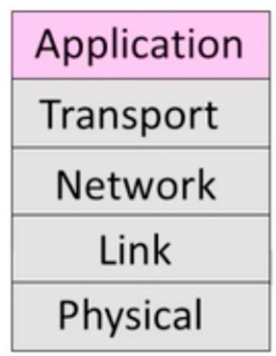
Computer Network Design

Application Layer

Yalda Edalat – Spring 23

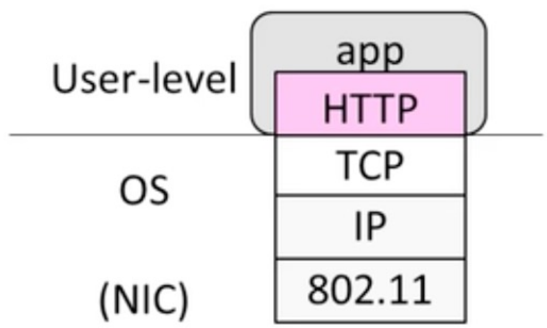
Where we are in the Course

- Starting the application layer!
  - Builds distributed "network services" (DNS, Web) on transport services



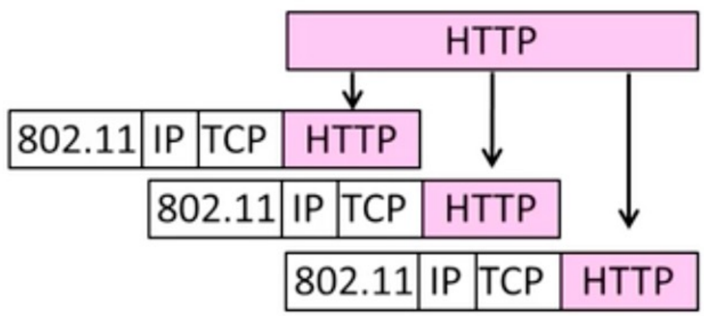
Recall

- Application layer protocols are often part of an "app"
  - But do not need a GUI, e.g., DNS



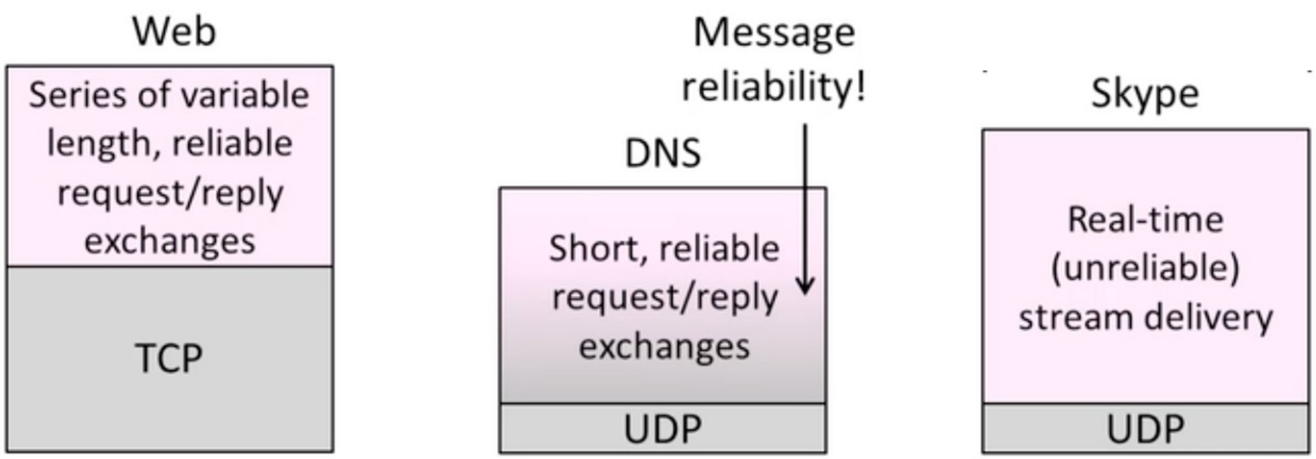
Recall (2)

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

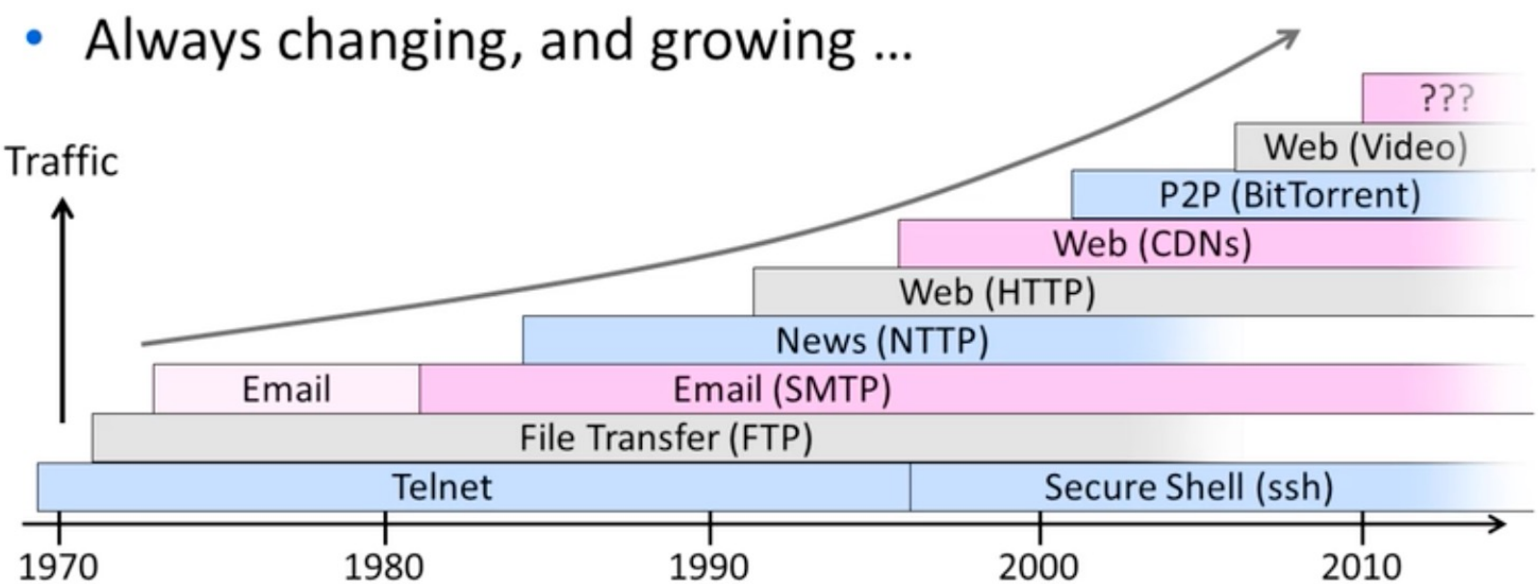


Application Communication Needs

- Vary widely with app; must build on transport services

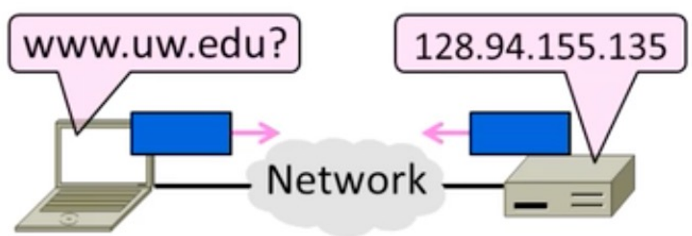


Evolution of Internet Applications



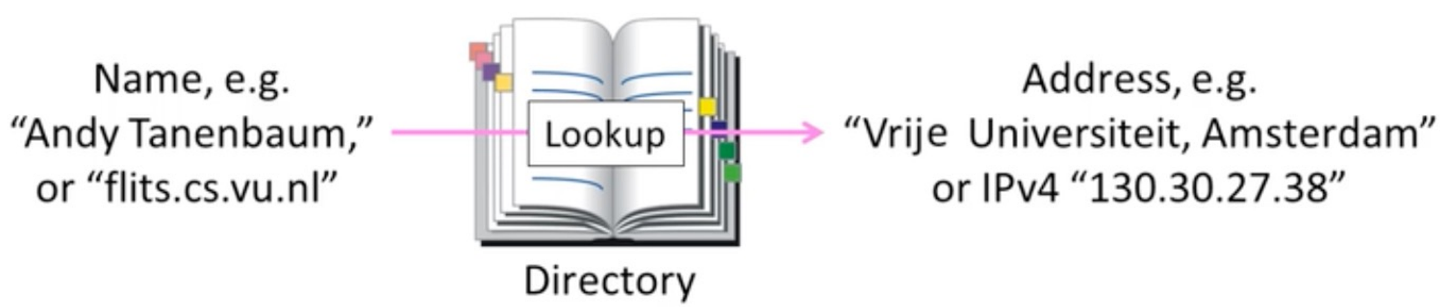
First Application: DNS

- The DNS (Domain Name System)
  - 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 address
- **Resolution** (or lookup) is mapping a name to an address



DNS

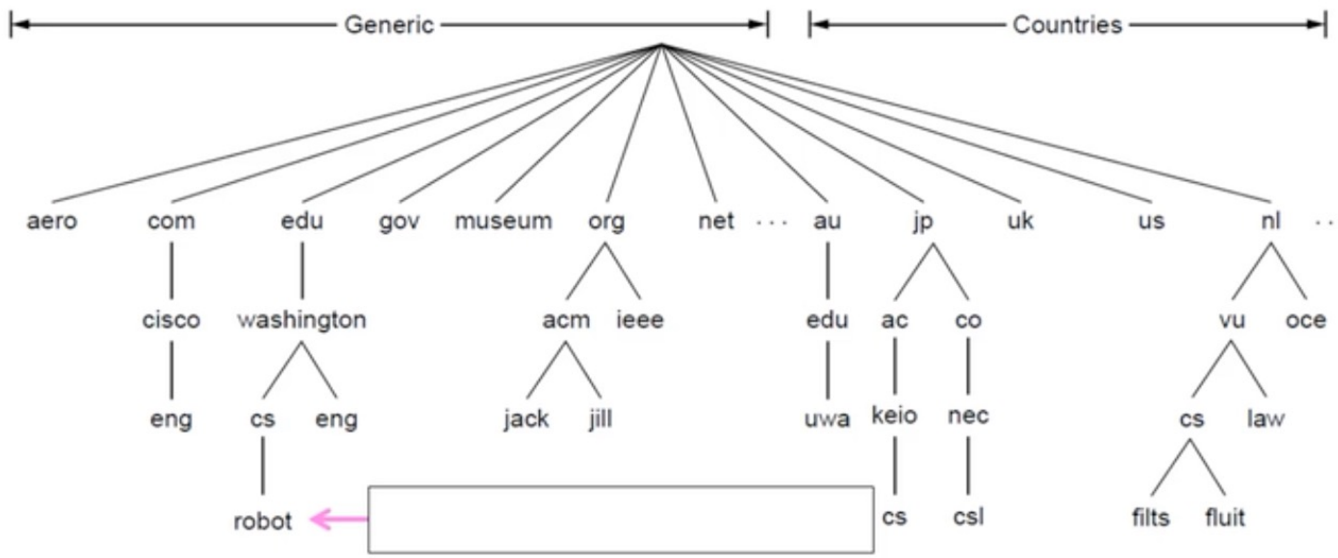
- A naming service to map between host names and their IP addresses
  - www.uwa.edu.au -> 130.95.128.149

- 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

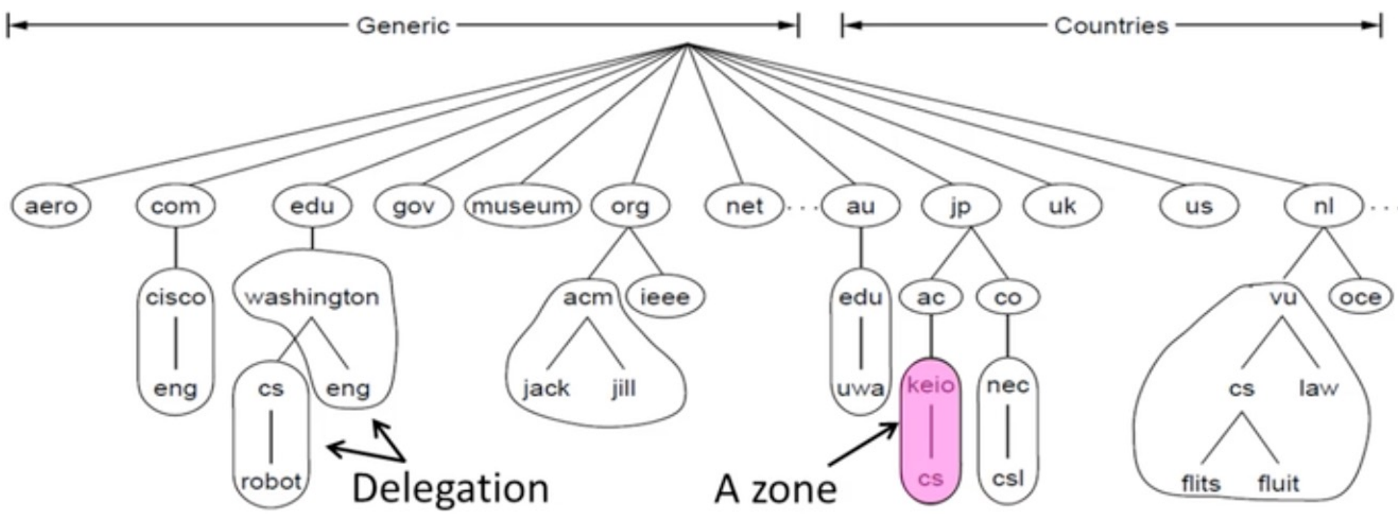
DNS Namespace

- Hierarchical, starting from "." (dot, typically omitted)



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
  - CS&E 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 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)

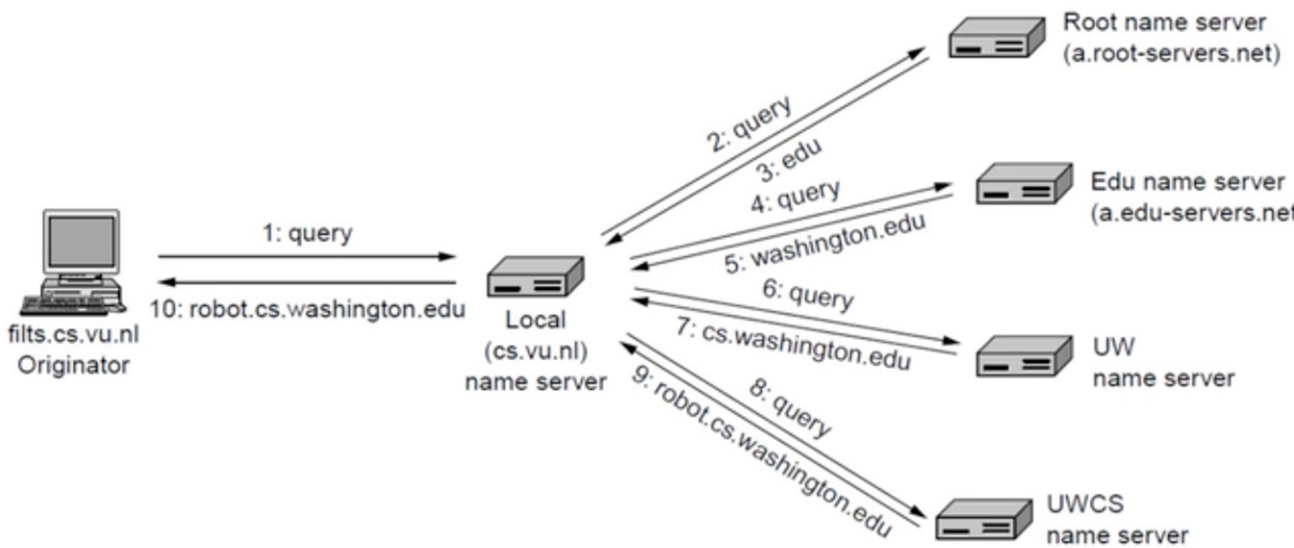
```
; Authoritative data for cs.vu.nl
cs.vu.nl.       86400   IN  SOA     star boss (9527,7200,7200,241920,86400)
cs.vu.nl.       86400   IN  MX      1 zephyr
cs.vu.nl.       86400   IN  MX      2 top
cs.vu.nl.       86400   IN  NS      star                Name server

star            86400   IN  A       130.37.56.205
zephyr          86400   IN  A       130.37.20.10
top             86400   IN  A       130.37.20.11        IP addresses
www             86400   IN  CNAME   star.cs.vu.nl       of computers
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           Mail gateways
                        IN  MX      2 zephyr

little-sister           IN  A       130.37.62.23

laserjet                IN  A       192.31.231.216
```

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 and example first …

DNS Resolution (2)

- Flits.cs.vu.nl resolves robot.cs.Washington.edu

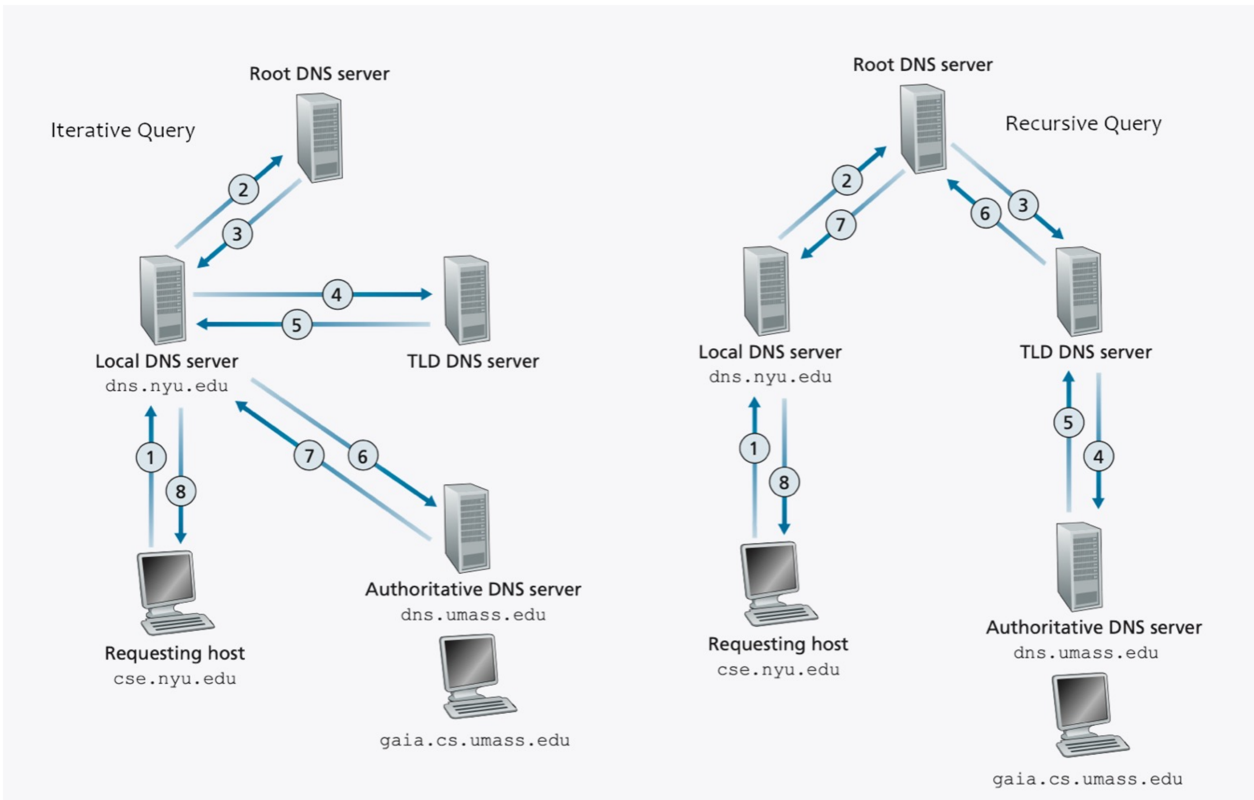


Iterative vs. Recursive Queries

- Recursive query:
  - Nameserver completes resolution and returns the final answer
  - E.g., flits -> local nameserver

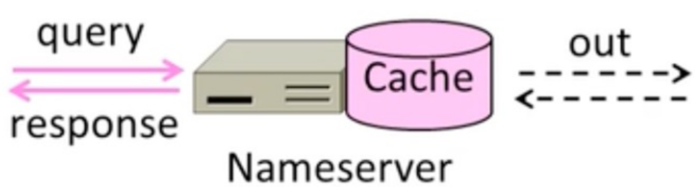
- Iterative query:
  - Nameserver returns the answer or who to contact next for the answer
  - E.g., local nameserver -> all others

Iterative vs. Recursive Queries (2)



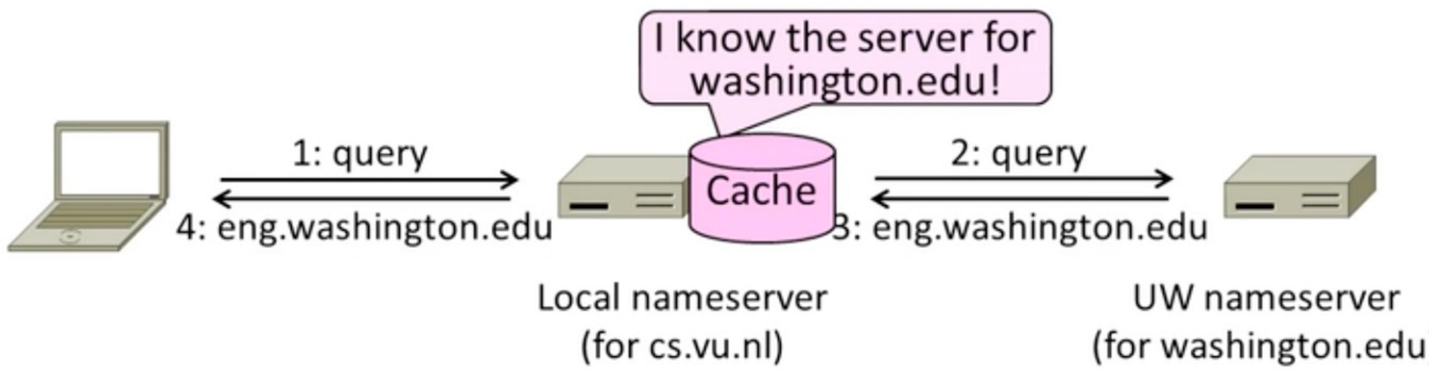
Caching

- Resolution latency should be low
  - Adds delay to web browsing
- Cache query/response 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

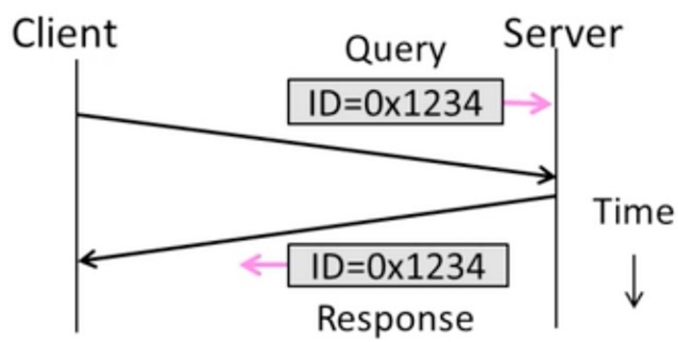


Local Nameservers

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

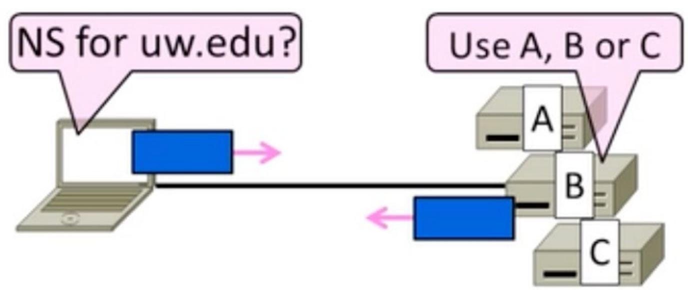
DNS Protocol

- Query and response messages
  - Built on UDP messages, port 53
  - 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



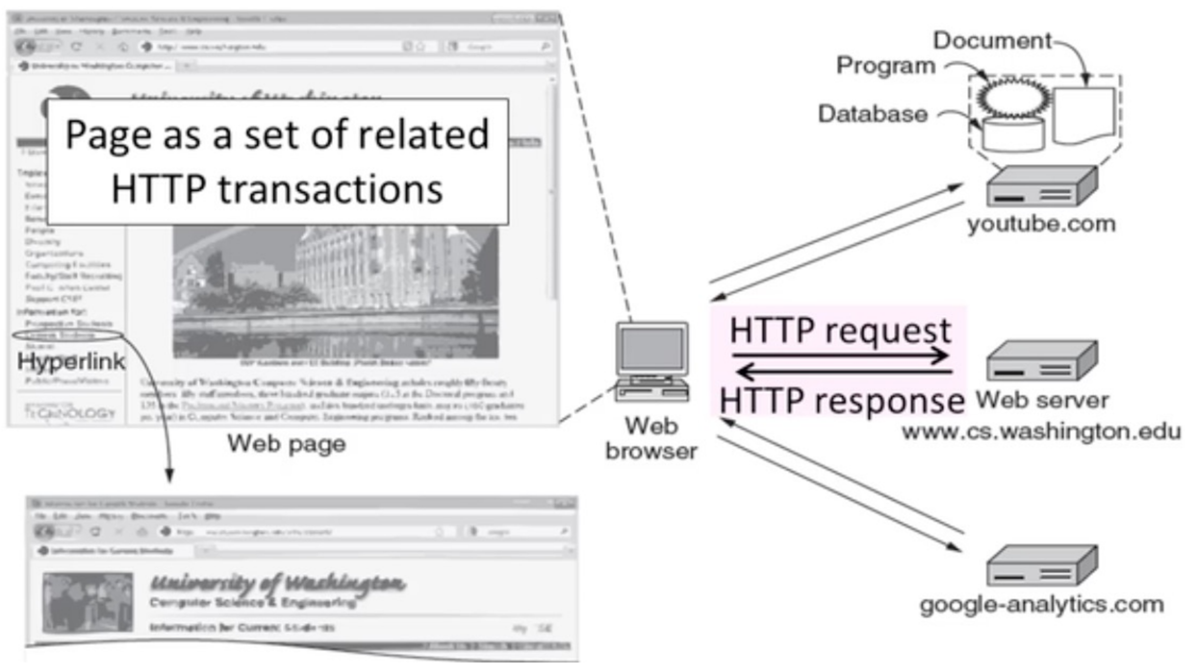
DNS Protocol (3)

- 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.

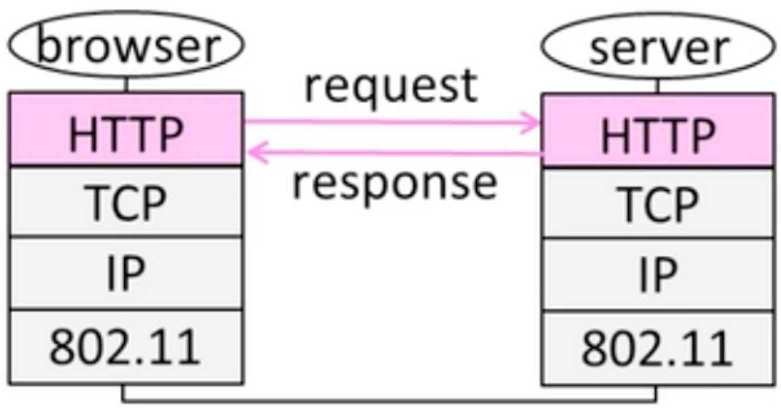
HyperText Transfer Protocol (HTTP)

- Basic for fetching Web pages



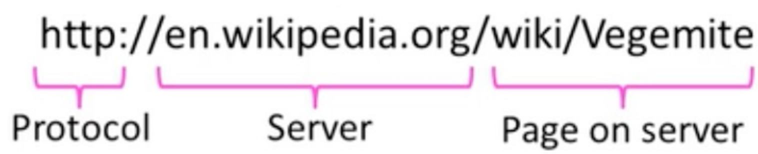
HTTP Context

- 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

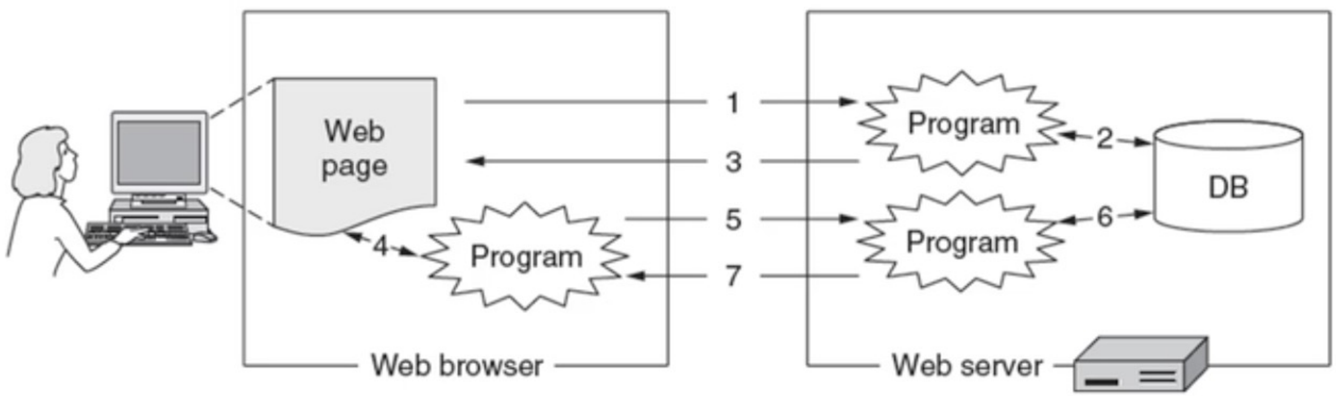
- Start with the page URL:



- 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

Static vs. Dynamic Web Pages

- Static web page is a file contents, e.g., image
- Dynamic web page is the result of program execution
  - Javascript on client, PHP on server, or both



HTTP Protocol

- Originally a simple protocol, with many options added over time
  - Text-based commands, headers
- 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
	DELETE	Remove the Web page
	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
Yes! →	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

HTTP Protocol (4)

- Many header fields specify capabilities and content
  - E.g., Content-Type: text/html, Cookie: lect=8-4-http

Function	Example Headers
Browser capabilities (client → server)	User-Agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language
Caching related (mixed directions)	If-Modified-Since, If-None-Match, Date, Last-Modified, Expires, Cache-Control, ETag
Browser context (client → server)	Cookie, Referer, Authorization, Host
Content delivery (server → client)	Content-Encoding, Content-Length, Content-Type, Content-Language, Content-Range, Set-Cookie

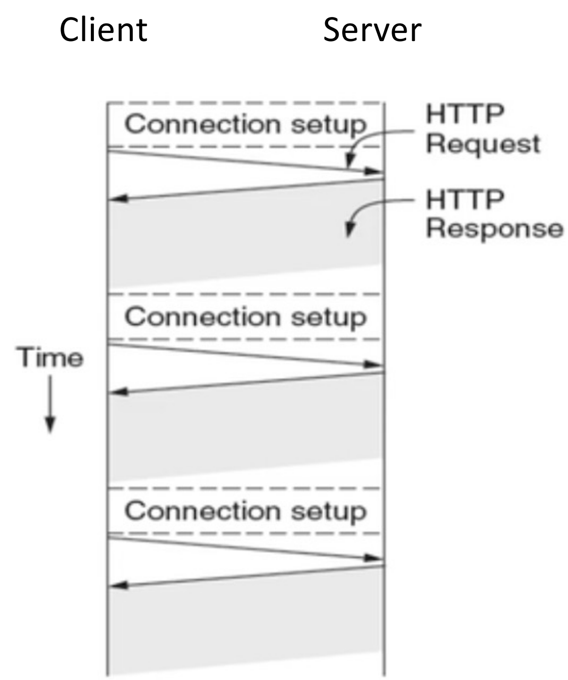
PLT (Page Load Time)

- 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 …
- Many reasons why PLT is larger than necessary
  - Sequential request/responses, even when to different servers
  - Multiple TCP connections setups to 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
4. Move content close to client
   - CDNs

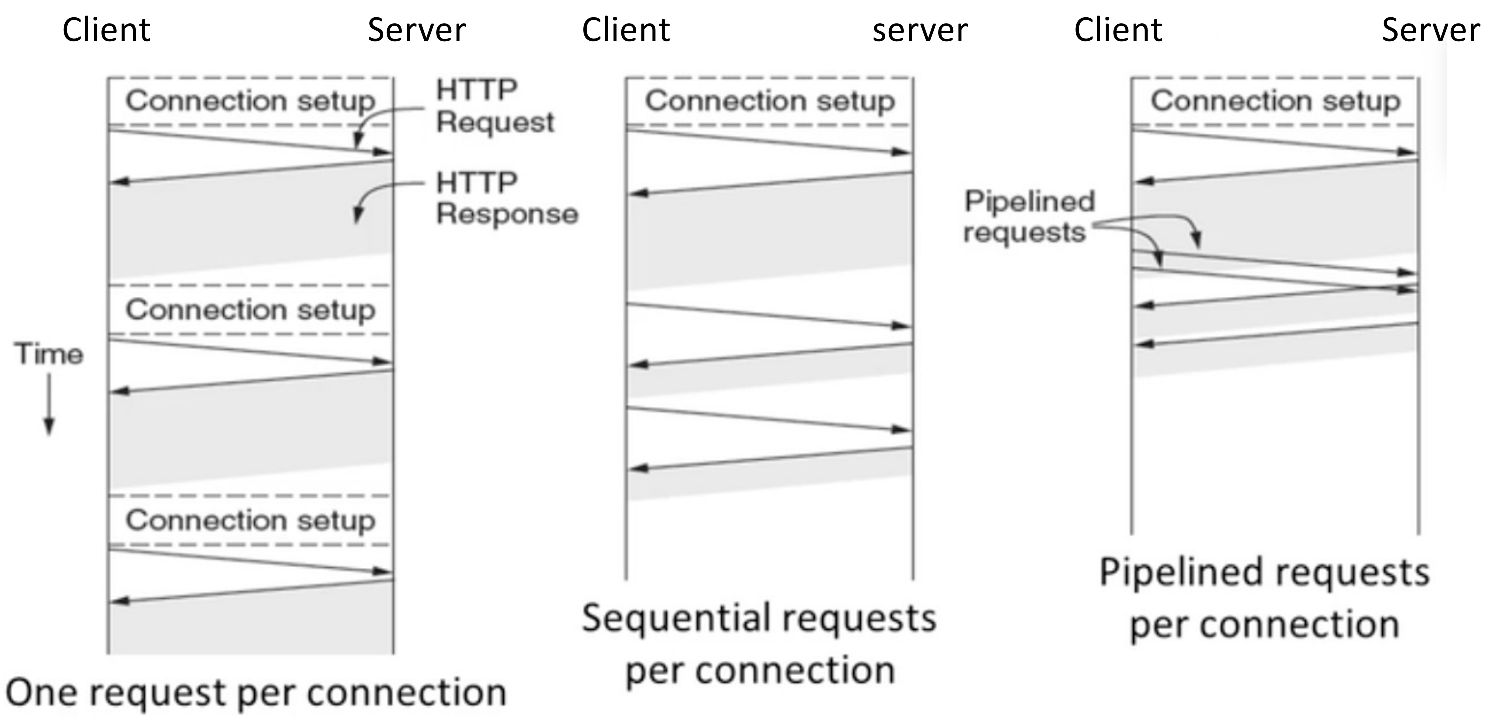
Parallel Connections

- 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 was not using network much …
  - So parallel connections are not slowed much

Persistent Connections

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

Persistent Connections (2)

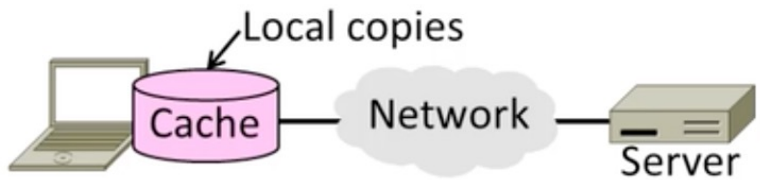


Persistent Connections (3)

- 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?

Web Caching

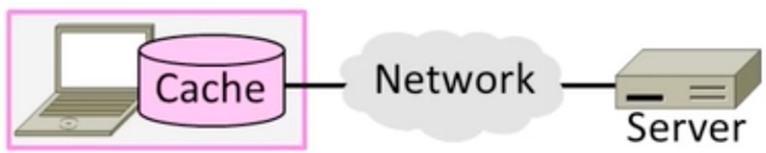
- Users often revisit web pages
  - Big win from reusing local copy!
  - This is caching



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

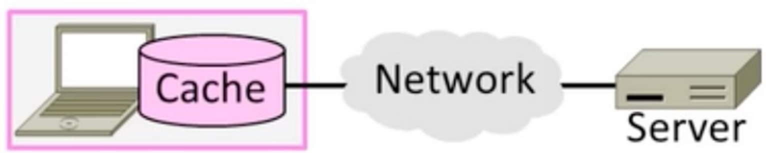
Web Caching (2)

- 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



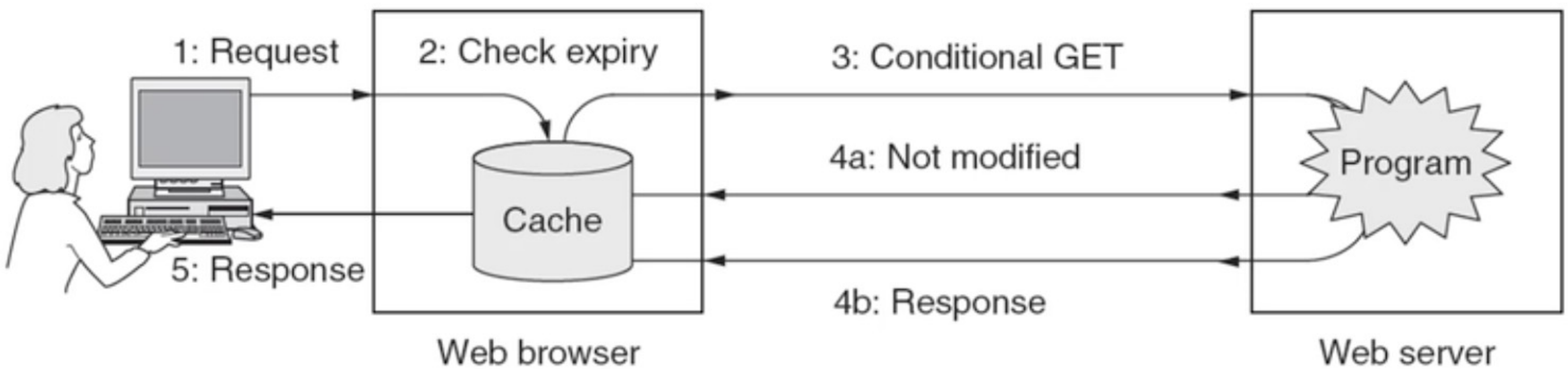
Web Caching (3)

- Revalidate copy with remote 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:



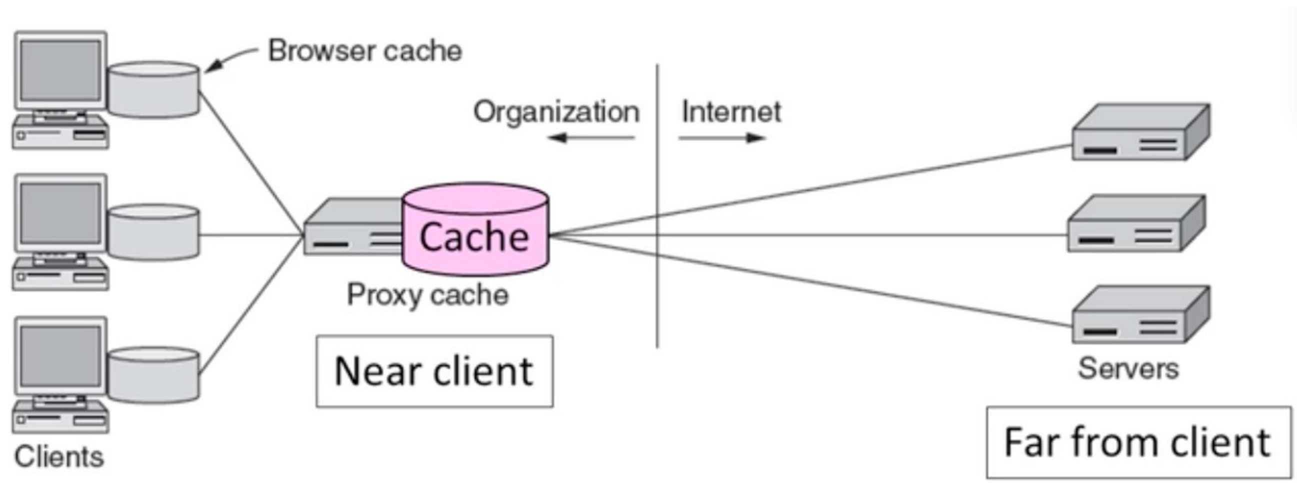
Web Proxies

- 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
  - Client benefit from larger, shared cache
  - Benefits limited by secure/dynamic content, as well as “long tail”

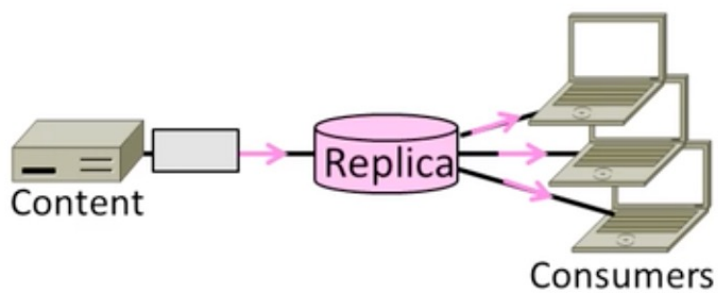
Web Proxies (2)

- Clients contact proxy; proxy contacts server



CDN (Content Delivery Networks)

- Efficient distribution of popular content; faster delivery for clients



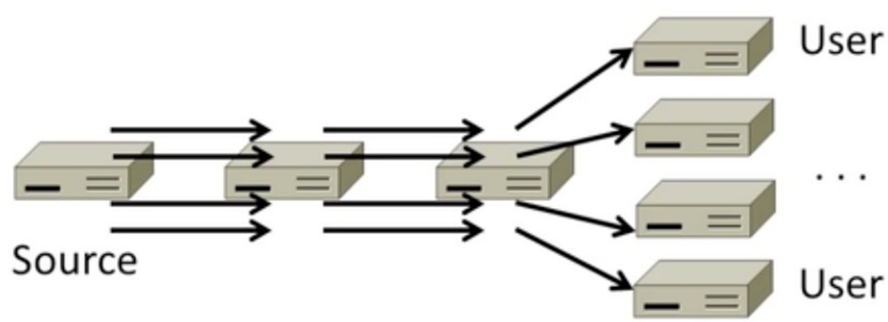
Context

- As the web took off in the 90s, traffic volumes 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
    - Help with all three issues above

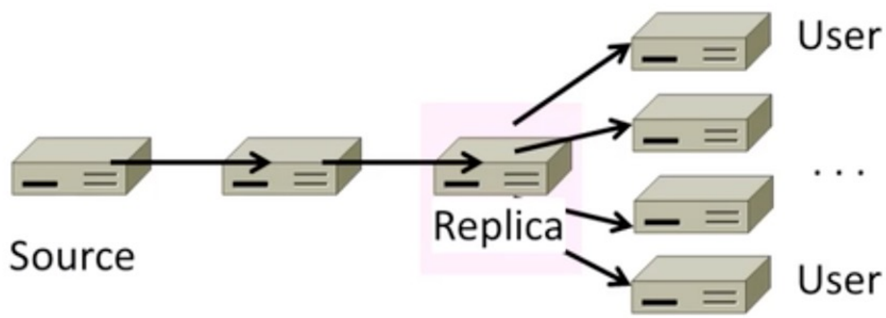
Before CDNs

- Sending content from the source to 4 users takes 4x3=12 “network hops” in the example



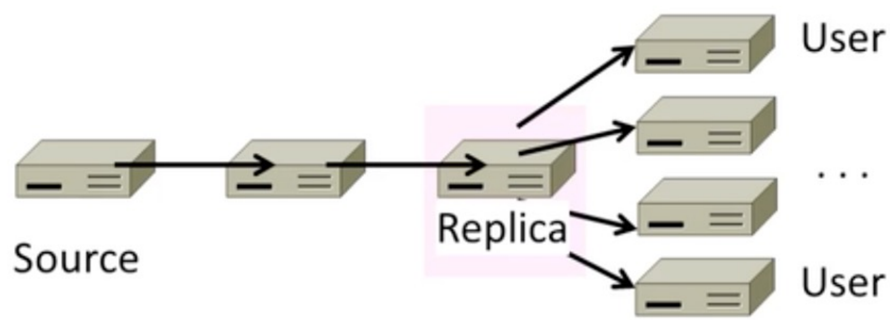
After CDNs

- 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 (PLT)

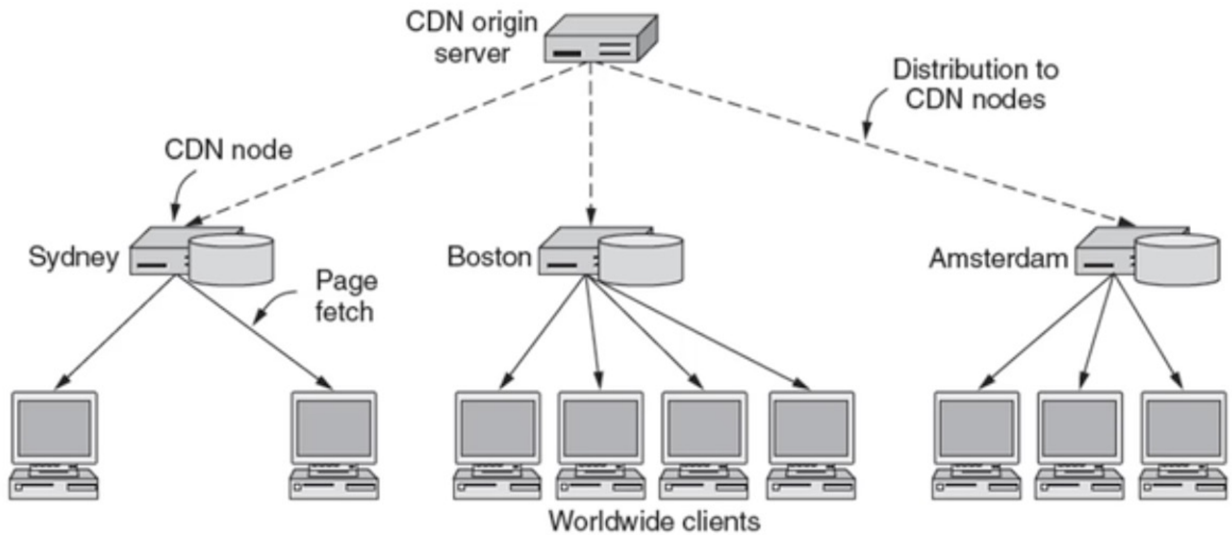


How to Place Content near Clients?

- Use browser an proxy caches
  - Helps, but limited to one client or clients in one organization

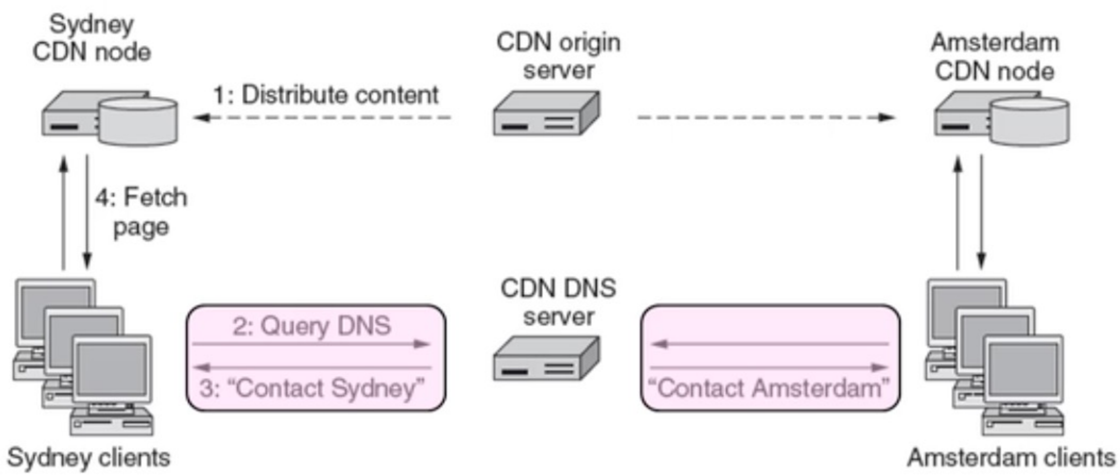
- Want to place replicas across the Internet for use by all nearby clients
  - Done by clever use of DNS

Content Delivery Network



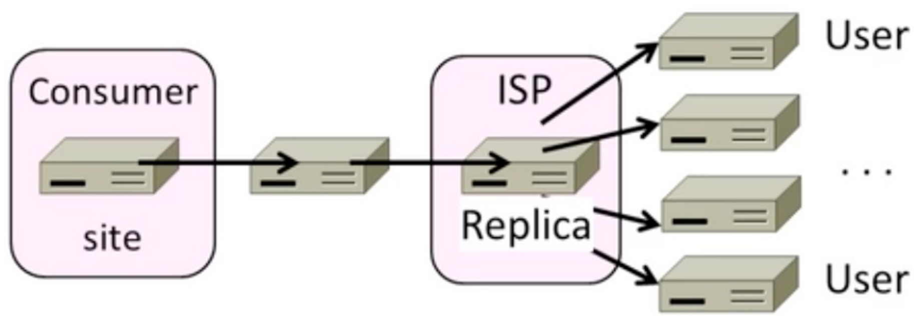
Content Delivery Network (2)

- DNS resolution of site gives different answers to clients
  - Tell each client the site is 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 bandwidth usage of ISP



To-do

- Quiz on next week
- Presentations will start next week