

CN (IT-3001)

Transport Layer: Congestion Control

Prof. Amit Jha

School of Electronics Engineering (SOEE)

KIIT Deemed to be University



Disclaimer: The contents in this slide have been referred from many sources which I do not claim as my own. Some of the content has been modified for easier understanding of the students without any malafide intention. This slide is only for educational purpose strictly, and not for the commercial purpose.

Congestion Control at the Transport Layers

- We discuss following protocols in the subsequent slides.
 1. User Datagram Protocol (UDP)
 2. Transmission Control Protocol (TCP)

Congestion

- An important issue in a packet-switched network is congestion.
- Congestion in a network may occur if;
 - the load on the network is greater than the *capacity* of the network.
- **Reason for Congestion:**
 - An accident during rush hour.
 - Routers and switches have queues buffers that hold the packets before and after processing.

Congestion vs Network Performance

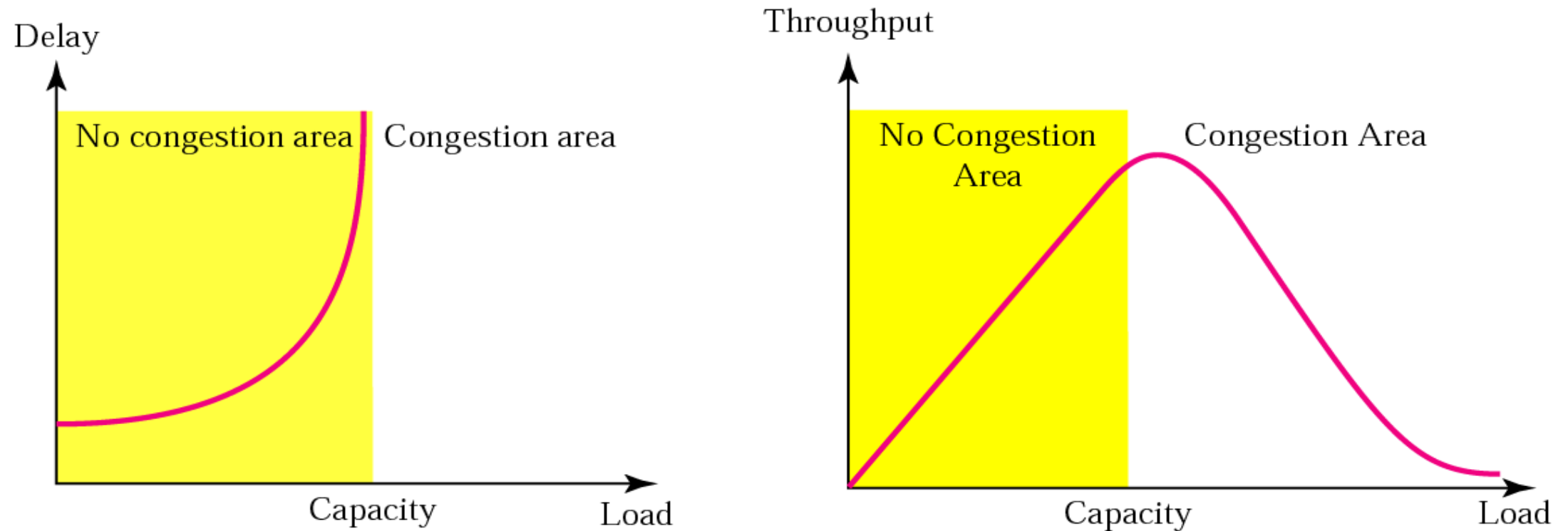


Fig: Packet delay and throughput as functions of load

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 other words, it refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.

Windows method/policy of congestion control in TCP

- ***Congestion Window:***

- While discussing flow control, we said that the sender window size is determined by the available buffer space in the receiver (*rwnd*).
- However, We totally ignored another entity here-the network. If the network can not deliver the data as fast as they are created by the sender, it must tell the sender to slow down.
- Today, the sender's window size is determined not only by the receiver but also by congestion in the network.
- The sender has two pieces of information: the receiver-advertised window size (*rnwnd*) and the congestion window size (*cnwnd*).
- The actual size of the window is the minimum of these two.

Actual window Size = minimum (*rwnd*, *cnwnd*)

Congestion Detection

- In TCP, congestion is detected on the occurrence of following two events:
 1. Time-out
 2. Three duplicate ACKs received
- The congestion due to time-out is more severe as compared the congestion due to three duplicate ACKs.
- The reception of three duplicate ACKs implies either network is less congested or has recovered from congestion.

Congestion Handling Policy

- TCP's general policy for handling congestion is based on following three algorithms:
 1. Slow-start (SS)
 2. Congestion Avoidance (CA), and
 3. Fast Recovery (FR)
- In the slow-start phase,
 - the sender starts with a very slow rate of transmission, but increases the rate rapidly to reach a threshold.
- When the threshold is reached,
 - the data rate is reduced to avoid congestion.
- Finally if congestion is detected, the sender goes back to the slow-start or congestion avoidance phase based on how the congestion is detected.

1. *Slow Start: Exponential Increase* (SS:EI)

- It starts with congestion window size ($cwnd$) = one maximum segment size (MSS) i.e., $cwnd = 1 \text{ MSS}$.
- The size of $cwnd$ increases by one after every ACK received until it reaches a threshold, called *slow-start threshold* ($ssthresh$).
- In this, the growth is exponential after every round.

The size of the $cwnd$ and ACK are function of each other as shown below:

- If an ACK arrives then, $cwnd = cwnd + 1$

Illustration of *Slow Start: Exponential Growth*

Note: **Slow Start:** *cwnd* increases by 1 MSS

Note: *Growth is exponential after every round, (red colour box)*


Start: $cwnd = 2^0 = 1$


After 1st round: $cwnd = 2^1 = 2$


After 2nd round: $cwnd = 2^2 = 4$

After 3rd round: $cwnd = 2^3 = 8$

Rnd: Round of transmission

$cwnd = 2^0 = 1$ 

$cwnd = 2^1 = 2$ 

$cwnd = 2^2 = 4$ 

$cwnd = 2^3 = 8$ 

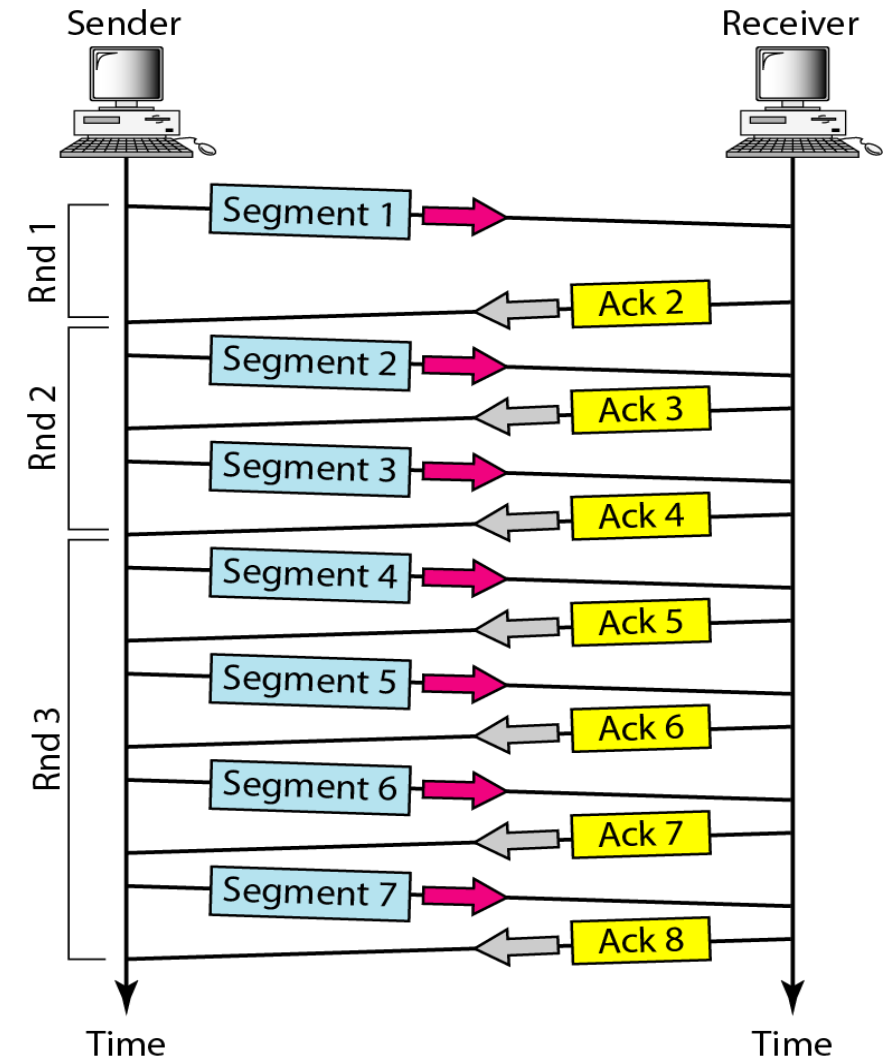


Fig: Slow-Strat and Exponential growth

2. Congestion Avoidance: Additive Increase (CA:AI)

- When the size of the congestion window reaches the slow-start threshold, the slow-start phase stops and the additive phase i.e., CA phase begins.
- In this, each time the whole window of segments is acknowledged (**one round**), the size of the congestion window is increased by 1.

The size of the *cwnd* and ACK are function of each other as shown below:

- If an ACK arrives then, $cwnd = cwnd + \frac{1}{cwnd}$

Illustration of Congestion Avoidance: Additive Increase


Rnd: Round of transmission


Sender


Receiver

Note: Congestion avoidance algorithm

usually starts when the
size of the window is
much greater than 1

$cwnd = 1$ 

$cwnd = 1 + 1 = 2$ 


$cwnd = 2 + 1 = 3$ 

Start: $cwnd = 1$

After 1st round: $cwnd = 2$

After 2nd round: $cwnd = 3$

After 3rd round: $cwnd = 4$

$cwnd = 3 + 1 = 4$ 

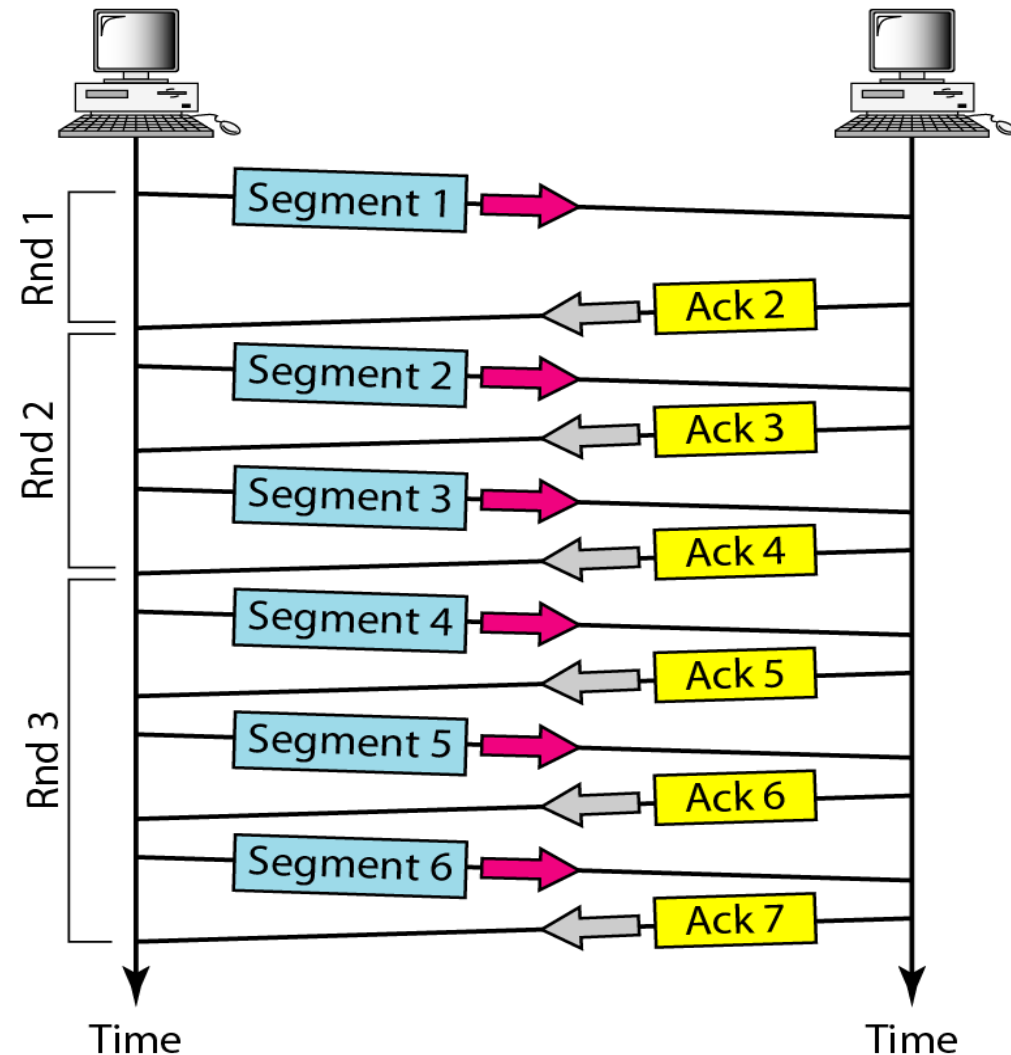


Fig: Congestion avoidance, additive increase

3. *Fast Recovery (FR)*

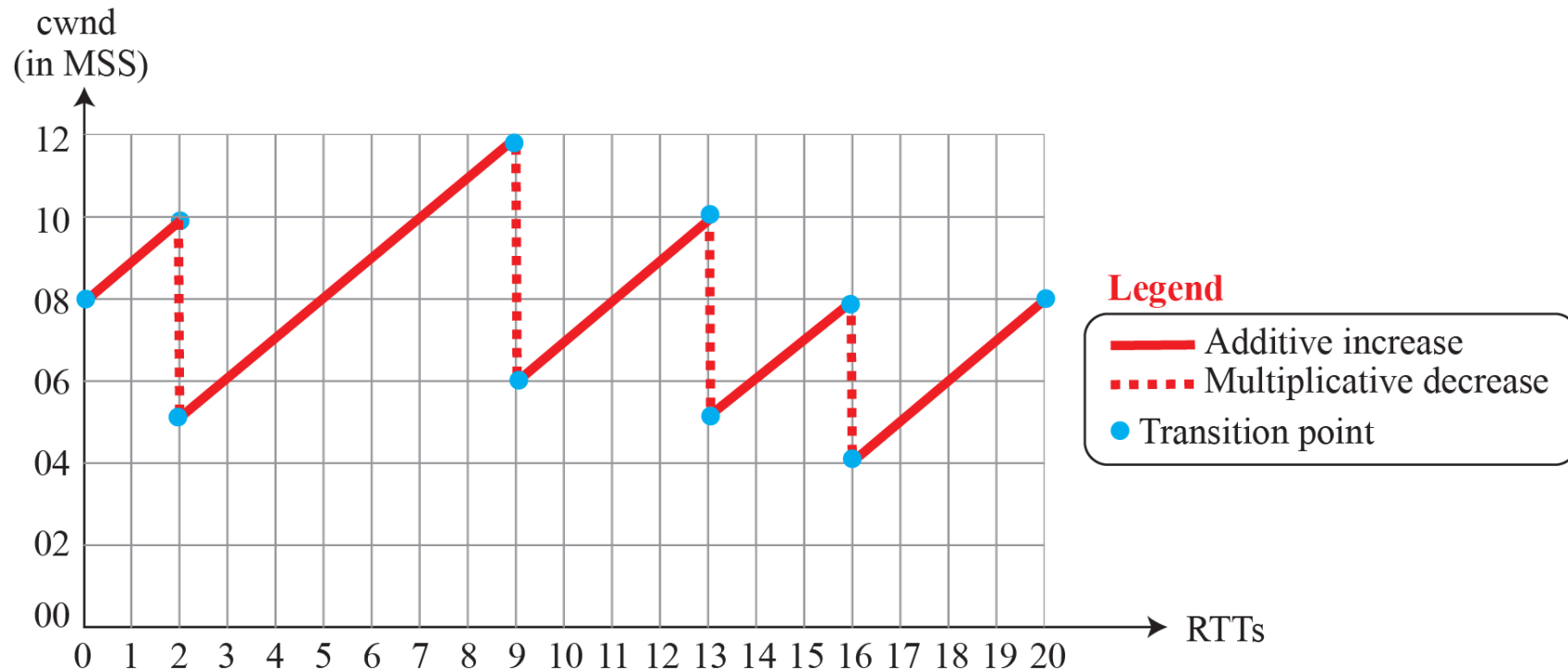
- The fast recovery algorithm is optional in TCP.
- It **starts when three duplicate ACKs arrives** → which is the interpretation of the light congestion in the network.
- It sets $ssthresh = \frac{cwnd}{2}$. And $cwnd = ssthresh + 3$
- Like congestion avoidance, this algorithm is also an additive increase, but it increases the size of the congestion window when **a duplicate** ACK arrives (after three duplicate ACKs that trigger the use of this algorithm).
- When TCP is in FR state, and if
 - **Time-out occurs** → it moves to **SS** phase and follows the rule of **SS** phase
 - **A new ACK** arrives → it moves to **CA** phase.
 - **A duplicate ACK** arrives → it remains in **FR** phase

The size of the *cwnd* and ACK are function of each other as shown below:

- If a **duplicate** ACK arrives then, $cwnd = cwnd + \frac{1}{cwnd}$

Additive Increase and Multiplicative Decrease (AIMD)

- In long TCP connection, if we ignore the slow start phase, then
 - When a congestion is detected, the *cwnd* becomes **half** of the current *ssthresh*. → This is called **Multiplicative Decrease (MD)**.
 - And congestion avoidance phase starts. → this is called **Additive Increase (AI)**.
- The AIMD phase is shown in the diagram below which is observed like a **saw tooth** pattern.



- Now, the question comes?
 1. When each of these policies will be used?
 2. When TCP moves from one policy to another?

Answers:

- The use of different algorithm and the way to move from one policy to another is implemented differently in following three versions of TCP.
- This is also referred to as **Policy Transition** in TCP.
- Three version of TCP are:
 1. **Tahoe TCP**
 2. **Reno TCP**
 3. **New Reno TCP**

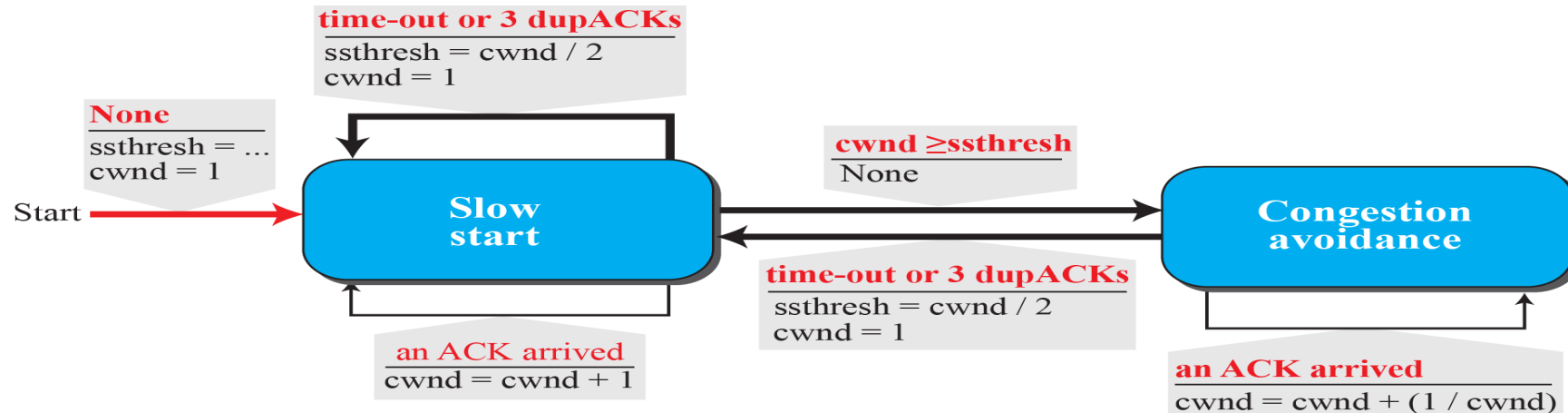


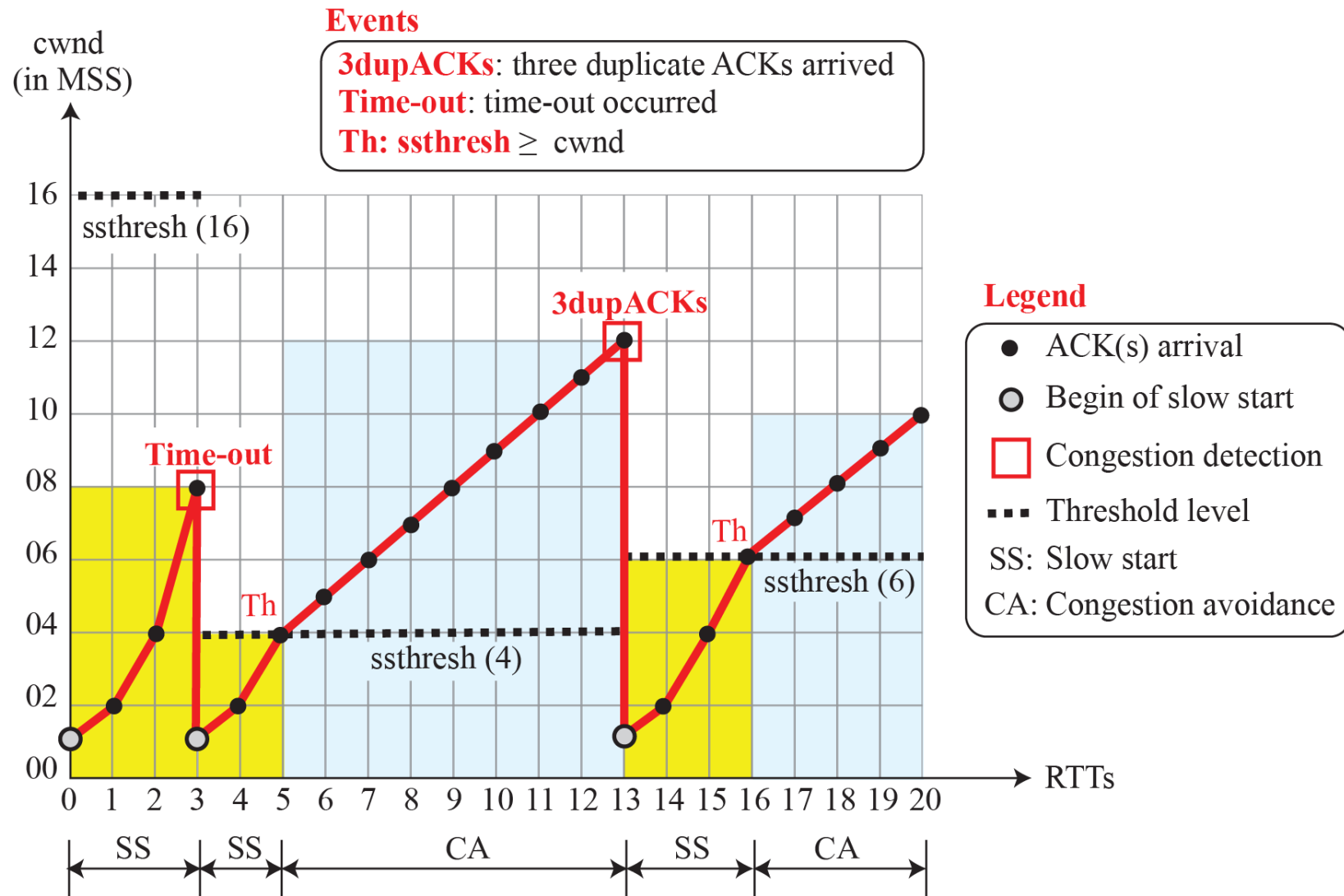
Tahoe TCP

- This is the first way to implementation of congestion control mechanism in TCP.
- It uses only two algorithms (policies)
 1. *Slow-Start* and
 2. *Congestion Avoidance*
- If the congestion is detected due to
 - time-out → *Slow-Start* algorithm starts and TCP follows the rule of SS
 - three duplicate ACKs received → *Slow-Start* algorithm starts and TCP follows the rule of SS

Note:- It treats both the congestion in the same way.

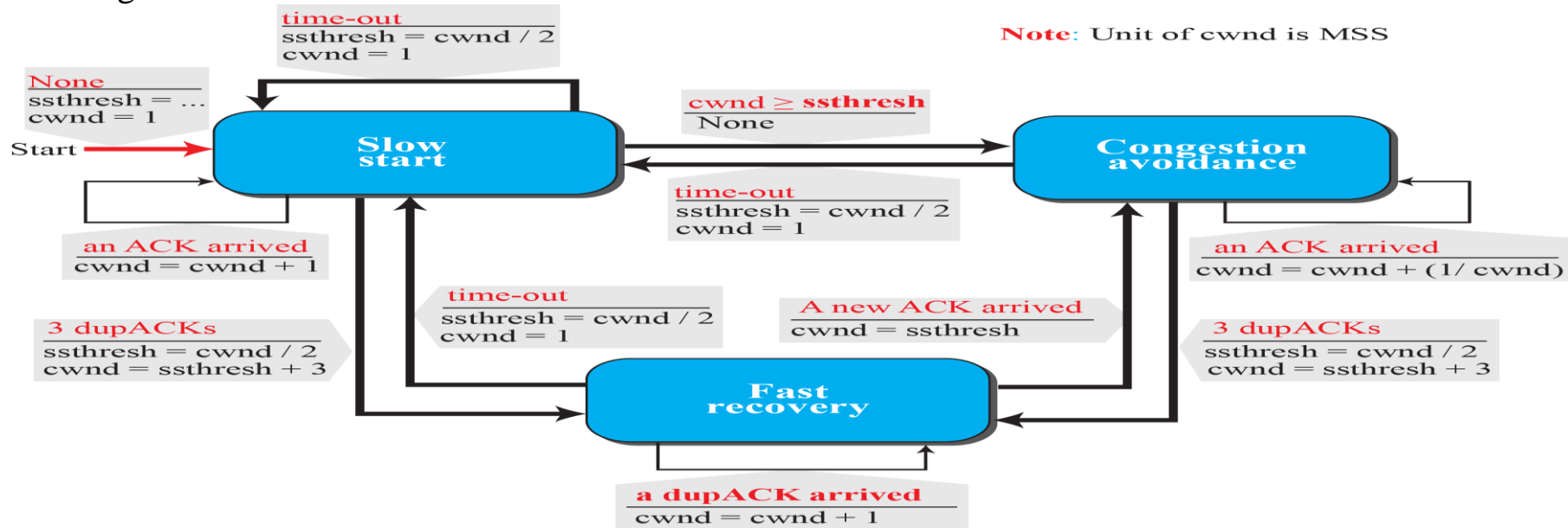
- The FSM diagram for Tahoe TCP is shown below.

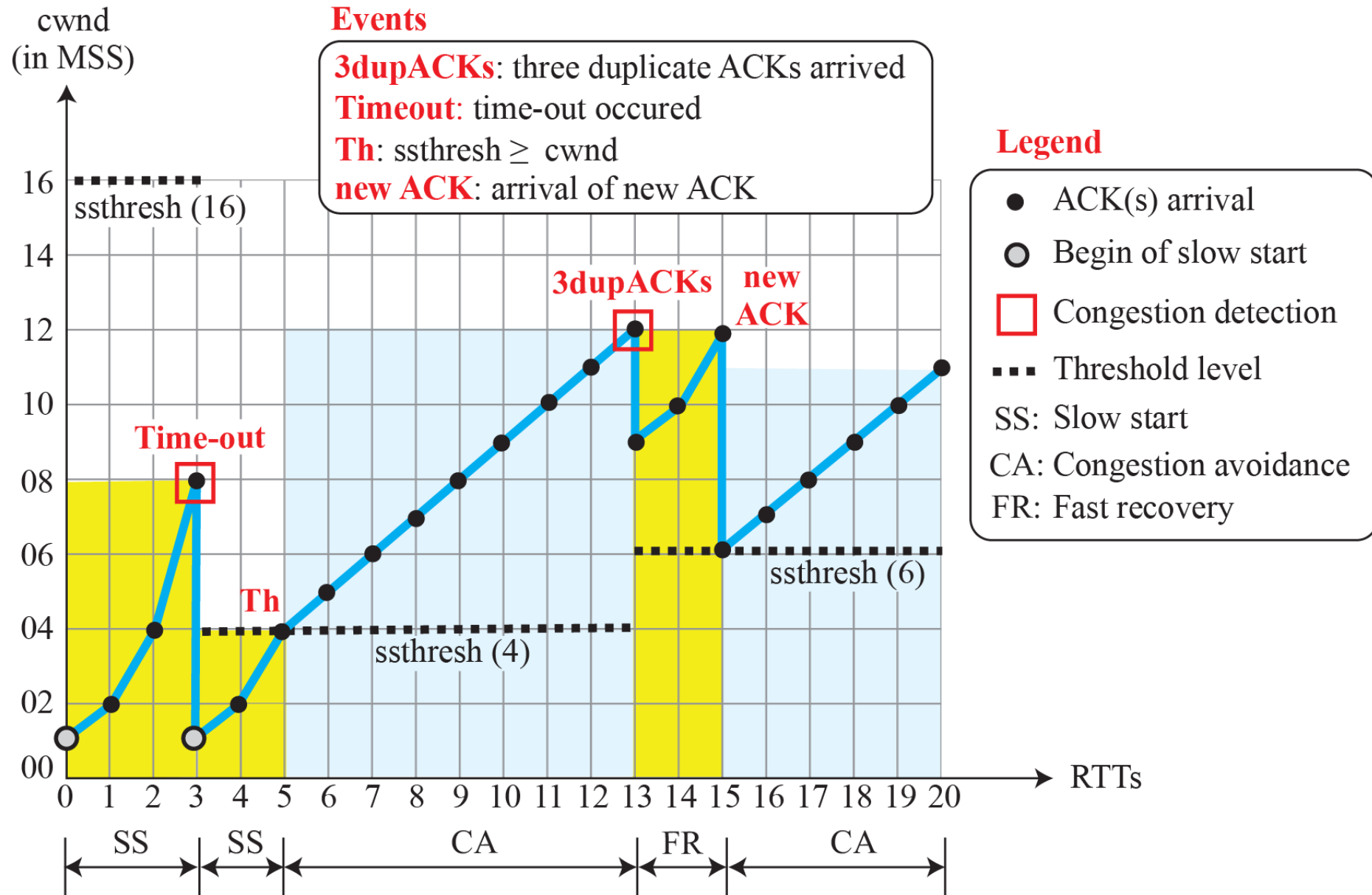




Reno TCP

- This version of TCP handles the two signals of congestion differently.
- It uses all three algorithms (policies)
 1. *Slow-Start*
 2. *Congestion Avoidance*
 3. *Fast Recovery*
- If the congestion is detected due to
 - time-out → *Slow-Start* algorithm starts and TCP follows the rule of SS phase
 - three duplicate ACKs received → *Fast Recovery* algorithm starts and TCP follows the rule of FR phase
- The FSM diagram for Reno TCP is shown below.



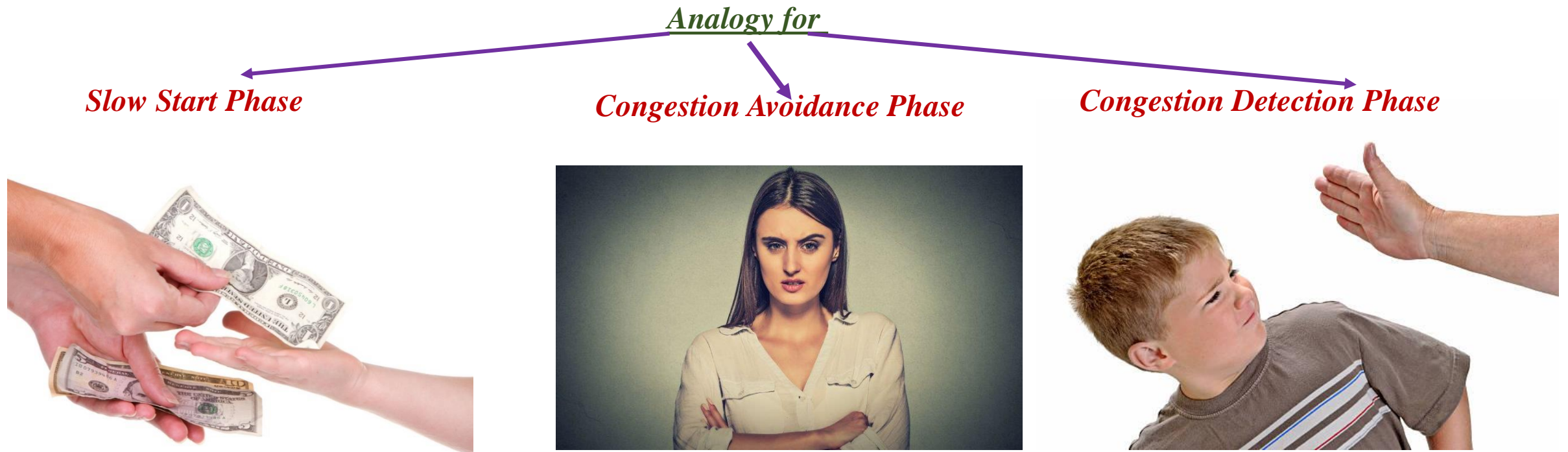


New Reno TCP

- This version made an extra optimization of existing Reno TCP. For more detail, refer Page no -221 of the text book.

To understand Slow start, CA, and Congestion Detection and its effect through the graph..

- Imagine the example covered in the class of **Mummy** and **You**..



Disclaimer:- The images are taken from the Google for explaining the concept to students. Its intention is not to hurt sentiments of any students or person portrayed in this.

Summary Table for different algorithms

Condition	Size of the <i>ssthresh</i> when the corresponding phase starts	Size of the <i>cwnd</i> when the corresponding phase starts	Growth of <i>Cwnd</i>
<i>Slow Start</i>	$ssthresh = \frac{cwnd}{2}$	$cwnd = 1$	$cwnd = cwnd + 1$ When an ACK arrives
<i>Congestion Avoidance</i>	$ssthresh = \frac{cwnd}{2}$	$cwnd = ssthresh$	$cwnd = cwnd + \frac{1}{cwnd}$ When an ACK arrives
<i>Fast Recovery</i>	$ssthresh = \frac{cwnd}{2}$	$cwnd = ssthresh + 3$	$cwnd = cwnd + \frac{1}{cwnd}$ When a Duplicate ACK arrives

Summary Table for different policy transition methods

Policy transition Method	Reason of Congestion	Algorithm to be Adopted
<i>Taho TCP</i>	<i>Time – Out</i>	<i>SS</i>
	<i>Three Duplicate ACK</i>	<i>SS</i>
<i>Reno TCP</i>	<i>Time – Out</i>	<i>SS</i>
	<i>Three Duplicate ACK</i>	<i>Fast Recovery</i>

Homework

- 1) Explain the window methods of congestion control implementation in Tahoe TCP with the help of graph plotted for '*rounds*' on x axis against '*cwnd*' on y axis. Assume, the maximum window size is 32 segments. Show the regions like Slow-Start (SS), Additive Increase (AI), Multiplicative Decrease (MD) explicitly in the graph for the following situations.
 - A congestion is detected due to time out when *cwnd* reaches to 20.
 - Further, the congestion is detected due to reception of three ACKs when *cwnd* reaches to 12.
- 2) Repeat above homework using Reno TCP.

End of Module-3

You are advised to go through the chapter 3 of the text book as mentioned in the class for more clarity and details.