Content Distribution

Professor Keith W. Ross
Institut Eurécom
Sophia Antipolis, France

http://www.eurecom.fr/~ross

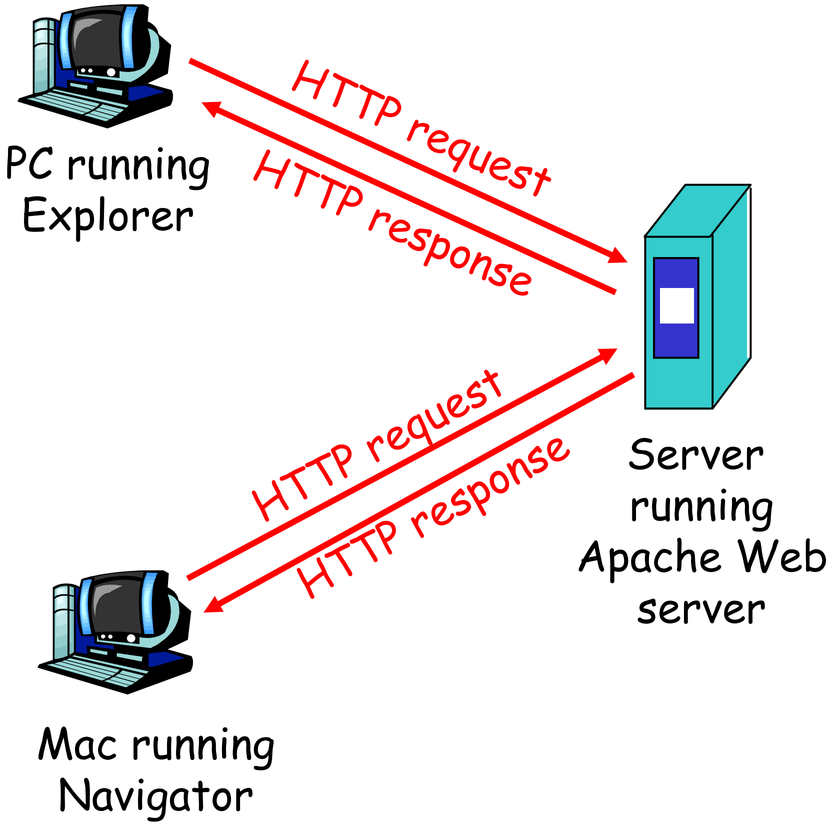
Content Distribution

- Review of HTTP, DNS, TCP
- Server Farms
- Proxy Web Caches
- Content Distribution Networks (CDNs)
- Peer-to-peer file sharing (P2P)

HTTP overview

HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, "displays" Web objects
  - *server:* Web server sends objects in response to requests
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



HTTP request message

- two types of HTTP messages: *request, response*
- HTTP request message:
  - ASCII (human-readable format)

request line (GET, POST, HEAD commands)

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr

(extra carriage return, line feed)
```

header lines

Carriage return, line feed indicates end of message

HTTP response message

status line
(protocol
status code
status phrase)

header
lines

data, e.g.,
requested
HTML file

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., gaia.cs.umass.edu - used by humans

Q: map between IP addresses and name ?

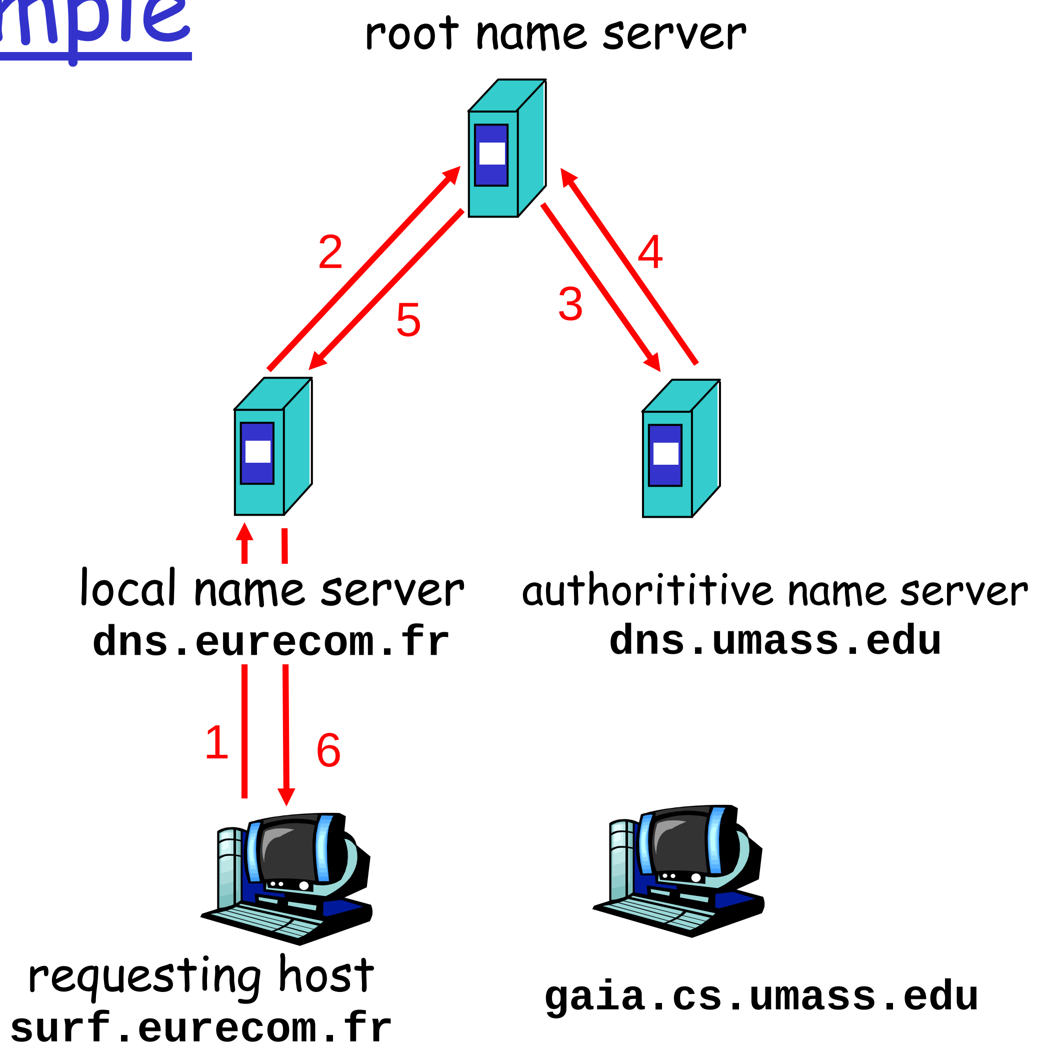
Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*

Simple DNS example

host **surf.eurecom.fr** wants IP address of **gaia.cs.umass.edu**

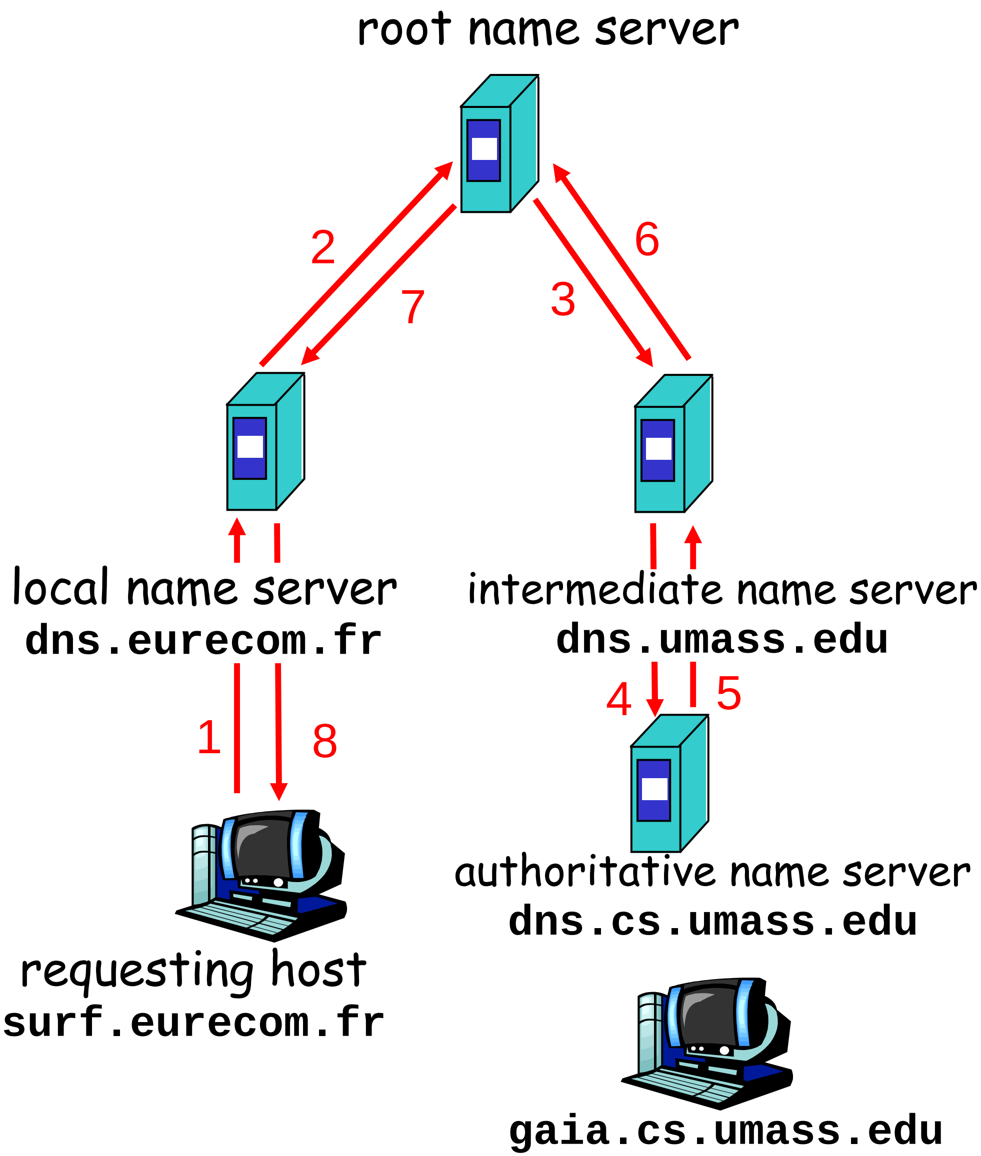
1. contacts its local DNS server, **dns.eurecom.fr**
2. **dns.eurecom.fr** contacts root name server
3. root name server contacts authoritative name server, **dns.umass.edu**



DNS example

Root name server:

- may not know authoritative name server
- may know *intermediate name server:* who to contact to find authoritative name server



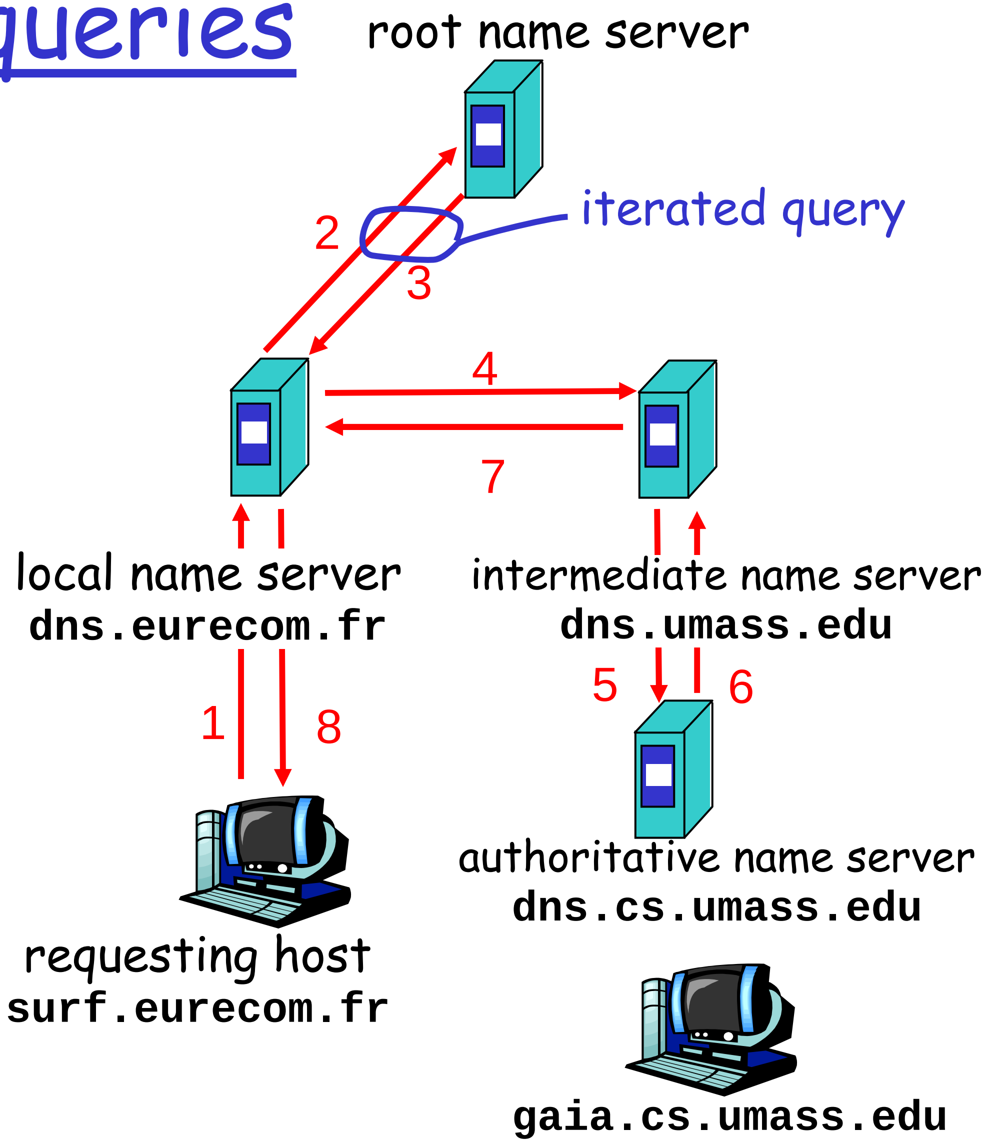
DNS: iterated queries

recursive query:

- puts burden of name resolution on contacted name server
- heavy load?

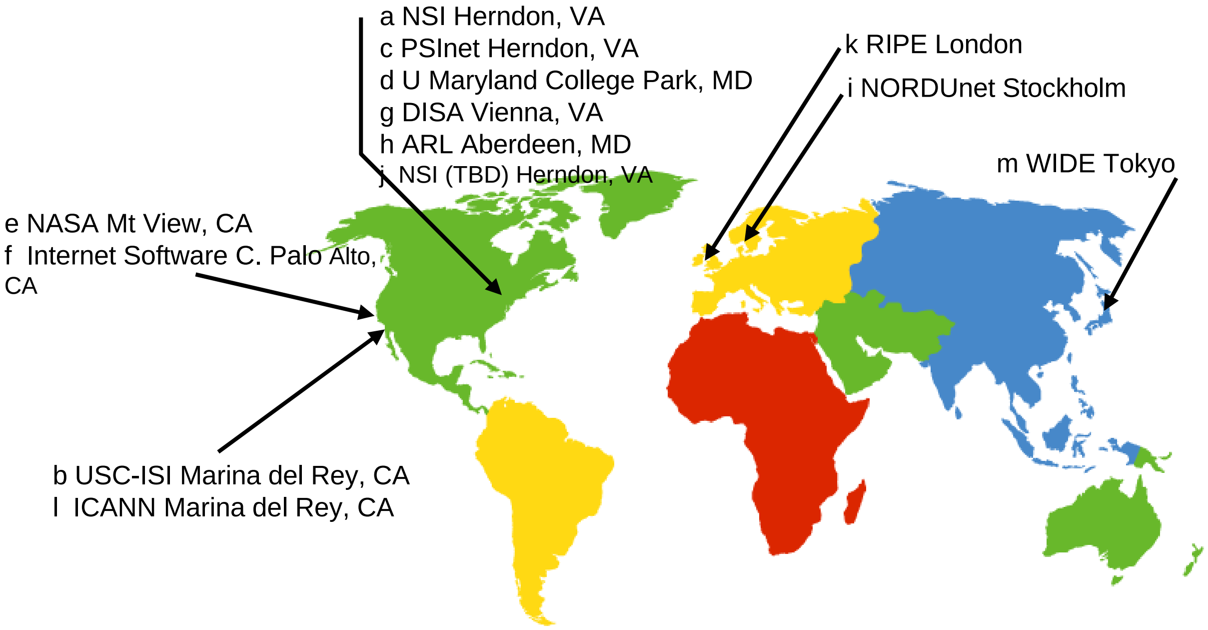
iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS: Root name servers

- contacted by local name server that can not resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server



13 root name servers worldwide

DNS: caching and updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time
- update/notify mechanisms under design by IETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

TCP connection establishment

TCP 3-way handshake:

- client sends to server TCP segment with SYN bit is set.
- server responds with segment that has both SYN and ACK bits set.
- client responds with another segment with ACK bit set.

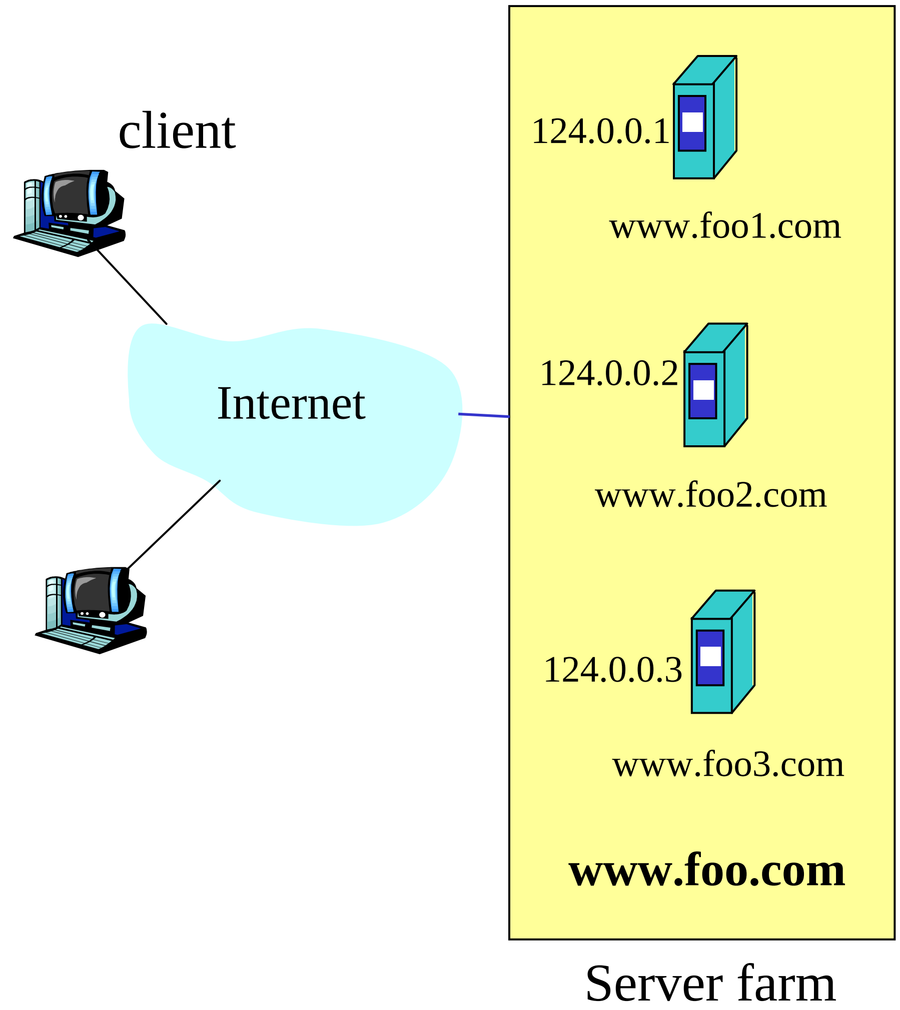
Content Distribution

- Review of HTTP, DNS, TCP
- Server Farms
- Proxy Web Caches
- Content Distribution Networks (CDNs)
- Peer-to-peer file sharing (P2P)

Architectures for server farms

Issue: In a Web application, multiple server machines may be needed to handle traffic.

Goal: Architect server farm so that it appears to be running on a single machine to the client.



Architectures for server farms

Running example:

Site:

- http://www.foo.com

Three machines:

- foo1, foo2, foo3

Company address space:

- 124.x.x.x

For simplicity:

- assume static Web pages

3 architectures

- DNS rotation
- Surrogate server
- NAT

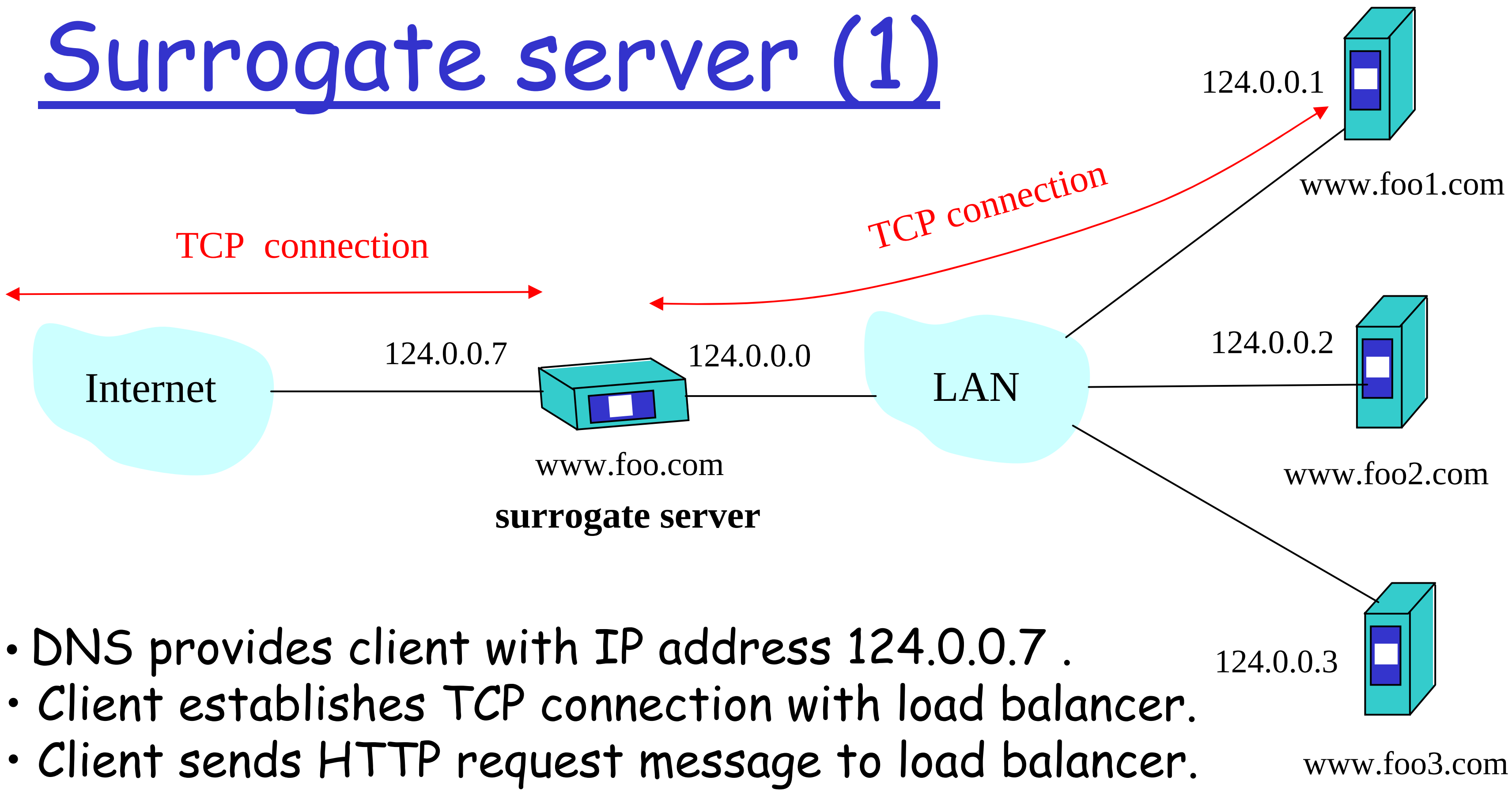
DNS rotation

- DNS translates host.names to IP addresses
- eg, www.foo.com to 124.0.0.7
- Authoritative DNS server can provide multiple IP addresses:
  - 124.0.0.1, 124.0.0.2, 124.0.0.3
- Rotates addresses

Issues

- DNS caching in name servers
- Name server directs its clients to the same Web server
- Hard to control how much traffic is directed at each of the servers

Surrogate server (1)



- DNS provides client with IP address 124.0.0.7 .
- Client establishes TCP connection with load balancer.
- Client sends HTTP request message to load balancer.
- Surrogate (load balancer) chooses one of the three Web servers.
- Surrogate establishes TCP connection with chosen server and forwards request.
- Surrogate forwards data it receives from server to client.

Surrogate server (2)

Flexibility in determining how to direct requests to servers:

- load on servers
- requested content
- cookie in request
- application-layer switch: L7 switch

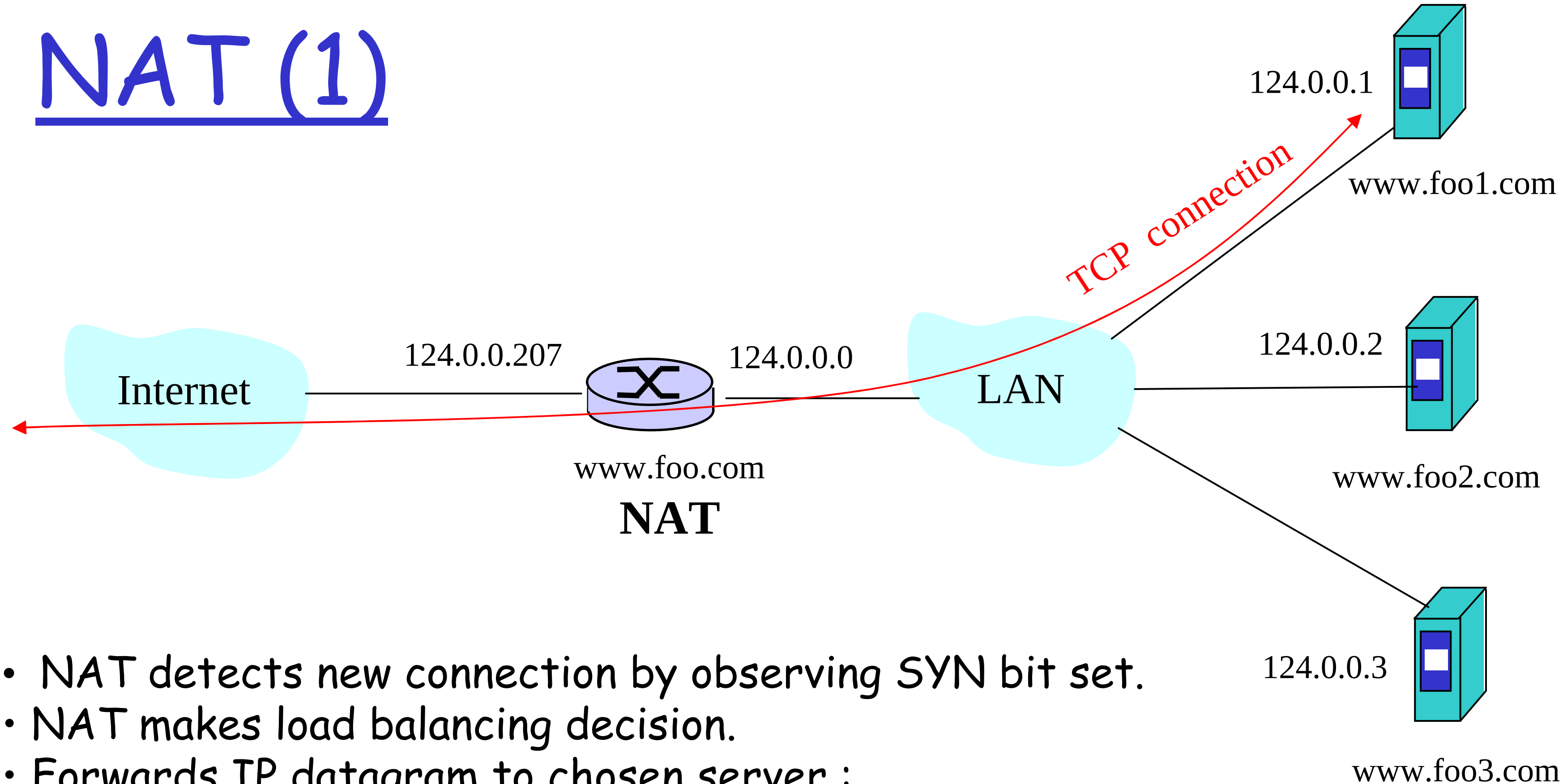
Inefficient:

- surrogate has to serve at the combined rate of servers

Implementation

- Develop on top of Linux, Solaris, Windows 2000.
- Performance bottleneck: lots of TCP processing, application-layer systems calls
- Can develop customized operating system for surrogate server, resulting in efficient load balancer

NAT (1)



- NAT detects new connection by observing SYN bit set.
- NAT makes load balancing decision.
- Forwards IP datagram to chosen server :
  - Needs to modify destination IP address
  - Needs to forward subsequent packets in same connection to the same server; modify destination IP addresses
  - Needs to modify datagrams arriving from servers
- TCP connection is established end-to-end.
  - But client thinks it has TCP connection with 124.0.0.7

NAT (2): Table

- Suppose client has IP address 227.68.68.19 and source port number 9876
- NAT table might have for this connection:

Internet	to	LAN
SA: 227.68.68.19		SA: 227.68.68.19
DA: 124.0.0.7		DA: 124.0.0.1
SP: 9876		SP: 9876
DP: 80		DP: 80
LAN	to	Internet
SA: 124.0.0.1		SA: 124.0.0.7
DA: 227.68.68.19		DA: 227.68.68.19
SP: 9876		SP: 9876
DP: 80		DP: 80

NAT (3): Benefits

- No TCP processing by NAT
  - no TCP stack handling TCP buffers, congestion windows, connection establishment, etc.
- All the packet manipulation takes place by special purpose device

Stickiness in load balancing

- Cookies are often used to maintain session state, e.g., shopping cart
  - Cookie header line identifies user to server
  - Server maintains a file about user
- Desirable to send same user to same server in server farm

NAT problem:

- Load balancing taken when SYN segment arrives
- SYN segment does not include HTTP header; thus no cookie header

Surrogate solution:

- Surrogate terminates TCP connection and receives HTTP request before choosing server

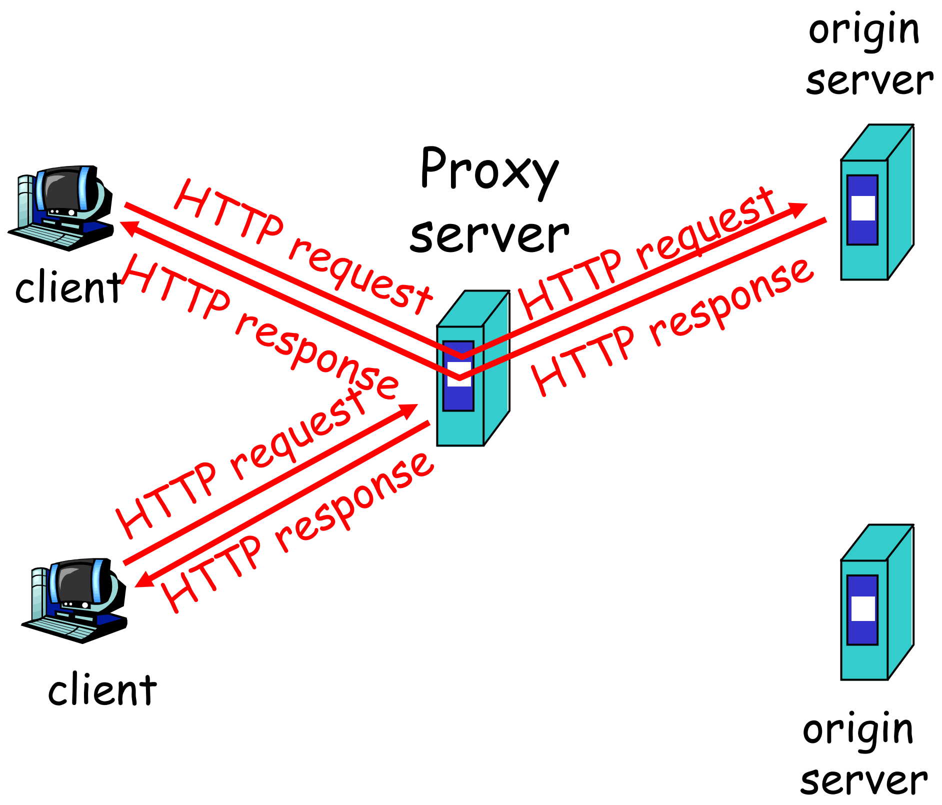
Content Distribution

- Server Farms
- Proxy Web Caches
- Content Distribution Networks (CDNs)
- Peer-to-peer file sharing (P2P)

Web caches (proxy server)

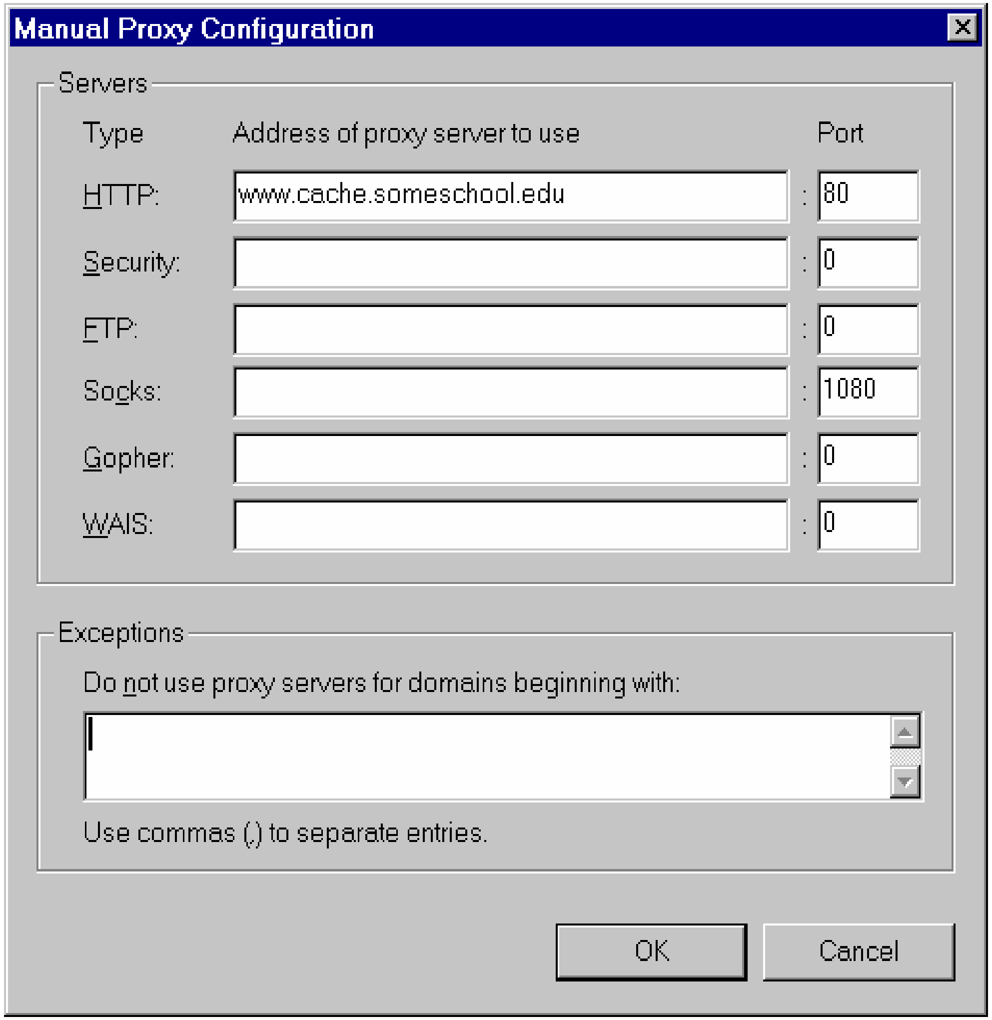
Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client



Configuring a Web Browser

In Netscape, go to Edit, Preferences, Advanced, Proxies:



More about Web caching

- Cache acts as both client and server
- Cache can do up-to-date check using `If-modified-since` HTTP header
  - Issue: should cache take risk and deliver cached object without checking?
  - Heuristics are used.
- Typically cache is installed by ISP (university, company, residential ISP)

Why Web caching?

- Reduce response time for client request.
- Reduce traffic on an institution's access link.
- Internet dense with caches enables "poor" content providers to effectively deliver content

Caching example (1)

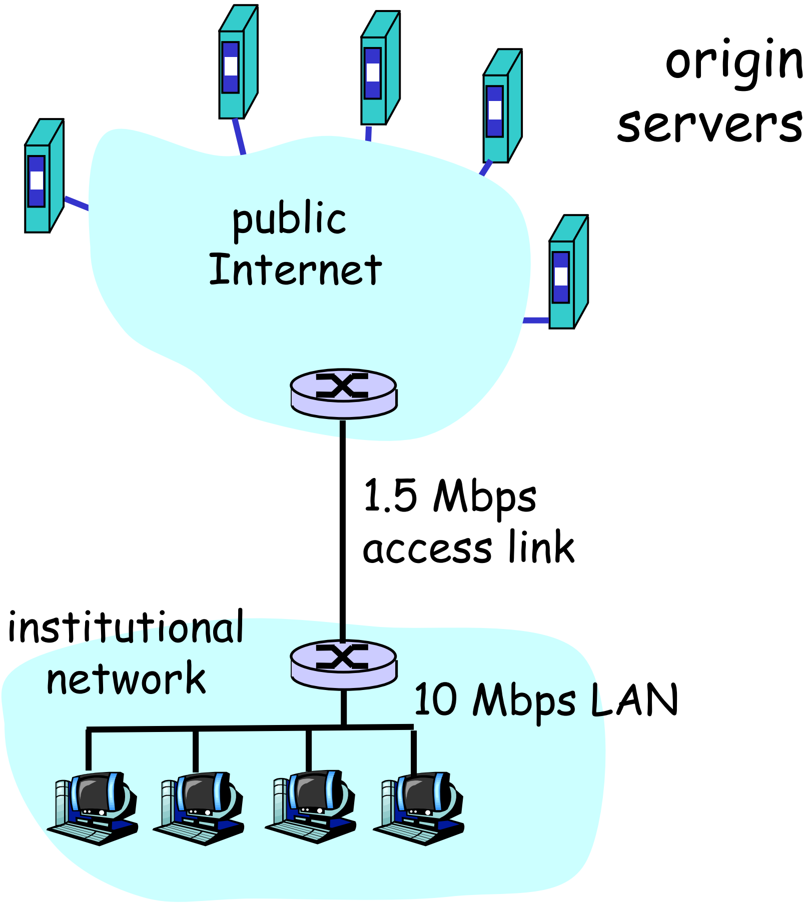
Assumptions

- average object size = 100,000 bits
- avg. request rate from institution's browser to origin serves = 15/sec
- delay from institutional router to any origin server and back to router = 2 sec

Consequences

- utilization on LAN = 15%
- utilization on access link = 100%
- total delay = Internet delay + access delay + LAN delay

= 2 sec + minutes + milliseconds



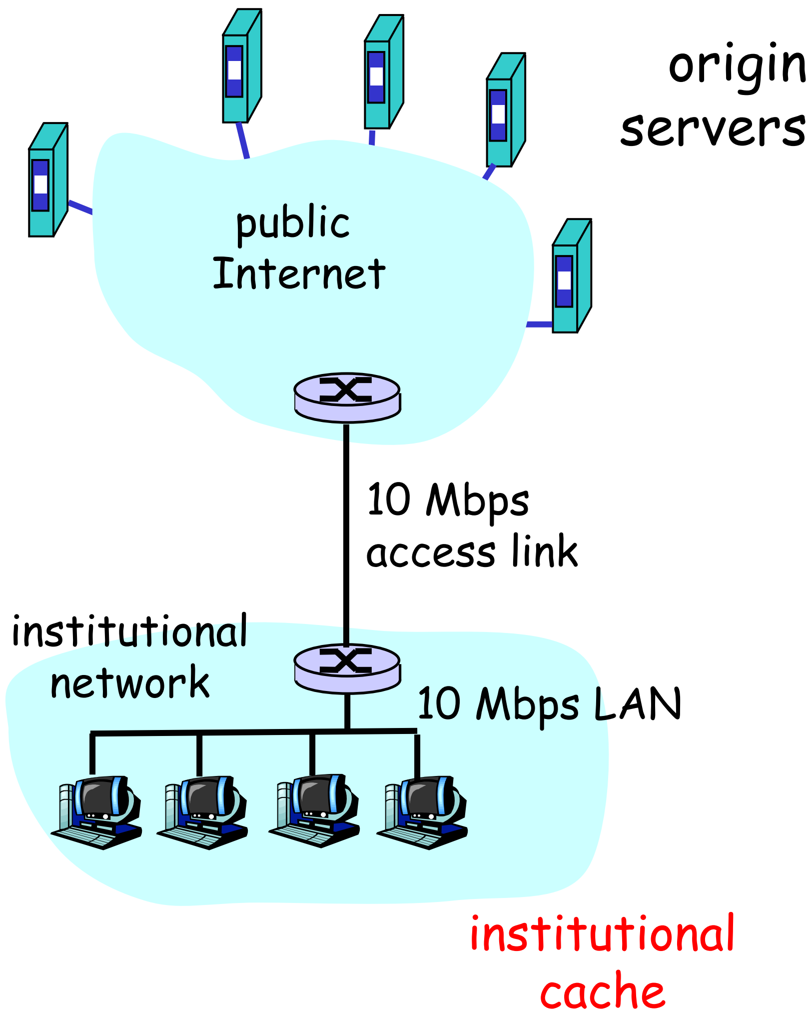
Caching example (2)

Possible solution

- increase bandwidth of access link to, say, 10 Mbps

Consequences

- utilization on LAN = 15%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay
  = 2 sec + msecs + msecs
- often a costly upgrade



Caching example (3)

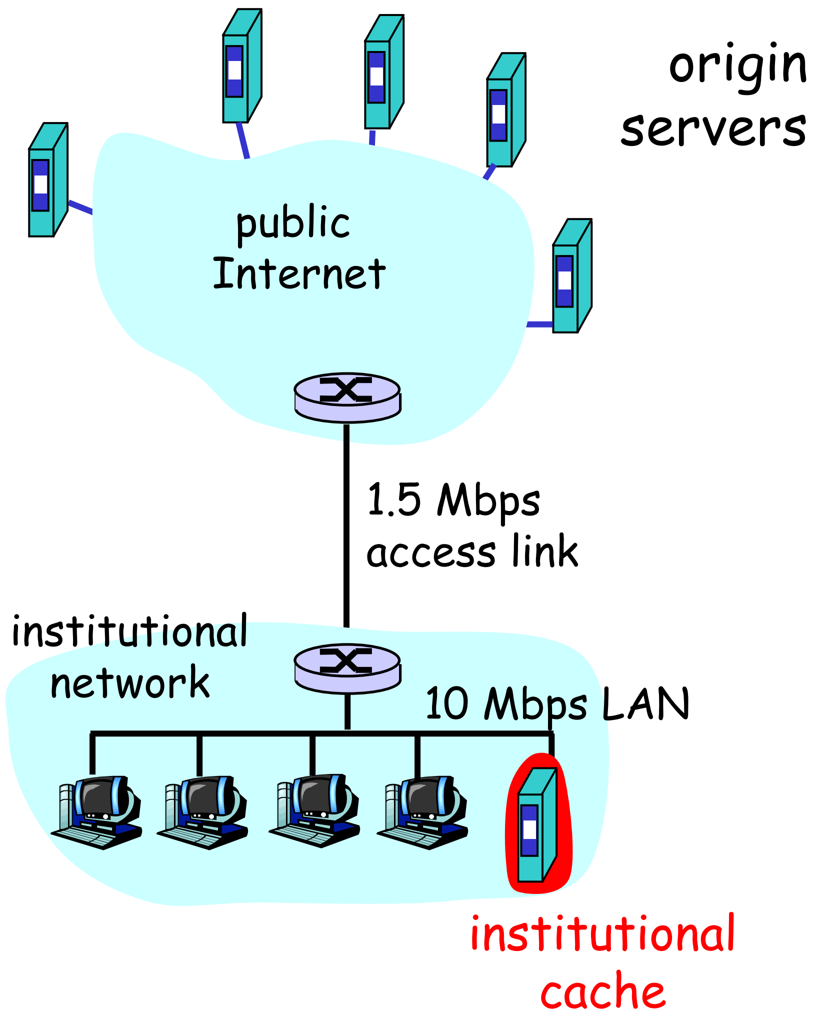
Install cache

- suppose hit rate is .4

Consequence

- 40% requests will be satisfied almost immediately
- 60% requests satisfied by origin server
- utilization of access link reduced to 60%, resulting in negligible delays (say 10 msec)
- total delay = Internet delay + access delay + LAN delay

= .6*2 sec + .6*.01 secs + milliseconds < 1.3 secs



Caching Challenges

- Cache consistency:
  - Caches often must guess whether a stored object is stale or fresh.
- Dynamic content:
  - Caches shouldn't cache outputs of CGI scripts.
- Hit counts and personalization:
  - Caches can cause hit count calculations and cookie transactions to fail.
- Less-savvy users and privacy-concerned users:
  - How do you get a user to point his browser to a cache?
- Enormous multimedia files:
  - Fortunately, disk storage is increasing at a rate of 60% a year!

Cache Storage Management

- LRU (Least Recently Used): Remove objects that have not been accessed for a long time. Example:
  - $value_{LRU}$ = (50 - days_since_accessed)/50
  - purge objects which have small $value_{LRU}$ when disk space starts to fill.
- Weighted by retrieval time:
  - record transfer time of object: retrieval_time
  - value = $value_{LRU}$ * log (retrieval_time +1)
- Weighted by object size:
  - value = $value_{LRU}$ * log (size +1)
- Purge documents that have expired:
  - But only if space is tight, as an up-to-date check can make an expired document fresh.

Transparent Caching

Non-transparent caching

- Each browser in an ISP is manually configured to point to the correct cache.
  - All AOL browsers
  - System admin can configure campus browsers
- But not when system administrator has no control over configuration parameters of browsers.

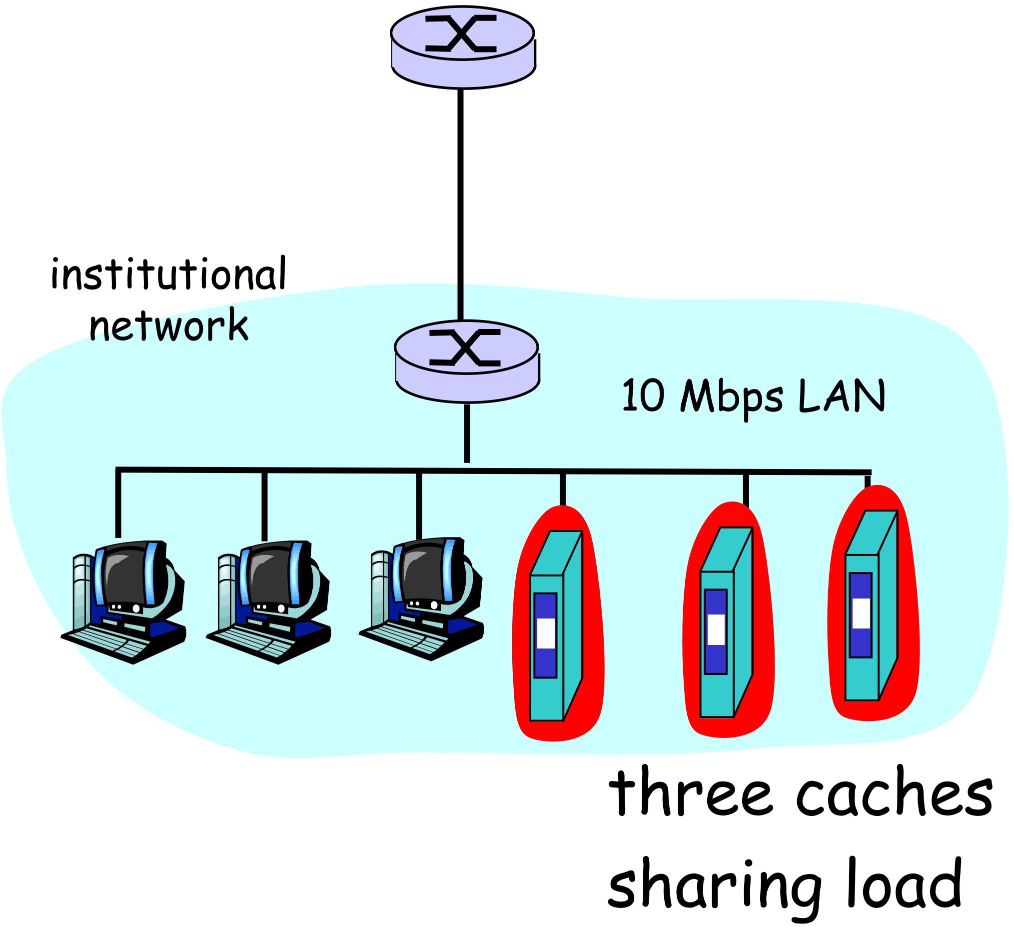
Transparent caching

- ISP redirects all datagrams with port number 80 (HTTP) to cache: "layer-4 switching".
- Cache either retreives object from its cache or fetches document from Internet.
- In the IP datagrams that carry the HTTP response message, cache must insert origin server's IP address!

Hash Routing

- Load balancing issue
- Overview of hash routing
- Robust hashing
- Heterogeneous caches
- CARP
- Proxy Automatic Configuration

Load balancing to multiple caches



Hash Routing Overview

- Choose a hash function h() which maps URLs to a hash space.
  - Example:
    - hash space is {1,…,60}.
    - h() is the sum of the ASCII representation of the characters in the URL, modulo 60.
- Partition hash space: one set for each cache.
  1) Client hashes URL, determines set to which hashed URL belongs, and sends request to corresponding cache.
     - Example: N=2 caches, set for cache 1 = {1,..,30}, set for cache 2 = {31,..,60}. If h(URLa) = 35, then client sends HTTP request to cache 2.
  2) If cache does not have object, it obtains object from origin server, stores a copy, and forwards a copy to the client.
- Each object resides in at most one cache!
- Client is immediately directed to the correct cache.
- Do not pass through surrogate or NAT

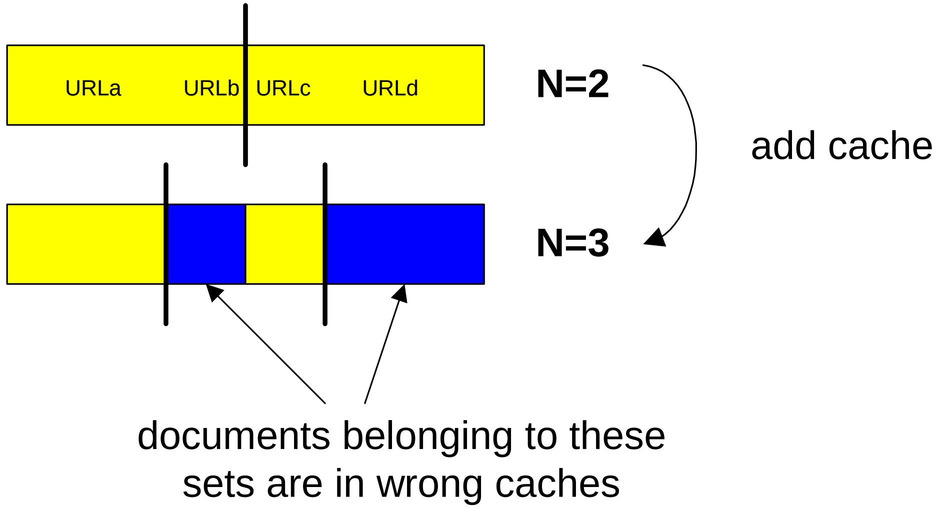
Hash Routing: Robustness Problem

When a cache is added or removed, a cached object can reside in the wrong cache.

Example: hash space = {1,2,..,60}
h(URLa) = 10, h(URLb) = 25, h(URLc) = 35, h(URLd) = 50

When there are N=2, URL space partitioned into {1,…,30}, {31,…,60}.

URL space:



- When we add a third cache, URL space is partitioned into {1,..,20}, {21,..,40}, {41,..,60}.
- Request for URLb is directed to cache 2; but URLb is in cache 1 so there is a MISS.
- Cache 2 will get URLb from origin server, so both caches 1 and 2 will contain URLb.

Disruption Coefficient

D= Disruption Coefficient = fraction of objects in incorrect cache after adding or deleting a cache.

Without loss of generality, suppose hash space is set continuous interval [0,1] . Suppose there are N caches. The set [0,1] is partitioned:

$$[0,\frac{1}{N}],[\frac{1}{N},\frac{2}{N}],...,[\frac{N-1}{N},1]$$

Now suppose we add sibling N+1. The new partition is:

$$[0,\frac{1}{N+1}],[\frac{1}{N+1},\frac{2}{N+1}],...,[\frac{N}{N+1},1]$$

URLs in $[0,\frac{1}{N+1}]$ are in cache 1, which is correct. But URLs in $[\frac{1}{N+1},\frac{1}{N}]$ are in cache 1 when they should be in cache 2. Good intervals:

$$[0,\frac{1}{N+1}],[\frac{1}{N},\frac{2}{N+1}],[\frac{2}{N},\frac{3}{N+1}],...,[\frac{N-1}{N},\frac{N}{N+1}]$$

Sum of intervals is D=.5. Same result for deleting a cache.

Thus hit rate is cut in half after a disruption: Not good!

Robust Hashing

- Assign name to each cache: Joseph, Richard, Mary, Jane.
- Choose hash function h(u,s) which is a function of both the URL u and the cache name s.
  - for example, use h(u+s), where h() is a standard hash function and u+s is the string concatenation of u and s.
- When a client wants URL u, client calculates the "scores" $h(u,s_1),\ldots,h(u,s_N)$ for each of the N caches.
- Client directs request for URL u to the cache s that has the highest score.

➔ If a cache fails, all remaining objects are still were they are supposed to be. Disruption coefficient is 1/N, which is typically small.

➔ If a cache is added, roughly the fraction D = 1/(N+1) objects are in the wrong place.

Heterogeneous Siblings

- Processing power and storage capacity can vary greatly among siblings.
- Target probabilities: $p_1, p_2, \ldots, p_N$
- Introduce multipliers: $x_1, x_2, \ldots, x_N$
  (1) Calculate hash $h(u, s_n)$
  (2) Route URL u to sibling with highest weighted score

$$Z_n = x_n h(u, s_n)$$

- How do we pick multipliers $x_1, x_2, \ldots, x_N$ to achieve target probabilities: $p_1, p_2, \ldots, p_N$

EURECOM

Multipliers

$$x_1 = (Np_1)^{1/N}$$

$$x_n = \left[\frac{(N-n+1)(p_n - p_{n-1})}{\prod_{i=1}^{n-1} x_i} + x_{n-1}^{N-n+1} \right]^{\frac{1}{N-n+1}}$$

Example: $N = 3,\ p_1 = p_2 = \frac{1}{81},\ p_3 = \frac{79}{81}$

$$x_1 = x_2 = \frac{1}{3}, \quad x_3 = 9$$

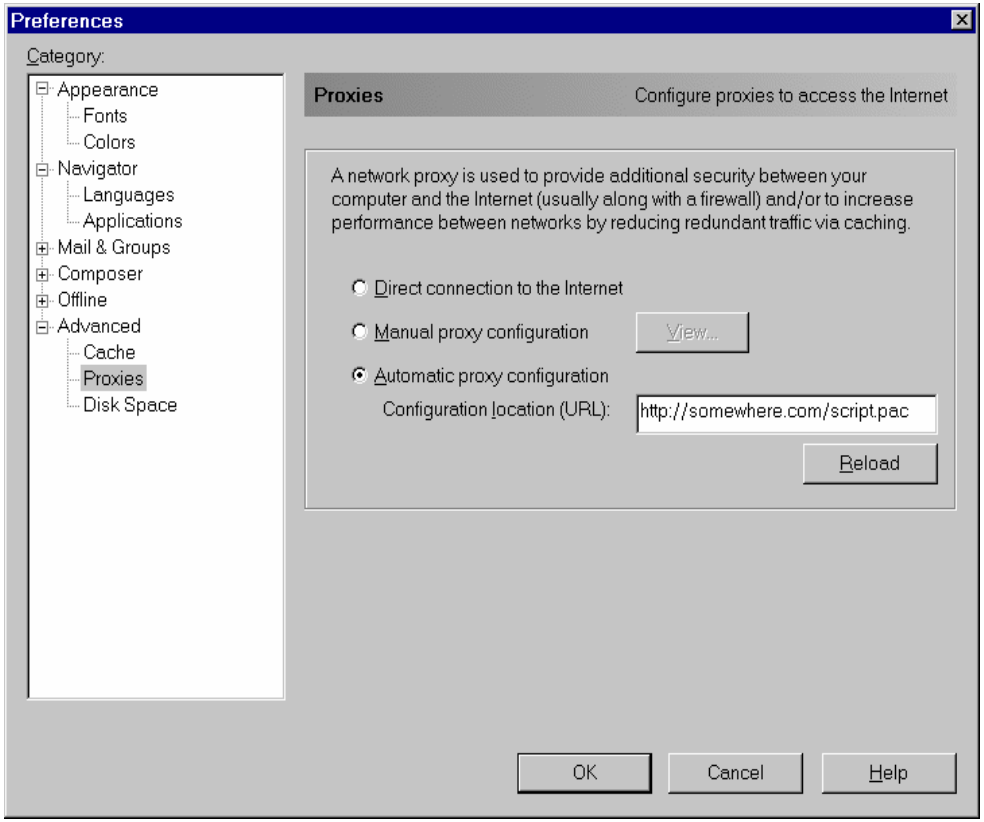
EURECOM

CARP (Cache Array Routing Protocol)

- Uses robust hash routing with multipliers.
- All queries done over HTTP
  - no new application-layer protocol such as ICP
  - can take advantage of HTTP/1.1's rich set of headers
- Internet draft (Valloppillil and Ross)
- Implemented in Microsoft and Netscape cache server products

Proxy Automatic Configuration (PAC)

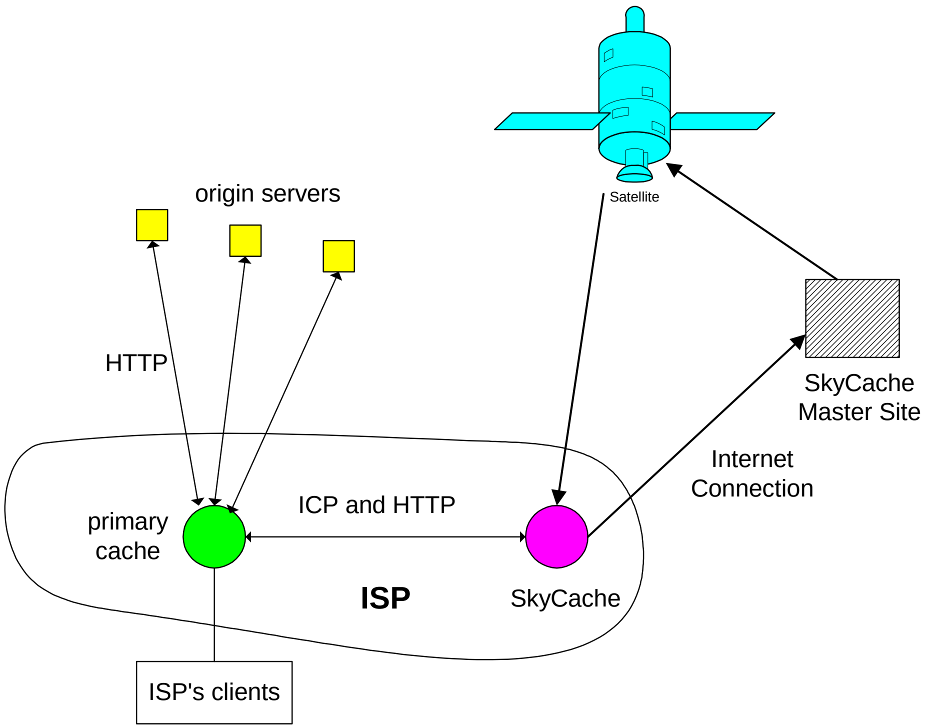
- Allows client browser to dynamically choose among a set of caches.
  - Create an auto-configuration JavaScript file.
  - Put file on nearby Web server (e.g., http://somewhere.com/script.pac).
  - Have clients configure their browsers with the URL of the JavaScript.



Each time browser is initiated, browser obtains script.pac from nearby server. Script is run for each URL request.

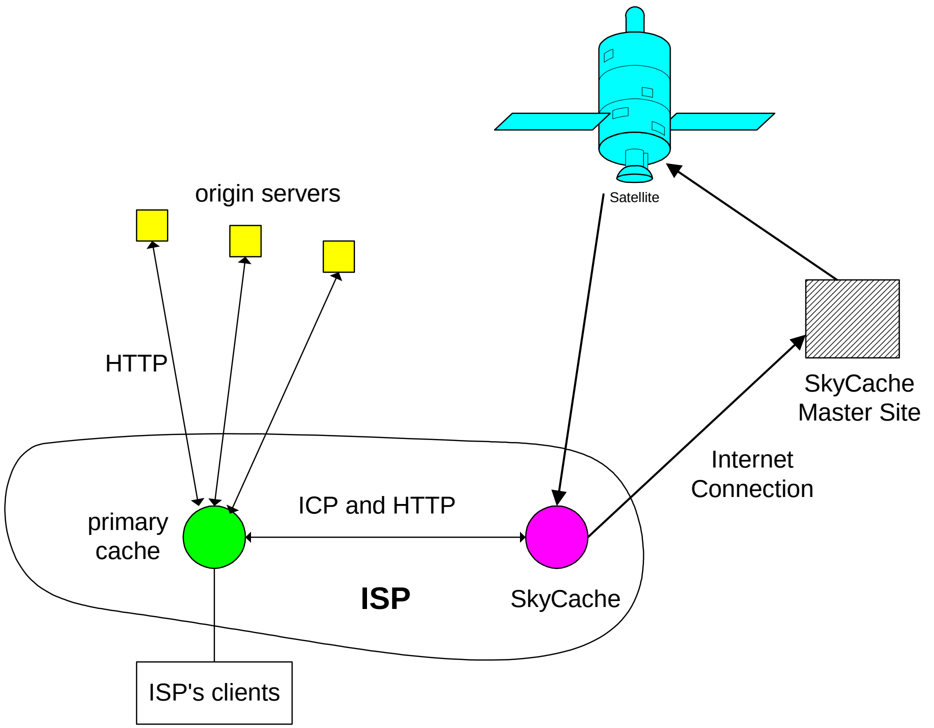
Satellite Technology

- Each local ISP has a cache with:
  - Internet connection
  - Huge storage capacity
  - Satellite dish for receiving
- Master site has:
  - Internet connection
  - Satellite transmitter



Satellite Technology

- How it works: When there is a miss at some local cache:
  - that local cache obtains document from origin server using HTTP.
  - local cache sends URL to master site.
  - master site obtains document from origin server using HTTP.
  - master site transmits document into satellite channel.
  - *all* local caches receive document and cache it.

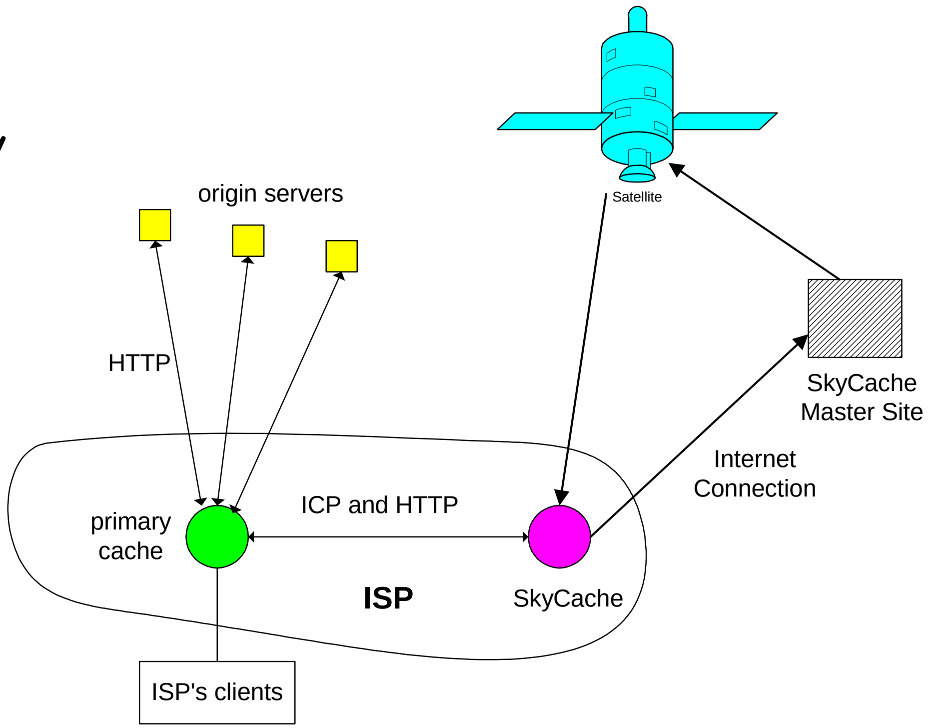


Satellite Technology: The Result

- The user populations at each of the local ISPs are aggregated together to form one huge user population.
  - The greater the user population, the greater the likelihood of repeated requests, the greater the hit rate.
- Brings the Web to the edge of the network.

A Product: SkyCache

- SkyCache: founded in 1997.
- Performance enhancer for an ISP's primary cache.
- Leases all equipment: fixed monthly payment to ISPs
- 4 Mbps one-way satellite link
- When there is a miss at primary cache:
  - primary cache queries SkyCache using ICP.
  - If SkyCache has the object, primary cache obtains object from SkyCache using HTTP.
  - If SkyCache doesn't have the object, primary cache directly obtains object from origin server.
  - SkyCache remembers the URLs for the misses and reports misses to Master Site.



Content Distribution

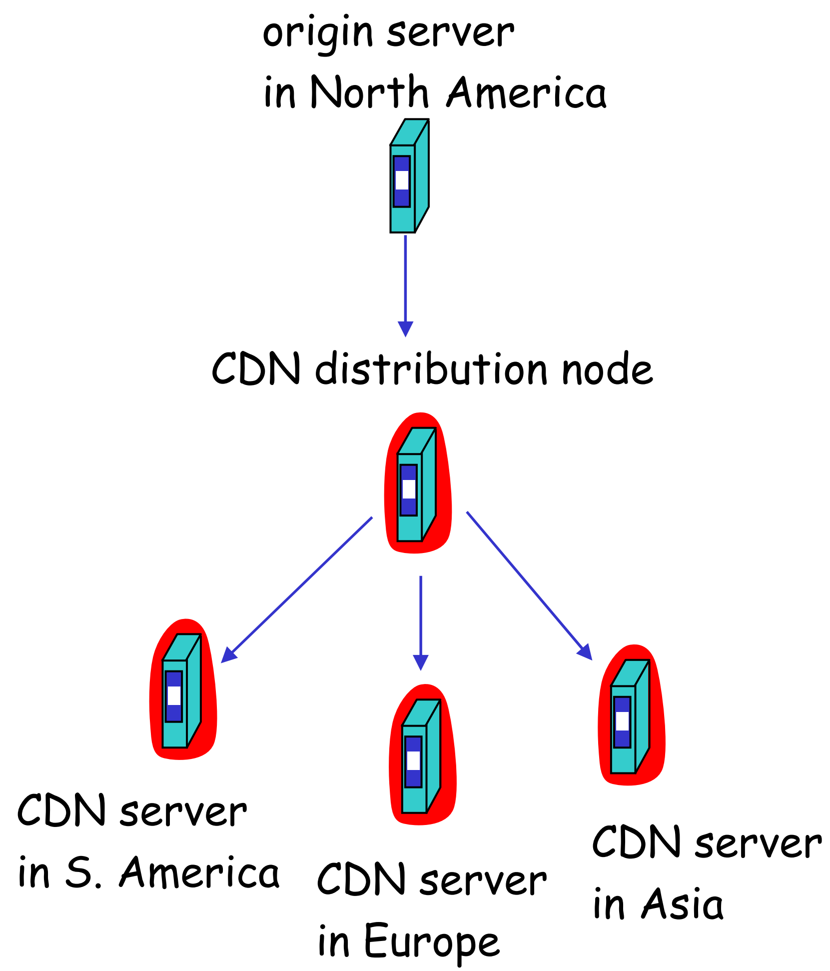
- Server Farms
- Proxy Web Caches
- Content Distribution Networks (CDNs)
- Peer-to-peer file sharing (P2P)

Content distribution networks (CDNs)

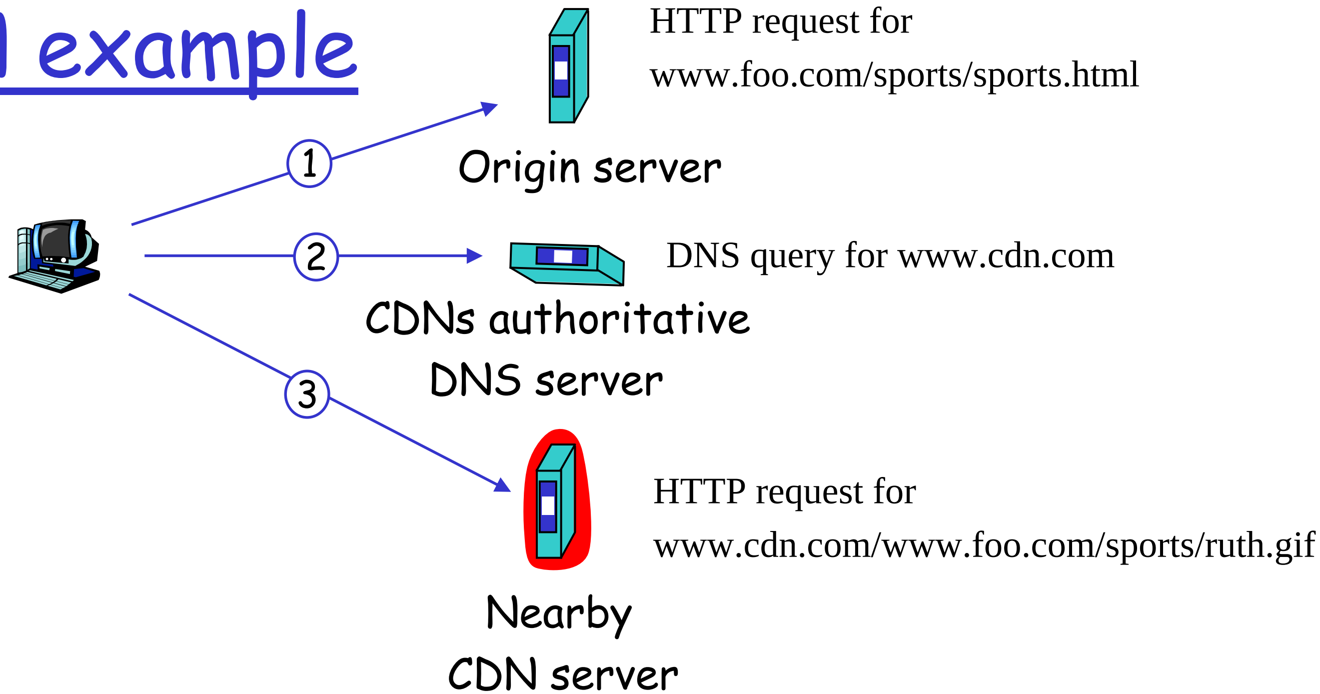
- The content providers are the CDN customers.

Content replication

- CDN company installs hundreds of CDN servers throughout Internet
  - in lower-tier ISPs, close to users
- CDN replicates its customers' content in CDN servers. When provider updates content, CDN updates servers



CDN example



origin server

- www.foo.com
- distributes HTML
- Replaces:

  http://www.foo.com/sports.ruth.gif

  with

  http://www.cdn.com/www.foo.com/sports/ruth.gif

CDN company

- cdn.com
- distributes gif files
- uses its authoritative DNS server to route redirect requests

More about CDNs

routing requests

- CDN creates a "map", indicating distances from leaf ISPs and CDN nodes
- when query arrives at authoritative DNS server:
  - server determines ISP from which query originates
  - uses "map" to determine best CDN server

not just Web pages

- streaming stored audio/video
- streaming real-time audio/video
  - CDN nodes create application-layer overlay network

Content Distribution

- Server farms
- Proxy Web Caches
- Content Distribution Networks (CDNs)
- Peer-to-peer file sharing (P2P)

P2P file sharing

Example

- Alice runs P2P client application on her notebook computer
- Intermittently connects to Internet; gets new IP address for each connection
- Asks for "Hey Jude"
- Application displays other peers that have copy of Hey Jude.
- Alice chooses one of the peers, Bob.
- File is copied from Bob's PC to Alice's notebook: HTTP
- While Alice downloads, other users uploading from Alice.
- Alice's peer is both a Web client and a transient Web server.

All peers are servers = highly scalable!

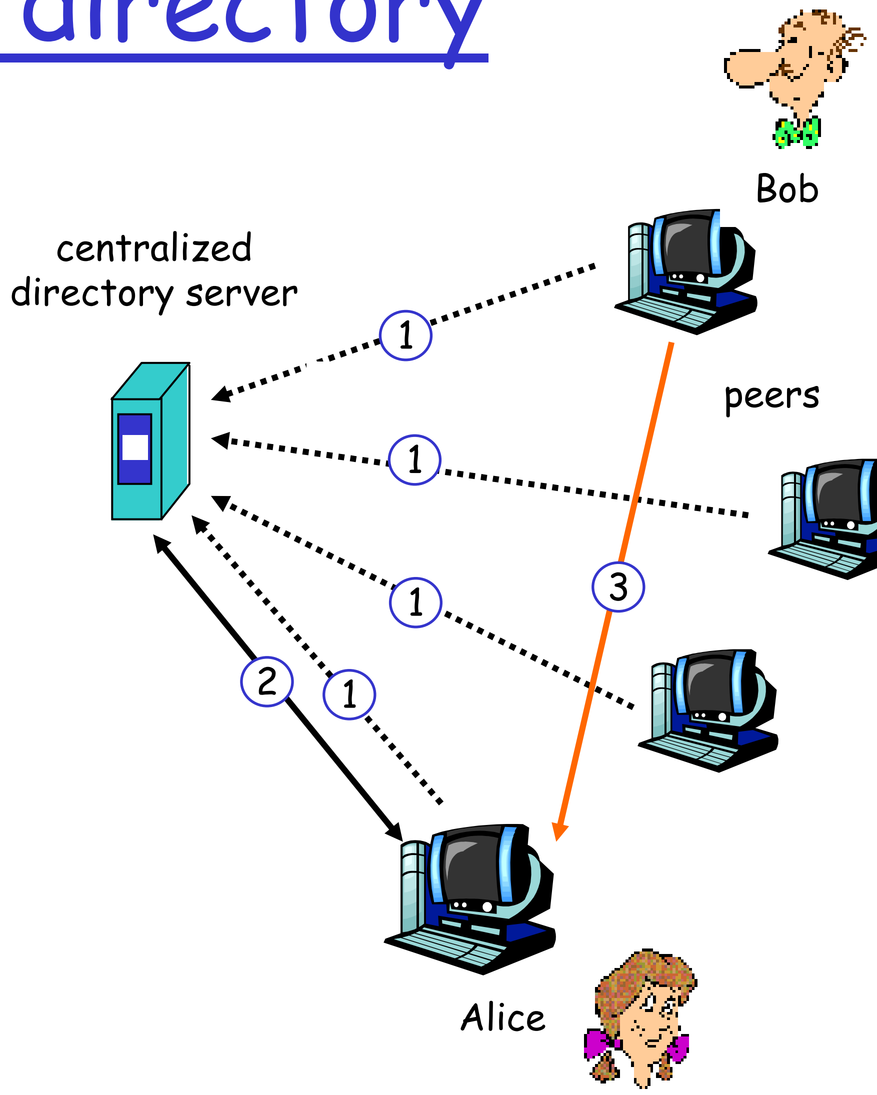
P2P: centralized directory

original "Napster" design

1) when peer connects, it informs central server:
   - IP address
   - content

2) Alice queries for "Hey Jude"

3) Alice requests file from Bob



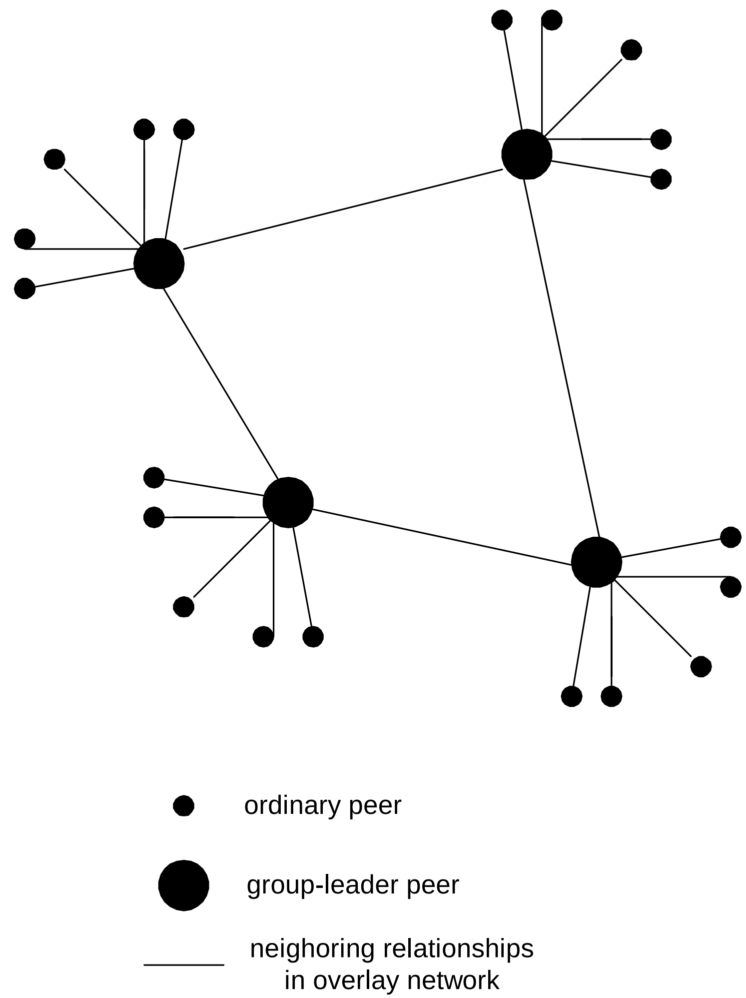
P2P: problems with centralized directory

- Single point of failure
- Performance bottleneck
- Copyright infringement

file transfer is decentralized, but locating content is highly decentralized

P2P: decentralized directory

- Each peer is either a group leader or assigned to a group leader.
- Group leader tracks the content in all its children.
- Peer queries group leader; group leader may query other group leaders.



More about decentralized directory

overlay network

- peers are nodes
- edges between peers and their group leaders
- edges between some pairs of group leaders
- virtual neighbors

bootstrap node

- connecting peer is either assigned to a group leader or designated as leader

advantages of approach

- no centralized directory server
  - location service distributed over peers
  - more difficult to shut down

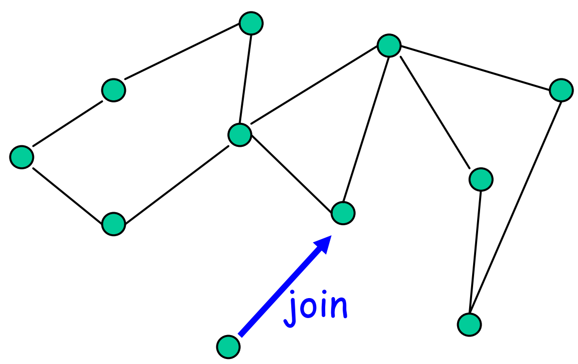
disadvantages of approach

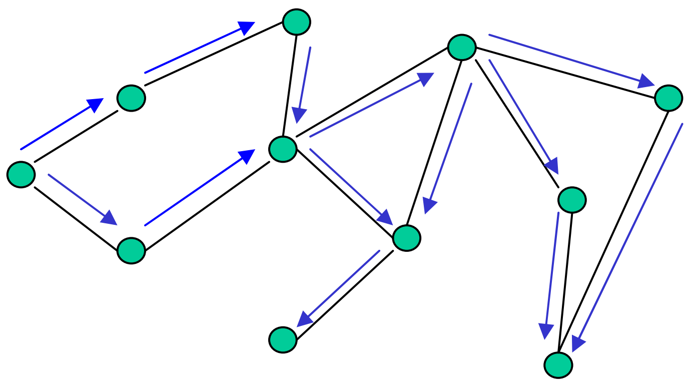
- bootstrap node needed
- group leaders can get overloaded

P2P: Query flooding

- Gnutella
- no hierarchy
- use bootstrap node to learn about others
- join message

- Send query to neighbors
- Neighbors forward query
- If queried peer has object, it sends message back to querying peer





P2P: more on query flooding

Pros

- peers have similar responsibilities: no group leaders
- highly decentralized
- no peer maintains directory info

Cons

- excessive query traffic
- query radius: may not have content when present
- bootstrap node
- maintenance of overlay network