Computer Network And Network Design(CNND) ITC402

Subject Incharge

Ms. Jesleena Gonsalves

Assistant Professor

Room No. 326

email: jesleenagonsalves@sfit.ac.in

Whatsapp Group Invite Link -

Module 4

Transport Layer & Session Layer

Outline

Transport Layer:

- Transport Layer Services
- Connectionless & Connection-oriented Protocols
- Transport Layer protocols:

1. User Datagram Protocol: UDP Services, UDP Applications
2. Transmission Control Protocol: TCP Services, TCP Features, Segment, A TCP Connection, Windows in TCP, Flow Control, Error Control, TCP Congestion Control, TCP Timers.

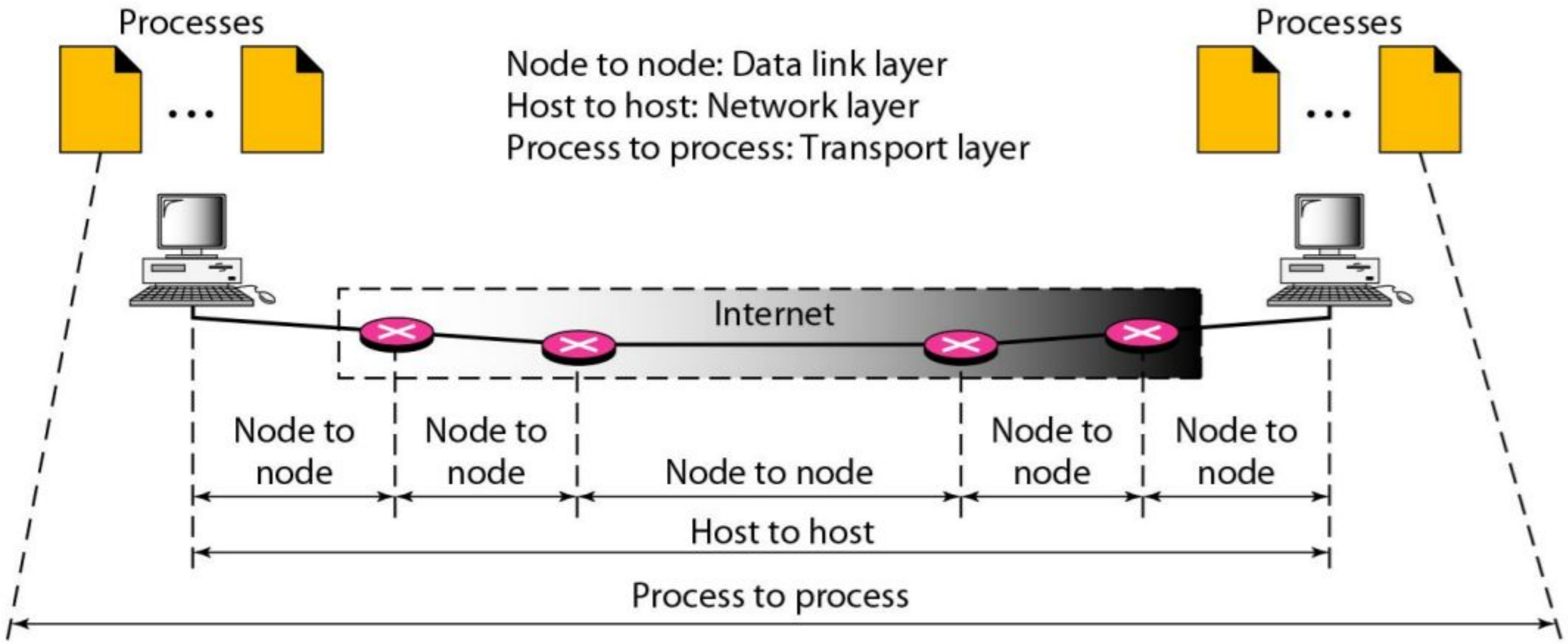
Session Layer:

- Session layer design issues
- Session Layer protocol - Remote Procedure Call (RPC)

TRANSPORT LAYER SERVICES

Process-to-Process Delivery:

- A process is a application layer entity that uses the services of the transport layer.
- A network layer protocol can deliver the message only to the destination computer. However this is an incomplete delivery.



TRANSPORT LAYER SERVICES

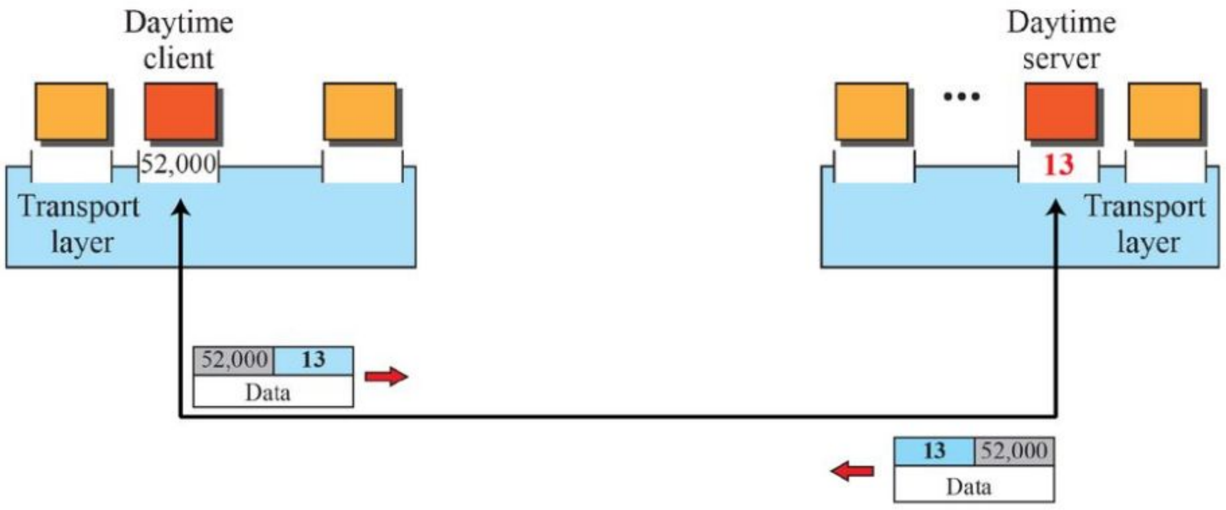
Client/Server Paradigm:

- To achieve process to process communication, the most common approach is through client/server paradigm.
- A process on the local host (client) needs service from a process on the remote host (server).
- At transport layer the addresses required are called port addresses (numbers).
- In Internet model the port numbers are 16-bit integers from 0 to 65,535.
- The client program defines itself with a port number chosen randomly.
- The server process must also define itself with a port number.
- Internet uses well-known port numbers for server processes

TRANSPORT LAYER SERVICES

Addressing: Port Numbers

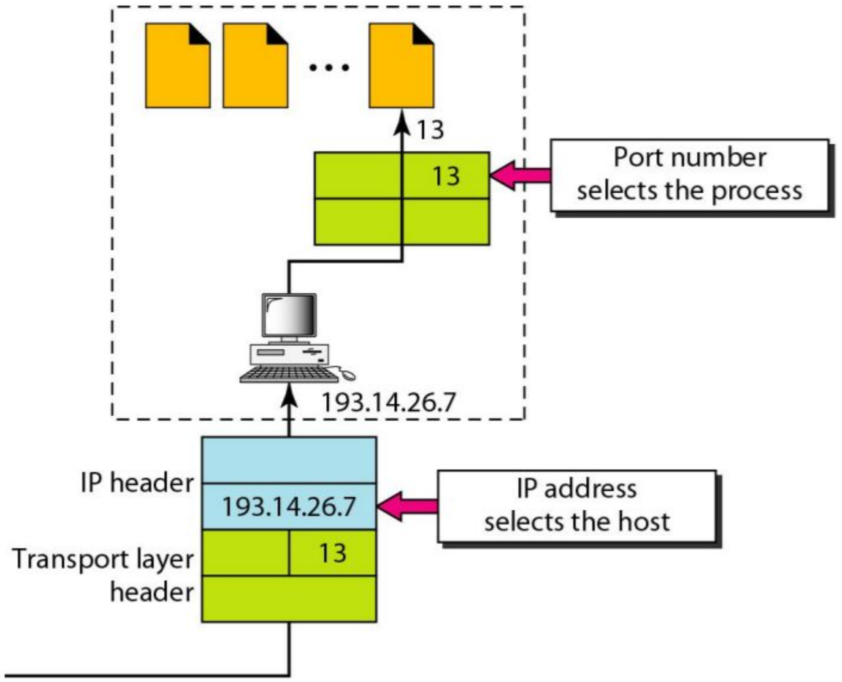
1. A client program defines itself with a ephemeral port number. It is recommended to have an ephemeral port number greater than 1023 for some client/server program to work properly.
2. A server process identifies itself with a well known port number.



TRANSPORT LAYER SERVICES

Socket Address:

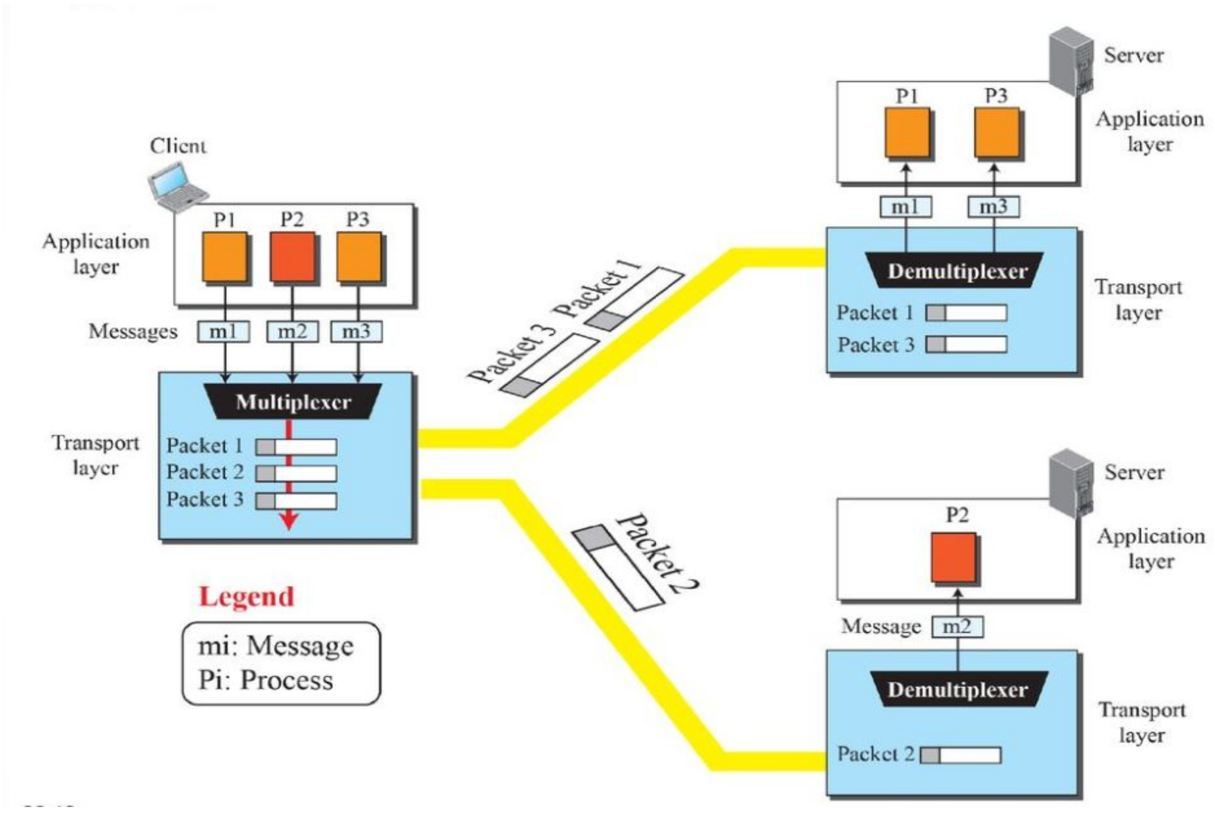
1. A transport layer protocol in the TCP suite needs both the IP address and the port number , at each end to make a connection.
2. The combination of IP address and Port number is called as Socket Address.
3. The client socket address defines the client process uniquely



TRANSPORT LAYER SERVICES

Multiplexing and Demultiplexing:

Multiplexing: An entity accepts items from more than one source.
Demultiplexing: Whenever an entity delivers items to more than one source.



TRANSPORT LAYER SERVICES

Error Control:

Error control at the transport layer is responsible for:

1. Detecting and Discarding corrupted packets
2. Keeping track of lost and discarded packets and resending them
3. Recognizing duplicate packets and discarding them
4. Buffering out of order packets until the missing packets arrive.

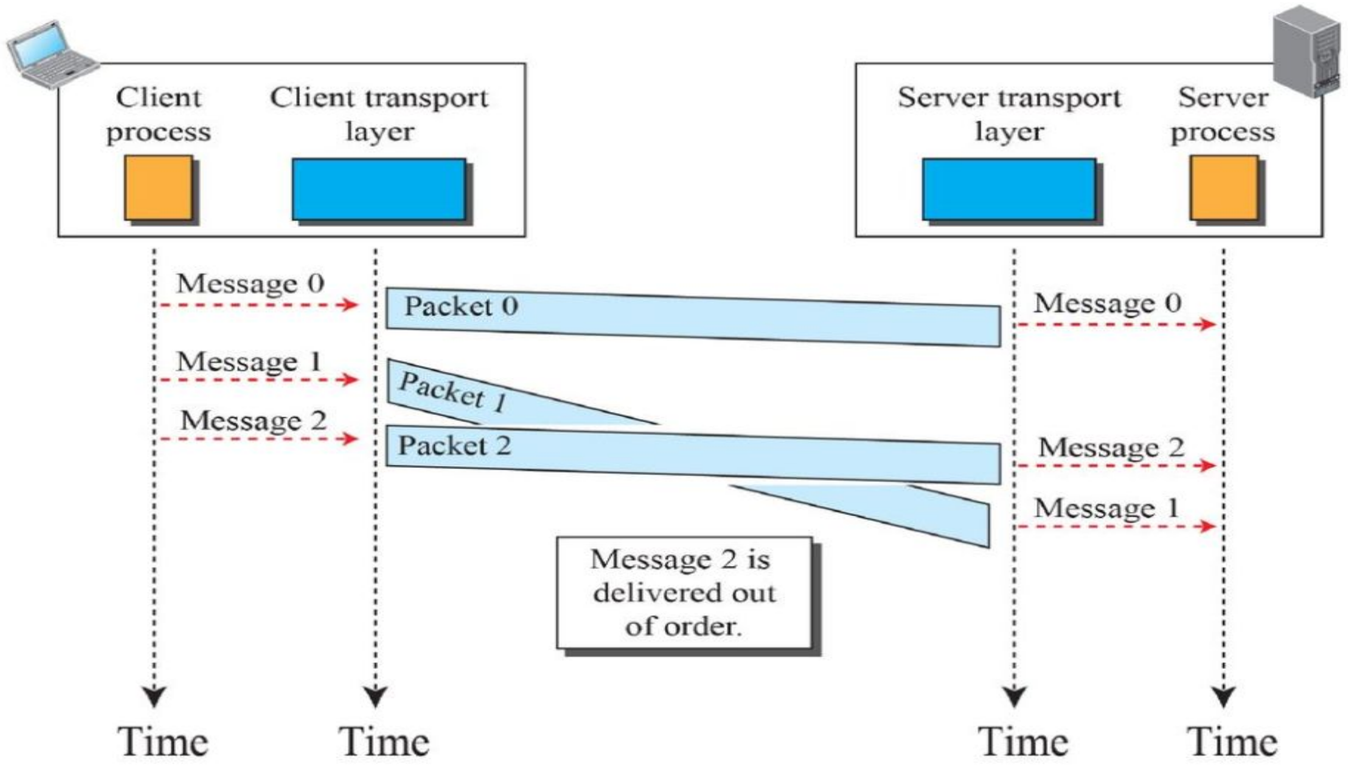
Congestion Control:

- Congestion in a network occurs when the number of packets sent to the network is greater than the capacity of the network.
- Congestion control refers to the mechanism and techniques that control the congestion and keep the load below capacity.

TRANSPORT LAYER SERVICES

Connectionless Service:

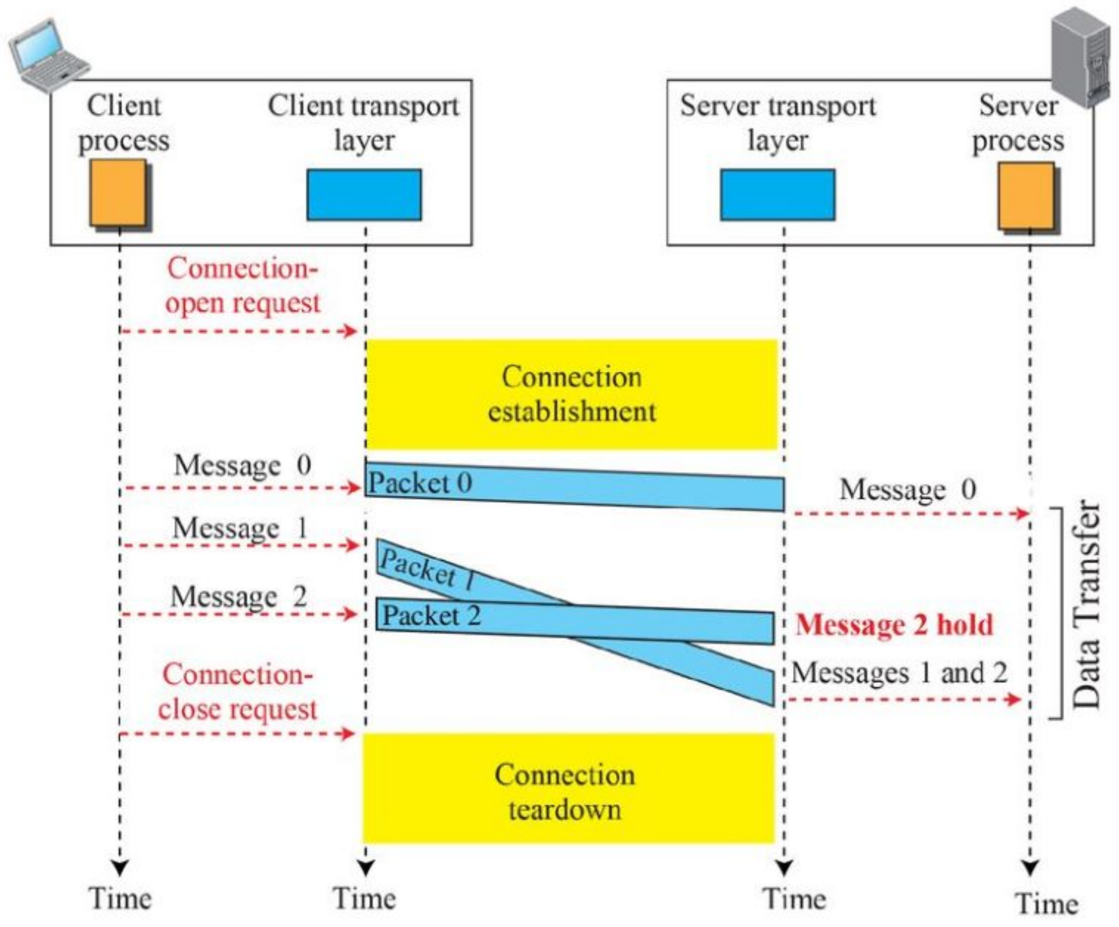
- In a connectionless service , the source process needs to divide its messages into chunks of data.
- The transport layer treats each chunk as a single unit without any relation between the chunks.



TRANSPORT LAYER SERVICES

Connection oriented service:

A connection is first established, data is transferred, then connection is released.



TRANSPORT LAYER SERVICES

Reliable versus Unreliable Service

- Transport layer supports reliable and unreliable service.

Reliable Service:

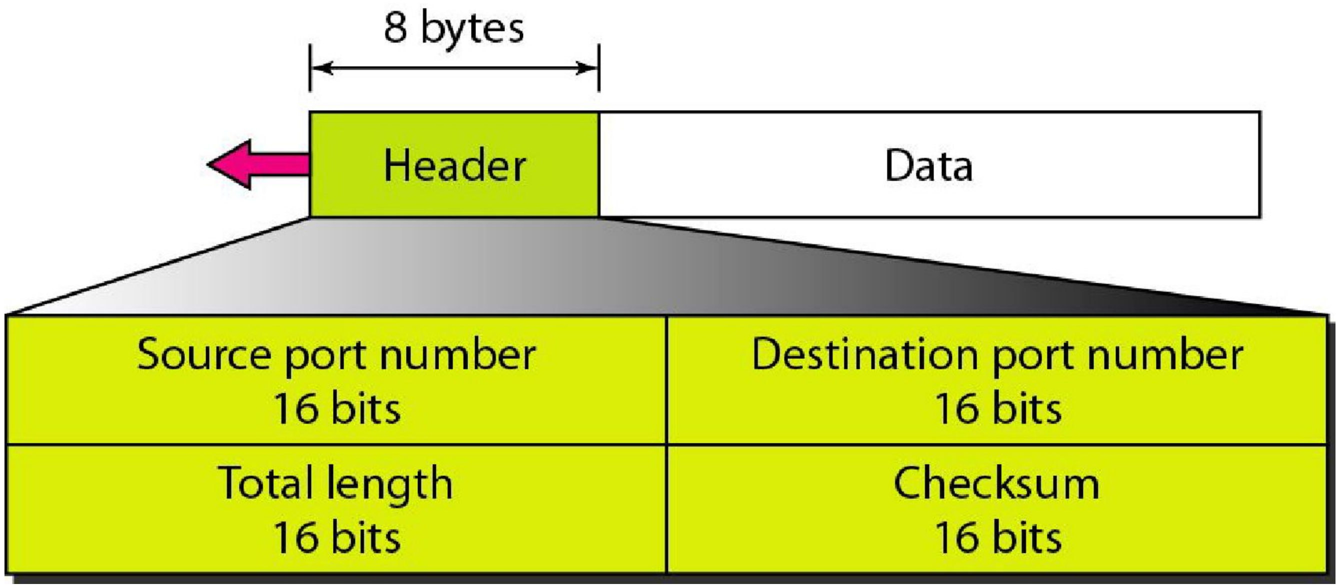
- If the application program needs reliability, we can use reliable protocol by implementing error and flow control.
- It is slower and more complex service
- E.g. TCP and SCTP protocol. TCP uses sliding window protocol for error and flow control

Unreliable Service:

- If the application needs faster services, then an unreliable protocol can be used.
- E.g. UDP protocol.

USER DATAGRAM PROTOCOL

- The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol.
- UDP is a simple protocol using minimum of overhead .
- If any process wants to send small data without much reliability, it can use UDP.
- UDP needs much less interaction between sender and receiver than TCP.
- UDP packet is called "User Datagram", have a fixed size header of 8 bytes.



USER DATAGRAM PROTOCOL

UDP Services:

1. Connectionless Services
2. Flow and Error Control
3. Encapsulation and Decapsulation
4. Queuing

USER DATAGRAM PROTOCOL

Use of UDP

- The following lists some uses of the UDP protocol:

1. UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
2. UDP is suitable for a process with internal flow and error control mechanisms.
3. UDP is a suitable transport protocol for multicasting.
4. UDP is used for management processes such as SNMP.
5. UDP is used for some route updating protocols such as Routing Information Protocol.

TRANSMISSION CONTROL PROTOCOL

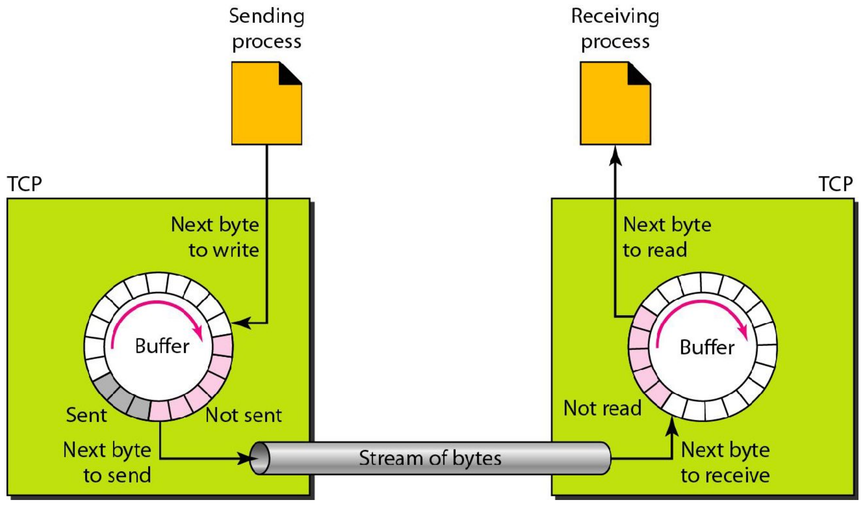
- TCP is a connection-oriented protocol.
- It creates a virtual connection between two TCPs to send data.
- In addition, TCP uses flow and error control mechanisms at the transport level.

TCP SERVICES

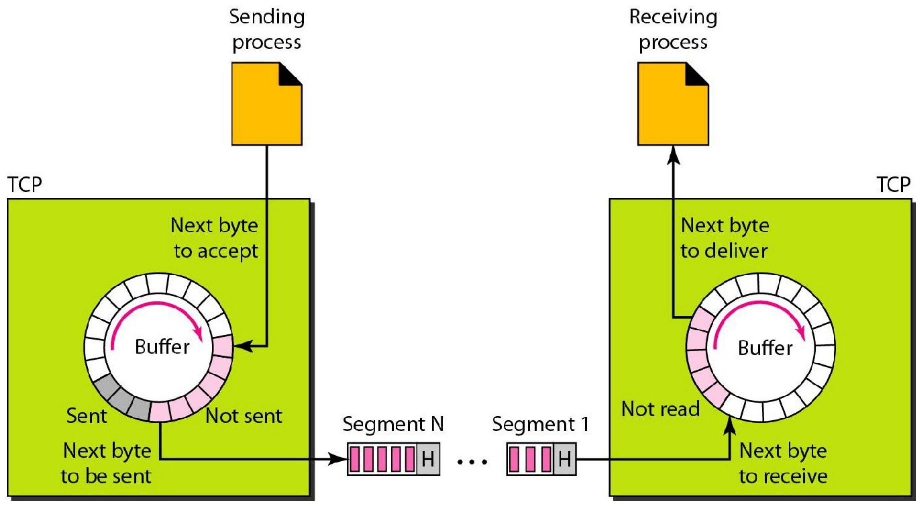
- TCP offers following services to the processes at the application layer

1. Process to process Communication
2. Stream Delivery Service
3. Sending and Receiving Buffers
4. Segmentation
5. Full duplex communication
6. Connection oriented service
7. Reliable service

TCP SERVICES



Sending and Receiving Buffers



Segmentation

TCP FEATURES

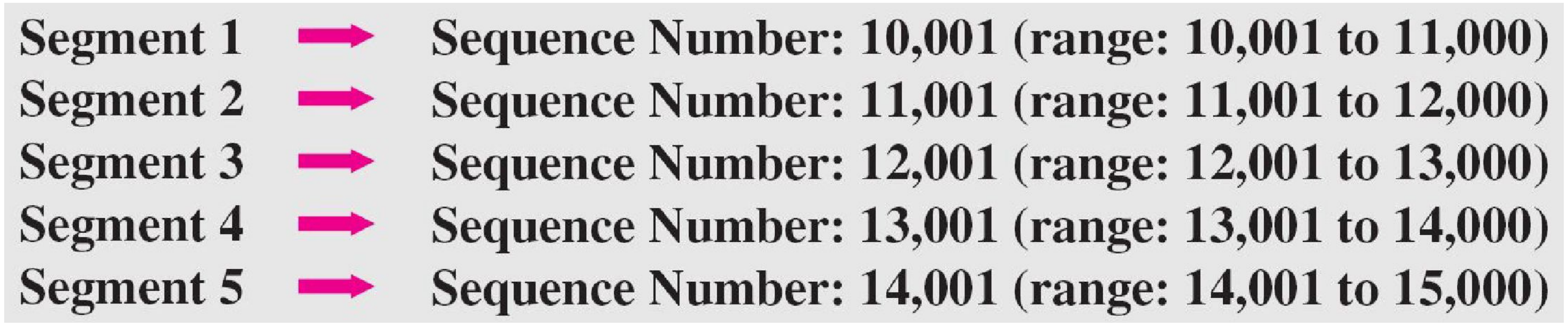
- TCP has several features for supporting the services provided by it

1. Numbering System
2. Flow Control
3. Error Control
4. Congestion(Traffic) Control

TCP FEATURES

Example:

- TCP wants to transfer a file of 5000 bytes. The first byte number is 10001. what are the sequence numbers, if data is needed to send in 5 segments?



Segment 1	→	Sequence Number: 10,001 (range: 10,001 to 11,000)
Segment 2	→	Sequence Number: 11,001 (range: 11,001 to 12,000)
Segment 3	→	Sequence Number: 12,001 (range: 12,001 to 13,000)
Segment 4	→	Sequence Number: 13,001 (range: 13,001 to 14,000)
Segment 5	→	Sequence Number: 14,001 (range: 14,001 to 15,000)

SEGMENT

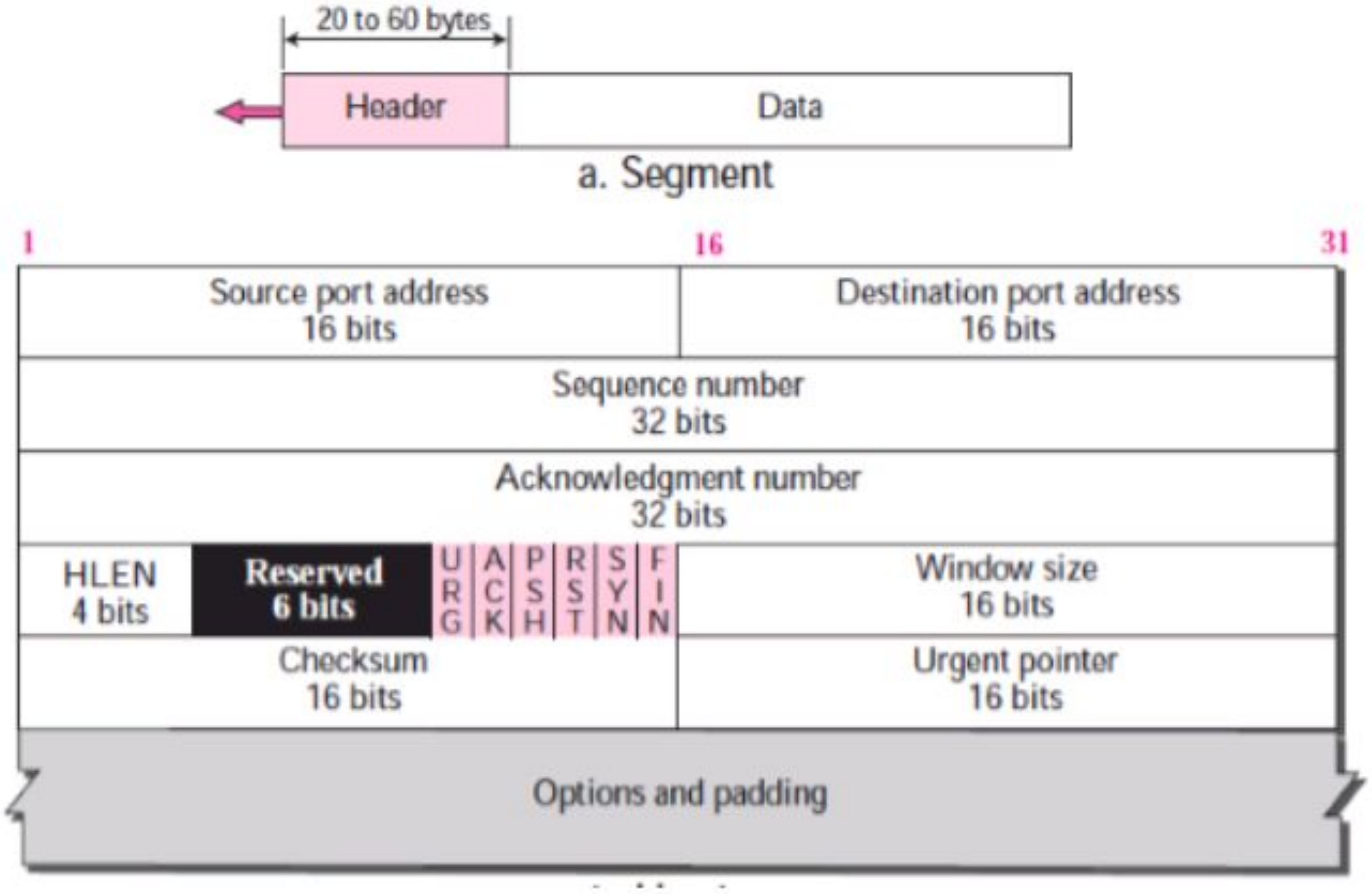
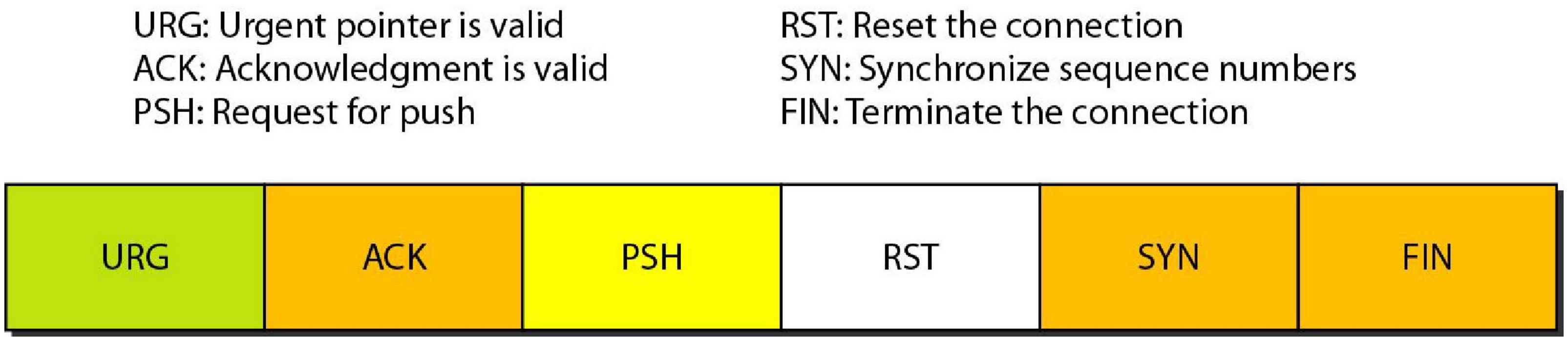


Figure 1. TCP segment Header

SEGMENT


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
URG
ACK
PSH
RST
SYN
FIN

Control field

A TCP CONNECTION

- TCP is connection oriented.
- It establishes a virtual path between the source and destination
- All the segments belonging to a message are sent over this virtual path.
- This virtual path now is responsible for data transfer, the acknowledgement process as well as retransmission of damaged or lost frames
- TCP requires three phases for transmission:

1. Connection Establishment
2. Data transfer
3. Connection Release(termination)

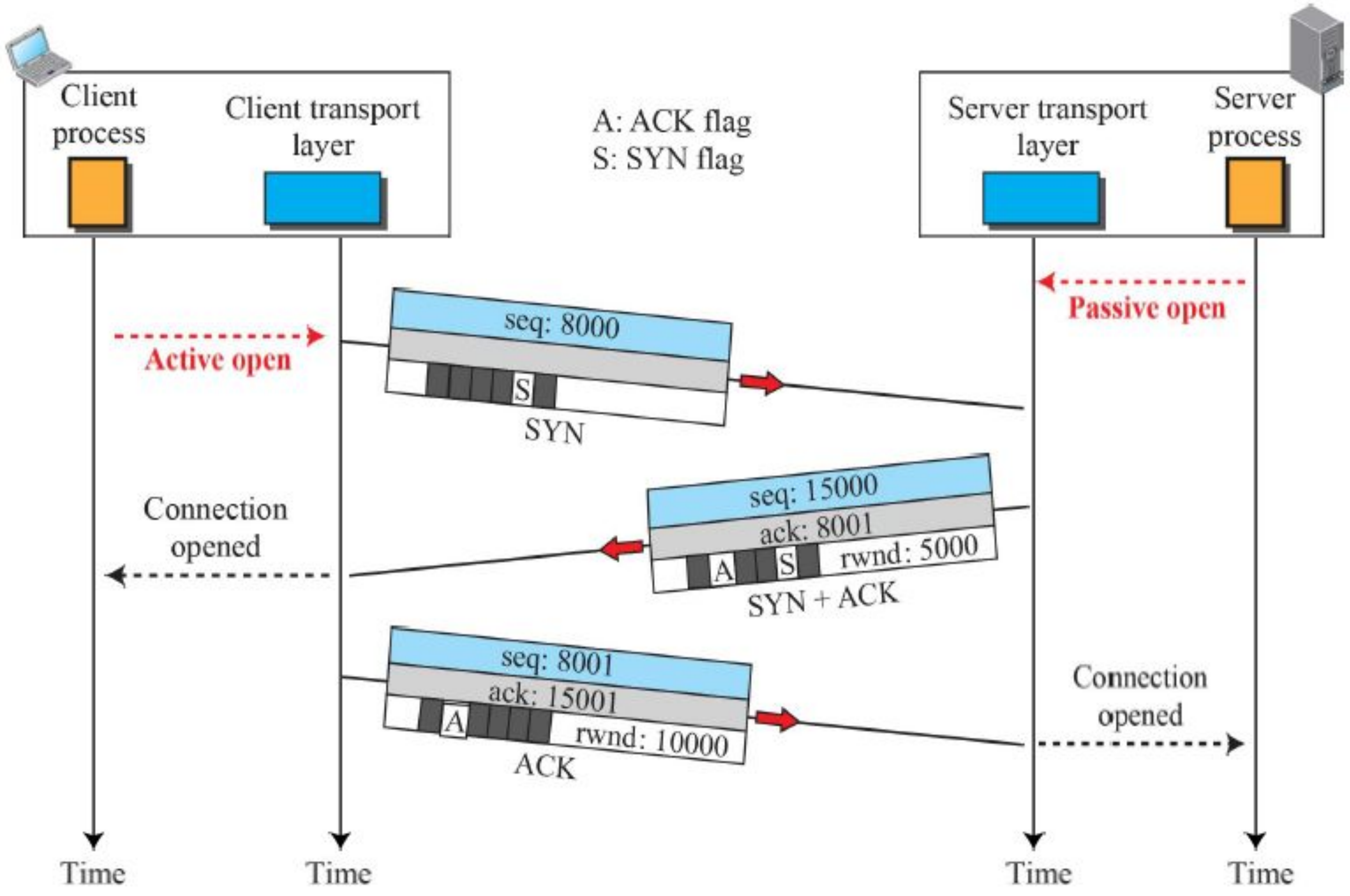
CONNECTION ESTABLISHMENT

- TCP transmits data in full duplex mode.
- Before actual data transfer, each party must initialize communication.
- TCP does this by a "Three-Way handshaking"

Three-Way Handshaking

- Assume, an application program called 'client', wants to make a connection with another application program.
- The process starts with server, server TCP first initiates a ***passive open*** (telling TCP entity that it is ready to accept a connection).
- The client program issues a request for an ***active open.***
- TCP on the client machine starts this by three-way handshaking process (three signals SYN, ACK+SYN, ACK).
- The handshaking signals are segments shown with only important fields

CONNECTION ESTABLISHMENT



A TCP CONNECTION

Simultaneous open

- When both processes, at client and server issues an active open, it is called simultaneous open.
- In this case, both TCPs transmit a SYN+ACK segment to each other and a single connection is established between them.

Resource Allocation

- Once three way handshake is successful, both TCP's allocate required resources to the connection

1. Port activation
2. Sending and receiving buffer allocation
3. Stream Delivery Service
4. Segmentation
5. Full duplex communication

A TCP CONNECTION

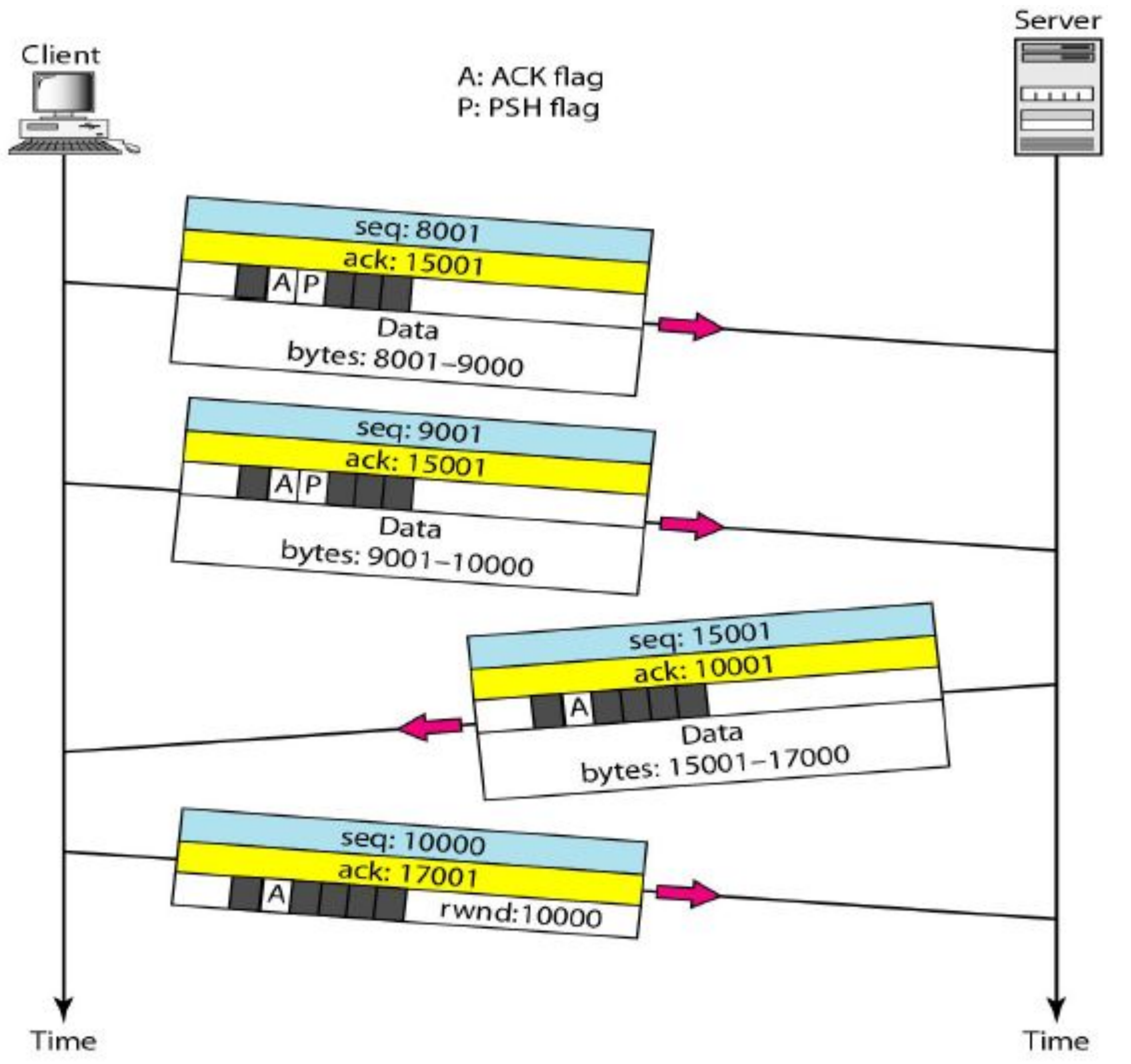
SYN Flooding Attack

- Here a malicious attacker sends a large number of SYN segments to a server with fake IP addresses
- Server considers them as from different clients.
- The server allocates necessary resources for each connection and sends SYN+ACK segments to this fake clients, which are lost
- If this SYN segments are large in number, eventually server runs out of resources (which are not at all used) and may crash
- To avoid this situation, servers are programmed for Denial-of-service security attack
- Possible implementation are: limit on connection request, filter out datagrams from unwanted IP addresses, postpone resource allocation until the entire connection is setup

A TCP CONNECTION- DATA TRANSFER

- After connection establishment, Data & acknowledgements can flow in both directions simultaneously.
- The acknowledgment is usually piggybacked

A TCP CONNECTION- DATA TRANSFER



A TCP CONNECTION- DATA TRANSFER

Pushing Operation

- Required in case of real time data transfer.
- The application programs require on time services
- Set PSH bit means

1. Sending TCP should not wait to make segment
2. Receiving TCP should deliver the data fast

Urgent Data

- Applications might need to send urgent data to the other party
- Set URG bit means

1. Sending TCP should place this data ahead of the any other data.
2. Receiving TCP should first deliver this data to receiving program

CONNECTION TERMINATION

- Any of the two parties (client or server) can close the connection
- The available closing options are:

1. Three way handshaking
2. Four way handshaking with a half-close option

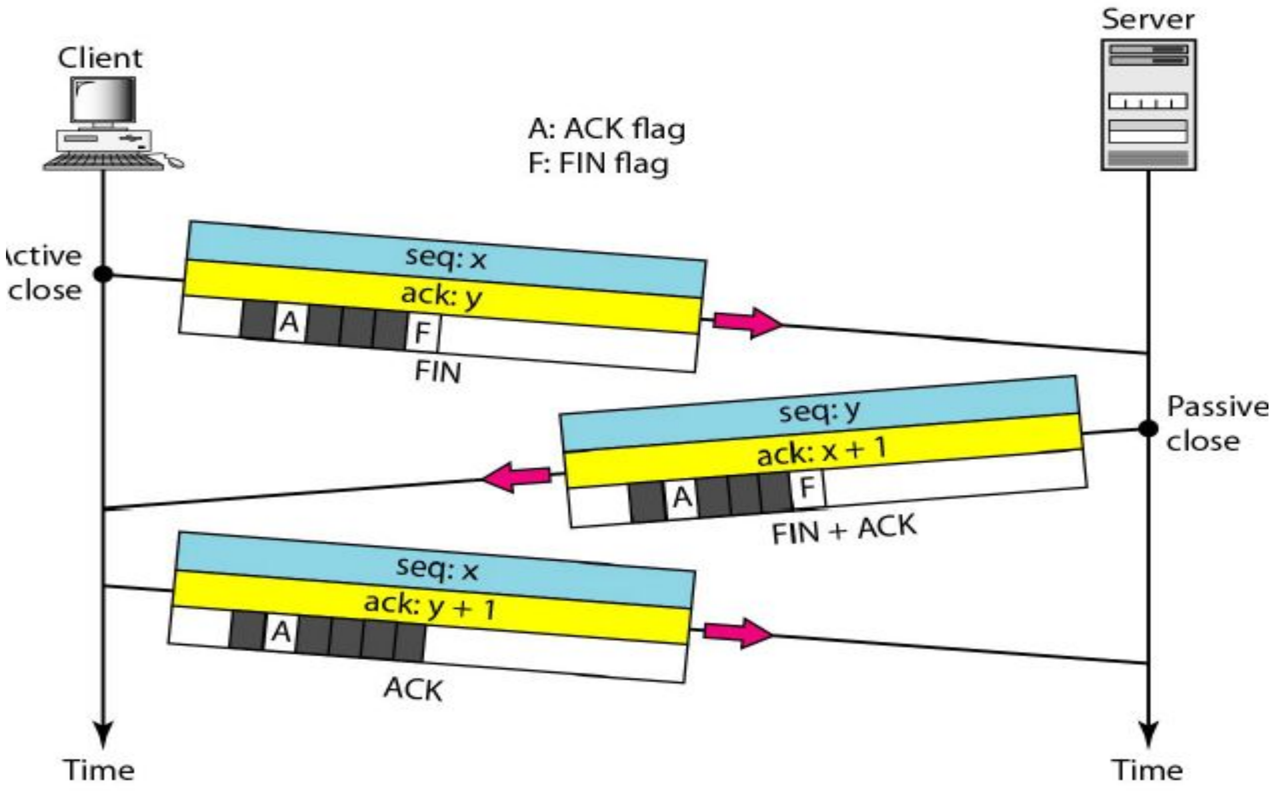
CONNECTION TERMINATION

Three way handshaking

- The client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment (with a FIN flag set).
- The FIN segment can include the last chunk of data from client OR it can be just a control segment.
- The FIN segment consumes one sequence number even if it does not carry data.
- The server now sends a FIN+ACK segment, to confirm the receipt of FIN segment as well as announcing the closing of the connection.
- This segment also can contain the last chunk of data from server
- The FIN+ACK segment also consumes one sequence number if it does not carry data.

CONNECTION TERMINATION

- The client TCP sends the last segment, an ACK segment to confirm the receipt of FIN segment from server.
- This segment contains ACK number (seq.no+1).
- This segment can not carry data and consumes no sequence number

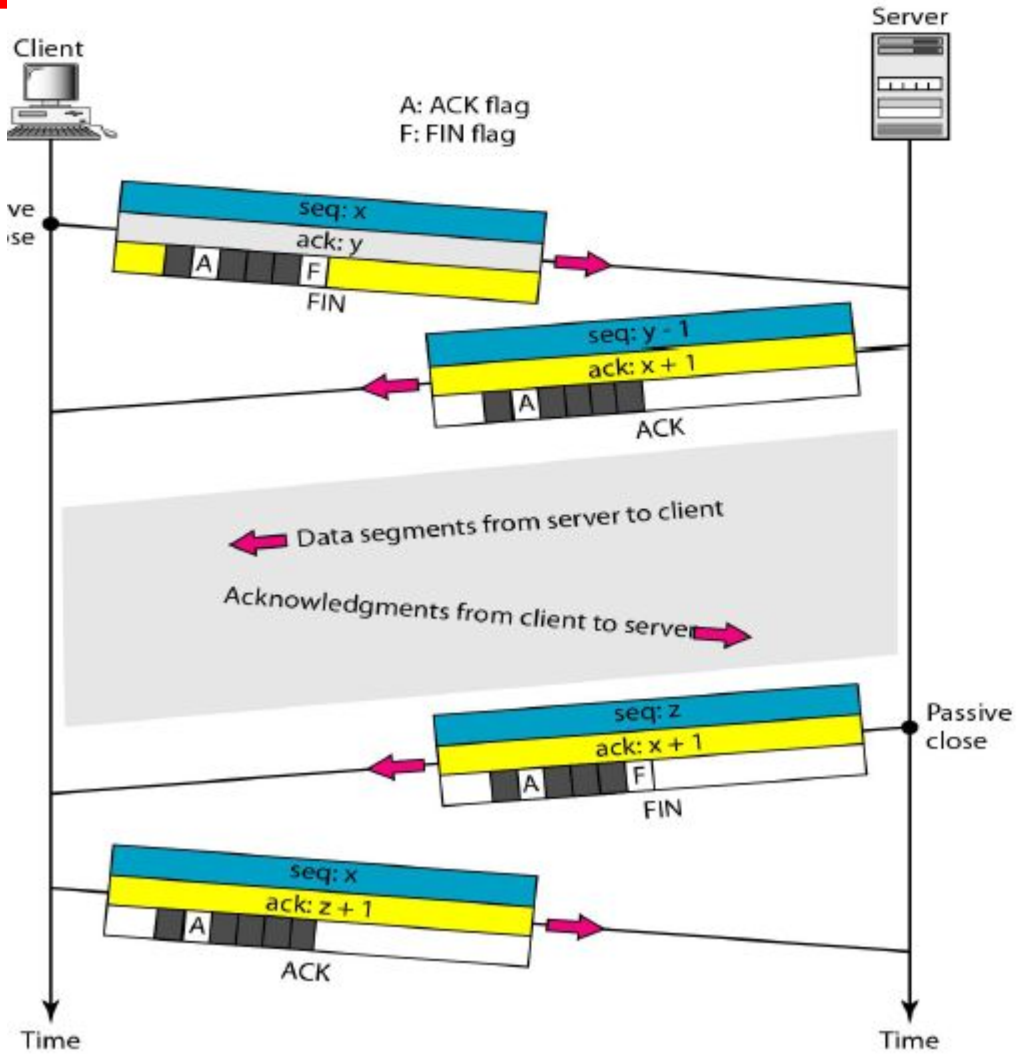


CONNECTION TERMINATION

Four way handshaking with a half-close option.

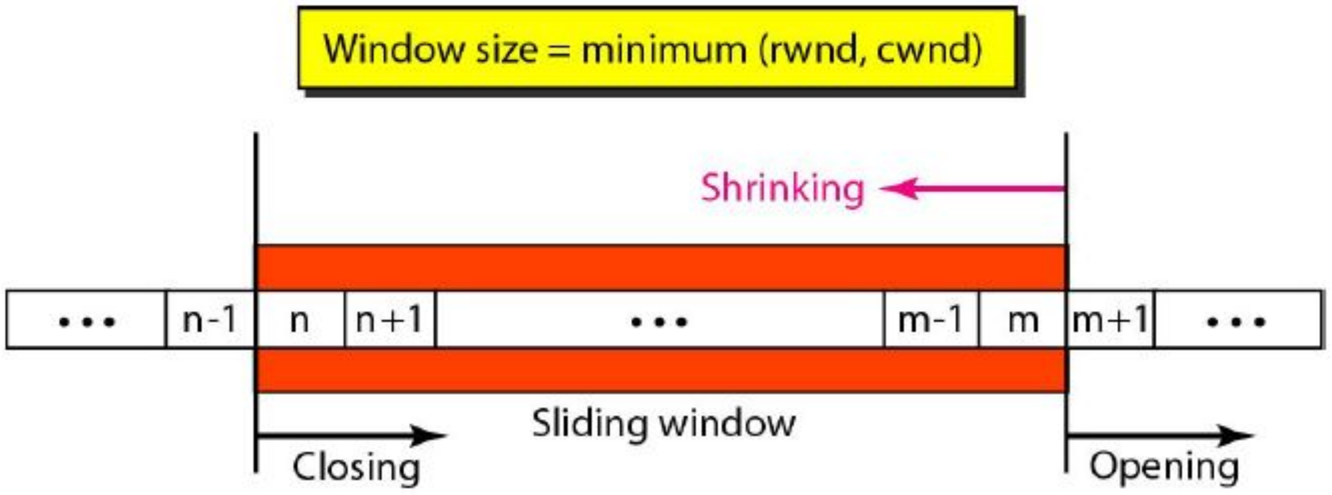
- In TCP, one end can stop sending data while still receiving data.
- The case is called Half-Close.
- It can occur when the server needs all the data before processing can begin. E.g. – sorting.
- In half-closing, client half closes the connection by sending a FIN segment.
- The server accepts the half-close by sending the ACK segment.
- Now the data transfer from client to server stops.
- The server can still send the data. E.g. result of sort.
- When server has sent all the processed data, it sends a FIN segment, which is acknowledged by an ACK from the client.

CONNECTION TERMINATION



FLOW CONTROL

- TCP uses sliding window protocol to handle flow control.
- It's a combination of Go-back-N and Selective Repeat Sliding window.
- The sliding window used by TCP differs from normal sliding window.
- It is byte oriented and of variable size.



FLOW CONTROL

- The window at one end is determined by the lesser of the two values: receiver window (rwnd) and congestion window (cwnd)
- The receiver window is the value advertised by the opposite end in a segment containing acknowledgement.
- It is the number of bytes the other end can accept before its buffer overflows and data are discarded.
- The congestion window is determined by the network to avoid congestion.

ERROR CONTROL

- TCP is a reliable transport layer protocol.
- Error control includes mechanisms for detecting corrupted segments, lost segments, out-of-order segments, and duplicated segments.
- Error control also includes a mechanism for correcting errors after they are detected.
- Error detection and correction in TCP is achieved through the use of three simple tools: checksum, acknowledgment, and time-out.

ERROR CONTROL

Checksum

- Each segment includes a checksum field which is used to check for a corrupted segment.
- If the segment is corrupted, it is discarded by the destination TCP and is considered as lost.
- TCP uses a 16-bit checksum that is mandatory in every segment.

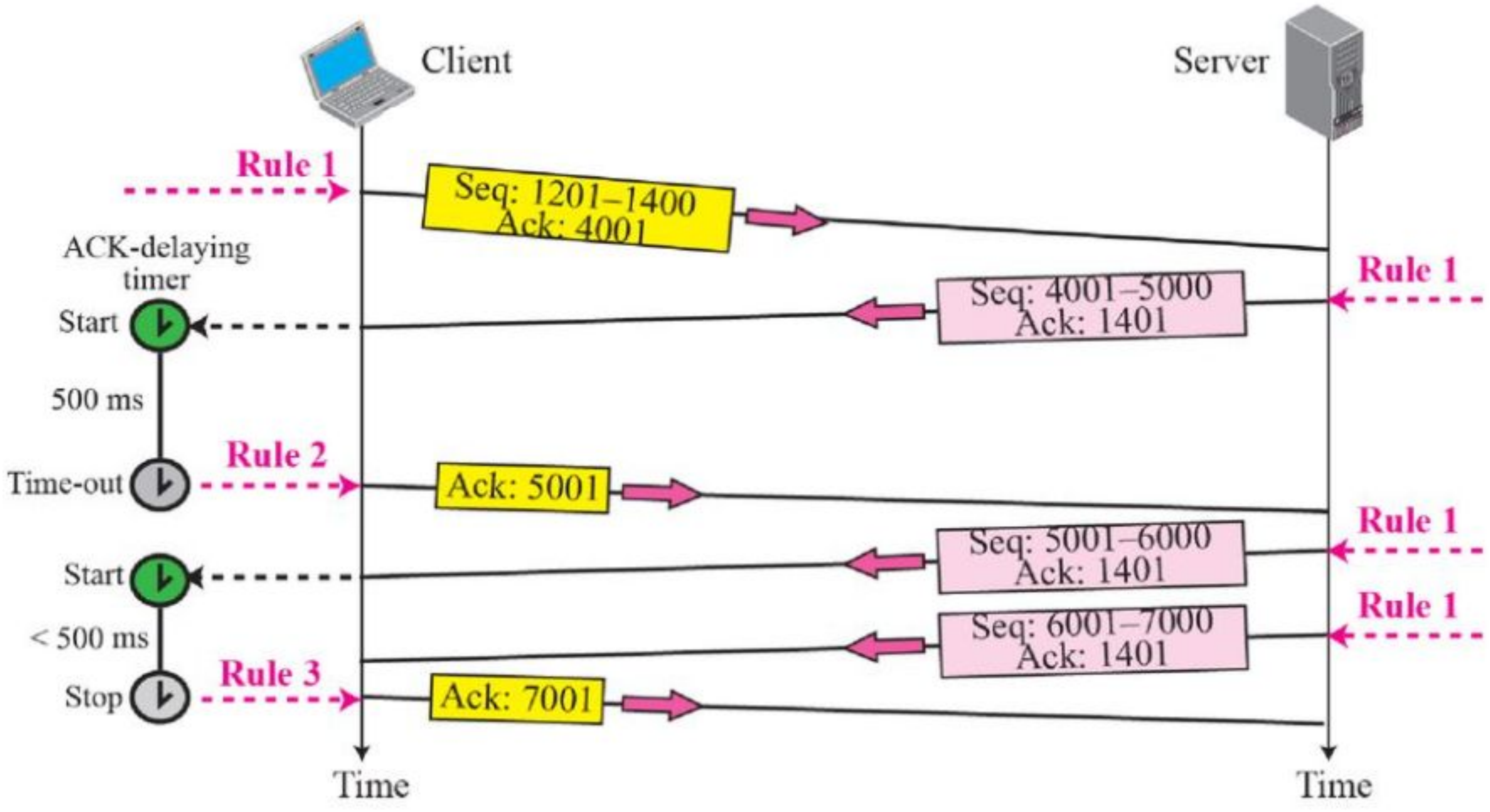
Acknowledgment

- TCP uses acknowledgments to confirm the receipt of data segments.
- Control segments that carry no data but consume a sequence number are also acknowledged.
- ACK segments are never acknowledged.
- ACK segments do not consume sequence numbers and are not acknowledged.

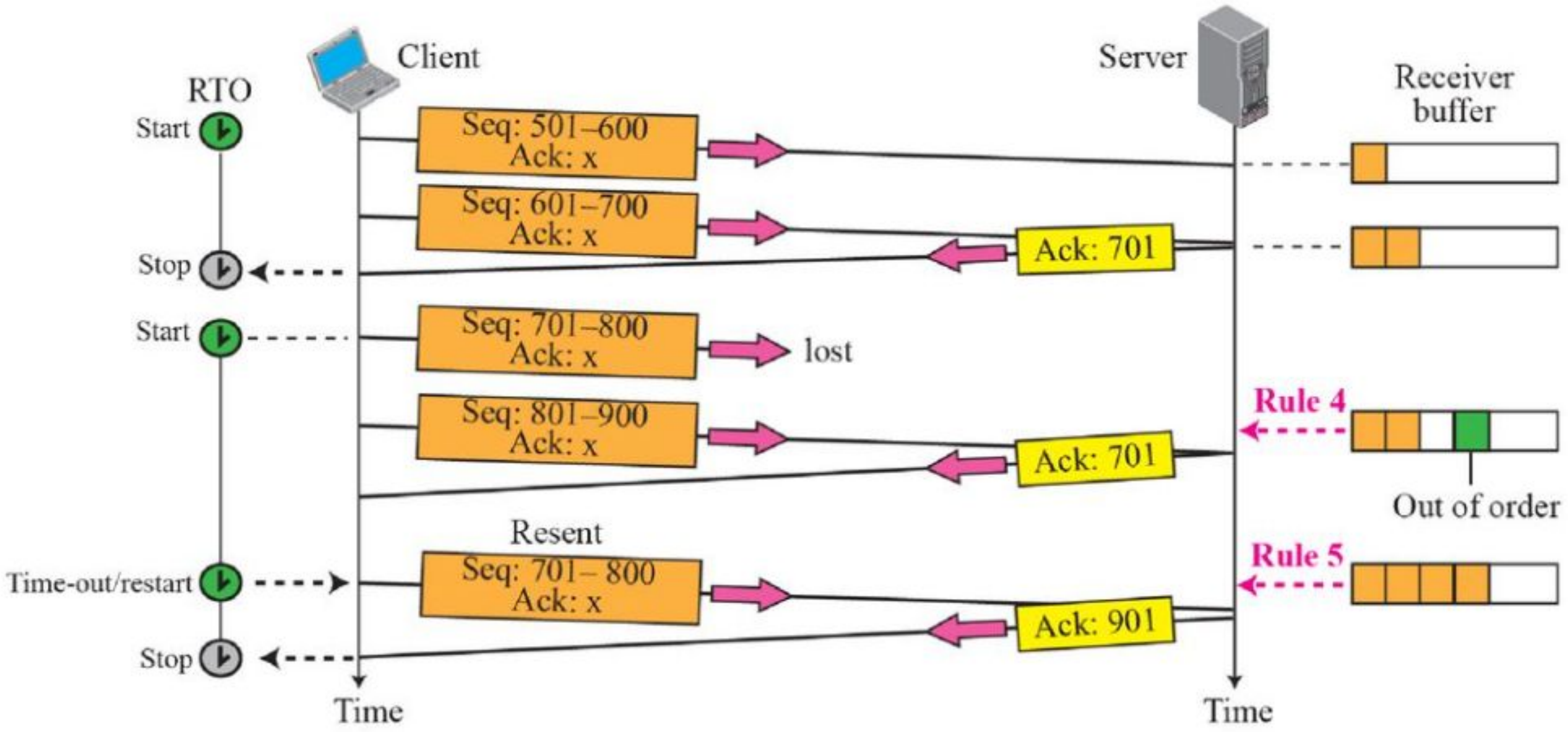
Retransmission

- The heart of the error control mechanism is the retransmission of segments.
- When a segment is corrupted, lost, or delayed, it is retransmitted.
- In modern implementations, a segment is retransmitted on two occasions: when a retransmission timer expires or when the sender receives three duplicate ACKs

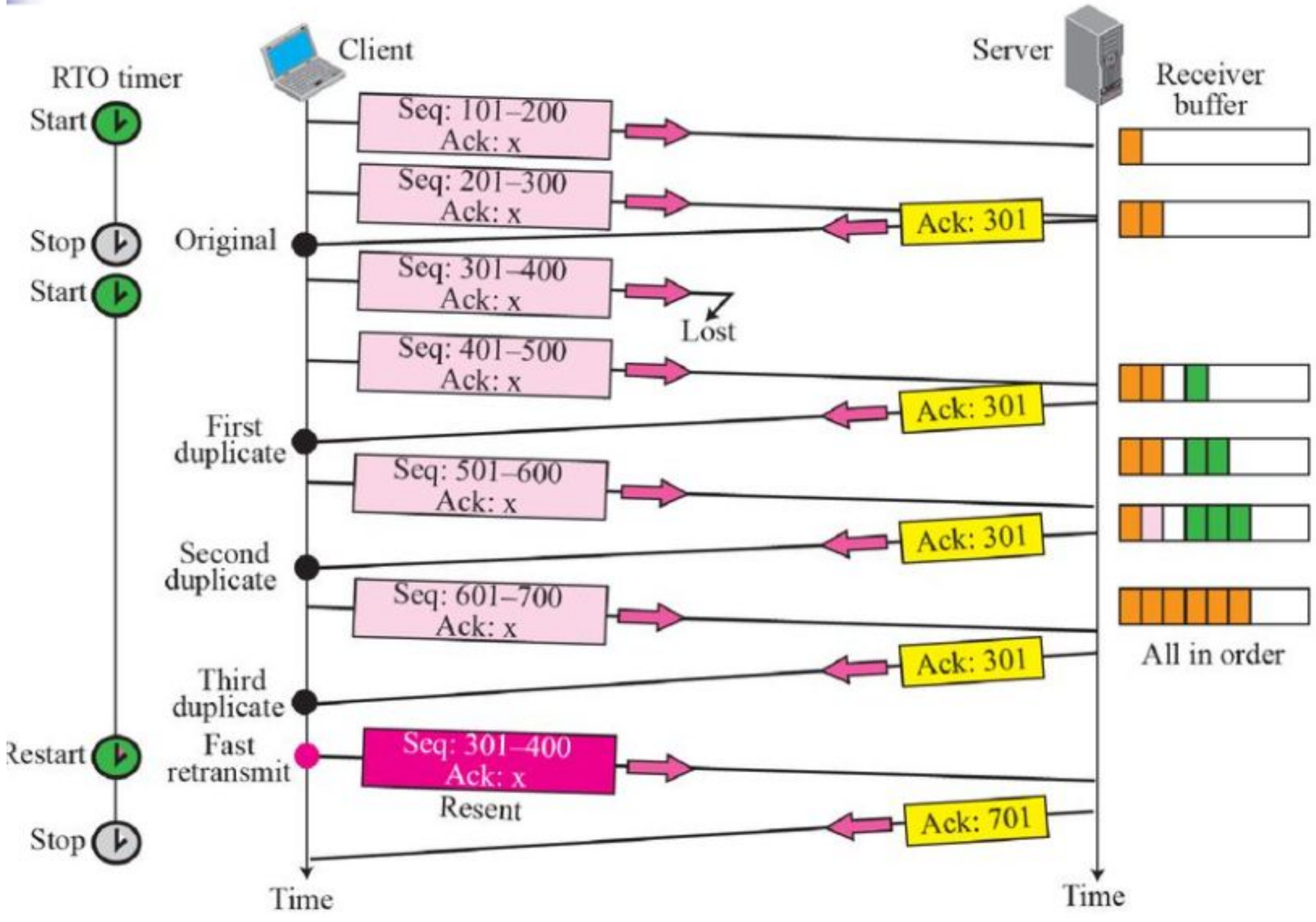
ERROR CONTROL- NORMAL OPERATION



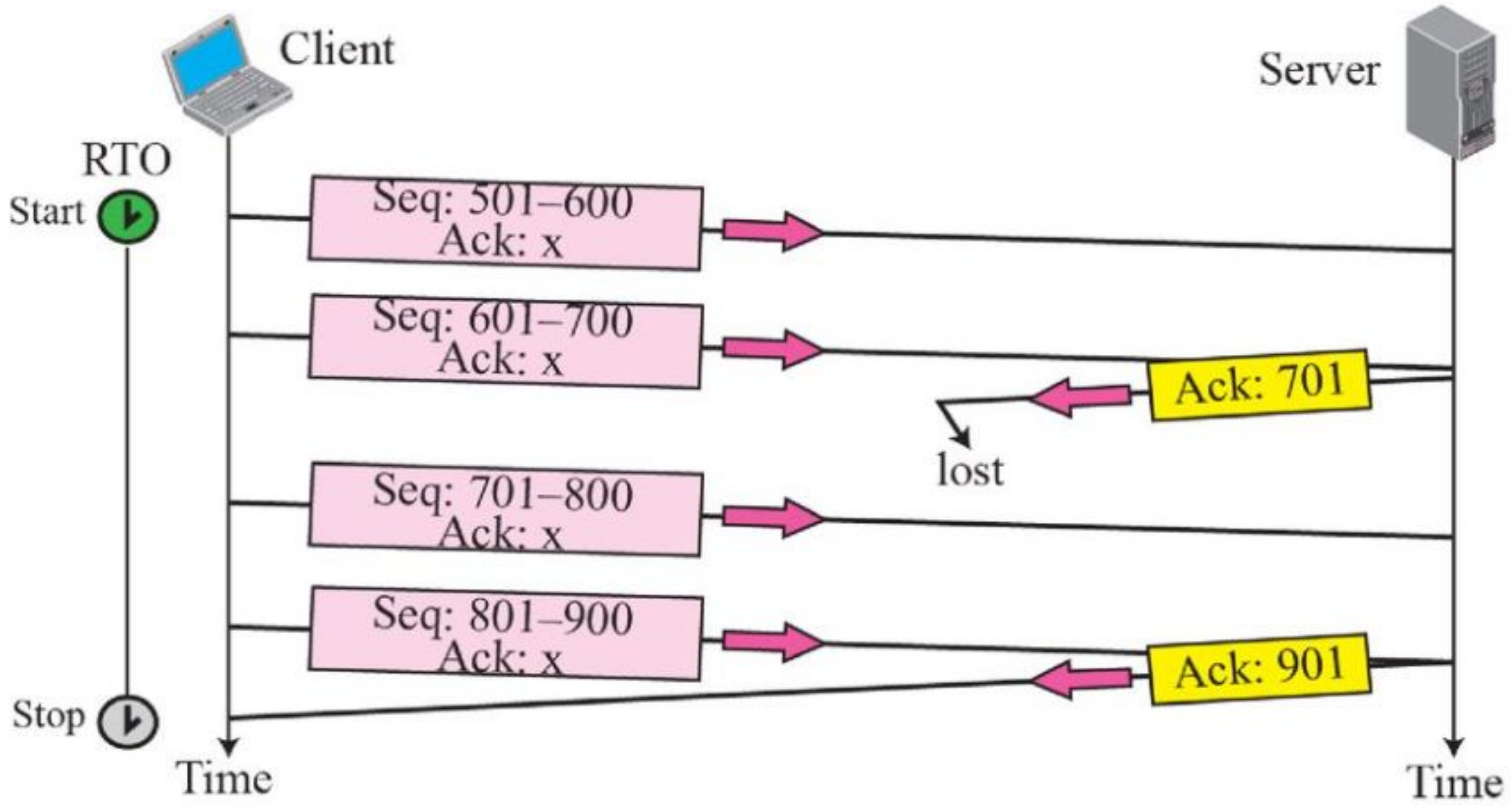
ERROR CONTROL- LOST SEGMENT



ERROR CONTROL-FAST RETRANSMISSION



ERROR CONTROL- LOST ACKNOWLEDGEMENT

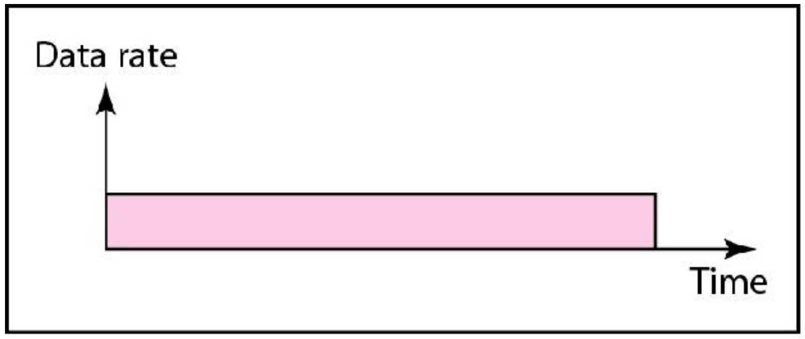


TCP CONGESTION CONTROL

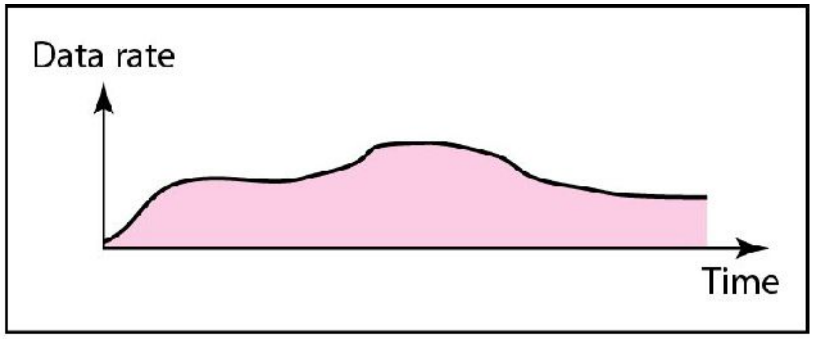
- Congestion in a network may occur if the load on the network (number of packets send to the network) is greater than the capacity of the network (number of packets a network can handle).
- Thus the main focus of congestion is **data traffic**
- The **three types** of traffic Profiles are:

1. Constant Bit Rate: the data rate does not change. Network can handle this type of traffic easily since it is predictable.
2. Variable Bit Rate: the rate of the data flow changes with time. But still manageable. Does not need reshaping.
3. Bursty traffic: the data rate changes suddenly in a very short time. Main reason for congestion and needs to be reshaped.

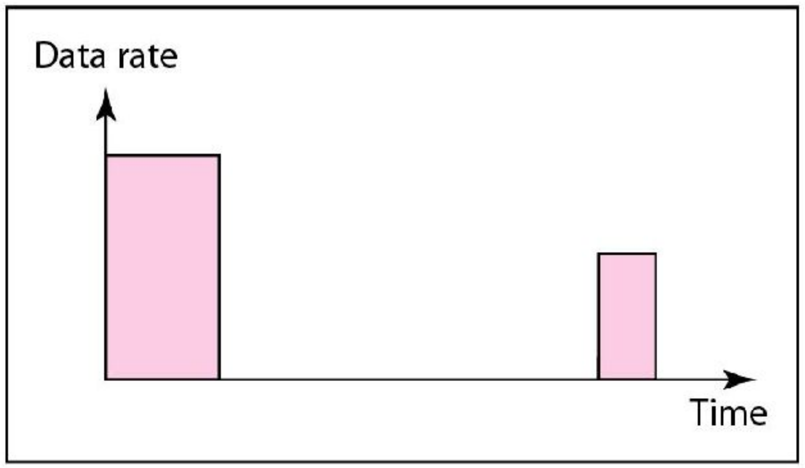
TCP CONGESTION CONTROL



a. Constant bit rate



b. Variable bit rate

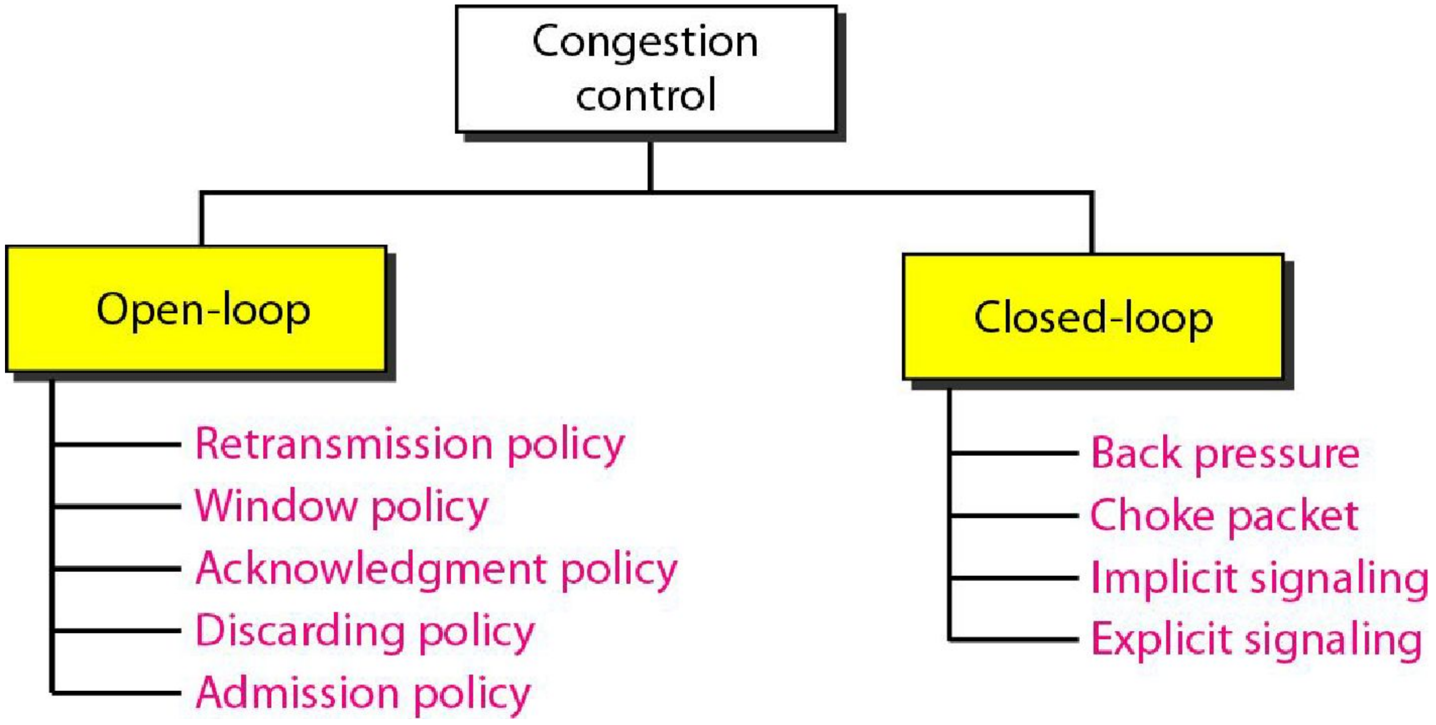


c. Bursty

TCP CONGESTION CONTROL

- Congestion control refers to techniques and mechanisms that can either prevent congestion or remove congestion.
- Congestion control mechanisms are divided into two categories:

1. Open-loop congestion control (prevention)
2. Closed-loop congestion control (removal)



OPEN LOOP CONGESTION CONTROL

Retransmission Policy

- Retransmission is sometimes unavoidable.
- If the sent packets are corrupted or lost, it needs to be retransmitted.
- This may increase congestion.
- Thus transport layer needs to implement good retransmission policy.
- The retransmission policy and retransmission timer must be designed to optimize efficiency and prevent congestion

Window Policy

- The window policy may also affect congestion.
- E.g.- the selective repeat window is better than Go-back-N window.

Acknowledgement Policy

- Sending ACK for every packet may make congestion worst.
- The receiver should use efficient policy for deciding when to send ACK.

OPEN LOOP CONGESTION CONTROL

Discarding Policy

- If the routers in the network could decide upon which packets to discard, this will also help in congestion avoidance.

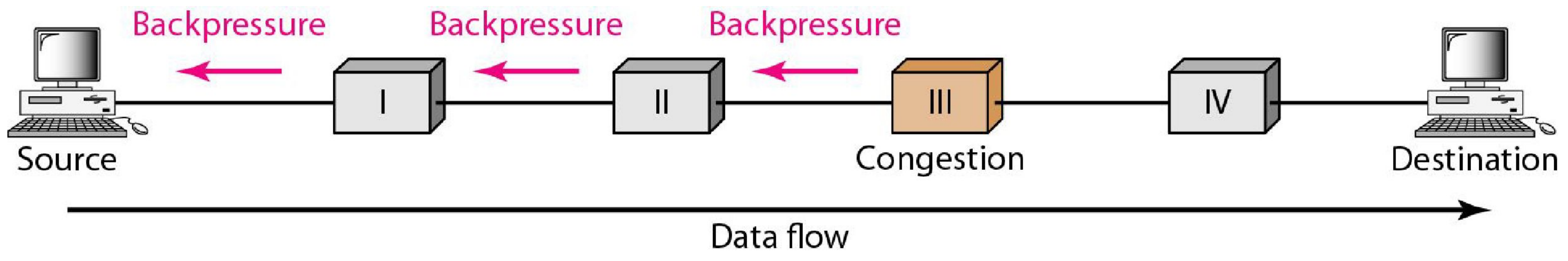
Admission Policy

- Used for congestion control in virtual-circuit networks.
- A router can deny establishing a virtual circuit connection if there is congestion in the network.

CLOSED LOOP CONGESTION CONTROL

Backpressure

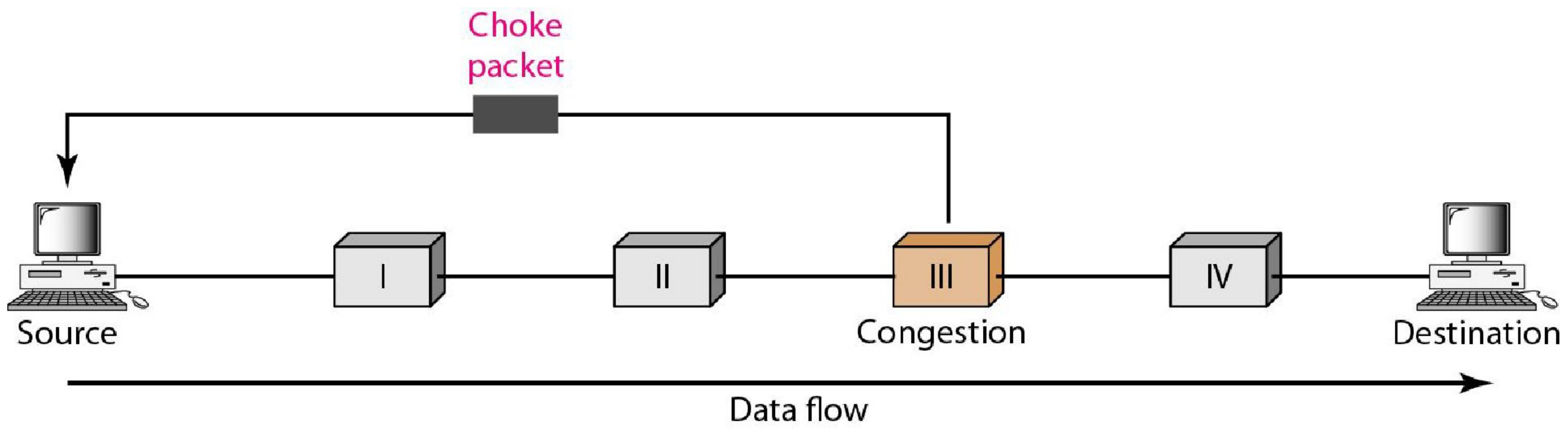
- Here the congested node stops receiving data from the previous node.
- This may cause the previous node to become congested, and so on.
- Thus the congestion is pressured from node to node till the source.
- This technique is applicable only in virtual circuit networks.



CLOSED LOOP CONGESTION CONTROL

Choke packet

- A choke packet is a packet sent by a node to the source to inform it of congestion.
- Thus in this case the congested router sends signal directly to the source.



CLOSED LOOP CONGESTION CONTROL

Implicit Signalling

- In this signaling there is no communication between the congested node and the source.
- The source guesses the congestion in the network from surrounding circumstances.
- E.g. – delay in acknowledgements

Explicit Signalling

- The congested node can explicitly send a signal to the source or destination.
- Here the signal is included in the packets that carry data and does not use separate packet as in choke packet method.
- Can either use backward signaling (to the source) or forward signaling (to the destination)

TCP CONGESTION CONTROL

Congestion Window

- A window management technique using ‘rwnd’ guarantees that the receiver never get overflowed.
- It also assures no end congestion.
- But what if intermediate buffers, buffers in routers get congested?
- Congestions can happen at either ends or in the middle of the communication path.
- Router are implemented at network layer, therefore its responsibility of network layer to handle congestion at routers level (congestion in the middle).
- ‘cwnd’ parameter helps in this problem

Window size = minimum (rwnd, cwnd)

TCP CONGESTION CONTROL

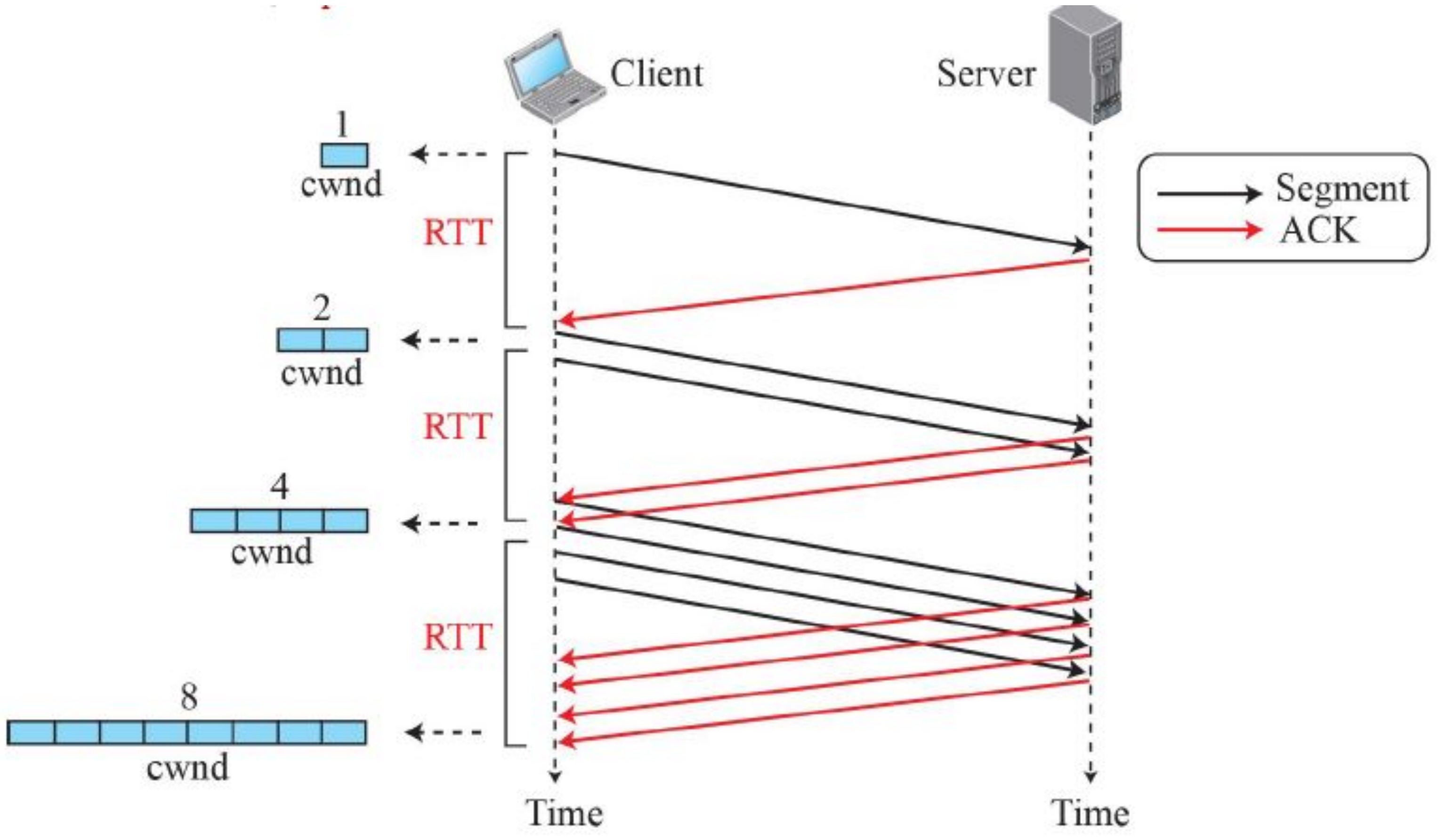
Congestion Policies

- TCP's general policy for handling congestion is based on three phases: slow start, congestion avoidance, and congestion detection.

SLOW START

- One of the algorithms used in TCP congestion control is called slow start.
- This algorithm is based on the idea that the size of the congestion window (cwnd) starts with one maximum segment size (MSS).
- The MSS is determined during connection establishment by using an option of the same name.
- The size of the window increases one MSS each time an acknowledgment is received.
- As the name implies, the window starts slowly, but grows exponentially.
- A threshold ‘ssthresh’ (slow start threshold) is maintained by sender.
- Most common value of ssthresh = 65,535 bytes

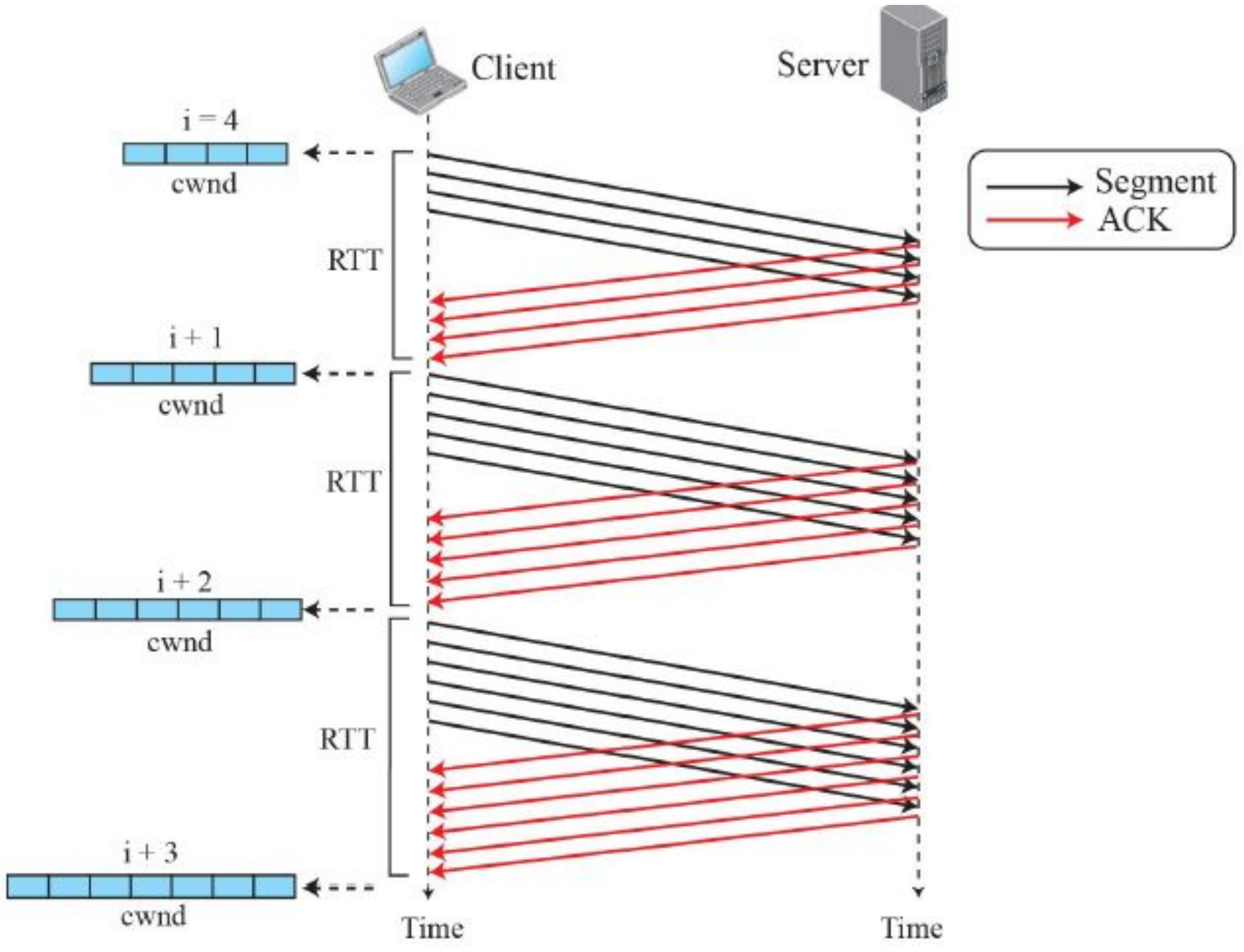
SLOW START


Client
Server
Segment
ACK
1
cwnd
RTT
2
cwnd
RTT
4
cwnd
RTT
8
cwnd
Time
Time

CONGESTION AVOIDANCE

- TCP defines another algorithm called congestion avoidance, which undergoes an additive increase instead of an exponential one.
- When the size of the congestion window reaches the slow-start threshold, the slow-start phase stops and the additive phase begins.
- In this algorithm, each time the whole window of segments is acknowledged (one round), the size of the congestion window is increased by 1.
- Note that cwnd size is much greater than 1.

CONGESTION AVOIDANCE



CONGESTION DETECTION

- If congestion occurs, the congestion window size must be decreased.
- The only way the sender can guess that congestion has occurred is by the need to retransmit a segment.
- However, retransmission can occur in one of two cases: when a timer times out or when three ACKs are received.
- In both cases, the size of the threshold is dropped to one-half, a multiplicative decrease.

CONGESTION DETECTION

- Most TCP implementations have two reactions:

1. If a time-out occurs, there is a stronger possibility of congestion; a segment has probably been dropped in the network, and there is no news about the sent segments.

In this case TCP reacts strongly:

a. It sets the value of the threshold to one-half of the current window size.
b. It sets cwnd to the size of one segment.
c. It starts the slow-start phase again.

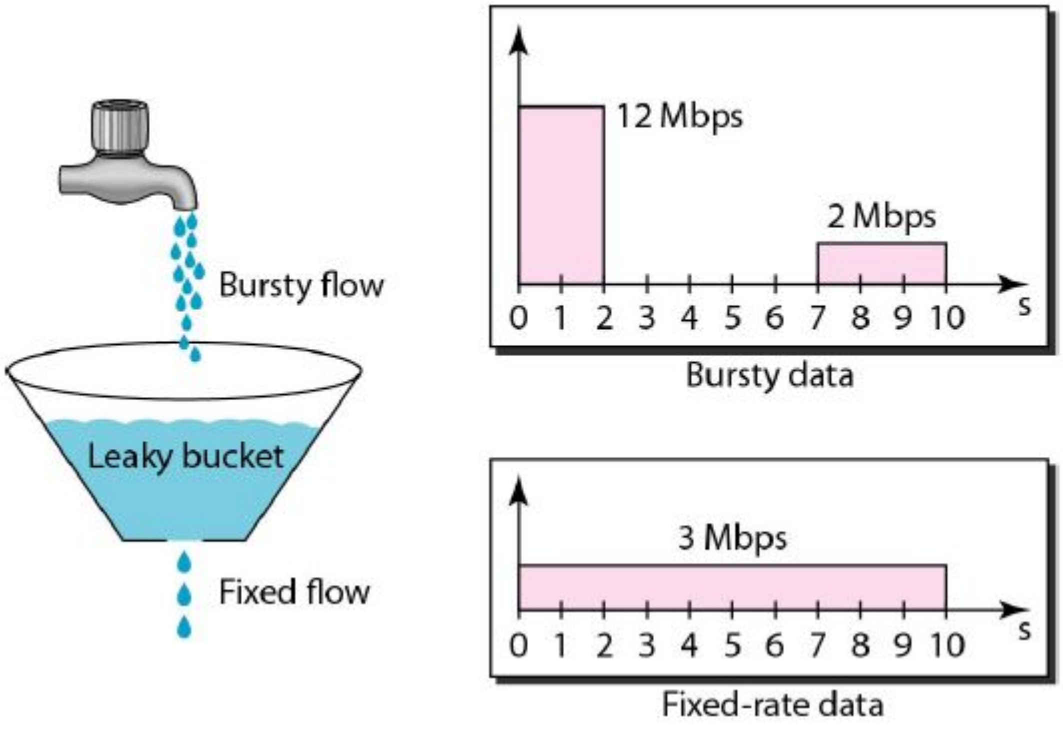
2. If three ACKs are received, there is a weaker possibility of congestion; a segment may have been dropped, but some segments after that may have arrived safely since three ACKs are received. This is called fast transmission and fast recovery. In this case, TCP has a weaker reaction:

a. It sets the value of the threshold to one-half of the current window size.
b. It sets cwnd to the value of the threshold (some implementations add three segment sizes to the threshold).
c. It starts the congestion avoidance phase

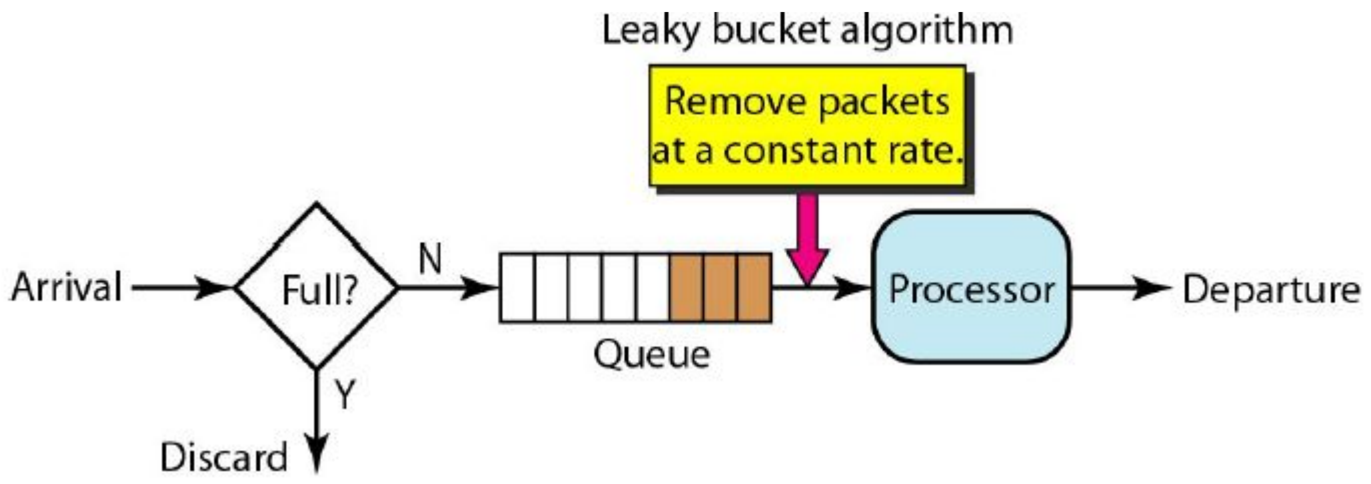
TRAFFIC SHAPING

- Traffic Shaping is a mechanism to control the amount and the rate of the traffic sent to the network.
- Technique widely used is Leaky Bucket Algorithm.
- If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket.
- The output rate does not depend on the input rate unless the bucket is empty.
- Thus the input rate can vary but the output rate is still constant.
- This technique can smooth out the bursty traffic in networking scenarios.
- Bursty chunks are stored in a bucket and sent out at an average rate.

LEAKY BUCKET ALGORITHM



LEAKY BUCKET ALGORITHM



TOCKEN BUCKET ALGORITHM

Disadvantage of Leaky Bucket

- It does not consider idle host condition.
- Remedy: Token Bucket Algorithm
- It accumulates credit for future in forms of Tokens.

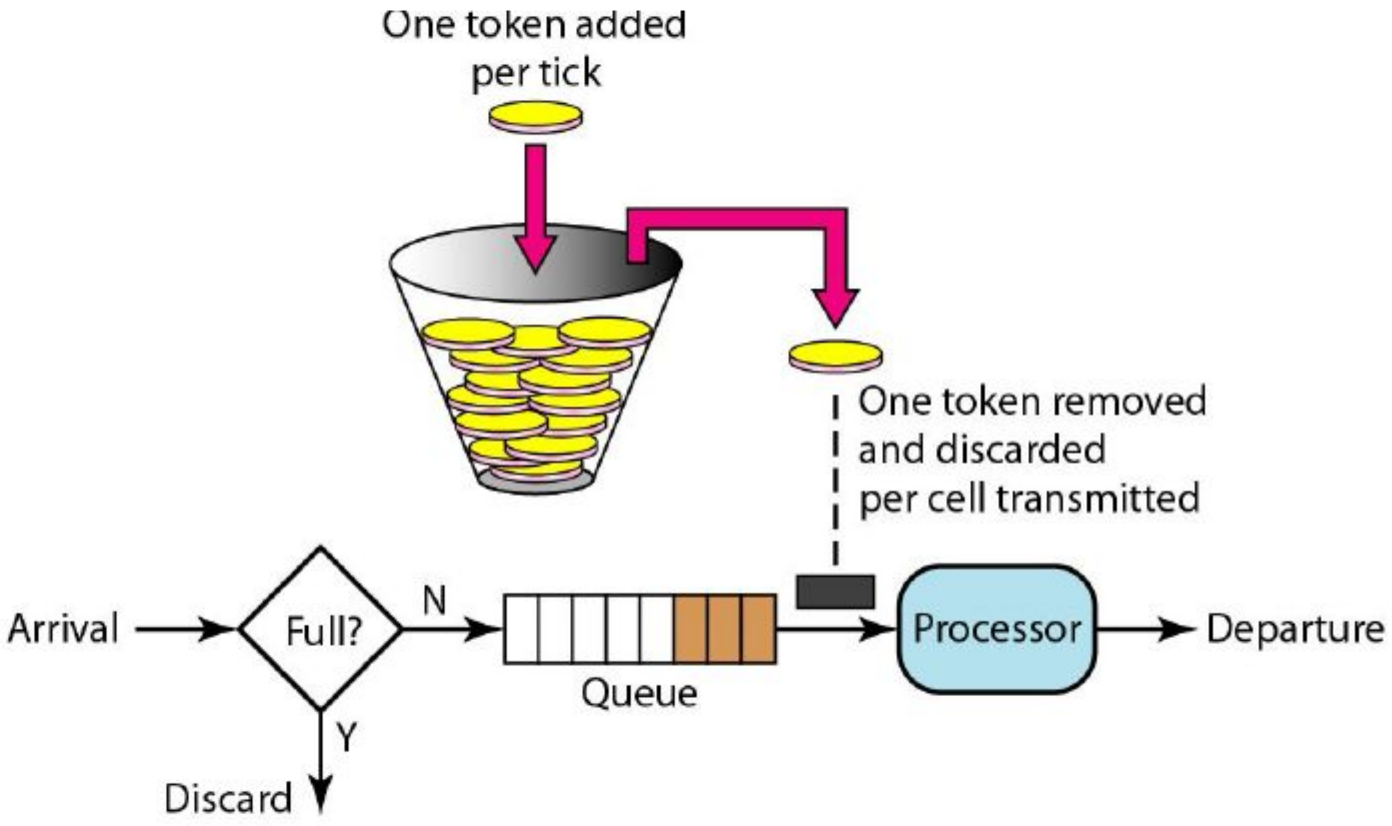
Working:

- Token bucket is a control mechanism that dictates when traffic can be transmitted, based on the presence of tokens in the bucket.
- Bucket :-an abstract container that holds aggregate network traffic to be transmitted.
- The bucket contains tokens, each of which can represent a unit of bytes or a single packet of predetermined size.

TOCKEN BUCKET ALGORITHM

- Tokens in the bucket are removed ("cashed in") for the ability to send a packet.
- The network traffic engineer specifies how many tokens are needed to transmit how many bytes.
- When tokens are present, a flow is allowed to transmit traffic.
- If there are no tokens in the bucket, a flow cannot transmit its packets. Therefore, a flow can transmit traffic up to its peak burst rate if there are adequate tokens in the bucket and if the burst threshold is configured appropriately.

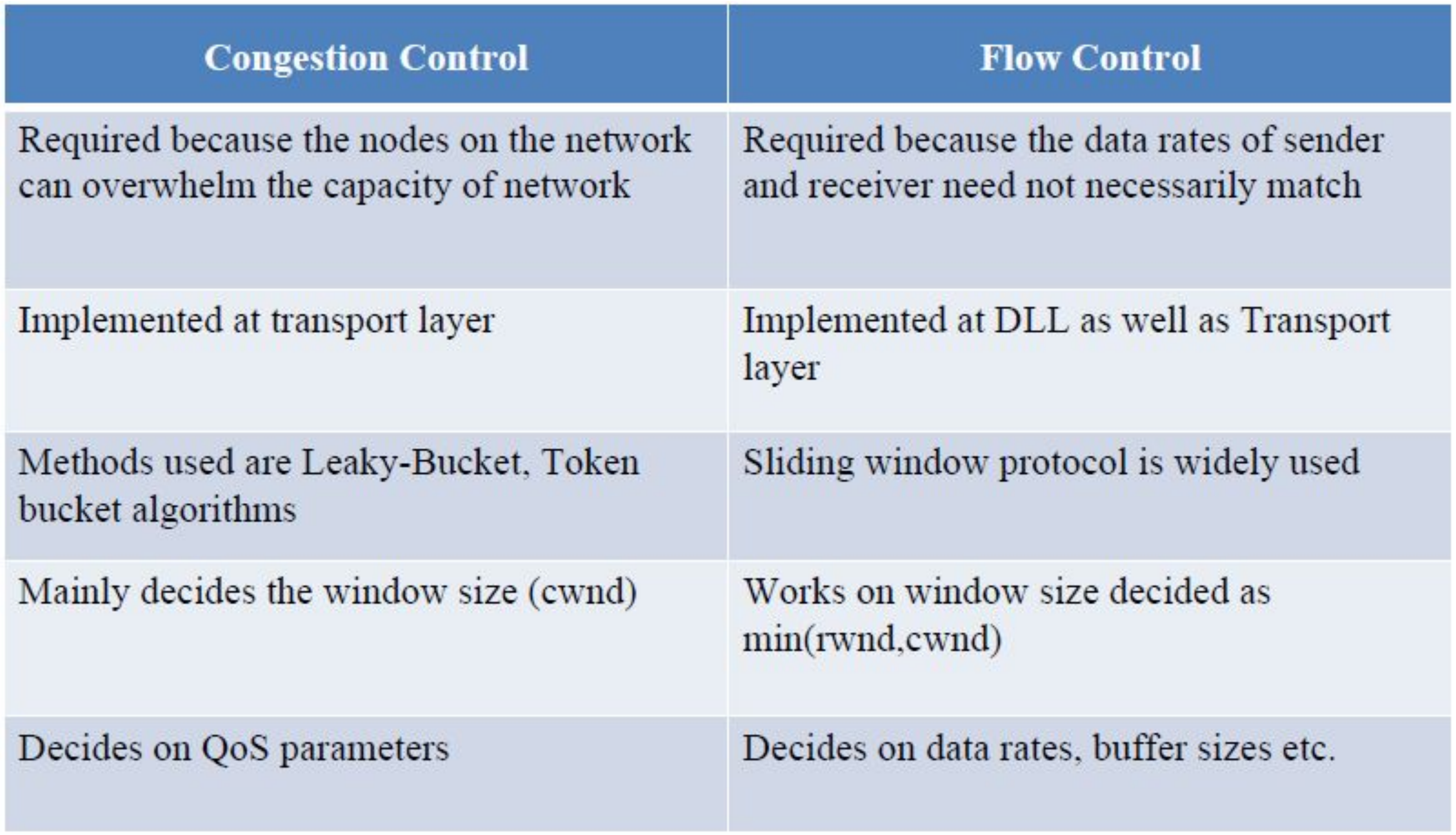
TOCKEN BUCKET ALGORITHM



TOCKEN BUCKET ALGORITHM

- For each tick of the clock, the system sends n tokens to the bucket.
- The system removes one token for every cell (packet or byte) of data send.
- The host can consume all these tokens in one tick (e.g. when bursty data arrives).
- Token bucket can be implemented by a counter.
- The token is initialized to zero
- When token is added, the counter is incremented by one
- Each time a unit of data is send, the counter is decremented by one
- When counter is zero, host can not send data

CONGESTION CONTROL vs FLOW CONTROL



Congestion Control	Flow Control
Required because the nodes on the network can overwhelm the capacity of network	Required because the data rates of sender and receiver need not necessarily match
Implemented at transport layer	Implemented at DLL as well as Transport layer
Methods used are Leaky-Bucket, Token bucket algorithms	Sliding window protocol is widely used
Mainly decides the window size (cwnd)	Works on window size decided as min(rwnd,cwnd)
Decides on QoS parameters	Decides on data rates, buffer sizes etc.

TCP TIMERS

To perform their operations smoothly, most TCP implementations use atleast four timers: retransmission, persistence, keepalive and TIME-WAIT.

Retransmission Timer

To retransmit lost segments, TCP employs one retransmission timer that handles the retransmission time-out (RTO), the waiting time for an acknowledgement of a segment.

Persistence Timer

To deal with zero window size advertisement, TCP needs another timer. When the sending TCP receives an acknowledgement with a window size zero, it starts a persistence timer. When the timer goes off TCP sends a special segment called a probe.

TCP TIMERS

Keepalive Timer

A keepalive timer is used in some implementations to prevent a long idle connection between two TCPs.

TIME- WAIT Timer

The TIME –WAIT timer is used during connection termination. Sender starts the time wait timer after sending the ACK for the second FIN segment. It allows to resend the final acknowledgement if it gets lost. It prevents the just closed port from reopening again quickly to some other application.

SESSION LAYER

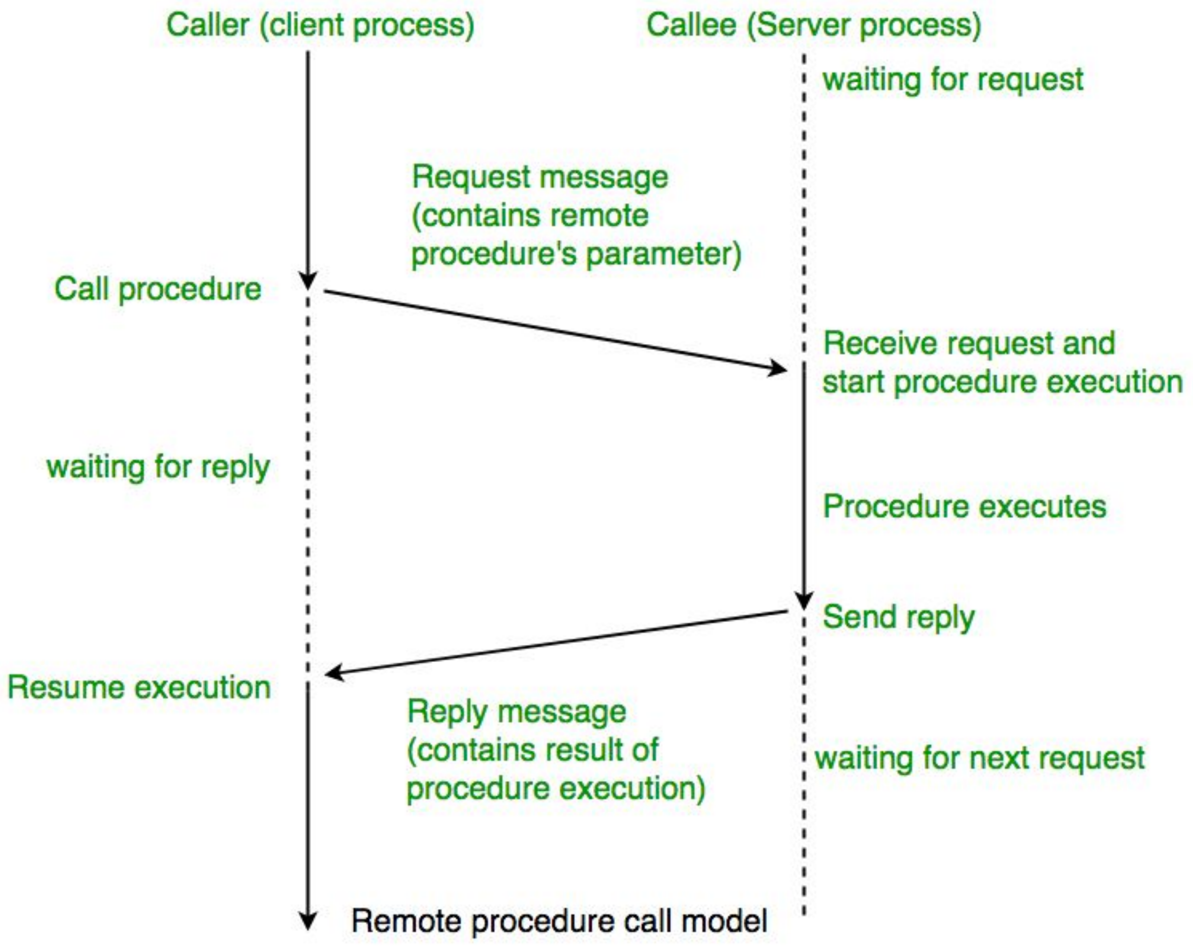
SESSION LAYER

- It is a network dialog controller.
- It establishes, maintains and synchronizes the interaction among communicating sessions.

Responsibilities:

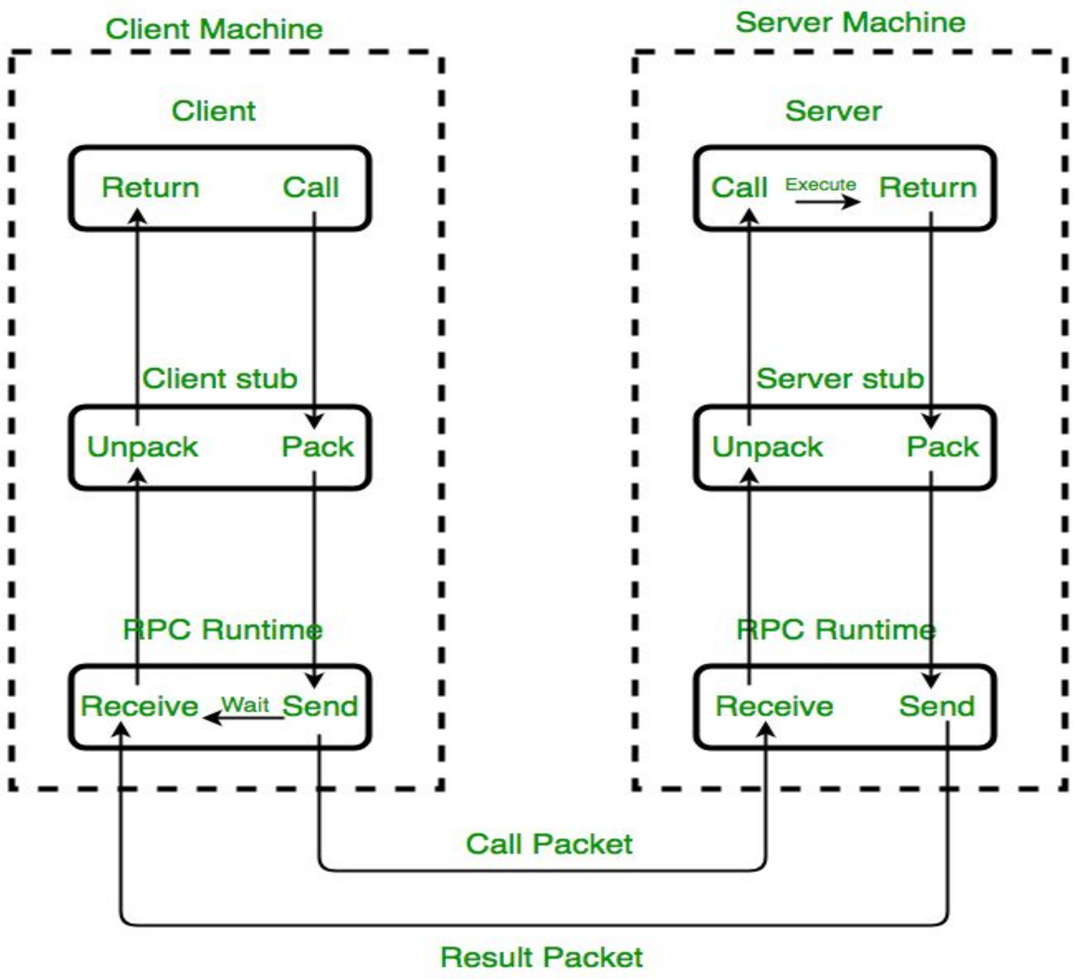
- Dialog control – use half or full duplex mode for dialog
- Synchronization – support check points or synchronization points

REMOTE PROCEDURE CALL



Remote procedure call model

REMOTE PROCEDURE CALL



Implementation of RPC mechanism

REMOTE PROCEDURE CALL

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(pack) the parameters into a 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(unpack) 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, 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.

Thank you …