TCP

Topics

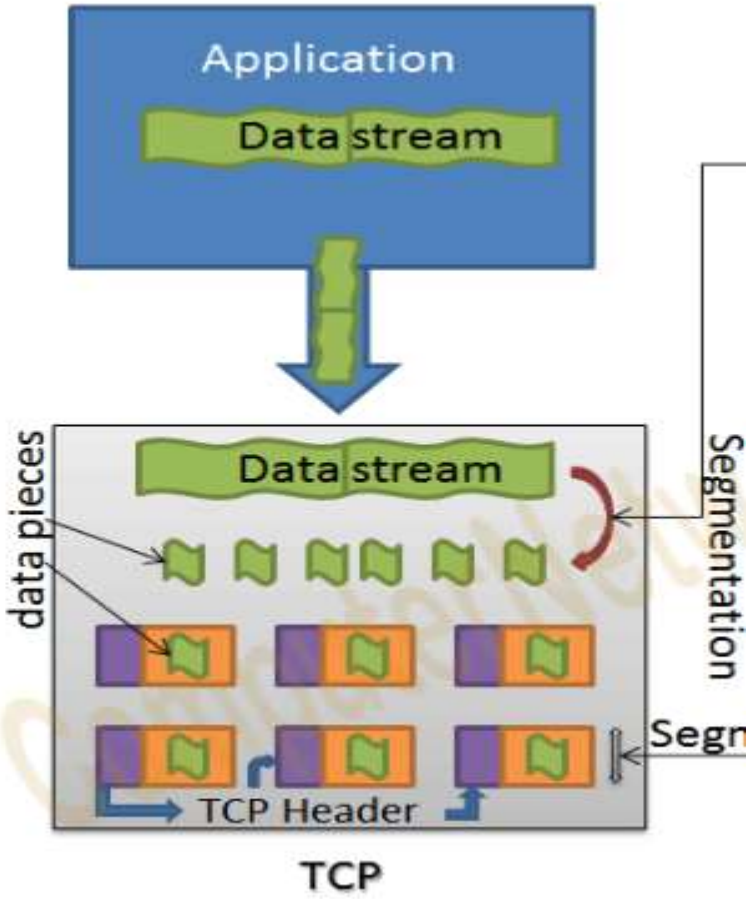
TCP Services
TCP Features
Segment
A TCP Connection
Flow Control
Error Control

Transport layer

- A process is an application-layer entity (running program) that uses the services of the transport layer.
- Track conversation at application endpoints(process to process delivery)
- Software that runs as service/process
- Sockets –buffer in transport layer

1. Tracks conversation
2. Identifies the application
3. Segmentation and reassembly
4. Multiplexing

Segmentation



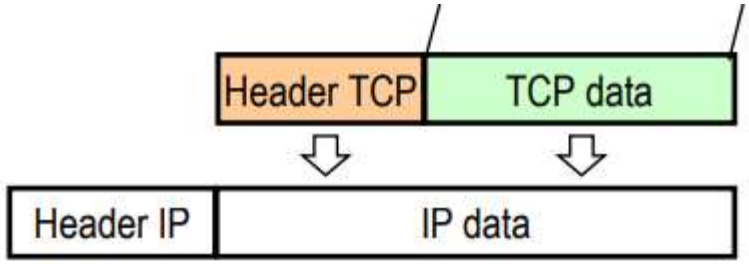
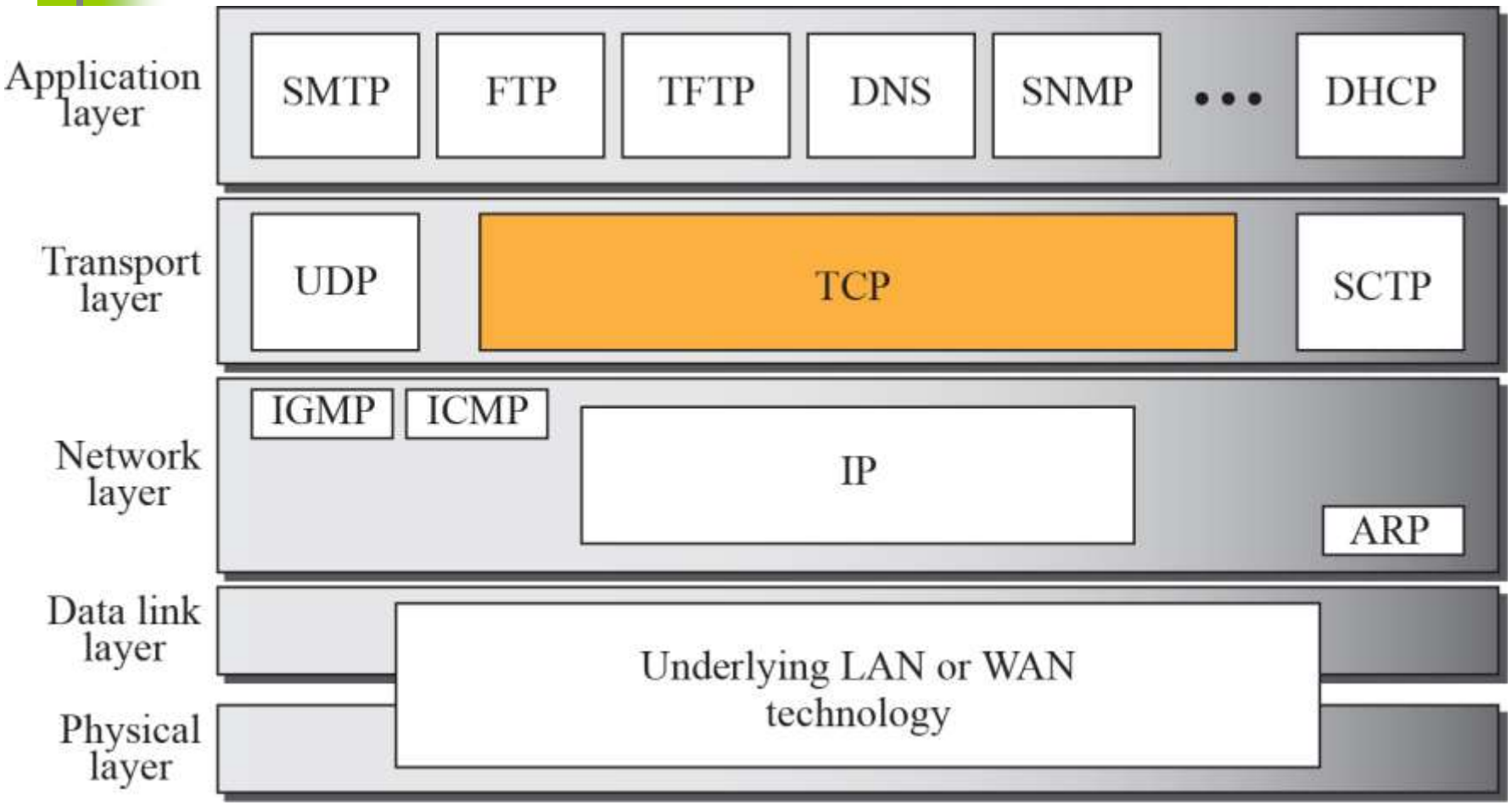


Figure 15.1 *TCP/IP protocol suite*



UDP –Fast Tcp-slow ,reliable a,ack

UDP--Real time streaming and broadcasting.,IPTV,VOD,VOIP,DHCP ,DNS, SNMP

TCP-HTTP,FTP,Telnet,SMTP

TCP

TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs (process) to send data.

In addition,TCP uses flow and error control mechanisms at the transport level.

- TCP adapts to internetwork failures and provides reliable service

- All TCP connections are full-diplex and point-to-point. TCP does not support multicasting or broadcasting.
- Uses **Sockets** to define an end-to-end connection (Source IP, Source Port, Source Initial Sequence Number, Destination IP, Destination Port, Destination Initial Sequence Number)

TCP Services

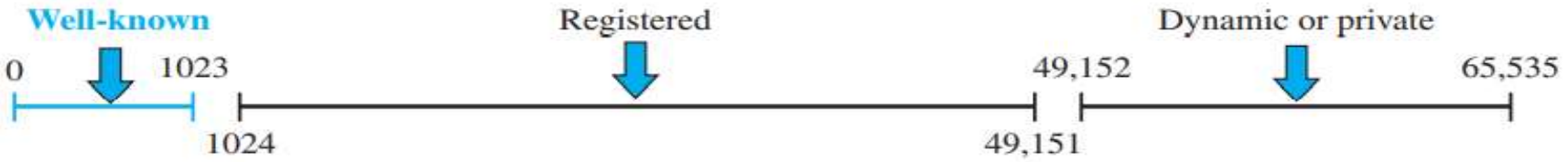
1)TCP SERVICES

a)Well known ports for TCP

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

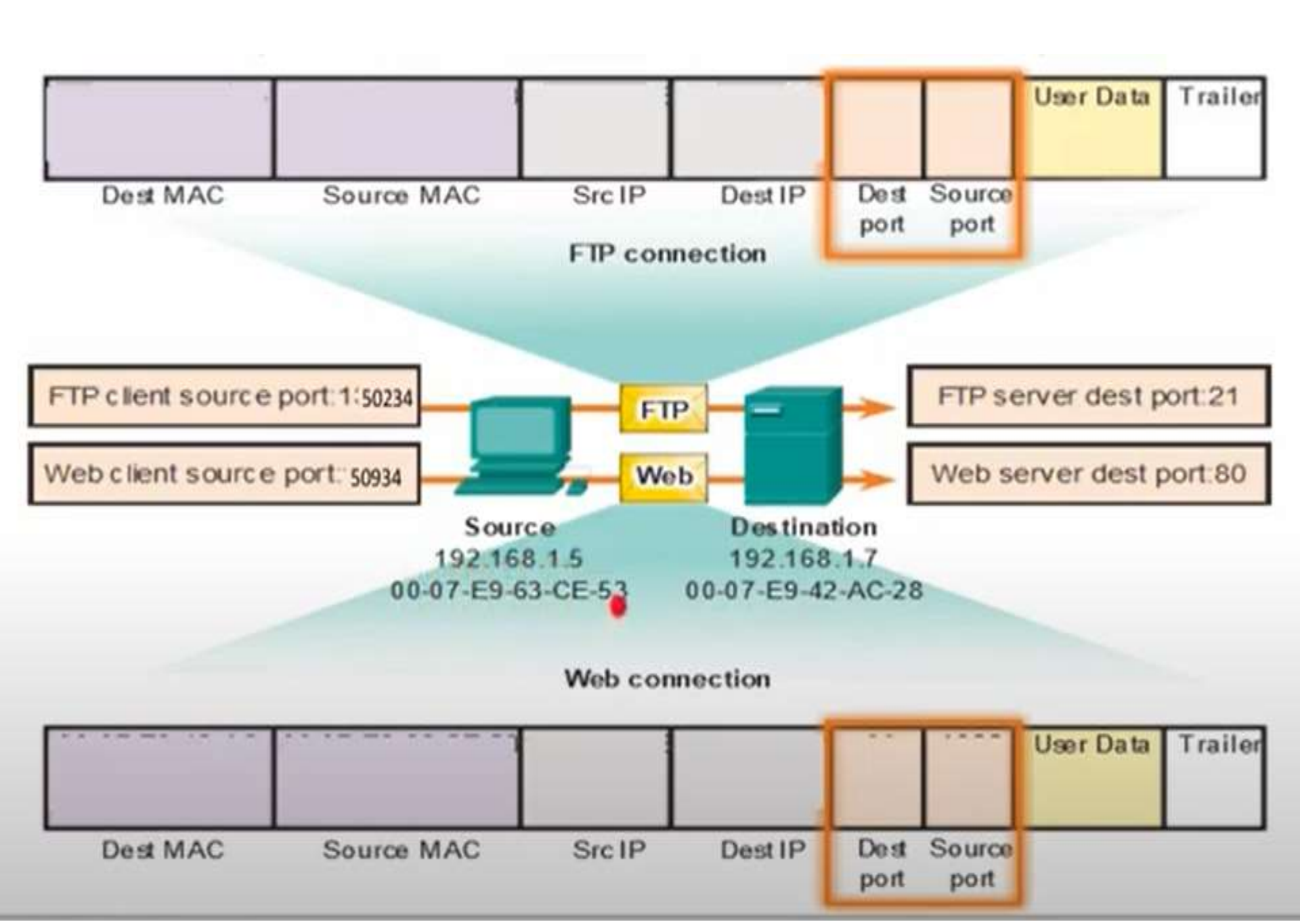
Process-to-Process Communication

Figure 23.5 *ICANN ranges*



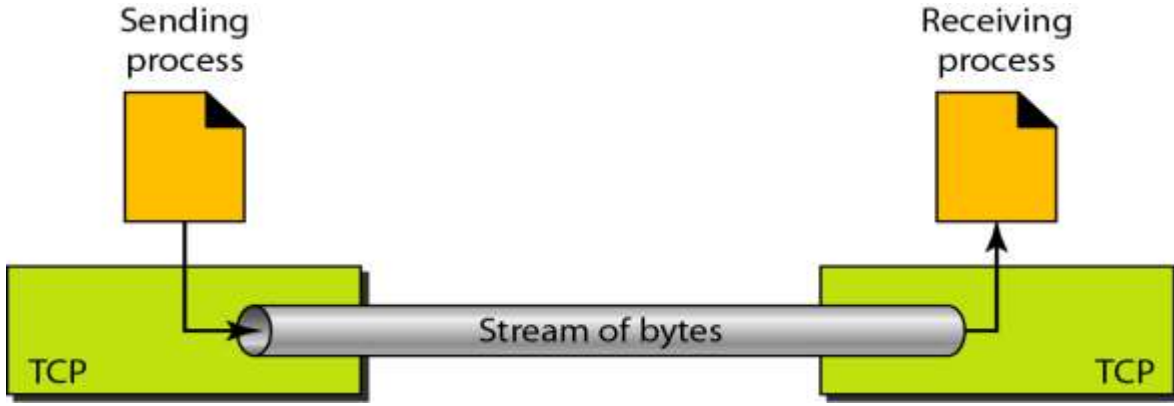
- ***Well-known ports.*** The ports ranging from 0 to 1023 are assigned and controlled by ICANN. These are the well-known ports.
- ***Registered ports.*** The ports ranging from 1024 to 49,151 are not assigned or controlled by ICANN. They can only be registered with ICANN to prevent duplication.
- ***Dynamic ports.*** The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used as temporary or private port numbers.

https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml


User Data
Trailer
Dest MAC
Source MAC
Src IP
Dest IP
Dest port
Source port
FTP connection
FTP client source port: 1:50234
Web client source port: 50934
FTP
Web
FTP server dest port:21
Web server dest port:80
Source
192.168.1.5
00-07-E9-63-CE-53
Destination
192.168.1.7
00-07-E9-42-AC-28
Web connection
User Data
Trailer
Dest MAC
Source MAC
Src IP
Dest IP
Dest port
Source port

b)Steam delivery service

- Neither IP nor UDP recognizes relation between datagrams
- But TCP allows the sending process to deliver data as stream of bytes
- TCP creates an imaginary pipe between sending sending and receiving process
- **TCP numbers each byte flowing thru the stream**
- Sending process produces a stream of bytes and receiving process consumes production and consumption have different speeds buffers for storage



b)Steam delivery service

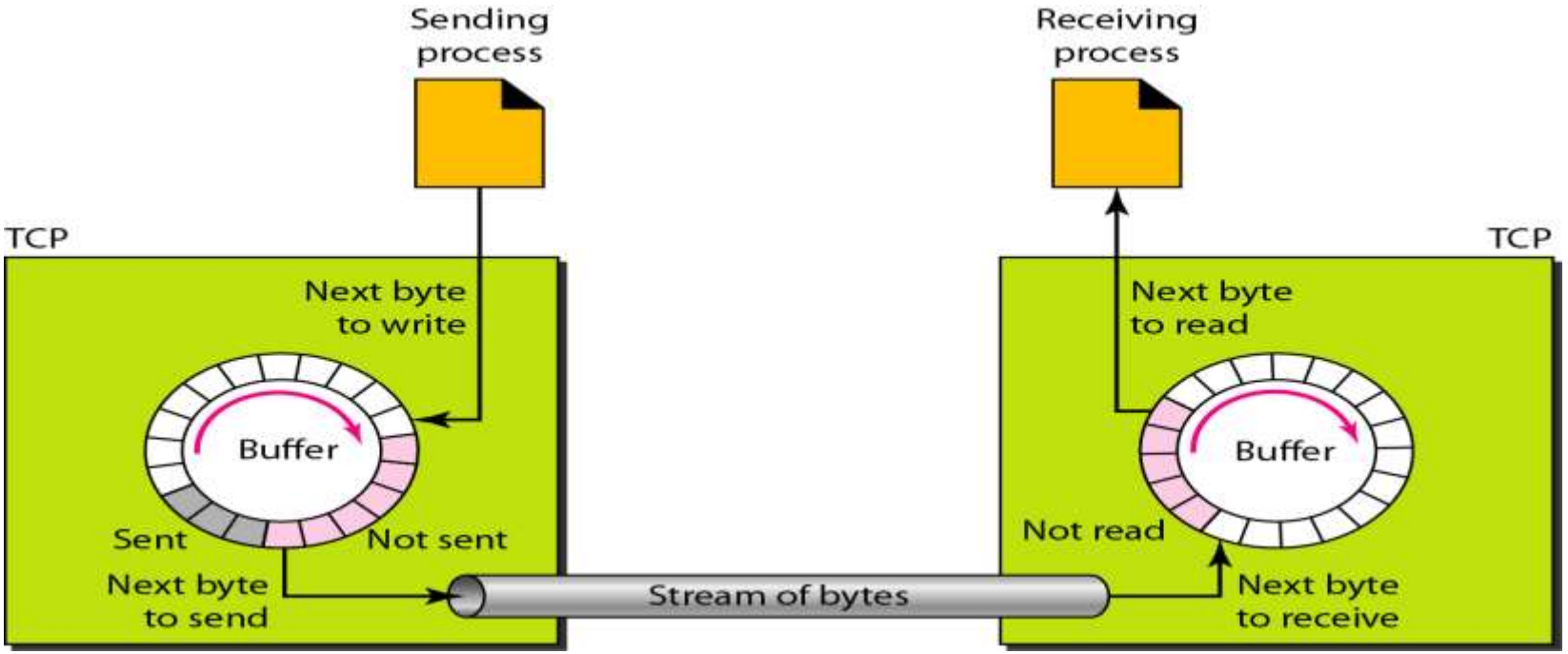
- Message boundaries are not preserved end-to-end

Example:

- The sending process does four 512 byte writes to a TCP stream – for `write()` call to the TCP socket
- These data may be delivered as – four 512 byte chunks, two 1024 byte chunks, one 2048 byte chunk or some other way
- There is no way for the receiver to detect the unit(s) in which the data were written by the sending process.

c)Sending and receiving buffers

Buffer for storage
Necessary for flow and error control
Buffer is a circular array of 1 byte locations

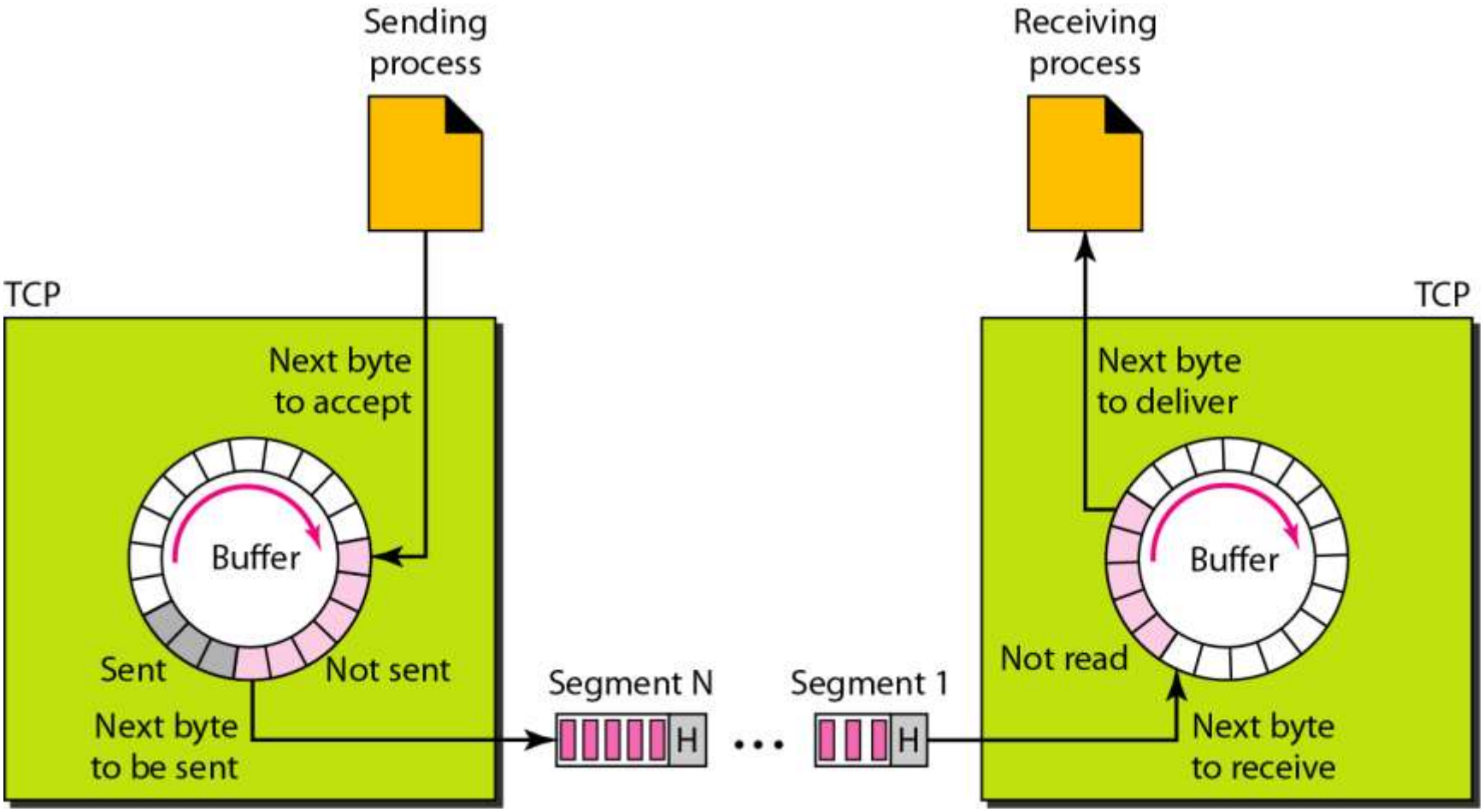


- Sending buffer
- Empty locations that can be filled by sending process
- Bytes that have been sent but not yet acknowledged
- Bytes to be sent by sending TCP
- Receiving buffer
- Empty locations to be filled with bytes received from the network
- Bytes that can be consumed by the receiving process

d)Segments

- IP layer sends data in packets, not as stream of bytes
- TCP at the sending site gathers bytes into a packet called a **segment**
- TCP adds a header to each segment and delivers it to IP for transmission
- **Segments encapsulated in an IP datagram** and transmitted
- Segments can arrive out of order, lost, or corrupted and resent
- Size of the segment varies

d)Segments


Sending process
Receiving process
TCP
TCP
Next byte to accept
Next byte to deliver
Buffer
Buffer
Sent
Not sent
Not read
Segment N
Segment 1
H
H
Next byte to be sent
Next byte to receive

e)Full duplex communication

- Data can flow in both directions at the same time
- Each TCP has a sending and receiving buffer

f) Connection-oriented service

1)Connection establishment

2)data transfer

3)connection termination

- When process at site A wants to send and receive data from another process at site B
- A’s TCP informs B’s TCP and gets approval from B’s TCP
- A’s TCP and B’s TCP exchange data in both directions
- After both processes have no data left to send and the buffers are empty, two TCPs destroy their buffers
- A virtual connection is created

g)Reliable service

- TCP uses an **ACK** mechanism to ensure arrival of data
- Reliable transport protocol

TCP Features

2)TCP Features

- To provide services TCP has several features

✓ Numbering System

✓ Flow Control

✓ Error Control

✓ Congestion Control

a)Numbering System

- Two fields are used **sequence number and acknowledgment number.**
- These are used to keep **track of the segments** being transmitted or received
- Both refer to byte number and not segment number
- All data bytes transmitted are numbered (independent in each
- direction)
- Numbers generated randomly from 0 to 2^{32}-1
- **TCP numbers each byte flowing thru the stream**
- **Byte numbering is used for flow and error control**

Sequence Number

- **After bytes are numbered, TCP assigns a sequence number to each segment that is being sent**

- **It is the number of first byte of data carried in that segment**

TCP NUMBERING – AN EXAMPLE

Imagine a TCP connection is transferring a file of 6000 bytes.

Tcp generates a random number 1057

So the bytes are numbered from 1057 to 7057

What are the sequence numbers for each segment if data are sent in five segments with the first four segments carrying 1000 bytes and the last segment carrying 2000 bytes?

The first byte is numbered 10010.

The following shows the sequence number for each segment:

Segment 1 ==> sequence number: 10 010 (range: 10,010 to 11,009)
Segment 2 ==> sequence number: 11 010 (range: 11,010 to 12,009)
Segment 3 ==> sequence number: 12 010 (range: 12,010 to 13,009)
Segment 4 ==> sequence number: 13 010 (range: 13,010 to 14,009)
Segment 5 ==> sequence number: 14 010 (range: 14,010 to 16,009)

Acknowledgment Number

- Sender sends TCP segment with data to receiver
- Receiver wants to let sender know:
  - **the bytes it received correctly so far**
  - **the next byte that it waits to receive**
- **ACK number** defines the next byte that the party (receiver) expects to receive
- ACK is cumulative

b)Flow control

- The amount of data a source can send before receiving an ACK from the destination
- Whether to send 1 byte of data and wait for ACK or send all bytes and wait for the ACK for the complete message?
- Sliding window protocol
  - byte oriented
- **Receiving TCP controls how much data can be sent by the sending TCP**
- Receiver must not be overwhelmed with data
- **Flow control** - implemented through TCP window – **"rwnd"** (receiver window)
- The window size is set by the receiving TCP and sender TCP must accept it.

d)Error control

- **TCP provides reliability using error control**
- **Detect corrupted segments; lost segments; out-of-order segments & duplicated segments-retransmission**
- **Error control in TCP is achieved through the use of three simple tools:**
  - **Checksum – MANDATORY IN TCP**
  - **Acknowledgement**
  - **Time-out**

- **Retransmission is heart of error control. Happens when:**

- **1) Retransmission time-out (RTO) timer expiry causes associated TCP segment to be retransmitted**

- **2) 3 dup ACK: for faster retransmission, instead of waiting for timer expiry**

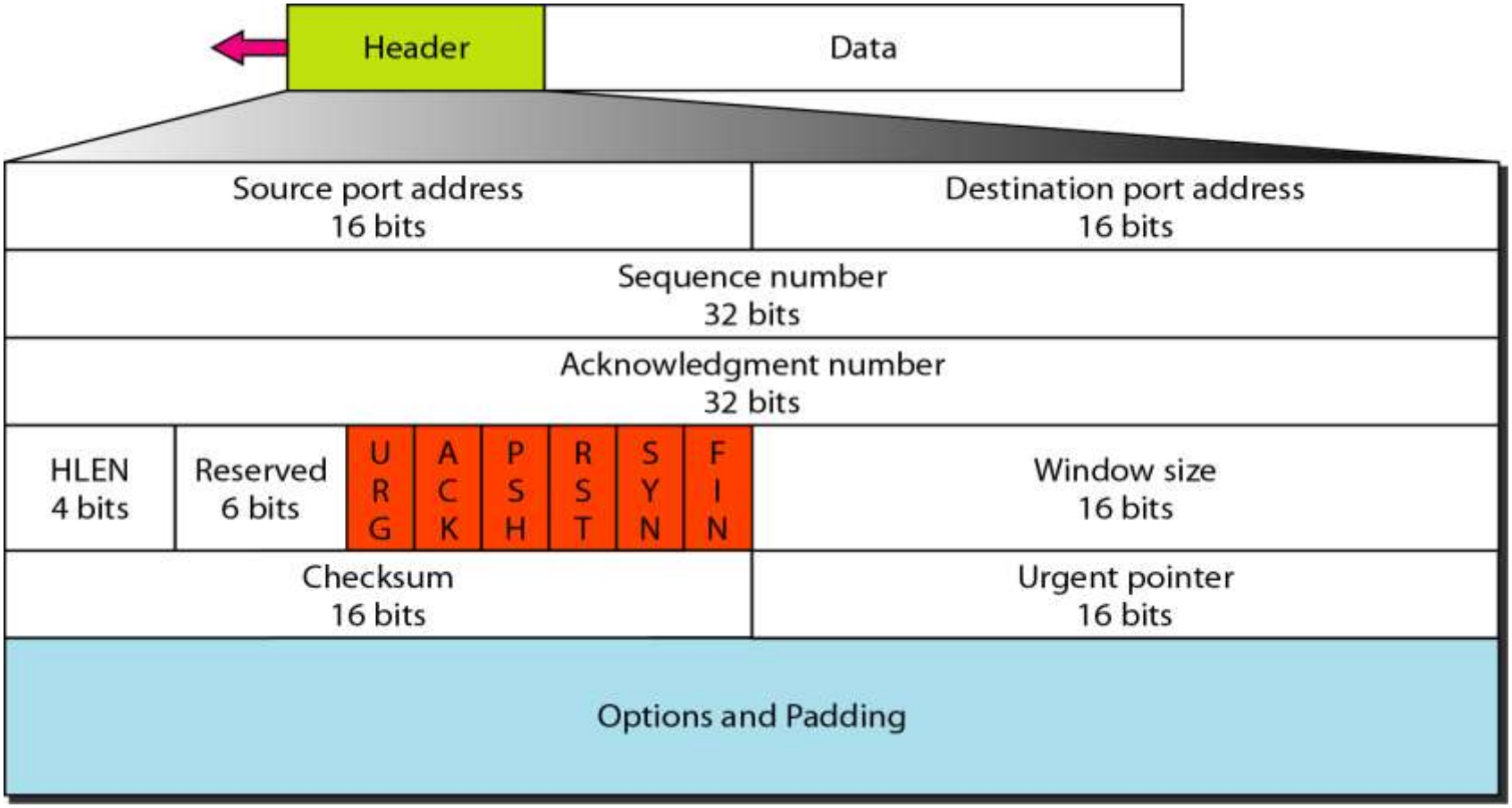
e)Congestion control

- The amount of data sent by a sender is not only controlled by the receiver (flow control)
- Also determined by the level of congestion in the network

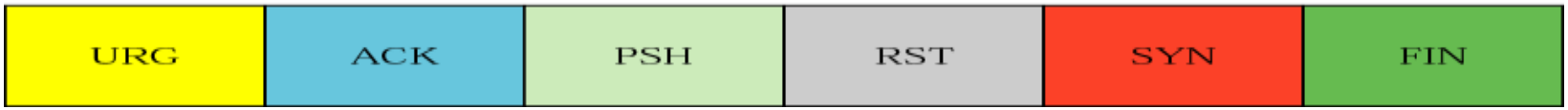
TCP Segment

A TCP Connection
Flow Control
Error Control

3)Segment

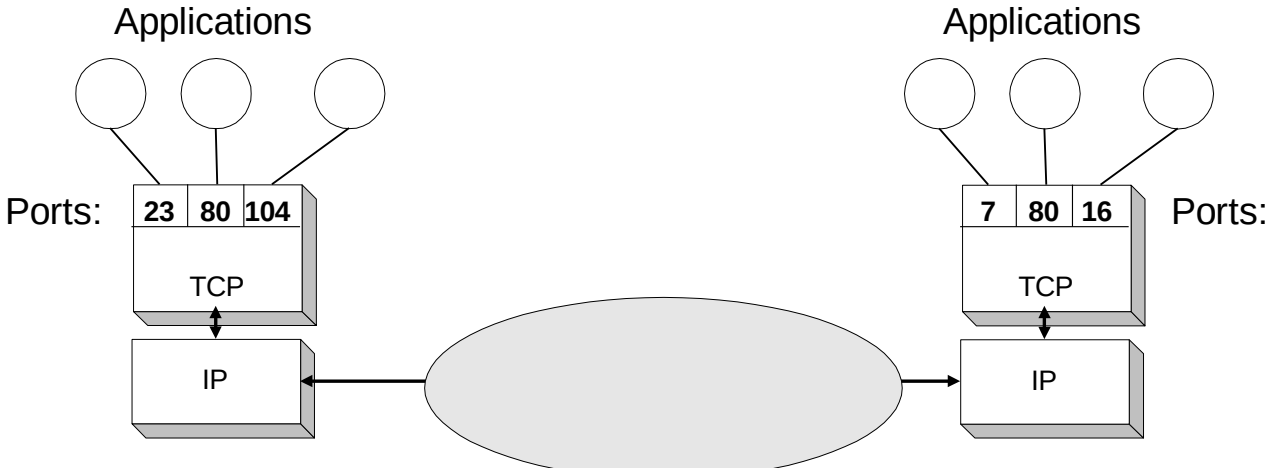


URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection



TCP header fields

- **Port Number:**
  - A port number identifies the endpoint of a connection.
  - A pair `<IP address, port number>` identifies one endpoint of a connection.
  - Two pairs `<client IP address, server port number>` and `<server IP address, server port number>` identify a TCP connection.



TCP header fields

- **Sequence Number (SeqNo):**
  - Sequence number is 32 bits long.
  - So the range of SeqNo is

    $0 <= SeqNo <= 2^{32} - 1 \approx 4.3$ Gbyte
  - Each sequence number identifies a byte in the byte stream
  - Initial Sequence Number (ISN) of a connection is set during connection establishment

TCP header fields

- **Acknowledgement Number (AckNo):**
  - Acknowledgements are piggybacked, I.e

    a segment from A -> B can contain an acknowledgement for a data sent in the B -> A direction

  - A hosts uses the AckNo field to send acknowledgements. (If a host sends an AckNo in a segment it sets the "**ACK flag**")
  - The AckNo contains the next SeqNo that a hosts wants to receive

    Example: The acknowledgement for a segment with sequence numbers 0-1500 is AckNo=1501

TCP header fields

- **Acknowledge Number (cont'd)**
  - TCP uses the sliding window flow protocol to regulate the flow of traffic from sender to receiver
  - TCP uses the following variation of sliding window:
    - **no NACKs (Negative ACKnowledgement)**
    - **only cumulative ACKs**
- Example:

  Assume: Sender sends two segments with "1..1500" and "1501..3000", but receiver only gets the second segment.

  In this case, the receiver cannot acknowledge the second packet. It can only send AckNo=1

TCP header fields

- **Header Length (4bits):**
  - Length of header in 32-bit words
  - Note that TCP header has variable length (with minimum 20 bytes)

TCP header fields

- **Flag bits:**
  - **URG: Urgent pointer is valid(part of the segment)**
    - If the bit is set, the following bytes contain an urgent message in the range:

      SeqNo <= urgent message <= SeqNo+urgent pointer
  - **ACK: Acknowledgement Number is valid**
  - **PSH: PUSH Flag(Entire Segment)**
    - Notification from sender to the receiver that the receiver should pass all data that it has to the application.
    - Normally set by sender when the sender's buffer is empty

TCP header fields

- **Flag bits(Connection):**
  - **RST: Reset the connection**
    - The flag causes the receiver to reset the connection
    - Receiver of a RST terminates the connection and indicates higher layer application about the reset
  - **SYN: Synchronize sequence numbers**
    - Sent in the first packet when initiating a connection
  - **FIN: Sender is finished with sending**
    - Used for closing a connection
    - Both sides of a connection must send a **FIN**

TCP header fields

- **Window Size:**
  - Each side of the connection advertises the **window size**
  - Window size is the maximum number of bytes that a receiver can accept.
  - Maximum window size is 2^{16}-1= 65535 bytes
- **TCP Checksum:**
  - TCP checksum covers over both TCP header **and** TCP data (also covers some parts of the IP header)
- **Urgent Pointer:**
  - Only valid if **URG** flag is set

A TCP Connection

Flow Control
Error Control

4)TCP CONNECTION

TCP is connection-oriented.

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

All of the segments belonging to a message are then sent over this virtual path.

TCP uses a **three-way handshake** to open a connection

A connection-oriented transmission requires three phases:

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

1)CONNECTION ESTABLISHMENT

- The process starts with the server
- Server tells the TCP its ready to accept a connection
- This is called a request for passive open
- A client wishes to connect to an open server tells its TCP it needs to be connected to a particular server
- The client requests for an active open
- Now TCP can start the three way handshaking process

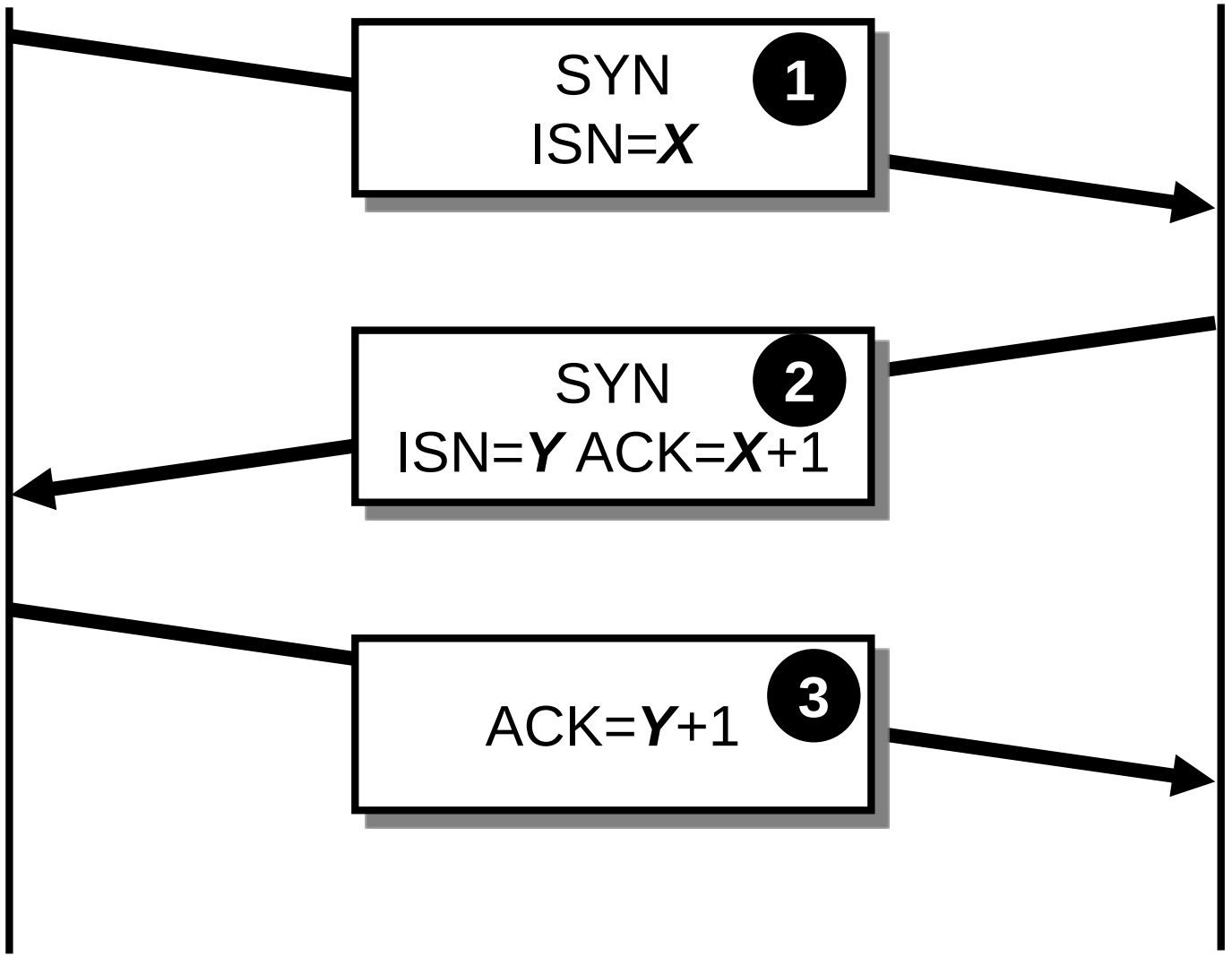
TCP 3-way handshake

1. Client: “I want to talk, and I’m starting with byte number ***X***”.

2. Server: “OK, I’m here and I’ll talk. My first byte will be called number ***Y***, and I know your first byte will be number ***X+1***”.

3. Client: “Got it - you start at byte number ***Y+1***”.

TCP 3-way handshake

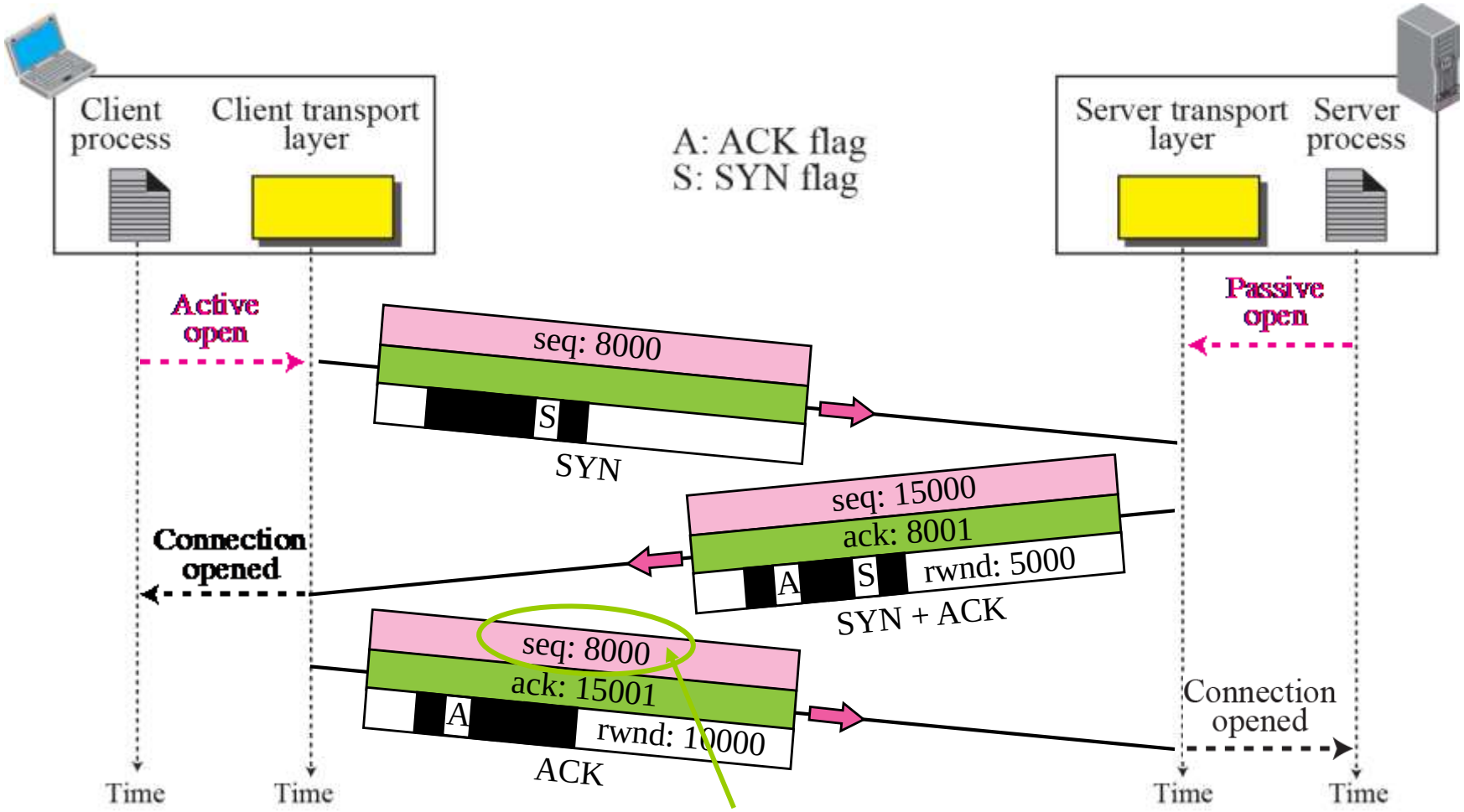


A synchronize at byte 300 and B at 100,show seq no and ack no and the flags

TCP conversation-Memes

"Hi, I'd like to hear a TCP joke."
"Hello, would you like to hear a TCP joke?"
"Yes, I'd like to hear a TCP joke."
"OK, I'll tell you a TCP joke."
"Ok, I will hear a TCP joke."
"Are you ready to hear a TCP joke?"
"Yes, I am ready to hear a TCP joke."
"Ok, I am about to send the TCP joke. It will last 10 seconds, it has two characters, it does not have a setting, it ends with a punchline."
"Ok, I am ready to get your TCP joke that will last 10 seconds, has two characters, does not have an explicit setting, and ends with a punchline."
"I'm sorry, your connection has timed out. ... Hello, would you like to hear a TCP joke?"

Connection establishment using three-way handshake



seq: 8001 if piggybacking

a)First step

- A client starts by sending a SYN segment syn flag set.
- consumes one sequence number
- data transfer starts,seq number is incremented
- carries no data

b)Second step

- The server sends the syn+ack segment with 2 flag bits set
- This serves as two purposes
  - Synchronization for communication
  - Acknowledgement for the syn segment from the client side
  - Consumes one sequence number

c)Third step

- Client sends the third acknowledgement segment
- Sets the ack flag
- Acknowledges the packet received from the server by sending the next seq no
- **Uses the same seq no ,does not consume seq no**

- Simultaneous open –when both client and server issue an active open

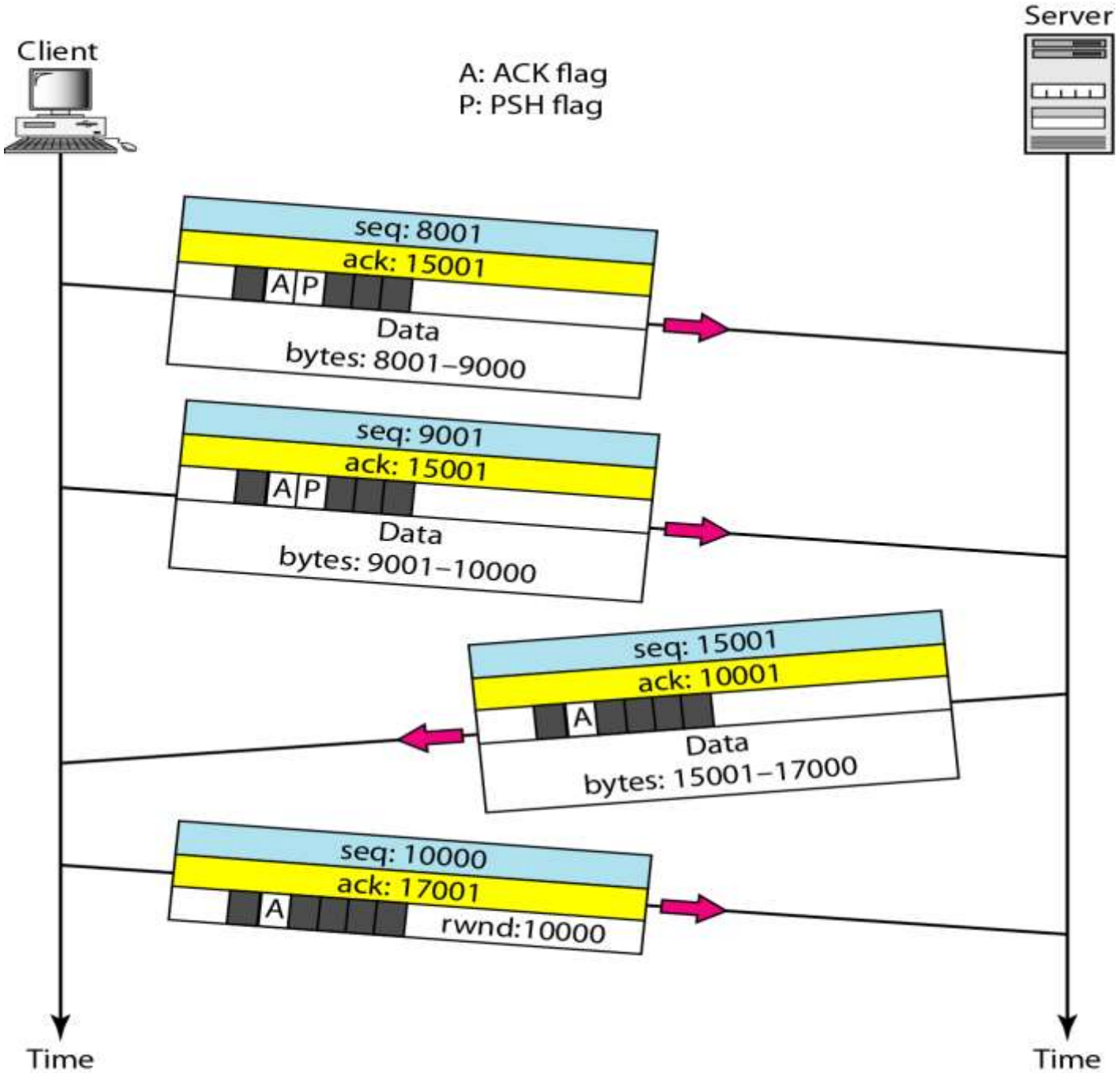
- Half close-one end stop sending data while still receiving data

- Syn flooding attack-a malicious attacker sends a large number of syn segment to the server ,faking as if coming from different IP adddress

- Attack belongs to a type of security attack called denial of service where an attacker monopolizes a system with requests so the server will deny every request

2)Data Transfer

- Once the connection is established, data can be sent.
- Data can be sent bidirectional
- Each **data segment includes a sequence number** identifying the first byte in the segment.
- Each segment (data or empty) includes a request number indicating what data has been received.
- The **data is put in a buffer**, where it stays until the data is ACK'd.


Client
Server
A: ACK flag
P: PSH flag
seq: 8001
ack: 15001
A P
Data
bytes: 8001–9000
seq: 9001
ack: 15001
A P
Data
bytes: 9001–10000
seq: 15001
ack: 10001
A
Data
bytes: 15001–17000
seq: 10000
ack: 17001
A
rwnd:10000
Time
Time

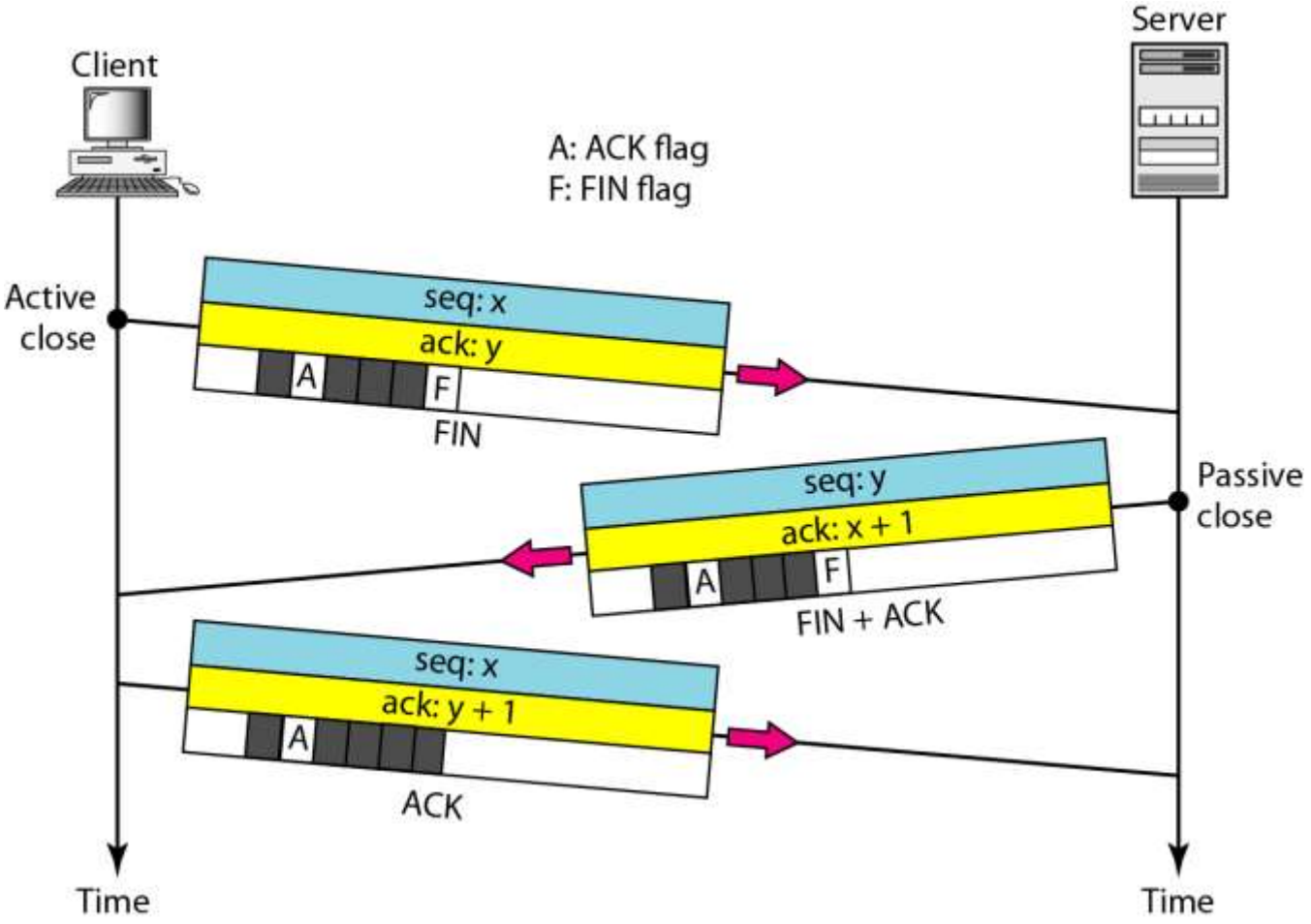
- Pushing data
  - Tells that the sending site must frame a segment immediately and send it to the receiving site
  - Must not wait for more data to come
- Urgent data
  - Sending application wants a piece of data to be read out of order by receiving application program
  - Ex:sending application is sending large amount of data to the receiving appln for processing .The R.appln process and sends the result.The S.appln finds the result wrong and wants to abort the process so if it issues abort,it will get stored in the last of queue.Any way the data will be processed before abort process.so the soln is to send urg where it gets processed immediately

3)Connection Termination

TCP Termination also involves three way handshake

1. App1: “I have no more data for you”.
2. App2: “OK, I understand you are done sending.”
3. App2: “OK - Now I’m also done sending data”.

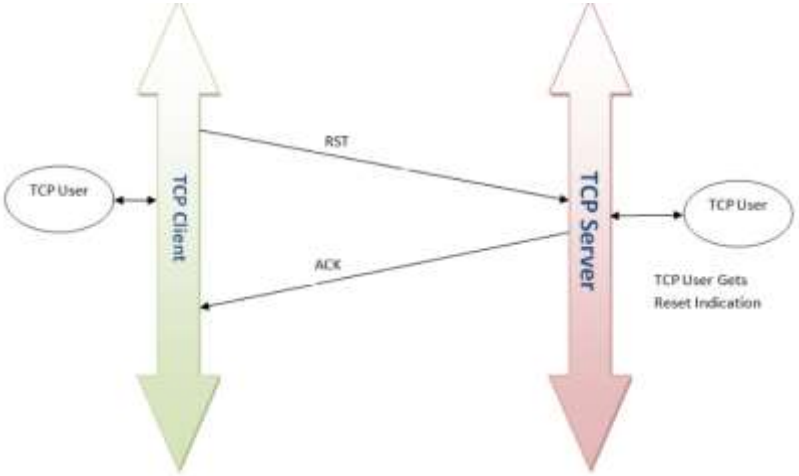
3)Connection Termination



- The TCP layer can send a RST segment that terminates a connection if something is wrong.
- Usually the application tells TCP to terminate the connection politely with a FIN segment.
- Either end of the connection can initiate termination.
- A FIN is sent, which means the application is done sending data.
- The FIN is ACK’d.
- The other end must now send a FIN.
- That FIN must be ACK’d.

RST connection

- For example, a TCP end receives a packet for which there is no connection. The receiving side will send a TCP RST to the remote, to close the connection and again set up if required. The other ends send the TCP RST Ack.



Flow Control

Flow Control

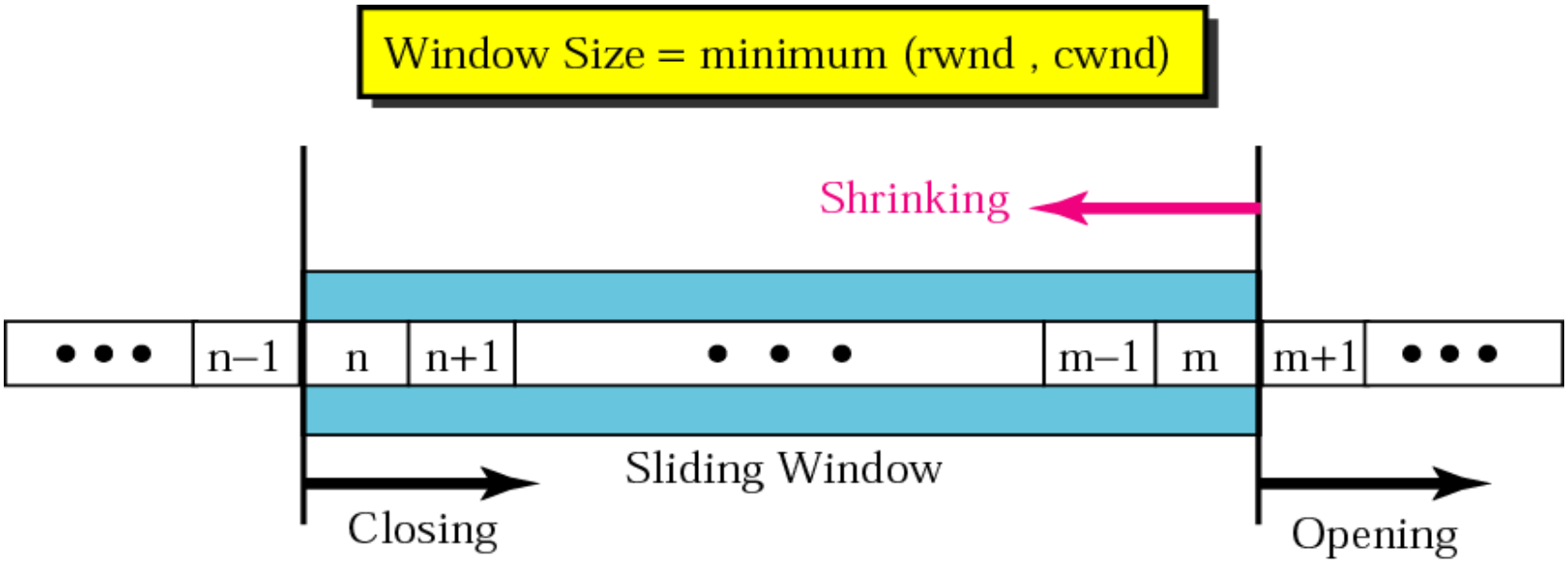
- Flow Control is the mechanism of adjusting the sending rate according to the resources available at the destination.
- During the TCP connection establishment process, the source and destination learn about the resources (i.e., the buffer space) that each side can allocate for the connection and then periodically update the available buffer space through the 'Advertised Window Size' field in the TCP header of the Acknowledgment and data packets.
- The Sliding Window algorithm is used to dynamically adjust the number of outstanding packets (packets that have been sent but not yet acknowledged).
- **Classic TCP:** Acknowledgments are sent only for the bytes that have arrived in-order so far. The application at the receiver side can read only the bytes received in-order so far.
- The bytes received out-of-order are simply buffered at the receiver side. When the missing bytes come, a cumulative ACK indicating the sequence number of the last byte received in-order is sent.

Flow control

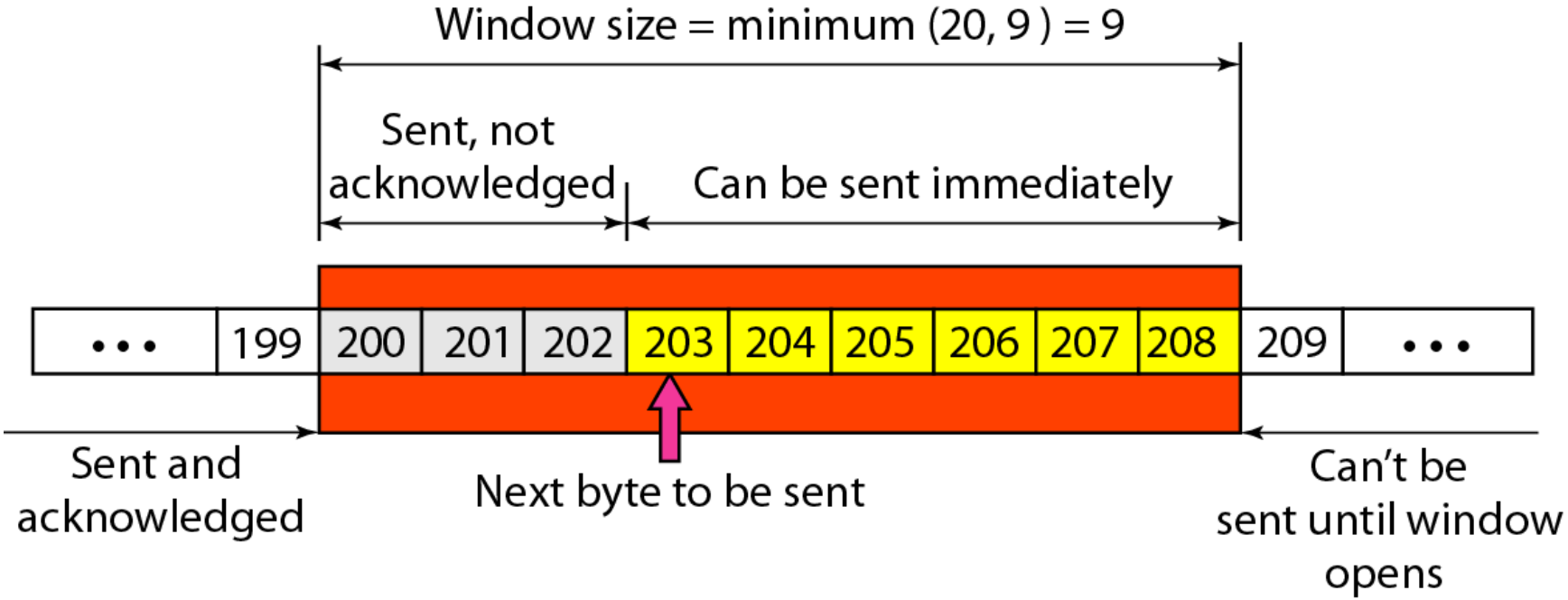
- **A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data.**
- **TCP sliding windows are byte-oriented.**
- **Sliding window in-between go back N and selective repeat sliding window**
- **Go back N –does not use NAK**
- **Selective repeat – holds the out of order frames till it receives the missing ones**
- **The bytes inside the window can be sent without worrying for acknowledgement**
- **It has two wall left and right**

- The window is opened,closed or shrunk
- These three activities are in the control of the receiver and not the sender
- Opening the window means moving the right wall to the right
- Allows more bytes in the buffer that are eligible for sending
- Closing moving left wall to right
- This means some bytes have been acknowledged
- Shrinking means moving the right wall to the left
- Not used-revoking the eligibility of the bytes that can be sent

- The size of the window is the lesser of rwnd and cwnd.
- Rwnd is advertised by the sending site
- Cwnd Is determined by the network to avoid congestion



Example


Window size = minimum (20, 9) = 9
Sent, not acknowledged
Can be sent immediately
• • •
199
200
201
202
203
204
205
206
207
208
209
• • •
Sent and acknowledged
Next byte to be sent
Can't be sent until window opens

Error Control

Error control

TCP provides reliability using error control, which detects corrupted, lost, out-of-order, and duplicated segments.

Error control in TCP is achieved through the use of the

1) **Checksum**
   -Segments include checksum field,to check for corrupted segment
   -corrupted segments is discarded by destination tcp and is considered lost

2)Acknowledgment
   -uses ack to confirm the receipt of data segments
   - no NACKs

3) time-out
   – one time-out counter for each segment sent

4)Retransmisison

Retransmission

- Segment is retransmitted
  a)Retransmission timer expires
  b)Sender receives three duplicat ACK’s

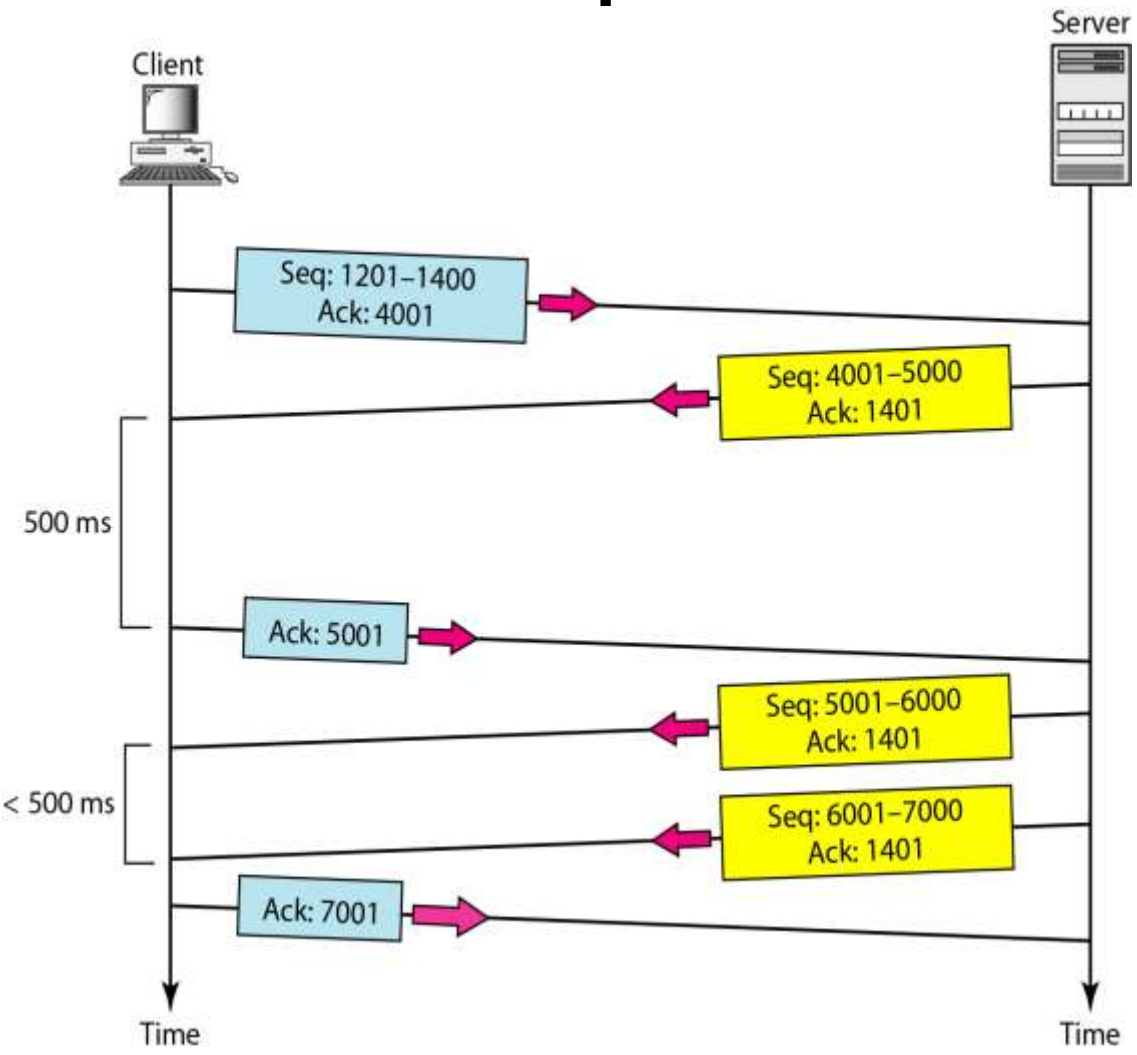
a)Retransmission timer expires

- One retransmission time out (RTO)timer for all outstanding (sent but not acknowledged) segments
- Timer is out, earliest outstanding segment is retransmitted
- RTO is dynamic and is based on RTT(round trip time) of segments

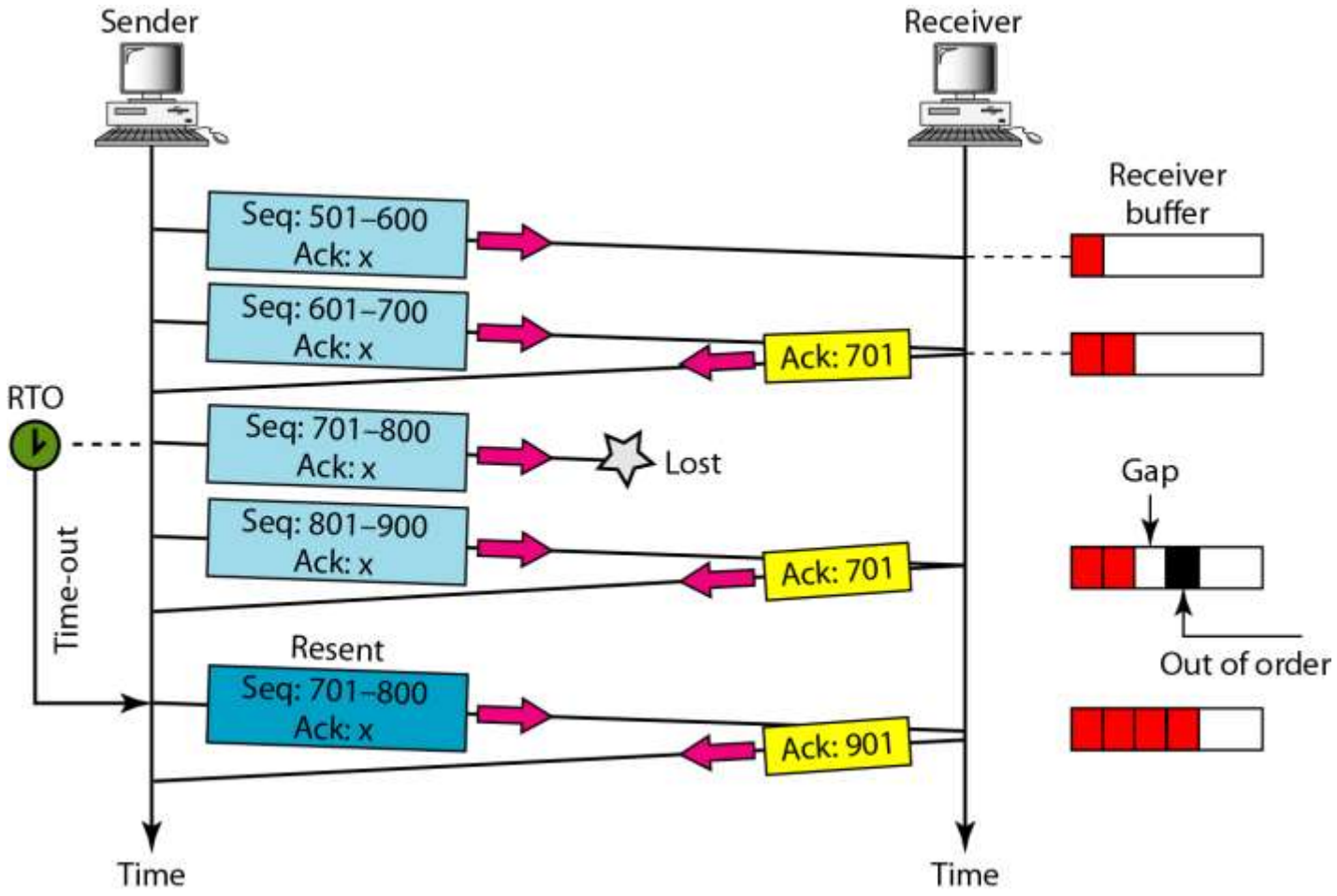
b)Retransmission after 3 duplicate ACK segments

- RTO is good if its value is not large
- If suppose one segment is lost
- Receiver receive many out of order segments
- Buffer exceeds
- So it can send 3 duplicate acks
- **Which will send the missing frame immediately –fast transmission**

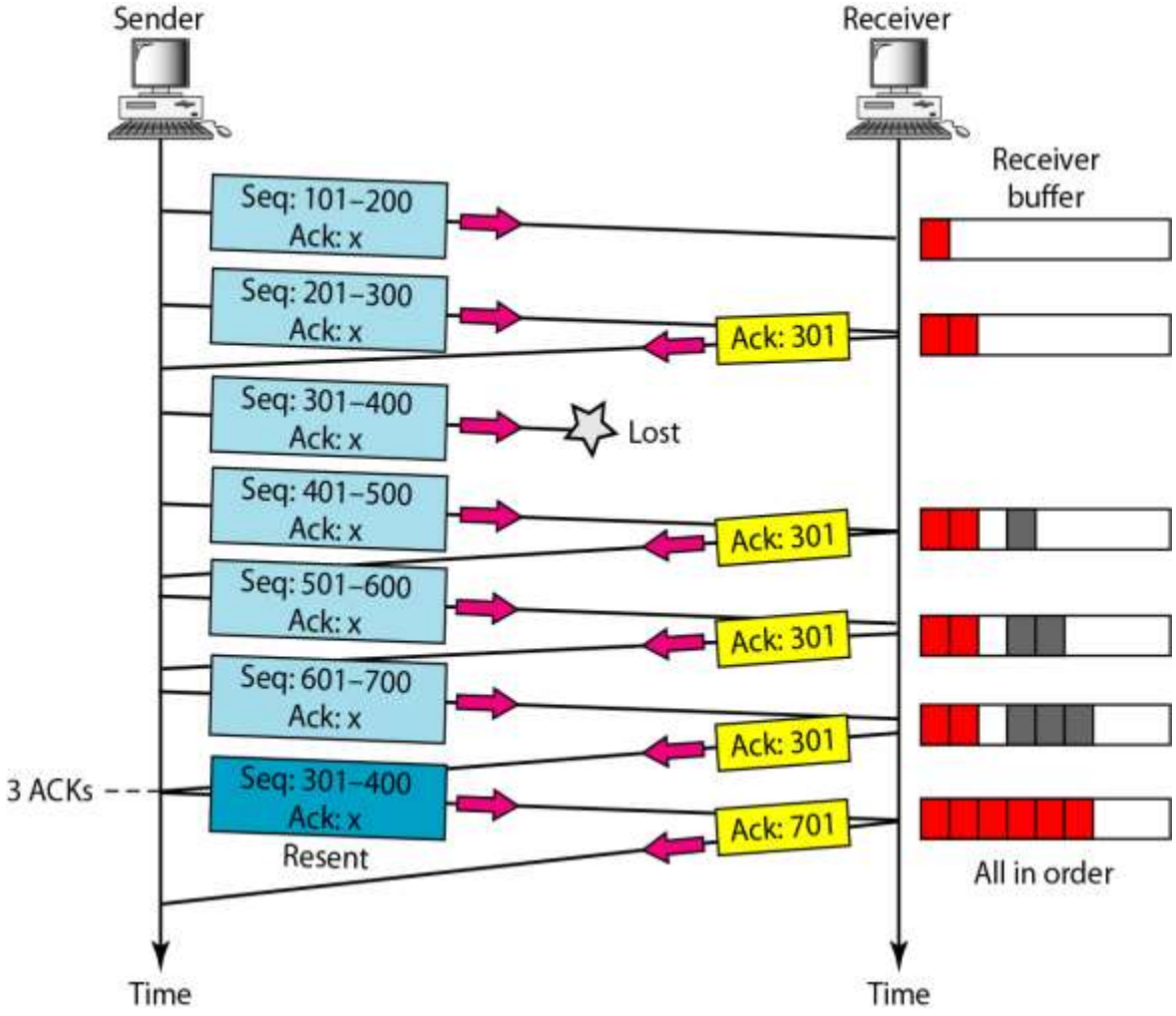
Normal operation



Lost Segment



Fast Retransmission



Congestion Control

Congestion Control

- Congestion Control is the mechanism of adjusting the sending rate according to the resources (i.e., bandwidth and router queue size) available in the intermediate networks.

- Congestion Control is heavily dependent on the 'Timeout' value set at the source in order to decide about retransmitting a data packet that has not been acknowledged yet.
- As the Round-trip-time (RTT) between a source and destination across the Internet dynamically changes, estimating a proper RTT is key to setting the appropriate Timeout value to avoid unnecessary retransmissions and at the same time effectively utilize the channel bandwidth.

- The effective window (i.e., the amount of data the sender can send to the receiver satisfying the conditions of flow control and congestion control) is MIN(CongestionWindow, AdvertisedWindow)

TCP congestion control: high-level idea

- End hosts adjust sending rate
- Based on implicit feedback from the network
  - Implicit: router drops packets because its buffer overflows, not because it's trying to send message
- Hosts probe network to test level of congestion
  - Speed up when no congestion (i.e., no packet drops)
  - Slow down when when congestion (i.e., packet drops)
- How to do this efficiently?
  - Extend TCP's existing window-based protocol…
  - Adapt the window size based in response to congestion

All These Windows…

- **Flow control window**: Advertised Window (RWND)
  - How many bytes can be sent without overflowing receivers buffers
  - Determined by the receiver and reported to the sender
- **Congestion Window** (CWND)
  - How many bytes can be sent without overflowing routers
  - Computed by the sender using congestion control algorithm
- **Sender-side window** = minimum{CWND,RWND}
  - Assume for this lecture that RWND >> CWND

Congestion control in TCP

- **TCP uses congestion window and congestion policy to**
  - **Avoid congestion**
  - **Detect and remove congestion after it occurred**

- Idea of congestion control: As packet losses are rarely to occur due to hardware errors/ transmission errors, a **packet loss is considered by the sender as a sign of congestion** in the network and hence it begins to slow down.
- **Retransmission occurs:**
  - **If timeout occurs – strong possibility of congestion**
  - **3 dup ACK received – weak possibility of congestion**

- Congestion policy
  - is composed of three phases

  a) slow start

  b) congestion avoidance

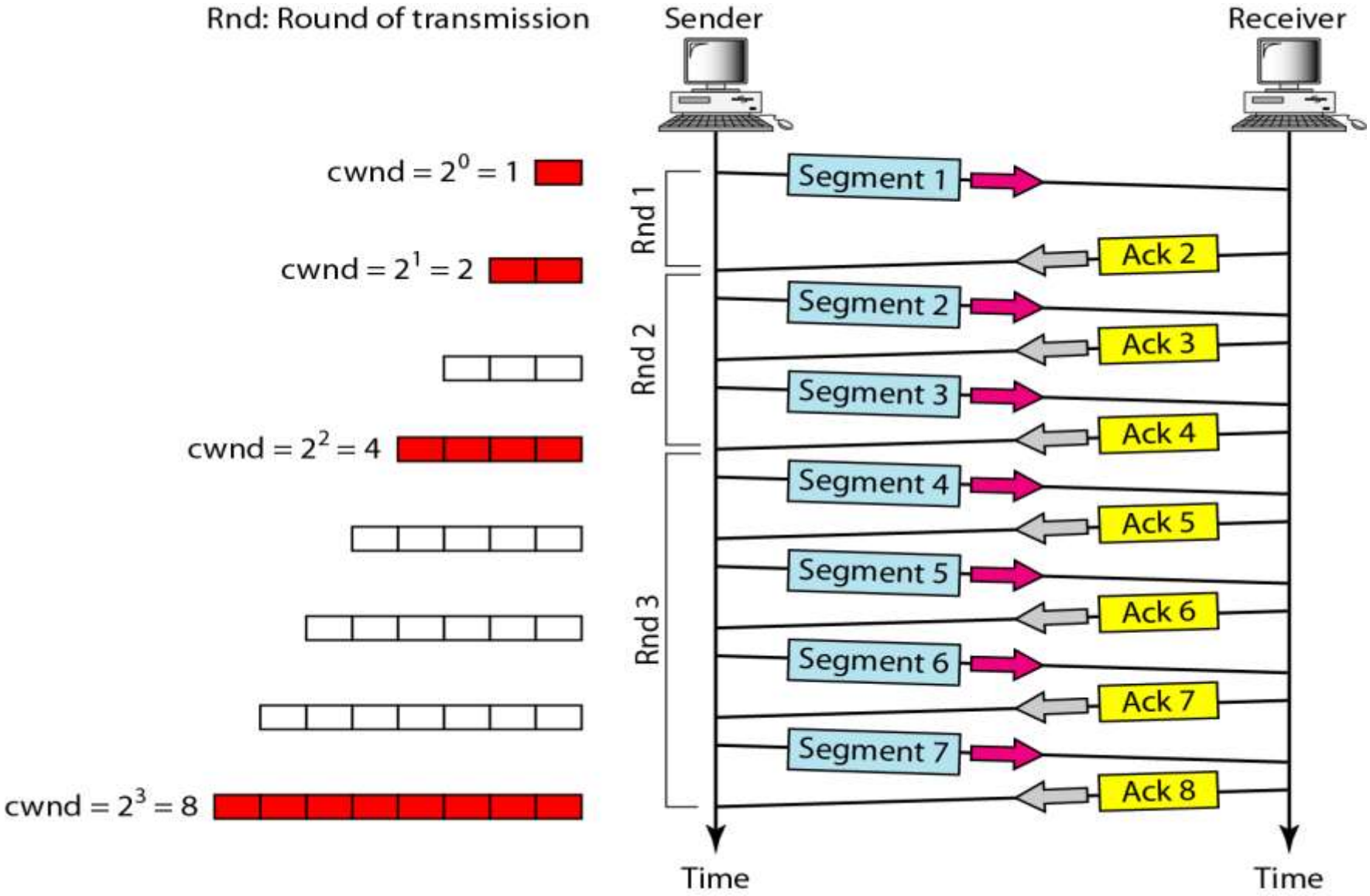
  c) congestion detection.

  These three phases determine the congestion window size.

a)Slow Start (exponential increase)

- Initially, the sender does not know the congestion window. So, it starts very conservatively sending only one segment per RTT (i.e., congestion window = 1 segment)
- The algorithm starts the congestion window with one transmission unit, called MSS or maximum segment size.
- That could be pretty wasteful
- Might be much less than the actual bandwidth
- Linear increase takes a long time to accelerate
- Slow-start phase (actually “fast start”)
- Sender starts at a slow rate (hence the name)
- • … but increases exponentially until first loss

Example-slow start

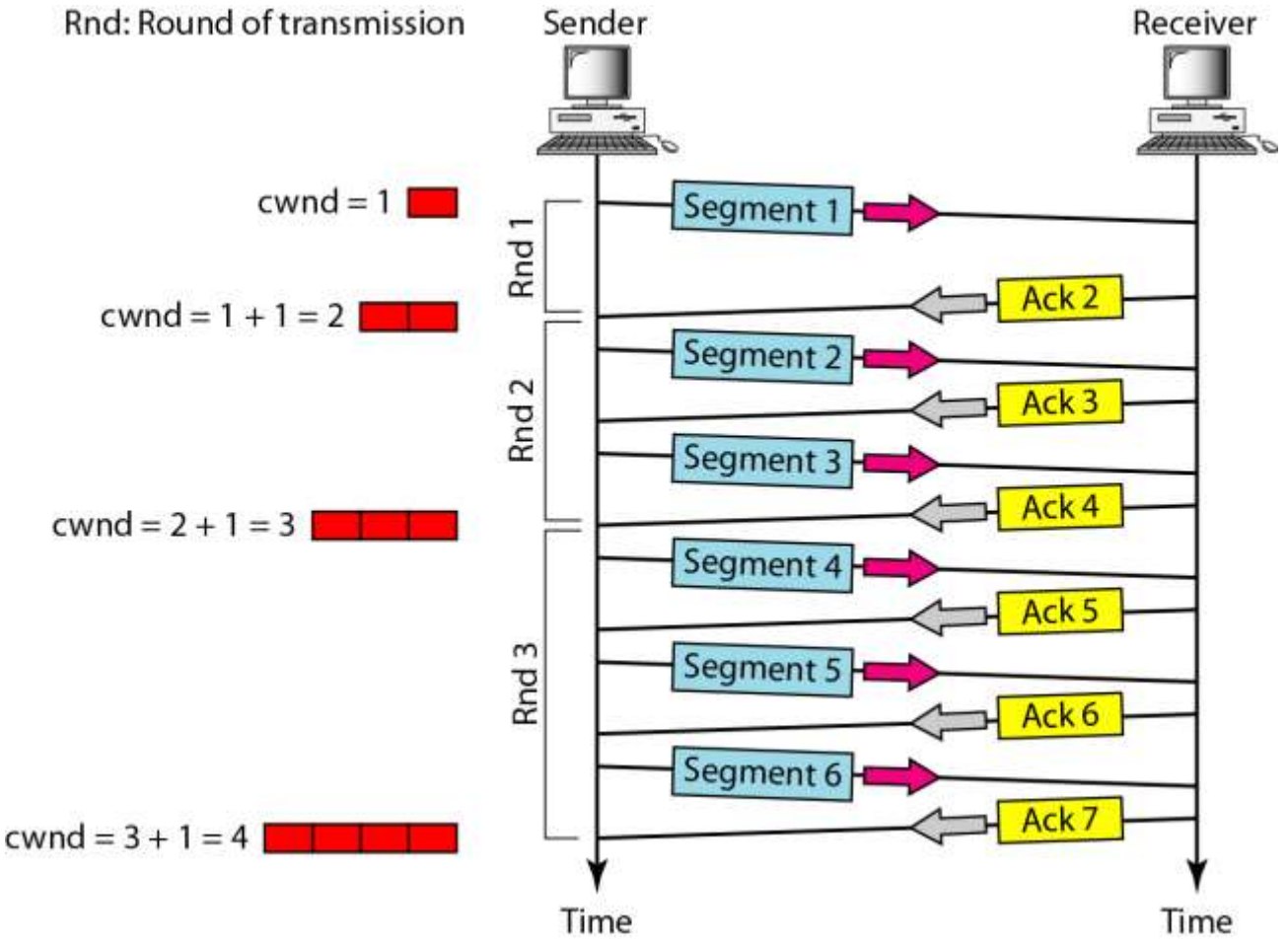


1. To understand the slow start phase, assume segment number instead of byte number (seq number), the sender window size is always equal to congestion window (rwnd is large), and each segment is acknowledged individually.
2. The sender start with cwnd =1 MSS. The sender only sends one segment
3. The cwnd increases by one MSS every time an ACK is received.
4. If the ACKs are received with delay, cwnd increase at the rate less than the power of two for the next RTT.
5. There is **a threshold of 65,535 bytes to stop this phase**.

b)Congestion Avoidance (additive increase)

- We must slow down the exponential growth in the slow start phase to avoid congestion before it happens.
- When the size of the c**ongestion window reaches the slow-start threshold**, the slow start stops and the additive phase (congestion avoidance) begins.
- In this algorithm, each time the whole window of segments is acknowledged (one round), the size of cwnd increases by 1.
- The cwnd increases until congestion is detected.

Example-congestion avoidance



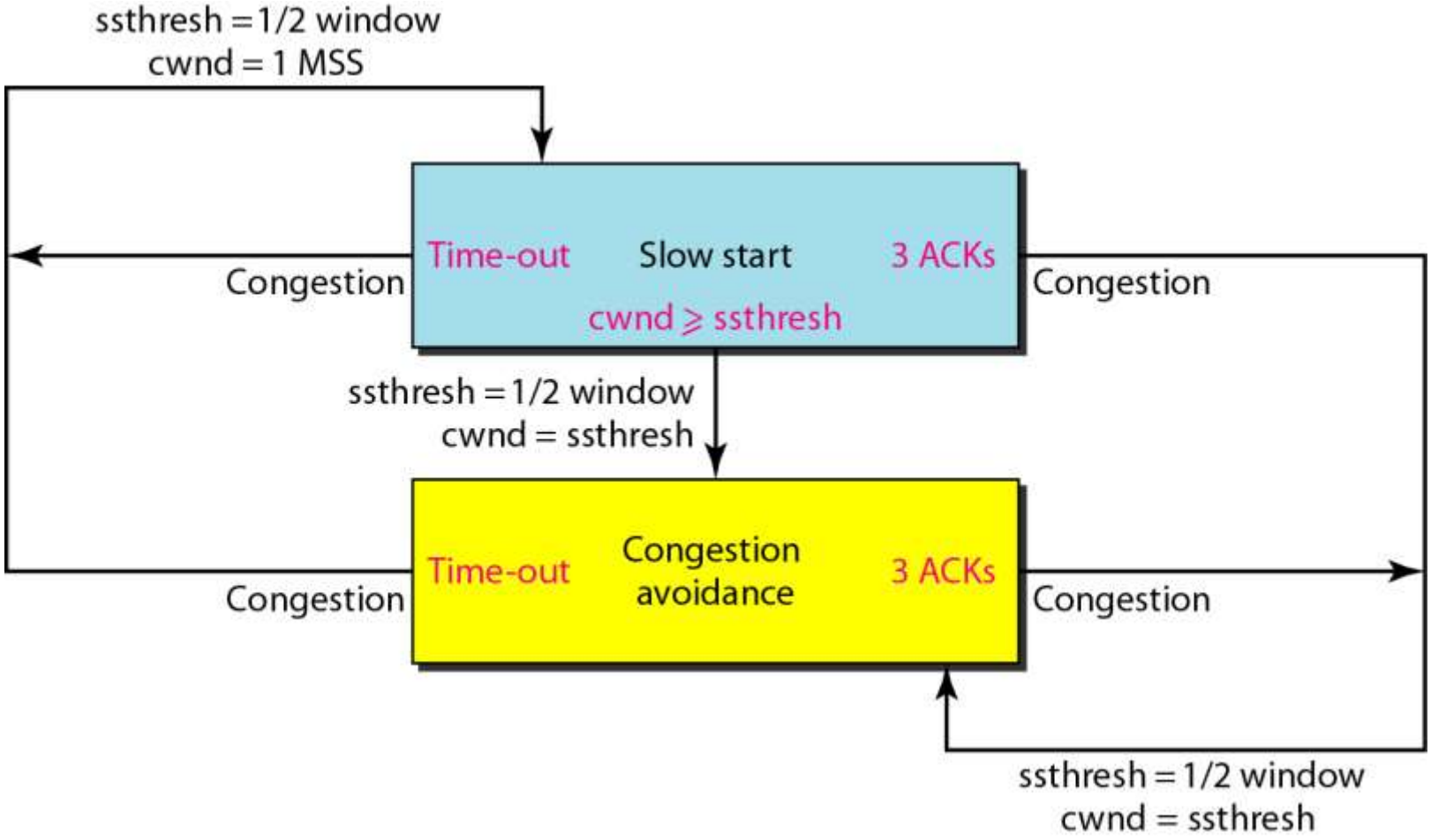
c)Congestion Detection (multiplicative decrease)

- When congestion occurs, cwnd must be decreased.
- Usually, the sender guesses congestion by the need to retransmit a segment; when timer expires or when three ACKs are received.
- If **time out**, there is a strong possibility of congestion, TCP reacts strongly
- It sets the value of threshold to one-half of the current window size.
- It resets the cwnd to the size of one segments (one MSS).
- It starts the slow start phase again.

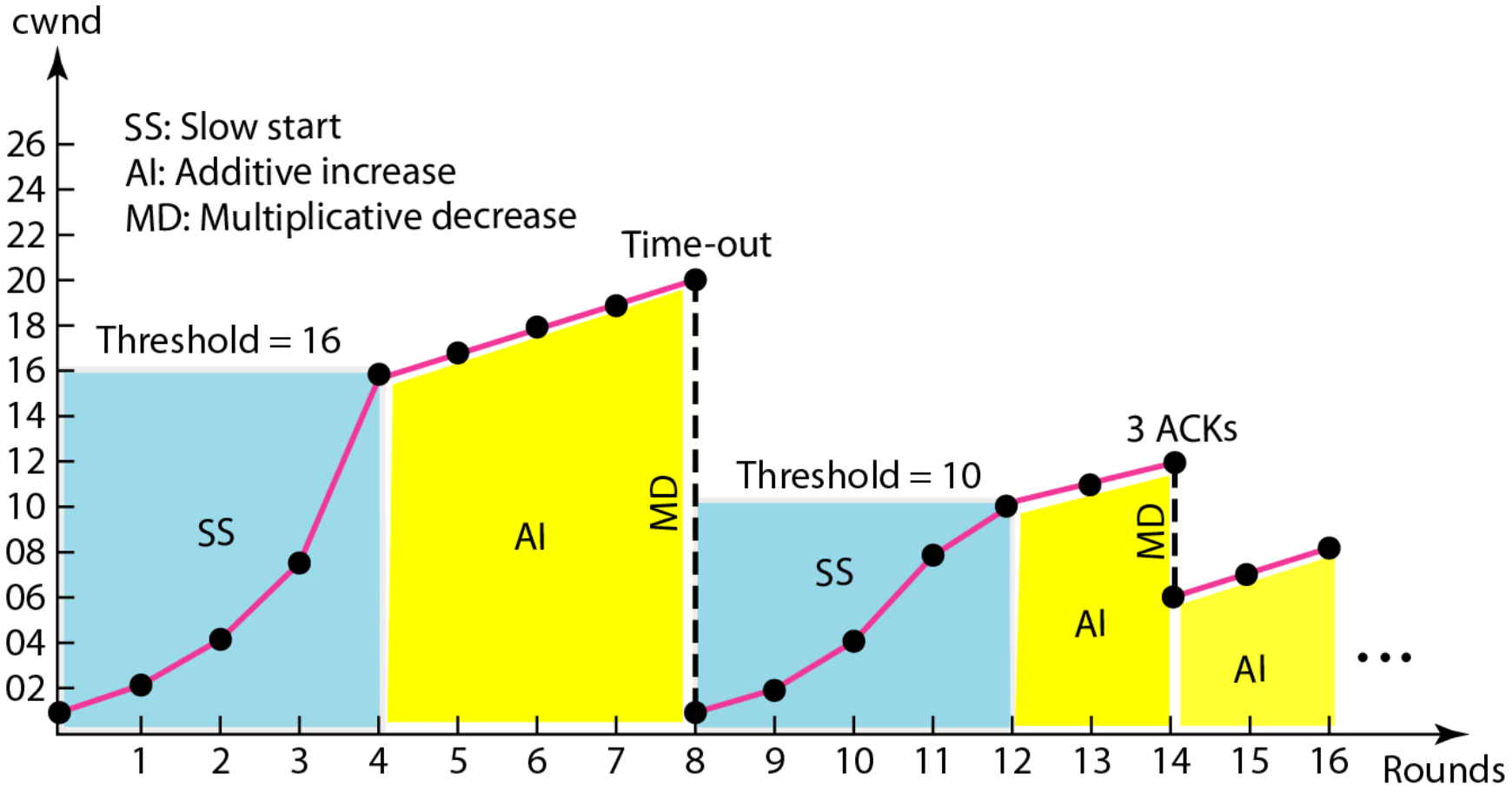
Congestion detection

- If three ACKs are received, TCP needs to retransmit a specific packet (fast retransmission or fast recovery).
- And, it assume there is a weak possibility of congestion.
- TCP has a weaker reaction.
  - It sets the value of threshold to one-half of the current window size.
  - It sets the cwnd to the value of the threshold.
  - It starts the congestion avoidance phase.

TCP congestion control policy - summary



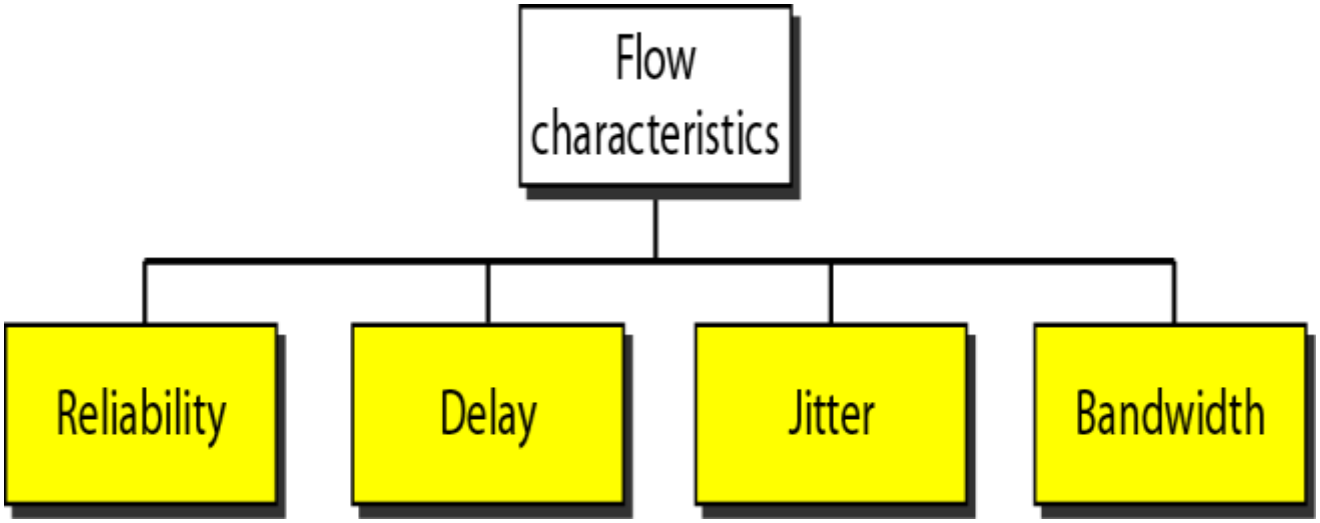
Congestion example



QUALITY OF SERVICE

Quality of service (QoS) is an internetworking issue. We can informally define quality of service as something a flow seeks to attain.

Flow characterestics



Application	*Reliability*	*Delay*	*Jitter*	*Bandwidth*
FTP	High	Low	Low	Medium
HTTP	High	Medium	Low	Medium
Audio-on-demand	Low	Low	High	Medium
Video-on-demand	Low	Low	High	High
Voice over IP	Low	High	High	Low
Video over IP	Low	High	High	High

- Reliability: Lack of reliability means losing a packet or acknowledgement, which entails retransmission. Different application programs need different levels of reliability.
- Delay: Source-to-destination delay. Delay tolerance varies between applications.
- Jitter: Variation in delay for packets belonging to the same flow. Real-time audio and video applications cannot tolerate high jitter.
- Bandwidth: bits per second
- Flow classes: Depend on flow characteristics, we can classify flow into groups e.g., CBR, UBR, etc.

TECHNIQUES TO IMPROVE QoS

Techniques that can be used to improve the quality of service.

- Scheduling
  - FIFO Queuing
  - Priority Queuing
  - Weighted Fair Queuing
- Traffic Shaping
  - Leaky Bucket
  - Token Bucket
  - Combination of Leaky Bucket and Token Bucket.
- Resource Reservations
  - Integrated Services
  - Differentiated Services
- Admission Control

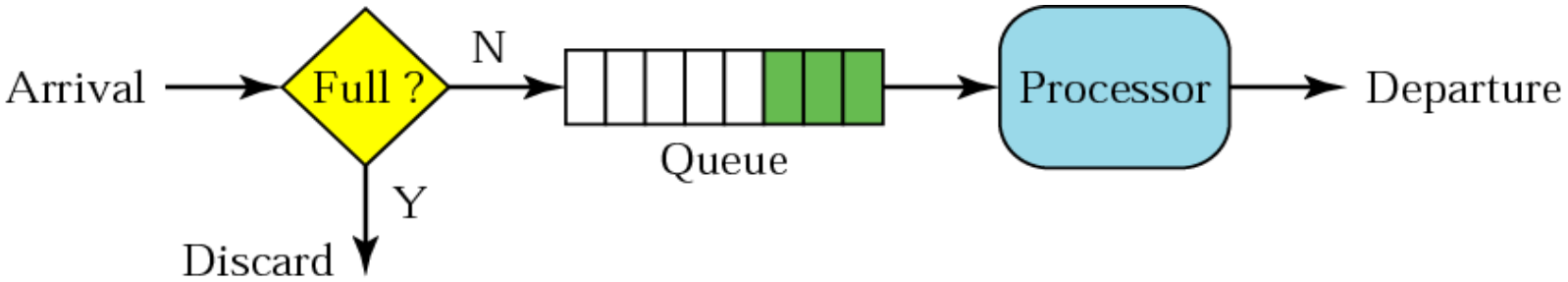
1)scheduling

- Treating packets (datagrams) in the Internet based on their required level of service can mostly happen at the routers.
- Scheduling: The method of processing the flows. A good scheduling technique treats the different flows in a fair and appropriate manner.
- Three Types of Scheduling Algorithms

  a)FIFO Queuing:

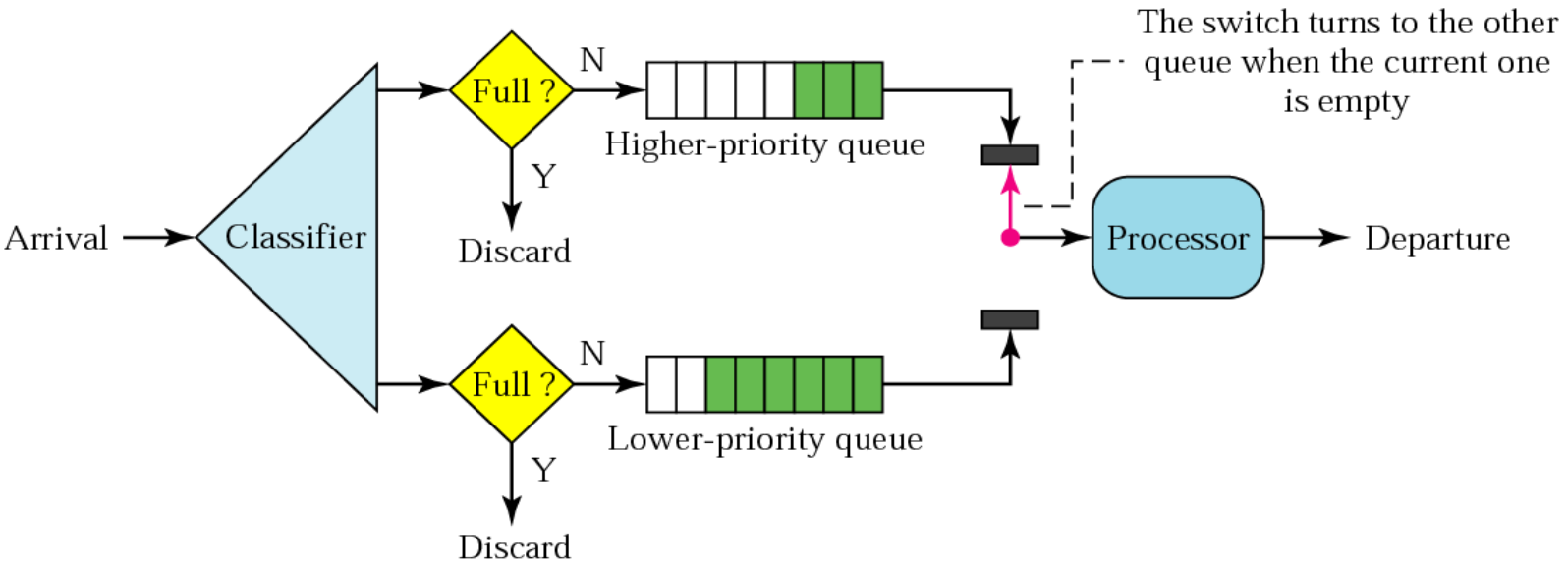
  - First-in first-out queuing
  - Packets wait in a buffer (queue) until the node (router or switch) is ready to process them.
  - If average arrival rate is higher than average processing rate, the queue will fill up and new packets will be discarded.

FIFO Queuing



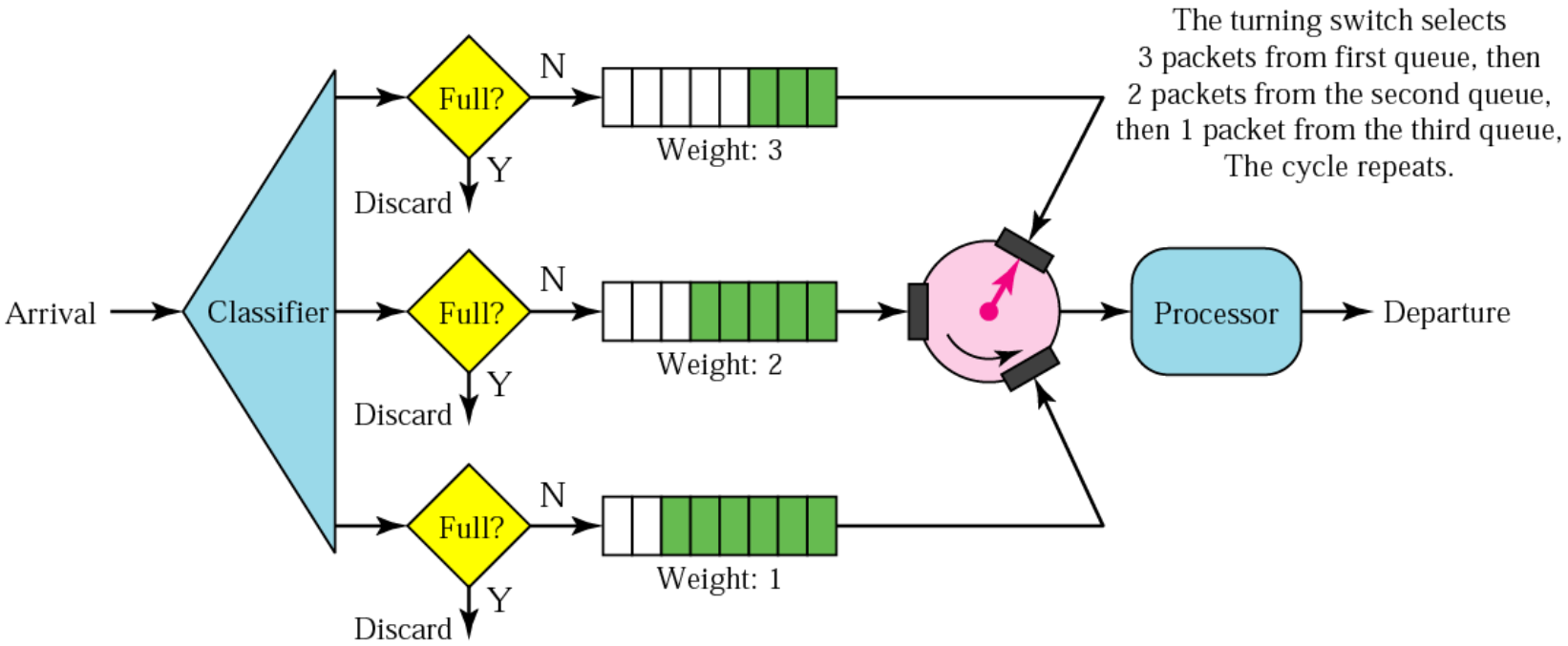
b)Priority Queuing

- Packets are first assigned to a priority class.
- Each priority class has its own queue.
- Packets in highest-priority queue are processed first. Packets in lowest-priority queue are processed last.
- System does not stop serving a queue until it is empty.
- Good for multimedia traffic.
- Starvation is possible: If there is a continuous flow in a high-priority queue, the packets in lower-priority queues will never have a chance to be processed.



c) Weighted Fair Queuing

- Packets are assigned to different classes and admitted to different queues.
- System processes packets in each queue in round-robin fashion with the number of packets selected from each queue based on the corresponding weight.

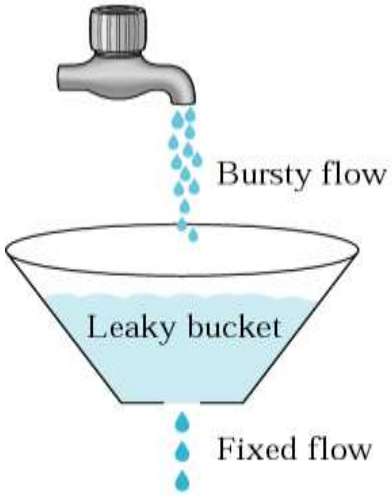


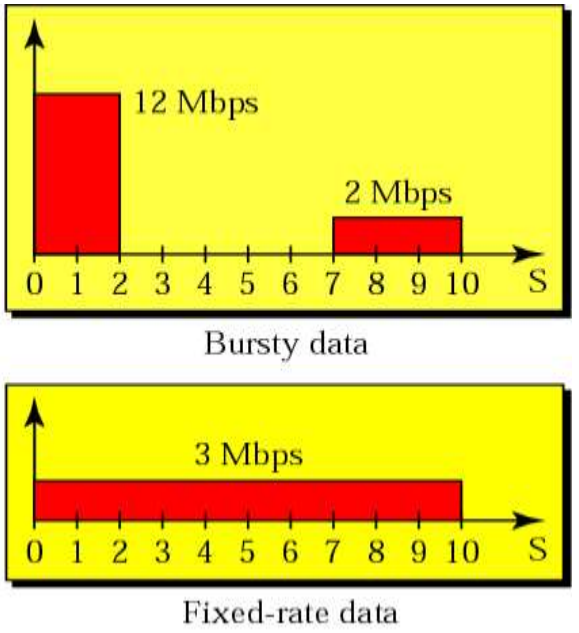
2)Traffic Shaping

- Traffic shaping: Mechanism to control the amount and rate of the traffic sent to the network.
- The first term is used when the traffic leaves a network; the second term is used when the data enters the network
- Two Techniques for Traffic Shaping
  a)Leaky Bucket
  b)Token Bucket

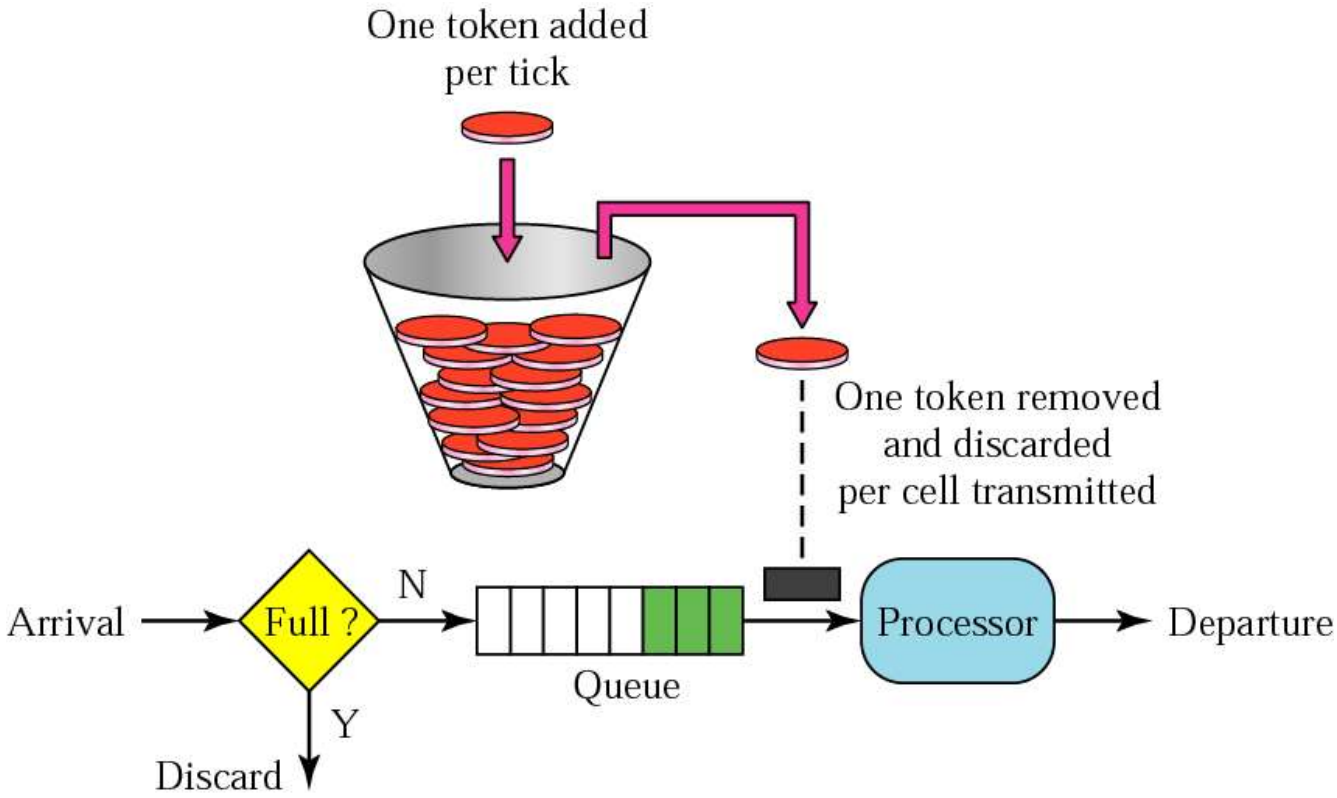
a) Leaky Bucket

- Leaky bucket
  - Input rate varies; Output rate is fixed.
  - Using FIFO queue, if traffic consists of fixed-size packets,the process removes a fixed number of packets from the queue at each tick of the clock. If traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.





Traffic Shaping: Token bucket



Combining Token Bucket and Leaky Bucket:

Leaky bucket is applied after the token bucket; the rate of the leaky bucket to be higher than the rate of tokens dropped in the bucket

3)Resource Reservation

- Resource Reservation:
  - Flow of data needs resources such as a buffer, bandwidth, CPU time, and so on.
  - Resources are reserved beforehand.
    - Integrated Services
    - Differentiated Services
    - The **main difference** between integrated services and differentiated services is that **integrated services involve a prior reservation of resources before the required quality of service is achieved, while differential services mark the packets with priority and send them to the network without prior reservation.**

4)Admission Control

- 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.

End

How a TCP Segment is Created

- `Write()` calls from the applications write data to the TCP sender buffer.
- Sender maintains a dynamic window size based on the flow and congestion control algorithm