Outline

- Introduce Application Layer
- Socket Programming
- Domain Name Service (DNS)
- Standard Application-level Protocols
  - email (SMTP)
  - HTTP
  - Video streaming

DNS: Domain Name System

- distributed database implemented in hierarchy of many name servers
- Names: sequence of subnames, local to global order, delimited by periods
- Application-layer protocol: hosts and name servers communicate to resolve names (address/name translation)
- Note: This is a core Internet function, but implemented as application-layer protocol!
- Why? Put complexity at network “edge.”
DNS functionality

DNS services
- hostname to IP address translation
- host aliasing
  - Canonical: relay1.west-www.enterprise.com
  - Alias: enterprise.com
- mail server aliasing
  - bob@enterprise.com
- load distribution
  - replicated Web servers: set of IP addresses for one canonical name

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

Distributed, Hierarchical Database

Client wants IP for www.amazon.com; 1st approx:
- client queries local name server, which contacts 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 cannot 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 and Authoritative Servers

- Top-level domain (TLD) servers:
  - responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
  - Network Solutions maintains servers for com TLD
  - Educause for edu TLD
- Authoritative DNS servers:
  - organization’s DNS servers, providing authoritative hostname to IP mappings for organization’s servers (e.g., Web, mail).
  - can be maintained by organization or service provider
Local 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
  - 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
- Iterated approach is more common

DNS: caching and updating records

- once (any) name server learns mapping, it caches mapping
  - cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers
    - Thus root name servers not often visited
- update/notify mechanisms under design by IETF
  - RFC 2136
DNS records

- **DNS**: actually more general than mapping names to IP addresses. Any name can have a set of resource records associated with it.

- **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

How it works

- **Name resolver**: client software, which may communicate with more than one name server in resolving a name. Resolver must know how to contact at least one name server. Every name server knows the address of at least one root server.

- **Application-level** protocol!
**Resolver implementation**

- In Unix/Linux, comprises several *(libc)* routines
  - `res_mkquery()` - compose a query
  - `res_send()` - send a query
  - `res_init()` - initialize data
- Linked with program via `-lresolv` argument
- Global information that is used by the resolver routines is kept in the variable `_res`, which is automatically included in your program with `<resolv.h>`.

**resolv.conf**

- Identities of name servers are read from `/etc/resolv.conf`
- On `arctic.cse.msu.edu`:
  ```
  domain cse.msu.edu
  search cse.msu.edu
  nameserver 35.9.20.31
  nameserver 35.9.20.10
  ```
Example: gethostbyname()

```c
struct hostent *
gethostbyname(name) char *name;
{
    int n;
    querybuf buf;
    register struct hostent *hp;
    extern struct hostent *
        _gethtbyname();
    n = res_mkquery(QUERY, name, C_IN, T_A, (char *)NULL,
                    0, NULL, (char *)&buf, sizeof(buf));
    if (n < 0) return (NULL);
    hp = getanswer((char *)&buf, n, 0);  //calls res_send()
    if (hp == NULL && errno == ECONNREFUSED)
        hp = _gethtbyname(name);
    return(hp);
}
```

Outline

Introduce Application Layer
Socket Programming
Domain Name Service (DNS)
Standard Application-level Protocols
  - email (SMTP)
  - HTTP
  - Video streaming
Electronic Mail

Three major components:
- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent
- a.k.a. “mail reader”
- composing, editing, reading mail messages
- e.g., Eudora, Outlook, elm, Mozilla Thunderbird
- outgoing, incoming messages stored on server

Electronic Mail: mail servers

Mail Servers
- mailbox contains incoming messages for user
- message queue of outgoing (to be sent) mail messages
- SMTP protocol between mail servers to send email messages
  - client: sending mail server
  - “server”: receiving mail server
**Electronic Mail: SMTP [RFC 2821]**

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII

---

**Scenario: Alice sends message to Bob**

1) Alice uses UA to compose message and “to” bob@someschool.edu
2) Alice’s UA sends message to her mail server; message placed in message queue
3) Client side of SMTP opens TCP connection with Bob’s mail server
4) SMTP client sends Alice’s message over the TCP connection
5) Bob’s mail server places the message in Bob’s mailbox
6) Bob invokes his user agent to read message
Sample SMTP interaction after establishing connection

S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection

Spooling

- Outgoing mail placed in local directory along with details of recipient
- Mail transfer begun as a background activity
- Once address is resolved, TCP connection is set up to destination, message transferred, old copy deleted.
- If not delivered, copy remains in storage.
- Periodically, background process checks local storage for undelivered messages and attempts to deliver them.
- Gives up after certain period (e.g. 3 days).
Mail access protocols

- SMTP: delivery/storage to receiver’s server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
    - authorization (agent <-> server) and download
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - more features (more complex)
    - manipulation of stored msgs on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.