CSC 525:
Computer Networks

World Wide Web

- Invented by Tim Berners-Lee in 1989 at CERN.
  - Now standardized by by the world wide web consortium (W3C)
- It has three essential components:
  - **URL**: a naming scheme to identify content objects.
  - **HTML**: a markup language to annotate documents with hyperlinks (pointing to content objects).
  - **HTTP**: an application-layer protocol to transfer content objects.
- Web servers (e.g., apache, nginx)
  - Process requests, dynamically generate contents, and serve them.
- Web clients (e.g., Safari, Chrome, Firefox)
  - Retrieve and display contents, may run embedded program locally.

URL

- A web page consists of multiple objects
- An object can be an HTML file, JPEG image, Java applet, audio, video, ...
- A web page has the base HTML file which includes references to other objects.
- Each object is addressable by a URL

- **scheme:**[**//**[**user**[**:password**]**@**]**host**[**:port**]][**/path**][?**query**][#**fragment**]

➢ http://www.cs.arizona.edu/classes/cs425/schedule.htm

➢ http://www.cs.arizona.edu:8080/index.htm

➢ http://bzhang:mypasswd@www.cs.arizona.edu/index.htm

➢ http://www.cs.arizona.edu/area.php?width=2in&length=3in

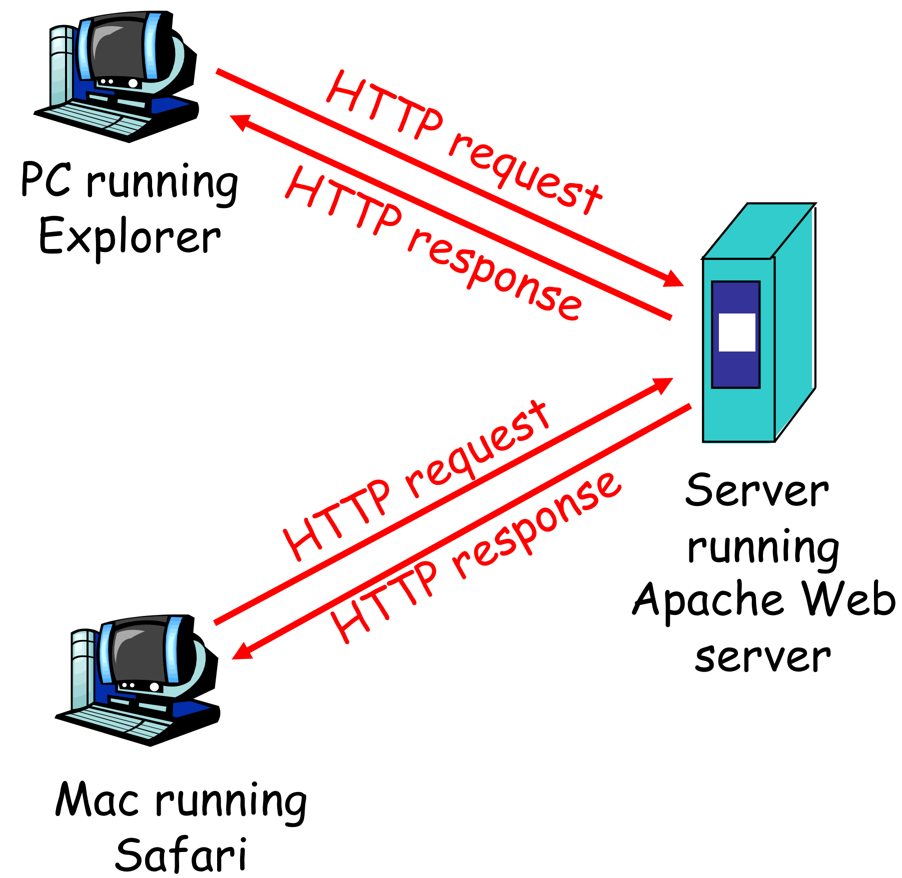
➢ Scheme can be protocols other than http. E.g., https, ftp, mailto, file, etc. They require the support from the client app.

Web Pages

- Three core technologies for modern web contents:
  - HTML, CSS, and JavaScript.
- HTML defines a number of *elements* delineated by *tags* to make up a document.
  - E.g., U of A
- CSS (style sheet) to describe presentation of the document
- JavaScript allows running programs in the browser to make the page interactive

HTTP overview

- request/response
  - *client:* browser requests, receives, displays Web objects
  - *server:* Web server sends objects in response to requests.
  - Default TCP port 80
  - Stateless: server maintains no information about past client requests
  - Can add states in the server software.
- HTTP 1.0 (RFC 1945), HTTP 1.1 (RFC 2068), HTTP 2 (RFC 7540 in 2015), and HTTP 3 proposed.



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

```

header lines

A blank line indicates end of header

Requests can carry payload too, e.g., submit a user-filled form.

Request Commands

Operation	Description
OPTIONS	Request information about available options
GET	Retrieve document identified in URL
HEAD	Retrieve metainformation about document identified in URL
POST	Give information (e.g., annotation) to server
PUT	Store document under specified URL
DELETE	Delete specified URL
TRACE	Loopback request message
CONNECT	For use by proxies

The most common ones are GET, POST and HEAD.

HTTP response message

status line
(protocol
status code
status phrase)

```
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 ...
```

header
lines

A blank line

data, e.g.,
requested
HTML file

HTTP response code

Code	Type	Example Reasons
1xx	Informational	request received, continuing process
2xx	Success	action successfully received, understood, and accepted
3xx	Redirection	further action must be taken to complete the request
4xx	Client Error	request contains bad syntax or cannot be fulfilled
5xx	Server Error	server failed to fulfill an apparently valid request

HTTP response status codes

In the first line in server $\Rightarrow$ client response message.

A few common ones :

`200 OK`

– request succeeded, requested object later in this message

`301 Moved Permanently`

– requested object moved, new location specified later in this message (see the "Location" header)

`400 Bad Request`

– request message not understood by server

`404 Not Found`

– requested document not found on this server

`505 HTTP Version Not Supported`

Trying out HTTP for yourself

1. telnet to a web server:

```
telnet cis.poly.edu 80
```

Opens TCP connection to port 80
(default HTTP server port) at cis.poly.edu.
Anything typed in sent
to port 80 at cis.poly.edu

2. Type in a GET HTTP request:

```
GET /~ross/ HTTP/1.1
Host: cis.poly.edu
```

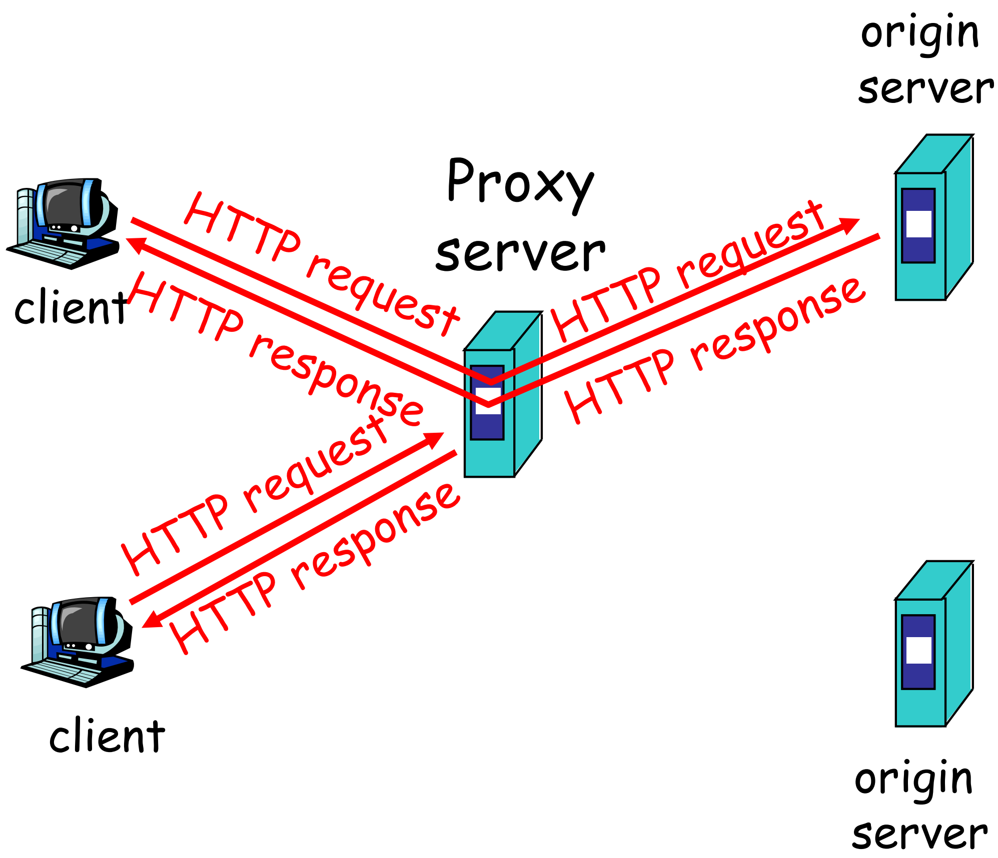
By typing this in (hit carriage
return twice), you send
this minimal (but complete)
GET request to HTTP server

3. Look at response message sent by HTTP server!

Web Proxy

Going through a web proxy before reaching the origin server.

- user configures browser with proxy's address.
- browser sends all HTTP requests to proxy, which then forwards to the origin server.
  - Two separate HTTP/TCP connections.
  - Proxy can examine the traffic and take actions, e.g., blocking access, collect statistics, provide caching.



Improving web performance

- Server side processing
  - High throughput measured by #requests/second: need to respond to each request fast
  - High concurrency: need to schedule every request fast.
- Client side
  - Rendering page fast, executing JavaScript fast.
  - Pipelining or using concurrent connections to download objects
- HTTP protocol
  - Support persistent connection with pipelining requests
  - Support caching
- Infrastructure
  - Scaling the service globally
  - The Content Distribution Networks (CDNs)

Non-Persistent vs. Persistent Connection

Nonpersistent HTTP issues:

- One connection for each obj
- takes 2 RTTs to retrieve it
- OS must work and allocate host resources for each TCP connection
- browsers often open parallel TCP connections to get multiple objects.

Persistent HTTP

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server are sent over the same connection

Persistent without pipelining:

- client issues new request only when previous response has been received
- one RTT for each referenced object

Persistent with pipelining:

- default in HTTP/1.1
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects if enough bandwidth and no packet loss.

Web Caching

- Cache and serve web objects from proxies.
  - One of the most important techniques to scale the web
- Caching offers many benefits.
  - Clients experience shorter delay
  - Origin server gets reduced workload.
  - Network sees less traffic.
- Caching can be done at different places.
  - Client, proxy, server.
- Caching can be transparent or non-transparent to the client.
  - Whether the user is aware of it, i.e., need to do configuration or not.

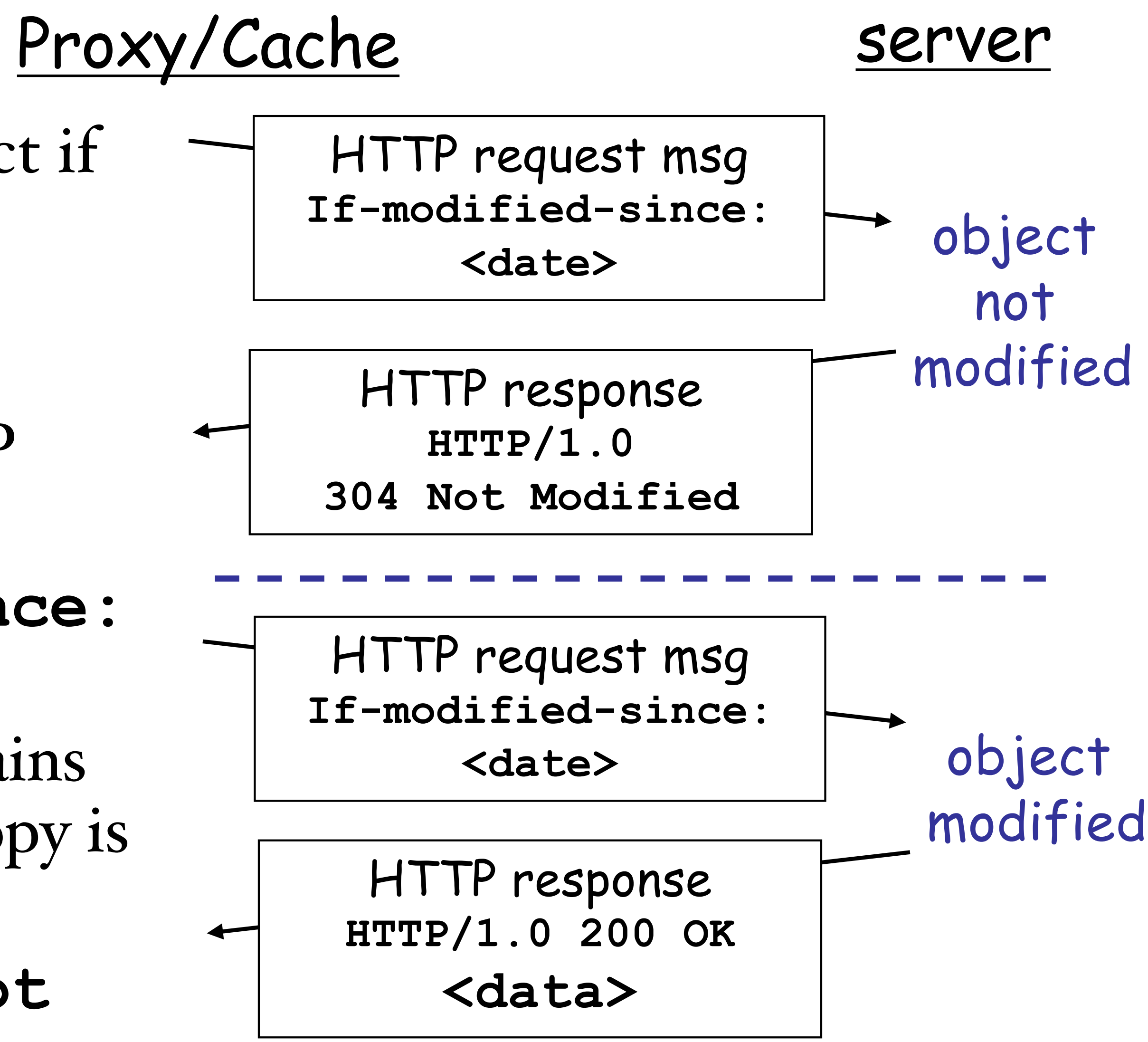
Supporting caching: conditional GET

- Goal: don't send object if proxy has up-to-date cached version
- proxy: specify date of cached copy in HTTP request

  `If-modified-since: <date>`

- server: response contains no object if cached copy is up-to-date:

  `HTTP/1.0 304 Not Modified`

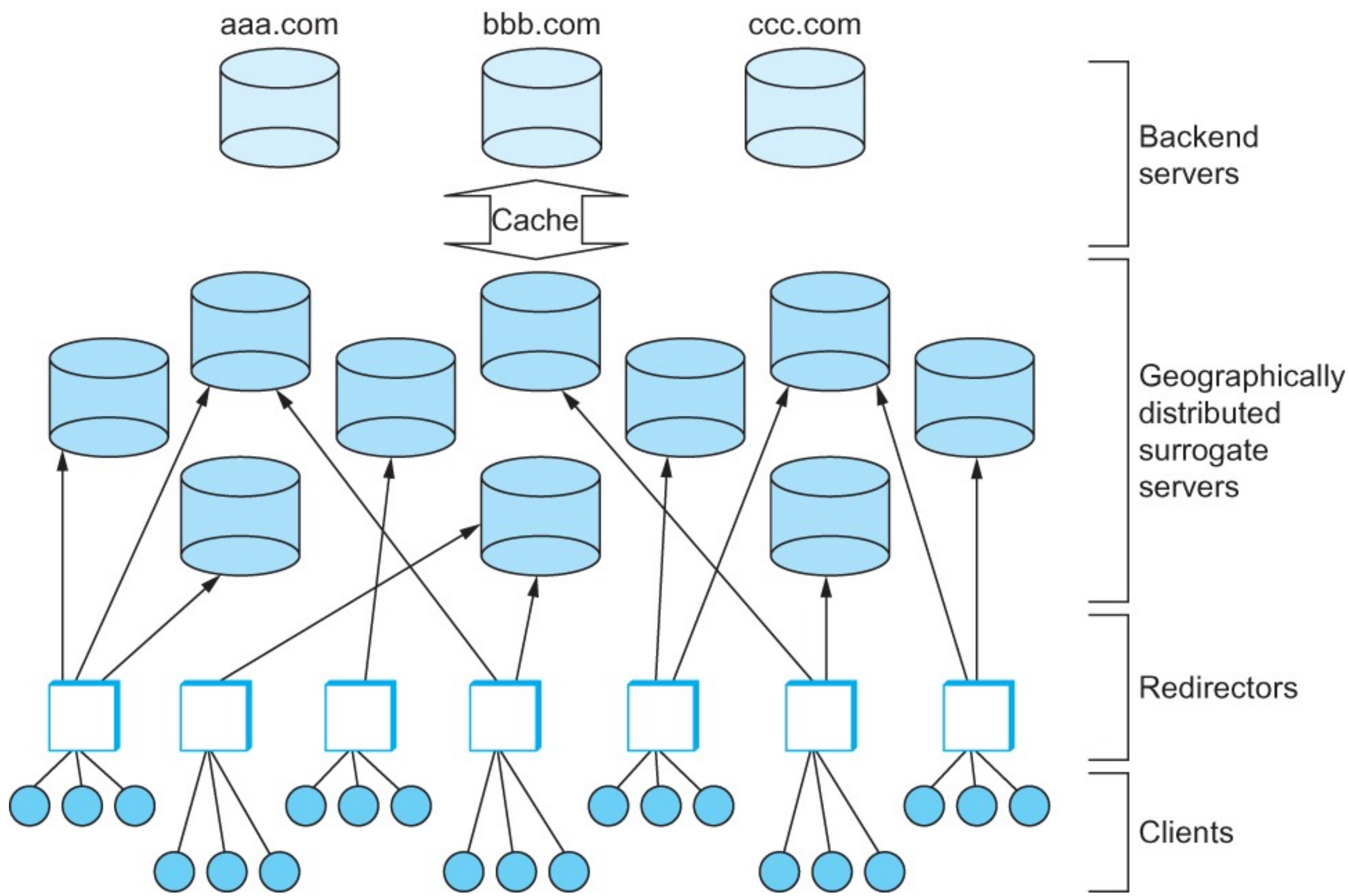


Content Distribution Network (CDN)

- Content providers, ISPs, and end users.
- CDN is another party to scale content distribution.
  - The idea is to geographically (in different ISPs) distribute a collection of *server surrogates* (owned and operated by CDNs) that cache contents (provided by the content owners) normally maintained in some set of *backend servers*.
  - CDN companies: Akamai, Limelight, …
- Spread load to many surrogate servers
  - E.g., when a big news breaks or a populate event happens.
- Shorter delay perceived by users.
- Less cost for ISPs as well.
- Absorb denial-of-service attack traffic.

CDN mechanisms

- Request redirection may use different techniques, e.g., DNS, URL rewriting, HTTP redirect.
- Caching and management of the contents.



YouTube

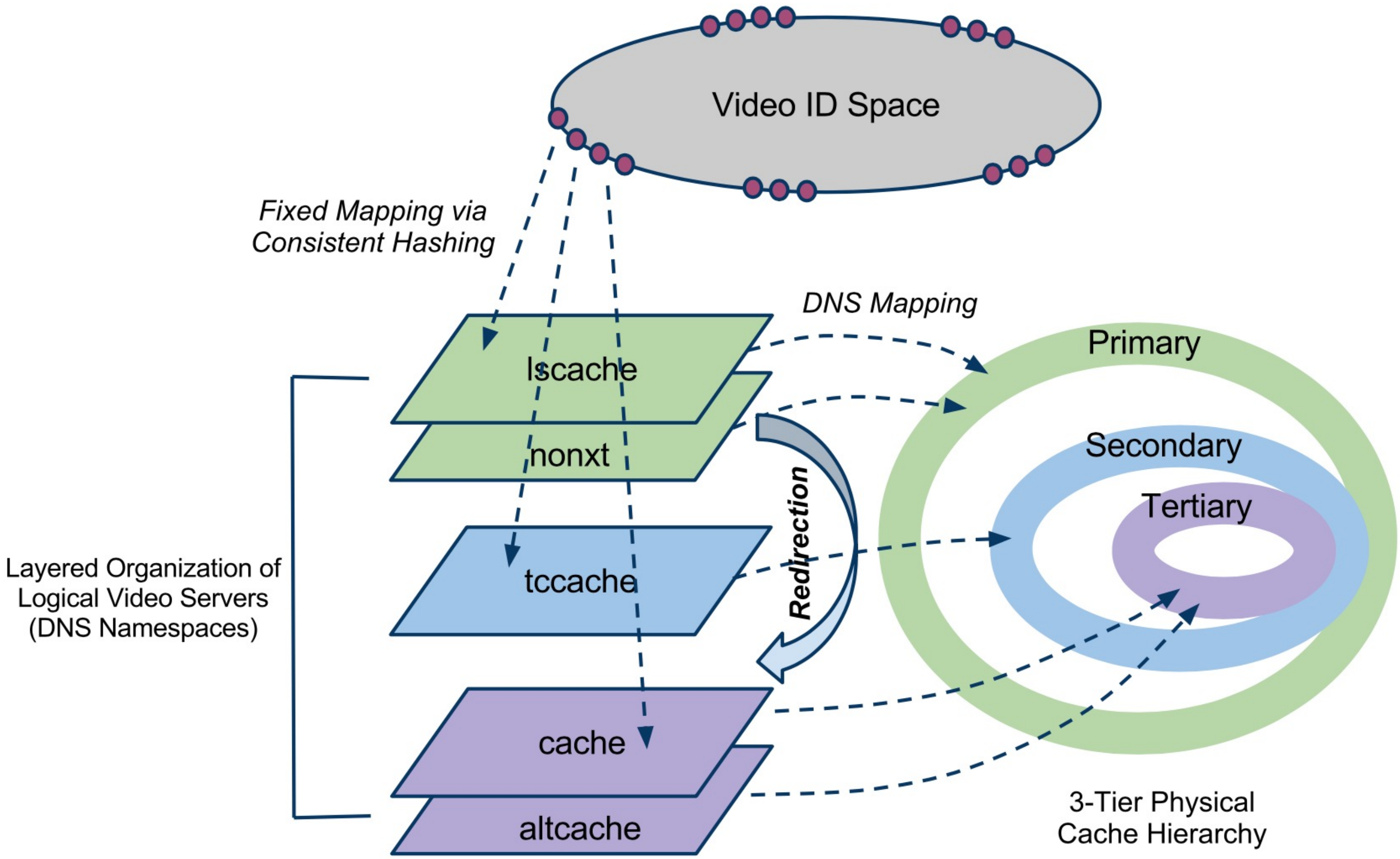
- Before Google: use 3rd party CDNs.

- After Google: integrated into Google's network infrastructure and use a hierarchy of cache servers.

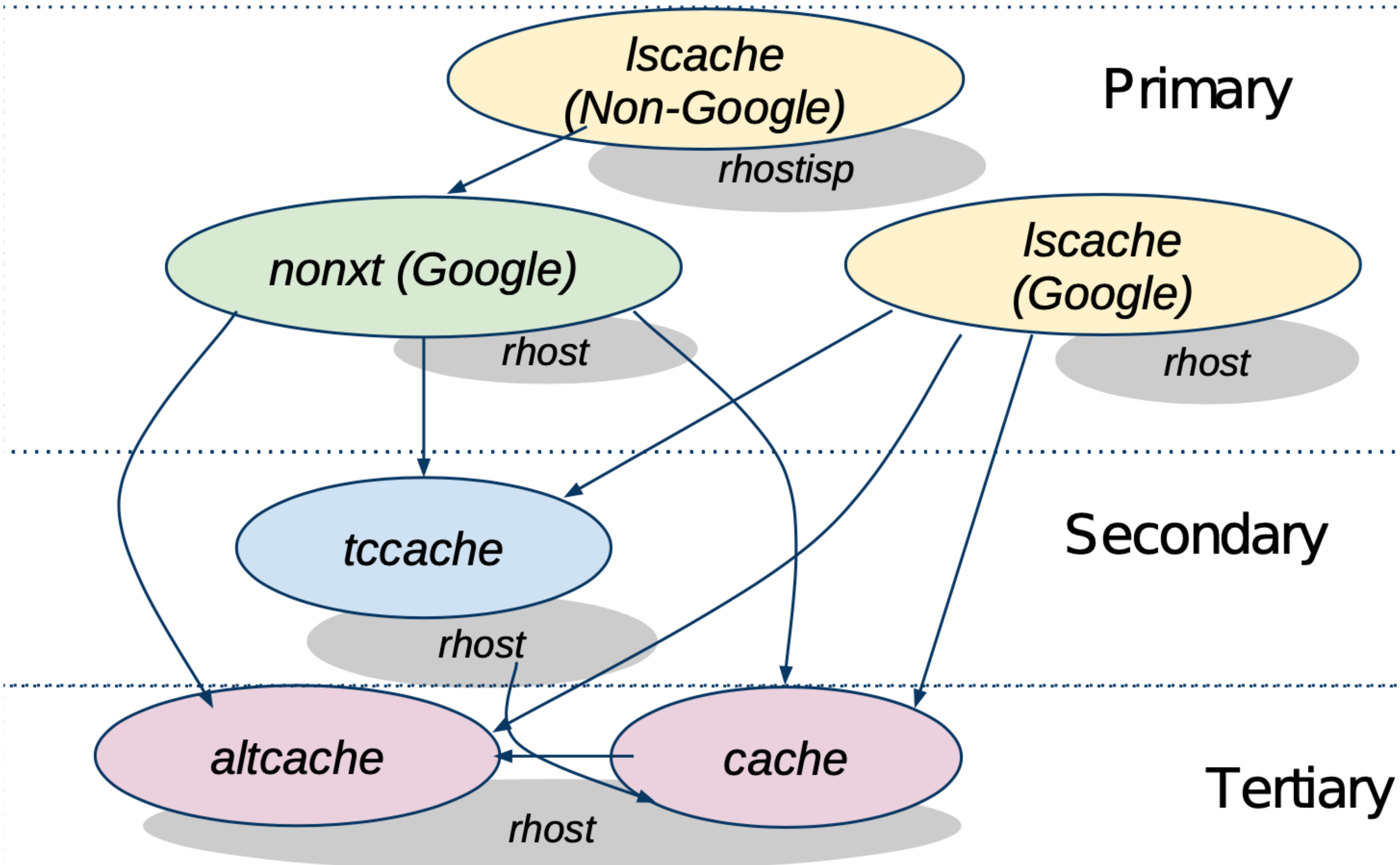
Probing YouTube

- Research resources:
  - PlanetLab nodes: 471
  - Open DNS resolvers: 843
  - A custom video downloader/player
- Steps:
  - Crawl YouTube pages to collect video URLs.
    - Including both popular and unpopular ones.
  - Download and play the videos.
    - Recording HTTP redirection sequences.
  - Measure delays.

YouTube Architecture Overview



- video ID: 11 letters/numbers evenly distributed
- hierarchical cache servers.


lscache
(Non-Google)
rhostisp
Primary
nonxt (Google)
rhost
lscache
(Google)
rhost
tccache
rhost
Secondary
altcache
cache
rhost
Tertiary

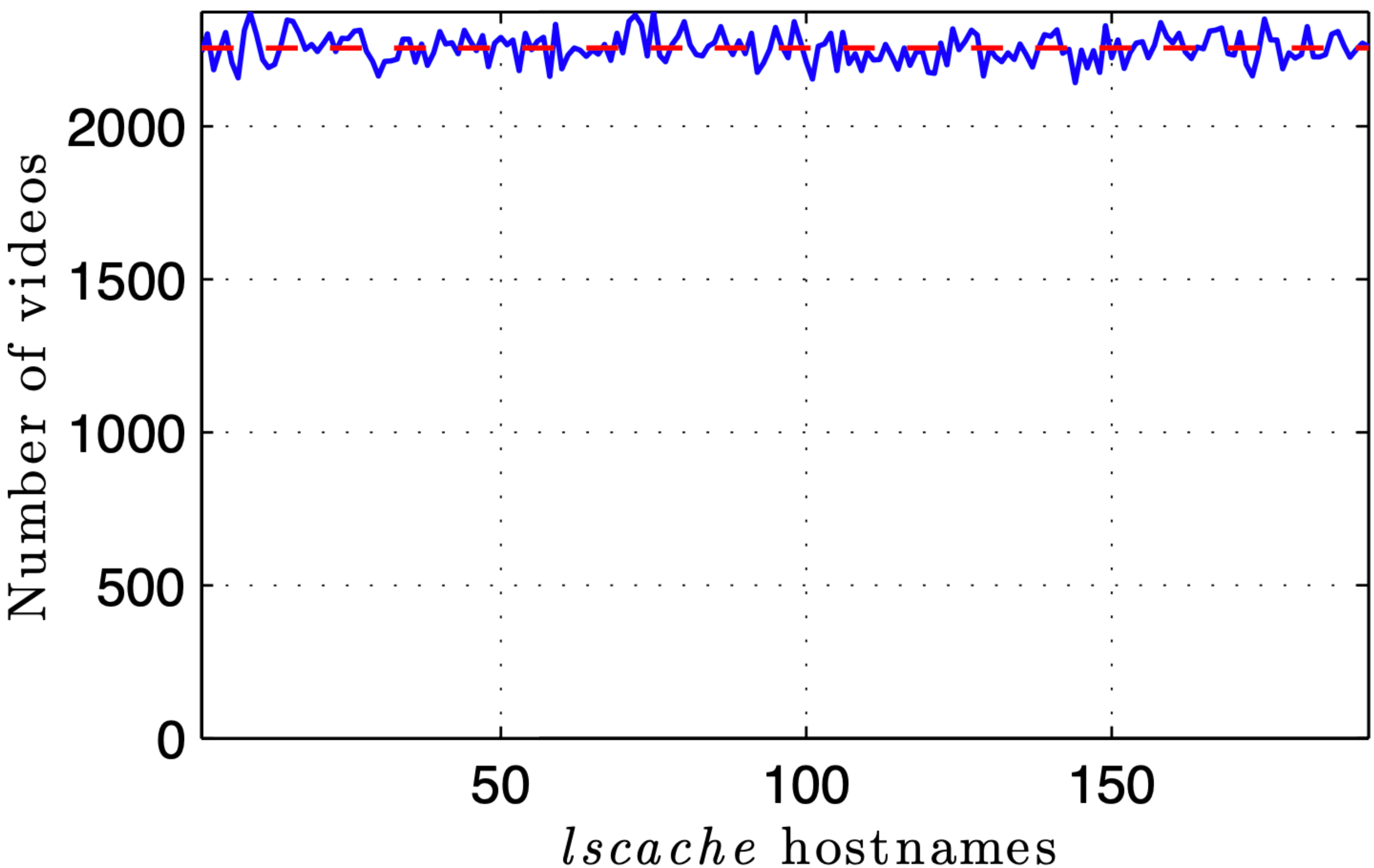

2000
1500
1000
500
0
50
100
150
Number of videos
lscache hostnames

TABLE I

YOUTUBE *Anycast* (FIRST FIVE) AND *Unicast* (LAST TWO) NAMESPACES.

DNS namespace	format	# hostnames	# IPs	# prefixes	# locations	any/uni-cast
lscache	v[1-24].lscache[1-8].c.youtube.com	192	4,999	97	38	anycast
nonxt	v[1-24].nonxt[1-8].c.youtube.com	192	4,315	68	30	anycast
tccache	tc.v[1-24].cache[1-8].c.youtube.com	192	636	15	8	anycast
cache	v[1-8].cache[1-8].c.youtube.com	64	320	5	5	anycast
altcache	alt1.v[1-24].cache[1-8].c.youtube.com	64	320	5	5	anycast
rhost	r[1-24].*city*[01-16][s,g,t][0-16].c.youtube.com	5,044	5,044	79	37	unicast
rhostisp	r[1-24].*isp*-*city*[1-3].c.youtube.com	402	402	19	13	unicast


North America
Atlantic Ocean
Europe
Asia
Africa
South America
Pacific Ocean
Indian Ocean
Australia

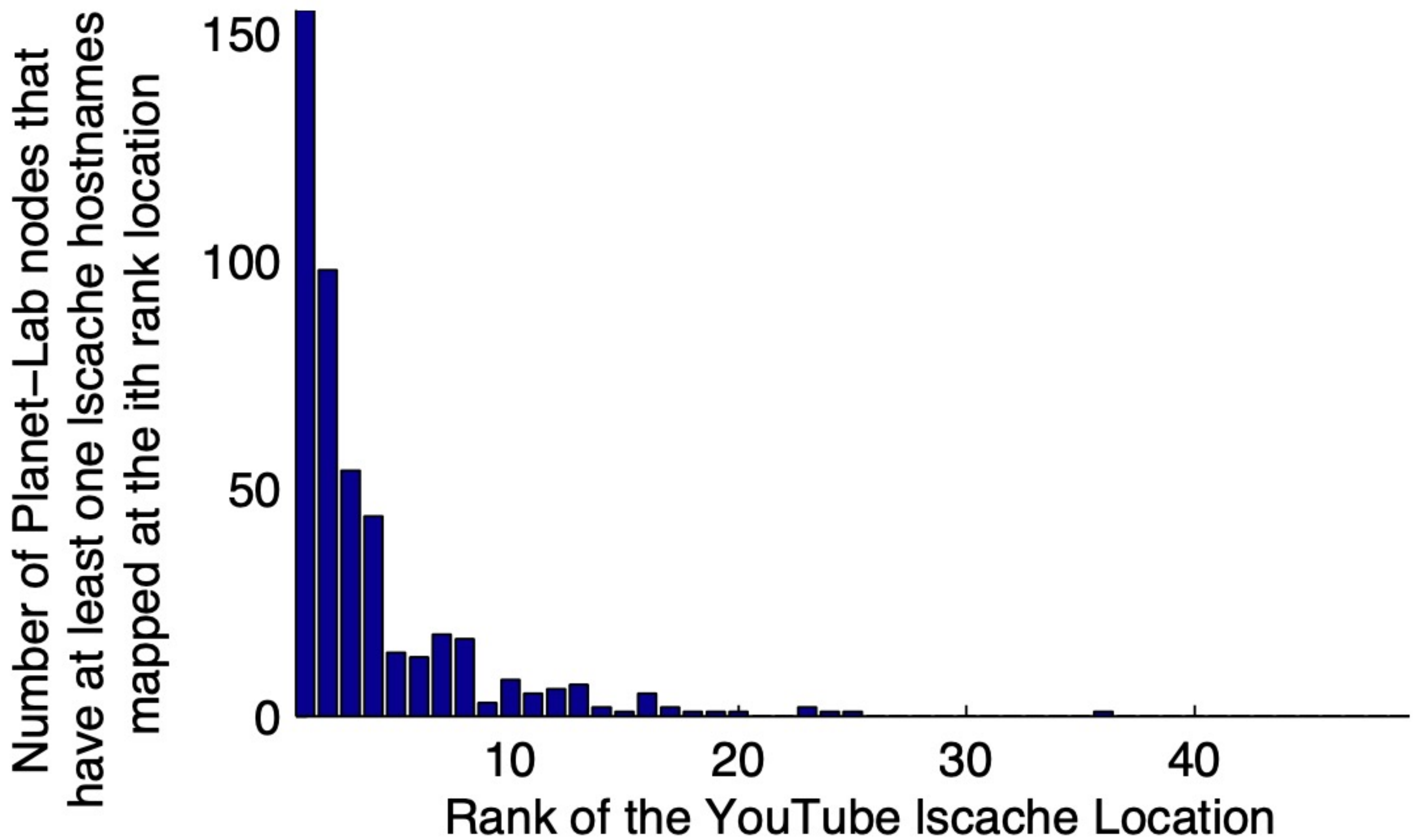


Fig. 5. Locality aware DNS mappings for *anycast* hostnames.

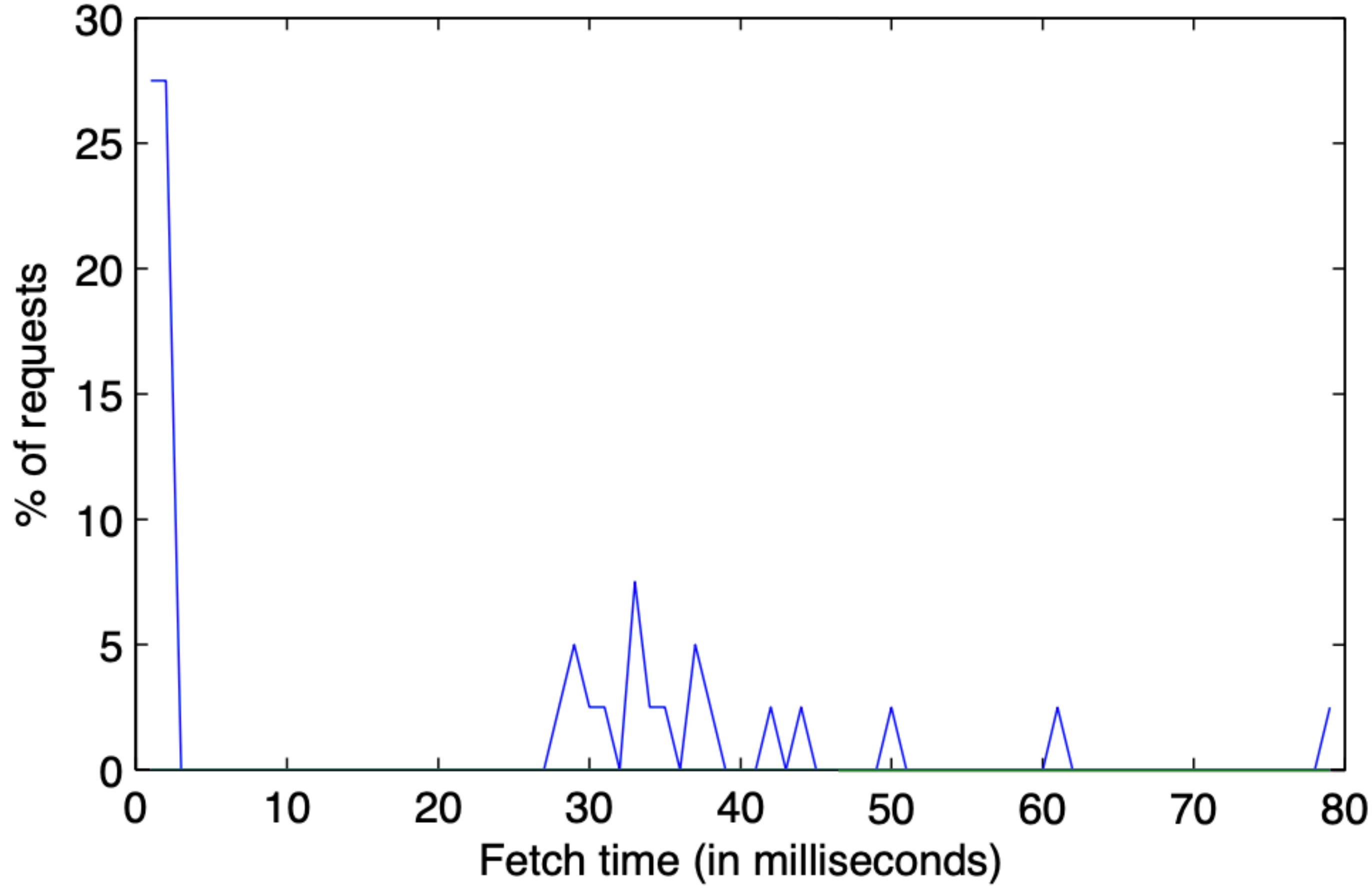


Fig. 7. Fetch time distribution at a YouTube cache server.

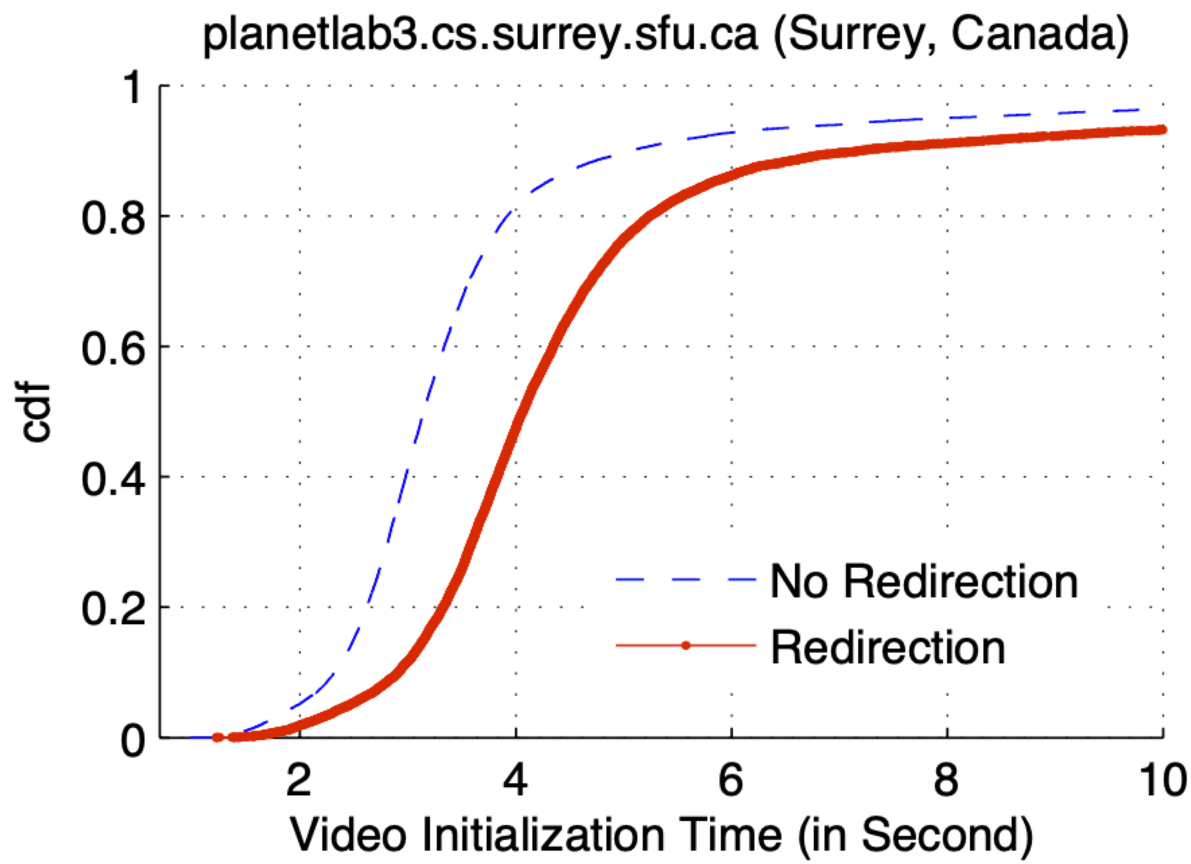


Fig. 9. An example distribution of video initialization time.