Introduction and Transport-Layer Services

- A transport-layer protocol provides for **logical communication** between application processes running on different hosts.
- Transport-layer protocols are **implemented in the end systems** but not in network routers.
- The transport layer converts the **application-layer messages** into transport-layer packets, known as **transport-layer segments**.
- Network routers act only on the network-layer fields of the **datagram**.
- Different transport layer protocols provides a **different set of transport-layer services** to the invoking application.

Introduction and Transport-Layer Services

Relationship Between Transport and Network Layers

- **Transport-layer** protocol provides logical communication between **processes running on different hosts**, a **network-layer protocol** provides logical communication between **hosts.**
- The services that a transport protocol can provide are often **constrained by the service model** of the underlying network-layer protocol.
- Certain services can be offered by a transport protocol even when the underlying network protocol **doesn't offer the corresponding service** at the network layer.

Introduction and Transport-Layer Services

Overview of the Transport Layer in the Internet

- The Internet, and more generally a TCP/IP network, makes two distinct transport-layer protocols available to the application layer.
- **UDP (User Datagram Protocol)**, which provides an **unreliable**, **connectionless** service to the invoking application.
- **TCP (Transmission Control Protocol)**, which provides a **reliable**, **connection-oriented** service to the invoking application.
- The Internet's network-layer protocol, IP provides logical communication between hosts. The IP service model is a **best-effort delivery service**. IP is said to be an **unreliable service.**

Introduction and Transport-Layer Services

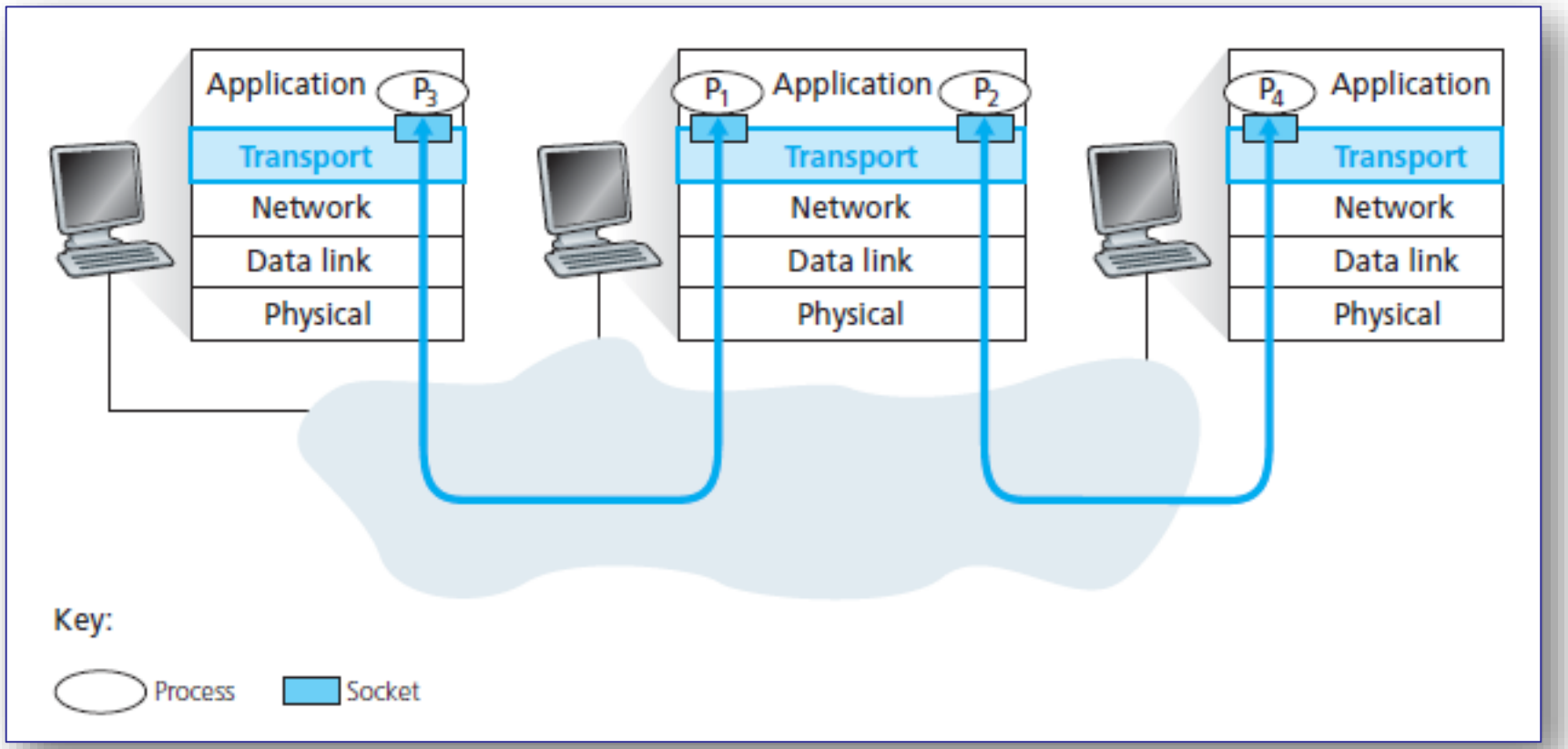
Overview of the Transport Layer in the Internet

- Extends **host-to-host delivery** to **process-to-process delivery,** called transport-layer **multiplexing** and **demultiplexing.**
- Provide **integrity checking** by including error detection fields in their segments' headers.
- These are the only two services that **UDP** provides. like IP, **UDP** is an **unreliable service.**
- TCP converts IP's unreliable service into a **reliable data transport service,** using **flow control**, **sequence numbers**, **acknowledgments** and **timers**.
- TCP also provides **congestion control.**
- TCP traffic is **regulated**, on the other hand, UDP traffic is **unregulated**

Multiplexing and Demultiplexing

- Suppose you are **downloading Web pages** while running **one FTP session** and **two Telnet sessions.**
- When the transport layer in your computer receives data from the network layer below, it needs to **direct the received data to one of these four processes.**
- Transport layer in the receiving host does not actually deliver data directly to a process, but instead to an **intermediary socket.**
- Each socket has a **unique identifier.**
- This job of delivering the data in a **transport-layer segment** to the **correct socket** is called **demultiplexing.**
- The job of **gathering data chunks** at the source host from different sockets, **encapsulating each data chunk** with header information to create segments, and **passing the segments to the network layer** is called **multiplexing.**

Multiplexing and Demultiplexing

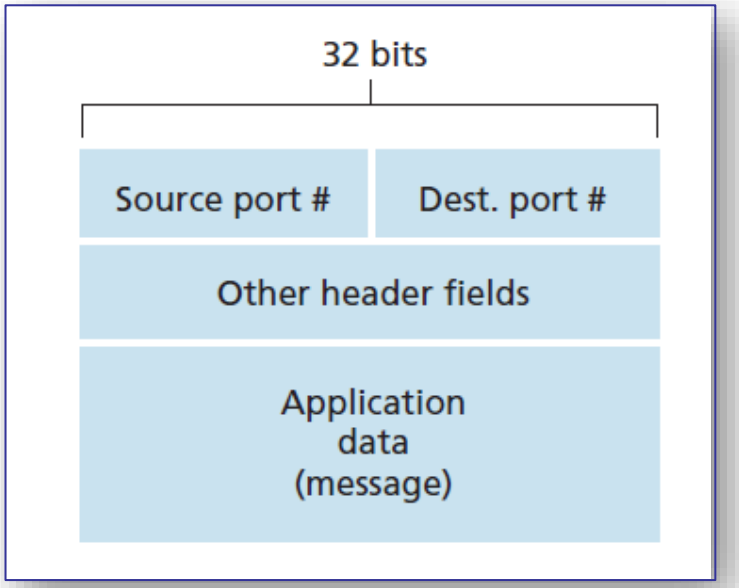


Transport-layer Multiplexing and Demultiplexing

Multiplexing and Demultiplexing

- So, the transport-layer multiplexing requires;
  1. That sockets have unique identifiers
  2. That each segment have special fields that indicate the socket to which the segment is to be delivered.
- These special fields are the **source port number field** and the **destination port number field.**

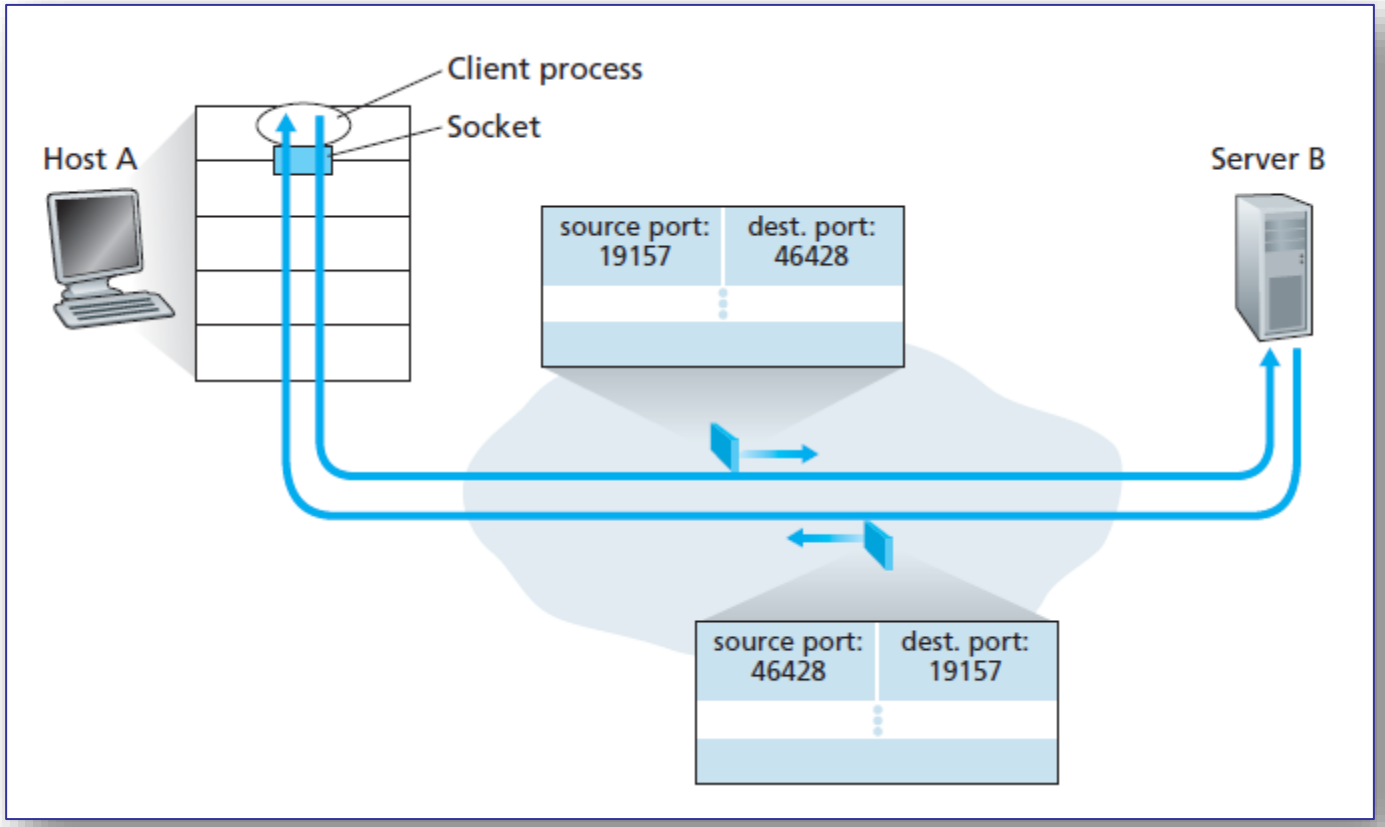
Source and destination port-number fields in a transport-layer segment



Multiplexing and Demultiplexing

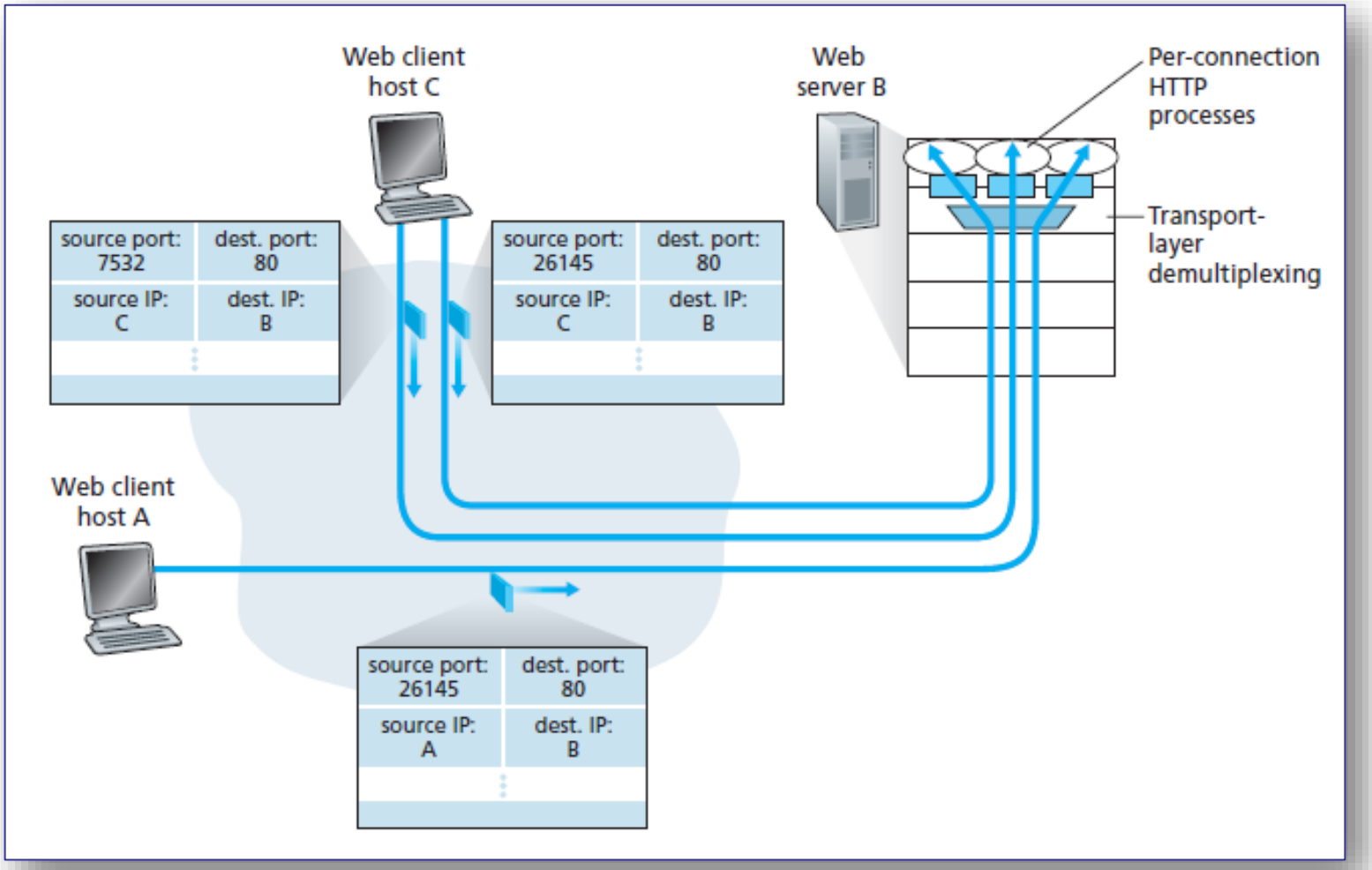
- Each socket in the host could be assigned a **port number.**
- Each port number is a **16-bit number**, ranging from **0 to 65535.**
- The port numbers ranging from **0 to 1023** are called **well-known port numbers** and are **restricted.**
- UDP socket is fully identified by a **two-tuple: Destination IP address** and a **destination port number.**
- TCP socket is identified by a **four-tuple**: **Source IP address**, **Source port number**, **Destination IP address**, **Destination port number**.

Multiplexing and Demultiplexing



The inversion of source and destination port numbers

Multiplexing and Demultiplexing



Two clients, using the same destination port number (80) to communicate with the same Web server application

Connectionless Transport: UDP

- Aside from the multiplexing/demultiplexing function and some light error checking, it **adds nothing to IP.**
- There is no handshaking between sending and receiving transport-layer entities before sending a segment. For this reason, UDP is said to be **connectionless.**
- **DNS** is an example of an application-layer protocol that typically uses UDP.
- Many applications are better suited for UDP due to the following reasons:
  - **Finer application-level control** over what data is sent, and when.
  - **No connection establishment.** DNS runs over UDP while HTTP uses TCP.
  - **No connection state.** A server can typically support many more active clients when the application runs over UDP rather than TCP.
  - **Small packet header overhead.** The TCP segment has 20 bytes of header overhead in every segment, whereas UDP has only 8 bytes of overhead.

Connectionless Transport: UDP

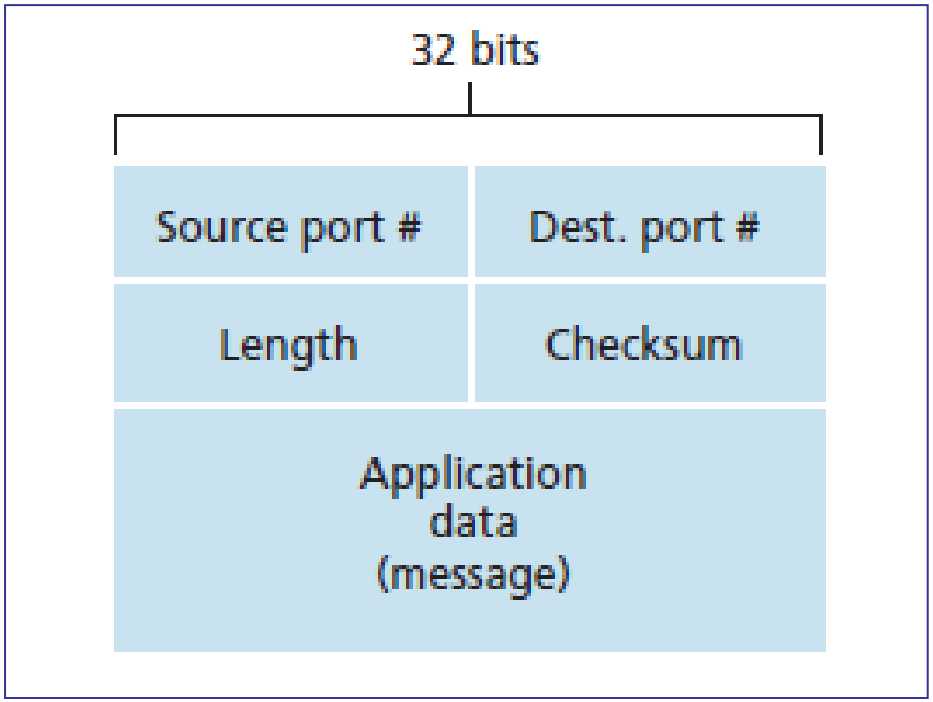
Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
Network management	SNMP	Typically UDP
Routing protocol	RIP	Typically UDP
Name translation	DNS	Typically UDP

Popular Internet applications and their underlying transport protocols

Connectionless Transport: UDP

- Running multimedia applications over **UDP is controversial.**
- It is possible for an application **to have reliable data transfer when using UDP.**

UDP Segment Structure



Connectionless Transport: UDP

UDP Checksum

- The UDP checksum provides **error detection.**
- UDP at the sender side performs the **1s complement** of the sum of all the 1**6-bit words in the segment.**
- Suppose that we have the following three **16-bit words:**
  0110011001100000, 0101010101010101, 1000111100001100
- The sum of first two of these 16-bit words is:

  0110011001100000
  0101010101010101

  1011101110110101

- Adding the third word to the above sum gives

  1011101110110101
  1000111100001100

  0100101011000010

Connectionless Transport: UDP

UDP Checksum

- 1s complement of the sum 0100101011000010 is **1011010100111101**, which becomes the checksum.
- At the receiver, **all four 16-bit words are added, including the checksum.**
- If no errors are introduced into the packet, then clearly the sum at the receiver will be **1111111111111111.**
- UDP must provide error detection at the transport layer, on an **end-end basis.**

Connection-Oriented Transport: TCP

- Internet's transport-layer, **Connection-oriented**, **Reliable** transport protocol.
- Relies on **Error Detection**, **Retransmissions, Cumulative Acknowledgments, Timers,** and header fields for **Sequence** and **Acknowledgment Numbers**.

Connection-Oriented Transport: TCP

The TCP Connection:

- Two processes must first **“handshake”** with each other.
- The TCP “connection” is **not an end-to-end** TDM or FDM circuit as in a circuit switched network nor is it a virtual circuit.
- The intermediate network elements do not maintain **TCP Connection State.**
- A TCP connection provides a **full-duplex** service and **Point-to-point** service.
- Establishment of connection between client process and server process using **three-way handshake.**
- TCP directs this data to the connection’s **send buffer.**

Connection-Oriented Transport: TCP

The TCP Connection:

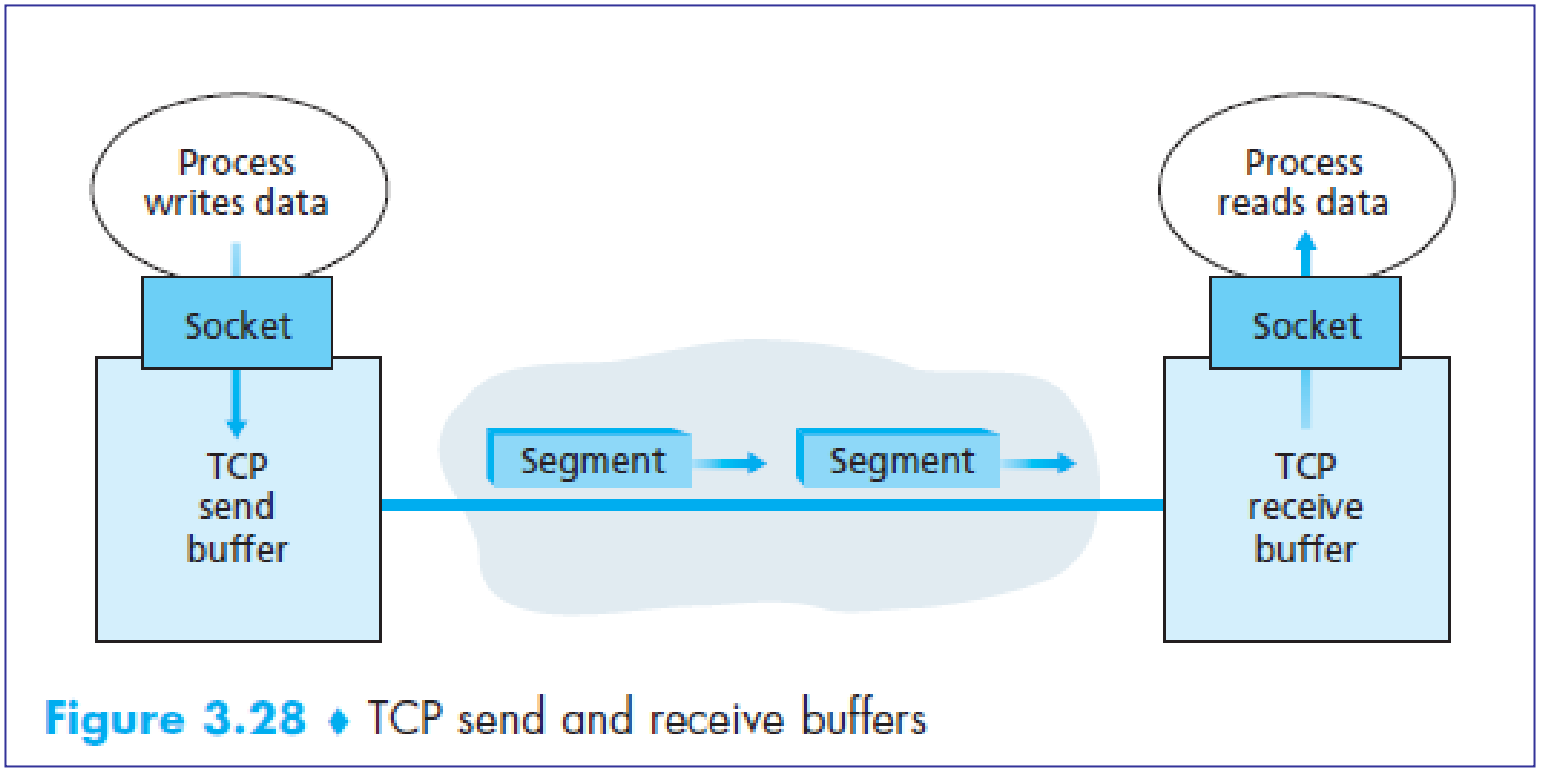


Figure 3.28 ♦ TCP send and receive buffers

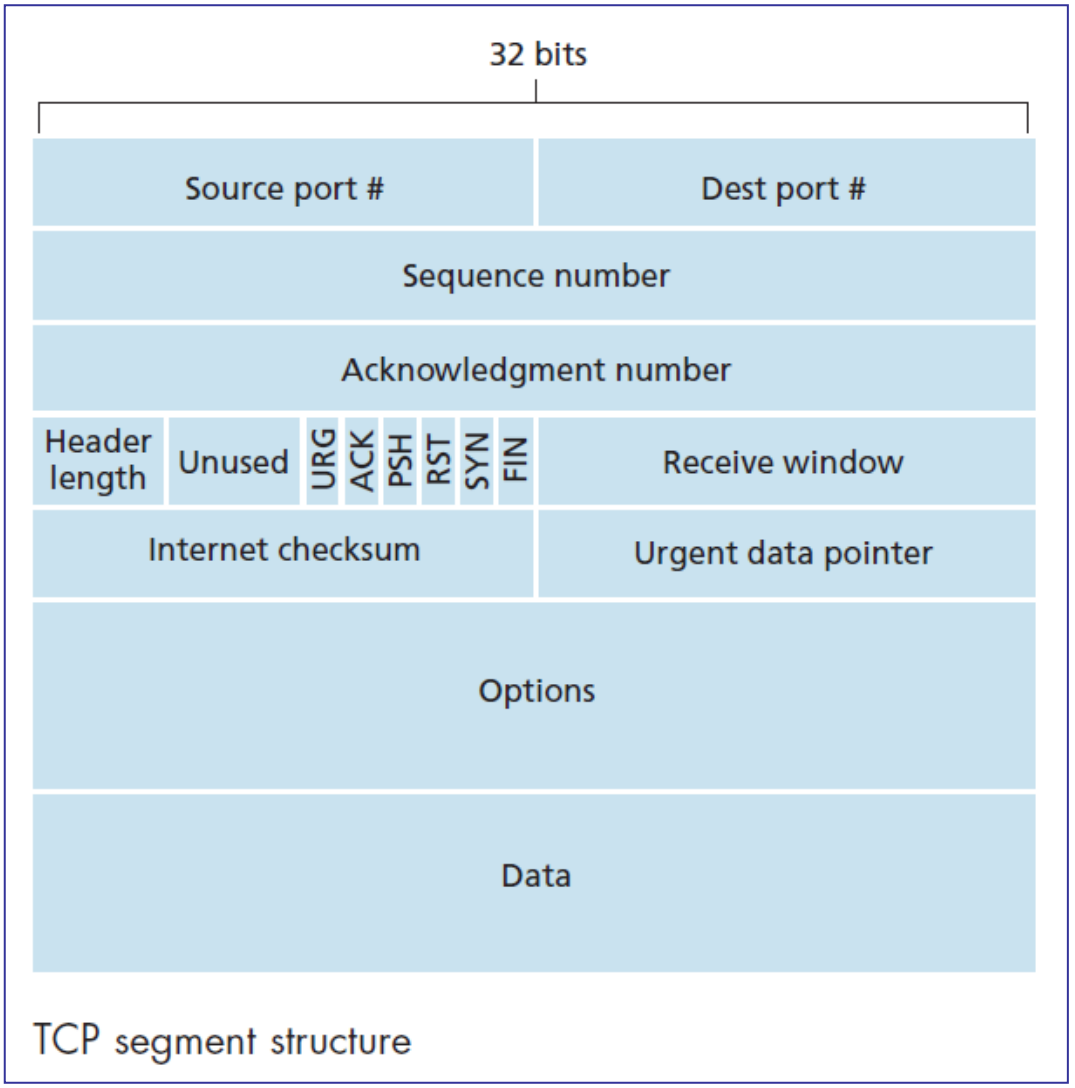
Connection-Oriented Transport: TCP

The TCP Connection:

- The maximum amount of data that can be grabbed and placed in a segment is limited by the **maximum segment size(MSS).**
- MSS is typically set by first determining the length of the **largest link-layer frame.**

Connection-Oriented Transport: TCP

TCP Segment Structure:



TCP segment structure

Connection-Oriented Transport: TCP

TCP Segment Structure:

- The TCP segment consists of **header fields** and a **data field.**
- TCP header is typically of **20 bytes.**
- **Source and Destination Port Numbers**, which are used for multiplexing/demultiplexing data from/to upper-layer applications.
- The **32-bit sequence number** field and the **32-bit acknowledgment number** field are used by the TCP sender and receiver in implementing a reliable data transfer service.
- The **16-bit receive window** field is used for flow control.
- The **4-bit header length** field specifies the length of the TCP header in 32-bit words.

Connection-Oriented Transport: TCP

TCP Segment Structure:

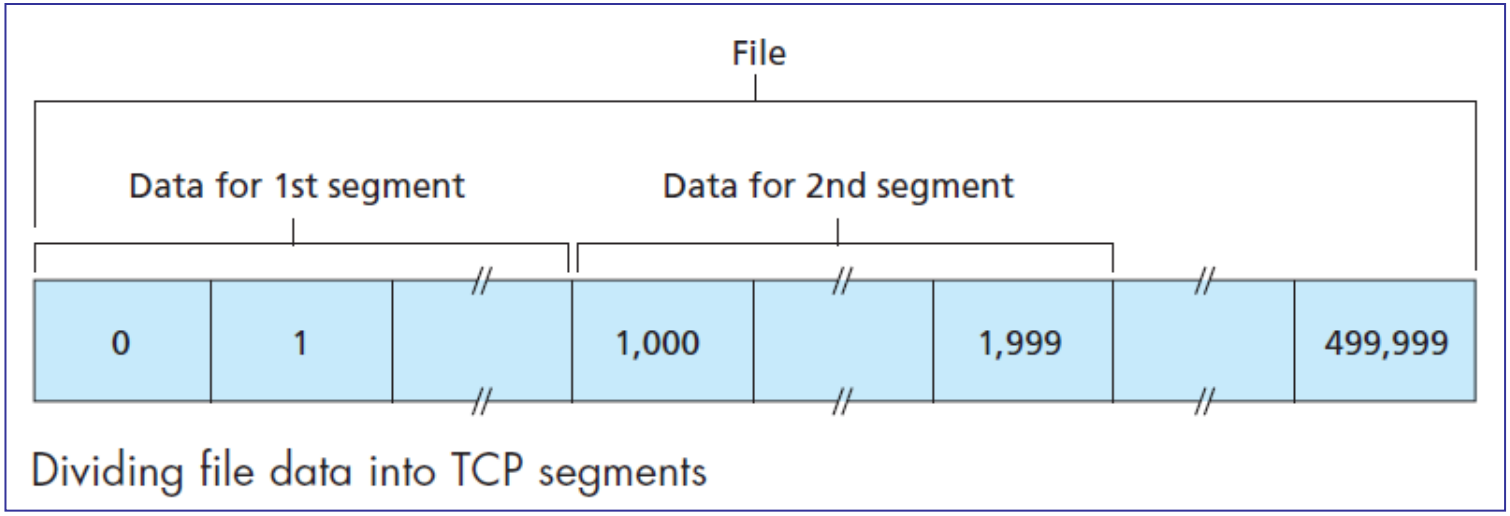
- The **flag field** contains **6 bits.**
  - **ACK:** Indicate that the value carried in the acknowledgment field is valid;
  - **RST, SYN, and FIN:** Used for connection setup and teardown.
  - **PSH:** Indicates that the receiver should pass the data to the upper layer immediately.
  - **URG:** Indicates that there is data in this segment that the sending-side upper-layer entity has marked as “urgent.” The location of the last byte of this urgent data is indicated by the **16-bit urgent data pointer** field.

Connection-Oriented Transport: TCP

TCP Segment Structure:

Sequence Numbers and Acknowledgment Numbers:

- Sequence numbers are over the **stream of transmitted bytes** and not over the series of transmitted segments.
- The sequence number for a segment is therefore the **byte-stream number of the first byte in the segment.**



Dividing file data into TCP segments

Connection-Oriented Transport: TCP

TCP Segment Structure:

Sequence Numbers and Acknowledgment Numbers:

- The **acknowledgment number** that Host A puts in its segment is the sequence number of the next byte **Host A is expecting from Host B.**
- TCP is said to provide **cumulative acknowledgments.**
- TCP connection **randomly choose an initial sequence number.**

Connection-Oriented Transport: TCP

Reliable Data Transfer:

- The data stream that a process reads out of its TCP receive buffer is **uncorrupted**, **without gaps**, **without duplication**, and **in sequence;**
- The recommended TCP timer management procedures use only a single **retransmission timer.**
- **Timeouts** and **Three Duplicate Acknowledgement** based polices to recover from lost segments.

Connection-Oriented Transport: TCP

Reliable Data Transfer:

A Few Interesting Scenarios

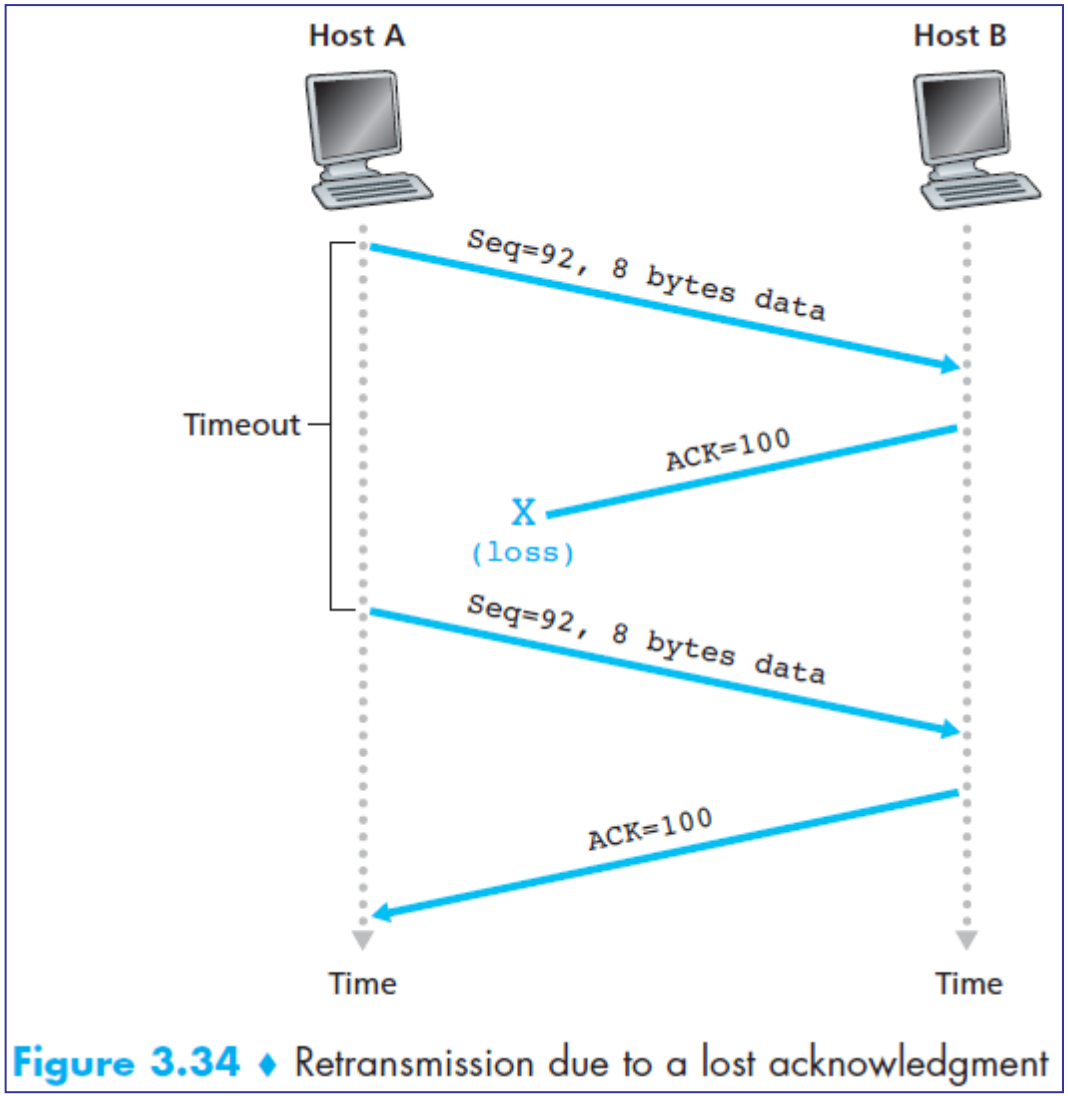


Figure 3.34 ♦ Retransmission due to a lost acknowledgment

Connection-Oriented Transport: TCP

Reliable Data Transfer:

A Few Interesting Scenarios

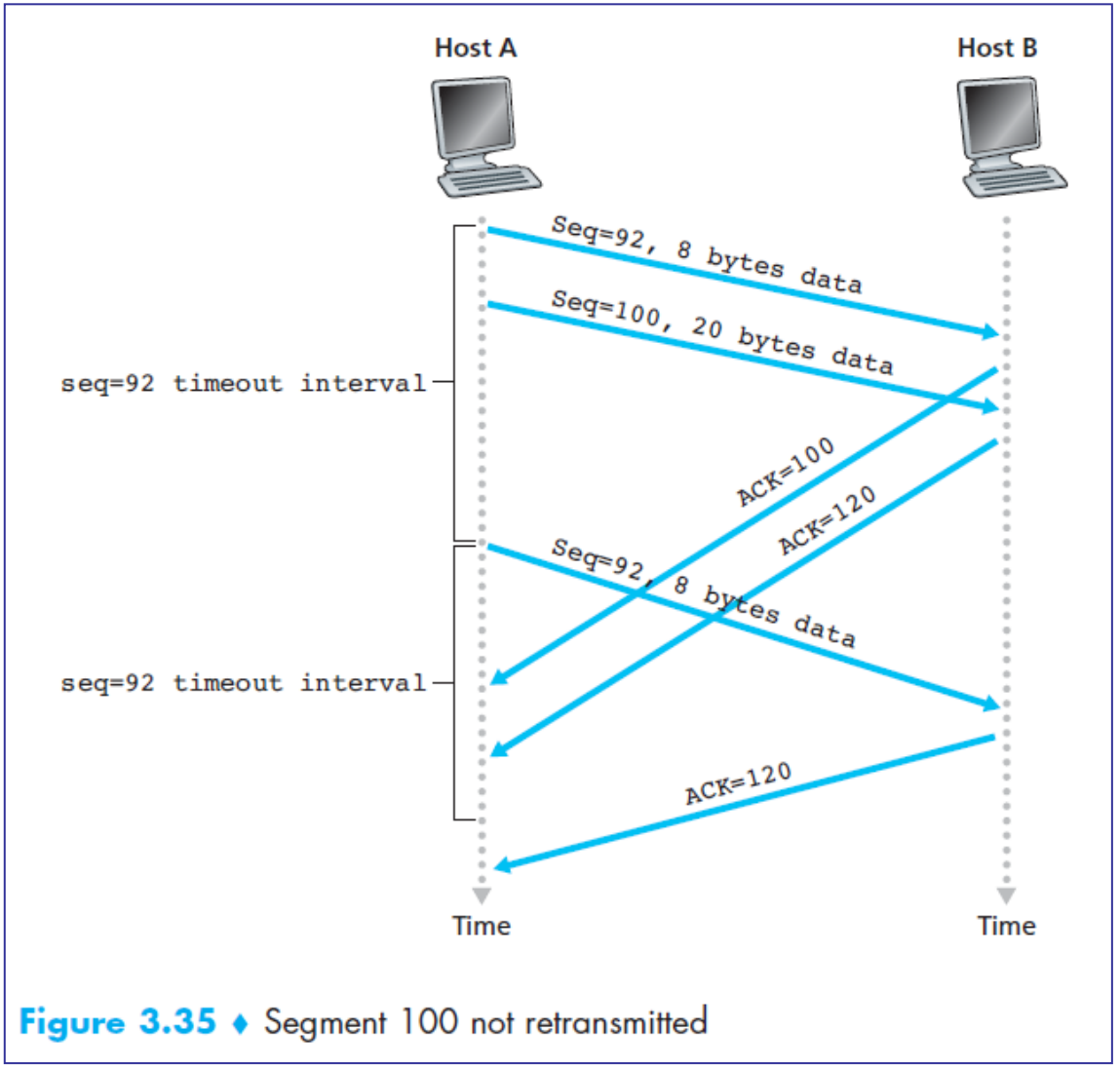


Figure 3.35 ♦ Segment 100 not retransmitted

Connection-Oriented Transport: TCP

Reliable Data Transfer:

A Few Interesting Scenarios

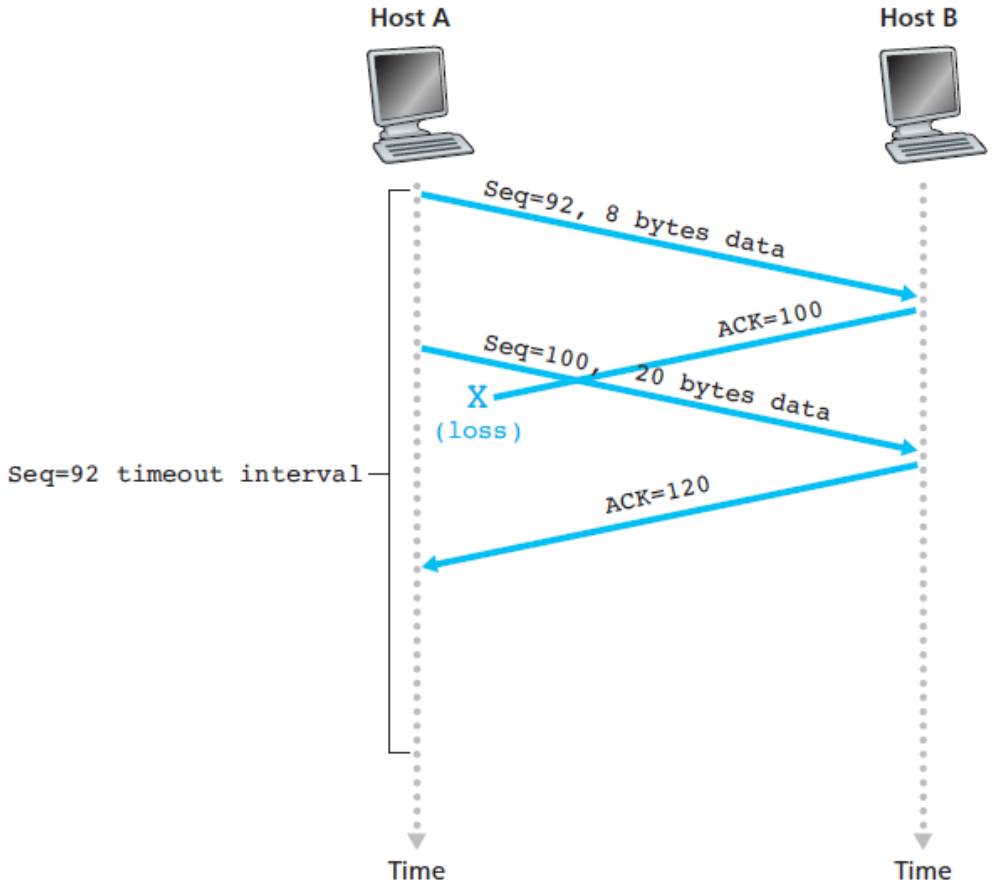


Figure 3.36 ♦ A cumulative acknowledgment avoids retransmission of the first segment

Connection-Oriented Transport: TCP

TCP Connection:

- TCP is connection Oriented.
- It uses the service of IP, but IP is connection less.
- Full Duplex mode.
- Required Three Phase:

    1. **Connection Establishment**
    2. **Data Transfer**
    3. **Connection Termination**

Connection-Oriented Transport: TCP

TCP Connection:

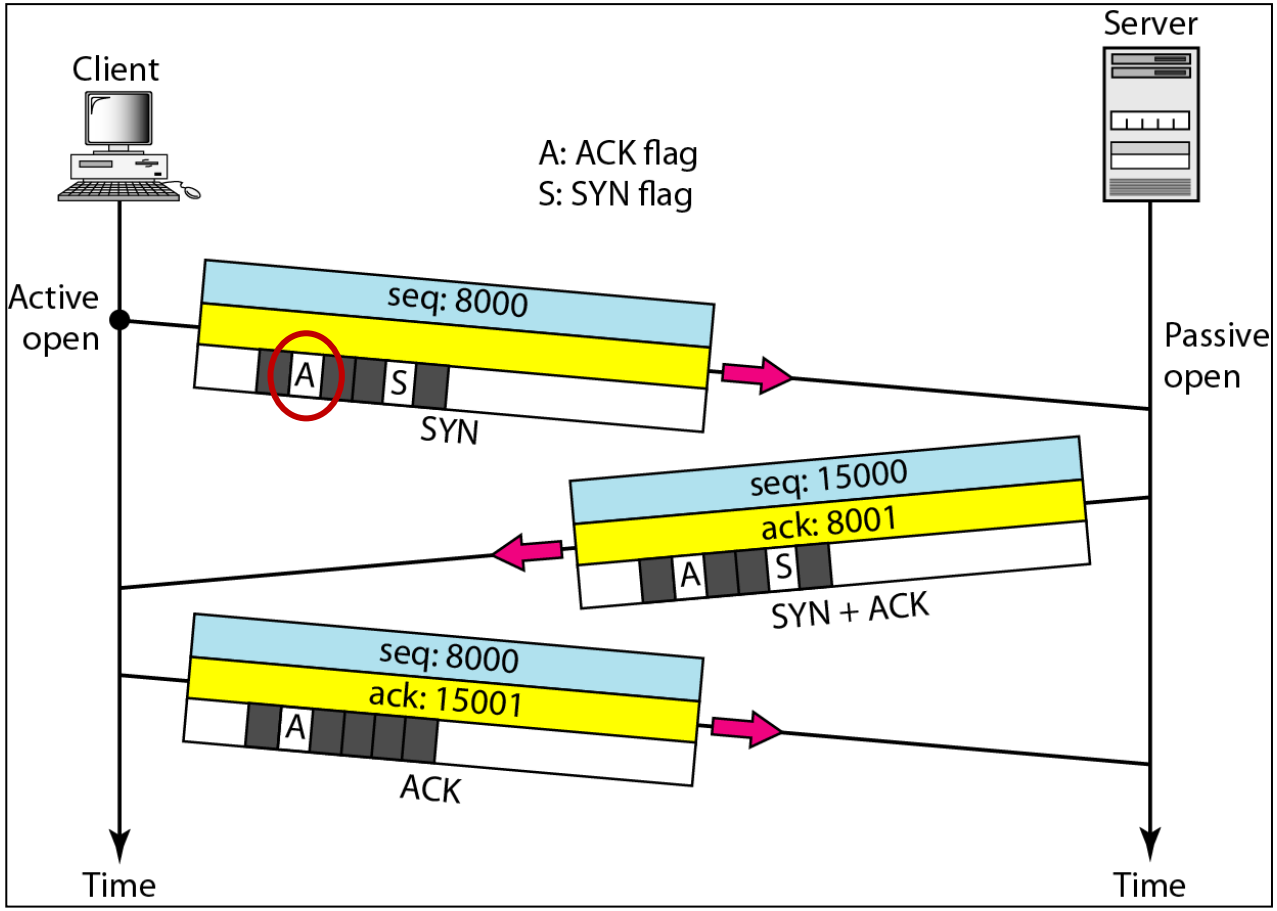
Connection Establishment:

- Three Way Handshaking.
- Process starts with server. The server process tells it's TCP that it is ready to accept the connection, known as **Passive Open.**
- Process on client need to tell TCP on client machine for the connection, known as **Active Open.**

Connection-Oriented Transport: TCP

TCP Connection:

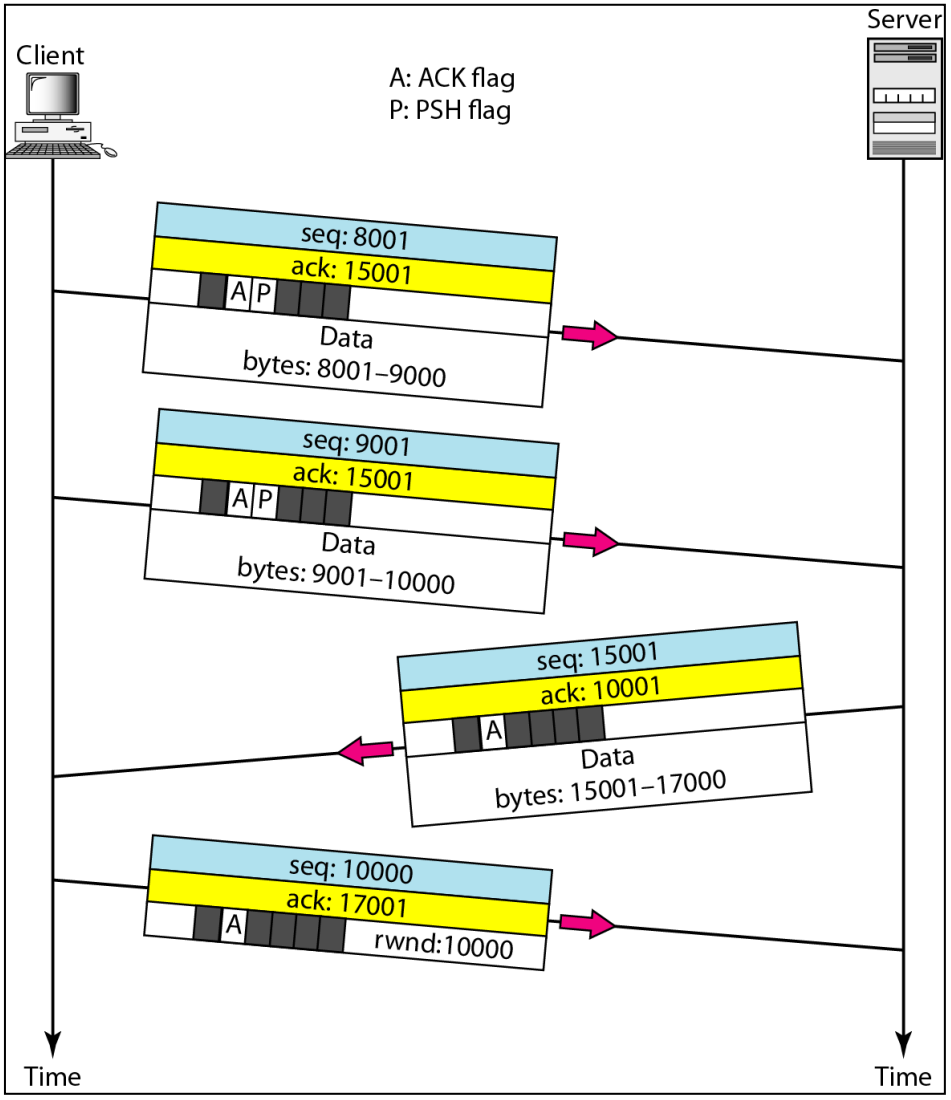
Connection establishment using three-way handshaking:



Connection-Oriented Transport: TCP

TCP Connection:

Data Transfer



Connection-Oriented Transport: TCP

TCP Connection:

Data Transfer

Push Flag:

- At Sender size, segment size is predefined and node will wait for the segment to be full.
- Suppose sender wants to send some keystroke value and want the immediate response.
- Delayed delivery of data not accepted.
- The Application program at sending side can request the push operation.
- TCP will not wait for the segment to be full, and send it immediately.

Connection-Oriented Transport: TCP

TCP Connection:

Data Transfer

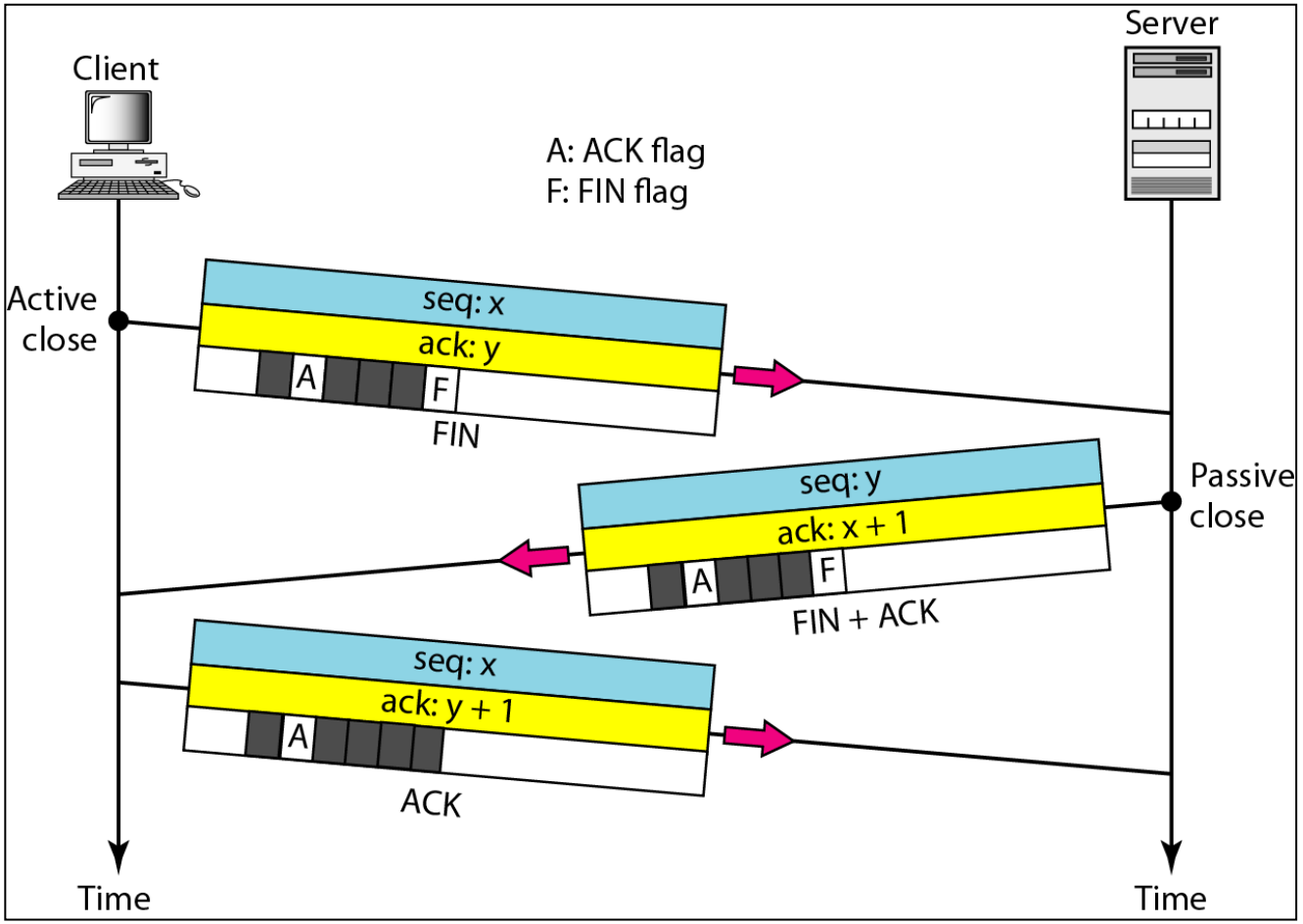
Urgent Flag:

- Sending process may want some particular bytes to read first.
- Example.
- The sending application program will tell the sending TCP that this data is urgent.
- The urgent pointer field in the header defines the end of the urgent data from one segments, remaining bytes are normal data from buffer.

Connection-Oriented Transport: TCP

TCP Connection:

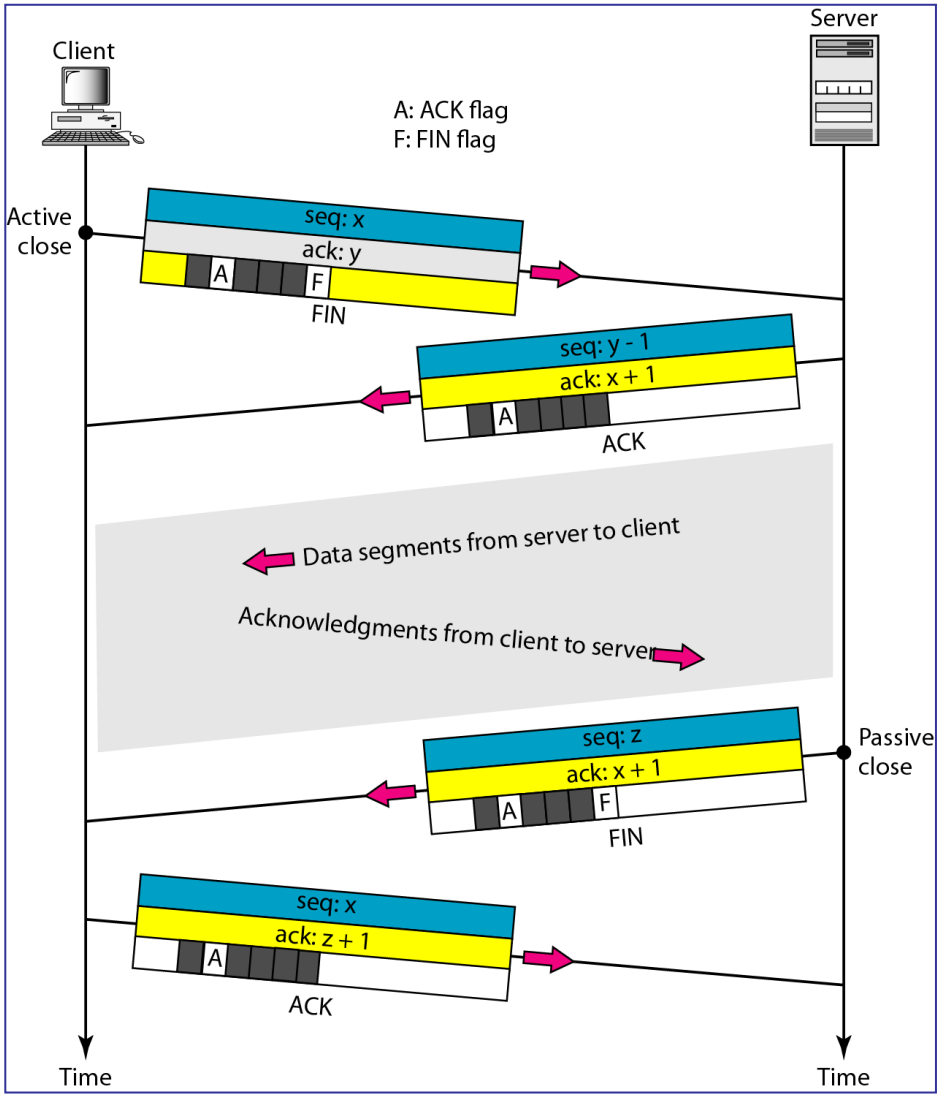
Connection Termination with Full Close



Connection-Oriented Transport: TCP

TCP Connection:

Connection Termination with Half Close:



Connection-Oriented Transport: TCP

TCP Congestion Control:

- TCP has built-in mechanisms to behave in ‘network friendly’ manner, not the UDP.
- Handles reliability, in-order delivery and solve congestion.
- Temporary overload at some point in the transmission path.
- Congestion may appear.
- The sum of the input rates of packets destined for one output link is higher than the capacity of the output link.
- Retransmission for missing acknowledgement.
- Slows down the transmission rate dramatically.
- Main reason of survival against UDP.

Connection-Oriented Transport: TCP

TCP Congestion Control:

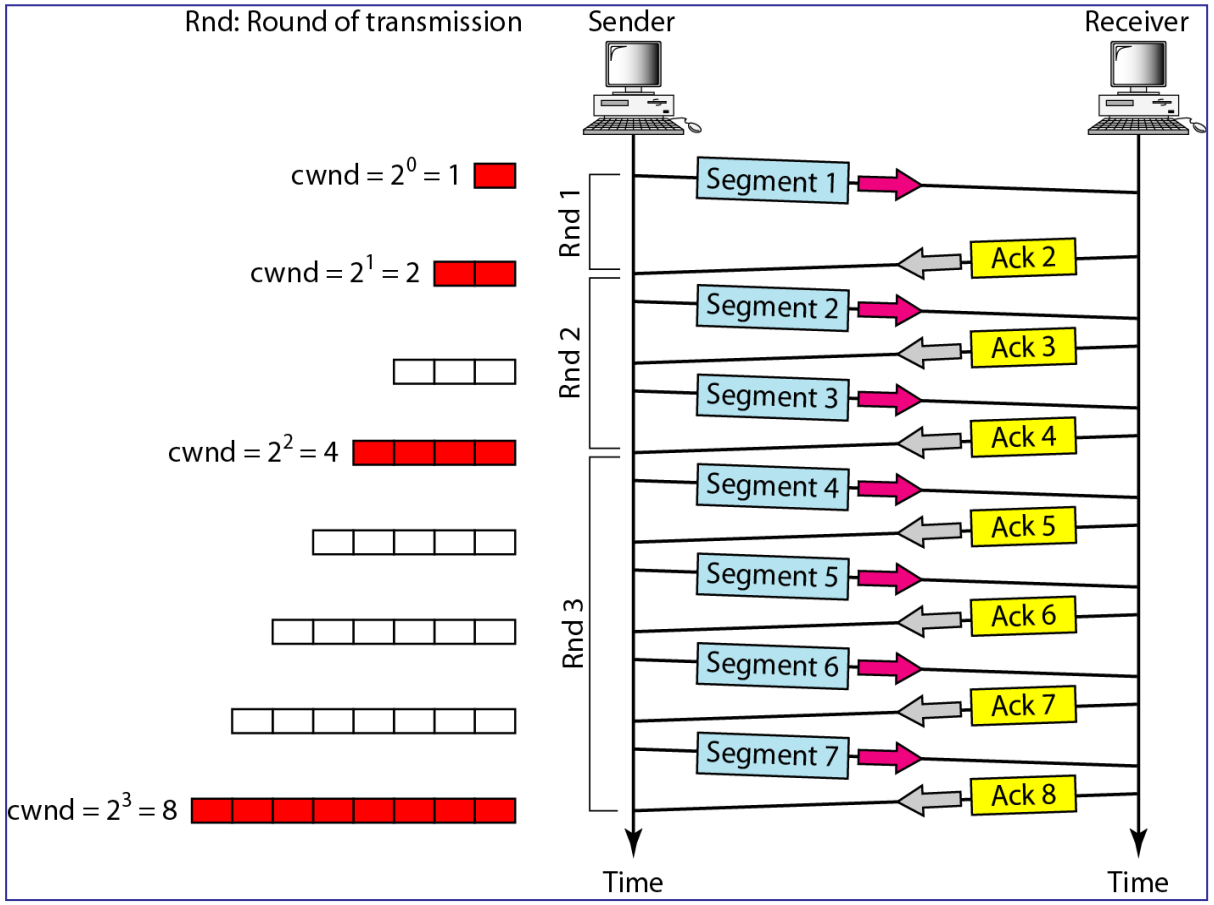
SLOW START

- The behavior TCP shows after the detection of congestion is called slow start.
- The sender always calculates a congestion window for a receiver.
- In the slow-start algorithm, the size of the congestion window increases exponentially until it reaches a threshold.

Connection-Oriented Transport: TCP

TCP Congestion Control:

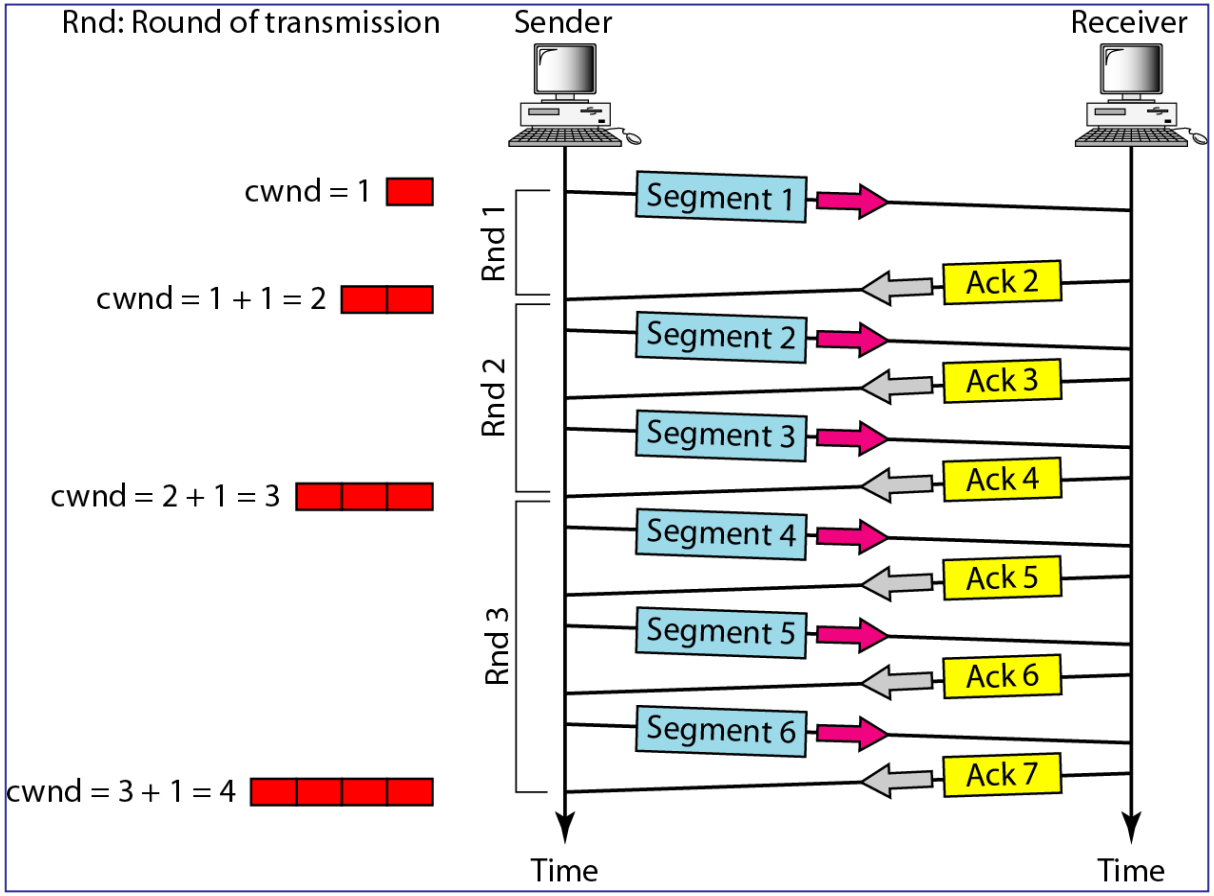
Slow Start, Exponential Increase



Connection-Oriented Transport: TCP

TCP Congestion Control:

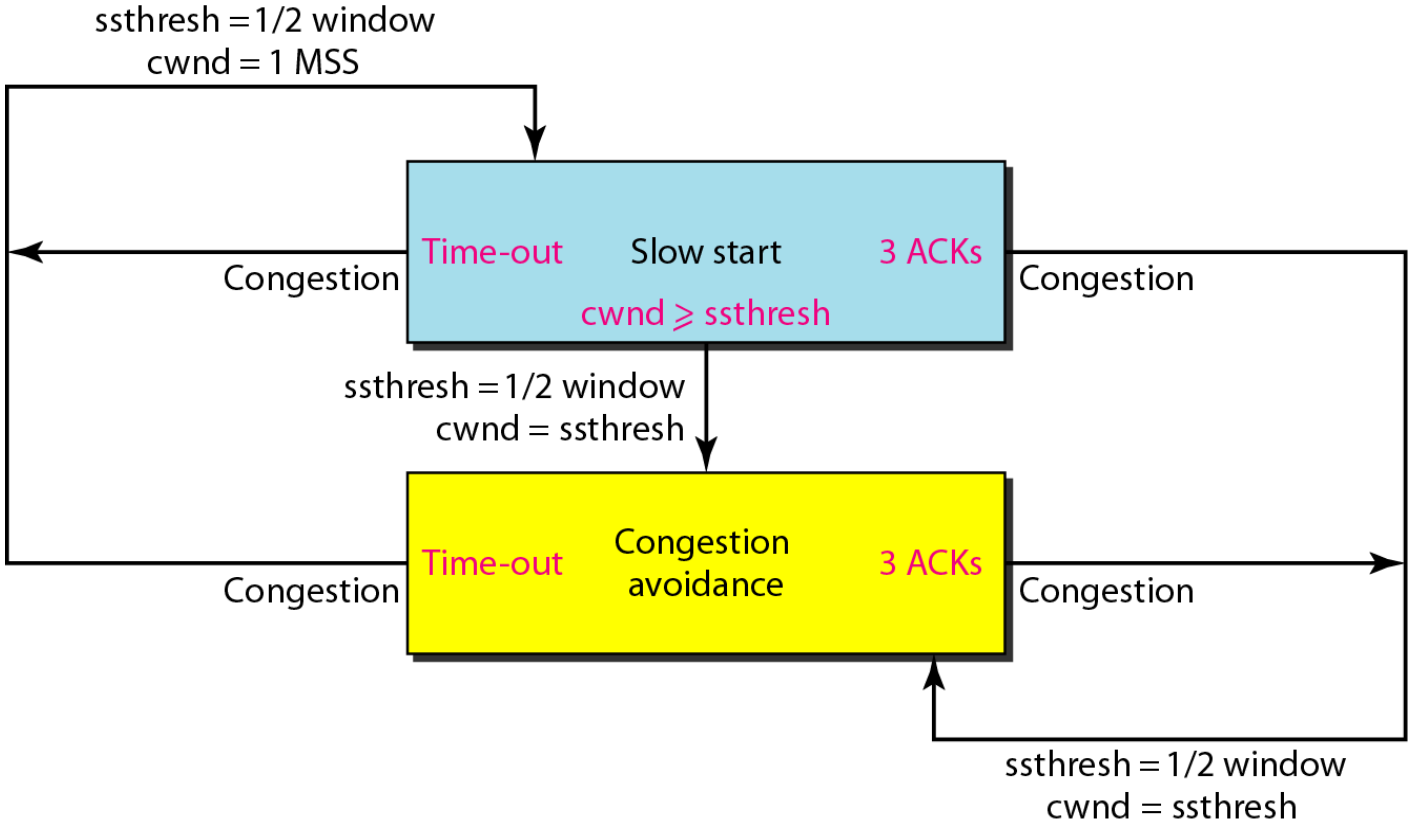
Congestion Avoidance, Additive Increase



Connection-Oriented Transport: TCP

TCP Congestion Control:

- In the congestion avoidance algorithm, the size of the congestion window increases additively until congestion is detected.



Connection-Oriented Transport: TCP

TCP Congestion Control:

Fast Retransmit/Fast Recovery

Two things lead to a reduction of the congestion threshold.

1. **Continuous acknowledgements for the same packet.**
   - Simple packet loss due to transmission error.
   - The sender can now retransmit the missing packet(s) before the timer expires. This behavior is called fast retransmit.
   - The sender performs a fast recovery from the packet loss and activate congestion avoidance.

Connection-Oriented Transport: TCP

TCP Congestion Control:

Fast Retransmit/Fast Recovery

2. Time-out due to a missing acknowledgement. Activates the slow start mechanism.