ECE-647
TCP: Transmission Control Protocol in the Internet
Part II

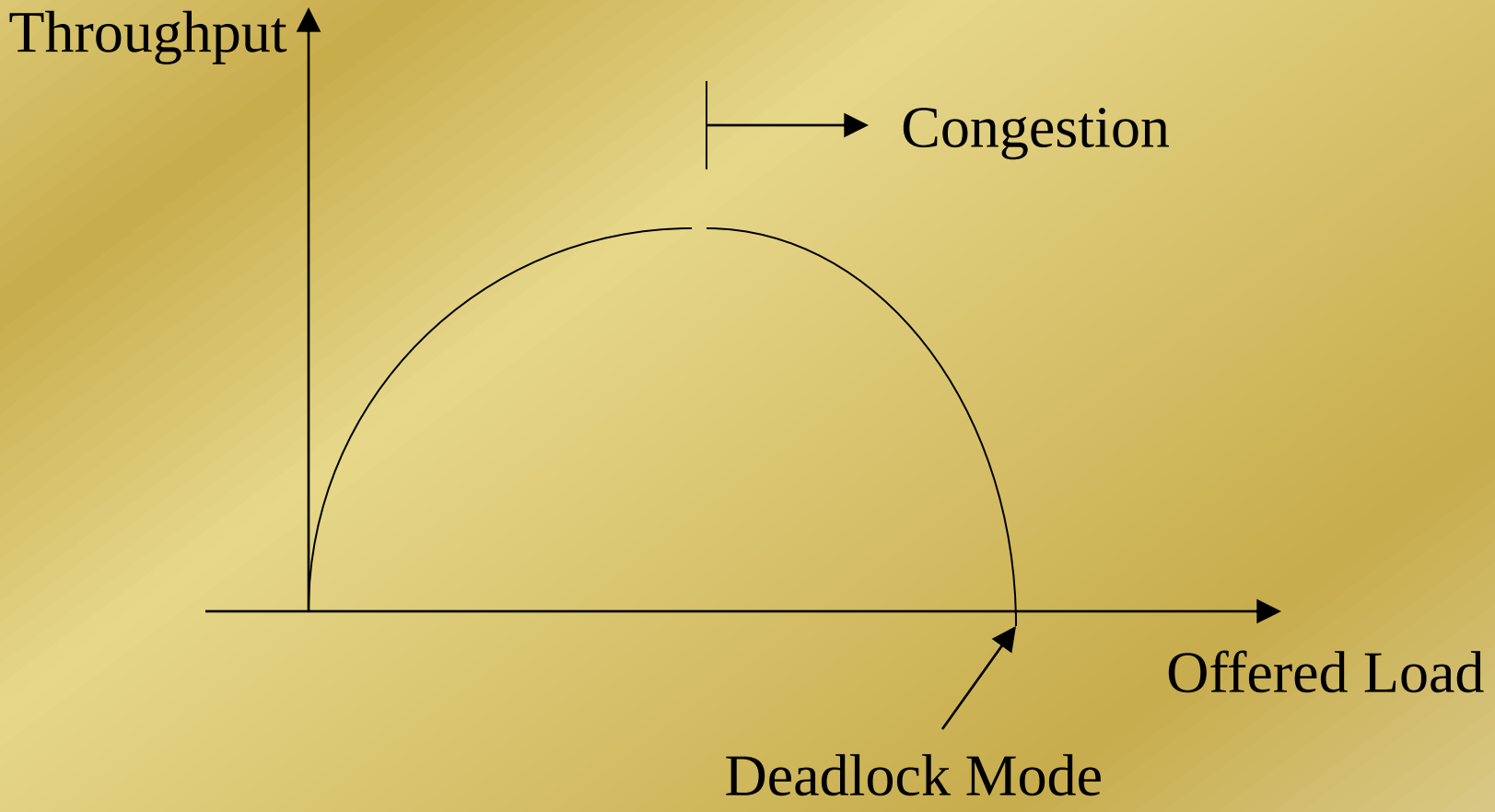
Instructor: Xiaojun Lin

Purdue University

Window Size

- Window size determines the number of packets that can be sent before waiting for an ack
- Large window size leads to higher rate
- However, the network may become congested.

Congestion Collapse



Flow Control/Congestion Control

- Two mechanisms in TCP that control the window size (and hence the flow of information):
  - Flow Control mechanism (awnd)
  - Congestion Control mechanism (cwnd)
- Effective window = min(awnd, cwnd)

Flow Control in TCP

- "Flow Control" mechanism is simple.
  - Receiver uses a *window size* field in the ack to advertise the size of the window *awnd*, which reflects its buffer capacity.
  - An inadequate receiver buffer size may constrain the throughput of the connection regardless of the state of the network.
- Congestion Control mechanism is more complex.

Congestion Control in TCP

- There are several details:
- Initially – there is a multiplicative increase of the window size (slow start)
- Normal operation: AIMD – additive increase and multiplicative decrease of the window size (congestion avoidance phase).

TCP congestion control:

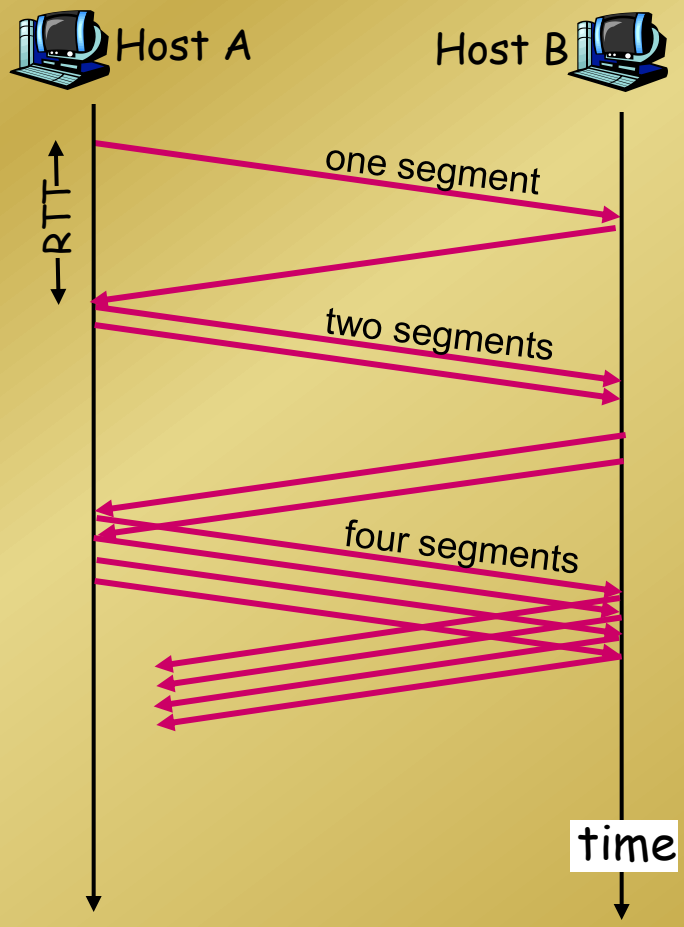
- "probing" for usable bandwidth:
  - ideally: transmit as fast as possible (**`Cwnd`** as large as possible) without loss
  - *increase* **`Cwnd`** until loss (congestion)
  - loss: *decrease* **`Cwnd`**, then begin probing (increasing) again

- two "phases"
  - slow start
  - congestion avoidance
- important variables:
  - **`Cwnd`**
  - **`ssthresh:`** defines threshold between the two phases (i.e., between the slow start phase and the congestion avoidance phase)

Slow Start

- initially, congestion window size cwnd = 1 MSS (maximum segment size)
- increment window size by 1 MSS on each new ack
- slow start phase ends when window size reaches (or exceeds) the slow-start threshold
  - Or when packet loss occurs

TCP Slowstart



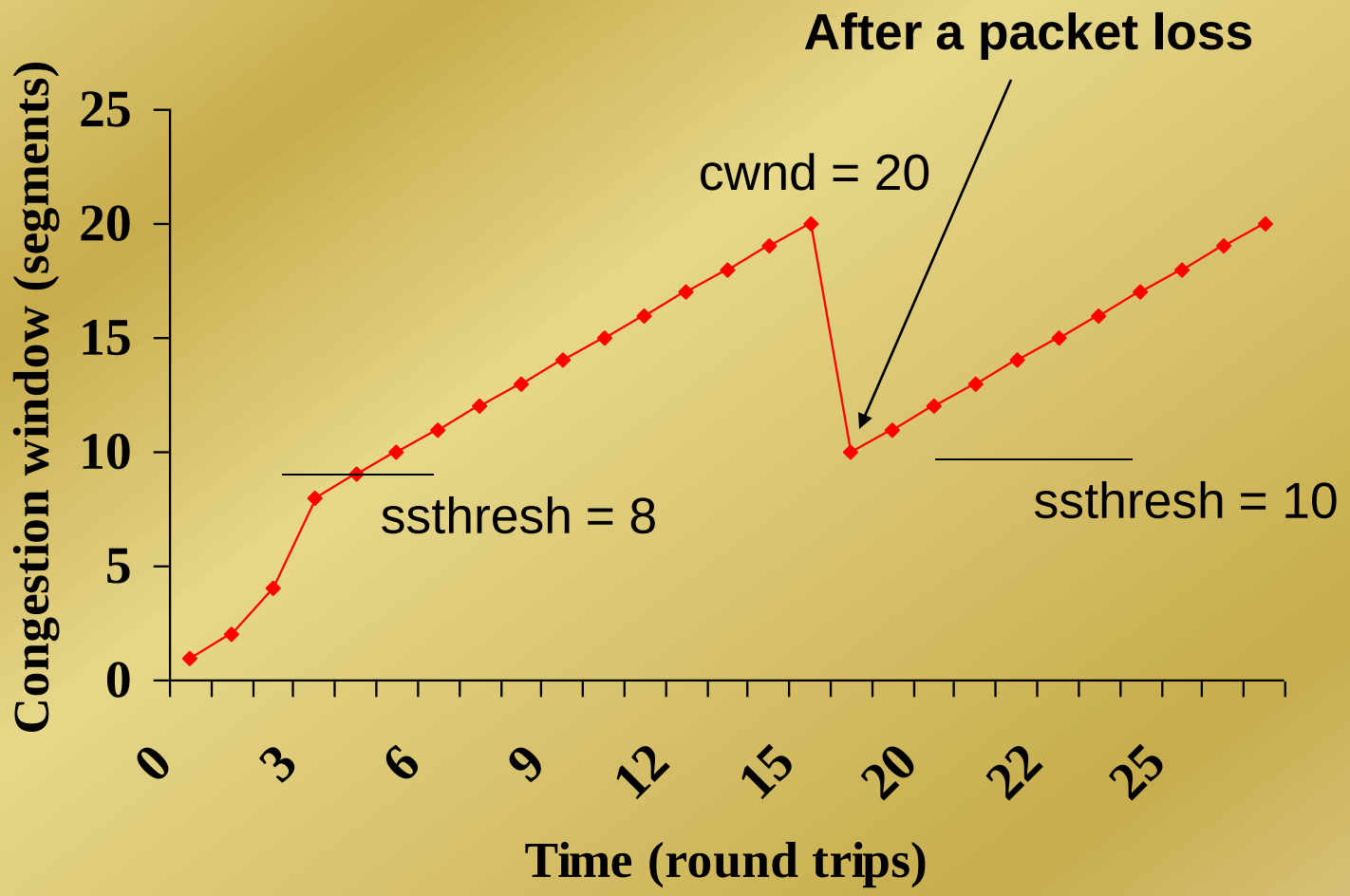
"Slow Start" is not slow!

- cwnd grows **exponentially** with time during slow start
  - factor of 1.5 per RTT if every other segment ack'd
  - factor of 2 per RTT if every segment ack'd
  - Could be less if sender does not always have data to send

Congestion Avoidance

- On each new ack, increase cwnd by 1/cwnd packets
- cwnd increases **linearly** with time during congestion avoidance
  - 1/2 MSS per RTT if every other segment ack'd
  - 1 MSS per RTT if every segment ack'd
- When there is a packet loss, reduce both cwnd and ssthreshold to cwnd/2

TCP Congestion Control



AIMD

TCP congestion avoidance:

- AIMD: *additive increase, multiplicative decrease*
  - increase window by 1 per RTT
  - decrease window by factor of 2 on loss event
- More conservative than slow-start
- Objectives:
  - High utilization
  - Avoid the onset of the congestion
  - Fairness

TCP Fairness

Fairness goal: if N TCP sessions share same bottleneck link, each should get 1/N of link capacity

- Suppose the window size of two flows are x and y, respectively
  - Each additive increase does not change the difference in window size
  - Each multiplicative decrease reduce the difference by a factor of 2