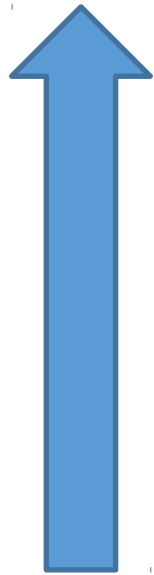
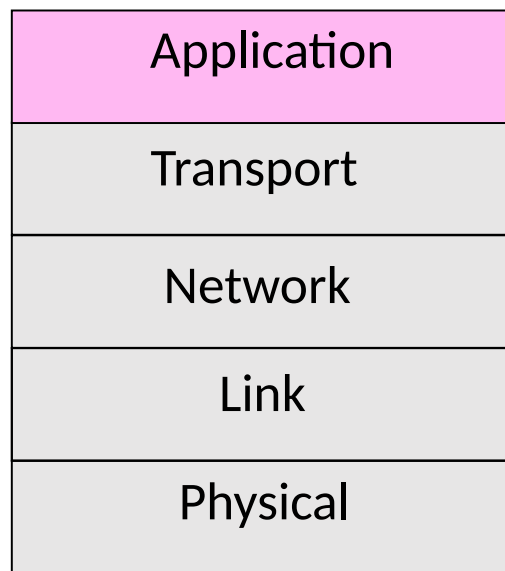


Applications!

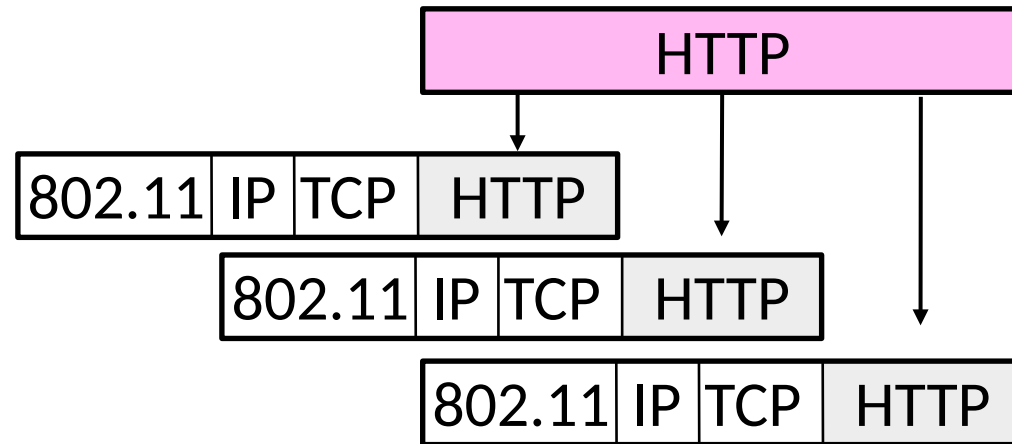
Where we are in the Course

- Application layer protocols are often part of “app”
 - But don't need a GUI, e.g., DNS



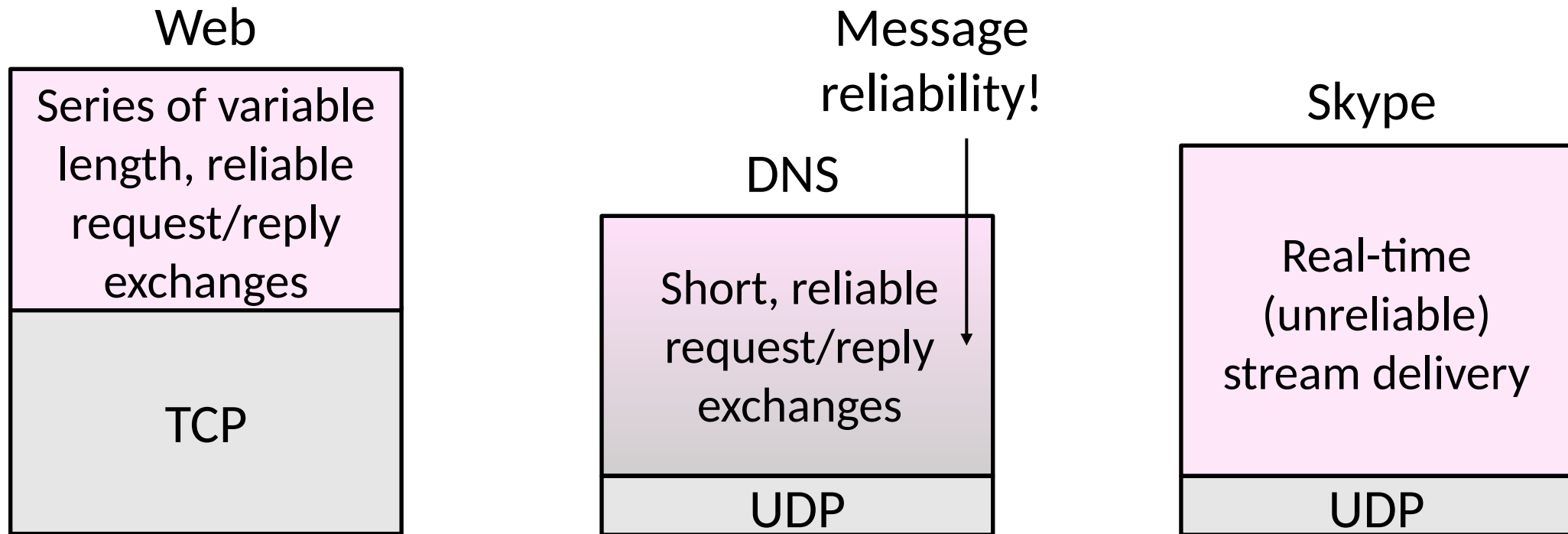
Recall

- Application layer messages are often split over multiple packets
 - Or may be aggregated in a packet ...



Application Communication Needs

- Vary widely; must build on Transport services



OSI Session/Presentation Layers

- Remember this? Two relevant concepts ...

Considered part of the application, not strictly layered!

7	Application	- Provides functions needed by users
6	Presentation	- Converts different representations
5	Session	- Manages task dialogs
4	Transport	- Provides end-to-end delivery
3	Network	- Sends packets over multiple links
2	Data link	- Sends frames of information
1	Physical	- Sends bits as signals

Session Concept

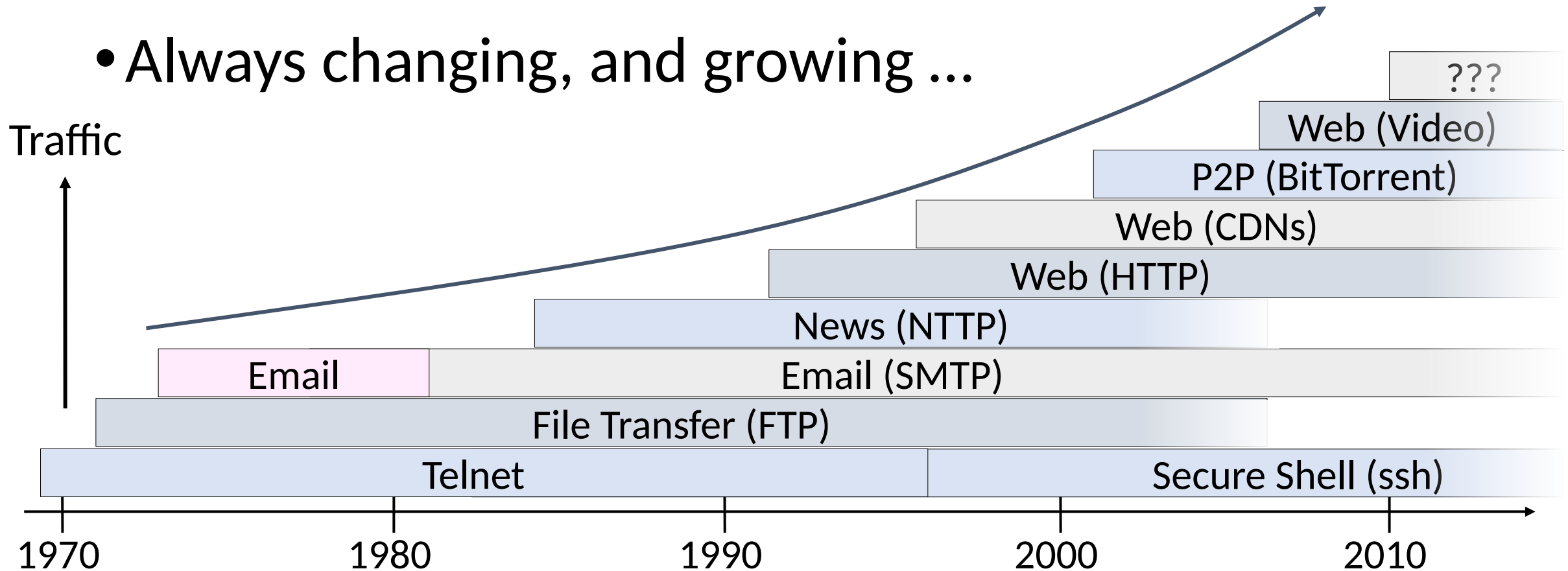
- A session is a series of related network interactions in support of an application task
 - Often informal, not explicit
- Examples:
 - Web page fetches multiple resources
 - Skype call involves audio, video, chat

Presentation Concept

- Apps need to identify the type of content, and encode it for transfer
 - These are Presentation functions
- Examples:
 - Media (MIME) types, e.g., image/jpeg, identify content type
 - Transfer encodings, e.g., gzip, identify the encoding of content
 - Application headers are often simple and readable versus packed for efficiency

Evolution of Internet Applications

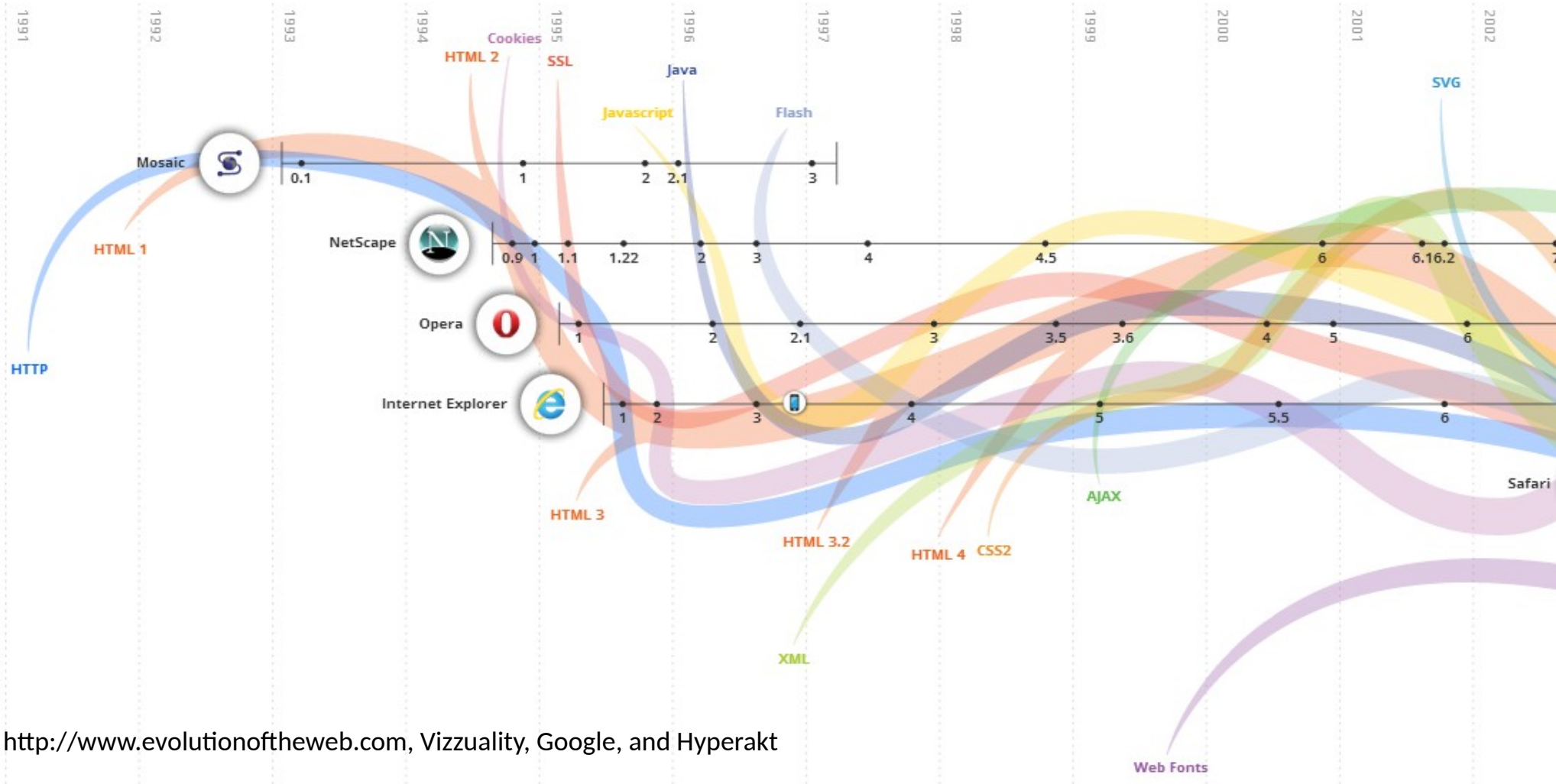
- Always changing, and growing ...



Evolution of Internet Applications (2)

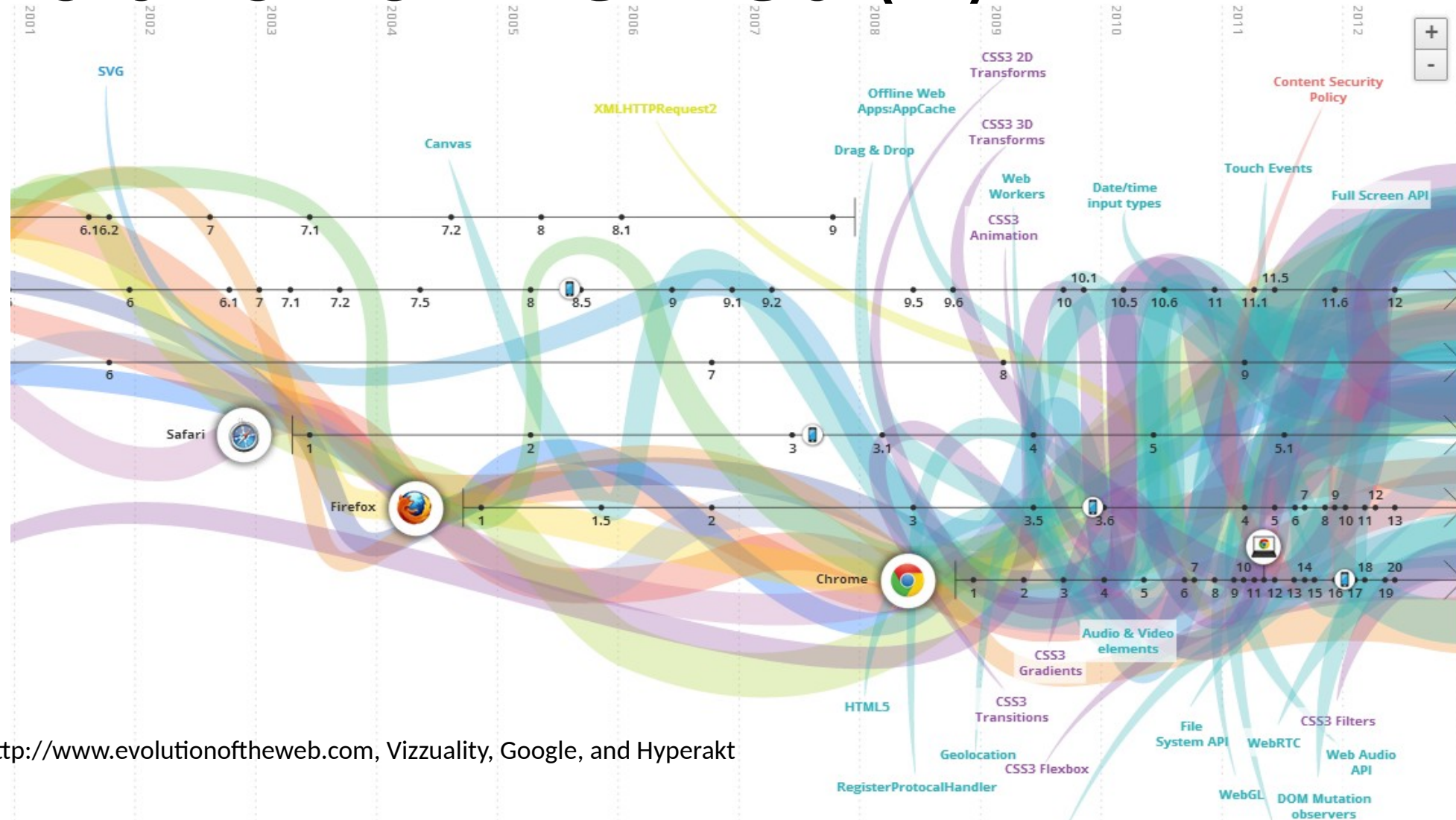
- For a peek at the state of the Internet:
 - Akamai's State of the Internet Report (quarterly)
 - Cisco's Visual Networking Index
 - Mary Meeker's Internet Report
- Robust Internet growth, esp. video, wireless, mobile, cats
 - Most (70%) traffic is video (expected 80% in 2019)
 - Mobile traffic overtakes desktop (2016)
 - 15% of traffic is cats (2013)
 - Growing attack traffic from China, also U.S. and Russia

Evolution of the Web



Source: <http://www.evolutionoftheweb.com>, Vizzuality, Google, and Hyperakt

Evolution of the Web (2)

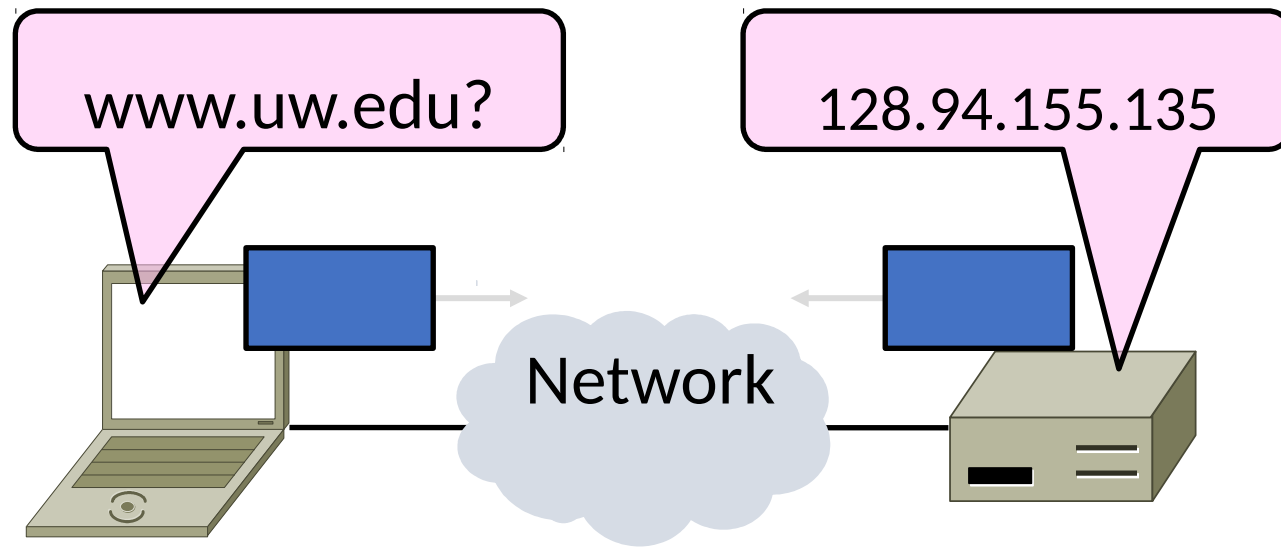


Source: <http://www.evolutionoftheweb.com>, Vizzuality, Google, and Hyperakt

Domain Name System

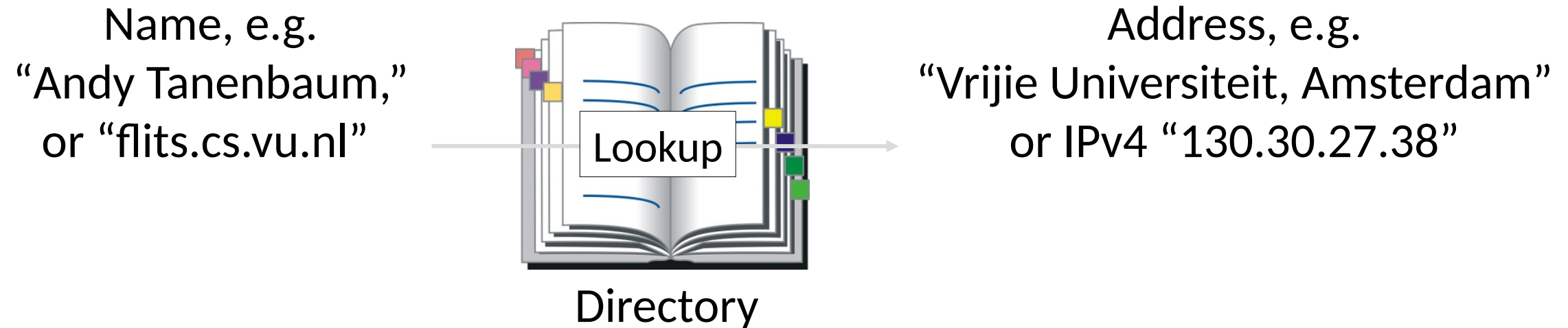
DNS

- Human-readable host names, and more



Names and Addresses

- Names are higher-level identifiers for resources
- Addresses are lower-level locators for resources
 - Multiple levels, e.g. full name → email → IP address → Ethernet addr
- Resolution (or lookup) is mapping a name to an address



Before the DNS - HOSTS.TXT

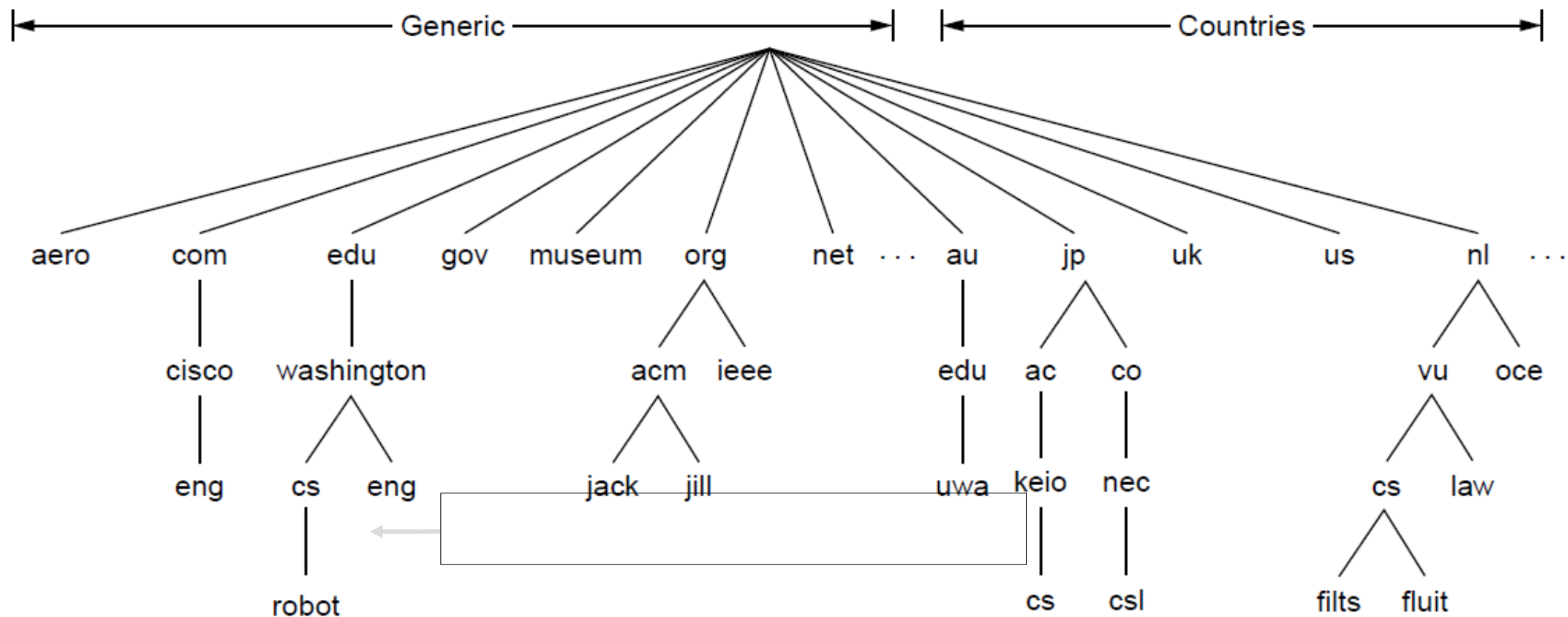
- Directory was a file HOSTS.TXT regularly retrieved for all hosts from a central machine at the NIC (Network Information Center)
- Names were initially flat, became hierarchical (e.g., lcs.mit.edu) ~85
- Not manageable or efficient as the ARPANET grew ...

DNS

- A naming service to map between host names and their IP addresses (and more)
 - `www.uwa.edu.au` → `130.95.128.140`
- Goals:
 - Easy to manage (esp. with multiple parties)
 - Efficient (good performance, few resources)
- Approach:
 - Distributed directory based on a hierarchical namespace
 - Automated protocol to tie pieces together

DNS Namespace

- Hierarchical, starting from “.” (dot, typically omitted)

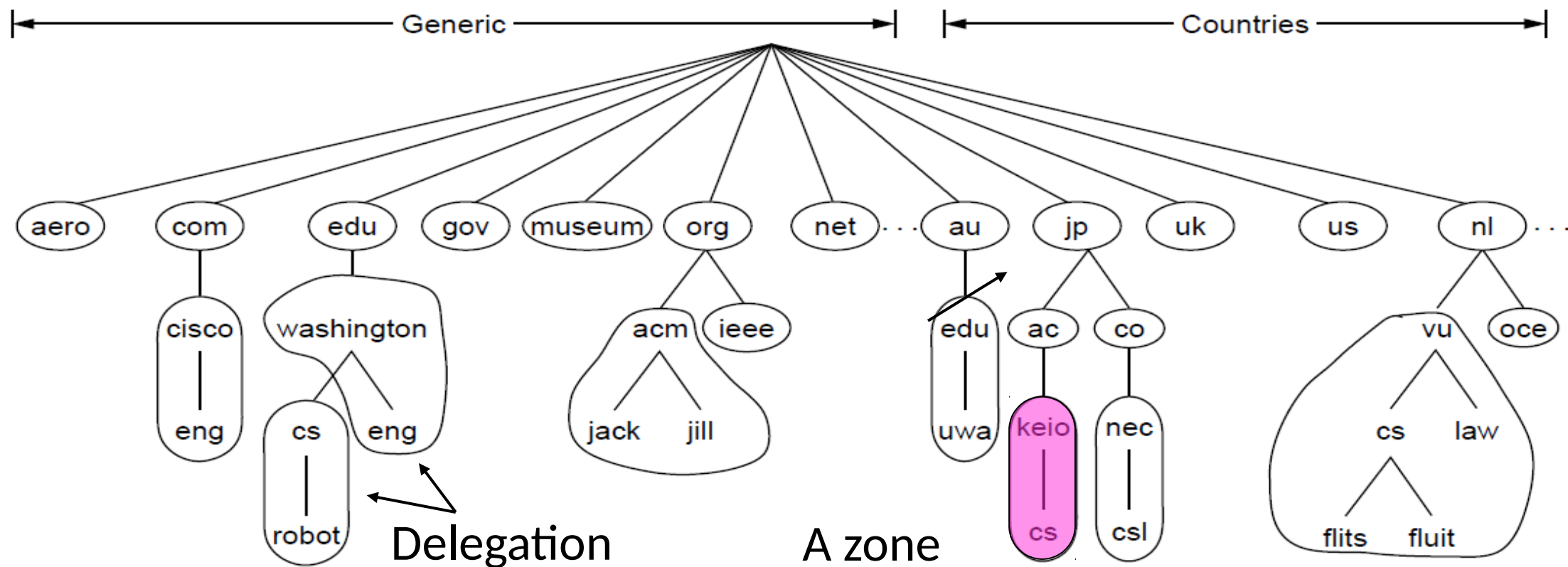


TLDs (Top-Level Domains)

- Run by ICANN (Internet Corp. for Assigned Names and Numbers)
 - Starting in '98; naming is financial, political, and international
- 700+ generic TLDs
 - Initially .com, .edu, .gov., .mil, .org, .net
 - Unrestricted (.com) vs Restricted (.edu)
 - Added regions (.asia, .kiwi), Brands (.apple), Sponsored (.aero) in 2012
- ~250 country code TLDs
 - Two letters, e.g., “.au”, plus international characters since 2010
 - Widely commercialized, e.g., .tv (Tuvalu)
 - Many domain hacks, e.g., instagr.am (Armenia), kurti.sh (St. Helena)

DNS Zones

- A zone is a contiguous portion of the namespace



DNS Zones (2)

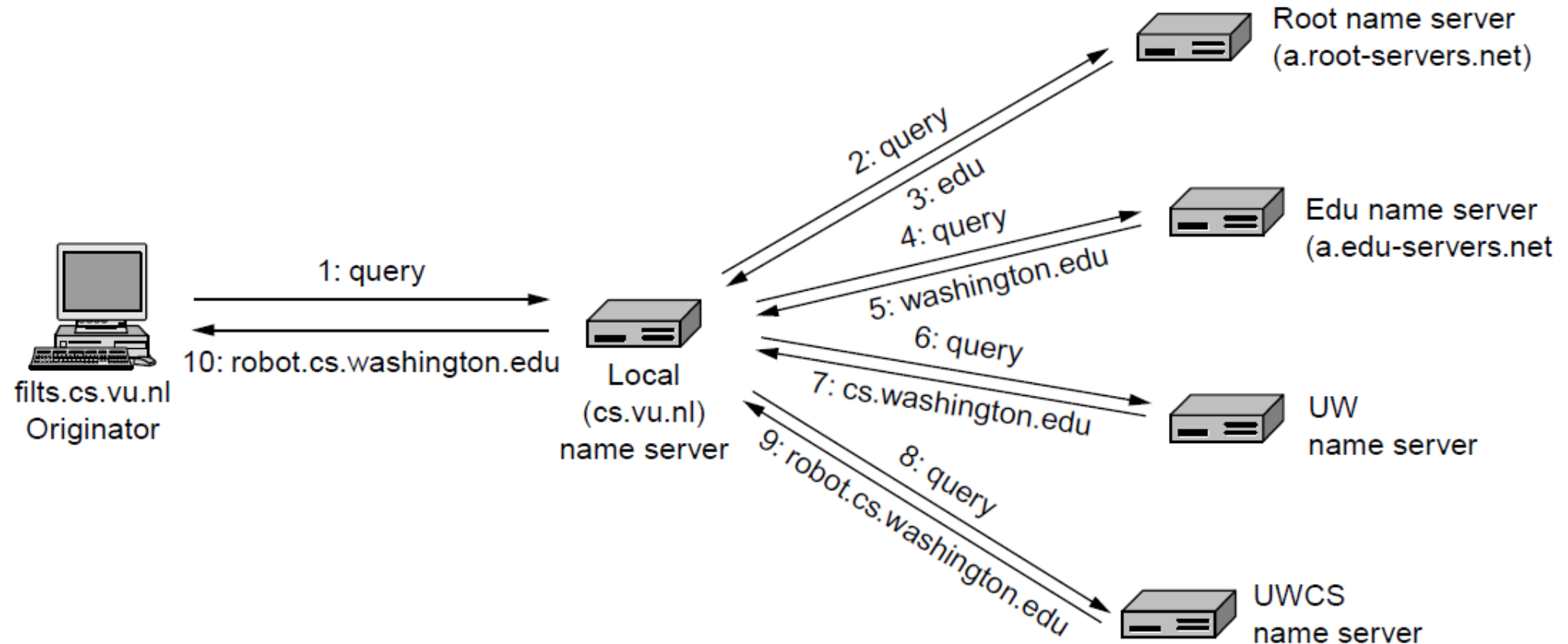
- Zones are the basis for distribution
 - EDU Registrar administers .edu
 - UW administers washington.edu
 - CSE administers cs.washington.edu
- Each zone has a nameserver to contact for information about it
 - Zone must include contacts for delegations, e.g., .edu knows nameserver for washington.edu

DNS Resolution

- DNS protocol lets a host resolve any host name (domain) to IP address
- If unknown, can start with the root nameserver and work down zones
- Let's see an example first ...

DNS Resolution (2)

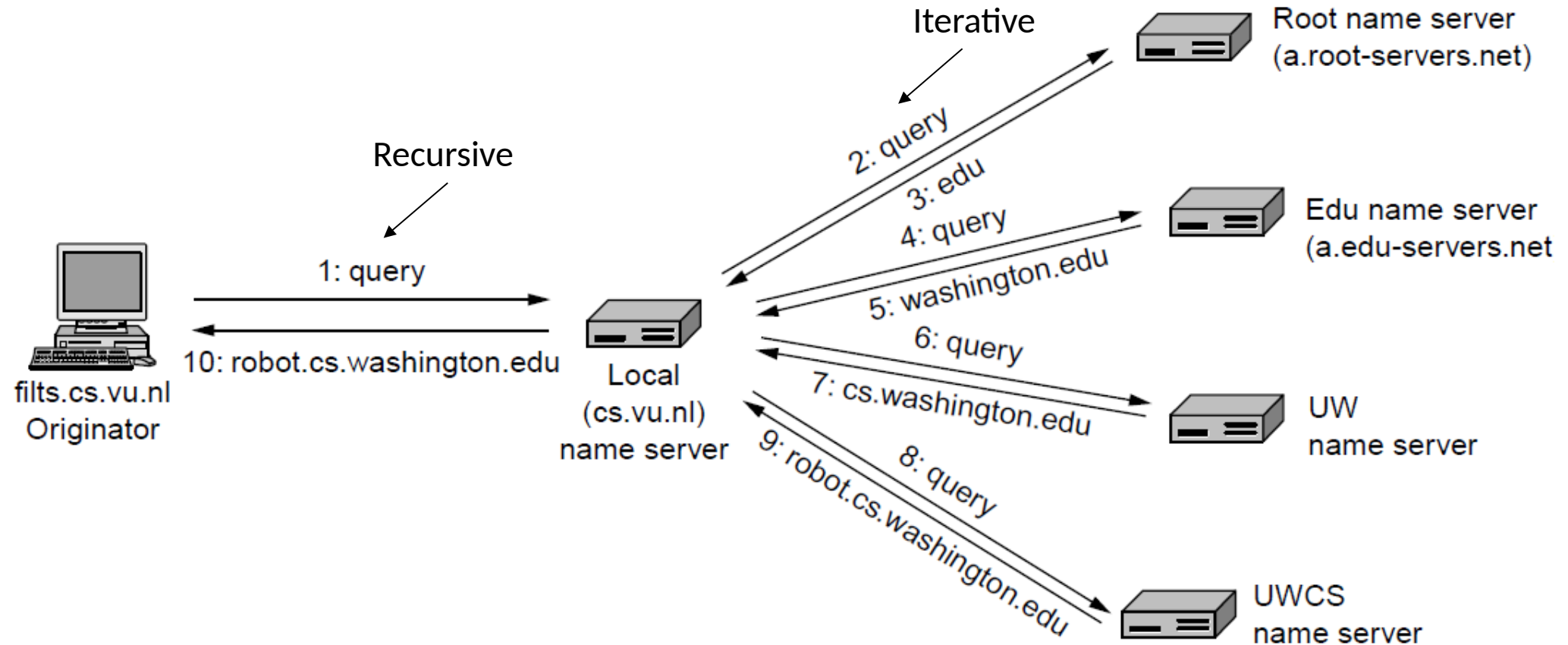
- flits.cs.vu.nl resolves robot.cs.washington.edu



Iterative vs. Recursive Queries

- Recursive query
 - Nameserver resolves and returns final answer
 - E.g., flits → local nameserver
- Iterative (Authoritative) query
 - Nameserver returns answer or who to contact for answer
 - E.g., local nameserver → all others

Iterative vs. Recursive Queries (2)



Iterative vs. Recursive Queries (3)

- Recursive query
 - Lets server offload client burden (simple resolver) for manageability
 - Lets server cache results for a pool of clients
- Iterative query
 - Lets server “file and forget”
 - Easy to build high load servers

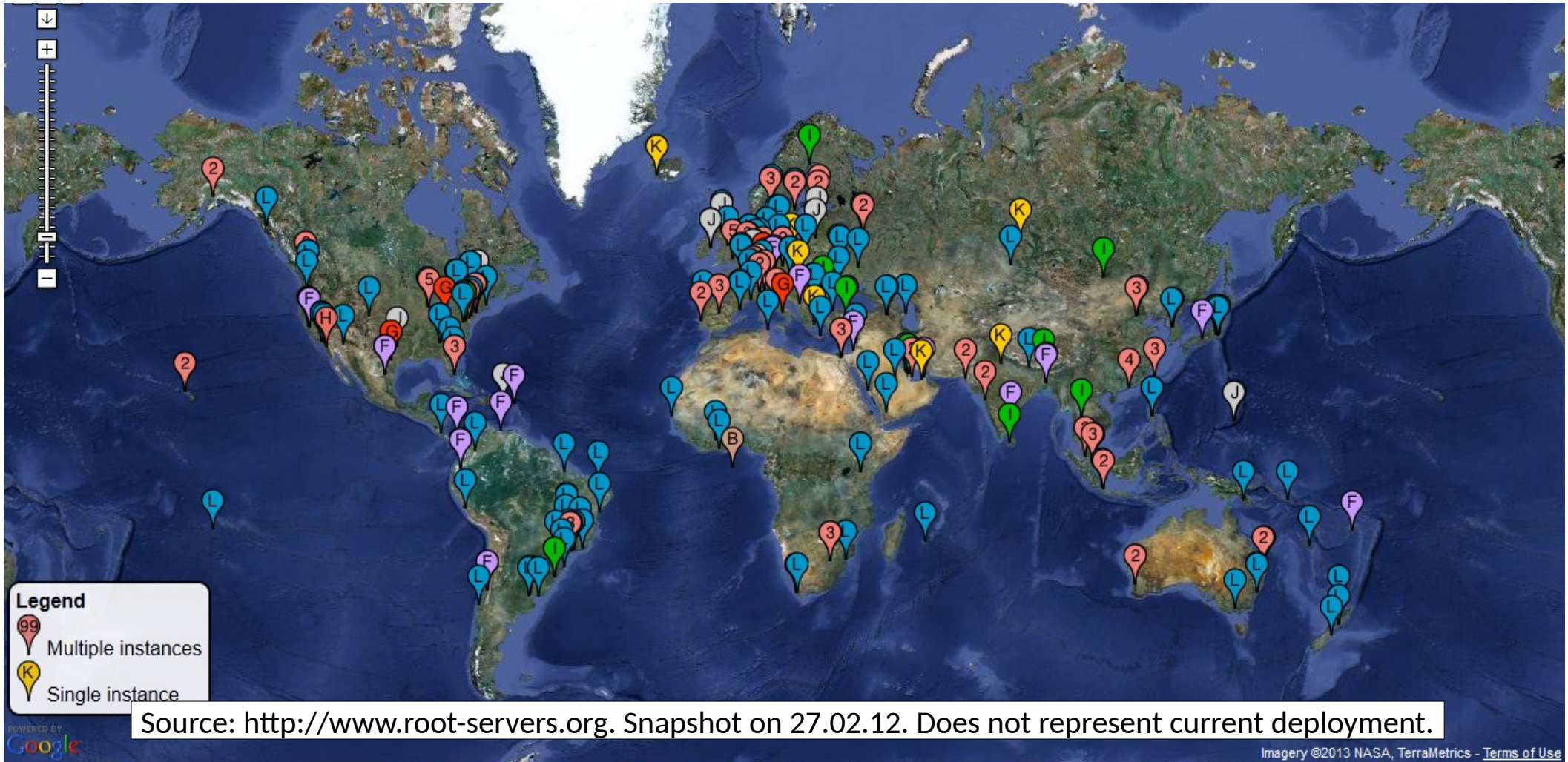
Local Nameservers

- Local nameservers often run by IT (enterprise, ISP)
 - But may be your host or AP
 - Or alternatives e.g., Google public DNS (8.8.8.8)
Cloudflare's public DNS (1.1.1.1)
- Clients need to be able to contact local nameservers
 - Typically configured via DHCP

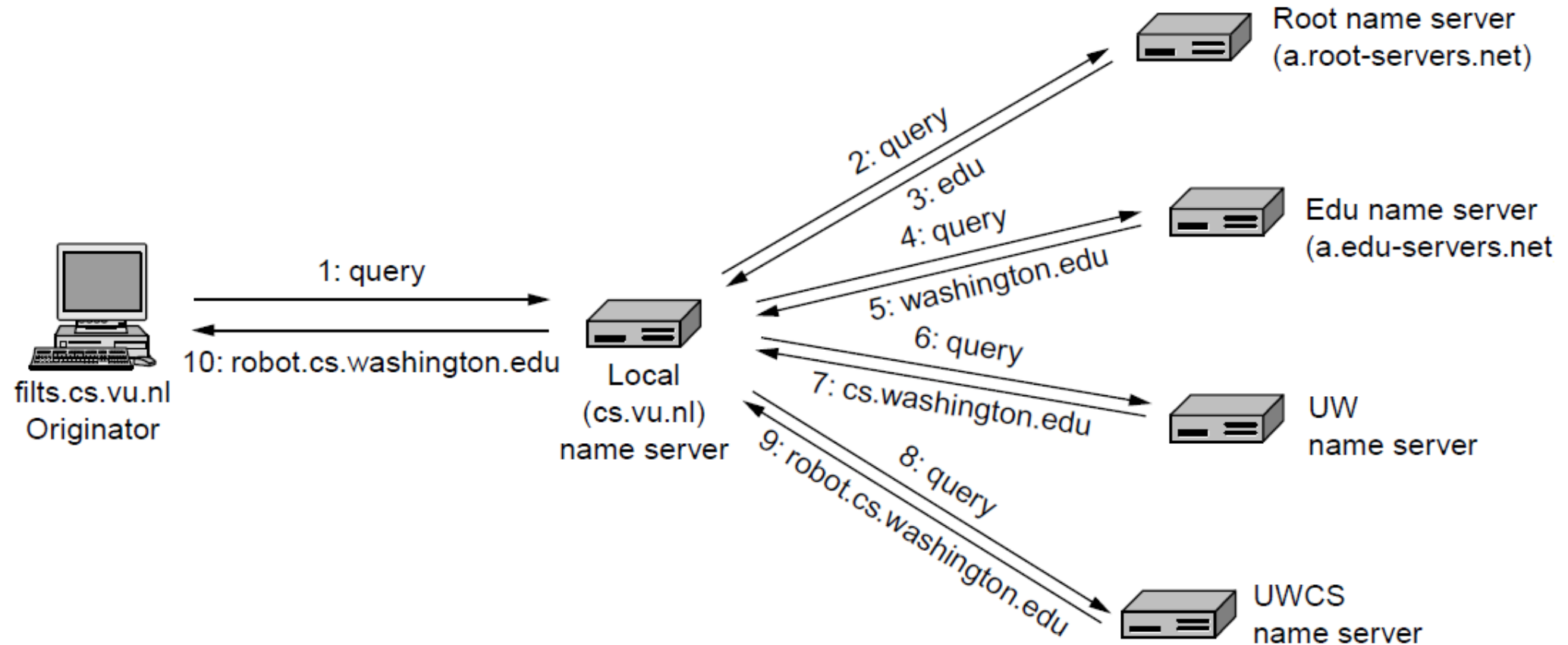
Root Nameservers

- Root (dot) is served by 13 server names
 - a.root-servers.net to m.root-servers.net
 - All nameservers need root IP addresses
 - Handled via configuration file (named.ca)
- There are >1000 distributed server instances
 - Highly reachable, reliable service
 - Most servers are reached by IP anycast (Multiple locations advertise same IP! Routes take client to the closest one.)
 - Servers are IPv4 and IPv6 reachable

Root Server Deployment

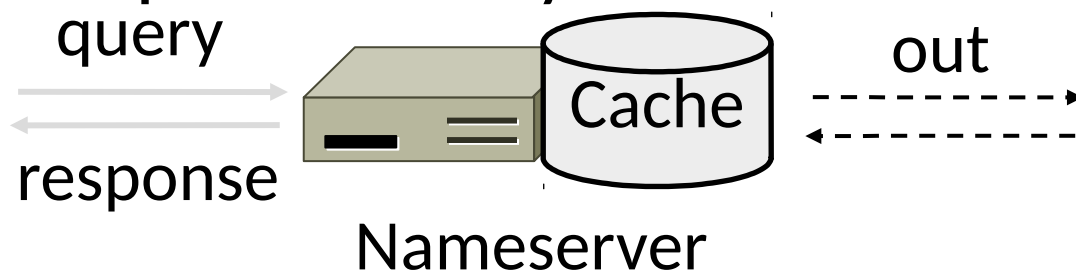


Iterative vs. Recursive Queries (2)



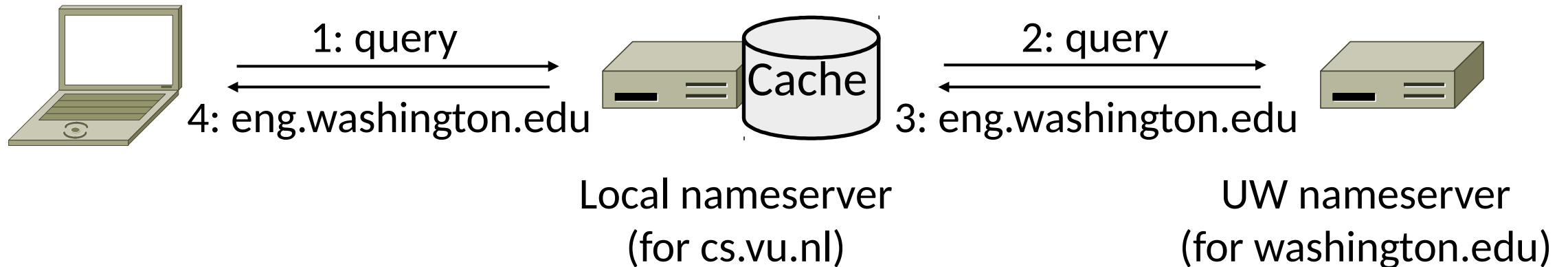
Caching

- Resolution latency needs to be low
- **URLs don't have much churn**
- Cache query/responses to answer future queries immediately
 - Including partial (iterative) answers
 - Responses carry a TTL for caching



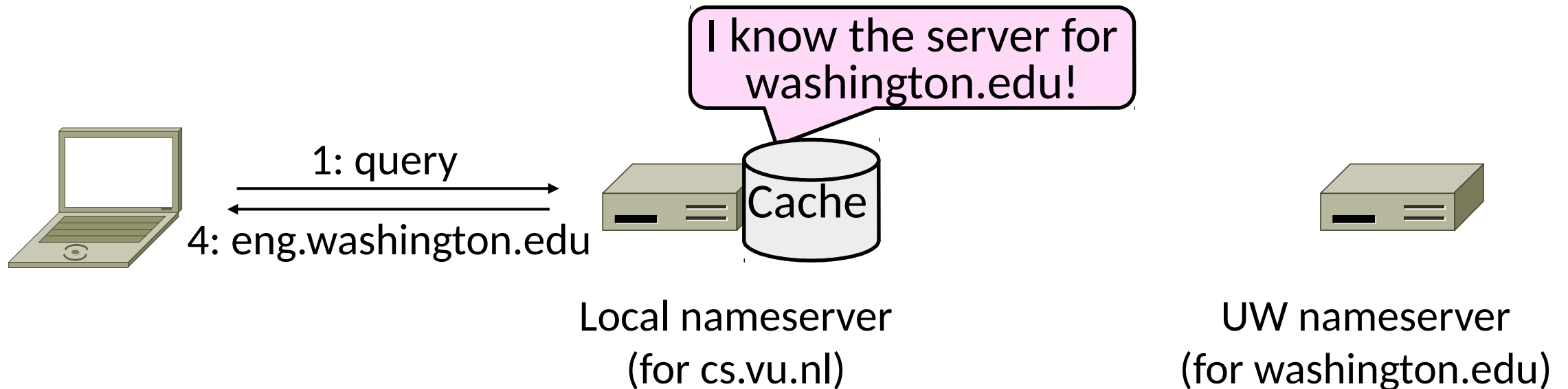
Caching (2)

- flits.cs.vu.nl looks up and stores eng.washington.edu



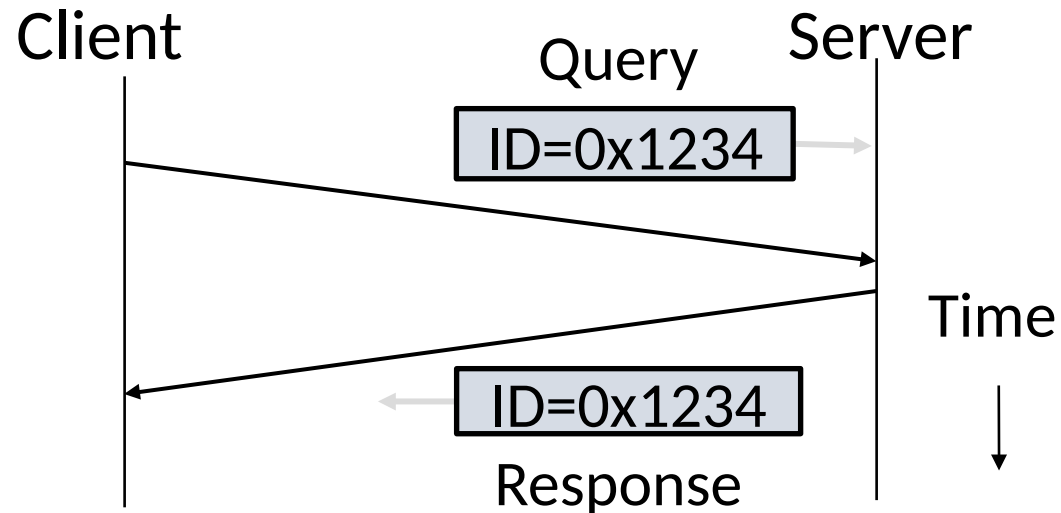
Caching (3)

- flits.cs.vu.nl now directly resolves eng.washington.edu



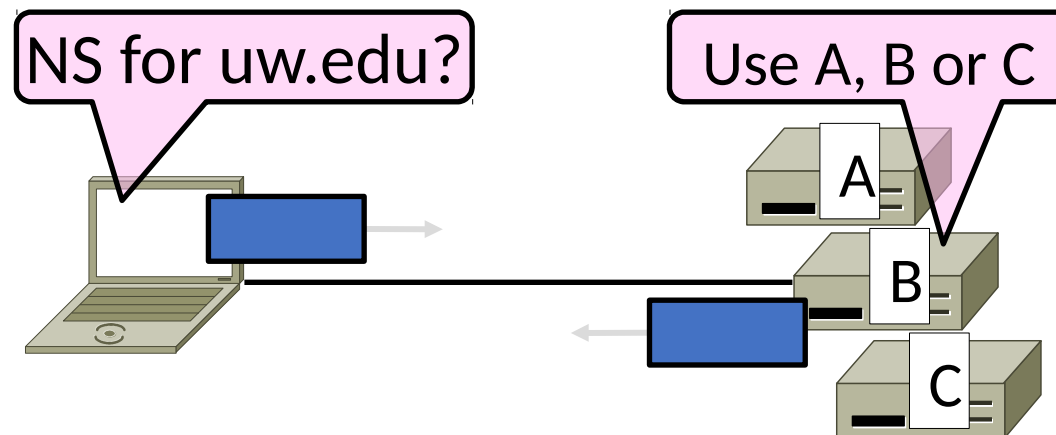
DNS Protocol

- Query and response messages
 - Built on UDP messages, port 53
 - ARQ for reliability; server is stateless!
 - Messages linked by a 16-bit ID field



DNS Protocol (2)

- Service reliability via replicas
 - Run multiple nameservers for domain
 - Return the list; clients use one answer
 - Helps distribute load too



DNS Resource Records

- A zone is comprised of DNS resource records that give information for its domain names

Type	Meaning
SOA	Start of authority, has key zone parameters
A	IPv4 address of a host
AAAA (“quad A”)	IPv6 address of a host
CNAME	Canonical name for an alias
MX	Mail exchanger for the domain
NS	Nameserver of domain or delegated subdomain

DNS Resource Records (2)

; Authoritative data for cs.vu.nl

cs.vu.nl.	86400	IN	SOA	star boss (9527,7200,7200,241920,86400)
cs.vu.nl.	86400	IN	MX	1 zephyr
cs.vu.nl.	86400	IN	MX	2 top
cs.vu.nl.	86400	IN	NS	star
star	86400	IN	A	130.37.56.205
zephyr	86400	IN	A	130.37.20.10
top	86400	IN	A	130.37.20.11
www	86400	IN	CNAME	star.cs.vu.nl
ftp	86400	IN	CNAME	zephyr.cs.vu.nl
flits	86400	IN	A	130.37.16.112
flits	86400	IN	A	192.31.231.165
flits	86400	IN	MX	1 flits
flits	86400	IN	MX	2 zephyr
flits	86400	IN	MX	3 top
rowboat		IN	A	130.37.56.201
		IN	MX	1 rowboat
		IN	MX	2 zephyr
little-sister		IN	A	130.37.62.23
laserjet		IN	A	192.31.231.216

← Start of Authority

← Name server

← IP addresses
of computers

← Mail gateways

DIG DEMO

DNSSEC (DNS Security Extensions)

- Extends DNS with new record types
 - RRSIG for digital signatures of records
 - DNSKEY for public keys for validation
 - DS for public keys for delegation
 - First version in '97, revised by '05
- Deployment requires software upgrade at both client and server
 - Root servers upgraded in 2010
 - Followed by uptick in deployment

Inside 'Operation Black Tulip': DigiNotar hack analysed

CA systems falsely told Iranians they were secure

By [John Leyden](#) 6 Sep 2011 at 14:01

28 

SHARE ▼

The Google webmail of as many as 300,000 Iranians may have been intercepted using fraudulently issued security certificates made after a hack against Dutch certificate authority outfit DigiNotar, according to the preliminary findings of an official report into the megahack.

Fox-IT, the security consultancy hired to examine the breach against DigiNotar, reveals that DigiNotar was hacked on or around 6 June – a month before hackers began publishing rogue certificates.

Between 10 July and 20 July hackers used compromised access to DigiNotar's systems to issue rogue 531 SSL certificate for Google and other domains, including Skype, Mozilla add-ons, Microsoft update and others. DigiNotar only began revoking rogue certificates on 19 July and waited more than a month after this to go public. The fake *.google.com



Not a member of Pastebin yet? [Sign Up](#), it unlocks many cool features!

text 6.00 KB

[raw](#) [download](#) [clone](#) [embed](#) [report](#) [print](#)

```
1. Certificate:
2. Data:
3.   Version: 3 (0x2)
4.   Serial Number:
5.     05:e2:e6:a4:cd:09:ea:54:d6:65:b0:75:fe:22:a2:56
6.   Signature Algorithm: sha1WithRSAEncryption
7.   Issuer:
8.     emailAddress      = info@diginotar.nl
9.     commonName        = DigiNotar Public CA 2025
10.    organizationName   = DigiNotar
11.    countryName        = NL
12.   Validity
13.     Not Before: Jul 10 19:06:30 2011 GMT
14.     Not After : Jul  9 19:06:30 2013 GMT
15.   Subject:
16.     commonName         = *.google.com
17.     serialNumber       = PK000229200002
18.     localityName      = Mountain View
19.     organizationName   = Google Inc
20.     countryName       = US
21.   Subject Public Key Info:
22.     Public Key Algorithm: rsaEncryption
23.     RSA Public Key: (2048 bit)
24.       Modulus (2048 bit):
```


Threat Research

Global DNS Hijacking Campaign: DNS Record Manipulation at Scale

January 10, 2019 | by [Muks Hirani](#), [Sarah Jones](#), [Ben Read](#)

DNS

IRAN

Introduction

FireEye's Mandiant Incident Response and Intelligence teams have identified a wave of DNS hijacking that has affected dozens of domains belonging to government, telecommunications and internet infrastructure entities across the Middle East and North Africa, Europe and North America. While we do not currently link this activity to any tracked group, initial research suggests the actor or actors responsible have a nexus to Iran. This campaign has targeted victims across the globe on an almost unprecedented scale, with a high degree of success. We have been tracking this activity for several months, mapping and understanding the innovative tactics, techniques and procedures (TTPs) deployed by the attacker. We have also worked closely with victims, security organizations, and law enforcement agencies where possible to reduce the impact of the attacks and/or prevent further compromises.

While this campaign employs some traditional tactics, it is differentiated from other Iranian activity we have seen by leveraging DNS hijacking at scale. The attacker uses this technique for their initial foothold, which can then be exploited in a variety of ways. In this blog post, we detail the three different ways we have seen DNS records be manipulated to enable victim compromises. Technique 1, involving the creation of a Let's Encrypt certificate and changing the A record, was [previously documented by Cisco's TALOS team](#). The activity described in their blog post is a subset of the activity we have observed.

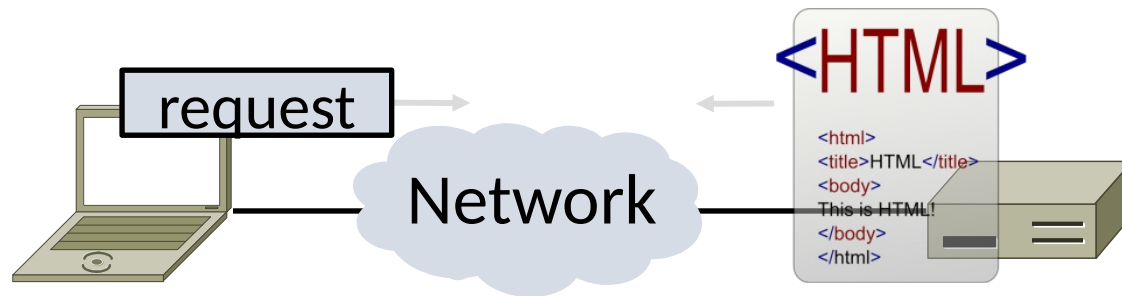
Initial Research Suggests Iranian Sponsorship

Attribution analysis for this activity is ongoing. While the DNS record manipulations described in this post are noteworthy and sophisticated, they may not be exclusive to a single threat actor as the activity spans disparate timeframes, infrastructure, and service providers.

HTTP

HTTP, (HyperText Transfer Protocol)

- Basis for fetching Web pages



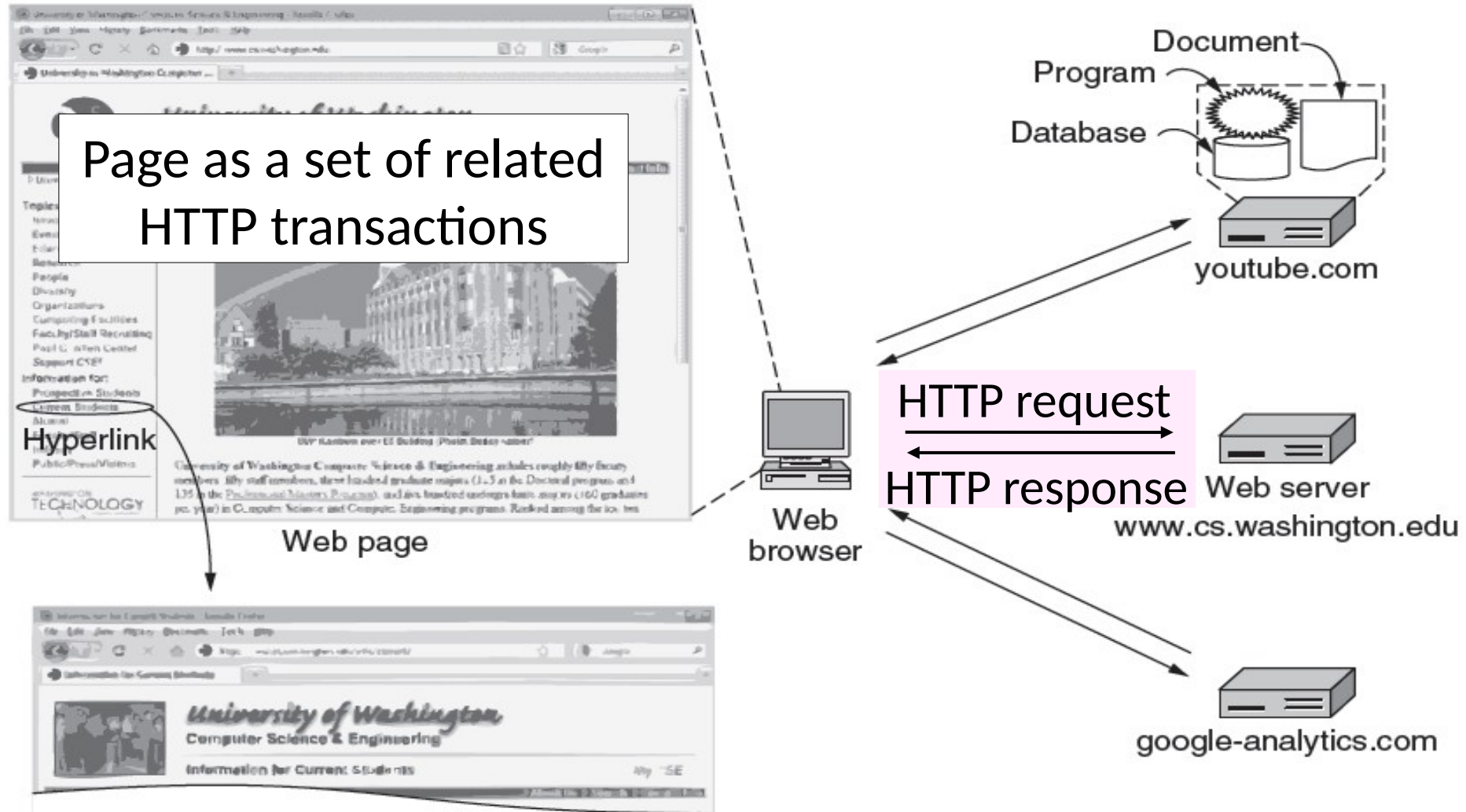
Sir Tim Berners-Lee (1955-)

- Inventor of the Web
 - Dominant Internet app since mid 90s
 - He now directs the W3C
- Developed Web at CERN in '89
 - Browser, server and first HTTP
 - Popularized via Mosaic ('93), Netscape
 - First WWW conference in '94 ...



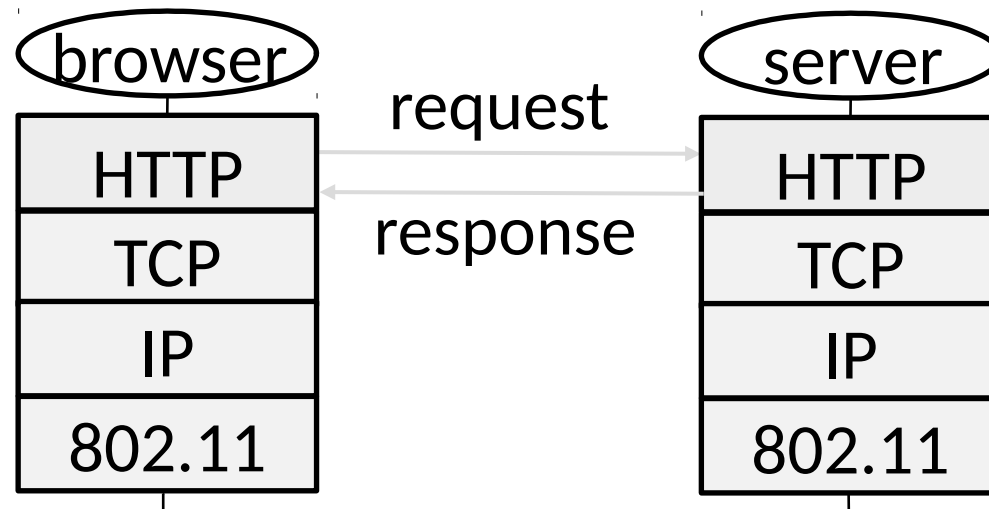
Source: By Paul Clarke, CC-BY-2.0, via Wikimedia Commons

Web Context



Web Protocol Context

- HTTP is a request/response protocol for fetching Web resources
 - Runs on TCP, typically port 80
 - Part of browser/server app



Fetching a Web page with HTTP

- Start with the page URL (Uniform Resource Locator):

`http://en.wikipedia.org/wiki/Vegemite`

Protocol

Server

Page on server

- Steps:
 - Resolve the server to IP address (DNS)
 - Set up TCP connection to the server
 - Send HTTP request for the page
 - (Await HTTP response for the page)
 - Execute/fetch embedded resources/render
 - Clean up any idle TCP connections

HTML

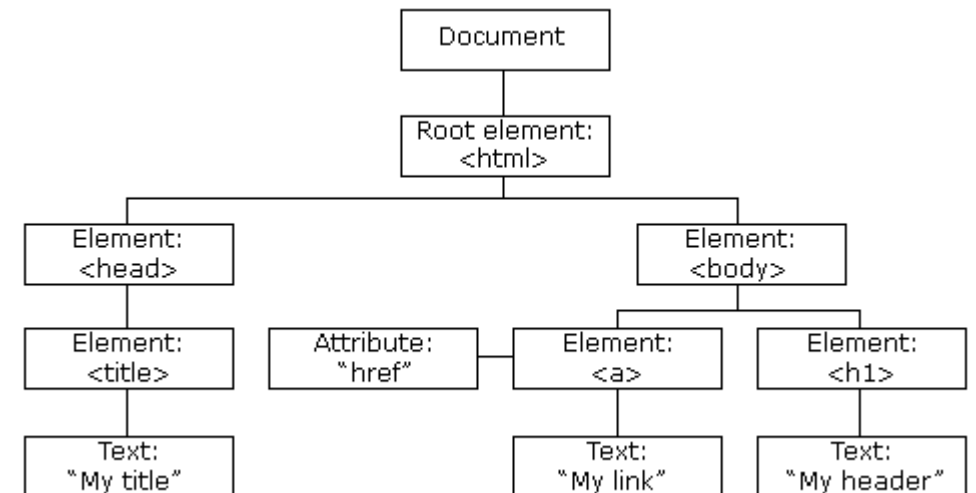
- Hypertext Markup Language (HTML)
 - Uses Extensible Markup Language (XML) to build a markup language for web content
 - Key innovation was the “hyperlink”, an HTML element linking to other HTML elements using URLs
 - Also includes Cascading Style Sheets (CSS) for maintaining look-and-feel across a domain
 - Specific standards have been the subject of many “browser wars”

HTML



DOM (Document Object Model)

- Base primitive for web browsers interacting with HTML
- Use HTML (XML) to create a tree of elements
- Javascript code is embedded in the page and modifies the DOM based on:
 - User actions
 - Asynchronous Javascript
 - Other server-side actions



DOM Example

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
|
```

DOM Examples

- Go to browser and show DOM

HTTP Protocol

- Originally a simple protocol, with many options added over time
 - Text-based commands, headers
- Try it yourself:
 - As a “browser” fetching a URL
 - Run “telnet en.wikipedia.org 80”
 - Type “GET /wiki/Vegemite HTTP/1.0” to server followed by a blank line
 - Server will return HTTP response with the page contents (or other info)

HTTP Protocol (2)

- Commands used in the request

	Method	Description	
Fetch page →	GET	Read a Web page	
	HEAD	Read a Web page's header	
Upload data →	POST	Append to a Web page	
	PUT	Store a Web page	← Basically
	DELETE	Remove the Web page	← defunct
	TRACE	Echo the incoming request	
	CONNECT	Connect through a proxy	
	OPTIONS	Query options for a page	

HTTP Protocol (3)

- Codes returned with the response

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

Yes! →

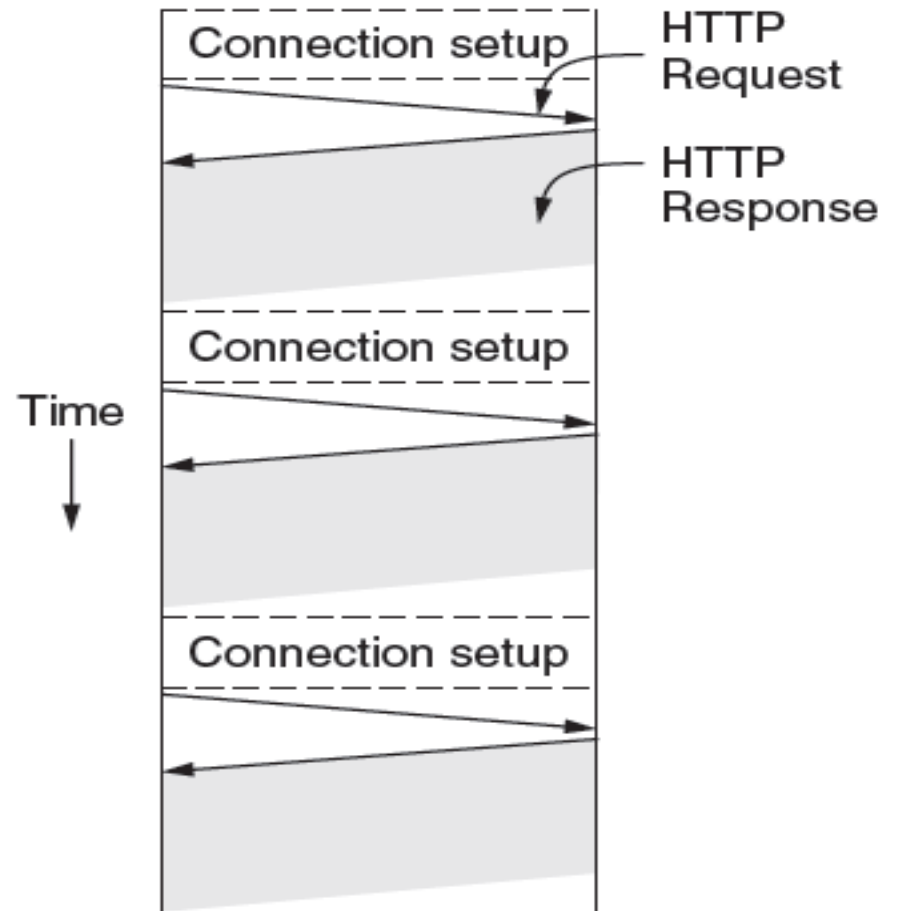
HTTP Performance

PLT (Page Load Time)

- PLT was the key measure of web performance
 - From click until user sees page
 - Small increases in PLT decrease sales
- PLT depends on many factors
 - Structure of page/content
 - HTTP (and TCP!) protocol
 - Network RTT and bandwidth

Early Performance

- HTTP/1.0 used one TCP connection to fetch one web resource
 - Made HTTP very easy to build
 - But gave fairly poor PLT ...



Remember: DOM Example

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

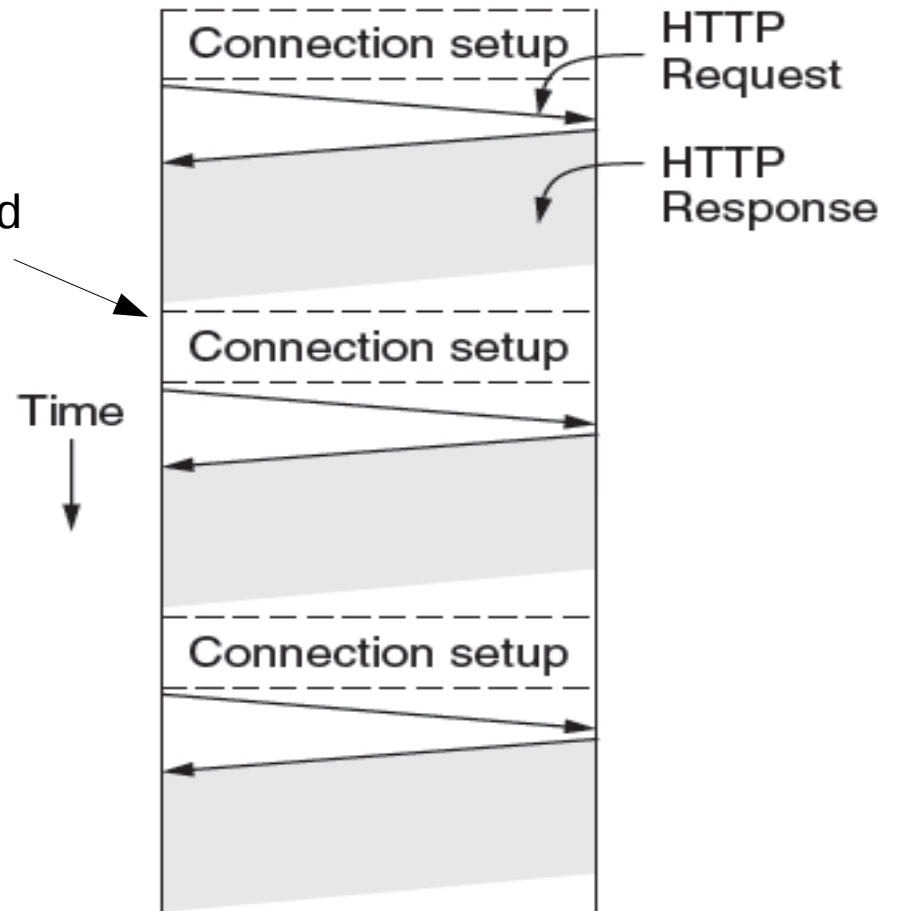
<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
|
```

Early Performance (2)

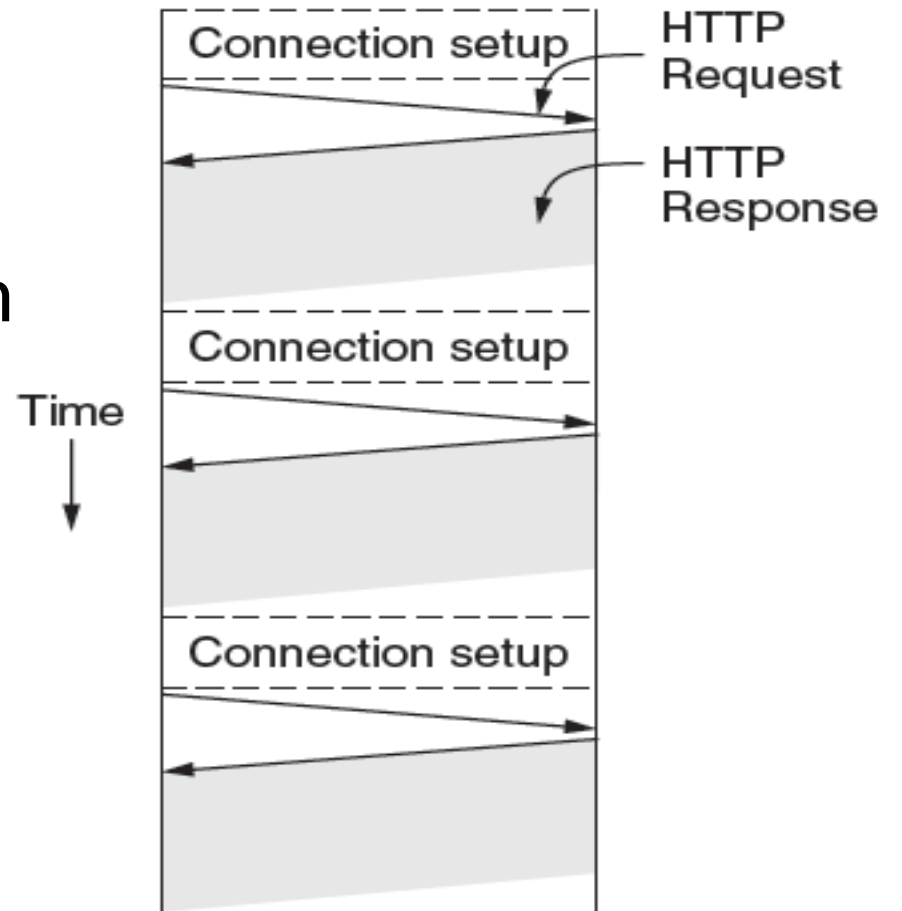
- HTTP/1.0 used one TCP connection to fetch one web resource
 - Made HTTP very easy to build
 - But gave fairly poor PLT...

Oh we need styles.css



Early Performance (3)

- Many reasons why PLT is larger than necessary
 - Sequential request/responses, even when to different servers
 - Multiple TCP connection setups to the same server
 - Multiple TCP slow-start phases
- Network is not used effectively
 - Worse with many small resources / page



Ways to Decrease PLT

1. Reduce content size for transfer
 - Smaller images, gzip
2. Change HTTP to make better use of bandwidth
3. Change HTTP to avoid repeat sending of same content
 - Caching, and proxies
4. Move content closer to client
 - CDNs [later]

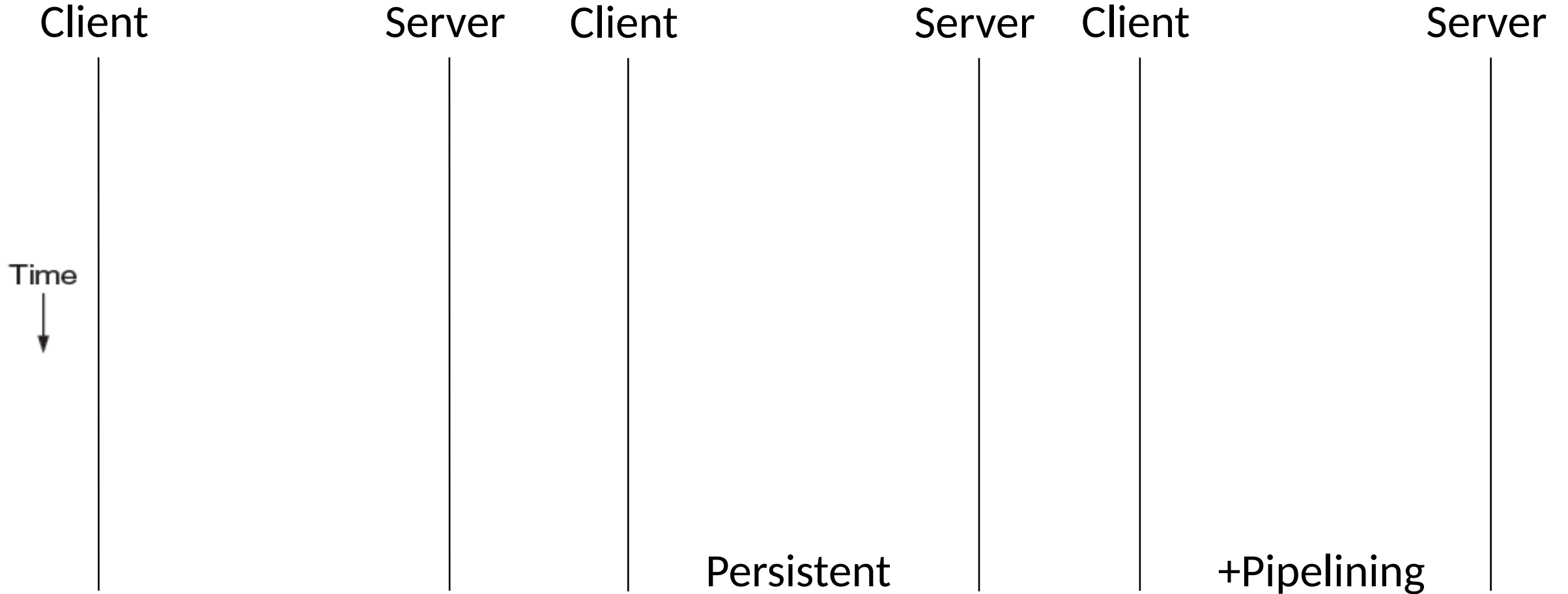
Parallel Connections

- One simple way to reduce PLT
 - Browser runs multiple (8, say) HTTP instances in parallel
 - Server is unchanged; already handled concurrent requests for many clients
- How does this help?
 - Single HTTP wasn't using network much ...
 - So parallel connections aren't slowed much
 - Pulls in completion time of last fetch

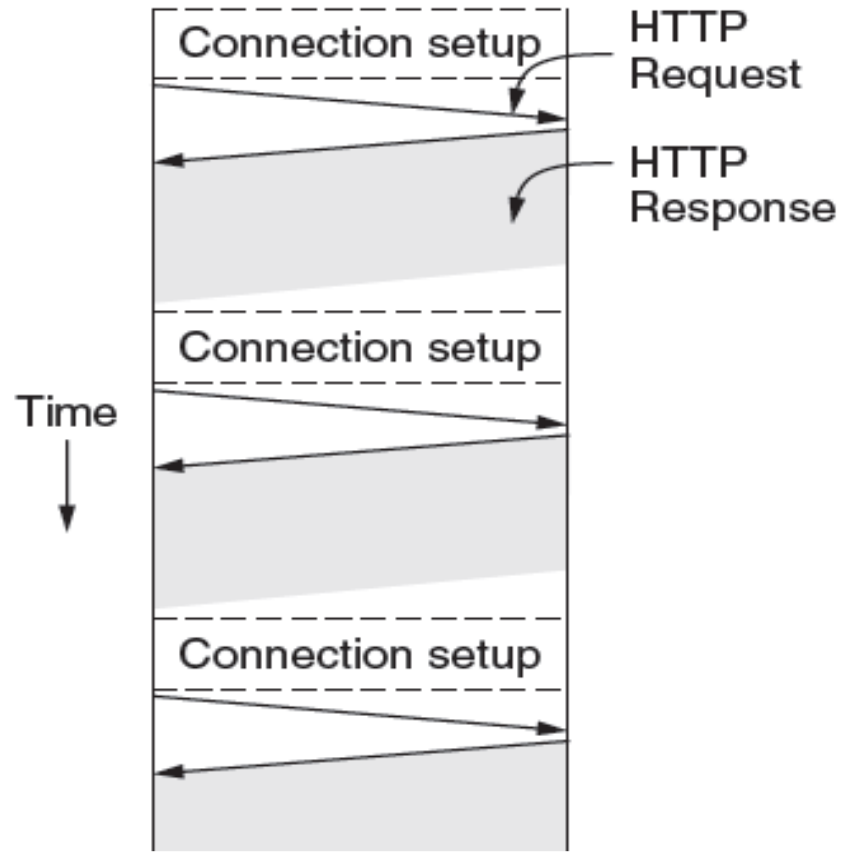
Persistent Connections

- Parallel connections compete with each other for network resources
 - 1 parallel client \approx 8 sequential clients?
 - Exacerbates network bursts, and loss
- Persistent connection alternative
 - Make 1 TCP connection to 1 server
 - Use it for multiple HTTP requests

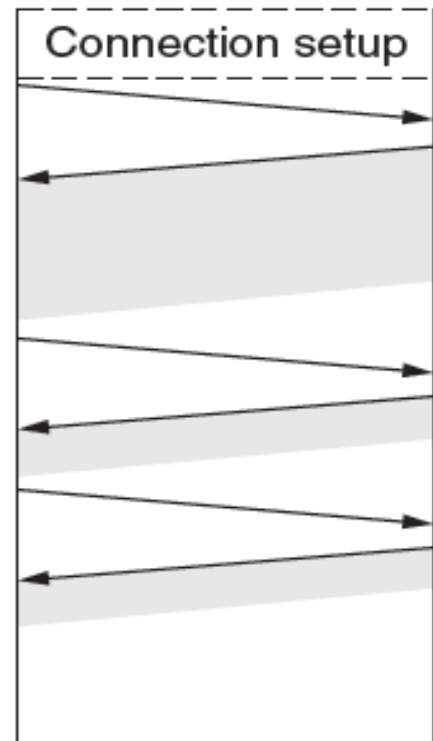
Persistent Connections (2)



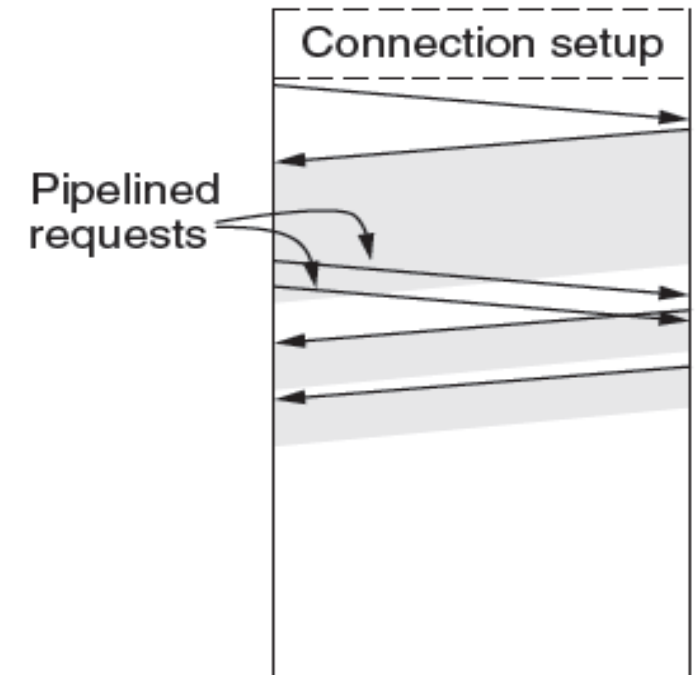
Persistent Connections (3)



One request per connection



Sequential requests per connection



Pipelined requests per connection

Persistent Connections (4)

- Widely used as part of HTTP/1.1
 - Supports optional pipelining
 - PLT benefits depending on page structure, but easy on network

HTTP Futures

HTTP 1.1

- This was it! Standard protocol until circa 2015.
- HTTP 1.1 everywhere for all web access
- Until our favorite massive web company started noticing some trends....

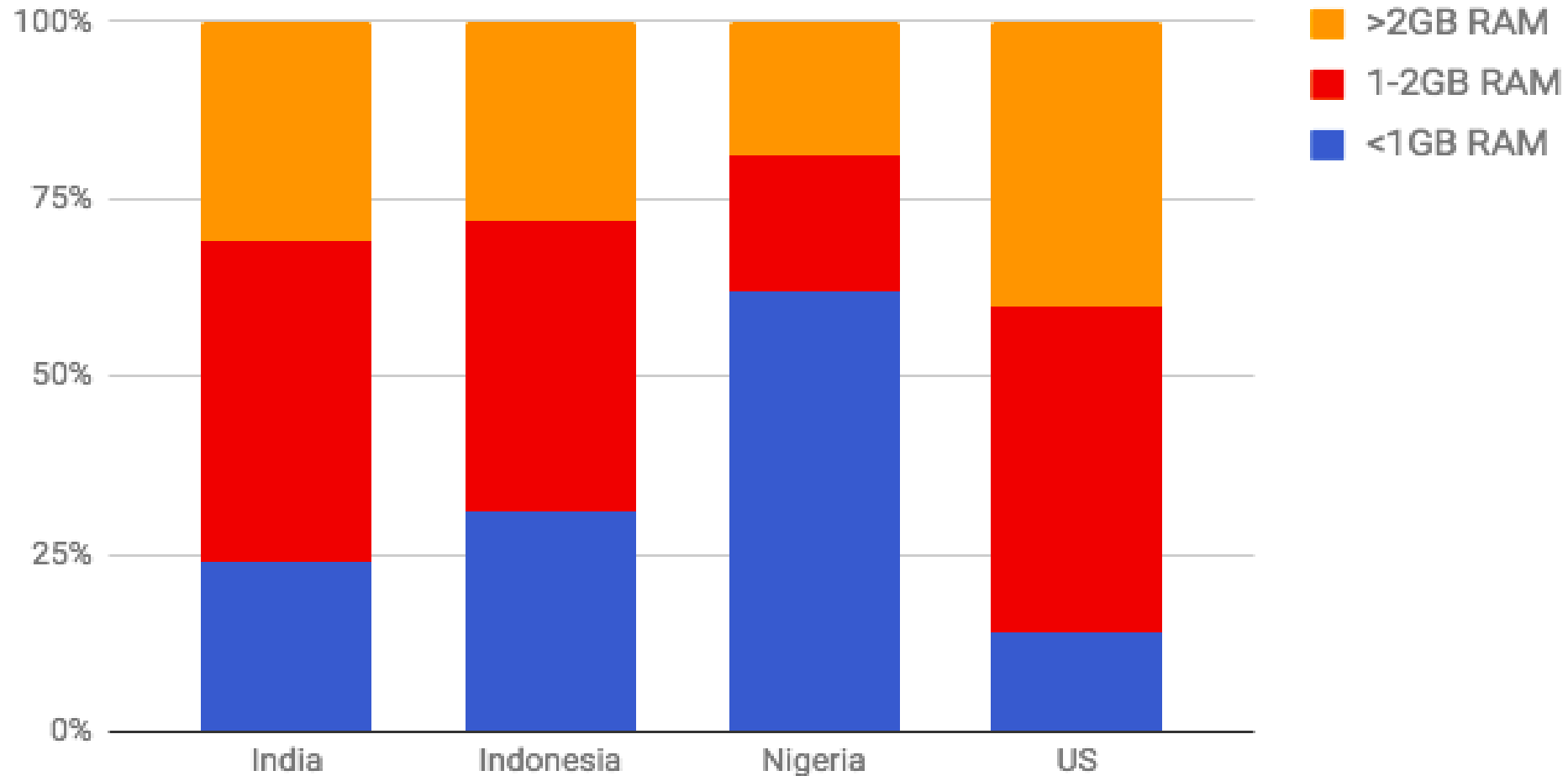
Continued Growth

Country	Mobile-Only Internet Users
Egypt	70%
India	59%
South Africa	57%
Indonesia	44%
United States	25%



Continued Growth (2)

RAM on Android Devices



Continued Growth (3)



Tecno Y2

512MB RAM, 8GB ROM
1.3GHz dual-core Cortex-A7
2G & 3G only
4" (480x800)



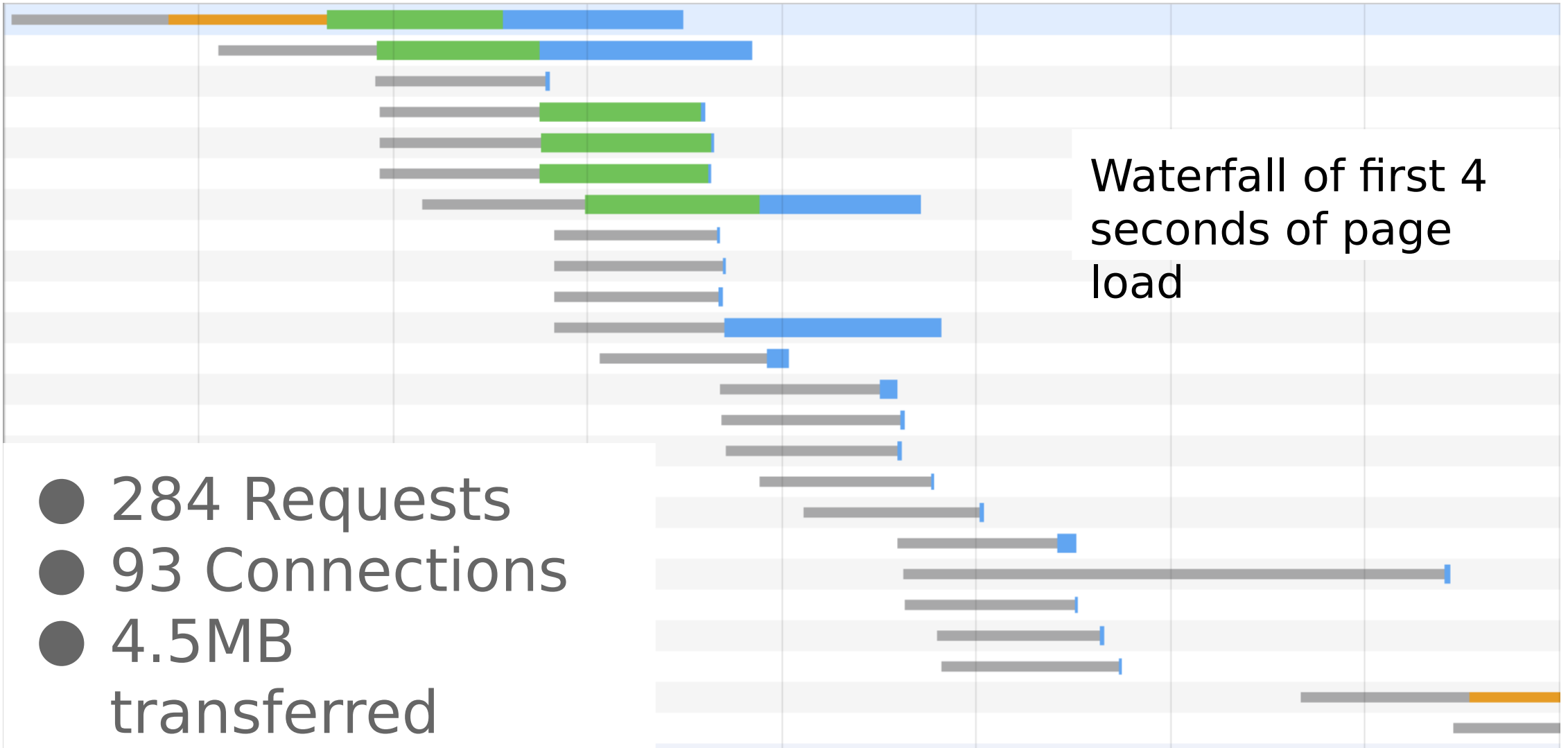
Tecno W3

1GB RAM, 8GB ROM
1.3GHz dual-core Cortex-A7
2G & 3G only
5" (480x854)



Infinix Hot 4 Lite

1GB RAM, 16GB ROM
1.3GHz quad-core Cortex-A7
2G & 3G only
5.5" (720x1280)



- 284 Requests
- 93 Connections
- 4.5MB transferred
- Lots of gaps

Key user moments (PLT is Dumb)



■ First Contentful Paint (FCP) “is it happening?”

■ First Meaningful Paint (FMP) “is it useful?”

■ Time to Interactive (TTI) “is it usable?”

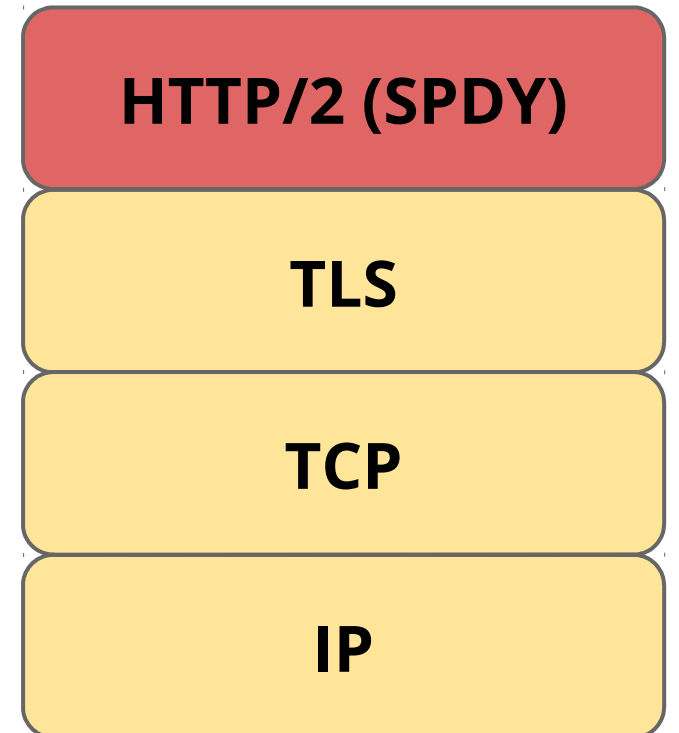
HTTP Changes

HTTP/1.0: TCP connection per request

HTTP/1.1: Persistence and pipelining

HTTP2/SPDY: Targeted performance specifically

- All happens below HTTP layer
- Prioritized stream multiplexing
- Header compression
- Server push
- Started as SPDY, standardized as HTTP/2 in 2015 after ~~every possible bikeshed~~ deep discussion



HTTP 2 Optimizations

Prioritized Stream Multiplexing

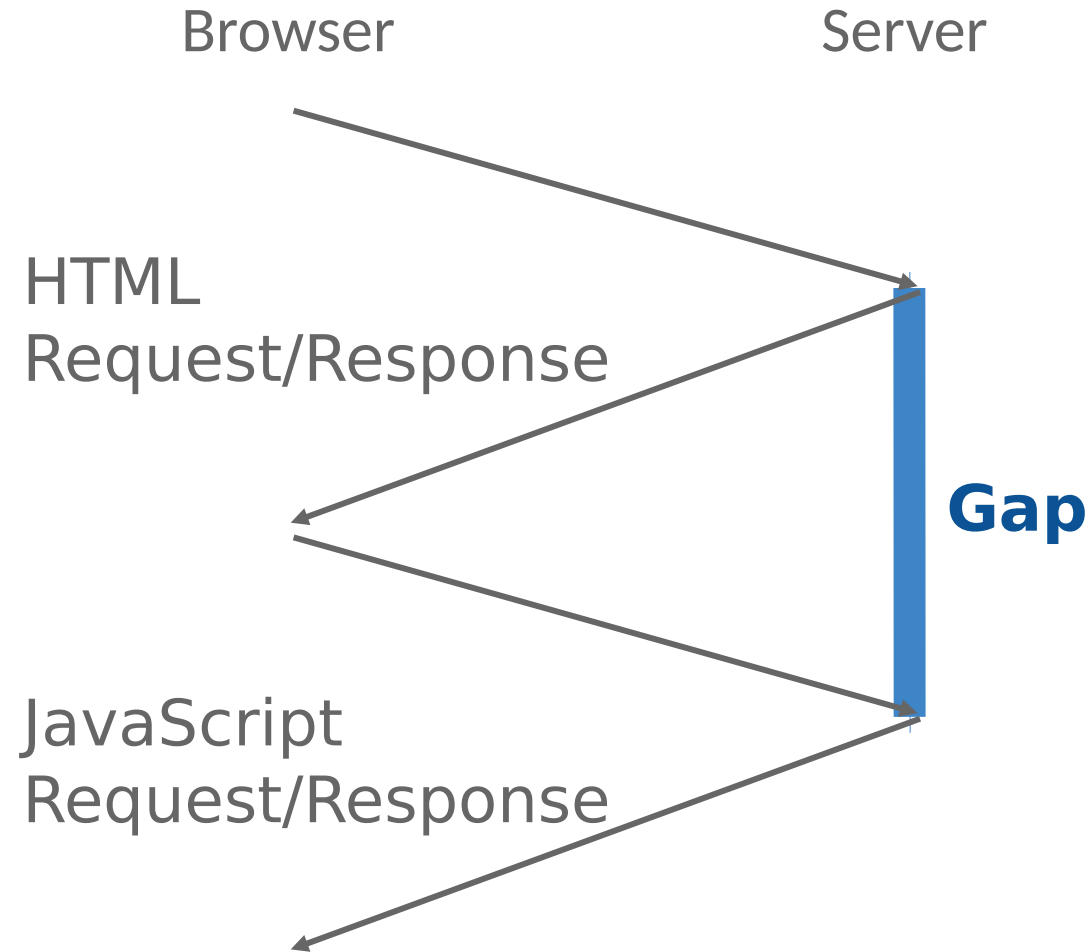
- HTTP 1.0: Each HTTP connection has own TCP
- HTTP 1.1: Share one TCP connection to save setup
- HTTP 2.0: Allow multiple *concurrent* HTTP connections in a single TCP flow to avoid head-of-line blocking

Header Compression

- HTTP Headers very wordy; Designed to be human readable
- This was dumb. Lets compress them (usually gzip).

Server Push: example resource loading gap

- Browser requests and receives HTML, encounters `<script src="...">`
- Similarly, JavaScript might src a dependent JavaScript file

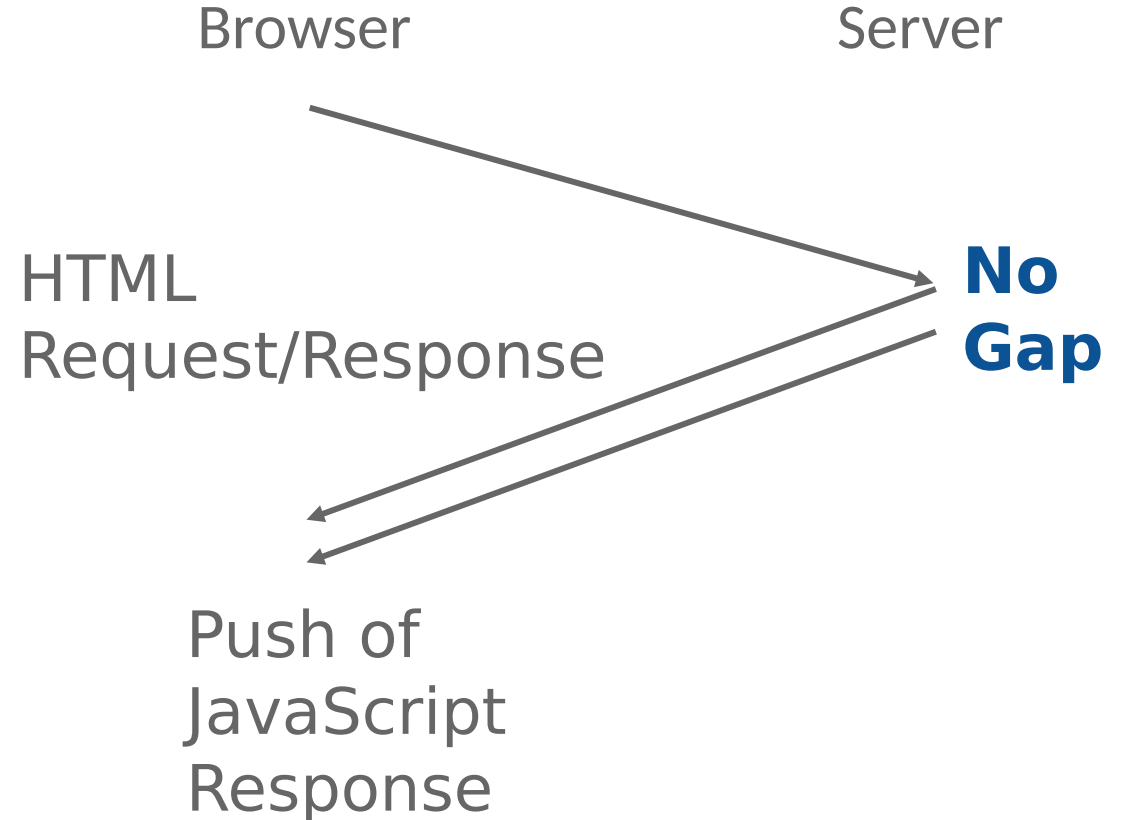


Server Push: example resource loading gap

Use HTTP/2 server push to close gaps

Or use `Link: rel=preload`

- Particularly useful for **hidden render blocking resources (HRBRs)**



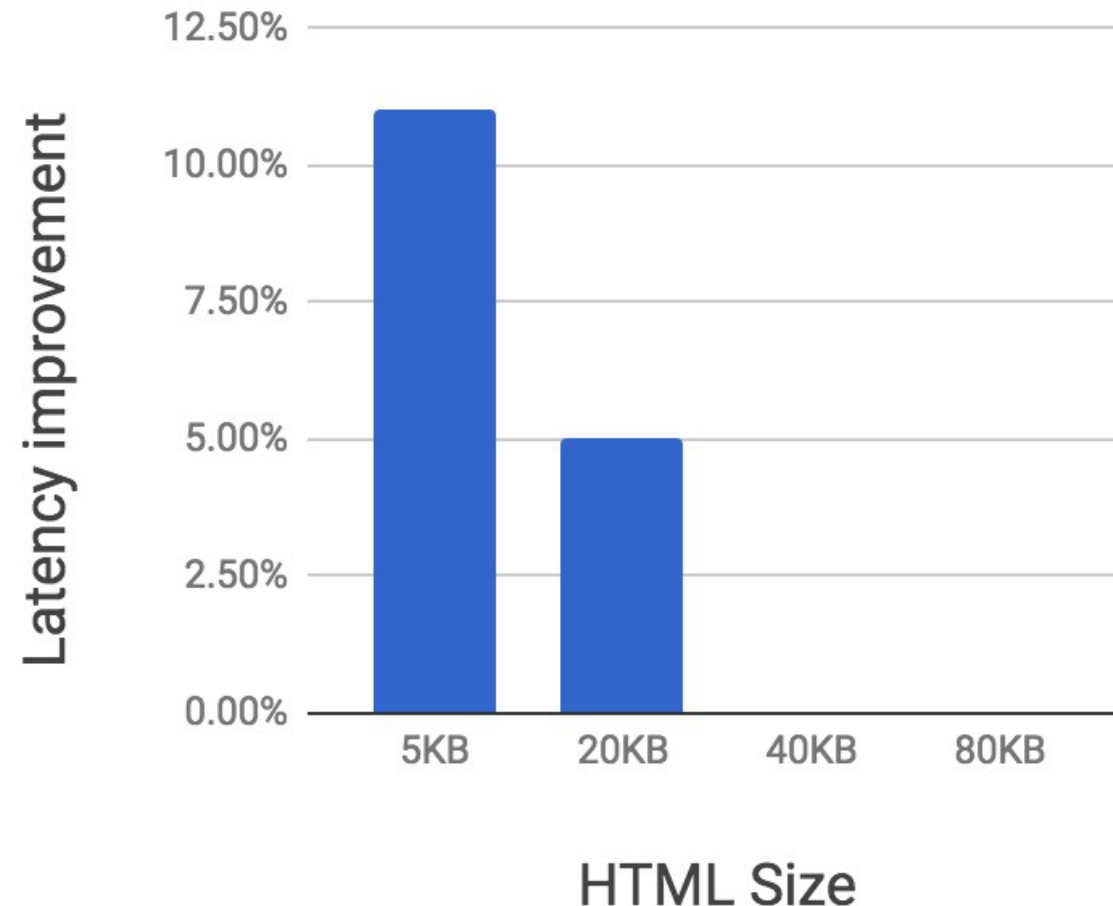
Simple server push lab experiment

Result: No benefit when
HTML size > BD Product

Why? No gap even
without push.

Opportunity only on
high BDP networks,
e.g., LTE and Cable

Latency improvement vs. HTML Size
(3G, BDP = 35KB)



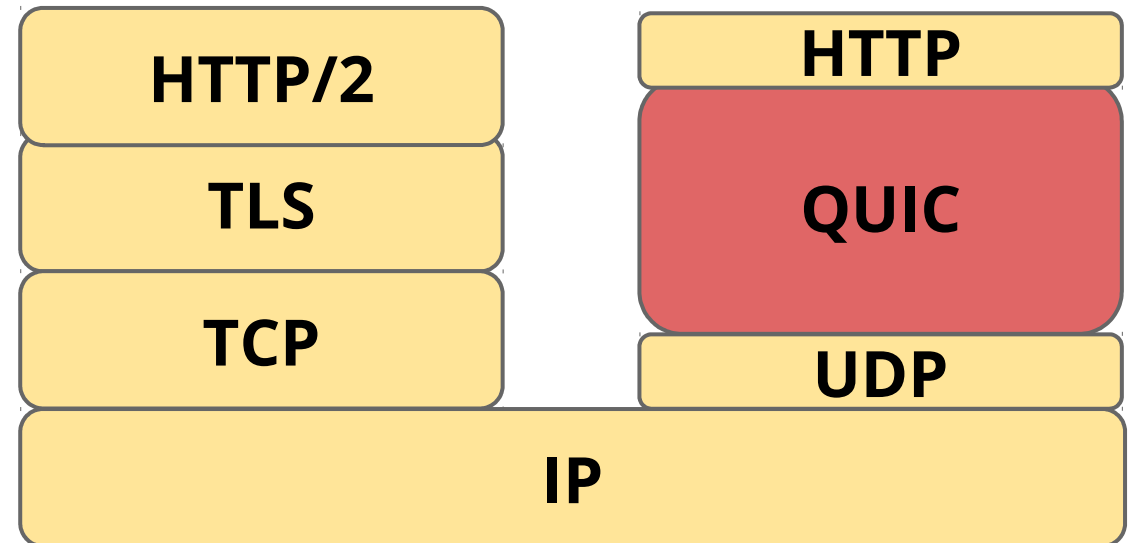
QUIC/HTTP 3.0

Goal: make HTTPS transport **even faster!**

Deployed at Google starting **2014**

IETF working group formed in **2016**

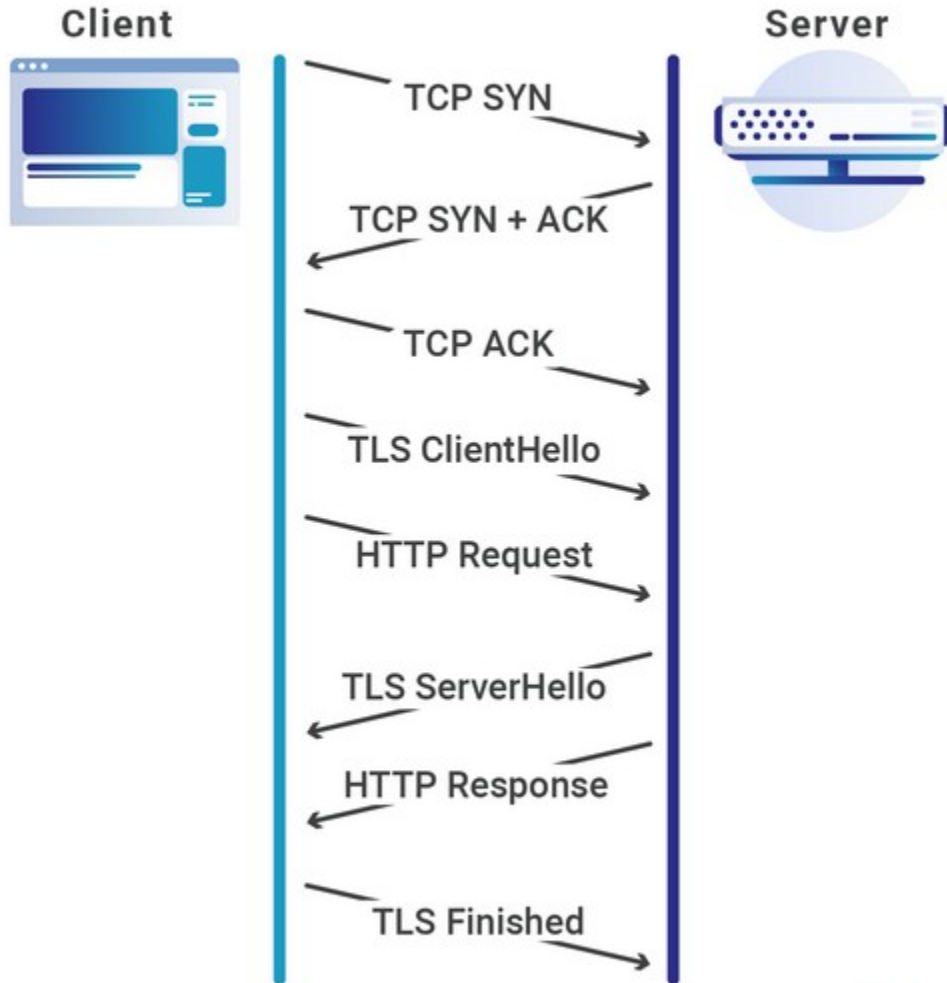
Standardized as HTTP 3.0 in
October **2018**



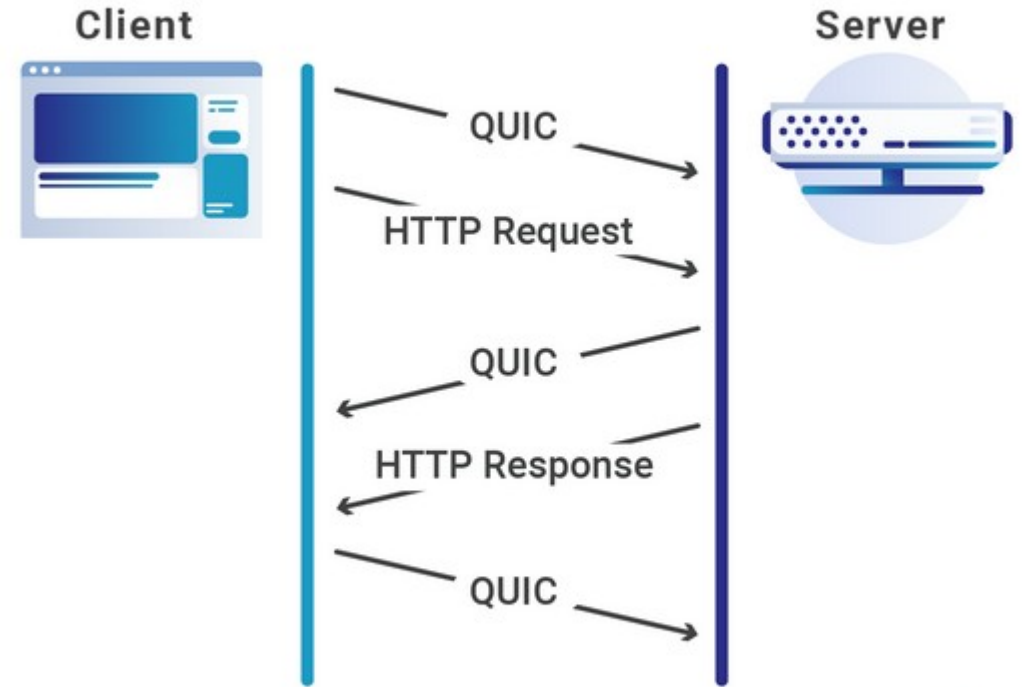
QUIC/HTTP 3.0 Innovations (1)

- Speed up connection establishment
 - Include TLS/Encryption in setup (TLS 1.3)
 - Similarly pack HTTP content into setup

HTTP Request over TCP+TLS (with 0-RTT)



HTTP Request over QUIC (with 0-RTT)



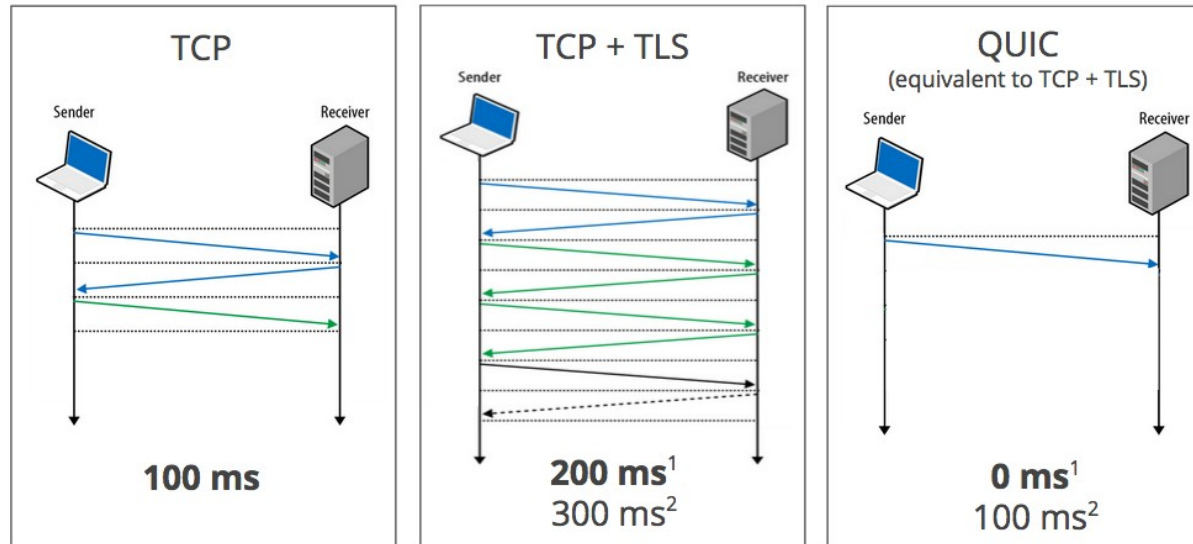
QUIC/HTTP 3.0 Innovations (2)

- Remove TCP/Switch to UDP
 - Error correction: Groups of packets contain a FEC packet which can be used to recreate lost packet.
 - Congestion control: Move congestion control to user space with pluggable implementations
 - BBR Implementation: all packets carry new sequence numbers, allows for precise roundtrip-time calculation.
 - Per-packet encryption (rather than flow)

QUIC/HTTP 3.0 Innovations (3)

- Support mobility through 64-bit stream IDs
 - This means you can change IP address or ports but still keep your connection alive

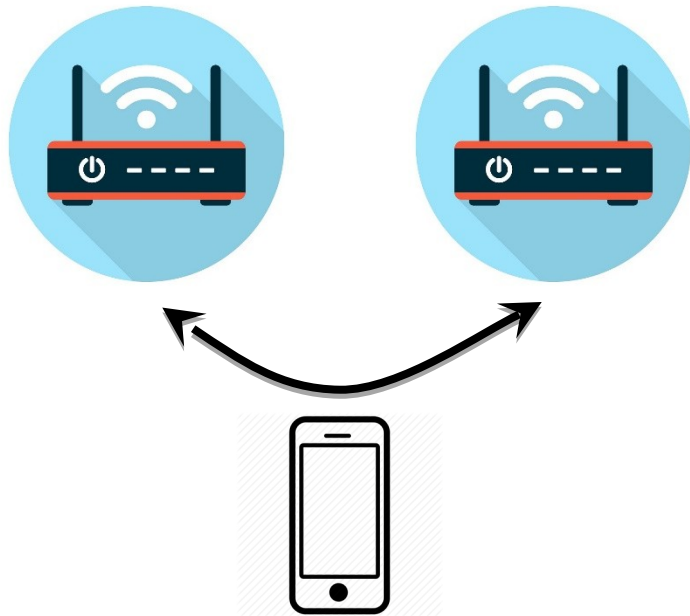
Zero RTT Connection Establishment



1. Repeat connection
2. Never talked to server before

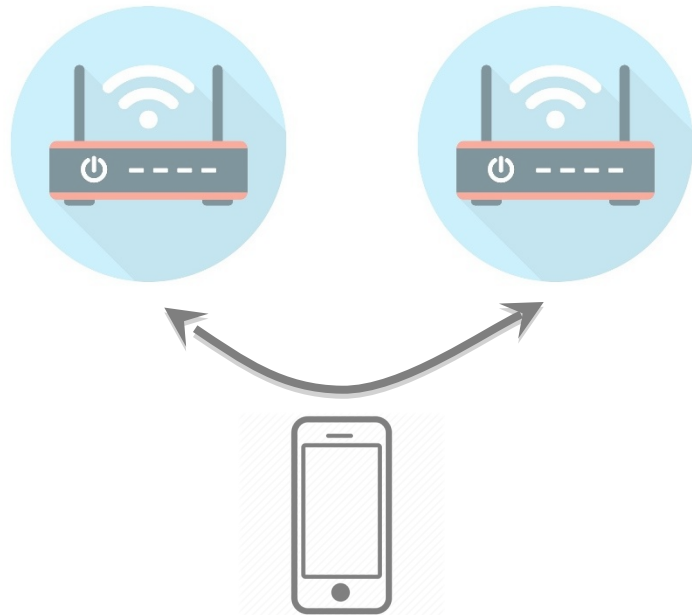
QUIC/HTTP 3.0: Problem of Mobility

- What happens to IP addresses and HTTP sessions when a user moves between wifi APs?

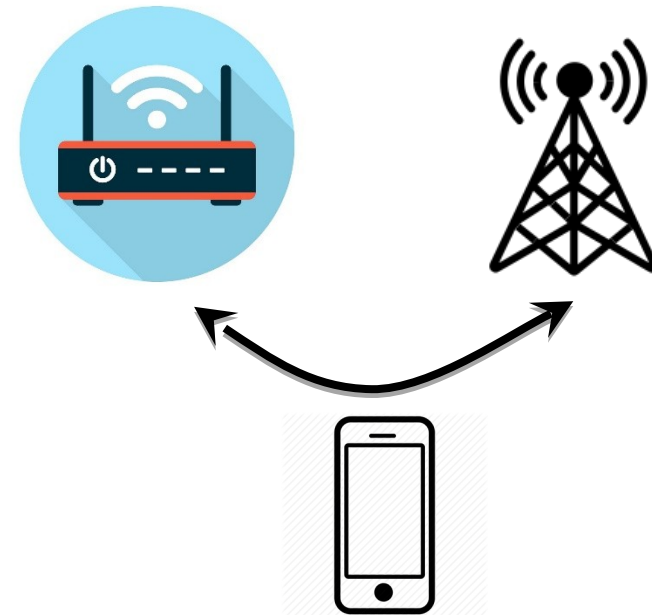


QUIC/HTTP 3.0: Problem of Mobility

- What happens to IP addresses and HTTP sessions when a user moves between wifi APs?

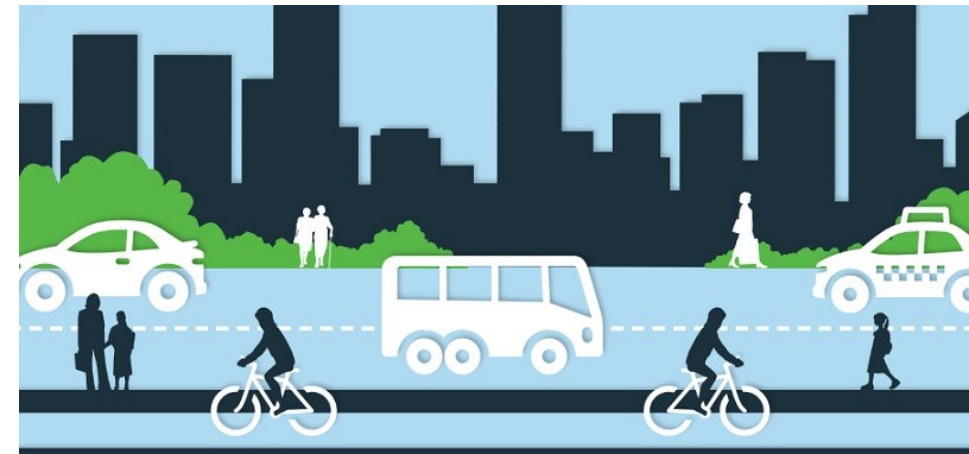


- What happens to IP addresses and HTTP sessions when a user moves between cellular and wifi?



IP Mobility

- Hard problem: IP addresses are supposed to identify nodes in the network but change as nodes move around.
- Proposed solutions:
 - IP Anchor: Place a server at an IP and tunnel traffic to user.
 - DNS Anchor: Have DNS server which rapidly updates as user moves between IP addresses
 - All try to keep some global state constant: IP or DNS Name

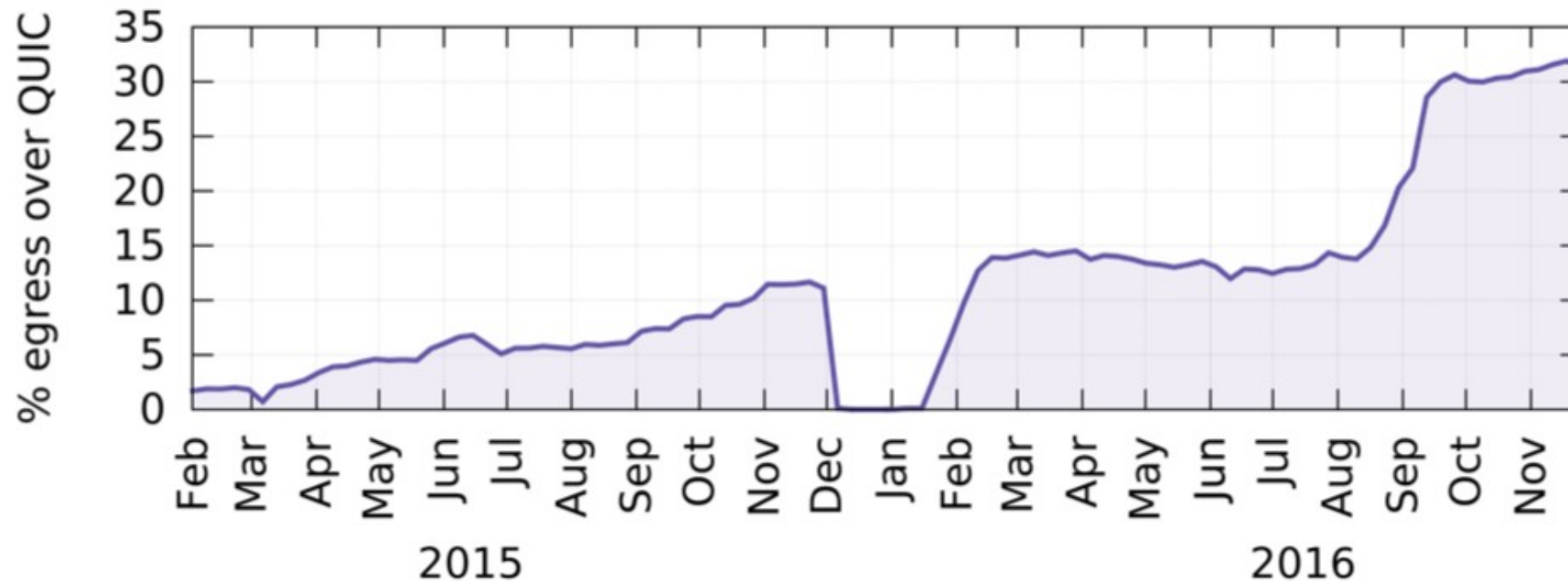


QUIC summary

Makes HTTPS faster, particularly in the tail

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

Deploying at Google was 3+ years of hard work



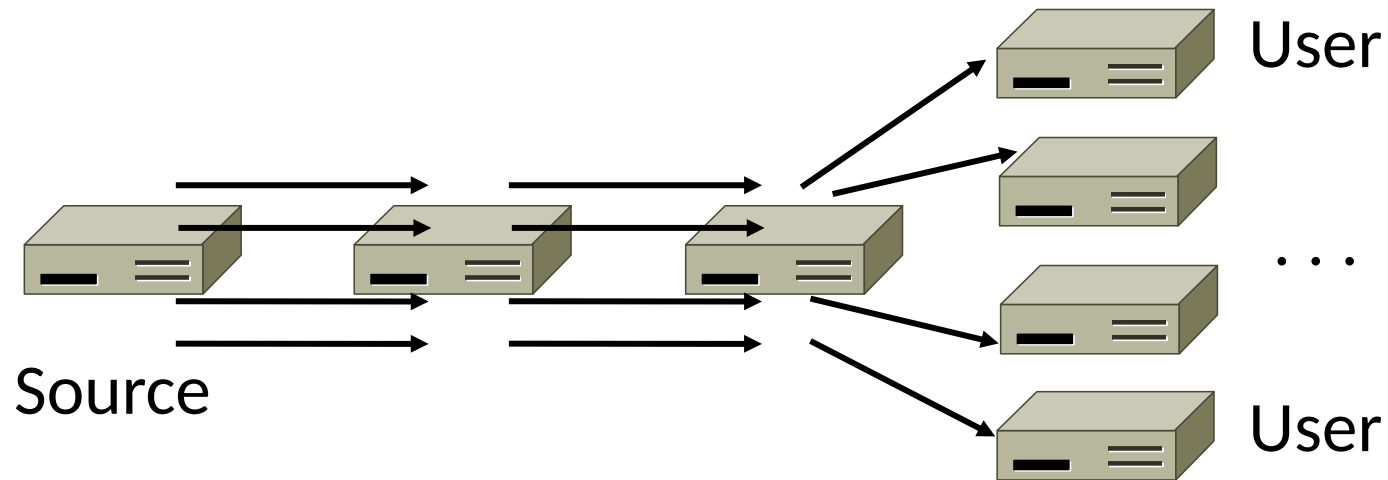
CDNs

Content Delivery Networks

- As the web took off in the 90s, traffic volumes grew and grew. This:
 1. Concentrated load on popular servers
 2. Led to congested networks and need to provision more bandwidth
 3. Gave a poor user experience
- Idea:
 - Place popular content near clients
 - Helps with all three issues above

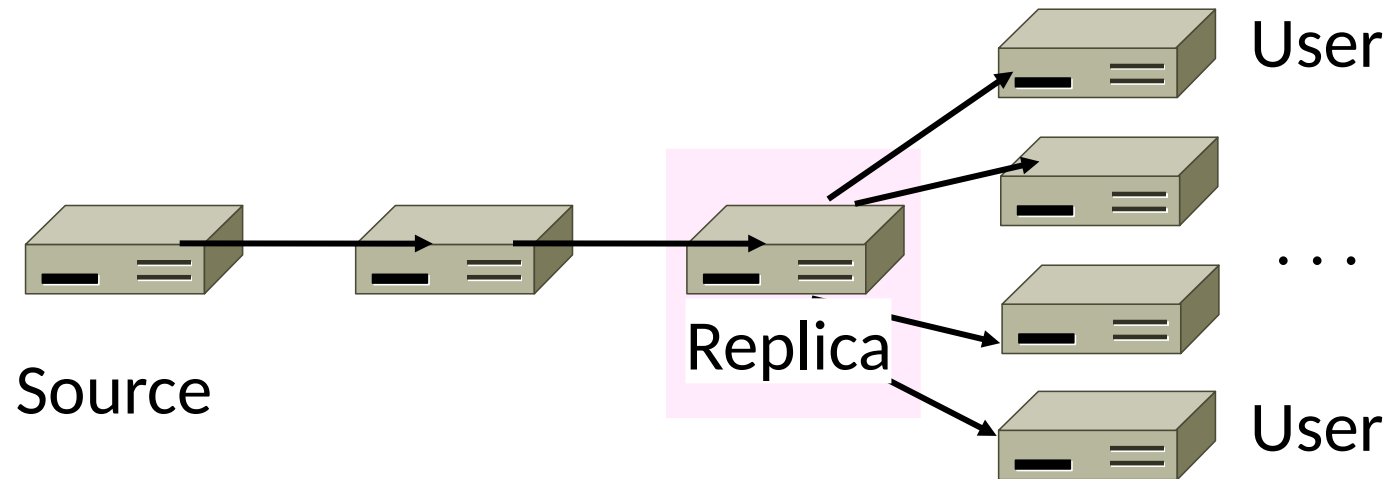
Before CDNs

- Sending content from the source to 4 users takes $4 \times 3 = 12$ “network hops” in the example



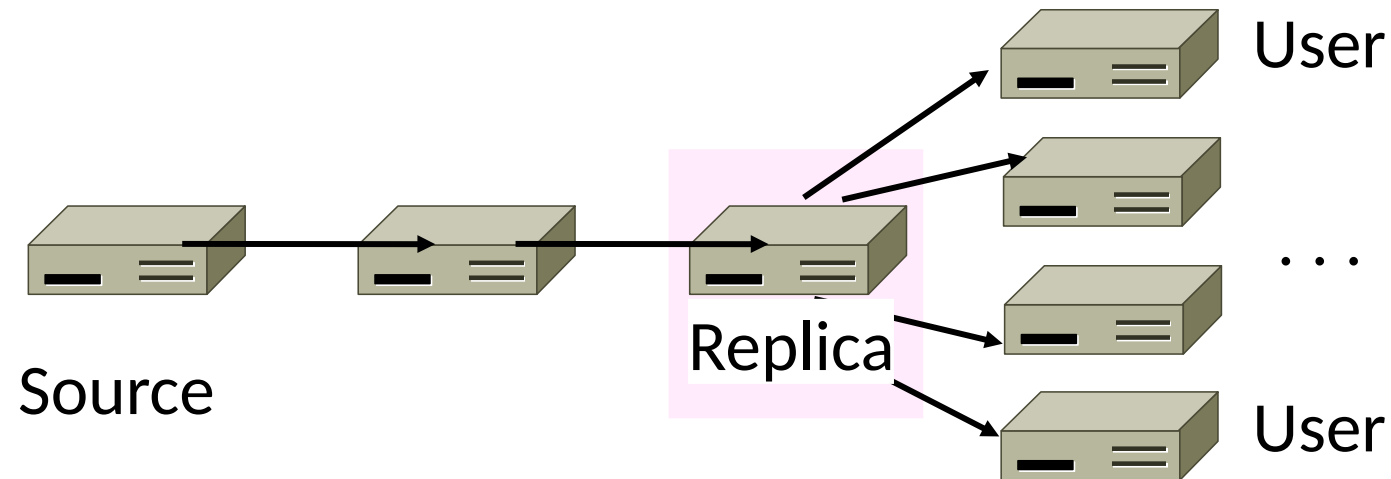
After CDNs

- Sending content via replicas takes only $4 + 2 = 6$ “network hops”



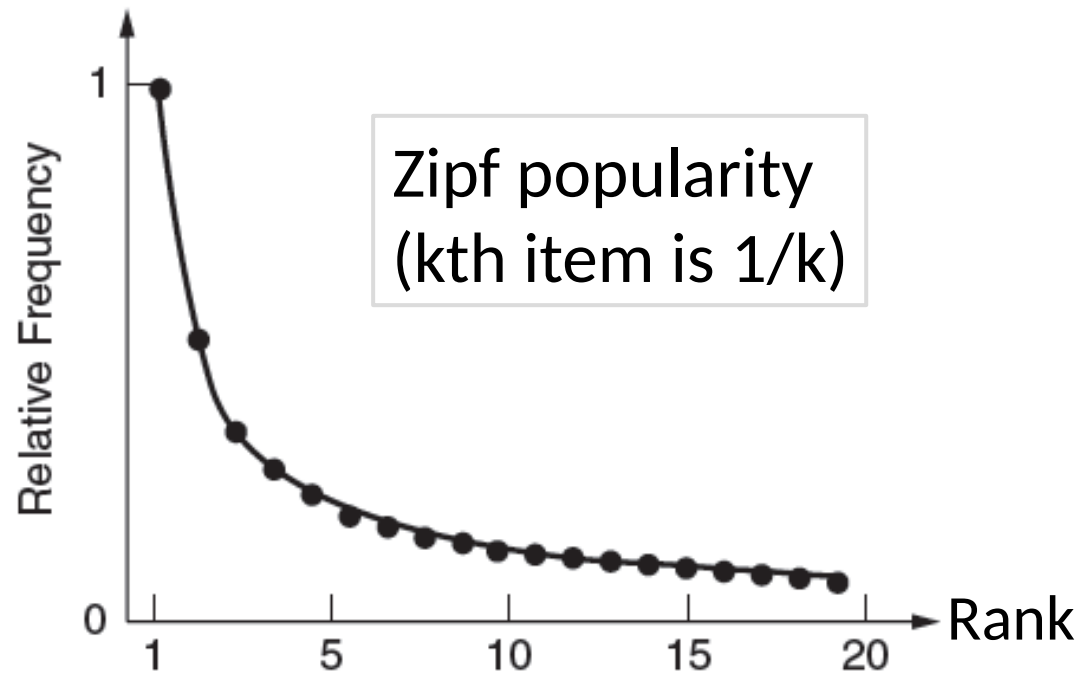
After CDNs (2)

- Benefits assuming popular content:
 - Reduces server, network load
 - Improves user experience



Popularity of Content

- Zipf's Law: few popular items, many unpopular ones; both matter



George Zipf (1902-1950)



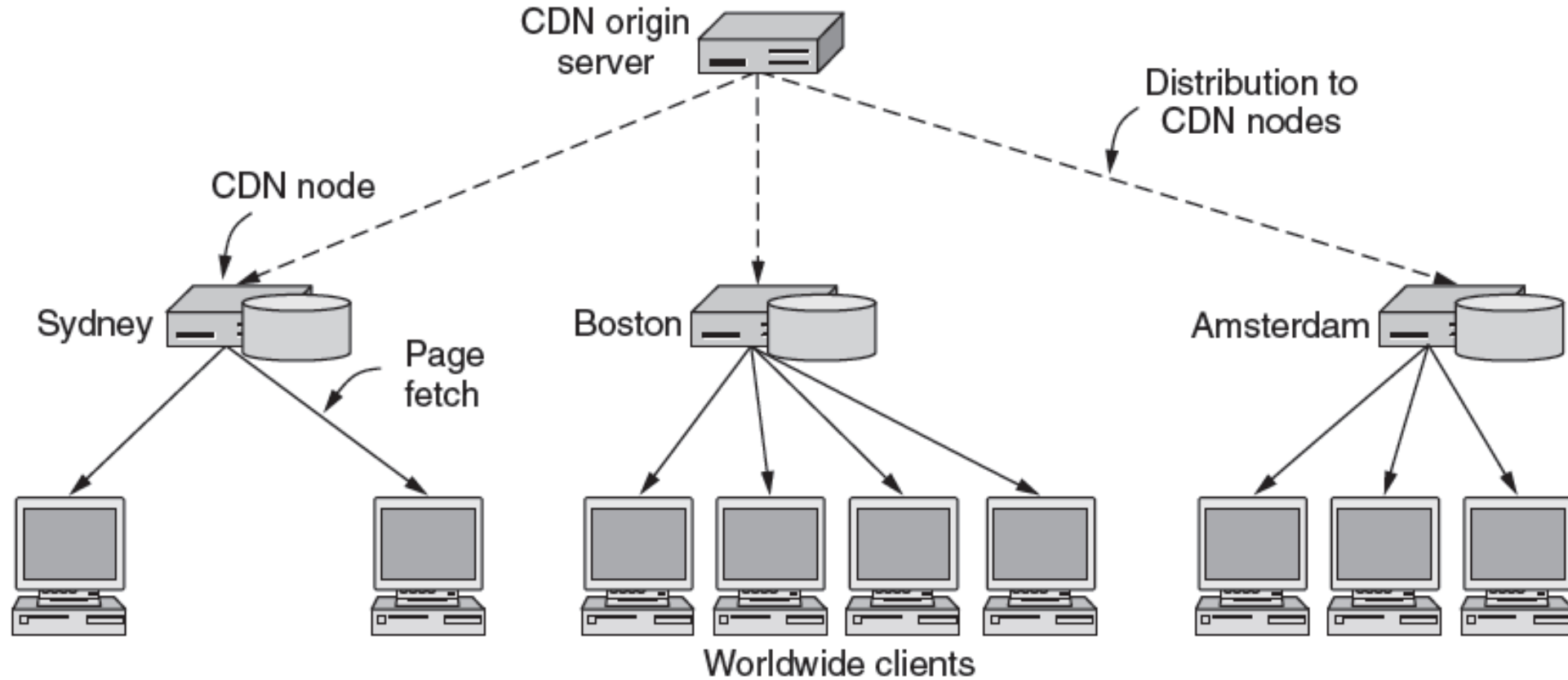
Source: Wikipedia

How to place content near clients?

How to place content near clients?

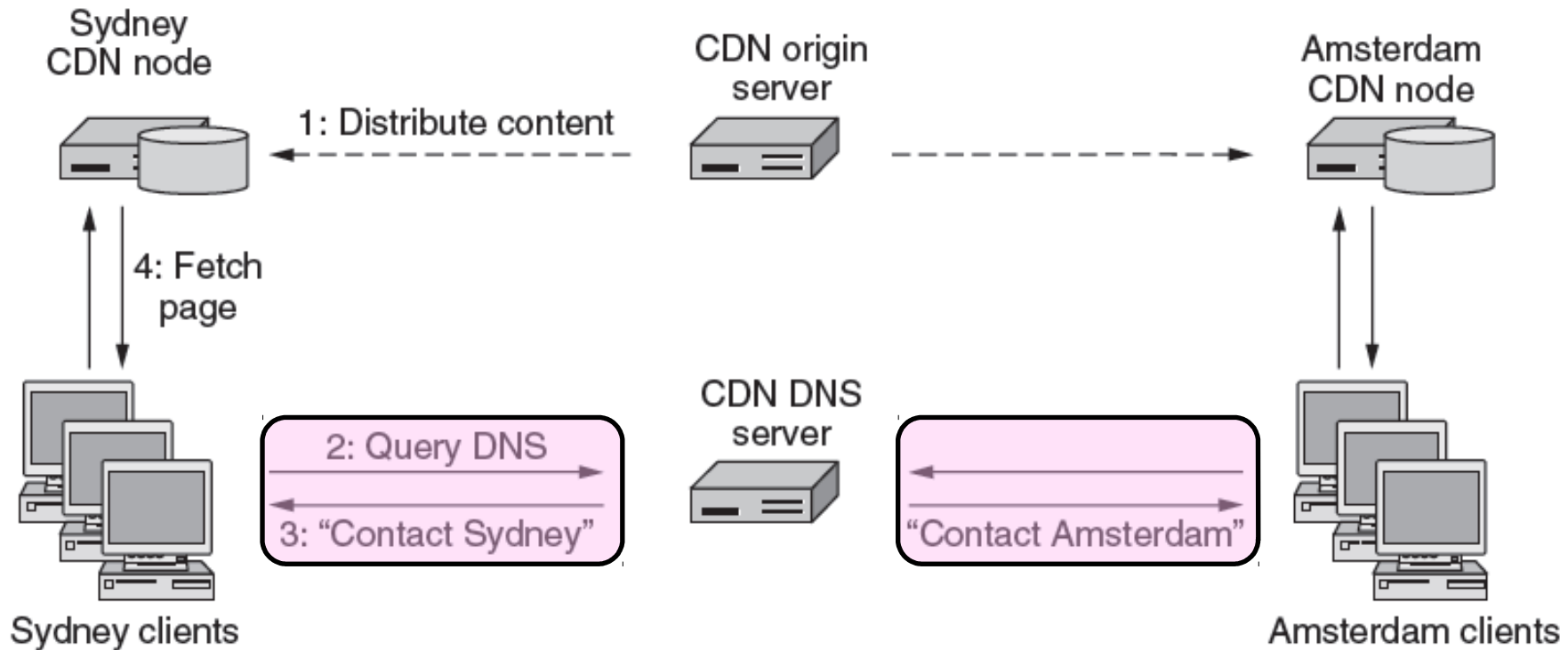
- Use browser and proxy caches
 - Helps, but limited to one client or clients in one organization
- Want to place replicas across the Internet for use by all nearby clients
 - Done by clever use of DNS

Content Delivery Network



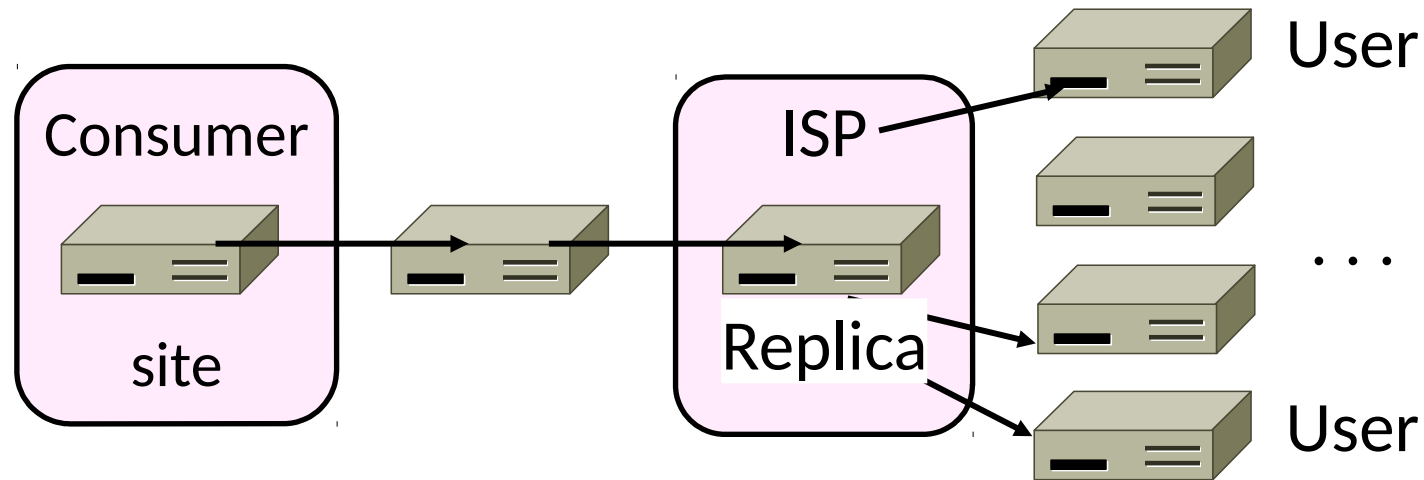
Content Delivery Network (2)

- DNS gives different answers to clients
 - Tell each client the nearest replica (map client IP)



Business Model

- Clever model pioneered by Akamai
 - Placing site replica at an ISP is win-win
 - Improves site experience and reduces ISP bandwidth usage



CDNs - Issues

- Security
 - What about private information?
 - How to cache/forward encrypted content?
 - Basically can't! Big players just share/ship keys.
- Net neutrality
 - I.org, FreeBasics -> Basically CDNs
 - But for reasons of price, not efficiency
 - Who decides who gets to place CDNs?