CS 194: Distributed Systems
WWW and Web Services

Scott Shenker and Ion Stoica
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720-1776

The Web – History (I)

Vannevar Bush (1890-1974)

- 1945: Vannevar Bush, Memex:
- *"a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility"*

(See http://www.iath.virginia.edu/elab/hfl0051.html)

Memex

The Web – History (II)

Ted Nelson

- 1967, Ted Nelson, Xanadu:
  - A world-wide publishing network that would allow information to be stored not as separate files but as connected literature
  - Owners of documents would be automatically paid via electronic means for the virtual copying of their documents
- Coined the term "Hypertext"

The Web – History (III)

Tim Berners-Lee

- World Wide Web (WWW): a distributed database of "pages" linked through Hypertext Transport Protocol (HTTP)
  - First HTTP implementation - 1990
    - Tim Berners-Lee at CERN
  - HTTP/0.9 – 1991
    - Simple GET command for the Web
  - HTTP/1.0 –1992
    - Client/Server information, simple caching
  - HTTP/1.1 - 1996

The Web

- Core components:
  - Servers: store files and execute remote commands
  - Browsers: retrieve and display "pages"
  - Uniform Resource Locators (URLs): way to refer to pages
- A protocol to transfer information between clients and servers
  - HTTP

Uniform Record Locator (URL)

```
protocol://host-name:port/directory-path/resource
```

- Extend the idea of hierarchical namespaces to include anything in a file system
  - ftp://www.cs.berkeley.edu/~istoica/cs194/05/lecture.ppt
- Extend to program executions as well…
  - http://us.f413.mail.yahoo.com/ym/ShowLetter?box=%40B%40Bulk&MsgId=2604_1744106_29699_1123_1261_0_28917_3552_1289957100&Search=&Nhead=f&YY=31454&order=down&sort=date&pos=0&view=a&head=b
  - Server side processing can be incorporated in the name

Web and DNS

- URLs use hostnames
- Thus, content names are tied to specific hosts
- This is bad!
- Uniform Resource Names (URNs) are one proposal to achieve persistence
  - Not discussed in this lecture

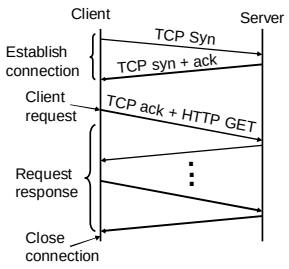
7

Hyper Text Transfer Protocol (HTTP)

- Client-server architecture
- Synchronous request/reply protocol
  - Runs over TCP, Port 80
- Stateless

8

Big Picture



9

Hyper Text Transfer Protocol Commands

- GET – transfer resource from given URL
- HEAD – GET resource metadata (headers) only
- PUT – store/modify resource under given URL
- DELETE – remove resource
- POST – provide input for a process identified by the given URL (usually used to post CGI parameters)

10

Response Codes

- 1x informational
- 2x success
- 3x redirection
- 4x client error in request
- 5x server error; can't satisfy the request

11

Client Request

- Steps to get the resource:

  http://www.eecs.berkeley.edu/index.html

  1. Use DNS to obtain the IP address of www.eecs.berkeley.edu
  2. Send to an HTTP request:

     ```
     GET /index.html HTTP/1.0
     ```

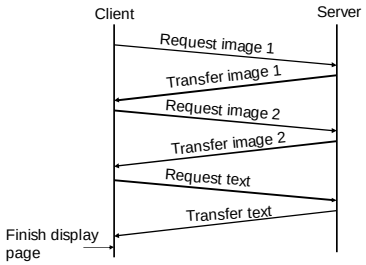
12

Server Response

```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 1234
Last-Modified: Mon, 19 Nov 2001 15:31:20 GMT
<HTML>
<HEAD>
<TITLE>EECS Home Page</TITLE>
</HEAD>
…
</BODY>
</HTML>
```

13

HTTP/1.0 Example



14

HHTP/1.0 Performance

- Create a new TCP connection for each resource
  - Large number of embedded objects in a web page
  - Many short lived connections
- TCP transfer
  - Too slow for small object
  - May never exit slow-start phase
- Connections may be set up in parallel (5 is default in most browsers)

15

HTTP/1.0 Caching Support

- Exploit locality of reference
- A modifier to the GET request:
  - If-modified-since – return a "not modified" response if resource was not modified since specified time
- A response header:
  - Expires – specify to the client for how long it is safe to cache the resource
- A request directive:
  - No-cache – ignore all caches and get resource directly from server
- These features can be best taken advantage of with HTTP proxies
  - Locality of reference increases if many clients share a proxy

16

HTTP/1.1 (1996)

- Performance:
  - Persistent connections
  - Pipelined requests/responses
  - …
- Efficient caching support
  - Network Cache assumed more explicitly in the design
  - Gives more control to the server on how it wants data cached
- Support for virtual hosting
  - Allows to run multiple web servers on the same machine

17

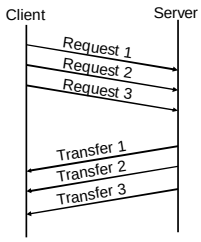
Persistent Connections

- Allow multiple transfers over one connection
- Avoid multiple TCP connection setups
- Avoid multiple TCP slow starts

18

Pipelined Requests/Responses

- Buffer requests and responses to reduce the number of packets
- Multiple requests can be contained in one TCP segment
- Note: order of responses has to be maintained



19

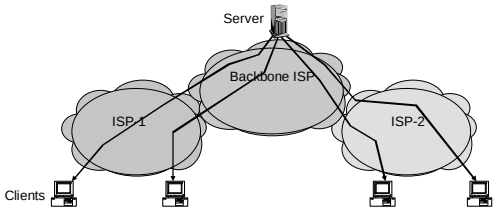
Caching and Replication

- Problem: You are a web content provider
  - How do you handle millions of web clients?
  - How do you ensure that all clients experience good performance?
  - How do you maintain availability in the presence of server and network failures?
- Solutions:
  - Add more servers at different locations → If you are CNN this might work!
  - Caching
  - Content Distribution Networks (Replication)

20

"Base-line"

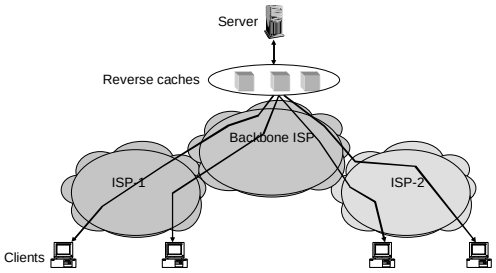
- Many clients transfer same information
  - Generate unnecessary server and network load
  - Clients experience unnecessary latency



21

Reverse Caches

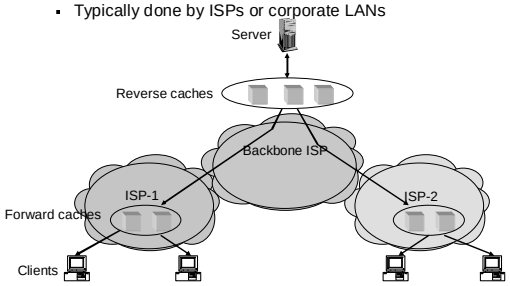
- Cache documents close to server → decrease server load
- Typically done by content providers



22

Forward Proxies

- Cache documents close to clients → reduce network traffic and decrease latency
- Typically done by ISPs or corporate LANs



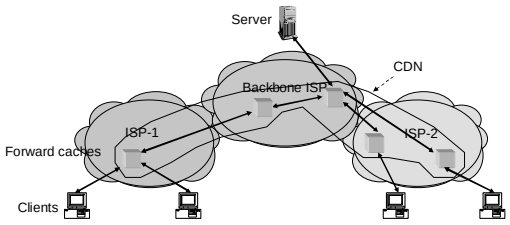
23

Content Distribution Networks (CDNs)

- Integrate forward and reverse caching functionalities into one overlay network (usually) administrated by one entity
  - Example: Akamai
- Documents are cached both
  - As a result of clients' requests (pull)
  - Pushed in the expectation of a high access rate
- Beside caching do processing, e.g.,
  - Handle dynamic web pages
  - Transcoding

24

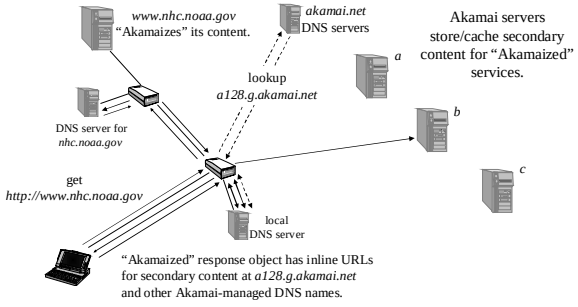
CDNs (cont'd)



Example: Akamai

- Akamai creates new domain names for each client content provider.
  - e.g., *a128.g.akamai.net*
- The CDN's DNS servers are authoritative for the new domains
- The client content provider modifies its content so that embedded URLs reference the new domains.
  - "Akamaize" content, e.g.: *http://www.cnn.com/image-of-the-day.gif* becomes *http://a128.g.akamai.net/image-of-the-day.gif*.

Example: Akamai



Core Web Technologies

- HTML
- CGI
- XML

What is HTML?

- HTML is the *lingua franca* for web publishing.
- Hyper Text Markup Language is based on SGML (Standard Generalized Markup Language)
  - HTML 4.0: http://www.w3.org/TR/html4/intro/intro.html
- Initial version invented by Tim Berners-Lee
- Originally developed for sharing scientific documents on the web

What is HTML?

- HTML documents are plain text files
- Contain text and HTML **mark-up tags**
- **Markup tags** describe elements representing the style and structure of the visual document

Markup Tags

- An HTML element may include a name, some attributes and some text or hypertext, and will appear in an HTML document as
  - <tagName> text </tagName>
  - <tagName attribute=argument> text </tagName>, or just
  - <tagName>
- Examples:

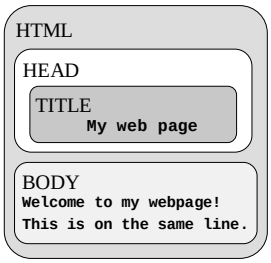
<title> My Document </title>
<a href=http://www.cs.berkeley.edu/>Berkeley CS Web page<a>

31

A trivial HTML document

```
<HTML>
  <HEAD>
    <TITLE>
     My web page
    </TITLE>
  </HEAD>
  <BODY>
   Welcome to my webpage!
   This is on the same line.
  </BODY>
</HTML>
```

Nesting structure



32

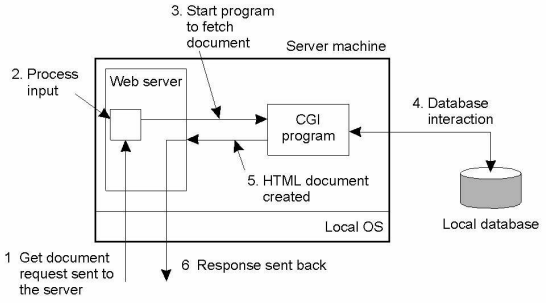
Common Gateway Interface (CGI)

- CGI – general standard specifying how programs can be run on server, from the WWW
- Any program in any language can be a CGI program - it just has to follow the CGI rules
- These rules define how programs get data (e.g., HTML form data) and how to make sure web server knows it's a CGI program
- Call of a CGI program (like any HTML page):

```
<a href="http://www.mysite/cgi-bin/myprog">
Run my CGI program </a>
```

33

Client-Server CGI Architecture



34

CGI Examples

- Any programming language can be used for CGI (e.g., shell script)
- Every CGI program must write out data to send back to web browser.
- The first thing they must write out is MIME type of file (e.g., text/plain, text/html)

```
#!/bin/sh
echo "Content-type: text/plain"
echo
echo "Hello World"
```

35

CGI and Forms

- CGI programs can process data from forms:

```
<form method="get"
  action="http://www.foo.org/cgi-bin/cgiwrap/example.cgi">
<p> Name: <input type="text" name="username" /> </p>
<p> Age: <input type="text" name="age" /> </p>
<p> <input type="submit" value="Do it" /> </p>
</form>
```

- If method="get" then the form data gets put in variable QUERY_STRING available to CGI programs

36

GET vs POST

- Using "get" method:
  - Data added to URL as ..prog?var=val etc.
  - This data is put in QUERY_STRING variable available to CGI programs
  - E.g.: http://us.f413.mail.yahoo.com/ym/ShowLetter?box=%40B%40Bulk&MsgId=2604_1744106_29699_1123_1261_0_28917_3552_1289957100&Search=&Nhead=f&YY=31454&order=down&sort=date&pos=0&view=a&head=b
- Alternative is to use "post" method:
  - Data is sent separately to URL.
  - CGI program reads this data from its standard input.

37

CGI Security

- CGI programs let anyone in the world run a program on your system
- Special *wrapper* programs may be used to do some security checks

38

XML: eXtensible Markup Language

- A simple, very flexible text format derived from SGML
- Rapidly emerging as the language of choice for data sharing on the Internet

39

XML Example

- An XML definition for referring to a journal article.

```
(1)   <!ELEMENT article (title, author+,journal)>
(2)   <!ELEMENT title (#PCDATA)>
(3)   <!ELEMENT author (name, affiliation?)>
(4)   <!ELEMENT name (#PCDATA)>
(5)   <!ELEMENT affiliation (#PCDATA)>
(6)   <!ELEMENT journal (jname, volume, number?, month? pages, year)>
(7)   <!ELEMENT jname (#PCDATA)>
(8)   <!ELEMENT volume (#PCDATA)>
(9)   <!ELEMENT number (#PCDATA)>
(10)  <!ELEMENT month (#PCDATA)>
(11)  <!ELEMENT pages (#PCDATA)>
(12)  <!ELEMENT year (#PCDATA)>
```

40

XML Example (cont'd)

- XML document using XML definitions from previous slide

```
(1)   <?xml = version "1.0">
(2)   <!DOCTYPE article SYSTEM "article.dtd">
(3)   <article>
(4)     <title> Prudent Engineering Practice for Cryptographic
Protocols</title>
(5)     <author><name>M. Abadi</name></author>
(6)     <author><name>R. Needham</name></author>
(7)     <journal>
(8)       <jname>IEEE Transactions on Software
Engineering</jname>
(9)       <volume>22</volume>
(10)      <number>12</number>
(11)      <month>January</month>
(12)      <pages>6 – 15</pages>
(13)      <year>1996</year>
(14)    </journal>
(15)  </article>
```

41

XML vs HTML?

- HTML combines structure and display, while XML separates them
  - HTML – presentation markup language: it describes the look, feel, and actions of web pages
  - XML describes document structure: what words in documents are
- Flexibility:
  - HTML – only one standard definition of all of the tags
  - XML – custom documents defining the meaning of tags
- XML may replace HTML in the future

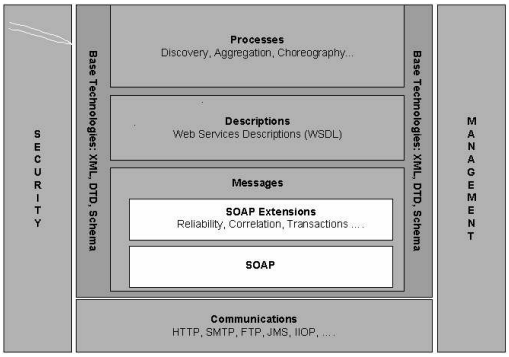
42

Web Services

- WS are applications that communicate using internet-based middleware
- WS are network-based software applications developed to interact with other applications using Internet standard technologies and connections to seamlessly perform business process

43

Web Services Architecture Stacks



44

WS Components

1. A standard way for communication (SOAP)
2. A uniform data representation and exchange mechanism (XML)
3. A standard meta language to describe the services offered (WSDL)
4. A mechanism to register and locate WS based applications (UDDI)

45

What is SOAP?

- Lightweight protocol used for exchange of messages in a decentralized, distributed environment
- Platform-independent
- Used for Remote Procedure Calls
- W3C note defines the use of SOAP with XML as payload and HTTP as transport

46

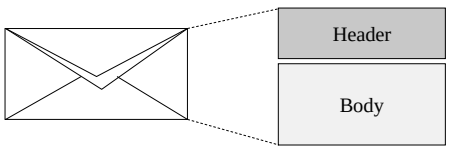
SOAP Elements

- Envelope (mandatory)
  - Top element of the XML document representing the message
- Header (optional)
  - Determines how a recipient of a SOAP message should process the message
  - Adds features to the SOAP message such as authentication, transaction management, payment, message routes, etc…
- Body (mandatory)
  - Exchanges information intended for the recipient of the message
  - Typical use is for RPC calls and error reporting

47

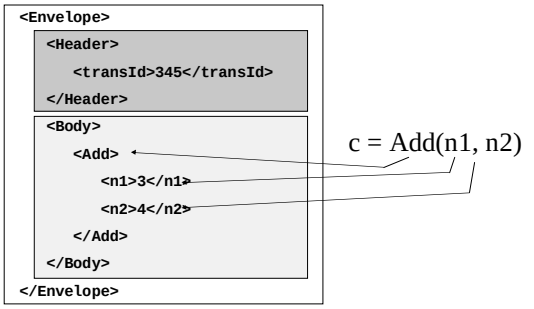
SOAP Elements

- SOAP Encoding
- Envelope package
- Header/Body pattern
  - Similar to how HTTP works



48

Simple Example



49

SOAP Request

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Header>
      <t:transId xmlns:t="http://a.com/trans">345</t:transId>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <m:Add xmlns:m="http://a.com/Calculator">
         <n1>3</n1>
         <n2>4</n2>
      </m:Add>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

50

SOAP Request

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Header>
      <t:transId xmlns:t="http://a
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <m:Add xmlns:m="http://a
         <n1>3</n1>
         <n2>4</n2>
      </m:Add>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Scopes the message to the SOAP namespace describing the SOAP envelope

Establishes the type of encoding that is used within the message (different data types supported)

51

SOAP Request

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Header>
      <t:transId xmlns:t="http://a.com/trans">345</t:transId>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <m:Add xmlns:m="http://a.com/Calculator">
         <n1>3</n1>
         <n2>4</n2>
      </m:Add>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Qualifies transaction Id

Defines the method

52

SOAP Response

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encodir
   <SOAP-ENV:Header>
      <t:transId xmlns:t="http://a.com/trans">345</t:transId>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <m:AddResponse xmlns:m="http://a.com/Calculator">
         <result>7</result>
      </m:AddResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response typically uses method name with "Response" appended

53

XML-RPC vs SOAP

- XML-RPC: lower common denominator form of communication
  - Simple, easy to understand (only 7 pages specification)
- SOAP: can transfer more sophisticated information (could define virtually any data structure)
  - Flexible, but complex
  - Supported by industry

54

WSDL

- Web Services Description Language is an XML document
- Describes WS functionality
- How WS communicate & where it is accessible (What, Where & How)

55

UDDI

- Universal Description Definition Interface
- A standard discovery mechanism for WS
- Users can query a UDDI registry (company name, service type, Industry category or other criteria)
- Provides pointers to WSDL document
- UDDI is also based on XML

56