Computer Networks

Wenzhong Li

Nanjing University

Chapter 8. Internet Applications

- Internet Applications Overview
- Domain Name Service (DNS)
- Electronic Mail
- File Transfer Protocol (FTP)
- WWW and HTTP
- Content Distribution Networks (CDNs)

Client-Server and P2P

Skype

- Voice-over-IP P2P application
- Centralized server: finding address of remote party
- Direct client-client connection

Instant messaging

- Chatting between two users is P2P
- Centralized service: user presence detection/location
- User registers its IP address with central server when it comes online
- User contacts central server to find IP addresses of parties

Web and HTTP

- Web jargons
  - A Web page consists of objects
  - An Object can be HTML file, JPEG image, Java applet, audio file, …
  - Web page is composed of base HTML-file which includes several referenced objects
  - Each object is addressable by a URL
- HTTP (Hypertext Transfer Protocol)
  - Underlying protocol of the WWW (World Wide Web)
  - Transfer objects (plain text, hypertext, audio, images, and other accessible info) over Internet

URL – Uniform Resource Locator

- A unique identifier for an object on WWW
- URL format

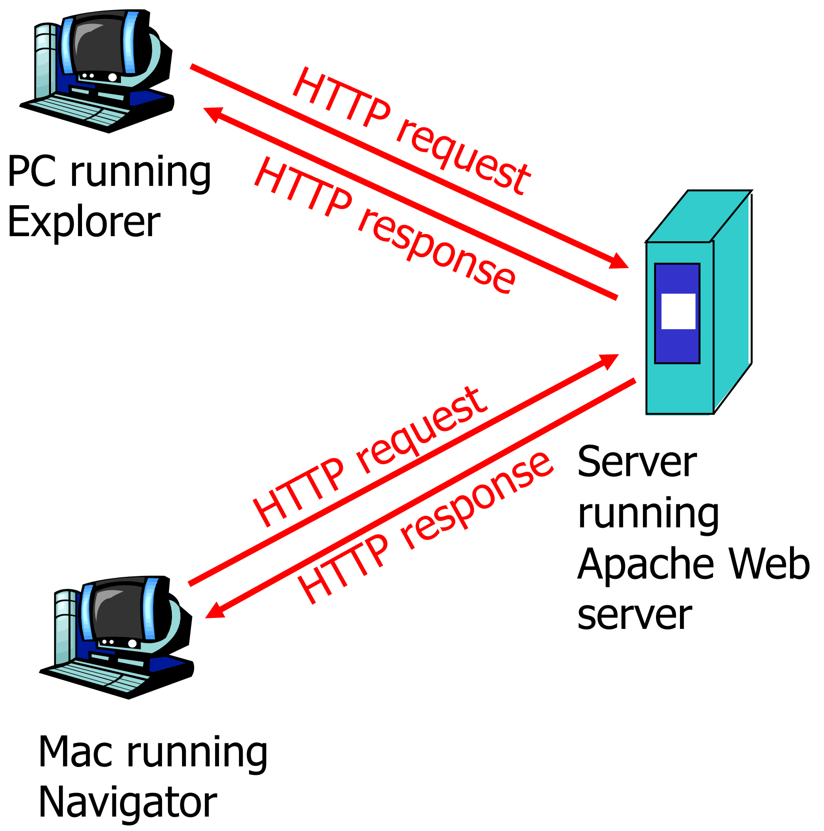
  <protocol>://<host>:<port>/<path>?query_string

  - Protocol: method for transmission or interpretation of the object, e.g. http, ftp, Gopher
  - Host: DNS name or IP address of the host where object resides
  - Path: pathname of the file that contains the object
  - Query_string: name/value pairs sent to app on the server
- An example

  http://www.nju.edu.cn:8080/somedir/page.htm

HTTP Overview

- Web's application layer protocol
- Uses TCP connections
- Client/Server model
  - Client: browser that requests, receives, "displays" Web objects
  - Server: Web server sends objects in response to requests
- HTTP versions
  - HTTP 1.0: RFC 1945
  - HTTP 1.1: RFC 2068



HTTP Procedure

- Based on TCP connection
  - Client initiates TCP connection (creates socket) to server, use port 80
  - Server accepts TCP connection from client
  - HTTP msgs exchanged between browser (HTTP client) and Web server (HTTP server)
  - TCP connection closed by server after that
- HTTP is stateless
  - Each transaction (connection) treated independently
  - Server maintains no information about past client requests

HTTP Connections

- Nonpersistent HTTP
  - At most one object is sent over a TCP connection
  - By Http 1.0
- Persistent HTTP
  - Multiple objects can be sent over single TCP connection between client and server
  - HTTP 1.1 uses persistent connections in default mode

Nonpersistent HTTP

- When user enters URL

  `http://www.someSchool.edu/someDepartment/home.index`

Http Client (Browser)

1a. C initiates TCP connection to S at www.someSchool.edu on port 80

2. C sends HTTP request msg indicating that C wants object someDepartment/home.index

5. C receives response msg, parses and displays html file, finds 10 referenced jpeg objects

6. Steps 1~5 repeated for each of 10 jpeg objects

Http Server

1b. S at www.someSchool.edu listening at port 80, accepts connection, notifying C

3. S receives request msg, forms response msg containing requested object, and sends back

4. S closes TCP connection

time ↓

Persistent HTTP (1)

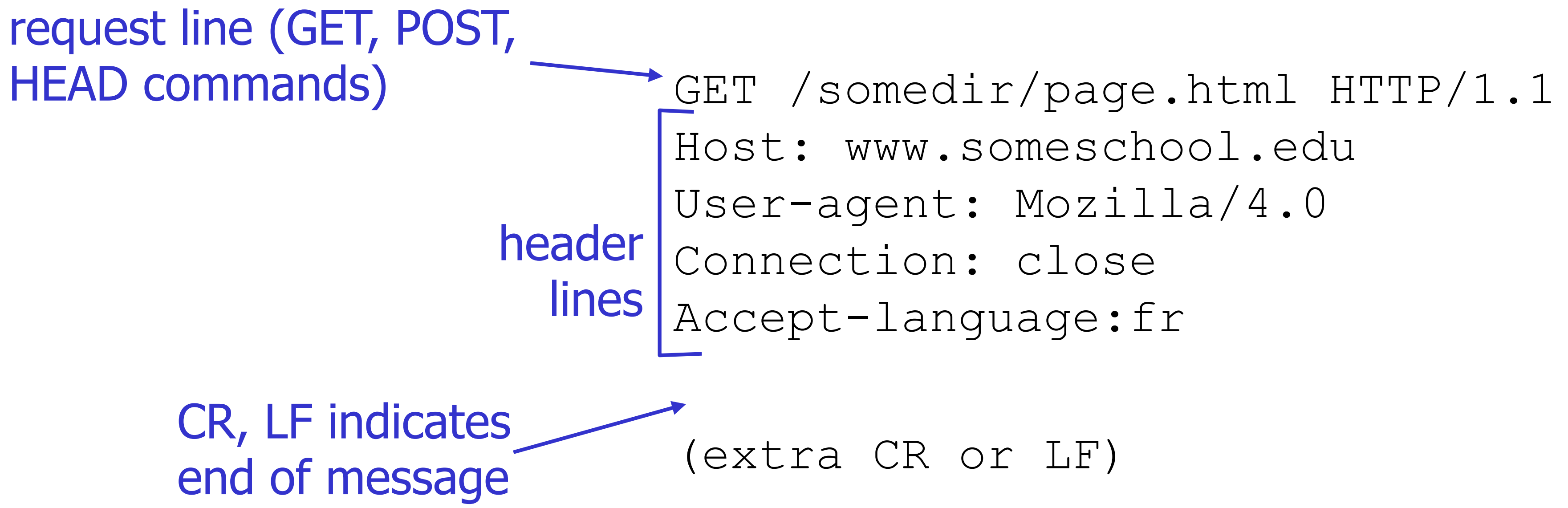
- Nonpersistent HTTP
  - Requires one transaction per object
  - Browsers often open parallel TCP connections to fetch referenced objects
  - OS must work and allocate host resources for each TCP connection
- Persistent HTTP
  - Server leaves connection open after sending response
  - Subsequent HTTP messages between same client / server are sent over connection

Persistent HTTP (2)

- Persistent without pipelining
  - Client issues new request only when previous response has been received
  - One RTT for each referenced object
- Persistent with pipelining
  - Client sends requests as soon as it encounters a referenced object
  - As little as one RTT for all the referenced objects
  - Default in HTTP 1.1

HTTP Request Message

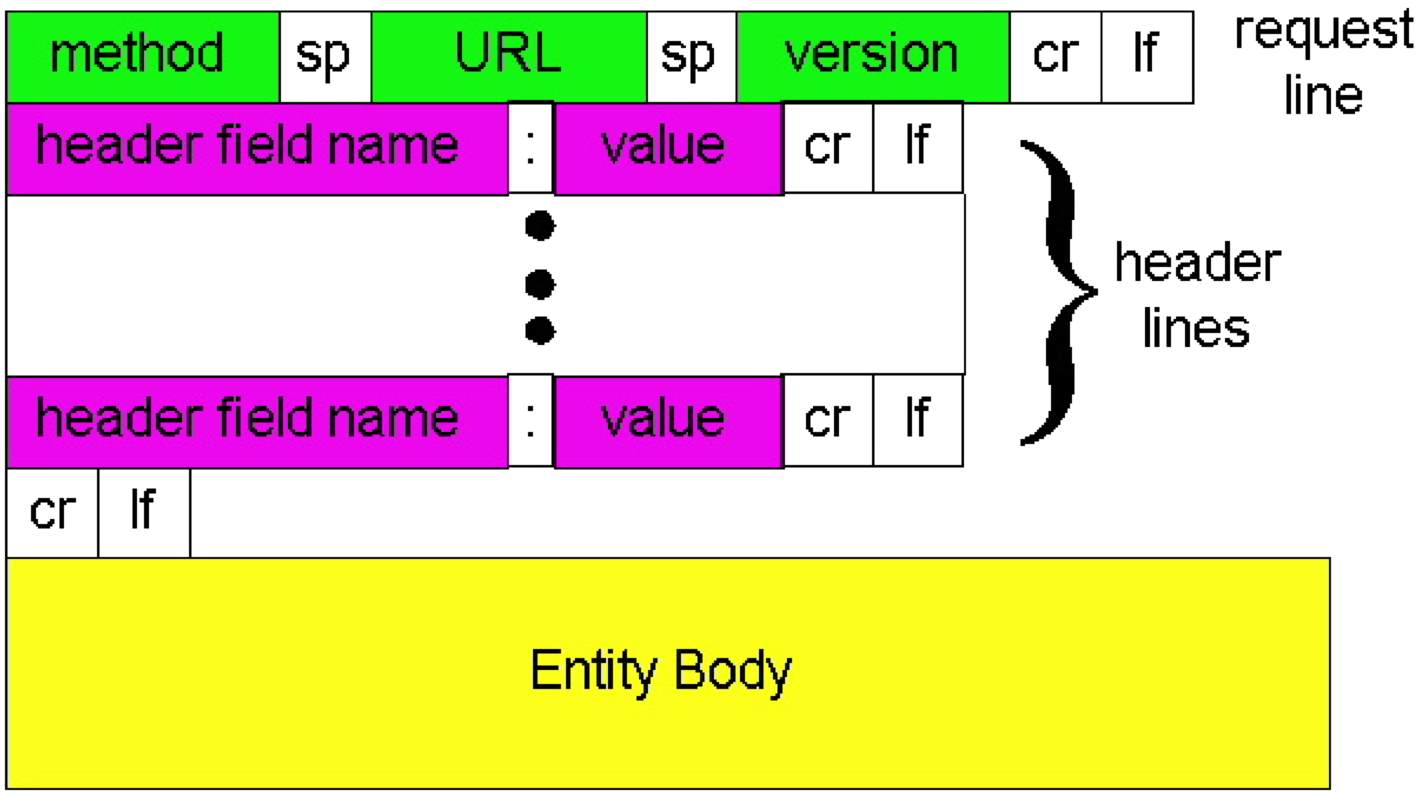
- 2 types of HTTP messages: Request, Response
- Message in 7-bit *ASCII* (human-readable format)


request line (GET, POST, HEAD commands)
header lines
CR, LF indicates end of message

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

 (extra CR or LF)
```

Request Message in Detail



2 Common Request Methods

Post method

- Web page often includes form input
- Input is uploaded to server in entity body using post

Get method

- Retrieve information on Server by URL, and display

Other common methods

- Head (retrieve only headers)
- By HTTP 1.1: Put, Delete

HTTP Response Message

- Message also in 7-bit *ASCII* (human-readable format)

status line
(code + phase)

header
lines

data, e.g. requested
HTML file

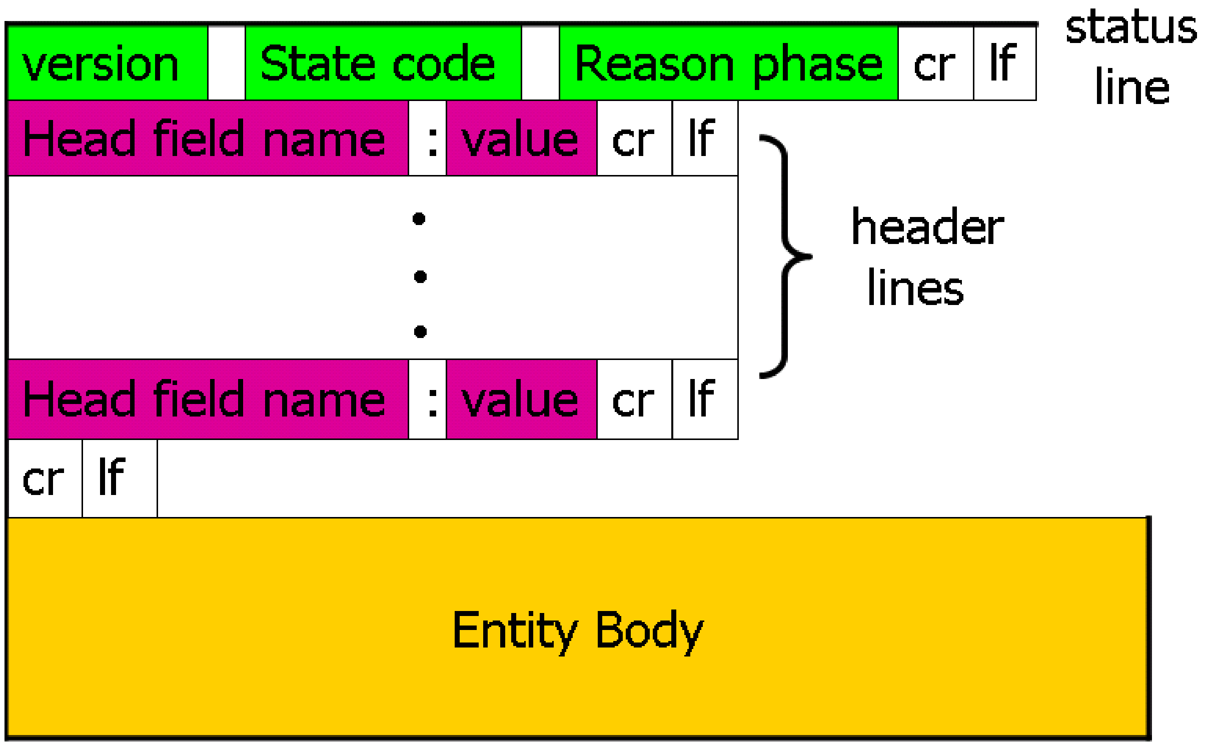
```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 2003 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …
Content-Length: 6821
Content-Type: text/html

data data data data data … …
```

Response Message in Detail

- Status-Line

  HTTP-Version <SP> Status-Code <SP> Reason-Phrase <CRLF>



Typical HTTP Status Codes

- 200 OK
  - Request succeeded, requested object later in this message
- 301 Moved Permanently
  - Requested object moved, new location specified later in this message (Location:)
- 400 Bad Request
  - Request message not understood by server
- 404 Not Found
  - Requested document not found on this server
- 505 HTTP Version Not Supported

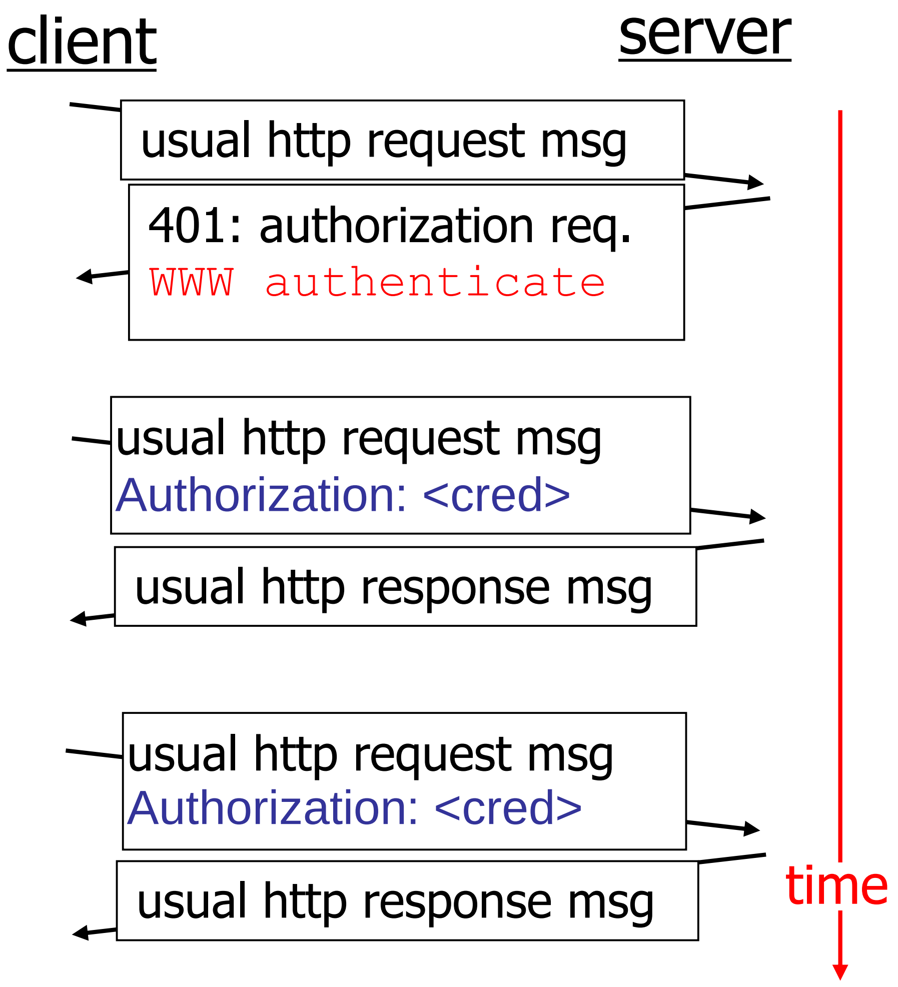
Entity Body

- Arbitrary sequence of octets specifying the resource
- HTTP transfers any type of data
  - Text, Binary data
  - Audio, Images, Video
- Interpretation of data determined by header fields
  - Content-Type: text/html; charset = ISO-8859-4
  - Content-Encoding: gzip
  - Transfer-Encoding: chunked

User-Server Interaction: Authorization

Authorization: control access to server content

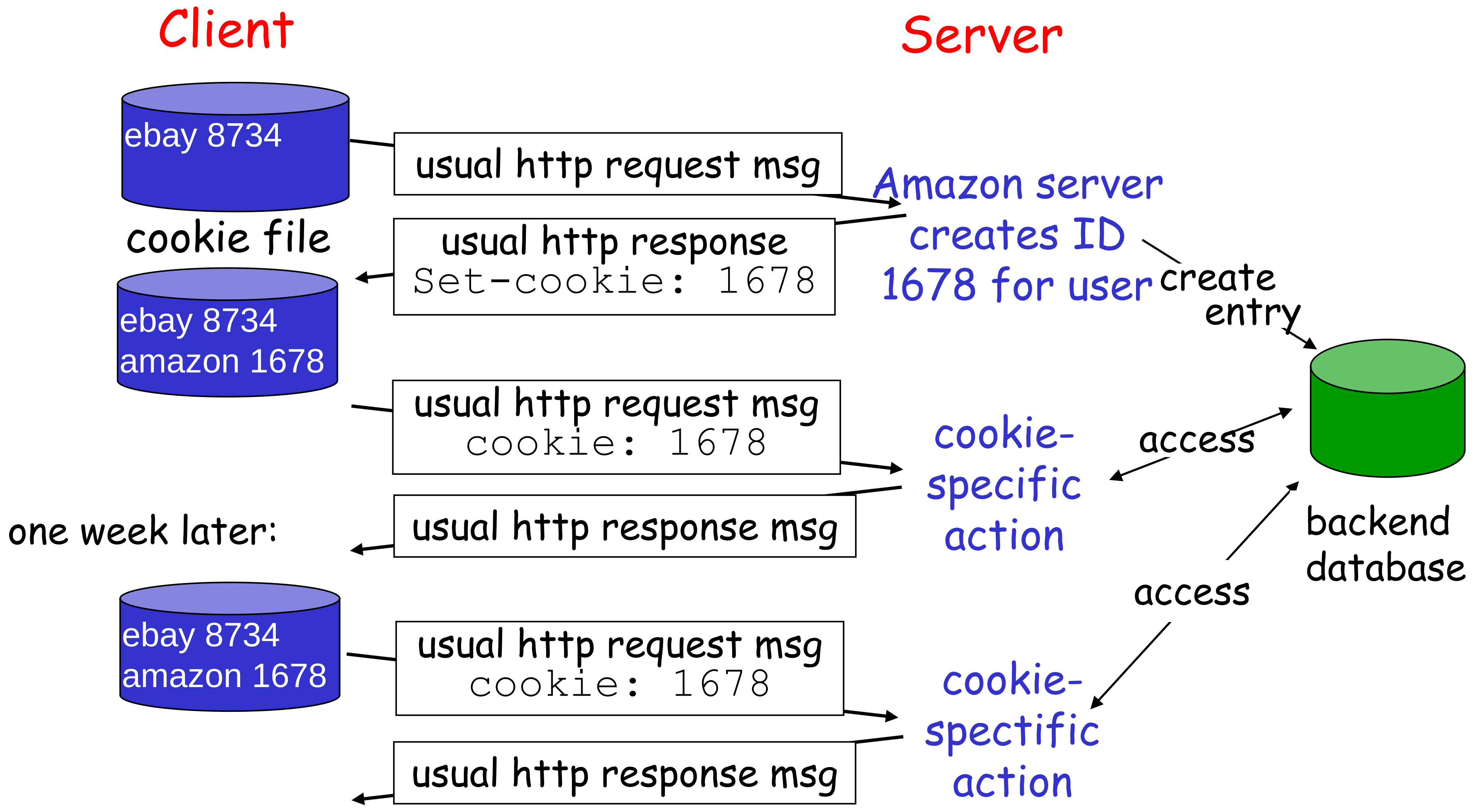
- Authorization header line in each request
- Authorization credentials: typically name, password
- Stateless: client must present authorization in each request



Cookies: Keeping State

- Many major Web sites use *cookies*
  - Keep track of client's status on server
- Major components
  - Cookie header line in the HTTP *request* / *response* message
  - Cookie file kept on client's host and managed by client's browser
  - Back-end database at Web server site

A Cookies Example



Application of Cookies

What cookies can bring

- Authorization
- Shopping carts
- Recommendations
- User session state (Web Email)

aside

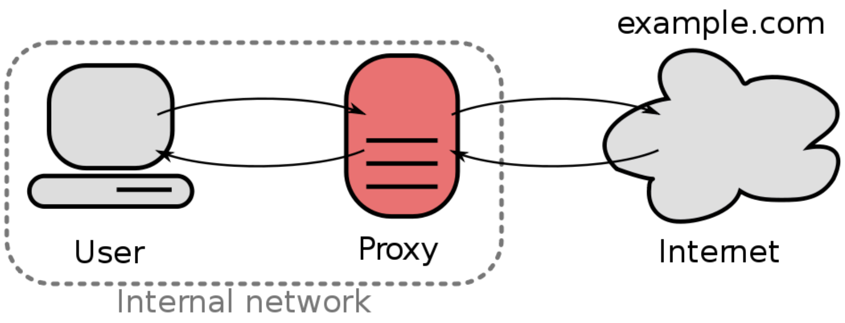
Cookies and privacy

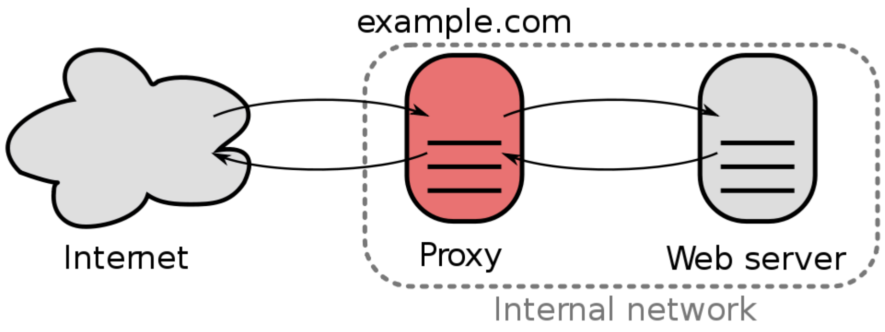
- Cookies permit servers to learn a lot about user
- User may supply name and Email to servers
- Search engines may use cookies to obtain info across sites
- Hacked browser may do bad things with cookies

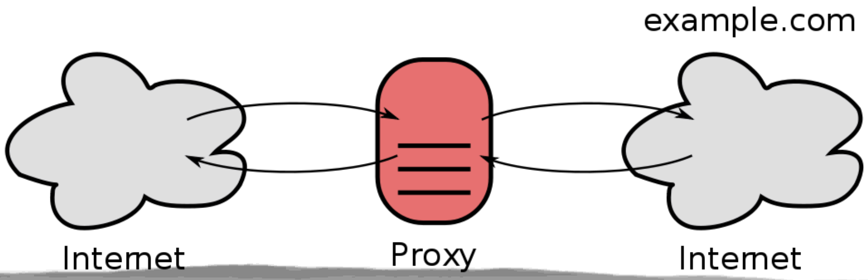
Proxy Server

- An intermediary app that
  - Represents the *client* to issue *request*, and
  - Represents the *server* to give *response*

- Different types

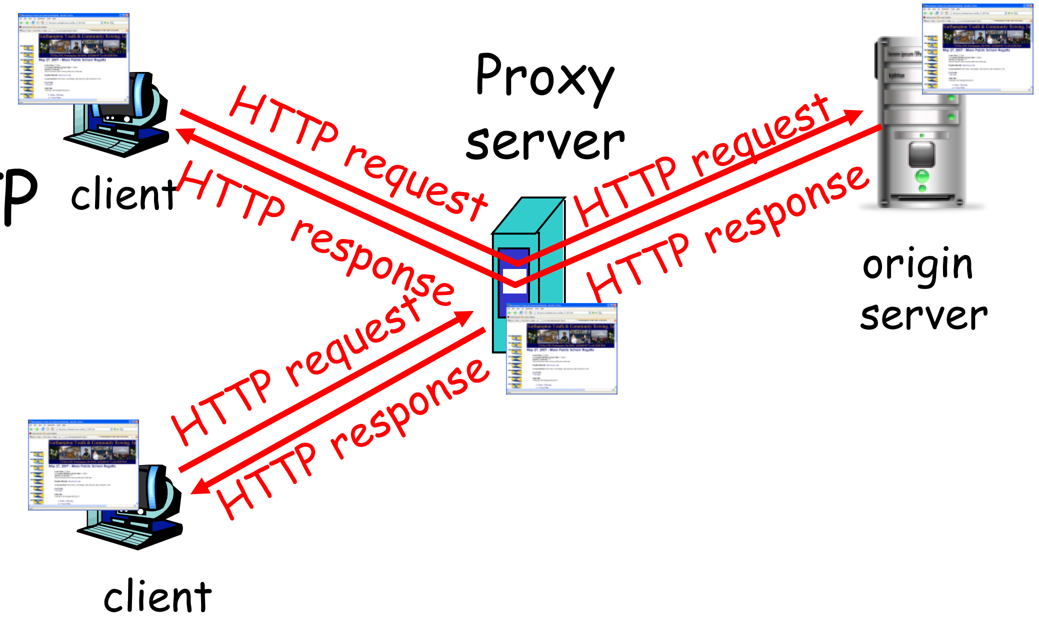






Web Caches

- User sets browser: Web accesses via cache
- Browser sends all HTTP requests to cache on proxy server
  - Object in cache: cache returns object
  - Or cache requests object from origin server, then returns object to client



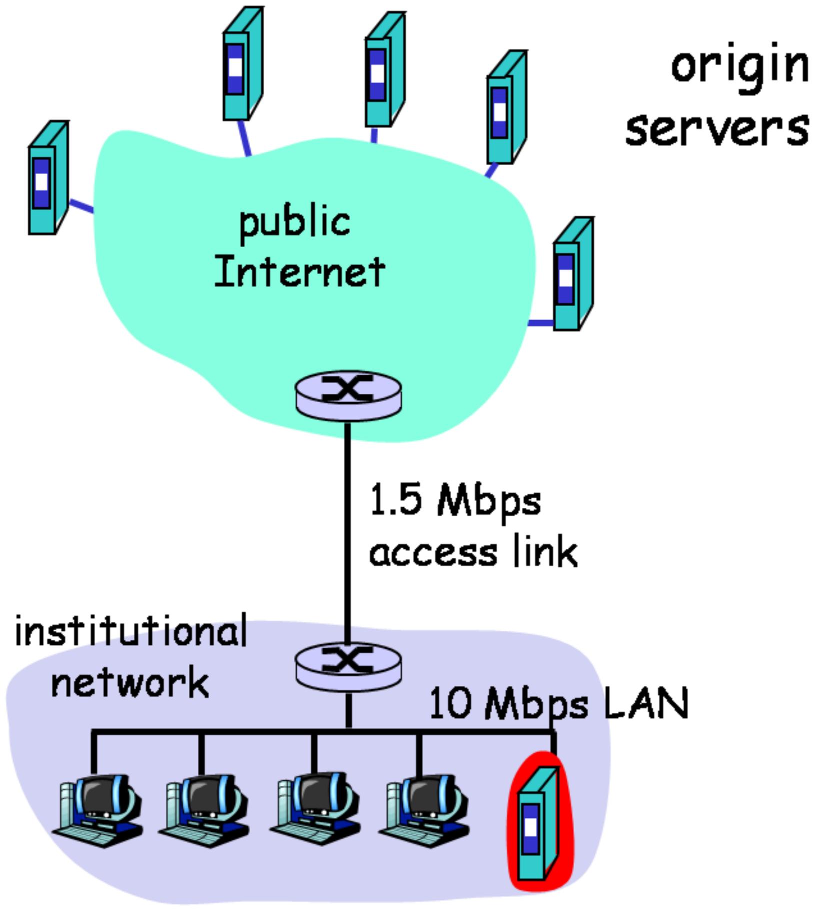
Caching Example

Institutional cache

- Satisfy internal client *request* without involving origin server

Considerations

- Smaller response time
- Decrease traffic to distant servers
- Load balancing



Caching Example

- One trip delay = Internet delay + access delay + LAN delay

Suppose

- Internet delay = 2 sec
- LAN delay = 2 msec
- Access delay = 10 msec

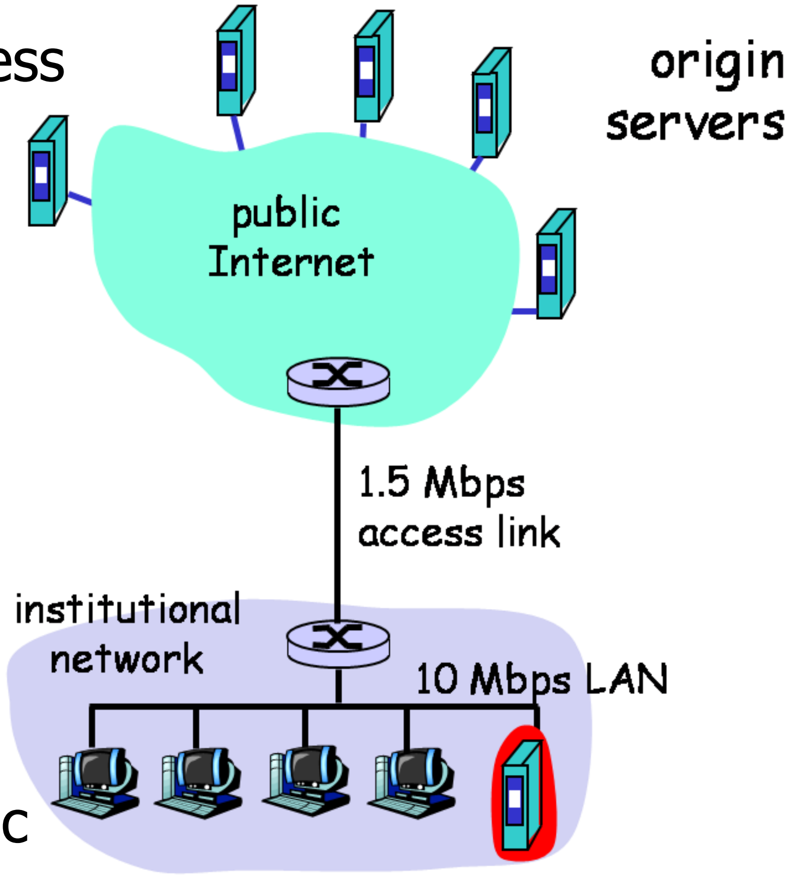
- Suppose hit rate is 0.4 (40%)

- Access without cache:

$(2000+2+10)\times 2$ = 4024 msec = 4.02 sec

- Access with cache:

$(2+10 + 0.6\times 2000)\times 2$ = 2424 msec = 2.4 sec



Conditional GET

Goal

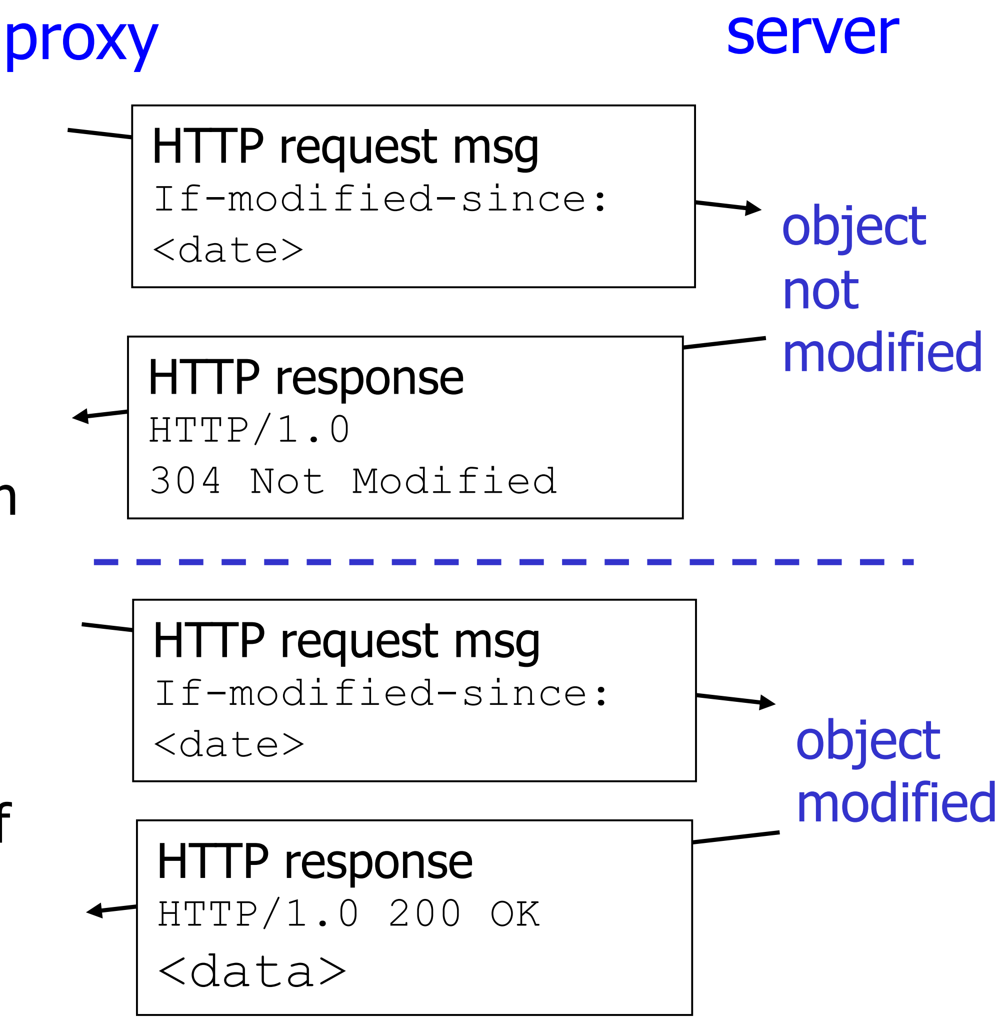
- Don't send object if proxy has up-to-date cached version

Client (Proxy)

- Specify date of cached copy in HTTP request

Server

- Response contains no object if cached copy is up-to-date



HTTP request msg
```
If-modified-since:
<date>
```

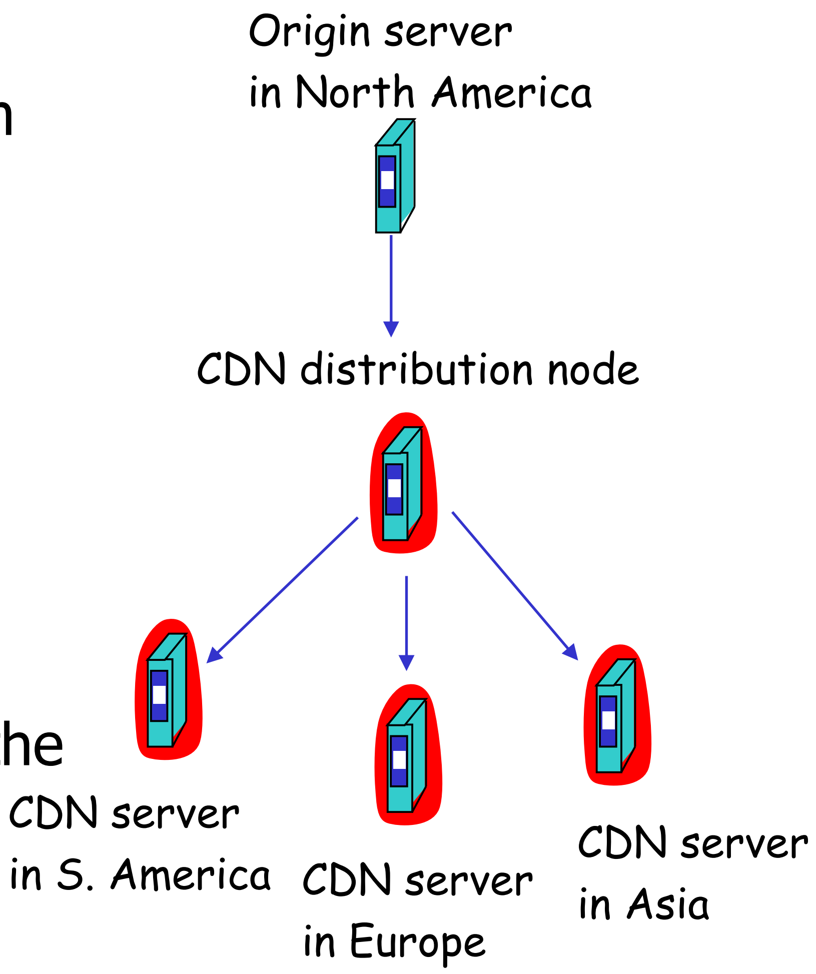
HTTP response
```
HTTP/1.0
304 Not Modified
```

HTTP request msg
```
If-modified-since:
<date>
```

HTTP response
```
HTTP/1.0 200 OK
<data>
```

Content Distribution Networks (CDNs)

- Challenge
  - Stream large files (e.g. video) from single origin server in real time
  - Protect origin server from DDOS attacks
- Solution
  - Replicate content at hundreds of servers throughout Internet
  - CDN distribution node coordinate the content distribution
  - Placing content close to user



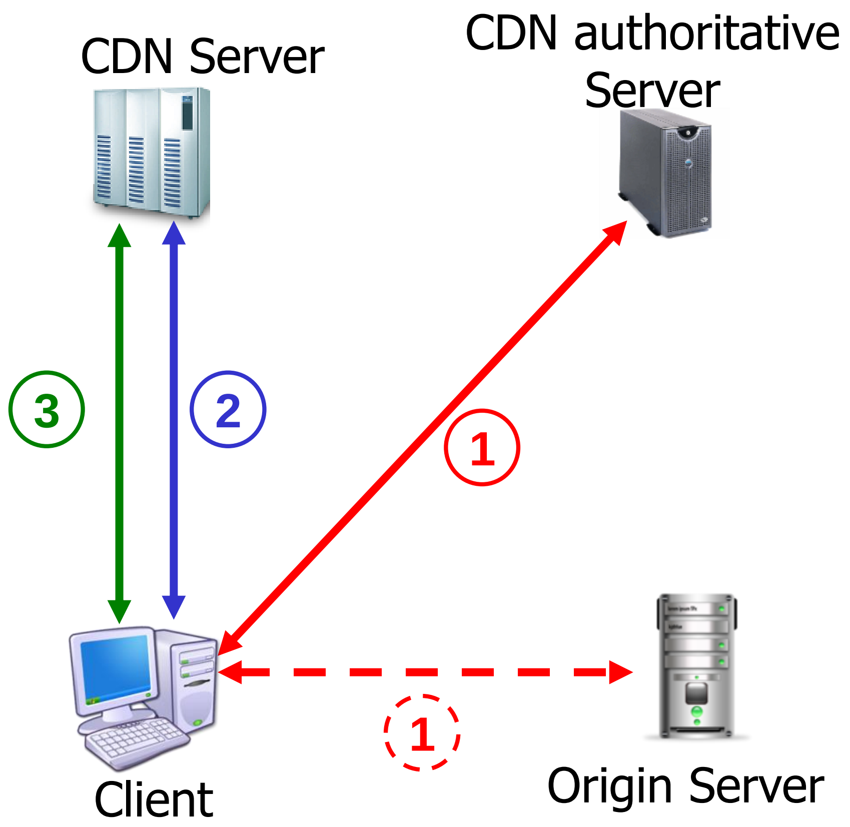
Content Replication

- Content provider (origin server) is CDN customer
- CDN replicates customers' content in CDN servers
- When provider updates content, CDN updates its servers
- Use authoritative DNS server to redirect requests

Supporting Techniques

- DNS
  - One name maps onto many addresses
- Routing
  - Content-based routing (to nearest CDN server)
- URL Rewriting
  - Replaces “http://www.sina.com/sports/tennis.mov” with “http://www.cdn.com/www.sina.com/sports/tennis.mov”
- Redirection strategy
  - Load balancing, network delay, cache/content locality

CDN Operation



1′ URL rewriting – get authoritative server

1. Get near CDN server IP address
2. Warm up CDN cache
3. Retrieve pages/media from CDN Server

Redirection

- CDN creates a "map", indicating distances from leaf ISPs and CDN servers
- When query arrives at authoritative DNS server
  - Server determines ISP from which query originates
  - Uses "map" to determine best CDN server
- CDN servers create an application-layer overlay network

Summary

- Conceptual, implementation aspects of network application protocols
  - Client-Server vs. Peer-to-Peer
  - Data presentation formatting
- Examining popular application-level protocols
  - DNS, SNMP / MIB
  - HTTP, FTP, SMTP / POP3 / MIME
  - Content distribution networks (CDNs)