TCP Congestion Control -- Overview

Kameswari Chebrolu

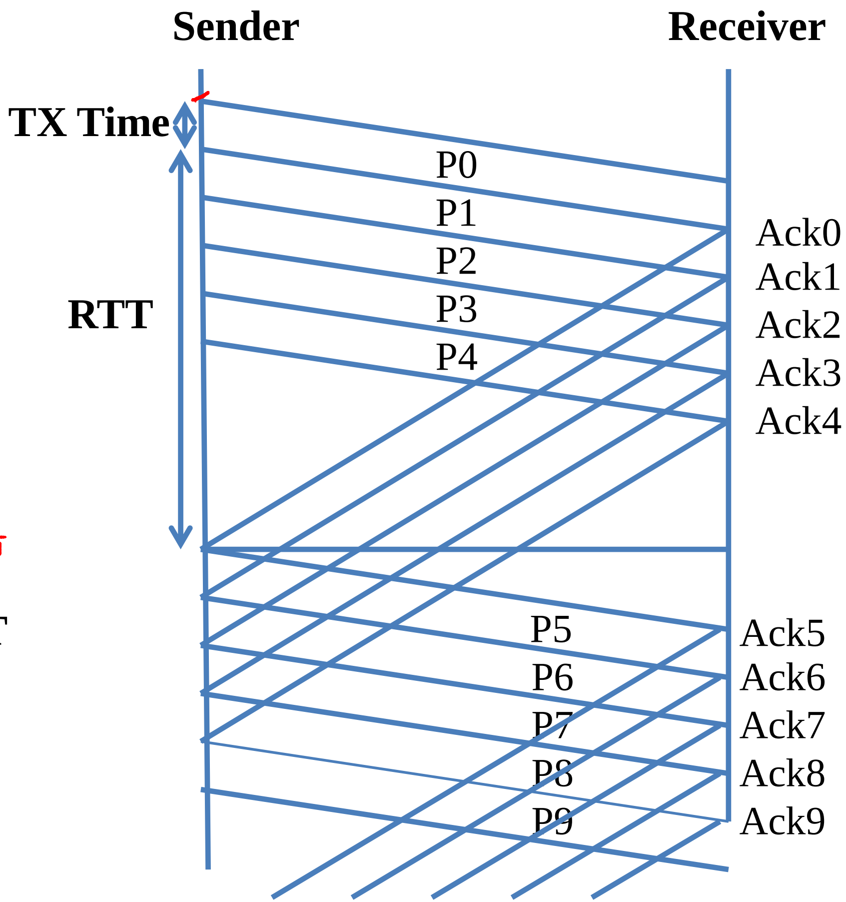
Seminal Paper: Congestion Avoidance and Control
by Van Jacobson and Michael J. Karels

Recap: TCP Services

- Multiplexing/Demultiplexing
- Reliable point-to-point data transfer
- Full-duplex
- Congestion control
- Flow control

Sliding window Protocol

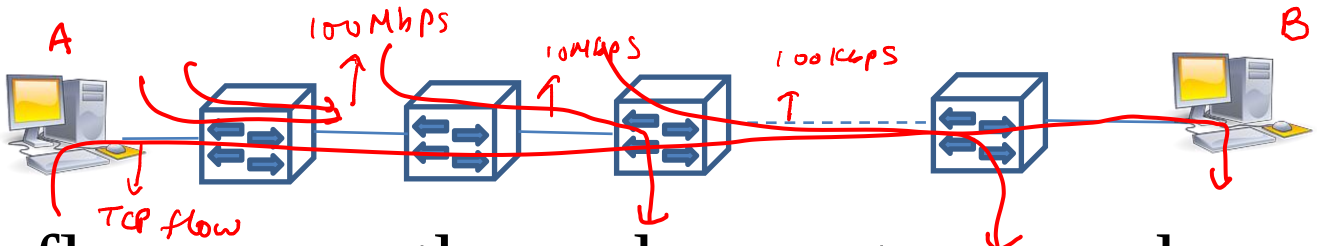
Recap: Sliding Window



wind. size pkts

Throughput ~= (W * MSS)/RTT

Congestion Control: Problem Statement



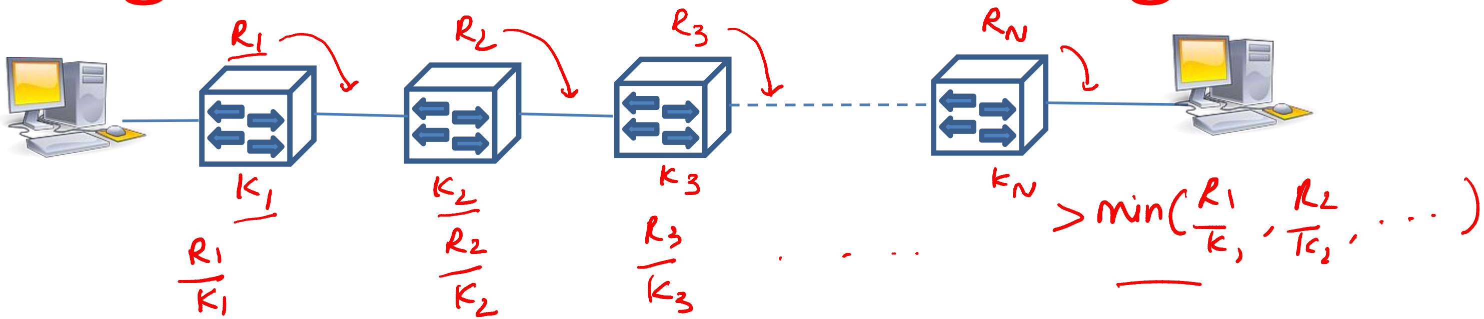
- Many flows pass through a router; number varies with time

fairness

blasts

- Flows can be TCP or UDP
- The link capacities of the routers are different
- End Result: Throughput achieved by a given flow function of many factors

Congestion Control: Challenge

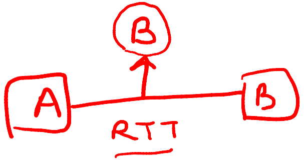


- Need to estimate W (of sliding window) such that each flow gets its fair share
  - Estimate small → underutilization; Estimate large → Congestion
- W will vary over time
- Congestion Control: Preventing sources from sending too much data too fast and thereby ‘congest’ the network

Sliding Window Protocol

- Roughly, idea translates to the following:
- View network as a pipe
- Determine the capacity of the pipe (Bandwidth-delay product)
- Fill the pipe with data
- As you remove one packet form the pipe, add another
  - ACKs help clock out data (Self Clocking)

Self- Clocking



$W \rightarrow$ window size in bits

① $\rightarrow W = B \times RTT$

② Max Thr A & B - ? = B

③ Rate @ which Acks are coming?

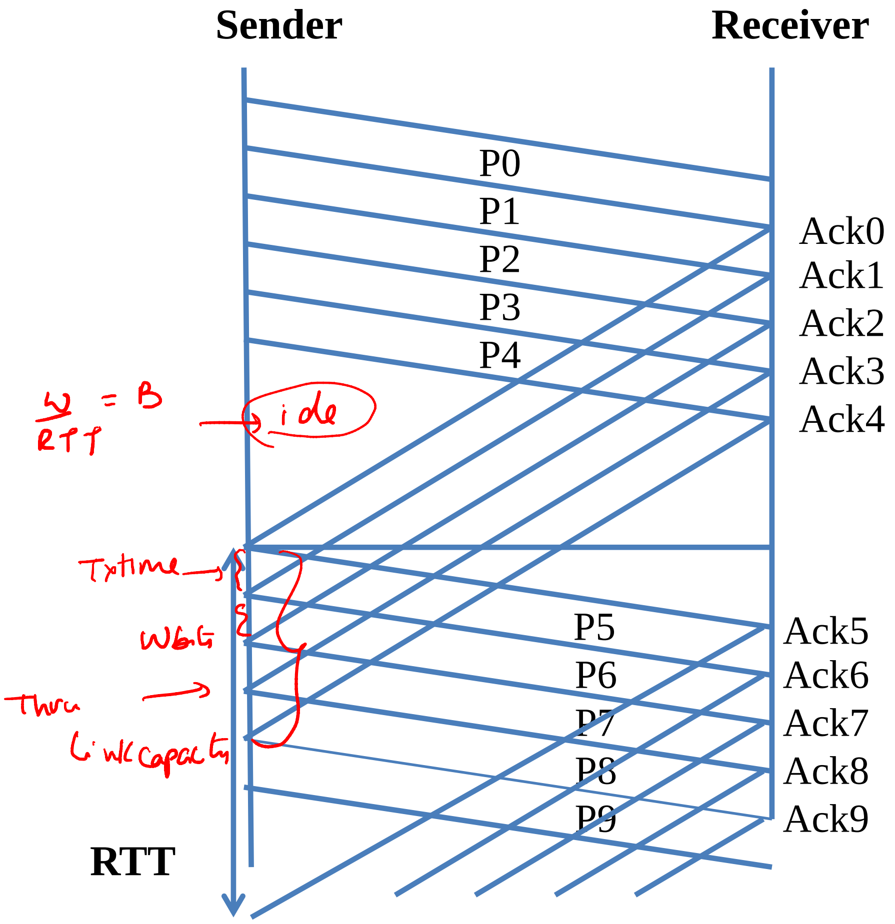
→ Rate @ which Pkt

↳ Link capacity

$\boxed{W < B \times RTT}$

→ Link capacity

Self-clocking



Self- Clocking

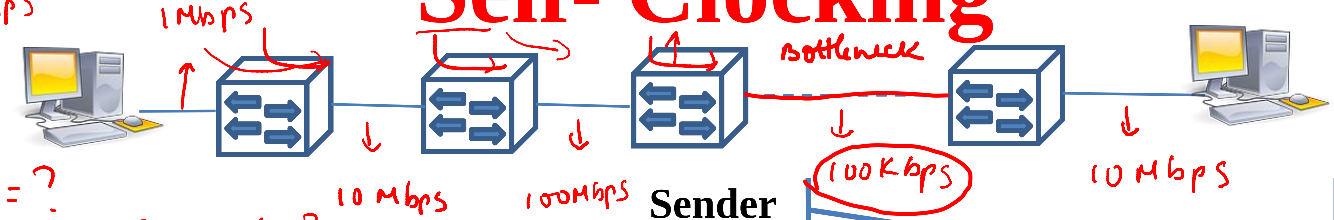
RTT x 100 Kbps

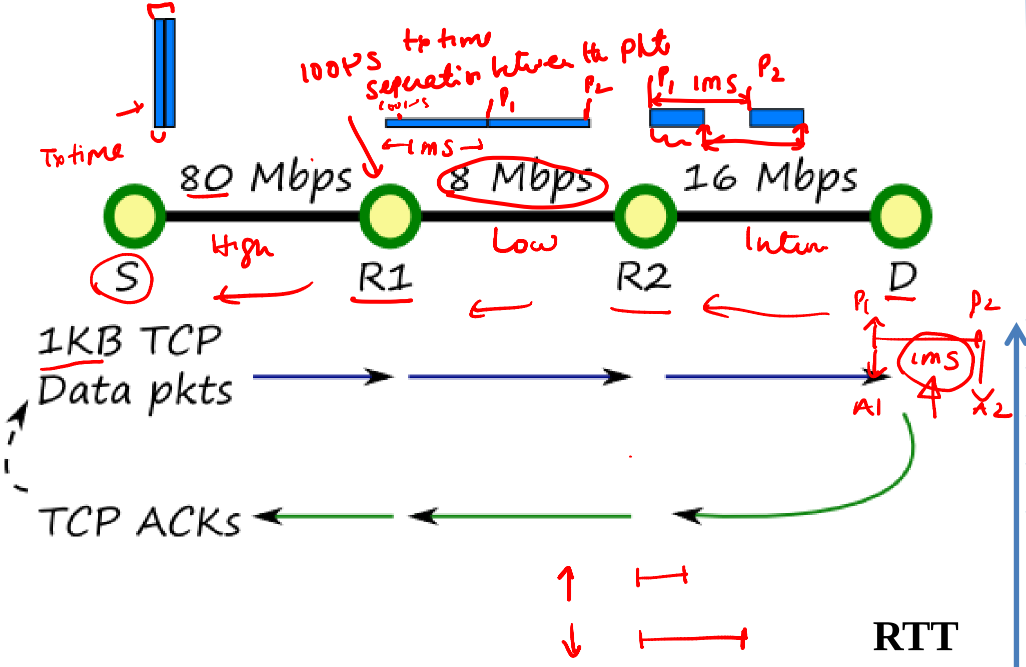
① w = ?

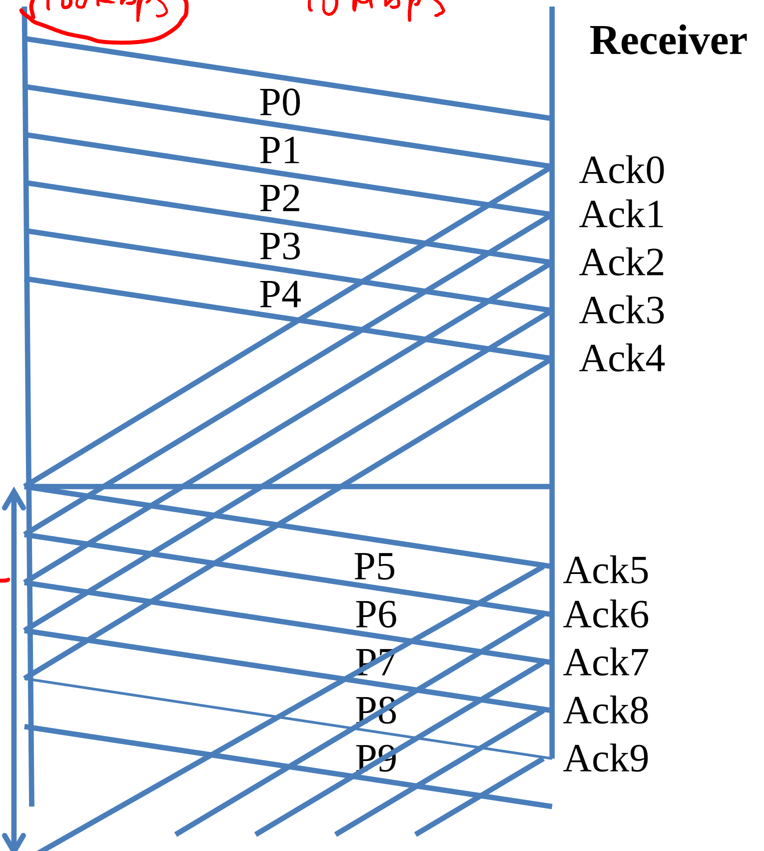
② Max Thr. = ?

③ rate @ ack @ sender ?

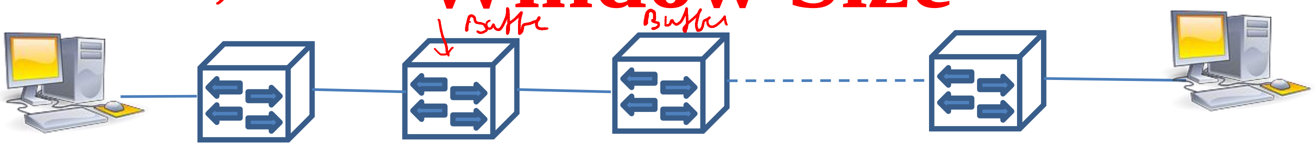
→ rate @ pkts on this bottleneck link







Window Size



- W > BDP (in bits)

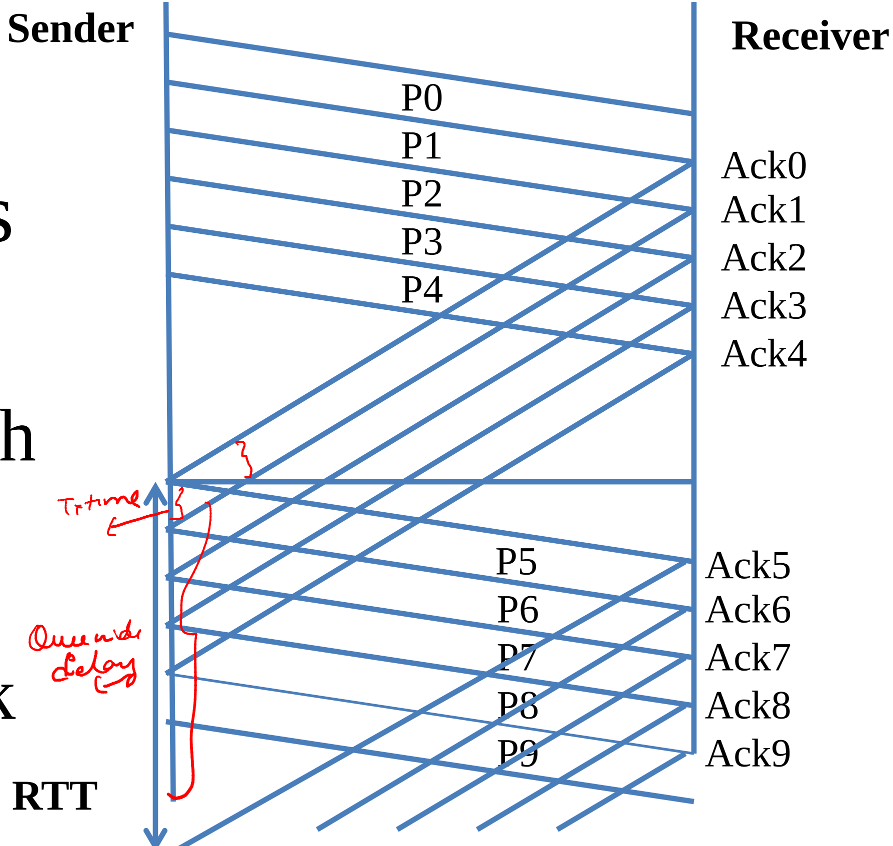
  RTT noload

- Sliding Window says

  RTT_load

  – Thr = W/RTT > Bottleneck bandwidth

- Self-clocking says
  – Max thr is bottleneck bandwidth



Window Size


BDP
Throughput
winsize
delay
winsize
queue utilization
winsize

Same Idea

- View network as a pipe
- Estimate Bandwidth-delay product (capacity) dynamically
  - Uses the variable Congestion Window (CW) to track it
- Self Clocking: Captures bottleneck bandwidth
  - Use ACKs to clock out data
  - Not perfect (With competing traffic ack spacing will not be preserved)

3 Steps

- Getting to Equilibrium
- Conservation at equilibrium
  - Don’t put new packet unless old one is removed
- Adapting to Path Dynamics

Summary

- Congestion Control is a complex problem
- Need to implement it in the context of the sliding window protocol
  - Self clocking is a useful feature (we will rely on this to capture bottleneck bandwidth)
  - Need to determine and adapt W (window size) such that you don't underutilize bandwidth or congest the network
- Ahead: Actual details