UNIT-6

Transport Layer & Application Layer

Syllabus:

Transport Layer: Services provided to the upper layers, elements of transport protocol-addressing connection establishment, connection release, Connection Release, Crash Recovery.

Application layer: (WWW and HTTP): ARCHITECTURE: Client (Browser), Server, Uniform Resource Locator HTTP: HTTP Transaction, HTTP Operational Model and Client/Server Communication, HTTP Generic Message Format, HTTP Request Message Format, HTTP Response Message Format The wireless web: WAP—The Wireless Application Protocol

Transport layer:

- Transport layer is the heart of the whole protocol hierarchy.
- Transport layer task is to provide reliable, cost-effective data transport from the source machine to the destination machine, independently of the physical network or networks currently in use.

Transport Layer Services:

- **Process-To-Process Delivery:**

▪ Several processes may be running on the source host and several on the destination host. To complete the delivery, we need a mechanism to deliver data from one of these processes running on the source host to the corresponding process running on the destination host. That mechanism is called **" Process- Process Delivery"**.

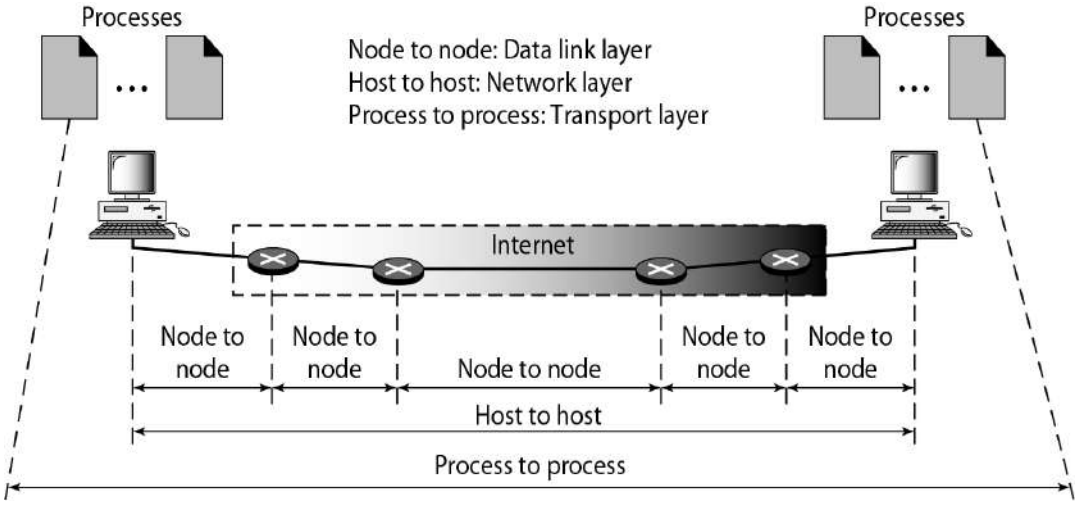


Figure: Process-To-Process Delivery

▪ Two processes communicate in a client/server relationship.

▪ **Client-Server Paradigm:**

  - A process on the local host, called a client, needs services from a process usually on the remote host, called a server.

- **Addressing:**
  - The address at transport layer is called " ***Service-Point address***" or " ***Port Address***". If the network model is OSI, then it is called " Service- Point address". If the network model is TCP/IP it is called " Port Address".
  - The port numbers are 16-bit integers between 0 and 65,535.
  - **The client** program defines itself with a port number, chosen randomly by the transport layer software running on the client host. This is the ***Ephemeral Port Number.***

- The server process must also define itself with a port number. This port number, cannot be chosen randomly. This is called " ***Well-Known Port Number***".

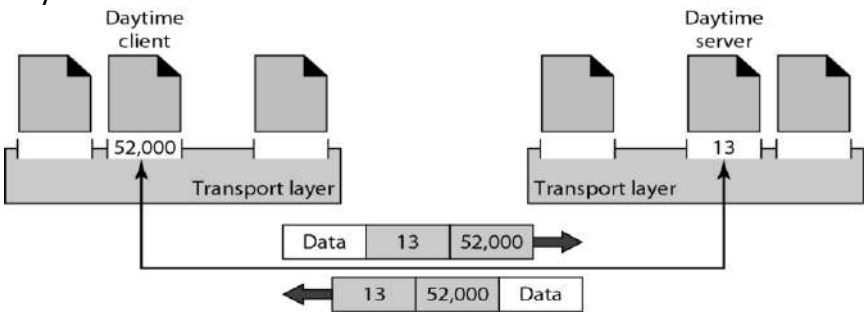


Figure: Port Addresses

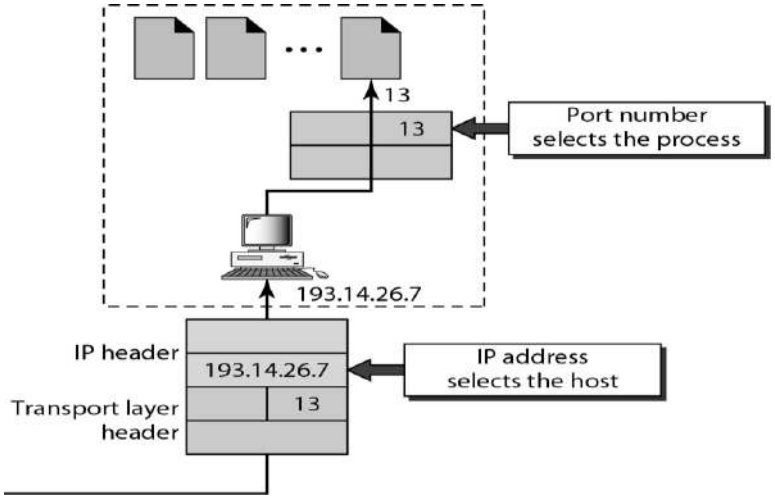


Figure: Relation Between IP address and Port address

- **IANA Ranges of Port Numbers:**
  - The lANA (Internet Assigned Number Authority) has divided the port numbers into three ranges: ***well known, registered, and dynamic (or private)***, as shown in Figure.

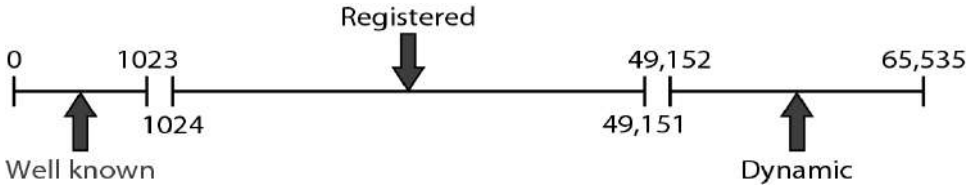


Figure: IANA ranges for Port numbers.

  - **Well-known ports:** The ports ranging from 0 to 1023 are assigned and controlled by IANA. These are the well-known ports.
  - **Registered ports:** The ports ranging from 1024 to 49,151 are not assigned or controlled by IANA. They can only be registered with lANA to prevent duplication.
  - **Dynamic ports:** The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used by any process. These are the ephemeral ports.
- **Socket Address:** Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection. The combination of an IP address and a port number is called a **Socket Address**.
- A transport layer protocol needs a pair of socket addresses: the client socket address and the server socket address.

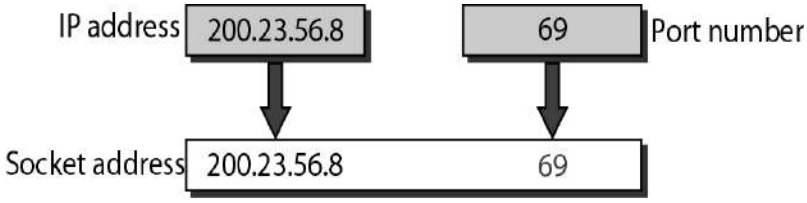


Figure: Socket Address

- **Multiplexing and Demultiplexing:**

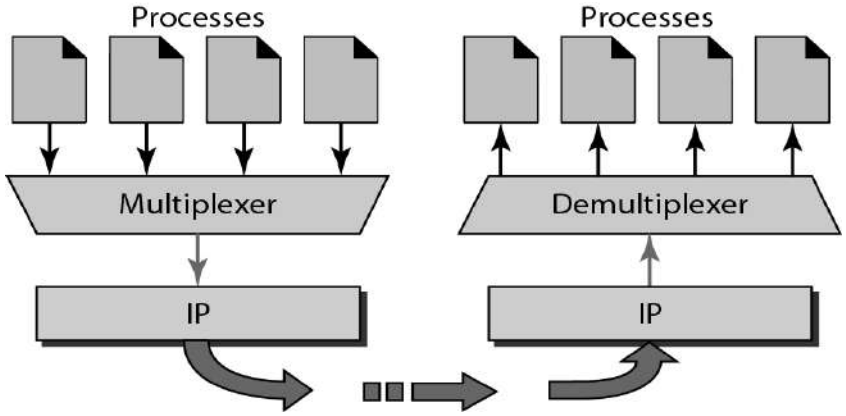


Figure: Multiplexing and Demultiplexing in Transport layer

- **Multiplexing:** At the sender site, there may be several processes that need to send packets. However, there is only one transport layer protocol at any time. This is a many-to-one relationship and requires multiplexing.
- **Demultiplexing:** At the receiver site, the relationship is one-to-many and requires Demultiplexing. The transport layer receives datagrams from the network layer. After error checking and dropping of the header, the transport layer delivers each message to the appropriate process based on the port number.

- **Connectionless Versus Connection-Oriented Service:**

  - A transport layer protocol can either be connectionless or connection-oriented.
  - ***Connectionless Service:*** In a connectionless service, the packets are sent from one party to another with no need for connection establishment or connection release.
    - The packets are not numbered
    - The packets may be delayed or lost or may arrive out of sequence.
    - There is no acknowledgment either.

  Ex: UDP

  - ***Connection Oriented Service:*** In a connection-oriented service, a connection is first established between the sender and the receiver. Data are transferred. At the end, the connection is released.

  Ex: TCP and SCTP

UDP (User Datagram Protocol):

- The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to process communication instead of host-to-host communication.
- It performs very limited error checking.
- **Well-Known Ports for UDP**

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
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
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

Table: Well-known port numbers for UDP

- **User Datagram:**
  - UDP packets, called user datagrams, have a fixed-size header of 8 bytes.

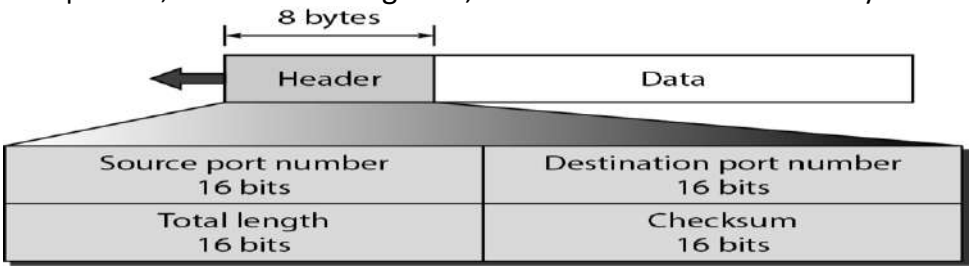


Figure: UDP Header

  - The fields are as follows:
    - **Source port number:** This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535.
      - If the source host is the client, the port number is an ephemeral port number.
      - If the source host is the server, the port number, in most cases, is a well-known port number.
    - **Destination port number:** This is the port number used by the process running on the destination host. It is also 16 bits long.
      - If the destination host is the , the port number, in most cases, is a well-known port number.

- If the destination host is the client, the port number, is an ephemeral port number.

- **Length:** This is a 16-bit field that 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.

  UDP length = IP length - IP header's length

- **Checksum:** This field is used to detect errors over the entire user datagram (header plus data).

TCP (Transmission Control Protocol):

- TCP is called a *connection-oriented, reliable* transport protocol. It adds connection-oriented and reliability features to the services of IP.
- Services Provided by TCP:
  - Besides to the transport layer services the TCP Protocol provide **Stream Delivery Service** and **Source and Destination Buffers**
  - **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 data across the Internet.
    - The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them.

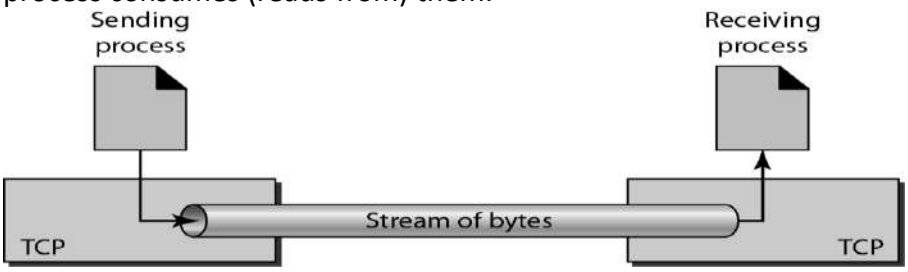


Figure: Stream Delivery

  - **Source and Destination Buffers:**
    - Sending and Receiving Buffers Because the sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction.

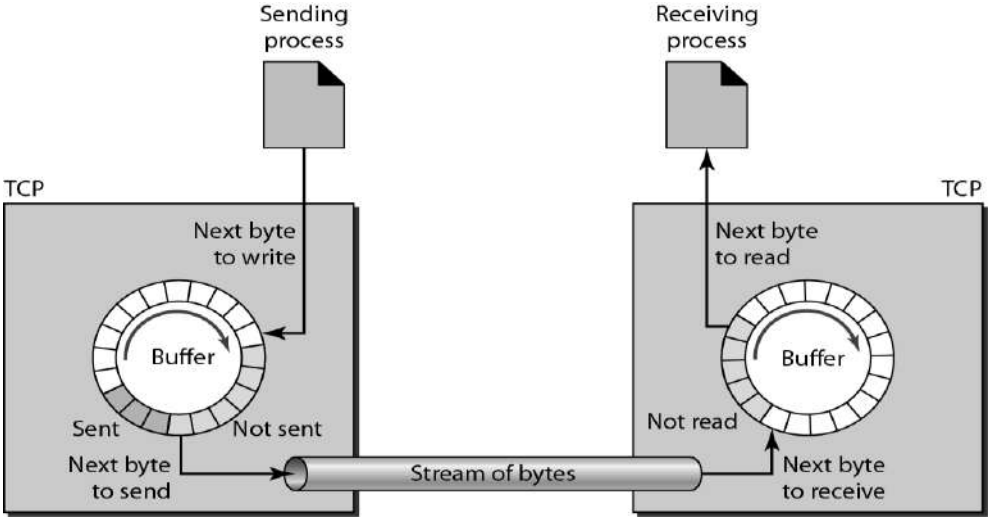


Figure: Sending and Receiving Buffers

- **Well Known Port numbers:**

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
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	FIP, Data	File Transfer Protocol (data connection)
21	FIP, Control	File Transfer Protocol (control connection)
23	TELNET	Tenninal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

Figure: Well-Known Port Addresses for TCP

- **<u>TCP Segment Format:</u>**
  - The segment consists of a 20- to 60-byte header, 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.

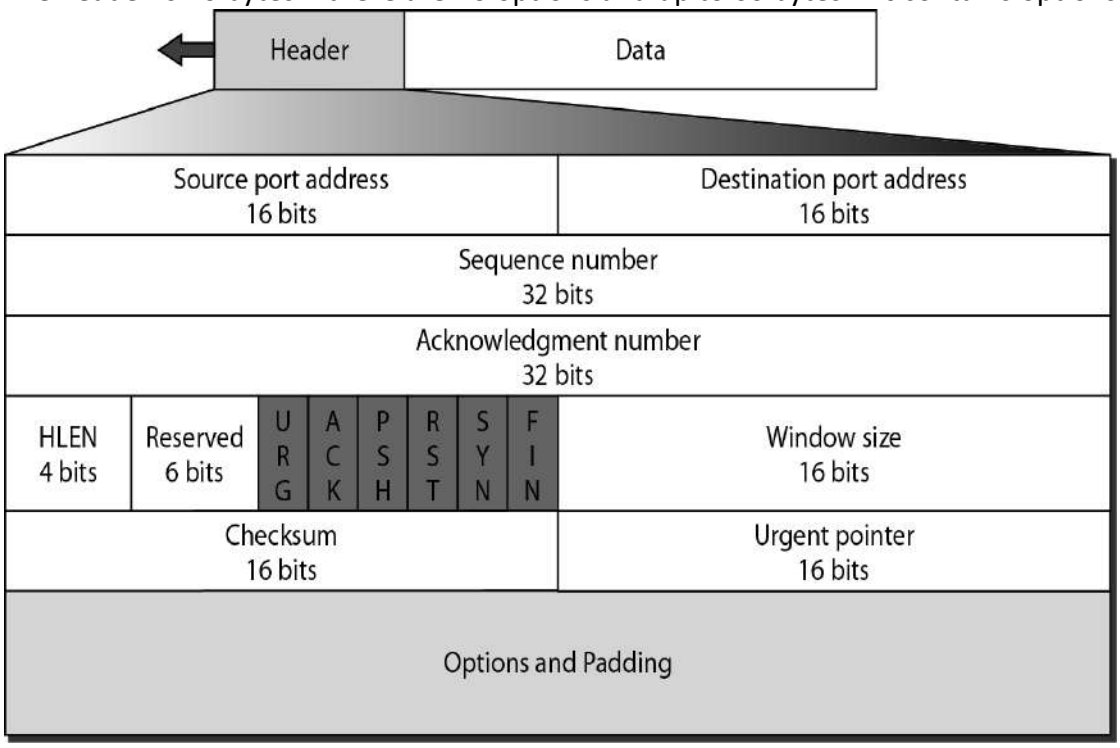


Figure: TCP Segment Format

- **Source port address:** This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.
- **Destination port address:** This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.
- **Sequence number:** This 32-bit field defines the number assigned to the first byte of data contained in this segment. TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence comprises the first byte in the segment.
- **Acknowledgment number:** This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number *x* from the other party, it defines *x* +1 as the acknowledgment number. Acknowledgment and data can be piggybacked together.
- **Header length:** This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 (5 x 4 =20) and 15 (15 x 4 =60).
- **Reserved:** This is a 6-bit field reserved for future use.
- **Control:** This field defines 6 different control bits or flags. One or more of these bits can be set at a time.

Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

Figure: Description of the flags in Control field.

- **Window size:** This field defines the size of the window, in bytes, that the other party must maintain. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.
- **Checksum:** This 16-bit field contains the checksum. This is used to detect errors in the header.
- **Urgent pointer:** This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment. This will be discussed later in this chapter.
- **Options:** There can be up to 40 bytes of optional information in the TCP header.

TCP Connection Management:

- TCP is connection-oriented. A connection-oriented transport protocol establishes a virtual path between the source and destination.
- All the segments belonging to a message are then sent over this virtual path.
- Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.
- The point is that a TCP connection is virtual, not physical. TCP operates at a higher level.
- TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself.
- If a segment is lost or corrupted, it is retransmitted. Unlike TCP, IP is unaware of this retransmission.
- If a segment arrives out of order, TCP holds it until the missing segments arrive; IP is unaware of this reordering.
- In TCP, connection-oriented transmission requires three phases:
    - **Connection Establishment**,
    - **Data Transfer**, and
    - **Connection Termination.**
- **Connection Establishment:**
    - TCP transmits data in full-duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously.
    - This implies that each party must initialize communication and get approval from the other party before any data are transferred.
    - **Three-Way Handshaking:**
        - The connection establishment in TCP is called three-way handshaking.
        - An application program, called the client, wants to make a connection with another application program, called the server, using TCP as the transport layer protocol.
        - The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This is called a request for a *passive open.* Although the server.
        - TCP is ready to accept any connection from any machine in the world, it cannot make the connection itself. The client program issues a request for an *active open.*
        - A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server.
        - TCP can now start the three-way handshaking process as shown in Figure.
        - Each segment has values for all its header fields and perhaps for some of its option fields, too.
- **SYN:** The client sends the first segment, a SYN segment, in which only the SYN flag is set. This segment is for synchronization of sequence numbers. It consumes one sequence number. When the data transfer starts, the sequence number is incremented by 1. The SYN segment carries no real data.
- **SYN+ACK:** The server sends the second segment, a SYN +ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number.
- **ACK:** The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. Note that the sequence

number in this segment is the same as the one in the SYN segment; the ACK segment does not consume any sequence numbers.

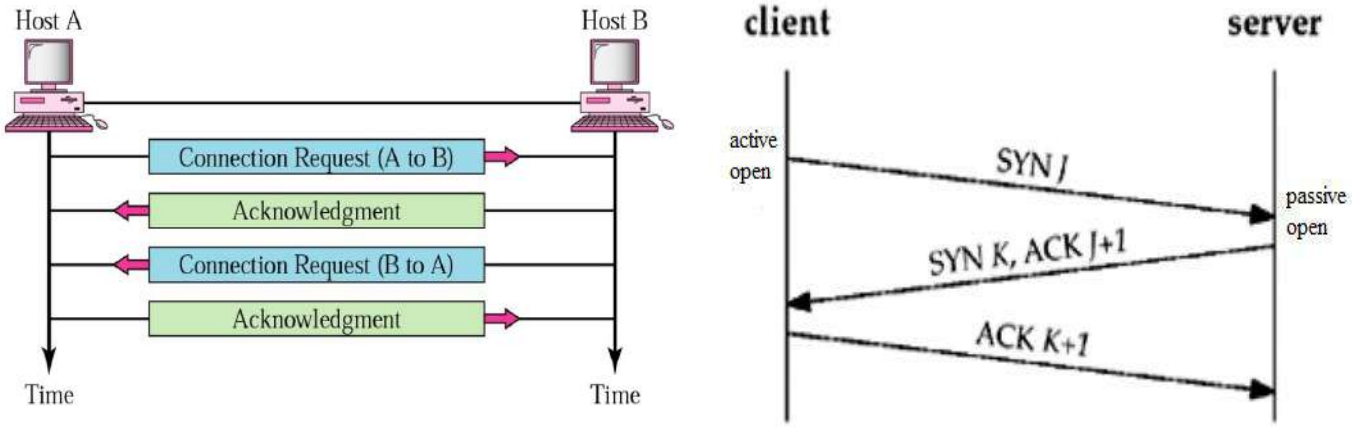


Figure: TCP Connection Establishment

- **Data Transfer:**
  - After connection is established, bidirectional data transfer can take place.
  - The client and server can both send data and acknowledgments.
  - The acknowledgment is piggybacked with the data.

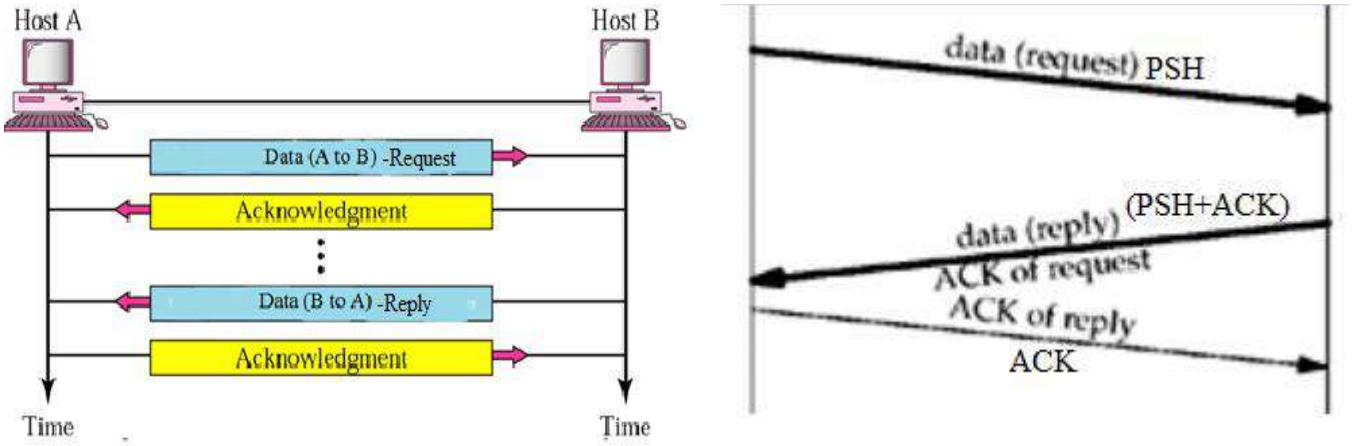


Figure: TCP Data transfer

- The data segments sent by the client have the PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received.
- The segment from the server, on the other hand, does not set the push flag.
- TCP uses a buffer to store the stream of data coming from the sending application program.
- The sending TCP can select the segment size.
- The receiving TCP also buffers the data when they arrive and delivers them to the application program when the application program is ready or when it is convenient for the receiving TCP.
- This type of flexibility increases the efficiency of TCP.

- **Connection termination:**
  - Any of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client.
  - TCP uses three-way handshaking for connection termination.
    1. In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set. Note that a FIN segment can include the last chunk of data sent by the client, or it can be just a control segment.
    2. The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN +ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction. This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number.
    3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is 1 plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

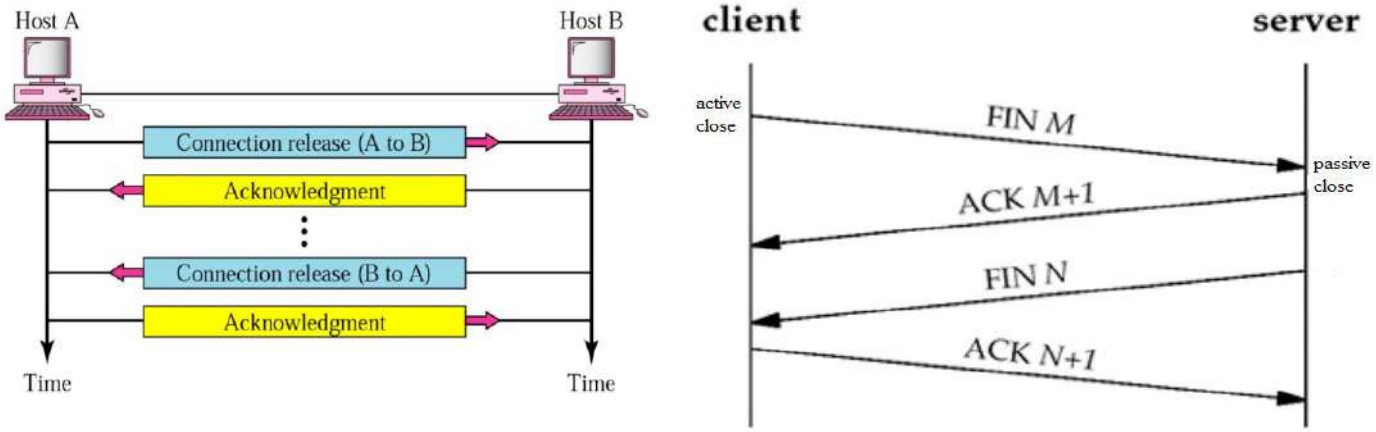


Figure: TCP Connection Termination

APPLICATION LAYER:

The application layer enables the user, whether human or software, to access the network. It provides user interfaces and support for services such as electronic mail, file access and transfer, access to system resources, surfing the world wide web, and network management.

The application layer is responsible for providing services to the user.

WWW (WORLD WIDE WEB):

The **World Wide Web (WWW)** is a repository of information linked together from points all over the world. The WWW has a unique combination of flexibility, portability, and user-friendly features that distinguish it from other services provided by the Internet. The WWW project was initiated by CERN (European Laboratory for Particle Physics) to create a system to handle distributed resources necessary for scientific research.

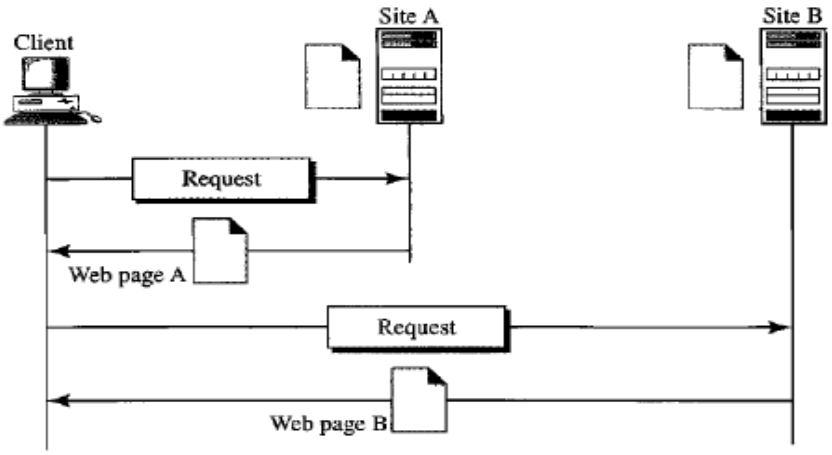
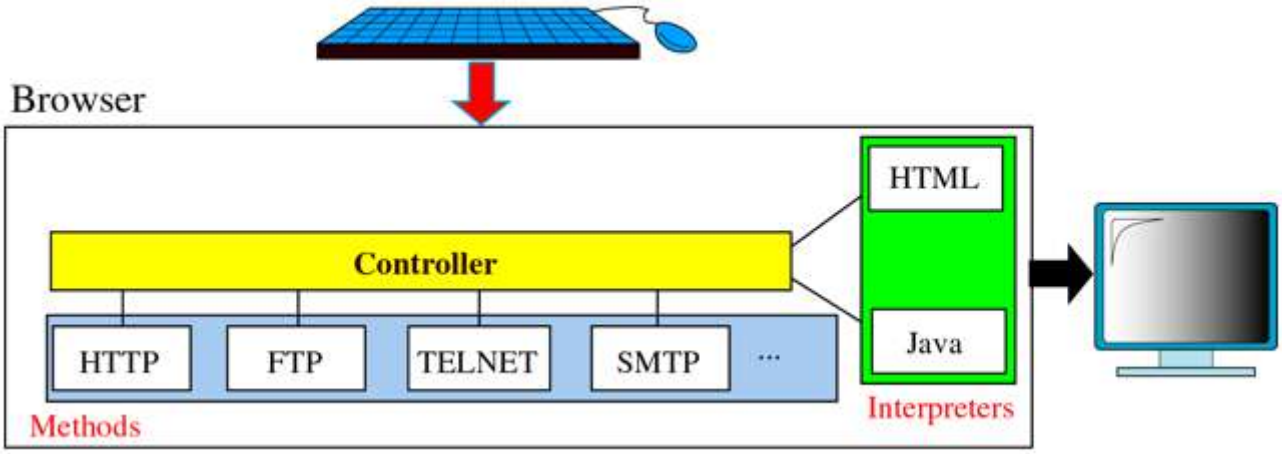


Figure: Architecture of WWW

Each site holds one or more documents, referred to as *Web pages*. Each Web page can contain a link to other pages in the same site or at other sites. The pages can be retrieved and viewed by using browsers. Let us go through the scenario shown in Figure . The client needs to see some information that it knows belongs to site A. It sends a request through its browser, a program that is designed to fetch **Web** documents. The request, among other information, includes the address of the site and the Web page, called the URL (Uniform Resource Locator) . The server at site A finds the document and sends it to the client. When the user views the document, they find some references to other documents, including a Web page at site B. The reference has the URL for the new site. The user is also interested in seeing this document. The client sends another request to the new site, and the new page is retrieved.

Client (Browser)

A variety of vendors offer commercial browsers that interpret and display a Web document, and all use nearly the same architecture. Each **browser** usually consists of three parts: a controller, client protocol, and interpreters. The controller receives input from the keyboard or the mouse and uses the client programs to access the document. After the document has been accessed, the controller uses one of the interpreters to display the document on the screen. The client protocol can be one of the protocols described previously such as FTP or HTTP (described later in the chapter). The interpreter can be HTML, Java, or JavaScript, depending on the type of document.



Server

The Web page is stored at the server. Each time a client request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than disk. A server can also become more efficient through multithreading or multiprocessing. In this case, a server can answer more than one request at a time.

****

URL (UNIFORM RESOURCE LOCATOR):

A client that wants to access a Web page needs the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators. The **uniform resource locator (URL)** is a standard for specifying any kind of information on the Internet. The URL defines four things: protocol, host computer, port, and path.

The *protocol* is the client/server program used to retrieve the document. Many different protocols can retrieve a document; among them are FTP or HTTP. The most common today is HTTP.

The **host** is the computer on which the information is located, although the name of the computer can be an alias. Web pages are usually stored in computers, and computers are given alias names that usually begin with the characters "www". This is not mandatory, however, as the host can be any name given to the computer that hosts the Web page.

The URL can optionally contain the port number of the server. If the *port* is included, it is inserted between the host and the path, and it is separated from the host by a colon.

Path is the pathname of the file where the information is located. Note that the path can itself contain slashes that, in the UNIX operating system, separate the directories from the subdirectories and files.

****

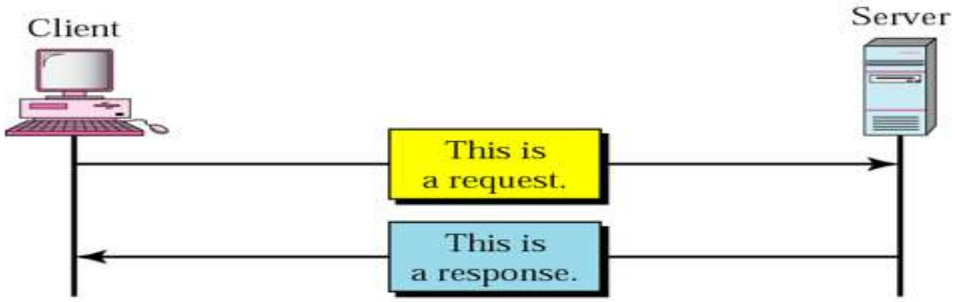
HTTP (Hyper Text Transfer Protocol):

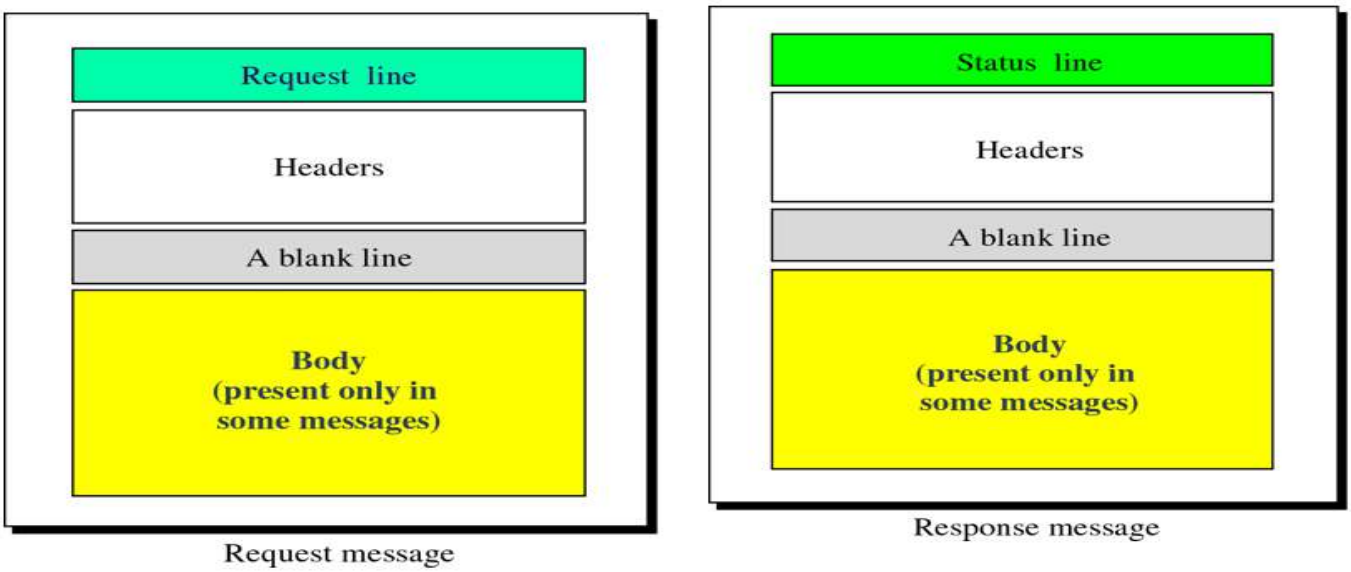
The **Hypertext Transfer Protocol (HTTP)** is a protocol used mainly to access data on the World Wide Web. HTTP functions as a combination of FTP and SMTP. It is similar to FTP because it transfers files and uses the services of TCP. However, it is much simpler than FTP because it uses only one TCP connection. There is no separate control connection; only data are transferred between the client and the server.

HTTP is like SMTP because the data transferred between the client and the server look like SMTP messages. In addition, the format of the messages is controlled by MIME-like headers. Unlike SMTP, the HTTP messages are not destined to be read by humans; they are read and interpreted by the HTTP server and HTTP client (browser). SMTP messages are stored and forwarded, but HTTP messages are delivered immediately. The commands from the client to the server are embedded in a request message. The contents of the requested file or other information are embedded in a response message. HTTP uses the services of TCP on well-known port 80.

HTTP Transaction:

Below figure illustrates the HTTP transaction between the client and server. Although HTTP uses the services of TCP, HTTP itself is a stateless protocol. The client initializes the transaction by sending a request message. The server replies by sending a response.





Request and Status Lines The first line in a request message is called a **request line;** the first line in the response message is called the **status line.** There is one common field, as shown in Figure .

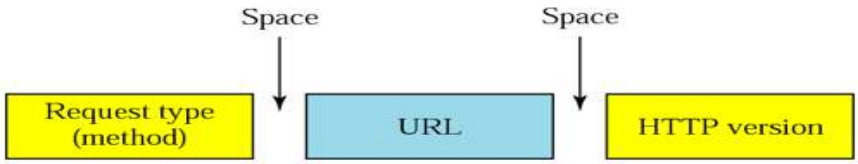


Figure: Request Line

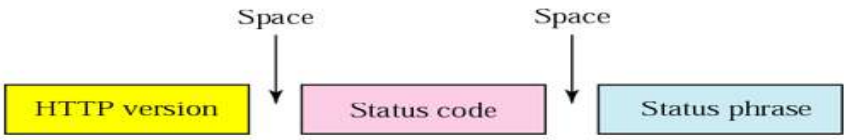


Figure: Status Line

- **Request type.** This field is used in the request message. In version 1.1 of HTTP, several request types are defined. The **request type** is categorized into *methods* as defined in Table.

Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
POST	Sends some information from the client to the server
PUT	Sends a document from the server to the client
TRACE	Echoes the incoming request
CONNECT	Reserved
OPTION	Inquires about available options

- **URL** :Uniform Resource Locator.
- **Version.** The most current version of HTTP is 1.1.
- **Status code.** This field is used in the response message. The **status code** field is similar to those in the FTP and the SMTP protocols. It consists of three digits. Whereas the codes in the 100 range are only informational, the codes in the 200 range indicate a successful request. The codes in the 300 range redirect the client to another URL, and the codes in the 400 range indicate an error at the client site. Finally, the codes in the 500 range indicate an error at the server site. We list the most common codes in Table
- **Status phrase.** This field is used in the response message. It explains the status code in text form.

Code	*Phrase*	*Description*
	Informational	
100	Continue	The initial part of the request has been received, and the client may continue with its request.
101	Switching	The server is complying with a client request to switch protocols defined in the upgrade header.
	Success	
200	OK	The request is successful.
201	Created	A new URL is created.
202	Accepted	The request is accepted, but it is not immediately acted upon.
204	No content	There is no content in the body.
	Redirection	
301	Moved permanently	The requested URL is no longer used by the server.
302	Moved temporarily	The requested URL has moved temporarily.
304	Not modified	The document has not been modified.
	Client Error	
400	Bad request	There is a syntax error in the request.
401	Unauthorized	The request lacks proper authorization.
403	Forbidden	Service is denied.
404	Not found	The document is not found.
405	Method not allowed	The method is not supported in this URL.
406	Not acceptable	The format requested is not acceptable.
	Server Error	
500	Internal server error	There is an error, such as a crash, at the server site.
501	Not implemented	The action requested cannot be performed.
503	Service unavailable	The service is temporarily unavailable, but may be requested in the future.

Header The header exchanges additional information between the client and the server. For example, the client can request that the document be sent in a special format, or the server can send extra information about the document. The header can consist of one or more header lines. Each header line has a header name, a colon, a space, and a header value .

A header line belongs to one of four categories: **general header, request header, response header,** and **entity header.** A request message can contain only general, request, and entity headers. A response message, on the other hand, can contain only general, response, and entity headers.

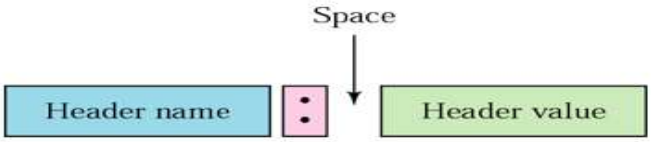


Figure: Header Format

- **General header** The general header gives general information about the message and can be present in both a request and a response.

Table : *General headers*

Header	*Description*
Cache-control	Specifies information about caching
Connection	Shows whether the connection should be closed or not
Date	Shows the current date
MIME-version	Shows the MIME version used
Upgrade	Specifies the preferred communication protocol

- **Request header** The request header can be present only in a request message. It specifies the client's configuration and the client's preferred document format.

Table: Request Headers

Header	*Description*
Accept	Shows the medium format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
From	Shows the e-mail address of the user
Host	Shows the host and port number of the server
If-modified-since	Sends the document if newer than specified date
If-match	Sends the document only if it matches given tag
If-non-match	Sends the document only if it does not match given tag
If-range	Sends only the portion of the document that is missing
If-unmodified-since	Sends the document if not changed since specified date
Referrer	Specifies the URL of the linked document
User-agent	Identifies the client program

- **Response header** The response header can be present only in a response message. It specifies the server's configuration and special information about the request.

Table *Response headers*

Header	*Description*
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age of the document
Public	Shows the supported list of methods
Retry-after	Specifies the date after which the server is available
Server	Shows the server name and version number

- **Entity header** The entity header gives information about the body of the document. Although it is mostly present in response messages, some request messages, such as POST or PUT methods, that contain a body also use this type of header.

Table : *Entity headers*

Header	*Description*
Allow	Lists valid methods that can be used with a URL
Content-encoding	Specifies the encoding scheme
Content-language	Specifies the language
Content-length	Shows the length of the document
Content-range	Specifies the range of the document
Content-type	Specifies the medium type
Etag	Gives an entity tag
Expires	Gives the date and time when contents may change
Last-modified	Gives the date and time of the last change
Location	Specifies the location of the created or moved document

Body The body can be present in a request or response message. Usually, it contains the document to be sent or received.

WAP (Wireless Application Protocol):

- **Wireless Application Protocol** (**WAP**) is a technical standard for accessing information over a mobile wireless network.
- A **WAP browser** is a web browser for mobile devices such as mobile phones that uses the protocol. Introduced with much hype in 1999, WAP achieved some popularity in the early 2000s, but by the 2010s it had been largely superseded by more modern standards.
- Most modern handset internet browsers now fully support HTML, so they do not need to use WAP markup for web page compatibility, and therefore, most are no longer able to render and display pages written in WAP.
- **WAP architecture:**
  - The following figure shows the basic WAP architecture. There are three participating entities: the WAP browser, the WAP gateway (also called WAP proxy) and a server on the Internet.

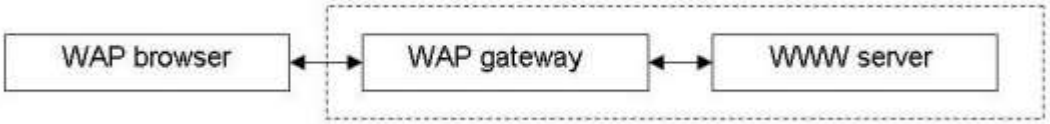


Figure: WAP Architecture

  - When the mobile device wants to connect to the Internet, all the communication passes through the WAP gateway. This WAP gateway translates all the protocols used in WAP to the protocols used on the Internet.
  - For example, the WAP proxy encodes (and decodes) the content to reduce the size of the data that has been sent over the wireless link.
  - Another example is the WTLS protocol. The communication between the mobile device and the WAP gateway is secured with WTLS. WTLS is only used between the mobile device and the WAP gateway, while SSL/TLS can be used between the gateway and the Internet.
  - This means that the WAP gateway first has to decrypt the encrypted WTLS-traffic and then has to encrypt it again (using SSL/TLS).

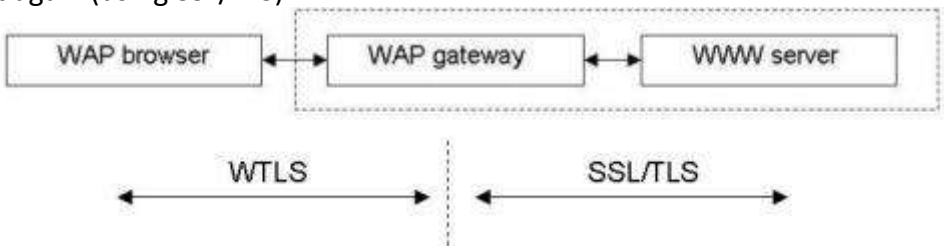


Figure: WTLS-traffic gets translated to SSL/TLS-traffic

- **WAP protocol stack**:
  - Many years ago, a theoretical protocol stack was developed by the OSI (Open Systems Initiative). This was done to facilitate a common understanding of the functionality provided by a protocol stack and to facilitate comparisons between different vendor's implementations.
  - The mapping of the WAP protocol stack to the OSI model is shown in figure.
  - The WAP protocol stack contains the following elements
  - **Physical and Data Link Layer:** In WAP, Point to Point Protocols (PPP) are used over one or more Over-The-Air (OTA) bearer protocols.
  - **Network Layer:** IP is the network layer of choice. However, not all wireless networks are capable of transmitting IP. That is why SMS or some other non-packet network protocol can be used.

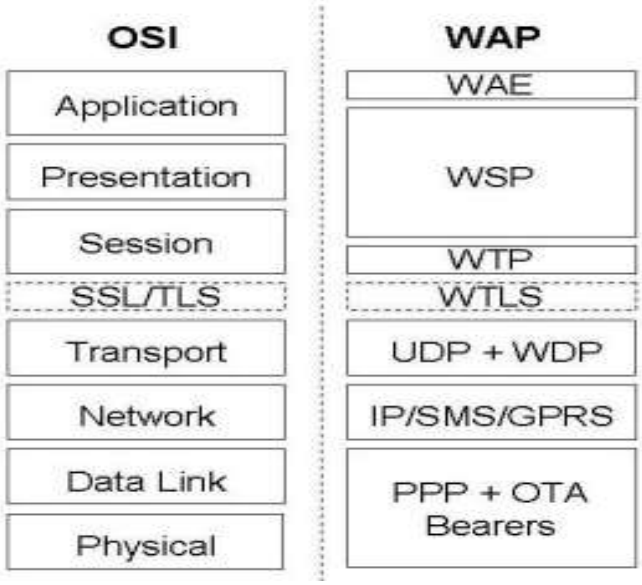


Figure: WAP protocol stack

- **Transport Layer:** The protocol used in the transport layer is UDP. However, this may not be feasible over non-IP networks. That is why (there are also other reasons) that WAP defines an additional transport layer protocol, WDP, which can be used when UDP cannot.
- **Session Layer:** The functionality of the session layer is partially included in WTP. Other aspects of the functionality are implemented in WSP.
- **Presentation Layer:** The functionality of the presentation layer is included in WSP.
- **Application Layer:** Some aspects of the functionality of the application layer are included in WSP, the others are implemented in WAE.

****

Crash Recovery in Transport Layer:

- If hosts and routers are subject to crashes, recovery from these crashes becomes an issue.
- If the transport entity is entirely within the hosts, recovery from network and router crashes is straight forward.
- If the network layer provides datagram service, the transport entities expect lost TPDUs all the time and know how to cope (deal effectively with something difficult) with them.
- If the network layer provides connection-oriented service, then loss of a virtual circuit is handled by establishing a new one and then probing the remote transport entity to ask it which TPDUs it has received and which ones it has not received.
- In particular, it may be desirable for clients to be able to continue working when servers crash and then quickly reboot.
- No matter how the client and server are programmed, there are always situations where the protocol fails to recover properly.
- **The server can be programmed in one of two ways**:
  - acknowledge first
  - write first.
- **The client can be programmed in one of four ways:**
  - always retransmit the last TPDU,
  - never retransmit the last TPDU,
  - retransmit only in state S0,
  - retransmit only in state S1

- This gives eight combinations, but as we shall see, for each combination there is some set of events that makes the protocol fail.
- Three events are possible at the server:
  - sending an acknowledgement
  - writing to the output process (W)
  - crashing (C).
- The three events can occur in six different orderings: AC(W), AWC, C(AW), C(WA), WAC, and WC(A), where the parentheses are used to indicate that neither A nor W can follow C (i.e., once it has crashed, it has crashed).

Strategy used by receiving host

Strategy used by sending host	First ACK, then write			First write, then ACK		
	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S1	LOST	OK	OK	OK	OK	DUP

OK = Protocol functions correctly
DUP = Protocol generates a duplicate message
LOST = Protocol loses a message

- Figure shows all eight combinations of client and server strategy and the valid event sequences for each one.
- Notice that for each strategy there is some sequence of events that causes the protocol to fail. For example, if the client always retransmits, the AWC event will generate an undetected duplicate, even though the other two events work properly.

****