多播

单播用于两个主机之间的端对端通信，广播用于一个主机对整个局域网上所有主机上的数据通信。单播和广播是两个极端，要么对一个主机进行通信，要么对整个局域网上的主机进行通信。实际情况下，经常需要对一组特定的主机进行通信，而不是整个局域网上的所有主机，这就是多播的用途。

IP 多播（也称多址广播或组播）技术，是一种允许一台或多台主机（多播源）发送单一数据包到多台主机（一次的，同时的）的 TCP/IP 网络技术。多播是 IPv6 数据包的 3 种基本目的地址类型之一，多播是一点对多点的通信,　IPv6 没有采用 IPv4 中的组播术语，而是将**广播看成是多播的一个特殊例子**。

**多播作为一点对多点的通信，数据的收发仅仅在同一分组中进行，是节省网络带宽的有效方法之一。**在网络应用中，当需要将一个节点的信号传送到多个节点时，无论是采用重复点对点通信方式，还是采用广播方式，都会严重浪费网络带宽，只有多播才是最好的选择。多播能使一个或多个多播源只把数据包发送给特定的多播组，而只有加入该多播组的主机才能接收到数据包。

IP 多播应用大致可以分为三类:点对多点应用，多点对点应用和多点对多点应用。

1）点对多点应用是指一个发送者，多个接收者的应用形式，这是最常见的多播应用形式。典型的应用包括：媒体广播、媒体推送、信息缓存、事件通知和状态监视等。

2）多点对点应用是指多个发送者，一个接收者的应用形式。通常是双向请求响应应用，任何一端（多点或点）都有可能发起请求。典型应用包括：资源查找、数据收集、网络竞拍、信息询问等。

3）多点对多点应用是指多个发送者和多个接收者的应用形式。通常，每个接收者可以接收多个发送者发送的数据，同时，每个发送者可以把数据发送给多个接收者。典型应用包括：多点会议、资源同步、并行处理、协同处理、远程学习、讨论组、分布式交互模拟（DIS）、多人游戏等。

**多播地址**

IP 多播通信必须依赖于 IP 多播地址，在 IPv4 中它是一个 D 类 IP 地址，范围从 224.0.0.0 到 239.255.255.255，并被划分为**局部链接多播地址、预留多播地址和管理权限多播地址**三类：

1）局部链接多播地址范围在 224.0.0.0~224.0.0.255，这是为路由协议和其它用途保留的地址，路由器并不转发属于此范围的IP包；

2）预留多播地址为 224.0.1.0~238.255.255.255，可用于全球范围（如Internet）或网络协议；

3）管理权限多播地址为 239.0.0.0~239.255.255.255，可供组织内部使用，类似于私有 IP 地址，不能用于 Internet，可限制多播范围。

一些多播组地址被 IANA 确定为知名地址，它们也被当作永久主机组，这和 TCP 及 UDP 中的知名端口相似。同样，这些知名多播地址在 RFC 最新分配数字中列出，注意这些多播地址所代表的组是永久组，而它们的组成员却不是永久的。这些地址如下：

224.0.0.1    所有组播主机

224.0.0.2    所有组播路由器

224.0.0.4    DRMRP 路由器

224.0.0.5    所有 OSPF 的路由器

224.0.0.6    OSPF 指派路由器

224.0.0.9    RPIv2 路由器

224.0.0.10  EIGRP 路由器

224.0.0.13  PIM 路由器

224.0.0.22  IGMPv3

224.0.0.25  RGMP

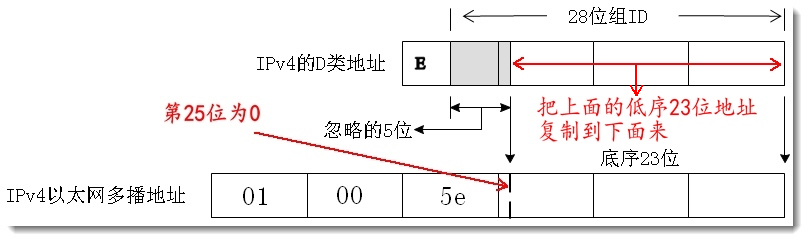
224.0.1.1    NTP 网络时间协议

**多播地址与 MAC 地址的映射**

使用同一个 IP 多播地址接收多播数据包的所有主机构成了一个主机组，也称为**多播组**。一个多播组的成员是随时变动的，一台主机可以随时加入或离开多播组，多播组成员的数目和所在的地理位置也不受限制，一台主机也可以属于几个多播组。

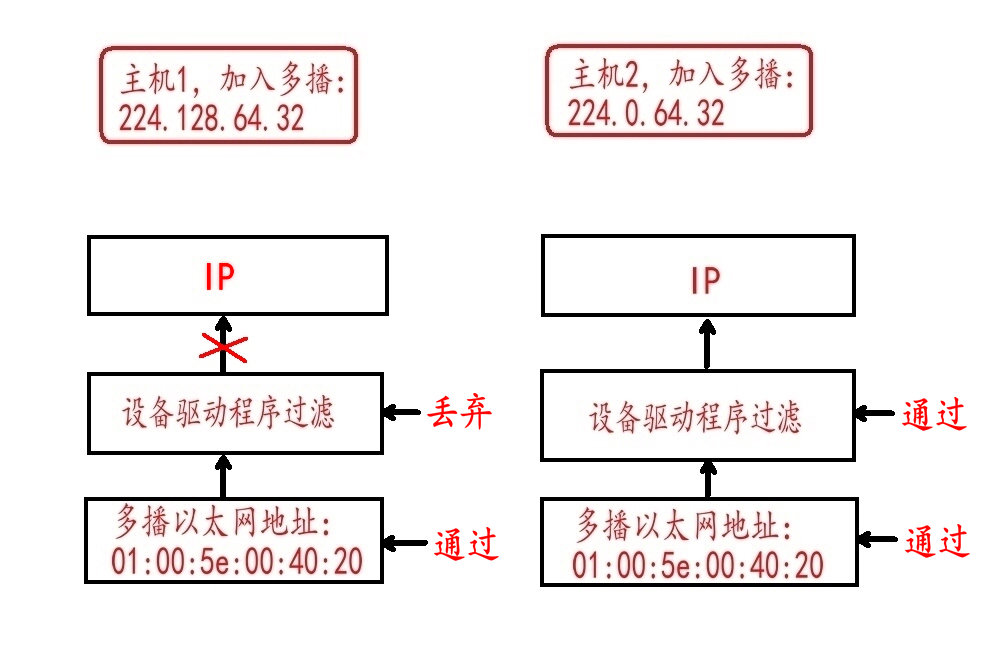
这个我们可以这样理解，**多播地址就类似于 QQ 群号，多播组相当于 QQ 群，一个个的主机就相当于群里面的成员**。

**IPv4 的 D 类地址是多播地址**。IEEE 把一块以太网多播组地址分给 IANA 以支持IP多播。块的地址都以 01:00:5e 开头，第 25 位为 0，低 23 位为 IPv4 多播地址( D类地址 )的低 23 位。IPv4 多播地址与 MAC 地址的映射关系如图所示：



**由于多播地址( D类地址 )中的最高 5bit 在映射过程中被忽略，因此每个以太网多播地址对应的多播组是不唯一的**。32 个不同的多播组号被映射为一个以太网地址。例如，多播地址 224.128.64.32（十六进制 e0.80.40.20）和 224.0.64.32（十六进制 e0.00.40.20）都映射为同一以太网地址 01:00:5e:00:40:20。

**既然地址映射是不唯一的，那么设备驱动程序或 IP 层就必须对数据报进行过滤。**因为网卡可能接收到主机不想接收的多播数据帧，如下图，假如主机 1 加入的多播为 224.128.64.32， 主机 2 加入的多播为 224.0.64.32，我们想给 224.0.64.32 所在的多播组 ( 主机 2 ) 发送信息，数据经过网卡时，224.128.64.32 (主机 1 ) 和 224.0.64.32 (主机 2 ) 所在多播组的网卡都会收到数据，因为它们的 MAC 地址都是 01:00:5e:00:40:20。**这时候，如果网卡不提供足够的多播数据帧过滤功能，设备驱动程序就必须接收所有多播数据帧，然后对它们进行过滤，这个过滤过程是网络驱动或IP层自动完成。**



**Linux多播编程**

**套接口选项**

**int setsockopt( int sockfd, int level,int optname,**

**const void \*optval, socklen\_t optlen );**



成功执行返回0，否则返回-1

**选项 IP\_ADD\_MEMBERSHIP 和 IP\_DROP\_MEMBERSHIP**

加入或者退出一个多播组，通过选项 IP\_ADD\_MEMBERSHIP 和 IP\_DROP\_MEMBERSHIP，对一个结构 struct ip\_mreq 类型的变量进行控制，struct ip\_mreq 原型如下：

struct in\_addr  
{

in\_addr\_t s\_addr;

}  
  
  
struct ip\_mreq            
{

struct in\_addr imn\_multiaddr; // 多播组 IP，类似于 QQ 群号

struct in\_addr imr\_interface;   // 将要添加到多播组的 IP，类似于QQ 成员号

};

**多播只能用 UDP 或原始 IP 实现，不能用 TCP。**

**加入多播实例：**

**[cpp]** [view plain](http://blog.csdn.net/tennysonsky/article/details/44493407) [copy](http://blog.csdn.net/tennysonsky/article/details/44493407)

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <unistd.h>
4. #include <fcntl.h>
5. #include <sys/socket.h>
6. #include <netinet/in.h>
7. #include <arpa/inet.h>
8. #include <string.h>
9. #include <errno.h>
10. #include <sys/types.h>
12. **int** main(**int** argc, **char**\*argv[])
13. {
14. **int** sockfd; // 套接字文件描述符
15. **struct** sockaddr\_in local\_addr; // 本地地址
16. **int** err = -1;
17. **char** group[16] = "224.0.0.88"; // 多播组 IP
19. sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);  //建立套接字
20. **if** (sockfd == -1)
21. {
22. perror("socket()");
23. **return** -1;
24. }
26. // 初始化地址
27. memset(&local\_addr, 0, **sizeof**(local\_addr));
28. local\_addr.sin\_family = AF\_INET;
29. local\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);
30. local\_addr.sin\_port = htons(8000);
32. // 绑定socket
33. err = bind(sockfd,(**struct** sockaddr\*)&local\_addr, **sizeof**(local\_addr));
34. **if**(err < 0)
35. {
36. perror("bind()");
37. **return** -2;
38. }
40. **struct** ip\_mreq mreq; // 多播地址结构体
42. // 加入多播组，相当于创建一个QQ群，某人加入此群
43. mreq.imr\_multiaddr.s\_addr = inet\_addr(group); // 多播地址，类似于 QQ 群号
44. mreq.imr\_interface.s\_addr = htonl(INADDR\_ANY);// 将本机加入多播组，类似于某人加入此群
45. // 加入多播组
46. err = setsockopt(sockfd, IPPROTO\_IP, IP\_ADD\_MEMBERSHIP,&mreq, **sizeof**(mreq));
47. **if** (err < 0)
48. {
49. perror("setsockopt():IP\_ADD\_MEMBERSHIP");
50. **return** -4;
51. }
53. **int** times = 0;
54. **int** addr\_len = 0;
55. **char** buff[256] = {0};
56. **int** n = 0;
58. // 循环接收广播组的消息，5次后退出
59. **for**(times = 0; times<5; times++)
60. {
61. addr\_len = **sizeof**(local\_addr);
62. memset(buff, 0, **sizeof**(buff));
64. // 接收数据
65. n = recvfrom(sockfd, buff, **sizeof**(buff), 0,(**struct** sockaddr\*)&local\_addr, &addr\_len);
66. **if**( n== -1)
67. {
68. perror("recvfrom()");
69. }
71. printf("Recv %dst message from server:%s\n", times, buff);
72. sleep(2);
73. }
75. // 退出广播组
76. err = setsockopt(sockfd, IPPROTO\_IP, IP\_DROP\_MEMBERSHIP,&mreq, **sizeof**(mreq));
78. close(sockfd);
80. **return** 0;
81. }

以上代码编译运行时，可以会出现这样的错误：**No such device**。这主要和网络配置有关，解决方法请点此链接：[http://blog.csdn.net/tennysonsky/article/details/49050579](http://blog.csdn.net/tennysonsky/article/details/49050579" \t "_blank)。

**向多播组发送信息的测试示例：**

**[cpp]** [view plain](http://blog.csdn.net/tennysonsky/article/details/44493407) [copy](http://blog.csdn.net/tennysonsky/article/details/44493407)

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <unistd.h>
4. #include <fcntl.h>
5. #include <sys/socket.h>
6. #include <netinet/in.h>
7. #include <arpa/inet.h>
8. #include <string.h>
9. #include <errno.h>
10. #include <sys/types.h>
12. **int** main(**int** argc, **char**\*argv)
13. {
14. **int** sockfd; // 套接字文件描述符
15. **struct** sockaddr\_in dest\_addr; // 目标ip
16. **char** buf[] = "BROADCAST TEST DATA";
18. sockfd = socket(AF\_INET, SOCK\_DGRAM, 0); // 建立套接字
19. **if** (s == -1)
20. {
21. perror("socket()");
22. **return** -1;
23. }
25. // 初始化目标 ip 信息
26. memset(&dest\_addr, 0, **sizeof**(dest\_addr));
27. dest\_addr.sin\_family = AF\_INET;
28. dest\_addr.sin\_addr.s\_addr = inet\_addr("224.0.0.88"); // 目的地址，为多播地址
29. dest\_addr.sin\_port = htons(8000);   // 多播服务器的端口也是 8000
31. // 向多播地址发送数据
32. **while**(1){
33. **int** n = sendto(sockfd, buf, strlen(buf), 0,(**struct** sockaddr\*)&dest\_addr, **sizeof**(dest\_addr));
34. **if**( n < 0)
35. {
36. perror("sendto()");
37. **return** -2;
38. }
40. sleep(1);
41. }
43. **return** 0；
44. }