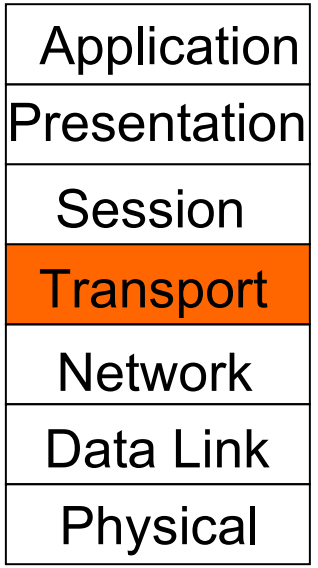
CSE/EE 461
DNS / HTTP / CDNs / BitTorrent

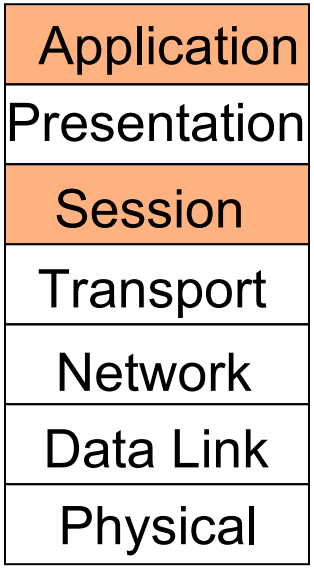
Last Time …

- The Transport Layer
- Focus
  - How does TCP share bandwidth?
- Topics
  - AIMD
  - Slow Start
  - Fast Retransmit / Fast Recovery



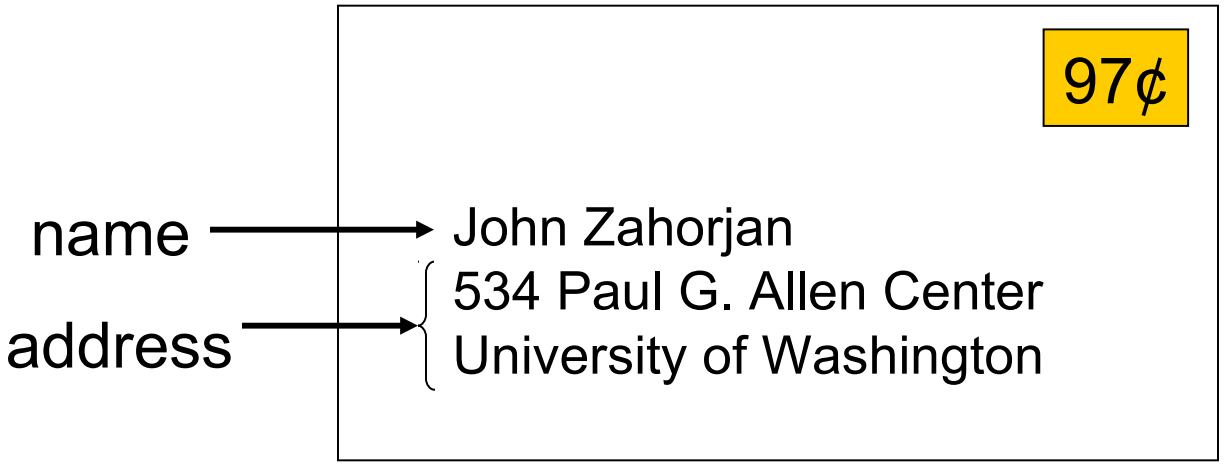
This Lecture

- DNS
- HTTP and the Web (but not HTML)
- Content Distribution Networks (CDNs)
- BitTorrent



Part 1: The Domain Name System (DNS)

Names and Addresses



Names are identifiers for objects/services (high level)

Addresses are locators for objects/services (low level)

Binding is the process of associating a name with an address

Resolution is the process of looking up an address given a name

But, addresses are really lower-level names; many levels used

Internet Hostnames

- Hostnames are human-readable identifiers for end-systems based on an administrative hierarchy
  - peshastin.cs.washington.edu is my desktop machine
- IP addresses are a fixed-length binary encoding for end-systems based on their position in the network
  - 128.208.2.83 is peshastin's IP address
- Original name resolution: HOSTS.TXT
- Current name resolution: Domain Name System (DNS)

Domain Name System (DNS)

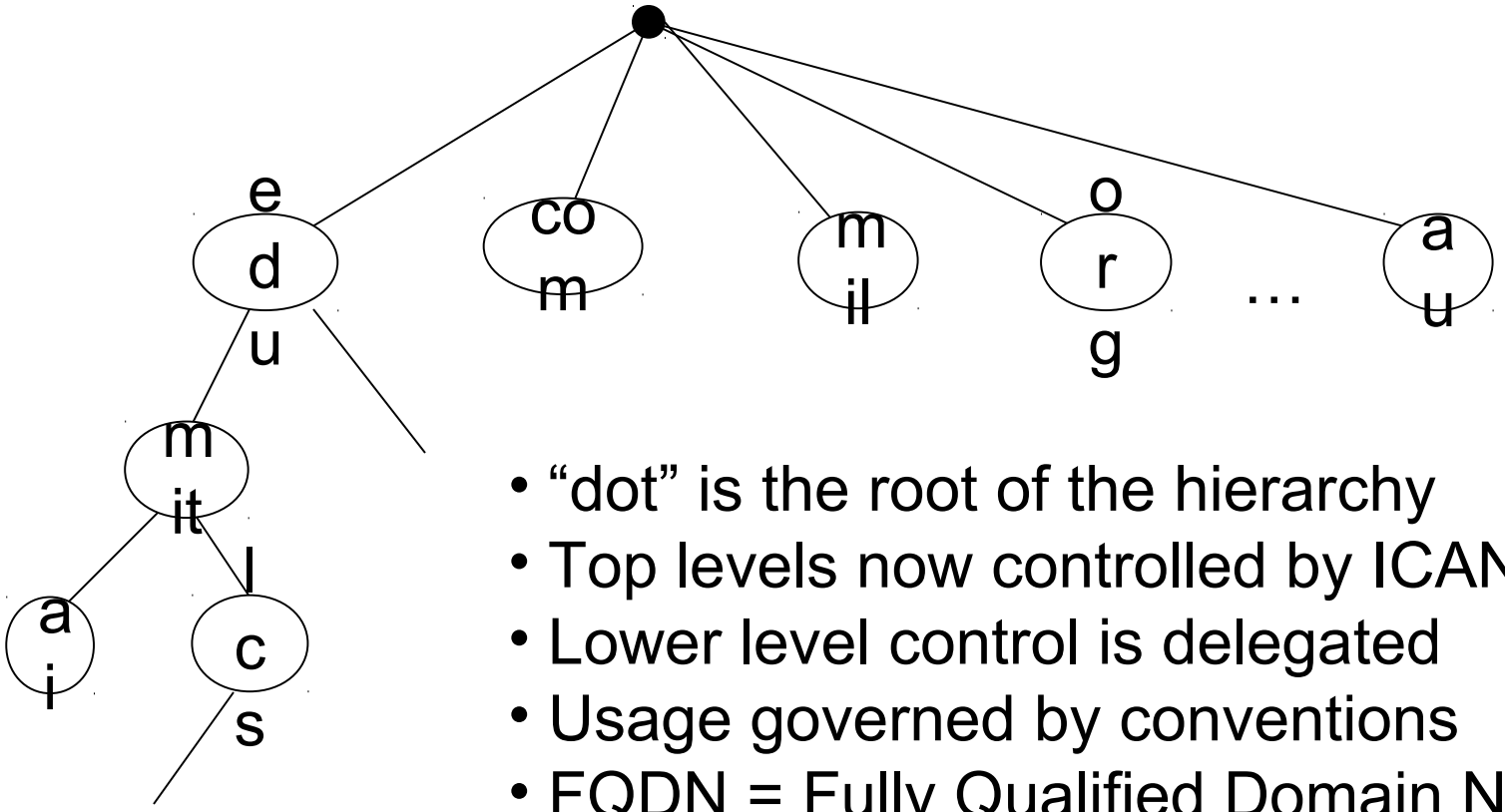
- Designed in the mid '80s

- Namespace is hierarchical
  - Decentralized administration
    - *.cs.washington.edu managed by CSE
    - *.washington.edu managed by UW Office of Technology
    - *.edu managed by EDUCAUSE (http://net.educause.edu)
    - . managed by ICANN (Internet Consortium for Assigned Names and Numbers: www.icann.org)

- Name service (DNS) is distributed
  - Allows much better scaling of data structures
    - e.g., peshastin.cs.washington.edu

- Resolution is by query/response
  - With replicated servers for redundancy
  - With heavy use of caching for performance

DNS Hierarchy



- “dot” is the root of the hierarchy
- Top levels now controlled by ICANN
- Lower level control is delegated
- Usage governed by conventions
- FQDN = Fully Qualified Domain Name

DNS Distribution

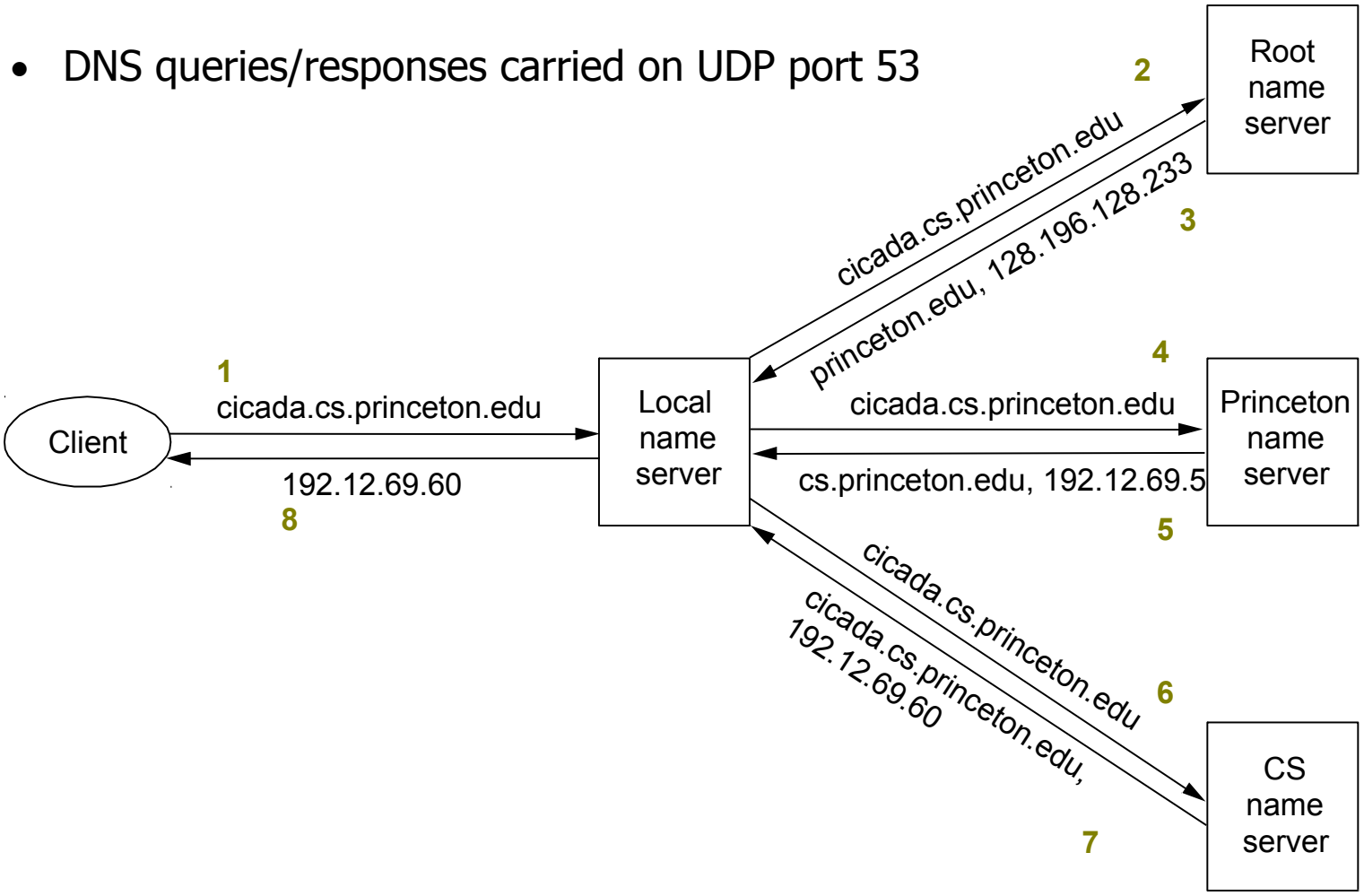
- Data managed by zones that contain resource records
  - Zone is a complete description of a portion of the namespace
  - e.g., all hosts and addresses for machines in washington.edu with pointers to subdomains like cs.washington.edu

- One or more nameservers manage each zone
  - Zone transfers performed between nameservers for consistency
  - Multiple nameservers provide redundancy

- Client resolvers query nameservers for specified records
  - Multiple messages may be exchanged per DNS lookup to navigate the name hierarchy (coming soon)

DNS Lookups/Resolution

- DNS queries/responses carried on UDP port 53



DNS Bootstrapping

- Need to know IP addresses of root servers before we can make any queries
- Addresses for 13 root servers ([a-m].root-servers.net) handled via initial configuration (named.ca file)
- To avoid overloading roots, heavy use of caching (at all levels)
  - Cache entries are timed out
  - DNS also caches negative entries!
    - Mistyped names are the most frequent single request to root servers.

DNS Resource Records

DNS: distributed DB storing resource records (RR)

RR format: (name, value, type, ttl)

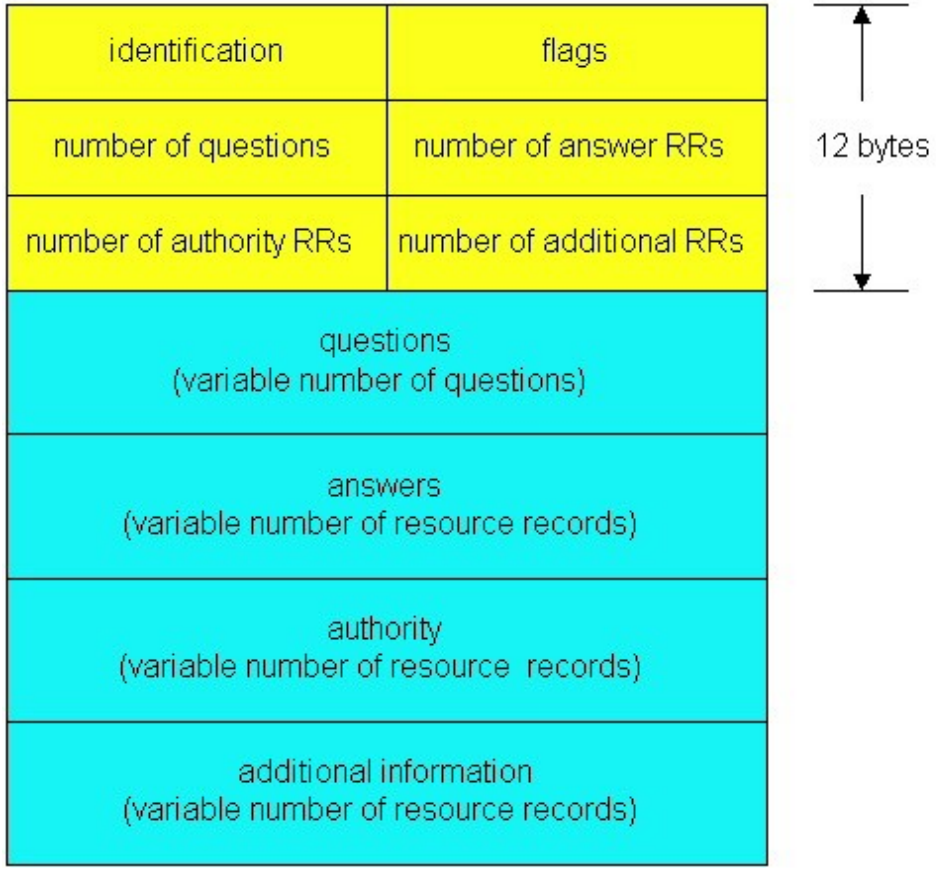
- Type=A
  - name is hostname
  - value is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain
- Type=CNAME
  - name is alias name for some "canonical" (the real) name www.ibm.com is really servereast.backup2.ibm.com
  - value is canonical name
- Type=MX
  - value is name of mailserver associated with name

DNS Protocol

DNS protocol : *query* and *reply* messages, both with same *message format*

Message header

- Identification: 16 bit # for query, reply to query uses same #
- Flags:
  - Query or reply
  - Recursion desired
  - Recursion available
  - Reply is authoritative



Reliability

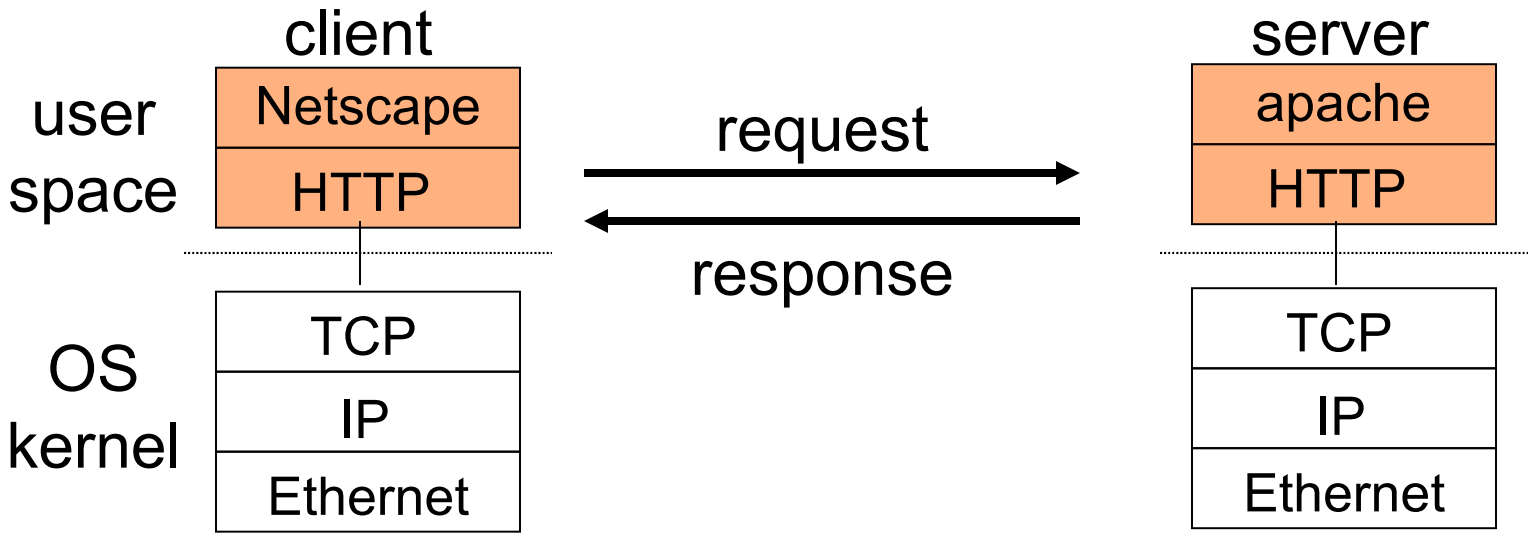
- DNS servers are replicated
  - Name service available if at least one replica is up
  - Queries can be load balanced between replicas
- UDP used for queries
  - Need reliability: must implement this on top of UDP
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Same identifier for all queries
  - Don’t care which server responds

DNS Live Demo

- nslookup *FQDN*
  - Maps FQDN to IP
- dig *FQDN*
  - Better nslookup
- whois *ip_address*
  - Consults ARIN (American Registry for Internet Numbers)

Part 2: The Web

Web Protocol Stacks



- To view the URL http://server/page.html the client makes a TCP connection to port 80 of the server, by it's IP address, sends the HTTP request, receives the HTML for page.html as the response, repeats the process for inline images, and displays it.

HTTP/HTML

- Original goal of the web: file sharing / viewing

- Big idea: display of content should be up to viewer, not author
  - Data might be astronomical observations
  - Client may or may not be able to run a graphical visualizer
  - Data of many different types: plain text, formatted text, tables of values, images, etc.
  - (Compare this idea with WYSWIG, e.g., Word)

- **HTTP** (Hyper Text Transfer Protocol) is the protocol used to communicate between the browser and the server

- **HTML** (Hyper Text Markup Language) is the language for marking format marks in text
  - <p> means "start a new paragraph"
  - xxx means "display xxx in bold"
  - italicize me
  - Etc.

HTTP

- HTTP was originally intended to support file fetch
- HTTP is a request-response protocol
  - Client ⇒ Server: Give me file `index.html`
  - Server ⇒ Client: Here it is…
- Like every other protocol we've seen, it defines a header, which is used to communicate meta-data between the client and the server
  - Client header: operation, URL, HTTP version, etc.
    - `GET /education/courses/cse461/07au/index.html HTTP/1.0`
  - Server header: result code, file metadata
    - `HTTP/1.0 200`

HTTP Protocol Headers

- HTTP runs on top of TCP, which runs on top of IP
  - So?
- HTTP headers are very flexible
  - Expressed as text (ASCII)
  - Keyword delimited fields, not position (mostly)
    - First line IS position sensitive
    - As always, need framing
      - A blank line (CrLf) separates header from payload

Example Exchange

- ```
  GET /index.html HTTP/1.1
  User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
  rv:1.8.0.8) Gecko/20061025 Firefox/1.5.0.8
  Host: www.w3.org
  Accept: text/plain, text/html
  ```

- ```
  HTTP/1.1 200 OK
  Date: Thu, 30 Nov 2006 09:55:58 GMT
  Server: Apache/1.3.37 (Unix) mod_pubcookie/3.3.2b
  mod_ssl/2.8.28 OpenSSL/0.9.8a
  Transfer-Encoding: chunked
  Content-Type: text/html; charset=iso-8859-1
  Content-Language: en 48a

  48a
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
  "http://www.w3.org/TR/REC-html40/loose.dtd">
  <html>
  ```

Payload is typed (MIME types)

- Multi-part Internet Mail Exchange (MIME) types
  - `text/plain`
  - `text/html`
  - `application/msword`
  - `application/octet-stream`
  - `img/jpeg`
  - `audio/mpeg`
  - `video/quicktime`
  - `multipart/encrypted`

multipart/encrypted

```
Content-Type: multipart/encrypted; protocol="TYPE/STYPE";
boundary="Encrypted Boundary"

--Encrypted Boundary
Content-Type: TYPE/STYPE

   CONTROL INFORMATION for protocol "TYPE/STYPE" would be
here

--Encrypted Boundary
Content-Type: application/octet-stream

   Content-Type: text/plain; charset="us-ascii"
```

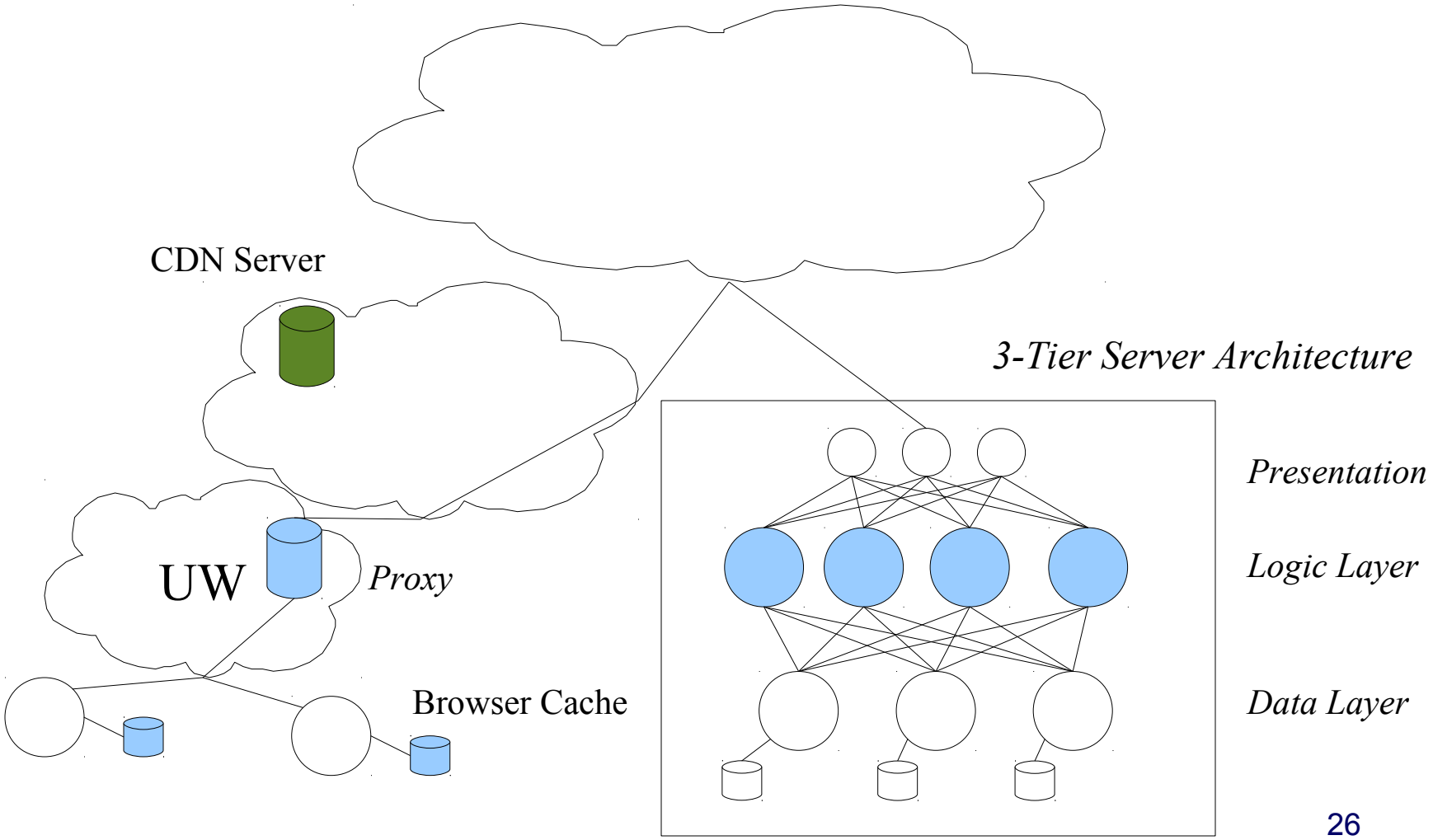
HTTP/HTML Redux

- Original idea of HTTP/HTML was to support fetch and display of remote data
  - Everyone is a publisher, everyone is their audience
- Original vision was that object being fetched is a file
  - Modified to allow *dynamic content*: pages generated on-the-fly
    - Example: Amazon pages
  - "cgi" (common gateway interface) means a program that is run because of an page fetch request given to a web server
  - "server side include" is a cruder form of server-side dynamic content
    - `http://www.cs.washington.edu/education/courses/cse461/07au/index.shtml`
  - Applets / Javascript are dynamic content, but running on client side
  - Many more modifications to the original idea have evolved over time

Improving Performance #1: Caching

- Caches can be (and are) deployed in three places:
  - The client
    - The browser keeps a cache of recently viewed pages
      - Avoid network latency/overhead
  - The server
    - Avoid disk IO
  - The network
    - Proxy caches intercept HTTP requests and return cached page
    - Typically deployed between local network(s) and ISP
      - E.g., a UW proxy cache
- Both client and server can influence if / how long a page is cached
  - Server: cacheable/non-cacheable, expiration time
  - Client: browser settings for how often to check for page changed

Caching (and more...)



How Effective Is Caching?

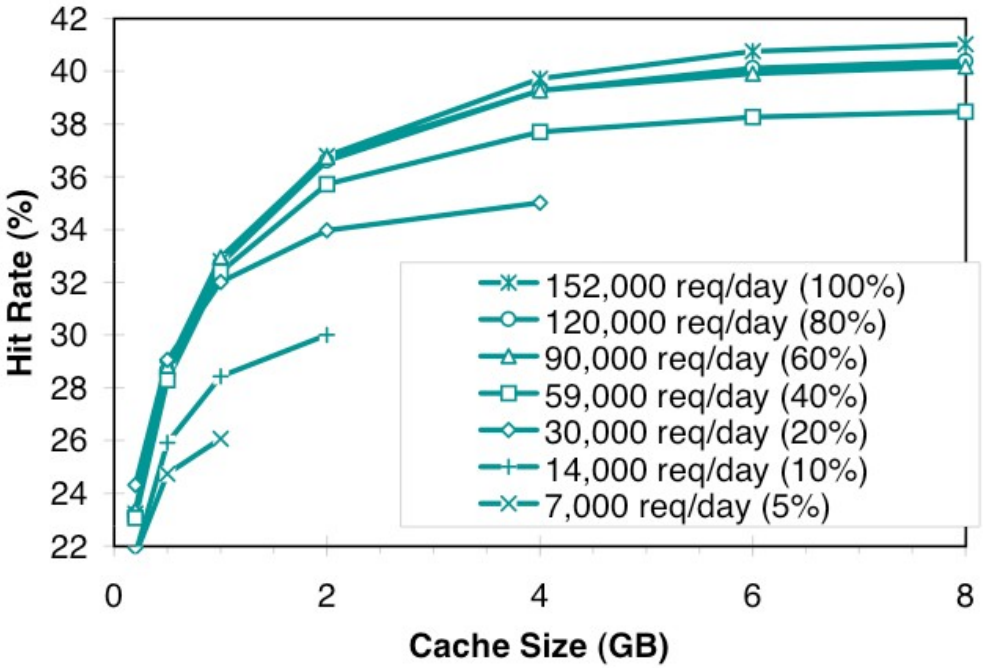
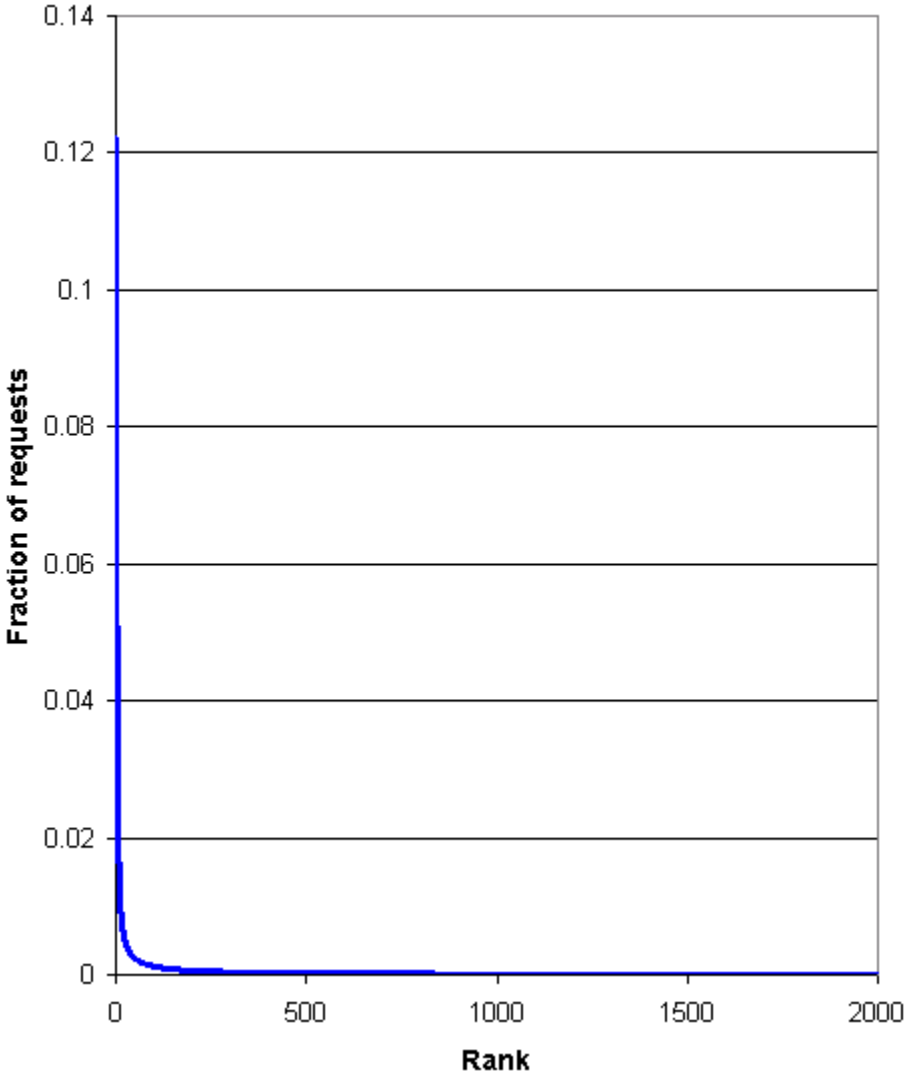


Figure 3: Cache hit rate for KOR as a function of cache size for a range of request rates.

Why?

- Page popularity follows Zipf's law
  - N^{th} most popular page is accessed with frequency proportional to 1/N
  - "Heavy tail" of this distribution means there are a lot of pages not visited by many clients



Improving Performance #2: HTTP Protocol Tweak

HTTP Request/Response in Action

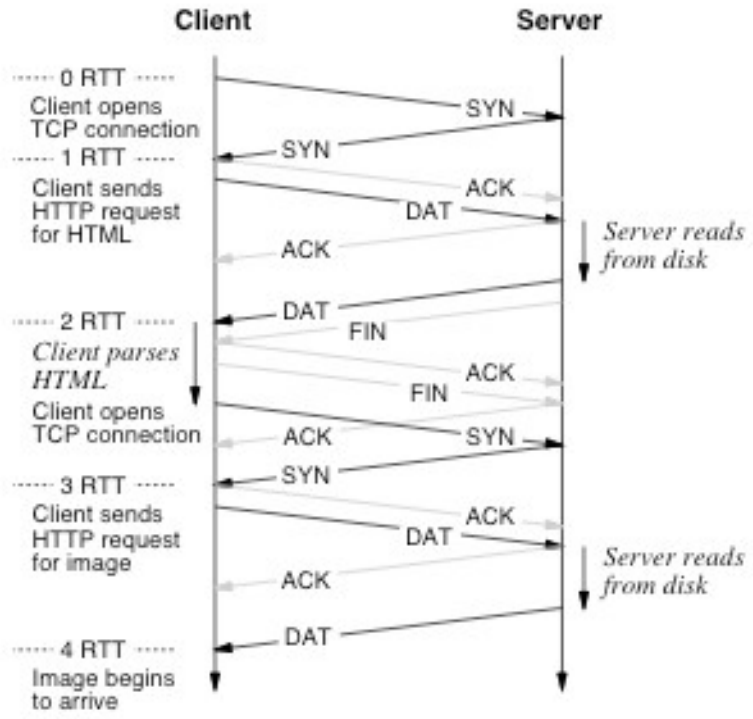


Figure 3-1: Packet exchanges and round-trip times for HTTP

- Problem is that:
  - Web pages are made up of many files
    - Most are very small (< 10k)
  - files are mapped to connections

  For each file
  - Setup/Teardown
    - Time-Wait table bloat
  - 2RTT "first byte" latency
  - Slow Start+ AIMD Congestion Avoidance

  The goals of HTTP and TCP protocols are not aligned.
  - Implications

TCP Behavior for Short Connections Over Slow Networks

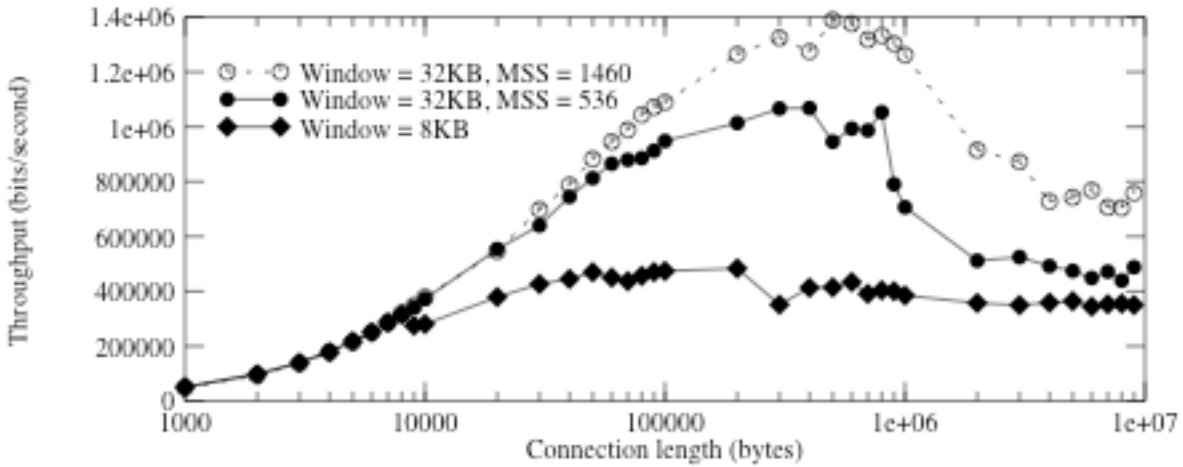


Figure 3-2: Throughput vs. connection length, RTT = 70 msec

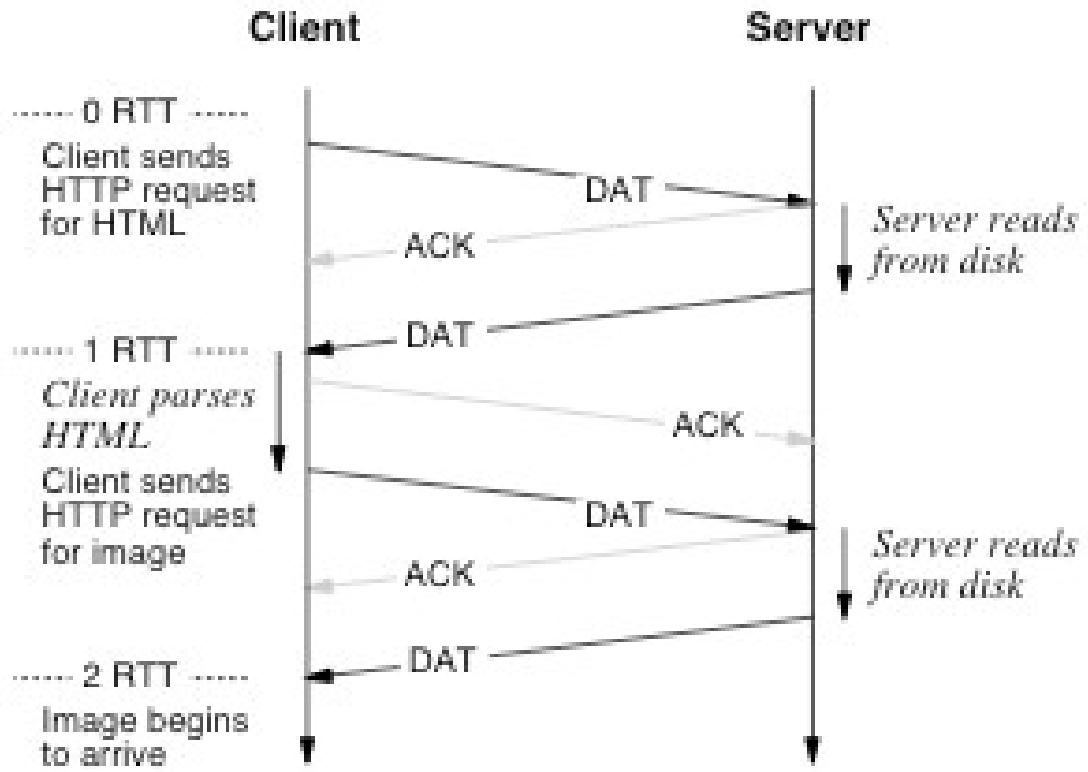
Figure 3-2 shows that, in the remote case, using a TCP connection to transfer only 2 Kbytes results in a throughput less than 10% of best-case value. Even a 20 Kbyte transfer achieves only about 50% of the throughput available with a reasonable window size. This reduced throughput translates into increased latency for document retrieval. The figure also shows that, for this 70 msec RTT, use of too small a window size limits the throughput no matter how many bytes are transferred.

RTT=70ms

HTTP 1.1: Persistent Connections

- Bright Idea: Use one TCP connection for multiple page downloads (or just HTTP methods)
- Q: What are the advantages?
- Q: What are the disadvantages?
  - *Application layer multiplexing*

HTTP/1.1


Client
Server
0 RTT
Client sends HTTP request for HTML
DAT
ACK
Server reads from disk
DAT
1 RTT
Client parses HTML
ACK
Client sends HTTP request for image
DAT
ACK
Server reads from disk
DAT
2 RTT
Image begins to arrive

Effect of Persistent HTTP

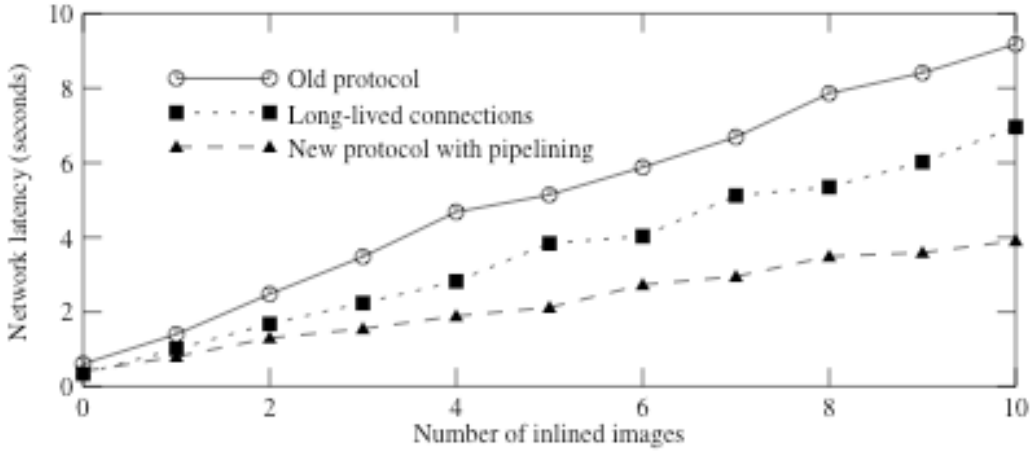


Figure 6-1: Latencies for a remote server, image size = 2544 bytes

Image size=2544

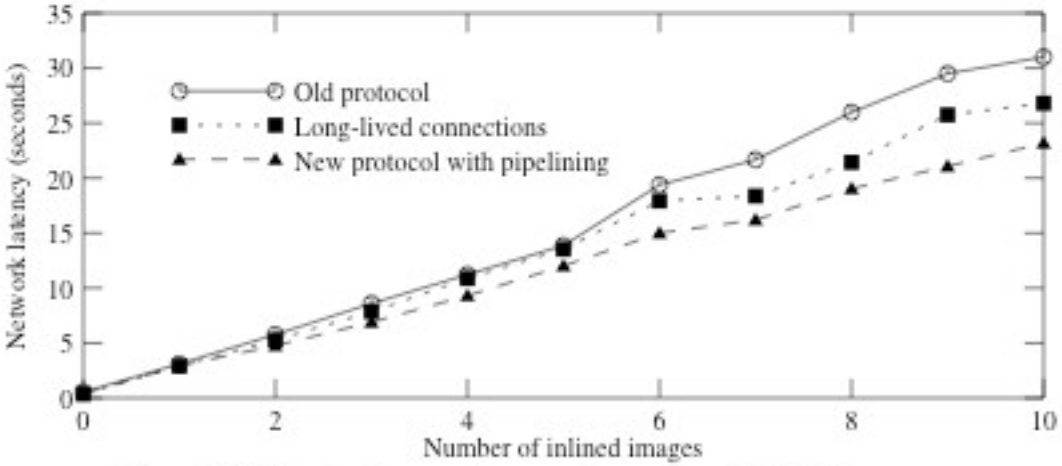


Figure 6-2: Latencies for a remote server, image size = 45566 bytes

Image size=45566

HTTP Live Demo

- HTTP Live Headers
  - Firefox add-on

Part 3: Content Distribution Networks

CDNs: Motivation

- Imagine someone on campus fetches
  `www.nytimes.com/2006/12/04/technology/04adcol.html`

- It benefits both the UW and the NYT if that page is cached at the UW
  - UW pays less for bandwidth to request page
  - NYT pays less for bandwidth/servers to deliver the page

- But
  - NYT can't rely on everyone being behind an effective cache, and..
  - NYT wants to be able to customize its content

- Basic idea of CDNs:
  - Push content to servers operating near clients
    - E.g., Akamai

CDNs: Leverage DNS

- Uniform Resource Locators (URLs) begin with a host name
  - `www.nytimes.com/2006/12/04/technology/04adcol.html`
    - Host is `www.nytimes.com`
    - Requested file is `/2006/12/04/technology/04adcol.html`
- Web pages embed links to other pages
  - Example:
    `<img src="http://graphics10.nytimes.com/ads/fox/article-sponsor.gif">`
- Key idea: use DNS to direct fetch of content (like images) to CDN servers
  - `graphics10.nytimes.com` is a CDN server, not one on the NYT local network

DNS & CDNs

```
$ dig www.nytimes.com
;; ANSWER SECTION:
www.nytimes.com.        294     IN      A       199.239.137.245
www.nytimes.com.        294     IN      A       199.239.136.200
www.nytimes.com.        294     IN      A       199.239.136.245
www.nytimes.com.        294     IN      A       199.239.137.200
```

```
$ dig graphics10.nytimes.com
;; ANSWER SECTION:
graphics10.nytimes.com. 300     IN      CNAME   graphics8.nytimes.com.
graphics8.nytimes.com.  300     IN      CNAME   graphics478.nytimes.com.edgesuite.net.
graphics478.nytimes.com.edgesuite.net. 5495 IN CNAME a1116.x.akamai.net.
a1116.x.akamai.net.     20      IN      A       72.246.103.40
a1116.x.akamai.net.     20      IN      A       72.246.103.59
```

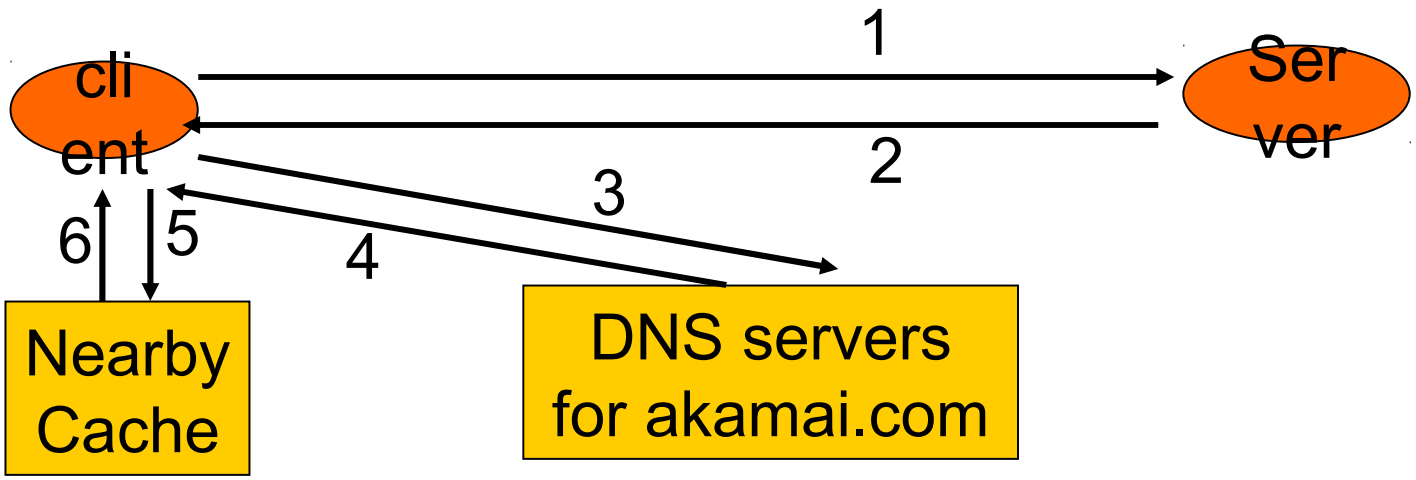
Future Evolution of the DNS

- Design constrains us in two major ways that are increasingly less appropriate

- Static host to IP mapping
  - What about mobility (Mobile IP) and dynamic address assignment (DHCP)
- Location-insensitive queries
  - What if I don't care what server a Web page comes from, as long as it's the right page?
  - e.g., a yahoo page might be replicated

Akamai

- Use the DNS to effect selection of a nearby Web cache

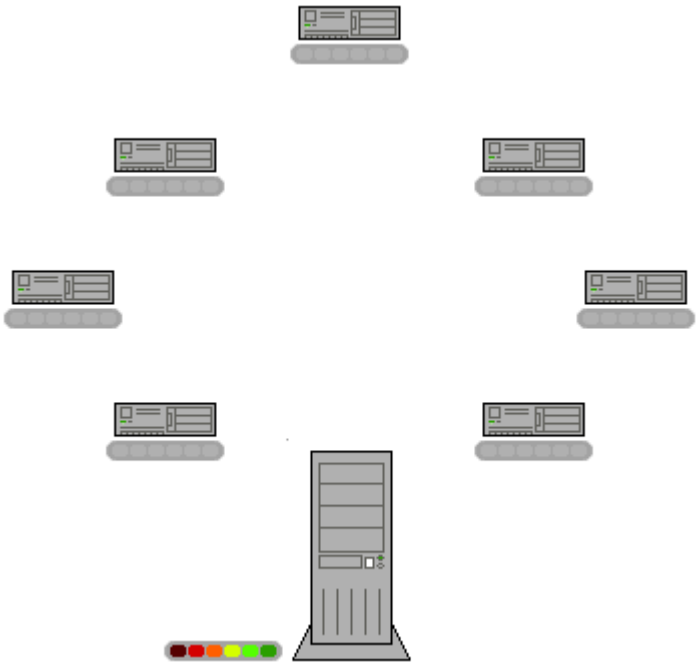


- Leverage separation of static/dynamic content
- Beware DNS caching

Part 4: BitTorrent

- History of file sharing:
  - 1999: Napster
    - 2000/1: *A&M Records vs. Napster*
  - 2000- : Gnutella, Kazaa, …
    - 2006: Kazaa loses $100M award to Sony, EMI, etc.
  - 2001: first BitTorrent version
    - Designed as a solution to "the slashdot effect"

BitTorrent: The Idea



Source: *http://en.wikipedia.org/wiki/BitTorrent_%28protocol%29*

BitTorrent Overview

- A *torrent file* points to a *tracker*
- A tracker serves a single copy of the content, and tracks what content other machines have
- Peers in a *swarm* exchange file chunks among themselves

- A *tit-for-tat* mechanism is used to try to discourage *freeloading*

BitTorrent: Peer and Chunks

- Large file divided into smaller pieces
  - Fixed-sized chunks
  - Typical chunk size of 16KB - 256 KB
- Allows simultaneous transfers between peers
  - Downloading chunks from different neighbors
  - Uploading chunks to other neighbors
  - Favor neighbors that are contributing (incentives)
- Learning what chunks your neighbors have
  - Broadcast to neighbors when you have a chunk
- File done when all chunks are downloaded

BitTorrent: The Spec

http://wiki.theory.org/BitTorrentSpecification#bencoding