Elementary Name and Address Conversions

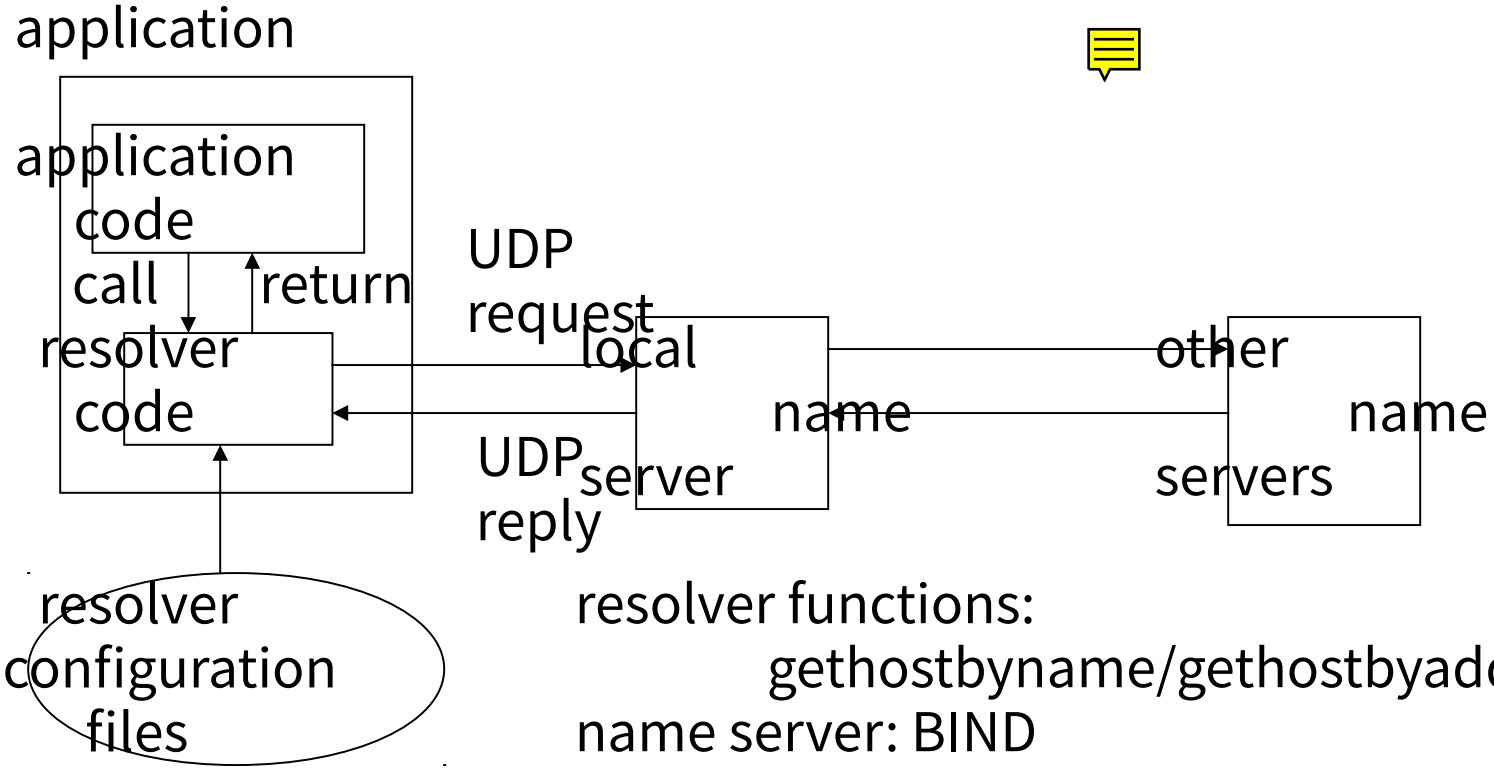
- Domain name system
- gethostbyname Function
- RES_USE_INET6 resolver option
- gethostbyname2 Function and IPv6 support
- gethostbyaddr Function
- uname and gethostname Functions
- getservbyname and getservbyport Functions
- Other networking information

Domain Name System

- Entries in DNS: resource records (RRs) for a host
  - A record: maps a hostname to a 32-bit IPv4 addr
  - AAAA (quad A) record: maps to a 128-bit IPv6 addr
  - PTR record: maps IP addr to hostname
  - MX record: specifies a mail exchanger of the host
  - CNAME record: assigns canonical name for common services

e.g.	solaris	IN	A	206.62.226.33
		IN	AAAA	5f1b:df00:ce3e:e200:0020:0800:2078:e3e3
		IN	MX	5 solaris.kohala.com
		IN	MX	10 mailhost.kohala.com
		IN	PTR	33.226.62.206.in-addr.arpa
	www	IN	CNAME	bsdi.kohala.com

DNS: Application, Resolver, Name Servers



resolver functions:
- gethostbyname/gethostbyaddr

name server: BIND
- (Berkeley Internet Name Domain)

static hosts files (DNS alternatives):
- /etc/hosts

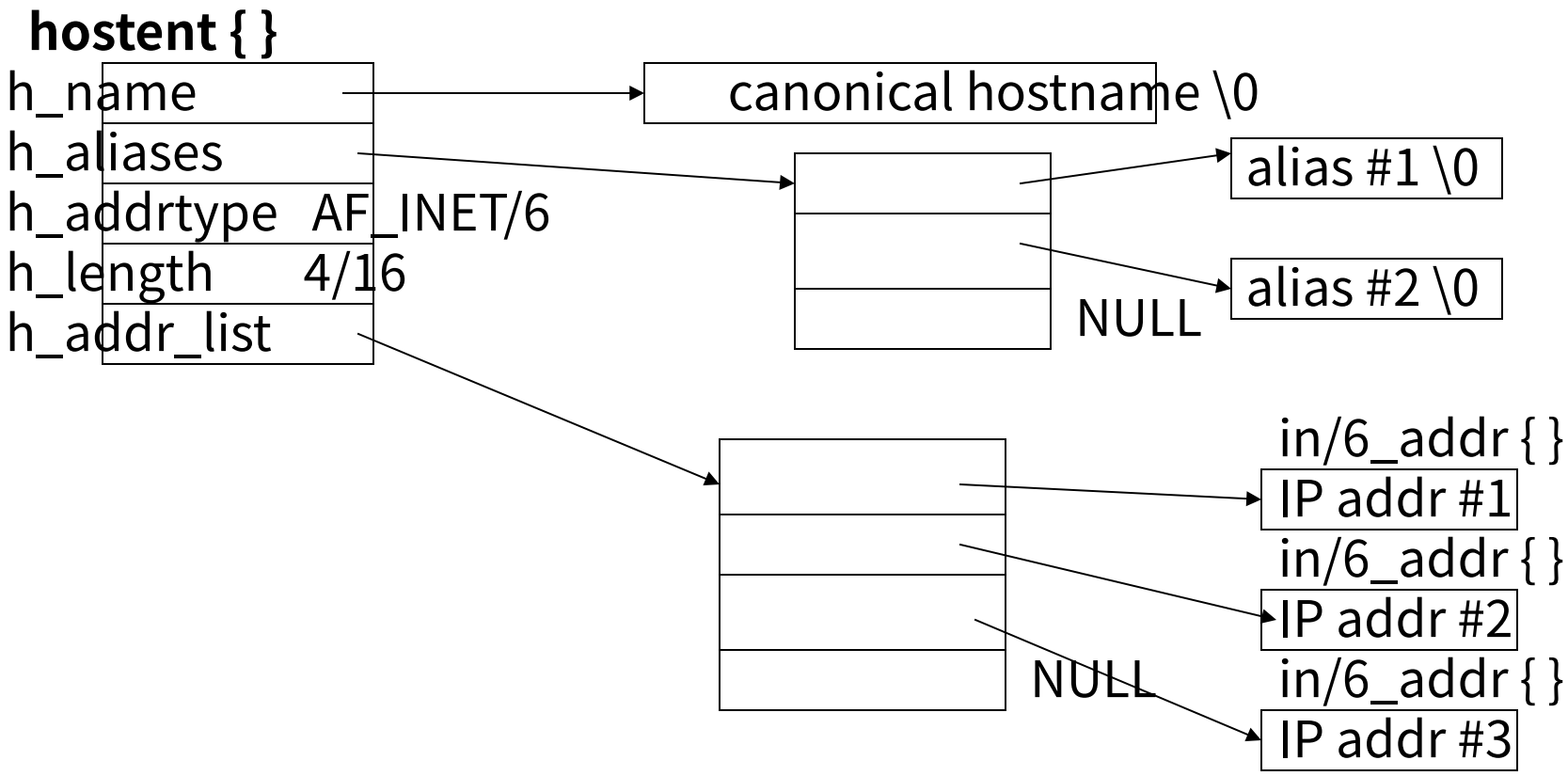
resolver configuration file (specifies name server IPs):
- /etc/resolv.conf

gethostbyname Function

performs a DNS query for an A record or a AAAA record

```
#include <netdb.h>
struct hostent *gethostbyname (const char *hostname);
        returns: nonnull pointer if OK, NULL on error with h_errno set
struct hostent {
        char    *h_name;        /* official (canonical) name of host */
        char    **h_aliases;    /* ptr to array of ptrs to alias names */
        int     h_addrtype;     /* host addr type: AF_INET or AF_INET6 */
        int     h_length;       /* length of address: 4 or 16 */
        char    **h_addr_list;  /* ptr to array of ptrs with IPv4/IPv6 addrs */
};
#define h_addr h_addr_list[0]  /* first address in list */
```

hostent Structure Returned by gethostbyname



```
 1 #include     "unp.h"

 2 int
 3 main(int argc, char **argv)
 4 {
 5     char   *ptr, **pptr;
 6     char     str [INET_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 }
```

RES_USE_INET6 Resolver Option

- Per-application: call res_init
  - #include <resolv.h>
  - res_init ( );
  - _res.options |= RES_USE_INET6
- Per-user: set environment variable RES_OPTIONS
  - export RES_OPTIONS=inet6
- Per-system: update resolver configuration file
  - options inet6 (in /etc/resolv.conf)
- For a host without a AAAA record, IPv4-mapped IPv6 addresses are returned.

gethostbyname2 Function and IPv6 Support

```
#include <netdb.h>
struct hostent *gethostbyname2 (const char *hostname, int family);
        returns: nonnull pointer if oK, NULL on error with h_errno set
```

	RES_USE_INET6 option	
	off	**on**
gethostbyname (*host*)	A record	AAAA record or A record returning IPv4-mapped IPv6 addr
gethostbyname2 (*host*, AF_INET)	A record	A record returning IPv4-mapped IPv6 addr
gethostbyname2 (*host*, AF_INET6)	AAAA record	AAAA record

gethostbyaddr Function
binary IP address to hostent structure

#include <netdb.h>
struct hostent *gethostbyaddr (const char **addr*, size_t *len*, int *family*);
returns: nonnull pointer if OK, NULL on error with h_errno set

addr argument: a pointer to an in_addr or in6_addr structure
h_name in hostent: canonical hostname
gethostbyaddr: queries a DNS name server for a PTR record
in the in-addr.arpa domain for IPv4 or a PTR record
in the ip6.int domain for IPv6.

uname and gethostname Functions

```
#include <sys/utsname.h>
int uname (struct utsname *name);
                    returns: nonnegative value if OK, -1 on error
struct utsname {
        char    sysname[_UTS_NAMESIZE]; /* name of OS */
        char    nodename[_UTS_NODESIZE]; /* name of this node */
        char    release[_UTS_NAMESIZE]; /* OS release level */
        char    version[_UTS_NAMESIZE]; /* OS version level */
        char    machine[_UTS_NAMESIZE]; /* hardware type */


#include <unistd.h>
int gethostname (char *name, size_t namelen);
                    returns: 0 if OK, -1 on error
```

getservbyname and getservbyport Functions

```
#include <netdb.h>
struct servent *getservbyname (const char *servname, const char *protoname)
                    returns: nonnull pointer if OK, NULL on error
struct servent *getservbyport (int port, const char *protoname);
                    returns: nonnull pointer if OK, NULL on error
struct servent {
        char    *s_name;        /* official service name */
        char    **s_aliases;    /*alias list */
        int     s_port;         /* port number, network-byte order */
        char    *s_proto;       /* protocol, TCP or UDP, to use */
```

Mapping from name to port number: in /etc/services

Services that support multiple protocols often use the same TCP and UDP port number. But it's not always true:

```
shell   514/tcp
syslog  514/udp
```

daytime Client
using gethostbyname and getservbyname

- (see Figure 9.8)
- Call gethostbyname and getservbyname
- Try each server address
- Call connect
- Check for failure
- Read server’s reply

```
 8    struct in_addr **pptr;
 9    struct in_addr *inetaddrp [2];
10    struct in_addr inetaddr;
11    struct hostent *hp;
12    struct servent *sp;

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

15    if ( (hp = gethostbyname (argv [1]) ) == NULL) {
16        if (inet_aton (argv [1], &inetaddr) == 0) {
17            err_quit ("hostname error for %s: %s", argv [1],
18                    hstrerror (h_errno) );
19        } else {
20            inetaddrp [0] = &inetaddr;
21            inetaddrp [1] = NULL;
22            pptr = inetaddrp;
23        }
24    } else {
25        pptr = (struct in_addr **) hp->h_addr_list;
26    }

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

29    for ( ; *pptr != NULL; pptr++) {
```

```
        pptr = inetaddrp;
    }
} else {
    pptr = (struct in_addr **) hp->h_addr_list;
}

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

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 */
```

Other Networking Info

- Four types of info:
  - hosts (gethostbyname, gethostbyaddr)
    - through DNS or /etc/hosts, hostent structure
  - networks (getnetbyname, getnetbyaddr)
    - through DNS or /etc/networks, netent structure
  - protocols (getprotobyname, getprotobynumber)
    - through /etc/protocols, protoent structure
  - services (getservbyname, getservbyport)
    - through /etc/services, servent structure