Chapter 2
Application Layer

A note on the use of these Powerpoint slides:
We’re making these slides freely available to all (faculty, students, readers). They’re in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we’d like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

© All material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved
Chapter 2: outline

2.1 principles of network applications
2.2 Web and HTTP
2.3 electronic mail
   • SMTP, POP3, IMAP
2.4 DNS
2.5 P2P applications
2.6 video streaming and content distribution networks
2.7 socket programming with UDP and TCP
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; 1\textsuperscript{st} approximation:

- 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 logical root name “servers” worldwide
- each “server” replicated many times
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 database storing resource records (RR)

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

**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*

**message 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>
<tr>
<td>questions (variable # of questions)</td>
<td></td>
</tr>
<tr>
<td>answers (variable # of RRs)</td>
<td></td>
</tr>
<tr>
<td>authority (variable # of RRs)</td>
<td></td>
</tr>
<tr>
<td>additional info (variable # of RRs)</td>
<td></td>
</tr>
</tbody>
</table>
# DNS protocol, messages

<table>
<thead>
<tr>
<th></th>
<th>identification</th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td># questions</td>
<td></td>
<td># answer RRs</td>
</tr>
<tr>
<td># authority RRs</td>
<td></td>
<td># additional RRs</td>
</tr>
<tr>
<td>questions</td>
<td></td>
<td>answers</td>
</tr>
<tr>
<td>(variable #</td>
<td></td>
<td>(variable # of RRs)</td>
</tr>
<tr>
<td>of questions)</td>
<td></td>
<td>authority</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(variable # of RRs)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>additional info</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(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 relies to DNS server, which caches

exploit DNS for DDoS

- send queries with spoofed source address: target IP
- requires amplification
Wireshark trace - DNS
Chapter 2: outline

2.1 principles of network applications
2.2 Web and HTTP
2.3 electronic mail
   • SMTP, POP3, IMAP
2.4 DNS

2.5 P2P applications
2.6 video streaming and content distribution networks
2.7 socket programming with UDP and TCP
Pure P2P architecture

- no always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

examples:
- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)
**Question**: how much time to distribute file (size $F$) from one server to $N$ peers?

- peer upload/download capacity is limited resource
File distribution time: client-server

- **server transmission**: must sequentially send (upload) N file copies:
  - time to send one copy: \( F/u_s \)
  - time to send \( N \) copies: \( NF/u_s \)

- **client**: each client must download file copy
  - \( d_{min} = \) min client download rate
  - min client download time: \( F/d_{min} \)

\[
\text{time to distribute } F \\
to N clients using \\
\text{client-server approach}
\]

\[
D_{c-s} \geq \max\{NF/u_s, F/d_{min}\}
\]

Increases linearly in \( N \)
File distribution time: P2P

- **server transmission**: must upload at least one copy
  - time to send one copy: $F/u_s$
- **client**: each client must download file copy
  - min client download time: $F/d_{min}$
- **clients**: as aggregate must download $NF$ bits
  - max upload rate (limiting max download rate) is $u_s + \sum u_i$

Time to distribute $F$ to $N$ clients using P2P approach:

$$D_{P2P} \geq \max\{F/u_s, F/d_{min}, NF/(u_s + \sum u_i)\}$$

increases linearly in $N$ …

… but so does this, as each peer brings service capacity
Client-server vs. P2P: example

client upload rate = \( u \), \( F/u = 1 \) hour, \( u_s = 10u \), \( d_{\text{min}} \geq u_s \)
P2P file distribution: BitTorrent

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks

**tracker**: tracks peers participating in torrent

**torrent**: group of peers exchanging chunks of a file

Alice arrives ...
... obtains list of peers from tracker
... and begins exchanging file chunks with peers in torrent
P2P file distribution: BitTorrent

- peer joining torrent:
  - has no chunks, but will accumulate them over time from other peers
  - registers with tracker to get list of peers, connects to subset of peers ("neighbors")

- while downloading, peer uploads chunks to other peers
- peer may change peers with whom it exchanges chunks
- **churn**: peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent
BitTorrent: requesting, sending file chunks

requesting chunks:
- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first

sending chunks: tit-for-tat
- Alice sends chunks to those four peers currently sending her chunks at highest rate
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - “optimistically unchoke” this peer
  - newly chosen peer may join top 4
BitTorrent: tit-for-tat

(1) Alice “optimistically unchokes” Bob
(2) Alice becomes one of Bob’s top-four providers; Bob reciprocates
(3) Bob becomes one of Alice’s top-four providers

higher upload rate: find better trading partners, get file faster!