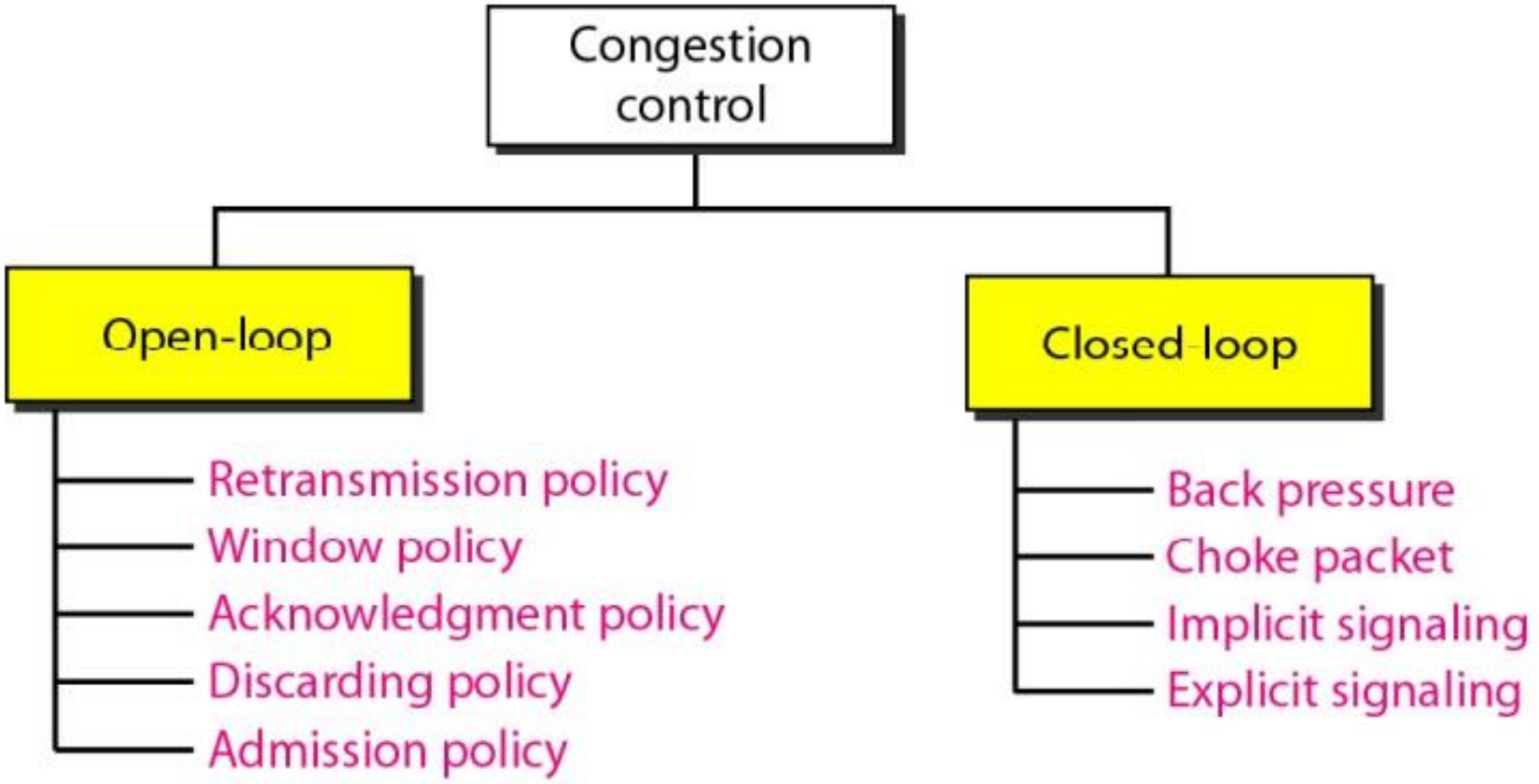
Introduction to Computer Networking

Dr. Mahmoud M. Elkhouly
www.elkhouly.net

Congestion control

- Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened. In general, we can divide 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

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.

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.

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.

Several approaches are used in this case.

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.

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.

Open-Loop Congestion Control

5. Admission Policy

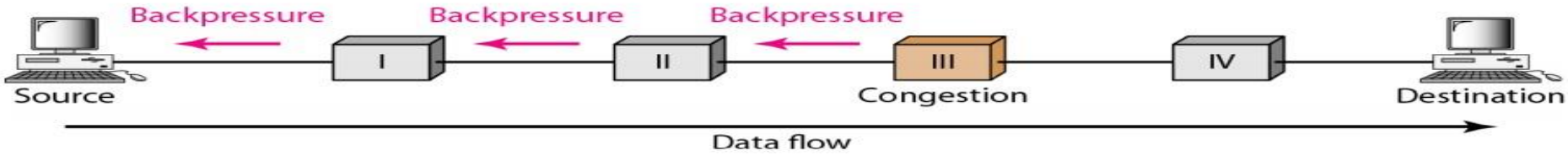
An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual circuit networks. Switches in a flow first check the resource requirement of a flow before admitting it to the network. A router can deny establishing a virtual circuit connection if there is congestion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

Closed-Loop Congestion Control

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

1. Back pressure

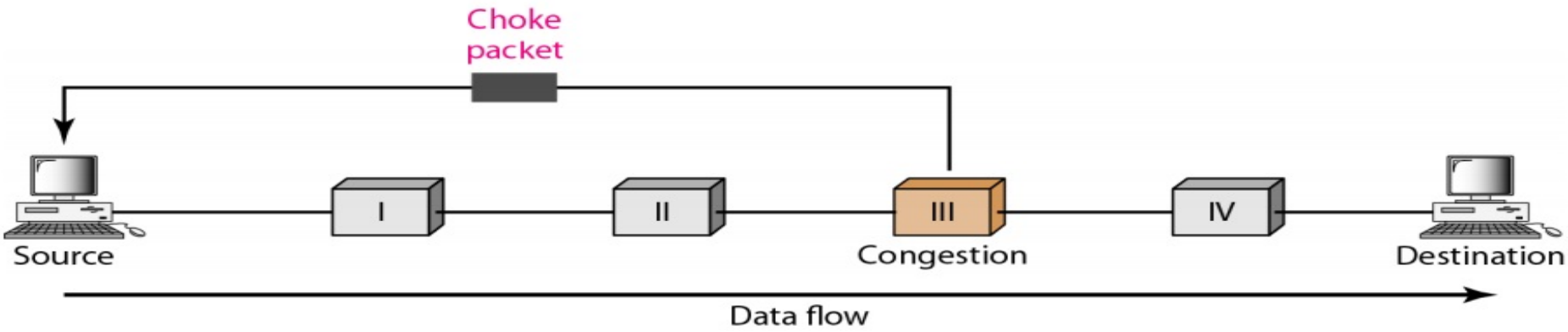
The technique of back pressure refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes. Back pressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source. The back pressure technique can be applied only to virtual circuit networks, in which each node knows the upstream node from which a flow of data is corning.



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 the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly. The intermediate nodes through which the packet has traveled are not warned. When a router in the Internet is overwhelm with IP datagrams, it may discard some of them; but it informs the source host, using a source quench ICMP message.



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 a 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

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, however, 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.

Closed-Loop Congestion Control

- **Backward Signaling**

A bit can be set in a packet moving in the direction opposite to the congestion. This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.

- **Forward Signaling**

A bit can be set in a packet moving in the direction of the congestion. This bit can warn the destination that there is congestion. The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion.

TCP Congestion Control

- Congestion occurs, if the load offered to any network is more than its capability.
- Once the connection is established, the sender can initialize the congestion window size (according to the available buffer space in the receiver).
- The sender's window size can be determined by the receiver and congestion in the network.
- The actual size of the window can be minimum of the receiver-advertised window size and the congestion window size.
- Generally, the methods followed by TCP to handle the congestion consists of three phases:

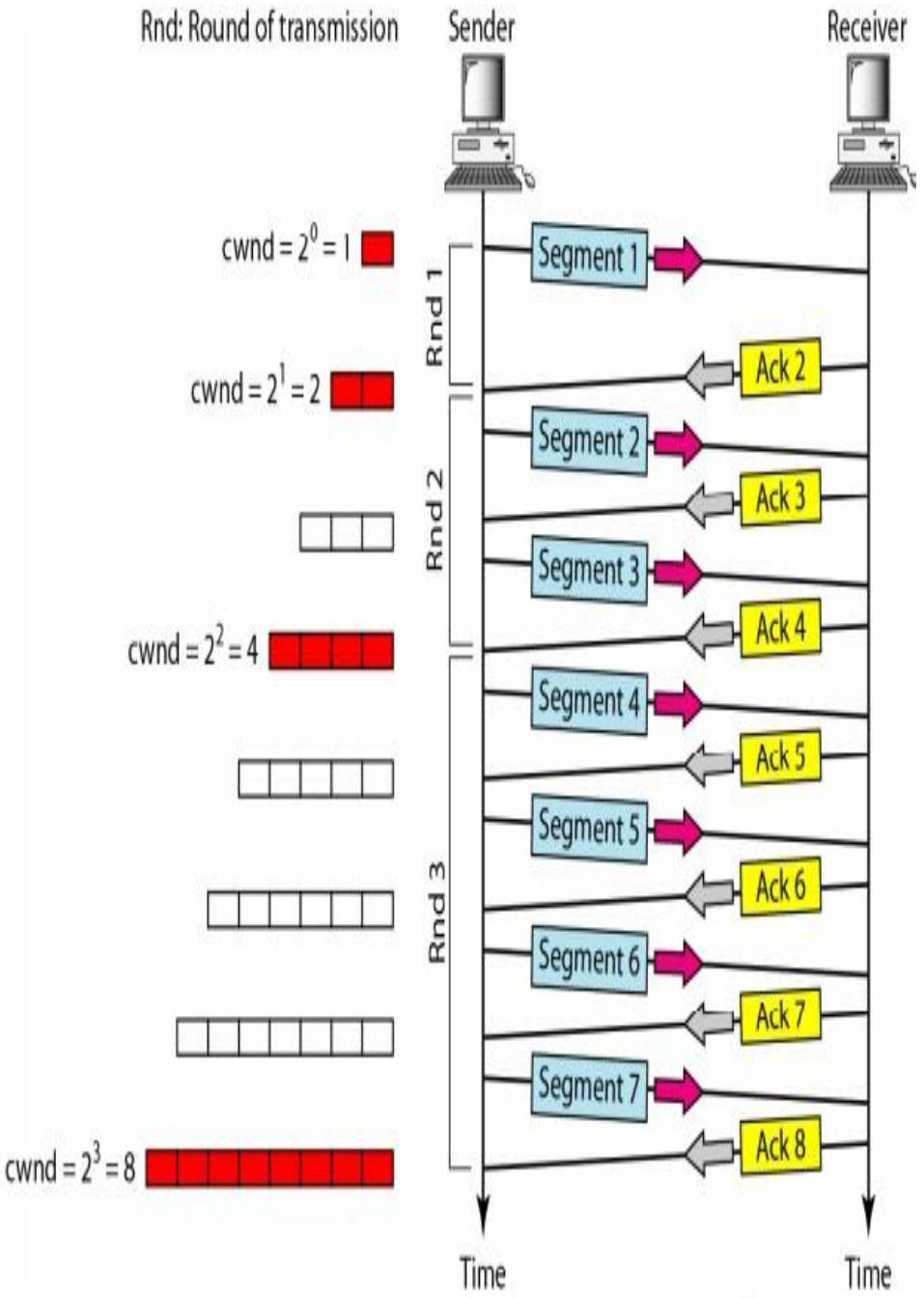
TCP Congestion Control

- **1. Slow start algorithm**

Slow start algorithm is used to control the congestion in TCP. In this algorithm, the size of window grows exponentially till the timeout occurs or the receiver's window is reached to the maximum threshold (64 kb = 65535 bytes).

Ex.

- Suppose that, the sender starts with **congestion window = 2^0 = 1 MSS** (Maximum Segment Size, each segment consists of 1byte). After receiving acknowledgement for segment 1, the size of window can be increased by one. Thus, the total size of congestion window = 2^1 = 2. When all the segments are acknowledged, the total size of congestion window = 2^3 = 8.



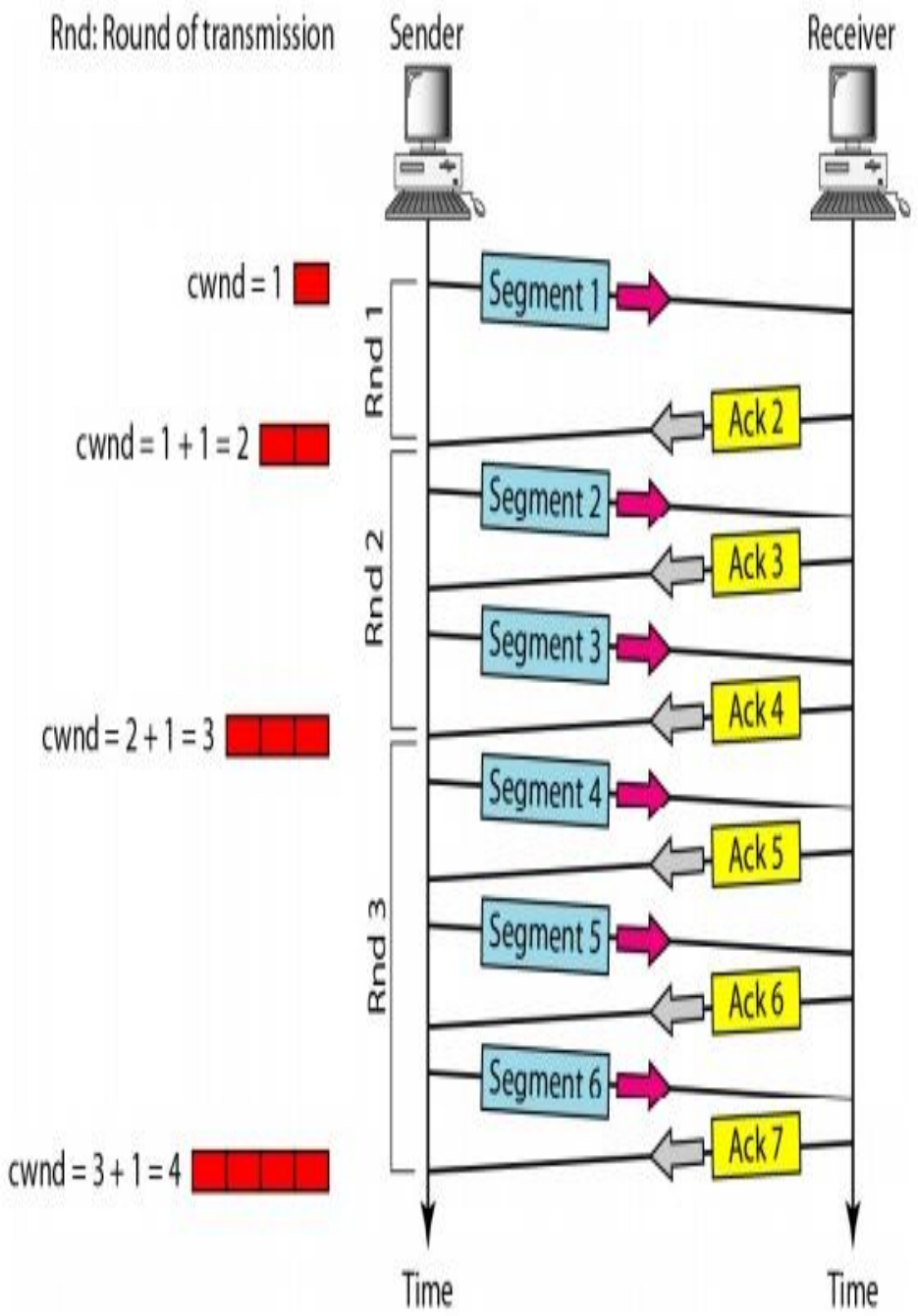
TCP Congestion Control

2. Congestion Avoidance Algorithm

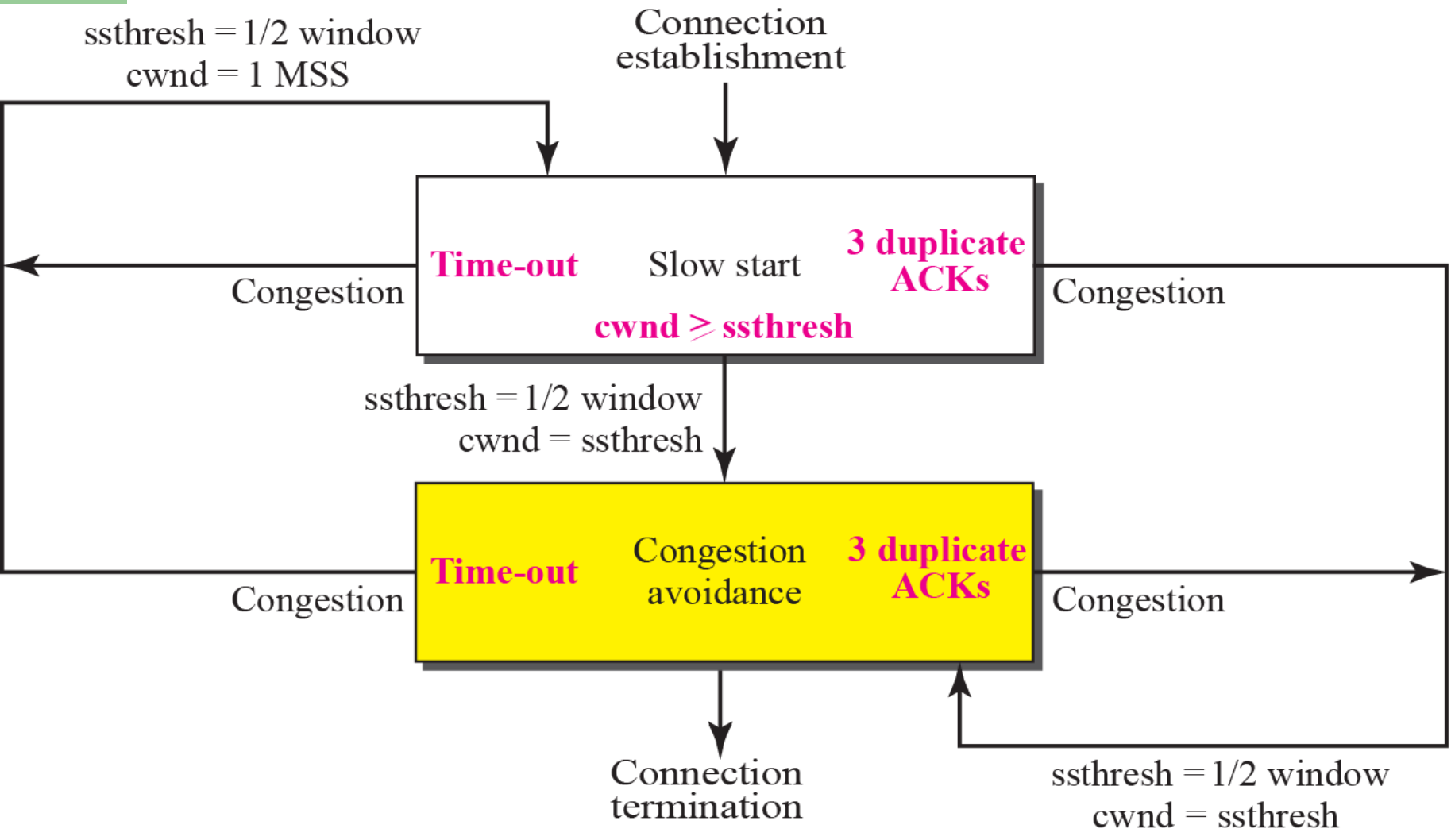
- In slow start algorithm, the size of the congestion window increases exponentially.
- To avoid the congestion before it happens, it is necessary to slow down the exponential growth.
- The Congestion avoidance algorithm works with an additive increase instead of the exponential growth.
- In this algorithm, after receiving each acknowledgement receipt (for one round), the size of the congestion window can be increased by one.

Ex.

- Start with, Congestion window = 1
- **By adding 1,** congestion window = 1 + 1 = 2



TCP Congestion Policy Summary



TCP Congestion Control

3. Congestion detection algorithm

- If the congestion occurs, it is required to decrease the congestion window size.
- There are two conditions in which congestion may occur: the connection time out and the reception of three acknowledgement.
- In both cases, the threshold size can be dropped to **one half of the previous window size.**

Ex.

- Assume that the window size = 64 and threshold is set to 32.
- In slow start algorithm, the window size starts from window size = 1, and then grows exponentially.
- Thereafter, when it reaches to threshold value, the congestion avoidance method allows the window size to increase linearly (by adding 1).
- However, the connection time-out occurs and the window size = 36. At this phase, the multiplicative decrease method is applied to the window size 36/2 = 18 (new threshold).
- Again, TCP starts working with slow start algorithm and the transmission is carried out.

Congestion Detection : Multiplicative Decrease

So, TCP assumes there is congestion if it detects a packet loss

- A TCP sender can detect lost packets via:
  - Timeout of a retransmission timer
  - Receipt of a 3 or more duplicate ACK

- TCP interprets a Timeout as a binary congestion signal. When a timeout occurs, the sender performs:

– cwnd is reset to one:

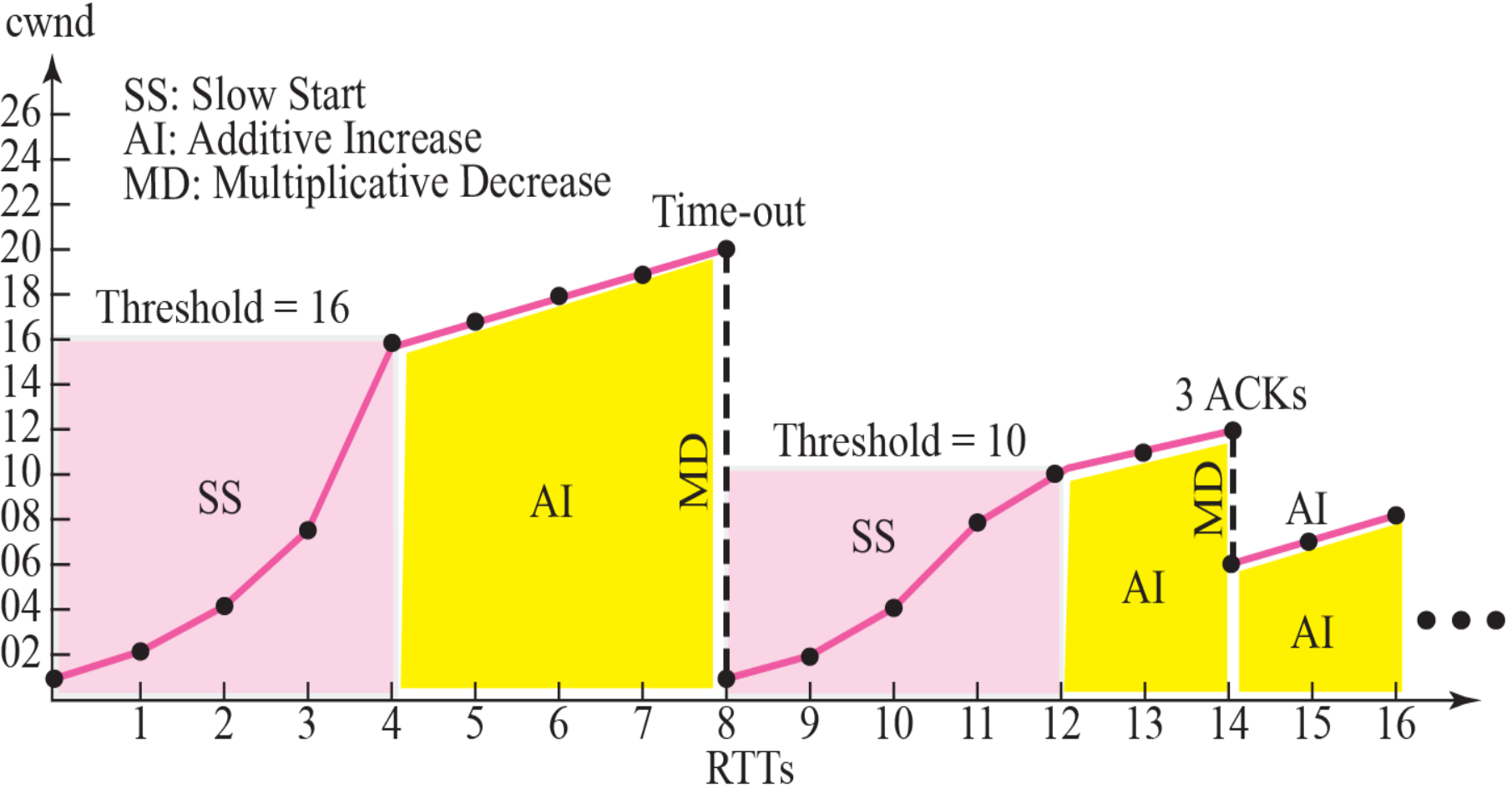
$$cwnd = 1$$

– ssthresh is set to half the current size of the congestion window:

$$ssthressh = cwnd / 2$$

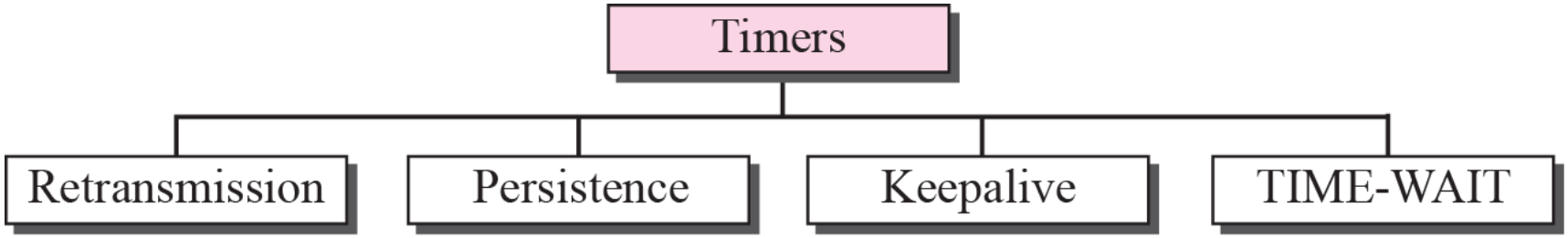
– and slow-start is entered

Congestion Example



TCP Timers

To perform its operation smoothly, most TCP implementations uses at least four timers



TCP Timers

Round Trip Time(RTT)

- To calculate the retransmission(RTO), we first need to calculate the round-trip time(RTT)
- In TCP, there can be only one RTT measurement in progress at any time
- **Measured** RTT (RTT_M) : how long it takes to send a segment and receive an acknowledgment of it.

- **Smoothed** RTT (RTT_S) : Weighted average of RTT_M and previous RTT_S

TCP Timers

RTT Deviation (RTT_D)

Original → No Value

After first measurement → $RTT_D = RTT_M/2$

After any other measurement

$$\rightarrow RTT_D = (1-\beta)\, RTT_D + \beta \cdot | RTT_S - RTT_M |$$

- **The value of** β is also implementation dependent, but is it is usually is sent to ¼.

Retransmission Timeout (RTO)

Original → Initial Value

After any measurement → $RTO = RTT_S + 4\, RTT_D$

TCP Timers

Persistence Timer

- When acknowledgment with non-zero window size after zero window size is lost, to correct deadlock, TCP uses a persistence timer for each connection
- When the sending TCP receives an acknowledgment with a window size of zero, the persistence timer is started
- When persistence timer goes off, the sending TCP sends a special segment called a *probe*
- The probe alerts the receiving TCP that the acknowledgment was lost and should be resent.
- If a response is not received, the sender continues sending the probe segments and doubling, and resetting the value of the persistence timer until the value reaches a threshold (usually 60 seconds).
- After that sender sends one probe segment every 60s until the window is reopened.

TCP Timers

KeepaliveTimer

- Used to prevent a long idle connection between two TCPs.
- Each time the server hears from a client, it resets this timer.
- Time-out is usually 2 hours.
- After 2 hours, sending 10 probes to client (each 75 secs), then terminates connection.

TIME-WAIT Timer

- The time-wait timer is used during connection termination.