QUIC

Context, Design, Deployment

Jana Iyengar

Fastly

ACM SIGCOMM 2017

The QUIC Transport Protocol: Design and Internet-Scale Deployment

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

ABSTRACT

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

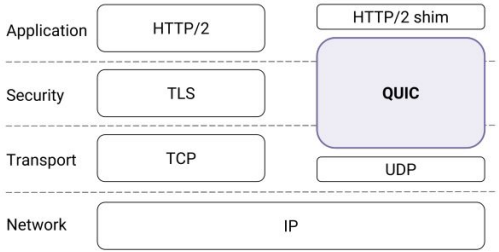


Figure 1: QUIC in the traditional HTTPS stack.

TCP (Figure 1). We developed QUIC as a user-space transport with UDP as a substrate. Building QUIC in user-space facilitated its deployment as part of various applications and enabled iterative changes to occur at application update timescales. The use of UDP

A QUIC history

Protocol for HTTPS transport, deployed at Google starting 2014

Between Google services and Chrome / mobile apps

A QUIC history

Protocol for HTTPS transport, deployed at Google starting 2014

- Between Google services and Chrome / mobile apps

Improved application performance

- YouTube Video Rebuffers: 15 - 18%
- Google Search Latency: 3.6 - 8%

A QUIC history

Protocol for HTTPS transport, deployed at Google starting 2014

Between Google services and Chrome / mobile apps

Improved application performance

YouTube Video Rebuffers: 15 - 18%

Google Search Latency: 3.6 - 8%

35% of Google's egress traffic (7% of Internet)

A QUIC history

Protocol for HTTPS transport, deployed at Google starting 2014

Between Google services and Chrome / mobile apps

Improved application performance

YouTube Video Rebuffers: 15 - 18%

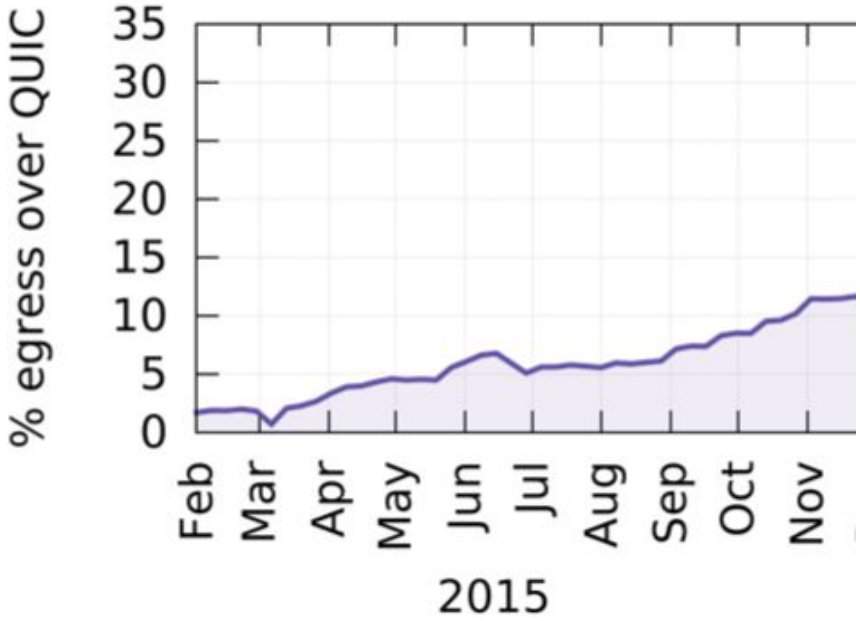
Google Search Latency: 3.6 - 8%

35% of Google's egress traffic (7% of Internet)

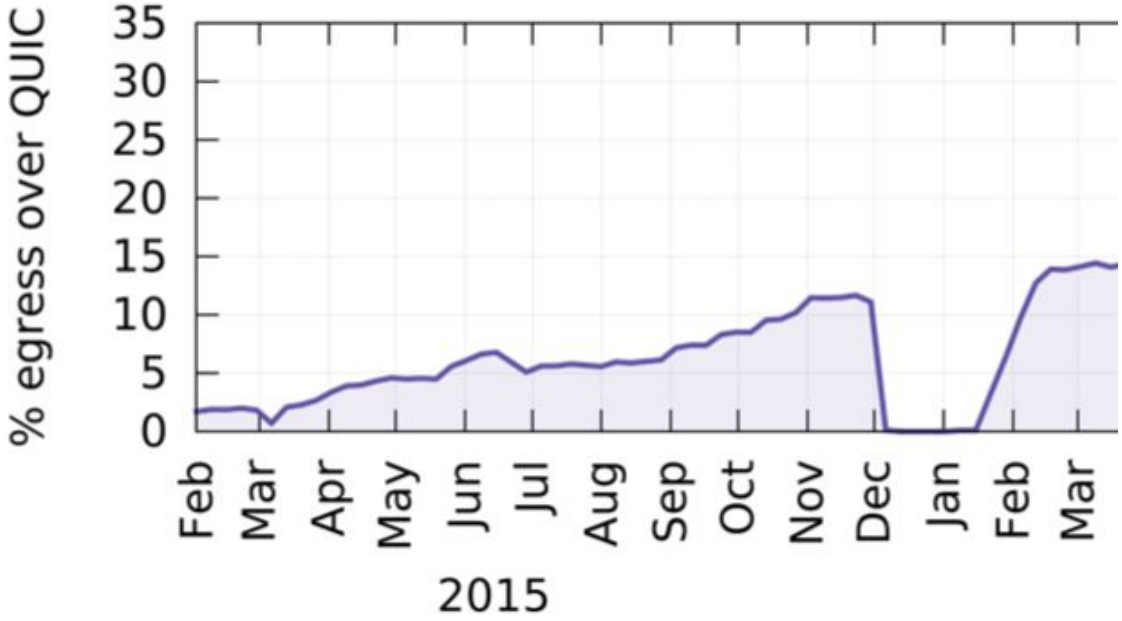
IETF QUIC working group formed in Oct 2016

Modularize and standardize QUIC

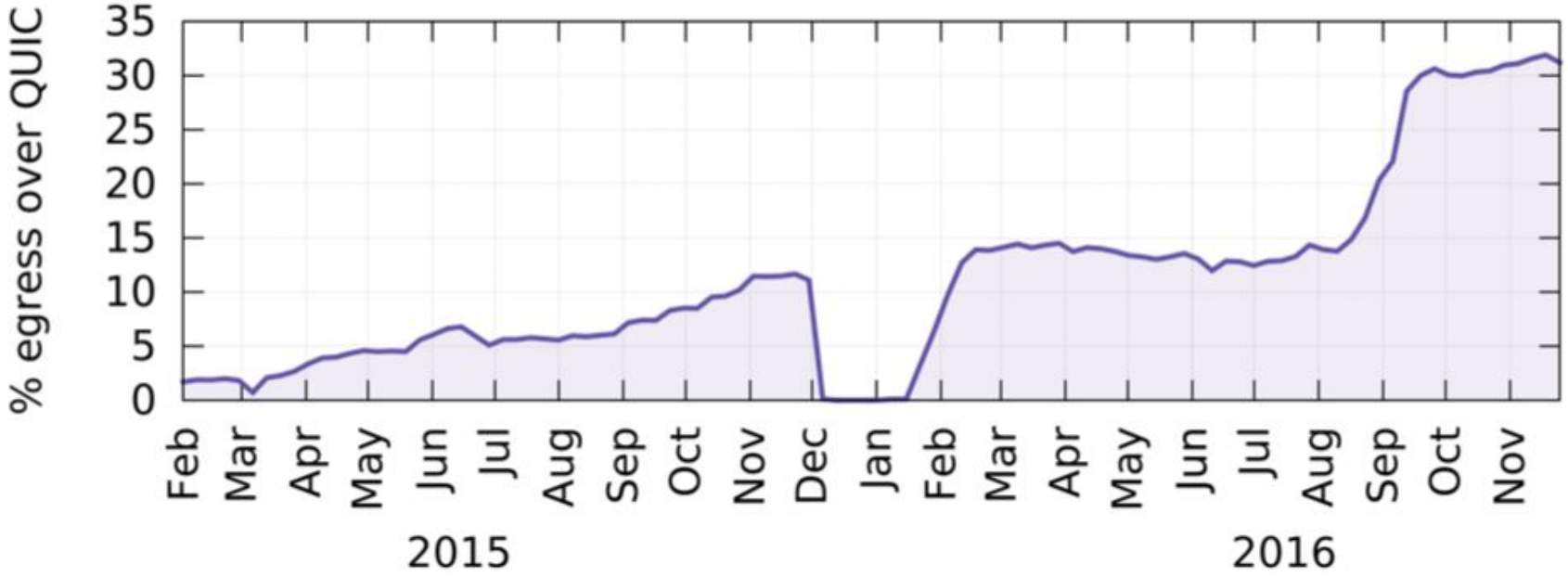
Google's QUIC deployment



Google's QUIC deployment



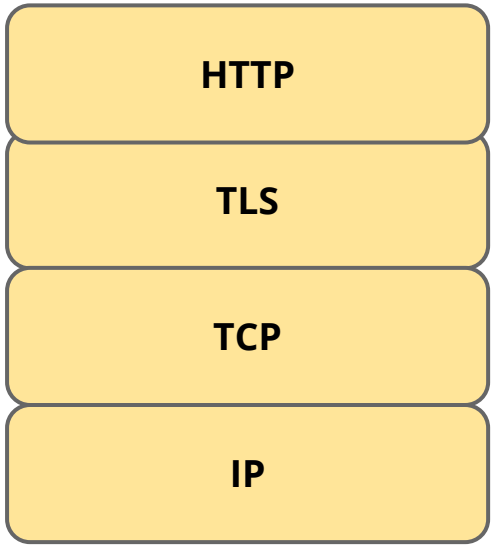
Google's QUIC deployment



"The life of the dead is placed in the memory of the living"
(Cicero)

QUIC comes in a long line of work on web transport

What are we talking about?



The HTTP Story

HTTP/1.0

: independent file transfers
(open, write, close)

The HTTP Story

HTTP/1.0

: independent file transfers
(open, write, close)

The Case for Persistent-Connection HTTP

Jeffrey C. Mogul

May, 1995

Abstract

The success of the World-Wide Web is largely due to the simplicity, hence ease of implementation, of the Hypertext Transfer Protocol (HTTP). HTTP, however, makes inefficient use of network and server resources, and adds unnecessary latencies, by creating a new TCP connection for each request. Modifications to HTTP have been proposed that would transport multiple requests over each TCP connection. These modifications have led to debate over their actual impact on users, on servers, and on the network. This paper reports the results of log-driven simulations of several variants of the proposed modifications, which demonstrate the value of persistent connections.

This Research Report is an expanded version of a paper to appear in the *Proceedings of the SIGCOMM '95 Conference on Communications Architectures and Protocols*.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

digital **Western Research Laboratory** 250 University Avenue Palo Alto, California 94301 USA

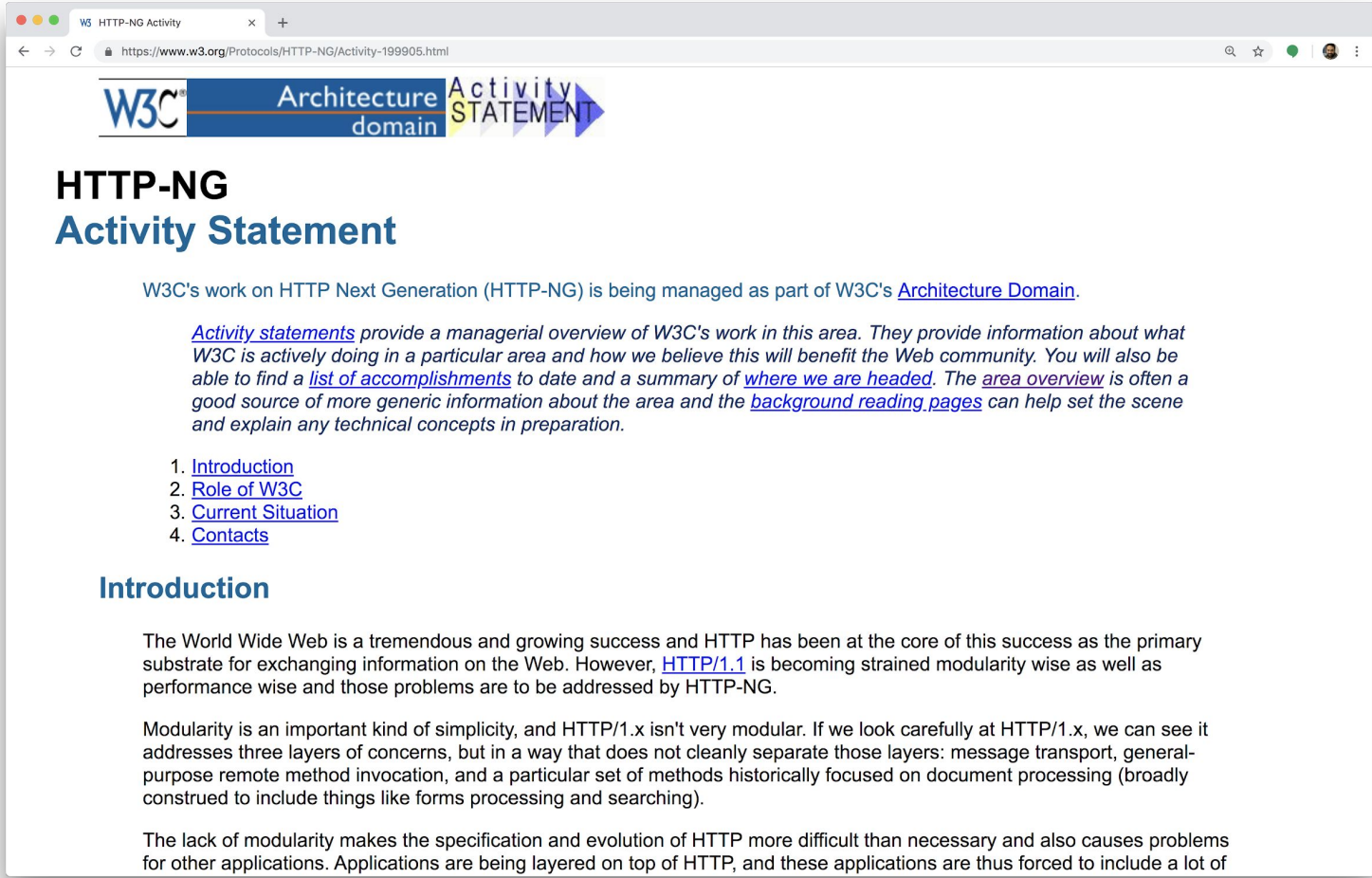
The HTTP Story

HTTP/1.0
HTTP/1.1
: connection persistence
: pipelining

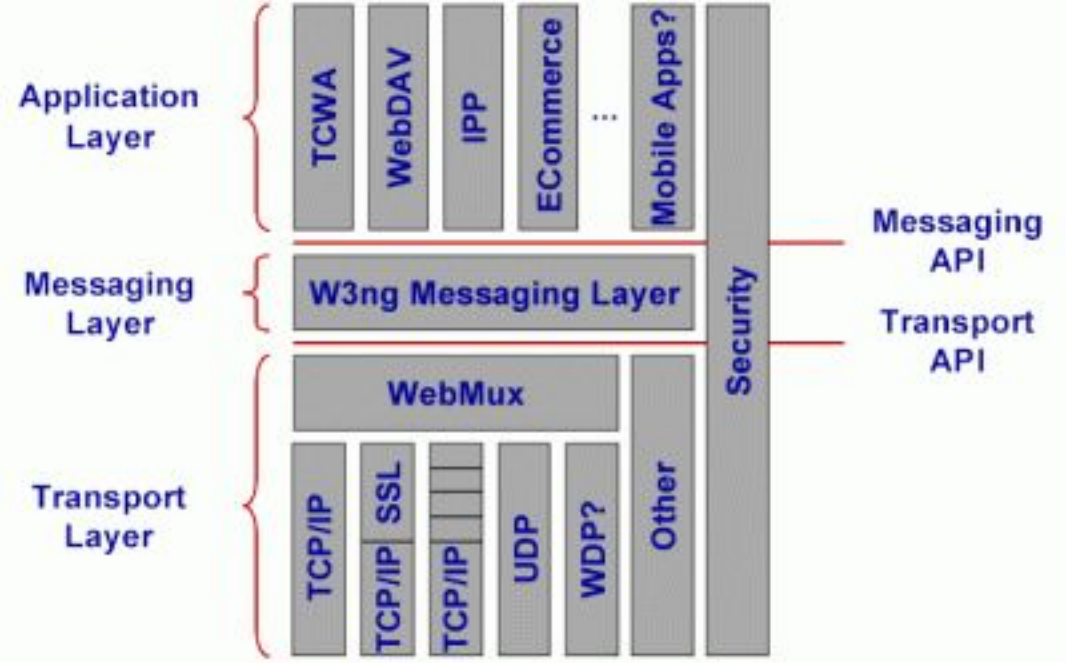
The HTTP Story

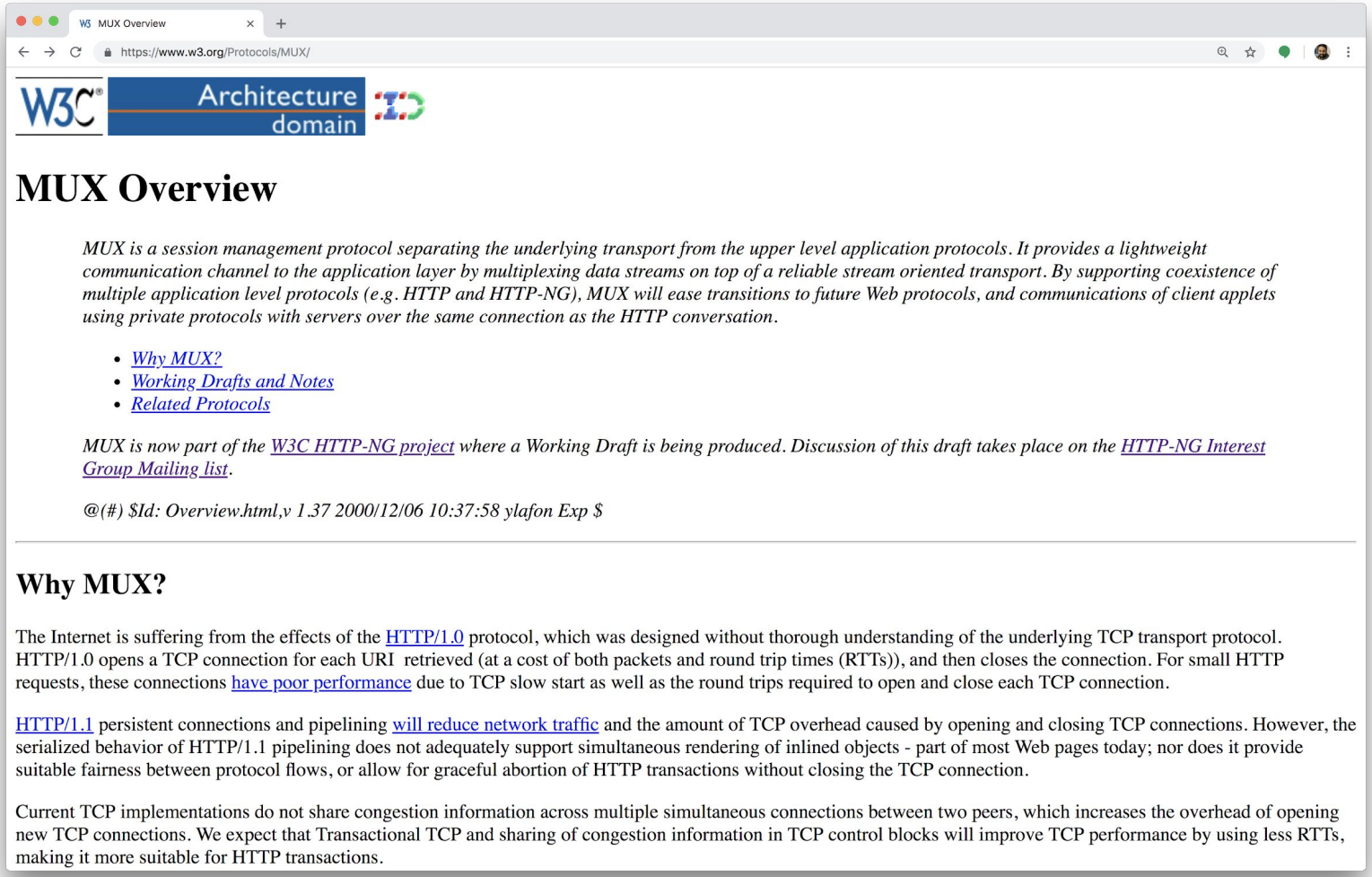
HTTP/1.0
HTTP/1.1

Then around 1998 ...


HTTP-NG Activity
https://www.w3.org/Protocols/HTTP-NG/Activity-199905.html
W3C
Architecture domain
Activity STATEMENT
HTTP-NG
Activity Statement
W3C's work on HTTP Next Generation (HTTP-NG) is being managed as part of W3C's Architecture Domain.
Activity statements provide a managerial overview of W3C's work in this area. They provide information about what W3C is actively doing in a particular area and how we believe this will benefit the Web community. You will also be able to find a list of accomplishments to date and a summary of where we are headed. The area overview is often a good source of more generic information about the area and the background reading pages can help set the scene and explain any technical concepts in preparation.
1. Introduction
2. Role of W3C
3. Current Situation
4. Contacts
Introduction
The World Wide Web is a tremendous and growing success and HTTP has been at the core of this success as the primary substrate for exchanging information on the Web. However, HTTP/1.1 is becoming strained modularity wise as well as performance wise and those problems are to be addressed by HTTP-NG.
Modularity is an important kind of simplicity, and HTTP/1.x isn't very modular. If we look carefully at HTTP/1.x, we can see it addresses three layers of concerns, but in a way that does not cleanly separate those layers: message transport, general-purpose remote method invocation, and a particular set of methods historically focused on document processing (broadly construed to include things like forms processing and searching).
The lack of modularity makes the specification and evolution of HTTP more difficult than necessary and also causes problems for other applications. Applications are being layered on top of HTTP, and these applications are thus forced to include a lot of

“Modularity is an important kind of simplicity, and HTTP/1.x isn't very modular.”




MUX Overview
https://www.w3.org/Protocols/MUX/
W3C
Architecture domain
MUX Overview
MUX is a session management protocol separating the underlying transport from the upper level application protocols. It provides a lightweight communication channel to the application layer by multiplexing data streams on top of a reliable stream oriented transport. By supporting coexistence of multiple application level protocols (e.g. HTTP and HTTP-NG), MUX will ease transitions to future Web protocols, and communications of client applets using private protocols with servers over the same connection as the HTTP conversation.
Why MUX?
Working Drafts and Notes
Related Protocols
MUX is now part of the W3C HTTP-NG project where a Working Draft is being produced. Discussion of this draft takes place on the HTTP-NG Interest Group Mailing list.
@(#) $Id: Overview.html,v 1.37 2000/12/06 10:37:58 ylafon Exp $
Why MUX?
The Internet is suffering from the effects of the HTTP/1.0 protocol, which was designed without thorough understanding of the underlying TCP transport protocol. HTTP/1.0 opens a TCP connection for each URI retrieved (at a cost of both packets and round trip times (RTTs)), and then closes the connection. For small HTTP requests, these connections have poor performance due to TCP slow start as well as the round trips required to open and close each TCP connection.
HTTP/1.1 persistent connections and pipelining will reduce network traffic and the amount of TCP overhead caused by opening and closing TCP connections. However, the serialized behavior of HTTP/1.1 pipelining does not adequately support simultaneous rendering of inlined objects - part of most Web pages today; nor does it provide suitable fairness between protocol flows, or allow for graceful abortion of HTTP transactions without closing the TCP connection.
Current TCP implementations do not share congestion information across multiple simultaneous connections between two peers, which increases the overhead of opening new TCP connections. We expect that Transactional TCP and sharing of congestion information in TCP control blocks will improve TCP performance by using less RTTs, making it more suitable for HTTP transactions.

“[...] these connections have poor performance due to TCP slow start as well as the round trips required to open and close each TCP connection”

“[...] these connections have poor performance due to TCP slow start as well as the round trips required to open and close each TCP connection”

“[...] the serialized behavior of HTTP/1.1 pipelining does not adequately support simultaneous rendering of inlined objects”

“[...] these connections have poor performance due to TCP slow start as well as the round trips required to open and close each TCP connection”

“[...] the serialized behavior of HTTP/1.1 pipelining does not adequately support simultaneous rendering of inlined objects”

“[...] nor does it provide suitable fairness between protocol flows”

“[...] these connections have poor performance due to TCP slow start as well as the round trips required to open and close each TCP connection”

“[...] the serialized behavior of HTTP/1.1 pipelining does not adequately support simultaneous rendering of inlined objects”

“[...] nor does it provide suitable fairness between protocol flows”

“[...] or allow for graceful abortion of HTTP transactions without closing the TCP connection”

“[...] these connections have poor performance due to TCP slow start as well as the round trips required to open and close each TCP connection”

“[...] the serialized behavior of HTTP/1.1 pipelining does not adequately support simultaneous rendering of inlined objects”

“[...] nor does it provide suitable fairness between protocol flows”

“[...] or allow for graceful abortion of HTTP transactions without closing the TCP connection”

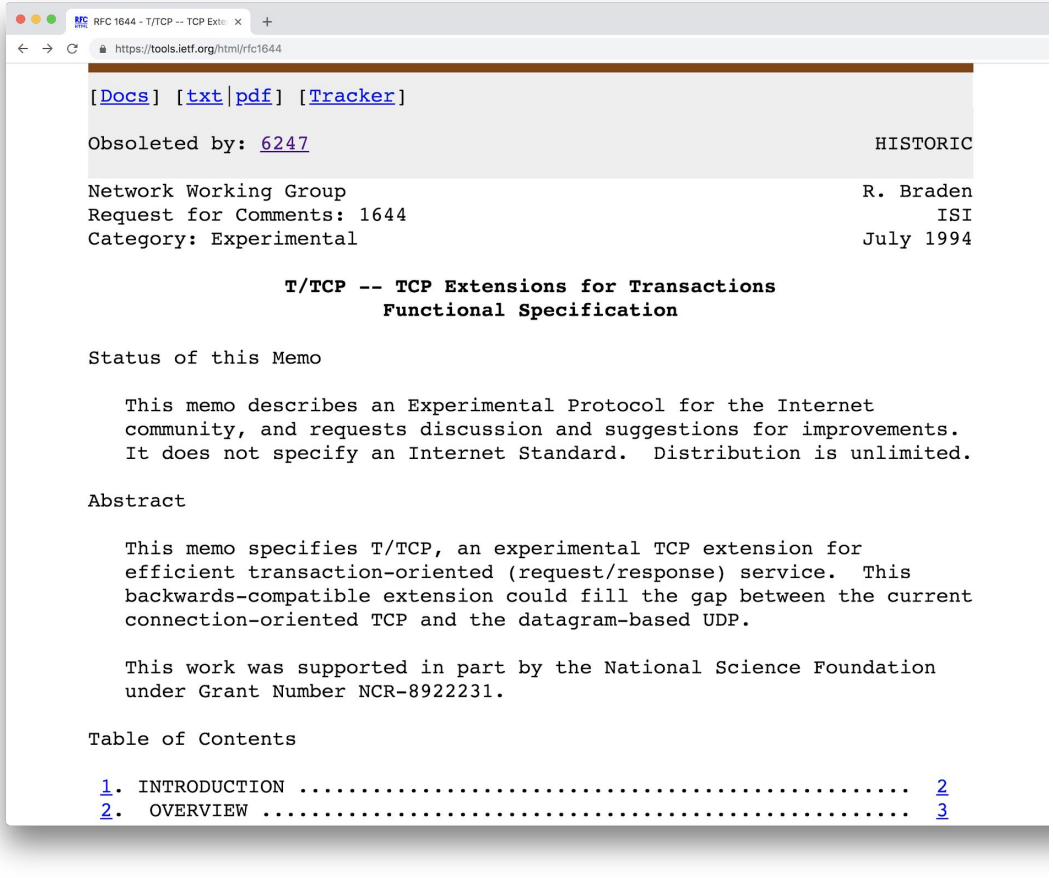
“[...] TCP implementations do not share congestion information across multiple simultaneous connections between two peers”

“[...] multiplexing multiple lightweight HTTP transactions onto the same underlying transport connection and deploying smart output buffer management”

The Transport Story

The Transport Story

T/TCP


RFC 1644 - T/TCP -- TCP Exte

https://tools.ietf.org/html/rfc1644

[Docs] [txt|pdf] [Tracker]

Obsoleted by: 6247 HISTORIC

Network Working Group R. Braden
Request for Comments: 1644 ISI
Category: Experimental July 1994

T/TCP -- TCP Extensions for Transactions
Functional Specification

Status of this Memo

This memo describes an Experimental Protocol for the Internet community, and requests discussion and suggestions for improvements. It does not specify an Internet Standard. Distribution is unlimited.

Abstract

This memo specifies T/TCP, an experimental TCP extension for efficient transaction-oriented (request/response) service. This backwards-compatible extension could fill the gap between the current connection-oriented TCP and the datagram-based UDP.

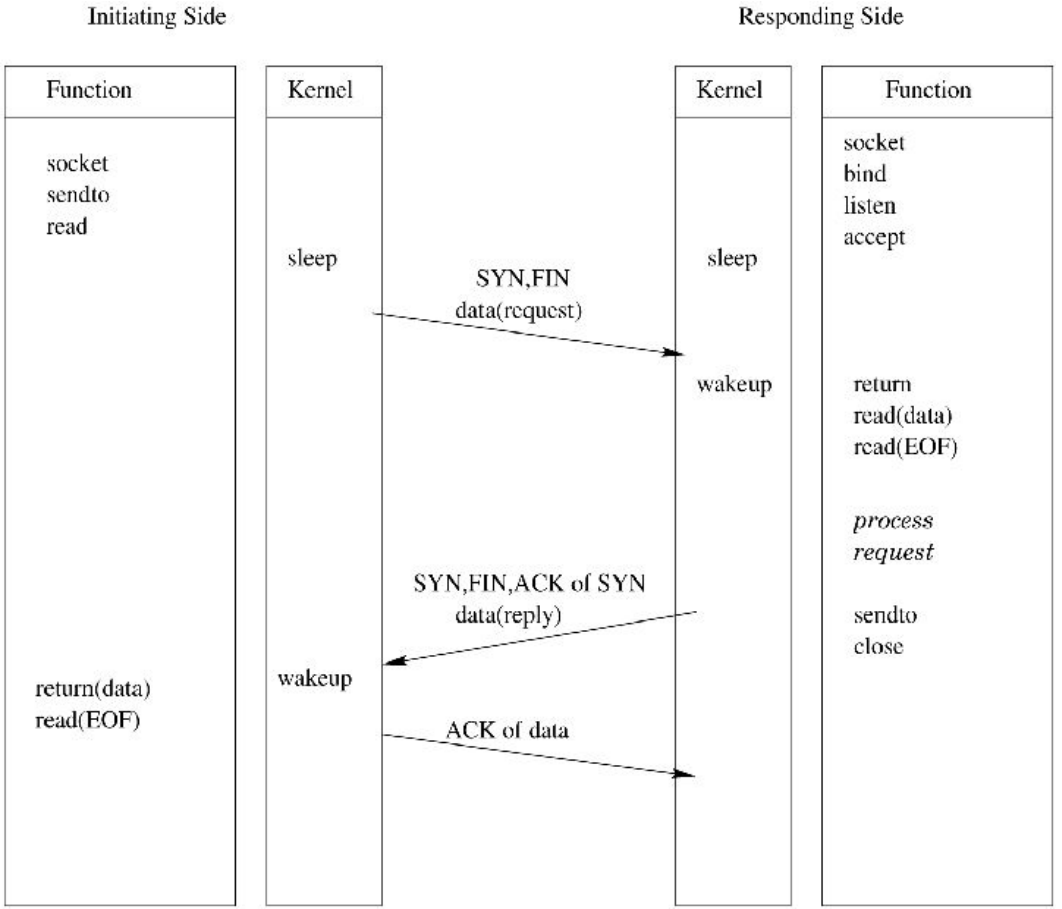
This work was supported in part by the National Science Foundation under Grant Number NCR-8922231.

Table of Contents

1. INTRODUCTION 2
2. OVERVIEW 3

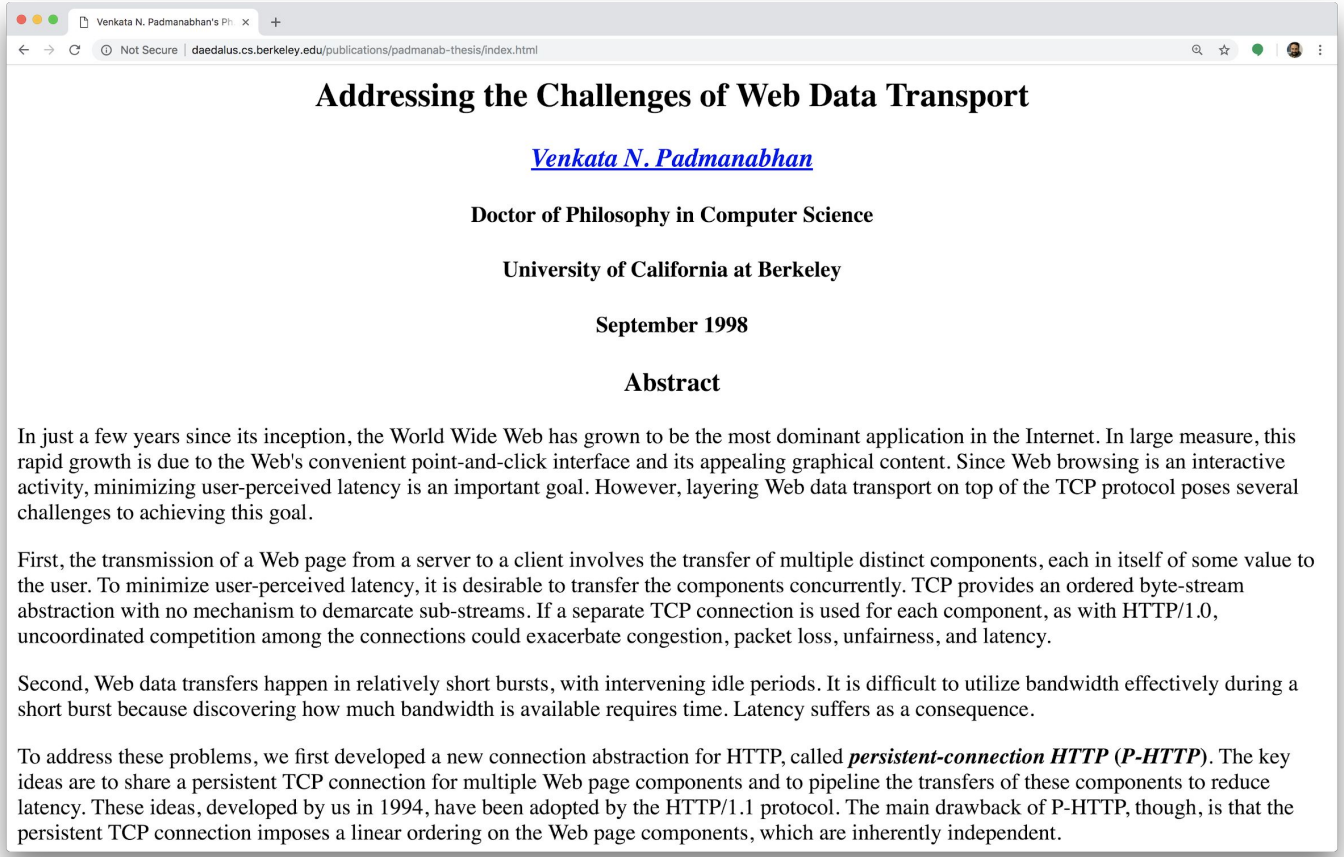
The Transport Story

T/TCP



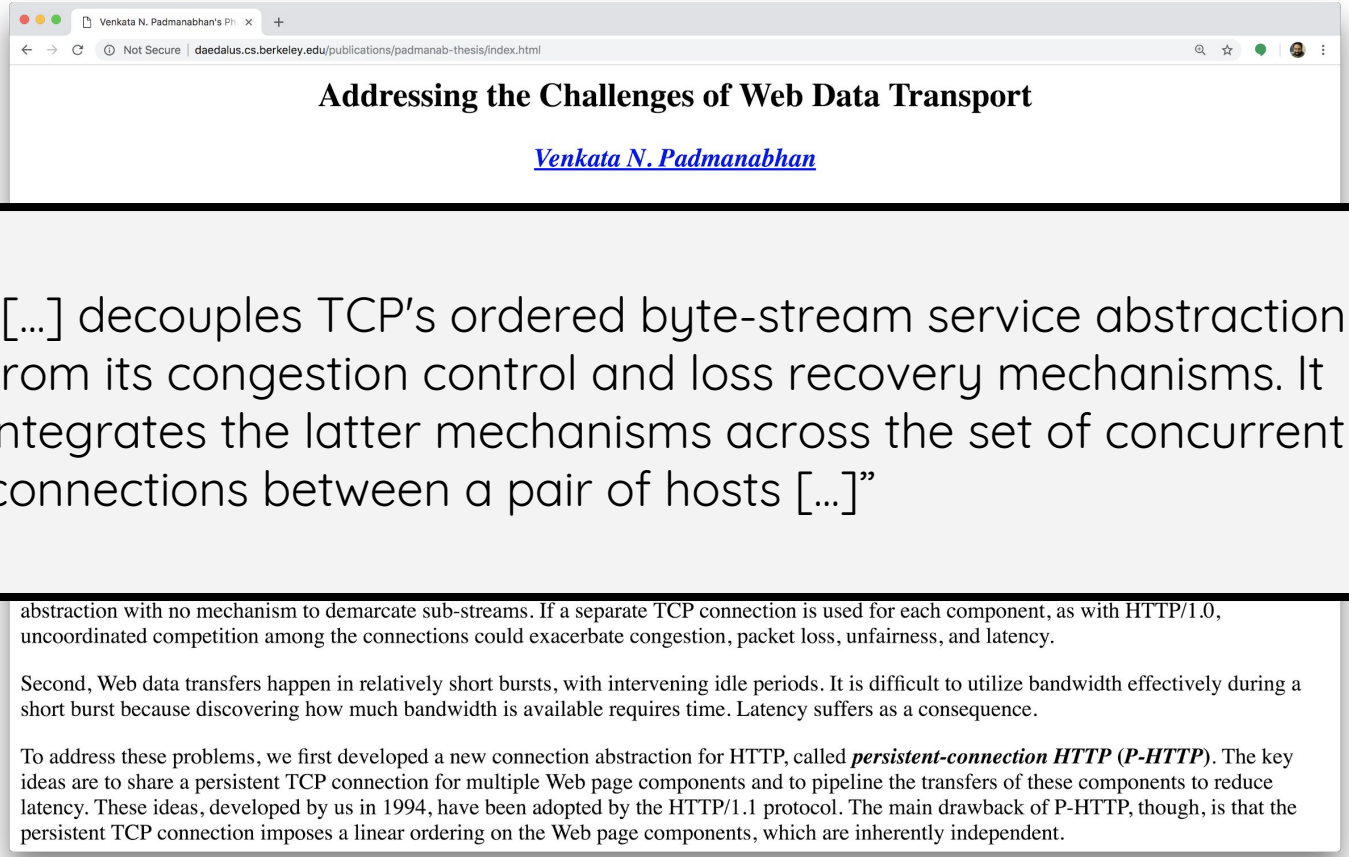
The Transport Story

T/TCP
TCP Session



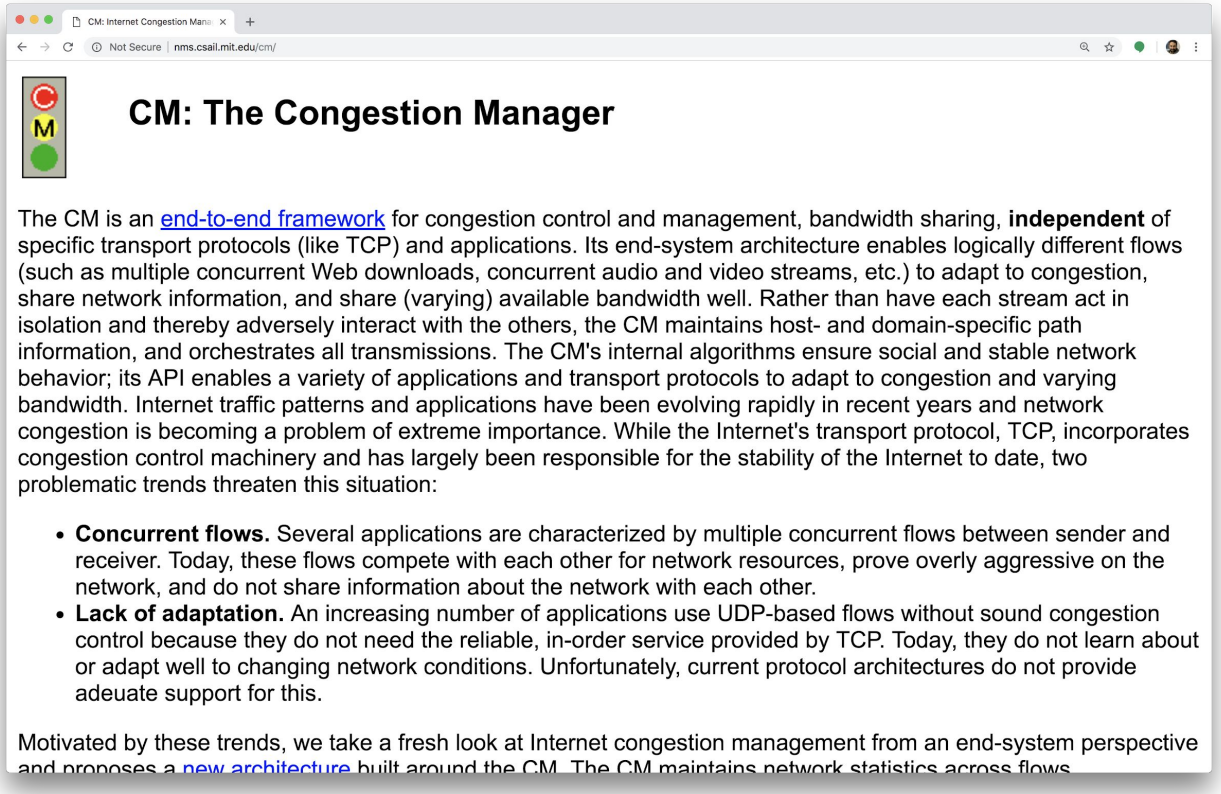
The Transport Story

T/TCP
TCP Session



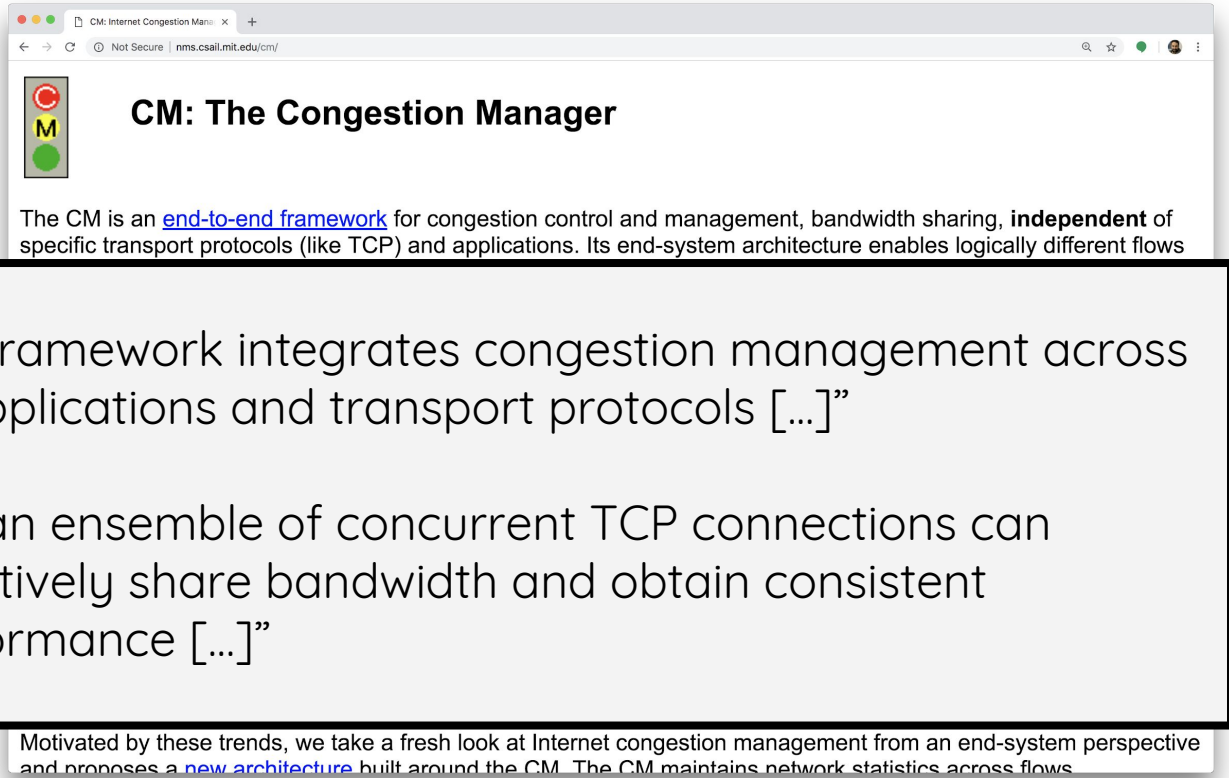
The Transport Story

T/TCP
TCP Session
Congestion Manager



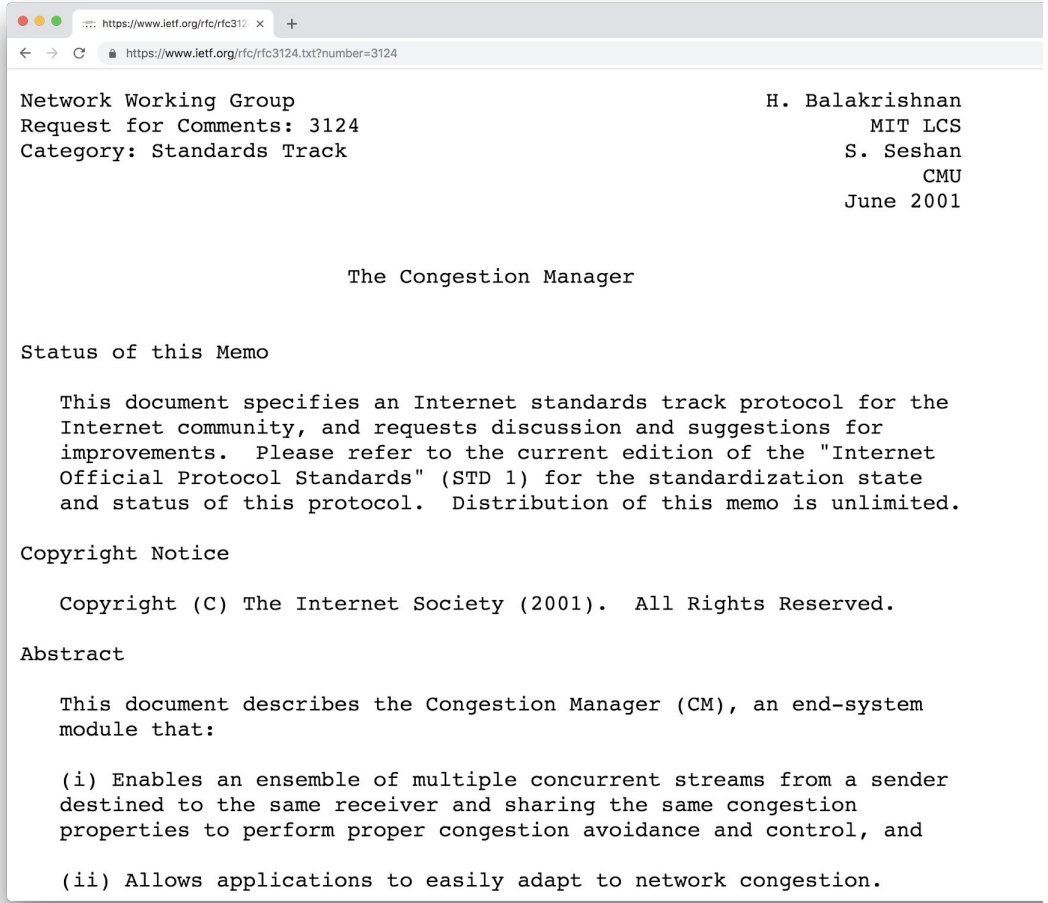
The Transport Story

T/TCP
TCP Session
Congestion Manager



The Transport Story

T/TCP
TCP Session
Congestion Manager



https://www.ietf.org/rfc/rfc3124.txt?number=3124

Network Working Group
Request for Comments: 3124
Category: Standards Track

H. Balakrishnan
MIT LCS
S. Seshan
CMU
June 2001

The Congestion Manager

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2001). All Rights Reserved.

Abstract

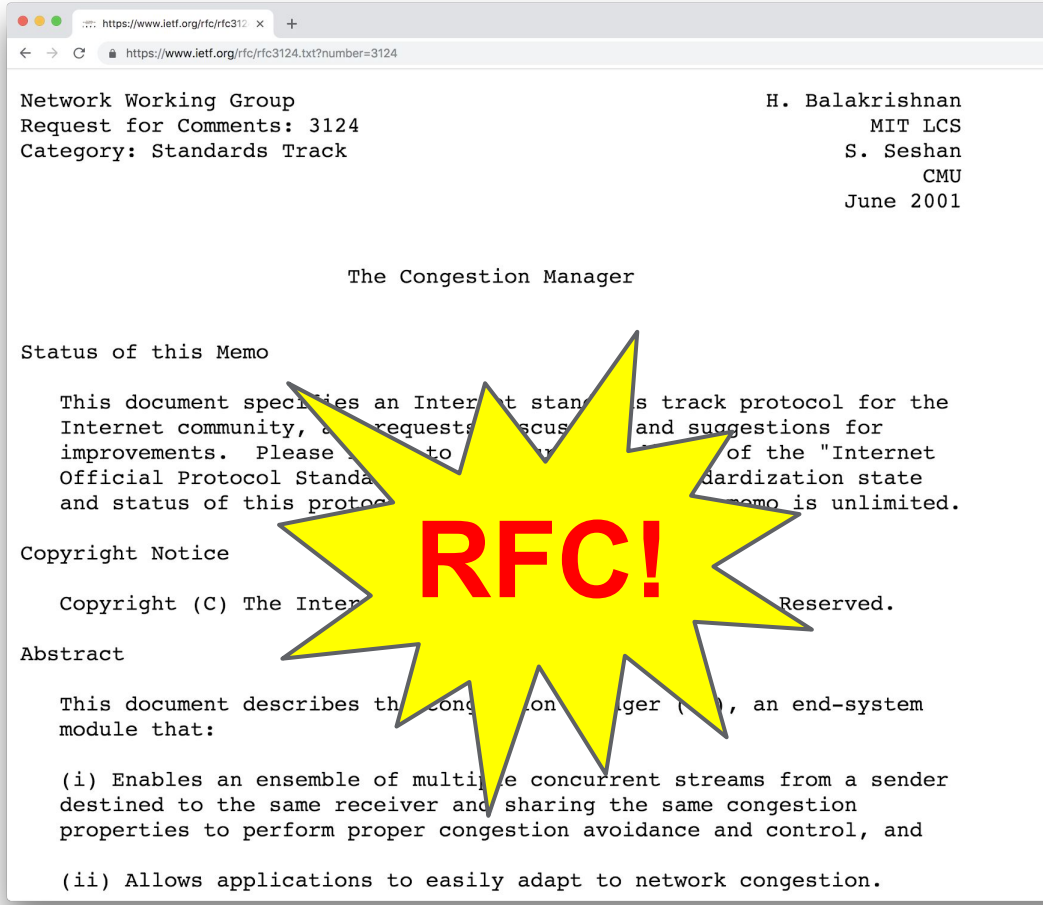
This document describes the Congestion Manager (CM), an end-system module that:

(i) Enables an ensemble of multiple concurrent streams from a sender destined to the same receiver and sharing the same congestion properties to perform proper congestion avoidance and control, and

(ii) Allows applications to easily adapt to network congestion.

The Transport Story

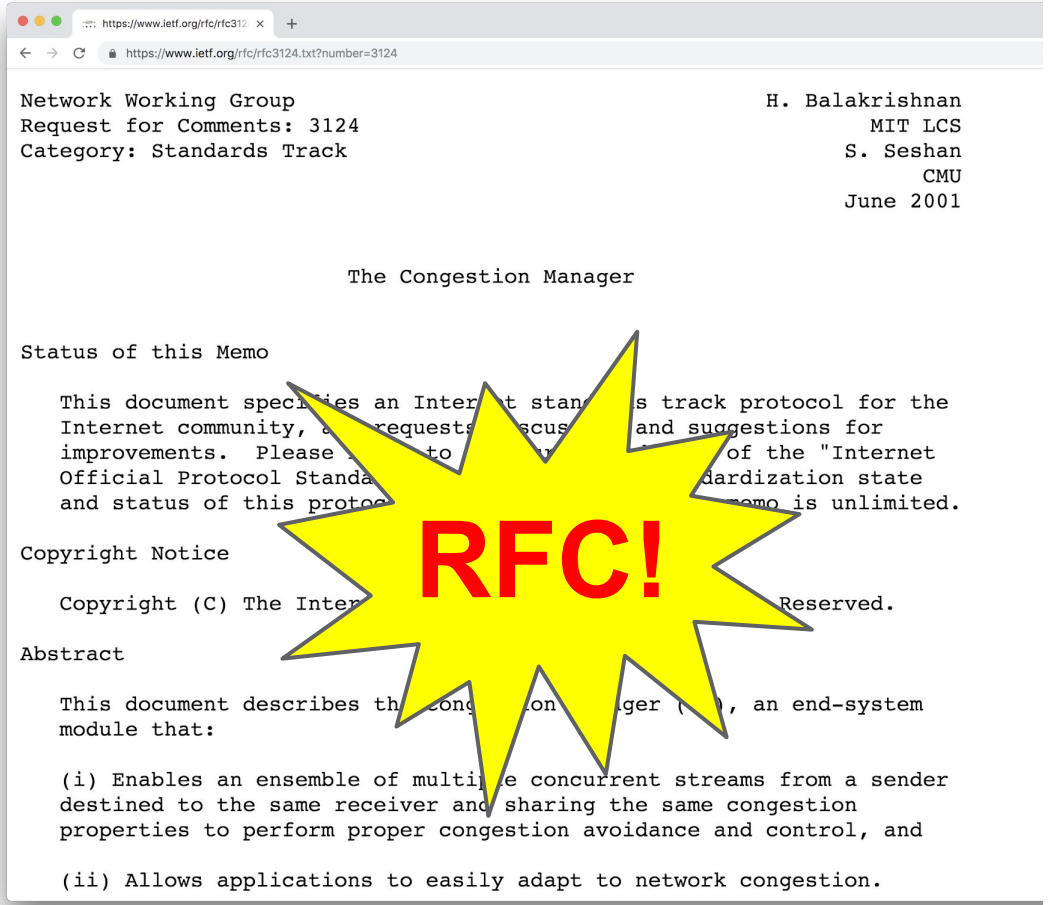
T/TCP
TCP Session
Congestion Manager



IETF Detour!

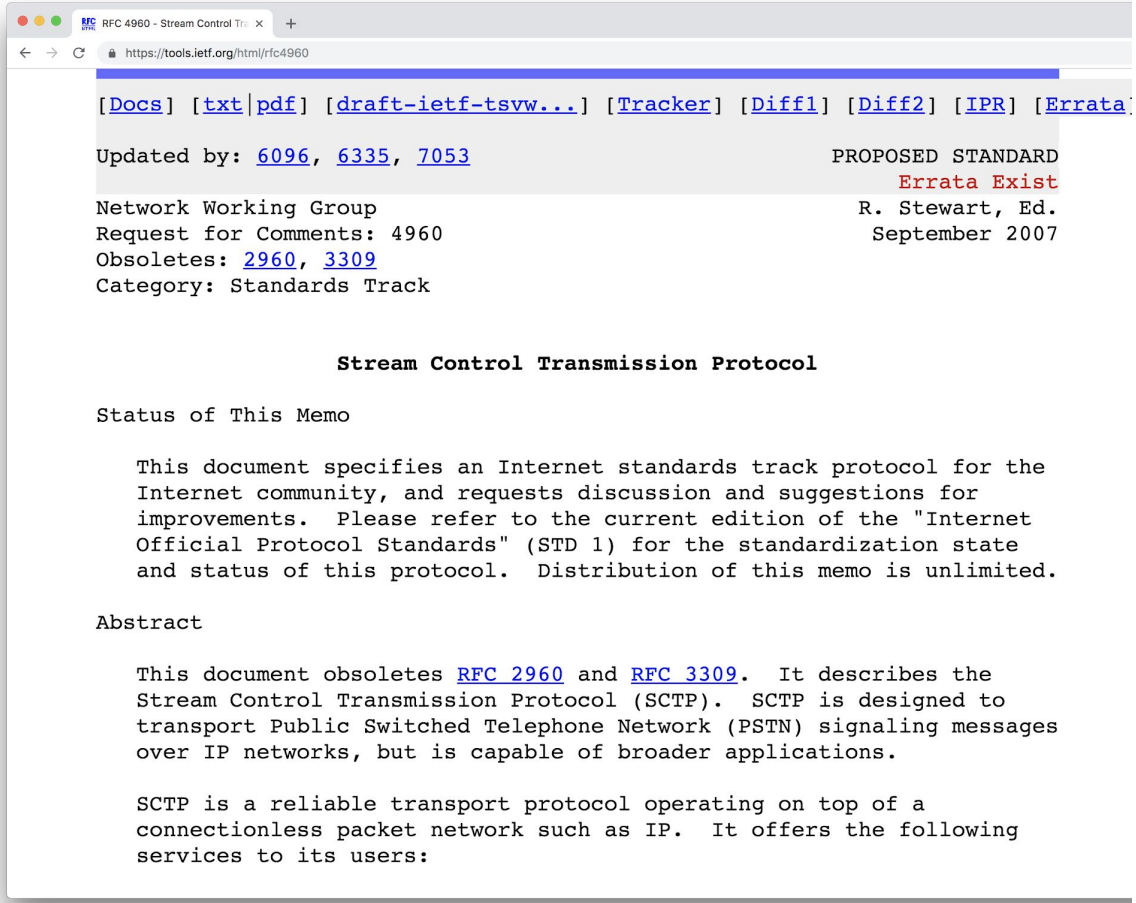
The Transport Story

T/TCP
TCP Session
Congestion Manager



The Transport Story

T/TCP
TCP Session
Congestion Manager
SCTP



The Transport Story

T/TCP
TCP Session
Congestion Manager
SCTP
…
…

The Transport Story

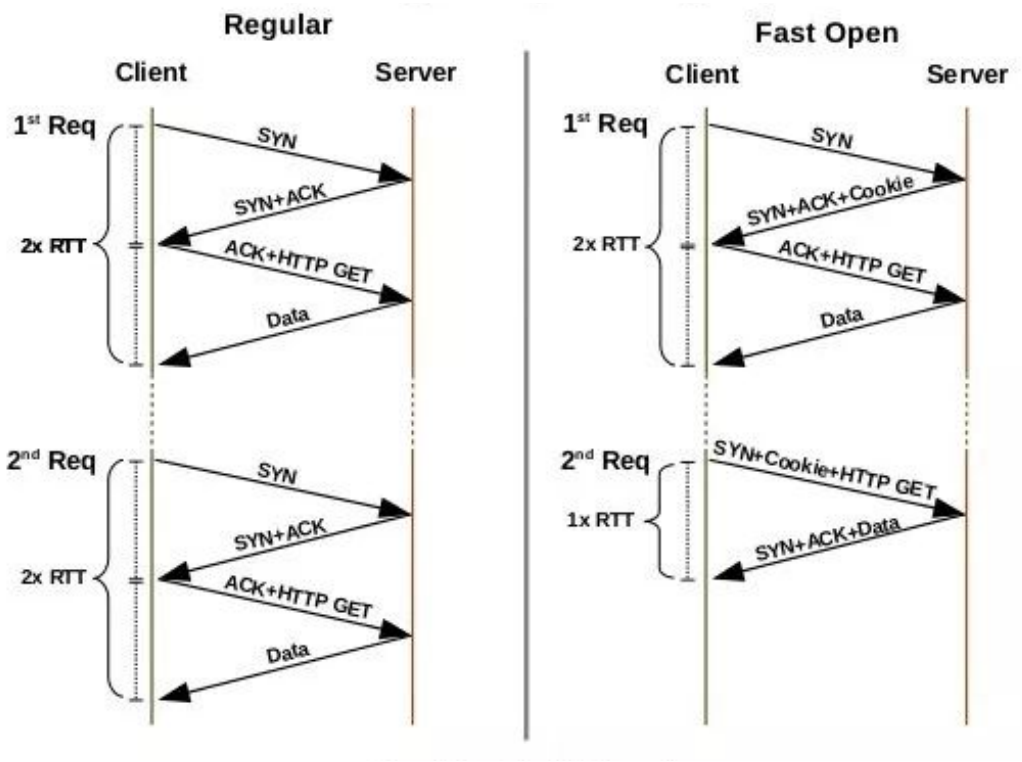
T/TCP
TCP Session
Congestion Manager
SCTP
…
…
SST (UDP-based)
Minion (TCP and TLS based)

The Transport Story

T/TCP
TCP Session
Congestion Manager
SCTP
…
…
SST (UDP-based)
Minion (TCP and TLS based)
TCP Fast Open

The Transport Story

T/TCP
TCP Session
Congestion Manager
SCTP
...
...
SST (UDP-based)
Minion (TCP and TLS based)
TCP Fast Open



Middleboxes

“[...] intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host” - RFC 3234

Home routers (NATs)
Firewalls
Application load balancers (HTTP)
Protocol accelerators (PEPs)

Middleboxes

Stages of grief

The HTTP Story (contd.)

HTTP/1.0
HTTP/1.1
HTTPng (?)

The HTTP Story

HTTP/1.0
HTTP/1.1
HTTPng (?)
...
...
...
...
...

The HTTP Story

HTTP/1.0
HTTP/1.1
HTTPng (?)

…

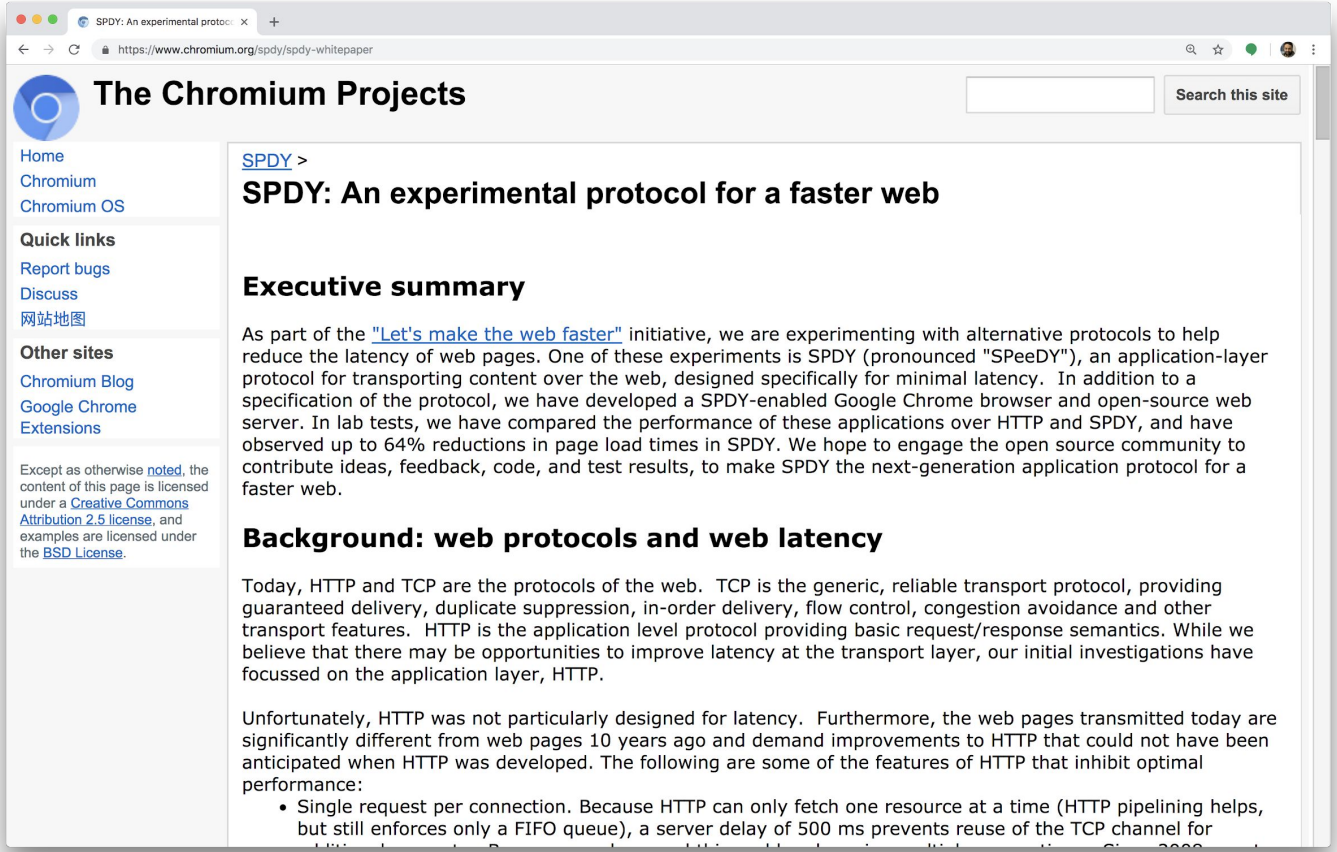
…

…

…

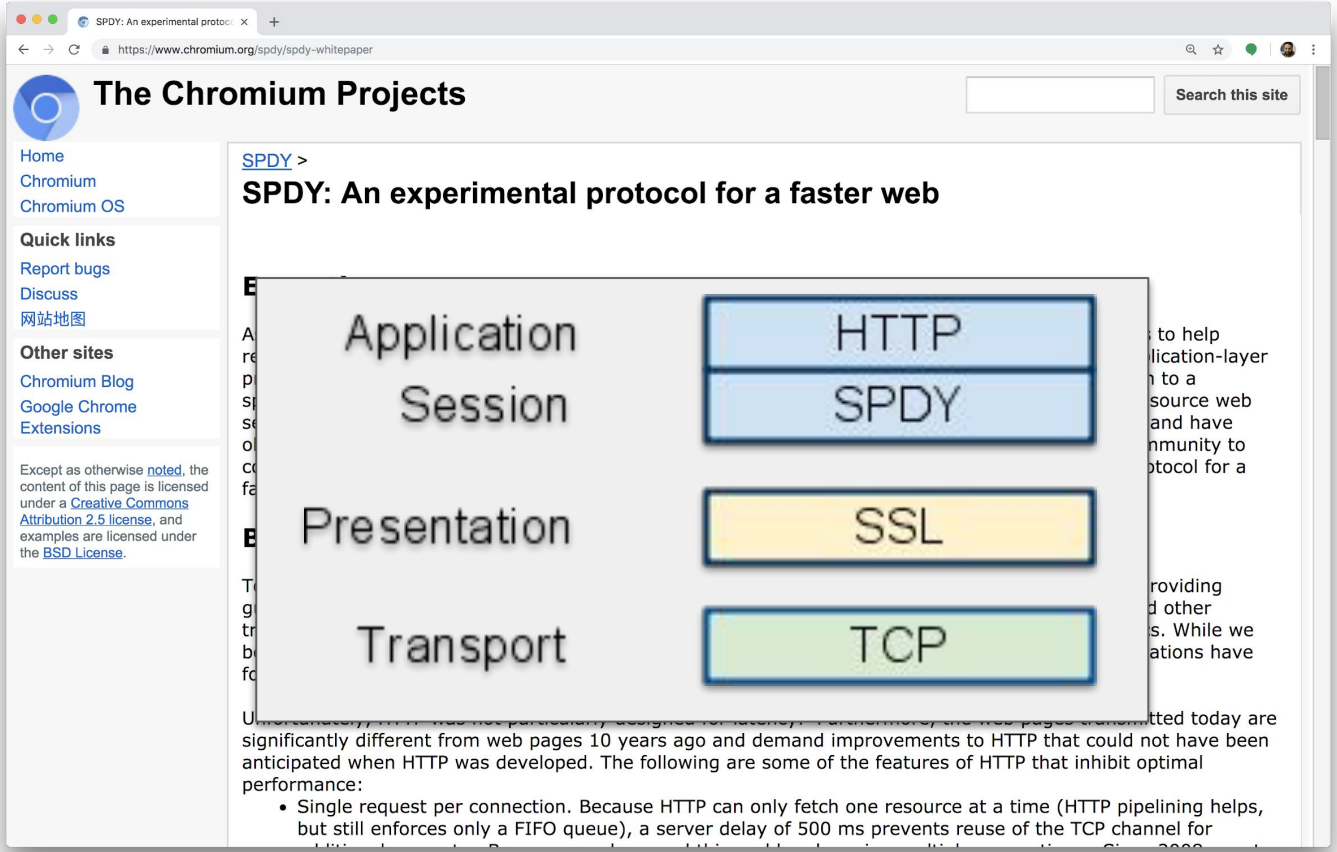
…

SPDY



The HTTP Story

HTTP/1.0
HTTP/1.1
HTTPng (?)
...
...
...
...
...
SPDY



The HTTP Story

HTTP/1.0
HTTP/1.1
HTTPng (?)

…

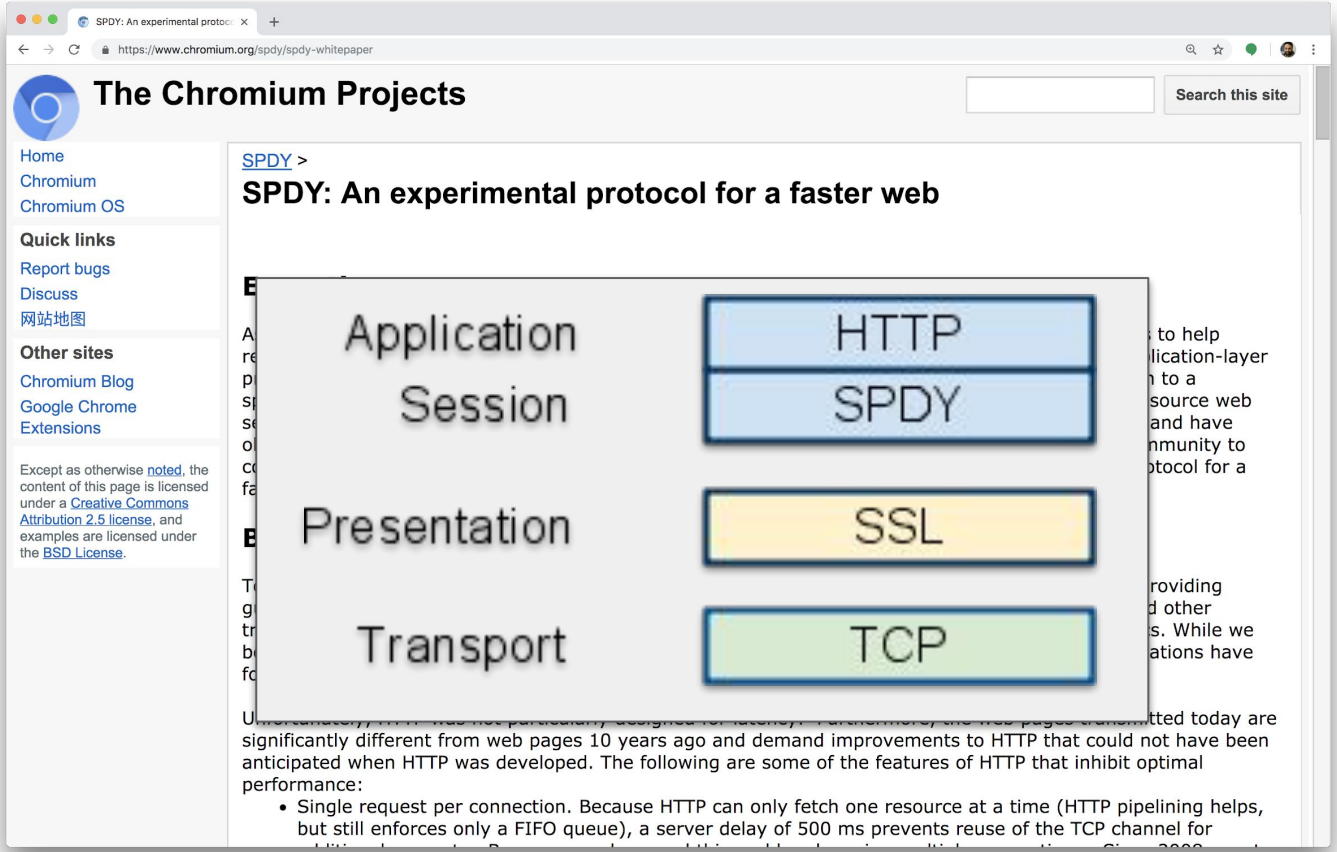
…

…

…

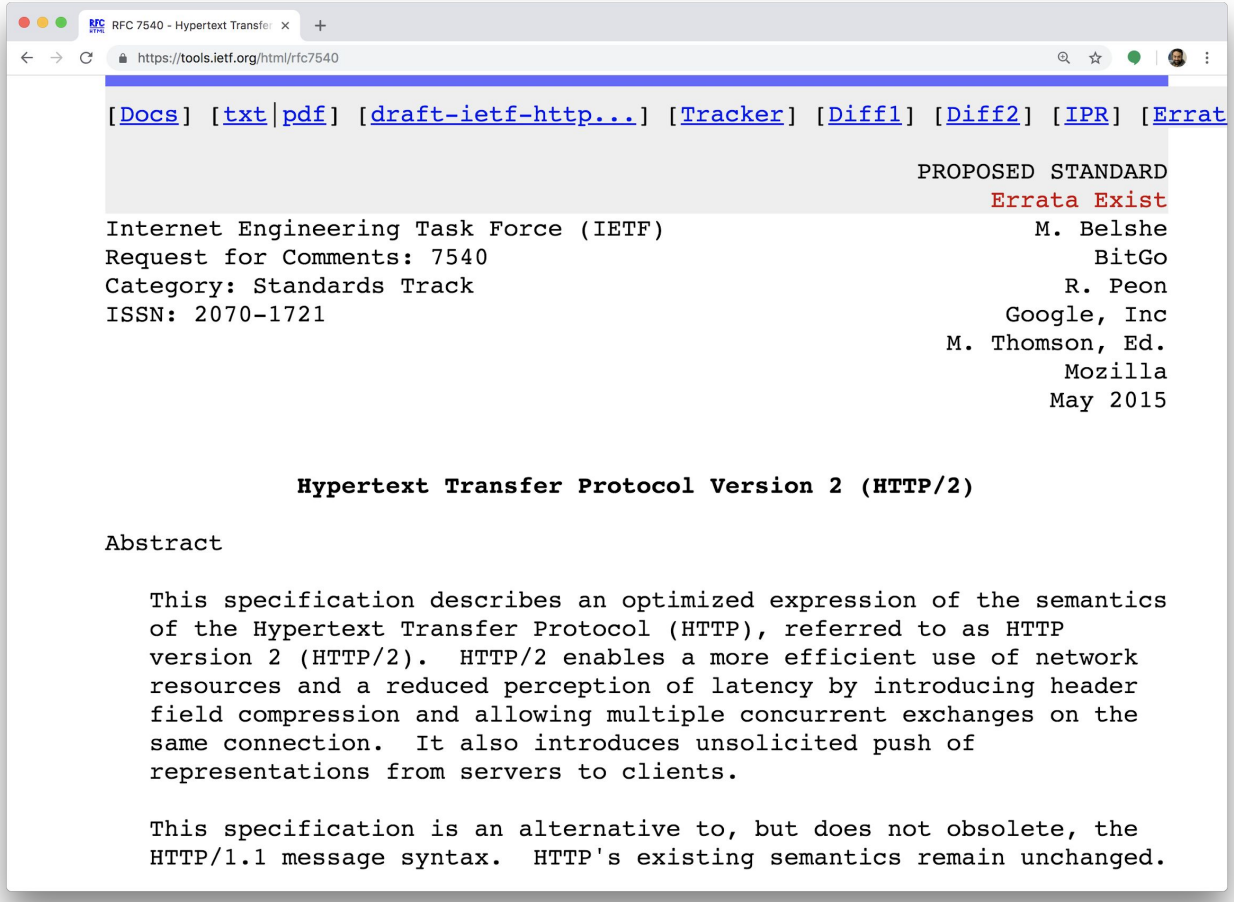
…

SPDY
- streams
- multiplexing
- flow control
- priorities

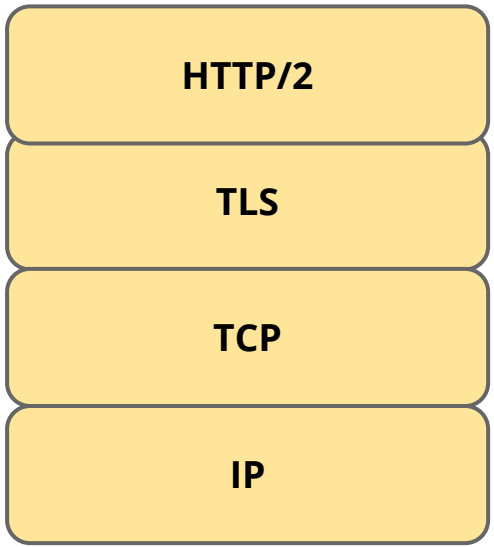


The HTTP Story

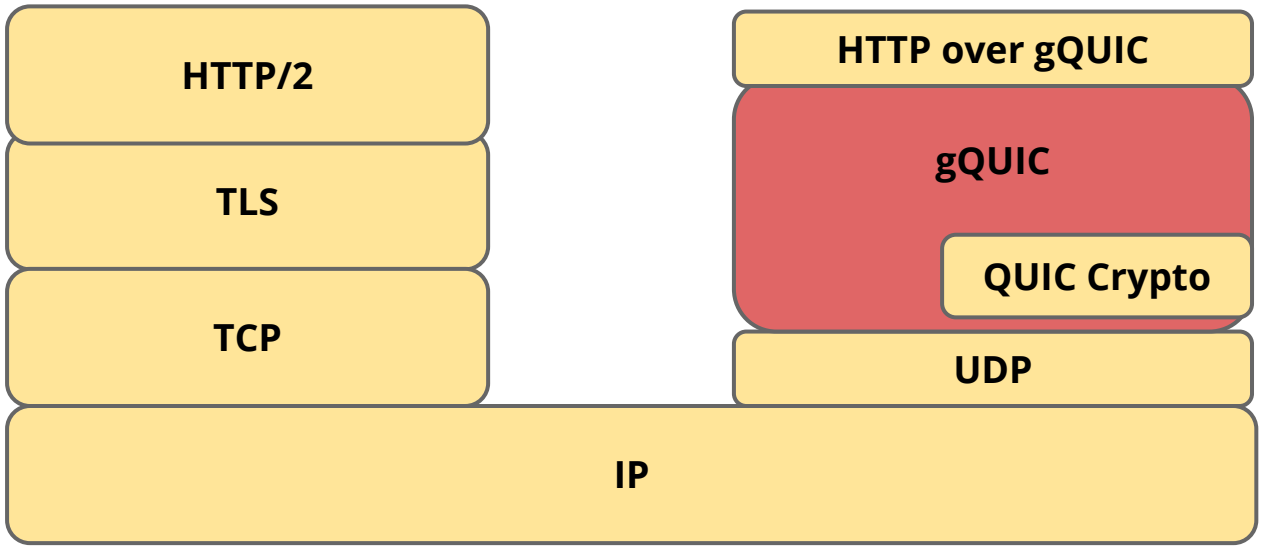
HTTP/1.0
HTTP/1.1
HTTPng (?)
...
...
...
...
...
HTTP/2
- streams
- multiplexing
- flow control
- priorities



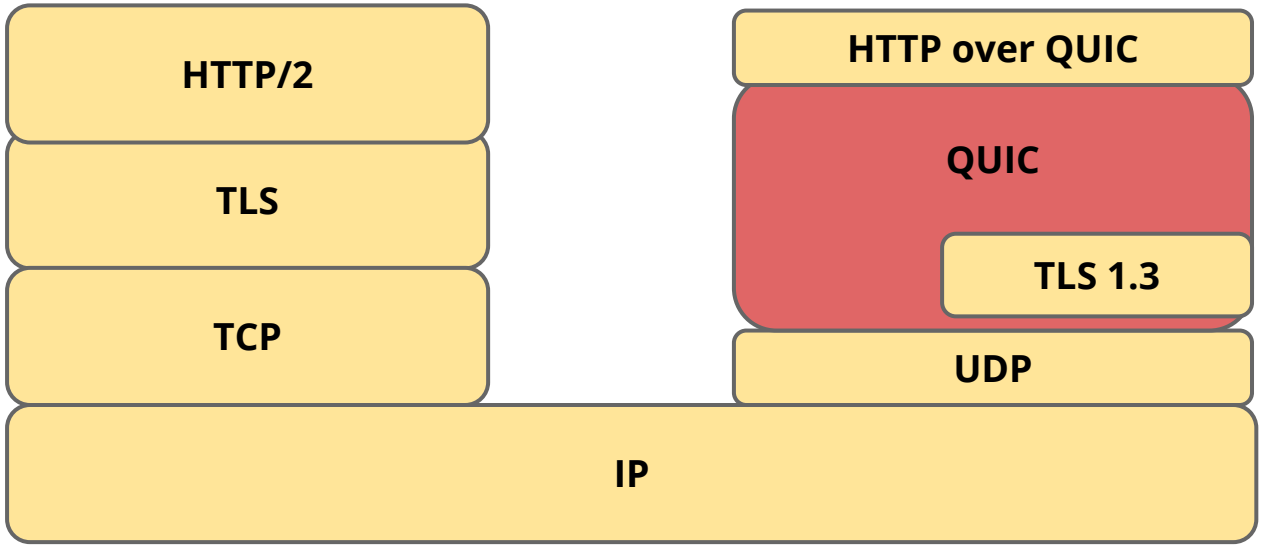
What are we talking about?



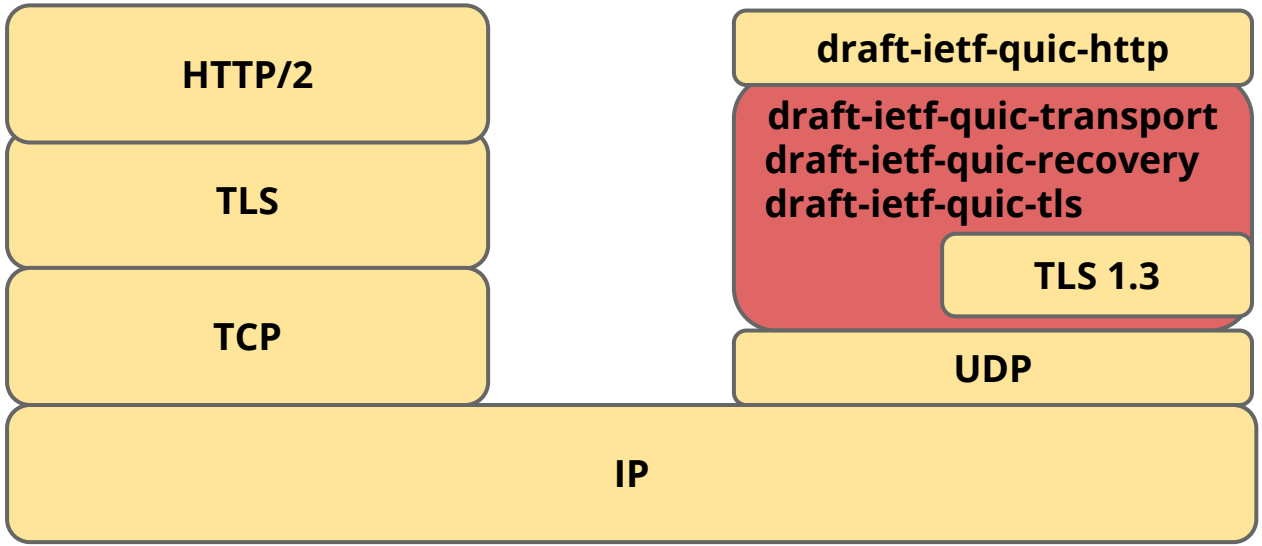
Google's QUIC Experiment



The QUIC Standard



The QUIC Standard



QUIC Design Goals (1 of 2)

Deployability and evolvability

in userspace, atop UDP

encrypted and authenticated headers

Sometime, a while ago

First byte of gQUIC packet was *flags*

Sometime, a while ago

First byte of gQUIC packet was *flags*

: unencrypted, and had been 0x07 for a while

Sometime, a while ago

First byte of gQUIC packet was *flags*

: unencrypted, and had been 0x07 for a while

We flipped a bit.

Sometime, a while ago

First byte of gQUIC packet was *flags*
: unencrypted, and had been 0x07 for a while

We flipped a bit.

“users cannot reach any Google property over Chrome!”

What had happened

Firewall

: allowed first packet in both directions

: blackholed all subsequent packets

What had happened

Firewall

: allowed first packet in both directions

: blackholed all subsequent packets

“in wireshark, noticed that first byte was always the same”

What had happened

```
if  udp_payload[0] == 7:  QUIC
```

Protocol design maxim

"the ultimate defense of the end to end mode is end to end encryption"

David Clark, J. Wroclawski, K. Sollins, and R. Braden, *Tussle in Cyberspace: Defining Tomorrow's Internet*. IEEE/ACM ToN, 2005.

QUIC Design Goals (1 of 2)

Deployability and evolvability

in userspace, atop UDP

encrypted and authenticated headers

Low-latency secure connection establishment

mostly 0-RTT, sometimes 1-RTT

(similar to TCP Fast Open + TLS 1.3)

QUIC Design Goals (1 of 2)

Deployability and evolvability

in userspace, atop UDP

encrypted and authenticated headers

Low-latency secure connection establishment

mostly 0-RTT, sometimes 1-RTT

(similar to TCP Fast Open + TLS 1.3)

Streams and multiplexing

lightweight abstraction within a connection

avoids head-of-line blocking in TCP

QUIC Design Goals (2 of 2)

Better loss recovery and flexible congestion control

unique packet number, receiver timestamp

QUIC Design Goals (2 of 2)

Better loss recovery and flexible congestion control

unique packet number, receiver timestamp

Connection migration

18-byte Connection ID

also, resilience to NAT rebinding

QUIC Performance

			% Reduction in Search Latency	
Country	Mean Min RTT (ms)	Mean TCP Rtx %	Desktop	Mobile
South Korea	38	1	1.3	1.1

QUIC Performance

			% Reduction in Search Latency	
Country	Mean Min RTT (ms)	Mean TCP Rtx %	Desktop	Mobile
South Korea	38	1	1.3	1.1
USA	50	2	3.4	2.0

QUIC Performance

			% Reduction in Search Latency	
Country	Mean Min RTT (ms)	Mean TCP Rtx %	Desktop	Mobile
South Korea	38	1	1.3	1.1
USA	50	2	3.4	2.0
India	188	8	13.2	5.5

QUIC Performance

Country	Mean Min RTT (ms)	Mean TCP Rtx %	% Reduction in Rebuffer Rate	
			Desktop	Mobile
South Korea	38	1	0.0	10.1
USA	50	2	4.1	12.9
India	188	8	22.1	20.2

Current Status

Work at IETF for past 2 years

Strong focus on security and privacy

- Network operator woes

Strong focus on avoiding ossification

- Encryption
- GREASEing

Several implementation efforts

- Google, Facebook, Fastly, Firefox, Microsoft, Apple, F5