LINUX PITER

TECH CONFERENCE
SAINT PETERSBURG

2018 NOVEMBER 2-3

Increase data transfer speed in mobile networks

Alexander Tobol

2 OK.RU

71 mln

MAU

>2 Tbit/s

Traffic

7000 servers

H/W

Who sets up servers that work with mobile clients?

?

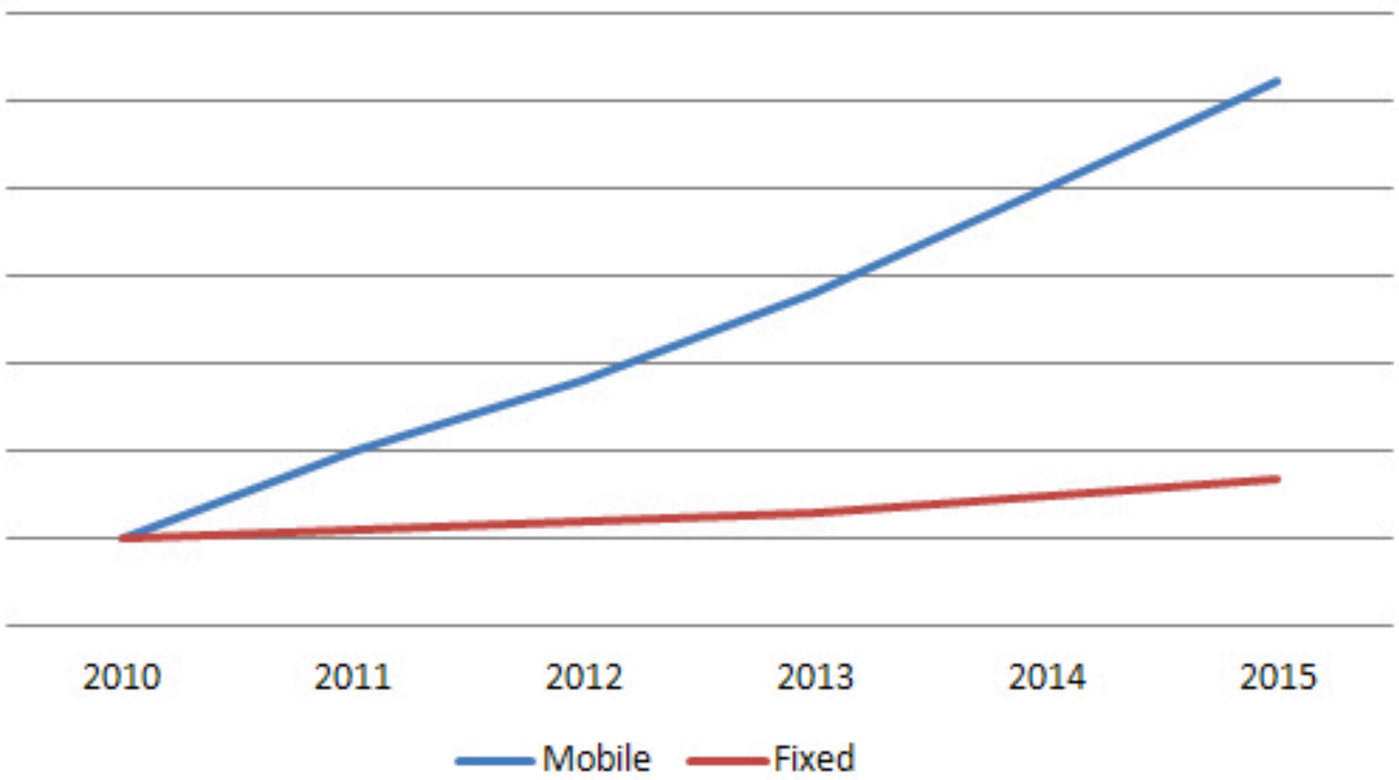
Mobile internet

or mobile world victory

Mobile first

TCP/IP

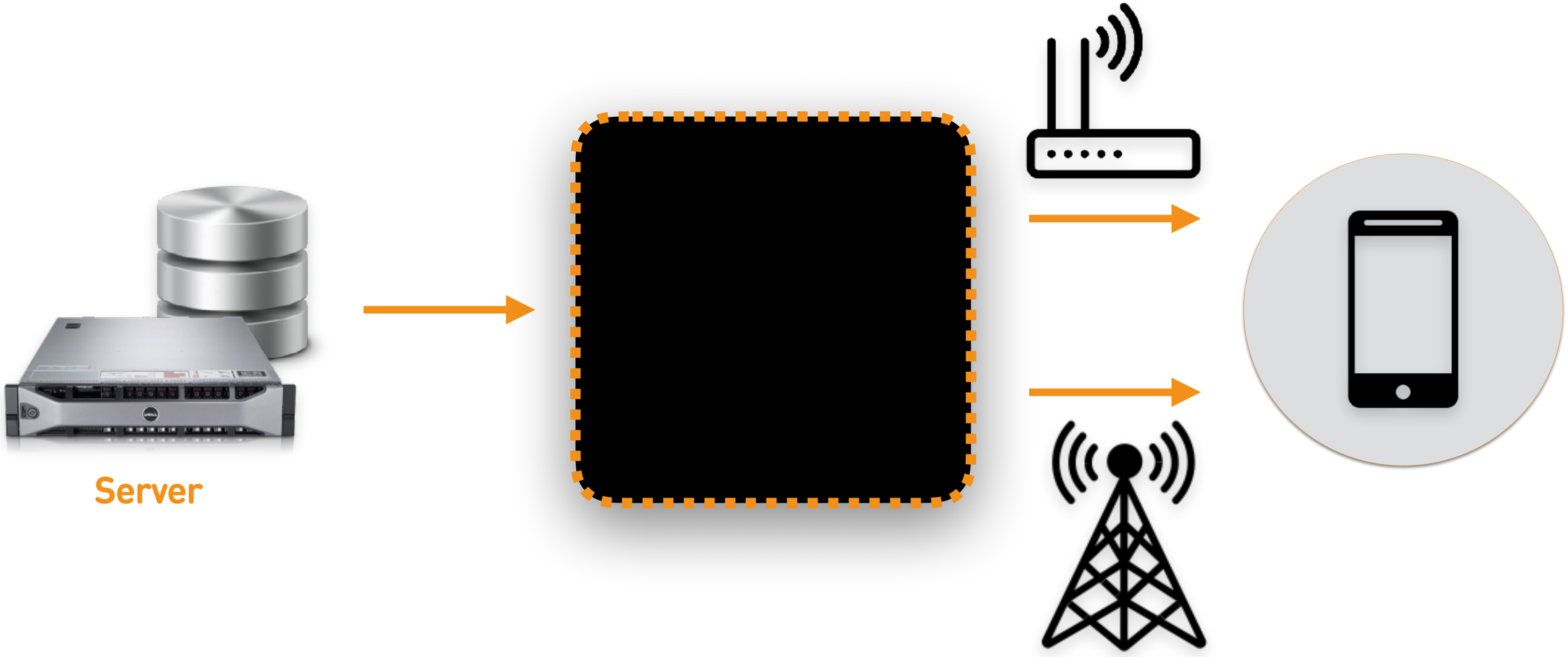
1974



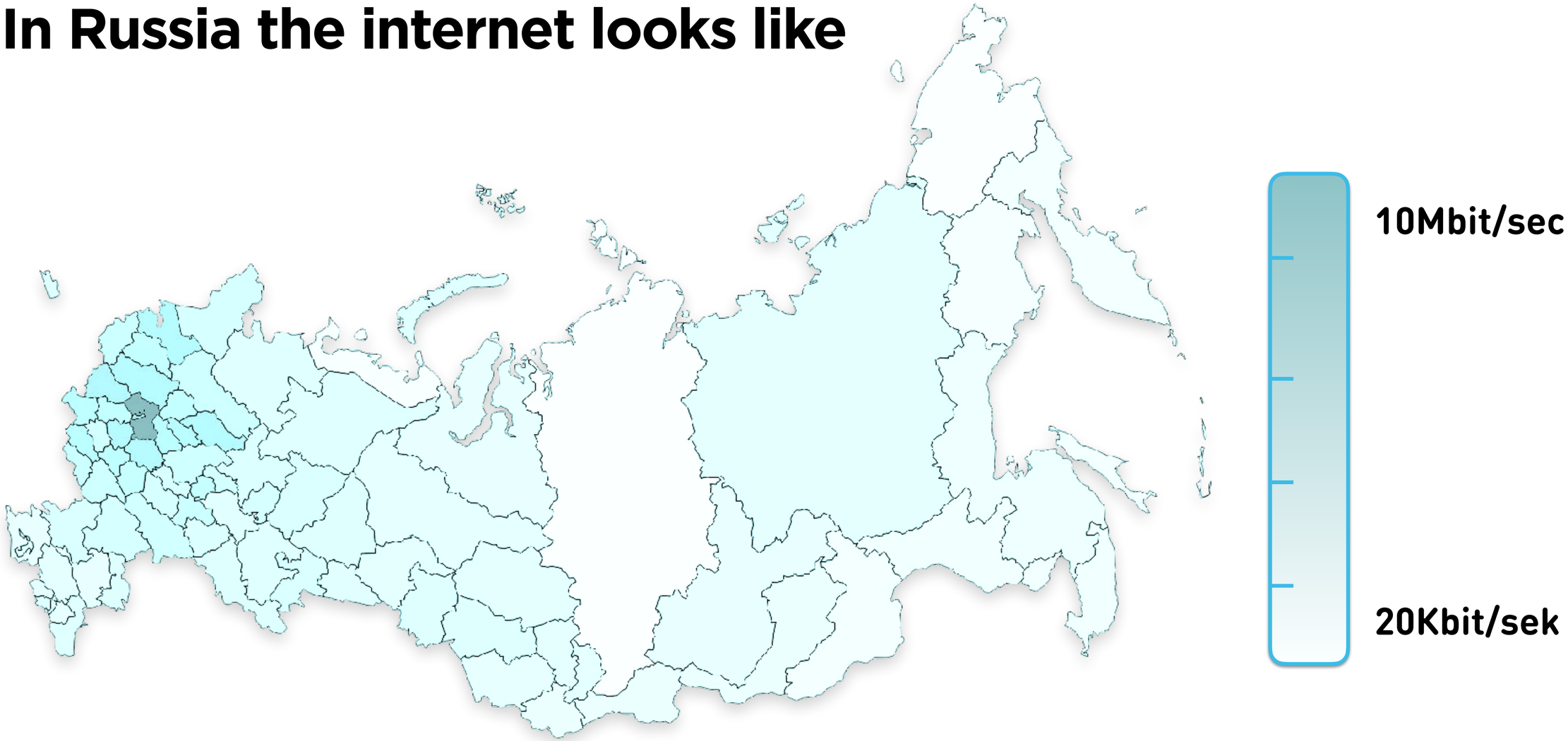
TCP/IP and mobile technologies

1974	1991	1998	1998	2008	2020
TCP	2G	WiFi	3G	4G	5G

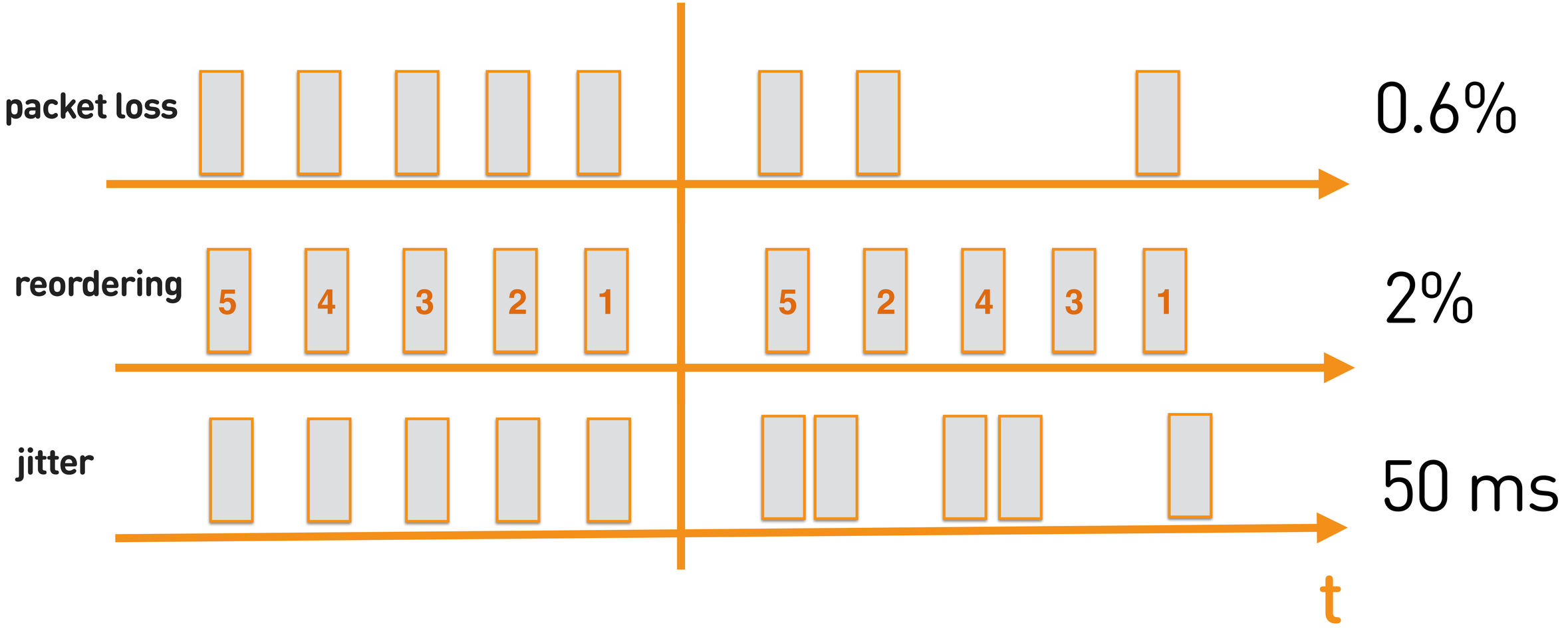
80% of cases the internet looks like



In Russia the internet looks like



In numbers the internet looks like



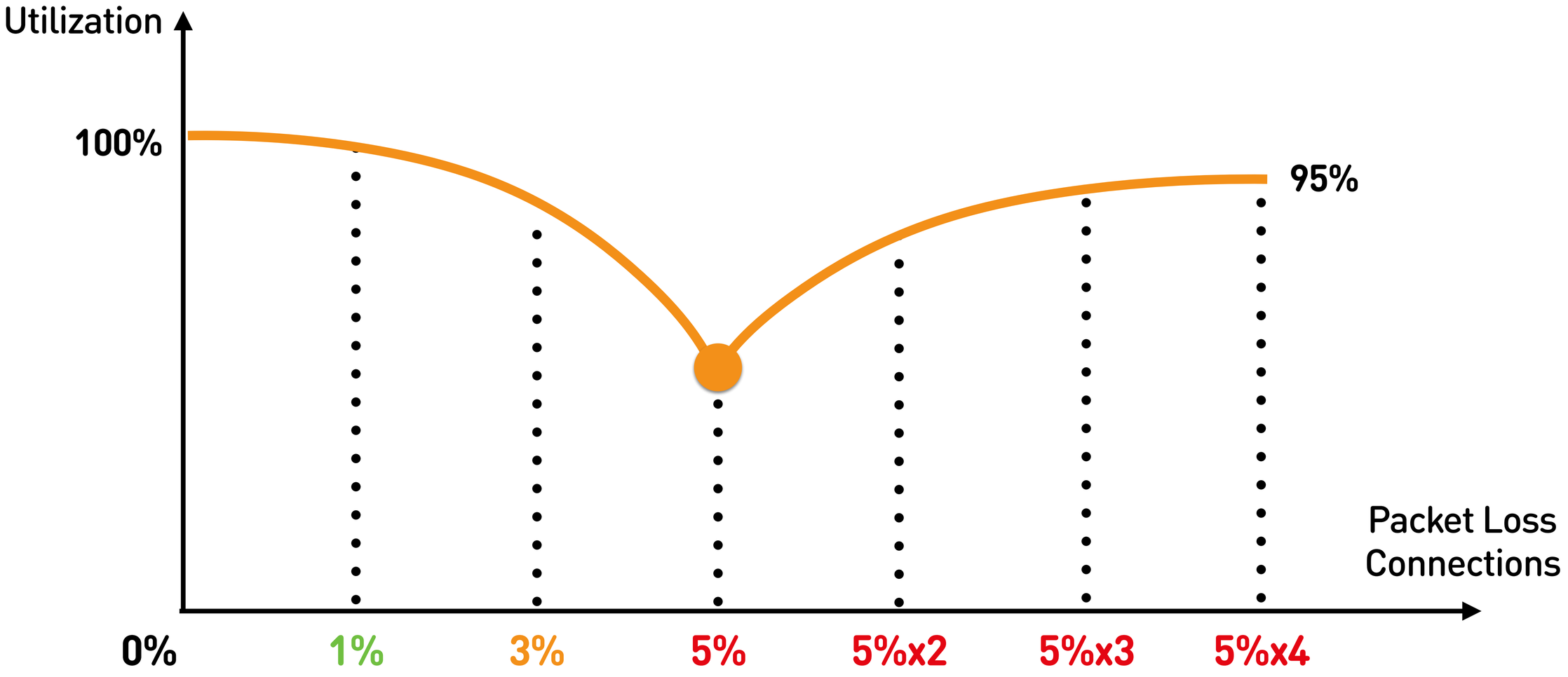
TCP inefficiency curve

it'll do

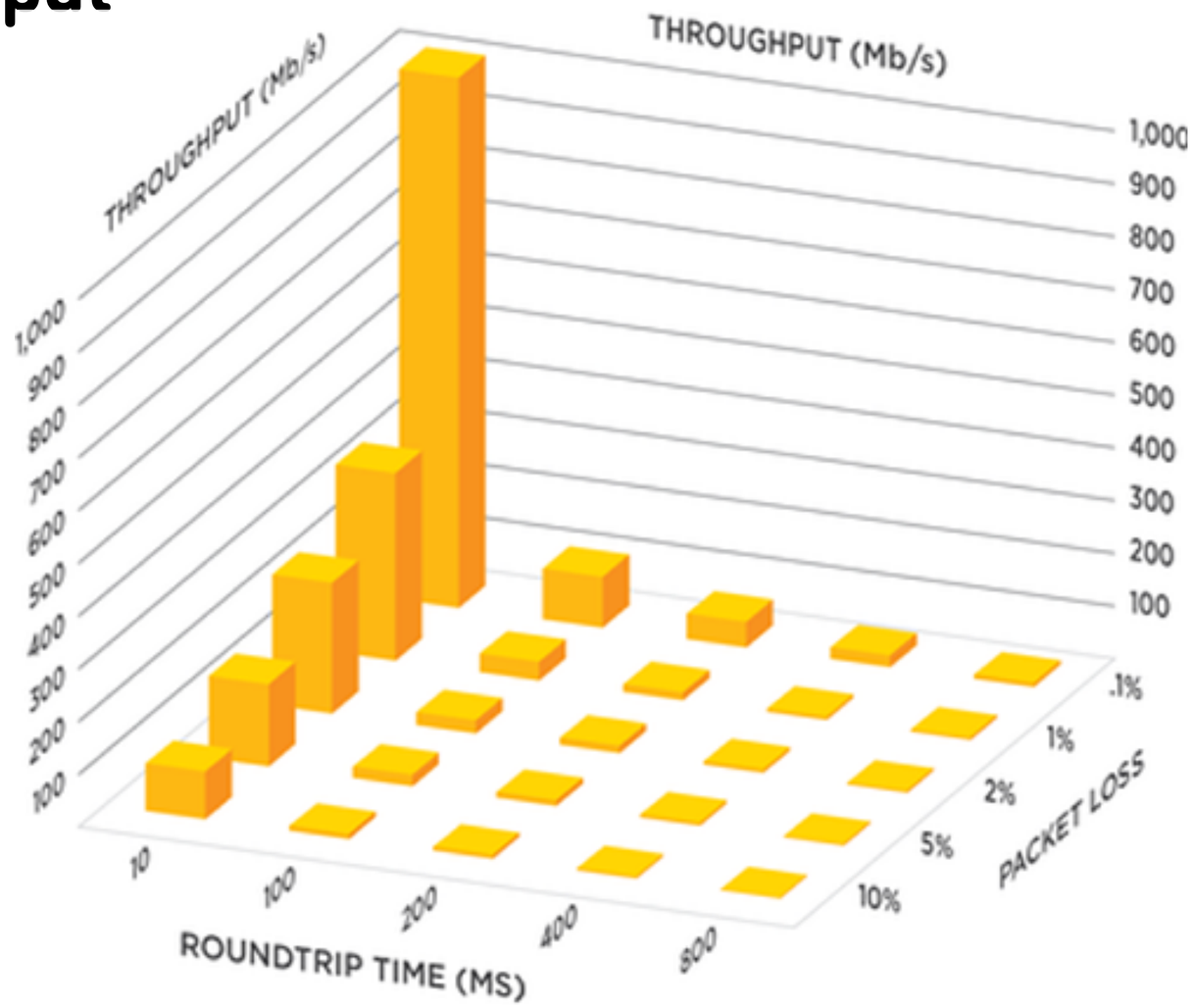
Entertaining tests

download	upload	RTT	loss	utilization
1.01 Mbit/s	0.96 Mbit/s	20 ms	0 %	99%
0.73 Mbit/s	0.75 Mbit/s	20 ms	**5 %**	74%
0.50 Mbit/s	0.39 Mbit/s	**220 ms**	5 %	45%
0.31**x2** Mbit/s	0.31**x2** Mbit/s	220 ms	5 %	62%

TCP inefficiency curve



TCP throughput



! until your packet loss 0% and RTT <10ms everything is OK

What happened?

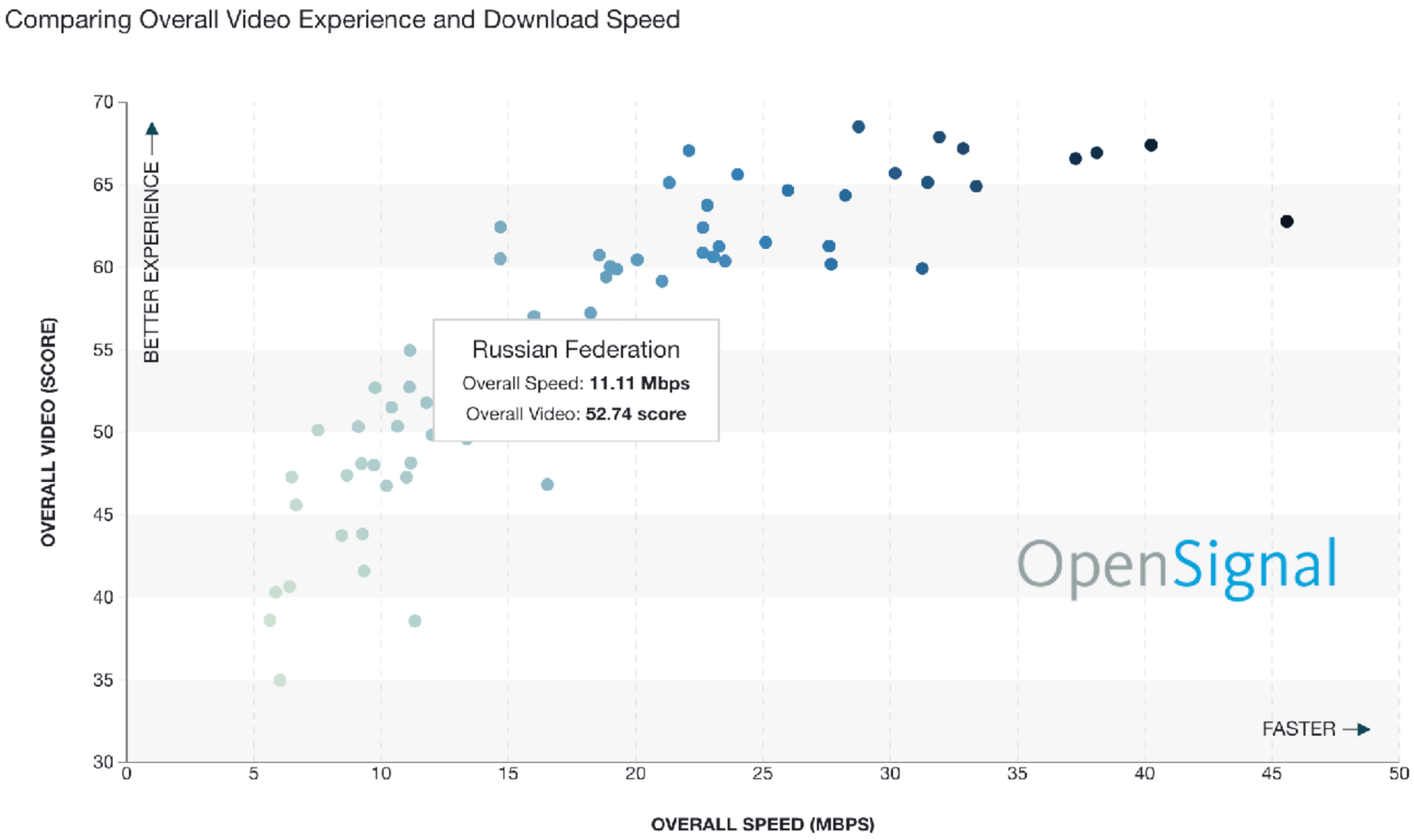
1. TCP invented in the 70s in the era of wired internet
2. Mobile networks has high packet loss, jitter, reordering
3. Dynamically changing network conditions
4. Asymmetric channel, IP migration

! This is usually ignored

Dependency between speed and engagement

do well and will work

Video consumption depends on network speed



https://opensignal.com/reports/2018/09/state-of-mobile-video

Mobile networks

mts.arm
>3% packet loss
~600ms RTT

1.1 Mbit/s

avg bandwidth

0.6 %

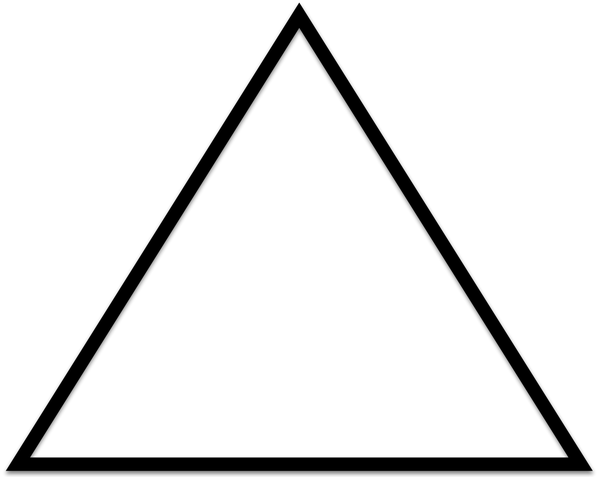
avg packet loss

250 ms

avg RTT

The major challenges of the report

content consumption
depends on network speed



poor mobile
internet

non optimal
protocols

What awaits us today

or what is not waiting for us today

What awaits us today?

1 Part1: TCP tuning in mobile networks

- puzzle
- connection establishment
- congestion control
- HTTP 2.0
- TCP troubles: head-of-line blocking, buffer bloat, IPMigration

What awaits us today?

2

Part2: QUIC
- transport protocols over UDP in user space
- IPMigration
- UDP performance in Linux

What awaits us today?

3

Part3: selfmade UDP protocols

- OKMP - protocol for video streaming
- UT2 - protocol for impulse data transferring
    - scheduler
    - TLP & FEC

What awaits us today?

4 Part4: Future of applications stack
- merging of security and transport layer
- multicast

What is not waiting for us today?

Application	High-level APIs, including resource sharing, remote file access
Presentation	Translation of data between a networking service and an application
Session	Managing communication sessions
Transport	Reliable transmission of data segments between points on a network
Network	Structuring and managing a multi-node network
Data link	Reliable transmission of data frames between two nodes connected by a physic
Physical	Transmission and reception of raw bit streams over a physical medium

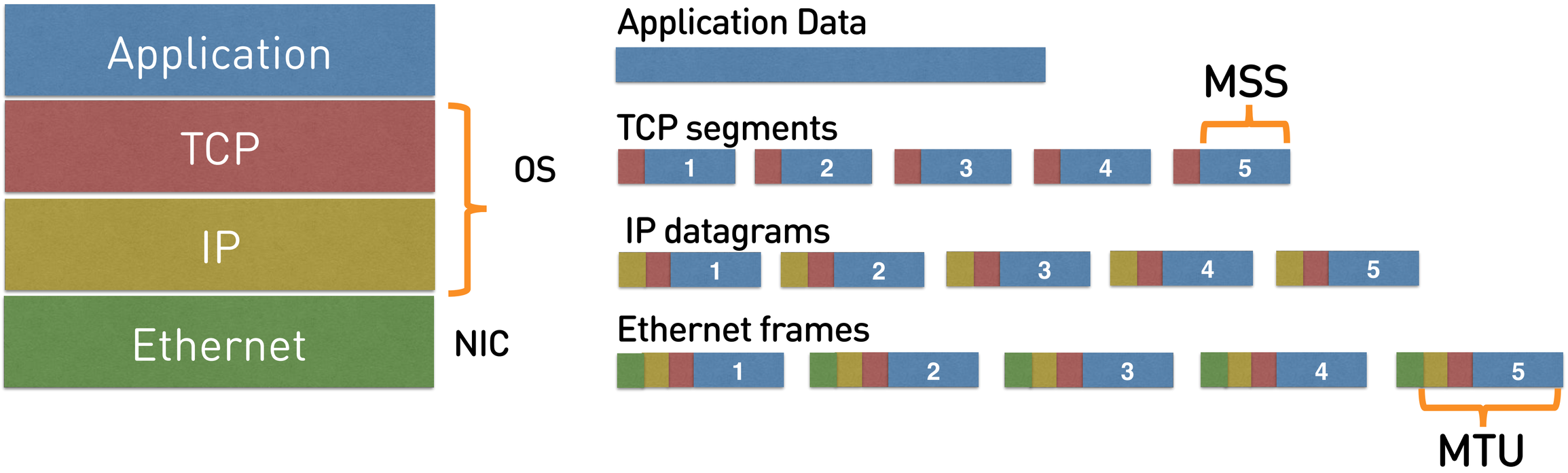
$$B(p) = \min\left(\frac{W_{\max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1+32p^2)}\right)$$

! admins don’t worry about mobile clients

Part1: TCP

introduction

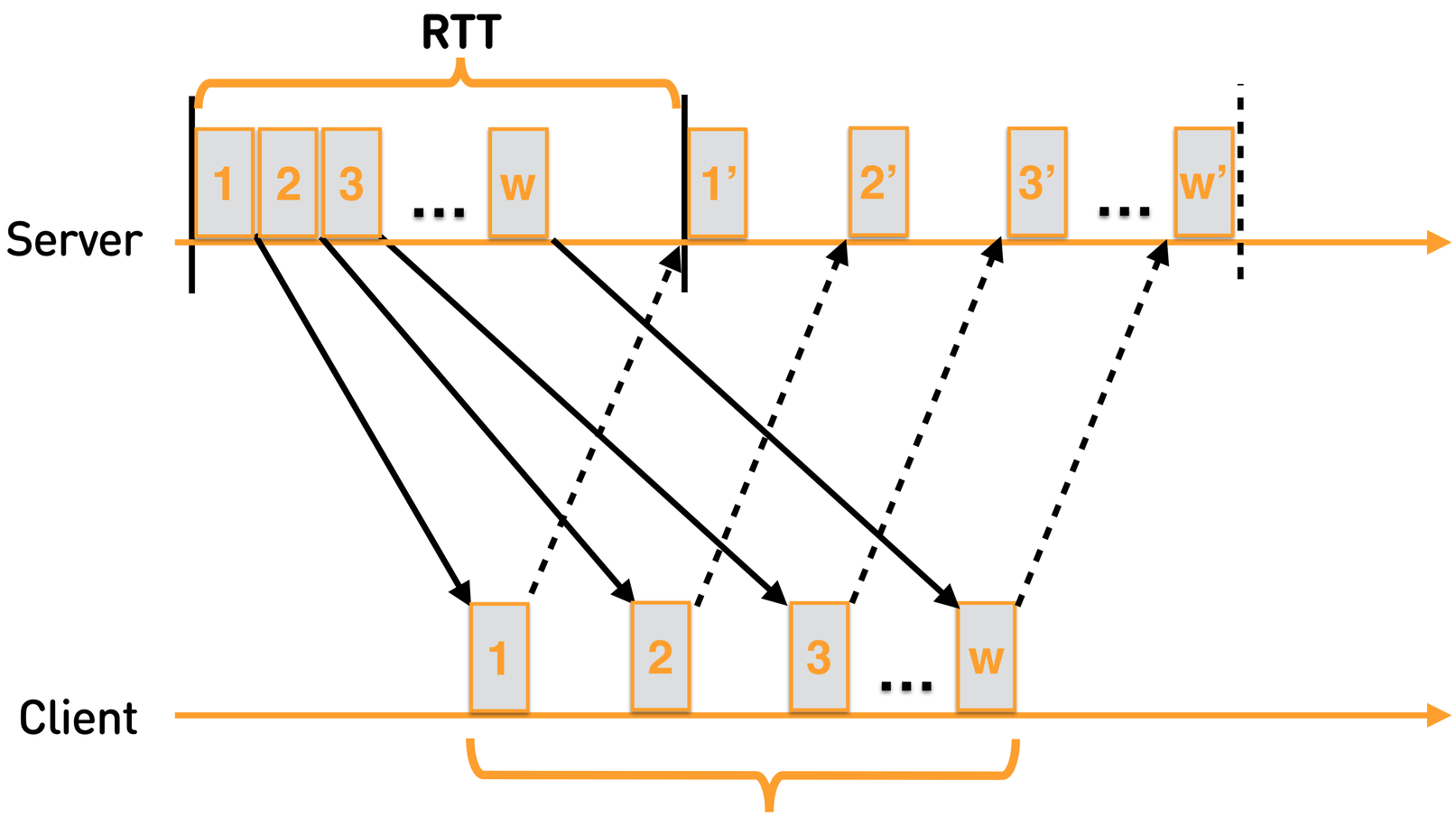
Data transmission and TCP



Goals of TCP:

1. data consistency
2. transmission result notification

A model of TCP data transmission



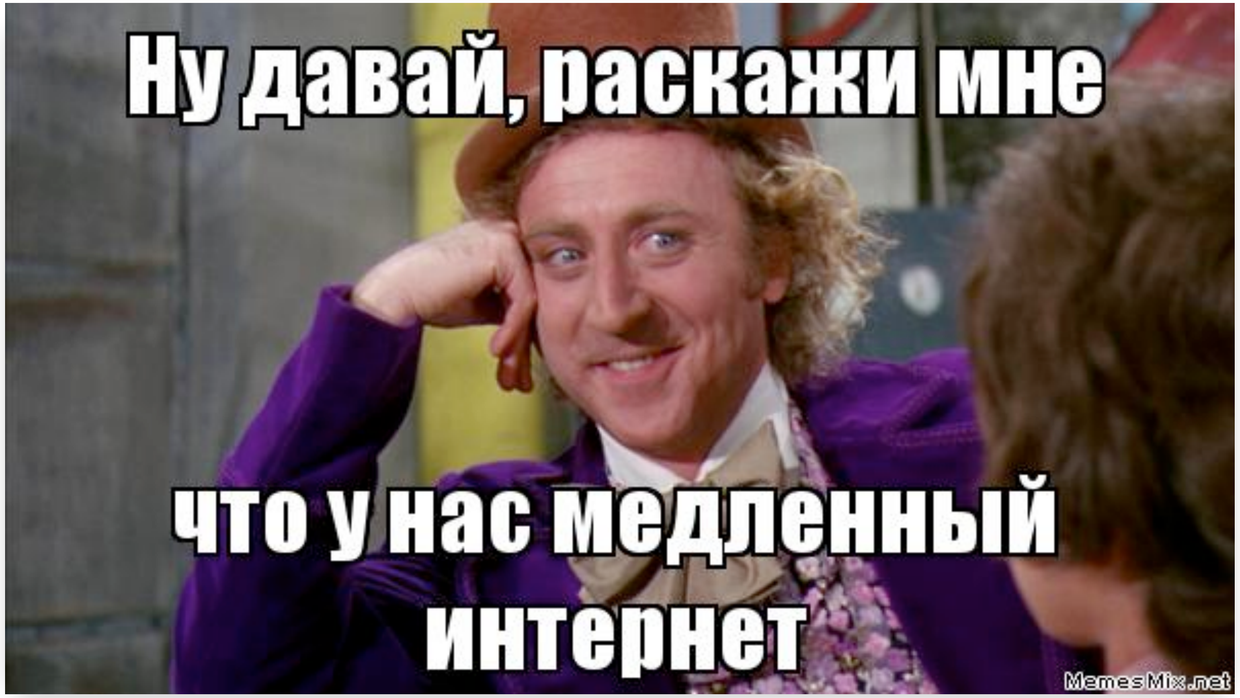
Bandwidth = Packets * Size / sec

Part1: TCP

1 Part1: TCP tuning in mobile networks

- puzzle

- connection establishment
- congestion control
- HTTP 2.0
- TCP troubles: head-of-line blocking, buffer bloat, IPMigration



Puzzle: slow internet

real task from wild world

Puzzle: you have slow internet

250 ms

round trip time

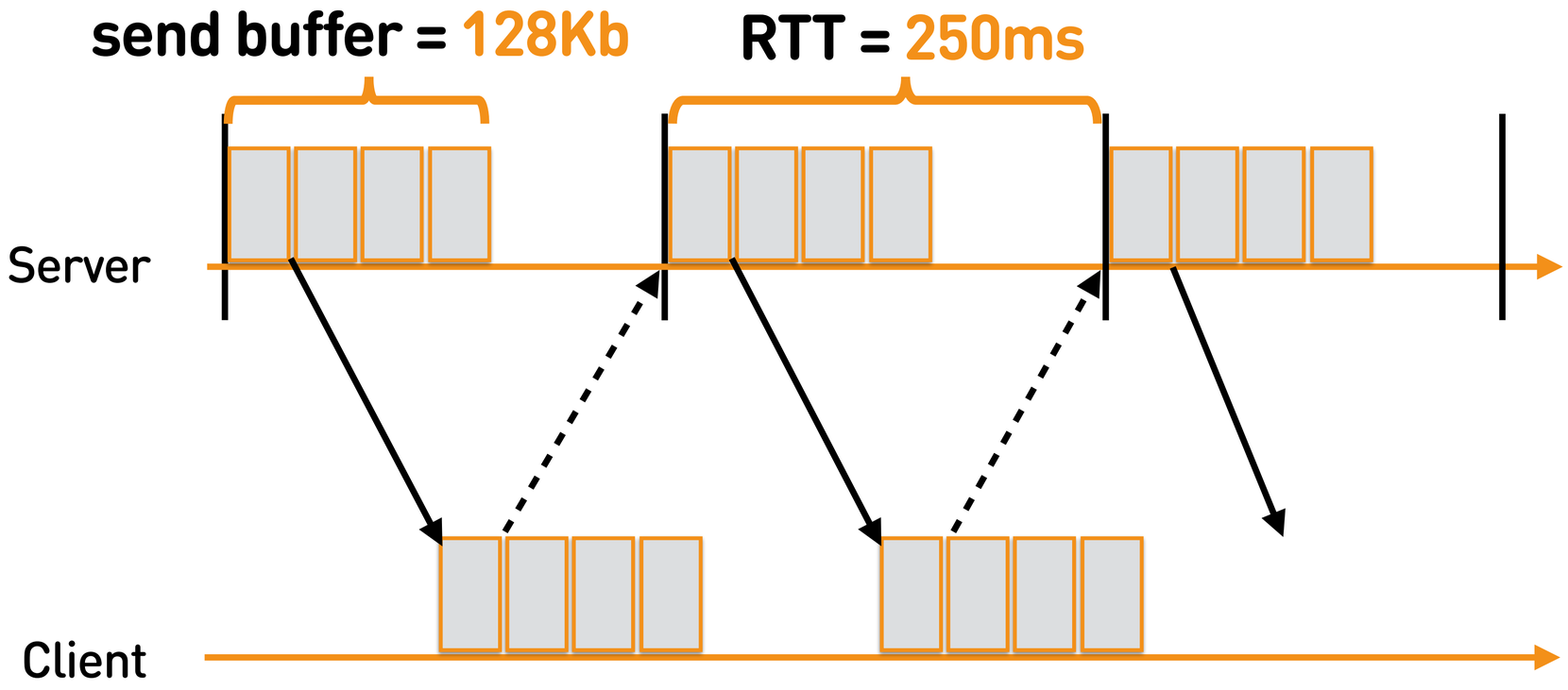
400 Mbit/s

bandwidth

Full HD 1080

???

TCP send/recv buffer



bandwidth = buffer size / RTT

bandwidth = 128 * 8 / 0.25 = 4 Mbit/sec

1%

Example of increase buffer size

	64Kb	128Kb	256Kb
10Mb	1640ms	1100ms	733ms
12Mb	1780ms	1200ms	830ms
bandwidth	54Mbit/sec	80Mbit/sec	115Mbit/sec

! Check net.ipv4.tcp_wmem and SO_RCVBUF

The history of TCP

or a jump into the leaving train

Timeline of TCP - RFC

1974	2013	2014	2016	2018
TCP	draft-dukkipati-tcpm-tcp-loss-probe-01 TLP	RFC 7413 TFO	draft-ietf-tcpm-rack-03.html RACK	draft-cardwell-iccrg-bbr-congestion-control-00 BBR

? Who knows why so many changes have gone now?

Part1: TCP

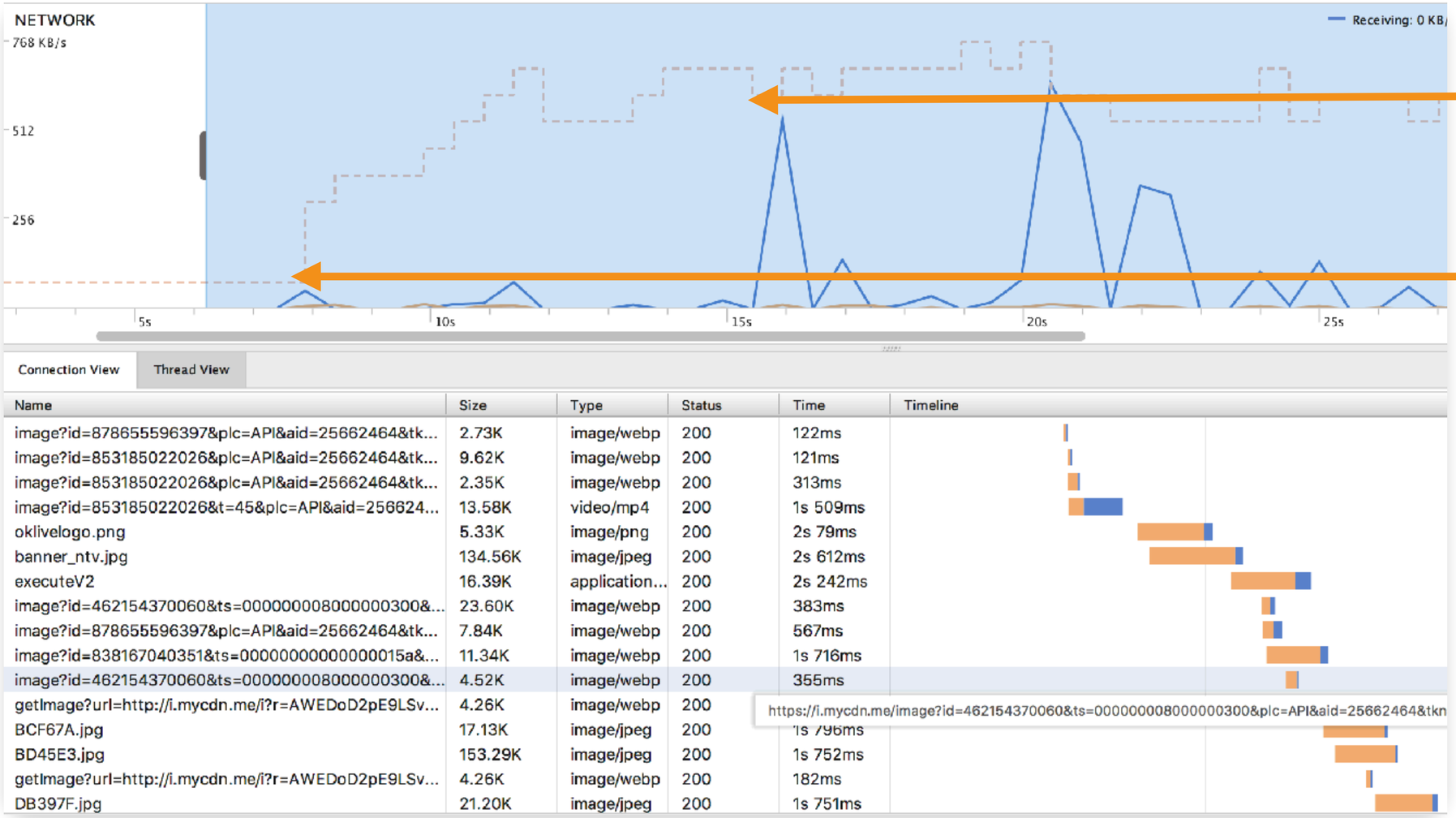
1 Part1: TCP tuning in mobile networks

- puzzle
- **connection establishment**
- congestion control
- HTTP 2.0
- TCP troubles: head-of-line blocking, buffer bloat, IPMigration

Connection establishment

or Latency numbers in mobile networks

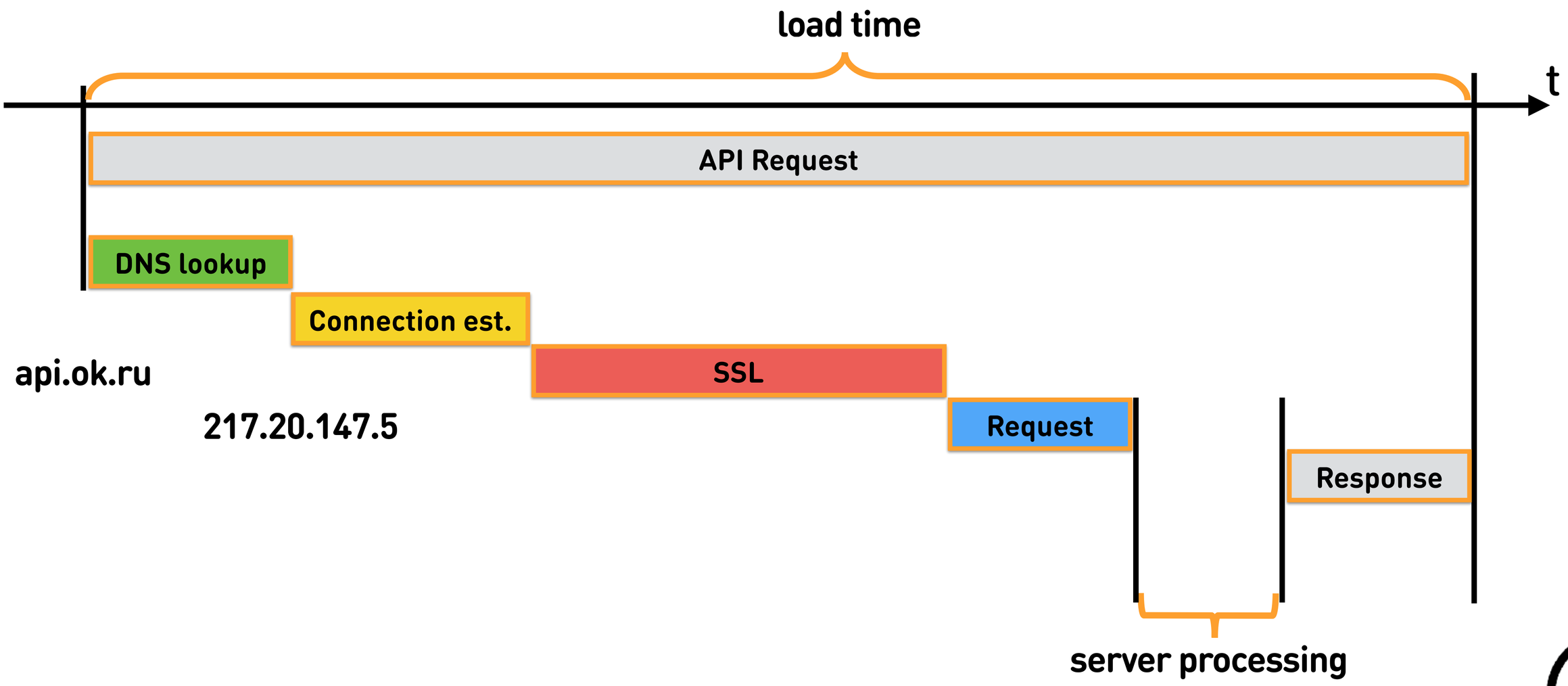
37 Android network profiling



Data

Open Connections

Stages of execution of API request



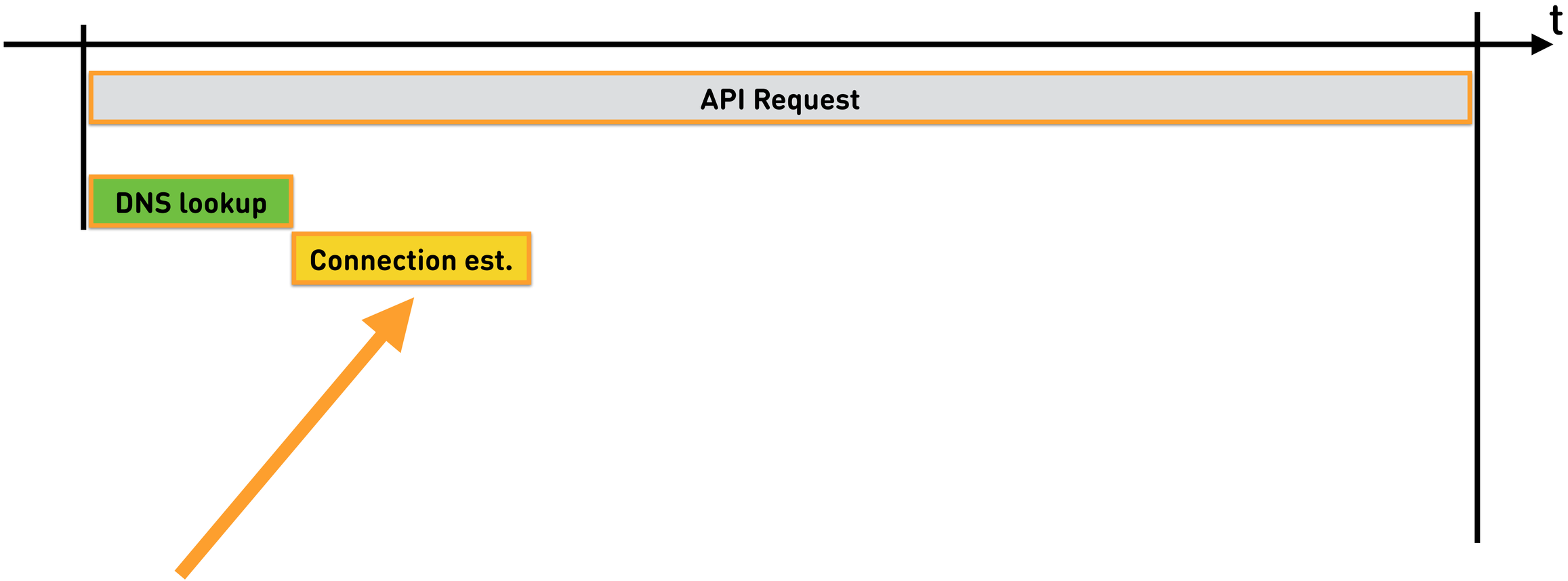
Latency numbers in mobile networks

	RTT/ms	3G	4G
DNS lookup	1 RTT	200ms	100ms
TCP handshake	1 RTT	200ms	100ms
TLS handshake	1-2 RTT	200-400ms	100-200ms
HTTP request	1 RTT	200ms	100ms
Result	4-5 RTT	1-3.5 sec	450-700 ms

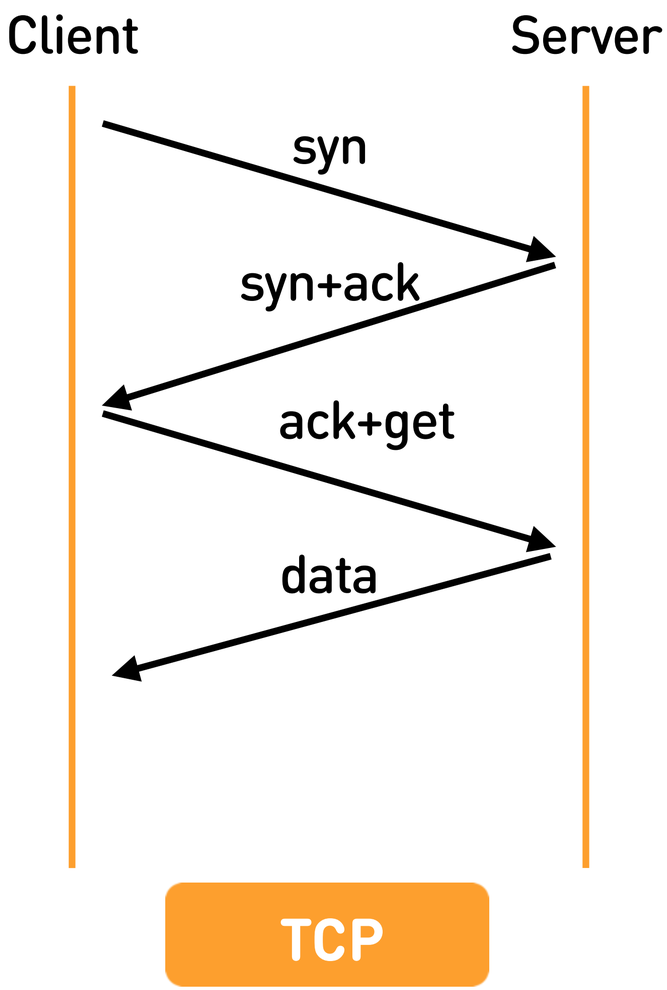
Connection establishment

step by step

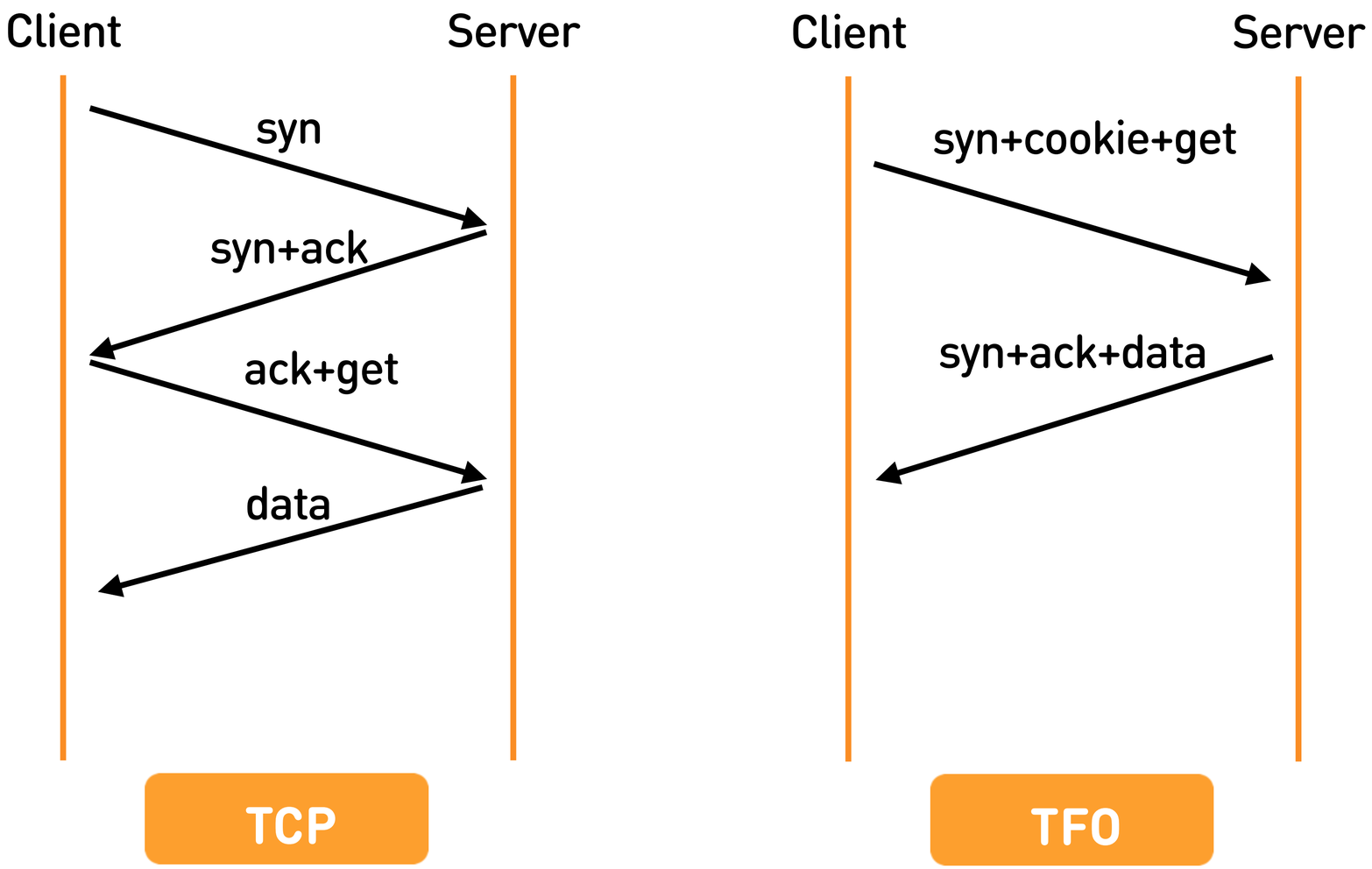
TCP connection establishment



TCP handshake



TFO: TCP Fast Open - RFC 7413



TFO: server

```
/* create the socket */
fd = socket();

/* connect and send out some data */
sendto(fd, buffer, buf_len, MSG_FASTOPEN, ...);

/* write more data */
send(fd, buf, size);
```

	Version	Check
Linux	kernel 4.1+	net.ipv4.tcp_fastopen = 3

! Check tcp_fastopen

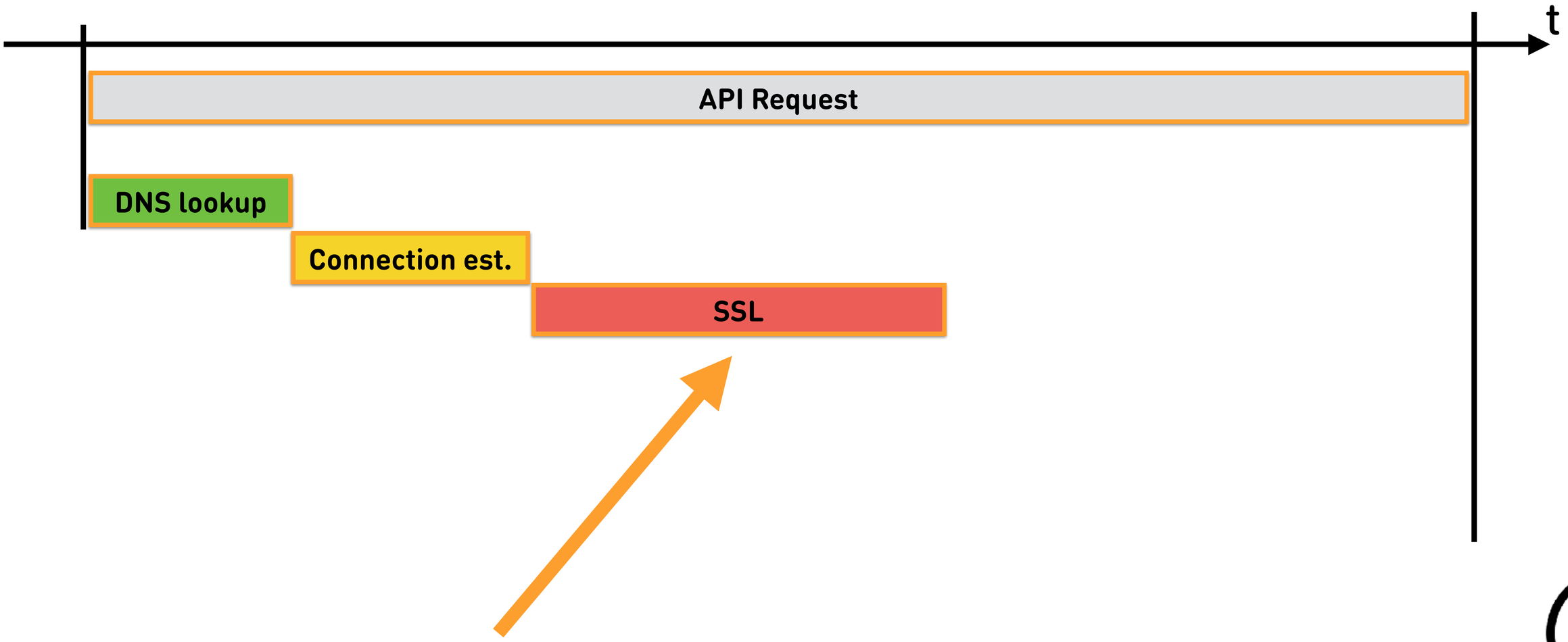
TFO: client

	Version	Check
Android	9+ (8.1.0 еще нет)	tcp_fastopen = 1
iOS	9+	documentation
Web	Chrome, edge	+

-10%

- 1 RTT

TLS connection establishment



TLS or what's taking so long?

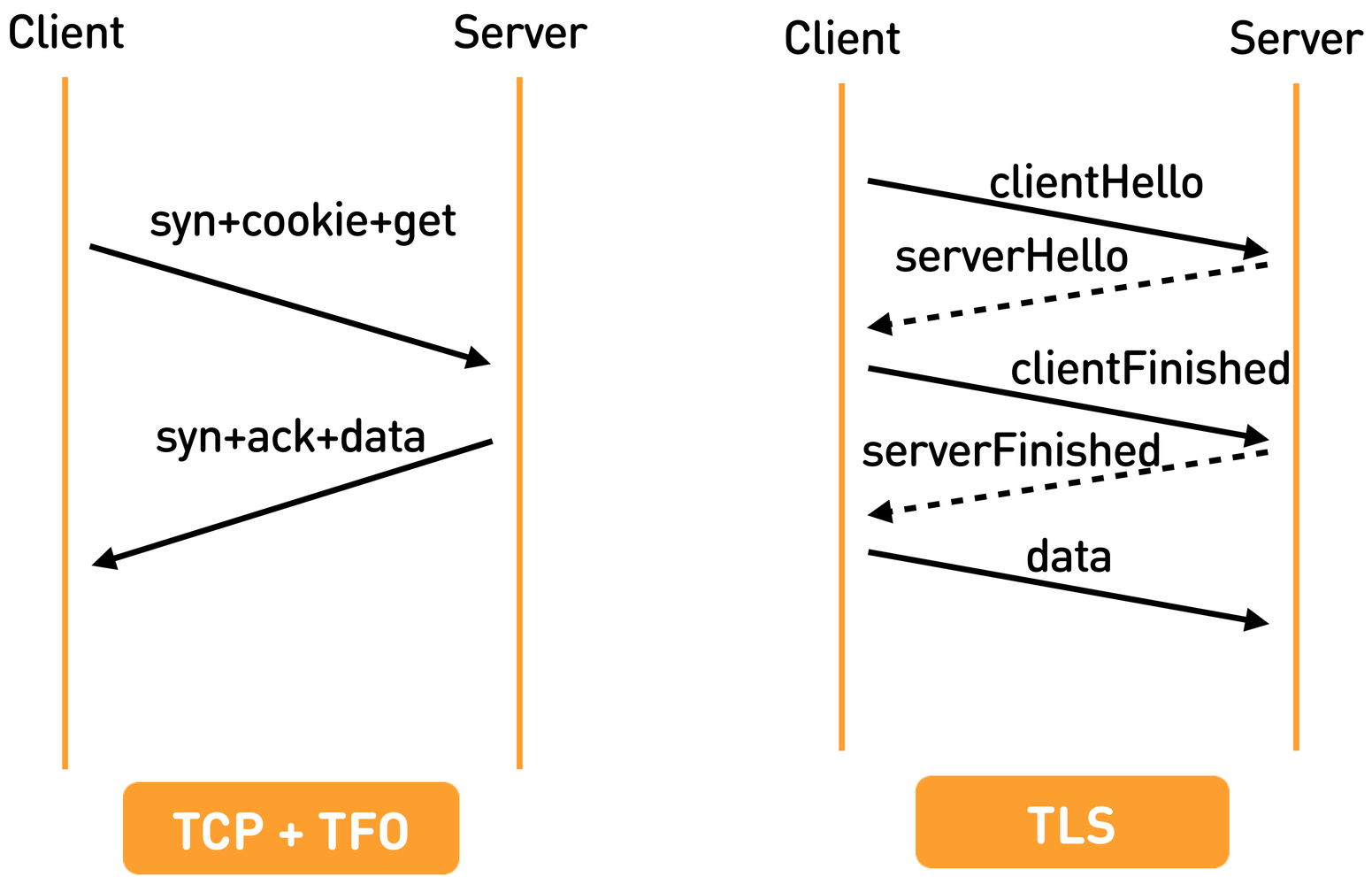
```
$ ping google.com
64 bytes from 173.194.73.139: icmp_seq=5 ttl=44 time=211.847 ms
round-trip min/avg/max/stddev = 209.471/220.238/266.304/19.062 ms
```

RTT = 220ms

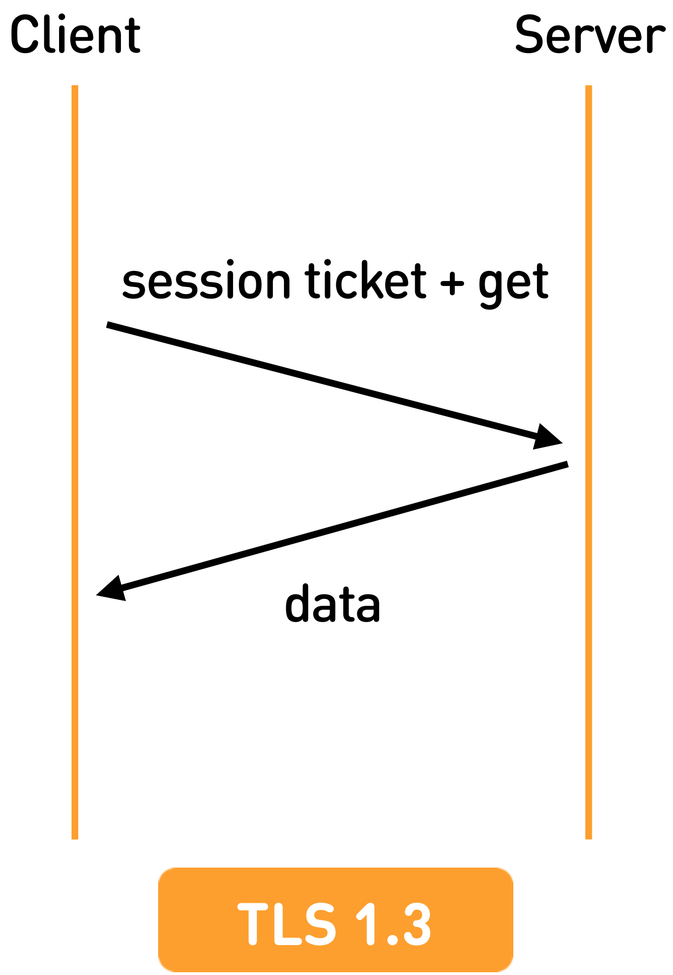
```
$ curl -o /dev/null -w "HTTP time taken: %{time_connect}\nHTTPS time taken: %{time_appconnect}\n" -s https://www.google.com
HTTP time taken: 0,231
HTTPS time taken: 0,797
```

HTTP = 230ms
HTTPS = 800ms

TCP + TLS connection establishment



zeroRTT TLS 1.3




```
/* NGINX >=1.13.0 */
ssl_protocols TLSv1.1 TLSv1.2 TLSv1.3;
```

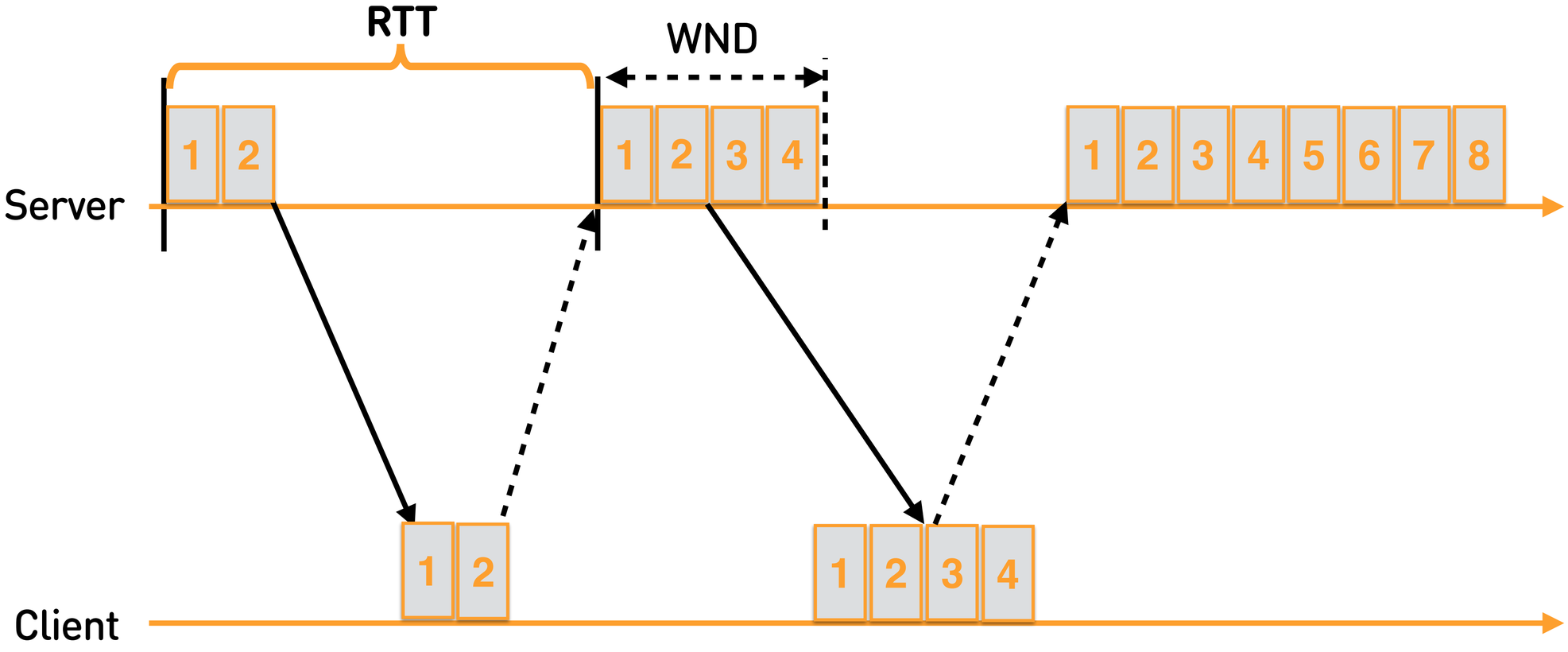
Fizz

TLS 1.3 not enabled at clients

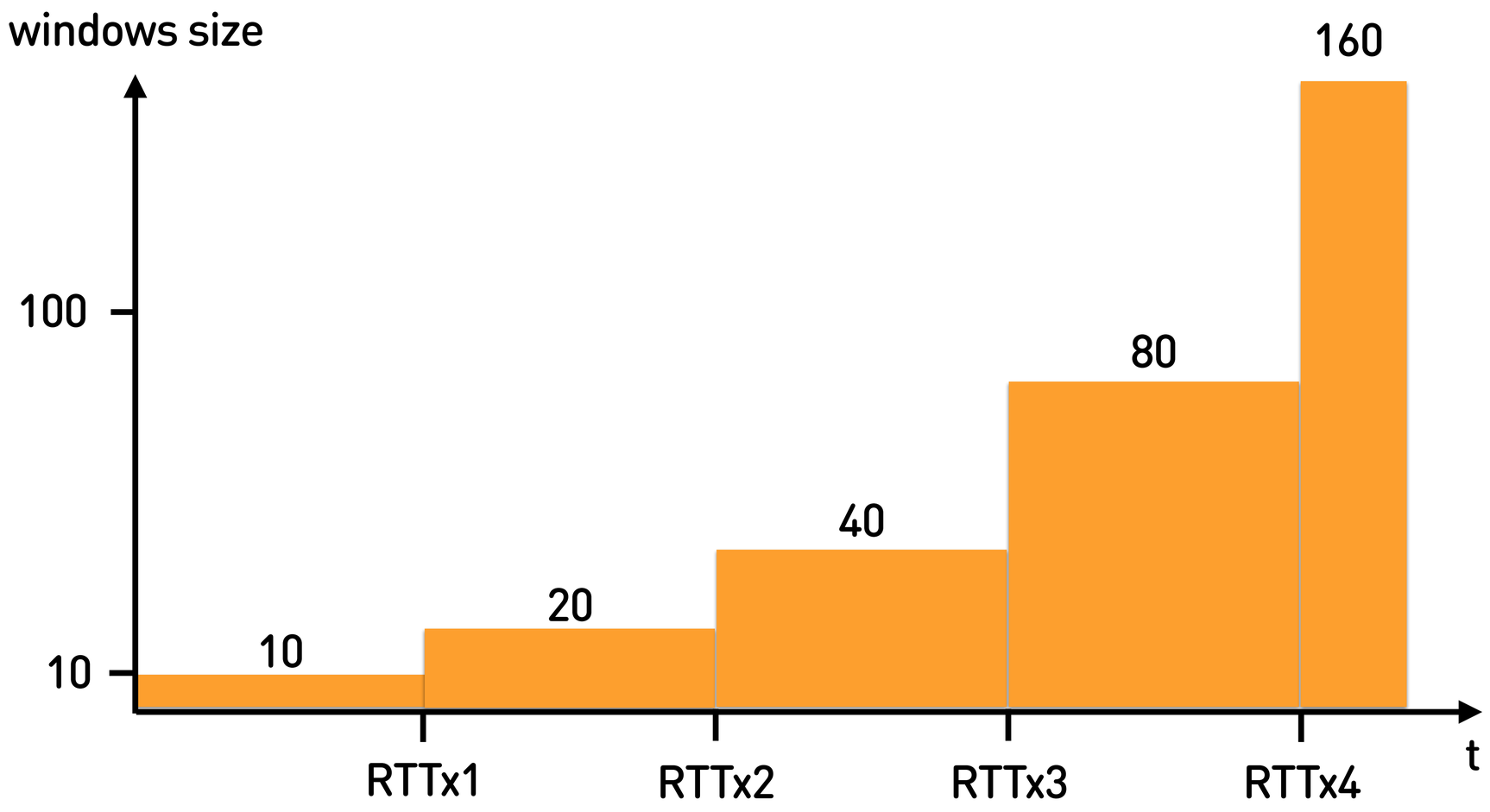
Warming up the connection

or congestion window

Congestion window



Slow-start



Connection establishment stages

and statistic

Mobile networks of OK users

mts.arm
>3% packet loss
~600ms RTT

1.1 Mbit/s

avg bandwidth

0.6 %

avg packet loss

300 ms

avg RTT

Connection establishment stages

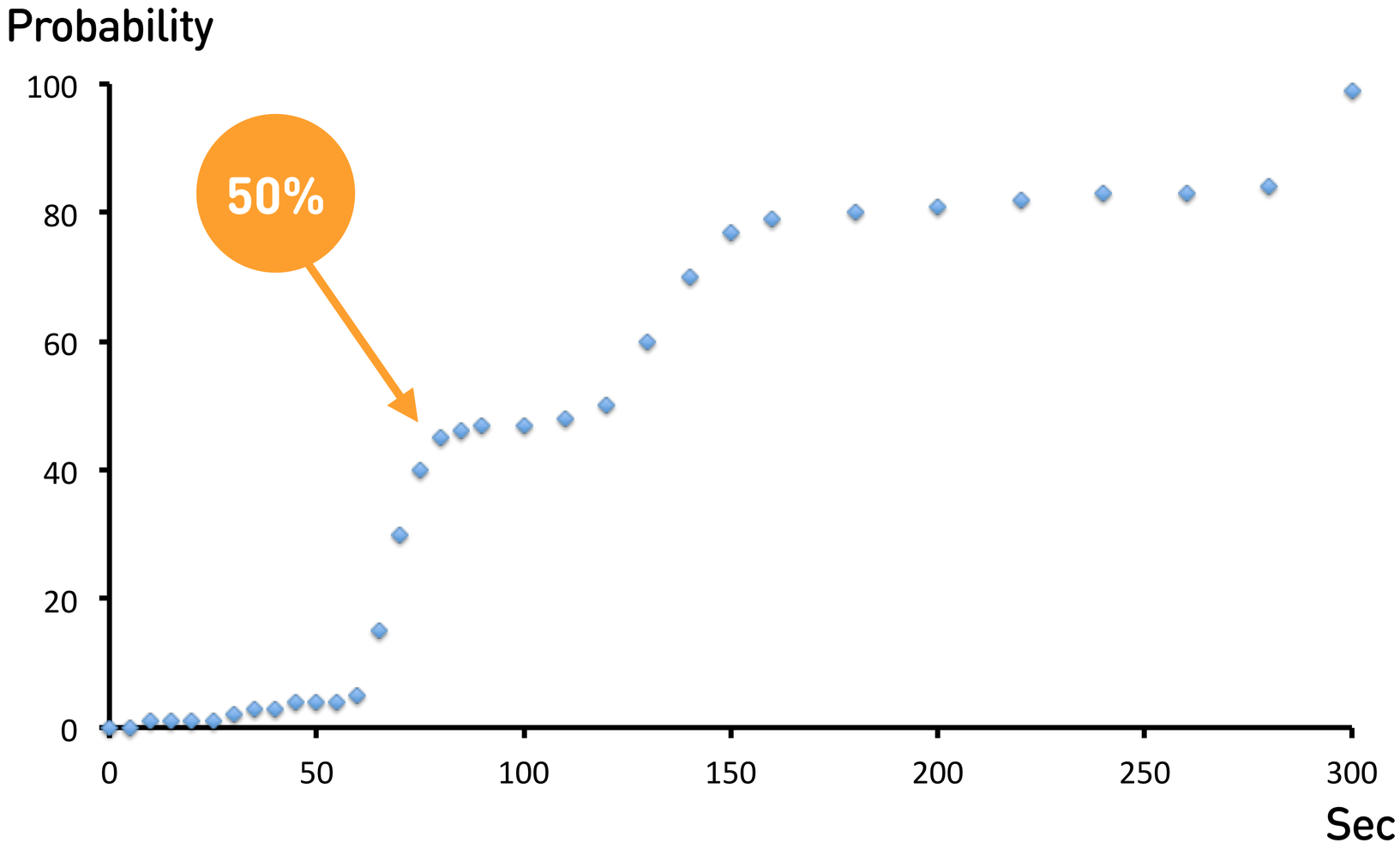
1. DNS resolve
2. connection establishment
3. TLS
4. slow start

! Reuse connections

Reuse of connections

underwater rocks

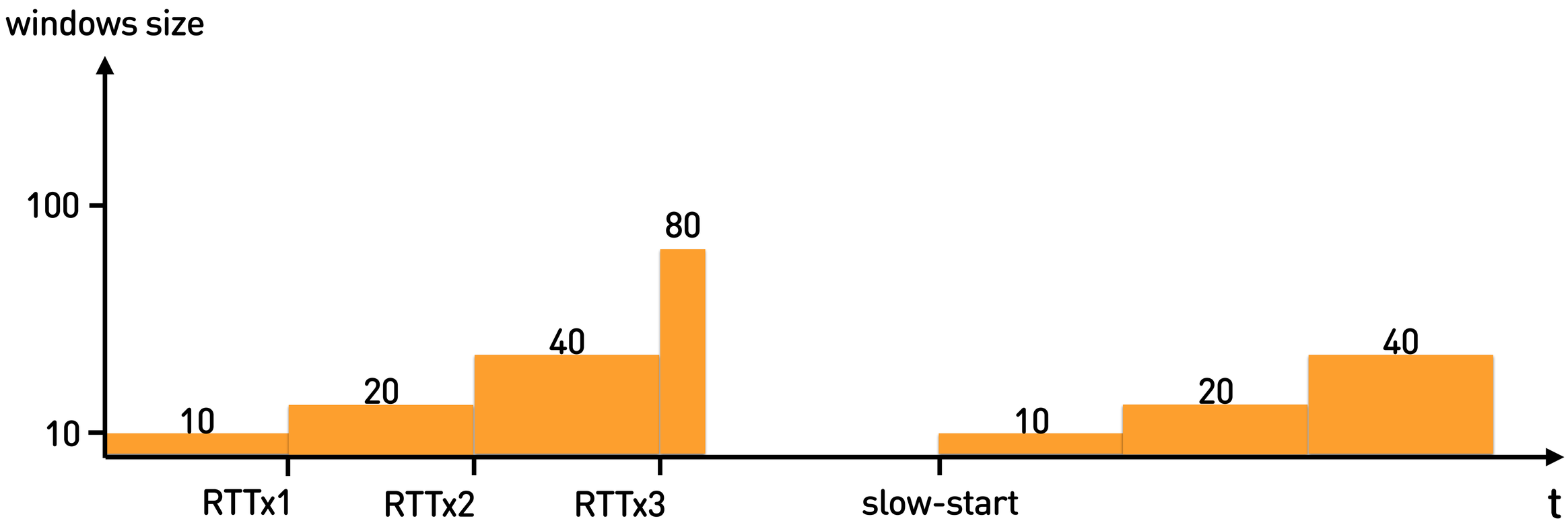
Reuse connections: NAT unbinding



ping-pong
30-60sec

! Use keep-alive

Reuse connections: Slow-Start Restart




```
/* disable slow-start restart */
sysctl -w net.ipv4.tcp_slow_start_after_idle=0
```

Part1: TCP

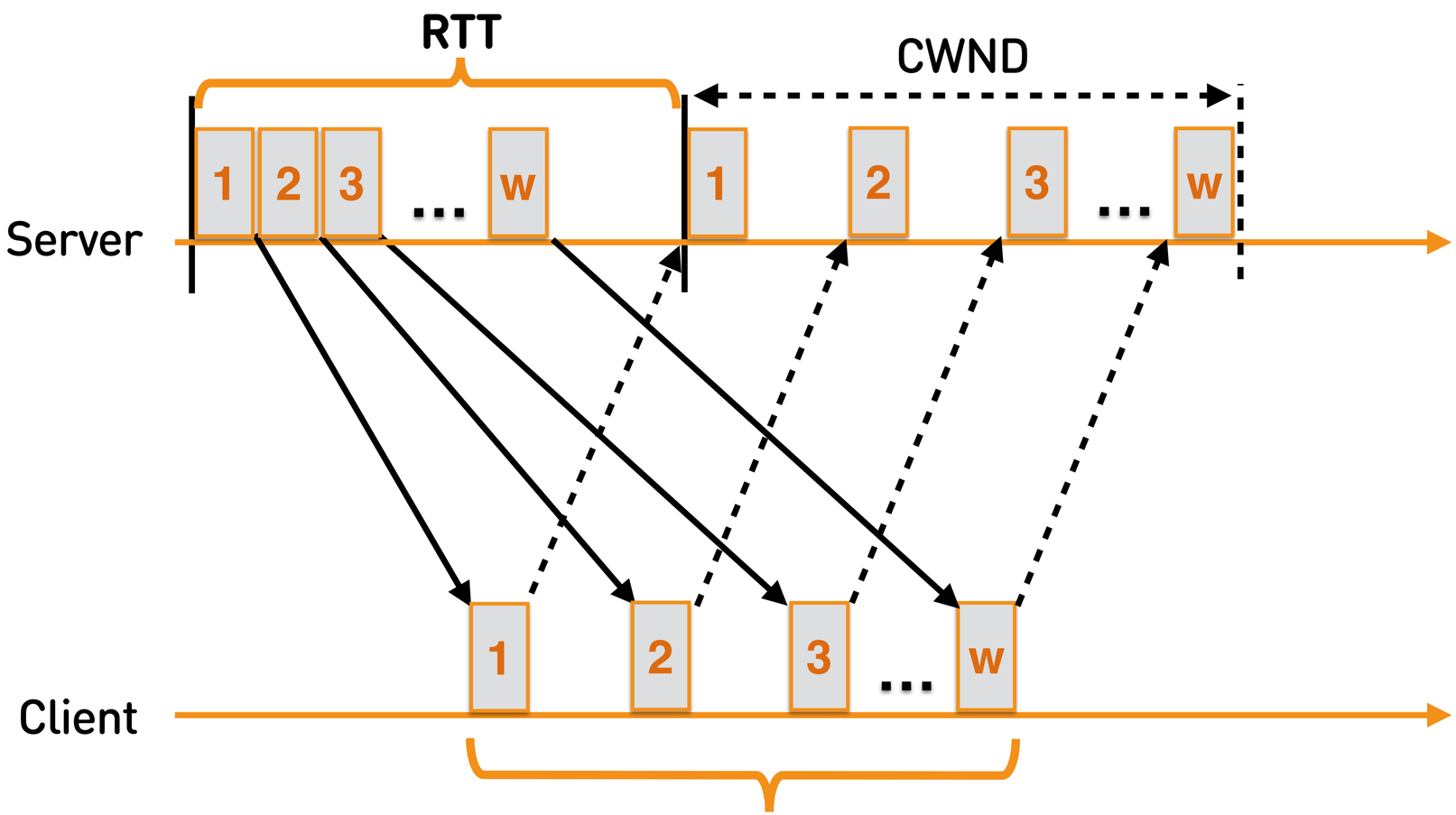
1 Part1: TCP tuning in mobile networks

- puzzle
- connection establishment
- **congestion control**
- HTTP 2.0
- TCP troubles: head-of-line blocking, buffer bloat, IPMigration

Congestion control

why do you need it?

Congestion control

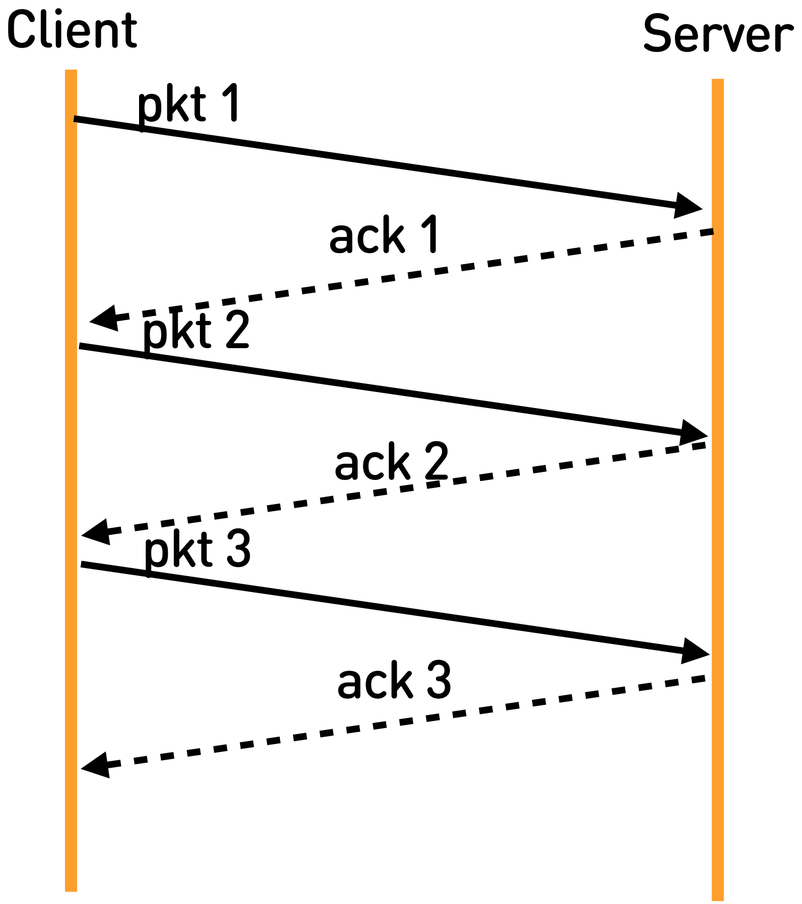


Bandwidth = Packets * MTU / sec

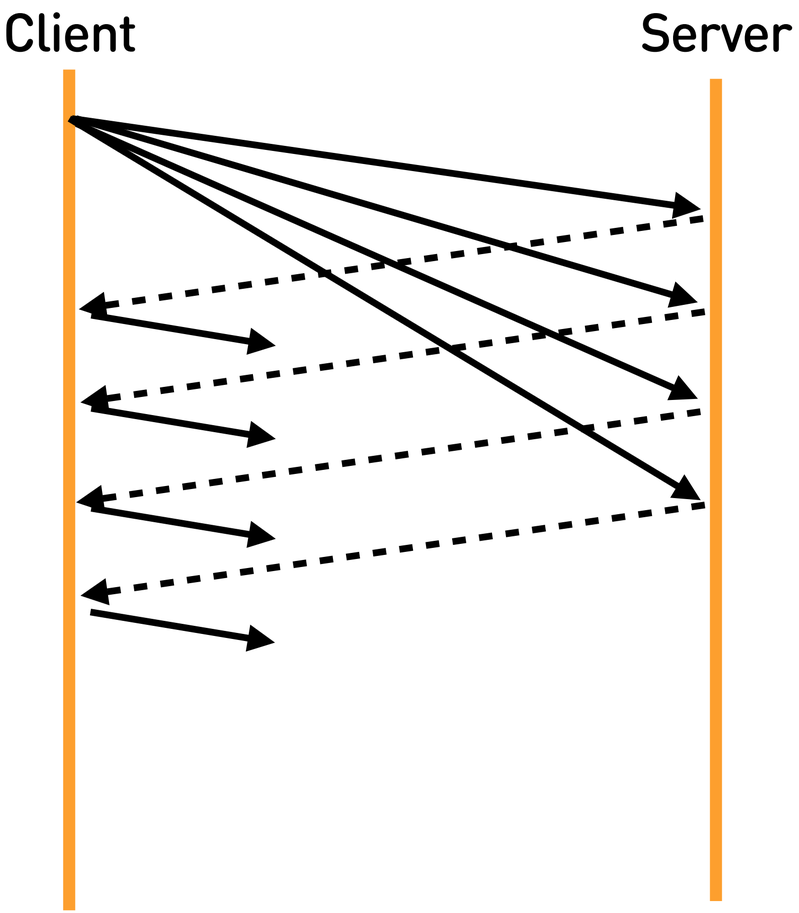
Congestion Control

Window via Feedback

TCP congestion window

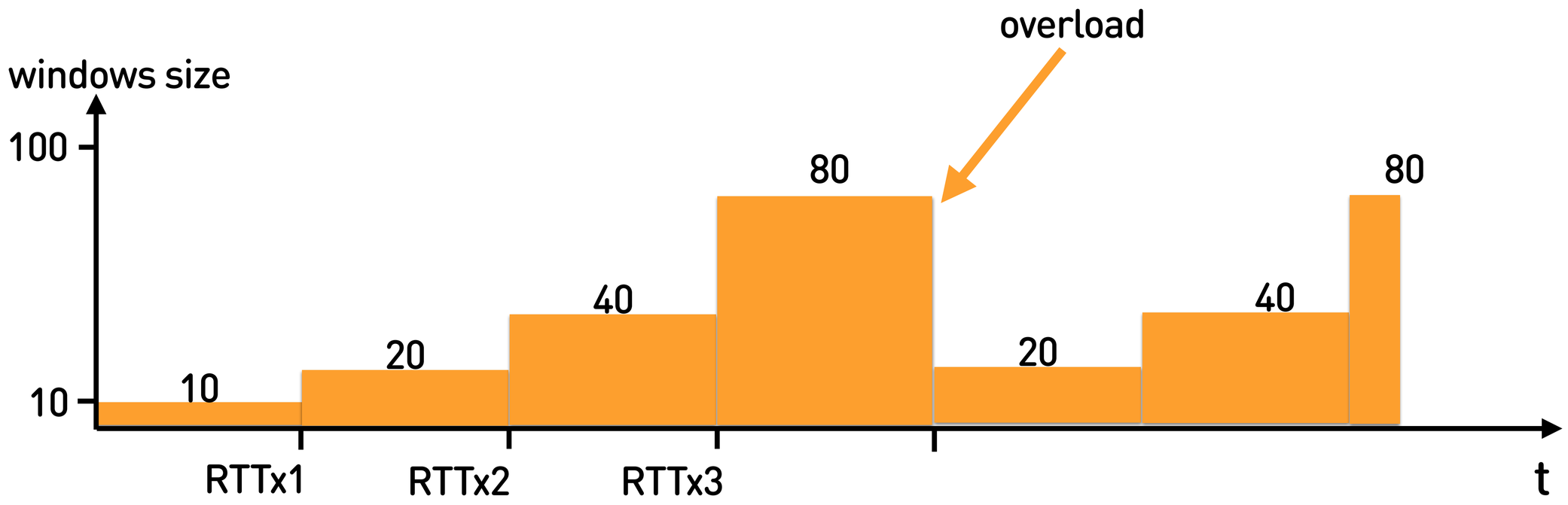


wnd=1

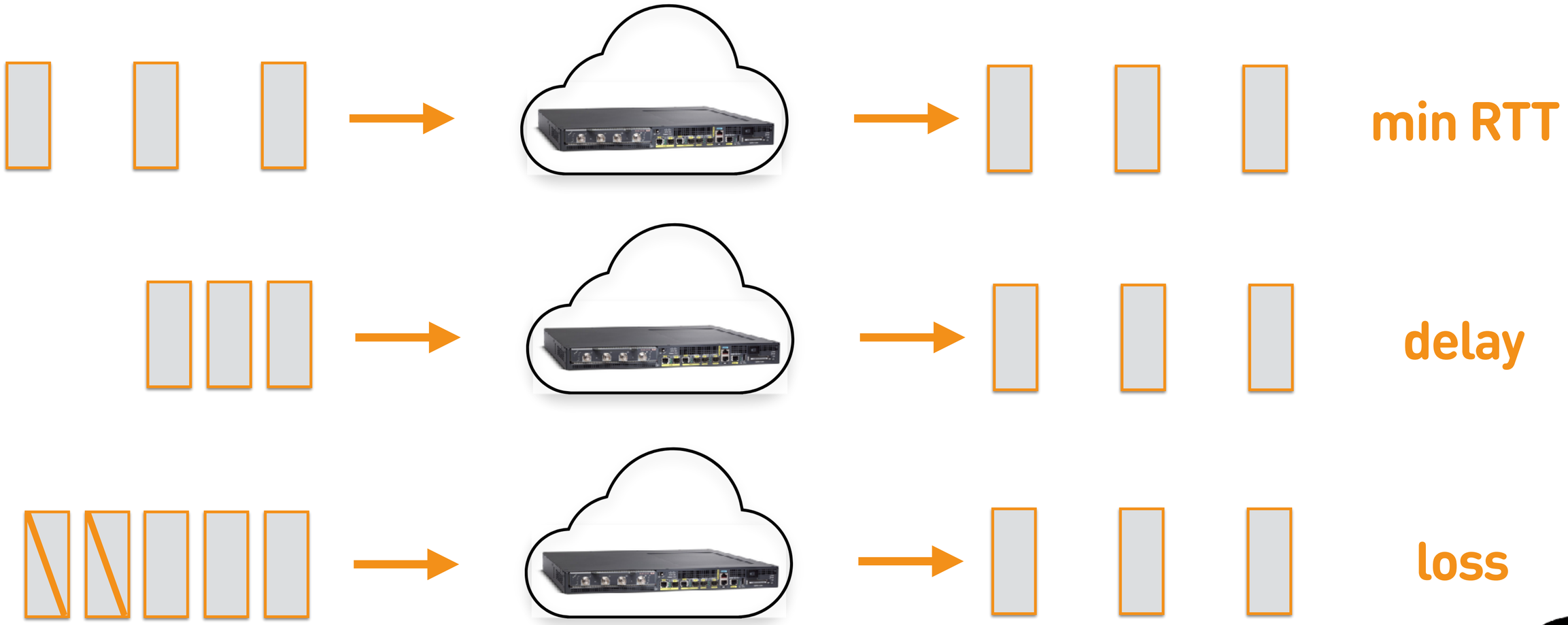


wnd=4

Congestion Control

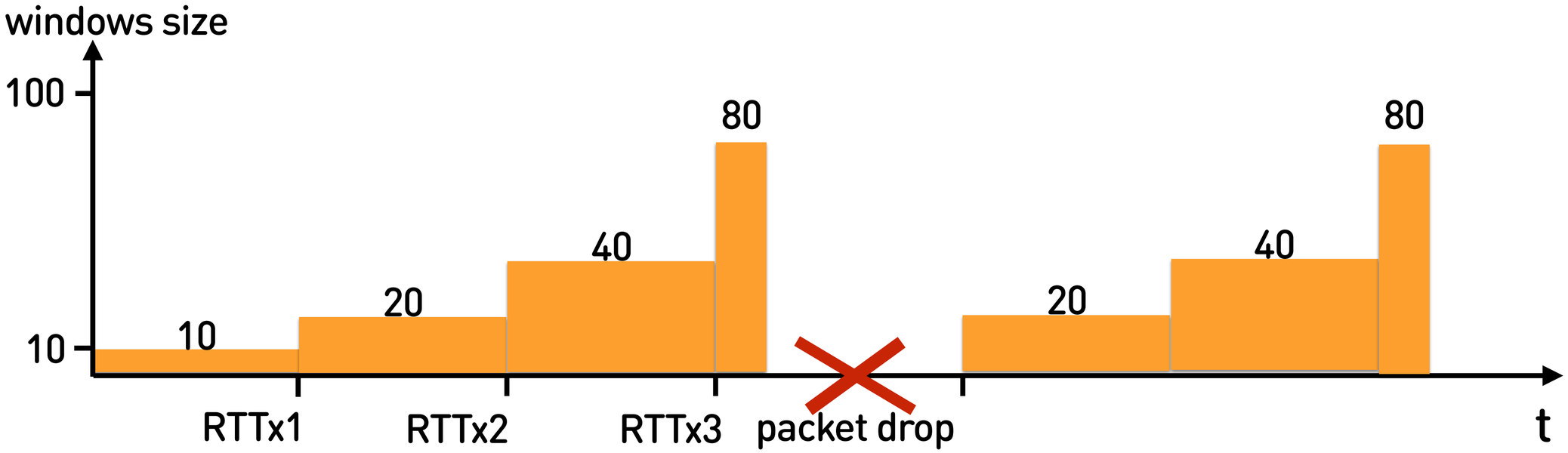


Congestion Control - feedback

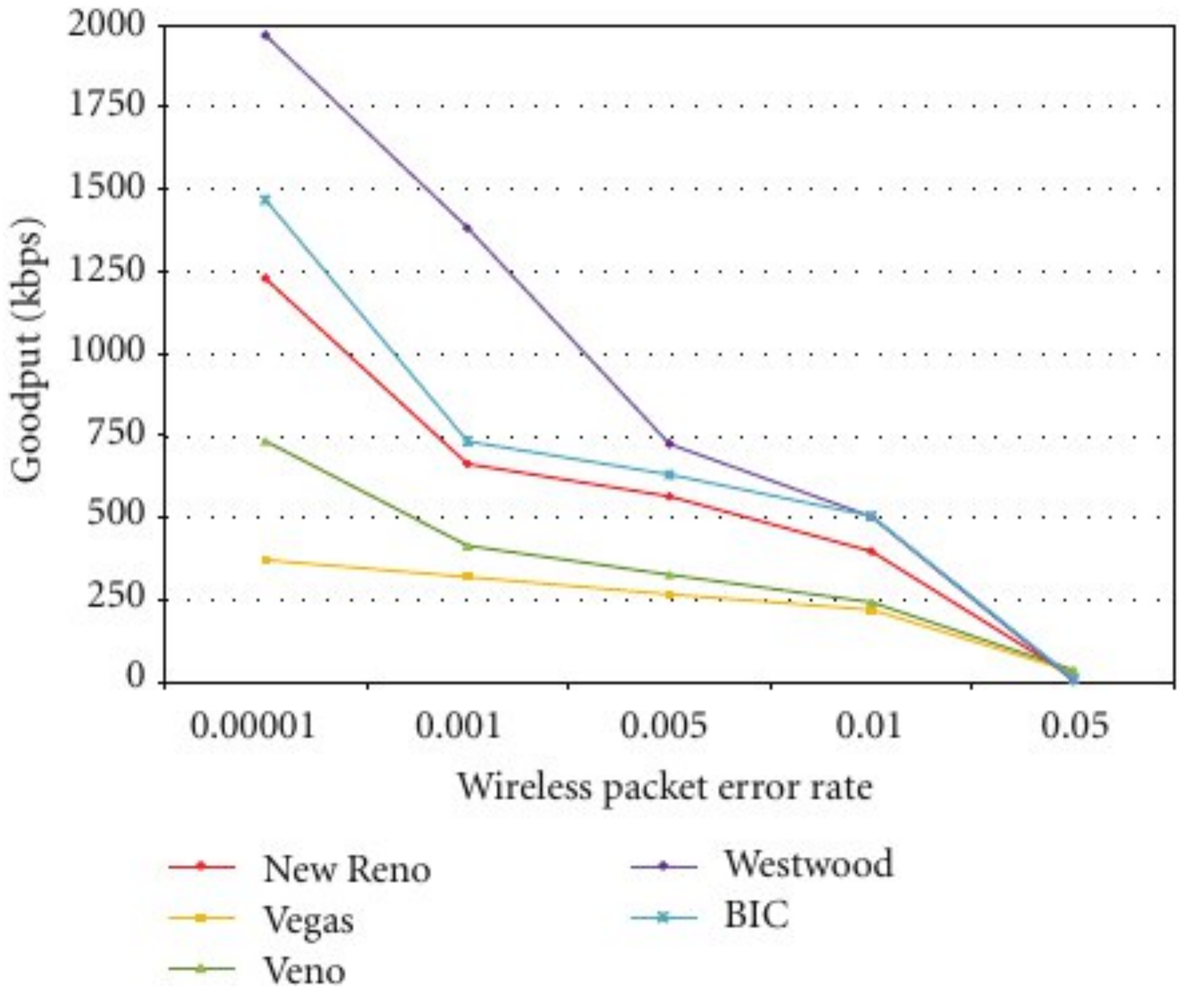


Packet loss model by router

Bottle neck router



Legacy CC algorithms



packet loss
==
congestion packet loss

! mobile internet always has packet loss

Congestion Control - feedback

Tahoe

1988 year

Slow Start and Congestion Avoidance

Reno

1990 year

Fast Recovery after Fast Retransmit

loss

New Reno

1996 year

New Fast Recovery Algorithm

West wood+

2001 year

Wireline and wireless networks

loss/ delay

Cubic

2006 year

Fast Long-Distance Networks

linux 2.6.19

loss

PCC

2015 year

New Fast Recovery Algorithm

congestion loss
random loss

BBR

2016 year

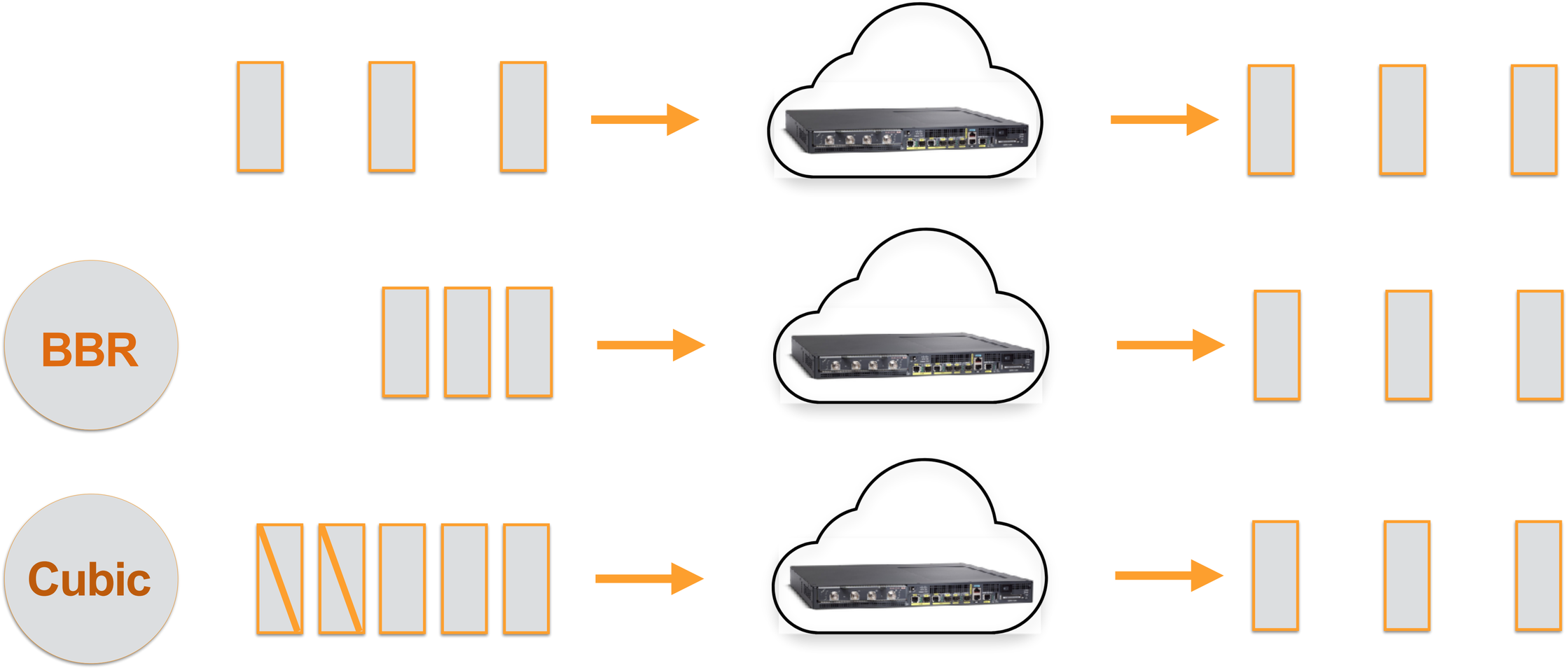
bandwidth-delay product (BDP)

delay

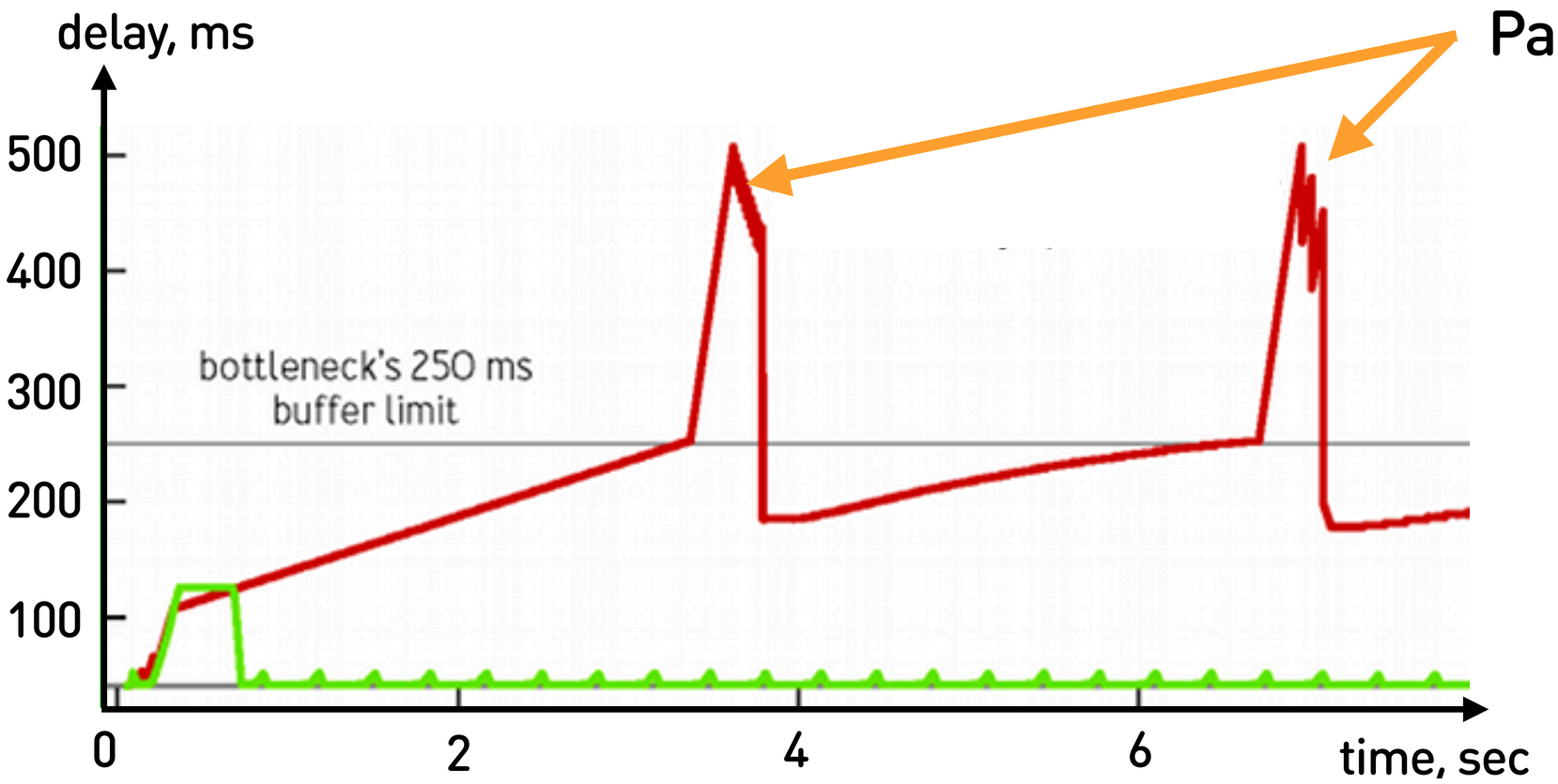
BBR Congestion Control

or Google helps us

Cubic vs BBR: feedback



Cubic vs BBR: delay



Packet Loss

Cubic

BBR

loss

delay

BBR & jitter

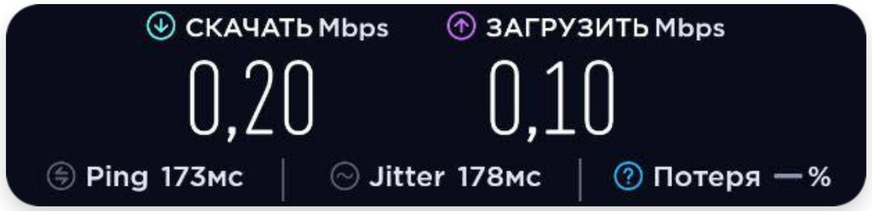
```
PING highload.ru (178.248.233.16): 56 data bytes
icmp_seq=11 ttl=43 time=117.177 ms
icmp_seq=12 ttl=43 time=132.868 ms
icmp_seq=13 ttl=43 time=176.413 ms
icmp_seq=14 ttl=43 time=225.981 ms
```

```
117ms to 132ms = 15ms
132ms to 176ms = 44ms
176ms to 225ms = 79ms

(14 + 44 + 79) / 3 = 46ms
```

50 ms

avg jitter

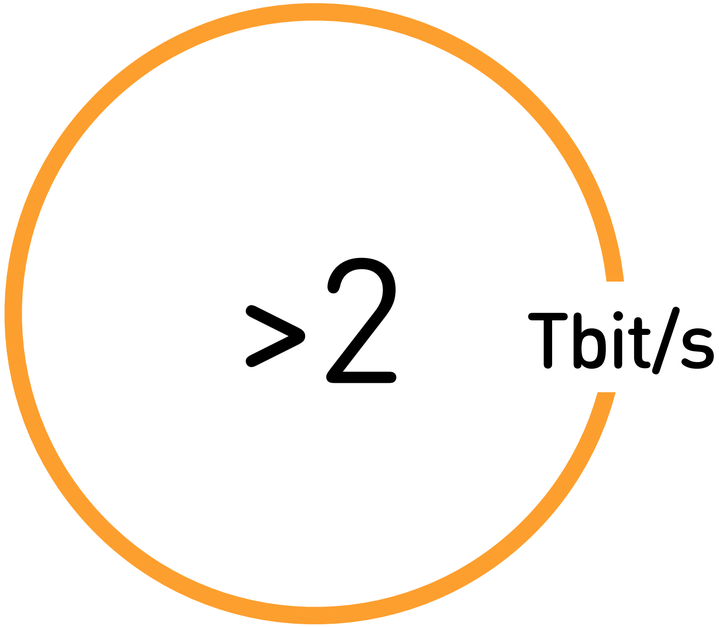


! BBR minimizes latency, ignores packet loss, but does not like jitter

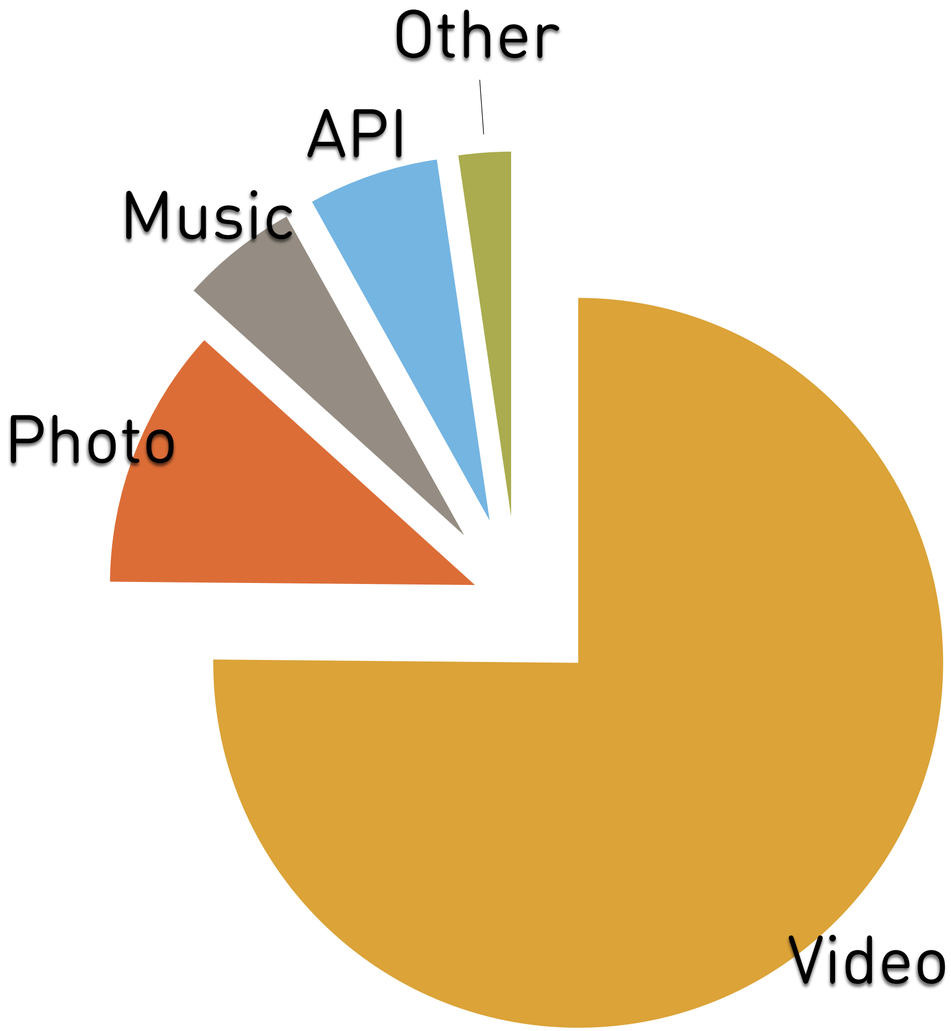
Congestion Control

what should we choose

Traffic at OK



traffic



Congestion control recommendation

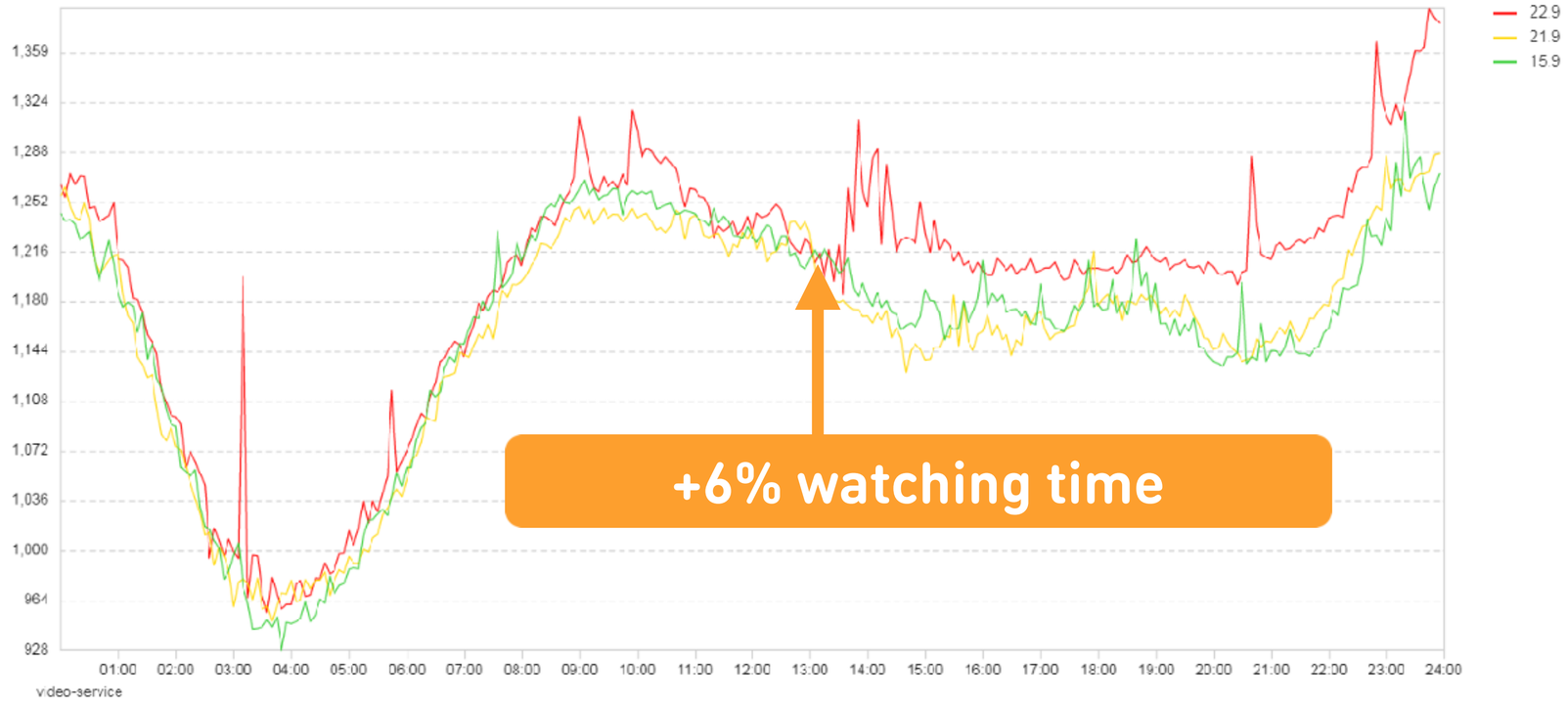
Algorithm	Network	AppProfile	Server
westwood+	high latency networks	speed	API
cubic	high bandwith/long RTT	stability	Photo
BBR	high loss	streaming	Video

! Collect statistic and set custom congestion control

Practice: BBR и video

	Kernel	Setting
Linux	4.9+	net.ipv4.tcp_congestion_control=bbr

Cumulative watching time (sec)



TCP at mobile networks: recommendations

1. set send/recv buffer size
2. reuse connection
3. check TFO, TLS, slow-start, keep-alive
4. select congestion control

Part1: TCP

1 Part1: TCP tuning in mobile networks

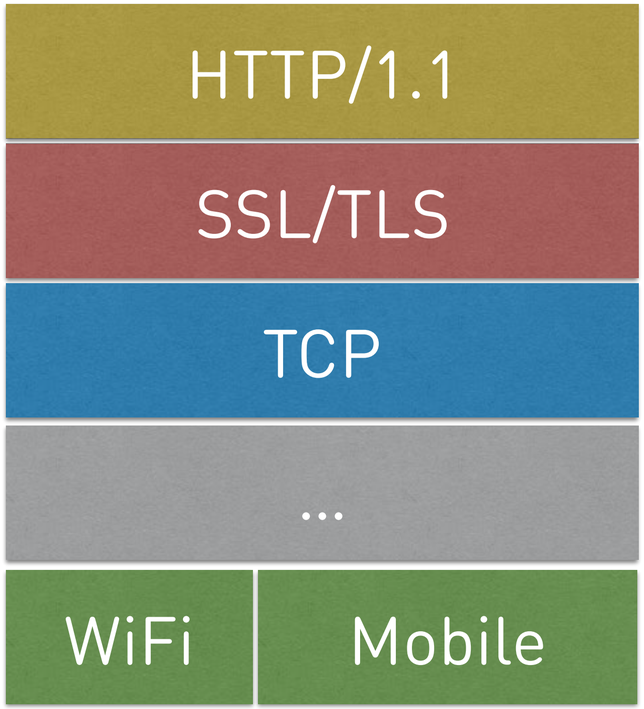
- puzzle
- connection establishment
- congestion control
- **HTTP 2.0**
- TCP troubles: head-of-line blocking, buffer bloat, IPMigration

HTTP 2.0

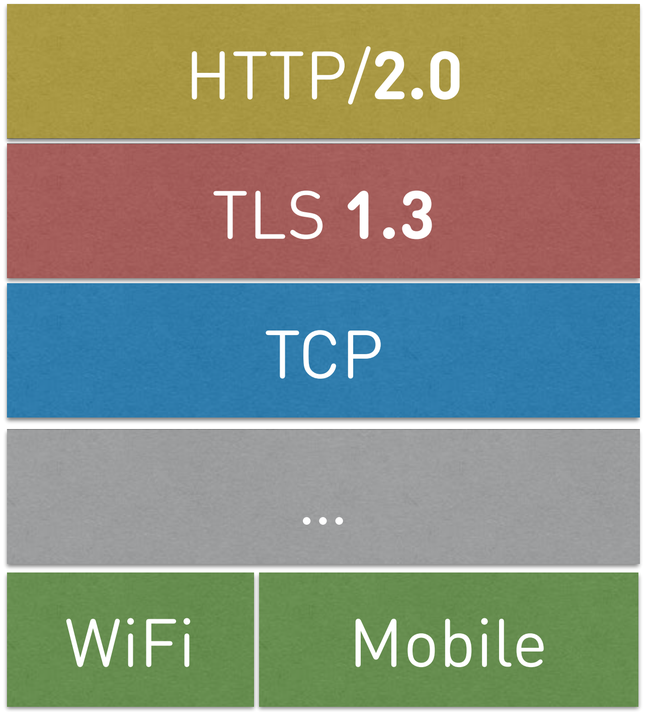
good protocol but not over TCP in mobile networks

HTTP 1.1 vs HTTP 2.0

Legacy app



Modern app



Easy to enable HTTP 2.0 at nginx

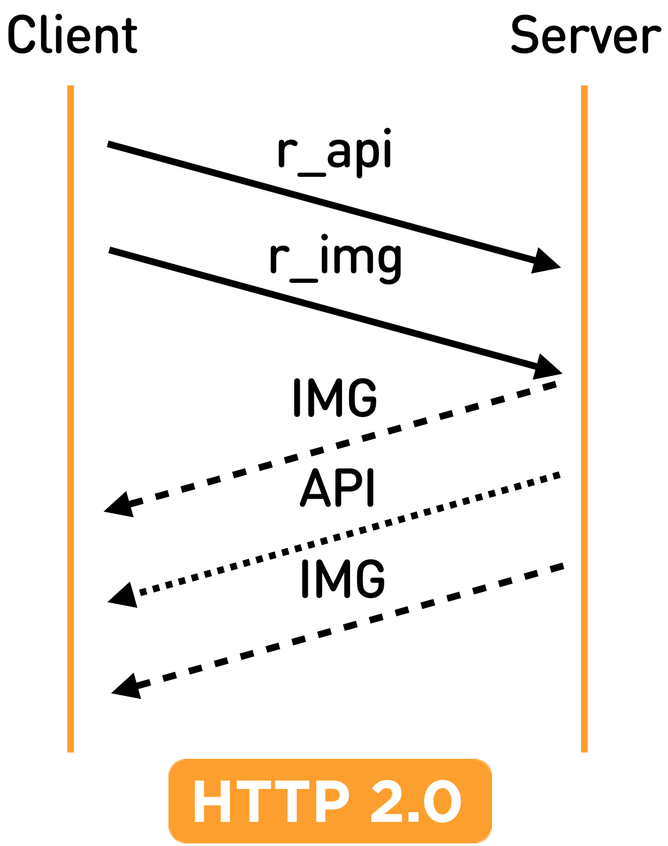
enable http2

```
server {
        listen 443 ssl http2;
        server_name http://appsconf.ru  www.http://appsconf.ru;
        ssl on;
...}
```

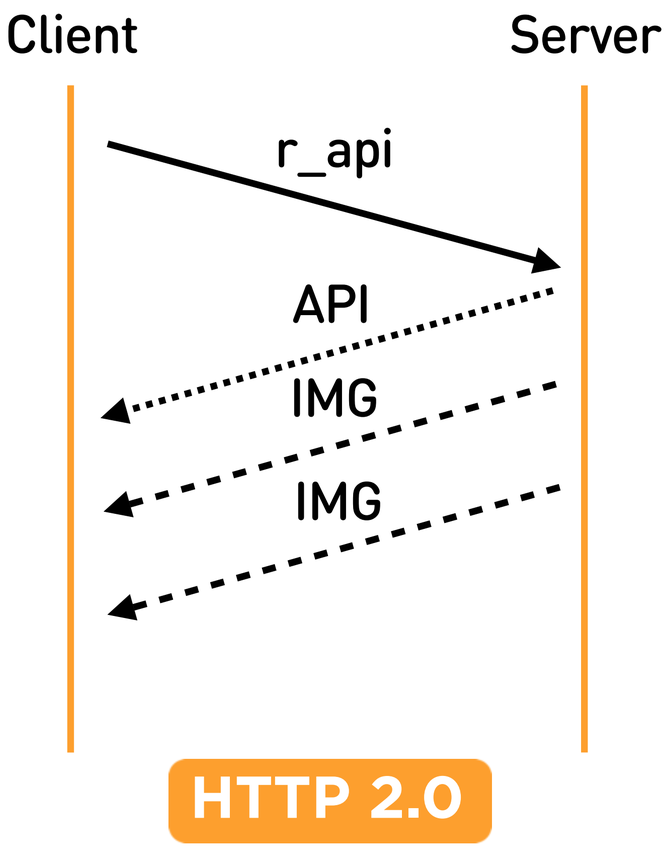
HTTP 2.0 - the differences

1. binary, Data compression of HTTP headers
2. multiplexing
3. prioritization
4. stream cancellation
5. server push

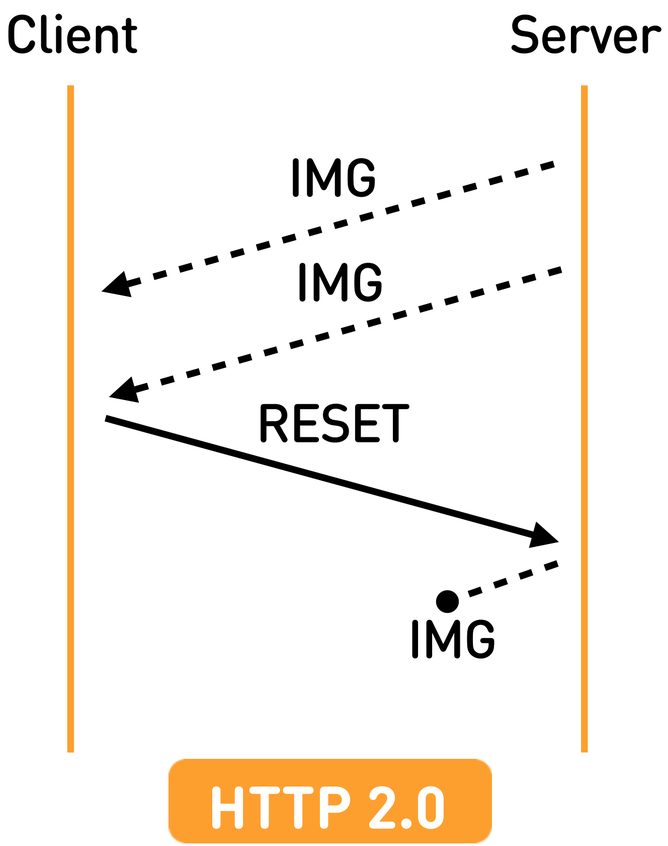
HTTP 2.0 multiplexing and prioritization



HTTP 2.0: server push



HTTP 2.0: stream cancellation

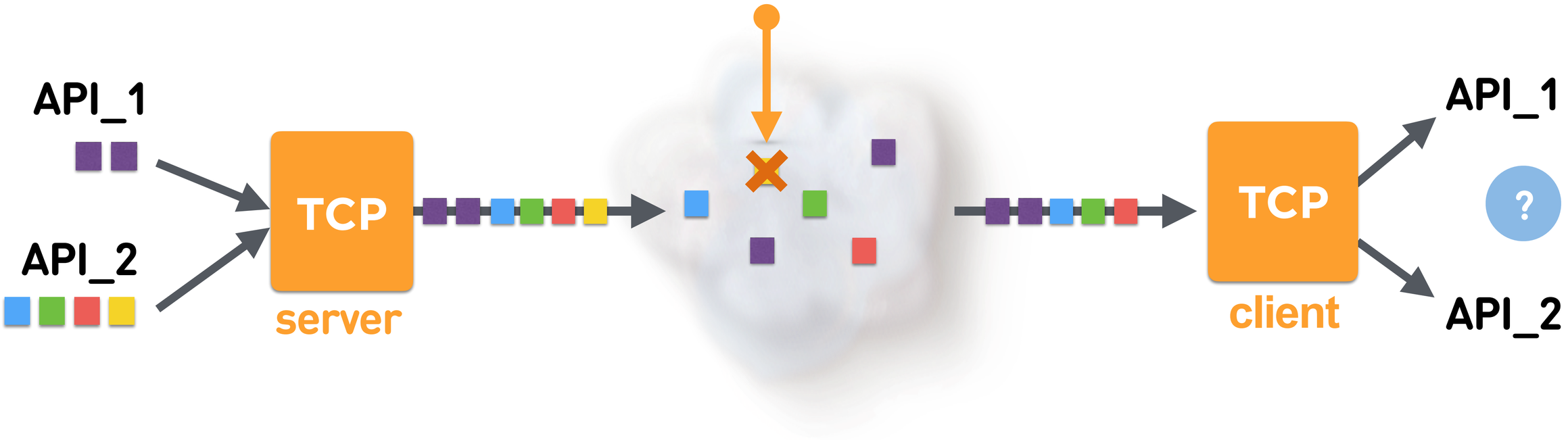


Part1: TCP

1 Part1: TCP tuning in mobile networks

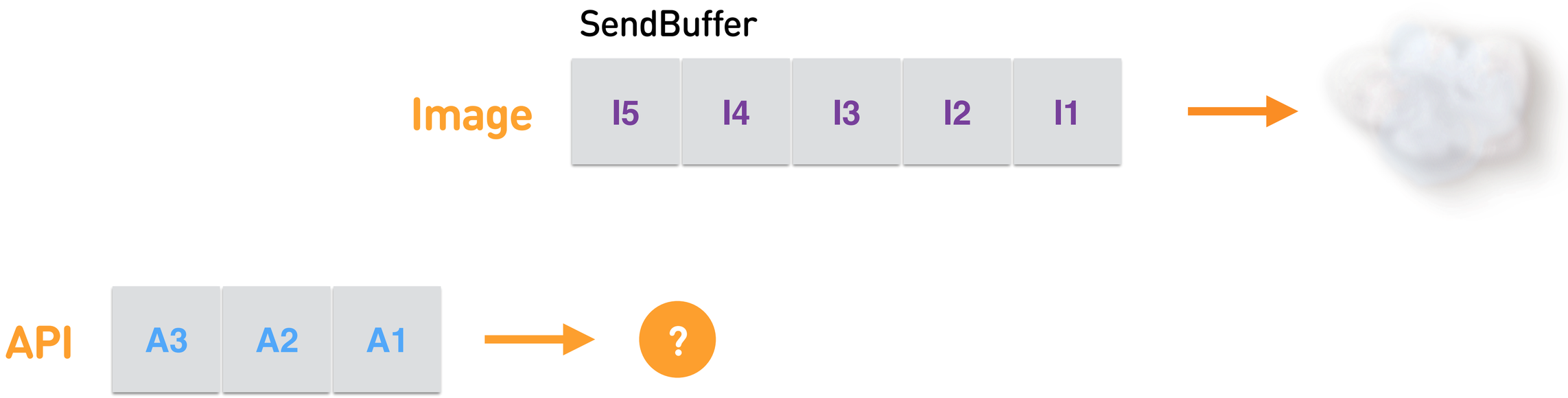
- puzzle
- connection establishment
- congestion control
- HTTP 2.0
- **TCP troubles: head-of-line blocking, buffer bloat, IP Migration**

TCP: Head-of-line blocking



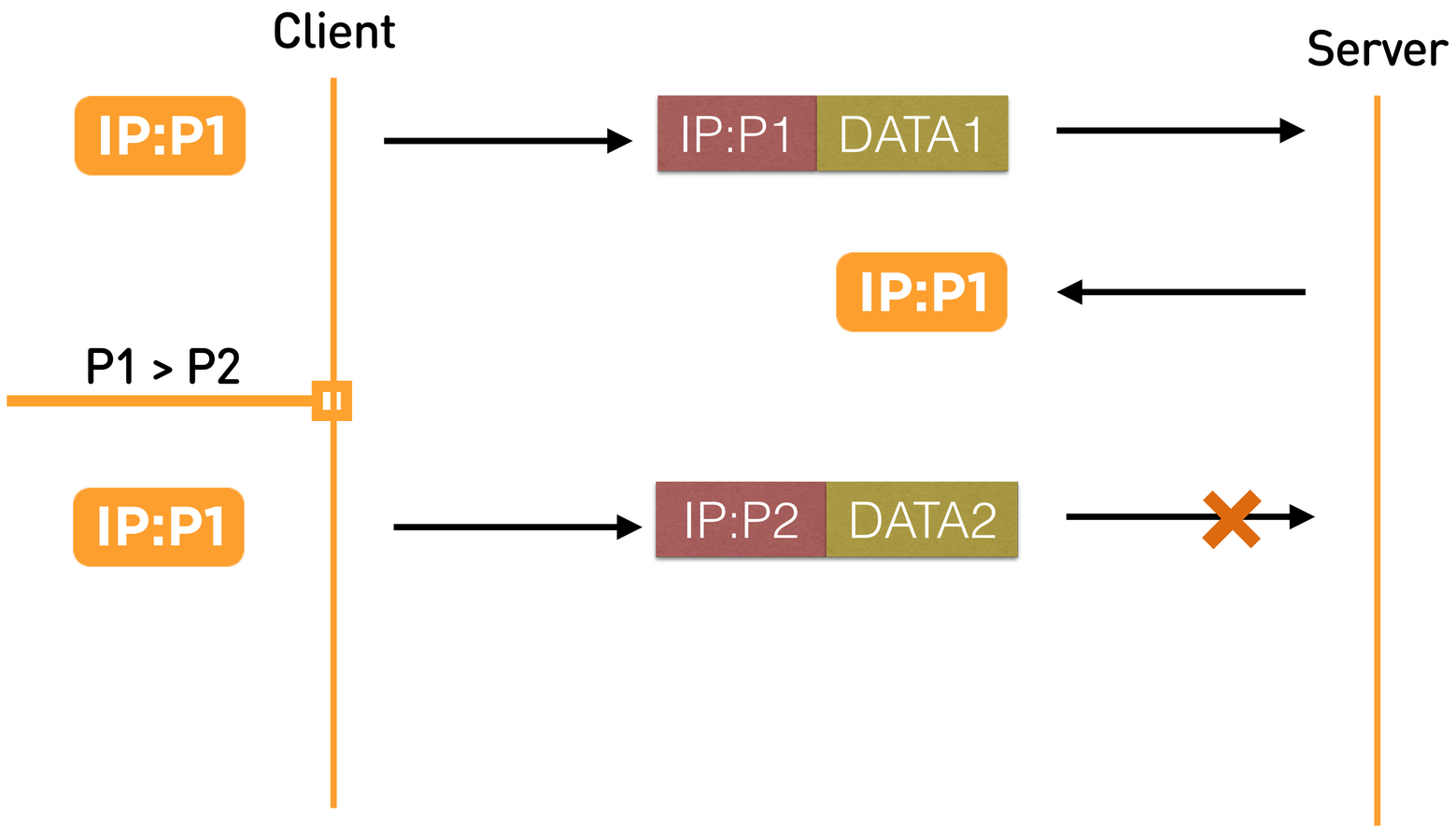
? HTTP2.0 and multiplexing

TCP: buffer bloat



? prioritization, stream cancelation, server push

TCP: IP migration



HTTP/2 over TCP troubles

packet loss → head-of-line blocking

variable bandwidth → buffering

Doesn't work:

1. multiplexing & head-of-line blocking
2. prioritization & buffering
3. stream cancelation & head-of-line blocking
4. server push & buffering

Part2: QUIC

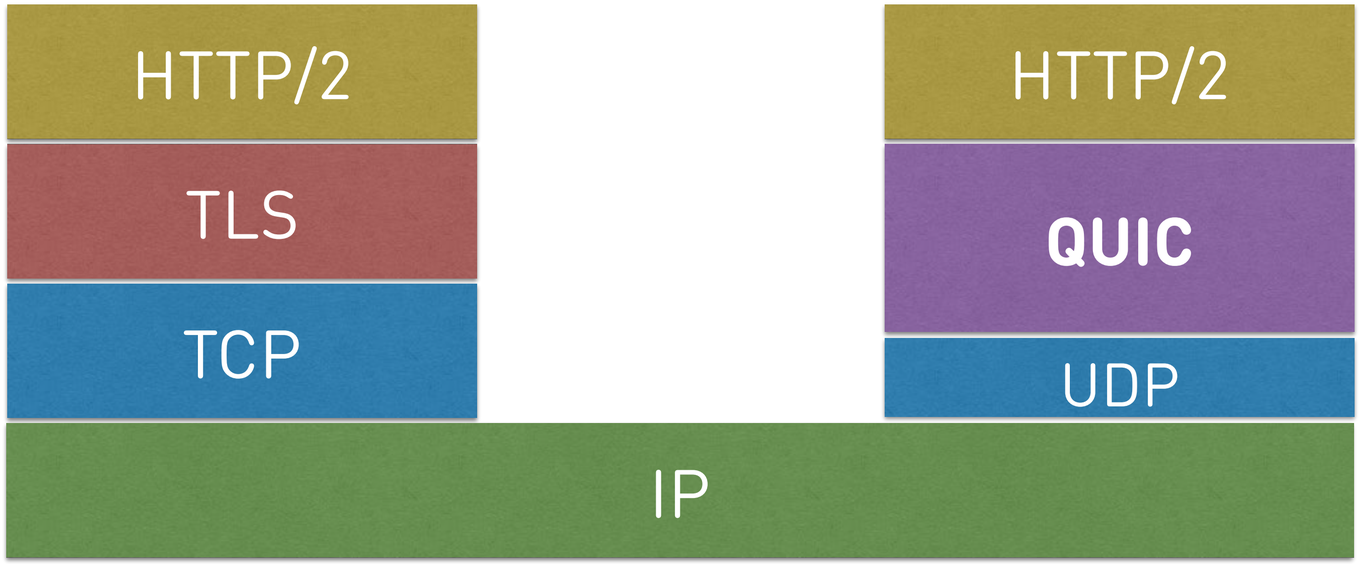
2

Part2: QUIC

- transport protocols over UDP in user space

- IP Migration
- UDP performance in Linux

QUIC



1. data integrity
2. multiplexing and prioritization
3. FEC and IP Migration

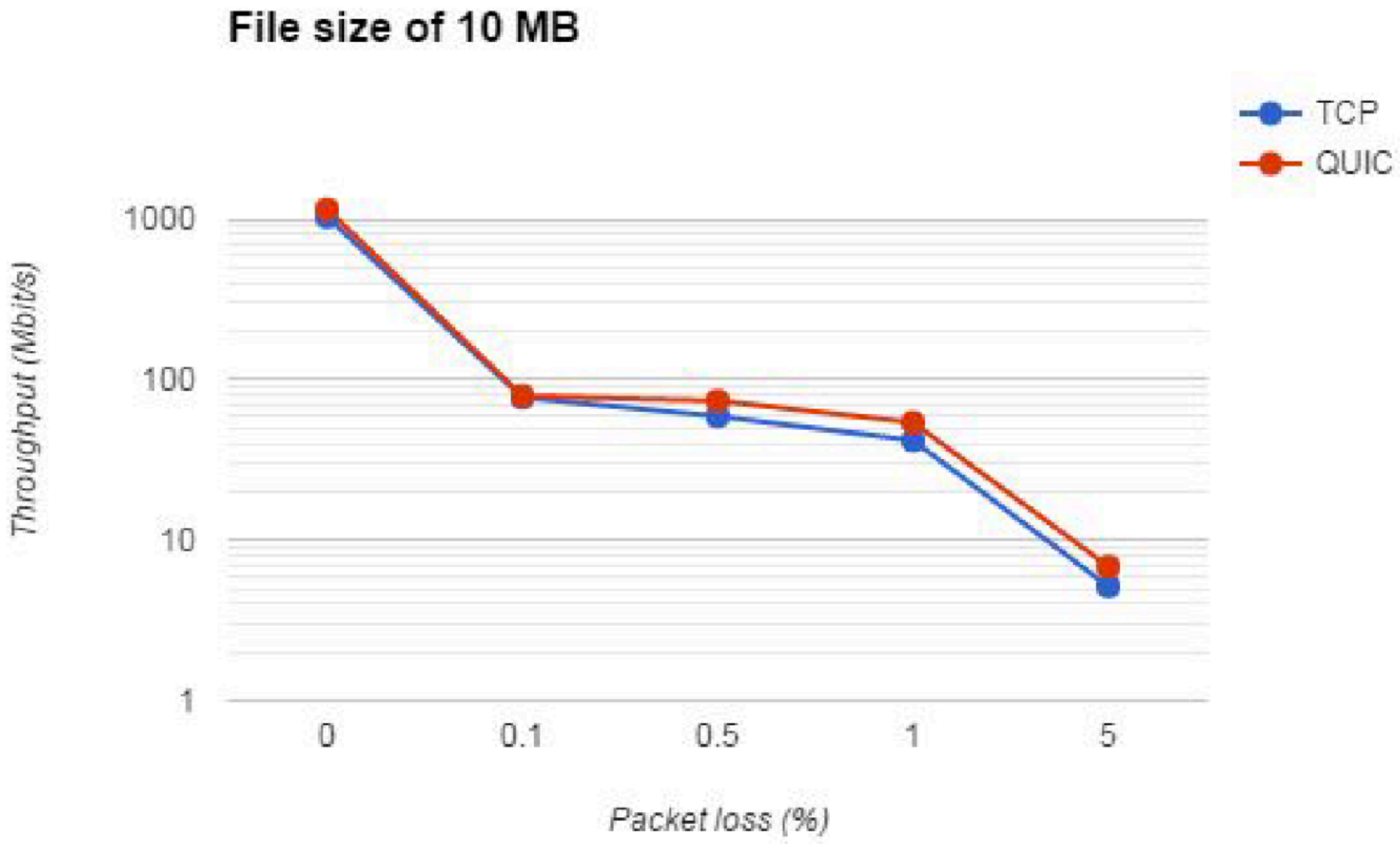
Why QUIC is not so quick?

or make the right measurements

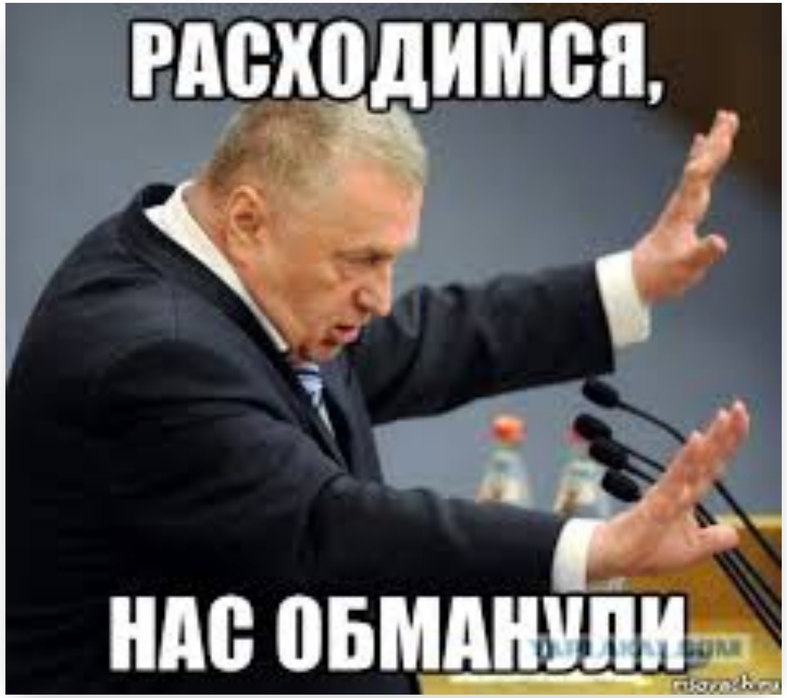
Why QUIC is not so quick?

> While Google-reported performance for **QUIC** is promising — **3%** page load time (PLT) improvement on **Google search** and **18%** reduction in buffer time on **YouTube** — they are aggregated statistics and not reproducible by others (such as ourselves).

Why QUIC is not so quick?



Why QUIC is not so quick?



1. muliplexing & pipelining
2. prioritization
3. IP Migration

QUIC

the future has come

QUIC or the future has come

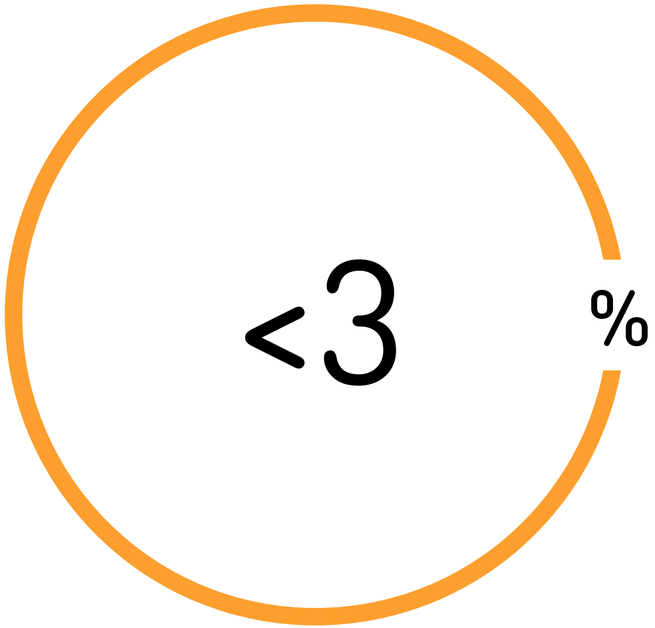
1. google.com
2. youtube
3. search
4. google drive coming soon
5. wwdc coming soon

https://datatracker.ietf.org/doc/draft-ietf-quic-transport/

UDP availability

skepticism?

UDP availability



no UDP available

1. VOIP (webRTC)
2. QUIC
3. Games

Part2: QUIC

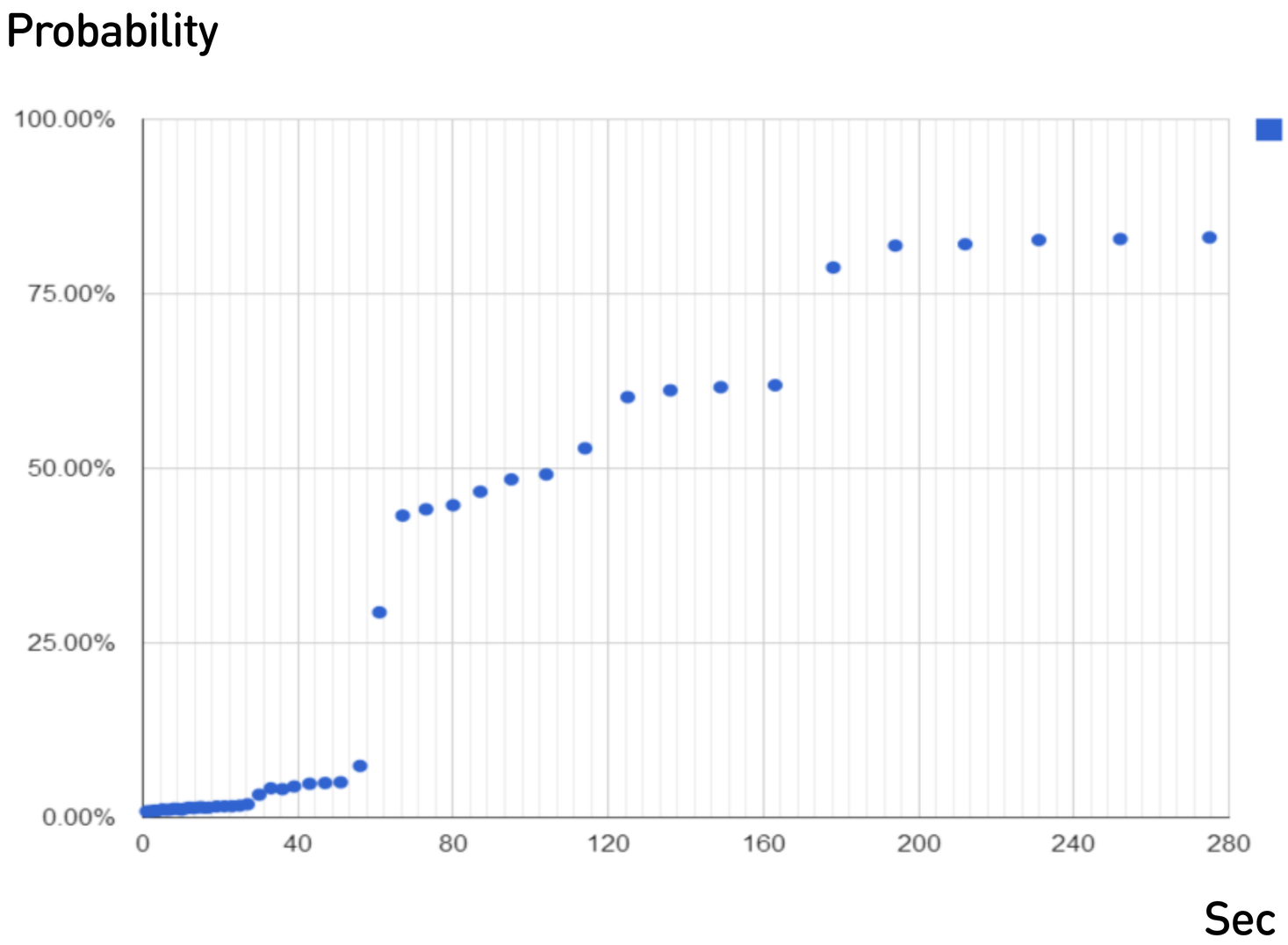
2

Part2: QUIC
- transport protocols over UDP in user space
- **IP Migration**
- UDP performance in Linux

UDP NAT unbinding

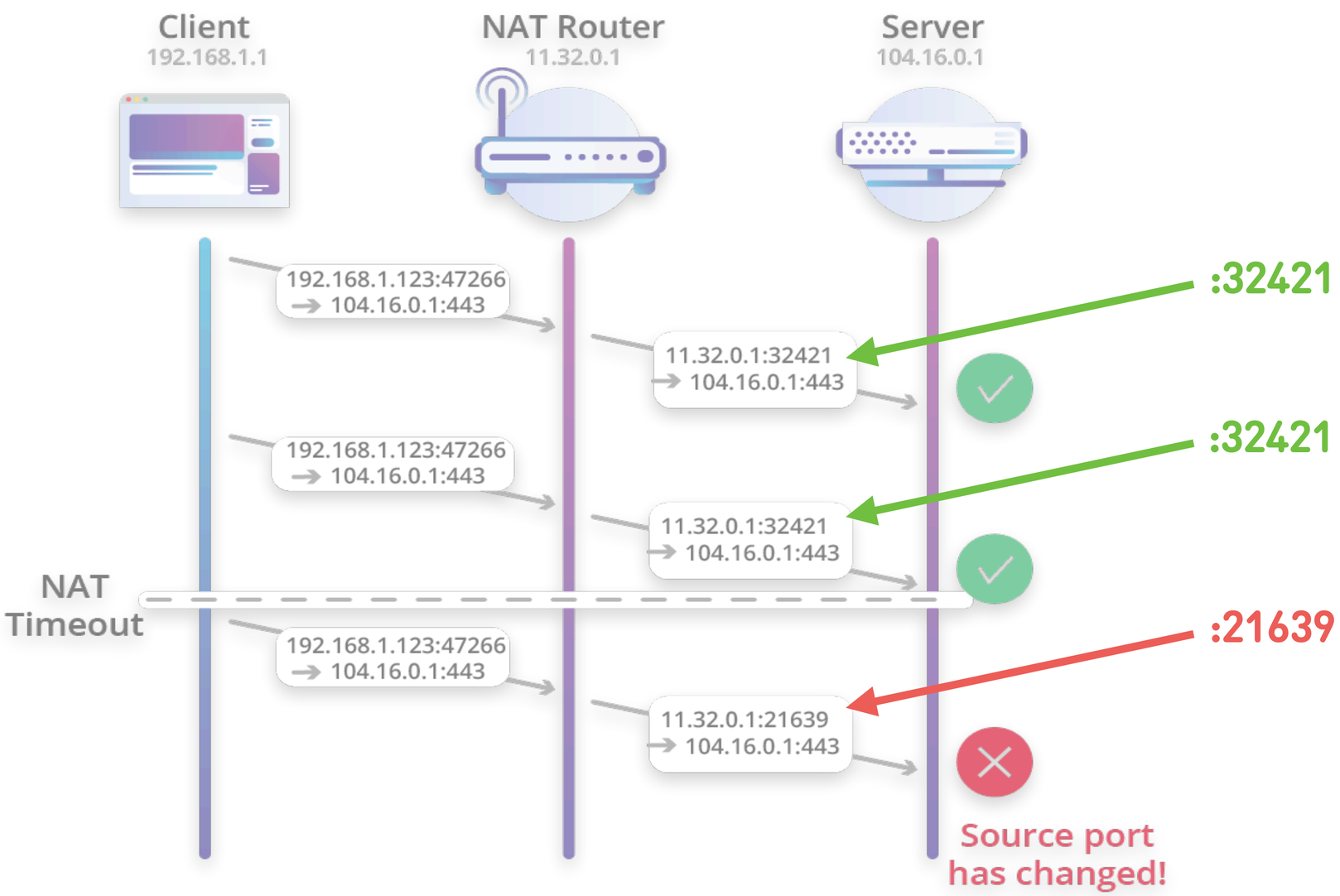
skepticism?

NAT unbinding

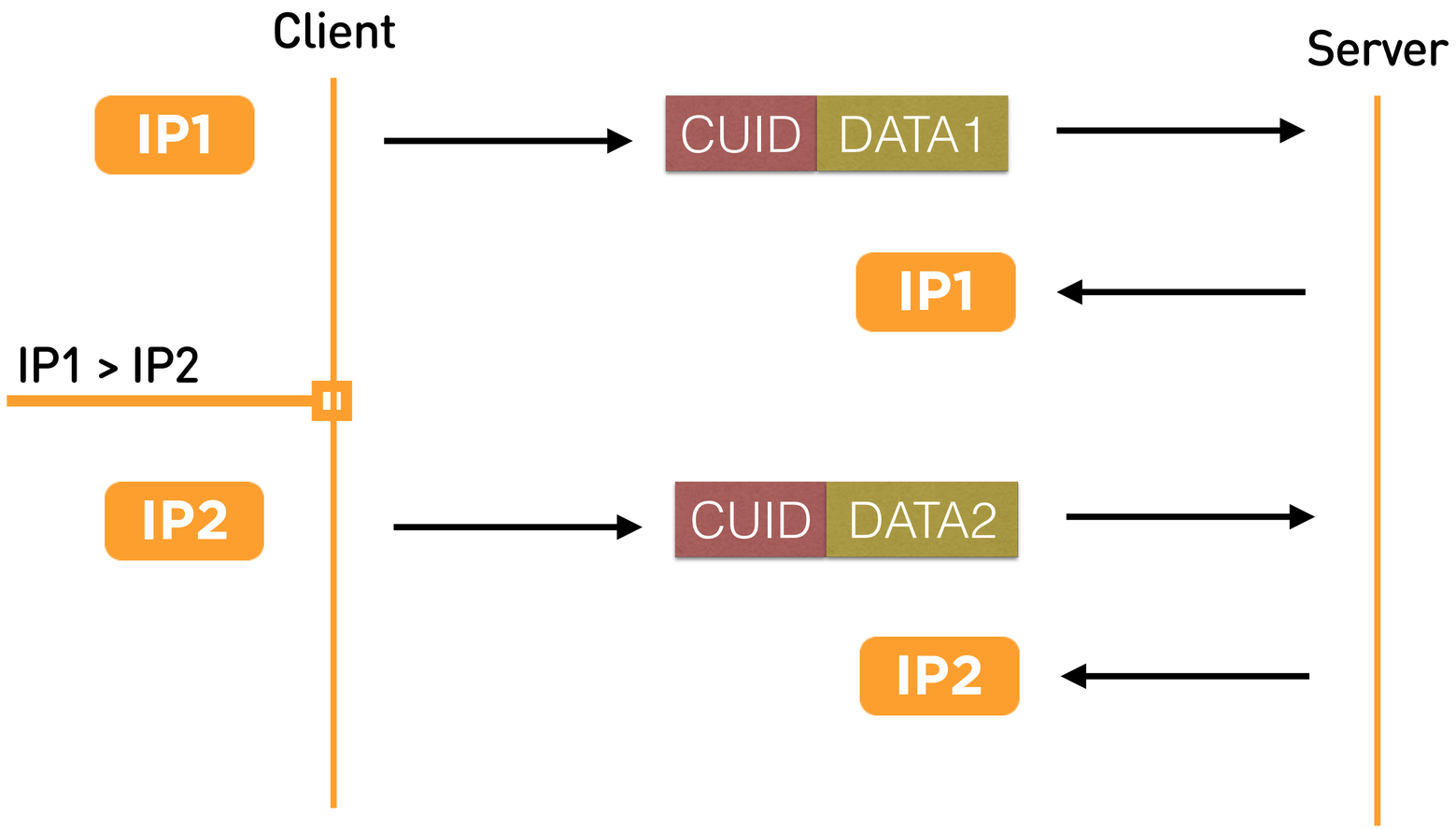


ping-pong
30-60sec

But sometimes NAT unbinding happens



IP Migration



Part2: QUIC

2

Part2: QUIC
- transport protocols over UDP in user space
- IP Migration
- **UDP performance in Linux**

UDP performance issue at Linux

1. route table lookup
2. large segment offload
3. etc.

vger.kernel.org/netconf2017_files/rx_hardening_and_udp_gso.pdf

UDP performance issue at Linux

"One challenge with QUIC at the moment is the increased CPU cost of sending UDP packets vs TCP payloads.

I've seen this ***across every platform*** *Google has deployed QUIC on, so it's a widespread issue."*

– Ian Swett, Google.

Performance vs bug fixes in OS release

Do QUIC in linux kernel?

QUIC

"The reason we want to be able to work at the application level is backwards compatibility. It has to be possible to ***deploy QUIC on any machine even without OS support*** *or it won't be deployable.*

It does not have to be performant on every platform. If people are using QUIC, ***whatever needs to be moved into the kernel for performance reasons will move there****."*

– Phillip Hallam-Baker.

Why user space protocol so demanded

	Server	Client
Increasing TCP's Initial Window	+	-
TCP Fast Open	+	50/50 IOS, Android
TLP (Tail loss probe)	+	-
Early Retransmit for TCP	+	-
RACK: a time-based fast loss detection algorithm for TCP	+	-
TLS 1.3	+	FIZZ

UDP at mobile networks: recommendations

1. do migration to QUIC or other UDP
2. IP migration helps us
3. don’t worry about user space
4. QUIC is already everywhere

Part 3: self made UDP protocols

3

Part3: selfmade UDP protocols
- OKMP - protocol for video streaming
- UT2 - protocol for impulse data transferring

OKMP and UT2

SUV can't win at Nurburgring

Mobile networks bandwidth/packet loss/RTT

0.2	Mbit/s	2.5	%	900	ms	EGDE
1.0	Mbit/s	0.5	%	550	ms	3G
2.0	Mbit/s	0.7	%	220	ms	LTE
2.2	Mbit/s	0.5	%	110	ms	WiFi

! can’t solve all tasks by one protocol

QUIC

Ethernet

3G/LTE

WIFI

MobileApp

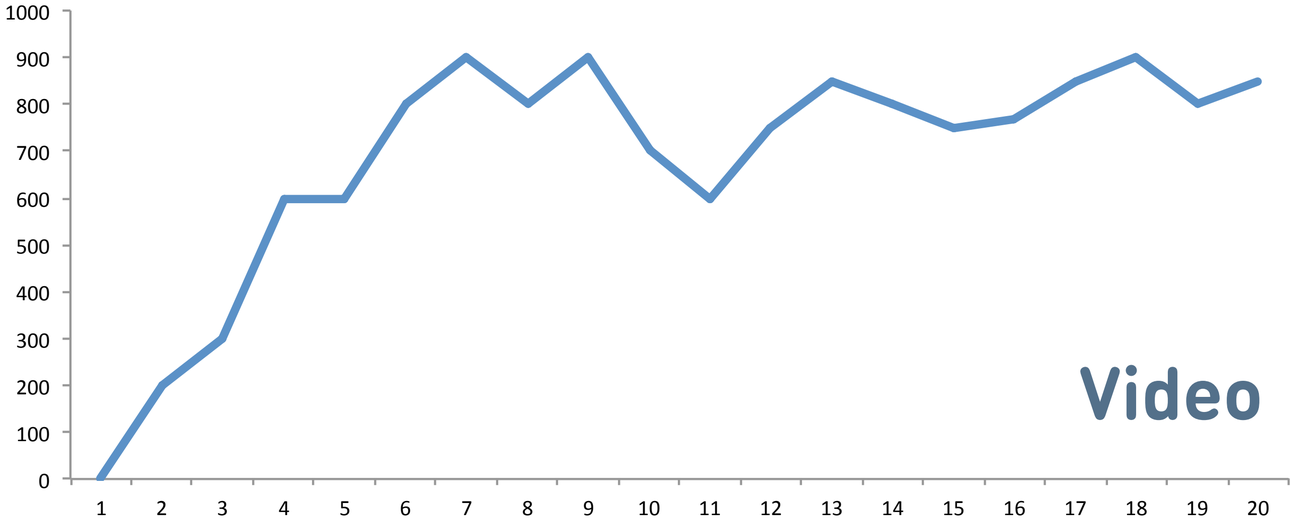
WEB

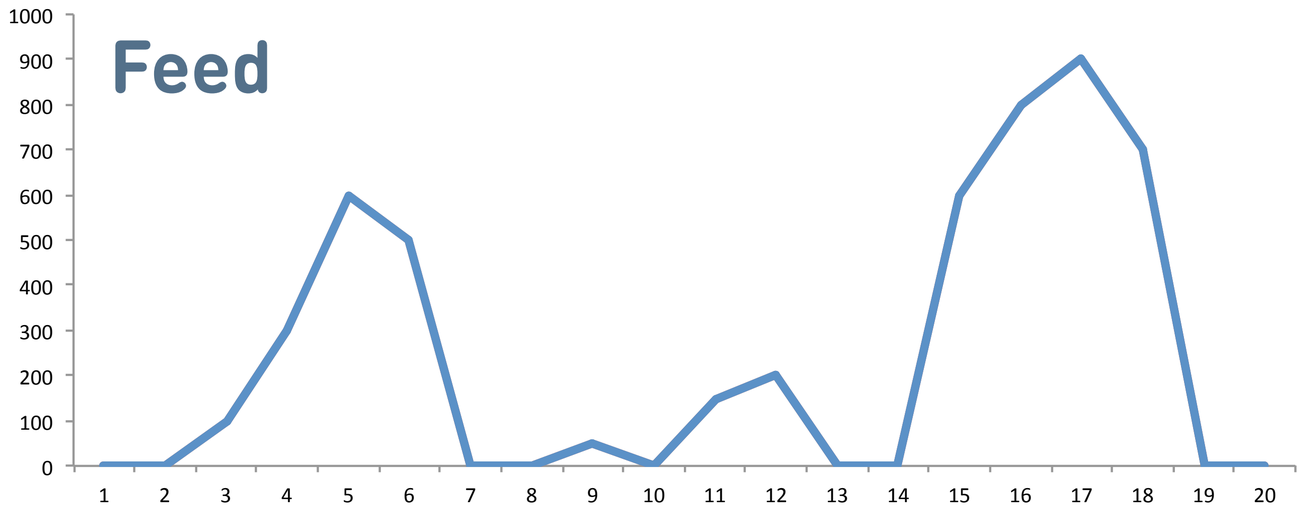
Video

Photo

API

OKMP and UT2





OKMP:

1. video streaming
2. low latency

UT2:

1. impulse data transfer
2. scheduler
3. transfer cancellation

Part 3: self made UDP protocols

3

Part3: selfmade UDP protocols

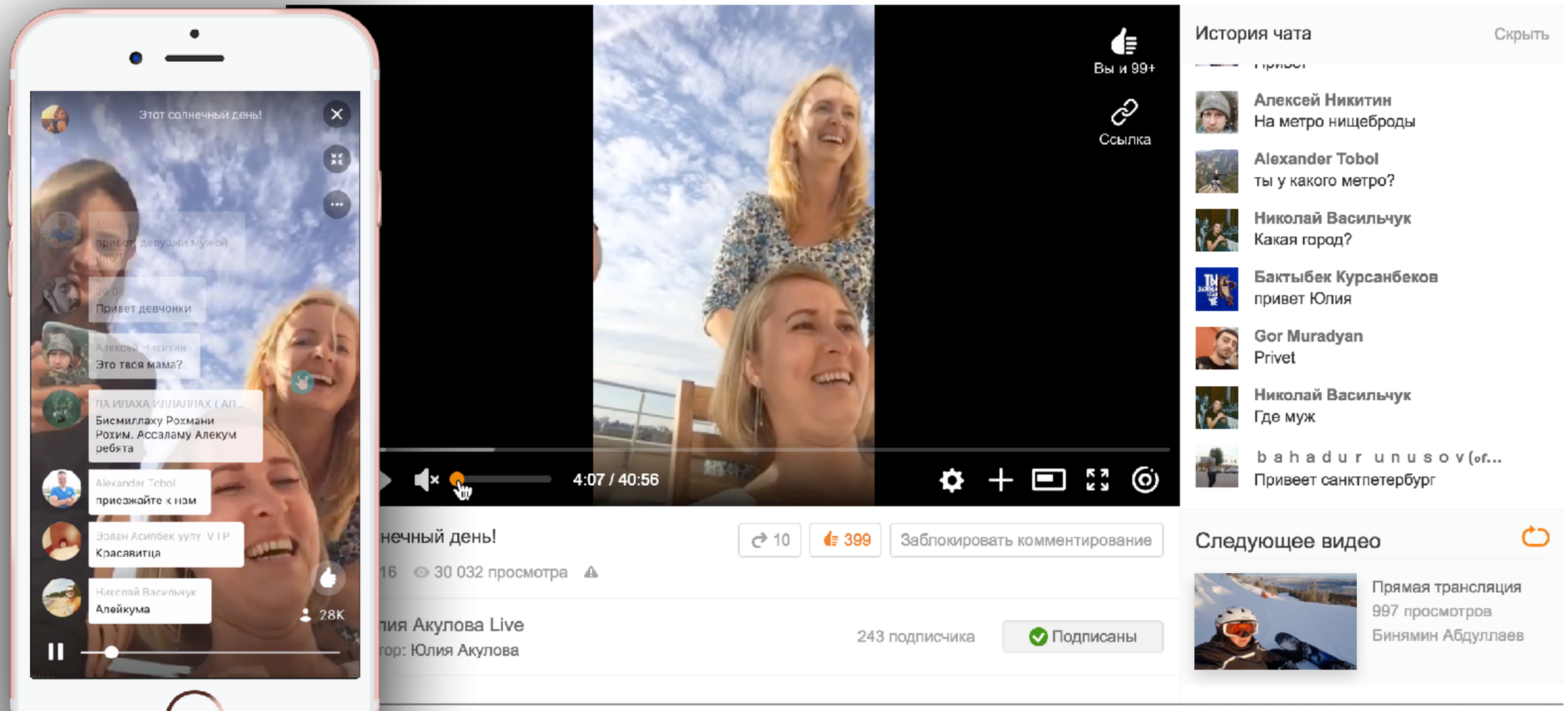
- OKMP - protocol for video streaming

- UT2 - protocol for impulse data transferring

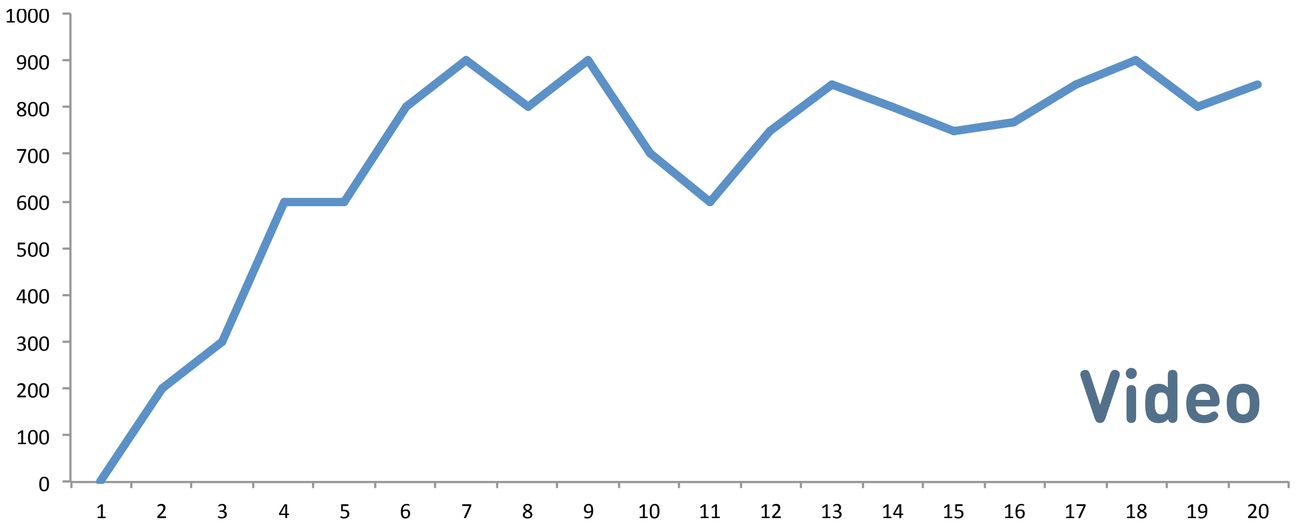
OKMP

video streaming in mobile networks

OKMP: application



OKMP: domain area



Задача:

1. video streaming
2. low latency

Data types	Priority	Guaranteed delivery	Encryption
Metadata	High	Guaranteed	enabled
Audio	Medium	< 500* мс	optional
Video	Low	< 300* мс	optional

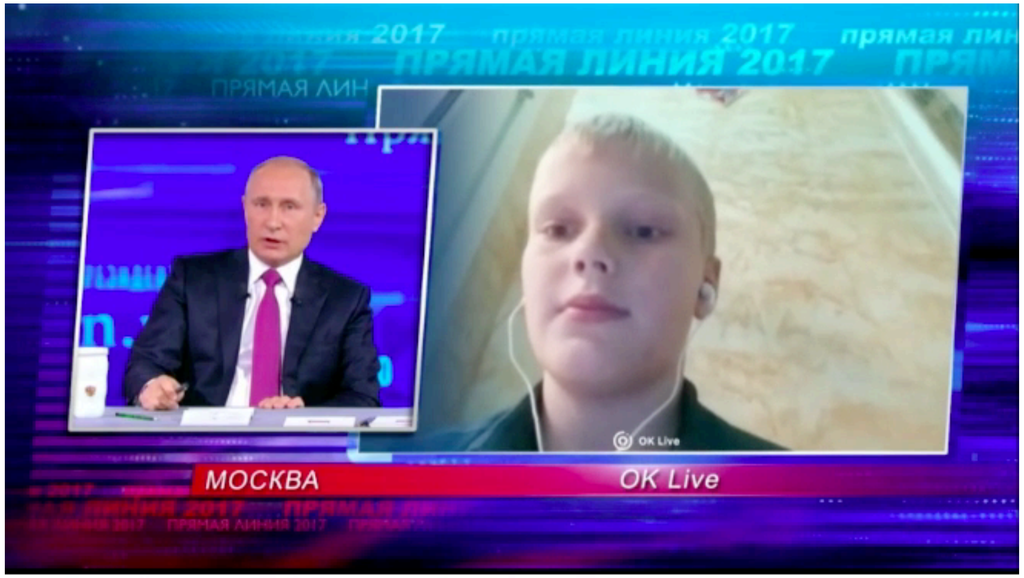
OKMP: features

1. multiplexing and prioritisation
2. optional guaranteed delivery
3. guaranteed latency (deliver in N ms or drop)
4. MTU discovery and CC over FEC

https://habr.com/company/oleg-bunin/blog/413479/

OKMP: results

1. streaming latency less than 1 sec
2. Full HD video streaming at mobile



Прямая линия с Владимиром Путиным

3 587 | 30 421 | Заблокировать комментирование

30 мая · 2 637 337 просмотров

Part 3: self made UDP protocols

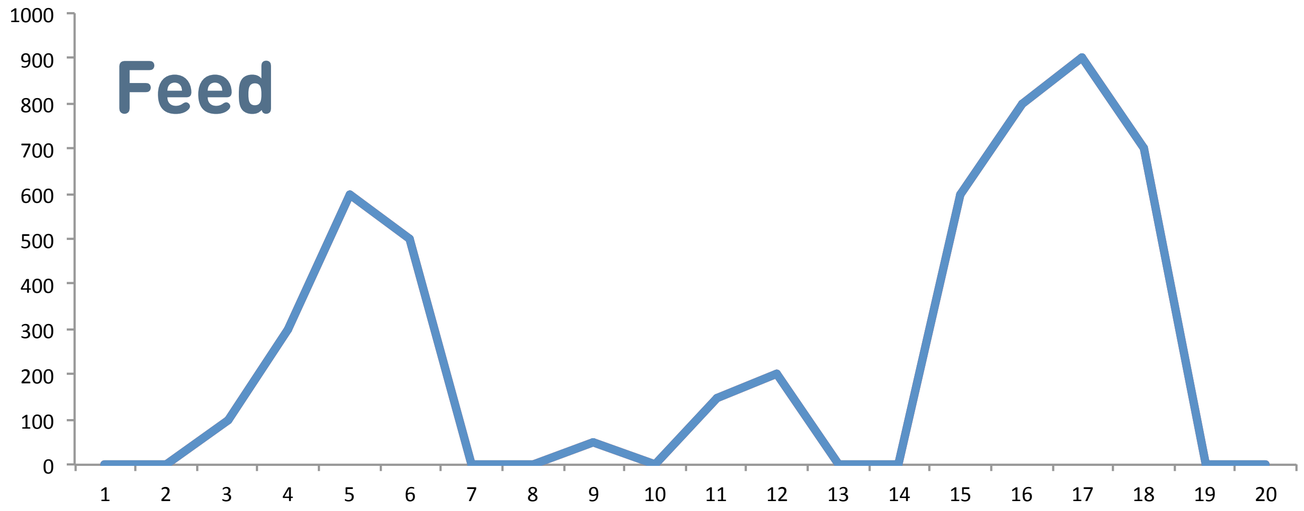
3

Part3: selfmade UDP protocols
- OKMP - protocol for video streaming
- UT2 - protocol for impulse data transferring

UT2

or transport protocol for impulse data transfer in mobile networks

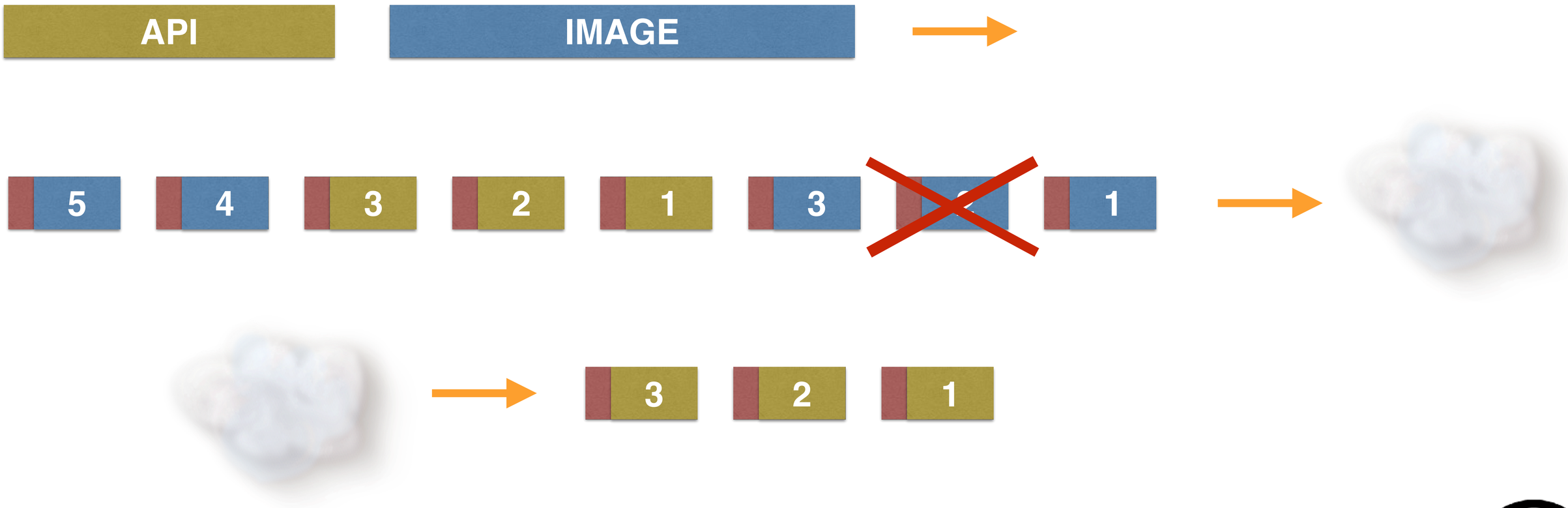
UT2: domain area



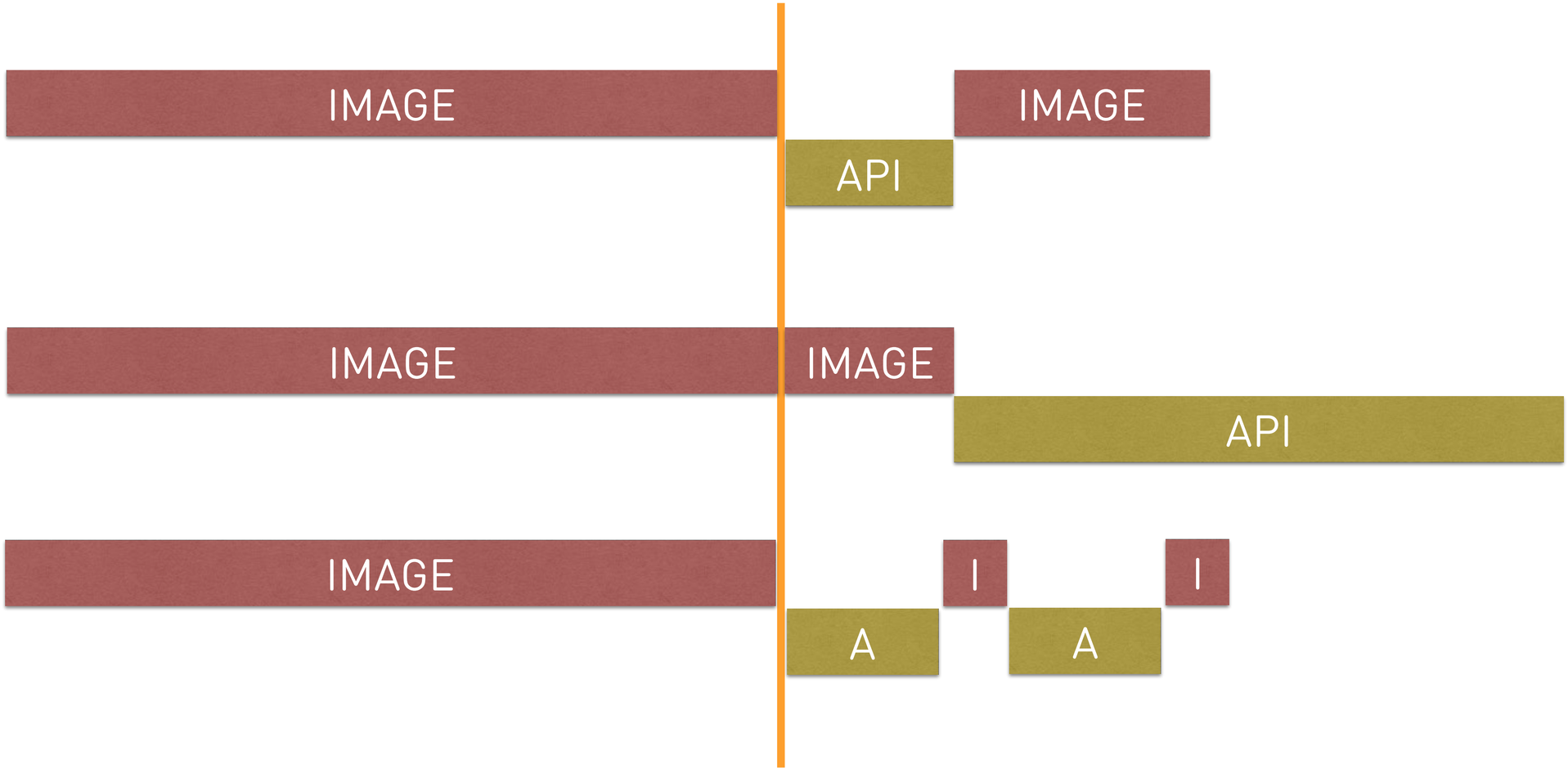
UT2:

1. impulse transfer
2. avoid DNS lookup
3. scheduler
4. IP Migration
5. TLP and soft FEC
6. zeroRTT

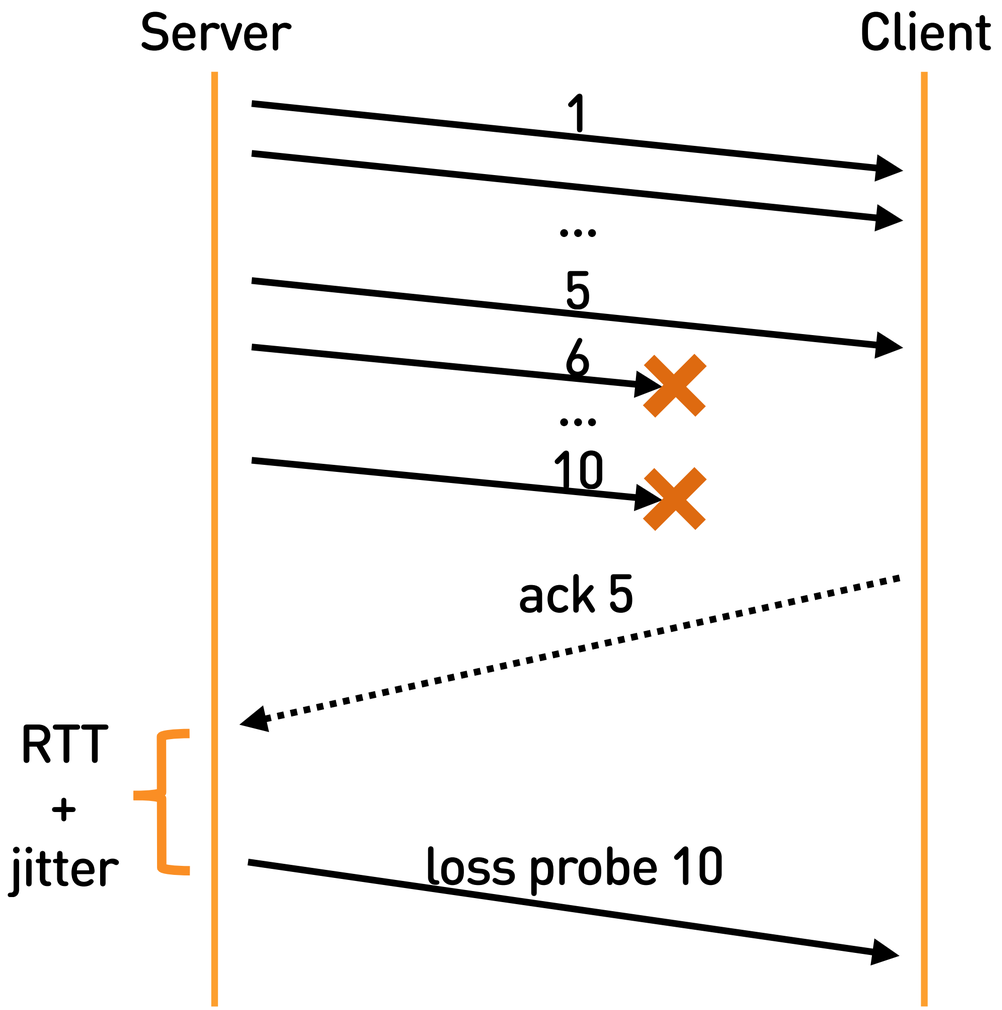
UT2 multiplexing and prioritisation looks like QUIC



UT2: scheduler



TLP: tail loss probe and soft FEC



TLP:

1. loss probe (packet number)
2. FEC (loss data)

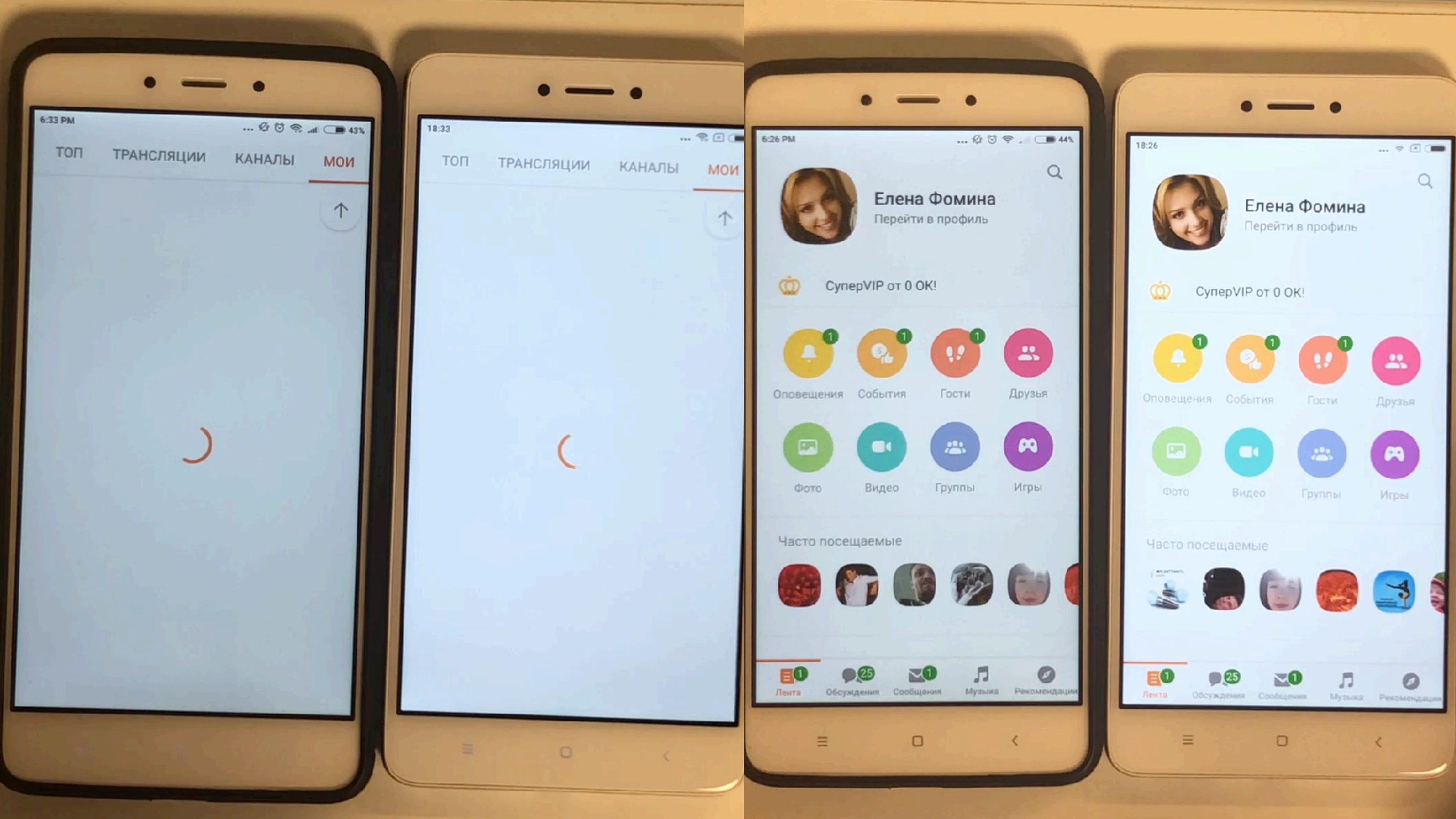
! slow delivery TLP/FEC to TCP production

QUIC & UT2

1. traffic at wireless networks 30/70 upload/download
2. increase ACK size and update legacy CC

Latency numbers in mobile networks

	TLS + TCP	QUIC	UT2
Radio			
DNS lookup	1 RTT	1	0
TCP handshake	1 RTT	0.5	0
TLS handshake	1-2 RTT	0	0
request	1 RTT	1	1
Result	4-5 RTT	2.5 RTT	1 RTT


6:33 PM
43%
ТОП
ТРАНСЛЯЦИИ
КАНАЛЫ
МОИ
18:33
ТОП
ТРАНСЛЯЦИИ
КАНАЛЫ
МОИ
6:26 PM
44%
Елена Фомина
Перейти в профиль
СуперVIP от 0 ОК!
Оповещения
События
Гости
Друзья
Фото
Видео
Группы
Игры
Часто посещаемые
Лента
Обсуждения
Сообщения
Музыка
Рекомендации
18:26
Елена Фомина
Перейти в профиль
СуперVIP от 0 ОК!
Оповещения
События
Гости
Друзья
Фото
Видео
Группы
Игры
Часто посещаемые
Лента
Обсуждения
Сообщения
Музыка
Рекомендации

UT2 and results at OK

-10%

API

-7%

Image

+1%

views

! can’t be measured simply since the success queries quantity increases

UT2 Performance

1. 5 Gbit/sec per sever
2. Encryption & Deflate

Part4: Future of applications stack

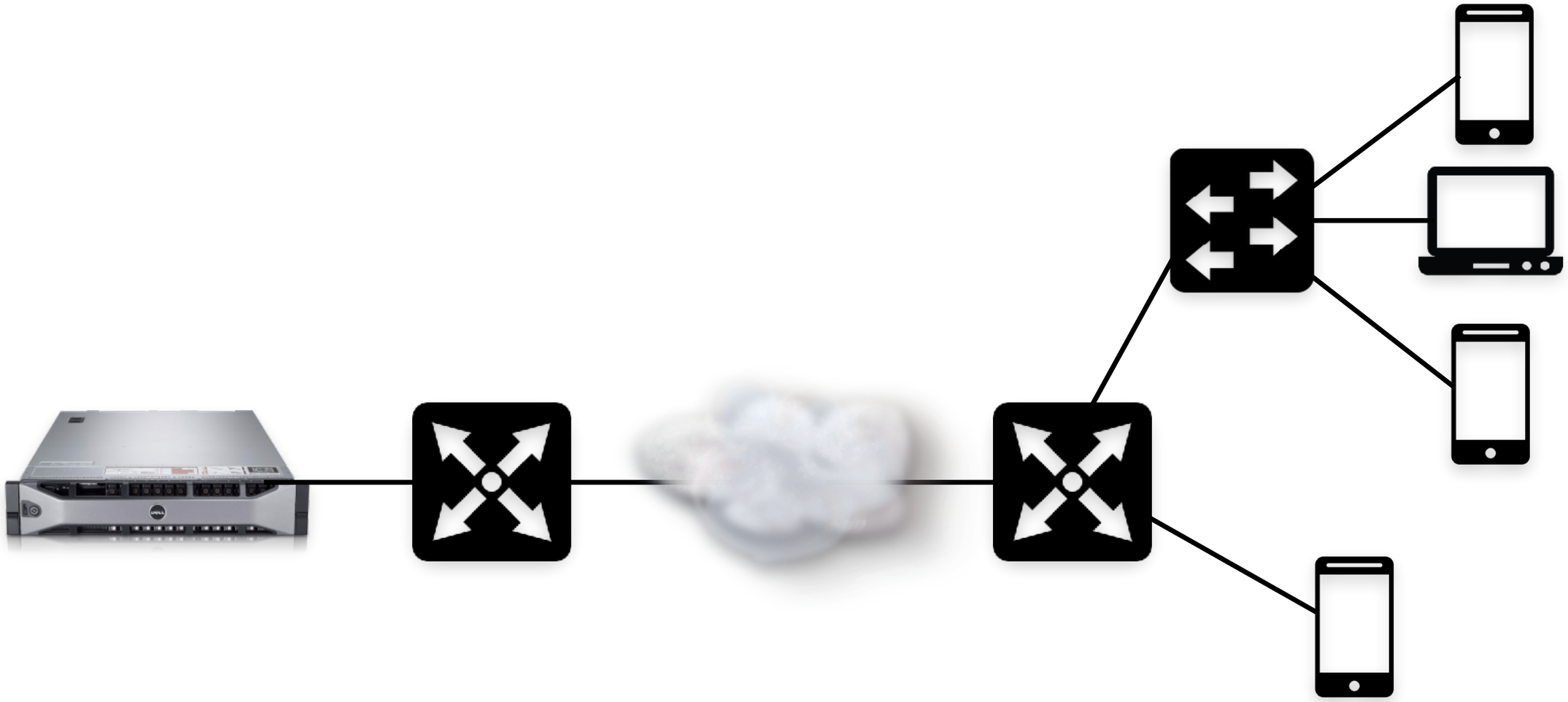
4

Part4: Future of applications stack
- merging of security and transport layer
- multicast

Future of applications stack

2000	2020	202X
HTTP/1.1	HTTP/**2.0**	HTTP/2.0
SSL/TLS	TLS **1.3**	**QUIC**
TCP	TCP	
IPv4	IP**v6**	IPv6

IPv6 multicast over UDP/QUIC



Conclusion

Alexander, now it's time to finish

I’ve heard everything, what should I do now?

1. improve speed of mobile connection
2. check TFO, send/recv buffer, cwnd initial size
3. select custom congestion control
4. collect statistics
5. try UDP protocols or QUIC

Contacts

Thank you

Alexander Tobol

Head of engineering, Video and News Feed Platforms

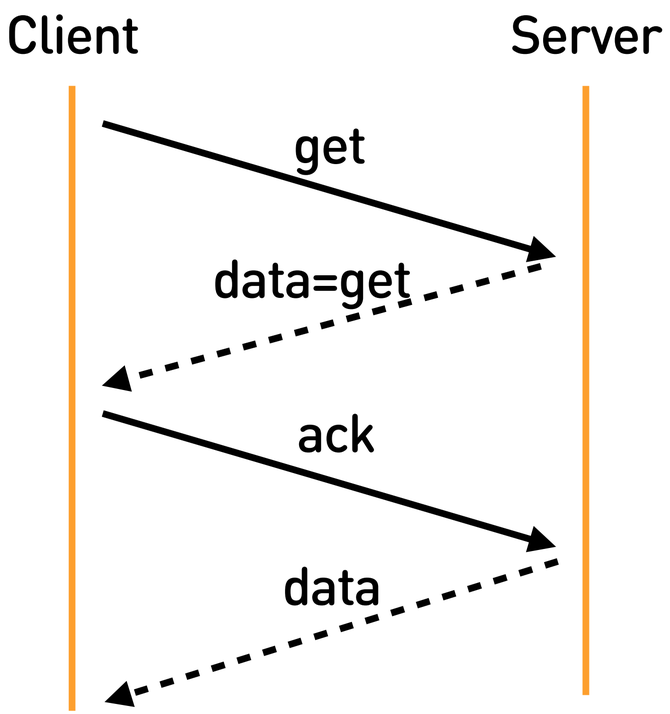
OK.RU social network

alexander.tobol@corp.mail.ru

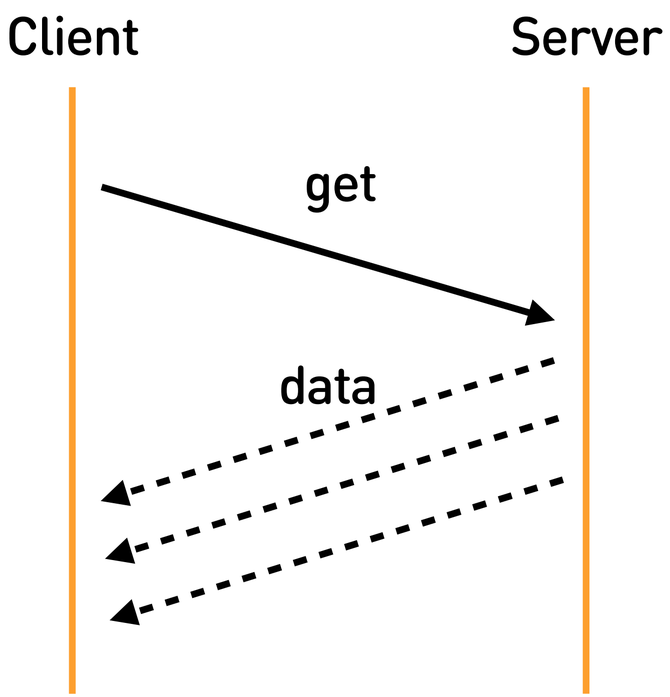
true ZeroRTT

or zeroRTT vs UDP amplification

zeroRTT & UDP Amplification



QUIC



UT2

! fallback to QUIC