UNIT-IV

# What are Transport layer Primitives?

The primitives of the transport layer are

* 1. LISTEN, 2.CONNECT, 3.SEND, 4.RECEIVE , 5.DISCONNECT

# What are the different services of Transport Layer?

The two services of transport layer are

* + 1. Connection Oriented Service (TCP-Reliable)
    2. Connectionless Service (UDP-Unreliable)

# What is Congestion Control?

Congestion is a situation in Communication Networks in which too many packets are present in a part of the subnet, performance degrades. ***Congestion Control*** refers to techniques and mechanisms that can reduce the congestion.

# What is Multiplexing?

Multiplexing is transmitting multiple signals over a single communications line or computer channel. There are three types

* + - 1. Frequency Division Multiplexing(FDM).
      2. Time Division Multiplexing (TDM)
      3. Space Division Multiplexing(SDM)
      4. Code Division Multiplexing(CDM)

# What is piggybacking?

To achieve full duplex (bi-directional transmission) we use a concept called piggybacking.

It is a method to combine data frame with an ACK instead of sending two separate frames, thus ACK gets a free ride on the next out going data frame. It saves bandwidth.

# Explain about the transport layer protocols

The TCP/IP protocol uses a transport-layer protocol that is either a modification or a combination of some of these protocols.

## Connectionless protocol Simple Protocol

* **Simple Protocol**
  + First protocol is a simple connectionless protocol.
  + No flow control and error control with neither flow nor error control.
  + The receiver can never be overwhelmed with incoming packets.
  + Figure 1 shows the layout for this protocol.



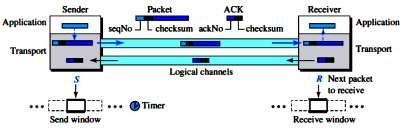
**Figure 1: Simple protocol**

**Connection-oriented protocol ** Stop-and-Wait protocol  Go-Back-N protocol

 Selective-Repeat Protocol  Piggybacking

## Stop-and-Wait Protocol

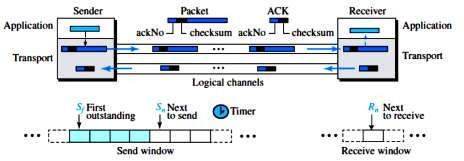
* + Second protocol is a connection-oriented protocol called the **Stop-and-Wait protocol.**
  + Provide flow and error control
  + Both the sender and the receiver use a sliding window of size 1.
  + The sender sends one packet at a time and waits for an acknowledgment before sending the next one.
  + To detect corrupted packets, we need to add a checksum to each data packet.
  + Figure 1 shows the outline for the Stop-and-Wait protocol.



**Figure 2: Stop-and-Wait protocol**

## Go-Back-*N* Protocol (GBN)

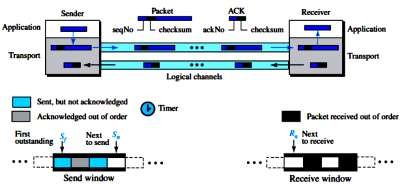
* + Efficient version of Stop-and-Wait protocol
  + The first is called ***Go-Back-N* (GBN)** (the rationale for the name will become clear later).
  + The key to Go-back-*N* is that we can send several packets before receiving acknowledgments, but the receiver can only buffer one packet.
  + Figure 3 shows the outline of the protocol.



## Selective-Repeat Protocol

**Figure 3: Go-Back-N protocol**

* + Suited to handle packet loss
  + The Go-Back-*N* protocol simplifies the process at the receiver.
  + The outline of this protocol is shown in Figure 4.

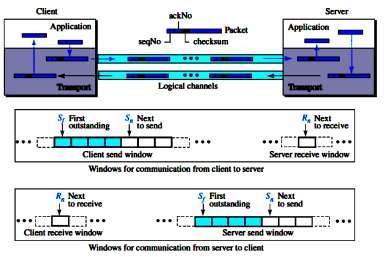


**Figure 4: Outline of Selective-Repeat**

## Bidirectional Protocols: Piggybacking

* + Data packets flow in only one direction and acknowledgments travel in the other direction.
  + Data packets are normally flowing in both directions:

 From client to server and from server to client. This means that acknowledgments also need to flow in both directions.

* + This technique called **piggybacking** is used to improve the efficiency of the bidirectional protocols.

**Figure 5: Design of piggybacking in Go-Back-N**

# Discuss about the User Datagram Protocol

* UDP – Unreliable connectionless transport protocol

## User Datagram

* UDP packets, called ***user datagrams,*** have a fixed-size header of 8 bytes made of four fields, each of 2 bytes (16 bits).
* Figure 1 shows the format of a user datagram.



**Figure 1: User datagram packet format**

* The first two fields define the source and destination port numbers.
* The third field defines the total length of the user datagram, header plus data.
* The 16 bits can define a total length of 0 to 65,535 bytes.

 Source Port – from 0 – 65,535 bytes

 Destination Port - from 0- 65,535 bytes

 UDP length- Total Length of the user datagram (header + data)

 Checksum- Detect errors over the entire datagram

## 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

As mentioned previously, UDP provides a *connectionless service.* This means that each user datagram sent by UDP is an independent datagram.

### Flow Control

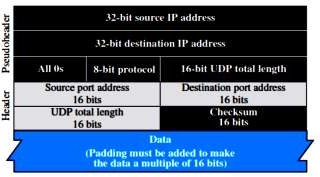
UDP is a very simple protocol. There is no *flow control,* and hence no window mechanism. The receiver may overflow with incoming messages.

### 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.

### Checksum

UDP checksum calculation includes three sections: a pseudoheader, the UDP header, and the data coming from the application layer (Figure 2).



**Figure 2: Pseudoheader for checksum calculation**

### Optional Inclusion of Checksum

The sender of a UDP packet can choose not to calculate the checksum.

### Congestion Control

Since UDP is a connectionless protocol, it does not provide congestion control.

### Encapsulation and Decapsulation

To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages.

## UDP Applications

* Simple request-response communication with little concernabout flow & error control
* In processes with internal flow & error control

– Trivial File Transport Protocol

* Multicasting
* In conjunction with Real Time Transport Protocol (RTP)for real-time data
* A process (application program)sends a chunk of bytes to UDP for delivery
* UDP adds its own header to this chunk of data (user datagram) and delivers it to IP
* UDP treats each chunk independently`

# Explain Transmission Control Protocol (TCP)

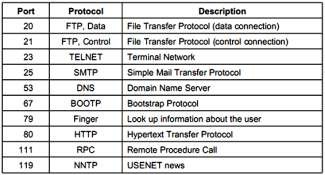
* **Transmission Control Protocol (TCP)** is a connection-oriented, reliable protocol.
* TCP explicitly defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service.
* TCP uses a combination of GBN and SR protocols to provide reliability.
* TCP is the most common transport-layer protocol in the Internet.

## TCP Services

### Process-to-Process Communication

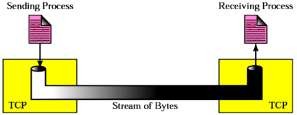
* + Some Assigned Port numbers (Table 1).

**Table 1: Port numbers**



### Stream Delivery Service

* + Byte-stream connection-oriented & reliable transport protocol
  + The sending process produces (writes to) the stream and the receiving process consumes (reads from) it (Figure 1).



**Figure 1: Stream delivery**

###  Sending and Receiving Buffers:

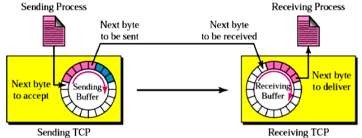
* + - Processes do not consume data at the same speed

 ***Sending site*:**

* + - * White section: empty locations to be filled by sending process
      * Blue section: bytes sent but not yet acknowledged
      * Red section: bytes to be sent by sending TCP

 ***Receiving site*:**

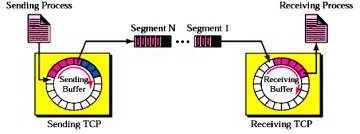
* + - * Section: empty locations to be filled by bytes from the networks
  + Figure 2 Shows how segments are created from the bytes in the buffers



###  Segments

**Figure 2: Sending and receiving buffers**

* + - TCP at the sending site gathers bytes into a packet called a segment (Figure 3).
    - TCP adds a header to each segment and delivers it to IP for transmission
    - Segments can arrive out of order
    - Size of the segment varies



### Full-Duplex Communication

**Figure 3: TCP segments**

* + 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

* + Like UDP, TCP performs multiplexing at the sender and demultiplexing at the receiver.

### Connection-Oriented Service

* + TCP, unlike UDP, is a connection-oriented protocol.
  + The following three phases occur:
    - * 1. The two TCP’s establish a logical connection between them.
        2. Data are exchanged in both directions.
        3. 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 Features

### Numbering System

TCP software keeps track of the segments being transmitted or received,

###  Byte Number

Numbering is used for flow & error control Segments are not numbered, only bytes Full-duplex connection – numbering is independent in each direction Numbers generated randomly from 0 to 232-1

 ***Sequence number*:**

* The number of the first byte carried in the segment

### Acknowledgement number:

* To confirm received bytes

**TCP numbering – an example**

* + Imagine a TCP connection is transferring a file of 5000bytes. Defines the number of the next byte the party expects to receive
  + The first byte is numbered 10001
  + What are the sequence numbers for each segment if data are sent in five segments, each carrying 1000 bytes?

**Solution:**

The following shows the sequence number for each segment:

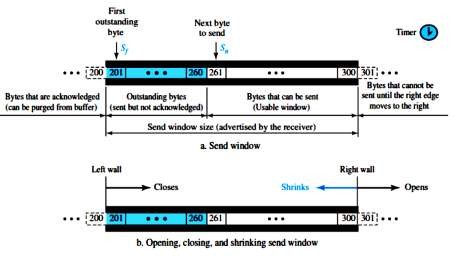


# Discuss about the windows in TCP

* TCP uses two windows (send window and receive window) for each direction of data transfer, which means four windows for a bidirectional communication.

### Send Window

* Figure 1 shows an example of a send window. The window size is 100 bytes. The figure shows how a send window *opens*, *closes*, or *shrinks*.



**Figure 1: Send window in TCP**

* + The send window in TCP is similar to the one used with the Selective-Repeat protocol, but with some differences:
    - * 1. One difference is the nature of entities related to the window.
        2. The second difference is that, in some implementations, TCP can store data received from the process and send them later, but we assume that the sending TCP is capable of sending segments of data as soon as it receives them from its process.
        3. Another difference is the number of timers.

### Receive Window

* Figure 2 shows an example of a receive window. The window size is 100 bytes.
* The figure also shows how they receive window opens and closes; in practice, the window should never shrink.

# Discuss about the flow control

* + Slow down the sender when the data is coming too fast for the receiver

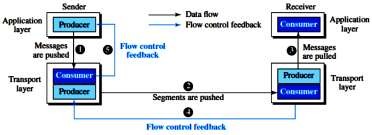
## Two approaches:

### Feedback-based flow control:

* + - * The receiver sends back permission to the sender.

### Rate-based flow control:

* + - * Built-in mechanism that limits senders’ rate. Rarely used in the data link layer.
  + Figure 1 shows unidirectional data transfer between a sender and a receiver; bidirectional data transfer can be deduced from the unidirectional process.



**Figure 1: Data flow and flow control feedbacks in TCP**

* + The figure shows that data travel from the sending process down to the sending TCP, from the sending TCP to the receiving TCP, and from the receiving TCP up to the receiving process (paths 1, 2, and 3).
  + Flow control feedbacks, however, are traveling from the receiving TCP to the sending TCP and from the sending TCP up to the sending process (paths 4 and 5).

### Opening and Closing Windows

* + To achieve flow control, TCP forces the sender and the receiver to adjust their window sizes, although the size of the buffer for both parties is fixed when the connection is established.
  + The opening, closing, and shrinking of the send window is controlled by the receiver.

### Shrinking of Windows

* + The receive window cannot shrink.
  + The send window, on the other hand, can shrink if the receiver defines a value for

*rwnd* that results in shrinking the window.

### Silly window syndrome

* + When either a sending application sends data slowly or a receiving application consumes data slowly
* Example: when 1 byte sent, 40 bytes overhead – not efficient

### Syndrome created by the sender

* **Nagle’s algorithm** to prevent TCP from sending data byte by byte

 Send the 1st byte

 Wait for either the received ACK or the maximum-size segment full

 Repeat step 2

### Syndrome created by the receiver

* Clark’s solution

Send ACK as a data arrive, but advertise 0 size window

* Delayed ACK

# Discuss about the error control

* + Detect and/or correct errors
  + Ensure that all frame are eventually delivered in order use acknowledgement, timer, and sequence number

### Checksum

* + Each segment includes a checksum field, which is used to check for a corrupted segment.
  + If a segment is corrupted, as detected by an invalid checksum, the segment is discarded by the destination TCP and is considered as lost.

### Acknowledgment

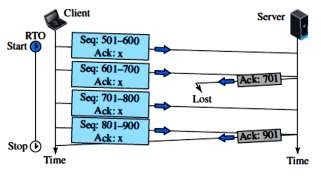
* + TCP uses acknowledgments to confirm the receipt of data segments.
  + ACK segments do not consume sequence numbers and are not acknowledged.

### Acknowledgment Type

* In the past, TCP used only one type of acknowledgment: cumulative acknowledgment.
* Today, some TCP implementations also use selective acknowledgment.

### Cumulative Acknowledgment (ACK)

* TCP was originally designed to acknowledge receipt of segments cumulatively.
* The receiver advertises the next byte it expects to receive, ignoring all segments received and stored out of order. This is sometimes referred to as *positive cumulative acknowledgment,* or ACK.`
* Figure 1 shows a lost acknowledgment sent by the receiver of data.



**Figure 1: Lost acknowledgment**

### Selective Acknowledgment (SACK)

* More and more implementations are adding another type of acknowledgment called

*selective acknowledgment,* or SACK.

### Retransmission

* The heart of the error control mechanism is the retransmission of segments.
* When a segment is sent, it is stored in a queue until it is acknowledged.
* When the retransmission timer expires or when the sender receives three duplicate ACKs for the first segment in the queue, that segment is retransmitted.

### Out-of-Order Segments

* TCP implementations today do not discard out-of-order segments. They store them temporarily and flag them as out-of-order segments until the missing segments arrive.

# Discuss about the TCP congestion control

* + TCP uses different policies to handle the congestion in the network.
  + Congestion occurs when the number of packets being transmitted through the network approaches the packet handling capacity of the network.

## Congestion Window

* + To control the number of segments to transmit, TCP uses another variable called a *congestion window, cwnd,* whose size is controlled by the congestion situation in the network (as we will explain shortly).
  + The *cwnd* variable and the *rwnd* variable together define the size of the send window in TCP.
  + The first is related to the congestion in the middle (network).
  + The second is related to the congestion at the end.
  + The actual size of the window is the minimum of these two.



## Congestion Detection

* + The TCP sender uses the occurrence of two events as signs of congestion in the network: time-out and receiving three duplicate ACKs.
  + The first is the *time-out*. If a TCP sender does not receive an ACK for a segment or a group of segments before the time-out occurs, it assumes that the corresponding segment or segments are lost and the loss is due to congestion.
  + Another event is the receiving of three duplicate ACKs (four ACKs with the same acknowledgment number).

## Congestion Policies

### Slow Start: Exponential Increase:`

* + The **slow-start algorithm** is based on the idea that the size of the congestion window (*cwnd*) starts with one maximum segment size (MSS), but it increases one MSS each time an acknowledgment arrives.
  + The name of this algorithm is misleading; the algorithm starts slowly, but grows exponentially. To show the idea, let us look at Figure 1.
  + When congestion window is below Threshold, sender in slow-start phase, window grows exponentially.

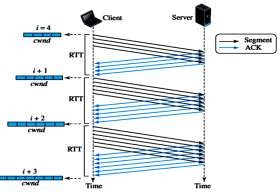


**Figure 1: Slow start, exponential increase**

### Congestion Avoidance: Additive Increase:

When congestion window is above Threshold, sender is in congestion- avoidance phase, window grows linearly.

A window is the number of segments transmitted during RTT. Figure 2 shows the idea.



## Fast Recovery

**Figure 2: Congestion avoidance, additive increase**

* + The **fast-recovery** algorithm is optional in TCP.
  + The old version of TCP did not use it, but the new versions try to use it.

## Additive Increase, Multiplicative Decrease

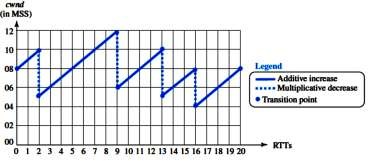
* + CongestionWindow (cwnd) is a variable held by the TCP source for each connection.



* + cwnd is set based on the perceived level of congestion. The Host receives *implicit*

(packet drop) or *explicit* (packet mark) indications of internal congestion.

* + The first is called *additive increase;* the second is called *multiplicative decrease*. This means that the congestion window size, after it passes the initial slow-start state, follows a saw tooth pattern called ***additive increase, multiplicative decrease (AIMD),*** as shown in Figure 3.



### Additive Increase

**Figure 3: Additive increase, multiplicative decrease (AIMD)**

* + - Additive Increase is a reaction to perceived available capacity.
    - Linear Increase basic idea:: For each “cwnd’s worth” of packets sent, increase cwnd by 1 packet.
    - In practice, cwnd is incremented fractionally for each arriving ACK.



### Multiplicative Decrease

* + - The key assumption is that a dropped packet and the resultant timeout are due to congestion at a router or a switch.

**Multiplicate Decrease:** TCP reacts to a timeout by **halving cwnd**.

* + - Although **cwnd** is defined in bytes, the literature often discusses congestion control in terms of packets (or more formally in MSS == Maximum Segment Size).
    - **cwnd** is not allowed below the size of a single packet.

## TCP Throughput

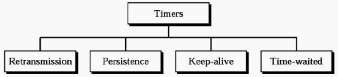
* + What’s the average throughout of TCP as a function of window size and RTT?

– Ignore slow start

* + Let W be the window size when loss occurs.
  + When window is W, throughput is W/RTT
  + Just after loss, window drops to W/2, throughput to W/2RTT.
  + Average throughout: .75 W/RTT

# Write short notes on TCP timers

To perform their operations smoothly, most TCP implementations use at least four timers: retransmission, persistence, keep alive, and TIME-WAIT (Figure 1).



## Retransmission timer

**Figure 1: TCP timers**

* + - * 1. If an ACK is received before the timer goes off – destroy the timer
        2. If the timer goes off before ACK arrives– retransmit the segment & reset the timer

 Retransmission time =2\* RTT (Round-Trip Time)

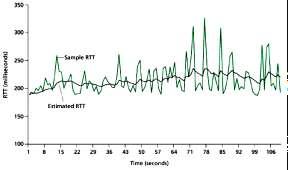
Not fixed since paths that IP packets take may differ

If too short – retransmissions waste of bandwidth

If too large – delay for the application program

RTT = α\* previous RTT + (1-α)\* current RTT, α usually 90 %

### RTT samples and estimates



* **Persistence timer**

**Figure 1: RTT samples and RTT estimates**

* To deal with the zero-size windows
* What if the receiver advertises that the window size is ≠ 0 (by sending ACK) and this ACK is lost?
  + ACK are not acknowledged in TCP
* Start persistence timer
  + When this goes off send a probe (1 byte of data)
  + It is set to the retransmission time &doubled every time a response is not received (until 60s, then sent every 60s)

## Keep alive timer

* To prevent a long idle connection between a client and a server

– Either client or server crash

* Usually set to 2h

## Time-Waited Timer

* Used during connection termination to allow duplicate FIN segments to be discarded at the destination
* Usually 2 times the expected lifetime of a segment.

UNIT-V

Application Layers: Client Server Programming-Word Wide Web&HTTP–FTP-Email–DNS

# What is WWW?

The World Wide Web (WWW) is a repository of information linked together from points all over the world.

# What is browser?

A web browser is a ***software application*** for ***retrieving, presenting, and passing over information resources on the World Wide Web.*** An information resource is identified by a Uniform Resource Identifier (URI) and may be a web page, image, video, or other piece of content.

Or

It is a software application to accesses information provided by Web servers.

# Define electronic mail system (Email)?

Electronic mail defined as a system that allows people to send messages to each other by computer electronically.

# Define domain name system

The Domain Name System is a distributed database with hierarchal structure and serve the basis for name resolution process in TCP/IP network.

# What is Berkeley Socket?

**Berkeley sockets** is an application programming interface (API) for Internet **sockets** and Unix domain **sockets**, used for inter-process communication.

# What are TCP & UDP?

TCP means Transmission Control Protocol. It is a connection-oriented protocol allows packets from one machine to another machine without errors.

UDP means User Dataram Protocol. It is unreliable connectionless protocol for applications that do not want TCP.

# What is DNS?

The Domain Name System (**DNS**) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network.

# What are Static and Dynamic documents?

Static Documents are designed HTML which keep information unchanged in the web page.

Dynamic Documents are designed by XML ,XHTML which keep information changed dynamically.

# Define HTTP

HTTP means Hyper Text Transfer Protocol. It is the protocol used to access information from the Internet Server.

# Define FTP

File Transfer Protocol (FTP) is a TCP/IP client-server application for copying files from one host to another. FTP requires two connections for data transfer: a control connection and a data connection.

# Briefly explain the Client Server Programming

**Client-server** is a network architecture which separates the client from the server.

 Each client software can send requests to a server.

 The server handles the request received from a client, prepares a result, and sends the result back to the client.

## Application Programming Interface

 A computer language has a set of instructions for mathematical operations, a set of instructions for string manipulation, a set of instructions for input/output access, and so on.

 A new set of instructions to tell the lowest four layers of the TCP/IP suite to open the connection.

 Send and receive data from the other end, and close the connection.

 A set of instructions of this kind is normally referred to as an **application programming interface (API)**.

 Several APIs have been designed for communication.

 Three among them are common: s**ocket interface, Transport Layer Interface**

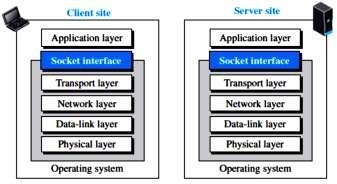
## (TLI), and STREAM.

 Socket interface started in the early 1980s at UC Berkeley as part of a UNIX environment.

 The socket interface is a set of instructions that provide communication between the application layer and the operating system, as shown in Figure 1.

 The idea of sockets allows us to use the set of all instructions already designed in a programming language for other sources and sinks.

**For example,** In most computer languages, like C, C++, or Java.



**Figure 1:Position of the socket interface**

## (ii).Using Services of the Transport Layer

There are three common transport-layer protocols in the TCP/IP suite: UDP, TCP, and SCTP.

### UDP Protocol

* + It is Connectionless protocol i.e. no need to establish & terminate connection.
  + UDP delivers independent messages, called datagrams between applications or processes on host computers - end-to-end protocol.
  + Message-oriented: An application using UDP sends and receives individual message. UDP does not divide a message into packets. Message must fit datagram.

***TCP Protocol***

* + TCP provides connection-oriented, reliable, byte-stream service.
  + TCP requires that two ends first create a logical connection between themselves by exchanging some connection-establishment packets.
  + The two ends can send chunks of data in segments in each direction.

### SCTP Protocol

* + SCTP provides a service which is a combination of the two other protocols.
    - TCP, SCTP provides a connection-oriented, reliable service, but it is not byte- stream oriented.
    - It is a message-oriented protocol like UDP.

## Iterative Communication Using UDP

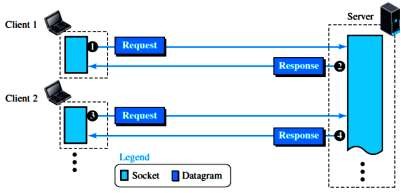
 Communication between a client program and a server program can occur iteratively or concurrently.

 Several client programs can access the same server program at the same time, the server program can be designed to respond iteratively or concurrently.

 An iterative server can process one client request at a time; it receives a request, processes it, and sends the response to the requestor before handling another request.

 The received and queued requests are handled in the first-in, first-out fashion.

 In UDP communication, the client and server use only one socket each. The socket created at the server site lasts forever; the socket created at the client site is closed (destroyed) when the client process terminates.

 Figure 2 shows the lifetime of the sockets in the server and client processes.

**Figure 2: Sockets for UDP communication**

## Iterative Communication Using TCP

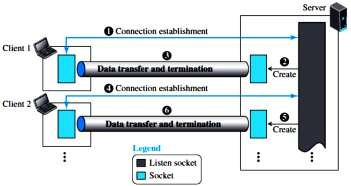
 TCP is a connection-oriented protocol.

 Before sending or receiving data, a connection needs to be established between the client and the server.

 After the connection is established, the two parties can send and receive chunks of data as long as they have data to do so.

 Iterative communication using TCP is not very common.

 The TCP server uses two different sockets, one for connection establishment and the other for data transfer (Figure 3).



**Figure 2: Sockets used in TCP communication**

## (v).Concurrent Communication

 A concurrent server can process several client requests at the same time.

 This can be done using the available provisions in the underlying programming language.

 In C, a server can create several child processes, in which a child can handle a client.

 In Java, threading allows several clients to be handled by each thread.

# Describe the client side and the server side in World Wide Web& HTTP

The World Wide Web (abbreviated WWW or Web) consists of a world wide collection of electronic documents.

The HyperText Transfer Protocol (HTTP),the most common client server application program used in relation to the Web.

## World Wide Web

 The idea of the Web was first proposed by Tim Berners-Lee in 1989 at *CERN*,

 The commercial Web started in the early 1990s.

### Web

Each electronic document on the web is called a web page, which can contain text, graphics, audio, and video.

* + Web Pages usually built0in connections to other documents.
  + Some web pages are static (Fixed), others are dynamic (changing).

**Static Web Page:** Visitors to a Static Web page all see the same content.

**Dynamic Web Page:** Visitor can customize some or all the viewed contents. Such as desired stock quotes, ticket availability for flights.

### Website

Website is a collection of related web pages and associated items, such as documents and pictures, stored on a web server.

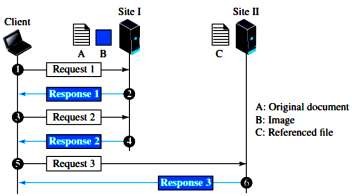
* + A website may be used for advertisement, advocacy, and news etc.
  + **Home Page:** The first page of the website display

### Architecture

The WWW today is a distributed client-server service, in which a client using a browser can access a service using a server.

* + The service provided is distributed over many locations called *sites.*
  + Each site holds one or more web pages.

**Example 1:** Figure 1 shows the situation.



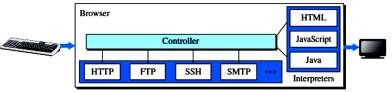
### Web Client (Browser)

**Figure 1: Example 1**

A Web Browser (or Browser) is application software that allows the users to access and view web pages.

The more widely used web browsers for personal computers are: Internet Explorer, Fire Fox, Opera, Netscape, and Safari.

Each browser usually consists of three parts: a controller, client protocols, and interpreters. (Figure 2).



**Figure 2: Browser**

**Downloading:** Downloading is a process of computer receiving information, such as web page, from a server on the internet.

**Connecting to a Web Browser: Step 1:** Connect to the internet. **Step 2:** Open the Web Browser. **Step 3:** Type the web URL.

**Step 4:** Connection to the server occurs and a home page is displayed.

### Web Server

A Web Server is a computer that delivers requested web pages to your computer.

* + The same web server can store multiple websites.

### Uniform Resource Locator (URL)

web page has a unique address, called a URL (Uniform Resource Locator).

* + Every Web Site has a unique address called a URL.
  + We need to enter the web address correctly in your web browser otherwise it would not open the desired page for you.
  + The entered web address will deliver the requested web pages from the computer called the web servers.
  + **Web Servers:** The computers that hold the web pages are called the web server.

There are three identifiers: ***Host, Port,*** and ***Path*.** we need to tell the browser what client server application we want to use, which is called the ***protocol*.**

To combine these four pieces together, the **uniform resource locator (URL)** has been designed; it uses three different separators between the four pieces as shown below:



## Example 2:

The URL [***http://www.mhhe.com/compsci/forouzan/***](http://www.mhhe.com/compsci/forouzan/)

***Web Documents***

The documents in the WWW can be grouped into three broad categories:

 Static,

 Dynamic,

 Active.

### Static Documents

**Static documents** are fixed-content documents that are created and stored in a server.

### Dynamic Documents

A **dynamic document** is created by a web server whenever a browser requests the document.

### Active Documents

For many applications, we need a program or a script to be run at the client site. These are called ***active documents*.**

## Hyper Text Transfer Protocol (HTTP)

* HTTP implemented in 2 programs
  + Client
  + Server
* HTTP defines how client and server talk to each other by exchanging messages.

### Non-persistent versus Persistent Connections

* ***Non-persistent Connections***
  + In a **nonpersistent connection**, one TCP connection is made for each request/response.

The following lists the steps in this strategy:

* 1. The client opens a TCP connection and sends a request.
  2. The server sends the response and closes the connection.
  3. The client reads the data until it encounters an end-of-file marker; it then closes the connection.

### Persistent Connections

* + Two Versions of Persistent connections
    - With Pipelining
    - Without Pipelining
  + The connection will be closed by the server after some time.

# Explain the File Transfer Protocol

 **File Transfer Protocol (FTP)** is the standard protocol provided by TCP/IP for copying a file from one host to another.

 For example

* Two systems may use different file name conventions.
* Two systems may have different ways to represent data.
* Two systems may have different directory structures.

 All of these problems have been solved by FTP in a very simple and elegant approach.

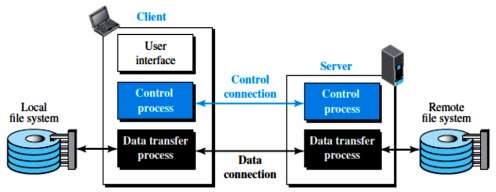
 Figure 1 shows the basic model of FTP.

 The client has three components:

* User interface,
* Client control process,
* Client data transfer process.

 The server has two components:

* Server control process
* Server data transfer process.



### Two Connections

**Figure 1: FTP**

 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

 FTP uses two well-known TCP ports: port 21 is used for the control connection, and port 20 is used for the data connection.

### Control Connection

 For control communication, FTP uses the same approach as TELNET (discussed later).

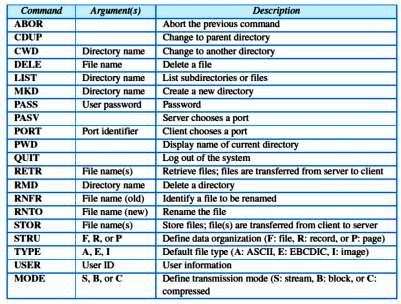
 It uses the NVT ASCII character set as used by TELNET.

 Communication is achieved through commands and responses.

 Each line is terminated with a two-character (carriage return and line feed) end-of-line token. During this control connection, commands are sent from the client to the server and responses are sent from the server to the client.

 Some of the most common commands are shown in Table 1.

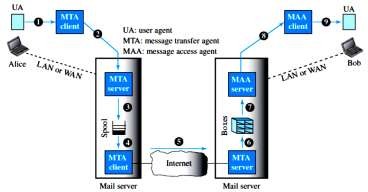
**Table 1: Some FTP commands**



# Elaborate note on Electronic Mail System (Email).

E-mail (short for electronic mail) is the transmission of messages and files via a computer network.

## Architecture

To explain the architecture of e-mail, we give a common scenario, as shown in Figure1. Another possibility is the case in which Alice or Bob is directly connected to the corresponding mail server, in which LAN or WAN connection is not required.

**Figure 1: Common scenario**

* + It normally consist of the following two agents. They are U**ser Agent (UA)**

It allows people to read and send e-mail. The user agents are local programs that provide a command-based, menu-based, or graphical method for interacting with the e-mail system.

##  Message transfer agent

 **Message access agent**

### Message Transfer Agent (MTA):

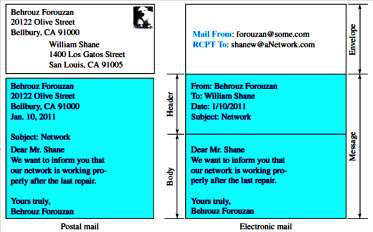
M**essage Transfer Agents**, which move the messages from the source to the destination.

### Message Access Agent (MAA):

The direction of the bulk data (messages) is from the client to the server and server to the client.

### Sending mail

* + - To send an e-mail message, a user must provide the message, the destination address, and possibly some other parameters.
    - To send mail, the user, through the UA, creates mail that looks very similar to postal mail. It has an *envelope* and a *message* (Figure 2).



**Figure 2: Format of an e-mail**

### Receiving Mail

* + - The user agent is triggered by the user (or a timer). If a user has mail, the UA informs the user with a notice. If the user is ready to read the mail, a list is displayed in which each line contains a summary of the information about a particular message in the mailbox.
    - The summary usually includes the sender mail address, the subject, and the time the mail was sent or received.
    - The user can select any of the messages and display its contents on the screen.

### Addresses

* + - To deliver mail, a mail handling system must use an addressing system with unique addresses.
    - In the Internet, the address consists of two parts: a *local part* and a *domain name,*

separated by an @ sign (Figure 3).



**Figure 3: E-mail address**

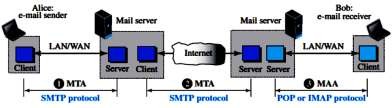
### Mailing List or Group List

* + - Electronic mail allows one name, an *alias,* to represent several different e-mail addresses; this is called a mailing list.

## Message Transfer Agent: SMTP

* + Based on the common scenario (Figure 1), we can say that the e-mail is one of those applications that needs three uses of client-server paradigms to accomplish its task.
  + It is important that we distinguish these three when we are dealing with e-mail (Figure 4).
  + shows these three client-server applications. We refer to the first and the second as Message Transfer Agents (MTAs), the third as Message Access Agent (MAA).
  + The formal protocol that defines the MTA client and server in the Internet is called

### Simple Mail Transfer Protocol (SMTP).

* + SMTP is used two times, between the sender and the sender’s mail server and between the two mail servers.

**Figure 4:Protocols used in electronic mail**

### Commands and Responses

SMTP uses commands and responses to transfer messages between an MTA client and an MTA server.

### Commands

Commands are sent from the client to the server. The format of a command is shown below:

**Keyword:** argument(s)

### Responses

Responses are sent from the server to the client. A response is a three digit code that may be followed by additional textual information.

## (ii).Web-Based Mail

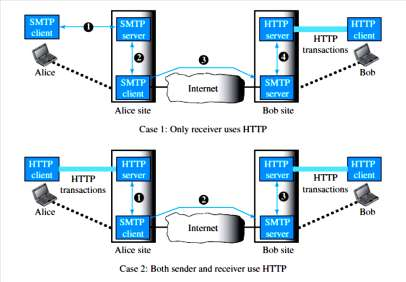
* + E-mail is such a common application that some websites today provide this service to anyone who accesses the site.
  + Three common sites are Hotmail, Yahoo, and Google mail.
  + The idea is very simple. Figure 5 shows two cases:

### Case I

In the first case, Alice, the sender, uses a traditional mail server; Bob, the receiver, has an account on a web-based server.

### Case II

In the second case, both Alice and Bob use web servers, but not necessarily the same server.



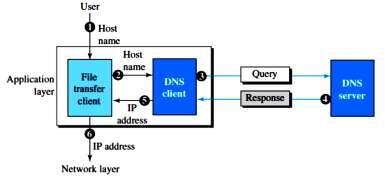
## E-Mail Security

**Figure 5: Web-based e-mail, cases I and II**

* + Secure/Multipurpose Internet Mail Extension- Secure enhancement to MIME- Internet e-mail format standard.
  + S/MIME will probably emerge as the industry standard for commercial and organizational use.
  + PGP (Pretty Good Privacy) use for personal e-mail security.

# Explain Domain Name System with example.

* **Domain Name System** (or Service or Server), an Internet service that translates **domain names** into IP addresses.
* Figure 1 shows how TCP/IP uses a DNS client and a DNS server to map a name to an address.
* The user knows only the file transfer server name, such as *afilesource.com.*
* The TCP/IP suite needs the IP address of the file transfer server to make the connection.



**Figure 1: Purpose of DNS**

The following six steps map 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.
  4. The DNS server responds with the IP address of the desired file transfer server.
  5. The DNS server passes the IP address to the file transfer client.
  6. The file transfer client now uses the received IP address to access the file transfer server.

### Name Space

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

ways:

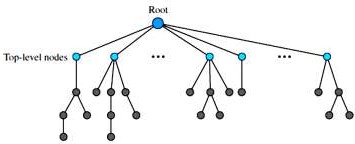
Flat or hierarchical.

 In a *flat name space,* a name is assigned to an address.

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

### 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.
* The tree can have only 128 levels: level 0 (root) to level 127 (Figure 2).



### Label

**Figure 2: Domain name space**

 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 (empty string).

### Domain Name

 Each node in the tree has a domain name. A full **domain name** is a sequence of labels separated by dots (**.**).

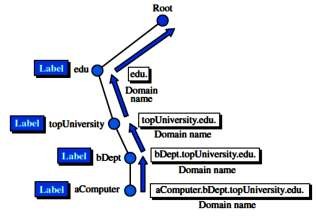
 If a label is terminated by a null string, it is called a **fully qualified domain name (FQDN)**.

 If a label is not terminated by a null string, it is called a **partially qualified domain name (PQDN).**

### Distribution of Name Space

The information contained in the domain name space must be stored (Figure 3).

* The original generic domains are *com (commercial), edu* (educational institutions), *gov*, *int* (certain international organizations), *mil* (the U.S. armed forces), *net* (network providers), and *org* (nonprofit organizations).



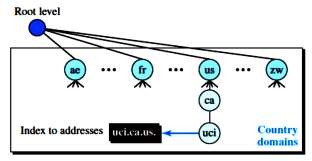
**Figure 3 Domain names and labels**

* Domain names are case insensitive, so *edu*, *Edu*, and *EDU* mean the same thing. Component names can be up to 63 characters long, and full path names must not exceed 255 characters.
* Once a new domain has been created and registered, it can create sub domains, such as *cs.unsd.edu*, without getting permission from anybody higher up the tree.
* Each domain is named by the path upward from it to the (unnamed) root. The components are separated by periods (pronounced ''dot'').

### Country Domains

 The **country domains** section uses two-character country abbreviations (e.g., us for United States). Second labels can be organizational, or they can be more specific national designations.

 The United States, **for example,** uses state abbreviations as a subdivision of us (e.g., ca.us.). Figure 4 shows the country domains section.



**Figure 4:Country domains**

 The address ***uci.ca.us.*** can be translated to University of California, Irvine, in the state of California in the United States.