Introduzione al Networking

Guida introduttiva alla configurazione di rete in ambiente GNU/Linux

Valeria Mazzola
mazzolavale1@gmail.com

Corsi GNU/Linux Avanzati 2017

POLITECNICO OPEN
unix LABS
Come hack with us.

Sezione 1

Di cosa parliamo oggi

- Come funziona l'internet[1]
- Come configurare un computer GNU/Linux in rete
- Alcuni strumenti di debugging

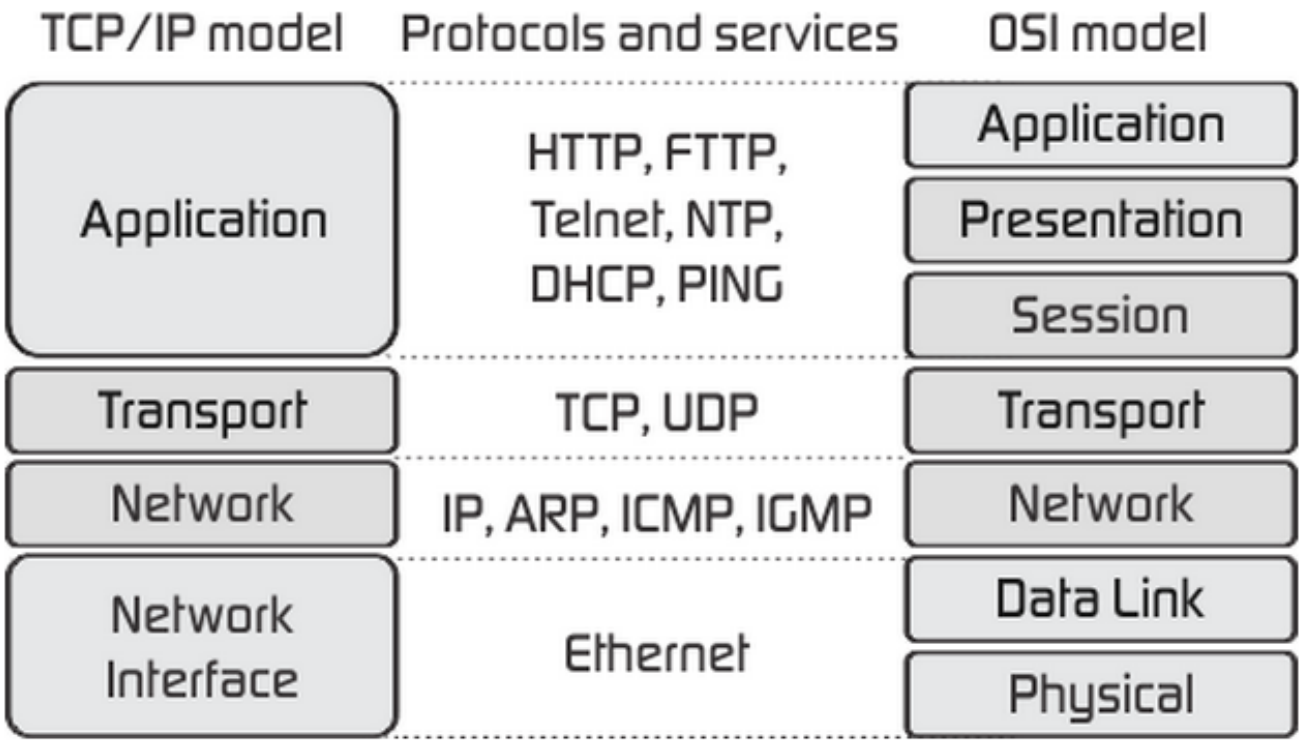
[1] o almeno una parte dei concetti fondamentali :)

Sezione 2

I modelli ISO/OSI e TCP/IP

- Le comunicazioni di rete sono gestite da molteplici protocolli, che operano a diversi livelli
- Ogni livello ha degli specifici obiettivi
- I protocolli di rete sono "impilati" logicamente uno sopra l'altro, quelli superiori fanno affidamento su quelli inferiori
- Insieme formano il cosiddetto **stack di rete**

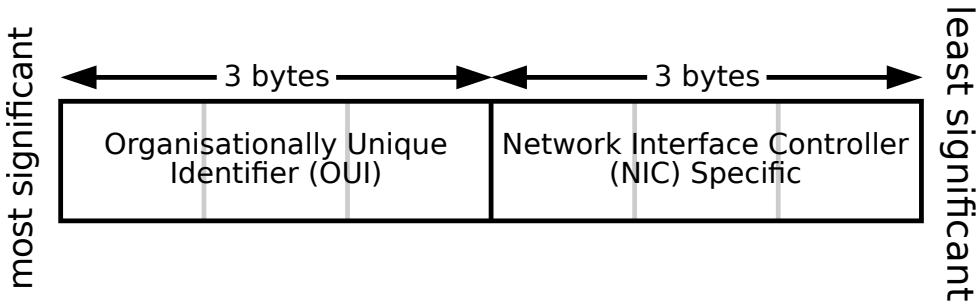
I modelli ISO/OSI e TCP/IP



Layer 1: Physical

Layer 2: Data Link - MAC

- Il livello 2 è il livello di collegamento dati
- Al livello 2 vive il protocollo **Ethernet**:
  - Permette alle schede di rete di comunicare dei dati *in modo affidabile*
  - Gestisce eventuali errori e collisioni dei dati sul mezzo fisico (cavi, onde elettromagnetiche, piccioni[2]...)
- Una scheda di rete è identificata da un indirizzo *univoco* Ethernet (**MAC[3] Address**) formato da 6 byte



[2] https://it.wikipedia.org/wiki/IP_over_Avian_Carriers
[3] Medium Access Control

Layer 3: Network - IP

- Il livello 3 è il livello di rete
- Il layer di IP permette ai nostri pacchetti di essere spediti correttamente verso la destinazione (purtroppo senza *sani e salvi*)
- I pacchetti IP sono instradati lungo la rete e arrivano all'host...
- ...oppure muoiono provandoci

Layer 3: Network - IP

- Ogni macchina sulla rete può avere uno o più indirizzi IP
- Un indirizzo IPv4 ha l'aspetto di quattro numeri decimali separati da punti: 94.142.241.111
- Un indirizzo IP è composto da due pezzi: il primo pezzo indica la rete, il secondo pezzo indica l'host all'interno della rete
- I due pezzi sono separati da una subnet mask di lunghezza variabile (**VLSM: Variable Length Subnet Mask**)

```
IP address:    192      168      0        2
              11000000 10101000 00000000 00000010
Netmask:      11111111 11111111 11111111 00000000
                 255      255      255        0
              +--------------------------+--------+
                        Network             Host
```

Layer 3: Routing

- Host sulla nostra stessa sottorete sono immediatamente raggiungibili
- Per raggiungere gli host esterni dobbiamo mandare i pacchetti al **gateway** che provvederà ad instradarli al di fuori della rete per conto nostro[4]
- Il gateway si occupa di instradare i pacchetti sulla base della propria **routing table**
- Vedremo come configurare le rotte, con particolare attenzione al **default gateway**

[4] il gateway deve risiedere nella nostra rete locale, altrimenti ci è impossibile contattare la rete esterna

Layer 4: Trasporto

- Il livello 4 è il livello di trasporto
- Il livello 4 permette connessioni multiple tra due stessi host per mezzo di **porte** (0-65535)
- Un'applicazione si connette ad un host remoto utilizzando *indirizzo IP dell'host remoto e la porta sulla quale connettersi*
- Molte applicazioni a lato server utilizzano una porta standard (esempio ssh 22, infatti non c'è bisogno di specificarla)
- I principali protocolli di trasporto sono TCP e UDP

Layer 4: Trasporto - TCP e UDP

- TCP consente di rendere la comunicazione **affidabile**, ovvero garantisce che i dati che inviamo arrivino sani e salvi dall'altra parte
  - Per farlo utilizza messaggi conferma (ack) della ricezione dei pacchetti e in caso di perdita si occupa del reinvio
- UDP è un protocollo di trasporto più semplice, che cerca di portare i dati dall'altra parte il più velocemente possibile, anche a costo di perderli
  - Per esempio molto spesso il DNS utilizza UDP

Layer 4: Trasporto - TCP e UDP

- TCP consente di rendere la comunicazione **affidabile**, ovvero garantisce che i dati che inviamo arrivino sani e salvi dall'altra parte
  - Per farlo utilizza messaggi conferma (ack) della ricezione dei pacchetti e in caso di perdita si occupa del reinvio
- UDP è un protocollo di trasporto più semplice, che cerca di portare i dati dall'altra parte il più velocemente possibile, anche a costo di perderli
  - Per esempio molto spesso il DNS utilizza UDP

Sezione 3

Network Administration/Level Debugging

- Tutto lo stack di rete su linux è basato sulle **interfacce**
  - *Reali*, cioè quelle realmente esistenti nella nostra macchina (*eth0*, *wlan0*, *enp...*, *wlp...*)
  - *Virtuali*, generate via software per emulare quelle vere (*tun*, *tap*, virtual devices)
  - *Bridge*, unione (virtuale) di più interfacce
- I tools che vedremo ora utilizzano le interfacce allo stesso modo, indipendentemente dal tipo
- Lo stesso vale per il firewalling (prossima lezione)

Network Administration/Level Debugging

- Tutto lo stack di rete su linux è basato sulle **interfacce**
  - *Reali*, cioè quelle realmente esistenti nella nostra macchina (*eth0*, *wlan0, enp..., wlp...*)
  - *Virtuali*, generate via software per emulare quelle vere (*tun*, *tap*, virtual devices)
  - *Bridge*, unione (virtuale) di più interfacce
- I tools che vedremo ora utilizzano le interfacce allo stesso modo, indipendentemente dal tipo
- Lo stesso vale per il firewalling (prossima lezione)

iproute2

- Per il debugging/configurazione delle interfacce ai livelli 2 e 3 si usa la suite **iproute2**
- Il nuovo tool **ip** sostituisce i vecchi net-tools[5] per operare sulle schede di rete
  - Performance migliori per l'utilizzo di **netlink**
  - Comandi più semplici

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

  - Feature di rete più recenti meglio implementate in un nuovo e unico tool

[5] ifconfig, route, netstat, e i tool presenti nel pacchetto *net-tools* sono a tutti gli effetti deprecati

iproute2

- Per il debugging/configurazione delle interfacce ai livelli 2 e 3 si usa la suite **iproute2**
- Il nuovo tool **ip** sostituisce i vecchi net-tools[5] per operare sulle schede di rete
  - Performance migliori per l'utilizzo di **netlink**
  - Comandi più semplici

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

  - Feature di rete più recenti meglio implementate in un nuovo e unico tool

[5] ifconfig, route, netstat, e i tool presenti nel pacchetto *net-tools* sono a tutti gli effetti deprecati

iproute2: link layer

- mostra le interfacce con i loro MAC address e lo stato UP/DOWN

  ```
  $ ip link show
  ```

- permette di accendere o spegnere un'interfaccia

  ```
  $ ip link set dev <interface> <up|down>
  ```

- consente di cambiare il MAC address dell'interfaccia con uno che preferiamo...

  ```
  $ ip link set dev <interface> address <MAC address>
  ```

iproute2: link layer

- È possibile anche modificare manualmente il contenuto della tabella **ARP**[6], che associa i MAC address agli indizzi IP dei corrispondenti host
- Per visualizzare la tabella

```
$ ip neigh show
```

- Aggiunge una riga alla tabella

```
$ ip neigh add <IP address> lladdr <MAC address>
  dev <device>
```

- Modifica una riga della tabella

```
$ ip neigh change <IP address> lladdr <MAC address>
  dev <device>
```

[6] Address Resolution Protocol

iproute2: network layer

- Per mostrare le interfacce di rete con i loro indirizzi IP, se li hanno

```
$ ip address show
```

- Per aggiungere un indirizzo ad un'interfaccia

```
$ ip addr add <IP address>/<netmask length>
  dev <device>
```

- La netmask length è il numero di bit dedicati alla sottorete, ad esempio:

```
$ ip addr add 192.168.1.42/24 dev eth0
```

- Per rimuovere un indirizzo associato a un'interfaccia

```
$ ip addr del <IP address>/<netmask length>
  dev <device>
```

iproute2: routing

- Come detto prima, per raggiungere host esterni alla nostra sottorete dobbiamo passare per il **gateway**
- Per mostrare le rotte attualmente impostate

```
$ ip route show
```

- Per aggiungere una rotta per un insieme di indirizzi (e volendo anche specificare l'interfaccia d'uscita)

```
$ ip route add <address>/<netmask length>
  via <gateway address> [via <interface>]
```

iproute2: routing

- Si può specificare esplicitamente il default gateway

```
$ ip route add default via <gateway address>
  [via <interface>]
$ ip route add 0.0.0.0/0.0.0.0 via <gateway address>
  [via <interface>]
```

- Per rimuovere una rotta

```
$ ip route del <address>/<netmask length>
  via <gateway address>
```

moar on routing

- Volendo, ogni macchina GNU/Linux può essere configurata per comportarsi da router
- È sufficiente abilitare l'**ip forwarding**

```
$ sysctl -w net.ipv4.ip_forward=1
$ echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Così facendo il pc, oltre che inviare i propri pacchetti al gateway, cercherà di occuparsi di instradare i pacchetti altrui

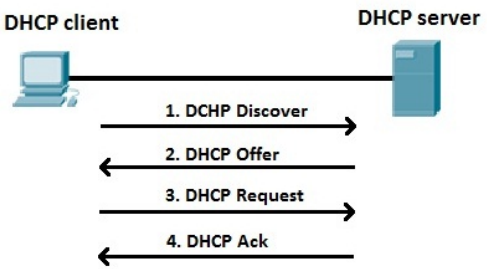
DHCP

- What if it just worked?
- Il DHCP è un protocollo che consente di assegnare indirizzi IP alle interfacce in modo automatico
  - utile se ci sono tanti host che si collegano e discollegano di continuo
  - permette di gestire facilmente gli ip della rete

DHCP

- What if it just worked?
- Il DHCP è un protocollo che consente di assegnare indirizzi IP alle interfacce in modo automatico
  - utile se ci sono tanti host che si collegano e discollegano di continuo
  - permette di gestire facilmente gli ip della rete

DHCP



- Per utilizzare il DHCP è necessario che sulla rete ci sia un *DHCP server*
- Il server, su richiesta, rilascia un ip per un certo lasso di tempo (**lease**)
- I client dhcp si trovano solitamente preinstallati (dhclient, dhcpcd)
  - per utilizzare dhcp su di un'interfaccia si usa «dhcp_client» «nome_interfaccia»
- Se correttamente configurato il dhcp provvederà a fornire tutte le informazioni necessarie per il corretto funzionamento della rete (IP address, default gateway, netmask, broadcast, dns servers)

Sezione 4

Ping

- Il comando **ping** serve per sapere se due host riescono a parlarsi a vicenda
- Di fatto utilizza il protocollo ICMP[7]

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

[7] https://tools.ietf.org/html/rfc792

Ping

- ICMP prevede l'invio di una **echo request** seguita da una **echo reply**, oppure da un codice di errore
- Se i due host non comunicano, ping ci dà degli indizi su dove può essere il problema
  - Network unreachable: il pacchetto non viene instradato correttamente verso la subnet di destinazione
  - Host unreachable: il pacchetto arriva nella subnet di destinazione, ma non trova l'host giusto

Netcat

- **Netcat** è un client/server TCP da linea di comando

```
$ nc <hostname> <port>
$ nc -l -p <port>
```

- *-l -p <porta>* per restare in ascolto per connessioni sulla porta data
- È possibile anche usare netcat su UDP con l'opzione *-u*

Tcpdump

- Quando qualcosa non funziona e non sappiamo perché, è utile dare un'occhiata a "cosa passa davvero sul cavo"

```
# tcpdump -i <interface>
```

- È possibile specificare dei filtri (più o meno complessi) per visualizzare solo i pacchetti che ci interessano

```
# tcpdump host snowball
# tcpdump 'tcp port 80 and (((ip[2:2] -
((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)'[8][9]
```

[8] Esempio copiato direttamente dal manuale di tcpdump.

[9] To print all IPv4 HTTP packets to and from port 80, i.e. print only packets that contain data, not, for example, SYN and FIN packets and ACK-only packets.

Netstat

- Spesso usato per visualizzare informazioni sui socket presenti nel sistema
- Visualizza tutti i socket TCP in ascolto, e le applicazioni corrispondenti

```
$ netstat -ltp
```

- Visualizza tutti i socket UDP in ascolto, e le applicazioni corrispondenti

```
$ netstat -lup
```

- Funzione simile è svolta dalla più recente utility **ss** (socket stat)

Per esercitarsi: netkit

- Per esercitarsi a effettuare configurazioni di rete con Linux si può usare **netkit**[10]
- Utilizza macchine virtuali basate su *User Mode Linux*
- Ogni macchina virtuale può essere configurata proprio come una macchina reale per ottenere setup di rete anche piuttosto complessi

- Una versione più aggiornata di netkit è **netkit-ng**
- Utilizza macchine virtuali con una versione di Debian più recente
- Differisce da netkit per alcune features[11]

[10] http://wiki.netkit.org

[11] https://netkit-ng.github.io/

Per esercitarsi: netkit

- Per esercitarsi a effettuare configurazioni di rete con Linux si può usare **netkit**[10]
- Utilizza macchine virtuali basate su *User Mode Linux*
- Ogni macchina virtuale può essere configurata proprio come una macchina reale per ottenere setup di rete anche piuttosto complessi

- Una versione più aggiornata di netkit è **netkit-ng**
- Utilizza macchine virtuali con una versione di Debian più recente
- Differisce da netkit per alcune features[11]

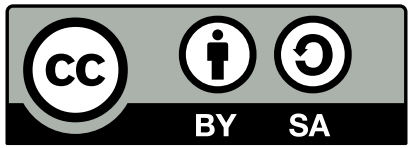
[10] http://wiki.netkit.org
[11] https://netkit-ng.github.io/

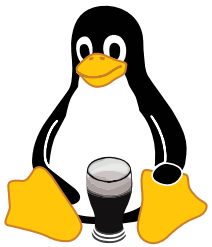
Fonti e approfondimenti

- Tutte le slide sul networking degli anni passati, sparse su poul.org
- Unix and Linux System Administration Handbook, Fourth Edition - by Ben Whaley, Trent R. Hein, Garth Snyder, Evi Nemeth
- Manpages
- https://wiki.gentoo.org/wiki/.../Networking
- http://www.penguintutor.com/linux/basic-network-reference
- Le slide di otacon22 su IPv6

Grazie

Grazie per l'attenzione!





POLITECNICO OPEN
unix LABS
Come hack with us.