Address Conversion Functions and The Domain Name System

Refs: Chapter 9
RFC 1034
RFC 1035

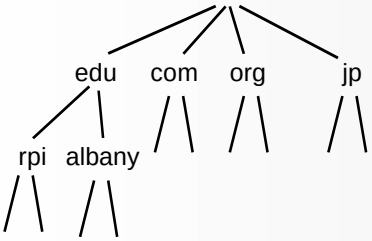
Hostnames

- IP Addresses are great for computers
  - IP address includes information used for routing.
- IP addresses are tough for humans to remember.
- IP addresses are impossible to guess.
  - ever guessed at the name of a WWW site?

The Domain Name System

- The *domain name system* is usually used to translate a host name into an IP address .
- Domain names comprise a hierarchy so that names are unique, yet easy to remember.

DNS Hierarchy



Host name structure

- Each host name is made up of a sequence of *labels* separated by periods.
  - Each label can be up to 63 characters
  - The total name can be at most 255 characters.
- Examples:
  - whitehouse.gov
  - barney.the.purple.dinosaur.com
  - monica.cs.rpi.edu

Domain Name

- The domain name for a host is the sequence of labels that lead from the host (leaf node in the naming tree) to the top of the worldwide naming tree.
- A domain is a subtree of the worldwide naming tree.

Top level domains

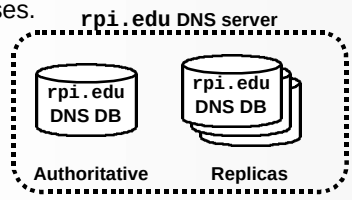
- **`edu, gov, com, net, org, mil`**, …
- Countries each have a top level domain (2 letter domain name).
- New top level domains proposed.

DNS Organization

- Distributed Database
  - The organization that owns a domain name is responsible for running a DNS server that can provide the mapping between hostnames within the domain to IP addresses.
  - So - some machine run by RPI is responsible for everything within the rpi.edu domain.

DNS Distributed Database

- There is one primary server for a domain, and typically a number of secondary servers containing replicated databases.



DNS Clients

- A DNS client is called a *resolver*.
- A call to **`gethostbyname()`** is handled by a resolver (typically part of the client).
- Most Unix workstations have the file /etc/resolv.conf that contains the local domain and the addresses of DNS servers for that domain.

/etc/resolv.conf

```
domain rpi.edu
128.113.1.5
128.113.1.3
```


`nslookup`

- nslookup is an interactive resolver that allows the user to communicate directly with a DNS server.
- nslookup is usually available on Unix workstations.

DNS Servers

- Servers handle requests for their domain directly.
- Servers handle requests for other domains by contacting remote DNS server(s).
- Servers cache external mappings.

Server - Server Communication

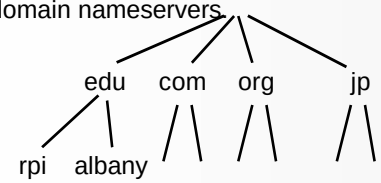
- If a server is asked to provide the mapping for a host outside it's domain (and the mapping is not in the server cache):
  - The server finds a nameserver for the target domain.
  - The server asks the nameserver to provide the host name to IP translation.
- To find the right nameserver, use DNS!

DNS Data

- DNS databases contain more than just hostname-to-address records:
  - Name server records NS
  - Hostname aliases CNAME
  - Mail Exchangers MX
  - Host Information HINFO

The Root DNS Server

- The root server needs to know the address of 1st (and many 2nd) level domain nameservers.



Server Operation

- If a server has no clue about where to find the address for a hostname, ask the root server.
- The root server will tell you what nameserver to contact.
- A request may get forwarded a few times.

DNS Message Format

HEADER
QUERIES
***Response* RESOURCE RECORDS**
***Response* AUTHORITY RECORDS**
***Response* ADDITIONAL INFORMATION**

DNS Message Header

16 bit fields

- query identifier
- flags
- # of questions
- # of RRs
- # of authority RRs
- # of additional RRs

} **Response**

Message Flags

- QR: Query=0, Response=1
- AA: Authoritative Answer
- TC: response truncated (> 512 bytes)
- RD: recursion desired
- RA: recursion available
- rcode: return code

Recursion

- A request can indicate that recursion is desired - this tells the server to find out the answer (possibly by contacting other servers).
- If recursion is not requested - the response may be a list of other name servers to contact.

Question Format

- Name: domain name (or IP address)
- Query type (A, NS, MX, …)
- Query class (1 for IP)

Response Resource Record

- Domain Name
- Response type
- Class (IP)
- Time to live (in seconds)
- Length of resource data
- Resource data

UDP & TCP

- Both UDP and TCP are used:
  - TCP for transfers of entire database to secondary servers (replication).
  - UDP for lookups
  - If more than 512 bytes in response - requestor resubmits request using TCP.

Lots more

- This is not a complete description !
- If interested - look at:
  - RFC 1034: DNS concepts and facilities.
  - RFC 1035: DNS implementation and protocol specification.
  - play with nslookup.
  - Look at code for BIND (DNS server code).

Name to Address Conversion

- There is a library of functions that act as DNS client (resolver).
  - you don't need to write DNS client code to use DNS!
- With some OSs you need to explicitly link with the DNS resolver library:
  `-lnsl` (**`nsl`** is "Name Server Library")

Suns (Solaris) need this!

DNS library functions

`gethostbyname`

`gethostbyaddr`

`gethostbyname2` ← IPV6!

gethostbyname

```
struct hostent *gethostbyname(
  const char *hostname);
```

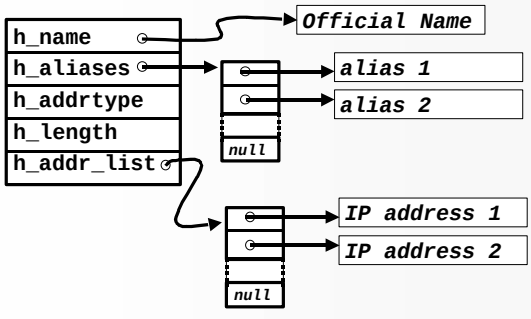
struct hostent is defined in netdb.h:

```
#include <netdb.h>
```


struct hostent

```
struct hostent {
  char *h_name;       official name (canonical)
  char **h_aliases;   other names
  int h_addrtype;     AF_INET or AF_INET6
  int h_length;       address length (4 or 16)
  char **h_addr_list;
};                    array of ptrs to addresses
```


hostent picture



Which Address?

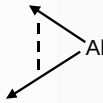
On success, gethostbyname returns the address of a hostent that has been created.

- has an array of ptrs to IP addresses
- Usually use the first one:

```
#define h_addr h_addr_list[0]
```


`gethostbyname` and errors

- On error **`gethostbyname`** return null.
- **`Gethostbyname`** sets the global variable h_errno to indicate the exact error:
  - **`HOST_NOT_FOUND`**
  - **`TRY_AGAIN`**
  - **`NO_RECOVERY`**
  - **`NO_DATA`**
  - **`NO_ADDRESS`**



All defined in **`netdb.h`**

Getting at the address: `char **h_addr_list;`

```
h = gethostbyname("joe.com");
sockaddr.sin_addr.s_addr =
  *(h->h_addr_list[0]);
```

This won't work!!!!

`h_addr_list[0]` is a **`char*`** !

Using **memcpy**

- You can copy the 4 bytes (IPv4) directly:

```
h = gethostbyname("joe.com");

memcpy(&sockaddr.sin_addr,
       h->h_addr_list[0],
       sizeof(struct in_addr));
```


Network Byte Order

- All the IP addresses returned via the hostent are in network byte order!
- Repeat after me:
  "Thank you **gethostbyname**!"

gethostbyaddr

```
struct hostent *gethostbyaddr(
  const char *addr
  size_t len,   ← sizeof(struct in_addr)
  int family);
```

AF_INET (could be AF_INET6)

Some other functions

`uname` : get hostname of local host

`getservbyname` : get port number for a named service

`getservbyaddr` : get name for service associated with a port number