**ЛЕКЦІЯ 3: ПРОТОКОЛИ ПЕРЕДАЧІ ДАНИХ**

**ВСТУП**

Протоколи передачі даних є основою функціонування Інтернету та багатьох мережевих додатків. Вони визначають правила, за якими дані передаються між клієнтами та серверами, забезпечуючи надійність, цілісність і безпеку переданої інформації. Ця лекція охоплює основи протоколів TCP/IP, роботу з HTTP та HTTPS, а також протоколи прикладного рівня, такі як FTP, SMTP та WebSocket.

**3.1 ОСНОВИ ПРОТОКОЛІВ TCP/IP**

**Протоколи TCP/IP** (Transmission Control Protocol/Internet Protocol) є основою для передачі даних в Інтернеті та мережах загального користування. Вони складаються з декількох рівнів, кожен з яких відповідає за певний аспект передачі даних.

**Історія та еволюція TCP/IP**

Протокол TCP/IP був розроблений у 1970-х роках дослідниками DARPA (Агентство передових дослідницьких проектів Міністерства оборони США). Спочатку TCP/IP був розроблений для військових цілей, але швидко став стандартом для академічних та комерційних мереж.

**Чотирирівнева модель TCP/IP**

TCP/IP базується на чотирьох рівнях, кожен з яких відповідає за певний етап передачі даних:

1. **Канальний рівень** - Забезпечує фізичну передачу даних між пристроями в межах однієї мережі.
2. **Мережевий рівень** - Відповідає за маршрутизацію пакетів через різні мережі до їх кінцевої точки. Основний протокол цього рівня — IP (Internet Protocol).
3. **Транспортний рівень** - Забезпечує надійність передачі даних між пристроями. Основними протоколами цього рівня є TCP (Transmission Control Protocol) і UDP (User Datagram Protocol).
4. **Прикладний рівень** - Включає протоколи, які забезпечують взаємодію між додатками, такі як HTTP, FTP, SMTP.

**IP-адресація та маршрутизація**

IP-адресація використовується для ідентифікації пристроїв в мережі. Вона поділяється на IPv4 та IPv6. IPv4 використовує 32-бітні адреси, тоді як IPv6 — 128-бітні, що дозволяє підтримувати більшу кількість унікальних адрес.

Маршрутизація в TCP/IP забезпечує перенаправлення пакетів даних між мережами. Це досягається завдяки маршрутизаторам, які приймають рішення про найкращий шлях для передачі пакетів до їх призначення.

**Протокол TCP**

TCP є основним протоколом транспортного рівня. Він забезпечує надійну передачу даних, використовуючи механізми контролю помилок, сегментації даних і підтвердження отримання. TCP встановлює з'єднання між двома пристроями перед початком передачі даних, що робить його більш надійним, але повільнішим, ніж UDP.

**Протокол UDP**

UDP, на відміну від TCP, не встановлює з'єднання і не забезпечує надійність передачі. Він використовується в ситуаціях, де швидкість є критичною, наприклад, у стрімінгу відео чи аудіо.

**3.2 РОБОТА З HTTP ТА HTTPS**

**HTTP (HyperText Transfer Protocol)** - це основний протокол, який використовується для передачі гіпертекстових документів, таких як веб-сторінки, через Інтернет. Він функціонує на прикладному рівні моделі TCP/IP.

**Історія HTTP**

HTTP був вперше введений у 1990-х роках як частина проекту World Wide Web, розробленого Тімом Бернерсом-Лі. Початкова версія HTTP (HTTP/0.9) була дуже простою і підтримувала лише основні функції, такі як отримання HTML-документів. Сучасні версії HTTP, такі як HTTP/1.1 та HTTP/2, значно розширили можливості цього протоколу, забезпечуючи більш ефективну передачу даних і підтримку сучасних веб-технологій.

**Механізм роботи HTTP**

HTTP є безстатевим протоколом, що означає, що кожен запит і відповідь є незалежними. Це робить HTTP простим у використанні, але додає певні виклики, такі як потреба в механізмах збереження стану, наприклад, через використання cookies.

**Основні методи HTTP**

HTTP підтримує різні методи для виконання різних операцій, серед яких:

* **GET**: Запит на отримання даних з сервера.
* **POST**: Надсилання даних на сервер для обробки.
* **PUT**: Оновлення або створення ресурсу на сервері.
* **DELETE**: Видалення ресурсу на сервері.

**Проблеми безпеки HTTP**

Оскільки HTTP не шифрує дані, передані між клієнтом і сервером, він є вразливим до атак, таких як перехоплення даних (man-in-the-middle attacks). Для вирішення цих проблем був розроблений HTTPS.

**HTTPS (HyperText Transfer Protocol Secure)** - це розширення HTTP, яке використовує SSL/TLS для шифрування даних. Це забезпечує захищений канал передачі даних, що робить його незамінним для веб-сайтів, які обробляють конфіденційну інформацію, таку як логіни або платіжні дані.

**Механізм роботи HTTPS**

HTTPS працює на тому ж рівні, що й HTTP, але включає в себе додатковий етап — шифрування даних за допомогою SSL/TLS. Це шифрування гарантує, що дані не можуть бути прочитані або змінені під час їх передачі між клієнтом і сервером.

**Впровадження HTTPS**

Перехід з HTTP на HTTPS потребує придбання SSL-сертифікату і налаштування веб-сервера для використання цього сертифікату. SSL-сертифікати можуть бути отримані у сертифікаційних центрів і можуть мати різні рівні верифікації, від базової до розширеної.

**3.3 (FTP, SMTP, WebSocket)**

**FTP (File Transfer Protocol)**

FTP є одним з найстаріших і найбільш використовуваних протоколів для передачі файлів між клієнтом і сервером. Він функціонує на прикладному рівні TCP/IP і забезпечує надійну передачу файлів через мережу.

**Основні функції FTP**

FTP підтримує різні режими передачі, такі як активний і пасивний режими, що дозволяє забезпечувати безперебійний обмін даними навіть за умов обмеженого доступу до мережі. Протокол також підтримує аутентифікацію користувачів, дозволяючи контролювати доступ до серверів.

**Переваги та недоліки FTP**

Переваги FTP включають високу надійність передачі файлів і підтримку великих обсягів даних. Проте протокол не шифрує дані за замовчуванням, що робить його вразливим до перехоплення.

**SMTP (Simple Mail Transfer Protocol)**

SMTP є стандартним протоколом для передачі електронної пошти через Інтернет. Він працює на прикладному рівні TCP/IP і забезпечує надійну передачу повідомлень між поштовими серверами.

**Принципи роботи SMTP**

SMTP функціонує на основі моделі клієнт-сервер, де клієнт (поштовий клієнт або сервер) відправляє повідомлення на сервер одержувача через низку проміжних серверів. SMTP забезпечує перевірку правильності адреси одержувача, а також обробку помилок під час передачі.

**Розширення та безпека SMTP**

SMTP був розширений через додаткові протоколи, такі як SMTPS (SMTP Secure), що використовує SSL/TLS для забезпечення безпечної передачі повідомлень.

**WebSocket**

WebSocket є сучасним протоколом, який забезпечує двонаправлене спілкування між клієнтом і сервером у режимі реального часу. Він функціонує на прикладному рівні TCP/IP і є ідеальним для додатків, які потребують постійного оновлення даних, таких як чати, онлайн-ігри або біржова торгівля.

**Протокол WebSocket** був розроблений як рішення для додатків, які потребують постійного зв'язку між клієнтом і сервером, що неможливо забезпечити за допомогою традиційних HTTP-запитів. WebSocket дозволяє встановлювати стійке з'єднання між клієнтом і сервером, через яке дані можуть передаватися в обох напрямках без необхідності встановлення нового з'єднання для кожного запиту.

**Архітектура та механізм роботи WebSocket**

**Архітектура WebSocket** кардинально відрізняється від традиційного HTTP, оскільки вона дозволяє встановлювати стійке, двонаправлене з'єднання між клієнтом і сервером. Це досягається за рахунок специфічного "потиску рук" (handshake) на початку сесії, під час якого клієнт запитує сервер про можливість використання WebSocket. Якщо сервер підтримує цей протокол, він відповідає з підтвердженням, і з'єднання встановлюється.

**Основні кроки встановлення з'єднання WebSocket**:

1. **HTTP-запит**: Клієнт надсилає стандартний HTTP-запит із заголовком Upgrade: websocket. Це є сигналом для сервера, що клієнт хоче встановити WebSocket-з'єднання.
2. **Підтвердження від сервера**: Якщо сервер підтримує WebSocket, він відповідає HTTP-відповіддю з кодом 101 Switching Protocols, що підтверджує перехід до протоколу WebSocket.
3. **Встановлення з'єднання**: Після цього з'єднання стає стійким, і клієнт і сервер можуть обмінюватися даними в обох напрямках без необхідності повторного відкриття з'єднання.
4. **Закриття з'єднання**: Обидві сторони можуть ініціювати закриття з'єднання, після чого всі ресурси звільняються.

**Переваги WebSocket**:

1. **Низькі затримки**: Оскільки з'єднання стійке, немає потреби витрачати час на встановлення нового з'єднання для кожного запиту, що значно зменшує затримки при передачі даних.
2. **Ефективність**: WebSocket дозволяє передавати дані у форматі бінарних або текстових повідомлень, що робить його більш гнучким і менш затратним у порівнянні з традиційними HTTP-запитами.
3. **Підтримка реального часу**: Завдяки двонаправленості з'єднання WebSocket є ідеальним рішенням для додатків, які потребують постійного оновлення даних у режимі реального часу.

**Недоліки WebSocket**:

1. **Складність впровадження**: Оскільки WebSocket передбачає використання спеціалізованих серверів і бібліотек, його впровадження може бути складнішим у порівнянні з традиційними HTTP-додатками.
2. **Безпека**: Незважаючи на підтримку шифрування через TLS (wss://), WebSocket-з'єднання можуть бути вразливими до певних типів атак, таких як атаки "людина посередині" (MITM), якщо не налаштовані належним чином.

**Протоколи передачі даних: Порівняння та висновки**

Протоколи передачі даних, такі як TCP/IP, HTTP, HTTPS, FTP, SMTP та WebSocket, відіграють ключову роль у забезпеченні роботи Інтернету та мережевих додатків. Кожен з них має свої переваги та недоліки, а також сфери застосування, які визначають їхню ефективність у тих чи інших умовах.

* **TCP/IP** є основним транспортним протоколом, що забезпечує надійну передачу даних між пристроями, а також маршрутизацію пакетів через складні мережеві інфраструктури.
* **HTTP/HTTPS** є стандартом для веб-технологій, який забезпечує передачу гіпертексту та інших даних через Інтернет. HTTPS додає додатковий рівень безпеки, що є необхідним для захисту конфіденційної інформації.
* **FTP** залишається популярним для передачі великих файлів, хоча його використання зменшується через відсутність вбудованих засобів шифрування.
* **SMTP** є основним протоколом для передачі електронної пошти, що забезпечує надійну доставку повідомлень між серверами.
* **WebSocket** забезпечує новий рівень взаємодії в режимі реального часу, що робить його ідеальним для додатків, які потребують швидкого та безперервного обміну даними.

**Висновок**

Вивчення протоколів передачі даних є ключовим для розуміння сучасних мережевих технологій і додатків. Ці протоколи формують основу Інтернету, забезпечуючи ефективну і безпечну передачу інформації між пристроями. Вони постійно розвиваються, щоб відповідати новим вимогам і викликам, що виникають у міру розвитку технологій і збільшення обсягу переданих даних. Розуміння принципів роботи цих протоколів є важливим для майбутніх розробників, адміністраторів мереж і фахівців з кібербезпеки, оскільки вони формують основи мережевої архітектури та взаємодії в сучасному цифровому світі.