Notes - Session 9 - Communication Protocols

Communication Protocols:

Transmission Control Protocol (TCP):

Example: Phone Call

TCP is a connection-oriented, reliable, byte-stream protocol. It's one of the main protocols in the Internet protocol suite and operates on top of the Internet Protocol (IP)

Features:

- **Reliability:** It ensures data is delivered without errors and in the correct order. Lost packets are retransmitted

- **Connection-oriented:** A connection is established between the sender and the receiver before any data is sent

- **Flow Control:** It ensures data is sent at a rate the receiver can handle

- **Ordered Data Transfer:** If packets are sent in the order A, B, C, they'll be received in the same order

- **Error Checking:** Checksums are used to verify data integrity

Use Cases:

- Web browsing (HTTP/HTTPS operates over TCP)
- File transfer (e.g., FTP)
- Email (e.g., SMTP)

WebSockets

Example: Walkie Talkie

WebSockets provide a full-duplex communication channel over a single, long-lived connection, designed to work over the same ports as HTTP and HTTPS.

Features:

- **Full-duplex:** Allows simultaneous bidirectional communication

- **Low Latency:** Reduces the overhead and latency introduced by opening a new connection or using a polling mechanism

- **Persistent Connection:** The connection remains open, allowing real-time data transfer as soon as the server has new information

- **Operates over HTTP:** Begins with an HTTP handshake (Upgrade header) and then upgrades to the WebSocket protocol

Use Cases:

- Real-time applications such as chat applications, online gaming, and live sports updates
- Financial tickers
- Collaborative tools like real-time document editors

How does long polling?

1. Client Request: In long polling, the client sends a request to the server, just like in traditional polling. However, instead of responding immediately with available data or a lack of data, the server holds the request open.

2. Server Holding Request: The server waits until it has new data to send or until a certain condition is met (like a timeout). This is the key difference from regular polling, where the server responds immediately whether or not there's new data.

3. Server Response: Once new data is available or the condition is met, the server responds to the client's request, sending the data back.

4. New Request: After the client processes the server's response, it sends another long polling request, and the cycle repeats

Advantages of Long Polling

- **Reduced Latency:** Since the server sends data as soon as it's available, there's less latency in communication compared to traditional polling

- **Efficiency:** This method can be more efficient than regular polling, especially in scenarios where updates are infrequent. It avoids the overhead of the client continuously checking for data when there's none

Disadvantages

- **Resource Intensive on the Server:** Holding connections open can be resource-intensive for the server, especially if there are many clients

- **Complexity:** Implementing long polling can be more complex than traditional polling or other alternatives like WebSockets

- **Scalability Issues:** As the number of clients increases, the number of open connections can strain the server, impacting scalability

Unified Data Protocol (UDP):

Example: Postcards, One way lectures by mentors

UDP is a connectionless, unreliable protocol. It sends datagrams without establishing a connection and without guaranteeing delivery or ordering

Features:

- **Connectionless:** No connection is established, and no connection teardown is required

- **Unreliable:** Delivery is not guaranteed. Datagrams might be lost, duplicated, or received out of order

- **Low Overhead:** Because of its simplicity, it has less overhead and can be faster for some applications

- **No Error Recovery:** The application must implement its own error-checking and correction if needed

Use Cases:

- Streaming media (e.g., video, voice)
- Online gaming
- Broadcasting
- Some DNS queries and responses

System Design Considerations

Choosing the Right Protocol:

In system design, choosing between TCP, UDP, and WebSockets depends on the application's requirements.
TCP is used for applications where reliability and order are critical,
UDP is chosen for speed

WebSockets are ideal for real-time, bi-directional communication.

Scalability and Performance:

Each protocol impacts the scalability and performance of the system differently. For instance, TCP's error-checking mechanism might slow down the data transfer.

Security Considerations:

Security measures vary across these protocols. While TCP/IP and WebSockets can implement robust security protocols, UDP is inherently less secure due to its connectionless nature.