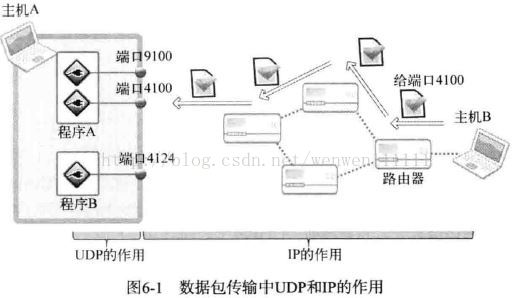
1.UDP套接字原理

  UDP的工作原理可以类比于信件：寄件前先写上寄件人和收件人地址，之后贴上邮票放入邮筒。我们无法确认对方是否收到，邮寄过程中也有可能丢失信件。信件是一种不可靠的传输方式。

  TCP类比于打电话， UDP类比于信件，只是从工作方式进行类比。但是速率上，TCP无法超越UDP，只能接近。应用中可以这样，传递压缩文件需要使用TCP，使数据接收可靠；如果是视频、语音等，丢失一两包数据是没问题的，速度指标更加重要。另外，如果收发的数据量小但是需要频繁连接，UDP也比TCP高效。

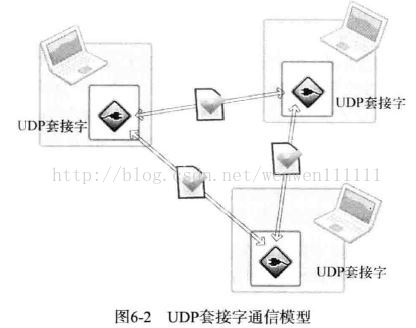
  可靠性方面来说，TCP的确比UDP好，但是UDP的结构比TCP简洁，不会发送类似ACK应答消息，也不会有SEQ序号，性能有时比TCP高出很多。同时区分TCP和UDP的重要标志是流控制：TCP的生命在于流控制。



  图中，IP的作用是让离开主机B的UDP数据包准确传送到主机Ａ，但是最终交给主机Ａ的某一UDP套接字的过程则有UDP完成，它会根据端口号进行区分。

2.基于UDP的服务器端/客户端

  在TCP中，套接字之间是一一对应的，若为10个客户端提供服务，则除了守门的服务器套接字外，还需要10个服务器端套接字。UDP中，服务器客户端都只要一个套接字就OK。



  UDP的数据IO函数：

1. #include <sys/socket.h>
3. ssize\_t sendto(**int** sock, **void** \*buff,**size\_t** nbytes,**int** flags,**struct** sockaddr \*to, socklen\_t addrlen);
5. 成功时返回传输的字节数，失败返回-1；
7. sock：传输数据的UDP套接字文件描述符；
8. buff：保存待传输数据的缓冲地址值；
9. nbytes：待传输的数据长度，字节为单位
10. flags：可选参数，没有则0；
11. to：目标地址信息
12. addrlen：to地址值结构体长度
13. #include <sys/socket.h>
15. ssize\_t recvfrom(**int** sock, **void** \*buff, **size\_t** nbytes, **int** flags, **struct** sockaddr \*from, socklen\_t \*addrlen);
17. 成功时返回接收的字节数，失败时返回-1；
19. sock：用于接收数据的UDP套接字文件描述符
20. buff：保存接收数据的缓冲地址
21. nbytes：可接受的最大字节数，不能超过buff的缓冲大小
22. flags：可选参数，没有则0
23. from：发送端地址信息
24. addrlen：from地址变量大小的地址值

服务器端代码uecho\_server.c

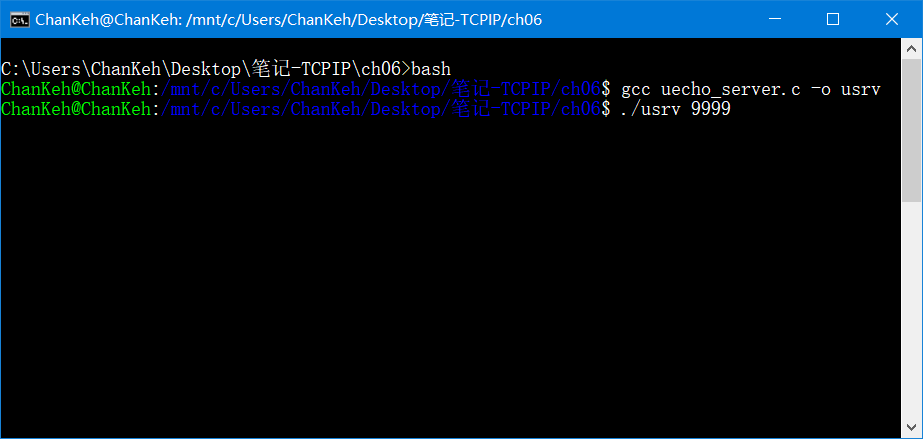
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4. #include <unistd.h>
5. #include <arpa/inet.h>
6. #include <sys/socket.h>
8. #define BUF\_SIZE    30
10. **void** error\_handling(**char** \*message);
12. **int** main(**int** argc, **char** \*argv[]){
13. **int** serv\_sock;
14. **char** message[BUF\_SIZE];
15. **int** str\_len;
16. **struct** sockaddr\_in serv\_adr;
17. **struct** sockaddr\_in clnt\_adr;
18. socklen\_t clnt\_addr\_sz;
20. **if**(argc != 2){
21. printf("Usage : %s <port>\n",argv[0]);
22. exit(1);
23. }
25. serv\_sock = socket(PF\_INET,SOCK\_DGRAM,0);
26. **if**(serv\_sock == -1){
27. error\_handling("socket() error");
28. }
30. memset(&serv\_adr,0,**sizeof**(serv\_adr));
31. serv\_adr.sin\_family=AF\_INET;
32. serv\_adr.sin\_addr.s\_addr=htonl(INADDR\_ANY);
33. serv\_adr.sin\_port=htons(atoi(argv[1]));
35. **if**(bind(serv\_sock,(**struct** sockaddr\*)&serv\_adr,**sizeof**(serv\_adr)) == -1){
36. error\_handling("bind() error");
37. }
39. **while**(1){
41. clnt\_addr\_sz = **sizeof**(clnt\_adr);
42. str\_len = recvfrom(serv\_sock,message,BUF\_SIZE,0,(**struct** sockaddr\*)&clnt\_adr,&clnt\_addr\_sz);
43. sendto(serv\_sock,message,str\_len,0,(**struct** sockaddr\*)&clnt\_adr,clnt\_addr\_sz);
45. }
47. close(serv\_sock);
49. **return** 0;
51. }
53. **void** error\_handling(**char** \*message){
55. fputs(message,stderr);
56. fputs("\n",stderr);
57. exit(1);
58. }

客户端代码uecho\_client.c

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4. #include <unistd.h>
5. #include <arpa/inet.h>
6. #include <sys/socket.h>
8. #define BUF\_SIZE    30
10. **void** error\_handling(**char** \*message);
12. **int** main(**int** argc, **char** \*argv[]){
13. **int** sock;
14. **char** message[BUF\_SIZE];
15. **int** str\_len;
16. socklen\_t adr\_sz;
17. **struct** sockaddr\_in serv\_adr, from\_adr;
19. **if**(argc != 3){
20. printf("Usage : %s <IP> <port>\n",argv[0]);
21. exit(1);
22. }
24. sock = socket(PF\_INET,SOCK\_DGRAM,0);
25. **if**(sock == -1){
26. error\_handling("socket() error");
27. }
29. memset(&serv\_adr,0,**sizeof**(serv\_adr));
30. serv\_adr.sin\_family=AF\_INET;
31. serv\_adr.sin\_addr.s\_addr=inet\_addr(argv[1]);
32. serv\_adr.sin\_port=htons(atoi(argv[2]));
34. **while**(1){
36. fputs("Insert message(q to quit):",stdout);
37. fgets(message,**sizeof**(message),stdin);
38. **if**(!strcmp(message,"q\n") || !strcmp(message,"Q\n"))
39. **break**;
40. sendto(sock,message,strlen(message),0,(**struct** sockaddr \*)&serv\_adr,**sizeof**(serv\_adr));
41. adr\_sz = **sizeof**(from\_adr);
42. str\_len = recvfrom(sock,message,BUF\_SIZE,0,(**struct** sockaddr \*)&from\_adr,&adr\_sz);
43. message[str\_len] = 0;
45. printf("Message from server: %s",message);
46. }
48. close(sock);
50. **return** 0;
52. }
54. **void** error\_handling(**char** \*message){
56. fputs(message,stderr);
57. fputs("\n",stderr);
58. exit(1);
59. }

运行结果：

服务器端：



客户端：

