# 基于UDP协议的套接字编程

### 1、TCP和UDP

1）传输层主要应用的协议模型有两种，一种是TCP协议，另外一种则是UDP协议。流式传输的默认协议是TCP协议，报式传输的默认协议是UDP协议。TCP协议被称为“面向连接的可靠的数据包传输”，而UDP协议被称为“无连接的不可靠报文传递”。

2）与TCP相比，UDP由于无需创建连接，所以开销较小，数据传输速度快，实时性较强。多用于对实时性要求较高的通信场合，如视频会议、电话会议等。但随之也伴随着数据传输不可靠，传输数据的正确率、传输顺序和流量都得不到控制和保证。所以，通常情况下，使用UDP协议进行数据传输，为保证数据的正确性，我们需要在应用层添加辅助校验协议来弥补UDP的不足，以达到数据可靠传输的目的。

3）与TCP类似的，UDP也有可能出现缓冲区被填满后，再接收数据时丢包的现象。由于它没有TCP滑动窗口的机制，通常采用如下两种方法解决：

（1）服务器应用层设计流量控制，控制发送数据速度。

（2）借助setsockopt函数改变接收缓冲区大小。如：

**int n = 220x1024**

**setsockopt(sockfd, SOL\_SOCKET, SO\_RCVBUF, &n, sizeof(n));**

### 2、函数介绍

1）recvfrom 函数：

ssize\_t recvfrom(int sockfd, void \*buf, size\_t len, int flags, struct sockaddr \*src\_addr, socklen\_t \*addrlen);

flags：默认取0；

src\_addr：传出参数，接受发送端的地址信息；

addrlen：sizeof(src\_sddr)。

2）sendto函数：

ssize\_t sendto(int sockfd, const void \*buf, size\_t len, int flags, const struct sockaddr \*dest\_addr, socklen\_t addrlen);

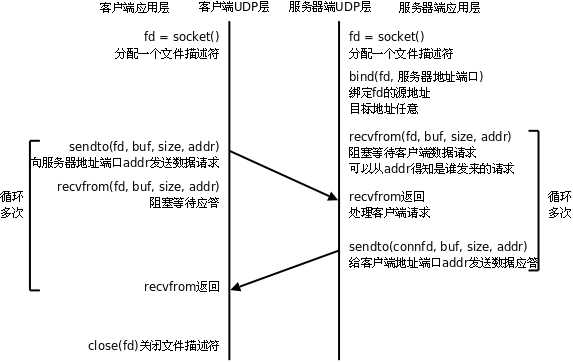
flags：默认取0；

src\_addr：传入参数，目的端的地址信息；

addrlen：sizeof(src\_sddr)。

### 3、C/S模型-UDP

1）基于UDP协议的客户端/服务器程序的一般流程：



2）server：

**#include <string.h>**

**#include <netinet/in.h>**

**#include <stdio.h>**

**#include <unistd.h>**

**#include <strings.h>**

**#include <arpa/inet.h>**

**#include <ctype.h>**

**#define MAXLINE 80**

**#define SERV\_PORT 6666**

**int main(void)**

**{**

**struct sockaddr\_in servaddr, cliaddr;**

**socklen\_t cliaddr\_len;**

**int sockfd;**

**char buf[MAXLINE];**

**char str[INET\_ADDRSTRLEN];**

**int i, n;**

**sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);**

**bzero(&servaddr, sizeof(servaddr));**

**servaddr.sin\_family = AF\_INET;**

**servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);**

**servaddr.sin\_port = htons(SERV\_PORT);**

**bind(sockfd, (struct sockaddr \*)&servaddr, sizeof(servaddr));**

**printf("Accepting connections ...\n");**

**while (1) {**

**cliaddr\_len = sizeof(cliaddr);**

**n = recvfrom(sockfd, buf, MAXLINE,0, (struct sockaddr \*)&cliaddr, &cliaddr\_len);**

**if (n == -1)**

**perror("recvfrom error");**

**printf("received from %s at PORT %d\n",**

**inet\_ntop(AF\_INET, &cliaddr.sin\_addr, str, sizeof(str)),**

**ntohs(cliaddr.sin\_port));**

**for (i = 0; i < n; i++)**

**buf[i] = toupper(buf[i]);**

**n = sendto(sockfd, buf, n, 0, (struct sockaddr \*)&cliaddr, sizeof(cliaddr));**

**if (n == -1)**

**perror("sendto error");**

**}**

**close(sockfd);**

**return 0;**

**}**

3）client：

**#include <stdio.h>**

**#include <string.h>**

**#include <unistd.h>**

**#include <netinet/in.h>**

**#include <arpa/inet.h>**

**#include <strings.h>**

**#include <ctype.h>**

**#define MAXLINE 80**

**#define SERV\_PORT 6666**

**int main(int argc, char \*argv[])**

**{**

**struct sockaddr\_in servaddr;**

**int sockfd, n;**

**char buf[MAXLINE];**

**sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);**

**bzero(&servaddr, sizeof(servaddr));**

**servaddr.sin\_family = AF\_INET;**

**inet\_pton(AF\_INET, "127.0.0.1", &servaddr.sin\_addr);**

**servaddr.sin\_port = htons(SERV\_PORT);**

**while (fgets(buf, MAXLINE, stdin) != NULL) {**

**n = sendto(sockfd, buf, strlen(buf), 0, (struct sockaddr \*)&servaddr, sizeof(servaddr));**

**if (n == -1)**

**perror("sendto error");**

**n = recvfrom(sockfd, buf, MAXLINE, 0, NULL, 0);**

**if (n == -1)**

**perror("recvfrom error");**

**write(STDOUT\_FILENO, buf, n);**

**}**

**close(sockfd);**

**return 0;**

**}**

### 4、广播

1）发送端发送数据时，指定接收端IP地址是广播地址（xxx.xxx.xxx.255），并且指定端口号。接收端需要调用bind()来绑定固定端口。

2）发送端发送数据前需要开放广播权限：

**Int flag = 1;**

**setsockopt(fd, SOL\_SOCKET, SO\_BROADCAST, &flag, sizeof(flag));**

### 5、组播

1）组播组可以是永久的也可以是临时的。组播组地址中，有一部分由官方分配的，称为永久组播组。永久组播组保持不变的是它的ip地址，组中的成员构成可以发生变化。永久组播组中成员的数量都可以是任意的，甚至可以为零。那些没有保留下来供永久组播组使用的ip组播地址，可以被临时组播组利用。

224.0.0.0～224.0.0.255为预留的组播地址（永久组地址），地址224.0.0.0保留不做分配，其它地址供路由协议使用；

224.0.1.0～224.0.1.255是公用组播地址，可以用于Internet；欲使用需申请。

224.0.2.0～238.255.255.255 为用户可用的组播地址（临时组地址），全网范围内有效；

239.0.0.0～239.255.255.255 为本地管理组播地址，仅在特定的本地范围内有效。

2）可使用ip ad命令查看网卡编号；if\_nametoindex 函数可以根据网卡名，获取网卡编号。

3）程序设计要点：

struct ip\_mreqn {

struct in\_addr imr\_multiaddr; /\* 组播地址 \*/

struct in\_addr imr\_address; /\* 本地地址 \*/

int imr\_ifindex; /\* 网卡编号 \*/

};

（1）服务器端：

①设置组播组：

**struct ip\_mreqn group;**

**/\*设置组地址\*/**

**inet\_pton(AF\_INET, GROUP, &group.imr\_multiaddr);**

**/\*本地任意IP\*/**

**inet\_pton(AF\_INET, "0.0.0.0", &group.imr\_address);**

**/\*设置网卡编号\*/**

**group.imr\_ifindex = if\_nametoindex("eth0");**

**/\*设置组播组\*/**

**setsockopt(sockfd, IPPROTO\_IP, IP\_MULTICAST\_IF, &group, sizeof(group));**

②发送数据时指定接收端的IP地址为组地址，并且指定端口号。

（2）客户端：

①加入组播组：

**struct ip\_mreqn group;**

**/\*设置组地址\*/**

**inet\_pton(AF\_INET, GROUP, &group.imr\_multiaddr);**

**/\*本地任意IP\*/**

**inet\_pton(AF\_INET, "0.0.0.0", &group.imr\_address);**

**/\*设置网卡编号\*/**

**group.imr\_ifindex = if\_nametoindex("eth0");**

**/\*设置client 加入多播组 \*/**

**setsockopt(confd, IPPROTO\_IP, IP\_ADD\_MEMBERSHIP, &group, sizeof(group));**

②需要调用bind()来绑定固定端口号。

4）server：

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <sys/types.h>**

**#include <sys/socket.h>**

**#include <string.h>**

**#include <unistd.h>**

**#include <arpa/inet.h>**

**#include <net/if.h>**

**#define SERVER\_PORT 6666**

**#define CLIENT\_PORT 9000**

**#define MAXLINE 1500**

**#define GROUP "239.0.0.2"**

**int main(void)**

**{**

**int sockfd, i ;**

**struct sockaddr\_in serveraddr, clientaddr;**

**char buf[MAXLINE] = "itcast\n";**

**char ipstr[INET\_ADDRSTRLEN]; /\* 16 Bytes \*/**

**socklen\_t clientlen;**

**ssize\_t len;**

**struct ip\_mreqn group;**

**sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);**

**bzero(&serveraddr, sizeof(serveraddr));**

**serveraddr.sin\_family = AF\_INET; /\* IPv4 \*/**

**serveraddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);**

**serveraddr.sin\_port = htons(SERVER\_PORT);**

**bind(sockfd, (struct sockaddr \*)&serveraddr, sizeof(serveraddr));**

**/\*设置组地址\*/**

**inet\_pton(AF\_INET, GROUP, &group.imr\_multiaddr);**

**/\*本地任意IP\*/**

**inet\_pton(AF\_INET, "0.0.0.0", &group.imr\_address);**

**/\* eth0 --> 编号 命令：ip ad \*/**

**group.imr\_ifindex = if\_nametoindex("eth0");**

**setsockopt(sockfd, IPPROTO\_IP, IP\_MULTICAST\_IF, &group, sizeof(group));**

**/\*构造 client 地址 IP+端口 \*/**

**bzero(&clientaddr, sizeof(clientaddr));**

**clientaddr.sin\_family = AF\_INET; /\* IPv4 \*/**

**inet\_pton(AF\_INET, GROUP, &clientaddr.sin\_addr.s\_addr);**

**clientaddr.sin\_port = htons(CLIENT\_PORT);**

**while (1) {**

**//fgets(buf, sizeof(buf), stdin);**

**sendto(sockfd, buf, strlen(buf), 0, (struct sockaddr \*)&clientaddr, sizeof(clientaddr));**

**sleep(1);**

**}**

**close(sockfd);**

**return 0;**

**}**

5）client：

**#include <netinet/in.h>**

**#include <stdio.h>**

**#include <sys/types.h>**

**#include <sys/socket.h>**

**#include <arpa/inet.h>**

**#include <string.h>**

**#include <stdlib.h>**

**#include <sys/stat.h>**

**#include <unistd.h>**

**#include <fcntl.h>**

**#include <net/if.h>**

**#define SERVER\_PORT 6666**

**#define MAXLINE 4096**

**#define CLIENT\_PORT 9000**

**#define GROUP "239.0.0.2"**

**int main(int argc, char \*argv[])**

**{**

**struct sockaddr\_in serveraddr, localaddr;**

**int confd;**

**ssize\_t len;**

**char buf[MAXLINE];**

**struct ip\_mreqn group;**

**confd = socket(AF\_INET, SOCK\_DGRAM, 0);**

**//初始化本地端地址**

**bzero(&localaddr, sizeof(localaddr));**

**localaddr.sin\_family = AF\_INET;**

**inet\_pton(AF\_INET, "0.0.0.0" , &localaddr.sin\_addr.s\_addr);**

**localaddr.sin\_port = htons(CLIENT\_PORT);**

**bind(confd, (struct sockaddr \*)&localaddr, sizeof(localaddr));**

**/\*设置组地址\*/**

**inet\_pton(AF\_INET, GROUP, &group.imr\_multiaddr);**

**/\*本地任意IP\*/**

**inet\_pton(AF\_INET, "0.0.0.0", &group.imr\_address);**

**/\* eth0 --> 编号 命令：ip ad \*/**

**group.imr\_ifindex = if\_nametoindex("eth0");**

**/\*设置client 加入多播组 \*/**

**setsockopt(confd, IPPROTO\_IP, IP\_ADD\_MEMBERSHIP, &group, sizeof(group));**

**while (1) {**

**len = recvfrom(confd, buf, sizeof(buf), 0, NULL, 0);**

**write(STDOUT\_FILENO, buf, len);**

**}**

**close(confd);**

**return 0;**

**}**