

TCP congestion management

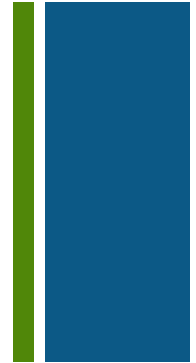
How TCP prevents and detects congestion in the Internet



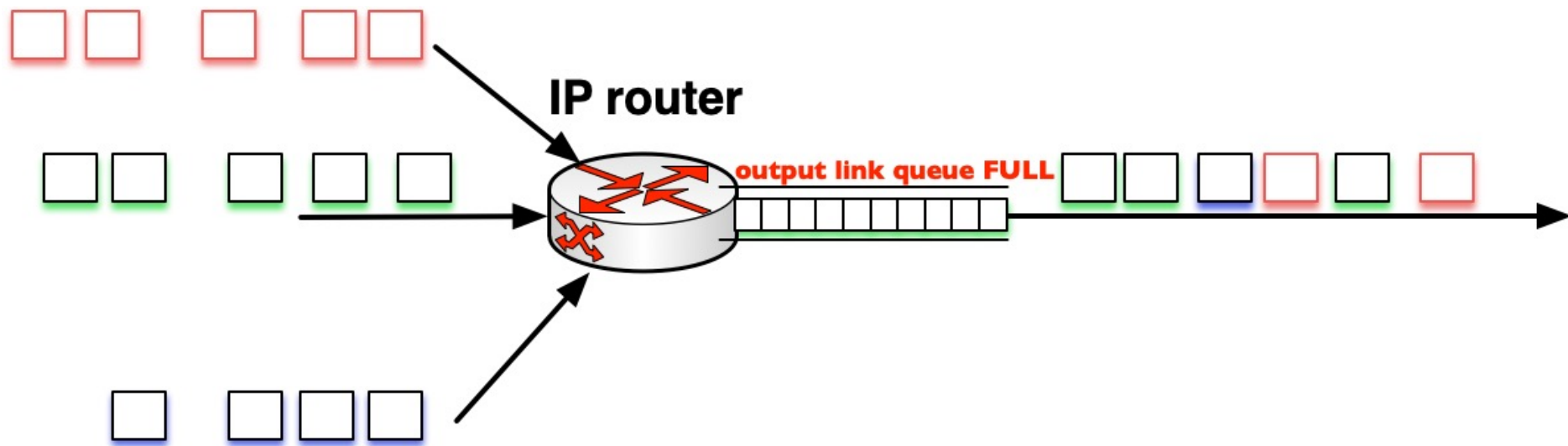
The basics of Internet congestion

When excessive network delay compromises service

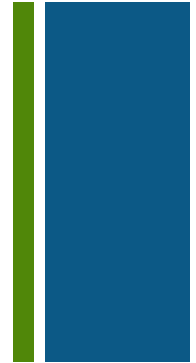
+ Basic structure of an IP Router



- At this moment, *the output link*, receives traffic from three *input links*
- The output link, when demand is high, queues packets in a buffer
 - Increases the delay undergone by each packet
 - In the limit, when the link is congested, it begins to drop packets (Packets get lost)



+ Queue length: Little's Law



If t is sufficiently large:

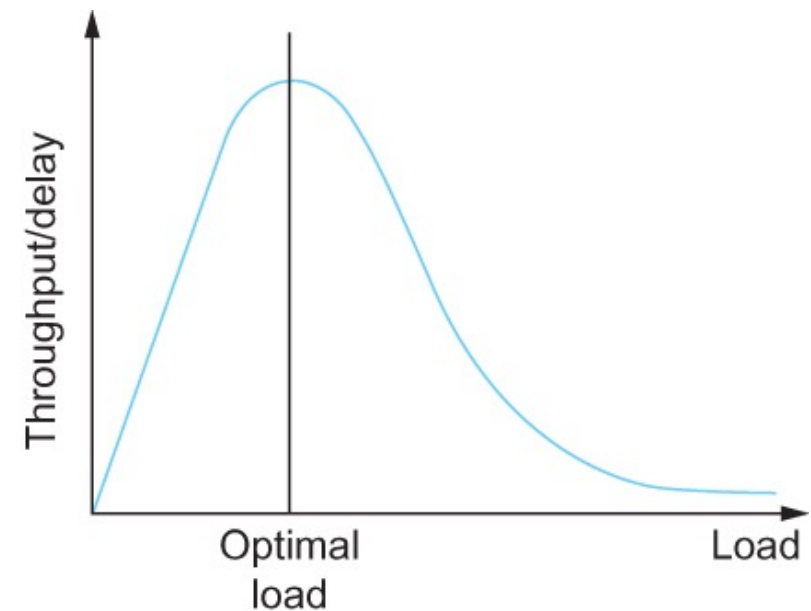
- $N = \lambda \cdot T_a$
 - The *average* queue length is given by the product of average packet rate and the average residence time
 - The interarrival time is given a by a Poisson probability distribution $A(t) = P(\text{interravival time} \leq t)$
- Applies to a variety of queue disciplines, not only FIFO
- Input probability distribution



+ Power curve of a network

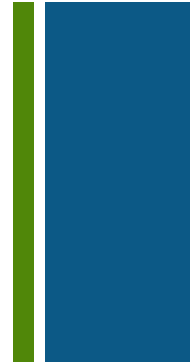


- As the offered load increases, the ratio Throughput/delay also increases
 - Offered load
 - Achieved throughput/delay (T/d)
- A time point comes when the T/d flattens and then begins to decrease as offered load keeps increasing
 - This is due to the increasing delay at each router

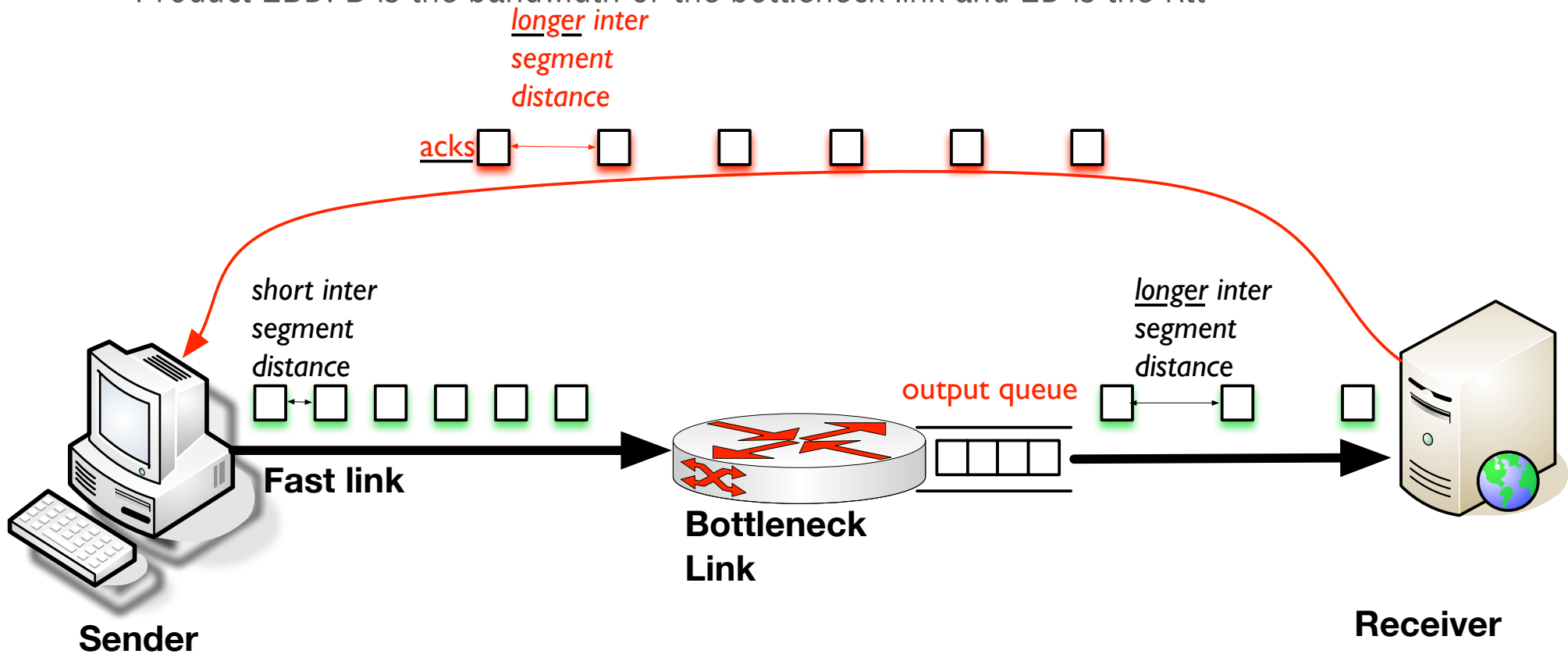


© Morgan-Kaufmann 2012, Prof. L. Peterson and Bruce Davie

+ Bottleneck link at an IP router



- The bottleneck link limits the maximum number of segments present in the network
- Product $2BD$: B is the bandwidth of the bottleneck link and $2D$ is the R_{tt}



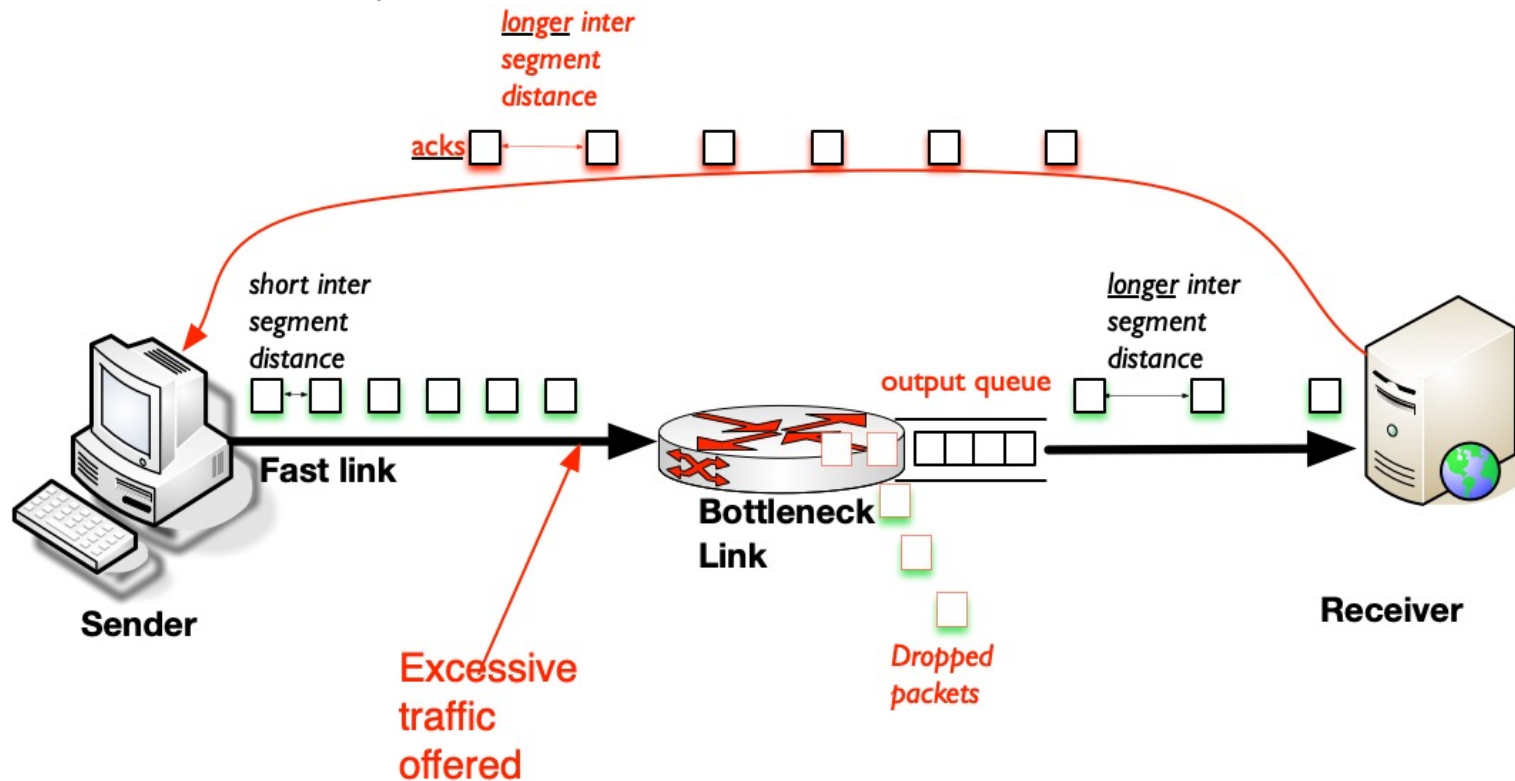


+

TCP, congestion control

+ How TCP discovers the end-to-end capacity of a TCP connection

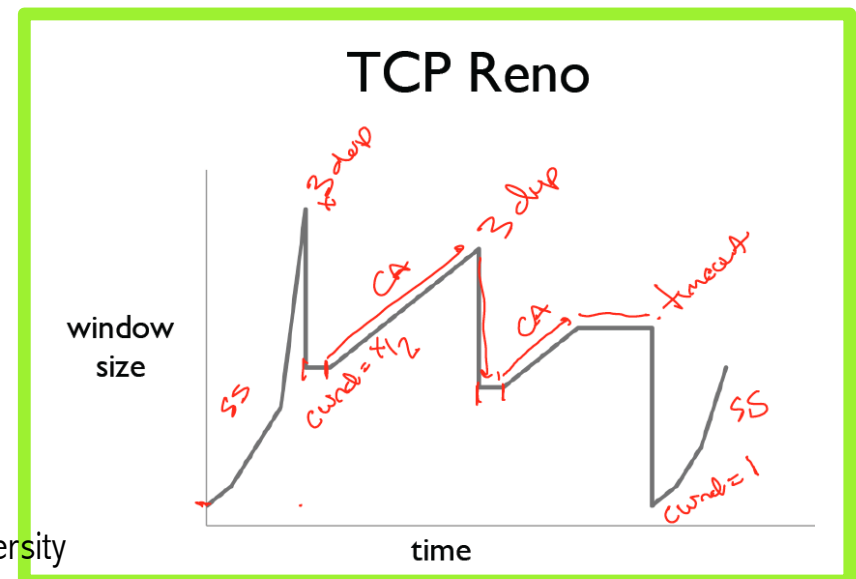
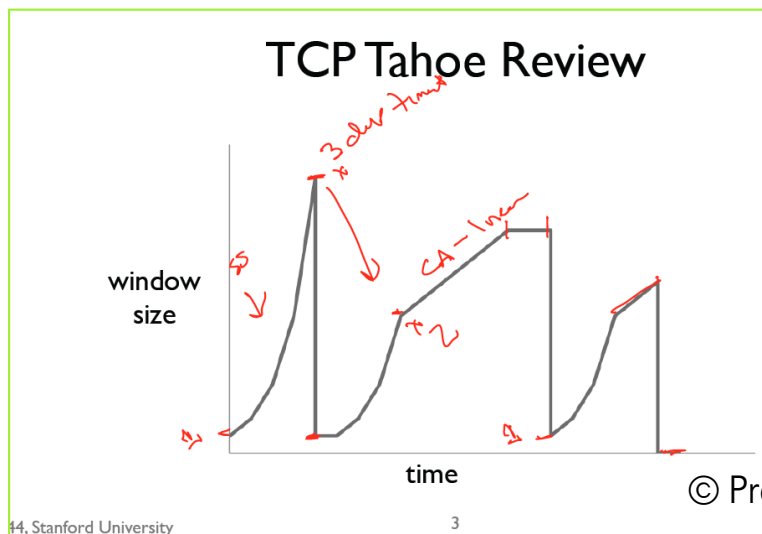
- TCP needs to discover how many packets/sec can be injected into the network, *safely*
- With a limited packet loss



+ AIMD: Additive Increase, Multiplicative Decrease

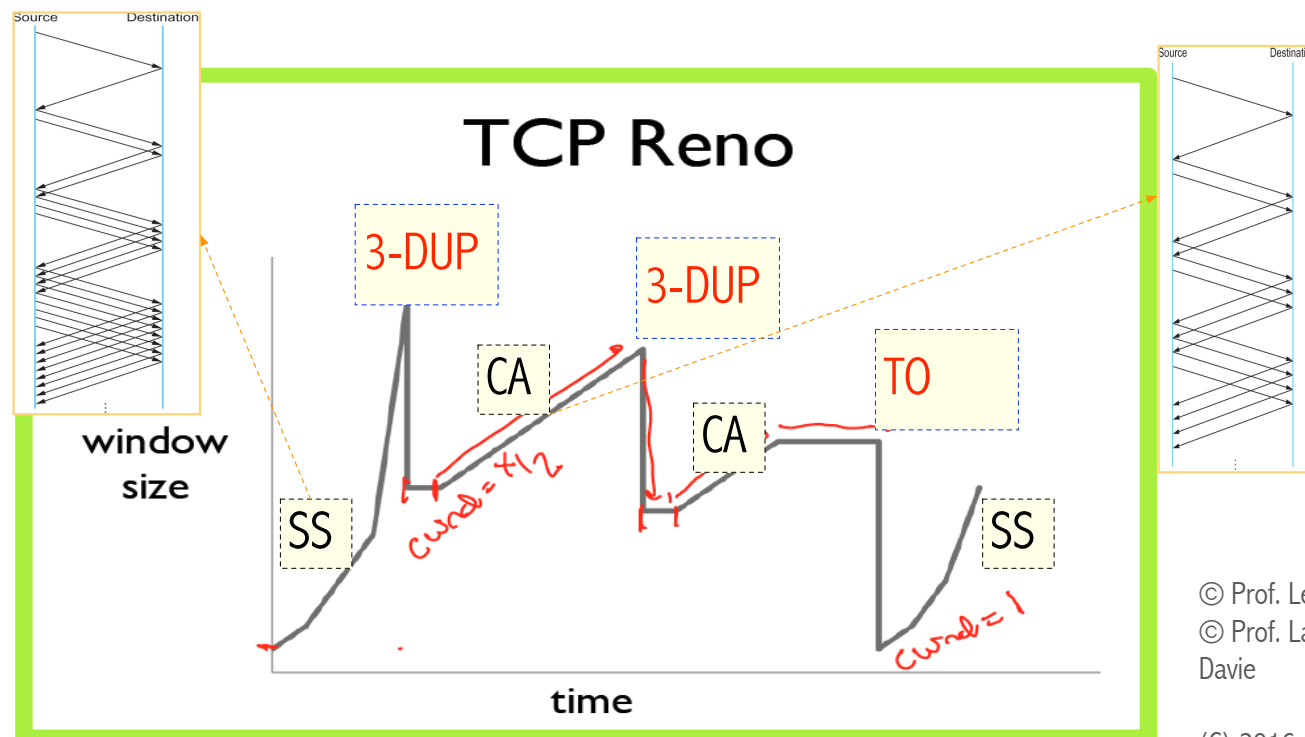
(C) 2016 José María Foces Morán

- TCP needs to discover how many packets/sec can be injected into the network, safely
- Without packet loss
- The effective TCP's transmit window becomes $= \text{MIN}(\text{CongestionWindow}, \text{AdvWindow})$
- $\text{CW} = \text{CongestionWindow}$



+ How discovers network capacity

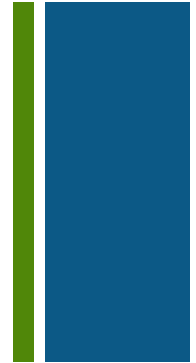
- Slow Start (SS)
 - Probe for network capacity by growing CW (Congestion window)
 $CW = 2 * CW$ each Rtt
 - Initially, $CW = 1$
- 3-DUP causes transition to CA (Congestion Avoidance) with $CW = SSthresh / 2$
- TO (Timeout) causes SS to start again
 - Linux implements TCP Reno and CUBIC congestion control



© Prof. Levis, Stanford University
© Prof. Larry Peterson and Bruce Davie

(C) 2016 José María Foces Morán

+ Reno, Fast Retransmit and Fast Recovery



■ Fast Retransmit:

- Upon a 3-DUP the transmitter will retransmit the missing segment, only

■ Fast Recovery

- Also, artificially increase $CW = CW + 3$ to compensate for the 3-DUP that didn't advance LastByteAked and which, therefore, could not be used to spur the transmitter to transmit 3 new segments
- Use the remaining, upcoming ACKS to keep the transmission pace
- NO Slow Start in Reno upon 3-DUP

