CS4220
Computer Networks

Lecture 7 The Transport Layer

Dr. Xiaobo “Charles” Zhou
Department of Computer Science
Application Layer
Chapter 7

- DNS – Domain Name System
- Electronic Mail
- The Web
- Streaming Audio and Video
- Content Delivery
The Application Layer

- Uses transport services to build distributed applications

<table>
<thead>
<tr>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport</td>
</tr>
<tr>
<td>Network</td>
</tr>
<tr>
<td>Link</td>
</tr>
<tr>
<td>Physical</td>
</tr>
</tbody>
</table>
1. DNS

- URL contains Internet name of machine ([www.nytimes.com](http://www.nytimes.com)), but not Internet address
- Internet needs Internet address to send information to a machine
- Browser software uses Domain Name System (DNS) protocol to send query for Internet address
- DNS system responds with Internet address

Q. www.nytimes.com?

A. 64.15.247.200
Browser software uses HyperText Transfer Protocol (HTTP) to send request for document.

HTTP server waits for requests by listening to a well-known port number (80 for HTTP).

HTTP client sends request messages through an “ephemeral port number,” e.g. 1127.

HTTP needs a Transmission Control Protocol (TCP) connection between the HTTP client and the HTTP server to transfer messages reliably.
3. HTTP

- HTTP client sends its request message: “GET …”
- HTTP server sends a status response: “200 OK”
- HTTP server sends requested file
- Browser displays document
- Clicking a link sets off a chain of events across the Internet!
- Let’s see how protocols & layers come into play…
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
The DNS Name Space (2)

- Generic top-level domains are controlled by ICANN who appoints registrars to run them

<table>
<thead>
<tr>
<th>Domain</th>
<th>Intended use</th>
<th>Start date</th>
<th>Restricted?</th>
</tr>
</thead>
<tbody>
<tr>
<td>com</td>
<td>Commercial</td>
<td>1985</td>
<td>No</td>
</tr>
<tr>
<td>edu</td>
<td>Educational institutions</td>
<td>1985</td>
<td>Yes</td>
</tr>
<tr>
<td>gov</td>
<td>Government</td>
<td>1985</td>
<td>Yes</td>
</tr>
<tr>
<td>int</td>
<td>International organizations</td>
<td>1988</td>
<td>Yes</td>
</tr>
<tr>
<td>mil</td>
<td>Military</td>
<td>1985</td>
<td>Yes</td>
</tr>
<tr>
<td>net</td>
<td>Network providers</td>
<td>1985</td>
<td>No</td>
</tr>
<tr>
<td>org</td>
<td>Non-profit organizations</td>
<td>1985</td>
<td>No</td>
</tr>
<tr>
<td>aero</td>
<td>Air transport</td>
<td>2001</td>
<td>Yes</td>
</tr>
<tr>
<td>biz</td>
<td>Businesses</td>
<td>2001</td>
<td>No</td>
</tr>
<tr>
<td>coop</td>
<td>Cooperatives</td>
<td>2001</td>
<td>Yes</td>
</tr>
<tr>
<td>info</td>
<td>Informational</td>
<td>2002</td>
<td>No</td>
</tr>
<tr>
<td>museum</td>
<td>Museums</td>
<td>2002</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>People</td>
<td>2002</td>
<td>No</td>
</tr>
<tr>
<td>pro</td>
<td>Professionals</td>
<td>2002</td>
<td>Yes</td>
</tr>
<tr>
<td>cat</td>
<td>Catalan</td>
<td>2005</td>
<td>Yes</td>
</tr>
<tr>
<td>jobs</td>
<td>Employment</td>
<td>2005</td>
<td>Yes</td>
</tr>
<tr>
<td>mobi</td>
<td>Mobile devices</td>
<td>2005</td>
<td>Yes</td>
</tr>
<tr>
<td>tel</td>
<td>Contact details</td>
<td>2005</td>
<td>Yes</td>
</tr>
<tr>
<td>travel</td>
<td>Travel industry</td>
<td>2005</td>
<td>Yes</td>
</tr>
<tr>
<td>xxx</td>
<td>Sex industry</td>
<td>2010</td>
<td>No</td>
</tr>
</tbody>
</table>

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)

; Authoritative data for cs.vu.nl

<table>
<thead>
<tr>
<th>Domain</th>
<th>Type</th>
<th>Class</th>
<th>TTL</th>
<th>Record Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs.vu.nl</td>
<td>SOA</td>
<td>A</td>
<td>86400</td>
<td>star boss (9527,7200,7200,241920,86400)</td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>MX</td>
<td>A</td>
<td>86400</td>
<td>1 zephyr</td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>MX</td>
<td>A</td>
<td>86400</td>
<td>2 top</td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>NS</td>
<td>A</td>
<td>86400</td>
<td>star</td>
</tr>
<tr>
<td>star</td>
<td>A</td>
<td>A</td>
<td>86400</td>
<td>130.37.56.205</td>
</tr>
<tr>
<td>zephyr</td>
<td>A</td>
<td>A</td>
<td>86400</td>
<td>130.37.20.10</td>
</tr>
<tr>
<td>top</td>
<td>A</td>
<td>A</td>
<td>86400</td>
<td>130.37.20.11</td>
</tr>
<tr>
<td>www</td>
<td>CNAME</td>
<td>A</td>
<td>86400</td>
<td>star.cs.vu.nl</td>
</tr>
<tr>
<td>ftp</td>
<td>CNAME</td>
<td>A</td>
<td>86400</td>
<td>zephyr.cs.vu.nl</td>
</tr>
<tr>
<td>flits</td>
<td>A</td>
<td>A</td>
<td>86400</td>
<td>130.37.16.112</td>
</tr>
<tr>
<td>flits</td>
<td>MX</td>
<td>A</td>
<td>86400</td>
<td>1 flits</td>
</tr>
<tr>
<td>flits</td>
<td>MX</td>
<td>A</td>
<td>86400</td>
<td>2 zephyr</td>
</tr>
<tr>
<td>flits</td>
<td>MX</td>
<td>A</td>
<td>86400</td>
<td>3 top</td>
</tr>
<tr>
<td>rowboat</td>
<td>A</td>
<td>MX</td>
<td>1</td>
<td>1 rowboat</td>
</tr>
<tr>
<td></td>
<td>MX</td>
<td></td>
<td>2</td>
<td>2 zephyr</td>
</tr>
<tr>
<td>little-sister</td>
<td>A</td>
<td>A</td>
<td></td>
<td>130.37.62.23</td>
</tr>
<tr>
<td>laserjet</td>
<td>A</td>
<td>A</td>
<td></td>
<td>192.31.231.216</td>
</tr>
</tbody>
</table>

- **Name server**
- **IP addresses of computers**
- **Mail gateways**

° A portion of a possible DNS database for cs.vu.nl.
Name Servers (1)

- Name servers contain data for portions of the name space called zones (circled).

![Diagram of the domain name system](image-url)

One zone
Name Servers (2)

- 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
Example of a computer looking up the IP for a name
The World Wide Web

• Architectural overview »
• Static Web pages »
• Dynamic pages and Web applications »
• HTTP – HyperText Transfer Protocol »
• The mobile Web »
• Web search »
HTTP transfers pages from servers to browsers
Pages are named with URLs (Uniform Resource Locators)


<table>
<thead>
<tr>
<th>Protocol</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>

*Common URL protocols*
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
Architectural Overview (5)

- To scale performance, Web servers can use:
  - Caching, multiple threads, and a front end
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) . . .
# Static Web Pages (2)

## 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></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Lists</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Active maps &amp; images</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Forms</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equations</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Toolbars</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Tables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Accessibility features</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Object embedding</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Style sheets</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</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>x</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
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
<h1>Reply: </h1>
Hello <php echo $name; ?>.
Prediction: next year you will be <php echo $age + 1; ?></php>
</body>
</html>
```

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)

JavaScript program produces result page in the browser

```html
<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.
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>
```
Dynamic Pages & Web Applications (8)

Web applications use a set of technologies, revisited:
HTTP (1)

- 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>

- Fetch a page
- Used to send input data to a server program
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>
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 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)
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
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
Internet traffic:

1. Shifts seismically (email → FTP → Web → P2P → 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
  - Cloud Computing
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 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
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]
Peer-to-Peer Networks (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