CpSc 360: Distributed and Network Programming

DNS and IP Address Conversion

James Wang

Textbook Chapter 11

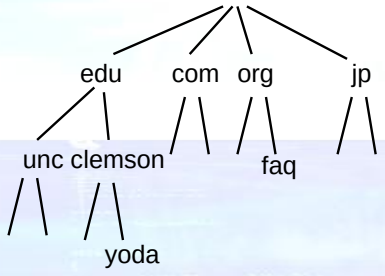
http://www.faqs.org/rfcs/rfc1034.html
http://www.faqs.org/rfcs/rfc1035.html
http://www.faqs.org/rfcs/rfc1886.html

Why Name Conversion?

- **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* 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.**

2

DNS Hierarchy



3

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:**
  - nsf.gov
  - barney.the.purple.dinosaur.com
  - yoda.cs.clemson.edu

4

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 include:

    .aero .biz .coop .info .name .pro

5

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

6

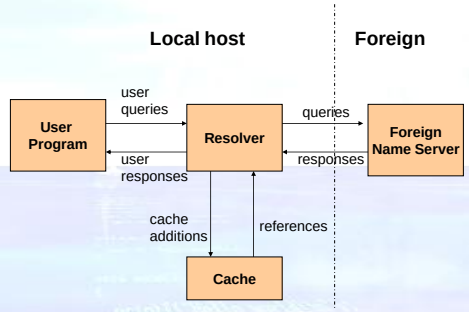
DNS Clients

- 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.
- Content of "/etc/resolv.conf" in machine mmlab in G17:

```
; generated by /sbin/dhclient-script
search cs.clemson.edu clemson.edu
nameserver 130.127.8.8
nameserver 130.127.28.14
nameserver 130.127.48.3
```

7

Typical Configuration



8

nslookup

- `nslookup` is an interactive resolver that allows the user to communicate directly with a DNS server.
- `nslookup` is usually available on Unix workstations. (`dig` and `host` are also DNS clients).

9

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.
- 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!

10

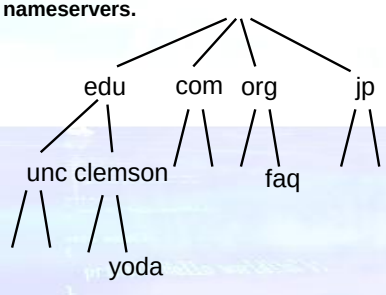
DNS Data

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

11

The Root DNS Server

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



12

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.
- Try run "dig".

13

Domain Name Space Definitions

- Domain names in messages are expressed in terms of a sequence of labels. Each label is represented as a one octet length field followed by that number of octets.
- Since every domain name ends with the null label of the root, a domain name is terminated by a length byte of zero.
- The high order two bits of every length octet must be zero, and the remaining six bits of the length field limit the label to 63 octets or less.
- To simplify implementations, the total length of a domain name (i.e., label octets and label length octets) is restricted to 255 octets or less.
- Name servers and resolvers must compare labels in a case-insensitive manner (i.e., A=a), assuming ASCII with zero parity. Non-alphabetic codes must match exactly.

14

Resource Record (RR) Definitions

- All RRs have the same top level format that include 6 fields:
  - **NAME:** an owner name, i.e., the name of the node to which this resource record pertains.
  - **TYPE:** two octets containing one of the RR TYPE codes.
  - **CLASS:** two octets containing one of the RR CLASS codes.
  - **TTL:** a 32 bit signed integer that specifies the time interval that the resource record may be cached before the source of the information should again be consulted.
  - **RDLENGTH:** an unsigned 16 bit integer that specifies the length in octets of the RDATA field.
  - **RDATA:** a variable length string of octets that describes the resource. The format of this information varies according to the TYPE and CLASS of the resource record.

15

Communications

- All communications inside of the domain protocol are carried in a single format called a message.
- The top level format of message is divided into 5 sections (some of which are empty in certain cases).

Header	
Question	the question for the name server
Answer	RRs answering the question
Authority	RRs pointing toward an authority
Additional	RRs holding additional information

16

DNS Message Format

- The header section is always present. The header includes fields that specify which of the remaining sections are present, and also specify whether the message is a query or a response, a standard query or some other opcode, etc.
- The question section contains fields that describe a question to a name server. These fields are a query type (QTYPE), a query class (QCLASS), and a query domain name (QNAME).
- The last three sections have the same format: a possibly empty list of concatenated resource records (RRs).
  - The answer section contains RRs that answer the question;
  - the authority section contains RRs that point toward an authoritative name server;
  - the additional records section contains RRs which relate to the query, but are not strictly answers for the question.

17

DNS Message Header

16 bit fields

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

} Response

18

Message Flags

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

19

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.

20

Question section format

- The question section is used to carry the "question" in most queries, i.e., the parameters that define what is being asked.
  - **QNAME:** a domain name represented as a sequence of labels, where each label consists of a length octet followed by that number of octets. The domain name terminates with the zero length octet for the null label of the root. Note that this field may be an odd number of octets; no padding is used.
  - **QTYPE:** a two octet code which specifies the type of the query. The values for this field include all codes valid for a TYPE field, together with some more general codes which can match more than one type of RR.
  - **QCLASS:** a two octet code that specifies the class of the query. For example, the QCLASS field is IN for the Internet.

21

Message Transmission

- 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.
- More:
  - RFC 1034: DNS concepts and facilities.
  - RFC 1035: DNS implementation and protocol specification.
  - play with nslookup, dig.
  - Look at code for BIND (DNS server code).

22

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!

23

DNS library functions

All defined in "netdb.h"

- gethostbyname() (pp. 307)
- gethostbyaddr() (pp. 310)
- getservbyname() and getservbyport() (pp. 311)
- getaddrinfo() (pp. 315 to 324)
- gethostbyname2() (for IPv6 only)
- getnameinfo() (pp. 340)
- more … please check "/usr/include/netdb.h"

24

gethostbyname

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

`struct hostent` **is defined in netdb.h:**

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

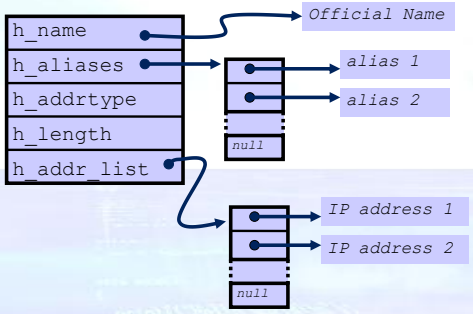
25

struct hostent

```
struct hostent {
  char *h_name;  /* Official name */
  char **h_aliases; /* other names */
  int h_addrtype; /* AF_INET or AF_INET6 */
  int h_length; /* address length */
  char **h_addr_list;
       /* array of pointers to addresses */
};
```

26

hostent picture



27

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]
```

28

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`
- **All the IP addresses returned via the hostent are in network byte order!**

29

Don't try assigning pointer

?? 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*` **!**

30

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));
```

31

`gethostbyaddr()`

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

sizeof(struct in_addr)

AF_INET (could be AF_INET6)

32

Reading Assignment

- **Please read textbook chapter 11.**
- **Read RFCs if your have time.**
- **Try nslookup, dig, host.**
- **Have fun!**

33