Transport Layer

Compiled by: Hiranya Prasad Bastakoti

Contents

- Introduction Functions and Services
- Transport Protocols TCP, UDP and Their Comparisons
- Connection Oriented and Connectionless Services
- Congestion Control Open Loop & Closed Loop, TCP Congestion Control
- Traffic Shaping Algorithms
- Techniques to improve QOS Scheduling, traffic shaping, resource reservation, admission control
- Queuing Techniques for Scheduling
- Introduction to Ports and Sockets,
- Socket Programming

Introduction

- The transport layer is the fourth layer from the bottom in the OSI reference model.
- It is responsible for message delivery from process running in source computer to the process running in the destination computer.
- Transport layer does not perform any function in the intermediate nodes.
- It is active only in the end systems.
- *The transport layer has the critical role of providing communication services directly to the application processes running on different hosts in the network*

- Network Layer is responsible for delivery of datagrams between two hosts.
  - This is called ***host-to-host delivery***.
- Transport Layer is responsible for delivery of entire message from one process running on source to another process running on destination.
  - This is called ***process-to process delivery***

Functions

- The transport layer delivers the message from one process to another process running on two different hosts.
- Thus, it has to perform number of functions to ensure the accurate delivery of message.
- The various functions of transport layer are:
  - Establishing, Maintaining & Releasing Connection
  - Multiplexing and Demultiplexing
  - Addressing
  - Data Transfer
  - Flow Control
  - Error Control
  - Congestion Control

- **Establishing, Maintaining & Releasing Connection:**
  - The transport layer establishes, maintains & releases end-to-end transport connection on the request of upper layers.
  - Establishing a connection involves allocation of buffers for storing user data, synchronizing the sequence numbers of packets etc.
  - A connection is released at the request of upper layer.
- **Addressing:**
  - In order to deliver the message from one process to another, an addressing scheme is required.
  - Several process may be running on a system at a time.
  - In order to identify the correct process out of the various running processes, transport layer uses an addressing scheme called ***port number***.
  - Each process has a specific port number

Multiplexing and Demultiplexing

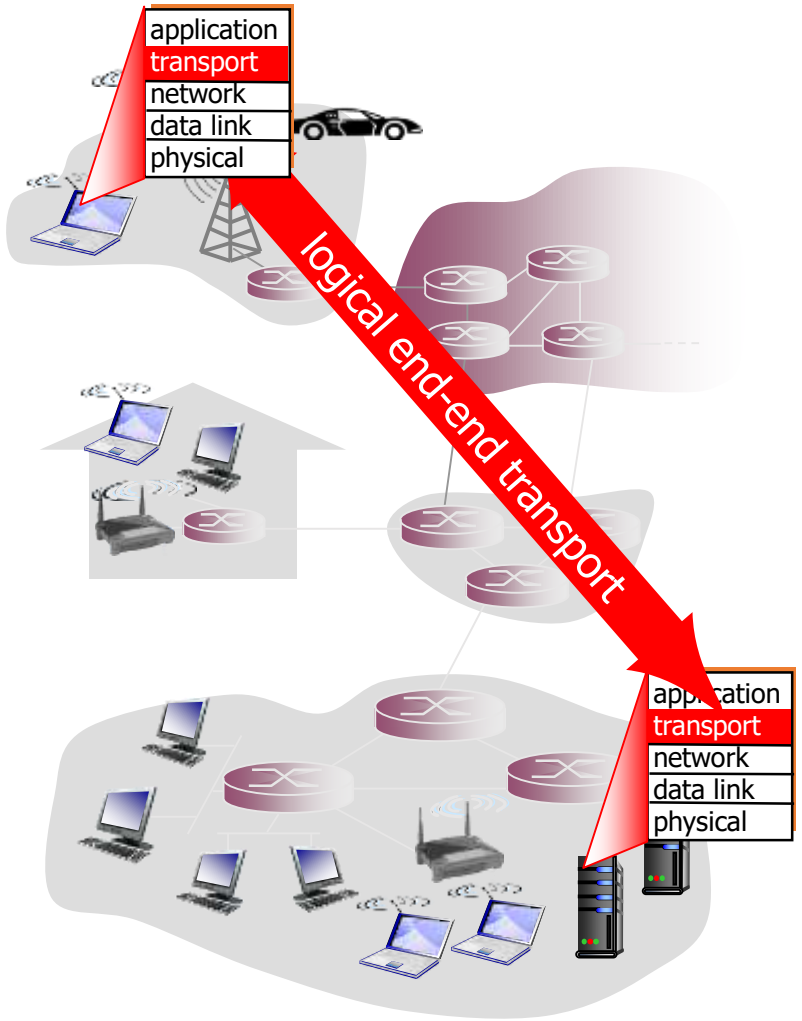
- Multiplexing allows simultaneous use of different applications over a network which are running on a host.
- Transport layer provides this mechanism which enables us to send packet streams from various applications simultaneously over a network.
- Transport layer accepts these packets from different processes differentiated by their port numbers and passes them to network layer after adding proper headers.
- Similarly Demultiplexing is required at the receiver side to obtain the data coming from various processes.
- Transport receives the segments of data from network layer and delivers it to the appropriate process running on the receiver's host.

- **Data Transfer:**
  - Transport layer breaks user data into smaller units and attaches a transport layer header to each unit forming a TPDU (TransPort Layer Data Unit).
  - The TPDU is handed over to the network layer for its
  - delivery to destination.
  - The TPDU header contains port number, sequence number, acknowledgement number, checksum and other fields.
- **Flow Control:**
  - Like data link layer, transport layer also performs flow control.
  - However, flow control at transport layer is performed end-to-end rather than node-to-node.
  - Transport Layer uses a sliding window protocol to perform flow control

- **Error Control:** Transport layer also provides end-to-end error control facility.
  - Transport layer deals with several different types of errors:
    - Error due to damaged bits.
    - Error due to non delivery of TPDUs.
    - Error due to duplicate delivery of TPDUs.
    - Error due to delivery of TPDU to a wrong destination.
- **Congestion Control:** Transport layer also handles congestion in the networks.
  - Congestion is a situation in which too many sources over a network attempt to send data and the router buffers start overflowing due to which loss of packets occur.
  - Several different congestion control algorithms are used to avoid congestion

Transport Layer services and protocols

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into *segments*, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



Transport layer Protocol Services

- Transport layer protocols can provide two types of services:
  - Connection Oriented Service
  - Connectionless Service

Transport Layer Services

- **Connection Oriented Service:**
  - In connection oriented service, a connection is first established between sender and the receiver.
  - Then, transfer of user data takes place.
  - At the end, connection is released.
  - The connection oriented service is generally reliable.
  - Transport layer protocols that provide connection oriented service are **TCP** and **SCTP** (Stream Control Transmission Protocol).

Transport Layer Services

- **Connectionless Service:**
  - In the service, the packets are sent from sender to receiver without the establishment of connection.
  - In such service, packets are not numbered.
  - The packets may be lost, corrupted, delayed or disordered.
  - Connectionless service is unreliable.
  - Transport layer protocol that provides this service is **UDP**.

Connection oriented and Connectionless Service

BASIS OF COMPARISON	CONNECTION-ORIENTED SERVICE	CONNECTION-LESS SERVICE
Prior Connection Requirement	Necessary	Not required
Reliability	Ensures reliable transfer of data.	Not guaranteed.
Congestion	Unlikely	Occur likely.
Transferring mode	It can be implemented using circuit switching and virtual circuit.	It is implemented using packet switching.

Lost data retransmission	Feasible	Practically, not possible.
Suitability	Suitable for long and steady communication.	Suitable for bursty Transmission.
Signalling	Used for connection establishment.	There is no concept of signalling.
Packet forwarding	Packets sequentially travel to their destination node and follows the same route.	Packets reach the destination randomly without following the same route.
Delay	There is a delay in transfer of information, but once the connection is established faster delivery can be achieved.	Because to the absence of connection establishment phase, the transmission is faster.
Resource Allocation	Need to be allocated.	No prior allocation of the resource is required.

Elements of Transport Protocols

- **Addressing:**
  - In order to deliver data from one process to another, address is required.
  - In order to deliver data from one node to another, MAC address is required.
  - Such an address is implemented at Data Link Layer and is called **Physical Addressing**.
  - In order to deliver data from one network to another, IP address is required.
  - Such an address is implemented at Network Layer and is called **Logical Addressin**
  - Similarly, in order to deliver data from a process running on source to process runnir on destination, transport layer defines the **Service Point Address** or **Port Number**

Elements of Transport Protocols

- **Port Numbers:**
  - Each communicating process is assigned a specific port number.
  - In order to select among multiple processes running on a destination host, a port number is required.
  - The port numbers are 16-bit integers between 0 and 65,535.

Elements of Transport Protocols

- **Socket Address:**
  - Socket address is a combination of IP address and port number.
  - In order to provide communication between two different processes on different networks, both IP address and port number, i.e. socket address is required.

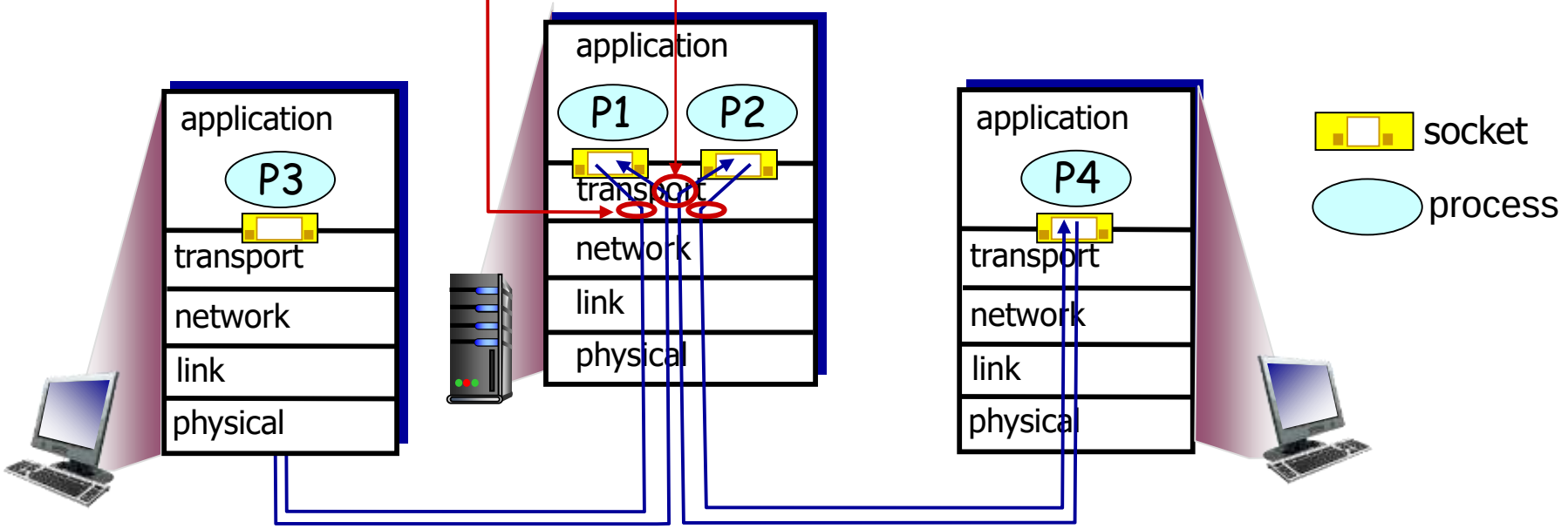
Elements of Transport Protocols

- **Multiplexing & Demultiplexing:**
  - A network connection can be shared by various applications running on a system.
  - There may be several running processes that want to send data and only one transport layer connection available, then transport layer protocols may perform multiplexing.
  - The protocol accepts the messages from different processes having their respective port numbers, and add headers to them.

Multiplexing/demultiplexing

multiplexing at sender:
handle data from multiple sockets, add transport header (later used for demultiplexing)

demultiplexing at receiver:
use header info to deliver received segments to correct socket

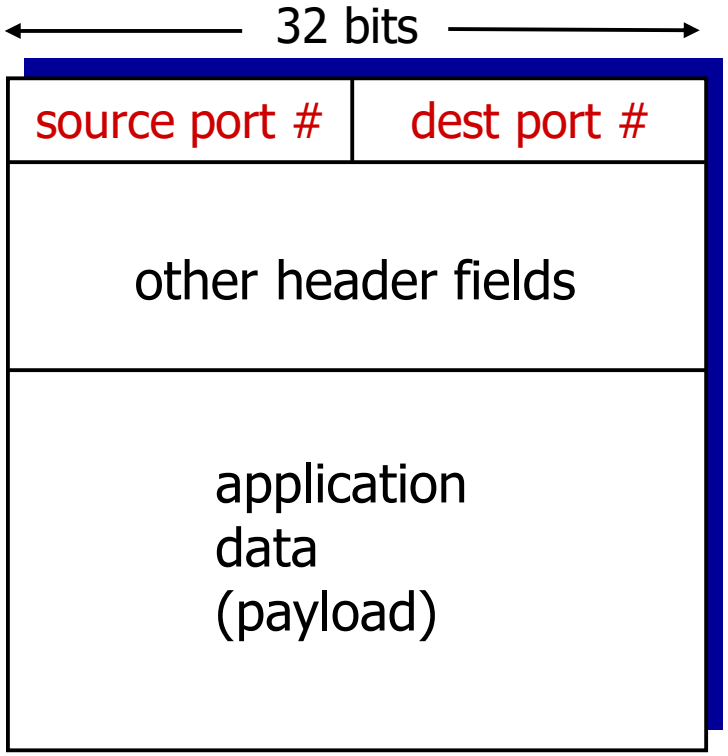


Elements of Transport Protocols

- **Multiplexing & Demultiplexing (Cont.):**
  - The transport layer at the receiver end performs demultiplexing to separate the messages for different processes.
  - After checking for errors, the headers of messages are dropped and each message is handed over to the respective processes based on their port numbers.

How demultiplexing works

- host receives IP datagrams
  - each datagram has source IP address, destination IP address
  - each datagram carries one transport-layer segment
  - each segment has source, destination port number
- host uses *IP addresses & port numbers* to direct segment to appropriate socket



TCP/UDP segment format

Elements of Transport Protocols

- **Connection Establishment:**
  - Before communicating, the source device must first determine the availability of the other to exchange data.
  - Path must be found through the network by which the data can be sent.
  - This is called Connection Establishment.
  - Connection establishment involves **Three-Way Handshaking** mechanism:
    - The source sends a **connection request** packet to the destination.
    - The destination returns a confirmation packet back to the source.
    - The source returns a packet acknowledging the confirmation.

- **Connection Release:**
  - Once all of the data has been transferred, the connection must be released.
  - It also requires a **Three-Way Handshaking** mechanism:
    - The source sends a **disconnect request** packet to the destination.
    - The destination returns a confirmation packet back to the source.
    - The source returns a packet acknowledging the confirmation.

Transport Layer Protocols

- Transport layer provides two types of services:
  - Connection Oriented Service
  - Connectionless Service
- For this, transport layer defines two different protocols:
  - Transmission Control Protocol (TCP)
  - User Datagram Protocol (UDP)

Transmission Control Protocol

- Transmission Control Protocol (TCP) is a connection oriented protocol that provides reliable services between processes on different hosts.
- It uses the services of lower layer which provide connectionless and unreliable service.

Transmission Control Protocol

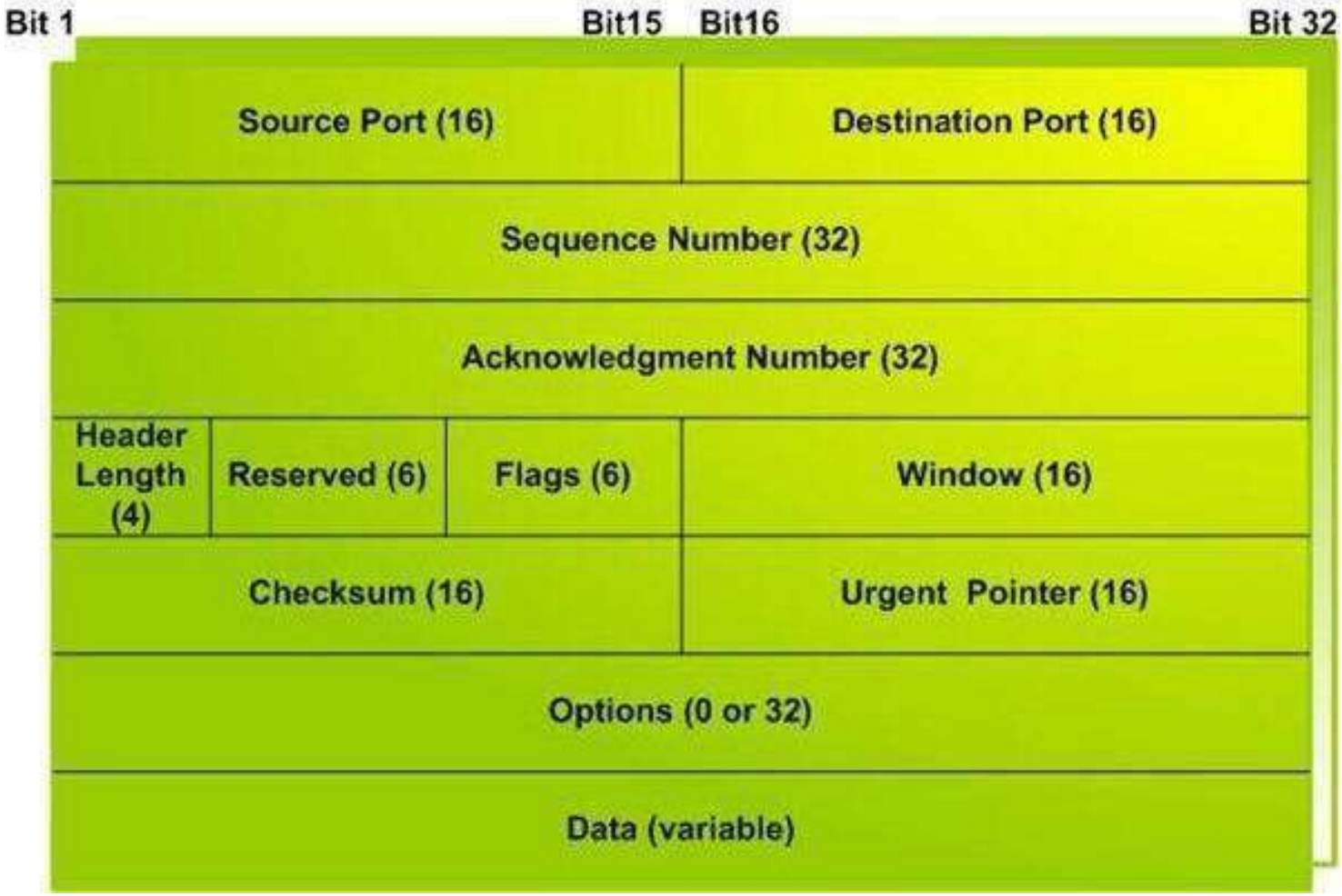
- The basic features of TCP are:

- It provides efficient method for numbering different bytes of data.
  - It provides stream data transfer.
- It offers reliability.
- It provides efficient flow control.
- It provides full duplex operation.
- It provides multiplexing.
- It provides connection oriented service.

TCP Segment

- TCP segment is the unit of data transferred between two processes.
- Each TCP segment consists of two parts:
  - Header Part
  - Data Part

Format of TCP Segment



Format of TCP Segment

- **Source Port:**
  - It indicates the port number of a source process. It is of 2 bytes.
- **Destination Port:**
  - It indicates the port number of destination process. It is also 2 bytes.
- **Sequence Number:**
  - It specifies the number assigned to the current message. It is of 4 bytes.

Format of TCP Segment

- **Acknowledgement Number:**
  - It indicates the sequence number of the next byte of data. It is of 4 bytes.
- **Header Length:**
  - It indicates number of words in the TCP header. It is a 4 bit field.
- **Reserved:**
  - This 6 bit field is reserved for future use.

Format of TCP Segment

- **Flags:**
  - This 6 bit field consists of 6 different flags:
    - UGR (Urgent Pointer)
    - ACK (Acknowledgement)
    - PSH (Request for Push)
    - RST (Reset the Connection)
    - SYN (Synchronize)
    - FIN (Final or Terminate the Connection)

Format of TCP Segment

- **Window:**

  It specifies the size of sender's receiving window, i.e., the buffer space available for incoming data. It is of 2 bytes.

- **Checksum:** This 16-bit field contains the checksum.
- **Urgent Pointer:** This 16-bit field is valid only if urgent pointer in flags is set to 1.
- **Options:** It contains the optional information in the TCP header. It is of 32 bytes.
- **Data:** This field contains the upper layer information. It is of variable size

User Datagram Protocol

- User Datagram Protocol (UDP) is a connectionless, unreliable transport protocol.
- Like TCP, UDP also provides process-to-process communication.
- Unlike TCP, it does not provide flow control and error control mechanisms.
- It is connectionless, therefore, it transfers data without establishing a connection.

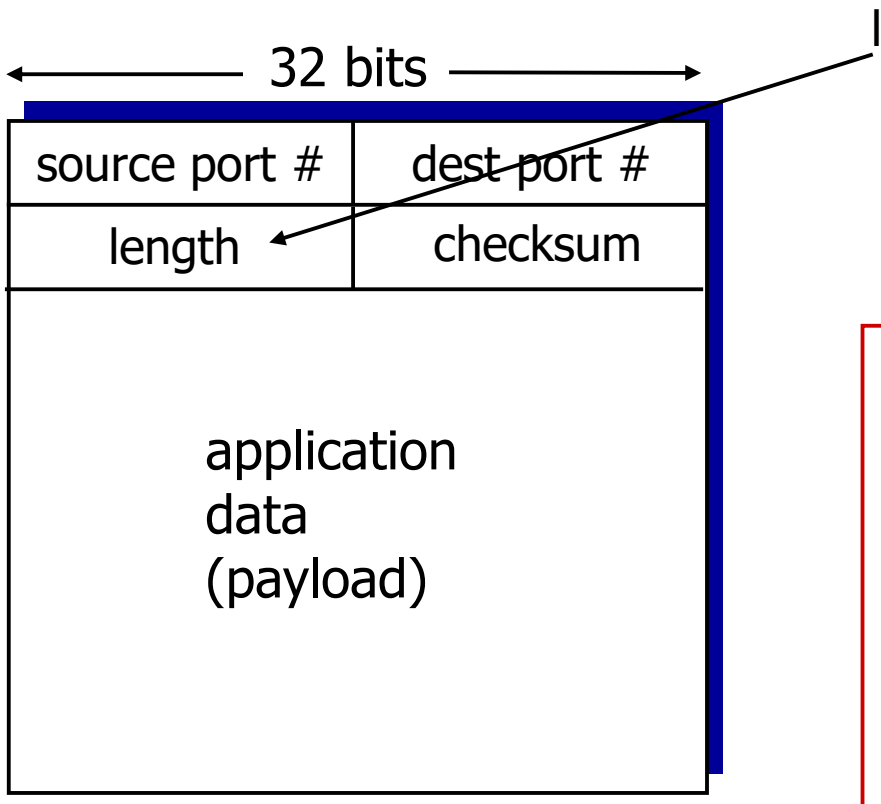
User Datagram Protocol

- The various features of UDP are:
  - It provides connectionless transport service.
  - It is unreliable.
  - It does not provide flow control and error control.
  - It is less complex and is simple than TCP, and easy to implement.
  - User datagrams (packets) are not numbered.

UDP Datagram

- A datagram is the unit of data transferred between two processes.
- Each UDP datagram consists of two parts:
  - Header Part
  - Data Part.

UDP: segment header



UDP segment format

length, in bytes of UDP segment, including header

why is there a UDP?

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small header size
- no congestion control: UDP can blast away as fast as desired

UDP Datagram

- **Source Port:**
  - It indicates the port number of source process. It is of 16 bits.
- **Destination Port:**
  - This 16 bit field specifies the port number of destination process.
- **Length:**
  - It specifies the total length of the user datagram (header + data). It is of 16 bits.
- **Checksum:**
  - The contains the checksum, and is optional. It is also of 16 bits.

TRANSMISSION CONTROL PROTOCOL (TCP)	USER DATAGRAM PROTOCOL (UDP)
TCP is a connection-oriented protocol. Connection-orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data.	UDP is the Datagram oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, and terminating a connection. UDP is efficient for broadcast and multicast type of network transmission.
TCP is reliable as it guarantees delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.
TCP provides extensive error checking mechanisms. It is because it provides flow control and acknowledgment of data.	UDP has only the basic error checking mechanism using checksums.
Sequencing of data is a feature of Transmission Control Protocol (TCP). this means that packets arrive in-order at the receiver.	There is no sequencing of data in UDP. If ordering is required, it has to be managed by the application layer.

TCP is comparatively slower than UDP.	UDP is faster, simpler and more efficient than TCP.
Retransmission of lost packets is possible in TCP, but not in UDP.	There is no retransmission of lost packets in User Datagram Protocol (UDP).
TCP header size is 20 bytes.	UDP Header size is 8 bytes.
TCP is heavy-weight.	UDP is lightweight.
TCP is used by HTTP, HTTPs, FTP, SMTP and Telnet	UDP is used by DNS, DHCP, TFTP, SNMP, RIP, and VoIP

Congestion and Congestion Control

- Congestion can be defined as the state of the network where total demand for resources is more than availability of the resource.
- In other words the congestion can defined as the situation where total number of packets are more than it can accumulate thus resulting in deterioration of performance.
- Usually congestion occurs when availability of resources is insufficient to meet the demand of the network.

Causes of Congestion

- Insufficient memory to store arriving packets.
- Packet arrival rate exceeds the outgoing link capacity
- Slow processor.
- Bursty traffic

Congestion Control:

Congestion control refers to the techniques used to control or prevent congestion.

Congestion control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity

- Congestion control techniques can be broadly classified into two categories:

➢ open-loop congestion control

➢ closed-loop congestion control

Open Loop Congestion Control

- Open loop congestion control policies are applied to prevent congestion before it happens. The congestion control is handled either by the source or the destination.
- **Policies adopted by open loop congestion control are:**
- **Retransmission Policy**
- **Window Policy**
- **Discarding Policy**
- **Acknowledgment Policy**
- **Admission Policy**

- **Retransmission Policy :**
  It is the policy in which retransmission of the packets are taken care. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. This transmission may increase the congestion in the network.
  To prevent congestion, retransmission timers must be designed to prevent congestion and also able to optimize efficiency.
- **Window Policy :**
  The type of window at the sender side may also affect the congestion. Several packets in the Go-back-n window are resent, although some packets may be received successfully at the receiver side. This duplication may increase the congestion in the network and making it worse.
  Therefore, Selective repeat window should be adopted as it sends the specific packet that may have been lost.

- **Discarding Policy :**
  A good discarding policy adopted by the routers is that the routers may prevent congestion and at the same time partially discards the corrupted or less sensitive package and also able to maintain the quality of a message.
  In case of audio file transmission, routers can discard less sensitive packets to prevent congestion and also maintain the quality of the audio file.
- **Acknowledgment Policy :**
  Since acknowledgement are also the part of the load in network, the acknowledgment policy imposed by the receiver may also affect congestion. Several approaches can be used to prevent congestion related to acknowledgment.
  The receiver should send acknowledgement for N packets rather than sending acknowledgement for a single packet. The receiver should send a acknowledgment only if it has to sent a packet or a timer expires.

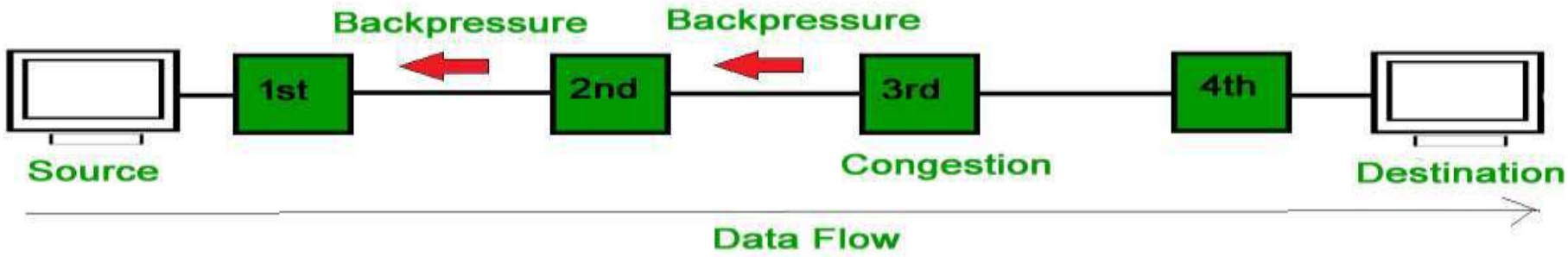
- **Admission Policy :**
  In admission policy a mechanism should be used to prevent congestion. Switches in a flow should first check the resource requirement of a network flow before transmitting it further. If there is a chance of a congestion or there is a congestion in the network, router should deny establishing a virtual network connection to prevent further congestion

Closed Loop Congestion Control

- Closed loop congestion control technique is used to treat or alleviate congestion after it happens. Several techniques are used by different protocols; some of them are:
- **Backpressure**
- **Choke Packet Technique**
- **Implicit Signaling**
- **Explicit Signaling**

Backpressure

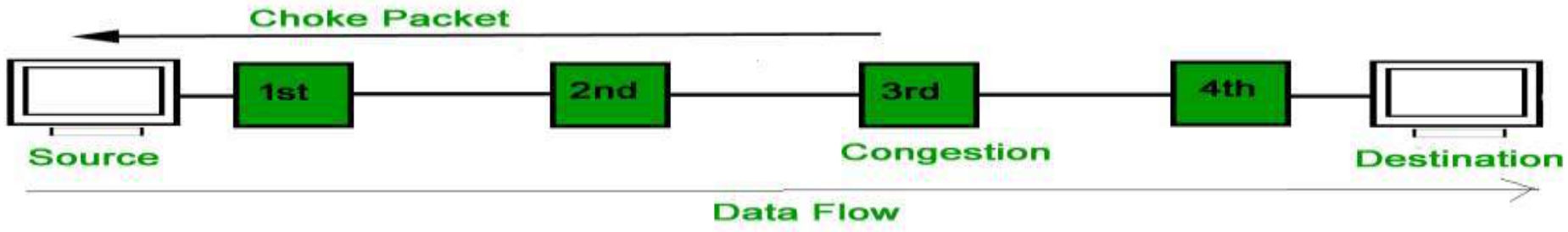
- Backpressure is a technique in which a congested node stop receiving packet from upstream node.
- This may cause the upstream node or nodes to become congested and rejects receiving data from above nodes.
- Backpressure is a node-to-node congestion control technique that propagate in the opposite direction of data flow.
- The backpressure technique can be applied only to virtual circuit where each node has information of its above upstream node



- In above diagram the 3rd node is congested and stops receiving packets as a result 2nd node may be get congested due to slowing down of the output data flow. Similarly 1st node may get congested and informs the source to slow down.

Choke Packet

- Choke packet technique is applicable to both virtual networks as well as datagram subnets.
- A choke packet is a packet sent by a node to the source to inform it of congestion. Each router monitor its resources and the utilization at each of its output lines. whenever the resource utilization exceeds the threshold value which is set by the administrator, the router directly sends a choke packet to the source giving it a feedback to reduce the traffic. The intermediate nodes through which the packets has traveled are not warned about congestion.



- **Implicit Signaling :**
  In implicit signaling, there is no communication between the congested nodes and the source. The source guesses that there is congestion in a network. For example when sender sends several packets and there is no acknowledgment for a while, one assumption is that there is a congestion.
- **Explicit Signaling :**
  In explicit signaling, if a node experiences congestion it can explicitly sends a packet to the source or destination to inform about congestion. The difference between choke packet and explicit signaling is that the signal is included in the packets that carry data rather than creating different packet as in case of choke packet technique.

- Explicit signaling can occur in either forward or backward direction.
- **Forward Signaling :** In forward signaling signal is sent in the direction of the congestion. The destination is warned about congestion. The reciever in this case adopt policies to prevent further congestion.
- **Backward Signaling :** In backward signaling signal is sent in the opposite direction of the congestion. The source is warned about congestion and it needs to slow down.

TCP Congestion Control

- Network congestion may occur when a sender overflows the network with too many packets.
- At the time of congestion, the network cannot handle this traffic properly, which results in a degraded quality of service (QoS).
- The typical symptoms of a congestion are: excessive packet delay, packet loss and retransmission.
- Insufficient link bandwidth, legacy network devices, greedy network applications or poorly designed or configured network infrastructure are among the common causes of congestion.

- The function of TCP (Transmission Control Protocol) is to control the transfer of data so that it is reliable.
- The main TCP features are connection management, reliability, flow control and congestion control.
- Connection management:
- It includes connection initialization (a 3-way handshake) and its termination.
- The source and destination TCP ports are used for creating multiple virtual connections.
- A reliable P2P transfer between hosts is achieved with the sequence numbers (used for segments reordering) and retransmission.
- A retransmission of the TCP segments occurs after a timeout, when the acknowledgement (ACK) is not received by the sender or when there are three duplicate ACKs received (it is called fast retransmission when a sender is not waiting until the timeout expires)

- Flow control
- It ensures that a sender does not overflow a receiving host. The receiver informs a sender on how much data it can send without receiving ACK from the receiver inside of the receiver's ACK message. This option is called the sliding window and it's amount is defined in Bytes

Congestion control

It ensures that the sender does not overflow the network.

Comparing to the flow control technique where the flow control mechanism ensures that the source host does not overflow the destination host, congestion control is more global. It ensures that the capability of the routers along the path does not become overflowed.

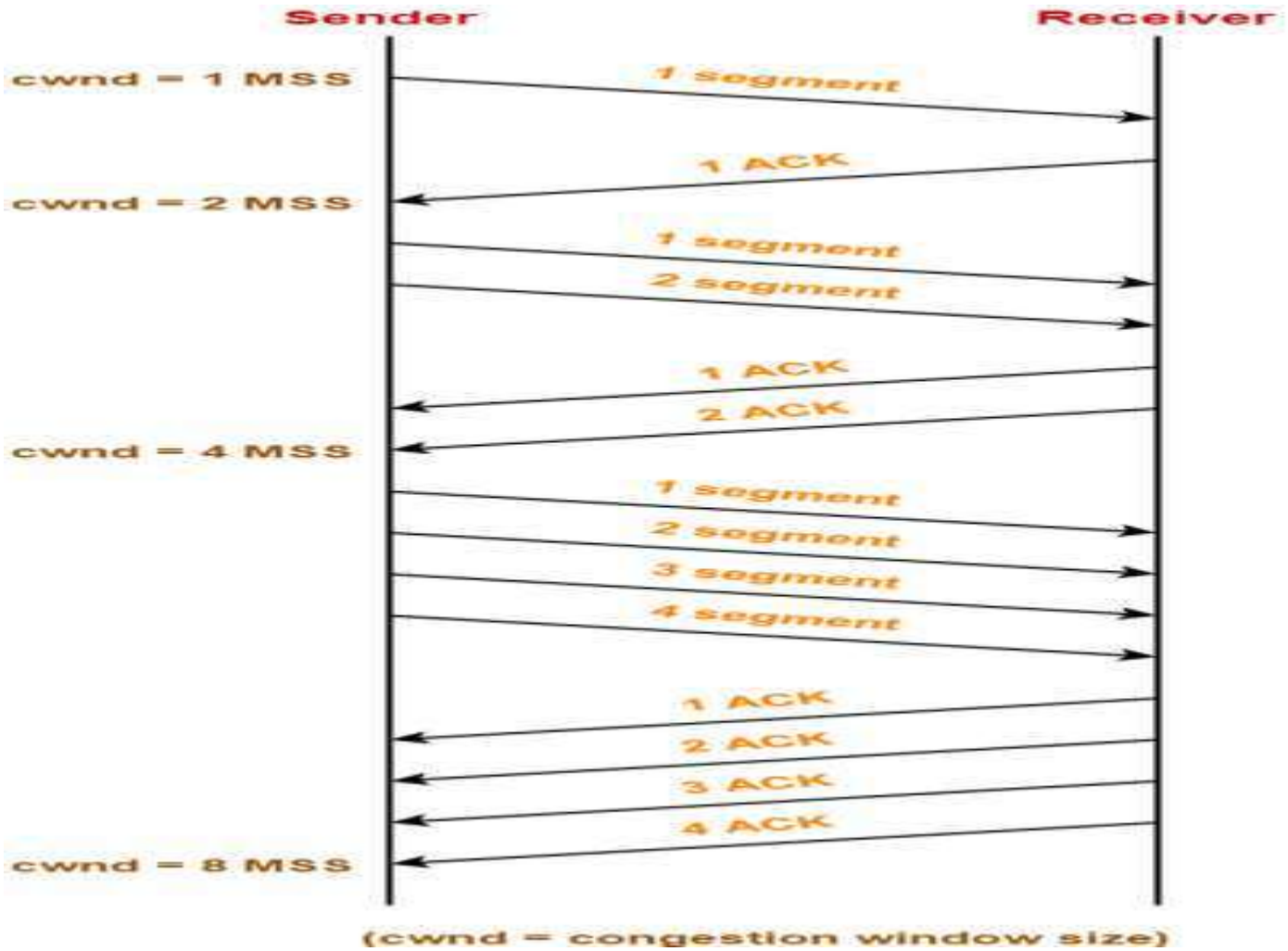
TCP Congestion Control

- TCP uses a congestion window and a congestion policy that avoid congestion.
- Previously, we assumed that only receiver can dictate the sender's window size. We ignored another entity here, the network.
- If the network cannot deliver the data as fast as it is created by the sender, it must tell the sender to slow down.
- In other words, in addition to the receiver, the network is a second entity that determines the size of the sender's window.

Congestion policy in TCP

- Slow Start Phase: starts slowly increment is exponential to threshold
- Congestion Avoidance Phase: After reaching the threshold increment is by 1
- Congestion Detection Phase: Sender goes back to Slow start phase or Congestion avoidance phase.
- **Slow Start Phase : exponential increment –** In this phase after every RTT(Round Trip Time) the congestion window size(cwnd) increments exponentially.
- Initially cwnd = 1
- After 1 RTT, cwnd = 2^(1) = 2
- 2 RTT, cwnd = 2^(2) = 4
- 3 RTT, cwnd = 2^(3) = 8

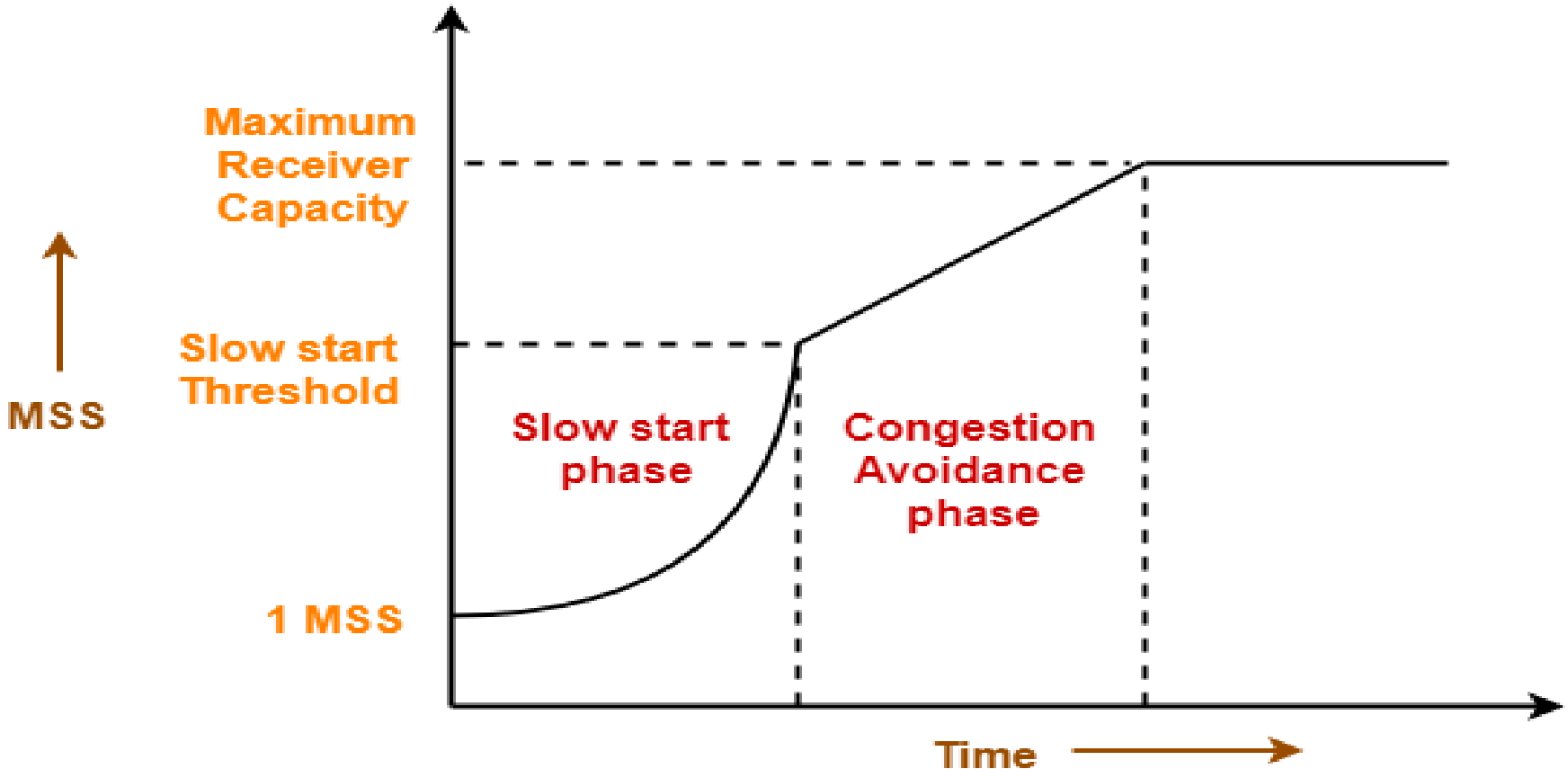
Slow Start Phase : exponential increment



MSS-Maximum Window Size

- **Congestion Avoidance Phase : additive increment** – This phase starts after the threshold value also denoted as *ssthresh*.
- The size of *cwnd*(congestion window) increases additive.
- After each RTT cwnd = cwnd + 1.
- Initially cwnd = i
- After 1 RTT, cwnd = i+1
- 2 RTT, cwnd = i+2
- 3 RTT, cwnd = i+3

Congestion Avoidance Phase



- **Congestion Detection Phase : multiplicative decrement –**
- If congestion occurs, the congestion window size is decreased. The only way a sender can guess that congestion has occurred is the need to retransmit a segment.
- Retransmission is needed to recover a missing packet which is assumed to have been dropped by a router due to congestion.
- Retransmission can occur in one of two cases: when the RTO timer times out or when three duplicate ACKs are received.

- **Case 1 : Retransmission due to Timeout –** In this case congestion possibility is high.(a) ssthresh is reduced to half of the current window size.
  (b) set cwnd = 1
  (c) start with slow start phase again.
- **Case 2 : Retransmission due to 3 Acknowledgement Duplicates –** In this case congestion possibility is less.(a) ssthresh value reduces to half of the current window size.
  (b) set cwnd= ssthresh
  (c) start with congestion avoidance phase

Congestion control/traffic shaping algorithms

- Traffic shaping (also referred to as packet shaping) is the technique of delaying and restricting certain packets traveling through a network to increase the performance of packets that have been given priority.
- Classes are defined to separate the packets into groupings so that they can each be shaped separately allowing some classes to pass through a network more freely than others. Traffic shapers are usually placed at the boundaries of a network to either shape the traffic going entering or leaving a network.
- Traffic shaping is a mechanism to control the amount and rate of the traffic sent to the network. The two traffic shaping techniques are:
- Leaky bucket algorithm
- Token bucket algorithm

Leaky Bucket Algorithm

- It is a traffic shaping mechanism that controls the amount and the rate of the traffic sent to the network.
- A leaky bucket algorithm shapes bursty traffic into fixed rate traffic by averaging the data rate.
- Imagine a bucket with a small hole at the bottom.
- The rate at which the water is poured into the bucket is not fixed and can vary but it leaks from the bucket at a constant rate. Thus (as long as water is present in bucket), the rate at which the water leaks does not depend on the rate at which the water is input to the bucket

- Same technique is applied to control congestion in network traffic. Every host in the network is having a buffer with finite queue length
- Packets which are put in the buffer is full are thrown away. The buffer may drain onto the subnet either by some number of packets per unit time,or by some total number of bytes per unit time.
- A FIFO queue is used for holding the packets.
- If the arriving packets are of fixed size, then the process removes a fixed number of packets from the queue at each tick of the clock.
- If the arriving packets are of different size,then the fixed output rate will not be based on the number of departing packets.
- Instead it will be based on the number of departing bytes or bit

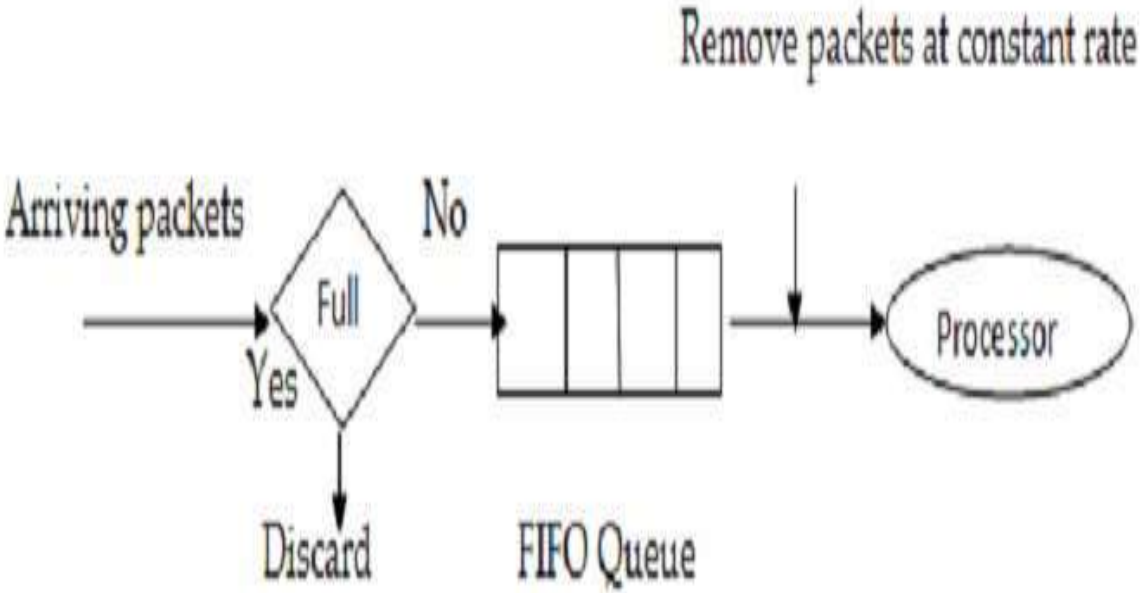
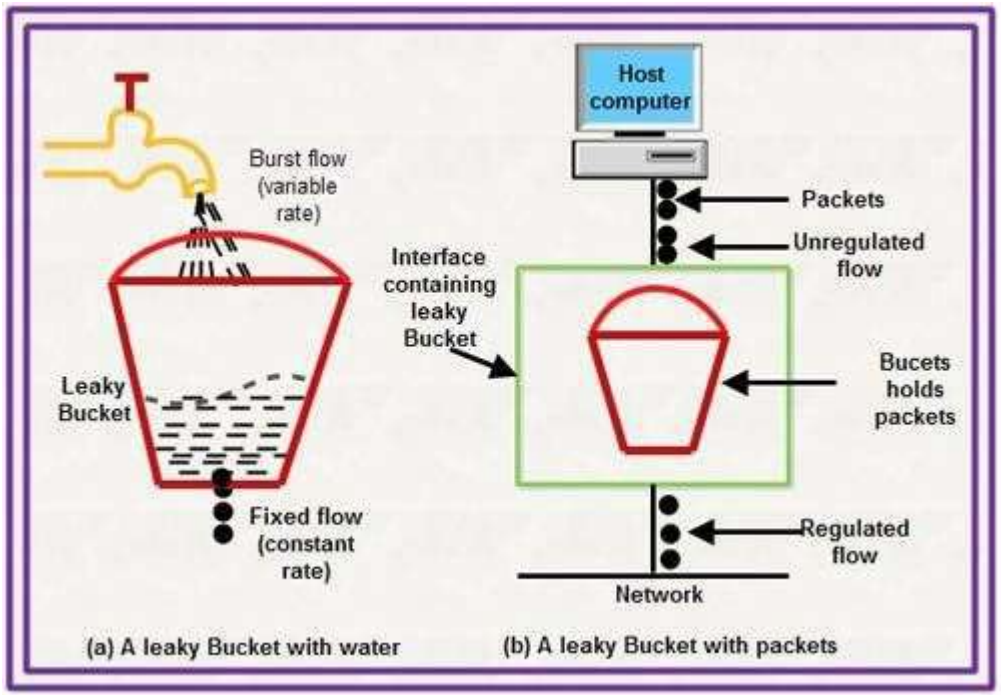


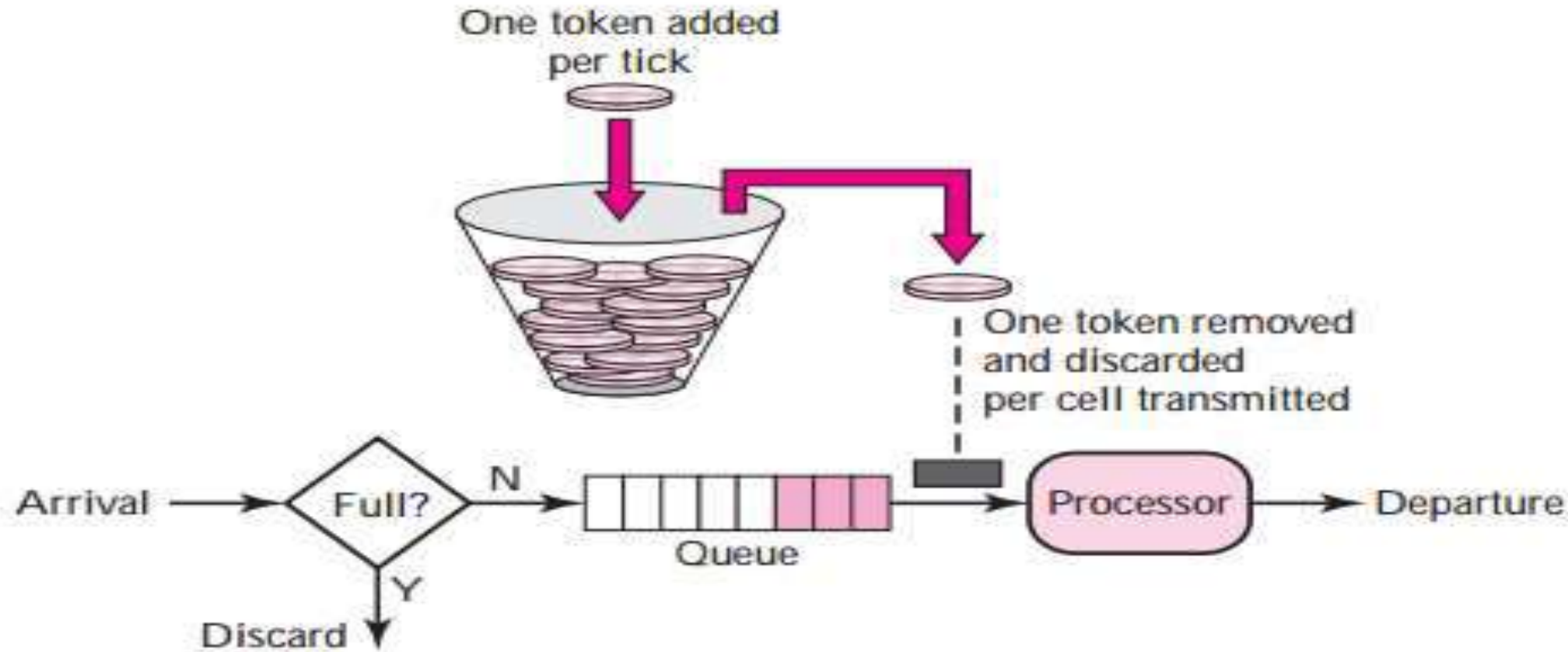
Figure 4.5 The implementation of leaky bucket



Token bucket Algorithm

- The leaky bucket algorithm allows only an average (constant) rate of data flow. Its major problem is that it cannot deal with bursty data.
- A leaky bucket algorithm does not consider the idle time of the host. For example, if the host was idle for 10 seconds and now it is willing to sent data at a very high speed for another 10 seconds, the total data transmission will be divided into 20 seconds and average data rate will be maintained. The host is having no advantage of sitting idle for 10 seconds.
- To overcome this problem, a token bucket algorithm is used. A token bucket algorithm allows bursty data transfers.
- A token bucket algorithm is a modification of leaky bucket in which leaky bucket contains tokens.
- In this algorithm, a token(s) are generated at every clock tick. For a packet to be transmitted, system must remove token(s) from the bucket.

- Thus, a token bucket algorithm allows idle hosts to accumulate credit for the future in form of tokens.
- For example, if a system generates 100 tokens in one clock tick and the host is idle for 100 ticks. The bucket will contain 10,000 tokens.
- Now, if the host wants to send bursty data, it can consume all 10,000 tokens at once for sending 10,000 cells or bytes.
- Thus a host can send bursty data as long as bucket is not empty.


One token added
per tick
One token removed
and discarded
per cell transmitted
Arrival
Full?
N
Y
Discard
Queue
Processor
Departure

- Host is connected to the network by an interface. This interface is actually a bucket. A token is generated in the bucket every ΔT seconds.
- The host sends an unregulated flow to the bucket. For a packet to be transmitted to the network, it must capture and destroy a token present in the bucket.
- If the lost is not sending packets to the bucket the tokens keep getting accumulated in the bucket. Generally there is a maximum amount of tokens that can be accumulated in the bucket.
- Due to this feature of tokens getting accumulated, bursts can be handled better. Therefore in this case the rate increases if tokens are saved in the bucket, whereas in leaky bucket the rate will always be constant (1 packet per clock tic).

Comparision

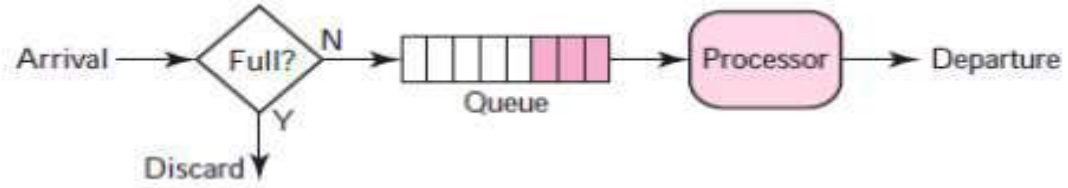
Token Bucket	Leaky Bucket
Token dependent	Token independent
If bucket is full token are discarded, but not the packet.	If bucket is full packet or data is discarded
Packets can only transmitted when there are enough token	Packets are transmitted continuously
It allows large bursts to be sent faster rate after that constant rate.	It sends the packet at constant rate.
It saves token to send large bursts	It does not save token.
Token Bucket does not have discard or priority policy	Leaky Bucket has priority policy

Techniques to improve QoS Scheduling

- Quality of service (QoS) is an internetworking issue that has been discussed more than defined. We can informally define quality of service as something a flow seeks to attain.
- **Techniques to Improve QoS:**
- Scheduling
- Traffic Shaping
- Resource Reservation
- Admission Control

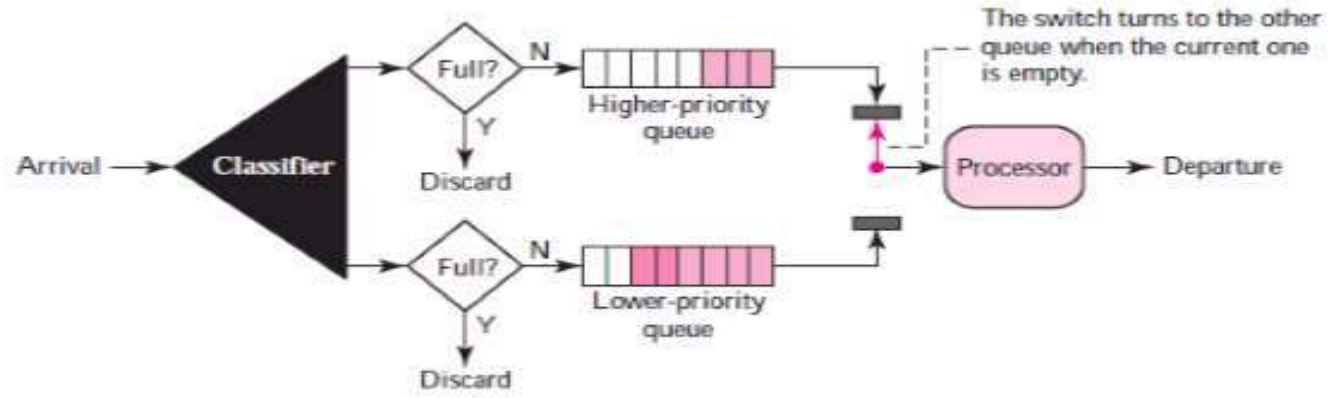
Scheduling

- Packets from different flows arrive at a switch or router for processing. A good scheduling technique treats the different flows in a fair and appropriate manner.
- Several scheduling techniques are designed to improve the quality of service.
- Three of them here: FIFO queuing, priority queuing, and weighted fair queuing.
- **FIFO Queuing: In first-in, first-out (FIFO)** queuing, packets wait in a buffer (queue) until the node (router or switch) is ready to process them. If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will be discarded. Figure9 shows a conceptual view of a FIFO queue.



Priority Queuing

- In priority queuing, packets are first assigned to a priority class. Each priority class has its own queue.
- The packets in the highest-priority queue are processed first. Packets in the lowest-priority queue are processed last.
- Note that the system does not stop serving a queue until it is empty.
- Figure shows priority queuing with two priority levels (for simplicity).
- A priority queue can provide better QoS than the FIFO queue because higher priority traffic, such as multimedia, can reach the destination with less delay.



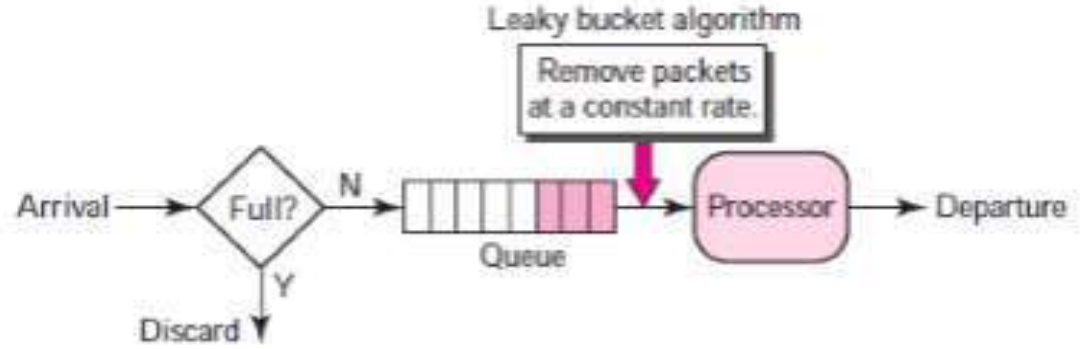
Weighted Fair Queuing

- A better scheduling method is weighted fair queuing. In this technique, the packets are still assigned to different classes and admitted to different queues.
- The queues, however, are weighted based on the priority of the queues; higher priority means a higher weight.
- The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight

Traffic Shaping :

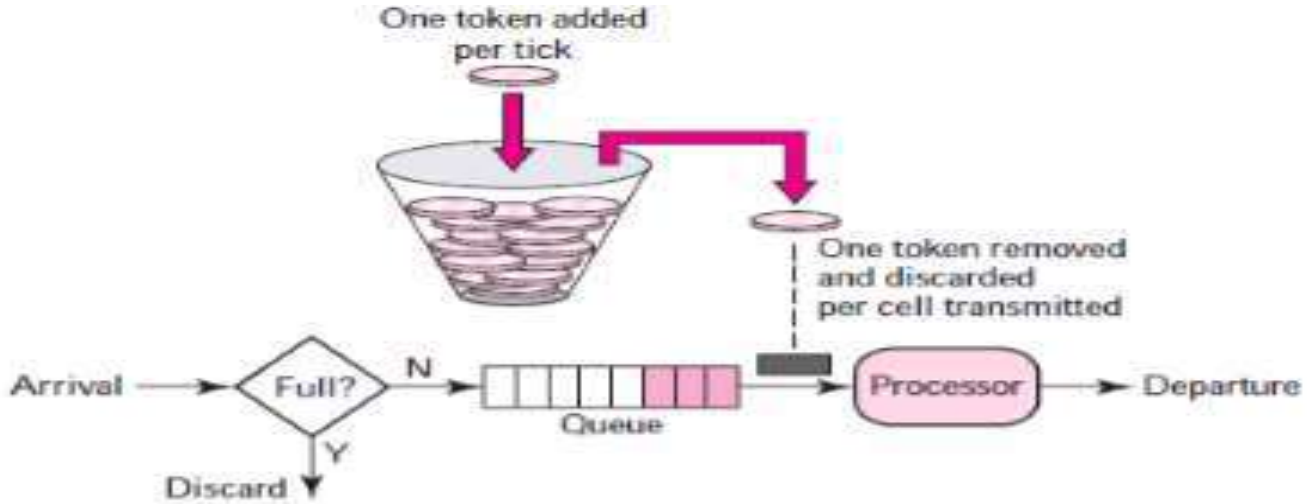
- Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network. Two techniques can shape traffic: leaky bucket and token bucket.
- **Leaky Bucket:**
- A technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate.
- A simple leaky bucket implementation is shown in Figure.
- A FIFO queue holds the packets. If the traffic consists of fixed-size packets, the 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:
- Initialize a counter to n at the tick of the clock.
- 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.
- Reset the counter and go to step 1.



Token Bucket

- The 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 1,000 ticks with 10 cells per tick. In other words, the host can send bursty data as long as the bucket is not empty



- **Resource Reservation :**
- A flow of data needs resources such as a buffer, bandwidth, CPU time, and so on.
- The quality of service is improved if these resources are reserved beforehand.
- **Admission Control :**
- Admission control refers to the mechanism used by a router, or a switch, to accept or reject a flow based on predefined parameters called flow specifications.
- Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity (in terms of bandwidth, buffer size, CPU speed, etc.) and its previous commitments to other flows can handle the new flow.

Ports and socket

- **The network port identifies the application or service running on the computer.**
- **The use of ports allow computers/devices to run multiple services/applications.**
- Ports is signified by optimistic (16-bit) numeral value between 0 and 65535
- The port numbers are allocated into three ranges
- Well Known Ports 0 to 1023. Assigned & controlled by IANA.
- Registered Ports, 1024 to 49151. Not assigned & controlled by IANA.But registered by IANA.
- Dynamic and/or Private Ports, 49152 to 65535.Not assigned & registered by IANA

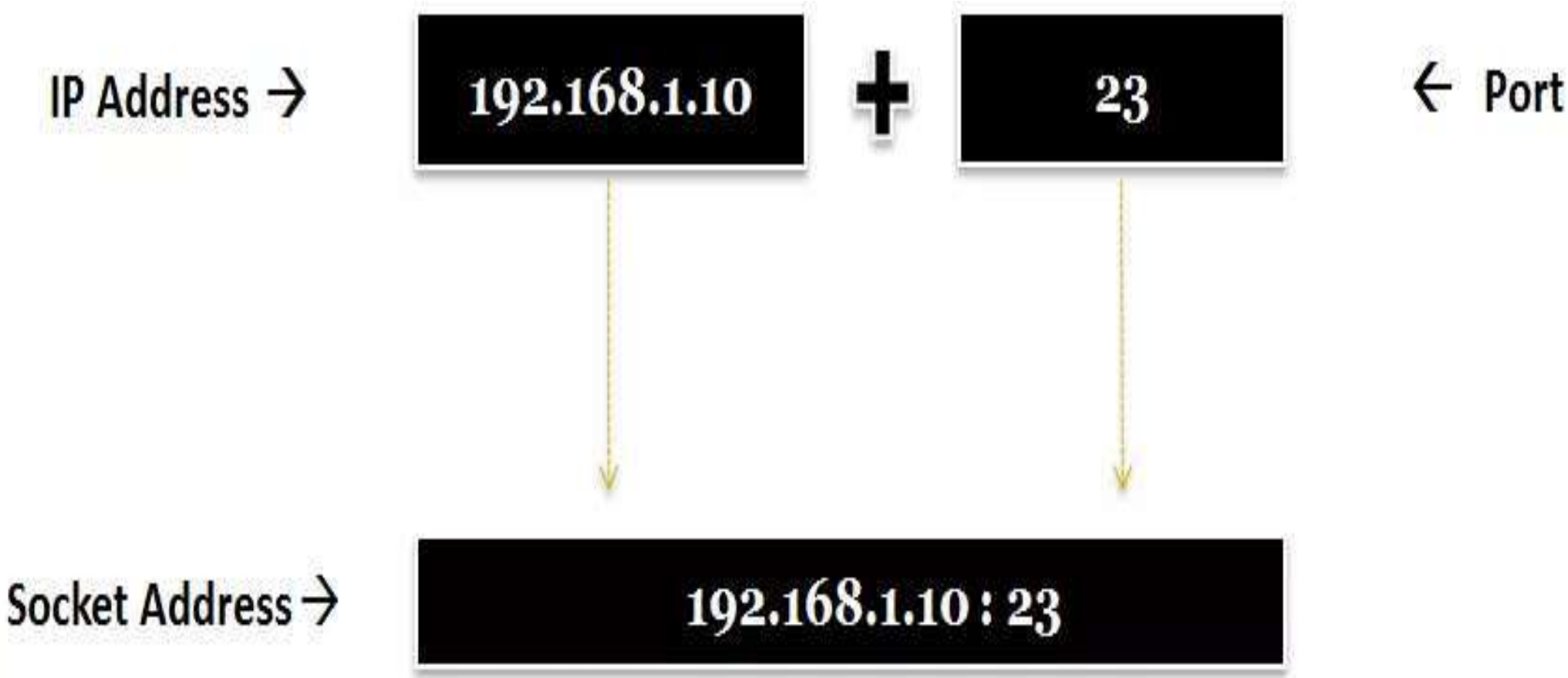
Transport Layer Port Numbers

Application Protocol	Transport Protocol	Port Number	Description
HTTP	TCP	80	Used by web browsers and web servers
Telnet	TCP	23	Used for terminal emulation
SSH	TCP	22	Used for secure terminal emulation
FTP	TCP	20, 21	Used for file transfer
DNS	UDP	53	Used for name-to-IP resolution
SMTP	TCP	25	Used to send Email
POP3	TCP	110	Used to receive Email
IMAP	TCP	143	Used to receive Email
SSL	TCP	443	Used to encrypt data for secure transaction
SNMP	UDP	161, 162	Used to manage TCP/IP networks

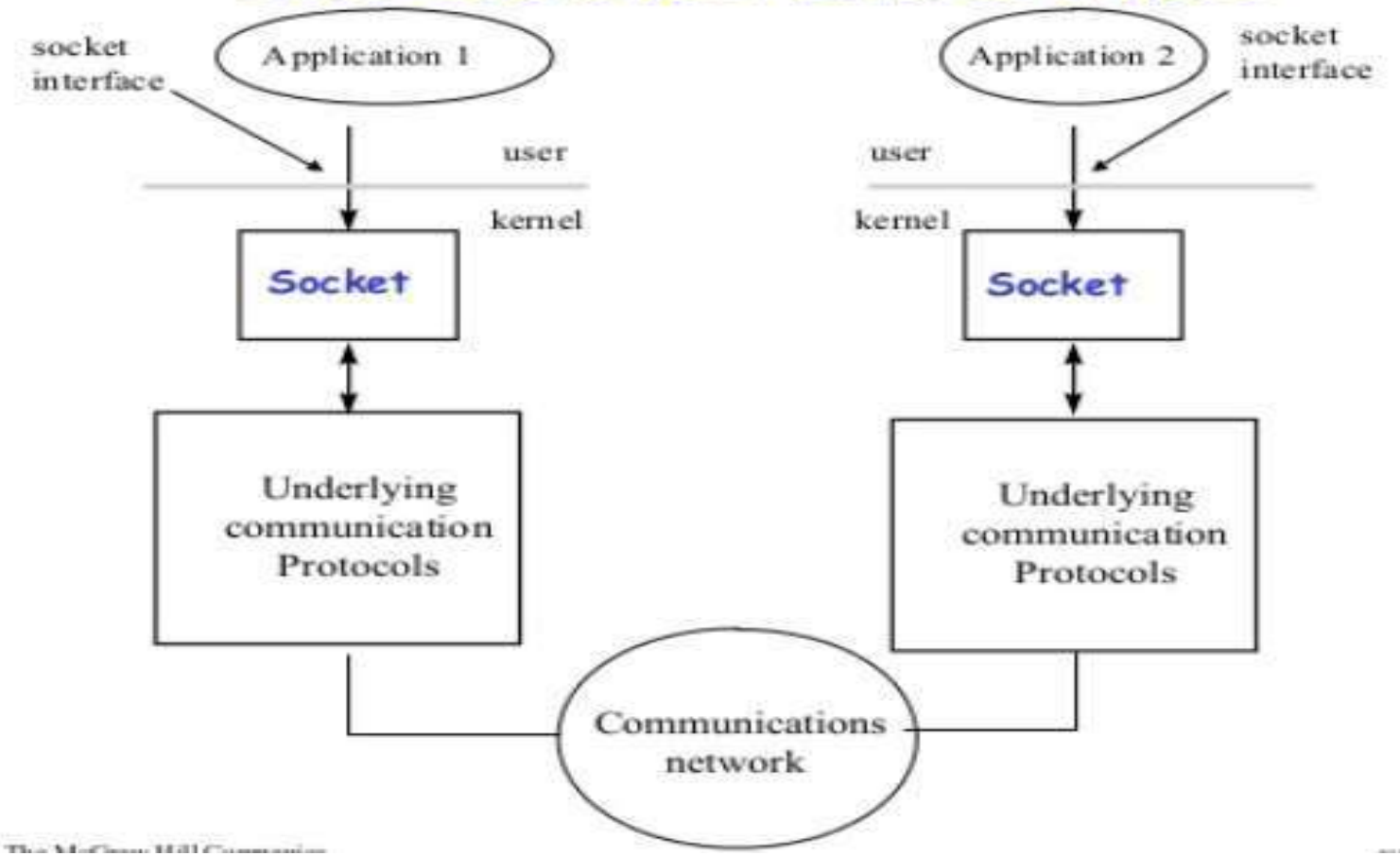
Common Application Protocols and Their Well-known Port Numbers *Table 10-*

Socket

- **A network socket** is one endpoint in a statement flow in the middle of two programs running over a network, also it is maintaining and allow communication between two different processes on the same or different machines.
- socket address is the combination of an IP address and a port number.
- In networking, a **socket** is used to allow many processes within a single or different host to use TCP communication simultaneously.
- A connection between two computers uses a socket.


IP Address →
192.168.1.10
23
← Port
Socket Address →
192.168.1.10 : 23

The Socket Interface



Types of Socket

- Stream **sockets** allow processes to communicate using TCP.
- A stream **socket** provides bidirectional, reliable, sequenced, and unduplicated flow of data with no record boundaries.
- Datagram **sockets** allow processes to use UDP to communicate. Raw **sockets** provide access to ICMP.

PORT	SOCKET
"Port" is a number used by a particular software. The same port may be used in different computers/servers running same software.	Socket is combination of “Port” and “IP address” to identify particular software and particular computer/Server
A socket is an end point of a bidirectional communication that occurs in a computer network that is based on the Internet protocol	A port is a logical data connection that can be used to exchange data without the use of a temporary file or storage
Ports operate at the Transport layer of the OSI	Sockets are a means of plugging the application layer in
Port is request running on that socket and port uses socket to deliver the packet to correct application.	A socket is the way a server and a client keep track of requests.
A port functions like a telephone number, identifying the machine and giving the socket an area to connect.	While the socket functions like a cord that ties the computers together.

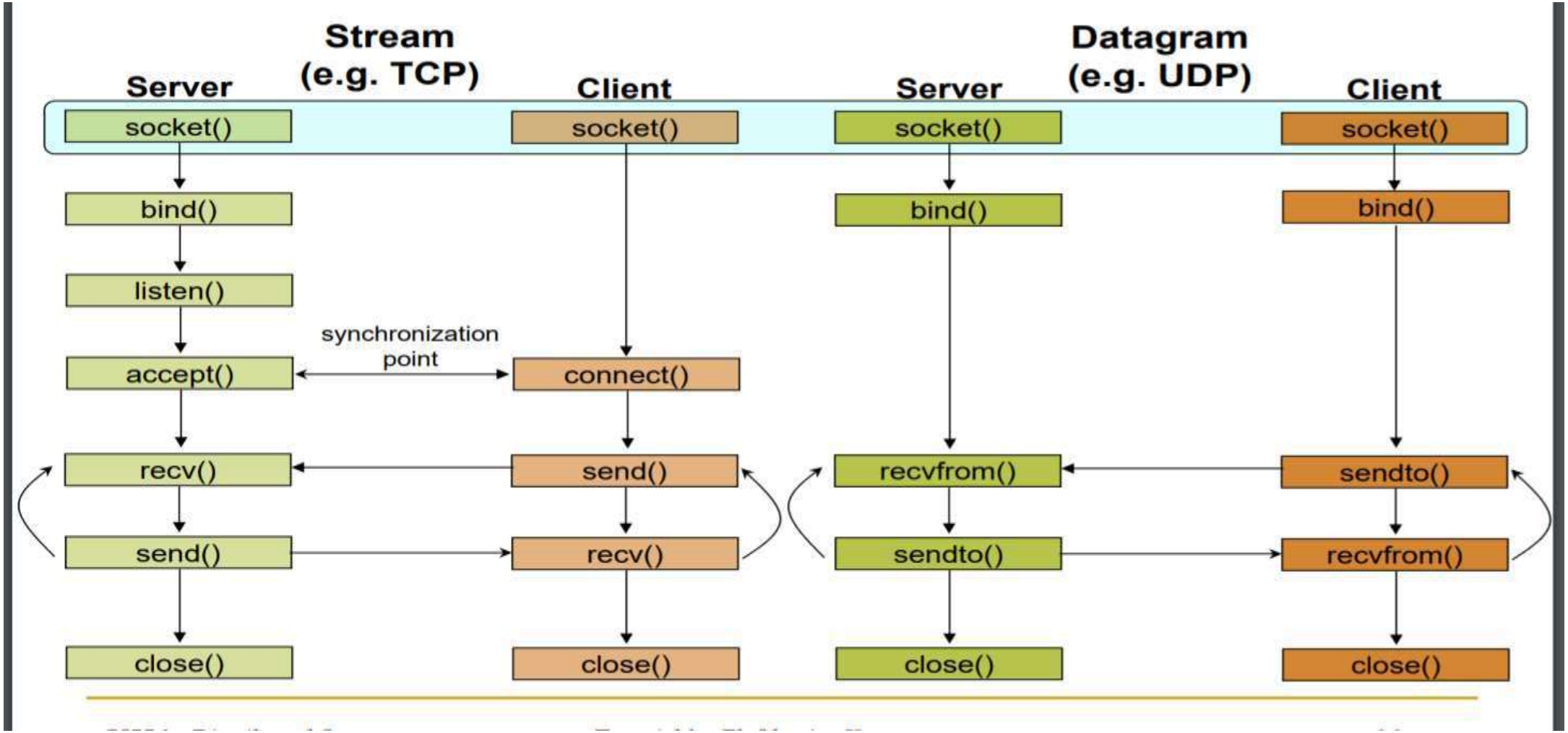
Socket Programming

- Socket programming is a way of connecting two nodes on a network to communicate with each other.
- One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

Socket Procedure

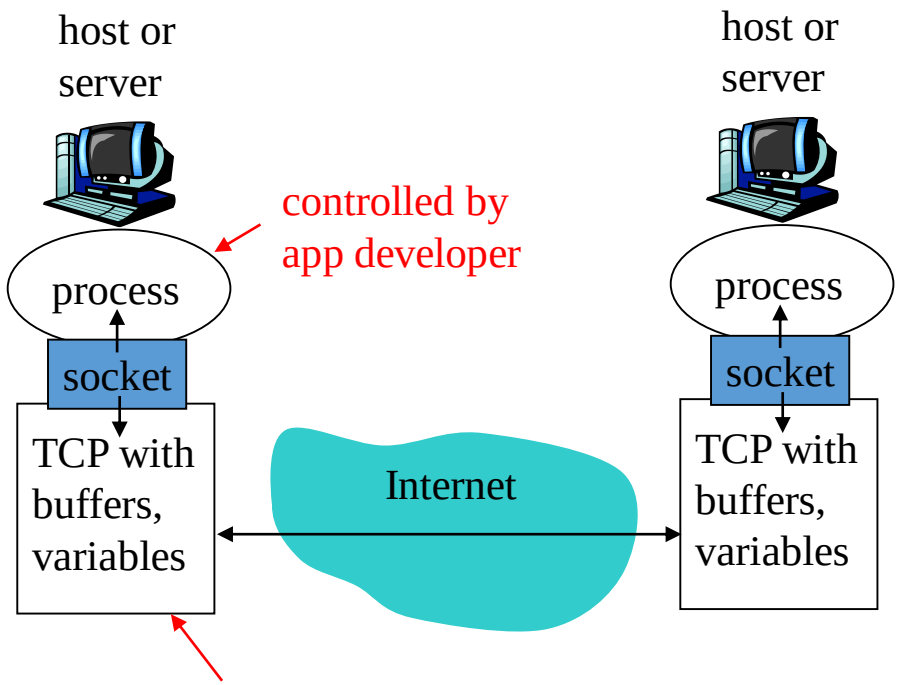
Primitive	Meaning
Socket	Create a new communication endpoint
Bind	Attach a local address to a socket
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Send	Send some data over the connection
Receive	Receive some data over the connection
Close	Release the connection

Socket Programming



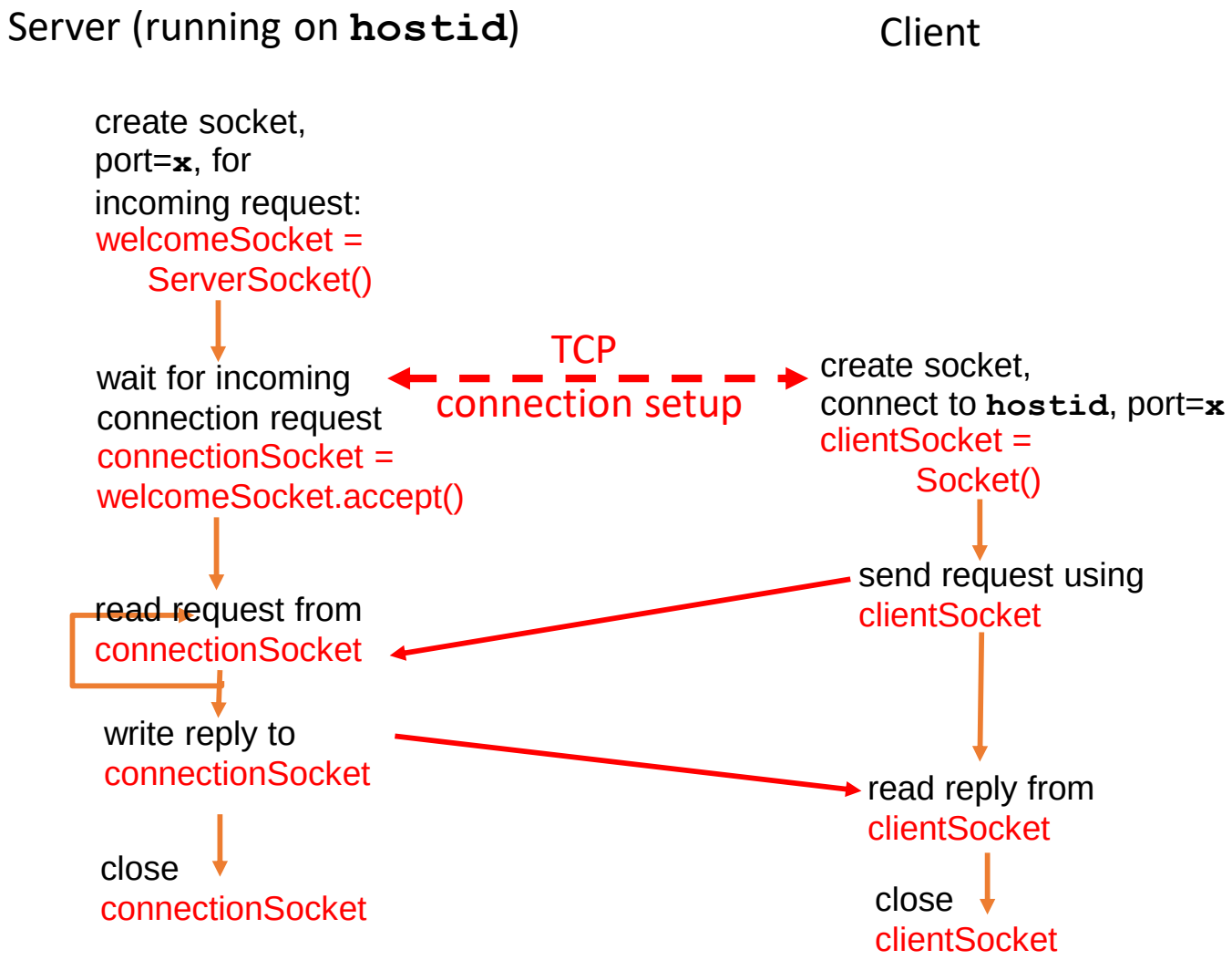
Socket programming

1. A client reads a line from its standard input (keyboard) and sends the line out its socket to the server
2. The server reads a line from its socket
3. The server converts the line to uppercase
4. The server sends the modified line out its socket to the client
5. The client reads the modified line from its socket and prints the line on its standard output(monitor)


host or server
host or server
controlled by app developer
process
process
socket
socket
TCP with buffers, variables
TCP with buffers, variables
Internet

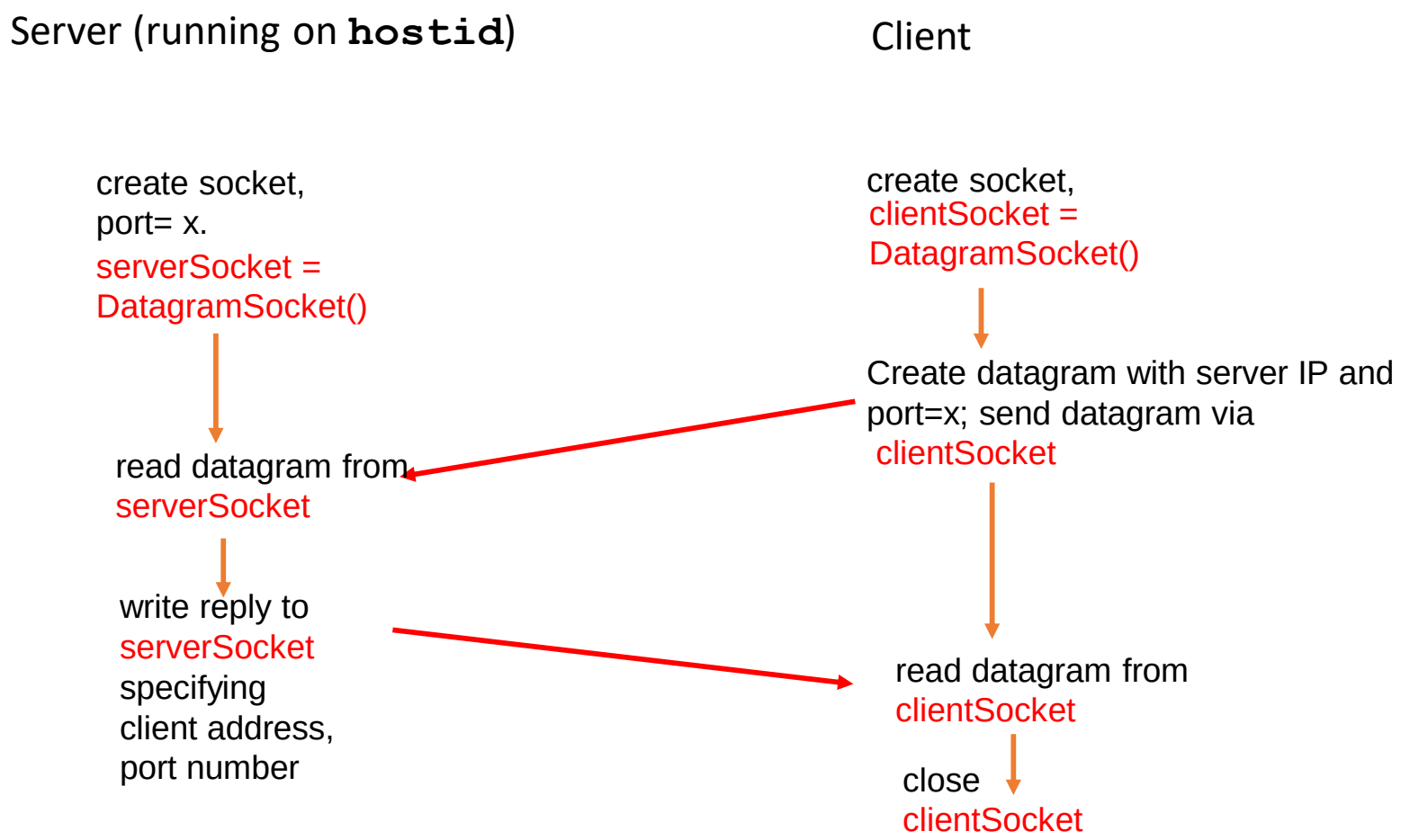
Socket programming with TCP

- Reliable, byte stream-oriented
- Three-way handshake
- Three sockets
- Streams


Server (running on hostid)
Client
create socket, port=x, for incoming request: welcomeSocket = ServerSocket()
wait for incoming connection request connectionSocket = welcomeSocket.accept()
TCP connection setup
create socket, connect to hostid, port=x clientSocket = Socket()
send request using clientSocket
read request from connectionSocket
write reply to connectionSocket
read reply from clientSocket
close connectionSocket
close clientSocket

Socket programming with UDP

- No streams
- No connection
- Datagram socket
- Individual packets


Server (running on hostid)
Client
create socket,
port= x.
serverSocket =
DatagramSocket()
read datagram from
serverSocket
write reply to
serverSocket
specifying
client address,
port number
create socket,
clientSocket =
DatagramSocket()
Create datagram with server IP and
port=x; send datagram via
clientSocket
read datagram from
clientSocket
close
clientSocket