ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN
Tel. (84-511) 736 949, Fax. (84-511) 842 771
Website: itf.dut.edu.vn, E-mail: cntt@dut.udn.vn

BÁO CÁO
ĐỒ ÁN CƠ SỞ NGÀNH MẠNG

Đề tài: **Xây dựng phần mềm giám sát truy cập mạng**

SINH VIÊN: **NGUYỄN HỮU HÙNG**
LỚP: **13T1**
MSSV: **102130014**
GVHD: **Ts. HUỲNH CÔNG PHÁP**

Đà Nẵng, 30/05/2017

LỜI MỞ ĐẦU

Nguyên lý hệ điều hành và **Lập trình mạng** là những nền tảng mà người lập trình phải hiểu rõ để có thể nắm bắt các kiến thức cao hơn.

Hệ điều hành là thành phần trung gian, là cầu nối cho sự giao tiếp giữa người dùng và máy tính. Thông qua hệ điều hành, người sử dụng dễ dàng làm việc và khai thác hiệu quả thiết bị phần cứng. Cũng như vậy, với sự phát triển trong lĩnh vực mạng hiện nay, thì việc nghiên cứu, nắm vững về Mạng là thực sự quan trọng đối với người lập trình.

Dựa trên những hiểu biết của mình và tham khảo thêm, em đã thực hiện, nghiên cứu về đề tài: Xây dựng phần mềm giám sát mạng.

Em xin chân thành cảm ơn thầy cô giáo trong khoa Công Nghệ Thông Tin, và hơn hết là thầy Huỳnh Công Pháp đã nhiệt tình theo dõi, hướng dẫn em nhiệt tình trong quá trình em thực hiện đề tài này.

Em đã rất cố gắng hoàn thành đề tài, nhưng không thể tránh khỏi những thiếu sót, mong nhận được sự góp ý của thầy cô để em hoàn thiện đề tài hơn.

Sinh viên thực hiện

Nguyễn Hữu Hùng

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI
"XÂY DỰNG PHẦN MỀM GIÁM SÁT TRUY CẬP MẠNG"

Ngành Công nghệ thông tin (CNTT) càng ngày càng phát triển đáp ứng hiệu quả mọi nhu cầu của các lĩnh vực trong xã hội. Cùng với sự phát triển không ngừng đó thì Mạng máy tính (Computer network)- môi trường làm việc thiết yếu hiện nay - cũng càng trở nên phức tạp và đa dạng. Từ các doanh nghiệp, công ty hay tổ chức lớn đến các dịch vụ công cộng đều xuất hiện bóng dáng của mạng máy tính. Chính vì sự phổ biến rộng rãi đó, việc quản lý, giám sát môi trường mạng để đảm bảo tính bảo mật, an toàn và lành mạnh quả là một việc không mấy đơn giản.

Trong phạm vi đề tài này, em xin được trình bày về ý tưởng xây dựng một phần mềm máy tính, cho phép giám sát gói tin mà mỗi máy tính trong cùng mạng LAN đã giao tiếp với các máy tính khác (kể cả trong cùng một mạng và ra mạng Internet). Qua việc giám sát gói tin trên giúp ta nắm được những thông tin của địa chỉ IP mà các máy đã truy cập, từ đó tiến hành chặn hay cho phép máy tiếp tục được truy cập. Thiết nghĩ đề tài này cũng rất hữu ích khi chúng ta muốn giám sát sự truy cập của nhân viên trong công ty, trong phòng thực hành, trong việc cho thuê dịch vụ Internet, hay ngay cả trong gia đình khi muốn kiểm soát con em đã truy cập vào những gì…

Để thực hiện được đề tài này, em xây dựng phần mềm trên môi trường hệ điều hành Linux, áp dụng hai mảng kiến thức đã được tích lũy, đó là môn **Nguyên lý hệ điều hành** và môn **Lập trình mạng** cùng những kĩ năng về lập trình, ngôn ngữ lập trình cần thiết.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

I KIẾN THỨC NGUYÊN LÝ HỆ ĐIỀU HÀNH

1. Hệ điều hành Linux

Linux là một hệ điều hành máy tính giống Unix (Unix-Like) được phát triển dựa vào mô hình "Phần mềm tự do" và việc phát triển phần mềm mã nguồn mở. Thành phần cơ bản của Linux là hạt nhân Linux (thường được gọi là nhân Linux – Linux kernel), là nhân hệ điều hành được phát triển bởi Linus Torvalds được công bố lần đầu tiên vào tháng 9 năm 1991 với phiên bản 0.01.

Sự khác nhau cơ bản giữa Linux và nhiều hệ điều hành phổ biến đương thời là nhân Linux và các thành phần khác đều là phần mềm tự do, mã nguồn mở.

Các bản phân phối của Linux được các nhà phát triển cho phép giao tiếp với các hệ điều hành khác và thành lập tiêu chuẩn tính toán. Hệ thống linux gắn chặt với các chuẩn POSIX, SUS, LSB, ISO và ANSI trong khả năng có thể.

- Các đặc tính của hệ điều hành:

- **Tính động** - Nghĩa là các phần mềm có thể làm việc trên các kiểu khác nhau của phần cứng theo cách giống nhau. Kernel và các chương trình ứng dụng hỗ trợ sự cài đặt của nó trên bất kỳ nền cứng nào.

- **Nguồn mở** - Mã nguồn Linux là có sẵn miễn phí và nó là sở hữu chung dựa trên dự án phát triển. Nhiều team làm việc trong sự cộng tác để tăng hiệu suất của Hệ điều hành Linux và nó tiếp tục tiến triển.

- **Đa người dùng** - Linux là một hệ thống đa người dùng, nghĩa là nhiều người sử dụng có thể truy cập vào các nguồn tài nguyên hệ thống như bộ nhớ/ram/các chương trình ứng dụng tại cùng một thời điểm.

- **Đa chương trình** - Linux là một hệ thống đa chương trình nghĩa là nhiều ứng dụng có thể chạy tại cùng một thời gian.
- **Hệ thống file có thứ bậc** - Linux cung cấp một cấu trúc file tiêu chuẩn trong đó các file hệ thống/các file người dùng được sắp xếp.
- **Shell** - Linux cung cấp một chương trình biên dịch đặc biệt mà có thể được sử dụng để thực hiện các lệnh của Hệ điều hành, thực hiện các kiểu đa dạng của các hoạt động, gọi các chương trình ứng dụng…
- **Bảo mật** - Linux cung cấp sự bảo vệ sử dụng các tính năng xác nhận như mật khẩu bảo vệ/sự truy cập được kiểm soát tới các file đặc trưng.

2. Lập trình C/C++ trên Linux

C/C++ là một ngôn ngữ rất phổ biến, rất cơ bản trong lập trình mà mỗi một lập trình viên ai cũng phải nắm rõ. Linux đã từ lâu hỗ trợ biên dịch C/C++ thông qua trình biên dịch GCC/G++ của GNU.

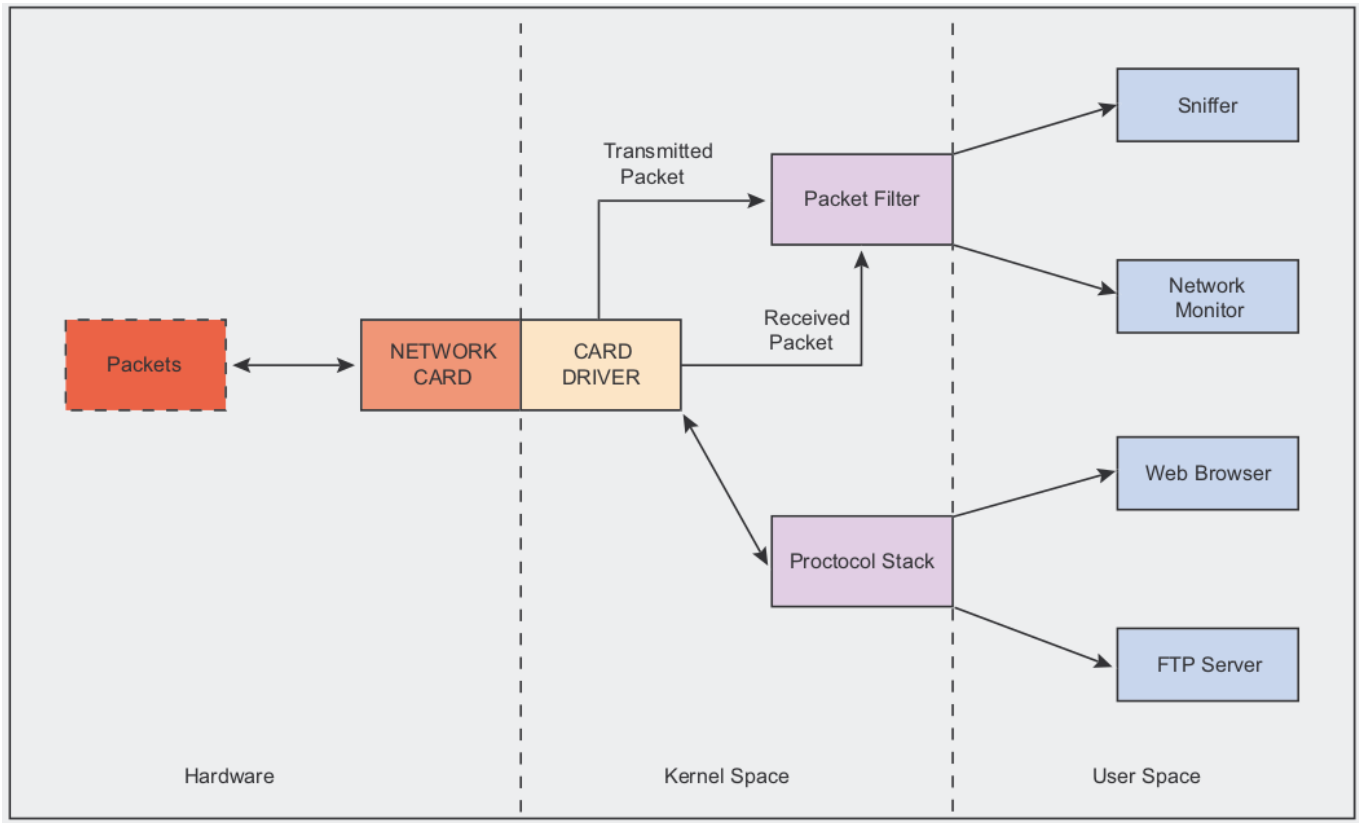
C/C++ có ưu điểm là rất phù hợp thực hiện những thao tác lập trình ở mức thấp trong hệ điều hành, là ngôn ngữ của đa nền tảng (Windows và Linux khác nhau về những thư viện hệ thống), ngoài ra C/C++ có tốc độ thực thi khá tốt và rất gọn nhẹ. Tuy nhiên việc thành thạo lập trình C/C++ cũng như C/C++ trên Linux là một quá trình khó khăn, ngôn ngữ này phức tạp trong việc cấp phát bộ nhớ động và thu hồi bộ nhớ, tạo một chương trình C/C++ "sạch đẹp", không để lại "rác" trong memory thật sự không phải ai cũng làm được.

3. Kĩ thuật hook và hook network:

Hook là kĩ thuật khá quen thuộc trong việc lập trình với bất kì một hệ điều hành nào, hook nói đến thao tác bắt hay chặn một sự kiện của người dùng tác động đến nhân của hệ điều hành, sau đó xử lý để lấy thông tin đầu vào. Chúng ta có nhiều loại kĩ thuật hook, có

thể ví dụ: mouse hook, keyboard hook, … hay ở đề tài này hook network sẽ được áp dụng.

Hook network ở đây cụ thể là chúng ta sao chép gói tin mà máy user giao tiếp với môi trường mạng trước khi nó được đưa vào kernel xử lý và hiển thị lên người dùng.



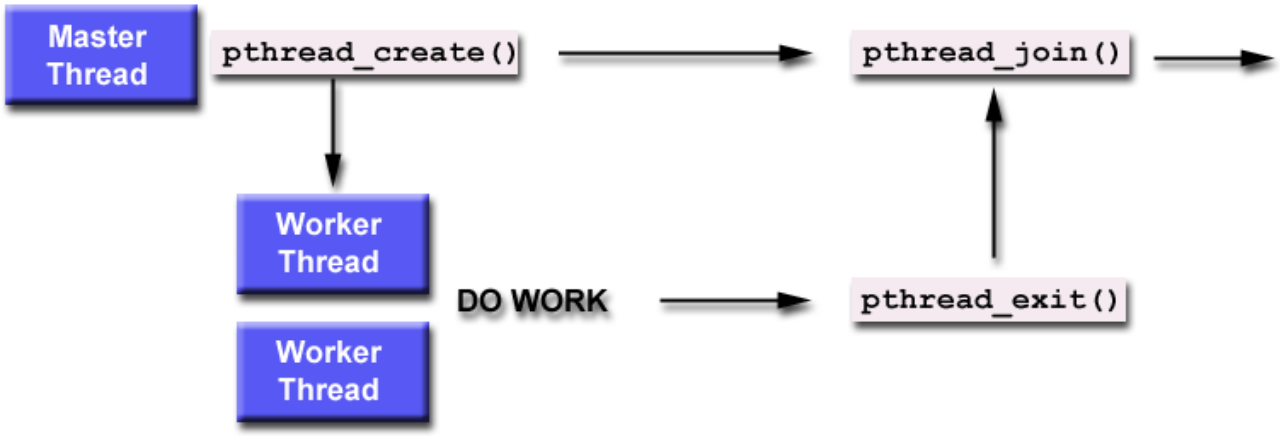
Hình 1: Nguyên lý bắt gói tin

Libpcap (Winpcap) là một thư viện hỗ trợ việc bắt gói tin rất phổ biến hiện nay, libpcap được sử dụng bởi phần mềm nổi tiếng về packet capture là **Wireshark,** và Wireshark cũng là một trong những tư liệu tham khảo của em. Để tìm kiếm và download thư viện này chúng ta có thể truy cập và xem hướng dẫn tại http://opensourceforu.com/2011/02/capturing-packets-c-program-libpcap/

4. Lập trình đa luồng trên Linux:

Một tiến trình phần mềm có thể có nhiều luồng, và trong đề tài này, việc thực hiện cùng lúc nhiều luồng là rất cần thiết. Hệ điều hành Linux có thư viện trong kernel là **pthread** hỗ trợ lập trình đa luồng trên ngôn ngữ C/C++. Để sử dụng thư viện này chúng ta có thể dùng lệnh biên dịch cùng tham số -lpthread. Các hàm thông dụng cần dùng đến như:

pthread_create(): tạo ra một tiến trình mới
pthread_join(): đợi một tiến trình khác kết thúc
pthread_exit(): kết thúc một tiến trình
pthread_cancel(): dùng để hủy tiến trình



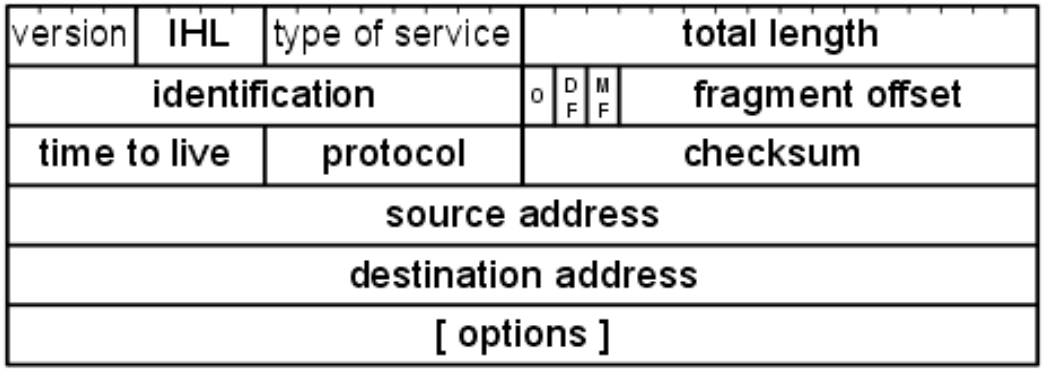
Hình 2: Đa tiến trình

II KIẾN THỨC LẬP TRÌNH MẠNG

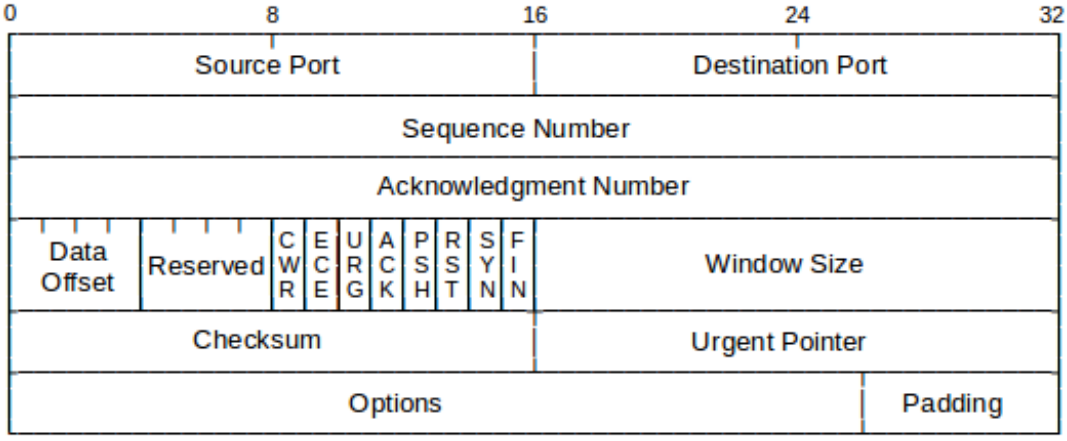
1. Cấu trúc các gói tin:

Trong lập trình mạng, việc nắm cấu trúc những gói tin và thứ tự các header là rất cần thiết. Các gói tin trong mạng máy tính phổ biến gồm có: gói tin IP, gói tin TCP, gói tin UDP, gói tin ICMP, gói tin IGMP,…

Sau đây là cấu trúc của những gói tin phổ biến trên đường truyền mạng, hỗ trợ việc phân tích một gói tin chính xác:



Hình 3: IP Header



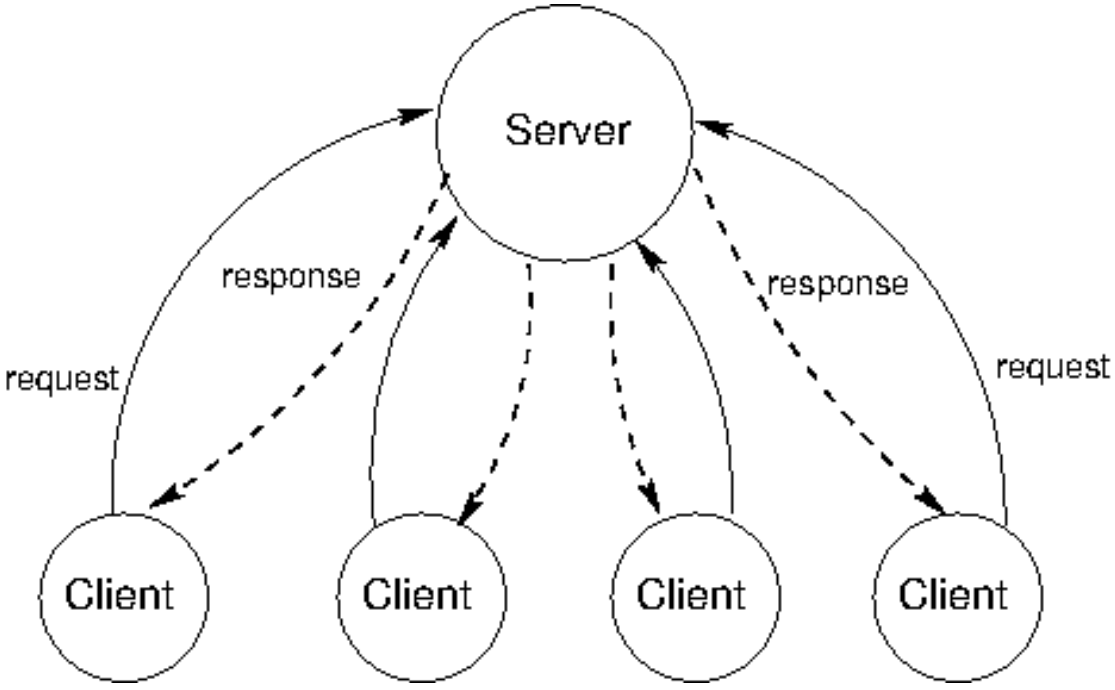
Hình 4: TCP Header

source port	destination port
length	checksum

Hình 5: UDP Header

2. Lập trình Socket Client-Server trên Java và C/C++:

Lập trình mô hình client-server thông qua cổng socket trên ngôn ngữ Java và C/C++ có bản chất như nhau, đều áp dụng mô hình xử lý trao đổi giữa các client và server.



Hình 6: Mô hình client-server

Tuy nhiên cú pháp trong java và C/C++ có sự khác nhau:

Java: **new ServerSocket(port)** để mở một cổng ở server

new Socket(host, port) để kết nối đến một cổng ở một server

.accept() để server chấp nhận kết nối đến của một client

C/C++:

bind(ser_soc, (struct sockaddr*)&ser_addr, sizeof(ser_addr));

để mở một cổng ở server

listen(ser_soc, max_connect);

để lắng nghe kết nối đến

connect (cli_sock, (struct sockaddr*)&ser_sock, sizeof(ser_sock));

để client kết nối đến cổng của server

CHƯƠNG 3: TRIỂN KHAI ĐỀ TÀI

I CHƯƠNG TRÌNH SERVER

1. Cơ chế hoạt động:

Chương trình server được chạy ở máy chủ giám sát, có nhiệm vụ ra lệnh (sms) đến chương trình client để lấy thông tin về các interface device của máy khách, điều khiển chặn hoặc cho phép máy khách truy cập vào địa chỉ ip nào đó.

Chường trình server được xây dựng bằng ngôn ngữ Java (thuận lợi cho việc thiết kế giao diện), gồm các tiến trình:

Tiến trình tiếp nhận kết nối từ các chương trình client:

Một khi các máy khách được bật lên, chương trình client sẽ được khởi động, mở cổng 9995 của máy khách, kết nối đến server. Đây chính là tiến trình có nhiệm vụ lắng nghe và tiếp nhận những kết nối đó

Tiến trình chính: ra lệnh (sms) gửi đến chương trình client đang chạy ngầm trên máy khách.

"iface" Yêu cầu danh sách các interface có thể bắt gói tin đang có trên máy khách

"start:devicename,filter" Yêu cầu client bắt đầu thực hiện bắt gói tin với tên card mạng devicename và bộ lọc gói tin filter

"stop" Yêu cầu client chấm dứt tiến trình bắt gói tin

"lock:ipaddress" Yêu cầu client chặn máy khách truy cập đến địa chỉ ip nào đó

"unlock:ipaddress" Yêu cầu client bỏ chặn máy khách truy cập đến địa chỉ ip nào đó

Tiến trình bắt gói tin gửi về từ chương trình client:

Tiến trình này luôn luôn tiếp nhận dữ liệu là những sms của client gửi về, trong đó đặc biệt là sms chứa thông tin của các gói tin bắt được. Sau khi nhận được gói tin bắt được, tiến trình này bắt đầu phân tích các header để thu thập địa chỉ MAC, địa chỉ IP, cổng Port...(dựa vào cấu trúc các gói tin đã trình bày) tạo thành một packet rõ ràng mà người quản lý có thể đọc được.

Tiến trình lấy thông tin thực của địa chỉ IP:

Nếu người quản trị viên kích hoạt tiến trình này, nó sẽ truy vấn đến trang web dịch vụ tìm ip http://ipfind.co với tham số là địa chỉ ip muốn tìm kiếm (http://ipfind.co?ip=<ipaddress>). Trang web này sẽ trả về những thông tin cần thiết gồm vị trí của địa chỉ ip (kinh độ vĩ độ, quốc gia, ngôn ngữ, múi giờ, …), tiến trình này sẽ hiển thị những thông tin lấy được lên giao diện.

Tiến trình lưu lại các gói tin bắt được xuống file:

Tiến trình này cho phép người quản lý có thể lưu lại dữ liệu bắt được thành file .txt, phục vụ cho công việc sau này, ví dụ như điều tra an ninh mạng.

2. Source code triển khai

Accept connect thread:

```
public void run(){
        while(true){
                try {
                        Host h;
                        h = new Host(serverSock.accept());
                        h.start();
                        listOfHosts.addElement(h);
                } catch (IOException e) {
                        e.printStackTrace();
                        break;
```

```
                }
        }
        for(int i= 0; i<listOfHosts.size();i++){
                listOfHosts.get(i).disconnect();
        }
}
```

Get packet thread:

```
public void run() {
        while (true) {
                byte[] buffer = new byte[BUFFER_SIZE];
                try {
                        int c = dis.read(buffer);
                        System.out.println("c ="+c);
                        byte[] receive= Arrays.copyOf(buffer, c);
                        if (c > 0) {
                                String tempStr = new String(receive);
                                String tempStrs[] = tempStr.split(":");
                                if (tempStrs[0].equals("iface")) {
                                        System.out.println(tempStr);
                        hostPanel.refreshIfaces(tempStrs[1].split(","));
                                } else if (tempStrs[0].equals("lock")) {
                                        if (tempStrs[1].equals("ok")) {

        JOptionPane.showMessageDialog(Main.mainView,"Locked successfully !", "IP
Locking",JOptionPane.INFORMATION_MESSAGE);
                                        } else {
        JOptionPane.showMessageDialog(Main.mainView,"IP already locked !","IP
Locking",JOptionPane.ERROR_MESSAGE);
```

```
                                        }
                                    }else if(tempStrs[0].equals("unlock")){
                                        if (tempStrs[1].equals("ok")) {
JOptionPane.showMessageDialog(Main.mainView,"Unlocked successfully !", "IP
Unlocking",JOptionPane.INFORMATION_MESSAGE);
                                        } else {
JOptionPane.showMessageDialog(Main.mainView,"IP already unlocked !", "IP
Unlocking",JOptionPane.ERROR_MESSAGE);
                                        }
                                    }
                                    else {
                                        System.out.println(receive.length);
                                        Packet p = new Packet(receive);
                                        packetTableModel.addPacket(p);
                                visitedIPTableModel.addIP(p.getIp().getSourceIP());
                                visitedIPTableModel.addIP(p.getIp().getDestIP());
                                        }
                                    }else if (c < 0)
                                        throw new IOException();
                                } catch (Exception e) {
                                        disconnect();
                                        e.printStackTrace();
                                        break;
                                }
                        }
                }
```

Send sms:

```
        public void sendRequest(String sms) {
```

```
            try {
                dos.write(sms.getBytes());
            } catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
```

Packet object:

```
public class Packet {
    public static int MAC_INDEX = 0;
    public static int IP_INDEX = 14;

    public enum PACKET_TYPE {
        OTHER, TCP, UDP, ICMP
    };

    private PACKET_TYPE type;

    private MACHeader mac;
    private IPHeader ip;

    private Header protocolHeader;

    private byte[] packetPayload;

    public Packet(byte[] packet) {
        mac = new MACHeader(packet, Packet.MAC_INDEX);
        ip = new IPHeader(packet, Packet.IP_INDEX);
```

```
            type = PACKET_TYPE.OTHER;
            int protoLen = 0;
            switch (ip.getProtocol()) {
            case 6:
                protocolHeader = new TCPHeader(packet, Packet.IP_INDEX
                        + ip.getHeaderLen());
                protoLen = ((TCPHeader) protocolHeader).getOff();
                this.type = PACKET_TYPE.TCP;
                break;
            case 17:
                protocolHeader = new UDPHeader(packet, Packet.IP_INDEX
                        + ip.getHeaderLen());
                protoLen = 8;
                this.type = PACKET_TYPE.UDP;
                break;
            case 1:
                protocolHeader = new ICMPHeader(packet, Packet.IP_INDEX
                        + ip.getHeaderLen());
                protoLen = 4;
                this.type = PACKET_TYPE.ICMP;
                break;
            }
            packetPayload = new byte[ip.getTotalLen()-ip.getHeaderLen()-protoLen];
            for (int i = 0; i < packetPayload.length; i++) {
                packetPayload[i] = packet[Packet.IP_INDEX + ip.getHeaderLen()
                        + protoLen + i];
            }
        }}
```

II CHƯƠNG TRÌNH CLIENT

1. Cơ chế hoạt động:

Chương trình client được xây dựng bằng ngôn ngữ C/C++ cho phép chạy ngầm trên các máy khách, mỗi khi máy khách được khởi động thì đồng nghĩa chương trình client cũng khởi động và làm nhiệm vụ của mình.

Chương trình client gồm có hai tiến trình:

Tiến trình chính

Tiến trình này có nhiệm vụ tiếp nhận lệnh của server đồng thời theo lệnh đó điều khiển tiến trình bắt gói tin sau đó trả kết quả về server, tiến trình bắt gói tin có nhiệm vụ luôn luôn bắt và sao chép gói tin từ card mạng của máy khách và gửi đến máy server giám sát.

Những message mà client trả về server:

"ifaces: devicename1, devicename2, ..." danh sách các interface device có thể bắt gói tin của máy khách

"**start: ok**" bắt đầu bắt gói tin thành công

"start: fail" bắt đầu bắt gói tin thất bại

"stop: ok" dừng bắt gói tin thành công

"stop: fail" dừng bắt gói tin thất bại

"lock: ok" chặn địa chỉ ip thành công

"lock: fail" chặn địa chỉ ip thất bại

"unlock: ok" bỏ chặn địa chỉ ip thành công

"unlock: fail" bỏ chặn địa chỉ ip thất bại

"packet" nội dung packet bắt được

Tiến trình bắt gói tin

Tiến trình này sử dụng thư viện libpcap thực hiện hàm **pcap_loop(descr, -1, capture_callback, NULL);**

để lặp đi lặp lại quá trình dò bắt các gói tin đi và đến. Trong đó capture_callback là hàm sử lý gói tín bắt được (cụ thể là gửi về server giám sát), descr là đối tượng device đã được mở sẵn bằng hàm pcap_open() để chờ gói tin, tham số thứ 2 là số gói tin cần bắt, nếu là -1 thì quá trình sẽ thực hiện bắt vô hạn.

2. Source code triển khai:

Main thread:

```
void *controller_thread(void *par){

	char readbuffer[BUFF_SIZE];
	char writebuffer[SMS_SIZE];
	int cliSoc = CLIENT->GetSock();
	while (1)
	{
		if (CLIENT->Receive(readbuffer, sizeof(readbuffer)) < 0)
		{
			perror("[ERROR] Read from server fail !");
			continue;
		}
		char *check = strtok(readbuffer, ":");

		//Yeu cau danh sach cac network interfaces;
		if (strcmp(check, "iface") == 0)
		{
			memset(writebuffer, 0, SMS_SIZE);
			strcpy(writebuffer, "iface:");
			CLIENT->GetIfaceNames(writebuffer);
			if (CLIENT->Send(writebuffer, strlen(writebuffer)) < 0)
			{
				perror("[ERROR] Write to server fail !");
				continue;
			}
		}
		else if (strcmp(check, "stop") == 0)
		{
			CLIENT->StopCapture();
```

```
}
else if (strcmp(check, "start") == 0)
{
	int devIndex = atoi(strtok(strtok(NULL, ":"), ","));
	char *filterStr = strtok(NULL, ",");
	CLIENT->StartCapture(devIndex, filterStr);
}
else if (strcmp(check, "lock") == 0)
{
	char *lockIp = strtok(NULL, ":");
	memset(writebuffer, 0, SMS_SIZE);
	if (Lock(lockIp))
	{
		strcpy(writebuffer, "lock:ok");
		if (CLIENT->Send(writebuffer, strlen(writebuffer)) < 0)
		{
			perror("[ERROR] Write to server fail !");
			continue;
		}
	}else{
		strcpy(writebuffer, "lock:fail");
		if (CLIENT->Send(writebuffer, strlen(writebuffer)) < 0)
		{
			perror("[ERROR] Write to server fail !");
			continue;
		}
	}
}else if(strcmp(check, "unlock") == 0){
	char *unlockIp = strtok(NULL, ":");
	memset(writebuffer, 0, SMS_SIZE);
	if (Unlock(unlockIp))
	{
		strcpy(writebuffer, "unlock:ok");
		if (CLIENT->Send(writebuffer, strlen(writebuffer)) < 0)
		{
			perror("[ERROR] Write to server fail !");
			continue;
		}
	}else{
		strcpy(writebuffer, "unlock:fail");
		if (CLIENT->Send(writebuffer, strlen(writebuffer)) < 0)
		{
			perror("[ERROR] Write to server fail !");
			continue;
		}
	}
```

```
                }
        }
}
```

Capture thread:

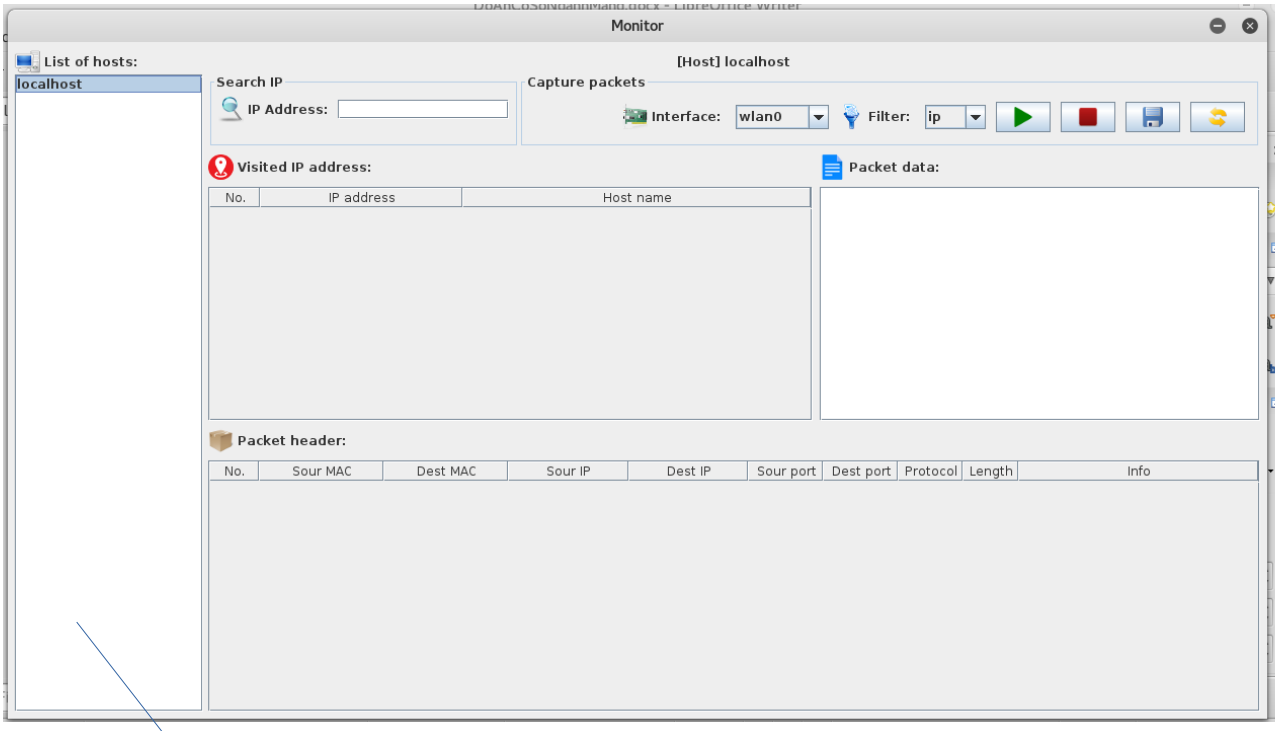
```
void *capture_thread(void *par){

        char errbuff[ERR_SIZE];
        int devIndex = CLIENT->GetDevIndex();
        const char *filterStr = CLIENT->GetFilterStr();
        int cliSoc = MonitorClient::GetInstance()->GetSock();

        printf("%d\n", devIndex);
        pcap_if_t *dev = NULL;
        pcap_t *descr;
        struct bpf_program fp;
        bpf_u_int32 maskp; /* subnet mask */
        bpf_u_int32 netp; /* ip */
        int i;
        for (i = 0, dev = CLIENT->GetDevs(); (i < devIndex) && dev; dev = dev->next, i+
+);

        pcap_lookupnet(dev->name, &netp, &maskp, errbuff);

        // Kich hoat device de bat goi tin
        descr = pcap_open_live(dev->name, BUFSIZ, 1, -1, errbuff);

        if (descr == NULL)
        {
                printf("pcap_open_live(): %s\n", errbuff);
                return NULL;
        }

        // Dich bo loc
        if (pcap_compile(descr, &fp, filterStr, 0, netp) == -1)
        {
                fprintf(stderr, "Error calling pcap_compile\n");
                return NULL;
        }

        // Gan bo loc cho device
        if (pcap_setfilter(descr, &fp) == -1)
        {
                fprintf(stderr, "Error setting filter\n");
                return NULL;
        }
```

```
	// Bat goi tin
	pcap_loop(descr, -1, capture_callback, NULL);
}

void capture_callback(u_char *args, const struct pcap_pkthdr *pkthdr, const u_char
*packet)
{
	int cliSoc = MonitorClient::GetInstance()->GetSock();
	u_char *temp = (u_char *)packet;
	struct iphdr *iph = (struct iphdr *)(temp + 14);
	write(cliSoc, temp, ntohs(iph->tot_len) + 14);
}
```

CHƯƠNG 4: KẾT QUẢ TRIỂN KHAI

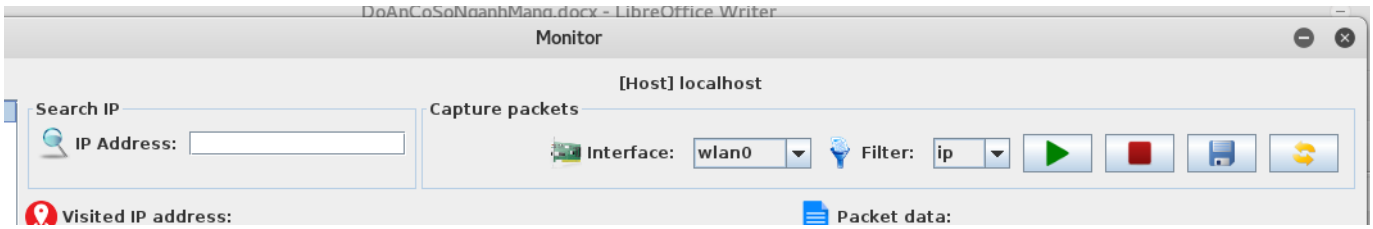
I CHƯƠNG TRÌNH SERVER

Hình 7: Giao diện chính

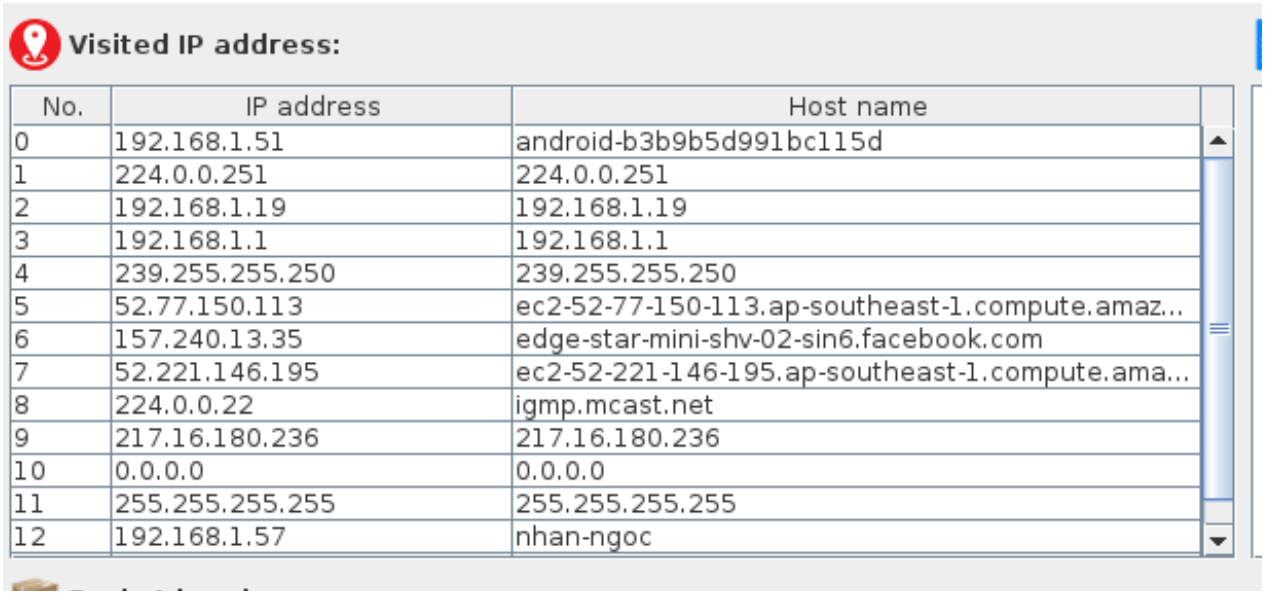


Danh sách các máy khách

Hình 8: Khung điều khiển chương trình client



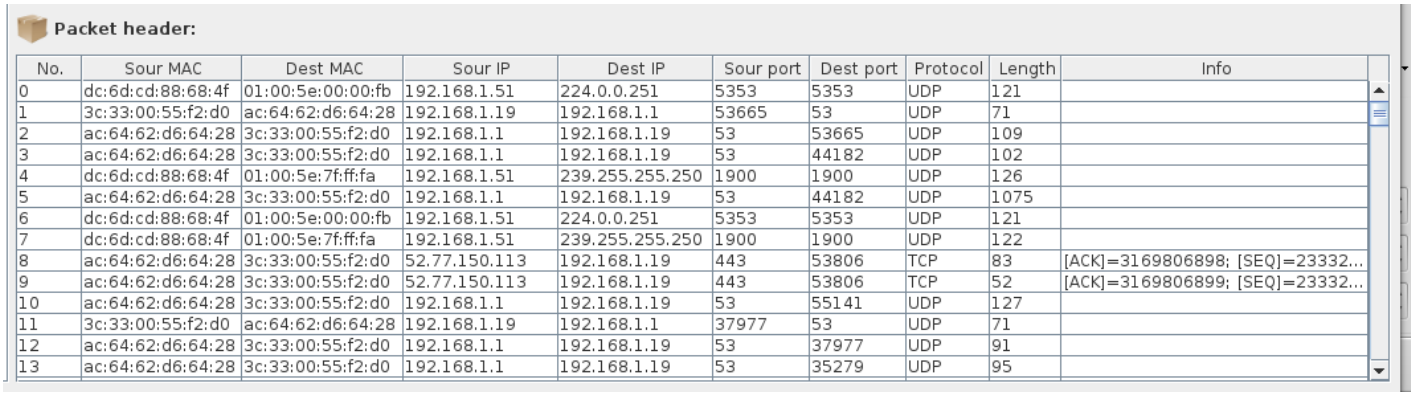
Hình 9: Danh sách các địa chỉ ip mà máy khách đã giao tiếp



Visited IP address:

No.	IP address	Host name
0	192.168.1.51	android-b3b9b5d991bc115d
1	224.0.0.251	224.0.0.251
2	192.168.1.19	192.168.1.19
3	192.168.1.1	192.168.1.1
4	239.255.255.250	239.255.255.250
5	52.77.150.113	ec2-52-77-150-113.ap-southeast-1.compute.amaz...
6	157.240.13.35	edge-star-mini-shv-02-sin6.facebook.com
7	52.221.146.195	ec2-52-221-146-195.ap-southeast-1.compute.ama...
8	224.0.0.22	igmp.mcast.net
9	217.16.180.236	217.16.180.236
10	0.0.0.0	0.0.0.0
11	255.255.255.255	255.255.255.255
12	192.168.1.57	nhan-ngoc

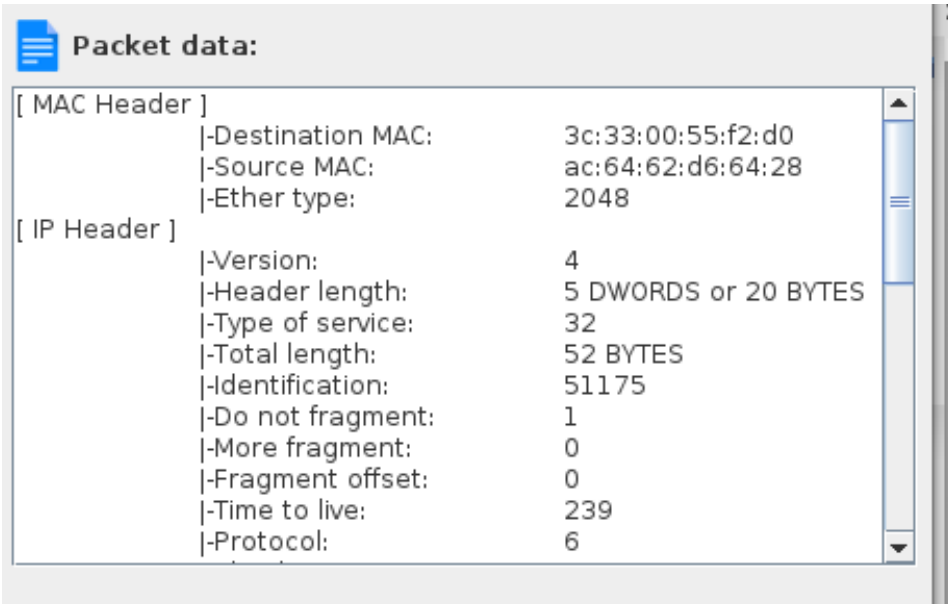
Hình 10: Các gói tin bắt được



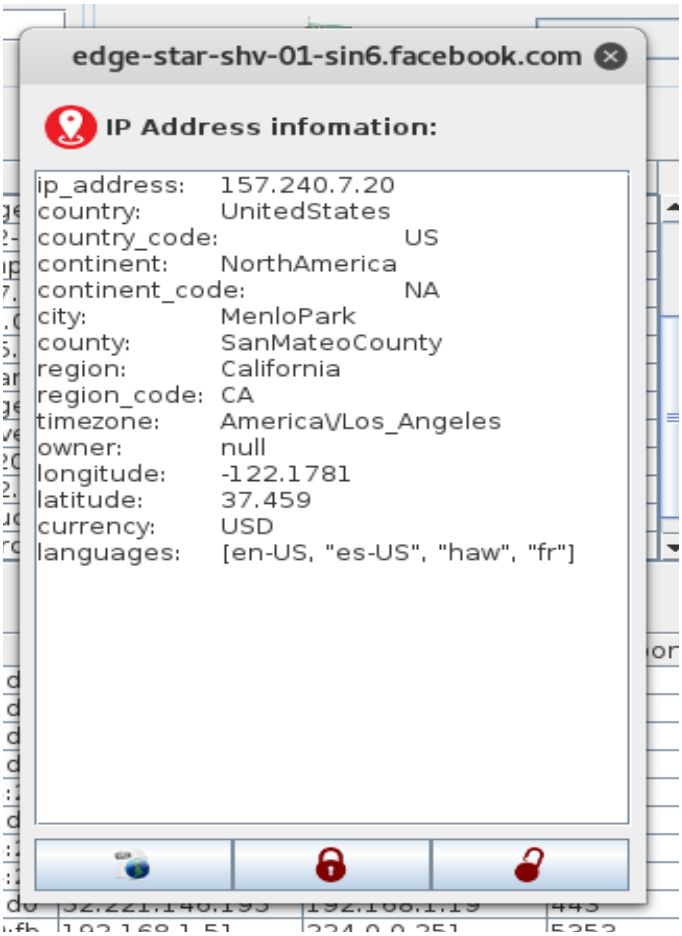
Packet header:

No.	Sour MAC	Dest MAC	Sour IP	Dest IP	Sour port	Dest port	Protocol	Length	Info
0	dc:6d:cd:88:68:4f	01:00:5e:00:00:fb	192.168.1.51	224.0.0.251	5353	5353	UDP	121	
1	3c:33:00:55:f2:d0	ac:64:62:d6:64:28	192.168.1.19	192.168.1.1	53665	53	UDP	71	
2	ac:64:62:d6:64:28	3c:33:00:55:f2:d0	192.168.1.1	192.168.1.19	53	53665	UDP	109	
3	ac:64:62:d6:64:28	3c:33:00:55:f2:d0	192.168.1.1	192.168.1.19	53	44182	UDP	102	
4	dc:6d:cd:88:68:4f	01:00:5e:7f:ff:fa	192.168.1.51	239.255.255.250	1900	1900	UDP	126	
5	ac:64:62:d6:64:28	3c:33:00:55:f2:d0	192.168.1.1	192.168.1.19	53	44182	UDP	1075	
6	dc:6d:cd:88:68:4f	01:00:5e:00:00:fb	192.168.1.51	224.0.0.251	5353	5353	UDP	121	
7	dc:6d:cd:88:68:4f	01:00:5e:7f:ff:fa	192.168.1.51	239.255.255.250	1900	1900	UDP	122	
8	ac:64:62:d6:64:28	3c:33:00:55:f2:d0	52.77.150.113	192.168.1.19	443	53806	TCP	83	[ACK]=3169806898; [SEQ]=23332...
9	ac:64:62:d6:64:28	3c:33:00:55:f2:d0	52.77.150.113	192.168.1.19	443	53806	TCP	52	[ACK]=3169806899; [SEQ]=23332...
10	ac:64:62:d6:64:28	3c:33:00:55:f2:d0	192.168.1.1	192.168.1.19	53	55141	UDP	127	
11	3c:33:00:55:f2:d0	ac:64:62:d6:64:28	192.168.1.19	192.168.1.1	37977	53	UDP	71	
12	ac:64:62:d6:64:28	3c:33:00:55:f2:d0	192.168.1.1	192.168.1.19	53	37977	UDP	91	
13	ac:64:62:d6:64:28	3c:33:00:55:f2:d0	192.168.1.1	192.168.1.19	53	35279	UDP	95	

Hình 11: Chi tiết từng gói tin



Hình 12: Thông tin của địa chỉ ip

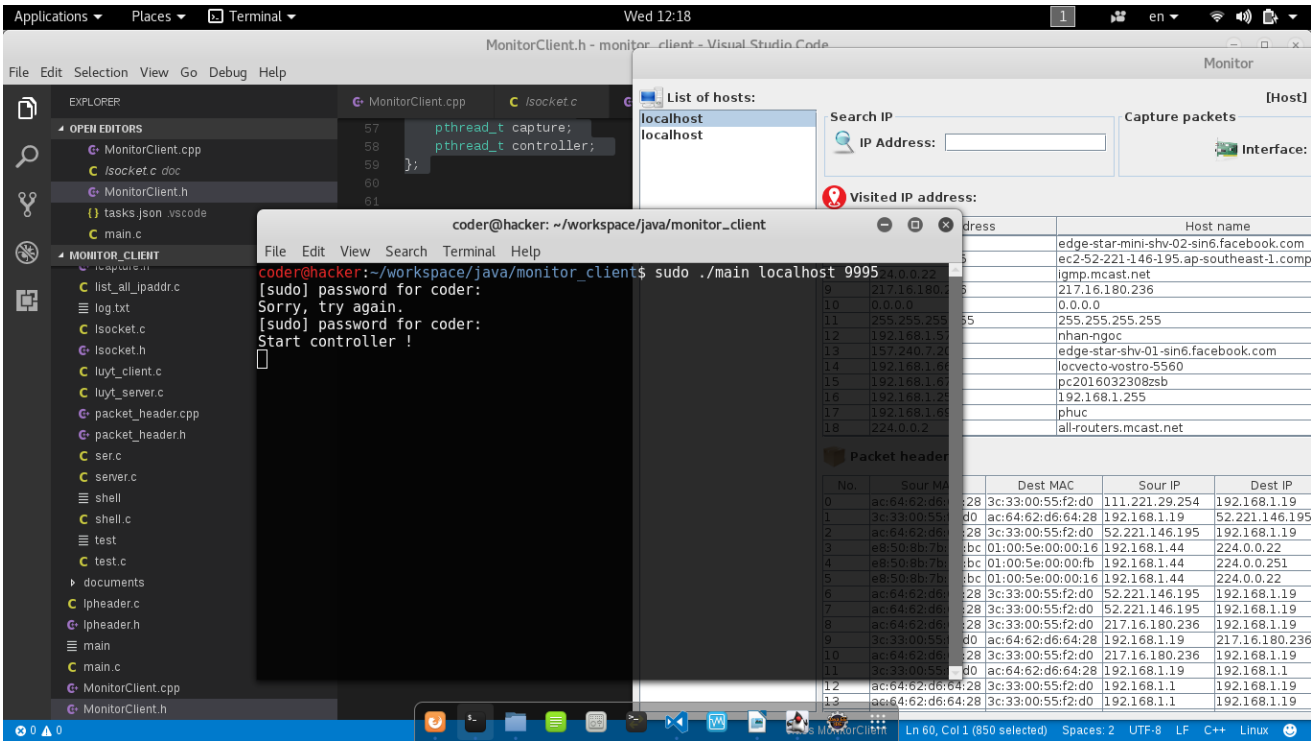


II CHƯƠNG TRÌNH CLIENT

Đối tượng client chạy ngầm trên máy khách

```
class MonitorClient{

        public:
                static MonitorClient *GetInstance();
                static void DestroyInstance();

                int GetSock();
                void SetFilterStr(const char *filterStr);
                char *GetFilterStr();
                void SetDevIndex(int devIndex);
                int GetDevIndex();
                pcap_if_t *GetDevs();

                int Connect(const char *host,const int port);
                int Send(const char *buffer, const int size);
                int Receive(char *buffer, const int size);
                void StartController();
                void StopController();
                void StartCapture(int devIndex, const char *filterStr);
                void StopCapture();
                int GetIfaceNames(char *ifaceNames);
                void Close();

                static MonitorClient *instance;
        private:
                ~MonitorClient();
                MonitorClient();
                int cliSoc;
                int devIndex;
                char *filterStr;
                pcap_if_t *devs;
                pthread_t capture;
                pthread_t controller;
};
```

Hình 14: Thực thi chương trình client trên một máy ảo



CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Bằng cách áp dụng những kiến thức về Nguyên lý hệ điều hành và Lập trình mạng, em đã xây dựng được một ứng dụng phục vụ cho việc quản lý hiệu quả quá trình truy cập của các máy tính trong mạng LAN - một mô hình khá phổ biến trong nhiều công ty, doanh nghiệp, trường học, …

Cũng qua việc xây dựng một phần mềm như thế này, em đã có thể tiếp xúc những trường hợp thực tế xảy ra trong kĩ thuật lập trình với hệ điều hành Linux cũng như có cơ hội để hiểu thêm về nguyên lý hoạt động của đường truyền, cấu trúc của gói tin như thế nào.

Phần mềm giám sát mạng này có thể được tiếp tục phát triển để áp dụng cho mạng Internet, mạng diện rộng bằng cách thông qua một máy chủ có địa chỉ IP public. Có thể phát triển thêm nhiều chức năng như hiển thị vị trí địa chỉ IP trên Google Map, chức năng cảnh báo cho người dùng máy khách đang truy cập nội dung không hợp lệ …

--- Kết thúc ---