# Network Labs

***Release latest***

**23 feb 2022**

# Tools

**Nota:** Le esperienze di laboratorio presentate qui realizzano il laboratorio del corso di Informatica del quinto anno delle Scienze Applicate. Le esperienze presentate sono tutte incentrate sulle reti e sulla suite TCP/IP.

La comprensione di ognuna di queste implica per forza aver studiato prima la teoria corrispondente. Per questo motivo, all'inizio di ogni esperienza sono indicati gli argomenti su cui essa si basa e gli strumenti necessari alla sua realizzazione.

Buon lavoro!

**Suggerimento:** Questa documentazione, oltre che usufruibile online, è disponibile in diversi formati: **PDF, ePUB, SingleHTML, etc.** Per scaricare il formato che preferite, accedete alla apposita finestra di selezione posizionata in basso nella barra laterale.

Se ne scaricate una copia, abbiate cura di verificare che essa sia sempre aggiornata, confrontando la sua data di rilascio con quella del sito, scritta in ogni pagina in basso (Version: AnnoMeseGiorno).

Il primo capitolo è dedicato all'installazione e alla comprensione di alcuni strumenti che utilizzeremo nel corso delle lezioni. Gli altri sono suddivisi per macroargomenti sulle esperienze di laboratorio che ho progettato.

CAPITOLO 1

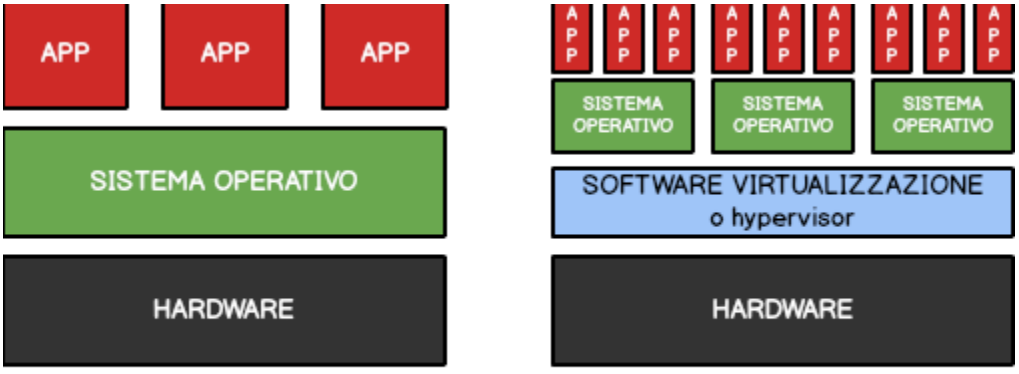
# VirtualBox

## 1.1 Macchine virtuali

Senza pietà, rivedo e correggo la prima frase di Wikipedia sull'argomento.

In informatica il termine **macchina virtuale (VM)** indica un software che, attraverso un processo di virtualizzazione, crea un ambiente virtuale che emula tipicamente il comportamento di una macchina fisica grazie all'assegnazione di risorse hardware (porzioni di disco rigido, RAM e risorse di processamento) ed in cui le applicazioni possono essere eseguite come se interagissero con tale macchina.

In pratica stiamo parlando di un software in grado di emulare diversi tipi di hardware, per realizzare tutto il necessario per eseguire *virtualmente* un altro sistema operativo, da cui il termine *Macchina Virtuale*.




Architettura tradizionale

Architettura virtualizzata

Il sistema operativo fisico su cui viene installato l'emulatore si definisce **Sistema Host**. Il sistema operativo che si installa sull'hardware virtuale viene definito **Sistema Guest**. A parte il nome e le prestazioni in presenza (chiaramente i sistemi fisici sono più veloci) a livello di rete o a livello remoto sistemi operativi fisici e virtuali sono spesso indistinguibili e assolutamente intercambiabili.

**Avvertimento:** Virtualizzare una intera macchina fisica non è un gioco da ragazzi per un hardware! Questa capacità risiede in una opzione del vostro processore che va assolutamente abilitata tramite BIOS.

Per processori Intel questa caratteristica si chiama **Intel Virtualization Technology (Intel VT)**. Per processori AMD si chiama **AMD Virtualization (AMD V)**.

Se siete fortunati sono già abilitate nel vostro hardware. Altrimenti bisogna accedere al BIOS e provvedere da sè.

## 1.2 VirtualBox

Fra le varie soluzioni software per la virtualizzazione, la mia selezione didattica considera sempre e solo le soluzioni gratuite, opensource e multipiattaforma. Ecco perché la mia scelta ricade sull'utilizzo di **Oracle VM VirtualBox** (https://www.virtualbox.org). Questo software, oltre ad essere rilasciato con una licenza libera ed essere disponibile su Windows, Mac, Linux ha anche altre 2 belle caratteristiche dalla sua parte: è probabilmente il software di virtualizzazione più semplice da utilizzare in assoluto e presenta una eccellente interfaccia grafica, costruita con le librerie Qt.

Per l'installazione e la configurazione (semplicissime) dell'emulatore, aprite il sito web indicato sopra e seguite le istruzioni riportate sul suo manuale online: https://www.virtualbox.org/manual/ch01.html.

**Buon lavoro!**

CAPITOLO 2

# Ubuntu Server

**Nota:** Prerequisti: **VirtualBox.**

In questo capitolo andremo ad installare una copia di Ubuntu (una sua configurazione chiamata **Ubuntu Server**) su VirtualBox. Chiaramente dovete aver già provveduto ad installare VirtualBox sulla vostra macchina. Poi si può procedere all'installazione di Ubuntu Server.

Ubuntu Server non è altro che una versione di Ubuntu (https://www.ubuntu.com) in cui sono state tolte le maggiori funzionalità grafiche e sono proposte durante l'installazione stessa del sistema operativo opzioni riguardanti l'installazione di software comuni per le reti quali un server HTTP, un server DNS, un server DHCP, un server SSH, etc...

## 2.1 Installazione

Per l'installazione di Ubuntu su VirtualBox dovete creare una macchina virtuale con VirtualBox, caricare la ISO di Ubuntu come se fosse un CD e poi avviare l'installazione.

Se avete bisogno di aiuto potete seguire la guida ufficiale di Ubuntu: https://ubuntu.com/tutorials/install-ubuntu-server#1-overview

Buon lavoro!

Al termine dell'installazione potete approfondire i seguenti prossimi argomenti.

## 2.2 Il terminale Linux

Adesso vogliamo dedicare un pò di tempo a prendere confidenza con il terminale Linux. Utilizzare l'interfaccia testuale può essere molto vantaggioso in diversi casi:

- Tutti i sistemi Linux hanno la stessa interfaccia testuale, ma le interfacce grafiche sono potenzialmente tutte diverse

- La connessione remota ad un dispositivo in modalità testuale è veloce, sicura e facile da stabilire
- L'interfaccia testuale è molto potente. Pensate all'interfaccia grafica del vostro Sistema Operativo preferito:
  - Come si fa a controllare l'IP della macchina?
  - Come si fa a cercare un file all'interno di tutto il computer?
  - Come si fa a disintallare un programma? Arrestare un servizio?

Tutte queste operazioni costano un unico comando, una riga di codice con l'interfaccia testuale. E richiedono un secondo o poco più per l'esecuzione.

Adesso che ho attirato la vostra attenzione sull'utilizzo della linea di testo, vediamo alcuni semplici comandi organizzati per utilizzo:

**Muoversi fra i file**

| Comando | Descrizione |
|---|---|
| ls | Elenca i file nella directory corrente (list) |
| cd | Cambia Directory. |
| pwd | Directory corrente |

**Manipolazione del testo**

| Comando | Descrizione |
|---|---|
| cat | Concatena i file e ne manda il contenuto nello standard output |
| less | Visualizza il contenuto di un file |
| nano | Editor testuale |

**Gestione di file e directory**

| Comando | Descrizione |
|---|---|
| mkdir | Crea una directory, una cartella |
| touch | Crea un file |
| cp | Copia un file o una directory |
| mv | Sposta un file o una directory |
| rm | Rimuove un file o una directory |

**Sistema**

| Comando | Descrizione |
|---|---|
| shutdown | Inizia la procedura di spegnimento |
| reboot | Riavvia il sistema |

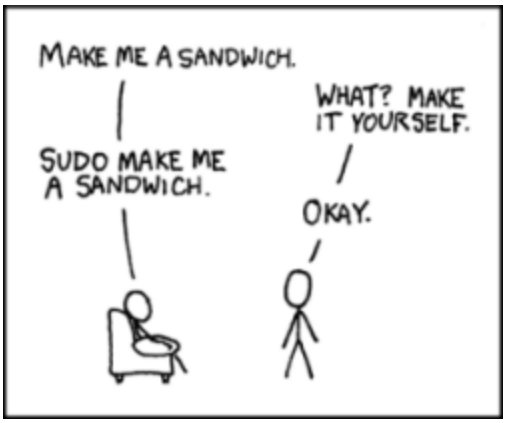
**Utilities**

| Comando | Descrizione |
|---|---|
| history | Elenca la cronologia dei comandi digitati |
| man | Apre il manuale richiesto |

---

**Nota:** Il comando **sudo** permette di eseguire qualsiasi comando con privilegi amministrativi.

Basta precedere *sudo* a qualsiasi comando per fare come se fosse l'amministratore del sistema ad eseguirlo.




## 2.3 Gestione software

L" **Advanced Packaging Tool** (conosciuto con l'acronimo APT) è il gestore standard di pacchetti software della distribuzione Debian e di tutte le sue derivate. In particolare vale la pena di ricordare Ubuntu e Raspberry come derivate di punta.

Questo sistema di gestione dei pacchetti è in grado di cercare, scaricare, installare qualsiasi software disponibile nei repository indicati nei file di configurazione per renderli disponibile all'istante!

**Avvertimento:** Poiché il comando APT si occupa di operazioni amministrative, deve essere sempre preceduto dal comando sudo.

Vediamo via via le opzioni di APT più importanti:

```
$ sudo apt update
```

Aggiorna l'elenco del software presente nel repository. In questo modo APT saprà qual è l'ultima versione del software disponibile online.

```
$ sudo apt upgrade
```

Sincronizza il software di sistema con quello presente nel repository. Praticamente permette di aggiornare tutto il software all'ultima versione disponibile.

```
$ sudo apt search package
```

Cerca il termine «package» fra i pacchetti software disponibili nel repository. Funziona anche senza sudo.

```
$ sudo apt install package
```

Scarica «package» e lo installa nel sistema, rendendolo disponibile all'utente.

```
$ sudo apt remove package
```

Rimuove «package» dal sistema.

CAPITOLO 3

# Collegamento da remoto

**Nota:** Prerequisti: **OS, prompt, terminale**

Argomenti trattati: **RDP, VNC, SSH**

Un sistema operativo, virtuale o fisico, installato nel dispositivo davanti a noi o a milioni di chilometri di distanza può essere comandato semplicemente con un collegamento **da remoto!**

I metodi più utilizzati per la connessione remota ad un dispositivo dipendono dal sistema operativo dello stesso, oltre che da quello del computer vicino a noi, grazie al quale ci vogliamo connettere al computer remoto. In questo elenco troviamo i protocolli più utilizzati per la connessione da remote con indicati i software server e client per determinati OS. Se volete conoscere atri software dedicati a combinazioni di sistemi operativi diversi (ad esempio connettersi ad un PC con Windows 10 da un Mac), documentatevi su Internet sui client per la corrispondente tecnologia.

| Protocollo | Tipologia | Server (su RPI) | Client (su Win10) |
|---|---|---|---|
| RDP | Grafica | Xrdp | Remote Desktop |
| VNC | Grafica | vnc | VNC Viewer |
| SSH | Testuale | sshd | Putty |

**Avvertimento:** Qualsiasi metodo sceglierai, ricordati che avrai bisogno di conoscere il **nome** e/o l'**indirizzo IP** del dispositivo remoto!

Cerca di capire **prima** come sia possibile ottenere (e magari modificare) queste informazioni!

## 3.1 RDP

**Remote Desktop Protocol** è un protocollo di rete proprietario sviluppato da Microsoft, che permette la connessione remota da un computer a un altro in maniera grafica. Il protocollo di default utilizza la porta TCP e UDP 3389.

I client RDP esistono per la maggior parte delle versioni di Microsoft Windows, Linux, Unix, macOS, Android, iOS. I server RDP ufficiali esistono per i sistemi operativi Windows nonostante ne esistano anche per i sistemi Unix-Like.

**Suggerimento: Su Ubuntu**

Installa il servizio xrdp:

```
$ sudo apt install xrdp
```

Fatto questo, riavvia.

**Suggerimento: Su Windows**

Non devi fare nulla! Ti basta cercare il software *Connessione a Desktop Remoto*

## 3.2 VNC

**Virtual Network Computing** è un protocollo per applicazioni software di controllo remoto, utilizzato per amministrare il proprio computer a distanza. Può essere utilizzato anche per controllare in remoto server che non posseggono né monitor né tastiera.

Il protocollo di comunicazione usato a livello di trasporto è il TCP sulla porta di default 5900, oppure tramite interfaccia HTTP sulla porta 5800/tcp.

**Suggerimento: Su Ubuntu**

Il server VNC si chiama vnc. Installa

Fatto questo, riavvia.

**Suggerimento: Su Windows**

Un client VNC gratuito è il VNC Viewer di RealVNC: https://www.realvnc.com/en/connect/download/viewer/windows/

Scaricalo, installalo su Windows e provalo.

## 3.3 SSH

**Secure Shell** è un protocollo che permette di stabilire una sessione remota cifrata tramite interfaccia a riga di comando con un altro host di una rete informatica. È il protocollo che ha sostituito l'analogo, ma insicuro, Telnet, perché basato su una comunicazione **non** cifrata.

A livello server utilizza la porta 22, sia tramite TCP che UDP.

**Suggerimento: Su Ubuntu**

Il server SSH è disponibile di default, ma va abilitato (come???)

Fatto questo, riavvia.

**Suggerimento: Su Windows**

Ti basta scaricare Putty e usarlo senza neanche installarlo!

Il sito ufficiale è: https://www.putty.org/

# CAPITOLO 4

# ipconfig

**Nota:** Prerequisti: **Windows: command prompt**

Argomenti trattati: **Indirizzamento IP**

Ipconfig è il comando basilare fra i vari tool di rete presenti sui sistemi Microsoft, perché è quello che permette di verificare la configurazione delle interfacce di rete presenti nel dispositivo.

E" utilizzabile dal prompt dei comandi per due ordini di motivi: il primo di essi è la *diagnostica* ovvero il controllo delle configurazioni di rete attualmente impostate (IP, subnet, gateway, DNS, MAC, TTL DHCP, etc. . . ). Il secondo di essi è il *reset* delle impostazioni, la invocazione del client DHCP, la pulizia della cache DNS, etc. . .

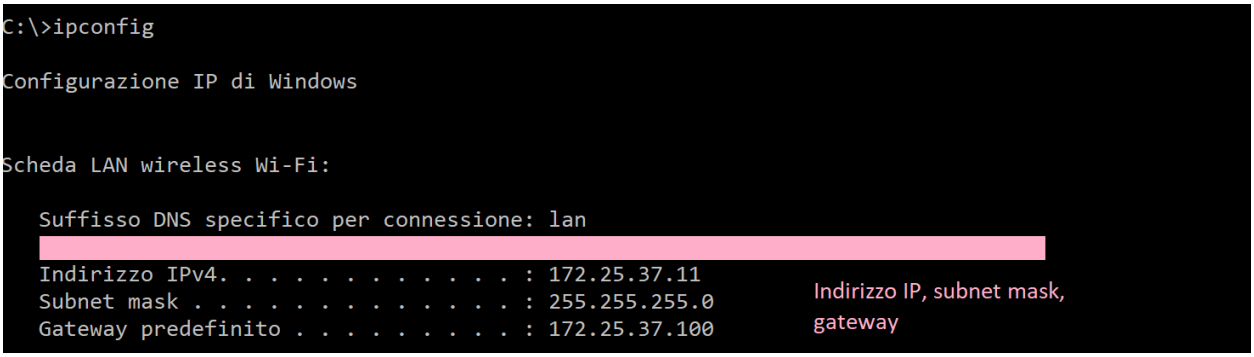
Vediamo la sintassi generale:

```
$ ipconfig [opzioni] [interfaccia]
```

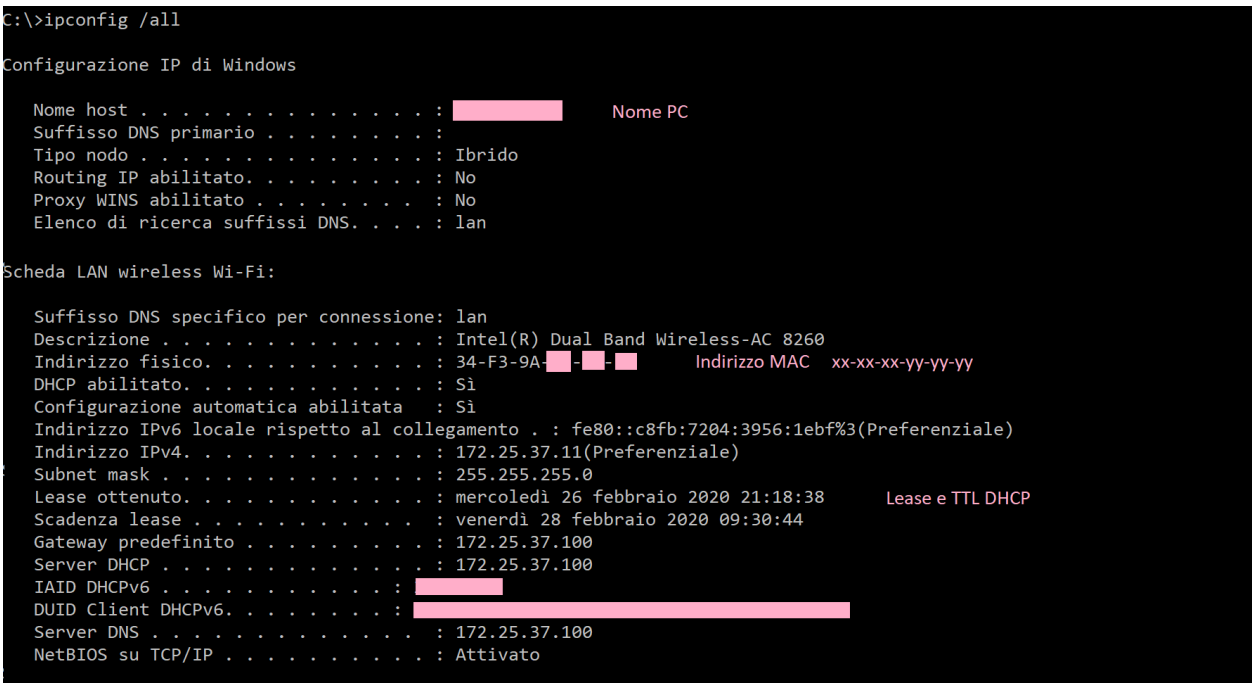
E le opzioni degne di nota:

| Opzione | Significato |
|---|---|
| /all | Visualizza le informazioni complete su tutte le NIC installate nel sistema |
| /release | Rilascia la configurazione di rete della NIC specificata |
| /renew | Rinnova la configurazione di rete della NIC specificata |
| /flushdns | Ripulisce la cache DNS |
| /registerdns | Aggiorna i lease DHCP e registra i nomi DNS |

Vediamo un esecuzione del comando ipconfig senza alcuna opzione.




Come si vede, tramite questo comando è possibile visualizzare le informazioni di base di configurazione della rete. Per accedere a informazioni più dettagliate dobbiamo utilizzare l'opzione */all*:




Le altre 4 opzioni vanno a coppia e per l'esecuzione richiedono privilegi amministrativi.

*release* e *renew* servono rispettivamente per cancellare le impostazioni di rete ottenute dal server DHCP e per richiedere allo stesso di inviarne di nuove. Può essere utile in alcune situazioni se la connettività è limitata perché il pacchetto di risposta DHCP è arrivato incompleto o corrotto.

```
$ ipconfig /release

... attendi qualche secondo...

$ ipconfig /renew
```

La seconda coppia di opzioni può essere utile quando qualche dispositivo di rete ha cambiato nome e si ha necessità di riaggiornare le informazioni senza aspettare la scadenza naturale delle stesse tramite il TTL.

```
$ ipconfig /flushdns

... attendi qualche secondo...
```

(continues on next page)

(continua dalla pagina precedente)

```
$ ipconfig /registerdns
```

Questo è tutto. Adesso siete dei mezzi draghi del comando *ipconfig* ;)

# CAPITOLO 5

## ifconfig, ip

**Nota:** Prerequisti: **Linux, MacOS: terminale**

Argomenti trattati: **Indirizzamento IP**

Nell'esperienza **ipconfig** abbiamo visto come è possibile reperire informazioni sulla configurazione di rete in un terminale Windows.

In questa esperienza faremo lo stesso nel caso dei terminali Linux e Mac. Ovviamente tutte le considerazioni qui fatte funzionano anche su Raspberry.

Nella mia disamina ho deciso di spiegare le cose 2 volte, prima con il comando **ifconfig** e poi con il comando **ip**. La scelta di raddoppiare il lavoro dipende da una serie di considerazioni che vado ad elencare:

- *ifconfig* è un utility più limitata rispetto alla più moderna *ip*
- Non sono sicuro che il comando *ip* sia disponibile su Mac (probabilmente sì...)
- In molte distribuzioni Linux potreste non trovare il (vecchio) comando *ifconfig*
- *ifconfig* è più vecchia e limitata ma più semplice
- *ip* è in grado di gestire praticamente tutte le configurazioni del livello di rete e inferiore, ma questo implica un diverso grado di complessità

Mettendo insieme tutti questi *statements* ho deciso di introdurre entrambi i tools. Vediamoli.

Nelle due trattazioni che seguono cercherò di dedurre la configurazione di rete di un generico dispositivo che ha una scheda di rete con cavo, chiamata **eth0** e una scheda di rete wifi chiamata **wlan0**. Potrebbe essere un portatile, oppure un Raspberry, etc... Indico qua le informazioni in modo tale che sia più semplice capire il funzionamento dei tool di rete, conoscendo in anticipo l'output che cerchiamo.

| Descrizione | Valore |
|---|---|
| Rete | 10.0.0.0 |
| Subnet mask | 255.255.255.0 |
| broadcast | 10.255.255.255 |
| Gateway | 10.0.0.1 |
| Server DNS | 10.0.0.2 |
| Server DHCP | 10.0.0.3 |
| IP scheda con cavo | 10.0.0.51 |
| IP scheda wifi | 10.0.0.52 |

## 5.1 ifconfig

Visualizzare informazioni sulle interfacce di rete

```
$ ifconfig

eth0:   flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.51  netmask 255.0.0.0  broadcast 10.255.255.255
        ether 54:53:ed:XX:XX:XX  txqueuelen 1000  (Ethernet)
        RX packets 95414  bytes 130702336 (124.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 43485  bytes 4338669 (4.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 240 (240.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0:  flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.52  netmask 255.0.0.0  broadcast 10.255.255.255
        ether b8:76:3f:YY:YY:YY  txqueuelen 1000  (Ethernet)
        RX packets 7035  bytes 1837515 (1.7 MiB)
        RX errors 0  dropped 1496  overruns 0  frame 0
        TX packets 377  bytes 33009 (32.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Imopostare un indirizzo (o cancellarlo) per una interfaccia di rete

```
$ sudo ifconfig eth0 add 192.168.0.51
$ sudo ifconfig eth0 del 192.168.0.51
```

Abilitare o disabilitare una interfaccia di rete

```
$ sudo ifconfig wlan0 up
$ sudo ifconfig wlan0 down
```

Visualizzare la tabella di routing (verificando il gateway predefinito). Con *ifconfig* non si può fare. In questo caso bisogna ricorrere al comando **route**:

```
$ route -n4

Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.0.1        0.0.0.0         UG    100    0        0 eth0
0.0.0.0         10.0.0.1        0.0.0.0         UG    600    0        0 wlan0
10.0.0.0        0.0.0.0         255.0.0.0       U     100    0        0 eth0
10.0.0.0        0.0.0.0         255.0.0.0       U     600    0        0 wlan0
```

Impostare il default gateway (oppure in caso di routing più complessi, aggiungere una route). Oppure rimuoverlo. Ancora tramite l'utility *route*:

```
$ sudo route add default gw 192.168.0.1 eth0
$ sudo route del default gw 192.168.0.1 eth0
```

Visualizzare i server DNS in uso. Ancora una volta **non** si usano opzioni del comando *ifconfig* ma si può controllare direttamente sul file ove sono scritti:

```
$ cat /etc/resolv.conf

nameserver 10.0.0.2
```

## 5.2 ip

Visualizzare informazioni sulle interfacce di rete

```
$ ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default␣
↪qlen 1000
    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
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group␣
↪default qlen 1000
    link/ether 54:53:ed:XX:XX:XX brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.51/8 brd 10.255.255.255 scope global dynamic noprefixroute eth0
    valid_lft 80942sec preferred_lft 80942sec
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group␣
↪default qlen 1000
    link/ether b8:76:3f:YY:YY:YY brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.52/8 brd 10.255.255.255 scope global dynamic noprefixroute wlan0
    valid_lft 80939sec preferred_lft 80939sec
```

Imopostare un indirizzo (o cancellarlo) per una interfaccia di rete

```
$ sudo ip a add 192.168.0.51 dev eth0
$ sudo ip a del 192.168.0.51 dev eth0
```

Abilitare o disabilitare una interfaccia di rete

```
$ sudo ip link set wlan0 up
$ sudo ip link set wlan0 down
```

Visualizzare la tabella di routing (verificando il gateway predefinito).

```
$ ip r

default via 10.0.0.1 dev eth0 proto dhcp metric 100
default via 10.0.0.1 dev wlan0 proto dhcp metric 600
10.0.0.0/8 dev eth0 proto kernel scope link src 10.0.0.51 metric 100
10.0.0.0/8 dev wlan0 proto kernel scope link src 10.0.0.52 metric 600
```

Impostare il default gateway (oppure in caso di routing più complessi, aggiungere una route). Oppure rimuoverlo.

```
$ sudo ip r add 192.168.0.0/24 via 192.168.0.1 dev eth0
$ sudo ip r del 192.168.0.0/24
```

Visualizzare i server DNS in uso. Qui anche l'utility *ip* non può arrivare perché il sistema di risoluzione degli indirizzi è diverso in ambito UNIX (Linux/MAC) rispetto al corrispettivo Windows. Il consiglio è ancora una volta di controllare il file di configurazione:

```
$ cat /etc/resolv.conf

nameserver 10.0.0.2
```

CAPITOLO 6

# Networking

**Nota:** Prerequisti: **Linux, MacOS. Terminale**

Argomenti trattati: **Indirizzamento IP**

In questo capitolo vado a riassumere alcuni concetti in maniera veloce su come verificare e/o impostare la configurazione di rete in un ambiente UNIX-like (Linux, MacOS) da terminale.

## 6.1 Configurazione di rete

Ricordate che per la configurazione di rete di un dispositivo, ovvero per permettergli di accedere alla rete Internet, occorre che esso conosca:

- **il suo Indirizzo IP e la sua subnet mask**. Questo gli permette di conoscere la sua rete, l'indirizzo della sua rete di appartenenza, il suo indirizzo di broadcast e di riconoscere quali dispositivi appartengono alla sua stessa rete. Un dispositivo con queste informazioni può comunicare 1 a 1 con un altro dispositivo della propria rete o fare broadcast e dunque utilizzare protocolli come ad esempio ARP.
- **il suo gateway**. L'indirizzo del dispositivo della propria rete di appartenenza che gli consente di collegarsi all'esterno di essa. Con questa informazione, più le precedenti, un dispositivo può connettersi a qualunque dispositivo remoto raggiungibile tramite la rete Internet.
- **Almeno un IP relativo ad un dispositivo che esponde un servizio DNS**, solitamente ne vengono forniti due per motivi di tolleranza agli errori, dato che DNS utilizza il protocollo UDP.

Queste informazioni possono essere fornite in due modi al dispositivo:

1. tramite **configurazione statica**, ovvero inserendo manualmente le informazioni necessarie.
2. tramite **configurazione dinamica**, ovvero ricorrendo all'utilizzo del protocollo DHCP.

## 6.2 Configurazione di rete attuale

Con le seguenti istruzioni saremo in grado di visualizzare la configurazione di rete attuale di un dispositivo UNIX-like tramite terminale.

Per visualizzare la configurazione attuale:

```
$ ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default␣
↪qlen 1000
    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
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group␣
↪default qlen 1000
    link/ether 54:05:db:XX:XX:XX brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.7/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
       valid_lft 84568sec preferred_lft 84568sec
```

Nella seguente configurazione vediamo la presenza di 2 interfacce:

1. l'interfaccia chiamata **lo**, ovvero l'interfaccia di loopback, avente indirizzo **127.0.0.1**
2. l'interfaccia chiamata **eth0**, avente indirizzo (inet) **192.168.1.7** e subnet mask **255.255.255.0** (/24)

Ovviamente quella che ci interessa è la seconda. Per visualizzare la tabella di routing e determinare il gateway facciamo:

```
$ ip r

default via 192.168.1.1 dev eth0 proto dhcp metric 100
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.7 metric 100
```

Chiaramente (almeno spero) il gateway è l'indirizzo IP **192.168.1.1**. Ultima informazione da desumere sono gli indirizzi dei DNS. Sono scritti nel file */etc/resolv.conf*:

```
$ cat /etc/resolv.conf

nameserver 1.1.1.1
nameserver 8.8.8.8
```

Ecco qua le informazioni necessarie :D

## 6.3 Configurazione di rete statica (temporanea)

Con le seguenti istruzioni saremo in grado di impostare una configurazione statica temporanea, ovvero che sarà resettata dopo il prossimo riavvio.

Facciamo l'ipotesi di voler configurare l'interfaccia di rete chiamata **eth0**, con le seguenti informazioni di rete:

| | |
|---|---|
| IP | 192.168.0.10 |
| SUBNET MASK | 255.255.255.0 |
| GW | 192.168.0.1 |
| DNS | 1.1.1.1, 9.9.9.9 |

Vediamo i passaggi commentati uno per uno:

```
# questo è un commento
questo è un comando

# TUTTI I COMANDI QUI SOTTO VANNO ESEGUITI CON PRIVILEGI AMMINISTRATIVI
# QUINDI SONO PRECEDUTI DALL'ISTRUZIONE sudo

# attiva l'interfaccia eth0
sudo ip link set eth0 up

# imposta IP e SUBNET MASK
sudo ip a add 192.168.0.10/24 dev eth0

# imposta il GW
sudo ip r add default via 192.168.0.1
```

Al termine di questi comandi abbiamo impostato IP, SUBNET MASK e GATEWAY. Se casualmente ti capita di combinare qualcosa tale per cui vuoi resettare la configurazione e ricominciare daccapo:

```
# resetta la configurazione
sudo ip a flush eth0

# spegne l'interfaccia
sudo ip link set eth0 down
```

Per inserire il DNS basta aprire il file */etc/resolv.conf*, cancellare il suo contenuto e scriverci dentro *nameserver 1.1.1.1* e a capo *nameserver 9.9.9.9*. Salvare e chiudere.

```
sudo nano /etc/resolv.conf

# cancella tutto il contenuto, scrivici solo
nameserver 1.1.1.1
nameserver 9.9.9.9

# salva e chiudi
```

Finito!!!

## 6.4 Configurazione di rete statica (netplan)

Questa configurazione che vediamo qui funziona solo su **Ubuntu Server**, che utilizza uno strumento di configurazione chiamato **netplan**.

La configurazione va scritta in un file *yaml* da creare nella cartella */etc/netplan* eventualmente eliminando ogni ulteriore configurazione.

```
# Andiamo nella cartella /etc/netplan
cd /etc/netplan

# visualizziamo il suo contenuto
ls

00_default.yaml
```

Come vedete c'è un solo file attualmente chiamato **00_default.yaml**. Va rinominato per disattivarlo

```
# rinomino il file precedente
sudo mv 00_default.yaml 00_default.old
```

Adesso creo il nuovo file *99_config.yaml* con le seguenti istruzioni:

```
sudo nano 99_config.yaml

# adesso scriveteci dentro ESATTAMENTE questo (attenti anche all'indentazione)

network:
    version: 2
    renderer: networkd
    ethernets:
        eth0:
            addresses:
                192.168.0.10/24
            gateway4: 192.168.0.1
            nameservers:
                addresses:  [ 1.1.1.1 , 9.9.9.9 ]

# poi salva e chiudi
```

Adesso per applicare le impostazioni:

```
sudo netplan apply
```

## 6.5 Configurazione di rete dinamica (netplan)

Se per qualche motivo il vostro server non necessita un IP statico potete usufruire della comodità del protocollo DHCP con pochi passi.

Come prima, dovete eliminare ogni file dalla cartella */etc/netplan*, ricrearne uno con la configurazione corretta e riapplicare le modifiche. Le cose da scrivere nel file 99_dynamic.yaml sono:

```
network:
    version: 2
    renderer: networkd
    ethernets:
        eth0:
            dhcp4: true
```

Salvate e applicate ancora con:

```
sudo netplan apply
```

## 6.6 Bridge

I bridge sono periferiche virtuali di livello 2 (il livello inferiore, relativo alla distinzione dei dispositivi tramite indirizzo MAC) che non può ricevere o trasmettere nulla finché non viene collegato ad una o più periferiche reali.

La sua utilità sta nel fatto che può collegare a livello inferiore 2 periferiche senza intaccare in alcun modo il livello di rete soprastante.

- Se entrambe le due periferiche sono reali, il bridge serve per la conversione a livello fisico dei pacchetti SENZA un instradamento (che richiederebbe una nuova conversione in digitale).
- Se almeno una delle due periferiche è virtuale (ad esempio con una macchina virtuale) il bridge può essere utilizzato per connettere fisicamente la periferica virtuale alla rete su cui è esposta la periferica reale, permettendo alla macchina virtuale di essere esposta sulla rete a cui la periferica fisica appartiene

Per creare un bridge:

```
# nell'esempio che segue:
# il bridge si chiamerà bridge0
# la periferica fisica si chiamerà phisdev0
# la periferica virtuale si chiamerà virtdev0

# creare il bridge e attivarlo
sudo ip link add name bridge0 type bridge
sudo ip link set bridge0 up

# aggiungere le due periferiche al bridge
sudo ip link set phisdev0 master bridge0
sudo ip link set virtdev0 master bridge0
```

## 6.7 TUN/TAP devices

Le interfacce TUN/TAP sono 2 tipologie di interfacce virtuali inventate per permettere operazioni di rete virtuali. A metà tra l'informatica e la magia nera, le 2 interfacce hanno scopi specializzati:

- Le periferiche **TUN** servono per connettere **a livello di rete** 2 software allo stesso modo in cui una interfaccia di rete REALE connette 2 dispositivi. Questa cosa fa pensare ai 2 software (che stanno in realtà nello stesso dispositivo) di trovarsi in 2 dispositivi diversi.
- Le periferiche **TAP** servono per connettere **a livello inferiore** 2 software allo stesso modo in cui una interfaccia fisica connette 2 dispositivi. In questo modo la periferica realizza un bridge virtuale fra i 2 software, che pensano di trovarsi in 2 dispositivi diversi.

La ovvia differenza fra le 2 periferiche virtuali è che una lavora a livello di rete e quindi può servire come punto di connessione ad un altro dispositivo posizionato ovunque sulla rete Internet (mai sentito parlare di VPN? Si realizzano tramite periferiche TUN), mentre l'altra lavora a livello inferiore, quindi può servire a connettere al livello di rete un livello fisico non... tanto fisico (mai sentito parlare di macchine virtuali? Hardware... non tanto hard).

```
# per creare una interfaccia TUN, indicando CHI può usarla
sudo ip tuntap add tun0 mode tun group users
sudo ip link set tun0 up

# per creare una interfaccia TAP, indicando CHI può usarla
sudo ip tuntap add tap0 mode tap group users
sudo ip link set tap0 up
```

CAPITOLO 7

# ping

**Nota:** Prerequisti: **Windows: command prompt. Linux, Mac: terminale**

Argomenti trattati: **Indirizzamento IP**

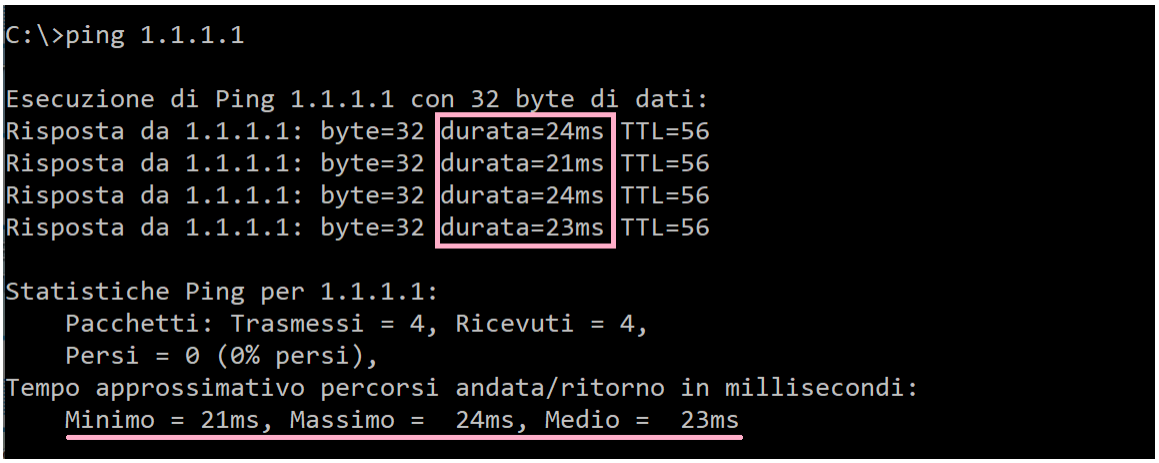
ping è un software di diagnostica di rete, implementato in tutti i sistemi operativi, che misura il tempo (in millisecondi) impiegato da uno o più pacchetti ICMP a raggiungere un altro dispositivo di rete e tornare indietro.

Tecnicamente ping invia un pacchetto ICMP di echo request e rimane in attesa di un pacchetto ICMP di echo response in risposta. Solitamente infatti la parte di OS dedicata alla gestione delle reti (stack di rete) è impostata per rispondere automaticamente con un pacchetto echo response alla ricezione di un pacchetto di echo request.

Il comando ping su Windows è impostato per inviare 4 pacchetti ICMP, attendere 4 risposte e poi calcolare le statistiche di ricezione e velocità. La sintassi del comando ping è la seguente:

```
$ ping [opzioni] host
```

Evitiamo di addentrarci nel discorso delle opzioni del comando ping e vediamo semplicemente il funzionamento e l'utilità dello stesso.



Come si vede, basta scegliere un target host (tramite IP o hostname) e lanciare il comando. Al termine dell'esecuzione bisogna osservare i pacchetti ritornati e le statistiche di viaggio degli stessi per valutare la salute e le performance della rete stessa.

> **Avvertimento:** Il comando ping su Windows è impostato automaticamente ad interrompersi dopo l'invio di 4 pacchetti.
>
> In ambienti Linux o Mac invece, il comando ping continua indefinitamente ad inviare pacchetti finché non lo si interrompe con un comando **CTRL + C** (mela + C su Mac). Quando viene interrotto l'invio vengono generate le statistiche.

Il comando ping insieme ai comandi ipconfig e traceroute sono un ottimo strumento di diagnostica e sono semplicissimi da utilizzare. L'importante è ragionare sui risultati che si ottengono e formulare deduzioni appropriate.

## 7.1 Ping test #1

Nel primo test, valutiamo la connettività della nostra rete. Controllando (con ipconfig/ifconfig/ip a seconda del sistema) abbiamo visto che il nostro IP è 192.168.0.7 e che il nostro gatweway è 192.168.0.1.

I test da eseguire sono nell'ordine i seguenti

```
# ping al nostro indirizzo
# se risponde significa che la scheda è attiva e funzionante
$ ping 192.168.0.7

# ping al nostro gatweway
# se risponde significa che la nostra rete è attiva e funzionante
$ ping 192.168.0.1

# ping ad un IP esterno
# se risponde significa che abbiamo connettività Internet
$ ping 1.1.1.1

# ping ad un host
# se risponde significa che anche la risoluzione DNS funziona!
$ ping google.it
```

Se tutti questi test hanno funzionato correttamente, potete stare certi che il vostro dispositivo è ben connesso alla rete Internet. In caso negativo, valutando quale test fallisce potete in maniera semplice individuare il problema.

## 7.2 Ping test #2

Nel secondo test valutiamo la velocità dei server DNS che stiamo utilizzando. Si tratta di risolvere un host tramite una risoluzione DNS manuale e poi valutare la differenza di prestazioni fra il ping con hostname e il ping con IP.

```
# risoluzione
$ nslookup example.com
1.2.3.4

# ping HOSTNAME
$ ping example.com
...
```

(continues on next page)

(continua dalla pagina precedente)

```
# ping IP
$ ping 1.2.3.4
...
```

Ovviamente questo test è molto empirico e non sempre funziona poiché non tutti i dispositivi rispondono ai ping :(

## 7.3 Ping test #3

Nel terzo test cercheremo di valutare la velocità della propria rete. L'idea di base è questa. Si scelgono 3 siti a caso (ad esempio: youtube.com, quotidiani.net, autoscout24.it) e si fanno i ping ad ognuno di essi. Si osservano i valori scegliendo il più alto riportato nei tre test e si valuta la rete secondo la seguente tabella:

| ping time (ms) | velocità rete |
|---|---|
| 0 - 20 | Ottima |
| 20 - 40 | Buona |
| 40 - 60 | Discreta |
| 60 - 80 | Sufficiente |
| oltre 80 | ... |

**Avvertimento:** Questa tabella e questo modo di valutare la velocità di una rete hanno pochissime basi scientifiche e sono solo una stima di massima che io di solito faccio per valutare una rete.

La velocità della rete dipende da moltissimi fattori, tra cui: i siti che visitate, l'orario di utilizzo, l'hardware a disposizione, la connessione wifi vs cablata, etc...

# CAPITOLO 8

# arp

**Nota:** Prerequisti: **Windows: command prompt, Linux, Mac: terminale**

Argomenti trattati: **Protocolli IP, ARP**

L'utility di rete ARP serve per accedere alla cache arp (appunto) in cui vengono mantenute le informazioni sul **neighbourhood** del dispositivo, ovvero sui dispositivi contattabili direttamente da esso, senza l'ausilio del gateway predefinito, tramite un invio diretto.

Come sappiamo il protocollo ARP funge da strumento di risoluzione degli indirizzi (ARP sta per Address Resolution Protocol) e infatti abbina ogni indirizzo IP contattabile (*reachable*) nella rete del dispositivo al suo indirizzo MAC.

Vediamo la sintassi generale:

```
$ arp [opzioni]
```

Le opzioni realmente interessanti sono solo 3 e sono quelle che servono per:

1. visualizzare la cache ARP,
2. aggiungere una voce alla cache (una coppia IP-MAC)
3. cancellare una voce (o tutte) da essa.

Ovviamente le operazioni di modifica della cache (inserimento e/o cancellazione) richiedono privilegi amministrativi mentre la visualizzazione della stessa è sempre abilitata perché qualunque software per accedere alla rete deve poterne usufruire.

Vediamo le opzioni:

| Opzione | Significato |
|---|---|
| -a | Visualizza la cache ARP |
| -s indirizzo_IP indirizzo_MAC | Inserisce la coppia IP:MAC nella cache ARP |
| -d indirizzo_IP | Elimina la voce corrispondente a indirizzo_IP dalla cache ARP |
| -d | Cancella tutta la cache ARP |

Come vedete le opzioni sono poche e semplici. Ancora più semplice se si considera che nel 99% dei casi l'unica che si utilizza è quella per visualizzare la cache corrente:

```
C:\>arp -a

Interfaccia: 172.25.37.11 --- 0x3
  Indirizzo Internet    Indirizzo fisico      Tipo
  172.25.37.40          00-19-fb-[redacted]    dinamico
  172.25.37.83          84-25-19-[redacted]    dinamico
  172.25.37.100         10-13-31-[redacted]    dinamico
  172.25.37.255         ff-ff-ff-ff-ff-ff     statico
  224.0.0.22            01-00-5e-00-00-16     statico
  224.0.0.251           01-00-5e-00-00-fb     statico
  224.0.0.252           01-00-5e-00-00-fc     statico
  239.255.255.250       01-00-5e-7f-ff-fa     statico
  255.255.255.255       ff-ff-ff-ff-ff-ff     statico
```

## 8.1 Esercizio di comprensione

1. Visualizza la cache ARP del PC nel laboratorio.
2. Prova a pingare un altro dispositivo del laboratorio acceso ma non presente in cache
3. Visualizza di nuovo la cache ARP
4. Prova a pingare un sito web qualsiasi
5. Visualizza di nuovo la cache ARP

Osserva quello che è cambiato nelle 3 visualizzazioni e conferma quanto abbiamo studiato.

# CAPITOLO 9

## traceroute

**Nota:** Prerequisti: **Windows: command prompt. Linux, Mac: terminale**

Argomenti trattati: **Protocolli IP, ICMP. Routing**

Prima d tutto chiariamoci su una importante banalità: il comando che esegue l'operazione di *tracerouting* ha un nome diverso su Window, su Linux, su Mac. Su Linux e Mac troviamo l'implementazione originale del tool di rete, che si chiama come l'operazione che esegue: **traceroute**.

Su Windows è stato progettato un nuovo tool, che si chiama **tracert**, fa esattamente la stessa cosa del precedente, con una interfaccia testuale sicuramente più carina e ordinata, al prezzo di ben tre ordini di grandezza (millisecondi contro microsecondi) e di una divergenza dallo standard che depone una volta di più sulla simpatica politica dell'azienda con le finestre.

Allego 2 screenshot (il primo su Windows, il secondo su Linux, su Mac è uguale a Linux) dell'esecuzione di un traceroute verso il target **1.1.1.1** (scelto solo per la sua semplicità numerica).

Il traceroute utilizza i pacchetti ICMP di tipo, appunto, *traceroute* per determinare i nodi di rete che il pacchetto attraversa. E verificare dunque i salti che il pacchetto compie per andare dal punto A al punto B.

Immagine dell'esecuzione di **tracert su Windows**.

```
C:\>tracert 1.1.1.1

Traccia instradamento verso one.one.one.one [1.1.1.1]
su un massimo di 30 punti di passaggio:

  1     1 ms    <1 ms    <1 ms  OpenWrt.lan [172.25.37.100]
  2    18 ms    16 ms    16 ms  10.8.131.27
  3    11 ms     9 ms     9 ms  10.250.140.34
  4    18 ms    17 ms    16 ms  10.8.132.225
  5    16 ms    13 ms    13 ms  10.7.128.21
  6    30 ms    28 ms    28 ms  10.254.1.1
  7    24 ms    24 ms    22 ms  93.57.68.102
  8    20 ms    20 ms    20 ms  93.57.68.137
  9    25 ms    22 ms    22 ms  cloudflare-nap.namex.it [193.201.28.33]
 10    23 ms    21 ms    21 ms  one.one.one.one [1.1.1.1]

Traccia completata.
```

Immagine dell'esecuzione di **traceroute su Linux o Mac**.

```
/home/adjam> traceroute 1.1.1.1
traceroute to 1.1.1.1 (1.1.1.1), 30 hops max, 60 byte packets
 1  router (172.25.37.100)  0.804 ms  1.009 ms  1.410 ms
 2  10.8.131.27 (10.8.131.27)  16.276 ms  23.645 ms  16.473 ms
 3  10.250.140.42 (10.250.140.42)  18.257 ms  25.814 ms  42.401 ms
 4  10.8.132.225 (10.8.132.225)  27.133 ms  21.049 ms  27.057 ms
 5  10.7.128.21 (10.7.128.21)  24.316 ms  24.347 ms  24.240 ms
 6  10.254.1.1 (10.254.1.1)  47.446 ms  46.269 ms 10.254.11.249 (10.254.11.249)  30.445 ms
 7  89.97.200.186 (89.97.200.186)  38.090 ms 93.57.68.102 (93.57.68.102)  21.120 ms 89.97.200.186 (89.97.200.186)  27.357 ms
 8  93.57.68.149 (93.57.68.149)  32.653 ms 93.57.68.141 (93.57.68.141)  19.304 ms 93.57.68.145 (93.57.68.145)  27.376 ms
 9  cloudflare-nap.namex.it (193.201.28.33)  41.690 ms  44.593 ms  45.547 ms
10  one.one.one.one (1.1.1.1)  49.769 ms  52.701 ms  45.405 ms
/home/adjam>
```

Analizzate i risultati ottenuti per comprendere al meglio il processo di routing dei pacchetti.

CAPITOLO 10

# Router fai da te *

**Nota:** Prerequisti: **Putty. Linux: terminale**

Argomenti trattati: **Protocolli IP, ICMP. Routing**

Questo si riesce a fare solo in laboratorio... serve *Armando*...

CAPITOLO 11

# netstat

**Nota:** Prerequisti: **Windows: command prompt. Linux, Mac: terminale**

Argomenti trattati: **Protocolli di trasporto**

L'utility di rete **netstat** permette di esaminare le connessioni attive, individuando le porte attive nel proprio dispositivo e quelle disponibili a ricevere dati in ingresso.

Permette inoltre (cose che a noi interessano meno) di visualizzare statistiche relative a connessioni di rete, tabelle di routing, interfacce di rete, masquerading e multicasting.

Vediamo le principali opzioni:

| Opzione per Windows | Opzione per Linux e Mac | Significato |
|---|---|---|
| -a | -a | Visualizza tutte le connessioni, attive e non |
| -n | -n | Visualizza i dati (IP e porta) in forma numerica, senza risoluzione |
| -o | -p | Visualizza il PID (identificatore) del processo che occupa quella porta |
| -p TCP | -t | Visualizza tutte (e solo) le connessioni TCP |
| -p UDP | -u | Visualizza tutte (e solo) le connessioni UDP |

Provate a combinare le varie opzioni, seguendo la sintassi del sistema operativo che state utilizzando, osservando i risultati.

CAPITOLO 12

# Wireshark

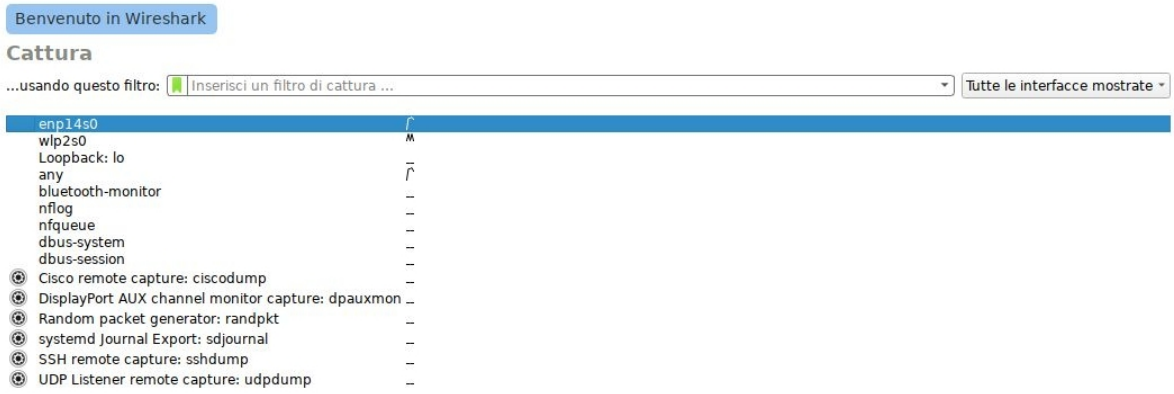
**Nota:** Prerequisti: **Windows, Mac, Linux**

Argomenti trattati: **Protocolli di rete e di trasporto. Più tutto il resto**

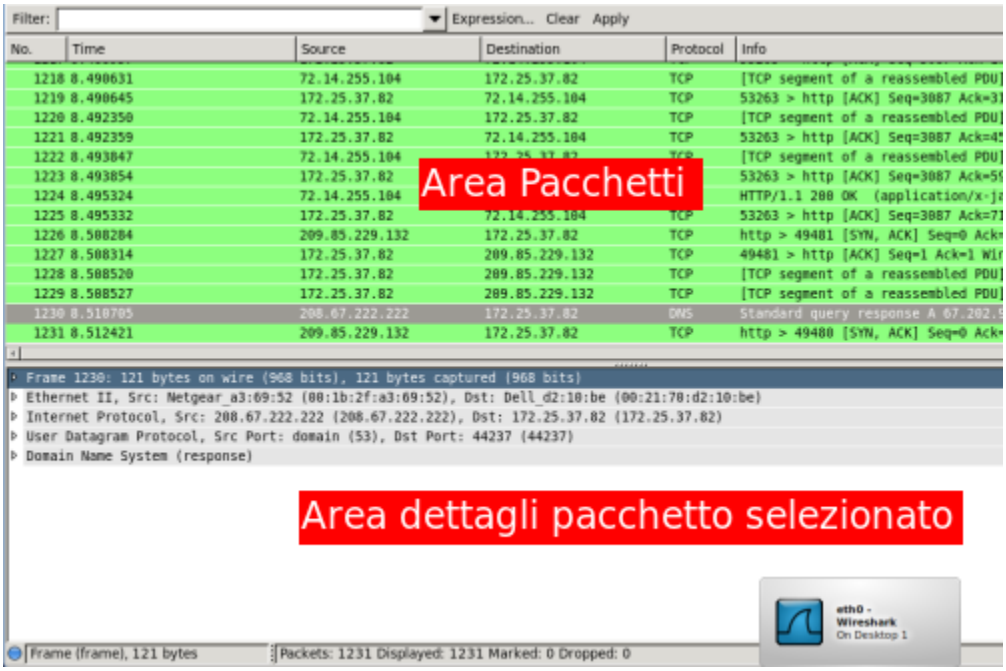
WireShark è un analizzatore di rete molto diffuso e molto potente, appartenente alla classe degli sniffer di rete. Usarlo tuttavia, almeno per operazioni di base o di studio del funzionamento della rete, è relativamente semplice.

Vedremo qui semplicemente una minima parte delle funzioni più importanti, come la cattura dei pacchetti, la selezione di questi ultimi per mezzo dei filtri e alcune funzioni che tornano comode soprattutto quando c'è una grossa mole di dati.

Per prima cosa occorre selezionare l'interfaccia di rete dalla quale si vogliono catturare i pacchetti in transito. Questa operazione attiverà la scheda in modalità "promiscua", cioè la renderà capace di intercettare tutti i pacchetti in transito sulla rete e non solo quelli destinati all'host.




Vediamo l'interfaccia principale di WireShark, cercando di capirne l'organizzazione:




Come si deve, il colpo d'occhio è chiaro e semplice: i pacchetti sniffati sono studiati ed elencati in ordine di cattura. Selezionandone uno si hanno tutte le informazioni relative.

Tra le *funzioni utili* di WireShark, elenchiamo le due più usate:

1. L'opzione **follow X Stream** dove X è uno tra TCP e UDP, permette di evidenziare il flusso di una comunicazione a cui il pacchetto selezionato appartiene. In questo modo si esce dall'ordinamento classico "per cattura" e si ottiene una lista di tutti i pacchetti relativi ad una connessione.
2. L'opzione **filtra pacchetti** permette una selezione dei pacchetti interessanti secondo un modello prestabilito.

La sintassi dei filtri è analoga alle condizioni dei linguaggi di programmazione. Immaginiamo di voler vedere tutti i pacchetti diretti al dispositivo con IP 1.2.3.4. Allora il filtro diventa:

```
ip.addr == 10.2.4.200
```

Oppure, vogliamo visualizzare tutti i pacchetti del protocollo HTTP e DNS. I nomi dei protocolli funzionano come variabili booleane, le operazioni logiche utilizzano la sintassi del linguaggio C ( && sta per AND, || sta per OR, ! sta per NOT), allora:

```
http && dns
```

In ogni caso, ci sono 2 auti fondamentali che vengono da Wireshark per la compilazione dei filtri:

1. l'autocompletamento: il software suggerisce come completare il filtro che si inizia a scrivere
2. il colore di sfondo: la barra dei filtri diventa rossa mentre si scrive un filtro e passa al verde appena il filtro è sintatticamente corretto.

## 12.1 Usare WireShark

Capisco che introdurre un software come WireShark ci dica tutto e niente delle sue potenzialità. Per capire davvero occorre provare!

Proviamo a fare allora dei semplici esercizi per capire il funzionamento del software.

1. Ricostruire una connessione HTTP (richiesta/risposta) elencando TUTTI i pacchetti inviati.
2. Filtrare tutti i pacchetti delle connessioni DNS, evidenziando i record presenti nei pacchetti.
3. Filtrare il traffico ARP, cercando di dedurre la struttura del protocollo stesso dall'osservazione dei suoi pacchetti.
4. Individuare un 3-way handshake TCP, valutando tempistiche e funzionamento tramite osservazione dei pacchetti.

# CAPITOLO 13

## nmap

**Nota:** Prerequisti: **Windows, Linux, Mac**

Argomenti trattati: **Protocolli di trasporto**

Nmap è un software libero distribuito con licenza GNU GPL da Insecure.org (https://insecure.org) e disponibile per il download sul sito https://nmap.org/ creato per effettuare port scanning, cioè mirato all'individuazione di porte aperte su un computer bersaglio o anche su range di indirizzi IP, in modo da determinare quali servizi di rete siano disponibili.

Vediamo la sintassi di base:

```
$ nmap OPTIONS TARGET
```

Il **TARGET** è il dispositivo o l'elenco di dispositivi che nmap deve scannerizzare per capire il tipo di dispositivo che ci troviamo davanti (produttore, sistema operativo, servizi attivi) e può essere uno tra questi:

- `HOST`: ad esempio *ciccio*, oppure *192.168.1.1*
- `RANGE`: ad esempio *192.168.1.1-20*
- `SUBNET`: ad esempio *192.168.1.0/24*

Ricorda comunque che la scansione che nnap esegue richiede del tempo quindi occhio ad impostare RANGE o SUBNET troppo ampi...

Le **OPTIONS** disponibili sono tranquillamente un migliaio... noi qui ovviamente noi vediamo le più semplici e interessanti:

**Suggerimento:** nmap è un software molto *veloce* considerato quello che fa e come lo fa.

Il problema è che impiega comunque un buon minuto per una scansione, quindi agli occhi delle persone normali risulta *lentissimo*.

Fra le opzioni ne abbiamo anche una per la velocità, con 5 possibilità: **-T0 (più lento ed accurato), -T1, -T2, -T3, -T4, -T5 (più veloce e potenzialmente impreciso)**.

Sta a voi decidere se e quale usare se l'attesa diventa insopportabile :)

L'opzione più veloce di scansione controlla solo le 100 porte più comuni:

```
$ nmap -F TARGET
```

La scansione di default senza privilegi amministrativi viene fatta con il 3-way handshake (TCP scan):

```
$ nmap -sT HOST
```

Se si hanno privilegi amministrativi è meglio procedere ad un SYN scan (più discreto e più veloce)

```
$ sudo nmap -sS HOST
```

**Avvertimento: TCP scan vs SYN scan**

Mentre un 3way handshake prevede l'invio e la ricezione di 3 pacchetti con l'attivazione di una connessione fra i 2 host e permette dunque al sistema target di *accorgersi* del dispositivo che lo scansiona, un SYN scan invia al TARGET solo il primo pacchetto dell'handshake e determina l'apertura di un servizio con la ricezione del pacchetto SYN+ACK, ma **non** risponde con un ACK ad esso, non aprendo la connessione sul target!

Le due precedenti scansioni ci elencano semplicemente le porte che rispondono o no al 3 way handshake. Per determinare i servizi realmente attivi dietro alle porte attive:

```
$ nmap -sV TARGET
```

Per cercare di capire il sistema operativo del dispositivo target (richiede privilegi amministrativi):

```
$ sudo nmap -O TARGET
```

Per una scansione *generica* di quale potrebbe essere il sistema operativo e i servizi attivi:

```
$ nmap -A HOST
```

Se si vuole capire quali sono gli host online in una rete:

```
$ nmap -sn SUBNET
```

## 13.1 Porte scansionate

Quando nmap scansiona le porte logiche di un dispositivo TARGET può ritornare risultati di 6 tipi:

| Classificazione Porta | Descrizione |
|---|---|
| open | Una porta che accetta connessioni |
| closed | Accessibile ma senza una applicazione in ascolto su di essa. Permette di capire che un sistema è attivo e senza firewall. |
| filtered | Non si può determinare con esattezza se la porta sia aperta o no. Le porte protette dai firewall sono così. |
| unfiltered | Una porta non protetta da firewall, ma che non si capisce se sia aperta o meno. Un amministratore acuto si nasconde di solito dietro ad essa... |
| open\|filtered | nmap è indeciso fra i 2 stati, ma è sicuro sia uno dei due. |
| closed\|filtered | nmap è indeciso fra i 2 stati, ma è sicuro sia uno dei due. |

## 13.2 Nmap Scripting Engine (NSE)

**Avvertimento:** Da un grande potere deriva una grande responsabilità

*(zio Ben)*

La caratteristica migliore di nmap è la possibilità di aumentare a dismisura le sue capacità di scanning grazie al meccanismo degli script e al suo NSE ovvero il software in grado di eseguirli.

Sono presenti centinaia di script per le scansioni più disparate, organizzati nelle seguenti categorie:

| Categoria | Descrizione |
|---|---|
| auth | Script per l'autenticazione e i privilegi utente. |
| broadcast | Network discovery basato su broadcast. |
| brute | Attacchi di tipo brute-force per indovinare le credenziali di accesso. |
| default | Gli script più popolari e considerati più utili. |
| discovery | Network, Service and Host discovery |
| dos | Attacchi di tipo "Denial of service" |
| exploit | Service exploitation on different CVEs |
| external | Scripts che si appoggiano a servizi o dati esterni per funzionare |
| fuzzer | Attacchi di tipo *fuzzing* ad app, servizi, reti. |
| intrusive | Attacchi aggressivi che potrebbero danneggiare il funzionamento della rete. |
| malware | Malware detections and exploration scripts |
| safe | Safe and non-intrusive/noisy scripts |
| version | OS, service and software detection scripts |
| vuln | Vulnerability detection and exploitation scripts |

Viste le categorie complete, sappiate che un elenco completo degli script disponibili con una descrizione esplicativa accanto si trova sul sito https://nmap.org/nsedoc/.

Per quanto riguarda il nostro corso, diciamo che prima di poter utilizzare gli script è bene assicurarsi che essi siano presenti, aggiornati all'ultima versione disponibile e catalogati nel database del sistema. Si ottiene questo risultato eseguendo il comando:

```
$ sudo nmap --script-updatedb
```

Fatto questo, la sintassi per eseguire gli script è molto semplice e si basa sull'opzione *–script*: ho fatto alcuni esempi per capire il funzionamento.

```
// SINTASSI GENERALE
$ sudo nmap --script QUALCOSINA TARGET

// Per eseguire tutti gli script di default verso un TARGET
sudo nmap --script default TARGET

// Per eseguire gli script dei gruppi broadcast e discovery verso un TARGET
sudo nmap --script broadcast,discovery TARGET

// come sopra, esattamente equivalente
sudo nmap --script "broadcast or discovery" TARGET

// Per eseguire tutti gli script relativi ad HTTP verso un target
sudo nmap --script http* TARGET

// Per eseguire lo script chiamato dhcp-discover verso un target
sudo nmap --script dhcp-discover TARGET

// Per eseguire solo gli script relativi ad HTTP del gruppo discovery verso un target
sudo nmap --script "http* and discovery" TARGET
```

## 13.3 Esempi ed Esercizi

Nel primo esempio proveremo ad interrogare il server DHCP per ottenere le informazioni di rete, fingendo di essere un client DHCP (con un MAC inventato) e visualizzando le informazioni ottenute senza realmente applicarle.

```
// l'opzione -sU indirizza la scansione sul protocollo UDP
// l'opzione -p 67 individua la porta del server DHCP: velocizza la scansione
// lo script si chiama dhcp-discover
$ sudo nmap -sU -p 67 --script dhcp-discover IP_SERVER_DHCP
```

Nel secondo esempio proviamo ad elencare le cartelle condivise da un generico PC con Windows, per ottenere informazioni su cartelle condivise eventualmente accessibili.

```
// opzione (-sU) per scansione UDP, opzione (-sS) per scansione TCP SYN
// Le porte elencate (137/udp e 139/tcp) servono per velocizzare le operazioni
// lo script si chiama smb-enum-shares
$ sudo nmap -sU -sS -p U:137,T:139 --script smb-enum-shares IP_SERVER_SMB
```

Nel terzo esempio proviamo ad ottenere informazioni dettagliate sul PC Windows che ci interessa studiare.

```
// opzione (-sU) per scansione UDP, opzione (-sS) per scansione TCP SYN
// Le porte elencate (137/udp e 139/tcp) servono per velocizzare le operazioni
// lo script si chiama smb-system-info
$ sudo nmap -sU -sS -p U:137,T:139 --script smb-system-info IP_SERVER_SMB
```

Nel quarto esempio faremo fare a nmap una scansione tipo traceroute di tutti gli hop attraversati con la localizzazione geografica delle posizioni di ognuna.

```
$ sudo nmap --traceroute --script traceroute-geolocation TARGET
```

Nel quinto esempio simuleremo un attacco (di 10 minuti) ad un server DNS allo scopo di testare la qualità della rete e del servizio DNS di quest'ultima. Attenzione...

```
$ sudo nmap -sU --script dns-fuzz TARGET
```

Nel sesto e ultimo esempio utilizzeremo uno script di tipo brute per tentare di indovinare nome utente e password di un utente collegato ad un Mac. Anche questo script ha ovviamente l'unico scopo di scoraggiare l'utilizzo di nomi utente e password semplici da indovinare.

```
$ sudo nmap -p 548 --script afp-brute IP_COMPUTER_MAC
```

# CAPITOLO 14

## Port Forwarding

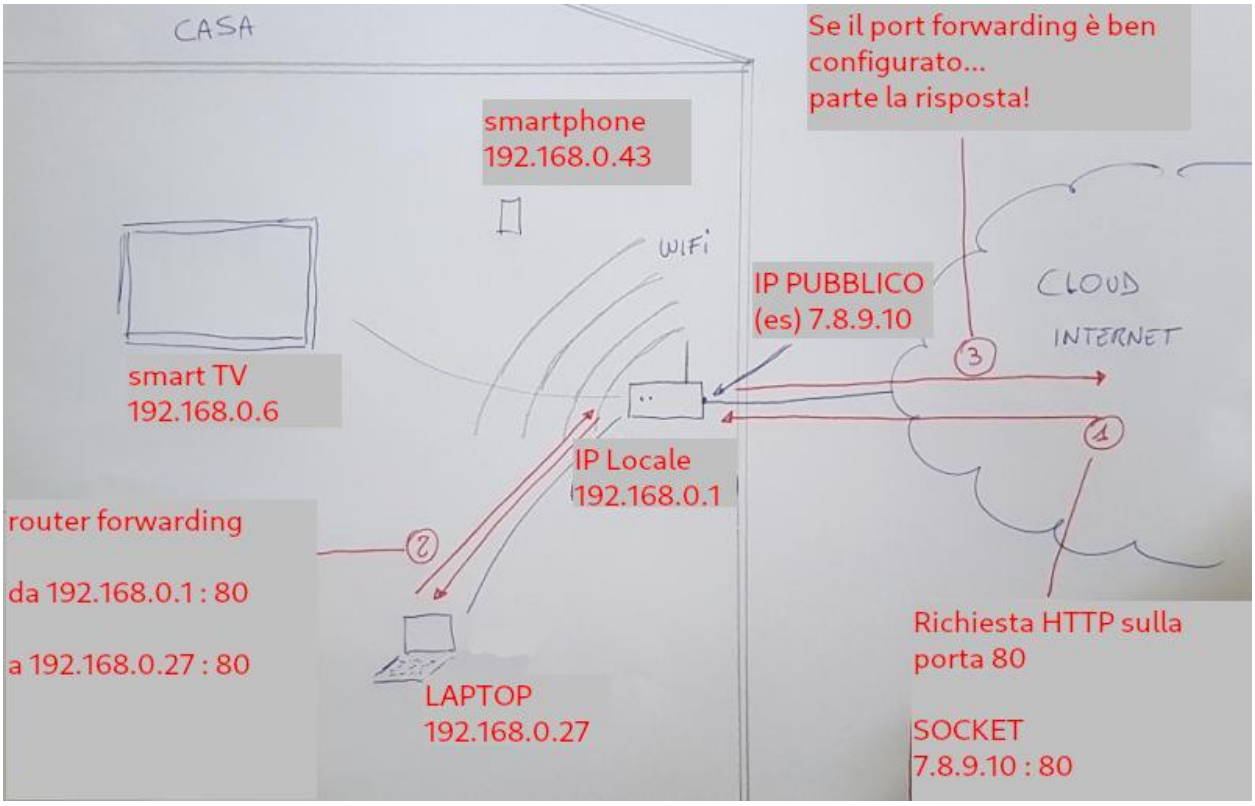
**Nota:** Prerequisti: **Browsing**

Argomenti trattati: **Protocolli di trasporto**

Il Port Forwarding è una abilità del livello di trasporto che permette di reindirizzare una connessione destinata ad un certo socket verso un altro socket, quando questa attraversa un dispositivo di rete (un router o un firewall). Esso è dunque una applicazione di NAT (Network Address Translation) poiché, spostando uno dei socket target della connessione rende possibile modificare l'IP (ovvero il mittente, oppure il destinatario) della stessa.

L'idea di questa esercitazione, da fare a casa, è quella di rendere disponibile un servizio installato su un computer locale sulla rete internet, ridirigendo le richieste che arrivano al modem router di casa dalla linea esterna verso il dispositivo interno che esegue un servizio.

Nella seguente immagine ho cercato di descrivere il problema immaginando un generico server WEB disponibile sul LAPTOP sulla porta 80 e raggiungibile dall'esterno tramite l'indirizzo IP (pubblico) del router.




Qualche minuto di tempo per analizzare (e godere) del disegno...

Bene!

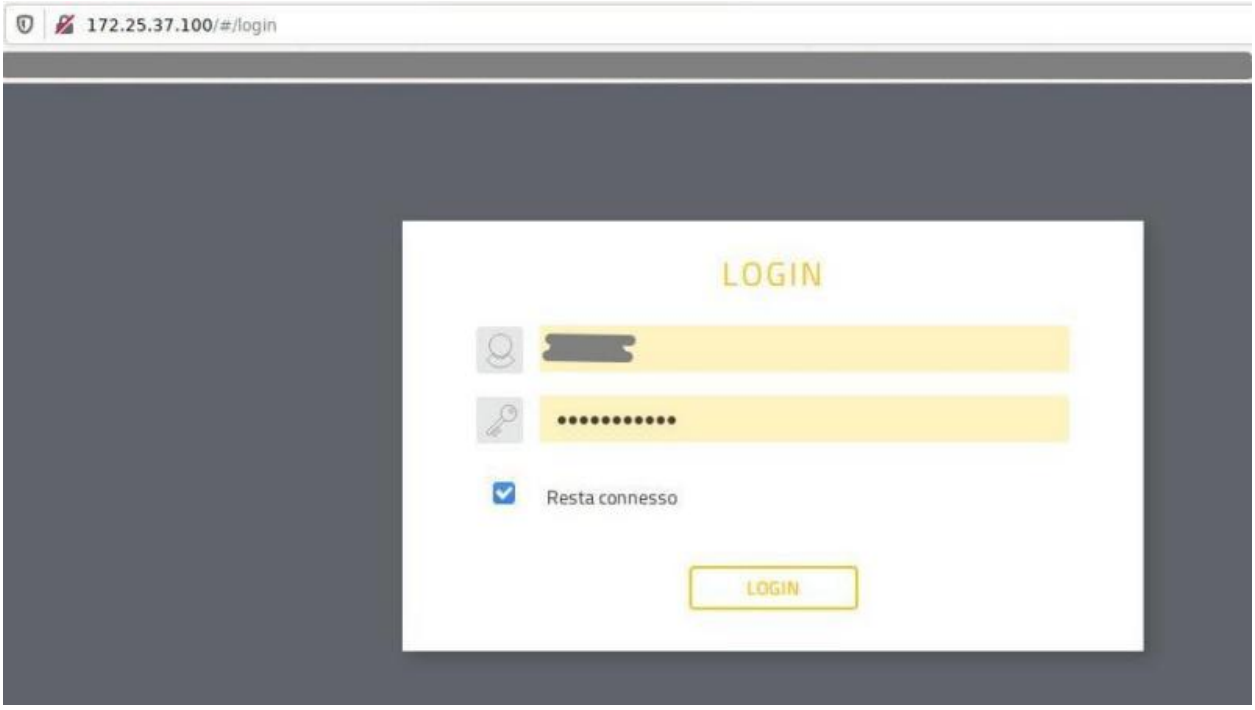
Per questa esperienza vi servono alcune informazioni relative alla rete di casa vostra. Un buon inizio sarebbe individuare:

- l'IP del vostro dispositivo (possibilmente un PC, oppure un Raspberry)
- l'IP del modem/router/access point di casa vostra

Non dovrebbe essere difficile desumerli osservando l'output del comando *ipconfig* (Windows) oppure *ifconfig* (Linux, Mac). L'IP del proprio dispositivo è quello indicato accanto alla scheda di rete; l'IP del proprio dispositivo di rete è solitamente il gatewar/server DHCP che desumete dalle informazioni di rete.

Adesso dovete collegarvi al router di casa vostra. Di solito le informazioni di accesso (IP, user, pass) sono proprio sul router, su una etichetta o su una placchetta sul retro/fondo del dispositivo.

Collegatevi utilizzando il browser e digitando l'URL: **http://IP_del_router**. Ad esempio potrebbe essere qualcosa tipo *http://192.168.1.1*. A casa mia la rete è un pò particolare :)




Adesso all'interno dell'interfaccia di configurazione del router dovete trovare qualcosa tipo **configurazione porte**, **port Forwarding**, **virtual server**, etc. . .


Da lì la configurazione è diversa per ogni router. . . posso aiutarvi poco :(

Quello che dovete fare è abbinare i due socket che devono comunicare (quello sul dispositivo e quello sul router).

**IP del router : 80 —–> IP del PC/Raspberry : 80**

Ricordate che siete sull'interfaccia del router, quindi il suo IP è solitamente implicito. Resta da indicare la porta da ascoltare (nel nostro esempio la porta 80, HTTP) e il socket che la deve ricevere (IP del dispositivo, il PC o il Raspberry, e ancora porta 80 oppure porta HTTP).

**Buona Fortuna!**

PS: fortuna, un corno! Dovete riuscirci…

PPS: quello che poi potete farci con il port Forwarding dipende dalle vostre conoscenze sul livello superiore!

CAPITOLO 15

# Firewall *

**Nota:** Prerequisti: **Raspberry, terminale**

Argomenti trattati: **Protocolli di rete e di trasporto. Più tutto il resto**

Va abbinato a router DIY. Vorrei provare con nftables. Serve *Armando*...

# CAPITOLO 16

## Sito web su Raspberry

**Nota:** Prerequisti: **Linux: terminale**

Argomenti trattati: **HTTP, URL**

Linguaggi per il web introdotti: **HTML, CSS, PHP**

Il nostro obiettivo è installare un server web su Raspberry. La scelta del software ricadrà ovviamente su Apache (https://www.apache.org).

Apache è il server web più famoso e utilizzato nel mondo. Si è sviluppato e migliorato assieme al protocollo HTTP e la sua struttura modulare gli permette di interagire anche con linguaggi di programmazione web lato server, come PHP, Perl, CGI o Python, per servire pagine dinamiche, oppure di implementare nuove specifiche opzionali, come il supporto per HTTP 2.0.

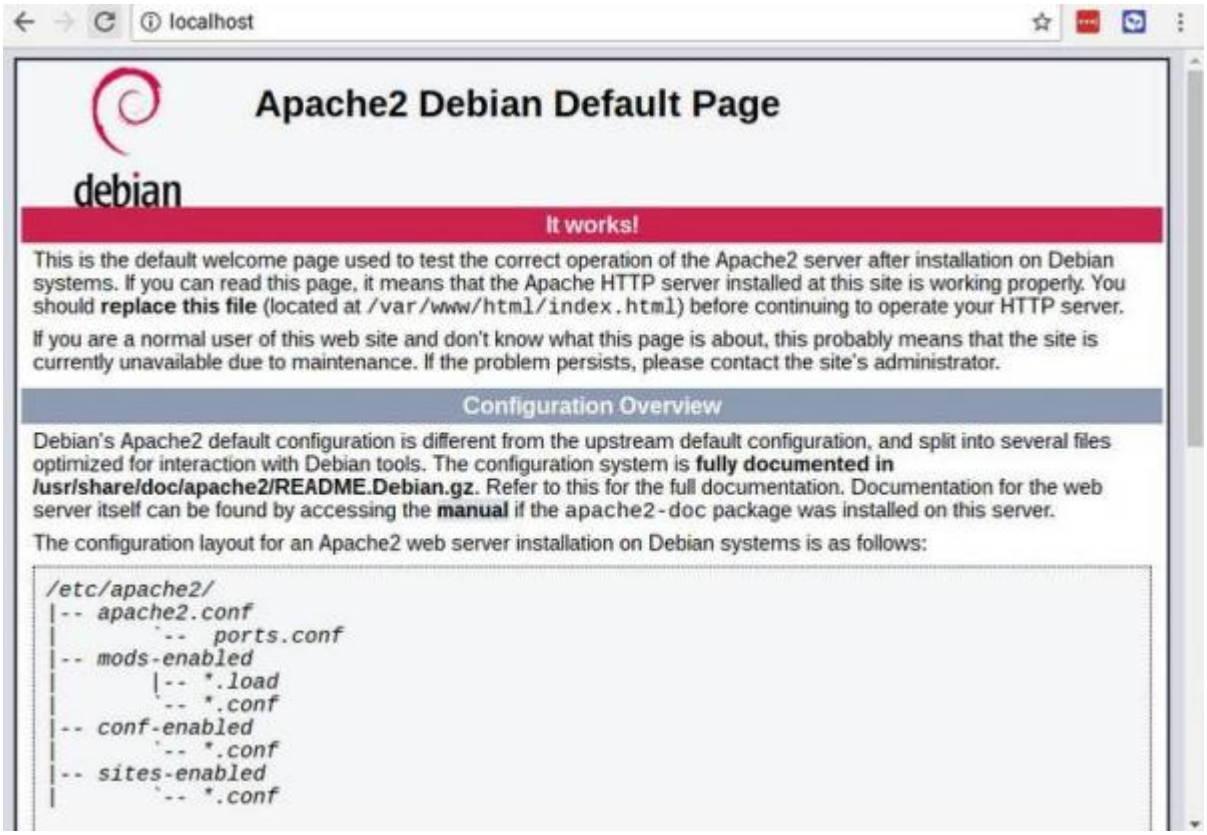
Per installare Apache, dobbiamo accedere ad un terminale e, dopo aver verificato che il sistema sia aggiornato, installare il software:

```
$ sudo apt install apache2
```

Questa semplice istruzione si occupa di tutto, ovvero:

- scarica il pacchetto software contenente il web server Apache dalla Rete
- lo installa
- abilita il servizio Apache per l'attivazione
- attiva il servizio Apache

Terminata l'esecuzione della linea di codice, si può procedere direttamente a testare il webserver! Per collegarsi ad esso, ci basta aprire il sito **http://localhost/** e vedere la pagina iniziale!




Questo significa che l'installazione è andata a buon fine! ;)

Per permettere a chiunque altro di vedere la nostra pagina possiamo utilizzare due strategie:

**Utilizzando l'indirizzo IP** Questa opzione è la migliore nel senso che funzionerà con tutti i dispositivi che possono connettersi al proprio Raspberry. Dovete controllare l'indirizzo IP del Raspberry tramite il comando **ip addr** oppure tramite il comando **ifconfig** e poi far connettere chiunque a **http://X.X.X.X/** dove (ovviamente) X.X.X.X è l'indirizzo IP del Raspberry.

**Utilizzando l'hostname del Raspberry** Questa opzione è la più semplice, ma può funzionare solo se il nome è registrato in un servizio DNS a cui entrambi i dispositivi (il Raspberry e il PC o il telefono da cui guardare il sito web). Si può controllare l'hostname del Raspberry (se non lo ricordate) tramite il comando **hostname -I**

## 16.1 Modificare il sito web

I file che rappresentano il sito web esposto tramite Apache si trovano in una cartella (e fino a qui...). La cartella in questione è la cartella

`/var/www/html`

quindi per accedervi e controllarne il contenuto dovete digitare:

```
$ cd /var/www/html
$ ls -al
```

Questo comando vi mostrerà, oltre che all'elenco dei file, informazioni su di essi come il proprietario, la dimensione, etc... Una roba tipo questa:

```
total 12
drwxr-xr-x  2 root root 4096 Jan  8 01:29 .
drwxr-xr-x  3 root root 4096 Jan  8 01:28 ..
-rw-r--r--  1 root root  177 Jan  8 01:29 index.html
```

In particolare vorrei farvi notare che il proprietario dei file del sito e della cartella che lo contiene è il Signore Onnipotente: **root**! Questo significa che ogni operazione di modifica, cancellazione o aggiunta file dovrebbe essere fatta con sudo... Mooooooolto noioso!

Cambiamo il proprietario della cartella:

```
$ sudo chown pi /var/www/html -R
```

Adesso potete aprire i file e lavorare sulla cartella del sito anche con i programmi grafici :)

## 16.2 Installare PHP

PHP è un linguaggio di programmazione web lato server; il suo nome è uno di quegi acronimi ricorsivi che tanto piacciono agli informatici. PHP infatti sta per **Php: Hypertext Preprocessor**, richiamando il nome PHP.

Esso è un *preprocessore*: è codice che viene eseguito quando il server web riceve una richiesta per una pagina web. La pagina richiesta è dunque *il programma* che viene eseguito al momento della richiesta. L'esecuzione di quel codice crea dunque una pagina HTML *al volo*, una **pagina dinamica**, che sarà poi rinviata all'utente tramite il server web.

PHP è il linguaggio di programmazione web lato server più famoso: Facebook, Twitter e Wikipedia sono interamente scritti in PHP (e già questi esempi bastano... mi pare).

Per installare PHP è sufficiente scrivere i seguenti comandi:

```
$ sudo apt install php
$ sudo systemctl restart apache2
```

Per testare se tutto ha funzionato, basta scrivere una pagina, ad esempio **test.php** da salvare nella cartella principale del sito e scrivere il seguente codice:

```
<?php
echo "<h1>Ciao</h1>";
echo "Sono la tua prima pagina in PHP :)";
?>
```

Ovviamente, per vedere il risultato, bisogna puntare il browser su **http://localhost/test.php**

CAPITOLO 17

# Sito web a casa tua

**Nota:** Prerequisti: **PC oppure Raspberry, terminale. esperienza "port forwarding"**

Argomenti trattati: **HTTP, URL, DNS, livello di trasporto**

Linguaggi per il web introdotti: **HTML, CSS, PHP**

Sì, hai capito bene… proviamo a *hostare* un sito web a casa di ognuno di voi. Ovviamente sarà un compito per casa e avrà alcune condizioni…

L'esperienza si snoda in 3 fasi distinte:

1. installare un server web sul proprio dispositivo (PC/Raspberry)
2. configurare il port forwarding fra il vostro dispositivo e il router di casa
3. aggiungere un DNS dinamico al vostro router in modo da rendere semplicemente raggiungibile il sito

## 17.1 Fase 1: web server

Per quanto riguarda la prima fare, se avete un Raspberry (oppure un computer con Linux) potete consultare la guida sul server web qui sopra.

Se avete solo un PC con Windows o Mac, potete installare un server web con **XAMPP** (https://www.apachefriends.org/index.html).

Nella pagina linkata ci dovrebbe essere anche un video di 90 secondi che spiega come si installa. Potete tranquillamente deselezionare tutte le opzioni possibili (apache e php vengono installati per forza). Tutto qui!

Create una pagina web semplice e divertente e salvatela come *index.html* nella cartella *htdocs* dentro la cartella *xampp* (magari rinominando quella presente inizialmente).

### 17.1.1 Se davvero funziona...

Controllate l'IP del dispositivo in cui avete installato il server web e provate a collegarvi con il vostro telefono (collegato al wifi di casa) al sito **http://IP_del_dispositivo**.

E se davvero funziona... potete già cominciare ad essere soddisfatti di voi!

## 17.2 Fase 2: port forwarding

Per quanto riguarda il port forwarding dovete seguire la guida a proposito nella sezione del livello di trasporto ed esporre la porta 80 del dispositivo in cui avete installato apache all'esterno.

### 17.2.1 Se davvero funziona...

Controllate con un sito web tipo https://whatismyipaddress.com/ il vostro indirizzo pubblico e provate con il vostro telefono (non collegato al wifi) a collegarvi al sito **http://vostro_IP_pubblico**.

E se davvero funziona... fate sapere al prof quanto siete tosti!!!

## 17.3 Fase 3: Dynamic DNS

Ultima fase... configurazione di un DNS dinamico per rendere il vostro sito raggiungibile con semplicità da chiunque!

Al di là di questa fase c'è lo zen delle conoscenze di rete, quindi non starò a spiegarvi molto se non a darvi due imbeccate:

1. nei router solitamente è già presente il supporto per il DNS dinamico. Provate a controllare nella sua interfaccia (lo avete già fatto per il port forwarding) e utilizzate uno dei servizi indicati lì.
2. i servizi di DNS dinamico forniscono gratuitamente, per un periodo limitato, host del terzo livello per alcuni servizi. Alcuni fra i più famosi siti per il DNS dinamico sono:
    - https://dyndns.it/
    - https://www.noip.com/
    - https://cloudns.net/

Fatelo! Se volete un posto in paradiso... fatelo!

### 17.3.1 Se davvero funziona...

Collegatevi tramite il vostro telefono (non collegato al wifi) al sito **http://vostro.host.dinamico**.

E se davvero funziona... lasciate il dispositivo acceso per pavoneggiarvi a lezione con il prof estasiato e i vostri compagni ammirati.

CAPITOLO 18

# Telnet e HTTP

**Nota:** Prerequisti: **OS, terminale**

Argomenti trattati: **Telnet, HTTP**

## 18.1 Introduzione

Telnet è un piccolo client da terminale, disponibile su ogni sistema operativo, che serve per simulare connessioni in chiaro a qualsiasi socket desideriamo. La sua utilità sta nel fatto che permette di *dialogare* con qualunque protocollo del livello superiore che utilizza testo semplice per le sua sintassi.

A pensarci bene, tutti i protocolli del livello superiore che abbiamo studiato (o che studieremo) utilizzano dati codificati in ASCII, quindi... telnet può essere un semplice strumento per provare ad analizzarli!

La sua sintassi è semplicissima:

```
$ telnet HOST PORT
```

Da dopo l'avvenuta connessione bisogna scrivere.. nella lingua del servizio con cui vogliamo dialogare, ovvero bisogna usare le specifiche del protocollo!

## 18.2 TELNET e HTTP

Praticamente con telnet ci colleghiamo ad un server HTTP e poi possiamo provare a scrivere a mano una richiesta HTTP. Nell'esempio proviamo a connetterci al generico sito **www.esempio.com** e a chiedere la homepage con una richiesta GET.

```
$ telnet www.esempio.com 80

... risposta ...

GET / HTTP/1.1
Host: www.esempio.com

... risposta ...
HTTP/1.1 200 OK
Date: Wed, 21 Jan 2004 22:13:05 GMT
Server: Apache/1.3.12-Turbo
Content-Type: text/html
<html>
<head>
<title>Esempio</title>
</head>
<body>
<h1>Esempio</h1>
Hai capito?
</body>
<html>
```

Tutto qui... semplice, ma efficace! Secondo me, anche molto affascinante.

# CAPITOLO 19

# DNS Clients

**Nota:** Prerequisti: **OS, terminale**

Argomenti trattati: **DNS, URL, Indirizzamento IP**

La funzionalità di risoluzione DNS ai software che ne hanno bisogno (praticamente tutti quelli che si connettono alla rete) viene solita fornita dal Sistema Operativo tramite le API di sistema.

Se però si vogliono ottenere risoluzioni DNS da poter consultare e studiare, è possibile utilizzare un client DNS testuale. Fra i pochi che esistono la scelta ricade forzatamente su **nslookup** (name server lookup) che ha il pregio di essere molto semplice (purtroppo a volte a scapito della dovizia di particolari) e di essere disponibile su tutti i sistemi operativi importanti.

Il client nslookup può essere eseguito in due modalità:

1. command line mode
2. interactive mode

## 19.1 nslookup: command line mode

Questa modalità è la più semplice e scarna. Si digita **nslookup HOST** e questi ritorna la risoluzione (i record A, AAAA e CNAME presenti) dell'host fornito e termina

```
$ nslookup www.adjam.org
Server:         172.104.237.57
Address:        172.104.237.57#53

Non-authoritative answer:
Name:   www.adjam.org
Address: 217.64.195.209
Name:   www.adjam.org
Address: 2001:4b78:1001::5701
```

Come si vede, in *command line mode* nslookup restituisce una risposta standard contentente il server che ha fatto la risoluzione (un server non autorevole, come vediamo sotto) e le risoluzioni trovate, evidentemente, in questo caso, un record A e un record AAAA.

## 19.2 nslookup: interactive mode

Per entrare in *interactive mode* basta digitare **nslookup** senza parametri od opzioni. In questo modo cambia il prompt (diventa un maggiore, **>**) e lì si può richiedere la risoluzione di qualunque host ci interessi. Quando abbiamo finito, possiamo uscire dalla modalità interattiva digitando **exit**.

```
$ nslookup

> google.it
Server:         172.104.237.57
Address:        172.104.237.57#53

Non-authoritative answer:
Name:   google.it
Address: 172.217.17.67
Name:   google.it
Address: 2a00:1450:400e:805::2003

> libero.it
Server:         172.104.237.57
Address:        172.104.237.57#53

Non-authoritative answer:
Name:   libero.it
Address: 213.209.17.209

> exit

$
```

la modalità interattiva permette inoltre di fornire alcuni parametri ad nskooup per modificare la risoluzione richiesta, specificandola meglio:

| comando | descrizione |
|---|---|
| host | l'host da risolvere. |
| server ip | l'ip del server DNS da utilizzare nelle prossime risoluzioni interattive |
| set type=record | serve per richiedere la risoluzione del tipo di record indicato. Valori possibili per *record* sono: A, AAAA, CNAME, MX, NS, PTR ... |

Qua di seguito farò 2 esempi per vedere come si utilizzano le 2 opzioni non testate: *server* e *set type*

### 19.2.1 Esempio #1: cambiare server DNS

Voglio specificare ancora una volta che questa opzione è disponibile solo in modalità interattiva e che il cambio di server DNS vale solo per le risoluzioni DNS fatte con nslookup in **questa** modalità interattiva! Il sistema non è minimamente interessato da questa modifica e alla chiusura di nslookup neanche esso se ne ricorderà:

```
$ nslookup

(la prima risoluzione, fatta col server DNS di default)
> www.liceodavincijesi.edu.it
Server:         172.104.237.57
Address:        172.104.237.57#53

Non-authoritative answer:
Name:   www.liceodavincijesi.edu.it
Address: 89.46.109.18

(cambio di server DNS per la risoluzione)
> server 1.1.1.1
Default server: 1.1.1.1
Address: 1.1.1.1#53

(la prossima risoluzione viene fatta verso il server 1.1.1.1)
> gazzetta.it
Server:         1.1.1.1
Address:        1.1.1.1#53

Non-authoritative answer:
Name:   gazzetta.it
Address: 40.1
```

### 19.2.2 Esempio #2: risolvere un record MX

Può essere utile e interessante risolvere un record MX tramite nslookup. Per farlo dobbiamo impostare il tipo di record da risolvere con *set type*. Ricordo però che il record MX ci dirà praticamente il record A che individua il dispositivo che fornisce il servizio di posta, che dovrà dunque poi essere risolto per ottenere l'indirizzo IP.

```
$ nslookup

(chiedo di risolvere record MX)
> set type=MX

(risolvo libero.it come MX. Il record MX punta il record con host indicato sotto)
> libero.it
Server:         172.104.237.57
Address:        172.104.237.57#53

Non-authoritative answer:
libero.it       mail exchanger = 10 smtp-in.libero.it.

(ritorno a risolvere record A, come di default)
> set type=A

(risolvo il record A abbinato al record MX)
> smtp-in.libero.it
Server:         172.104.237.57
Address:        172.104.237.57#53

Non-authoritative answer:
Name:   smtp-in.libero.it
Address: 213.209.1.129
```

## 19.3 Web clients

Capita spesso di voler utilizzare il DNS non tanto per verificare il funzionamento di un server, ma per verificare l'effettiva risolvibilità di un sito web. In questo caso tornano utili alcuni client DNS implementati direttamente come servizi WEB. Ne cito alcuni, ritrovati rigorosamente nella prima pagina del mio motore di ricerca preferito all'input *DNS web clients*:

- https://www.whatsmydns.net/

  Esso è un sito che permette di specificare una stringa di ricerca, un tipo di record ed effettua la stessa ricerca su una serie di almeno 20 server DNS sparsi per il mondo. È un servizio molto comodo e interessante, nel caso si voglia studiare la propagazione di un certo nome di dominio.

- https://toolbox.googleapps.com/apps/dig/

  Rappresente la versione web del famoso client testuale *dig*, sviluppato al server *BIND*, il server web più famoso al mondo. Poco da dire e facile da usare. Molto completo nelle risposte (troppo per noi?), ma molto semplice da utilizzare nella sua interfaccia web.

# CAPITOLO 20

# DNS Server

**Nota:** Prerequisti: **OS, terminale**

Argomenti trattati: **DNS, URL**

In questo capitolo andremo ad installare un server DNS su un dispositivo con OS Linux. Questo dispositivo monta una versione di Linux con package manager **apt**, quindi potrebbe essere un RaspberryPi oppure una installazione di Ubuntu Server.

Il server DNS che andiamo ad installare si chiama **BIND** (https://www.isc.org/bind/). Bind è un server DNS open source sviluppato dall'Internet Systems Consortium e che rappresenta il server DNS in assoluto più utilizzato sulla rete Internet. Credo che molti (se non tutti i) server DNS della root zone siano implementati con Bind e così molti dei server autoritativi esistenti. Per quanto riguarda il mercato dei resolver invece... il problema si allarga.

Per installare il server DNS BIND sul nostro OS, dopo un aggiornamento generale dei pacchetti installati, ci basterà fare un semplice:

```
$ sudo apt install bind9
```

I file di configurazione dello stesso si trovano nella directory */etc/bind*. Il file principale si chiama *named.conf* che non contiene altro che riferimenti agli altri file che organizzano in questo modo la configurazione del server a blocchi indipendenti.

Il file contiene già nella sua installazione un elenco degli IP dei server della **DNS root zone**: il file */usr/share/dns/root.hints*, quindi tecnicamente è già pronto a partire e funzionare come resolver per qualunque sito: avrà bisogno solo di un pò di tempo (e richieste) per costruire la sua cache e velocizzarsi.

Avviatelo, eventualmente configuratelo come vostro server DNS e testatelo.

CAPITOLO 21

# DNS Dinamico *

**Nota:** Prerequisti: **Raspberry, terminale linux**

Argomenti trattati: **DNS, URL**

Magari lo abbiniamo al port forwarding se si riesce a trovare una metodologia semplice per farlo fare a tutti..

# CAPITOLO 22

# DHCP Server

**Nota:** Prerequisti: **Linux: terminale**

Argomenti trattati: **DHCP, DNS, Indirizzamento IP**

In questo capitolo andremo ad installare un server DHCP su un dispositivo con OS Linux. Questo dispositivo monta una versione di Linux con package manager **apt**, quindi potrebbe essere un RaspberryPi oppure una installazione di Ubuntu Server.

Se avete studiato il protocollo DHCP, sapete che per far funzionare un server DHCP c'è bisogno di un dispositivo con un indirizzo statico. Il capitolo sul **Networking** spiega come ottenere questo risultato.

Il server DHCP che andiamo ad installare si chiama **KEA** (https://www.isc.org/kea/). KEA è un server DHCP open source sviluppato dall'Internet Systems Consortium come (futuro) sostituto dello storico *ISC DHCP SERVER*. Se volete saperne di più... leggete dal sito che ho linkato o guardate Wikipedia ;)

Per installare il server DHCP KEA, relativo al protocollo IPv4 sul nostro OS, dopo un aggiornamento generale dei pacchetti installati, ci basterà fare un semplice:

```
$ sudo apt install kea-dhcp4-server
```

Il file di configurazione dello stesso si trova nella directory */etc/kea*. Noi dobbiamo modificare il file *kea-dhcp4.conf*. Poiché in esso trovate una marea di commenti e spiegazioni a tutti i parametri possibili e immaginabili, ho provato a scriverne una versione *light*, mettendo in evidenza SOLO i parametri importanti o indispensabili

```
// Va modificato in questi 3 punti (leggilo TUTTO per trovarli):
// - cambia il nome della tua interfaccia di rete in cui vuoi attivare il server DHCP
// - cambia lo SCOPE degli indirizzi che vuoi servire tramite DHCP
// - cambia le impostazioni di rete, come GW e DNS da inviare ai client
//
// Salva il file modificato su /etc/kea/kea-dhcp4.conf (richiede privilegi␣
↪amministrativi per farlo)
// In bocca al lupo... e buona lettura!
```

(continues on next page)

(continua dalla pagina precedente)

```
{

"Dhcp4": {
    "interfaces-config": {
        "interfaces": [ "eth0" ]

        // USEFUL FOR RELAY
        // "dhcp-socket-type": "udp"
    },

    // NON TOCCARE ;)
    "control-socket": {
        "socket-type": "unix",
        "socket-name": "/tmp/kea4-ctrl-socket"
    },

    // KEA può utilizzare diversi database per mantenere i propri lease
    // - "memfile" : un semplice file CSV salvato da qualche parte
    // - "mysql", "postgres", "cassandra": database esterni (complicato)
    // Per saperne di più, cerca su internet "KEA LEASE STORAGE"
    "lease-database": {
        "type": "memfile",
        "lfc-interval": 3600
    },


    // PARAMETRI AVANZATI PER IL LEASE
    // meglio non toccare anche qui...
    "expired-leases-processing": {
        "reclaim-timer-wait-time": 10,
        "flush-reclaimed-timer-wait-time": 25,
        "hold-reclaimed-time": 3600,
        "max-reclaim-leases": 100,
        "max-reclaim-time": 250,
        "unwarned-reclaim-cycles": 5
    },


    // LEASE TIMER(s)
    "renew-timer": 900,
    "rebind-timer": 1800,
    "valid-lifetime": 3600,


    // PARAMETRI ADDIZIONALI
    "option-data": [

        // DNS
        {
            "name": "domain-name-servers",
            "data": "192.0.2.1, 192.0.2.2"
        },


        // NOME DI DOMINIO DELLA RETE
        // ("completa" il nome di un pc, come un "cognome".
        // Il PC chiamato PC1, nella rete sarà conosciuto come PC1.adjam.org)
```

(continues on next page)

(continua dalla pagina precedente)

```
        {
            "name": "domain-name",
            "data": "adjam.org"
        },


        // DOMINIO DI RICERCA
        // se questo parametro vale "pippo.com" e tu cerchi tramite DNS un nome
        // semplice, tipo "ciccio", prova a risolvere sia "ciccio" che "ciccio.pippo.
↪com"
        {
            "name": "domain-search",
            "data": "mydomain.example.com, example.com"
        },


        // Giuro... non ho mai capito. Lasciamoli stare...
        {
            "name": "boot-file-name",
            "data": "EST5EDT4\\,M3.2.0/02:00\\,M11.1.0/02:00"
        },

        {
            "name": "default-ip-ttl",
            "data": "0xf0"
        }

    ],


    // LA SOTTORETE CHE CONTIENE LO SCOPE
    // (per ogni sottorete ci può essere un solo scope)
    "subnet4": [
        {
            // LA SUBNET INTERA
            "subnet": "192.0.2.0/24",


            // LO "SCOPE" degli indirizzi assegnabili tramite DHCP per la subnet␣
↪precedente
            "pools": [ { "pool": "192.0.2.1 - 192.0.2.200" } ],


            // IL GATEWAY ("router" nella terminologia DHCP)
            "option-data": [
                {
                    "name": "routers",
                    "data": "192.0.2.1"
                }
            ],


            // PRENOTAZIONI
            // indirizzi FUORI dallo SCOPE assegnati a precisi client per motivi...
            // (se non volete usare le prenotazioni, potete cancellare il segmento␣
↪fino al segno)
            "reservations": [
```

(continues on next page)

(continua dalla pagina precedente)

```
                // PRENOTAZIONE semplice CON MAC/IP
                {
                    "hw-address": "1a:1b:1c:1d:1e:1f",
                    "ip-address": "192.0.2.201"
                },

                // PRENOTAZIONE con parametri alterati.
                // Solo questo client avrà una configurazione dedicata
                {
                    "hw-address": "1a:1b:1c:1d:1e:12",
                    "ip-address": "192.0.2.203",
                    "option-data": [ {
                        "name": "domain-name-servers",
                        "data": "10.1.1.202, 10.1.1.203"
                    } ]
                },

            ]
            // FINE PRENOTAZIONI
        }
    ],


// Logging configuration.
// Direi che va IGNORATA e LASCIATA così com'è...
    "loggers": [
    {
        "name": "kea-dhcp4",
        "output_options": [
            {
                // IL FILE DOVE POTRETE SFROGIARE I LEASE...
                "output": "/var/log/kea-dhcp4.log"
            }
        ],

        // Può valere: FATAL, ERROR, WARN, INFO, DEBUG
        // Ogni valore comprende TUTTI i precedenti. Lasciatelo a INFO
        // E bravi per essere arrivati a leggere fino a qui!
        "severity": "INFO",

        // Un numero da 0 (poche info) a 99 (un botto di info)
        "debuglevel": 0
    }
  ]
}
}
```

Come vedete, viene lungo uguale... però potete scaricare una copia qui .

Fatto quanto richiesto e salvato il file, attiviamo (o riattiviamo) il servizio *kea-dhcp4*.

```
$ sudo systemctl (re)start kea-dhcp4
```

Da questo in momento in poi, se non ci sono errori, il servizio DHCP è attivo :)

CAPITOLO 23

# Access Point Fai da te

**Nota:** Prerequisti: **Raspberry, terminale linux**

Argomenti trattati: **DHCP, DNS, Indirizzamento IP**

L'idea è quella di implementare un Access Point WIFI analogo a quelli che si hanno a casa per la connessione, con server DHCP e server DNS configurati manualmente.

Prima di andare avanti, ricordiamoci di aggiornare il sistema, ripulire e riavviare.

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt autoremove
$ sudo reboot
```

## 23.1 Scegliere un piano di indirizzamento

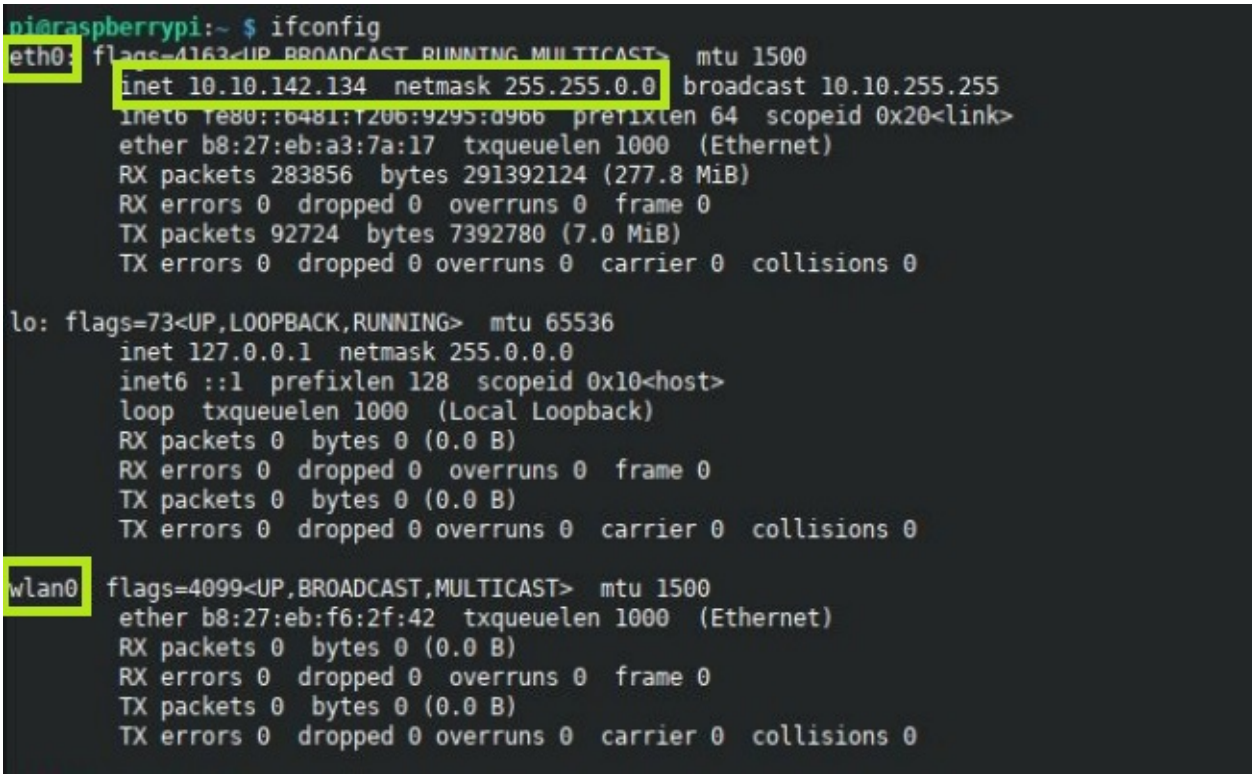
Da quello che abbiamo studiato sappiamo che ogni scheda di rete ha il suo indirizzo IP. Nel nostro Raspberry ci sono 2 schede di rete:

1. la schede di rete cablata, che si chiama *eth0*
2. la scheda di rete wifi, che si chiama *wlan0*

La scheda di rete cablata dovrebbe essere collegata alla rete della scuola, quindi ha già un indirizzo IP fornito dai server DHCP della scuola. La scheda di rete wifi non dovrebbe essere collegata a nulla e quindi non avere alcun indirizzo.

Per verificare i nomi delle schede di rete e l'indirizzamento attuale del nostro Raspberry procediamo da terminale con il comando *ifconfig* (Valido anche da terminale Mac e analogo del comando ipconfig di Windows).

Si dovrebbe vedere qualcosa di simile a questo:




L'indirizzo IP della scheda WIFI dovrà essere statico e scelto da noi. Per il mio esempio e come riferimento in quello che scriverò dopo, io scelgo l'indirizzo 192.168.0.1/24.

A questo punto, immaginando che collegherò il server DHCP all'interfaccia *wlan0* dovrò scegliere uno scope e identificare tutte le informazioni da passare ai client DHCP.

Io ho scelto le seguenti:

1. scope: 192.168.0.11-30 con TTL di 1 ora
2. gateway sarà ovviamente l'IP del Raspberry: 192.168.0.1
3. DNS sarà ancora una volta il Raspberry.

Ultima cosa, poiché si va ad implementare una rete wifi con password, bisogna scegliere il nome della rete Wifi (che si definisce SSID) e la chiave di accesso (la *password* del wifi).

Tenendo bene a mente (no, meglio se segnate su un foglio) le precedenti informazioni, possiamo procedere nell'implementazione del router/AccessPoint con Raspberry.

## 23.2 Installare e configurare dnsmasq

L'installazione è facile:

```
$ sudo apt install dnsmasq
```

Il file di configurare da modificare è il seguente

```
$ sudo nano /etc/dnsmasq.d/dnsmasq.conf
```

Va impostato in questo modo, trovando le sezioni opportune:

```
# to activate dhcp server on 1 interface
interface=wlan0
bind-interfaces

# scope
dhcp-range=192.168.0.11,192.168.0.30,255.255.255.0,1h

# options
dhcp-option=option:router,192.168.0.1

# dns configuration
listen-address=192.168.0.1
server=1.1.1.1
domain-needed
bogus-priv
```

## 23.3 Installare hostapd

Qui l'installazione è più complicata :)

```
$ sudo apt install hostapd
$ sudo systemctl unmask hostapd
$ sudo systemctl stop hostapd
```

Il file di configurazione va inserito nel percorso */etc/hostapd/hostapd.conf*, quindi con il comando:

```
$ sudo nano /etc/hostapd/hostapd.conf
```

Va copiato dentro pari pari il seguente codice, modificando opportunamente l'SSID scelto e la chiave di accesso:

```
interface=wlan0
driver=nl80211
ssid=NOMESSIDSCELTO
hw_mode=g
channel=2
ieee80211n=1
wmm_enabled=1
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=PASSWORDWIFIALMENO8CARATTERI
rsn_pairwise=CCMP
```

Fai in modo che il file di configurazione venga caricato dal demone hostapd: apri */etc/default/hostapd* e modificalo come indicato

```
$ sudo nano /etc/default/hostapd
```

Va modificata un'unica riga, in corrispondenza della voce DAEMON_CONF che va decommentata e riempita come indicato.

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Ok, siamo pronti!

## 23.4 Ultime impostazioni

Le ultime operazioni da fare servono per far funzionare il Raspberry come un router e permettergli dunque di condividere la sua connettività con tutti quelli connessi al suo Wifi tramite hostapd. Sono operazioni standard che vanno eseguite una ad una nella riga di comando.

Le divido blocco per blocco per evidenziare i vari passaggi.

Impostazione IP statico rete Wifi

```
$ sudo ip link set wlan0 up
$ sudo ip addr add 192.168.0.1/24 dev wlan0
```

Abilitazione traffico di rete attraverso il Raspberry

```
$ sudo sysctl -w net.ipv4.ip_forward=1
$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
$ sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j␣
↪ACCEPT
$ sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

Ultimo, se non ci sono messaggi di errori precedenti, avviare hostapd e dnsmasq.

```
$ sudo systemctl start hostapd
$ sudo systemctl start dnsmasq
```

Ecco fatto, dovrebbe funzionare tutto!

Prendete il vostro telefono e provate a connettervi alla rete Wifi del Raspberry e a navigare!

CAPITOLO 24

# Mail Server

**Nota:** Prerequisti: **Raspberry, terminale linux**

Argomenti trattati: **SMTP, POP, IMAP, URL, DNS**

L'idea è quella di implementare un Mail Server completo su Raspberry, con un MTA per il protocollo SMTP, un server POP e un server IMAP per la ricezione, l'invio e lo smistamento della posta elettronica.

Prima di andare avanti, ricordiamoci di aggiornare il sistema, ripulire e riavviare.

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt autoremove
$ sudo reboot
```

## 24.1 Configurare l'hostname

La prima cosa da fare per raggiungere il nostro obiettivo è di sistemare **l'FQDN (Fully Qualified Domain Name)** del nostro computer. Nel mio esempio esso sarà:

- HOSTNAME: **raspberrypi**
- DOMAIN NAME: **scuola.lan**

l'FQDN che identifica il sistema sarà dunque **raspberrypi.scuola.lan** con il servizio di posta installato in esso a servire il dominio **@scuola.lan**.

**Nota:** Se durante l'esperienza di gestione della posta si vuole configurare anche il DNS, ricordatevi che dovete creare una zona autoritativa per il primo livello **.lan** e configurare in essa almeno i seguenti record:

**Record A: scuola.lan.** Punta l'IP del Raspberry.

**Record A: raspberrypi.scuola.lan.** Punta l'IP del Raspberry.

**Record MX: scuola.lan.** Punta il record A raspberrypi.scuola.lan

Volendo è possibile aggiungere anche i terzi livelli *mail*, *pop*, *imap* (su record A) tutti che puntano l'IP del Raspberry.

Per impostare FQDN come desiderato, eseguiamo i seguenti compiti:

1. Modifica così il file `host.conf`

```
$ sudo nano /etc/host.conf

order hosts,bind
multi on
```

1. Modifica hostname completo (FQDN)

```
$ sudo hostnamectl set-hostname raspberrypi.scuola.lan
```

1. Modifica file `hosts`

```
$ sudo nano /etc/hosts

127.0.0.1       localhost
127.0.1.1       raspberrypi.scuola.lan  raspberrypi
```

Riavvia il Raspberry e poi controlla il risultato con:

```
$ hostname --short
raspberrypi

$ hostname --domain
scuola.lan

# hostname --fqdn
raspberrypi.scuola.lan
```

## 24.2 Installare Postfix

L'installazione è (come al solito) una riga di codice:

```
$ sudo apt install postfix
```

Al termine dell'installazione c'è una fase iniziale di configurazione in cui verranno poste due domande:

1. il target del sistema di posta: selezionare **INTERNET**
2. il nome di host del sistema di posta: inserire il nome di dominio. Nel nostro esempio: **scuola.lan**.

Fatto questo vanno configurati alcuni file per il nostro caso specifico:

```
$ sudo cp /etc/postfix/main.cf /etc/postfix/main.cf.BACKUP
$ sudo nano /etc/postfix/main.cf

smtpd_banner = $myhostname ESMTP $mail_name (Raspbian)
biff = no

# appending .domain is the MUA's job.
```

(continues on next page)

(continua dalla pagina precedente)

```
append_dot_mydomain = no
readme_directory = no

# defaults to 2 on fresh installs
compatibility_level = 2

# TLS parameters
smtpd_use_tls=no

# general
myhostname = raspberrypi.scuola.lan
mydomain = scuola.lan

alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases

mydestination = $mydomain, $myhostname, localhost
relayhost =
mynetworks = 127.0.0.0/8 172.25.37.0/24
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = ipv4

# use Maildir instead of mbox
home_mailbox = Maildir/
```

Sistemate i valori delle variabili `myhostname`, `mydomain` e `mynetworks` in base alle vostre necessità.

**Avvertimento: MAILDIR vs BOX**

Nell'ultima riga del file abbiamo impostato il sistema Maildir di gestione della casella di posta invece del metodo di default chiamato mbox.

In questo modo il sistema sistemerà la posta degli utenti nella cartella *Maildir* di ogni home, con evidenti vantaggi per l'amministratore (basta creare un utente per assicurargli anche una casella di posta) e per la sicurezza (nessun file esterno alla propria home a cui dover accedere).

Fatto questo siamo pronti per il primo step, l'avvio dell'MTA Postfix:

```
$ sudo systemctl start postfix
$ sudo systemctl status postfix
```

## 24.3 Aggiungere utenti

Ogni utente che aggiungeremo al sistema operativo che ospita l'MTA avrà una casella di posta della forma *user@scuola.lan*.

Nei nostri test a scuola io aggiungo di solito una ventina di utenti con nome utente e password uguali a *studXX* con XX che va da 01 a 20 (o 25, o 30, a seconda della quinta...).

Per fare un esperimento che ha senso occorre aggiungere almeno due utenti. Per farlo decidete i nomi e poi eseguite per ogni utente l'utitlity **adduser** come amministratore, così:

```
$ sudo adduser NOME_UTENTE_DA_CREARE
```

Poi rispondete a tutte le domande che vengono poste. Oppure saltatele tutte premendo INVIO, meno quelle sulla password (da inserire 2 volte).

Tutto qui!

## 24.4 Server POP e IMAP

Per i server POP e IMAP si usa spesso la soluzione modulare **dovecot**, un software che contiene come moduli tutti i software di supporto ad un MTA. A noi servono i server POP e IMAP e l'installazione è semplice come al solito.

```
$ sudo apt install dovecot-pop3d dovecot-imapd
```

La configurazione di entrambi i moduli si basa su pochi file che dobbiamo modificare per il funzionamento *classico* che ci interessa.

Primo file, il file di configurazione principale *dovecot.conf* che va impostato per accettare tutte le connessioni in ingresso:

```
$ sudo nano /etc/dovecot/dovecot.conf

listen = *
```

Secondo file, quello che specifica quale tipo di contenitore di posta utilizza l'MTA

```
$ sudo nano /etc/dovecot/conf.d/10-mail.conf

mail_location = maildir:~/Maildir
```

Terzo e ultimo file, quello che configurare l'accesso senza cifratura

```
$ sudo nano /etc/dovecot/conf.d/10-auth.conf

disable_plaintext_auth = no
auth_mechanisms = plain login
```

Salvato tutto, basta avviare e controllare:

```
$ sudo systemctl start dovecot
$ sudo systemctl status dovecot
```

## 24.5 Mail Test

Per fare un test approfondito dell'ambaradan che abbiamo messo su occorrerebbe testare il sistema con almeno 3 MUA (Mail User Agent), di cui almeno 2 configurati per la ricezione con IMAP (per testare la possibilità di ritrovare la mail in entrambi) e almeno uno con POP verificando successivamente con uno dei client IMAP che la posta è effettivamente scomparsa.

Si può inoltre provare anche l'esperienza **Telnet e Mail** da qualche parte in questo stesso sito.

Per quanto riguarda i MUA provo a suggerirvi alcuni software da testare da soli:

- le web applications `squirrelmail` (http://squirrelmail.org) oppure `rainloop` (https://rainloop.net). Necessitano entrambe di un server web con PHP. Sono entrambe forzatamente client IMAP
- l'applicazione `E-Mail` dello smartphone, che contiene sia un client IMAP che POP, ma che è chiaramente ottimizzata per un utilizzo con IMAP.
- l'applicazione desktop `Mozilla Thunderbird` (https://www.thunderbird.net/) che ovviamente contiene sia un client IMAP che POP.

**Buon divertimento!**

CAPITOLO 25

# Telnet e Mail

**Nota:** Prerequisti: **OS, terminale**

Argomenti trattati: **Telnet, SMTP, POP, IMAP**

## 25.1 Introduzione

Telnet è un piccolo client da terminale, disponibile su ogni sistema operativo, che serve per simulare connessioni in chiaro a qualsiasi socket desideriamo. La sua utilità sta nel fatto che permette di *dialogare* con qualunque protocollo del livello superiore che utilizza testo semplice per le sua sintassi.

A pensarci bene, tutti i protocolli del livello superiore che abbiamo studiato (o che studieremo) utilizzano dati codificati in ASCII, quindi... telnet può essere un semplice strumento per provare ad analizzarli!

La sua sintassi è semplicissima:

```
$ telnet HOST PORT
```

Da dopo l'avvenuta connessione bisogna scrivere.. nella lingua del servizio con cui vogliamo dialogare, ovvero bisogna usare le specifiche del protocollo!

## 25.2 Telnet e SMTP

Per estrema semplicità simulerò semplicemente una connessione ad un MTA generico, di cui elenco le caratteristiche:

- host: **scuola.lan**
- porta: **25**
- user1: **pippo**
- pass1: **attentialcane**

- user2: **ciccio**
- pass2: **pescegatto**

Tramite telnet l'utente *pippo* proverà ad inviare una mail all'utente *ciccio*.

```
$ telnet scuola.lan 25

... risposta ...

HELO pippo

... risposta ...

MAIL FROM: pippo@scuola.lan

... risposta ...

RCPT TO: ciccio@scuola.lan

... risposta ...

DATA

... risposta ...

Subject: Titolo
Una mail con testo qualsiasi che termina quando
andate a capo, scrivete un punto e riandate a capo.
.

... risposta ...

QUIT
```

Se avete scritto tutto bene (lo capite osservando le risposte) avete inviato una mail scrivendo *a mano* le istruzioni SMTP per l'MTA!!! Avete notato che non avete mai inserito la password???

## 25.3 TELNET e IMAP

Adesso proviamo a consultare la mail dell'utente *ciccio* per trovare la mail che ha ricevuto. Facciamo prima con IMAP così il messaggio rimane sul server e proviamo successivamente provare con POP ;)

```
$ telnet scuola.lan 143

... risposta ...

a1 LOGIN ciccio pescegatto

... risposta ad a1 ...

a2 LIST "" "*"

... risposta ad a2 ...

a3 EXAMINE INBOX
```

(continues on next page)

(continua dalla pagina precedente)

```
... risposta ad a3 ...

a4 FETCH 1 BODY[]

... risposta ad a4 ...

a5 LOGOUT
```

Ecco qua! Avanti...

## 25.4 TELNET e POP3

Adesso proviamo a consultare la mail dell'utente *ciccio* con POP, cancellando il messaggio alla fine della consultazione.

```
$ telnet scuola.lan 110

... risposta ...

USER ciccio

... risposta ...

PASS pescegatto

... risposta ...

LIST

... risposta ...

RETR 1

... risposta ...

DELE 1

... risposta ...

QUIT
```

Come avete intuito leggendo i comandi, vi siete connessi al server POP con le credenziali di *ciccio*, avete elencato i suoi messaggi, avete letto (e poi cancellato) il messaggio numero 1.