Information Technology Fundamentals

Mohammad Hossein Manshaei

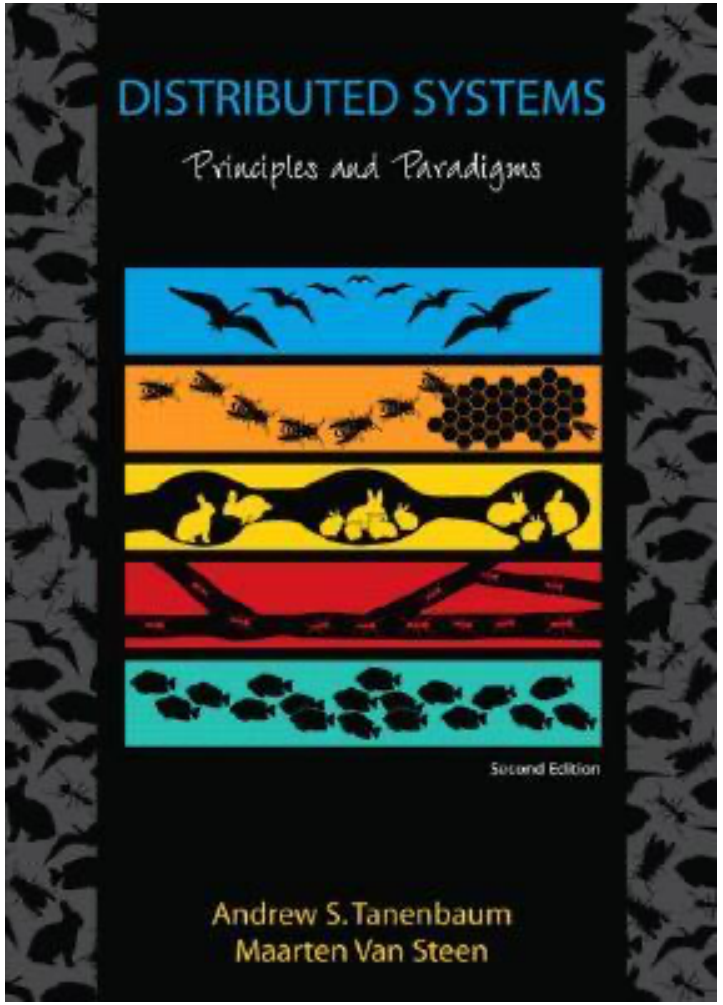
manshaei@gmail.com

Web Systems: Communication, Consistency, and Replication
Module 5: Part 2

Main Reference

Tanenbaum, Andrew S., and Maarten Van Steen. ***Distributed systems: principles and paradigms.*** Prentice-Hall, 2007

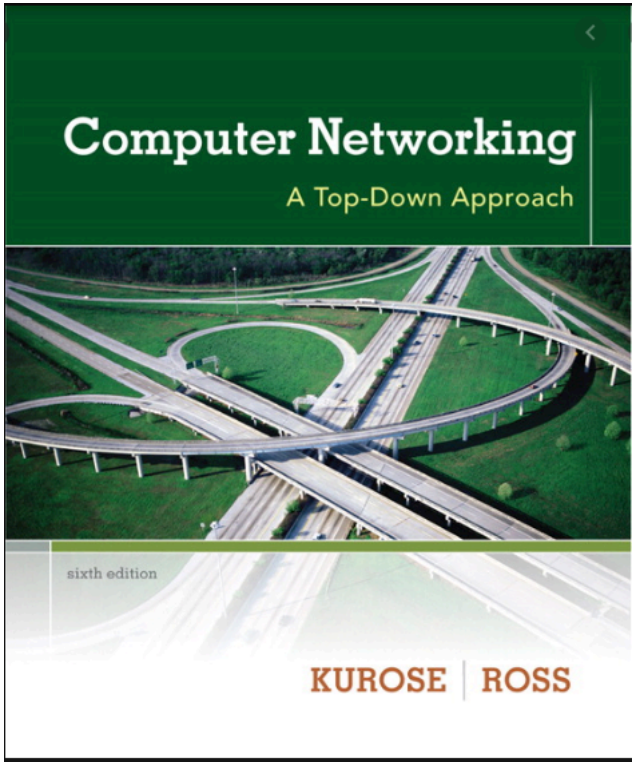
Chapter 12: Distributed Web-Based Systems



Main Reference

Kurose and Ross.
Computer Networking: A Top-Down Approach.
Sixth Edition.

Chapter 7 and 8



Contents

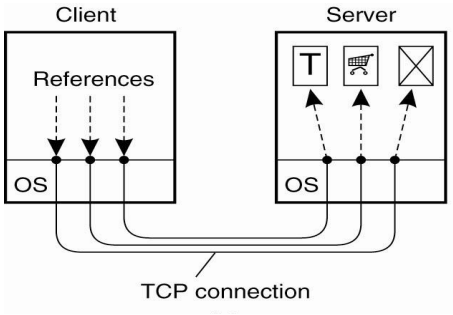
- Web System Communication
  - HTTP Review
  - SOAP
  - Naming
- Consistency and Replication
  - Web Caching
  - Content Distribution Networks

Web-Based Communication

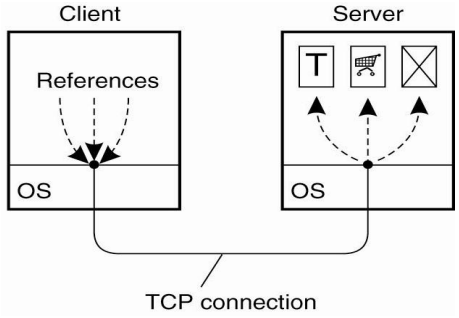
1. Hypertext Transfer Model for Traditional Web
2. SOAP for Web Service

HTTP/1 Review

- **Non-Persistent HTTP:** Each request to a server required setting up a separate connection. When the server had responded, the connection was broken down again.



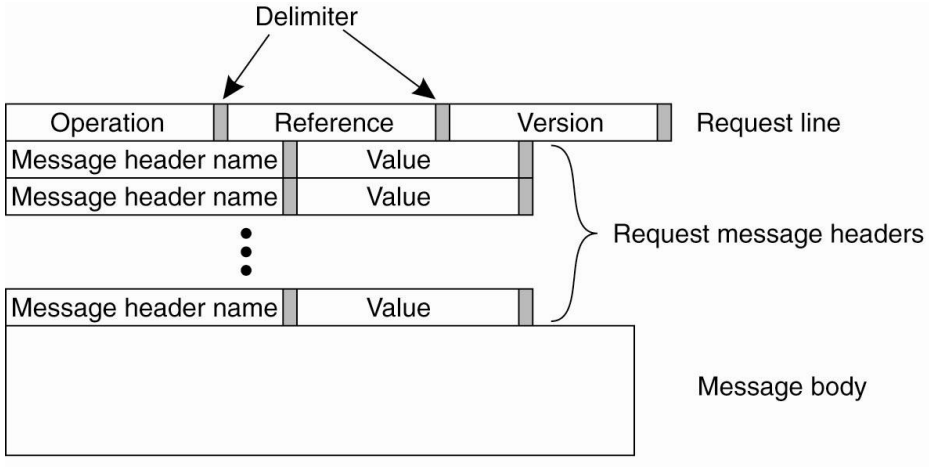
- **Persistent HTTP 1.1:** Make use of a persistent connection, which can be used to issue several requests (and their respective responses), without the need for a separate connection for each (request/response)-pair

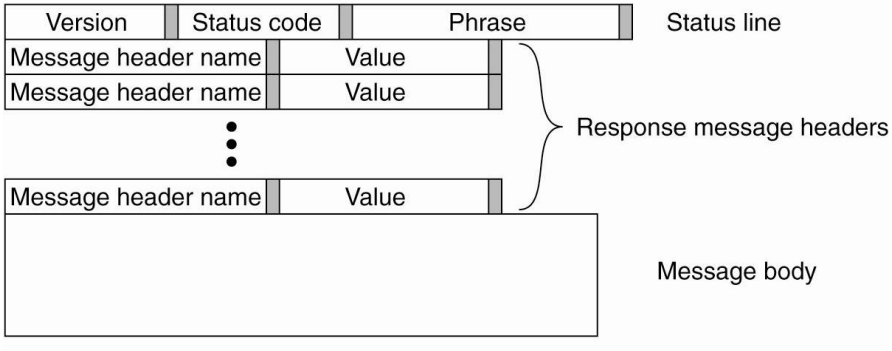


HTTP Methods

Operation	Description
Head	Request to return the header of a document
Get	Request to return a document to the client
Put	Request to store a document
Post	Provide data that are to be added to a document (collection)
Delete	Request to delete a document

HTTP Messages: Request and Response Messages





HTTP Headers

Header	Source	Contents
Accept	Client	The type of documents the client can handle
Accept-Charset	Client	The character sets are acceptable for the client
Accept-Encoding	Client	The document encodings the client can handle
Accept-Language	Client	The natural language the client can handle
Authorization	Client	A list of the client's credentials
WWW-Authenticate	Server	Security challenge the client should respond to
Date	Both	Date and time the message was sent
ETag	Server	The tags associated with the returned document
Expires	Server	The time for how long the response remains valid
From	Client	The client's e-mail address
Host	Client	The DNS name of the document's server

Header	Source	Contents
If-Match	Client	The tags the document should have
If-None-Match	Client	The tags the document should not have
If-Modified-Since	Client	Tells the server to return a document only if it has been modified since the specified time
If-Unmodified-Since	Client	Tells the server to return a document only if it has not been modified since the specified time
Last-Modified	Server	The time the returned document was last modified
Location	Server	A document reference to which the client should redirect its request
Referer	Client	Refers to client's most recently requested document
Upgrade	Both	The application protocol the sender wants to switch to
Warning	Both	Information about the status of the data in the message

HTTP/2

Key goal: decreased delay in multi-object HTTP requests

HTTP1.1: introduced multiple, pipelined GETs over single TCP connection

- server responds *in-order* (FCFS: first-come-first-served scheduling) to GET requests
- with FCFS, small object may have to wait for transmission (head-of-line (HOL) blocking) behind large object(s)
- loss recovery (retransmitting lost TCP segments) stalls object transmission

HTTP/2

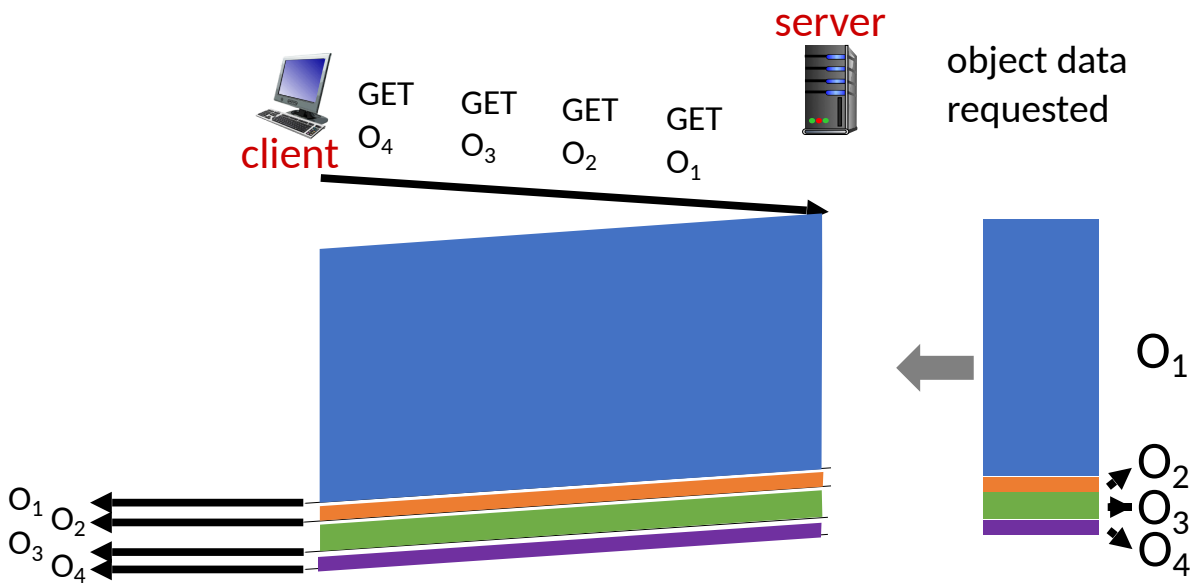
Key goal: decreased delay in multi-object HTTP requests

HTTP/2: [RFC 7540, 2015] increased flexibility at *server* in sending objects to client:

- methods, status codes, most header fields unchanged from HTTP 1.1
- transmission order of requested objects based on client-specified object priority (not necessarily FCFS)
- *push* unrequested objects to client
- divide objects into frames, schedule frames to mitigate HOL blocking

HTTP/2: Mitigating HOL Blocking

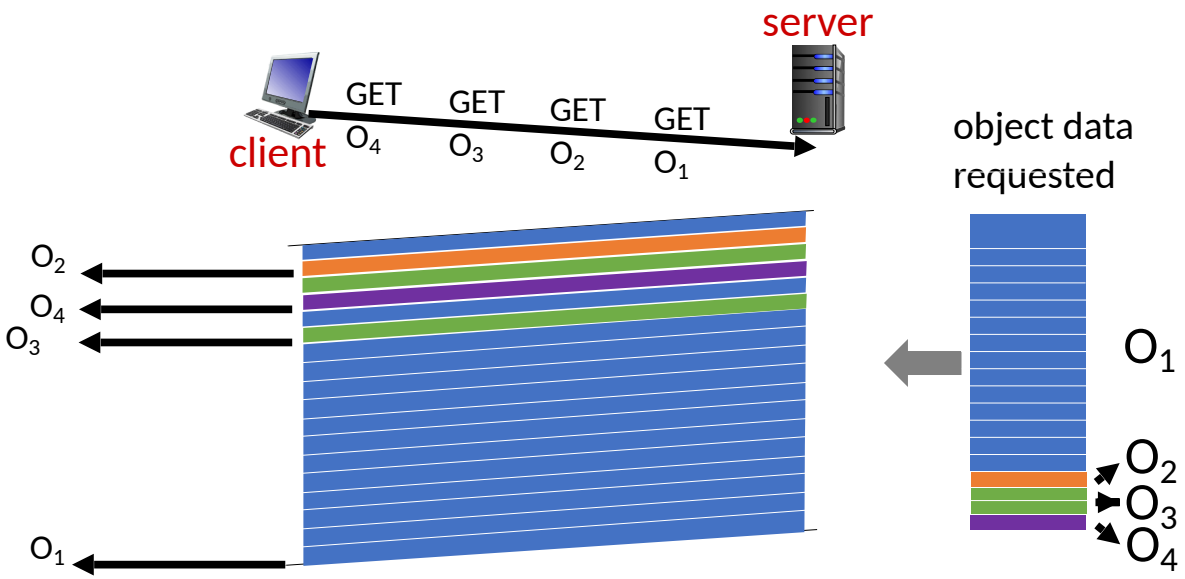
HTTP 1.1: client requests 1 large object (e.g., video file, and 3 smaller objects)



objects delivered in order requested: O_2, O_3, O_4 wait behind O_1

HTTP/2: Mitigating HOL Blocking

HTTP/2: objects divided into frames, frame transmission interleaved



O_2, O_3, O_4 delivered quickly, O_1 slightly delayed

HTTP/2 to HTTP/3 (Aug 2021)

Key goal: decreased delay in multi-object HTTP requests

HTTP/2 over single TCP connection means:

- recovery from packet loss still stalls all object transmissions
  - as in HTTP 1.1, browsers have incentive to open multiple parallel TCP connections to reduce stalling, increase overall throughput
- no security over vanilla TCP connection
- HTTP/3: adds security , per object error- and congestion-control (more pipelining) over UDP

HTTP/2 to HTTP/3: RFC 9000 and 9001

1. Transport Protocol:

HTTP/3 uses the QUIC (Quick UDP Internet Connections) protocol, which runs on top of UDP. QUIC is designed to provide a more efficient and secure transport compared to TCP. It incorporates key features from TCP and TLS at the transport layer and is specifically designed for HTTP/3.

2. Multiplexing and Reduce Head-of-Line Blocking

HTTP/3: It is able to handle head-of-line blocking better, which is a limitation in HTTP/2 over TCP. Lead to even better performance, especially in scenarios with high packet loss.

3. Connection Establishment:

HTTP/3: Because HTTP/3 is built on top of QUIC, it benefits from the reduced connection establishment latency. No TCP and TLS handshake

4. Improved Congestion Control

6. Improved Security

Since QUIC integrates both transport and security (encryption), there's less latency during connection establishment, improving overall performance.

7. Better NAT Traversal

Contents

- Web System Communication
  - HTTP Review
  - SOAP
  - Naming
- Consistency and Replication
  - Web Caching
  - Content Distribution Networks

Simple Object Access Protocol: SOAP

1. Simple Object Access Protocol (SOAP) forms the standard for communication with Web services
2. Designed based on this assumption: "Two Communication Parties have a very little common knowledge" ==> Based on XML

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
        <n:priority>1</n:priority>
        <n:expires>2001-06-22T14:00:00-05:00</n:expires>
      </n:alertcontrol>
    </env:Header>
    <env:Body>
      <m:alert xmlns:m="http://example.org/alert">
        <m:msg>Pick up Mary at school at 2pm</m:msg>
      </m:alert>
    </env:Body>
</env:Envelope>
```

Simple Object Access Protocol

- A SOAP message generally consists of two parts, which are jointly put inside what is called a SOAP envelope:

1. The **body** contains the actual message
2. The **header** (is optional) containing information relevant for nodes along the path from sender to receiver

- Nodes consist of the various processes in a multi-tiered implementation of a Web service
- Everything in the envelope is expressed in XML

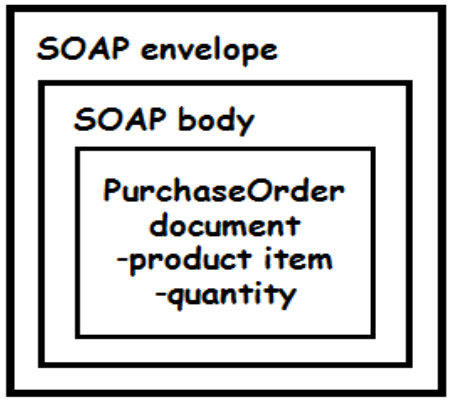
Simple Object Access Protocol

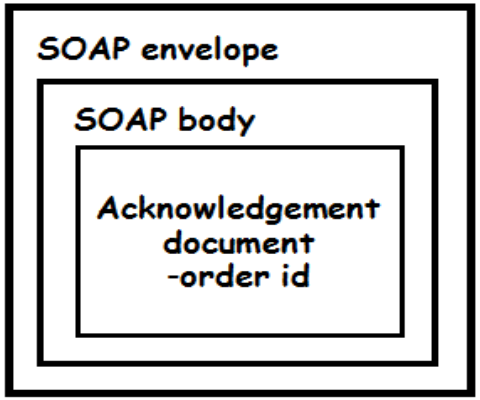
- A SOAP envelope does not contain the address of the recipient. SOAP specifies bindings to underlying transfer protocols.
  1. **HTTP** (The recipient will be specified in the form of a URL)
  2. **SMTP** (The recipient in the form of an email address)
- Two different styles of interactions:
  1. The **Conversational Exchange Style** (will be bound to either SMTP or HTTP)
  2. **RPC-Style** (Remote Procedure Call, binding to HTTP)

Conversational Exchange Style

Two parties essentially exchange structured documents

For example, such a document may contain a complete purchase order as one would fill in when electronically booking a flight. The response to such an order could be a confirmation document, now containing an order number, flight information, a seat reservation, and perhaps also a bar code that needs to be scanned when boarding

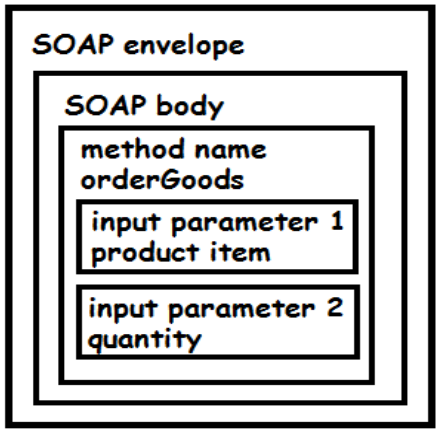


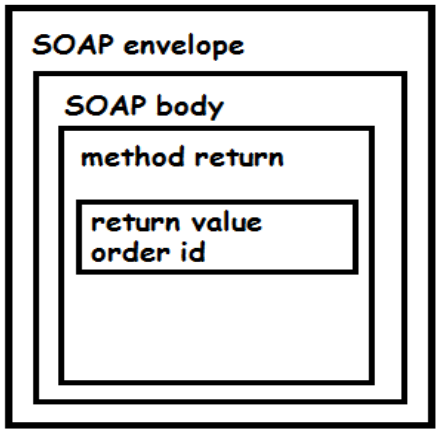


RPC-Style Exchange

The SOAP message will identify explicitly the procedure to be called

- Provide a list of parameter values as input to that call
- The response will be a formal message containing the response to the call.





Contents

- Web System Communication
  - HTTP Review
  - SOAP
  - Naming
- Consistency and Replication
  - Web Caching
  - Content Distribution Networks

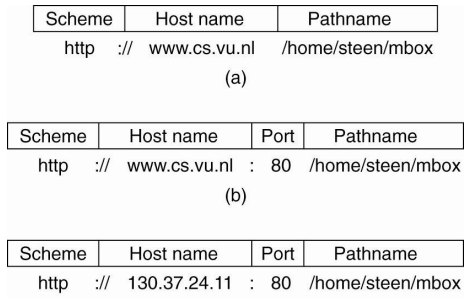
Naming

The Web uses a single naming system to refer to documents. The names used are called Uniform Resource Identifiers (URIs):

1. **A Uniform Resource Locator (URL)** is a URI that identifies a document by including information on how and where to access the document. In other words, a URL is a location-dependent reference to a document.
2. **A Uniform Resource Name (URN)** acts as true identifier. A URN is used as a globally unique, location-independent, and persistent reference to a document.

The http scheme is the best known, but it is not the only one.

Name	Used for	Example
http	HTTP	http://www.cs.vu.nl:80/globe
mailto	E-mail	mailto:steen@cs.vu.nl
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:/edu/book/work/chp/11/11
data	Inline data	data:text/plain;charset=iso-8859-7,%e1%e2%e3
telnet	Remote login	telnet://flits.cs.vu.nl
tel	Telephone	tel:+31201234567
modem	Modem	modem:+31201234567;type=v32



Contents

- Web System Communication
  - HTTP Review
  - SOAP
  - Naming
- Consistency and Replication
  - Web Caching
  - Content Distribution Networks

Consistency and Replication

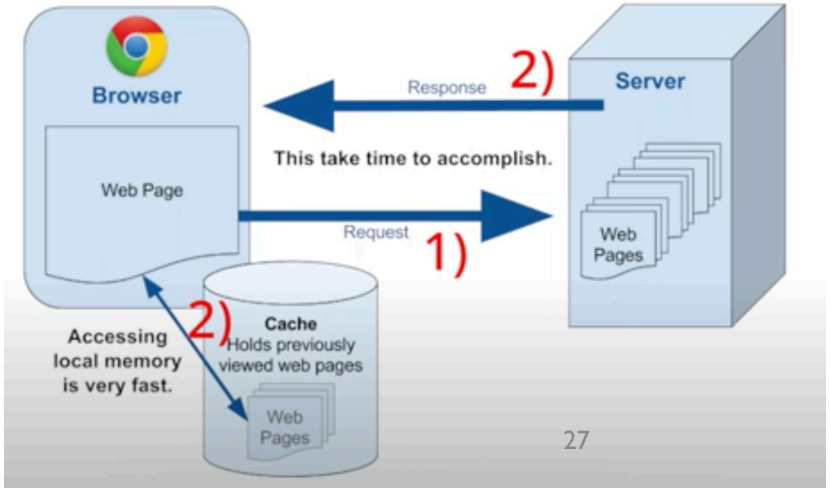
The most important systems-oriented developments in Web-based distributed systems is ensuring that access to Web documents meets stringent performance and availability requirements

These requirements have led to numerous proposals for caching and replicating Web content

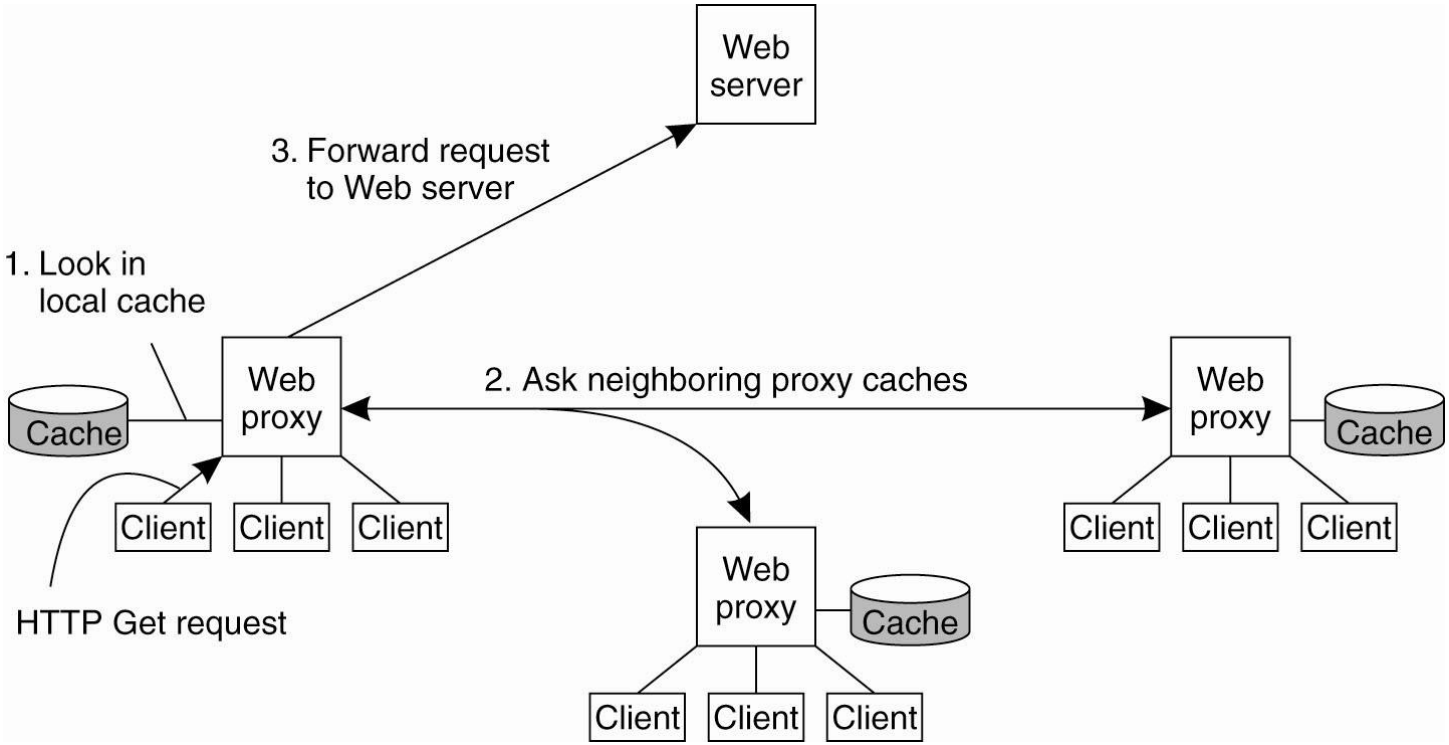
Both for **Static** and **Dynamic** Documents

Web Proxy Caching

- **Client-Side Caching**
  - Simple Caching Facility (e.g., Browser)
  - Web Proxy (e.g., accept requests from local clients)
- We can also deploy caches to cover region or a country, i.e., hierarchical caches (higher latency is caused by the need for the client to check multiple caches)



Cooperative Caching



Comparison between Hierarchical and Cooperative Caching

1. Cooperative caches are generally connected through high-speed links, the transmission time needed to fetch a document is much lower than for a hierarchical cache.
2. Storage requirements are less strict for cooperative caches than hierarchical ones.
3. Expected latencies for hierarchical caches are lower than for distributed caches (Only for the cases that cached are on the web)

Expiration Time

To improve performance at the cost of weaker consistency, the widely- used Squid Web proxy assigns an expiration time that depends on how long ago the document was last modified when it is cached:

$$T_{expire} = \alpha(T_{cached} - T_{last\_modified}) + T_{cached}$$

Replication for Web Hosting Systems

Maintaining the content of a Web site and making sure that the site is easily and continuously accessible

Content Delivery/Distribution Networks (CDN): The main idea underlying these CDNs is that they act as a Web hosting service, providing an infrastructure for distributing and replicating the Web documents of multiple sites across the Internet.

General Organization of a CDN as a Feedback-Control System

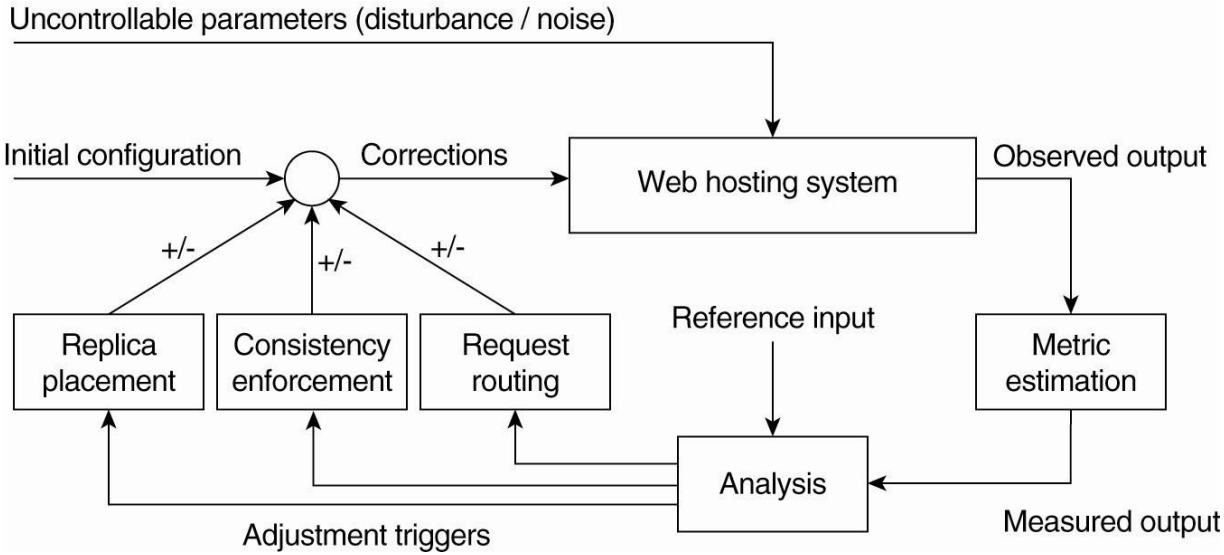
Three different kinds of aspects related to replication:

1) Metric estimation

2) Adaptation triggering

3) Taking appropriate measures:

A. Replica placement decisions **B.** Consistency enforcement **C.** Client-request routing

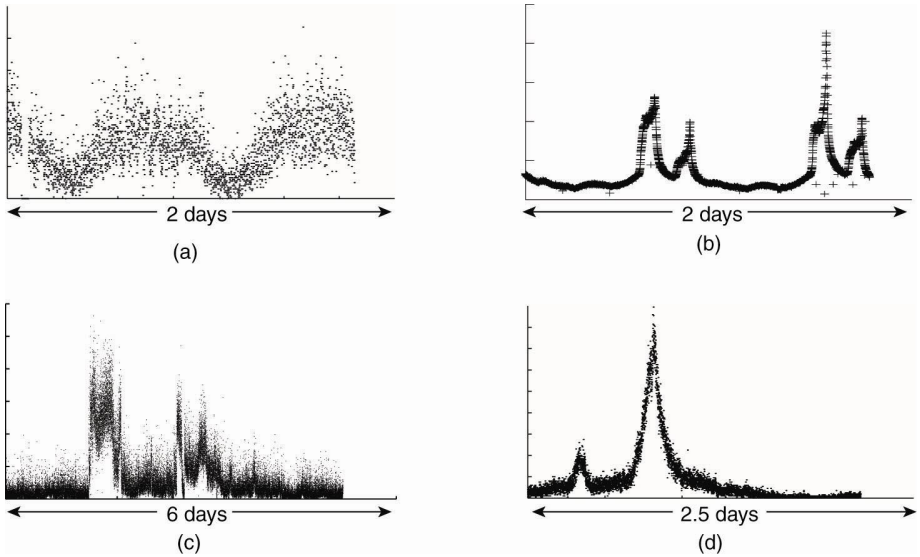


CDN-Metrics and Trade-off

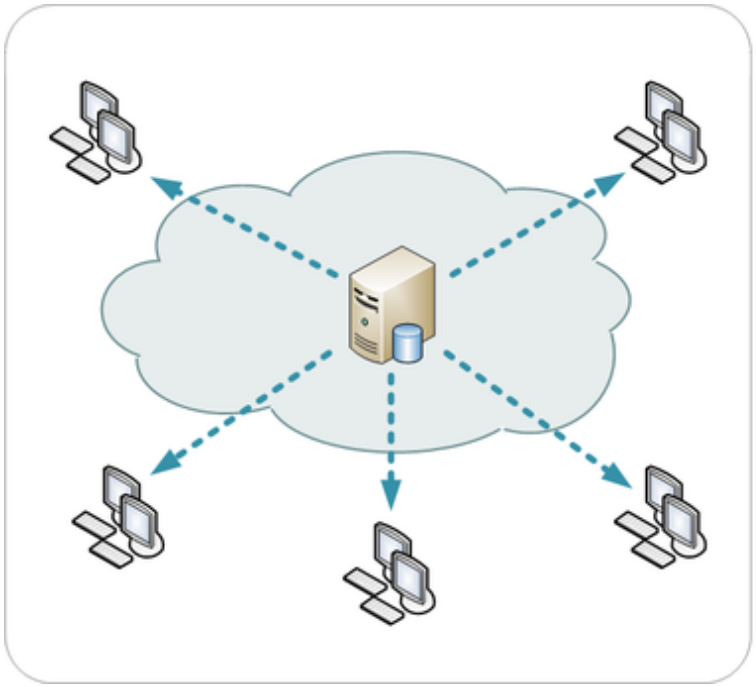
- **Access Time - Latency** Metrics and Estimation
- **Financial** Costs
- **Bandwidth** Usage - Available bandwidth between two nodes
- **Hops** between autonomous systems
- **Consistency** Metrics

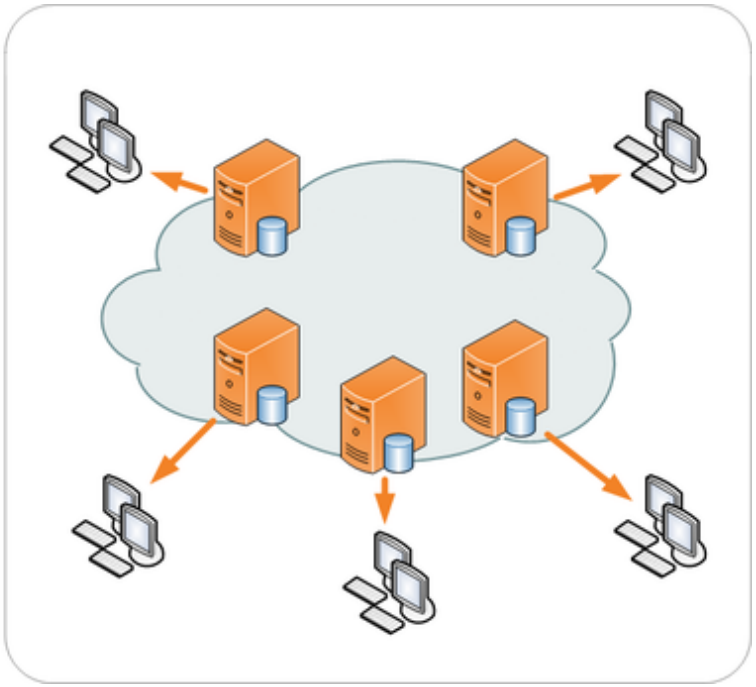
CDN-Adaptation Triggering

- When and how adaptations are triggered:
  - **Periodically** but it does not work for flash crowds
  - **Flash Crowd Predictor**



Traditional Scheme vs. CDN





Content Distribution Networks

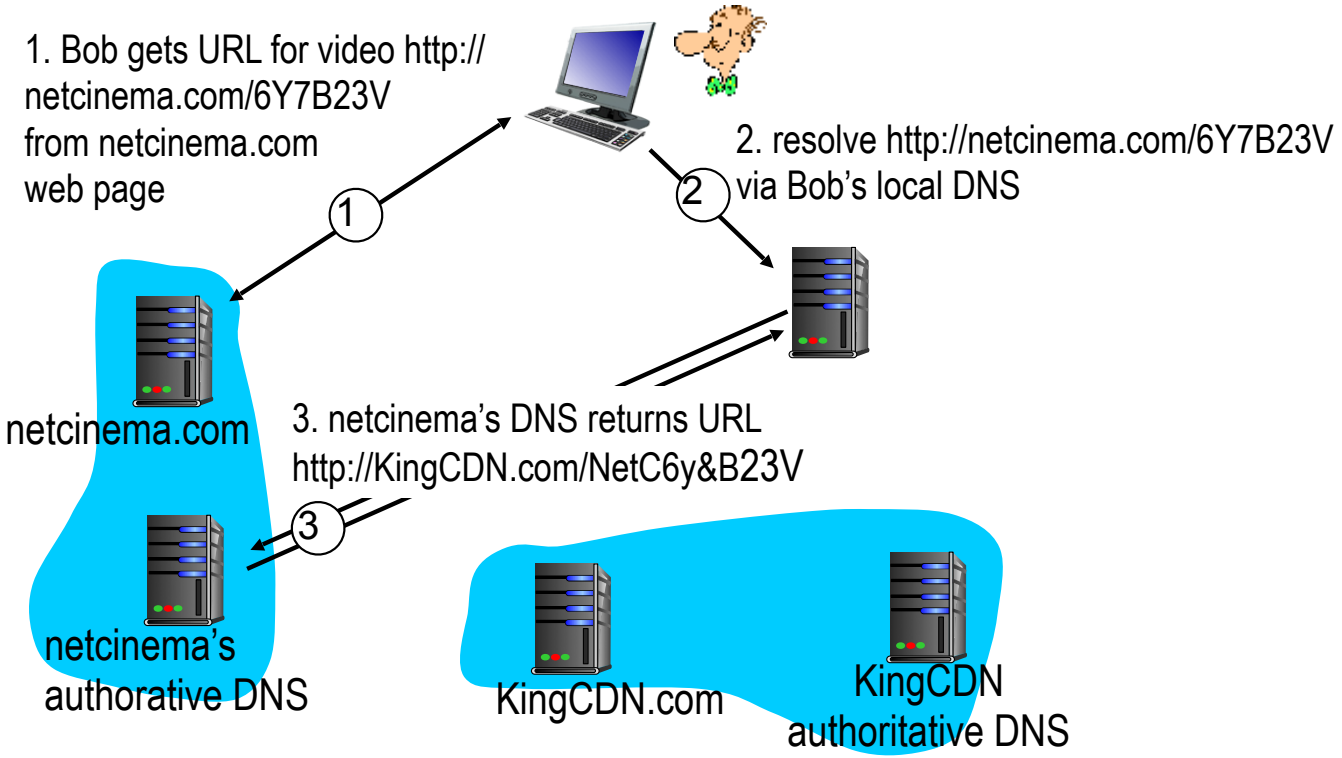
- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

- Store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)*
  - *enter deep:* push CDN servers deep into many access networks
    - close to users
    - used by Akamai, 1700 locations
  - *bring home:* smaller number (10's) of larger clusters in POPs near (but not within) access networks
    - used by Limelight

CDN: "simple" Content Access Scenario

Bob (client) requests video http://netcinema.com/6Y7B23V

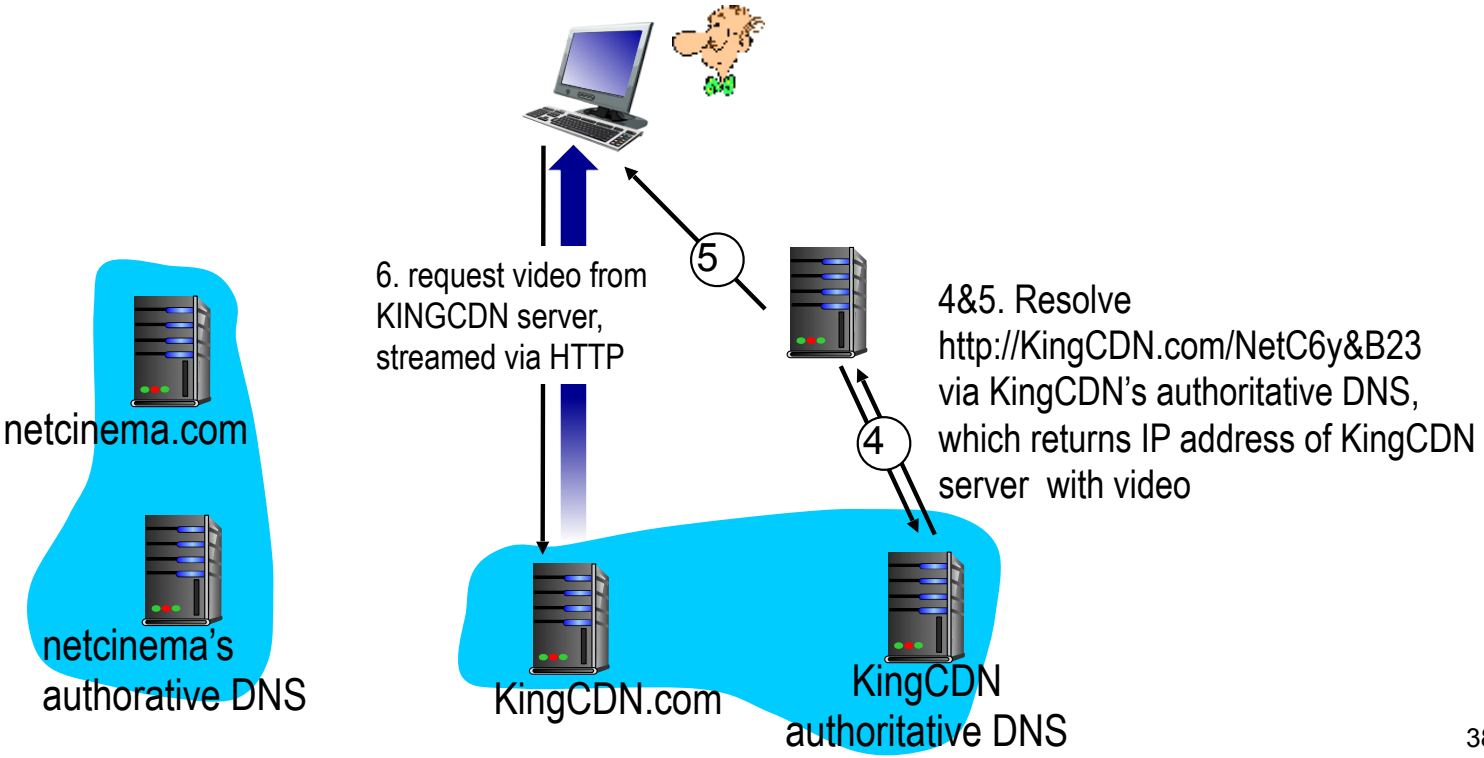
- video stored in CDN at http://KingCDN.com/NetC6y&B23V



CDN: "simple" Content Access Scenario

Bob (client) requests video http://netcinema.com/6Y7B23V

- video stored in CDN at http://KingCDN.com/NetC6y&B23V

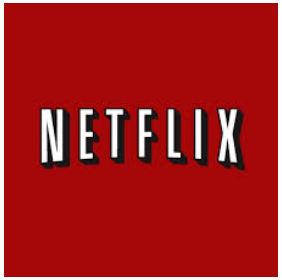


CDN Cluster Selection Strategy

- *challenge:* how does CDN DNS select "good" CDN node to stream to client
  - pick CDN node geographically closest to client
  - pick CDN node with shortest delay (or min # hops) to client (CDN nodes periodically ping access ISPs, reporting results to CDN DNS)
  - IP anycast

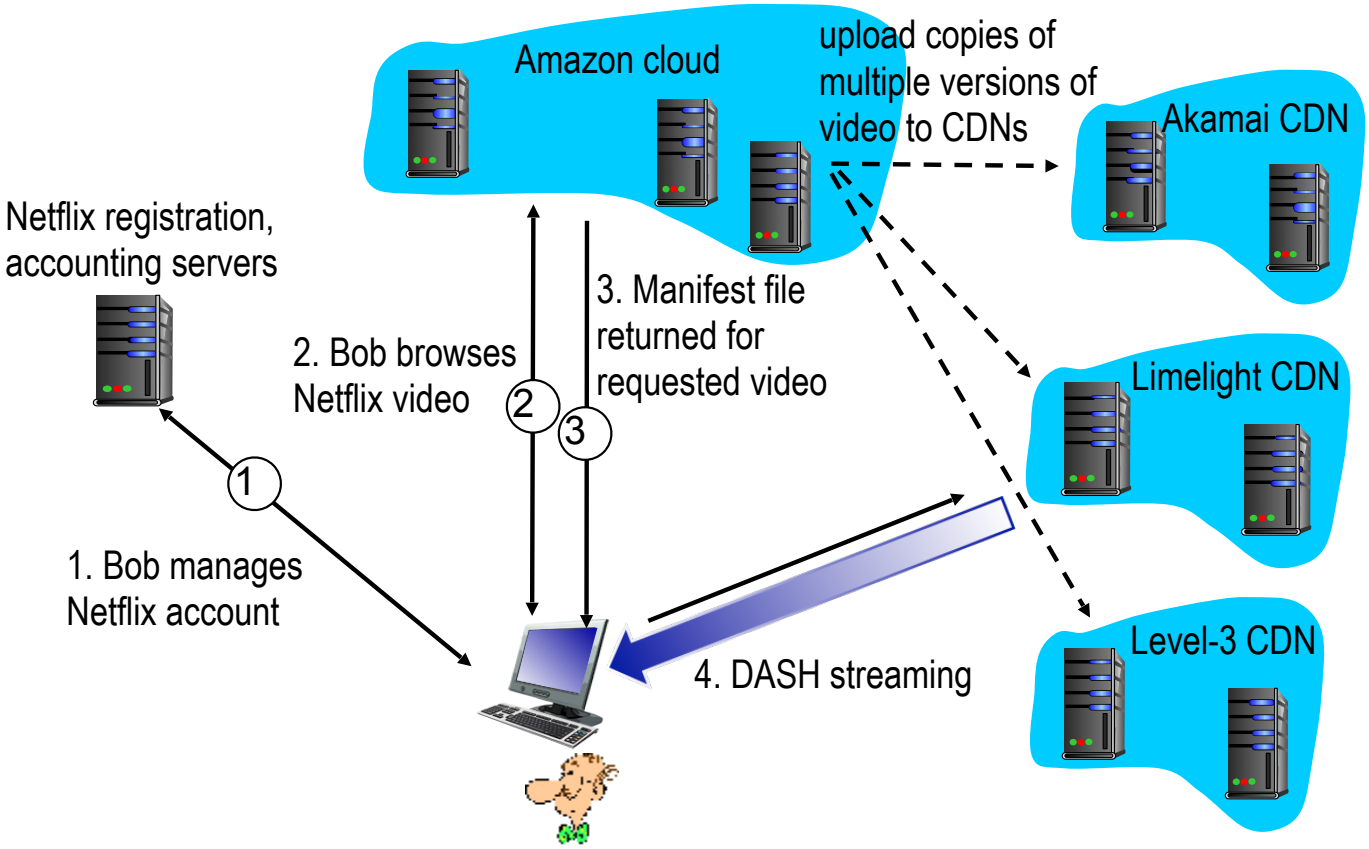
- *alternative:* let *client* decide - give client a list of several CDN servers
  - client pings servers, picks "best"
  - Netflix approach

Case study: Netflix



- 30% downstream US traffic in 2011
- owns very little infrastructure, uses 3rd party services:
  - own registration, payment servers
  - Amazon (3rd party) cloud services:
    - Netflix uploads studio master to Amazon cloud
    - create multiple version of movie (different encodings) in cloud
    - upload versions from cloud to CDNs
    - Cloud hosts Netflix web pages for user browsing
  - *Three* 3rd party CDNs host/stream Netflix content: Akamai, Limelight, Level-3

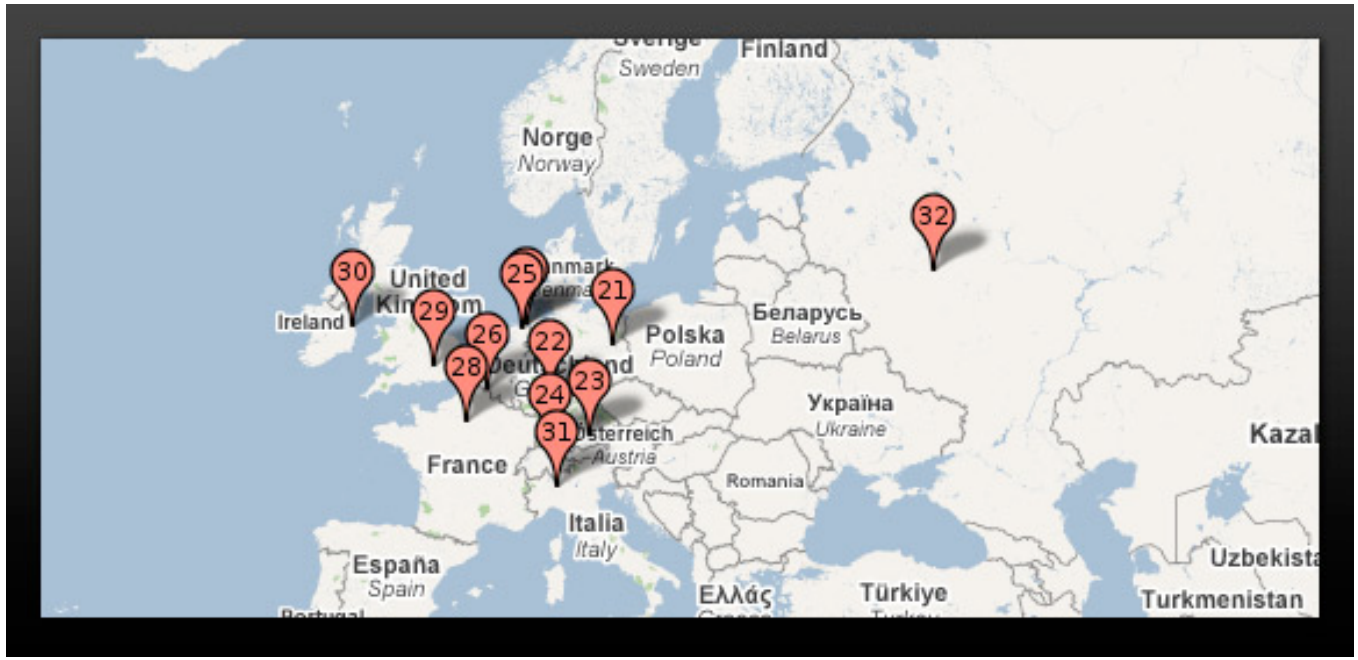
Case study: Netflix



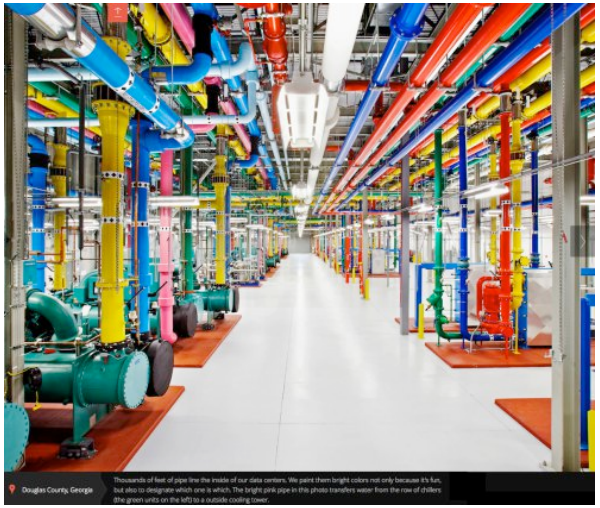
Google Data Centers

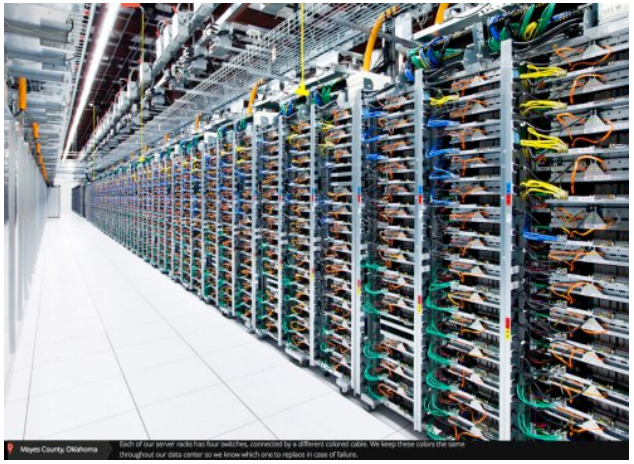
Google Data Centers

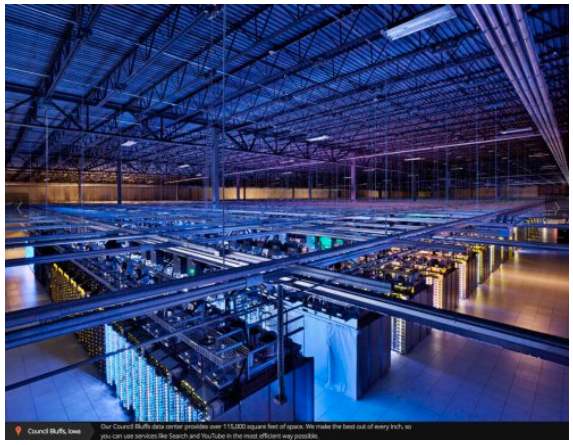
Google Data Centers



Google Data Centers







Replication of Web Applications

Web is increasingly offering more dynamically generated content, but that it is also expanding toward offering services that can be called by remote applications