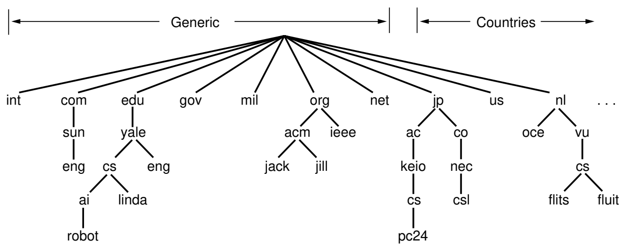
The Domain Name System

Basic idea: Every host has a worldwide unique name that is bound to an IP address. DNS provides name-lookup facilities: when given a hostname, it returns that host's IP address.

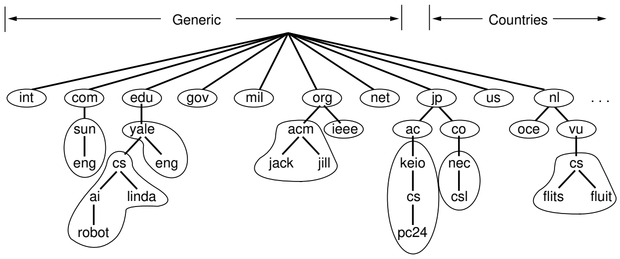


A **domain name** is a path from a leaf node up to the root.
A **domain** is a subtree in the domain name space.

Each domain can have a set of **resource records** which are stored (in a file) at name servers. There are different types of records, including one for storing the binding between a name and an IP address.

DNS Name Servers

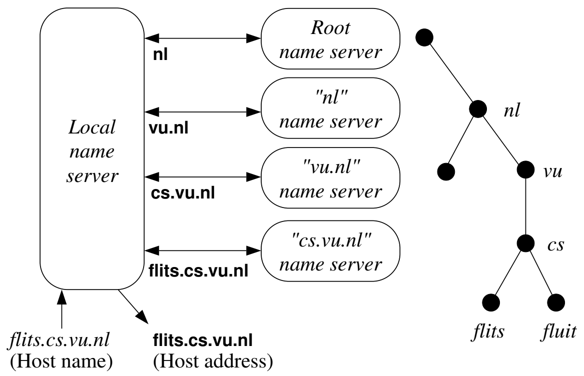
Basic idea: Divide the name space into a collection of non-overlapping **zones**, and let each zone be taken care of by one or more **name servers**:



Note: There can be several servers per zone. Most of the time, *secondary masters* pull in their information from a *primary master*. The latter gets its info from domain administrators.

A **resolver** is capable of sending DNS queries to a name server. A resolver is often just a collection of library routines that can be linked to an application.

DNS Name Resolution



Note: Name resolution can be **iterative** (as drawn), or **recursive**. In the latter case, a higher level server passes the query to a lower one instead of passing it back to the querying server.

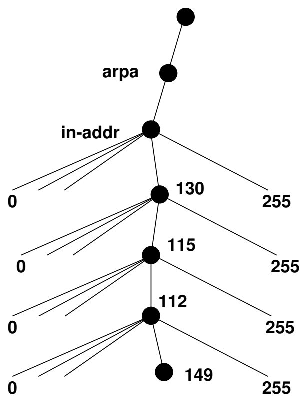
Important: The main reason why DNS works is that it assumes that name–to–address bindings hardly change. This means that we can effectively cache bindings locally, saving the trouble of having to go to the actual name server.

DNS Address Resolution

Guess what: You can also find the name of a host when given its address:

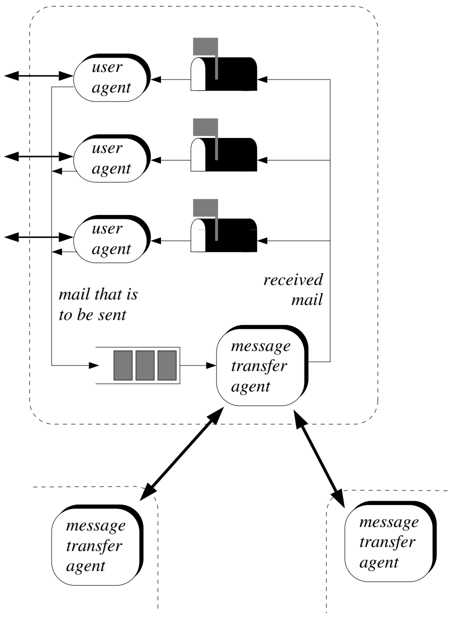
```
sealine % host 130.115.112.149
Name:  iridium.cs.few.eur.nl
Address:  130.115.112.149
Aliases:

sealine %
```

Solution: IP addresses are stored in the special in-addr.arpa domain:



Electronic Mail

Main issue: Mail is really not that interesting: you just send mail to a mail server who takes care about delivery:



Email: Message Transfer

Basic idea: The message transfer agent extracts the destination host from the message, and queries DNS to obtain the destination address. DNS keeps track of mailers in MX records:

cs.vu.nl mail is handled (pri=10) by zephyr.cs.vu.nl
* cs.vu.nl mail is handled (pri=10) by tornado.cs.vu.nl
cs.vu.nl mail is handled (pri=20) by top.cs.vu.nl
cs.vu.nl mail is handled (pri=30) by solo.cs.vu.nl

Example: mail for steen@cs.vu.nl is first sent to the message transfer agent on zephyr.cs.vu.nl, then tornado, etc. Of course, zephyr is looked up as well.

The message transfer agent extracts the user and makes an attempt to deposit the incoming message into the user's mailbox. The user may then be notified.

Note: We're assuming that the user's mailbox is accessible for the agent, not necessarily the user. Protocols exist that allow a user to remotely access the mailbox.

*

USENET News

Basic idea: Users can post articles and responses to articles, which are subsequently stored at the local News server.

Spreading the news: News servers jointly constitute a worldwide directed graph. A News server can act as a **newsfeed** to other servers:

- A server can request articles from its newsfeed(s) ⇒ it **pulls** articles to its own site.
- A server can automatically upload articles to the servers for it is a newsfeed ⇒ articles are **pushed** to other sites.

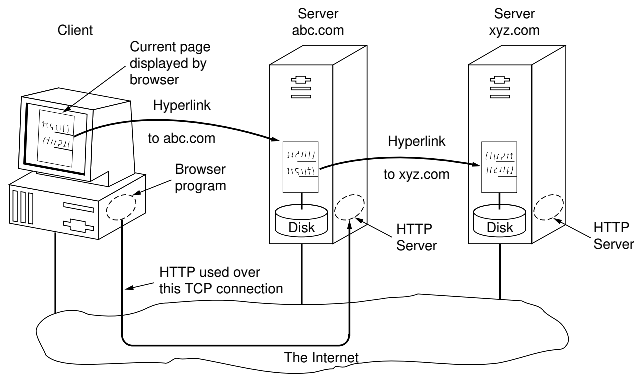
Note: In this model, it is possible that a response to an article is posted earlier at a specific site than the article itself (which may need to come from another newsfeed).

World Wide Web

Basic model: Users and organizations maintain pages of information that contain references to each other as **hyperlinks**. Selecting a link has the effect of pulling in the referenced page.

- The **client** has a Web browser that can display Web pages. Web pages are formatted in a special **markup language** which is interpreted by the browser. This allows for fancy typefonts and the like.
- A hyperlink identifies a (remote) Web server that has access to the referred Web page. When selecting a link, the browser establishes a TCP connection to the server, and the page is transferred to the user.
- A Web server listens for connection requests, accepts one request, returns the page over the connection, and closes it again. A connected client and server speak HTTP (HyperText Transfer Protocol).

WWW – Client/Server Interaction

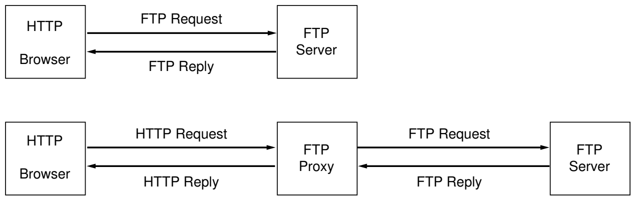


Note: When a client receives a new Web page, it can access other servers through the links contained in that new page.

Note: Each special feature in a page (i.e. in-line images) is transferred *separately* after the page has been copied to the client. This also means establishing and releasing several connections to the original server.

WWW – Proxy Servers

Basic idea: It is also possible to transfer information referenced by a different transfer protocol (notably FTP). If the browser does not speak that protocol, a proxy can be used:



Note: It is also possible to install proxies to restrict Web access, or to do Web page caching. The latter turns out to be reasonably effective: the proxy cache simply keeps previously requested pages. Yes, there is also a serious drawback...

URL – Uniform Resource Locators

Essence: A **URL** contains three informative parts: (1) the name of a page, (2) the name of its location, (3) the access protocol:

Name	Usage	Example
http	Hypertext	http://www.cs.vu.nl/~steen/cn/
ftp	File transfer	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:/home/steen/www/cn/index.html
news	News group	news:comp.os.minix
news	News article	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libs
mailto	Sending email	mailto:kim@acm.org
telnet	Remote login	telnet://www.w3.org:80

Disadvantage: URLs contain location information: they refer to the location where a page is found. This makes it much harder to move pages around and to replicate them. In both cases, you don't care where the page is, but just that it has a worldwide unique name.

The Web's New Toy: Java

Basic idea: Why just download static stuff from a Web server? It's much more exciting if you can download a program as part of the page, and that is subsequently *automatically* executed.

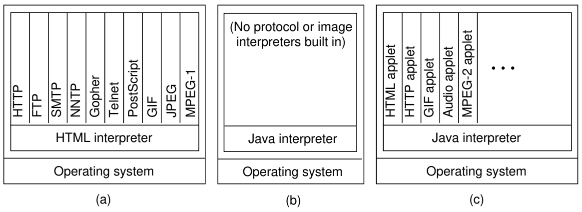
Great 1: We can have animations on our screen! But also programs that allow you to fill in forms (i.e. interact with the user), and when you're finsihed, do some calculations, and contact some server.

Great 2: If a Web page requires some special protocol that you don't have, just attach an implementation of that protocol to the page, download when fetching that page, and execute at the client's side.

Not so great: Would you trust foreign code? We have to be ensured that the attached program doesn't do any harm at the client's side. Not a trivial problem.

Java – The Basics

- As a developer, you construct a Java program (which is very similar to C++).
- The Java program is compiled into interpretable and portable **bytecode**. A Java-enabled browser has a built-in bytecode interpreter.
- The compiled program is referenced in the Web page so that it is automatically downloaded when the page is fetched.



Multimedia

Observation: Multimedia applications are (going to be) the hot topics for computer networks. Audio and video form an important component in information exchange between users, i.e. communication, and need network support.

Main issue: We need a lot of bandwidth to transmit audio/video data, while at the same time the data stream should be highly stable (i.e. no jitter).

Audio: We sample at a sufficiently high rate (think of Nyquist), and encode the values into bit strings. CDs are sampled at 44.1 kHz and encoded into 16 bits. This gives us a raw bandwidth of 705.6 kbps per CD channel.

Video: (Consider only digital systems.) We need a sequence of frames, each frame being a matrix of pixels. A pixel has **luminance** (brightness) and **chrominance** (color), jointly encoded into 8, 16, or 24 bits. We need at least 25 frames/second to avoid flickering. An SVGA display (1024×768 pixels at 24-bit encoding) gives a raw bandwidth of 472 Mbps.

Data Compression

Entropy encoding: manipulate the stream of bits as is – don't make use of what those bits stand for:

- **Run-length encoding:** consecutive repeated symbols are replaced by one symbol.
- **Statistical encoding:** common symbols are encoded in less bits than uncommon symbols.
- **CLUT encoding:** assume only small fraction of all possible colors are used. Store used values in a table, and encode them by their index.

Data Compression II

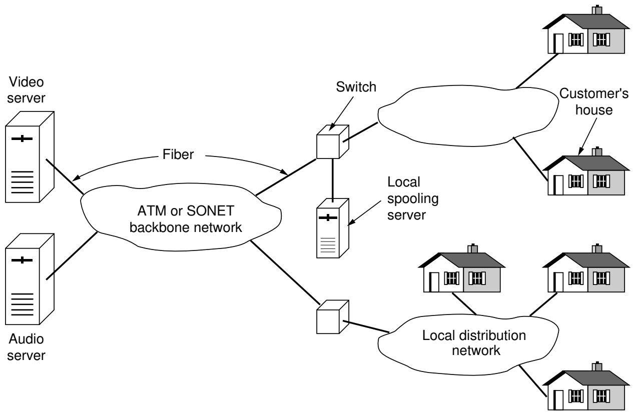
Source encoding: Use the properties of the data to do the encoding. May actually result in loss of information (but which may not be that bad):

- **Differential encoding:** encode a block of values by considering the difference with respect to its predecessor. Assumes representing differences requires less bits than the actual value.
- **Transformation:** encode a block of data by transforming it into another domain, perhaps with some loss of information (example: Fourier).
- **Vector quantization:** Similar to CLUT: you build a **code book** and send the index (perhaps with the code book).

Note: Read Tanenbaum to get an impression of what JPEG and MPEG do. Don't learn those standards.

Video-on-Demand

Two models: (1) The user can request any video, and stop the incoming stream at any time. (2) The provider broadcasts popular videos, but starts several at short intervals (near video on demand).



Video-on-Demand II

Note: The video server can be pretty hot stuff: it requires *a lot* of storage, which will include tape archives, optical disks (as in a juke box), and arrays of magnetic disks (RAIDs).

Note: The client will have a **set-top box**, which is just a simple computer that does the (MPEG) decoding, has an interface to the network, and controls the TV set (monitor & remote control).

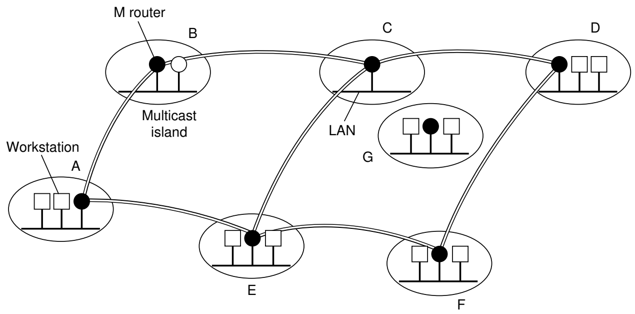
Big problem: How are we ever going to get the bandwidth to our homes (the local loop problem).

Really big problem: There are so many possible solutions, and all of them cost too much. What we'll need is a convergence of solutions to standards. This means a lot of experimentation in the coming years.

MBone: Multimedia on the Internet

Back to basics: Let's do it the Internet way and just a hack a simple multimedia subsystem on top of what we already have: a worldwide data network.

Basic idea: Construct an **overlay network** on top of the Internet, where each node constitutes a **multicast island**. Islands communicate by tunneling packets. Construct multicast trees on the overlay network:



MBone: Routing

- Each island has one or more **multicast routers**, which form the end points of the tunnels.
- We start a broadcast by picking a class D IP address (reserved for multicasting) and multicast it on our island. The mrouter picks it up and forwards it onto its tunnels.
- Incoming packets at an mrouter are forwarded along a spanning tree using reverse shortest path first. This means that only those packets are forwarded that came in along the best route to the source.
- From time to time, mrouters query the hosts on their island to figure out which broadcasts they are interested in. This information is also used to prune multicast trees on the MBone.

Major MBONE Nodes and Links