Research and Implementation of Application Server Compatible with IPv4 and IPv6

Qi Xiong and Yucai He

School of Computer Science and Technology
Hunan University of Arts and Science, Changde, PRC
abcxq@21cn.com

Abstract. Nowadays, most networks are constructed based on IPv4. But with the IP addresses are becoming less and less, networks based on IPv6 are becoming more and more. So, some application programs based on IPv4 are needed to be improved to suit IPv4/IPv6 environment. The difference between IPv4 and IPv6 socket is first compared in the paper. Then a FTP program based on IPv4/IPv6 is designed by dual stack software. System test shows that FTP server can support IPv4/IPv6, and the basic functions such as login, authentication of passwords, list of sending files, download files, upload file and delete files were implemented.

Keywords: IPv6, FTP server, Socket, Dual stack.

1 Introduction

The biggest issue of IPv4, the internet technology which is used wildly nowadays, is that the resources of network IP addresses are in short supply day by day. With the number of internet users increase rapidly, the IP address of IPv4 will soon be used up. IPv6 is an next generation IP protocol which adopts 128-bit address that differs to IPv4, which adopts 32-bit address, and that means infinite address space. In the future IPv4 must be displaced by IPv6. But it is impossible to upgrade the entire networks from IPv4 to IPv6 in a short time. There will be a period when the two version of the protocol co-exist. It is very important to interconnect systems of supporting different versions of IP.

2 Comparison of IPv4 with IPv6

From the literature[1], we can see there are many conspicuous differences between socket address structure of IPv4 and IPv6. But two general function named getaddrinfo() and getnameinfo() was introduced into IPv6. These functions are protocol-independent. Function getaddrinfo() translates the name of a service location or a service name to a set of socket addresses. It is described as follows:

getaddrinfo (in const char *nodename, in const char *servname, in const struct addrinfo *hints, out struct addrinfo **res);
The first parameter of getaddrinfo() is host name or IP address which is denoted by IPv4 or IPv6. The second one is services name which is indicated by port NO or existent services name, such as ftp, http, etc. The third parameter is a pointer to addrinfo structure which is filled some return information. The structure addrinfo is described as follows:

```c
struct addrinfo {
    int     ai_flags;          /* address information */
    int     ai_family;         /* AF_xxx */
    int     ai_socktype;       /* type of socket */
    int     ai_protocol;       /* 0 or IPPROTO_xxx for TCP or UDP */
    socklen_t ai_addrlen;     /* length of address */
    char   *ai_canonname;    /* canonical name for nodename */
    struct sockaddr  *ai_addr;  /* binary address for socket address structure */
    struct addrinfo  *ai_next;  /* next structure in linked list */
};
```

Another socket function getnameinfo() is used to translate address to name. Initialized socket structure will sent to it and name or services corresponding to address or port will be returned.

3 Making Application Compatible with IPv4 and IPv6

3.1 Special Address in IPv6

From those principles showed above, according to the compatible support of socket API, we try to modify IPv6 FTP server, which was designed in literature [2], to make it compatible with IPv4. We need two special IPv6 address to do this.

**IPv4-mapped IPv6 Address.** If an node, which have only IPv4 address, uses the IPv4-mapped IPv6 address to access IPv6 server. The IPv4 address is encoded into the low-order 32 bits of the IPv6 address, and the high-order 96 bits hold the fixed prefix 0:0:0:0:0:FFFF. These addresses can be generated automatically by the getaddrinfo() function.

**IPv6 Wildcard Address.** The source IP address of UDP packets and TCP connections will be assigned by bind() function. In this paper, we use wildcard address to make the kernel to select the source address for the server. With IPv4, we use the symbolic constant INADDR_ANY in the bind() function. With IPv6, we use a global structure variable named "in6addr_any", which is assigned by system and initialized as a constant. The extern declaration for this variable is defined in <netinet/in.h>:extern const struct in6_addr in6addr_any.