



Politecnico di Bari

Dipartimento di
Ingegneria Elettrica
e dell'Informazione



C3LAB

Control of Computing
and Communication
Systems Lab

QUIC evaluation

HTTP Workshop

28 July 2015

Münster - Germany

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

Politecnico di Bari, Italy

FOCUS OF THE TALK

We want to answer to these questions:

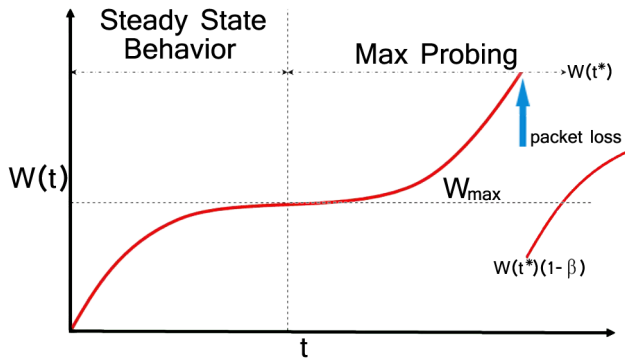
Can QUIC be safely deployed in the Internet?

Does QUIC reduce the Web Page Load Time wrt
to HTTP/2.0 and HTTP/1.1?

Through an experimental investigation.

QUIC - PLUGGABLE CONGESTION CONTROL

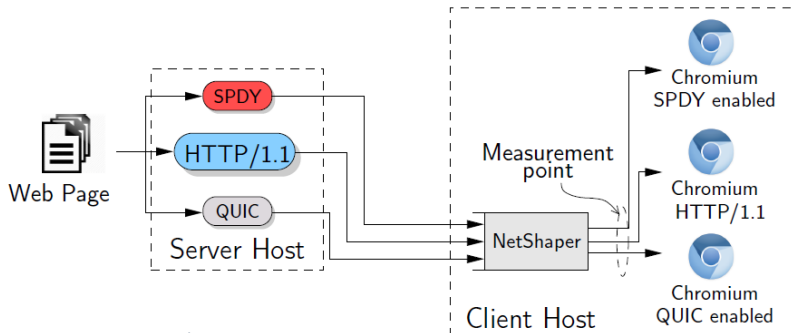
In the current implementation TCP CUBIC algorithm is supported by default.



$$IW = 10$$

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

THE TESTBED



HTTP server: Apache/2.4.10 with TLS 1.2.

SPDY server: nghttp2 with TLS 1.2. We have employed nghttp2 which supports SPDY 4 that is the Google implementation of HTTP/2.0.

QUIC server: (version 24) server in Chromium code base with *QUIC-Crypto*

Netshaper: sets channel capacity and propagation delays.

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

METRICS

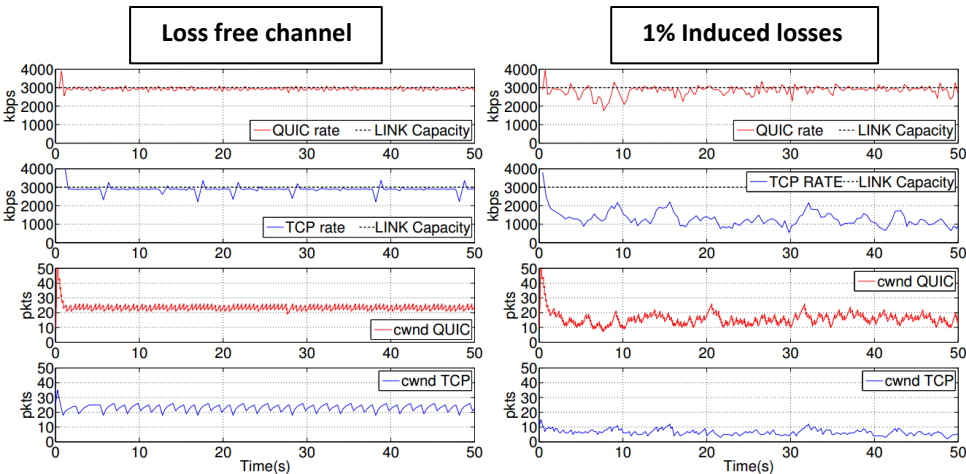
- ▶ **Goodput G**, measured as the average network received rate (without considering retransmissions and FEC)
- ▶ **Channel Utilization U**, measured as r/b , where r is the average received rate and b the link capacity
- ▶ **Loss Ratio I**, defined as $(\text{lost_byte}/\text{sent_byte})$ measured 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 Dynamics - Impact on induced losses

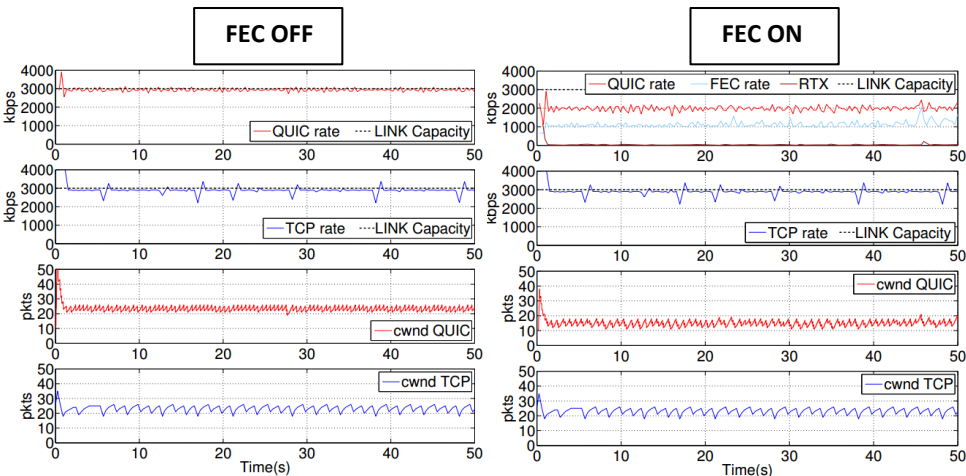
Bottleneck buffer sizes: $Q = \text{BDP}$ - Base RTT: 50ms - Bottleneck Capacity: 3 Mbps
- FEC: OFF



QUIC shows higher throughput over a lossy link

SCENARIO 1: Single QUIC Dynamics - Impact on FEC

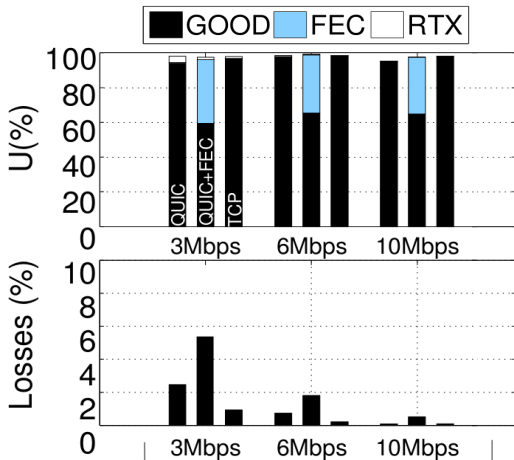
Bottleneck buffer sizes: $Q = \text{BDP}$ - Base RTT: 50ms - Bottleneck Capacity: 3 Mbps
- Loss free channel



When FEC is ON, one FEC packet is sent roughly every 2 data packets

SCENARIO 1: Single QUIC flow

Loss free channel

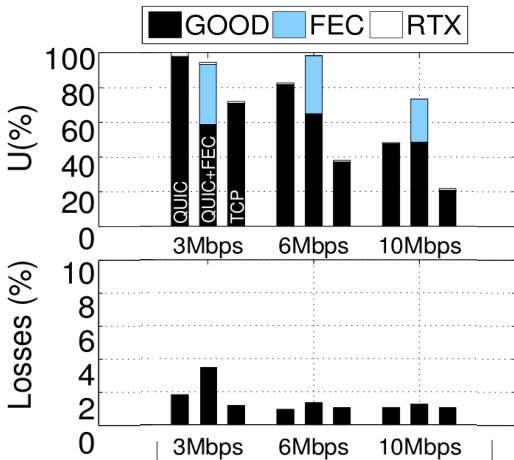


Other parameters:

- Bottleneck buffer sizes: $Q = \text{BDP}$
- Base RTT: 50ms
- Bottleneck Capacity: {3,6,10} Mbps
- FEC: {ON, OFF}

SCENARIO 1: Single QUIC flow

Lossy link: 1% induced random losses

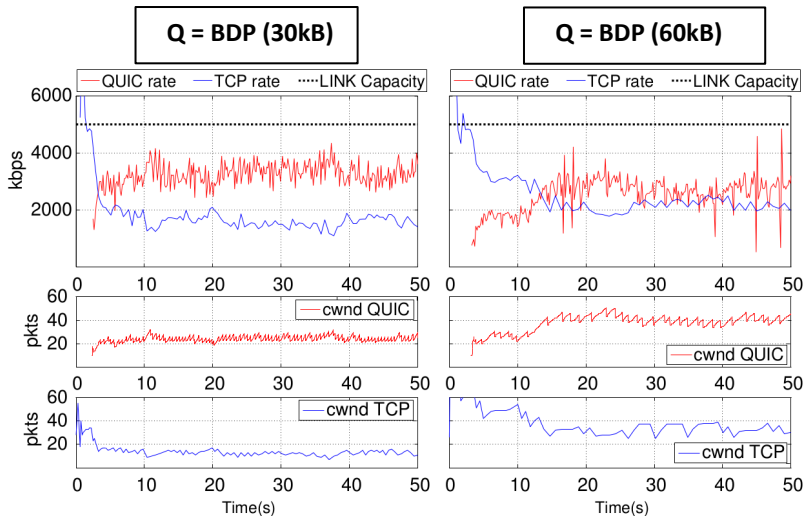


Other parameters:

- Bottleneck buffer sizes: $Q = \text{BDP}$
- Base RTT: 50ms
- Bottleneck Capacity: {3,6,10} Mbps
- FEC: {ON, OFF}

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

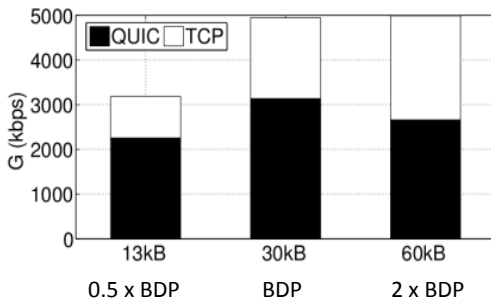
Base RTT: 50ms - Bottleneck Capacity: 5 Mbps - Loss free channel - FEC OFF



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

Base RTT: 50ms - Bottleneck Capacity: 5 Mbps - Loss free channel - FEC OFF

Bandwidth Share



QUIC prevails over TCP CUBIC in under-buffered networks

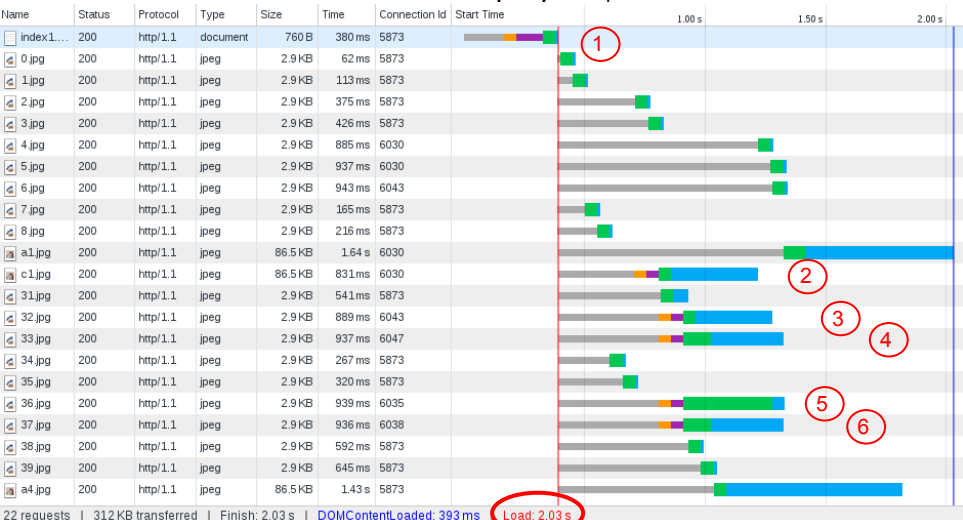
SCENARIO 3: Page Load Time Comparison

Three Web Pages considered:

- **A Small Web page** which contains 18 jpeg images large 2.6kB and 3 larger jpeg images of 86.5kB. (**307kB**)
- **A Medium Web page** which contains 40 jpeg images large 2.6kB and 7 jpeg images large 86.5kB. (**710kB**)
- **A Large Web page** which contains 200 jpeg images large 2.6kB and 17 jpeg images large 86.5kB. (**2.1MB**)
- No JavaScript code and no css file.

SCENARIO 3: Page Load Time HTTP/1.1 - Waterfall diagram

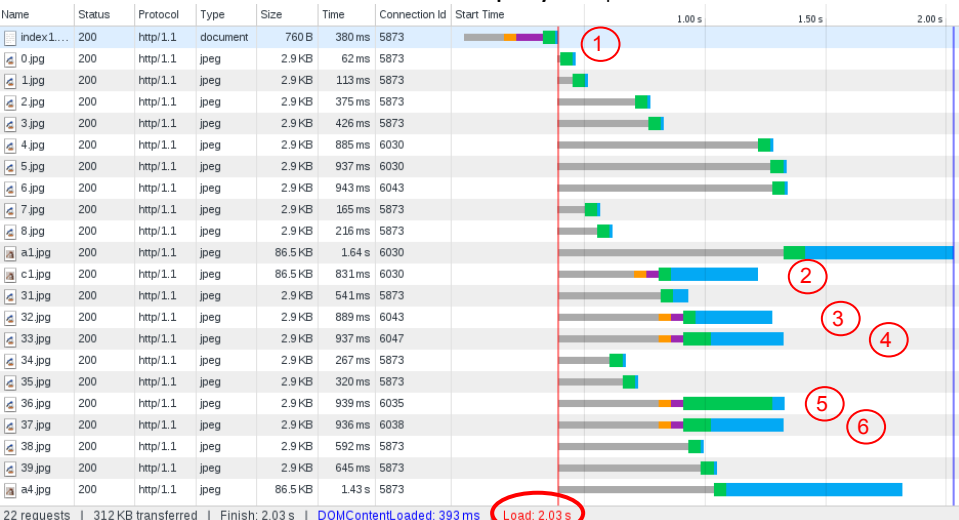
Buffer sizes: Q = BDP - Base RTT: 50ms - Bottleneck Capacity: 3 Mbps - FEC: OFF - Loss free channel



Six TCP connections are opened (each one requires a TCP and TLS handshake)

SCENARIO 3: Page Load Time HTTP/1.1 - Waterfall diagram

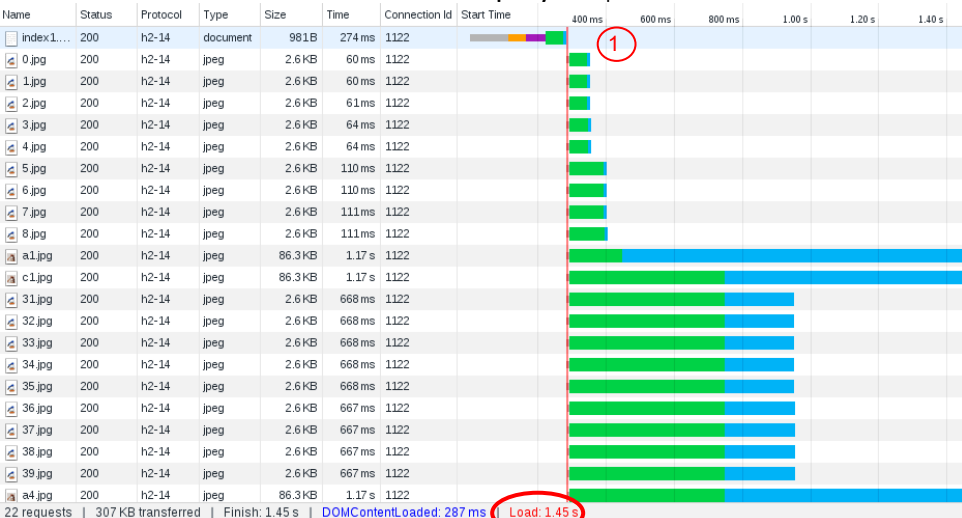
Buffer sizes: Q = BDP - Base RTT: 50ms - Bottleneck Capacity: 3 Mbps - FEC: OFF - Loss free channel



One resource per time is transferred over a connection.

SCENARIO 3: Page Load Time HTTP/2.0 - Waterfall diagram

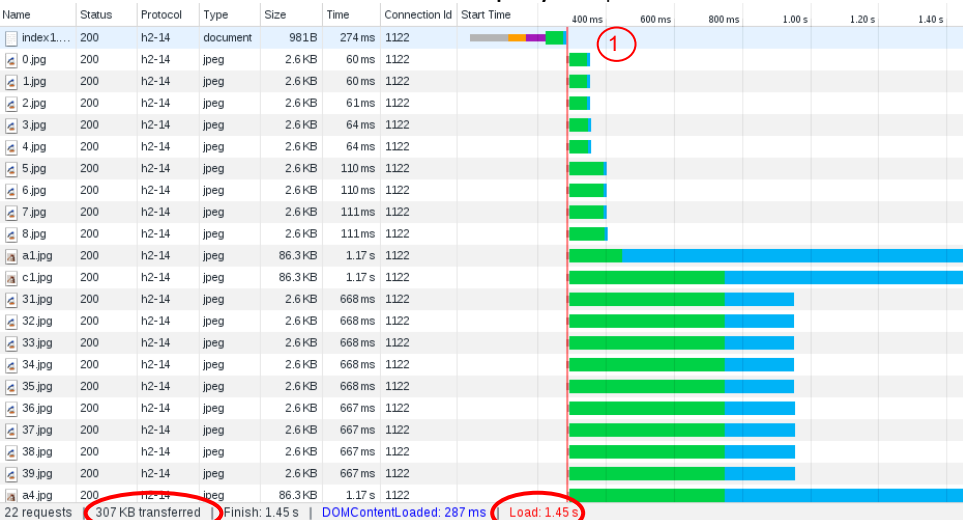
Buffer sizes: Q = BDP - Base RTT: 50ms - Bottleneck Capacity: 3 Mbps - FEC: OFF - Loss free channel



One TCP connection is opened. All the requests are sent when the DOM is loaded.

SCENARIO 3: Page Load Time HTTP/2.0 - Waterfall diagram

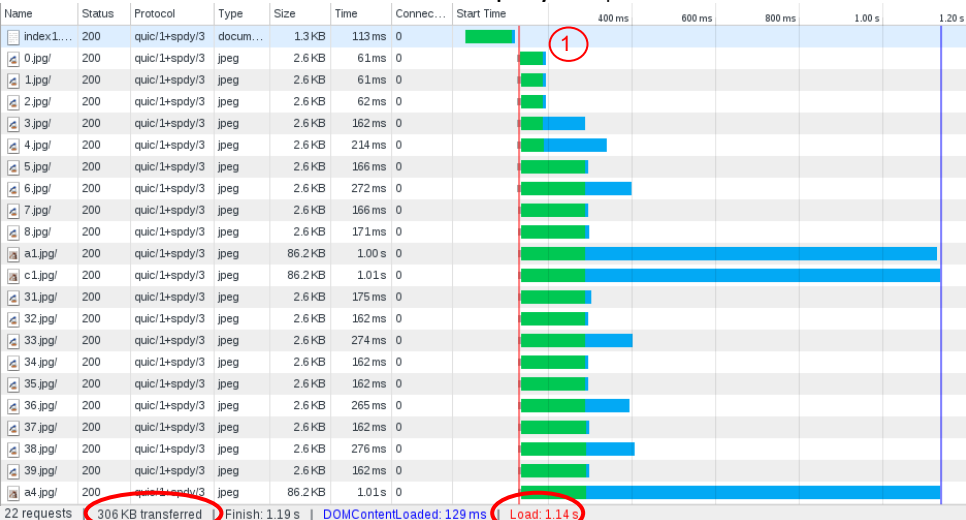
Buffer sizes: Q = BDP - Base RTT: 50ms - Bottleneck Capacity: 3 Mbps - FEC: OFF - Loss free channel



Resources are multiplexed and sent simultaneously over the single TCP connection.

SCENARIO 3: Page Load Time QUIC - Waterfall diagram

Buffer sizes: Q = BDP - Base RTT: 50ms - Bottleneck Capacity: 3 Mbps - FEC: OFF - Loss free channel

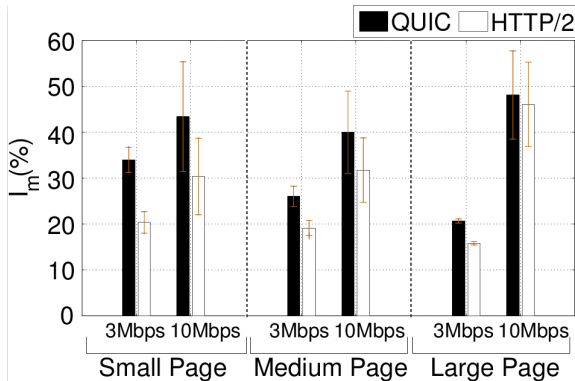


One UDP connection. No handshake required.

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

Buffer sizes: Q = BDP - Base RTT: 50ms - Bottleneck Capacity: {3,10} Mbps - FEC: OFF

Loss free channel



Metric: Page Load Time Improvement wrt HTTP/1.1

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

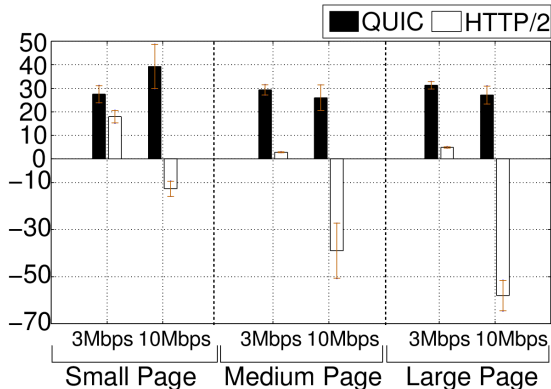
When the page size increases the improvement of QUIC and HTTP/2.0 converges

QUIC and HTTP/2.0 better than HTTP/1.1. QUIC better than HTTP/2.0.

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

Buffer sizes: Q = BDP - Base RTT: 50ms - Bottleneck Capacity: {3,10} Mbps - FEC: OFF

1% Induced losses



Metric: Page Load Time Improvement wrt HTTP/1.1

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

QUIC always better than HTTP/1.1. HTTP/2.0 worse than HTTP/1.1 over a lossy link as the page size increases. (β is smaller, better loss recovery).

CONCLUSION

Experimental investigation of QUIC has shown:

- ▶ QUIC/UDP could be a promising protocol
- ▶ There are several issues that require further investigations:
 - Is Cubic like congestion control appropriate 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 HTTP/2.0 in case of a channel with/without induced random losses

REFERENCES

- Jim Roskind, "[QUIC: Quick UDP Internet Connections Multiplexed Stream Transport over UDP](#)", IETF IETF-88 TSV Area Presentation, Nov 2013
- J. Iyengar, I. Swett, "QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2", IETF Draft, 2015
- J. Iyengar, I. Swett, "QUIC Loss Recovery And Congestion Control", IETF Draft, 2015
- Gaetano Carlucci, Luca De Cicco, Saverio Mascolo, "HTTP over UDP: an experimental investigation of QUIC", Proc. of 30th ACM/SIGAPP Symposium On Applied Computing (SAC 2015), Salamanca, Spain, April 2015

CONCLUSION









QUESTIONS?

THANK YOU

BACKUP

BACK UP

Waterfall diagram explanation

 Stalled/Blocking	Time the request spent waiting before it could be sent. This time is inclusive of any time spent in proxy negotiation. Additionally, this time will include when the browser is waiting for an already established connection to become available for re-use, obeying Chrome's maximum six TCP connection per origin rule.
 Proxy Negotiation	Time spent negotiating with a proxy server connection.
 DNS Lookup	Time spent performing the DNS lookup. Every new domain on a page requires a full roundtrip to do the DNS lookup.
 Initial Connection / Connecting	Time it took to establish a connection, including TCP handshakes/retries and negotiating a SSL.
 SSL	Time spent completing a SSL handshake.
 Request Sent / Sending	Time spent issuing the network request. Typically a fraction of a millisecond.
 Waiting (TTFB)	Time spent waiting for the initial response, also known as the Time To First Byte. This time captures the latency of a round trip to the server in addition to the time spent waiting for the server to deliver the response.
 Content Download / Downloading	Time spent receiving the response data.