JGI JAIN DEEMED-TO-BE UNIVERSITY | SCHOOL OF COMPUTER SCIENCE AND IT

Master of Computer Applications

23MCAC102 – Advanced Computer Networks

Unit - IV

TRANSPORT & APPLICATION LAYER

Module-4 Syllabus

TRANSPORT & APPLICATION LAYER

- Introduction
- Transport Layer Protocols
- TCP Services
- Port Numbers
- User Datagram Protocol
- Transmission Control Protocol
- HTTP
- DHCP
- FTP
- Email
- Telnet – SSH – DNS

TRANSPORT LAYER FUNCTIONS / SERVICES

- The **services** that can be **provided by the transport layer** are
  - i) Process-to-Process Communication
  - ii) Service Point Addressing : **Port Numbers**
  - iii) Encapsulation and Decapsulation
  - iv) Multiplexing and Demultiplexing
  - v) Flow Control
  - vi) Error Control

i) Process-to-Process Communication

- The Transport Layer is **responsible for delivering data to the appropriate application process** on the host computers.
- This involves **multiplexing of data from different application processes**, **i.e. forming data packets,** and adding source and destination port numbers in the header of each Transport Layer data packet.
- **Together with the source and destination IP address, the port numbers constitutes a network socket,** i.e. an identification address of the process-to-process communication.

ii) Service Point Addressing: Port Numbers

- **Ports are the essential ways to address multiple entities** in the same location.
- **Using port addressing it is possible to use more than one network-based application at the same time**.

Three types of Port numbers are used:

- ***Well-known ports*** - These are **permanent port numbers**. They range **between 0 to 1023**. These port numbers are **used by Server Process.**
- ***Registered ports*** **-** The ports ranging from **1024 to 49,151** are **not assigned or controlled.**
- ***Ephemeral ports (Dynamic Ports)*** **–** These are temporary port numbers. They range between **49152–65535**. These port numbers are **used by Client Process**

iii) Encapsulation and Decapsulation

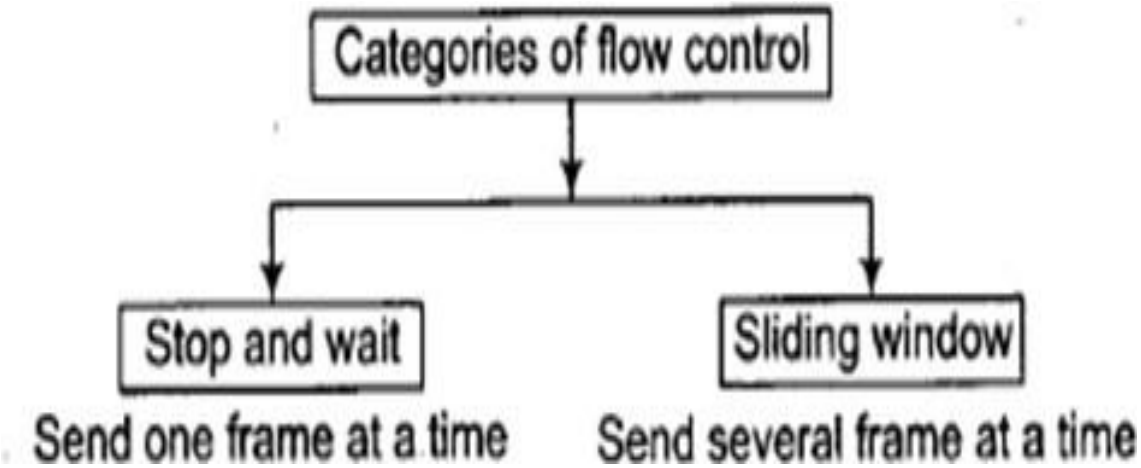
- **To send a message from one process to another**, the transport-layer protocol encapsulates and decapsulates messages.
- **Encapsulation happens at the sender site.** The transport layer receives the data and adds the transport-layer header.
- Decapsulation happens at the receiver site.
- **When the message arrives at the destination transport layer**, the header is dropped and the transport layer delivers the message to the process running at the application layer.

iv) Multiplexing and Demultiplexing

- **Whenever an entity accepts items from more than one source**, this is referred to as ***multiplexing*** (many to one).
- **Whenever an entity delivers items to more than one source**, this is referred to as ***demultiplexing*** (one to many).
- **The transport layer** at the **source performs multiplexing**
- The transport layer at the **destination performs demultiplexing**

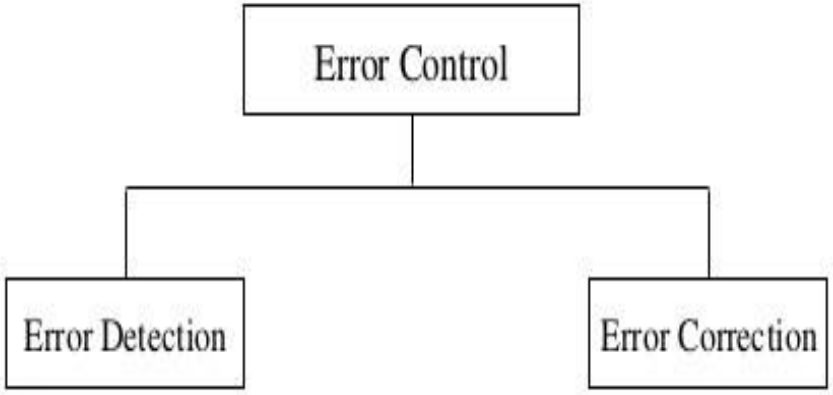
v) Flow Control

- Flow Control is the process of **managing the rate of data transmission between two nodes** to prevent a fast sender from overwhelming a slow receiver.
- It provides a **mechanism for the receiver to control the transmission speed**, so that the receiving node is not overwhelmed with data from transmitting node



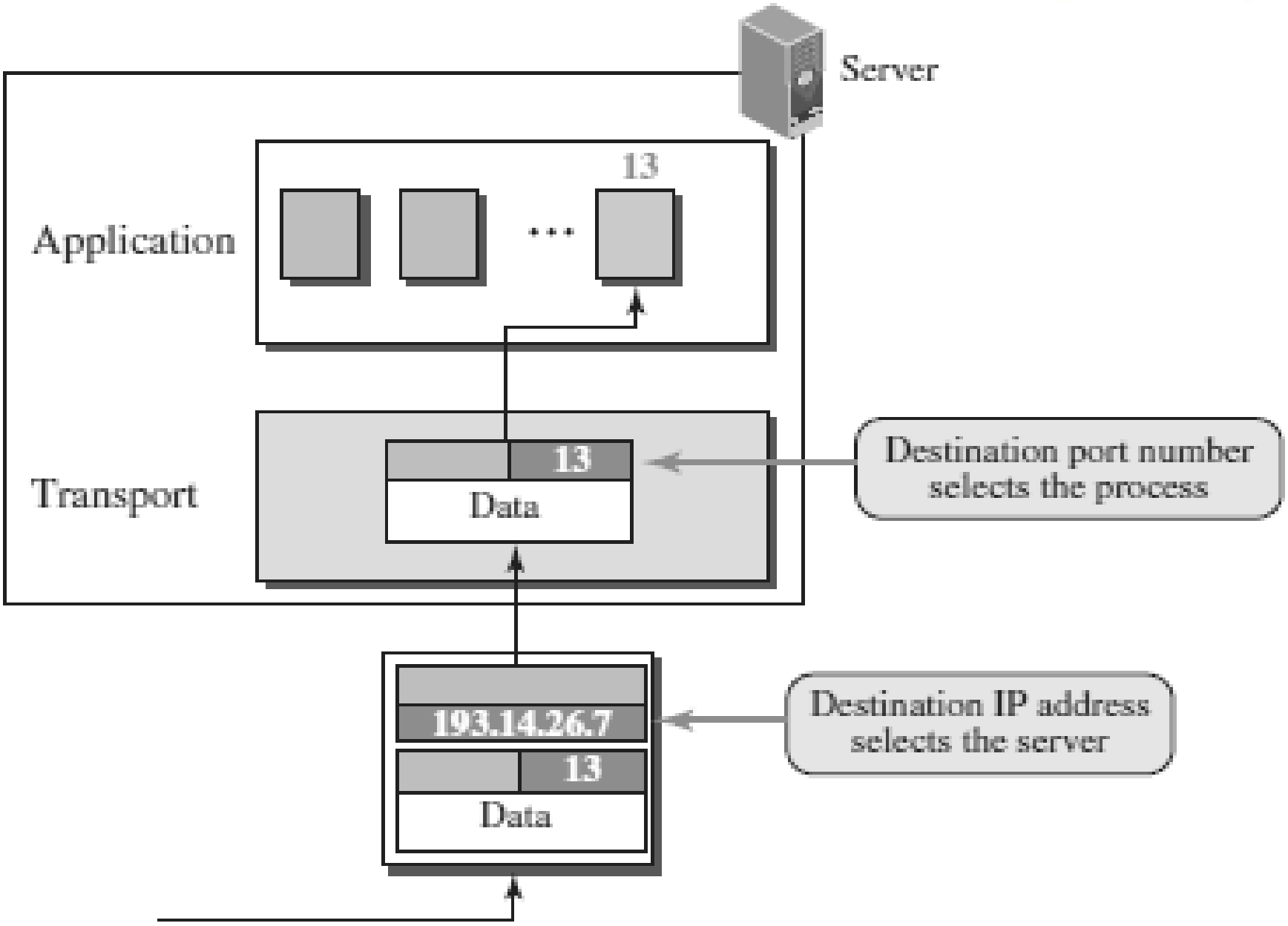
vi) Error Control

- Error control at the transport layer is responsible for
  - **Detecting and discarding corrupted packets**.
  - Keeping track of lost and discarded packets and resending them.
  - **Recognizing duplicate packets and discarding** them.
  - Buffering out-of-order packets **until the missing packets arrive.**
- Error Control involves Error Detection and Error Correction

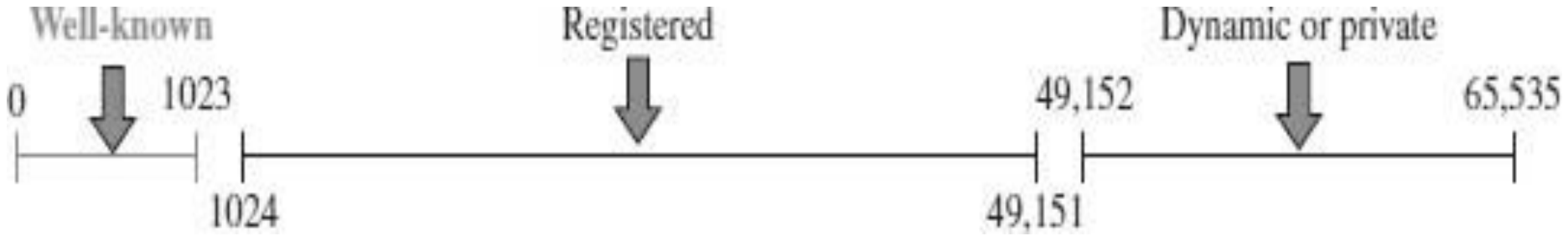


PORT NUMBERS

- A transport-layer protocol usually has several responsibilities.
- One is to create a process-to-process communication.
- **Processes are** programs that run on hosts. It could be **either *server* or *client*.**
- A **process on the local host, called a *client*,** needs services from a **process usually on the remote host, called a *server.***
- **Processes are assigned a unique 16-bit *port number*** on that host.
- Port numbers provide **end-to-end addresses** at the transport layer
- They also provide multiplexing and demultiplexing at this layer.
- The **port numbers are integers between 0 and 65,535** .


Server
13
Application
Transport
13
Data
Destination port number selects the process
193.14.26.7
13
Data
Destination IP address selects the server

- **ICANN** (Internet Corporation for Assigned Names and Numbers) has divided the **port numbers into three ranges:**
  - **Well-known ports**
  - **Registered**
  - **Ephemeral ports (Dynamic Ports)**



WELL-KNOWN PORTS

- These are **permanent port numbers used by the servers**.
- They **range is between 0 to 1023.**
- This port number **cannot be chosen randomly**.
- These port numbers are **universal port numbers for servers**.
- **Every client process knows the well-known port number of the corresponding server process.**

For example:

while the **daytime client process**, a **well-known client program**, **can use an ephemeral (temporary) port number, 52,000**, to identify itself,

→ the **daytime server process must use the well-known (permanent) port number 13.**

Some well-known ports

Port	*Protocol*	*Description*
7	Echo	Echoes back a received datagram
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP-data	File Transfer Protocol
21	FTP-21	File Transfer Protocol
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Service
67	DHCP	Dynamic Host Configuration Protocol
69	TFTP	Trivial File Transfer Protocol
80	HTTP	HyperText Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP-server	Simple Network Management Protocol
162	SNMP-client	Simple Network Management Protocol

EPHEMERAL PORTS (DYNAMIC PORTS)

- The **client program defines itself** with a port number, called the ***ephemeral port number.***
- The word ***ephemeral*** **means "short-lived"** and is used because the life of a client is normally short.
- An ephemeral port number is **recommended to be greater than 1023.**
- These port number ranges from **49,152 to 65,535** .
- They are **neither controlled nor registered**. They can be used as temporary or private port numbers.

REGISTERED PORTS

- The **ports ranging from 1024 to 49,151 are not assigned or controlled.**

- **Three protocols are associated with the Transport layer**
  - **UDP – User Datagram Protocol**
  - **TCP – Transmission Control Protocol**
  - **SCTP - Stream Control Transmission Protocol**

- **UDP -** UDP is an unreliable connectionless transport-layer protocol used for its simplicity and efficiency in applications where error control can be provided by the application-layer process.

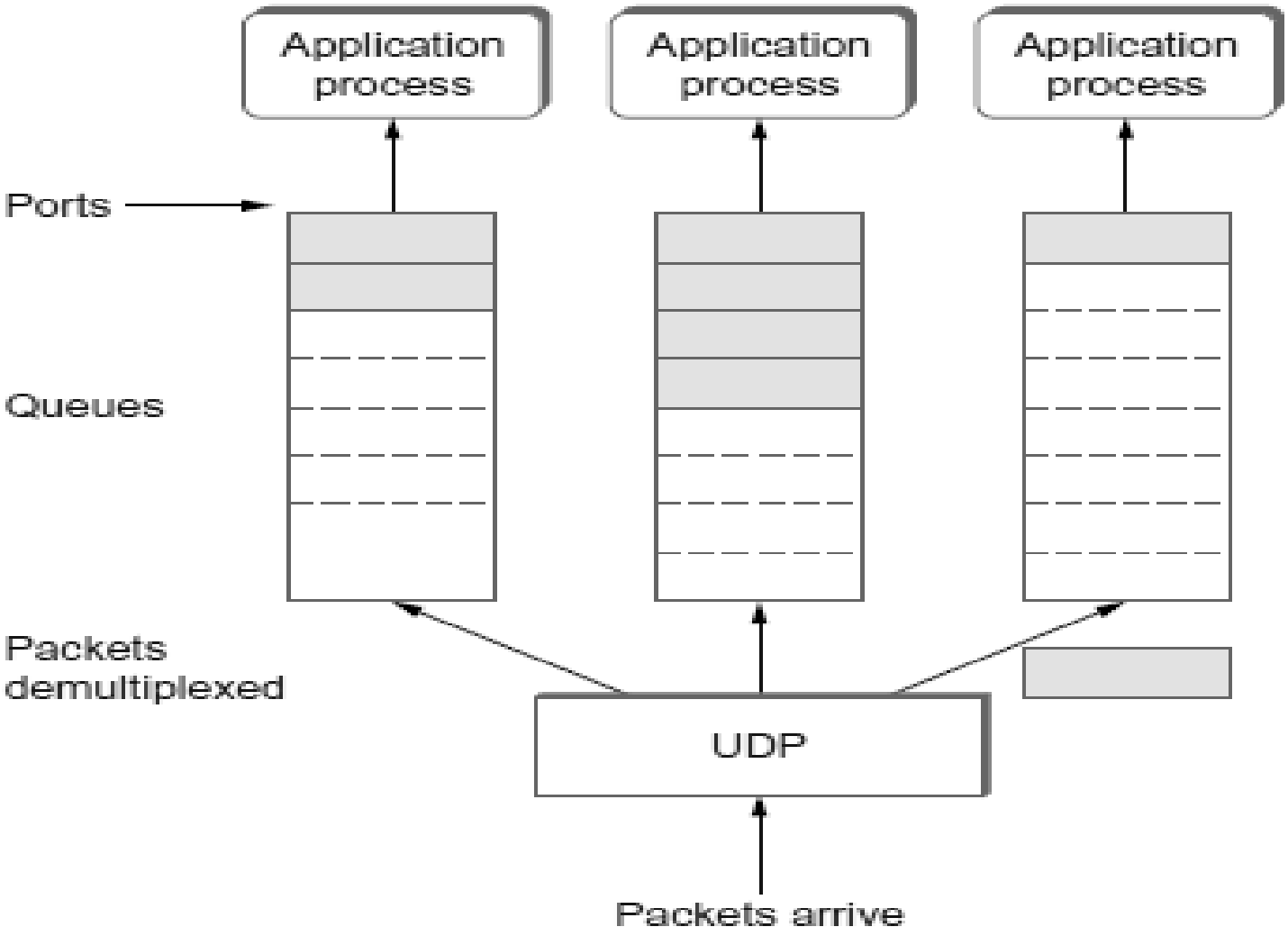
- **TCP -** TCP is a reliable connection-oriented protocol that can be used in any application where reliability is important.

- **SCTP -** SCTP is a new transport-layer protocol designed to combine some features of UDP and TCP in an effort to create a better protocol for multimedia communication.

USER DATAGRAM PROTOCOL

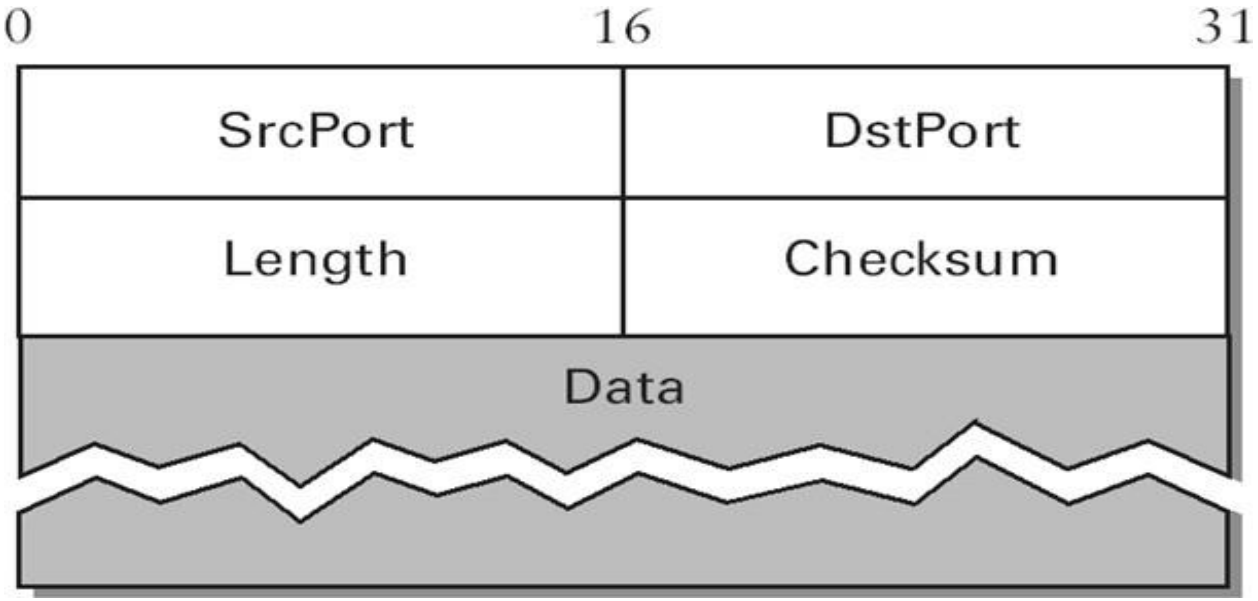
USER DATAGRAM PROTOCOL

- User Datagram Protocol (UDP) is a **connectionless, unreliable transport protocol.**
- UDP adds process-to-process communication to **best-effort service provided by IP.**
- UDP is a very **simple protocol using a minimum of overhead**.
- UDP is a **simple demultiplexer**, which **allows multiple processes on each host to communicate.**
- UDP **does not provide flow control, reliable or ordered delivery.**
- UDP can be **used to send small message where reliability is not expected.**
- **Sending a small message using UDP takes** much less interaction between the sender and receiver.
- **Processes (server/client) are identified by an abstract locator known as port.**


Application process
Application process
Application process
Ports
Queues
Packets demultiplexed
UDP
Packets arrive

UDP DATAGRAM (PACKET) FORMAT

- UDP packets are known as **user *datagrams***.
- These ***user datagrams,*** have a fixed-size header of 8 bytes made of four fields, each of 2 bytes (16 bits).



Source Port Number

- Port number used by process on **source host with 16 bits long.**
- **If the source host is client** (sending request) **then the port number is an temporary one** requested by the process and chosen by UDP.
- **If the source is server** (sending response) **then it is well known port number.**

Destination Port Number

- Port number used by process on Destination host with 16 bits long.
- If the **destination host is the server** (a client sending request) then the **port number is a well known** port number.
- If the **destination host is client** (a server sending response) then **port number is an temporary** one copied by server from the request packet.

Length

- This field denotes the **total length of the UDP Packet (Header plus data)**
- The total length of any UDP datagram **can be from 0 to 65,535 bytes.**

Checksum

- UDP **computes its checksum over the UDP header, the contents of the message body, and something called the pseudoheader.**
- The pseudoheader consists of three fields from the IP header—protocol number, source IP address, destination IP address plus the UDP length field.

Data

- Data field defines the actual payload to be transmitted.
- Its size is variable.

UDP SERVICES

- **Process-to-Process Communication**
  - UDP provides process-to-process communication **using socket addresses, a combination of IP addresses and port numbers.**
- **Connectionless Services**
  - UDP provides a connectionless service.
  - There is no connection establishment and no connection termination .
  - Each user datagram sent by UDP is an independent datagram.
  - There is no relationship between the different user datagrams **even if they are coming from the same source process** and going to the same destination program.
  - The user datagrams are not numbered.
  - Each user datagram **can travel on a different path.**

- **Flow Control**
  - There is **no flow control**, and hence no window mechanism.
  - The **receiver may overflow with incoming messages**.
  - The **lack of flow control means** that the process using UDP should provide for this service, if needed.
- **Error Control**
  - There is **no error control mechanism** in UDP except for the checksum.
  - This means that **the sender does not know if a message has been lost or duplicated.**
  - When the **receiver detects an error through the checksum**, the user **datagram is silently discarded.**
  - The lack of error control means that the process using UDP should provide for this service, if needed.

- **Checksum**
  - UDP **checksum calculation includes three sections**:
  - **a pseudoheader, the UDP header, and the data** coming from the application layer.
  - The pseudoheader is the part of the header in which the user datagram is to be encapsulated with some fields filled with 0s.
- ***Optional Inclusion of Checksum***
  - The **sender of a UDP packet can choose not to calculate the checksum.**
  - In this case, the **checksum field is filled with all 0s** before being sent.
  - In the situation where the sender decides to calculate the checksum, but **it happens that the result is all 0s**, **the checksum is changed to all 1s before the packet is sent**.

- **Congestion Control**
  - Since UDP is a connectionless protocol, it **does not provide congestion control.**
  - **UDP assumes that the packets sent are small** and sporadic (occasionally or at irregular intervals) and cannot create congestion in the network.
  - This **assumption may or may not be true**, when UDP is used for interactive real-time transfer of audio and video.
- **Encapsulation and Decapsulation**
  - To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages.
- **Queuing**
  - In UDP, queues are associated with ports.
  - At the client site, when a process starts, it requests a port number from the operating system.

- **Multiplexing and Demultiplexing**
  - In a host running a transport protocol suite, there is only one UDP but possibly several processes that may want to use the services of UDP.
  - To handle this situation, UDP multiplexes and demultiplexes.

APPLICATIONS OF UDP

- UDP is **used for management processes such as SNMP.**
- UDP is **used for route updating protocols such as RIP.**
- UDP is a **suitable transport protocol for multicasting**. Multicasting capability is embedded in the UDP software
- UDP is **suitable for a process with internal flow and error control** mechanisms such as **Trivial File Transfer Protocol** (TFTP).
- UDP is **suitable for a process that requires simple request-response communication with little concern for flow and error control.**
- UDP is normally used for **interactive real-time applications that cannot tolerate uneven delay** between sections of a received message.

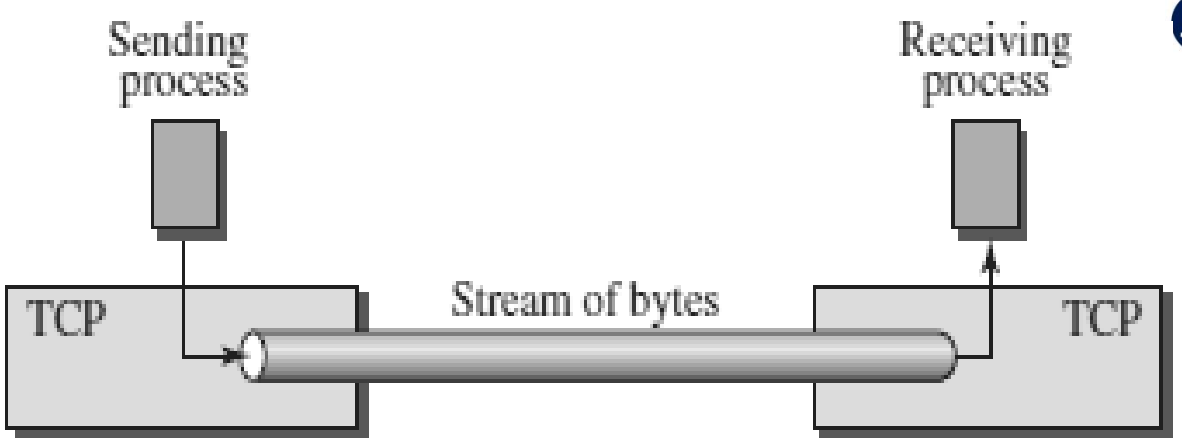
TRANSMISSION CONTROL PROTOCOL (TCP)

TRANSMISSION CONTROL PROTOCOL (TCP)

- TCP is a **reliable, connection-oriented, byte-stream protocol**.
- TCP **guarantees the reliable, in-order delivery of a stream of bytes.**
- **TCP includes a flow-control mechanism** for each of these byte streams that allow the receiver to limit how much data the sender can transmit at a given time.
- TCP **supports a demultiplexing mechanism** that **allows multiple application programs on any given host to simultaneously** carry on a conversation with their peers.
- TCP **also implements congestion-control mechanism**. The idea of this mechanism **is to prevent sender from overloading the network**.

TCP SERVICES

- **Process-to-Process Communication**
  - TCP provides **process-to-process communication using port numbers.**
- **Stream Delivery Service**
  - TCP is a stream-oriented protocol.
  - TCP **allows the sending process to deliver data as a stream of bytes** and allows the receiving process to obtain data as a stream of bytes.
  - TCP creates an environment in which the **two processes seem to be connected by an imaginary "tube"** that carries their bytes across the Internet.
  - The sending process produces (writes to) the stream and the receiving process consumes (reads from) it.



- **Full-Duplex Communication**
  - TCP **offers full-duplex service**, where data can flow in both directions at the same time.
  - Each **TCP endpoint then has its own sending and receiving buffer**, and segments move in both directions.

- **Multiplexing and Demultiplexing**

- TCP performs **multiplexing at the sender and demultiplexing at the receiver.**

Connection-Oriented Service

- TCP is a connection-oriented protocol.
- A **connection needs to be established for each pair of processes**.
- When a process at site A wants to send to and receive data from another process at site B, the following **three phases occur**:

i) The two TCP's **establish a logical connection** between them.

ii) **Data are exchanged** in both directions.

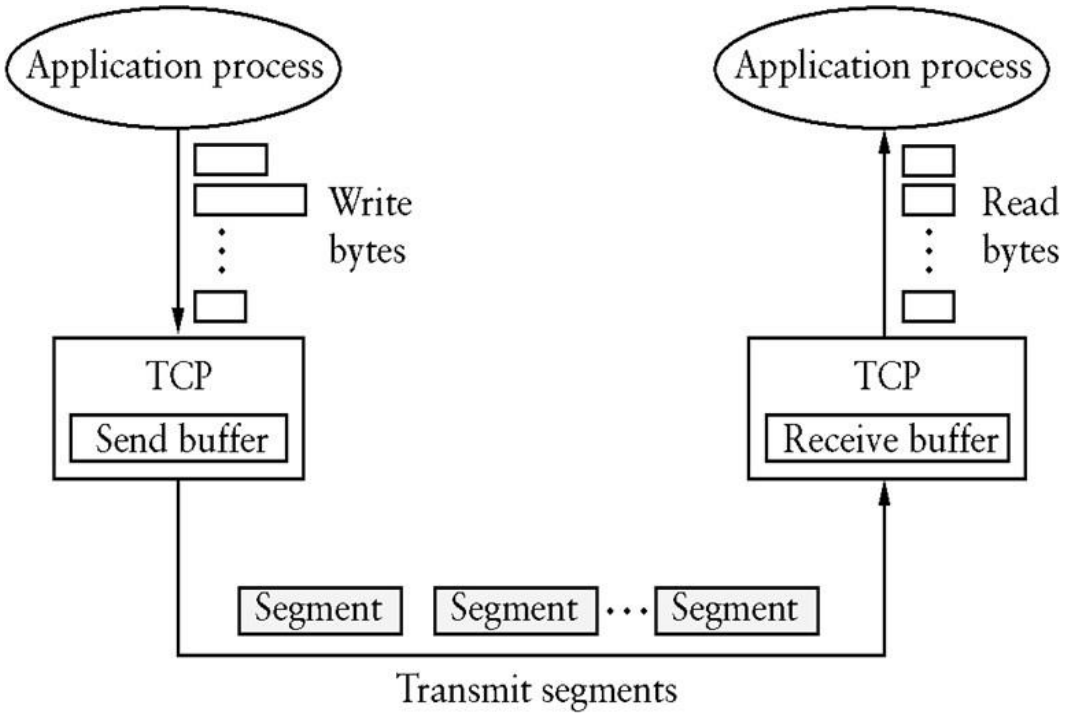
iii) The **connection is terminated.**

Reliable Service

- TCP is a reliable transport protocol.
- It **uses an acknowledgment mechanism to check the safe and sound arrival of data.**

TCP SEGMENT

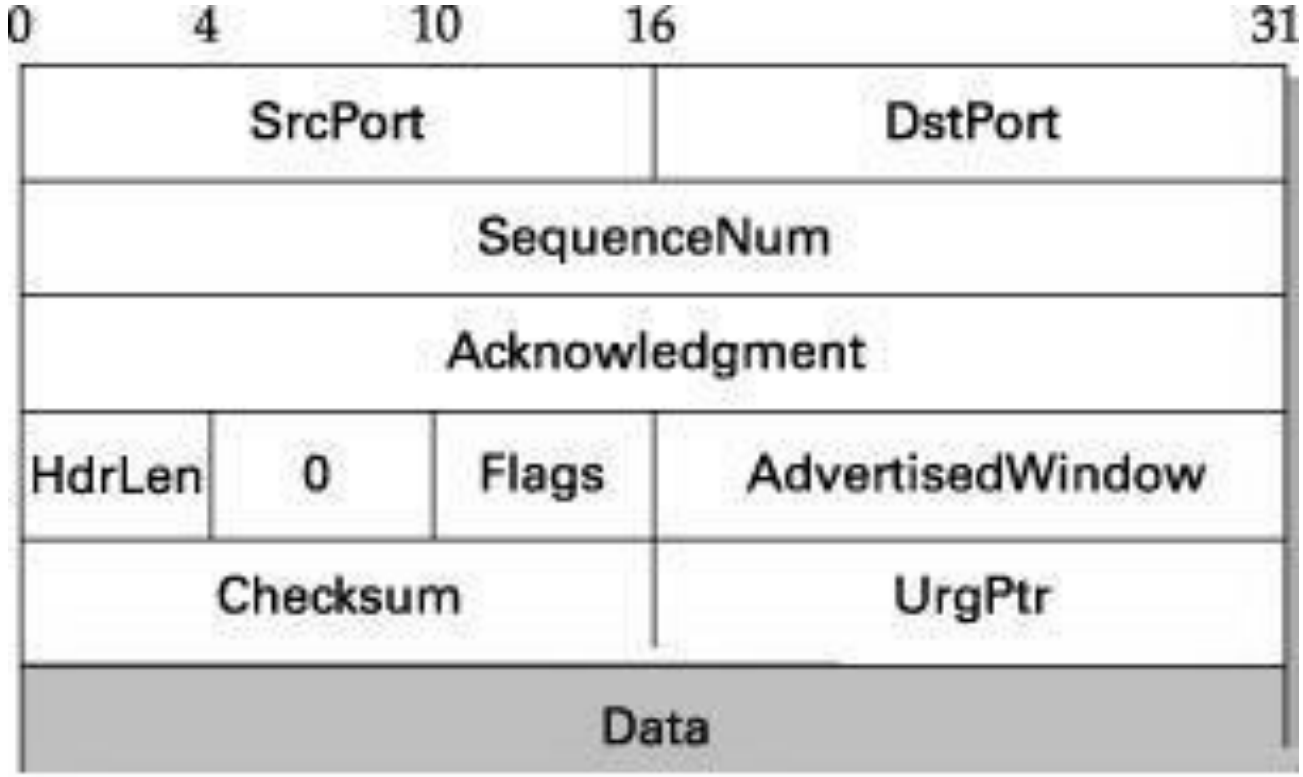
- A **packet in TCP is called a segment**.
- Data unit exchanged between TCP peers are called ***segments*.**
- A **TCP segment encapsulates** the **data received from the application layer.**
- The **TCP segment is encapsulated in an IP datagram**, which in turn is encapsulated in a frame at the data-link layer.



- TCP is a **byte-oriented protocol**, which means that the **sender writes bytes into a TCP connection** and **the receiver reads bytes out of the TCP connection.**
- TCP does not, itself, transmit individual bytes over the Internet.
- TCP on the **source host buffers enough bytes from the sending process to fill a reasonably sized packet** and **then sends this packet to its peer on the destination host.**
- TCP on the destination host then empties the contents of the packet into a receive buffer, and the receiving process reads from this buffer at its leisure.
- TCP connection supports byte streams flowing in both directions.
- The packets exchanged between TCP peers are called segments, since each one carries a segment of the byte stream.

TCP PACKET FORMAT

- Each TCP segment **contains the header plus the data**.
- The **segment consists of a header of 20 to 60 bytes**, followed by data from the application program.
- The **header is 20 bytes if there are no options and up to 60 bytes if it contains options.**



TCP PACKET FORMAT (Contd...)

- **SrcPort & DstPort**—port number of source & destination process.
- **SequenceNum**—contains sequence number, i.e. first byte of data segment.
- **Acknowledgment**— byte number of segment, the receiver expects next.
- **HdrLen**—Length of TCP header as 4-byte words.

Flags— contains ***six*** control bits known as flags.

i) URG — segment contains urgent data.

ii) ACK — value of acknowledgment field is valid.

iii) PUSH — sender has invoked the push operation.

iv) RESET — receiver wants to abort the connection.

v) SYN — **To Initiate a new connection** during connection establishment & synchronize sequence numbers

vi) FIN — terminates the TCP connection.

TCP PACKET FORMAT (Contd…)

- **Advertised Window**—defines receiver's window size and acts as flow control.
- **Checksum**—It is computed over TCP header, Data, and pseudo header containing IP fields (Length, SourceAddr & DestinationAddr).
- **UrgPtr** — used when the segment contains urgent data. It defines a value that must be added to the sequence number.
- **Options -** There can be up to 40 bytes of optional information in the TCP header.

TCP CONNECTION MANAGEMENT

– In TCP, connection-oriented transmission **requires three phases:**

i) Connection Establishment

ii) Data Transfer and

iii) Connection Termination

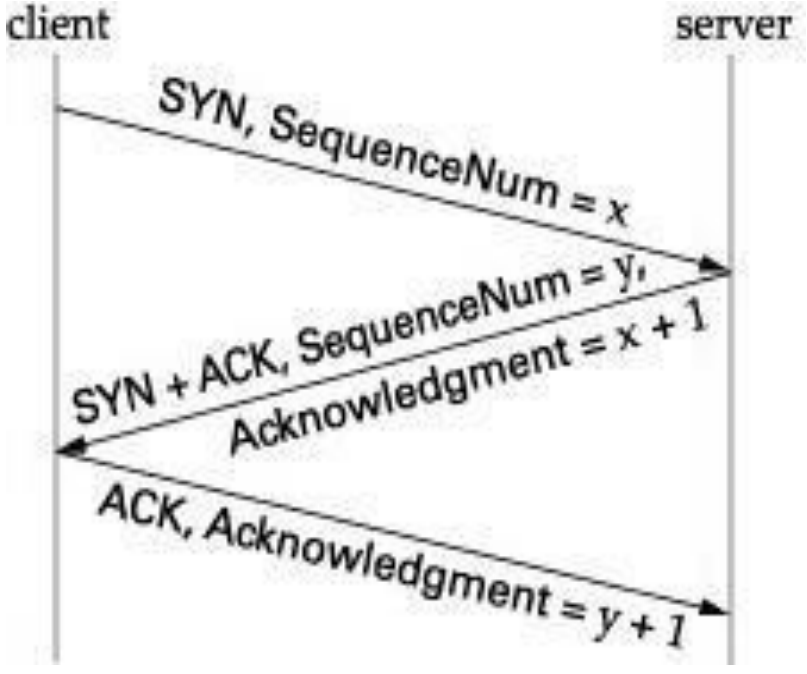
i) Connection Establishment

- While opening a TCP connection the **two nodes (client and server) want to agree on a set of parameters.**
- **The parameters are the starting sequence numbers** that is to be used for their respective byte streams.
- Connection establishment in TCP is a ***three-way handshaking*.**

TCP CONNECTION MANAGEMENT

(Contd…)

- **Client sends a** SYN segment **to the server** containing its initial sequence number (Flags = SYN, SequenceNum = *x*)
- **Server responds** with a segment that acknowledges client's segment and specifies its initial sequence number (Flags = SYN + ACK, ACK = *x* + 1 SequenceNum = *y*).
- Finally, **client responds** with a segment that acknowledges server's sequence number (Flags = ACK, ACK = *y* + 1).

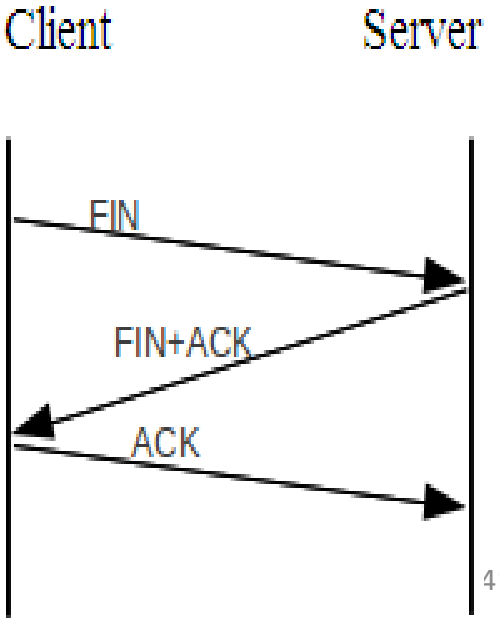


ii) Data Transfer

- After connection is established, **bidirectional data transfer can take place.**
- The **client and server can send data and acknowledgments in both directions**.
- The data traveling in the same direction as an acknowledgment are carried on the same segment.
- The **acknowledgment is piggybacked** with the data.

iii) Connection Termination

→ Connection termination or teardown can be done through ***Three-way Close***—Both client and server close *simultaneously*



DHCP – DYNAMIC HOST CONFIGURATION PROTOCOL

DHCP – DYNAMIC HOST CONFIGURATION PROTOCOL

- The dynamic host configuration protocol is **used to simplify the installation and maintenance of networked computers**.
- DHCP is derived from an earlier protocol called BOOTP.
- **Ethernet addresses** are configured into network by manufacturer and **they are unique.**
- **IP addresses must be unique on a given internetwork** but also must reflect the structure of the internetwork
- Most host **Operating Systems provide a way to manually configure the IP** information for the host

DHCP – DYNAMIC HOST CONFIGURATION PROTOCOL (Contd…)

Drawbacks of Manual Configuration

- A **lot of work to configure** all the hosts in a large network
- Configuration process is error-prune
- It is necessary to **ensure that every host gets the correct network number** and that **no two hosts receive the same IP address.**

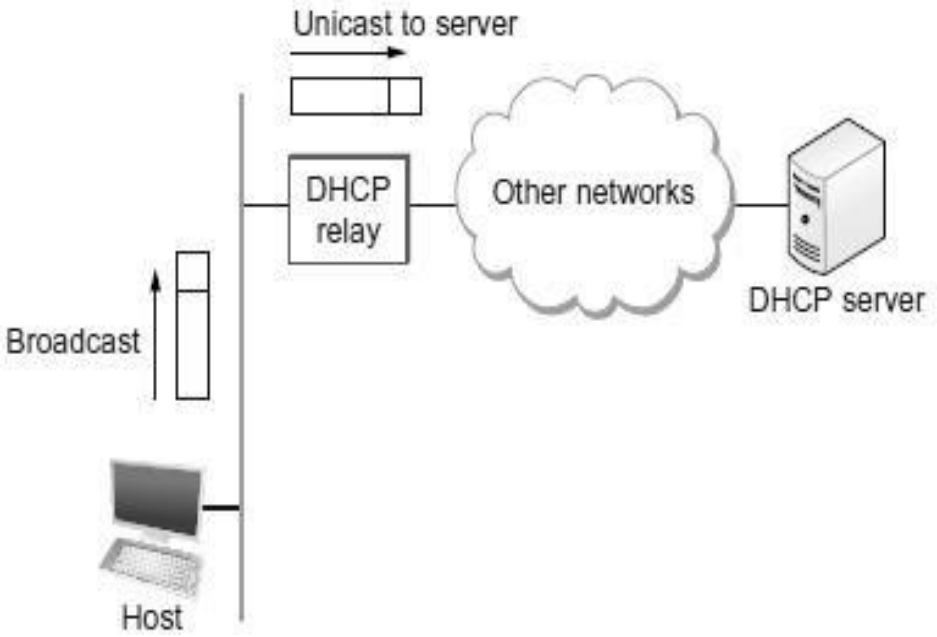
DHCP

- The primary method uses a protocol known as the *Dynamic Host Configuration Protocol* (DHCP).
- The main **goal of DHCP is to minimize the amount of manual configuration** required for a host.

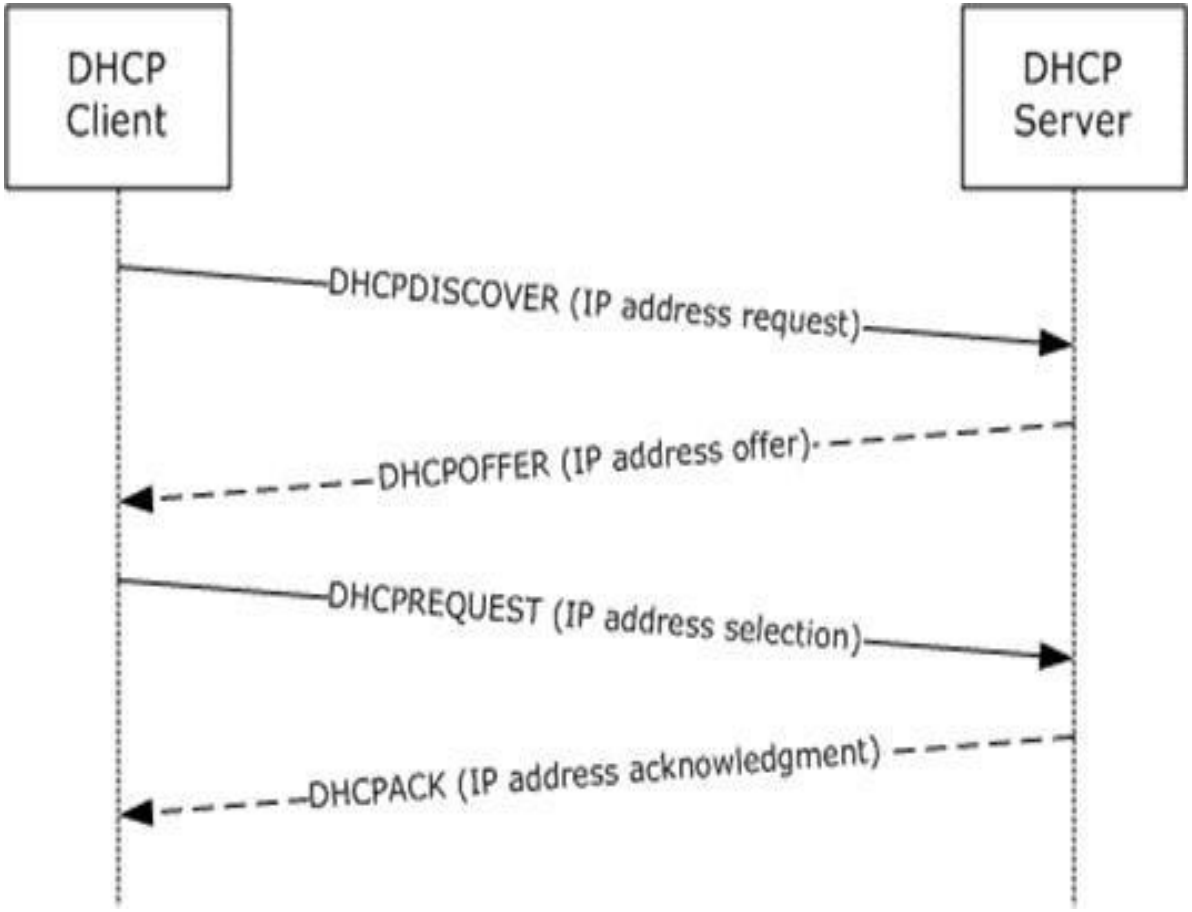
DHCP – DYNAMIC HOST CONFIGURATION PROTOCOL

- **If a new computer is connected to a network**, DHCP can provide it with all the necessary information for full system integration into the network.
- **DHCP is based on a client/server model.**
- DHCP **clients send a request to a DHCP server** to which the **server responds with an IP address**
- **DHCP server is responsible for providing configuration information to hosts.**
- There is **at least one DHCP server for an administrative domain.**
- The DHCP server can function just as a **centralized repository for host configuration** information.

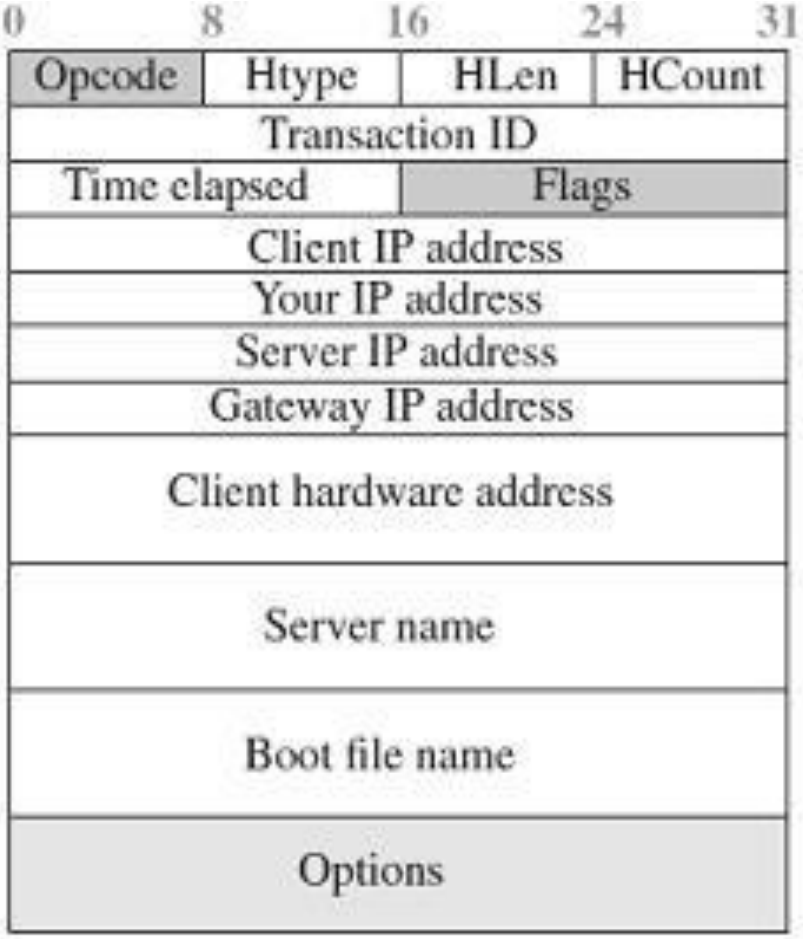
- The **DHCP server maintains a pool of available addresses** that it hands out to hosts on demand.
- A **newly booted or attached host sends a DHCPDISCOVER message to a special IP address** (255.255.255.255., which is an IP broadcast address)
- This means it will be **received by all hosts and routers on that network.**
- DHCP **uses** the concept of a ***relay agent***. There is **at least one relay agent on each network.**
- **DHCP relay agent** is **configured with the IP address of the DHCP server.**



When a relay agent receives a DHCPDISCOVER message, it unicasts it to the DHCP server and awaits for the response, which **it will then send back to the requesting client.**



- **DHCP Message Format**
  - A DHCP packet is actually **sent using the *User Datagram Protocol* (UDP).**



Opcode: Operation code, request (1) or reply (2)
Htype: Hardware type (Ethernet, ...)
HLen: Length of hardware address
HCount: Maximum number of hops the packet can travel
Transaction ID: An integer set by the client and repeated by the server
Time elapsed: The number of seconds since the client started to boot
Flags: First bit defines unicast (0) or multicast (1); other 15 bits not used
Client IP address: Set to 0 if the client does not know it
Your IP address: The client IP address sent by the server
Server IP address: A broadcast IP address if client does not know it
Gateway IP address: The address of default router
Server name: A 64-byte domain name of the server
Boot file name: A 128-byte file name holding extra information
Options: A 64-byte field with dual purpose described in text

FTP (FILE TRANSFER PROTOCOL)

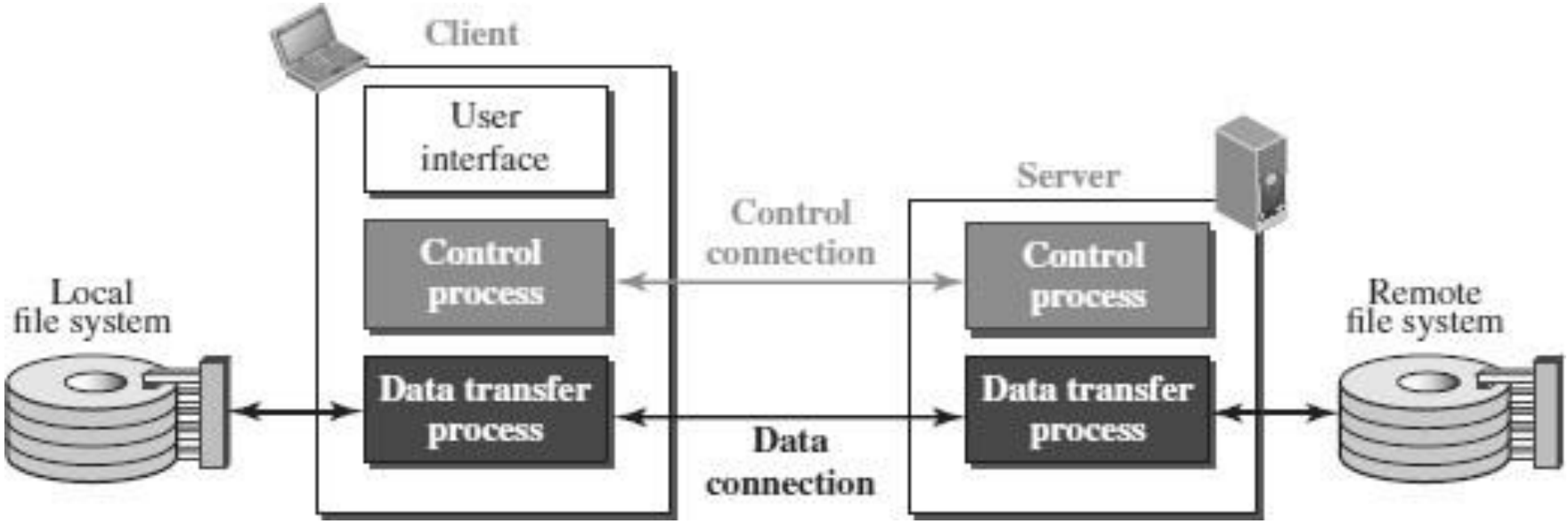
FTP (FILE TRANSFER PROTOCOL)

- FTP stands for File transfer protocol.
- FTP is a standard internet protocol **provided by TCP/IP used for transmitting the files from one host to another**.
- It is mainly used for **transferring the web page files** from their **creator to the computer that acts as a server** for other computers on the internet.
- It is also used **for downloading the files** to computer from other servers.
- Although we **can transfer files using HTTP**, **FTP is a better choice to transfer large files** or to transfer files using different formats.

FTP OBJECTIVES

- It provides the **sharing of files**.
- It is used to **encourage the use of remote computers**.
- It **transfers the data more reliably and efficiently.**

FTP MECHANISM



- The **FTP client has three components:**

  ➔ **User interface, control process, and data transfer process.**

- The **server has two components:**

  ➔ **Server control process and server data transfer process.**

FTP CONNECTIONS

- There are **two types of connections** in FTP

Control Connection and Data Connection.

- The two connections in FTP have different lifetimes.
- The **control connection remains connected** during the entire interactive FTP session.
- The **data connection is opened and then closed for each file transfer activity**.
- When a user starts an FTP session, the control connection opens.
- **While the control connection is open,** the data connection can be opened and closed multiple times if several files are transferred.
- **FTP uses two well-known TCP ports:**
  - **Port 21** is used for the **control connection**
  - **Port 20** is used for the **data connection**

Control Connection:

- The control connection uses very **simple rules for communication**
- Through control connection, **we can transfer a line of command or line of response at a time**
- The control connection is made between the control processes
- The control connection **remains connected during the entire interactive FTP session**

Data Connection:

- The Data Connection uses very **complex rules as data types may vary**
- The data connection is made between data transfer processes
- The data connection **opens when a command comes for transferring the files** and **closes when the file is transferred**

FTP COMMUNICATION

- FTP Communication is achieved through commands and responses.
- FTP **Commands are sent from the client to the server**
- FTP **responses are sent from the server to the client.**
- FTP **Commands are in the form of ASCII uppercase**, which may or may not be followed by an argument

Some of the most common commands are

Command	*Description*
ABOR	Abort the previous command
CDUP	Change to parent directory
CWD	Change to another directory
DELE	Delete a file
LIST	List subdirectories or files
MKD	Create a new directory
PASS	Password
PASV	Server chooses a port
PORT	Client chooses a port
PWD	Display name of current directory
QUIT	Log out of the system
RETR	Retrieve files; files are transferred from server to client
RMD	Delete a directory
RNFR	Identify a file to be renamed
RNTO	Rename the file
STOR	Store files; file(s) are transferred from client to server
STRU	Define data organization (**F**: file, **R**: record, or **P**: page)
TYPE	Default file type (**A**: ASCII, **E**: EBCDIC, **I**: image)
USER	User information
MODE	Define transmission mode (**S**: stream, **B**: block, or **C**: compressed

- Every FTP command generates at least one response.
- A **response has two parts: a three-digit number followed by text.**
- The **numeric part defines the code**; the text part defines needed parameter

Code	*Description*	*Code*	*Description*
125	Data connection open	250	Request file action OK
150	File status OK	331	User name OK; password is needed
200	Command OK	425	Cannot open data connection
220	Service ready	450	File action not taken; file not available
221	Service closing	452	Action aborted; insufficient storage
225	Data connection open	500	Syntax error; unrecognized command
226	Closing data connection	501	Syntax error in parameters or arguments
230	User login OK	530	User not logged in

EMAIL

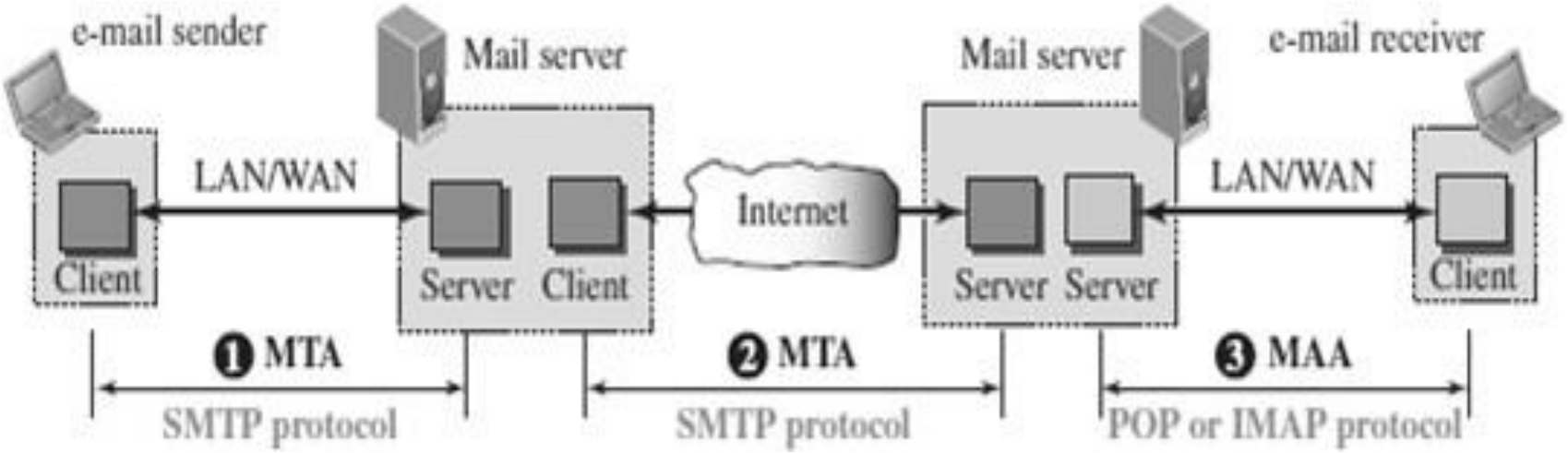
(SMTP, MIME, IMAP, POP)

EMAIL

- One of the most popular Internet services is electronic mail (E-mail).
- Email is one of the oldest network applications.

The **three main components of an Email** are

- **User Agent (UA)**
- **Message Transfer Agent (MTA) – SMTP**
- **Message Access Agent (MAA) - IMAP , POP**



User Agent

- **A mail client application used by an end user to access a mail server to read, compose, and send email messages**

- **When the sender and the receiver of an e-mail are on the same system,**

  ➔ we **need only two User Agents** and

  no Message Transfer Agent

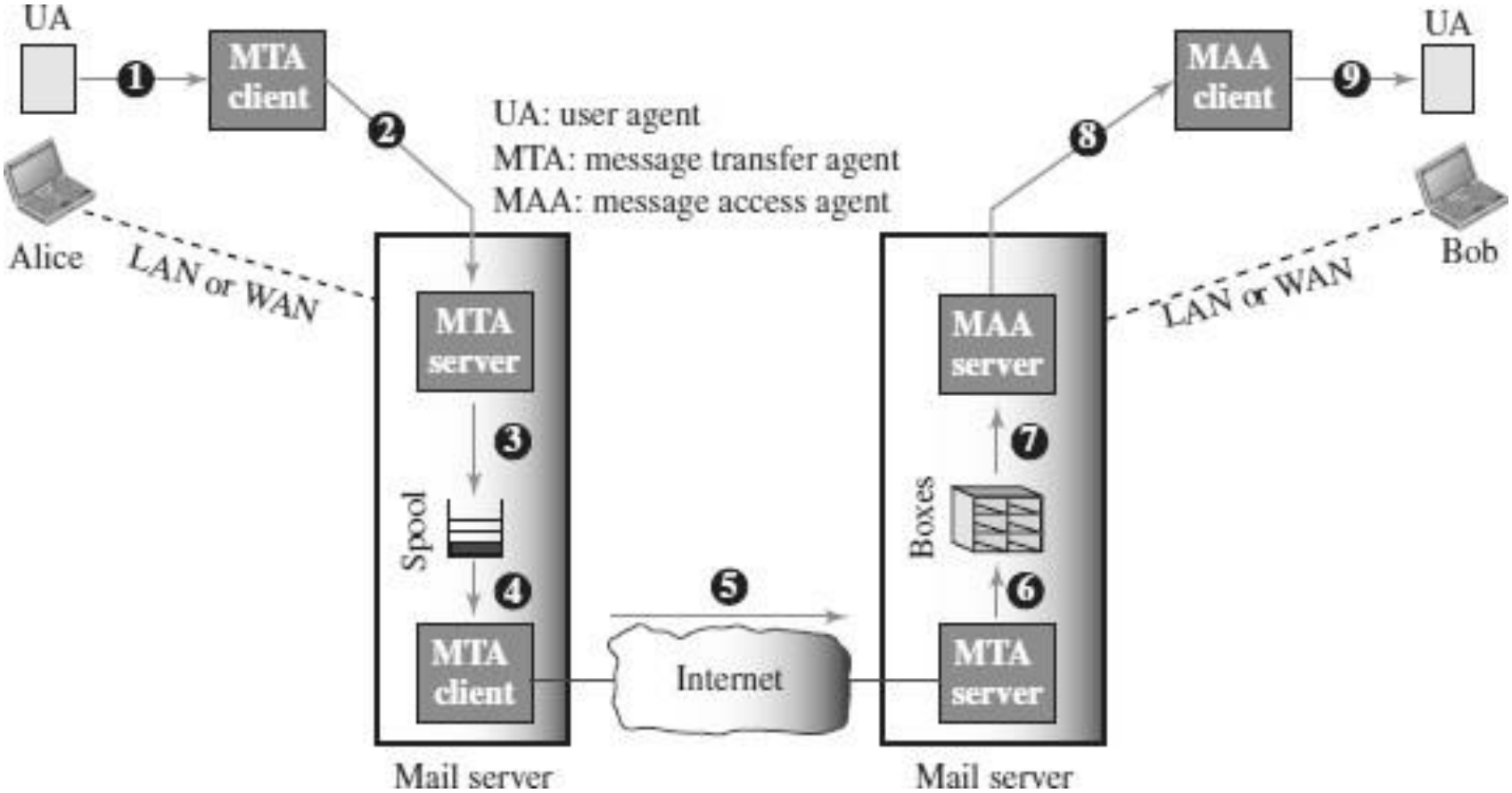
- When the **sender and the receiver of an e-mail are on different system, we**

  ➔ **need two UA,**

  ➔ **two pairs of MTA** (client and server), and

  ➔ **two MAA (**client and server**)**

WORKING OF EMAIL



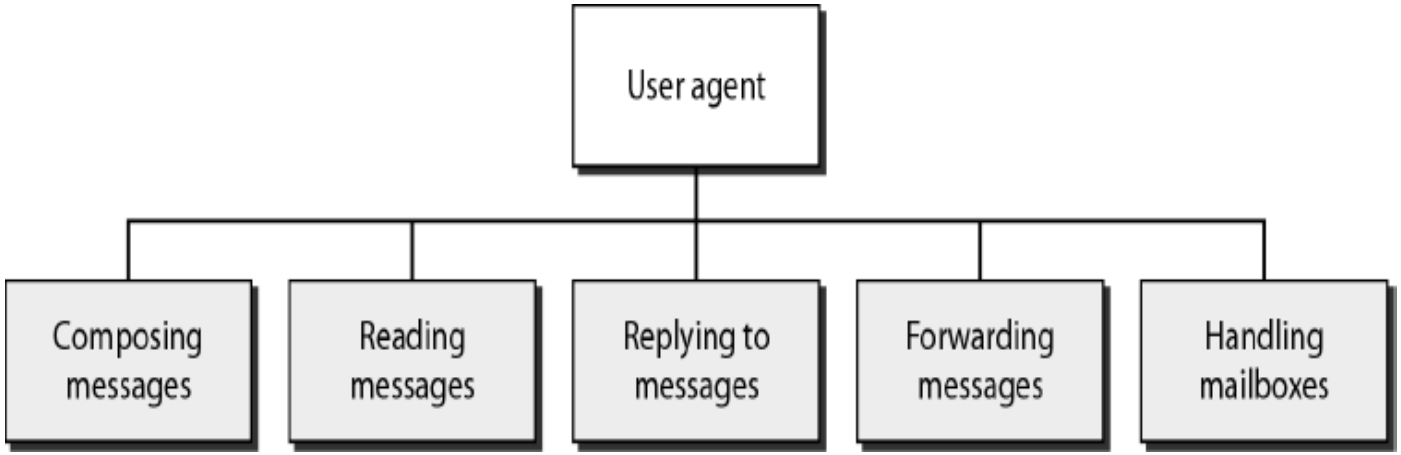
- When Alice needs to send a message to Bob, she runs a UA program to prepare the message and send it to her mail server.
- The mail server at her site uses a queue (spool) to store messages waiting to be sent. The message, however, needs to be sent through the Internet from Alice's site to Bob's site using an MTA.
- Here **two message transfer agents are needed: one client and one server**.
- The **server needs to run all the time** because it does not know when a client will ask for a connection.
- The **client can be triggered by the system when there is a message** in the queue to be sent.
- The **user agent at the Bob site allows Bob to read the received message.**
- Bob later **uses an MAA client to retrieve the message from an MAA server** running on the second server.

USER AGENT (UA)

- The first component of an electronic mail system is the user agent (UA).

- It **provides service to the user to make the process of sending and receiving a message easier**

- A user agent is a software package that **composes, reads, replies to, and forwards messages**. It also handles local mailboxes on the user computers



MESSAGE TRANSFER AGENT (MTA)

- The **actual mail transfer is done through** message transfer agents (MTA).
- To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA.
- The formal **protocol that defines the MTA client and server** in the Internet is called **Simple Mail Transfer Protocol (SMTP).**

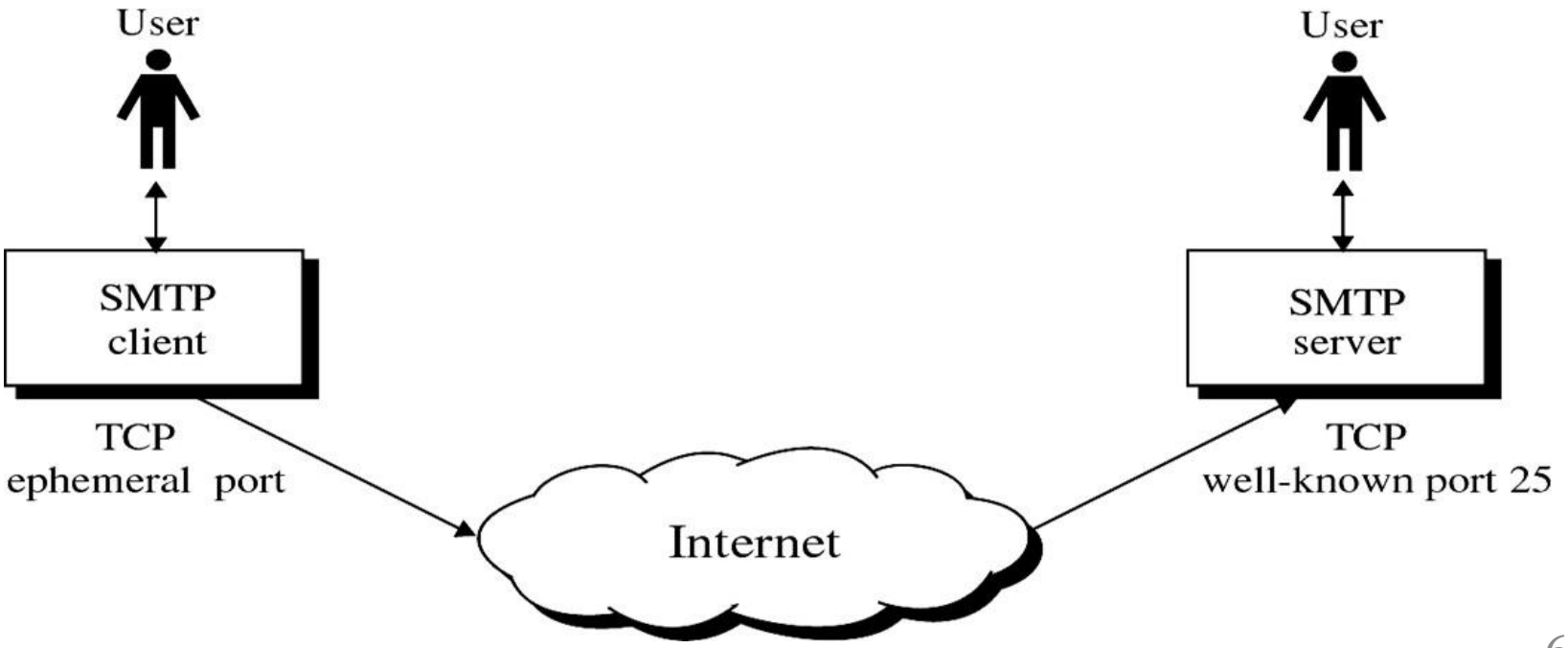
MESSAGE ACCESS AGENT (MAA)

- **MAA is a software that pulls messages out of a mailbox.**
- **POP3 and IMAP4 are examples of MAA.**

SIMPLE MAIL TRANSFER PROTOCOL (SMTP)

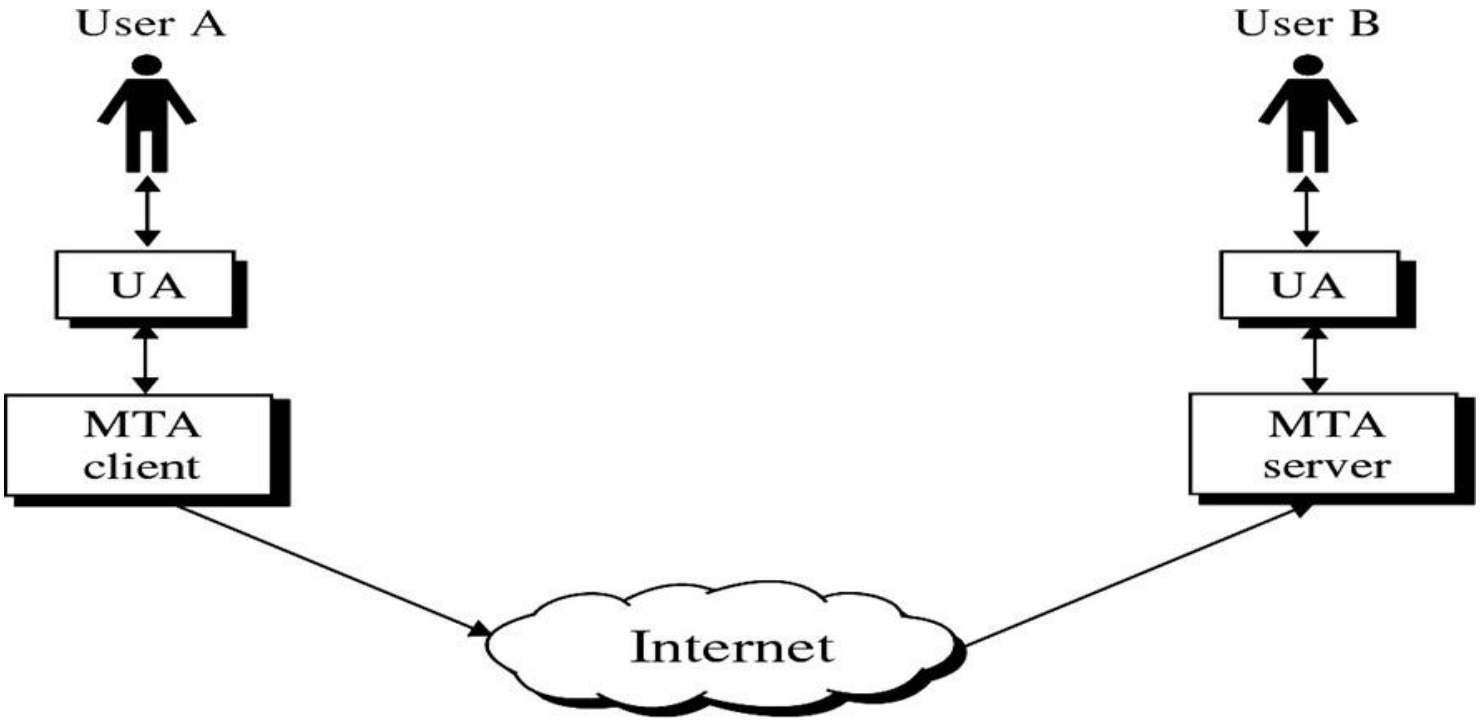
SMTP

- SMTP is the standard protocol **for transferring mail between hosts in the TCP/IP** protocol suite.
- SMTP is **not concerned with the format or content** of messages themselves.
- SMTP uses information written on the *envelope* of the mail (**message header**), but does not look at the *contents* (message body) of the envelope.

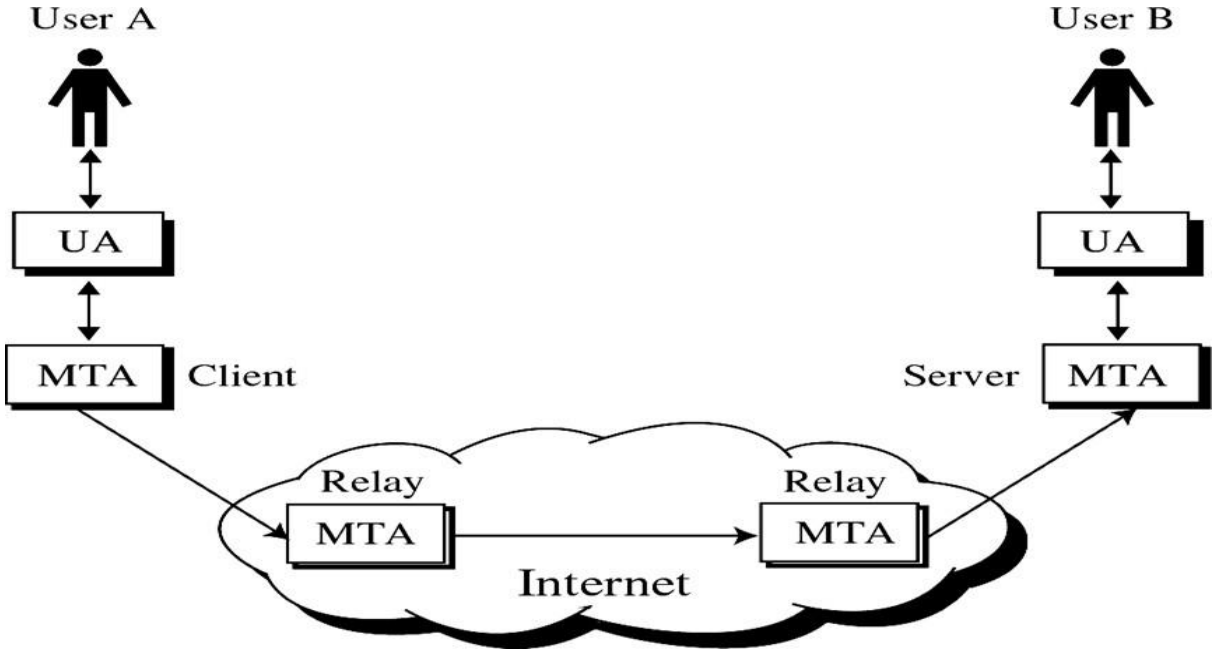


SMTP clients and servers have two main components

- **User Agents(UA)** – Prepares the message, encloses it in an envelope.
- **Mail Transfer Agent (MTA)** – Transfers the mail across the internet



- SMTP also **allows the use of Relays allowing other MTAs to relay the mail.**



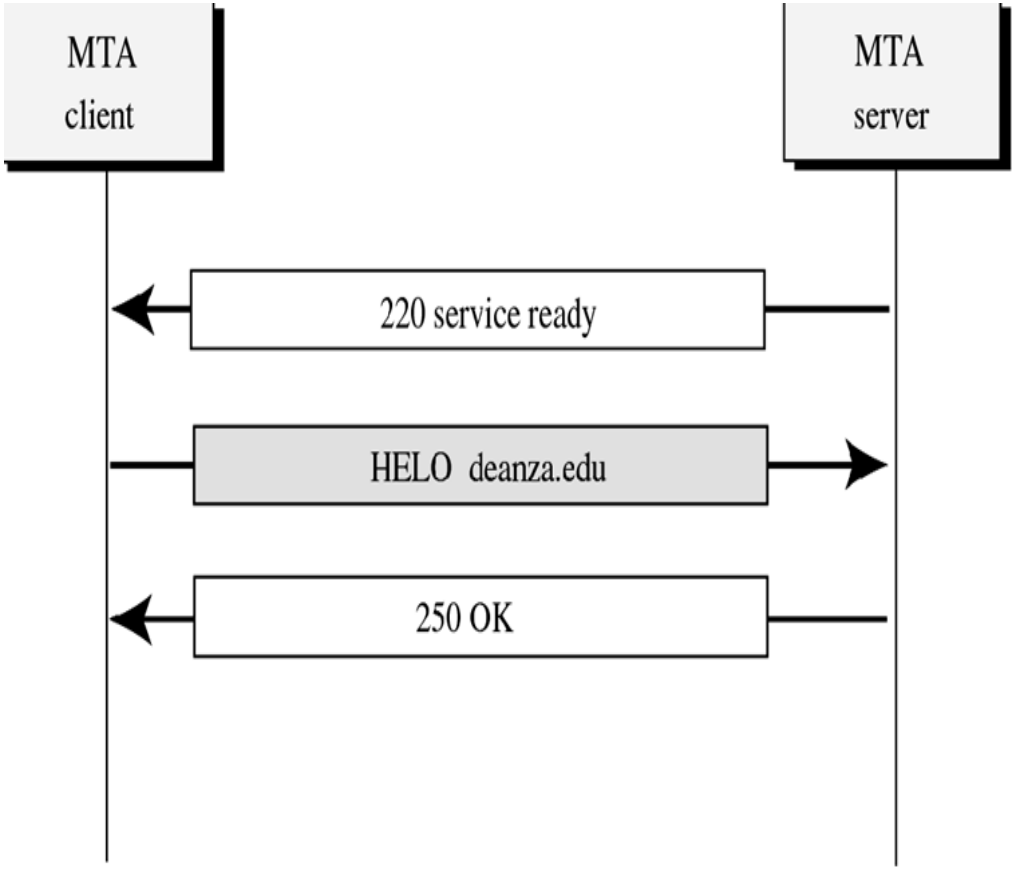
SMTP OPERATIONS

Basic **SMTP operation occurs in three phases:**

1. Connection Setup
2. Mail Transfer
3. Connection Termination

Connection Setup

- An **SMTP sender will attempt to set up a TCP connection with a target host when it has one or more mail messages to deliver** to that host.

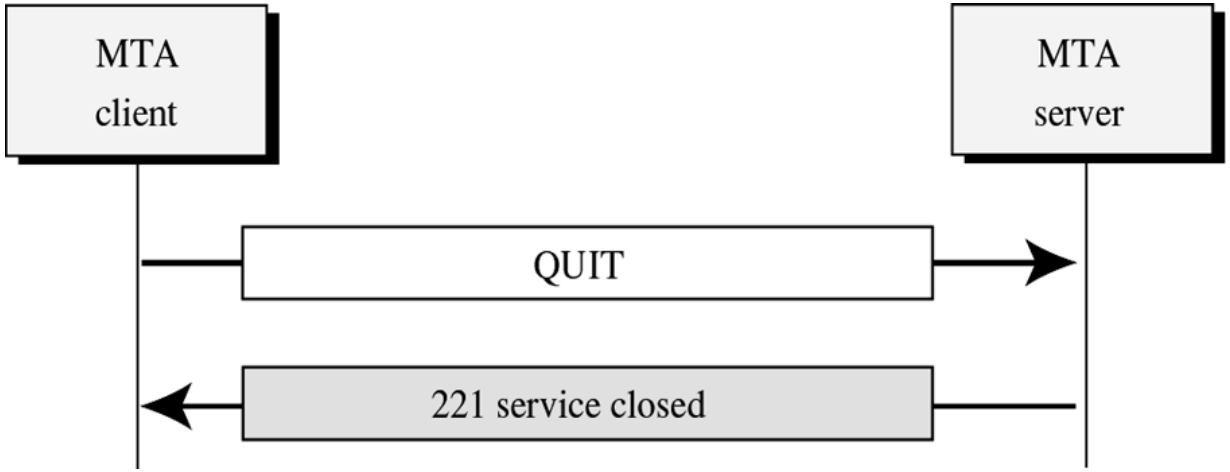


Mail Transfer

- Once a connection has been established, the SMTP sender may send one or more messages to the SMTP receiver.
- There are three logical phases to the transfer of a message:
  1. A **MAIL command identifies the originator** of the message.
  2. One or more **RCPT commands identify the recipients** for this message.
  3. A **DATA command transfers the message** text.

Connection Termination

- The SMTP sender closes the connection in two steps.



LIMITATIONS OF SMTP

- SMTP **cannot transmit executable files or other binary objects.**
- SMTP **cannot transmit text data that includes national language characters**, as these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
- SMTP servers **may reject mail message over a certain size.**
- SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, **resulting in translation problems.**

Common problems include the following:

- Deletion, addition, or recording of carriage return and linefeed.
- Truncating or wrapping lines longer than 76 characters.
- Removal of trailing white space (tab and space characters).
- Padding of lines in a message to the same length.
- Conversion of tab characters into multiple-space characters.

MULTIPURPOSE INTERNET MAIL EXTENSION (MIME)

MIME

- SMTP provides a basic email service, while MIME adds multimedia capability to SMTP.
- MIME is an **extension to SMTP** and is used **to overcome the problems and limitations of SMTP.**

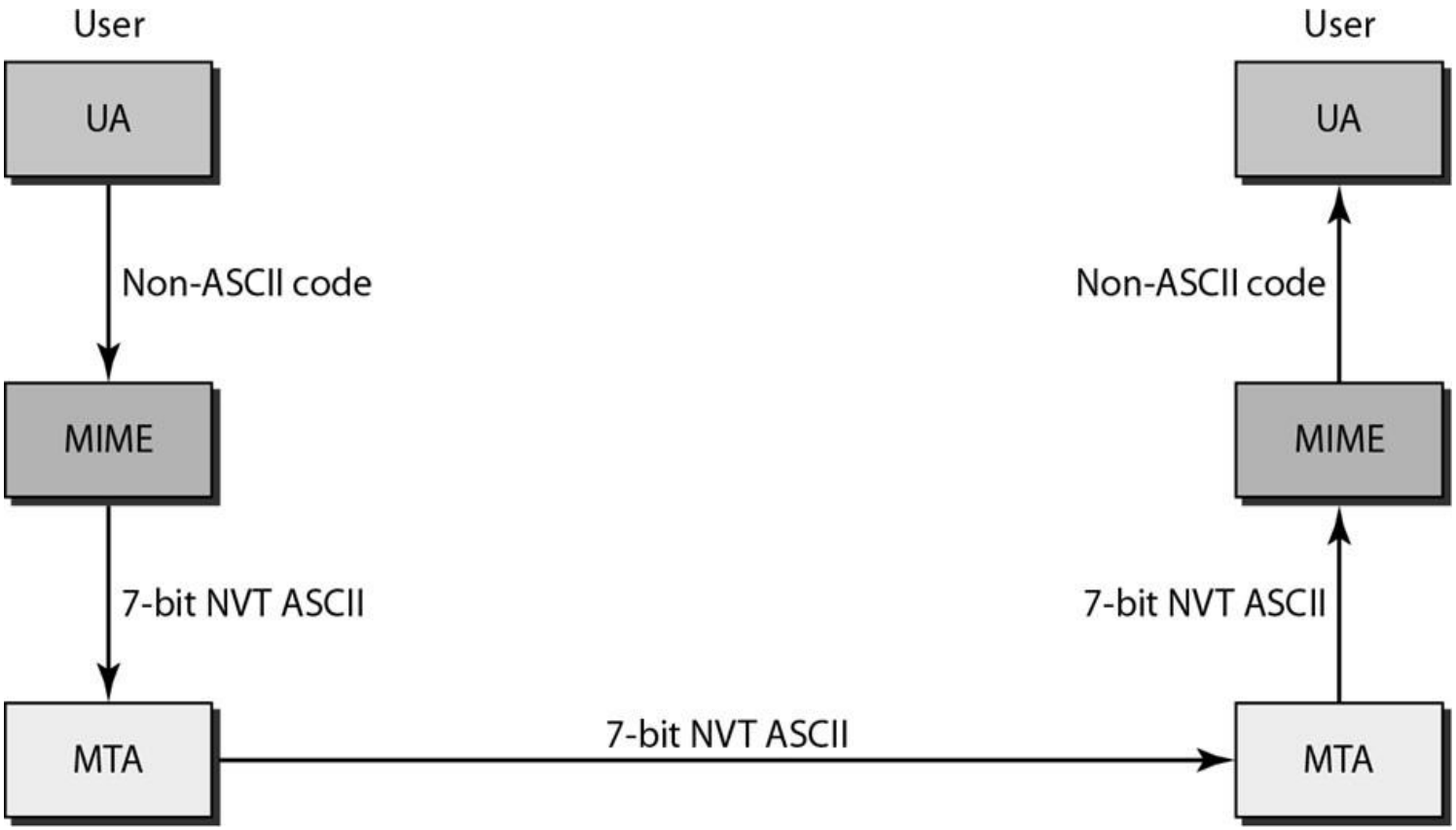
SMTP based Email system was designed **to send messages only in *ASCII* format.**

- Languages such as French, Chinese, other national languages are not supported.
- Image, audio and video files cannot be sent.

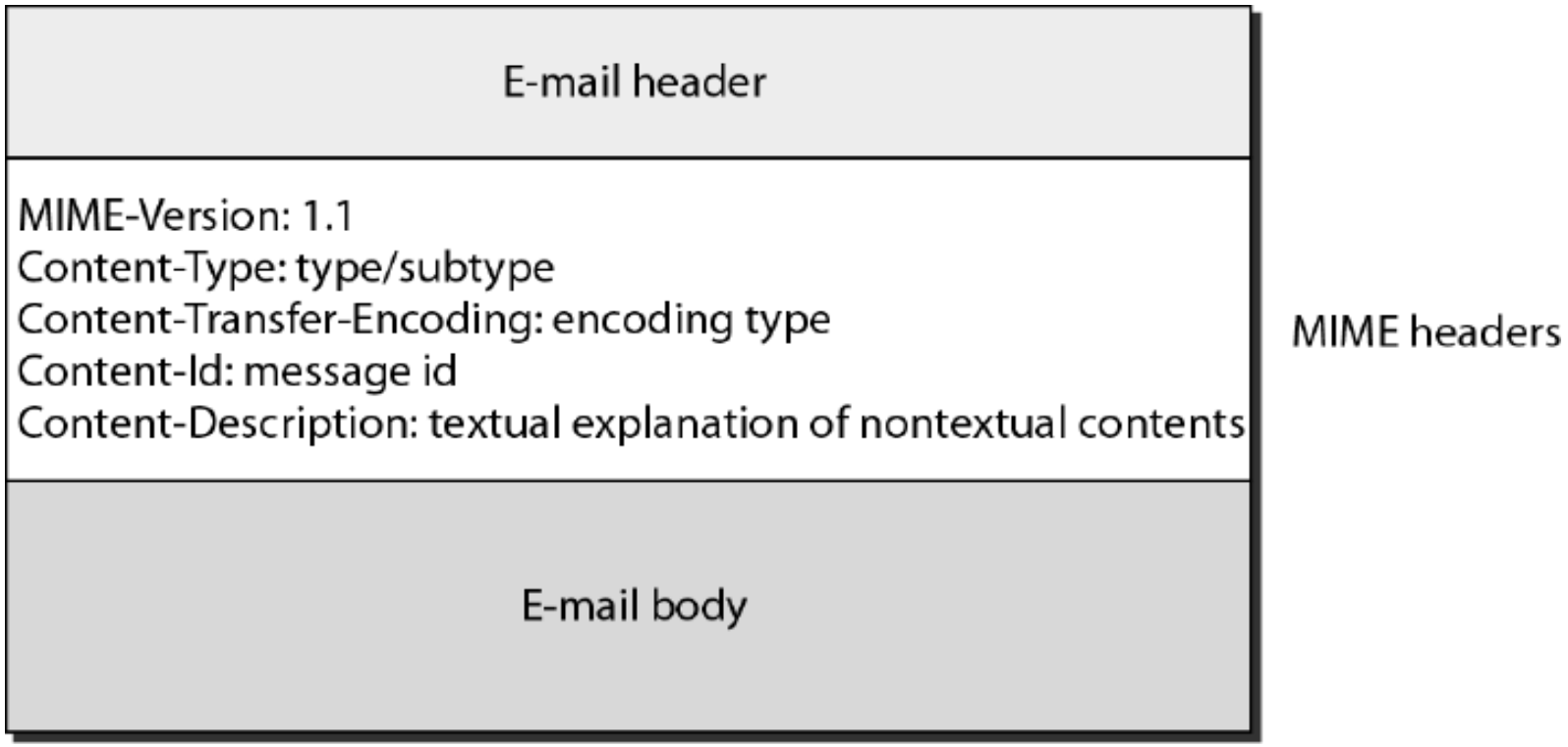
MIME adds the following features to email service:

- Be able to send **multiple attachments with a single message**;
- Unlimited message length;
- Use of **character sets other than ASCII code;**
- Use of rich text (layouts, fonts, colors, etc)
- **Binary attachments** (executables, images, audio or video files, etc.), which may be divided if needed.

- MIME is a protocol that ***converts*** **non-ASCII data to 7-bit NVT (Network Virtual Terminal) ASCII** and vice-versa



MIME HEADERS

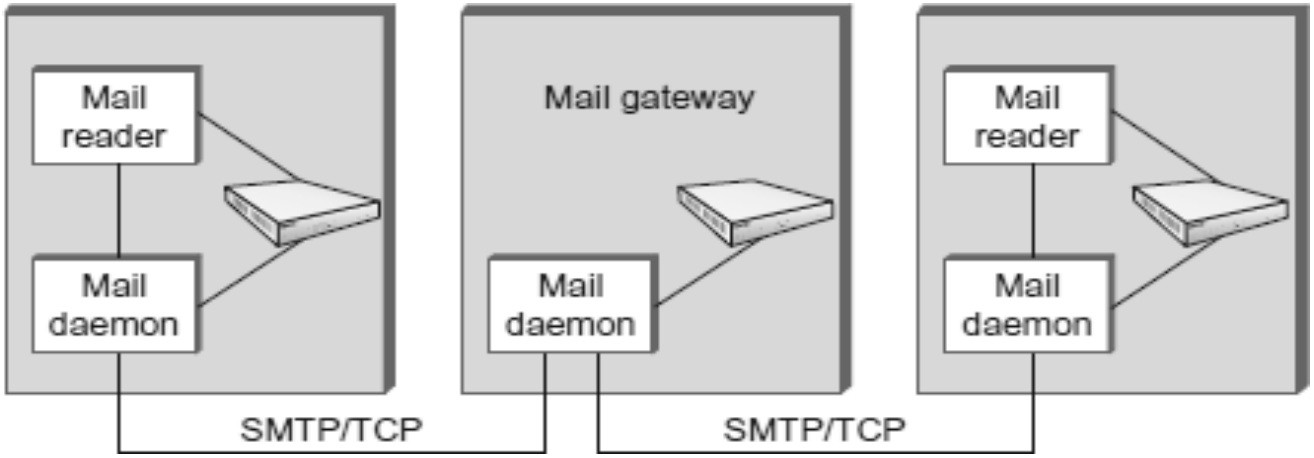


- MIME-Version- current version
- Content-Type - message type (text/html, image/jpeg, application/pdf)
- Content-Transfer-Encoding - message encoding scheme (eg base64).
- Content-Id - unique identifier for the message.
- Content-Description - describes type of the message body

MIME CONTENT TYPES

Type	*Subtype*	*Description*
Text	Plain	Unformatted
	HTML	HTML format
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Same as above, but no order
	Digest	Similar to mixed subtypes, but the default is message/ RFC822
	Alternative	Parts are different versions of the same message
Message	RFC822	Body is an encapsulated message
	Partial	Body is a fragment of a bigger message
	External-Body	Body is a reference to another message
Image	JPEG	Image is in JPEG format
	GIF	Image is in GIF format
Video	MPEG	Video is in MPEG format
Audio	Basic	Single-channel encoding of voice at 8 kHz
Application	PostScript	Adobe PostScript
	Octet-stream	General binary data (8-bit bytes)

MESSAGE TRANSFER IN MIME



- MTA is a mail daemon (sendmail) active on hosts having mailbox, used to send an email.
- Mail passes through a sequence of *gateways* before it reaches the recipient mail server.
- **Each gateway stores and forwards the mail** using SMTP.
- SMTP defines communication between MTAs over **TCP on port 25.**
- In an SMTP session, **sending MTA is *client* and receiver is *server***.
- Client posts a command (HELO, MAIL, RCPT, DATA, QUIT, VRFY, etc.)
- Server responds with a code (250, 550, 354, 221, 251 etc) and an explanation.
- Client is identified using HELO command and verified by the server
- Client forwards message to server, if server is willing to accept.
- **Eventually client terminates the connection.**

IMAP
(INTERNET MAIL ACCESS PROTOCOL)

- **IMAP** is an Application Layer Internet protocol that **allows an e-mail client to access e-mail on a remote mail server.**
- It is a **method of accessing electronic mail messages** that are kept on a possibly **shared mail server.**
- IMAP is a **more capable wire protocol (Point to Point)**
- IMAP is a **client / server protocol running over TCP on port 143**.
- IMAP **allows multiple clients simultaneously connected to the same mailbox,** and through flags stored on the server, **different clients accessing the same mailbox at the same** or different times can detect state changes made by other clients.
- In other words, it permits a **"client" email program to access remote message stores as if they were local.**

- For example, **email stored on an IMAP server can be manipulated from a desktop computer at home,** a workstation at the office, and a notebook computer while travelling, without the need to transfer messages or files back and forth between these computers.

IMAP can support email serving in three modes:

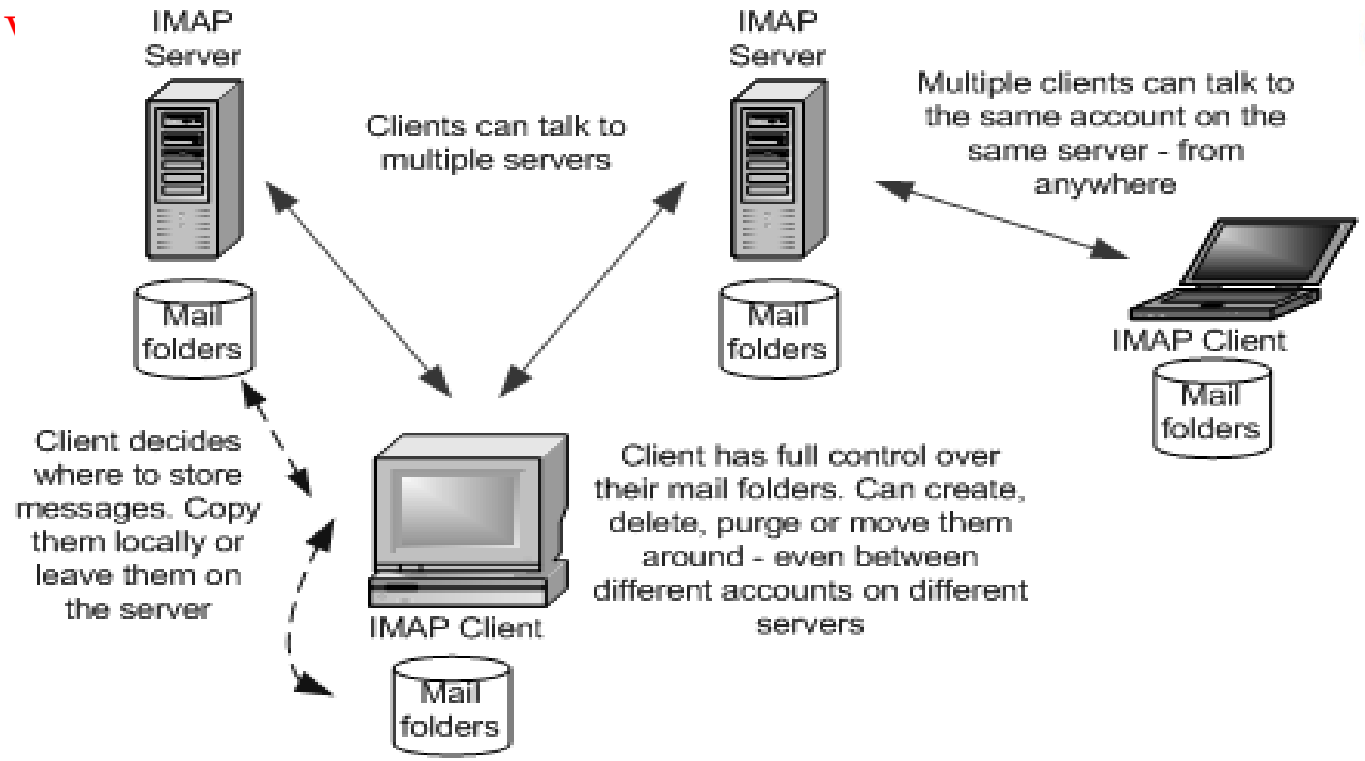
i) Offline

ii) Online: Users may connect to the server, look at what email is available, and access it online. This looks to the user very much like having local spool files, but they're on the mail server.

iii) Disconnected operation

➔ A mail client connects to the server, **can make a "cache" copy of selected messages, and disconnects from the server.**

➔ The user can then work on the messages offline, and **connect to the server later** and **resynchronize the server status with the cache.**

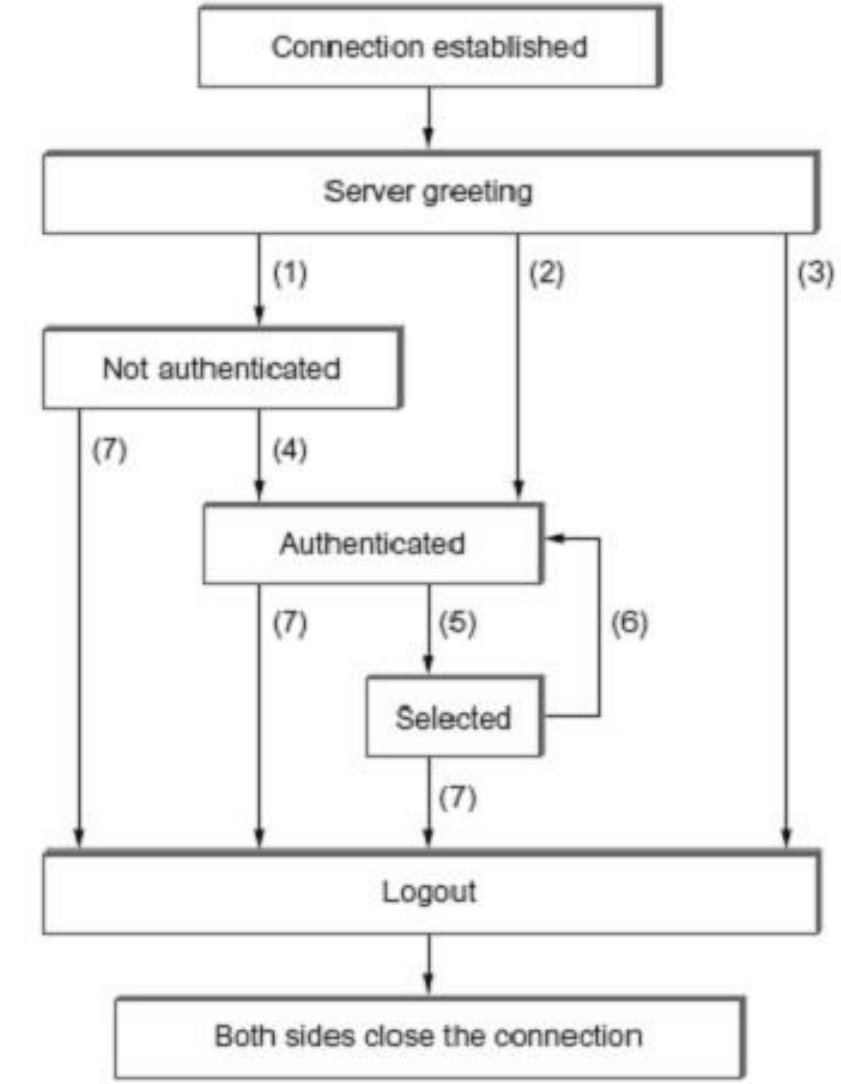


OPERATION OF IMAP

- The **mail transfer begins with the client authenticating the user** and identifying the mailbox they want to access.
- ***Client Commands***

  LOGIN, AUTHENTICATE, SELECT, EXAMINE, CLOSE, and LOGOUT
- ***Server Responses***

  OK, NO (no permission), BAD (incorrect command),
- When user wishes to FETCH a message, **server responds in MIME format.**
- Message *attributes* such as size are also exchanged.



(1) Connection without preauthentication (OK greeting)
(2) Preauthenticated connection (PREAUTH greeting)
(3) Rejected connection (BYE greeting)
(4) Successful LOGIN or AUTHENTICATE command
(5) Successful SELECT or EXAMINE command
(6) CLOSE command, or failed SELECT or EXAMINE command
(7) LOGOUT command, server shutdown, or connection closed

ADVANTAGES OF IMAP

- With IMAP, the **primary storage is on the server,** not on the local machine.
- Email being put away for storage **can be foldered on local disk**, or **can be foldered on the IMAP server**.
- The **protocol allows full user of remote folders**, including a remote folder hierarchy and multiple inboxes.
- **It keeps track of explicit status of messages**, and **allows for user-defined status.**

POST OFFICE PROTOCOL (POP3)

Post Office Protocol (POP3) is an **application-layer Internet standard protocol used by local e-mail clients to retrieve e-mail from a remote server** over a TCP/IP connection.

POST OFFICE PROTOCOL (POP3)

There are **two versions of POP.**

- The first, called ***POP2,*** **became a standard in the mid-80's** and requires SMTP to send messages.

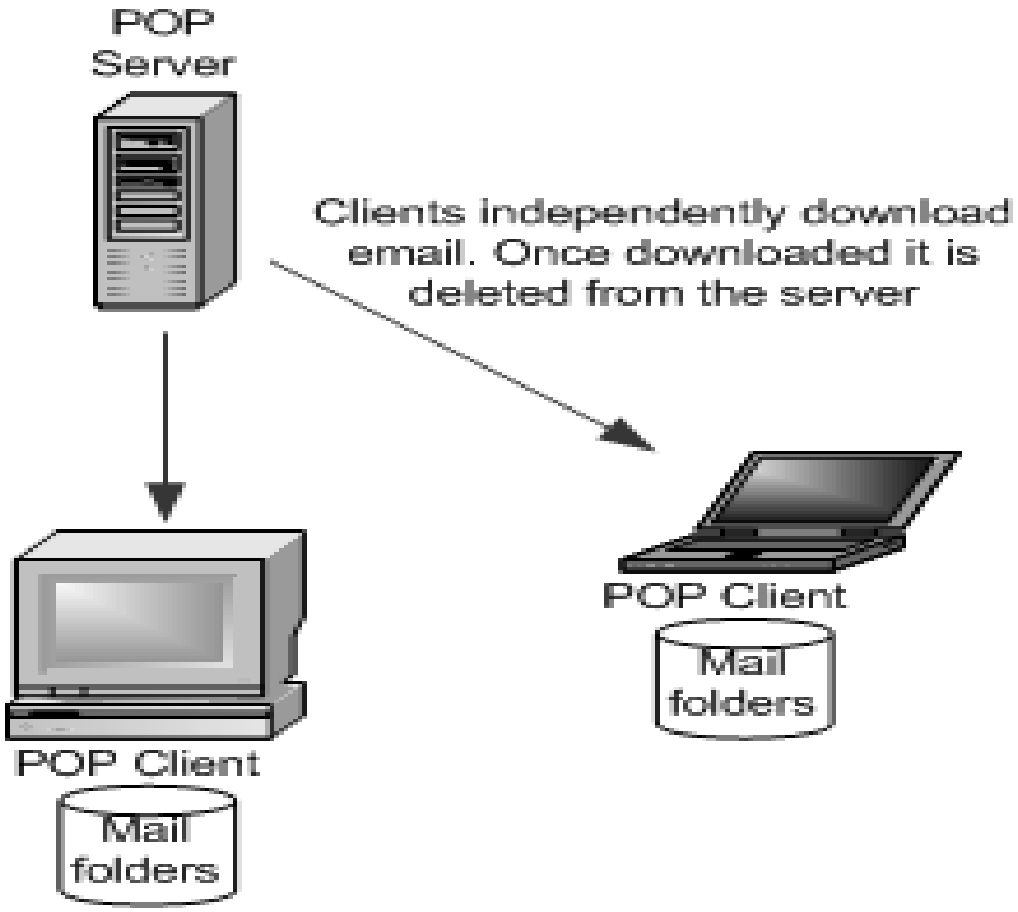
- The current version, **POP3, can be used with or without SMTP**.

- **POP3 uses TCP/IP port 110.**

  - POP is a much **simpler protocol**, making **implementation easier.**
  - POP **supports offline access to the messages,** thus **requires less internet usage time**
  - POP **does not allow search facility.**
  - In order **to access the messages, it is necessary to download them.**
  - It **allows only one mailbox to be created on server.**
  - It is not suitable for accessing non mail data.

- POP mail **moves the message from the email server onto the local computer**, although there is usually an option to leave the messages on the email server as well.
- POP **treats the mailbox as one store, and has no concept of folders.**
- POP works in **two modes namely**, ***delete*** **and** ***keep*** mode.

- In ***delete mode***, **mail is *deleted* from the mailbox after retrieval**. The delete mode is normally **used when the user is working at their permanent computer and can save and organize the received mail** after reading or replying.
- In ***keep mode***, **mail after reading is *kept* in mailbox** for later retrieval. The **keep mode is normally used when the user accesses her mail away** from their primary computer



- POP3 **client is *installed* on the recipient computer** and **POP server on the mail server.**
- **Client *opens* a connection to the server using TCP on port 110.**
- **Client sends username and password to *access* mailbox** and to retrieve messages

Advantages of IMAP over POP

- IMAP is **more powerful and more complex than POP**.
- User can ***check*** **the e-mail header prior to downloading.**
- User can ***search*** **e-mail for a specific string** of characters **prior to downloading.**
- User can **download *partially***, very useful in case of limited bandwidth.
- User can **create, delete, or rename *mailboxes* on the mail server**

TELNET (TERMINAL NETWORK)

TELNET (TERMINAL NETWORK)

- TELNET is the original **remote logging protocol, based on client-server program.**
- Telnet **provides a connection to the remote computer** in such a way that a local terminal appears to be at the remote side.
- **Network administrators** often **use TELNET for diagnostic and debugging purposes.**
- TELNET **requires a logging name and password.**
- It is **vulnerable to hacking** because **it sends all data including the password in plaintext (not encrypted).**
- A **hacker can eavesdrop and obtain the logging name and password**.
- **Because of this security issue**, the **use of TELNET has diminished.**

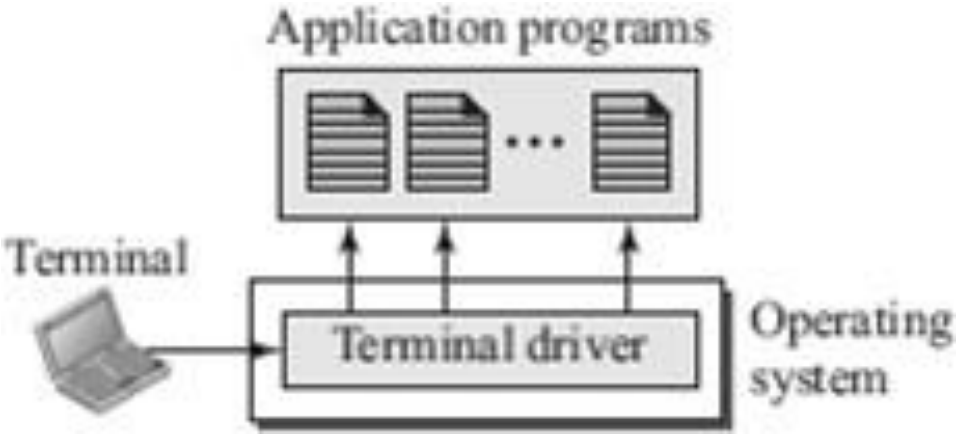
TYPES OF TELNET LOGGING:

→ There are two types of TELNET logging:

i) Local Logging and

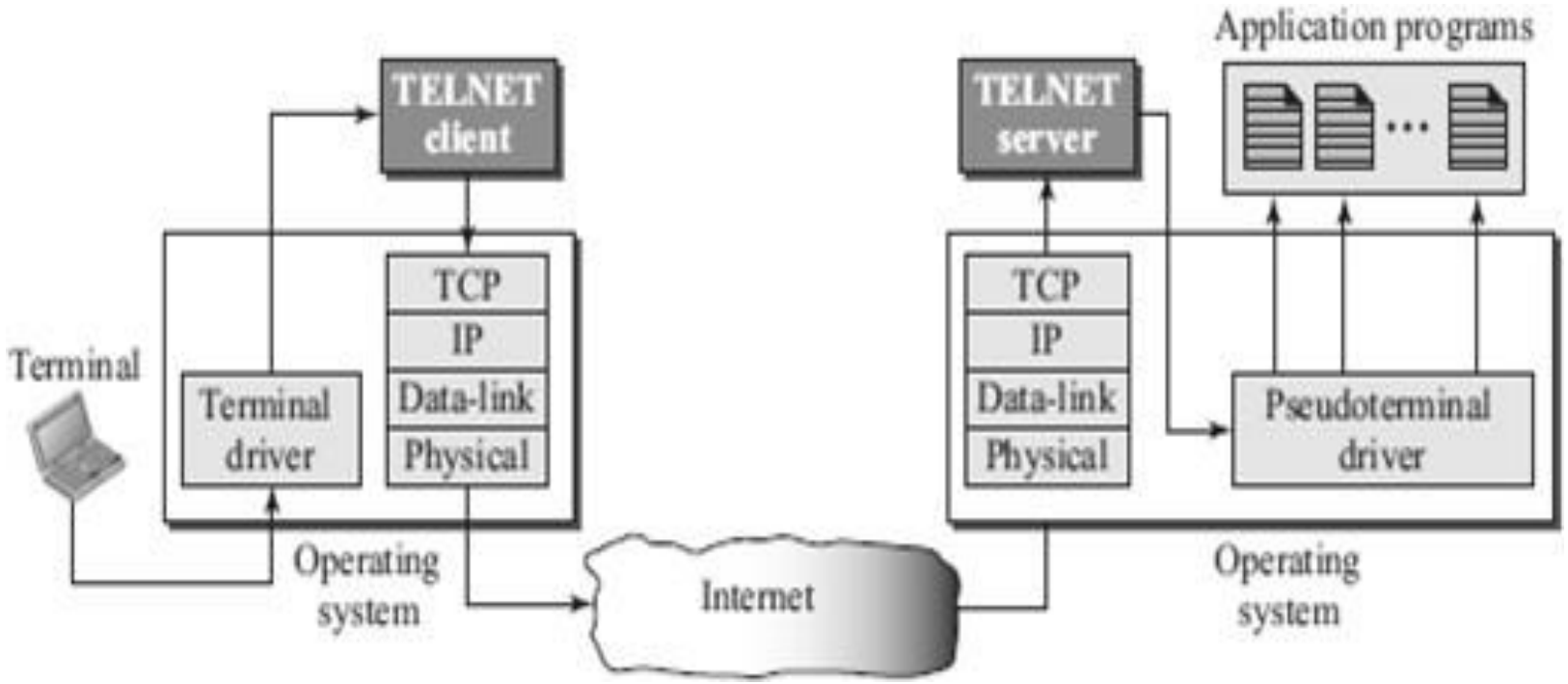
ii) Remote Logging

i) Local Login



- When a user logs into a local system, it is called local logging.
- As a **user types at a terminal** or at a workstation running a terminal emulator, the **keystrokes are accepted by the terminal driver.**
- The **terminal driver passes the characters to the operating system.**
- The operating system, in turn, **interprets the combination of characters and invokes the desired application program** or utility

ii) Remote Login



- When a **user wants to access an application program or utility located on a remote machine**, they perform remote logging.
- Remote Logging **uses TELNET client and TELENT server programs.**

- The **user sends the keystrokes to the terminal driver where the local operating system accepts the characters but does not interpret them.**

- The **characters are sent to the TELNET client**, **which transforms the characters into a universal character set** called **Network Virtual Terminal (NVT) characters** and **delivers them to the local TCP/IP stack.**

- The **commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine**.

- The **characters are delivered to the operating system and passed to the TELNET server**, **which changes the characters to the corresponding characters understandable by the remote computer.**

- The **characters cannot be passed directly to the operating system** because the **remote operating system is not designed to receive characters from a TELNET server**; **it is designed to receive characters from a terminal driver.**

NETWORK VIRTUAL TERMINAL (NVT)

- The **mechanism to access a remote computer is complex.**

- We are **dealing with heterogeneous systems.**

- This is because **every computer and its operating system accepts a special combination of characters** as tokens.

- **For example,** the **end-of-file token in a computer running the DOS operating system is Ctrl+z,** while the **UNIX operating system recognizes Ctrl+d.**

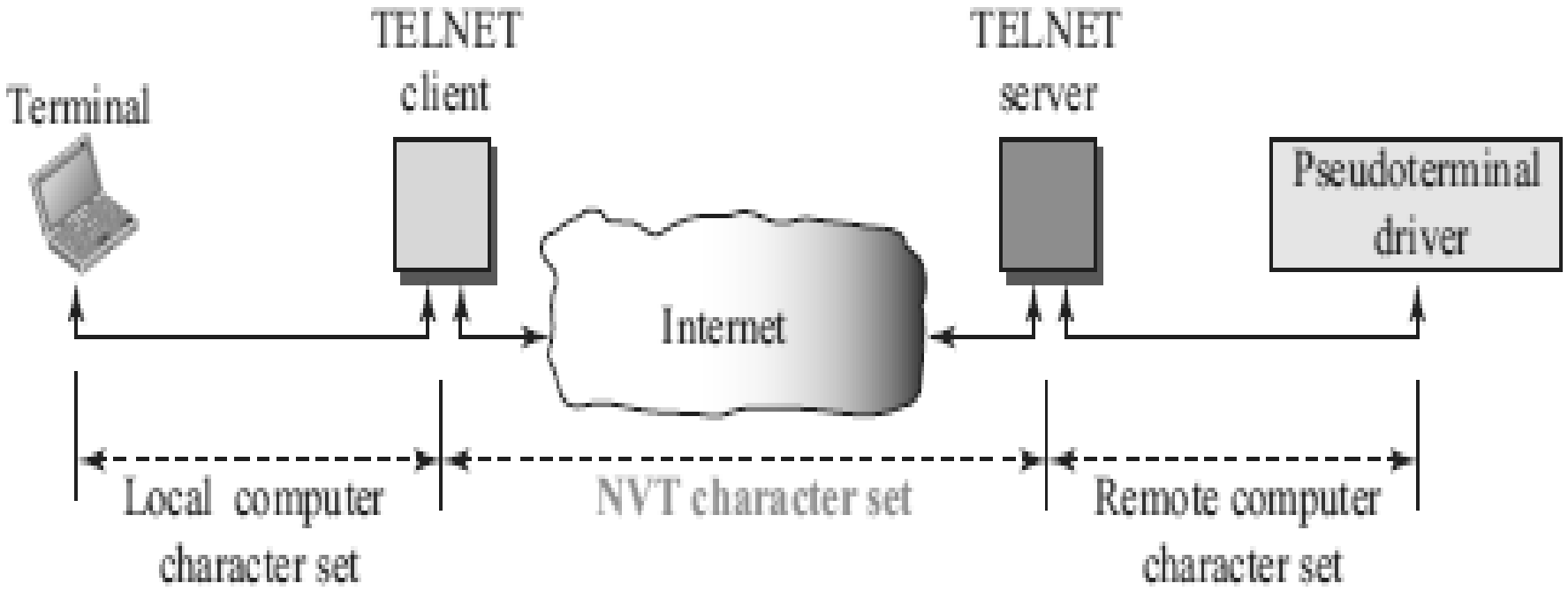
- If we want to access any remote computer in the world, we must first know what type of computer we will be connected to, and we must also install the specific terminal emulator used by that computer.

NETWORK VIRTUAL TERMINAL (NVT)

- **TELNET solves this problem by defining a universal interface called the Network Virtual Terminal (NVT) character set.**

- Via this interface, the **client TELNET translates characters** (data or commands) **that come from the local terminal into NVT form** and delivers them to the network.

- The **server TELNET,** on the other hand, **translates data and commands from NVT form into the form acceptable** by the remote computer

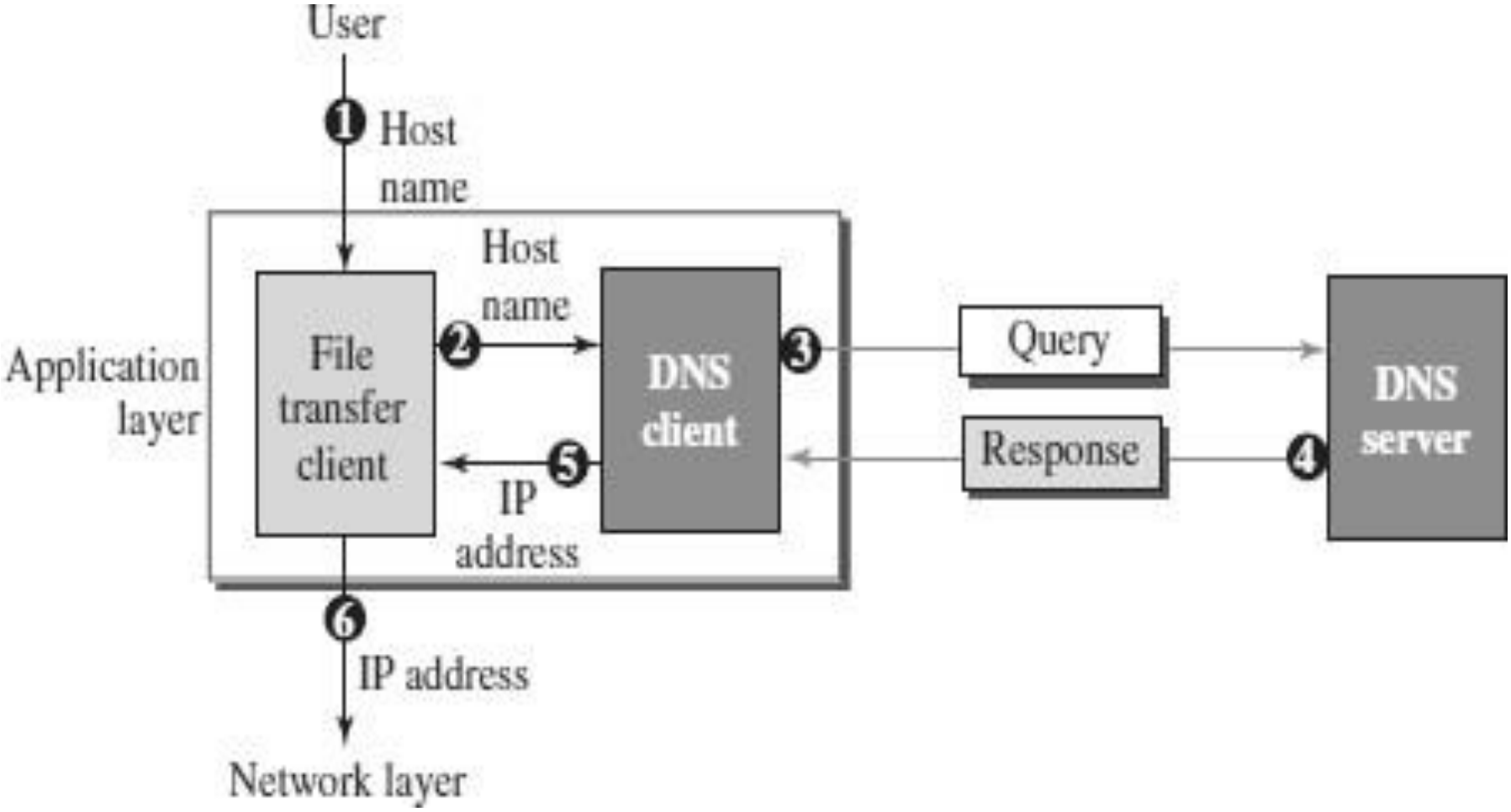

TELNET
client
TELNET
server
Terminal
Pseudoterminal
driver
Internet
Local computer
character set
NVT character set
Remote computer
character set

DNS (DOMAIN NAME SYSTEM)

DNS (DOMAIN NAME SYSTEM)

- Domain Name System was designed in 1984.
- DNS is used for **name-to-address mapping.**
- The DNS **provides the protocol which allows clients and servers to communicate** with each other.
- Eg: Host name like www.yahoo.com is translated into numerical IP addresses like **207.174.77.131**
- **Domain Name System (DNS) is a distributed database used by TCP/IP applications to map between hostnames and IP addresses** and to provide electronic mail routing information.
- **Each site maintains its own database of information** and runs a server program that other systems across the Internet can query.

WORKING PRINCIPLE OF DNS



The following **six steps shows the working of a DNS.** It maps the host name to an IP address:

1. The **user passes the host name to the file transfer client.**

2. The **file transfer client passes the host name to the DNS client.**

3. Each computer, **after being booted, knows the address of one DNS server.**

➔ The **DNS client sends a message to a DNS server with a query that gives the file transfer server name using the known IP address of the DNS server**.

1. The **DNS server responds with the IP address of the desired file transfer server.**

2. The **DNS server passes the IP address to the file transfer client**.

The file transfer client now uses the received IP address to access the file transfer server.

NAME SPACE

• **To be unambiguous, the names assigned to machines must be carefully selected from a name space** with complete control over the **binding between the names and IP address.**

- The **names must be unique because the addresses are unique.**

•A name space that maps each address to a unique name can be **organized in two ways:**

flat* (or) *hierarchical.

Flat Name Space

- In a flat name space, a **name is assigned to an address.**
- A name in this space is a **sequence of characters without structure.**
- The **main disadvantage** of a flat name space is that **it cannot be used in a large system such as Internet because it must be centrally controlled to avoid ambiguity and duplication**.

Hierarchical Name Space

- In a hierarchical name space, **each name is made of several parts**.

- The **first part can define the organization**,
  → **the second part can define the name**,
  → **the third part can define departments**, and so on.

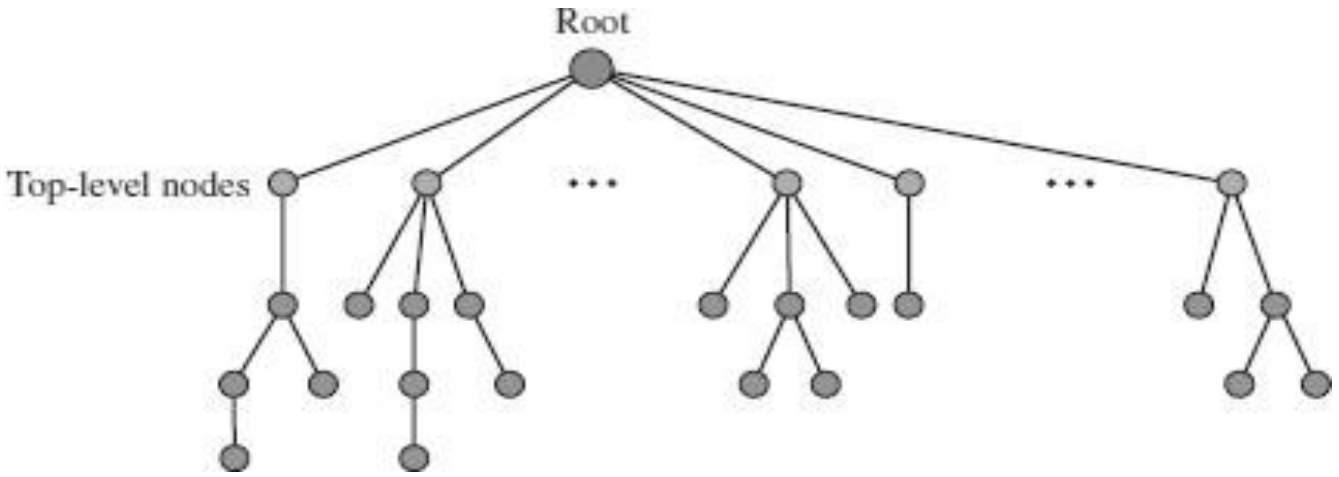
- In this case, **the authority to assign and control the name spaces can be decentralized.**

- A **central authority can assign the part of the name** that **defines the nature of the organization** and the name.

- The responsibility for the **rest of the name can be given to the organization itself. Suffixes can be added to the name to define host or resources.**

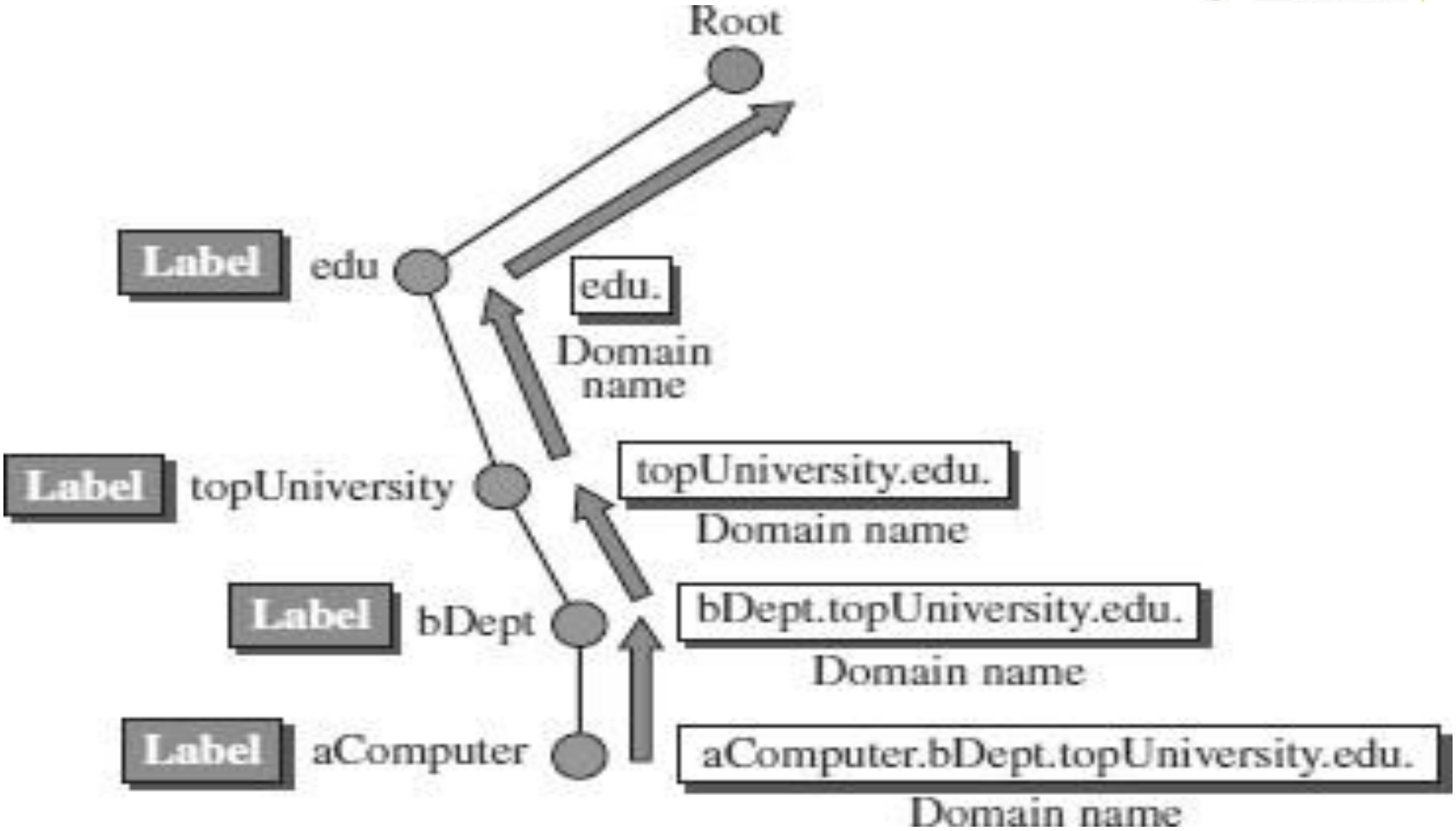
DOMAIN NAME SPACE

- To have a hierarchical name space, a domain name space was designed. In this design, the **names are defined in an inverted-tree structure with the root at the top.**
- **Each node in the tree has a label**, which is **a string with a maximum of 63 characters.**
- The **root label is a null string.**
- DNS requires that **children of a node have different labels**, which guarantees the uniqueness of the domain names.



Domain Name

- **Each node in the tree has a label called as domain name.**
- A full **domain name is a sequence of labels separated by dots (.)**
- The domain names are always **read from the node up to the root.**
- This means that a **full domain name always ends in a null label**, which means the **last character is a dot because the null string is nothing.**
- **If a label is terminated by a null string**, it is called a ***fully qualified domain name (FQDN)***
- If a label is not terminated by a null string, it is called a ***partially qualified domain name (PQDN)***


Root
Label
edu
edu.
Domain name
Label
topUniversity
topUniversity.edu.
Domain name
Label
bDept
bDept.topUniversity.edu.
Domain name
Label
aComputer
aComputer.bDept.topUniversity.edu.
Domain name