Multi-Path Congestion Control

Lecture 20, Computer Networks (198:552)

Fall 2019

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Review: TCP congestion control

- Keep some in-flight (un-ACK'ed) packets: congestion window

- Adjust window based on several algorithms:
  - Startup: slow start
  - Steady state: AIMD
  - Loss: fast retransmission, fast recovery

- Classically, TCP uses a single path provided by the underlying network routing

If a TCP conn could use multiple paths…

- Better resilience
  - If one path becomes unavailable, keep traffic flowing over others
- Higher throughput
  - Use multiple paths to overcome single-path bottlenecks
- Seamless mobility
  - More paths as they become available, “handing off” as needed

- Example uses
  - Mobile phone (WiFi/cellular)
  - High-end servers (multiple NICs)
  - Data centers (many paths available)

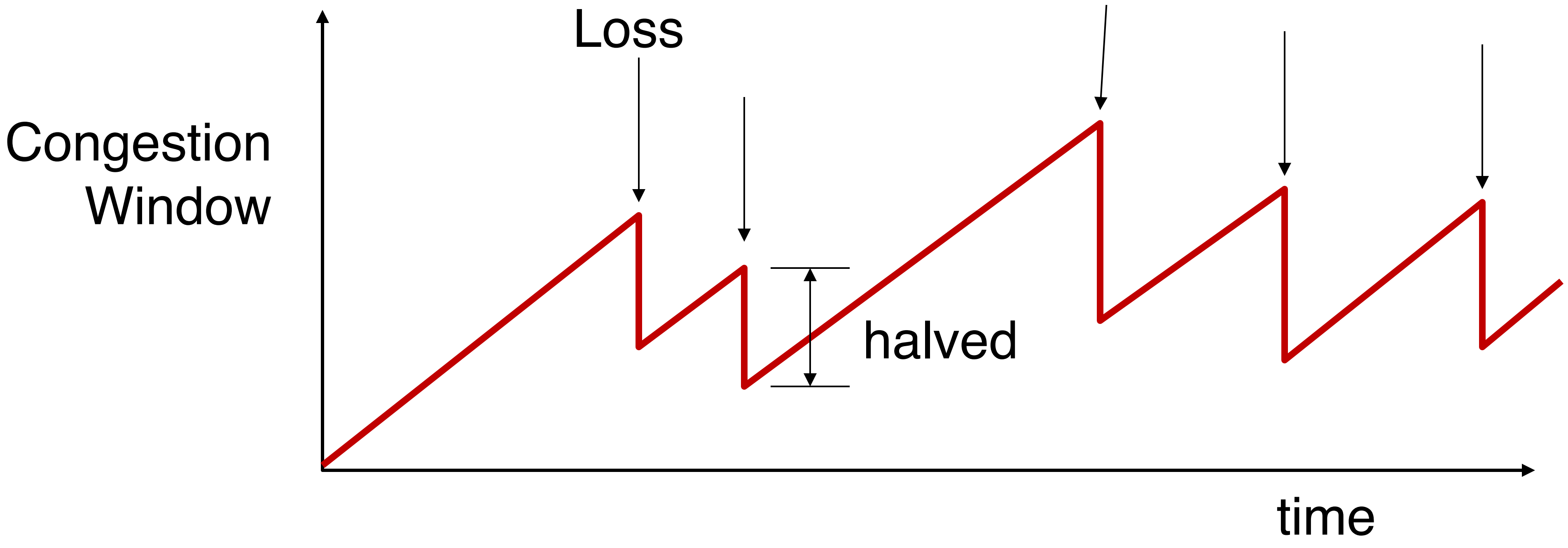
Goal: Do all this without application-level changes

TCP throughput equation

Steady-state behavior

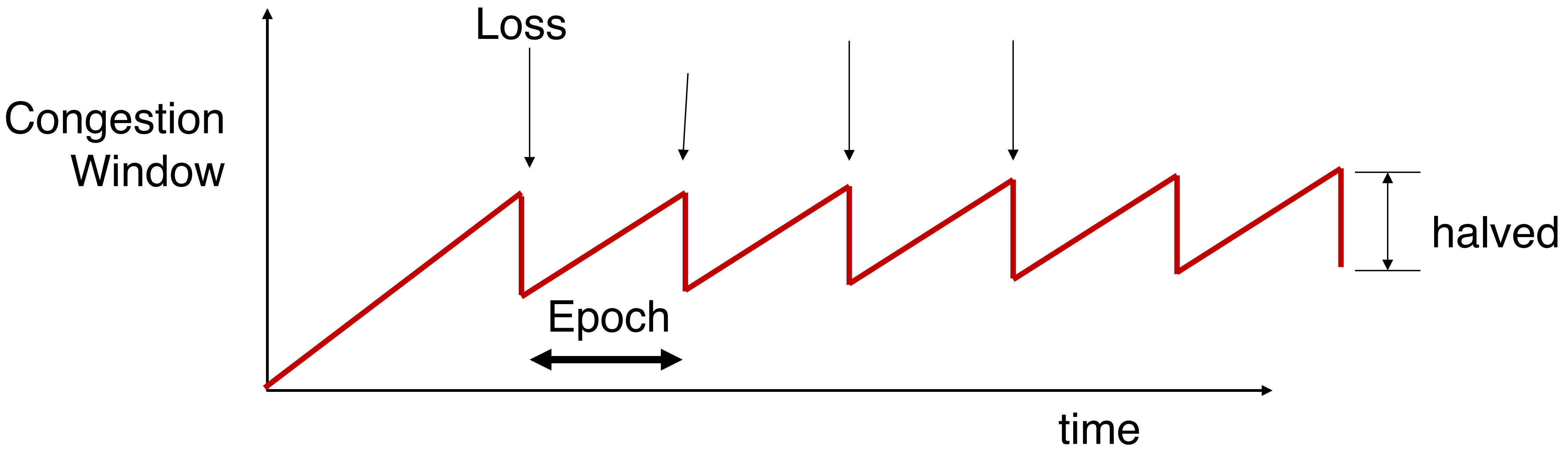
Model of single-path TCP

- AIMD: W := W + 1 (RTT), W := W/2 (drop)
- Only a single flow using the bottleneck link



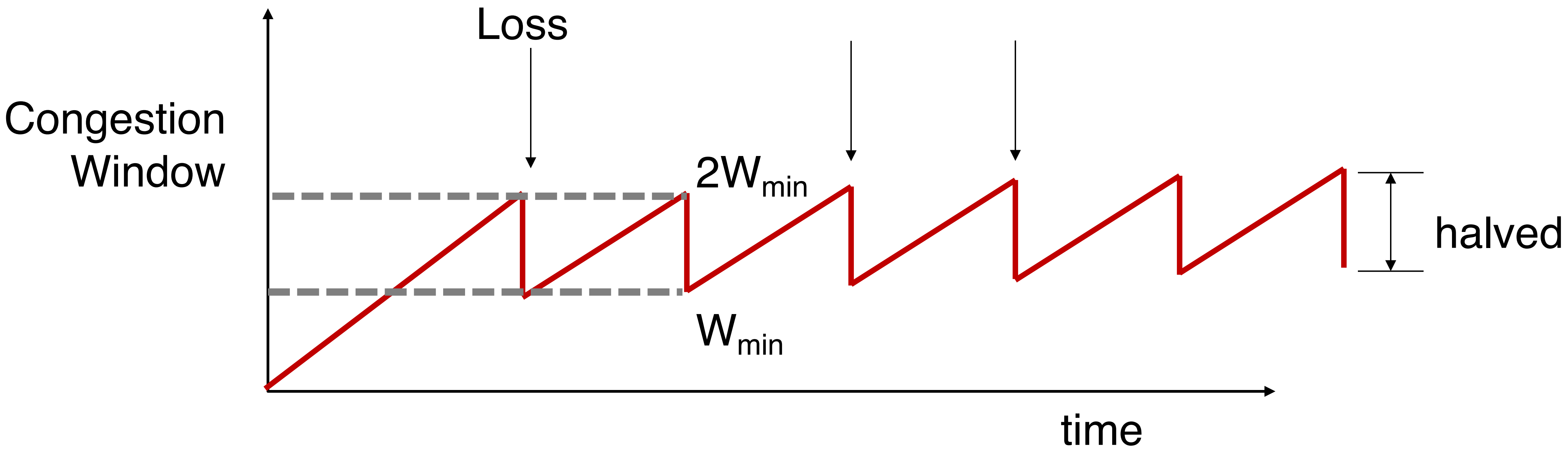
Model of single-path TCP

- Idealization: Repeat exactly same window evolution over multiple epochs of a single packet loss



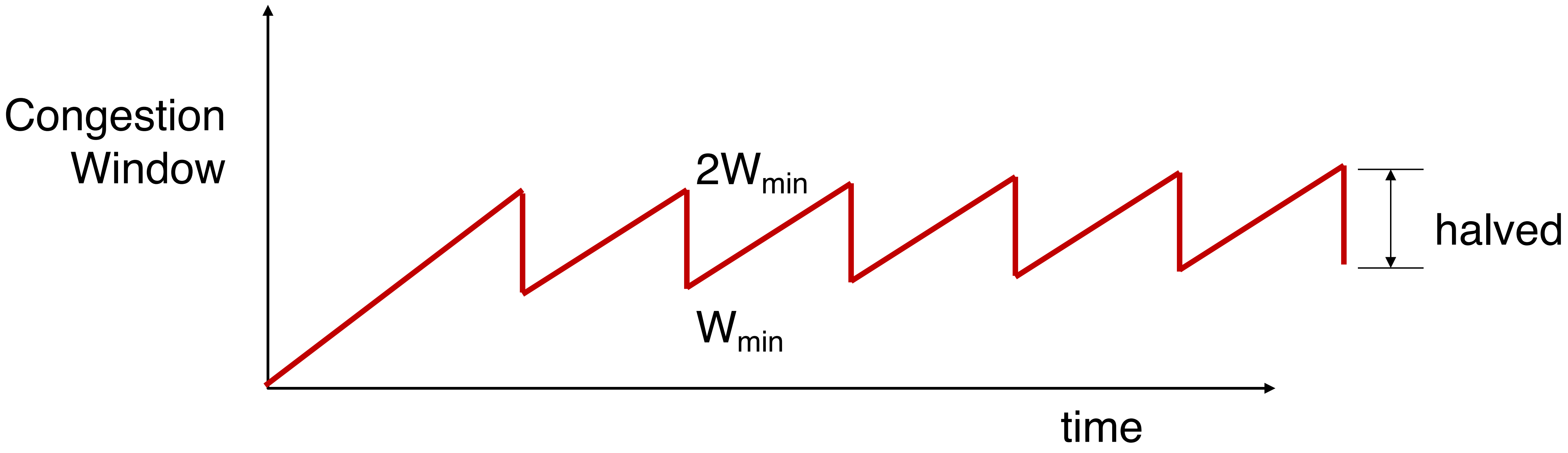
Model of single-path TCP

- Window goes from W_{min} to $2 * W_{min}$
- Loss rate p: number of packets dropped per packet sent



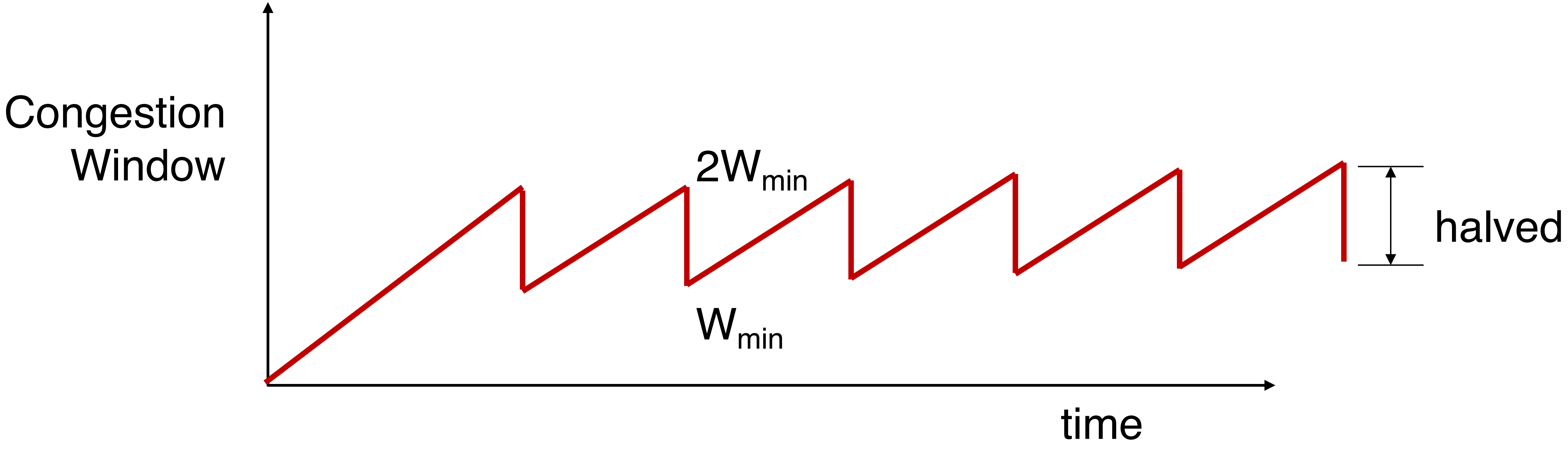
Model of single-path TCP

- Loss rate assumed to be independent of sending rate
  - In reality: more you send, more links traversed, more you drop
- Loss assumed deterministic (e.g., buffer full)
  - In reality: AQM, stochastic channel loss (e.g., cellular)



Model of single-path TCP

- Goal: Find TCP's throughput as a function of link rate, RTT, and packet loss rate
- Throughput = (#packets sent per epoch) / (time per epoch)



TCP throughput equation

- Throughput depends inversely on sqrt of loss rate, $p^{-1/2}$
  - Throughput drops steeply with increasing loss rate
  - Ideal: want it to be linear drop, ie: C*(1-p)

- Throughput depends inversely on the RTT
  - An issue known as RTT unfairness: lower-RTT connection on a bottleneck gets higher throughput than higher-RTT connection
  - Ideal: independent of RTT

- Is throughput independent of the link rate and buffer size?

Multipath TCP

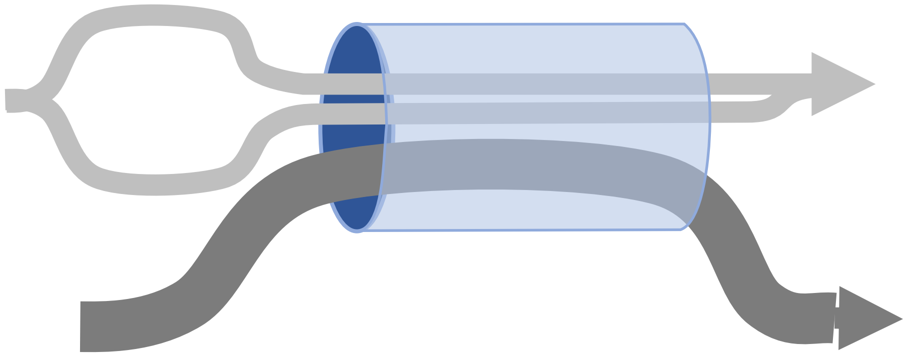
Design of the congestion control algorithm

Slides adapted from Damon Wischik

Goal #1: Fairness at Shared Bottlenecks

A multipath TCP flow with
two subflows

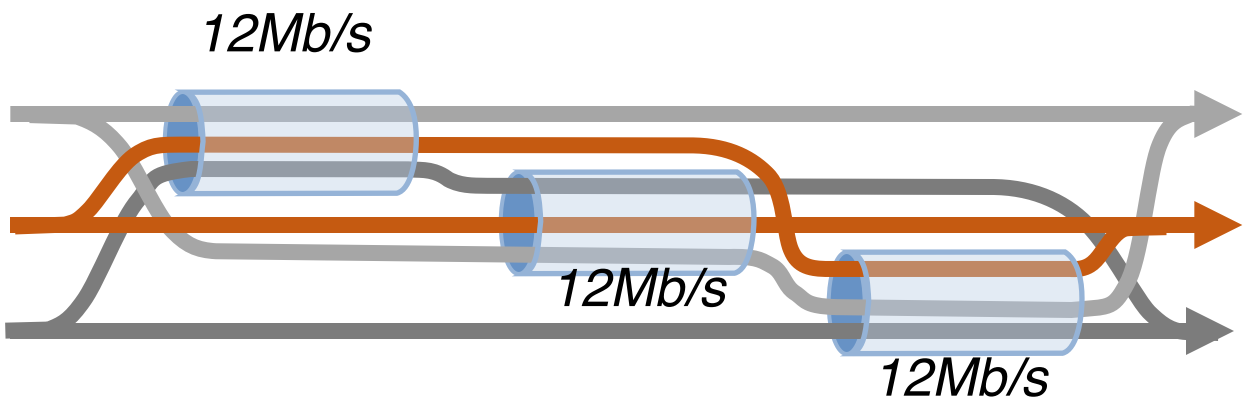
Regular TCP



Why not just open multiple TCP connections?

To be fair, Multipath TCP should take as much capacity as TCP at a bottleneck link, no matter how many paths it is using.

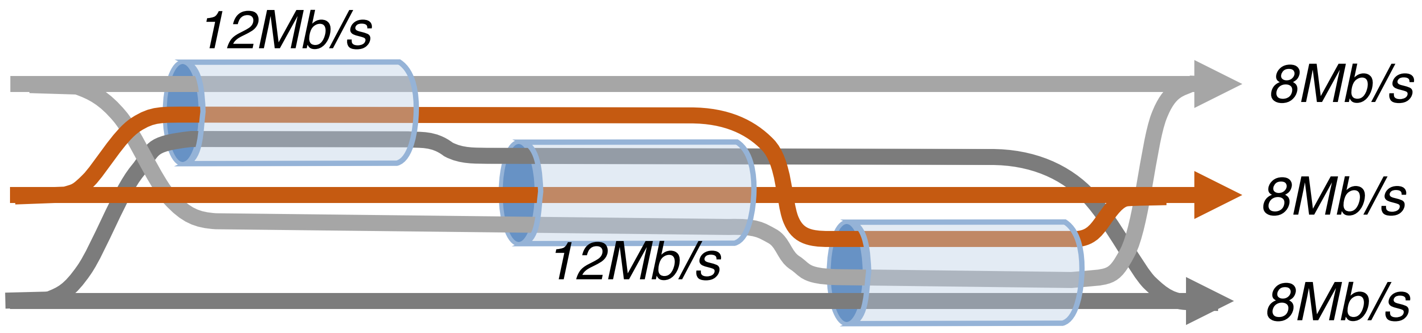
Goal #2: Use Efficient Paths



Each flow has a choice of a 1-hop and a 2-hop path.

How should each flow split its traffic?

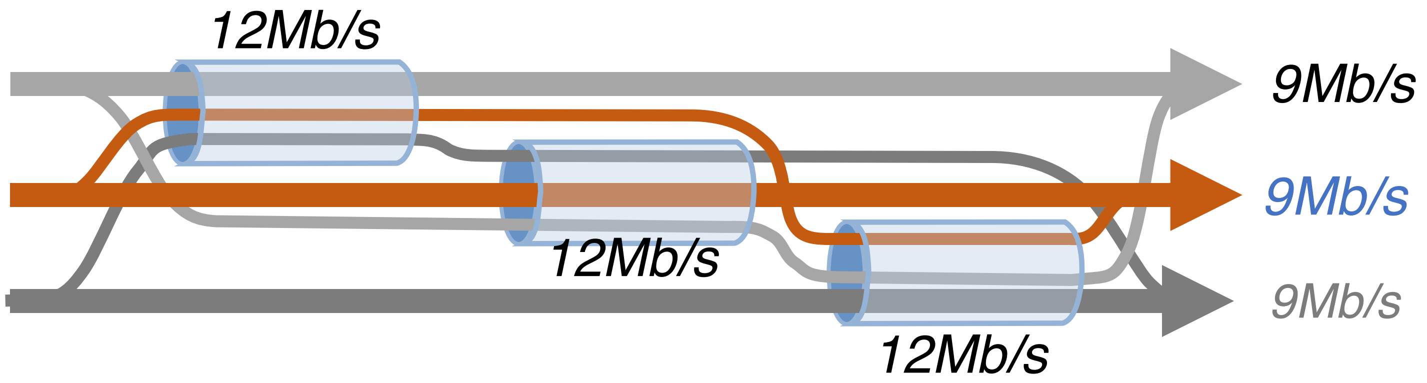
Use Efficient Paths



Equal-window TCP (EWTCP): split flow traffic 1:1 over paths

Achieve fairness using $W_s := W_s + a$ (RTT), $a < 1$

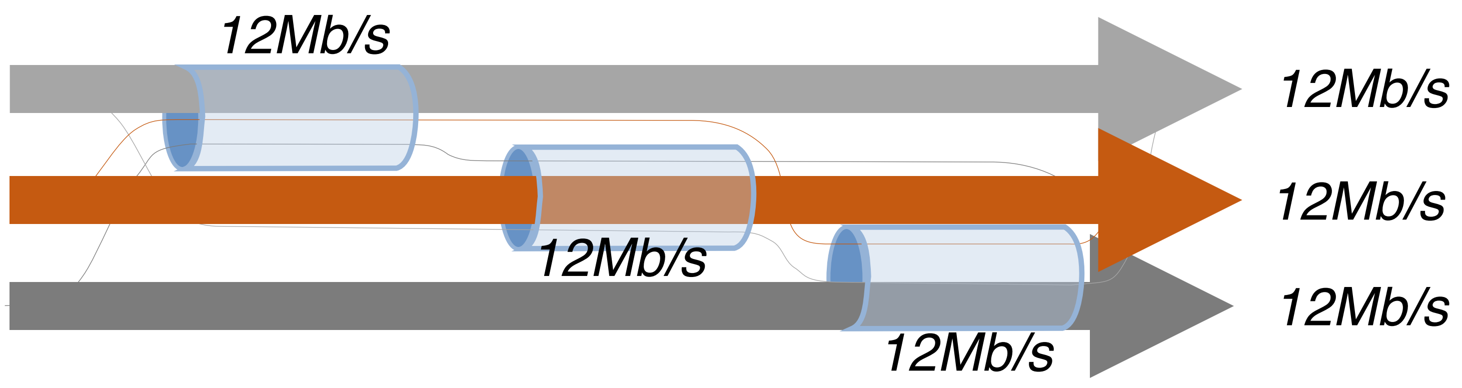
Use Efficient Paths



Move some traffic to better paths:

What if each flow split its traffic 2:1?

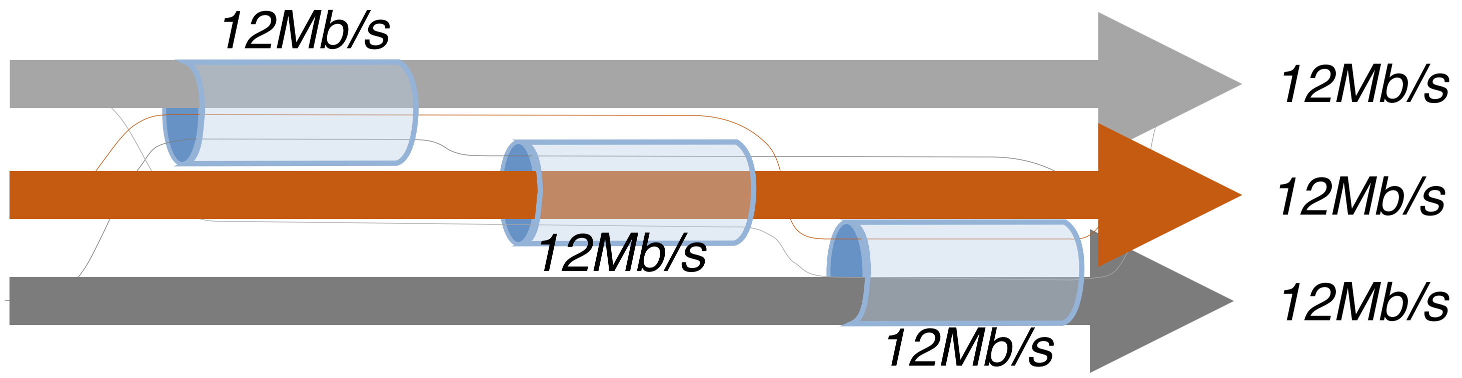
Use Efficient Paths



Best: Each connection on its one-hop path

→ Least congested path

Use Efficient Paths

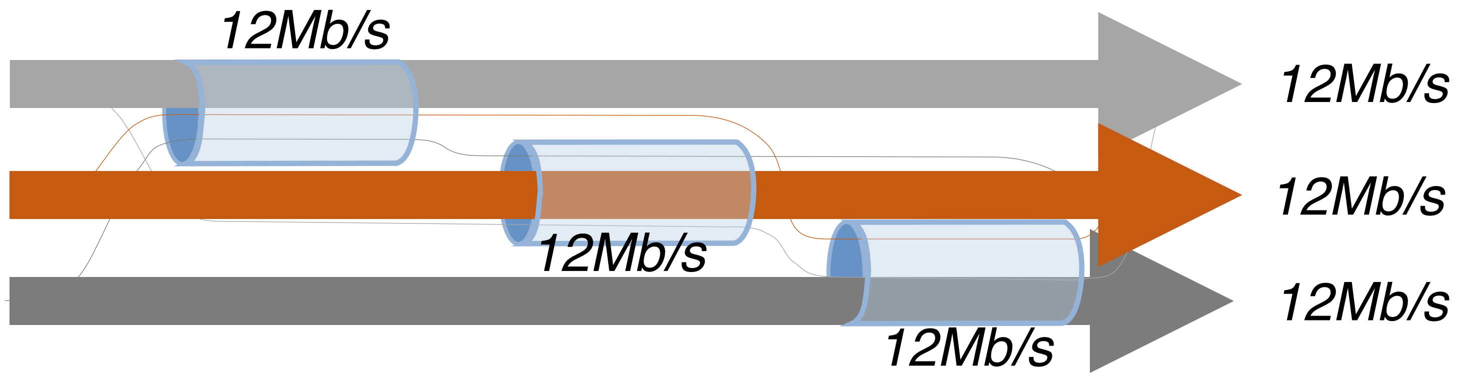


Goal: Get each flow to send all its traffic on the least-congested path

Achieve by coupling the subflow TCP window updates

$W_s := W_s + 1/W_{total}$ (ack), $W_s := W_s - W_{total}/2$ (drop)

Use Efficient Paths

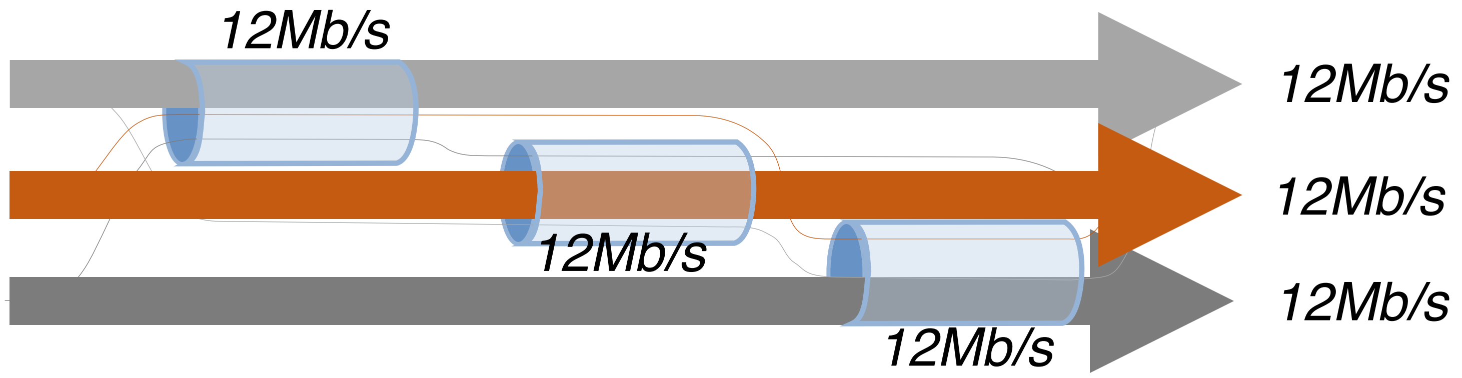


$W_s := W_s + 1/W_{total}$ (ack), $W_s := W_s - W_{total}/2$ (drop)

Consequence: the more drops a subflow sees, the faster its window W_s reduces (note: increments same across all paths)

More lossy paths have zero window in the limit

Use Efficient Paths

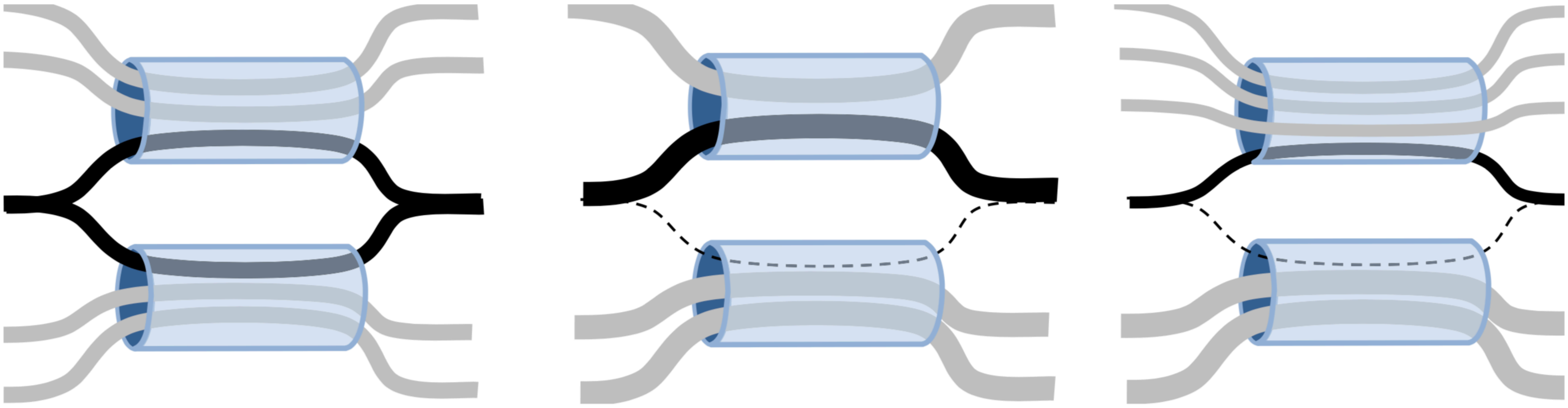


$W_s := W_s + 1/W_{total}$ (ack), $W_s := W_s - W_{total}/2$ (drop)

Consequence: Remaining paths have balanced loss rate ➔ equal window if RTTs same

If loss not balanced, window would drop to zero

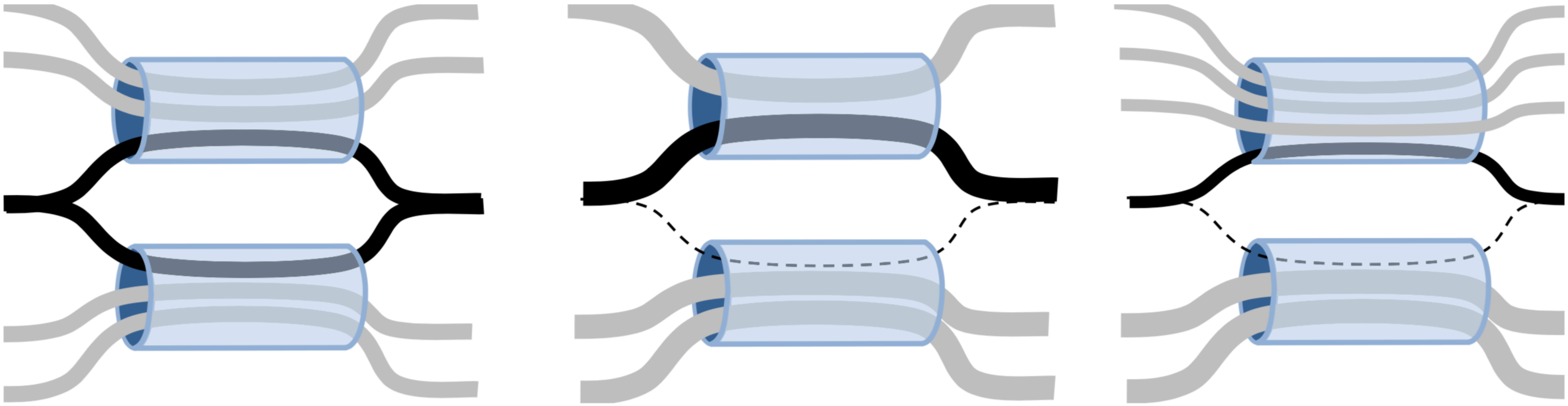
Coupled CC can get trapped



Keep a little traffic on each path (even if congested) to probe for capacity always

i.e., keep your options open ☺

Coupled CC can get trapped



Semi-coupled TCP

$W_s := W_s + a/W_{total}$ (ack), $W_s := W_s - W_s/2$ (drop)

Compare with coupled:

$W_s := W_s + 1/W_{total}$ (ack), $W_s := W_s - W_{total}/2$ (drop)

Goal #3: Be Fair Compared to TCP

- Least-congested paths may not be best
    - Low loss rate, but low throughput due to differences in RTT

- Example: Two paths
    - WiFi: high loss, low RTT
    - Cellular: low loss, high RTT

- Using the least-congested path
    - Choose the cellular path, due to low loss
    - But, the RTT is high
    - So throughput is low!

- Formalize fairness requirement using actual throughput

Be Fair Compared to TCP

- To be fair, Multipath TCP should give a connection at least as much throughput as it would get with a single-path TCP on the best of its paths, given the same loss rate
  - Ensure incentive for deploying MPTCP

- A Multipath TCP should take no more capacity on any path (or collection of paths) than if it was a single-path TCP flow using the best of those paths, given the same loss rate
  - Do no harm

Achieving These Goals

- Regular TCP
  - Maintain a congestion window w
  - On an ACK, increase by 1/w (increase 1 per window)
  - On a loss, decrease by w/2
- MPTCP
  - Maintain a congestion window per path w_s
  - On an ACK on path s, increase w_s
  - On a loss on path s, decrease by $w_s/2$
- How much to increase w_s on an ACK?
  - If s is the only path at that bottleneck, increase by $1/w_s$

If Multiple Paths Share Bottleneck?

- Don’t take any more bandwidth on a link than the best of the TCP paths would
  - But, where might the bottlenecks be?
  - Multiple paths might share the same bottleneck
  - This is hard to know across the Internet

- So, consider all possible subsets of the paths
  - Set R of paths
  - Subset S of R that includes path r

- E.g., consider path 3
  - Suppose paths 1, 3, and 4 share a bottleneck
  - … but, path 2 does not
  - Then, we care about S = {1,3,4}

Achieving These Goals

- What is the *best* of these subflows achieving?
  - Path s is achieving throughput of w_s/RTT_s
  - So best path is getting $\max_s(w_s/RTT_s)$
- What *total* bandwidth are these subflows getting?
  - Across *all* subflows sharing that bottleneck
  - Sum over s in S of w_s/RTT_s
- Consider the *ratio* of the two
  - Increase by less if many subflows are sharing
- And pick the results for the set S with min ratio
  - To account for the *most* paths sharing a bottleneck

$$\frac{\max_{s\in S} w_s/\mathrm{RTT}_s^2}{\left(\sum_{s\in S} w_s/\mathrm{RTT}_s\right)^2}$$

MPTCP Implementation

Implementation Issues

- Buffer space: per-subflow or shared for entire connection?
- Reassembly across multiple paths
  - Different sequence spaces across subflows
  - But shared flow control
  - Ensure packets across subflows reach at approx. the same time
- Middleboxes
  - Avoid impact due to rewritten sequence numbers
- Initiating new subflow: new TCP flag; auth token