Transport Layer

- It is responsible for process-to-process delivery of the entire message.
- TL looks after the delivery of entire message considering all its packets & make sure that all packets are in order. On the other hand n/w layer treated each packet independently.
- At the receiver side, TL provides services to application layer & takes services form n/w layer.
- At the source side, TL receives message from upper layer into packets and reassembles these packets again into message at the destination.

Transport Layer

- Transport Layer provides two types of services:

Connection Oriented Transmission: In this type of transmission the receiving devices sends an acknowledge back to the source after a packet or group of packet is received. It is slower transmission method.

Connectionless Transmission: In this type of transmission the receiving devices does not sends an acknowledge back to the source. It is faster transmission method.

Functions of Transport Layer

- **Segmentation of message into packet & reassembly of packets into message.**
- **Port addressing:** Computers run several processes. TL header include a port address with each process.
- **Flow Control:** Flow control facility prevents the source form sending data packets faster than the destination can handle.
- **Error control**: TL ensures that the entire message arrives at the receiving TL without error.

Contents

Transport Layer overview

- Convert data from upper layer (Application) to segments
- Transport layer is responsible for process-to process delivery and congestion control
- Distinguish a particular process in a system using port number
- 3 type of delivery mechanisms available TCP, UDP,SCTP
- Connection oriented (TCP) and connection less(UDP)
- Reliable (TCP) and unreliable (UDP) services
- Port address provide process to process delivery

1. Process to Process Delivery

- The transport layer is responsible for process-to-process delivery
- Types of data delivering
  1. Node to node delivery (data link Layer)
  2. Host to host delivery (Network Layer)
  3. Process to process delivery (Transport Layer)
- Process to process delivery is needed because many process will be running in one host simultaneously
- To identify, from which process is sending the request source port address is assigned
- To identify, to which process request is send is destination address is assigned

1. Process to Process Delivery

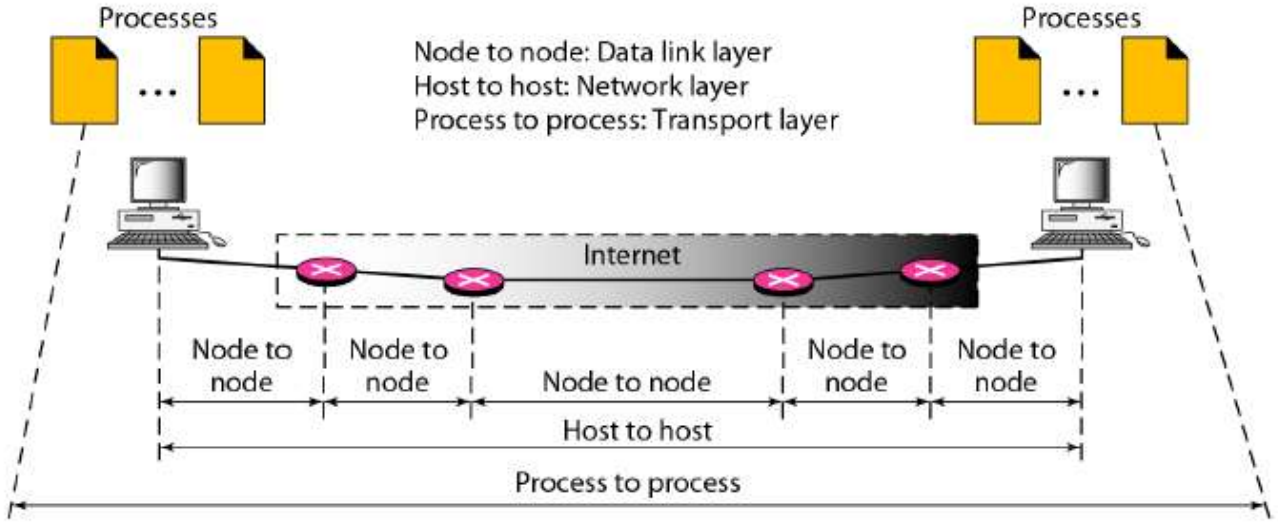
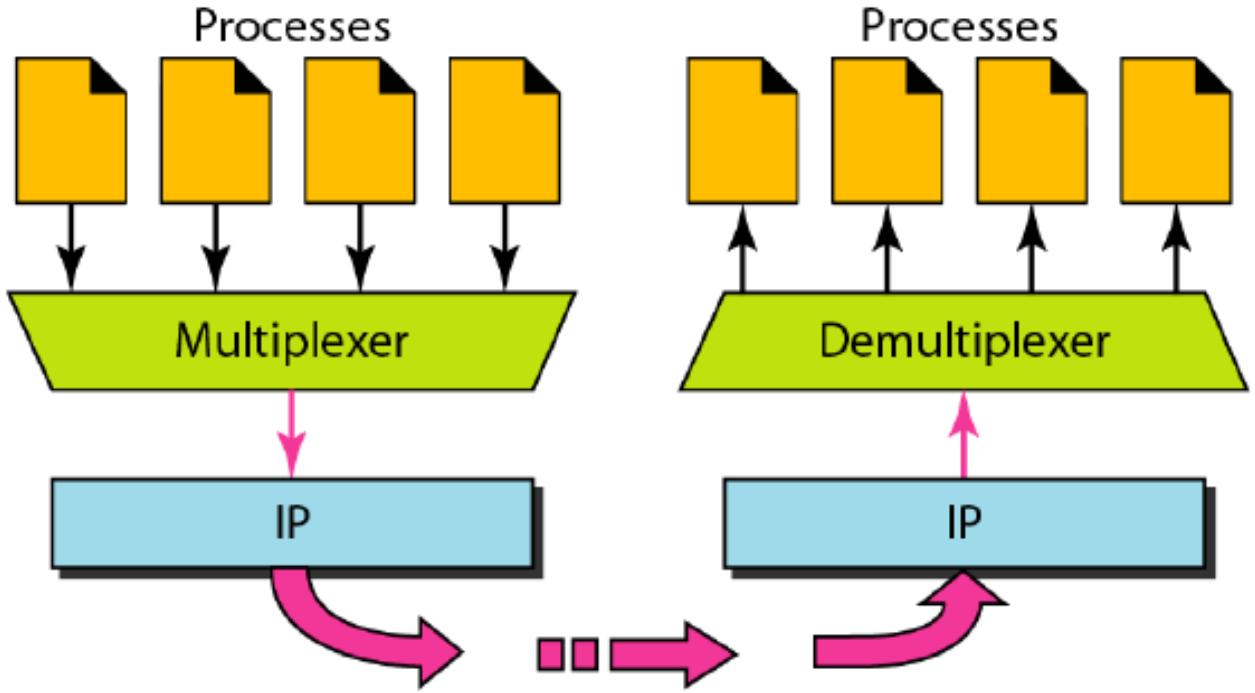


Figure: Process to Process Delivery

- Two processes communicate in a client/server relationship

1.1. Multiplexing & De-multiplexing

- The addressing mechanism allows multiplexing and De-multiplexing by the transport layer



1.1.1. Multiplexing

- At the sender site, there may be several processes that need to send packets. However, there is only one transport layer protocol at any time. This is a many-to-one relationship and requires multiplexing.
- The protocol accepts messages from different processes, differentiated by their assigned port numbers. After adding the header, the transport layer passes the packet to the network layer

1.1.2. De-multiplexing

- At the receiver site, the relationship is one-to-many and requires De-multiplexing. The transport layer receives datagrams from the network layer.
- After error checking and dropping of the header, the transport layer delivers each message to the appropriate process based on the port number

1.2. Client Server

- Most common way to achieve process-to-process communication in internet is through the client/server paradigm.
- A process on the local host, called a client, needs services from a process usually on the remote host, called a server
- Operating systems today support both multiuser an multiprogramming environments.
- 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

1.2. Client Server

- Here 2 types of address is needed for communication **Host Address and Process Port number**
- For communication, we must define the following:
  1. Local host : IP address of Client
  2. Local process : Port number of client
  3. Remote host: IP address of Server
  4. Remote process: Port address of server

1.2. Client Server

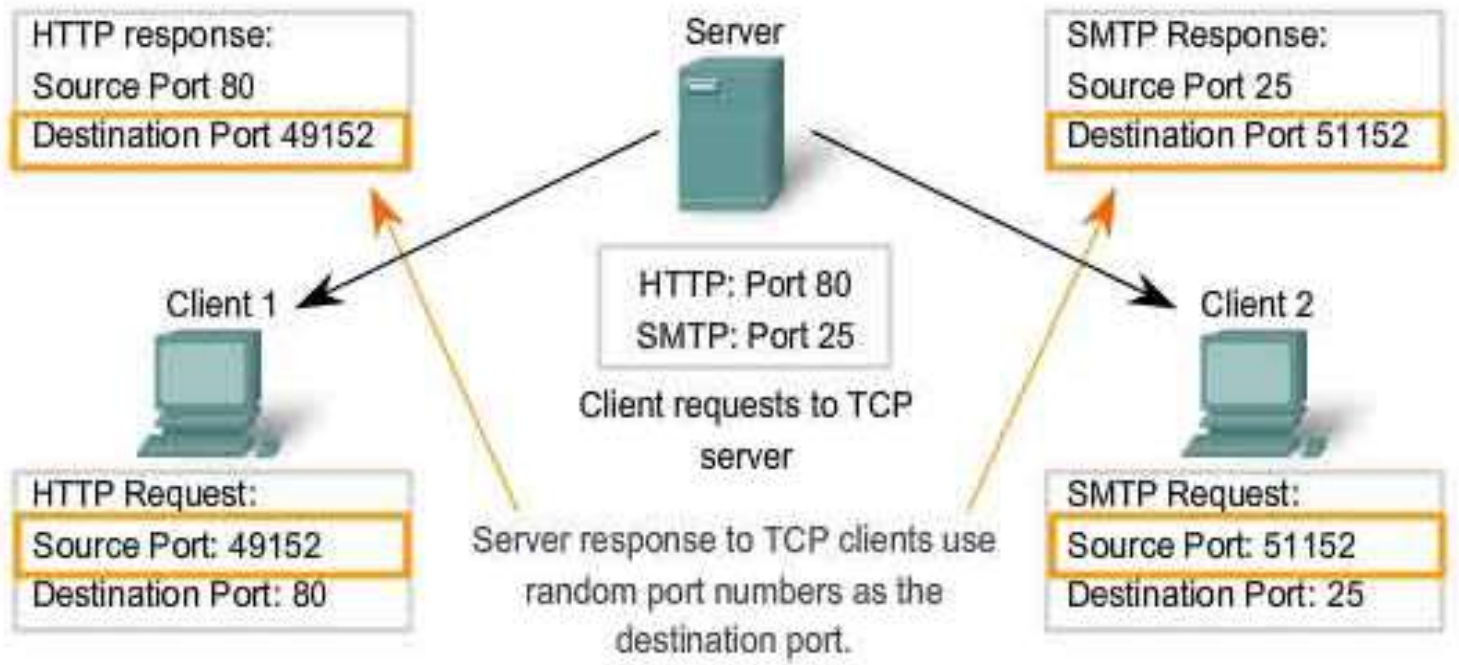


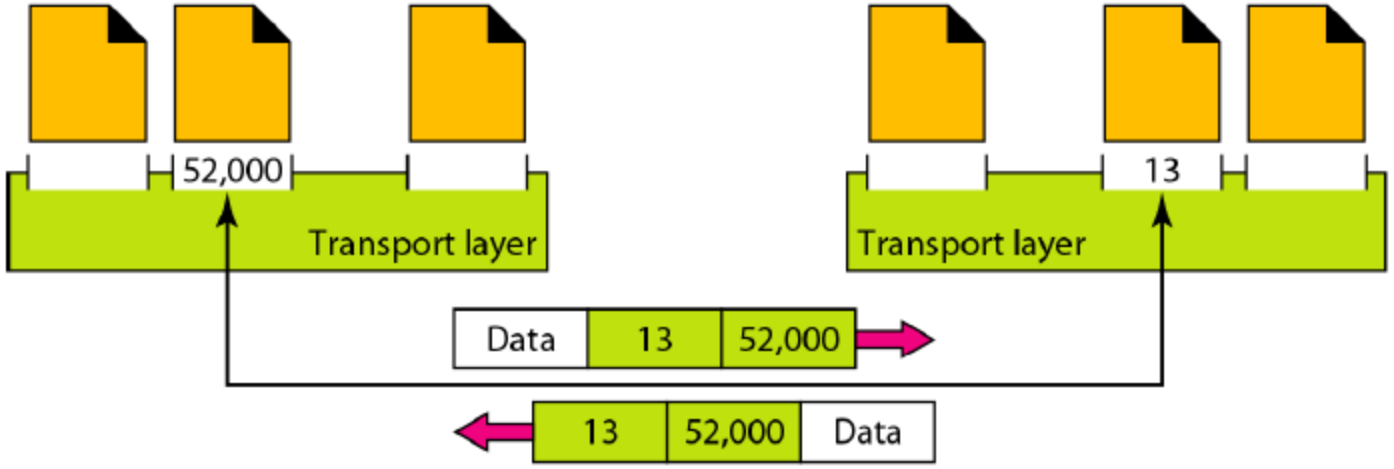
Figure Clie ent 2 sending

1.3. Port Numbers

- Port Address (Number) is needed to choose among multiple processes running on the destination host
- The destination port number is needed for delivery and the source port number is needed for the reply
- Port number is 16 bit integer
- Range between 0-65535
- Source port number is chosen randomly(Dynamic range)
- Destination port is chose based on application(For ex, HTTP-80, FTP-20,21)

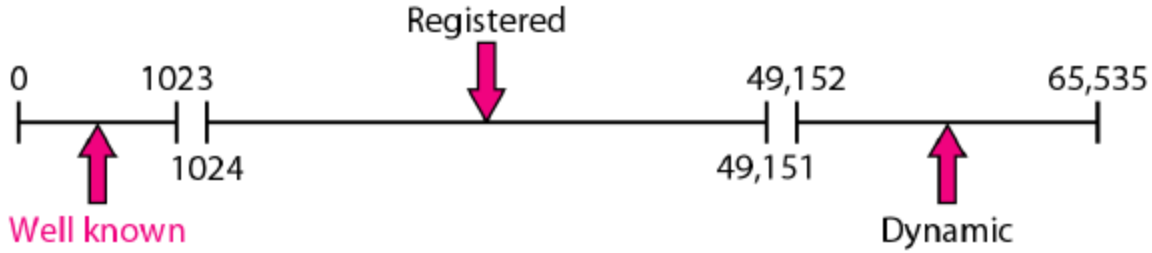
1.3. Port Numbers

- Data is sent from system one have process with port number 52000 and other system have a process with port number 13



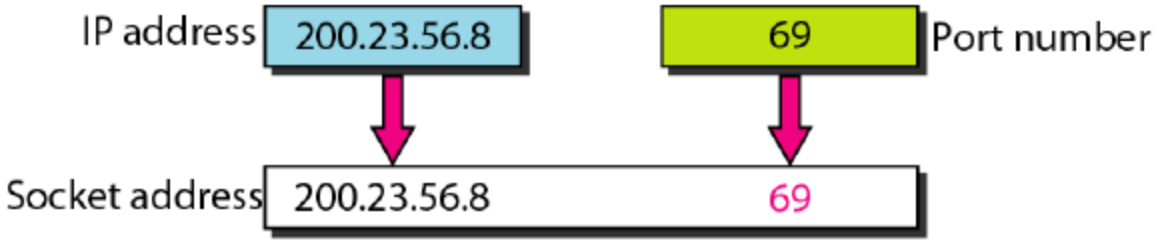
1.3.1. Port numbers- IANA ranges

- The lANA (Internet Assigned Number Authority) has divided the port numbers into three ranges:
- Well known ports (Range between 0-1023): Controlled by IANA
- Registered ports (Range between 1024-49151): not controlled by IANA but can be register to prevent duplication
- Dynamic(Private) ports (Range between 49152-65535): The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used by any process



1.4. Socket Address

- Process-to-process delivery needs **two identifiers**, **IP address** and the **port number**, at each end to make a connection
- Combination of an **IP address and a port number** is called a **socket address**
- The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely
- A transport layer protocol (TCP/UDP)needs a pair of socket addresses: the source socket address and the destination socket address



2.1 Connection Less(UDP)

- 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
- UDP is connectionless

2.2 Connection Oriented(TCP)

- Connection is first established between the sender and the receiver
- Data are transferred
- Connection is released
- TCP and SCTP are connection-oriented protocols

3 Reliable vs Unreliable

Reliable

- Have error control
- Have Flow control
- Connection oriented
- Have ACK and sequence number
- TCP &SCTP

Unreliable

- No error control
- No Flow control
- Connectionless
- No ACK and sequence number
- UDP

4. Transmission Control Protocol(TCP)

- Connection oriented & Reliable Service
- TCP uses flow and error control mechanisms at the transport level
- TCP uses port numbers for communication

Well Known TCP Ports

- FTP- 20(data), 21(control)
- TELNET- 23
- SMTP-25
- DNS-53
- HTTP-80
- RPC-111

4.1. TCP Features

- Stream delivery (No boundary defined)
- Full duplex communication
- Numbering systems
  - TCP software keeps track of the segments being transmitted or received, there is no field for a segment number value in the segment header
  - The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number
- Sequence Number:
  - The sequence number for each segment is the number of the first byte (randomly assigned) carried in that segment

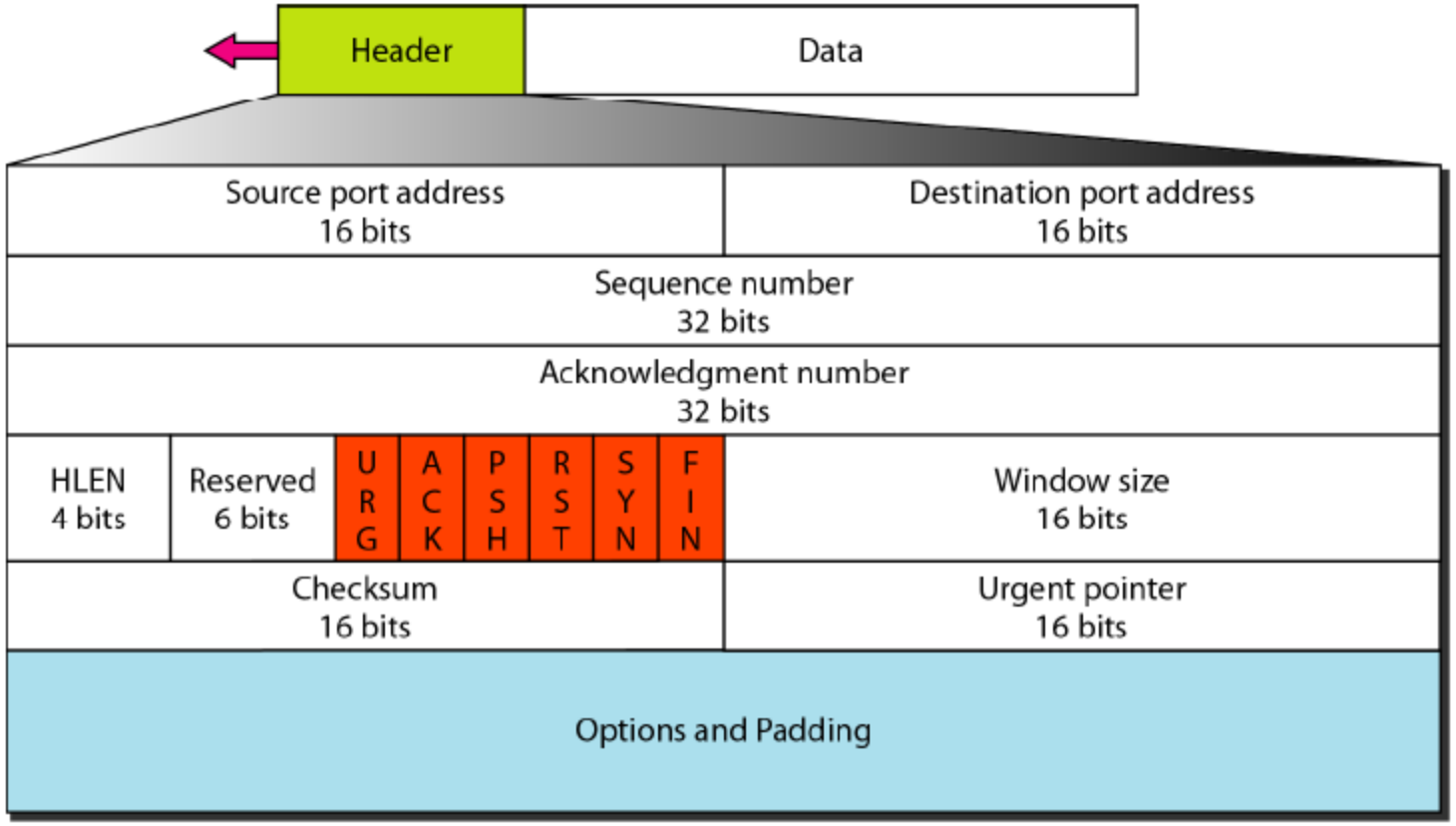
4.1. TCP Features

- Acknowledgment number:
  - Full duplex sequence number and Acknowledgment in same segment
  - If receiver receives **x** bytes Ack number is equal to **x+1**
- Flow control& Error control:
  - TCP uses byte oriented flow & error control
- Congestion control
  - Also control the congestion in the network

4.1.1 TCP Sequence numbering

- Suppose a TCP connection is transferring a file of 5000 bytes. The first byte is numbered 1O,001. each carrying segment carry 1000 bytes The following shows the sequence number for each segment:
- 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)

4.2. TCP Segment Format



4.2. TCP Segment Format

- Source port address: Port address of sender
- Destination port address: port address of receiver
- Sequence number: 32 bit integer usually represented by first bytes number
- Acknowledge number:32 bit integer, x bytes are received Ack will be x+1
- Header Length (HLEN) : usually between 20-60 bytes
- Reserved: future use

4.2. TCP Segment Format

- Flag (6 bits):
    1. URG: Urgent pointer is valid
    2. ACK: Acknowledgment is valid
    3. PSH: Request for push
    4. RST: Reset the connection
    5. SYN: Synchronize sequence numbers
    6. FIN: Terminate the connection
- Window size
    - This field defines the size of the window, in bytes, that the other party must maintain. 16 bit so maximum size of the window is 65,535 bytes. Usually determined by reciever

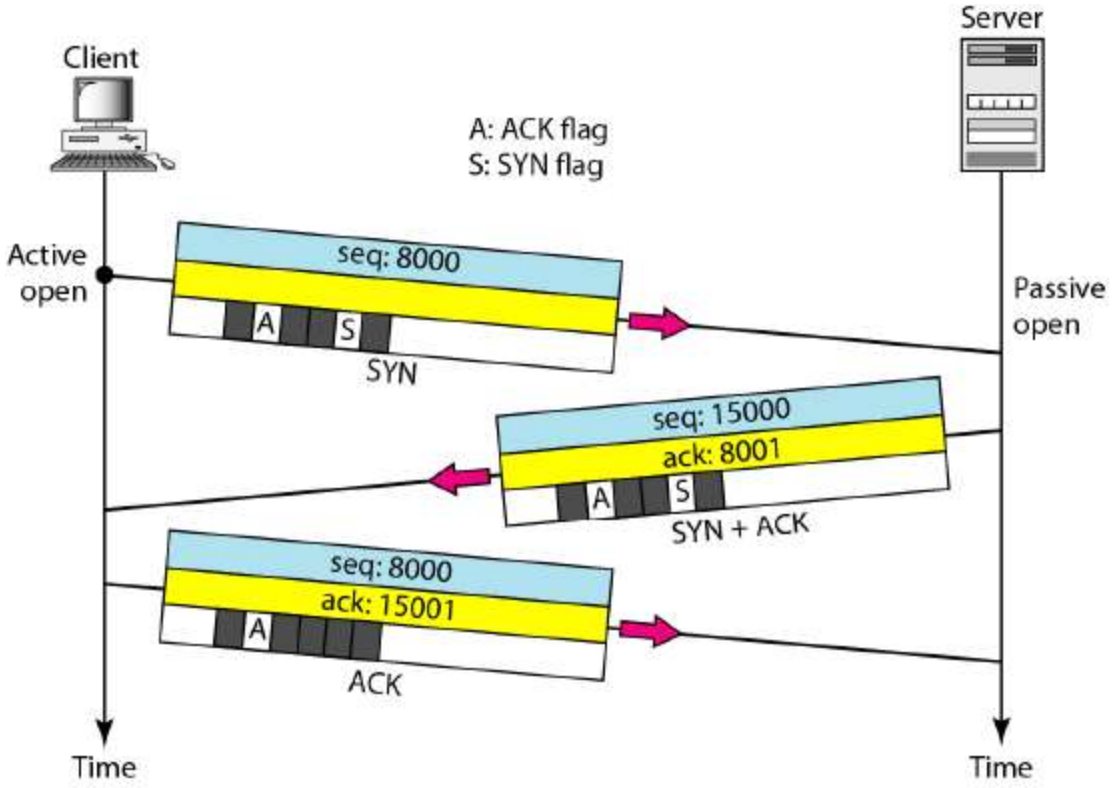
4.2. TCP Segment Format

- Urgent pointer.
  - This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data.
- Options
  - There can be up to 40 bytes of optional information in the TCP header.

4.3. 3-Way Handshaking

- TCP is connection oriented which means before data transfer connection should be established
- Step in TCP communication
  1. Connection establishment
  2. Data Transfer
  3. Connection termination
- Connection is established with help of 3 way handshaking protocol
- Connection termination is also done with 3 way handshaking protocol

4.3.1. Connection Establishment (Handshaking)



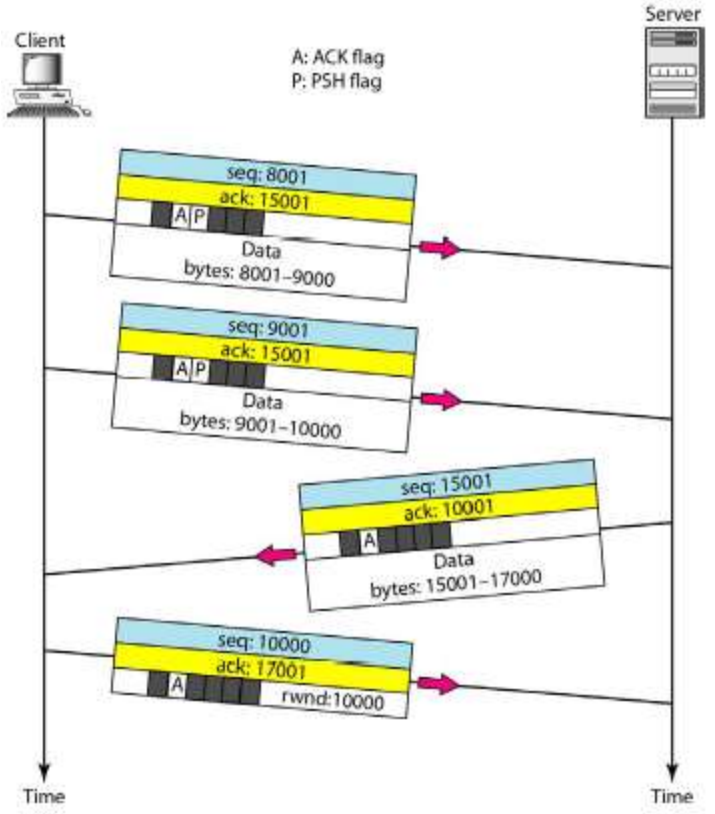
4.3.1. Connection Establishment (Handshaking)

- SYN-Synchronization message used for connection establishment
- ACK- Acknowledgement

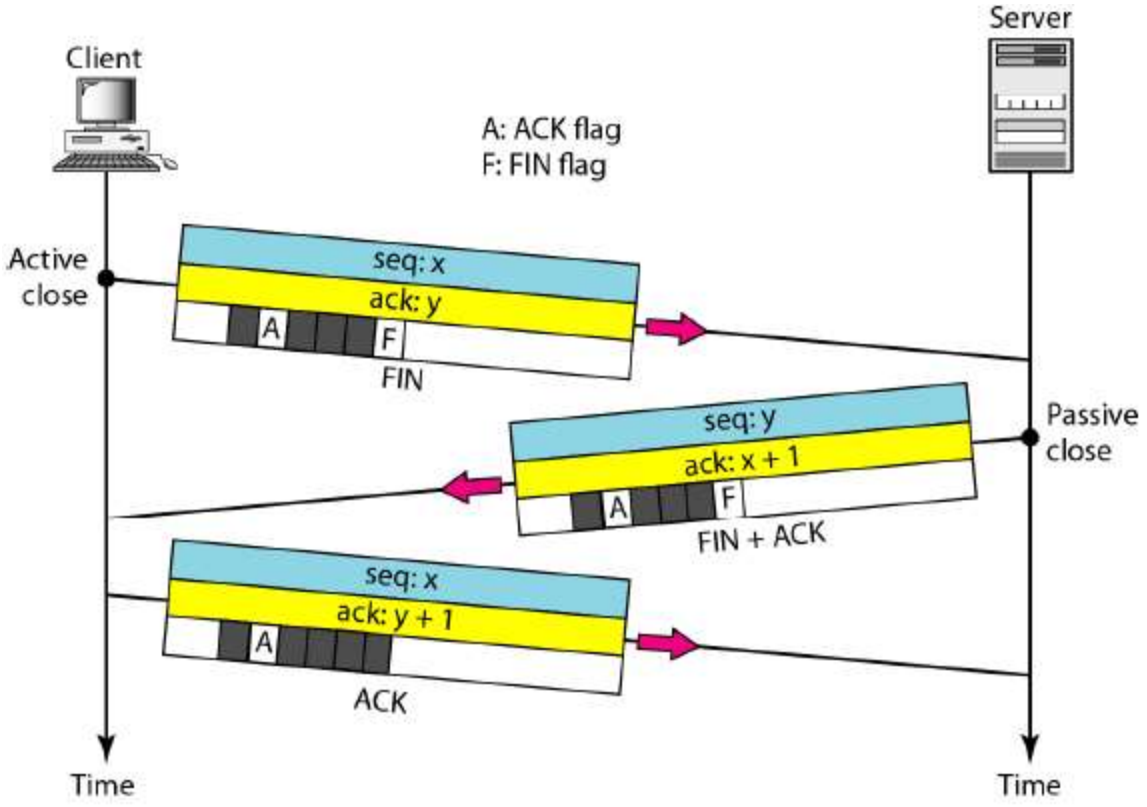
Steps

1. First client send SYN message to server
2. Server responds ACK to SYN and SYN to client
3. Client responds ACK to SYN and connection is established

4.3.2. Data Transfer(ACK Number + SEQ Number)



4.3.3. Connection Termination (Handshaking)



4.3.3. Connection Termination (Handshaking)

- FIN(Finish) – Connection Termination
- ACK- Acknowledgement

Steps

1. First client send FIN message to server
2. Server responds ACK to FIN from client and send FIN to client
3. Client responds ACK to FIN and connection is established

5. User Datagram Protocol(UDP)

- Connection less & Unreliable Service
- No flow and error control mechanisms at the transport level
- UDP uses port numbers for communication

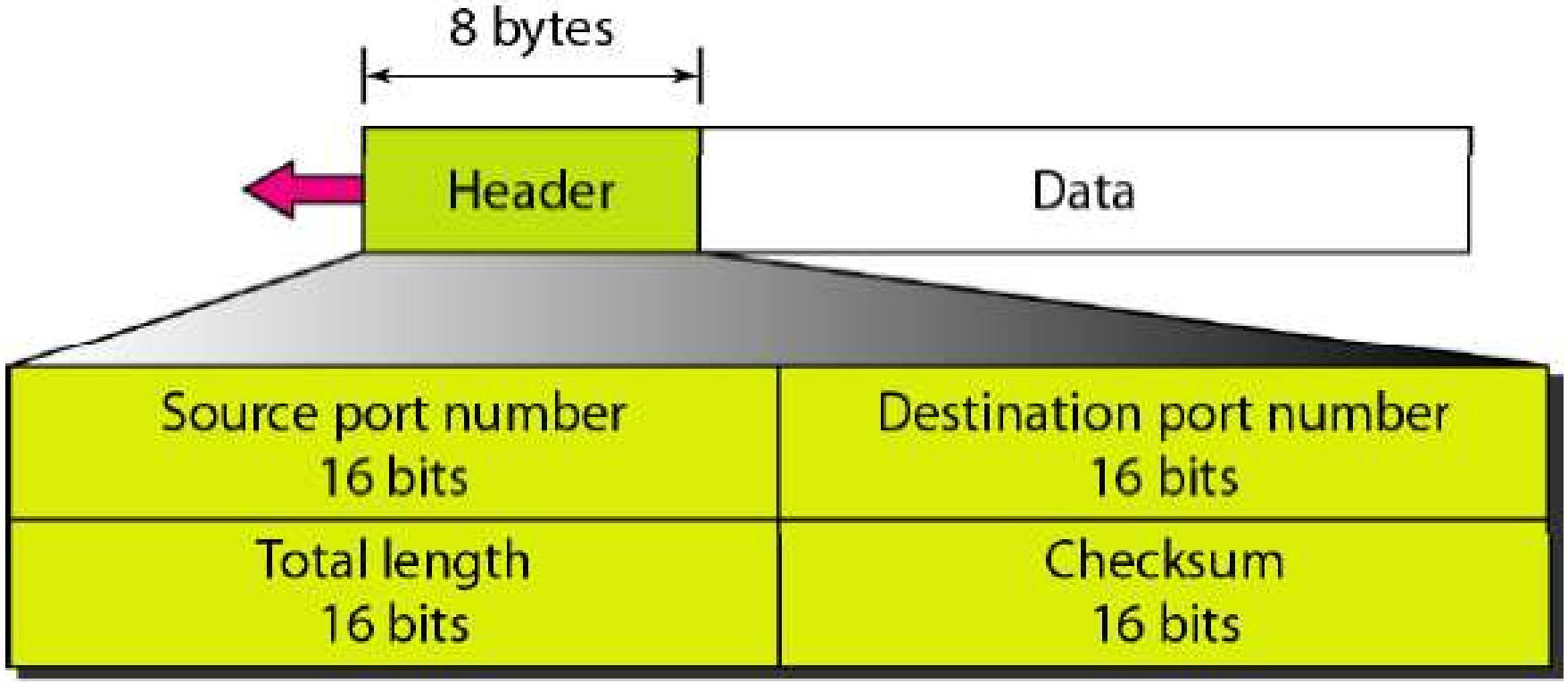
Well Known TCP Ports

- Daytime-13
- DNS-53
- TFTP-69
- RPC-111
- NTP-123

5.1. Features of UDP

- Connectionless:
  - No connection is established before data transfer
  - user datagram(segment) sent by UDP is an independent datagram
- Flow and Error Control:
  - No error flow control (Only optional error checking)
  - No retransmission
  - No acknowledgments

5.2. UDP Segment



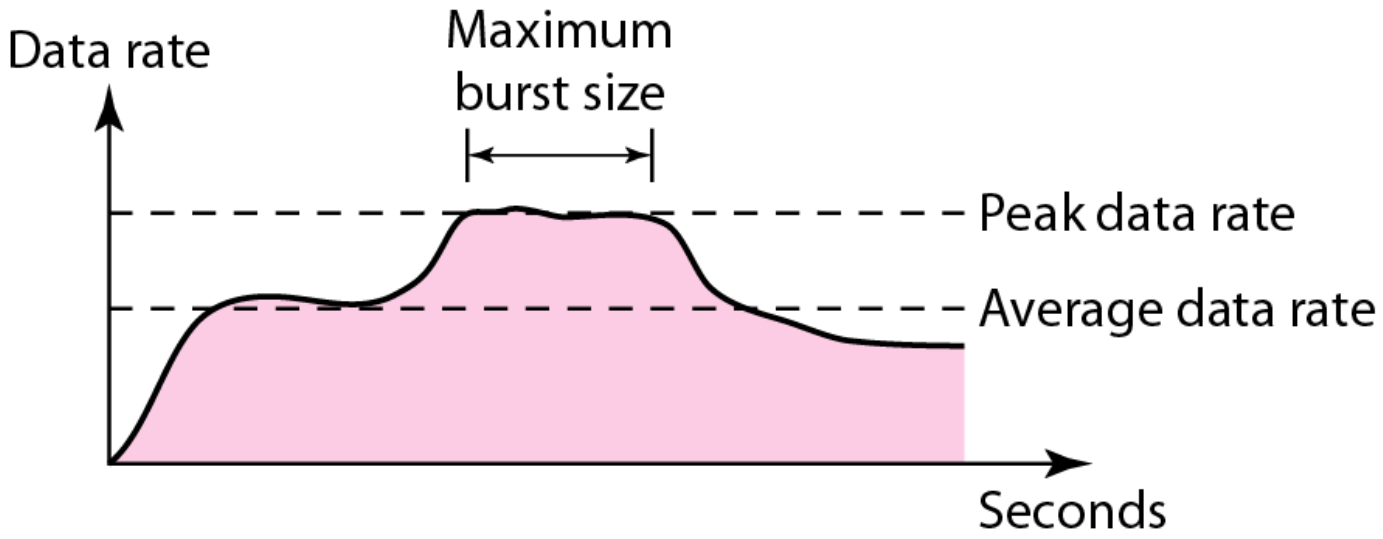
5.2. UDP Segment

- Source port number(16 bits)
  - This is the port number used by the process running on the source host
  - Range 0-65535
- Destination port number(16 bits)
  - This is the port number used by the process running on the destination host
  - Range 0-65535
- Length
  - Defines the total length of the user datagram, header plus data.
  - Length ranges between 8 to 65,535 bytes
- Checksum
  - For error checking

Contents

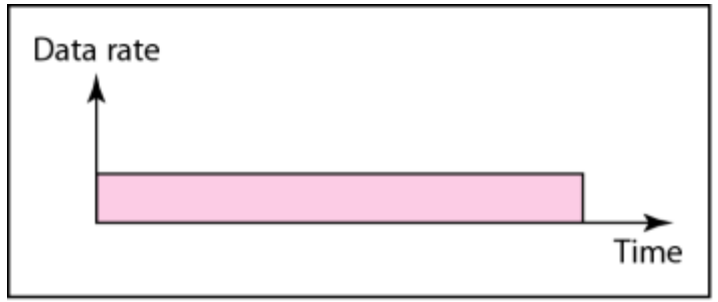
Introduction

- The main focus of congestion control and quality of service is data traffic
- Traffic descriptors are qualitative values that represent a data flow

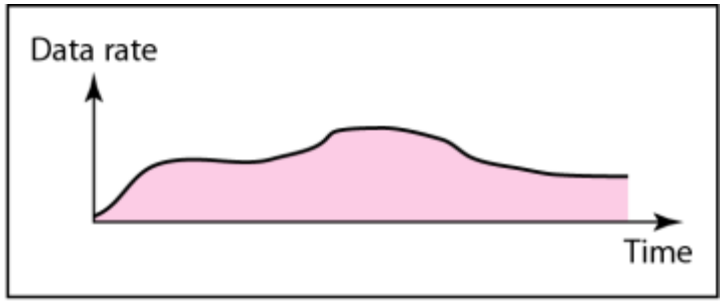


Introduction

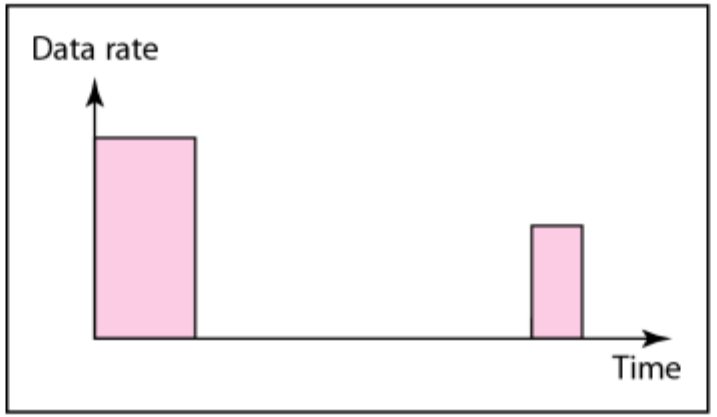
- 3 Traffic Profiles



a. Constant bit rate



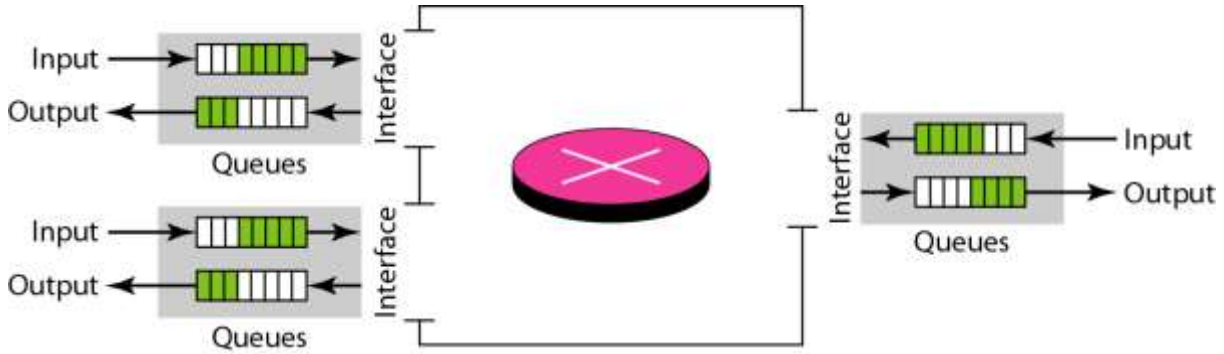
b. Variable bit rate



c. Bursty

1. Congestion

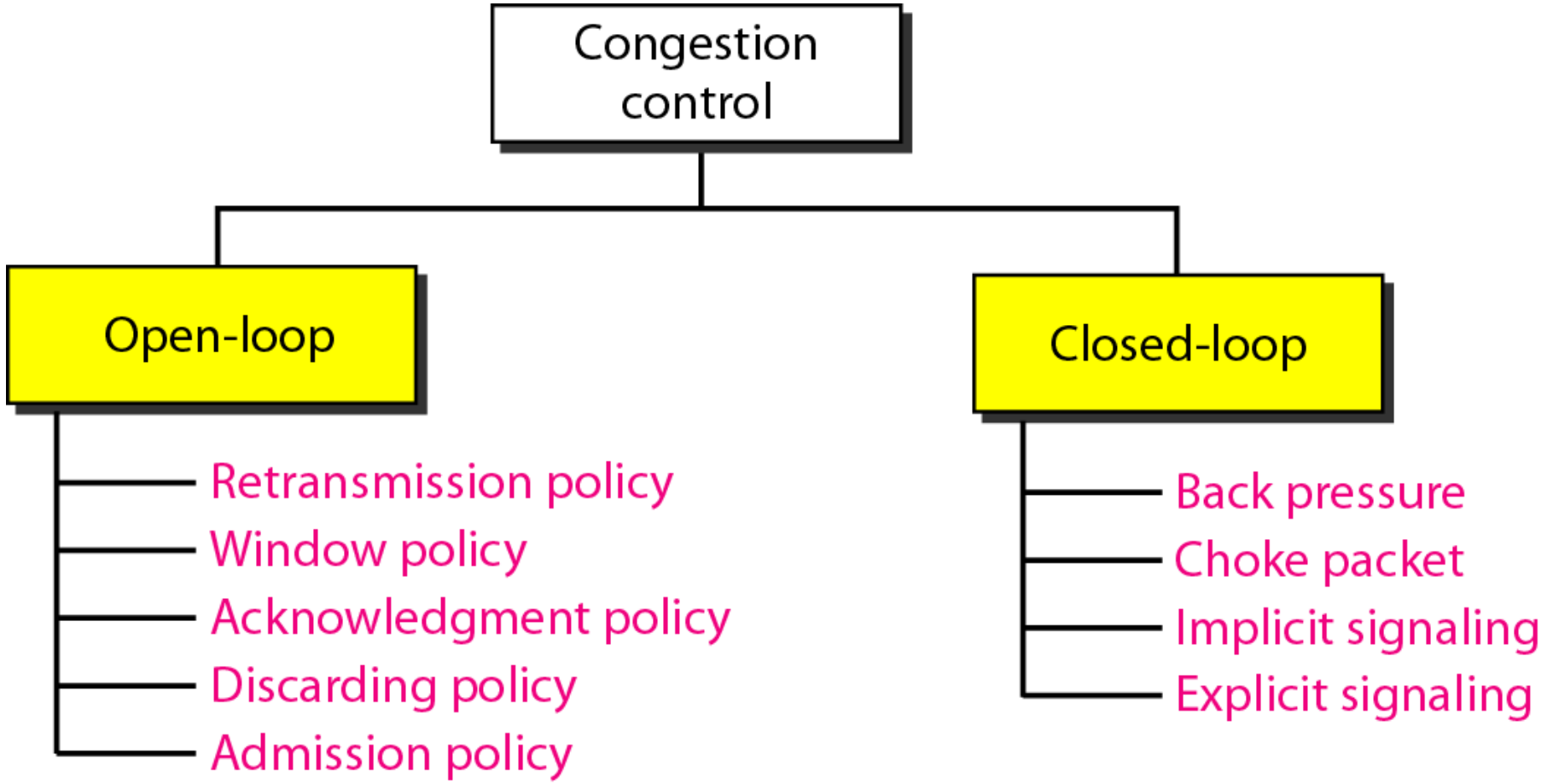
- Load of network → The number of packets sent to the network
- Capacity of network → The number of packets that can be handle
- Congestion occurs when the load of network is greater than capacity of network
- Congestion occurs because of the following factor
    1. Processing capacity of router
    2. No of Packets in input and output interface



2. Congestion Control

- Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened
- Congestion control mechanisms into two broad categories:
  1. Open-loop congestion control (prevention)
  2. Closed-loop congestion control (removal)

2. Congestion Control



2.1. Open Loop Congestion

- Congestion Prevention mechanism
- Policies are applied to prevent congestion **before it happens**
- Congestion control is handled by either the source or the destination
    1. Retransmission policy
    2. Windowing policy
    3. Acknowledge policy
    4. Discard policy
    5. Admission policy

2.1.1. Retransmission policy

- Retransmission is sometimes unavoidable
- If the sender feels that a sent packet is **lost or corrupted**, the packet needs to be **retransmitted**
- Retransmission in general may increase congestion in the network
- Good retransmission policy can prevent congestion
- The retransmission policy and the retransmission timers must be designed to **optimize efficiency** and at the same time **prevent congestion**

2.1.2.Windowing policy

- The Selective Repeat window is better than the Go-Back-N window for congestion control
- In the *Go-Back-N* window, when the timer for a packet times out, several packets may be resent, even if some may have arrived safe and sound at the receiver
- The Selective Repeat window, on the other hand, tries to send the specific packets that have been lost or corrupted

2.1.3. Acknowledge Policy

- Acknowledgments are also part of the load in a network
- The acknowledgment policy imposed by the receiver may also affect congestion
- If receiver acknowledges every packet there is change for congestion in network(Stop and wait)
- If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion. (Sliding window like **Go back-N** and **Selective repeat**)

2.1.4. 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
- According to priority packet is discarded
- Less sensitive packets should be discarded

2.1.5. Admission Policy

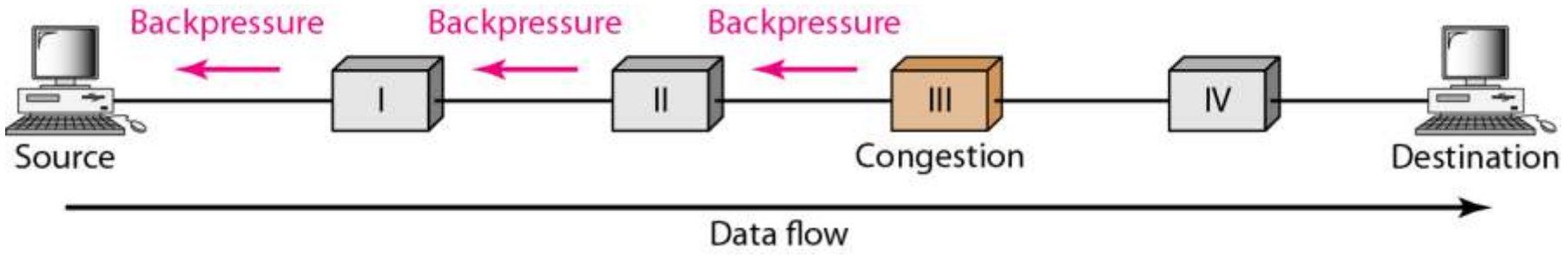
- Prevent congestion in virtual-circuit networks
- Before creating virtual circuit check the check the resource requirement
- A router can deny establishing a virtual circuit connection if there is congestion in the network or if there is a possibility of future congestion

2.2. Closed-Loop Congestion Control

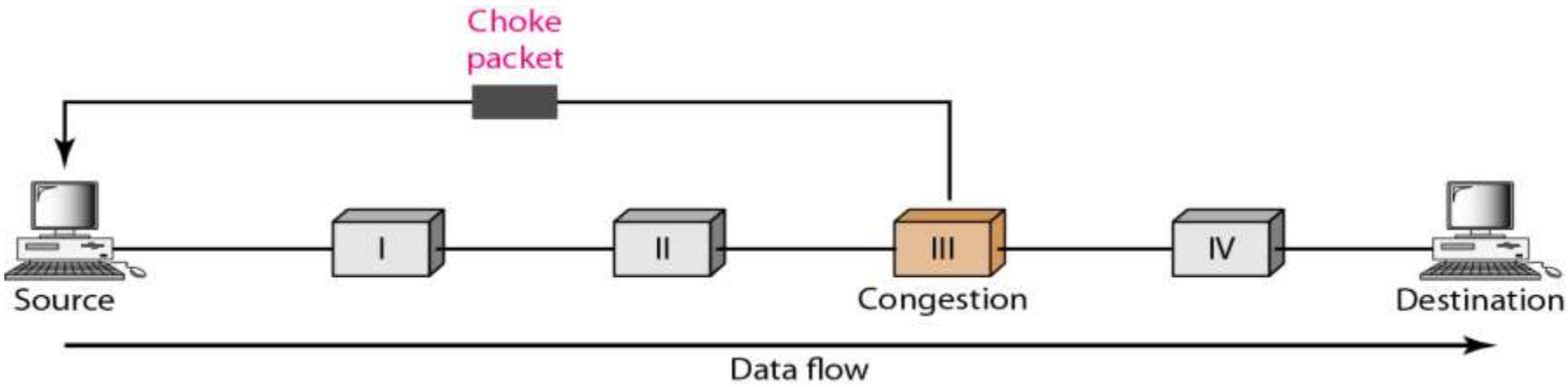
- Closed-loop congestion control mechanisms try to **alleviate congestion after it happens**
- Several mechanisms have been used by different protocols, They are
    1. Back pressure
    2. Choke packet
    3. Implicit signalling
    4. Explicit signalling
        1. Forward signalling
        2. Backward signalling

2.2.1. Back Pressure

- **Backpressure** refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes.
- Backpressure is a node-to-node congestion control
- Starts from congested node propagates, in the opposite direction of data flow, to the source
- Backpressure technique can be applied only to virtual circuit



2.2.2. Choke packet



- A choke packet is a packet sent by a node to the source to inform it of congestion
- Warning is from the congested encountered router to the source station directly
- Intermediate nodes doesn’t know about congestion.

2.2.3. Implicit Signalling

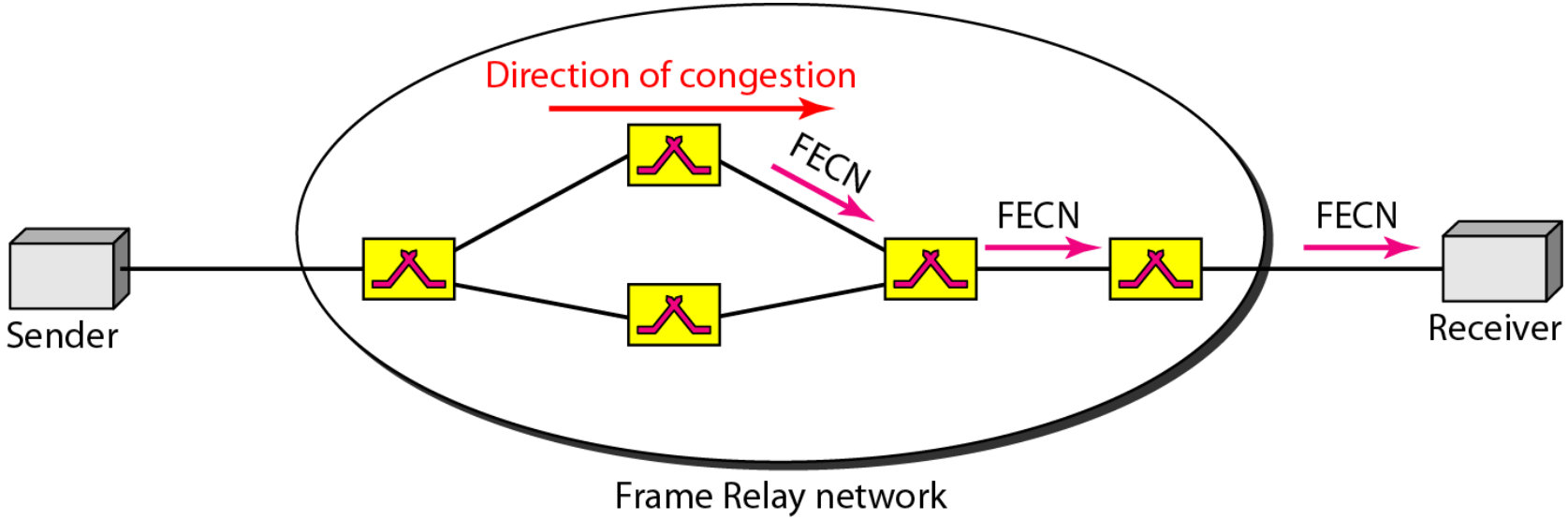
- No communication between the congested nodes and the source
- Source guesses that there is a congestion somewhere in the network from other symptoms
- Signals like acknowledgement is used
- If **ACK** is delayed the source assume there is congestion in destination and slows down the data transfer
- Used mainly in TCP network

2.2.4. Explicit Signalling

- The node that experiences congestion can explicitly send a signal to the source or destination
- Here only difference from choke packet is no separate packet is used where as in the choke packet separate packet is used
- It is used in Frame relay
- 2 types of signalling
  1. Forward signalling
  2. Backward signalling

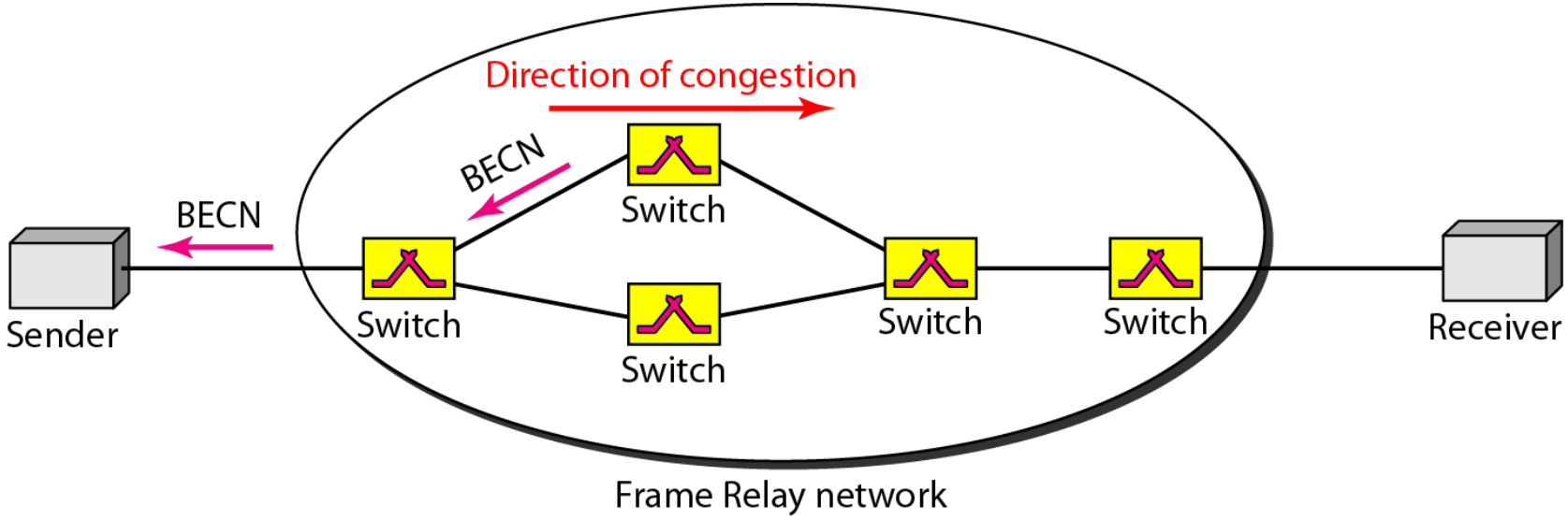
2.2.4.1. Forward Explicit Signalling

- 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 I congestion.



2.2.4.2. Backward Explicit Signalling

- 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



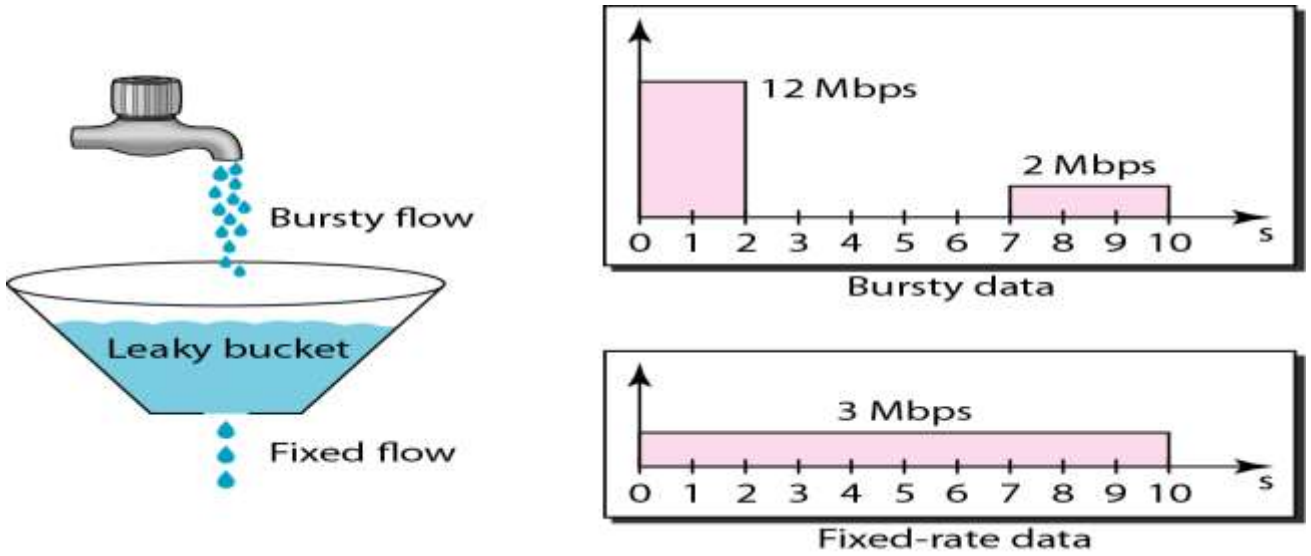
3. Traffic Shaping(QoS)

- Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network
- Two techniques can shape traffic
    1. Leaky bucket
    2. Token bucket

3.1. Leaky Bucket

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 rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty

NB: In Network Bucket is router and water is data packets

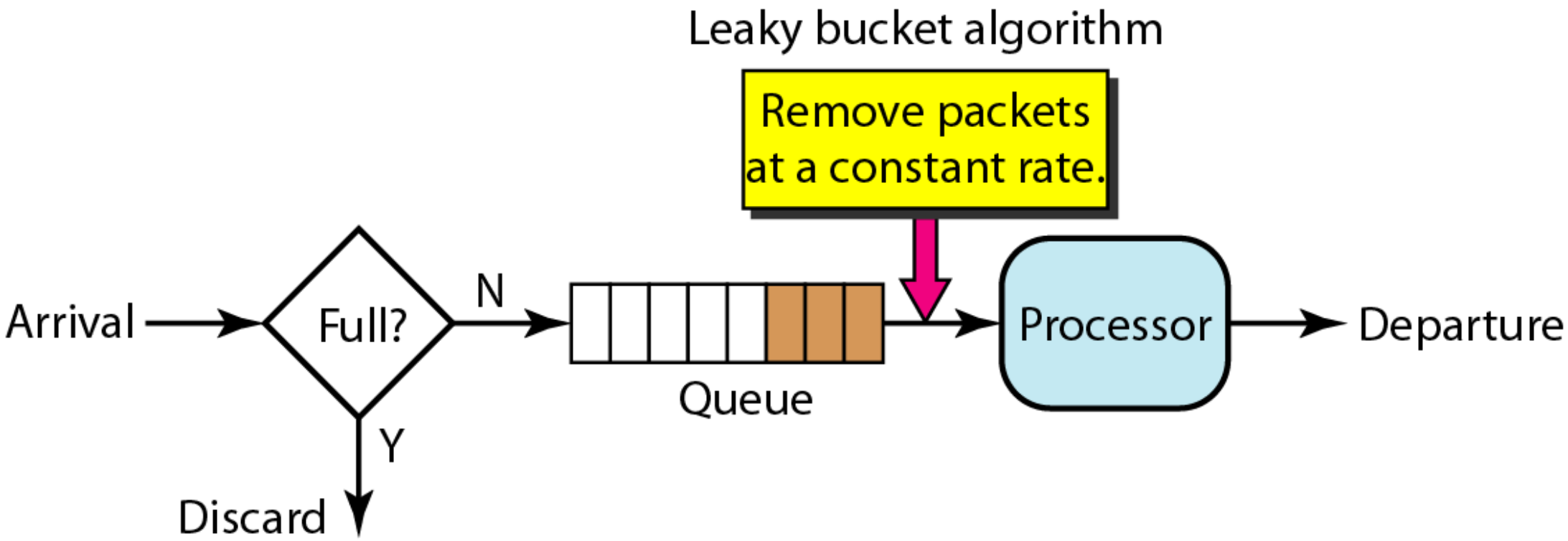


3.1. Leaky Bucket

NB: In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. The host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits of data. The host is silent for 5 s and then sends data at a rate of 2Mbps for 3 s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in 10 s

- The input rate can vary, but the output rate remains constant
- Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic, Bursty chunks are stored in router and sent out at an average rate
- It may also drop the packet if the bucket is full

3.1.1. Leaky bucket Implementation



3.1.1. Leaky bucket Implementation

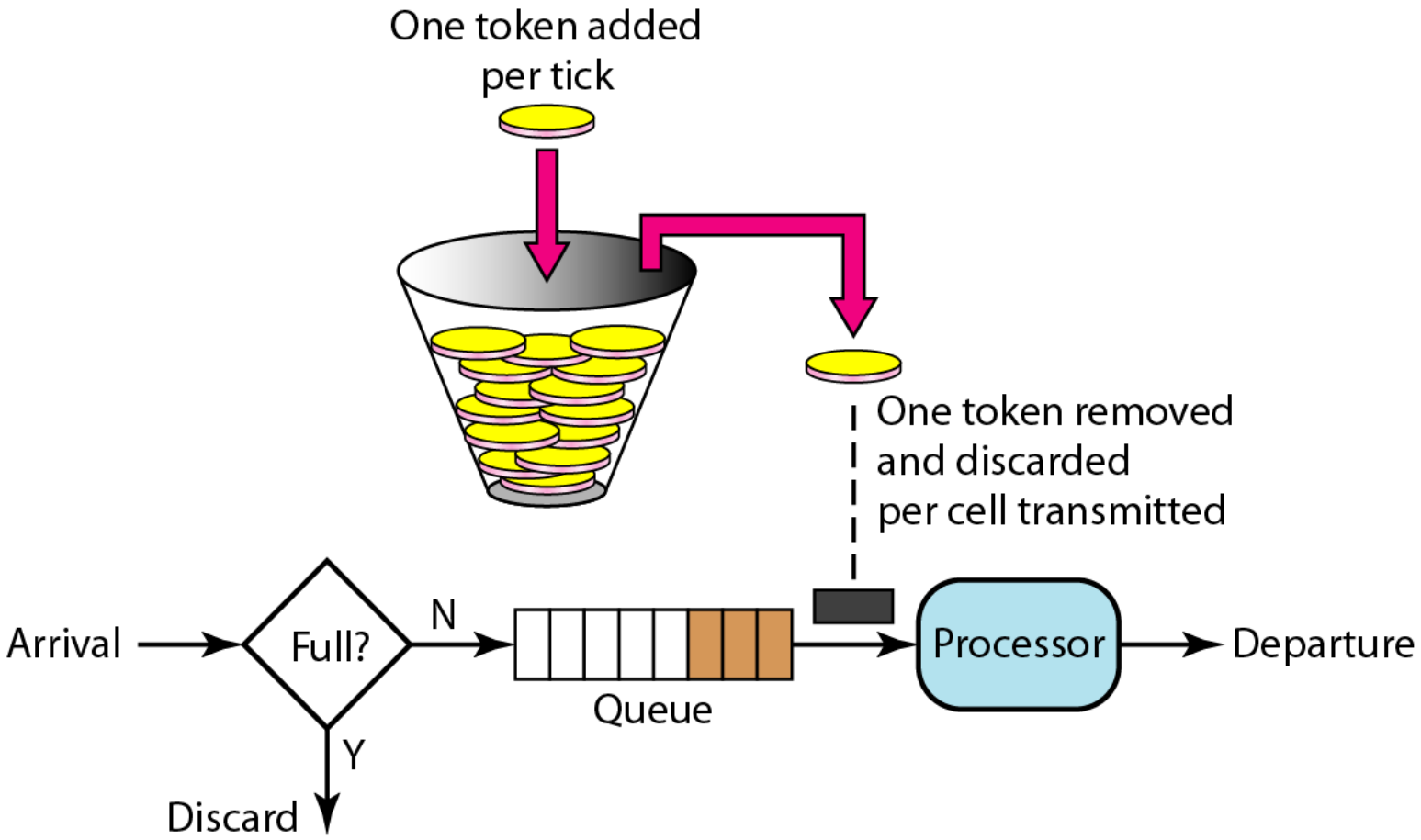
- Process removes a fixed number of packets from the queue at each tick of the clock
- If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.
- The following is an algorithm for variable-length packets:
  1. Initialize a counter to *n* at the tick of the clock.
  2. If *n* is greater than the size of the packet, send the packet and decrement the counter by the packet size. Repeat this step until *n* is smaller than the packet size.
  3. Reset the counter and go to step 1

3.1.2. Token Bucket

- The host can send bursty data as long as the bucket is not empty
- Token bucket algorithm allows idle hosts to accumulate credit for the future in the form of tokens
- For each tick of the clock, the system sends *n* tokens to the bucket
- The system removes one token for every cell (or byte) of data sent

For example, if n is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens. Now the host can consume all these tokens in one tick with 10,000 cells, or the host takes 1000 ticks with 10 cells per tick

3.1.2. Token Bucket



3.1.2. Token Bucket

- The token bucket can easily be implemented with a counter
  1. The token is initialized to zero
  2. Each time a token is added, the counter is incremented by 1
  3. Each time a unit of data is sent, the counter is decremented by 1
  4. When the counter is zero, the host cannot send data

4. Congestion Control in TCP

- TCP uses congestion control to avoid congestion or alleviate congestion in the network

1. Congestion Window
2. Congestion Policy
    1. Slow start
    2. Congestion Avoidance
    3. Congestion Detection

4.1. Congestion Window

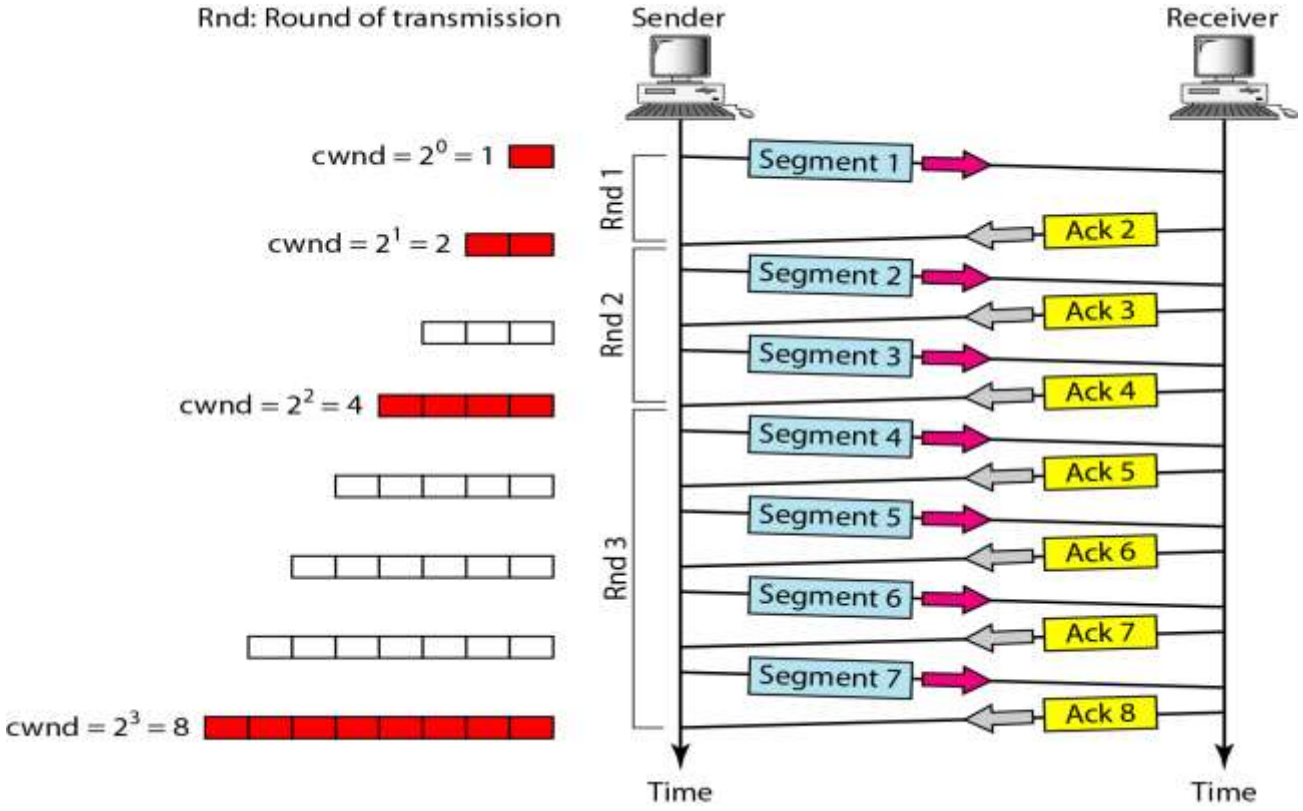
- Sender can send only that amount of data before waiting for an acknowledgment, the amount data is termed a **window size**
- The sender has two pieces of information:
  1. Receiver window size : Maximum available buffer size of receiver
  2. Congestion window size : Congestion window size is determined by congestion policy
- Window size >= minimum of receiver window size , congestion window size

4.2. Congestion policy

- Handling of congestion is done by 3 phases
  1. Slow start
  2. Congestion avoidance
  3. Congestion detection

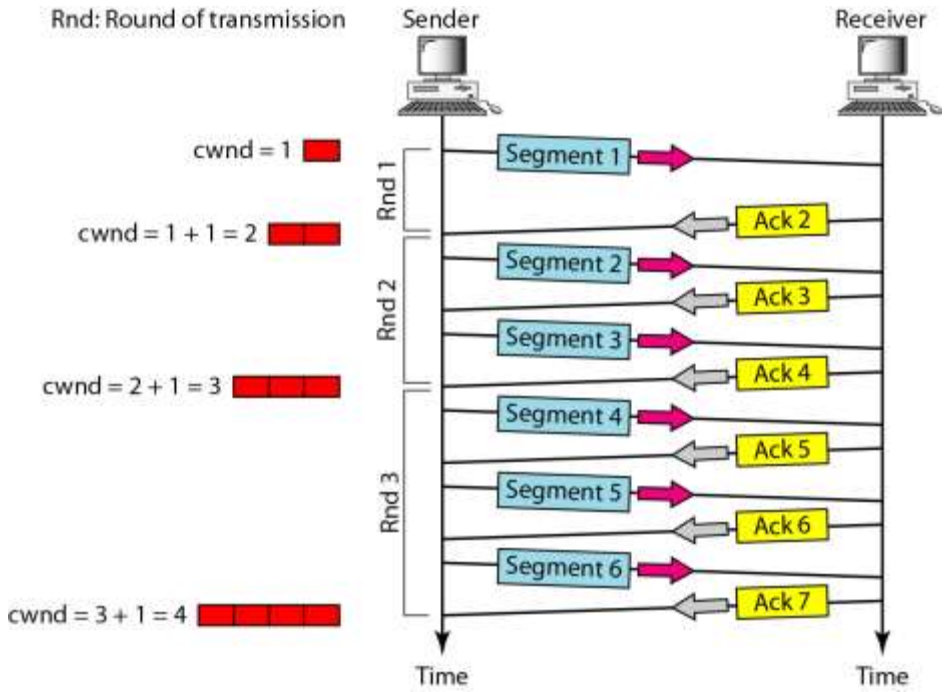
4.2.1. Slow Start

- Size of the congestion window increases exponentially until it reaches a threshold



4.2.2. Congestion Avoidance

- Congestion Avoidance uses Additive inverse
- Congestion window increases additively until congestion is detected
- Slow-start phase stops and the additive phase begins



4.2.3. 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:
  1. when a timer times out
  2. when three ACKs are received
- In both cases, the size of the threshold is dropped to one-half, a multiplicative decrease