



IPv6 Courses

©G6 Association

December 20, 2010



Table of Contents

- 1 Programming IPv6 Applications



- Group of IPv6 actors in France (researchers, engineers. . .)
- Academic & industrial partners
 - CNRS, Institut TELECOM, INRIA, Universities. . .
 - AFNIC, 6Wind, Bull. . .
- Launched in 1995 by:
 - Alain Durand
 - Bernard Tuy
- Is today a legal association under French Law (1901)
 - Laurent Toutain, President
- For further information: <http://www.g6.asso.fr/>






- Share experience gained from IPv6 experimentations and deployment
- Spread IPv6 information
 - Tutorials and trainings (ISPs, Engineers, netadmins. . .)
 - Online book (in French), "IPv6, Théorie et pratique":
<http://livre.g6.asso.fr/>
- Initiate research activities around IPv6
- Active in RIPE & IETF working groups
- Promotion of IPv6: French Task Force



Hypertext Symbols

Programming
IPv6
Applications

- Several symbols are used in this document:
 - All RFCs and Internet Drafts are hypertext links.
 - Check that there is no more recent version of the document.
 -  is a link to a *Techniques de l'Ingénieur* article on the subject (in French, access may be restricted).
 -  is a link to the online edition of *IPv6, Théorie et Pratique* (in French)
 -  is a link to other information on the web.
- Material concerning IPv6 is taken from the G6 tutorial and copyrighted from G6.



Programming
IPv6
Applications
CC++ API
JAVA API

IPv6 socket API in C, C++



Socket API

Programming
IPv6
Applications
CC++ API
JAVA API

- Socket Unix API has been extended to IPv6
- New protocol and address family PF_INET6 and AF_INET6
- New structures :
 - in6_addr
 - sockaddr_in6
 - sockaddr_storage
- New functions for names to addresses conversion

Reference

RFC 2553 & Posix 1003.1g



Structure for sockets

Programming
IPv6
Applications
CC++ API
JAVA API

Structure in C, C++

```
struct sockaddr_in6 {
    uint8_t      sin6_len;          /* structure length
    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
};
```

- Similar to sockaddr_in for IPv4
- New fields for scope and flow label

`sizeof(sockaddr_in6) > sizeof(sockaddr_in)`

- sockaddr_in6 can not be stored in struct sockaddr
- Programs have to be modified to be AF-independent !



Managing Sockets in C, C++



Managing sockets

- Creation : Same as in IPv4
 - `int s = socket(PF_INET6, SOCK_STREAM, 0);`
- Other functions are not modified
 - `bind`, `connect`, `listen`, `accept`, `send*`, `recv*`, `getpeername`, `getsockname`
- New functions to manage options
 - `getsockopt`, `setsockopt`



Sockets and address families

Programming
IPv6
Applications
CC++ API
JAVA API

2 options for applications :

- Only use PF_INET6 socket
 - On a IPv4 networks, use IPv4-mapped IPv6 addresses
 - **Problem: when IPv6 stack is not available ...**
- Use one PF_INET socket and one PF_INET6 socket
 - Client knows which socket to open with getaddrinfo
 - Server should wait for packets on both sockets

Examples found with netstat -taun (MacOSX)

```
Proto Rec Send Local Foreign State
tcp46 0 0 *.80 *.* LISTEN ← Apache server uses first option
...
tcp4 0 0 *.22 *.* LISTEN ← SSH server uses second option
tcp6 0 0 *.22 *.* LISTEN ←
```



Example : Client connection

Programming
IPv6
Applications
CC++ API
JAVA API

```
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netdb.h>
int open_conn(const char *host) {
    int sock = -1, ecode;
    struct addrinfo *res, *r, hints = {
        0, PF_UNSPEC, SOCK_STREAM, 0};

    if ((ecode = getaddrinfo(host, "daytime", &hints, &res)))
        errx(1, "getaddrinfo: %s", gai_strerror(ecode));
    for (r = res; r && sock < 0; r = res->ai_next)
        if ((sock = socket(res->ai_family, res->ai_socktype, res->ai_protocol)) < 0 ||
            connect(sock, res->ai_addr, res->ai_addrlen))
            sock = -1;
    freeaddrinfo(res);
    return sock;
}
```



Example : Server socket

Programming
IPv6
Applications
CC++ API
JAVA API

```
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netdb.h>
int open_serv(const char *serv) {
    int sock, ecode;
    struct addrinfo *res, hints = {
        AI_PASSIVE, PF_UNSPEC, SOCK_STREAM, 0};

    if ((ecode = getaddrinfo(NULL, serv, &hints, &res))
        errx(1, "getaddrinfo: %s", gai_strerror(ecode));
    if ((sock = socket(res->ai_family, res->ai_socktype, res->ai_protocol)) < 0) ||
        bind(sock, res->ai_addr, res->ai_addrlen) ||
        listen(sock, 1))
        err(1, "socket");
    freeaddrinfo(res);
    return sock;
}
```



Example : Server connection

Programming
IPv6
Applications
CC++ API
JAVA API

```
main() {
    int sock = open_serv("1000");

    for(;;) {
        struct sockaddr_storage from;
        int s, len = sizeof from;
        char name[NI_MAXHOST];

        if ((s = accept (sock, (struct sockaddr*)&from, &len) < 0)
            err(1, "accept");

        if (getnameinfo((struct sockaddr*)&from, &len, name,
            sizeof name, NULL, 0, NI_NUMERICHOST))
            name[0] = 0;
        printf("connexion %s\n", name);
        /* utiliser socket s ? */
        close (s);
    }
}
```



Rules to anticipate integration of IPv6 protocol



Generic structure for sockets

- Programs should use struct `sockaddr_storage` to be AF-independent
- Cast depending of AF when needed

Socket containers

```
struct sockaddr_storage ss;  
foo((struct sockaddr *)&ss);    // AF independent function  
  
void foo(struct sockaddr *s) {  
    // If we need IPv4 socket  
    struct sockaddr_in *sin = (struct sockaddr_in *) s;  
    // If we need IPv6 socket  
    struct sockaddr_in6 *sin6 = (struct sockaddr_in6 *) s;  
}
```




Address manipulation : getaddrinfo()

Programming
IPv6
Applications
CC++ API
JAVA API

getaddrinfo() Prototype

```
int getaddrinfo(const char *nodename,  
               const char *servname,  
               const struct addrinfo *hints,  
               struct addrinfo **res);
```

- Generic function for name resolution, AF-independent
- Replace function gethostbyname
- servname: String for protocol name ("http") or port number ("80")
- hints: Refine request (IPv4 only, IPv6 only, IPv4/IPv6)
- **May return more than one result !**



Address manipulation : getnameinfo()

Programming
IPv6
Applications
CC++ API
JAVA API

getnameinfo() Prototype

```
int getnameinfo(const struct sockaddr *sa,  
               socklen_t salen,  
               char *host,  
               socklen_t hostlen,  
               char *serv, socklen_t servlen,  
               int flags);
```

- Generic function for reverse resolution, AF-independent
- Replace function gethostbyaddr



Macros to test nature of address:

- `IN6_IS_ADDR_UNSPECIFIED (struct in6_addr *)`;
- `IN6_IS_ADDR_LOOPBACK (struct in6_addr *)`;
- `IN6_IS_ADDR_MULTICAST (struct in6_addr *)`;
- `IN6_IS_ADDR_LINKLOCAL (struct in6_addr *)`;

Macros to test address equality :

- `IN6_ARE_ADDR_EQUAL (struct in6_addr *, struct in6_addr *)`;



Migrate existing applications



Porting applications to IPv6 (in a nutshell)

Programming
IPv6
Applications
CC++ API
JAVA API

1: Replace IPv4-only structures and functions with AF-independent version

Generic Structure & Functions

```
hostent → addrinfo  
sockaddr_in → sockaddr_storage  
gethostbyname → getaddrinfo  
gethostbyaddr → getnameinfo
```

2: Look for particular usage of IP address structure `in_addr`

- Applications sometimes use IP addresses as host identifier
- This should be made AF-independent



Porting applications to IPv6 (in a nutshell)

Programming
IPv6
Applications
CC++ API
JAVA API

3: Choose a strategy when opening socket (one or two sockets ?)

4: Consider one host may have more than one address !

- With `getaddrinfo` you may have one IPv4 and several IPv6 addresses for one host
- To be also considered when using address as host identifier

5: Beware of textual representation of IP addresses

Beware

```
http://[2001:660:7301:1::1]  
scp foo.bar [2001:660:7301:1::1]:/tmp
```



IPv6 JAVA API



IPv6 Support in Java

- Java support IPv6 since JDK 1.2, extended with JDK 1.4
- Extension have been made for class InetAddress
- Inheritance and polymorphism ensures relative transparency for version of manipulated addresses



Inet6Address

Programming
IPv6
Applications
CC++ API
JAVA API

New subclass of InetAddress (with Inet4Address)

- Class for instantiate IPv6 addresses
- Methods for checking address scope :
 - isIPv4CompatibleAddress (for IPv4-mapped addresses)
 - isLinkLocalAddress
 - isMulticastAddress



InetAddress

Programming
IPv6
Applications
CC++ API
JAVA API

InetAddress objects may be either IPv4 or IPv6 address
InetAddress class extended for DNS resolution

- Method getByName returns only IPv4 name resolution
- New method getAllByName returns all possible name resolutions (IPv4 and IPv6)
- Reverse resolution unchanged

Changes for IPv6 support

Name resolution using getByName should be changed to use getAllByName and uses the returned array of addresses



- Socket API is based on super-class InetAddress → no major change
- By choosing binding address, change protocol enabled for socket
 - IPv4 binding address → Socket listening for IPv4
 - IPv6 binding address → Socket listening for IPv4 and IPv6

Consequences

- Integration of IPv6 is harmless for IPv4 operations
- IPv6 will be used when correspondent address is IPv6