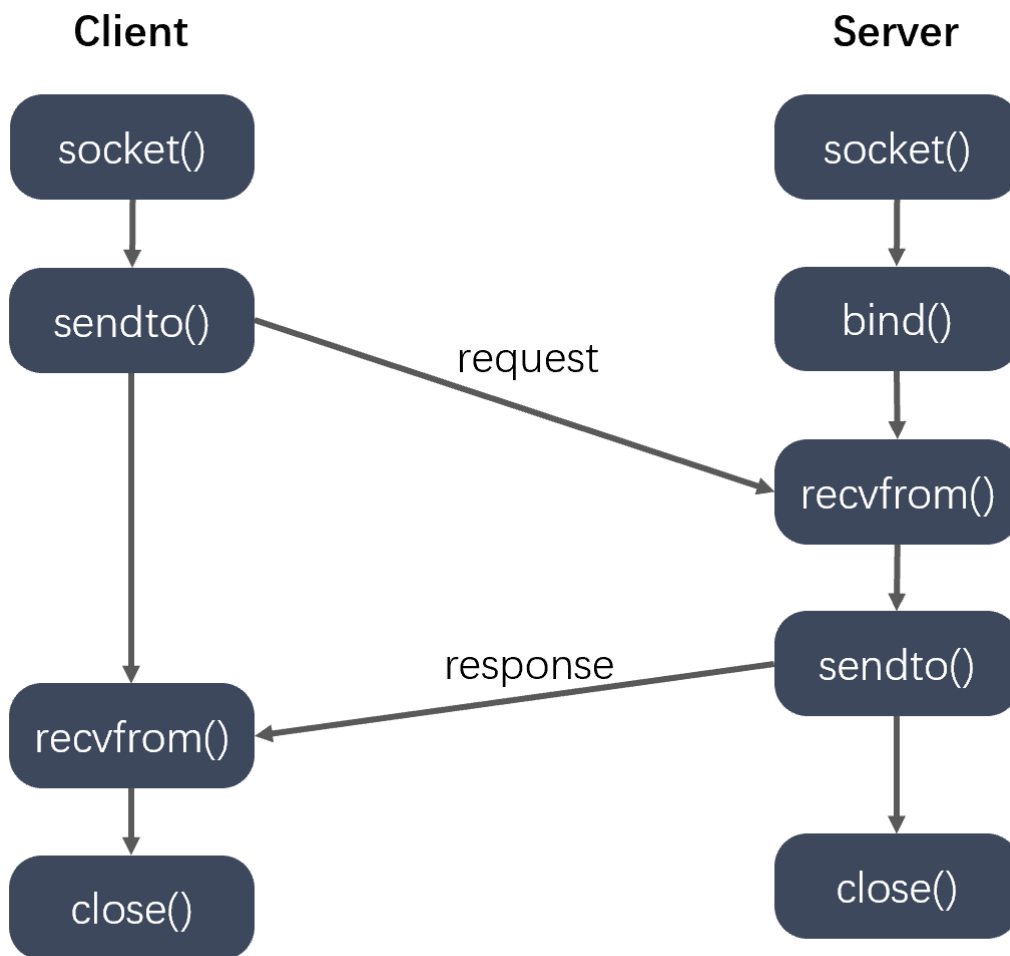


# 1. UDP

## 1.1 UDP 通信



```
#include <sys/types.h>
#include <sys/socket.h>
ssize_t sendto(int sockfd, const void *buf, size_t len, int flags,
               const struct sockaddr *dest_addr, socklen_t addrlen);
```

- 参数:

- sockfd : 通信的fd
- buf : 要发送的数据
- len : 发送数据的长度
- flags : 0
- dest\_addr : 通信的另外一端的地址信息
- addrlen : 地址的内存大小

```
ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags,
                 struct sockaddr *src_addr, socklen_t *addrlen);
```

- 参数:

- sockfd : 通信的fd
- buf : 接收数据的数组
- len : 数组的大小

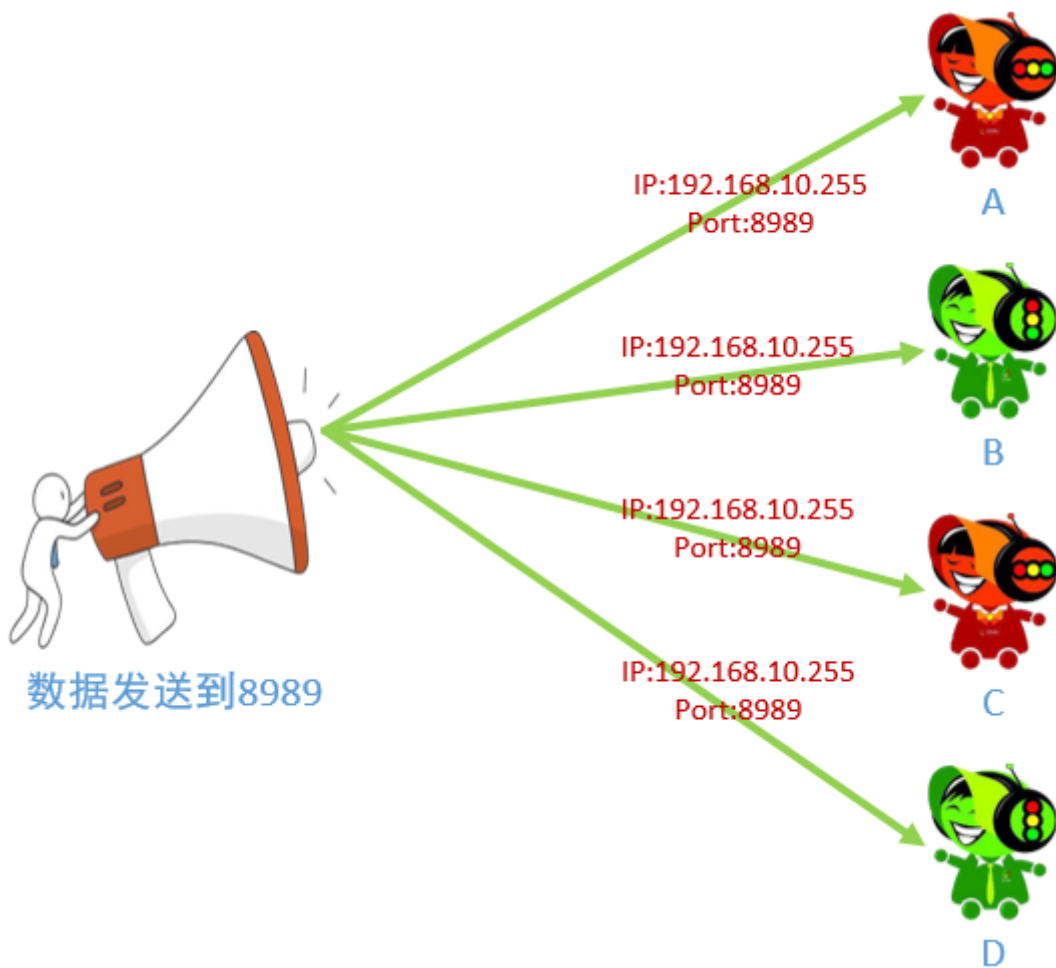
- flags : 0
- src\_addr : 用来保存另外一端的地址信息，不需要可以指定为NULL
- addr\_len : 地址的内存大小

## 1.2 广播

向子网中多台计算机发送消息，并且子网中所有的计算机都可以接收到发送方发送的消息，每个广播消息都包含一个特殊的IP地址，这个IP中子网内主机标志部分的二进制全部为1。

a.只能在局域网中使用。

b.客户端需要绑定服务器广播使用的端口，才可以接收到广播消息。



// 设置广播属性的函数

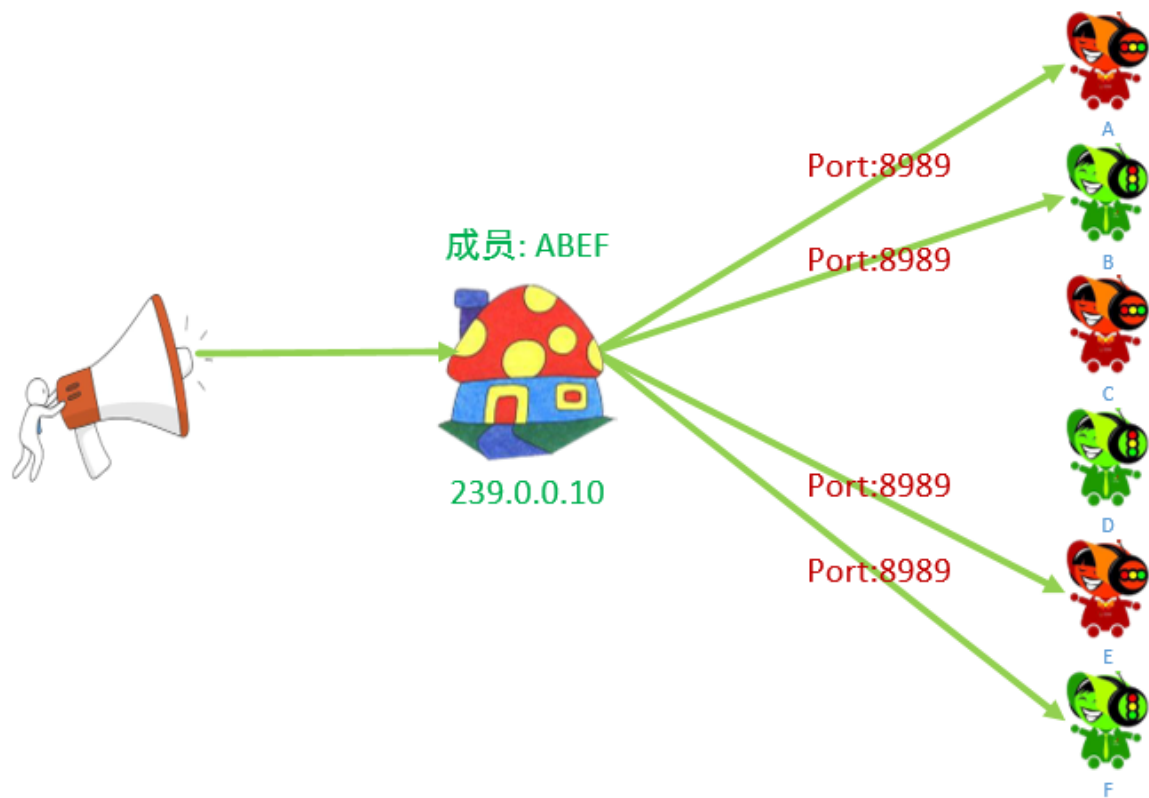
```
int setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen);
```

- sockfd : 文件描述符
- level : SOL\_SOCKET
- optname : SO\_BROADCAST
- optval : int类型的值，为1表示允许广播
- optlen : optval的大小

## 1.3 组播(多播)

单播地址标识单个 IP 接口，广播地址标识某个子网的所有 IP 接口，多播地址标识一组 IP 接口。单播和广播是寻址方案的两个极端（要么单个要么全部），多播则意在两者之间提供一种折中方案。多播数据报只应该由对它感兴趣的接口接收，也就是说由运行相应多播会话应用系统的主机上的接口接收。另外，广播一般局限于局域网内使用，而多播则既可以用于局域网，也可以跨广域网使用。

- a.组播既可以用于局域网，也可以用于广域网
- b.客户端需要加入多播组，才能接收到多播的数据



## 组播地址

IP 多播通信必须依赖于 IP 多播地址，在 IPv4 中它的范围从 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：

IP地址	说明
224.0.0.0~224.0.0.255	局部链接多播地址：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
224.0.1.0~224.0.1.255	预留多播地址：公用组播地址，可用于Internet；使用前需要申请
224.0.2.0~238.255.255.255	预留多播地址：用户可用组播地址(临时组地址)，全网范围内有效
239.0.0.0~239.255.255.255	本地管理组播地址，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

### • 设置组播

```
int setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen);
```

// 服务器设置多播的信息，外出接口

```

- level : IPPROTO_IP
- optname : IP_MULTICAST_IF
- optval : struct in_addr

// 客户端加入到多播组:
- level : IPPROTO_IP
- optname : IP_ADD_MEMBERSHIP
- optval : struct ip_mreq

struct ip_mreq
{
    /* IP multicast address of group. */
    struct in_addr imr_multiaddr;    // 组播的IP地址

    /* Local IP address of interface. */
    struct in_addr imr_interface;    // 本地的IP地址
};

typedef uint32_t in_addr_t;
struct in_addr
{
    in_addr_t s_addr;
};

```

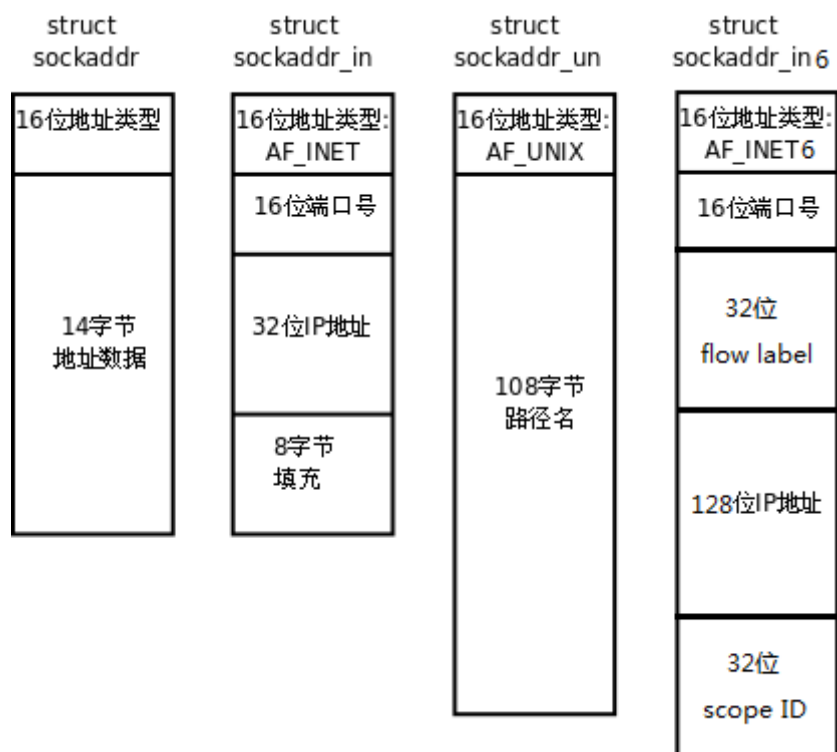
## 2. 本地套接字

本地套接字的作用：本地的进程间通信

有关系的进程间的通信

没有关系的进程间的通信

本地套接字实现流程和网络套接字类似，一般呢采用TCP的通信流程。



```
// 本地套接字通信的流程 - tcp

// 服务器端
1. 创建监听的套接字
   int lfd = socket(AF_UNIX/AF_LOCAL, SOCK_STREAM, 0);
2. 监听的套接字绑定本地的套接字文件 -> server端
   struct sockaddr_un addr;
   // 绑定成功之后，指定的sun_path中的套接字文件会自动生成。
   bind(lfd, addr, len);
3. 监听
   listen(lfd, 100);
4. 等待并接受连接请求
   struct sockaddr_un cliaddr;
   int cfd = accept(lfd, &cliaddr, len);
5. 通信
   接收数据: read/recv
   发送数据: write/send
6. 关闭连接
   close();
```

```
// 客户端的流程
1. 创建通信的套接字
   int fd = socket(AF_UNIX/AF_LOCAL, SOCK_STREAM, 0);
2. 监听的套接字绑定本地的IP 端口
   struct sockaddr_un addr;
   // 绑定成功之后，指定的sun_path中的套接字文件会自动生成。
   bind(lfd, addr, len);
3. 连接服务器
   struct sockaddr_un serveraddr;
   connect(fd, &serveraddr, sizeof(serveraddr));
4. 通信
   接收数据: read/recv
   发送数据: write/send
5. 关闭连接
   close();
```

```
// 头文件: sys/un.h
#define UNIX_PATH_MAX 108
struct sockaddr_un {
    sa_family_t sun_family; // 地址族协议 af_local
    char sun_path[UNIX_PATH_MAX]; // 套接字文件的路径，这是一个伪文件，大小永远=0
};
```