哈爾濱工業大學

实验报告

实 验（三）

题　　目　基于 libnet 的程序设计

专　　业　计　算　机　类

学　　号　1160300620

班　　级　1603006

学　　生　张　恒

实 验 地 点　格物 214

实 验 日 期　2019.3.28

计算机科学与技术学院

目 录

第 1 章 实验基本信息

1.1 实验目的

（1）了解构造数据包的原理及流程;

（2）能够熟练使用 libnet 构造数据包;

（3）能结合前面实验验证构造的数据包；

1.2 实验内容

（1） 编程实现基于 libnet 的数据包构造；

（2） 结合前面实验给出验证过程；

（3） 并能够对源码进行解释；

1.3 实验环境

本实验中使用的系统环境是 Ubuntu 18.04，编程语言为 C 语言，使用了 gcc 编译器；

第 2 章 实验原理

2.1 libnet 的安装

打开 ubuntu 的 bash shell，输入如下的指令

$$sudo\ \ apt\ \ install\ \ libnet-dev$$

等待安装成功即可。

2.2 libnet 的原理

现代计算机广泛采用 5 层模型，不同的操作系统都抽象出了五个层次，每个曾对相邻的层通过 API 提供服务。如图 2-1 所示，实际上，5 个层次中的物理层程序员是无法直接操控的，所以可以视为 4 个层次。Libnet 实际上就是让程序员有一个更简单的方式调用操作系统中各层提供的 API。

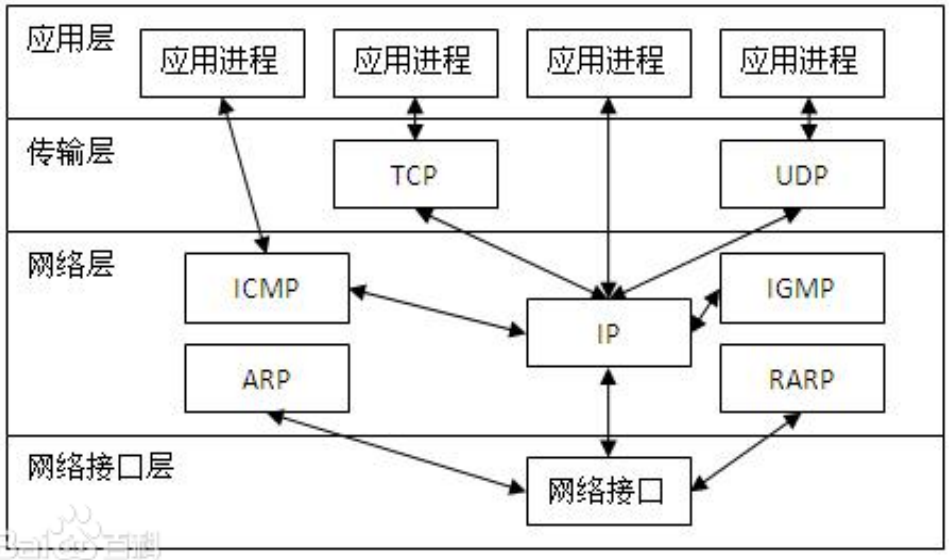

图 2-1 操作系统中程序员可操纵的四个层

2.3 本实验中用到的 API

```
// 数据包内存初始化及环境建立。函数返回一个 libnet * 类型的指针，后面的操作
// 都得使用这个指针
libnet_t *libnet_init(int injection_type, char *device, char *err_buf);

// 释放资源
void libnet_destroy(libnet_t *l);

// 将点分十进制数串转换为网络字节序 ip 地址
u_int32_t libnet_name2addr4(libnet_t *l,  char *host_name,  u_int8_t use_name);

// 获取接口设备硬件地址
struct libnet_ether_addr* libnet_get_hwaddr(libnet_t *l);

// 构造 UDP 数据包
libnet_ptag_t libnet_build_udp(u_int16_t sp, u_int16_t dp, u_int16_t len, u_int16_t sum,
u_int8_t *payload, u_int32_t payload_s, libnet_t *l, libnet_ptag_t ptag);

// 构造 TCP 数据包
libnet_ptag_t libnet_build_tcp(u_int16_t sp, u_int16_t dp, u_int32_t seq, u_int32_t ack,
u_int8_t control, u_int16_t win, u_int16_t sum, u_int16_t urg, u_int16_t len, u_int8_t
*payload, u_int32_t payload_s, libnet_t *l, libnet_ptag_t ptag );

// 构造 ipv4 数据包
libnet_ptag_t libnet_build_ipv4(u_int16_t ip_len, u_int8_t tos, u_int16_t id, u_int16_t
flag, u_int8_t ttl, u_int8_t prot, u_int16 sum, u_int32_t src, u_int32_t dst, u_int8_t
*payload, u_int32_t payload_s,libnet_t *l, libnet_ptag_t ptag );

// 构造以太网数据包
libnet_ptag_t  libnet_build_ethernet(u_int8_t*dst,  u_int8_t  *src,  u_int16_ttype,
u_int8_t*payload, u_int32_tpayload_s, libnet_t*l, libnet_ptag_t ptag );

// 发送数据包
int libnet_write(libnet_t * l);
```

2.4 利用 libnet 构造数据包的流程

如图 2-2 所示，使用 libnet 时，首先需要初始化，然后才能够构造数据包并发送，最后再释放初始化时申请的资源。其中，初始化之后，可以多次构造数据包，多次发送数据包。

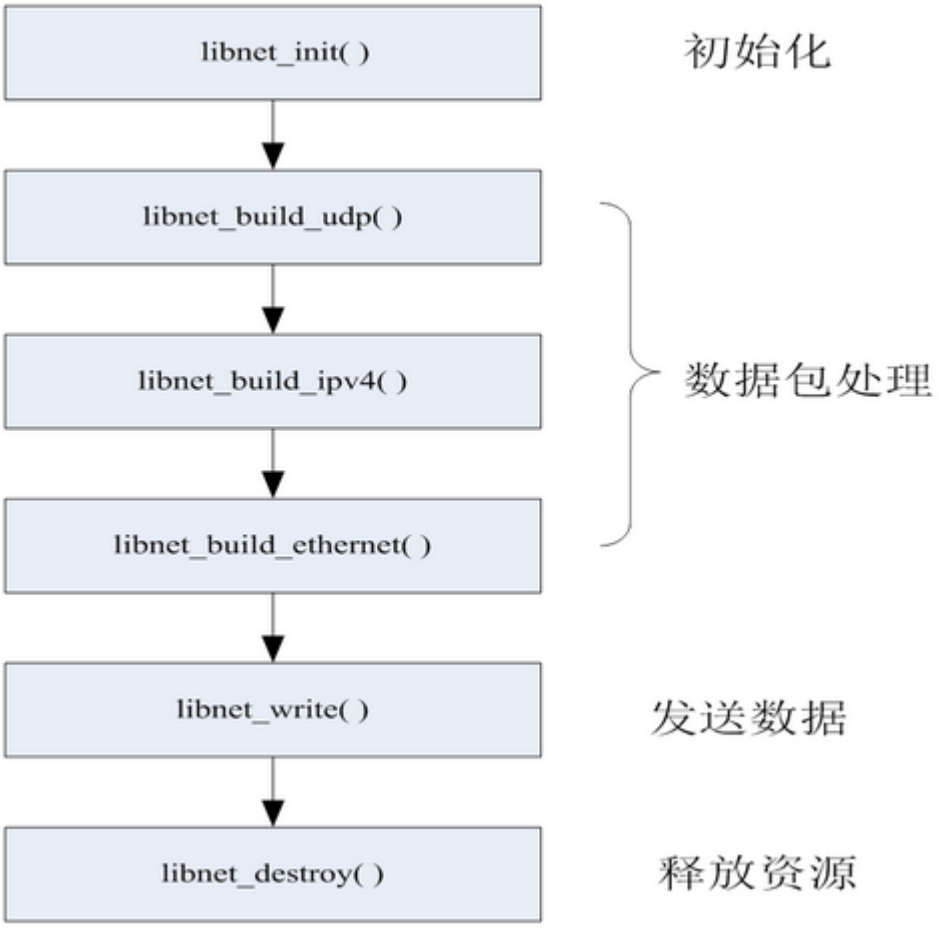

图 2-2 使用 libnet 的流程

第 3 章 程序流程设计

程序的设计流程比较简单。如图 3-1 所示，用户打开程序后，程序自动进行初始化，并给用户展示一个菜单，可以选择：1）构造 TCP 包；2）构造 UDP 包；3）退出。

若是选择退出，程序会释放申请的资源并退出。

若是选择构造 TCP 包或者是 UDP 包，程序会要求用户输入相应的信息，并且发出。以构造一个 UDP 包为例：

（1） 输入源端口和目的端口；
（2） 输入需要携带的数据，允许为空；
（3） 输入源 ip，可以为空，默认为 127.0.0.1；
（4） 输入目的 ip；
（5） 输入源 mac，可以为空，默认为当前网卡的 mac；
（6） 输入目的 mac；

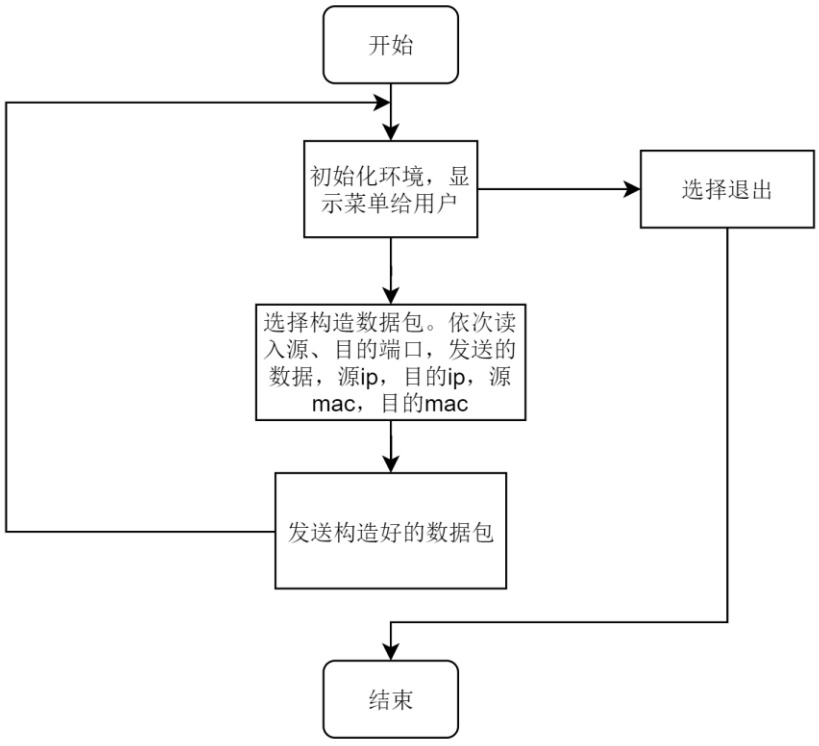

图 3-1 程序的整体流程

第 4 章 代码实现

4.1 main 函数的实现

main 函数有一个无线循环执行代码块。在进入循环代码块前，会初始化 libent，在离开循环代码块前，会释放初始化时申请的资源。在循环代码块中，每一次循环都能够构造一个 tcp 包或者是 udp 包并发送。

值得注意的是，在每次循环中，程序是否上一次构造 udp（tcp）包，而这次构造 tcp（udp）包。当构造第 i+1 个 udp（tcp）包时，libnet_build_udp（libnet_build_tcp）、libnet_build_ip、libnet_build_ether 函数中参数 ptag 需要是构造第 i 个 udp（tcp）时的相应函数返回值。因此，在从构造 udp（tcp）转换为构造 tcp（udp）时，需要将上一次构造 udp（tcp）时的返回值缓存下来，并设置为上一次构造 tcp（udp）时的返回值。具体实现的代码如图 4-1 所示。

```
if(choice!=pre_choice){
    pre_choice = choice;
    int tmp = ptag_ip;
    ptag_ip = pre_ptag_ip;
    pre_ptag_ip = tmp;
    tmp = ptag_ether;
    ptag_ether = pre_ptag_ether;
    pre_ptag_ether = tmp;
}
```

图 4-1 检查构造包的类型是否发生变化，并进行相应处理

4.2 init_libnet 函数的实现

该函数的定义如下

int init_libnet(libnet_t **device);

（1） libnet_t **device：用以存储打开的设备的句柄；
（2） 返回值为 1 表示成功，0 表示失败；

该函数完成环境的初始化。

4.3 buildtcp_libnet 函数的实现

该函数的定义如下：

libnet_ptag_t buildtcp_libnet(libnet_t *device, libnet_ptag_t ptag, u_int16_t *len);

（1） libnet_t *device：用以指明网卡设备；
（2） libnet_ptag_t ptag：指明本次构造 tcp 包的 ptag；
（3） u_int16_t *len：用以存储 tcp 包的总长度
（4） 返回本次构造 tcp 包返回的 ptag；

该函数的功能是让用户输入源端口、目的端口和需要携带的信息，并构建 tcp 包。

4.4 buildudp_libnet 函数的实现

该函数的定义如下：

libnet_ptag_t buildudp_libnet(libnet_t *device, libnet_ptag_t ptag, u_int16_t *len);

（1） libnet_t *device：用以指明网卡设备；
（2） libnet_ptag_t ptag：指明本次构造 udp 包的 ptag；
（3） u_int16_t *len：用以存储 udp 包的总长度
（4） 返回本次构造 udp 包返回的 ptag；

该函数的功能是让用户输入源端口、目的端口和需要携带的信息，并构建 udp 包。

4.5 buildip_libnet 函数的实现

该函数的定义如下：

libnet_ptag_t buildip_libnet(libnet_t *device, libnet_ptag_t ptag, u_int16_t *len, int proc);

（1） libnet_t *device：用以指明网卡设备；
（2） libnet_ptag_t ptag：指明本次构造 ip 包的 ptag；
（3） u_int16_t *len：用以存储 ip 包的总长度；
（4） int proc：用以指明上层协议的类型；
（5） 返回本次构造 ip 包返回的 ptag；

该函数的功能是让用户输入源 ip（允许为空，为空时使用默认值 127.0.0.1）、目的 ip，并构建 ip 包。

4.6 buildether_libnet 函数的实现

该函数的定义如下：

libnet_ptag_t buildlink_libnet(libnet_t *device, libnet_ptag_t ptag, int proc);

（1） libnet_t *device：用以指明网卡设备；
（2） libnet_ptag_t ptag：指明本次构造以太网包的 ptag；
（3） int proc：用以指明上层协议的类型；
（4） 返回本次构造以太网包返回的 ptag；

该函数的功能是让用户输入源 mac、目的 mac，并构建以太网包。

第 5 章 运行结果

函数的结果比较简单。

打开程序，会出现一个菜单，可以选择 1 或者 2 进行组包。如图 5-1 所示是选择 1 的组包的过程。为求方便，这里面省略了一些输入过程，默认源端口和目的端口均为 8080，源 ip 和目的 ip 均为 127.0.0.1，目的 mac 和源 mac 均为当前网卡的 mac 地址。

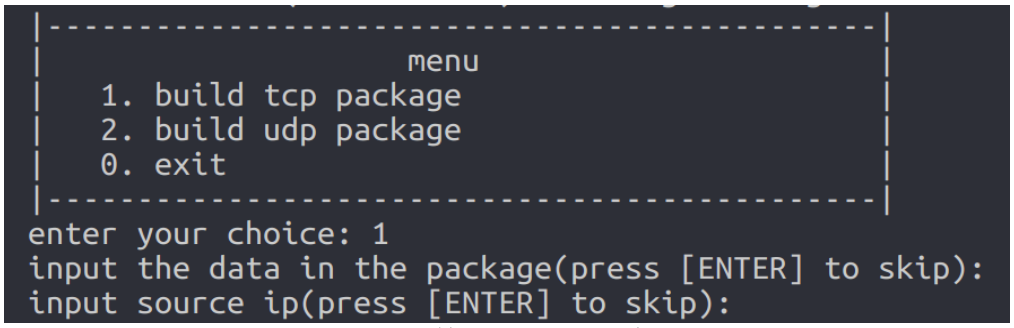

图 5-1 构造 tcp 包的过程

如图 5-2 所示，所示是选择 2 的组包的过程，这是在组了若干个 tcp 包后切换组包的类型。同样，为求方便，这里面省略了一些输入过程，默认源端口和目的端口均为 8080，源 ip 和目的 ip 均为 127.0.0.1，目的 mac 和源 mac 均为当前网卡的 mac 地址。

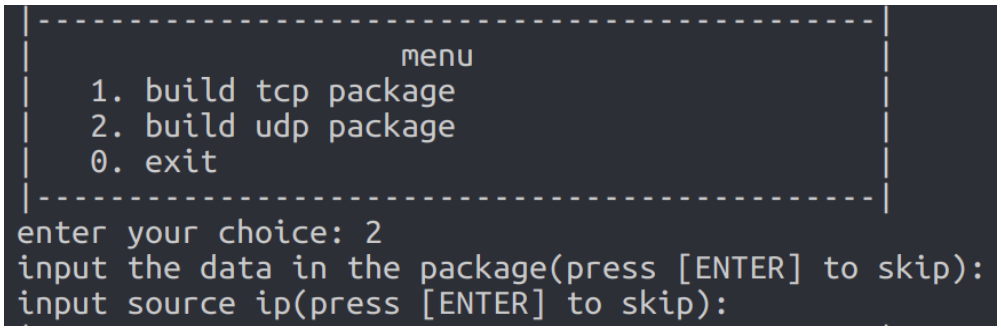

图 5-2 构造 udp 包的过程

如图 5-3 所示，是利用上次实验中的程序进行抓包，将源 ip、源端口、目的 ip、目的端口存储在文件中的结果。

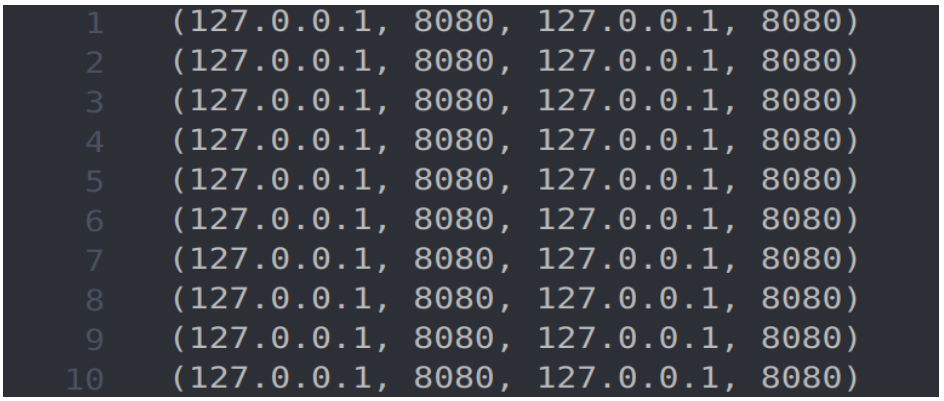

图 5-3 抓包结果

第 6 章 总结

6.1 本次实验的收获

（1）了解并掌握了使用 libnet 提供的 API 编写抓包程序技能；
（2）理解了构造数据包并发送的原理；

6.2 本次实验的难点

libnet 单次初始化后，允许多次组包并发送。初始化后组第一个包的 ptag 为 0，之后每次组包该参数是相应函数上一次组包的返回值。难点就在于，这里所谓的上一次需要区分上层协议的类型。以构造 tcp 包和 udp 包为例。

当构造第 i+1 个 udp（tcp）包时，libnet_build_udp（libnet_build_tcp）、libnet_build_ip、libnet_build_ether 函数中参数 ptag 需要是构造第 i 个 udp（tcp）时的相应函数返回值。因此，在从构造 udp（tcp）转换为构造 tcp（udp）时，需要将上一次构造 udp（tcp）时的返回值缓存下来，并设置为上一次构造 tcp（udp）时的返回值。

第 7 章 参考文献

[1] https://blog.csdn.net/tennysonsky/article/details/44944849.