文章编号: 1673-1549(2010)02-0171-04

与协议无关的 Winsock API编程研究

蓝集明[1], 张海燕[2], 谭功全[3]

(1.四川理工学院计算机学院, 四川 自贡 643000; 2.四川理工学院理学院, 四川 自贡 643000;
3.四川理工学院自动化与电子信息学院, 四川 自贡 643000)

摘 要: Internet正悄然无声地发生着一项重大的技术革命,这就是 IPv4向 IPv6的逐步演进。在 IPv4/IPv6过渡时期,如何开发出能够适应 IPv4/IPv6过渡环境的网络应用程序,这是目前所面临的一个重要问题。本文在深刻理解纯 IPv4网络编程和纯 IPv6网络编程的基础上,通过比较两者的异同点,抛弃它们的差异性,吸收它们的共同点,实现了一种与协议无关的 Winsock API编程方法,并在自行搭建的一个IPv4/IPv6过渡实验平台上,成功开发了一个与协议无关的网络聊天程序,具有很强的实际应用价值。

关键词: 协议无关; Winsock; IPv6;网络编程
中图分类号: TP393.09 文献标识码: A

引 言

IPv6必将取代 IPv4,这是 Internet未来发展的必然。但是,目前的 Internet是基于 IPv4协议的,互联网中存在大量的 IPv4主机及各种 IPv4网络设备,要想一蹴而就地完成从 IPv4到 IPv6的过渡,是不现实的。为此, IPv6也必将和 IPv4在一段较长的时期内共存交互、和睦相处[1-2]。如何从 IPv4平滑、无缝、安全地向 IPv6过渡,又如何开发能够适应这种过渡环境(同时支持 IPv4和 IPv6)的网络应用程序[3-4],这些都是 IPv4/IPv6过渡时期所面临的重要问题。

本文在笔者从事 IPv6网络编程的过程中,总结了一些实际开发的编程经验,利用 Winsock所提供的 API函数,实现了与协议无关的网络编程,并附了一个简单的开发实例。

1 与协议无关的编程接口

众所周知, IPv4地址采用的是 32比特编址,书写格式常用点分十进制数来表示,而 IPv6地址采用的是 128比特编址,书写格式常用冒号十六进制数来表示,这就决定了 IPv4与 IPv6的 socket编程接口必然会存在一些差异。比如他们在地址族、地址结构、地址转换函数、域名解析函数等方面都有很大的不同,这里我们就不再累述。本文主要关心的是如何抛开两者的差异性,找到两者的协议无关性,编写出两者兼容的网络程序[5-6]。

1.1 与协议无关的结构

1.1.1 sockaddr_storage结构

```
typedef struct sockaddr_storage {
short ss_family; char __ss_pad1[_SS_PAD1SIZE]; __int64 __ss_align;
char __ss_pad2[_SS_PAD2SIZE];
} SOCKADDR_STORAGE, * PSOCKADDR_STORAGE;
```

由于用于 IPv4的 sockaddr_in和用于 IPv6的 sockaddr_in6具有不同的结构和长度,所以不能用 struct sockaddr来为 sockaddr_in和 sockaddr_in6静态分配存储空间,而要用 sockaddr_storage结构来实现。该结构定义了足够大的地址空间,用来容纳相应的地址信息,而且它和 struct sockaddr结构是同构的[7]。在实际编程时我们只需要为它的成员 ss_family指定具体的地址族取值便可根据此值确定要处理的地址结构类型,既适用于 IPv4也适用于 IPv6。

收稿日期: 2009-11-28
基金项目: 人工智能四川省(高校)重点实验室科研基金项目(2008RK012)
作者简介: 蓝集明(1973-),男,四川自贡人,讲师,硕士,主要从事计算机网络技术方面的研究。

1.1.2 addrinfo结构

```
struct addrinfo {
int ai_flags; /* 地址信息标志* / int ai_family; /
* 地址族* / int ai_socktype; /* Socket类型* / int ai_
protocol; /* 协议类型* / size_t ai_addrlen; /* socket
接口地址长度* / char* ai_canonname; /* 主机名* /
struct sockaddr* ai_addr; /* socket接口地址* / struct
addrinfo* ai_next; /* 下一个地址信息的指针* /
};
```

该结构用来容纳主机地址信息,在下面将要讲到的getaddrinfo函数中使用。特别是该结构中的 struct sockaddr* 指针,既可以指向 IPv4也可以指向 IPv6的 sockaddr地址。这就实现了对 IPv4和 IPv6的 sockaddr操作上的统一。

1.2 与协议无关的函数

通过对 IPv4和 IPv6两种网络编程的仔细比较,努力寻找它们的共同点,在支持 IPv4网络编程的函数中找到了一些可以在与协议无关的网络编程中继续使用的函数,再加上一些自定义的函数,便构成了一套与协议无关的函数集,这个函数集包括下面三部分的内容:

1.2.1 沿用 IPv4中的部分函数

在 IPv4网络编程中有不少处理和 IPv6是相同的,这些处理所对应的函数在 IPv6中是可以沿用的。因此,这些函数都是与协议无关的函数。它们包括 socket、bind、listen、accept、connect、send、sendto、recv、recvfrom、close、htons/htonl、ntohs/ntohl等。

1.2.2 IPv6中新增的部分函数

(1) getaddrinfo函数:原型是 intWSAAPI getaddrinfo(__in const char* nodename, __in const char* servname, __in const struct addrinfo* hints, __out struct addrinfo** res);该函数实现对主机名或 IP地址串的解析。其中参数 hints十分重要,相当于一个过滤器,只有符合 hints所指向的结构体的内容才会返回到 res指针所指向的链表中。关于 hints需要注意以下几点:①若 hints取 NULL,则 getaddrinfo函数会默认 addrinfo结构体中的 ai_flag、ai_socktype、ai_protocol都取 0,ai_family取 AF_UNSPEC;②用在服务器端时,ai_flags通常设为 AI_PASSIVE,用于 bind,而 nodename通常会设为 NULL,返回通配地址“::”;③用在客户端时,ai_flags不能设为 AI_PASSIVE,而 nodename和 servname也不能设为 NULL。

(2) getnameinfo函数:原型是 intWSAAPI getnameinfo(__in const struct sockaddr FAR* sa, __in socklen_t salen, __out char FAR* host, __in DWORD hostlen, __out char FAR* serv, __in DWORD servlen, __in int flags);该函数实现对 sockaddr地址结构的解析,得到相应的主机名和服务名。在此不再详述。

(3) freeaddrinfo函数:原型是 void freeaddrinfo(__in struct addrinfo* ai);该函数用来释放参数 ai所指向的 addrinfo结构体链表。该链表是在 getaddrinfo函数中动态分配的。

(4) inet_pton和 inet_ntop函数:需要说明的是,这两个函数要求客户端至少要 Windows Vista版本,服务器端至少要 Windows Server 2008版本。

上面提到的这几个函数不仅适合 IPv6网络编程也兼容 IPv4,因此,这些函数也是与协议无关的函数。

1.2.3 一些自定义的函数

除了以上这些函数以外,在与协议无关的网络编程中还会遇到一些处理不能利用现成的 IPv4函数和 IPv6函数来实现,这就需要程序员自行设计实现。比如:在 Windows XP和 VC 6.0环境中没有支持 IPv6的地址转换函数,前面提到的 inet_pton和 inet_ntop函数在此环境中不能使用。在本文中,它们的功能是由自己编写的一个函数 ntoh IPv6和前面提到的 getaddrinfo函数来实现的。

2 一个与协议无关的网络聊天程序实例

本实例采用了与协议无关的 Socket API编程方法,实现了一个能在双栈主机上运行的网络聊天程序,兼容 IPv4和 IPv6网络。

2.1 开发平台

本实例在选择开发平台时尽量考虑了通用性和一般性问题,实验条件不苛刻,在一般的网络实验室都可以顺利实现。

2.1.1 硬件平台

本实例的硬件平台是在四川理工学院网络实验室搭建的,网络拓扑图见图 1所示。

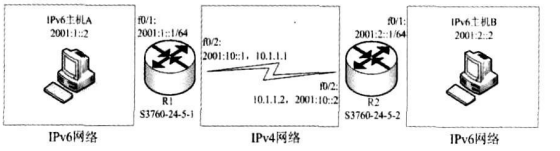

图 1 网络拓扑图

其中,R1和 R2由两个锐捷三层交换机 RG-S3760-24来实现的,IPv6主机由学生用机配置而成。在 R1和 R2两边分别是一个 IPv6网络,在 R1和 R2之间是一个模拟的 IPv4网络,通过建立隧道 Tunnel0实现两个 IPv6网络之间的通信。在 IPv4/IPv6过渡的初期这个实验很具有代表性。

2.1.2 软件平台

W indows XP SP3系统, VC6.0+ SP6+ PSDK2003。

2.2 部分重要代码及相关说明

```
/* 解析本机地址信息, 在双栈主机上为不同类型的地址建立不同的套接字, 并绑定。本程序使用了一个套接字数组来和获取到的地址结构体链表对应。* /
BOOL CChatDlg: InitSocket( )
{……
short * IPv6Addr; struct addrinfo hints, * res= NULL; memset(&hints, 0, sizeof(hints)); hints ai_family = PF_UNSPEC; hints ai_flags = AI_PASSIVE; /* 服务器端, 用于 bind* / hints ai_socktype = SOCK_DGRAM;
err= getaddrinfo(NULL, PORT, &hints, &res); /* 主机名设 NULL, 返回通配地址 "::", 可得到本机中所有 IPv4和 IPv6的地址结构体信息, 这些地址结构体形成一个链表, 由 res指向* /
……
for( m_resPtr= res; res! = NULL; res= res- > ai_next, i+ + )
{m_socket[ i] = WSASocket( res- > ai_family, res- > ai_socktype, res- > ai_protocol, NULL, 0, 0);
if ( INVALID_SOCKET = = m_socket[ i])
{ sprintf( strTempt, "UdpServer socket( ) error with% d% s\n", WSAGetLastError( ), DecodeError(WSAGetLastError( )));
MessageBox( strTempt); continue;
}
else{……}
if (bind(m_socket[ i], res- > ai_addr, res- > ai_addrlen) = = SOCKET_ERROR) { …… }
switch ( res- > ai_family) {
case AF_INET: ……
case AF_INET6:
m_strPrompt+ = "AF_INET6\r\n";
m_strPrompt+ = " Local Server IPv6 Address: ";
IPv6Addr= ntohIPv6( ( struct sockaddr_in6* ) ( res- > ai_addr)); /* 自己编写的一个函数, 实现 IPv6网络字节序到主机字节序的转换* /
for ( j= 0; j< = 6; j+ + )
{ sprintf ( strTempt, "% 02x ", IPv6Addr[ j]); m_strPrompt+ = strTempt;
} /* 实现主机字节序到地址串的转换* /
sprintf( strTempt, "% 02x \n", IPv6Addr[ j]); m_strPrompt+ = strTempt;
break;
case AF_NETBIOS: ……
default: ……
}
SetDlgItemText( IDC_EDIT_PROMPT, m_strPrompt);
if( SOCKET_ERROR = = WSAAsyncSelect(m_socket[ i], m_hWnd, UM_SOCK, FD_READ)) /* 为相应的套接字请求基于 windows消息的网络事件通知, 并自动将该套接字设置为非阻塞模式, 实现了异步套接字编程* /{……}
}
return TRUE;
}
/* 下面是用户自定义消息 UM_SOCK的响应函数, 实现接收来自不同类型的地址上的数据* /
void CChatDlg: OnSock ( WPARAM wParam, LPARAM lParam)
{……
for ( int i= 0; wParam! = m_socket[ i]; i+ + )
res= res- > ai_next; /* 寻找 wParam套接字所对应 addrinfo结构体信息, 这是自行设计的一种处理技巧* /
switch( LOWORD( lParam))
{ case FD_READ: ……
if( SOCKET_ERROR = = WSARecvFrom ( wParam, &wsabuf, 1, &dwRead, &dwFlag, res- > ai_addr, ( int* ) &( res- > ai_addrlen), NULL, NULL))
{…… return;
}
if( res- > ai_family= = AF_INET) /* 实现对不同地址类型的处理* /
{ str Format( "% s说: % s", inet_ntoa( ( ( struct sockaddr_in* ) ( res- > ai_addr)) - > sin_addr), wsabuf buf);
str+ = "\r\n";
}
else if( res- > ai_family= = AF_INET6)
{……
sprintf( buffTempt, "% 02x说: % s\r\n", IPv6Addr[ j], wsabuf buf); str+ = buffTempt;
}
…… break;
}
}
```

2.3 运行结果

经过对以上实例程序的调试,本程序实现了以下功能: 1)在 IPv4-Only主机之间可以正常的聊天通信; 2)在 IPv6-Only主机之间可以正常的聊天通信; 3)在 IPv6/IPv4主机之间可以正常的聊天通信; 4)在 IPv4-Only主机和 IPv6/IPv4主机之间可以正常的聊天通信; 5)在 IPv6-Only主机和 IPv6/IPv4主机之间可以正常的聊天通信。

3 结束语

由于 IPv4和 IPv6在许多方面都存在差异,要编写一个能同时支持 IPv4和 IPv6的应用程序并不容易,稍有不慎就可能写出与协议相关的代码。本文旨在寻找一种完全独立于 IPv4和 IPv6协议的网络编程方法,而不是分别将处理 IPv4和 IPv6的方法在一个应用程序中进行简单地叠加。本文实例实现了无论是在 IPv4还是在 IPv6网络环境中都能自动识别和正确处理网络数据,用户不用关心具体的网络环境,只需要知道对方的 IP地址或域名就可以正常地通信了。

参 考 文 献:

[1] 蓝集明,符长友. IPv6-in-IPv4隧道发现技术研究[J]. 四川理工学院学报:自然科学版, 2009, 22(2): 40-42.

[2] 蓝集明,陈 林. 对 IPSec中 AH和 ESP协议的分析与建议[J]. 计算机技术与发展, 2009, 19(11): 15-17.

[3] Gilligan R, Thomson S, Bound J, et al. IETF RFC 3493: Basic Socket Interface Extensions for IPv6[S/OL]. http://www.ietf.org/rfc/rfc3493.txt?number=3493.

[4] Stevens W, Thomas M, Nordmark E, et al. IETF RFC 3542: Advanced Sockets Application Program Interface (API) for IPv6[S/OL]. http://www.ietf.org/rfc/rfc3542.txt?number=3542.

[5] Shin M-K Ed, Hong Y-G, Hagino J, et al. IETF RFC 4038: Application Aspects of IPv6 Transition[S/OL]. http://www.ietf.org/rfc/rfc4038.txt?number=4038.

[6] 马仝芬,付晓玲. IPv4与 IPv6兼容网络编程模式[J]. 北方工业大学学报, 2005, 17(3): 25-29.

[7] 袁德明. IPv6新型套接字的网络编程剖析[J]. 通信技术, 2007, 40(12): 363-366.

Study on Protocol-independent Winsock Programming

LAN Ji-ming[1], *ZHANG Hai-yan*[2], *TAN Gong-quan*[3]

(1. School of Computer Science, Sichuan University of Science & Engineering, Zigong 643000, China;
2. School of Science, Sichuan University of Science & Engineering, Zigong 643000, China;
3. School of Automation and Electronic Information, Sichuan University of Science & Engineering, Zigong 643000, China)

Abstract There's a technological revolution going on in the Internet, which is the transition from IPv4 to IPv6. How to develop the application software in the IPv4/IPv6 transition period that is able to adapt to the IPv4/IPv6 transition environment is an important problem. On the basis of IPv4-only network programming and IPv6-only network programming, the paper illustrates the differences between the two ways, discards their differences, and absorbs their common. A kind of protocol-independent Winsock programming method is introduced in this paper, an IPv4/IPv6 transition experiment platform is constructed, and a protocol-independent network chat application software is developed, which is of strong practical application value.

Key words protocol-independent; Winsock; IPv6; network programming