第三章地址族与数据序列

本章注重给socket（电话机）分配IP地址和端口号（电话号码）的方法
为了使计算机连接到网络并收发数据，必须向其分配IP地址（IPv4、IPv6）
IPv4：4字节地址族（32位）
IPv6：16字节地址族（128位）

IPv4地址

分为网络地址和主机地址，并且分为A、B、C、D、E等类型

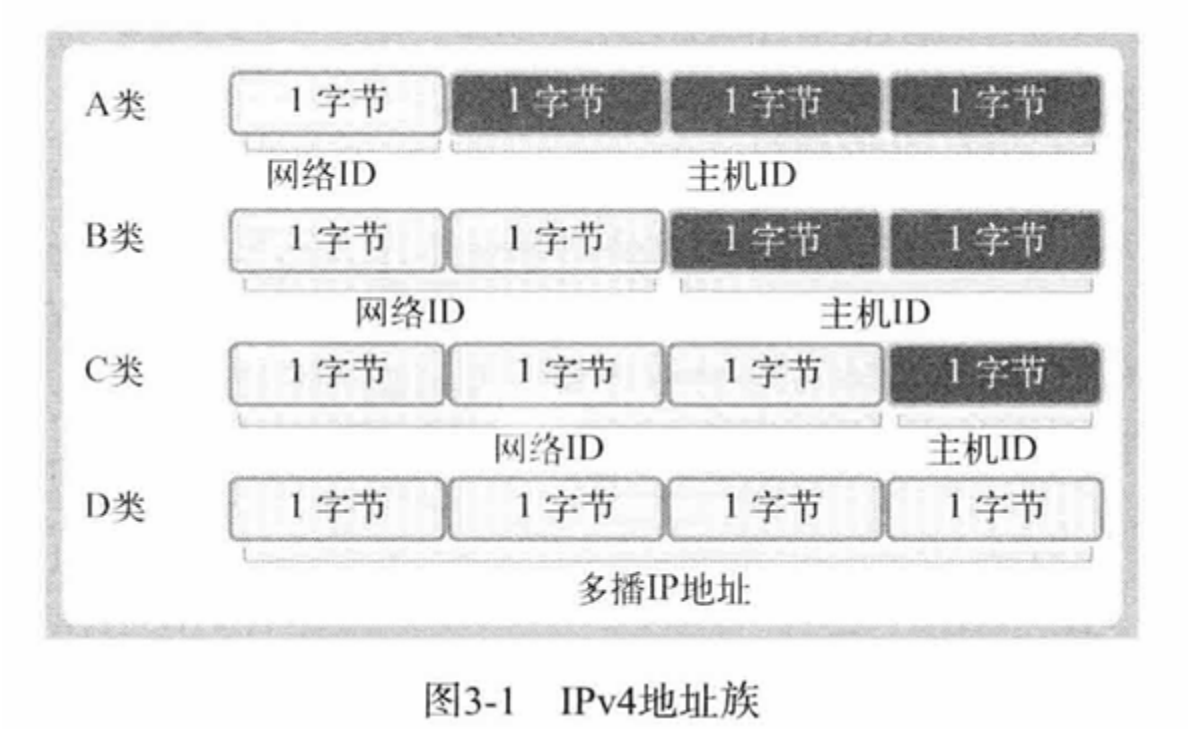

图3-1　IPv4地址族

网络地址是为了区分网络而设置的一部分IP地址，数据传输时，先找到目标IP的网络号（交换机或者路由器），继而交换机/路由器根据主机号确定主机，从而完成数据的传输

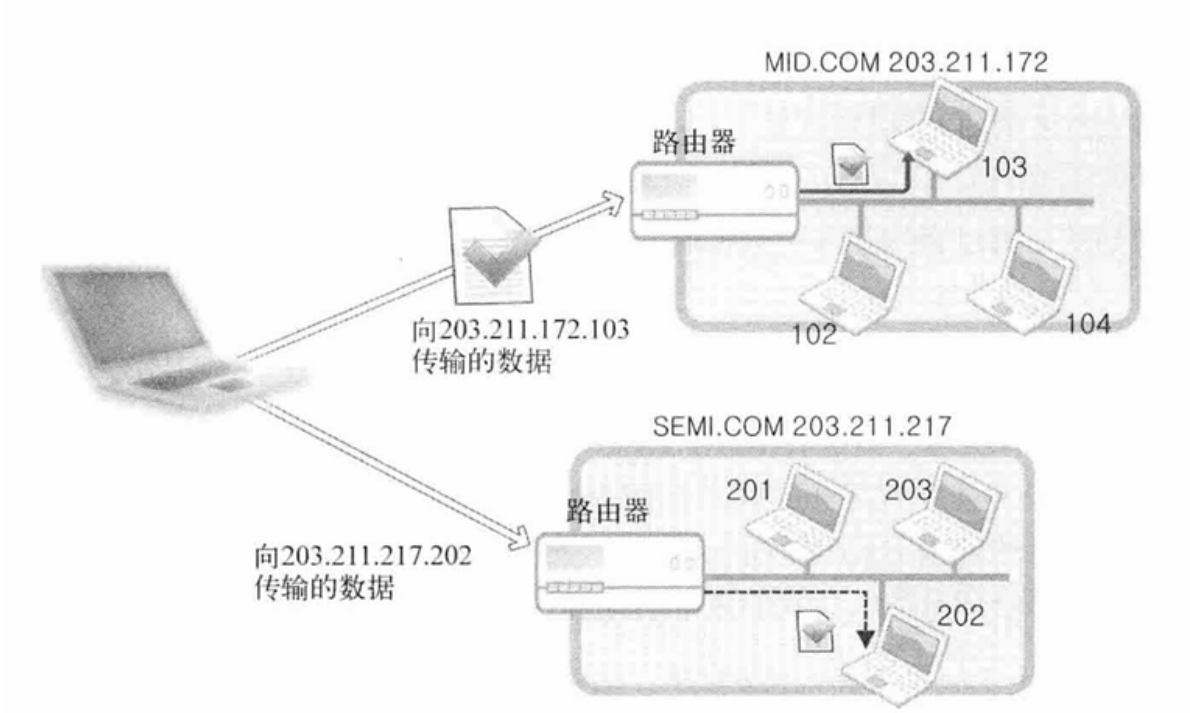

图3-2　基于IP地址的数据传输过程

不同类型地址的首字节范围如下：

A类地址：0~127　B类地址：128~191　C类地址：192~223
A类地址的首位以0开始 B类地址的前2位以10开始　C类地址的前3位以110开始

用于区分套接字的端口号

不同类型的数据需要用多个套接字来进行传输，所以计算机中一般配有NIC（Network Interface Card，网络接口卡）数据传输设备。通过NIC接收的数据内有端口号，操作系统正是参考此端口号把数据传输给相应端口的套接字；另外，端口号由16位构成，可分配的端口号的范围是0-65535，0-1023是知名端口，一般分配给特定应用程序，所以应当分配此范围之外的值。另外，虽然端口号不能重复，但TCP套接字和UDP套接字不会公用端口号，所以允许重复。

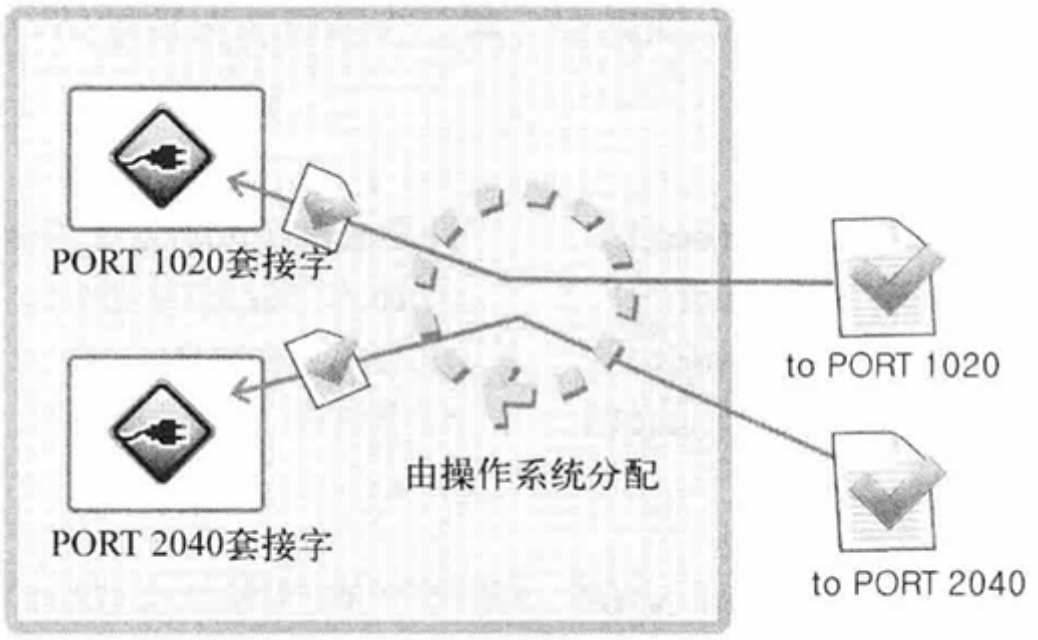

图3-3　数据分配过程

地址信息的表示

应用程序中使用一个结构体sockaddr_in来保存ip地址和端口号

```
struct sockaddr_in{
    sa_family_t         sin_family;   //地址族
    uint16_t            sin_port;     //16位TCP/UDP端口号
    struct in_addr      sin_addr;     //32位IP地址
    char                sin_zero[8]; //不使用
};
```

```
struct in_addr{
    In_addr_t           s_addr;      //32位IPv4地址
};
```

CPU的数据保存方式

小端存储：高字节存放在高位地址
大端存储：高字节存放在低位地址

0x1234（0x12为高字节）（0x34为低字节）
在网络传输数据时约定统一方式，这种约定称为网络字节序——统一为大端序，固如果计算机系统为小端序，那么在传输数据时应转换为大端序排列方式

```
/*转换字节序的函数*/
unsigned short htons(unsigned short);
//把short型数据从主机字节序转化为网络字节序，其中h指的是host，n指的是network
unsigned short ntohs(unsigned short);
//把short型数据从网络字节序转化为主机字节序
unsigned long htonl(unsigned long);
//把long型数据从主机字节序转化为网络字节序
unsigned long ntohl(unsigned long);
//把long型数据从网络字节序转化为主机字节序
```

网络地址的初始化与分配

sockaddr_in中保存地址信息的成员是32位整数型。因此需要将字符串数据转换成32位整数型数据，下面inet_addr函数具有该功能

```
#include <arpa/inet.h>
in_addr_t inet_addr(const char* string);
//成功时返回32位大端序整数型值，失败时返回INADDR_NONE;
//in_addr_t在其内部声明为了32位整数型
//inet_addr函数不仅可以把IP地址转化成32位整数型，而且可以检测无效的IP地址
```

inet_aton函数与inet_addr函数在功能上完全相同，也将字符串形式IP地址转换成32位网络字节序整数并返回。只不过该函数利用了in_addr结构体，且其使用频率更高。

```
#include <arpa/inet.h>
int inet_aton(const char* string, struct in_addr* addr);
//成功时返回1，失败时返回0
```

inet_ntoa函数与**inet_aton**函数相反，此函数可以把网络字节序整数型IP地址转化成我们熟悉的字符串形式。

```
#include <arpa/inet.h>
char* inet_ntoa(struct in_addr addr);
//成功时返回转换的字符串地址值，失败时返回-1
//该函数将通过参数传入的整数型IP地址转换为字符串格式并返回。但调用时需小心，返回值类型为char指针。返回字符串地址意味着字符串已保存到内存空间，但该函数未向程序员要求分配内存，而是在内部申请了内存并保存了字符串。也就是说，调用完该函数后，应立即将字符串信息复制到其他内存空间。因为，若再次调用inet_ntoa函数，则有可能覆盖之前保存的字符串信息。总之，再次调用inet_ntoa函数前返回的字符串地址值是有效的。若需要长期保存，则应将字符串复制到其他内存空间。
```

INADDR_ANY自动获取运行服务端的计算机IP地址，不必亲自输入，服务端优先考虑这种方式。只要端口号一致就行。而客户端除非带有一部分服务器端功能，否则不会采用。

向套接字分配网路地址

当sockadd_in结构体初始化后，接下来就是把初始化的地址信息分配给套接字。bind函数负责这项操作。

```
#include <sys/socket.h>
int bind(int sockfd, struct sockaddr * myaddr, socklen_t addrlen);
//成功时返回0，  失败时返回-1
```

第三章习题

1、IP地址族IPv4和IPv6有何区别？在何种背景下诞生了IPv6？

答：IPv4和IPv6主要是所能代表的IP地址位数的不同，IPv4占据4个字节，总共32位；IPv6占据16个字节，总共128位；随着互联网的普及，IPv4的地址不够分配，从而提出了具有更大范围的IPv6地址，可以解决IP地址资源不够分配的问题。

2、通过IPv4网络ID，主机ID及路由器的关系说明向公司局域网中的计算机传输数据的过程。

答：标准IPv4地址是由一个网络号和一个主机号组成。当需要向位于公司局域网的主机传输数据时，首先会根据网络号找到公司局域网所连接的路由器，路由器再获取目标主机号进行确认，然后将数据传输给对应的主机号。

3、套接字地址分为IP地址和端口号。为什么需要IP地址和端口号？或者说，通过IP可以区分哪些对象？通过端口号可以区分哪些对象？

答：ip地址可以区别不同的计算机，端口号则是用来区分不同的进程。

4、请说明IP地址的分类方法，并据此说出下面这些ip地址的分类。

答：IP地址分类网络号和主机号，根据网络号和主机号所占的字节数将ip地址划分为A、B、C、D、E类型。其中A类地址的取值返回是0-127 B类地址的取值范围位128-191 C类地址的取值范围位192-223。其中 214.121.212.102属于C类地址；120.101.122.89属于A类地址；129.78.102.211属于B类地址

5、计算机通过路由器或交换机连接到互联网。请说出路由器和交换机的作用。

答：路由器用于连接不同的网络（如局域网与互联网），决定数据如何从一个网络到达另一个网络。

交换机用于连接统一局域网内的设备，并根据MAC地址来转发数据，确保局域网内的设备能够高效通信。

6、什么是知名端口？其范围是多少？知名端口中具有代表性的HTTP和FTP端口号各是多少？

答：知名端口号一般分配给特殊程序，其值的范围为0-1023，HTTP的端口号是80、FTP的端口号是21。

7、为什么bind()函数里传的是sockaddr_in结构体指针，而非sockaddr结构体指针？

答：

```
/* Structure describing a generic socket address.  */
struct sockaddr
  {
    __SOCKADDR_COMMON (sa_);  /* Common data: address family and length.  */
    char sa_data[14];   /* Address data.  */
  };
```

```
/* Structure describing an Internet socket address.  */
struct sockaddr_in
  {
    __SOCKADDR_COMMON (sin_);
    in_port_t sin_port;     /* Port number.  */
    struct in_addr sin_addr;    /* Internet address.  */

    /* Pad to size of `struct sockaddr'.  */
    unsigned char sin_zero[sizeof (struct sockaddr)
                           - __SOCKADDR_COMMON_SIZE
                           - sizeof (in_port_t)
                           - sizeof (struct in_addr)];
  };
```

从图中可以看出sockaddr的ip地址和端口号以及协议簇的设置都在一个char类型的数组里，这对开发人员并不友好，非常容易造成混乱；而sockaddr_in结构体为ip地址和端口号以及协议簇都定义了相关的变量，直接将对应的数据填入该结构体即可，并且当需要给bind附上形参时，只需要强制类型转换成sockaddr类型。

8、请解释大端序、小端序、网络字节序，并说明为何需要网络字节序。

答：大端序是高位数据存储在低地址，低位数据在高地址；小端序则相反，高位数据在高地址，低位数据在低地址；操作系统的不同，其字节序也有差异，为了保证数据传输的有效性，提出了网络字节序，通过获取不同操作系统的IP地址和端口，将其一并转化为大端字节序，再进行后续的操作。

9、大端序计算机希望把4字节整数型数据12传递到小端序计算机。请说出数据传输过程中发生的字节序变换过程。

答：12对应的二进制数为 0000 0000 0000 1100，对应的十六进制为0x000C，在大端序计算机中表示为0x000C。在数据传输中，将地址转换成大端的网络字节序，由于本身是大端地址，所以经过转换，网络字节序保持不变。还是0x000C。

10、怎样表示回送地址？其含义是什么？如果向回送地址传输数据将发生什么情况？

答：回送地址是一个本地回环地址，专门用来测试本机网络通信功能的一个特殊地址，其通常表示为：127.0.0.1；数据在传输过程中，数据不会离开本机，只是模拟网络通信，通过本地网络协议栈循环返回。