Chapter 2 outline

- 2.1 Principles of app layer protocols
  - clients and servers
  - app requirements
- 2.2 Web and HTTP
- 2.3 Telnet, rlogin, smtp
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Socket programming with TCP
- 2.7 Socket programming with UDP
- 2.8 Building a Web server
- 2.9 Content distribution
  - Network Web caching
  - Content distribution networks
  - P2P File sharing

DNS: Domain Name System

People: many identifiers:
  - SSN, name, passport #
Internet hosts, routers:
  - IP address (32 bit) - used for addressing datagrams
  - "name", e.g., aegis.cs.wmich.edu - used by humans
Q: map between IP addresses and name?

Domain Name System:
  - distributed database implemented in hierarchy of many name servers
  - application-layer protocol
    - host, routers, name servers to communicate to resolve names (address/name translation)
    - note: core Internet function, implemented as application-layer protocol
    - complexity at network's "edge"

DNS name servers

Why not centralize DNS?
  - single point of failure
  - traffic volume
  - distant centralized database
  - maintenance
  - doesn't scale!

no server has all name-to-IP address mappings

local name servers:
  - each ISP, company has
    - local (default) name server
  - host DNS query first goes to local name server

authoritative name server:
  - for a host: stores that host's IP address, name
  - can perform name/address translation for that host's name
**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

**Simple DNS example**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Host surfeurecom.fr wants IP address of gaia.cs.umass.edu</td>
</tr>
<tr>
<td>2</td>
<td>Contacts its local DNS server, dns.eurecom.fr</td>
</tr>
<tr>
<td>3</td>
<td>Dns.eurecom.fr contacts root name server, if necessary</td>
</tr>
<tr>
<td>4</td>
<td>Root name server contacts authoritative name server, dns.cs.umass.edu, if necessary</td>
</tr>
<tr>
<td>5</td>
<td>Authoritative name server, dns.cs.umass.edu, returns mapping to local name server</td>
</tr>
<tr>
<td>6</td>
<td>Local name server, dns.eurecom.fr, returns mapping to host surfeurecom.fr</td>
</tr>
</tbody>
</table>

**DNS example**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Requesting host surfeurecom.fr contacts root name server</td>
</tr>
<tr>
<td>2</td>
<td>Root name server, if necessary</td>
</tr>
<tr>
<td>3</td>
<td>Contacts authoritative name server, dns.cs.umass.edu</td>
</tr>
<tr>
<td>4</td>
<td>Authoritative name server, dns.cs.umass.edu, returns mapping to local name server</td>
</tr>
<tr>
<td>5</td>
<td>Local name server, dns.eurecom.fr, returns mapping to host surfeurecom.fr</td>
</tr>
</tbody>
</table>

13 root name servers worldwide.
DNS: iterated queries

**recursive query:**
- puts burden of name resolution on contacted name server
- heavy load?

**iterated query:**
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"

<table>
<thead>
<tr>
<th>Requesting Host</th>
<th>Intermediate Name Server</th>
<th>Authoritative Name Server</th>
<th>Local Name Server</th>
<th>Root Name Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>surf.eurecom.fr</td>
<td>dns.eurecom.fr</td>
<td>dns.cs.umass.edu</td>
<td>gaia.cs.umass.edu</td>
<td></td>
</tr>
</tbody>
</table>

DNS: caching and updating records

- once (any) name server learns mapping, it caches mapping
- cache entries timeout (disappear) after some time
- update/notify mechanisms under design by IETF
  - RFC 2136

DNS records

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

<table>
<thead>
<tr>
<th>RR format: (name, value, type, ttl)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type=A</td>
</tr>
<tr>
<td>name is hostname</td>
</tr>
<tr>
<td>value is IP address</td>
</tr>
<tr>
<td>Type:NS</td>
</tr>
<tr>
<td>name is domain (e.g. foo.com)</td>
</tr>
<tr>
<td>value is IP address of authoritative name server for this domain</td>
</tr>
<tr>
<td>Type:CNAME</td>
</tr>
<tr>
<td>name is alias name for some &quot;canonical&quot; (the real) name</td>
</tr>
<tr>
<td>value is IP address of authoritative name server for this domain</td>
</tr>
<tr>
<td>Type=MX</td>
</tr>
<tr>
<td>value is name of mailserver associated with name</td>
</tr>
</tbody>
</table>
**DNS protocol, messages**

**DNS protocol:** 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

**DNS protocol, messages**

**Name, type fields for a query**
- RR in response to query
- records for authoritative servers
- additional “helpful” info that may be used

**DNS (Domain name Service)**

- Domain Name Service (DNS) provides the translation between hostname and IP address.
- DNS uses the UDP protocol and port 53
Domain Name Space Tree

Generic domains define registered hosts according to their generic behavior:
- com: Commercial organization
- edu: Educational institutions
- gov: Government institutions
- int: International organization
- mil: Military groups
- net: Network support centers
- org: Nonprofit organizations

Country domains use two-character country abbreviations:
- us: for United States
- in: for India

The reverse domain is used to map an address to a name.

Chapter 2 outline

- 2.1 Principles of app layer protocols
  - clients and servers
  - app requirements
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Socket programming with TCP
- 2.7 Socket programming with UDP
- 2.8 Building a Web server
- 2.9 Content distribution
  - Network Web caching
  - Content distribution networks
  - P2P file sharing

Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returned object
  - else cache requests object from origin server, then returns object to client

Proxy server

Origin server

Client
More about Web caching

- Cache acts as both client and server.
- Cache can do up-to-date check using If-modified-since HTTP header.
- Issue: should cache take risk and deliver cached object without checking?
- Heuristics are used.
- Typically cache is installed by ISP (university, company, residential ISP).

Why Web caching?
- Reduce response time for client request.
- Reduce traffic on an institution’s access link.
- Internet dense with caches enables “poor” content providers to effectively deliver content.

Caching example (1)

Assumptions
- average object size = 100,000 bits
- avg. request rate from institution’s browser to origin servers = 15/sec
- delay from institutional router to any origin server and back to router = 2 sec.

Consequences
- utilization on LAN = 15%
- utilization on access link = 100%
- total delay = Internet delay + access delay + LAN delay
- delay = 2 sec + minutes + milliseconds
- Delay UNACCEPTABLE!

Caching example (2)

Possible solution
- increase bandwidth of access link to, say, 10 Mbps

Consequences
- utilization on LAN = 15%
- utilization on access link = 15%
- total delay = Internet delay + access delay + LAN delay
- delay = 2 sec + msecs + msecs
- often a costly upgrade.
Caching example (3)

- Install cache
  - Suppose hit rate is .4

Consequence:
- 40% requests will be satisfied almost immediately
- 60% requests satisfied by origin server
- Utilization of access link reduced to 60%, resulting in negligible delays (say 10 milliseconds)
- Total delay = Internet delay + access delay + LAN delay
- $\Delta t^2 sec + \Delta t^01 sec + \text{milliseconds} \times 1.3 sec$

Content distribution networks (CDNs)

- The content providers are the CDN customers.
- Content replication
  - CDN company installs hundreds of CDN servers throughout Internet
  - In lower-tier ISPs, close to users
  - CDN replicates its customers' content in CDN servers
    - When provider updates content, CDN updates servers

CDN example

- Origin server
- CDN company
  - cdn.com
- Distributes GIF files
  - Uses its authoritative DNS server to route requests

Content provider Replaces:
- http://www.foo.com/sports/ruth.gif
  - with
More about CDNs

Routing requests
- CDN creates a "map", indicating distances from leaf ISPs and CDN nodes
- When query arrives at authoritative DNS server:
  - Server determines ISP from which query originates
  - Uses "map" to determine best CDN server

Not just Web pages
- Streaming stored audio/video
- Streaming real-time audio/video
- CDN nodes create application-layer overlay network

P2P file sharing

Example
- Alice runs P2P client application on her notebook computer
- Intermittently connects to Internet; gets new IP address for each connection
- Asks for "Hey Jude"
- Application displays other peers that have copy of Hey Jude.

- Alice chooses one of the peers, Bob.
- File is copied from Bob's PC to Alice's notebook: HTTP
- While Alice downloads, other users uploading from Alice.
- Alice's peer is both a Web client and a transient Web server.

All peers are servers = highly scalable!

P2P: centralized directory

Original "Napster" design
1) When peer connects, it informs central server:
   - IP address
   - Content
2) Alice queries for "Hey Jude"
3) Alice requests file from Bob

Centralized directory server
P2P: problems with centralized directory
- Single point of failure
- Performance bottleneck
- Copyright infringement

file transfer is decentralized, but locating content is highly centralized

P2P: decentralized directory
- Each peer is either a group leader or assigned to a group leader.
- Group leader tracks the content in all its children.
- Peer queries group leader; group leader may query other group leaders.

More about decentralized directory
- overlay network
  - peers are nodes
  - edges between peers and their group leaders
  - edges between some pairs of group leaders
  - virtual neighbors
  - bootstrap node (provides initial connection point)
- advantages of approach
  - no centralized directory server
  - location service distributed over peers
  - more difficult to shut down
- disadvantages of approach
  - bootstrap node needed
  - group leaders can get overloaded
P2P: Query flooding

- Gnutella
  - (for file-sharing, no directory service at all)
- no hierarchy
- use bootstrap node to learn about others
- join message

Object Location

- Send query to neighbors
- Neighbors forward query
- If queried peer has object, it sends message back to querying peer

P2P: more on query flooding

Pros
- peers have similar responsibilities: no group leaders
- highly decentralized
- no peer maintains directory info

Cons
- excessive query traffic
- query radius: may not have content when present
- bootstrap node
- maintenance of overlay network

Chapter 2: Summary

Our study of network apps now complete!

- application service requirements:
  - reliability, bandwidth, delay
- client-server paradigm
- Internet transport service model
  - connection-oriented, reliable: TCP
  - unreliable, datagrams: UDP
- specific protocols:
  - HTTP
  - FTP
  - telnet, rlogin, snmp
  - SMTP, POP, IMAP
  - DNS
- socket programming
- content distribution
  - caches, CDN
  - P2P
Chapter 2: Summary

Most importantly: learned about protocols

- typical request/reply message exchange:
  - client requests info or service
  - server responds with data, status code

- message formats:
  - headers: fields giving info about data
  - data: info being communicated

- control vs. data msg
  - in-band, out-of-band
  - centralized vs. decentralized
  - stateless vs. stateful
  - reliable vs. unreliable msg transfer

- "complexity at network edge"

- security: authentication