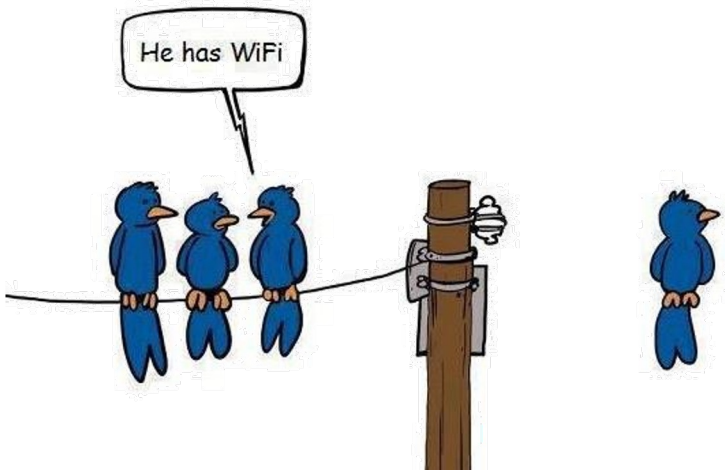
COM-405: Mobile Networks

Lecture 14.0: Multipath TCP

Haitham Hassanieh

slides shamelessly stolen from Prof. JP Hubaux & others



EPFL

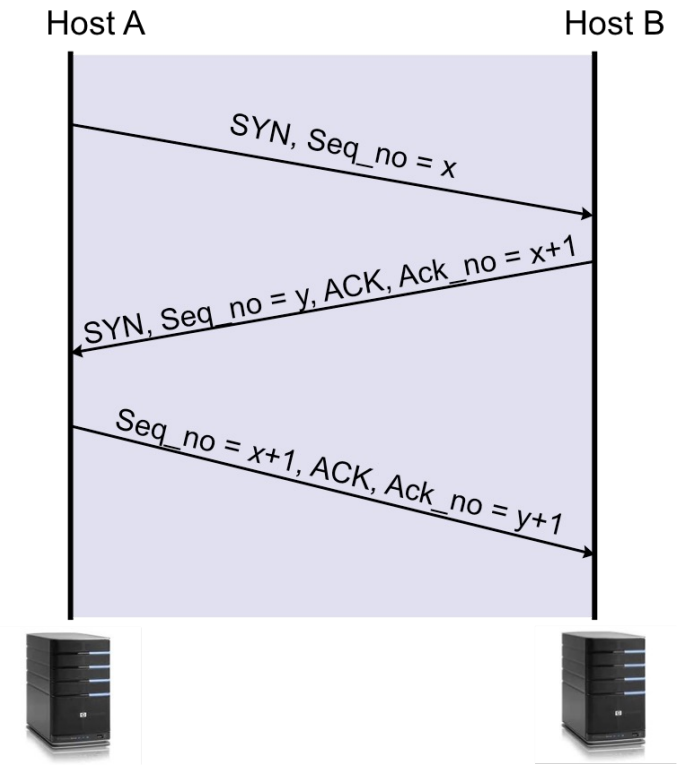
Laboratory of SEnsing & Networking Systems

Reminder: Transmission Control Protocol (TCP)

- Reliable, in-order data delivery service
- Single path
  - **tied** to the **source** and **destination** addresses of the endpoints
- Flow control
- Congestion avoidance and control
- End-to-end semantics

TCP Connection Setup

- Three-way handshake



TCP Data Transmission

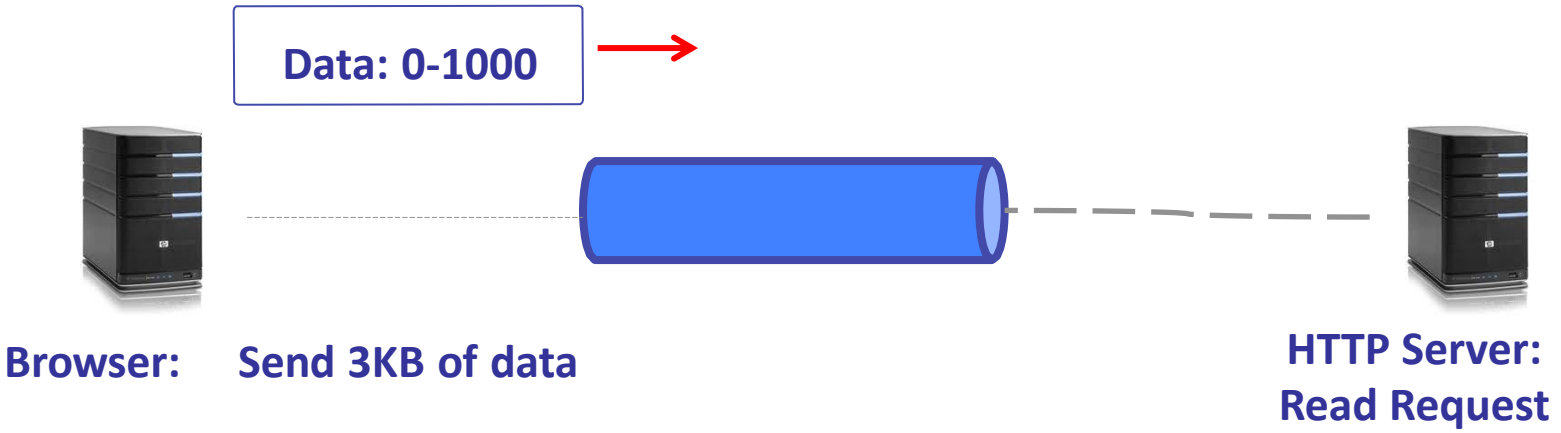
- Assuming a browser wants to send 3KB of data in 3 TCP segments to an HTTP Server

Browser: **Send 3KB of data**

HTTP Server: Read Request

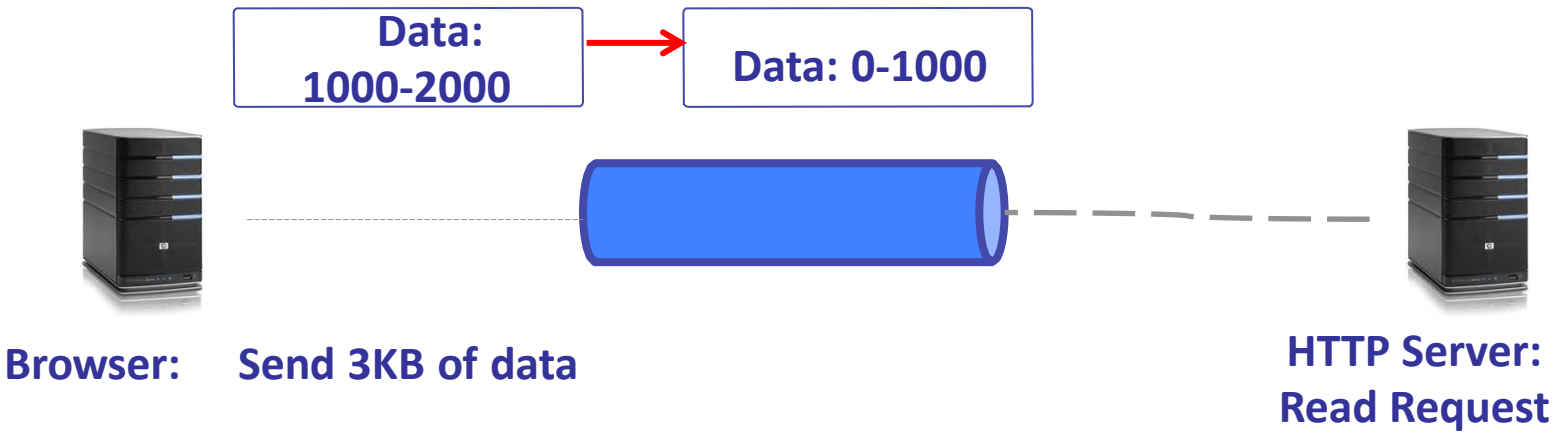
TCP Data Transmission

- Assuming a browser wants to send 3KB of data in 3 TCP segments to an HTTP Server



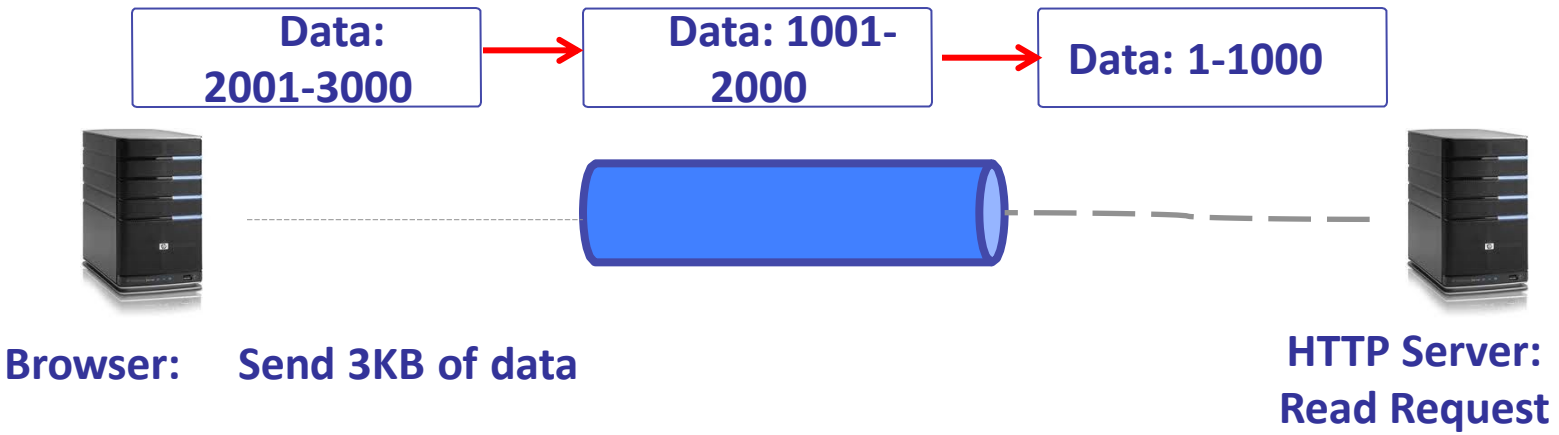
TCP Data Transmission

- Assuming a browser wants to send 3KB of data in 3 TCP segments to an HTTP Server



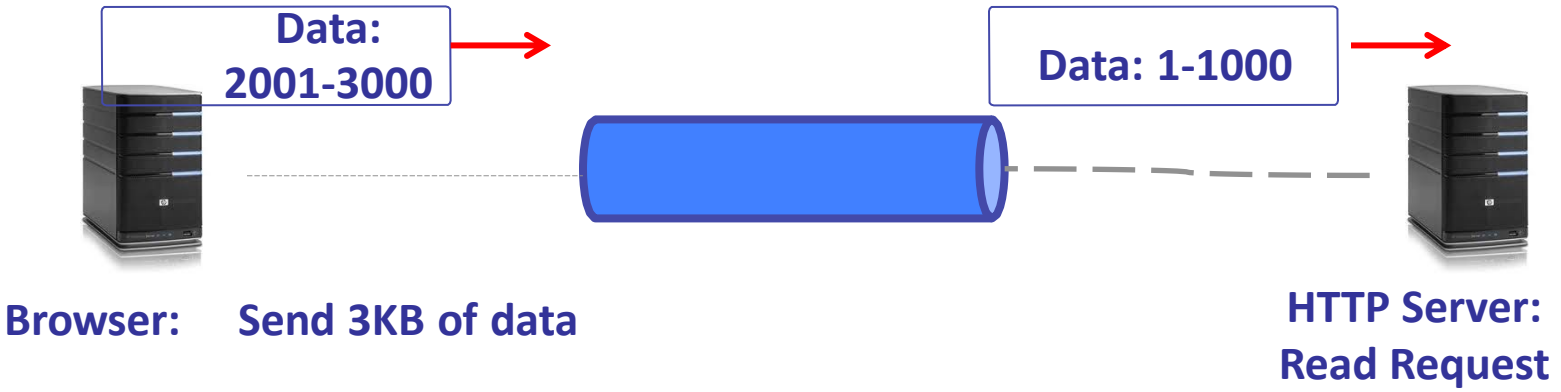
TCP Data Transmission

- Assuming a browser wants to send 3KB of data in 3 TCP segments to an HTTP Server



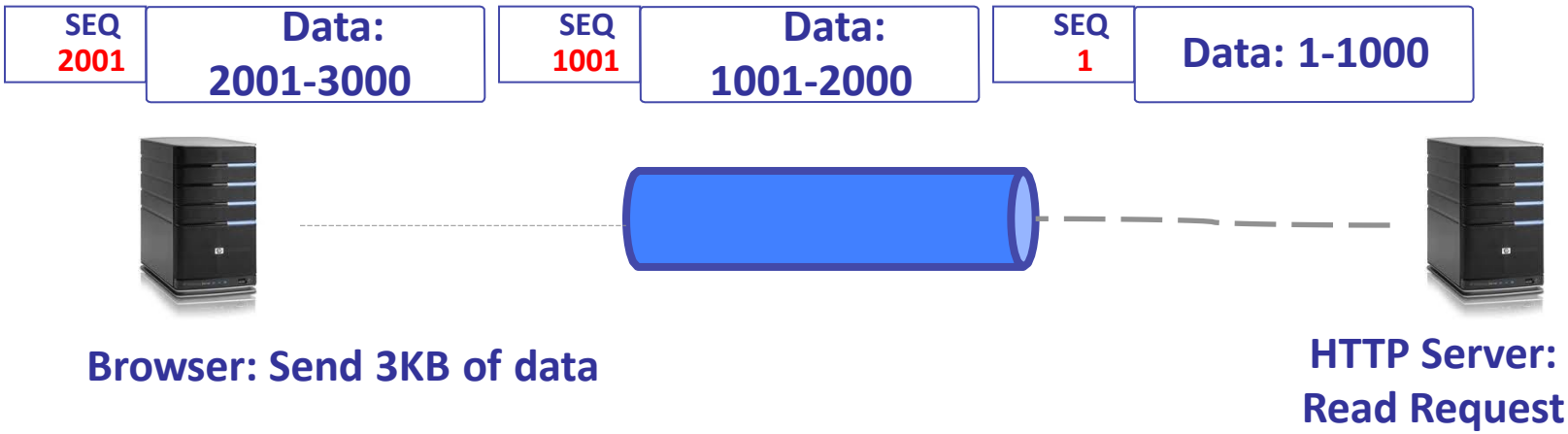
TCP: Lost Packets

- Assuming a browser wants to send 3KB of data in 3 TCP segments to an HTTP Server
  - How do the browser and the server know that a packet is lost?



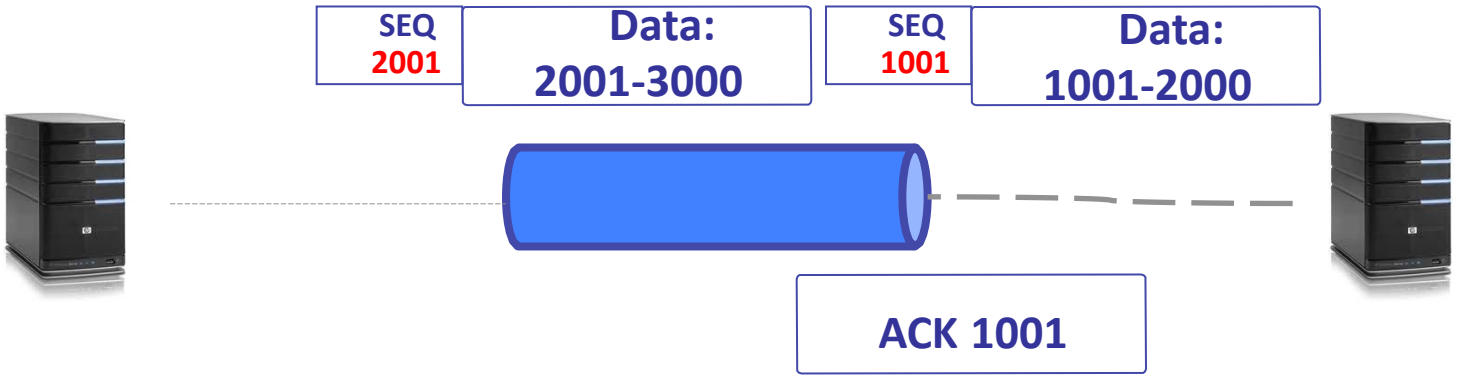
TCP: Sequence #s and ACKs

- Assuming a browser wants to send 3KB of data in 3 TCP segments to an HTTP Server
  - How do the browser and the server know that a packet is lost?
    - Based on the sequence numbers and ACKs



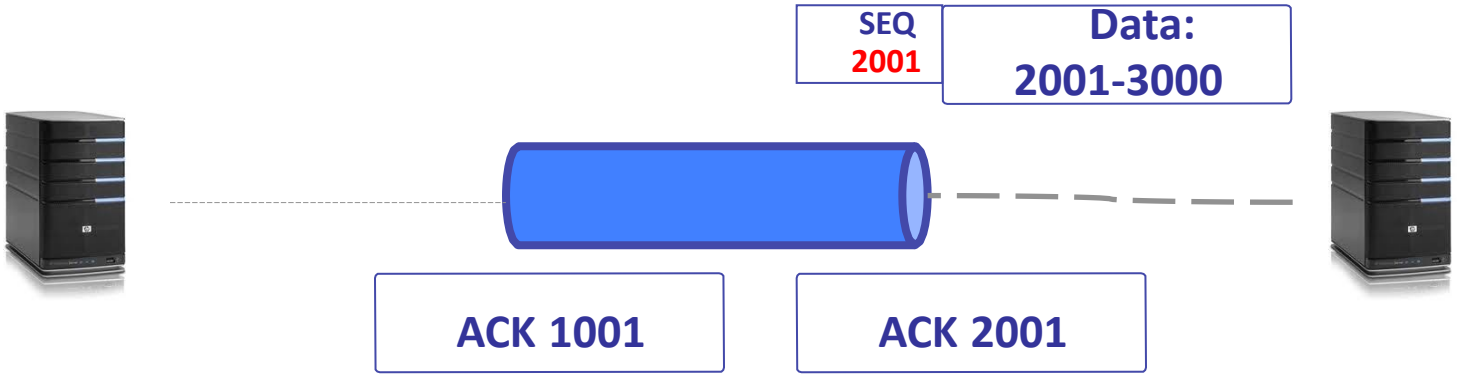
TCP: Sequence #s and ACKs

- Assuming a browser wants to send 3KB of data in 3 TCP segments to an HTTP Server
  - How do the browser and the server know that a packet is lost?
    - Based on the sequence numbers and ACKs



TCP: Sequence #s and ACKs

- Assuming a browser wants to send 3KB of data in 3 TCP segments to an HTTP Server
  - How do the browser and the server know that a packet is lost?
    - Based on the sequence numbers and ACKs



TCP: Sequence #s and ACKs

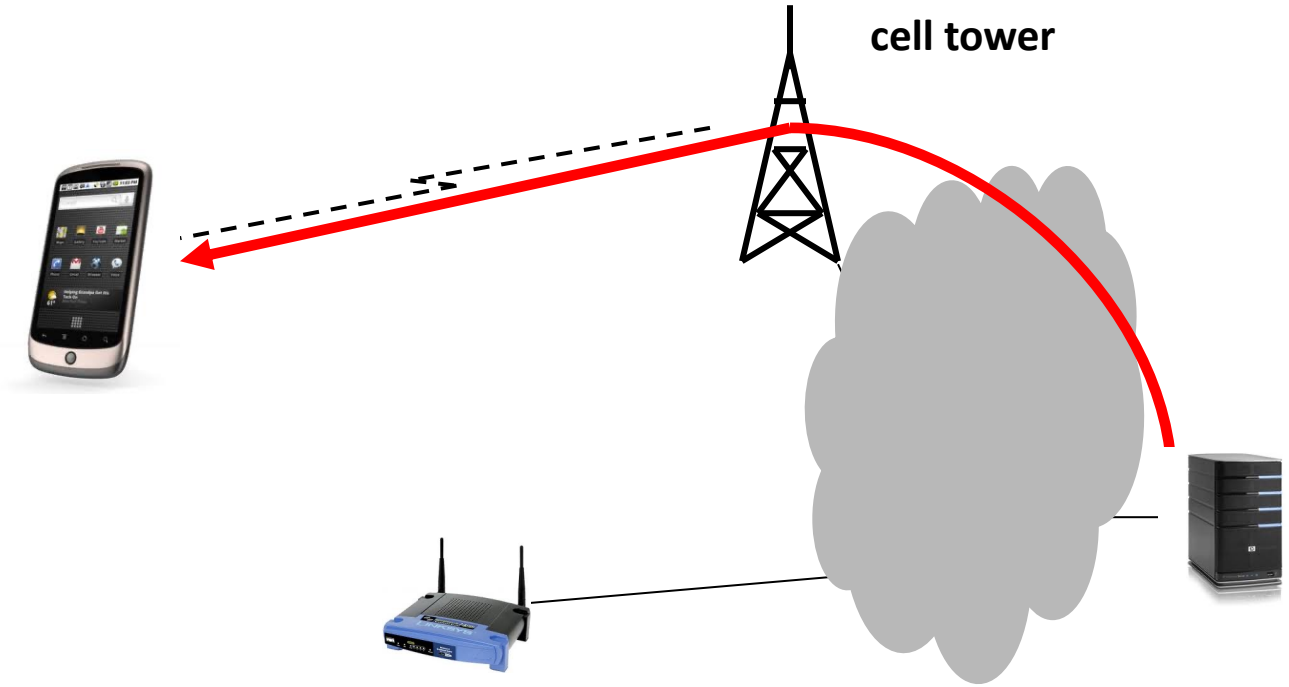
- Assuming a browser wants to send 3KB of data in 3 TCP segments to an HTTP Server
  - How do the browser and the server know that a packet is lost?
    - Based on the sequence numbers and ACKs

Browser: **Send 3KB of data**

HTTP Server: Read Request

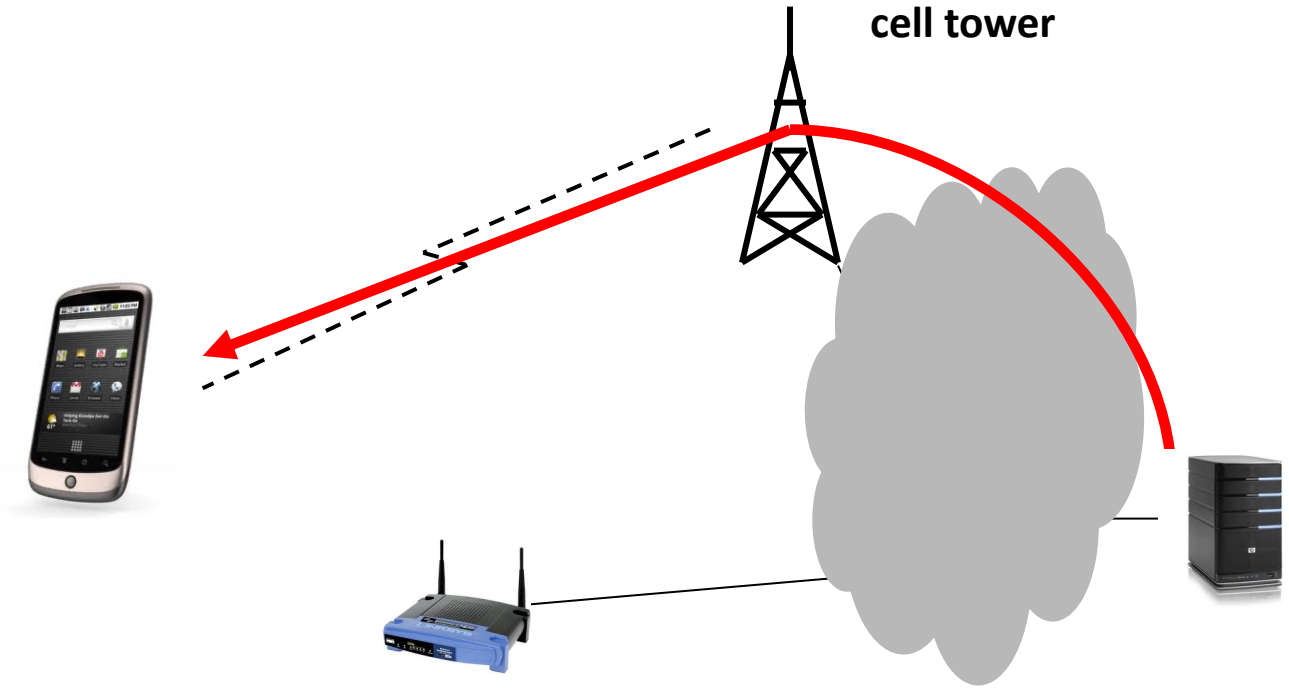
TCP for Wireless and Mobile Networks

- Imagine a phone with cellular network and WiFi, moving around a campus
  - Would like to use the cellular network as semi-reliable baseline service, supplemented by WiFi when available, and switching between WiFi APs as they come and go



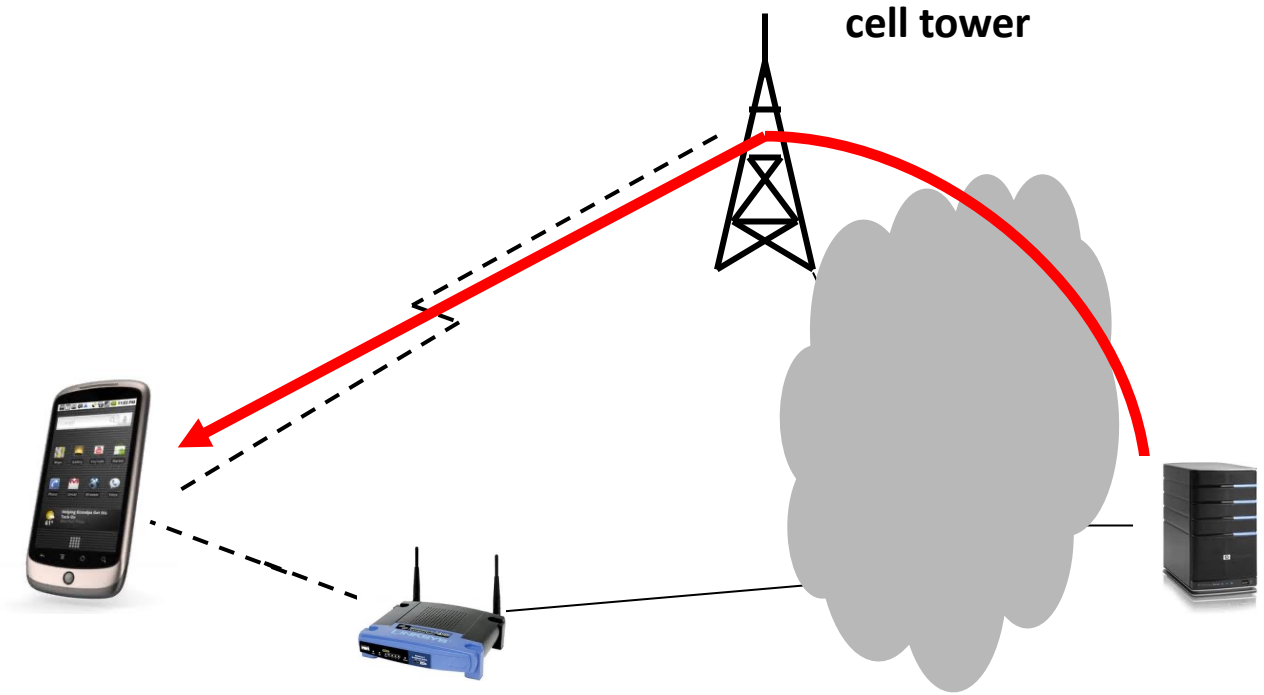
TCP for Wireless and Mobile Networks

- Imagine a phone with cellular network and WiFi, moving around a campus
  - Would like to use the cellular network as semi-reliable baseline service, supplemented by WiFi when available, and switching between WiFi APs as they come and go



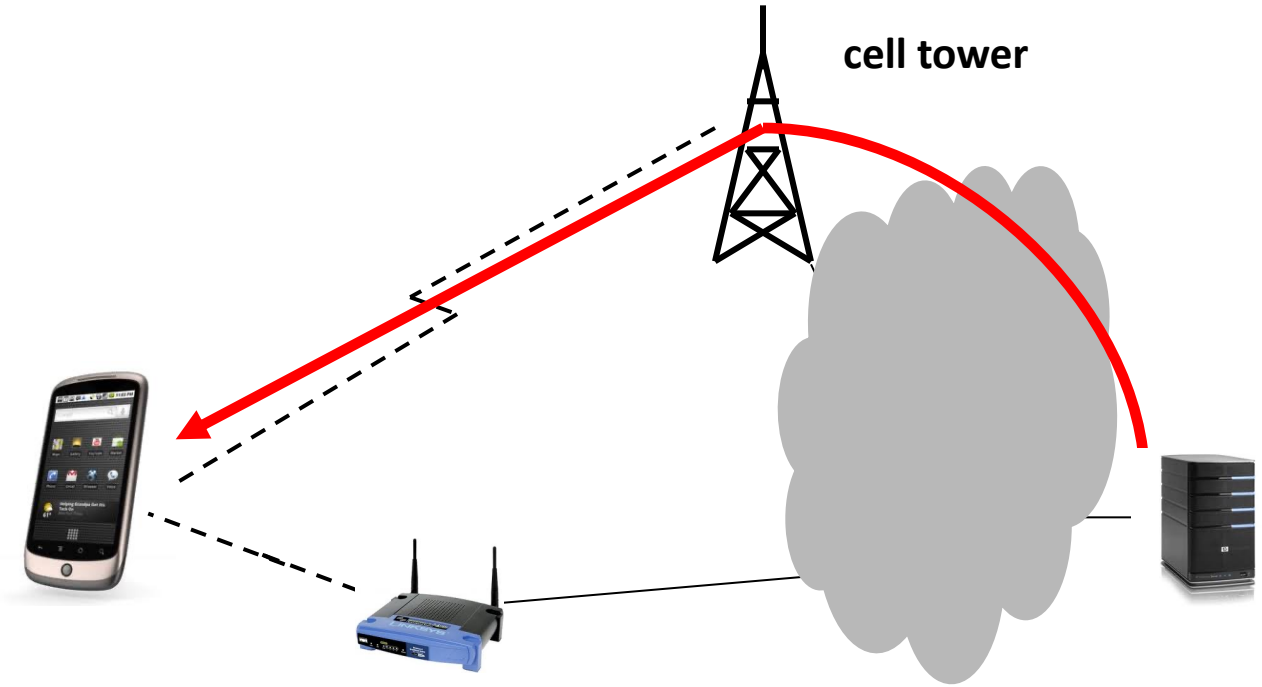
TCP for Wireless and Mobile Networks

- Imagine a phone with cellular network and WiFi, moving around a campus
  - Would like to use cellular network as semi-reliable baseline service, supplemented by WiFi when available, and switching between WiFi APs as they come and go



TCP for Wireless and Mobile Networks

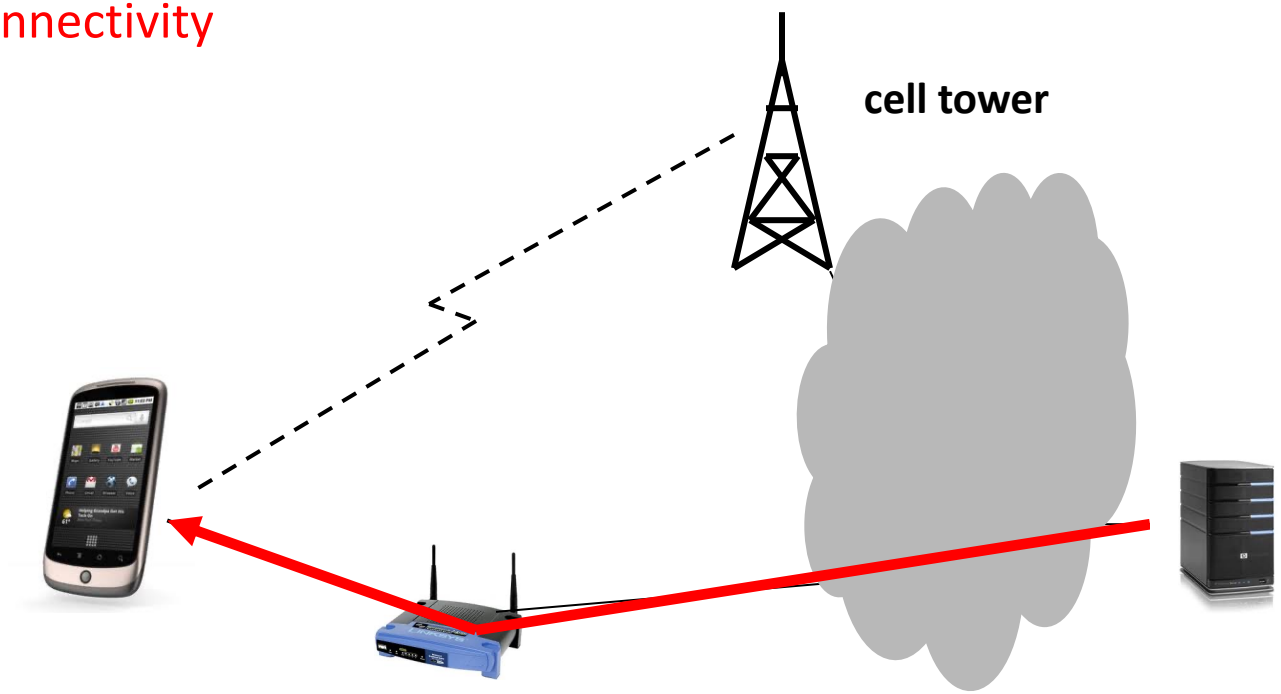
- Imagine a phone with cellular network and WiFi, moving around a campus
  - Would like to use cellular network as semi-reliable baseline service, supplemented by WiFi when available, and switching between WiFi APs as they come and go



Offload to WiFi

TCP for Wireless and Mobile Networks

- Imagine a phone with cellular network and WiFi, moving around a campus
  - Would like to use the cellular network as semi-reliable baseline service, supplemented by WiFi when available, and switching between WiFi APs as they come and go
  - Conventional TCP is not designed around such agile network connectivity

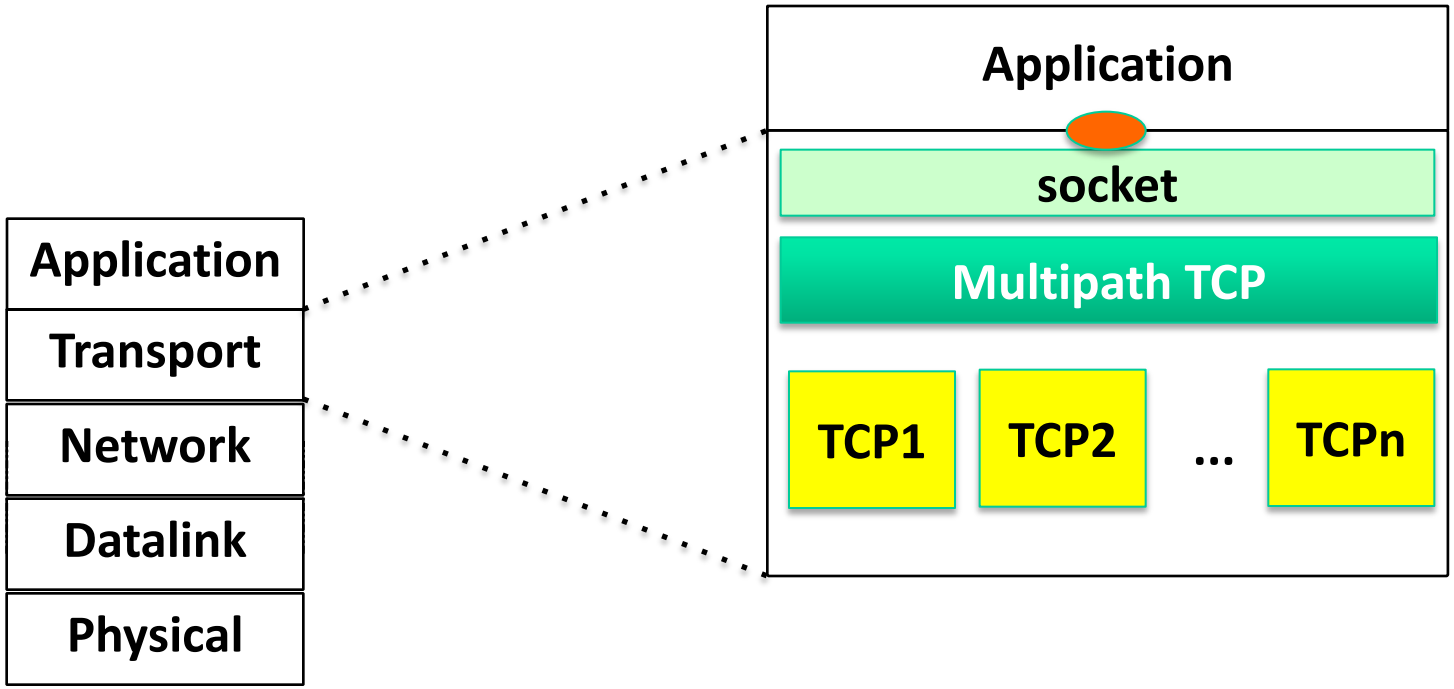


All ongoing TCP connections die

Bringing Multipath TCP to the End Host

- MPTCP is a modification of TCP that presents a regular TCP interface to applications, while spreading data across several subflows
  - Specified in RFC 6824, obsoleted by **RFC 8684** (March 2020)
  - Not a new idea; originally proposed more than 15 years ago
  - MPTCP goes further to solve issues of fairness when competing with regular TCP and deployment issues
- An **MPTCP connection is composed of one or more regular TCP subflows that are combined**
  - Each host maintains states that glue the TCP subflows of a MPTCP connection together
  - Each TCP subflow is sent over a single path and appears like a regular TCP connection along this path
- Benefits
  - Higher **throughput**
  - **Failover** from one path to another
  - Seamless **mobility**

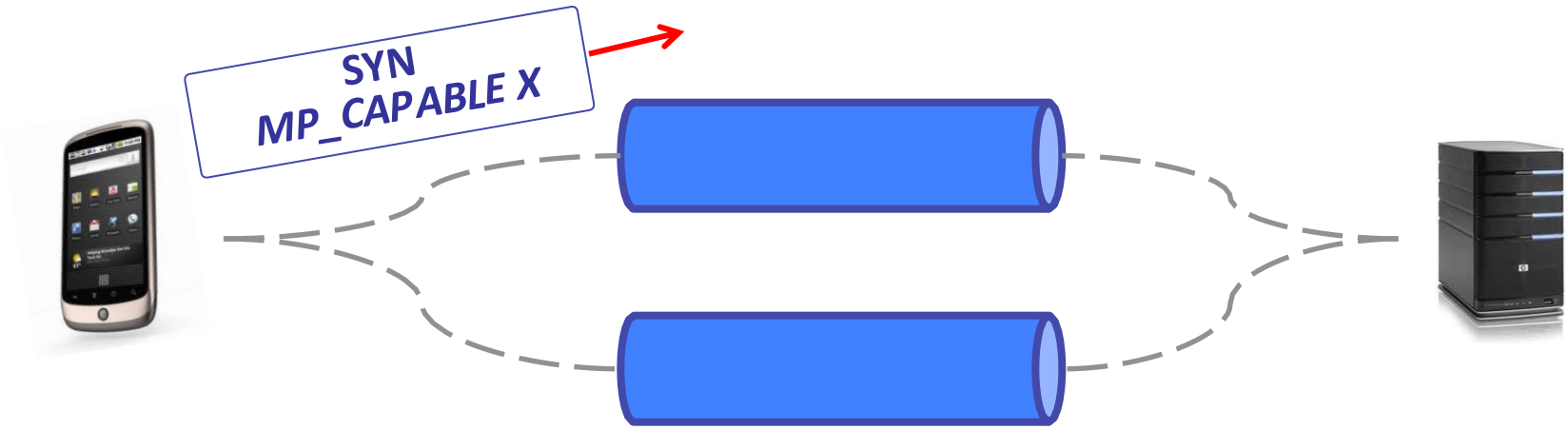
MPTCP Architecture



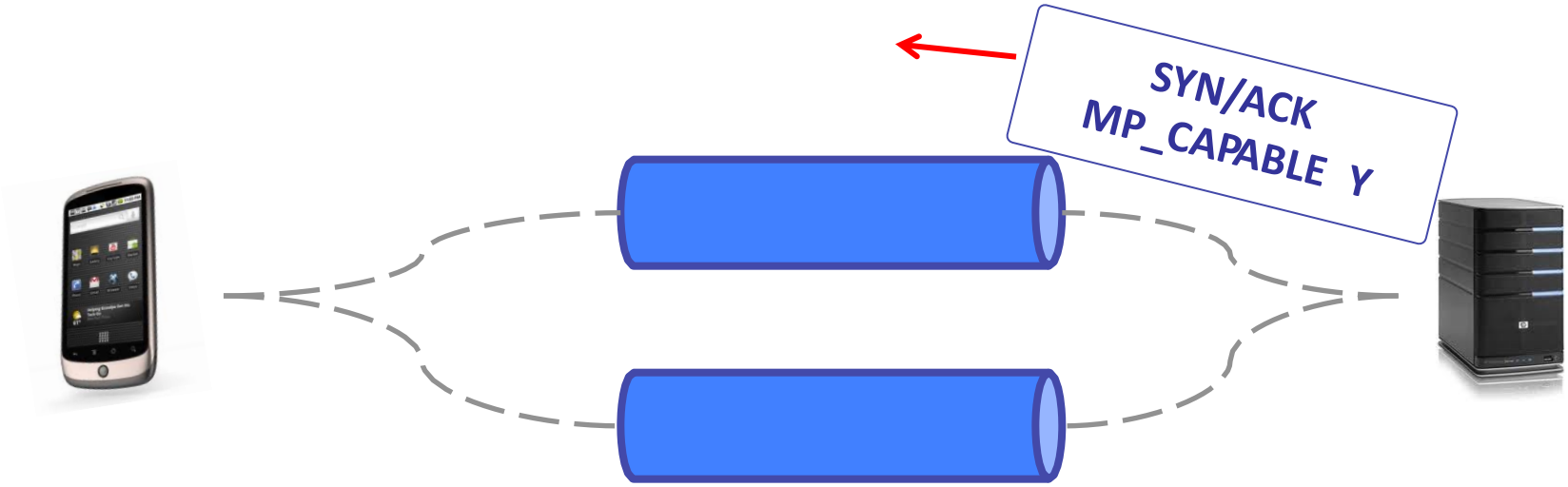
How to set up an MPTCP session?

- Establish the initial subflow
  - Three-way TCP handshake with **MP_CAPABLE** option
- Then additional subflows can be established
  - Three-way TCP handshake with **MP_JOIN** option
- Each subflow looks similar to a regular TCP connection
- Which subflows can be associated to a MPTCP connection?
  - At least one of the elements of the four-tuple needs to differ between two subflows
    - Local IP address
    - Remote IP address
    - Local port
    - Remote port

MPTCP Operation – Setting up the Initial Subflow



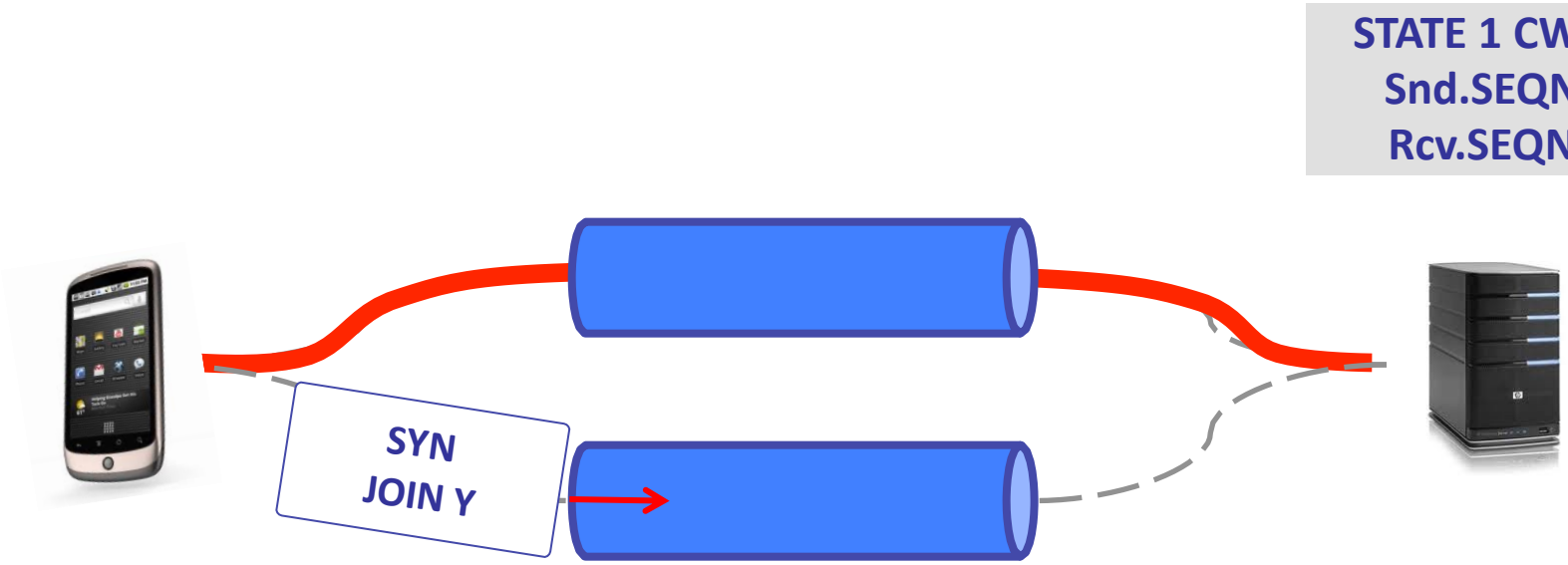
MPTCP Operation – Setting up the Initial Subflow



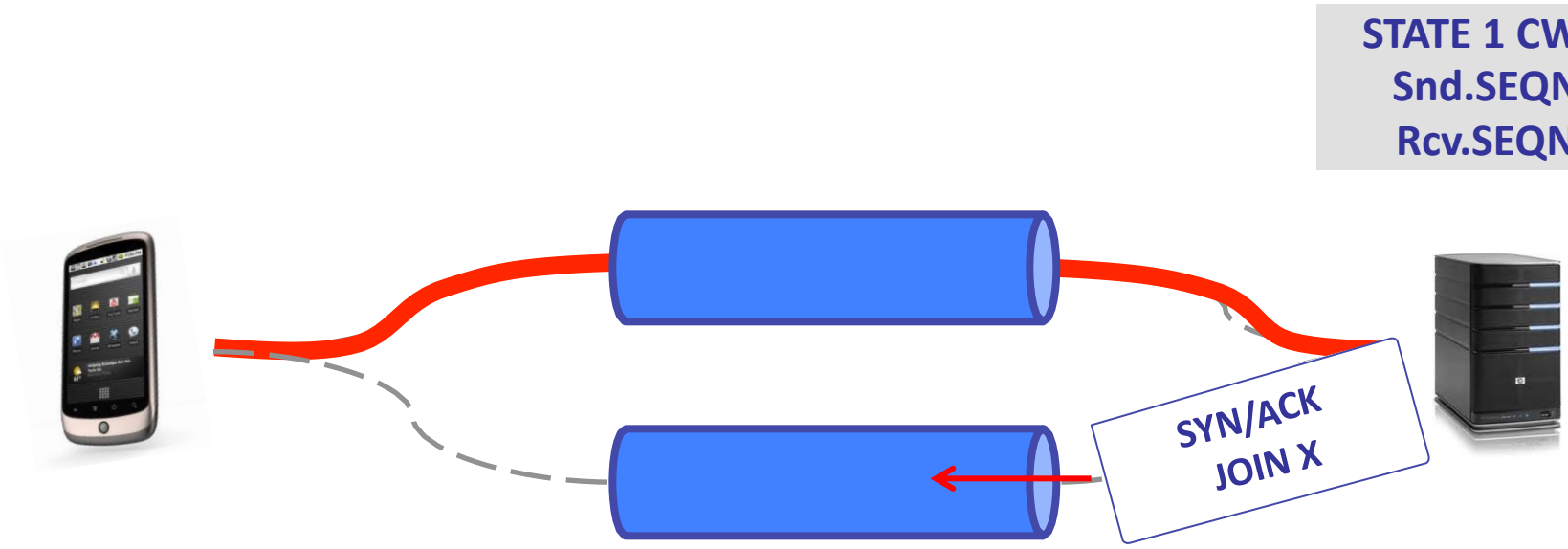
MPTCP Operation – Setting up the Initial Subflow

STATE 1 CWND
Snd.SEQNO
Rcv.SEQNO

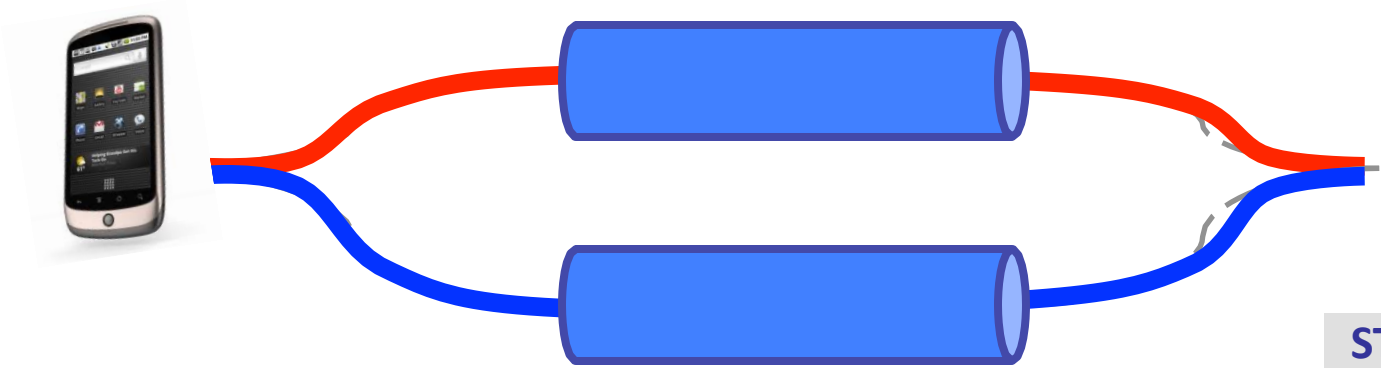
MPTCP Operation – Adding an Additional Subflow



MPTCP Operation – Adding an Additional Subflow



MPTCP Operation – Adding an Additional Subflow

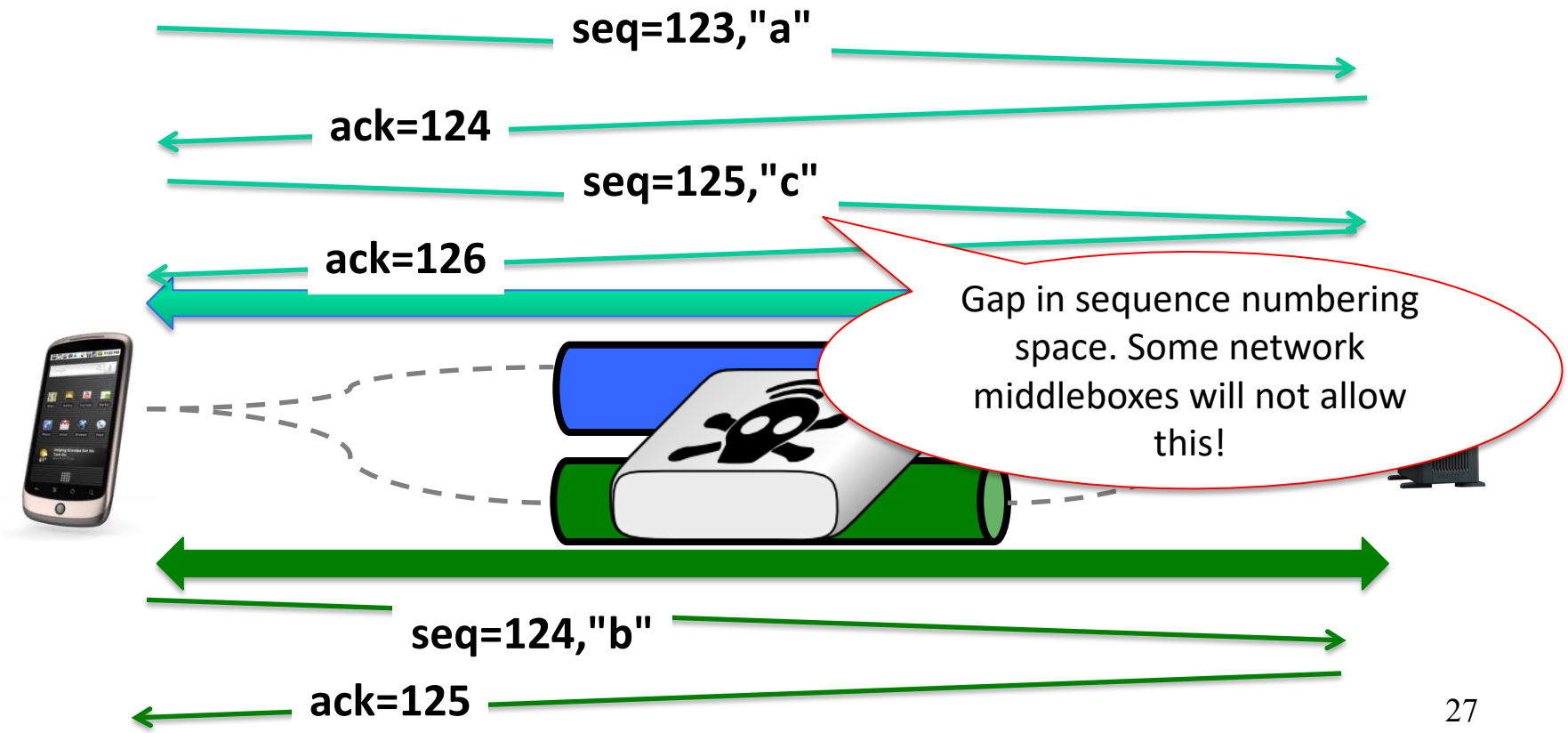


STATE 1 CWND
Snd.SEQNO
Rcv.SEQNO

STATE 2 CWND
Snd.SEQNO
Rcv.SEQNO

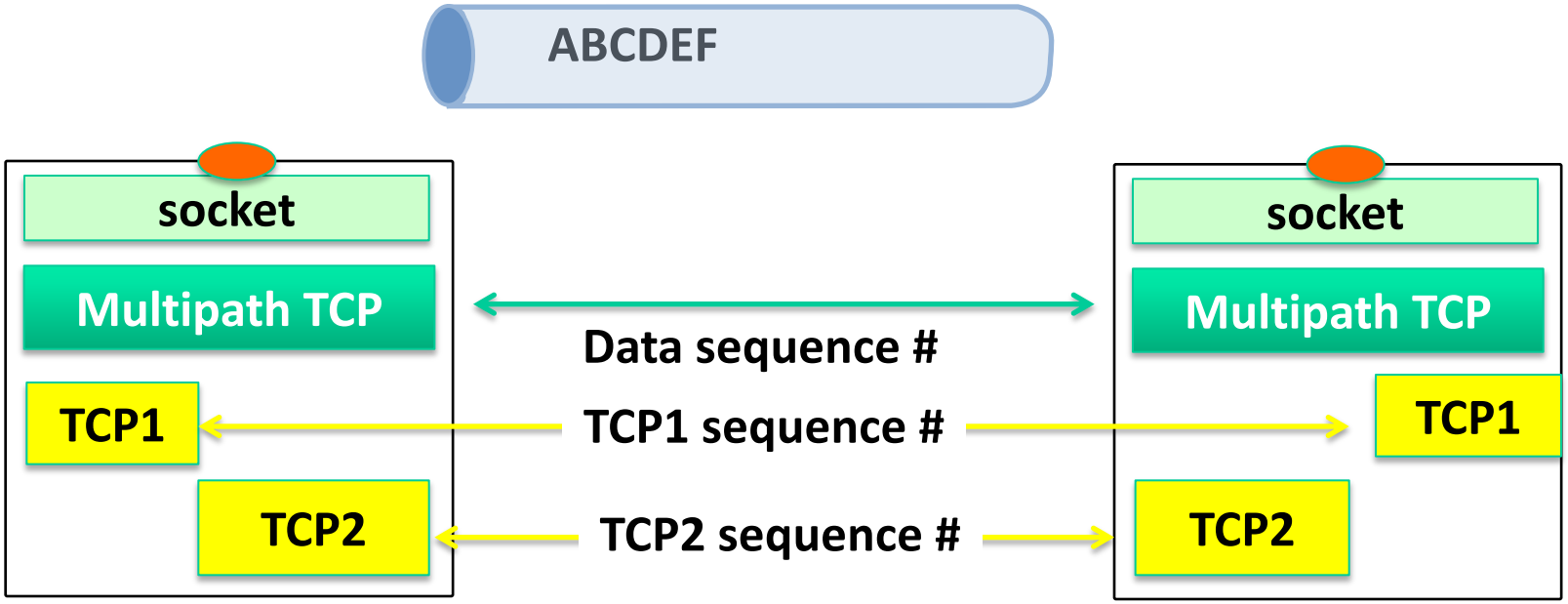
How to Transfer Data?

- Assuming a MPTCP connection with 2 subflows and data for the connection can then be sent over any of the active subflows that has the capacity to take it



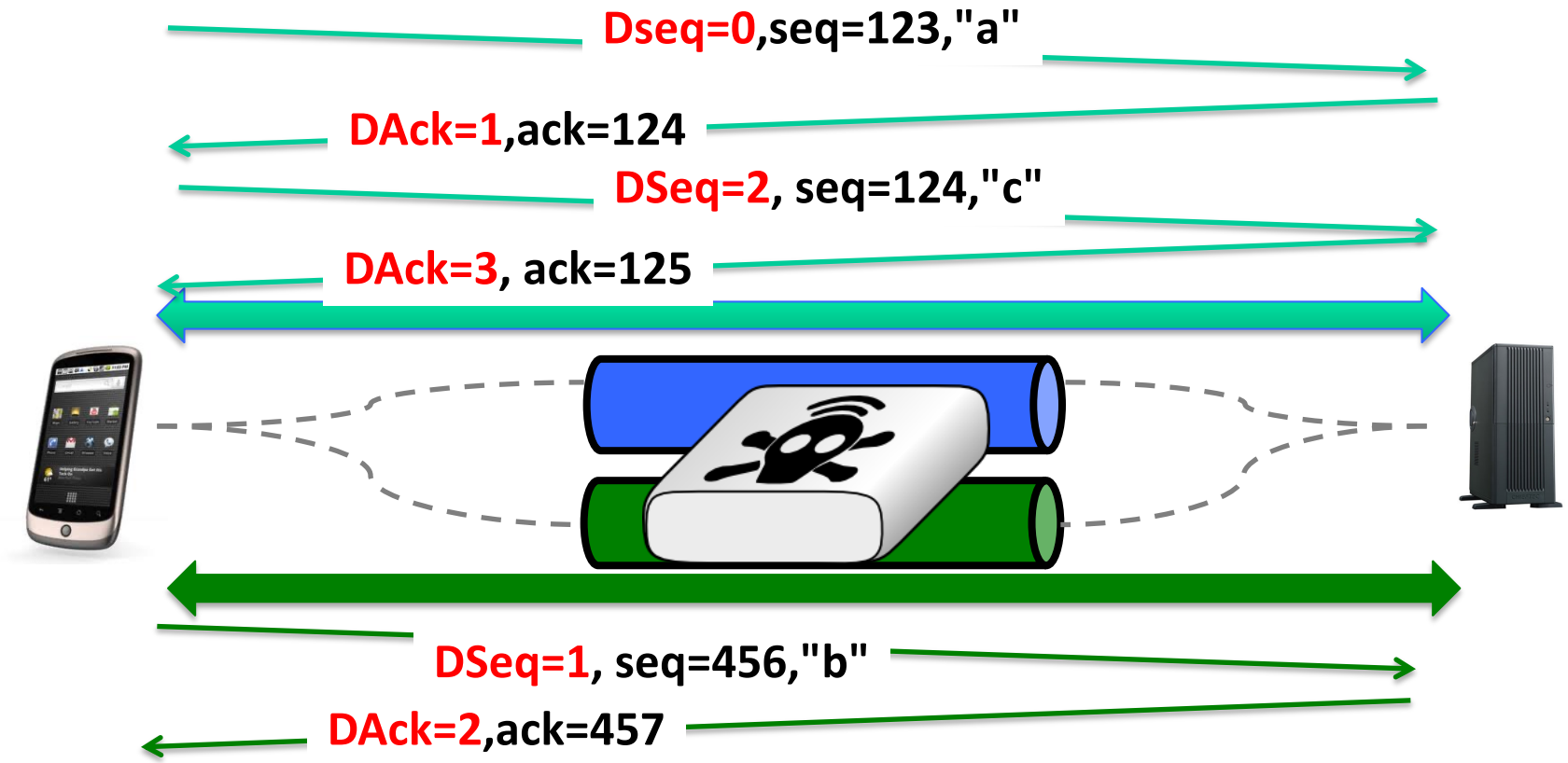
MPTCP Data Transfer

- To solve the problem of gaps in the sequence numbering space, MPTCP uses two levels of sequence numbers



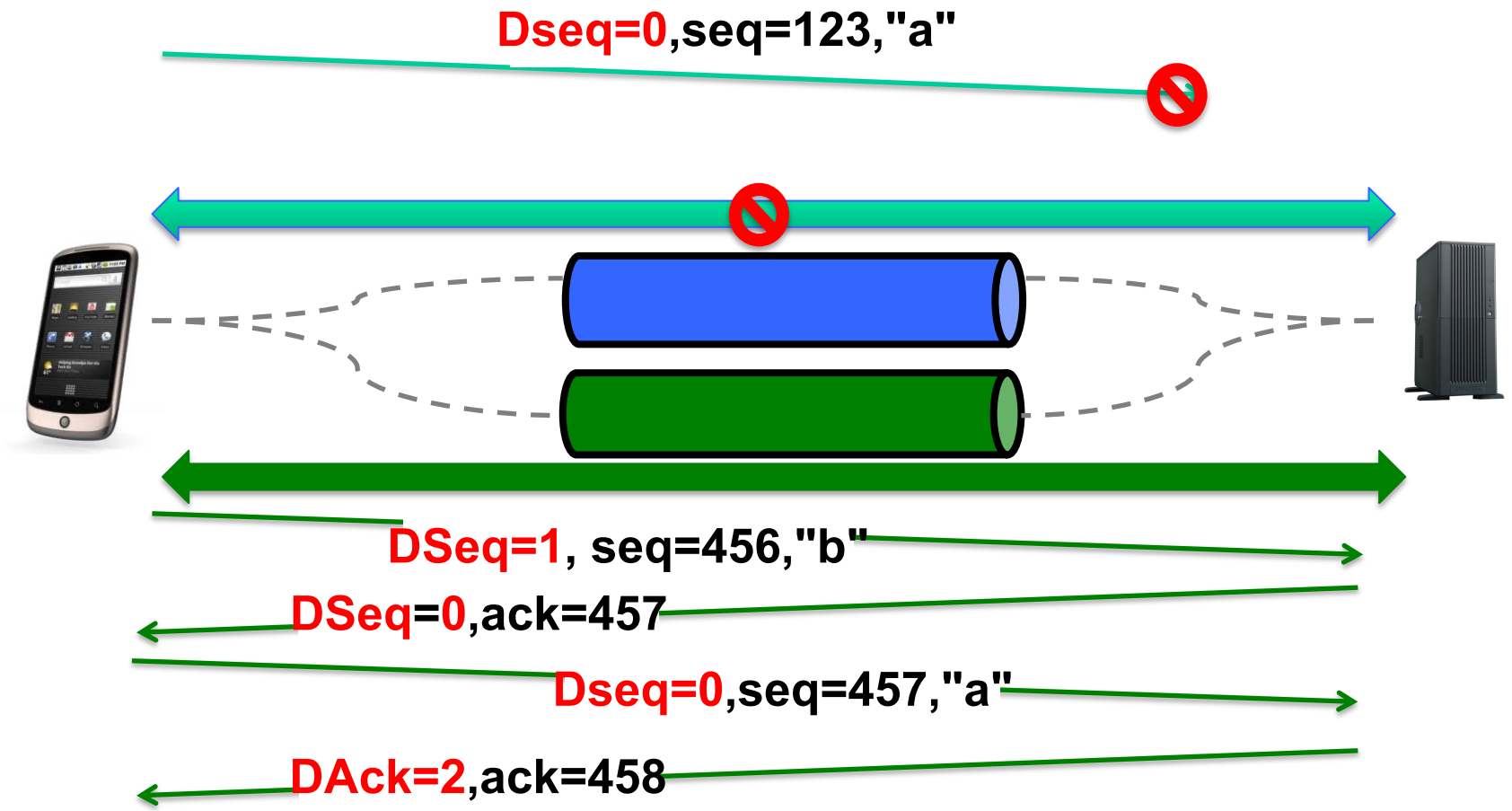
MPTCP Data Transfer

- Dseq is the data sequence number (SQN)
- seq is the additional SQN carried inside the TCP option
  - ensures that the segments sent on any given subflow have consecutive SQNs (to not upset the middleboxes)



seq is the sequence for a flow
dseq is for the MPTCP connection

What Happens When a TCP Subflow Fails ?

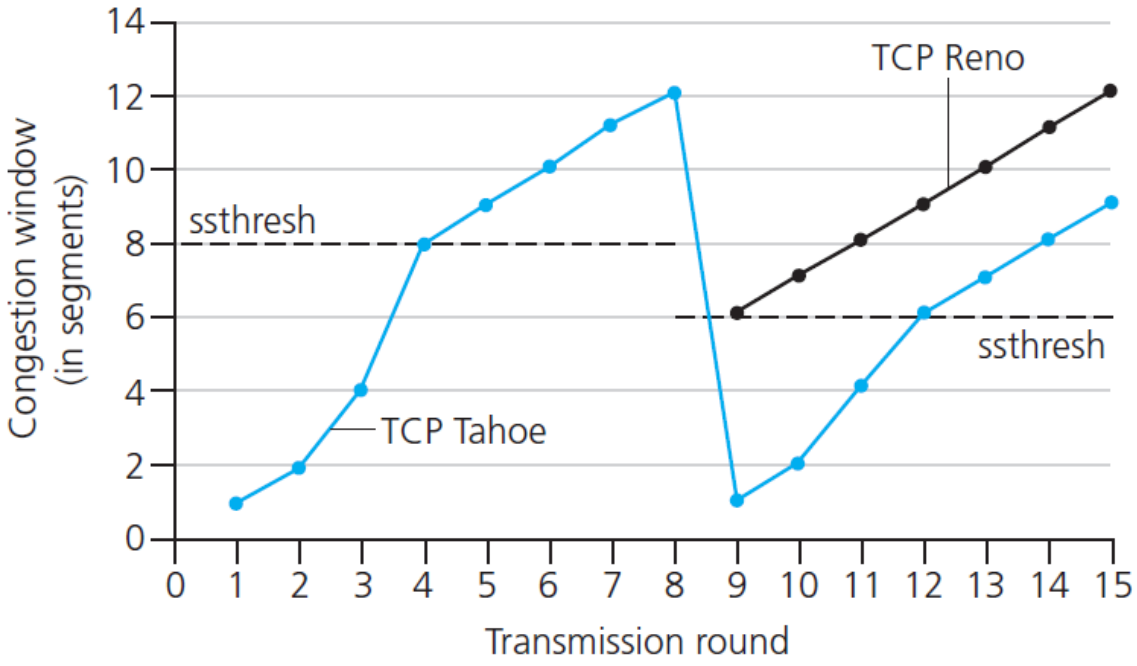


Retransmission Heuristics

- Heuristics used by current Linux implementation
  - Fast retransmit is performed on the same subflow as the original transmission
  - Upon timeout expiration, re-evaluate whether the packet could be retransmitted over another subflow
  - Upon loss of a subflow, all the unacknowledged data are retransmitted on other subflows

An Important Question

- How does the congestion control algorithm work?



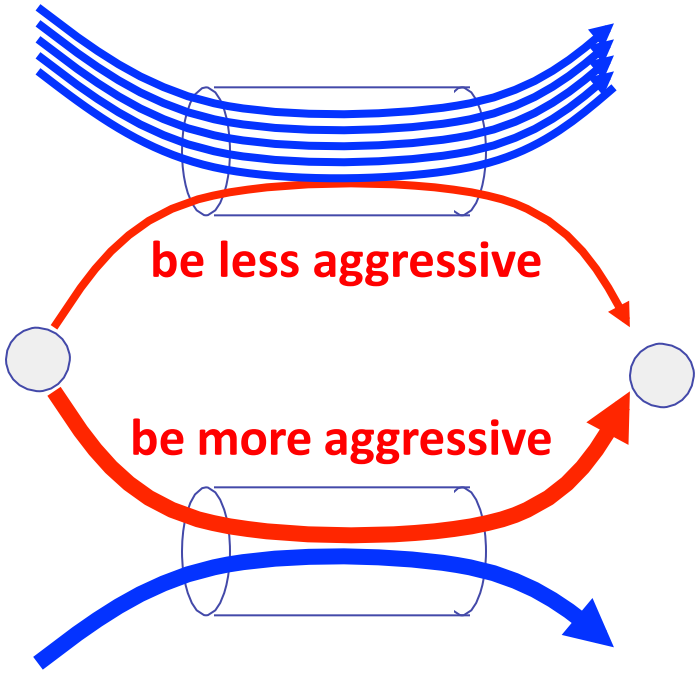
An Important Question

- How does the congestion control algorithm work?

MPTCP Congestion Control

Each path runs its own congestion control, to detect and respond to the congestion it sees.

But link the congestion control parameters, so as to move traffic away from the more congested paths.



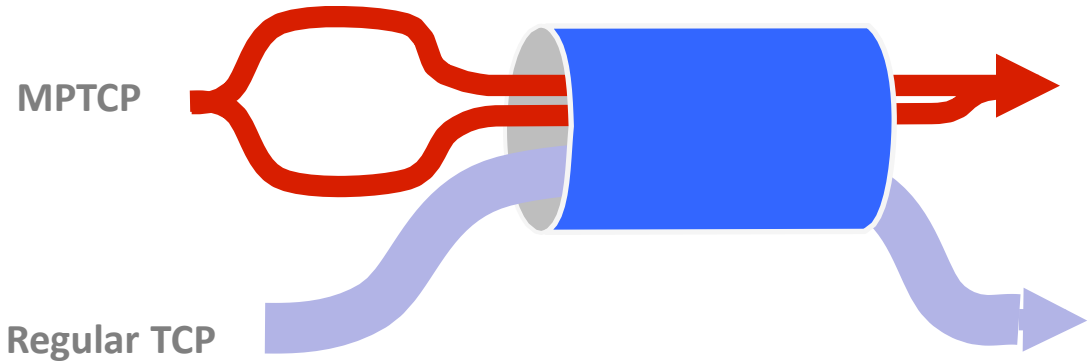
Design Goals for MPTCP

- Goal 1: Be fair to regular TCP
- Goal 2: Use efficient (meaning uncongested) paths
- Goal 3: Perform as well as TCP

We detail them one by one in the next slides

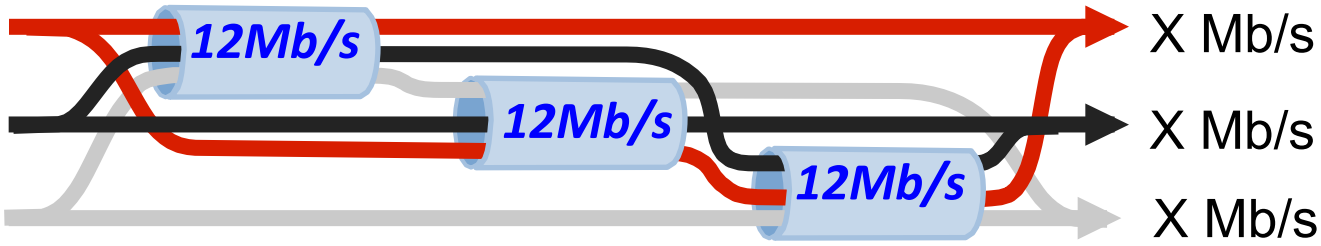
Design Goal 1: Be Fair to Regular TCP

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



Design Goal 2: MPTCP Should Use Efficient Paths

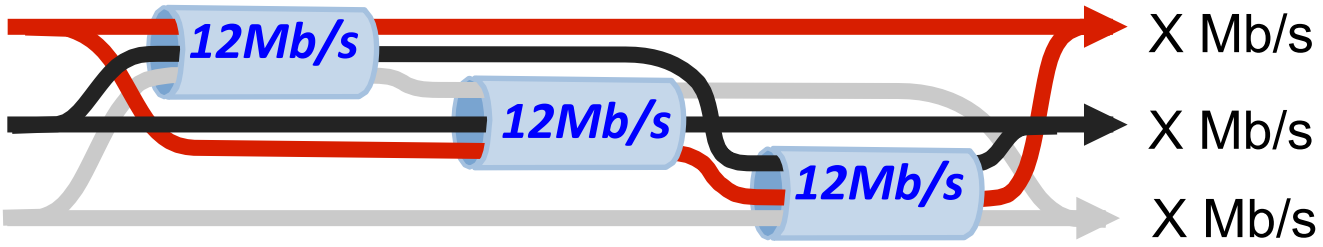
- Assume there are 3 MPTCP connections with the same data rates of X Mb/s, each consisting of 2 subflows



  - Each connection has a choice of a 1-hop and a 2-hop path
  - How should each connection split its traffic between its 1-hop and 2-hop subflows?

Design Goal 2: MPTCP Should Use Efficient Paths

- Assume there are 3 MPTCP connections with the same data rates of X Mb/s, each consisting of 2 subflows

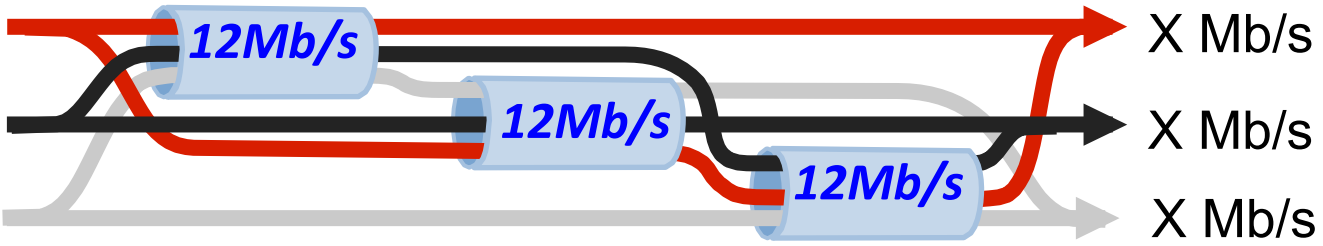


  - If each connection splits its traffic between its 1-hop and 2-hop subflows with ratio 1:1

$X/2 + X/2 + X/2 = 12 \rightarrow X = 8\text{Mb/s}$

Design Goal 2: MPTCP Should Use Efficient Paths

- Assume there are 3 MPTCP connections with the same data rates of X Mb/s, each consisting of 2 subflows

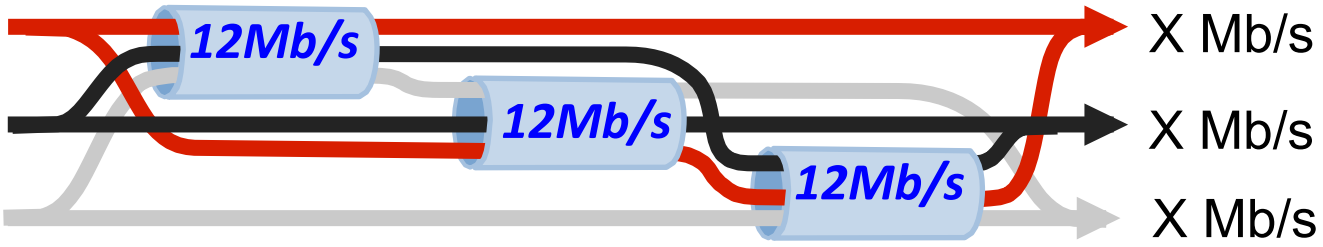


  - If each connection splits its traffic between its 1-hop and 2-hop subflows with ratio 2:1

$2X/3 + X/3 + X/3 = 12 \rightarrow X = 9\text{Mb/s}$

Design Goal 2: MPTCP Should Use Efficient Paths

- Assume there are 3 MPTCP connections with the same data rates of X Mb/s, each consisting of 2 subflows

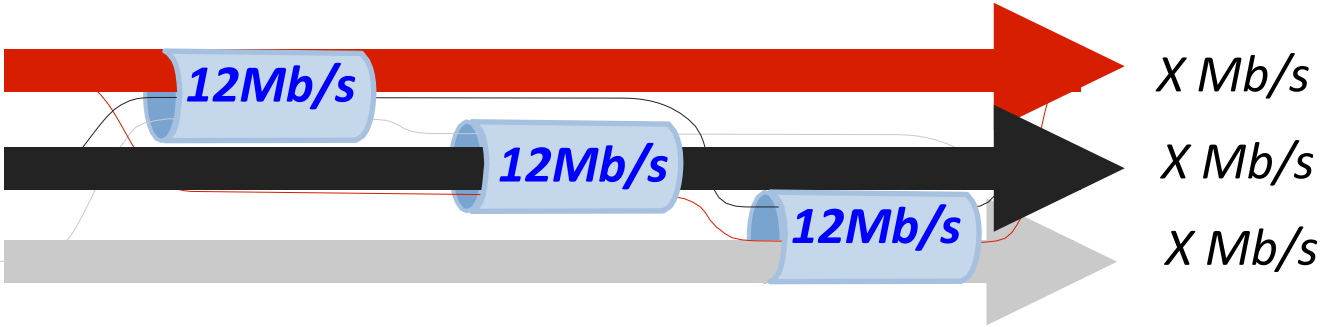


  - If each connection splits its traffic between its 1-hop and 2-hop subflows with ratio 4:1

    $4X/5 + X/5 + X/5 = 12 \rightarrow X = 10$ Mb/s

Design Goal 2: MPTCP Should Use Efficient Paths

- Assume there are 3 MPTCP connections with the same data rates of X Mb/s, each consisting of 2 subflows

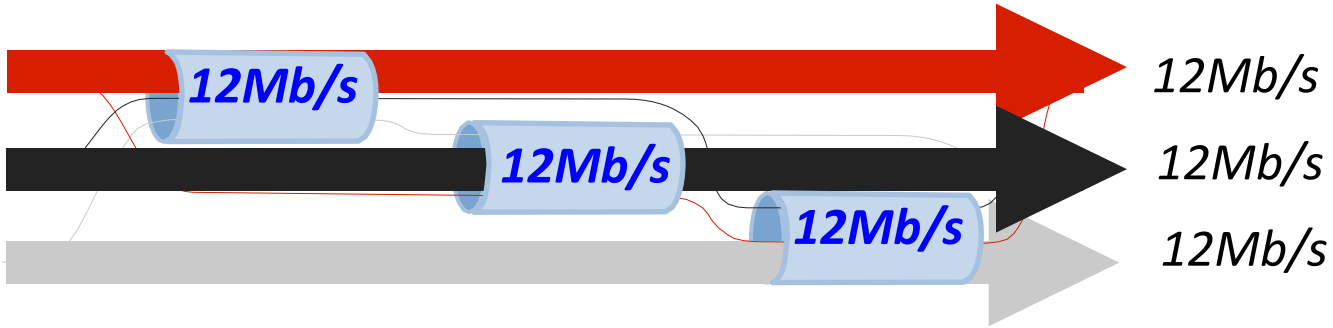


  - If each connection splits its traffic between its 1-hop and 2-hop subflows with ratio ∞ :1

All data is sent through the 1-hop subflow → X = 12 Mb/s

Design Goal 2: MPTCP Should Use Efficient Paths

- Assume there are 3 MPTCP connections, each consisting of 2 subflows

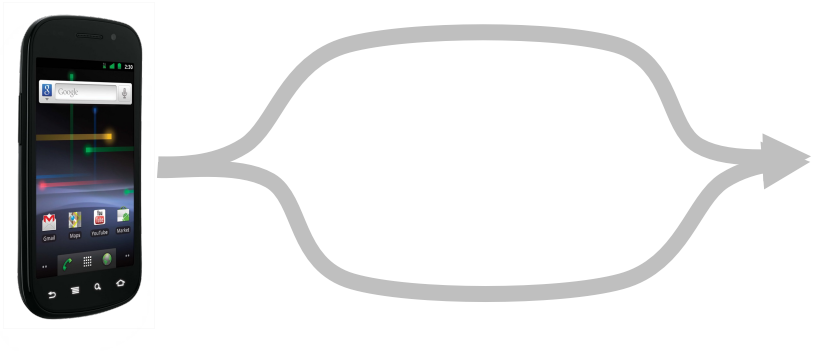


- Theoretical solution (Kelly + Voice 2005)
  - MPTCP should send all its traffic on its least-congested paths.
  - Theorem: This will lead to the most efficient allocation possible, given a network topology and a set of available paths.

F. Kelly and T. Voice: Stability of end-to-end algorithms for joint routing and rate control, ACM SIGCOMM Communication Review, 2005

Design Goal 3: Perform as well as TCP

RTT = Round-Trip Time



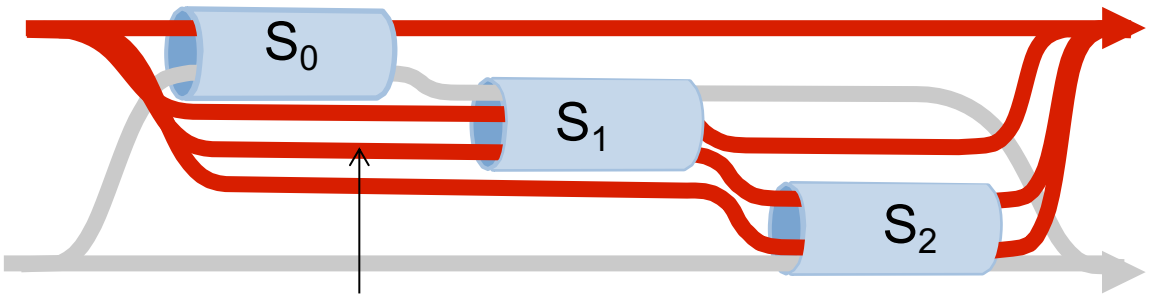
- Goal 3a.
  - A Multipath TCP user should get at least as much throughput as a single-path TCP would on the best of the available paths
- Goal 3b.
  - A Multipath TCP flow should take no more capacity on any link than a single-path TCP would
  - This is actually Design Goal 1

MPTCP Congestion Control Algorithm

- Needs to satisfy design goals 2 and 3
- The authors do not claim that:
  - The algorithm is optimal
  - It is an exhaustive survey of the design space
- The design space was signposted by examples, analyzed by calculations and experiments

MPTCP Congestion Control Algorithm

- A connection consists of set of subflows R
- Each subflow $r \in R$ maintains a congestion window w_r
- Denote RTT_r the round trip time measured by subflow r
- Denote S the subset of R s.t. subflows in S share a bottleneck with r
- Example:



Assume R consists of 4 subflows in red and this is r,
then the bottlenecks for r are S_1 and S_2 as shown in the figure.

MPTCP Congestion Control Algorithm

- For each ACK on path *r*, increase w_r by:

$$\min_{S \subseteq R : r \in S} \frac{max_{s \in S} w_s / RTT_s^2}{(\sum_{s \in S} w_s / RTT_s)^2}$$

- For each drop on path r, decrease w_r by w_r /2

Reminder: TCP sender maintains a congestion window, which governs the number of packets that the sender can send without waiting for an acknowledgment. The congestion window is updated dynamically, growing linearly when there is no congestion and halved when packet loss occurs. TCP congestion control ensures fairness: when multiple connections utilize the same congested link, each of them will independently converge to the same average value of the congestion window.

Reminder – Algorithm for regular TCP
- For each ACK, increase the congestion window w by 1/w, resulting in an increase of one packet per RTT.1
- For each loss, decrease w by w/2.

MPTCP Congestion Control Algorithm

- For each ACK on path *r*, increase w_r by:

Design goal 3: At any potential bottleneck S that path r might be in, look at the best that a single-path TCP could get, and compare to what I'm getting.

$$\min_{S \subseteq R : r \in S} \frac{max_{s \in S} w_s / RTT_s^2}{(\sum_{s \in S} w_s / RTT_s)^2}$$

- For each drop on path r, decrease w_r by w_r /2

Intuition:
Numerator -> the best that regular TCP could get
Denominator -> what MPTCP gets
Min -> least congested or used part

MPTCP Congestion Control Algorithm

- For each ACK on path *r*, increase w_r by

Design goal 2: We want to shift traffic away from congestion.
To achieve this, we increase windows in proportion to their size.

$$\min_{S \subseteq R : r \in S} \frac{max_{s \in S} w_s / RTT_s^2}{(\sum_{s \in S} w_s / RTT_s)^2}$$

- For each drop on path r, decrease w_r by w_r /2

References

1. RFC 8684 (March 2020): https://www.rfc-editor.org/rfc/rfc8684.txt
2. Q. Peng, A. Walid, J. Hwang and S. H. Low, "Multipath TCP: Analysis, Design, and Implementation," in IEEE/ACM Transactions on Networking, vol. 24, no. 1, pp. 596-609, Feb. 2016.
3. O. Bonaventure, M. Handley, and C. Raiciu, An Overview of Multipath TCP, ;login:, Oct. 2012: http://inl.info.ucl.ac.be/system/files/bonaventure_0.pdf