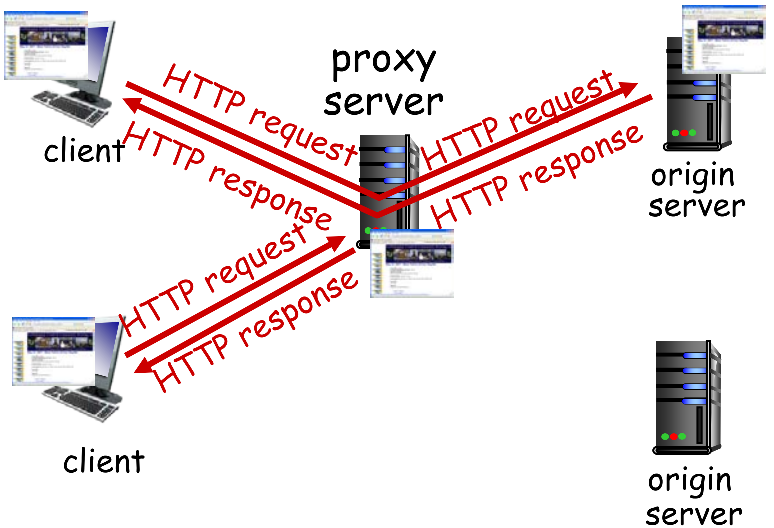
Web caches (proxy servers)

Goal: satisfy client request without involving origin server

- user configures browser to point to a *Web cache*
- browser sends all HTTP requests to cache
  - *if* object in cache: cache returns object to client
  - *else* cache requests object from origin server, caches received object, then returns object to client



Web caches (proxy servers)

- Web cache acts as both client and server
  - server for original requesting client
  - client to origin server
- typically cache is installed by ISP (university, company, residential ISP)

Why Web caching?

- reduce response time for client request
  - cache is closer to client
- reduce traffic on an institution's access link
- Internet is dense with caches
  - enables "poor" content providers to more effectively deliver content

Caching example

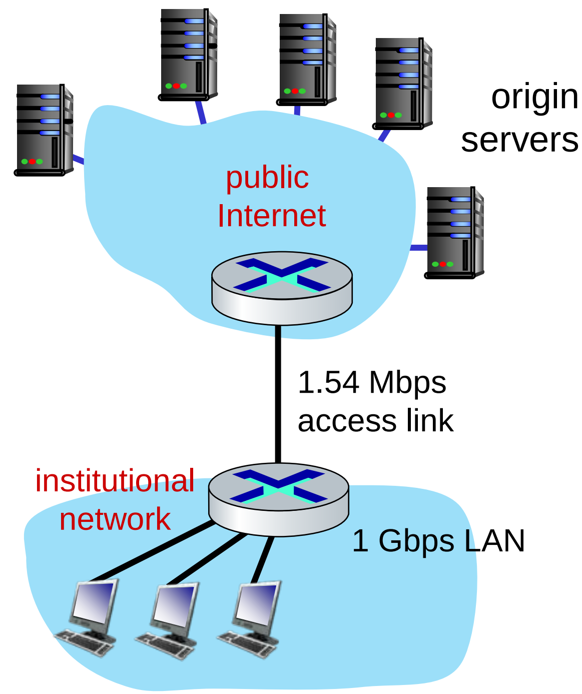
Scenario:

- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- Web object size: 100K bits
- Average request rate from browsers to origin servers: 15/sec
  - average data rate to browsers: 1.50 Mbps

Performance:

- LAN utilization: .0015
- access link utilization = .97
- end-end delay = Internet delay + access link delay + LAN delay
  = 2 sec + minutes + usecs

problem: large delays at high utilization!

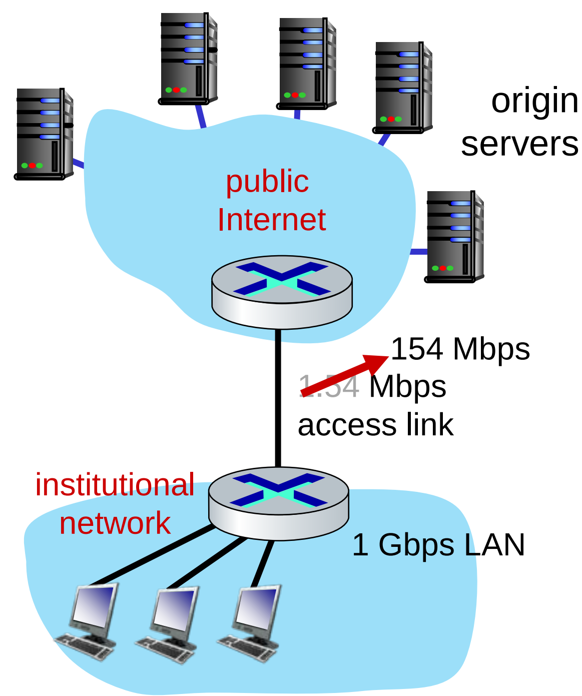


Caching example: buy a faster access link

Scenario:

154 Mbps

- access link rate: ~~1.54~~ Mbps
- RTT from institutional router to server: 2 sec
- Web object size: 100K bits
- Average request rate from browsers to origin servers: 15/sec
  - average data rate to browsers: 1.50 Mbps



Caching example: buy a faster access link

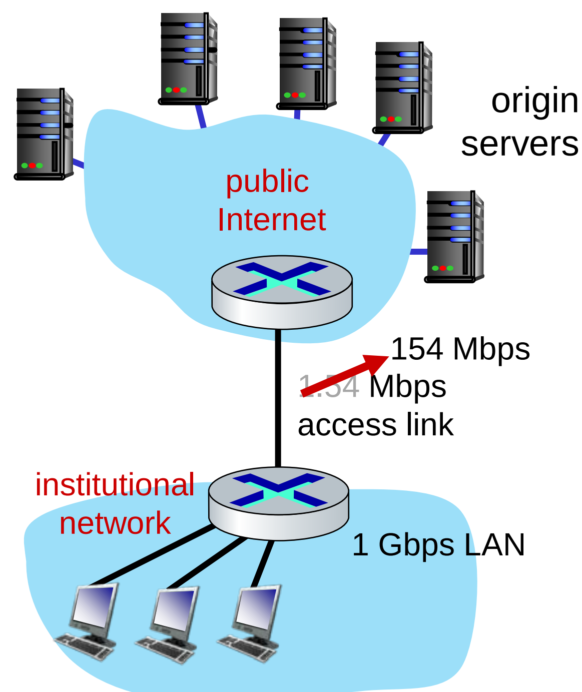
Scenario: 154 Mbps

- access link rate: ~~1.54~~ Mbps
- RTT from institutional router to server: 2 sec
- Web object size: 100K bits
- Average request rate from browsers to origin servers: 15/sec
  - average data rate to browsers: 1.50 Mbps

Performance:

- LAN utilization: .0015
- access link utilization = ~~.97~~ → .0097
- end-end delay = Internet delay + access link delay + LAN delay

  = 2 sec + ~~minutes~~ + usecs → msecs



Cost: faster access link (expensive!)

Caching example: install a web cache

Scenario:

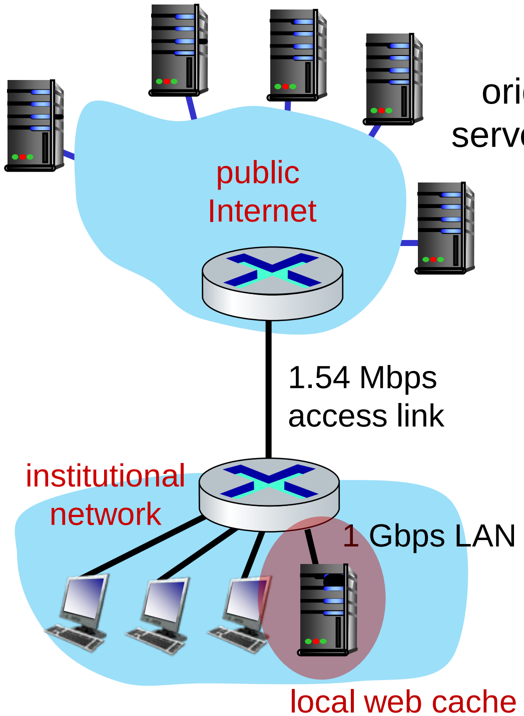
- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- Web object size: 100K bits
- Average request rate from browsers to origin servers: 15/sec
  - average data rate to browsers: 1.50 Mbps

Performance:

- LAN utilization: .?
- access link utilization = ?
- average end-end delay = ?

How to compute link utilization, delay?

Cost: web cache (cheap!)



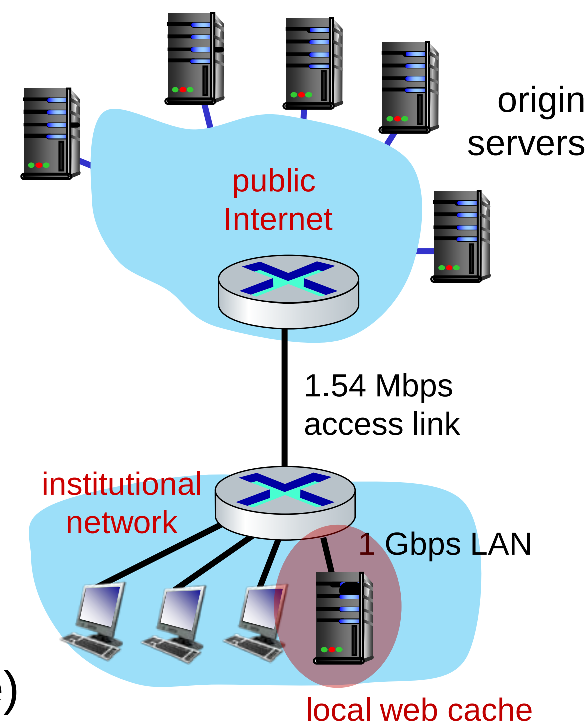
	Miss	Hit
`mul = 1`	1 1	
`j = 0`	1 1	
`loop: read j`		1 1 512
`if (j >= 512) exit` `else`		
`read g[j]`	1 16	1 496
`read mul`		1 1 512
`compute mul *g[j]`		
`write mul`		1 1 512
`read j`		1 1 512
`compute j+1`		
`write j`		1 1 512
`jump to loop`		

$$\frac{\text{\# of cache hits}}{(\text{\# of cache hits} + \text{\# of cache misses})} = \text{Hit ratio}$$

OR

Hit ratio = 1 - Miss ratio

Caching example: install a web cache



Calculating access link utilization, end-end delay with cache:

- suppose cache hit rate is 0.4: 40% requests satisfied at cache, 60% requests satisfied at origin
- access link: 60% of requests use access link
- data rate to browsers over access link
  = 0.6 * 1.50 Mbps = .9 Mbps
- utilization = 0.9/1.54 = .58
- average end-end delay
  = 0.6 * (delay from origin servers)
  + 0.4 * (delay when satisfied at cache)
  = 0.6 (2.01) + 0.4 (~msecs) = ~ 1.2 secs

lower average end-end delay than with 154 Mbps link (and cheaper too!)

HTTP/2

Key goal: decreased delay in multi-object HTTP requests

HTTP1.1: introduced multiple, pipelined GETs over single TCP connection

- server responds *in-order* (FCFS: first-come-first-served scheduling) to GET requests
- with FCFS, small object may have to wait for transmission (head-of-line (HOL) blocking) behind large object(s)
- loss recovery (retransmitting lost TCP segments) stalls object transmission

HTTP/2

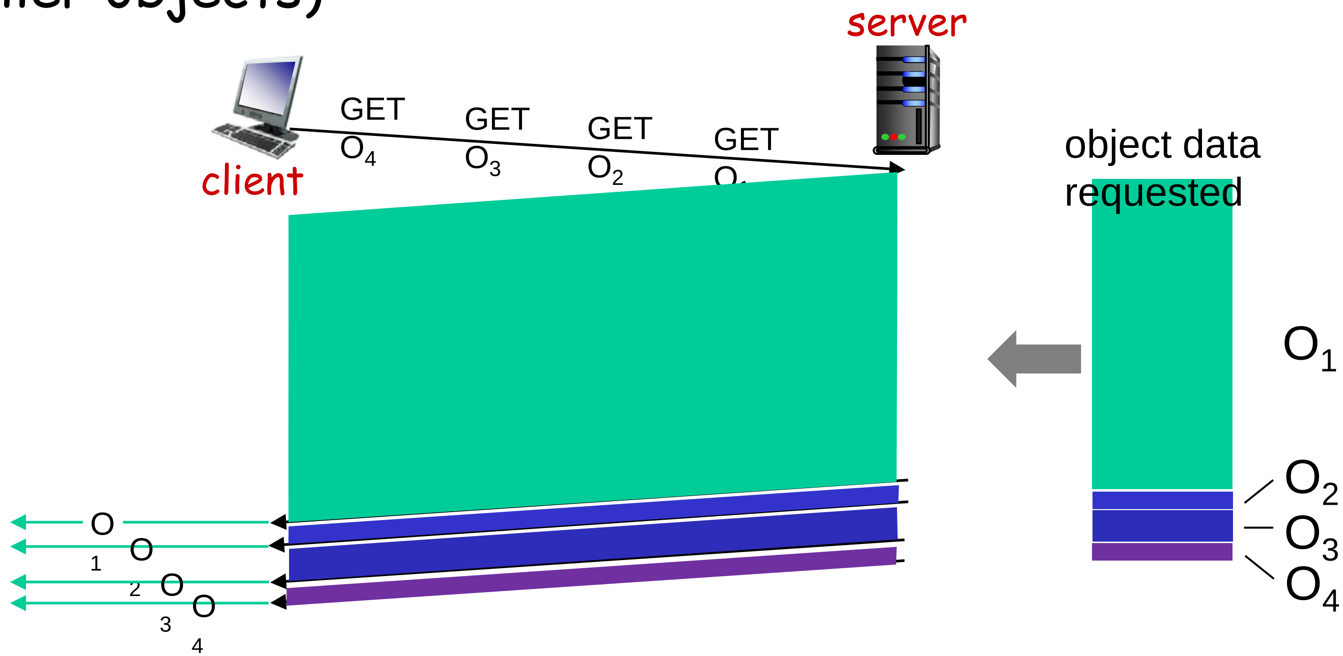
Key goal: decreased delay in multi-object HTTP requests

HTTP/2: [RFC 7540, 2015] increased flexibility at *server* in sending objects to client:

- methods, status codes, most header fields unchanged from HTTP 1.1
- transmission order of requested objects based on client-specified object priority (not necessarily FCFS)
- *push* unrequested objects to client
- divide objects into frames, schedule frames to mitigate HOL blocking

HTTP/2: mitigating HOL blocking

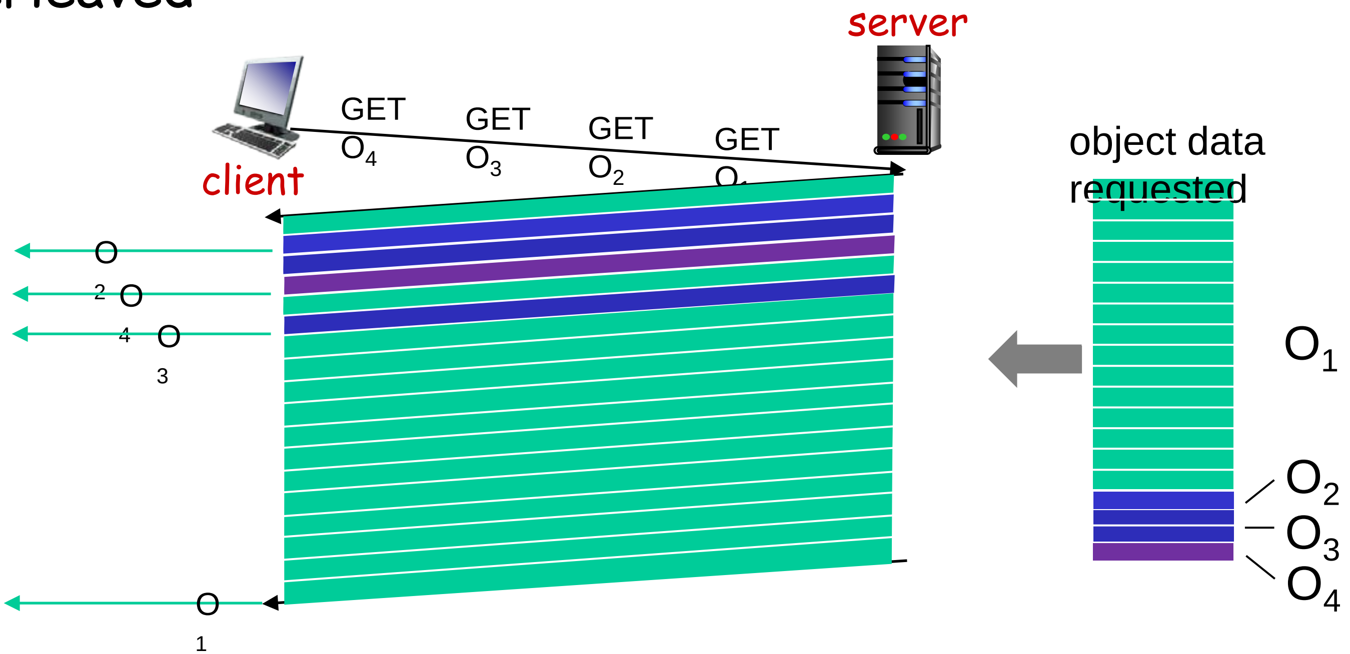
HTTP 1.1: client requests 1 large object (e.g., video file, and 3 smaller objects)



objects delivered in order requested: O_2, O_3, O_4 wait behind O_1

HTTP/2: mitigating HOL blocking

HTTP/2: objects divided into frames, frame transmission interleaved



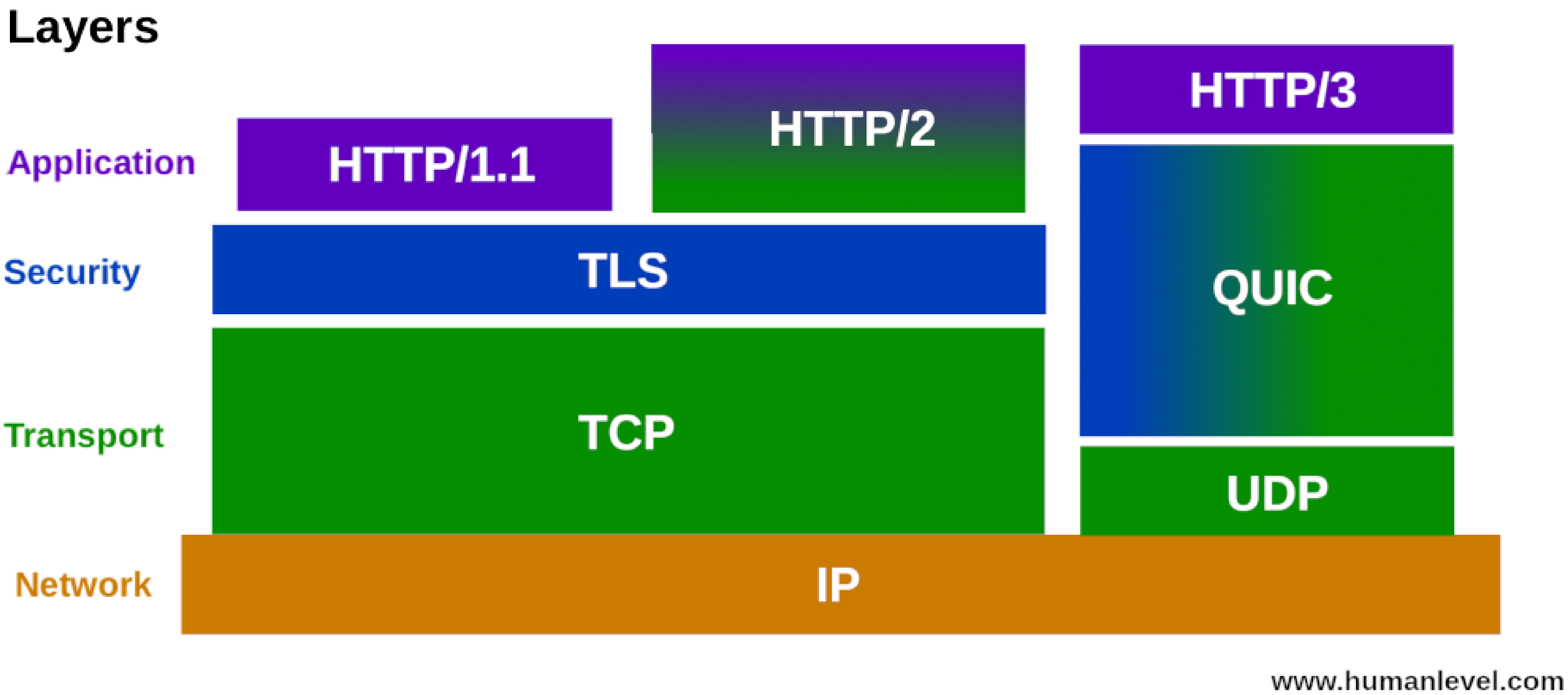
O_2, O_3, O_4 delivered quickly, O_1 slightly delayed

HTTP/2 to HTTP/3

Key goal: decreased delay in multi-object HTTP requests

HTTP/2 over single TCP connection means:

- recovery from packet loss still stalls all object transmissions
  - as in HTTP 1.1, browsers have incentive to open multiple parallel TCP connections to reduce stalling, increase overall throughput
- no security over vanilla TCP connection
- HTTP/3: adds security , per object error- and congestion-control (more pipelining) over UDP
  - more on HTTP/3 in transport layer

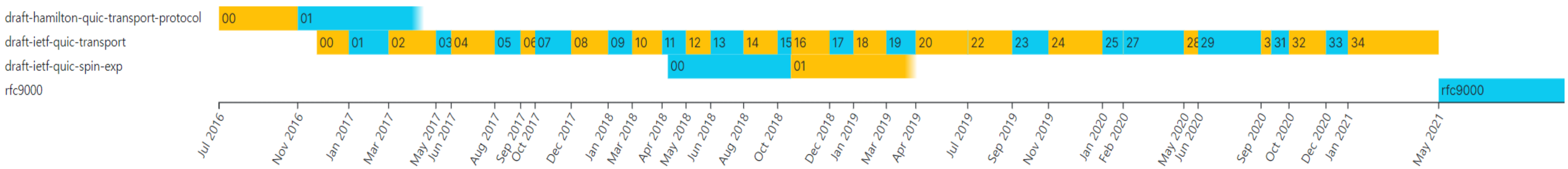

Layers
Application
Security
Transport
Network
HTTP/1.1
HTTP/2
HTTP/3
TLS
QUIC
TCP
UDP
IP
www.humanlevel.com

Datatracker Groups Documents Meetings Other User Report a bug Sign in Document search

QUIC: A UDP-Based Multiplexed and Secure Transport

RFC 9000

Status | Email expansions | History



Document	**Type**	RFC - Proposed Standard (May 2021) Errata Was draft-ietf-quic-transport (quic WG)
	Authors	Jana Iyengar, Martin Thomson
	Last updated	2022-02-19
	RFC stream	Internet Engineering Task Force (IETF)
	Formats	txt html xml pdf htmlized w/errata bibtex
	Additional resources	Mailing list discussion
IESG	**Responsible AD**	Magnus Westerlund
	Send notices to	(None)

Email authors | Email WG | IPR | ← References | → Referenced by | Search Lists

RFC 9000

```
Internet Engineering Task Force (IETF)                  J. Iyengar, Ed.
Request for Comments: 9000                                       Fastly
Category: Standards Track                               M. Thomson, Ed.
ISSN: 2070-1721                                                 Mozilla
                                                               May 2021

           QUIC: A UDP-Based Multiplexed and Secure Transport

Abstract
```

Application Layer: Overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- P2P applications
- video streaming and content distribution networks
- socket programming with UDP and TCP