Section 1: Review

1.1 what is the Internet?
1.2 network edge
   • end systems, access networks, links
1.3 network core
   • packet switching, circuit switching, network structure
1.4 delay, loss, throughput in networks
1.5 protocol layers, service models
1.6 networks under attack: security
1.7 history
Chapter 2
Application Layer
App-layer protocol defines

- Types of messages exchanged,
  - e.g., request, response

- Message format:
  - Syntax: what fields in messages
  - Semantics: meaning of information in fields

- Rules for when and how processes send & respond to messages

Public-domain protocols:
- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

Proprietary protocols:
- e.g., Skype, Hangout
Consider: Two applications on 2 different hosts connected by a network.

In order to communicate, need to identify the parties.

Phone network: phone number (10 digits)

Computer network: IP address

- IPv4 (32 bits) 128.6.24.78
- IPv6 (128 bits)
  2001:4000:A000:C000:6000:B001:412A:8000

In addition to host address, we need one more.

More than one program executing on a host.

Which Program to talk to?

We need another identity: port #
IP address & port number

Socket is the door between network and the application/process
A network connection is a 4-tuple

\[ \text{IP}_S, \text{Port}_S, \text{IP}_D, \text{Port}_D \]
Recall: Services provided by lower layers
Client-server architecture

server:
- always-on host
- permanent IP address
- server farms for scaling

clients:
- communicate with server
- may not be always connected
- may have dynamic IP addresses
- do not communicate directly with each other
DNS

Why?
For any networked application, we need to know the IP address of a host given its name
Domain Name System (DNS)

Problem statement:

- Average brain can easily remember **7 digits** for a few names
- On average, IP addresses have **12 digits**
- We need an easier way to remember IP addresses

Solution:

- Use **names** to refer to hosts
- Just as a contacts or telephone **book** (white pages)
- Add a **service** (called DNS) to map between host names and binary IP addresses
- We call this **Address Resolution**
Simple DNS

- Simple but does not scale
- Every new host needs to be entered in this table
- Performance? Failure?

<table>
<thead>
<tr>
<th>DOMAIN NAME</th>
<th>IP ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://WWW.YAHOO.COM">WWW.YAHOO.COM</a></td>
<td>98.138.253.109</td>
</tr>
<tr>
<td>cs.rutgers.edu</td>
<td>128.6.4.2</td>
</tr>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
<td>74.125.225.243</td>
</tr>
<tr>
<td><a href="http://www.princeton.edu">www.princeton.edu</a></td>
<td>128.112.132.86</td>
</tr>
</tbody>
</table>
DNS

Centralize DNS?
- single point of failure
- traffic volume
- Distant centralized database
- maintenance

doesn't scale!
Distributed, Hierarchical Database

RFC 1034
DNS Protocol

- Client and Server (CS Model)
- Client connects to Port 53
- DNS server address should be known
  - Either manually configured or automatically

- Two types of messages
  - Queries
  - Responses

- Type of Query methods
  - Standard query
    - Request IP address
  - Updates
    - Provide a binding of IP address to domain name

- Each type has a common message format that follows the header
DNS Protocol

- When client wants to know an IP address for a host name
  - Client sends a DNS query to the primary name server in its zone
  - If name server contains the mapping, it returns the IP address to the client
  - Otherwise, the name server forwards the request to the root name server
  - The request works its way down the tree toward the host until it reaches a name server with the correct mapping
Example

- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu
**Query type**

**iterated query:**

- contacted server replies with name of server to contact

- “I don’t know this name, but ask this server”
Query type

recursive query:
- puts burden of name resolution on contacted name server

Diagram:
- Requesting host: cis.poly.edu
- Local DNS server: dns.poly.edu
- CS DNS server: dns.cs.umass.edu
- GAIA DNS server: gaia.cs.umass.edu
- Edu DNS server
- Root DNS server

Steps:
1. Requesting host to local DNS server
2. Local DNS server to root DNS server
3. Root DNS server to .edu DNS server
4. .edu DNS server to CS DNS server
5. CS DNS server to GAIA DNS server
6. GAIA DNS server to .edu DNS server
7. .edu DNS server to CS DNS server
8. CS DNS server to local DNS server
DNS: caching and updating records

- once (any) name server learns mapping, it `caches` mapping
  - cache entries `timeout` (disappear) after some time

- TLD (Top Level Domain) servers typically `cached` in local name servers
  - Thus root name servers not often `visited`
**DNS protocol, messages**

**DNS protocol**: query and reply messages, both with same **message format**

**msg header**
- $QR = 0$ for Q, 1 for response
- $Opcode= 0$ standard
- **identification**: 16 bit # for query, reply to query uses same #
- **flags**:
  - Authoritative answer
  - recursion desired
  - recursion available
  - reply is authoritative

<table>
<thead>
<tr>
<th>QR</th>
<th>Opcode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>flag</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>QR</td>
<td>Authoritative answer</td>
</tr>
<tr>
<td>Opcode</td>
<td>recursion desired</td>
</tr>
<tr>
<td>identification</td>
<td>recursion available</td>
</tr>
<tr>
<td>number of questions</td>
<td>number of answer RRs</td>
</tr>
<tr>
<td>number of authority RRs</td>
<td>number of additional RRs</td>
</tr>
<tr>
<td>questions</td>
<td>(variable number of questions)</td>
</tr>
<tr>
<td>answers</td>
<td>(variable number of resource records)</td>
</tr>
<tr>
<td>authority</td>
<td>(variable number of resource records)</td>
</tr>
<tr>
<td>additional information</td>
<td>(variable number of resource records)</td>
</tr>
</tbody>
</table>
## DNS protocol, messages

Name, type fields for a query

RRs in response to query

records for authoritative servers

Information about nameserver

additional "helpful" info that may be used

<table>
<thead>
<tr>
<th>QR</th>
<th>OPCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>identification</td>
<td>flags</td>
</tr>
<tr>
<td>number of questions</td>
<td>number of answer RRs</td>
</tr>
<tr>
<td>number of authority RRs</td>
<td>number of additional RRs</td>
</tr>
</tbody>
</table>

- questions (variable number of questions)
- answers (variable number of resource records)
- authority (variable number of resource records)
- additional information (variable number of resource records)
DNS records

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

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

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

- **Type=AAAA**
  - name is hostname
  - value is IPv6 address

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

- **Type=MX**
  - value is name of mailserver associated with name
### DNS Record example

#### RRs in response to query

<table>
<thead>
<tr>
<th>NAME</th>
<th>Design.cs.rutgers.edu</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>A</td>
</tr>
<tr>
<td>CLASS</td>
<td>IN</td>
</tr>
<tr>
<td>TTL</td>
<td>1 day (86400)</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>192.26.92.30</td>
</tr>
</tbody>
</table>

#### records for authoritative servers

<table>
<thead>
<tr>
<th>NAME</th>
<th>Cs.rutgers.edu</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>NS</td>
</tr>
<tr>
<td>CLASS</td>
<td>IN</td>
</tr>
<tr>
<td>TTL</td>
<td>1 day (86400)</td>
</tr>
<tr>
<td>NSDNAME</td>
<td>Ns-lcsr.rutgers.edu</td>
</tr>
</tbody>
</table>
DNS

DNS services

- Hostname to IP address translation

- Load distribution
  - Replicated Web servers: set of IP addresses for the same name
Bootstrapping DNS

- How does a host contact the name server if all it has is the name and no IP address?
- IP address of at least 1 nameserver must be given in advance
  - or with another protocol (DHCP, bootp)

- File /etc/resolv.conf in unix

- Start -> settings-> control panel-> network ->TCP/IP -> properties in windows
Themes

- Request/response nature of these protocols
- How Messages are structured
  - HTTP, SMTP, FTP - simple ASCII protocols
- Caching
- Name Lookup
  - Division of concerns (e.g. zones)
  - Hierarchy structure
HTTP
Web and HTTP

First some jargon
- Web page consists of objects
- Object can be HTML file, JPEG image, Java applet, audio file,…
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL (Uniform Resource Locator)
- Example URL: www.cs.rutgers.edu/undergraduate/pic.gif
  - host name
  - path name
HTTP overview

HTTP: hypertext transfer protocol

- client/server model
  - **client**: browser that requests, receives, “displays” Web objects
  - **server**: Web server sends objects in response to requests

- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068
HTTP messages: request message

- HTTP request message:
  - ASCII (human-readable format)

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr
```

(request line (GET, POST, HEAD commands))

(header lines)

(extra carriage return, line feed)

Carriage return, line feed indicates end of message
Client server connection

IP Address, 80

http messages
HTTP request message: general format

```
method  sp  URL  sp  version  cr  lf
header field name : value  cr  lf
                   ...
header field name : value  cr  lf
cr  lf
```

Entity Body

http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14
**Method types**

- **GET**
  - Get the file specified in the path URL field in entity body

- **POST**
  - accept the entity enclosed in the entity body as a new subordinate of the resource identified by the URL field

- **PUT**
  - uploads file in entity body to path specified in URL field

- **DELETE**
  - deletes file specified in the URL field
Post method - Upload form input

Post method:
- Web page often includes form input
- Input is uploaded to server in entity body

URL method:
- Uses GET method
- Input is uploaded in URL field of request line
Example: Client POST request

POST /cgi-bin/rats.cgi HTTP/1.0
Referer: http://nes:8192/cgi-bin/rats.cgi
Connection: Keep-Alive
User-Agent: Mozilla/4.73 [en] (X11; U; Linux 2.2.12-20 i686)
Host: nes:8192
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
        image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Content-type: application/x-www-form-urlencoded
Content-length: 93

Account=cs111fall&First=Alice&Last=White&SSN=123456789&Bday=01011980&State=CreateAccount

http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14
http response message: general format

Unlike http request, No method name

```
version  sp  status code  sp  phrase
header field name  :  value  cr  lf

(header lines)
```

Entity Body
HTTP message: response message

status line
  (protocol
  status code
  status phrase)

HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 ..... 
Content-Length: 6821
Content-Type: text/html

data, e.g.,
  requested
  HTML file

data data data data data data ...

HTTP response status codes

In first line in server->client response message.
A few sample codes:

200 OK
  ❖ request succeeded, requested object later in this message

301 Moved Permanently
  ❖ requested object moved, new location specified later in this message (Location:)

400 Bad Request
  ❖ request message not understood by server

404 Not Found
  ❖ requested document not found on this server

505 HTTP Version Not Supported
Additional about HTTP

- Persistent vs. Nonpersistent HTTP connections
- Cookies (User-server state)
- Web caches
HTTP connections

Nonpersistent HTTP
- At most one object is sent over a single TCP connection.
- HTTP/1.0 uses nonpersistent HTTP

Persistent HTTP
- Multiple objects can be sent over a single TCP connection between client and server.
- HTTP/1.1 uses persistent connections in default mode

TCP is a kind of communication service provided by the transport layer. It requires the connection to be set up before data communication.
Nonpersistent HTTP

Suppose user enters URL
www.someSchool.edu/someDepartment/home.index

1a. HTTP client initiates TCP connection to HTTP server

1b. HTTP server at host “accepts” connection, notifying client

2. HTTP client sends HTTP request message

3. HTTP server receives request message, replies with response message containing requested object
Nonpersistent HTTP (cont.)

4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects
HTTP: Response time

Definition of RTT (Round-Trip Time): time to send a small packet to travel from client to server and back.

Response time:
- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time

total = 2RTT + transmit time
**Persistent vs. Nonpersistent**

**Nonpersistent HTTP issues:**
- requires 2 RTTs per object
  - TCP Connection and HTTP Request
- Browsers can open parallel TCP connections to fetch referenced objects

**Persistent HTTP**
- server leaves TCP connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
HTTP: user-server state

HTTP is “stateless”

- server maintains no information about past client requests

- What state can bring:
  - authorization
  - shopping carts
  - recommendations
  - user session state
Cookies: keeping “state”

- **Client**
  - Usual HTTP request msg
  - Usual HTTP response + Set-cookie: 1678

- **Server**
  - Server creates ID 1678 for user

- **Cookie File**
  - Amazon: 1678
  - Ebay: 8734

- **One Week Later:**
  - Usual HTTP request msg
  - Cookie: 1678
  - Usual HTTP response msg
Cookies (continued)

Four components:
1) cookie header line of HTTP response message
2) cookie header line in HTTP request message
3) cookie file kept on user’s host, managed by user’s browser
4) back-end database at Web site
Cookies (continued)

Cookies and privacy:
- cookies permit sites to learn a lot about you
Web caches (proxy server)

Why?
- Reduce response time for client request.
- Reduce traffic on an institution’s access link.
Web caches (proxy server)

- browser sends all HTTP requests to cache
  - Hit: cache returns object
  - Miss: cache requests object from origin server, then returns object to client
Web caches: implementation

- Conditional Get guarantees cache content is up-to-date while still saves traffic and response time whenever possible.

```
HTTP request msg
If-modified-since: <date>
```

```
HTTP response
HTTP/1.0
304 Not Modified
```

```
HTTP request msg
If-modified-since: <date>
```

```
HTTP response
HTTP/1.0
200 OK
<data>
```
Content Distribution Networks (CDN)

Why?
- Reduce bandwidth requirements of content provider
- Reduce $$ of maintaining Servers
- Cache for server content
- Reduce traffic on the link to the content provider.
- Improve response time to user
Without CDN

- Huge B/W requirements
- Does not scale
- So, Distribute content to geographically distributed servers
- Use DNS to redirect request to copies of user content

<table>
<thead>
<tr>
<th>DOMAIN NAME</th>
<th>IP ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://WWW.YAHOO.COM">WWW.YAHOO.COM</a></td>
<td>98.138.253.109</td>
</tr>
<tr>
<td>cs.rutgers.edu</td>
<td>128.6.4.2</td>
</tr>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
<td>74.125.225.243</td>
</tr>
<tr>
<td><a href="http://www.princeton.edu">www.princeton.edu</a></td>
<td>128.112.132.86</td>
</tr>
</tbody>
</table>

Origin Server

98.138.253.109
CDN terms

- **Origin server**
  - Server that holds the authoritative copy of the content

- **CDN server**
  - A replica server owned by the CDN provider

- **CDN name server**
  - A DNS-like name server used for redirection

- **Client**
## With CDN

### CDN Name Server (124.8.9.8)

<table>
<thead>
<tr>
<th>DOMAIN NAME</th>
<th>IP ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://WWW.YAHOO.COM">WWW.YAHOO.COM</a></td>
<td>12.1.2.3</td>
</tr>
<tr>
<td><a href="http://www.yahoo.com">www.yahoo.com</a></td>
<td>12.1.2.4</td>
</tr>
<tr>
<td><a href="http://www.yahoo.com">www.yahoo.com</a></td>
<td>12.1.2.5</td>
</tr>
<tr>
<td><a href="http://www.yahoo.com">www.yahoo.com</a></td>
<td>12.1.2.6</td>
</tr>
</tbody>
</table>

### Using CDN

- **CDN servers**: 12.1.2.3, 12.1.2.4, 12.1.2.5, 12.1.2.6
- **Origin server**: 98.138.253.109
- **Client**
FTP
Client server connection

- Host name
- IP Address

DNS

IP Address, port 21

ftp commands

Data transfer, port 20
FTP: the file transfer protocol

- transfer file to/from remote host
- client/server model
  - **client**: side that initiates transfer (either to/from remote)
  - **server**: remote host
- ftp: RFC 959
- ftp server: port 21, port 20 (data connection)
FTP: separate control, data connections

- “out of band” control
  - Control connection:
    - Authorization
    - Directory browse
    - Commands
  - Data connection
    - Transfer files
- FTP server maintains “state”: current directory, earlier authentication
FTP commands, responses

Sample commands:
- sent as ASCII text over control channel
- USER username
- PASS password
- LIST return list of file in current directory
- RETR filename retrieves (gets) file
- STOR filename stores (puts) file onto remote host

Sample return codes:
- status code and phrase (as in HTTP)
- 331 Username OK, password required
- 125 data connection already open; transfer starting
- 425 Can’t open data connection
- 452 Error writing file
FTP Active connection (Data connection initiated form server)

- Client opens a connection from port x for sending commands to server port 21
- Server opens a connection from port 20 to send data at port x+1
FTP passive connection (always client initiated)

- Client opens a connection from port x for sending commands to server port 21
- Client sends a request for PASSIVE connection with PASV command
- Server replies with a new port number Sp on which it is listening
- Client opens a connection from port x+1 to server port Sp
FTP

- Sends passwords in plain ASCII text
  - Eavesdropper can recover passwords
  - Fatal flaw, turned off at a lot of sites
  - Replaced with scp, sftp instead
SMTP
Electronic Mail

Three major components:

1. user agents
   - a.k.a. “mail reader”
   - e.g., gmail, Outlook, yahoo

![Diagram showing the components of electronic mail system: user agents, mail server, outgoing message queue, user mailbox, using SMTP protocol.](image-url)
2. Mail Servers
   - mailbox contains incoming messages for user
   - message queue of outgoing (to be sent) mail messages

   - Sender mail server makes connection to Receiver mail server
     - IP address, port 25

3. SMTP protocol
   - Used to send messages
   - Client: sending user agent or sending mail server
   - server: receiving mail server
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

220 hill.com SMTP service ready

HELO town.com
250 hill.com Hello town.com, pleased to meet you

MAIL FROM: <jack@town.com>
250 <jack@town.com>… Sender ok

RCPT TO: <jill@hill.com>
250 <jill@hill.com>… Recipient ok

DATA
354 Enter mail, end with “.” on a line by itself

Jill, I’m not feeling up to hiking today. Will you please fetch me a pail of water?
.

250 message accepted

QUIT

221 hill.com closing connection
Table 23.2  Responses

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Positive Completion Reply</strong></td>
</tr>
<tr>
<td>211</td>
<td>System status or help reply</td>
</tr>
<tr>
<td>214</td>
<td>Help message</td>
</tr>
<tr>
<td>220</td>
<td>Service ready</td>
</tr>
<tr>
<td>221</td>
<td>Service closing transmission channel</td>
</tr>
<tr>
<td>250</td>
<td>Request command completed</td>
</tr>
<tr>
<td>251</td>
<td>User not local; the message will be forwarded</td>
</tr>
<tr>
<td></td>
<td><strong>Positive Intermediate Reply</strong></td>
</tr>
<tr>
<td>354</td>
<td>Start mail input</td>
</tr>
<tr>
<td></td>
<td><strong>Transient Negative Completion Reply</strong></td>
</tr>
<tr>
<td>421</td>
<td>Service not available</td>
</tr>
<tr>
<td>450</td>
<td>Mailbox not available</td>
</tr>
<tr>
<td>451</td>
<td>Command aborted: local error</td>
</tr>
<tr>
<td>452</td>
<td>Command aborted; insufficient storage</td>
</tr>
<tr>
<td></td>
<td><strong>Permanent Negative Completion Reply</strong></td>
</tr>
<tr>
<td>500</td>
<td>Syntax error; unrecognized command</td>
</tr>
<tr>
<td>501</td>
<td>Syntax error in parameters or arguments</td>
</tr>
<tr>
<td>502</td>
<td>Command not implemented</td>
</tr>
<tr>
<td>503</td>
<td>Bad sequence of commands</td>
</tr>
<tr>
<td>504</td>
<td>Command temporarily not implemented</td>
</tr>
<tr>
<td>550</td>
<td>Command is not executed; mailbox unavailable</td>
</tr>
<tr>
<td>551</td>
<td>User not local</td>
</tr>
<tr>
<td>552</td>
<td>Requested action aborted; exceeded storage location</td>
</tr>
<tr>
<td>553</td>
<td>Requested action not taken; mailbox name not allowed</td>
</tr>
<tr>
<td>554</td>
<td>Transaction failed</td>
</tr>
</tbody>
</table>
Mail access protocols

- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
  - IMAP: Internet Mail Access Protocol [RFC 1730]
  - HTTP: Hotmail, Yahoo! Mail, etc.
Mail message (stored on server) format

SMTP: protocol for exchanging email msgs
RFC 822: standard for text message format:
  - header lines, e.g.,
    - To:
    - From:
    - Subject:
      different from SMTP commands!
  - body
    - the “message”, ASCII characters only
Message format: multimedia extensions

- MIME: multimedia mail extension, RFC 2045, 2056
- additional lines in msg header declare MIME content type

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data ......
...........................
......base64 encoded data
```
SMTP: final words

Comparison with HTTP:

- HTTP: pull
- SMTP: push

- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg
Try SMTP interaction for yourself:

- telnet servername 25
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

above lets you send email without using email client (reader)
Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

   telnet www.cs.rutgers.edu 80

   Opens connection to port 80
   (default HTTP server port).
   Anything typed in sent
   to port 80 at www.eden.rutgers.edu

2. Type in a GET HTTP request:

   GET /~badri/index.php
   Host: www.eden.rutgers.edu

   By typing this in (hit carriage
   return twice), you send
   this minimal (but complete)
   GET request to HTTP server

3. Look at response message sent by HTTP server!