X_405082
Advanced Computer Networks

Network Transport

Lin Wang (lin.wang@vu.nl)

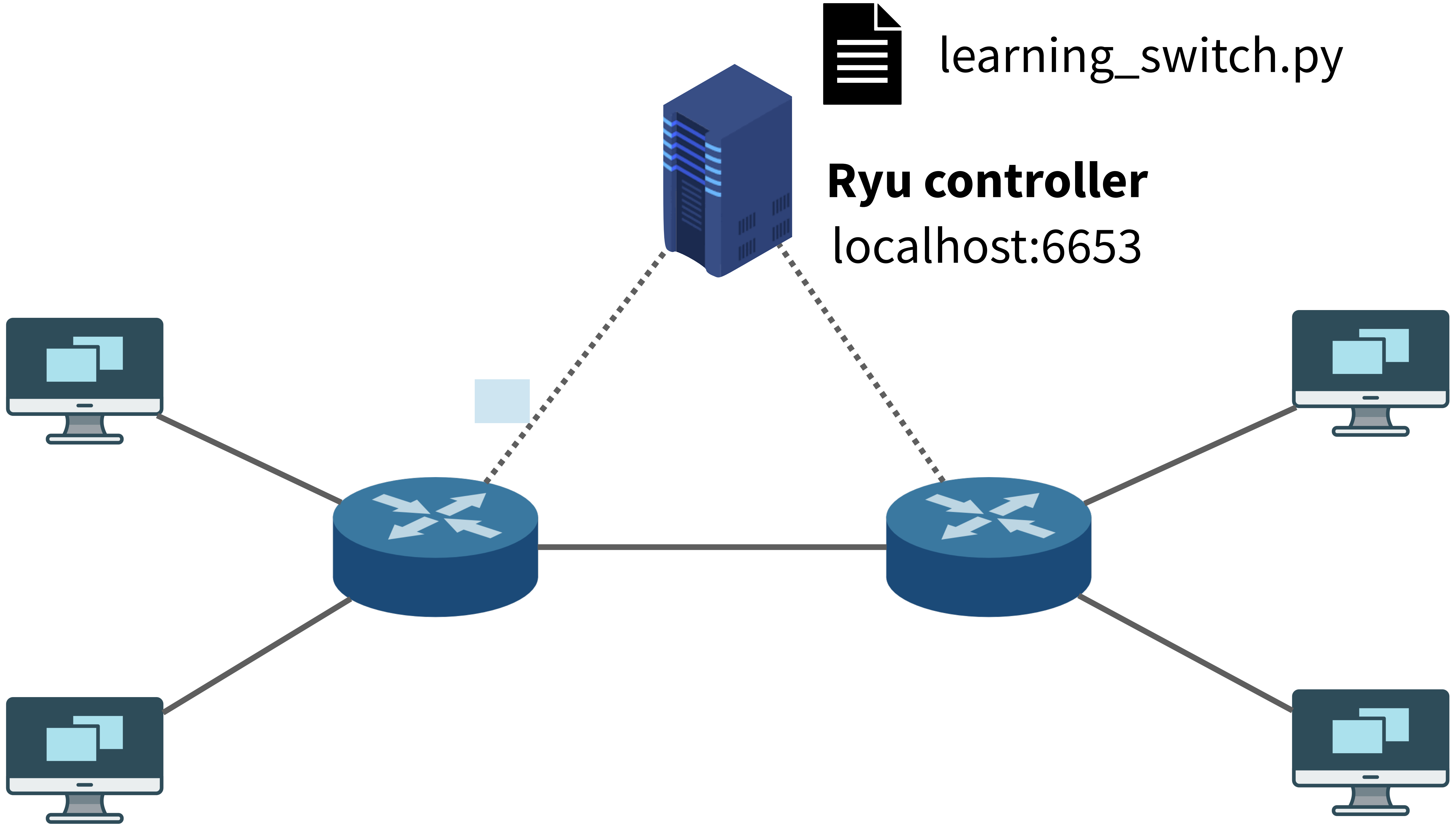
Period 2, Fall 2020

VRIJE UNIVERSITEIT AMSTERDAM

Part of the content is adapted from Nick Mckeown and Phillipa Gill

Ryu controller for lab1

Some questions to think about: (1) which switch did we get the packet? (2) do I know how to deal with this packet and future similar packets already? (3) if so, how to tell the switch not to send in the packet again?



Course outline

Warm-up

- Fundamentals
- Forwarding and routing
- **Network transport** 👈

Data centers

- Data center networking
- Data center transport

Programmability

- Software defined networking
- Programmable forwarding

Video

- Video streaming
- Video stream analytics

Networking and ML

- Networking for ML
- ML for networking

Mobile computing

- Wireless and mobile

Learning objectives

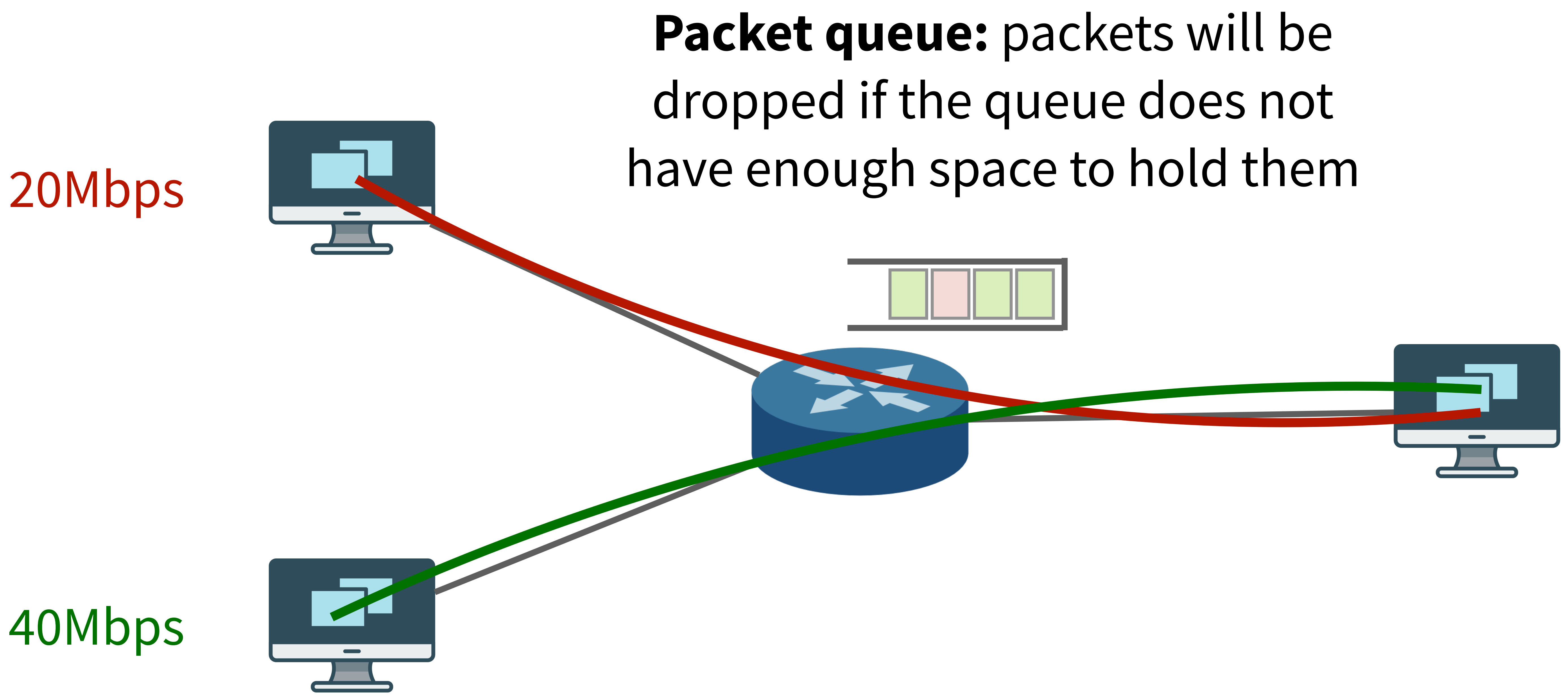
What are the new advancements in network transport design?

TCP congestion control

Multi-path TCP

QUIC & HTTP3.0

What is congestion control



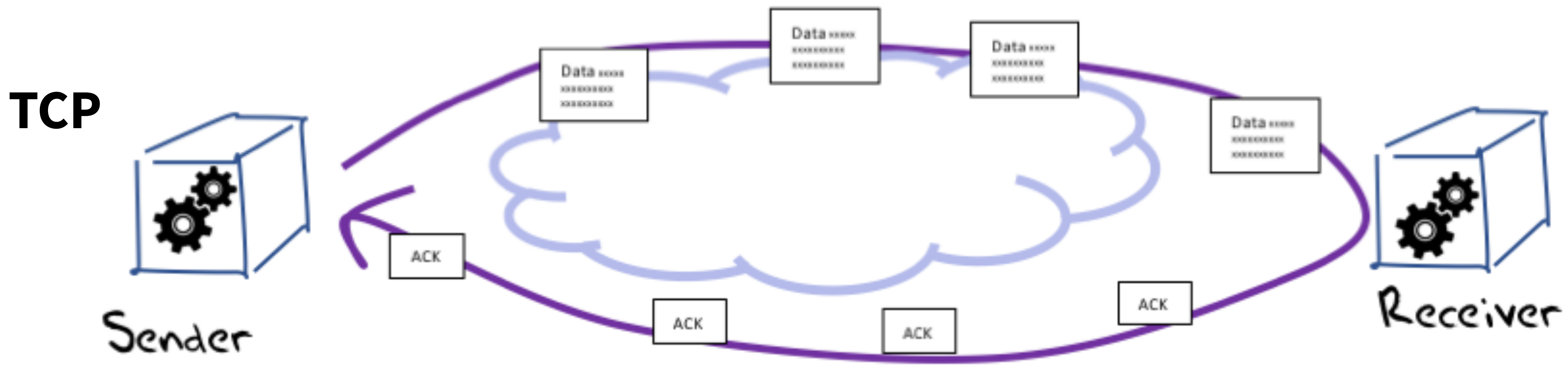
Congestion control aims to determine the **rate to send data** on a connection, such that (1) the sender does not overrun the network capability and (2) the network is efficiently utilized (in a **distributed** manner without a centralized coordinator)

Transport Control Protocol (TCP)

Send data across the network **as fast as possible**, but **not faster**

End-to-end design

- The end hosts control the sending rate, based on ACKs
- Does not rely on functionalities provided by the network
- May leverage in-network mechanisms like ECN for performance improvements (more in next week)



How did TCP work before 1988

```
1.5.  Operation

  As noted above, the primary purpose of the TCP is to provide reliable,
  securable logical circuit or connection service between pairs of
  processes.  To provide this service on top of a less reliable internet
  communication system requires facilities in the following areas:

    Basic Data Transfer
    Reliability
    Flow Control
    Multiplexing
    Connections
    Precedence and Security
```

RFC 793

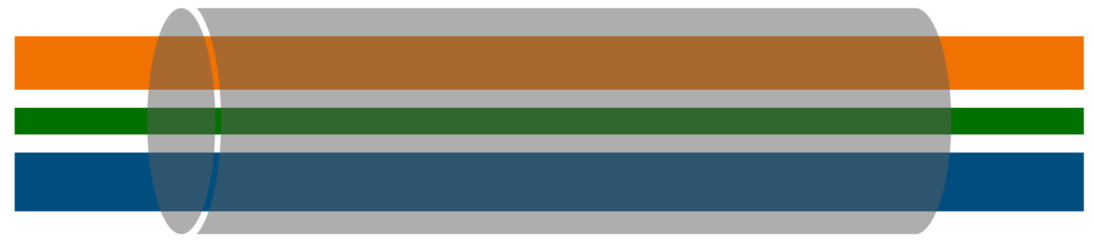
Congestion collapse (breakdowns in performance) noted on NSFNet in 1986, 40Kbps links operating at as low as 32bps

What do we really mean by congestion

Water pipe analogue

- The path of a TCP connection as a water pipe that has space to hold water
- The available space of the water pipe changes depending on the other connections
- If we pump too fast, the water pipe will overflow; too slow, not efficient
- TCP tries to keep the water pipe just full

We do not need congestion control if there is no congestion which is also created by us!

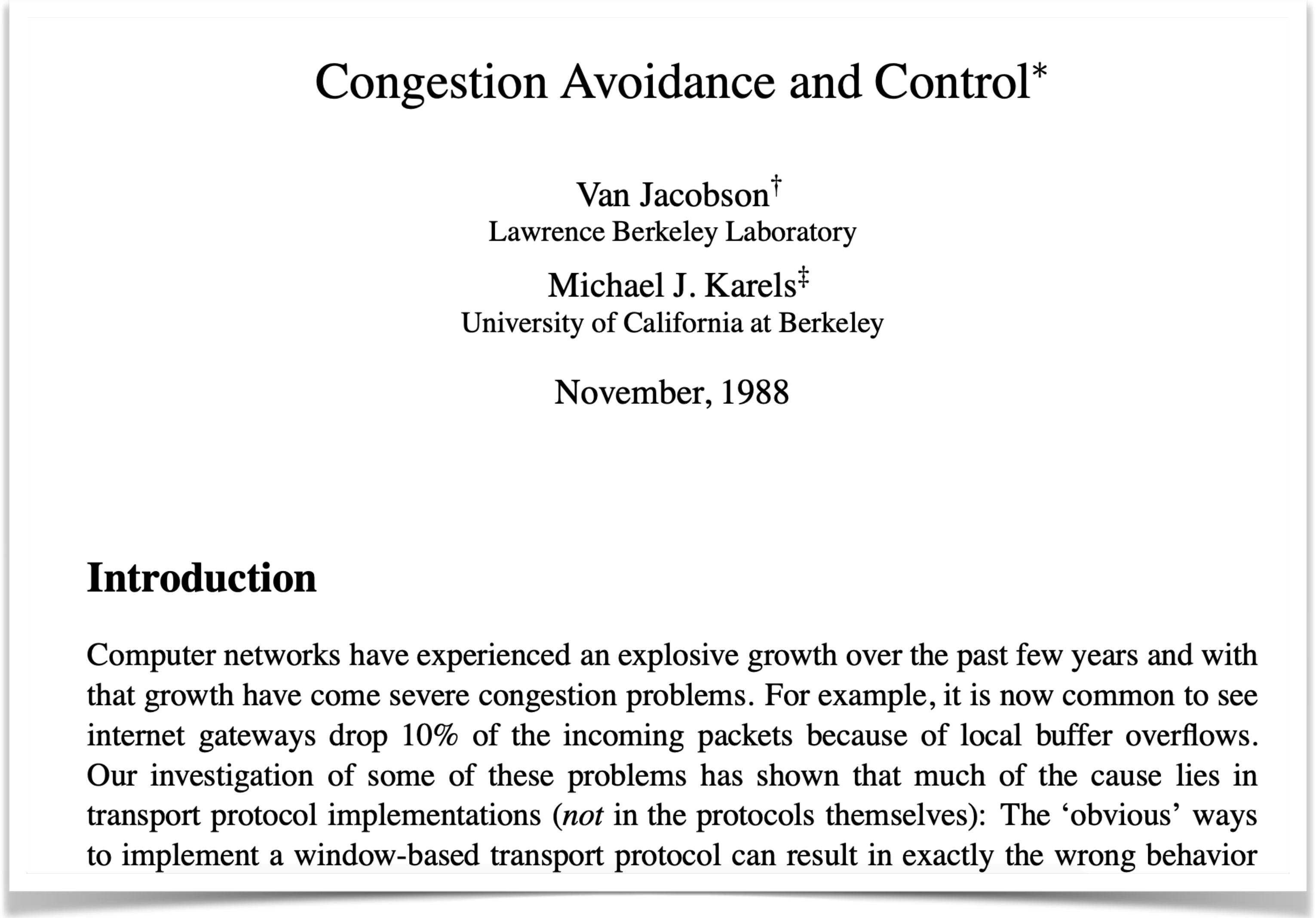


Less about congestion control, more about **resource allocation**

- How to wisely allocate space for all connections

“**Packet conservation**” is key to this idea

Packet conservation


Congestion Avoidance and Control*

Van Jacobson†
Lawrence Berkeley Laboratory

Michael J. Karels‡
University of California at Berkeley

November, 1988

Introduction

Computer networks have experienced an explosive growth over the past few years and with that growth have come severe congestion problems. For example, it is now common to see internet gateways drop 10% of the incoming packets because of local buffer overflows. Our investigation of some of these problems has shown that much of the cause lies in transport protocol implementations (*not* in the protocols themselves): The 'obvious' ways to implement a window-based transport protocol can result in exactly the wrong behavior

ACM SIGCOMM 1988

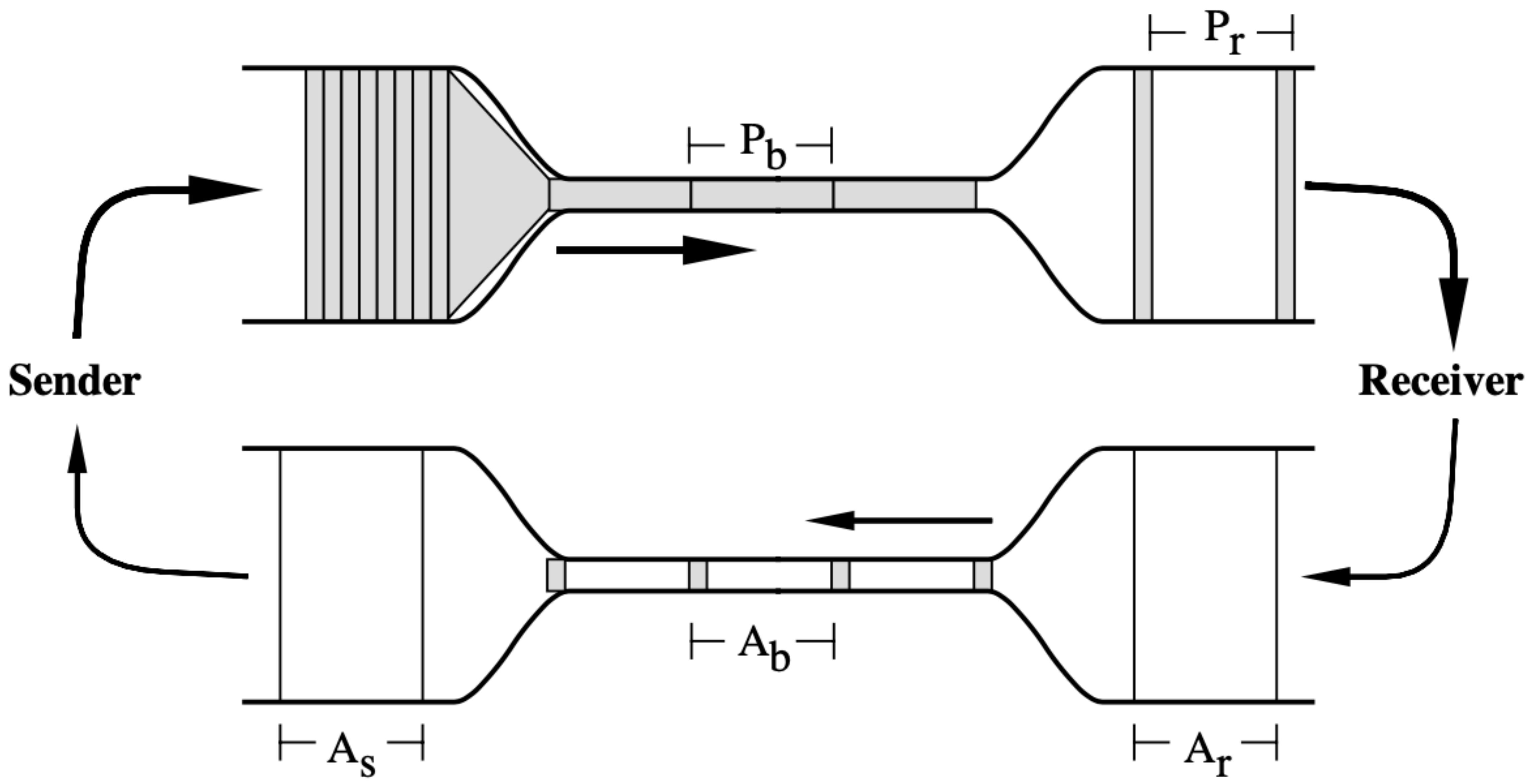
Packet Conservation Principle

For a connection in equilibrium, i.e., running stably with a full window of data in transit, a new packet should not be put into the network until an old packet leaves.

How does TCP achieves this?

Self-clocking

Assume the following connection is in an equilibrium



"So, if packets after the first burst are sent only in response to an ACK, the sender's packet spacing will exactly match the packet time on the slowest link in the path."

$$A_s = A_b = A_r = P_b = P_r$$

Key observation: new packets are naturally sent (1) **when a packet leaves** (2) **at the bottleneck rate**

Two more challenges to solve

How to reach equilibrium in the first place?

TCP idea: slow-start

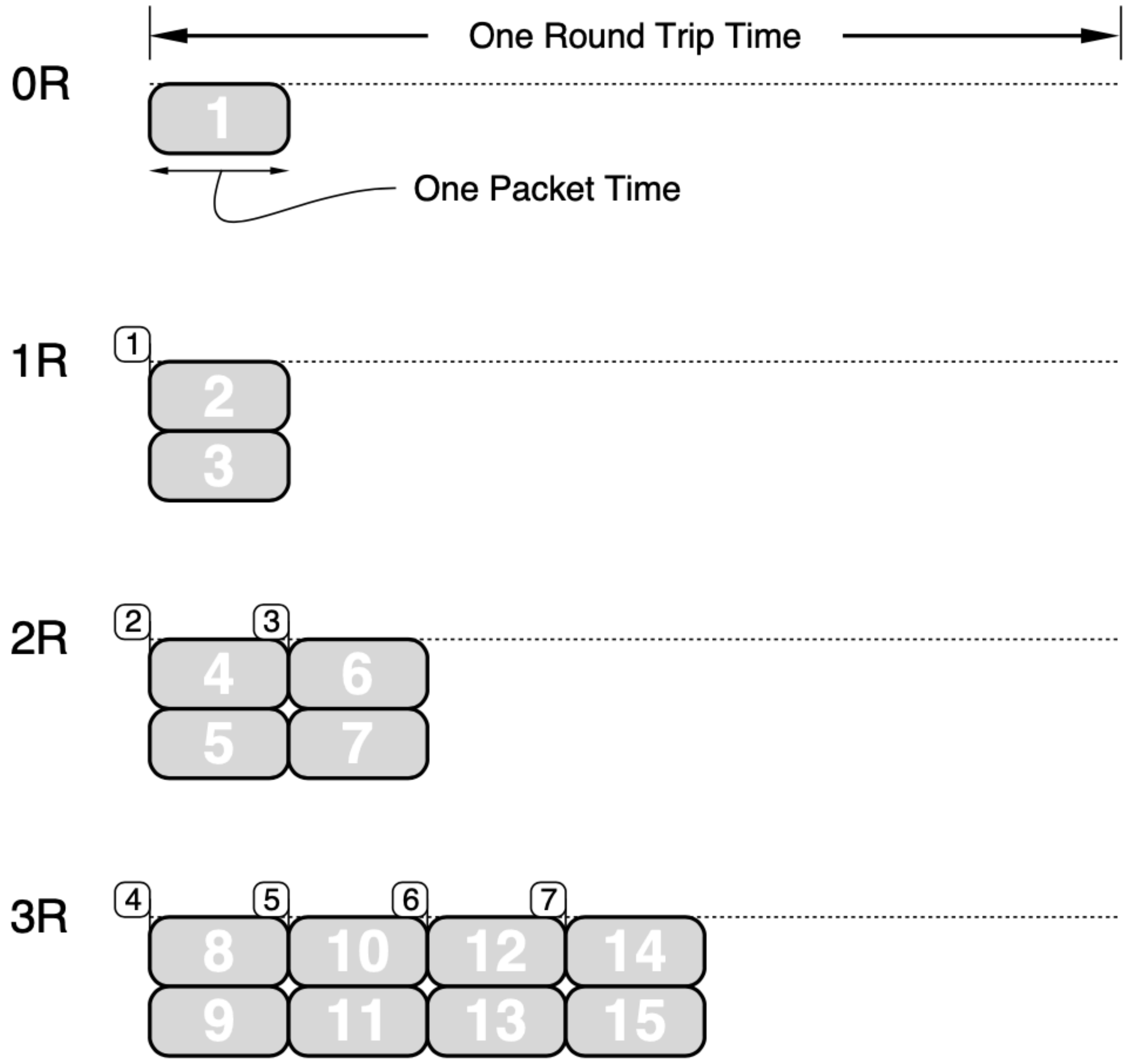
How to adapt to the available space in the path accordingly?

TCP idea: additive increase multiplicative decrease (AIMD)

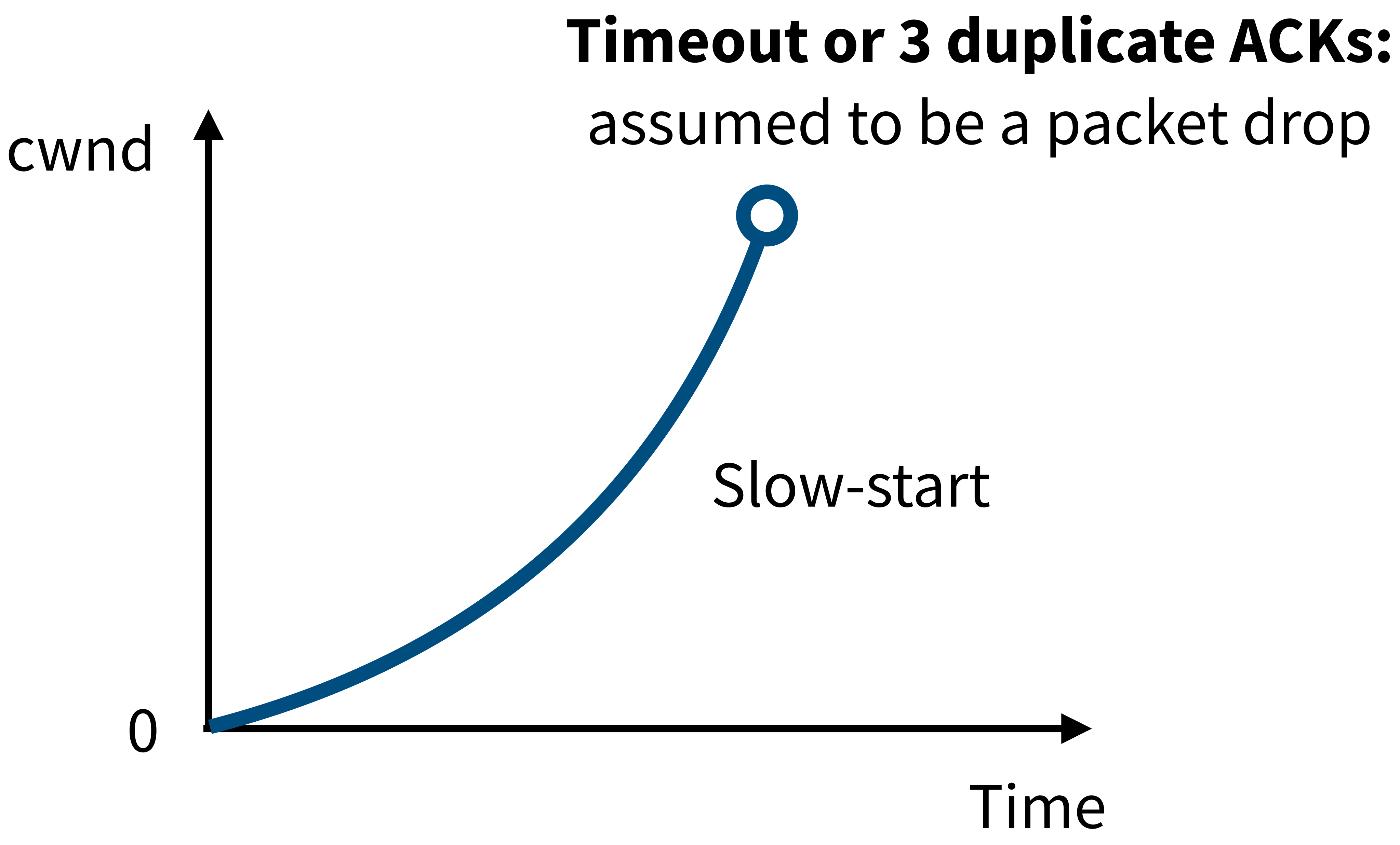
These two ideas are overloaded on the TCP sliding window mechanism for congestion avoidance

TCP slow-start

Upon receiving an ACK, increase the congestion window (cwnd) by 1



Window size = min(cwnd, rwnd)



Recall the data transfer time problem

Assume a network link with 10Gbps bandwidth and 1ms round-trip time, how long does it take to transfer 100KB of data on the link?

- Slow start before reaching the maximum bandwidth
- 100KB = 1460B * (0 + 1 + 2 + 4 + 8 + 16 + 32 + 6)
- In total 7 round-trips in the slow-start phase → 7ms

Do you know how we come up with 1460B here? Refresh your knowledge about Ethernet MTU.

More generally, how long does it take to reach a window size of W in slow-start?

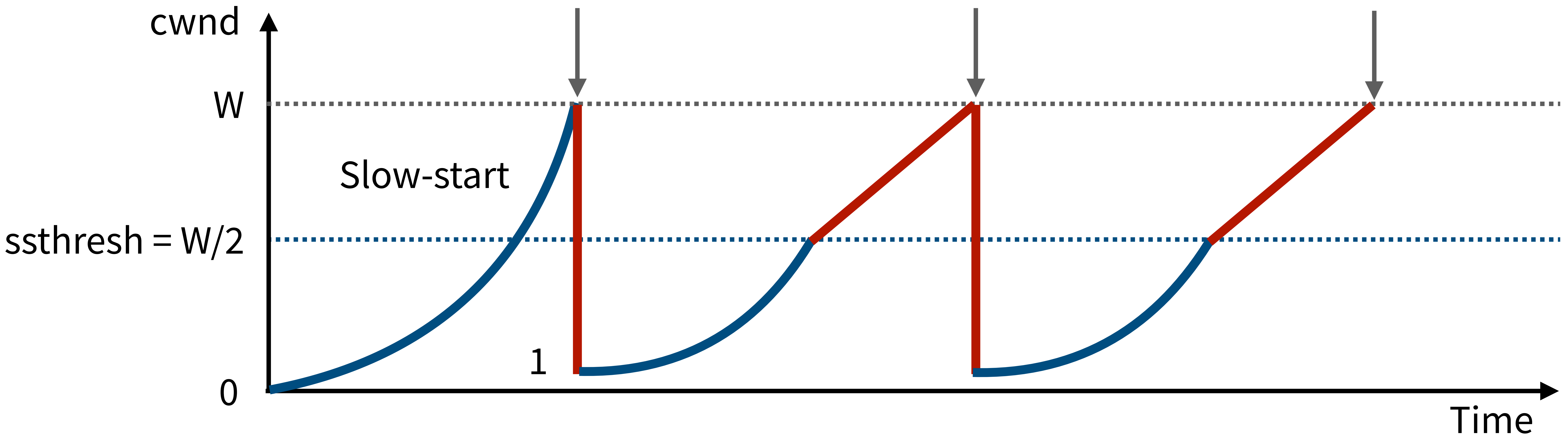
- RTT * $\log_2 W$
- Exponential growth → slow-start is not slow at at!

TCP AIMD

What is a duplicate ACK?

TCP Tahoe

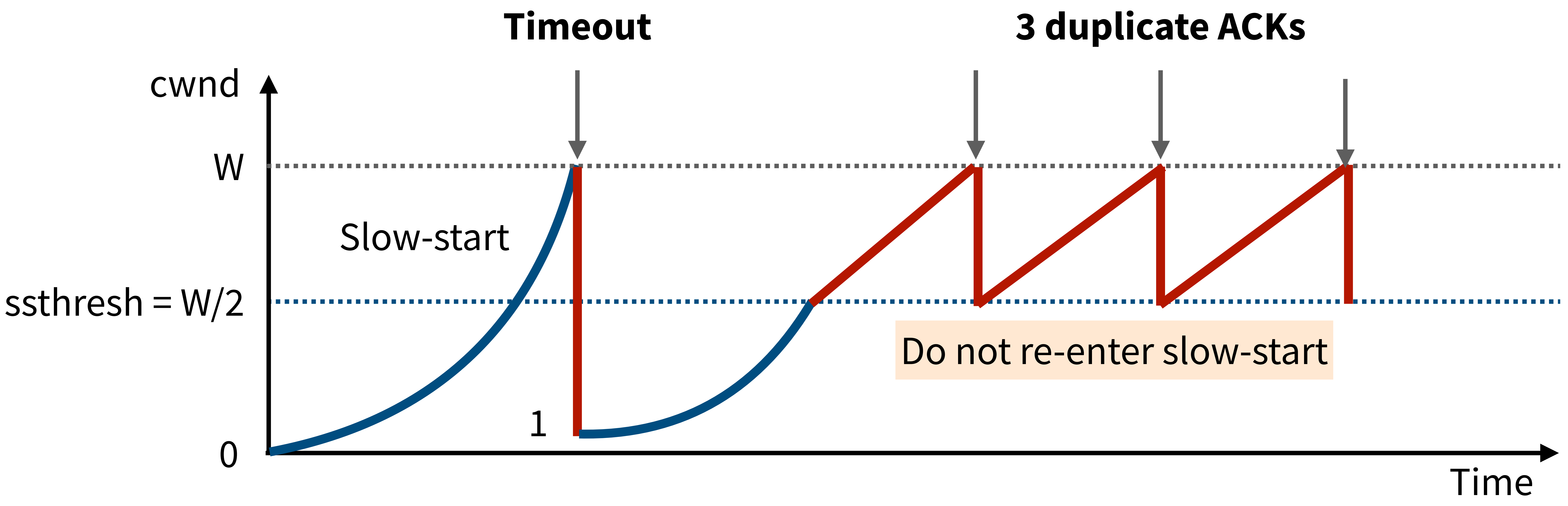
Timeout or 3 duplicate ACKs:
assumed to be a packet drop




```
if cwnd < ssthresh:      //slow-start
  cwnd += 1
else:                    // AIMD
  cwnd += 1 / cwnd
```

TCP AIMD

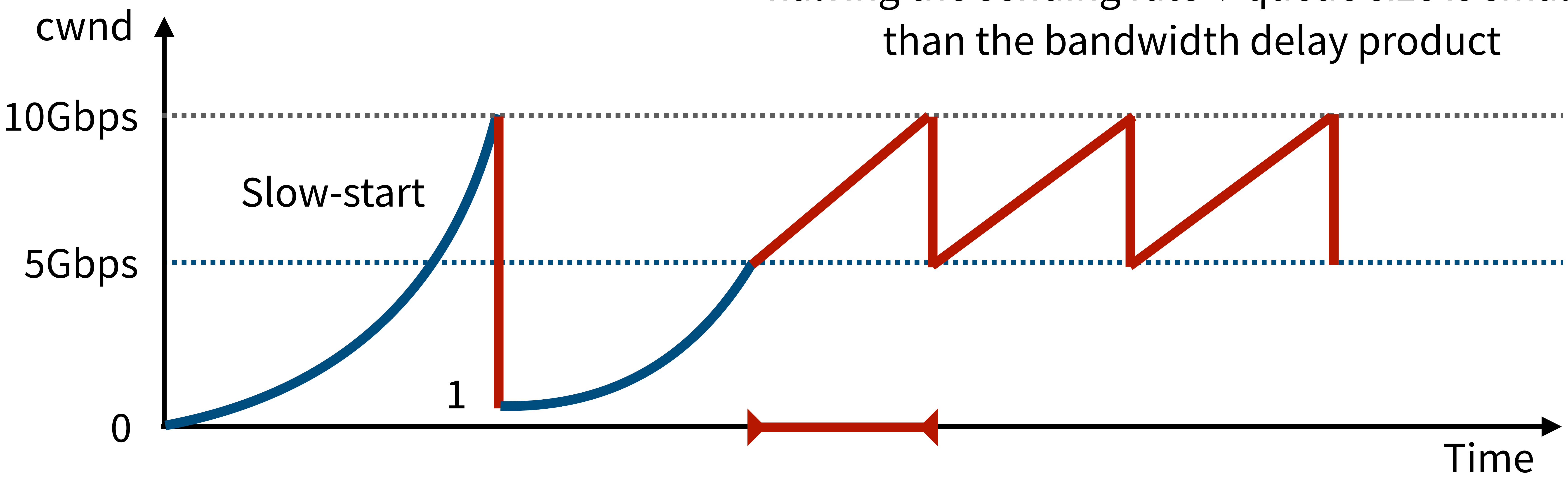
TCP Reno




```
if cwnd < ssthresh:     //slow-start
  cwnd += 1
else:                   // AIMD
  cwnd += 1 / cwnd
```

TCP Reno problems

Problem 1: Assume queue will drain when halving the sending rate → queue size is smaller than the bandwidth delay product

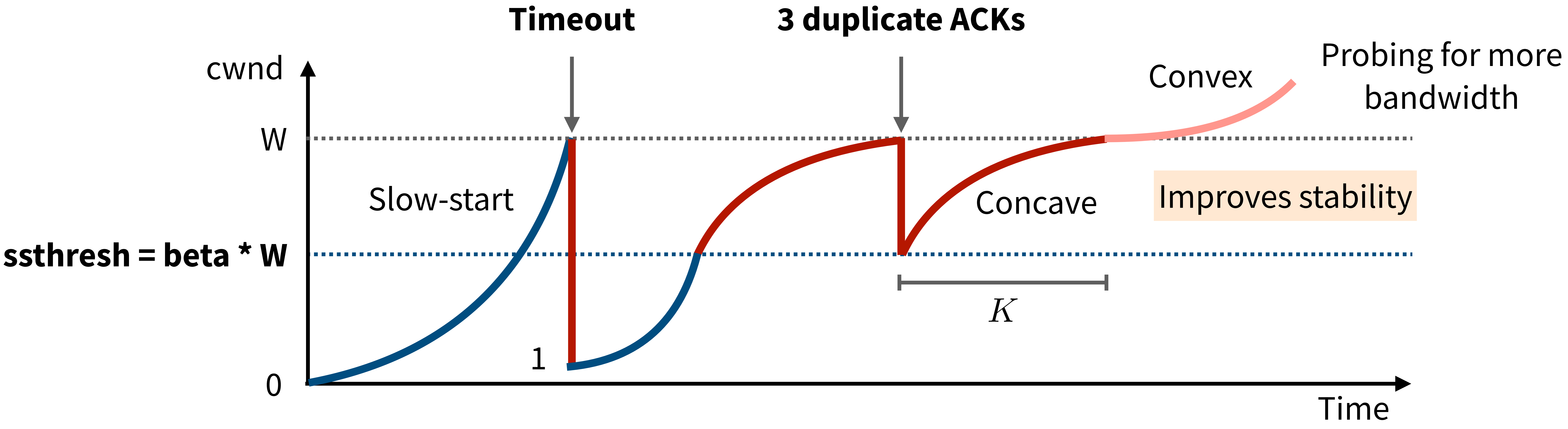


Problem 2: How long does it take to ramp up? The cwnd increases by 1460B every RTT → 428K RTT needed. For a connection with RTT of 30ms, this is around 3.6 hours!

Other TCP variants

In Linux kernel since v2.6.19

TCP CUBIC



Use a cubic function to adjust window size based on time, not RTT:

$$W(t) = C(t - K)^3 + W_{max}$$

https://www.cs.princeton.edu/courses/archive/fall16/cos561/papers/Cubic08.pdf

Other TCP variants

Variant	Feedback	Required changes	Benefits	Fairness
(New) Reno	Loss	—	—	Delay
Vegas	Delay	Sender	Less loss	Proportional
High Speed	Loss	Sender	High bandwidth	
BIC	Loss	Sender	High bandwidth	
CUBIC	Loss	Sender	High bandwidth	
C2TCP[9][10]	Loss/Delay	Sender	Ultra-low latency and high bandwidth	
NATCP[11]	Multi-bit signal	Sender	Near Optimal Performance	
Elastic-TCP	Loss/Delay	Sender	High bandwidth/short & long-distance	
Agile-TCP	Loss	Sender	High bandwidth/short-distance	
H-TCP	Loss	Sender	High bandwidth	
FAST	Delay	Sender	High bandwidth	Proportional
Compound TCP	Loss/Delay	Sender	High bandwidth	Proportional
Westwood	Loss/Delay	Sender	L	
Jersey	Loss/Delay	Sender	L	
BBR[12]	Delay	Sender	BLVC, Bufferbloat	
CLAMP	Multi-bit signal	Receiver, Router	V	Max-min
TFRC	Loss	Sender, Receiver	No Retransmission	Minimum delay
XCP	Multi-bit signal	Sender, Receiver, Router	BLFC	Max-min
VCP	2-bit signal	Sender, Receiver, Router	BLF	Proportional
MaxNet	Multi-bit signal	Sender, Receiver, Router	BLFSC	Max-min
JetMax	Multi-bit signal	Sender, Receiver, Router	High bandwidth	Max-min
RED	Loss	Router	Reduced delay	
ECN	Single-bit signal	Sender, Receiver, Router	Reduced loss	

https://en.wikipedia.org/wiki/TCP_congestion_control#TCP_Tahoe_and_Reno

Check your Linux TCP variant

```
cat /proc/sys/net/ipv4/tcp_congestion_control
```

```
wang@woody:~$ cat /proc/sys/net/ipv4/tcp_congestion_control
cubic
wang@woody:~$ uname -a
Linux woody 4.15.0-96-generic #97-Ubuntu SMP Wed Apr 1 03:25:46 UTC 2020 x86_64
x86_64 x86_64 GNU/Linux
wang@woody:~$
```

Why is TCP congestion control so complex?

Packet loss is not a good indicator of congestion — it is usually too late. How can we do better?

Instead of shooting for packet losses, try to find the best operating point more explicitly

- Provide a model for the network
- Estimate the parameters for the model based on probing
- Decide sending rate using the network model

BBR

- Bottleneck Bandwidth and Round-trip propagation time (BBR), developed at Google in 2016
- Congestion-based congestion control


BBR Congestion-Based Congestion Control

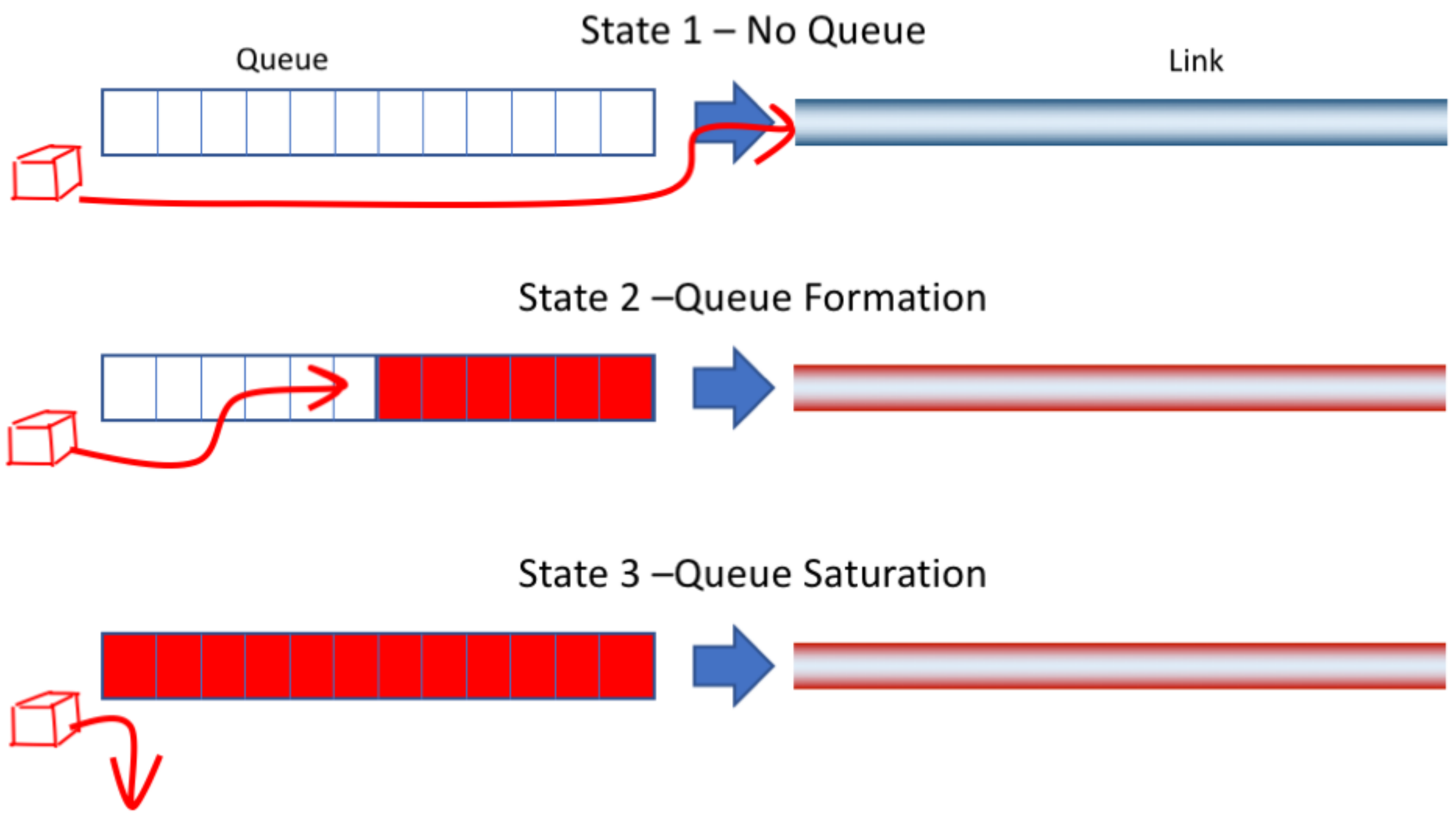
MEASURING BOTTLENECK BANDWIDTH AND ROUND-TRIP PROPAGATION TIME

NEAL CARDWELL
YUCHUNG CHENG
C. STEPHEN GUNN
SOHEIL HASSAS YEGANEH
VAN JACOBSON

By all accounts, today's Internet is not moving data as well as it should. Most of the world's cellular users experience delays of seconds to minutes; public Wi-Fi in airports and conference venues is often worse. Physics and climate researchers need to exchange petabytes of data with global collaborators but find their carefully engineered multi-Gbps infrastructure often delivers at only a few Mbps over intercontinental distances.[6]

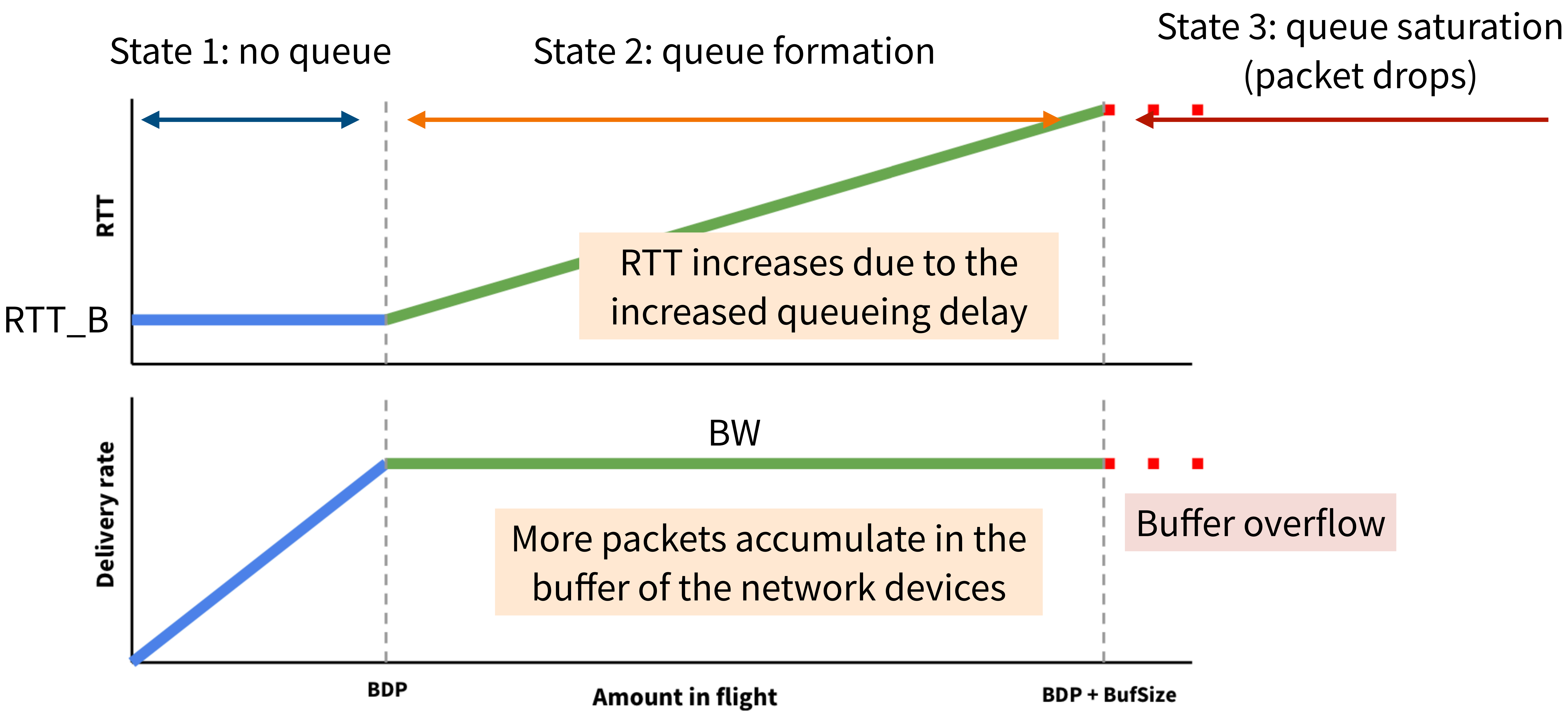
ACM Queue 2016

A simple model of links and queues



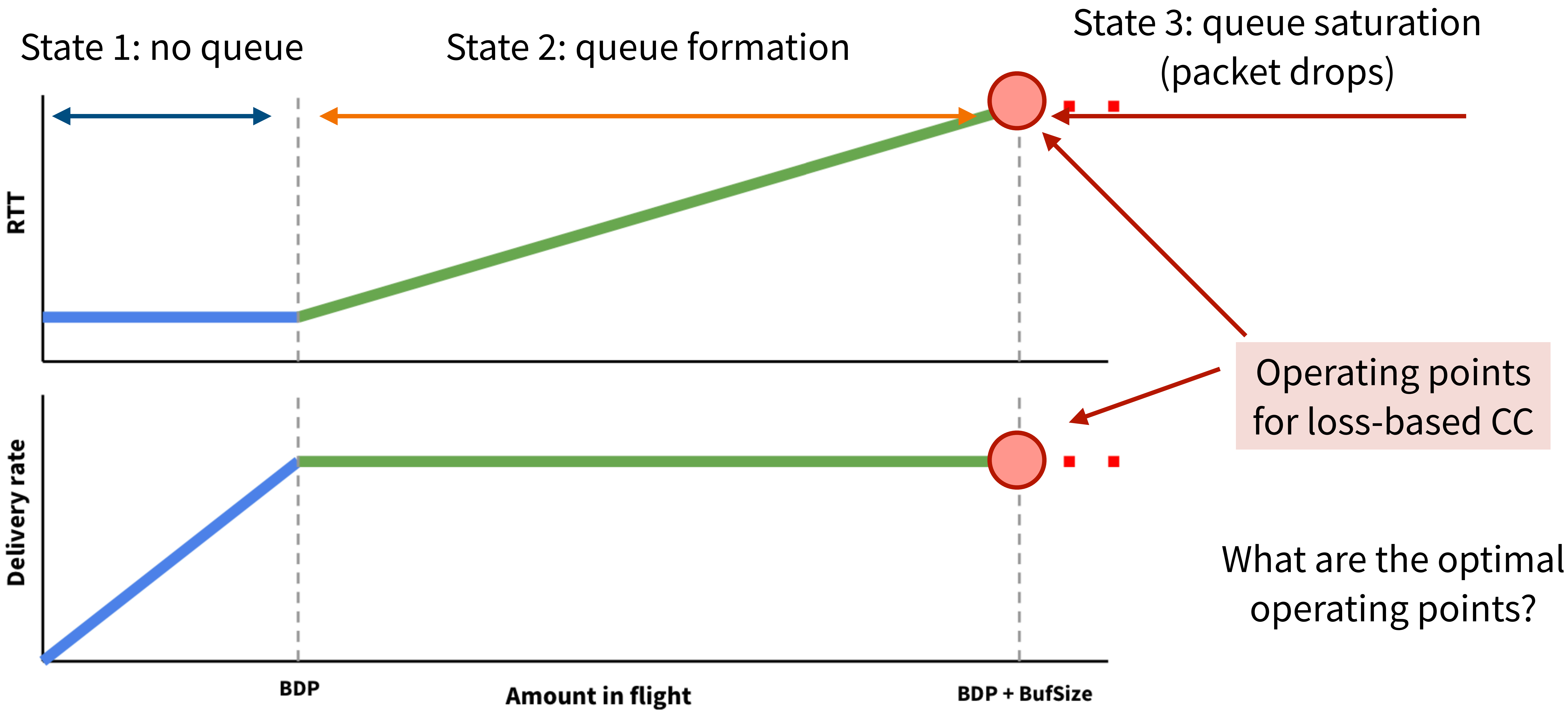
What are the behaviors of the delivery rate and RTT under the above states?

Modeling sending rate and RTT



Bandwidth delay product (BDP)

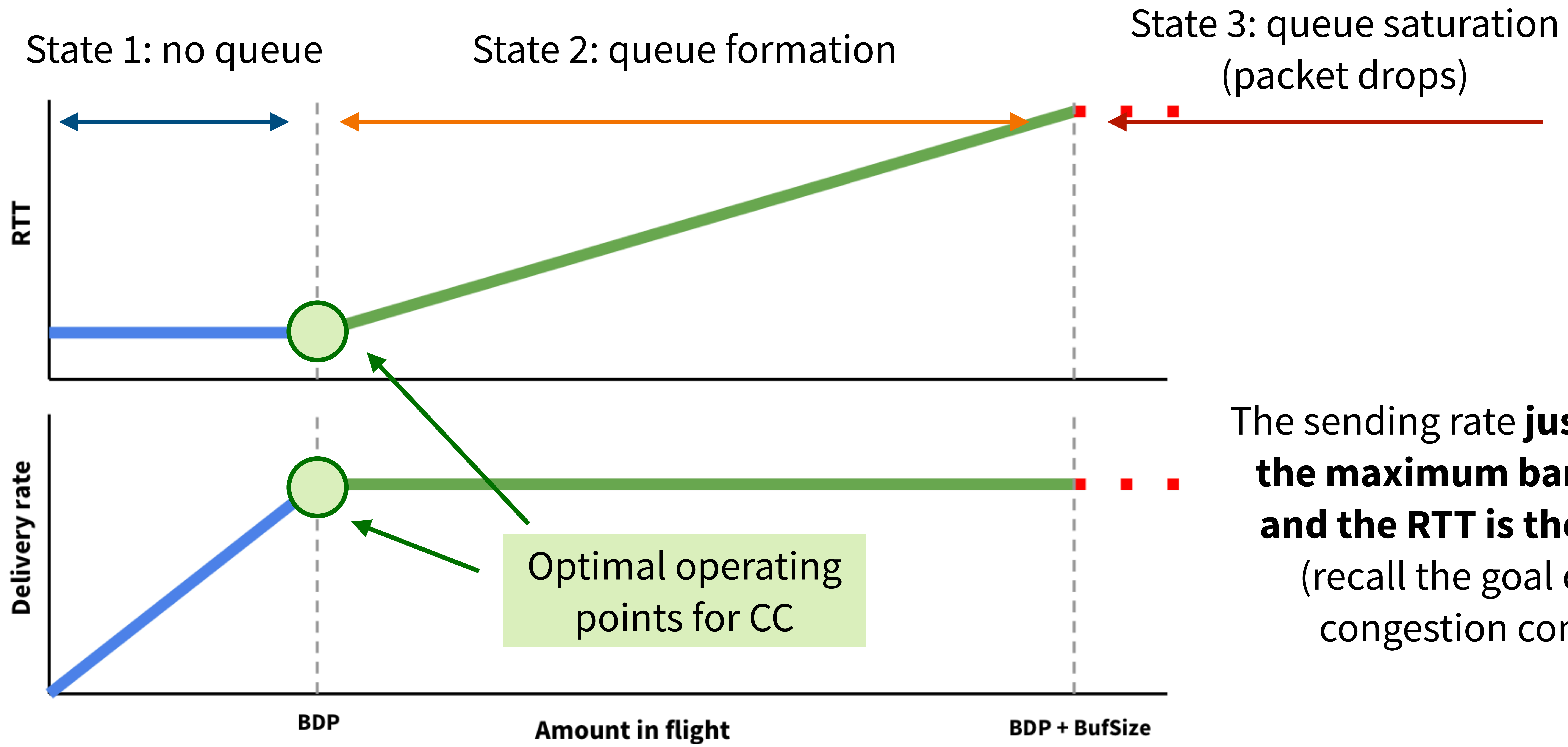
Operating point of loss-based CC algorithms



What are the optimal operating points?

Bandwidth delay product (BDP)

Optimal operating point

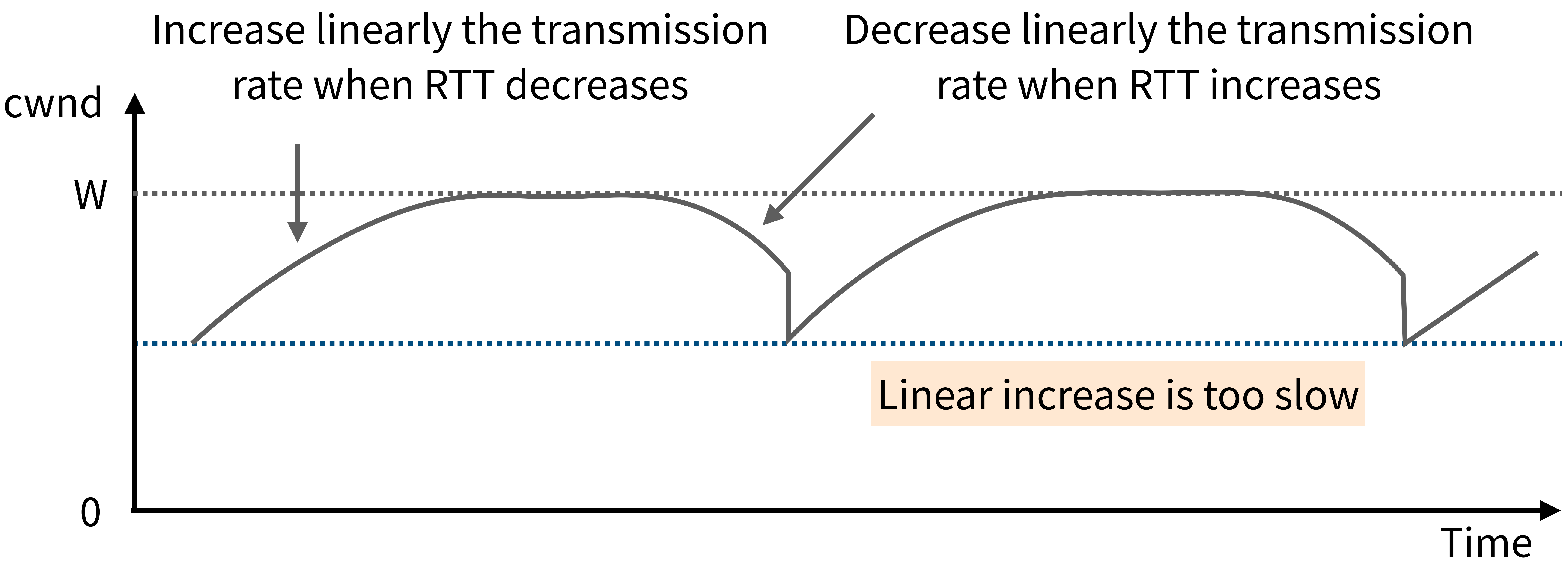


The sending rate **just reaches the maximum bandwidth and the RTT is the lowest** (recall the goal of TCP congestion control)

Bandwidth delay product (BDP)

TCP Vegas

Adjust sending rate based on measured RTT — trying to operate at the optimal point



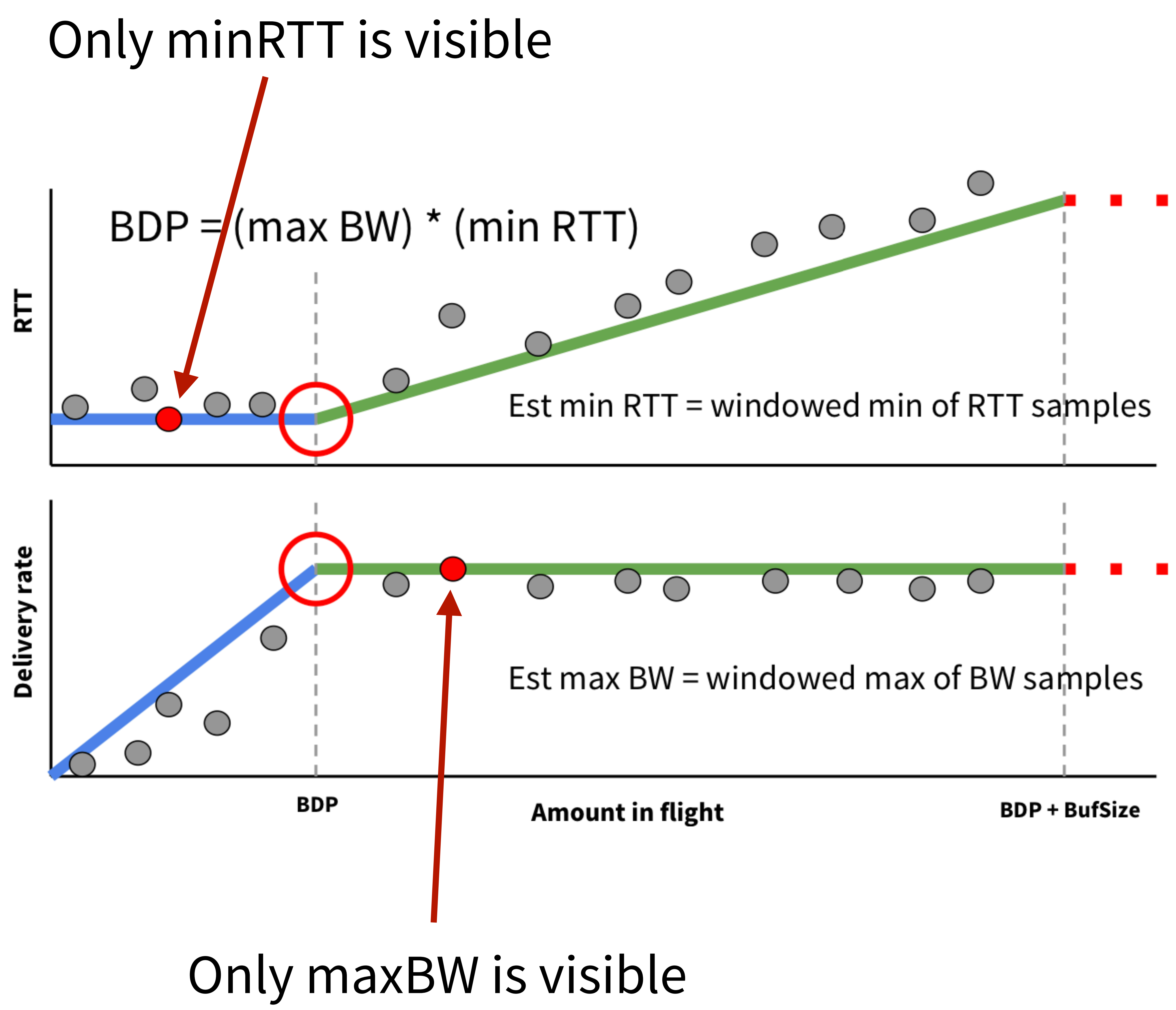
Not competitive to flows with loss-based congestion control algorithms, why?

BBR

Model the network: estimates **maxBW** and **minRTT** on every ACK (that is not marked as application limited)

Control sending rate based on the model:

- Probe both maxBW and minRTT, to feed the model samples
- Pace near estimated BW, to reduce queues and losses
- Vary pace rate to keep in-flight near BDP (full pipe but small queue)



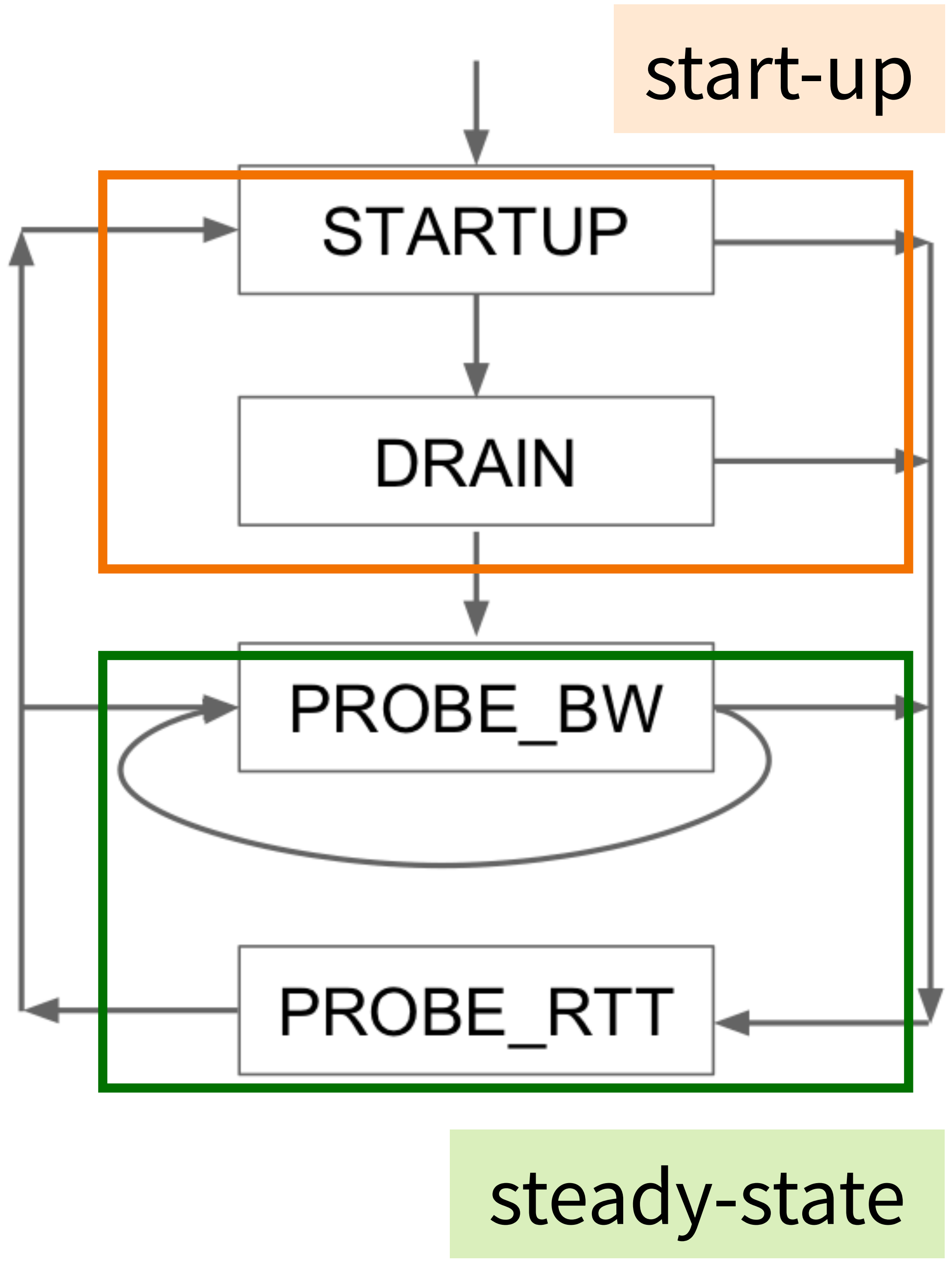
BBR state machine

STARTUP: exponential growth to quickly fill pipe (like slow-start)

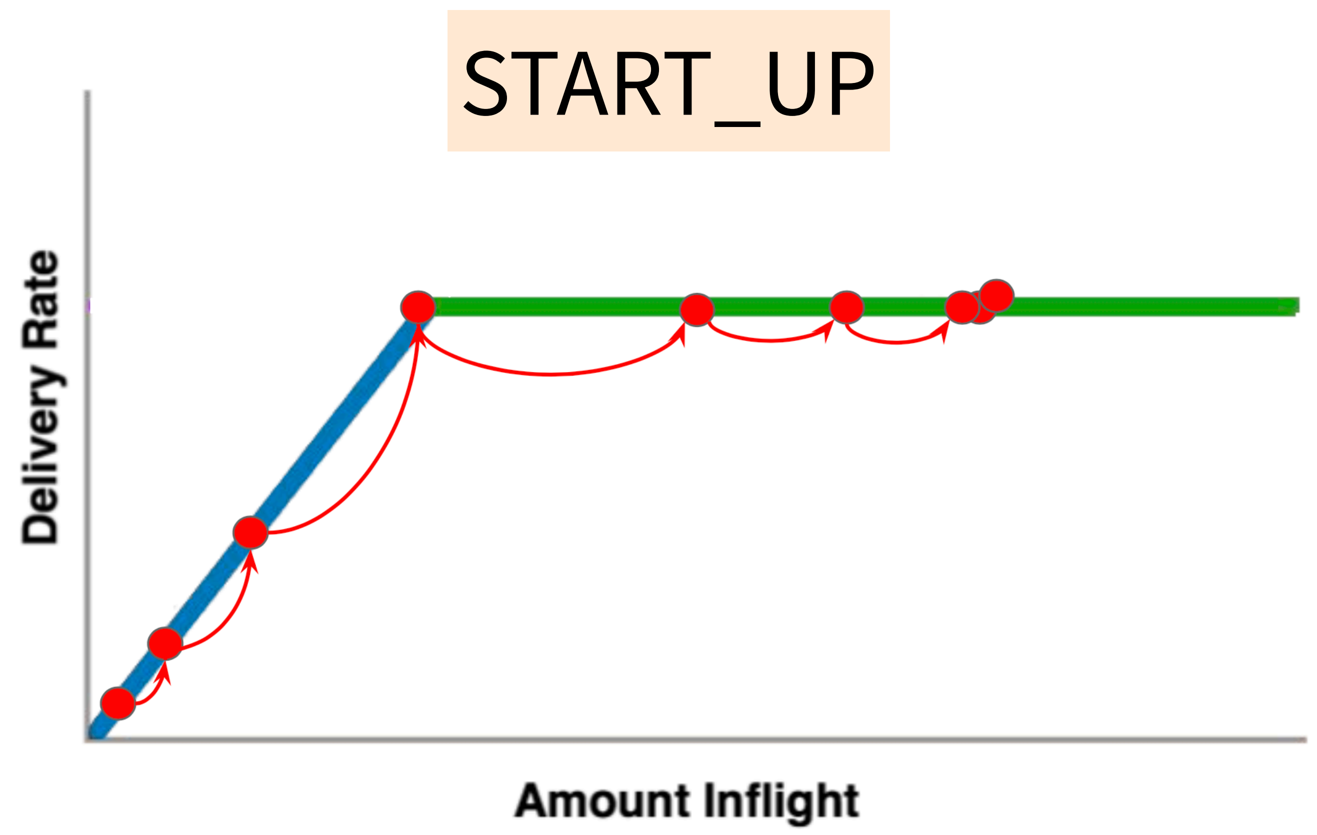
DRAIN: drain the queue created in STARTUP

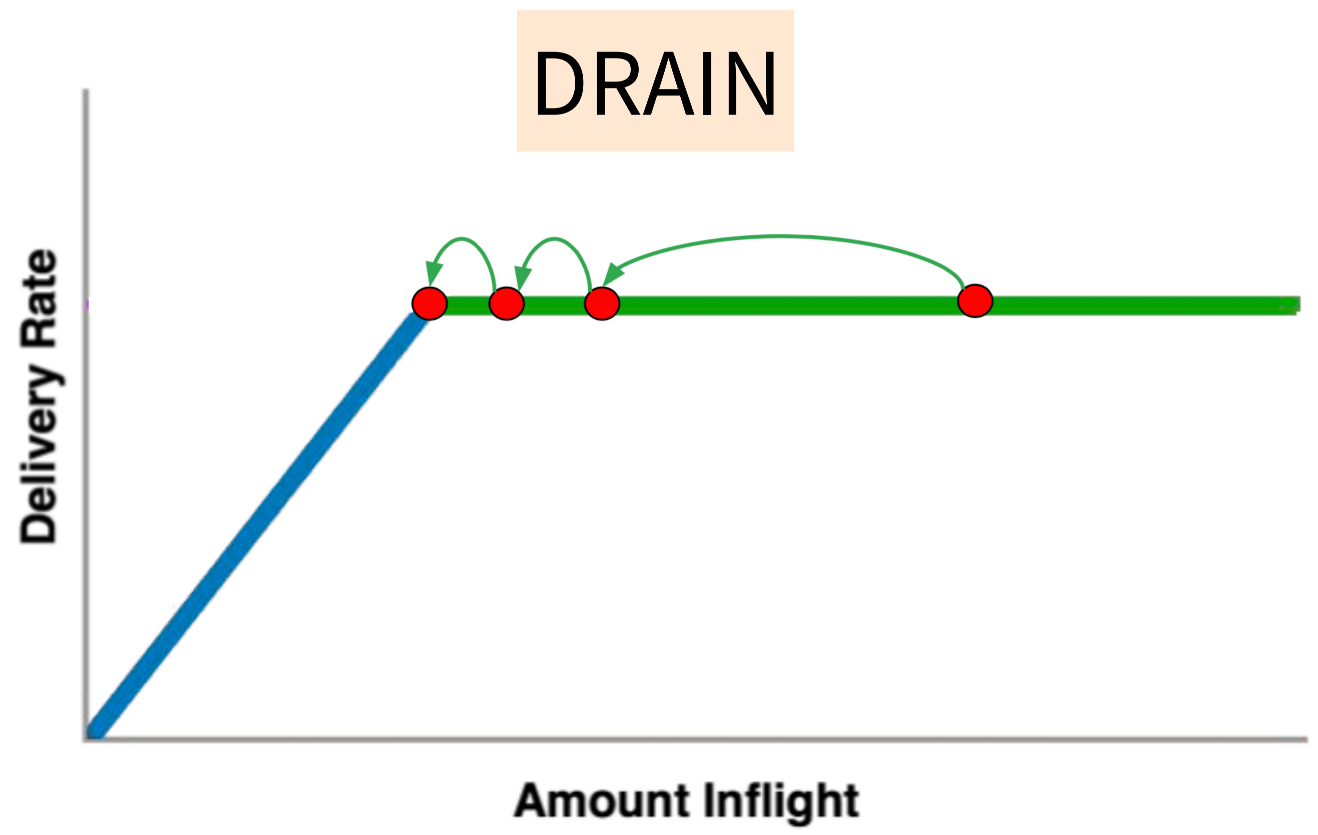
PROBE_BW: cycle pacing_gain to explore the fair share bandwidth → avoiding the flow to be kicked out by flows that use loss-based congestion control algorithms

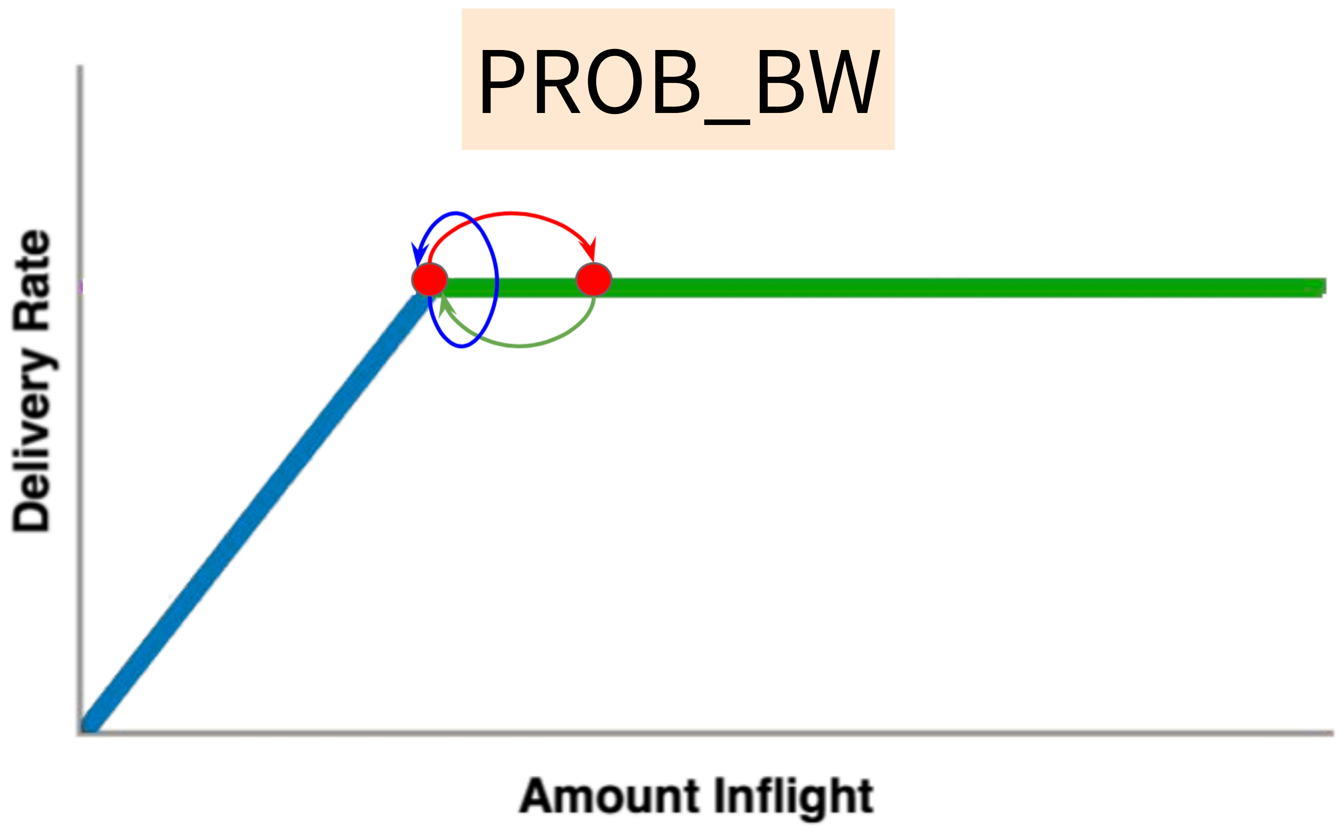
PROBE_RTT: if needed, occasionally send slower to probe minRTT

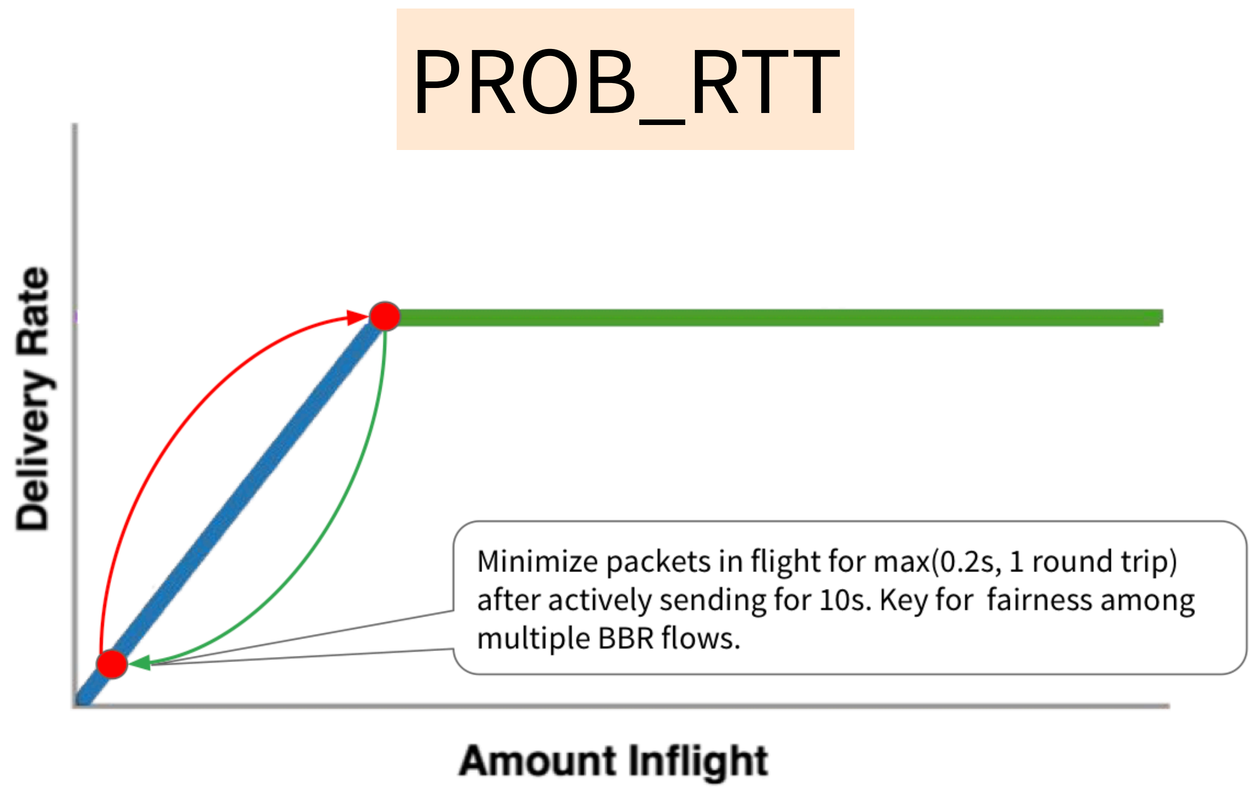


BBR states

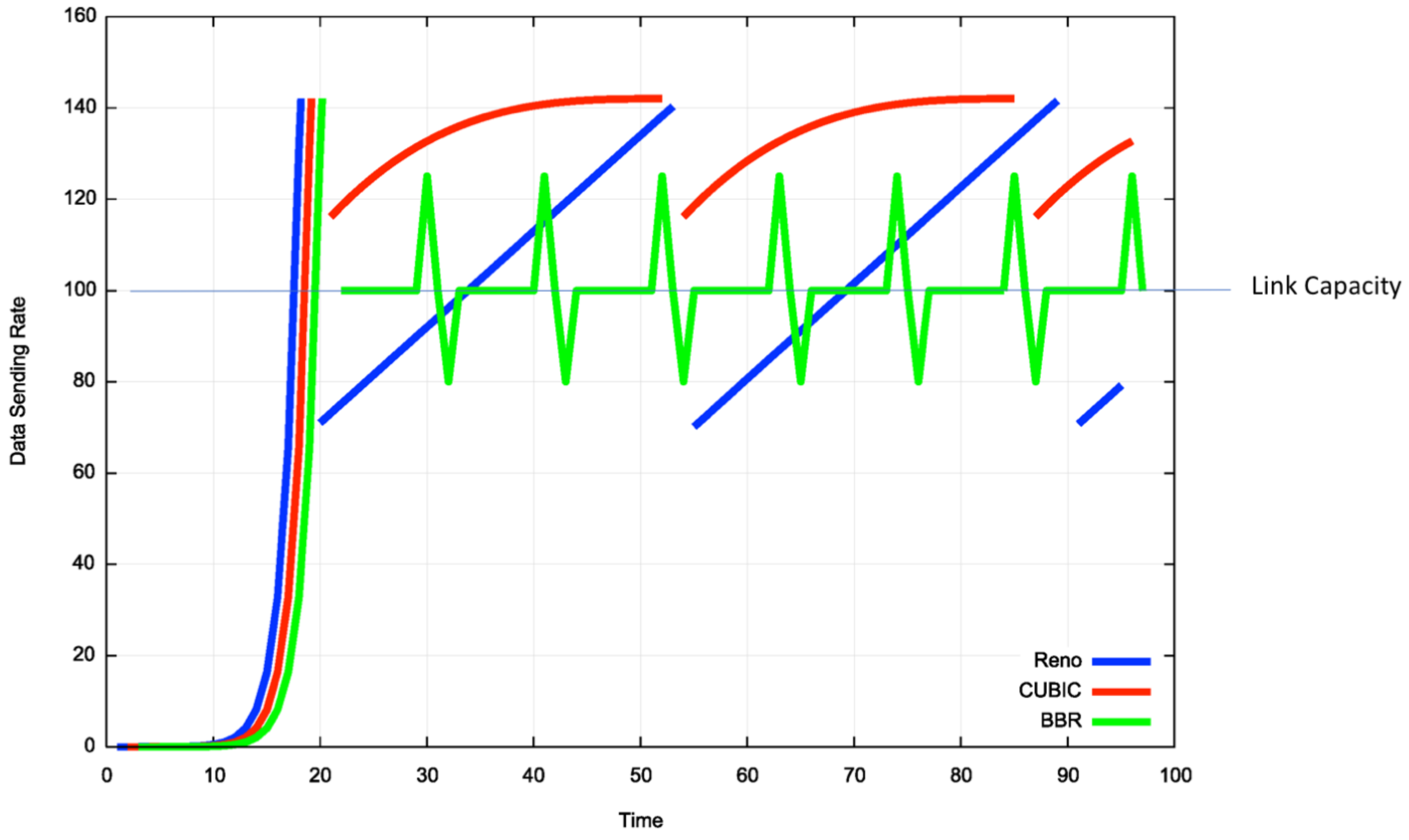

START_UP
Delivery Rate
Amount Inflight


DRAIN
Delivery Rate
Amount Inflight

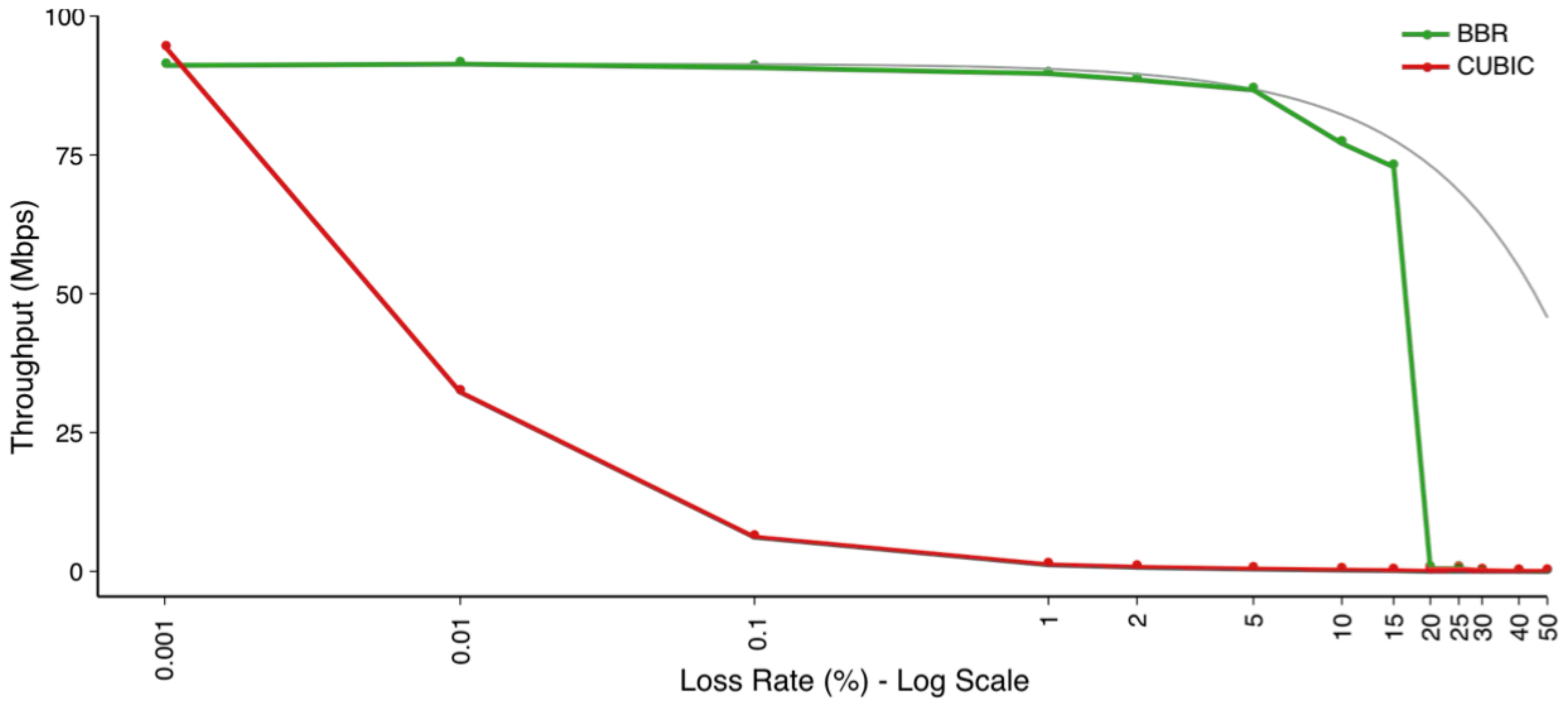

PROB_BW
Delivery Rate
Amount Inflight


PROB_RTT
Delivery Rate
Minimize packets in flight for max(0.2s, 1 round trip) after actively sending for 10s. Key for fairness among multiple BBR flows.
Amount Inflight

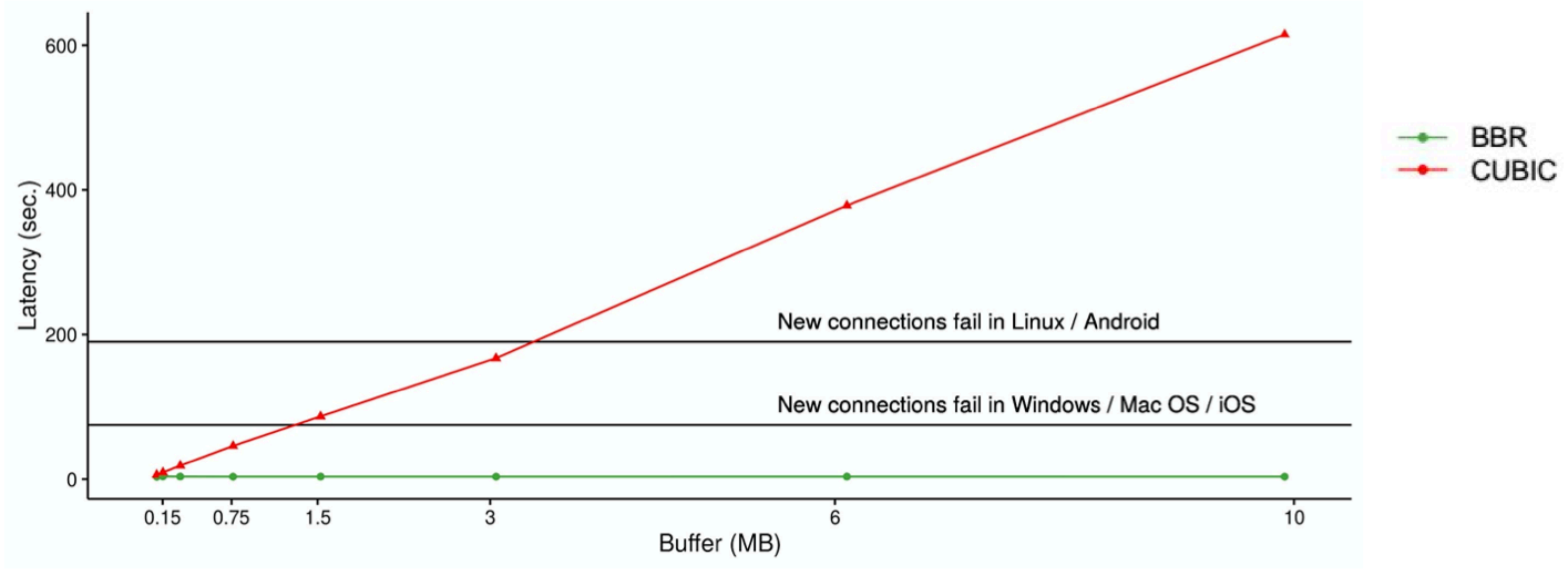
BBR, Reno, and CUBIC



BBR performance gains

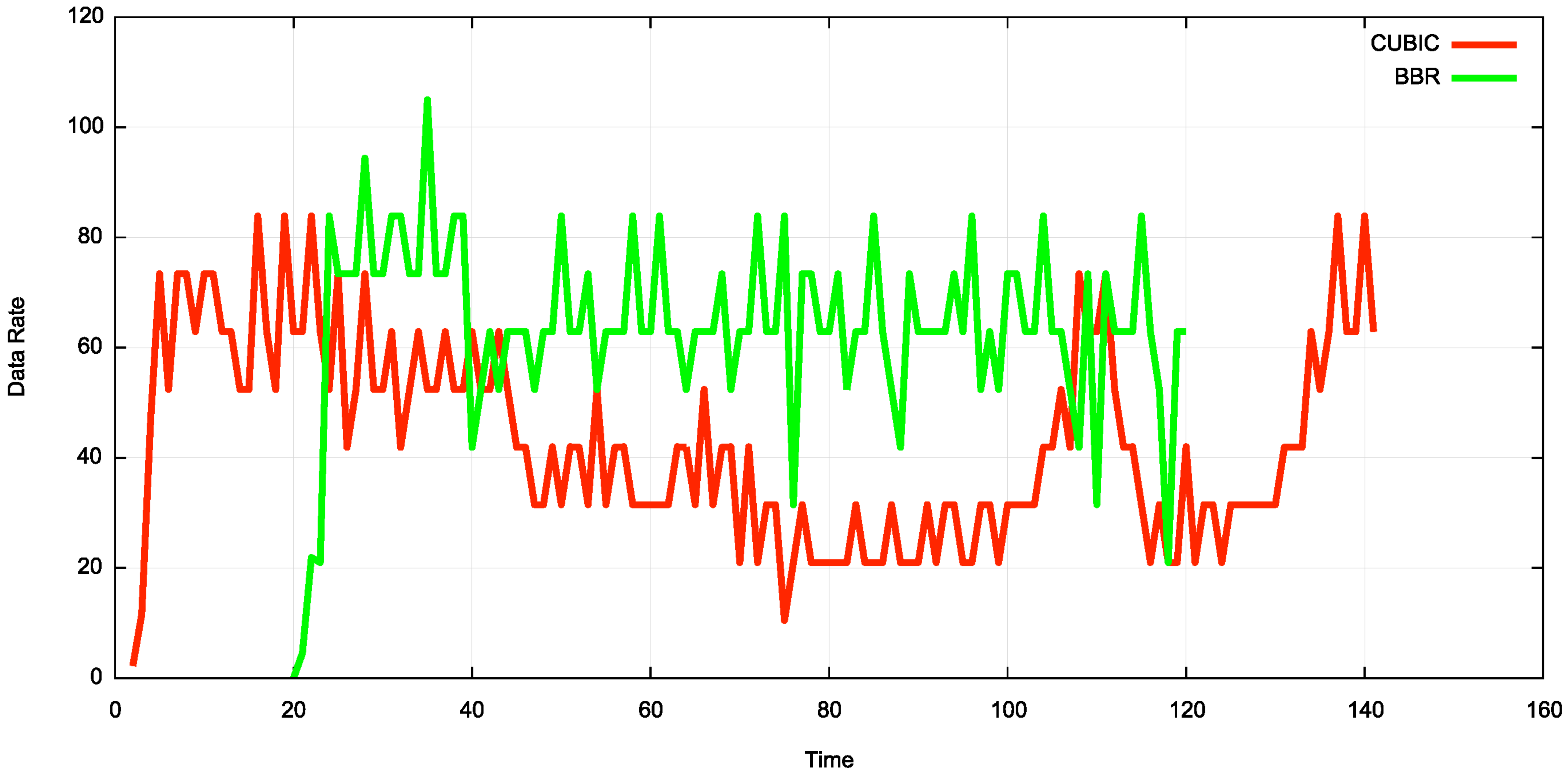


Fully utilizes the bandwidth, despite the high loss



Low queue delay, despite bloated buffer

BBR defending against CUBIC



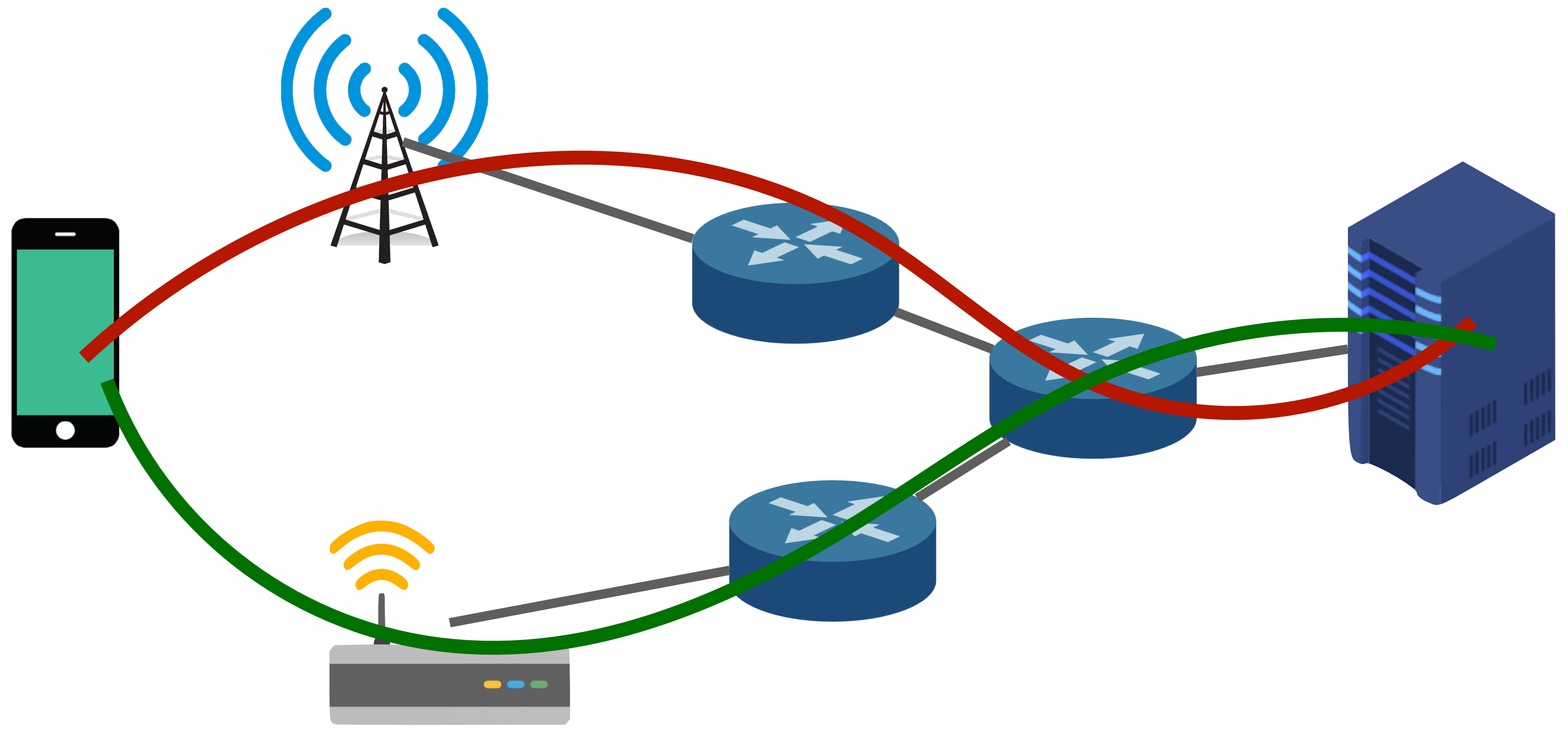
BBR is able to compete with other flows and defend its bandwidth

Questions?

Multi-path transport

Multi-homed devices become popular

- Mobile devices (with cellular and WiFi at the same time)
- High-end servers (multiple NICs)
- Data centers (rich connectivity with many paths)



Benefits of multi-path

- Higher **throughput**, **failover** from one path to another
- Seamless **mobility**

Working with unmodified applications

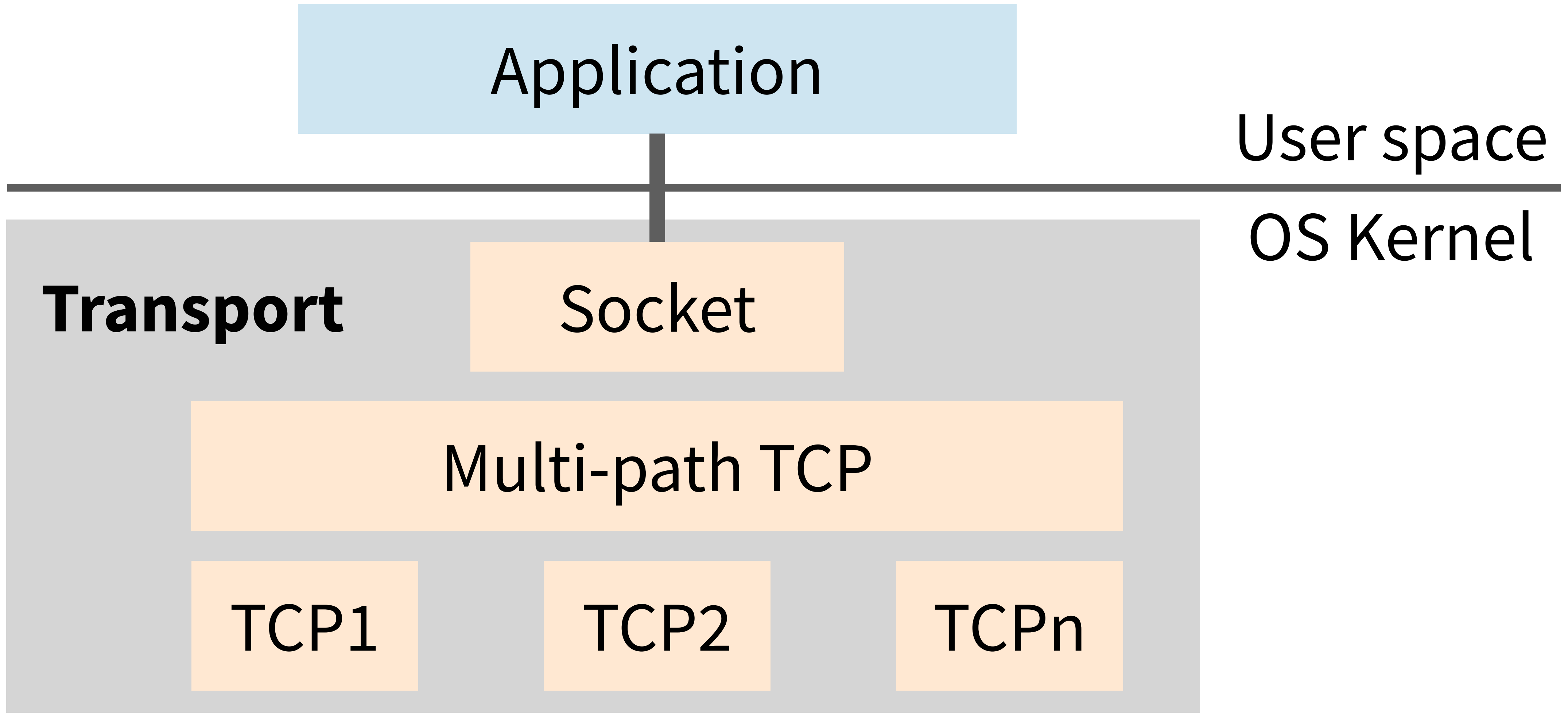
Present the same socket API and expectations

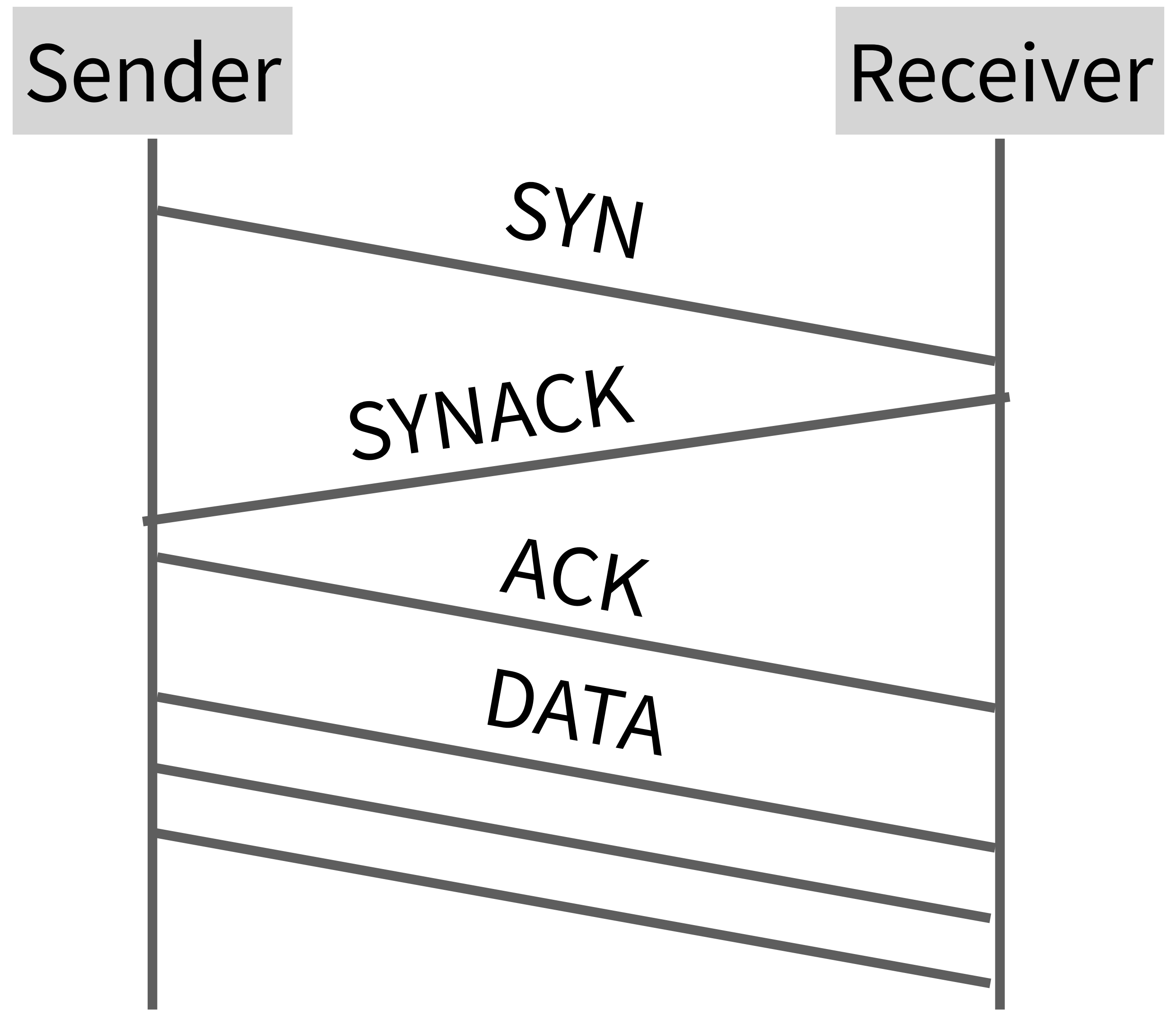
- Connection identified by the five tuples (source/dest IP, source/dest port, protocol)

Establish TCP connections in the normal way

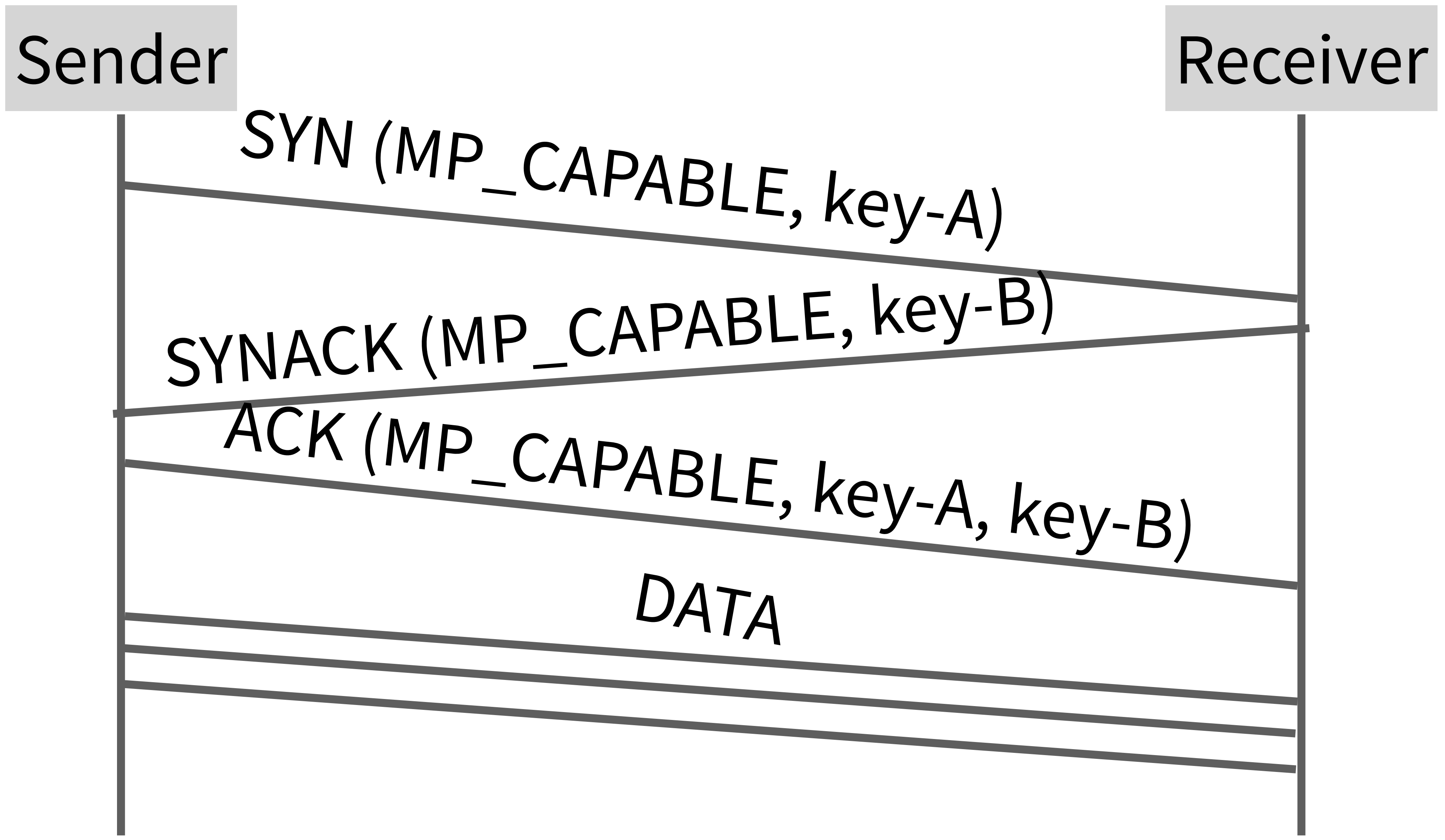
- Create a socket to a single IP address/ port

Single flow by default, adding sub-flows if possible





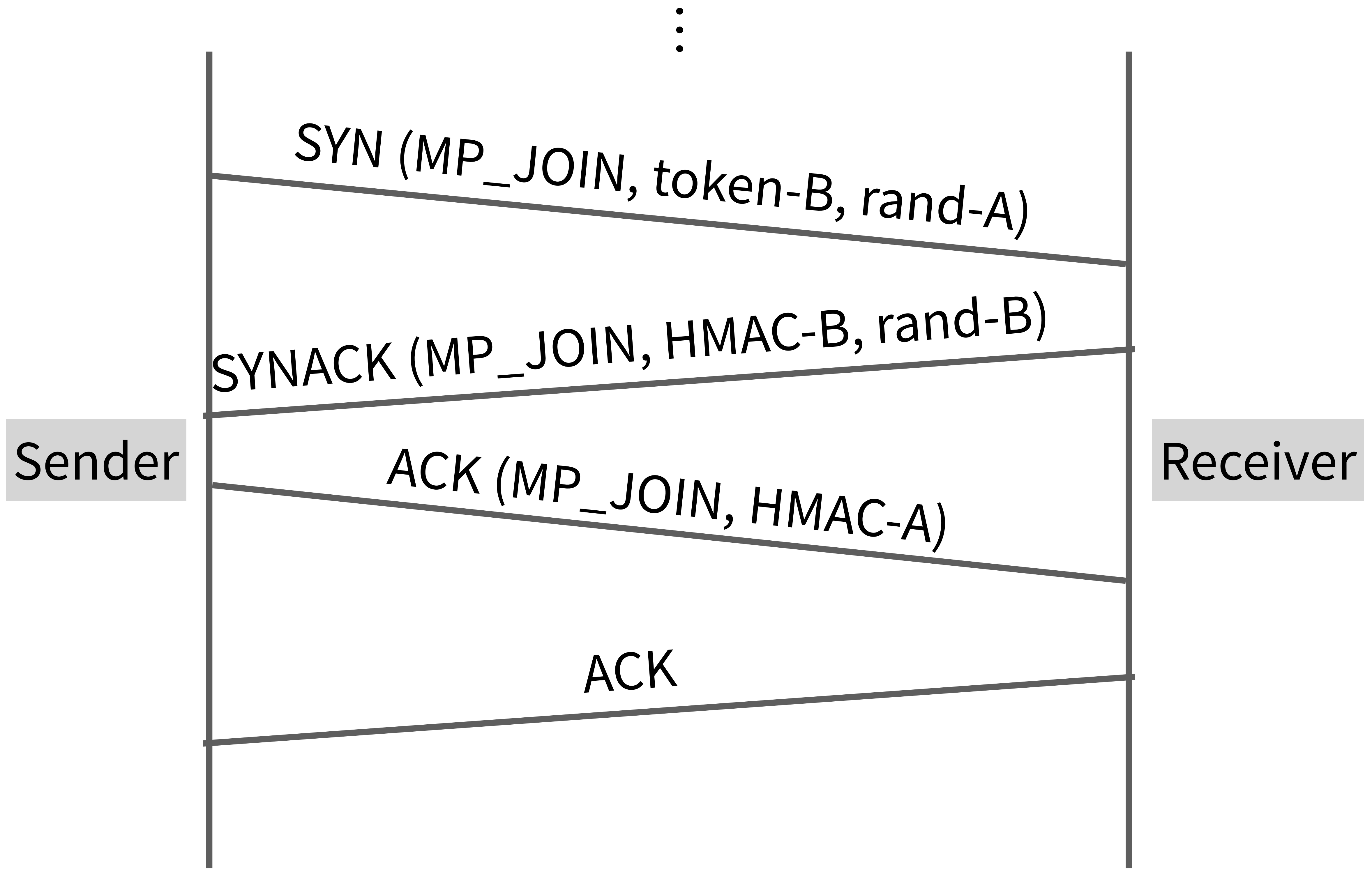
MPTCP: capability negotiation



During the TCP 3-way handshake phase, set the option "MP_CAPABLE" in TCP header

If SYNACK does not contain MP_CAPABLE flag, do not try to add sub-flows

MPTCP: adding sub-flows



How to associate a new sub-flow with the connection

- Use a token hashed from the exchanged key
- Hash-based Message Authentication Code (HMAC) for authentication based on exchanged keys

How to start using the new sub-flow(s)

- Start sending packets with the new IP/port pair
- Associate the sub-flow(s) with the existing connection

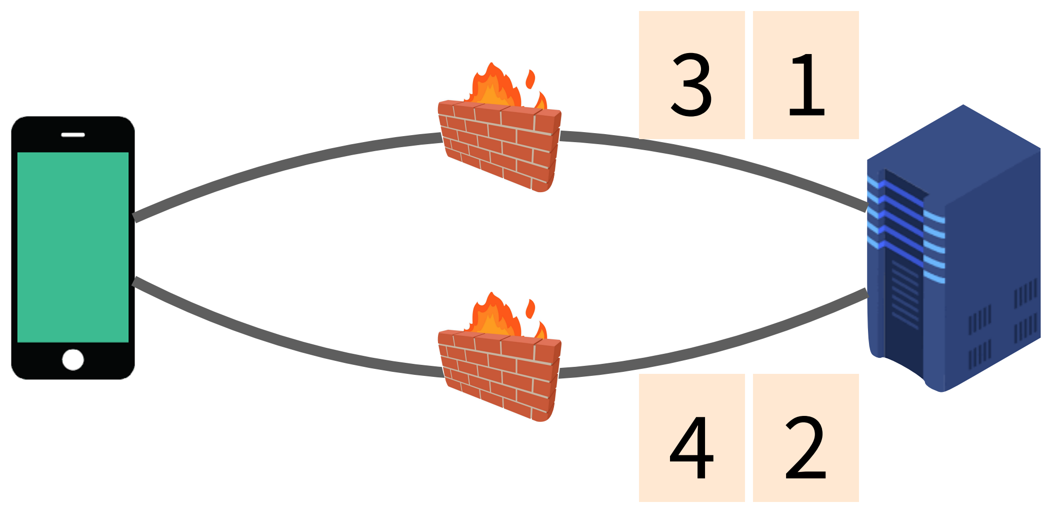
How to learn about extra IPs for new sub-flow

- One end host first establishes a new sub-flow to a known address at the other end-host

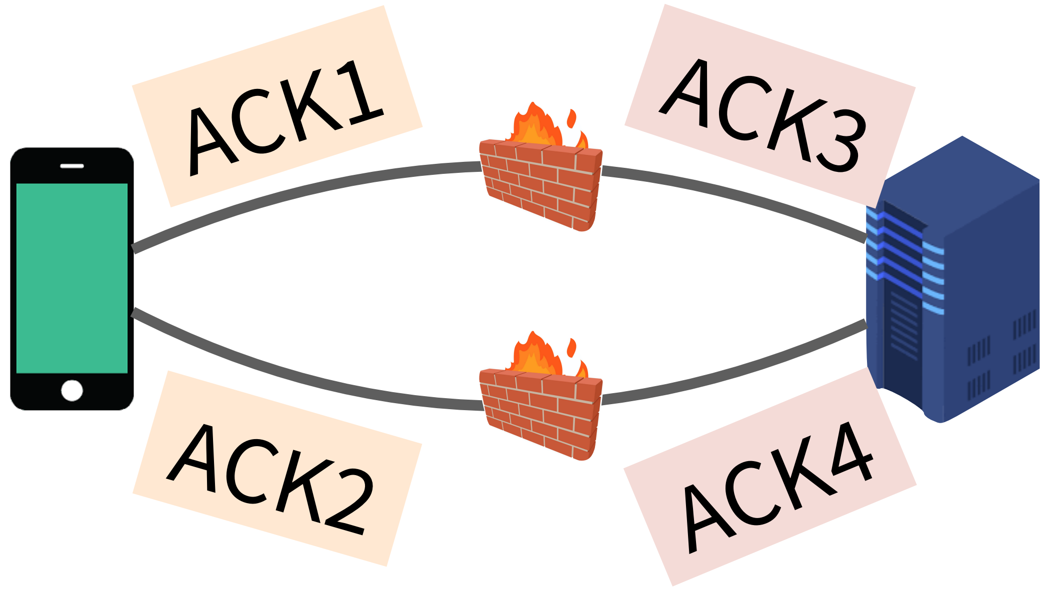
The curse of the middlebox

Middleboxes are network equipments that apply special operations on the path of network packets

- Firewall, network address translator (NAT), deep packet inspection, etc.
- Some of them inspect TCP traffic: check TCP sequence numbers

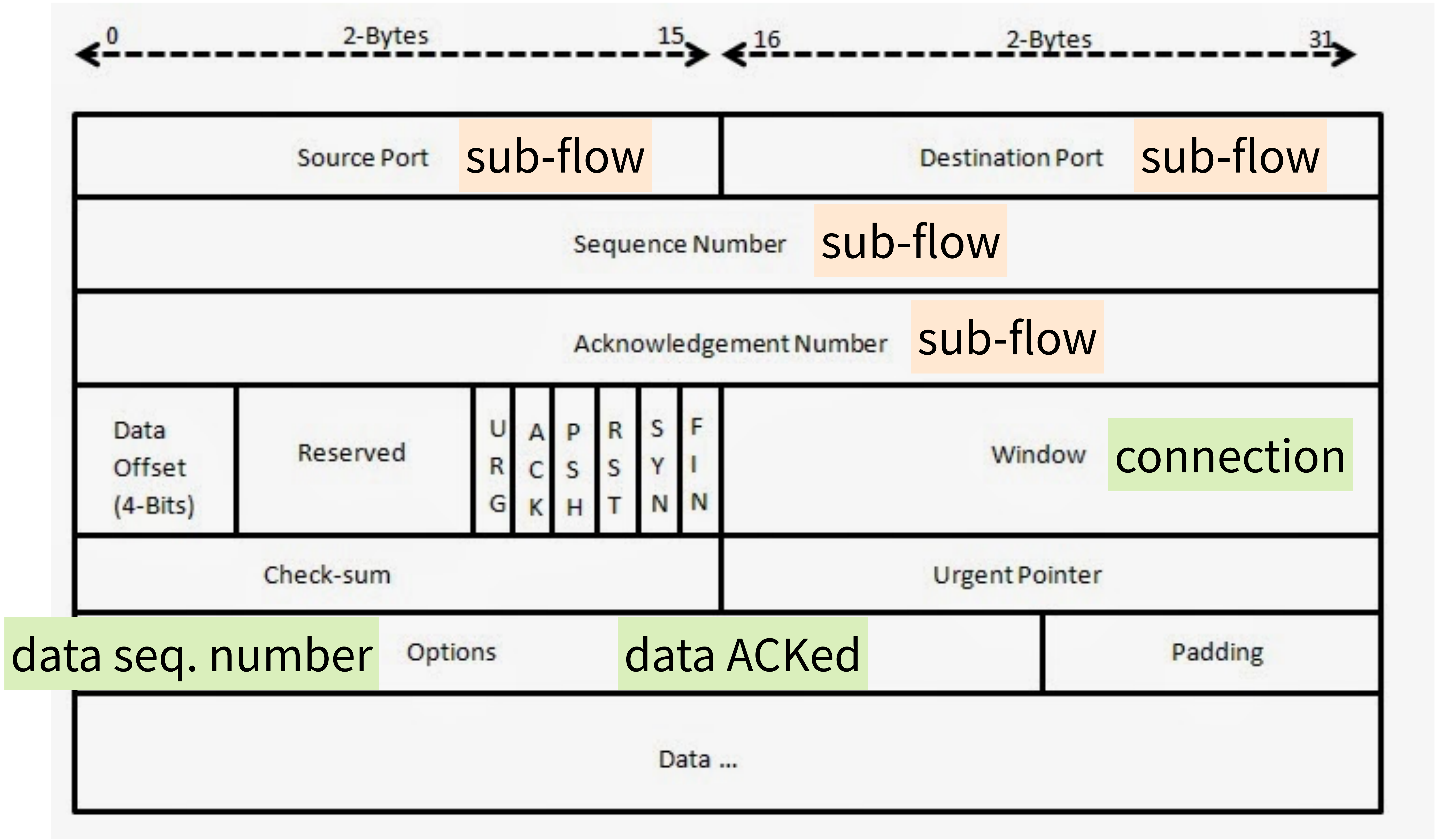


The middlebox may rewrite ACK3 to ACK2 since it has not seen ACK2 yet

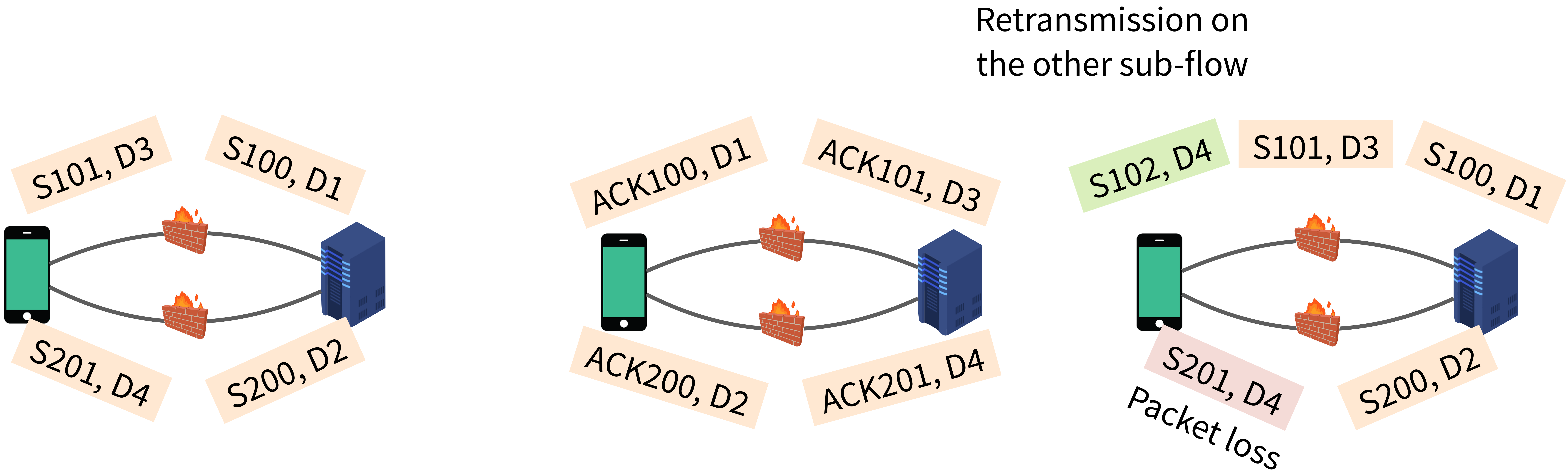


The middlebox may drop ACK4 since ACK3 is not seen

Solution: per-flow sequence number



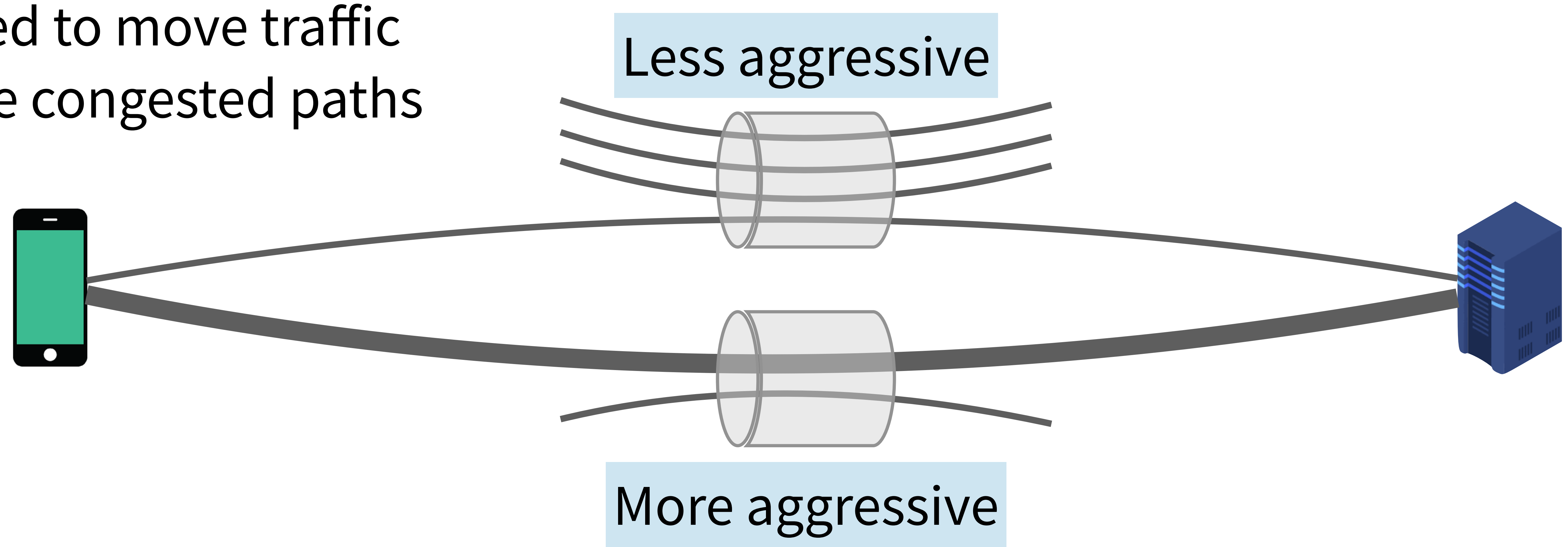
MPTCP: protocol details



All sub-flows share the same receive buffer and use the same receive window: **prevent starvation of sub-flows**, **achieve TCP-level fairness**, **prevent from buffer fragmentation**

MPTCP congestion control

Congestion control parameters for all sub-flows are linked to move traffic away from the more congested paths



Each sub-flow runs its own congestion control policy, to detect and respond to congestion it sees

MPTCP congestion control: goals

Goal 1: be fair to TCP at bottleneck links

Goal 2: use efficient paths

Goal 3: perform as well as TCP

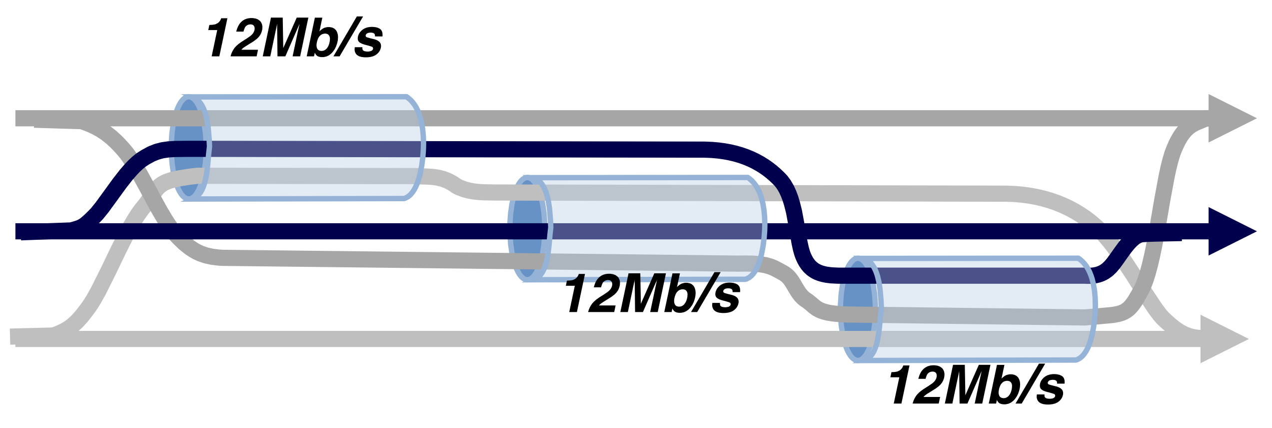
Goal 4: do not oscillate

Goal 1: be fair to TCP

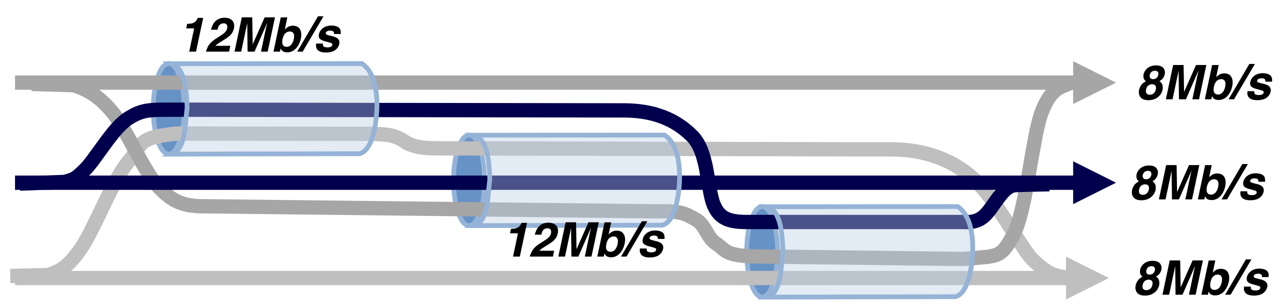
To be fair, MPTCP should take as much capacity as TCP at bottleneck link, no matter how many paths it is using

Goal 2: use efficient paths

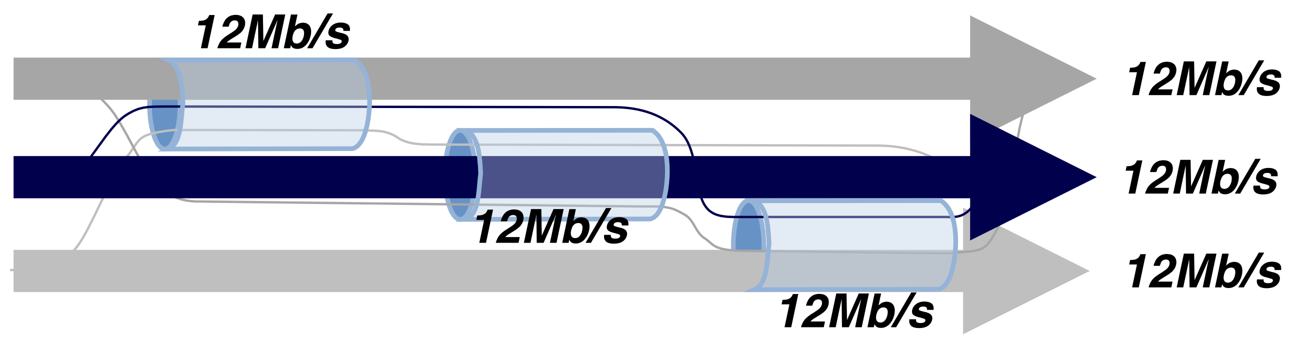
Each flow has a choice of a 1-hop and a 2-hop path. How should its traffic be split?



Fair-share



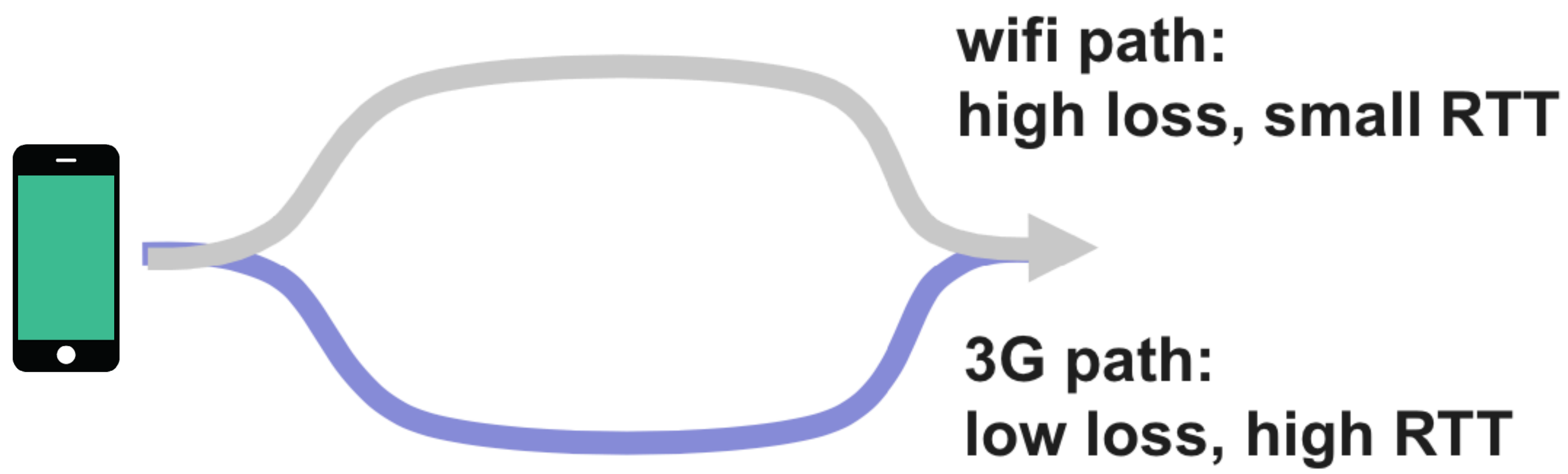
1-hop-only



Principle: each connection should send all traffic on the least-congested paths, but keep some traffic on the alternative paths as a prob

Goal 3: perform as well as TCP

Goal 2 says that we should send traffic on least congested paths, in this case the 3G path?



- **Goal 3a:** An MPTCP user should get at least as much throughput as a single-path TCP would on the best of the available paths.
- **Goal 3b:** An MPTCP flow should take no more capacity on any link than a single-path TCP would.

MPTCP congestion control

Maintain a congestion window W_r for each sub-flow $r \in R$

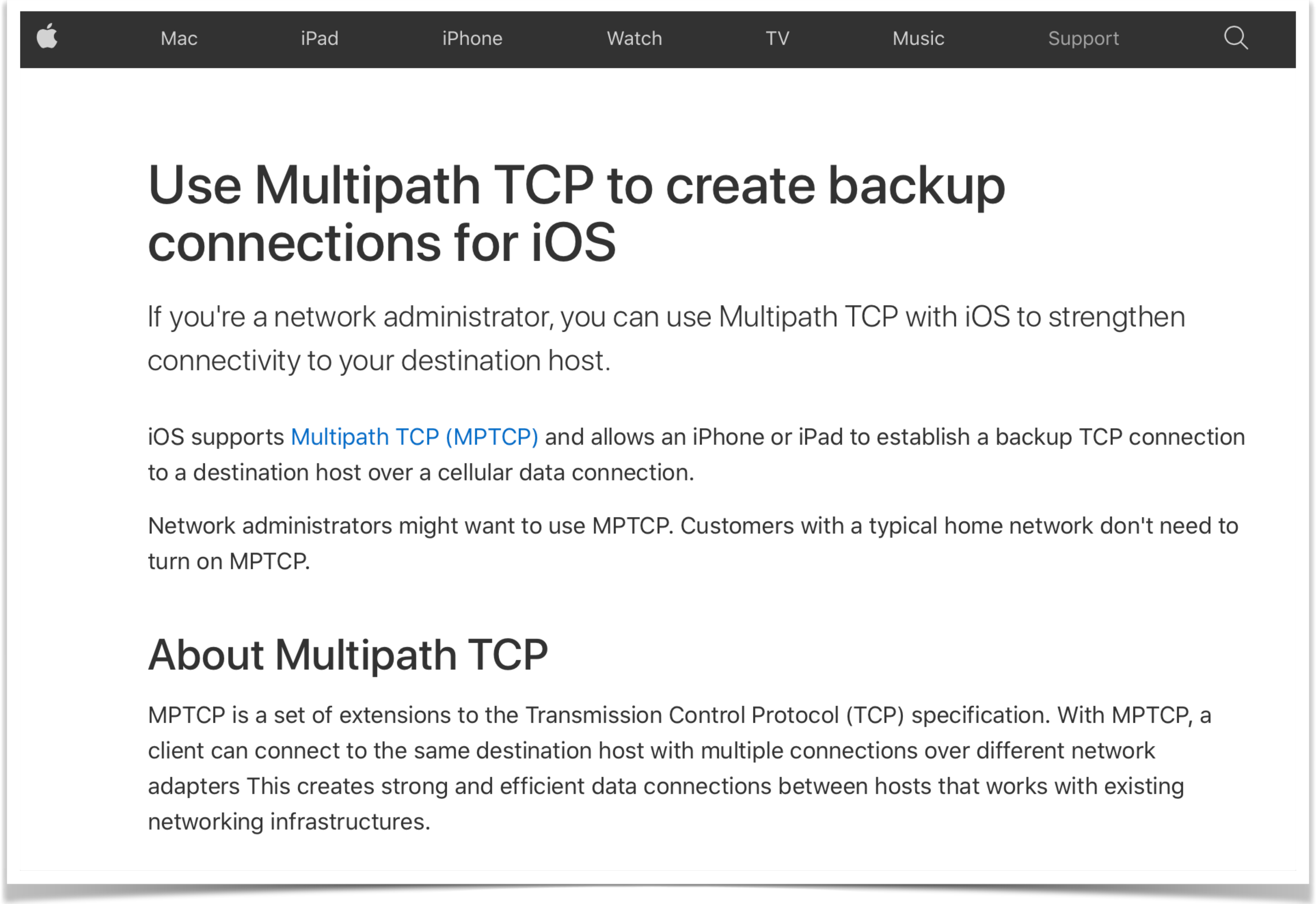
Increase W_r for each ACK on path r, by

α can be fine-tuned such that the aggregated throughput is similar to what you can get from a single-path TCP

$$\frac{\alpha(W_r)}{\sum_{r \in R} W_r}$$

Decrease W_r for each drop, by $W_r/2$

MPTCP in action



Use Multipath TCP to create backup connections for iOS

If you're a network administrator, you can use Multipath TCP with iOS to strengthen connectivity to your destination host.

iOS supports Multipath TCP (MPTCP) and allows an iPhone or iPad to establish a backup TCP connection to a destination host over a cellular data connection.

Network administrators might want to use MPTCP. Customers with a typical home network don't need to turn on MPTCP.

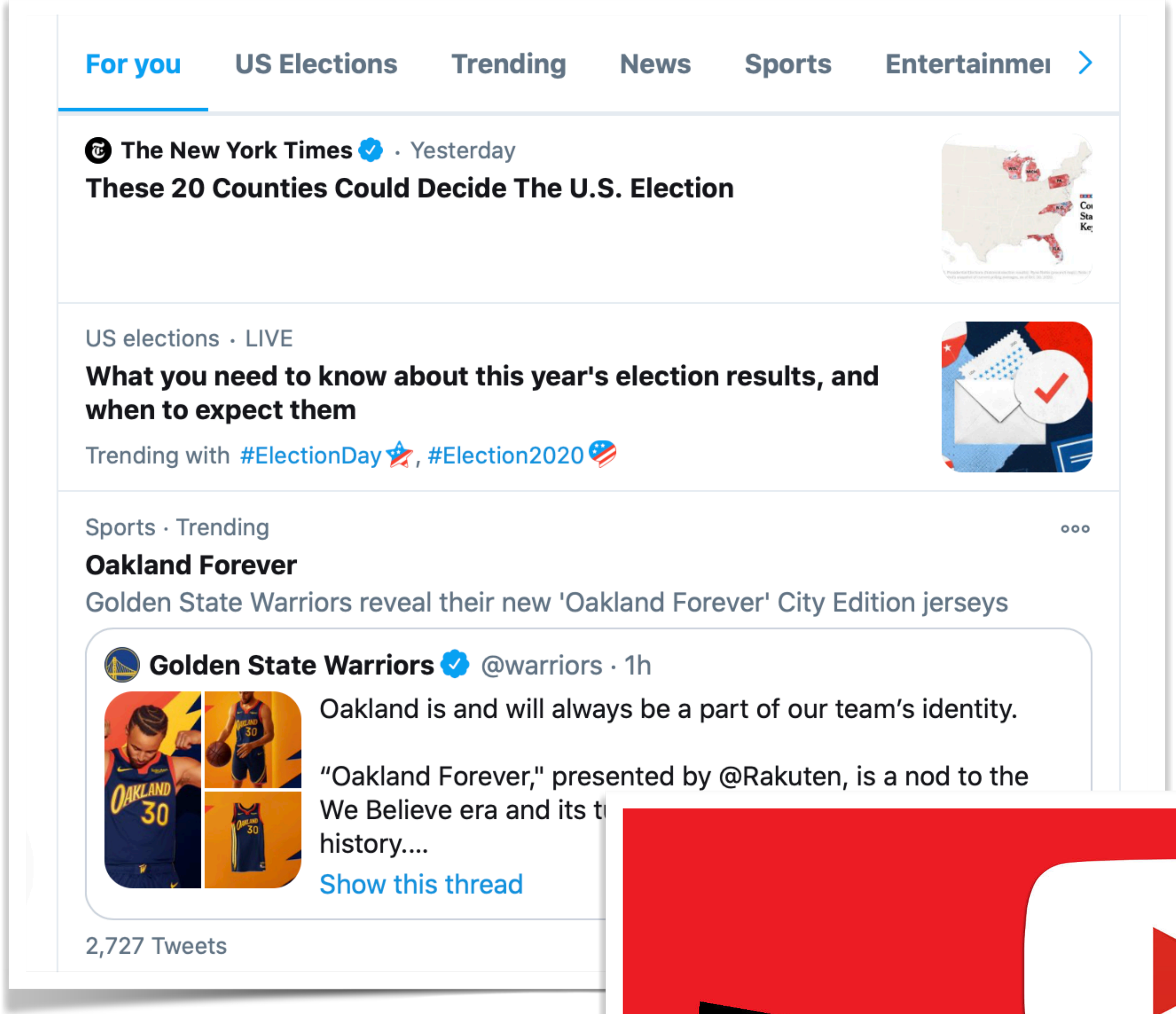
About Multipath TCP

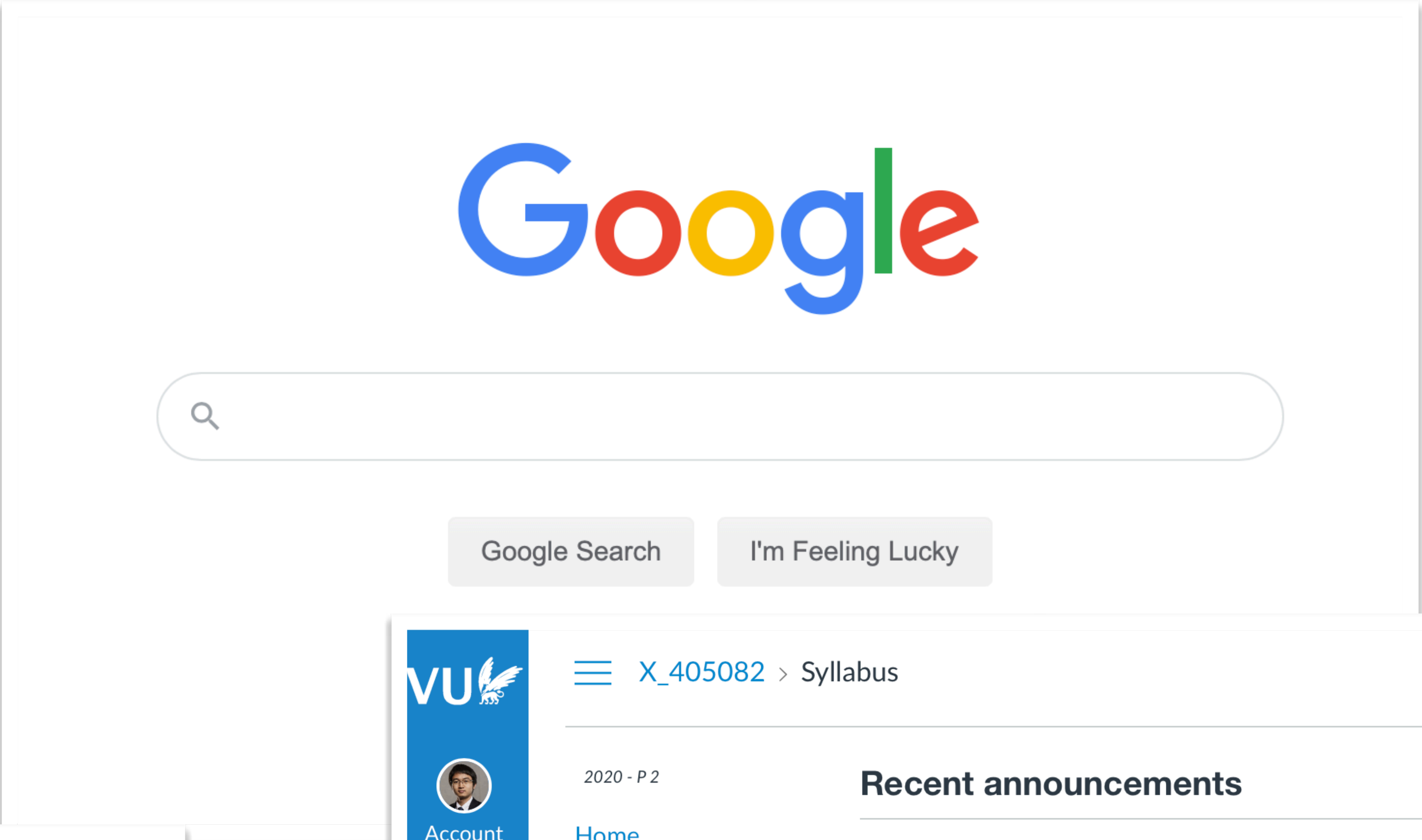
MPTCP is a set of extensions to the Transmission Control Protocol (TCP) specification. With MPTCP, a client can connect to the same destination host with multiple connections over different network adapters This creates strong and efficient data connections between hosts that works with existing networking infrastructures.

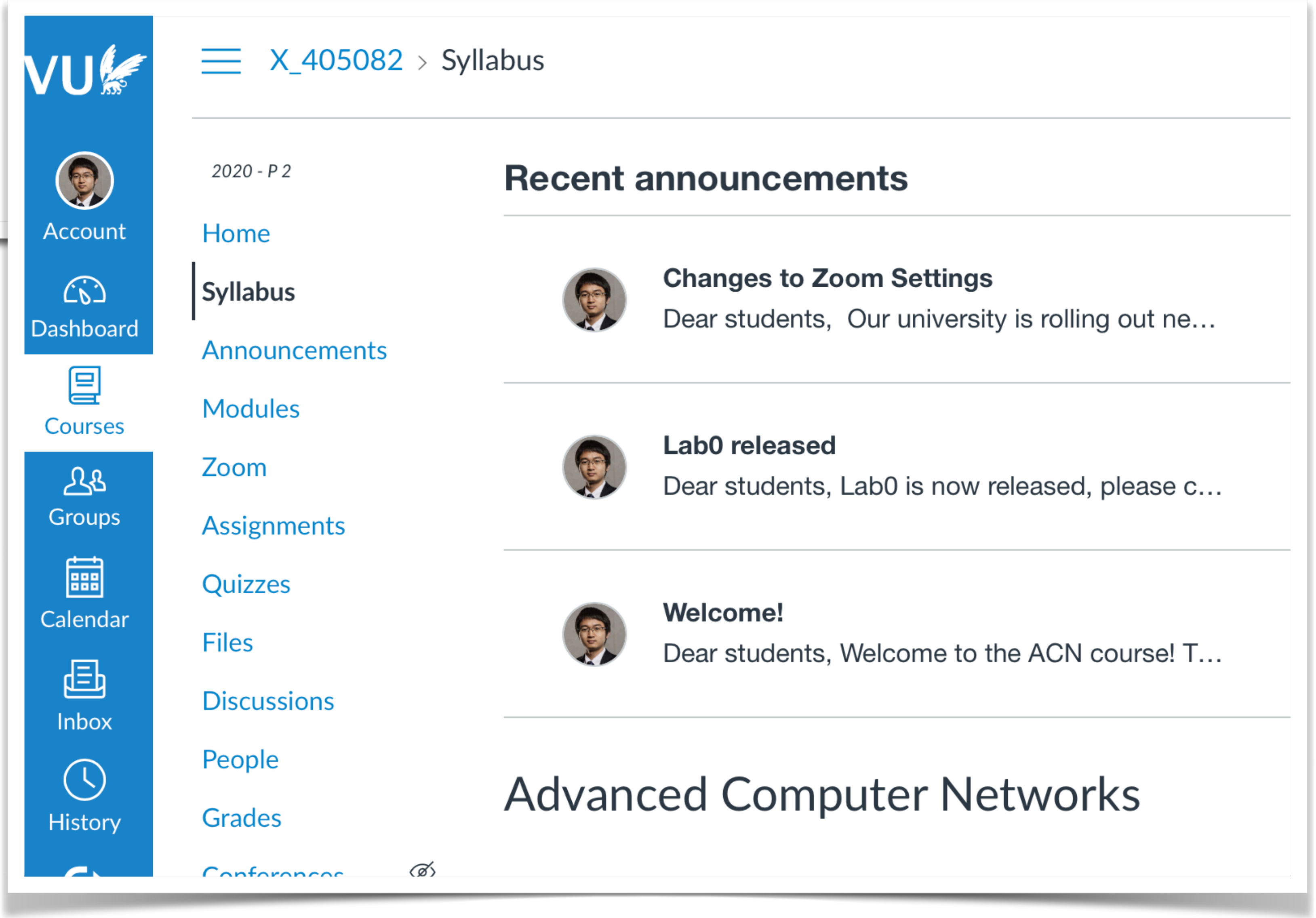
Since iOS 7 (2013), MPTCP is available on iOS devices. Cellular data is used as backup for WiFi connections when the WiFi signal is poor, for Apple services like Siri.

Questions?

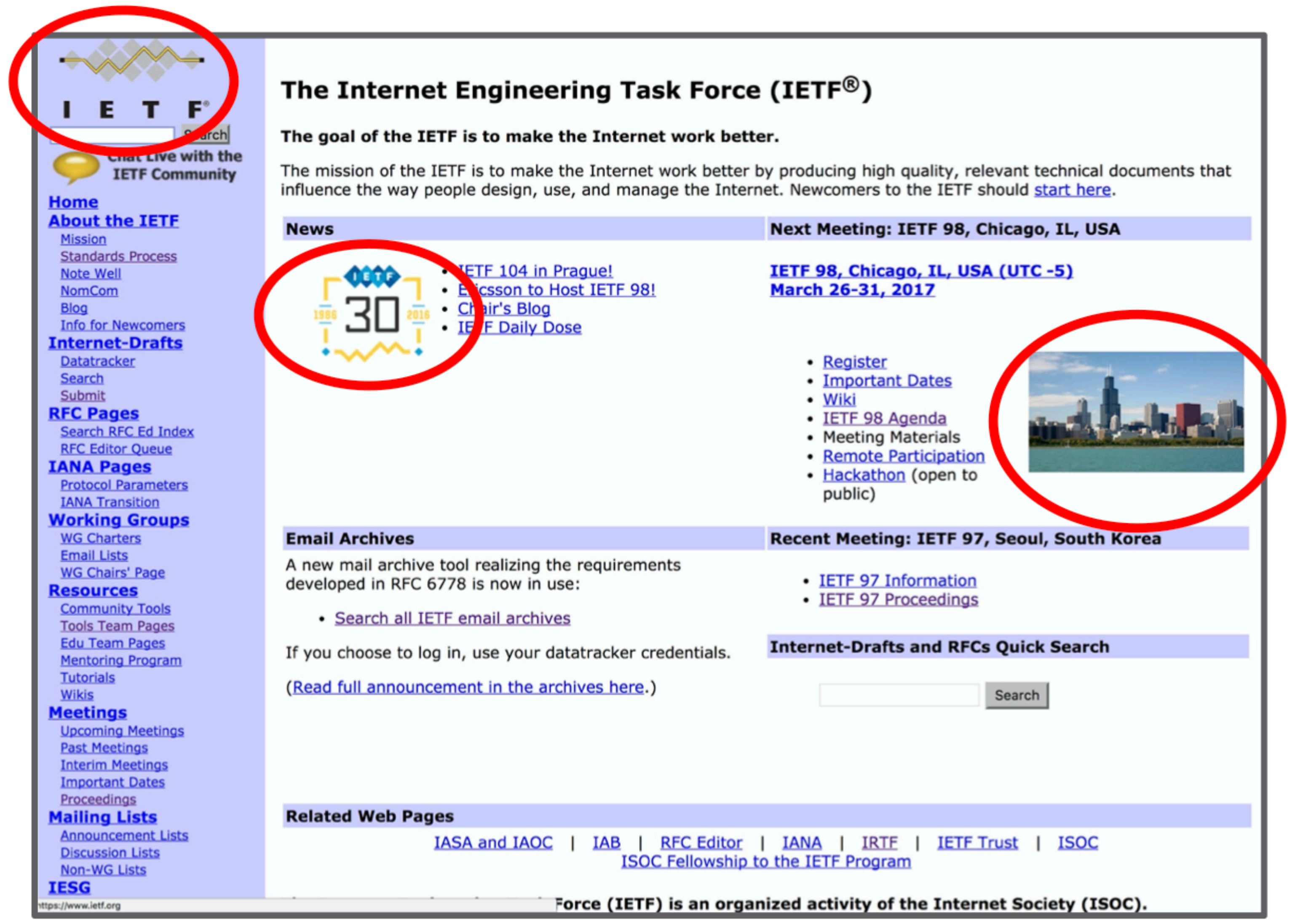
What is the most popular Internet application?

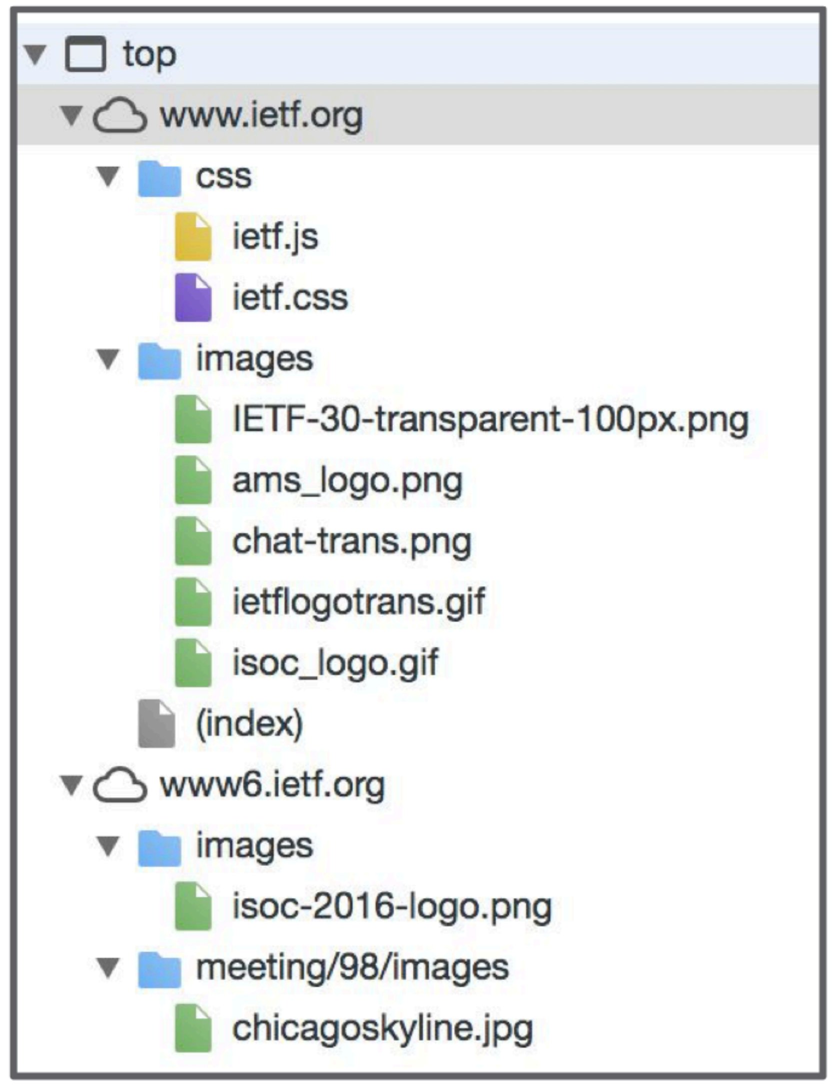






Web applications based on HTTP





A typical HTTP webpage contains **multiple objects** (text, javascript, css, images, etc.)

HTTP/1

HTTP/1

TLS*

TCP

IP

ETH

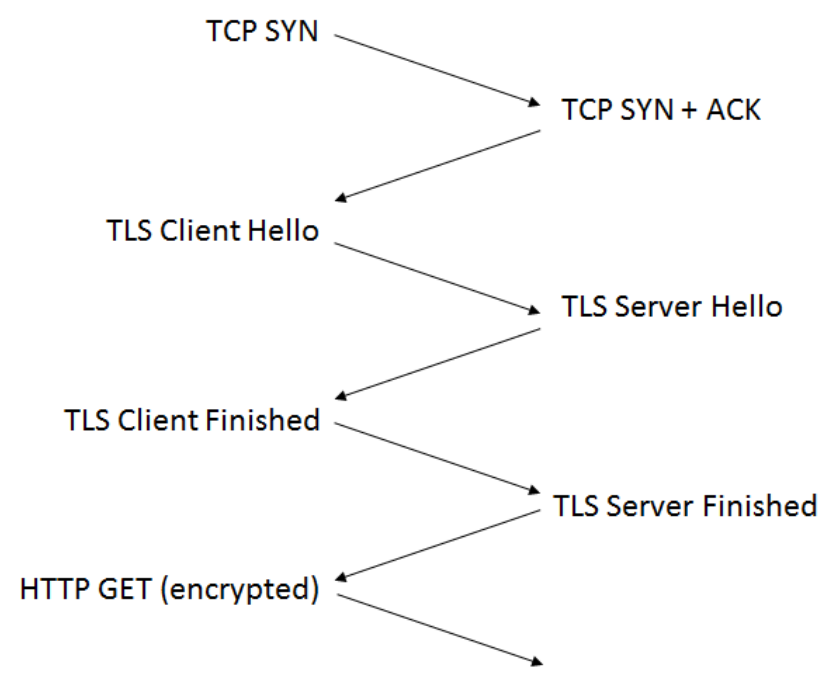
PHY

Connection setup… the long way

- 1 round trip to set up a TCP connection
- 2 round trips to set up a TLS 1.2 connection

After setup, HTTP requests/responses flow over the connection

- Only one request/response is possible at a time
- Head-of-Line (HoL) blocking on HTTP connection



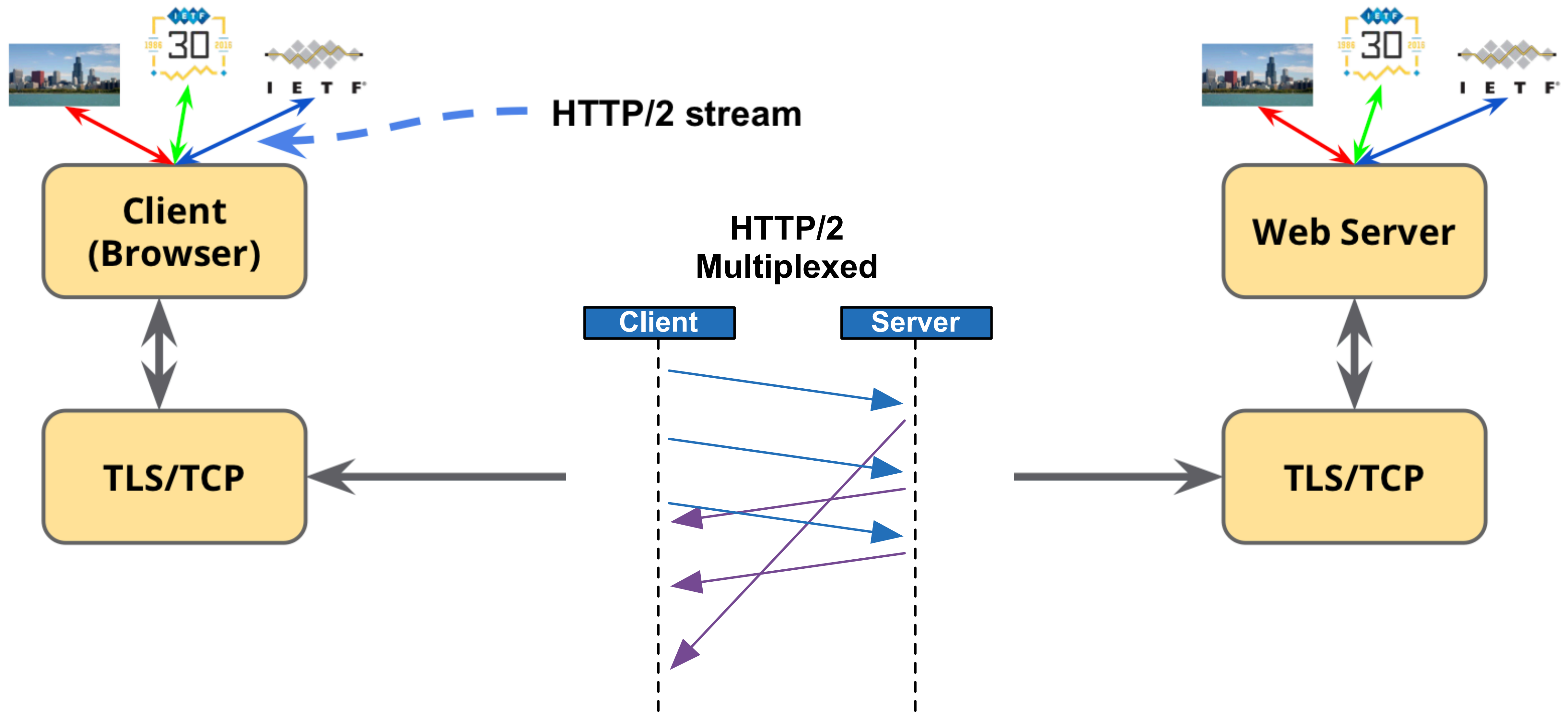
HTTP/1.1: avoid HoL blocking on the HTTP connection

Single TCP connection allows one HTTP request/response at a time, leading to HoL blocking

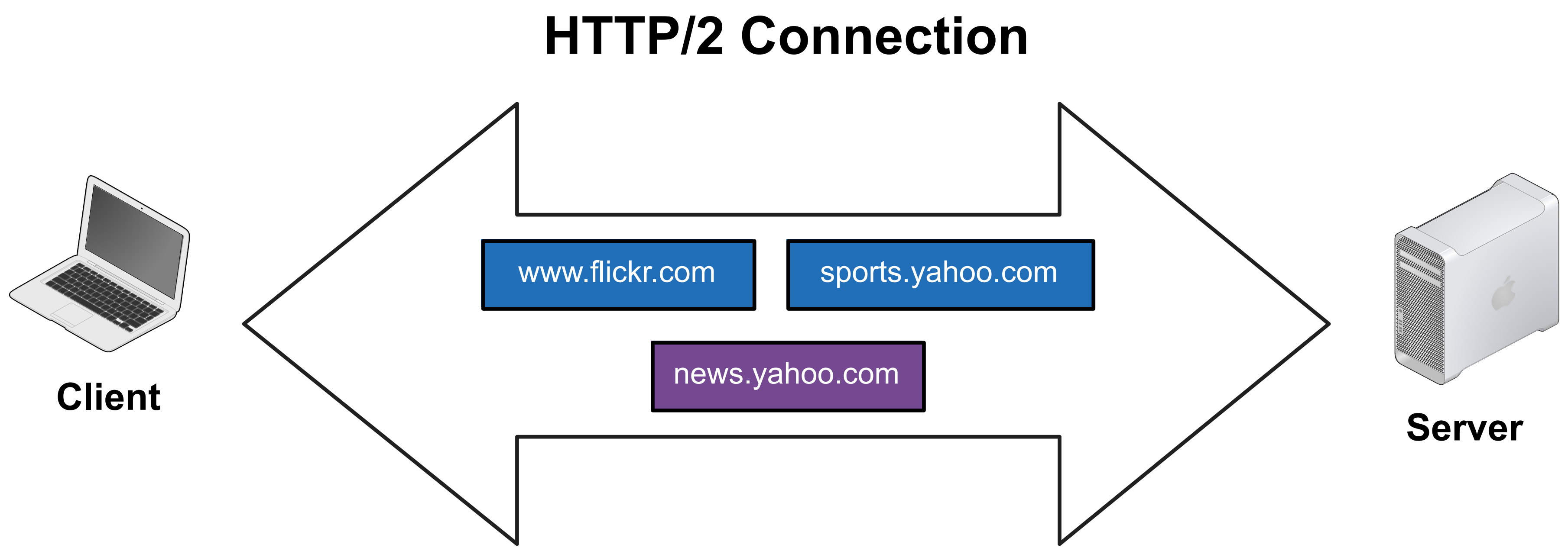
Multiple TCP connections allow multiple objects to be fetched through concurrent HTTP requests/responses

HTTP/2: stream multiplexing

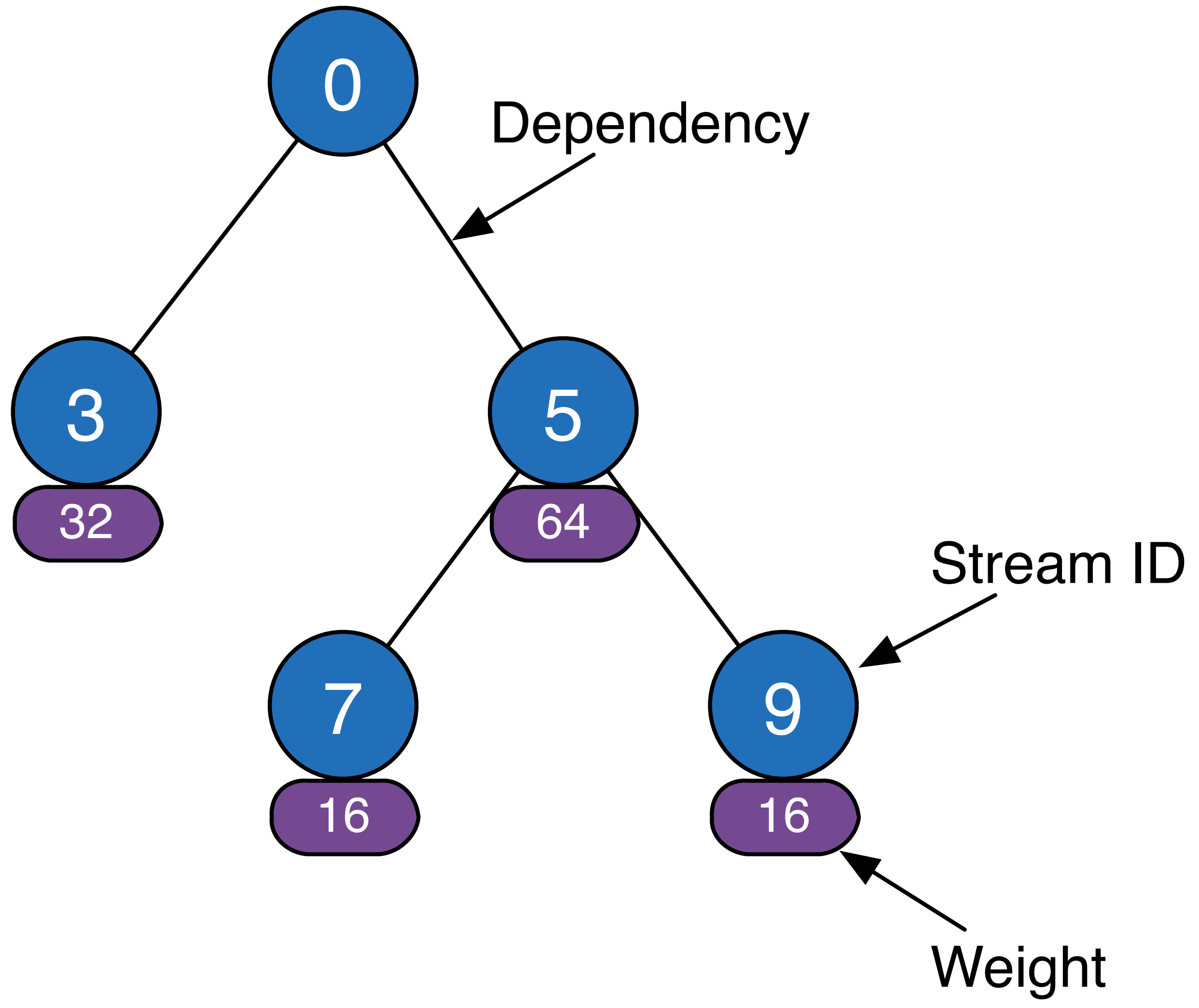
Multiple streams (each for an object) are multiplexed on the same TCP connection



HTTP/2: other properties



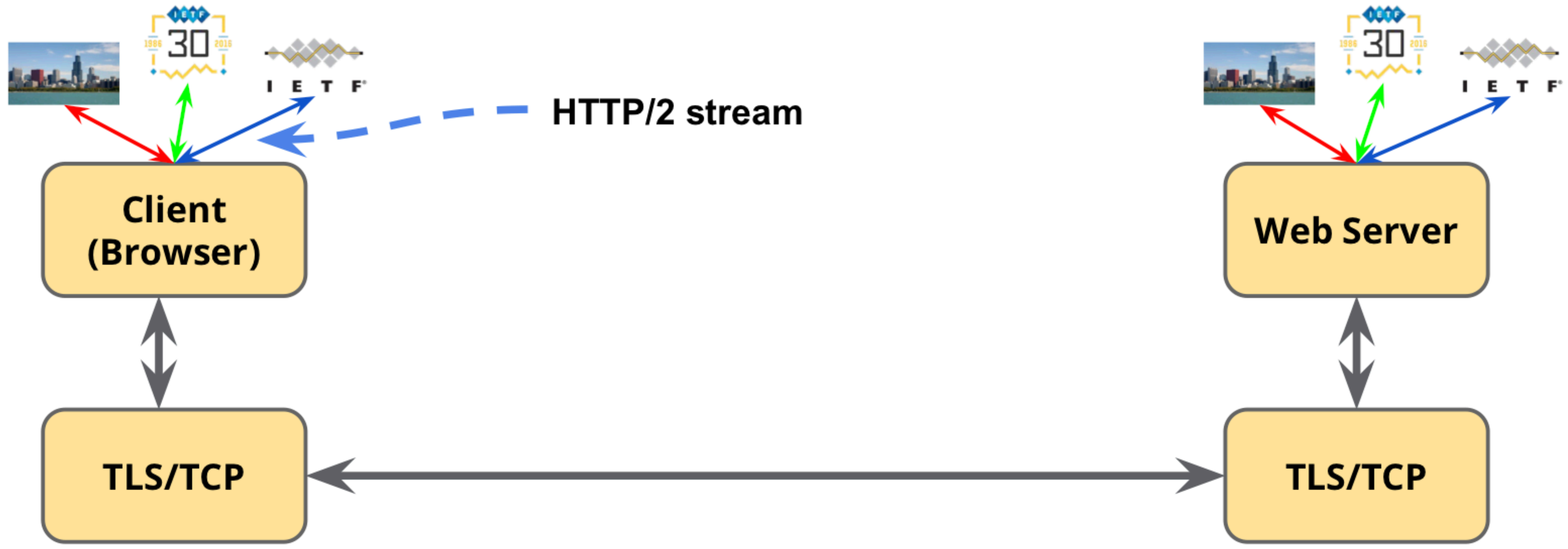
Even multiple domains (beyond the single website) can share the same TCP connection



Supports priority of streams set by the client (dependency tree)

HTTP/2 HoL problem

No HoL blocking on the HTTP connection since HTTP requests can be fired concurrently



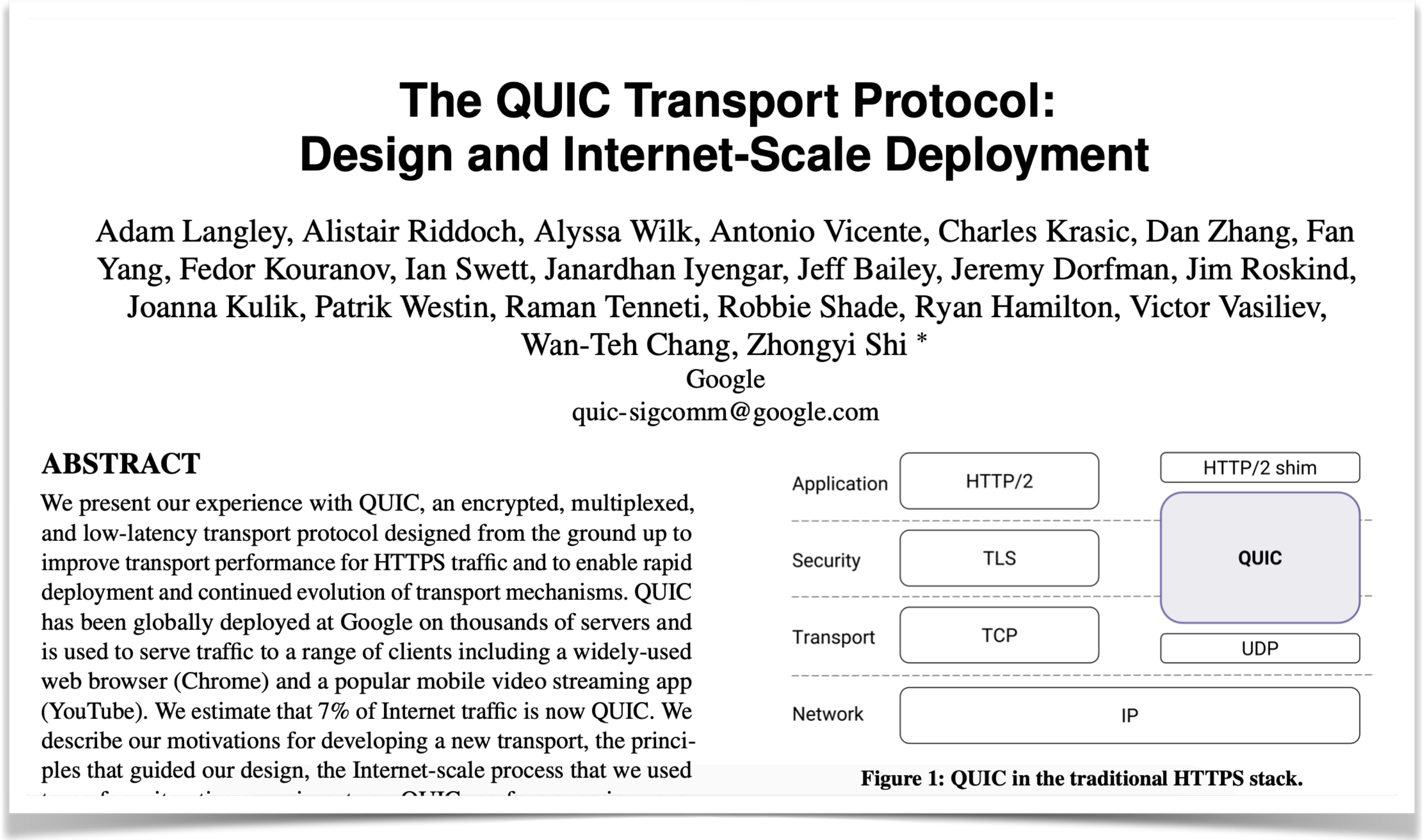
HoL blocking on the TCP connection: a retransmission for a packet for one object would delay the transmissions of others

QUIC

A new streaming protocol to make streaming **faster**

Experimental protocol, deployed at Google starting in 2014

- Between Google services and Chrome
- Improved page load latency, video rebuffer rate
- More than 75% Internet traffic based on QUIC/ HTTP3.0 by 2020
- Akamai deployment in 2016, Facebook deployment in 2020



The QUIC Transport Protocol: Design and Internet-Scale Deployment

Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, Zhongyi Shi *
Google
quic-sigcomm@google.com

ABSTRACT

We present our experience with QUIC, an encrypted, multiplexed, and low-latency transport protocol designed from the ground up to improve transport performance for HTTPS traffic and to enable rapid deployment and continued evolution of transport mechanisms. QUIC has been globally deployed at Google on thousands of servers and is used to serve traffic to a range of clients including a widely-used web browser (Chrome) and a popular mobile video streaming app (YouTube). We estimate that 7% of Internet traffic is now QUIC. We describe our motivations for developing a new transport, the principles that guided our design, the Internet-scale process that we used

Figure 1: QUIC in the traditional HTTPS stack.

ACM SIGCOMM 2017

HTTP/3 over QUIC

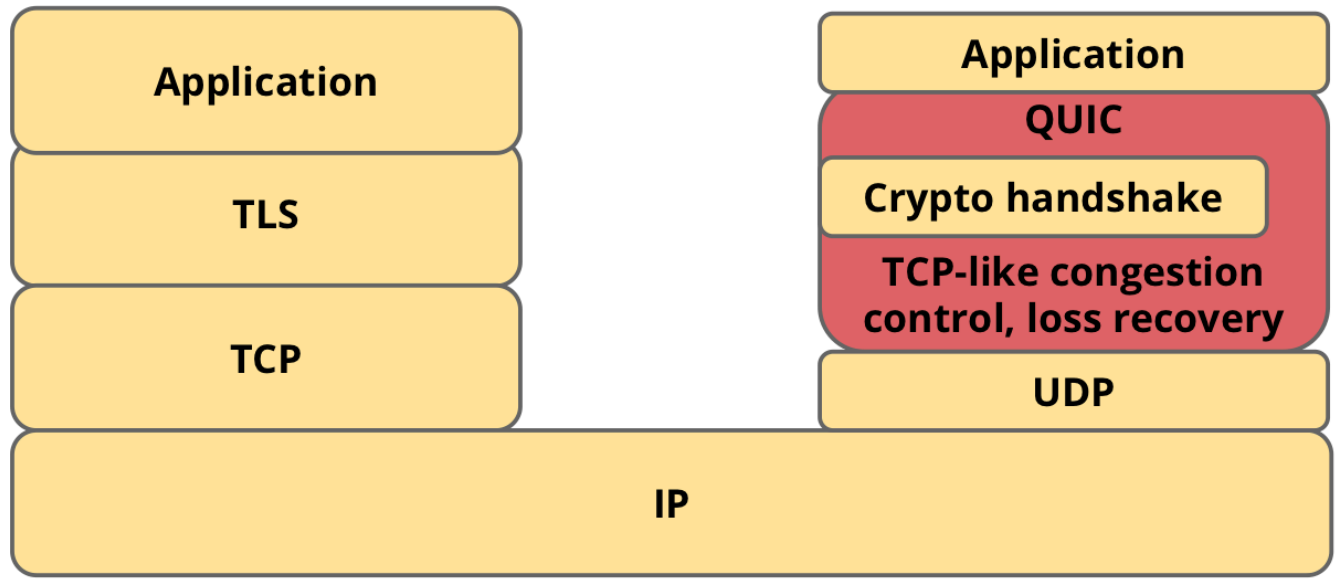
HTTP/2 → HTTP/3 for QUIC compatibility

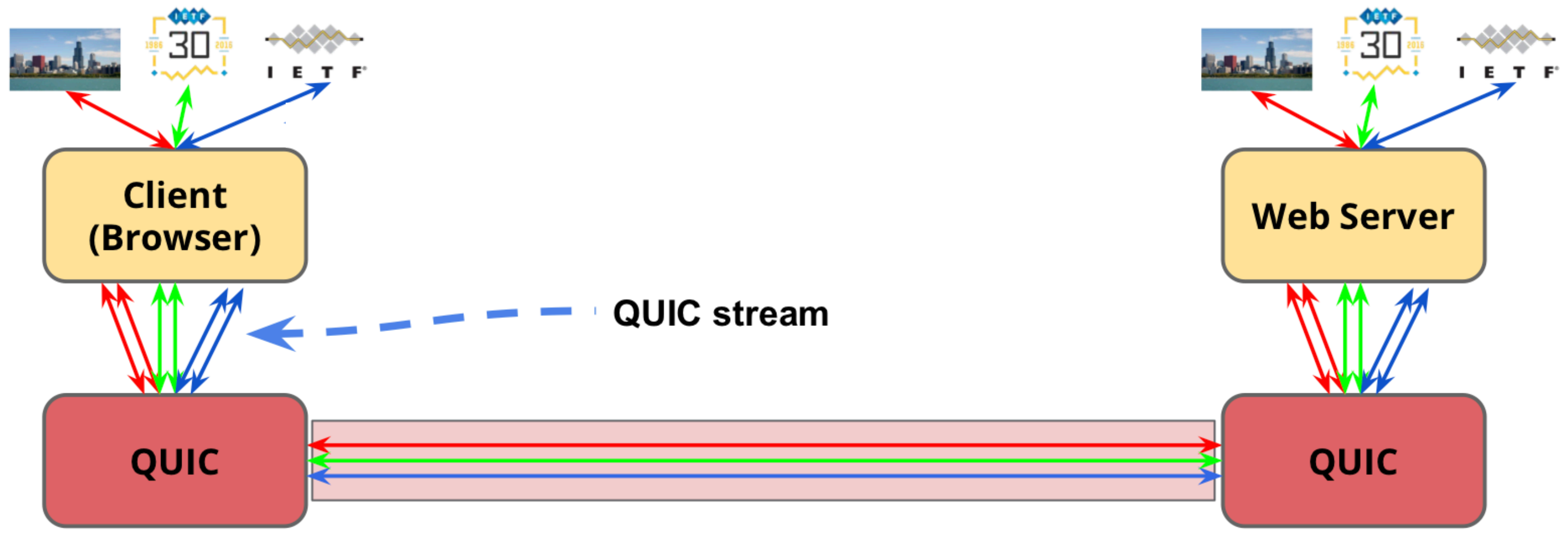
Connection setup, the QUIC way

- 0 round trip to a known server (common)
- 0 round trip if crypto keys are not new
- Connections survive IP addresses change

After setup, HTTP requests/responses flow over the connection via QUIC streams

User-space protocol, can adopt congestion control protocols like BBR





Avoid HoL blocking on the transport by having multiple streams over UDP

Summary

Network transport

- How does TCP work: goal of TCP, slow-start, AIMD
- What are the different congestion avoidance mechanisms (TCP variants)
- BBR: congestion-based congestion control

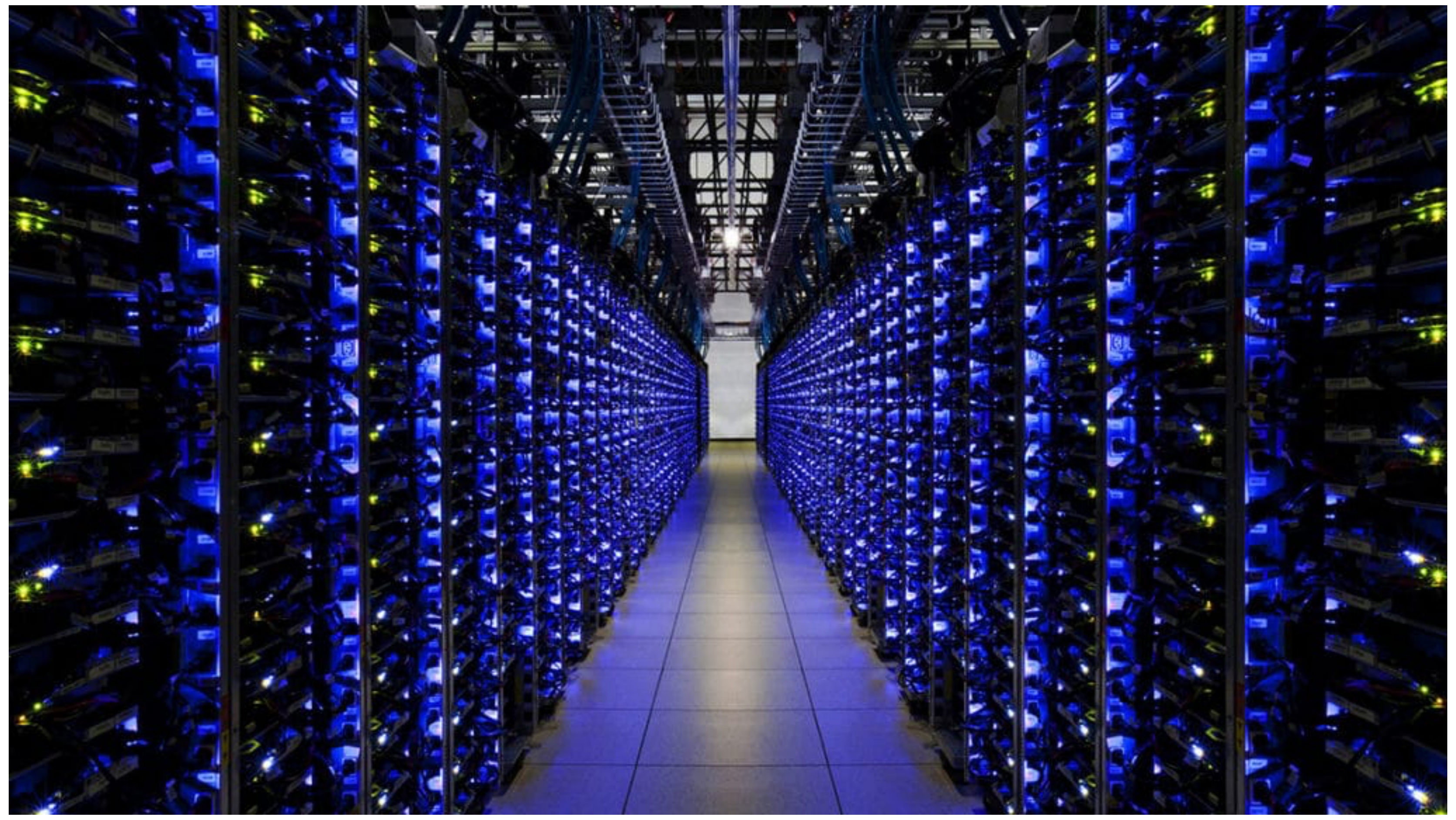
Multi-path TCP

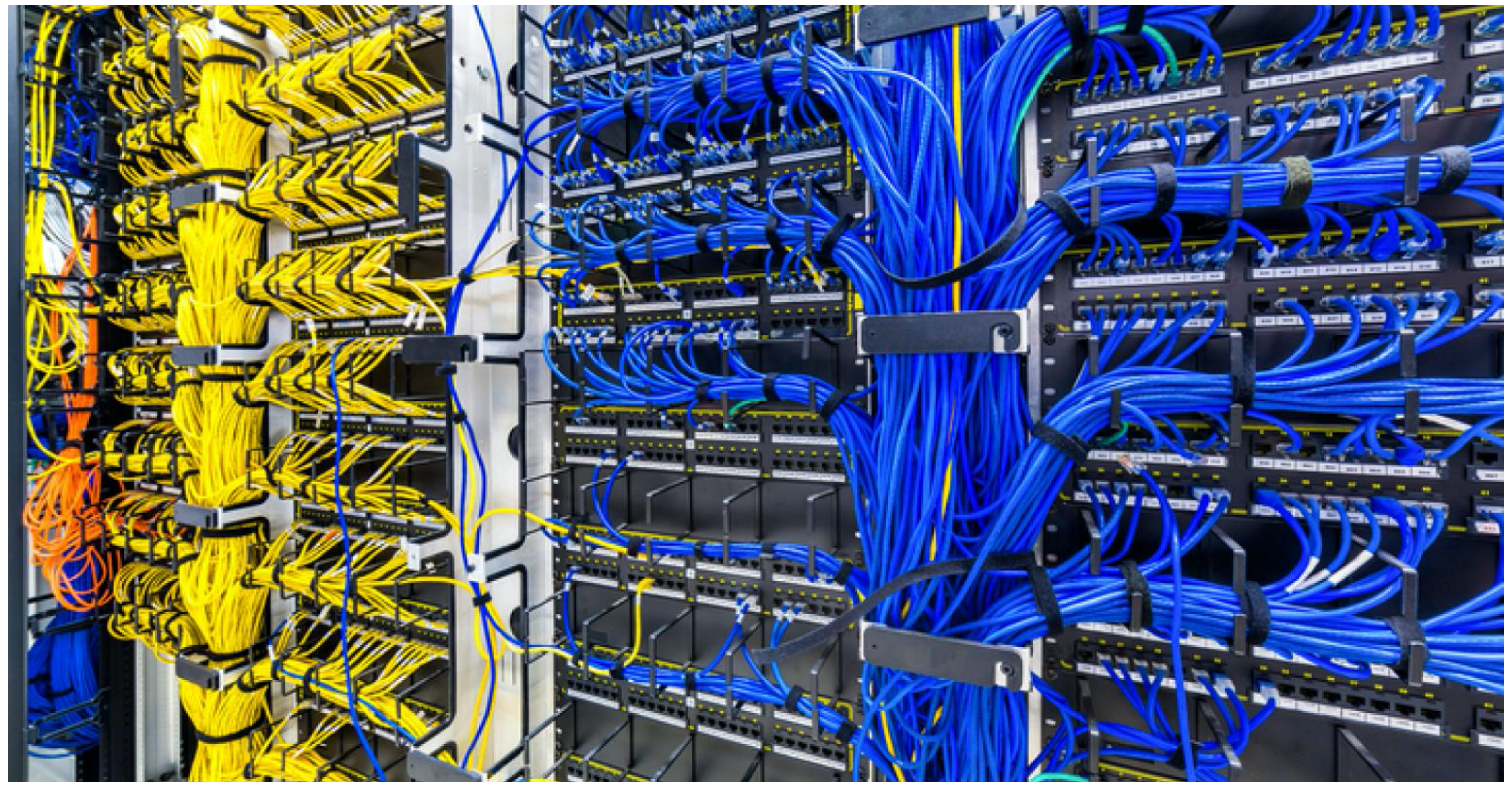
- How does MPTCP work: connection establishment, adding sub-flows, sequence numbers
- Congestion control in MPTCP: separate congestion control on sub-flows, but linked to achieve goal goals

QUIC

- The evolution of HTTP: what are the main problems in each version of HTTP
- HTTP/3 over QUIC: zero-RTT connection establishment, no HoL blocking

Next time: data center networking





How to build a high-performance network for data centres?