

## TCP Congestion Control

These slides are created by Dr. Yih Huang of George Mason University. Students registered in Dr. Huang's courses at GMU can make a single machine-readable copy and print a single copy of each slide for their own reference, so long as each slide contains the copyright statement, and GMU facilities are not used to produce paper copies. Permission for any other use, either in machine-readable or printed form, must be obtained from the author in writing.

## Network Congestion

- ❑ When the network is overloaded, packets are buffered at routers.
- ❑ If the situation persists, routers run out of buffer space and have no choice but to drop some packets.
- ❑ In the Internet, it is TCP modules at sources that reduce traffic in response to congestion.

## Basic Ideas

- ❑ The assumption is that packet loss caused by transmission errors is rare and thus packet losses signify congestion.
- ❑ Sources do not really see packet losses. They detect “signs” of losses, called **packet loss indications**.
  - Timeouts
  - Triple duplicates

CS 756

3

## Controlling Bandwidth Consumption

- ❑ With sliding window protocols,
  - A large window allows a sender to be aggressive in transmission
  - a small one forces it to stop-and-wait frequently and thus curbs traffic volume (bytes per sec).
- ❑ Thus, window size is our major concern.

CS 756

4

## Slow Start

- ❑ The sender maintains an integer variable, **cwnd** (congestion window).
- ❑ When a connection is established,  $cwnd = \text{segsize}$  (segment size).
- ❑ The sender transmits up to the minimum of  $cwnd$  and **rwnd**, the window size advertised by receiver.
- ❑ Each time an ACK is received,  $cwnd += \text{segsize}$ .

CS 756

5

## How Slow Is Slow Start ?

CS 756

6

## Handling Congestion

- ❑ In the face of congestion, the sender reduces traffic by setting `cwnd` to one segment.
- ❑ It should then probe network capacity again.
- ❑ However, since we already experienced congestion, probing by slow start may be too aggressive.
- ❑ The solution is to allow `cwnd` to grow exponentially halfway and linearly afterwards.
- ❑ The “halfway” point is recorded in variable `ssthresh`, standing for slow start threshold.

CS 756

7

## Computing CWND

- ❑ Upon connection establishment:
  - `cwnd = segsize`
  - `ssthresh = 65535`
- ❑ Un-acked bytes  $\leq \min(\text{cwnd}, \text{rwnd})$ .
- ❑ When a timeout occurs,
  - `ssthresh = max{cwnd/2, segsize*2}`
  - `cwnd = segsize`
- ❑ When a new Ack is received,
  - If `cwnd ≤ ssthresh`, `cwnd += segsize`
  - Otherwise, `cwnd += segsize*segsize/cwnd`

CS 756

8

## Discussion

- Please note how the algorithm *approximates* linear growth by

$\text{cwnd} += \text{segsz} * \text{segsz} / \text{cwnd}$

– Assuming  $\text{segsz}=1000$  and initial  $\text{cwnd}=1000$ .

– In the first RTT, one segment is sent:

$1000 += 1000 * 1000 / 1000 \Rightarrow 2000$

– In the second round, two segments are sent:

$2000 += 1000 * 1000 / 2000 \Rightarrow 2500$

$2500 += 1000 * 1000 / 2500 \Rightarrow 2900$

- Roughly one  $\text{segsz}$  added to  $\text{cwnd}$  per RTT

CS 756

9

## TCP Acknowledgments

- TCP acknowledges the next byte expected (not the last one received).
- $\text{Ack}=x$  indicates *all* bytes up to and including  $x-1$  has been successfully received.

CS 756

10

## Exercise

- ❑ Assume that all bytes up to 999 have been received and acked. Give the acks in response to segments
  - Seg(1000, 200)
  - Seg(1300, 100)
  - Seg(1400, 300)
  - Seg(1700, 100)
  - Seg(1200, 100)

CS 756

11

## Fast Retransmission

- ❑ When the receiver receive segments  $x-1$ ,  $x+1$ ,  $x+2$ ,  $x+3$ , ..., it acknowledges  $x$ ,  $x$ ,  $x$ , ..., indicating its expecting a segment with seq #  $x$ .
- ❑ Duplicate ACKs of  $x$  are strong indications of the loss of  $x$ .
  - Why can't we be 100% certain ? Out of order arrival
- ❑ However, the network does not seem congested, for subsequent segments do get through.

CS 756

12

- ❑ When the sender sees 3 duplicate  $\text{Ack}(x)$ , it retransmits  $x$  immediately (not to await timeout); hence the name “fast” retransmission.
- ❑ Subsequently, we want to proceed cautiously but not as pessimistically as the cases of timeouts.

## Fast Recovery

- ❑ After fast retransmission of segment  $x$ , half the value of  $\text{cwnd}$ 
  - Notice that  $\text{cwnd}$  is now automatically greater than  $\text{ssthresh}$
- ❑ The result is to skip the slow-start phase; hence the name “fast” recovery.

# Example

