# Introduzione al Networking

## Guida introduttiva alla configurazione di rete e al firewalling in ambiente GNU/Linux

Andrea Grazioso
grazioandre@gmail.com

Corsi GNU/Linux avanzati 2015 - Amministrazione di Sistema

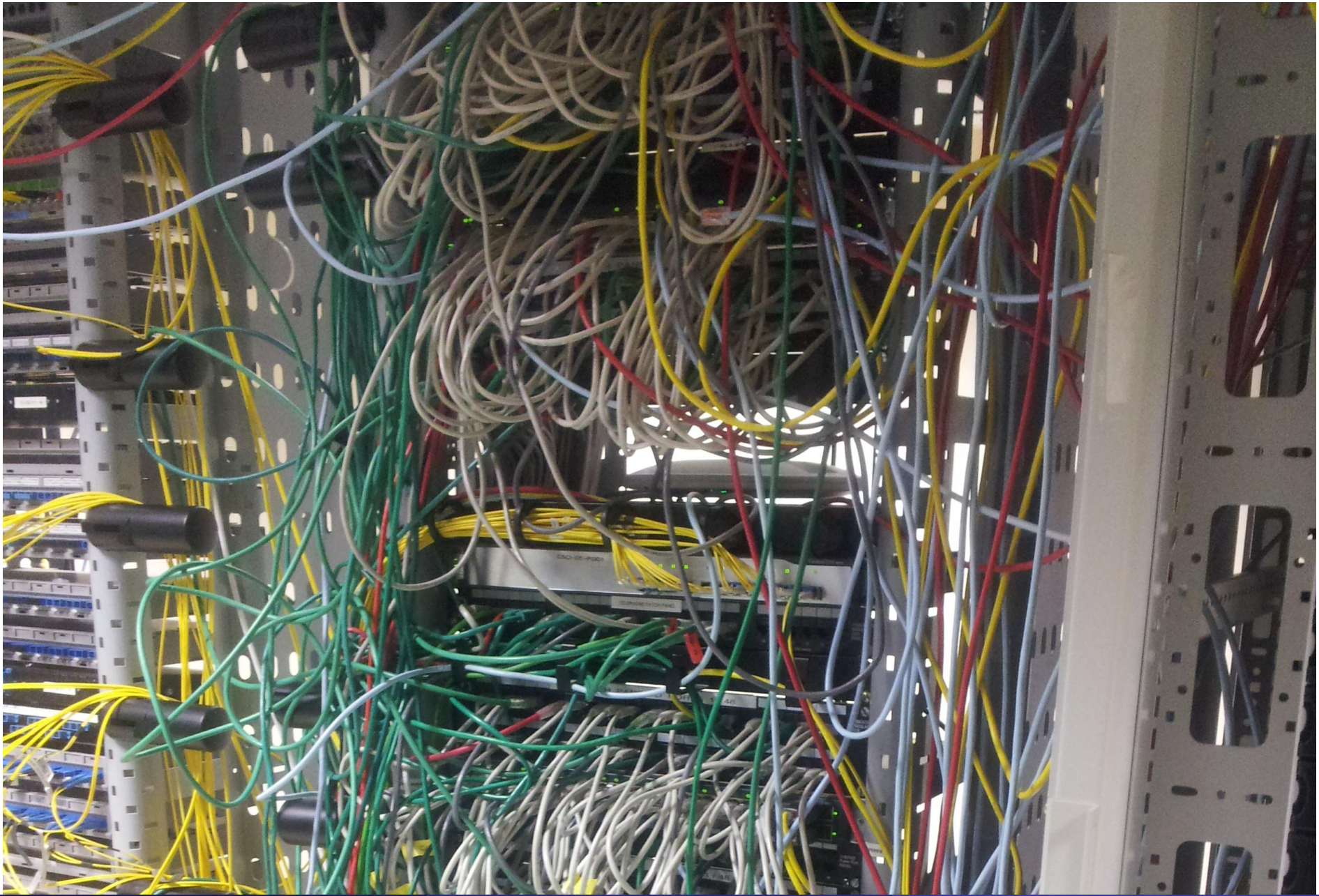
POLITECNICO OPEN
unix LABS
Come hack with us.

# Sezione 1

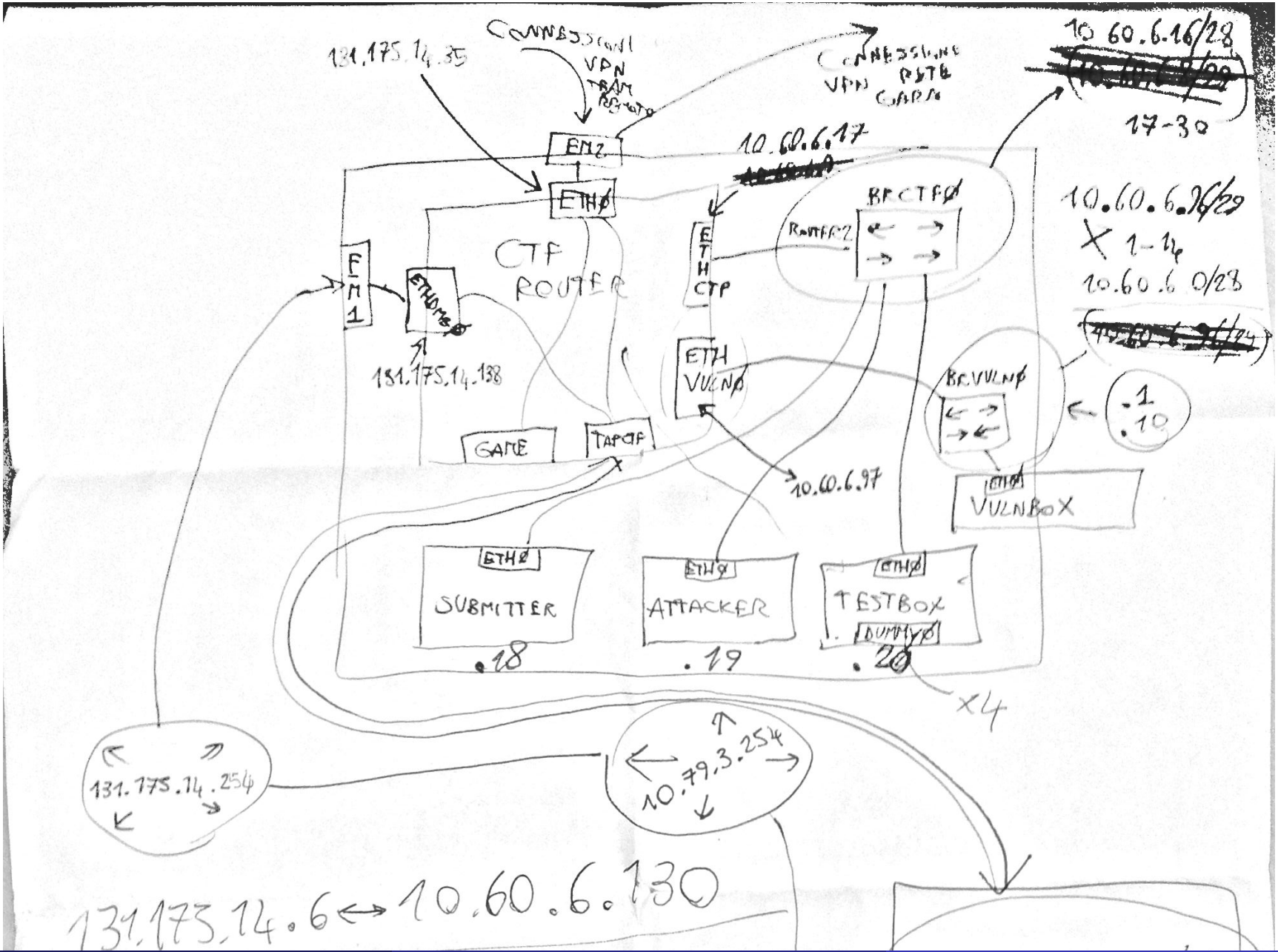
# Networking General

Networking vuol dire costruire, progettare e usare una rete, inclusa la parte fisica (cablaggio, trasmissione del segnale, ecc), la selezione e l'uso di un protocollo telecomunicativo e stack software per utilizzare e gestire la rete, e l'assegnazione di adeguate policy e procedure di utilizzo della rete

# Networking General



# Networking General



# Ambienti di Test

**Netkit** The poor man's system to experiment computer networking[1]

**Mininet** realistic virtual network, running real kernel, switch and application code, on a single machine

[1] Netkit NG

# Ambienti di Test

**Netkit** The poor man's system to experiment computer networking[1]

**Mininet** realistic virtual network, running real kernel, switch and application code, on a single machine

[1] Netkit NG

# Sezione 2

# It Works!

- Maybe it just works?

Se l'host viene connesso ad una rete in cui è presente un router/switch con DHCP configurato è probabile che sia già tutto pronto per accedere ad Internet

# Maybe not...

- Può capitare che le interfacce di rete non siano visualizzate dal sistema operativo “out of the box”
- Partendo dalle basi:
- Per scoprire quali moduli del kernel sono disponibili per il networking possiamo cercare direttamente nel filesystem

```
root@Hello:~# ls /lib/modules/‘uname -r‘/kernel/drivers/net
appletalk  can       ethernet  ieee802154  macvtap.ko     nlmon.ko
ppp        sungem_phy.ko  vmxnet3   wireless arcnet     dsa
fddi       ifb.ko      mdio.ko        ntb_netdev.ko  rionet.ko
team           vxlan.ko  xen-netback bonding    dummy.ko  hamradio
irda        mii.ko         phy            sb1000.ko  usb
wan caif       eql.ko    hyperv    macvlan.ko  netconsole.ko  plip
slip       veth.ko        wimax
```

- Se sappiamo quale driver necessita la nostra scheda di rete possiamo usare “modprobe” per caricare il modulo del kernel associato.

# Maybe not...

- Può capitare che le interfacce di rete non siano visualizzate dal sistema operativo “out of the box”
- Partendo dalle basi:
- Per scoprire quali moduli del kernel sono disponibili per il networking possiamo cercare direttamente nel filesystem

```
root@Hello:~# ls /lib/modules/‘uname -r‘/kernel/drivers/net
appletalk  can       ethernet  ieee802154  macvtap.ko     nlmon.ko
ppp        sungem_phy.ko  vmxnet3   wireless arcnet     dsa
fddi       ifb.ko      mdio.ko        ntb_netdev.ko  rionet.ko
team           vxlan.ko  xen-netback bonding    dummy.ko  hamradio
irda        mii.ko         phy             sb1000.ko  usb
wan caif       eql.ko    hyperv    macvlan.ko  netconsole.ko  plip
slip       veth.ko        wimax
```

- Se sappiamo quale driver necessita la nostra scheda di rete possiamo usare “modprobe” per caricare il modulo del kernel associato.

# It Works! Again

Con i giusti moduli caricati possiamo verificare quali interfacce di rete sono rilevate nel sistema semplicemente con **ls**:

```
root@Hello:~# ls /sys/class/net/
eth0  eth1  lo
```

# mii-tool e autonegoziazione

Occasionalmente le schede Ethernet hanno delle opzioni configurabili riguardo le loro capacità Ethernet e possono essere interrogate a riguardo:

- Se il cavo Ethernet è collegato alla scheda e se arriva segnale sul cavo
- Se la scheda supporta 10/100/1000 Mb/s in modalità half-duplex o full-duplex
- Quali sono le capacità dello switch a cui è collegata la nostra scheda
- Se l'***autonegoziazione***[2] ha funzionato o no[3]

[2] Subito dopo aver collegato il cavo alla scheda e aver ricevuto il link, la scheda tenta di inviare una serie di frame di varie dimensioni per capire quali sono le velocità supportate dal partner

[3] Problemi di negoziazione possono causare **gravi** problemi di performance della scheda di rete

# mii-tool e autonegoziazione

Occasionalmente le schede Ethernet hanno delle opzioni configurabili riguardo le loro capacità Ethernet e possono essere interrogate a riguardo:

- Se il cavo Ethernet è collegato alla scheda e se arriva segnale sul cavo
- Se la scheda supporta 10/100/1000 Mb/s in modalità half-duplex o full-duplex
- Quali sono le capacità dello switch a cui è collegata la nostra scheda
- Se l'***autonegoziazione***[2] ha funzionato o no[3]

[2] Subito dopo aver collegato il cavo alla scheda e aver ricevuto il link, la scheda tenta di inviare una serie di frame di varie dimensioni per capire quali sono le velocità supportate dal partner

[3] Problemi di negoziazione possono causare **gravi** problemi di performance della scheda di rete

# mii-tool e autonegoziazione

Occasionalmente le schede Ethernet hanno delle opzioni configurabili riguardo le loro capacità Ethernet e possono essere interrogate a riguardo:

- Se il cavo Ethernet è collegato alla scheda e se arriva segnale sul cavo
- Se la scheda supporta 10/100/1000 Mb/s in modalità half-duplex o full-duplex
- Quali sono le capacità dello switch a cui è collegata la nostra scheda
- Se l'***autonegoziazione***[2] ha funzionato o no[3]

[2] Subito dopo aver collegato il cavo alla scheda e aver ricevuto il link, la scheda tenta di inviare una serie di frame di varie dimensioni per capire quali sono le velocità supportate dal partner

[3] Problemi di negoziazione possono causare **gravi** problemi di performance della scheda di rete

# mii-tool e autonegoziazione

Occasionalmente le schede Ethernet hanno delle opzioni configurabili riguardo le loro capacità Ethernet e possono essere interrogate a riguardo:

- Se il cavo Ethernet è collegato alla scheda e se arriva segnale sul cavo
- Se la scheda supporta 10/100/1000 Mb/s in modalità half-duplex o full-duplex
- Quali sono le capacità dello switch a cui è collegata la nostra scheda
- Se l'***autonegoziazione***[2] ha funzionato o no[3]

[2] Subito dopo aver collegato il cavo alla scheda e aver ricevuto il link, la scheda tenta di inviare una serie di frame di varie dimensioni per capire quali sono le velocità supportate dal partner

[3] Problemi di negoziazione possono causare **gravi** problemi di performance della scheda di rete

# mii-tool e autonegoziazione

Occasionalmente le schede Ethernet hanno delle opzioni configurabili riguardo le loro capacità Ethernet e possono essere interrogate a riguardo:

- Se il cavo Ethernet è collegato alla scheda e se arriva segnale sul cavo
- Se la scheda supporta 10/100/1000 Mb/s in modalità half-duplex o full-duplex
- Quali sono le capacità dello switch a cui è collegata la nostra scheda
- Se l'***autonegoziazione***[2] ha funzionato o no[3]

[2] Subito dopo aver collegato il cavo alla scheda e aver ricevuto il link, la scheda tenta di inviare una serie di frame di varie dimensioni per capire quali sono le velocità supportate dal partner

[3] Problemi di negoziazione possono causare **gravi** problemi di performance della scheda di rete

# mii-tool e autonegoziazione

- Se la scheda è dotata di mii transreceiver è possibile utilizzare "mii-tool" per verificare le capacità dell'interfaccia

```
mii-tool -v eth0
eth0: negotiated 1000baseTx-FD flow-control, link ok
  product info: vendor 00:08:18, model 54 rev 6
  basic mode:autonegotiation enabled
  basic status: autonegotiation complete, link ok
  capabilities: 1000baseT-FD 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  advertising:  1000baseT-FD 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-control
  link partner: 1000baseT-FD 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
```

- In caso di problemi di autoconfigurazione si può forzare una scheda ad usare una determinata modalità, ricordandoci ovviamente di fare lo stesso anche dall'altro lato del cavo!

```
mii-tool -force=100BaseTx-FD eth0
```

# iproute2

- Le interfacce di rete possono anche essere “spente” via **software**
- Il nuovo tool **ip** sostituisce gran parte dei vecchi tool per operare sulle schede di rete (ifconfig, route, netstat...)
- Comandi più semplici

```
ip [options] object command
```

- Feature di rete più recenti meglio implementate in un unico tool (tunnel)

# iproute2: informazioni

- Vedere le **informazioni** a livello datalink delle interfacce, come ad esempio il MAC address e se risulta “up”

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
     qlen 1000 link/ether 00:de:ad:be:ef:ca brd ff:ff:ff:ff:ff:ff
```

# iproute2: spegnere ed accendere le luci

- Possiamo cambiare lo stato di un'interfaccia e **attivarla** o disattivarla

```
$ ip link set dev eth0 up
$ ip link set dev eth1 down
```

# iproute2: MAC address

- Possiamo cambiare il **MAC** address di un'interfaccia

```
$ ip link set dev eth0 address XX:XX:XX:XX:XX:XX
```

- Il MAC address è **scritto indelebilmente** all'interno di ciascuna scheda di rete per identificarla in modo univoco, tuttavia il sistema operativo è in grado di "chiudere un occhio"

# iproute2: MAC address

- Possiamo cambiare il **MAC** address di un'interfaccia

```
$ ip link set dev eth0 address XX:XX:XX:XX:XX:XX
```

- Il MAC address è **scritto indelebilmente** all'interno di ciascuna scheda di rete per identificarla in modo univoco, tuttavia il sistema operativo è in grado di "chiudere un occhio"

# iproute2: ARP

- Possiamo abilitare o disabiliare ARP se dovesse servire (e se sappiamo quello che stiamo facendo!)

```
$ ip link set dev eth0 arp off
$ ip link set dev eth0 arp on
```

- Possiamo visualizzare la **tabella ARP**

```
$ ip neigh show
10.99.0.254 dev eth0 lladdr 00:11:22:33:44:55 REACHABLE
```

# iproute2: ARP

- Possiamo abilitare o disabiliare ARP se dovesse servire (e se sappiamo quello che stiamo facendo!)

```
$ ip link set dev eth0 arp off
$ ip link set dev eth0 arp on
```

- Possiamo visualizzare la **tabella ARP**

```
$ ip neigh show
10.99.0.254 dev eth0 lladdr 00:11:22:33:44:55 REACHABLE
```

# ip: Indirizzi IP

- Una volta che abbiamo un'interfaccia funzionante, possiamo **assegnarle uno o più indirizzi IP**

```
$ ip address add 192.168.0.2/24 dev eth0
$ ip address add 1234:40ac:1:8d6b::1/64 dev eth0
```

- Il formato utilizzato per indicare gli indirizzi IP è **VLSM** (Variable-length subnet masking). Indichiamo quindi sempre la lunghezza della netmask [4]

- Rimuovere indirizzi

```
$ ip address del 192.168.0.2/24 dev eth0
$ ip address flush dev eth0
```

[4] L'indirizzamento a classi IP (Classe A,B,C....) è deprecato e non viene più utilizzato dal 1993.

# ip: Indirizzi IP

- Una volta che abbiamo un'interfaccia funzionante, possiamo **assegnarle uno o più indirizzi IP**

```
$ ip address add 192.168.0.2/24 dev eth0
$ ip address add 1234:40ac:1:8d6b::1/64 dev eth0
```

- Il formato utilizzato per indicare gli indirizzi IP è **VLSM** (Variable-length subnet masking). Indichiamo quindi sempre la lunghezza della netmask [4]

- Rimuovere indirizzi

```
$ ip address del 192.168.0.2/24 dev eth0
$ ip address flush dev eth0
```

[4] L'indirizzamento a classi IP (Classe A,B,C....) è deprecato e non viene più utilizzato dal 1993.

# ip: Indirizzi IP

- Una volta che abbiamo un'interfaccia funzionante, possiamo **assegnarle uno o più indirizzi IP**

```
$ ip address add 192.168.0.2/24 dev eth0
$ ip address add 1234:40ac:1:8d6b::1/64 dev eth0
```

- Il formato utilizzato per indicare gli indirizzi IP è **VLSM** (Variable-length subnet masking). Indichiamo quindi sempre la lunghezza della netmask [4]

- Rimuovere indirizzi

```
$ ip address del 192.168.0.2/24 dev eth0
$ ip address flush dev eth0
```

[4] L'indirizzamento a classi IP (Classe A,B,C....) è deprecato e non viene più utilizzato dal 1993.

# IP e Netmask

```
IP Address (numbers):   192.168.0.2
IP Address (bits):      11000000 10101000 00000000 00000010
                        -------- -------- -------- --------
                           192      168       0        2
```

Figure : IpAddress

- L'Indirizzo IP è unico per un host che deve essere accessibile tramite Internet. Per far si che sia distinguibile tra host dentro e fuori la rete, l'indirizzo IP si divide in due parti
  - Rete
  - Host

# IP e Netmask

- La separazione è scritta con la netmask, che permette di identificare quale sia l'una e quale l'altra parte

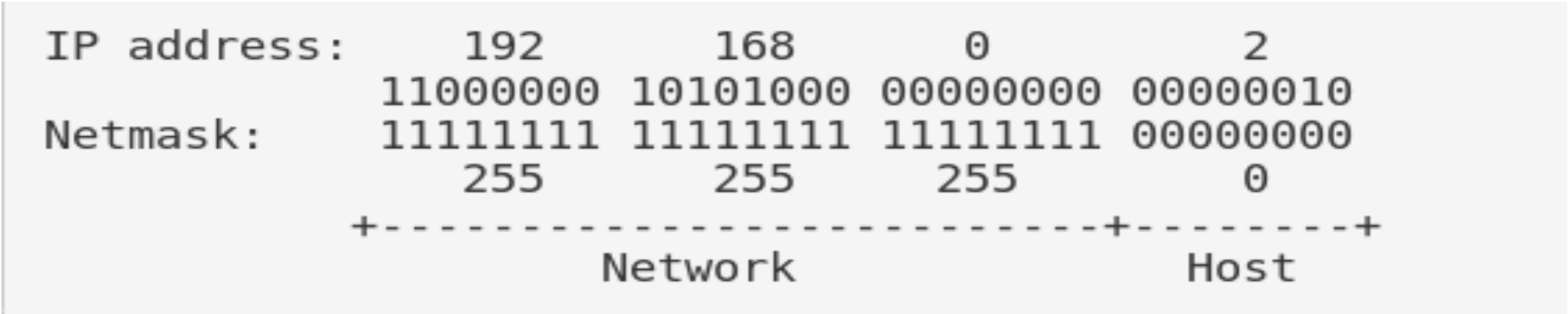
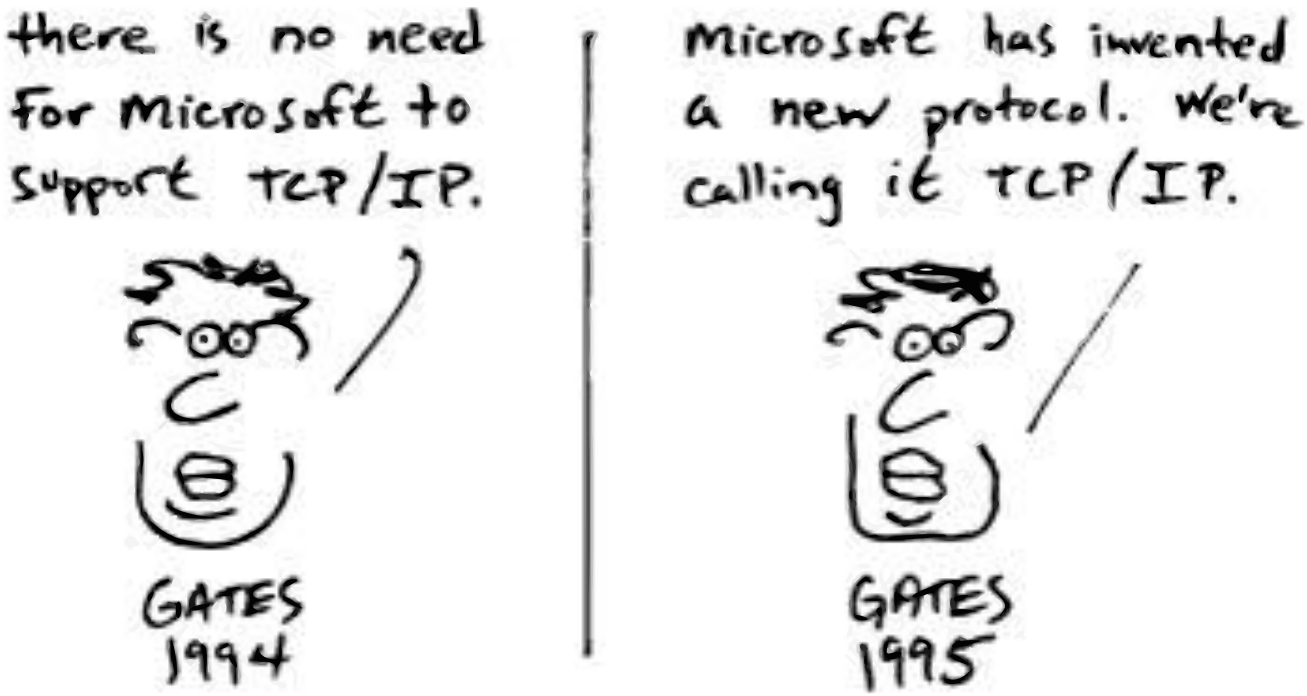



Figure : Netmask

- Per accedere ad Internet, ogni computer nella rete deve sapere quale host condivide la connessione ad Internet, il cosidetto **gateway** [5]

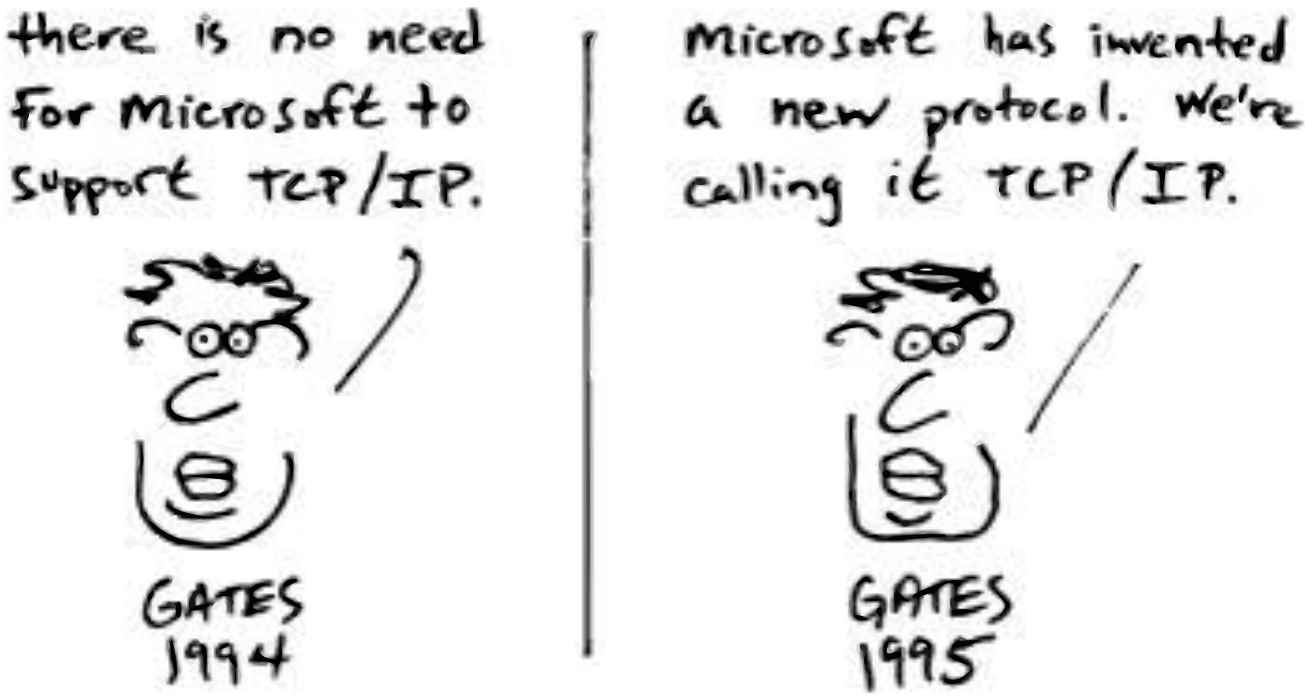
[5] Il gateway è anch'esso un regolare host e pertanto possiede un ip, tipicamente il primo o l'ultimo della rete (ad esempio 192.168.0.1)

# TCP/IP




- TCP è un layer sopra il protocollo IP che assicura il corretto recapito dei pacchetti e in caso di fallimento provvede a reinviarli
- Inoltre permette connessioni multiple tra due indirizzi IP
- Queste connessioni sono effettuate tramite **porte**, identificate da

# TCP/IP




- TCP è un layer sopra il protocollo IP che assicura il corretto recapito dei pacchetti e in caso di fallimento provvede a reinviarli
- Inoltre permette connessioni multiple tra due indirizzi IP
- Queste connessioni sono effettuate tramite **porte**, identificate da

# ip: indirizzi IP

```
$ ip address show

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default

    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo

           valid_lft forever preferred_lft forever

    inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000

    link/ether 04:01:47:d6:2a:01 brd ff:ff:ff:ff:ff:ff
    inet 178.62.152.115/18 brd 178.62.191.255 scope global eth0

           valid_lft forever preferred_lft forever

    inet6 2a03:b0c0:0:1010::85:5001/64 scope global

           valid_lft forever preferred_lft forever

    inet6 fe80::601:47ff:fed6:2a01/64 scope link

           valid_lft forever preferred_lft forever

3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000

    link/ether 04:01:47:d6:2a:02 brd ff:ff:ff:ff:ff:ff
    inet 10.129.212.70/16 brd 10.129.255.255 scope global eth1

           valid_lft forever preferred_lft forever

    inet6 fe80::601:47ff:fed6:2a02/64 scope link

           valid_lft forever preferred_lft forever
```

# ip: indirizzi IP

```
$ ip address show

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default

       link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
       inet 127.0.0.1/8 scope host lo

              valid_lft forever preferred_lft forever

       inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000

       link/ether 04:01:47:d6:2a:01 brd ff:ff:ff:ff:ff:ff
       inet 178.62.152.115/18 brd 178.62.191.255 scope global eth0

              valid_lft forever preferred_lft forever

       inet6 2a03:b0c0:0:1010::85:5001/64 scope global

              valid_lft forever preferred_lft forever

       inet6 fe80::601:47ff:fed6:2a01/64 scope link

              valid_lft forever preferred_lft forever

3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000

       link/ether 04:01:47:d6:2a:02 brd ff:ff:ff:ff:ff:ff
       inet 10.129.212.70/16 brd 10.129.255.255 scope global eth1

              valid_lft forever preferred_lft forever

       inet6 fe80::601:47ff:fed6:2a02/64 scope link

              valid_lft forever preferred_lft forever
```

# iproute2: Route

- Possiamo visualizzare le rotte configurate dal sistema

```
$ ip route show
default via 192.168.0.254 dev eth0
192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.0.2  metric 2
```

- E aggiungerne/rimuoverne

```
$ ip route add 1.2.3.0/24 via 192.168.0.254 dev eth0
$ ip route del 1.2.3.4 via 192.168.0.254 dev eth0
```

- Per poter "collegarci" ad Internet dobbiamo avere una route di default, delegata ad un gateway presente sulla nostra rete locale

```
$ ip route add default via 192.168.0.254 dev eth0
```

# iproute2: Route

- Possiamo visualizzare le rotte configurate dal sistema

```
$ ip route show
default via 192.168.0.254 dev eth0
192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.0.2  metric 2
```

- E aggiungerne/rimuoverne

```
$ ip route add 1.2.3.0/24 via 192.168.0.254 dev eth0
$ ip route del 1.2.3.4 via 192.168.0.254 dev eth0
```

- Per poter "collegarci" ad Internet dobbiamo avere una route di default, delegata ad un gateway presente sulla nostra rete locale

```
$ ip route add default via 192.168.0.254 dev eth0
```

# iproute2: Route

- Possiamo visualizzare le rotte configurate dal sistema

```
$ ip route show
default via 192.168.0.254 dev eth0
192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.0.2  metric 2
```

- E aggiungerne/rimuoverne

```
$ ip route add 1.2.3.0/24 via 192.168.0.254 dev eth0
$ ip route del 1.2.3.4 via 192.168.0.254 dev eth0
```

- Per poter "collegarci" ad Internet dobbiamo avere una route di default, delegata ad un gateway presente sulla nostra rete locale

```
$ ip route add default via 192.168.0.254 dev eth0
```

# Routing

- Ogni host GNU/Linux è anche un router
  Per permettere il “forwarding” di pacchetti ip dobbiamo abilitarlo con

  ```
  echo "1" > /proc/sys/net/ipv4/ip_forward
  ```

- Altri host possono utilizzare noi come router (o gateway di default) e noi provvederemo ad inoltrare Il traffico in base alle nostre regole di routing

# Routing

- Ogni host GNU/Linux è anche un router
  Per permettere il “forwarding” di pacchetti ip dobbiamo abilitarlo con

  ```
  echo "1" > /proc/sys/net/ipv4/ip_forward
  ```

- Altri host possono utilizzare noi come router (o gateway di default) e noi provvederemo ad inoltrare Il traffico in base alle nostre regole di routing

# DHCP

- Se non si vuole assegnare un indirizzo IP statico alla macchina si può utilizzare un client **DHCP** per ottenere dal server DHCP installato nella rete locale tutte le informazioni necessarie per la connessione
- DHCP (Dynamic Host Configuration Protocol) permette di ricevere automaticamente le informazioni di networking (IP address, netmask, broadcast address, gateway, nameservers etc.), tramite i tool *dhcpcd/dhclient*

```
$ dhclient eth0
Internet Systems Consortium DHCP Client V3.1.1
Copyright 2004-2008 Internet Systems Consortium.
All rights reserved. For info, please visit http://www.isc.org/sw/dhcp/
Listening on LPF/eth0/6e:5f:98:37:0c:07
Sending on   LPF/eth0/6e:5f:98:37:0c:07
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 4
DHCPOFFER from 10.5.5.1
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 10.5.5.1
bound to 10.5.5.26 -- renewal in 297 seconds.
```

# Configurazione su Debian/Ubuntu

- Su Debian e Ubuntu possiamo impostare le interfacce di rete in modo permanente tramite il file */etc/network/interfaces*, che verrà letto e utilizzato all'avvio del sistema per configurare le interfacce

```
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
address 192.168.0.2
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.0.254
```

- Se utilizziamo DHCP è più semplicemente

```
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
```

# Configurazione su Debian/Ubuntu

- Su Debian e Ubuntu possiamo impostare le interfacce di rete in modo permanente tramite il file */etc/network/interfaces*, che verrà letto e utilizzato all'avvio del sistema per configurare le interfacce

```
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
address 192.168.0.2
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.0.254
```

- Se utilizziamo DHCP è più semplicemente

```
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
```

# Configurazione su Debian/Ubuntu

- Su Debian e Ubuntu possiamo impostare le interfacce di rete in modo permanente tramite il file */etc/network/interfaces*, che verrà letto e utilizzato all'avvio del sistema per configurare le interfacce

```
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
address 192.168.0.2
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.0.254
```

- Se utilizziamo DHCP è più semplicemente

```
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
```

# DNS

- Dato che ogni host è identificato da un indirizzo IP per raggiungerlo tramite nome abbiamo bisogno di un servizio che traduca i nomi in indirizzi IP
- DNS (Domain Name Service) effettua questo servizio, per cui è necessario che i name server siano definiti in /etc/resolv.conf

```
$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 192.168.0.1
search local
```

- In generale il gateway di default agisce da nameserver, ma alternativamente si possono usare dei nameserver pubblici

| Google | Opendns |
|---|---|
| 8.8.8.8 | 208.67.222.222 |
| 8.8.4.4 | 208.67.220.220 |

# proxies

Se si accede ad internet tramite proxy, allora è necessario configurare le impostazioni del proxy durante la configurazione di rete[6]
Nella maggior parte dei casi è sufficiente definire le variabili usando gli hostname dei server, ad esempio proxy.poul.org con la porta 8080

- HTTP proxy (traffico HTTP e HTTPS)

```
root #export http_proxy="http://proxy.poul.org:8080"
```

- FTP proxy

```
root #export ftp_proxy="ftp://proxy.poul.org:8080"
```

- RSYNC proxy

```
root #export RSYNC_PROXY="proxy.poul.org:8080"
```

[6]Se il proxy richiede una combinazione username-password, la usata la seguente sintassi per la variabile: http://username:password@proxy.poul.org:8080

# proxies

Se si accede ad internet tramite proxy, allora è necessario configurare le impostazioni del proxy durante la configurazione di rete[6]
Nella maggior parte dei casi è sufficiente definire le variabili usando gli hostname dei server, ad esempio proxy.poul.org con la porta 8080

- HTTP proxy (traffico HTTP e HTTPS)

```
root #export http_proxy="http://proxy.poul.org:8080"
```

- FTP proxy

```
root #export ftp_proxy="ftp://proxy.poul.org:8080"
```

- RSYNC proxy

```
root #export RSYNC_PROXY="proxy.poul.org:8080"
```

[6]Se il proxy richiede una combinazione username-password, la usata la seguente sintassi per la variabile: http://username:password@proxy.poul.org:8080

# proxies

Se si accede ad internet tramite proxy, allora è necessario configurare le impostazioni del proxy durante la configurazione di rete[6]
Nella maggior parte dei casi è sufficiente definire le variabili usando gli hostname dei server, ad esempio proxy.poul.org con la porta 8080

- HTTP proxy (traffico HTTP e HTTPS)

```
root #export http_proxy="http://proxy.poul.org:8080"
```

- FTP proxy

```
root #export ftp_proxy="ftp://proxy.poul.org:8080"
```

- RSYNC proxy

```
root #export RSYNC_PROXY="proxy.poul.org:8080"
```

[6]Se il proxy richiede una combinazione username-password, la usata la seguente sintassi per la variabile: http://username:password@proxy.poul.org:8080

# proxies

Se si accede ad internet tramite proxy, allora è necessario configurare le impostazioni del proxy durante la configurazione di rete[6]
Nella maggior parte dei casi è sufficiente definire le variabili usando gli hostname dei server, ad esempio proxy.poul.org con la porta 8080

- HTTP proxy (traffico HTTP e HTTPS)

```
root #export http_proxy="http://proxy.poul.org:8080"
```

- FTP proxy

```
root #export ftp_proxy="ftp://proxy.poul.org:8080"
```

- RSYNC proxy

```
root #export RSYNC_PROXY="proxy.poul.org:8080"
```

[6]Se il proxy richiede una combinazione username-password, la usata la seguente sintassi per la variabile: http://username:password@proxy.poul.org:8080

# Testing the network: ping

- Per testare la rete di solito si usa il comando *ping*

```
root@Hello:~# ping -c 3 www.poul.org
PING poul.org (91.121.220.9) 56(84) bytes of data.
64 bytes from poul.org (91.121.220.9): icmp_seq=1 ttl=59 time=7.93 ms
64 bytes from poul.org (91.121.220.9): icmp_seq=2 ttl=59 time=6.59 ms
64 bytes from poul.org (91.121.220.9): icmp_seq=3 ttl=59 time=6.74 ms
--- poul.org ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 6.593/7.088/7.932/0.603 ms
```

# Wireless Access

Quando si utilizza una scheda wireless (802.11), prima di tutto occorre definire le impostazioni di configurazione. Per accedere a tali impostazioni si usa *iwconfig*

```
root # iwconfig eth0
eth0      IEEE 802.11-DS  ESSID:"GentooNode"
          Mode:Managed  Frequency:2.442GHz  Access Point: 00:09:5B:11:CC:F2
          Bit Rate:11Mb/s   Tx-Power=20 dBm   Sensitivity=0/65535
          Retry limit:16   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality:25/10  Signal level:-51 dBm  Noise level:-102 dBm
          Rx invalid nwid:5901 Rx invalid crypt:0 Rx invalid frag:0 Tx
          excessive retries:237 Invalid misc:350282 Missed beacon:84
```

Figure : iwconfig

# Wireless Access

```
iwconfig wlan0 essid POuL-Spot
```

- Per collegarsi ad Access Point che richiedono configurazioni di sicurezza avanzate serve *wpa_supplicant*

```
sudo wpa_supplicant -iwlan0 -c/etc/wpa_supplicant.conf
 -Dnl80211,wext
```

# Wireless Access

```
iwconfig wlan0 essid POuL-Spot
```

- Per collegarsi ad Access Point che richiedono configurazioni di sicurezza avanzate serve *wpa_supplicant*

```
sudo wpa_supplicant -iwlan0 -c/etc/wpa_supplicant.conf
 -Dnl80211,wext
```

# Wireless Access

```
iwconfig wlan0 essid POuL-Spot
```

- Per collegarsi ad Access Point che richiedono configurazioni di sicurezza avanzate serve *wpa_supplicant*

```
sudo wpa_supplicant -iwlan0 -c/etc/wpa_supplicant.conf
 -Dnl80211,wext
```

# Wireless Access

```
$ cat /etc/wpa_supplicant.conf
# The below line not be changed otherwise wpa_supplicant refuses to work ctrl_

# Ensure that only root can read the WPA configuration
ctrl_interface_group=0

# Let wpa_supplicant take care of scanning and AP selection
ap_scan=1

#Simple case: WPA-PSK, PSK as an ASCII passphrase, allow all valid ciphers
network={
ssid="simple"
psk="very secret passphrase"
#request SSID-specific scanning (for APs that reject # broadcast SSID)
scan_ssid=1
# The higher the priority the sooner we are matched
priority=5
}
```

# Recap

La Configurazione di una rete si può sintetizzare in 3 step:

1. Assegnazione di un IP

```
root #ip address add 192.168.0.2/24 dev eth0
```

1. Configurazione del corretto routing

```
ip route add 1.2.3.0/24 via 192.168.0.1 dev eth0
```

1. Impostazione dei DNS in /etc/resolv.conf

```
root #nano -w /etc/resolv.conf
nameserver 8.8.8.8
nameserver 8.8.4.4
```

# Recap

La Configurazione di una rete si può sintetizzare in 3 step:

1. Assegnazione di un IP

```
root #ip address add 192.168.0.2/24 dev eth0
```

1. Configurazione del corretto routing

```
ip route add 1.2.3.0/24 via 192.168.0.1 dev eth0
```

1. Impostazione dei DNS in /etc/resolv.conf

```
root #nano -w /etc/resolv.conf
nameserver 8.8.8.8
nameserver 8.8.4.4
```

# Recap

La Configurazione di una rete si può sintetizzare in 3 step:

1. Assegnazione di un IP

```
root #ip address add 192.168.0.2/24 dev eth0
```

1. Configurazione del corretto routing

```
ip route add 1.2.3.0/24 via 192.168.0.1 dev eth0
```

1. Impostazione dei DNS in /etc/resolv.conf

```
root #nano -w /etc/resolv.conf
nameserver 8.8.8.8
nameserver 8.8.4.4
```

# Recap

| Network Item | Example |
|---|---|
| The system IP address | 192.168.0.2 |
| Netmask | 255.255.255.0 |
| Broadcast | 192.168.0.255 |
| Gateway | 192.168.0.1 |
| Nameserver(s) | 195.130.130.5, 195.130.130.133 |

# Sezione 3

# ping

- Per diagnostica di rete si utilizza solitamente il comando ping

```
root@Hello:~# ping -c3 poul.org
PING poul.org (91.121.220.9) 56(84) bytes of data.
64 bytes from poul.org (91.121.220.9): icmp_seq=1 ttl=59 time=6.83 ms
64 bytes from poul.org (91.121.220.9): icmp_seq=2 ttl=59 time=6.72 ms
64 bytes from poul.org (91.121.220.9): icmp_seq=3 ttl=59 time=6.68 ms
--- poul.org ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 6.685/6.748/6.839/0.093 ms
```

# ping

- Per diagnostica di rete si utilizza solitamente il comando ping

```
root@Hello:~# ping -c3 poul.org
PING poul.org (91.121.220.9) 56(84) bytes of data.
64 bytes from poul.org (91.121.220.9): icmp_seq=1 ttl=59 time=6.83 ms
64 bytes from poul.org (91.121.220.9): icmp_seq=2 ttl=59 time=6.72 ms
64 bytes from poul.org (91.121.220.9): icmp_seq=3 ttl=59 time=6.68 ms
--- poul.org ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 6.685/6.748/6.839/0.093 ms
```

# host

- Possiamo controllare la corretta risoluzione dei nomi utilizzando il comando *host*

```
$ host poul.org
poul.org has address 91.121.182.81
poul.org has IPv6 address 2001:41d0:1:f751::1
poul.org mail is handled by 1 poul.org.

$ host -t MX poul.org
poul.org mail is handled by 1 poul.org.
```

# host

- Possiamo controllare la corretta risoluzione dei nomi utilizzando il comando *host*

```
$ host poul.org
poul.org has address 91.121.182.81
poul.org has IPv6 address 2001:41d0:1:f751::1
poul.org mail is handled by 1 poul.org.
```

```
$ host -t MX poul.org
poul.org mail is handled by 1 poul.org.
```

# host

- Possiamo controllare la corretta risoluzione dei nomi utilizzando il comando *host*

```
$ host poul.org
poul.org has address 91.121.182.81
poul.org has IPv6 address 2001:41d0:1:f751::1
poul.org mail is handled by 1 poul.org.

$ host -t MX poul.org
poul.org mail is handled by 1 poul.org.
```

# Netcat

- Uno dei tool per fare scripting di rete e debugging è netcat
- Netcat è un client/server tcp/udp da linea di comando. Una sorta di «telnet client» avanzato

```
$ nc [hostname] [port]
$ nc -l -p [port]
```

- *-l -p [porta]* per restare in ascolto per connessioni sulla porta data
- *-u* per indicare l'utilizzo di UDP invece di TCP
- *-v* per avere informazioni aggiuntive durante la connessione

# Netcat

- Uno dei tool per fare scripting di rete e debugging è netcat
- Netcat è un client/server tcp/udp da linea di comando. Una sorta di «telnet client» avanzato

```
$ nc [hostname] [port]
$ nc -l -p [port]
```

- *-l -p [porta]* per restare in ascolto per connessioni sulla porta data
- *-u* per indicare l'utilizzo di UDP invece di TCP
- *-v* per avere informazioni aggiuntive durante la connessione

# Netcat

- Uno dei tool per fare scripting di rete e debugging è netcat
- Netcat è un client/server tcp/udp da linea di comando. Una sorta di «telnet client» avanzato

```
$ nc [hostname] [port]
$ nc -l -p [port]
```

- *-l -p [porta]* per restare in ascolto per connessioni sulla porta data
- *-u* per indicare l'utilizzo di UDP invece di TCP
- *-v* per avere informazioni aggiuntive durante la connessione

# Netcat

- Uno dei tool per fare scripting di rete e debugging è netcat
- Netcat è un client/server tcp/udp da linea di comando. Una sorta di «telnet client» avanzato

```
$ nc [hostname] [port]
$ nc -l -p [port]
```

- *-l -p [porta]* per restare in ascolto per connessioni sulla porta data
- *-u* per indicare l'utilizzo di UDP invece di TCP
- *-v* per avere informazioni aggiuntive durante la connessione

# Netcat

- Uno dei tool per fare scripting di rete e debugging è netcat
- Netcat è un client/server tcp/udp da linea di comando. Una sorta di «telnet client» avanzato

```
$ nc [hostname] [port]
$ nc -l -p [port]
```

- *-l -p [porta]* per restare in ascolto per connessioni sulla porta data
- *-u* per indicare l'utilizzo di UDP invece di TCP
- *-v* per avere informazioni aggiuntive durante la connessione

# Sezione 4

# Firewalling

- Un firewall è un “programma” che si occupa di decidere se accettare o meno i pacchetti in transito su un host
- Un firewall serve per:
  - Evitare connessioni non autorizzate
  - Correzione dei pacchetti durante il filtraggio degli stessi
  - Applicare strategie di NATting agli host che sono dietro un firewall server
- “Blocchiamo” certi tipi di pacchetti in transito, in base a certe regole

# Firewalling

- Un firewall è un “programma” che si occupa di decidere se accettare o meno i pacchetti in transito su un host
- Un firewall serve per:
  - Evitare connessioni non autorizzate
  - Correzione dei pacchetti durante il filtraggio degli stessi
  - Applicare strategie di NATting agli host che sono dietro un firewall server
- “Blocchiamo” certi tipi di pacchetti in transito, in base a certe regole

# Firewalling

- Un firewall è un “programma” che si occupa di decidere se accettare o meno i pacchetti in transito su un host
- Un firewall serve per:
  - Evitare connessioni non autorizzate
  - Correzione dei pacchetti durante il filtraggio degli stessi
  - Applicare strategie di NATting agli host che sono dietro un firewall server
- “Blocchiamo” certi tipi di pacchetti in transito, in base a certe regole

# Firewalling

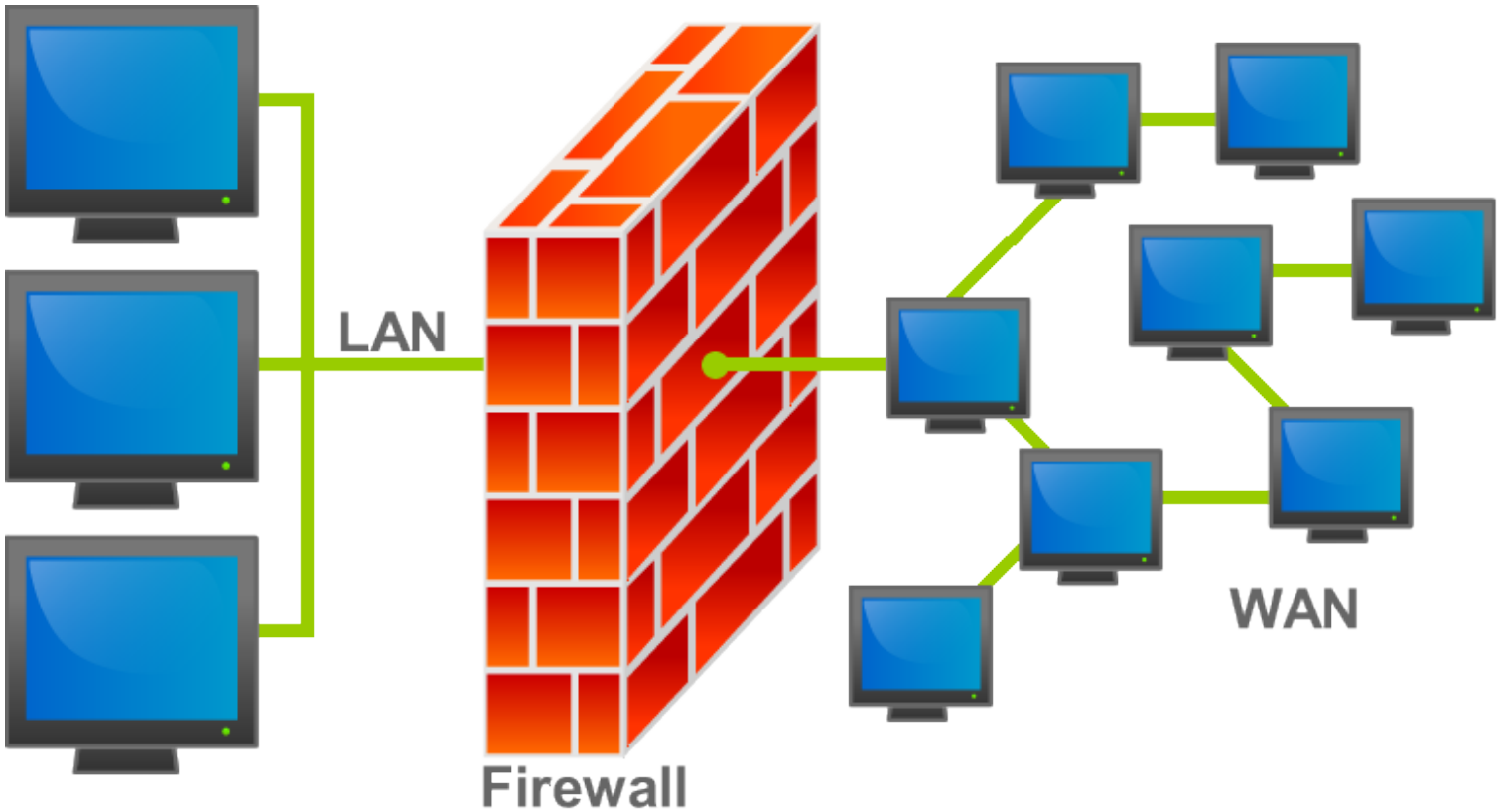
- Un firewall è un “programma” che si occupa di decidere se accettare o meno i pacchetti in transito su un host
- Un firewall serve per:
  - Evitare connessioni non autorizzate
  - Correzione dei pacchetti durante il filtraggio degli stessi
  - Applicare strategie di NATting agli host che sono dietro un firewall server
- “Blocchiamo” certi tipi di pacchetti in transito, in base a certe regole

# Firewalling

- Un firewall è un “programma” che si occupa di decidere se accettare o meno i pacchetti in transito su un host
- Un firewall serve per:
  - Evitare connessioni non autorizzate
  - Correzione dei pacchetti durante il filtraggio degli stessi
  - Applicare strategie di NATting agli host che sono dietro un firewall server
- “Blocchiamo” certi tipi di pacchetti in transito, in base a certe regole

# Firewalling

- Il firewall deve essere l'unico punto di contatto della rete con l'esterno[7]




[7] Per evitare il problema di generare *Single Point Of Failure* è eventualmente possibile utilizzare più firewall sincronizzati costantemente tra di loro: `http://conntrack-tools.netfilter.org/`

# Firewalling

- All'interno del kernel Linux è presente una parte dedicata esclusivamente a gestire le operazioni di firewalling, chiamata ***netfilter***
- In userspace possiamo utilizzare **iptables/ip6tables/ebtables** come strumenti per comunicare al kernel le regole da applicare[8] a livello IP(v4 e v6) e (se necessario) Ethernet/datalink
- Netfilter è un firewall di tipo ***stateful***, cioè permette di tenere traccia delle connessioni che lo attraversano e del loro stato
- Netfilter si occupa anche di implementare le operazioni di NAT e port forwarding spesso utilizzate

[8] A breve si migrerà ad un nuovo unico tool, chiamato **nftables**, che tuttavia non modifica la struttura delle tabelle e hooks che vedremo tra un attimo.

# Firewalling

- All'interno del kernel Linux è presente una parte dedicata esclusivamente a gestire le operazioni di firewalling, chiamata ***netfilter***
- In userspace possiamo utilizzare **iptables/ip6tables/ebtables** come strumenti per comunicare al kernel le regole da applicare[8] a livello IP(v4 e v6) e (se necessario) Ethernet/datalink
- Netfilter è un firewall di tipo ***stateful***, cioè permette di tenere traccia delle connessioni che lo attraversano e del loro stato
- Netfilter si occupa anche di implementare le operazioni di NAT e port forwarding spesso utilizzate

[8] A breve si migrerà ad un nuovo unico tool, chiamato **nftables**, che tuttavia non modifica la struttura delle tabelle e hooks che vedremo tra un attimo.

# Firewalling

- All'interno del kernel Linux è presente una parte dedicata esclusivamente a gestire le operazioni di firewalling, chiamata ***netfilter***
- In userspace possiamo utilizzare **iptables/ip6tables/ebtables** come strumenti per comunicare al kernel le regole da applicare[8] a livello IP(v4 e v6) e (se necessario) Ethernet/datalink
- Netfilter è un firewall di tipo ***stateful***, cioè permette di tenere traccia delle connessioni che lo attraversano e del loro stato
- Netfilter si occupa anche di implementare le operazioni di NAT e port forwarding spesso utilizzate

[8] A breve si migrerà ad un nuovo unico tool, chiamato **nftables**, che tuttavia non modifica la struttura delle tabelle e hooks che vedremo tra un attimo.

# Firewalling

- All'interno del kernel Linux è presente una parte dedicata esclusivamente a gestire le operazioni di firewalling, chiamata ***netfilter***
- In userspace possiamo utilizzare **iptables/ip6tables/ebtables** come strumenti per comunicare al kernel le regole da applicare[8] a livello IP(v4 e v6) e (se necessario) Ethernet/datalink
- Netfilter è un firewall di tipo ***stateful***, cioè permette di tenere traccia delle connessioni che lo attraversano e del loro stato
- Netfilter si occupa anche di implementare le operazioni di NAT e port forwarding spesso utilizzate

[8] A breve si migrerà ad un nuovo unico tool, chiamato **nftables**, che tuttavia non modifica la struttura delle tabelle e hooks che vedremo tra un attimo.

# Netfilter

- L'infrastruttura di Netfilter è basata su 5 "hooks" presenti nel percorso che i pacchetti devono compiere all'interno del sistema.
- In ognuno di questi "punti" è possibile
  - Permettere il passaggio del pacchetto (ACCEPT)
  - Scartare il pacchetto e interromperlo (DROP)
  - Redirigere il pacchetto (REDIRECT)
  - Modificare alcuni campi del pacchetto (MANGLED)

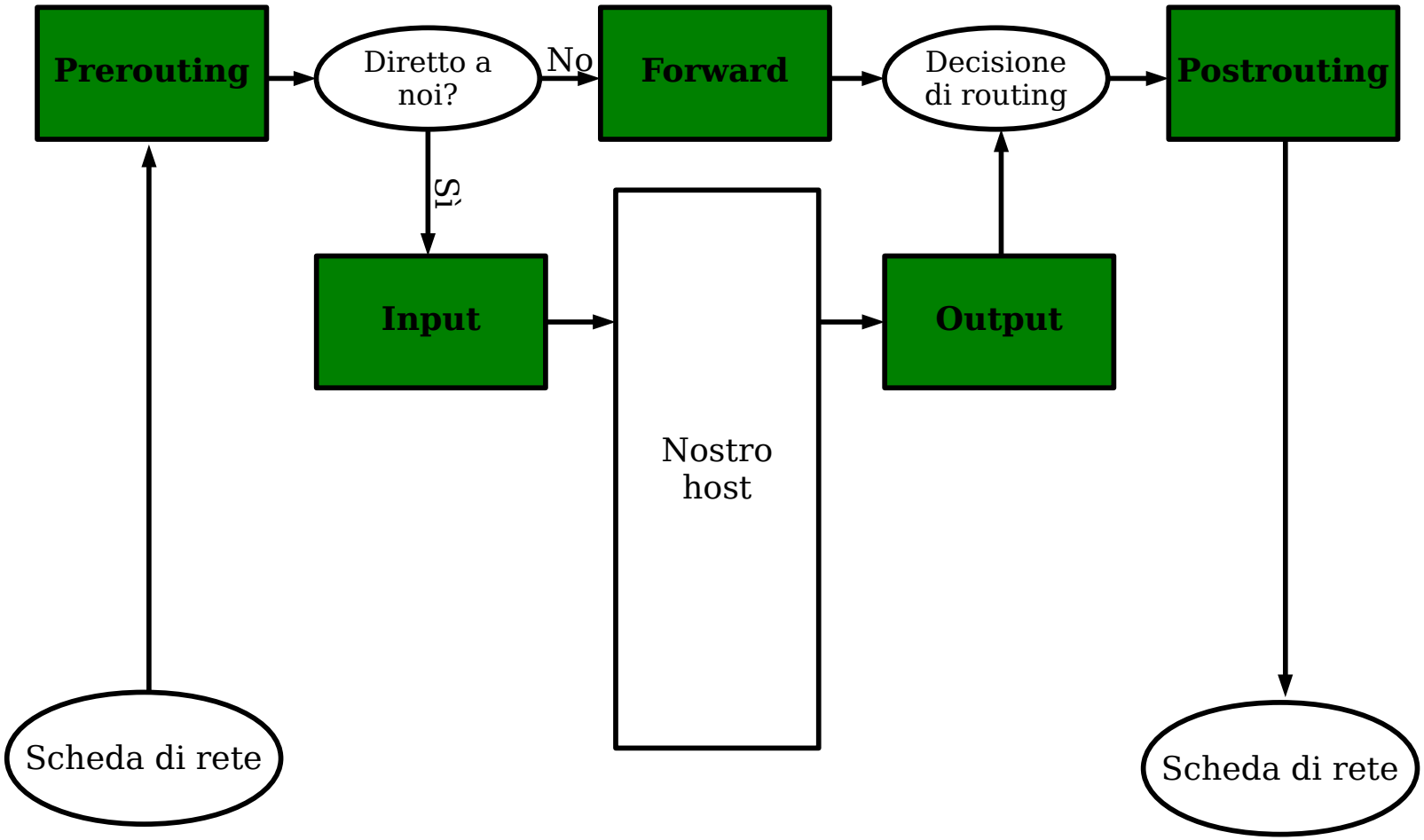
# Netfilter

- L'infrastruttura di Netfilter è basata su 5 "hooks" presenti nel percorso che i pacchetti devono compiere all'interno del sistema.
- In ognuno di questi "punti" è possibile
  - Permettere il passaggio del pacchetto (ACCEPT)
  - Scartare il pacchetto e interromperlo (DROP)
  - Redirigere il pacchetto (REDIRECT)
  - Modificare alcuni campi del pacchetto (MANGLED)

# Netfilter

- L'infrastruttura di Netfilter è basata su 5 "hooks" presenti nel percorso che i pacchetti devono compiere all'interno del sistema.
- In ognuno di questi "punti" è possibile
  - Permettere il passaggio del pacchetto (ACCEPT)
  - Scartare il pacchetto e interromperlo (DROP)
  - Redirigere il pacchetto (REDIRECT)
  - Modificare alcuni campi del pacchetto (MANGLED)

# Netfilter


Prerouting
Diretto a noi?
No
Forward
Decisione di routing
Postrouting
Si
Input
Nostro host
Output
Scheda di rete
Scheda di rete

# Hooks

- **PREROUTING**: pacchetti in ingresso
- **INPUT**: pacchetti destinati all'host stesso
- **FORWARD**: pacchetti non destinati all'host stesso, che dobbiamo inoltrare
- **OUTPUT**: pacchetti in uscita dall'host stesso
- **POSTROUTING**: pacchetti in uscita

# Hooks

- **PREROUTING**: pacchetti in ingresso
- **INPUT**: pacchetti destinati all'host stesso
- **FORWARD**: pacchetti non destinati all'host stesso, che dobbiamo inoltrare
- **OUTPUT**: pacchetti in uscita dall'host stesso
- **POSTROUTING**: pacchetti in uscita

# Hooks

- **PREROUTING**: pacchetti in ingresso
- **INPUT**: pacchetti destinati all'host stesso
- **FORWARD**: pacchetti non destinati all'host stesso, che dobbiamo inoltrare
- **OUTPUT**: pacchetti in uscita dall'host stesso
- **POSTROUTING**: pacchetti in uscita

# Hooks

- **PREROUTING**: pacchetti in ingresso
- **INPUT**: pacchetti destinati all'host stesso
- **FORWARD**: pacchetti non destinati all'host stesso, che dobbiamo inoltrare
- **OUTPUT**: pacchetti in uscita dall'host stesso
- **POSTROUTING**: pacchetti in uscita

# Hooks

- **PREROUTING**: pacchetti in ingresso
- **INPUT**: pacchetti destinati all'host stesso
- **FORWARD**: pacchetti non destinati all'host stesso, che dobbiamo inoltrare
- **OUTPUT**: pacchetti in uscita dall'host stesso
- **POSTROUTING**: pacchetti in uscita

# Netfilter

- Gli “hooks” non possono essere utilizzati direttamente. Non possiamo metterci delle regole dentro
- Le regole devono essere inserite all’interno di **tabelle**. Ogni tabella ha dei punti di aggancio agli hook: le chain
- Le tabelle utilizzate sono: *filter*, *mangle*, *nat* e *raw*
  - filter è la tabella con le regole principali
  - mangle è la tabella dove risiedono le regole che modificano i pacchetti
  - nat è la tabella dove avvengono le operazioni che permettono il NAT

# Netfilter

- Gli “hooks” non possono essere utilizzati direttamente. Non possiamo metterci delle regole dentro
- Le regole devono essere inserite all’interno di **tabelle**. Ogni tabella ha dei punti di aggancio agli hook: le chain
- Le tabelle utilizzate sono: *filter*, *mangle*, *nat* e *raw*
  - filter è la tabella con le regole principali
  - mangle è la tabella dove risiedono le regole che modificano i pacchetti
  - nat è la tabella dove avvengono le operazioni che permettono il NAT

# Netfilter

- Gli “hooks” non possono essere utilizzati direttamente. Non possiamo metterci delle regole dentro
- Le regole devono essere inserite all’interno di **tabelle**. Ogni tabella ha dei punti di aggancio agli hook: le chain
- Le tabelle utilizzate sono: *filter*, *mangle*, *nat* e *raw*
  - filter è la tabella con le regole principali
  - mangle è la tabella dove risiedono le regole che modificano i pacchetti
  - nat è la tabella dove avvengono le operazioni che permettono il NAT

# Netfilter

- Gli “hooks” non possono essere utilizzati direttamente. Non possiamo metterci delle regole dentro
- Le regole devono essere inserite all’interno di **tabelle**. Ogni tabella ha dei punti di aggancio agli hook: le chain
- Le tabelle utilizzate sono: *filter*, *mangle*, *nat* e *raw*
  - filter è la tabella con le regole principali
  - mangle è la tabella dove risiedono le regole che modificano i pacchetti
  - nat è la tabella dove avvengono le operazioni che permettono il NAT

# Netfilter

- Gli “hooks” non possono essere utilizzati direttamente. Non possiamo metterci delle regole dentro
- Le regole devono essere inserite all’interno di **tabelle**. Ogni tabella ha dei punti di aggancio agli hook: le chain
- Le tabelle utilizzate sono: *filter*, *mangle*, *nat* e *raw*
  - filter è la tabella con le regole principali
  - mangle è la tabella dove risiedono le regole che modificano i pacchetti
  - nat è la tabella dove avvengono le operazioni che permettono il NAT

# Recap

- il pacchetto segue il suo percorso
- quando il pacchetto arriva in un hook si vanno a consultare le varie tabelle che hanno regole in quell'hook (le tabelle da verificare sono in un certo ordine)
- il pacchetto entra nella *chain* di una tabella e lì vengono consultate tutte le regole presenti
- se al termine delle verifiche, la decisione è "ACCEPT", allora il pacchetto prosegue il suo percorso e viene verificato nelle tabelle successive
- poi se il pacchetto prosegue ancora passerà nell'hook successivo

# Recap

- il pacchetto segue il suo percorso
- quando il pacchetto arriva in un hook si vanno a consultare le varie tabelle che hanno regole in quell'hook (le tabelle da verificare sono in un certo ordine)
- il pacchetto entra nella *chain* di una tabella e lì vengono consultate tutte le regole presenti
- se al termine delle verifiche, la decisione è "ACCEPT", allora il pacchetto prosegue il suo percorso e viene verificato nelle tabelle successive
- poi se il pacchetto prosegue ancora passerà nell'hook successivo

# Recap

- il pacchetto segue il suo percorso
- quando il pacchetto arriva in un hook si vanno a consultare le varie tabelle che hanno regole in quell'hook (le tabelle da verificare sono in un certo ordine)
- il pacchetto entra nella *chain* di una tabella e lì vengono consultate tutte le regole presenti
- se al termine delle verifiche, la decisione è "ACCEPT", allora il pacchetto prosegue il suo percorso e viene verificato nelle tabelle successive
- poi se il pacchetto prosegue ancora passerà nell'hook successivo

# Recap

- il pacchetto segue il suo percorso
- quando il pacchetto arriva in un hook si vanno a consultare le varie tabelle che hanno regole in quell'hook (le tabelle da verificare sono in un certo ordine)
- il pacchetto entra nella *chain* di una tabella e lì vengono consultate tutte le regole presenti
- se al termine delle verifiche, la decisione è "ACCEPT", allora il pacchetto prosegue il suo percorso e viene verificato nelle tabelle successive
- poi se il pacchetto prosegue ancora passerà nell'hook successivo

# Recap

- il pacchetto segue il suo percorso
- quando il pacchetto arriva in un hook si vanno a consultare le varie tabelle che hanno regole in quell'hook (le tabelle da verificare sono in un certo ordine)
- il pacchetto entra nella *chain* di una tabella e lì vengono consultate tutte le regole presenti
- se al termine delle verifiche, la decisione è "ACCEPT", allora il pacchetto prosegue il suo percorso e viene verificato nelle tabelle successive
- poi se il pacchetto prosegue ancora passerà nell'hook successivo

# iptables

- Iniziamo a guardare la tabella di default, la tabella filter

```
$ sudo iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out     source     destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out     source     destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out     source     destination
```

# iptables

- Nella tabella filter abbiamo 3 chains che si agganciano agli hooks di **INPUT**, **FORWARD** e **OUTPUT**
- Ogni **regola** è costituta da due parti principali
  - *Match:* È la parte della regola che specifica quali sono le caratteristiche del pacchetto che cerchiamo
  - *Target*: È la parte della regola che specifica che operazione effettuare sul pacchetto che fa match
- Durante il percorso del pacchetto, quando si arriva nella chain della tabella, si analizzano una per una le regole presenti dalla prima all'ultima. Se nessuna regola fa **match**, si applica la *policy di default* della chain (ACCEPT o DROP)

# iptables

- Nella tabella filter abbiamo 3 chains che si agganciano agli hooks di **INPUT**, **FORWARD** e **OUTPUT**
- Ogni **regola** è costituta da due parti principali
  - *Match:* È la parte della regola che specifica quali sono le caratteristiche del pacchetto che cerchiamo
  - *Target*: È la parte della regola che specifica che operazione effettuare sul pacchetto che fa match
- Durante il percorso del pacchetto, quando si arriva nella chain della tabella, si analizzano una per una le regole presenti dalla prima all'ultima. Se nessuna regola fa **match**, si applica la *policy di default* della chain (ACCEPT o DROP)

# iptables

- Nella tabella filter abbiamo 3 chains che si agganciano agli hooks di **INPUT**, **FORWARD** e **OUTPUT**
- Ogni **regola** è costituta da due parti principali
  - *Match:* È la parte della regola che specifica quali sono le caratteristiche del pacchetto che cerchiamo
  - *Target*: È la parte della regola che specifica che operazione effettuare sul pacchetto che fa match
- Durante il percorso del pacchetto, quando si arriva nella chain della tabella, si analizzano una per una le regole presenti dalla prima all'ultima. Se nessuna regola fa **match**, si applica la *policy di default* della chain (ACCEPT o DROP)

# iptables

- Nella tabella filter abbiamo 3 chains che si agganciano agli hooks di **INPUT**, **FORWARD** e **OUTPUT**
- Ogni **regola** è costituta da due parti principali
  - *Match:* È la parte della regola che specifica quali sono le caratteristiche del pacchetto che cerchiamo
  - *Target*: È la parte della regola che specifica che operazione effettuare sul pacchetto che fa match
- Durante il percorso del pacchetto, quando si arriva nella chain della tabella, si analizzano una per una le regole presenti dalla prima all'ultima. Se nessuna regola fa **match**, si applica la *policy di default* della chain (ACCEPT o DROP)

# iptables: sintassi

```
iptables -A [CHAIN] [condizioni di match] -j [target]
iptables -D [CHAIN] [condizioni di match] -j [target]
iptables -I [CHAIN] [condizioni di match] -j [target]
```

```
iptables -P [CHAIN] [ACCEPT|DROP]
```

- Policy di default della chain

```
iptables -F [CHAIN]
```

- Per fare un flush delle regole nella chain (senza cambiare la policy di default)

# iptables: sintassi

```
iptables -A [CHAIN] [condizioni di match] -j [target]
iptables -D [CHAIN] [condizioni di match] -j [target]
iptables -I [CHAIN] [condizioni di match] -j [target]
```

```
iptables -P [CHAIN] [ACCEPT|DROP]
```

- Policy di default della chain

```
iptables -F [CHAIN]
```

- Per fare un flush delle regole nella chain (senza cambiare la policy di default)

# iptables: sintassi

```
iptables -A [CHAIN] [condizioni di match] -j [target]
iptables -D [CHAIN] [condizioni di match] -j [target]
iptables -I [CHAIN] [condizioni di match] -j [target]
```

```
iptables -P [CHAIN] [ACCEPT|DROP]
```

- Policy di default della chain

```
iptables -F [CHAIN]
```

- Per fare un flush delle regole nella chain (senza cambiare la policy di default)

# iptables: target

I possibili target sono:

- ACCEPT/DROP
- REJECT: il pacchetto viene scartato, ma in aggiunta si genera una notifica per il mittente
- LOG: Viene segnalato il match della regola nel file di log del kernel
- MIRROR: Scambia sorgente con destinazione e manda immediatamente il pacchetto senza passare per le altre regole
- RATEEST: il pacchetto passa nel sistema e inoltre viene tracciato secondo parametri statistici

# iptables: target

I possibili target sono:

- ACCEPT/DROP
- REJECT: il pacchetto viene scartato, ma in aggiunta si genera una notifica per il mittente
- LOG: Viene segnalato il match della regola nel file di log del kernel
- MIRROR: Scambia sorgente con destinazione e manda immediatamente il pacchetto senza passare per le altre regole
- RATEEST: il pacchetto passa nel sistema e inoltre viene tracciato secondo parametri statistici

# iptables: target

I possibili target sono:

- ACCEPT/DROP
- REJECT: il pacchetto viene scartato, ma in aggiunta si genera una notifica per il mittente
- LOG: Viene segnalato il match della regola nel file di log del kernel
- MIRROR: Scambia sorgente con destinazione e manda immediatamente il pacchetto senza passare per le altre regole
- RATEEST: il pacchetto passa nel sistema e inoltre viene tracciato secondo parametri statistici

# iptables: target

I possibili target sono:

- ACCEPT/DROP
- REJECT: il pacchetto viene scartato, ma in aggiunta si genera una notifica per il mittente
- LOG: Viene segnalato il match della regola nel file di log del kernel
- MIRROR: Scambia sorgente con destinazione e manda immediatamente il pacchetto senza passare per le altre regole
- RATEEST: il pacchetto passa nel sistema e inoltre viene tracciato secondo parametri statistici

# iptables: target

I possibili target sono:

- ACCEPT/DROP
- REJECT: il pacchetto viene scartato, ma in aggiunta si genera una notifica per il mittente
- LOG: Viene segnalato il match della regola nel file di log del kernel
- MIRROR: Scambia sorgente con destinazione e manda immediatamente il pacchetto senza passare per le altre regole
- RATEEST: il pacchetto passa nel sistema e inoltre viene tracciato secondo parametri statistici

# iptables: sintassi

```
iptables -A INPUT --source 192.168.0.153 -j DROP
```

- **-A** significa che vogliamo aggiungere questa regola **in coda alle altre** nella chain **INPUT**
- –**source** indica che la regola vale solo per i pacchetti IPv4 provenienti dall' indirizzo sorgente 192.168.0.153
- **-j DROP** specifica che se le precedenti condizioni di match sono soddisfatte, l'operazione da effettuare è un **jump** al target DROP, cioè scartiamo il pacchetto che non proseguirà nelle altre regole

# iptables: sintassi

```
iptables -A INPUT --source 192.168.0.153 -j DROP
```

- **-A** significa che vogliamo aggiungere questa regola **in coda alle altre** nella chain **INPUT**
- –**source** indica che la regola vale solo per i pacchetti IPv4 provenienti dall' indirizzo sorgente 192.168.0.153
- **-j DROP** specifica che se le precedenti condizioni di match sono soddisfatte, l'operazione da effettuare è un **jump** al target DROP, cioè scartiamo il pacchetto che non proseguirà nelle altre regole

# iptables:sintassi

- Nel match possiamo sempre utilizzare la negazione mettendo un punto esclamativo davanti ad una delle condizioni del match

```
iptables -A INPUT -s 192.168.0.0/24 ! -i eth1 -j DROP
```

- **-*s*** è come *–source*, utilizzando la notazione VLSM "/24" specifichiamo il **prefisso**, non l'indirizzo esatto
- **-i** specifica il match sull'interfaccia di arrivo del pacchetto

# iptables:sintassi

- Nel match possiamo sempre utilizzare la negazione mettendo un punto esclamativo davanti ad una delle condizioni del match

```
iptables -A INPUT -s 192.168.0.0/24 ! -i eth1 -j DROP
```

- **-*s*** è come *–source*, utilizzando la notazione VLSM "/24" specifichiamo il **prefisso**, non l'indirizzo esatto
- **-i** specifica il match sull'interfaccia di arrivo del pacchetto

# iptables:sintassi

- Nel match possiamo sempre utilizzare la negazione mettendo un punto esclamativo davanti ad una delle condizioni del match

```
iptables -A INPUT -s 192.168.0.0/24 ! -i eth1 -j DROP
```

- **-s** è come *–source*, utilizzando la notazione VLSM "/24" specifichiamo il **prefisso**, non l'indirizzo esatto
- **-i** specifica il match sull'interfaccia di arrivo del pacchetto

# Cancellare una regola

- Possiamo riscriverla uguale mettendo -D al posto di -A
- Oppure possiamo elencare le regole con i numeri a fianco

```
$ iptables -nvL --line-numbers
```

- ...e cancellare in base al numero

```
$ iptables -t [table] -D [CHAIN] [numero regola]
```

# Cancellare una regola

- Possiamo riscriverla uguale mettendo -D al posto di -A
- Oppure possiamo elencare le regole con i numeri a fianco

```
$ iptables -nvL --line-numbers
```

- ...e cancellare in base al numero

```
$ iptables -t [table] -D [CHAIN] [numero regola]
```

# ACCEPT/DROP

- Abbiamo visto che con DROP il pacchetto viene fermato
- Cosa succede con ACCEPT ?
  - Il pacchetto viene accettato e passa alla chain successiva
  - Nel caso di INPUT non c'è una chain successiva, quindi viene passato al sistema operativo
  - Però se ad esempio siamo in PREROUTING, non è detto che il pacchetto arrivi, dato che potrebbe essere bloccato in INPUT

# ACCEPT/DROP

- Abbiamo visto che con DROP il pacchetto viene fermato
- Cosa succede con ACCEPT ?
  - Il pacchetto viene accettato e passa alla chain successiva
  - Nel caso di INPUT non c'è una chain successiva, quindi viene passato al sistema operativo
  - Però se ad esempio siamo in PREROUTING, non è detto che il pacchetto arrivi, dato che potrebbe essere bloccato in INPUT

# ACCEPT/DROP

- Abbiamo visto che con DROP il pacchetto viene fermato
- Cosa succede con ACCEPT ?
  - Il pacchetto viene accettato e passa alla chain successiva
  - Nel caso di INPUT non c'è una chain successiva, quindi viene passato al sistema operativo
  - Però se ad esempio siamo in PREROUTING, non è detto che il pacchetto arrivi, dato che potrebbe essere bloccato in INPUT

# Match comuni

- Interfaccia di rete di *input/output* del pacchetto[9]
  - ***-i [iface]***
  - ***-o [iface]***
- Indirizzo sorgente/destinazione
  - ***-s [address or network prefix]***
  - ***-d [address or network prefix]***
- Protocollo di livello 4 e numeri di porta destinazione
  - ***-p udp –dport 53***
  - **-p tcp –dport 22**

[9] Require Common Sense: non ha senso usare -i nella chain di OUTPUT

# Match comuni

- Interfaccia di rete di *input/output* del pacchetto[9]
  - ***-i [iface]***
  - ***-o [iface]***
- Indirizzo sorgente/destinazione
  - ***-s [address or network prefix]***
  - ***-d [address or network prefix]***
- Protocollo di livello 4 e numeri di porta destinazione
  - ***-p udp –dport 53***
  - **-p tcp –dport 22**

[9] Require Common Sense: non ha senso usare -i nella chain di OUTPUT

# Match comuni

- Interfaccia di rete di *input/output* del pacchetto[9]
  - ***-i [iface]***
  - ***-o [iface]***
- Indirizzo sorgente/destinazione
  - ***-s [address or network prefix]***
  - ***-d [address or network prefix]***
- Protocollo di livello 4 e numeri di porta destinazione
  - ***-p udp –dport 53***
  - **-p tcp –dport 22**

[9] Require Common Sense: non ha senso usare -i nella chain di OUTPUT

# Match stateful

- Tutti i match che abbiamo visto fino ad ora sono “integrati” in netfilter. Molti altri match e quelli aggiuntivi necessitano quasi sempre di “***-m nome_modulo_match***”, per dire a iptables di caricare il modulo relativo
- Possiamo fare match sullo stato della connessione correlata al pacchetto utilizzando il modulo di tracking delle connessioni: **conntrack**

```
$ iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED
```

- Questa regola dice se il pacchetto è parte di una connnessione stabilita (3-way handshake tcp completato, o nel caso di UDP è già “passato qualcosa recentemente”)

# Match stateful

- Tutti i match che abbiamo visto fino ad ora sono “integrati” in netfilter. Molti altri match e quelli aggiuntivi necessitano quasi sempre di “***-m nome_modulo_match***”, per dire a iptables di caricare il modulo relativo

- Possiamo fare match sullo stato della connessione correlata al pacchetto utilizzando il modulo di tracking delle connessioni: **conntrack**

```
$ iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED
```

- Questa regola dice se il pacchetto è parte di una connnessione stabilita (3-way handshake tcp completato, o nel caso di UDP è già “passato qualcosa recentemente”)

# REJECT or DROP?

Normalmente si preferisce non inviare nessun REJECT. Con il REJECT si spreca banda e si è leggermente vulnerabili ad attacchi DoS, oltre a rendere più facile il port scanning.

# Blacklisting o whitelisting

- “Vogliamo fare una festa. Facciamo la lista di tutti gli invitati o la lista di tutti i ’non invitati’ ? “ (cit.)
- Si fa sempre *whitelisting*. Se state facendo blacklisting state sbagliando qualcosa
- Solitamente si impostano le policy di default di INPUT e FORWARD a DROP, e poi si inseriscono una serie di regole per fare whitelisting del traffico legittimo. Di solito si lascia OUTPUT in ACCEPT, non ha molto senso bloccare anche quello che generiamo noi

# Blacklisting o whitelisting

- “Vogliamo fare una festa. Facciamo la lista di tutti gli invitati o la lista di tutti i ’non invitati’ ? “ (cit.)
- Si fa sempre *whitelisting*. Se state facendo blacklisting state sbagliando qualcosa
- Solitamente si impostano le policy di default di INPUT e FORWARD a DROP, e poi si inseriscono una serie di regole per fare whitelisting del traffico legittimo. Di solito si lascia OUTPUT in ACCEPT, non ha molto senso bloccare anche quello che generiamo noi

# Blacklisting o whitelisting

- “Vogliamo fare una festa. Facciamo la lista di tutti gli invitati o la lista di tutti i ’non invitati’ ? “ (cit.)
- Si fa sempre *whitelisting*. Se state facendo blacklisting state sbagliando qualcosa
- Solitamente si impostano le policy di default di INPUT e FORWARD a DROP, e poi si inseriscono una serie di regole per fare whitelisting del traffico legittimo. Di solito si lascia OUTPUT in ACCEPT, non ha molto senso bloccare anche quello che generiamo noi

# Interfaccia di loopback

- L'interfaccia di rete "lo" è quella associata all'indirizzo 127.0.0.1 e tutta la rete 127.0.0.0/8
- Molti programmi utilizzano questo indirizzamento per comunicazioni tra processi e altro
- **Non** dobbiamo bloccare questo traffico locale

```
$ iptables -A INPUT -i lo -j ACCEPT
```

# Interfaccia di loopback

- L'interfaccia di rete "lo" è quella associata all'indirizzo 127.0.0.1 e tutta la rete 127.0.0.0/8
- Molti programmi utilizzano questo indirizzamento per comunicazioni tra processi e altro
- **Non** dobbiamo bloccare questo traffico locale

```
$ iptables -A INPUT -i lo -j ACCEPT
```

# Interfaccia di loopback

- L'interfaccia di rete "lo" è quella associata all'indirizzo 127.0.0.1 e tutta la rete 127.0.0.0/8
- Molti programmi utilizzano questo indirizzamento per comunicazioni tra processi e altro
- **Non** dobbiamo bloccare questo traffico locale

```
$ iptables -A INPUT -i lo -j ACCEPT
```

# ICMP

- ICMP è un protocollo di servizio che si occupa di trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una rete
- Il tipo di pacchetto ICMP più famoso è senza dubbio la "echo request", che viene inviata con il comando ping e serve a sapere se un certo host è attivo (e viene utilizzato per controllare se arrivano i pacchetti o ci sono problemi sulla rete)
- A meno che non abbiate specifiche esigenze (ad esempio nascondere il fatto che il vostro host è acceso) **è considerata pratica scorretta oltre che "maleducazione" ignorare i messaggi ICMP**
- Se volete essere gentili semplicemente accettate tutto

```
$ iptables -A INPUT -p icmp -j ACCEPT
```

# ICMP

- ICMP è un protocollo di servizio che si occupa di trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una rete
- Il tipo di pacchetto ICMP più famoso è senza dubbio la "echo request", che viene inviata con il comando ping e serve a sapere se un certo host è attivo (e viene utilizzato per controllare se arrivano i pacchetti o ci sono problemi sulla rete)
- A meno che non abbiate specifiche esigenze (ad esempio nascondere il fatto che il vostro host è acceso) **è considerata pratica scorretta oltre che "maleducazione" ignorare i messaggi ICMP**
- Se volete essere gentili semplicemente accettate tutto

```
$ iptables -A INPUT -p icmp -j ACCEPT
```

# ICMP

- ICMP è un protocollo di servizio che si occupa di trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una rete
- Il tipo di pacchetto ICMP più famoso è senza dubbio la "echo request", che viene inviata con il comando ping e serve a sapere se un certo host è attivo (e viene utilizzato per controllare se arrivano i pacchetti o ci sono problemi sulla rete)
- A meno che non abbiate specifiche esigenze (ad esempio nascondere il fatto che il vostro host è acceso) **è considerata pratica scorretta oltre che "maleducazione" ignorare i messaggi ICMP**
- Se volete essere gentili semplicemente accettate tutto

```
$ iptables -A INPUT -p icmp -j ACCEPT
```

# ICMP

- ICMP è un protocollo di servizio che si occupa di trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una rete
- Il tipo di pacchetto ICMP più famoso è senza dubbio la "echo request", che viene inviata con il comando ping e serve a sapere se un certo host è attivo (e viene utilizzato per controllare se arrivano i pacchetti o ci sono problemi sulla rete)
- A meno che non abbiate specifiche esigenze (ad esempio nascondere il fatto che il vostro host è acceso) **è considerata pratica scorretta oltre che "maleducazione" ignorare i messaggi ICMP**
- Se volete essere gentili semplicemente accettate tutto

```
$ iptables -A INPUT -p icmp -j ACCEPT
```

# ICMP

Se volete evitare di accettare propro tutto il traffico, almeno accettate i tipi di messaggio ICMP fondamentali:

```
$ iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
$ iptables -A INPUT -p icmp --icmp-type 3 -j ACCEPT
$ iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
```

- Che corrispondono a
  - Echo request (type 8)
  - Destination Unreachable (type 3)
  - Echo reply (type 0)

# Chiudersi fuori di casa

- Fate molta attenzione se state configurando una macchina remota (ad esempio tramite ssh) quando impostate le policy di default!
- Caso tipico
  - "Questa macchina è completamente esposta all'internet...
  - beh... per prima cosa policy di default DROP!"

# Chiudersi fuori di casa

- Fate molta attenzione se state configurando una macchina remota (ad esempio tramite ssh) quando impostate le policy di default!
- Caso tipico
  - “Questa macchina è completamente esposta all’internet...
  - beh... per prima cosa policy di default DROP!”

# Chiudersi fuori di casa

- Fate molta attenzione se state configurando una macchina remota (ad esempio tramite ssh) quando impostate le policy di default!
- Caso tipico
  - "Questa macchina è completamente esposta all'internet...
  - beh... per prima cosa policy di default DROP!"

# Chiudersi fuori di casa

- Fate molta attenzione se state configurando una macchina remota (ad esempio tramite ssh) quando impostate le policy di default!
- Caso tipico
  - "Questa macchina è completamente esposta all'internet...
  - beh... per prima cosa policy di default DROP!"

# Chiudersi fuori di casa

Ricordatevi sempre di impostare le policy di default come ultimo passo, dopo esservi assicurati di aver messo una regola per accettare le connessioni nuove e stabilite per la vostra shell (ssh solitamente).

# Esempio di config essenziale

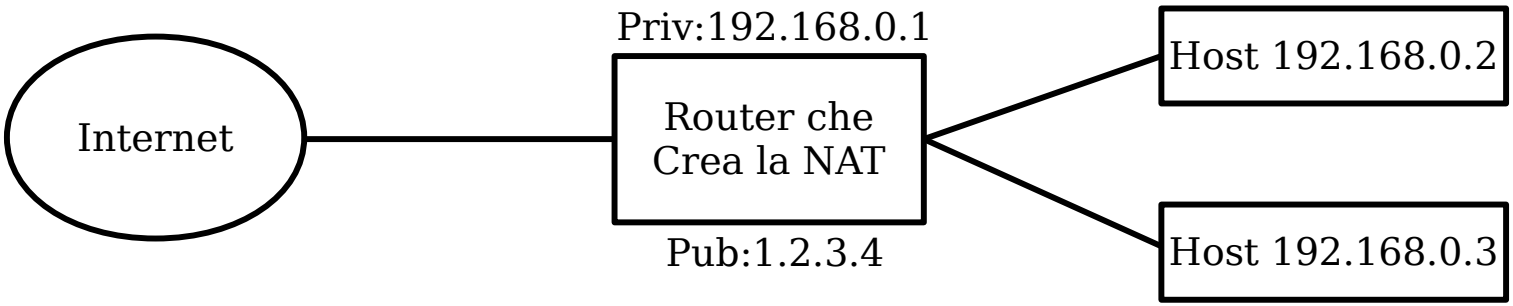
```
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 3 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -j ACCEPT
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
```

# esempio di config essenziale

```
root@Hello:~# iptables -nvL
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    6   452 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           ctstate RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0           icmptype 8
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0           icmptype 3
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0           icmptype 0
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22 ctstate NEW
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80 ctstate NEW

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 4 packets, 496 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

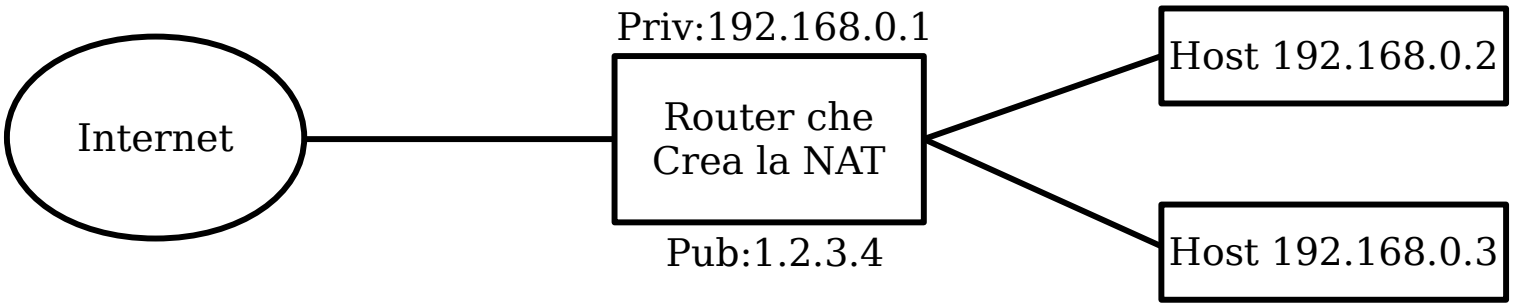
Figura : iptablesBasicConfig

# NAT - Network Address Translation



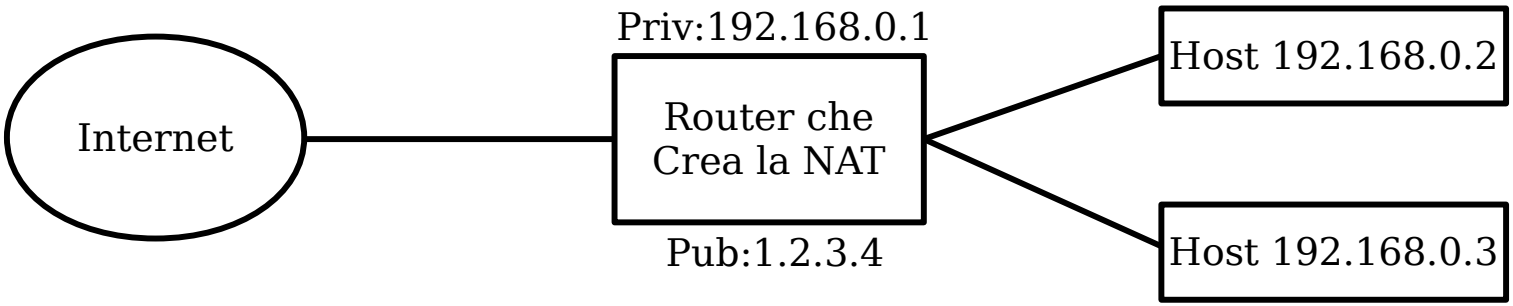

- Può servire per mascherare vari host sotto un unico IP
- Quando un host, vuole connettersi ad un server esterno, invia i pacchetti al router di frontiera (gateway) che prende l'indirizzo sorgente e lo sostituisce con il proprio (1.2.3.4)
- Il router sostituisce la porta TCP sorgente con una sua porta libera e si annota questo scambio in una tabella di connessioni stabilite
- Quando arrivano le risposte dal server il router consulta la tabella, capisce chi è il destinatario nella rete locale e sostiuisce indirizzo e porta di destinazione.

# NAT - Network Address Translation



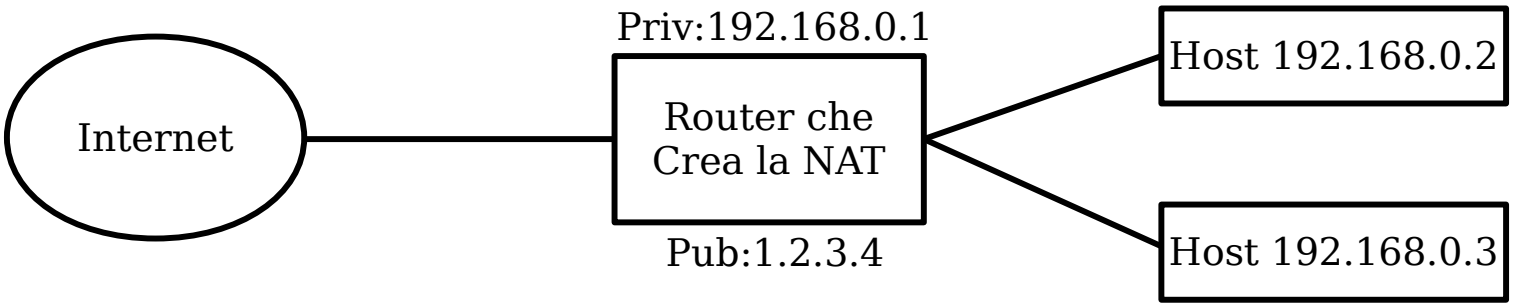

- Può servire per mascherare vari host sotto un unico IP
- Quando un host, vuole connettersi ad un server esterno, invia i pacchetti al router di frontiera (gateway) che prende l'indirizzo sorgente e lo sostituisce con il proprio (1.2.3.4)
- Il router sostituisce la porta TCP sorgente con una sua porta libera e si annota questo scambio in una tabella di connessioni stabilite
- Quando arrivano le risposte dal server il router consulta la tabella, capisce chi è il destinatario nella rete locale e sostiuisce indirizzo e porta di destinazione.

# NAT - Network Address Translation



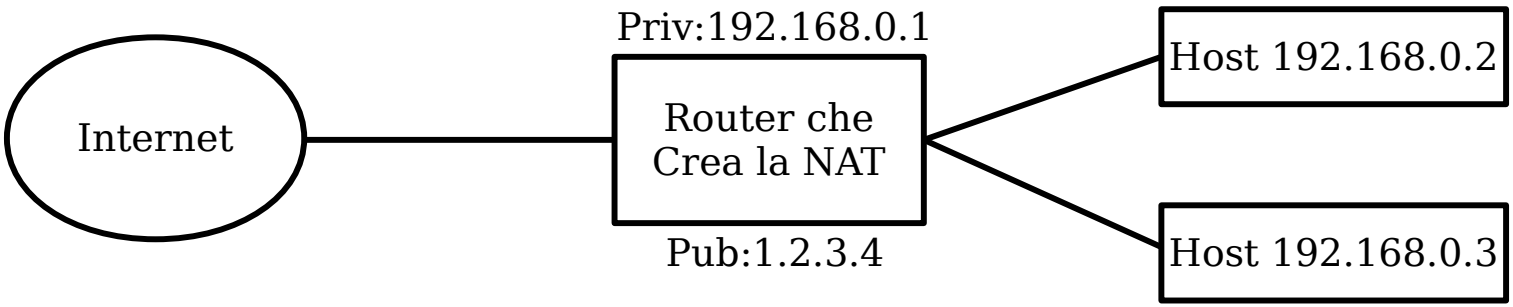

- Può servire per mascherare vari host sotto un unico IP
- Quando un host, vuole connettersi ad un server esterno, invia i pacchetti al router di frontiera (gateway) che prende l'indirizzo sorgente e lo sostituisce con il proprio (1.2.3.4)
- Il router sostituisce la porta TCP sorgente con una sua porta libera e si annota questo scambio in una tabella di connessioni stabilite
- Quando arrivano le risposte dal server il router consulta la tabella, capisce chi è il destinatario nella rete locale e sostiuisce indirizzo e porta di destinazione.

# NAT - Network Address Translation




- Può servire per mascherare vari host sotto un unico IP
- Quando un host, vuole connettersi ad un server esterno, invia i pacchetti al router di frontiera (gateway) che prende l'indirizzo sorgente e lo sostituisce con il proprio (1.2.3.4)
- Il router sostituisce la porta TCP sorgente con una sua porta libera e si annota questo scambio in una tabella di connessioni stabilite
- Quando arrivano le risposte dal server il router consulta la tabella, capisce chi è il destinatario nella rete locale e sostiuisce indirizzo e porta di destinazione.

# NAT

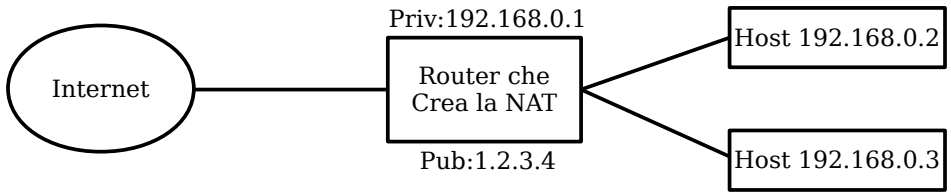



```
iptables -A FORWARD -s 192.168.0.0/24 -i eth1 -o eth0 -j ACCEPT

iptables -A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT

iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o eth0 -j SNAT --to-source 1.2.3.4


echo "1" > /proc/sys/net/ipv4/ip_forward
```

# NAT

- Se il router ha un indirizzo IP dinamico potete utilizzare **-j MASQUERADE** invece di SNAT, che identifica automaticamente l'indirizzo dell'interfaccia utilizzata per la route di default

- Come facciamo se vogliamo mettere su un servizio (ad. es. un server web) sull'host 192.168.0.3?

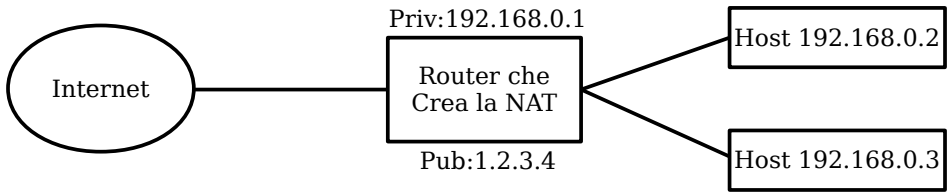



```
$ iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
$ iptables -t nat -A PREROUTING -d 1.2.3.4 -i eth0 -j DNAT --to-destination 19
```

# NAT

- Se il router ha un indirizzo IP dinamico potete utilizzare **-j MASQUERADE** invece di SNAT, che identifica automaticamente l'indirizzo dell'interfaccia utilizzata per la route di default

- Come facciamo se vogliamo mettere su un servizio (ad. es. un server web) sull'host 192.168.0.3?

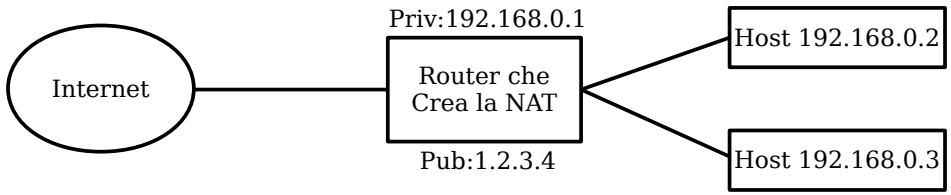



```
$ iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
$ iptables -t nat -A PREROUTING -d 1.2.3.4 -i eth0 -j DNAT --to-destination 19
```

# NAT

- Se il router ha un indirizzo IP dinamico potete utilizzare **-j MASQUERADE** invece di SNAT, che identifica automaticamente l'indirizzo dell'interfaccia utilizzata per la route di default

- Come facciamo se vogliamo mettere su un servizio (ad. es. un server web) sull'host 192.168.0.3?

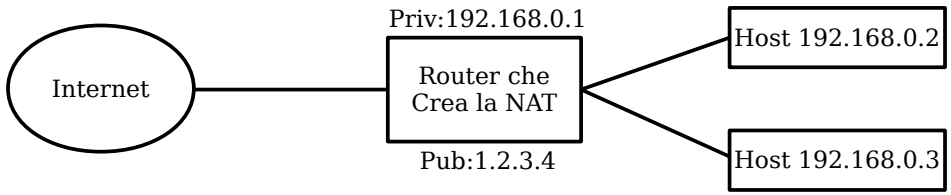



```
$ iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
$ iptables -t nat -A PREROUTING -d 1.2.3.4 -i eth0 -j DNAT --to-destination 19
```

# NAT

- Se il router ha un indirizzo IP dinamico potete utilizzare **-j MASQUERADE** invece di SNAT, che identifica automaticamente l'indirizzo dell'interfaccia utilizzata per la route di default

- Come facciamo se vogliamo mettere su un servizio (ad. es. un server web) sull'host 192.168.0.3?




```
$ iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
$ iptables -t nat -A PREROUTING -d 1.2.3.4 -i eth0 -j DNAT --to-destination 19
```

# Rendere le regole persistenti

- Problema: al riavvio il sistema operativo cancella tutte le regole e ripristina tutte le chain vuote
- *iptables-save* e *iptables-restore* fanno al caso nostro

```
iptables-save > /etc/firewall_rules
iptables-restore < /etc/firewall_rules
```

- Possiamo aggiungere in **/etc/rc.local** o altri script di startup a seconda della distro

# Rendere le regole persistenti

- Problema: al riavvio il sistema operativo cancella tutte le regole e ripristina tutte le chain vuote
- *iptables-save* e *iptables-restore* fanno al caso nostro

```
iptables-save > /etc/firewall_rules
iptables-restore < /etc/firewall_rules
```

- Possiamo aggiungere in **/etc/rc.local** o altri script di startup a seconda della distro

# NAT: un abominio

La NAT sarà tanto bella e comoda ma è un workaround (temporaneo fino ad IPv6). Quando è possibile non andrebbe usata. Specialmente bisognerebbe **evitare di fare doppie NAT** o altre cose simili.

- NAT viola il modello gerarchico di IP
- I processi su Internet non sono obbligati ad utilizzare TCP e UDP (vedi SCTP - RFC 2960)
- Il numero di connessioni contemporanee diminuisce a causa del numero limitato di numeri di porte
- NAPT trasforma Internet da una rete ad assenza di connessione
- La NAT **da sola** non è un meccanismo firewalling. Le caratteristiche **stateful** del firewall che implementa la NAT proteggono, non la NAT stessa

Approfondimenti su `http://tinyurl.com/nathorror`

# NAT: un abominio

La NAT sarà tanto bella e comoda ma è un workaround (temporaneo fino ad IPv6). Quando è possibile non andrebbe usata. Specialmente bisognerebbe **evitare di fare doppie NAT** o altre cose simili.

- NAT viola il modello gerarchico di IP
- I processi su Internet non sono obbligati ad utilizzare TCP e UDP (vedi SCTP - RFC 2960)
- Il numero di connessioni contemporanee diminuisce a causa del numero limitato di numeri di porte
- NAPT trasforma Internet da una rete ad assenza di connessione
- La NAT **da sola** non è un meccanismo firewalling. Le caratteristiche **stateful** del firewall che implementa la NAT proteggono, non la NAT stessa

Approfondimenti su `http://tinyurl.com/nathorror`

# NAT: un abominio

La NAT sarà tanto bella e comoda ma è un workaround (temporaneo fino ad IPv6). Quando è possibile non andrebbe usata. Specialmente bisognerebbe **evitare di fare doppie NAT** o altre cose simili.

- NAT viola il modello gerarchico di IP
- I processi su Internet non sono obbligati ad utilizzare TCP e UDP (vedi SCTP - RFC 2960)
- Il numero di connessioni contemporanee diminuisce a causa del numero limitato di numeri di porte
- NAPT trasforma Internet da una rete ad assenza di connessione
- La NAT **da sola** non è un meccanismo firewalling. Le caratteristiche **stateful** del firewall che implementa la NAT proteggono, non la NAT stessa

Approfondimenti su `http://tinyurl.com/nathorror`

# NAT: un abominio

La NAT sarà tanto bella e comoda ma è un workaround (temporaneo fino ad IPv6). Quando è possibile non andrebbe usata. Specialmente bisognerebbe **evitare di fare doppie NAT** o altre cose simili.

- NAT viola il modello gerarchico di IP
- I processi su Internet non sono obbligati ad utilizzare TCP e UDP (vedi SCTP - RFC 2960)
- Il numero di connessioni contemporanee diminuisce a causa del numero limitato di numeri di porte
- NAPT trasforma Internet da una rete ad assenza di connessione
- La NAT **da sola** non è un meccanismo firewalling. Le caratteristiche **stateful** del firewall che implementa la NAT proteggono, non la NAT stessa

Approfondimenti su `http://tinyurl.com/nathorror`

# NAT: un abominio

La NAT sarà tanto bella e comoda ma è un workaround (temporaneo fino ad IPv6). Quando è possibile non andrebbe usata. Specialmente bisognerebbe **evitare di fare doppie NAT** o altre cose simili.

- NAT viola il modello gerarchico di IP
- I processi su Internet non sono obbligati ad utilizzare TCP e UDP (vedi SCTP - RFC 2960)
- Il numero di connessioni contemporanee diminuisce a causa del numero limitato di numeri di porte
- NAPT trasforma Internet da una rete ad assenza di connessione
- La NAT **da sola** non è un meccanismo firewalling. Le caratteristiche **stateful** del firewall che implementa la NAT proteggono, non la NAT stessa

Approfondimenti su `http://tinyurl.com/nathorror`

# IPv6

- GNU/Linux ha il supporto per Ipv6 fin dai primi standard
  - Le assegnazioni di blocchi IPv4 da parte della IANA sono state esaurite nel 2011
  - In tutti i grandi datacenter internazionali IPv6 è già attivo e utilizzato
  - In netfilter ipv6 è già supportato
- Per chiarezza, il tool userspace per gestire le regole relative al traffico IPv6 è chiamato "ip6tables"
  - Funziona esattamente allo stesso modo di iptables normale.
  - La principale differenza è che non esiste la NAT

Alcune funzionalità di ICMPv6 sono obbligatorie (Neighbor Discovery Protocol, type 135 e 136) e se le bloccate avrete problemi di connessione

```
ip6tables -A INPUT -p ipv6-icmp -j ACCEPT
```

# IPv6

- GNU/Linux ha il supporto per Ipv6 fin dai primi standard
  - Le assegnazioni di blocchi IPv4 da parte della IANA sono state esaurite nel 2011
  - In tutti i grandi datacenter internazionali IPv6 è già attivo e utilizzato
  - In netfilter ipv6 è già supportato
- Per chiarezza, il tool userspace per gestire le regole relative al traffico IPv6 è chiamato "ip6tables"
  - Funziona esattamente allo stesso modo di iptables normale.
  - La principale differenza è che non esiste la NAT

Alcune funzionalità di ICMPv6 sono obbligatorie (Neighbor Discovery Protocol, type 135 e 136) e se le bloccate avrete problemi di connessione

```
ip6tables -A INPUT -p ipv6-icmp -j ACCEPT
```

# IPv6

- GNU/Linux ha il supporto per Ipv6 fin dai primi standard
  - Le assegnazioni di blocchi IPv4 da parte della IANA sono state esaurite nel 2011
  - In tutti i grandi datacenter internazionali IPv6 è già attivo e utilizzato
  - In netfilter ipv6 è già supportato
- Per chiarezza, il tool userspace per gestire le regole relative al traffico IPv6 è chiamato "ip6tables"
  - Funziona esattamente allo stesso modo di iptables normale.
  - La principale differenza è che non esiste la NAT

Alcune funzionalità di ICMPv6 sono obbligatorie (Neighbor Discovery Protocol, type 135 e 136) e se le bloccate avrete problemi di connessione

```
ip6tables -A INPUT -p ipv6-icmp -j ACCEPT
```

# IPv6

- GNU/Linux ha il supporto per Ipv6 fin dai primi standard
  - Le assegnazioni di blocchi IPv4 da parte della IANA sono state esaurite nel 2011
  - In tutti i grandi datacenter internazionali IPv6 è già attivo e utilizzato
  - In netfilter ipv6 è già supportato
- Per chiarezza, il tool userspace per gestire le regole relative al traffico IPv6 è chiamato “ip6tables”
  - Funziona esattamente allo stesso modo di iptables normale.
  - La principale differenza è che non esiste la NAT

Alcune funzionalità di ICMPv6 sono obbligatorie (Neighbor Discovery Protocol, type 135 e 136) e se le bloccate avrete problemi di connessione

```
ip6tables -A INPUT -p ipv6-icmp -j ACCEPT
```

# IPv6

- GNU/Linux ha il supporto per Ipv6 fin dai primi standard
  - Le assegnazioni di blocchi IPv4 da parte della IANA sono state esaurite nel 2011
  - In tutti i grandi datacenter internazionali IPv6 è già attivo e utilizzato
  - In netfilter ipv6 è già supportato
- Per chiarezza, il tool userspace per gestire le regole relative al traffico IPv6 è chiamato “ip6tables”
  - Funziona esattamente allo stesso modo di iptables normale.
  - La principale differenza è che non esiste la NAT

Alcune funzionalità di ICMPv6 sono obbligatorie (Neighbor Discovery Protocol, type 135 e 136) e se le bloccate avrete problemi di connessione

```
ip6tables -A INPUT -p ipv6-icmp -j ACCEPT
```

# IPv6

- GNU/Linux ha il supporto per Ipv6 fin dai primi standard
  - Le assegnazioni di blocchi IPv4 da parte della IANA sono state esaurite nel 2011
  - In tutti i grandi datacenter internazionali IPv6 è già attivo e utilizzato
  - In netfilter ipv6 è già supportato
- Per chiarezza, il tool userspace per gestire le regole relative al traffico IPv6 è chiamato “ip6tables”
  - Funziona esattamente allo stesso modo di iptables normale.
  - La principale differenza è che non esiste la NAT

**Alcune funzionalità di ICMPv6 sono obbligatorie (Neighbor Discovery Protocol, type 135 e 136) e se le bloccate avrete problemi di connessione**

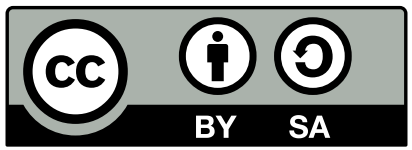
```
ip6tables -A INPUT -p ipv6-icmp -j ACCEPT
```

# Fonti

- Linux Administration Handbook
- Manpages
- https://wiki.gentoo.org/wiki/Handbook:AMD64/Installation/Networking
- http://www.penguintutor.com/linux/basic-network-reference
- Slides su firewalling su poul.org
  - http://www.poul.org/wp-content/uploads/2012/05/presentazione_netfilter.pdf
  - https://www.poul.org/wp-content/uploads/2014/04/Networking.pdf
  - https://www.poul.org/wp-content/uploads/2013/03/slides.pdf

# Fine

Grazie per l'attenzione!






`http://www.poul.org`