CS 471/571 Spring 2016

http 2
Internet Transport Layer

• TCP
  – Connection oriented
  – Reliable data transfer

• UDP
  – Connectionless
  – Data loss possible
  – It is possible that packets are delivered out of order
Internet Transport Layer

• No throughput guarantees
• No delay guarantees
• No security built into TCP or UDP
• SSL
  – Secure socket layer
  – Application layer services built on top of TCP
HyperText Transfer Protocol (http)

- Overview
- Connections
- Message Format
- Cookies
- Web Caching
Overview

• Created in the early 1990s by Tim Berners-Lee
• Client/Server Architecture
  – http client/http server
  – Request/response
• Web Page
  – Web page consist of one or more objects
• URL: hostname/pathname
  – cs.uwlax.edu/~gendreau/cs471
• Stateless
Hypertext

• Vannevar Bush: Memex (1954)
• Ted Nelson (1963)
• Douglas Engelbart (1962)
• Hypercard (MacIntosh 1987)
Request/Response

PC running Firefox browser

server running Apache Web server

iphone running Safari browser

HTTP request

HTTP response
Connections

- TCP connections
- Persistent
- Non-Persistent
Message Format

- Syntax
- Semantics
- Standard ASCII Text
- Methods
  - GET
  - POST
  - HEAD
  - PUT
  - DELETE
Example Request

GET /index.html HTTP/1.1\r\nHost: www-net.cs.umass.edu\r\nUser-Agent: Firefox/3.6.10\r\nAccept: text/html,application/xhtml+xml\r\nAccept-Language: en-us,en;q=0.5\r\nAccept-Encoding: gzip,deflate\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7\r\nKeep-Alive: 115\r\nConnection: keep-alive\r\n\r\n
HTTP request message: general format

- **method**
- **URL**
- **version**
- **header field name**
- **value**
- **entity body**

**request line**

**header lines**

**body**
HTTP response message

status line (protocol status code status phrase)

```
HTTP/1.1 200 OK
Date: Sun, 26 Sep 2010 20:09:20 GMT
Server: Apache/2.0.52 (CentOS)
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT
ETag: "17dc6-a5c-bf716880"
Accept-Ranges: bytes
Content-Length: 2652
Keep-Alive: timeout=10, max=100
Connection: Keep-Alive
Content-Type: text/html;
    charset=ISO-8859-1

...data data data data data data data...
```
HTTP response message: general format

version  sp  Status code  sp  phrase  cr  lf
header field name  sp  value  cr  lf
header field name  sp  value  cr  lf

status line
header lines
body

entity body
Message Format

• Status codes
  – 200 OK:
  – 301 Moved Permanently
  – 400 Bad Request
  – 404 Not Found:
  – 505 HTTP Version Not Supported:
Cookies

• Client maintains cookie file
• Server maintains database matching id number with information about the user
• Once the id is established it can be included in response messages
Cookies: keeping “state” (cont.)

Client

- eBay 8734
- Cookie file
- eBay 8734 Amazon 1678

Usual HTTP request msg

Usual HTTP response msg

**set-cookie:** 1678

Server

Amazon server creates ID 1678 for user

Create entry

Backend database

One week later:

- eBay 8734
- Amazon 1678

Usual HTTP request msg

**cookie:** 1678

Usual HTTP response msg

Cookie-specific action

Access

Access

Application Layer
Web Cache

- Proxy Server
- Conditional Get
Web Cache
File Transfer Protocol (ftp)

- Allows a user to sign in (an anonymous account can be used) and transfer files between the client machine and server machine
- ftp server listens on port 21
- Uses TCP
  - Separate control and data connection
  - Out-of-band control
  - http uses in-band control
- Maintains user state
ftp

user at host

FTP user interface

FTP client

File transfer

FTP server

local file system

remote file system
ftp Connections

TCP control connection, server port 21

TCP data connection, server port 20
ftp Commands

- Data transferred in plain text
- User username
- PASS password
- List
- RETR filename (get)
- STOR filename (put)
DNS: domain name system

**people:** many identifiers:
- SSN, name, passport #

**Internet hosts, routers:**
- IP address (32 bit) - used for addressing datagrams
- “name”, e.g., www.yahoo.com - used by humans

**Q:** how to map between IP address and name, and vice versa?

**Domain Name System:**
- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol:* hosts, name servers communicate to *resolve* names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network’s “edge”
DNS: services, structure

DNS services

- hostname to IP address translation
- host aliasing
  - canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: many IP addresses correspond to one name

why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

A: doesn’t scale!
DNS: a distributed, hierarchical database

Client wants IP for www.amazon.com; 1st approx:

- Client queries root server to find com DNS server
- Client queries .com DNS server to get amazon.com DNS server
- Client queries amazon.com DNS server to get IP address for www.amazon.com
DNS: root name servers

- contacted by local name server that can not resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server

13 root name “servers” worldwide
**TLD, authoritative servers**

*top-level domain (TLD) servers:*
- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, ca, jp
- Network Solutions maintains servers for .com TLD
- Educause for .edu TLD

*authoritative DNS servers:*
- organization’s own DNS server(s), providing authoritative hostname to IP mappings for organization’s named hosts
- can be maintained by organization or service provider
Local DNS name server

- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
  - also called “default name server”
- when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy
DNS name resolution example

• host at cis.poly.edu wants IP address for gaia.cs.umass.edu

iterated query:
  ▶ contacted server replies with name of server to contact
  ▶ “I don’t know this name, but ask this server”
DNS name resolution example

**recursive query:**
- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?
DNS: caching, updating records

• once (any) name server learns mapping, it *caches* mapping
  – cache entries timeout (disappear) after some time (TTL)
  – TLD servers typically cached in local name servers
    • thus root name servers not often visited
• cached entries may be *out-of-date* (best effort name-to-address translation!)
  – if name host changes IP address, may not be known Internet-wide until all TTLs expire
• update/notify mechanisms proposed IETF standard
  – RFC 2136
DNS records

**DNS**: distributed db storing resource records (RR)

RR format: \((\text{name}, \text{value, type, ttl})\)

- **type=A**
  - name is hostname
  - value is IP address

- **type=NS**
  - name is domain (e.g., foo.com)
  - value is hostname of authoritative name server for this domain

- **type=CNAME**
  - name is alias name for some “canonical” (the real) name
  - www.ibm.com is really servereast.backup2.ibm.com
  - value is canonical name

- **type=MX**
  - value is name of mailserver associated with name
**DNS protocol, messages**

- *query* and *reply* messages, both with same message format

**msg header**

- **identification**: 16 bit # for query, reply to query uses same #
- **flags**:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

<table>
<thead>
<tr>
<th>Identification</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td># questions</td>
<td># answer RRs</td>
</tr>
<tr>
<td># authority RRs</td>
<td># additional RRs</td>
</tr>
</tbody>
</table>

- questions (variable # of questions)
- answers (variable # of RRs)
- authority (variable # of RRs)
- additional info (variable # of RRs)
### DNS protocol, messages

<table>
<thead>
<tr>
<th>Identification</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td># questions</td>
<td># answer RRs</td>
</tr>
<tr>
<td># authority RRs</td>
<td># additional RRs</td>
</tr>
<tr>
<td>questions (variable # of questions)</td>
<td>answers (variable # of RRs)</td>
</tr>
<tr>
<td></td>
<td>authority (variable # of RRs)</td>
</tr>
<tr>
<td></td>
<td>additional info (variable # of RRs)</td>
</tr>
</tbody>
</table>

- **name, type fields** for a query
- **RRs in response to query**
- **records for authoritative servers**
- **additional “helpful” info that may be used**
Inserting records into DNS

- example: new startup “Network Utopia”
- register name networkuptopia.com at DNS registrar (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into .com TLD server:
    (networkutopia.com, dns1.networkutopia.com, NS)
    (dns1.networkutopia.com, 212.212.212.1, A)
- create authoritative server type A record for www.networkuptopia.com; type MX record for networkutopia.com
Attacking DNS

DDoS attacks
• Bombard root servers with traffic
  – Not successful to date
  – Traffic Filtering
  – Local DNS servers cache IPs of TLD servers, allowing root server bypass
• Bombard TLD servers
  – Potentially more dangerous

Redirect attacks
❖ Man-in-middle
  ▪ Intercept queries
❖ DNS poisoning
  ▪ Send bogus replies to DNS server, which caches

Exploit DNS for DDoS
❖ Send queries with spoofed source address: target IP
❖ Requires amplification