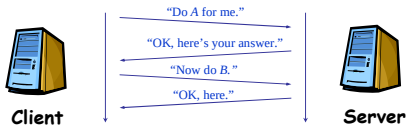
The Web and Content Networks: the Big Picture

Jeff Chase

Services



request/response paradigm ==> *client/server roles*
- Remote Procedure Call (RPC)
- object invocation, e.g., Remote Method Invocation (RMI)
- HTTP (the Web)
- device protocols (e.g., SCSI)

How does the Web work?

The canonical example in your Web browser

Click here

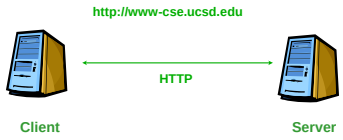
"here" is a Uniform Resource Locator (URL)

http://www-cse.ucsd.edu

It names the location of an object (document) on a server.

[courtesy of Geoff Voelker]
voelker@cs.ucsd.edu

In Action…



- Client uses DNS to resolves name of server (*www-cse.ucsd.edu*)
- Establishes an HTTP connection with the server over TCP/IP
- Sends the server the name of the object (null)
- Server returns the object

[Voelker]

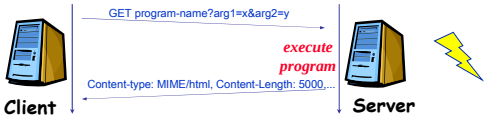
HTTP in a Nutshell

HTTP supports request/response message exchanges of arbitrary length.

Small number of request types: basically GET and POST, with supplements.
- object name, + content for POST
- optional *query string*
- optional *request headers*

Responses are self-typed objects (*documents*) with attributes and tags.
- optional *cookies*
- optional *response headers*

The Dynamic Web

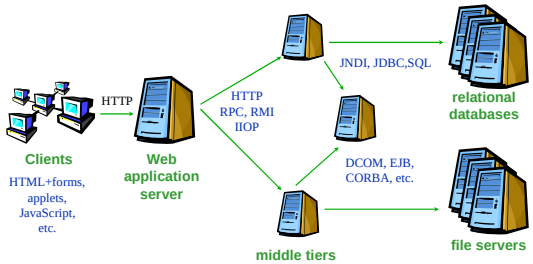


HTTP began as a souped-up FTP that supports hypertext URLs.

Service builders rapidly began using it for dynamically-generated content.

Web servers morphed into *Web Application Servers*.
- Common Gateway Interface (CGI)
- Java Servlets and JavaServer Pages (JSP)
- Microsoft Active Server Pages (ASP)
- "Web Services"

Multi-tier Services



Web Protocols

What kind of transport protocol should the Web use?

HTTP 1.0
- One TCP connection per request
- Complaints: inefficient, slow, burdensome…

HTTP 1.1
- One TCP connection/many requests (*persistent connections*)
- Solves all problems, right? Huge amount of complexity
  Clients, proxies, servers

How do they compare?
- Protocol differences [Krishnamurthy99], performance comparison [Nielsen97], effects on servers [Manley97], overhead of TCP connections [Caceres98]

HTTPS: HTTP with authentication and encryption

[Voelker]

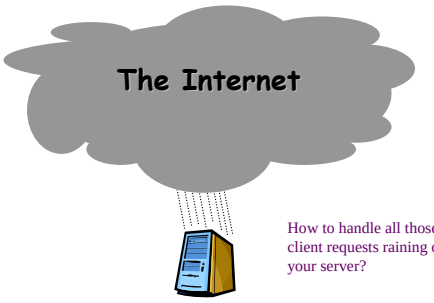
Persistent Connections

There are three key performance reasons for persistent connections:
- connection setup overhead
- TCP *slow start*: just do it and get it over with
- pipelining as an alternative to multiple connections

And some new complexities resulting from their use, e.g.:
- request/response framing and pairing
- unexpected connection breakage
  Just ask anyone from Akamai...
- large numbers of active connections
  How long to keep connections around?

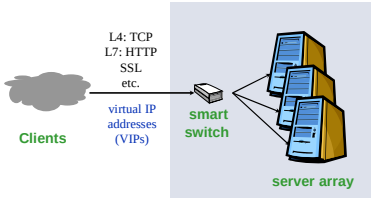
These motivations and issues manifest in HTTP, but they are fundamental for request/response messaging over TCP.

Web Service Scaling



How to handle all those client requests raining on your server?

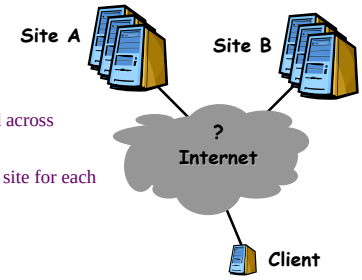
Scaling Server Sites: Clustering



Goals
server load balancing
failure detection
access control filtering
priorities/QoS
request locality
transparent caching

What to switch/filter on?
L3 source IP and/or VIP
L4 (TCP) ports etc.
L7 URLs and/or cookies
L7 SSL session IDs

Scaling Services: Replication



Distribute service load across multiple sites.

How to select a server site for each client or request?

Is it scalable?

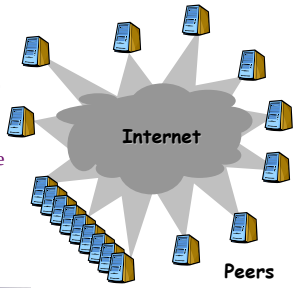
Scaling with Peer-to-Peer

Is (e.g.) Napster a service?

Is the peer-to-peer approach fundamentally more scalable?

More robust?

What does it assume about the clients?

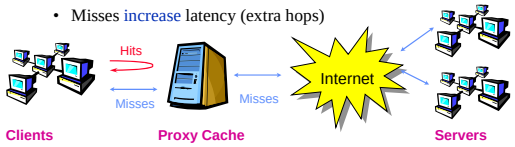


Caching for a Better Web

Performance is a major concern in the Web

Proxy caching is the most widely used method to improve Web performance

- Duplicate requests to the same document served from cache
- Hits reduce latency, bandwidth demand, server load
- Misses increase latency (extra hops)



[Source: Geoff Voelker]

Proxy Caching

How should we build caching systems for the Web?

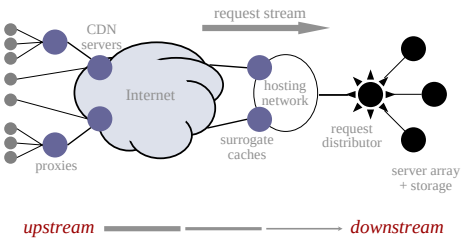
- Seminal paper [Chankhunthod96]
- Proxy caches [Duska97]
- Akamai DNS interposition [Karger99]
- Cooperative caching [Tewari99, Fan98, Wolman99]
- Popularity distributions [Breslau99]
- Proxy filtering and transcoding [Fox et al]
- Consistency [Tewari,Cao et al]
- Replica placement for CDNs [et al]

[Voelker]

Issues for Web Caching

- Binding clients to proxies, handling failover
  - Manual configuration, router-based "transparent caching", WPAD (Web Proxy Automatic Discovery)
- Proxy may confuse/obscure interactions between server and client.
- Consistency management
  - At first approximation the Web is a wide-area read-only file service...but it is much more than that.
  - caching *responses* vs. caching documents
  - deltas **[Mogul+Bala/Douglis/Misha/others@research.att.com]**
- Prefetching, scale, request routing, scale, performance
  - Web caching vs. *content distribution* (CDNs, e.g., Akamai)

End-to-End Content Delivery



Proxy Deployment and Use

Where to put it?

How to direct user Web traffic through the proxy?

Request redirection

- Much more to come on this topic…

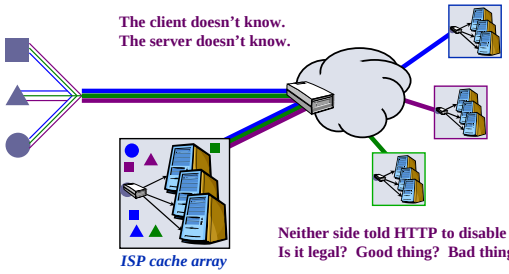
Must the server consent?

- Protected content
- Client identity

"Transparent" caching and the end-to-end principle

- Must the client consent?

Interception Switches



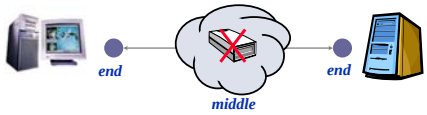
The client doesn't know.
The server doesn't know.

ISP cache array

Neither side told HTTP to disable it.
Is it legal? Good thing? Bad thing?

DUKE Systems & Architecture

Shouldn't This Be Illegal?



RFC 1122: The Internet Architecture (IPv4) specifies that each packet has a unique destination "host" address.

Problems
middle boxes may be subversive
IPsec and SSL
dynamic routing

DUKE Systems & Architecture