Lecture 11: Domain Name Systems

12/2/2020

Lecture Outline

1. Domain name systems, domain name servers, and resolvers
2. gethostbyname function
3. gethostbyname2 function
4. gethostbyaddr function
5. uname and gethostname functions
6. getservbyname and getservbyport functions

1. Domain name systems, domain name servers, and resolvers

(1) Domain name systems

a. Basic concept

b. Structure of DNS (Fig.11.1, p.306)

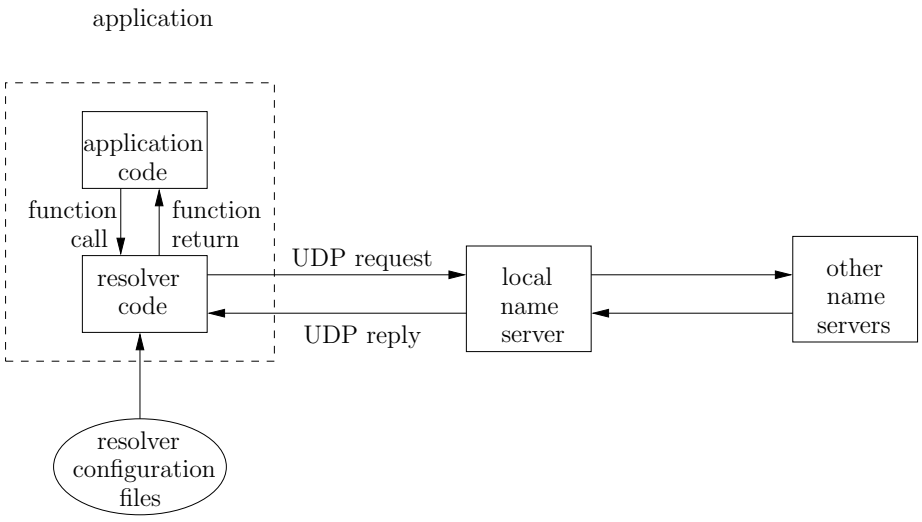

Figure 118: Typical arrangement of clients, resolvers, and name servers. (Fig.11.1,p.306)

(2) Domain name servers

a. Types of name servers

(a) Primary name servers

(b) Secondary name servers

(c) Cache servers

(d) Forwarder

(e) DNS clients

b. Types of files of a primary name server

(a) The forward mapping file

(b) The reverse mapping file

c. Resource records (RRs) in mapping files

(a) Entries in mapping files are called resource records

(b) Types of RRs

A Such a record maps a hostname into a 32-bit IPv4 address

AAAA Also called "quad A" record. Such a record maps a hostname into a 128-bit IPv6 address

PTR Such a record maps an IP address (IPv4 or IPv6) into a hostname.

MX Such a record specifies the so called *mail exchanger* for a specific host. Multiple MX RRs can be associated with any specific host, each of which can have a designated priority.

CNAME Such a record specifies the *canonical name* (i.e. another name) of a particular host.

d. BIND (Berkeley Internet Name Domain): a popular name server software

(3) Domain name resolvers

a. The *gethostbyname*() function

b. The *gethostbyaddr*() function

(4) DNS alternatives

a. The static approach: /etc/hosts file

b. NIS (network information system) and NIS+

2. The *gethostbyname* fucntion

(1) Syntax

```
#include <netdb.h>
struct hostent *gethostbyname(const char *hostname);
```

(2) Semantics: given an IP name (IPv4 or IPv6) as the single argument, it returns the official name, list of aliases (nicknames), and list of IPv4 address in the structure pointed by the return pointer.

(3) The *hostent* structure

a. The structure

```
struct hostent {
    char    *h_name;         /* canonical name of host */
    char    **h_aliases;     /* alias list */
    int     h_addrtype;      /* host address type */
    int     h_length;        /* length of address */
    char    **h_addr_list;   /* list of addresses */
};

#define h_addr h_addr_list[0]   /* first address in list */
```

b. Illustration of the structure (Fig.11.2, p.308)

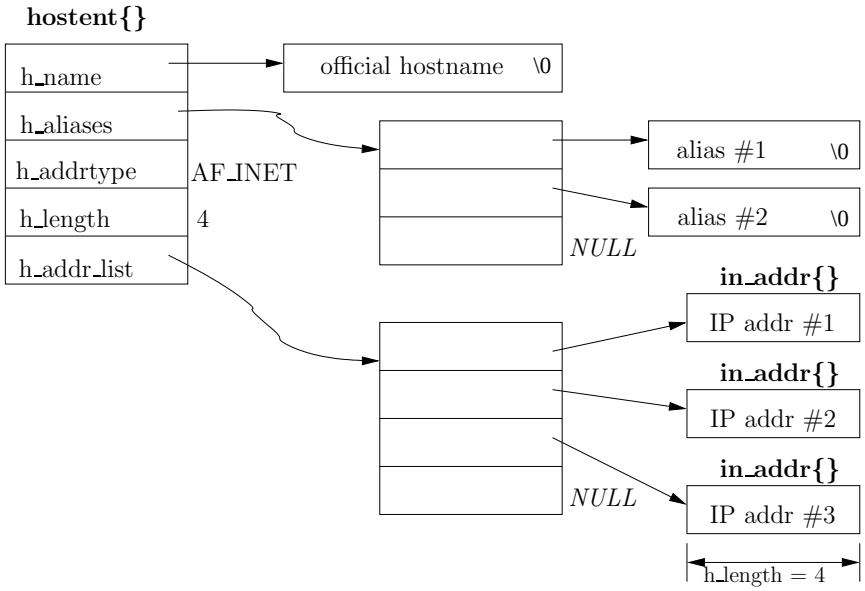

Figure 119: *hostent* structure and the information it contains. (Fig.11.2,p.308)

c. The changes in structure for IPv6 addresses: Fig.11.3, p.242 of the 2nd ed. of the textbook

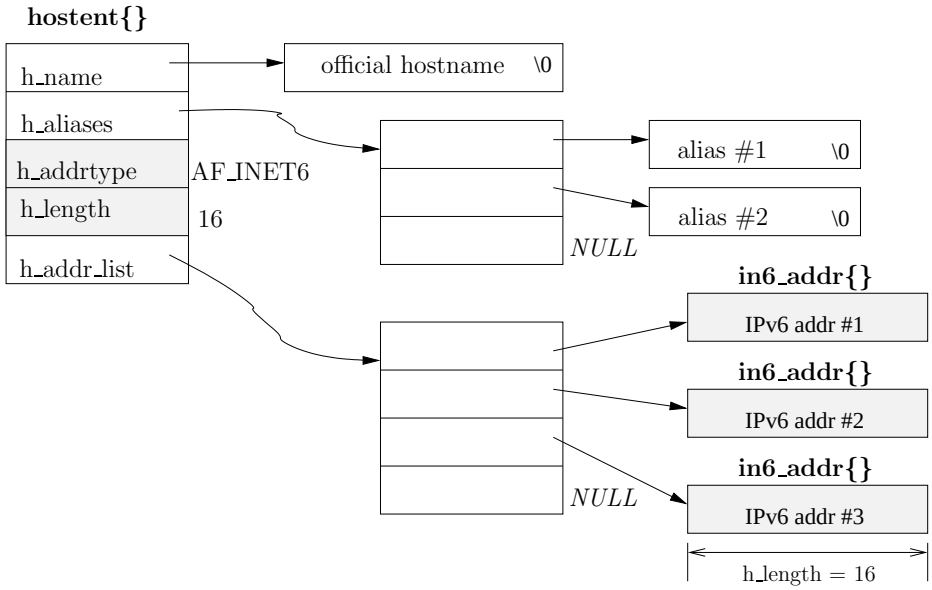

Figure 120: Changes in information returned in *hostent* structure with IPv6 addresses. (Fig.9.3,p.242 of 2nd ed. of textbook)

(4) Example (Fig.11.4,p.309): a sample program that calls the function *gethostbyname*. The program can take any number of command-line parameters, each of which is regarded as a hostname. The program calls the *gethostbyname* for each of such parameters.

```
1  #include "unp.h"

2  int
3  main(int argc, char **argv)
4  {
5      char *ptr, **pptr;
6      char str[INET6_ADDRSTRLEN];
7      struct hostent *hptr;

8      while (--argc > 0) {
9          ptr = *++argv;
10         if ( (hptr = gethostbyname(ptr)) == NULL) {
11             err_msg("gethostbyname error for host: %s: %s",
12                     ptr, hstrerror(h_errno));
13             continue;
14         }
15         printf("official hostname: %s\n", hptr->h_name);
```

```
16          for (pptr = hptr->h_aliases; *pptr != NULL; pptr++)
17               printf("\talias: %s\n", *pptr);

18          switch (hptr->h_addrtype) {
19          case AF_INET:
20              pptr = hptr->h_addr_list;
21              for ( ; *pptr != NULL; pptr++)
22                  printf("\taddress: %s\n",
23                        Inet_ntop(hptr->h_addrtype, *pptr, str, sizeof(str)));
24              break;
25          default:
26              err_ret("unknown address type");
27              break;
28          }
29      }
30      exit(0);
31  }
```

3. *gethostbyname2* function

(1) The *RES_USE_INET6* resolver option

a. This option was added to newer version of the BIND (4.9.4 and later) to allow a resolver to return (to the *gethostbyname* function) IPv6 addresses, instead of IPv4 addresses.

b. This option can be set using three different ways (p.245).

(2) Syntax

#include <netdb.h>
struct hostent *gethostbyname2(const char **hostname*, int *family*);

(3) Semantics: similar to gethostbyname. Its action depends on the second argument (AF_INET, returns A record, or AF_INET6, returns AAAA record).

4. *gethostbyaddr* function

(1) Syntax

#include <netdb.h>
struct hostent *gethostbyaddr(const char *_addr_, size_t _len_, int _family_);

(2) Semantics: given the IP address (IPv4 or IPv6) address as the first argument, it will return the official name, list of aliases (nicknames), and list of IPv4 address in the structure pointed by the return pointer.

(3) Notes:

a. Due to the last argument, there is no need of another function for IPv6.

b. However, when handling IPv6 addresses, this function behaves differently (three points, p.249)

5. *uname* and *gethostname* functions

(1) The *uname* function

a. This returns a structure that contains the name of the OS, the name of the computer, the OS release level, OS version number, and hardware architecturem of the current host.

b. Syntax:

#include <sys/utsname.h>
int uname(struct utsname *_name_);

The utsname structure has the following fields (on SUN Solaris):

```
struct utsname {
        char    sysname[_SYS_NMLN];
        char    nodename[_SYS_NMLN];
        char    release[_SYS_NMLN];
        char    version[_SYS_NMLN];
        char    machine[_SYS_NMLN];
};
```

c. Semantics: on successful return, the functions returns 0. The parameter *name* points to a location that contains the five fields of the utsname structure. On failure, it returns -1

d. Example: Fig.9.7, p.250. The function *my_addrs* calls the function *uname* first to make sure that the given hostname is indeed valid before calling the *gethostbyname* function:

```
#include "unp.h"
#include <sys/param.h>

char **
my_addrs(int *addrtype)
{
    struct hostent    *hptr;
    char            myname[MAXHOSTNAMELEN];

   /* if (gethostname(myname, sizeof(myname)) < 0) */
    if (uname(&myname) < 0)
        return(NULL);

    if ( (hptr = gethostbyname(myname)) == NULL)
        return(NULL);

    *addrtype = hptr->h_addrtype;
    return(hptr->h_addr_list);
}
```

e. The *uname* command: a utility command implemented using the *uname* function

(2) The *gethostname* function

a. This function is similar to the *uname* function. However, it only returns the name of the current host.

b. Syntax:

#include <unistd.h>
int gethostname(char **name*, size_t *namelen*);

c. Semantics: on successful call, it returns 0. The *name* parameter returns the name of the current host. It returns -1 on failure.

6. *getservbyname* and *getservbyport* functions

(1) Syntax

#include <netdb.h>
struct servent *getservbyname(const char **servname*, const char **protoname*);
struct servent *getservbyport(int *port*, const char **protoname*);

(2) Semantics:

a. *getservbyname*: given the name of service (such as telnet, ftp, finger), it returns the official service name, list of aliases of the service, port number for the service, and supported protocol (e.g. UDP, TCP)

b. *getservbyaddr*: given a port number for a service (such as 79, 19, 21), it returns the official service name, list of aliases of the service, port number for the service, and supported protocol (e.g. UDP, TCP)

(3) Example (Fig.9.8,p.253): a daytime client that uses *gethostbyname* and *getservbyname*

```
#include   "unp.h"

int main(int argc, char **argv)
{
   int               sockfd, n;
   char            recvline[MAXLINE + 1];
   struct sockaddr_in   servaddr;
   struct in_addr      **pptr;
   struct hostent      *hp;
   struct servent      *sp;

   if (argc != 3)
      err_quit("usage: daytimetcpcli1 <hostname> <service>");

   if ( (hp = gethostbyname(argv[1])) == NULL)
      err_quit("hostname error for %s: %s", argv[1], hstrerror(h_errno));

   if ( (sp = getservbyname(argv[2], "tcp")) == NULL)
      err_quit("getservbyname error for %s", argv[2]);

   pptr = (struct in_addr **) hp->h_addr_list;
   for ( ; *pptr != NULL; pptr++) {
      sockfd = Socket(AF_INET, SOCK_STREAM, 0);

      bzero(&servaddr, sizeof(servaddr));
      servaddr.sin_family = AF_INET;
      servaddr.sin_port = sp->s_port;
      memcpy(&servaddr.sin_addr, *pptr, sizeof(struct in_addr));
      printf("trying %s\n",
```

```
            Sock_ntop((SA *) &servaddr, sizeof(servaddr)));

        if (connect(sockfd, (SA *) &servaddr, sizeof(servaddr)) == 0)
            break;      /* success */
        err_ret("connect error");
        close(sockfd);
    }
    if (*pptr == NULL)
        err_quit("unable to connect");

    while ( (n = Read(sockfd, recvline, MAXLINE)) > 0) {
        recvline[n] = 0;   /* null terminate */
        Fputs(recvline, stdout);
    }
    exit(0);
}
```