Connection-Oriented Transport: TCP

- TCP—the Internet's transport-layer, connection-oriented, reliable transport protocol.

The TCP Connection

- TCP is said to be connection-oriented because before one application process can begin to send data to another, the two processes must first "handshake" with each other—that is, they must send some preliminary segments to each other to establish the parameters of the ensuing data transfer.
- A TCP connection provides a full-duplex service
- A TCP connection is also always point-to-point, that is, between a single sender and a single receiver.
- Connection-establishment procedure is often referred to as a three-way handshake.

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).
- The MSS is typically set by first determining the length of the largest link-layer frame that can be sent by the local sending host (the so-called maximum transmission unit, MTU)

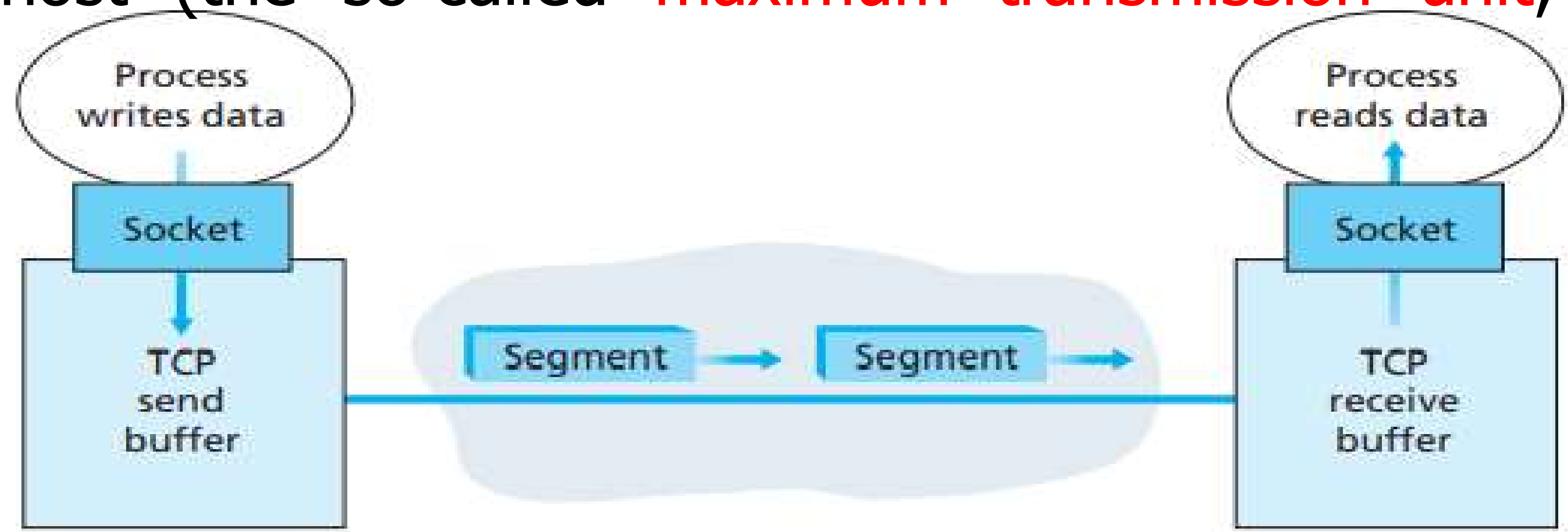


Figure 2.35 ◆ TCP send and receive buffers

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Connection-Oriented Transport: TCP

TCP Segment Structure

TCP header is typically 20 bytes (12 bytes more than the UDP header)

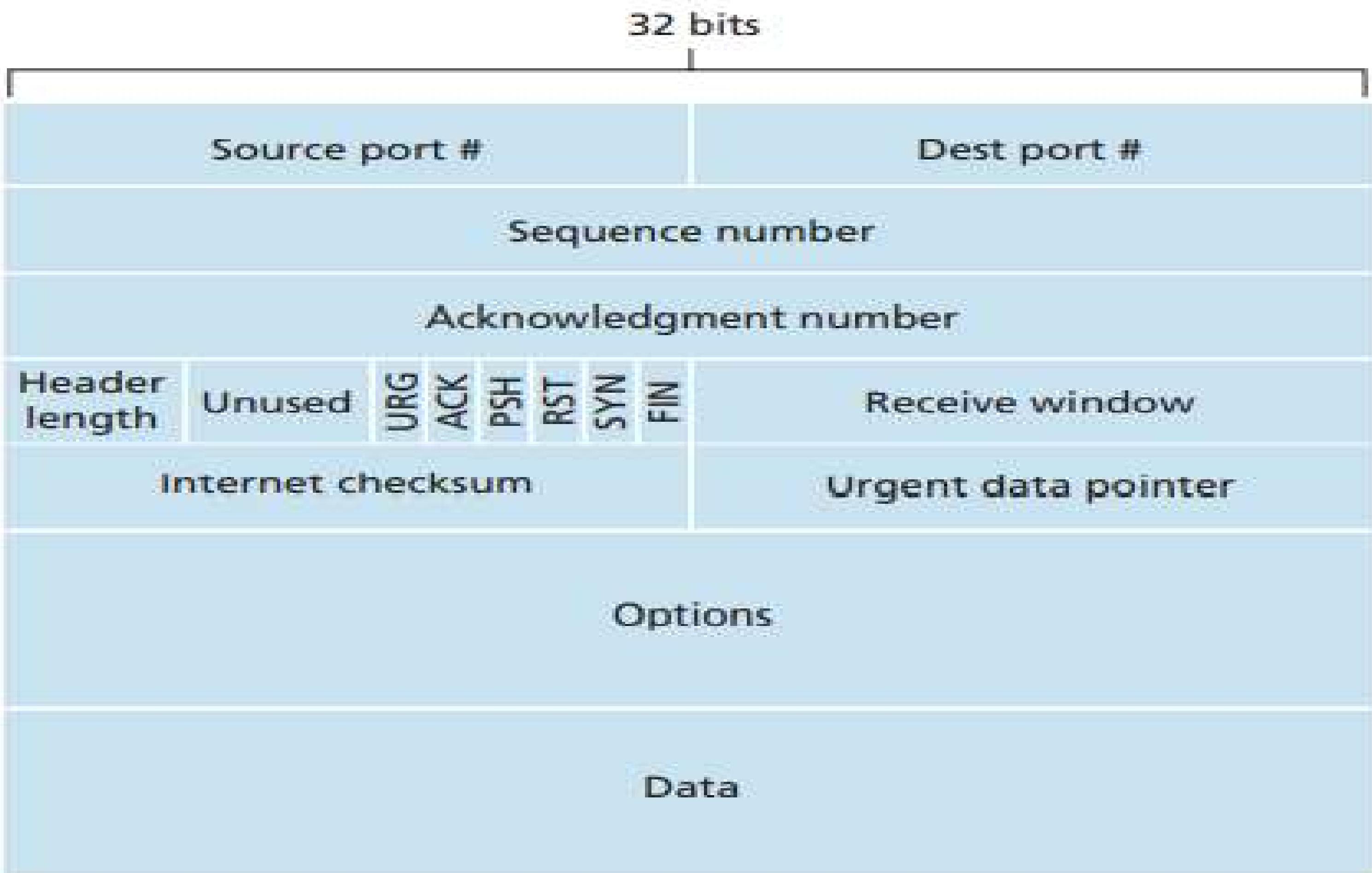


Figure 2.36 • TCP segment structure

James F Kurose and Keith W Ross, “Computer Networking: A Top - Down Approach”, Pearson Education; 6 th Edition (2017)

Connection-Oriented Transport: TCP

TCP Segment Structure

- 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.(error control)
- 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.
- The TCP header can be of variable length due to the TCP options field

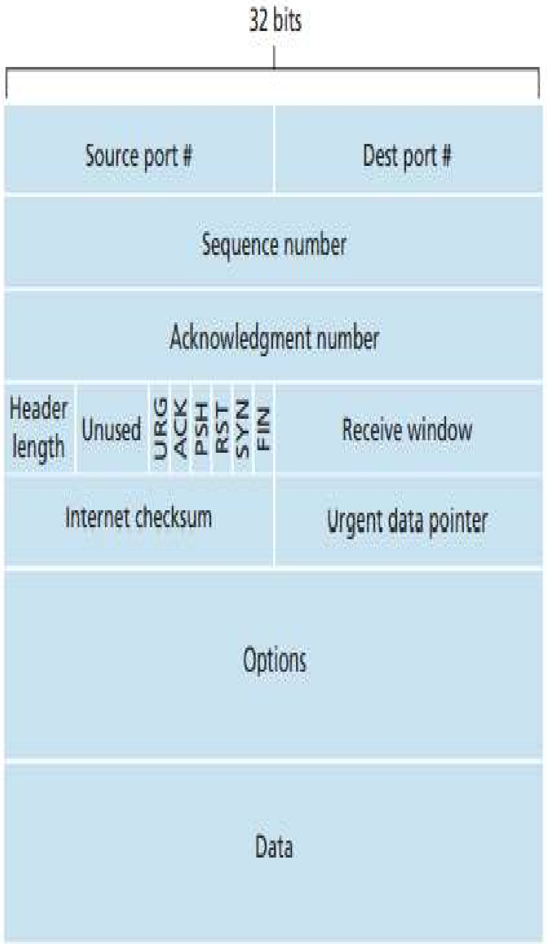


Figure 2.36 ♦ TCP segment structure

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Connection-Oriented Transport: TCP

TCP Segment Structure

- The flag field contains 6 bits.
- The ACK bit is used to indicate that the value carried in the acknowledgment field is valid.
- The RST, SYN, and FIN bits are used for connection setup and teardown.
- Setting the PSH bit indicates that the receiver should pass the data to the upper layer immediately.
- URG bit is used to indicate that there is data in this segment that the sending-side upper-layer entity has marked as "urgent."
- TCP provides cumulative acknowledgments

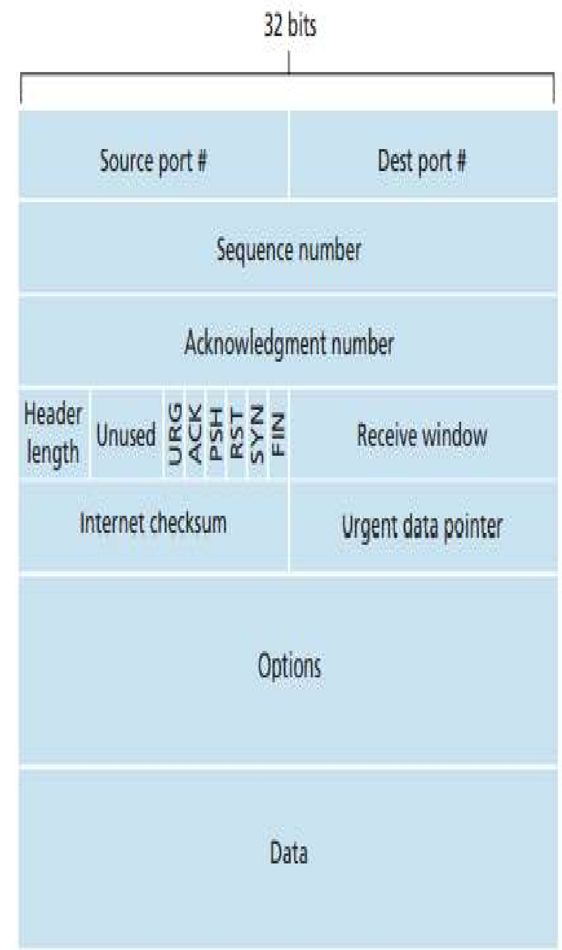


Figure 2.36 ♦ TCP segment structure

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Connection-Oriented Transport: TCP

Round-Trip Time Estimation and Timeout

TCP uses a timeout/retransmit mechanism to recover from lost segments.

Estimating the Round-Trip Time

- The sample RTT, denoted SampleRTT, for a segment is the amount of time between when the segment is sent (that is, passed to IP) and when an acknowledgment for the segment is received.
- TCP maintains an average, called EstimatedRTT, of the SampleRTT values.

$$\text{EstimatedRTT} = (1 - \alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}$$

- The new value of EstimatedRTT is a weighted combination of the previous value of EstimatedRTT and the new value for SampleRTT

Connection-Oriented Transport: TCP

Estimating the Round-Trip Time

The recommended value of α is = 0.125 (that is, 1/8) [RFC 6298]

$$\text{EstimatedRTT} = 0.875 \cdot \text{EstimatedRTT} + 0.125 \cdot \text{SampleRTT}$$

Connection-Oriented Transport: TCP

Round-Trip Time Estimation and Timeout

Estimating the Round-Trip Time

•In addition to having an estimate of the RTT, it is also valuable to have a measure of the variability of the RTT, DevRTT, as an estimate of how much SampleRTT typically deviates from EstimatedRTT:

$$\boxed{\mathtt{DevRTT} = (1 - \beta) \cdot \mathtt{DevRTT} + \beta \cdot |\mathtt{SampleRTT} - \mathtt{EstimatedRTT}|}$$

•The recommended value of β is 0.25

Connection-Oriented Transport: TCP

Round-Trip Time Estimation and Timeout

- The retransmission timeout interval

```
TimeoutInterval = EstimatedRTT + 4 · DevRTT
```

- An initial TimeoutInterval value of 1 second is recommended [RFC 6298].
- Also, when a timeout occurs, the value of TimeoutInterval is doubled to avoid a premature timeout occurring for a subsequent segment that will soon be acknowledged

Connection-Oriented Transport: TCP

Reliable Data Transfer

- TCP creates a reliable data transfer service on top of IP's unreliable best effort service.
- TCP's reliable data transfer service ensures that the data stream that a process reads out of its TCP receive buffer is uncorrupted, without gaps, without duplication, and in sequence.
- That is, the byte stream is exactly the same byte stream that was sent by the end system on the other side of the connection.

Connection-Oriented Transport: TCP

Reliable Data Transfer

- There are three major events related to data transmission and retransmission in the TCP sender: data received from application above; timer timeout; and ACK receipt.
- Upon the occurrence of the first major event, TCP receives data from the application, encapsulates the data in a segment, and passes the segment to IP.
- Note that each segment includes a sequence number that is the byte-stream number of the first data byte in the segment.
- If the timer is already not running for some other segment, TCP starts the timer when the segment is passed to IP. (It is helpful to think of the timer as being associated with the oldest unacknowledged segment.)
- The expiration interval for this timer is the TimeoutInterval, which is calculated from EstimatedRTT and DevRTT.

Connection-Oriented Transport: TCP

Reliable Data Transfer

- The second major event is the timeout.
- TCP responds to the timeout event by retransmitting the segment that caused the timeout.
- TCP then restarts the timer.
- The third major event that must be handled by the TCP sender is the arrival of an acknowledgment segment (ACK) from the receiver (more specifically, a segment containing a valid ACK field value).
- On the occurrence of this event, TCP compares the ACK value y with its variable SendBase.
- The TCP state variable SendBase is the sequence number of the oldest unacknowledged byte.

Connection-Oriented Transport: TCP

Reliable Data Transfer

- SendBase–1 is the sequence number of the last byte that is known to have been received correctly and in order at the receiver.
- TCP uses cumulative acknowledgments, so that y acknowledges the receipt of all bytes before byte number y.
- If y > SendBase then the ACK is acknowledging one or more previously unacknowledged segments. Thus the sender updates its SendBase variable.
- It also restarts the timer if there currently are any not-yet-acknowledged segments.

Connection-Oriented Transport: TCP

```
/* Assume sender is not constrained by TCP flow or congestion control, that data from above is less
than MSS in size, and that data transfer is in one direction only. */

NextSeqNum=InitialSeqNumber
SendBase=InitialSeqNumber

loop (forever) {
    switch(event)

        event: data received from application above
            create TCP segment with sequence number NextSeqNum
            if (timer currently not running)
                start timer
            pass segment to IP
            NextSeqNum=NextSeqNum+length(data)
            break;

        event: timer timeout
            retransmit not-yet-acknowledged segment with
                smallest sequence number
            start timer
            break;

        event: ACK received, with ACK field value of y
            if (y > SendBase) {
                SendBase=y
                if (there are currently any not-yet-acknowledged segments)
                    start timer
                }
            break;

    } /* end of loop forever */
```

Three major events related to data transmission and retransmission in the TCP sender: data received from application above; timer timeout; and ACK receipt.

Figure 2.36 a ◆ Simplified TCP sender

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Connection-Oriented Transport: TCP

Flow Control

- TCP provides a flow-control service to its applications to eliminate the possibility of the sender overflowing the receiver's buffer.
- Flow control is thus a speed-matching service—matching the rate at which the sender is sending against the rate at which the receiving application is reading.
- A TCP sender can also be throttled due to congestion within the IP network; this form of sender control is referred to as congestion control.

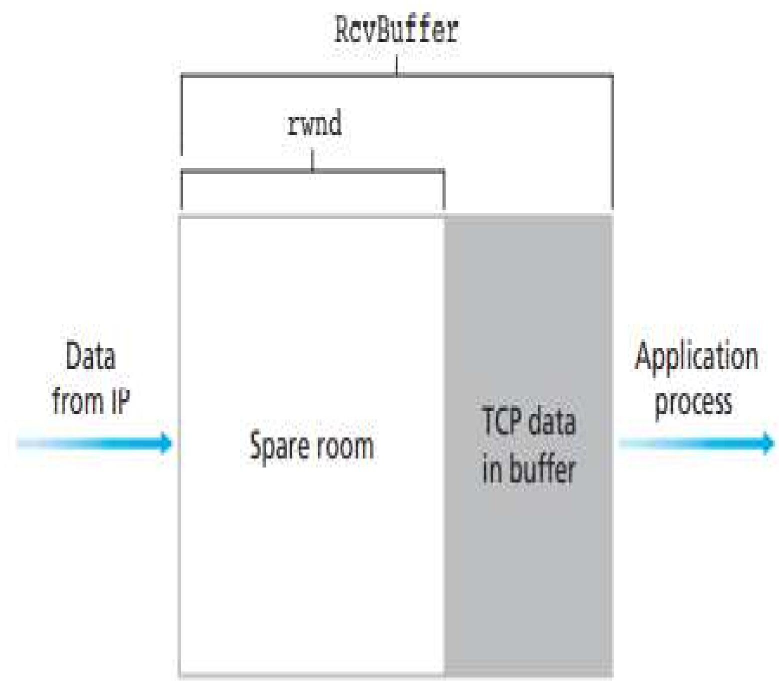


Figure 2.37 ♦ The receive window (rwnd) and the receive buffer (RcvBuffer)

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Connection-Oriented Transport: TCP

TCP Connection Management

The TCP in the client then proceeds to establish a TCP connection with the TCP in the server in the following manner:

Step 1. The client-side TCP first sends a special TCP segment to the server-side TCP. (connection request)

- This special segment contains no application-layer data. But one of the flag bits in the segment's header the SYN bit, is set to 1. For this reason, this special segment is referred to as a SYN segment.
- In addition, the client randomly chooses an initial sequence number (client_isn) and puts this number in the sequence number field of the initial TCP SYN segment.
- This segment is encapsulated within an IP datagram and sent to the server.

Connection-Oriented Transport: TCP

TCP Connection Management

Step 2. Once the IP datagram containing the TCP SYN segment arrives at the server host, the server extracts the TCP SYN segment from the datagram, allocates the TCP buffers and variables to the connection, and sends a connection-granted segment to the client TCP. (connection granted)

- It does contain three important pieces of information in the segment header.
  - First, the SYN bit is set to 1.
  - Second, the acknowledgment field of the TCP segment header is set to client_isn+1.
  - Finally, the server chooses its own initial sequence number (server_isn) and puts this value in the sequence number field of the TCP segment header

Connection-Oriented Transport: TCP

TCP Connection Management

Step 3. Upon receiving the SYNACK segment, the client also allocates buffers and variables to the connection. (ack)

- The client host then sends the server yet another segment; this last segment acknowledges the server's connection-granted segment (the client does so by putting the value server_isn+1 in the acknowledgment field of the TCP segment header).
- The SYN bit is set to zero, since the connection is established.

This third stage of the three-way handshake may carry client-to-server data in the segment payload.

Connection-Oriented Transport: TCP

TCP Connection Management

- In order to establish the connection, three packets are sent between the two hosts.
- For this reason, this connection establishment procedure is often referred to as a three-way handshake

Three-way handshake

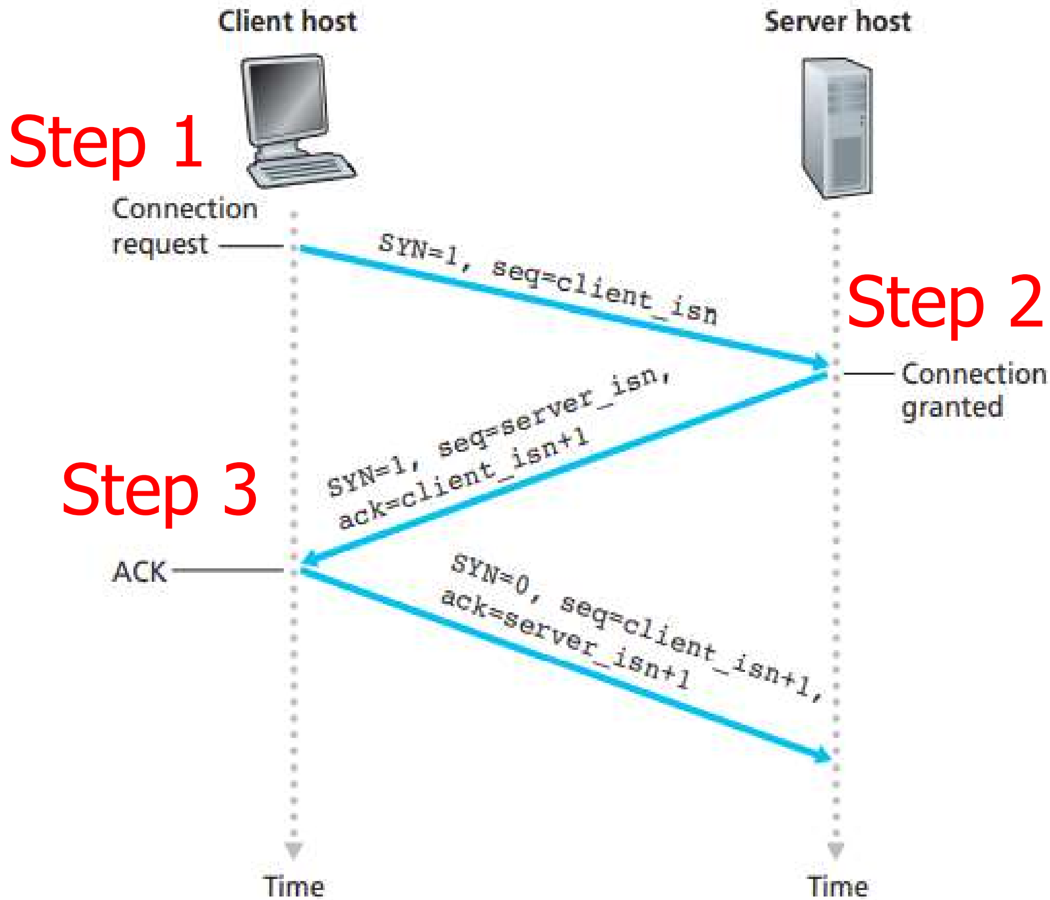


Figure 2.38 ♦ TCP three-way handshake: segment exchange

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Connection-Oriented Transport: TCP

TCP Connection Management

- Closing a connection

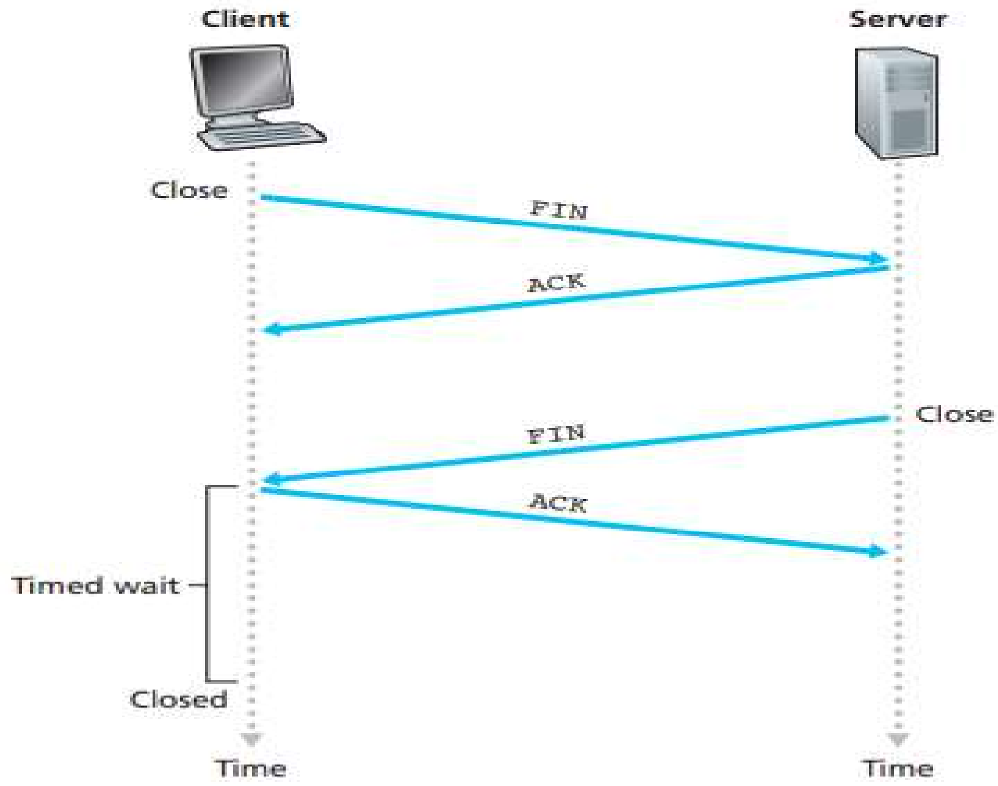


Figure 2.38 a • Closing a TCP connection

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Connection-Oriented Transport: TCP

TCP States – Client TCP

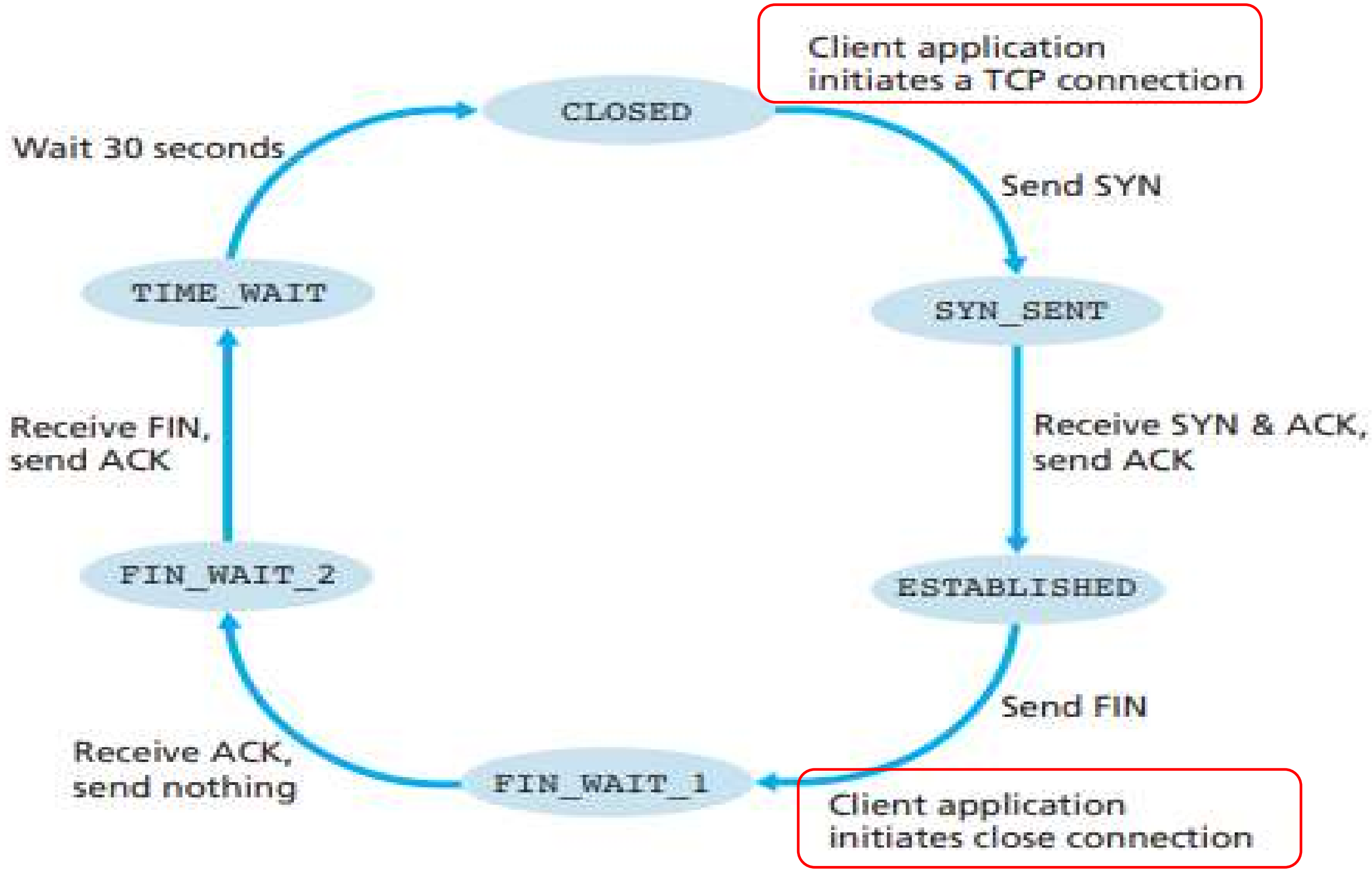


Figure 2.38 b ♦ A typical sequence of TCP states visited by a client TCP

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Connection-Oriented Transport: TCP

TCP States – Server side TCP

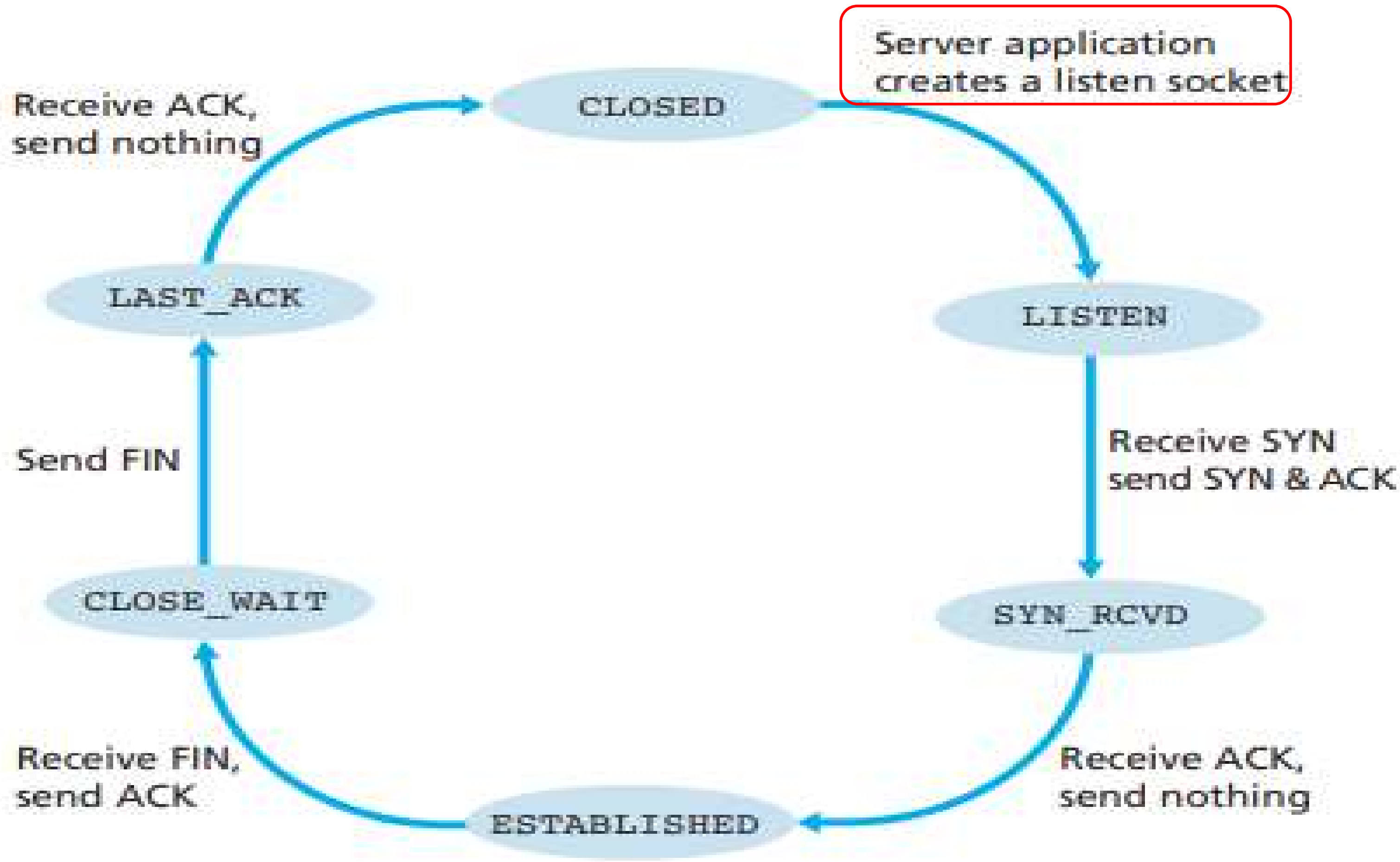


Figure 2.38 c ♦ A typical sequence of TCP states visited by a server-side TCP

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Principles of Congestion Control

The Causes and the Costs of Congestion

Scenario 1: Two Senders, a Router with Infinite Buffers

- Even in this (extremely) idealized scenario, we can see one cost of a congested network—large queuing delays are experienced as the packet arrival rate nears the link capacity.

The rate at which Host A offers traffic to the router

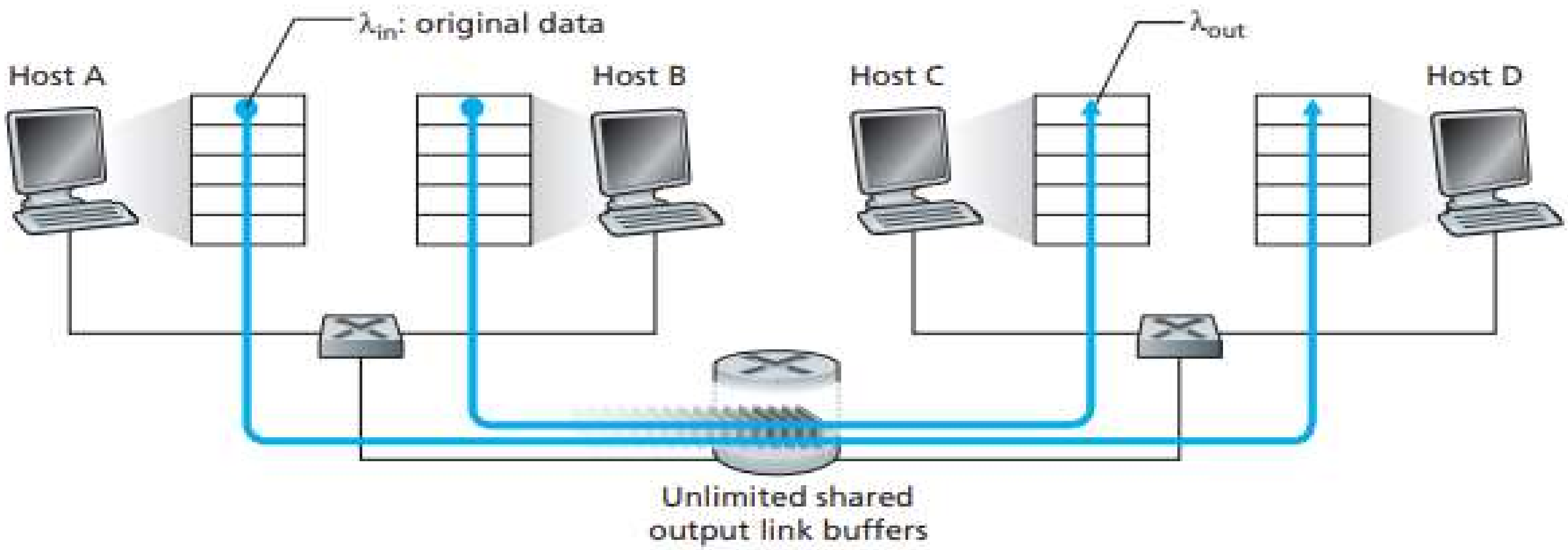


Figure 2.39 a ◆ Congestion scenario 1: Two connections sharing a single hop with infinite buffers

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Principles of Congestion Control

The Causes and the Costs of Congestion
Scenario 2: Two Senders and a Router with Finite Buffers

- Another cost of a congested network—the sender must perform retransmissions in order to compensate for dropped (lost) packets due to buffer overflow.

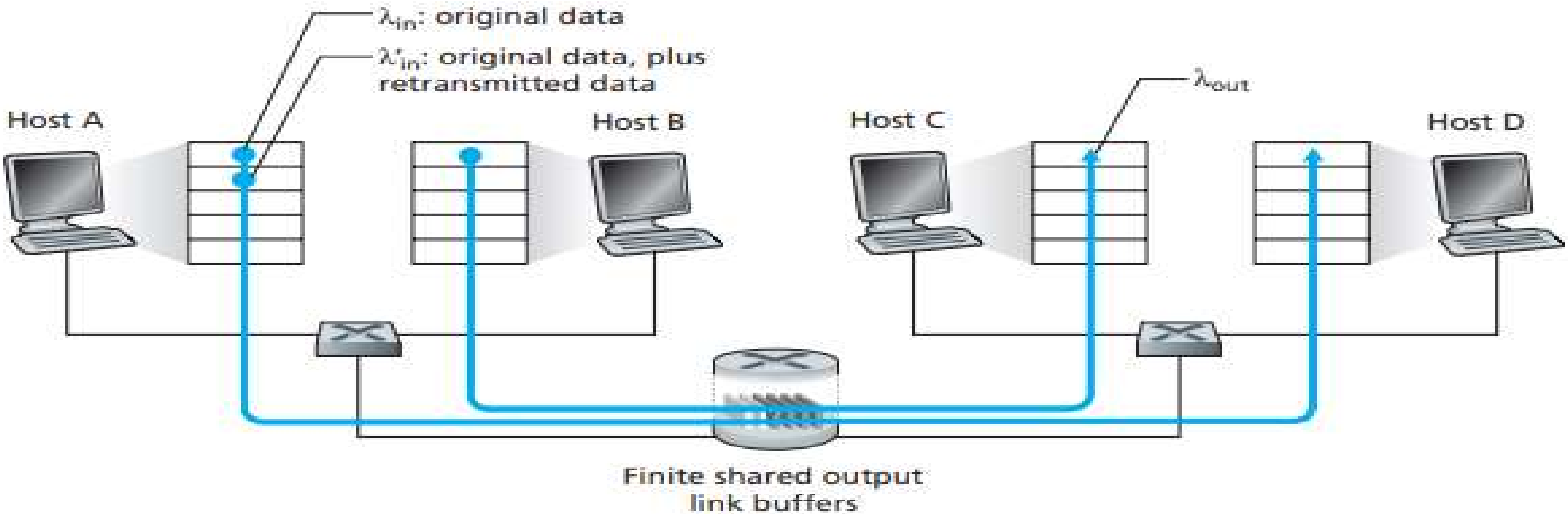


Figure2.39 b ◆ Scenario 2: Two hosts (with retransmissions) and a router with finite buffers

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Principles of Congestion Control

The Causes and the Costs of Congestion

Scenario 3: Four Senders, Routers with Finite Buffers, and Multihop Paths

- Another cost of dropping a packet due to congestion—when a packet is dropped along a path, the transmission capacity that was used at each of the upstream links to forward that packet to the point at which it is dropped ends up having been wasted.

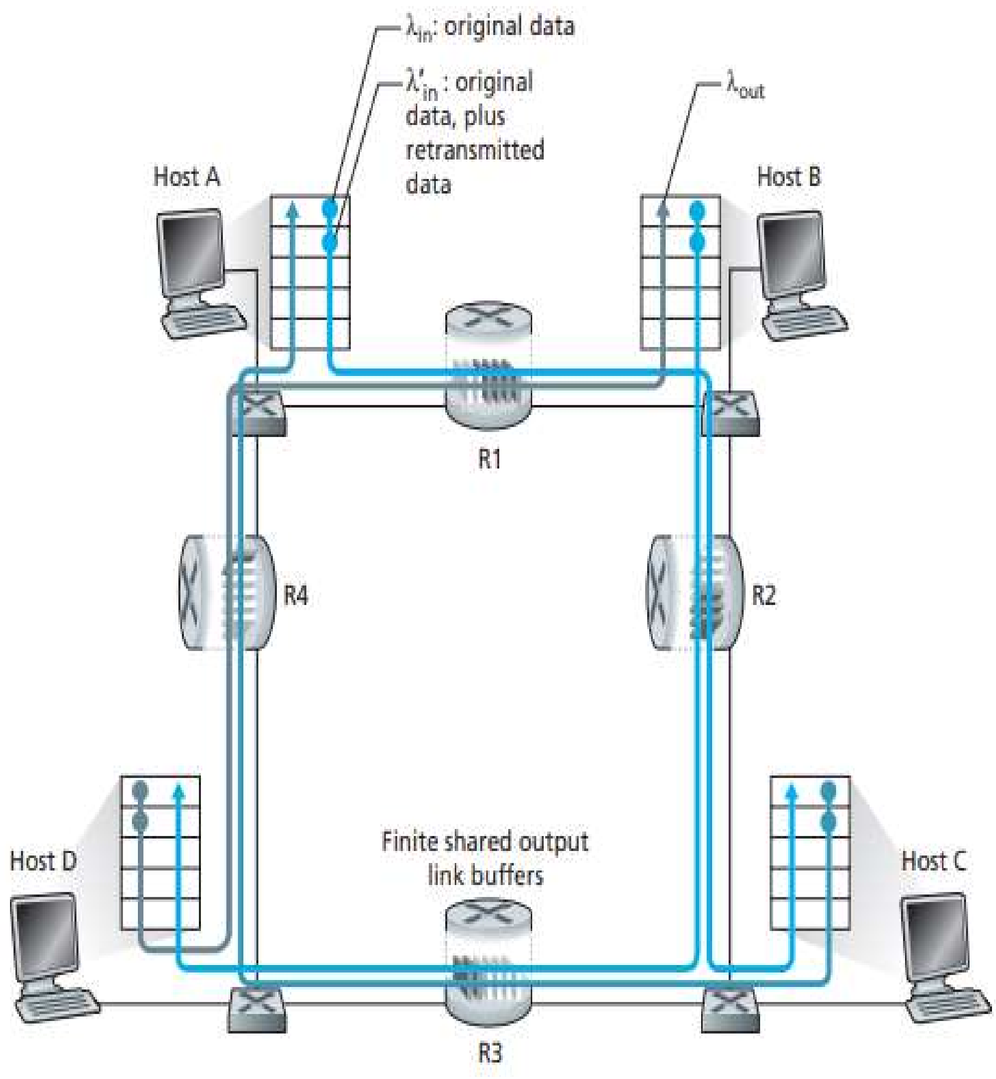


Figure 2.39c ♦ Four senders, routers with finite buffers, and multihop paths

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Principles of Congestion Control

Approaches to Congestion Control

- End-to-end congestion control

In an end-to-end approach to congestion control, the network layer provides no explicit support to the transport layer for congestion control purposes. Even the presence of congestion in the network must be inferred by the end systems based only on observed network behavior (for example, packet loss and delay)

- Network-assisted congestion control

With network-assisted congestion control, network-layer components (that is, routers) provide explicit feedback to the sender regarding the congestion state in the network. This feedback may be as simple as a single bit indicating congestion at a link.

Principles of Congestion Control

- For network-assisted congestion control, congestion information is typically fed back from the network to the sender in one of two ways, as shown in Figure 2.39.
- Direct feedback may be sent from a network router to the sender. This form of notification typically takes the form of a choke packet (essentially saying, “I’m congested!”).

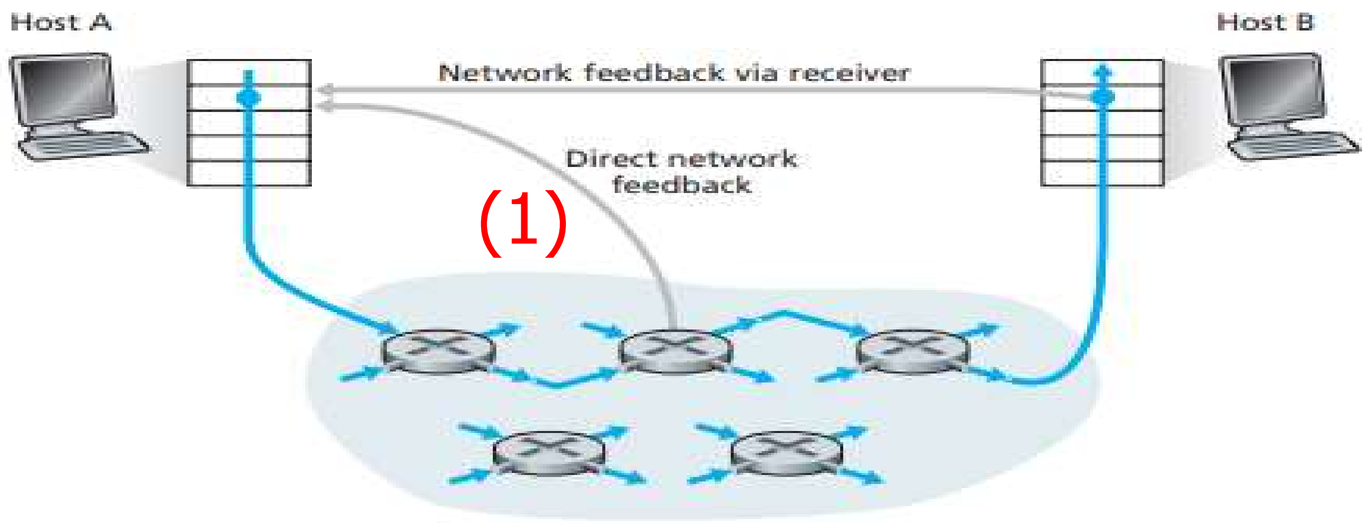


Figure 2.39 • Two feedback pathways for network-indicated congestion information

James F Kurose and Keith W Ross, “Computer Networking: A Top - Down Approach”, Pearson Education; 6 th Edition (2017)

Principles of Congestion Control

- The second form of notification occurs when a router marks/updates a field in a packet flowing from sender to receiver to indicate congestion.
- Upon receipt of a marked packet, the receiver then notifies the sender of the congestion indication. Note that this latter form of notification takes at least a full round-trip time

(2)

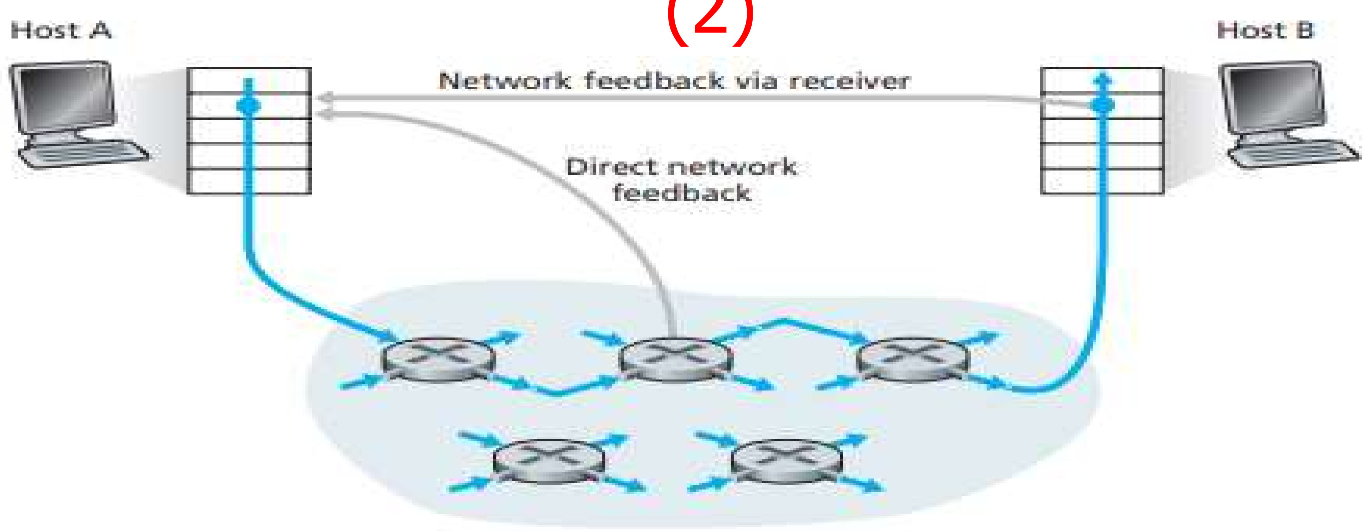


Figure 2.39 • Two feedback pathways for network-indicated congestion information

James F Kurose and Keith W Ross, “Computer Networking: A Top - Down Approach”, Pearson Education; 6 th Edition (2017)

TCP Congestion Control

Guiding Principles:

- A lost segment implies congestion, and hence, the TCP sender's rate should be decreased when a segment is lost.
- An acknowledged segment indicates that the network is delivering the sender's segments to the receiver, and hence, the sender's rate can be increased when an ACK arrives for a previously unacknowledged segment.
- Bandwidth probing. Given ACKs indicating a congestion-free source-to-destination path and loss events indicating a congested path, TCP's strategy for adjusting its transmission rate is to increase its rate in response to arriving ACKs until a loss event occurs, at which point, the transmission rate is decreased. The TCP sender thus increases its transmission rate to probe for the rate that at which congestion onset begins, backs off from that rate, and then to begins probing again to see if the congestion onset rate has changed.

TCP Congestion Control

- The TCP congestion-control mechanism operating at the sender keeps track of an additional variable, the congestion window(cwnd).
- TCP congestion-control algorithm

  The algorithm has three major components:

  (1) Slow start

  (2) Congestion avoidance

  (3) Fast recovery
- Slow start and congestion avoidance are mandatory components of TCP, differing in how they increase the size of cwnd in response to received ACKs. Fast recovery is recommended, but not required, for TCP senders.
- The congestion window, denoted cwnd, imposes a constraint on the rate at which a TCP sender can send traffic into the network.

TCP Congestion Control

- The TCP congestion-control mechanism operating at the sender keeps track of an additional variable, the congestion window(cwnd).
- TCP congestion-control algorithm

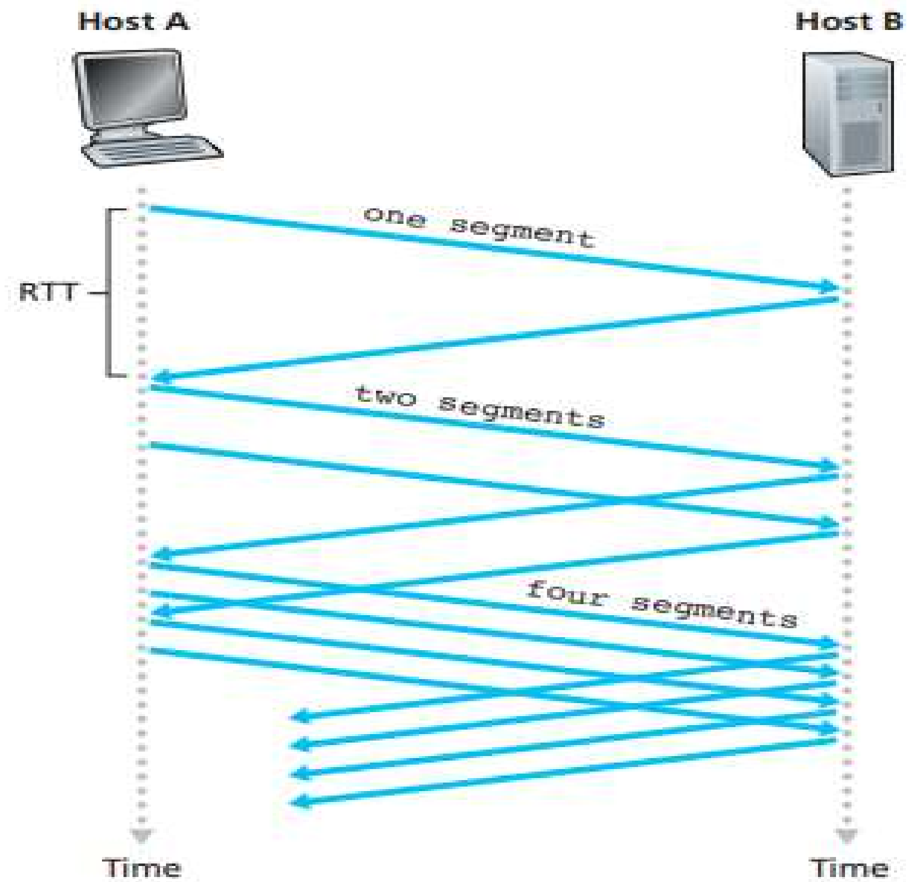


Figure 2.40 a • TCP slow start

James F Kurose and Keith W Ross, “Computer Networking: A Top - Down Approach”, Pearson Education; 6 th Edition (2017)

TCP Congestion Control

- The TCP congestion-control mechanism operating at the sender keeps track of an additional variable, the congestion window(cwnd).
- TCP congestion-control algorithm

- Threshold is initially equal to 8•MSS
- Congestion window is 12•MSS when loss event(triple duplicate-ACK event) occurs
- The value of ssthresh is then set to 0.5•cwnd = 6•MSS
- Under TCP Reno, the congestion window is set to cwnd = 9•MSS and then grows linearly.
- Under TCP Tahoe, the congestion window is set to 1 MSS and grows exponentially until it reaches the value of ssthresh, at which point it grows linearly

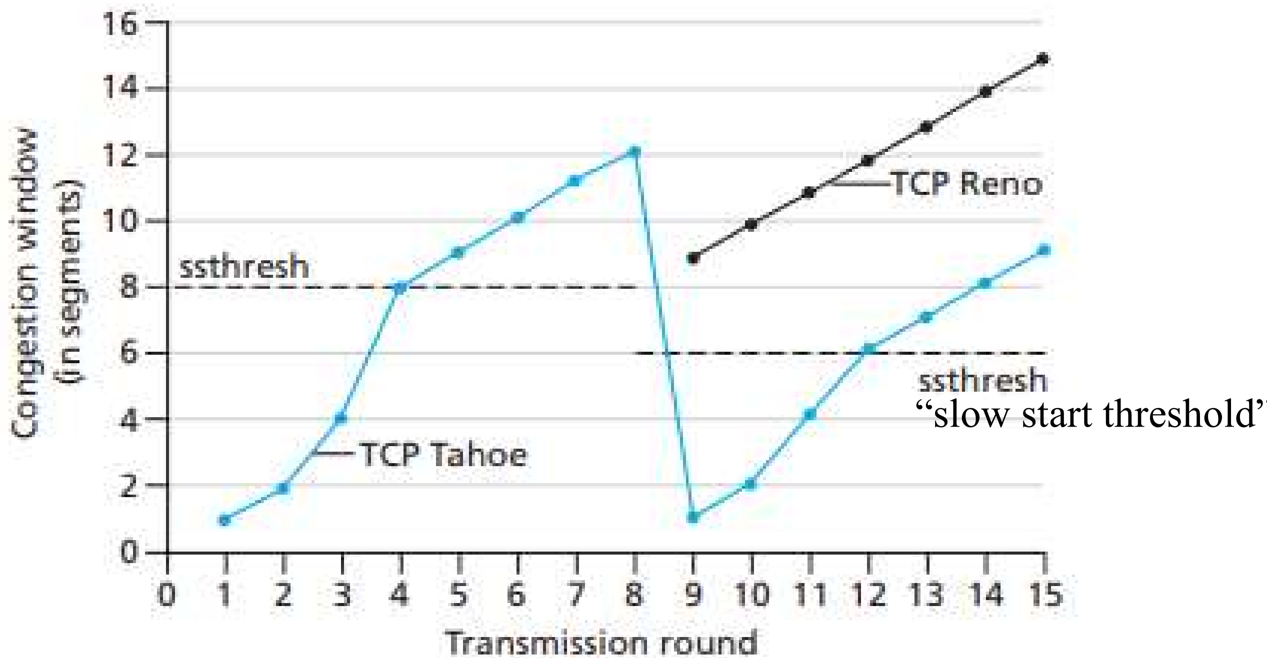


Figure 2.40 ♦ Evolution of TCP's congestion window (Tahoe and Reno)

maximum segment size (MSS)

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

TCP Congestion Control

- The TCP congestion-control mechanism operating at the sender keeps track of an additional variable, the congestion window(cwnd).
- TCP congestion-control algorithm

maximum segment size(MSS)

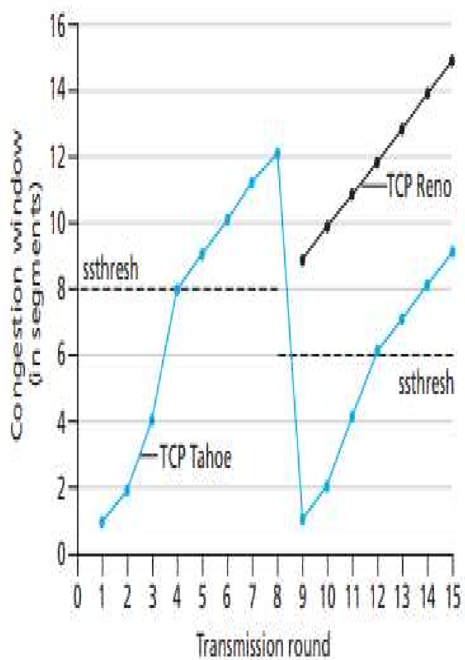


Figure 2.40 ♦ Evolution of TCP's congestion window (Tahoe and Reno)

- It is interesting that an early version of TCP, known as TCP Tahoe, unconditionally cut its congestion window to 1 MSS and entered the slow-start phase after either a timeout-indicated or triple-duplicate-ACK-indicated loss event.

- The newer version of TCP, TCP Reno, incorporated fast recovery.

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

TCP Congestion Control

- TCP congestion-control algorithm

  3 Stages:

  Slow Start: Exponential Increase

  Congestion Avoidance: Additive Increase

  Congestion Detection: Multiplicative Decrease

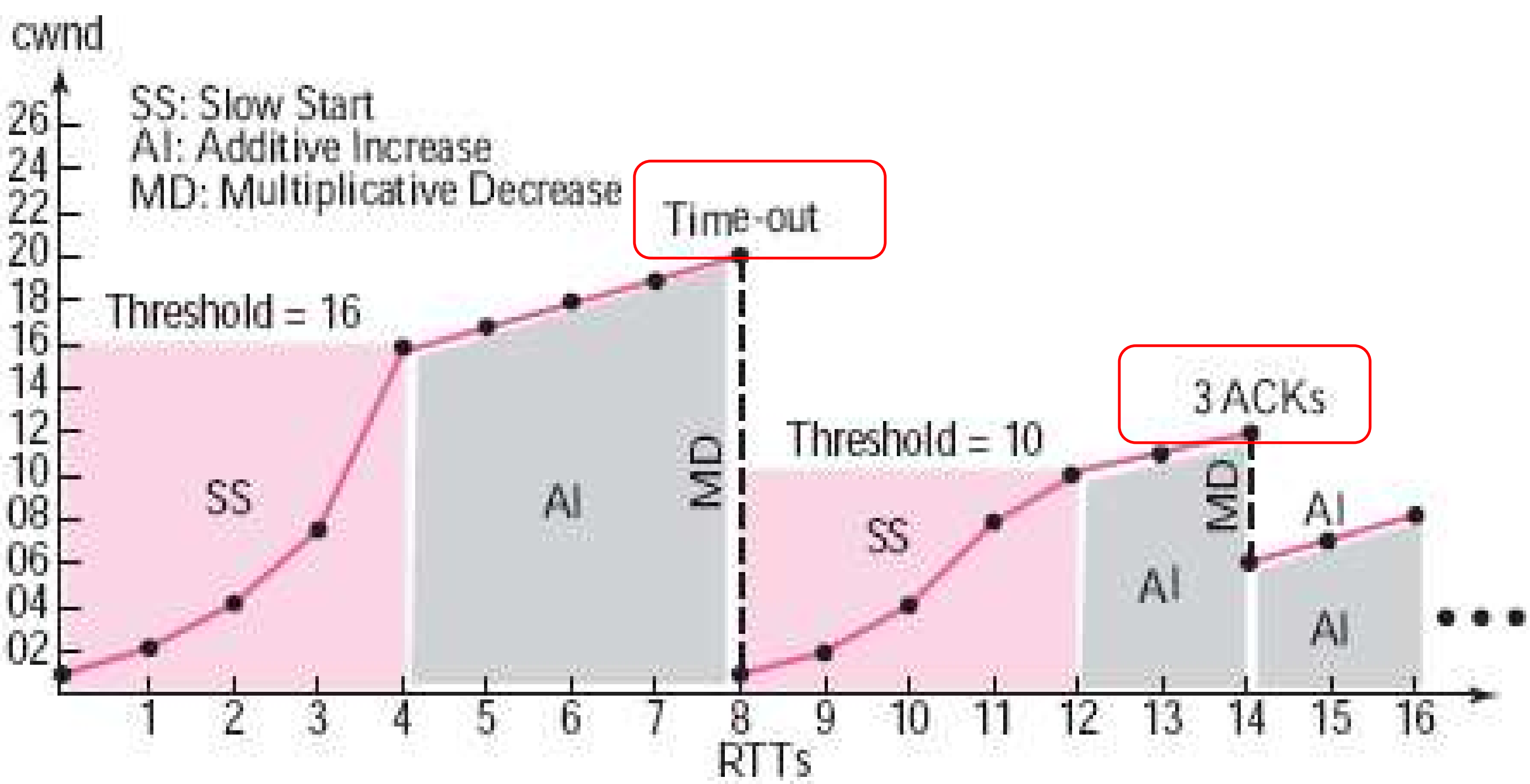


Figure 3.69 *Example of Taho TCP*

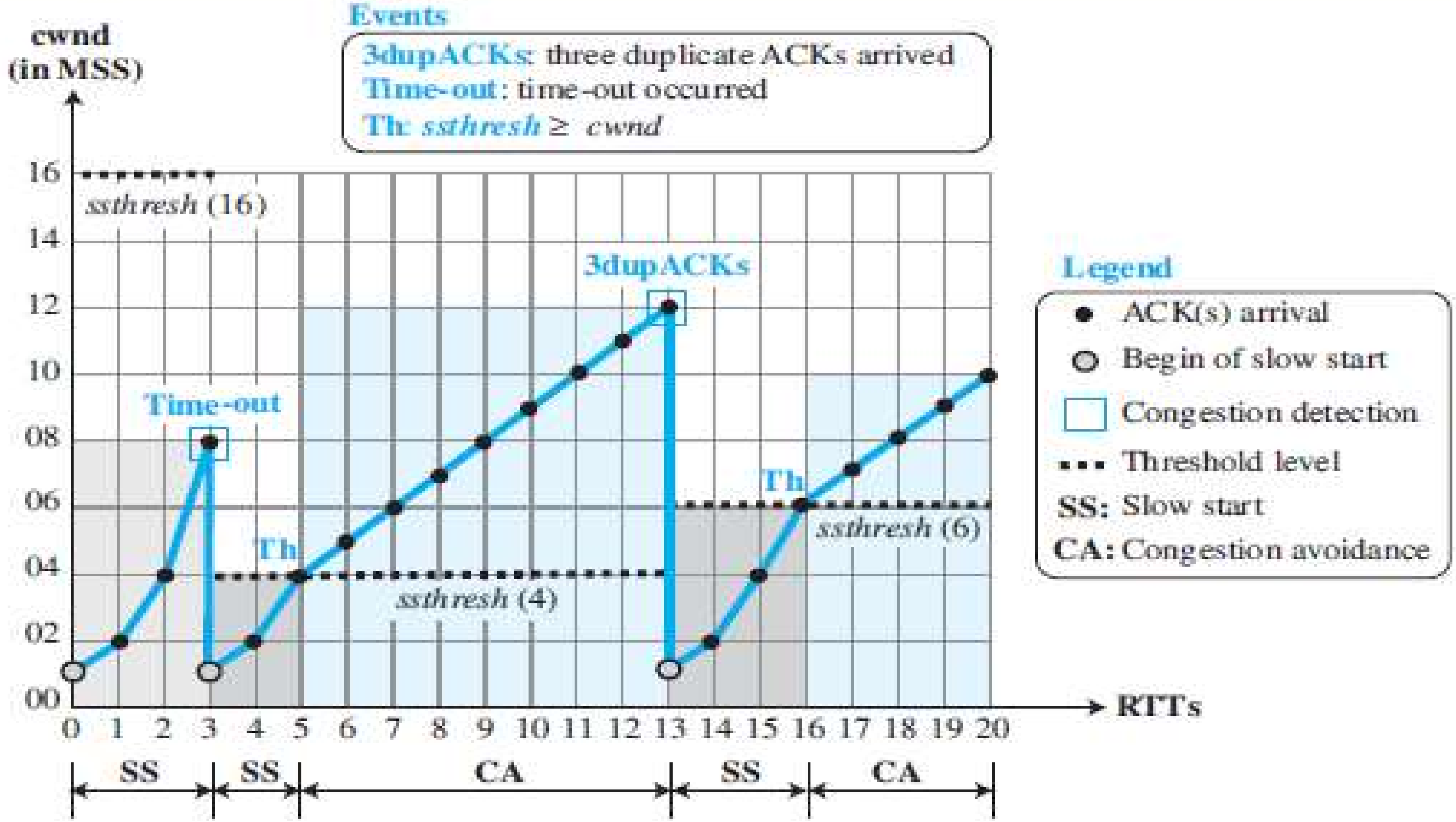


Figure 3.71 *Example of a Reno TCP*

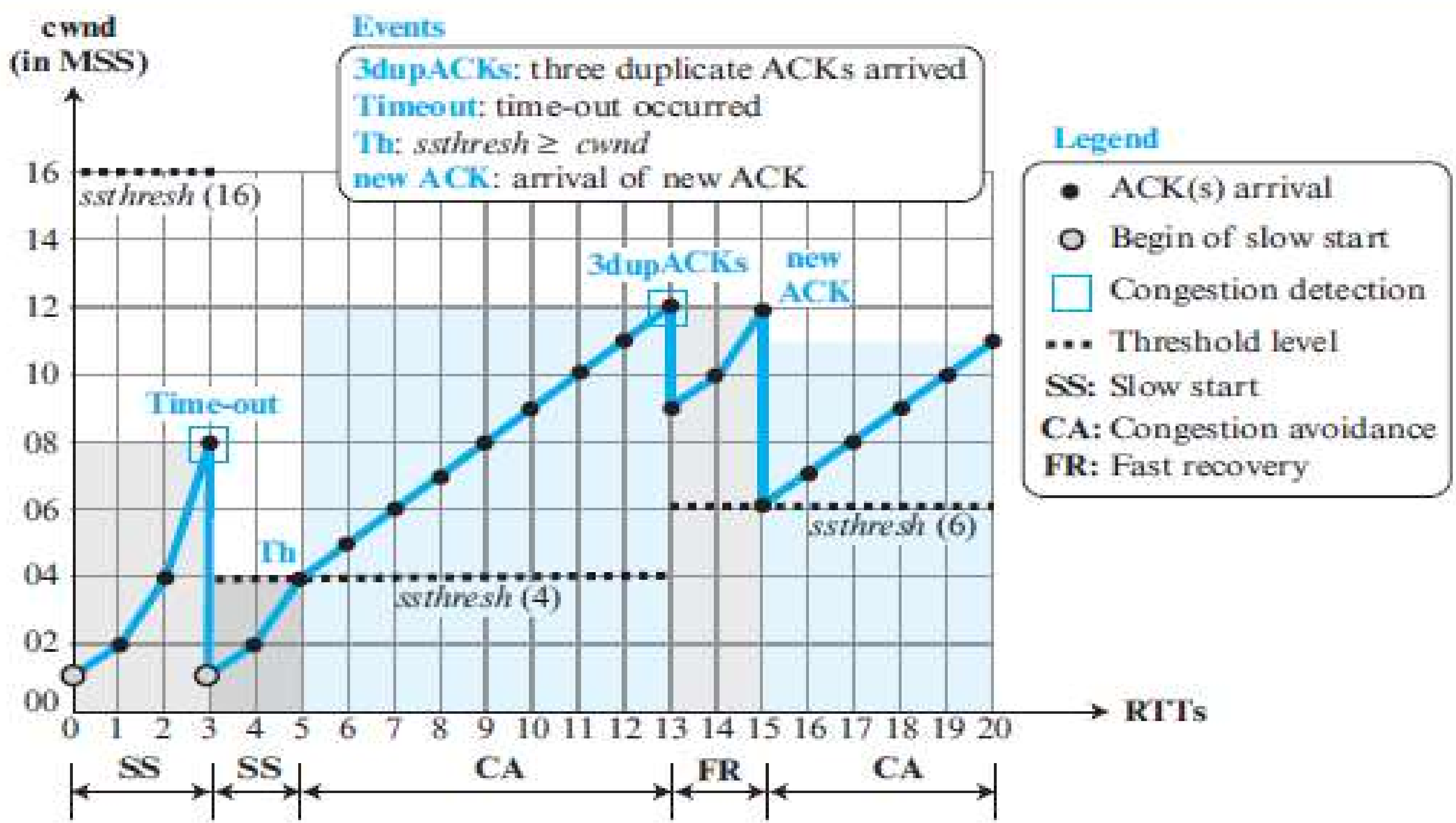
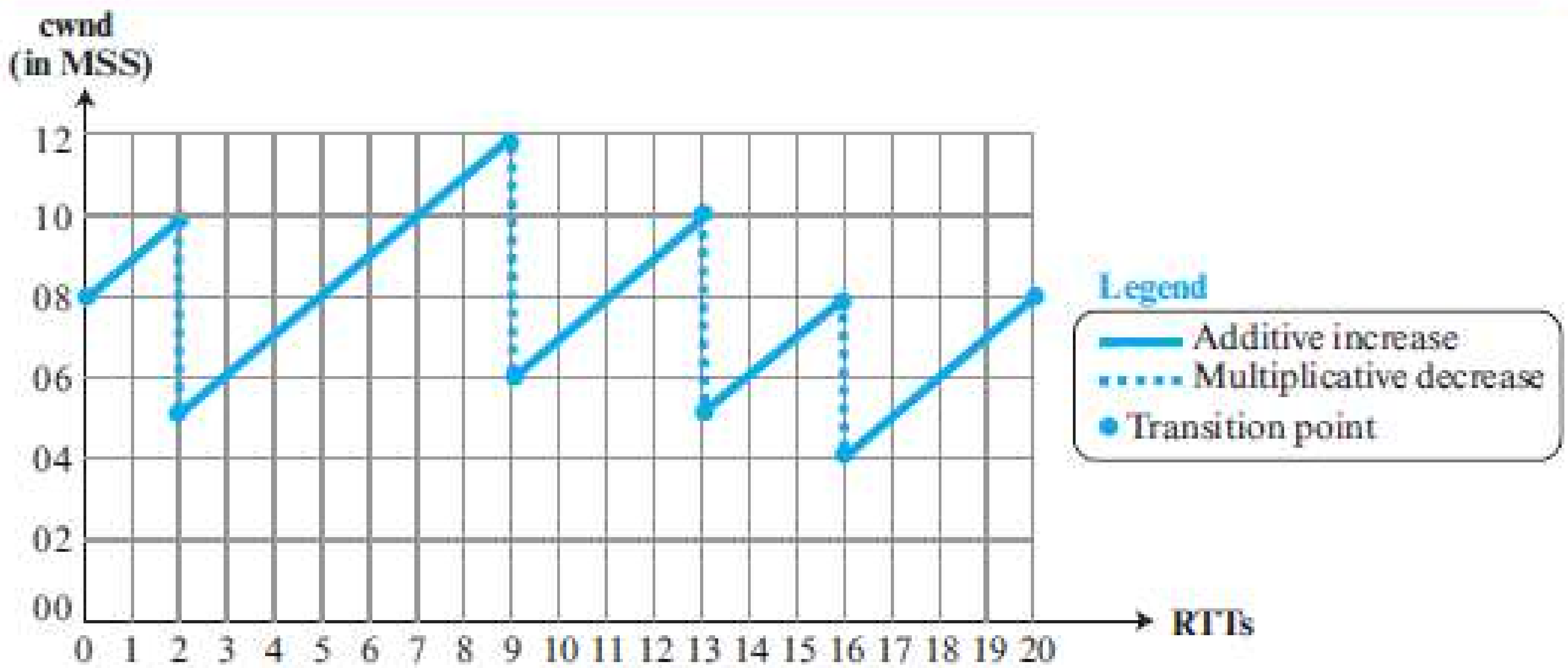


Figure 3.72 *Additive increase, multiplicative decrease (AIMD)*



TCP Throughput

Throughput = (0.75) Wmax / RTT

in which Wmax is the average of window sizes when the congestion occurs.

TCP Congestion Control

- TCP congestion control is often referred to as an additive-increase, multiplicative - decrease (AIMD) form of congestion control.
- AIMD congestion control gives rise to the “saw tooth” behavior which also nicely illustrates “probing” for bandwidth—TCP linearly increases its congestion window size (and hence its transmission rate) until a triple duplicate-ACK event occurs.
- It then decreases its congestion window size by a factor of two but then again begins increasing it linearly, probing to see if there is additional available bandwidth.

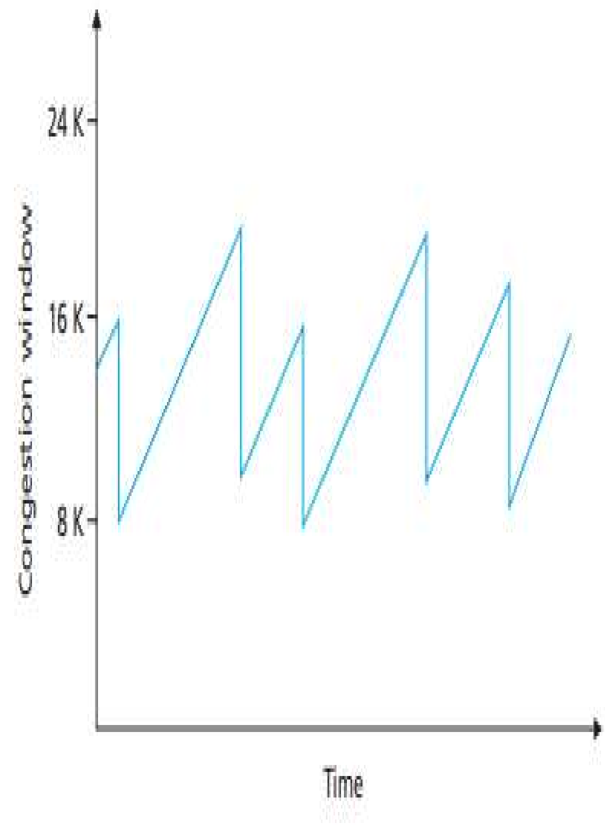


Figure 2.41 ♦ Additive-increase, multiplicative-decrease congestion control

James F Kurose and Keith W Ross, “Computer Networking: A Top - Down Approach”, Pearson Education; 6 th Edition (2017)

TCP Congestion Control

- The TCP congestion-control mechanism operating at the sender keeps track of an additional variable, the congestion window(cwnd).

` Fairness

- Consider K TCP connections, each with a different end-to-end path, but all passing through a bottleneck link with transmission rate R bps.

- A congestion-control mechanism is said to be fair if the average transmission rate of each connection is approximately R/K; that is, each connection gets an equal share of the link bandwidth.

Congestion Control

Congestion Control

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

Congestion Control

Open-Loop Congestion Control

- In open-loop congestion control, policies are applied to prevent congestion before it happens.
- In these mechanisms, congestion control is handled by either the source or the destination.

1. Retransmission Policy

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

Congestion Control

Open-Loop Congestion Control

2. Window Policy

- The type of window at the sender may also affect congestion.
- The Selective Repeat window is better than the Go-Back-N window for congestion control.
- In the Go-Back-N window, when the timer for a packet times out, several packets may be resent, although some may have arrived safe and sound at the receiver. This duplication may make the congestion worse.
- The Selective Repeat window, on the other hand, tries to send the specific packets that have been lost or corrupted.

Congestion Control

Open-Loop Congestion Control

3. Acknowledgment Policy

- The acknowledgment policy imposed by the receiver may also affect congestion.
- If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion.
- A receiver may send an acknowledgment only if it has a packet to be sent or a special timer expires.
- A receiver may decide to acknowledge only N packets at a time.
- We need to know that the acknowledgments are also part of the load in a network.
- Sending fewer acknowledgments means imposing less load on the network.

Congestion Control

Open-Loop Congestion Control

4. Discarding Policy

- A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission.
- For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented or alleviated.

Congestion Control

Open-Loop Congestion Control

5. Admission Policy

- An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual-circuit networks.
- Switches in a flow first check the resource requirement of a flow before admitting it to the network.
- A router can deny establishing a virtual-circuit connection if there is congestion in the network or if there is a possibility of future congestion.

Congestion Control

Closed-Loop Congestion Control

- Closed-loop congestion control mechanisms try to alleviate congestion after it happens.

1. Backpressure

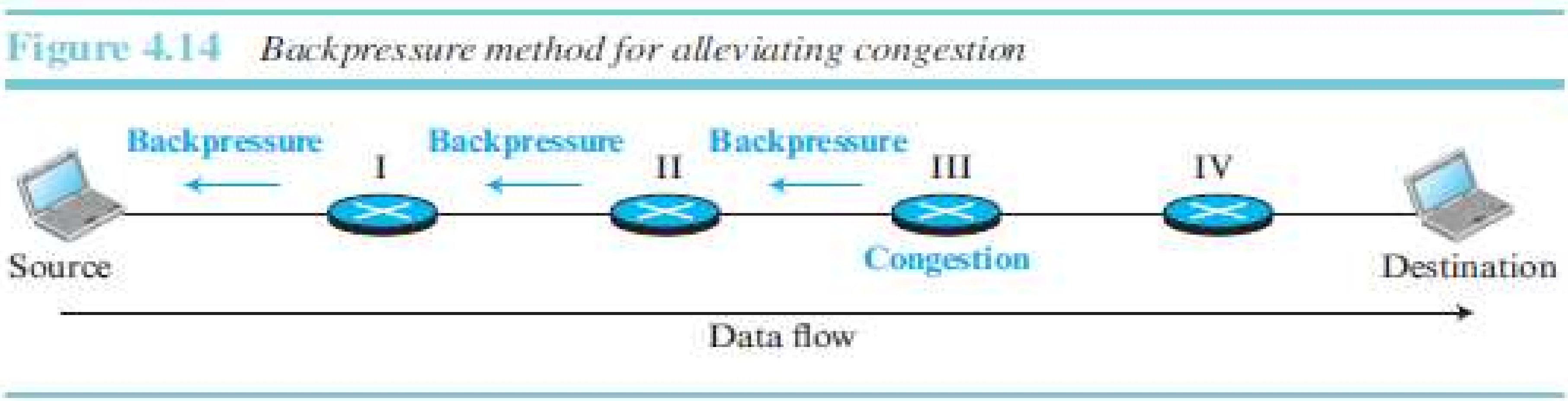
- The technique of backpressure refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes.
- This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream node or nodes, and so on.
- Backpressure is a node to node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source.
- The backpressure technique can be applied only to virtual circuit networks, in which each node knows the upstream node from which a flow of data is coming.

Congestion Control

Closed-Loop Congestion Control

1. Backpressure

[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]



- Node III in the figure has more input data than it can handle. It drops some packets in its input buffer and informs node II to slow down.
- Note that the pressure on node III is moved backward to the source to remove the congestion.
- The technique cannot be implemented in a datagram network, in which a node (router) does not have the slightest knowledge of the upstream router.

Congestion Control

Closed-Loop Congestion Control

2. Choke Packet

- A choke packet is a packet sent by a node to the source to inform it of congestion.
- In backpressure, the warning is from one node to its upstream node, although the warning may eventually reach the source station.
- In the choke-packet method, the warning is from the router, which has encountered congestion, directly to the source station.
- The intermediate nodes through which the packet has traveled are not warned. (Used in ICMP).

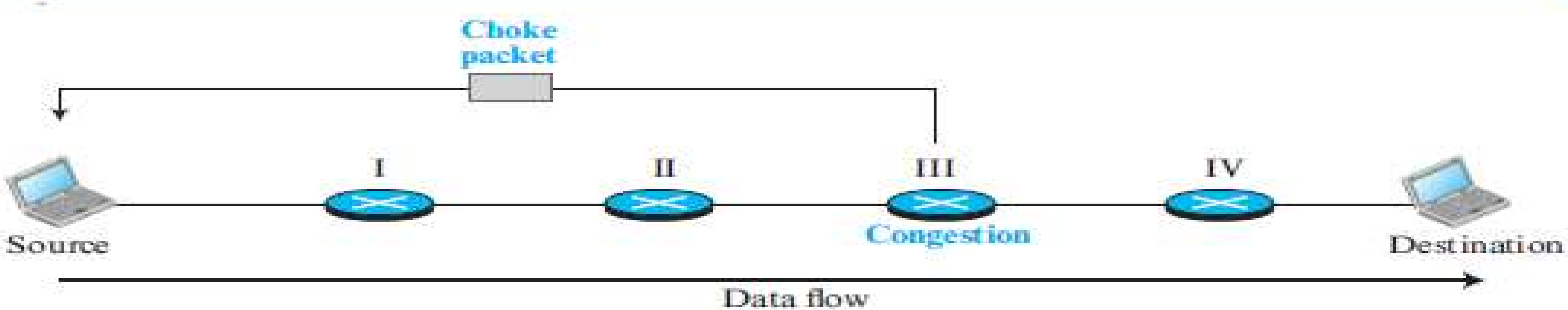
Congestion Control

Closed-Loop Congestion Control

2. Choke Packet

- When a router in the Internet is overwhelmed with IP datagrams, it may discard some of them, but it informs the source host, using a source quench ICMP message.
- The warning message goes directly to the source station; the intermediate routers do not take any action.

Figure 4.15 *Choke packet*



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Congestion Control

Closed-Loop Congestion Control

3. Implicit Signaling

- In implicit signaling, there is no communication between the congested node or nodes and the source.
- The source guesses that there is congestion somewhere in the network from other symptoms.
- For example, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested.
- The delay in receiving an acknowledgment is interpreted as congestion in the network; the source should slow down.
- We saw this type of signaling in TCP congestion control.

Congestion Control

Closed-Loop Congestion Control

4. Explicit Signaling

- The node that experiences congestion can explicitly send a signal to the source or destination.
- The explicit-signaling method is different from the choke-packet method.
- In the choke-packet method, a separate packet is used for this purpose; in the explicit-signaling method, the signal is included in the packets that carry data.
- Explicit signaling can occur in either the forward or the backward direction.
- This type of congestion control can be seen in an ATM network.