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：

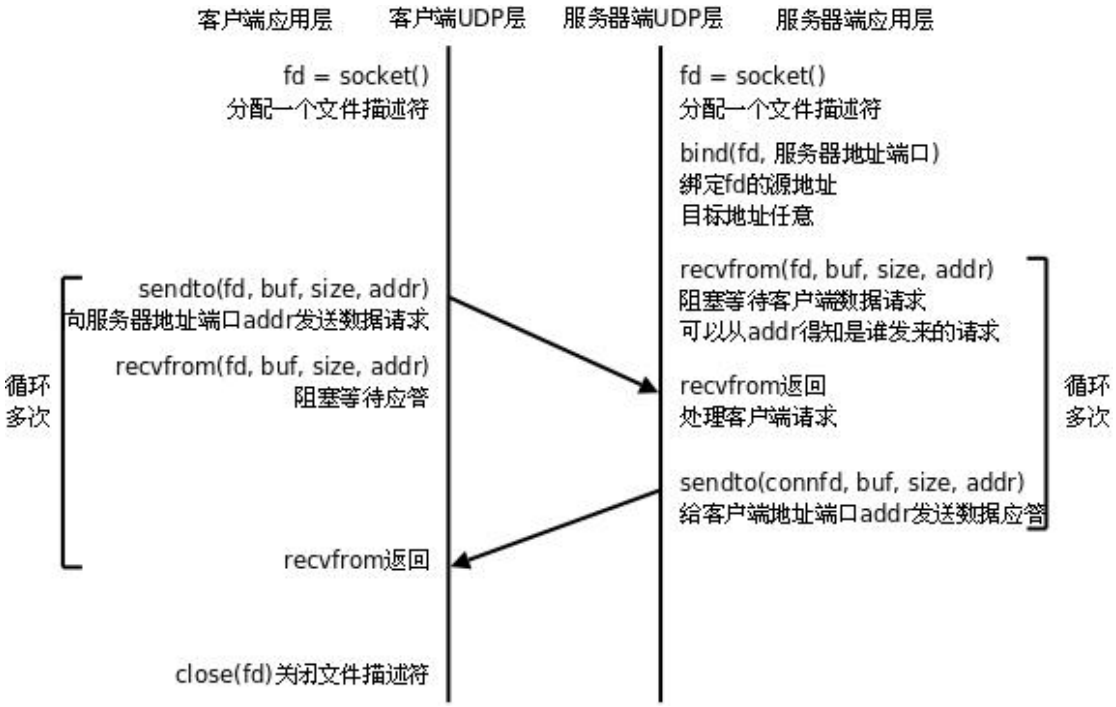
```
int n = 220x1024
setsockopt(sockfd, SOL_SOCKET, SO_RCVBUF, &n, sizeof(n));
```

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;
}
```