## The Application Layer

It involves interesting, real applications doing work for the end user.

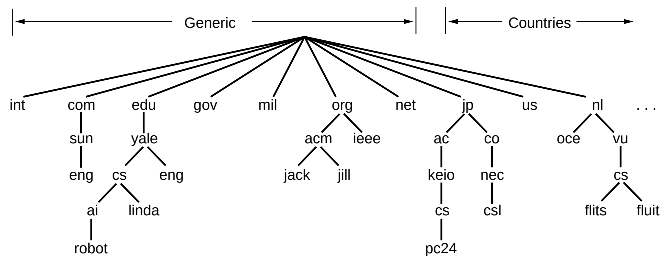
- Primary function: (probably) to provide an interface to the end user.
- Some protocols exist for enhancing services to applications:
  - ☐ Security protocols (see Chapter 8)
  - ☐ Domain Name System (DNS)
  - ☐ Electronic Mail
  - ☐ The World Wide Web
- Common applications:
  - ☐ Email readers
  - ☐ USENET
  - ☐ Web browsers
  - ☐ Multimedia:
    - ∘ Audio representation
    - ∘ Streaming audio, Internet radio, VoIP
    - ∘ Video representation
    - ∘ Video on Demand
  - ☐ ... (ad infinitum)

## The Domain Name System (DNS)

- Problem: how to resolve domain names to IP addresses?
- DNS properties
  - ☐ Hierarchical domain-based naming scheme
  - ☐ Distributed database to implement it
  - ☐ Original definition in RFCs 1034, 1035
- Operation:
  - ☐ Application program calls library routine **resolver**
  - ☐ Resolver sends UDP packet to DNS server
  - ☐ Server returns IP address to resolver
  - ☐ Resolver returns address to application

## DNS Name Space

- Similar to postal system naming hierarchy
- Several hundred top-level domains
  - ☐ Domains are partitioned into subdomains
  - ☐ Subdomains are partitioned further...
- Original top-level domains:
  - ☐ Countries
  - ☐ Generic: com, edu, gov, int, mil, net, org
- A domain is named by the path upward to an unnamed root.
- Illustration:



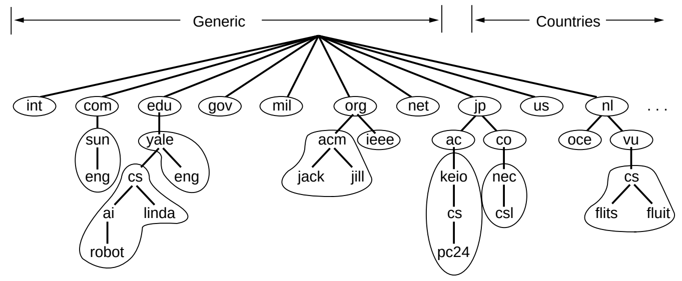
## Resource Records

- Most common: IP address
- Record format includes:
  - ☐ domain name
  - ☐ time to live
  - ☐ class
  - ☐ type
  - ☐ value
- Record types:
  - ☐ SOA - start of authority
  - ☐ A - IP address of a host
  - ☐ MX - mail exchange
  - ☐ NS - name server
  - ☐ CNAME - canonical name
  - ☐ PTR - pointer
  - ☐ HINFO - host description
  - ☐ TXT - text

## Name Servers

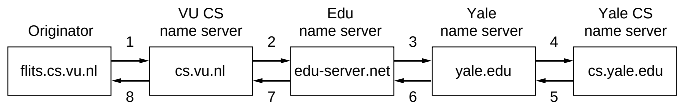
In theory, the whole world could be served by one name server.

- Practically inefficient due to
  - ☐ overloading
  - ☐ fault tolerance
- Solution: divide DNS name space into nonoverlapping *zones* (exclusive subnets served by a single name server).



## Recursive Query

- Name resolution:
  - ☐ Authoritative record: find it on disk
  - ☐ Non-authoritative record: send a query to the top-level server for the domain requested
- Example:



## Electronic Mail

One of the original Internet applications.

- History and background:
  - ☐ Used only in academics before 1990, now creates vastly more volume than *snail mail*.
  - ☐ Much less formal than postal mail (e.g. use of *emoticons* and stupid (IMNSHO) contractions)
- Development
  - ☐ Originally sent via ftp
  - ☐ Numerous complaints led to RFC 821 and RFC 822, the basis for modern email
  - ☐ 1984: CCITT drafts X.400 recommendation. This has disappeared ...

## Architecture and Services

Normally consists of two subsystems

- User agents: allow people to read/send email
- Message Transfer Agents: move the messages from source to destination
- Five basic functions are supported:
  - ☐ Composition
  - ☐ Transfer
  - ☐ Reporting
  - ☐ Displaying
  - ☐ Disposition

## The User Agent

Normally a program that accepts a variety of commands for composing, receiving, replying, etc.

- Addressing formats:
  - ☐ DNS-based: user@dns-address (example: swturner@umich.edu)
  - ☐ X.400: Attribute=value pairs separated by slashes
  - ☐ Example: /C=US/ST=MASSACHUSETTS/L=CAMBRIDGE...

## Message Formats (RFC 822)

Messages consist of a primitive envelope (see RFC 821), header fields, blank line, and message body

- Principal header fields:

| Header | Meaning |
|---|---|
| To: | E-mail address(es) of primary recipient(s) |
| Cc: | E-mail address(es) of secondary recipient(s) |
| Bcc: | E-mail address(es) for blind carbon copies |
| From: | Person or people who created the message |
| Sender: | E-mail address of the actual sender |
| Received: | Line added by each transfer agent along the route |
| Return-Path: | Can be used to identify a path back to the sender |

- Other common header fields:

| Header | Meaning |
|---|---|
| Date: | The date and time the message was sent |
| Reply-To: | E-mail address to which replies should be sent |
| Message-Id: | Unique number for referencing this message later |
| In-Reply-To: | Message-Id of the message to which this is a reply |
| References: | Other relevant Message-Ids |
| Keywords: | User-chosen keywords |
| Subject: | Short summary of the message for the one-line display |

## The Multipurpose Internet Mail Extensions (MIME)

In the early days, all email was in English and used ASCII.

- Problems with this approach:
  - ☐ Cannot represent various non-English characters or character sets
  - ☐ Cannot represent non-textual data (e.g. audio, video)
- MIME (RFC 1341, 2045-2049):
  - ☐ Continues to use RFC 822 format with more structure plus encoding rules
  - ☐ Five new message headers:

| Header | Meaning |
|---|---|
| MIME-Version: | Identifies the MIME version |
| Content-Description: | Human-readable string telling what is in the message |
| Content-Id: | Unique identifier |
| Content-Transfer-Encoding: | How the body is wrapped for transmission |
| Content-Type: | Type and format of the content |

## The Simple Mail Transfer Protocol (SMTP)

Message transfer is only concerned with relaying messages from the sender to the recipient.

- SMTP is implemented over TCP connections between source/destination:
  - ☐ SMTP daemon listens on port 25 at destination machine
  - ☐ SMTP daemon on source machine makes TCP connection to destination and simply transfers the data.
  - ☐ Simple ASCII dialog to connect, send data, and disconnect ensues.
- This assumes that the intended recipient works on a machine capable of sending and receiving email.

## Final Delivery

A problem occurs when the recipient is not always connected to the Internet.

- Solution is to store mail on ISP machine and use a final delivery protocol
- Two primary protocols exist: POP3 and IMAP
- POP3:
  - ☐ Assumes user will clear the mailbox on every contact, then work offline
  - ☐ Mail reader establishes TCP connection; protocol has three states:
    1. authorization
    2. transactions
    3. update
- IMAP:
  - ☐ Assumes all email remains on server indefinitely
  - ☐ Maintains extensive mechanisms for
    - ○ reading all/part of messages
    - ○ creating, destroying, manipulating multiple mailboxes
  - ☐ many features are supported...

## Comparison of POP3 and IMAP

- POP3 is simple/quick while IMAP has many features

| Feature | POP3 | IMAP |
|---|---|---|
| Where is protocol defined | RFC 1939 | RFC 2060 |
| TCP port used | 110 | 143 |
| Where is e-mail stored | User's PC | Server |
| Where is e-mail read | Off-line | On-line |
| Connect time required | Little | Much |
| Use of server resources | Minimal | Extensive |
| Multiple mailboxes | No | Yes |
| Who backs up mailboxes | User | ISP |
| Good for mobile users | No | Yes |
| User control over downloading | Little | Great |
| Partial message downloads | No | Yes |
| Are disk quotas a problem | No | Could be in time |
| Simple to implement | Yes | No |
| Widespread support | Yes | Growing |

## The World Wide Web

- History
  - ☐ Began at CERN, the European center for nuclear research
  - ☐ Grew out of need:
    - ◦ large teams of researchers from many countries
    - ◦ experiments taking years of planning and construction
    - ◦ Constantly changing collections of reports, blueprints, etc.
  - ☐ 1989: Tim Berners-Lee proposal; text-based prototype operational in late 1990
  - ☐ 1991: public demonstration at Hypertext '91 conference
  - ☐ 1993: Marc Andreessen releases Mosaic at UIUC
  - ☐ 1994: World Wide Web Consortium (W3C) created by CERN and MIT
  - ☐ 1994: Marc Andreessen founds Netscape

## Client Side Behavior

The browser is essentially a program that displays a Web page and catches mouse clicks to items on the displayed page.

- Typical sequence:
  - ☐ User clicks on a hyperlink and
    1. browser determines URL
    2. browser performs DNS lookup
    3. DNS server replies with translation
    4. browser makes TCP connection on port 80 to address
    5. browser sends a request using HTTP
    6. server sends requested file(s) using HTTP
    7. TCP connection is released
    8. browser displays the text
    9. browser repeats for associated files (e.g. embedded graphics)

## Displaying Other Things

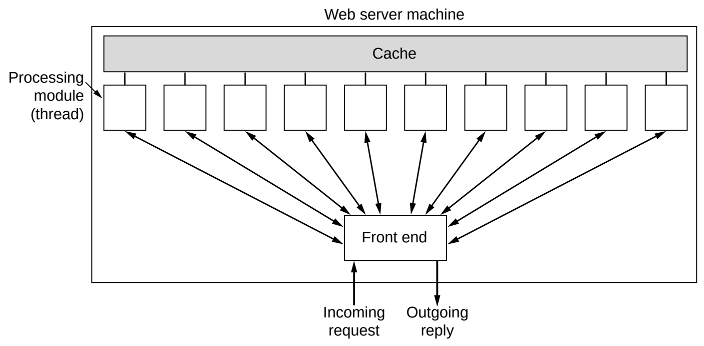
The browser can basically display text and graphics

- Problem: rapidly growing collection of file types
- Solution: pages contain information about page including MIME types for various embedded data
- When a browser cannot display it directly, it consults a table
- Two display possibilities
  - ☐ Plug-in: code module the browser fetches and installs within itself
  - ☐ Helper-application: separate process that is notified of content to display
- Pitfalls?

## Server Side Behavior

Performance issues:

- Simple “connect, get file off disk, return file, disconnect” model is inefficient:
  - ☐ Fastest disks require at least 5 msec, limiting performance
  - ☐ Performance improvements:
  - ☐ Caching
  - ☐ Multithreading
  - ☐ Parallel disks



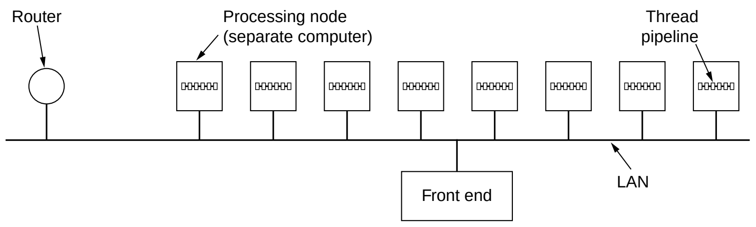
## Server Side Processing

Actual processing of requests can be complicated

- For a given request, front-end module passes request to first available processing module
- Request *may* require:
  1. Resolve the name of the requested page
  2. Authenticate client
  3. Access control on client
  4. Access control on web page
  5. Check the cache
  6. Fetch requested page from disk
  7. Determine MIME type to include in response
  8. Take care of miscellaneous items
  9. Return the reply to client
  10. Make entry in server log

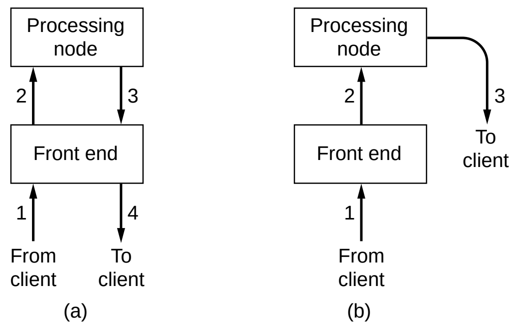
## Server Farms

- Single-CPU systems be overloaded even with parallel disks
- Solution is *server farm*:



## Server Farm Issues

- No shared cache except on shared-memory multiprocessors
  - ☐ Solution: front-end tracks requests and routes them
- Client TCP connection terminates at the front-end
  - ☐ Solution:



## Uniform Resource Locators

- Consist of three parts:
  - ☐ Protocol (or Scheme)
  - ☐ DNS name of machine on which page is located
  - ☐ Local name uniquely identifying page on that machine
- Common URLs:

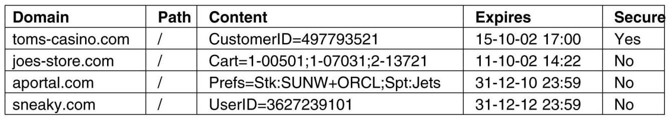
| Name | Used for | Example |
|---|---|---|
| http | Hypertext (HTML) | http://www.cs.vu.nl/~ast/ |
| ftp | FTP | ftp://ftp.cs.vu.nl/pub/minix/README |
| file | Local file | file:///usr/suzanne/prog.c |
| news | Newsgroup | news:comp.os.minix |
| news | News article | news:AA0134223112@cs.utah.edu |
| gopher | Gopher | gopher://gopher.tc.umn.edu/11/Libraries |
| mailto | Sending e-mail | mailto:JohnUser@acm.org |
| telnet | Remote login | telnet://www.w3.org:80 |

- Inherent weakness?

## Statelessness and Cookies

There is no concept of a login session on the web.

- Model of statelessness dictates that the server forgets the client once a file is served.
- Problems:
  - ☐ Requiring clients to pay money to use a service
  - ☐ How to track "shopping cart" items
  - ☐ Displaying custom web portal content
- Tracking the IP address will not work
- cookies: small files stored on client side containing vital information
  - ☐ Examples:



| Domain | Path | Content | Expires | Secure |
|---|---|---|---|---|
| toms-casino.com | / | CustomerID=497793521 | 15-10-02 17:00 | Yes |
| joes-store.com | / | Cart=1-00501;1-07031;2-13721 | 11-10-02 14:22 | No |
| aportal.com | / | Prefs=Stk:SUNW+ORCL;Spt:Jets | 31-12-10 23:59 | No |
| sneaky.com | / | UserID=3627239101 | 31-12-12 23:59 | No |