Unit -4 Process-to-Process Delivery UDP, TCP, and SCTP

INTRODUCTION

- The transport layer is responsible for the delivery of a message from one process to another

- The transport layer header must include a service – point –address in the OSI model or port number in the TCP/IP (internet model)
- The Internet model has three protocols at the transport layer: UDP, TCP, and SCTP.
  - **UDP**: Is the simplest of the three.
  - **TCP**: A complex transport layer protocol.
  - **SCTP**: The new transport layer protocol that is designed for specific applications such as multimedia. a new reliable, message-oriented transport layer protocol that combines the best features of UDP and TCP

PROCESS-TO-PROCESS DELIVERY

- The Data link layer is responsible for delivery of frames between nodes over a link node to node delivery using a MAC address to choose one node among several.

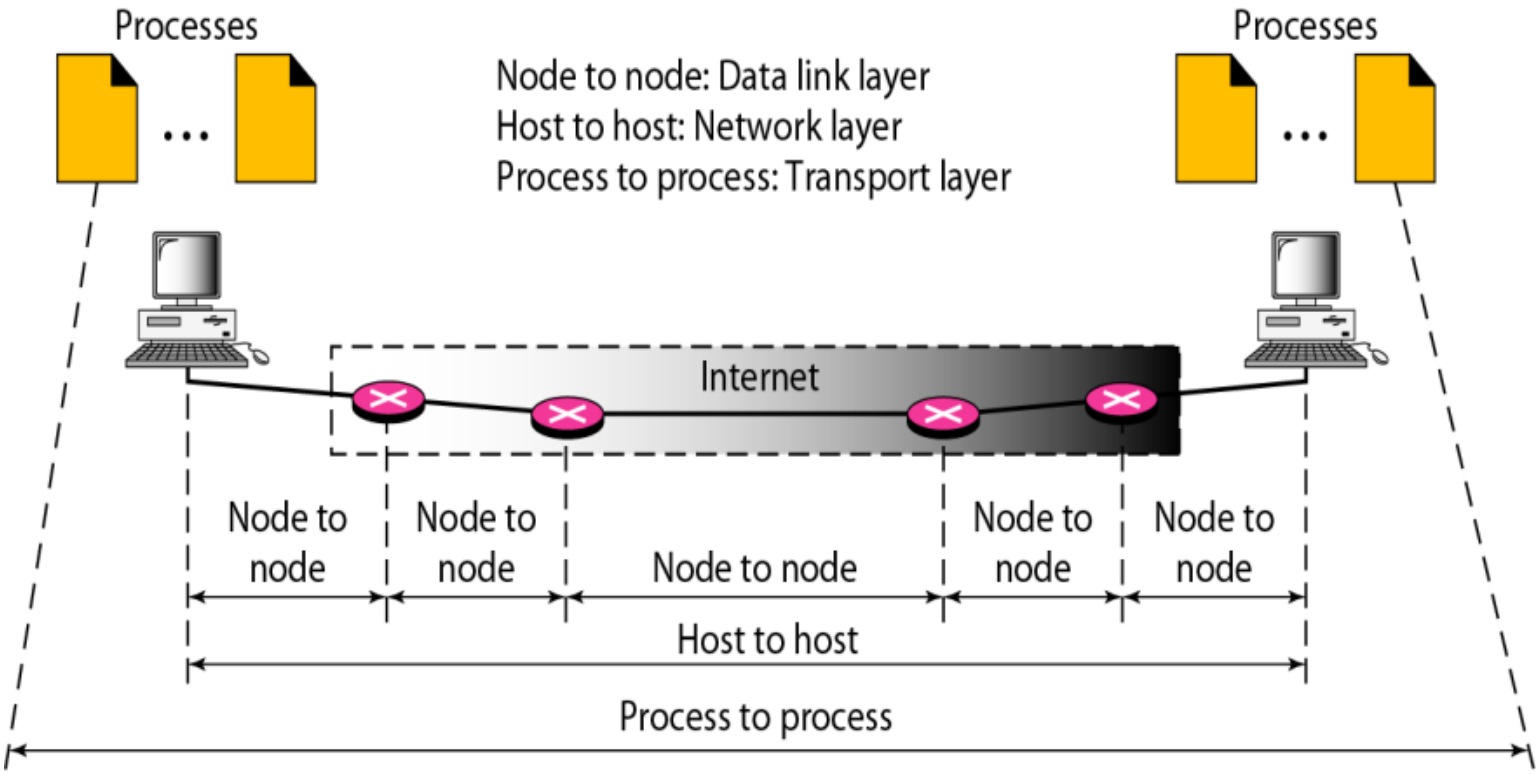
- The Network layer is responsible for delivery of datagrams between two hosts host to host delivery using an IP address to choose one host among millions.

- Real communication takes place between two processes (application programs). We need process-to-process delivery.

- We need a mechanism to deliver data from one of process running on the source host to the corresponding process running on the destination host.

- The Transport layer is responsible for process-to-process . We need a port number, to choose among multiple processes running on the destination host.

TYPES OF DATA DELIVERIES



CLIENT/SERVER PARADIGM

- process-to-process communication can be achieved through client/server

- A process on the local host, called a client, needs services from a process usually on the remote host, called a server.
  - Both processes (client and server) have the same name.
  - For example, to get the day and time from a remote machine, we need a Daytime client process running on the local host and a Daytime server process running on a remote machine.

- A remote computer can run several server programs at the same time, just as local computers can run one or more client programs at the same time.

PORT NUMBER

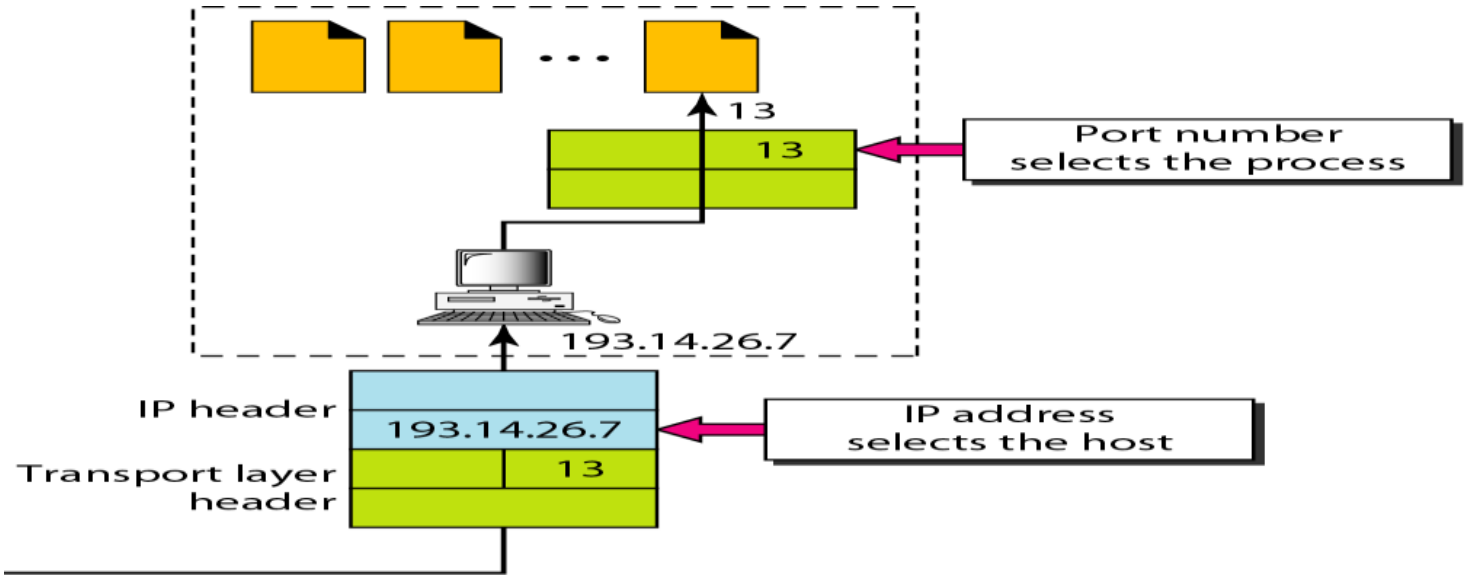
- In the Internet model, the port numbers are 16-bit integers between 0 and 65,535.
- The **client** program defines itself with a port number, chosen randomly by the transport layer software running on the client host
- The **server** process must also define itself with a port number This port number, however, cannot be chosen randomly
- The Internet uses port numbers for servers called well- known port numbers.
- Every client process knows the well-known port number of the œrresponding server process
- *For example*, while the Daytime client process, can use an ephemeral (temporary) port number 52,000 to identify itself, the Daytime server process must use the well-known (permanent) port number 13.

IP ADDRESSES VERSUS PORT NUMBERS

IP addresses and port numbers play different roles in selecting the final destination of data.

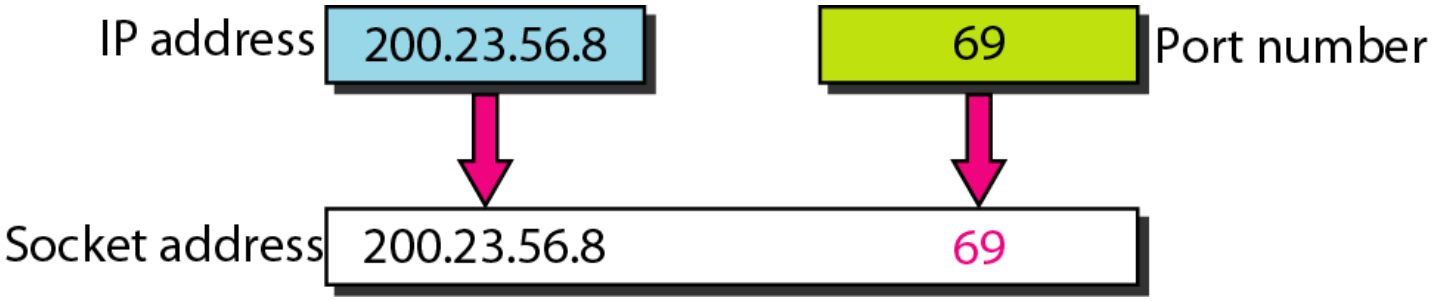
The destination IP address defines the host among the different hosts

After the host has been selected, the port number defines one of the processes on this particular host



SOCKET ADDRESSES

- 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.
- These four pieces of information are part of the IP header and the transport layer protocol header.
  - The IP header contains the IP addresses; the UDP or TCP header contains the port numbers.



MULTIPLEXING

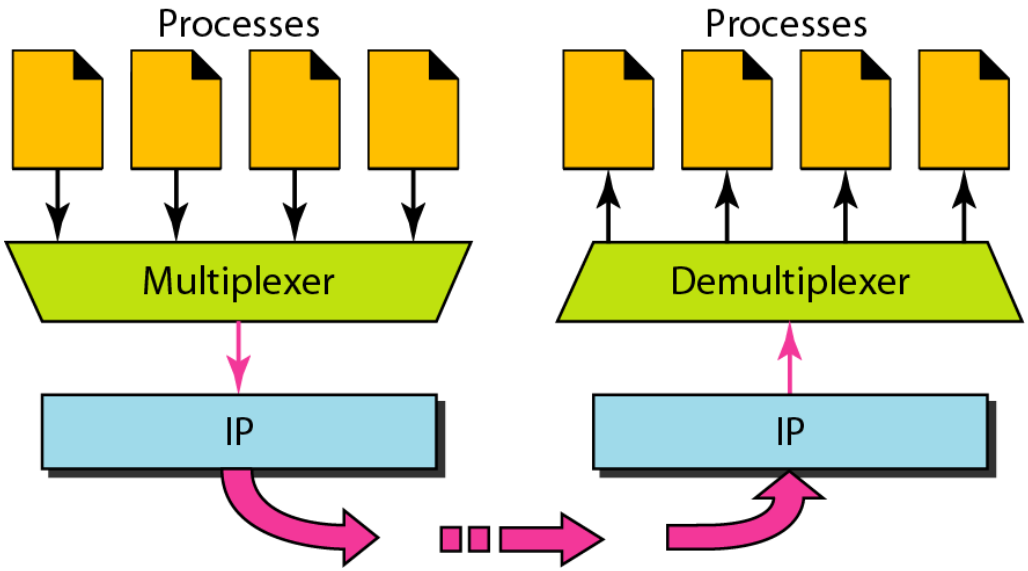
Upward Multiplexing

- One IP address shared by multiple Transport Layer processes (ports)

Downward Multiplexing

- Use multiple virtual circuits to get more bandwidth (e.g. join multiple ISDN lines to get a higher bandwidth)

MULTIPLEXING AND DEMULTIPLEXING



Sender: multiplexing of UDP datagrams.

UDP datagrams are received from multiple application programs.

A single sequence of UDP datagrams is passed to IP layer.

Receiver: demultiplexing of UDP datagrams.

Single sequence of UDP datagrams received from IP layer.

UDP datagram received is passed to appropriate application.

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; they may be delayed or lost or may arrive out of sequence.
  - There is no acknowledgment .

- 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. (virtual connection , not a physical connection)

RELIABLE VERSUS UNRELIABLE

- **The transport layer service can be reliable or unreliable.**
- If the application layer program needs reliability, we use a reliable transport layer protocol by implementing flow and error control at the transport layer. This means a slower and more complex service.
- On the other hand, if the application program does not need reliability then an unreliable protocol can be used.

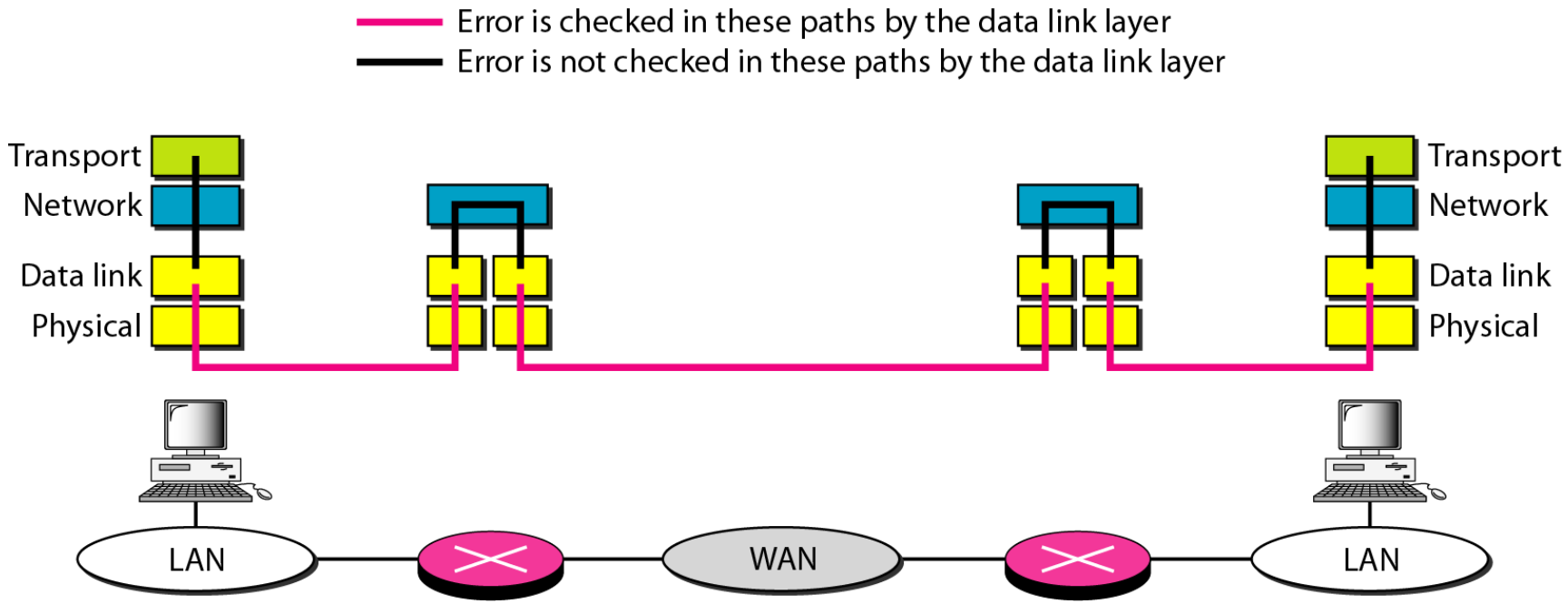
Note

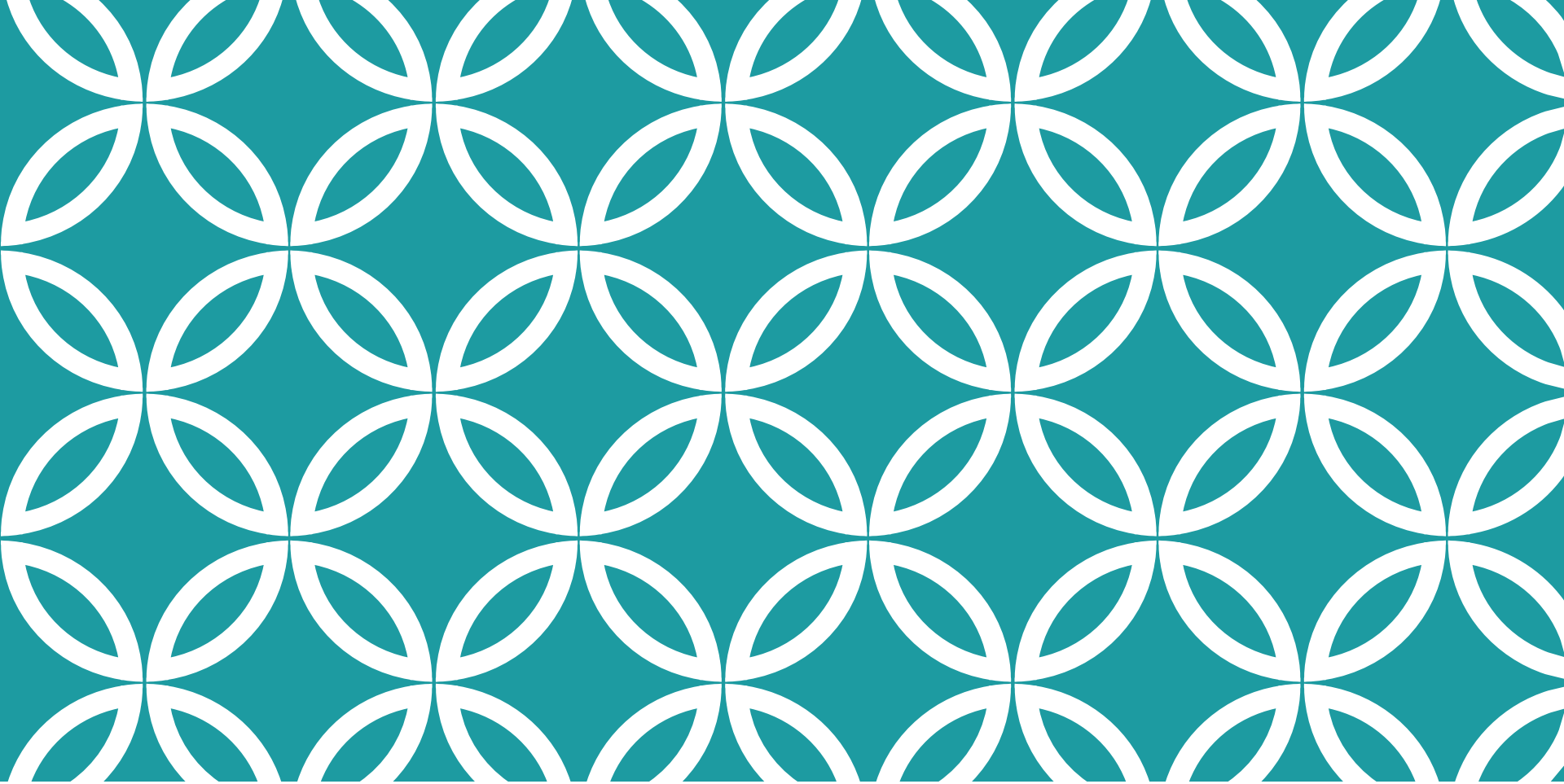
- UDP is connectionless and unreliable;
- TCP and SCTP are connection oriented and reliable.

These three protocols can respond to the demands of the application layer programs.

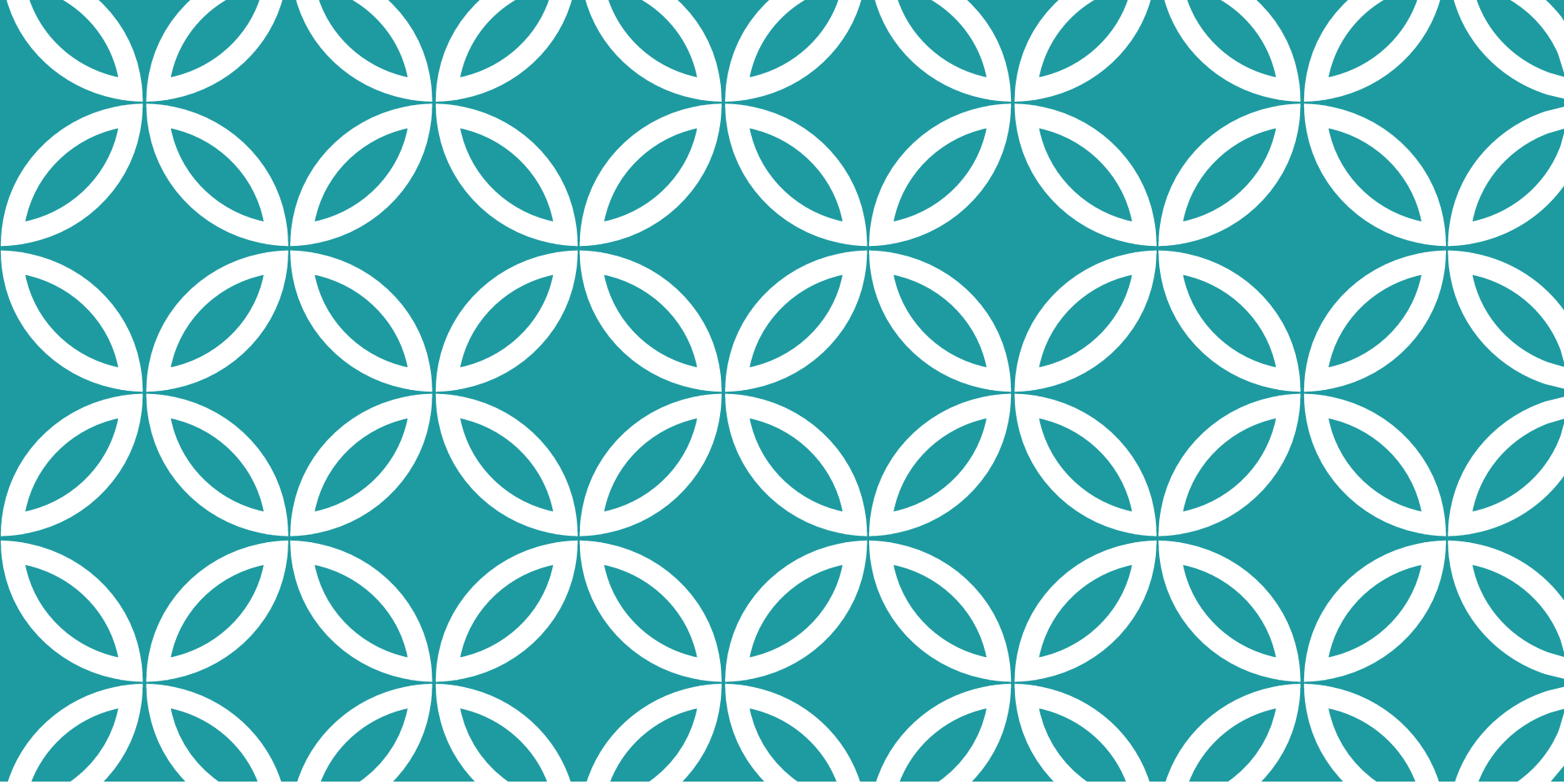
ERROR CONTROL

If the data link layer is reliable and has flow and error control, do we need this at the transport layer, too? Yes





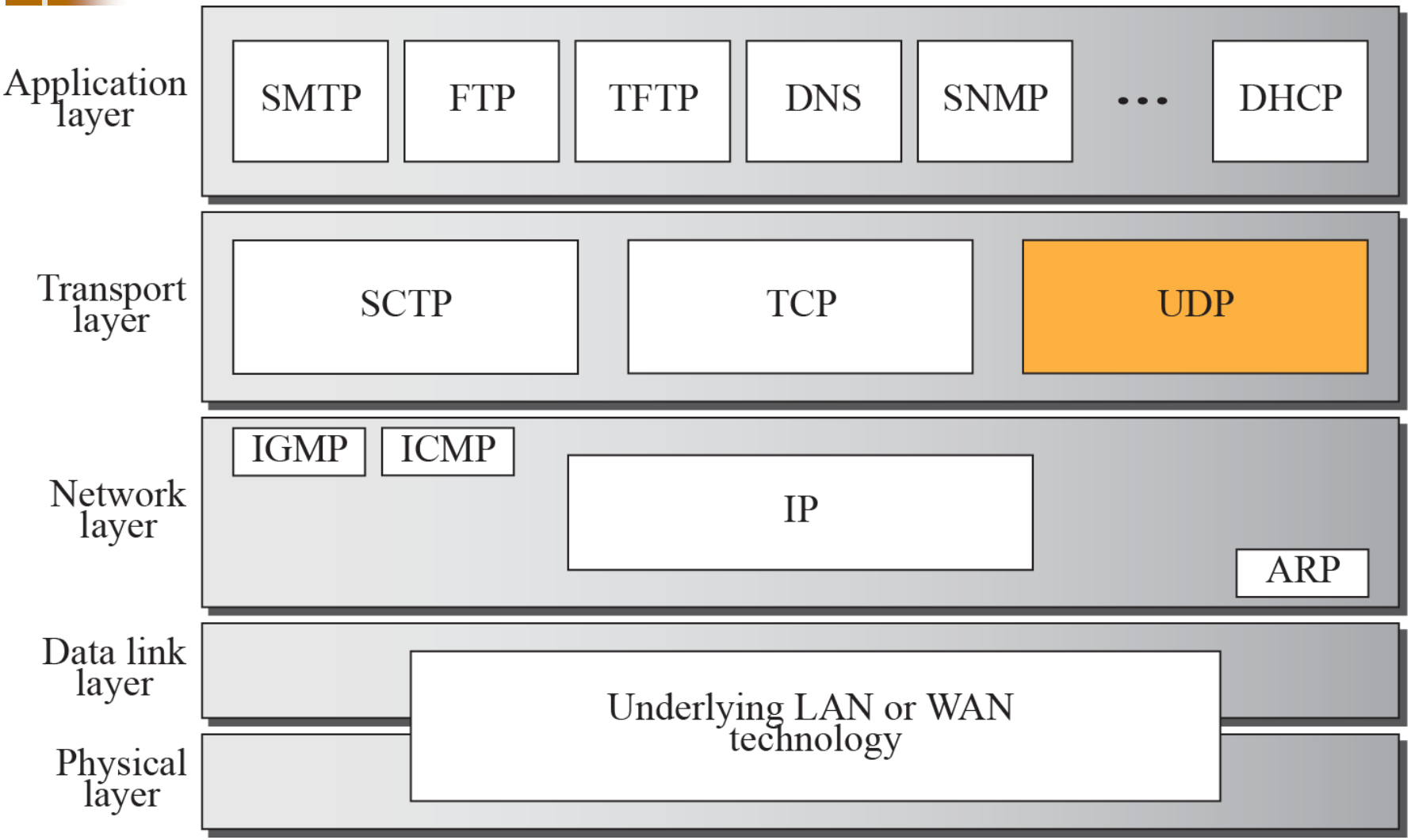
TRANSPORT LAYER PROTOCOLS



UDP

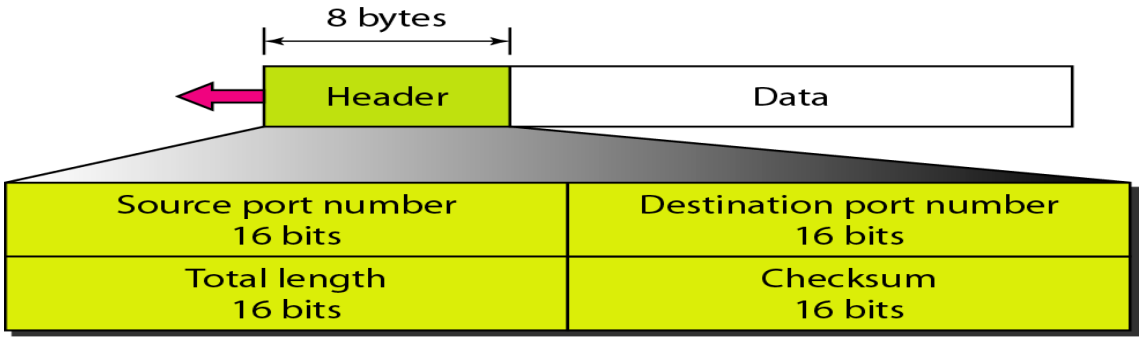
UDP is located between the application layer and the IP layer, and serves as the intermediary between the application programs and the network operations.

Figure 14.1 *Position of UDP in the TCP/IP protocol suite*



1. USER DATAGRAM PROTOCOL (UDP)

- UDP is a connectionless, unreliable transport protocol.
- It does not add anything to the services of IP (very simple) except to provide process-to process communication instead of host-to-host communication.
- (when to use it?) If a process wants to send a small message and does not care much about reliability, it can use UDP.



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

https://www.youtube.com/watch?v=bIV7WUZpkCE

UDP packets, called user datagrams, have a fixed size header of 8 bytes.

UDP CHARACTERISTICS

End-to-End: an application sends/receives data to/from another application.

Connectionless: Application does not need to preestablish communication before sending data; application does not need to terminate communication when finished.

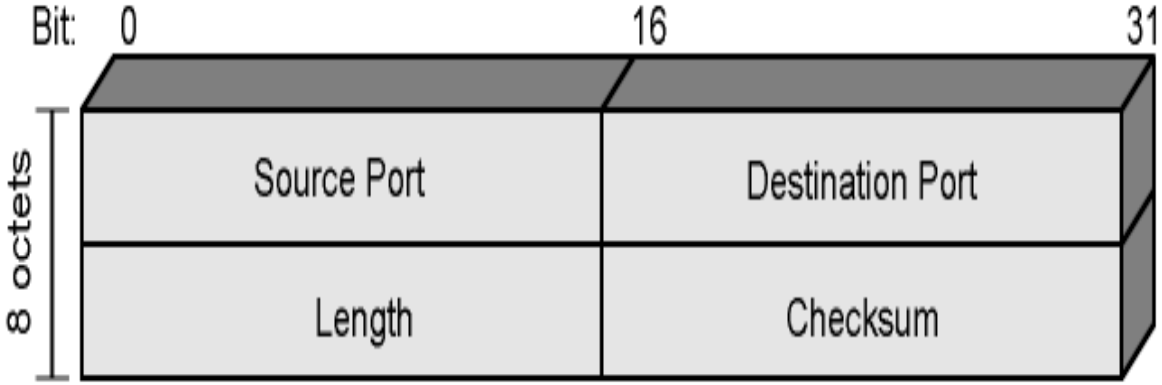
Message-oriented: application sends/receives individual messages (UDP datagram), not packets.

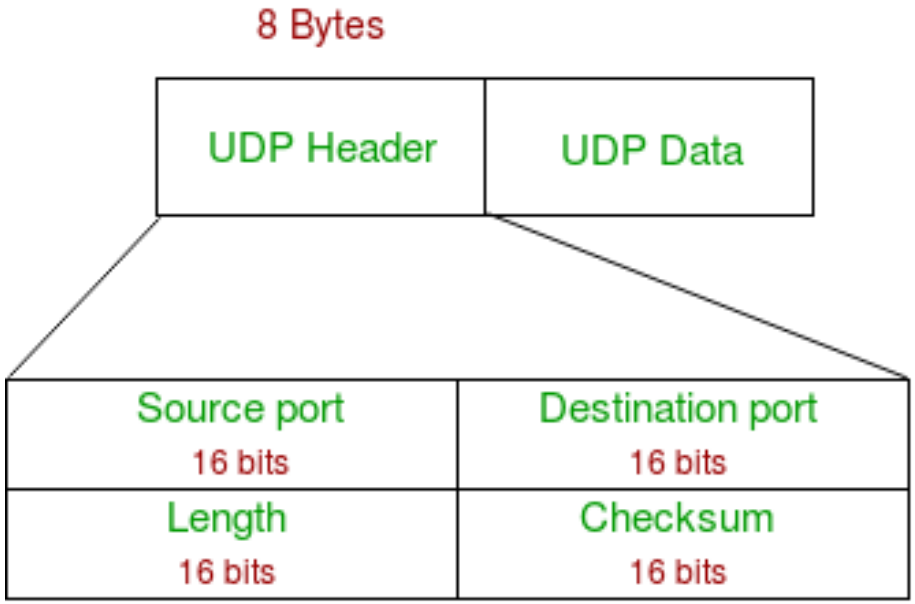
Best-effort: same best-effort delivery semantics as IP. I.e. message can be lost, duplicated, and corrupted.

Arbitrary interaction: application communicates with many or one other applications.

Operating system independent: identifying application does not depend on O/S.

UDP HEADER


Bit:
0
16
31
8 octets
Source Port
Destination Port
Length
Checksum


8 Bytes
UDP Header
UDP Data
Source port
16 bits
Destination port
16 bits
Length
16 bits
Checksum
16 bits

UDP HEADER

The UDP header consists of four fields each of 2 bytes in length:

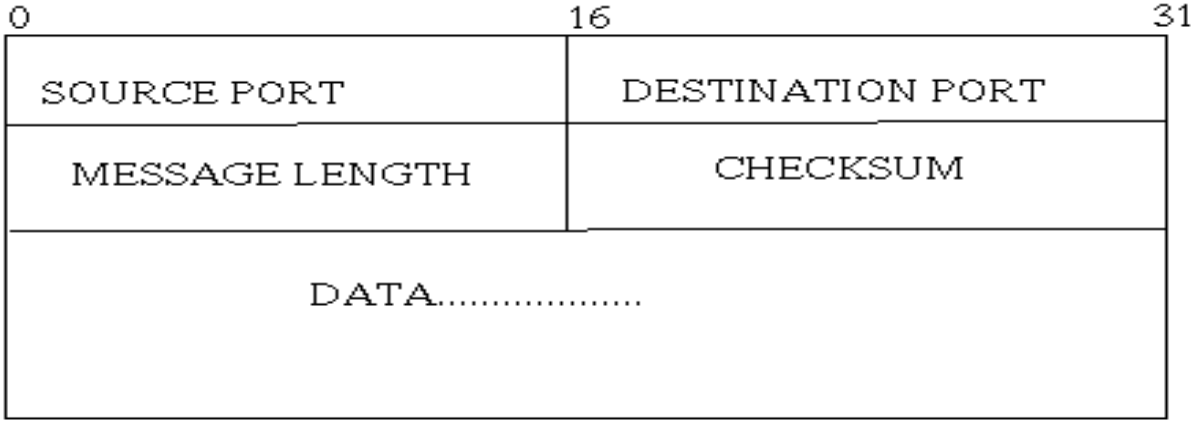
Source Port (UDP packets from a client use this as a service access point (SAP) to indicate the session on the local client that originated the packet. UDP packets from a server carry the server SAP in this field)

Destination Port (UDP packets from a client use this as a service access point (SAP) to indicate the service required from the remote server. UDP packets from a server carry the client SAP in this field)

UDP length (The number of bytes comprising the combined UDP header information and payload data)

UDP Checksum (A checksum to verify that the end to end data has not been corrupted by routers or bridges in the network or by the processing in an end system.

UDP DATAGRAM FORMAT



Source Port - 16 bit port number

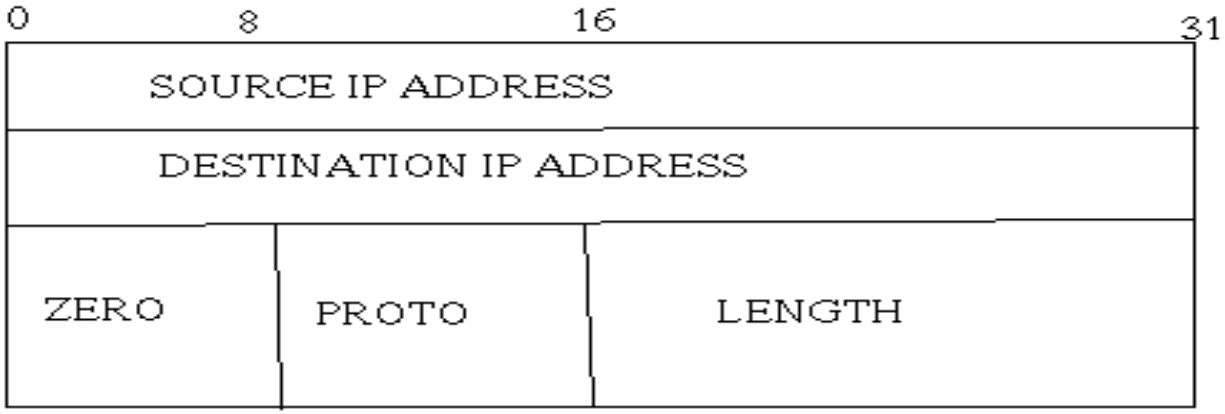
Destination Port - 16 bit port number

Length (of UDP header + data) - 16 bit count of octets

UDP checksum - 16 bit field. if 0, then there is no checksum, else it is a checksum over a pseudo header + UDP data area

CHECKSUM AND PSEUDO HEADER

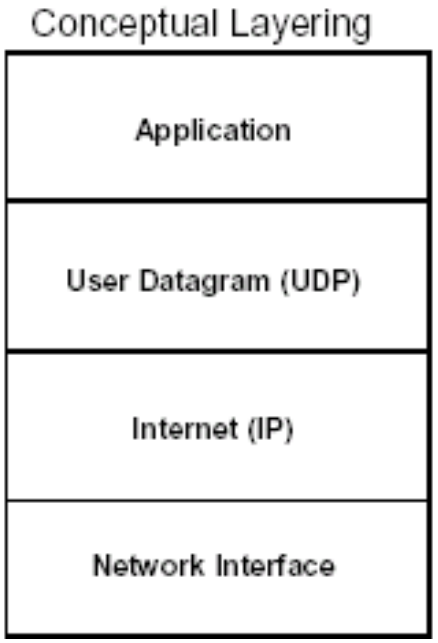
UDP uses a pseudo-header to verify that the UDP message has arrived at both the correct machine and the correct port.



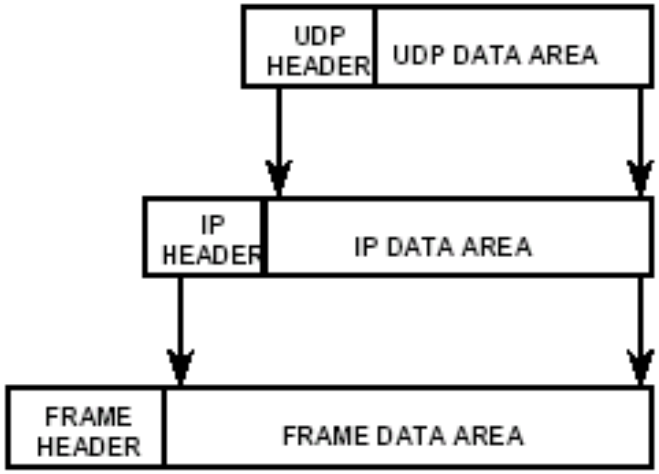
- Proto : IP protocol type code.
- Length : Length of the UDP datagram.

ENCAPSULATION AND LAYERING

- Protocol layering



- UDP encapsulation



UDP message is encapsulated into an IP datagram.

IP datagram in turn is encapsulated into a physical frame for actually delivery.

Example 14.1

The following is a dump of a UDP header in hexadecimal format.

CB84000D001C001C

a. What is the source port number?
b. What is the destination port number?
c. What is the total length of the user datagram?
d. What is the length of the data?
e. Is the packet directed from a client to a server or vice versa?
f. What is the client process?

Applications of UDP:

•Used for simple request response communication when size of data is less and hence there is lesser concern about flow and error control.

•It is suitable protocol for multicasting as UDP supports packet switching.

•UDP is used for some routing update protocols like RIP(Routing Information Protocol).

•Normally used for real time applications which can not tolerate uneven delays between sections of a received message.

•Following implementations uses UDP as a transport layer protocol:

- NTP (Network Time Protocol)
- DNS (Domain Name Service)
- BOOTP, DHCP.
- NNP (Network News Protocol)
- Quote of the day protocol
- TFTP, RTSP, RIP, OSPF.

•Application layer can do some of the tasks through UDP-

- Trace Route
- Record Route
- Time stamp

•UDP takes datagram from Network Layer, attach its header and send it to the user. So, it works fast.

•Actually UDP is null protocol if you remove checksum field.

2. TRANSMISSION CONTROL PROTOCOL(TCP)

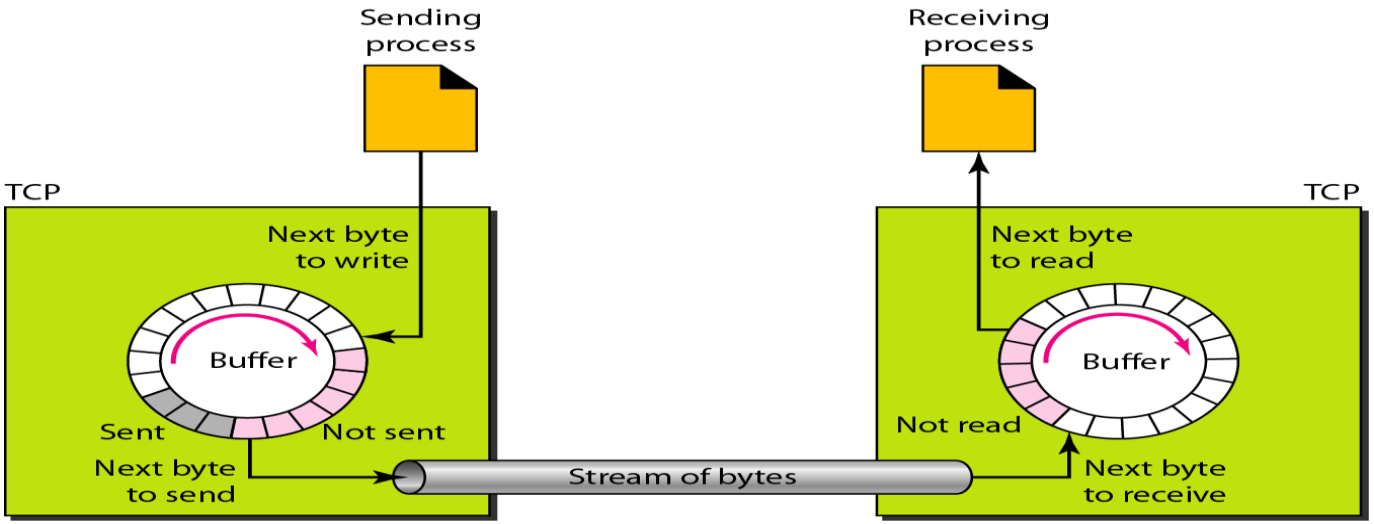
- TCP, like UDP, is a process-to-process (program-to-program) protocol uses port numbers.

- Unlike UDP, TCP is a connection oriented protocol; it creates a virtual connection between two TCPs to send data.

- TCP uses flow and error control mechanisms at the transport level.

2. TRANSMISSION CONTROL PROTOCOL(TCP)

- The sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage.
- Two buffers, the sending buffer and the receiving buffer, one for each direction.
- These buffers are also necessary for flow and error control mechanisms used by TCP.



A. TCP FEATURES

Numbering System:

There are two fields called the **sequence number** and the **acknowledgment number**.

These two fields refer to the **byte number** and **not the segment number.**

TCP numbers all data bytes that are transmitted in a connection.

Numbering is independent in each direction.

When TCP receives bytes of data from a process, it stores them in the **sending buffer** and **numbers them**.

The numbering does not necessarily start from 0.

TCP generates a random number between 0 and 2^{32} - 1 for the number of the first byte.

Byte numbering is used for **flow and error control**. For example: if the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056.

After the bytes have been numbered, TCP assigns a **sequence number to each segment** that is being sent.

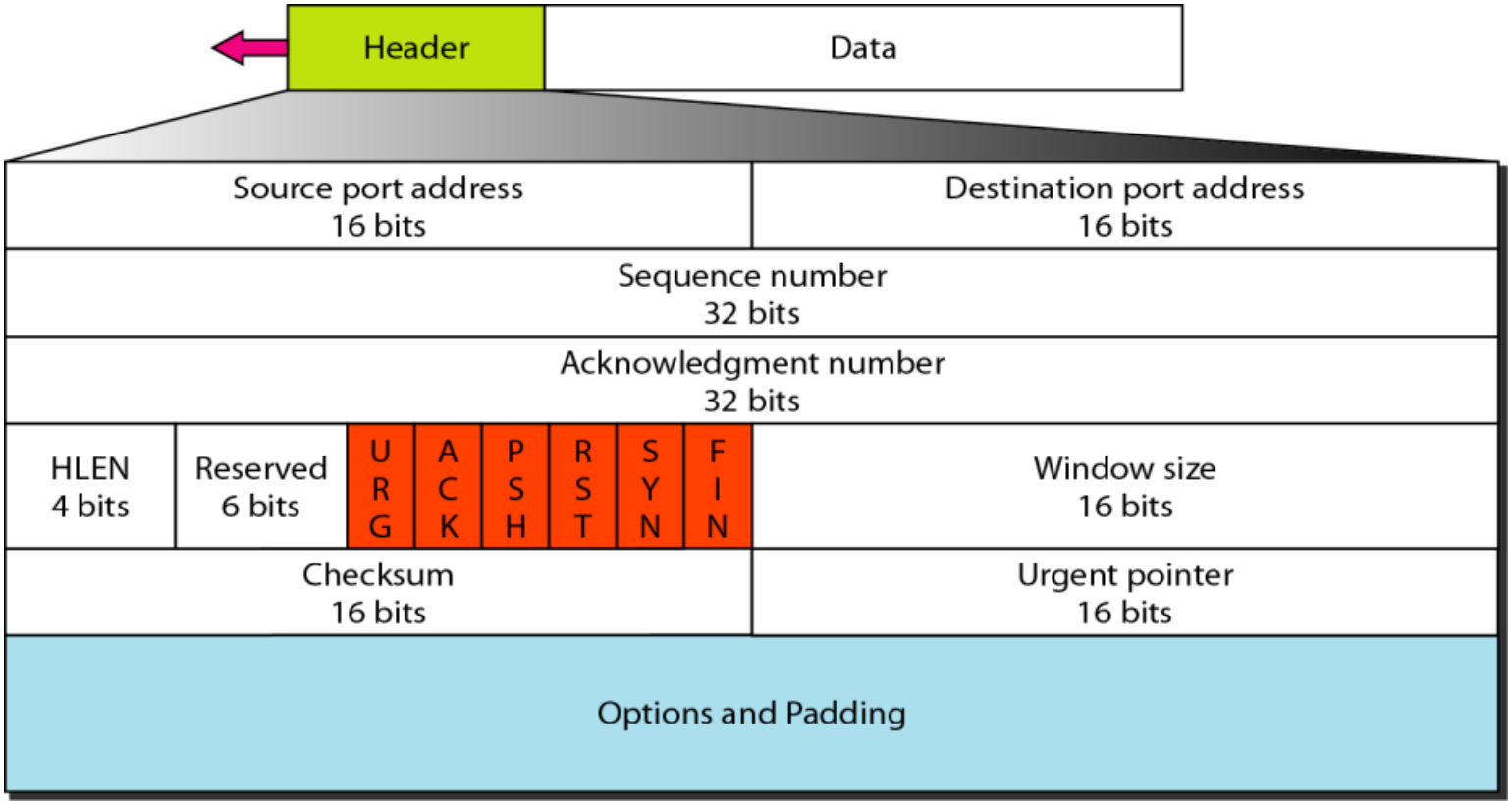
The **sequence number** for each segment is the **number of the first byte** carried in that segment

Acknowledgment Number

Each party uses an **acknowledgment number to confirm the bytes** it has received.

The acknowledgment number defines the number of the next byte that the party expects to receive.

B. TCP SEGMENT FORMAT



TCP Segment Format

The segment consists of a 20-60-byte header.

Source port address:

This is a 16-bit field , it 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, it 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.

Acknowledgment number: This 32 bit field defines the number of the next byte a party expects to receive.

Header length: A 4-bit field that 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).

TCP SEGMENT FORMAT

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. These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection

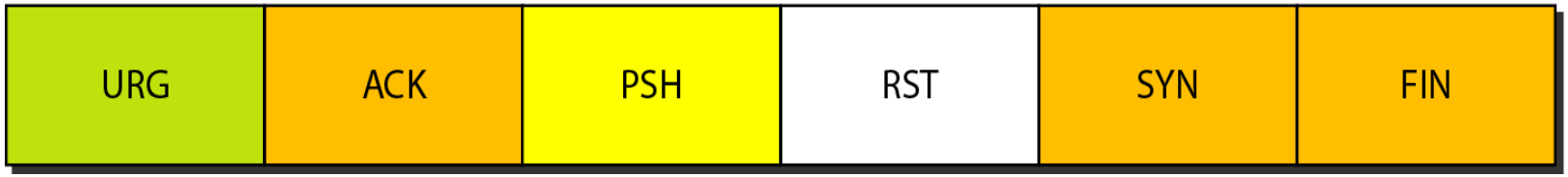


Table 12.2 Description of flags in the control field

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	The connection must be reset
SYN	Synchronize sequence numbers during connection
FIN	Terminate the connection

TCP SEGMENT FORMAT

Window size: Defines the size of the window, in bytes, that the other party must maintain. the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. 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. The inclusion of the checksum for TCP is mandatory.

Options: There can be up to 40 bytes of optional information in the TCP header.

TCP SEGMENT FORMAT

Urgent data :

- The data are presented from the application program to TCP as a stream of bytes.
- Each byte of data has a position in the stream.
- **If the sending application program wants a piece of data to be read out of order by the receiving application program., the sending TCP creates a segment and inserts the urgent data at the beginning of the segment.**
- The rest of the segment can contain normal data from the buffer
-

TCP SEGMENT FORMAT

Urgent data :

- The **urgent pointer field in the header** (if its set) defines the end of the urgent data and the start of normal data.

- When the receiving TCP receives a segment with the URG bit set, it extracts the urgent data from the segment using the value of the urgent pointer, and delivers them, out of order, to the receiving application program.

C. TCP CONNECTION

A Connection-oriented transport protocol establishes a virtual path between the source and destination.

In TCP, connection-oriented transmission requires three phases:

1. connection establishment
2. data transfer
3. connection termination

1) 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.

Each party must initialize communication and get approval from the other party before any data are transferred.

The connection establishment in TCP is called three way handshaking.

1) CONNECTION ESTABLISHMENT

Example:

Client-server communication using TCP as the transport layer protocol.

1.The server issues a request for a passive open: The server program tells its TCP that it is ready to accept a connection.

2.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

THREE-WAY HANDSHAKING PROCESS

1. 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

2. 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.

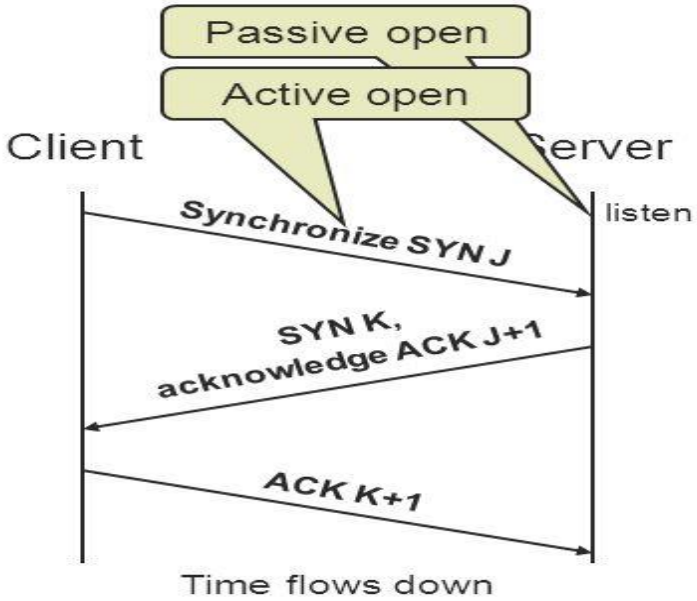
THREE-WAY HANDSHAKING PROCESS

3. 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.
- 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.

TCP Connection Establishment

- 3-Way Handshake
  - Sequence Numbers
    - J,K
  - Message Types
    - Synchronize (SYN)
    - Acknowledge (ACK)
  - Passive Open
    - Server listens for connection from client
  - Active Open
    - Client initiates connection to server



process 1 — process 2

passive OPEN,
waits for active request

Active OPEN
Send SYN, seq=n →

Receive SYN
Send SYN, seq=m, ACK n+1 →

Receive SYN+ACK
Send ACK m+1 →

The connection is now established and the two data streams (one in each direction) have been initialized (sequence numbers)

2) 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.

Example:

After connection is established,(shown before) the client sends 2000 bytes of data in two segments.

The server then sends 2000 bytes in one segment.

The client sends one more acknowledgment segment.

The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent.

3) CONNECTION TERMINATION

1. 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.

- A FIN segment can include the last chunk of data sent by the client, or it can be just a control segment
- If it is only a control segment, it consumes only one sequence number.

2. The server TCP, after receiving the FIN segment, 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.

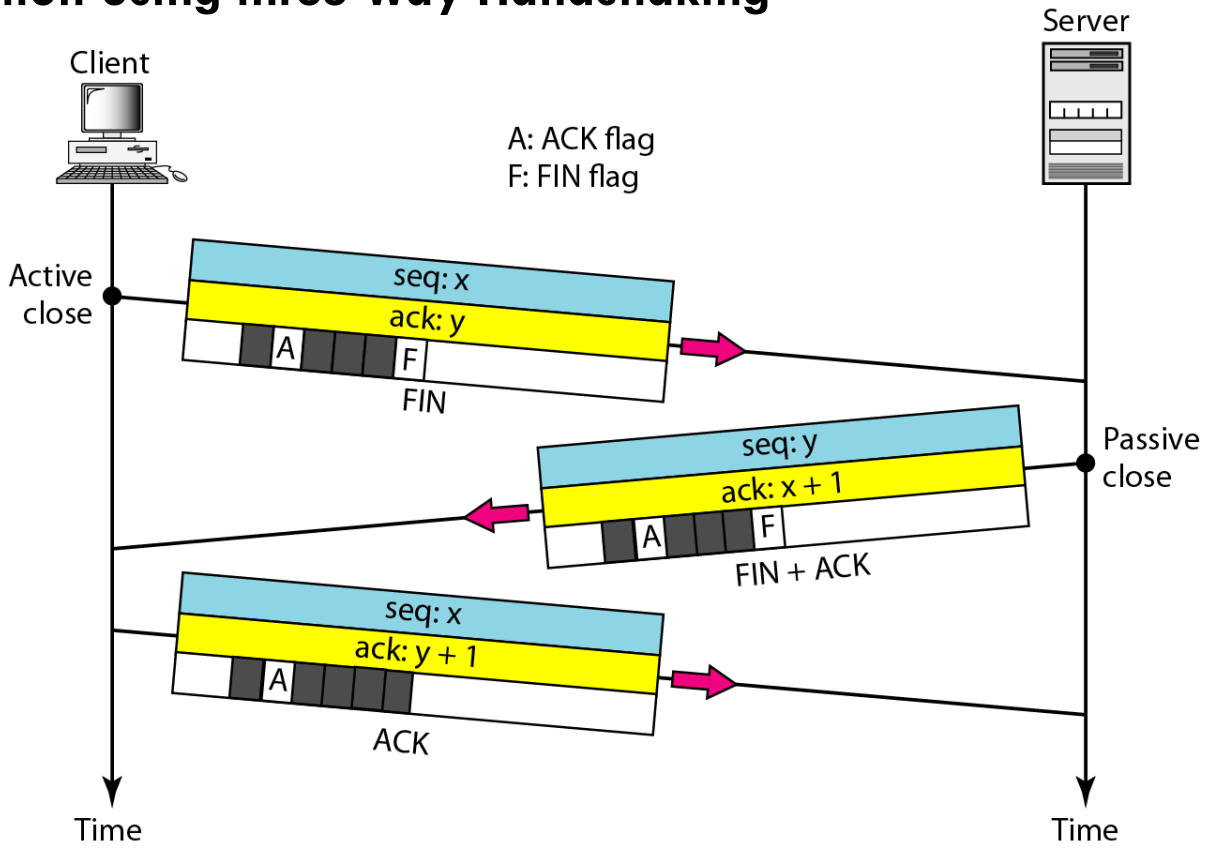
- This segment can also contain the last chunk of data from the server.
- If it does not carry data, it consumes only one sequence number.

3) CONNECTION TERMINATION

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.

Any of the two parties involved in exchanging data (client or server) can close the connection using three-Way Handshaking



D. FLOW CONTROL

TCP uses a sliding window to handle flow control.

TCP sliding window is of variable size.

The window is *opened, closed, or shrunk.*

- Opening a window means moving the right wall to the right.
- Closing the window means moving the left wall to the right.
- Shrinking the window means moving the right wall to the left.

The size of the window at one end is determined by the lesser of two values: *receiver window (rwnd) or congestion window (cwnd).*

1. CONGESTION

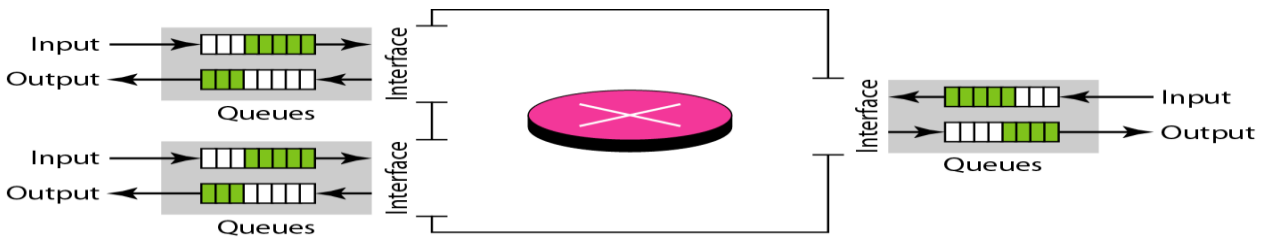
Congestion: the load on the network is greater than the capacity of the network

Congestion control: the mechanisms to control the congestion and keep the load below the capacity

Congestion occurs because routers and switches have queues- buffers that hold the packets before and after processing

The rate of packet arrival > packet processing time → input queue longer

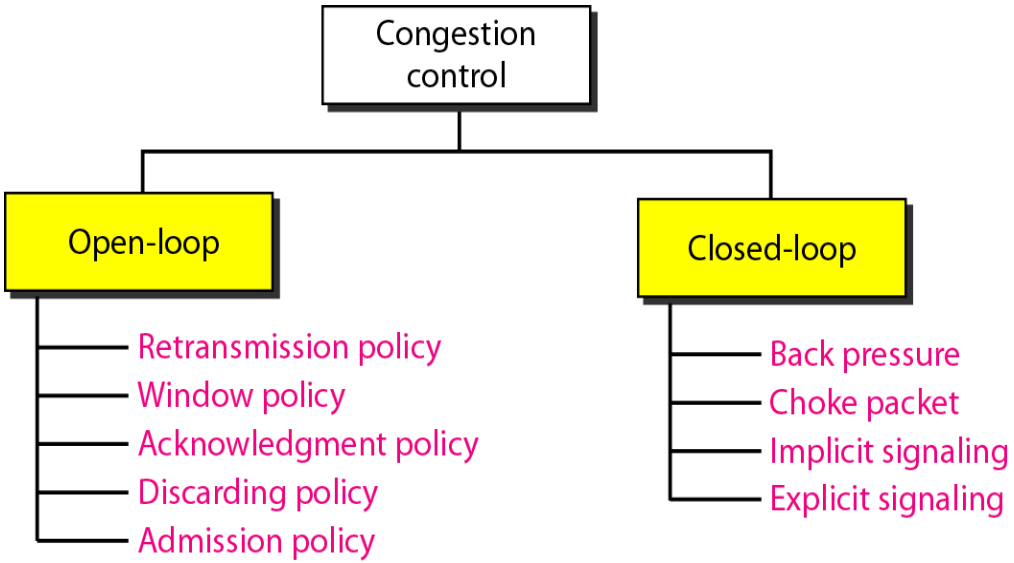
The packet departure time < packet processing time → output queue longer



CONGESTION CONTROL

Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened.

Two broad categories: **open-loop congestion control** (prevention) and **closed-loop congestion control** (removal).



A. OPEN LOOP CONTROL: PREVENTION

In open-loop congestion control, policies are applied to prevent congestion before it happens.

list of policies that can prevent congestion:

1. ***Retransmission*** policy and timers must to be designed to optimize efficiency and at the same time prevent congestion

2. ***Window policy:*** The type of window at the sender may also affect congestion. Selective Repeat is better than Go-back-N

3. ***Acknowledgement policy:*** The acknowledgment policy imposed by the receiver may also affect congestion.

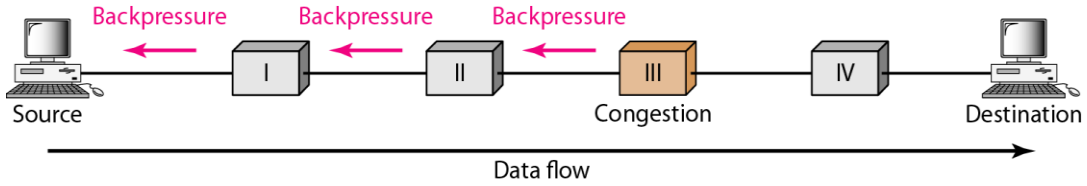
4. ***Discard policy:*** A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission.

5. ***Admission policy:*** Switch first check the resource requirement of a flow before admitting it to the network

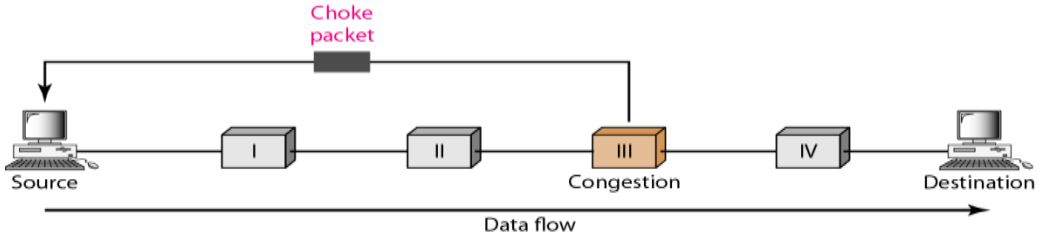
B. CLOSED-LOOP CONTROL: REMOVAL

Closed-loop congestion control mechanisms try to alleviate congestion after it happens. Several mechanisms have been used by different protocols.

1. ***Back pressure***: inform the previous upstream router to reduce the rate of outgoing packets if congested



2. ***Choke point:*** a packet sent by a router to the source to inform it of congestion, similar to ICMP's source quench packet



3. ***Implicit signaling :*** there is no communication between the congested node or nodes and the source. The source guesses that there is a congestion somewhere in the Network from other symptoms.

4. ***Explicit signaling:*** The node that experiences congestion can explicitly send a signal to the source or destination. Backward signaling / Forward signaling

- Backward Signaling A bit can be set in a packet moving in the direction opposite to the congestion. This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.

- Forward Signaling A bit can be set in a packet moving in the direction of the congestion. This bit can warn the destination that there is congestion. The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion.

CONGESTION CONTROL IN TCP

TCP assumes that the cause of a lost segment is due to congestion in the network.

If the cause of the lost segment is congestion, retransmission of the segment does not remove the cause—it aggravates it.

The sender has two pieces of information: the receiver-advertised window size and the congestion window size

TCP Congestion window

- *Actual window size = minimum (rwnd, cwnd)*

  (where rwnd = receiver window size, cwnd = congestion window size)

TCP Congestion Policy

Based on three phases: 1. slow start, 2. congestion avoidance, and 3. congestion detection

TCP CONGESTION POLICY

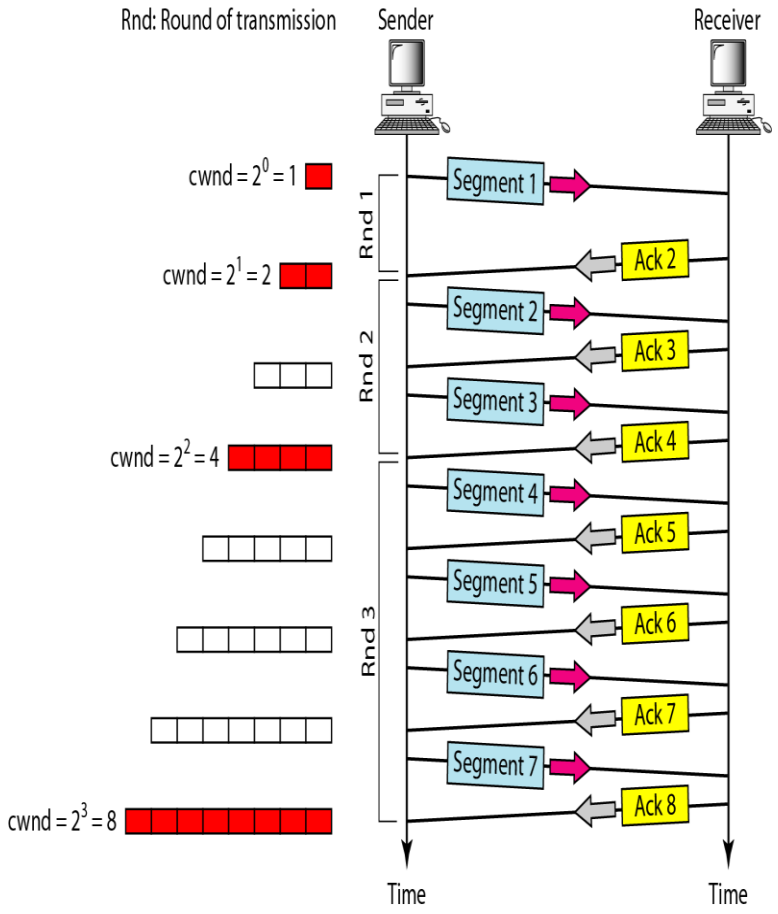
1. Slow Start: Exponential Increase

- In the slow-start algorithm, the size of the congestion window increases *exponentially* until it reaches a threshold

The sender keeps track of a variable named *ssthresh (slow-start threshold)*.

When the size of window in bytes reaches this threshold, slow start stops and the next phase starts (additive phase begins).

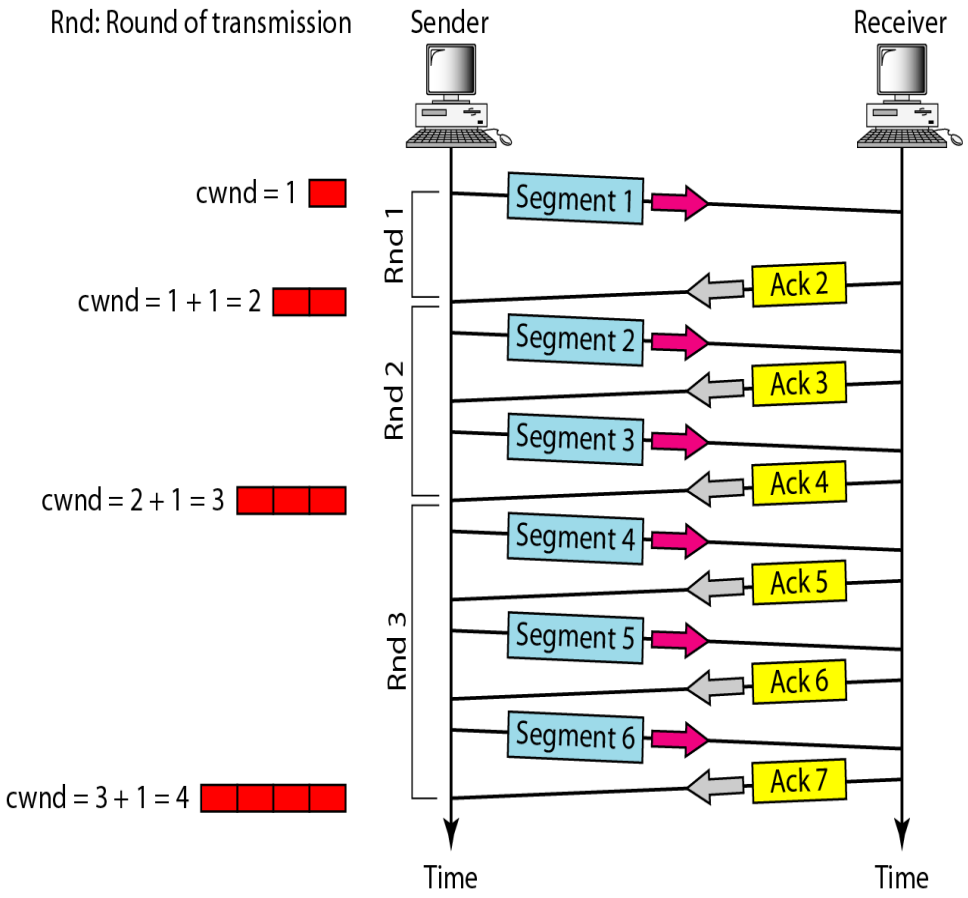
In most implementations the value of *ssthresh is 65,535 bytes.*



TCP CONGESTION POLICY

2. Congestion Avoidance: Additive

- Increase the size of the congestion window increases additively until congestion is detected



As Internet can be considered as a Queue of packets, where transmitting nodes are constantly adding packets and some of them (receiving nodes) are removing packets from the queue.

So, consider a situation where too many packets are present in this queue (or internet or a part of internet), such that constantly transmitting nodes are pouring packets at a higher rate than receiving nodes are removing them.

This degrades the performance, and such a situation is termed as Congestion.

Main reason of congestion is more number of packets into the network than it can handle.

So, the objective of congestion control can be summarized as to maintain the number of packets in the network below the level at which performance falls off dramatically.

The nature of a Packet switching network can be summarized in following points:

A network of queues

At each node, there is a queue of packets for each outgoing channel

If packet arrival rate exceeds the packet transmission rate, the queue size grows without bound

When the line for which packets are queuing becomes more than 80% utilized, the queue length grows alarmingly

When the number of packets dumped into the network is within the carrying capacity, they all are delivered, expect a few that have too be rejected due to transmission errors).

And then the number delivered is proportional to the number of packets sent.

However, as traffic increases too far, the routers are no longer able to cope, and they begin to lose packets.

This tends to make matter worse. At very high traffic, performance collapse completely, and almost no packet is delivered.

insufficient memory

Slow processors

bursty nature of traffic

REGULATING FLOW CONTROL

The bursty traffic in the network results in congestion

Traffic shaping reduces congestion and thus helps the carrier live up to its guarantees

Traffic shaping is about regulating the average rate (and burstiness) of data transmission

Congestion control refers to the mechanisms and techniques used to control congestion and keep the traffic below the capacity of the network. the congestion control techniques can be broadly classified two broad categories:

Open loop: Protocols to prevent or avoid congestion, ensuring that the system (or network under consideration) never enters a Congested State. • Close loop: Protocols that allow system to enter congested state, detect it, and remove it.

The first category of solutions or protocols attempt to solve the problem by a good design, at first, to make sure that it doesn't occur at all.

Once system is up and running midcourse corrections are not made.

These solutions are somewhat static in nature, as the policies to control congestion don't change much according to the current state of the system. Such Protocols are also known as Open Loop solutions.

These rules or policies include deciding upon when to accept traffic, when to discard it, making scheduling decisions and so on.

Main point here is that they make decision without taking into consideration the current state of the network.

The open loop algorithms are further divided on the basis of whether these acts on source versus that act upon destination.

Traffic shaping controls the *rate* at which packets are sent (not just how many)

At connection set-up time, the sender and carrier negotiate a traffic pattern (shape)

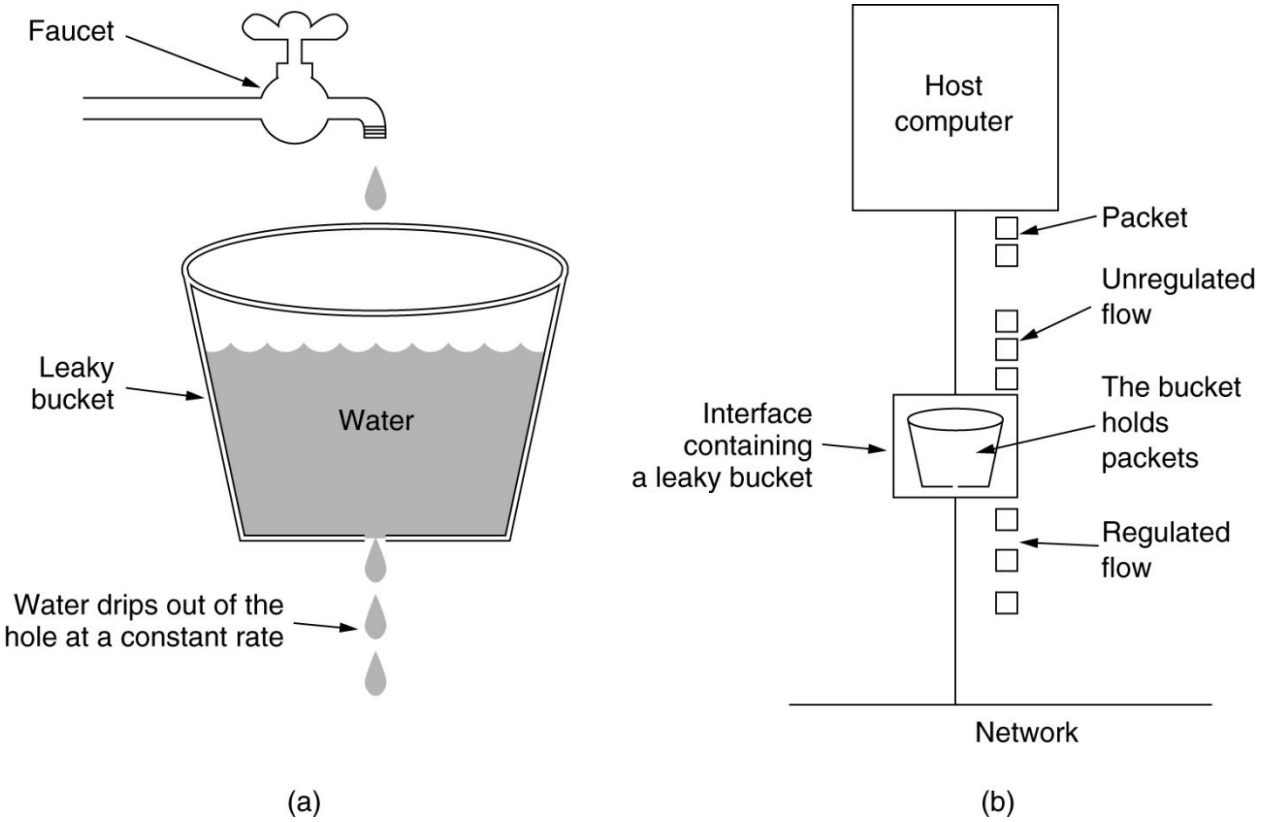
Two traffic shaping algorithms are:

- Leaky Bucket
- Token Bucket

THE LEAKY BUCKET ALGORITHM

The **Leaky Bucket Algorithm** used to control rate in a network. It is implemented as a single-server queue with constant service time. If the bucket (buffer) overflows then packets are discarded.

THE LEAKY BUCKET ALGORITHM



(a) A leaky bucket with water. (b) a leaky bucket with packets.

Leaky Bucket Algorithm (contd.)

The leaky bucket enforces a constant output rate regardless of the burstiness of the input. Does nothing when input is idle.

The host injects one packet per clock tick onto the network. This results in a uniform flow of packets, smoothing out bursts and reducing congestion.

When packets are the same size (as in ATM cells), the one packet per tick is okay. For variable length packets though, it is better to allow a fixed number of bytes per tick.

TOKEN BUCKET ALGORITHM

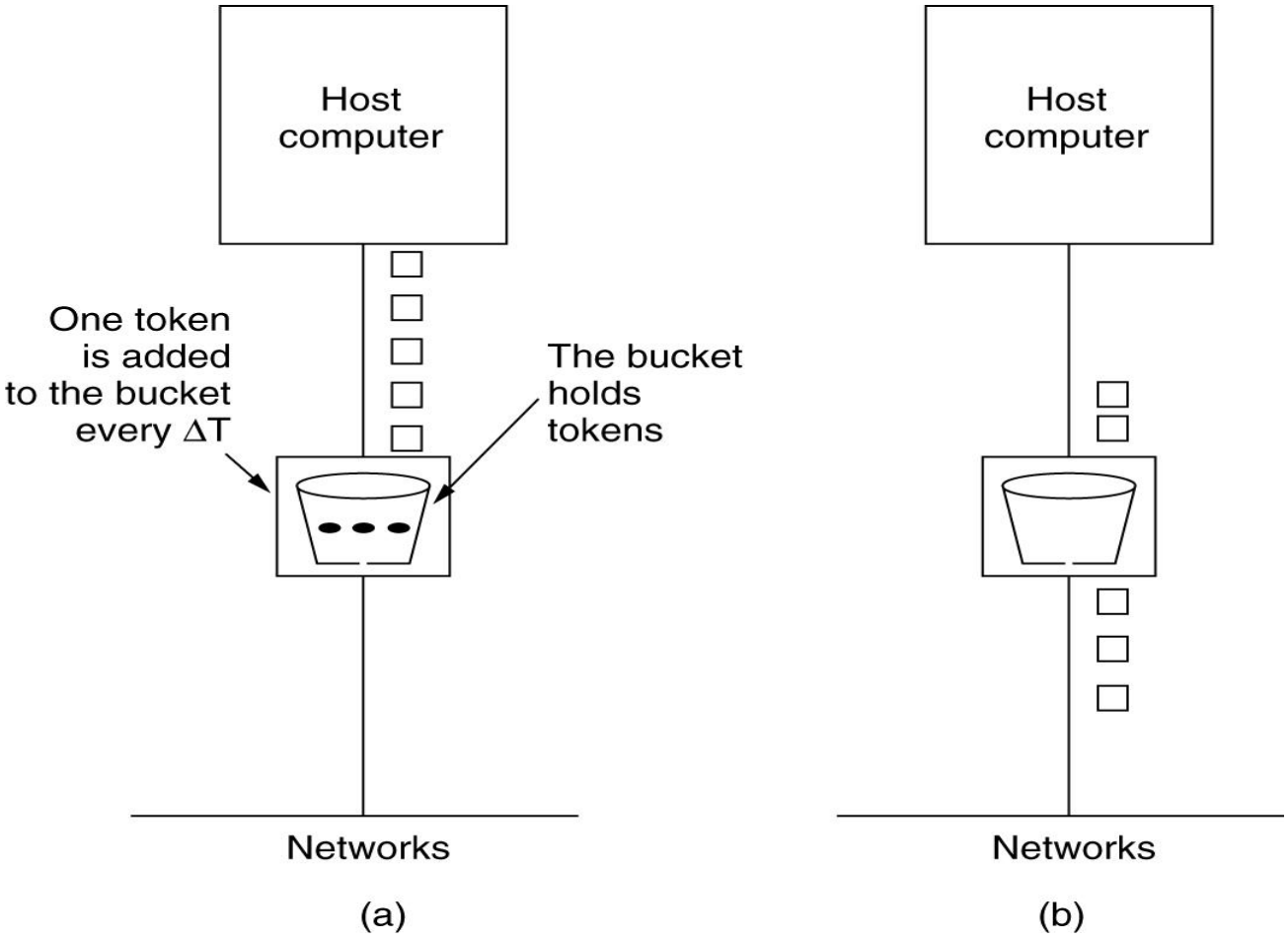
In contrast to the LB, the Token Bucket (TB) algorithm, allows the output rate to vary, depending on the size of the burst.

In the TB algorithm, the bucket holds tokens. To transmit a packet, the host must capture and destroy one token.

Tokens are generated by a clock at the rate of one token every Δt sec.

Idle hosts can capture and save up tokens (up to the max. size of the bucket) in order to send larger bursts later.

TOKEN BUCKET ALGORITHM (CONTD.)



(a) Before (b) After

TOKEN BUCKET OPERATION

TB accumulates fixed size tokens in a token bucket

Transmits a packet (from data buffer, if any are there) or arriving packet if the sum of the token sizes in the bucket add up to packet size

More tokens are periodically added to the bucket (at rate Δt). If tokens are to be added when the bucket is full, they are discarded

TOKEN BUCKET PROPERTIES

Does not bound the peak rate of small bursts, because bucket may contain enough token to cover a complete burst size

Performance depends only on the sum of the data buffer size and the token bucket size

TOKEN BUCKET - EXAMPLE

2 tokens of size 100 bytes added each second to the token bucket of capacity 500 bytes

- Avg. rate = 200 bytes/sec, burst size = 500 bytes
- Packets bigger than 500 bytes will never be sent
- Peak rate is unbounded – i.e., 500 bytes of burst can be transmitted arbitrarily fast

LB discards packets; TB does not. TB discards tokens.

With TB, a packet can only be transmitted if there are enough tokens to cover its length in bytes.

LB sends packets at an average rate. TB allows for large bursts to be sent faster by speeding up the output.

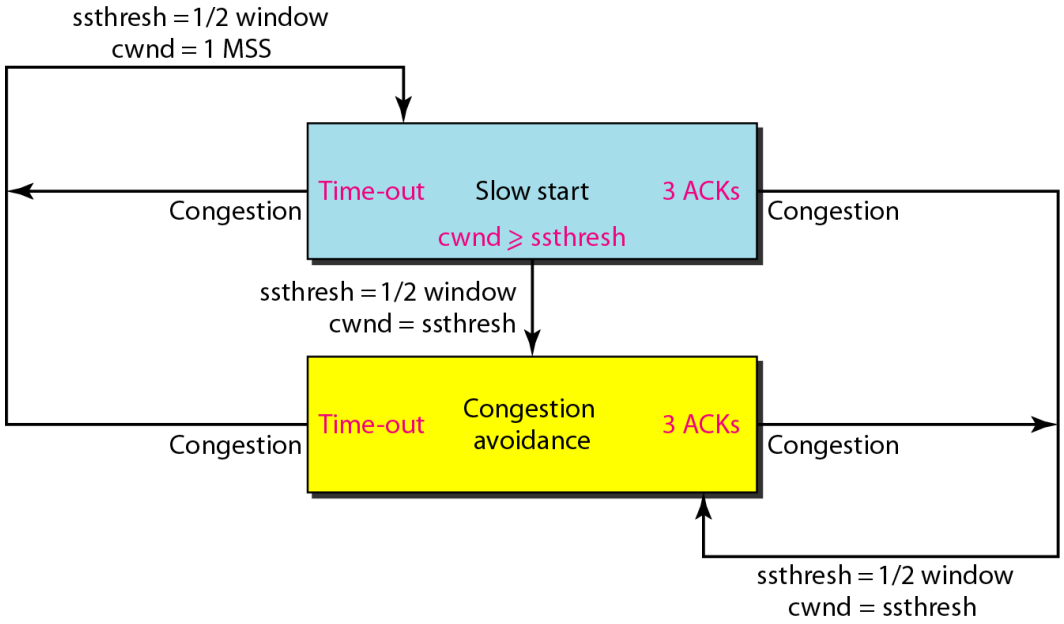
TB allows saving up tokens (permissions) to send large bursts. LB does not allow saving.

TCP CONGESTION POLICY

3. Congestion Detection: Multiplicative Decrease

An implementation reacts to congestion detection in one of two ways:

- If detection is by time-out, a new slow start phase starts
- If detection is by three ACKs, a new congestion avoidance phase starts



2. QUALITY OF SERVICE (QOS)

Flow Characteristics:

Reliability:

- needed by flow, Lack of reliability means losing a packet or acknowledgment, which entails retransmission.

Delay:

- applications can tolerate delay in different degrees.

Jitter:

- the variation in delay for packets belonging to the same flow
- High jitter means the difference between delays is large; low jitter means the variation is small.

Bandwidth:

- Different applications need different bandwidths.

QOS TECHNIQUES

Four common techniques that can be used to improve the quality of service :

- **Scheduling:** A good scheduling technique treats the different flows in a fair and appropriate manner.
- **Traffic shaping**: Leaky bucket, token bucket
- **Resource reservation**
- **Admission control**: accept or reject a flow based on predefined parameters called flow specification

SCTP

In computer **networking**, the Stream Control Transmission Protocol (**SCTP**) is a transport-layer protocol, serving in a similar role to the popular protocols TCP and UDP. It is standardized by IETF in RFC 4960.

SCTP

SCTP adds to the mix is **multi-homing.** Multi-homing allows the two endpoints of a connection to declare multiple interfaces (IP addresses). Providing and alternate route for the data in case the current interface in use fails for some reason.

The second feature is **multi-streaming**. Rather than use a single stream of data, SCTP can create multiple streams that can be used independently.

This doesn't really improve the speed of the medium but it allows the data to arrive concurrently; minimizing the wait time for pages to completely load. This feature also prevents control packets from getting blocked by data packets, like what usually happens in TCP; thereby improving data control.

TCP's 3-way handshake initiation, SCTP uses a 4-way handshake that allocates resources near the end of the entire handshake

SCTP

Data in TCP comes in packets. Packets have a specific size and a long stream would be divided to fit while short ones are spliced together. This means that message framing must be provided at the application layer to fully identify separate messages. SCTP implements message framing and each message would always have the same size when it comes out as it came in.

Summary:

1. SCTP is better at multi-homing than TCP
2. SCTP has multi-streaming while TCP doesn't
3. SCTP has initiation protection while TCP doesn't
4. SCTP has message framing while TCP doesn't
5. Ordered delivery is optional with SCTP but not with TCP