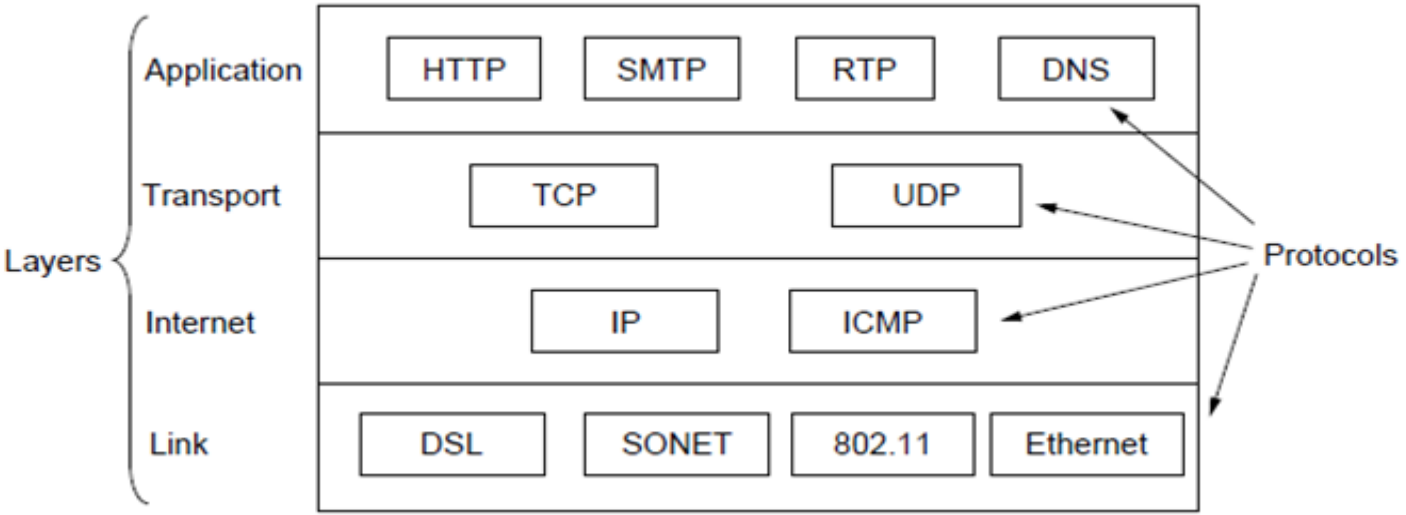
Transport Layer

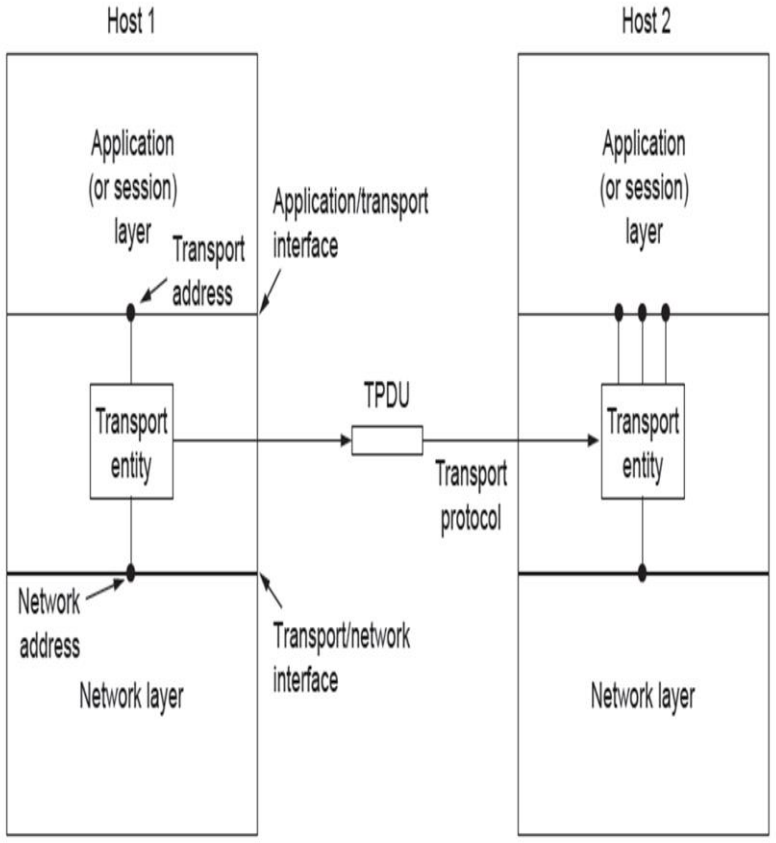
Transport Layer

- The transport layer is the fourth layer in the open system interconnection (OSI) model, and is responsible for end-to-end communication over a network.
- It provides logical communication between application processes running on different hosts within a layered architecture of protocols and other network components.
- The transport layer is also responsible for the management of error correction, providing quality and reliability to the end user.



Transport Layer

Services Provided to the Upper Layers



- The software and/or hardware within the transport layer that does the work is called the **Transport Entity**. The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card.
- It provides both connection oriented and connection less services
- In a connectionless network, packets are lost or damaged, the transport entity can detect the problem and ask for retransmissions.
- In in a connection-oriented network, a transport entity is informed halfway through a long transmission that its network connection has been abruptly terminated, with no indication of what has happened to the data currently in transit, it can set up a new network connection to the remote transport entity.

Transport Layer

Transport Service Primitives:

To allow users to access the transport service, the transport layer must provide some operations to application programs, that is, a transport service interfaces or primitives

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

Transport Layer

Transport Service Primitives:

These primitives allows application programs to establish, use, and then release connections, which is sufficient for many applications.

LISTEN: This primitive is executed by a server by calling a library procedure that makes a system call that blocks the server until a client turns up when it is ready to accept request of in coming connection. And allocates space to queue incoming calls for the case that several clients try to connect at the same time

CONNECT: This primitive is executed by client whenever it wants to communicate with the server

SEND: This primitive is put into action by the client to transmit its request that is followed by putting receive primitive into action to get the reply.

RECEIVE: Receive primitive simply waits for incoming message. SEND and RECEIVE

DISCONNECT : This primitive is simply used to terminate or end the connection after which no one will be able to send any of the message.

Transport Layer

Berkeley Sockets Primitives for TCP

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Transport Layer

Berkeley Sockets Primitives

The first four primitives in the list are executed in that order by servers.

SOCKET primitive creates a new endpoint and allocates table space for it within the transport entity. And specify the addressing format to be used, the type of service desired (e.g., reliable byte stream), and the protocol. SOCKET returns file pointer / Object

BIND: This primitive assigns address to the socket as newly created sockets do not have network addresses. Once a server has bound an address to a socket, remote clients can connect to it.

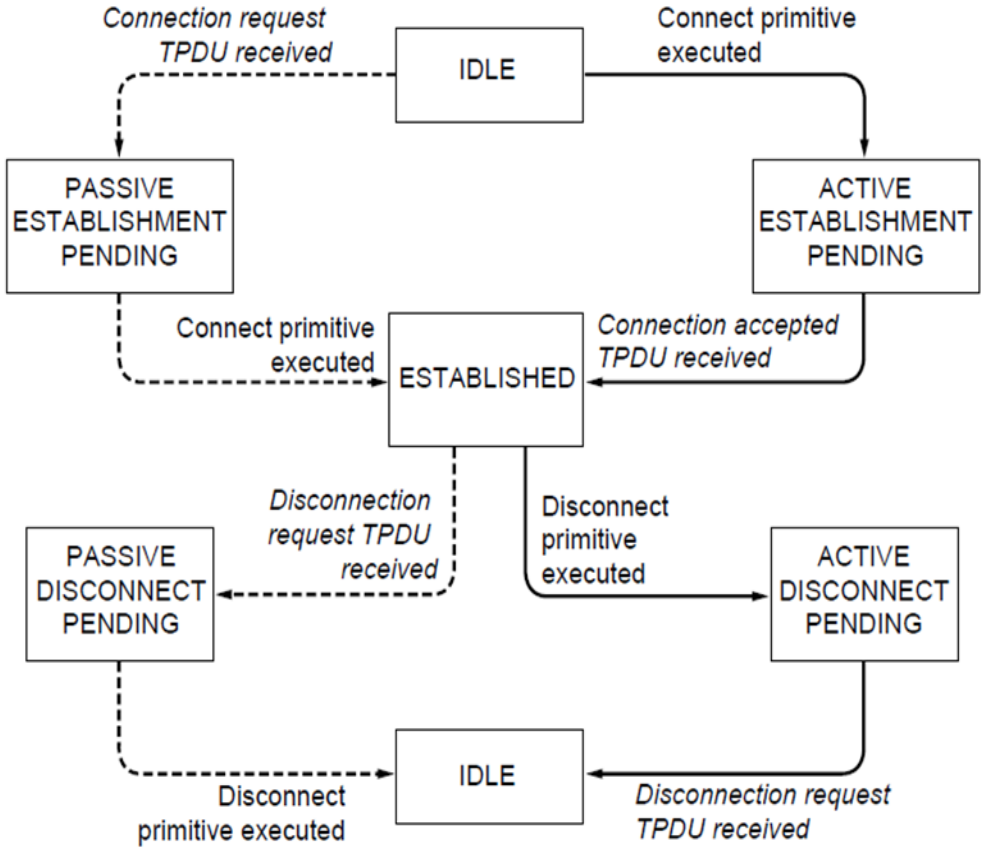
ACCEPT : Blocks the server and waits for an incoming connection. When segment asking for a connection arrives, the transport entity creates a new socket with the same properties as the original one and returns a file descriptor/ object for it.

CLOSE : When both sides have executed a CLOSE primitive, the connection is released.

Note: Example of Python socket programming is in LAB cycle

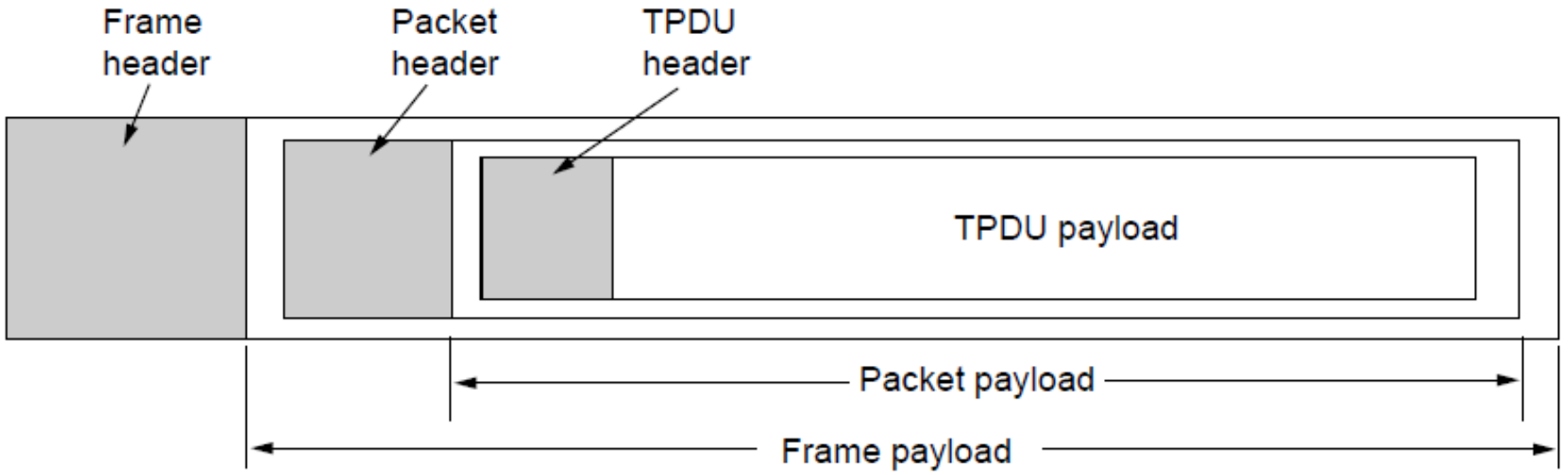
Transport Layer

Berkeley Sockets Primitives State Diagram



A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.

Nesting of segments, packets, and frames



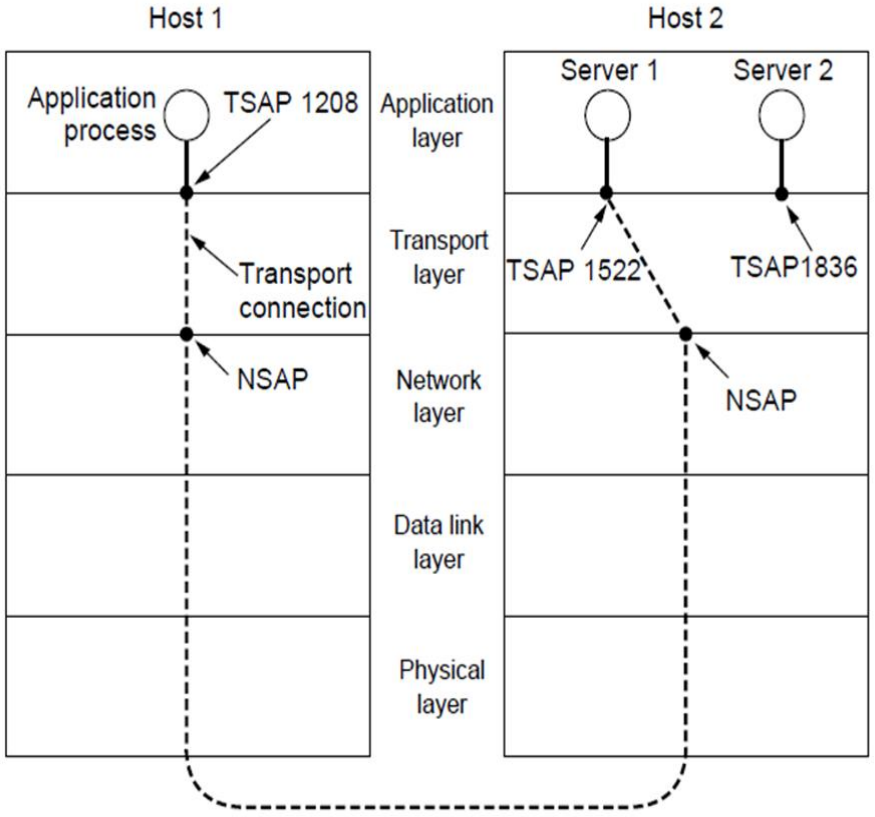
Transport Layer

Elements of Transport Protocols

- Addressing
- Connection establishment
- Connection release
- Error control and flow control
- Multiplexing
- Crash recovery

Transport Layer

Addressing



As the multiple application can be in process at both the sender and receiver host, it is required to set up a connection to a remote application process, which must specify destiny process. (Connectionless transport has the same problem: to whom should each message be sent?). The addressing of process can be done by defining transport addresses to each process on the host.

The communication end points in the internet are called **Ports**. Ports in the transport layer are called **TSAP (Transport Service Access Point).** The communication end points in network layer (i.e., network layer addresses) are called **NSAPs** (Network Service Access Points).

Application processes, both at clients and servers, can attach themselves to a local TSAP to establish a connection to a remote TSAP.

Addressing

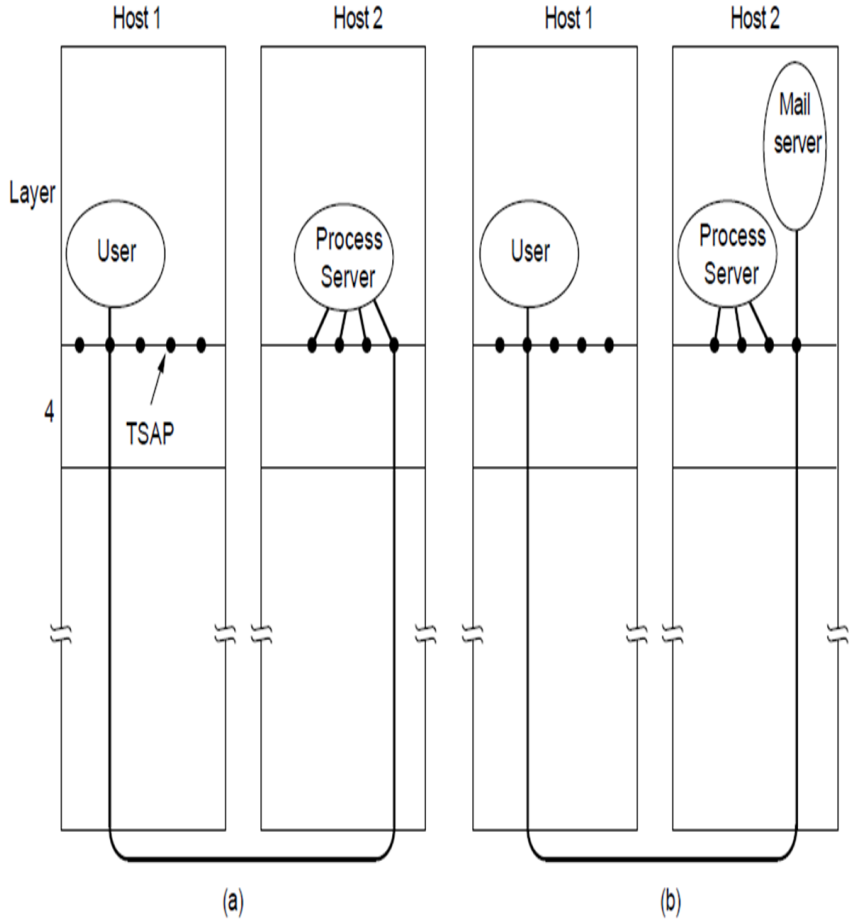
Generally the services have stable TSAP addresses that are listed in files. For example /etc/services file on UNIX systems lists which servers are permanently attached to which ports. But in general the user processes that do not have TSAP addresses that are known in advance, or that may exist for only a short time. To handle user process address, a special process called Portmapper is used.

Portmapper: is function that accepts the request to find the TSAP address corresponding to a given service name, and then sends back the TSAP address. When a new service is created, it must register itself with the portmapper, giving both its service name (typically, an ASCII string) and its TSAP. The portmapper records this information in its internal database.

The main drawback of portmapper is as many of the server processes are rarely used, It is wasteful to have each of them active and listening to a stable TSAP address. The solution to this problem is **Initial Connection Protocol.**

Transport Layer

Addressing



Process server

Acts as a proxy for less rarely used servers. This server is called *inetd* on UNIX systems. It listens to a set of ports at the same time, waiting for a connection request.

Users of a service request for CONNECT by specifying the TSAP address of the service they want. If no server is waiting for them, they get a connection to the process server.

After it gets the incoming request, the process server spawns the requested server, allowing it to inherit the existing connection with the user.

The new server does the requested work, while the process server goes back to listening for new requests.

How a user process in host 1 establishes a connection with a mail server in host 2 via a process server

Transport Layer

Connection Management

Process (end) to Process (end) delivery can be accomplished either using connection oriented or connectionless and most .

The connection oriented mode is most commonly used modes. A connection oriented protocol establishes a virtual circuited or path between Sender and Receiver.

All of the packets belonging to a message are then sent over this same path. Using a single pathway for the entire message facilitates the acknowledgement process and retransmission of damaged and lost frames connection oriented services is generally considered reliable.

Connection oriented mode has three phases

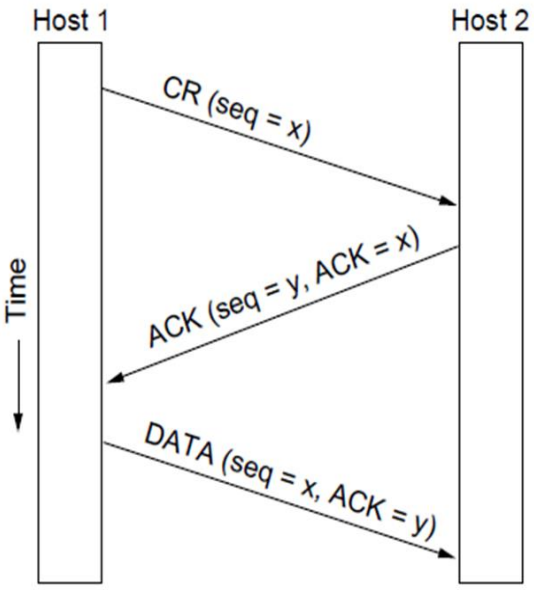
1. **Connection Establishment.**
2. **Data Transfer**
3. **Connection Termination**

Connection Management

Connection Establishment:

Before sending data to the other, the initializing device must first determine the availability of the receiving device at the other end and set a pathway through the network by which the data can be sent.

Establishment Phase performers three actions called **3-way Handshaking**

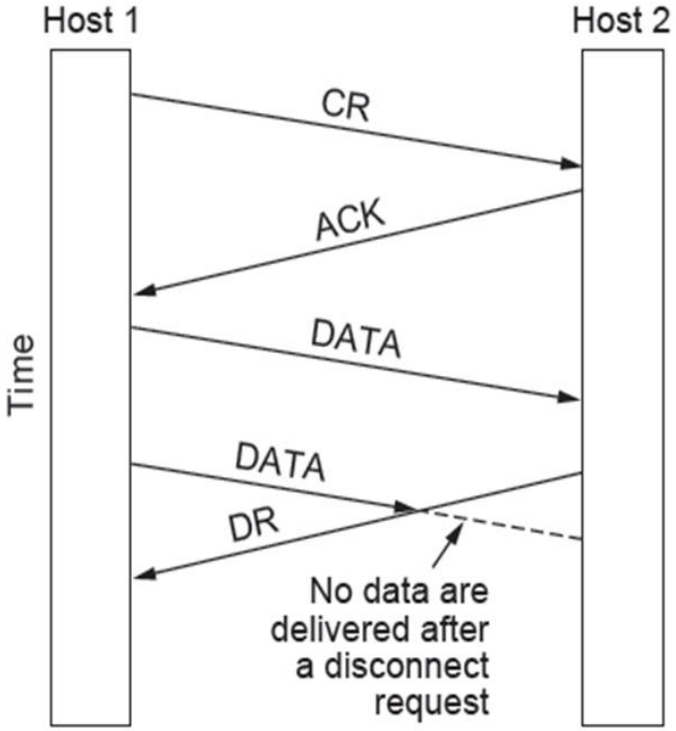


- The computer requesting the connection sends a connection request (CR) packet to the intended receiver with Seq= X
- The receiving computer returns a confirmation packet to the requesting computer by acknowledging x with Seq=Y
- The requesting sender returns a packet acknowledging the confirmation with the initial sequence number in the first data segment that it sends.
- TCP uses this three-way handshake to establish connections. Within a connection, a timestamp is used to extend the 32-bit sequence number so that it will not wrap within the maximum packet lifetime,

Transport Layer

Connection Management

Connection Release:



Asymmetric release

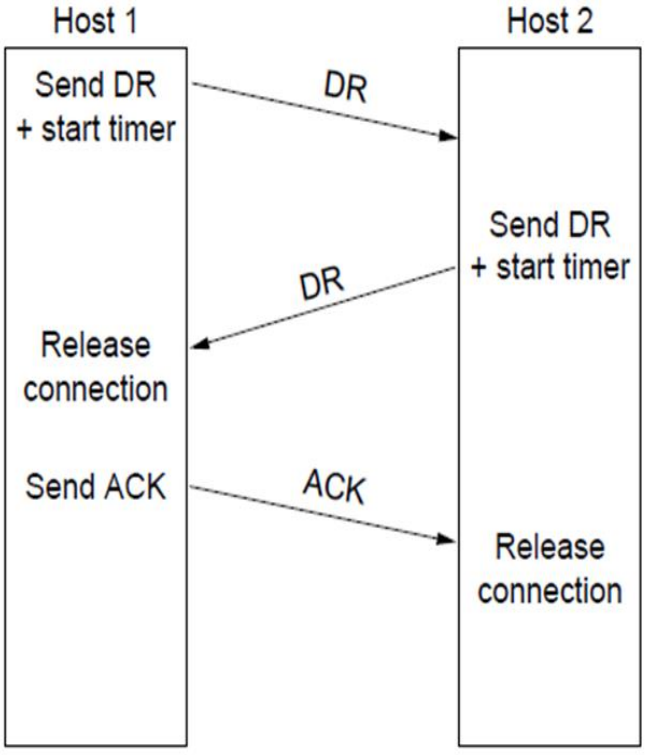
Once all the data have been transferred, the connection must be terminated. there are two ways of terminating a connection: **Asymmetric release** and **Symmetric release**.

Asymmetric release: is the way the telephone system works: when one party hangs up, the connection is broken.
Asymmetric release is abrupt and may result in data loss.

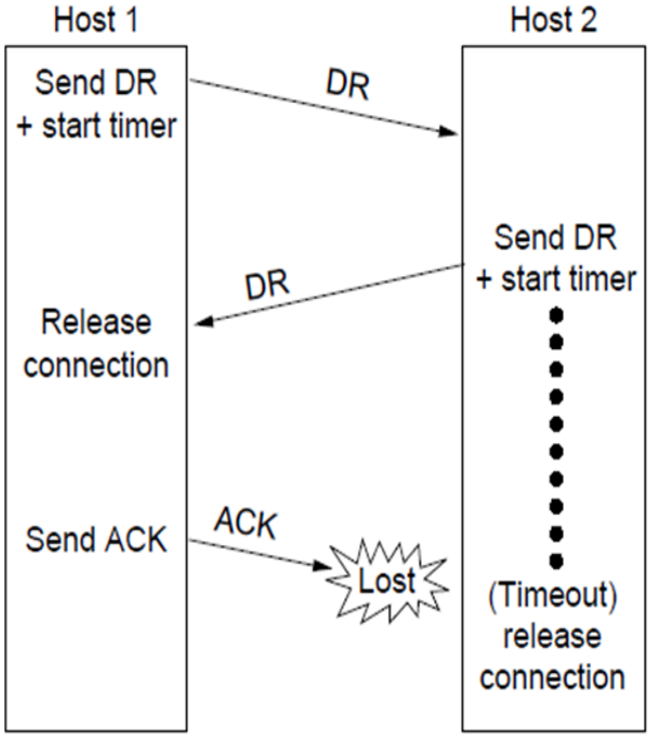
Transport Layer

Symmetric Connection Release:

Symmetric release treats the connection as two separate unidirectional connections and requires each one to be released separately.



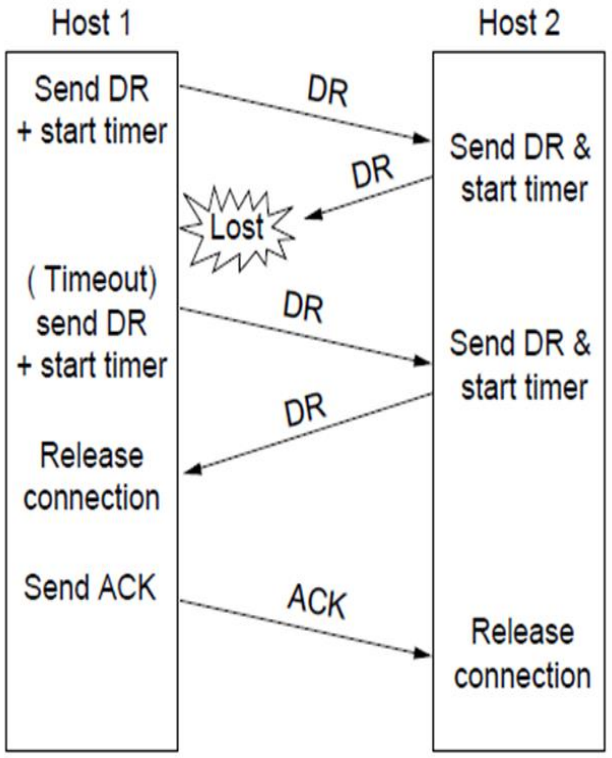
(a)Normal Release

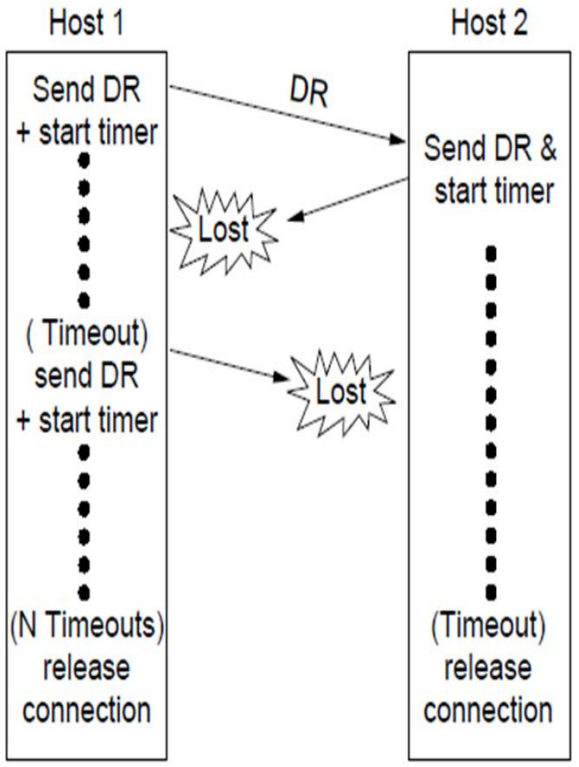


(b) Final ACK lost.

Transport Layer

Symmetric Connection Release:





(C) Response lost

(d) Response lost and subsequent DRs lost.

Transport Layer

FLOW CONTROL:

Transport layer is responsible for flow control. Flow control is performed end to end rather than across a single link. Transport layer flow control uses a sliding window protocol. The window at the transport layer can vary in size to accommodate buffer occupancy.

- Sliding window is used to make data transmission more efficient as well as to control the flow of data so that the receiver does not become overwhelmed.
- Sliding window used at the transport layer are usually byte oriented rather than frame oriented.
- The flow control is done in Transport layer using **Credit based** sliding window protocol.

Transport Layer

FLOW CONTROL:

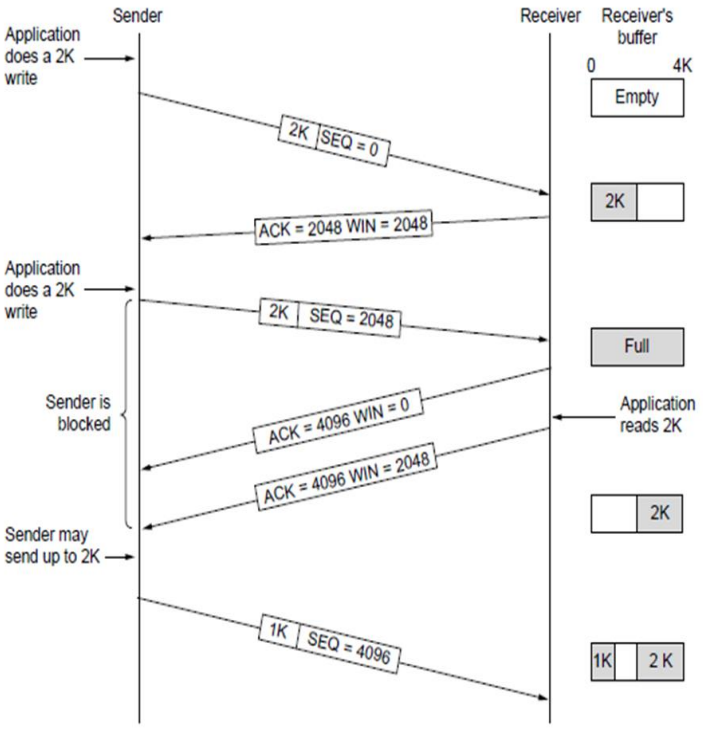
Assume that Host A needs to transmit 4KB of data to Host B with the buffer size of 4KB. Demonstrate how the flow control is done in transport layer with credit based slinding window protocol.

1) Initially buffer size = Empty 4KB
2) Sender Data = 2 K, Sequence No. = 0 > Reciever
   Buffer 2KB data 2kb empty
3) Sender <ACK = 2048, Sequence No. = 2048, Credit = 2048 Receiver

4) Sender Data = 2048, Sequence No. = 2048 > Receiver
   Buffer Full
5) Sender <ACK = 4096, Credit = 0 **Receiver**

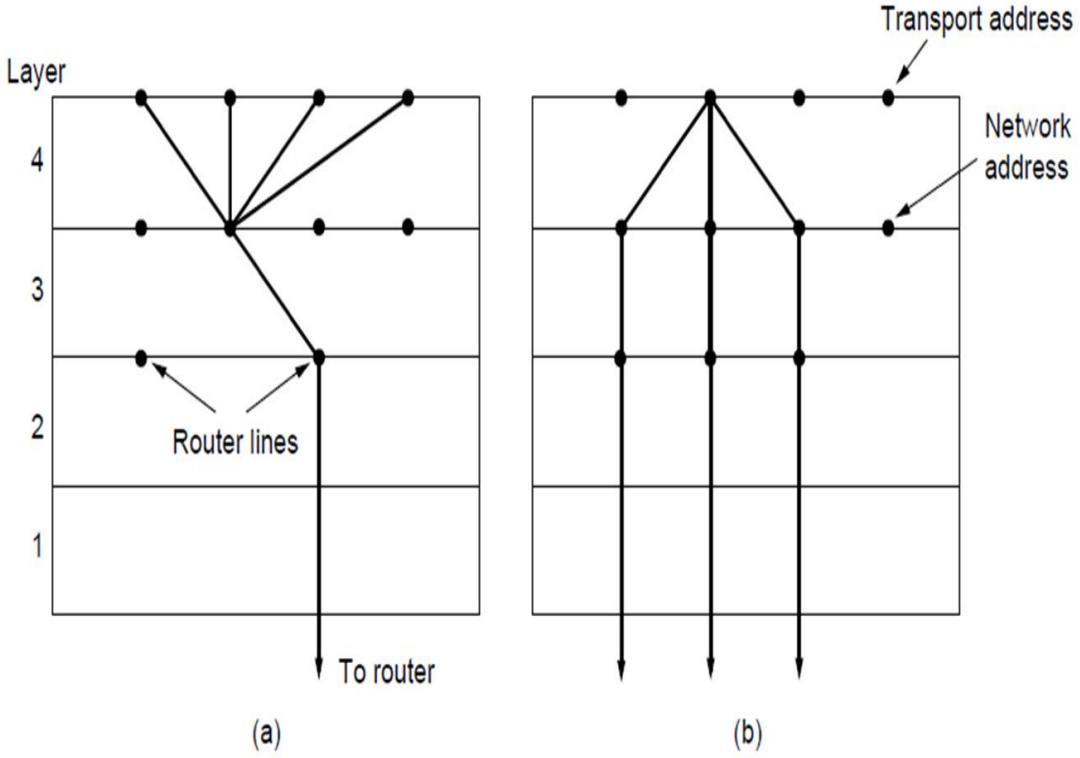
6) Application reads 2KB of data then
   Sender < ACK = 4096, Credit = 2048 Reciver
7) Sender transmits the remaing data

This is how flow control happens in Transport layer



Multiplexing

Sharing several conversations over connections, virtual circuits, and physical links improves transmission efficiency .



Upward Multiplexing: Sharing of single Ip for multiple ports. If only one network address is available on a host, all transport connections on that machine have to use it.

Inverse Multiplexing: One transport connection uses multiple network connection. This is also called as Downward multiplexing. If a user needs more bandwidth or more reliability than one of the network paths can provide, then a single transport connection distributes the traffic among multiple network paths on a round-robin basis.

Transport Layer

Internet Transport Protocol

The Internet has two main protocols in the transport layer, a connectionless protocol and a connection-oriented protocol.

UDP (User Datagram Protocol):

UDP provides a way for applications to send IP datagrams without establish a connection i,e through connection less service. UDP transmits segments consisting of an 8-byte header followed by the payload.

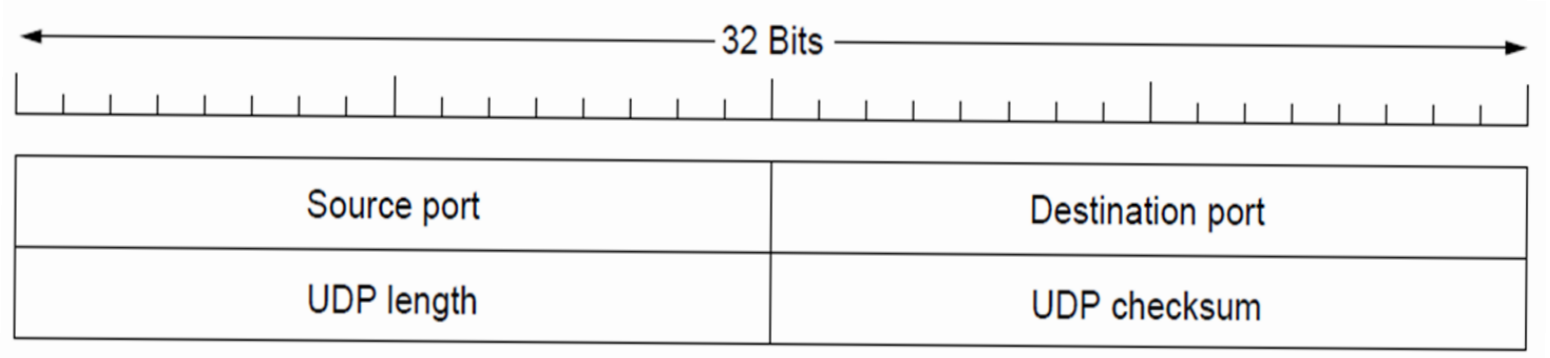
- It is a connectionless, unreliable and fast protocol
- It does not guarantee ordered delivery.
- It does not provide congestion control mechanism, but it is a good protocol for data flowing in one direction.
- It avoids the overhead of connection establishment and termination
  UDP is suitable for the applications like distributing routing information in RIP and OSPF Algorithms.

Used for real time applications which can not tolerate uneven delays between sections of a received message. Implementation of DNS, DHCP etc uses UDP as a transport layer protocol

Transport Layer

Internet Transport Protocol (UDP)

UDP Header format



Source Port: is a 16 bit field. It identifies the port id of the sending application.

Destination Port: is a 16 bit field. It identifies the port of the receiving application.

Length: is a 16 bit field. It identifies the combined length of UDP Header and Encapsulated data.

Checksum: a 16 bit field used for error control. It is calculated on UDP Header, encapsulated data and IP pseudo header. Checksum calculation is not mandatory in UDP.

Transport Layer

Remote Procedure Call (RPC)

The technique of calling a procedure on remote machine (server) by a process on other machine (client) through a network is called **Remote Procedure Call.**

Calling process on client is suspended and called procedure on the server is executed, Information can be transported from the caller to the callee in the parameters and can come back in the procedure result. No message passing is visible to the application programmer.

A **Stub** in distributed computing is a piece of code that converts parameters passed between client and server during a remote procedure call (RPC).

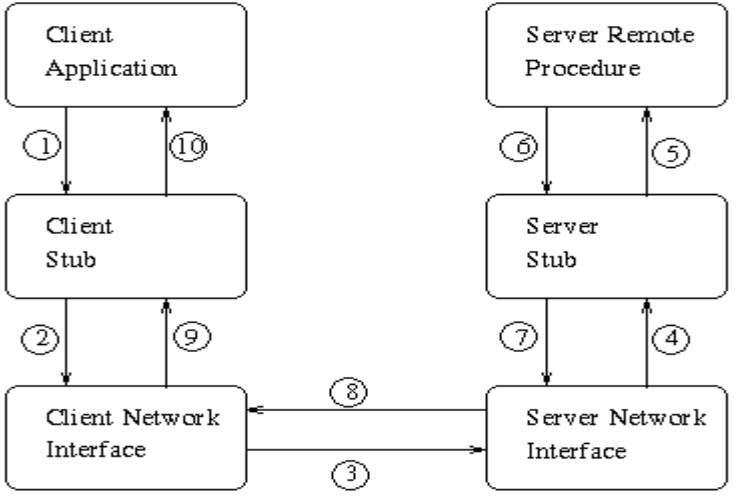
As the client and server use different address spaces, the parameters used in a function (procedure) call need to be converted, otherwise the values of those parameters could not be used, because pointers to parameters in one computer's memory would point to different data on the other computer and also the client and server may also use different data representations. Stub libraries must be installed on both the client and server side.

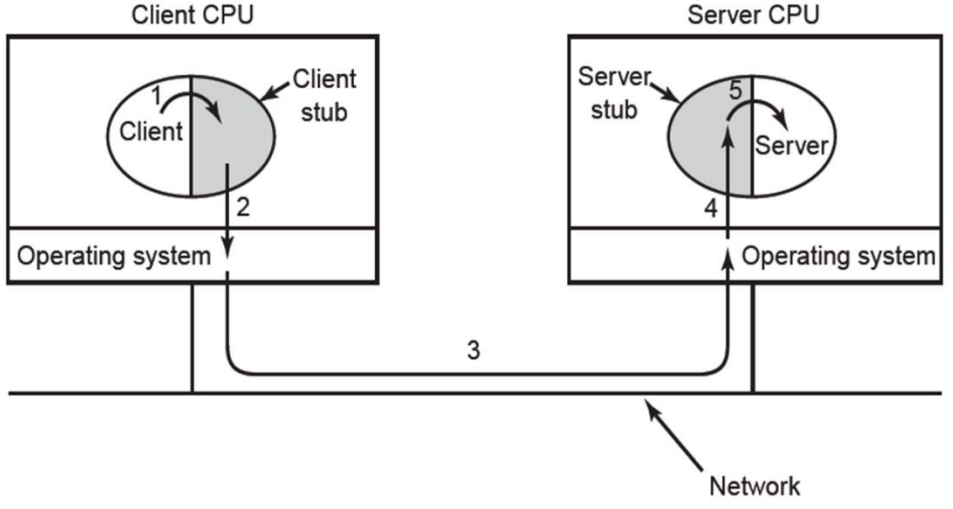
Transport Layer

Remote Procedure Call (RPC)

Client Stub: A small library procedure on client to call a remote procedure, which represents, the server procedure in the client's address space.

Server Stub: a piece of code on the server side, is responsible for de-conversion of parameters passed by the client and conversion of the results after the execution of the function.





Transport Layer

Remote Procedure Call (RPC)

Steps involved in RPC

1. A client invokes a *client stub* procedure, passing parameters in the usual way. The client stub resides within the client's own address space.
2. The client stub *marshalls* the parameters into a message. Marshalling includes converting the representation of the parameters into a standard format, and copying each parameter into the message.
3. The client stub passes the message to the transport layer, which sends it to the remote server machine.
4. On the server, the transport layer passes the message to a *server stub*, which demarshalls the parameters and calls the desired server routine using the regular procedure call mechanism.
5. When the server procedure completes, it returns to the server stub (e.g., via a normal procedure call return), which marshalls the return values into a message. The server stub then hands the message to the transport layer.
6. The transport layer sends the result message back to the client transport layer, which hands the message back to the client stub.
7. The client stub demarshalls the return parameters and execution returns to the caller.

Transport Layer

Internet Transport Protocol

Transmission Control Protocol(TCP Service Model)

- Provides a reliable end-to-end byte stream over an unreliable internetwork, TCP dynamically adapt to properties of the internetwork which may differ in topologies, bandwidths, delays, packet sizes, and other parameters.
- TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. Major internet applications such as the World Wide Web, email, remote administration, and file transfer rely on TCP
- All TCP connections are full duplex and point-to-point. each connection has exactly two end points. TCP does not support multicasting or broadcasting.
- TCP service is obtained by both the sender and the receiver creating end points, called **sockets.**
- Each socket has a socket number (address) consisting of the **IP address** of the host and a **16-bit number** local to that host, called a **port**.
- Port numbers below 1024 are reserved for standard services they are called **Well-known ports.**
- Other ports from 1024 through 49151 can be use by unprivileged users,

Transport Layer

Internet Transport Protocol

Transmission Control Protocol(TCP Service Model)

Some assigned ports

Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

Services

1. **Process-to-Process communication**
2. **Stream oriented**
3. **Full duplex service**
4. **Connection oriented service**
5. **Reliability**
6. **Multiplexing**

Transport Layer

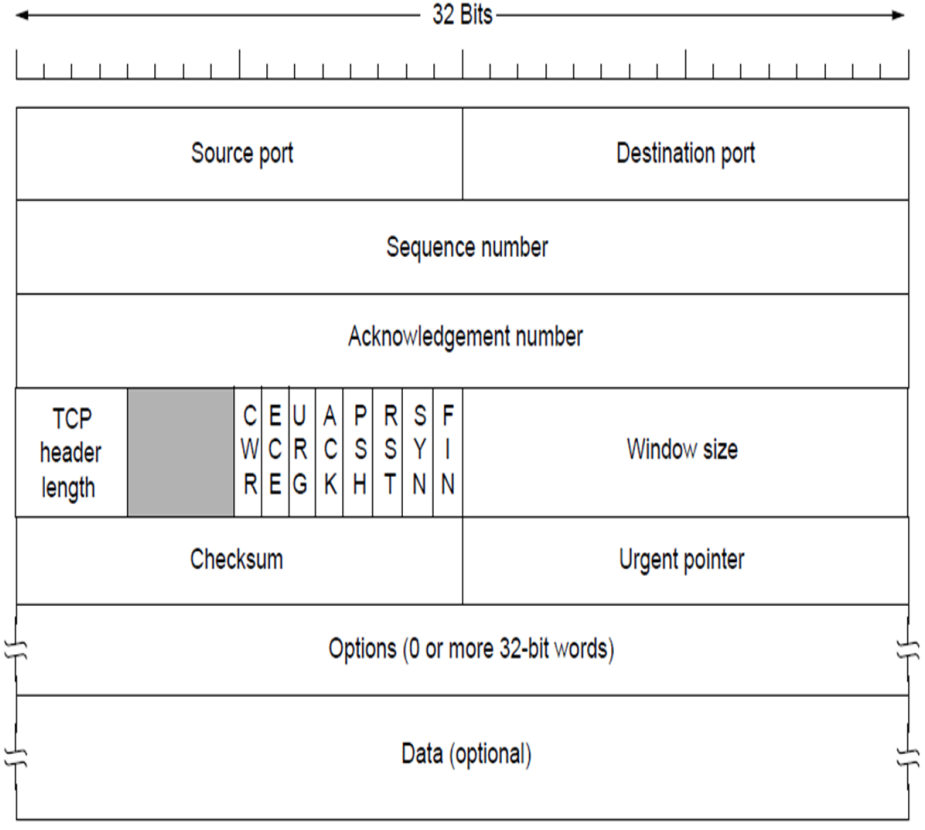
Internet Transport Protocol

TCP Protocol

- A TCP connection is a byte stream, not a message stream. Every byte on a TCP connection has its own 32-bit sequence number.
- The sending and receiving TCP entities exchange data in the form of **Segments**.
- A TCP segment consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes.
- The TCP software decides how big segments should be, two limits restrict the segment size.
- First, each segment, including the TCP header, must fit in the 65,515- byte IP payload.
- Second, each link has an MTU (Maximum Transfer Unit). Each segment must fit in the MTU at the sender and receiver so that it can be sent and received in a single, un-fragmented packet. In practice, the MTU is generally 1500 bytes (Ethernet Pay load size)
- However, it is still possible for IP packets carrying TCP segments to be fragmented when passing over a network path for which some link has a small MTU

Transport Layer

Internet Transport Protocol
TCP Segment Header



The initial 5 rows of the TCP header are always used. So, minimum length of TCP header = 5 x 4 bytes = 20 bytes. The size of the 6th row representing the Options field vary. The size of Options field can go up to 40 bytes.
So, maximum length of TCP header = 20 bytes + 40 bytes = 60 bytes.

Source Port-it is a 16 bit field that identifies the port of the sending application.
Destination Port- it is a 16 bit field, that identifies the port of the receiving application.

Transport Layer

Internet Transport Protocol
TCP Segment Header

Sequence Number- it is a 32 bit field. TCP assigns a unique sequence number to each byte of data contained in the TCP segment. This field contains the sequence number of the first data byte.

Acknowledgement Number- it is a 32 bit field, it contains sequence number of the data byte that receiver expects to receive next from the sender. it is always sequence number of the last received data byte incremented by 1.

Header Length- it is a 4 bit field. It contains the length of TCP header. It helps in knowing from where the actual data begins. So, the range of decimal values that can be represented is [0, 15]. But the range of header length is [20, 60]

Reserved Bits- The 6 bits are reserved, these bits are not used.

URG bit - When this bit is set to 1, it indicates the receiver that certain amount of data within the current segment is urgent. Urgent data is pointed out by evaluating the urgent pointer field. The urgent data has be prioritized. Receiver forwards urgent data to the receiving application on a separate channel.

ACK bit- When this bit is set to 1, it indicates that acknowledgement number contained in the TCP header is valid, for all TCP segments except request segment, ACK bit is set to 1.

Request segment is sent for connection establishment during Three Way Handshake.

PSH bit- When this bit is set to 1, all the segments in the buffer are immediately pushed to the receiving application. No wait is done for filling the entire buffer and makes the entire buffer to free up immediately. But It is not a good practice to set PSH bit = 1, because it disrupts the working of receiver's CPU and forces it to take an action immediately.

RST bit- When this bit is set to 1, it indicates the receiver to terminate the connection immediately.
It causes both the sides to release the connection and all its resources abnormally, the transfer of data ceases in both the directions. It may result in the loss of data that is in transit.
Used when there are unrecoverable errors and there is no chance of terminating the TCP connection normally.

SYN bit- When SYN bit is set to 1, it indicates the receiver that the sequence number contained in the TCP header is the initial sequence number.
Request segment sent for connection establishment during Three way handshake contains SYN bit set to 1.
FIN bit- When FIN bit is set to 1, It indicates the receiver that the sender wants to terminate the connection. FIN segment sent for TCP Connection Termination contains FIN bit set to 1.
CWR bit - The congestion window reduced flag is used by the sending host to indicate it received a packet with the ECE flag set
Window Size- it is a 16 bit field, it contains the size of the receiving window of the sender. It advertises how much data (in bytes) the sender can receive without acknowledgement. Thus, window size is used for Flow Control.

Checksum- it is a 16 bit field used for error control. It verifies the integrity of data in the TCP payload. Sender adds CRC checksum to the checksum field before sending the data. Receiver rejects the data that fails the CRC check.

Urgent Pointer- it is a 16 bit field, it indicates how much data in the current segment counting from the first data byte is urgent. Urgent pointer added to the sequence number indicates the end of urgent data byte. This field is considered valid and evaluated only if the URG bit is set to 1.

Options- this field is used for several purposes. The size of options field vary from 0 bytes to 40 bytes. Options field is generally used for the following purposes-

- Time stamp,
- Window size extension,
- Parameter negotiation,
- Padding

Transport Layer

Internet Transport Protocol
TCP Connection Establishment

TCP by means of the three-way handshaking for connection establishment

Case-1: To establish a connection,, the server may passively waits for an incoming connection by executing the LISTEN and ACCEPT primitives, either specifying a specific source or nobody in particular

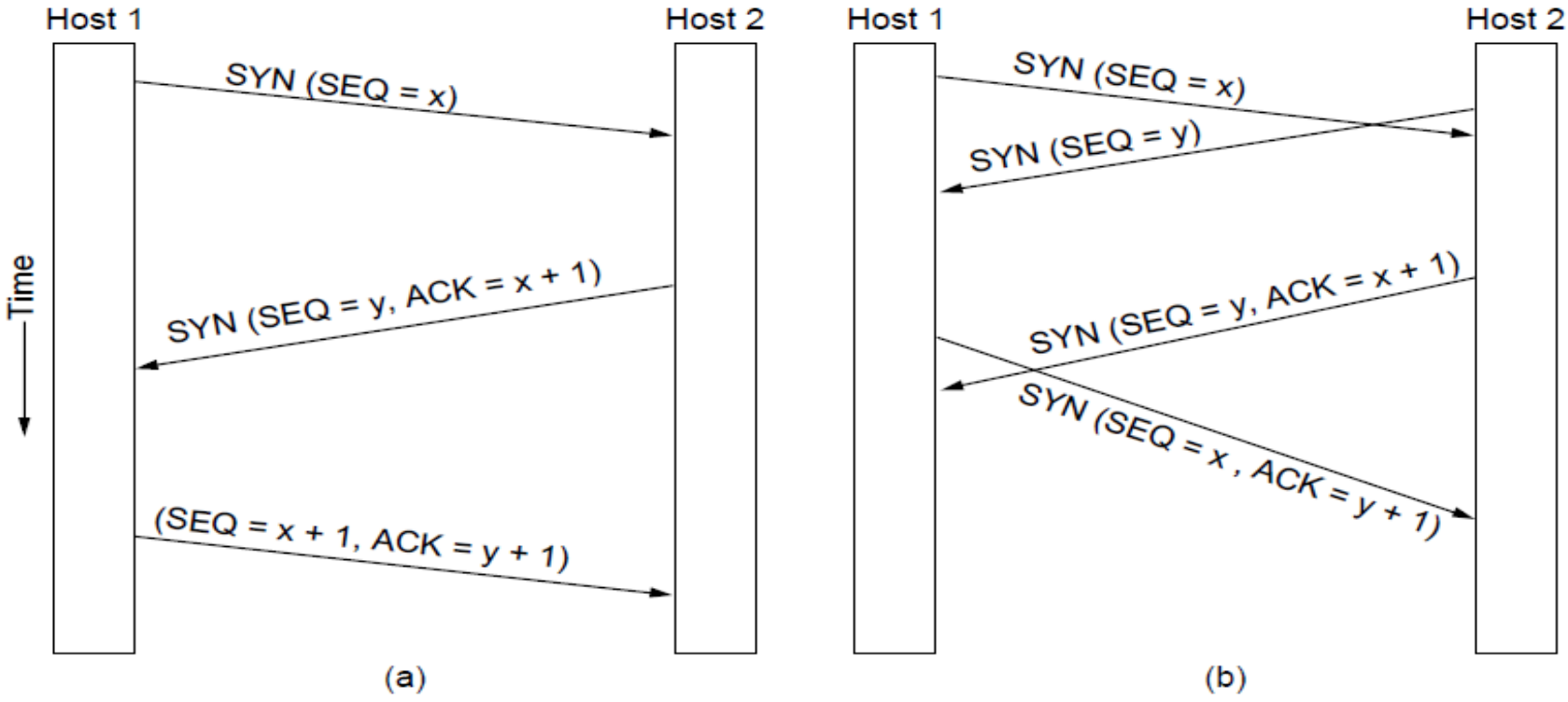
Case-2: The client, executes a CONNECT primitive, specifying the IP address and port to which it wants to connect, the maximum TCP segment size it is willing to accept, and optionally some user data (e.g., a password).

The CONNECT primitive sends a TCP segment with the SYN bit **ON** and ACK bit **OFF** and waits for a response. At server (destination) when this segment arrives, the TCP entity checks to see if there is a process that has done a LISTEN on the port given in the Destination port field. If not, it sends a reply with the RST bit on to reject the connection

Transport Layer

Internet Transport Protocol

TCP Connection Establishment



Transport Layer

TCP Connection Establishment

Steps

Step-01: SYN: Client sends a request segment to the server. Request segment consists only of TCP Header with random initial sequence number, SYN=1, max segment size and receiving window size, with an empty payload. Then, it waits for a reply segment from the server.

Step-02: SYN + ACK: After receiving the request segment, Server responds to the client by sending the reply segment having initial sequence number, SYN=1, Maximum segment size, Receiving window size, Acknowledgment number and ACK=1

Step-03: ACK- After receiving the reply segment, the client acknowledges the response of server by sending a pure acknowledgement.

Note : In any TCP segment,
a) If SYN bit = 1 and ACK bit = 0 then Request segment. b) If SYN bit = 1 and ACK bit = 1, then reply segment.
c) If SYN bit = 0 and ACK bit = 1, then pure ACK or segment meant for data transfer.
d) If SYN bit = 0 and ACK bit = 0, then this combination is not possible.

Transport Layer

TCP Connection Release

Steps

Step-01: For terminating the connection, Client sends a FIN segment to the server with FIN bit=1 and enters the FIN_WAIT_1 state, waits for an acknowledgement from the server.

Step-02: After receiving the FIN segment, Server frees up its buffers, sends an acknowledgement to the client and enters the CLOSE_WAIT state.

Step-03: After receiving the acknowledgement, client enters the FIN_WAIT_2 state.

Now, the connection from client to server is terminated, Client can not send any data to the server since server has released its buffers. But the connection from server to client is still open and Server can send both data and acknowledgements to the client.

Step-04: For terminating the connection, Server sends a FIN segment to the client with FIN =1. Server waits for an acknowledgement from the client.

Step-05: After receiving the FIN segment, Client frees up its buffers, sends an acknowledgement to the server (not mandatory) and enters the TIME_WAIT state

TIME_WAIT State- this state allows the client to resend the final acknowledgement if it gets lost. The time spent by the client in TIME_WAIT state depends on the implementation.

Transport Layer

TCP Connection Management

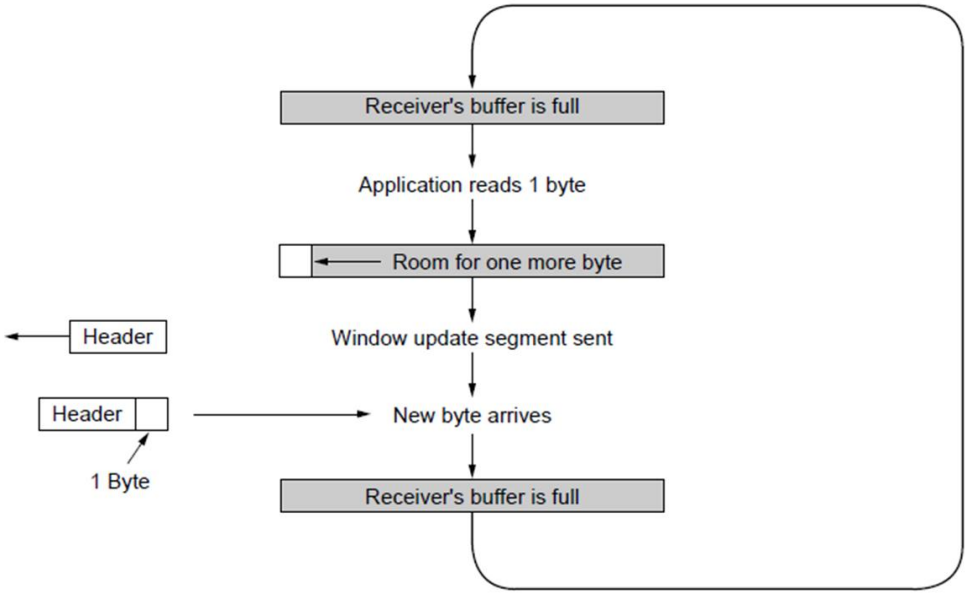
State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIME WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

The states used in the TCP connection management

Transport Layer

Silly window syndrome in Flow control

The problem that can degrade TCP performance is the Silly window syndrome



Silly window syndrome

This problem occurs when data are passed to the sending TCP entity in large blocks, but an interactive application on the receiving side reads data only 1 byte at a time.

Initially, the TCP buffer on the receiving side is full (window of size=0) and the sender knows this. Then the interactive application reads one character from the TCP stream. And receiver sends a window update to the sender saying that it is all right to send 1 byte. The sender obliges and sends 1 byte. The buffer is now full, so the receiver acknowledges the 1-byte segment and sets the window to 0.

Clark's solution is to prevent the receiver from sending a window update for 1 byte. Instead, it is forced to wait until it can handle the maximum segment size it advertised when the connection was established or until its buffer is half empty, whichever is smaller.

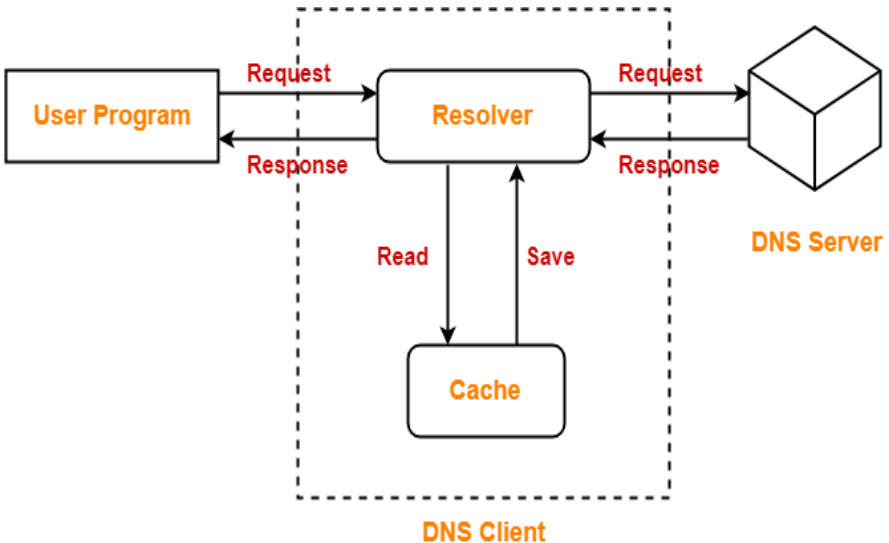
Application Layer

Domain Name System (DNS)

DNS is short for Domain Name Service or Domain Name System. It is an application layer protocol.

It converts the names we type in our web browser address bar to the IP Address of web servers hosting those sites. DNS is a directory service that provides a mapping between the name of a host on the network and its numerical address.

IP Addresses are a complex series of numbers. So, it is difficult to remember IP Addresses directly while it is easy to remember names.



Step-01: A user program sends a name query to a library procedure called the resolver.

Step-02: Resolver looks up the local domain name cache for a match. If a match is found, it sends the corresponding IP Address back. If no match is found, it sends a query to the local DNS server.

Step-03: DNS server looks up the name. If a match is found, it returns the corresponding IP Address to the resolver. If no match is found, the local DNS server sends a query to a higher level DNS server. This process is continued until a result is returned.

Application Layer

Domain Name System (DNS)

Uniform Resource Locator (URL) refers to a web address which uniquely identifies a document over the internet.

Domain Names: Domain Name is a symbolic string associated with an IP address. There are several domain names available; some of them are generic such as com, edu, gov, net etc, while some country level domain names such as au, in, za, us etc.

Domain Name System Architecture: The Domain name system comprises of Domain Names, Domain Name Space, Name Server.

Domain Name Space : The domain name space refers a hierarchy in the internet naming structure. This hierarchy has multiple levels (from 0 to 127), with a root at the top

For the Internet, the top of the naming hierarchy is managed by an organization called **ICANN** (Internet Corporation for Assigned Names and Numbers).
Conceptually, the Internet is divided into over **250** top-level domains, where each domain covers many hosts.
Each domain is partitioned into subdomains, and these are further partitioned, and so on.
Top-level domains are run by registrars appointed by ICANN
Each domain is named by the path upward from it to the root. The components are separated by dot. Thus, the engineering department at Cisco might be ***eng.cisco.com.***

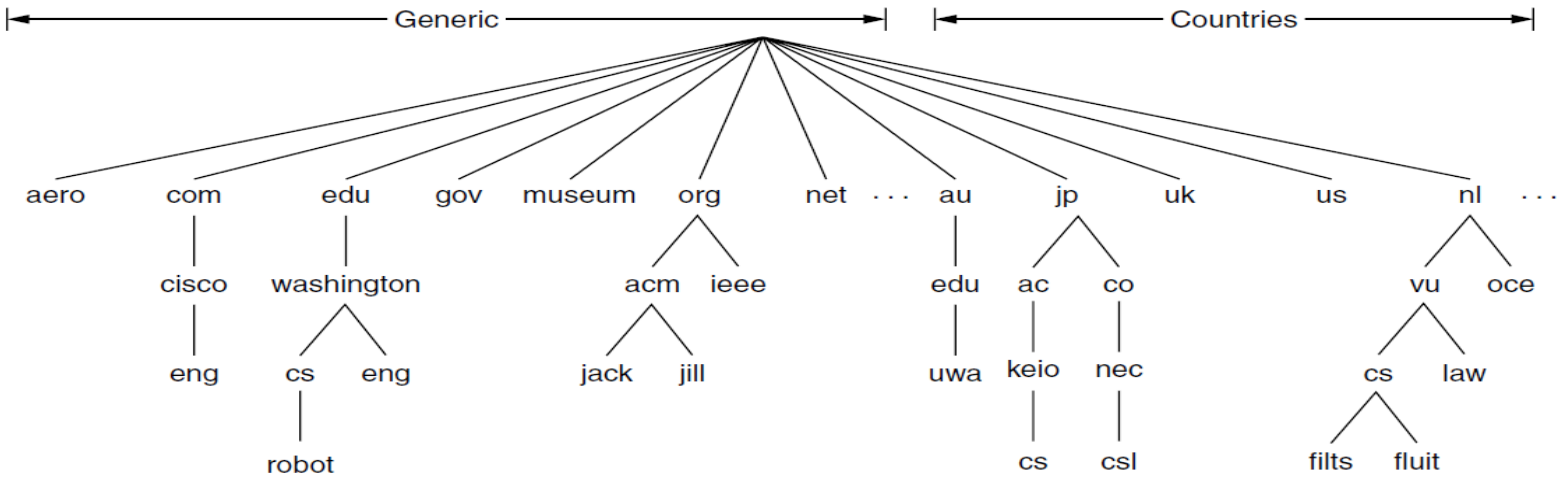
Application Layer

DNS Name Space

Host names are divided into several pieces called domains. Domains are designed in a hierarchical structure. The top-level domains refer to the type of organization to which the network belongs, and subdomains further identify the specific network on which the host is situated. The domain name space is a hierarchical tree structure.

The top-level domains come in two flavors: **Generic** and **Countries.**

In postal system the hierarchical addressing manages the address with Country→State→City→Street→DoorNo→Name.



Generic Top-Level Domain names:

Domain Name	Meaning
Com	Commercial business
Edu	Education
Gov	U.S. government agency
Int	International entity
Mil	U.S. military
Net	Networking organization
Org	Non profit organization

Country top-level domain names:

Domain Name	Meaning
au	Australia
in	India
cl	Chile
fr	France
us	United States
za	South Africa
uk	United Kingdom
jp	Japan
es	Spain
de	Germany
ca	Canada
ee	Estonia
hk	Hong Kong

The process of looking up a name and finding an IP address is called **Name Resolution**. When a resolver has a query about a domain name, it passes the query to a local name server

The below figure name resolution of ***student.cse.dtu.in***