

**TRƯỜNG ĐẠI HỌC CẦN THƠ    CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM  
KHOA CÔNG NGHỆ THÔNG TIN & TT    Độc lập – Tự do – Hạnh phúc**

**ĐỀ CƯƠNG CHI TIẾT**

**Môn học: Thực hành Mạng máy tính**

**Mã môn học: CT112, số tín chỉ: 3**

**Học kỳ áp dụng: Học kỳ II, năm học 2016-2017**

**Số tiết: 30 tiết (5 buổi), thi Thực hành vào buổi thứ 6**

**A. NỘI DUNG VÀ MỤC TIÊU MÔN HỌC**

- Phần thực hành Mạng máy tính hướng đến mục tiêu xây dựng các bài tập thực hành theo từng chủ đề liên quan đến các kiến thức đã được giới thiệu trong phần Lý thuyết. Thông qua các bài tập này, sinh viên thiết kế và phân tích được các hệ thống mạng khác nhau, các giao thức khác nhau, cách thức truyền tải dữ liệu khác nhau....
- Phần thực hành Mạng máy tính chia làm 5 buổi với nội dung các buổi như sau:
  - o Buổi 1: Làm quen Netkit Emulator, công cụ xây dựng hệ thống mạng ảo. Làm quen Wireshark, công cụ bắt gói tin và phân tích dữ liệu trên gói tin bắt được.
  - o Buổi 2: Xây dựng hệ thống mạng minh họa cho giao thức ARP kết hợp với sử dụng Wireshark để phân tích gói dữ liệu.
  - o Buổi 3: Xây dựng hệ thống mạng với giải thuật vạch đường nội miền (IGP – Internal Gateway Protocol) RIPv2 và OSPFv2. Phân tích dữ liệu với Wireshark để nhận biết khuôn dạng dữ liệu trao đổi, cập nhật bảng vạch đường của các router...
  - o Buổi 4: Sử dụng Wireshark để làm rõ giao thức bắt tay 3 chiều của TCP và UDP trên tầng vận chuyển. Khảo sát cơ chế điều khiển thông lượng.
  - o Buổi 5: Xây dựng hệ thống mạng minh họa cho hệ thống phân giải tên miền DNS và WebMail server.

**B. PHƯƠNG PHÁP GIẢNG DẠY**

Sinh viên cần đọc trước nội dung của buổi thực hành và các kiến thức liên quan.

Sinh viên cần hoàn thành các bài tập về nhà của buổi thực hành.

Sinh viên chuẩn bị 1 USB để lưu trữ bài thực hành vào cuối buổi.

Sinh viên vắng thực hành 1 buổi sẽ bị CẤM THI phần thực hành.

Sinh viên nên thực hành trước ở nhà và đặt câu hỏi cho giáo viên khi lên lớp.

**C. TÀI LIỆU THAM KHẢO**

1. [Ngô Bá Hùng, Phạm Thế Phi], Giáo trình Mạng máy tính, NXB Đại học Cần Thơ, 2014.
2. [J.F.Kurose, K.W.Ross], Supplements: Wireshark Labs – Computer Networking: A Top to Down Approach 6<sup>th</sup>, Protocols and Practice, Saylor Foundation, 2014
3. [Netkit Community], Labs Official: [http://wiki.netkit.org/index.php/Labs\\_Official](http://wiki.netkit.org/index.php/Labs_Official)
4. [Massimo Rimondini], Emulating Computer Networks with Netkit, 4<sup>th</sup> International Workshop on Internet Performance, Simulation, Monitoring and Measurement.

## ***MỤC LỤC***

<b>BUỔI THỰC HÀNH 1</b>	<b>1</b>
I. Giới thiệu các công cụ mạng phổ biến trên Linux	1
II. Phần mềm Netkit	1
1. Các đặc điểm chính của Netkit	1
2. Tổ chức thư mục của Netkit	2
3. Hệ thống tập lệnh của Netkit	2
4. Cài đặt Netkit trong môi trường Linux	3
5. Xây dựng mô hình mạng với Netkit	4
III. Bài tập thực hành với Netkit	4
I. Phần mềm Wireshark	7
1. Giới thiệu	7
1. Giao diện tương tác với Wireshark	7
2. Bài tập thực hành với Wireshark	8
<b>BUỔI THỰC HÀNH 2</b>	<b>9</b>
I. Ethernet II và Wifi 802.11 frame	9
II. Khảo sát ARP	12
<b>BUỔI THỰC HÀNH 3</b>	<b>15</b>
I. Giới thiệu	15
II. Bài tập thực hành	15
<b>BUỔI THỰC HÀNH 4</b>	<b>25</b>
I. Bài tập thực hành UDP	25
III. Bài tập thực hành TCP	25
<b>BUỔI THỰC HÀNH 5</b>	<b>27</b>
II. Bài tập thực hành	27

## BUỔI THỰC HÀNH 1

### **Mục đích:**

- Giới thiệu một số công cụ mạng tích hợp trên nền tảng Linux
- Giới thiệu phần mềm Netkit và Wireshark.
- Tìm hiểu cách thức cài đặt Netkit, tổ chức thư mục cho mô hình mạng trong Netkit.
- Xây dựng một số mô hình mạng đơn giản với Netkit.
- Tìm hiểu các tính năng chính của Wireshark sử dụng trong nội dung thực hành.
- Thực hành đơn giản phân tích gói dữ liệu mà Wireshark ghi nhận được.

### **I. Giới thiệu các công cụ mạng phổ biến trên Linux**

**ping:** công cụ cho phép gửi 1 gói tin đến từ địa chỉ IP máy nguồn đến địa chỉ IP máy đích. Nếu như địa chỉ IP máy đích là tồn tại, ping trên máy đích sẽ tự động hồi đáp bằng 1 gói tin ngược lại máy nguồn. Cả 2 gói tin ping này đều chứa thông điệp ICMP - Internet Control Message Protocol.

**ifconfig:** công cụ cho phép cấu hình giao diện mạng (network interface) của máy, ví dụ: đặt địa chỉ IP và netmask, tắt hoặc mở giao diện mạng.

**netstat:** công cụ hiển thị các kết nối mạng, bảng vạch đường và các thông số trên giao diện mạng. Dùng công cụ netstat để tìm các vấn đề liên quan đến kết nối mạng và xác định giao thông trên mạng.

**tcpdump:** công cụ cho phép bắt các gói tin luân chuyển qua một hoặc nhiều giao diện mạng. Công cụ này cung cấp 2 chức năng lớn, là packet sniffing và packet analyze với thư viện lệnh phong phú.

**route:** công cụ cho phép xem bảng dẫn đường hiện tại của host.

**traceroute:** công cụ cho phép lần vết của dữ liệu luân chuyển qua host.

**bind:** công cụ phục vụ cho việc cấu hình DNS, hoạt động trên nhiều nền tảng hệ điều hành khác nhau như AIX/BSD/Unix/Linux...

### **II. Phần mềm Netkit**

Netkit là một bộ công cụ mã nguồn mở trên nền tảng Linux cho phép người dùng giả lập (emulate) các hệ thống mạng phục vụ cho nhiều mục đích sử dụng khác nhau như từ đơn giản cho đến phức tạp. Giống như Packet Tracer hay GNS3, Netkit hỗ trợ mô phỏng một hệ thống mạng từ đơn giản đến phức tạp với các gói thư viện cho phép tạo ra các thiết bị mạng như Router, Switch, PC...

**Netkit có 1 ưu điểm rất lớn là việc xây dựng máy ảo trên Linux kernel cho phép người dùng có thể hack và modify lại thành 1 máy ảo phù hợp với mô hình mạng cần xây dựng.** Ngoài ra trong các đánh giá về hiệu năng hoạt động của các Network Emulator, Netkit đạt được đánh giá tốt theo nhiều tiêu chí khác nhau. Trong thực hành Mạng máy tính, ta sử dụng Netkit để mô phỏng mô hình mạng nhằm minh họa kiến thức lý thuyết đã được giảng dạy.

#### **1. Các đặc điểm chính của Netkit**

- Hoạt động dựa trên *User Mode Linux* (1 Linux kernel thực thi như 1 tiến trình người dùng trên hệ điều hành Linux).
- **Một tiến trình User Mode Linux được gọi là máy ảo (Virtual Machine - vm).** Các máy ảo này có thể tương tác như hệ điều hành Linux. Ví dụ: dùng lệnh *ls -l* liệt kê tập tin, lệnh *ps* để liệt kê các

tiền trình...Máy ảo có thể kết nối internet để cài đặt các gói thư viện cần thiết như: gcc, g++, libc6-dev.

- **Linux chứa các máy ảo đó được gọi là Host Machine – hm**. Như vậy, trên cùng 1 Host Machine, có thể có nhiều các Virtual Machine hoạt động cùng lúc.
- Các máy ảo được mô phỏng như một thành phần trong mô hình mạng thông qua thiết lập cấu hình, chẳng hạn: cấu hình máy ảo làm router hoặc làm PC.
- Các máy ảo nhận lệnh từ người dùng qua terminal. Bộ nhớ máy ảo được cấp phát từ tài nguyên của máy host.
- Các máy ảo kết nối lại với nhau tạo thành một *collision domain*.
- Việc khởi tạo, cấu hình, thiết lập kết nối giữa các máy ảo khác nhau gọi là quá trình mô phỏng hoạt động của một mạng máy tính.

## 2. Tổ chức thư mục của Netkit

Một hệ thống mạng máy tính tạo bởi Netkit có cấu trúc thư mục được tổ chức như sau:

- Một file **lab.conf** mô tả hình thái (topology) của mạng. Các thiết lập dành cho các máy ảo cũng được miêu tả trong file này, ví dụ: LAB\_DESCRIPTION, LAB\_VERSION...
- Một thư mục con chứa các file cấu hình cho từng thiết bị giả lập.
- File **.startup** và file **.shutdown** mô tả chuỗi hành động được thực hiện bởi máy ảo khi khởi động hoặc tắt. Trong đó:
  - o **shared.startup** và **shared.shutdown** ảnh hưởng đến toàn bộ máy ảo.
  - o **vm\_name.startup** và **vm\_name.shutdown** chỉ ảnh hưởng đến một máy ảo cụ thể.
- Một file **lab.dep** để mô tả quan hệ các máy ảo khi khởi động hệ thống mạng. Ví dụ: pc3 chỉ khởi động sau khi pc1 và pc2 đã khởi động thành công.

## 3. Hệ thống tập lệnh của Netkit

- Netkit cung cấp 2 tập lệnh với 2 tiếp đầu ngữ khác nhau: **v-commands** và **l-commands**. 2 tập lệnh này được sử dụng trên terminal của máy host.
- **v-commands** sử dụng để tương tác với một máy ảo đơn lẻ.
- **l-commands** sử dụng để tương tác với một mô hình mạng gồm nhiều máy ảo tham gia.
- Các lệnh phổ biến nhóm **v-commands**
  - o **vstart**: khởi tạo một máy ảo mới. Cấu trúc lệnh: **vstart [option] machine\_name**, trong đó option có thể là: ethN (collision domain), -M (dung lượng bộ nhớ)... Ví dụ: **vstart --eth0=A --eth1=B -M 256 mayao1**
  - o **vhalt**: dừng hoạt động (shutdown) máy ảo.
  - o **vrash**: hủy đi máy ảo, giải phóng tất cả tài nguyên đã được cấp phát.
  - o **vlist**: liệt kê tất cả các máy ảo đang chạy cùng với các thông số về bộ nhớ cấp phát, chỉ số tiến trình, các giao diện mạng tồn tại trên máy ảo...
  - o **vconfig**: cấu hình cho giao diện mạng trên máy ảo.
  - o **vclean**: xóa toàn bộ tất cả các *Netkit processes* kể cả các máy ảo đang chạy
- Các lệnh phổ biến thuộc nhóm **l-commands**
  - o **lstart**: khởi tạo mạng ảo mới. Các máy ảo được khởi động tự động sau lệnh này.
  - o **lhalt**: dừng hoạt động của mạng ảo.
  - o **lcrash**: hủy đi mạng ảo, giải phóng tất cả tài nguyên đã được cấp phát.

- **linfo**: thông tin cơ bản về mạng ảo
- **lclean**: xóa toàn các file tạm chưa được gán trong mạng ảo.
- **ltest**: test thử hoạt động của mạng ảo.

#### 4. Cài đặt Netkit trong môi trường Linux

*Yêu cầu của máy host để Netkit hoạt động hiệu quả*

- Kiến trúc 32 bit i386.
- CPU  $\geq$  600 MHZ.
- Dung lượng trống cần thiết trên máy host khoảng 600 Mb. Mỗi máy ảo tối thiểu 10 MB.

*Yêu cầu của máy ảo để Netkit hoạt động hiệu quả*

- Là một User Mode Linux
- Được tích hợp các công cụ phổ biến (awk, lsof...), hỗ trợ hầu hết các giao thức truyền tải dữ liệu.

*Các thao tác cài đặt*

- Truy cập vào trang chủ của Netkit, vào phần Download, tiến hành download 3 file nén dưới đây: *Netkit core version 2.8 and documentation, Netkit file system version 5.2, Netkit kernel version 2.8.*
- Tạo 1 folder mới có tên là **THMMT** trong **Home**, di chuyển 3 file nén đã download được vào trong thư mục này. Chuyển đến thư mục **THMMT**, thực hiện giải nén 3 file này bằng cách viết 1 script là **giainen.sh** với nội dung như sau:

```
tar -xjSf netkit-2.8.tar.bz2
tar -xjSf netkit-filesystem-i386-F5.2.tar.bz2
tar -xjSf netkit-kernel-i386-K2.8.tar.bz2
```

- Chú ý rằng *netkit-2.8.tar.bz2*, *netkit-filesystem-i386-F5.2.tar.bz2* và *netkit-kernel-i386-K2.8.tar.bz2* là tên của 3 file nén được download về. Sau khi giải nén xong, ta có được một thư mục có tên là **netkit**.
- Tiến hành cấu hình biến môi trường **MANPATH**, **PATH** để Netkit có thể hoạt động và thực thi. Cấu hình biến môi trường như sau:

```
$ export NETKIT_HOME=~/THMMT/netkit
$ export MANPATH=$NETKIT_HOME/man
$ export PATH=$NETKIT_HOME/bin:$PATH
```

- Để biến môi trường không cần tạo lại mỗi lần khởi động Netkit. Ta thiết lập biến môi trường trong file **~/.bashrc** và khởi động lại terminal.
- Kiểm tra cấu hình đã cài đặt bằng lệnh **./check\_configuration.sh** trong thư mục **netkit**. Nếu cài đặt thành công thì sẽ nhận được thông báo chúc mừng.

```
passed.
> Checking for availability of terminal emulator applications:
    xterm      : found
    konsole    : not found
    gnome-terminal : found
passed.
> Checking filesystem type... passed.
> Checking whether 32-bit executables can run... passed.
[ READY ] Congratulations! Your Netkit setup is now complete!
          Enjoy Netkit!
```

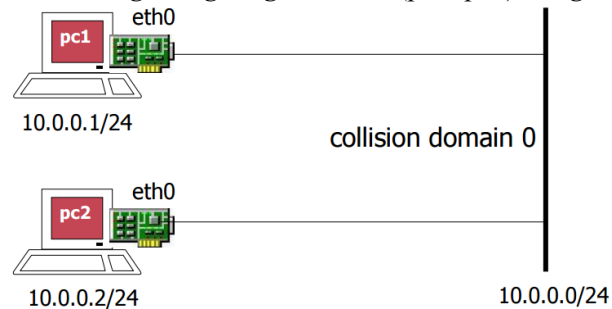
## 5. Xây dựng mô hình mạng với Netkit

Thuật ngữ *Netkit lab* để chỉ 1 tập hợp các máy ảo đã được cấu hình để có thể đưa vào hoạt động hoặc dừng hoạt động. *Netkit lab* có thể được cài đặt thông qua 2 cách:

- *Cách 1:* Viết 1 file script riêng (*lab-script*) trong đó gọi lệnh *vstart* cho từng máy ảo riêng lẻ. Script là 1 tập hợp các lệnh sẽ được hiểu và thực thi bởi máy host. Ta có thể không viết script mà tiến hành gọi từng lệnh *vstart* cho từng máy ảo trên terminal của máy host (*không khuyến khích*).
- *Cách 2:* Sử dụng tập lệnh *lcommands* để xây dựng hoàn chỉnh một mạng ảo (*nên sử dụng*). Với cách này, ta tạo 1 cấu trúc thư mục như đã trình bày trong phần II.2.

## 6. Bài tập thực hành với Netkit

**Bài tập 1:** Xây dựng mô hình mạng đơn giản gồm 2 host (pc1, pc2) bằng cách cấu hình từng máy



Bước 1: Khởi tạo máy ảo thứ nhất bằng lệnh: *vstart pc1 --eth0=A* hoặc *vstart --eth0=A pc1*. Thực hiện tương tự với máy ảo thứ hai.

```
/home/tran/Netkit# vstart pc2 --eth0=A
pc1
Cleaning up ifupdown...
Mounting kernel modules directory (/home/tran/Netkit/netkit/kernel/modules/lib/modules) on /lib/modules/...
Loading kernel modules...done.
Setting kernel variables (/etc/sysctl.conf)...done.
Setting up networking....
Configuring network interfaces...done.
Starting portmap daemon....
INIT: Entering runlevel: 2
phase 1 init script
```

Bước 2: Kiểm tra cấu hình mạng ban đầu của pc1 hoặc pc2 với *ifconfig*. Tại sao không có giao diện mạng eth0 đã khai báo?

```
pc2:~# ifconfig
lo:
Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:2 errors:0 dropped:0 overruns:0 frame:0
TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:100 (100.0 B) TX bytes:100 (100.0 B)

pc2:~#
```

Bước 3: Cập nhật cấu hình mạng của pc1 bằng lệnh: *ifconfig eth0 10.0.0.1 netmask 255.255.255.0 broadcast 10.0.0.255 up*. Thực hiện tương tự với pc2. Địa chỉ được gán cho 2 máy ảo là: **10.0.0.1** và **10.0.0.2**. Kiểm tra lại cấu hình mạng của 2 máy và nhận xét?

```
pc1:~# ifconfig eth0 10.0.0.1 netmask 255.255.255.0 broadcast 10.0.0.255 up
pc1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 6e:5f:98:37:0c:07
          inet addr:10.0.0.1  Bcast:10.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::6c5f:98ff:fe37:c07/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:384 (384.0 B)  TX bytes:468 (468.0 B)
          Interrupt:5

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
```

Bước 4: Sử dụng lệnh *ping* từ pc1 đến pc2 hoặc ngược lại; hoặc sử dụng lệnh *traceroute* để kiểm tra gói tin đến trên pc1 hoặc pc2. Nhận xét kết quả hiển thị sau khi ping.

```
pc2:~# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=5.81 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.195 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.174 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.768 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.588 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.128 ms
```

**Bài tập 2:** Thực hiện lại bài tập 1 nhưng viết các script để cấu hình cho 2 máy

**Lưu ý:** Trong bài tập này, địa chỉ của pc1 sẽ là 10.0.0.2, pc2 là 10.0.0.3

Bước 1: Tạo thư mục **Lab1.2**. Cấu trúc cây thư mục này được tổ chức như sau:

```
$NETKIT_HOME
----Lab1.2      (folder)
|
|----pc1        (folder)
|
|----pc2        (folder)
|
|----pc1.startup (file)
|
|----pc2.startup (file)
|
|----lab.conf   (file)
```

Trong đó, (*folder*) thể hiện cho các thư mục và (*file*) thể hiện cho tập tin.

- Cấu hình cho từng máy được tạo bằng lệnh: ***touch <tenmayao>.startup***
- Thư mục cho từng máy được tạo bằng lệnh ***mkdir <tenmayao>***
- Cấu hình cho mô hình mạng được tạo bằng lệnh ***touch lab.conf***

Bước 2: Thực hiện cấu hình mạng ảo bằng cách soạn thảo nội dung cho *pc1.startup* và *pc2.startup*. Soạn thảo nội dung cho *lab.conf* để thiết lập hình thái mạng ảo.

- Cấu hình cho pc1 như hình dưới đây và giải thích các lệnh trong *pc1.startup*. Nếu nội dung *pc1.startup* thay đổi với lệnh *ifconfig eth0 10.0.0.2 netmask 255.255.255.0 broadcast 10.0.0.255 up* thì có được không?

```
pc1.startup ✕
ip link set eth0 up
ip address add 10.0.0.2/24 dev eth0
ip route add default via 10.0.0.1
```

- Thực hiện tương tự với pc2.
- Cấu hình cho hình trạng mạng với **lab.conf** như sau.

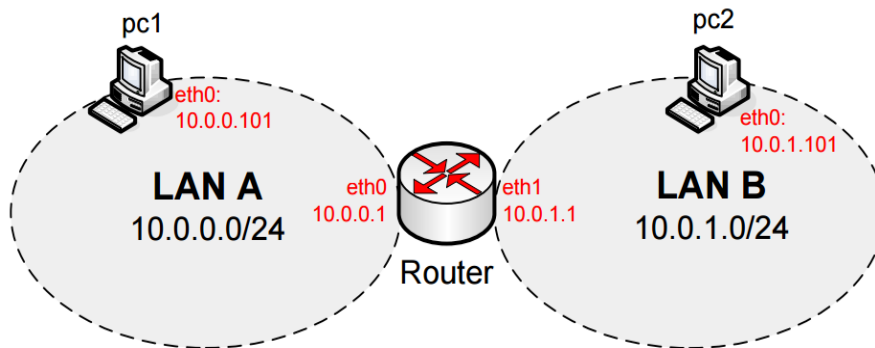
```
lab.conf ✕
pc1[0]="A"
pc2[0]="A"
```



Bước 3: Dùng *ping* kiểm tra kết nối giữa 2 máy hoặc sử dụng lệnh *traceroute* để kiểm tra gói tin đến máy ảo 1 hoặc 2

```
pc1:~# ping 10.0.03
PING 10.0.03 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.259 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.235 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.758 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.287 ms
```

**Bài tập 3:** Xây dựng mô hình mạng với 1 router và 2 máy ảo



Bước 1: Tạo thư mục **Lab1.3** với cấu trúc như sau:

```
$NETKIT_HOME
----Lab1.3          (folder)
----pc1             (folder)
----pc2             (folder)
----router          (folder)
----pc1.startup     (file)
----pc2.startup     (file)
----router.startup  (file)
----lab.conf        (file)
```

Bước 2: Thực hiện cấu hình mạng ảo trong **Lab1.3**

- Cấu hình pc1 như sau, pc2 thực hiện tương tự.

```
pc1.startup ✕
ip link set eth0 up
ip address add 10.0.0.101/24 dev eth0
ip route add default via 10.0.0.1
```

- Cấu hình router như sau:

```
router.startup ✕
ip link set eth0 up
ip link set eth1 up
ip address add 10.0.0.1/24 dev eth0
ip address add 10.0.1.1/24 dev eth1
```

- Cấu hình hình trạng mạng như sau:



```
lab.conf X
router[0]="A"
router[1]="B"
router[mem]=64
pc1[0]="A"
pc2[0]="B"
```

Bước 3: Tại thư mục **Lab1.3**, dùng lệnh **lstart** để khởi động mạng.

- Trên pc2 và router thực hiện lệnh **tcpdump** để bắt gói tin truyền tải đến.
- Trên pc1 **ping** đến pc2, quan sát kết quả trên pc2 và router.
- Thực hiện lại lệnh **ping** và **tcpdump**, lần này sử dụng cú pháp **tcpdump -w </hosthome/filename.pcap>** để ghi thông tin bắt được ra file thay vì hiển thị trên terminal của máy ảo. Thư mục **/hosthome** là thư mục để chia sẻ các tài nguyên (file, folder...) giữa máy host và máy ảo. File **.pcap** sẽ được sát kết quả cụ thể hơn với Wireshark.

**Bài tập 4:** Mở rộng mô hình mạng ở bài 3 bằng cách thêm vào 1 máy ở LAN A và 1 máy ở LAN B.

**Bài tập 5:** Từ bài tập 4, bổ sung LAN C gồm 2 máy, LAN C kết nối với LAN A qua một router mới.

### III. Phần mềm Wireshark

#### 1. Giới thiệu

Wireshark là một công cụ mã nguồn mở sử dụng phổ biến trên nhiều hệ điều hành khác nhau. Wireshark cho phép quan sát và phân tích các thành phần trong gói dữ liệu bắt được theo thời gian thực.

Khác với công cụ **tcpdump**, Wireshark cung cấp giao diện thân thiện và dễ tương tác. Vì vậy, trong phần thực hành, ta dùng **tcpdump** để bắt và lưu thông tin dữ liệu trong mạng ảo vào 1 file **.pcap**, và dùng **Wireshark** để phân tích các thông tin trong file này để hiểu rõ về định dạng dữ liệu và giao thức truyền tải ở mỗi tầng.

#### 2. Giao diện tương tác với Wireshark

The screenshot displays the Wireshark interface with the following components labeled:

- command menus:** File, Edit, View, Go, Capture, Analyze, Statistics, Help.
- display filter specification:** A field for entering display filters.
- listing of captured packets:** A table showing captured packets with columns: No., Time, Source, Destination, Protocol, Info.
- details of selected packet header:** A pane showing the hierarchical structure of the selected packet's headers (Ethernet II, Internet Protocol, Transmission Control Protocol, Hypertext Transfer Protocol).
- packet content in hexadecimal and ASCII:** A pane showing the raw data of the selected packet in hexadecimal and ASCII format.

- *Thanh menu lệnh*: chứa các lựa chọn để tương tác với file đang được mở.
- *Bộ lọc*: lọc và hiển thị dữ liệu tương ứng.
- *Giao diện liệt kê các gói dữ liệu*: thể hiện thông tin chi tiết của các gói dữ liệu bắt được như: protocol, source, destination,...
- *Giao diện thể hiện thông tin của dữ liệu*: số thứ tự frame, chiều dài frame, khuôn dạng frame, khuôn dạng gói tin IP, giao thức tầng ứng dụng...
- *Giao diện thể hiện nội dung của dữ liệu bằng mã HEX và mã ASCII*

### 3. Bài tập thực hành với Wireshark

#### ***Bài tập 1: Phân tích dữ liệu cơ bản***

- Trên máy host, mở Firefox và khởi động bắt gói tin trên Wireshark, truy cập vào địa chỉ: [www.ctu.edu.vn](http://www.ctu.edu.vn). Thực hiện một số tương tác trên trang web. Sau một thời gian, ngừng bắt gói tin trên Wireshark và quan sát cửa sổ thông tin. Trả lời các câu hỏi sau đây:
- Giao thức để hiển thị nội dung trang web trên Firefox là giao thức gì, thể hiện ở gói tin thứ mấy?
- Các gói tin màu xanh lá cây, màu xanh dương, màu xanh dương nhạt lần lượt đại diện cho các giao thức nào? Các gói tin màu khác có ý nghĩa gì?
- Chọn 1 gói tin bất kỳ, cho biết số thứ tự gói tin vừa chọn, độ dài gói tin, khuôn dạng dữ liệu của gói tin trong từng tầng.
- Cho biết địa chỉ MAC, IP và PORT của máy nguồn, máy đích;
- Thử lọc ra các gói tin thuộc các giao thức HTTP, DNS, TCP. Kết quả nhận được với thông tin lọc: ***http.host==www.ctu.edu.vn***
- Chọn 1 gói tin bất kỳ trong những gói tin vừa lọc được, chọn follow TCP Stream. Nhận xét về kết quả thu được?

#### ***Bài tập 2: Phân tích dữ liệu trên mô hình mạng ảo***

- Áp dụng cho mạng ảo đã tạo trong bài tập 3.
- Dùng lệnh tcpdump trên pc2 và router với chức năng ghi dữ liệu vào file *data\_pc2.pcap*, *data\_router.pcap* và lưu trữ trong /hosthome.
- Ping từ pc1 sang pc2.
- Trên máy host, dùng Wireshark mở file *data\_pc2.pcap* và *data\_router.pcap*. Quan sát kết quả.

## BUỔI THỰC HÀNH 2

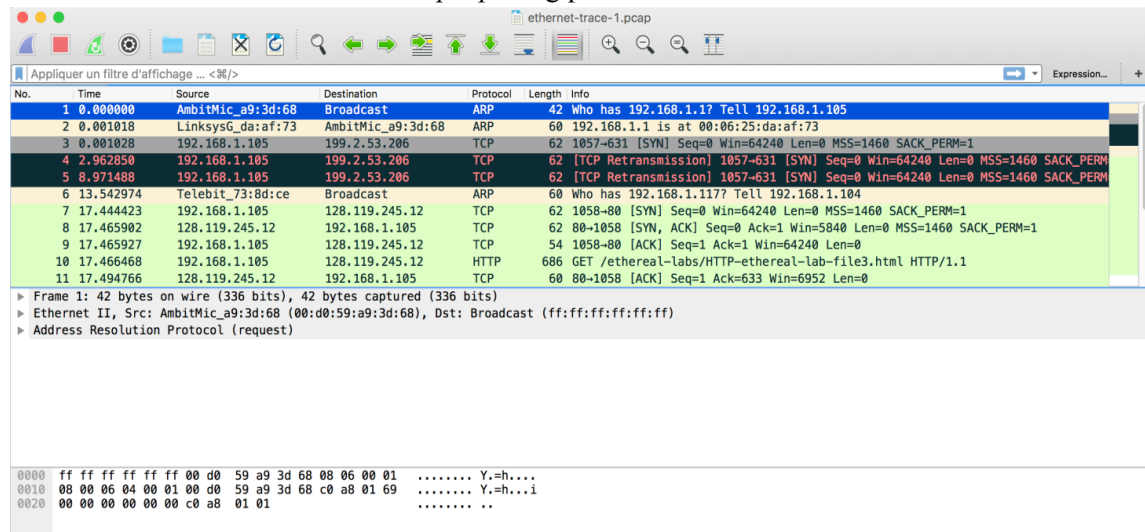
### Mục đích:

- Phân tích định dạng khung của 2 giao thức: **Ethernet II** và **Wifi 802.11**
- Xây dựng mô hình mạng ảo để khảo sát giao thức ARP.

### I. Ethernet II và Wifi 802.11 frame

#### Bài tập 3: Ethernet II frame trên Wireshark

Bước 1: Mở file *ethernet-trace1.pcap* bằng phần mềm Wireshark



Bước 2: Khảo sát về MAC address trong Ethernet

- Chọn gói tin số 10. Đây là gói tin chứa thông điệp GET/HTTP được gửi từ một máy tính đến Server chứa website *gaia.cs.umass.edu* để hiển thị nội dung của trang web này.
- Dựa vào thông tin của gói tin số 10 trên Wireshark, trả lời các câu hỏi sau:
  - o Địa chỉ MAC của máy gửi đi thông điệp GET/HTTP?
  - o Địa chỉ đích (destination) trong khung Ethernet II có phải là địa chỉ MAC của Server chứa website *gaia.cs.umass.edu* hay không?
  - o Thông tin hiển thị cho thấy dữ liệu được đóng gói theo khuôn dạng khung Ethernet II (DIX Ethernet). Hãy nêu sự giống và khác nhau giữa khung Ethernet II này với khung Ethernet 802.3.
  - o Giá trị hexadecimal của trường TYPE trong frame? Ý nghĩa của trường TYPE?
  - o Trong cùng Ethernet segment, có thể tồn tại cả khung Ethernet 802.3 và khung Ethernet II hay không? Nếu có, hãy giải thích cách thức chuyển đổi trường TYPE của khung Ethernet II về trường LENGTH của khung Ethernet 802.3
  - o Từ vị trí bắt đầu khung cho đến khi ký tự G của lệnh GET trong giao thức HTTP xuất hiện là bao nhiêu bytes? Các bytes đó đại diện cho các thông tin gì?

#### **Bài tập 4: Ethernet II frame với ARP trên Wireshark**

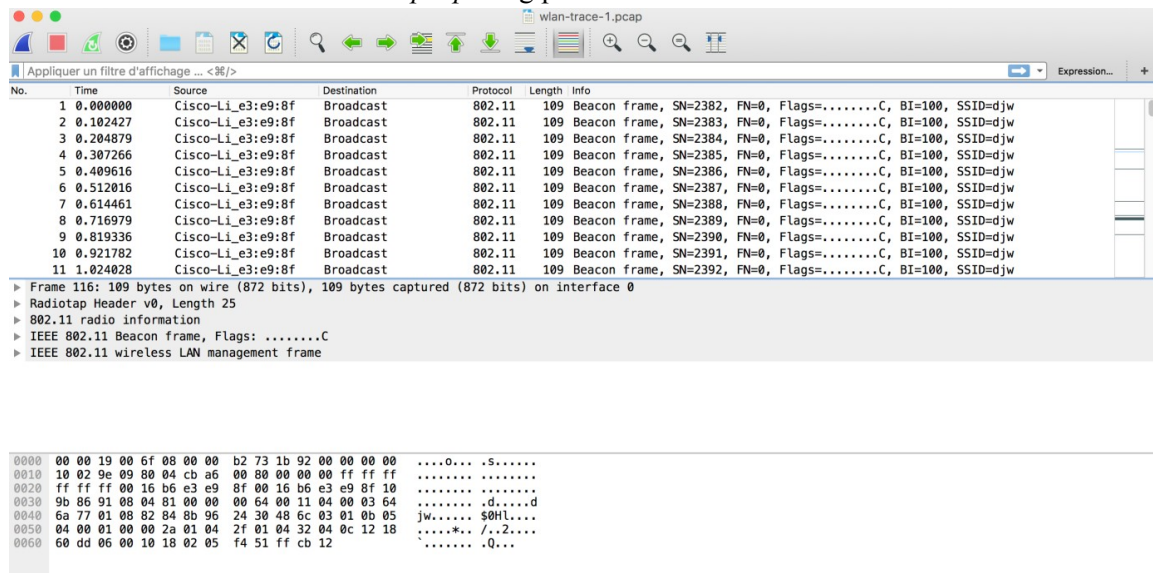
Bước 1: Mở file *ethernet-trace1.pcap* bằng phần mềm Wireshark

Bước 2: Khảo sát về giao thức ARP trong khung Ethernet II

- Chọn gói tin số 1. Giá trị hexadecimal cho MAC nguồn và MAC đích của khung Ethernet II chứa *ARP request* là gì?
- Tìm ARP opcode trong khung. Bắt đầu từ đầu khung đến ARP opcode là bao nhiêu byte? Các byte đại diện cho thông tin gì? Giá trị của opcode là bao nhiêu nếu đây là ARP request?
- Thông điệp ARP có chứa địa chỉ IP của người gửi hay không? Nếu có hãy chỉ ra địa chỉ đó là bao nhiêu?
- Chọn gói tin số 2. Giá trị hexadecimal cho MAC nguồn và MAC đích trong khung Ethernet II chứa *ARP reply* là gì? Giá trị opcode là bao nhiêu với *ARP reply*?

#### **Bài tập 5: Wifi frame 802.11 trên Wireshark cơ bản**

Bước 1: Mở file *wlan-trace-1.pcap* bằng phần mềm Wireshark.



Bước 2: Khảo sát về MAC address trong Wifi

- Chọn 1 gói dữ liệu bất kỳ và cho biết ý nghĩa của các thông tin sau: *Frame*, *Radiotap*, *IEEE 802.11*, *Data* (nếu có)
- Nhập vào bộ lọc: *wlan.fc.type==n* với n có các giá trị 0, 1, 2. Giải thích ý nghĩa lệnh này với tham số n. Sau khi lọc, thông tin hiển thị ra là gì?
- Nhập vào bộ lọc: *wlan.fc.type==2 && wlan.fc.retry==0*. Giải thích ý nghĩa lệnh này. Sau khi lọc, thông tin hiển thị ra là gì?

#### **Bài tập 6: Wifi frame 802.11 trên Wireshark nâng cao**

Bước 1: Mở file *Wireshark\_802\_11.pcap* bằng phần mềm Wireshark



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco-Li_f7:1d:51	Broadcast	802.11	183	Beacon frame, SN=2854, FN=0, Flags=.....C, BI=100, SSID=30 Munroe St
2	0.062101	b6:78:8c:c1:ae:c0	65:a8:d5:b2:c1:99	802.11	1624	802.11 Block Ack Req, Flags=op.P...T.
3	0.085474	Cisco-Li_f7:1d:51	Broadcast	802.11	183	Beacon frame, SN=2855, FN=0, Flags=.....C, BI=100, SSID=30 Munroe St
4	0.187919	Cisco-Li_f7:1d:51	Broadcast	802.11	183	Beacon frame, SN=2856, FN=0, Flags=.....C, BI=100, SSID=30 Munroe St
5	0.188100	IntelCor_d1:b6:4f	IntelCor_d1:b6:4f	802.11	54	QoS Null function (No data), SN=1482, FN=0, Flags=.....TC
6	0.188281	IntelCor_d1:b6:4f	Cisco-Li_f7:1d:51	802.11	38	Acknowledgement, Flags=.....C
7	0.188935	IntelCor_d1:b6:4f	Cisco-Li_f7:1d:51	802.11	54	QoS Null function (No data), SN=1483, FN=0, Flags=...P...TC
8	0.189834	IntelCor_d1:b6:4f	IntelCor_d1:b6:4f	802.11	38	Acknowledgement, Flags=.....C
9	0.290284	Cisco-Li_f7:1d:51	Broadcast	802.11	183	Beacon frame, SN=2857, FN=0, Flags=.....C, BI=100, SSID=30 Munroe St
10	0.294432	Linksys6_67:22:94	Broadcast	802.11	90	Beacon frame, SN=3072, FN=0, Flags=.....C, BI=62, SSID=L1\357\277\275\001
11	0.393174	Cisco-Li_f7:1d:51	Broadcast	802.11	183	Beacon frame, SN=2858, FN=0, Flags=.....C, BI=100, SSID=30 Munroe St

Frame 1: 183 bytes on wire (1464 bits), 183 bytes captured (1464 bits)  
 Radiotap Header v0, Length 24  
 802.11 radio information  
 IEEE 802.11 Beacon frame, Flags: .....C  
 IEEE 802.11 wireless LAN management frame

```

0000 00 00 18 00 ee 58 00 00 10 02 85 09 a0 00 c3 9c .....X.....
0010 52 00 00 47 08 26 7e 05 80 00 00 00 ff ff ff ff R..G.&.....
0020 ff ff 00 16 b6 f7 1d 51 00 16 b6 f7 1d 51 60 b2 .....0.....0
0030 82 e1 38 96 28 00 00 64 00 01 06 00 0c 33 30 ..8.(...d...30
0040 20 4d 75 6e 72 6f 65 20 53 74 01 04 82 84 8b 96 Munroe St....
0050 03 01 06 05 04 00 01 00 00 07 06 55 53 49 01 0b .....USI...
0060 1a 0c 12 0f 00 03 a4 00 00 27 a4 00 00 42 43 5e .....BC^...
0070 00 62 32 2f 00 2a 01 00 32 08 8c 12 98 24 d0 48 .b2/...2...S.H
0080 50 6c dd 15 00 8a f5 0a 02 40 c0 00 03 01 03 05 ^.....@.....
0090 0e 04 ff 00 03 00 11 01 01 dd 18 00 50 f2 02 01 .....P.....
00a0 01 0f 00 03 a4 00 00 27 a4 00 00 42 43 5e 00 62 .....BC^..b
00b0 32 2f 00 08 26 7e 05 .....2/..&..
    
```

Bước 2: Khảo sát về MAC address trong Wifi 802.11

- Chọn gói tin số 1. Cho biết giá trị hexadecimal của MAC nguồn, MAC đích.
- Hãy cho biết các Data rate (Mbps) được hỗ trợ truyền tải trên *Access Point 30 Munroe St* bằng cách kiểm tra các beacon frame đến từ địa chỉ này.
- Tiến hành lọc để chỉ hiển thị các gói tin thuộc giao thức TCP.
- Chọn gói tin số 474. Gói tin này chứa khung chứa *SYN TCP segment*.

No.	Time	Source	Destination	Protocol	Length	Info
474	24.811093	128.119.245.12	128.119.245.12	TCP	110	2538->0 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM=1
476	24.827751	128.119.245.12	128.119.245.12	TCP	110	80->2538 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0
478	24.828024	128.119.245.12	128.119.245.12	TCP	102	2538->80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
480	24.828253	128.119.245.12	128.119.245.12	HTTP	537	GET /wireshark-labs/alice.txt HTTP/1.1
482	24.846898	128.119.245.12	128.119.245.12	TCP	108	80->2538 [ACK] Seq=1 Ack=436 Win=6432 Len=0
484	24.847171	128.119.245.12	128.119.245.12	TCP	108	[TCP Dup ACK 482#1] 80->2538 [ACK] Seq=1 Ack=436 Win=6432 Len=0
486	24.848820	128.119.245.12	128.119.245.12	TCP	415	[TCP segment of a reassembled PDU]
488	24.850314	128.119.245.12	128.119.245.12	TCP	1562	[TCP segment of a reassembled PDU]
489	24.850800	128.119.245.12	128.119.245.12	TCP	1562	[TCP Retransmission] 80->2538 [ACK] Seq=314 Ack=436 Win=6432 Len=1460
490	24.851390	128.119.245.12	128.119.245.12	TCP	1562	[TCP Retransmission] 80->2538 [ACK] Seq=314 Ack=436 Win=6432 Len=1460
492	24.851620	128.119.245.12	128.119.245.12	TCP	1562	[TCP Retransmission] 80->2538 [ACK] Seq=314 Ack=436 Win=6432 Len=1460

Frame Control Field: 0x8801  
 .000 0000 0010 1100 = Duration: 44 microseconds  
 Receiver address: Cisco-Li\_f7:1d:51 (00:16:b6:f7:1d:51)  
 Destination address: Cisco-Li\_f4:eb:a8 (00:16:b6:f4:eb:a8)  
 Transmitter address: IntelCor\_d1:b6:4f (00:13:02:d1:b6:4f)  
 Source address: IntelCor\_d1:b6:4f (00:13:02:d1:b6:4f)  
 BSS Id: Cisco-Li\_f7:1d:51 (00:16:b6:f7:1d:51)  
 STA address: IntelCor\_d1:b6:4f (00:13:02:d1:b6:4f)  
 .... 0000 = Fragment number: 0  
 0000 0011 0001 .... = Sequence number: 49  
 Frame check sequence: 0xad57fc00 [correct]

```

0000 00 00 18 00 ee 58 00 00 10 02 85 09 c0 00 da 9c .....X.....
0010 60 00 00 3e e0 fc 57 ad 88 01 2c 00 00 16 b6 f7 .....>..W.....
0020 1d 51 00 13 02 d1 b6 4f 00 16 b6 f4 eb a8 10 03 .0.....0.....
0030 00 00 aa aa 03 00 00 00 08 00 45 00 00 30 13 24 .....E..0.$
0040 40 00 80 06 b0 0a c0 a8 01 6d 80 77 f5 0c 09 ea @.....m.W....
0050 00 50 71 af cd 46 00 00 00 70 02 40 00 c2 55 .Pq..F...p@..U
0060 00 00 02 04 05 b4 01 01 04 02 e0 fc 57 ad .....W.....
    
```

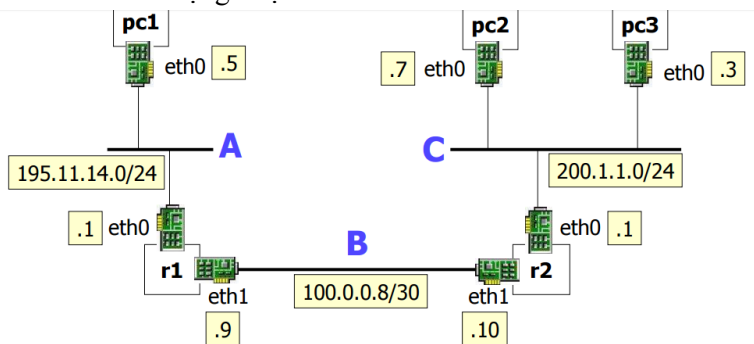
- o Giải thích ý nghĩa lần lượt của *Receiver Address*, *Destination Address*, *Transmitter Address*, và *Source Address* trong gói tin này.
- o Địa chỉ MAC nào trong frame gửi đi *SYN TCP segment*.
- o Địa chỉ MAC nào trong frame là địa chỉ của router đầu tiên mà host kết nối đến.
- o Địa chỉ MAC nào trong frame là địa chỉ BSS.
- Chọn gói tin số 476. Gói tin này chứa khung chứa *SYN ACK TCP segment*
  - o Địa chỉ MAC nào trong frame là địa chỉ MAC gửi đi *SYNACK TCP segment*.

- Địa chỉ MAC nào trong frame là địa chỉ của router đầu tiên mà host kết nối đến.
- Địa chỉ MAC nào trong frame này là địa chỉ BSS

#### IV. Khảo sát ARP

##### *Bài tập 1: Xây dựng mô hình mạng khảo sát giải thuật ARP*

Bước 1: Quan sát mô hình mạng được thiết kế như sau.



Bước 2: Cấu hình cho mô hình mạng

- Tổ chức cây thư mục như sau:

```
$NETKIT_HOME
----Lab2.12      (folder)
----pc1          (folder)
----pc2          (folder)
----pc3          (folder)
----r1           (folder)
----r2           (folder)
----pc1.startup  (file)
----pc2.startup  (file)
----pc3.startup  (file)
----r1.startup   (file)
----r2.startup   (file)
----lab.conf     (file)
```

- Cấu hình cho pc1 như sau. Đối với pc2 và pc3, thay đổi thành: `route add default gw 200.1.1.1`

```
pc1.startup ✕
ifconfig eth0 195.11.14.5 up
route add default gw 195.11.14.1
```

- Cấu hình cho router r1 như sau. Tương tự, cấu hình cho router r2.

```
r1.startup ✕
ifconfig eth0 195.11.14.1 up
ifconfig eth1 100.0.0.9 netmask 255.255.255.252 broadcast 100.0.0.11 up
route add -net 200.1.1.0 netmask 255.255.255.0 gw 100.0.0.10 dev eth1
```

- Hình trạng mạng được thiết lập như sau:

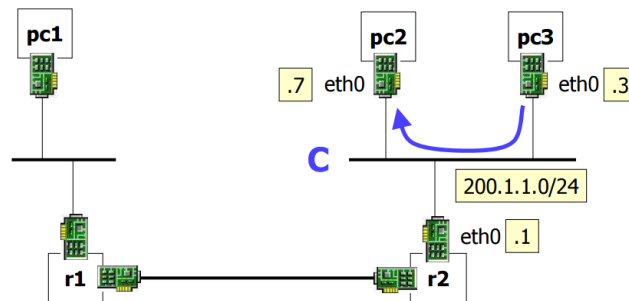
```
lab.conf
r1[0]="A"
r1[1]="B"

r2[0]="C"
r2[1]="B"

pc1[0]="A"
pc2[0]="C"
pc3[0]="C"
```

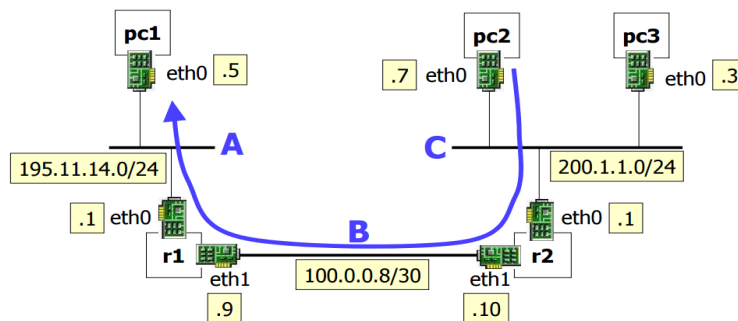
Bước 3: Khởi động mô hình mạng bằng lệnh **lstart** từ thư mục **Lab2.12**

Bước 4a: Khảo sát ARP cache khi truyền dữ liệu giữa 2 máy thuộc cùng 1 nhánh mạng.



- Trên pc3, gọi lệnh **arp**, nhận xét kết quả. Thực hiện **ping** đến pc2 tại địa chỉ 200.1.1.7. Gọi lại lệnh **arp** và nhận xét kết quả.
- Trên pc2, gọi lệnh **arp**. Giải thích vì sao trong **ARP cache** của pc2 đã có chứa sẵn phân giải địa chỉ IP thành địa chỉ MAC của pc3

Bước 4b: Khảo sát **ARP cache** khi truyền dữ liệu giữa 2 máy thuộc 2 nhánh mạng khác nhau



- Trên pc2, thực hiện lệnh **ping** đến pc1 tại địa chỉ 195.11.14.5. Kiểm tra **ARP cache** và nhận xét kết quả.

```
pc2:~# ping 195.11.14.5
PING 195.11.14.5 (195.11.14.5) 56(84) bytes of data:
64 bytes from 195.11.14.5: icmp_seq=1 ttl=62 time=0.554 ms
64 bytes from 195.11.14.5: icmp_seq=2 ttl=62 time=0.355 ms
64 bytes from 195.11.14.5: icmp_seq=3 ttl=62 time=0.300 ms
^Z
[3]+ Stopped                  ping 195.11.14.5
pc2:~# arp
Address      Hwtype      Hwaddress    Flags Mask      Iface
200.1.1.1    ether       3a:40:ee:31:9e:cd  C          eth0
200.1.1.3    ether       2e:fe:a2:81:23:ce  C          eth0
pc2:~#
```

- Trên router r1 và r2, kiểm tra **ARP cache**, nhận xét kết quả.

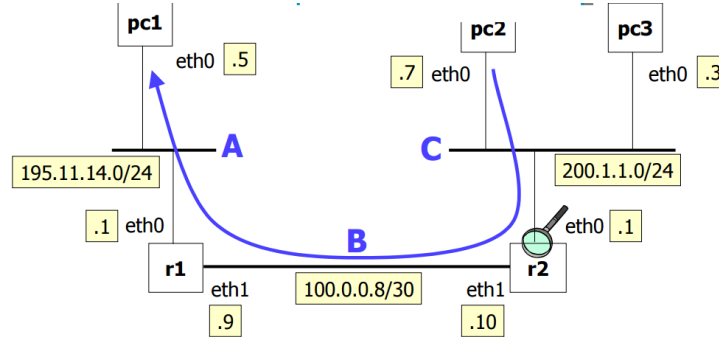


**Bài tập 2:** Khảo sát dữ liệu truyền tải trong mô hình mạng

Bước 1: Khởi động lại mô hình mạng để làm mới lại **ARP cache** trên các thiết bị, sử dụng lệnh: **lcrash** và **lstart**.

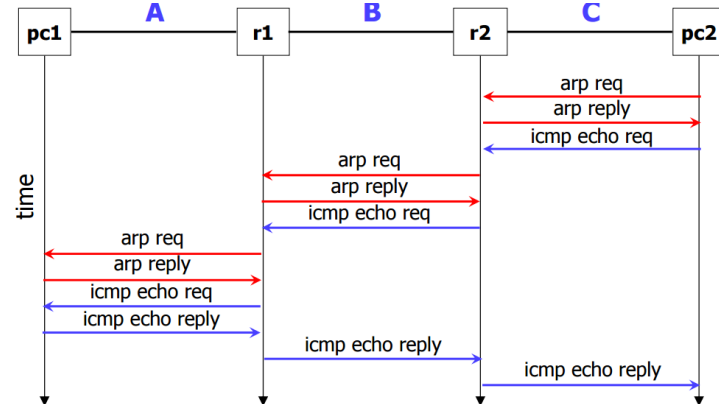
Bước 2: Khảo sát hoạt động của ARP

- Thực hiện truyền dữ liệu từ pc2 đến pc1 bằng lệnh **ping**. Ghi nhận dữ liệu truyền tải qua router r2 bằng lệnh **tcpdump -e -t -i <interfacename> -w <filename>**. Mở file bằng Wireshark và phân tích vai trò của giao thức ARP. Câu hỏi gợi ý cho việc phân tích:
  - o Giá trị hexadecimal của địa chỉ nguồn và địa chỉ đích trong khung của thông điệp ARP (request/reply)
  - o Giá trị hexadecimal của trường TYPE của khung?
  - o Thông điệp ARP (request/reply) có chứa địa chỉ IP của đích đến hay không? Trường Opcode trong thông điệp ARP (request/reply) có giá trị bao nhiêu, vai trò của trường này.
  - o Trong ARP request, cho biết địa chỉ MAC của máy có địa chỉ IP đang được truy vấn.
  - o Trong ARP reply, cho biết địa chỉ IP của đích đến có địa chỉ MAC đang được truy vấn.



- Việc quan sát dữ liệu cũng có thể thực hiện trên r1 hoặc pc1 với cùng cách thức.

Bước 3: Hoạt động của giải thuật ARP trong bài tập này được thể hiện qua sơ đồ tuần tự như sau.



- Kết hợp với việc phân tích dữ liệu của bước 2, giải thích hoạt động của mô hình này.

Bước 4: Thực hiện một số khảo sát khác

- Trên pc2, thực hiện kết nối đến một địa chỉ mạng *cùng nhánh mạng nhưng không có thực*. Nhận xét kết quả.
- Trên pc2, thực hiện kết nối đến một địa chỉ mạng *ngoài nhánh mạng nhưng không có thực*. Nhận xét kết quả.
- Trong 2 trường hợp trên, hãy cho biết dữ liệu được trao đổi ra sao tại các *collision domain*.

## BUỔI THỰC HÀNH 3

### Mục đích:

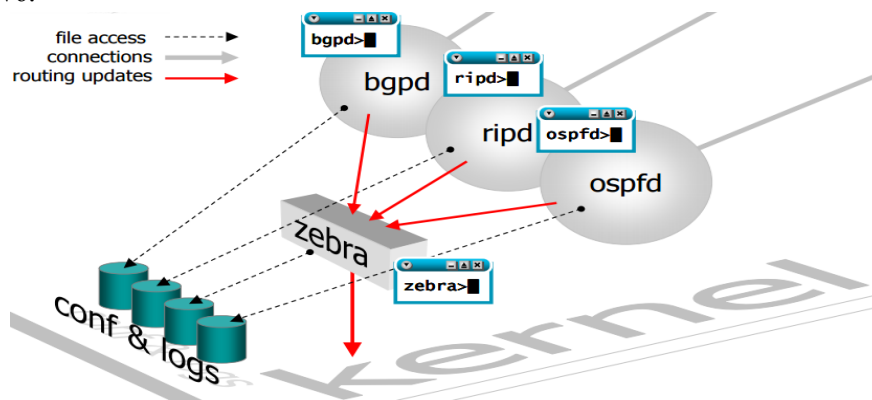
- Giới thiệu giao thức vạch đường nội miền IGP với OSPFv2 và RIPv2.
- Xây dựng mô hình mạng sử dụng RIPv2 và OSPFv2
- Khảo sát dữ liệu của gói tin trao đổi trong mô hình sử dụng giải thuật này.

### I. Giới thiệu

**RIP – Routing Information Protocol** là giao thức định tuyến nội miền sử dụng giải thuật distance vector. RIP quy định số lượng bước nhảy để thực hiện truyền dữ liệu từ nguồn đến đích được gọi là khoảng cách vạch đường (routing metric). Trong RIP, số lượng *hop* tối đa từ nguồn đến đích là 15 bước. Chu kỳ cập nhật cho bảng vạch đường tại các nodes là 30 giây. RIP sử dụng giao thức UDP trên tầng vận chuyển với *port* là 520. RIPv2 ra đời năm 1993, chuẩn hóa lần cuối năm 1998. RIPv2 cho phép truyền tải thông tin về mạng con tại mỗi chu kỳ cập nhật bảng vạch đường, đồng thời hỗ trợ cho cơ chế vạch đường liên miền không phân lớp. Năm 1997, RIPv2 chính thức hỗ trợ cơ chế xác thực tính toàn vẹn với giải thuật MD5.

**OSPF – Open Shortest Path First** là giao thức định tuyến nội miền sử dụng giải thuật Link State. Ưu điểm của OSPF đó là hội tụ nhanh, hỗ trợ mạng có kích thước lớn và không có tình trạng lặp vô tận việc vạch đường. OSPF hỗ trợ xác thực vạch đường theo dạng *plaintext* và *MD5*. Trong phần thực hành sẽ sử dụng giao thức OSPFv2 để minh họa. Khác với RIP, OSPF sử dụng *metric* là *cost* được tính dựa trên băng thông tại mỗi node sao cho tốc độ kết nối càng cao thì *cost* càng thấp.

**Zebra** là 1 gói phần mềm chứa các giải thuật vạch đường nội miền (RIP, OSPF) và liên miền (BGP) cho bộ giao thức IP. Zebra cung cấp các giao thức như RIP, OSPF, BGP. Zebra hỗ trợ cho IPv4 và cả giao thức IPv6.

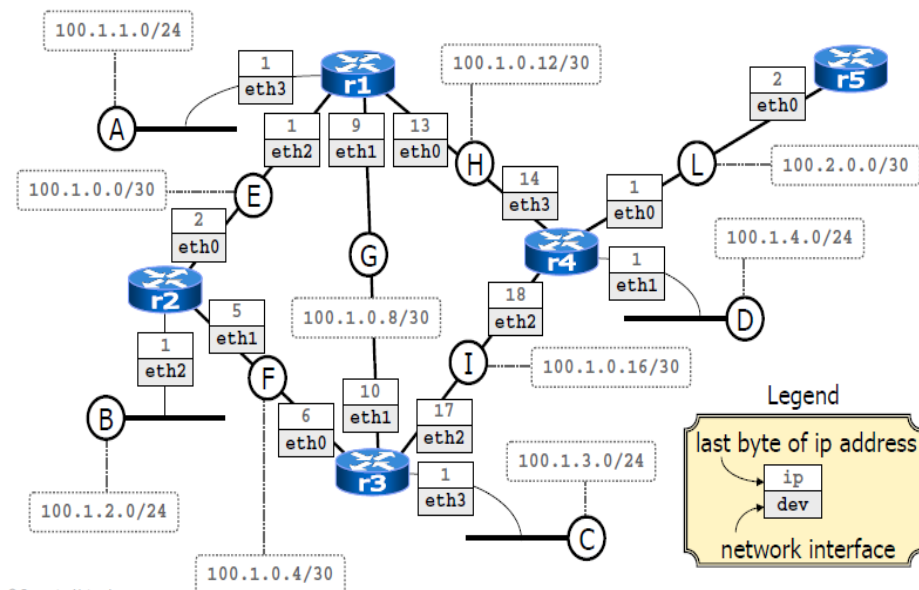


Zebra đã được cài đặt sẵn trong các máy ảo tạo ra bởi Netkit. Đường dẫn đến Zebra trong Netkit là `/etc/zebra/`. Hiện nay Zebra đã ngừng phát triển mà được thay thế bởi 1 gói phần mềm khác mạnh mẽ hơn là *Quagga*.

### II. Bài tập thực hành

#### Bài tập 1: Xây dựng mô hình mạng sử dụng RIPv2

Bước 1: Quan sát mô hình mạng được thiết kế như sau:



Trong đó, các chữ cái là các subnet có địa chỉ mạng riêng, ví dụ subnet A: 100.1.1.0/24.

Các subnet này kết nối lại với nhau qua các router r1, r2, r3, r4. Mỗi router sẽ có nhiều giao diện mạng để kết nối với router khác và kết nối với subnet, ví dụ: eth0 của r1 có địa chỉ 100.1.0.13/30 kết nối với subnet H, eth1 của r1 có địa chỉ 100.1.0.9/30 kết nối với subnet G.

Bước 2: Cấu hình cho mô hình mạng

- Tạo thư mục **Lab3.14** có cấu trúc như sau:

```
$NETKIT_HOME
----Lab3.14      (folder)
  ----r1         (folder)
  ----r2         (folder)
  ----r3         (folder)
  ----r4         (folder)
  ----r5         (folder)
  ----r1.startup (file)
  ----r2.startup (file)
  ----r3.startup (file)
  ----r4.startup (file)
  ----r5.startup (file)
  ----lab.conf   (file)
```

- Cấu hình cho các router. Trong đó r1 có nội dung như sau.

```
r1.startup ✖
/sbin/ifconfig eth0 100.1.0.13 netmask 255.255.255.252 broadcast 100.1.0.15 up
/sbin/ifconfig eth1 100.1.0.9 netmask 255.255.255.252 broadcast 100.1.0.11 up
/sbin/ifconfig eth2 100.1.0.1 netmask 255.255.255.252 broadcast 100.1.0.3 up
/sbin/ifconfig eth3 100.1.1.1 netmask 255.255.255.0 broadcast 100.1.1.255 up
```

- Hình trạng mạng được thiết lập như sau:

```

r1[0]="H"
r1[1]="G"
r1[2]="E"
r1[3]="A"

r2[0]="E"
r2[1]="F"
r2[2]="B"

r3[0]="F"
r3[1]="G"
r3[2]="I"
r3[3]="C"

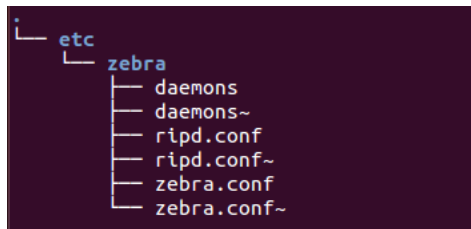
r4[0]="L"
r4[1]="D"
r4[2]="I"
r4[3]="H"

r5[0]="L"

```

Bước 3: Thiết lập **Zebra** trên từng router để kích hoạt giao thức RIPv2. Việc cấu hình thực hiện trên r1, r2, r3 và r4 (KHÔNG thực hiện trên r5). Cách thức thiết lập trình bày dưới đây ghi đề lên nội dung các file chứa trong thư mục **zebra**. Ta cũng có thể trực tiếp truy cập vào từng router và thực hiện thiết lập với nội dung tương tự. Minh hoạ với r1:

- Tạo cây thư mục trong r1 như sau:



- Thiết lập cho phần mềm Zebra như sau. Giải thích ý nghĩa của nội dung thiết lập này.

```

zebra.conf
! *- zebra *-
!
! zebra configuration file
!
hostname zebra
password zebra
enable password zebra
!
! Static default route sample.
!
!ip route 0.0.0.0/0 203.181.89.241
!
log file /var/log/zebra/zebra.log

```

- Mở file **daemons** đã tạo và soạn thảo nội dung như sau. Giải thích ý nghĩa của nội dung này. Có thể lược bỏ đi nội dung nào được?

```

daemons
# This file tells the zebra package
# which daemons to start.
# Entries are in the format: <daemon>=(yes|no|priority)
# where 'yes' is equivalent to infinitely low priority, and
# lower numbers mean higher priority. Read
# /usr/doc/zebra/README.Debian for details.
# Daemons are: bgpd zebra ospfd ospf6d ripd ripngd
zebra=yes
bgpd=no
ospfd=no
ospf6d=no
ripd=yes
ripngd=no

```

- Mở file **ripd.conf** đã tạo và soạn thảo nội dung như sau. Giải thích ý nghĩa của nội dung này.

```

ripd.conf ✖
!
hostname ripd
password zebra
enable password zebra
!
router rip
 redistribute connected
 network 100.1.0.0/16
!
log file /var/log/zebra/ripd.log

```

- Tại sao không cần thiết lập cho Zebra trên r5? Cấu hình cho r5 để kết nối vào mô hình mạng này?  
Gợi ý: xem lại **Lab2.12**

Bước 5: Khởi động **Lab3.14** bằng lệnh **lstart** từ terminal của máy host

Bước 6: Khảo sát sự kết nối của các router trong mạng *khi CHƯA KÍCH HOẠT* giải thuật RIPv2 trên Zebra.

- Từ router r4 lần lượt **ping** đến r1 (100.1.0.13) và r2 (100.1.2.1). Lần lượt giải thích kết quả cho 2 lệnh ping này.

```

r4:~# ping 100.1.2.1
connect: Network is unreachable

```

- Kiểm tra bảng vạch đường của r4 bằng lệnh **route**. Bảng vạch đường thể hiện thông tin gì?

```

r4:~# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
100.1.0.16 * 255.255.255.252 U 0 0 0 eth2
100.2.0.0 * 255.255.255.252 U 0 0 0 eth0
100.1.0.12 * 255.255.255.252 U 0 0 0 eth3
100.1.4.0 * 255.255.255.0 U 0 0 0 eth1

```

Bước 8: Khảo sát lại sự kết nối của các router trong mạng khi **ĐÃ KÍCH HOẠT** giải thuật RIPv2 trên Zebra.

- Kích hoạt RIPv2 bằng lệnh **/etc/init.d/zebra start** trên từng router r1, r2, r3 và r4.

```

r4:~# /etc/init.d/zebra start
Loading capability module if not yet done.
Starting Quagga daemons (prio:10): zebra ripd.

```

- Từ router r4 **ping** lại đến r2 (100.1.2.1), đồng thời kiểm tra lại bảng vạch đường của r4 bằng lệnh **route**. Nhận xét kết quả cho lệnh **ping** và lệnh **route** trên r4.

```

r4:~# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
100.1.0.16 * 255.255.255.252 U 0 0 0 eth2
100.1.0.0 100.1.0.13 255.255.255.252 UG 2 0 0 eth3
100.1.0.4 100.1.0.17 255.255.255.252 UG 2 0 0 eth2
100.2.0.0 * 255.255.255.252 U 0 0 0 eth0
100.1.0.8 100.1.0.13 255.255.255.252 UG 2 0 0 eth3
100.1.0.12 * 255.255.255.252 U 0 0 0 eth3
100.1.4.0 * 255.255.255.0 U 0 0 0 eth1
100.1.2.0 100.1.0.13 255.255.255.0 UG 3 0 0 eth3
100.1.3.0 100.1.0.17 255.255.255.0 UG 2 0 0 eth2
100.1.1.0 100.1.0.13 255.255.255.0 UG 2 0 0 eth3

```

- Từ router r4 thực hiện lệnh **traceroute** đến r2 (100.1.2.1). Nội dung nhận được là gì?

```

r4:~# traceroute 100.1.2.1
traceroute to 100.1.2.1 (100.1.2.1), 64 hops max, 40 byte packets
 1 100.1.0.13 (100.1.0.13) 0 ms 0 ms 0 ms
 2 100.1.2.1 (100.1.2.1) 0 ms 0 ms 0 ms

```

- Trên r2 thực hiện **tcpdump** với eth2 để bắt dữ liệu và ghi ra file khi thực hiện **ping** từ r4. Mở Wireshark với file vừa ghi được, phân tích dữ liệu. Một số câu hỏi gợi ý:
  - o Chọn thông điệp **ICMP Echo Request** đầu tiên được gửi bởi eth2 của r4, cho biết địa chỉ IP của r4 có giá trị hexadecimal bao nhiêu?
  - o Trong Header của gói tin IP, cho biết giá trị hexadecimal của giao thức của tầng phía trên

- Phần Payload của gói tin IP được thể hiện qua bao nhiêu bytes
- Trong các thông điệp ICMP đã được gửi đi bởi r4, trường nào trong gói tin IP luôn thay đổi và trường nào luôn được giữ nguyên?
- Giá trị của trường Identification và TTL là bao nhiêu? Các giá trị này có giữ nguyên hay thay đổi cho các thông điệp ICMP mà r4 trao đổi với first hop của nó không.

Bước 9a: Khảo sát bảng vạch đường trên các router chạy giải thuật RIPv2 qua Zebra

- Từ router r4, **telnet** vào Zebra với lệnh sau: **telnet localhost zebra**. Mật khẩu nhập vào là **zebra** (đã được thiết lập trong zebra.conf).

```
r4:~# telnet localhost zebra
Trying 127.0.0.1...
Connected to r4.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.10).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:
zebra>
```

- Kiểm tra thông tin của 1 giao diện mạng trên router r4 bằng lệnh **show interface <networkinterface>**. Nhận xét về kết quả.

```
zebra> show interface eth0
Interface eth0 is up, line protocol detection is disabled
index 3 metric 1 mtu 1500
flags: <UP,BROADCAST,RUNNING,MULTICAST>
HWaddr: 4a:e4:d9:3b:f2:04
inet 100.2.0.1/30 broadcast 100.2.0.3
inet6 fe80::48e4:d9ff:fe3b:f204/64
 6 input packets (0 multicast), 384 bytes, 0 dropped
 0 input errors, 0 length, 0 overrun, 0 CRC, 0 frame
 0 fifo, 0 missed
 6 output packets, 468 bytes, 0 dropped
 0 output errors, 0 aborted, 0 carrier, 0 fifo, 0 heartbeat
 0 window, 0 collisions
```

- Để kiểm tra đường đi từ Router r4 đến các địa chỉ khác trong miền (router, subnet), ta sử dụng lệnh **show ip route**. Nhận xét kết quả với các thông số C – connected, R – RIP

```
zebra> show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

R>* 100.1.0.0/30 [120/2] via 100.1.0.13, eth3, 00:09:31
R>* 100.1.0.4/30 [120/2] via 100.1.0.17, eth2, 00:08:12
R>* 100.1.0.8/30 [120/2] via 100.1.0.13, eth3, 00:09:31
C>* 100.1.0.12/30 is directly connected, eth3
C>* 100.1.0.16/30 is directly connected, eth2
R>* 100.1.1.0/24 [120/2] via 100.1.0.13, eth3, 00:09:31
R>* 100.1.2.0/24 [120/3] via 100.1.0.13, eth3, 00:08:50
R>* 100.1.3.0/24 [120/2] via 100.1.0.17, eth2, 00:08:12
C>* 100.1.4.0/24 is directly connected, eth1
C>* 100.2.0.0/30 is directly connected, eth0
C>* 127.0.0.0/8 is directly connected, lo
```

Bước 9b: Khảo sát bảng vạch đường trên các router chạy giải thuật RIPv2 với dịch vụ **ripd** trên Zebra bằng lệnh **telnet localhost ripd**

- Để kiểm tra đường đi từ Router r4 đến các địa chỉ khác trong mạng (router, subnet), ta sử dụng lệnh **show ip rip**. Nhận xét kết quả

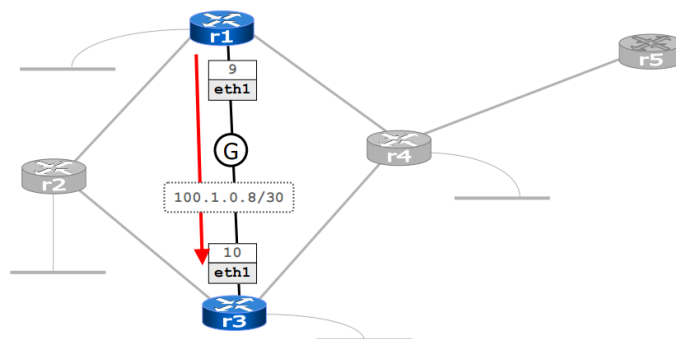
```

ripd> show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface
Network      Next Hop      Metric From      Tag Time
R(n) 100.1.0.0/30 100.1.0.13      2 100.1.0.13      0 02:30
R(n) 100.1.0.4/30 100.1.0.17      2 100.1.0.17      0 02:31
R(n) 100.1.0.8/30 100.1.0.13      2 100.1.0.13      0 02:30
C(i) 100.1.0.12/30 0.0.0.0         1 self           0
C(i) 100.1.0.16/30 0.0.0.0         1 self           0
R(n) 100.1.1.0/24 100.1.0.13      2 100.1.0.13      0 02:30
R(n) 100.1.2.0/24 100.1.0.13      3 100.1.0.13      0 02:30
R(n) 100.1.3.0/24 100.1.0.17      2 100.1.0.17      0 02:31
C(i) 100.1.4.0/24 0.0.0.0         1 self           0
C(r) 100.2.0.0/30 0.0.0.0         1 self (connected;1) 0

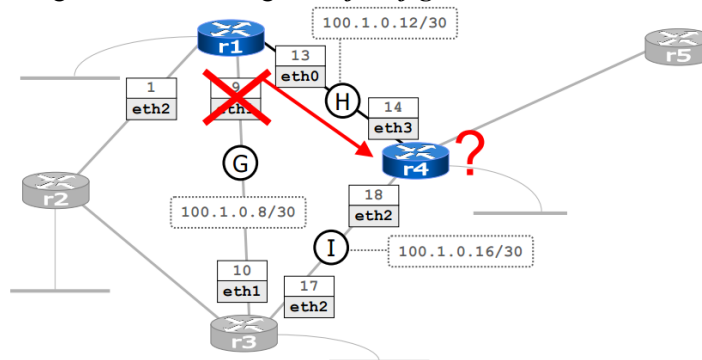
```

Bước 10: Khảo sát sự cập nhật của bảng vạch đường bằng giao thức RIPv2 khi mô hình mạng có sự thay đổi: *giao diện mạng của 1 router bị shutdown.*

- Trên r3, dùng lệnh tcpdump ghi dữ liệu bắt được từ r1 vào file.
- Từ r1, dùng lệnh traceroute kiểm tra thông tin vạch đường đến r3

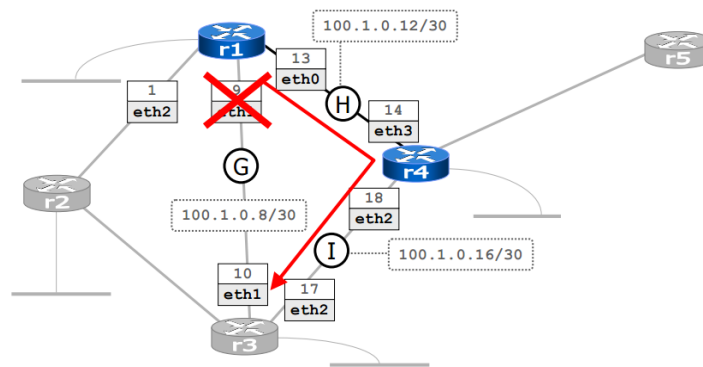


- Tắt giao diện mạng eth1 trên r1 dùng lệnh ***ifconfig eth1 down***



- Từ r1, kiểm tra lại thông tin vạch đường đến r3. Nhận xét thông tin tin vạch đường này. Lưu ý: khoảng thời gian thực hiện kiểm tra (sau 30 giây, sau 1 phút...)

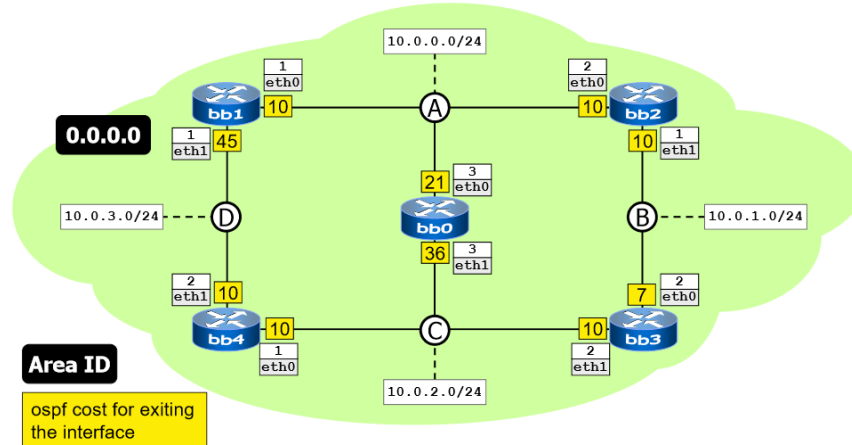




- Kiểm tra lại bảng vạch đường của r1 bằng lệnh **route**. Nhận xét nội dung của bảng vạch đường mới với các thông tin liên quan đến r3.
- Khảo sát thông tin ghi nhận được cho sự thay đổi về đường đi này trên Wireshark.

### **Bài tập 3: Xây dựng mô hình mạng sử dụng OSPFv2 để vạch đường**

Bước 1: Quan sát mô hình mạng được thiết kế như sau



Bước 2: Cấu hình cho mô hình mạng.

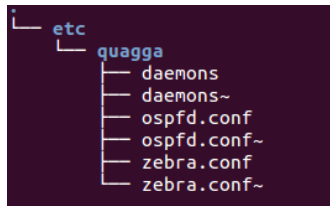
- Tạo thư mục **Lab3.15** có cấu trúc như sau:

```
$NETKIT_HOME
----Lab3.15          (folder)
  ----bb0            (folder)
  ----bb1            (folder)
  ----bb2            (folder)
  ----bb3            (folder)
  ----bb4            (folder)
  ----bb0.startup    (file)
  ----bb1.startup    (file)
  ----bb2.startup    (file)
  ----bb3.startup    (file)
  ----bb4.startup    (file)
  ----lab.conf        (file)
```

- Lần lượt cấu hình cho router và hình trang mạng tương tự như **Lab3.14**

Bước 3: Thiết lập cho **Quagga** trên từng router để sử dụng OSPFv2. Việc thiết lập được thực hiện trên router bb0, các router khác tương tự.

- Trong bb0, tao cây thư mục như sau:



- Thiết lập cho phần mềm Quagga như sau.

```
zebra.conf
! *- zebra *-
!
! zebra configuration file
!
hostname zebra
password zebra
enable password zebra
!
! Static default route sample.
!
!ip route 0.0.0.0/0 203.181.89.241
!
log file /var/log/quagga/zebra.log
```

- Mở file **daemons** đã tạo và soạn thảo nội dung như sau.

```
daemons
# This file tells the zebra package
# which daemons to start.
# Entries are in the format: <daemon>=(yes|no|priority)
# where 'yes' is equivalent to infinitely low priority, and
# lower numbers mean higher priority. Read
# /usr/doc/zebra/README.Debian for details.
# Daemons are: bgpd zebra ospfd ospf6d ripd ripngd
zebra=yes
bgpd=no
ospfd=yes
ospf6d=no
ripd=no
ripngd=no
```

- Mở file **daemons** đã tạo và soạn thảo nội dung như sau. Giải thích nội dung trong file?

```
ospfd.conf
!
hostname ospfd
password zebra
enable password zebra
!
! Default cost for exiting an interface is 10
interface eth1
ospf cost 45
!
router ospf
! Speak OSPF on all interfaces falling in 10.0.0.0/16
network 10.0.0.0/16 area 0.0.0.0
redistribute connected
!
log file /var/log/quagga/ospfd.log
!
```

Bước 4: Khởi động **Lab3.15** từ terminal của máy host với lệnh **lstart**

Bước 5: Khảo sát sự kết nối của các router trong mạng *khi CHƯA chạy giải thuật OSPF trên*

### Quagga.

- Trên bb1, kiểm tra đường đi đến eth0 và eth1 của bb4 bằng lệnh **traceroute**. Từ bb1 đến eth0 của bb4 đi đường nào, đến eth1 đi đường nào?

```
bb1:~# traceroute 10.0.2.1
traceroute to 10.0.2.1 (10.0.2.1), 64 hops max, 40 byte packets
 1  10.0.0.2 (10.0.0.2)  12 ms  0 ms  0 ms
 2  10.0.1.2 (10.0.1.2)  0 ms  0 ms  0 ms
 3  10.0.2.1 (10.0.2.1)  0 ms  0 ms  0 ms
```

- Kiểm tra bảng vạch đường của bb1.

```
bb1:~# route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
10.0.0.0          *                255.255.255.0    U        0      0      0 eth0
10.0.3.0          *                255.255.255.0    U        0      0      0 eth1
```

- Trên eth1, eth2 của bb2 dùng lệnh **tcpdump** để bắt dữ liệu khi ping từ bb1 đến eth0 của bb4, ghi ra file. Dùng Wireshark để quan sát dữ liệu thu được và chỉ ra thông tin về đường đi của bb1 đến eth0 của bb4.

Bước 6a: Khảo sát sự kết nối của các router trong mạng khi chạy OSPF trên Quagga bằng **route**

- Khởi động OSPF trên các router với lệnh **/etc/init.d/quagga start**

```
bb1:~# /etc/init.d/quagga start
Loading capability module if not yet done.
Starting Quagga daemons (prio:10): zebra ospfd.
```

- Kiểm tra bảng vạch đường của bb1 bằng lệnh **route**. Bảng vạch đường có gì thay đổi?

```
bb1:~# route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
10.0.0.0          *                255.255.255.0    U        0      0      0 eth0
10.0.1.0          10.0.0.2         255.255.255.0    UG       20     0      0 eth0
10.0.2.0          10.0.0.2         255.255.255.0    UG       30     0      0 eth0
10.0.3.0          *                255.255.255.0    U        0      0      0 eth1
```

- Kiểm tra gói tin đến giao diện mạng eth0 trên router bb1 với **tcpdump** và Wireshark. Các câu hỏi gợi ý như ở phần RIPv2.

Bước 6b: Sử dụng dịch vụ **ospf** trên Quagga

- Sử dụng dịch vụ ospf trên bb1 với lệnh **telnet localhost ospf**
- Chạy lệnh **show ip ospf interface**. Quan sát kết quả, giải thích về lệnh này?
- Chạy lệnh **show ip ospf database router**. Quan sát kết quả, giải thích về lệnh này? Gợi ý: *Link State ID: 10.0.1.1, 10.0.2.2...*
- Chạy lệnh **show ip ospf database network**. Quan sát kết quả, giải thích về lệnh này? Gợi ý: *subnet A (10.0.0.1), B (10.0.1.2)...*
- Chạy lệnh **vttysh -e "show ip ospf interface" | egrep "eth|Cost"**. Kết quả của lệnh là gì?
- Chạy lệnh **show ip ospf route**. Quan sát kết quả, giải thích về lệnh này?

```
ospfd> show ip ospf route
===== OSPF network routing table =====
N   10.0.0.0/24      [10] area: 0.0.0.0
    directly attached to eth0
N   10.0.1.0/24      [20] area: 0.0.0.0
    via 10.0.0.2, eth0
N   10.0.2.0/24      [30] area: 0.0.0.0
    via 10.0.0.2, eth0
N   10.0.3.0/24      [40] area: 0.0.0.0
    via 10.0.0.2, eth0

===== OSPF router routing table =====
R   10.0.1.1         [10] area: 0.0.0.0, ASBR
    via 10.0.0.2, eth0
R   10.0.2.2         [20] area: 0.0.0.0, ASBR
    via 10.0.0.2, eth0
R   10.0.2.3         [10] area: 0.0.0.0, ASBR
    via 10.0.0.3, eth0
R   10.0.3.2         [30] area: 0.0.0.0, ASBR
    via 10.0.0.2, eth0

===== OSPF external routing table =====
```

- Chạy lệnh **show ip ospf database router self-originate**. Quan sát kết quả, giải thích về lệnh này?

```
ospfd> show ip ospf database router self-originate

OSPF Router with ID (10.0.3.1)

Router Link States (Area 0.0.0.0)

LS age: 1291
Options: 0x2 : *I-I-I-I-I-EI*
LS Flags: 0x1
Flags: 0x2 : ASBR
LS Type: router-LSA
Link State ID: 10.0.3.1
Advertising Router: 10.0.3.1
LS Seq Number: 80000007
Checksum: 0x288d
Length: 48
Number of Links: 2

Link connected to: a Transit Network
(Link ID) Designated Router address: 10.0.0.3
(Link Data) Router Interface address: 10.0.0.1
Number of TOS metrics: 0
TOS 0 Metric: 10

Link connected to: a Transit Network
(Link ID) Designated Router address: 10.0.3.1
(Link Data) Router Interface address: 10.0.3.1
Number of TOS metrics: 0
TOS 0 Metric: 45
```

**Bài tập 4:** Khảo sát mô hình mạng khi có sự cố để làm rõ ưu điểm của OSPF

- **Thay đổi 1:** Sự cố xảy ra trên một giao diện mạng của router
  - Sử dụng tcpdump và Wireshark để quan sát việc trao đổi dữ liệu khi các router cập nhật bảng vạch đường mới trước và sau sự cố.
  - Chọn router bất kỳ và tắt đi 1 giao diện mạng của router đó bằng lệnh **ifconfig**
  - Kiểm tra sự cập nhật bảng vạch đường trên các router bằng lệnh **show ip ospf route**. Nhận xét thời gian cập nhật bảng vạch đường so với RIPv2.
  - Kiểm tra LSDB - Link State Database trên router bằng lệnh **show ip ospf database router**. Cho nhận xét thời gian cập nhật LSDB trong trường hợp router được chọn là Designated Router hoặc trong trường hợp router không phải là DR - Designated Router.
  - Reset lại mô hình mạng.
- **Thay đổi 2:** Sự cố xảy ra trên toàn bộ router
  - Sử dụng tcpdump và Wireshark để quan sát việc trao đổi dữ liệu khi các router cập nhật bảng vạch đường mới trước và sau sự cố.
  - Chọn router bất kỳ và dùng lệnh **halt** để tắt router hoặc tắt tất cả interface trên router đó bằng lệnh **ifconfig**.
  - Nếu router đó không phải là DR thì LSDB của các router khác sẽ thay đổi như thế nào? Ngược lại, nếu router đó là DR thì LSDB của các router khác sẽ thay đổi như thế nào?
  - Reset lại mô hình mạng.

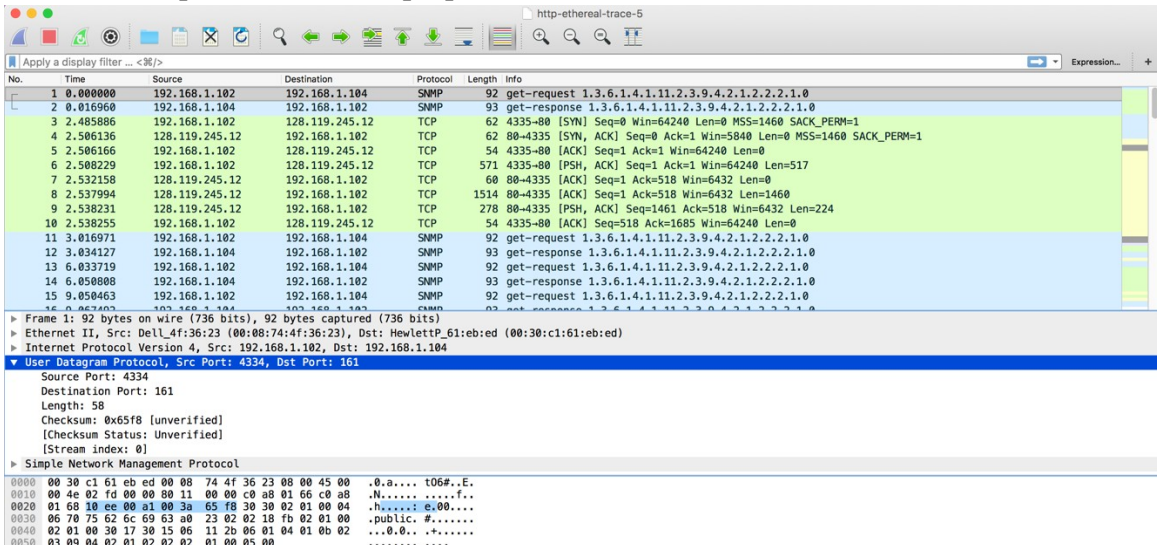
## BUỔI THỰC HÀNH 4

### Mục đích:

- Khảo sát khuôn dạng UDP segment với SNMP - Simple Network Management Protocol
- Khảo sát các đặc trưng trong giao thức TCP như: giao thức bắt tay 3 chiều, khuôn dạng TCP segment, điều khiển thông lượng trong TCP...

### I. Bài tập thực hành UDP

- SNMP là tập hợp giao thức để kiểm tra sự vận hành và quản lý từ xa các thiết bị trên mạng (router, switch, server...). SNMP sử dụng giao thức UDP để truyền tải thông tin giữa 2 thành phần chính là *Manager* và *Agent*. Ta sử dụng Wireshark để quan sát các thông điệp chứa UDP Header được trao đổi trong SNMP.
- Mở file *http-ethereal-trace-5.pcap*.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.102	192.168.1.104	SNMP	92	get-request 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.1.0
2	0.016960	192.168.1.104	192.168.1.102	SNMP	93	get-response 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.1.0
3	2.485886	192.168.1.102	128.119.245.12	TCP	62	4335->80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
4	2.506136	128.119.245.12	192.168.1.102	TCP	62	80->4335 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
5	2.506166	192.168.1.102	128.119.245.12	TCP	54	4335->80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
6	2.506229	192.168.1.102	128.119.245.12	TCP	571	4335->80 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=517
7	2.532158	128.119.245.12	192.168.1.102	TCP	60	80->4335 [ACK] Seq=1 Ack=518 Win=6432 Len=0
8	2.537994	128.119.245.12	192.168.1.102	TCP	1514	80->4335 [ACK] Seq=1 Ack=518 Win=6432 Len=1460
9	2.538231	128.119.245.12	192.168.1.102	TCP	278	80->4335 [PSH, ACK] Seq=1461 Ack=518 Win=6432 Len=224
10	2.538255	192.168.1.102	128.119.245.12	TCP	54	4335->80 [ACK] Seq=518 Ack=1685 Win=64240 Len=0
11	3.016971	192.168.1.102	192.168.1.104	SNMP	92	get-request 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.1.0
12	3.034127	192.168.1.104	192.168.1.102	SNMP	93	get-response 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.1.0
13	6.033719	192.168.1.102	192.168.1.104	SNMP	92	get-request 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.1.0
14	6.050808	192.168.1.104	192.168.1.102	SNMP	93	get-response 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.1.0
15	9.050463	192.168.1.102	192.168.1.104	SNMP	92	get-request 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.1.0

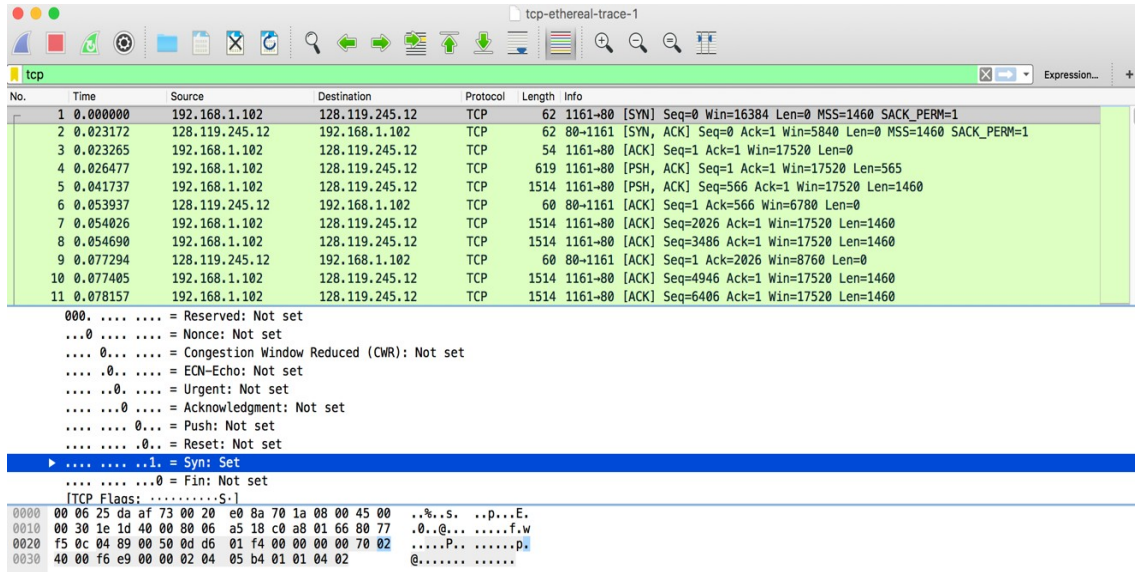
Frame 1: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0  
Ethernet II, Src: Dell\_4f:36:23 (00:08:74:4f:36:23), Dst: HewlettP\_61:eb:ed (00:30:c1:61:eb:ed)  
Internet Protocol Version 4, Src: 192.168.1.102, Dst: 192.168.1.104  
User Datagram Protocol, Src Port: 4334, Dst Port: 161  
Source Port: 4334  
Destination Port: 161  
Length: 58  
Checksum: 0x65f8 [unverified]  
[Checksum Status: Unverified]  
[Stream index: 0]  
Simple Network Management Protocol  
0000 00 30 c1 61 eb ed 00 08 74 4f 36 23 00 00 45 00 .0.a....t06#..E.  
0010 00 4e 02 fd 00 00 00 11 00 00 c0 a8 01 66 c0 a8 .N.....f..  
0020 01 68 10 ee 00 a1 00 30 65 f8 30 30 02 01 00 04 .h.....e.00..  
0030 06 70 75 62 6c 69 63 a0 23 02 02 18 fb 02 01 00 .public.#.....  
0040 02 01 00 30 17 30 15 06 11 2b 06 01 04 01 0b 02 ..0.0.#.....  
0050 03 09 04 02 01 02 02 02 01 00 05 00 .....0.....

- Chọn 1 gói tin bất kỳ. Có bao nhiêu trường (field) trong UDP Header của gói tin? Mỗi trường có độ dài là bao nhiêu bytes?
- Giá trị của trường Length trong UDP Header bằng bao nhiêu và thể hiện thông tin gì? Cách tính ra giá trị của trường Length này?
- Hãy chỉ ra thành phần UDP Payload của gói tin? UDP Payload chứa tối đa bao nhiêu bytes?
- Trình bày cách tính Checksum của UDP Header. Gợi ý: tham khảo cách tính trên wiki: [https://en.wikipedia.org/wiki/User\\_Datagram\\_Protocol](https://en.wikipedia.org/wiki/User_Datagram_Protocol)
- Trong cách tính Checksum vừa nêu, thành phần nào trong gói tin là IPv4 Pseudo Header?
- Dựa trên lý thuyết vừa trình bày, kiểm tra thử Checksum của UDP Header của gói tin bất kỳ.

### V. Bài tập thực hành TCP

- Mở file *tcp-ethereal-trace-1.pcap*.
- Lọc dữ liệu với *"tcp"* trong giao diện *Filter* của Wireshark

- Các gói tin thực hiện **giao thức bắt tay 3 chiều (three way handshake)** giữa Client và Web Server?
- Trình bày cơ chế bắt tay 3 chiều và vẽ hình minh hoạ cho trường hợp này.
- Cho biết IP và port của Client và Server.
- Loại bỏ hiển thị các thông điệp thuộc giao thức HTTP bằng cách: Chọn **Analyze**, chọn tiếp **Enabled Protocols**, tìm và bỏ lựa chọn giao thức **HTTP**.
- Sequence Number của **TCP SYN segment** để khởi tạo kết nối giữa Client-Server? Thành phần nào trong segment này chỉ ra đây là **TCP SYN segment**?



- Sequence Number của **TCP SYN ACK segment** để trả lời từ Server cho client? Thành phần nào trong segment này chỉ ra đây là **TCP SYN ACK segment**?
- Giá trị của trường Acknowledgement trong **SYN ACK segment** là gì? Làm sao Server có thể xác định được giá trị này?
- Sequence Number của TCP segment lệnh POST của HTTP là bao nhiêu?
- Bắt đầu từ TCP segment có chứa lệnh POST của HTTP, cho biết cơ chế điều khiển thông lượng của Server diễn ra như thế nào. Vẽ hình minh hoạ.
- Lập bảng như sau cho 6 TCP segment dữ liệu đầu tiên từ client

Segment	Packet Number	Sequence Number	Length	Time Sent	Time ACK received	Round Trip Time
1	4	1	565	0.026477	0.053937	0.02746
2						
3						
4						
5						
6						

- Giá trị nhỏ nhất của **buffer space (receiver window)** mà Server quảng bá (advertised) cho Client? Sau bao lâu thì kích thước của buffer space đạt giá trị tối đa, và bằng bao nhiêu?
- Trong quá trình truyền dữ liệu, Client có bị tắc nghẽn do thiếu hụt **buffer space** từ Server không?

- Trong quá trình truyền dữ liệu, Client có truyền tải lại segment nào không. Gợi ý: Chọn ***Statistics***, chọn tiếp ***TCP Steam Graph***, chọn ***Time Sequence (Stevens)***, ta nhận được 1 đồ thị theo *sequence number* và *time*. Dựa vào đồ thị để trả lời.



## BUỔI THỰC HÀNH 5

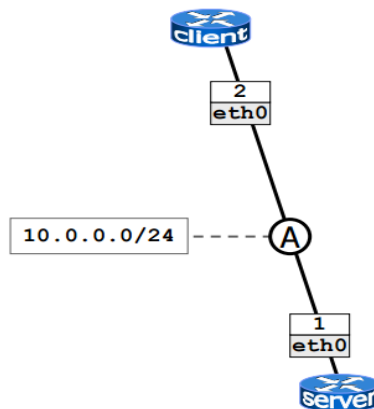
### Mục đích

- Minh họa giao thức HTTP qua mô hình Client – Web Server
- Minh họa hoạt động của DNS.
- Minh họa hoạt động của hệ thống Mail với các giao thức SMTP, POP3, IMAP.

### I. Giao thức HTTP

#### *Bài tập 1: Mô hình Client - Server với giao thức HTTP*

Bước 1: Xây dựng mô hình mạng như sau. Thư mục chứa cấu hình mạng là **Lab5.1**. Chọn một máy ảo làm Client và một máy ảo làm Server.



Bước 2: Khởi động mô hình mạng bằng lệnh **lstart**.

Bước 3: Trên máy ảo làm Server, khởi động **Apache2** bằng lệnh **/etc/init.d/apache2 start**. Kiểm tra tình trạng Apache2 bằng lệnh **/etc/init.d/apache2 status**.

Bước 4:

- Nếu Apache2 đã hoạt động, thì sẽ mở ra trang HTML mặc định là **/var/www/index.html** có nội dung **"It works!"** khi Client truy cập đến bằng lệnh **links**. Trên giao diện của Client, F10 để chuyển tới **Menu Bar**, chọn **"Go to URL"**, và nhập địa chỉ của **http://10.0.0.1/** để kết nối đến. Nếu như nhận được nội dung **"It works!"** là Server đã trả lời thành công cho Client.
- Một số lệnh phía Server được sử dụng để kiểm tra truy nhập từ Client:
  - o Lệnh **tail -f /var/log/apache2/access.log**. Nhận xét kết quả.
  - o Lệnh **tail -f /var/log/apache2/error.log**. Nhận xét kết quả.
- Trên Server sử dụng lệnh **tcpdump** kết hợp Wireshark để phân tích dữ liệu gửi từ Client sang.

Câu hỏi gợi ý:

- o Lọc dữ liệu hiển thị với **http** trên Wireshark.
- o Các ngôn ngữ hỗ trợ để hiển thị nội dung trang web?
- o Status code trả về từ Server cho Client? Độ dài thông điệp của Server gửi cho Client.
- o Thời điểm chỉnh sửa lần cuối của file **index.html** trên Server? Thể hiện qua thông tin gì?

- Thay đổi nội dung file index.html trên Server. Nội dung mới chứa nhiều thông tin phức tạp hơn, phức tạp hơn. Thực hiện phân tích dữ liệu với Wireshark để thấy sự khác nhau giữa download trang html đơn giản và trang html phức tạp như thế nào.

**Bài tập 2: Mô hình Client - Server có xử lý thông tin**

- Trên Server xây dựng một form để Client nhập thông tin và lựa chọn giới tính. Khi Client bấm **Submit** thì Server in ra lời chào: **“Have a nice day, Mr/Mrs/Unknown <lastname> <firstname>”**.

Personal information: \_\_\_\_\_

First name:

Last name:

☒ Male  
☐ Female  
☐ Other

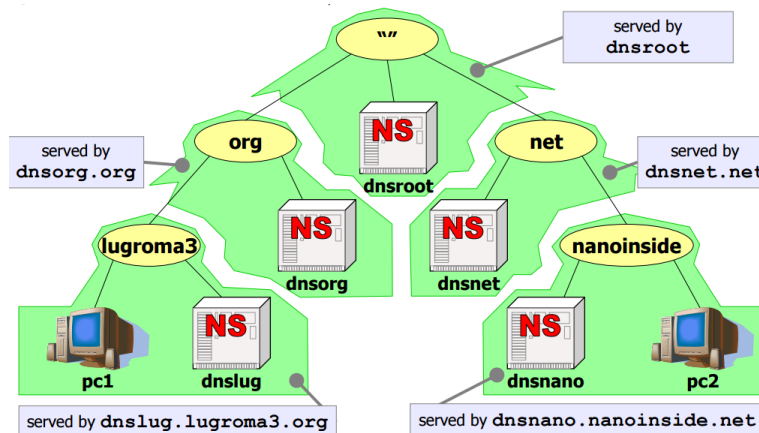
- Dùng lệnh **tcpdump** và Wireshark để phân tích dữ liệu trao đổi giữa Client và Server.

## VI. Dịch vụ DNS

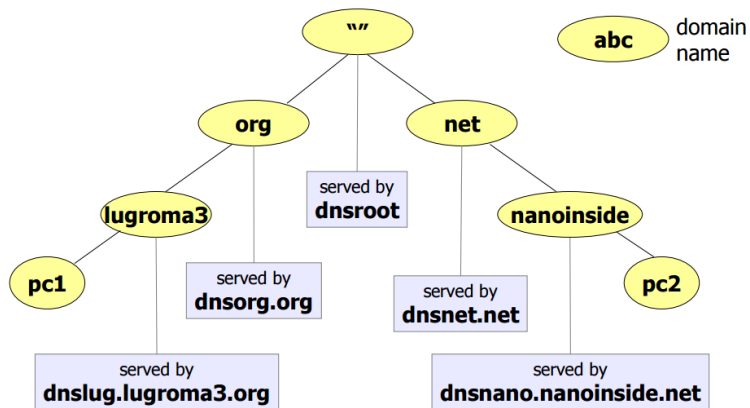
**Bài tập 1: Mô phỏng dịch vụ DNS**

Bước 1: Khảo sát mô hình DNS cần xây dựng

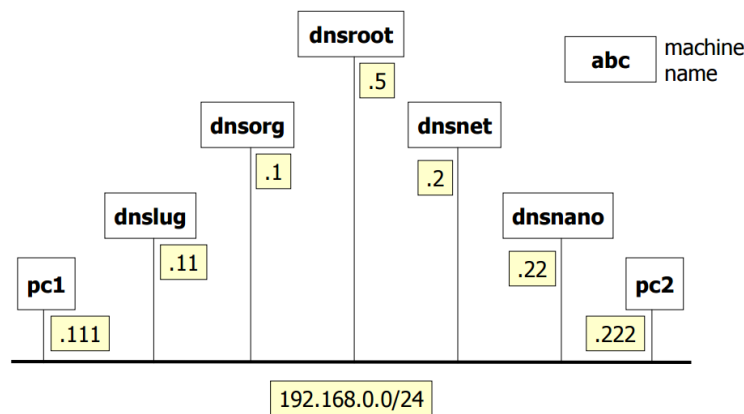
- Mô hình DNS phân chia theo các zone. Mỗi zone sẽ có 1 **Name Server (NS)**.



- Mô hình DNS phân chia theo cấu trúc hình cây



- Mô hình DNS phân chia theo địa chỉ IP



Bước 2: Cấu hình cho các thiết bị qua các file *.startup*, *lab.conf*

- Tạo thư mục **Lab5.3** làm để chứa mạng ảo. Cấu trúc thư mục được tổ chức như sau:

```
$NETKIT_HOME
----Lab5.20      (folder)
----dnslug      (folder)
----dnsnano     (folder)
----dnsnet      (folder)
----dnsorg      (folder)
----dnsroot     (folder)
----pc1         (folder)
----pc2         (folder)
----dnslug.startup (file)
----dnsnano.startup (file)
----dnsnet.startup (file)
----dnsorg.startup (file)
----dnsroot.startup (file)
----pc1.startup   (file)
----pc2.startup   (file)
----lab.conf      (file)
```

- Trong *pc1.startup* và *pc2.startup*, thêm vào nội dung: */sbin/ifconfig <InterfaceName> <IpAddress> up*
- Trong *dnslug.startup*, *dnsnano.startup*, *dnsnet.startup* và *dnsorg.startup*, thêm vào nội dung: */sbin/ifconfig <InterfaceName> <IpAddress> up*

*/etc/init.d/bind start*

- Trong *lab.conf*, hình trạng mạng được miêu tả như sau:

```
pc1[0]="A"
pc1[mem]=24
dnslug[0]="A"
dnslug[mem]=32
dnsorg[0]="A"
dnsorg[mem]=32
dnsroot[0]="A"
dnsroot[mem]=32
dnsnet[0]="A"
dnsnet[mem]=32
dnsnano[0]="A"
dnsnano[mem]=32
pc2[0]="A"
pc2[mem]=24
```

Bước 3: Xây dựng cấu trúc thư mục cho pc1 và pc2 như sau

```
$NETKIT_HOME
----Lab5.20                                (folder)
|
|----dnslug                                (folder)
|
|----dnsnano                              (folder)
|
|----dnsnet                               (folder)
|
|----dnsorg                               (folder)
|
|----dnsroot                              (folder)
|
|----pc1                                  (folder)
|
|----etc                                  (folder)
|
|    ----resolv.conf                      (file)
|
|----pc2                                  (folder)
|
|----etc                                  (folder)
|
|    ----resolv.conf                      (file)
```

- Trong *resolv.conf* của pc1 và pc2, thêm vào nội dung  
*nameserver <IpAddressOfNameServer>*  
*search <SuffixName>*

Bước 4: Cấu hình cho các Name Server.

- Sinh viên sử dụng phần cấu hình được cung cấp bởi giáo viên. Trong phần khảo sát dưới đây, Name Server *dnslug* được chọn để khảo sát các thông tin cấu hình

```
$NETKIT_HOME
----Lab5.20                                (folder)
|
|----dnslug                                (folder)
|
|----etc                                  (folder)
|
|    ----bind                             (folder)
|
|        ----named.conf                   (file)
|
|        ----db.root                      (file)
|
|        ----db.org.lugroma3              (file)
|
|----dnsnano                              (folder)
|
|----dnsnet                               (folder)
|
|----dnsorg                               (folder)
|
|----dnsroot                              (folder)
|
|----pc1                                  (folder)
|
|----pc2                                  (folder)
```

- Cấu hình của **Zone** và **Name Server**: Trong *dnslug/etc/bind/named.conf*, cho biết thông tin thể hiện cho **Root Name Server**, thông tin thể hiện bảng lưu trữ tên trong miền (**Name Database**), thông tin thể hiện *dnslug* là **Primary Master** cho zone *lugroma3.org*
- Cấu hình thông tin của **Root Name Server**: Trong *dnslug/etc/bind/db.root* của *dnslug*, cho biết các resource record có ý nghĩa gì, biết rằng các resource record đó có format như sau:

**format of a resource record**  
**<domain> <class> <type> <rdata>**  
**domain:** the record owner (=domain to which the record refers)  
**class:** usually IN (=Internet system); may be HS (=hesiod) or CH (=chaos)  
**type:** see next slide...  
**rdata:** record data (depends on the record type)

- Cấu hình thông tin chứng thực (Authoritative information): Trong *dnslug/etc/bind/db.org.lugroma3*, các thông tin **Authority Record** có nghĩa gì?

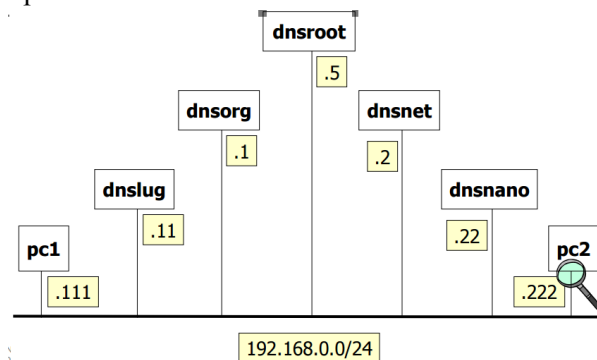
```
@ IN SOA dnslug.lugroma3.org. root.dnslug.lugroma3.org. (
    2006031201 ; serial
    28 ; refresh
    14 ; retry
    3600000 ; expire
    0 ; negative cache ttl
)
```

- Kết nối giữa logic name và địa chỉ IP: Trong *dnslug/etc/bind/db.org.lugroma3*, quan sát kết nối giữa logic name và IP đã được thực hiện. Cho biết trong zone mà *dnslug* làm **Name Server**, các máy được gán giữa logic name và IP ra sao.

```
@ IN NS dnslug.lugroma3.org.
dnslug IN A 192.168.0.11
pc1 IN A 192.168.0.111
```

Bước 5a: Khảo sát DNS khi kết nối đến địa chỉ có tồn tại trong mạng

- Trên pc2, dùng lệnh *tcpdump* với cú pháp: *tcpdump -n -t port domain -w <FileName.pcap>*.
- Trên pc1, **ping** đến *pc2.nanoinside.net*.



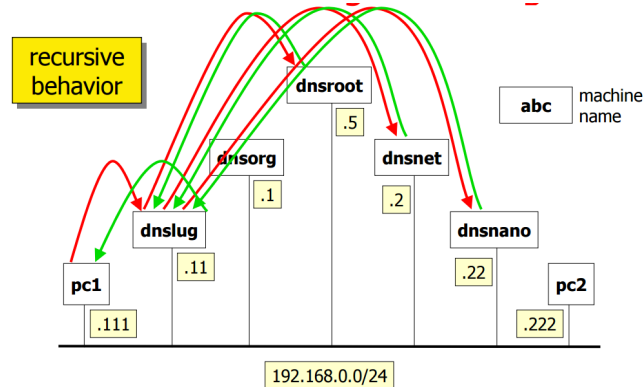
- Mở Wireshark với file ghi nhận được để khảo sát quá trình trao đổi thông tin giữa các Name Server của các Zone để *pc1* có thể ping đến *pc2*. Câu hỏi gợi ý: Gói tin thuộc loại nào (**Query** hay **Answer**). Gói tin được trao đổi giữa máy nào với nhau (dựa vào IP, dựa vào port). Nội dung trao đổi trong gói tin là gì?
  - o Ví dụ gói tin **Query**. Gói tin này là truy vấn từ máy nào đến máy nào, trên cổng nào? Giá trị A, *pc2.nanoinside.net*, 36 có ý nghĩa gì?

```
IP 192.168.0.111.3072 > 192.168.0.11.53:
29753+ A? pc2.nanoinside.net. (36)
```

- Ví dụ về **Answer**. Gói tin này là trả lời từ máy nào đến máy nào, trên cổng nào? Giá trị 0, 1, 2 có ý nghĩa gì?

```
IP 192.168.0.5.53 > 192.168.0.11.3073:  
18164 0/1/2 (84)
```

- Đánh số thứ tự cho quá trình trao đổi thông điệp (các đường màu xanh, đỏ) giữa Name Server trong các Zone để tìm ra địa chỉ pc2 mà pc1 muốn liên kết tới:



- Dùng **Sequence Diagram** miêu tả quá trình trao đổi thông điệp giữa các Name Server trong các Zone để pc1 ping được đến pc2.

Bước 5b: Khảo sát lại hoạt động của DNS để minh họa về tính năng của **Name Server cache**.

- Thực hiện lại khảo sát giống bước 5a. Việc trao đổi thông điệp giữa các Name Server trong các Zone quan sát được trên Wireshark có gì thay đổi? Sự thay đổi này có ý nghĩa gì?
- Vẽ lại hình cho lần khảo sát này.

Bước 6a: Khảo sát hoạt động của DNS khi kết nối đến địa chỉ KHÔNG tồn tại trong mạng

- Restart lại DNS trên các Name Server bằng lệnh `/etc/init.d/bind restart`
- Trên pc2, dùng lệnh `tcpdump` với cú pháp: `tcpdump -n -t port domain -w <FileName.pcap>`
- Từ pc1 **ping** đến pluto.nanoinside.net.
- Mở Wireshark với file ghi nhận được để khảo sát quá trình trao đổi thông tin giữa các Name Server của các Zone để pc1 cố gắng truyền dữ liệu đến pluto.nanoinside.net
- Dùng **Sequence Diagram** miêu tả quá trình trao đổi thông điệp giữa các Name Server trong các Zone để pc1 cố gắng ping đến pluto.nanoinside.net

Bước 6b: Khảo sát lại hoạt động của DNS khi kết nối đến địa chỉ KHÔNG tồn tại trong mạng để minh họa về tính năng của **Name Server negative cache**

- Thực hiện lại khảo sát giống bước 6a. Việc trao đổi thông điệp giữa các Name Server trong các Zone quan sát được trên Wireshark có gì thay đổi? Sự thay đổi này có ý nghĩa gì?
- Vẽ lại hình cho lần khảo sát này.

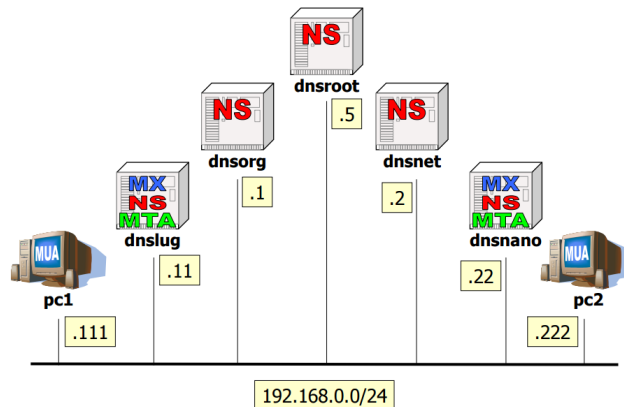
## VII. Dịch vụ Mail

### Bài tập 1: Mô phỏng dịch vụ gửi và nhận mail

MX - Mail Exchanger làm nhiệm vụ tập hợp và lưu trữ các mail chuyển đến trong miền quản lý qua giao thức **POP3 (port 110)** và **IMAP (port 143)**. Vì vậy MX còn gọi là **Incoming Mail Server**. Người dùng lấy mail qua MX phải thực hiện cơ chế chứng thực. Phần mềm **ipop3d**, **imapd** thuộc **inetd** được cài đặt trên Name Server để hoạt động như 1 MX.

MTA – Mail Transfer Agent làm nhiệm vụ giúp người dùng trong miền chuyển thư đến các địa chỉ mong muốn thông qua giao thức SMTP tại cổng 25. Vì vậy, MTA còn được gọi là **Outcoming Mail Server**. Khác với MX, người dùng không cần thực hiện chứng thực trên MTA. Tuy nhiên, trên MX cần được cài đặt cơ chế **relaying** cho phép chuyển tiếp mail từ 1 máy trong miền đến 1 máy nằm ở miền khác, ví dụ: gửi mail từ pc1 qua pc2. Trong bài thực hành này, phần mềm **exim4** sẽ được cài đặt để hoạt động như 1 MX.

Bước 1: Sử dụng mô hình DNS đã được xây dựng



- dnslug và dnsnano được cài đặt các vai trò sau:
  - o 1 **Name Server** trong phạm vi zone (lugroma3.org, nanoinside.net).
  - o 1 **Mail Exchanger (MX)** cho domain (lugroma3.org, nanoinside.net).
  - o Hoạt động như 1 **Mail Transfer Agent (MTA)**.
  - o Có thể kiểm tra các gói phần mềm MX và MTA đang chạy trên Name Server bằng lệnh **ps ax** thực hiện trên **dnslug**.
- Tại pc1 và pc2 nằm trong miền, ta cài đặt **Mail User Agent (MUA)**. MUA là 1 phần mềm cho phép kết nối đến Mail Server và quản lý hộp thư đến, hộp thư đi riêng tư, ví dụ : **Outlook, Mozilla, Thunderbird...** Trong bài thực hành này, MUA được lựa chọn và cài đặt lên pc1 và pc2 là gói phần mềm **pine**, được phát triển và phát hành bởi đại học Washington. Đây là 1 MUA nhỏ gọn, hoạt động ổn định trong môi trường Linux, thích hợp với các máy ảo có kích thước nhỏ do Netkit tạo ra.
  - o Lệnh để đăng nhập vào MUA trên pc là **pine**, mật khẩu là **guest**. Sau khi đăng nhập thành công sẽ di chuyển đến MAIN MENU của MUA.

```

PINE 4.64  MAIN MENU                               Folder: INBOX  No Messages

?  HELP                      - Get help using Pine
C  COMPOSE MESSAGE           - Compose and send a message
I  MESSAGE INDEX              - View messages in current folder
L  FOLDER LIST                - Select a folder to view
A  ADDRESS BOOK               - Update address book
S  SETUP                      - Configure Pine Options
Q  QUIT                       - Leave the Pine program
    
```

- o Chọn SETUP từ giao diện MAIN MENU, sau đó chọn (C) Config, quan sát và hiểu cơ bản các cấu hình có thể tùy chọn ở đây



```

PINE 4.64  MAIN MENU                               Folder: INBOX  No Messages
personal-name = Root user on PC1
user-domain   = lugroma3.org
smtp-server   = mail.lugroma3.org
nntp-server   = <No Value Set>
inbox-path    = {imap.lugroma3.org/user=guest}inbox
incoming-archive-folders = <No Value Set>
pruned-folders = <No Value Set>
default-fcc    = <No Value Set: using "sent-mail">
default-saved-msg-folder = <No Value Set: using "saved-messages">
postponed-folder = <No Value Set: using "postponed-msgs">
read-message-folder = <No Value Set>
form-letter-folder = <No Value Set>
literal-signature = <No Value Set>
signature-file = <No Value Set: using ".signature">
feature-list   =

```

- Có 1 nghịch lý là tài khoản người dùng duy nhất trên pc1 có là root, tài khoản để đăng nhập vào pine của pc1 chỉ có duy nhất guest, nếu người dùng root trên pc1 gửi mail đến máy khác, định dạng mail sẽ là root@..., việc thực hiện REPLY trên máy khác sẽ không thành công vì địa chỉ mail root@...là không tồn tại (chỉ tồn tại guest@...trên pine). Vì vậy, ta phải HOẶC tạo thêm tài khoản người dùng guest trên pc1 HOẶC cấu hình trên pine để tự động reply đúng địa chỉ (cách được sử dụng). Việc cấu hình rất đơn giản, chỉ cần chọn X cho trường *reply-always-uses-reply-to* và đặt thông số cho *customized-hdrs* = *Reply-To: guest@lugroma3.org*

```

PINE 4.64  MAIN MENU                               Folder: INBOX  No Messages
[ ] strip-whitespace-before-send
[ Reply Preferences ]
[ ] enable-reply-indent-string-editing
[ ] include-attachments-in-reply
[ ] include-header-in-reply
[ ] include-text-in-reply
[X] reply-always-uses-reply-to
[ ] signature-at-bottom
[ ] strip-from-sigdashes-on-reply
[ Sending Preferences ]
[ ] disable-sender
[ ] enable-8bit-esmtp-negotiation
[ ] enable-background-sending
[ ] enable-delivery-status-notification
[ ] enable-verbose-smtp-posting
[ ] fcc-on-bounce

[ ] unselect-will-not-advance
[ ] use-current-dir
[ ] use-regular-startup-rule-for-stayopen-folders
[ ] use-subshell-for-suspend
initial-keystroke-list = <No Value Set>
default-composer-hdrs  = <No Value Set>
customized-hdrs        = Reply-To: guest@lugroma3.org

```

- Tại *dnsorg*, *dnsnet* và *dnsroot*, cài đặt không có nhiều thay đổi đáng kể khác với việc cài đặt dịch vụ DNS trước đó,  
Bước 2 : Khảo sát việc gửi mail trong mô hình mạng
- Dùng lệnh tcpdump trên *dnsroot* để bắt dữ liệu được truyền tải trên mạng và lưu vào file .pcap, thực hiện phân tích với Wireshark với kịch bản gửi email như dưới đây. Việc phân tích cần chỉ rõ được quá trình diễn tiến của kịch bản với các lệnh được quy định trong giao thức.
- Trên pc1, đăng nhập *pine* với tài khoản *guest*, soạn 1 email có nội dung đơn giản, ví dụ : *Xin chào, tôi là pc1. Tam biệt !!* và chuyển đến pc2 thông qua địa chỉ [guest@lugroma3.org](mailto:guest@lugroma3.org)

PINE 4.64 MAIN MENU			Folder: INBOX No Messages
?	HELP	-	Get help using Pine
C	COMPOSE MESSAGE	-	Compose and send a message
I	MESSAGE INDEX	-	View messages in current folder
L	FOLDER LIST	-	Select a folder to view
A	ADDRESS BOOK	-	Update address book
S	SETUP	-	Configure Pine Options
Q	QUIT	-	Leave the Pine program

- Trên *Incoming Mail Server (dnsmail)* của pc2, dùng lệnh `cat /var/spool/mail/guest` để quan sát mail đang được lưu trữ và chờ pc2 đăng nhập và lấy về. Nội dung lấy được tương tự như hình sau :

```
dnsmail:~# cat /var/spool/mail/guest
From root@lugroma3.org Fri Apr 20 15:10:41 2007
Return-path: <root@lugroma3.org>
Envelope-to: guest@nanoinet.net
Delivery-date: Fri, 20 Apr 2007 15:10:41 +0200
Received: from [192.168.0.11] (port=4902 helo=dnsmail)
        by dnsmail with esmtp (Exim 4.50)
        id 1Hssr-00004N-OW
        for guest@nanoinet.net; Fri, 20 Apr 2007 15:10:41 +0200
Received: from [192.168.0.111] (port=3119 helo=pc1.lugroma3.org)
        by dnsmail with esmtp (Exim 4.50)
        id 1Hssr-00004O-7M
        for guest@nanoinet.net; Fri, 20 Apr 2007 15:10:41 +0200
Date: Fri, 20 Apr 2007 15:10:41 +0200 (CEST)
From: Root user on PC1 <root@lugroma3.org>
X-X-Sender: root@pc1
Reply-To: guest@lugroma3.org
To: guest@nanoinet.net
Subject: Test message
Message-ID: <Pine.LNX.4.64.0704201510200.254@pc1>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII; format=flowed

This is a test message.
Goodbye ;-)
```

- Sử dụng Sequence Diagram để mô phỏng lại quá trình truyền tải email từ pc1 sang Incoming Mail Server của pc2, biết rằng có 3 thực thể cơ bản tham gia vào quá trình này, gồm có : pc1, dnsmail, dnsmail.
- Bước 3 : Khảo sát việc nhận mail trong mô hình mạng
- Thực hiện lại lệnh tcpdump trên *dnsmail* để bắt dữ liệu được truyền tải trên mạng và lưu vào file .pcap, thực hiện phân tích với Wireshark với kịch bản nhận email như dưới đây. Việc phân tích cần chỉ rõ được quá trình diễn tiến của kịch bản với các lệnh được quy định trong giao thức.
- Trên pc2, đăng nhập pine với tài khoản guest, kiểm tra hộp thư đến

PINE 4.64 MAIN MENU		Folder: INBOX No Messages
?	HELP	- Get help using Pine
C	COMPOSE MESSAGE	- Compose and send a message
I	MESSAGE INDEX	- View messages in current folder
L	FOLDER LIST	- Select a folder to view
A	ADDRESS BOOK	- Update address book
S	SETUP	- Configure Pine Options
Q	QUIT	- Leave the Pine program

- Sử dụng Sequence Diagram để mô phỏng lại quá trình chứng thực và nhận email từ pc1 trên Incoming Mail Server của pc2, biết rằng có 2 thực thể cơ bản tham gia vào quá trình này, gồm có : dnsnano và pc2.

**Bài tập 2:** Thực hiện khảo sát với 1 kịch bản khác (không bắt buộc)

- Tạo 1 tài khoản người dùng mới trên dnsnano bằng lệnh adduser.
- Gửi 1 email từ pc1 sang người dùng mới này.
- Thay đổi cấu hình của pine trên pc2 để có thể login được vào **Incoming Mail Server (dnsnano)** bằng tài khoản người dùng mới này.
- Mở mail và reply lại mail đã nhận được.
- Kiểm tra các phiên giao dịch (**smtp, imap, pop3**) bắt được trên dnsroot bằng Wireshark.