли і як слід дозволити друге з'єднання між клієнтом і сервером для передачі бінарних даних.

## До того ж, розробка самого стандарту FTPS зайняла занадто багато часу, що призвело до одночасного існування кількох гібридних версій, погано сумісних між собою.

***3.12 Схеми авторизації***

## У FTP і HTTP є кілька задокументованих методів аутентифікації. Обидва протоколи пропонують базову аутентифікацію звичайним текстом (логін / пароль). Однак, для HTTP існують кілька часто використовуваних методів перевірки, що не відправляють пароль у вигляді звичайного тексту, на відміну від FTP.

***3.13 Скачування***

Обидва протоколи вміють це робити. У обох протоколів були проблеми при скачуванні файлів з розміром, більше ніж 2 гігабайти, але це вже в минулому. В сучасних клієнтах і серверах, на сучасних операційних системах цієї проблеми більше немає.

***3.14 Діапазони/поновлювання скачування***

FTP підтримує скачування і закачування, а також відновлення розірваних з'єднань і продовження передачі в обох напрямках. HTTP може похвалитися тільки відновленням при скачуванні, а при заливці файлів на сервер відновлення обірваного з'єднання і продовження заливки часто виявляються неможливими.

На противагу FTP, HTTP підтримує більш просунуті діапазони для скачування.

Також у FTP є проблеми при відновленні з'єднань при заливці або скачуванні файлів, починаючи з сегмента, більшого, ніж 2 GB.

***3.15 Постійні з’єднання***

HTTP клієнт може тримати одне постійне з'єднання з сервером для будь-якої кількості передач файлів.

FTP повинен створювати нове з'єднання для кожної нової передачі. Багаторазові виконання нових підключень погано позначаються на продуктивності через необхідність "рукостискань" (handshakes) для TCP з'єднань.

***3.16 Кодування HTTP-блоків***

## Щоб уникнути закриття з'єднання, коли у вас немає можливості повідомити віддаленої стороні про те, що передача файлу вже завершена, в HTTP було введено так зване кодування переданих блоків (чанків) з даними.

## Під час передачі сторона, яка віддає потік даних блоками (розмір блоку + самі дані) до тих пір, поки вони не закінчаться, а потім передає блок з нульовою довжиною, щоб просигналізувати про кінець файлу.

## Крім того, що з'єднання не потрібно закривати і відкривати заново для нових файлів, ще одним очевидним плюсом такої схеми є можливість виявлення передчасних аварійних відключень в процесі передачі.

***3.17 Стискання***

## HTTP надає серверу та клієнту можливість домовитися і вибрати один з алгоритмів стиснення. Алгоритм gzip є, мабуть, найбільш широко використовуваним. Є більш сучасний brotli, але він ще не повністю підтримується різними серверами і клієнтами, хоча дає краще стиснення (до + 20%), особливо на текстових html, javascript і css файлах.

## FTP надає офіційне вбудоване RLE стиснення, однак воно зазвичай неефективне для більшості бінарних і текстових даних. Є багато додаткових "хакерських" реалізацій для стиснення FTP трафіку, але жодна з них не стала офіційною і широко не використовується.

***3.18 FXP***

## FTP підтримує технологію для передачі даних з одного сервера на інший, як ніби-то передачу веде безпосередньо сам клієнт. Однак на більшості серверів ця можливість закрита через проблеми з безпекою, так як протокол FXP був недостатньо добре спроектований.

***3.19 IPv6***

## І HTTP, і FTP відмінно працюють з IPv6, проте спочатку в оригінальній специфікації протоколу FTP не було підтримки IPv6, і через це безліч серверів до сих пір не мають потрібних команд для його включення. Це також стосується міжмережевих екранів між клієнтами і серверами, які повинні розуміти FTP

***3.20 Віртуальний хостинг на основі імені***

## Використовуючи HTTP 1.1, ви легко можете розмістити безліч сайтів на одному сервері, і всі вони будуть відрізнятися за їхніми іменами.

В FTP ви взагалі не можете використовувати віртуальний хостинг на основі імен, поки команда HOST не буде реалізована на сервері, з яким ви з'єднані. Це свіжа специфікація, і вона ще мало поширена.

***3.21 Перегляд каталогів***

## В FTP можна отримати список файлів з папки на віддаленому сервері, без скачування їх, в той час як в HTTP немає такої можливості.

## Однак, в силу того, що автори специфікації FTP жили в різний час, команди для отримання списку файлів в каталозі (LIST і NLST) не мають чітко описаного формату виведення. Тому авторам FTP клієнтів доводиться займатися написання синтаксичних аналізаторів тексту, щоб спробувати правильно вгадати, що за дані їм передає сервер. Пізніші специфікації (RFC3659) передбачають нові команди типу MLSD, але вони ще не набули широкого поширення і погано підтримуються різними серверами і клієнтами.

## Списки файлів в каталогах через HTTP зазвичай передаються текстом в HTML форматі, або за допомогою WebDAV, який працює над HTTP.

***3.21 Підтримка проксі***

## Одна із серйозних переваг HTTP перед FTP - це підтримка проксі, вбудована в нього з самого початку. Технологія налагоджена і дуже добре працює. Багато протоколів можуть бути вміщені всередину HTTP, як на своєрідний "конверт" для проходження проксі-серверів.

## FTP завжди використовувався з проксі серверами, але це ніколи не було стандартизовано, і завжди вимагало спеціальних підходів в кожному конкретному випадку.

## 

***Висновки***

Мною було досліджено тему «Порівняння протоколів FTP і HTTP». В моїй роботі описано визначення цих термінів, схеми роботи, основні механізми взаємодії між клієнтом і сервером – команди і коди відповідей, а також висвітлено проблеми безпеки і шляхи їх подолання.

При порівнянні протоколів я проаналізував різні режими роботи, види з’єднань, швидкість передачі, можливості протоколів, надійність, підтримку проксі та ін., виявив сильні і слабкі сторони FTP і HTTP. Серед них:

* розширюваність протоколу HTTP
* навантаження трафіку HTTP-заголовками
* застарілість FTP (невдале використання TCP з’єднань, велика кількість повідомлень, якими обмінються сервер і клієнт, неможливість змінити протокол на інший)
* несумісність надбудов над FTP
* здатність FTP працювати з даними не лише в бінарному форматі
* нестійкість протоколів до різного роду атак

та ін.

Виявилось, що для подолання всіх недоліків недостатньо було створити новіші версії – довелося створити декілька надбудов над кожним з протоколів. А саме: HTTPS, SFTP, FTPS, WebDAV та ін.

Отже, не можна однозначно сказати, який протокол є кращим за інші: залежно від ситуації потрібно обирати чи FTP, чи HTTP.

***Список використаних джерел***

1. <https://developer.mozilla.org/ru/docs/Web/HTTP/Overview>
2. <http://www.4stud.info/web-programming/protocol-http.html>
3. <https://habr.com/ru/post/215117/>
4. https://nsoft-s.com/mychatarticles/1164-http-vs-ftp.html#:~:text=FTP%20%D0%B8%D0%BC%D0%B5%D0%B5%D1%82%20%D0%BF%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%BE%20%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B5,Windows%2FLinux%2F%D0%BC%D1%8D%D0%B9%D0%BD%D1%84%D1%80%D1%8D%D0%B9%D0%BC%D1%8B).
5. <https://ru.wikipedia.org/wiki/HTTP>
6. <https://webkyrs.info/post/chto-takoe-zagolovki-v-protokole-http>
7. <https://www.webnots.com/what-is-http/>
8. [https://zametkinapolyah.ru/servera-i-protokoly/tema-5-http-otvety-servera-stroka-sostoyaniya-http-otveta-kody-sostoyaniya-zagolovki-http-otveta-primery.html#:~:text=%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0%20HTTP%20%D0%BE%D1%82%D0%B2%D0%B5%D1%82%D0%B0%20%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%B0%20%D1%81%D0%BE%D1%81%D1%82%D0%BE%D0%B8%D1%82%20%D0%B8%D0%B7%3A&text=%D0%9D%D1%83%D0%BB%D1%8F%20%D0%B8%D0%BB%D0%B8%20%D0%BD%D0%B5%D1%81%D0%BA%D0%BE%D0%BB%D1%8C%D0%BA%D0%B8%D1%85%20%D0%BF%D0%BE%D0%BB%D0%B5%D0%B9%20HTTP,%D0%9D%D0%B5%D0%BE%D0%B1%D1%8F%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5%20%D1%82%D0%B5%D0%BB%D0%BE%20HTTP%20%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D1%8F](https://zametkinapolyah.ru/servera-i-protokoly/tema-5-http-otvety-servera-stroka-sostoyaniya-http-otveta-kody-sostoyaniya-zagolovki-http-otveta-primery.html#:~:text=%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0%20HTTP%20%D0%BE%D1%82%D0%B2%D0%B5%D1%82%D0%B0%20%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%B0%20%D1%81%D0%BE%D1%81%D1%82%D0%BE%D0%B8%D1%82%20%D0%B8%D0%B7%3A&text=%D0%9D%D1%83%D0%BB%D1%8F%20%D0%B8).
9. [Особенности работы HTTP протокола | Vaden Pro (vaden-pro.ru)](https://vaden-pro.ru/blog/php/http-protokol)