Basic Socket Interface Extensions for IPv6

yhmiu

RFC2553

Purpose

- Make changes to basic socket interface
  - Let it complete support both IPv4 and IPv6
- To support larger address size
  - IPv4 (32bits)
  - IPv6 (128bits)
- To support new features of IPv6
  - Some of new features of IPv6 must be made visible to applications via the API
    - Traffic class
    - flowlabel

What needs to be changed

- Core socket functions
- Address data structures
- Name-to-address translation functions
- Address conversion functions

Some new term about socket interface for IPv6

- IPv6 socket address structure
  - sockaddr_in6
- IPv6 Address Family and Protocol Family
  - AF_INET6 and PF_INET6
- IPv6 address structure
  - in6_addr

Socket address structure for 4.3BSD-based systems

```
struct sockaddr_in6 {
    sa_family_t        sin6_family;     /* AF_INET6 */
    in_port_t          sin6_port;       /* transport layer port # */
    uint32_t           sin6_flowinfo;   /* IPv6 traffic class & flow info */
    struct in6_addr    sin6_addr;       /* IPv6 address */
    uint32_t           sin6_scope_id;   /* set of interfaces for a scope */
};
```

Socket address structure for 4.4BSD-based systems

```
struct sockaddr_in6 {
    uint8_t            sin6_len;         /* length of this struct */
    sa_family_t        sin6_family;      /* AF_INET6 */
    in_port_t          sin6_port;        /* transport layer port # */
    uint32_t           sin6_flowinfo;    /* IPv6 flow information */
    struct in6_addr    sin6_addr;        /* IPv6 address */
    uint32_t           sin6_scope_id;    /* set of interfaces for a scope */
};
```

IPv6 address structure

```
struct in6_addr {
        union {
             uint8_t        _S6_u8[16];
             uint32_t       _S6_u32[4];
             uint64_t       _S6_u64[2];
        } _S6_un;
  };
  #define s6_addr    _S6_un._S6_u8
```

The socket functions (1)

- socket()
  - To create an IPv4/TCP socket
    - s=socket(PF_INET,SOCK_STERM,0)
  - To create an IPv4/UDP socket
    - s=socket(PF_INET,SOCK_DGRAM,0)
  - To create an IPv6/TCP socket
    - s=socket(PF_INET6,SOCK_STERM,0)
  - To create an IPv6/UDP socket
    - =socket(PF_INET6,SOCK_DGRAM,0)

The socket functions (2)

- bind(),connect(),sendmsg(),sendto(), accept(),recvfrom(),recvmsg(), getpeername(),getsockname()
  - **這些**socket functions**的**syntax**和原來版本的一樣，並且支援**IPv6

IPv6 wildcard address

- The applications want the system to select the source address for them
  - IPv4使用INADDR_ANY
  - IPv6使用in6addr_any
- Defined in <netinet6/in6.h>
  - extern const struct in6_addr in6addr_any;

Example of IPv6 wildcard address

```
struct sockaddr_in6 sin6;
    . . .
    sin6.sin6_family = AF_INET6;
    sin6.sin6_flowinfo = 0;
    sin6.sin6_port = htons(23);
    sin6.sin6_addr = in6addr_any;  /* structure assignment */
    . . .
    if (bind(s, (struct sockaddr *) &sin6, sizeof(sin6)) == -1)
          . . .
```

IPv6 loopback address

- Applications may need to send UDP packets to, or originate TCP connections to, services residing on the local node
  - IPv4使用INADDR_LOOPBACK
  - IPv6使用in6addr_loopback
- defined in <netinet6/in6.h>
  - extern const struct in6_addr in6addr_loopback;

Example of IPv6 loopback address

```
struct sockaddr_in6 sin6;
     . . .
    sin6.sin6_family = AF_INET6;
    sin6.sin6_flowinfo = 0;
    sin6.sin6_port = htons(23);
    sin6.sin6_addr = in6addr_loopback;  /* structure assignment */
     . . .
    if (connect(s, (struct sockaddr *) &sin6, sizeof(sin6)) == -1)
           . . .
```

Interface identification

- Name-to-Index
  - if_nametoindex()
- Index-to-Name
  - if_indextoname()
- Return all interface names and indexes
  - if_nameindex()
- Free memory
  - if_freenameindex()

Some new socket option defined for IPv6 (1)

- Can call getsockopt() and setsockopt() to manipulate the option associated with a socket
- The “level” parameter (the second parameter in getsockopt() and setsockopt() function call) is IPPROTO_IPV6
- The constant name prefix IPV6_ is used in all of the new socket option IPv6

Some new socket option defined for IPv6 (2)

- Such as
  - IPV6_UNICAST_HOPS
  - IPV6_MULTICAST_IF
  - IPV6_MULTICAST_HOPS
  - IPV6_MULTICAST_LOOP
  - IPV6_JOIN_GROUP
  - IPV6_LEAVE_GROUP

Example of socket options with setsockopt()

```
int  hoplimit = 10;
    if (setsockopt(s, IPPROTO_IPV6, IPV6_UNICAST_HOPS,
                (char *) &hoplimit, sizeof(hoplimit)) == -1)
       perror("setsockopt IPV6_UNICAST_HOPS");
```

Example of socket options with getsockopt()

```
int  hoplimit;
     size_t  len = sizeof(hoplimit);
     if (getsockopt(s, IPPROTO_IPV6, IPV6_UNICAST_HOPS,
                     (char *) &hoplimit, &len) == -1)
         perror("getsockopt IPV6_UNICAST_HOPS");
     else
         printf("Using %d for hop limit.\n", hoplimit);
```

New library functions needed for socket interface supporting IPv6

- Functions are needed to lookup IPv6 address in the DNS
- Functions are needed to convert IPv6 addresses between their binary and textual form

Nodename-to-address translation

- 在IPv4中使用gethostbyname()
- 在IPv6中使用getipnodebyname()
- Example
  - hptr = getipnodebyname(name, AF_INET6, AI_DEFAULT, &error_num);

Address-to-nodename translation

- 在IPv4中使用gethostbyaddr()
- 在IPv6中使用getipnodebyaddr()

其它和name-to-address translation有關的function

- freehostent()
  - 在記憶體中清掉getipnodebyname()和getipnodebyaddr()所產生的hostent structure和其所指的information
- getaddrinfo()和getnameinfo()
  - Protocol-independent
  - 會將其所有的socket address structures傳回
- freeaddrinfo()
  - 將getaddrinfo()和getnameinfo()所產生的addrinfo structure所佔的記憶體釋放

Address conversion functions

- inet_addr() and inetntoa()convert an IPv4 address between binary and text form
- inet_pton() and inet_ntop()convert an IPv6 address between binary and text form

Address testing macros

```
#include <netinet6/in6.h>
    int  IN6_IS_ADDR_UNSPECIFIED (const struct in6_addr *);
    int  IN6_IS_ADDR_LOOPBACK    (const struct in6_addr *);
    int  IN6_IS_ADDR_MULTICAST   (const struct in6_addr *);
    int  IN6_IS_ADDR_LINKLOCAL   (const struct in6_addr *);
    int  IN6_IS_ADDR_SITELOCAL   (const struct in6_addr *);
    int  IN6_IS_ADDR_V4MAPPED    (const struct in6_addr *);
    int  IN6_IS_ADDR_V4COMPAT    (const struct in6_addr *);

    int  IN6_IS_ADDR_MC_NODELOCAL(const struct in6_addr *);
    int  IN6_IS_ADDR_MC_LINKLOCAL(const struct in6_addr *);
    int  IN6_IS_ADDR_MC_SITELOCAL(const struct in6_addr *);
    int  IN6_IS_ADDR_MC_ORGLOCAL (const struct in6_addr *);
    int  IN6_IS_ADDR_MC_GLOBAL   (const struct in6_addr *);
```