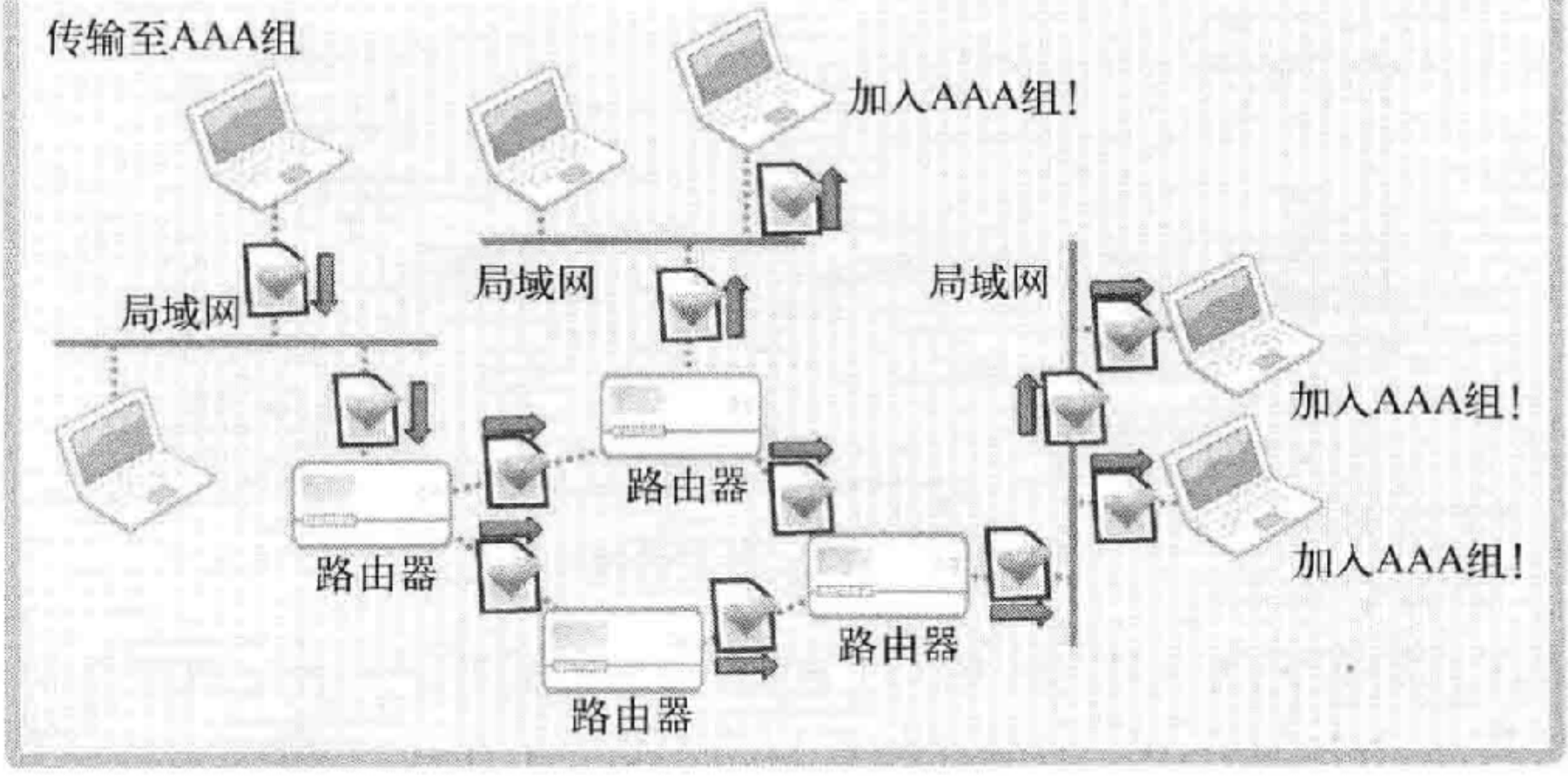
多播与广播，Socket的可选项

多播与广播

- 同时需要向网络中多个主机传播同样信息
- 采用TCP方式?
- 采用UDP方式?

多播及其特点

- 同时向多个主机发送信息
- 多播的服务器针对特定的多播组发送一次数据
- 组内所有客户端都能接受到数据
- 多播的组数只受IP地址的限制
- 加入特定组，即可接收发往该组的多播信息

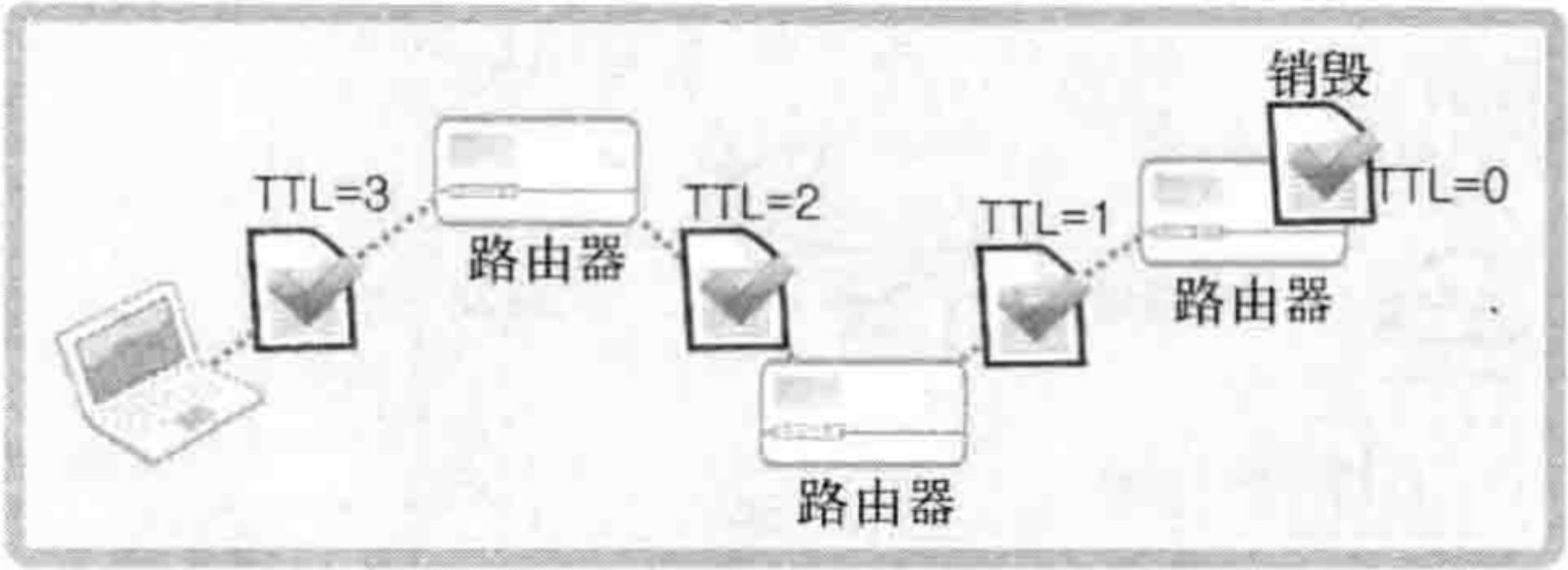

传输至AAA组
加入AAA组!
局域网
局域网
局域网
加入AAA组!
加入AAA组!
路由器
路由器
路由器
路由器

多播对网络负担的影响

- 多播适应的情况
  - 设1000台主机组成了一个子网
  - 若不采用多播，服务器向1000台主机发送消息，需要连接1000次/重复发送1000次
  - 可能造成沿途链路的堵塞
- 多播的开销
  - 网络中数据复制带来的开销
  - 需要路由的支持

多播数据包的TTL

- TTL = Time to Live
- 数据包在网络中存活最长的跳数
- 决定数据包在网络中生存的时间
- 必须设置合适的长度


TTL=3
路由器
TTL=2
TTL=1
销毁
TTL=0
路由器
路由器

TTL设置方法

- 通过设置Socket可选项来实现

```
int send_sock;
int time_live=64;
. . . .
send_sock=socket(PF_INET, SOCK_DGRAM, 0);
setsockopt(send_sock, IPPROTO_IP, IP_MULTICAST_TTL, (void*) &time_live,
sizeof(time_live));
. . . .
```

补充: 套接字的可选项

- 套接字具有不同的特性，可以通过调整可选项来改变，选择合适的通信参数
- 之前提到的例子基本使用默认的可选项参数
- 多层的参数均可以通过可选项来调整

协 议 层	选 项 名	读 取	设 置
SOL_SOCKET	SO_SNDBUF	O	O
	SO_RCVBUF	O	O
	SO_REUSEADDR	O	O
	SO_KEEPALIVE	O	O
	SO_BROADCAST	O	O
	SO_DONTROUTE	O	O
	SO_OOBINLINE	O	O
	SO_ERROR	O	X
	SO_TYPE	O	X
IPPROTO_IP	IP_TOS	O	O
	IP_TTL	O	O
	IP_MULTICAST_TTL	O	O
	IP_MULTICAST_LOOP	O	O
	IP_MULTICAST_IF	O	O
IPPROTO_TCP	TCP_KEEPALIVE	O	O
	TCP_NODELAY	O	O
	TCP_MAXSEG	O	O

getsockopt()

- 用于读取socket options
- getsockopt(

```
int     sock,
int     level,              // 可选项协议层
int     optname,            // 可选项名
void * optval,
socklen_t * optlen);
```

setsockopt()

- 用于设置socket options
- getsockopt(

int sock,

int level, // 可选项协议层

int optname, // 可选项名

void * optval,

socklen_t optlen);

```
7.  int main(int argc, char *argv[])
8.  {
9.      int tcp_sock, udp_sock;
10.     int sock_type;
11.     socklen_t optlen;
12.     int state;
13.
14.     optlen=sizeof(sock_type);
15.     tcp_sock=socket(PF_INET, SOCK_STREAM, 0);
16.     udp_sock=socket(PF_INET, SOCK_DGRAM, 0);
17.     printf("SOCK_STREAM: %d \n", SOCK_STREAM);
18.     printf("SOCK_DGRAM: %d \n", SOCK_DGRAM);
19.
20.     state=getsockopt(tcp_sock, SOL_SOCKET, SO_TYPE, (void*)&sock_type, &optlen);
21.     if(state)
22.         error_handling("getsockopt() error!");
23.     printf("Socket type one: %d \n", sock_type);
24.
25.     state=getsockopt(udp_sock, SOL_SOCKET, SO_TYPE, (void*)&sock_type, &optlen);
26.     if(state)
27.         error_handling("getsockopt() error!");
28.     printf("Socket type two: %d \n", sock_type);
29.     return 0;
30. }
31.
32. void error_handling(char *message)
33. {
34.     fputs(message, stderr);
35.     fputc('\n', stderr);
36.     exit(1);
37. }
```

TTL设置方法

- 通过设置Socket可选项来实现
- 更改选项在IPPROTO_IP协议层

```
int send_sock;
int time_live=64;
. . . .
send_sock=socket(PF_INET, SOCK_DGRAM, 0);
setsockopt(send_sock, IPPROTO_IP, IP_MULTICAST_TTL, (void*) &time_live,
sizeof(time_live));
. . . .
```

设置加入多播组的信息

- 通过设置Socket可选项设置加入多播组
- IP_ADD_MEMBERSHIP，在IPPROTO_IP协议层

```
int recv_sock;
struct ip_mreq join_adr;
. . . .
recv_sock=socket(PF_INET, SOCK_DGRAM, 0);
. . . .
join_adr.imr_multiaddr.s_addr="多播组地址信息";
join_adr.imr_interface.s_addr="加入多播组的主机地址信息";
setsockopt(recv_sock, IPPROTO_IP, IP_ADD_MEMBERSHIP, (void*) & join_adr,
sizeof(join_adr));
. . . .
```

- struct ip_mreq

  {

  struct in_addr imr_multiaddr; //加入的组

  struct in_addr imr_interface; //自身地址

  }

多播的发送与接收实例

```
8.  #define TTL 64
9.  #define BUF_SIZE 30
10. void error_handling(char *message);
11.
12. int main(int argc, char *argv[])
13. {
14.     int send_sock;
15.     struct sockaddr_in mul_adr;
16.     int time_live=TTL;
17.     FILE *fp;
18.     char buf[BUF_SIZE];
19.     if(argc!=3) {
20.         printf("Usage : %s <GroupIP> <PORT>\n", argv[0]);
21.         exit(1);
22.     }
```

Sender

```
24.     send_sock=socket(PF_INET, SOCK_DGRAM, 0);
25.     memset(&mul_adr, 0, sizeof(mul_adr));
26.     mul_adr.sin_family=AF_INET;
27.     mul_adr.sin_addr.s_addr=inet_addr(argv[1]);   // Multicast IP
28.     mul_adr.sin_port=htons(atoi(argv[2]));   // Multicast Port
29.
30.     setsockopt(send_sock, IPPROTO_IP,
31.         IP_MULTICAST_TTL, (void*)&time_live, sizeof(time_live));
32.     if((fp=fopen("news.txt", "r"))==NULL)
33.         error_handling("fopen() error");
34.
35.     while(!feof(fp))    /* Broadcasting */
36.     {
37.         fgets(buf, BUF_SIZE, fp);
38.         sendto(send_sock, buf, strlen(buf),
39.             0, (struct sockaddr*)&mul_adr, sizeof(mul_adr));
40.         sleep(2);
41.     }
42.     fclose(fp);
43.     close(send_sock);
44.     return 0;
45. }
```

Sender

Receiver

```
6.  {
7.      int recv_sock;
8.      int str_len;
9.      char buf[BUF_SIZE];
10.     struct sockaddr_in adr;
11.     struct ip_mreq join_adr;
12.     if(argc!=3) {
13.         printf("Usage : %s <GroupIP> <PORT>\n", argv[0]);
14.         exit(1);
15.     }
16.
17.     recv_sock=socket(PF_INET, SOCK_DGRAM, 0);
18.     memset(&adr, 0, sizeof(adr));
19.     adr.sin_family=AF_INET;
20.     adr.sin_addr.s_addr=htonl(INADDR_ANY);
21.     adr.sin_port=htons(atoi(argv[2]));
22.
23.     if(bind(recv_sock, (struct sockaddr*) &adr, sizeof(adr))==-1)
24.         error_handling("bind() error");
25.
26.     join_adr.imr_multiaddr.s_addr=inet_addr(argv[1]);
27.     join_adr.imr_interface.s_addr=htonl(INADDR_ANY);
28.
29.     setsockopt(recv_sock, IPPROTO_IP,
30.         IP_ADD_MEMBERSHIP, (void*)&join_adr, sizeof(join_adr));
```

广播及其特点

- 同时向同一个网络中多个主机发送信息
- 基于UDP实现
- 与多播的区别：
  - 只要加入了多播组，多播可跨不同的网络扩散信息
  - 广播仅限于同一个网络

广播地址

- 直接广播（Directed broadcast）
  - IP地址中，网络地址以外的位全部为1
- 本地广播（Local broadcast）
  - 全1地址255.255.255.255

设置广播的默认套接字

- 默认套接字会阻止广播，需要对SOL_SOCKET协议层的SO_BROADCAST选项进行如下设置

- 只需要设置sender端

```
int send_sock;
int bcast = 1;     // 对变量进行初始化以将 SO_BROADCAST 选项信息改为 1。
. . . .
send_sock = socket(PF_INET, SOCK_DGRAM, 0);
. . . .
setsockopt(send_sock, SOL_SOCKET, SO_BROADCAST, (void*) & bcast, sizeof(bcast));
```

广播发送与接收实例

```
23.     send_sock=socket(PF_INET, SOCK_DGRAM, 0);
24.     memset(&broad_adr, 0, sizeof(broad_adr));
25.     broad_adr.sin_family=AF_INET;
26.     broad_adr.sin_addr.s_addr=inet_addr(argv[1]);
27.     broad_adr.sin_port=htons(atoi(argv[2]));
28.
29.     setsockopt(send_sock, SOL_SOCKET,
30.         SO_BROADCAST, (void*)&so_brd, sizeof(so_brd));
31.     if((fp=fopen("news.txt", "r"))==NULL)
32.         error_handling("fopen() error");
33.
34.     while(!feof(fp))
35.     {
36.         fgets(buf, BUF_SIZE, fp);
37.         sendto(send_sock, buf, strlen(buf),
38.             0, (struct sockaddr*)&broad_adr, sizeof(broad_adr));
39.         sleep(2);
40.     }
41.     close(send_sock);
```

Sender

完