Lecture Notes: Application Layer

1. P2P (Peer-to-Peer) Applications

- **P2P Architecture**:
  - There is no central server; instead, computers, called **peers**, communicate directly with each other.
  - Each peer can be both a client (requesting data) and a server (providing data).
  - This architecture scales well, as each new peer adds more resources.
  - **Example**: File-sharing applications like BitTorrent, where users share parts of files directly with each other.

2. File Distribution: Client-Server vs. P2P

- **Client-Server Model**:
  - The server sends the entire file to each client.
  - Time to distribute a file increases with the number of clients, as the server's upload capacity is limited.
  - **Example**: Downloading a file directly from a single website.
- **P2P Model**:
  - Each peer downloads parts of the file and also uploads it to others.
  - This model can handle many clients efficiently because all peers share the work.
  - **Example**: Using BitTorrent to download a large file from multiple sources.

3. Video Streaming and Content Distribution Networks (CDNs)

- **Video Streaming**:
  - Video streaming is a major use of internet bandwidth (e.g., Netflix, YouTube).
  - Challenges include:
    - **Scalability**: Serving millions of users.
    - **Heterogeneity**: Different users have different devices and internet speeds.
  - **Solution**: CDNs (Content Delivery Networks), which are networks of servers around the world that deliver content efficiently.

4. Multimedia: Video Compression and Coding

- **Video Structure**:
  - A video is made up of many images (frames) shown quickly to create motion.
  - **Frame Rate**: Number of images displayed per second (e.g., 24 frames/sec).
  - Each frame is an array of pixels, where each pixel represents color and brightness.
- **Video Compression**:
  - **Spatial Coding**: Reduces data within a single image by encoding repeated colors only once.
  - **Temporal Coding**: Saves data between images by only storing changes from one image to the next.

    - **CBR (Constant Bit Rate)**: The rate of data is fixed.
    - **VBR (Variable Bit Rate)**: The rate changes based on video complexity.

5. Streaming Stored Video

- **How It Works**:
    - Videos are sent in small parts (streams) so viewers can watch without waiting for the whole video to download.
    - While the server sends one part, the viewer can start watching an earlier part.
- **Challenges**:
    - **Continuous Playout**: Video should play smoothly without breaks, requiring a client-side buffer.
    - **Interactivity**: The viewer might pause, rewind, or skip ahead.
    - **Network Delay**: Variable network speeds (jitter) require adjustments for smooth playback.

6. DASH (Dynamic Adaptive Streaming over HTTP)

- **DASH Process**:
    - The video is divided into chunks at different quality levels.
    - The client device (e.g., phone or computer) monitors its internet speed and requests the best quality chunk it can handle.
- **Benefits**:
    - Adjusts video quality based on available bandwidth, so viewers get the best experience.
    - **Example**: YouTube videos adjusting resolution automatically based on internet speed.

7. Content Distribution Networks (CDNs)

- **CDN Purpose**:
    - CDNs store content (like videos) on servers worldwide to reach viewers faster and more reliably.
    - This prevents a single server from being overwhelmed and reduces the distance data must travel.
- **Two CDN Strategies**:
    - **"Enter Deep"**: CDN servers are placed close to users (e.g., in neighborhoods or cities).
    - **"Bring Home"**: Fewer, larger CDN servers are placed near major access points.
- **Example**: Akamai CDN, which has thousands of servers worldwide to handle large amounts of web traffic.

8. How Netflix Uses CDNs

- **Process**:
    - When you select a video on Netflix, the app checks your internet speed.

- Based on your speed, Netflix delivers the highest quality video possible from the closest CDN server.
- If the internet connection weakens, Netflix switches to a lower-quality version to prevent buffering.

9. CDN Content Access in Detail

- **Example Scenario**:
  - A user (like Bob) wants to watch a video hosted by a website (e.g., netcinema.com).
  - **Steps**:
    1. Bob clicks the link to the video.
    2. Bob's DNS server finds the nearest CDN server hosting the video.
    3. Bob is directed to the CDN server to access the video at optimal speed.

Lecture Notes: Domain Name System (DNS) and Beyond

1. DNS: Domain Name System

- **Purpose of DNS**:
  - Translates human-friendly domain names (like example.com) into IP addresses that computers use to locate each other on the internet.
  - Works like a "phonebook" for the internet, helping to map website names to their IP addresses.
- **Why DNS is Needed**:
  - IP addresses are hard for humans to remember.
  - Domain names are easier, so DNS allows users to type names instead of numbers.

2. DNS Services and Structure

- **DNS Services**:
  - **Hostname to IP Translation**: Converts domain names into IP addresses.
  - **Host Aliasing**: Provides alternative names for the same domain (e.g., www.example.com and example.com can point to the same site).
  - **Mail Server Aliasing**: Finds the correct server for sending emails to a domain.
  - **Load Distribution**: Distributes requests across multiple servers for popular websites to handle large volumes of traffic.
- **Structure**:
  - DNS is a distributed database, meaning it's spread across multiple servers instead of being in one location. This setup avoids a single point of failure and improves speed.

3. DNS Hierarchical Database

- **Hierarchy Levels**:

    - **Root Servers**: The top level of the DNS, containing information about top-level domains (TLDs) like .com, .org, .net, etc.
    - **Top-Level Domain (TLD) Servers**: Responsible for TLDs (e.g., .com or .org). They direct requests to authoritative servers.
    - **Authoritative Servers**: Contain specific information for a domain (e.g., DNS records for amazon.com).
- **How DNS Queries Work**:
    - A user's request goes through a series of steps in this hierarchy, starting from the root server and narrowing down to the authoritative server for the domain.

4. Root DNS Servers

- **Role of Root Servers**:
    - They act as a "last resort" for directing DNS queries when other servers don't know the answer.
    - Managed by **ICANN** (Internet Corporation for Assigned Names and Numbers) to ensure global internet stability.
- **Reliability**:
    - There are 13 main logical root servers worldwide, replicated many times to ensure availability.

5. TLD Servers and Authoritative Servers

- **TLD Servers**:
    - Manage top-level domains and direct requests to the authoritative servers.
    - For example, the .com TLD server handles requests for .com websites, forwarding them to the right place.
- **Authoritative Servers**:
    - These servers hold DNS information for specific organizations, like the IP address for example.com.
    - Maintained by the organization or a service provider.

6. Local DNS Servers

- **What They Do**:
    - These are typically managed by ISPs or local networks to handle DNS queries quickly for users.
    - The local DNS server caches recent responses to speed up future requests.

7. DNS Resolution Methods

- **Iterated Query**:
    - If a DNS server doesn't know the answer, it suggests another server to ask. The querying server (e.g., a user's local DNS) continues the process until it finds the answer.
- **Recursive Query**:

    - The DNS server takes on the responsibility of finding the answer and contacts other DNS servers if needed.
    - This process is faster for the user but increases the load on the DNS server.

8. Caching DNS Information

- **Purpose**:
    - Reduces response time for frequent queries by storing previous DNS answers temporarily.
    - Cached data has a "Time-To-Live" (TTL), after which it's removed to ensure data stays updated.
- **Limitations**:
    - If a website changes its IP address, cached information may be outdated until the TTL expires.

9. DNS Records

- **Types of DNS Records**:
    - **A Record**: Maps a hostname to its IP address (e.g., example.com → 192.0.2.1).
    - **NS Record**: Shows the authoritative DNS server for a domain.
    - **CNAME Record**: Provides an alias for a domain (e.g., www.example.com can point to example.com).
    - **MX Record**: Specifies the mail server for handling emails for the domain.

10. DNS Protocol Messages

- **Structure**:
    - Both DNS queries and replies use the same message format, which includes:
        - **Identification**: A unique ID for each query.
        - **Flags**: Indicate if the message is a query or response, and if recursion is requested or allowed.
        - **Sections**: The message includes sections for questions, answers, authority, and additional information.

11. Getting Information into DNS

- **Registering a Domain**:
    - When a new domain is registered, information about it is entered into the DNS.
    - The registrar (like Network Solutions) stores the domain's details, including the authoritative name servers and IP addresses.

12. DNS Security

- **Threats to DNS**:
    - **DDoS Attacks**: Flooding DNS servers with traffic to disrupt their service.

    - **Spoofing Attacks**: Sending fake DNS responses to misdirect users to malicious sites.
- **DNSSEC**:
    - DNSSEC (DNS Security Extensions) helps secure DNS by ensuring message integrity and authenticity, protecting users from spoofing.

Lecture Notes: Transport Layer (Chapter 3)

1. Introduction to the Transport Layer

- **Purpose**:
    - The transport layer manages data transfer between devices on a network. It ensures data is transferred reliably, in order, and at the correct speed.
    - It provides essential services like **error correction**, **flow control**, and **connection management**.
- **Key Topics in Transport Layer**:
    - Transport-layer services
    - Multiplexing and demultiplexing
    - Connectionless transport (UDP)
    - Connection-oriented transport (TCP)
    - Congestion control

---

2. TCP (Transmission Control Protocol) Overview

- **TCP Features**:
    - **Connection-Oriented**: TCP requires a connection (handshake) between sender and receiver before data transfer.
    - **Reliable Data Transfer**: Ensures all data arrives accurately and in order.
    - **Full Duplex**: Data can flow in both directions at the same time.
    - **Flow Control**: Prevents the sender from overwhelming the receiver.
    - **Congestion Control**: Adjusts data transfer rate to avoid network congestion.

---

3. TCP Segment Structure

- TCP segments contain various fields:
    - **Source and Destination Port Numbers**: Identify the sender and receiver.
    - **Sequence Number**: Keeps track of data order.
    - **Acknowledgment Number**: Confirms receipt of data.
    - **Flags**: Indicate control information (e.g., SYN, ACK).
    - **Receive Window**: Helps manage flow control by specifying the amount of data the receiver can accept.

---

4. TCP Sequence Numbers and Acknowledgments (ACKs)

- **Sequence Numbers**:
  - Each byte in a TCP stream has a unique sequence number, which helps keep track of data order.
- **ACKs**:
  - The receiver sends an acknowledgment number (ACK) back to the sender to confirm receipt of data.
  - **Cumulative ACKs**: TCP acknowledges all bytes up to a certain point, which means any missing data will require retransmission.

---

5. Round-Trip Time (RTT) and Timeout in TCP

- **RTT**: Time taken for a packet to travel to the receiver and back as an acknowledgment.
- **Timeout**: If the ACK is not received within this time, TCP will resend the packet.
- **Adaptive Timeout**:
  - TCP adjusts the timeout based on estimated RTT and variation to ensure timely retransmission without premature resending.

---

6. TCP Sender and Receiver Behavior

- **Sender**:
  - Creates and sends segments based on sequence numbers.
  - Starts a timer for each unacknowledged segment, and resends it if the timer expires.
- **Receiver**:
  - Acknowledges received data.
  - Uses delayed ACKs to reduce network load.
  - May send duplicate ACKs if data is received out-of-order, signaling a potential missing packet.

---

7. TCP Flow Control

- **Purpose**:
  - Flow control ensures the sender does not overload the receiver.
- **Receive Window (rwnd)**:
  - The receiver specifies how much data it can handle by adjusting the receive window size.

- TCP uses the receive window size to control the sender's data rate.

---

8. TCP Connection Management

- **3-Way Handshake**:
  - **Step 1**: Client sends a **SYN** message to start the connection.
  - **Step 2**: Server replies with **SYN-ACK** to acknowledge the connection.
  - **Step 3**: Client responds with an **ACK**, confirming the connection.
- **Closing the Connection**:
  - Each side can close the connection by sending a **FIN** message, followed by an acknowledgment (ACK) to ensure both ends have finished communication.

---

9. Congestion Control in TCP

- **What is Congestion?**
  - Congestion happens when too many data packets are sent too quickly for the network to handle, causing delays and packet loss.
- **Causes of Congestion**:
  - **High Traffic**: Multiple devices sending large amounts of data.
  - **Router Buffer Overflow**: Routers get overwhelmed, leading to packet drops.
- **Impact**:
  - Increased delays and retransmissions, which waste network resources and reduce efficiency.

10. TCP Congestion Control Mechanisms

- **Additive Increase, Multiplicative Decrease (AIMD)**:
  - TCP increases the data rate gradually and reduces it sharply if congestion is detected.
- **Slow Start**:
  - When a connection begins, TCP slowly increases the sending rate to prevent sudden overloads.
- **Congestion Avoidance**:
  - Once TCP detects network congestion, it reduces the sending rate to stabilize network performance.
- **Fast Retransmit**:
  - If three duplicate ACKs are received, TCP assumes a packet was lost and resends it before the timeout.

---

11. Principles of Congestion Control

- **End-to-End Control**:
  - TCP detects congestion through delays and packet loss rather than explicit signals from the network.
- **Network-Assisted Control**:
  - Routers may directly inform devices of congestion (e.g., with explicit congestion notification or ECN), helping devices adjust their rates to reduce traffic.

---

12. TCP Performance on High-Capacity Networks

- **Long, Fat Pipes**:
  - High-speed, long-distance networks require TCP to handle large amounts of data efficiently without losing packets.
  - **Example**: 10 Gbps networks need very low packet loss rates to maintain high throughput.
- **Challenges**:
  - TCP throughput can be affected by high round-trip times (RTTs) and any packet loss on such networks.