1. 多播，同时向多个主机传递数据

a. 多播服务器针对特定多播组，只发送一次数据；

b. 虽只发送一次数据，但是该组内的所有客户端都会接收数据；

c. 多播组数可以在IP地址范围内任意增加；

d. 加入特定组即可接收发送该多播组的数据。

多播是基于MBone这个虚拟网络工作的，它是以物理网络为基础，通过软件方法实现的多播通信必备虚拟网络。

**多播组是D类IP地址**，224.0.0.0-239.255.255.255

2. TTL time to live：用整数表示，每经过一个路由器减一，变零时销毁。

1. **int** send\_sock;
2. **int** time\_live = 64;
3. ....
4. setsockopt(send\_sock,IPPROTO\_IP,IP\_MULTICAST\_TTL,(**void** \*)&time\_live,**sizeof**(time\_live))

3. 加入多播组

1. **struct** ip\_mreq join\_adr;
3. join\_adr.imr\_multiaddr.s\_addr = "多播组地址信息"
4. join\_adr.imr\_interface.s\_addr = "加入多播组的主机地址信息"
5. setsockopt(recv\_sock,IPPROTO\_IP,IP\_ADDR\_MEMBERSHIP,(**void** \*)&join\_adr,**sizeof**(join\_adr))

4. 示例，多播发送与接收

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
9. #define TTL 64
11. **void** error\_handling(**char** \*message);
13. **int** main(**int** argc, **char** \*argv[]){
14. **int** send\_sock;
15. **FILE** \*fp;
16. **char** buf[BUF\_SIZE];
17. **struct** sockaddr\_in mul\_adr;
18. **int** time\_live = TTL;
20. **if**(argc != 3){
21. printf("Usage : %s <group><port>\n",argv[0]);
22. exit(1);
23. }
25. send\_sock = socket(PF\_INET,SOCK\_DGRAM,0);
26. **if**(send\_sock == -1){
27. error\_handling("socket() error");
28. }
30. memset(&mul\_adr,0,**sizeof**(mul\_adr));
31. mul\_adr.sin\_family=AF\_INET;
32. mul\_adr.sin\_addr.s\_addr=inet\_addr(argv[1]); //multi ip
33. mul\_adr.sin\_port=htons(atoi(argv[2]));  //port
35. setsockopt(send\_sock,IPPROTO\_IP,IP\_MULTICAST\_TTL,(**void** \*)&time\_live,**sizeof**(time\_live));
36. **if**((fp = fopen("news.txt","r")) == NULL){
37. error\_handling("fopen");
38. }
40. **while**(!feof(fp)){
41. fgets(buf,BUF\_SIZE,fp);
42. sendto(send\_sock,buf,strlen(buf),0,(**struct** sockaddr \*)&mul\_adr,**sizeof**(mul\_adr));
43. sleep(2);
44. }
45. fclose(fp);
46. close(send\_sock);
48. **return** 0;
49. }
51. **void** error\_handling(**char** \*message){
53. fputs(message,stderr);
54. fputs("\n",stderr);
55. exit(1);
56. }

receiver.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** recv\_sock;
14. **int** str\_len;
15. **char** buf[BUF\_SIZE];
16. **struct** sockaddr\_in adr;
17. **struct** ip\_mreq join\_adr;
19. **if**(argc != 3){
20. printf("Usage : %s <group><port>\n",argv[0]);
21. exit(1);
22. }
24. recv\_sock = socket(PF\_INET,SOCK\_DGRAM,0);
25. **if**(recv\_sock == -1){
26. error\_handling("socket() error");
27. }
29. memset(&adr,0,**sizeof**(adr));
30. adr.sin\_family=AF\_INET;
31. adr.sin\_addr.s\_addr=htonl(INADDR\_ANY);  //multi ip
32. adr.sin\_port=htons(atoi(argv[2]));  //port
34. **if**(bind(recv\_sock,(**struct** sockaddr \*)&adr,**sizeof**(adr))==-1){
35. error\_handling("bind error");
36. }
38. join\_adr.imr\_multiaddr.s\_addr = inet\_addr(argv[1]);
39. join\_adr.imr\_interface.s\_addr = htonl(INADDR\_ANY);
41. setsockopt(recv\_sock,IPPROTO\_IP,IP\_ADD\_MEMBERSHIP,(**void** \*)&join\_adr,**sizeof**(join\_adr));
43. **while**(1){
44. str\_len = recvfrom(recv\_sock,buf,BUF\_SIZE-1,0,NULL,0);
45. **if**(str\_len < 0){
46. **break**;
47. }
48. buf[str\_len] = 0;
49. fputs(buf,stdout);
50. }
51. close(recv\_sock);
53. **return** 0;
54. }
56. **void** error\_handling(**char** \*message){
58. fputs(message,stderr);
59. fputs("\n",stderr);
60. exit(1);
61. }