Politecnico di Bari
Dipartimento di
Ingegneria Elettrica
e dell'Informazione

C3LAB
Control of Computing
and Communication
Systems Lab

HTTP over UDP: An experimental investigation of QUIC

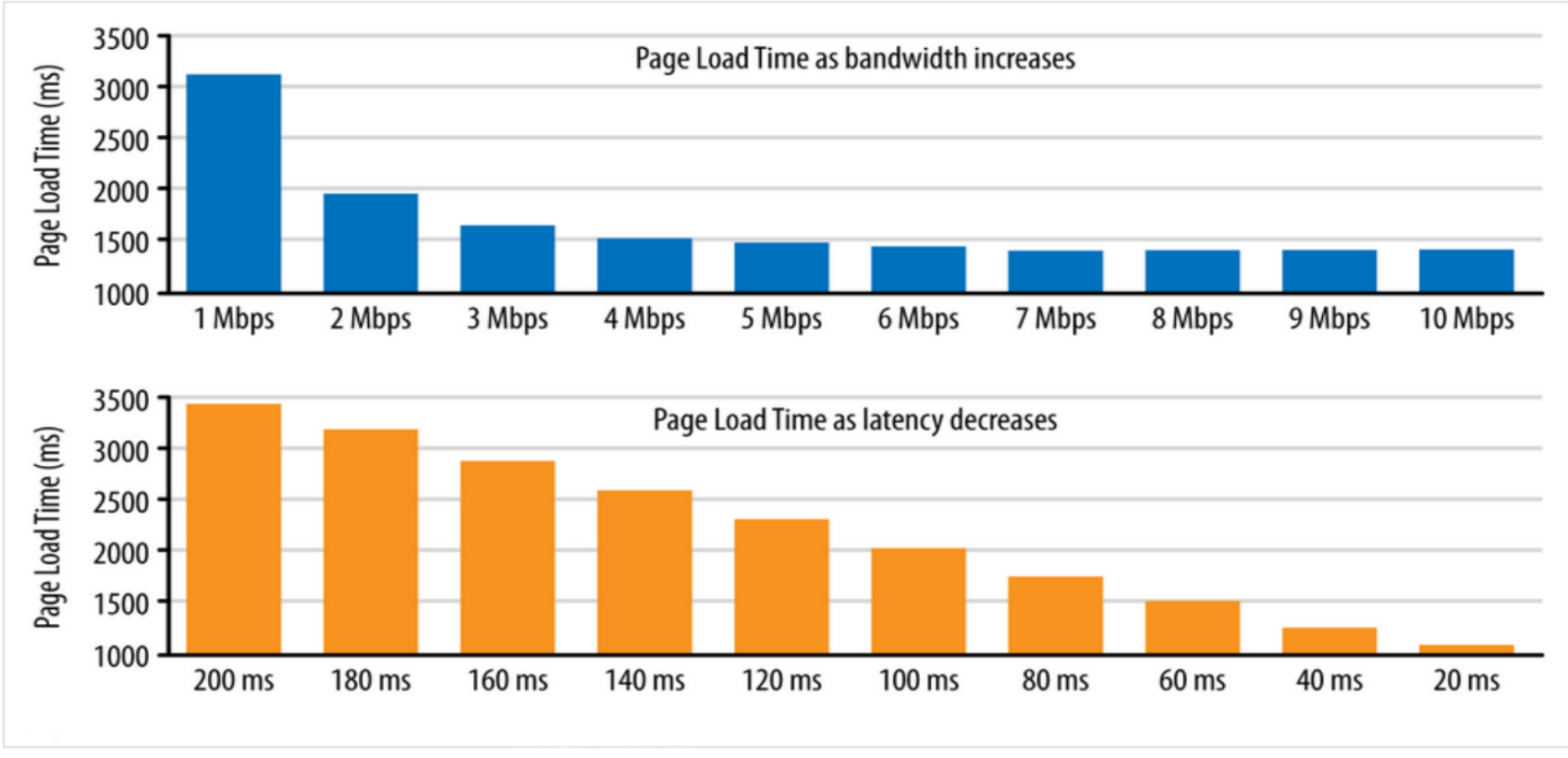
ACM/SIGAPP Symposium On Applied Computing

Salamanca, Spain, 13-17 April 2015

G. Carlucci, L. De Cicco, S. Mascolo

Politecnico di Bari, Italy

MOTIVATION



Grigorik Ilya, High Performance Browser Networking, O'Reilly Media, Inc., 2013

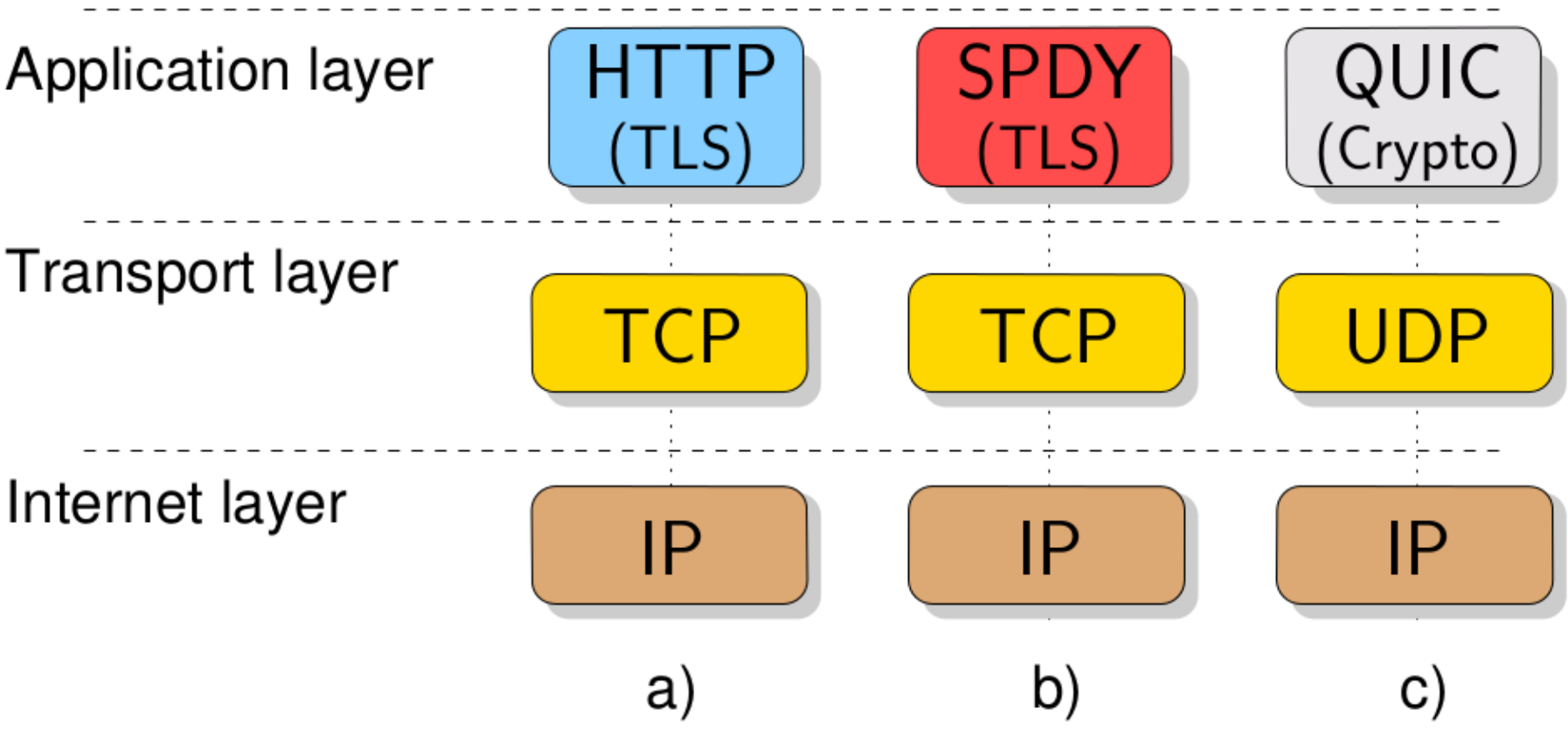
Page Load Time linearly decreases with Internet latency, while a further increase of the link capacity does not introduce any benefit

GOAL: REDUCING HTTP LATENCY

HTTP INEFFICIENCIES HINDERING A FASTER INTERNET

- HTTP can only fetch one resource at a time
- HTTP implements a client-server pull-based communication model; if the server knows a client needs a resource, there is no mechanism to push the content to the client;
- HTTP redundantly sends several headers on the same channel

NEW PROPOSALS TO REDUCE LATENCY: SPDY and QUIC



SPDY is an evolution of HTTP/TCP

Goal of QUIC: replace HTTP over TCP with QUIC over UDP

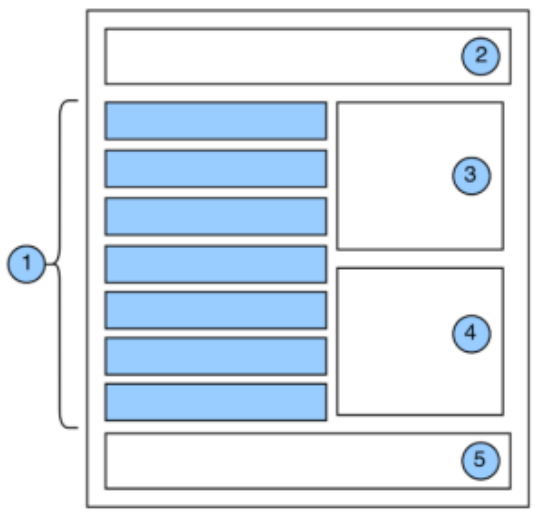
(Note: QUIC always encrypts data)

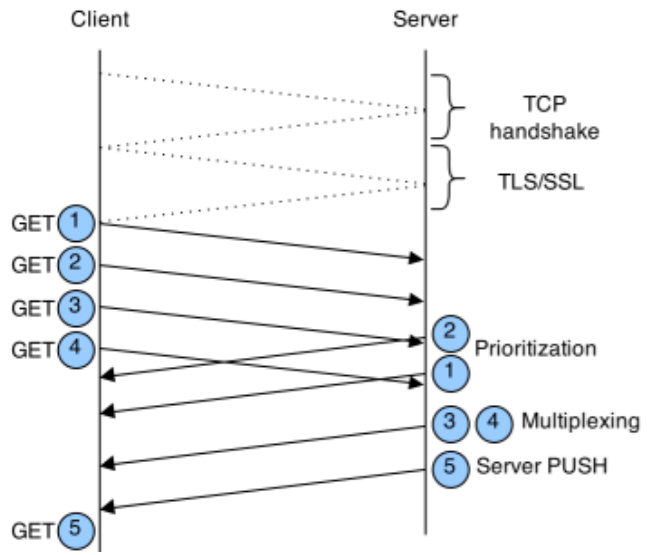
SPDY FEATURES

- **HTTP requests are multiplexed** on a single TCP socket to preserve server resources
- **HTTP headers compression**
- **Server Push**: server can push data to the client when possible
- **Requests prioritization**

Example:

Web Page resuorces





SPDY LIMITS

SPDY has already shown some limits:

- SPDY multiplexes streams over a single TCP connection, a disadvantage wrt opening parallel HTTP/1.1 connections each one with a separate congestion window

- an out-of-order packet delivery for TCP induces head of line blocking for all the SPDY streams multiplexed on that TCP connection

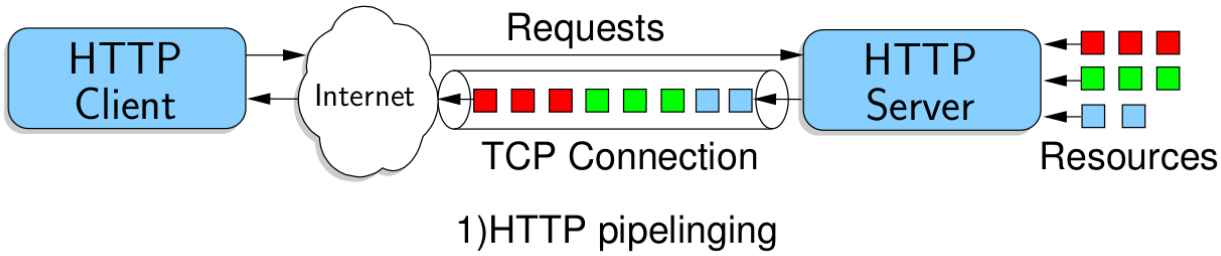
- SPDY connection startup latency depends on TCP handshake which requires one RTT or up to three RTTs if SSL/TLS is employed

These inefficiencies are due to the fact that SPDY employs TCP

The main Idea

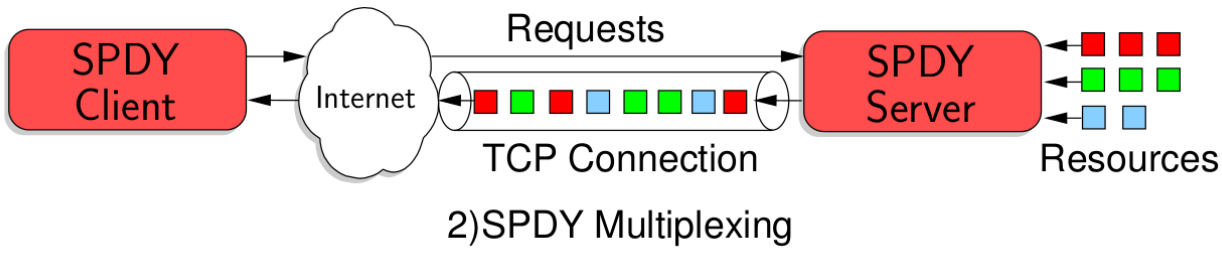
QUIC over UDP to replace HTTP over TCP

QUIC leverages most of the SPDY design choices to inherit benefits (i.e. Multiplexing)



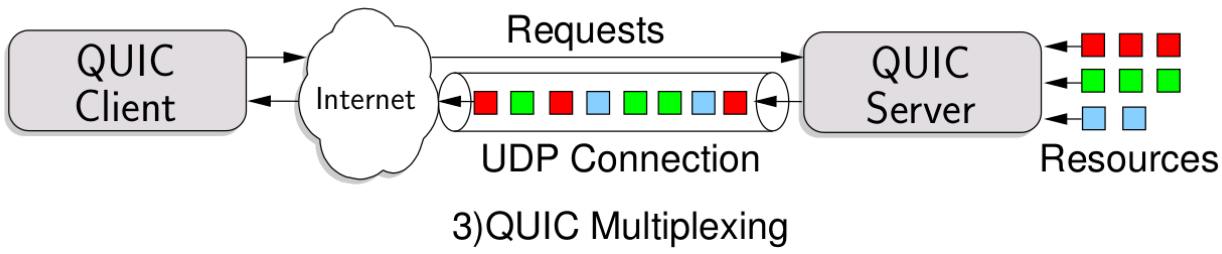
1)HTTP pipelinging

A client can fetch one resource at a time, which are are pipelined; to reduce Page Load Time, parellel TCP connections can be opened at the cost of ports consumption.



2)SPDY Multiplexing

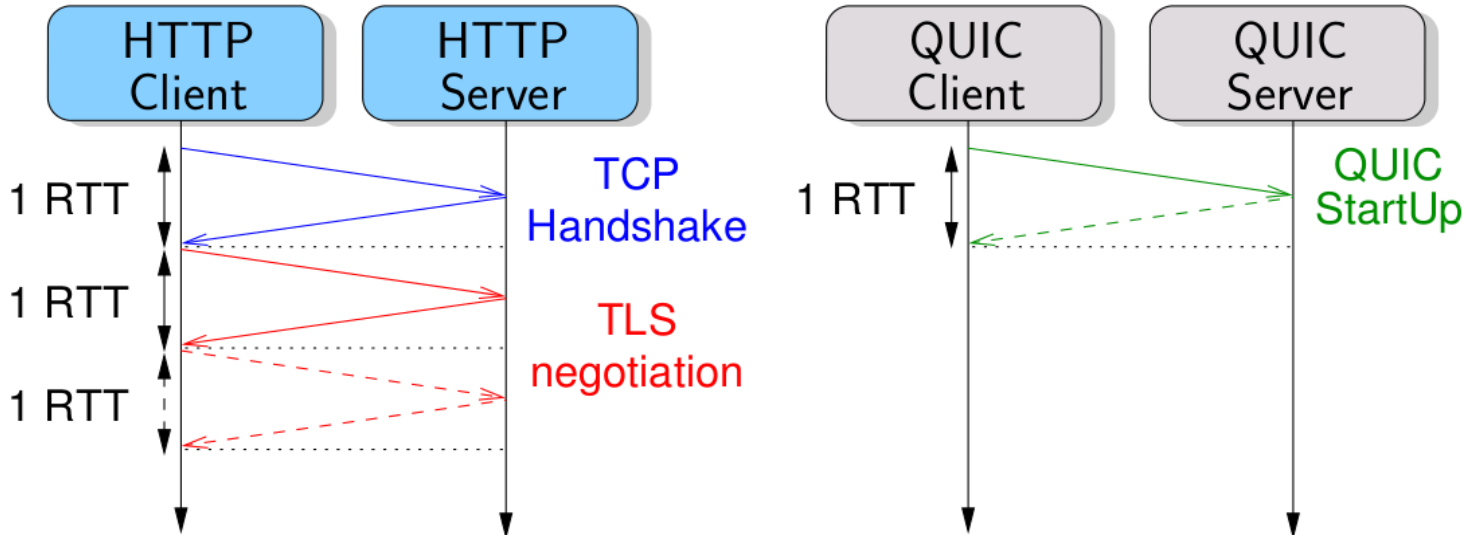
SPDY allows multiple requests/ responses be multiplexed on a TCP pipe; HOL still holds: if a packet is lost, the following packets must wait until the lost packet is retransmitted (TCP does not allow out-of-order delivery).



3)QUIC Multiplexing

QUIC replaces TCP with UDP. This solves the HOL issue. QUIC implements retransmissions and congestion control at the application layer.

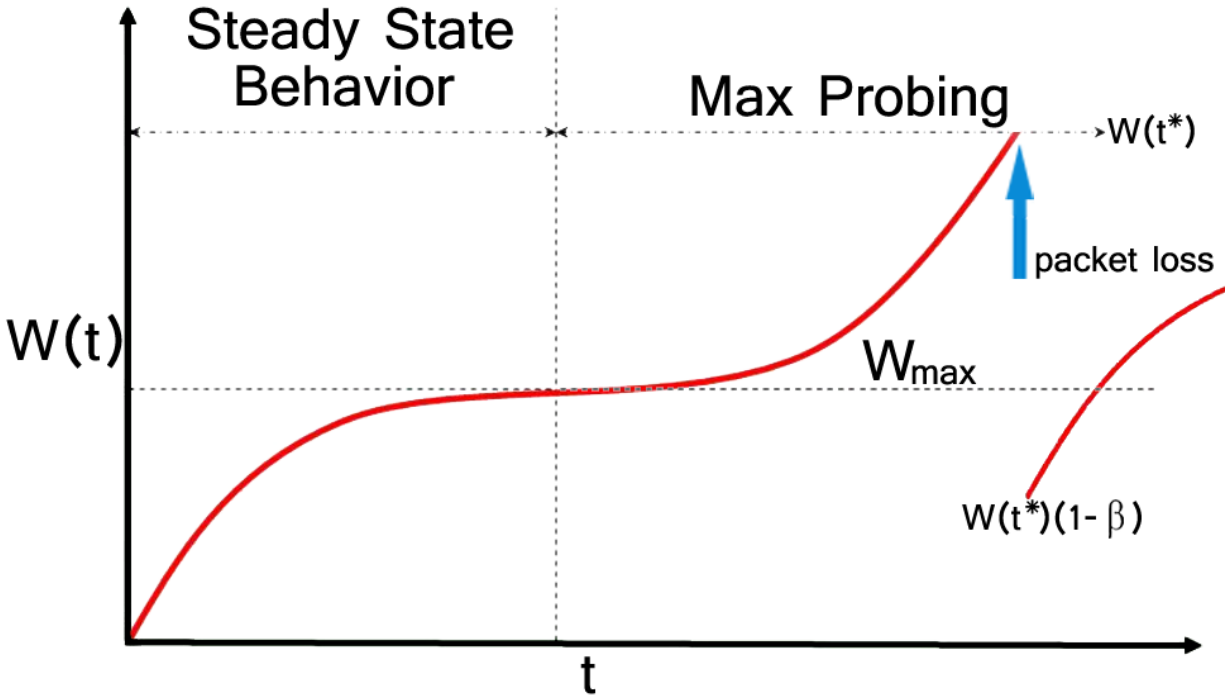
QUIC START-UP LATENCY



0-RTT latency: if the QUIC client has already talked to the QUIC server, the handshake is not required (thanks to QUIC-Crypto)

QUIC PLUGGABLE CONGESTION CONTROL

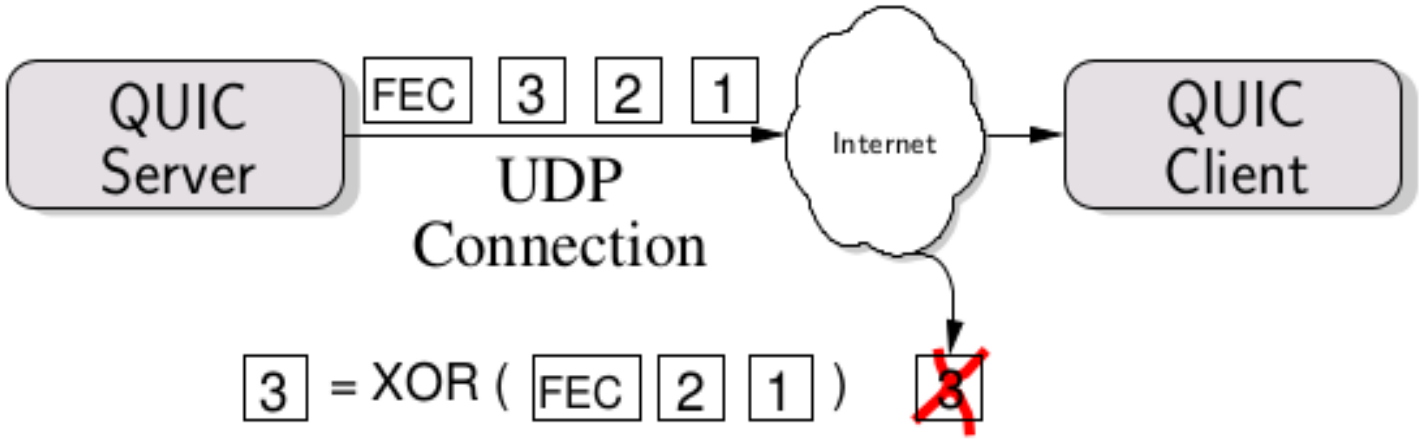
In the current implementation only TCP CUBIC algorithm is supported.



β_{TCP} = 0.3 whereas in β_{QUIC} = β_{TCP}/n, where n = 2.
In practice, β_{QUIC} = 0.15 is equivalent to β_{TCP} when 2 concurrent TCP CUBIC flows compete for the available bandwidth.

QUIC FEC

It has a Forward Error Correction (FEC) module in order to recover lost packets without asking for a retransmission



The FEC module can be useful to further reduce the head of line blocking in high RTT network.

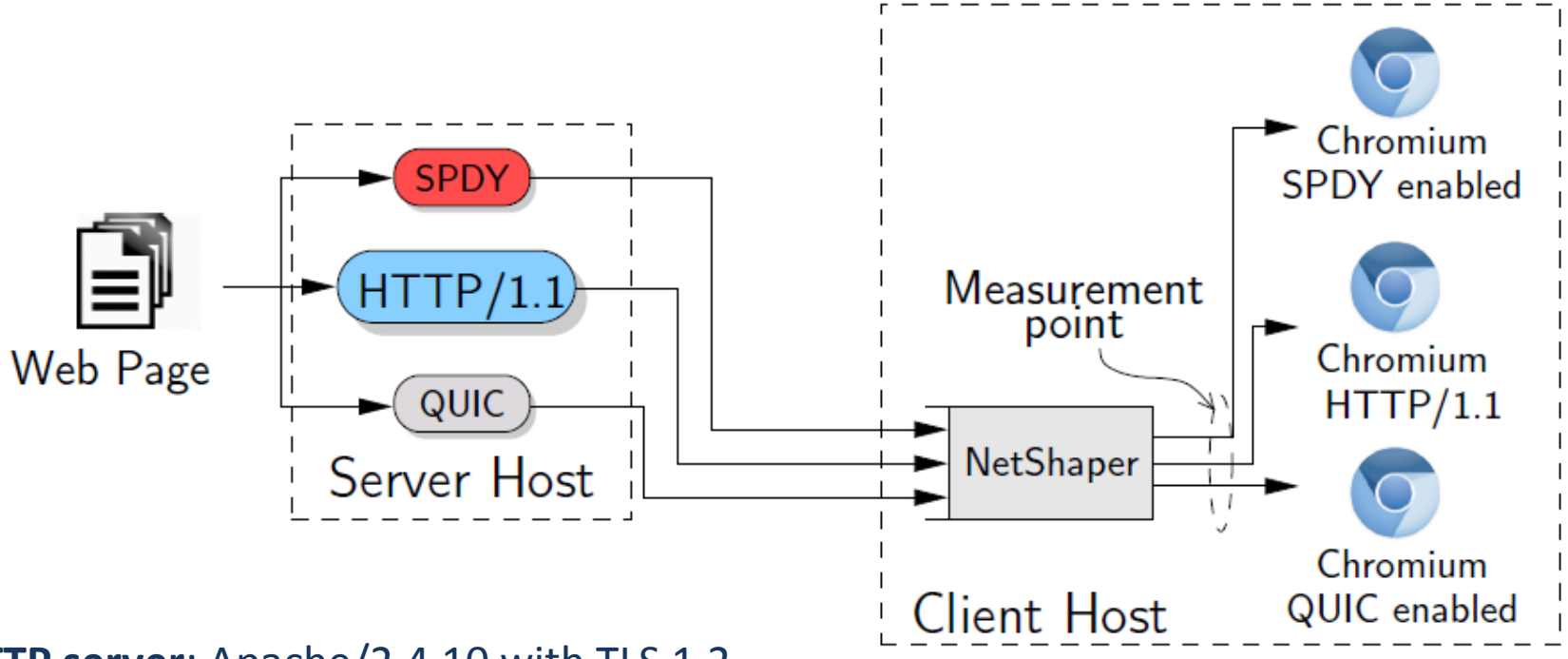
FOCUS OF THE PAPER

Through experiments, we have investigate:

Can QUIC be safely deployed in the Internet?

Does QUIC reduce the Web Page Load Time wrt to SPDY and HTTP?

THE TESTBED



HTTP server: Apache/2.4.10 with TLS 1.2.

SPDY server: nghttp2 with TLS 1.2.

QUIC server: (version 21) server in Chromium code base with *QUIC- Crypto*

Netshaper: sets channel capacity and propagation delays.

Real experiments using the open source browser **Chromium**.

METRICS

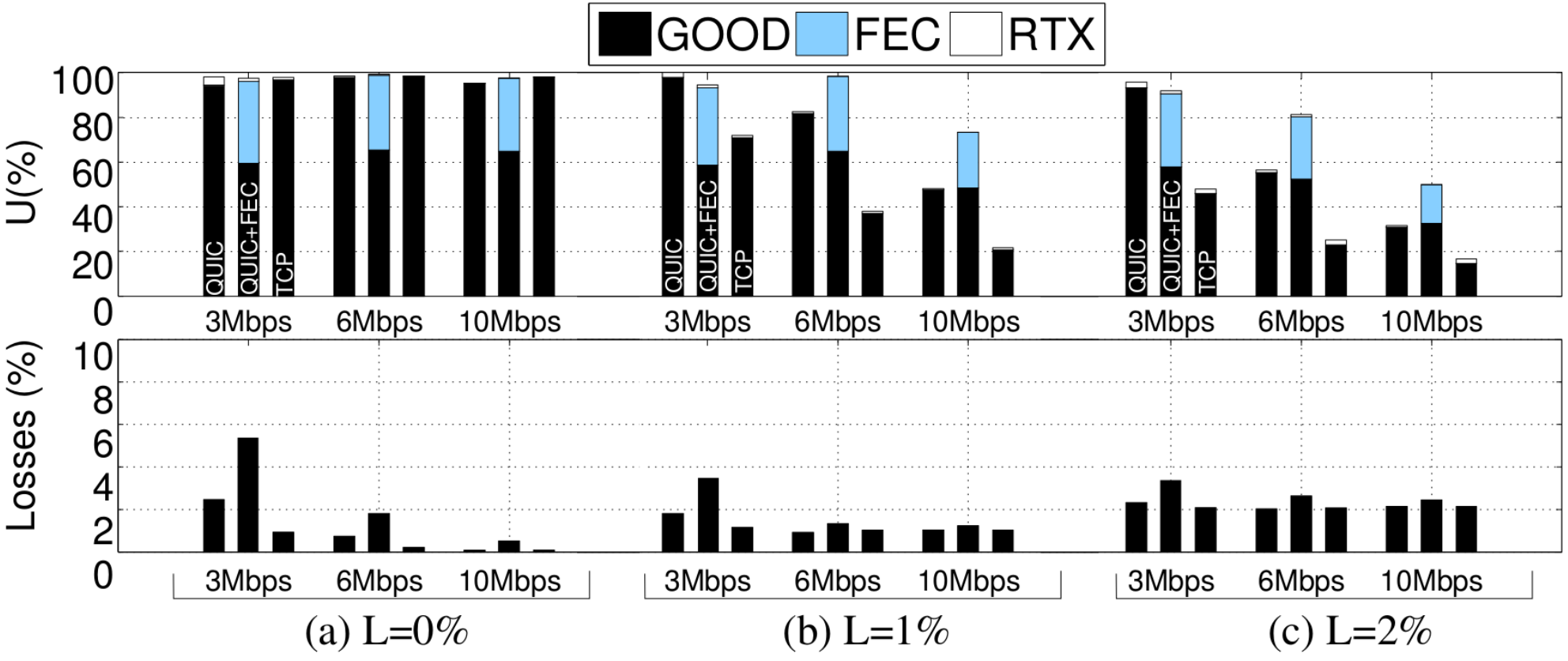
- **Goodput G**, measured as the average network received rate (without considering retransmissions and FEC)
- **Channel Utilization U**, measured as r/b, where r is the average received rate and b the link capacity
- **Loss Ratio l,** defined as (lost_byte/sent_byte) measured by the NetShaper tool
- **Page load time P,** defined as the time taken by the browser to download and process all the objects in a Web page.

EXPERIMENTAL SCENARIOS

- **Scenario 1:** Impact of link capacity, induced random losses and FEC (ON/OFF) on a single QUIC flow over a bottleneck
- **Scenario 2:** QUIC vs TCP CUBIC varying the bottleneck buffer sizes
- **Scenario 3:** Page Load Time comparison varying the Web page size

SCENARIO 1: Single QUIC flow

Impact of link capacity, induced random losses and FEC (ON/OFF)

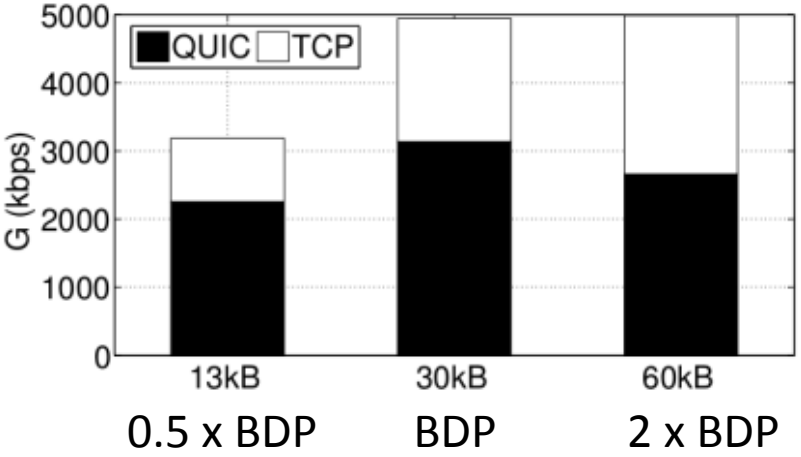


FEC worsens the performance of QUIC

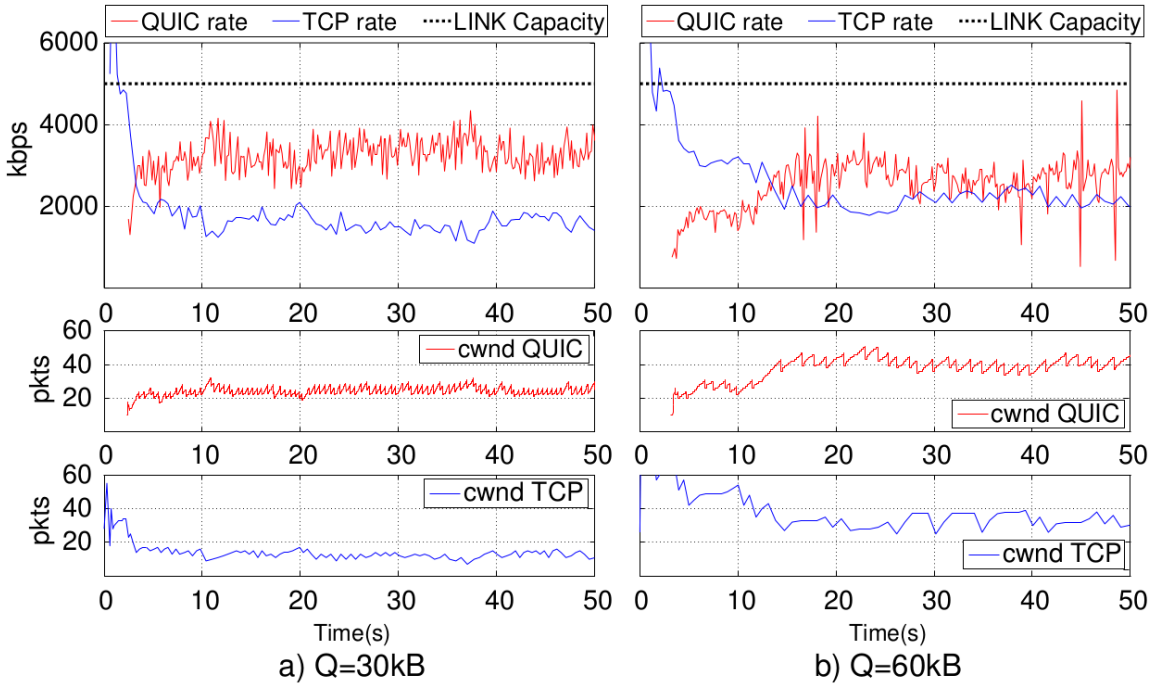
SCENARIO 2: one QUIC and one TCP Cubic flow sharing the bottlelenck

Considered bottleneck buffer sizes: Q = {0.5 BDP, BDP, 2 BDP}
Bottleneck Capacity: 5 Mbps

Bandwidth Share

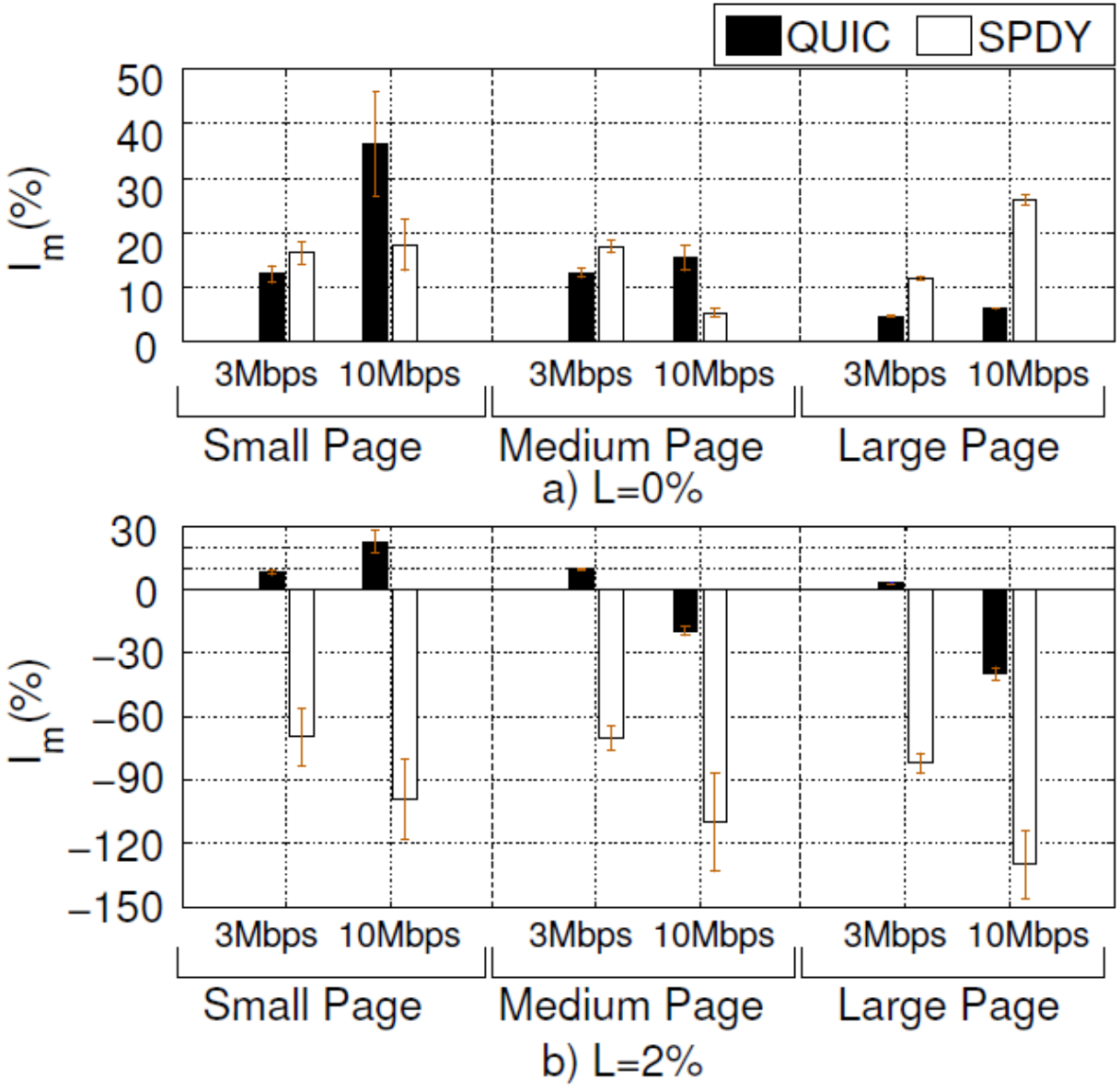


Dynamics



QUIC prevails over TCP CUBIC in under-buffered networks

SCENARIO 3: Page Load Time comparison varying the Web page size



Metric: Page Load Time Improvement wrt HTTP

$$I_m = \frac{P_{HTTP} - P_{QUIC/SPDY}}{P_{HTTP}} \cdot 100$$

QUIC better than HTTP when the channel is loss free (spdy multiplexing (40) larger than quic (6) at the time experiments)

QUIC better than SPDY in the case of a lossy channel.

CONCLUSION

Experimental investigation of QUIC has shown:

- QUIC/UDP could be a promising protocol to replace HTTP/TCP
- There are several issues that require further investigations:
  - Is Cubic like congestion control appropiate for low latency?
  - QUIC FEC module, when enabled, worsens the overall protocol performances
- It appears that QUIC reduces the overall page retrieval time wrt HTTP and SPDY in case of a channel with/without random losses

CONCLUSION

QUESTIONS?

THANK YOU