



SAPIENZA  
UNIVERSITÀ DI ROMA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI ROMA LA SAPIENZA

CORSO DI ENGINEERING IN COMPUTER SCIENCE

NETKIT LAB



*Author:*

Andrea LOMBARDO

*Teacher:*

Marco SPAZIANI

ACADEMIC YEAR 2019-2020

---

# INDICE

<b>1</b>	<b>Lecture 1</b>	<b>3</b>
1.1	DEFINITIONS . . . . .	3
1.2	Difference between Broadcast domain vs collision domain . . . . .	6
1.3	Exercise-LAB0 . . . . .	6
1.3.1	Application of the lab . . . . .	7
1.4	IP TUNNELLING . . . . .	12
<b>2</b>	<b>2 lecture</b>	<b>12</b>
2.1	How to solve problem of storing information . . . . .	12
2.2	2 fase . . . . .	14
2.2.1	TAP . . . . .	16
2.3	DHCP . . . . .	18
<b>3</b>	<b>3LAB</b>	<b>19</b>
3.1	Opens Shortest Path First . . . . .	19
<b>4</b>	<b>4LAB</b>	<b>23</b>
4.1	SSH . . . . .	24
4.2	Description of SSH . . . . .	25
4.3	Authentication phase . . . . .	25
4.4	In practice . . . . .	26

# 1

---

## LECTURE 1

### 1.1 DEFINITIONS

The IP networking protocol understands addresses as 32-bit numbers. Each machine must be assigned a number unique to the networking environment. If you are running a local network that does not have TCP/IP traffic with other networks, you may assign these numbers according to your personal preferences. There are some IP address ranges that have been reserved for such private networks. However, for sites on the Internet, numbers are assigned by a central authority, the Network Information Center (NIC). Network interface card: can implement many protocols (even bluetooth). It interacts with the OS with drivers (can virtualize NIC, the network interface card we'll use).

**IP ADDRESS** is on 32 BIT in case we refer to IPv4 otherwise if we refer to IPv6 it's 128 BIT. It identifies a host in the subnet, more precisely it's assigned to each device connected to a computer network that uses the Internet Protocol for communication. An IP address serves two main functions: host or network interface identification and location addressing. An IPv4 address and its subnet mask may be 192.0.2.1 and 255.255.255.0, respectively. IP addresses are split into a network number, which is contained in the leading octets, and a host number, which is the remainder.

**SUB MASK** is composed on 32 BIT and it's identified by the id for example (192.168.0.0/24) where the 24 is the prefix and gives us information on how much host I can have into my subnet.

The term subnet mask is only used within IPv4. How many hosts can I have?

$2^{32-\text{SUB-PREFIX}}$  (in our case 24) - 1 = 255 Possible addresses where 0 for the sub-net and 255 for the BROADCAST.

MASK 255.255.255.0

How can I recover the sub-id through the ip addr and subMask? Answer: Through AND logic IP AND WITH (AND) SUB MASK = SUB ID

A mechanism is needed to map IP addresses onto the addresses of the underlying net-

work. The mechanism used is the Address Resolution Protocol (ARP). In fact, ARP is not confined to Ethernet or Token Ring, but is used on other types of networks, . The idea underlying ARP is exactly what most people do when they have to find Mr. X in a throng of 150 people: the person who wants him calls out loudly enough that everyone in the room can hear her, expecting him to respond if he is there. When he responds, she knows which person he is. When ARP wants to find the Ethernet address corresponding to a given IP address, it uses an Ethernet feature called broadcasting, in which a datagram is addressed to all stations on the network simultaneously. The broadcast datagram sent by ARP contains a query for the IP address. Each receiving host compares this query to its own IP address and if it matches, returns an ARP reply to the inquiring host. The inquiring host can now extract the sender's Ethernet address from the reply. e to request specific ARP addresses from hosts on your network, and should it be necessary, network administrators can also modify, add, or remove ARP entries from their local cache. Also **SUB ID** is 32 BIT

Without wasting resource, so IP address we have 31 different host but in Cisco standard they described that in general we have 30 different IP address. Routing algorithms in the context of networking can be classified variously. The prior classification is based on the building and modification of a routing table. This can be done in two manners statically or dynamically. Once a host has discovered an Ethernet address, it stores it in its ARP cache so that it doesn't have to query for it again the next time it wants to send a datagram to the host in question.

**Static routing** • the table is set up and modified manually

- Static routing does not involve any change in routing table unless the network administrator changes or modify them manually. Static routing algorithms function well where the network traffic is predictable. This is simple to design and easy to implement. There is no requirement of complex routing protocols.
- Static routing is also known as non-adaptive routing which enables a pre-computed route to be fed into the routers offline. The administrative distance is a metric to measure the trustworthiness of the information received from a router.
- Static routes can be considered as an efficient method for a small and simple network that does not change frequently.

**Dynamic routing** • the table is built automatically with the help of the routing protocols

- Dynamic routing is a superior routing technique which alters the routing information according to the altering network circumstances by examining the arriving routing update messages. When the network change occurs, it sends out a message to the router to specify that change, then the routes are recalculated and sent

as a new routing update message. These messages pervade the network, enabling the router to change their routing tables correspondingly.

- Dynamic routing or otherwise called as adaptive routing. The routing decisions are altered in these algorithms to mirror the changes in the topology or traffic. There are various adaptive algorithms which can be classified according to the source of information

**Considerations:** Dynamic routing is preferred over static routing because of the major issue in static routing where in case of link/node failure the system cannot recover. The dynamic routing overcomes from the static routing limitations.

GATEWAY IN OUR SUBNET -THE HOST THAT KNOWS HOW TO SEND PACKET TO OTHER HOST. WE'VE A DEFAULT GATEWAY, USED WHEN I'VE NOT A RULE FOR GATWAY PACKET, WHEN THERE'RE A NETWORK DESTINATION 0.0.0.0 AND NETMASK IS 0.0.0.0 .



## General rule of thumb

$Pc_x$  network has  $y$  hosts.

We compute  $2^z - 2 = w$ .

With  $w$  addresses we can cover  $y$  hosts ( $w > y$ ) so  $32 - z = j$  bits  
subnet mask associated to  $Pc_x$  network is 195.10.0.something

**Content delivery network** aims to shortest latency .The **next hop** is the next possible destination for a packet.

To operate several Ethernets, you have to split your network into subnets. Note that subnetting is required only if you have more than one broadcast network—point-to-point links don't count

NIC IS A PHYSICAL DEVICE THROUGH IT INTERACT WITH OS VIA DRIVERS  
NETKIT IS A SOFTWARE AND EMULATE COMPUTER NETWORK

## 1.2 DIFFERENCE BETWEEN BROADCAST DOMAIN VS COLLISION DOMAIN

VV Broadcast domain: portion of the network when the packets can be received to all the recipients within the defined network boundary.

Collision domain: portion of the network where the packets can collide when sent across a shared media.

A Switch is a single broadcast domain (all interfaces of a switch belong to a common single broadcast domain). and multi collision domain (every interface of a switch has one collision domain).

Broadcast domain in switch can be split to many using the concept of VLAN.

In routers, each interface in a router is a broadcast domain. similarly each interface in a router has a collision domain.

## 1.3 EXERCISE-LAB0

Here we can see the basic routines:

```
vstart [options] VM-NAME #starts a netkit virtual
                           #machine called VM-NAME

vcrash [options] VM-NAME #crash a running netkit
                           #virtual machine called VM-NAME

lstart #starts a netkit laboratory sequentially

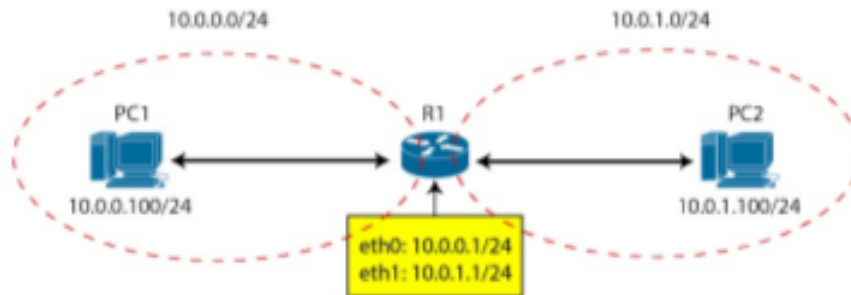
lstart -p0 #laboratory parallel startup

lcrash #crash a netkit laboratory
```

**pwd** let to know who I am-list of processes run, take care to not kill init process because it's that process which spawn all processes.(it's like a root of a tree).

**WARNING:** Whenever we start a VM Netkit creates a VMname.disk. Every time you create a file on the NVM and I crash it with vcrash this file will be deleted -> be careful with it

We want to reproduce the following network through the NETKIT SOFTWARE



In order to simulate network netkit has laboratories which specifies how many network and how they are connected together. The basic elements are:

**Lab directory** is where all the files of the lab are contained such lab.conf and VM directories

**VM directories** for each VM on the virtual network we must have a folder named as the VM inside the Lab directory

### lab.conf

We use a convention for IP address and subnet: 192.168.0.1/24. In general we will use the following schemes:

- a.b.c.0/m to define the subnet
- a.b.c.1/m to define the subnet gateway

#### 1.3.1 APPLICATION OF THE LAB

In this network topology we have 2 collision domains (at left we have A, right we have B). We also have:

- 2 broadcast domains because I have a router (overlaps with collision domains because I have router)
- 10.0.0.100/24 is the IP address of PC1, each PC has 1 interface.
- R1 is attached to 2 subnets 2 IP addresses.

First we should build a directory for a unique directory for the project through

**mkdir filename** routine than enter into it through **cd filename** routine and then start building a **file.conf** that say us information on how is the topology of the network. Then to do this work we should create a directory for each virtual machine so, **rc1** represent the router, **pc1** the virtual machine on the left, **pc2** the virtual machine on the right. More precisely the directories are built for the fact that each file inside that particular folder will be mapped (create or replace ) inside the filesystem of that particular virtual machine.

This file.conf called in our case **lab.conf** should be defined as follow:

**1 pc1[0]=A**

**2 r1[0]=A**

**3**

**4 pc2[0]=B**

**5 r1[1]=B**

-it's a comment- **pc1 0** interface connected to domain A

Here we can see that 0 or 1 represents the port, infact router have two ethernet port 0 and 1. A and B represent the match for link the two virtual machines.

Now with the command of **lstart** we're able to generate the virtual machines. In same bash we should to press **lcrash** for stopping the laboratory. After it we should to add the IP address for each port. How we've defined we have 1 port for the two virtual machines and 2 for the router.

Now that the ifaces are up, we are can assign IPv4 addresses to them. The command to assing IPv4 addresses to ifaces is ip address: the command is the following:

**ip addr add "ip addr" dev eth0**

In case we want to remove a ip addr we should replace the word "add" with "**del**"

Now we can continue analyzing the example:

**pc1's bash: ip addr add 10.0.0.100/24 dev eth0**

**pc2's bash: ip addr add 10.0.1.100/24 dev eth0**

**r1's bash: ip addr add 10.0.1.100/24 dev eth0 and**



**ip addr add 10.0.1.100/24 dev eth1** (To check this is correct I can check with "ip a")

After this we should activate how we define in the file.conf the link of pc1 that is eth0, the link of pc2 that is eth0, the links of r1 that is eth0, eth1. Rules are:

**ip link set eth0 up**, in case we've eth1 we should replace in the previous routine. (now with "ip l" you can check there's written IP)

Ora cosa accade nella realtà io attualmente posso mandare pacchetti però solo tra porte all'interno della mia subnet con il comando **ping "ip addr"** con ip addr che rappresenta il **source** cioè a chi mando.

Ora per mandare il pacchetto a una virtual machine che è presente in un'altra subnet devo abilitare la funzionalità legata al router. Perché? Because of the Address Resolution Protocol (ARP) -> he needs to learn the MAC addresses of the other devices... Abbiamo la necessità di gestire ip route. Esistono 2 modi per farlo: 1) define the default route (gateway) and it means that we tell to the host "give all the packets that are not for our subnet to the gateway, he knows how to route them. (gateway is r1)

2) specify how to reach pc2 subnet from pc1 and viceversa. In this approach we tell to the host "give all the packets that are for a specific subnet to a specific host, he knows how to route them"

**default ip route add default via ip-addr-port-router**

**No-default ip route add ip-addr-subnet via ip-addr-port-router**

vogliamo far comunicare per esempio eth0, porta di r1 legata alla subnet di pc1, con la subnet di pc2.

Nel nostro caso per ritornare all'esercizio devono essere scritti i seguenti comandi

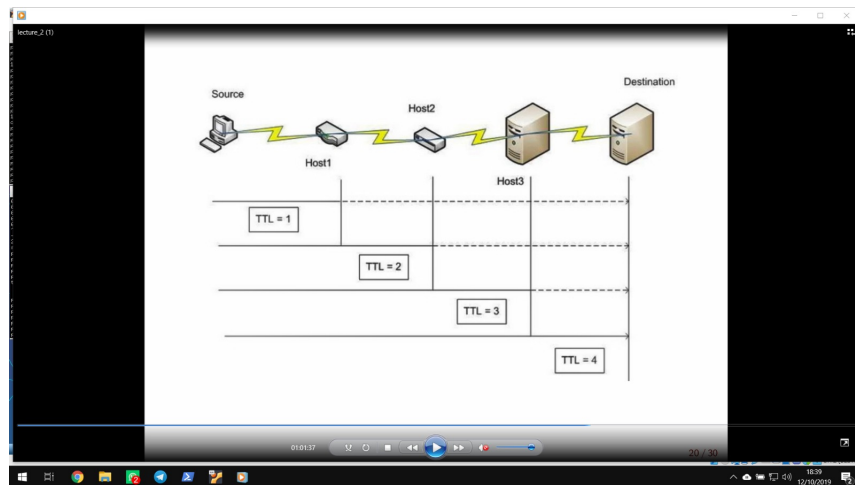
**pc2 ip route add 10.0.0.0/24 via 10.0.1.1**

**pc1 ip route add 10.0.1.0/24 via 10.0.0.1**

Per avere informazioni sulla rete esistono i seguenti comandi

- **ip a**
- **ip r** After I assign an IP address the routing table is populated through the following command is shown the routing table of the VM

- **ip l**
- **tracert ip** ci permette di far capire la distanza che c'è tra pc1 e pc2 per esempio se dal bash di pc2 chiamiamo `tracert 10.0.0.100` esce una mappa. A cosa serve?



Se chiamiamo **ping 10.0.0.100 -t 1** la comunicazione fallirà nel senso che prova ad inviare messaggi in maniera infinita infatti avremo "Time to live exceeded" se ad 1 sostituiamo 2 invece la stabiliremo la connessione e lo scambio di pacchetti

Set up interface:

Legacy utility	Obsoleted by	Note
<code>ifconfig</code>	<code>ip addr</code> , <code>ip link</code> , <code>ip -s</code>	Address and link configuration
<code>route</code>	<code>ip route</code>	Routing tables
<code>arp</code>	<code>ip neigh</code>	Neighbors
<code>iptunnel</code>	<code>ip tunnel</code>	Tunnels
<code>nameif</code>	<code>ifrename</code> , <code>ip link set name</code>	Rename network interfaces
<code>ipmaddr</code>	<code>ip maddr</code>	Multicast
<code>netstat</code>	<code>ip -s</code> , <code>ss</code> , <code>ip route</code>	Show various networking statistics

Il comando successivo che viene fatto è il seguente: mentre pc2 spedisce pacchetti attraverso il ping. pc1 si mette in ascolto attraverso il seguente comando

**tcpdump -n -i eth0** la cosa importante è la porta di ascolto.

Nel caso in cui io volessi salvarli su di un file tali pacchetti devo usare il seguente comando

**tcpdump -n -i eth0 -w icmp.pcap.**

**-i** specifies the interface where I want to catch packets,

**-n** tells to not try to convert the IP address I found (do the work of the DNS, find the human name for the IP address)

.Qui si vede l'utilità di creare per ogni pc una directory. Aprendo poi tale file con wireshark possiamo studiare i pacchetti.

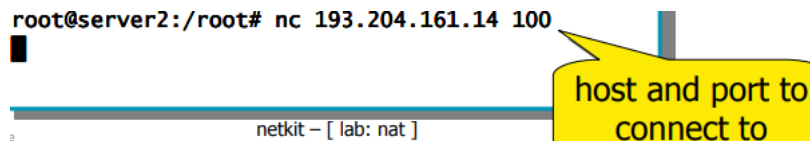
Ora siamo interessati a conoscere un modo per creare una conversazione immediata, per far cio dobbiamo usare il seguente comando in entrambe le pc(1 e 2). In detail we've:

- netcat (nc): is a simple utility to transfer stdin/stdout over a TCP connection
- start nc in server mode on host1



A terminal window on host1 shows the command `nc -l -p 100` being entered. A yellow callout bubble points to the `-l` flag with the text "listening mode". Another yellow callout bubble points to the `100` with the text "port to listen on".

- from server2, use nc to connect to the server running on host1



A terminal window on server2 shows the command `nc 193.204.161.14 100` being entered. A yellow callout bubble points to the IP address and port with the text "host and port to connect to". Below the terminal window, the text "netkit - [ lab: nat ]" is visible.

So after defining it in

**pc1** `nc -l -p 10000`

**pc2** `nc 10.0.0.100 10000`

After this the two virtual machines can start speaking each others like live messangers. Per vedere cosa gli arriva ad uno dei due dobbiamo prima farla partire, ipotizziamo facciamo partire pc1.. poi stoppiamo il processo con (Ctrl+Z) e successivamente utilizziamo il comando **bg** che ci permette di poter concatenare i 2 comandi ovvero quello di attivare la conversazione e quello di poter ascoltare cosa ci arriva iin versione di pacchetti per mezzo del seguente comando

**tcpdump -ni eth0 tcp** ricordo che tal comando è testato in pc1. Nel caso volessimo anche salvarlo nel file

**tcpdump -ni eth0 tcp -w tcp.pcap** Per verificare basta scrivere qualcosa da pc2. Ricordo che pc2 deve essere attivato solo dopo aver settato entrambi i comandi di pc1. Possiamo verificare che sono pacchetti tcp per mezzo del SYN nei pacchetti. Ora proviamo ad eseguire lo stesso procedimento basandoci su un protocollo diverso udp. Ora prima eseguiamo il comando di ascolto ovvero il seguente (sempre in pc1)

**tcpdump -ni eth0 udp -w udp.pcap** poi ancora

**bg** dopo aver stoppato il processo con shortcut precedentemente fatto vedere e facciamo partire la connessione udp

**nc -u -l -p 10000**. Il comando invece in pc2 rimane lo stesso con l'aggiunta del flag -u quindi **nc -u -l -p 10000**. Ritornando su pc1 ed interrompendo per poi riprendere il processo con i tasti Ctrl+Z e **fg** possiamo notare che ci dice quanti pacchetti sono stati catturati.

## 1.4 IP TUNNELLING

| IP HEADER 1 | IP HEADER 2 | It's the case when I've two IP addresses in the same packet.

The main way to do VPN is with IP tunneling .

If the inner IP header uses TCP, what protocol I'll use for address 1? It does not make sense to use TCP, because the reliability is guaranteed by the inner address so I don't need TCP on the external one and we use always UDP on the external one. TCP uses session control -> I have huge performance degradation if I put all this overhead stuff in the tunnel)

# 2

---

## 2 LECTURE

### 2.1 HOW TO SOLVE PROBLEM OF STORING INFORMATION

Netkit allows us to execute commands at startup via specific files. Taking the example of lab-0, we can configure the ifaces on startup creating the 3 startup files for pc1, pc2 and r1 named "VM NAME.startup". (So pc1.startup-pc2.startup-r1.startup) So, for each VM, we create a startup file which: 1)bring up the ifaces,2)assign an ip address to iface,3)populated the rou-

ting table(if needed).

Now we should put in the `pc1.startup`:

- `ip link set eth0 up`
- `ip addr add 10.0.0.100/24 dev eth0`
- `ip route add default via 10.0.0.1`
- 
- Ora devo fare lo stesso per gli altri host e un metodo veloce per riscrivere potrebbe essere il seguente **`cat pc1.startup > pc2.startup`**
- In `pc2.startup` devo modificare il file in modo da ottenere questo contenuto al suo interno:
- 
- `ip link set eth0 up`
- `ip addr add 10.0.1.100/24 dev eth0`
- `ip route add default via 10.0.1.1`
- 
- Ora gestisco il file `r1.startup`(posso riusare i comandi precedentemente analizzati)
- 
- `ip link set eth0 up`
- `ip link set eth1 up`
- `ip addr add 10.0.0.100/24 dev eth0`
- `ip addr add 10.0.1.100/24 dev eth1`

Ricordiamo che questi file devono essere modificati in quanto mi permettono di montare all'avvio le infrastrutture.

Ora posso tranquillamente ripartire dal bash e far partire il sistema con `-lstart`:(Notiamo che possiamo eseguirlo in parallelo con `-p0` ma per farlo dobbiamo installare `make`). Una volta avviato il tutto dobbiamo notare la seguente scritta in ogni bash che si è generata dopo l'avvio:

**r1 login:root(automatic login)** Cio deve esserci per ogni infrastruttura quindi r1,pc1 e pc2. Un'ulteriore verifica la possiamo fare dall'indirizzo ip settato. Andiamo su pc1 e controlliamo l'indirizzo..**ip addr**.

Una volta verificato proviamo a mandare il pacchetto con **:ping 10.0.1.100**.

## 2.2 2 FASE

Dopo aver eseguito questi passaggi e aver testato il tutto possiamo notare che i file sono virtualmente visibili. Adesso dobbiamo sapere che in ambiente linux l'ifaces configuration viene fatta attraverso la creazione del seguente file di configurazione con all'interno un particolare script che scriveremo sotto e con un particolare path che è il seguente: **"/etc/network/interfaces"** dove interfaces è il file. In tale file viene specificato e costruita la ifaces, assegnato l'ip la netmask e il gateway. In tale file dobbiamo inserire il seguente contenuto:

```
auto eth0
iface eth0 inet static
    address 10.0.0.100
    netmask 255.255.255.0
    gateway 10.0.0.1
```

Ricapitolando le istruzioni da eseguire sono le seguenti:

- da terminale
- `mkdir -p pc1/etc/network`
- `mkdir -p pc2/etc/network`
- `mkdir -p r1/etc/network`
- 
- Ricordo che -p, -parents no error if existing, make parent directories as needed
- ora costruisco il file con "touch "
- 
- `touch pc1/etc/network/interface`
- `touch pc2/etc/network/interface`

- touch r1/etc/network/interface
- 
- A seconda della directory in cui mi trovo dovrò settare in maniera diversa (Scrivo prima pc1 e poi in pc2):
- In pc1 avremo:
  - auto eth0
  - ifaces eth0 inet **static**
  - address 10.0.0.100
  - network 255.255.255.0
  - gateway 10.0.0.1
- In pc2 avremo:
  - auto eth0
  - ifaces eth0 inet **static**
  - address 10.0.1.100
  - network 255.255.255.0
  - gateway 10.0.1.1
- Nel router (r1) avremo:
  - auto eth0
  - ifaces eth0 inet static
  - address 10.0.0.1
  - network 255.255.255.0
  - auto eth1
  - ifaces eth1 inet static

- address 10.0.1.1
- network 255.255.255.0
- 

Adesso che siamo arrivati a questo punto dobbiamo ricordare che linux come anche il network è gestito da dei processi che lavorano in background che prendono il nome di **demo-ni** questo è configurato attraverso un file di configurazione e come nel nostro caso avremo `"/etc/network/interfaces/".`

Il processo boot di una VM di netkit prima lancia il demone e dopo monta il file e le cartelle presente in esso e nel lab per cui avremo la necessità di eseguire il **restart** su proprio tali file di configurazione. Per semplificarci il lavoro dobbiamo eseguire tale script nel file startup, cio ci permette di rendere il tutto automatico.

Per cui nei vari file.startup precedentemente analizzati devono essere svuotati per cui esegui il comando `cat "">pc1.startup.`

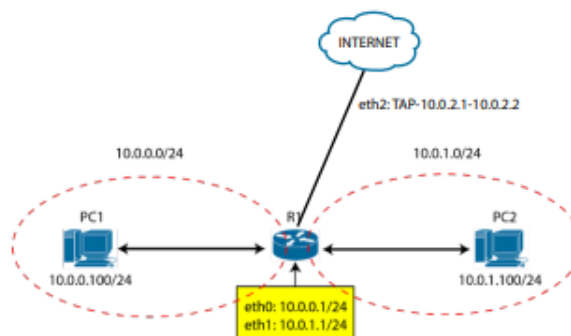
Adesso in ognuno di questi (pc1-pc2 e r1) inseriamo il seguente comando:

- `/etc/init.d/networking restart`

Adesso possiamo testare attraverso prima il controllo di `"ip a"` e poi `"ip r"`. Successivamente proviamo a mandare il pacchetto.. in pc1 bash inseriamo il seguente comando:  
`ping 10.0.1.100.`

### 2.2.1 TAP

Netkit permette di creare uno strato con 2 interfacce che prende il nome di TAP e si andrà a collocare tra un VM e l'host machine. Con tale interfaccia(TAP) possiamo mandare l'intera rete virtuale su internet. Prendendo in analisi la seguente topologia:





Per creare una interfaccia TAP dobbiamo inserire nel **lab.conf** la seguente riga di codice:

**r1[2]=tap,10.0.2.1,10.0.2.2** Questo comando ci permette di creare un eth2 interfacca di r1 con indirizzo 10.0.2.2 e creare un nk-tap-user interfaccia sull'host con indirizzo 10.0.2.1. Ora proviamo a eseguire il ping per accedere a google.. ping www.google.com ma non otteniamo alcuna risposta in quanto non abbiamo settato il DNS. Packets coming from pc1 and pc2 are correctly forwarded to the host machine, which forwards them to the outside world (use wireshark on your host machine). But when they come back, the host machine does not have any entry on it's routing table to forward the packets to pc1 and pc2 (use ip route on the host): Two solutions:

- Add manually the route for pc1 and pc2 on the host (boring)
- Every packet coming from pc1 and pc2 passes trough r1 tap interface, but the host machine knows the address of the tap interface!

r1 should change the source ip address of the packets coming from pc1 and pc2 before forwarding them to the host machine. Now, the host machine knows how to forward back them, because to her is if its like they are directed to r1. Once they are on r1, he changes back the correct destination ip addresses and finally pc1 and pc2 take the reply.

**Piccola parentesi:** per il motivo che non funziona adesso dovremmo entrare sul bash di r1 e runnare il seguente codice

**vim /etc/resolv.conf** in questo dovremmo aggiungere il seguente codice **nameserver 8.8.8.8** dopo aver salvato possiamo provare ad eseguire un ping da pc1 bash con il seguente comando "ping 8.8.8."

. In order to accomplish the task,  
in r1 dobbiamo runnare il seguente comando:

```
iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
```

Sempre nello stesso bash di r1:

- tcpdump -ni eth0 icmp
- tcpdump -ni eth2 icmp

Ora dobbiamo aggiungere in `r1.startup` sotto al comando precedentemente scritto il seguente codice:

```
r1 iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE.
```

Adesso posso riprovare a fare il ping da `pc1` bash(`ping 8.8.8.8`).

## 2.3 DHCP

Ora possiamo provare a fare DHCP in cui ricordiamo che abbiamo dei DHCP client che rappresentano VM come `pc1` e `pc2`. Poi ci saranno dei server che sono i router in questo caso. Come si configura? Riandiamo sul file `interfaces` di `pc1` e `pc2` e lo dobbiamo settare modificando la parola `static` in `dhcp`. Quindi al posto della prima riga dobbiamo mettere la seguente riga:

```
iface eth0 inet dhcp
```

Poi dobbiamo configurare il DHCP server che nel nostro caso è `r1`. Come? Creiamo un file **`dhcpd.conf`** in `/etc/dhcp3` con il seguente contenuto:

```
default-lease-time 3600;
option domain-name-servers 8.8.8.8;

subnet 10.0.0.0 netmask 255.255.255.0 {
    range 10.0.0.100 10.0.0.254;

    option routers 10.0.0.1;
}

subnet 10.0.1.0 netmask 255.255.255.0 {
    range 10.0.1.100 10.0.1.254;

    option routers 10.0.1.1;
}
```

Per concludere devo inserire in `r1.start` il seguente codice che mi permette di far avviare il server DHCP:

**`/etc/init.d/dhcp3-server start`**. if you manage to set up the DHCP correctly, disregard the IP addresses of the subnets, they will be assigned by the DHCP server.

# 3

---

## 3LAB

Il problema principale dei routing statici è relativo alla limitazione nei seguenti campi:

**scalability issues** quando il network cresce in complessità, configurando ogni singolo router per ogni singolo percorso inizia a diventare complicato e se vogliamo tener conto dei costi dei percorsi delle volte abbiamo delle ridondanze.

**reliability issues** se un link va giù l'approccio statico non può trovare un'alternativa ridondante.

**Heterogeneity issues** il network può essere composto da router costruiti con OS diversi e gestiti da un admin che deve essere in grado di amministrarli tutti.

I protocolli più usati sono (RIP,OSPF,IS-IS,IGRP).

### 3.1 OPENS SHORTEST PATH FIRST

It's the most spread routing protocol and one of the fastest. It belongs to the category of the Interior Gateway Protocols, in details in Link State Protocols, means that every node in the network must know the entire network topology. Every router running OSPF advertises its presence to neighbors routers and "floods" the network with particular messages called Link State Packets which contains information about the link. Every router constructs the network topology from LSP with the Dijkstra algorithm and builds the best path for every single host. In Linux OSPF is included into a suite called **quagga**. The daemon which implements OSPF is called **zebra**. For having up OSP and running we must first configure the zebra. This kind of protocols takes care about the shortest path, not only related on costs but also related on the path used. How to do that in practice?

We should access in `/etc/quagga/zebra.conf` and the file return is something like this:

```

! *- zebra *-
!
! zebra sample configuration file
!
! $Id: zebra.conf.sample,v 1.1.1.1 2002/12/13 20:15:30 paul Exp $
!
hostname r1
password zebra
enable password zebra

interface eth0
ip address 1.0.10.2/31
link-detect

interface eth1
ip address 1.0.10.9/31
link-detect

interface eth2
ip address 1.0.10.13/31
link-detect

interface eth3
ip address 1.0.10.19/31
link-detect

```

At the top there're different comments, then there're the description of the name of the host with the password of the daemon. That should be zebra. Now we should put the interfaces with a particular eth and the respect ip address of the interfaces. Instead **link-detect** defines if the link is up or down —————

Now we can see the **/etc/quagga/ospfd.conf** it's composed by interfaces that you want to broadcast packet.

For configuring skeleton-lab (file on the web site).. we can see that there's no way for ping from r1 the pc1 because it's got only information on its interfaces. First thing to do for solving it is to remove the word "no" in front of the word "yes" in the ospfd assignment into the file **/etc/quagga/daemons**.

Now we should to edit the file **/etc/quagga/zebra.conf**. Inside of it we should change the router with the id of the router for example r1.

- hostname r1
- password zebra

Now we should configure the ip address:

- interface eth0
- ip address 10.0.10.2/31
- link-detect

and the same with a different interfaces and ip address. (Per vedere a quale ip assegnarle- clicca ip e vedi il flag inet e a quale interfaccia fa riferimento.)

Now for finishing this process we should configure `/etc/quagga/ospfd.conf`. In it we should remove all informations but not hostname and password rows.

Now we should configure

- hostname **r1**
- password zebra
- 
- interface eth0
- ospf hello-interval 2

We're saying, "look for eth0,eth1 eth2,eth3, send the (hello packet) and advertise the fact that you are ospf router. The words(hello-interval says how many seconds is the frame-gap from one to the other)

The router ospf should be also configure, so in the same file we should configure which is the router, ospf router(in our case r1) is able to achieve different subnet. We should set-up this feature.How?

- router ospf
- network 1.0.10.2/31 area 0.0.0.0
- network 1.0.10.8/31 area 0.0.0.0
- network 1.0.10.12/31 area 0.0.0.0
- network 1.0.10.18/31 area 0.0.0.0

Per far uscire questi numeri, dobbiamo rivedere su ip a e vedere gli inet delle varie interfacce. Levare a queste -1. Questo è il contenuto:

```
hostname r1
password zebra

interface eth0
ospf hello-interval 2

interface eth1
ospf hello-interval 2

interface eth2
ospf hello-interval 2

interface eth3
ospf hello-interval 2

router ospf
network 1.0.10.2/31 area 0.0.0.0
network 1.0.10.8/31 area 0.0.0.0
network 1.0.10.12/31 area 0.0.0.0
network 1.0.10.18/31 area 0.0.0.0
```

After this we should restart the daemon with the following instruction: **/etc/init.d/quagga restart**. If we run it always in r1 we've a problem on permission denied. It's a particular problem on this Netkit distribution: we can solve it with the following commands:

- `ls -al /etc/quagga/ospfd.conf`
- `chmod +r+w /etc/quagga/ospfd.conf`
- `chmod +w /etc/quagga/ospfd.conf`

Abbiamo cambiato modalità avendo quindi la possibilità di eseguire alcune modalità. Adesso dobbiamo riloggarci il daemon : **/etc/init.d/quagga restart**

How we say before we should set r7. **IMPORTANT: OSPF should not be implemented in pc or servers.** In r7 o r6 non dobbiamo mettere le varie interfaces.

```
hostname r7
password zebra

interface eth0
ospf hello-interval 2

router ospf
network 10.0.3.0/24 area 0.0.0.0
network 1.0.10.18/31 area 0.0.0.0
```

Dopo aver configurato il file zebra.conf e il file ospf.conf.

`chmod +r /etc/quagga/ospfd.conf` lanciamo con il restart... e poi `watc ip r`.

Andando ora in r3 dobbiamo disabilitare il link con che rischia il collegamento con r4 quindi in shell r3.. `ip link set eth1 down..` ma andando su r4 il collegamento rimane perchè è settato in maniera dinamica quindi raggiungo r3 attraverso r2 e poi r3. Cio può essere testato attraverso il comando: `ping 10.0.1.100` che è un modo per raggiungere pc1. Un altro importante concetto è basato sul concetto di aree. Facendo riferimento sempre alla stessa topologia se vogliamo per esempio mandare dei pacchetti in pc4 non siamo interessati tanto a conoscere pc4 ma invece siamo interessati ad instradarlo fino a r7, sarà poi quest'ultimo essendo un componente intelligente che sarà in grado di passargli le informazioni. r7 è definito il "Border" dell'area compresa fra 0.0.0.0 e 1.1.1.1

Vedendo quindi tale esempio analizziamo il file ospfd.conf e in questo andiamo a modificare il network area che da r7 va verso r1 in quanto tale area non sarà più 0.0.0.0 ma 1.1.1.1

Stessa cosa deve essere applicata allo stesso indirizzo ip ma da parte di r1 verso r7. Stessa cosa tra r6 e r2 con la differenza che da 0.0.0.0 avremo 2.2.2.2. Allo stesso modo tra r3 e r5 da 0.0.0.0 a 3.3.3.3

Attenzione errore commesso anche da Marcolino Spaziani, se il pacchetto attraversa un'area che ha un particolare nome così verrà definito anche per le altre interfacce. Quindi in r7 non avremo area che punta a pc1 con 0.0.0.0 e l'altra a 1.1.1.1 ma tutte e due a 1.1.1.1 in maniera analitica anche negli altri router border.

# 4

---

## 4LAB

————— Security goals is to provide : 1) Confidentiality, 2) Integrity, if I have a message none can modify the message, 3) authentication, if you want to talk with an host you should log in in and prove that the message is for you. How can we achieve these three requirements? Confidentiality- is guarantee according **encryption** which could be symmetric or asymmetric. The pattern used for the encryption is having two functions: an Encryption and Decryption functions. One from plaintext to the ciphertext and the other one from ciphertext to plaintext. Message authentication code guarantee the data integrity the most famous is H-MAC. The authentication instead is guarantee with the Asymmetric crypto.

One time pad is the best algorithm for confidentiality

$$E: c = m \oplus k$$

$$D: c \oplus k = m \oplus k \oplus k = m$$

The problem of this approach is the length of the key (  $|k|=|m|$  ) and the key must be chosen at random in the key space.

While we use a xor operator?

Table of Xor:

A	B	Y
0	0	0
0	1	1
1	1	0
1	0	1

Integrity- I need a fingerprint so something that let to be a message unique. So we should use an hash function In our case we 're interest to encrypt data of packet without encrypt the header:

header	payload
MAC	E+MAC

Where Ethen MAC- (IPSEC)

MAC then E-(TLS/SSL)

E and MAC- (SSH)

About the asymmetric E s refer to RSA and DH.

---

Sometimes we need to reconfigure network nodes remotely cause is not always possible to physical access to network nodes, so we should use Telnet as protocol, but it's deprecated nowadays for the fact that it not use encryption when it sends data, for this fact we replace this with a SSH protocol.

## 4.1 SSH

SSH is a protocol that allows remote managing (host) over a secured "pipe".It is a client-server protocol. It's used to configure for example a server which I can't have physically here. The main features that SSH provides are:

- Authentication- want to be sure about the the users how to do that?(Asymmetric encryption or UserName and password).
- Encryption
- Integrity

Little recap of encryption:

in Asymmetric Cryptography, there are 2 main ingredients:

- Public key



- Private Key

The public key is meant to be shared to everyone who want to use an encrypted language with you. The private key is meant to remain secret. The asymmetry is in the fact that the public key is used to encrypt data and the private key to decrypt them.

## 4.2 DESCRIPTION OF SSH

SSH runs over TCP, on the default port 22. For UDP we should reconfigure SSH. When the client starts the session, send a packet to the server specifying the SSH version he (the client) supports. Then, the server answers with the SSH version he supports. After this initial handshake, the two exchange each other the encryption algorithms they support. The first that is common to both is chosen. The client and the server now can move forward and authenticate themselves. **example of asymmetric authentication:** `sudo tcpdump -ni any`

## 4.3 AUTHENTICATION PHASE

The server send his public key to the client, then the client is now asked to trust or not trust the server key.

View the example below;

```
lMac-dl-Marco:~ marcospaziani$ ssh marcos@stud.netgroup.uniroma2.it
The authenticity of host 'stud.netgroup.uniroma2.it (160.80.221.14)' can't be established.
ECDSA key fingerprint is SHA256:UQRu/pfFev8Vnd8v6nAjd0puetHfocxDSXsismY3Q28.
Are you sure you want to continue connecting (yes/no)?
```

If it trusts the ser-

ver identity, the client proceeds with authenticating himself. This step can be done in two ways:

- **Username and password** In this case, the client send its username and password to the server using the chosen encryption algorithm. If the username corresponds to a user on the server and the password is correct, the server grant access to the user and the ssh session can start.
- **Asymmetric Cryptography** can be used for authentication only if the client has a couple of public-private key and if the server has the public key of the client. The server picks up a random string and encrypt that with the public key of the client. This encrypted value is then transferred to the client through the secured pipe (via the common chosen algorithm before). The client decrypts the challenge with his private key and then crypts it again with the public key of the server. Once back, the server decrypts the challenge response: if the response is the same as the random value sent, with a certain

degree of probability the client is the one associated with that public key. At this point, the session can start.

#### 4.4 IN PRACTICE

Each Linux distribution has an ssh client installed called OpenSSH client. On the other hand, the server daemon, sshd, needs to be manually installed. On netkit, they are both already installed so we shouldn't install anything. The configuration file of the server is **/etc/ssh/sshd\_config** while for the client is **/etc/ssh/ssh\_config**.

The difference is the **d**. Now for starting a connection with a server:

```
ssh username@host_ip_addr
#starts ssh session with the host specified by
#host_ip_addr logging in as username
```

Quindi vogliamo creare un utente allo startup, su pc2, specificando un accesso ssh. ricordiamo che ogni utente deve avere una password. Chiamiamo l'utente come ssh\_user  
Suppose we want to connect from pc1 to pc2 via ssh.  
We want to create at startup a user, on pc2, specifically for ssh access.  
Let's call it "ssh\_user".  
In order to be accessed via ssh, every user must have a password. The startup file of pc2 will be:

```
/etc/init.d/networking restart
/etc/init.d/ssh restart
mkdir /home/ssh_user
useradd ssh_user -d /home/ssh_user
chown ssh_user:ssh_user /home/ssh_user
echo -e 'ilovessh\nilovessh\n' | passwd ssh_user
```

Linux ci permette di specificare stringhe per ip address dentro il /etc/hosts file . Qui sotto vediamo cosa ci sta all'interno di pc1

Linux allows us to specify string literals to ip addresses inside the /etc/hosts file.

The syntax is ip\_addr - tab - string.

For example, we assign the 10.0.1.100 to the string pc2 at startup on pc1:

```
/etc/init.d/networking restart  
echo "10.0.1.100      pc2" >> /etc/hosts
```

After that, if we type on pc1:

```
user@localhost:~$ ssh ssh_user@pc2
```

We will be prompted for password for ssh\_user, which is "ilovessh".

Scarica il primo link in Tap sottogruppo: quello che vogliamo fare : aggiungere in pc2  
adduser user

"ora richiede la password"

user

e poi confermiamo Ora in pc2 /etc/init.d/ssh start

In pc1 ora ssh user@10.0.1.100 eseguiamo e poi eseguiamo

Ci troviamo nella shell di pc2user

touch foo

e poi possiamo vedere il file nella shell di pc2 . per eseguire il logout (Ctrl+D oppure exit)

Adesso sempre in pc1 eseguiamo : **ssh-keygen** adesso cd .ssh e eseguendo i vari cat stampiamo informazioni diverse:

esempio cat id\_rsa.pub Ora vogliamo copiare l'informazione in maniera sicura **scp .ssh/id\_rsa.pub user@10.0.1.100** Nel nuovo user dobbiamo eseguire mkdir .ssh, cd .ssh/, poi touch authorized\_keys e poi cat..../id\_rsa.pub>authorized\_keys