UNIT-V

Transport Layer

Application layer

Unit – 5
Syllabus

- **UNIT –V:The Transport Layer:** Transport layer protocols: Introduction-services- port number-User data gram protocol-User datagram-UDP services-UDP applications- Transmission control protocol: TCP services- TCP features- Segment- A TCP connection- windows in TCP- flow control-Error control, Congestion control in TCP.
- **Application Layer** –- World Wide Web: HTTP, Electronic Mail-Architecture- web based mail-email security- TELENET-local versus remote Logging-Domain Name System.

Transport Layer

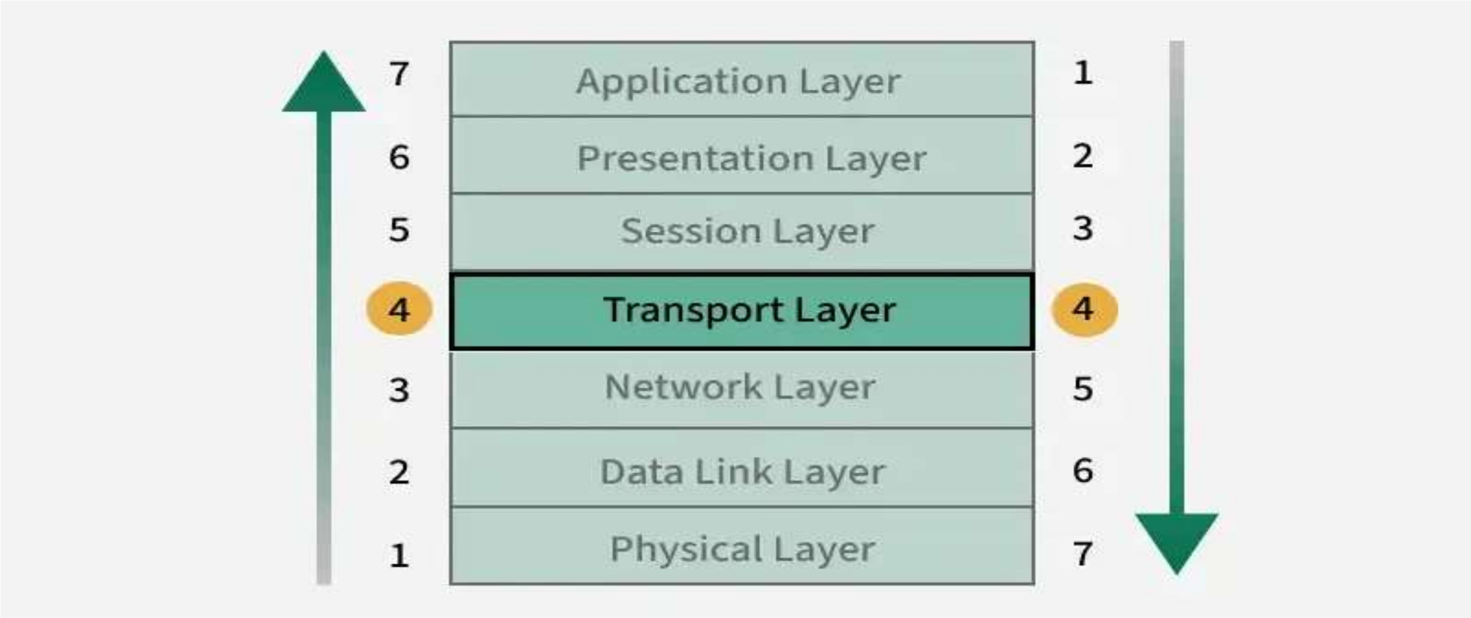
- The transport layer is the fourth layer in the OSI model and the second layer in the TCP/IP model. The transport layer provides with end to end connection between the source and the destination and reliable delivery of the services. Therefore transport layer is known as the end-to-end layer. The transport layer takes the services from its upward layer which is the application layer and provides it to the network layer. Segment is the unit of data encapsulation at the transport layer.

Functions of Transport Layer

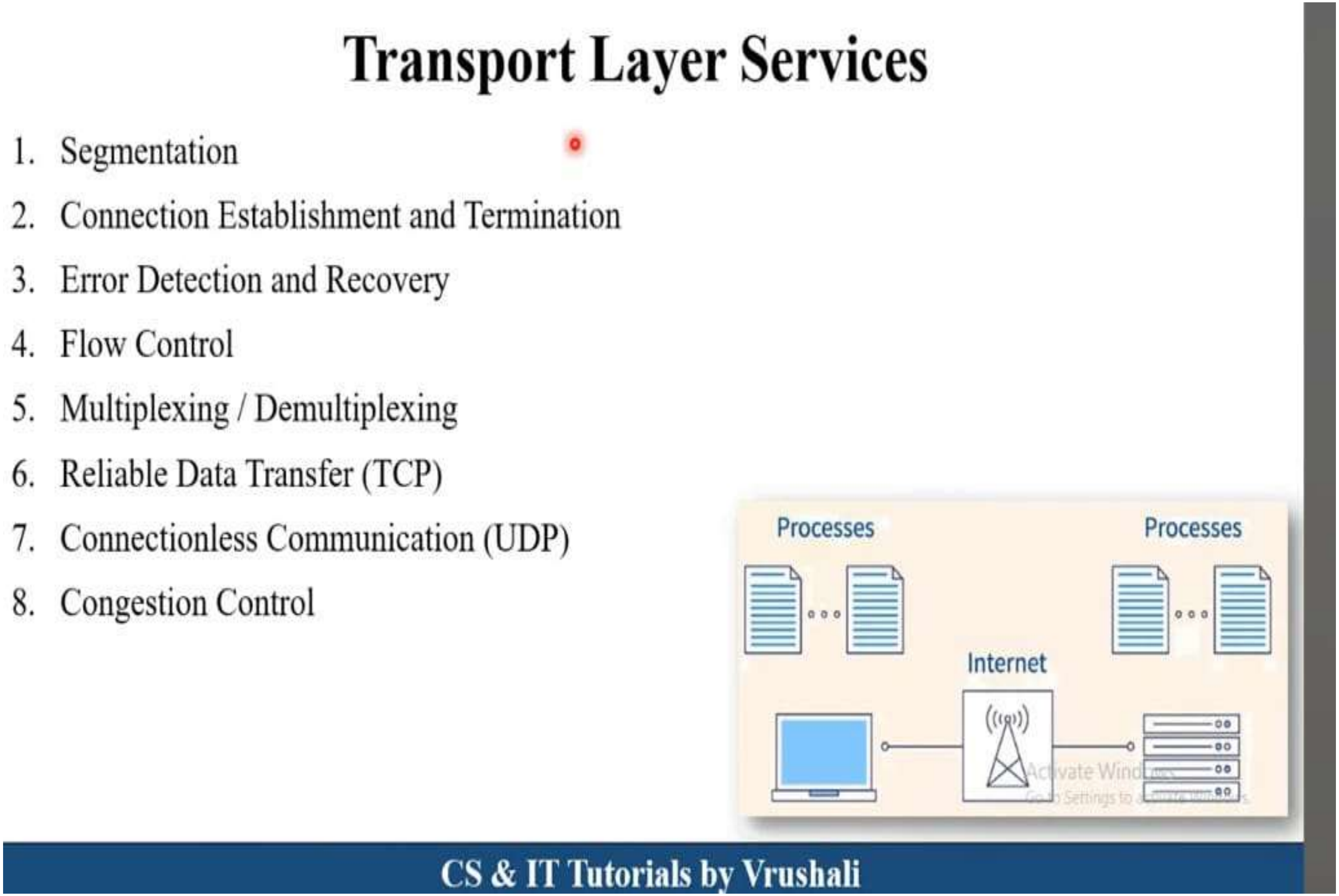
- The process to process delivery
- End-to-end connection between devices
- Multiplexing and Demultiplexing
- Data integrity and error Correction
- Congestion Control
- Flow Control

Transport Layer

Introduction : The Transport Layer ensures end-to-end communication between applications on different hosts. It is between the Network Layer (which delivers packets to the right machine) and the Session Layer (which manages communication sessions). Its main job is to deliver data reliably, efficiently, and in the correct order.



Transport Services



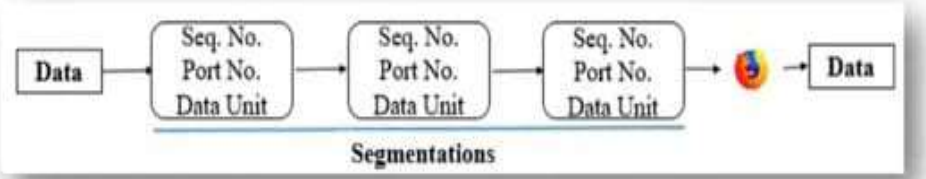
Transport Layer Services

1. Segmentation
2. Connection Establishment and Termination
3. Error Detection and Recovery
4. Flow Control
5. Multiplexing / Demultiplexing
6. Reliable Data Transfer (TCP)
7. Connectionless Communication (UDP)
8. Congestion Control

CS & IT Tutorials by Vrushali

1. Segmentation

- Data (File or messages) sent from application layer is too large to be transmitted in single packet.
- It divides this large data stream into smaller, manageable pieces called **segments.**
- Segments include Sequence Number, Port Number & Data.
- At the receiving end, these segments are reassembled into the original message.



Data

Seq. No.
Port No.
Data Unit

Seq. No.
Port No.
Data Unit

Seq. No.
Port No.
Data Unit

Data

Segmentations

➢Example:

✓Imagine downloading a *100 MB* video file.

✓Transport Layer breaks it into small segments, each around *1,500 bytes* (the maximum size a network can handle).

✓ The receiver collects these segments and reassembles them into the original *100 MB file*.

2. Connection Establishment and Termination

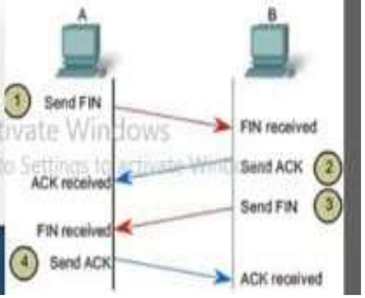
- Before data can be transferred, connection needs to be established between sender and receiver.
- Once the data exchange is complete, the connection is terminated.
- TCP used three-way handshake protocols like SYN, SYN-ACK, ACK.

➢**Example:**

✓When you visit a website, your browser (the client) establishes a TCP connection with the web server using the three-way handshake.

✓Then client assess data on web page.

✓After the data (like the web page) is exchanged, the connection is terminated.



3. Error Detection and Recovery

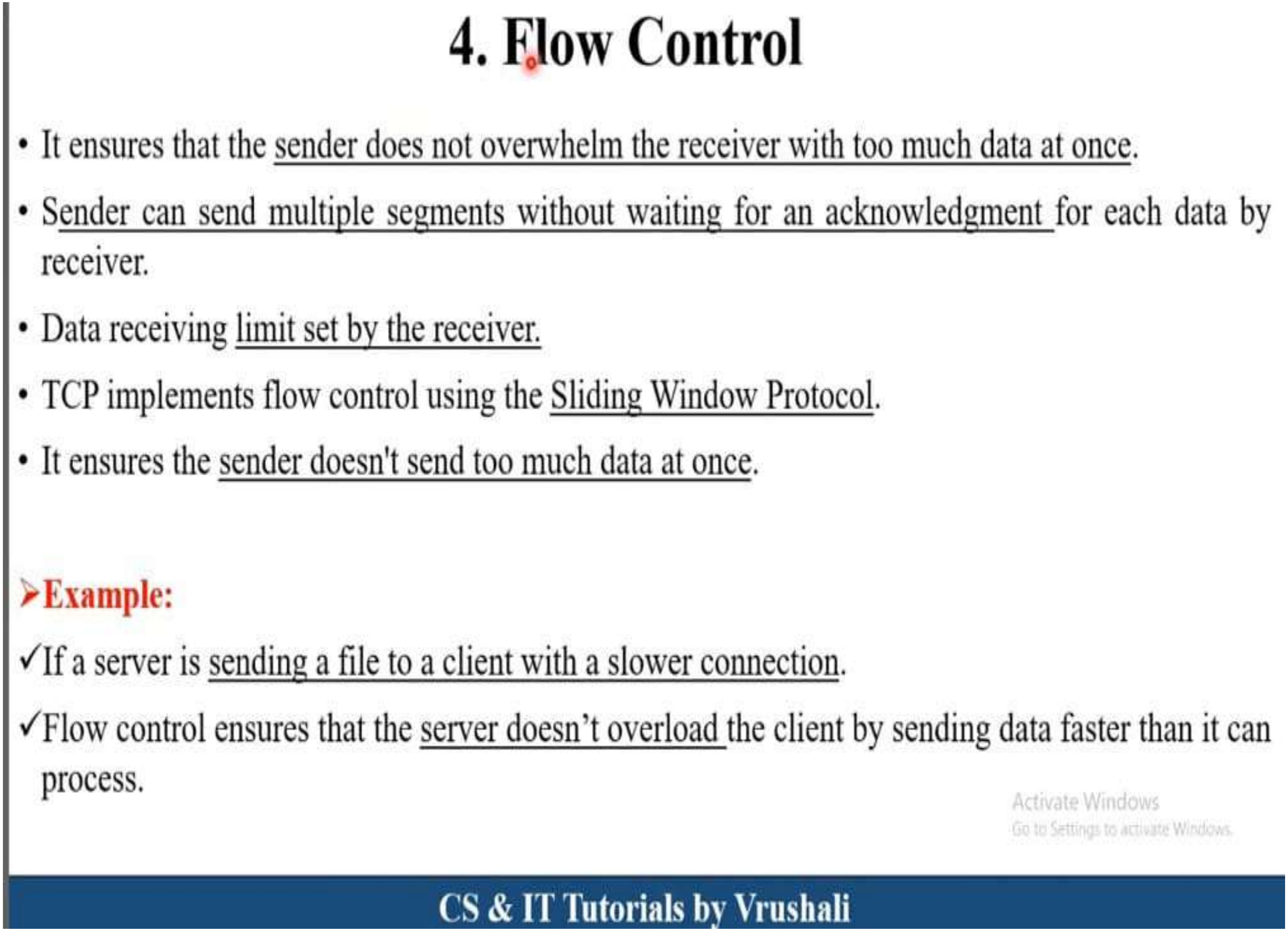
- Ensures that data sent across the network is error-free.
- If any errors are detected, TCP protocol uses a checksum to detect errors.
- Each segment has a checksum value that is calculated from the data it contains.
- When the receiver gets the segment, it recalculates the checksum.
- If values don't match, it means the data is corrupted, and the receiver requests a retransmission.

➢**Example:**

✓While sending an email, if part of the email gets corrupted during transmission TCP detects this using the checksum.

✓ The sender retransmits the corrupted portion.

✓It ensure that email is delivered accurately.



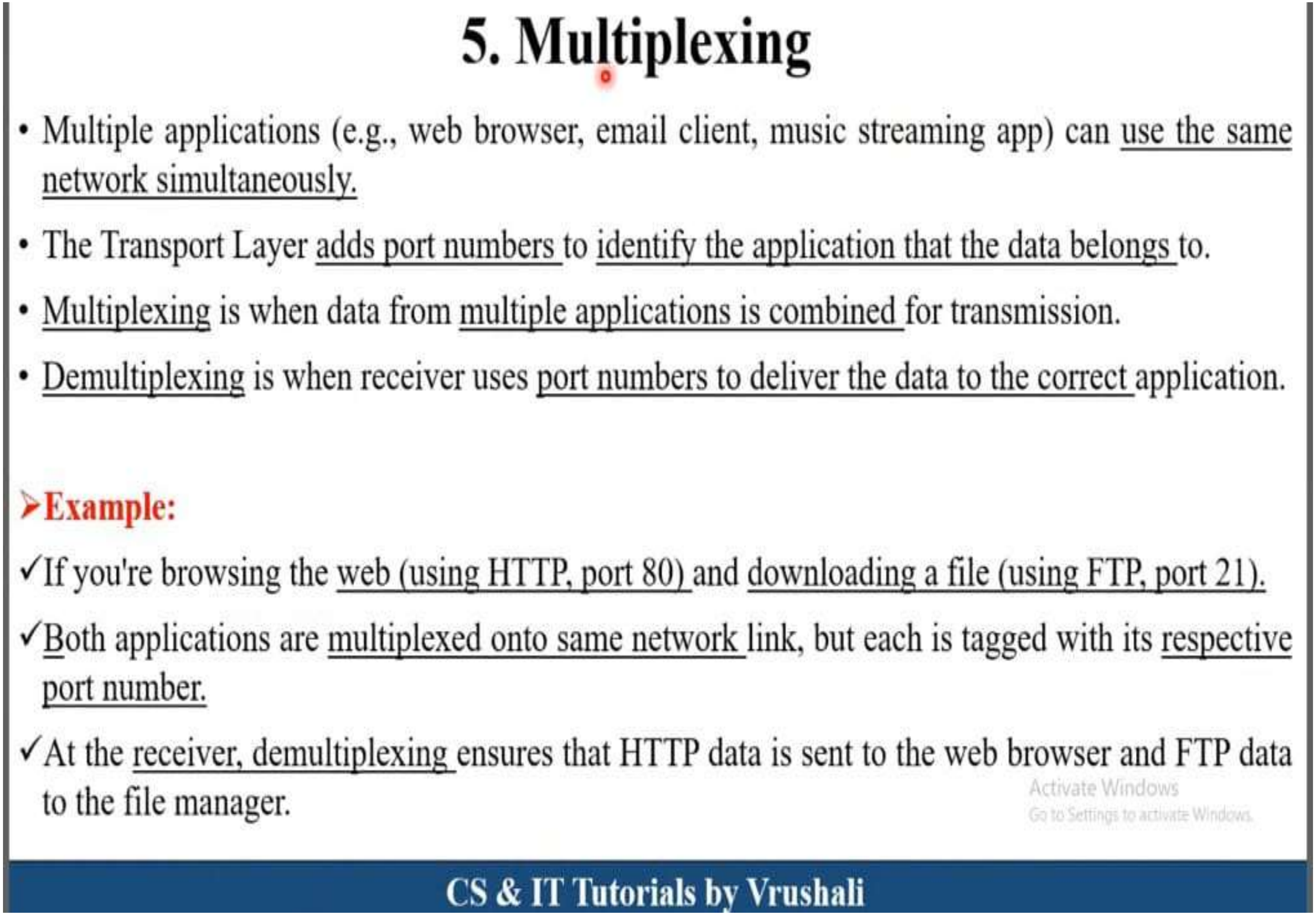
4. Flow Control

- It ensures that the sender does not overwhelm the receiver with too much data at once.
- Sender can send multiple segments without waiting for an acknowledgment for each data by receiver.
- Data receiving limit set by the receiver.
- TCP implements flow control using the Sliding Window Protocol.
- It ensures the sender doesn't send too much data at once.

➢**Example:**

✓If a server is sending a file to a client with a slower connection.

✓Flow control ensures that the server doesn't overload the client by sending data faster than it can process.



5. Multiplexing

- Multiple applications (e.g., web browser, email client, music streaming app) can use the same network simultaneously.
- The Transport Layer adds port numbers to identify the application that the data belongs to.
- Multiplexing is when data from multiple applications is combined for transmission.
- Demultiplexing is when receiver uses port numbers to deliver the data to the correct application.

➢**Example:**

✓If you're browsing the web (using HTTP, port 80) and downloading a file (using FTP, port 21).

✓Both applications are multiplexed onto same network link, but each is tagged with its respective port number.

✓At the receiver, demultiplexing ensures that HTTP data is sent to the web browser and FTP data to the file manager.

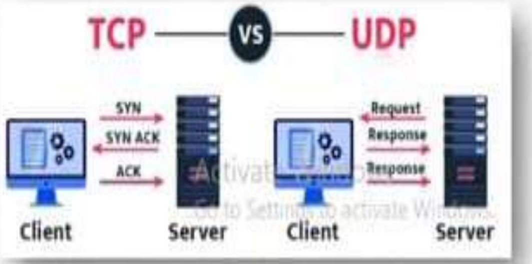
6. Reliable Data Transfer (TCP)

- TCP ensures that all data reaches its destination in correct order and without any loss or duplication.
- If a segment is lost or doesn't arrive at the destination within a certain time, TCP triggers a retransmission timer and resends the segment.
- It also arranges out-of-order segments based on sequence numbers.

➢Example:

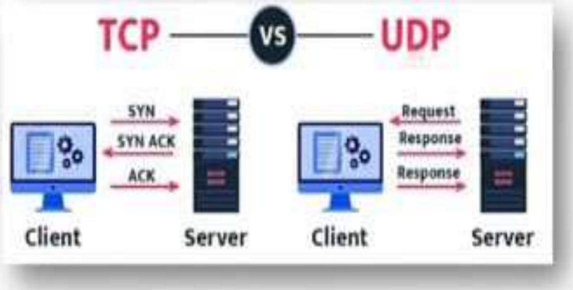
✓While downloading a video, if one segment of the video file is delayed or lost.

✓TCP will resend it until the entire video is successfully received without any gaps.



7. Connectionless Communication (UDP)

- UDP as User Datagram Protocol provides a faster, connectionless service.
- Here doesn't guarantee of accurate delivery & error recovery.
- It's used when speed is more important than reliability.
- It doesn't establish a connection before sending data nor does it wait for acknowledgments.
- It simply sends data and assumes it gets there.



➢**Example:**

✓In live video streaming like YouTube Live UDP is used.

✓Here, occasional lost packets are not needed and retransmitting them would cause delays.

✓The primary concern is maintaining a smooth stream with minimal latency.

8. Congestion Control

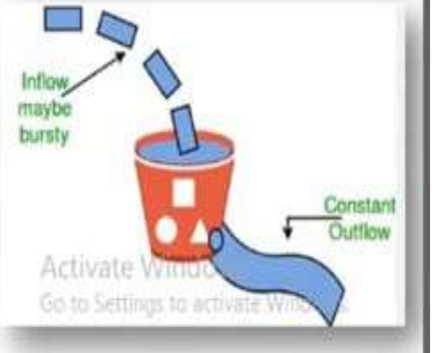
- TCP uses algorithms like TCP Reno or TCP Cubic to detect congestion like lost packet or delivery delayed.
- When congestion is detected, TCP slows down the transmission rate.
- If the network clears up, TCP gradually increases the transmission rate.

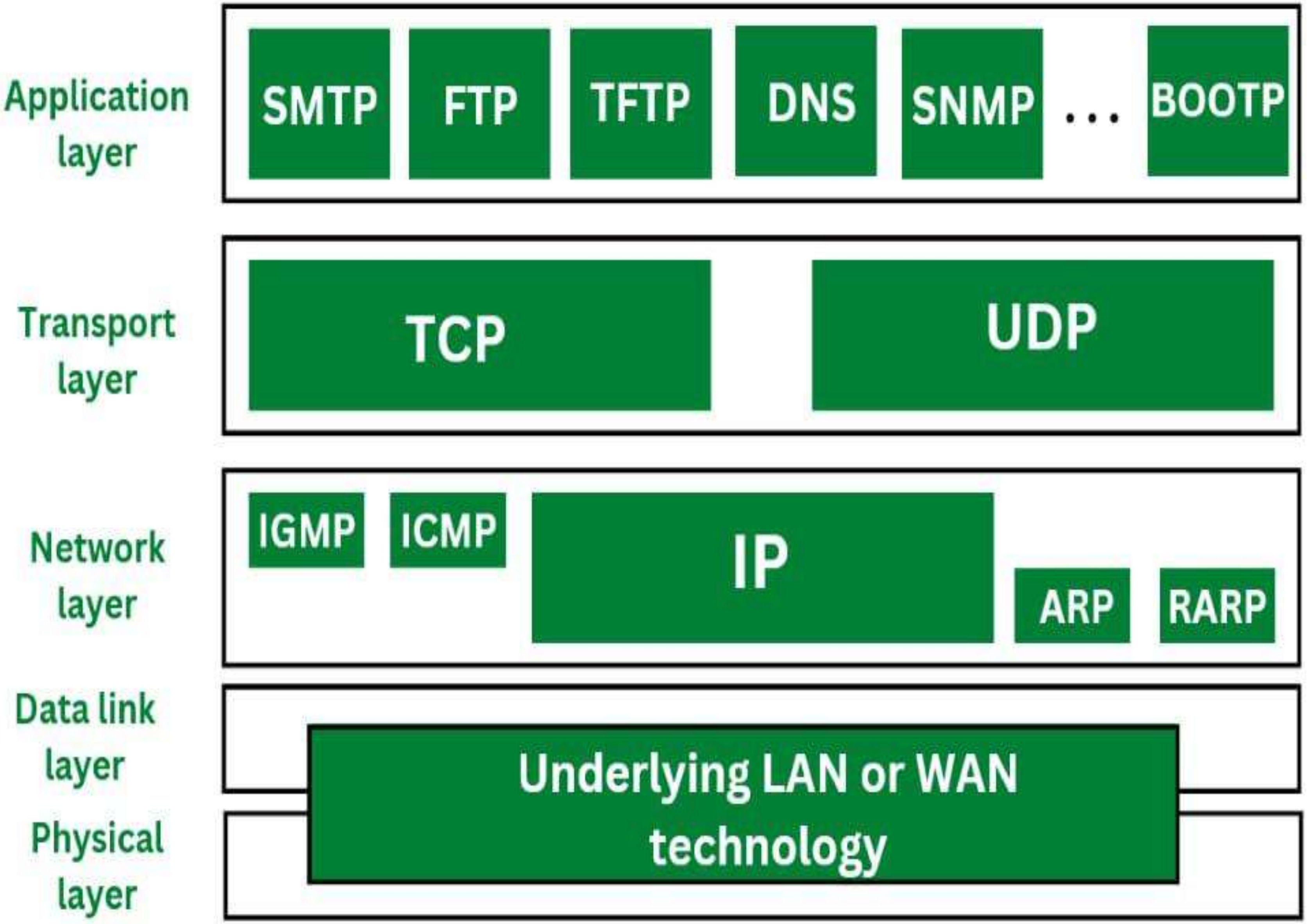
➢ **Example:**

✓When multiple users download large files simultaneously on a shared network.

✓TCP will detect congestion if packet loss or delays occur.

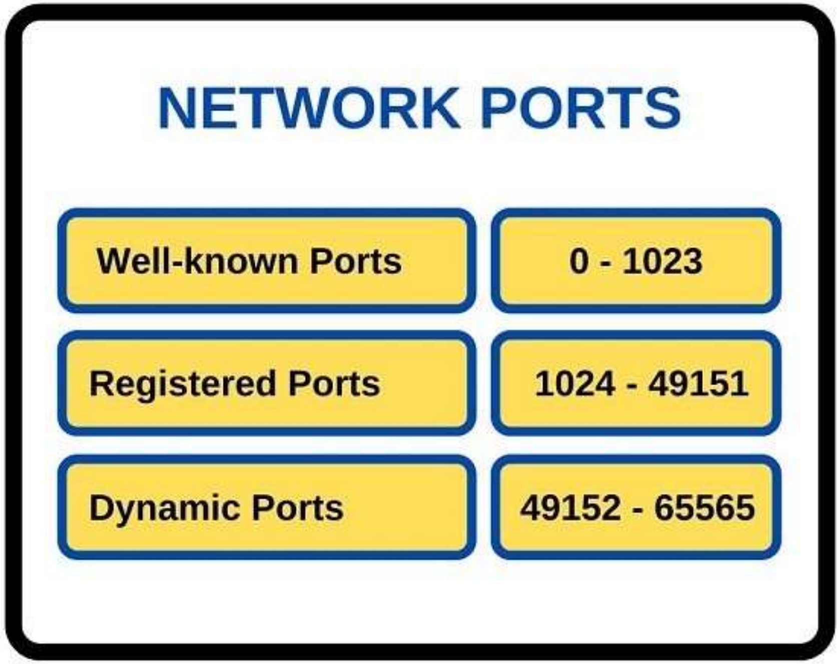
✓TCP adjust each user's download rate to ease the congestion.




Application layer
SMTP
FTP
TFTP
DNS
SNMP
...
BOOTP
Transport layer
TCP
UDP
Network layer
IGMP
ICMP
IP
ARP
RARP
Data link layer
Physical layer
Underlying LAN or WAN technology

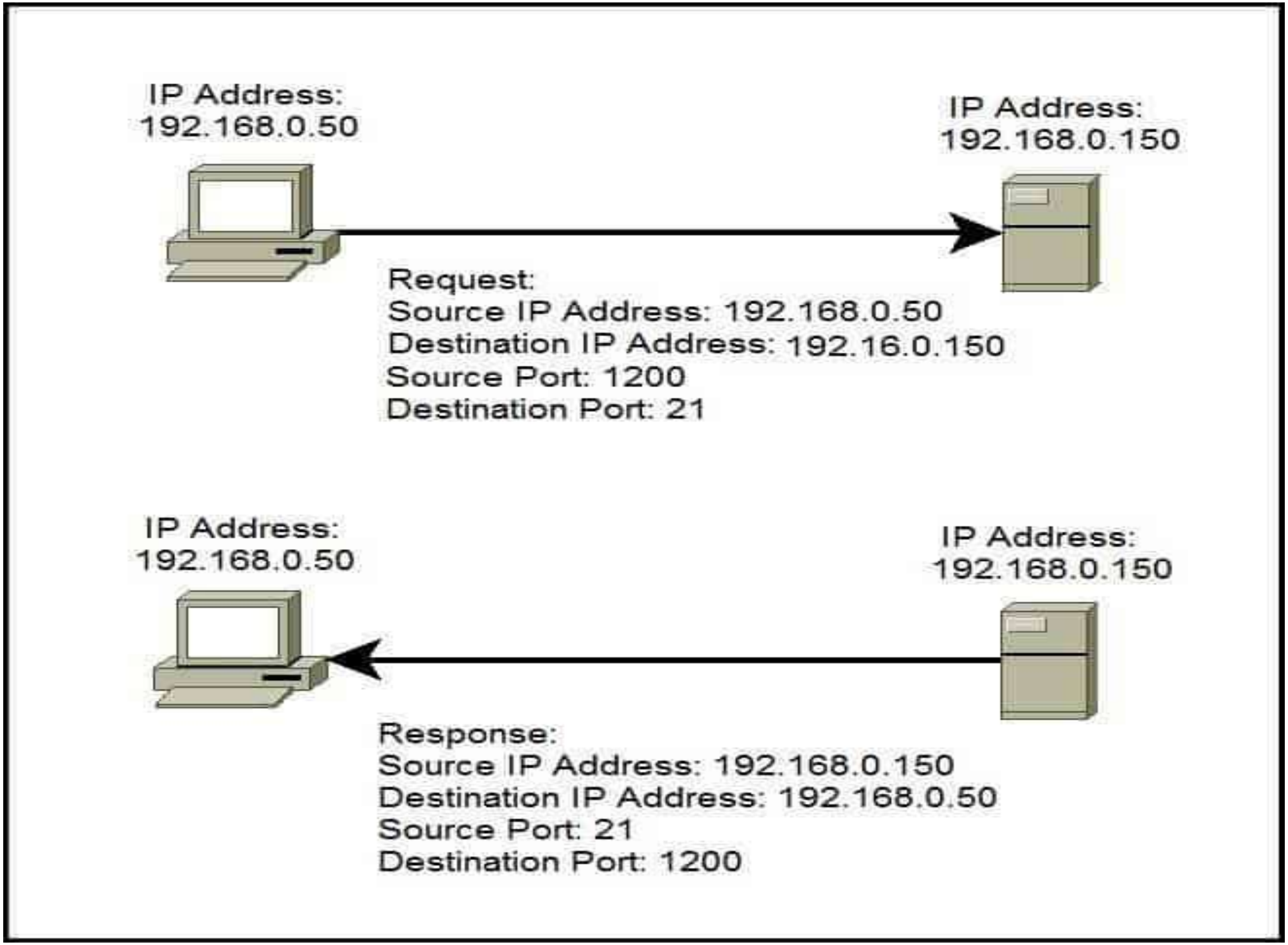
Port Number

- A port number is a 16-bit numerical identifier (0-65535) in computer networks that acts as a specific communication endpoint for a particular service or application on a device, allowing data to be routed to the correct software on a network. Combined with an IP address, a port number specifies both the destination device and the specific process within that device, such as a web server on port 80 or a secure website on port 443.
- For example, port 80 is used for HTTP web traffic, while port 443 is used for HTTPS.



NETWORK PORTS

Well-known Ports	0 - 1023
Registered Ports	1024 - 49151
Dynamic Ports	49152 - 65565



IP Address:
192.168.0.50

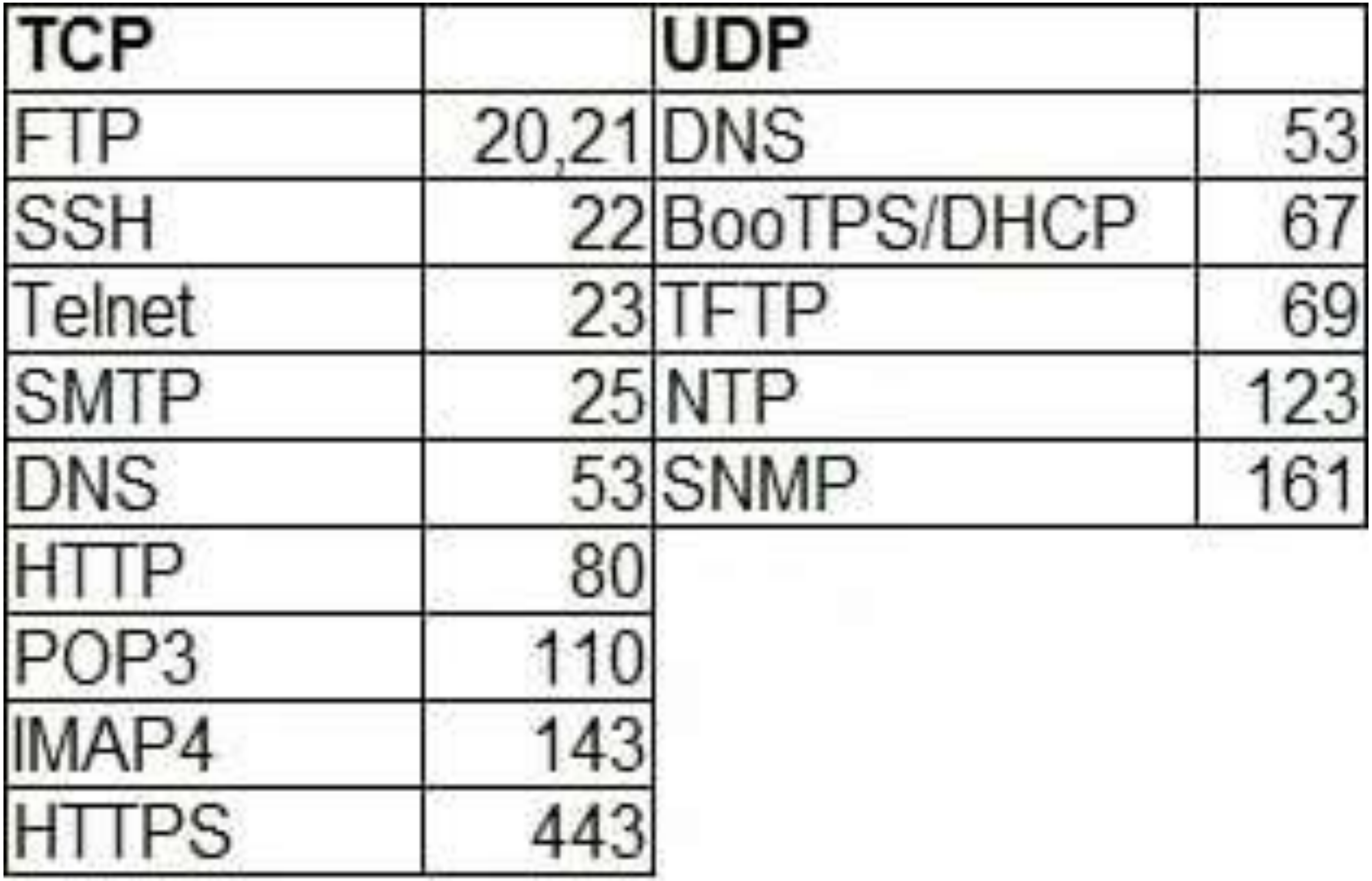
IP Address:
192.168.0.150

Request:
Source IP Address: 192.168.0.50
Destination IP Address: 192.16.0.150
Source Port: 1200
Destination Port: 21

IP Address:
192.168.0.50

IP Address:
192.168.0.150

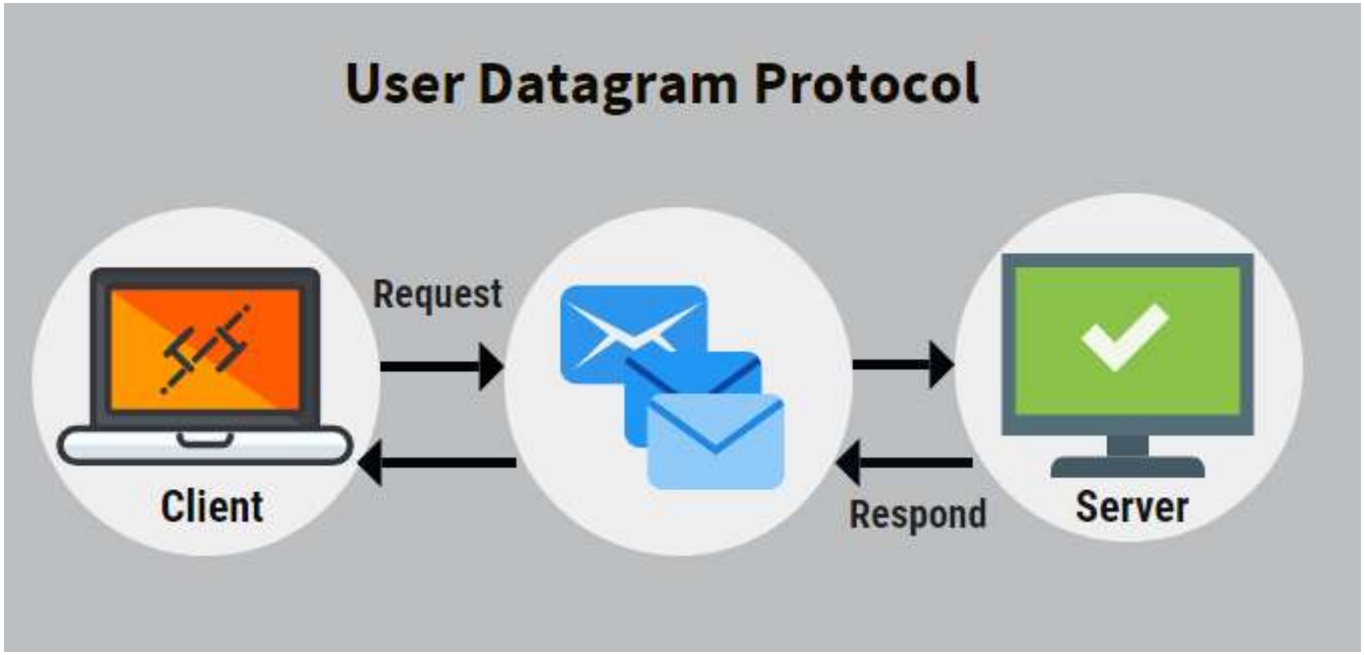
Response:
Source IP Address: 192.168.0.150
Destination IP Address: 192.168.0.50
Source Port: 21
Destination Port: 1200



TCP		UDP	
FTP	20,21	DNS	53
SSH	22	BooTPS/DHCP	67
Telnet	23	TFTP	69
SMTP	25	NTP	123
DNS	53	SNMP	161
HTTP	80		
POP3	110		
IMAP4	143		
HTTPS	443		

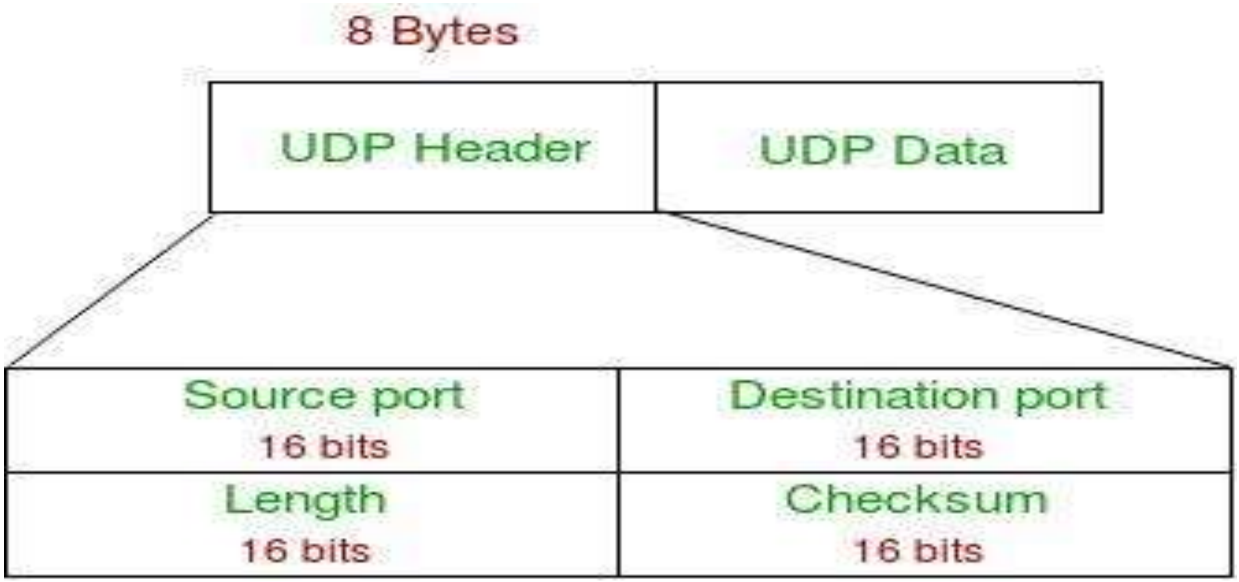
User Datagram Protocol (UDP)

- User Datagram Protocol (UDP) is a Transport Layer protocol of the Internet Protocol (IP) that provides fast, connectionless, and lightweight communication between processes. It does not guarantee delivery, order, or error checking, making it suitable for real-time and time-sensitive applications such as video streaming, DNS, and VoIP.



UDP Header

- UDP header is an **8-byte** fixed and simple header, while for TCP it may vary from 20 bytes to 60 bytes. The first 8 Bytes contain all necessary header information and the remaining part consists of data. UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535, port number 0 is reserved. Port numbers help to distinguish different user requests or processes.



- **Source Port:** Source Port is a 2 Byte long field used to identify the port number of the source.
- **Destination Port:** It is a 2 Byte long field, used to identify the port of the destined packet.
- **Length:** Length is the length of UDP including the header and the data. It is a 16-bits field.
- **Checksum:** A 2-byte field that stores the 16-bit one's complement of the sum of the UDP header, pseudo-header (from IP), and data (padded if needed).

Key services and characteristics

Connectionless and fast:

- UDP does not require a handshake to establish a connection, which reduces overhead and latency. Data is sent in independent datagrams.

Unreliable delivery:

- It does not guarantee that packets will be delivered, will arrive in the correct order, or will not be duplicated.

Minimal overhead:

- UDP is lightweight because it has a simple header and lacks features like acknowledgments, flow control, and error correction, which are handled by the application if needed.

Transaction-oriented:

- It is well-suited for simple, request-response protocols such as the Domain Name System (DNS) and Network Time Protocol (NTP).

Supports multicast:

- UDP can be used for broadcast and multicast communication, making it useful for applications that need to send information to multiple recipients simultaneously.

Real-time applications

Online Gaming:

- UDP's low overhead and speed are crucial for the real-time, high-frequency updates required in online games.

VoIP and Video Conferencing:

- Voice and video calls benefit from UDP's low latency, as occasional dropped packets are less disruptive than the delays caused by guaranteed delivery protocols like TCP.

Media Streaming:

- Services like Netflix and YouTube use UDP for continuous, real-time streaming where speed is more critical than the impact of a few lost packets.

Query and management applications

DNS (Domain Name System):

- UDP is ideal for DNS lookups, which are typically small, fast query-response transactions where a client can simply resend the request if a response is not received.

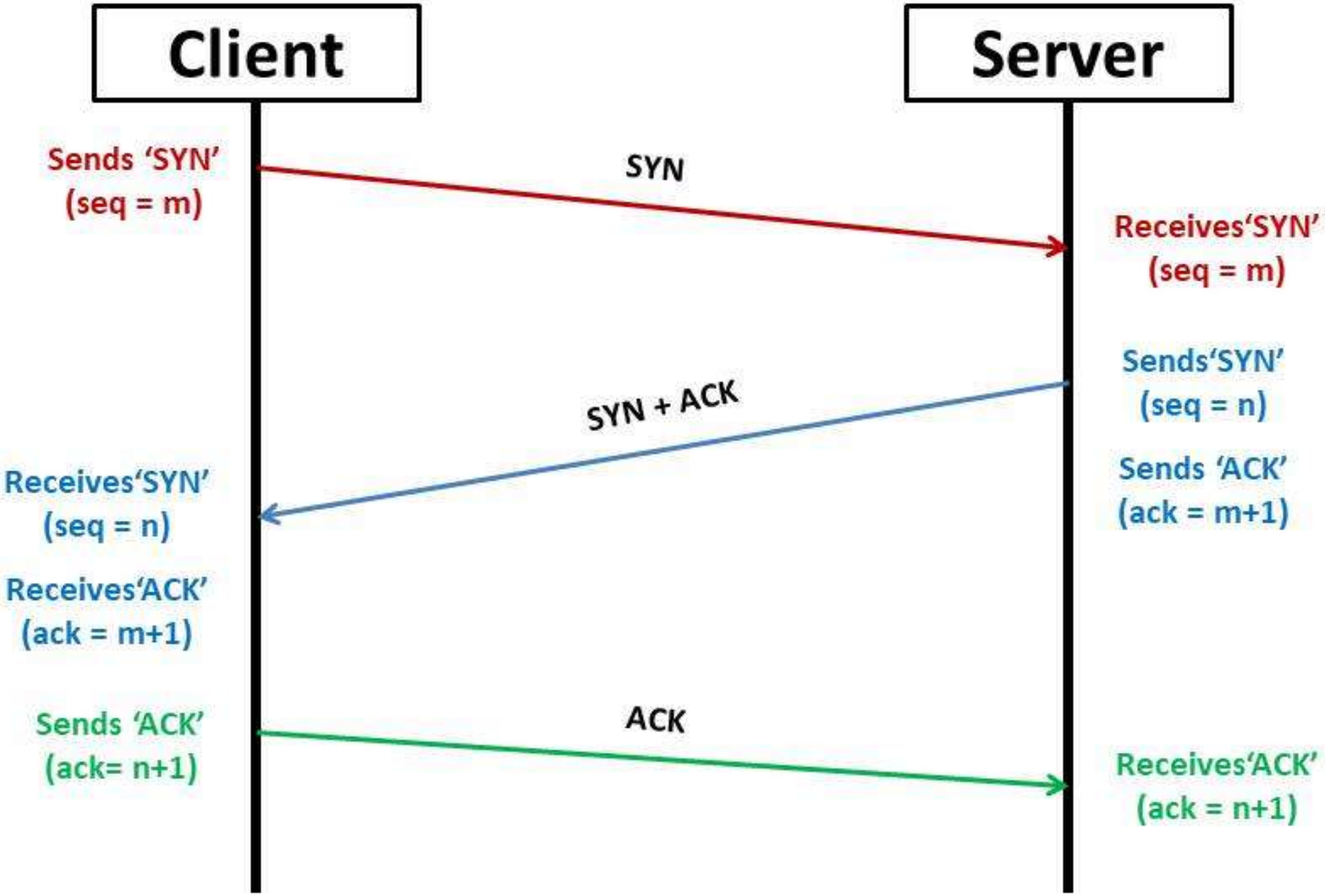
DHCP (Dynamic Host Configuration Protocol):

- This protocol uses UDP for assigning IP addresses to devices on a network.

SNMP (Simple Network Management Protocol):

- Network monitoring tools use UDP to send small management data packets where speed is more important than a guarantee of delivery.

- Transmission Control Protocol (TCP) is a connection-oriented protocol for communications that helps in the exchange of messages between different devices over a network. It is one of the main protocols of the TCP/IP suite. In OSI model, it operates at the transport layer(Layer 4). It lies between the Application and Network Layers which are used in providing reliable delivery services. The Internet Protocol (IP), which establishes the technique for sending data packets between computers, works with TCP.
- TCP establishes a reliable connection between sender and receiver using the **three-way handshake (SYN, SYN-ACK, ACK)** and it uses a **four-step handshake (FIN, ACK, FIN, ACK)** to close connections properly.



Client

Server

Sends 'SYN'
(seq = m)

SYN

Receives'SYN'
(seq = m)

Sends'SYN'
(seq = n)

Sends 'ACK'
(ack = m+1)

SYN + ACK

Receives'SYN'
(seq = n)

Receives'ACK'
(ack = m+1)

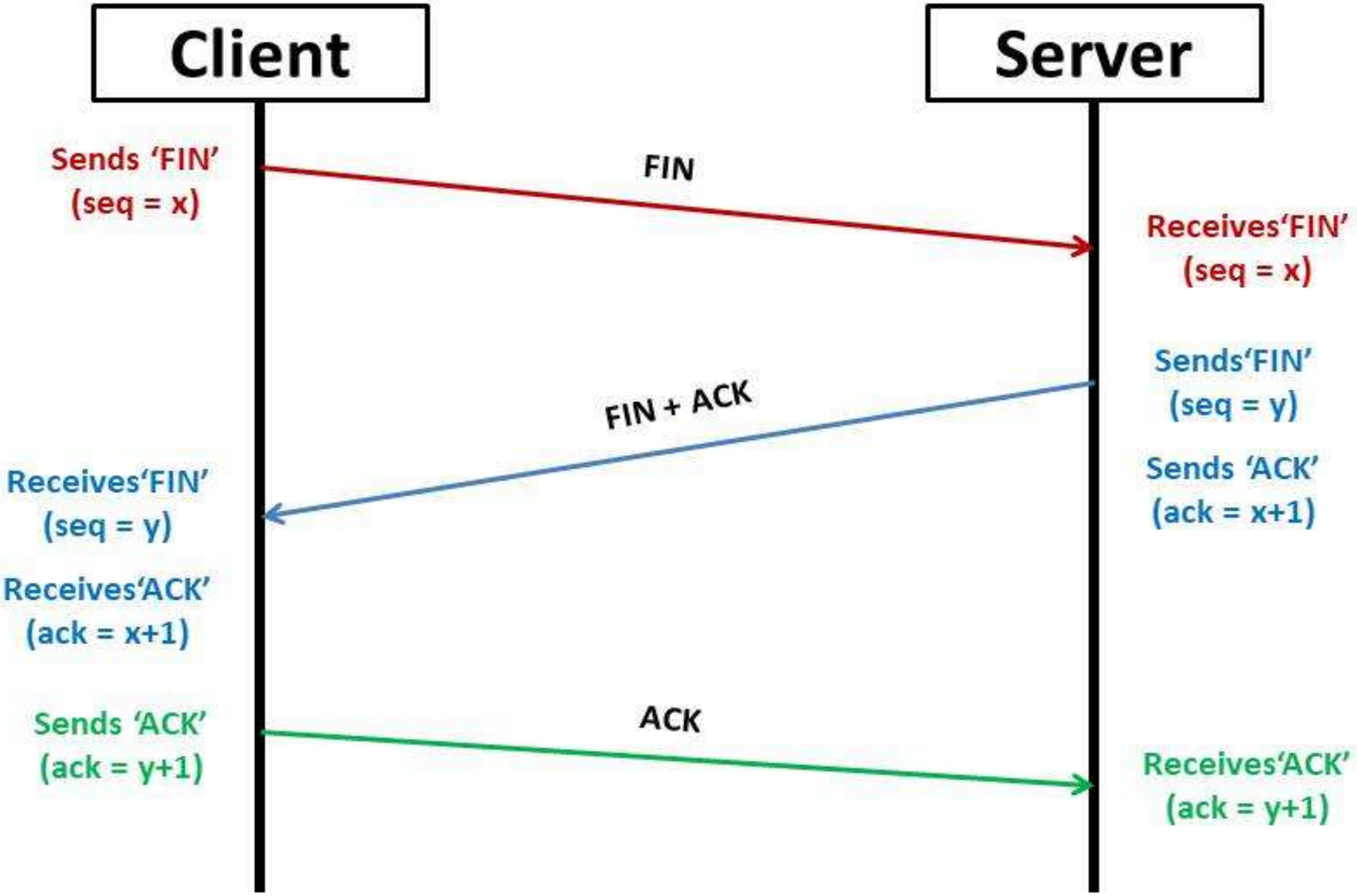
Sends 'ACK'
(ack= n+1)

ACK

Receives'ACK'
(ack = n+1)

3-Way Handshaking(for establishing connection)

1. **The client sends the SYN to the server:** When the client wants to connect to the server. It sets the 'SYN' flag as 1 and sends the message to the server.
2. **The server replies with the SYN and the ACK to the client:** After receiving the client's synchronization request, the server sends an acknowledge to the client by setting the ACK flag to '1'. The acknowledgement number of the ACK is one more than the received sequence number.
3. **The client sends the ACK to the server:** After receiving the SYN from the server, the client sets the ACK flag to '1' and sends it with an acknowledgement number 1 greater than the server's SYN sequence number to the client.



1. **The client sends the FIN to the server:** When the client wants to terminate the connection. It sets the FIN flag as '1' and sends the message to the server with a random sequence number.
2. **The server replies with the FIN and the ACK to the client:** After receiving the client's termination request, the server sends an acknowledge to the client by setting the ACK flag to '1'. The acknowledgement number of the ACK is one more than the received sequence number.
3. **The client sends the ACK to the server:** After receiving the FIN from the server, the client sets the ACK flag to '1' and sends it with an acknowledgement number 1 greater than the server's FIN sequence number to the client.

- It ensures **error-free, in-order delivery** of data packets.
- It uses **acknowledgments (ACKs)** to confirm receipt.
- It prevents data overflow by adjusting the data transmission rate according to the receiver’s buffer size.
- It prevents network congestion using algorithms like **Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery.**
- TCP header uses **checksum** to detect corrupted data and requests retransmission if needed.
- It is used in applications requiring **reliable** and **ordered** data transfer, such as web browsing, email, and remote login.

Services and Segment structure in TCP

- The Transmission Control Protocol is the most common transport layer protocol. It works together with IP and provides a reliable transport service between processes using the network layer service provided by the IP protocol. The various services provided by the TCP to the application layer are as follows:
- **1. Process-to-Process Communication:** TCP enables process-to-process communication using 16-bit port numbers to identify the sending and receiving processes.
  2. Stream oriented: TCP sends data as a byte stream, grouping it into segments with headers, which are then encapsulated into IP packets.

3. **Full-duplex service:** This means that the communication can take place in both directions at the same time.

4. **Connection-oriented service:** Unlike UDP, TCP provides a connection-oriented service. It defines 3 different phases:

- Connection establishment
- Data transfer
- Connection termination

5. **Reliability:** TCP ensures reliability through checksums, retransmissions, acknowledgements, sequencing, and congestion control.

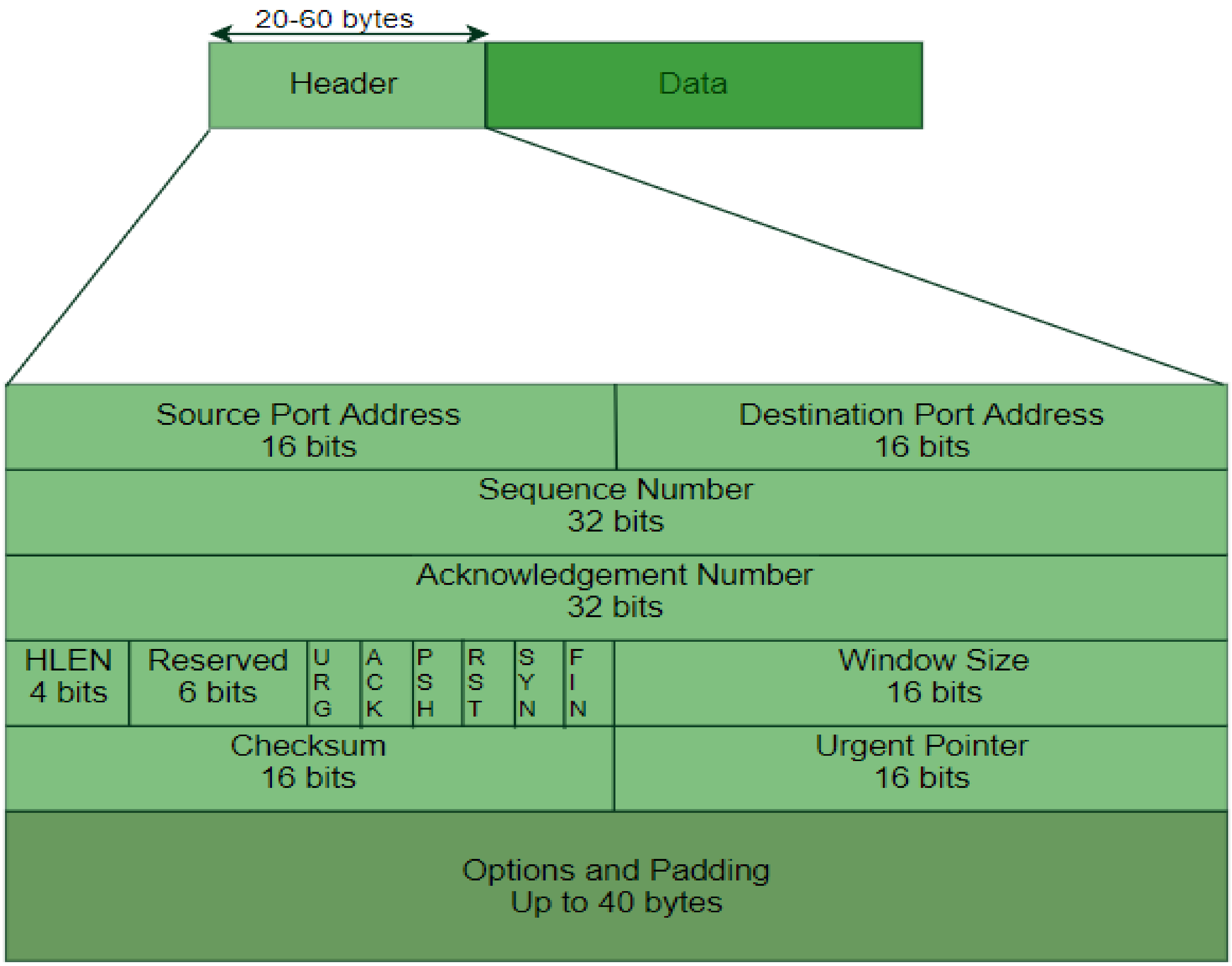
6. **Multiplexing:** TCP does multiplexing and de-multiplexing at the sender and receiver ends respectively as a number of logical connections can be established between port numbers over a physical connection.

TCP Segment structure

- A TCP segment consists of data bytes to be sent and a header that is added to the data by TCP
- The header of a TCP segment can range from 20-60 bytes. 40 bytes are for options. If there are no options, a header is 20 bytes else it can be of upmost 60 bytes.

Header fields

- **1. Source Port Address:** A 16-bit field that holds the port address of the application that is sending the data segment.
- **2. Destination Port Address:** A 16-bit field that holds the port address of the application in the host that is receiving the data segment.



Source Port Address 16 bits | Destination Port Address 16 bits

Sequence Number 32 bits

Acknowledgement Number 32 bits

HLEN 4 bits | Reserved 6 bits | URG | ACK | PSH | RST | SYN | FIN | Window Size 16 bits

Checksum 16 bits | Urgent Pointer 16 bits

Options and Padding Up to 40 bytes

- **Sequence Number (32 bits):** The byte number of the first byte of data in this segment. It is used for reassembling data in the correct order.
- **Acknowledgment Number (32 bits):** If the ACK flag is set, this field indicates the next sequence number the receiver expects to receive.
- **Data Offset (4 bits):** Specifies the length of the TCP header in 32-bit words. It ranges from 5 to 15 words, corresponding to a header size of 20 to 60 bytes.
- **Reserved (6 bits):** Reserved for future use and is set to zero.

- **Control Flags (9 bits):** These bits signal different actions. Key flags include:
- **URG:** Indicates that the urgent pointer field is valid.
- **ACK:** Indicates that the acknowledgment field is valid.
- **PSH:** Requests the sender to push the data immediately to the receiving application.
- **RST:** Resets the connection.
- **SYN:** Synchronizes sequence numbers to initiate a connection.
- **FIN:** Indicates the sender has no more data to send, used to terminate a connection.
- **Window Size (16 bits):** The number of bytes the receiver is willing to accept, used for flow control.

- **Checksum (16 bits):** Used for error detection on the header and data.
- **Urgent Pointer (16 bits):** If the URG flag is set, this points to the last byte of urgent data.
- **Options (Optional):** A variable-length field for options not covered in the fixed header, such as the maximum segment size (MSS).
- **Padding:** Used to ensure the header ends on a 32-bit boundary.
- **Data:** The actual payload or data from the application layer.

- Windows in TCP : In TCP, "windows" are mechanisms that control the amount of data sent before receiving an acknowledgment, used for flow control to prevent overwhelming the receiver and for congestion control to avoid network congestion. The key concepts are the receive window (RWIN) (receiver's buffer size) and the congestion window (CWIN) (sender's dynamically adjusted window). The sender uses the minimum of these two windows to determine how much data to send, creating a "sliding window" to manage data flow efficiently and reliably.

Key concepts

Receive Window (RWIN):

- This is the size of the receiver's buffer, which determines how much data the receiver is willing to accept. The receiver advertises this window size to the sender during the connection.

Congestion Window (CWIN):

- This is a variable on the sender's side that dynamically adjusts based on network conditions. If data is lost or delivered out-of-order, the CWIN is typically reduced to avoid overwhelming the network.

Sliding Window:

- The actual amount of data that can be sent is the minimum of the RWIN and the CWIN. This creates a "sliding window" where the sender can send a block of data without waiting for individual acknowledgments. As acknowledgments come in, the window "slides" forward, allowing more data to be sent.

Data Flow Control:

- By using the window size, TCP ensures the sender doesn't send data faster than the receiver can process it, preventing data loss due to a full buffer.

Reliability:

- The window mechanism is a key part of how TCP achieves reliability. The sender holds a copy of the data in its buffer until acknowledgments are received, allowing for retransmission of lost packets.

TCP Flow Control :

- TCP flow control prevents a sender from overwhelming a receiver's buffer by using a sliding window mechanism. The receiver advertises its available buffer space (Receive Window, or RWND) in the TCP header, and the sender adjusts its data transmission rate not to exceed this advertised amount. This process ensures that data is sent at a rate the receiver can handle, preventing packet loss and maintaining communication.

How TCP Flow Control Works

- **Buffer Allocation:** Both the sender and receiver have buffers to store data.
- **Receiver Advertisement:** During the connection setup (three-way handshake) and periodically thereafter, the receiver informs the sender about the size of its empty buffer space.

Sliding Window: The sender maintains a "window" representing the amount of unacknowledged data it can send at a time. This window is limited by the receiver's advertised window size.

Data Transmission: The sender transmits data within this window.

Window Updates: As the receiver processes data, its buffer space becomes available, and it sends an updated (larger) window size to the sender, allowing the sender to transmit more data.

Zero Window Condition: If the receiver's buffer becomes full, it advertises a window size of zero.

Persistence Timer: To prevent deadlock when a zero window is advertised, the sender uses a persistence timer. The sender periodically sends a small probe to the receiver to check if the buffer has available space.

- TCP uses a combination of checksum, acknowledgments, and retransmission to control errors and ensure reliable data delivery. It detects corrupted data with a checksum, confirms receipt of segments using acknowledgments, and retransmits lost or corrupted segments if an acknowledgment isn't received within a set timeout period or after duplicate ACKs are received. These mechanisms handle corrupted, lost, out-of-order, and duplicated segments.

Error detection:

- **Checksum**: Every TCP segment includes a 16-bit checksum calculated over the header and data. The receiving TCP calculates the checksum and compares it to the checksum in the header. If they don't match, the segment is considered corrupted and is discarded, prompting retransmission.

Error correction and acknowledgment

Acknowledgment (ACK):

- The receiver sends acknowledgments to the sender to confirm that it has received data segments. An ACK contains the sequence number of the next segment the receiver expects.

Retransmission:

- If a sender doesn't receive an ACK for a segment within a specific timeout period (Retransmission Time-Out or RTO), it assumes the segment was lost or corrupted and retransmits it.

Fast Retransmit:

- A sender can also trigger a retransmission without waiting for a timeout by receiving three duplicate ACKs from the receiver. This indicates that a segment was received out of order, and the sender can resend the missing segment.

Sequence Numbers:

- TCP assigns a unique sequence number to each byte of data. This allows the receiver to detect and reorder segments that arrive out of sequence and to discard duplicate segments.

Buffering:

- The receiver buffers out-of-order segments until the missing ones arrive, ensuring that data is delivered to the application in the correct, in-order sequence.

TCP Congestion Control

- TCP congestion control manages data flow to prevent network overload. It uses a congestion window and policies to adjust transmission rates. While the receiver sets a window size, the network also influences it by signaling the sender to slow down if it cannot handle the data rate.

Congestion Policy in TCP

- **Slow Start Phase:** Starts slow increment is exponential to the threshold.
- **Congestion Avoidance Phase:** After reaching the threshold increment is by 1.
- **Congestion Detection Phase:** The sender goes back to the Slow start phase or the Congestion avoidance phase.

1. Slow Start Phase

- TCP increases its congestion window exponentially to quickly probe the network capacity at the beginning of a connection.
- **Exponential Increment**: In this phase after every RTT the congestion window size increments exponentially.
- **Example:** If the initial congestion window size is 1 segment, and the first segment is successfully acknowledged, the congestion window size becomes 2 segments. If the next transmission is also acknowledged, the congestion window size doubles to 4 segments. This exponential growth continues as long as all segments are successfully acknowledged.
- Initially cwnd = 1
  After 1 RTT, cwnd = 2^(1) = 2
  2 RTT, cwnd = 2^(2) = 4
  3 RTT, cwnd = 2^(3) = 8

2. Congestion Avoidance Phase

- TCP increases its congestion window linearly to prevent network congestion after the slow start.
- **Additive Increment:** This phase starts after the threshold value also denoted as ssthresh. The size of CWND (Congestion Window) increases additive. After each RTT cwnd = cwnd + 1.
- **For example:** if the congestion window size is 20 segments and all 20 segments are successfully acknowledged within an RTT, the congestion window size would be increased to 21 segments in the next RTT. If all 21 segments are again successfully acknowledged, the congestion window size will be increased to 22 segments, and so on.
- Initially cwnd = i
  After 1 RTT, cwnd = i+1
  2 RTT, cwnd = i+2
  3 RTT, cwnd = i+3

3. Congestion Detection Phase

- TCP detects network congestion through packet loss or duplicate ACKs and triggers window reduction to control traffic
- **Multiplicative Decrement:** If congestion occurs, TCP reduces the congestion window. The sender detects congestion when a packet needs retransmission, either due to an RTO timeout or receiving three duplicate ACKs.
- **Case 1:** Retransmission due to Timeout – In this case, the congestion possibility is high.
- (a) ssthresh is reduced to half of the current window size.
  (b) set cwnd = 1
  (c) start with the slow start phase again.
- **Case 2:** Retransmission due to 3 Acknowledgement Duplicates – The congestion possibility is less.
- (a) ssthresh value reduces to half of the current window size.
  (b) set cwnd= ssthresh
  (c) start with congestion avoidance phase

TELNET

- **TELNET** stands for Teletype Network. It is a **client/server application protocol** that provides access to virtual terminals of remote systems on local area networks or the Internet. The local computer uses a telnet client program and the remote computers use a telnet server program.
- **TELNET** is a type of protocol that enables one computer to connect to the local computer. It is used as a standard **TCP/IP protocol** for virtual terminal service which is provided by **ISO**. The computer which starts the connection is known as the **local computer**. The computer which is being connected to i.e. which accepts the connection known as the **remote computer**. During telnet operation, whatever is being performed on the remote computer will be displayed by the local computer. Telnet operates on a client/server principle.

Logging in TELNET

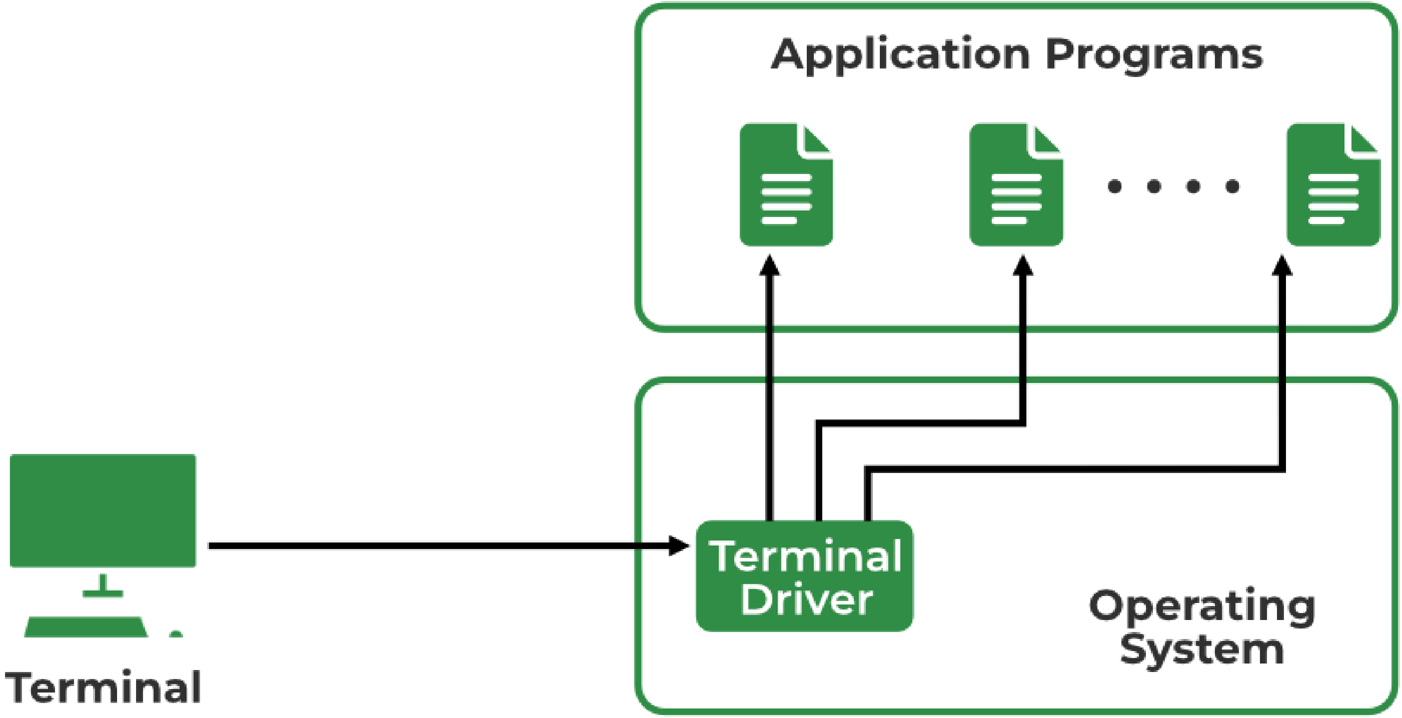
- The logging process can be further categorized into two parts:
- Local Login
- Remote Login

1. Local Login

- Whenever a user logs into its local system, it is known as local login.

The Procedure of Local Login

- Keystrokes are accepted by the terminal driver when the user types at the terminal.
- Terminal Driver passes these characters to OS.
- Now, OS validates the combination of characters and opens the required application.

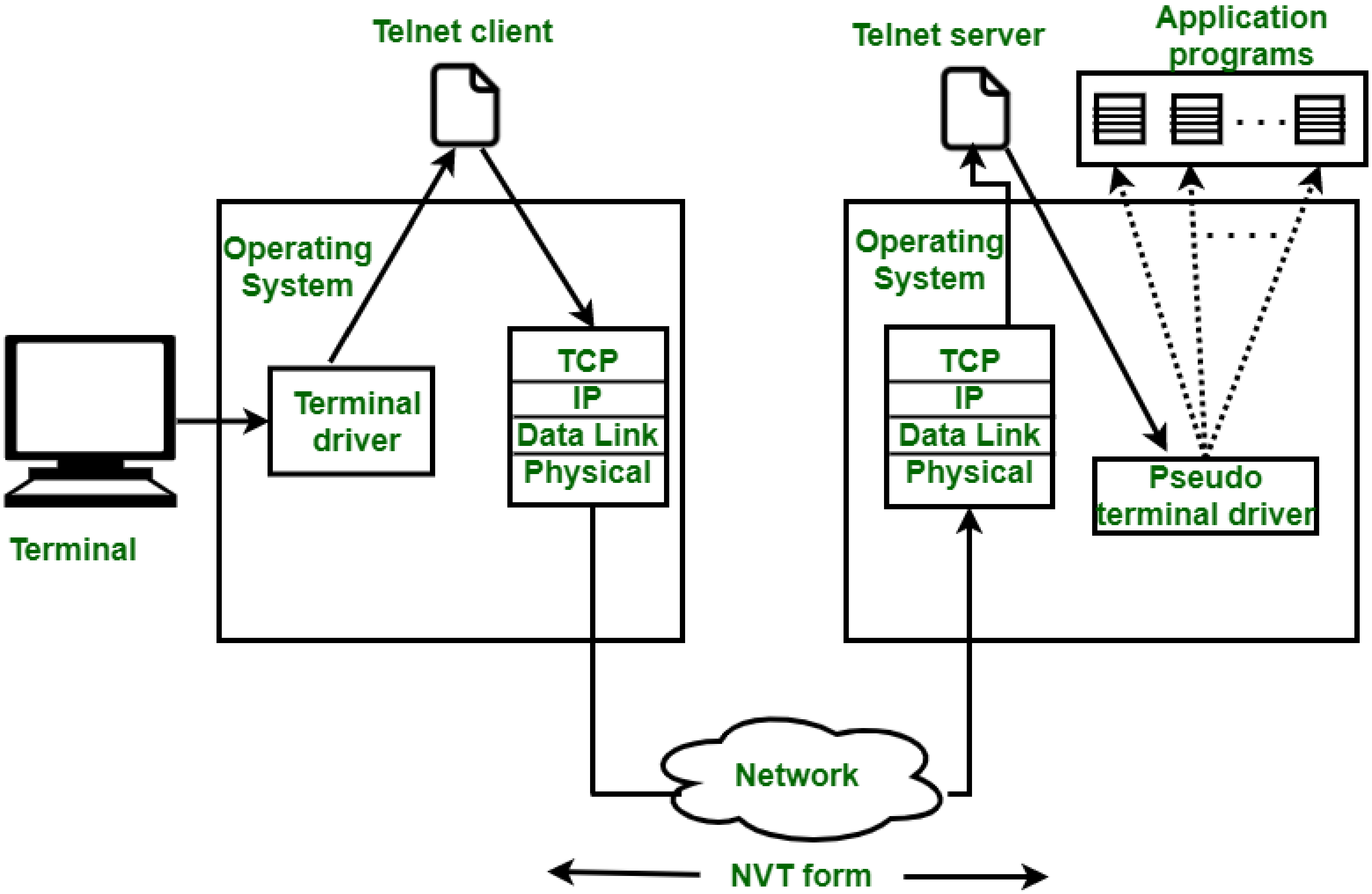

Application Programs
Terminal
Terminal Driver
Operating System

2. Remote Login

- Remote Login is a process in which users can log in to a remote site i.e. computer and use services that are available on the remote computer. With the help of remote login, a user is able to understand the result of transferring the result of processing from the remote computer to the local computer.

The Procedure of Remote Login

- When the user types something on the local computer, the local operating system accepts the character.
- The local computer does not interpret the characters, it will send them to the TELNET client.
- TELNET client transforms these characters to a universal character set called Network Virtual Terminal (NVT) characters and it will pass them to the local TCP/IP protocol Stack.


Telnet client
Telnet server
Application programs
Operating System
Operating System
Terminal driver
TCP
IP
Data Link
Physical
TCP
IP
Data Link
Physical
Pseudo terminal driver
Terminal
Network
NVT form

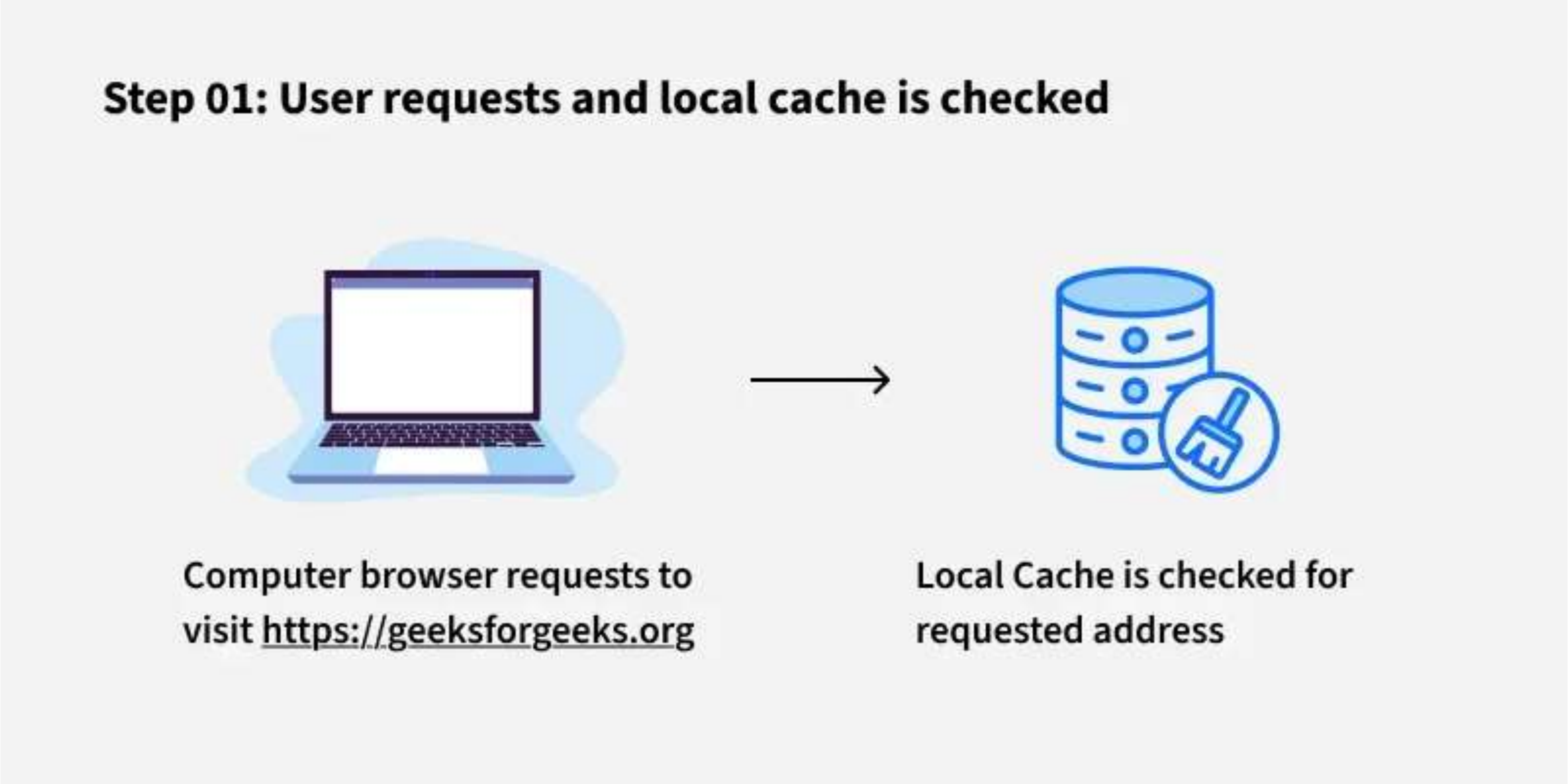
- Commands or text which are in the form of NVT, travel through the Internet and it will arrive at the TCP/IP stack at the remote computer.
- Characters are then delivered to the operating system and later on passed to the TELNET server.
- Then TELNET server changes those characters to characters that can be understandable by a remote computer.
- The remote operating system receives characters from a pseudo-terminal driver, which is a piece of software that pretends that characters are coming from a terminal.
- The operating system then passes the character to the appropriate application program.

Domain Name System

- DNS is a hierarchical and distributed naming system that translates domain names into IP addresses. When you type a domain name like www.srivasaviengg.ac.in into your browser, DNS ensures that the request reaches the correct server by resolving the domain to its corresponding IP address.

Working of DNS

- The DNS process can be broken down into several steps, ensuring that users can access websites by simply typing a domain name into their browser.



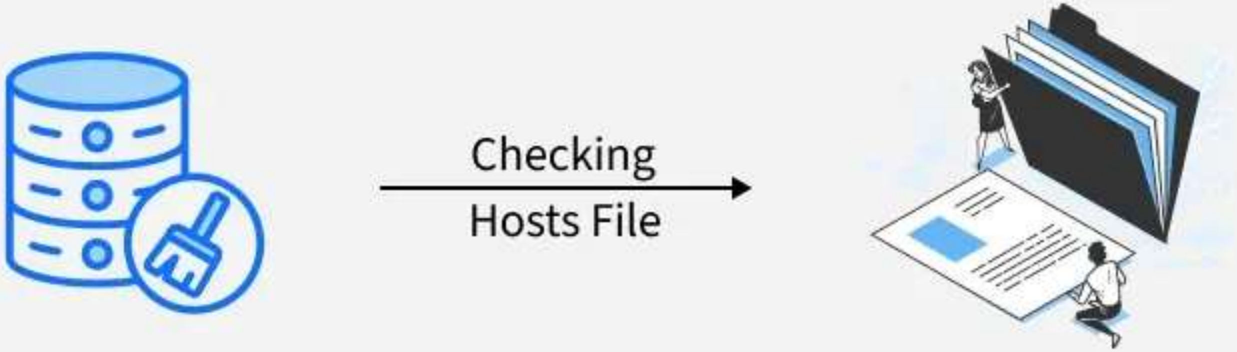
Step 01: User requests and local cache is checked

Computer browser requests to visit https://geeksforgeeks.org

Local Cache is checked for requested address

Step 02: Host Files are checked

Hosts File



Checking

Hosts File

If IP address not found in cache

The hosts file is a system file that maps domain names to IP addresses locally

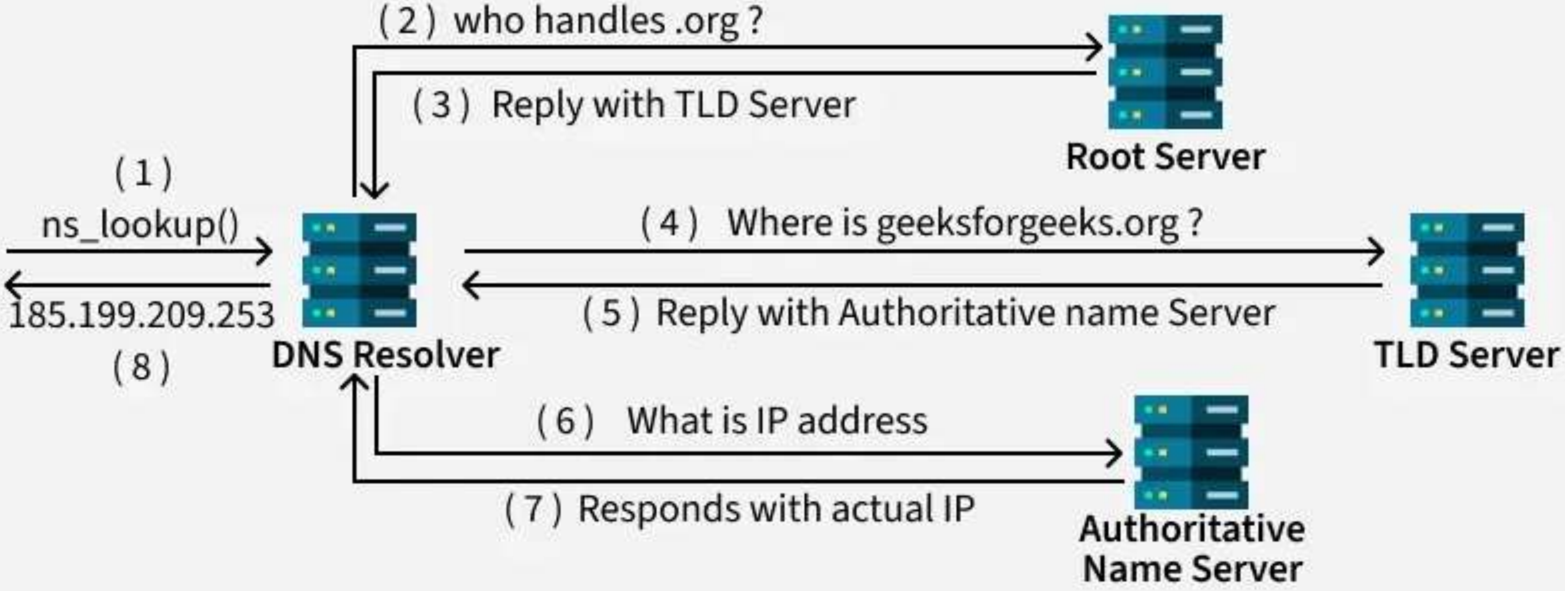
Step 03: Query DNS Resolver

If IP address not found locally

→

DNS Resolver is a server provided by ISP

Step 04: DNS Search by DNS Resolver



(1)
ns_lookup()

185.199.209.253
(8)

DNS Resolver

(2) who handles .org ?

(3) Reply with TLD Server

Root Server

(4) Where is geeksforgeeks.org ?

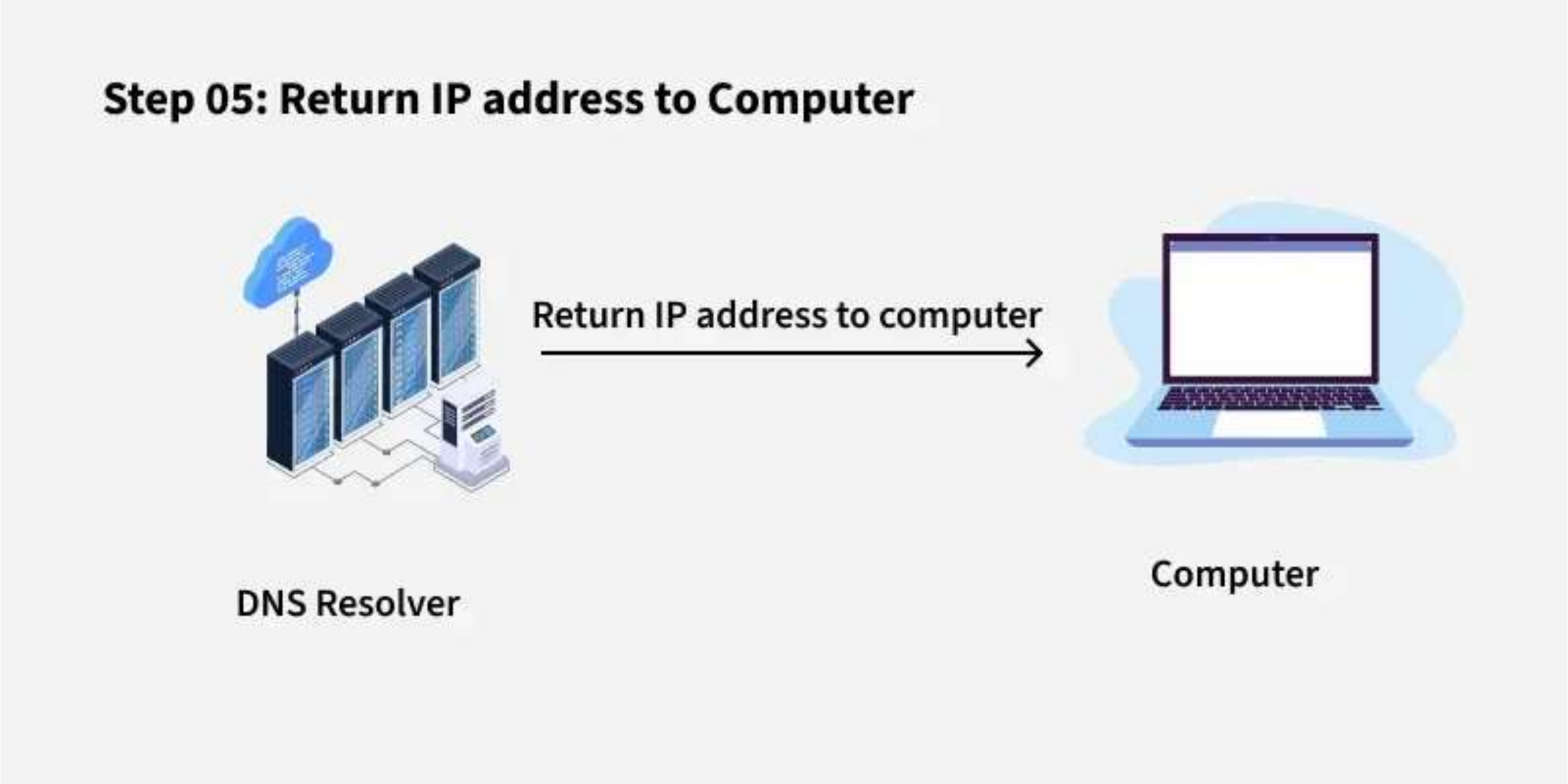
(5) Reply with Authoritative name Server

TLD Server

(6) What is IP address

(7) Responds with actual IP

Authoritative
Name Server



Step 05: Return IP address to Computer

Return IP address to computer

DNS Resolver

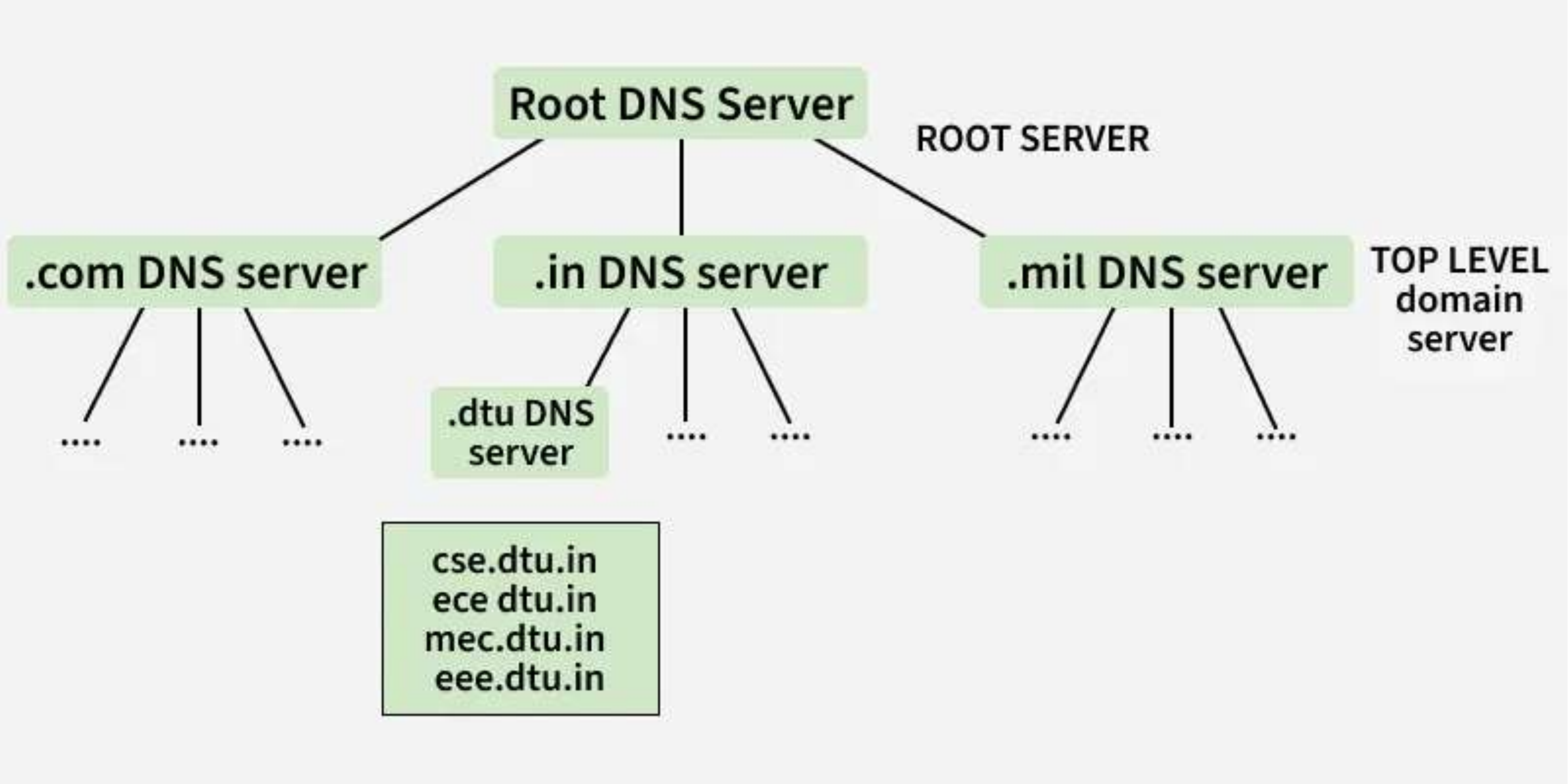
Computer

- **User Input:** You enter a website address (for example, www.geeksforgeeks.org) into your web browser.
- **Local Cache Check:** Your browser first checks its local cache to see if it has recently looked up the domain. If it finds the corresponding IP address, it uses that directly without querying external servers.
- **DNS Resolver Query:** If the IP address isn’t in the local cache, your computer sends a request to a DNS resolver. The resolver is typically provided by your Internet Service Provider (ISP) or your network settings.
- **Root DNS Server:** The resolver sends the request to a root DNS server. The root server doesn’t know the exact IP address for www.geeksforgeeks.org but knows which Top-Level Domain (TLD) server to query based on the domain’s extension (e.g., .org).

- **TLD Server:** The TLD server for .org directs the resolver to the authoritative DNS server for geeksforgeeks.org.
- **Authoritative DNS Server:** This server holds the actual DNS records for geeksforgeeks.org, including the IP address of the website’s server. It sends this IP address back to the resolver.
- **Final Response:** The DNS resolver sends the IP address to your computer, allowing it to connect to the website’s server and load the page.

Structure of DNS

- DNS operates through a hierarchical structure, ensuring scalability and reliability across the global internet infrastructure. Here's how it’s organized:
- **Root DNS Servers:** These are the highest-level DNS servers and know where to find the TLD servers. They are crucial for directing DNS queries to the correct locations.
- **TLD Servers:** These servers manage domain extensions like .com, .org, .net, .edu, .gov and others. They help route queries to the authoritative DNS servers for specific domains.
- **Authoritative DNS Servers:** These are the servers that store the actual DNS records for domain names. They are responsible for providing the correct IP addresses that allow users to reach websites.


Root DNS Server
ROOT SERVER
.com DNS server
.in DNS server
.mil DNS server
TOP LEVEL
domain
server
....
....
....
.dtu DNS
server
....
....
....
....
....
cse.dtu.in
ece dtu.in
mec.dtu.in
eee.dtu.in