# Network Transport Architectures

CS 2103

Data Communication and Computer Networks

Dr. Omar Haji Kombo

# Outline

* A brief discussion on IP Addressing
* Understand the transport layer's role in networking.
* Explore how TCP and UDP work in practical scenarios.
* Learn about TCP Congestion Control mechanisms.

# IPv4 Address Classes

IPv4 addresses are divided into **classes** based on the first octet:

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Range** | **Default Subnet Mask** | **Purpose** |
| A | 1 - 126 | 255.0.0.0 | Large networks |
| B | 128 - 191 | 255.255.0.0 | Medium networks |
| C | 192 - 223 | 255.255.255.0 | Small networks |
| D | 224 - 239 | N/A | Multicast addresses |
| E | 240 - 255 | N/A | Reserved for research |

# Subnet Mask

* A **subnet mask** determines which part of an IP address is the **network** portion and which is the **host** portion.
* Example:
  + IP Address: 192.168.1.1
  + Subnet Mask: 255.255.255.0
  + **Network Portion:** 192.168.1
  + **Host Portion:** 1
* **Default Subnet Masks:**
  + Class A: 255.0.0.0
  + Class B: 255.255.0.0
  + Class C: 255.255.255.0

# Subnetting

• **Subnetting** is the process of dividing a big network into smaller, more manageable portions known as *subnets.* This practices*:*

* Improve network performance.
* Enhance security.
* Efficiently utilize IP addresses.
* Reduce network congestion.

# Steps to Subnetting

**1. Determine the number of subnets or hosts required.**

* Formula for number of subnets: 2*n*, where *n* is the number of bits borrowed from the host portion.
* Formula for number of hosts: 2*n* −2, where *n* is the number of host bits left.

**Steps to Subnetting…Cont.**

**2. Borrow bits from the host portion.**

• Convert the subnet mask to binary and adjust the bits.

**3. Calculate the new subnet mask.**

• Example: Borrowing 2 bits from /24 results in /26 (255.255.255.192).

**4. Determine the subnet ranges.**

• Subnet increment 256-192 = 64

# Example 1

* Let’s say we have 192.168.1.0/24 (256 total addresses). We want to divide it into **4 subnets**.
* **Determine New Subnet Mask**
* Original Mask: /24 (255.255.255.0)
* Subnet Bits Needed: 2 (to divide into 4 subnets, 2*2* =4).
* New Mask: /26 (255.255.255.192).

## Calculating Subnet Ranges and Hosts

• Subnet range is 26=64 addresses ( and each host gets 64-2 = 62 for hosts).

|  |  |  |
| --- | --- | --- |
| Subnet | Range | Broadcast Address |
| Subnet 1 | 192.168.1.0 - 192.168.1.63 | 192.168.1.63 |
| Subnet 2 | 192.168.1.64 - 192.168.1.127 | 192.168.1.127 |
| Subnet 3 | 192.168.1.128 - 192.168.1.191 | 192.168.1.191 |
| Subnet 4 | 192.168.1.192 - 192.168.1.255 | 192.168.1.255 |

# Overview of Transport Layer

The Transport layer of the OSI model is responsible for reliable end-to-end communication between applications on different hosts.

**Key Functions:**

* Segmentation and reassembly.
* Flow control.
* Error control.

Two primary protocols operate at this layer:

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

# TCP

* **Characteristics:**
* Reliable and connection-oriented.
* Ensures data delivery in order.
* **Key Features:**
* Three-way handshake (connection establishment).
* Error detection using checksum.
* Flow control using sliding window protocol.
* **Practical Use Case:** Web browsing (HTTP/HTTPS).

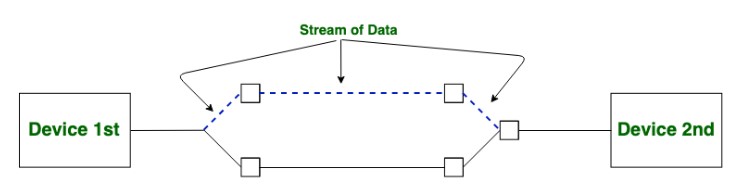
**Diagram:** Three-way handshake (SYN, SYN-ACK, ACK).

# TCP-Connection Establishment

• Before data transmission, TCP establishes a connection between the sender and receiver to ensure reliability:

* **Step 1 (SYN):** The sender sends a synchronization (SYN) packet to the receiver, indicating the intent to start communication.
* **Step 2 (SYN-ACK):** The receiver acknowledges by sending back a synchronization acknowledgment (SYN-ACK).
* **Step 3 (ACK):** The sender sends an acknowledgment (ACK) to confirm the connection is established.
* This handshake ensures that both parties are ready and agree on parameters like sequence numbers

# Connection Oriented Service



The diagram illustrates the connection between sender and receiver

## Socket -Address Family for Connection-oriented Services

|  |  |  |
| --- | --- | --- |
| **Address Family** | **Purpose** | **Example Address** |
| **AF\_INET** | IPv4 communication | 192.168.0.1 |
| **AF\_INET6** | IPv6 communication | 2001:db8::ff00:42:8329 |
| **AF\_UNIX** | Local IPC via Unix domain sockets | /tmp/socket |
| **AF\_PACKET** | Low-level network packet access | Ethernet frames |
| **AF\_RAW** | Custom protocol development | Raw IP packets |
| **AF\_BLUETOOTH** | Bluetooth communication | Bluetooth device addresses |
| **AF\_IRDA** | Infrared communication | Infrared devices |
| **AF\_CAN** | Controller Area Network | Embedded systems communication |

# TCP-Data Transmission

• Once the connection is established, TCP ensures reliable data transfer:

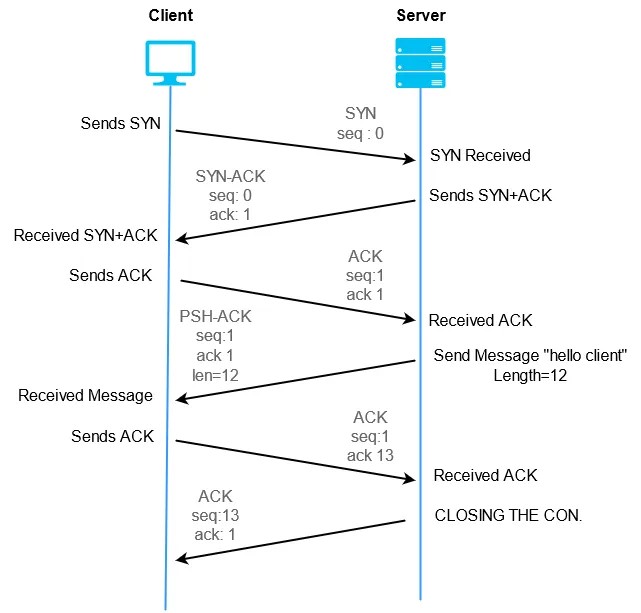
* **Segmentation:** Data is broken into smaller packets, each labeled with a sequence number.
* **Delivery Order:** TCP uses sequence numbers to ensure packets are reassembled in the correct order at the destination.
* **Acknowledgments (ACKs):** The receiver sends ACKs for received packets. If an ACK is not received within a timeout period, the sender retransmits the packet.
* **Flow Control:** TCP uses a sliding window mechanism to control the rate of data flow and avoid overwhelming the receiver.
* **Error Detection:** TCP checks for errors in the data using checksums and requests retransmission of corrupt packets.

# TCP-Connection Termination

• After data transmission, the connection is terminated:

* **Step 1 (FIN):** The sender sends a FIN (Finish) packet, indicating it has no more data to send.
* **Step 2 (ACK):** The receiver acknowledges the FIN packet.
* **Step 3 (FIN):** The receiver sends its own FIN packet.
* **Step 4 (ACK):** The sender acknowledges the receiver’s FIN packet, closing the connection.

# Three-way handshake for TCP



# TCP Practical Demonstration

* We can use Wireshark/Tcpdump/Tshark or any other appropriate tool to capture packets for an HTTP request.
* Then, we can highlight TCP headers (sequence number, acknowledgment).
* We can also show how retransmissions occur if packets are lost.

# TCP Congestion Control Overview

* **Problem:** Network congestion leads to packet loss and reduced performance.
* **TCP's Solution:** Adaptive mechanisms to avoid and recover from congestion.
* **Phases:**
* **Slow Start:** Gradual increase in transmission rate.
* **Congestion Avoidance:** Linear increase to avoid congestion.
* **Fast Retransmit:** Quickly resends lost packets.
* **Fast Recovery:** Avoids returning to slow start after a minor loss.

# TCP/IP Protocol Architecture

* Developed by the US Defense Advanced Research Project Agency (DARPA) for its packet switched network (ARPANET)
* Used by the global Internet
* No official model but a working one.
  + Application layer
  + Host to host or transport layer
  + Internet layer
  + Network access layer
  + Physical layer

# UDP

* Alternative to TCP is User Datagram Protocol
* Not guaranteed delivery
* No preservation of sequence
* No protection against duplication
* Minimum overhead
* Adds port addressing to IP

## UDP (User Datagram Protocol)

* **Characteristics:**
* Lightweight, connectionless, and unreliable.
* No flow or error control.
* **Key Features:**
* Low overhead, faster than TCP.
* Suitable for time-sensitive applications.
* **Practical Use Case:** Video streaming, VoIP. **Diagram:** Simple UDP header structure.

# UDP Practical Demonstration

* We can use a basic UDP clientserver Python script.
* To showcase message sending without connection establishment.
* To highlight how lost packets are not retransmitted.

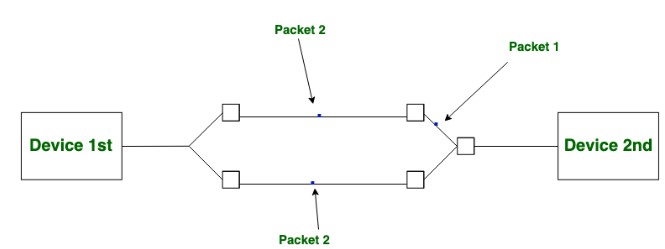
# Comparing TCP and UDP

|  |  |  |
| --- | --- | --- |
| **Feature** | **TCP** | **UDP** |
| **Connection** | Connection-oriented | Connectionless |
| **Reliability** | Reliable  (acknowledgment) | Unreliable |
| **Speed** | Slower (overhead) | Faster |
| **Use Cases** | HTTP, Email | VoIP, Gaming |

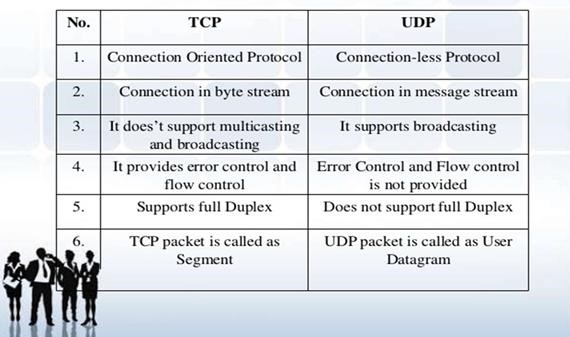
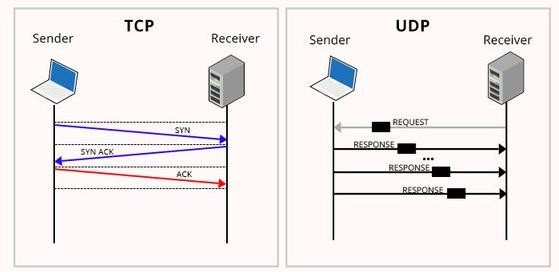
## Challenges with TCP and UDP

* TCP:
* Overhead due to reliability mechanisms.
* Poor performance in high-latency networks.
* UDP:
* Lack of reliability can lead to loss of critical data.
* Not suitable for file transfers or critical applications.

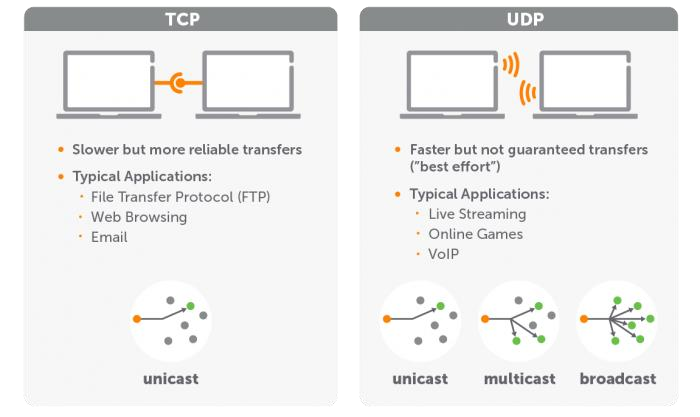
# Connection-less Service



# TCP and UDP



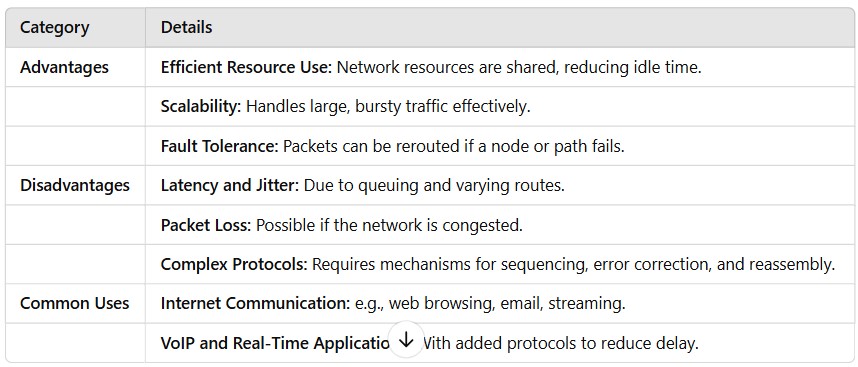
# UDP and TCP



# Packet-Switched Connection (PSC)

* In packet-switched connection, data is divided into small packets that are transmitted independently through the network. This type of connect has the following features:
* **Dynamic Path Selection:** Each packet can take a different path to reach the destination.
* **Store-and-Connectionless (e.g., UDP) or Connectionoriented (e.g., TCP):** Depending on the protocol, the communication may or may not establish a dedicated connection.
* **Best Effort Delivery:** No guarantee of order or delivery unless the protocol ensures it.
* **Forward:** Intermediate nodes store packets temporarily before forwarding them

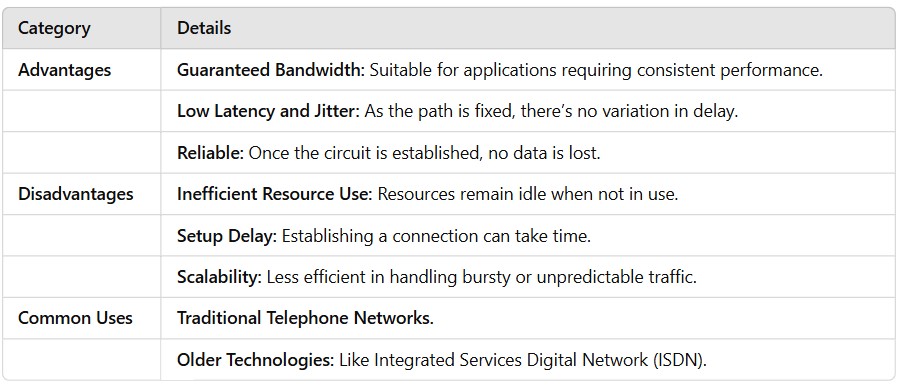
## Merits, Demerits and Applications of PSC



# Circuit-Switched Connection (CSC)

* In circuit-switched connection, a dedicated communication path is established between the source and destination before data transfer begins. This type of connect has the following features:
* **Dedicated Connection:** Resources (e.g., bandwidth) are reserved for the duration of the session.
* **Continuous Data Flow:** Once established, data flows without interruption.
* **Fixed Path:** All data follows the same path during the session.

## Merits, Demerits and Applications of CSC



## Emerging Trends in Transport Layer Architecture

* **QUIC Protocol:** Developed by Google, Used in HTTP/3, combines transport and security (TLS) for reduced latency.
* **Software-Defined Networking (SDN):** Centralized control for transport architectures, simplifies network management and enables dynamic path selection.
* **Transport Layer Security (TLS):** Ensures encrypted communication over networks.
* **5G and Beyond:** Enhanced transport architectures for ultra-low latency and high bandwidth.

# Application

* Reliable data transfers (TCP).
* Real-time gaming and video streaming (UDP, QUIC).
* IoT systems requiring lightweight protocols like MQTT or CoAP.

# Thank you