Application Layer
Chapter 7

• DNS – Domain Name System
• Electronic Mail
• The Web
• Streaming Audio and Video
• Content Delivery

Revised: August 2011
The Application Layer

Uses transport services to build distributed applications
DNS – Domain Name System

The DNS resolves high-level human readable names for computers to low-level IP addresses

- DNS name space »
- Domain Resource records »
- Name servers »
The DNS Name Space (1)

DNS namespace is hierarchical from the root down
• Different parts delegated to different organizations

The computer robot.cs.washington.edu
Generic top-level domains are controlled by ICANN who appoints registrars to run them.

This one was controversial.
Domain Resource Records (1)

The key resource records in the namespace are IP addresses (A/AAAA) and name servers (NS), but there are others too (e.g., MX)

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOA</td>
<td>Start of authority</td>
<td>Parameters for this zone</td>
</tr>
<tr>
<td>A</td>
<td>IPv4 address of a host</td>
<td>32-Bit integer</td>
</tr>
<tr>
<td>AAAA</td>
<td>IPv6 address of a host</td>
<td>128-Bit integer</td>
</tr>
<tr>
<td>MX</td>
<td>Mail exchange</td>
<td>Priority, domain willing to accept email</td>
</tr>
<tr>
<td>NS</td>
<td>Name server</td>
<td>Name of a server for this domain</td>
</tr>
<tr>
<td>CNAME</td>
<td>Canonical name</td>
<td>Domain name</td>
</tr>
<tr>
<td>PTR</td>
<td>Pointer</td>
<td>Alias for an IP address</td>
</tr>
<tr>
<td>SPF</td>
<td>Sender policy framework</td>
<td>Text encoding of mail sending policy</td>
</tr>
<tr>
<td>SRV</td>
<td>Service</td>
<td>Host that provides it</td>
</tr>
<tr>
<td>TXT</td>
<td>Text</td>
<td>Descriptive ASCII text</td>
</tr>
</tbody>
</table>
## Domain Resource Records (2)

A portion of a possible DNS database for cs.vu.nl.

```plaintext
; Authoritative data for cs.vu.nl
 cs.vu.nl. 86400 IN SOA star boss (9527,7200,7200,241920,86400)
 cs.vu.nl. 86400 IN MX 1 zephyr
 cs.vu.nl. 86400 IN MX 2 top
 cs.vu.nl. 86400 IN NS star

star 86400 IN A 130.37.56.205
zephyr 86400 IN A 130.37.20.10
top 86400 IN A 130.37.20.11
www 86400 IN CNAME star.cs.vu.nl
ftp 86400 IN CNAME zephyr.cs.vu.nl

flits 86400 IN A 130.37.16.112
flits 86400 IN A 192.31.231.165
flits 86400 IN MX 1 flits
flits 86400 IN MX 2 zephyr
flits 86400 IN MX 3 top

rowboat IN A 130.37.56.201
 IN MX 1 rowboat
 IN MX 2 zephyr

little-sister IN A 130.37.62.23

laserjet IN A 192.31.231.216
```

- **Name server**
- **IP addresses of computers**
- **Mail gateways**
Name Servers (1)

Name servers contain data for portions of the name space called zones (circled).
Finding the IP address for a given hostname is called resolution and is done with the DNS protocol.

Resolution:
• Computer requests local name server to resolve
• Local name server asks the root name server
• Root returns the name server for a lower zone
• Continue down zones until name server can answer

DNS protocol:
• Runs on UDP port 53, retransmits lost messages
• Caches name server answers for better performance
Name Servers (3)

Example of a computer looking up the IP for a name
Electronic Mail

- Architecture and services »
- The user agent »
- Message formats »
- Message transfer »
- Final delivery »
The key components and steps (numbered) to send email

1: Mail submission
2: Message transfer
3: Final delivery

Architecture of the email system
Architecture and Services (2)

Paper mail

Electronic mail

Envelope

Message (= header and body)
The User Agent

What users see – interface elements of a typical user agent
Message Formats (1)

Header fields related to message transport; headers are readable ASCII text

<table>
<thead>
<tr>
<th>Header</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>To:</td>
<td>Email address(es) of primary recipient(s)</td>
</tr>
<tr>
<td>Cc:</td>
<td>Email address(es) of secondary recipient(s)</td>
</tr>
<tr>
<td>Bcc:</td>
<td>Email address(es) for blind carbon copies</td>
</tr>
<tr>
<td>From:</td>
<td>Person or people who created the message</td>
</tr>
<tr>
<td>Sender:</td>
<td>Email address of the actual sender</td>
</tr>
<tr>
<td>Received:</td>
<td>Line added by each transfer agent along the route</td>
</tr>
<tr>
<td>Return-Path:</td>
<td>Can be used to identify a path back to the sender</td>
</tr>
</tbody>
</table>
Other header fields useful for user agents

<table>
<thead>
<tr>
<th>Header</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date:</td>
<td>The date and time the message was sent</td>
</tr>
<tr>
<td>Reply-To:</td>
<td>Email address to which replies should be sent</td>
</tr>
<tr>
<td>Message-Id:</td>
<td>Unique number for referencing this message later</td>
</tr>
<tr>
<td>In-Reply-To:</td>
<td>Message-Id of the message to which this is a reply</td>
</tr>
<tr>
<td>References:</td>
<td>Other relevant Message-Ids</td>
</tr>
<tr>
<td>Keywords:</td>
<td>User-chosen keywords</td>
</tr>
<tr>
<td>Subject:</td>
<td>Short summary of the message for the one-line display</td>
</tr>
</tbody>
</table>
Message Formats (3)

MIME header fields used to describe what content is in the body of the message

<table>
<thead>
<tr>
<th>Header</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIME-Version:</td>
<td>Identifies the MIME version</td>
</tr>
<tr>
<td>Content-Description:</td>
<td>Human-readable string telling what is in the message</td>
</tr>
<tr>
<td>Content-Id:</td>
<td>Unique identifier</td>
</tr>
<tr>
<td>Content-Transfer-Encoding:</td>
<td>How the body is wrapped for transmission</td>
</tr>
<tr>
<td>Content-Type:</td>
<td>Type and format of the content</td>
</tr>
</tbody>
</table>
Message Formats (4)

Common MIME content types and subtypes

<table>
<thead>
<tr>
<th>Type</th>
<th>Example subtypes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>plain, html, xml, css</td>
<td>Text in various formats</td>
</tr>
<tr>
<td>image</td>
<td>gif, jpeg, tiff</td>
<td>Pictures</td>
</tr>
<tr>
<td>audio</td>
<td>basic, mpeg, mp4</td>
<td>Sounds</td>
</tr>
<tr>
<td>video</td>
<td>mpeg, mp4, quicktime</td>
<td>Movies</td>
</tr>
<tr>
<td>model</td>
<td>vrml</td>
<td>3D model</td>
</tr>
<tr>
<td>application</td>
<td>octet-stream, pdf, javascript, zip</td>
<td>Data produced by applications</td>
</tr>
<tr>
<td>message</td>
<td>http, rfc822</td>
<td>Encapsulated message</td>
</tr>
<tr>
<td>multipart</td>
<td>mixed, alternative, parallel, digest</td>
<td>Combination of multiple types</td>
</tr>
</tbody>
</table>
Putting it all together:
a multipart message containing HTML and audio alternatives.

One part (HTML)

```
<p>Happy birthday to you<br>
Happy birthday to you<br>
Happy birthday dear <b>Bob</b> <br>
Happy birthday to you<br>
```

Another (audio)

```
content-type: audio/basic
content-transfer-encoding: base64
--qwertuyiopasdfghjklzxcvbnm--
```
Message Transfer (1)

Messages are transferred with SMTP (Simple Mail Transfer Protocol)

- Readable text commands
- Submission from user agent to MTA on port 587
- One MTA to the next MTA on port 25
- Other protocols for final delivery (IMAP, POP3)
Message Transfer (2)

Sending a message:

- From Alice to Bob
- SMTP commands are marked [pink]

```plaintext
S: 220 ee.uwa.edu.au SMTP service ready
C: HELO abcd.com
    S: 250 cs.washington.edu says hello to ee.uwa.edu.au
C: MAIL FROM: <alice@cs.washington.edu>
    S: 250 sender ok
C: RCPT TO: <bob@ee.uwa.edu.au>
    S: 250 recipient ok
C: DATA
    S: 354 Send mail; end with "." on a line by itself
C: From: alice@cs.washington.edu
C: To: bob@ee.uwa.edu.au
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@ee.uwa.edu.au>
C: Content-Type: multipart/alternative; boundary=qwertuiopasdfghjklzxcvbnm
C: Subject: Earth orbits sun integral number of times
C:
C: This is the preamble. The user agent ignores it. Have a nice day.
C:
C: --qwertuiopasdfghjklzxcvbnm
C: Content-Type: text/html
C:
C: <p>Happy birthday to you
C: Happy birthday to you

. . . (rest of message) . . .

C: --qwertuiopasdfghjklzxcvbnm
C: .
C: QUIT
    S: 250 message accepted
S: 221 ee.uwa.edu.au closing connection
```
Message Transfer (3)

Common SMTP extensions (not in simple example)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTH</td>
<td>Client authentication</td>
</tr>
<tr>
<td>BINARYMIME</td>
<td>Server accepts binary messages</td>
</tr>
<tr>
<td>CHUNKING</td>
<td>Server accepts large messages in chunks</td>
</tr>
<tr>
<td>SIZE</td>
<td>Check message size before trying to send</td>
</tr>
<tr>
<td>STARTTLS</td>
<td>Switch to secure transport (TLS; see Chap. 8)</td>
</tr>
<tr>
<td>UTF8SMTP</td>
<td>Internationalized addresses</td>
</tr>
</tbody>
</table>
Final Delivery (1)

User agent uses protocol like IMAP for final delivery

- Has commands to manipulate folders / messages [right]

Alternatively, a Web interface (with proprietary protocol) might be used

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPABILITY</td>
<td>List server capabilities</td>
</tr>
<tr>
<td>STARTTLS</td>
<td>Start secure transport (TLS; see Chap. 8)</td>
</tr>
<tr>
<td>LOGIN</td>
<td>Log on to server</td>
</tr>
<tr>
<td>AUTHENTICATE</td>
<td>Log on with other method</td>
</tr>
<tr>
<td>SELECT</td>
<td>Select a folder</td>
</tr>
<tr>
<td>EXAMINE</td>
<td>Select a read-only folder</td>
</tr>
<tr>
<td>CREATE</td>
<td>Create a folder</td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete a folder</td>
</tr>
<tr>
<td>RENAME</td>
<td>Rename a folder</td>
</tr>
<tr>
<td>SUBSCRIBE</td>
<td>Add folder to active set</td>
</tr>
<tr>
<td>LIST</td>
<td>List the available folders</td>
</tr>
<tr>
<td>LSUB</td>
<td>List the active folders</td>
</tr>
<tr>
<td>STATUS</td>
<td>Get the status of a folder</td>
</tr>
<tr>
<td>APPEND</td>
<td>Add a message to a folder</td>
</tr>
<tr>
<td>CHECK</td>
<td>Get a checkpoint of a folder</td>
</tr>
<tr>
<td>FETCH</td>
<td>Get messages from a folder</td>
</tr>
<tr>
<td>SEARCH</td>
<td>Find messages in a folder</td>
</tr>
<tr>
<td>STORE</td>
<td>Alter message flags</td>
</tr>
<tr>
<td>COPY</td>
<td>Make a copy of a message in a folder</td>
</tr>
<tr>
<td>EXPUNGE</td>
<td>Remove messages flagged for deletion</td>
</tr>
<tr>
<td>UID</td>
<td>Issue commands using unique identifiers</td>
</tr>
<tr>
<td>NOOP</td>
<td>Do nothing</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Remove flagged messages and close folder</td>
</tr>
<tr>
<td>LOGOUT</td>
<td>Log out and close connection</td>
</tr>
</tbody>
</table>

CN5E by Tanenbaum & Wetherall, © Pearson Education-Prentice Hall and D. Wetherall, 2011
The World Wide Web

- Architectural overview »
- Static Web pages »
- Dynamic pages and Web applications »
- HTTP – HyperText Transfer Protocol »
- The mobile Web »
- Web search »
Architectural Overview (1)

HTTP transfers pages from servers to browsers
Pages are named with URLs (Uniform Resource Locators)

- Example: http://www.phdcomics.com/comics.php

<table>
<thead>
<tr>
<th>Name</th>
<th>Used for</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>http</td>
<td>Hypertext (HTML)</td>
<td><a href="http://www.ee.uwa.edu/~rob/">http://www.ee.uwa.edu/~rob/</a></td>
</tr>
<tr>
<td>https</td>
<td>Hypertext with security</td>
<td><a href="https://www.bank.com/accounts/">https://www.bank.com/accounts/</a></td>
</tr>
<tr>
<td>file</td>
<td>Local file</td>
<td>file:///usr/suzanne/prog.c</td>
</tr>
<tr>
<td>mailto</td>
<td>Sending email</td>
<td><a href="mailto:JohnUser@acm.org">mailto:JohnUser@acm.org</a></td>
</tr>
<tr>
<td>rtsp</td>
<td>Streaming media</td>
<td>rtsp://youtube.com/montypython.mpg</td>
</tr>
<tr>
<td>sip</td>
<td>Multimedia calls</td>
<td>sip:<a href="mailto:eve@adversary.com">eve@adversary.com</a></td>
</tr>
<tr>
<td>about</td>
<td>Browser information</td>
<td>about:plugins</td>
</tr>
</tbody>
</table>

Our focus is on the protocol "http"
Architectural Overview (3)

Steps a client (browser) takes to follow a hyperlink:
- Determine the protocol (HTTP)
- Ask DNS for the IP address of server
- Make a TCP connection to server
- Send request for the page; server sends it back
- Fetch other URLs as needed to display the page
- Close idle TCP connections

Steps a server takes to serve pages:
- Accept a TCP connection from client
- Get page request and map it to a resource (e.g., file name)
- Get the resource (e.g., file from disk)
- Send contents of the resource to the client.
- Release idle TCP connections
Content type is identified by MIME types

- Browser takes the appropriate action to display
- Plug-ins / helper apps extend browser for new types
To scale performance, Web servers can use:

- Caching, multiple threads, and a front end
Architectural Overview (6)

Server steps, revisited:
• Resolve name of Web page requested
• Perform access control on the Web page
• Check the cache
• Fetch requested page from disk or run program
• Determine the rest of the response
• Return the response to the client
• Make an entry in the server log
Cookies support stateful client/server interactions

- Server sends cookies (state) with page response
- Client stores cookies across page fetches
- Client sends cookies back to server with requests

<table>
<thead>
<tr>
<th>Domain</th>
<th>Path</th>
<th>Content</th>
<th>Expires</th>
<th>Secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>toms-casino.com</td>
<td>/</td>
<td>CustomerID=297793521</td>
<td>15-10-10 17:00</td>
<td>Yes</td>
</tr>
<tr>
<td>jills-store.com</td>
<td>/</td>
<td>Cart=1-00501;1-07031;2-13721</td>
<td>11-1-11 14:22</td>
<td>No</td>
</tr>
<tr>
<td>aportal.com</td>
<td>/</td>
<td>Prefs=Stk:CSCO+ORCL;Spt:Jets</td>
<td>31-12-20 23:59</td>
<td>No</td>
</tr>
<tr>
<td>sneaky.com</td>
<td>/</td>
<td>UserID=4627239101</td>
<td>31-12-19 23:59</td>
<td>No</td>
</tr>
</tbody>
</table>

Examples of cookies
Static Web Pages (1)

Static Web pages are simply files
• Have the same contents for each viewing

Can be visually rich and interactive nonetheless:
• HTML that mixes text and images
• Forms that gather user input
• Style sheets that tailor presentation
• Vector graphics, videos, and more (over) . . .
### Progression of features through HTML 5.0

<table>
<thead>
<tr>
<th>Item</th>
<th>HTML 1.0</th>
<th>HTML 2.0</th>
<th>HTML 3.0</th>
<th>HTML 4.0</th>
<th>HTML 5.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyperlinks</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Images</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Lists</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Active maps &amp; images</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Forms</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Equations</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Toolbars</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Tables</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Accessibility features</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Object embedding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Style sheets</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scripting</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Video and audio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Inline vector graphics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML representation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Background threads</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Browser storage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Drawing canvas</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
Dynamic Pages & Web Applications (1)

Dynamic pages are generated by programs running at the server (with a database) and the client:

- E.g., PHP at server, JavaScript at client
- Pages vary each time like using an application
Dynamic Pages & Web Applications (2)

Web page that gets form input and calls a server program:

```html
<html>
<body>
<form action="action.php" method="post">
<p>Please enter your name: <input type="text" name="name"></p>
<p>Please enter your age: <input type="text" name="age"></p>
<input type="submit">
</form>
</body>
</html>
```

PHP server program that creates a custom Web page:

```php
<?php echo $name; ?>.
Prediction: next year you will be <?php echo $age + 1; ?>
```

Resulting Web page (for inputs “Barbara” and “32”):

```html
<html>
<body>
<h1>Reply: </h1>
Hello Barbara.
Prediction: next year you will be 33
</body>
</html>
```
Dynamic Pages & Web Applications (3)

First page with form, gets input and calls program above

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
    var person = test_form.name.value;
    var years = eval(test_form.age.value) + 1;
    document.open();
    document.writeln("<html> <body> ");
    document.writeln("Hello " + person + ".<br>");
    document.writeln("Prediction: next year you will be " + years + ".");
    document.writeln("</body> </html>");
    document.close();
}
</script>
</head>

<body>
<form>
Please enter your name: <input type="text" name="name">
<p>
Please enter your age: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```
Dynamic Pages & Web Applications (4)

The difference between server and client programs

Server-side scripting with PHP

Client-side scripting with JavaScript
Dynamic Pages & Web Applications (5)

Web applications use a set of technologies that work together, e.g. AJAX:

• HTML: present information as pages.
• DOM: change parts of pages while they are viewed.
• XML: let programs exchange data with the server.
• Asynchronous way to send and retrieve XML data.
• JavaScript as a language to bind all this together.
The DOM (Document Object Model) tree represents Web pages as a structure that programs can alter.
Dynamic Pages & Web Applications (7)

XML captures document structure, not presentation like HTML. Ex:

```xml
<?xml version="1.0" ?>
<book_list>
  <book>
    <title>Human Behavior and the Principle of Least Effort</title>
    <author>George Zipf</author>
    <year>1949</year>
  </book>
  <book>
    <title>The Mathematical Theory of Communication</title>
    <author>Claude E. Shannon</author>
    <author>Warren Weaver</author>
    <year>1949</year>
  </book>
  <book>
    <title>Nineteen Eighty-Four</title>
    <author>George Orwell</author>
    <year>1949</year>
  </book>
</book_list>
```
Web applications use a set of technologies, revisited:
HTTP (HyperText Transfer Protocol) is a request-response protocol that runs on top of TCP

- Fetches pages from server to client
- Server usually runs on port 80
- Headers are given in readable ASCII
- Content is described with MIME types
- Protocol has support for pipelining requests
- Protocol has support for caching
HTTP uses persistent connections to improve performance

- One connection for each request
- Sequential requests on one connection
- Pipelined requests on one connection
HTTP has several request methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Read a Web page</td>
</tr>
<tr>
<td>HEAD</td>
<td>Read a Web page’s header</td>
</tr>
<tr>
<td>POST</td>
<td>Append to a Web page</td>
</tr>
<tr>
<td>PUT</td>
<td>Store a Web page</td>
</tr>
<tr>
<td>DELETE</td>
<td>Remove the Web page</td>
</tr>
<tr>
<td>TRACE</td>
<td>Echo the incoming request</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Connect through a proxy</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>Query options for a page</td>
</tr>
</tbody>
</table>
Response codes tell the client how the request fared:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1xx</td>
<td>Information</td>
<td>100 = server agrees to handle client’s request</td>
</tr>
<tr>
<td>2xx</td>
<td>Success</td>
<td>200 = request succeeded; 204 = no content present</td>
</tr>
<tr>
<td>3xx</td>
<td>Redirection</td>
<td>301 = page moved; 304 = cached page still valid</td>
</tr>
<tr>
<td>4xx</td>
<td>Client error</td>
<td>403 = forbidden page; 404 = page not found</td>
</tr>
<tr>
<td>5xx</td>
<td>Server error</td>
<td>500 = internal server error; 503 = try again later</td>
</tr>
</tbody>
</table>
HTTP (5)

Many headers carry key information:

<table>
<thead>
<tr>
<th>Function</th>
<th>Example Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser capabilities (client → server)</td>
<td>User-Agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language</td>
</tr>
<tr>
<td>Caching related (mixed directions)</td>
<td>If-Modified-Since, If-None-Match, Date, Last-Modified, Expires, Cache-Control, ETag</td>
</tr>
<tr>
<td>Browser context (client → server)</td>
<td>Cookie, Referer, Authorization, Host</td>
</tr>
<tr>
<td>Content delivery (server → client)</td>
<td>Content-Encoding, Content-Length, Content-Type, Content-Language, Content-Range, Set-Cookie</td>
</tr>
</tbody>
</table>
HTTP (6)

HTTP caching checks to see if the browser has a known fresh copy, and if not if the server has updated the page

- Uses a collection of headers for the checks
- Can include further levels of caching (e.g., proxy)
The Mobile Web

Mobiles (phones, tablets) are challenging as clients:
- Relatively small screens
- Limited input capabilities, lengthy input.
- Network bandwidth is limited
- Connectivity may be intermittent.
- Computing power is limited

Strategies to handle them:
- Content: servers provide mobile-friendly versions; transcoding can also be used
- Protocols: no real need for specialized protocols; HTTP with header compression sufficient
Web Search

Search has proved hugely popular, in tandem with advertising that has proved hugely profitable

- A simple interface for users to navigate the Web

Search engine requires:

- Content from all sites, accessed by **crawling**. Follow links to new pages, but beware programs.
- Indexing, which benefits from known and discovered structure (such as XML) to increase relevance
Streaming Audio and Video

Audio and video have become key types of traffic, e.g., voice over IP, and video streaming.

- Digital audio
- Digital video
- Streaming stored media
- Streaming live media
- Real-time conferencing
Digital Audio (1)

ADC (Analog-to-Digital Converter) produces digital audio from a microphone

- Telephone: 8000 8-bit samples/second (64 Kbps); computer audio is usually better quality (e.g., 16 bit)
Digital Audio (2)

Digital audio is typically compressed before it is sent
- Lossy encoders (like AAC) exploit human perception
- Large compression ratios (can be >10X)

Sensitivity of the ear varies with frequency
A loud tone can mask nearby tones
Digital Video (1)

Video is digitized as pixels (sampled, quantized)
- TV quality: 640x480 pixels, 24-bit color, 30 times/sec

Video is sent compressed due to its large bandwidth
- Lossy compression exploits human perception
  - E.g., JPEG for still images, MPEG, H.264 for video
- Large compression ratios (often 50X for video)
- Video is normally > 1 Mbps, versus >10 kbps for speech and >100 kbps for music
Digital Video (2)

JPEG lossy compression sequence for one image:

Step 1: Block preparation
Step 2: Discrete cosine transform
Step 3: Quantization
Step 4: Differential quantization
Step 5: Run-length encoding
Step 6: Statistical output encoding
Output
Digital Video (3)

Step 1: Pixels are mapped to luminance/chrominance (YCbCr) color space and chrominance is sub-sampled
- The eye is less sensitive to chrominance

Input 24-bit RGB pixels

8-bit luminance pixels

8-bit chrominances for every 4 pixels
Digital Video (4)

Step 2: Each component block is transformed to spatial frequencies with DCT (Discrete Cosine Transformation)

- Captures the key image features
Step 3: DCT coefficients are quantized by dividing by thresholds; reduces bits in higher spatial frequencies.

- Top left element is differenced over blocks (Step 4)
Digital Video (6)

Step 5: The block is run-length encoded in a zig-zag order. Then it is Huffman coded before sending (Step 6)

Order in which the block coefficients are sent
Digital Video (7)

MPEG compresses over a sequence of frames, further using motion tracking to remove temporal redundancy
- I (Intra-coded) frames are self-contained
- P (Predictive) frames use block motion predictions
- B (Bidirectional) frames may base prediction on future frame

Three consecutive frames with stationary and moving components
Streaming Stored Media (1)

A simple method to stream stored media, e.g., for video on demand, is to fetch the video as a file download.

- But has large startup delay, except for short files.
Effective streaming starts the playout during transport

- With RTSP (Real-Time Streaming Protocol)
Streaming Stored Media (3)

Key problem: how to handle transmission errors

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use reliable transport (TCP)</td>
<td>Repairs all errors</td>
<td>Increases jitter significantly</td>
</tr>
<tr>
<td>Add FEC (e.g., parity)</td>
<td>Repairs most errors</td>
<td>Increases overhead, decoding complexity and jitter</td>
</tr>
<tr>
<td>Interleave media</td>
<td>Masks most errors</td>
<td>Slightly increases decoding complexity and jitter</td>
</tr>
</tbody>
</table>
Streaming Stored Media (4)

Parity packet can repair one lost packet in a group of N packets:
- Decoding is delayed for N packets
Streaming Stored Media (5)

Interleaving spreads nearby media samples over different transmissions to reduce the impact of loss.

Loss reduces temporal resolution; doesn’t leave a gap.
Streaming Stored Media (6)

Key problem: media may not arrive in time for playout due to variable bandwidth and loss/retransmissions

- Client buffers media to absorb jitter; we still need to pick an achievable media rate

![Diagram of client and server machines with buffers and watermarks]

Safety margin, to avoid a stall

Can pause server (or go ahead and save to disk)

CN5E by Tanenbaum & Wetherall, © Pearson Education-Prentice Hall and D. Wetherall, 2011
Streaming Stored Media (7)

RTSP commands
- Sent from player to server to adjust streaming

<table>
<thead>
<tr>
<th>Command</th>
<th>Server action</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIBE</td>
<td>List media parameters</td>
</tr>
<tr>
<td>SETUP</td>
<td>Establish a logical channel between the player and the server</td>
</tr>
<tr>
<td>PLAY</td>
<td>Start sending data to the client</td>
</tr>
<tr>
<td>RECORD</td>
<td>Start accepting data from the client</td>
</tr>
<tr>
<td>PAUSE</td>
<td>Temporarily stop sending data</td>
</tr>
<tr>
<td>TEARDOWN</td>
<td>Release the logical channel</td>
</tr>
</tbody>
</table>
Streaming Live Media (1)

Streaming live media is similar to the stored case plus:

- Can’t stream faster than “live rate” to get ahead
  - Usually need larger buffer to absorb jitter
- Often have many users viewing at the same time
  - UDP with multicast greatly improves efficiency. It is rarely available, so many TCP connections are used.
  - For very many users, content distribution is used [later]
Streaming Live Media (2)

With multicast streaming media, parity is effective

- Clients can each lose a different packet and recover
Streaming Live Media (2)

Production side of a student radio station.

As before
Real-Time Conferencing (1)

Real-time conferencing has two or more connected live media streams, e.g., voice over IP, Skype video call.

Key issue over live streaming is low (interactive) latency:
- Want to reduce delay to near propagation.
- Benefits from network support, e.g., QoS.
- Or, benefits from ample bandwidth (no congestion).
Real-Time Conferencing (2)

H.323 architecture for Internet telephony supports calls between Internet computers and PSTN phones.
Real-Time Conferencing (3)

H.323 protocol stack:
- Call is digital audio/video over RTP/UDP/IP
- Call setup is handled by other protocols (Q.931 etc.)

<table>
<thead>
<tr>
<th>Audio</th>
<th>Video</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.7xx</td>
<td>H.26x</td>
<td>H.225 (RAS)</td>
</tr>
<tr>
<td>RTP</td>
<td>RTCP</td>
<td>Q.931 (Signaling)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H.245 (Call Control)</td>
</tr>
<tr>
<td>UDP</td>
<td>TCP</td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td>Link layer protocol</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Physical layer protocol</td>
<td></td>
</tr>
</tbody>
</table>
Real-Time Conferencing (4)

Logical channels that make up an H.323 call

- Call signaling channel (Q.931)
- Call control channel (H.245)
- Forward data channel (RTP)
- Reverse data channel (RTP)
- Data control channel (RTCP)
Real-Time Conferencing (5)

SIP (Session Initiation Protocol) is an alternative to H.323 to set up real-time calls

- Simple, text-based protocol with URLs for addresses
- Data is carried with RTP / RTCP as before

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE</td>
<td>Request initiation of a session</td>
</tr>
<tr>
<td>ACK</td>
<td>Confirm that a session has been initiated</td>
</tr>
<tr>
<td>BYE</td>
<td>Request termination of a session</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>Query a host about its capabilities</td>
</tr>
<tr>
<td>CANCEL</td>
<td>Cancel a pending request</td>
</tr>
<tr>
<td>REGISTER</td>
<td>Inform a redirection server about the user’s current location</td>
</tr>
</tbody>
</table>
Real-Time Conferencing (6)

Setting up a call with the SIP three-way handshake
- Proxy server lets a remote callee be connected
- Call data flows directly between caller/callee
### Real-Time Conferencing (7)

<table>
<thead>
<tr>
<th>Item</th>
<th>H.323</th>
<th>SIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designed by</td>
<td>ITU</td>
<td>IETF</td>
</tr>
<tr>
<td>Compatibility with PSTN</td>
<td>Yes</td>
<td>Largely</td>
</tr>
<tr>
<td>Compatibility with Internet</td>
<td>Yes, over time</td>
<td>Yes</td>
</tr>
<tr>
<td>Architecture</td>
<td>Monolithic</td>
<td>Modular</td>
</tr>
<tr>
<td>Completeness</td>
<td>Full protocol stack</td>
<td>SIP just handles setup</td>
</tr>
<tr>
<td>Parameter negotiation</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Call signaling</td>
<td>Q.931 over TCP</td>
<td>SIP over TCP or UDP</td>
</tr>
<tr>
<td>Message format</td>
<td>Binary</td>
<td>ASCII</td>
</tr>
<tr>
<td>Media transport</td>
<td>RTP/RTCP</td>
<td>RTP/RTCP</td>
</tr>
<tr>
<td>Multiparty calls</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Multimedia conferences</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Addressing</td>
<td>URL or phone number</td>
<td>URL</td>
</tr>
<tr>
<td>Call termination</td>
<td>Explicit or TCP release</td>
<td>Explicit or timeout</td>
</tr>
<tr>
<td>Instant messaging</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Encryption</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Size of standards</td>
<td>1400 pages</td>
<td>250 pages</td>
</tr>
<tr>
<td>Implementation</td>
<td>Large and complex</td>
<td>Moderate, but issues</td>
</tr>
<tr>
<td>Status</td>
<td>Widcopen, cop. video</td>
<td>Alternative, cop. video</td>
</tr>
</tbody>
</table>

Comparison of H.323 and SIP.
Content Delivery

Delivery of content, especially Web and video, to users is a major component of Internet traffic

- Content and Internet traffic
- Server farms and Web proxies
- Content delivery networks
- Peer-to-peer networks
Content and Internet Traffic

Internet traffic:

1. Shifts seismically (email $\rightarrow$ FTP $\rightarrow$ Web $\rightarrow$ P2P $\rightarrow$ video)

2. Has many small/unpopular and few large/popular flows – mice and elephants

Zipf popularity distribution, $1/k$

Shows up as a line on log-log plot
Server Farms and Web Proxies (1)

Server farms enable large-scale Web servers:
- Front-end load-balances requests over servers
- Servers access the same backend database
Server Farms and Web Proxies (2)

Proxy caches help organizations to scale the Web
- Caches server content over clients for performance
- Also implements organization policies (e.g., access)
CDNs – Content Delivery Networks (1)

CDNs scale Web servers by having clients get content from a nearby CDN node (cache)
Content Delivery Networks (2)

Directing clients to nearby CDN nodes with DNS:

- Client query returns local CDN node as response
- Local CDN node caches content for nearby clients and reduces load on the origin server
Content Delivery Networks (3)

Origin server rewrites pages to serve content via CDN

Traditional Web page on server

```html
<html>
<head> <title> Fluffy Video </title> </head>
<body>
<h1> Fluffy Video's Product List </h1>
<p> Click below for free samples. </p>
<a href="koalas.mpg"> Koalas Today </a>
<a href="kangaroos.mpg"> Funny Kangaroos </a>
<a href="wombats.mpg"> Nice Wombats </a>
</body>
</html>
```

Page that distributes content via CDN

```html
<html>
<head> <title> Fluffy Video </title> </head>
<body>
<h1> Fluffy Video's Product List </h1>
<p> Click below for free samples. </p>
<a href="http://www.cdn.com/fluffyvideo/koalas.mpg"> Koalas Today </a>
<a href="http://www.cdn.com/fluffyvideo/kangaroos.mpg"> Funny Kangaroos </a>
<a href="http://www.cdn.com/fluffyvideo/wombats.mpg"> Nice Wombats </a>
</body>
</html>
```
Peer-to-Peer Networks (1)

P2P (Peer-to-Peer) is an alternative CDN architecture with no dedicated infrastructure (i.e., servers)

- Clients serve content to each other as peers

Challenges when servers are removed:
1. How do peers find each other?
2. How do peers support rapid content downloads?
3. How do peers encourage each other to upload?
Peer-to-Peer Networks (2)

BitTorrent lets peers download torrents

- Peers find each other via Tracker in torrent file
- Peers swap chunks (parts of content) with partners, preferring those who send most quickly [2]
- Many peers speed download; preference helps uploads [3]
Distributed Hash Tables (DHTs) are a fully distributed index that scales to very many clients/entries

- Need to follow $O(\log N)$ path for $N$ entries
- Can use as Tracker to find peers with no servers [1]
- Look up torrent (identifier) in DHT to find IP of peers
- Kademlia is used in BitTorrent
Peer-to-Peer Networks (3)

A Chord ring of 32 identifiers. Finger tables [at right, and as arcs] are used to navigate the ring.

- Example: path to look up 16 from 1 is 1 → 12 → 15

Identifier values are stored at predecessor

CN5E by Tanenbaum & Wetherall, © Pearson Education-Prentice Hall and D. Wetherall, 2011
End

Chapter 7