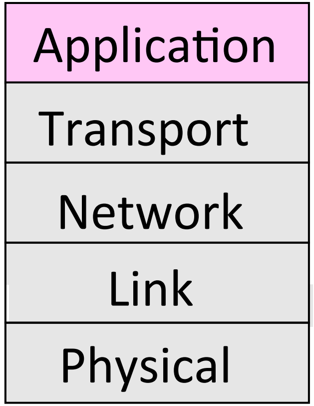
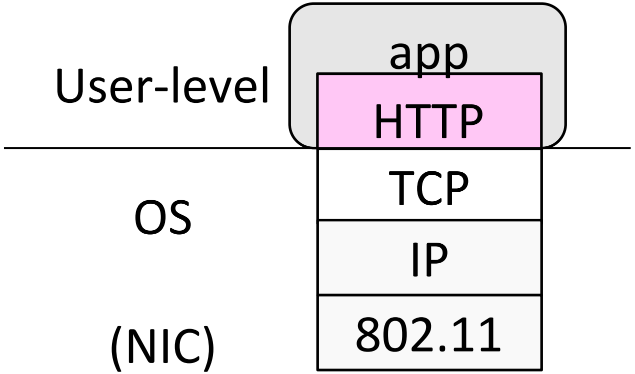
Where we are in the Course

- Starting the Application Layer!
  - Builds distributed "network services" (DNS, Web) on Transport services



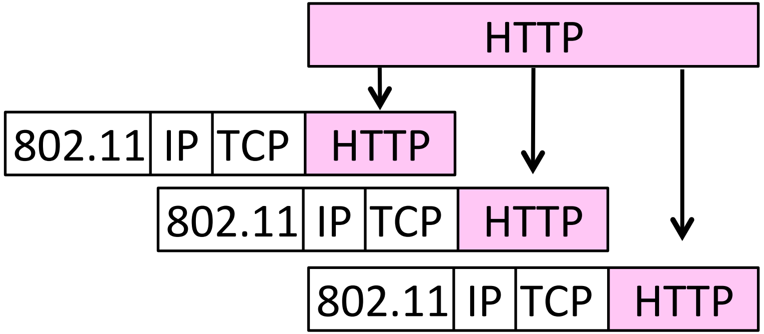
Recall

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



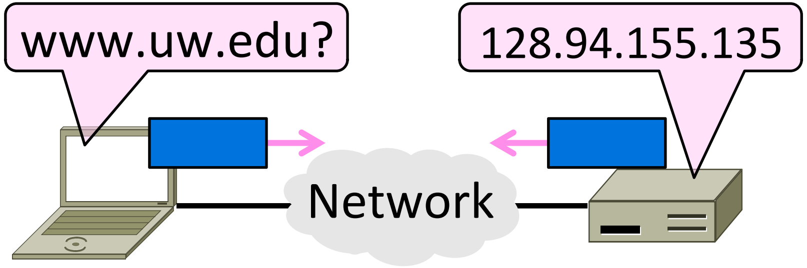
Recall (2)

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



Topic

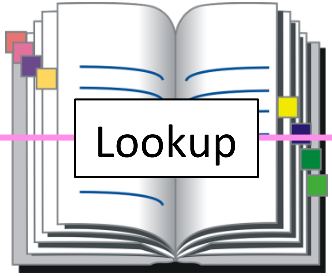
- The DNS (Domain Name System)
  - Human-readable host names, and more
  - Part 1: the distributed namespace



Names and Addresses

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

Name, e.g.
"Andy Tanenbaum,"
or "flits.cs.vu.nl"



Directory

Address, e.g.
"Vrijie Universiteit, Amsterdam"
or IPv4 "130.30.27.38"

Before the DNS – HOSTS.TXT

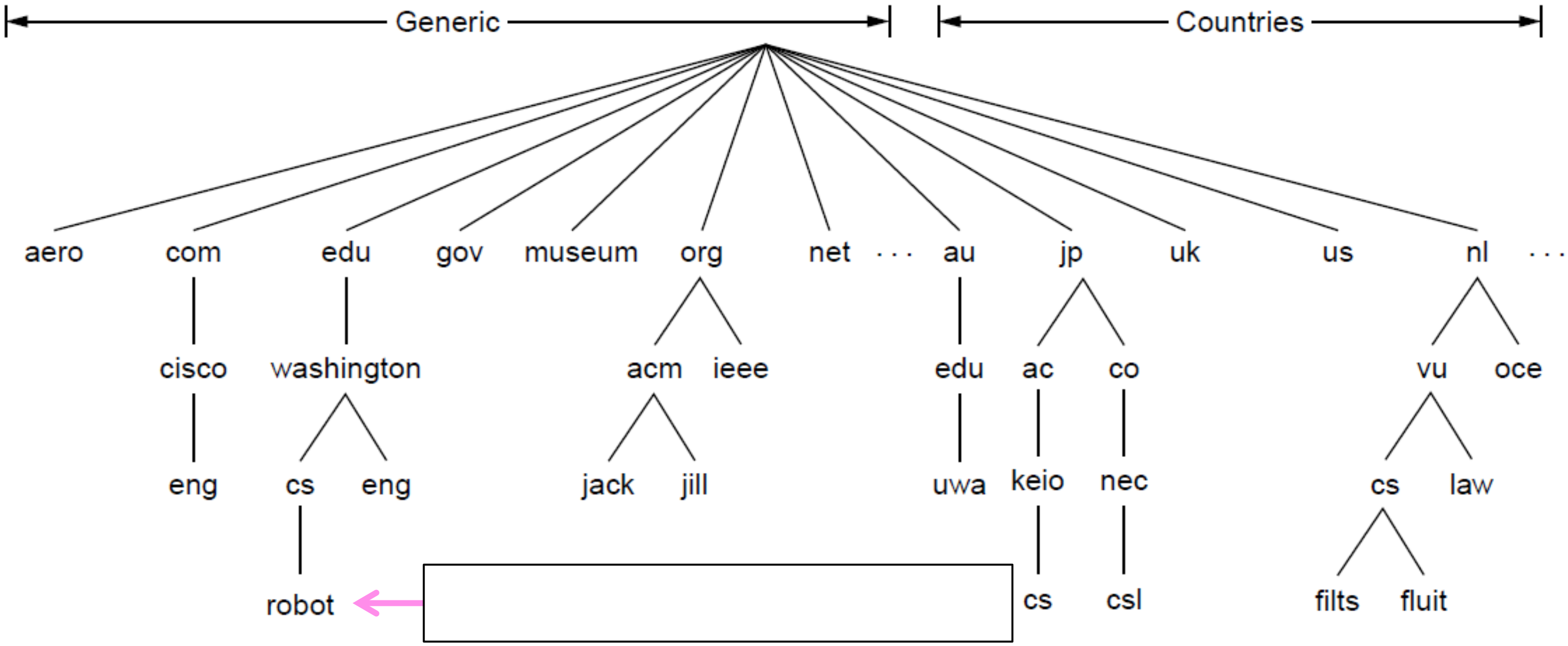
- Directory was a file HOSTS.TXT regularly retrieved for all hosts from a central machine at the NIC (Network Information Center)
- Names were initially flat, became hierarchical (e.g., lcs.mit.edu) ~85
- Neither manageable nor efficient as the ARPANET grew …

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)

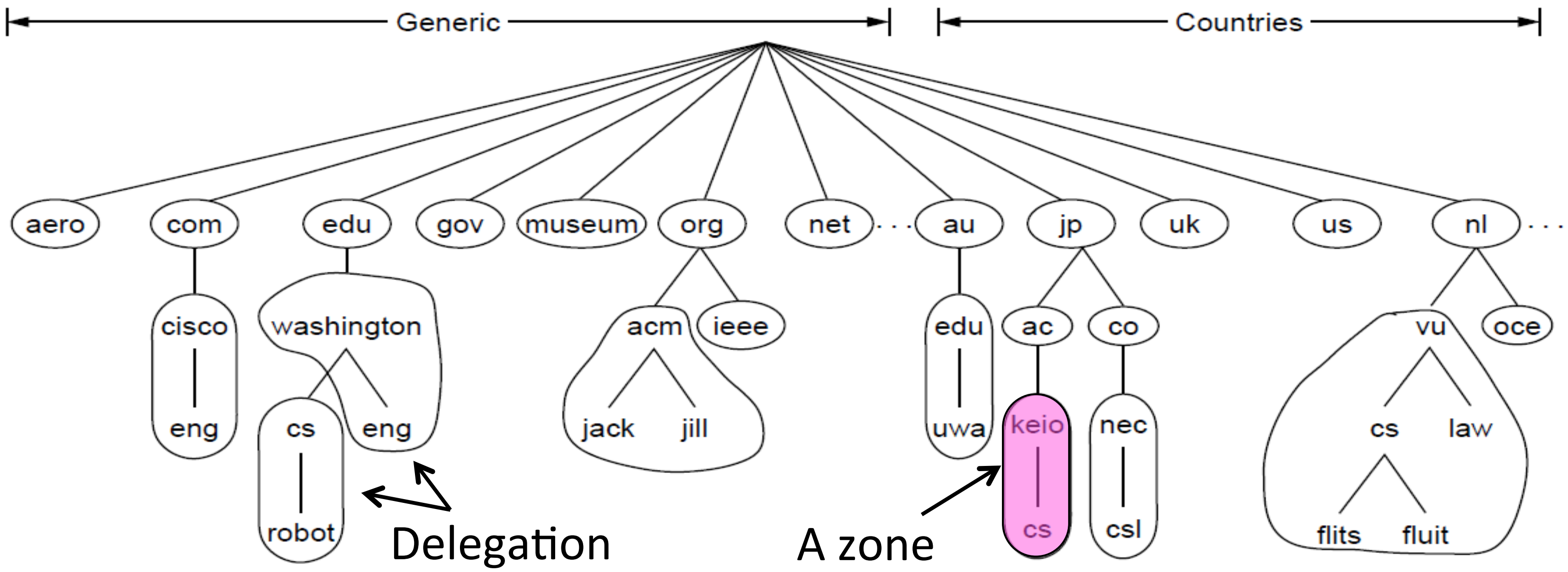


TLDs (Top-Level Domains)

- Run by ICANN (Internet Corp. for Assigned Names and Numbers)
  - Starting in '98; naming is financial, political, and international ☺
- 22+ generic TLDs
  - Initially .com, .edu , .gov., .mil, .org, .net
  - Added .aero, .museum, etc. from '01 through .xxx in '11
  - Different TLDs have different usage policies
- ~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), goo.gl (Greenland)

DNS Zones

- A <u>zone</u> is a contiguous portion of the namespace



DNS Zones (2)

- Zones are the basis for distribution
  - EDU Registrar administers .edu
  - UW administers washington.edu
  - CS&E administers cs.washington.edu

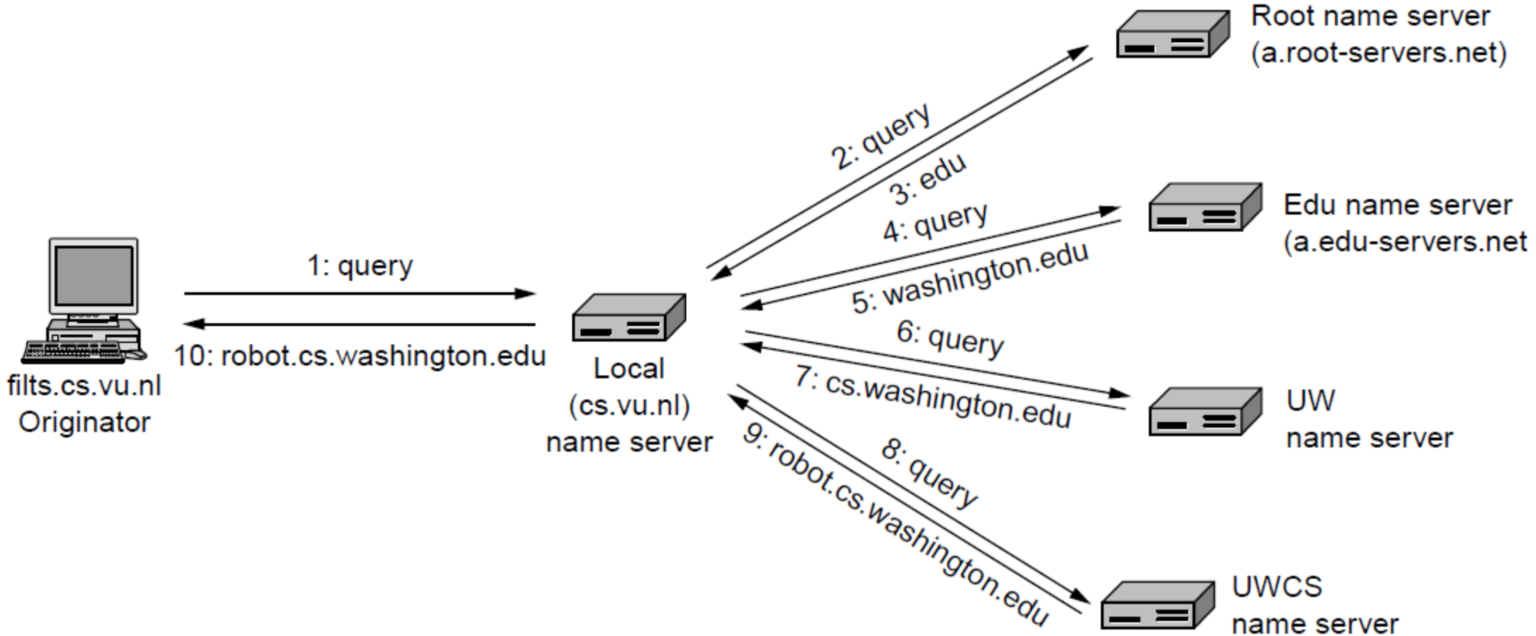
- Each zone has a <u>nameserver</u> to contact for information about it
  - Zone must include contacts for delegations, e.g., .edu knows nameserver for washington.edu

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

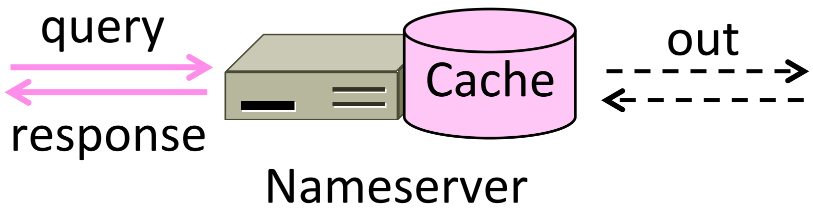
- Recursive query
  - Nameserver completes resolution and returns the final answer
  - E.g., flits → local nameserver
- Iterative query
  - Nameserver returns the answer or who to contact next for the answer
  - E.g., local nameserver → all others

Iterative vs. Recursive Queries (2)

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

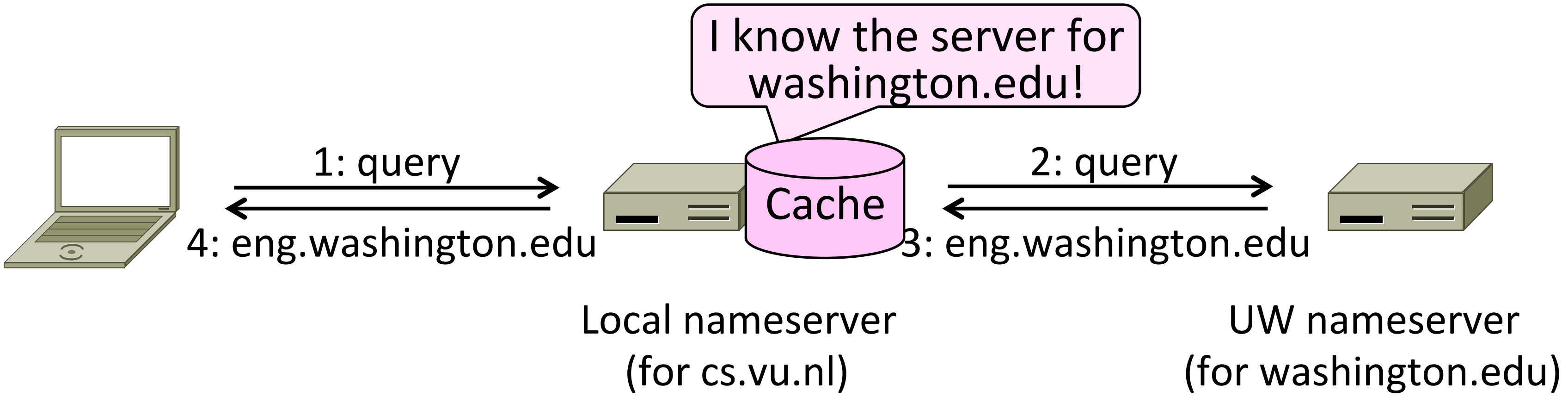
Caching

- Resolution latency should be low
  - Adds delay to web browsing
- Cache query/responses to answer future queries immediately
  - Including partial (iterative) answers
  - Responses carry a TTL for caching



Caching (2)

- flits.cs.vu.nl now resolves eng.washington.edu
  - And previous resolutions cut out most of the process



Local Nameservers

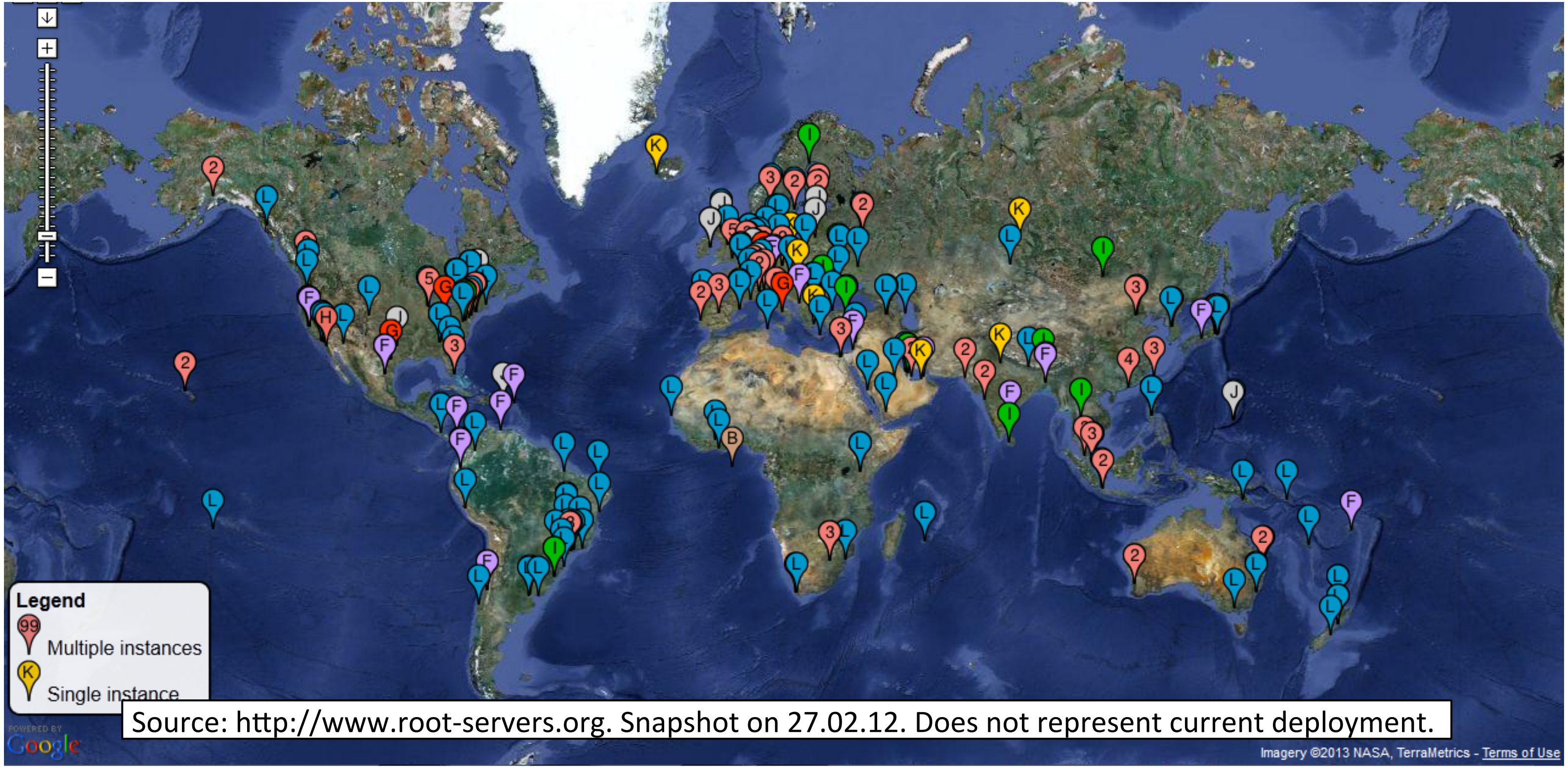
- Local nameservers typically run by IT (enterprise, ISP)
  - But may be your host or AP
  - Or alternatives e.g., Google public DNS
- Clients need to be able to contact their local nameservers
  - Typically configured via DHCP

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)

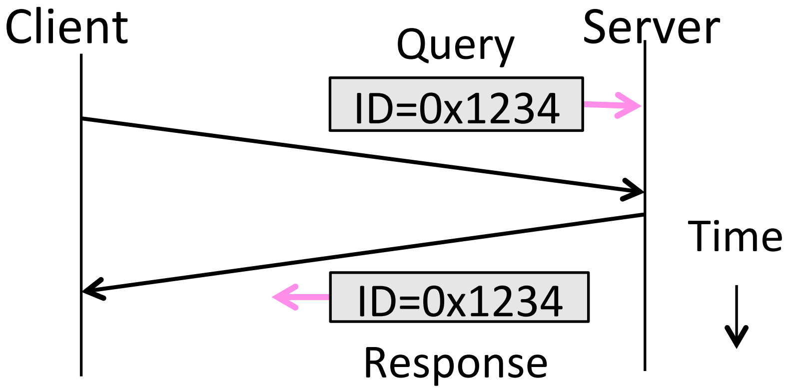
- There are >250 distributed server instances
  - Highly reachable, reliable service
  - Most servers are reached by IP anycast (Multiple locations advertise same IP! Routes take client to the closest one. See §5.2.9)
  - Servers are IPv4 and IPv6 reachable

Root Server Deployment



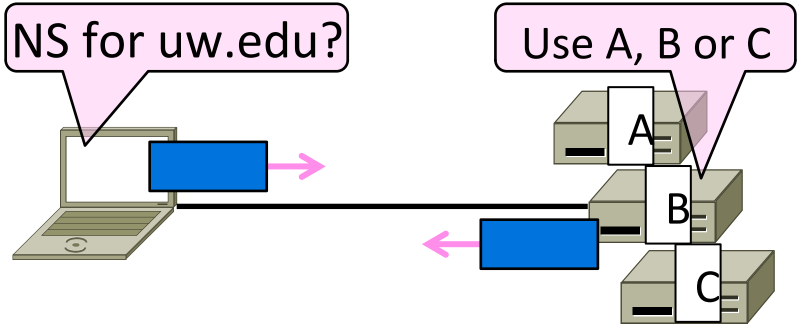
DNS Protocol

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



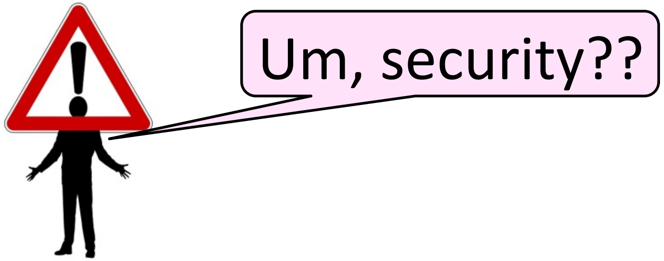
DNS Protocol (2)

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



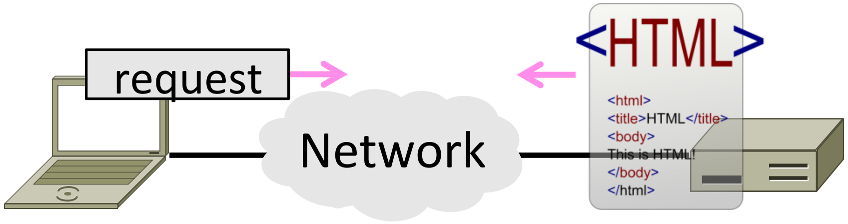
DNS Protocol (3)

- Security is a major issue
  - Compromise redirects to wrong site!
  - Not part of initial protocols ..
- DNSSEC (DNS Security Extensions)
  - Long under development, now partially deployed. We'll look at it later



Topic

- Performance of HTTP
  - Parallel and persistent connections
  - Caching for content reuse

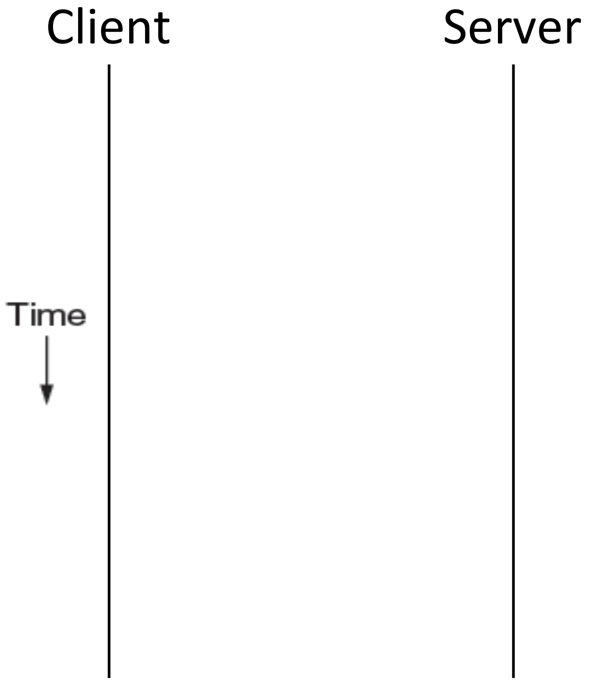


PLT (Page Load Time)

- PLT is 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

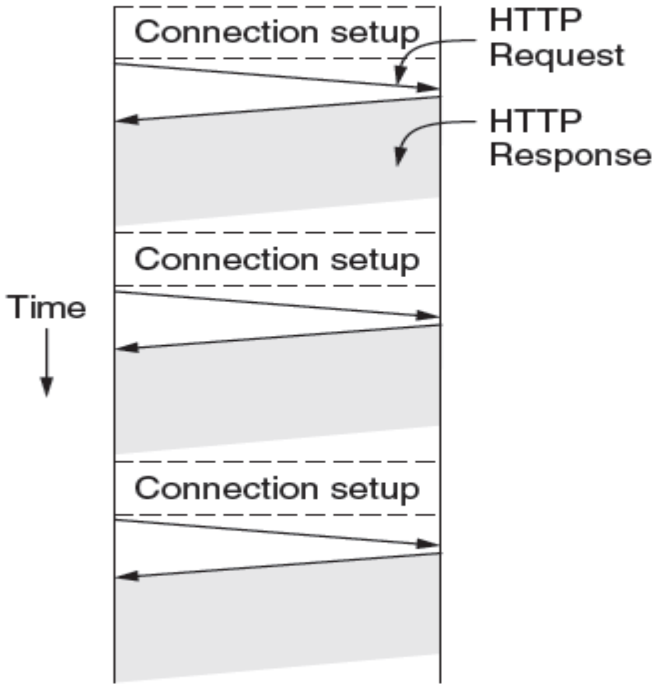
Early Performance

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



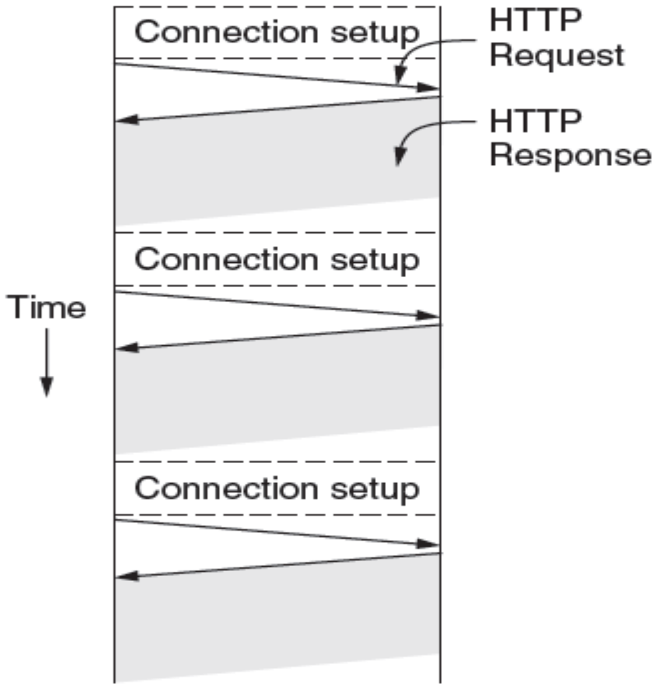
Early Performance (2)

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



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 available bandwidth
3. Change HTTP to avoid repeated transfers of the same content
   – Caching, and proxies

This time

4. Relocate content to reduce RTT
   – CDNs [later]

Later

Parallel Connections

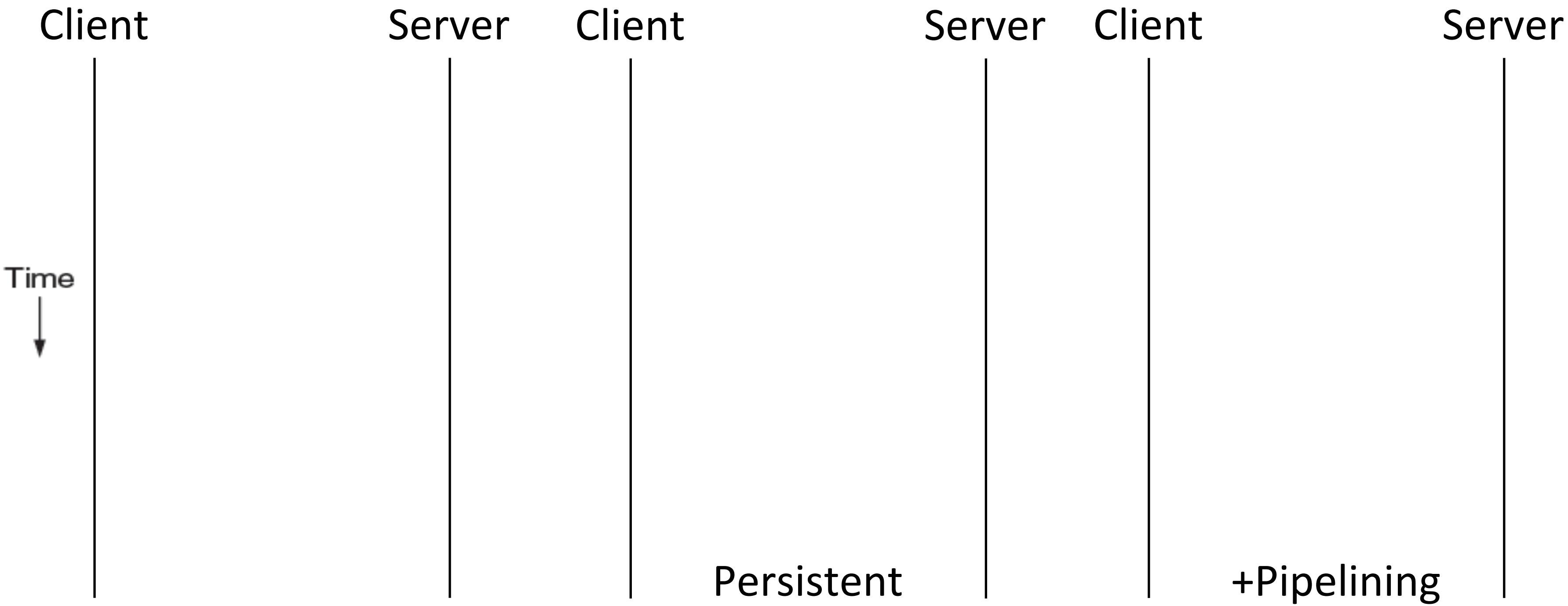
- One simple way to reduce PLT
  - Browser runs multiple (8, say) HTTP instances in parallel
  - Server is unchanged; already handled concurrent requests for many clients
- How does this help?
  - Single HTTP wasn't using network much …
  - So parallel connections aren't slowed much
  - Pulls in completion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

Persistent Connections

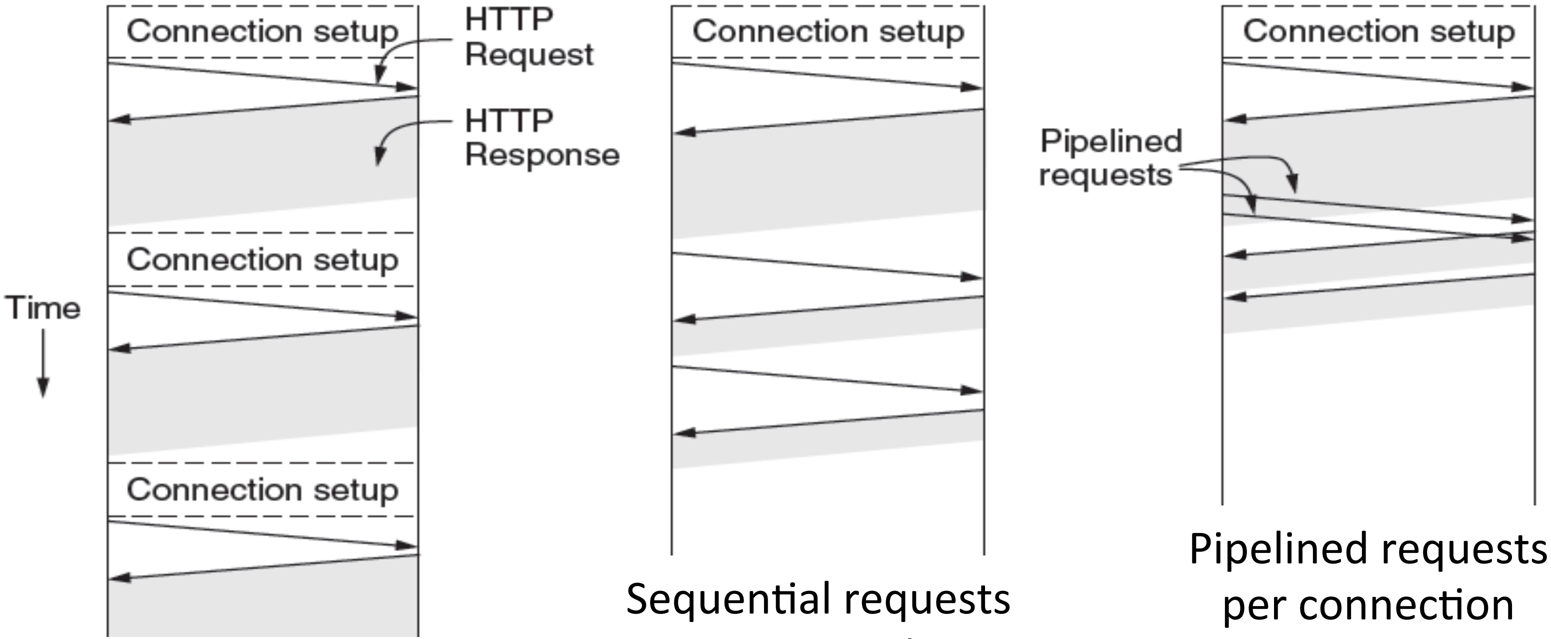
- Parallel connections compete with each other for network resources
  - 1 parallel client ≈ 8 sequential clients?
  - Exacerbates network bursts, and loss

- 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

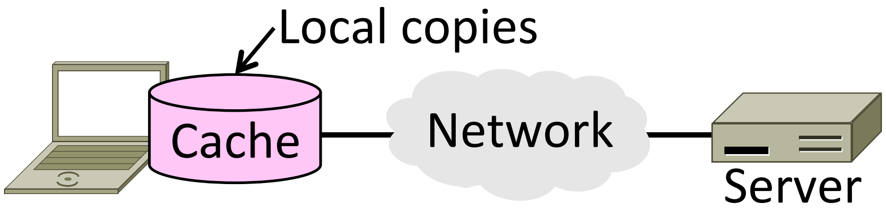
Pipelined requests per connection

Persistent Connections (4)

- Widely used as part of HTTP/1.1
  - Supports optional pipelining
  - PLT benefits depending on page structure, but easy on network
- Issues with persistent connections
  - How long to keep TCP connection?
  - Can it be slower? (Yes. But why?)

Web Caching

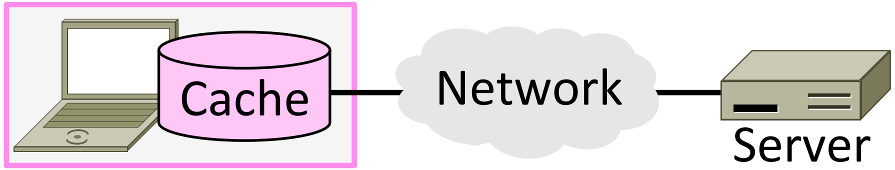
- Users often revisit web pages
  - Big win from reusing local copy!
  - This is caching



- Key question:
  - When is it OK to reuse local copy?

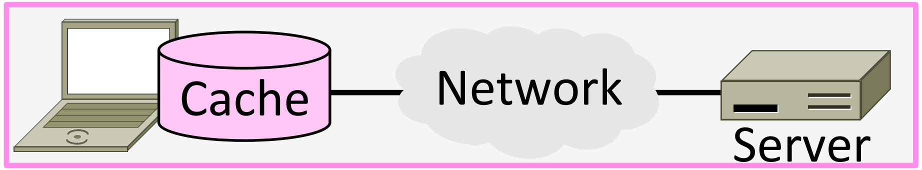
Web Caching (2)

- Locally determine copy is still valid
  - Based on expiry information such as "Expires" header from server
  - Or use a heuristic to guess (cacheable, freshly valid, not modified recently)
  - Content is then available right away



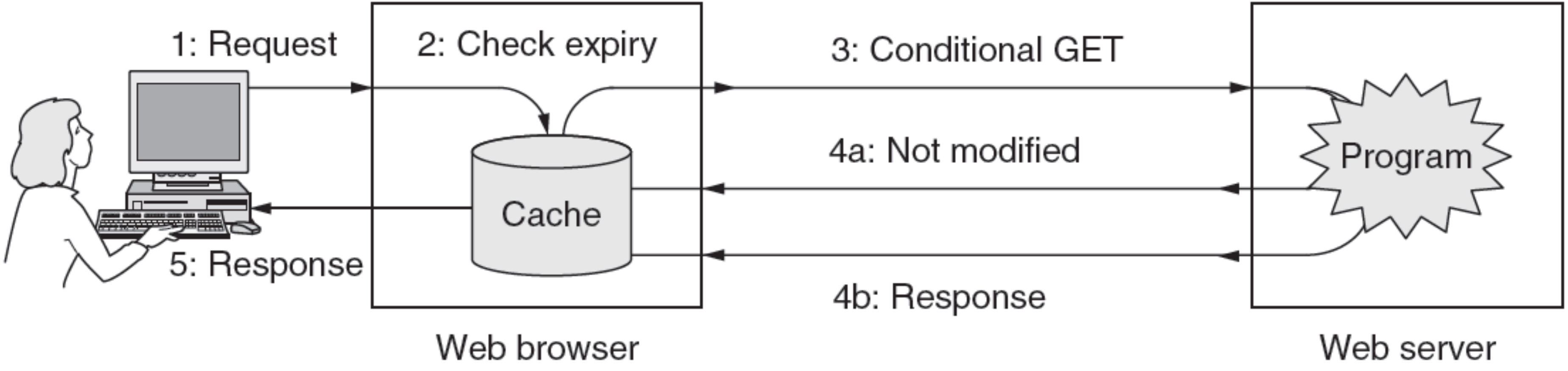
Web Caching (3)

- Revalidate copy with server
  - Based on timestamp of copy such as "Last-Modified" header from server
  - Or based on content of copy such as "Etag" header from server
  - Content is available after 1 RTT



Web Caching (4)

- Putting the pieces together:



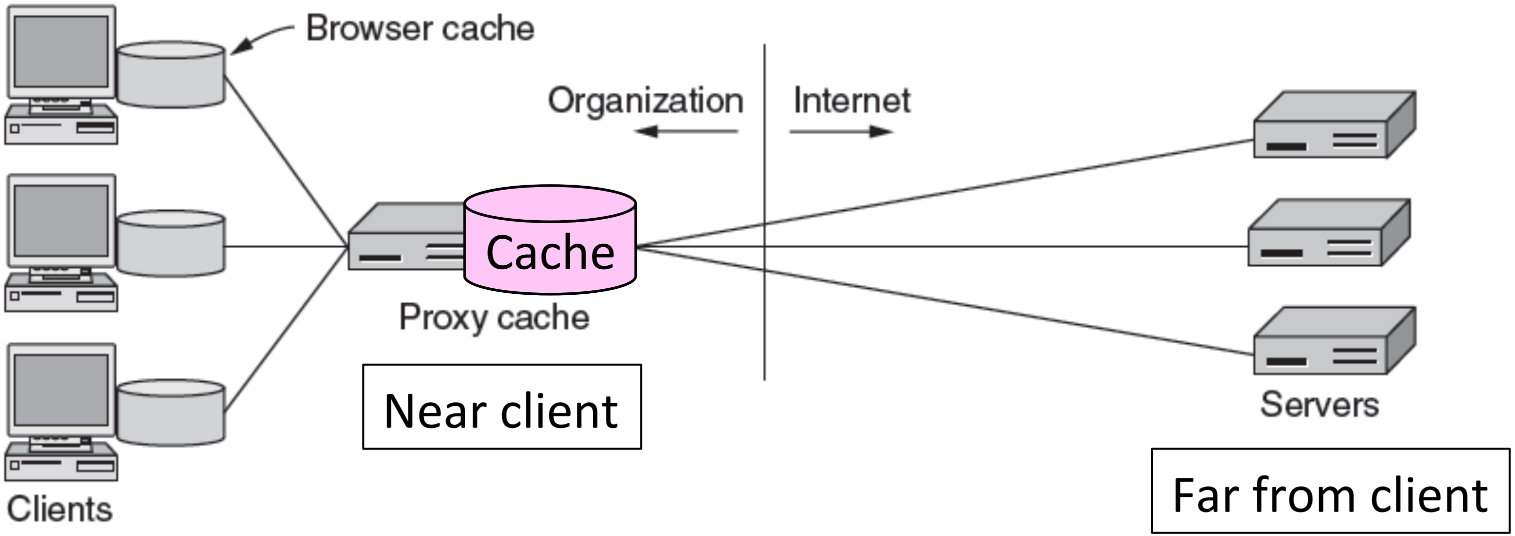
Web Proxies

- Place intermediary between pool of clients and external web servers
  - Benefits for clients include greater caching and security checking
  - Organizational access policies too!

- Proxy caching
  - Clients benefit from a larger, shared cache
  - Benefits limited by secure and dynamic content, as well as "long tail"

Web Proxies (2)

- Clients contact proxy; proxy contacts server



mod_pagespeed

- Observation:
  - The way pages are written affects how quickly they load
  - Many books on best practices for page authors and developers

- Key idea:
  - Have server re-write (compile) pages to help them load quickly!
  - mod_pagespeed is an example

mod_pagespeed (2)

- Apache server extension
  - Software installed with web server
  - Rewrites pages “on the fly” with rules based on best practices

- Example rewrite rules:
  - Minify Javascript
  - Flatten multi-level CSS files
  - Resize images for client
  - And much more (100s of specific rules)