# 一、构造原理:

自定义报文结构,实际上就是对报文协议所对应的结构体数据的自定义。系统提供原始套接字可以模拟发送报文，当我们自定义报文结构，让它符合协议族的报文头结构，那么就可以实现模拟发送报文的目的。

# 二、构造方法及步骤:

## 1.为构造变量分配空间

```
src_mac = (uint8_t *) malloc (6 * sizeof (uint8_t));
dst_mac = (uint8_t *) malloc (6 * sizeof (uint8_t));
data = (uint8_t *) malloc (IP_MAXPACKET * sizeof (uint8_t));
ether_frame = (uint8_t *) malloc (IP_MAXPACKET * sizeof (uint8_t));
interface = (char *) malloc (40 * sizeof (char));
target = (char *) malloc (40 * sizeof (char));
src_ip = (char *) malloc (INET_ADDRSTRLEN * sizeof (char));
dst_ip = (char *) malloc (INET_ADDRSTRLEN * sizeof (char));
ip_flags = (int *) malloc (4 * sizeof (int));
```

## 2.指定报文发送接口

```
strcpy (interface, "wlp6s0");
```

## 3.创建原始套接字,并设置报文头自定义,而不是系统提供

```
sd = socket (PF_PACKET, SOCK_RAW, 0)
```

## 4.构造报文头部,分别为 IP 头,ICMP 头和以太网 MAC 帧头

根据网络协议报文头结构,构造结构体,填充报文头.

*IPv4 头*

```
// IPv4 首部长度 (4 bits):20 字节
iphdr.ip_hl = IP4_HDRLEN / sizeof (uint32_t);

// IP 版本 (4 bits): IPv4
iphdr.ip_v = 4;

// TOS (8 bits)
iphdr.ip_tos = 0;

// 总长度 (16 bits): IP header + ICMP header + ICMP data
iphdr.ip_len = htons (IP4_HDRLEN + ICMP_HDRLEN + datalen);

// ID 标识 (16 bits): unused, since single datagram
iphdr.ip_id = htons (0);

// 标志 片偏移 (3, 13 bits): 0
```

```
// 标志 (1 bit)
ip_flags[0] = 0;

// 分片标志(不分片)(1 bit)
ip_flags[1] = 0;

// 剩余分片 (1 bit)
ip_flags[2] = 0;

// 片偏移 (13 bits)
ip_flags[3] = 0;
iphdr.ip_off = htons ((ip_flags[0] << 15)
                  + (ip_flags[1] << 14)
                  + (ip_flags[2] << 13)
                  +  ip_flags[3]);

// Time-to-Live (8 bits): 默认最大
iphdr.ip_ttl = 255;

// 传输协议 (8 bits): 1 -> ICMP
iphdr.ip_p = IPPROTO_ICMP;

// 源 IP (32 bits)
status = inet_pton (AF_INET, src_ip, &(iphdr.ip_src));

// 目的 IP (32 bits)
status = inet_pton (AF_INET, dst_ip, &(iphdr.ip_dst));

// IPv4 首部校验和(16 bits): 初始化设置为 0
iphdr.ip_sum = 0;
iphdr.ip_sum = checksum ((uint16_t *) &iphdr, IP4_HDRLEN);
```

*ICMP 头部*

```
// 信息类型 (8 bits): echo request
icmphdr.icmp_type = ICMP_ECHO;

// 代码(8 bits): echo request
icmphdr.icmp_code = 0;

//ICMP 首部校验和 (16 bits): 初始化设置为 0
icmphdr.icmp_cksum = icmp4_checksum (icmphdr, data, datalen);

// 标识符 (16 bits)
```

```
icmphdr.icmp_id = htons (1000);

// 序列号 (16 bits): 从0开始
icmphdr.icmp_seq = htons (0);
```

***设置以太网MAC帧头部***

```
// 以太网帧长度 = 以太网头部长(MAC + MAC + 以太网类型) + 以太网数据(IP头 + ICMP头 + ICMP数据)
frame_length = 6 + 6 + 2 + IP4_HDRLEN + ICMP_HDRLEN + datalen;

// 复制源MAC和目的MAC
memcpy (ether_frame, dst_mac, 6);
memcpy (ether_frame + 6, src_mac, 6);

// 以太网类型 (ETH_P_IP for IPv4).
ether_frame[12] = ETH_P_IP / 256;
ether_frame[13] = ETH_P_IP % 256;
```

***以太网数据(IPv4头 + ICMP头 + ICMP数据).***

```
// IPv4 头
memcpy (ether_frame + ETH_HDRLEN, &iphdr, IP4_HDRLEN);

// ICMP 头
memcpy (ether_frame + ETH_HDRLEN + IP4_HDRLEN, &icmphdr, ICMP_HDRLEN);

// ICMP 数据
memcpy (ether_frame + ETH_HDRLEN + IP4_HDRLEN + ICMP_HDRLEN, data, datalen);
```

### 5.发送报文

```
bytes = sendto (sd, ether_frame, frame_length, 0, (struct sockaddr *) &device, sizeof (device));
```

### 6.关闭原始套接字

```
close (sd);
```

## 三、注意事项:

1.本次实验构造的是ICMP数据包;

2.实验机器操作系统为 Linux Deepin,默认无线网络设备接口为"wlp6s0",所以指定报文接口赋值为"wlp6s0":strcpy (interface, "wlp6s0"),应按照实际情况修改此处代码,大部分应为"wlan0";

3.自定义的目的MAC地址为ff:ff:ff:ff:ff:ff,可以按照自己的需求更改

```
//设置目的MAC地址
```

```
dst_mac[0] = 0xff;
dst_mac[1] = 0xff;
dst_mac[2] = 0xff;
dst_mac[3] = 0xff;
dst_mac[4] = 0xff;
dst_mac[5] = 0xff;
```

4.自定义的源 IP 为 192.168.0.1,目的 IP 为 118.89.236.39(本人的远程主机 IP)

```
//源 IP
strcpy (src_ip, "192.168.0.1");
//目的 ip
strcpy (target, "www.nextlegend.cn");
//目标为本人的一个网址域名,地址解析为远程主机 IP:118.89.236.39
```

5.自定义的 ICMP 数据部分为"Test"

```
//ICMP 数据
datalen = 4;
data[0] = 'T';
data[1] = 'e';
data[2] = 's';
data[3] = 't';
```

6.部分函数说明

icmp 校验函数

uint16_t icmp4_checksum (struct icmp icmphdr, uint8_t *payload, int payloadlen)

IP 首部校验

uint16_t checksum (uint16_t *addr, int len)