Taking a Long Look at QUIC

Arash Molavi Kakhki, Samuel Jero, David Choffnes,
Cristina Nita-Rotaru, and Alan Mislove

ThousandEyes

Purdue University

Northeastern University
College of Computer and Information Science

Why do we need yet another protocol?

Internet connectivity is a critical service!

Why do we need yet another protocol?

Internet connectivity is a critical service!

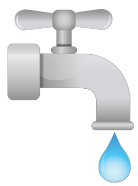
3.2 billion people with Internet access (2015)*

*Source: ITU and Cisco

Why do we need yet another protocol?

Internet connectivity is a critical service!

3.2 billion people with Internet access (2015)*



3 billion people with running water (2015)*

*Source: ITU and Cisco

Why do we need yet another protocol?

Internet connectivity is a critical service!

3.2 billion people with Internet access (2015)*

3 billion people with running water (2015)*

New techniques for more reliable and performant networks

*Source: ITU and Cisco

Why do we need yet another protocol?

Internet connectivity is a critical service!

3.2 billion people with Internet access (2015)*

3 billion people with running water (2015)*

New techniques for more reliable and performant networks

WebRTC

Quick
UDP
Internet
Connection

HTTPSpeed+Mobility

HTTP/2

SPDY

*Source: ITU and Cisco

Why do we need yet another protocol?

Internet connectivity is a critical service!

3.2 billion people with Internet access (2015)*

3 billion people with running water (2015)*

New techniques for more reliable and performant networks

WebRTC

Quick
UDP
Internet
Connection

HTTPSpeed+Mobility

HTTP/2

SPDY

*Source: ITU and Cisco

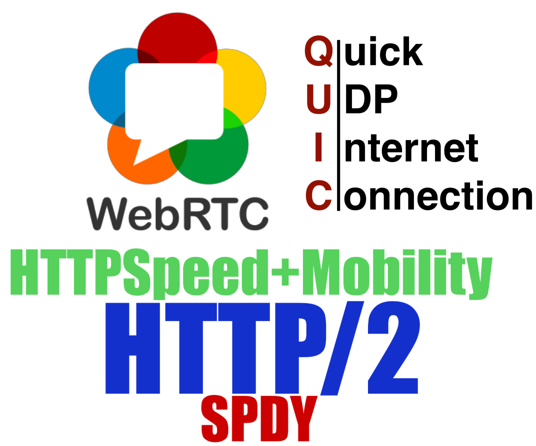
Why do we need yet another protocol?

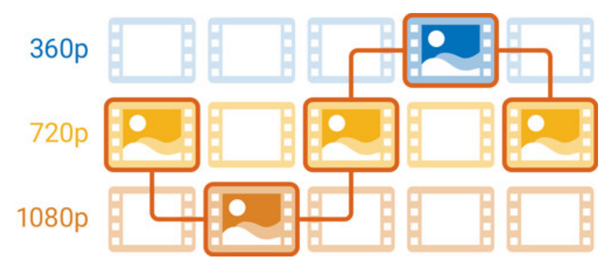
Internet connectivity is a critical service!

3.2 billion people with Internet access (2015)*

3 billion people with running water (2015)*

New techniques for more reliable and performant networks





*Source: ITU and Cisco

Why QUIC?

Quick UDP Internet Connection

Why QUIC?

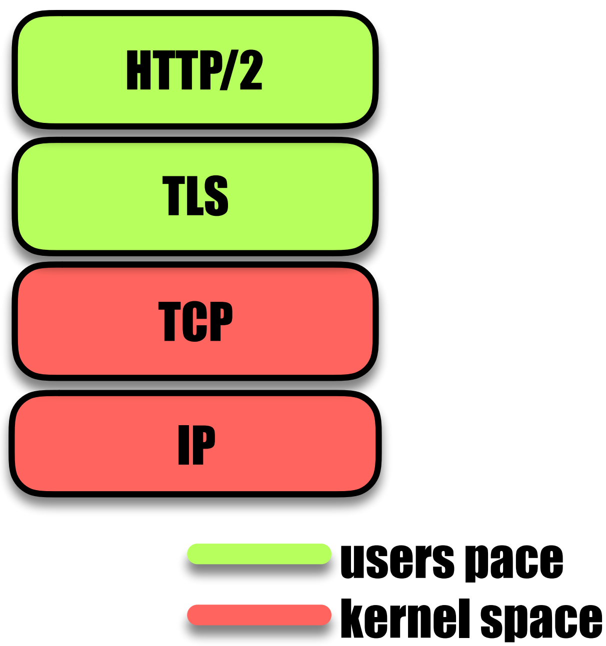
Quick UDP Internet Connection

1. Facilitate rapid deployment

Why QUIC?

Quick UDP Internet Connection

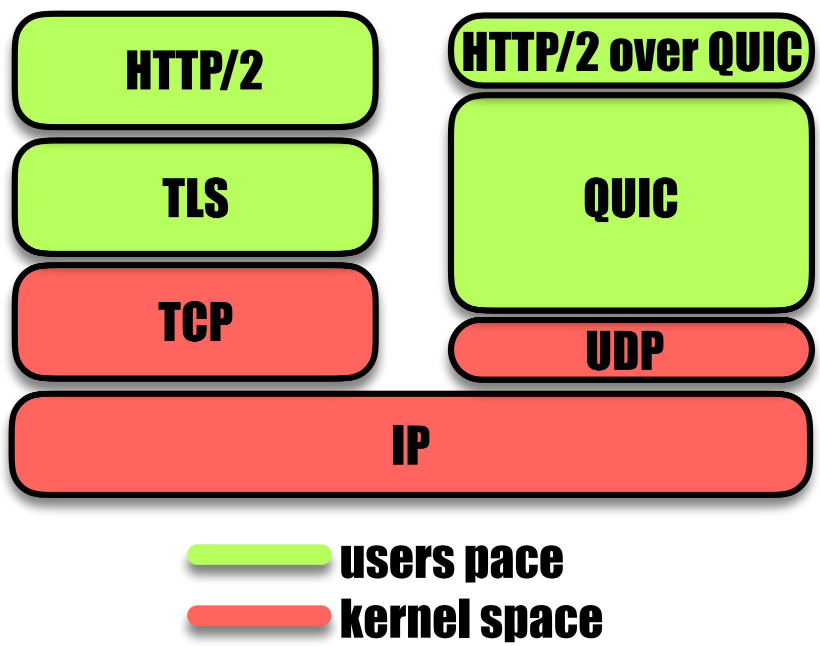
1. Facilitate rapid deployment



Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment



Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes

Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes

Joint study by Google and T-Mobile on YouTube's performance over T-Mobile's network (Velocity Conference 2014)

Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes

Joint study by Google and T-Mobile on YouTube's performance over T-Mobile's network (Velocity Conference 2014)

Summary Findings from Proxy Bypass

Bypassing proxies

- Lowers retransmission rates, increases throughput, and decreases bufferbloat
- Does not negatively impact network traffic
- Improves quality of experience for video
- Increases battery lifetime

Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes

> "Cloudflare … support TLS 1.3 by default on the server side. We expected the client side would follow suit … It has been over a year … and still, none of the major browsers have enabled TLS 1.3 by default.
> The reductive answer to why TLS 1.3 hasn't been deployed yet is ***middleboxes***."
>
> *Cloudflare blog, 2017*
> https://blog.cloudflare.com/why-tls-1-3-isnt-in-browsers-yet

Why QUIC?

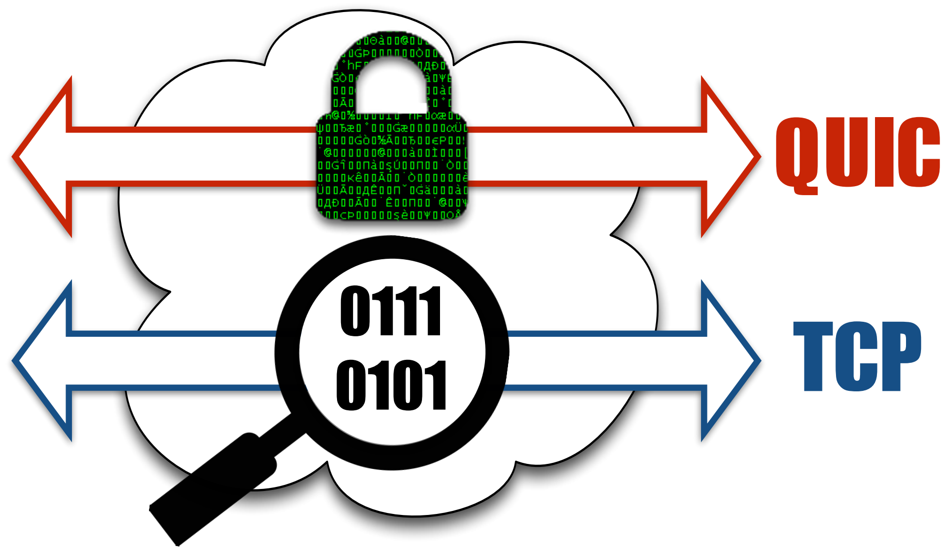
Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes

Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes



Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic

Why QUIC?

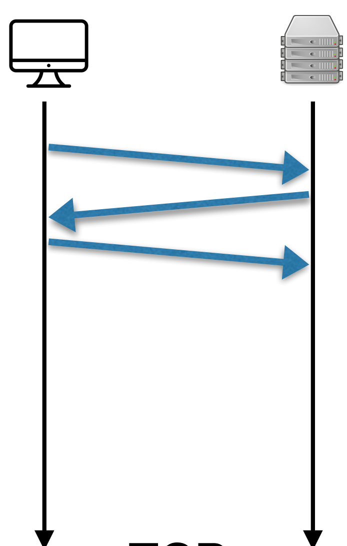
Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
    - Reduce handshake time (0-RTT)

Why QUIC?

Quick UDP Internet Connection

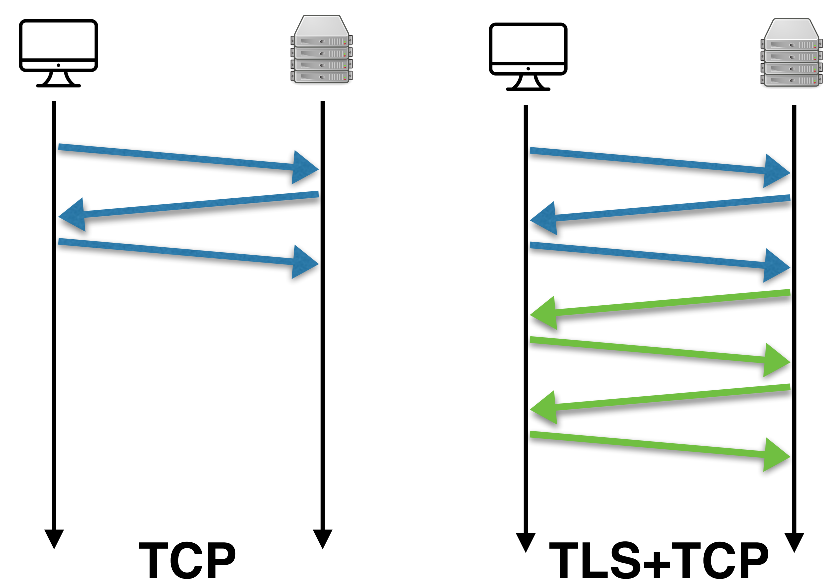
1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
   - Reduce handshake time (0-RTT)



Why QUIC?

Quick UDP Internet Connection

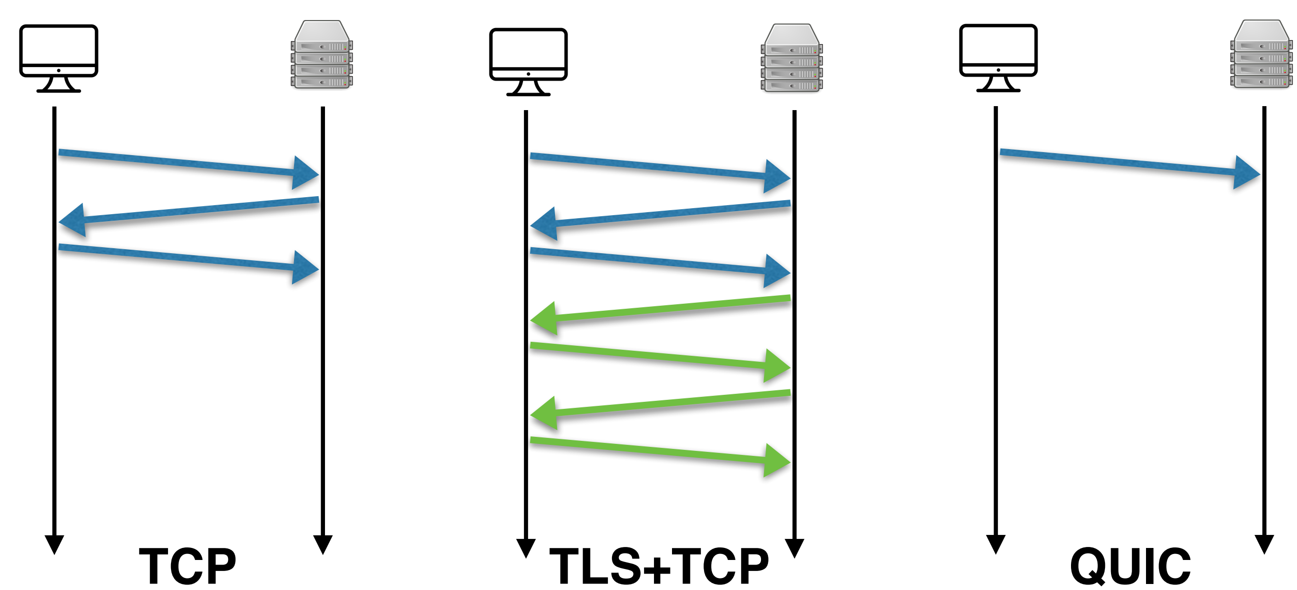
1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
   - Reduce handshake time (0-RTT)



Why QUIC?

Quick UDP Internet Connection

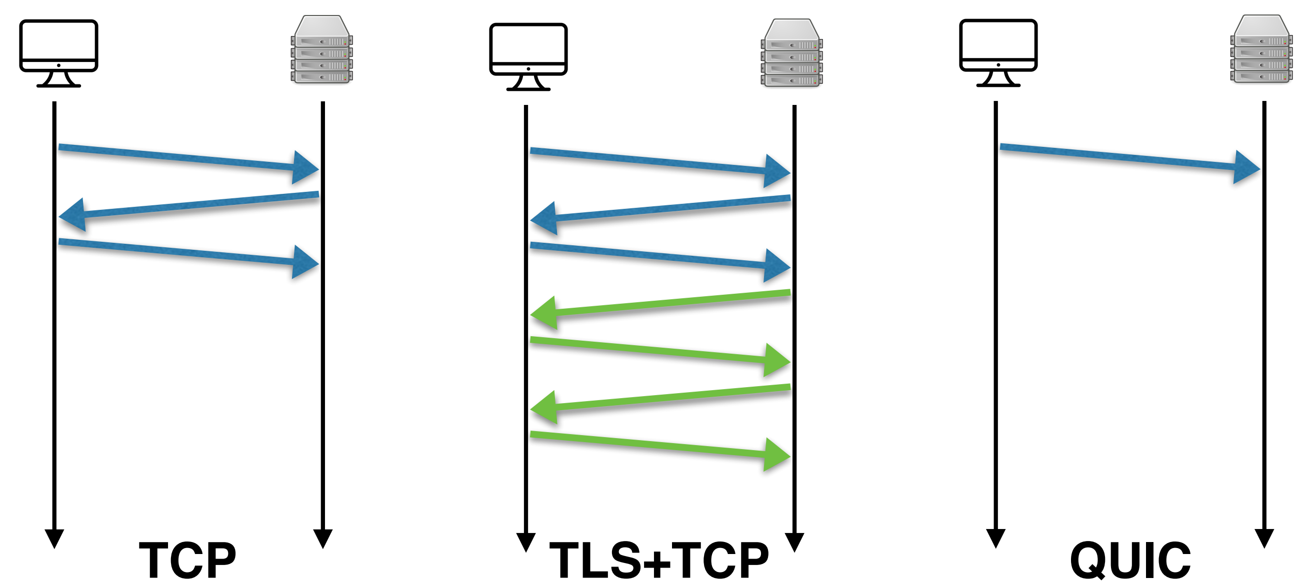
1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
    - Reduce handshake time (0-RTT)



Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
   - Reduce handshake time (0-RTT)



QUIC connection RTTs:

- 0-RTT: if keys are valid
- 1-RTT: if keys are old
- 2-RTT: if first time

Why QUIC?

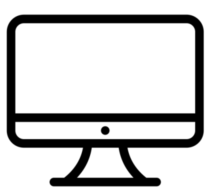
Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
    - Reduce handshake time (0-RTT)
    - Prevent head-of-line blocking

Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
    - Reduce handshake time (0-RTT)
    - Prevent head-of-line blocking



Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
    - Reduce handshake time (0-RTT)
    - Prevent head-of-line blocking

Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
    - Reduce handshake time (0-RTT)
    - Prevent head-of-line blocking

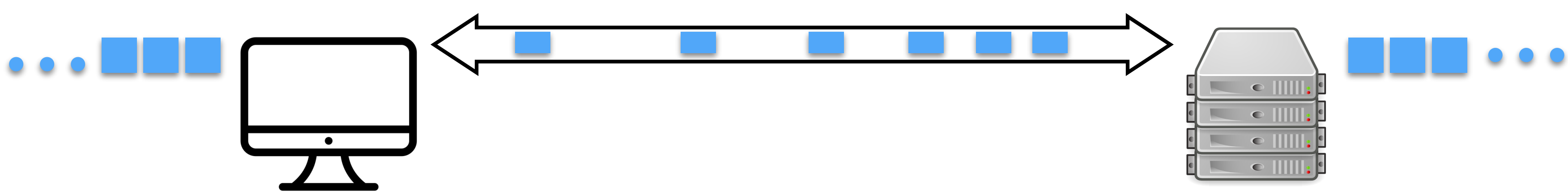
HTTP/1.1 over TCP

Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
   - Reduce handshake time (0-RTT)
   - Prevent head-of-line blocking

HTTP/1.1 over TCP

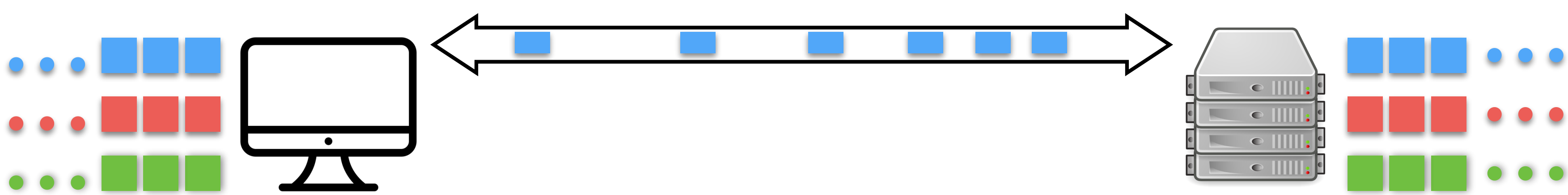


Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
    - Reduce handshake time (0-RTT)
    - Prevent head-of-line blocking

HTTP/1.1 over TCP

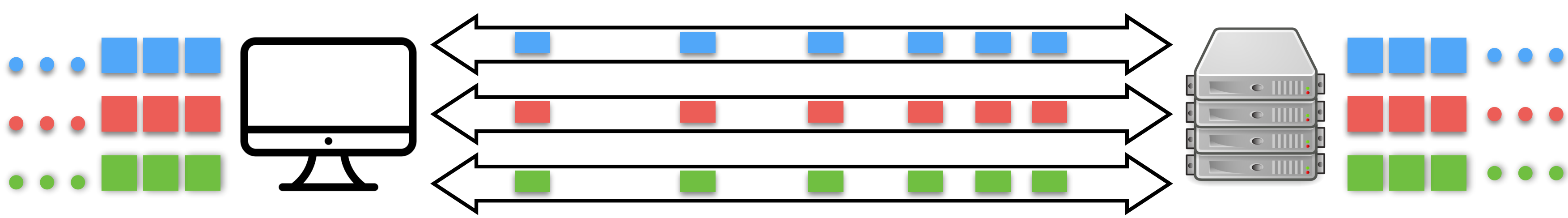


Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
   - Reduce handshake time (0-RTT)
   - Prevent head-of-line blocking

HTTP/1.1 over TCP

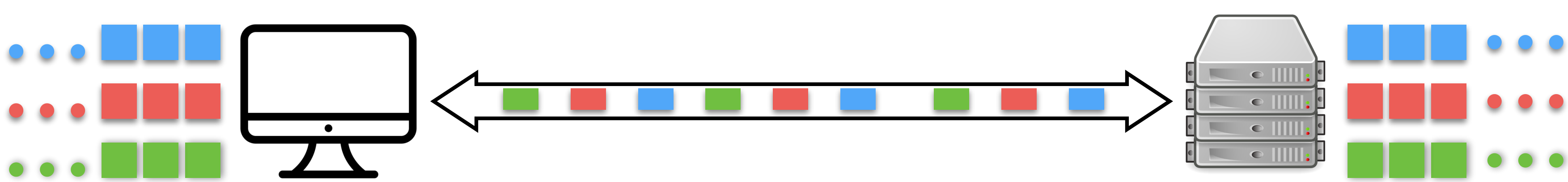


Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
    - Reduce handshake time (0-RTT)
    - Prevent head-of-line blocking

HTTP/2 over TCP

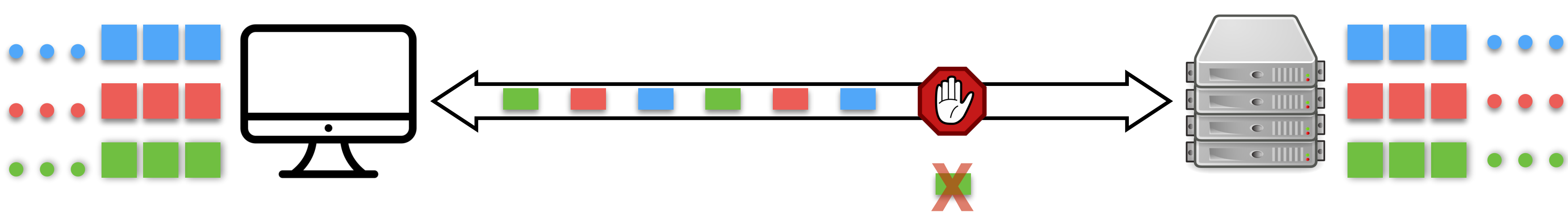


Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
    - Reduce handshake time (0-RTT)
    - Prevent head-of-line blocking

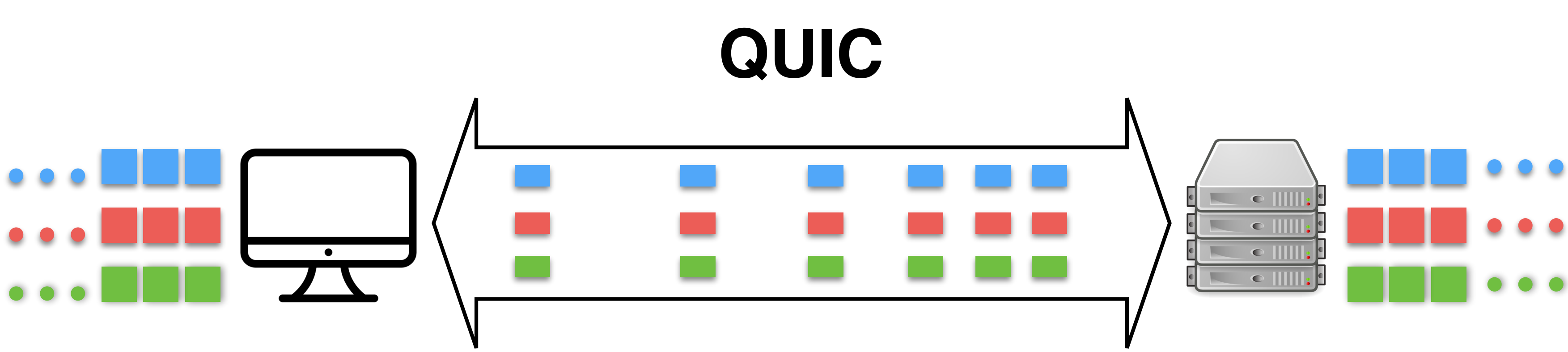
HTTP/2 over TCP



Why QUIC?

Quick UDP Internet Connection

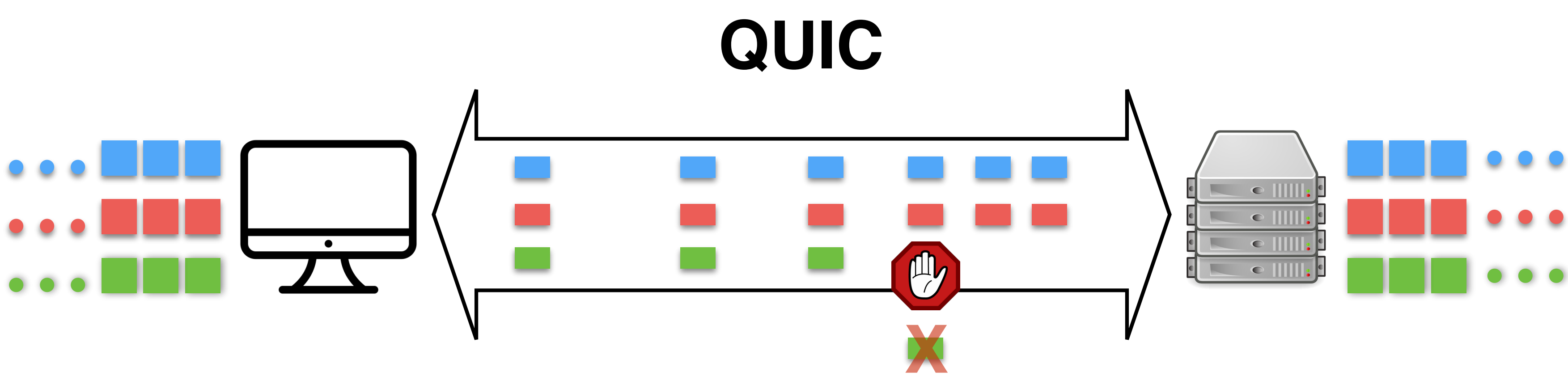
1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
   - Reduce handshake time (0-RTT)
   - Prevent head-of-line blocking



Why QUIC?

Quick UDP Internet Connection

1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
    - Reduce handshake time (0-RTT)
    - Prevent head-of-line blocking



Why QUIC?

Quick UDP Internet Connection

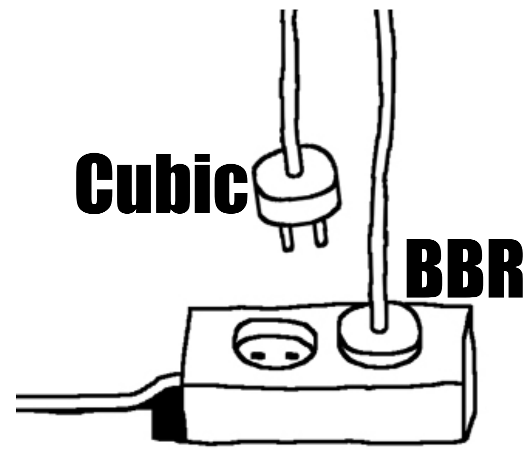
1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
    - Reduce handshake time (0-RTT)
    - Prevent head-of-line blocking
    - Improve loss recovery

- ✤ No ACK ambiguity
- ✤ Better RTT/BW estimation

Why QUIC?

Quick UDP Internet Connection

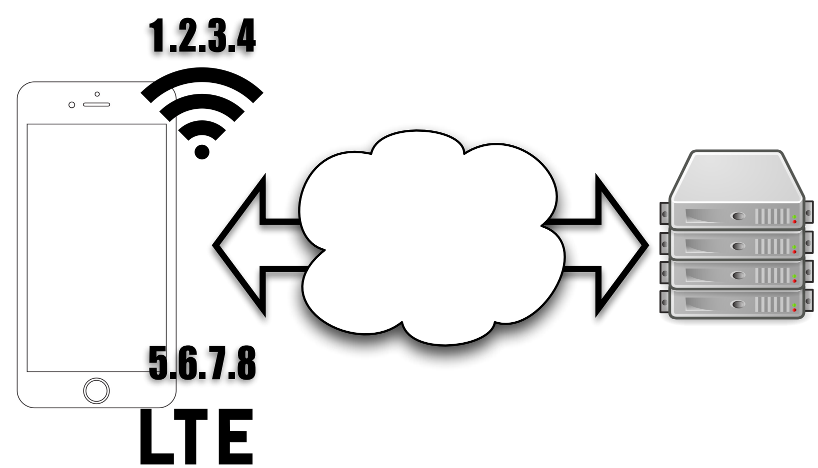
1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
    - Reduce handshake time (0-RTT)
    - Prevent head-of-line blocking
    - Improve loss recovery
    - Modular congestion control



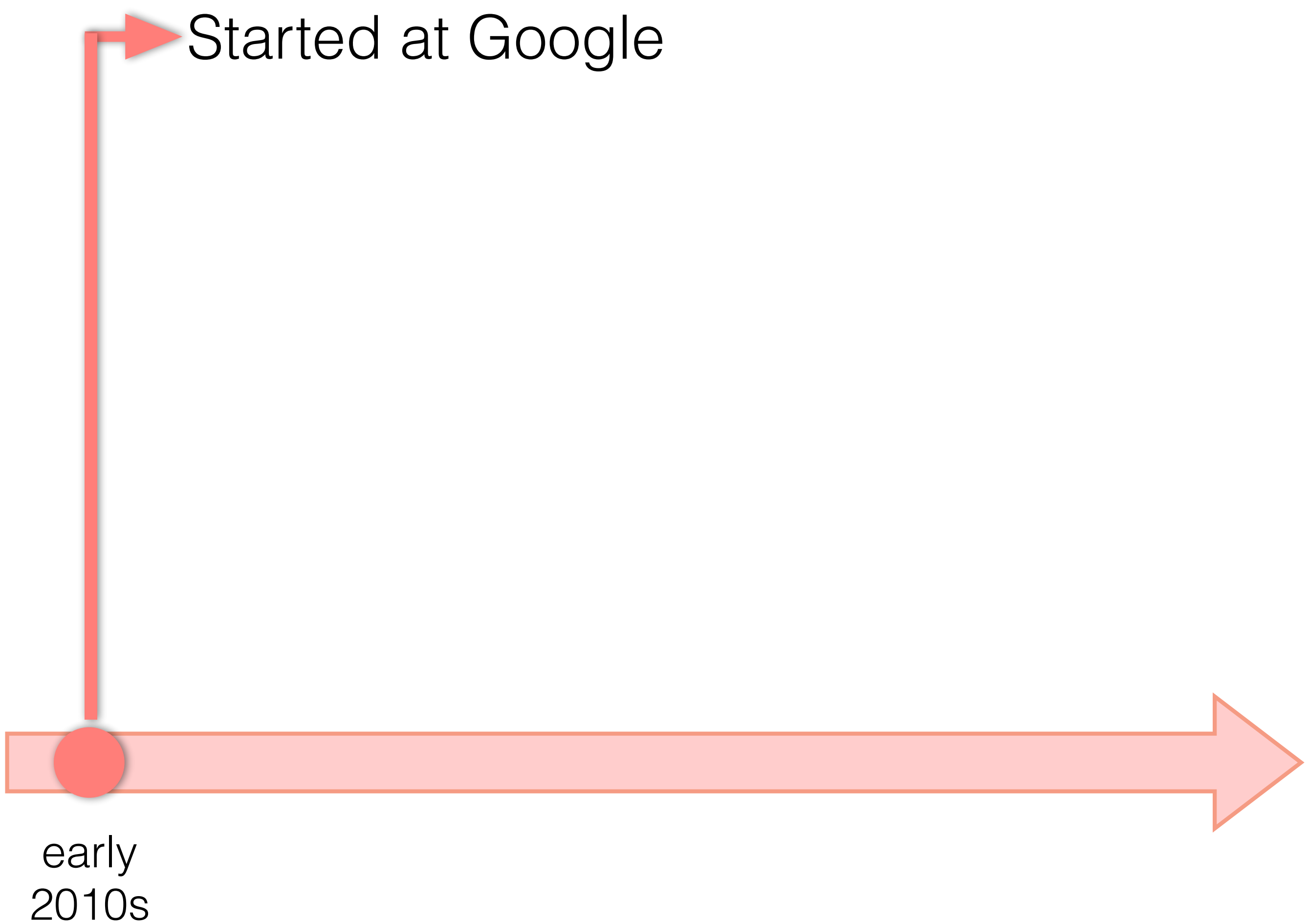
Why QUIC?

Quick UDP Internet Connection

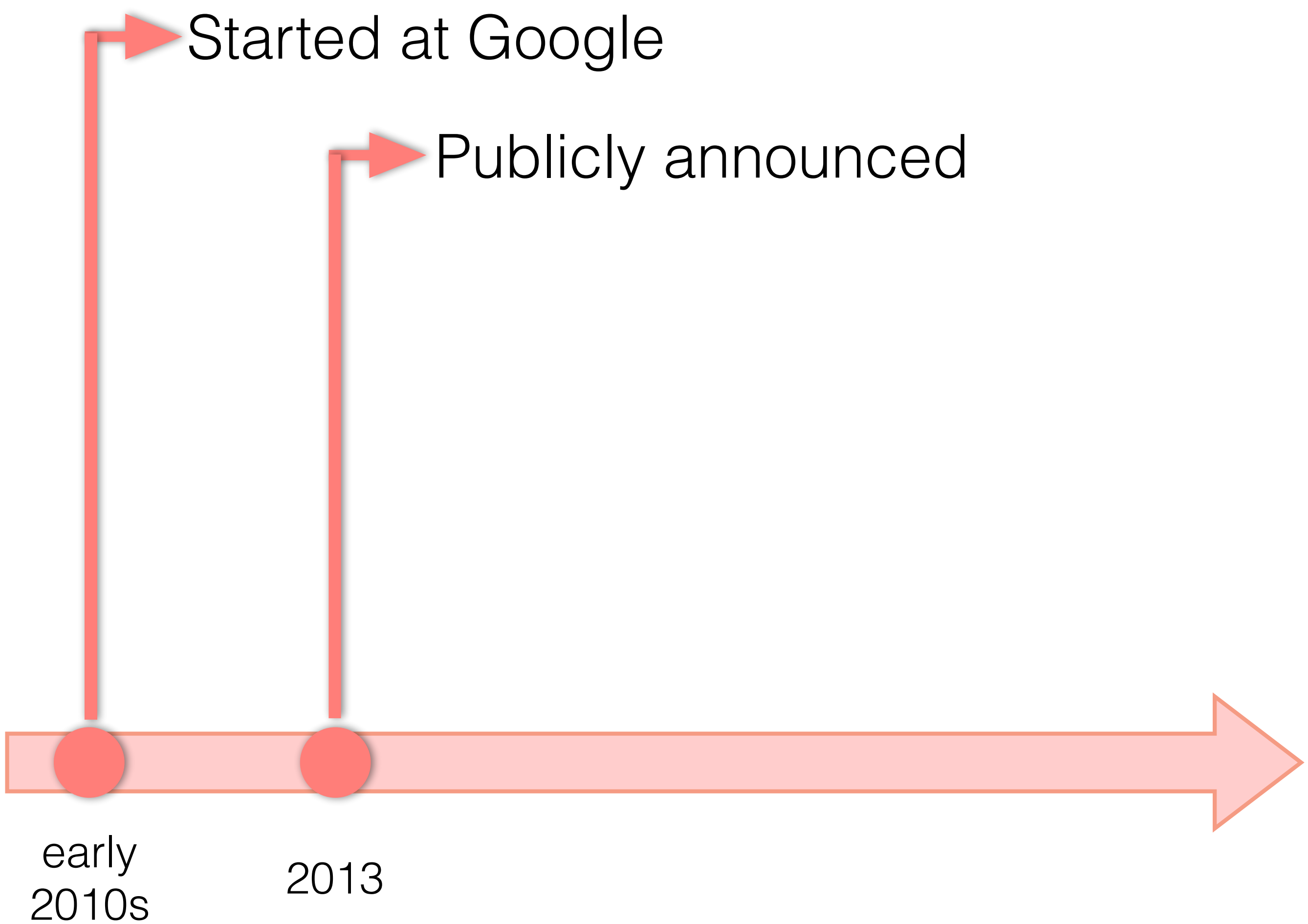
1. Facilitate rapid deployment
2. Avoid ossification and meddling by middleboxes
3. Improve performance for HTTP traffic
   - Reduce handshake time (0-RTT)
   - Prevent head-of-line blocking
   - Improve loss recovery
   - Modular congestion control
   - Connection migration across IPs



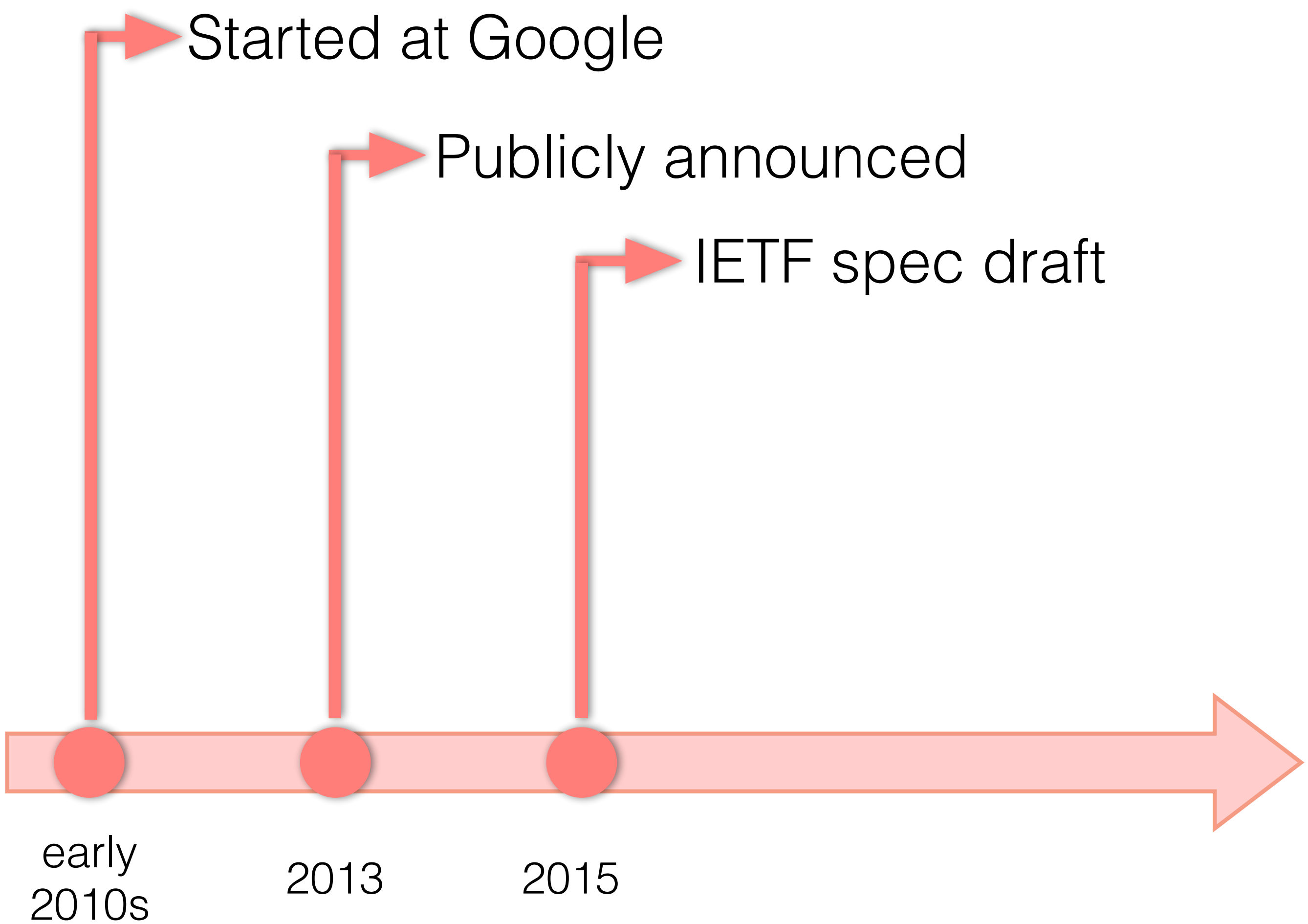
QUIC's Timeline



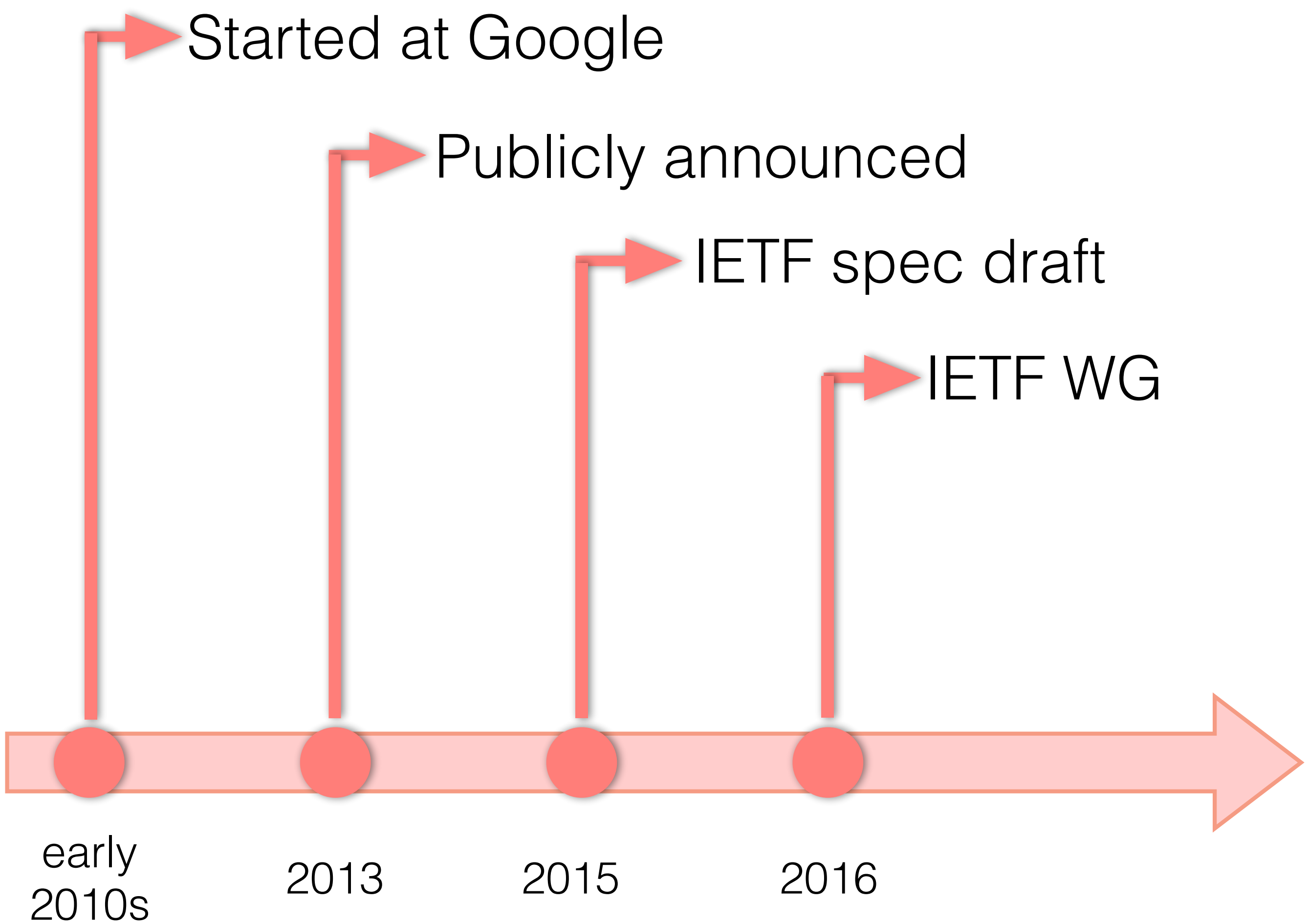
QUIC’s Timeline



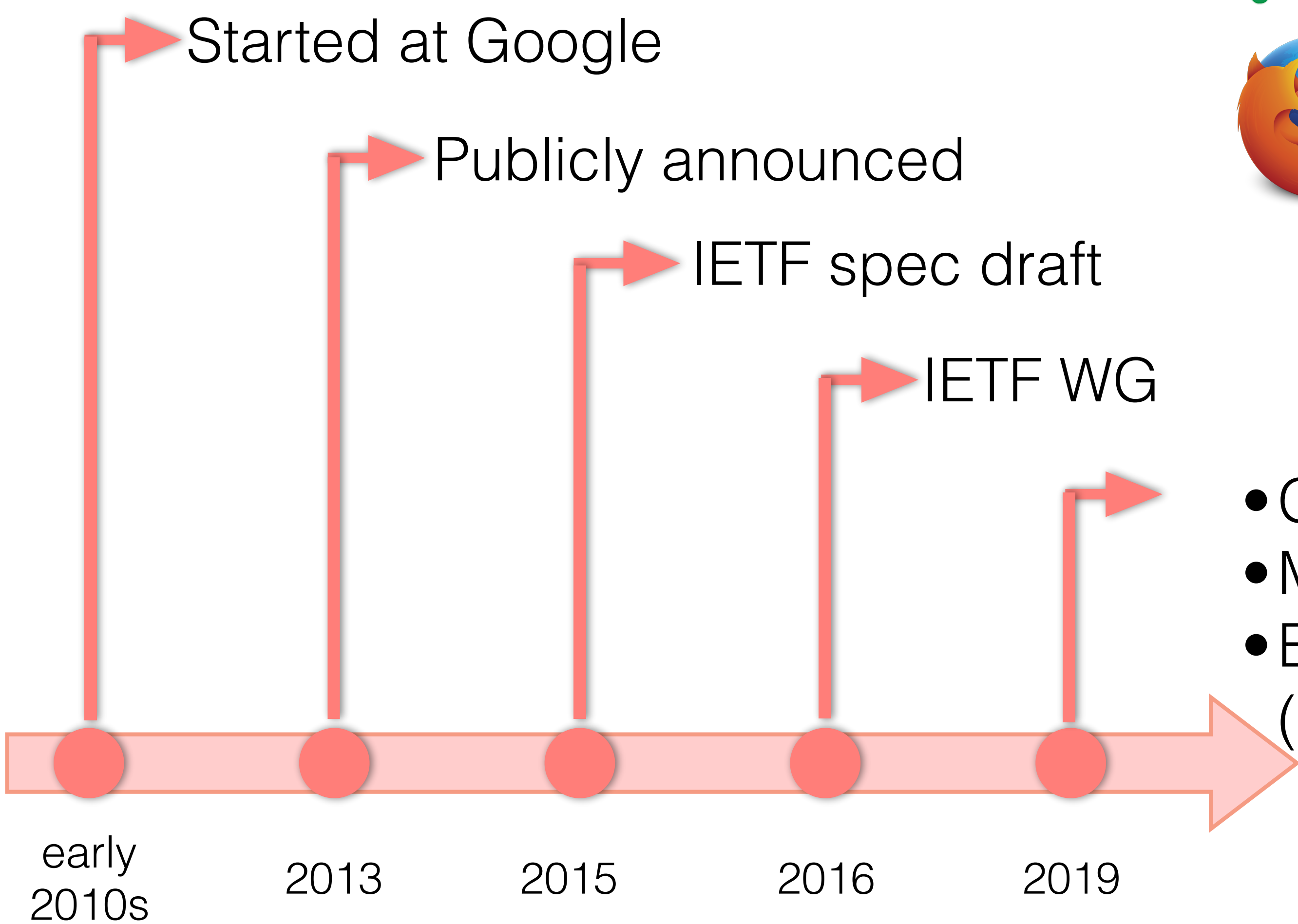
QUIC's Timeline



QUIC's Timeline



QUIC's Timeline



- Google QUIC v47
- Many implementations
- Enroute to standardization (IETF QUIC)

QUIC's Performance Reports

*Published at SIGCOMM 2017

QUIC's Performance Reports

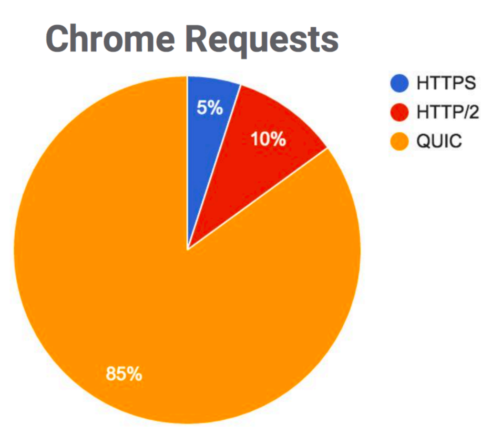
Google's reports*

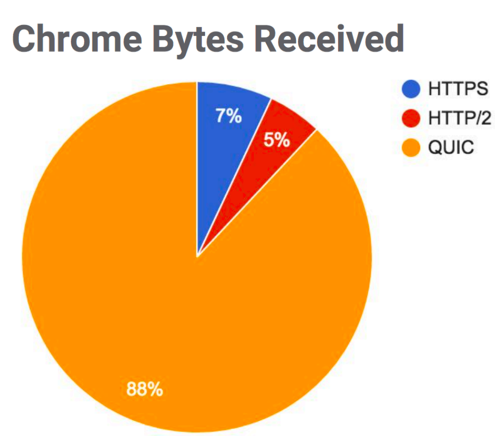
*Published at SIGCOMM 2017

QUIC’s Performance Reports

Google’s reports*

- >35% of Google’s egress traffic (>7% of Internet traffic)



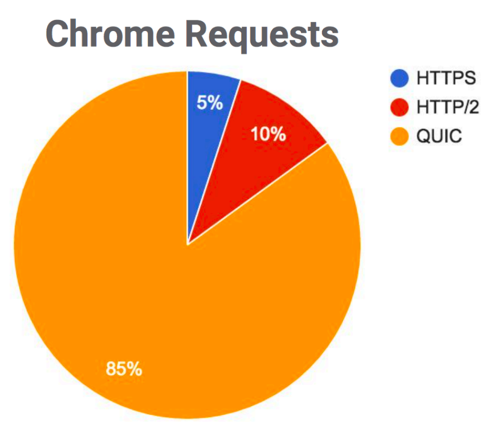


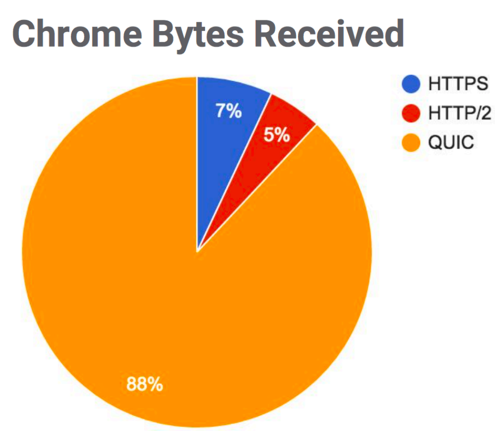
*Published at SIGCOMM 2017

QUIC’s Performance Reports

Google’s reports*

- >35% of Google’s egress traffic (>7% of Internet traffic)
- 3% PLT improvement on Google search



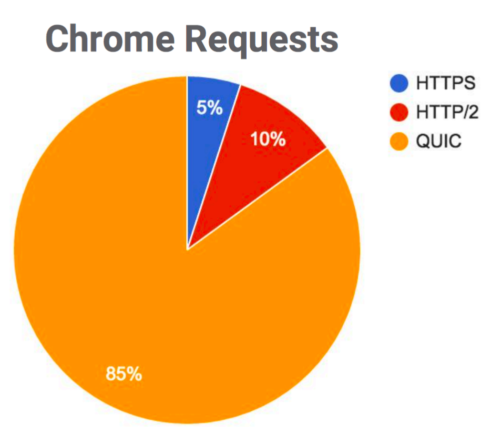


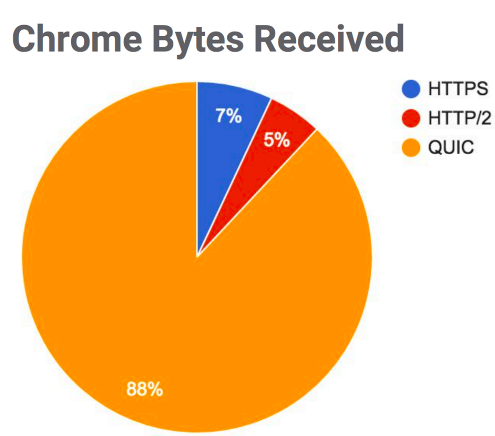
*Published at SIGCOMM 2017

QUIC’s Performance Reports

Google’s reports*

- >35% of Google’s egress traffic (>7% of Internet traffic)
- 3% PLT improvement on Google search
- Up to 8% reduced latency on Google search



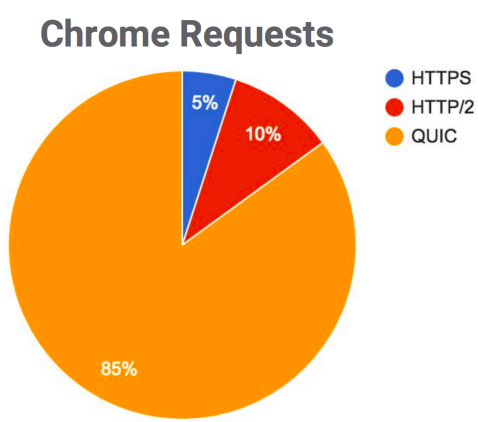


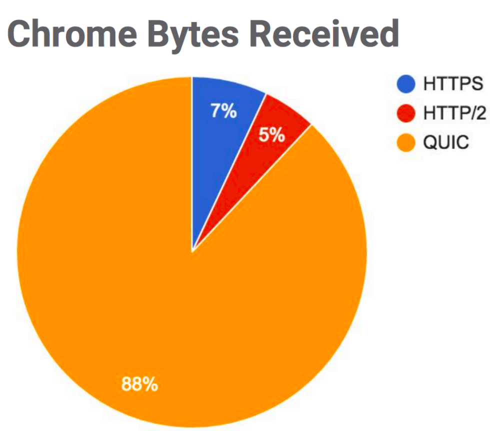
*Published at SIGCOMM 2017

QUIC's Performance Reports

Google's reports*

- >35% of Google's egress traffic (>7% of Internet traffic)
- 3% PLT improvement on Google search
- Up to 8% reduced latency on Google search
- Up to 18% reduced buffer time on YouTube





*Published at SIGCOMM 2017

QUIC's Performance Reports shortcomings

Google's reports

QUIC's Performance Reports shortcomings

Google's reports

- Aggregated statistics

QUIC's Performance Reports shortcomings

Google's reports

- Aggregated statistics
- Not reproducible

QUIC's Performance Reports shortcomings

Google's reports

- Aggregated statistics
- Not reproducible
- Limited controlled tests

QUIC's Performance Reports shortcomings

Google's reports

- Aggregated statistics
- Not reproducible
- Limited controlled tests

QUIC’s Performance Reports shortcomings

Google’s reports

- Aggregated statistics
- Not reproducible
- Limited controlled tests

Other QUIC evaluations

QUIC's Performance Reports shortcomings

Google's reports

- Aggregated statistics
- Not reproducible
- Limited controlled tests

Other QUIC evaluations

- Limited environments/networks

QUIC's Performance Reports shortcomings

Google's reports

- Aggregated statistics
- Not reproducible
- Limited controlled tests

Other QUIC evaluations

- Limited environments/networks
- Limited tests

QUIC's Performance Reports shortcomings

Google's reports

- Aggregated statistics
- Not reproducible
- Limited controlled tests

Other QUIC evaluations

- Limited environments/networks
- Limited tests
- One *old untuned version* of QUIC

QUIC’s Performance Reports shortcomings

Google’s reports

- Aggregated statistics
- Not reproducible
- Limited controlled tests

Other QUIC evaluations

- Limited environments/networks
- Limited tests
- One *old untuned version* of QUIC
- Results not statically sound

QUIC's Performance Reports shortcomings

Google's reports

- Aggregated statistics
- Not reproducible
- Limited controlled tests

Other QUIC evaluations

- Limited environments/networks
- Limited tests
- One *old untuned version* of QUIC
- Results not statically sound
- No root cause analysis

QUIC's Performance Reports shortcomings

Google's reports

- Aggregated statistics
- Not reproducible
- Limited controlled tests

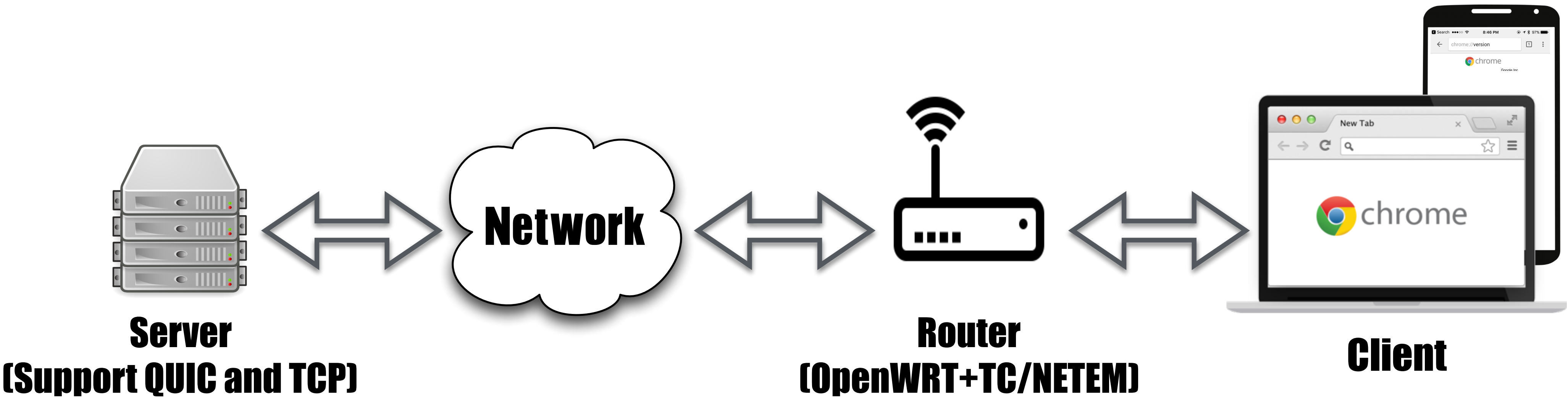
Other QUIC evaluations

- Limited environments/networks
- Limited tests
- One *old untuned version* of QUIC
- Results not statically sound
- No root cause analysis

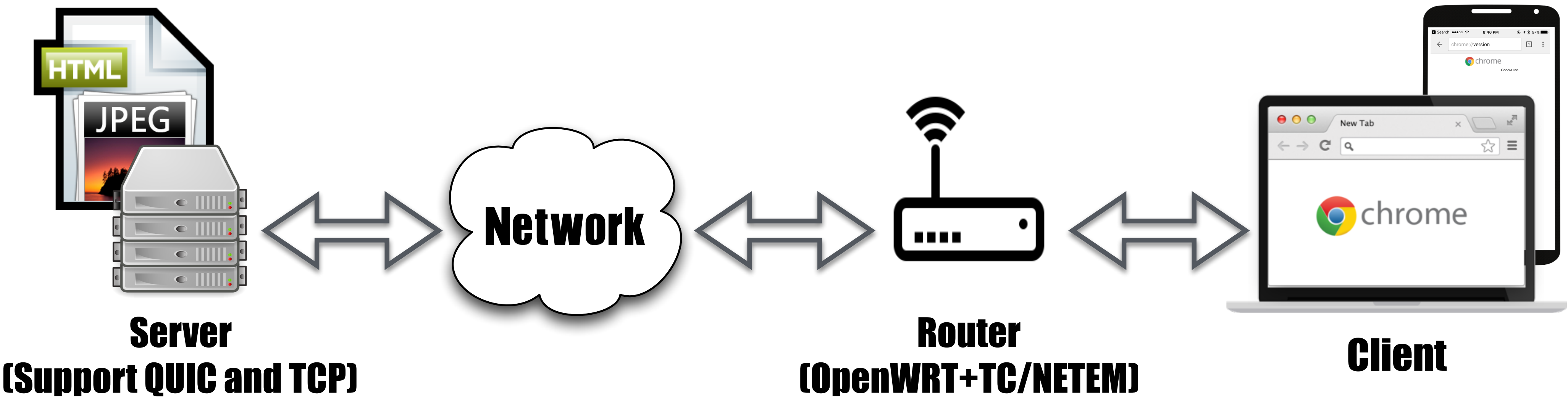
Our goal: provide a rigorous evaluation of QUIC and how it compares to TCP

HTTP Performance: QUIC vs. TCP

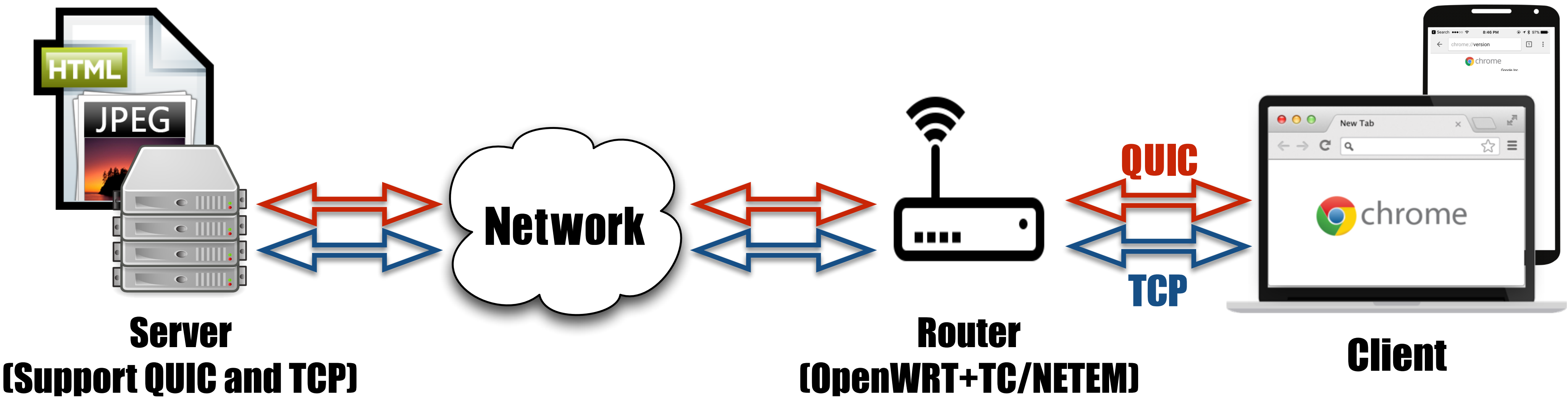
HTTP Performance: QUIC vs. TCP



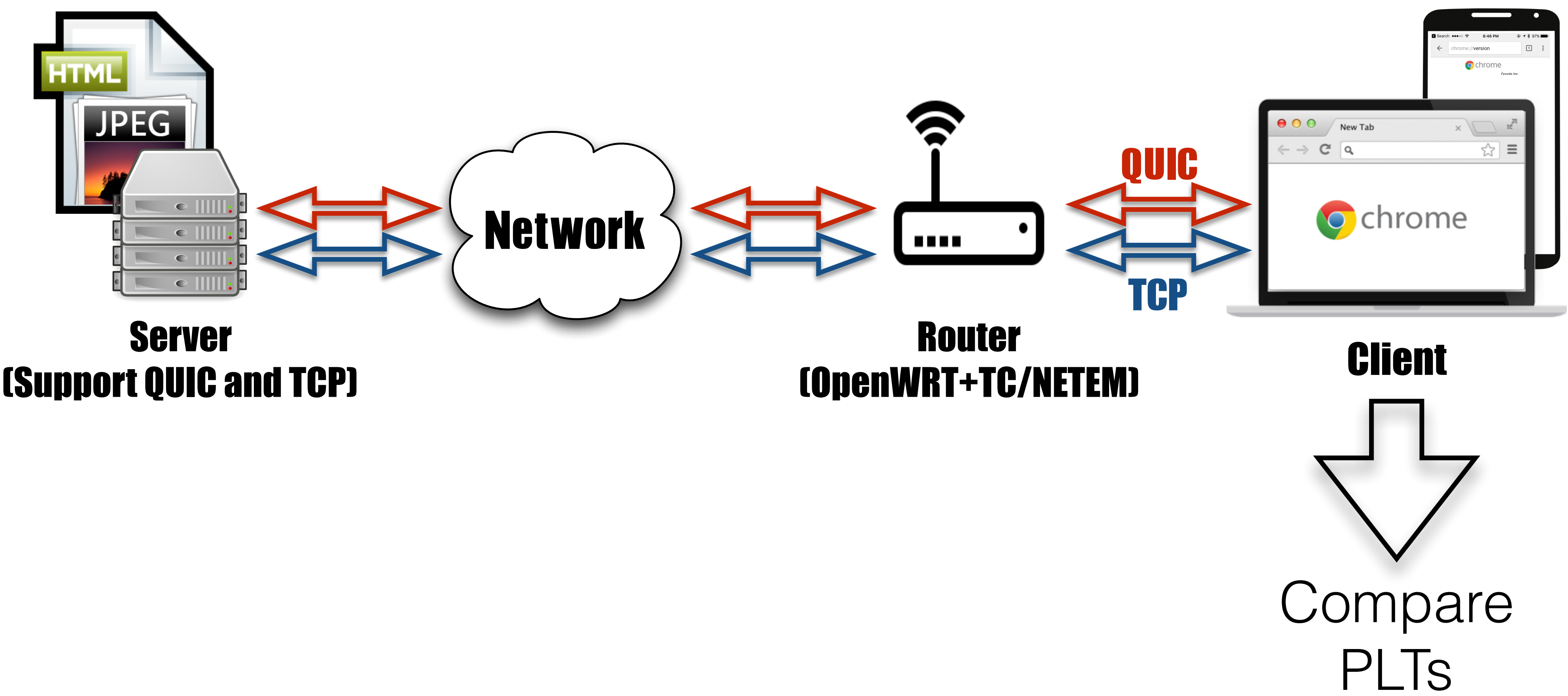
HTTP Performance: QUIC vs. TCP



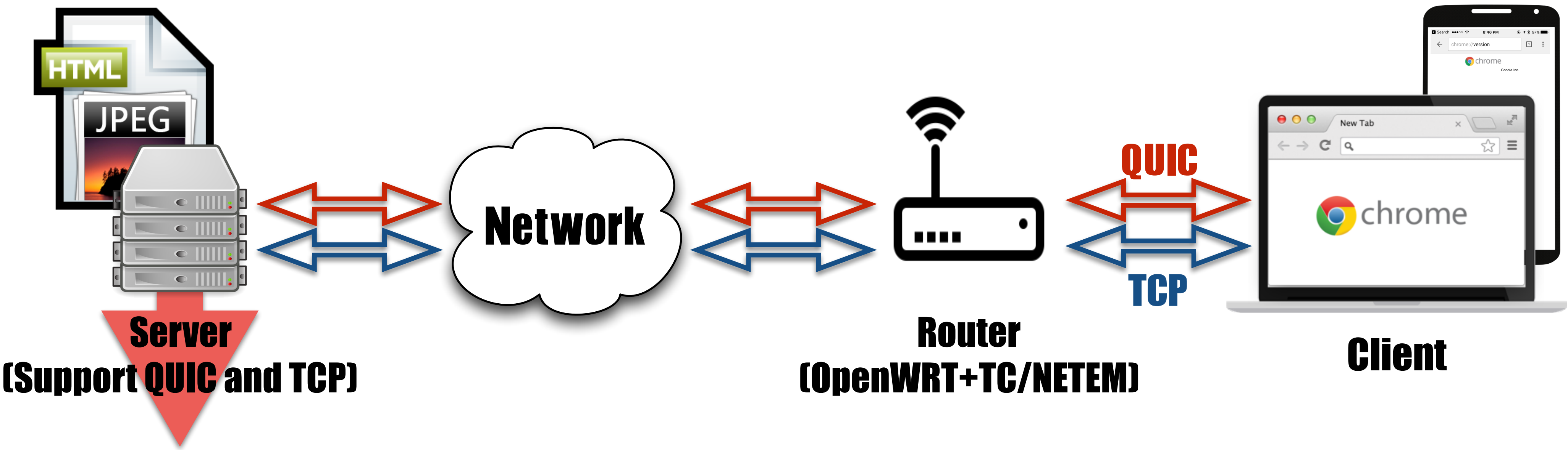
HTTP Performance: QUIC vs. TCP



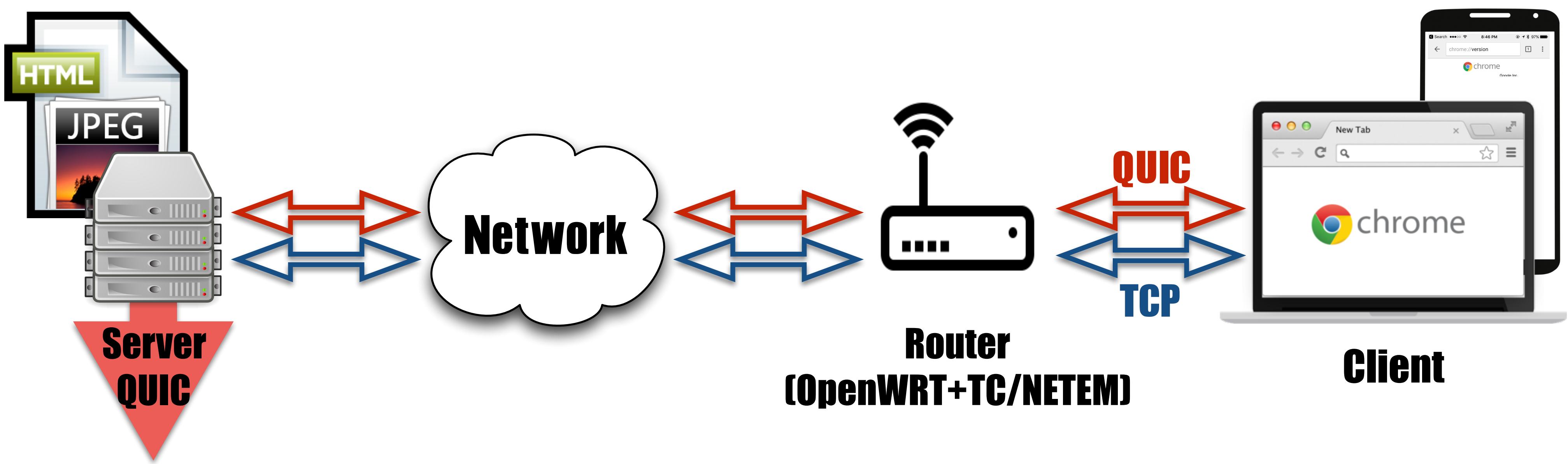
HTTP Performance: QUIC vs. TCP



HTTP Performance: QUIC vs. TCP

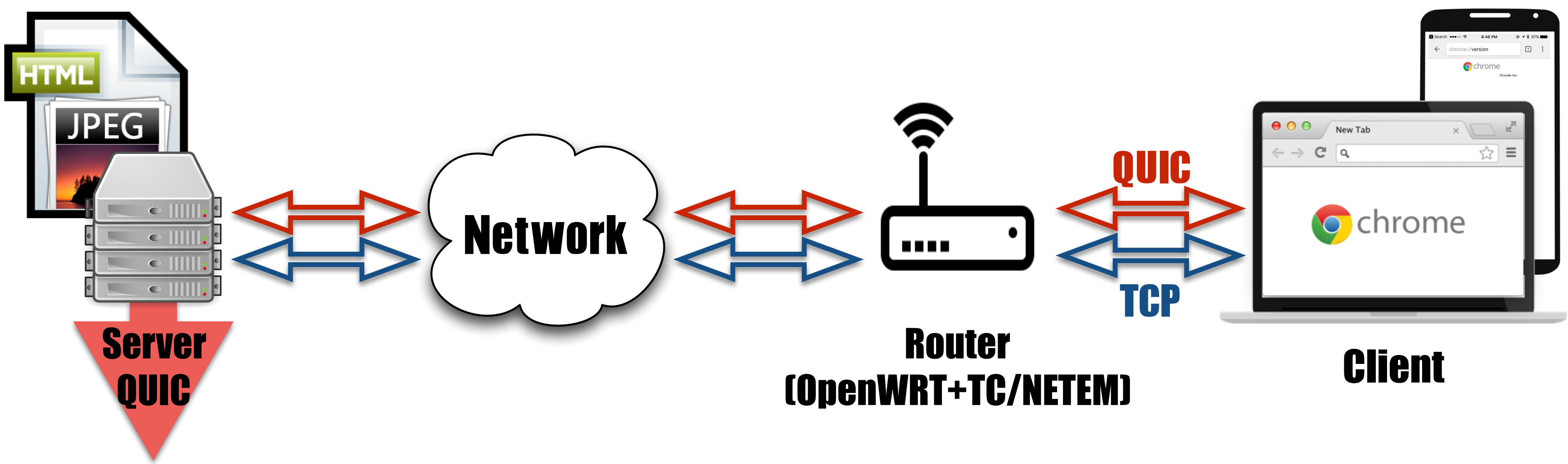


HTTP Performance: QUIC vs. TCP



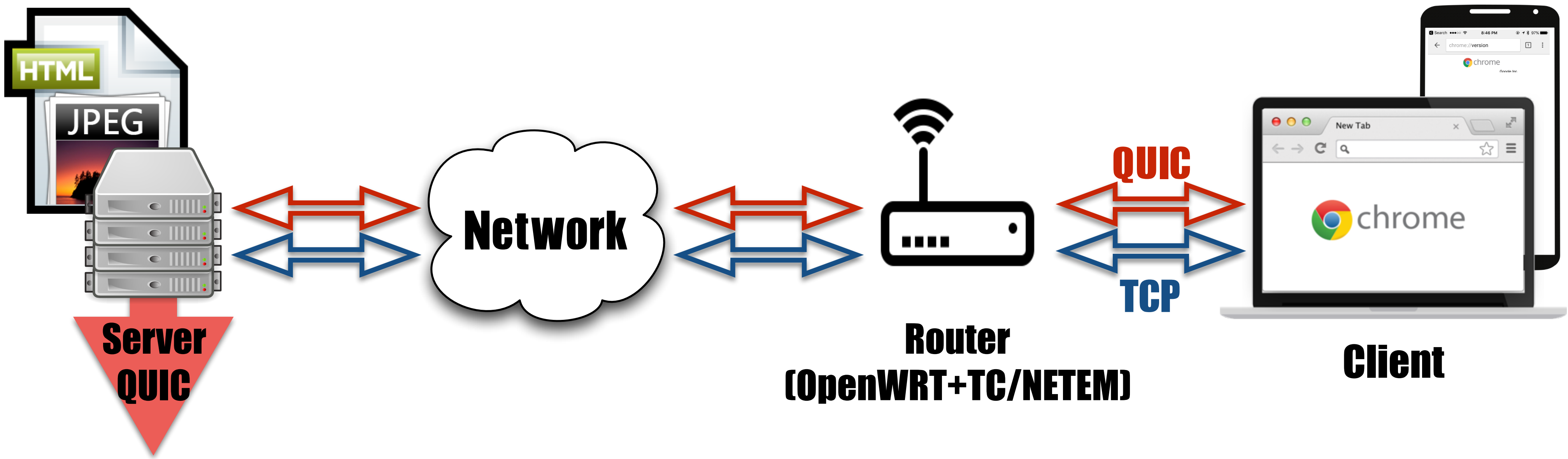
1- Google servers

HTTP Performance: QUIC vs. TCP



1- Google servers

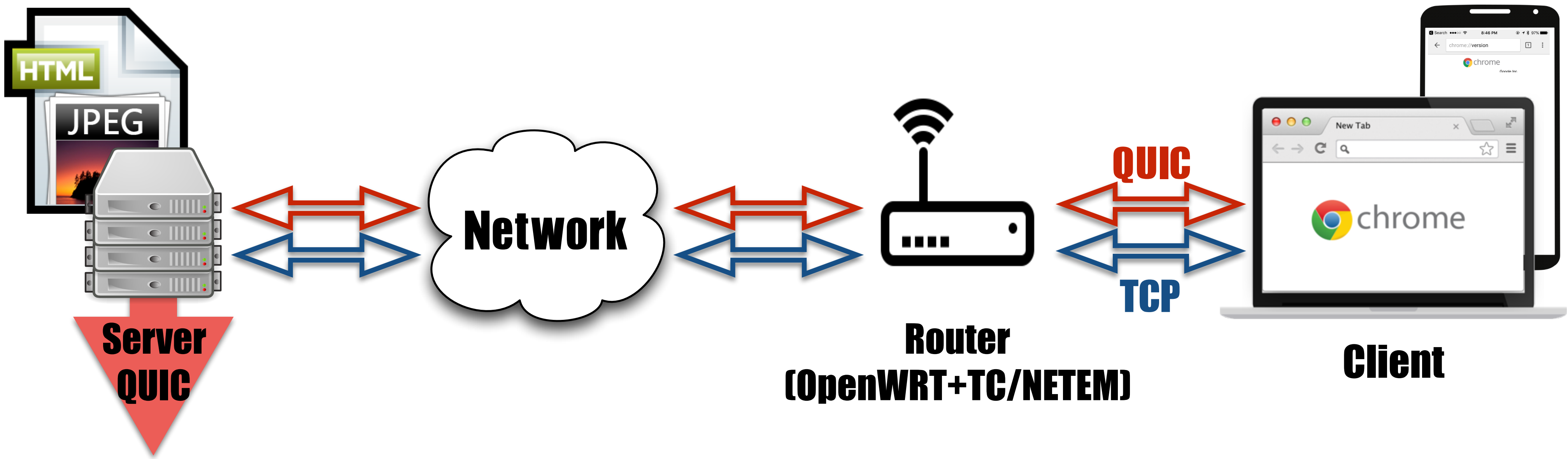
HTTP Performance: QUIC vs. TCP



1- Google servers

2- Server in Chromium

HTTP Performance: QUIC vs. TCP

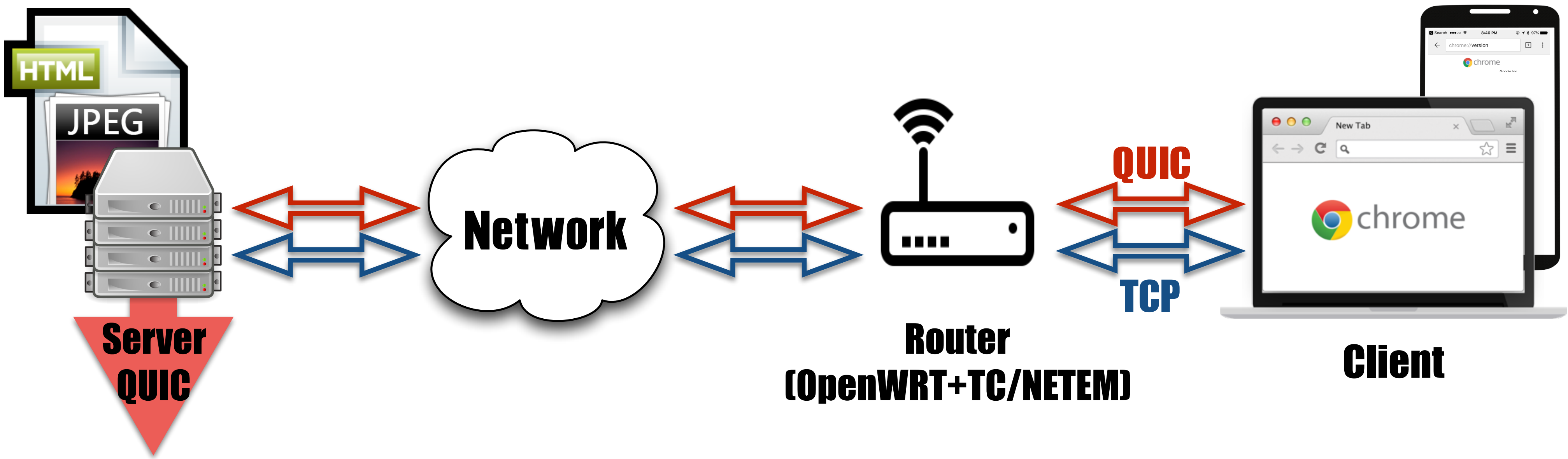


1- Google servers

- No control

2- Server in Chromium

HTTP Performance: QUIC vs. TCP

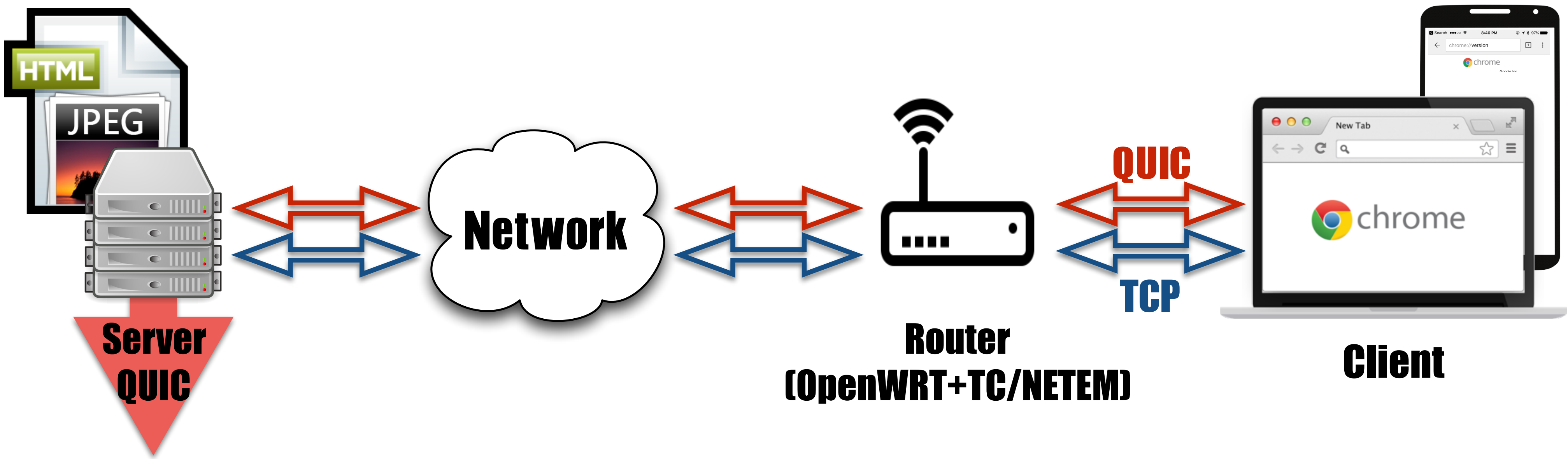


1- Google servers

- No control
- Unexpected behavior

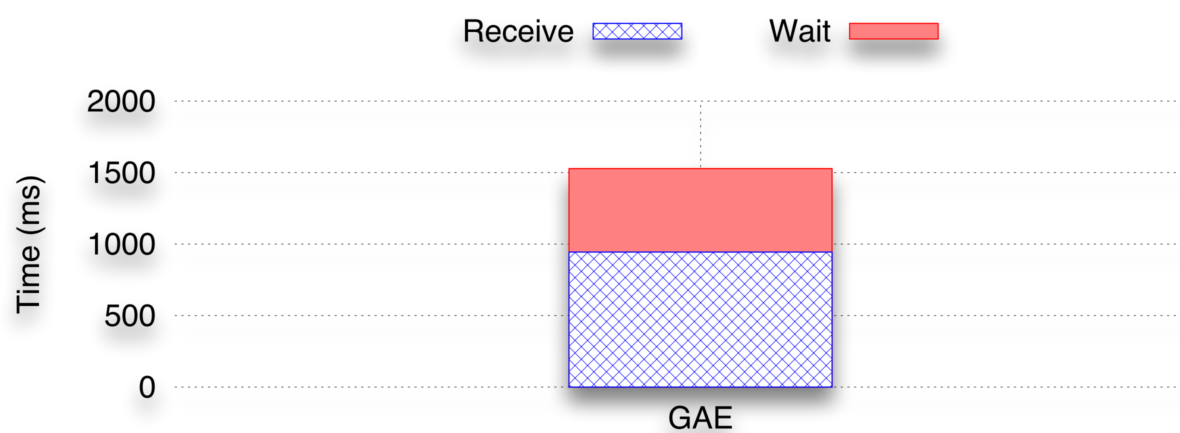
2- Server in Chromium

HTTP Performance: QUIC vs. TCP



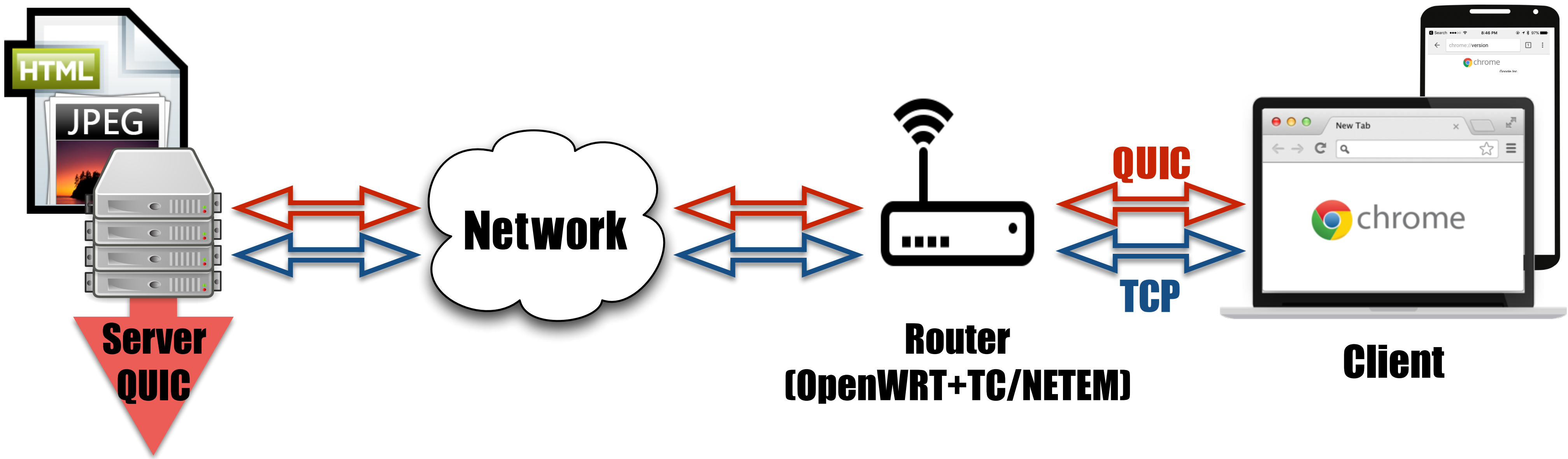
1- Google servers
- No control
- Unexpected behavior

2- Server in Chromium



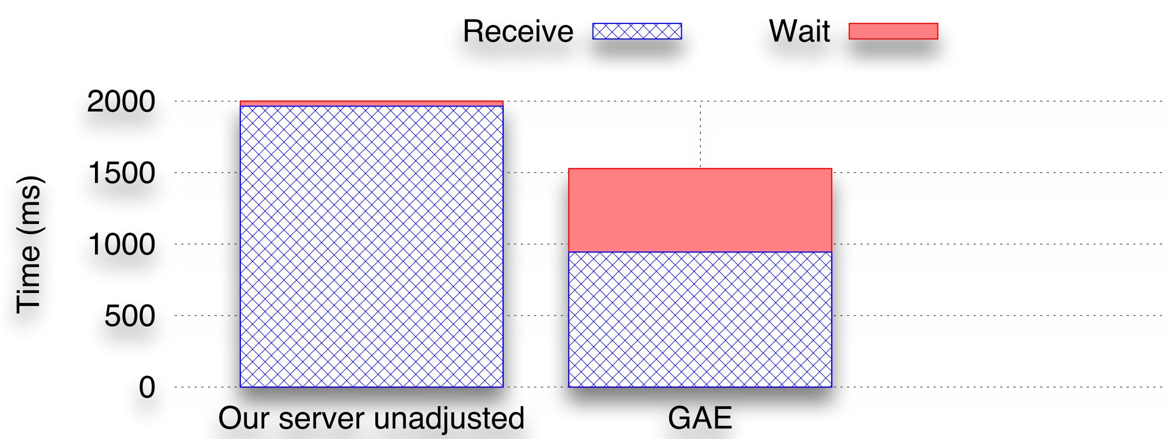
Downloading a 10MB obj at 100Mbps

HTTP Performance: QUIC vs. TCP



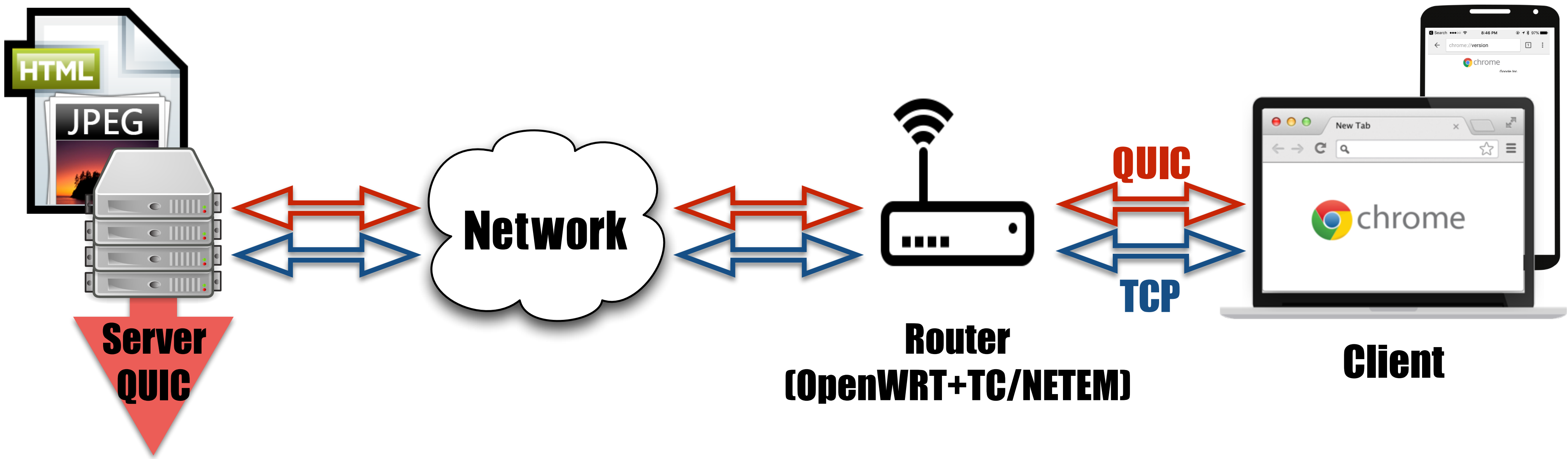
1- Google servers
- No control
- Unexpected behavior

2- Server in Chromium
- Not performant



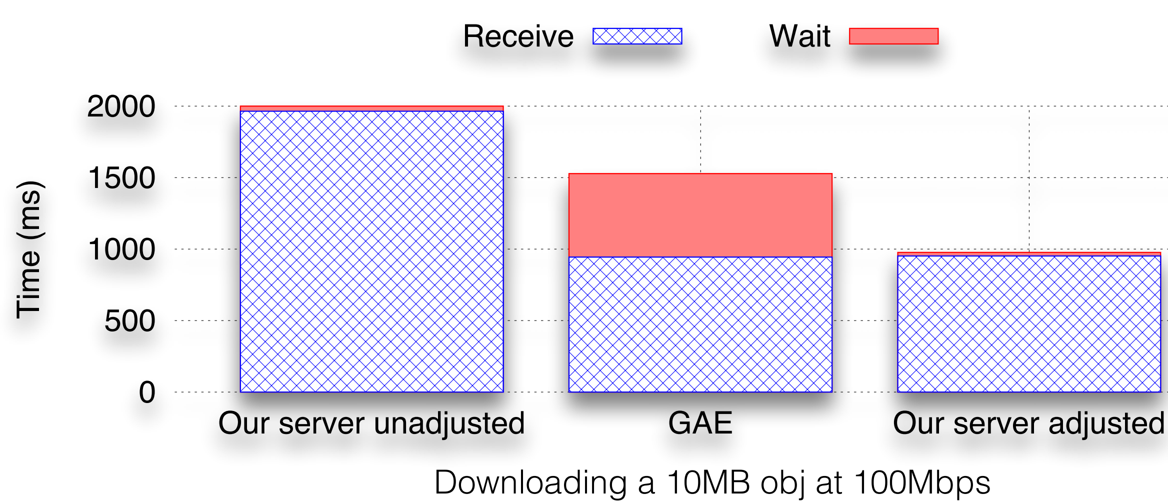
Downloading a 10MB obj at 100Mbps

HTTP Performance: QUIC vs. TCP

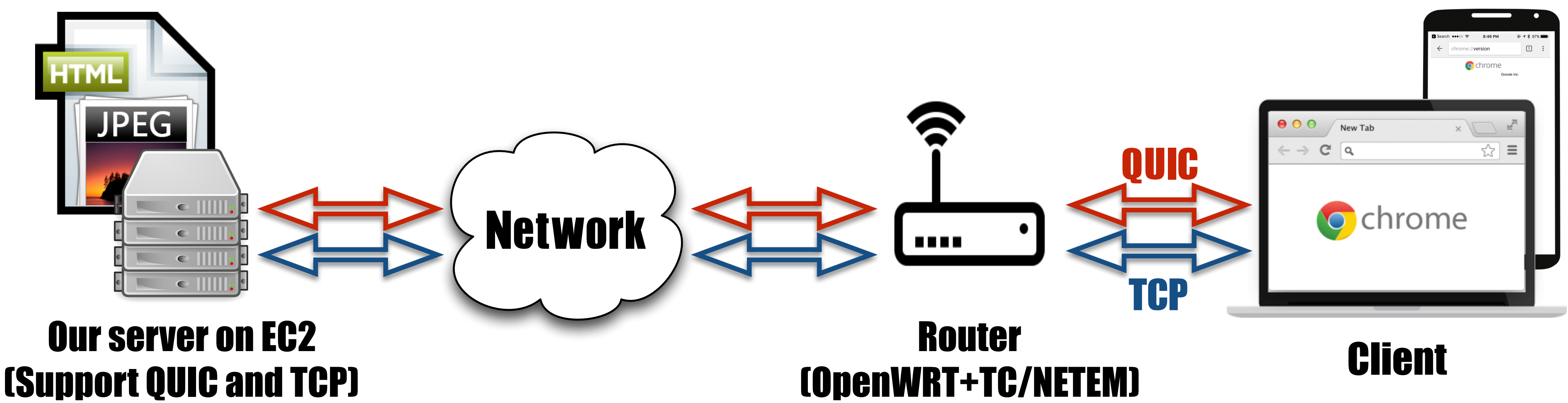


1- Google servers
- No control
- Unexpected behavior

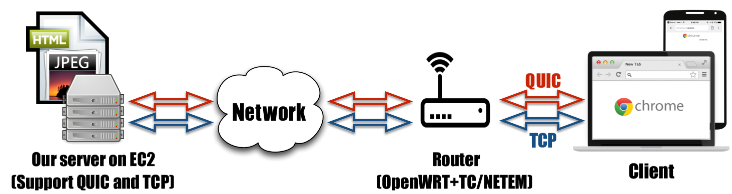
2- Server in Chromium
- Not performant



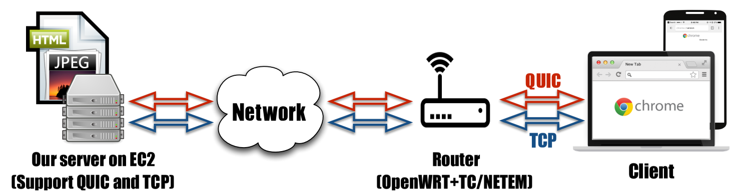
HTTP Performance: QUIC vs. TCP

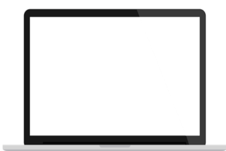


HTTP Performance: QUIC vs. TCP

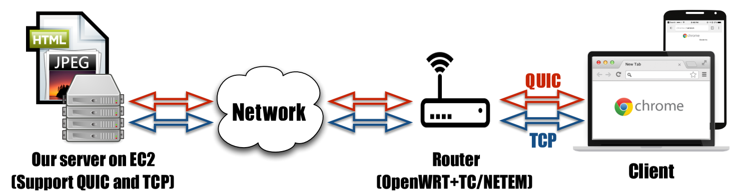


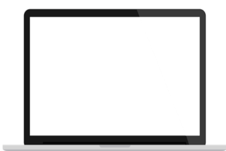
HTTP Performance: QUIC vs. TCP





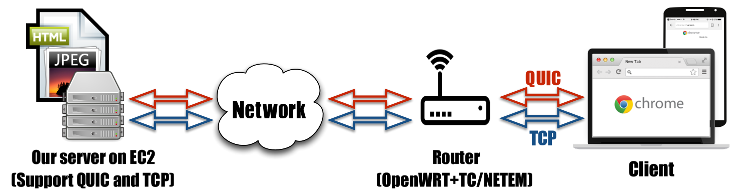
HTTP Performance: QUIC vs. TCP

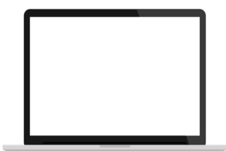




5KB 10KB 100KB 200KB 500KB 1MB 10MB

HTTP Performance: QUIC vs. TCP





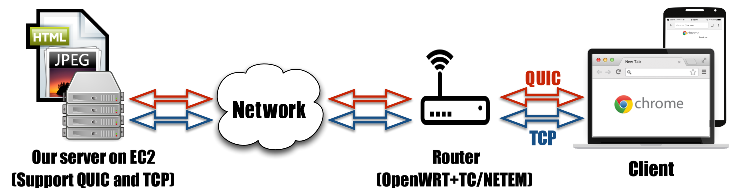
10Mbps

50Mbps

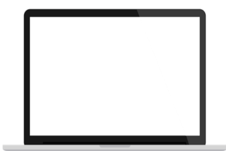
100Mbps

5KB 10KB 100KB 200KB 500KB 1MB 10MB

HTTP Performance: QUIC vs. TCP

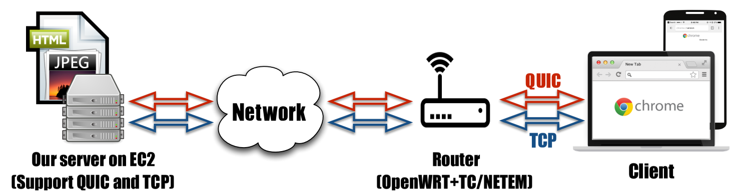


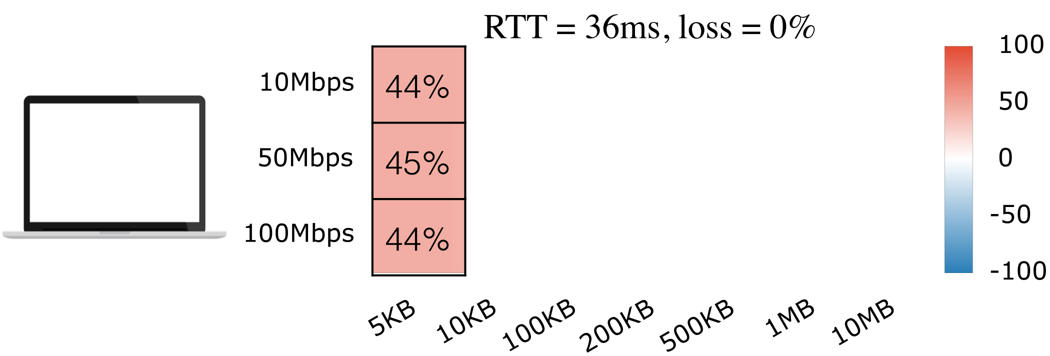
RTT = 36ms, loss = 0%



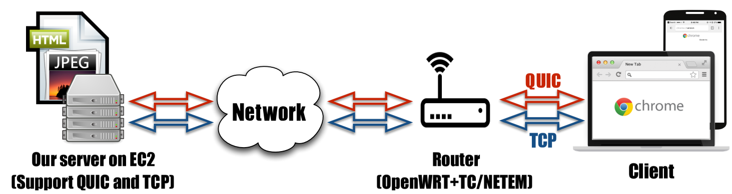
	5KB	10KB	100KB	200KB	500KB	1MB	10MB
10Mbps	44%						
50Mbps	45%						
100Mbps	44%						

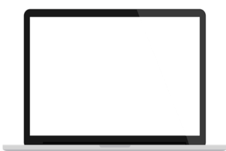
HTTP Performance: QUIC vs. TCP

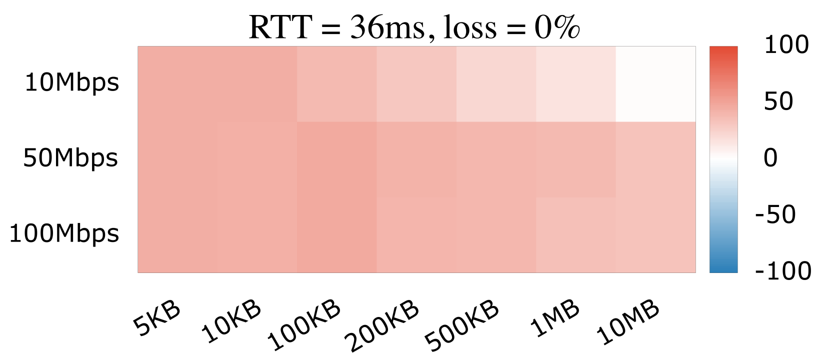




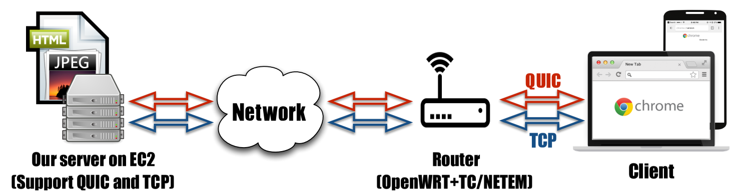
HTTP Performance: QUIC vs. TCP

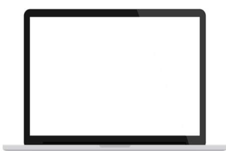


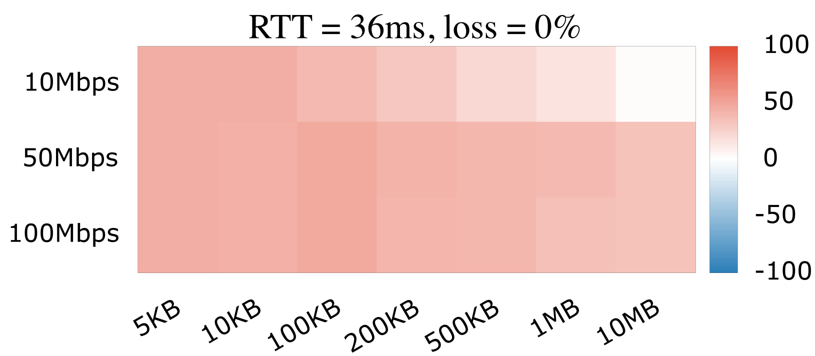


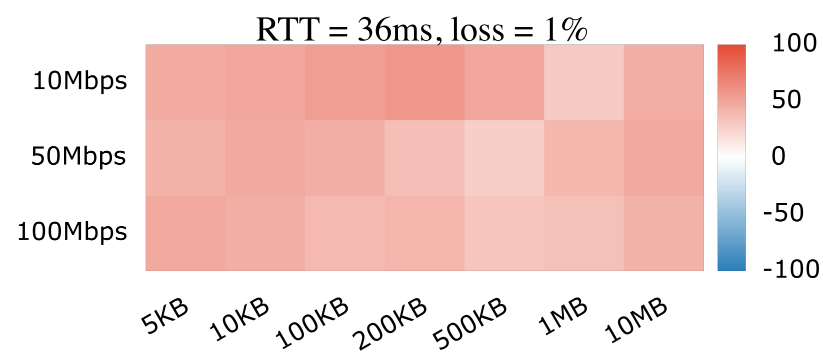


HTTP Performance: QUIC vs. TCP

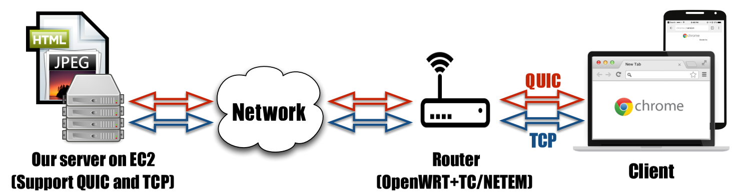


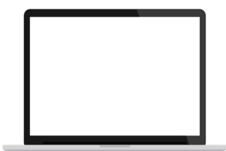


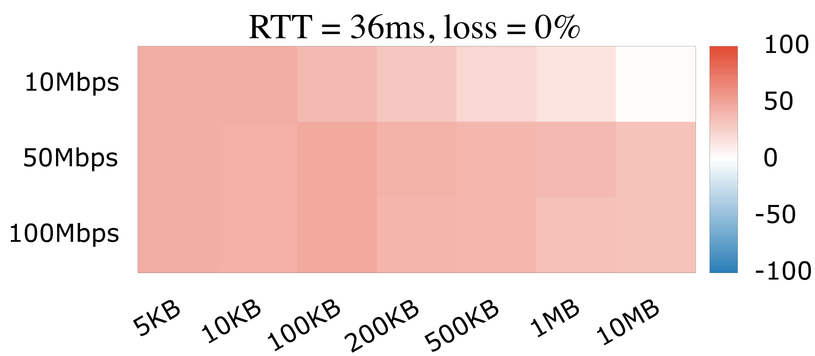


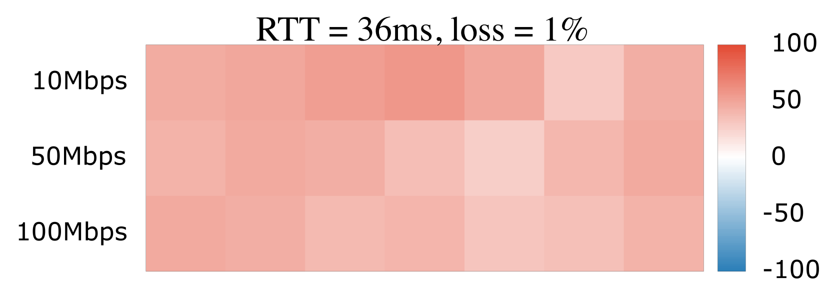


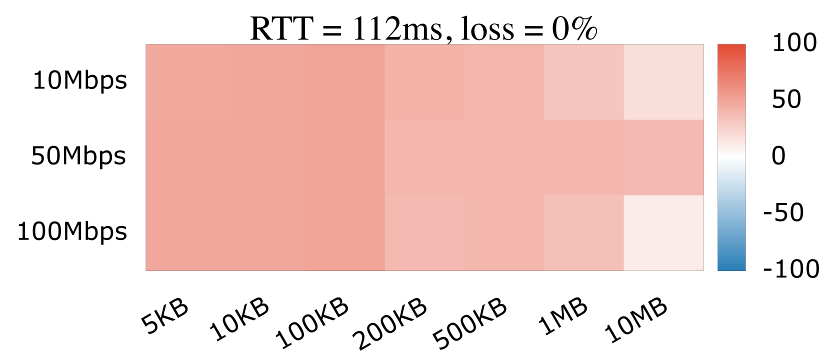
HTTP Performance: QUIC vs. TCP



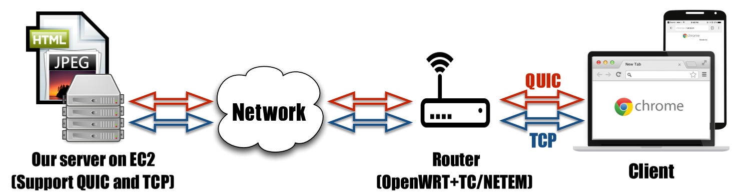


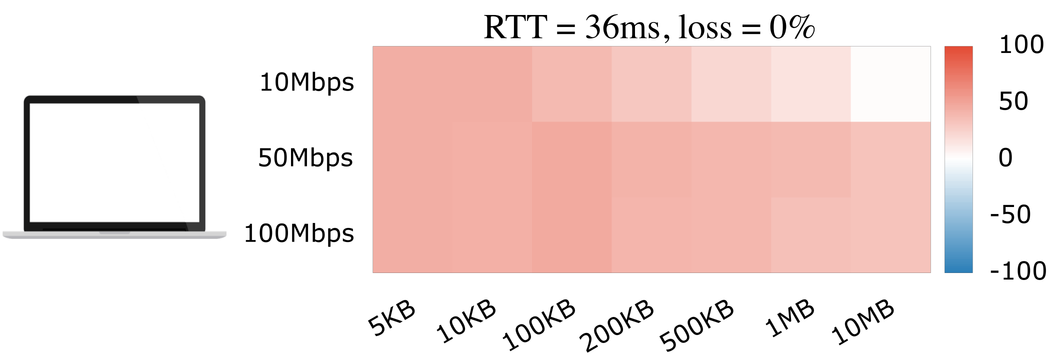


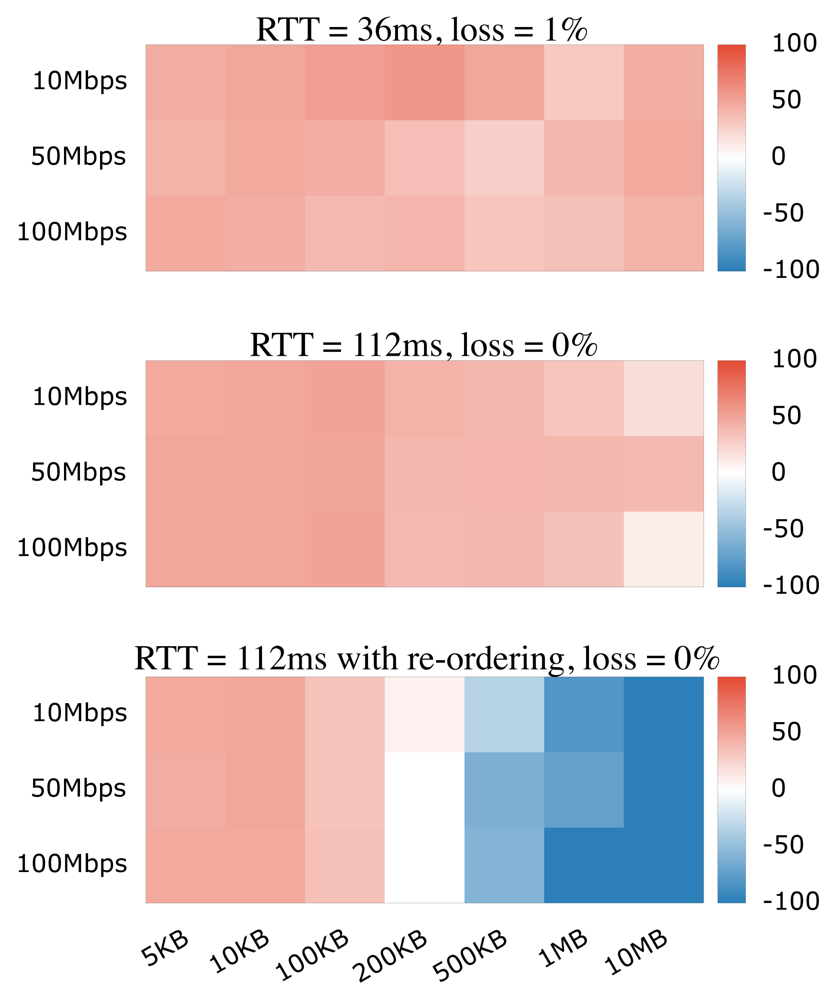




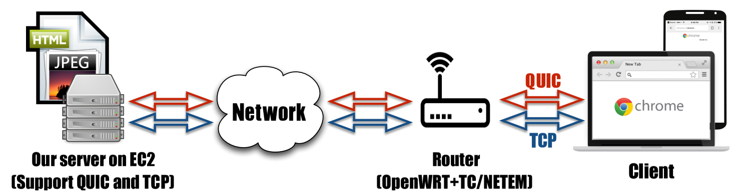
HTTP Performance: QUIC vs. TCP

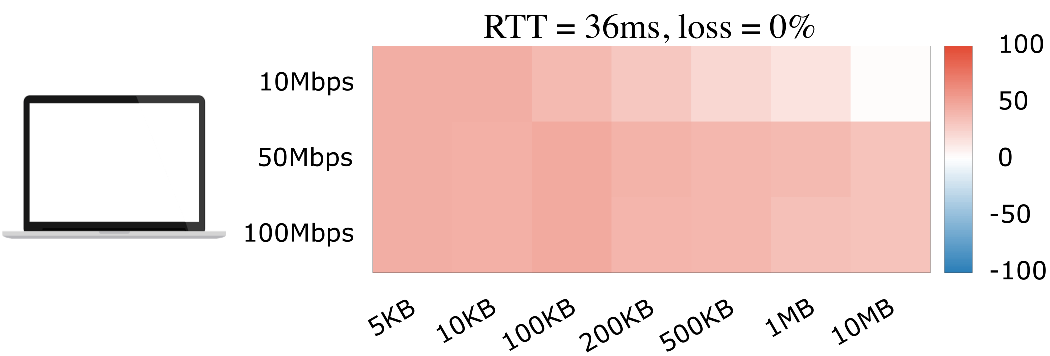


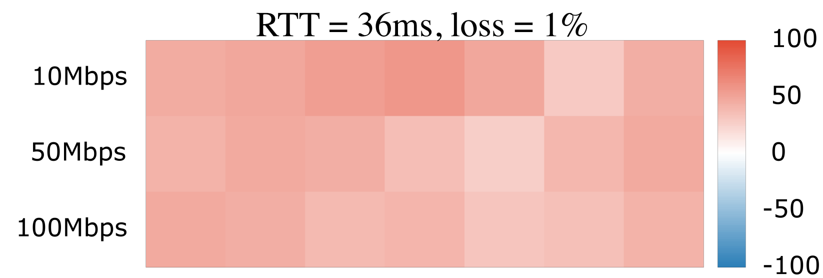


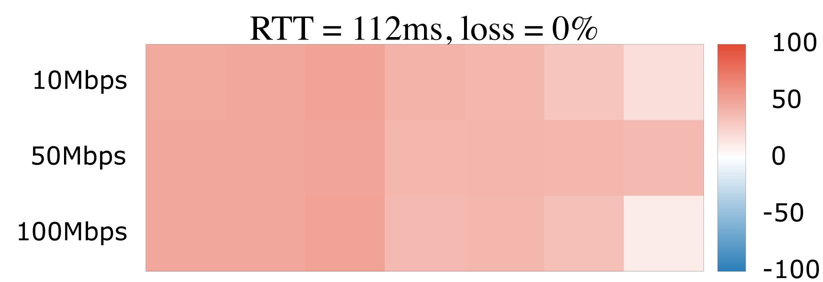


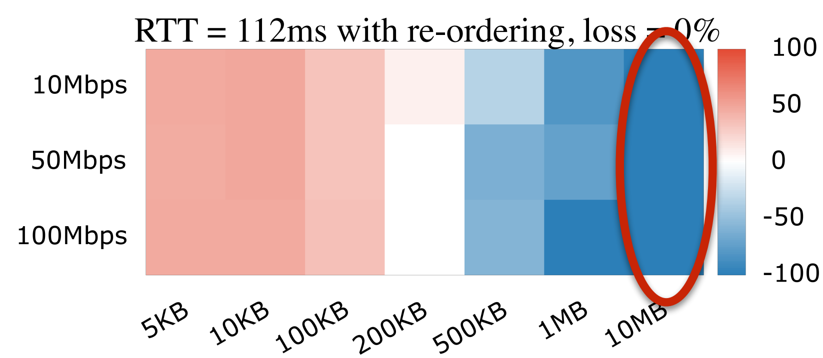
HTTP Performance: QUIC vs. TCP



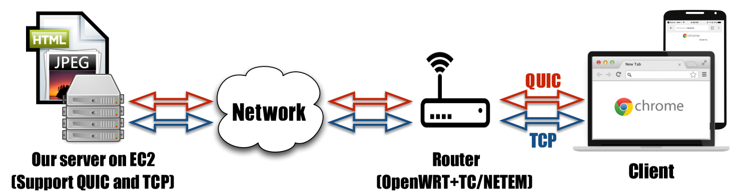


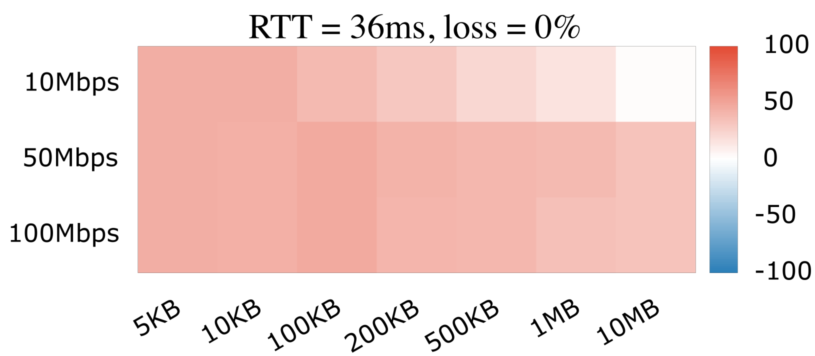


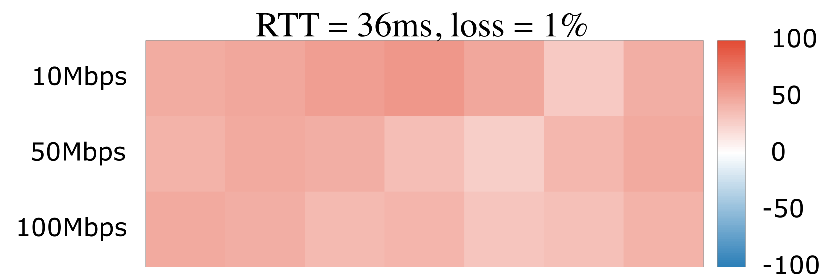


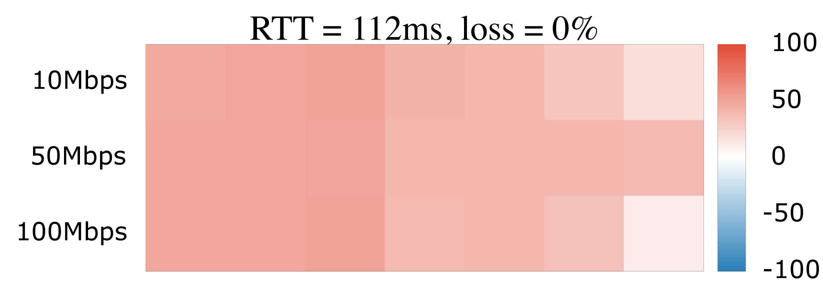


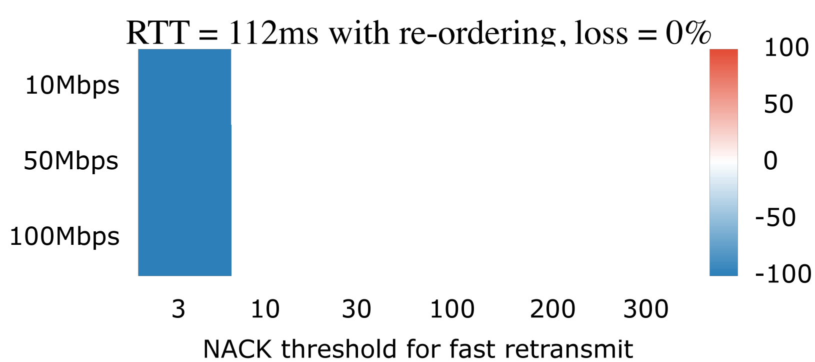
HTTP Performance: QUIC vs. TCP

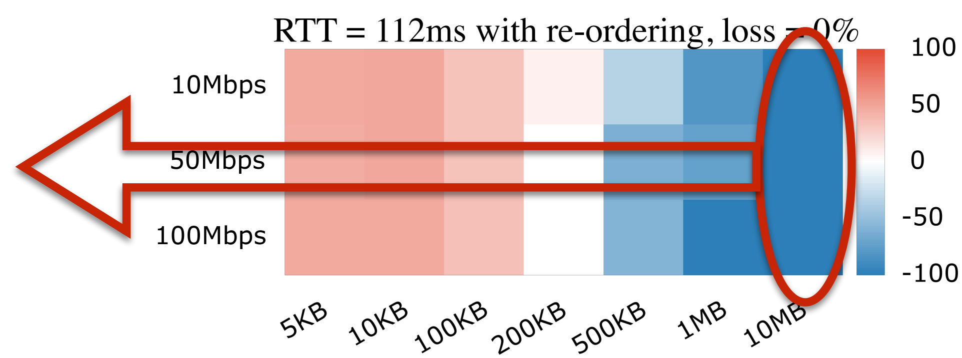




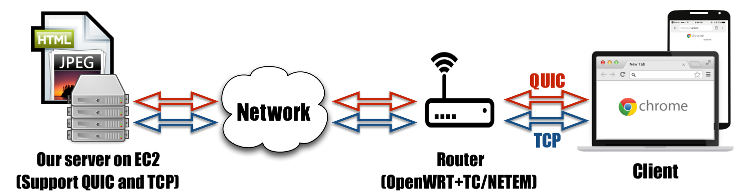


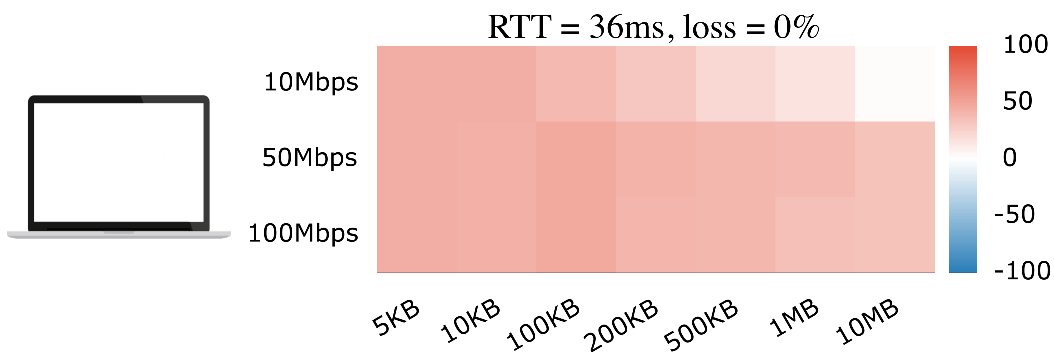


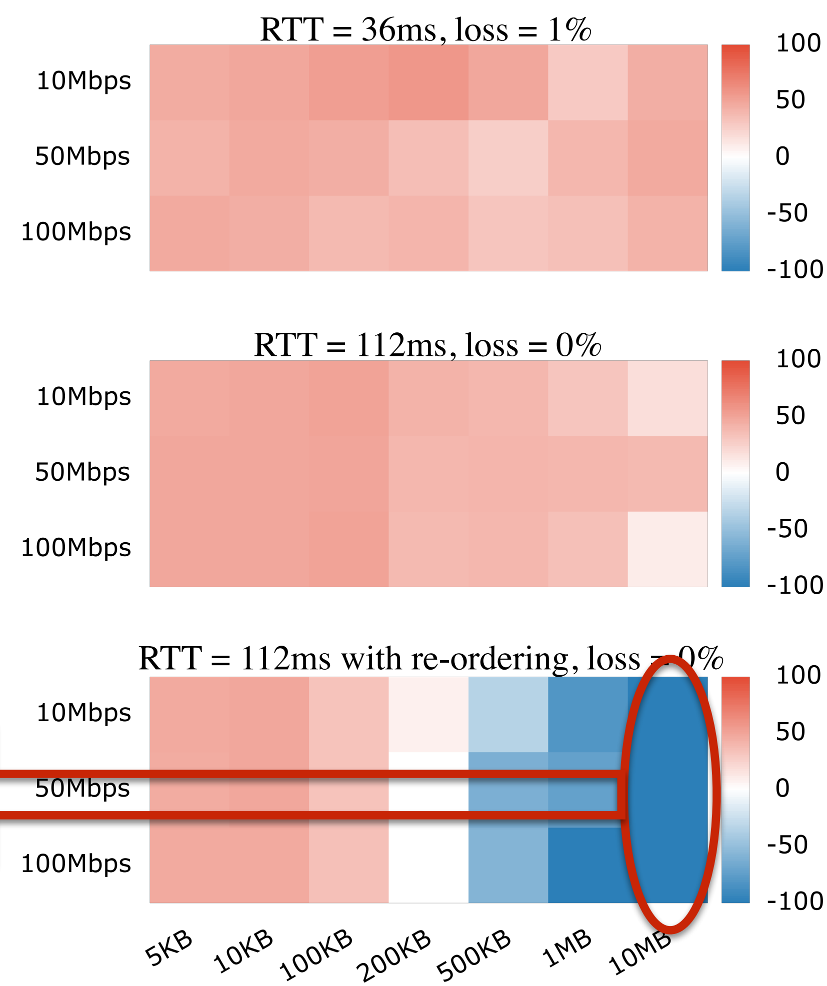


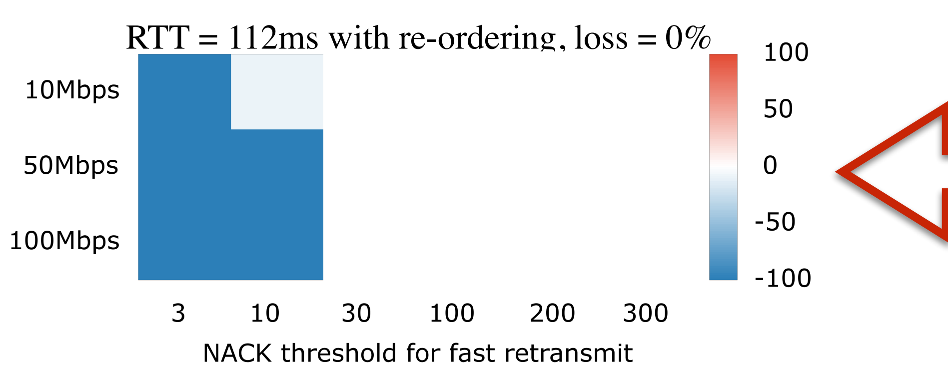


HTTP Performance: QUIC vs. TCP

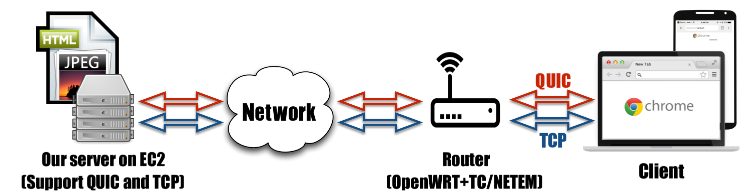


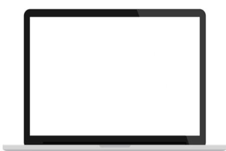


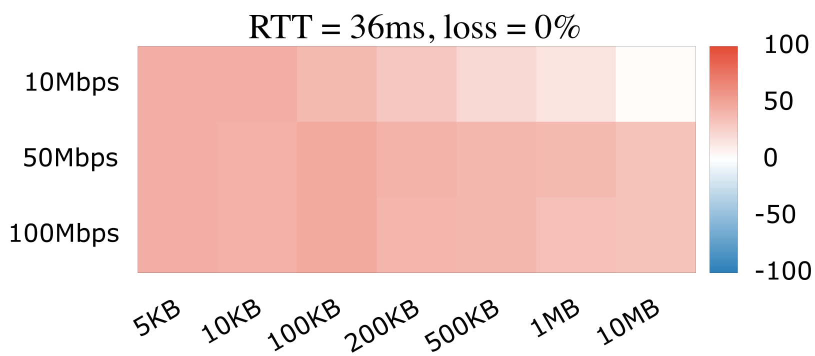


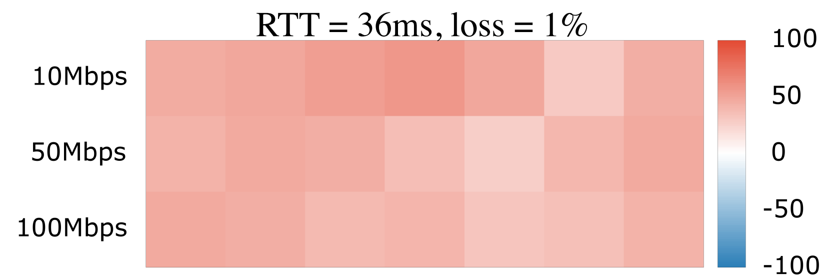


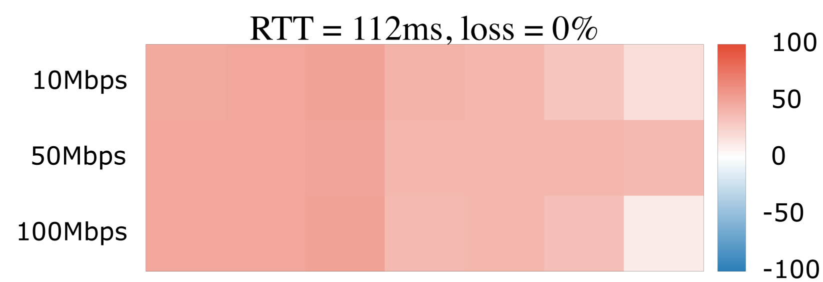
HTTP Performance: QUIC vs. TCP

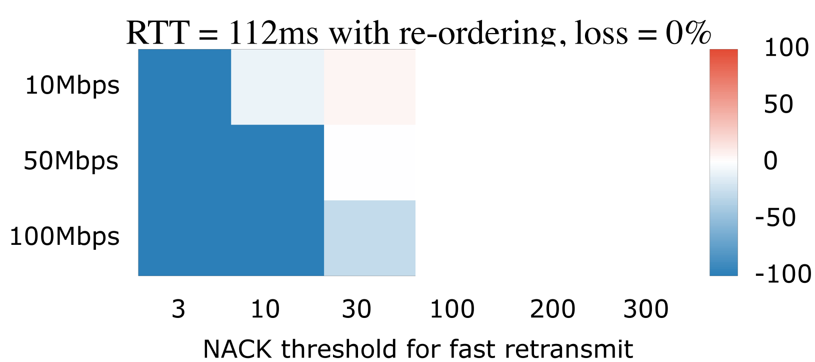


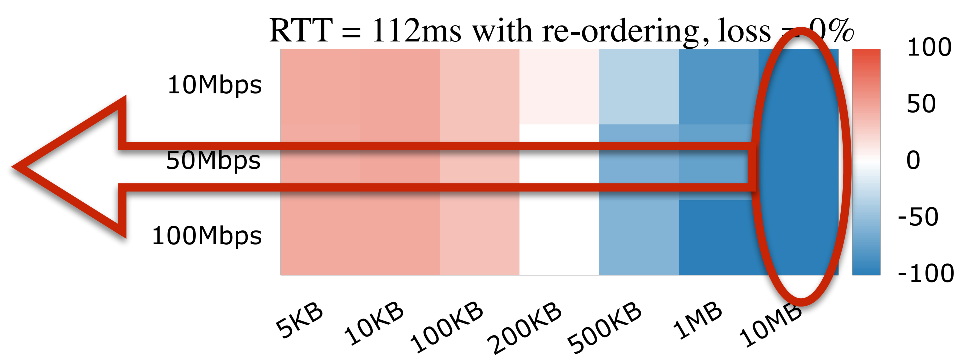




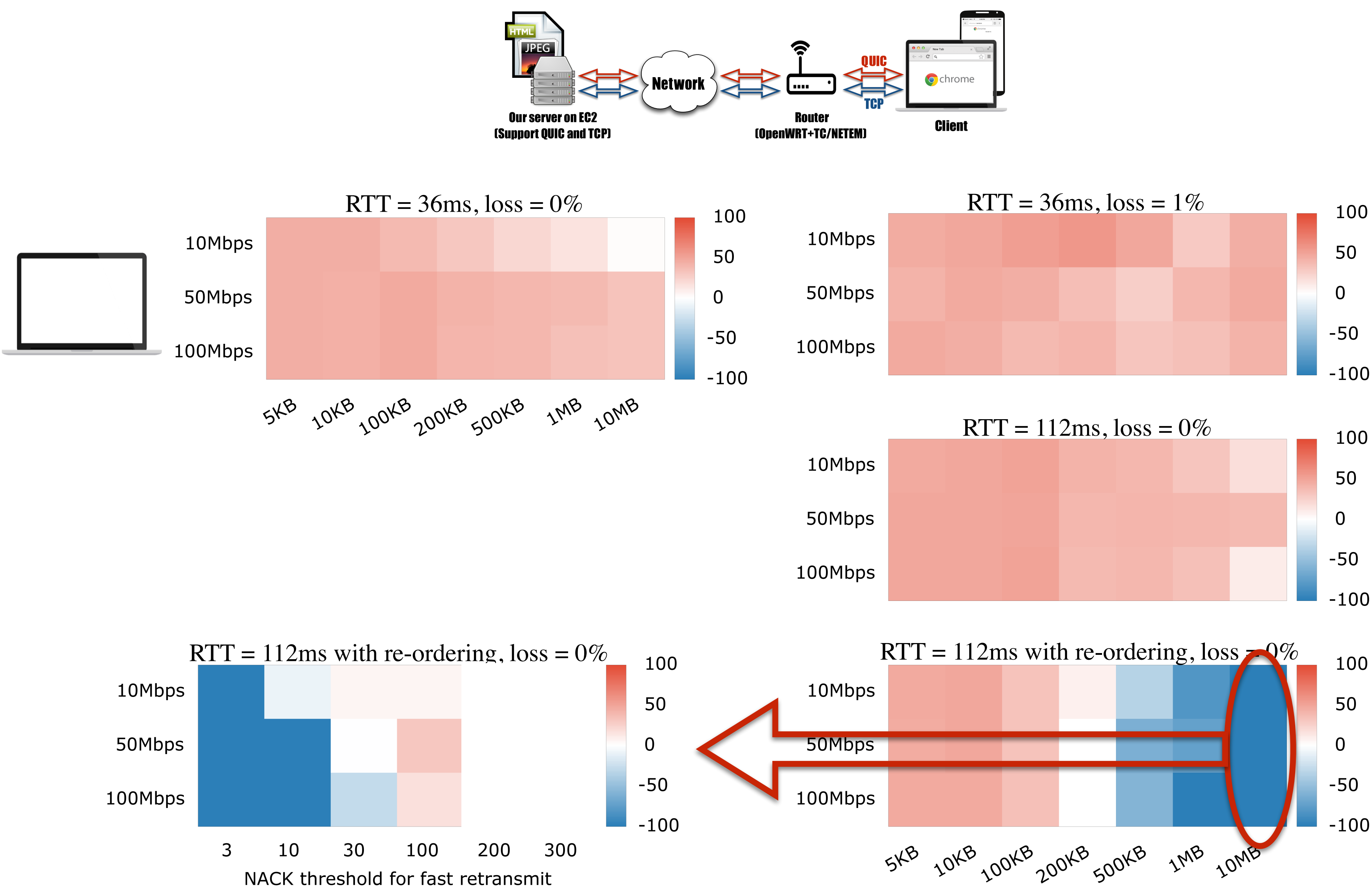




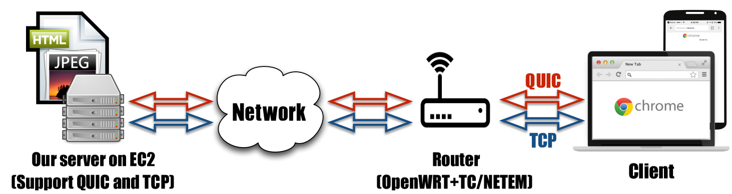


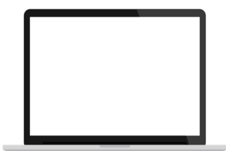


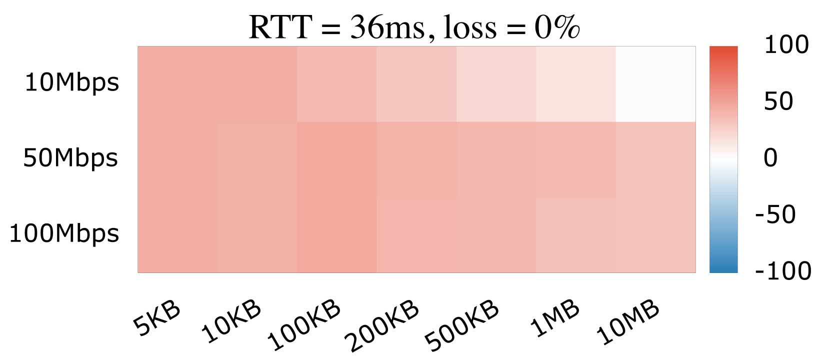
HTTP Performance: QUIC vs. TCP

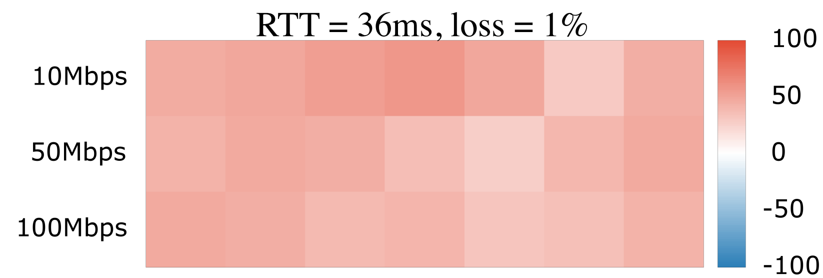


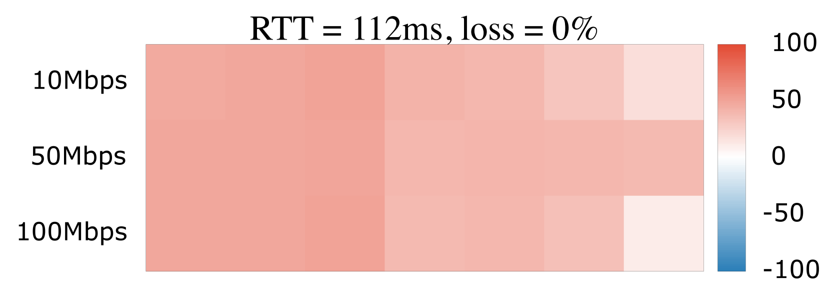
HTTP Performance: QUIC vs. TCP

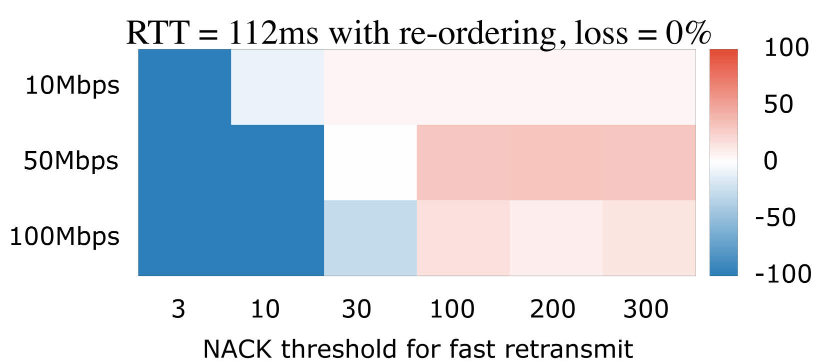


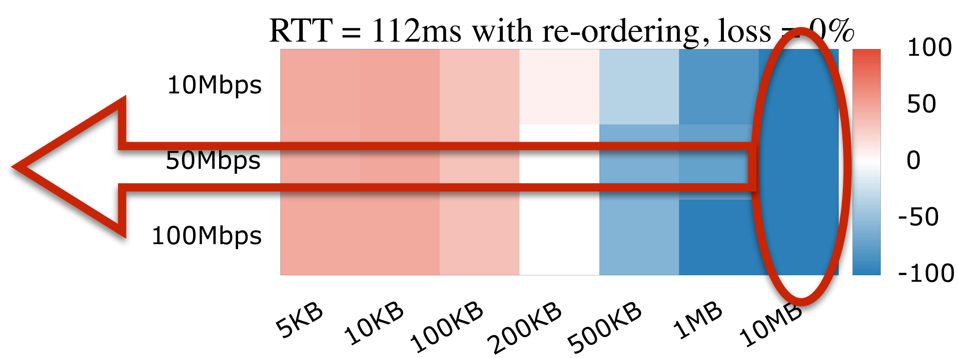




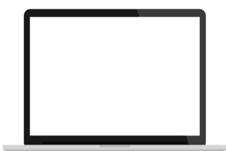


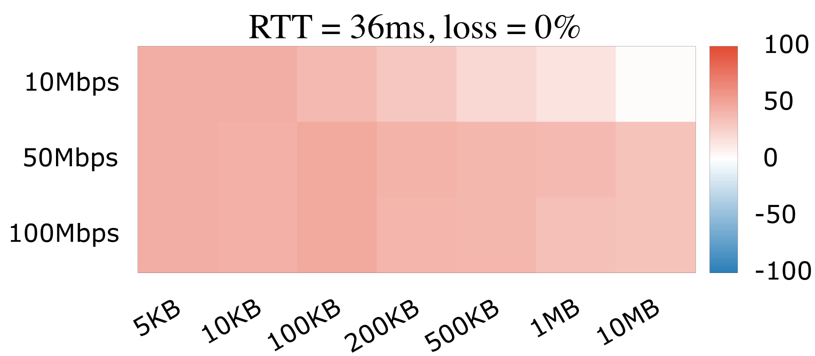




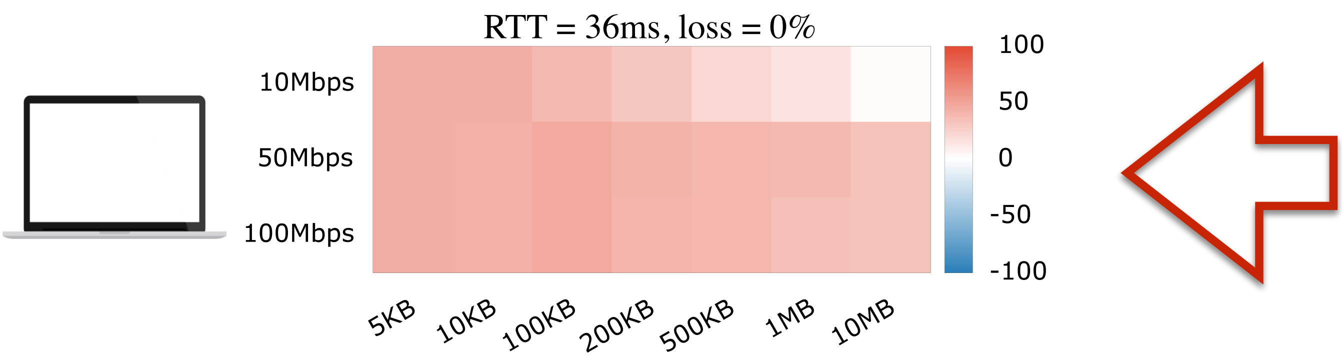


How much does 0-RTT help?



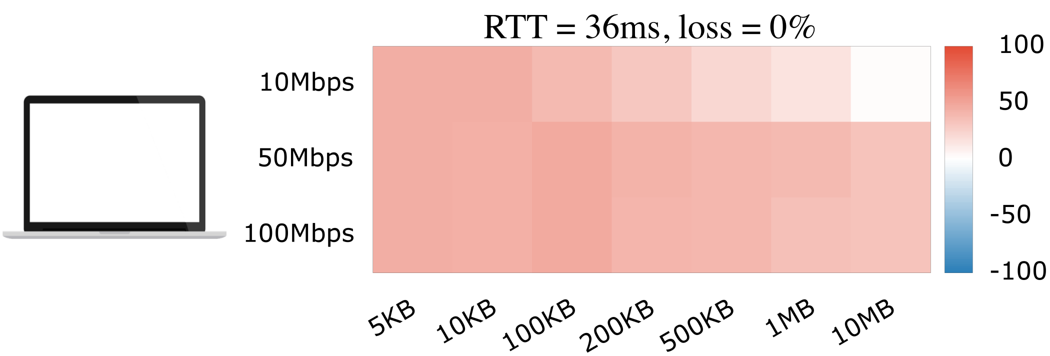


How much does 0-RTT help?

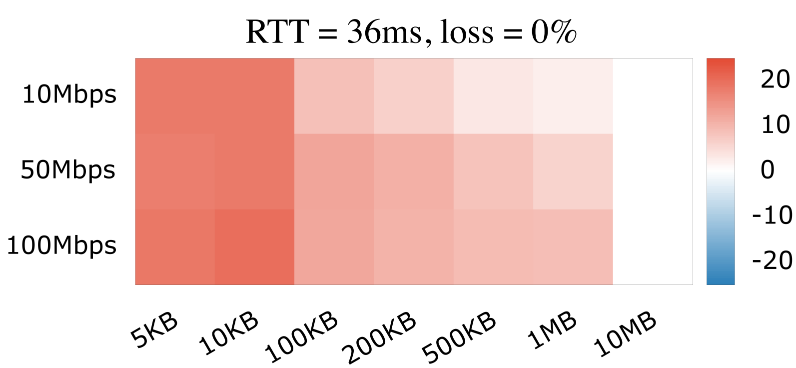


QUIC vs. TCP

How much does 0-RTT help?



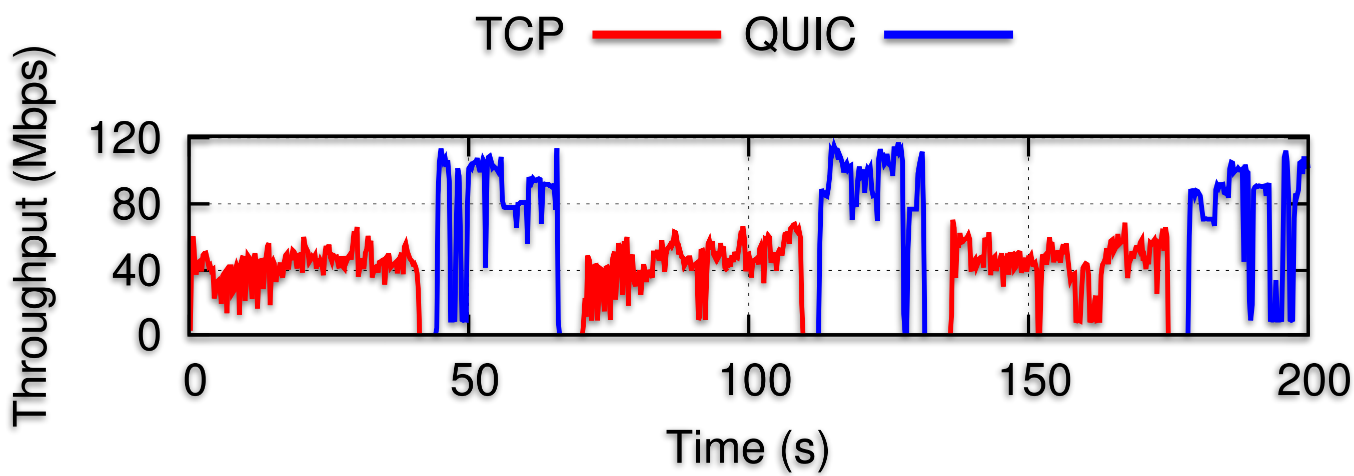
QUIC vs. TCP



QUIC with 0-RTT
vs.
QUIC without 0-RTT

Variable Bandwidth

Bandwidth fluctuates between 50 Mbps and 150 Mbps



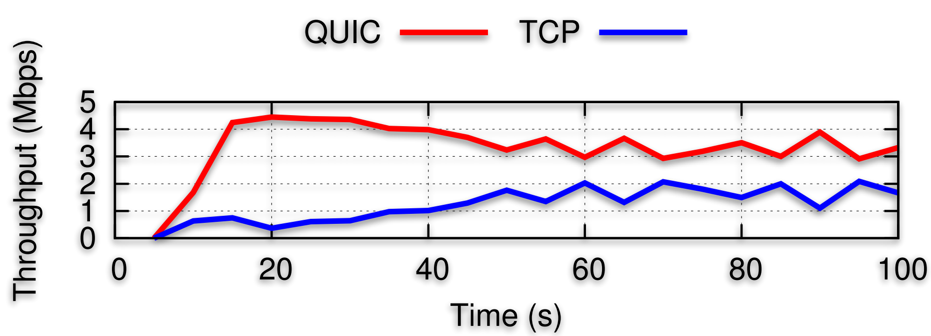
QUIC avg. throughput: 79 Mbps

TCP avg. throughput: 46 Mbps

Fairness

Fairness

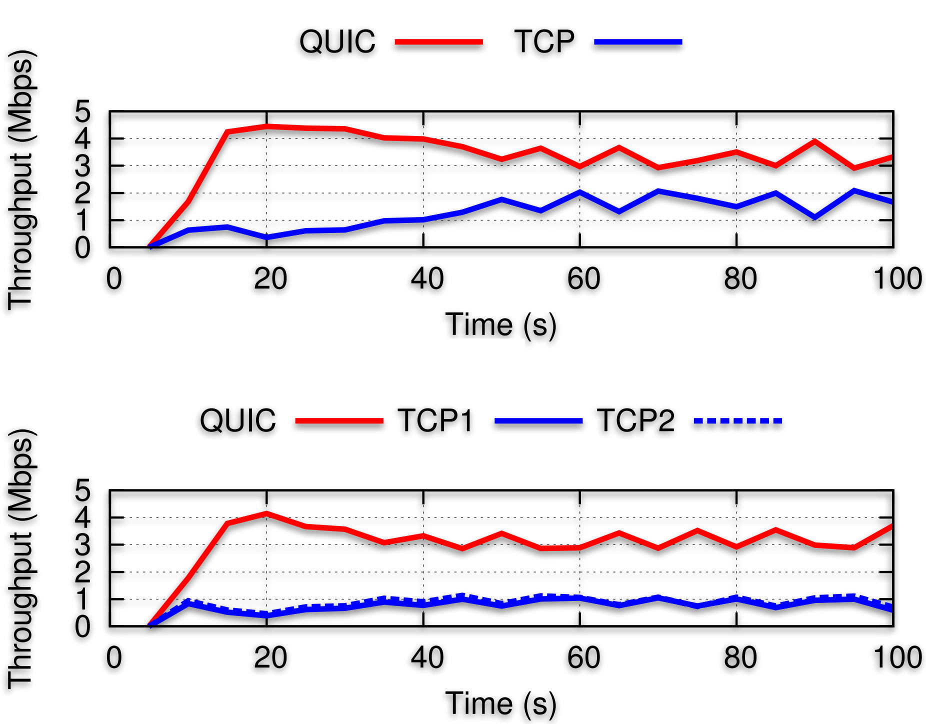
5Mbps bottleneck link, RTT=36ms



	Flow	Avg. Xput
QUIC vs. TCP	QUIC TCP	2.71 1.62

Fairness

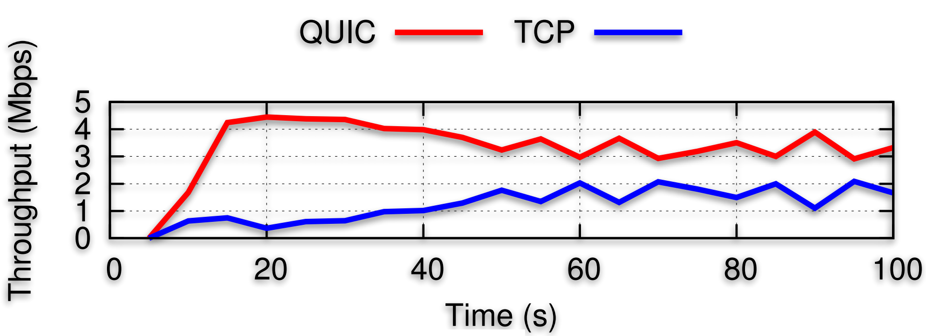
5Mbps bottleneck link, RTT=36ms

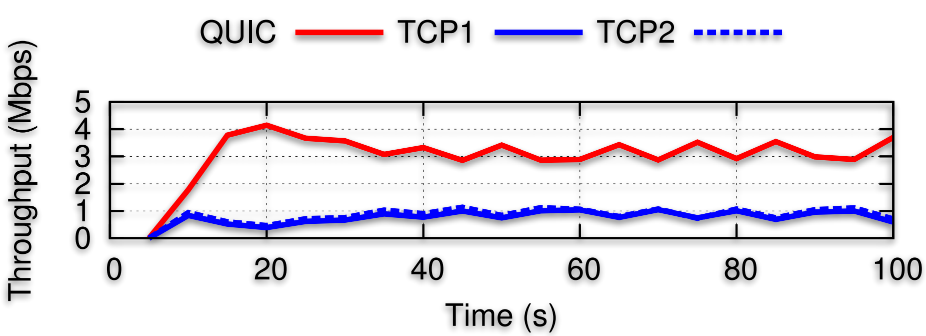


	Flow	Avg. Xput
QUIC vs. TCP	QUIC	2.71
	TCP	1.62
QUIC vs. TCPx2	QUIC	2.8
	TCP1	0.7
	TCP2	0.96
QUIC vs. TCPx4	QUIC	2.75
	TCP1	0.45
	TCP2	0.36
	TCP1	0.41
	TCP2	0.45

Fairness

5Mbps bottleneck link, RTT=36ms

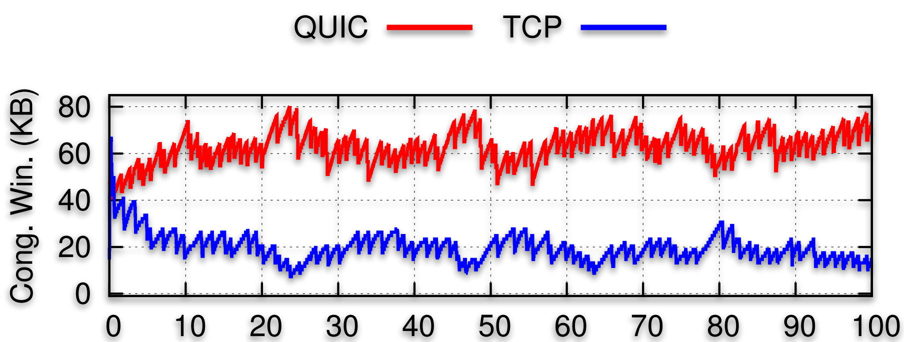




Under same conditions, our tests shows that QUIC vs. QUIC and TCP vs. TCP resulted in fair splits of the bandwidth

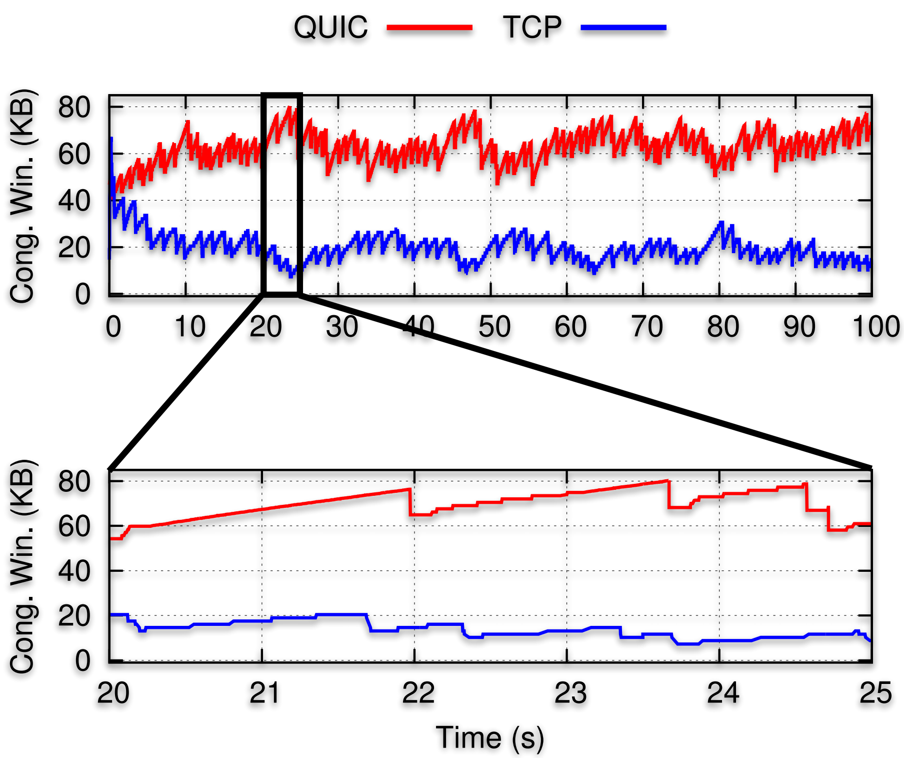
	Flow	Avg. Xput
QUIC vs. TCP	QUIC TCP	2.71 1.62
QUIC vs. TCPx2	QUIC TCP1 TCP2	2.8 0.7 0.96
QUIC vs. TCPx4	QUIC TCP1 TCP2 TCP1 TCP2	2.75 0.45 0.36 0.41 0.45

Fairness



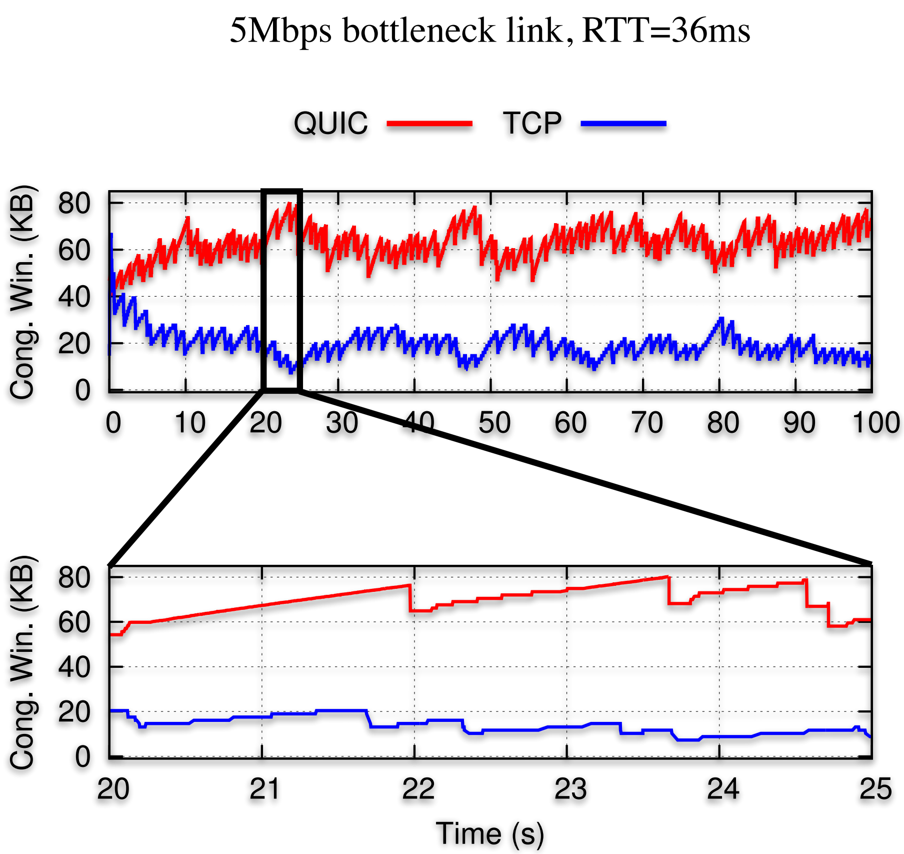
	Flow	Avg. Xput
QUIC vs. TCP	QUIC	2.71
	TCP	1.62
QUIC vs. TCPx2	QUIC	2.8
	TCP1	0.7
	TCP2	0.96
QUIC vs. TCPx4	QUIC	2.75
	TCP1	0.45
	TCP2	0.36
	TCP1	0.41
	TCP2	0.45

Fairness



	Flow	Avg. Xput
QUIC vs. TCP	QUIC	2.71
	TCP	1.62
QUIC vs. TCPx2	QUIC	2.8
	TCP1	0.7
	TCP2	0.96
QUIC vs. TCPx4	QUIC	2.75
	TCP1	0.45
	TCP2	0.36
	TCP1	0.41
	TCP2	0.45

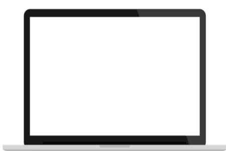
Fairness

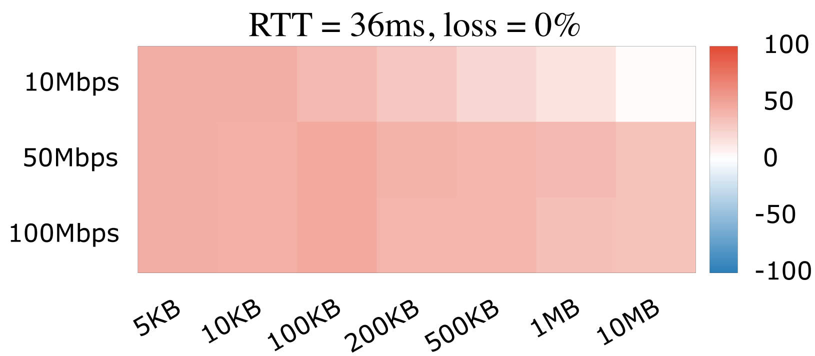


	Flow	Avg. Xput
QUIC vs. TCP	QUIC	2.71
	TCP	1.62
QUIC vs. TCPx2	QUIC	2.8
	TCP1	0.7
	TCP2	0.96
QUIC vs. TCPx4	QUIC	2.75
	TCP1	0.45
	TCP2	0.36
	TCP1	0.41
	TCP2	0.45

HTTP Performance: QUIC vs. TCP

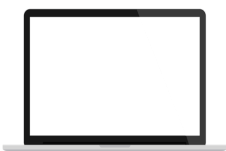
Desktop vs. Mobile

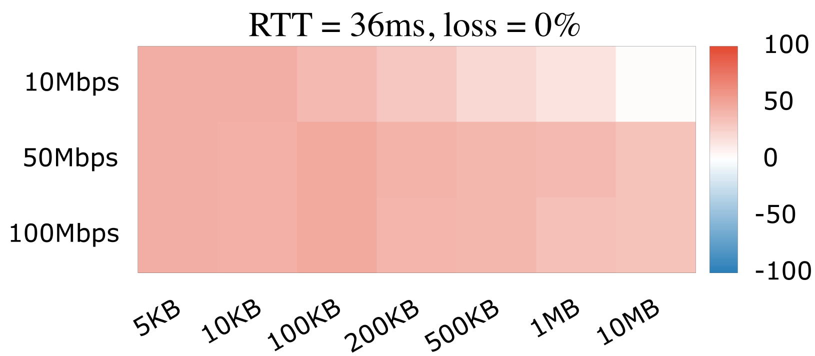


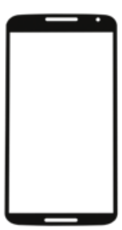


HTTP Performance: QUIC vs. TCP

Desktop vs. Mobile

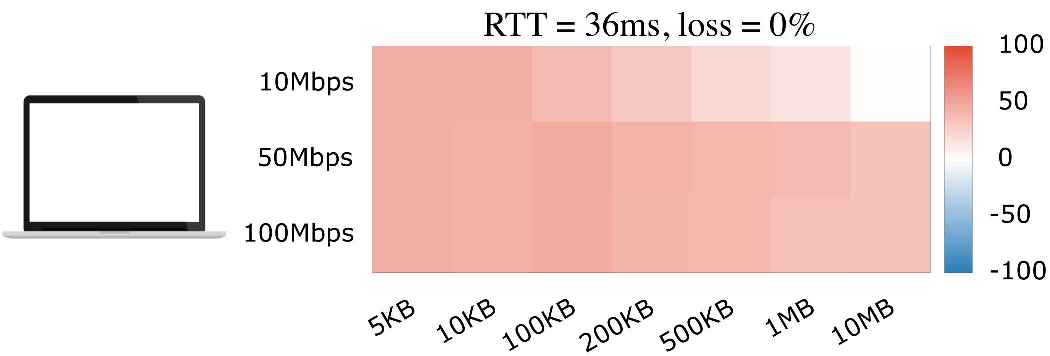


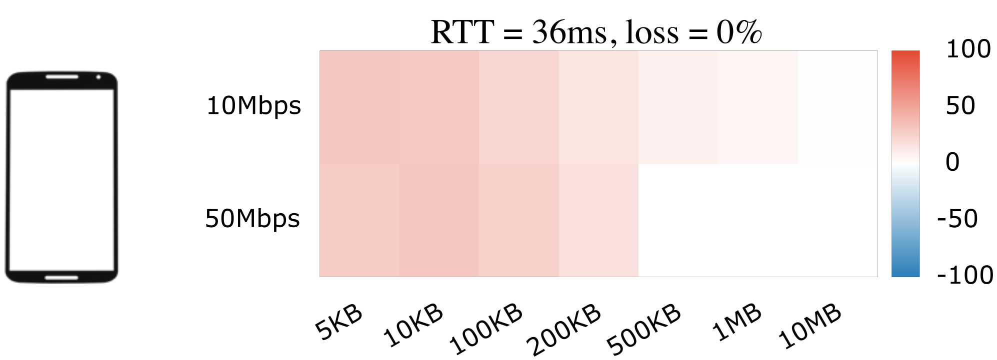




HTTP Performance: QUIC vs. TCP

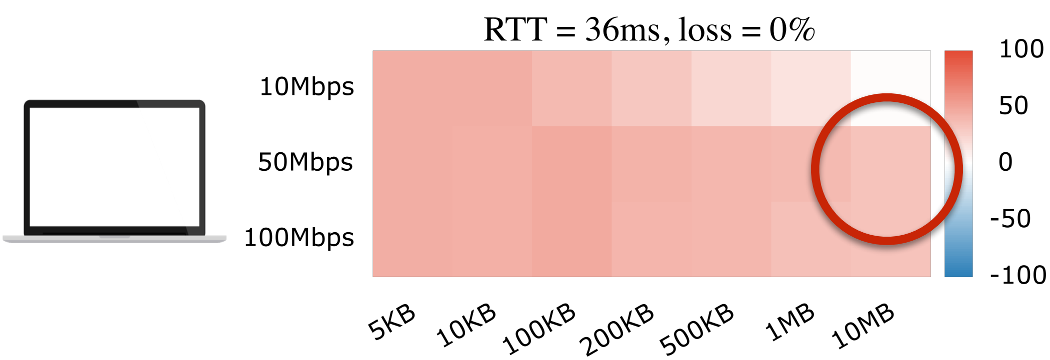
Desktop vs. Mobile

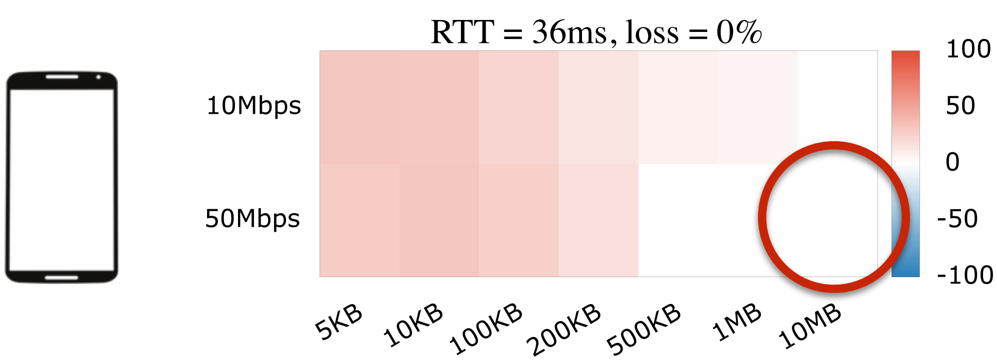




HTTP Performance: QUIC vs. TCP

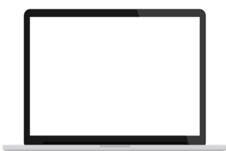
Desktop vs. Mobile

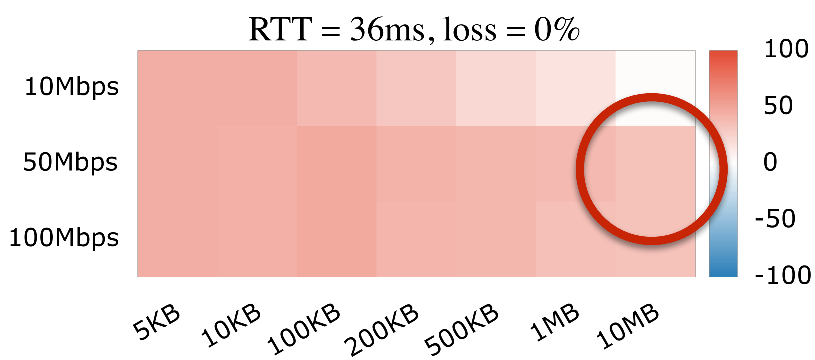


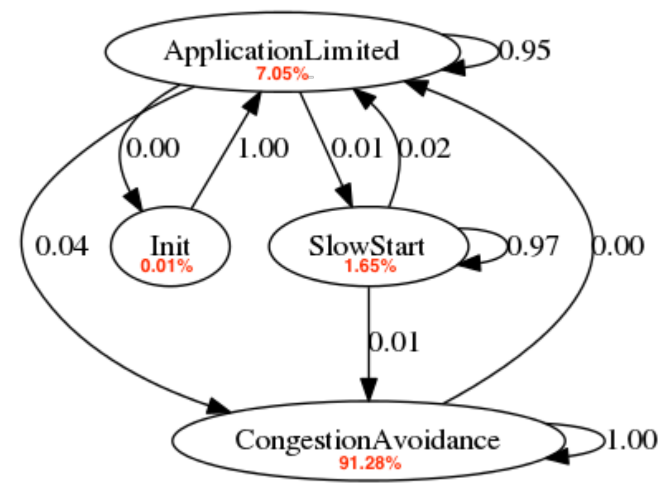


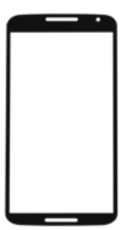
HTTP Performance: QUIC vs. TCP

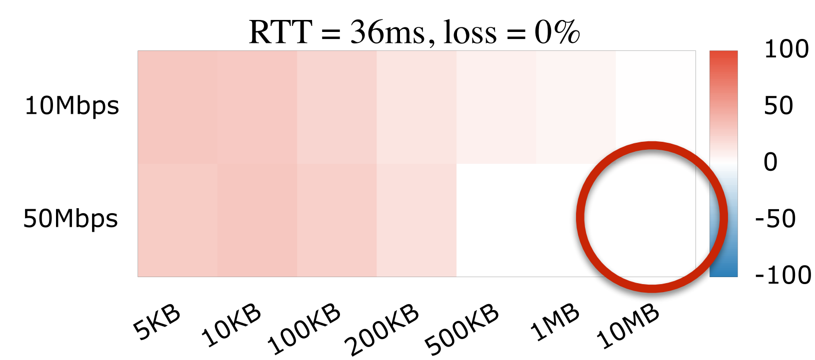
Desktop vs. Mobile

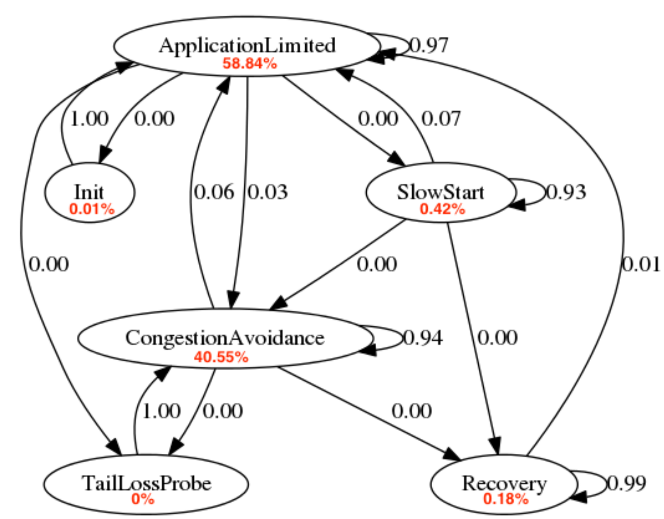






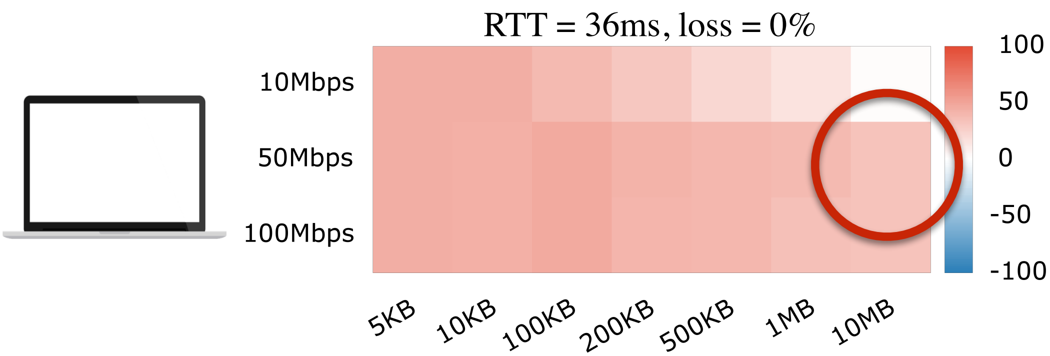


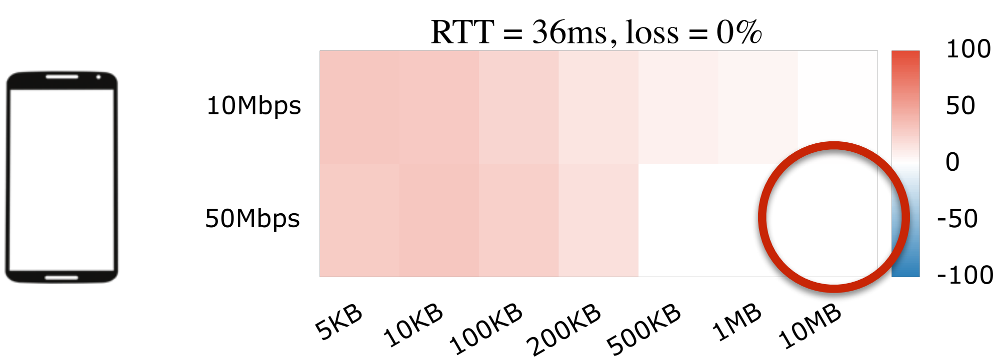




HTTP Performance: QUIC vs. TCP

Desktop vs. Mobile

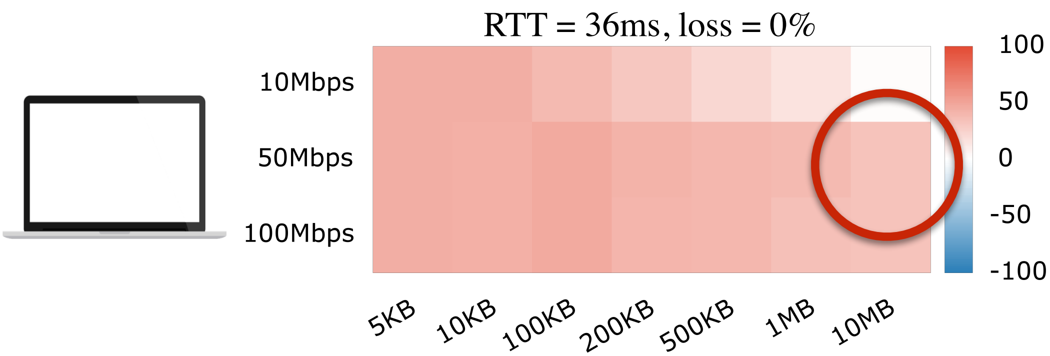


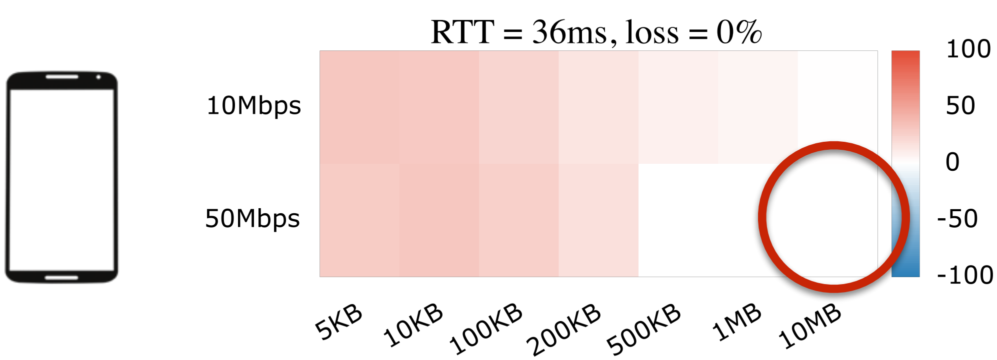


% of time spent in each state			
Slow Start	Cong. Avoid.	App. Limit.	Recovery
1.7%	91.5%	7.1%	0%
0.42%	40.6%	58.8%	0.2%

HTTP Performance: QUIC vs. TCP

Desktop vs. Mobile





% of time spent in each state			
Slow Start	Cong. Avoid.	App. Limit.	Recovery
1.7%	91.5%	7.1%	0%
0.42%	40.6%	58.8%	0.2%

QUIC[K] Summary

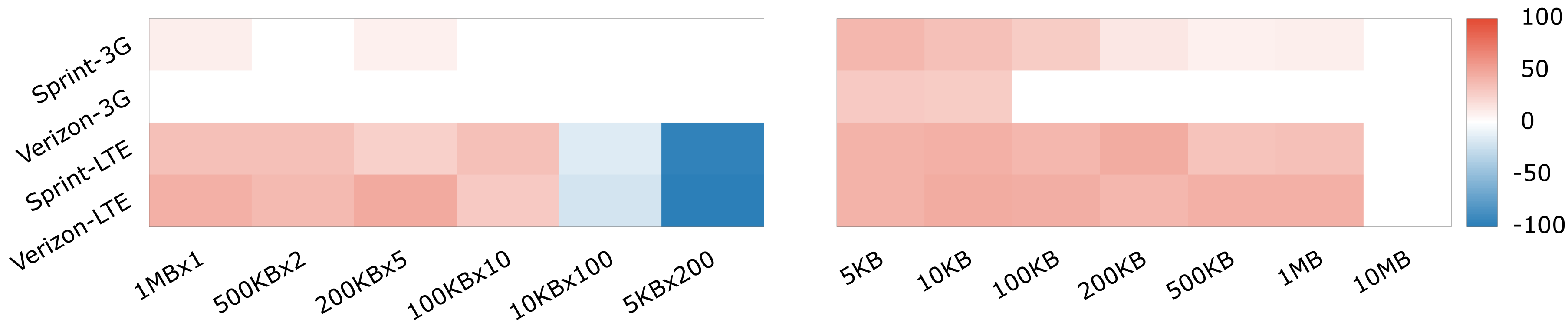
1. Evaluated an application-layer transport protocol
    - Rapidly evolving
    - Deployed at scale with nonpublic configuration parameters.
2. Evaluations used settings that approximate those deployed in the wild
3. Controlled experiments
    - Variety of conditions and environments
    - Multiple versions
4. Instrumented the protocol
    - Inferred state machines
    - Provided root cause analysis
5. Approach can be applied to future versions and protocols

Thank you!

Explored Space

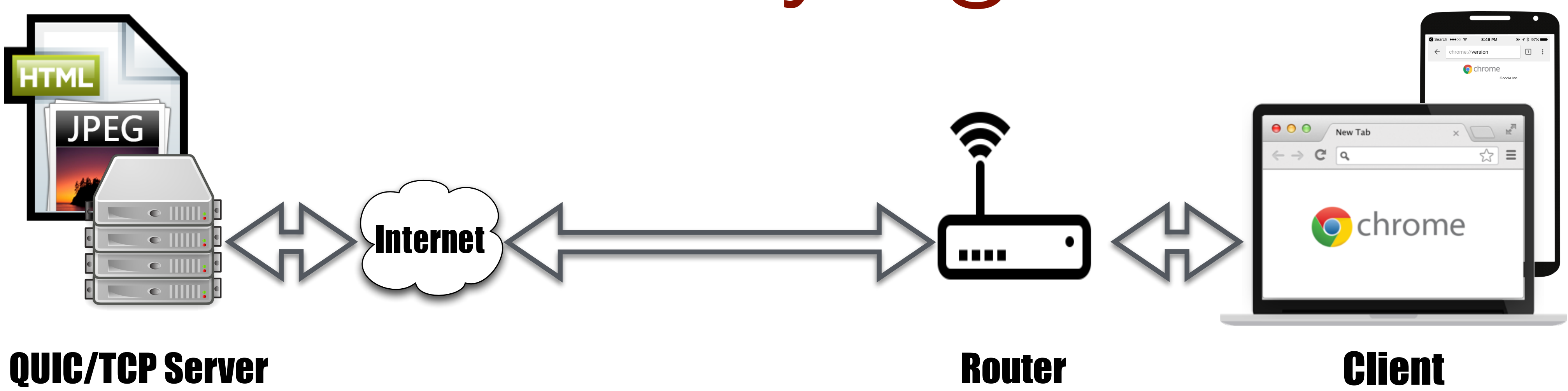
Parameter	Values tested
Rate limits (Mbps)	5, 10, 50, 100
Extra Delay (RTT)	0ms, 50ms, 100ms
Extra Loss	0.1%, 1%
Number of objects	1, 2, 5, 10, 100, 200
Object sizes (KB)	5, 10, 100, 200, 500, 1000, 10,000, 210,000
Proxy	QUIC proxy, TCP proxy
Clients	Desktop, Nexus6, MotoG
Video qualities	tiny, medium, hd720, hd2160

Real Cell Networks

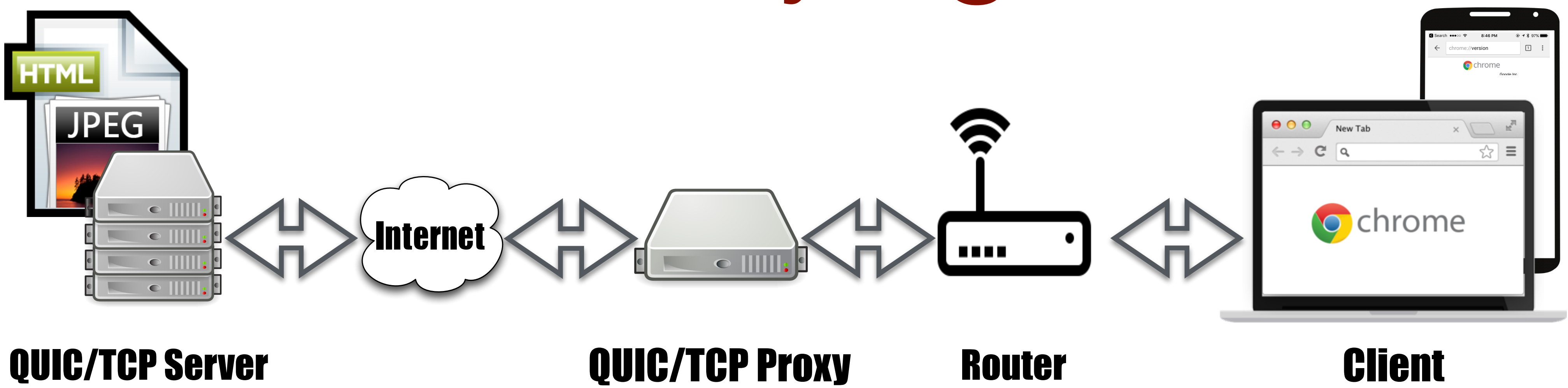


	Throughput (Mbps)		RTT (ms)		Reordering (10%)		Loss (%)	
	3G	LTE	3G	LTE	3G	LTE	3G	LTE
Verizon	0.17	4.0	109	62	9	0.25	0.05	0
Sprint	0.31	2.4	70	55	1.38	0.13	0.02	0.02

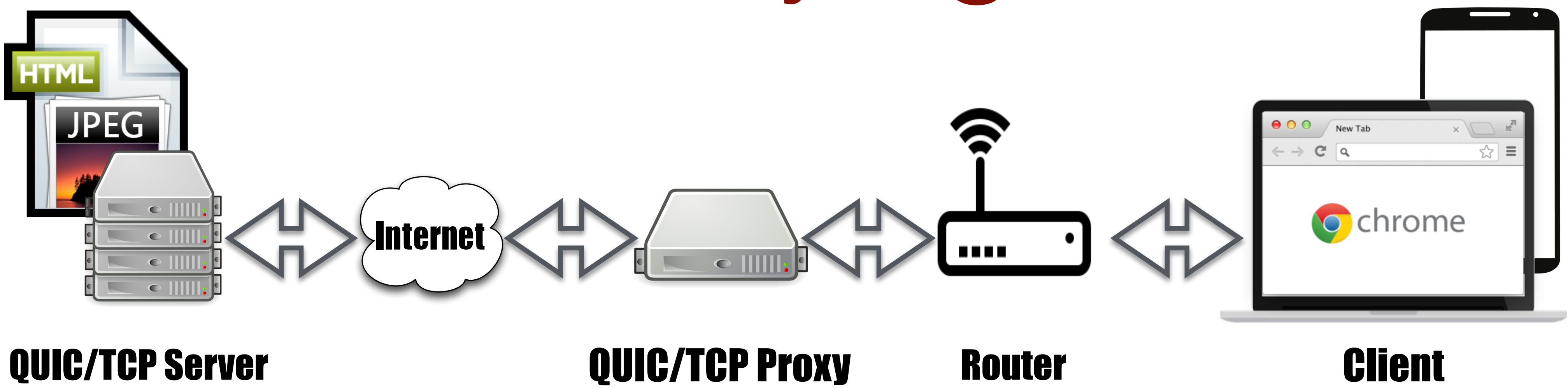
Proxying



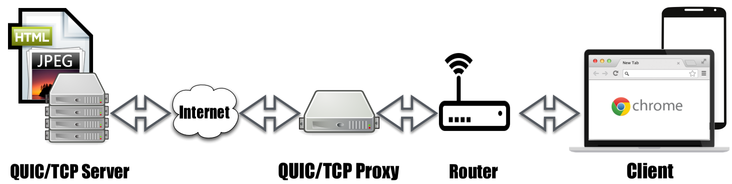
Proxying


HTML
JPEG
Internet
chrome
New Tab
QUIC/TCP Server
QUIC/TCP Proxy
Router
Client

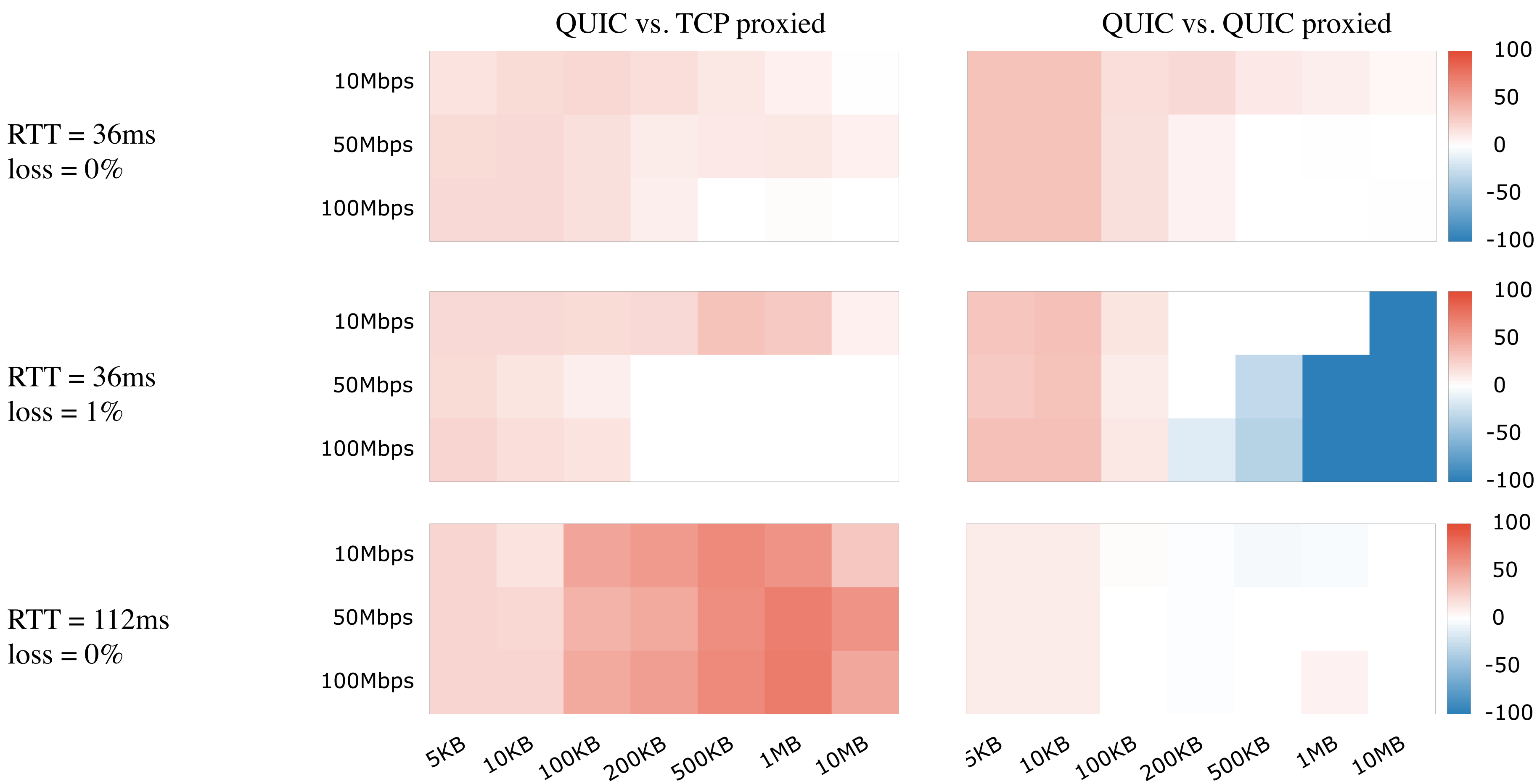
Proxying


HTML
JPEG
Internet
New Tab
chrome
QUIC/TCP Server
QUIC/TCP Proxy
Router
Client

Proxying



HTML
JPEG
Internet
chrome
QUIC/TCP Server
QUIC/TCP Proxy
Router
Client



QUIC vs. TCP proxied
QUIC vs. QUIC proxied
RTT = 36ms
loss = 0%
RTT = 36ms
loss = 1%
RTT = 112ms
loss = 0%
10Mbps
50Mbps
100Mbps
5KB
10KB
100KB
200KB
500KB
1MB
10MB
100
50
0
-50
-100

Other Experiments

- Historical analysis of QUIC over more than a year
- Proxying
- Video streaming over QUIC
- QUIC in cellular networks

Challenges of Evaluating QUIC

Challenges of Evaluating QUIC

- Rapidly evolving

Challenges of Evaluating QUIC

- Rapidly evolving
- Gap between what is publicly released and what is deployed in production by Google (and others)

Challenges of Evaluating QUIC

- Rapidly evolving
- Gap between what is publicly released and what is deployed in production by Google (and others)
- No formal model for how QUIC should behave