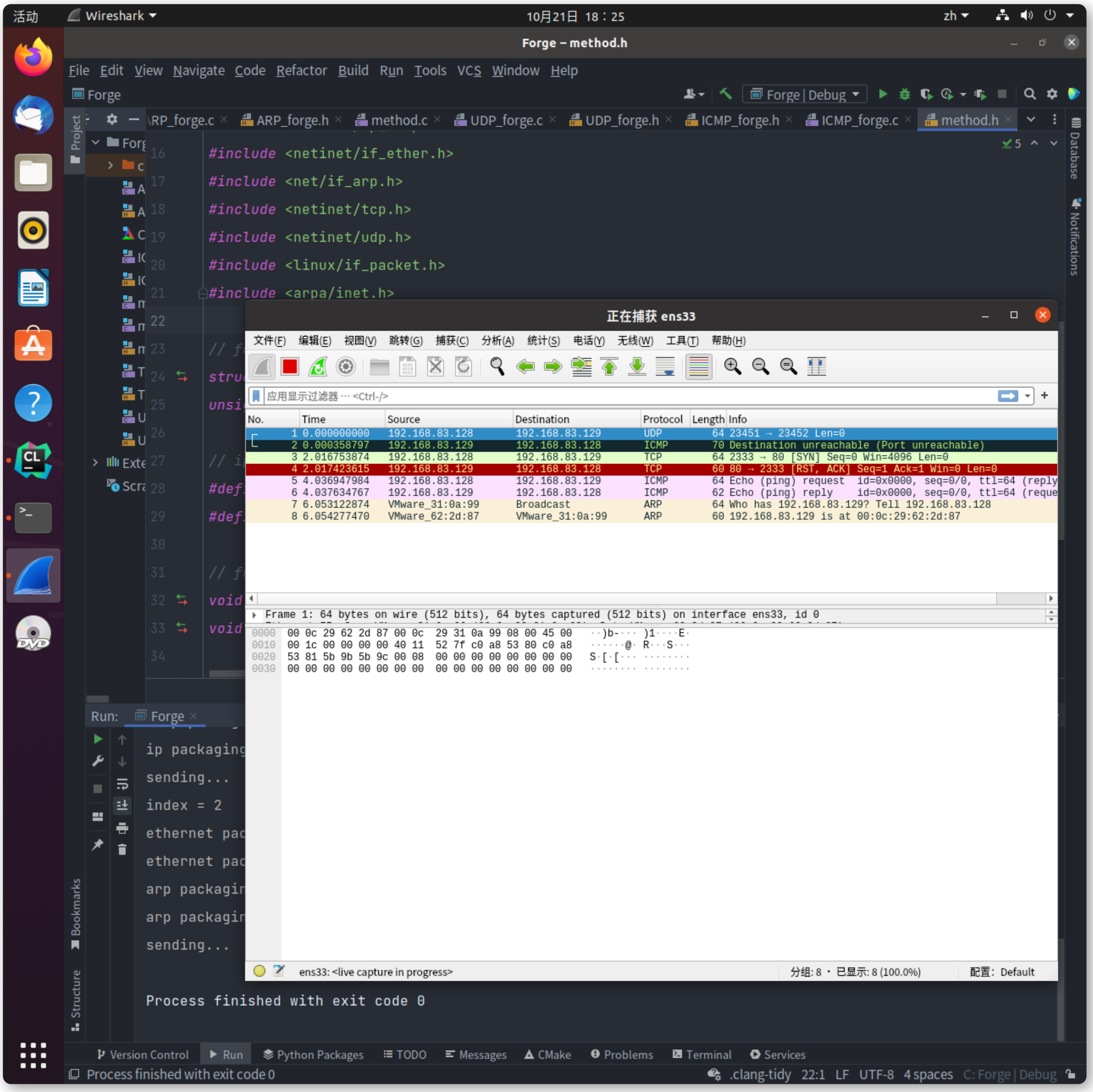
mini Network-Analyzer

Part D：packet spoofing

实验结果：前8个包



```
#include <netinet/if_ether.h>
#include <net/if_arp.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
#include <linux/if_packet.h>
#include <arpa/inet.h>
```

```
No. Time        Source           Destination      Protocol Length Info
  1 0.000000000 192.168.83.128   192.168.83.129   UDP      64 23451 → 23452 Len=0
  2 0.000358797 192.168.83.129   192.168.83.128   ICMP     70 Destination unreachable (Port unreachable)
  3 2.016753874 192.168.83.128   192.168.83.129   TCP      64 2333 → 80 [SYN] Seq=0 Win=4096 Len=0
  4 2.017423615 192.168.83.129   192.168.83.128   TCP      60 80 → 2333 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  5 4.036947984 192.168.83.128   192.168.83.129   ICMP     64 Echo (ping) request  id=0x0000, seq=0/0, ttl=64 (reply
  6 4.037634767 192.168.83.129   192.168.83.128   ICMP     62 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64 (reque
  7 6.053122874 VMware_31:0a:99  Broadcast        ARP      64 Who has 192.168.83.129? Tell 192.168.83.128
  8 6.054277470 VMware_62:2d:87  VMware_31:0a:99  ARP      60 192.168.83.129 is at 00:0c:29:62:2d:87
```

```
Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface ens33, id 0

0000  00 0c 29 62 2d 87 00 0c  29 31 0a 99 08 00 45 00   ··)b-··· )1····E·
0010  00 1c 00 00 00 00 40 11  52 7f c0 a8 53 80 c0 a8   ······@· R···S···
0020  53 81 5b 9b 5b 9c 00 08  00 00 00 00 00 00 00 00   S·[·[··· ········
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
```

ens33: <live capture in progress>　分组: 8 · 已显示: 8 (100.0%)　配置: Default

```
ip packaging
sending...
index = 2
ethernet pac
ethernet pac
arp packagin
arp packagin
sending...

Process finished with exit code 0
```

文件结构

```
|-- Forge
    |-- method.h
    |-- method.c
    |-- UDP_forge.h
    |-- UDP_forge.c
    |-- TCP_forge.h
    |-- TCP_forge.c
    |-- ARP_forge.h
    |-- ARP_forge.c
    |-- ICMP_forge.h
    |-- ICMP_forge.c
    |-- main
```

利用Raw Socket可以实现发送用户自定义的数据包,在实现各种forgery之前，先定义一个工具文件 `method.h/method.c` ，实现公共部分，定义的接口如下

```
void get_eth_index(int sock_raw);
void get_mac(int sock_raw, uint16_t protocol, int *total_len);

void str2ip(char *src, unsigned char *dst);
unsigned short checksum(unsigned char *buf, int size);

void forge(void(*pro)(int *total_len), uint16_t mac_protocol, uint8_t ip_protocol);      // forge template
void send_ip(void(*func)(int *total_len), int sock_raw, uint8_t protocol, int *total_len);
```

在forge函数中，通过传入相应的协议函数指针，即可生成一个对应协议的数据包，并且由于大多数网络层以上的协议都是基于IP的，故将 `send_ip` 也写成一个模板。

例如要发送TCP，参数为： `1：填充协议的函数指针；2：MAC的协议字段；3：IP的协议字段(ARP协议此参数填0即可)`

```
forge(send_tcp, ETH_P_IP, IPPROTO_TCP);
```

其中 `forge()` 实现如下

```
void forge(void(*pro)(int *total_len), uint16_t mac_protocol, uint8_t ip_protocol){
    int sock_raw = sock_raw = socket(AF_PACKET,SOCK_RAW,ETH_P_IP);
    if(sock_raw == -1)
        printf("error in socket\n");

    send_buff = (unsigned char*)malloc(64);            // increase in case of large data.
    memset(send_buff, 0, 64);

    int total_len = 0;
    get_eth_index(sock_raw);                    // interface number
    get_mac(sock_raw, mac_protocol, &total_len);

    if(mac_protocol == ETH_P_ARP)  pro(&total_len);
    else send_ip(pro, sock_raw, ip_protocol, &total_len);

    struct sockaddr_ll sadr_ll;
    sadr_ll.sll_ifindex = ifreq_idx.ifr_ifindex;
    sadr_ll.sll_halen   = ETH_ALEN;
    for(int i = 0; i < 6; i++)
        sadr_ll.sll_addr[i]  = dst_mac[i];

    int cnt = 1;
    printf("sending...\n");

    while(cnt-- > 0){
        int send_len = sendto(sock_raw, send_buff, 64, 0, (const struct sockaddr*)&sadr_ll, sizeof(struct sockaddr_ll));
        if(send_len < 0){
            printf("error in sending....sendlen = %d....errno = %d\n", send_len, errno);
            break;
        }
    }
    close(sock_raw);
}
```

1.UDP packet forgery

```c
void send_udp(int *total_len){
    printf("udp packaging start ... \n");
    struct udphdr *uh = (struct udphdr *)(send_buff + sizeof(struct iphdr) + sizeof(struct ethhdr));

    uh->source  = htons(23451);
    uh->dest    = htons(23452);
    uh->check   = 0;   //Many OSes ignore this field , so we do not calculate it.

    *total_len += sizeof(struct udphdr);
    uh->len     = htons((*total_len - sizeof(struct iphdr) - sizeof(struct ethhdr)));
    printf("udp packaging start ... \n");
}
```

2.TCP packet forgery

由于TCP的checksum需要添加伪首部才能计算正确，故在头文件定义伪首部

```c
struct psd_hdr{
    unsigned int src_ipaddr;
    unsigned int dst_ipaddr;
    unsigned char nop;
    unsigned char protocol;
    unsigned short tcp_len;
};
```

`TCP_forge.c` 如下

```c
void send_tcp(int *total_len){
    struct tcphdr *th = (struct tcphdr *)(send_buff + sizeof(struct iphdr) + sizeof(struct ethhdr));

    printf("tcp packaging start ... \n");
    //write udp_header
    th->th_sport     = htons(2333);
    th->th_dport     = htons(80);
    th->th_seq       = htons(0);
    th->th_ack       = htons(0);
    th->th_off       = sizeof (struct tcphdr) / 4;
    th->th_flags     = TH_SYN;
    th->th_win       = htons(4096);
    // define pseudo header for tcp checksum
    struct psd_hdr psd;
    psd.src_ipaddr = inet_addr(src_ip);
    psd.dst_ipaddr = inet_addr(dst_ip);
    psd.nop = 0;
    psd.protocol = 6;
    psd.tcp_len = htons(sizeof (struct tcphdr));

    unsigned char cur[1000];

    memcpy(cur, &psd, sizeof(struct psd_hdr));
    memcpy(cur + sizeof (struct psd_hdr), th, sizeof(struct
tcphdr));
    th->th_sum = htons(checksum(cur, sizeof(struct tcphdr) +
sizeof (struct psd_hdr)));

    *total_len += sizeof(struct tcphdr);
    printf("tcp packaging done ... \n");
}
```

3.ICMP packet forgery

由于struct icmp中含有union，为了方便起见，定义宏

```


// define icmp union
#define icmp_id         icmp_hun.ih_idseq.icd_id
#define icmp_seq        icmp_hun.ih_idseq.icd_seq
```

`ICMP_forge.c` 实现如下

```c
void send_icmp(int *total_len){
    printf("icmp packaging start ... \n");
    struct icmp *ih = (struct icmp *)(send_buff + sizeof(struct iphdr) + sizeof(struct ethhdr));

    ih->icmp_type = ICMP_ECHO;
    ih->icmp_code = 0;
    ih->icmp_cksum = htons(checksum((unsigned char *)(send_buff + sizeof (struct ethhdr) + sizeof (struct iphdr)), sizeof (struct icmp)));
    ih->icmp_id = 0;
    ih->icmp_seq = 0;

    *total_len += sizeof(struct icmp);
    printf("icmp packaging done ... \n");
}
```

4.ARP packet forgery

由于 `ARP` 与 `IP` 一样都是网络层协议，故在 `MAC` 层需要将 `protocol` 改为 `ETH_P_ARP`。

并且ARP协议的头部定义需要在 `<netinet/if_ether.h>` 头文件下找，而非 `<net/if_arp.h>`

`ARP_forge.c` 定义如下

```c
void send_arp(int *total_len){
    printf("arp packaging start ... \n");
    struct ether_arp *ah = (struct ether_arp *)(send_buff + sizeof (struct ethhdr));

    str2ip(src_ip, ah->arp_spa);
    str2ip(dst_ip, ah->arp_tpa);

    for(int i = 0; i < 6; i++){
        ah->arp_sha[i] = (unsigned char)(ifreq_mac.ifr_hwaddr.sa_data[i]);
        ah->arp_tha[i]   =  (unsigned char)0x00;
    }

    ah->ea_hdr.ar_hrd = htons(0x01);
    ah->ea_hdr.ar_pro = htons(ETH_P_IP);
    ah->ea_hdr.ar_hln = ETH_ALEN;
    ah->ea_hdr.ar_pln = 0x04;
    ah->ea_hdr.ar_op = htons(ARPOP_REQUEST);

    *total_len += sizeof (struct ether_arp);
    printf("arp packaging done ... \n");
}

```

Part E：Design your own MAC layer protocol

文件结构：两个主机源代码大部分相同

```
|-- final
    |-- protocol.h    // define struct own_hdr and interface for send or receive
    |-- send.c
    |-- receive.c
    |-- main.c
```

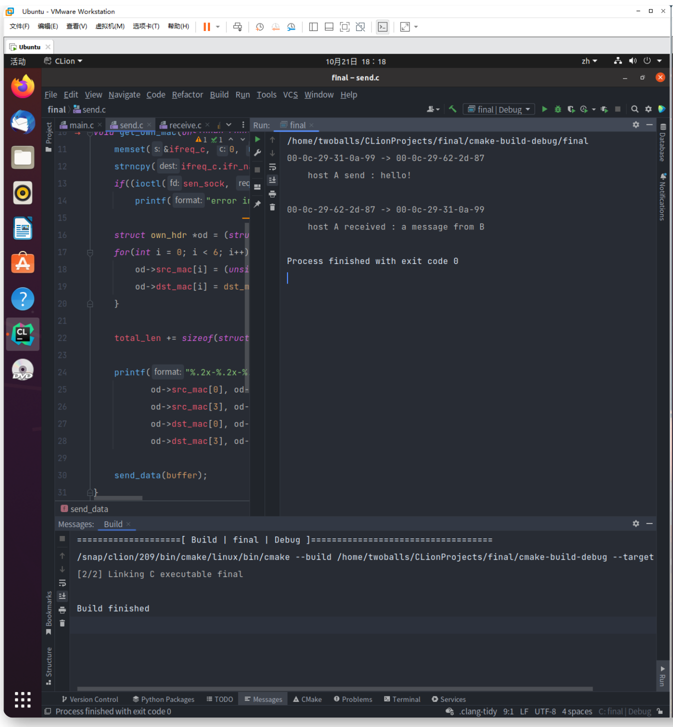
自定义了一个以太网头部格式如下，头部后面即要发送的数据

```c
struct own_hdr{
    unsigned char dst_mac[6];
    unsigned char src_mac[6];
};
```

分别定义send.c/receive.c用来实现发送和接收的方法

```c
// send
void get_eth_index();
void get_own_mac(unsigned char *buffer, unsigned char dst_mac[]);
void send_data(unsigned char *buffer);

// receive
void packet_print(unsigned char* buffer, int buffer_len);
void own_mac_header(unsigned char *buffer);
bool filterByMacAddress(unsigned char *buffer, unsigned char mac_addr[], int type);
bool equal(unsigned char *dst, unsigned char *src);
void get_data(unsigned char *buffer);
```

在主函数中，创建两个raw socket分别用来发送和接收数据包，主机A先发送，未收到主机B的回复之前不会再次发送，主机B同理。运行结果如下



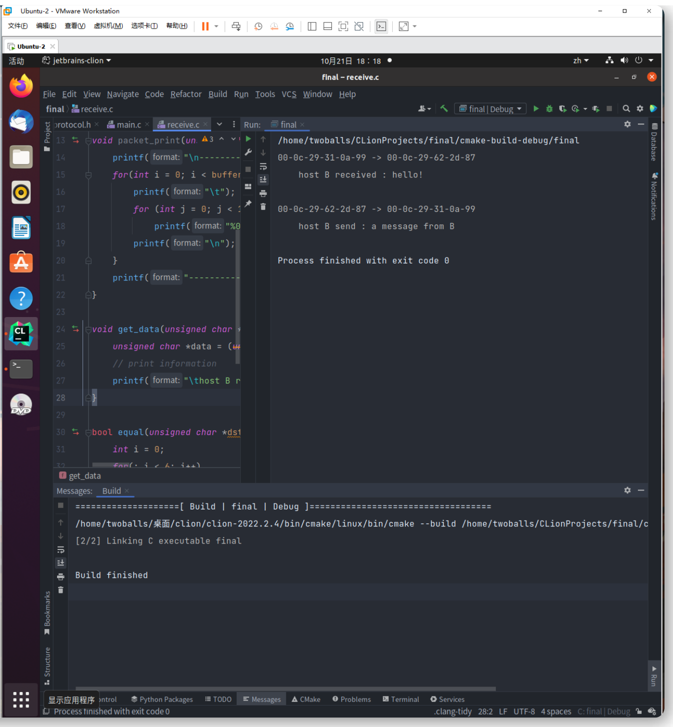
final – send.c

```
    memset( s: &ifreq_c,  c: 0,
    strncpy( dest: ifreq_c.ifr_n
    if((ioctl( fd: sen_sock,  req
        printf( format: "error ir

    struct own_hdr *od = (stru
    for(int i = 0; i < 6; i++)
        od->src_mac[i] = (unsi
        od->dst_mac[i] = dst_m
    }

    total_len += sizeof(struct

    printf( format: "%.2x-%.2x-%
           od->src_mac[0], od-
           od->src_mac[3], od-
           od->dst_mac[0], od-
           od->dst_mac[3], od-

    send_data(buffer);
}
```

```
/home/twoballs/CLionProjects/final/cmake-build-debug/final
00-0c-29-31-0a-99 -> 00-0c-29-62-2d-87
    host A send : hello!


00-0c-29-62-2d-87 -> 00-0c-29-31-0a-99
    host A received : a message from B

Process finished with exit code 0
```

```
====================[ Build | final | Debug ]==================================
/snap/clion/209/bin/cmake/linux/bin/cmake --build /home/twoballs/CLionProjects/final/cmake-build-debug --target
[2/2] Linking C executable final

Build finished
```

Process finished with exit code 0



final – receive.c

```
void packet_print(un
    printf( format: "\n--------
    for(int i = 0; i < buffer
        printf( format: "\t");
        for (int j = 0; j < 1
            printf( format: "%0
        printf( format: "\n");
    }
    printf( format: "---------
}

void get_data(unsigned char *
    unsigned char *data = (u
    // print information
    printf( format: "\thost B r
}

bool equal(unsigned char *dst
    int i = 0;
```

```
/home/twoballs/CLionProjects/final/cmake-build-debug/final
00-0c-29-31-0a-99 -> 00-0c-29-62-2d-87
    host B received : hello!


00-0c-29-62-2d-87 -> 00-0c-29-31-0a-99
    host B send : a message from B

Process finished with exit code 0
```

```
====================[ Build | final | Debug ]==================================
/home/twoballs/桌面/clion/clion-2022.2.4/bin/cmake/linux/bin/cmake --build /home/twoballs/CLionProjects/final/c
[2/2] Linking C executable final

Build finished
```

Process finished with exit code 0