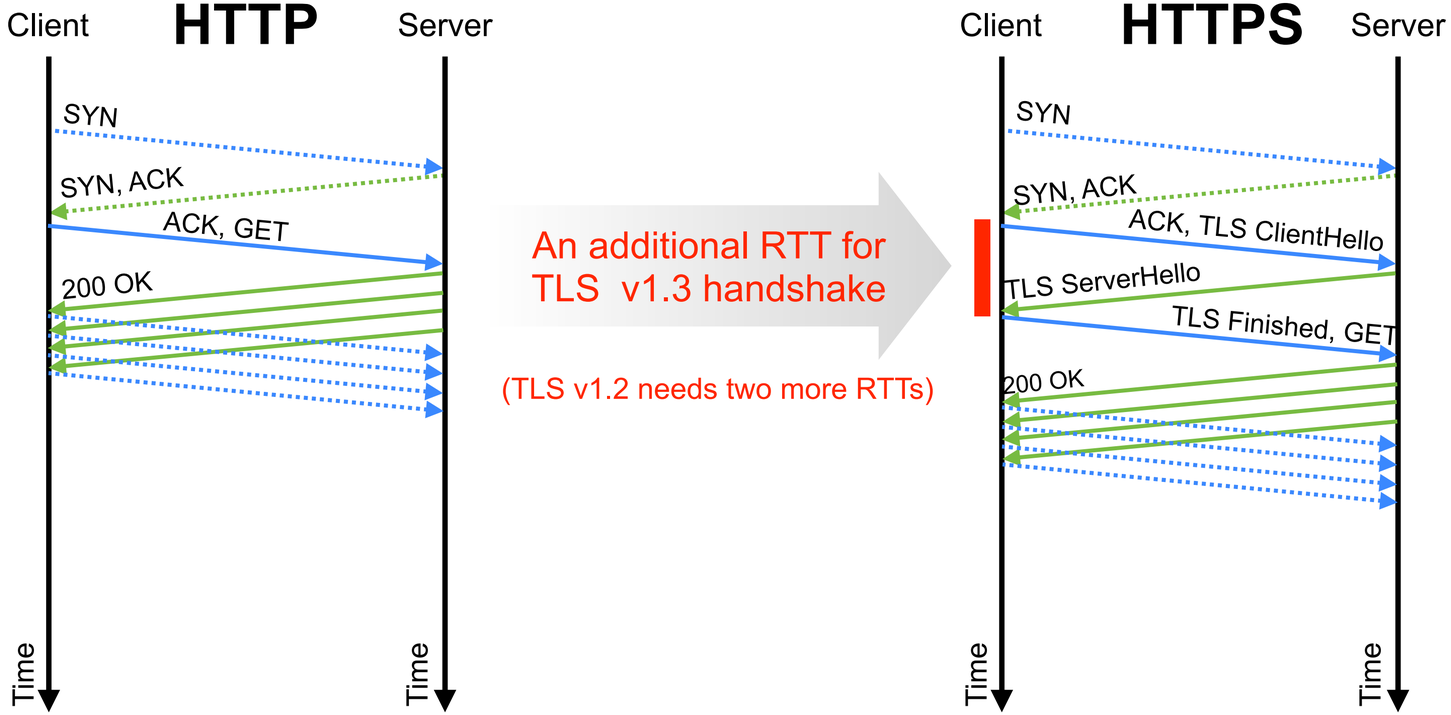
Impact of TLS and IPv6 on Connection Establishment

- Transport Layer Security (TLS) needs extra RTTs to negotiate security
- Dual-stack IPv4 and IPv6 hosts must race connections for good performance

Impact of Transport Layer Security (1/2)

- The protocol running over TCP can also impact performance
  - HTTP sends and retrieves data immediately the TCP connection is open
  - HTTPS opens a TCP connection, negotiates security parameters using TLS within that TCP connection, then starts to send and receive encrypted data – what impact does this have?



Impact of Transport Layer Security (2/2)

- Revisit simple web page download example:
  - 1x HTML file, 1x CSS file, and 1x image
  - HTML and CSS files hosted on one server, image on a different server
  - Everything retrieved via **HTTPS** over TCP/IP
- TLS v1.3 introduces two RTTs extra latency
  - One extra RTT per connection to negotiate security parameters
  - Further causes impact of RTT and connection establishment to dominate performance

RTT \ BW	1 Mbps	15 Mbps	25 Mbps	50 Mbps	100 Mbps	1 Gbps
1ms	43.9	2.9	1.7	0.9	0.4	0.04
50ms	44.2	3.2	2.1	1.2	0.7	0.34
100ms	44.5	3.5	2.4	1.5	1.0	0.64
150ms	44.8	3.8	2.7	1.8	1.3	0.94
300ms	45.7	4.7	3.6	2.7	2.2	1.84

→ -90%

→ - 36%

→ -16%

Download times, in seconds, for the simple web page described earlier, for a range of RTTs and bandwidths

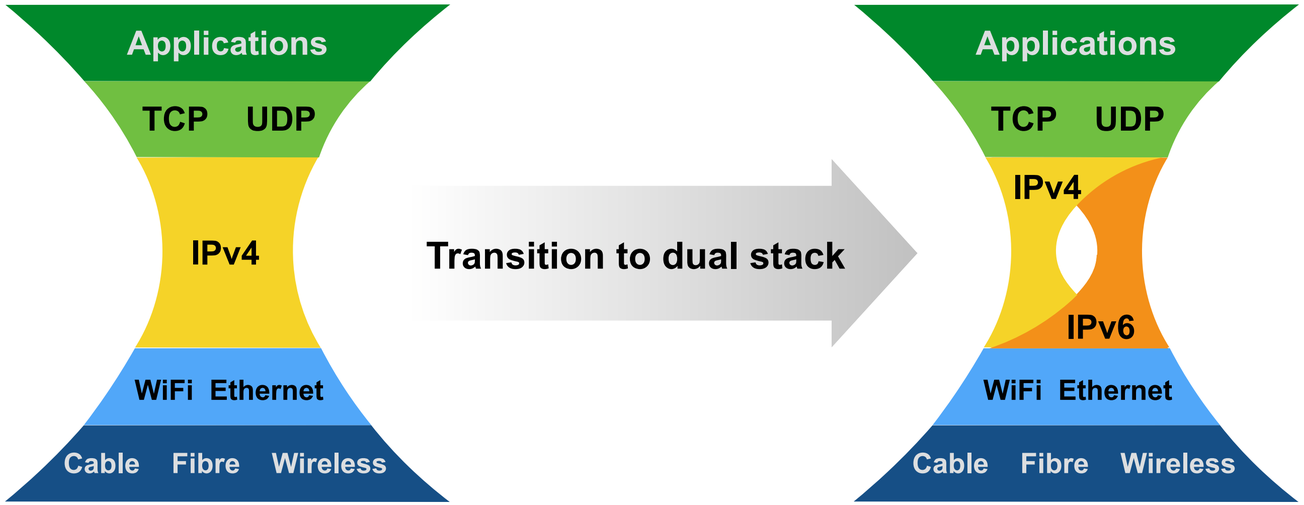
HTML = 500 kbytes, CSS = 100 kbytes, IMG = 5 Mbytes
Assuming no impact due to congestion control
With additional RTT to model TLS 1.3 handshake

Impact of Latency on TCP Performance

- For client-server applications, each request takes 1x RTT plus data serialisation time
  - Unless data is very large, RTT is often most significant performance factor
  - Each TCP connection has 1x RTT connection setup overhead
    - If TLS v1.3 is used inside TCP, an additional 1x RTT
    - If TLS v1.2 is used inside TCP, an additional 2x RTT

- To improve performance in your application:
  - Reduce the number of TCP connections used
  - Limit the number of request-response exchanges between client and server

Impact of IPv6 and Dual Stack Deployments



- IPv4 → IPv6 transition means we have **two** Internets
  - Some hosts only connect via IPv4, some only via IPv6, some have both types of address
  - Some links carry only IPv4 traffic, some only IPv6 traffic, some both types of traffic
  - Some firewalls block IPv4, some block IPv6, some block both types of traffic
- IPv6 network is **not** a subset of the IPv4 network; it's separate, but overlaps in places

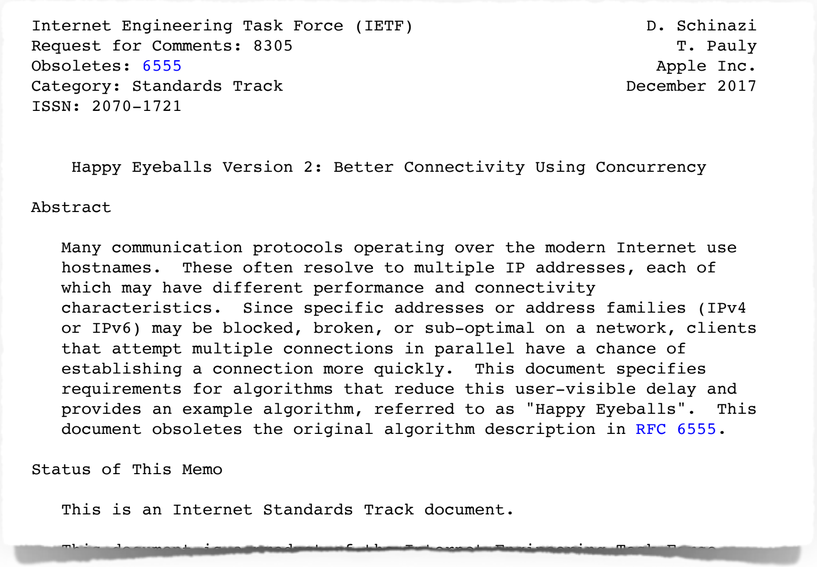
Happy Eyeballs

- How to connect to a host with more than one IP address?
  - Perform DNS lookups for IPv4 and IPv6 in parallel; start with whichever completes first
  - Call **`connect()`** for that address; if not connected within ~100ms, start **`connect()`** to next address on the list in parallel, alternating between IPv4 and IPv6
  - Use first **`connect()`** that succeeds, drop other successful connections
  - Balances speed to connect vs. network overload by trying all at once in parallel

```
[stlinux02] > ./dnslookup netflix.com
netflix.com IPv6 2a01:578:3::364c:3c27
netflix.com IPv6 2a01:578:3::3411:1b81
netflix.com IPv6 2a01:578:3::22fb:b596
netflix.com IPv6 2a01:578:3::36c2:57d0
netflix.com IPv6 2a01:578:3::36e5:444d
netflix.com IPv6 2a01:578:3::369a:5167
netflix.com IPv6 2a01:578:3::364d:1824
netflix.com IPv6 2a01:578:3::34d1:e0a1
netflix.com IPv4 54.171.116.69
netflix.com IPv4 52.211.94.145
netflix.com IPv4 52.31.182.100
netflix.com IPv4 34.250.41.147
netflix.com IPv4 52.211.42.108
netflix.com IPv4 54.194.155.146
netflix.com IPv4 52.19.40.147
netflix.com IPv4 52.210.7.69
[stlinux02] >
```

Happy Eyeballs

- How to connect to a host with more than one IP address?
  - Perform DNS lookups for IPv4 and IPv6 in parallel; start with whichever completes first
  - Call **`connect()`** for that address; if not connected within ~100ms, start **`connect()`** to next address on the list in parallel, alternating between IPv4 and IPv6
  - Use first **`connect()`** that succeeds, drop other successful connections
  - Balances speed to connect vs. network overload by trying all at once in parallel



Internet Engineering Task Force (IETF) D. Schinazi
Request for Comments: 8305 T. Pauly
Obsoletes: 6555 Apple Inc.
Category: Standards Track December 2017
ISSN: 2070-1721

Happy Eyeballs Version 2: Better Connectivity Using Concurrency

Abstract

Many communication protocols operating over the modern Internet use hostnames. These often resolve to multiple IP addresses, each of which may have different performance and connectivity characteristics. Since specific addresses or address families (IPv4 or IPv6) may be blocked, broken, or sub-optimal on a network, clients that attempt multiple connections in parallel have a chance of establishing a connection more quickly. This document specifies requirements for algorithms that reduce this user-visible delay and provides an example algorithm, referred to as "Happy Eyeballs". This document obsoletes the original algorithm description in RFC 6555.

Status of This Memo

This is an Internet Standards Track document.

https://datatracker.ietf.org/doc/rfc8305/

Summary: Connection Latency

- Two factors affect performance:
  - **Bandwidth** and **RTT**
  - In many cases, RTT dominates
- How to improve performance?
  - Overlap **`connect()`** calls for hosts with multiple addresses
  - Reduce number of TCP connections made
  - Reduce number of requests per connection
  - Overlap TCP and TLS handshakes
    - QUIC transport protocol → lecture 4
  - Reduce RTT
    - Increase speed of light or reduce queuing delay → lectures 5, 6

Impact of TLS and IPv6 on Connection Establishment

- Transport Layer Security (TLS) needs extra RTTs to negotiate security
- Dual-stack IPv4 and IPv6 hosts must race connections for good performance