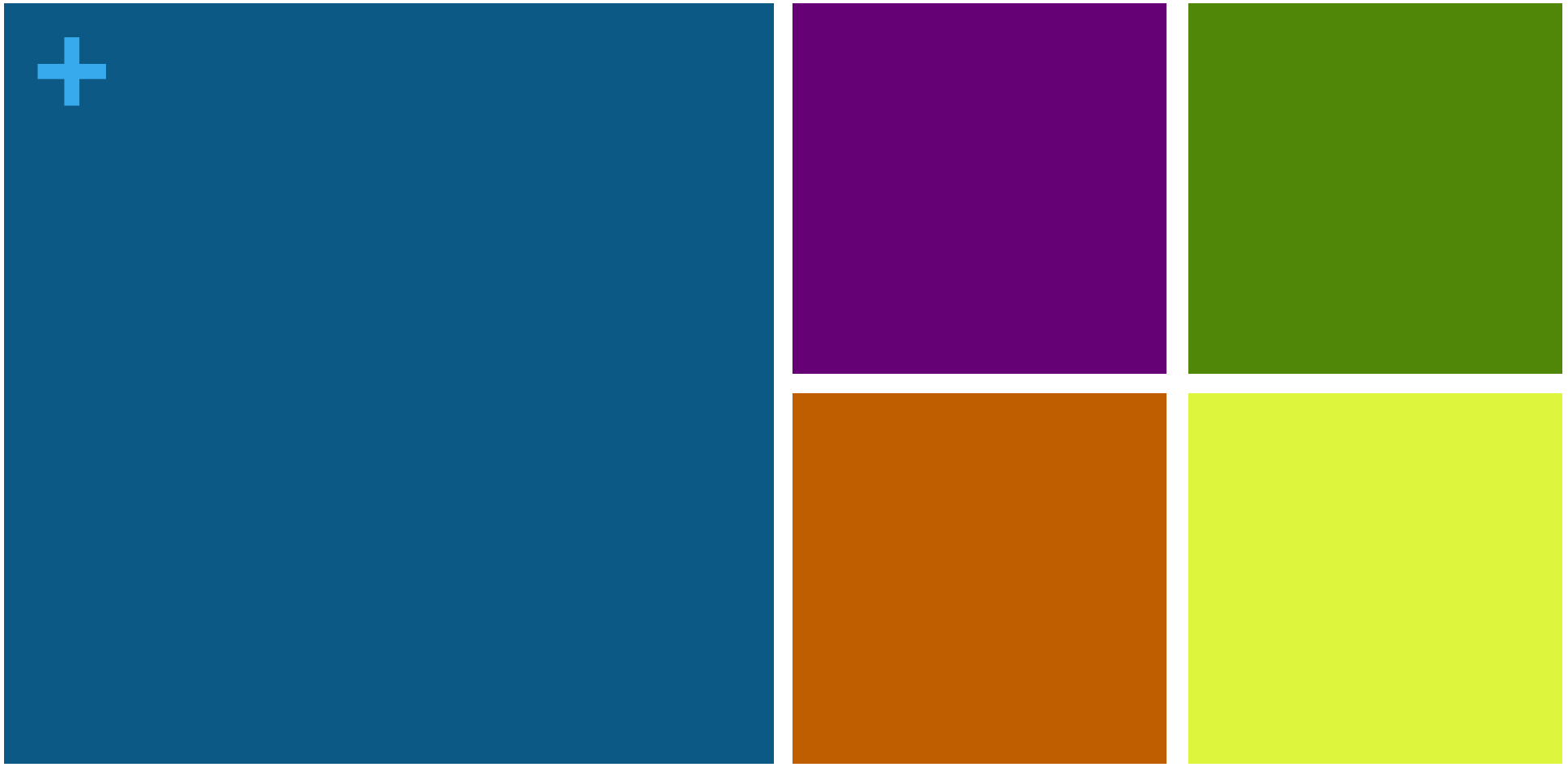


TCP congestion management

How TCP prevents and detects congestion in the network

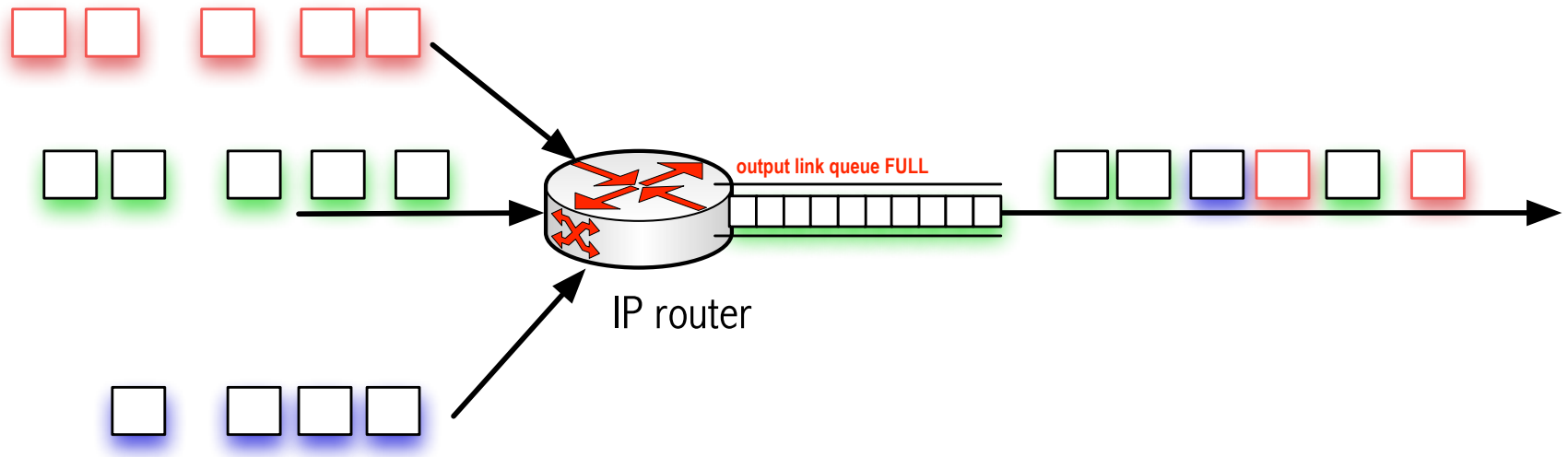
V 0.2 17-Nov-2016

The basics of Internet congestion

When excessive network delay compromises service

Basic structure of an IP Router

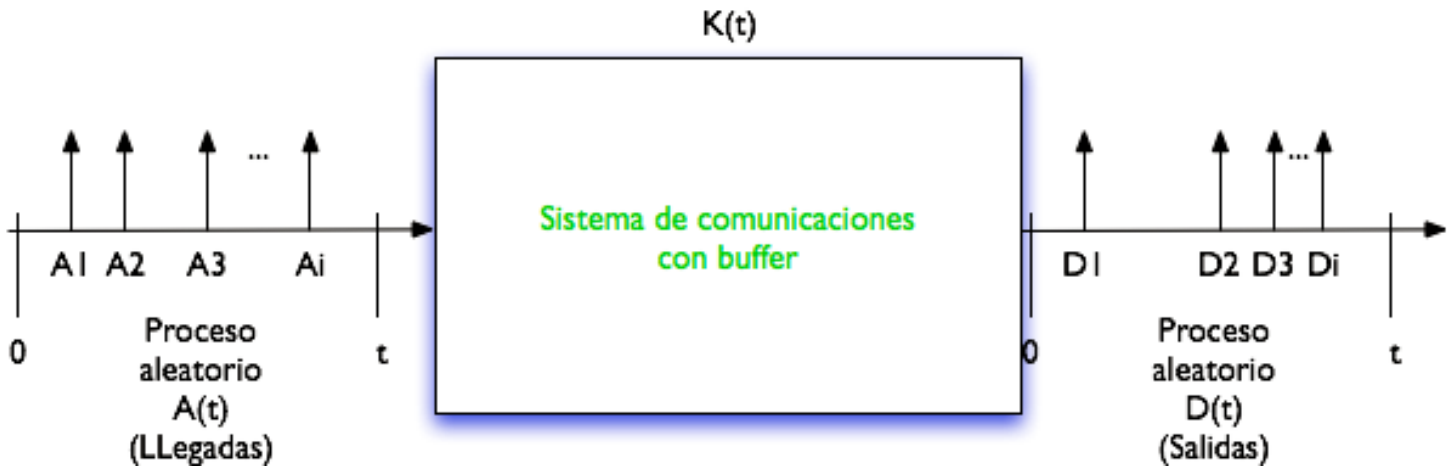
- At this moment, *the output link,* receives traffic from three *input links*
- The output link, when demand is high, queues packets in a buffer
  - Increases the delay undergone by each packet
  - In the limit, when the link is congested, it begins to drop packets (Packets get lost)



Queue length: Little's Law

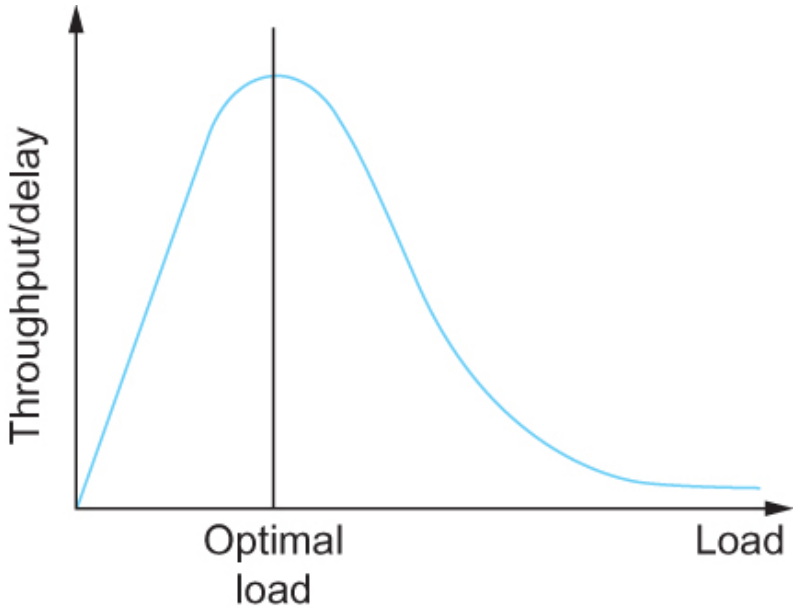
If t is sufficiently large:

- $N = \lambda \cdot T_a$
  - 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(interravival time <= t)**
- Applies to a variety of queue disciplines, not only FIFO



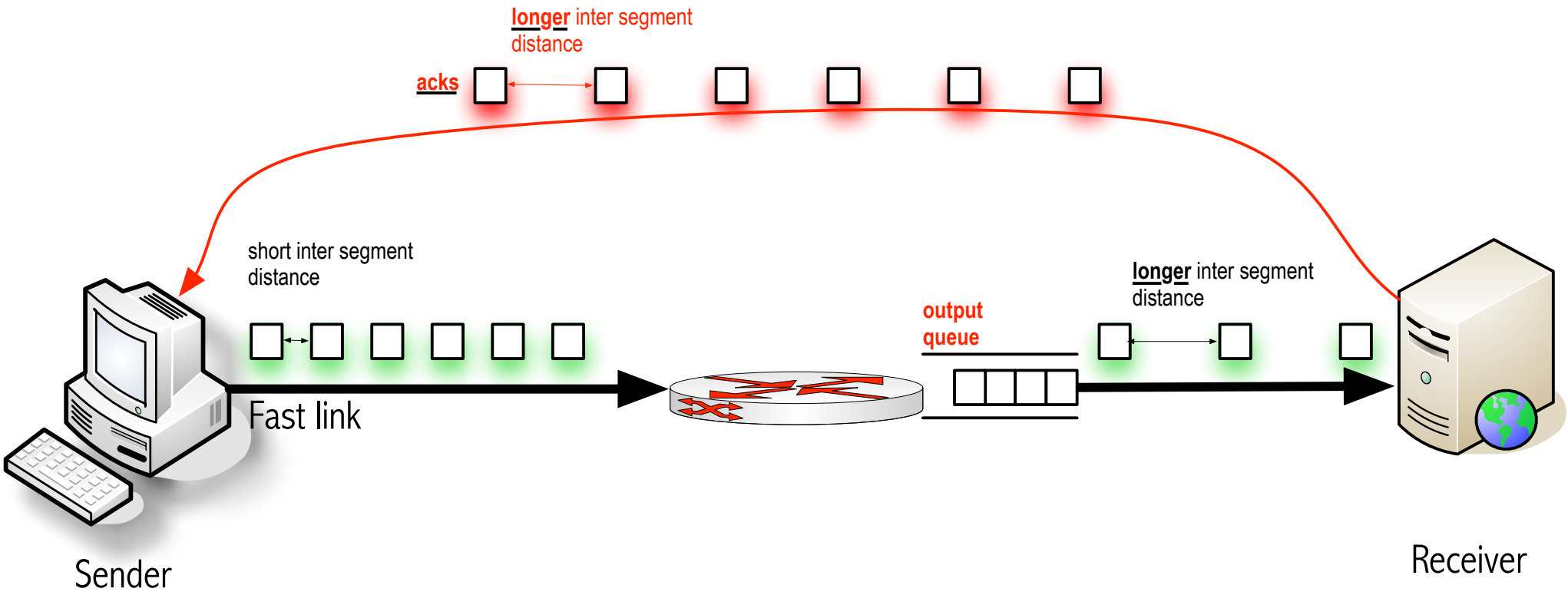
Power curve of a netowrk

- As the offered load increases, the ratio Throughput/delay also increases
- **A time point comes when the T/d flats and the begins to decrease**
  - **This is due to the increasing delay at each router**



Bottleneck link at an IP router

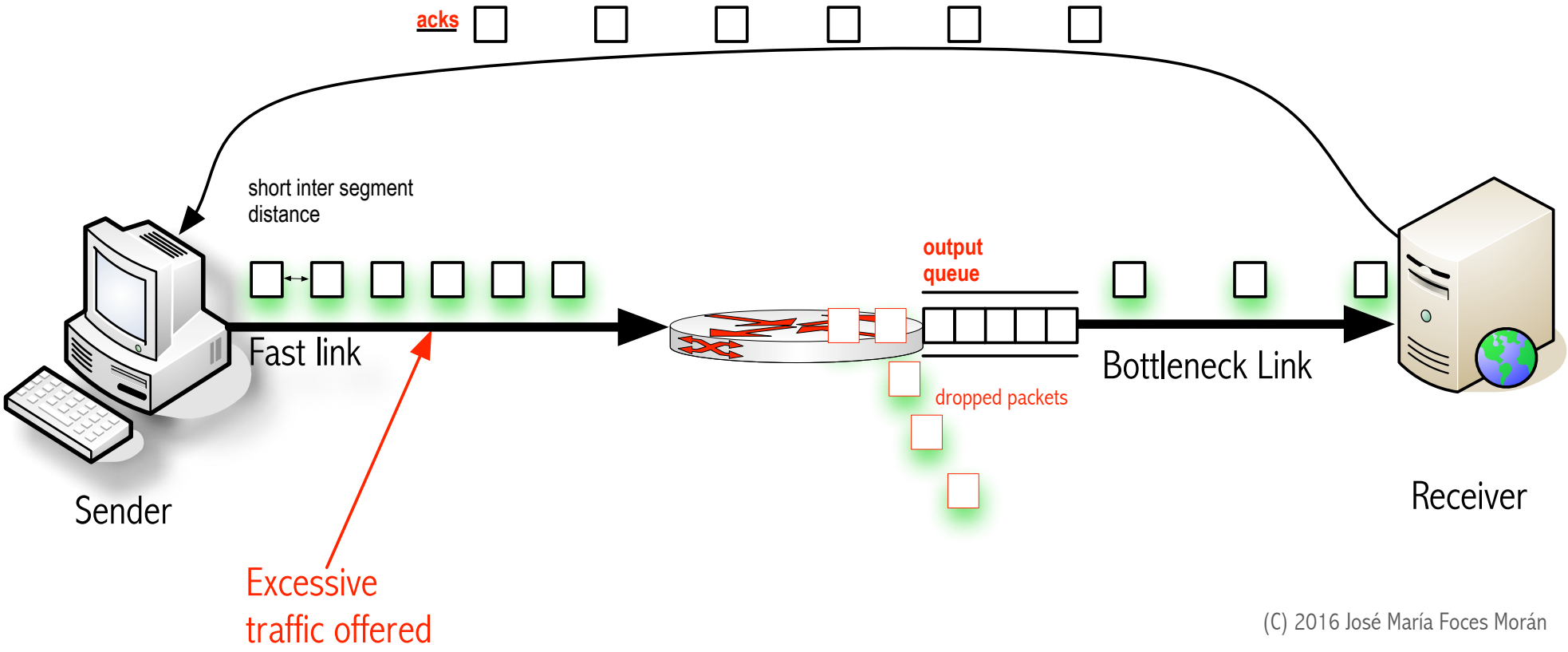
- The bottleneck link limits the maximum number of segments present in the network
- Product 2BD: B is the bandwidth of the bottleneck link and 2D is the Rtt



TCP, congestion control

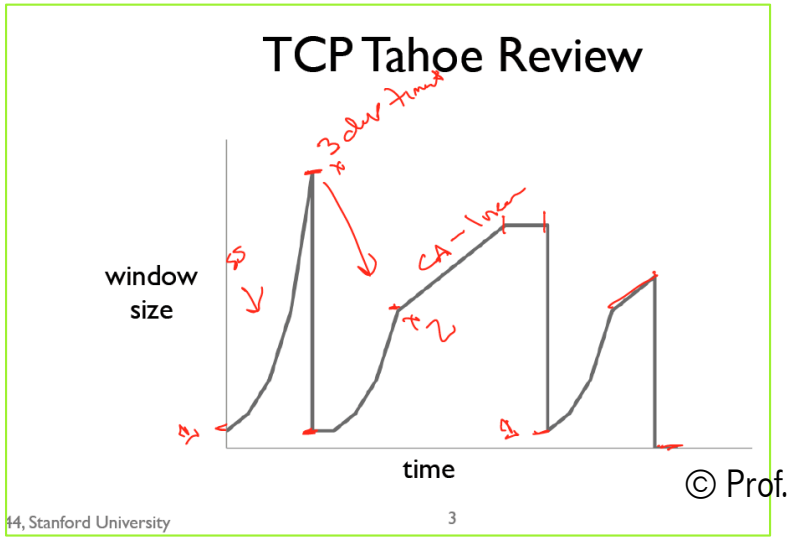
How TCP discovers capacity of an end-to-end link (A TCP connection)

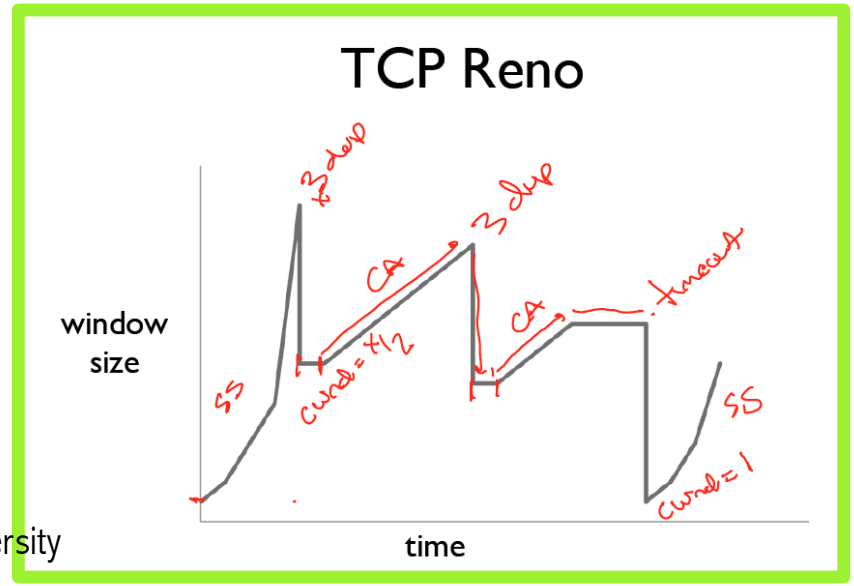
- TCP needs to discover how many packets can be injected into the network, safely
- Without packet loss



AIMD: Additive Increase, Multiplicative Decrease

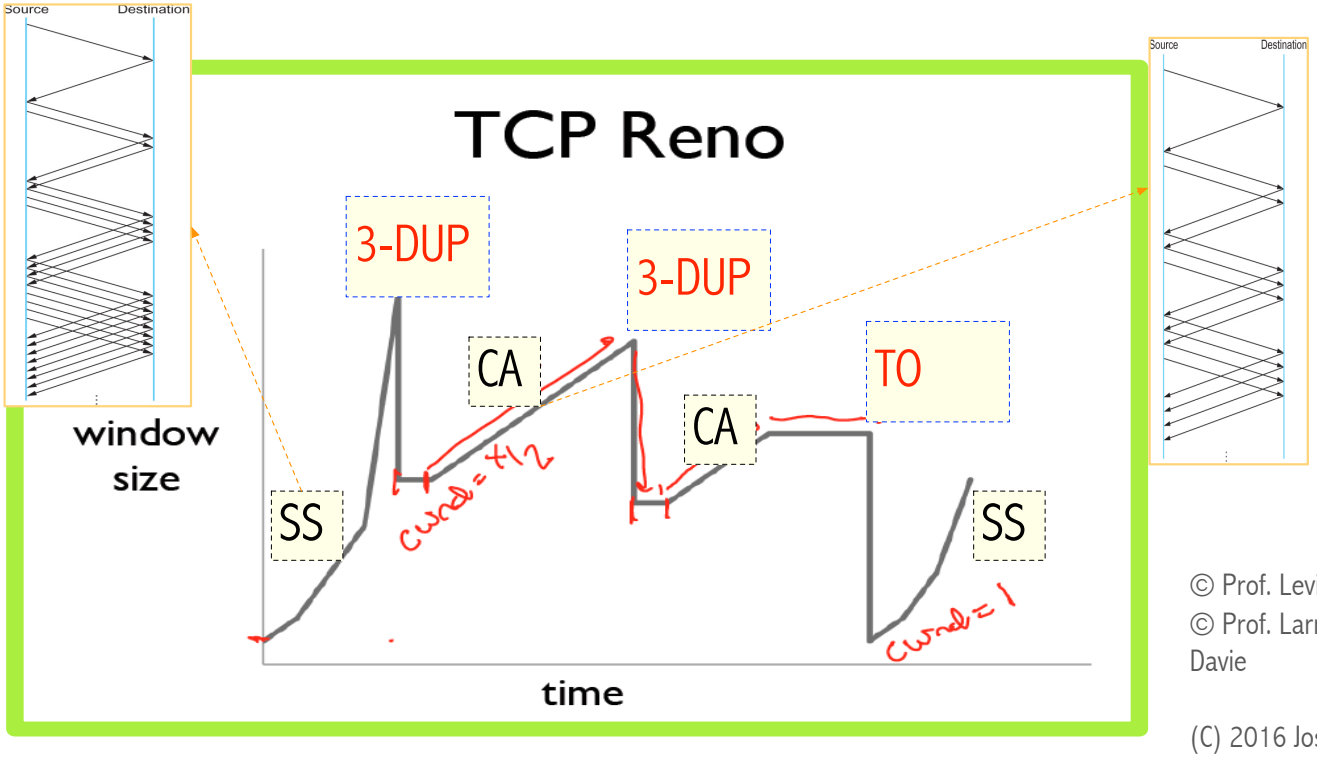
- TCP needs to discover how many packets can be injected into the network, safely
- Without packet loss
- The effective TCP's transmit window becomes =MIN (CongestionWindow, AdvWindow)
- CW = CongestionWindow





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 = SSthrsh / 2
- TO (Timeout) causes SS to start again



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 LastByteAcked 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