Marionnet

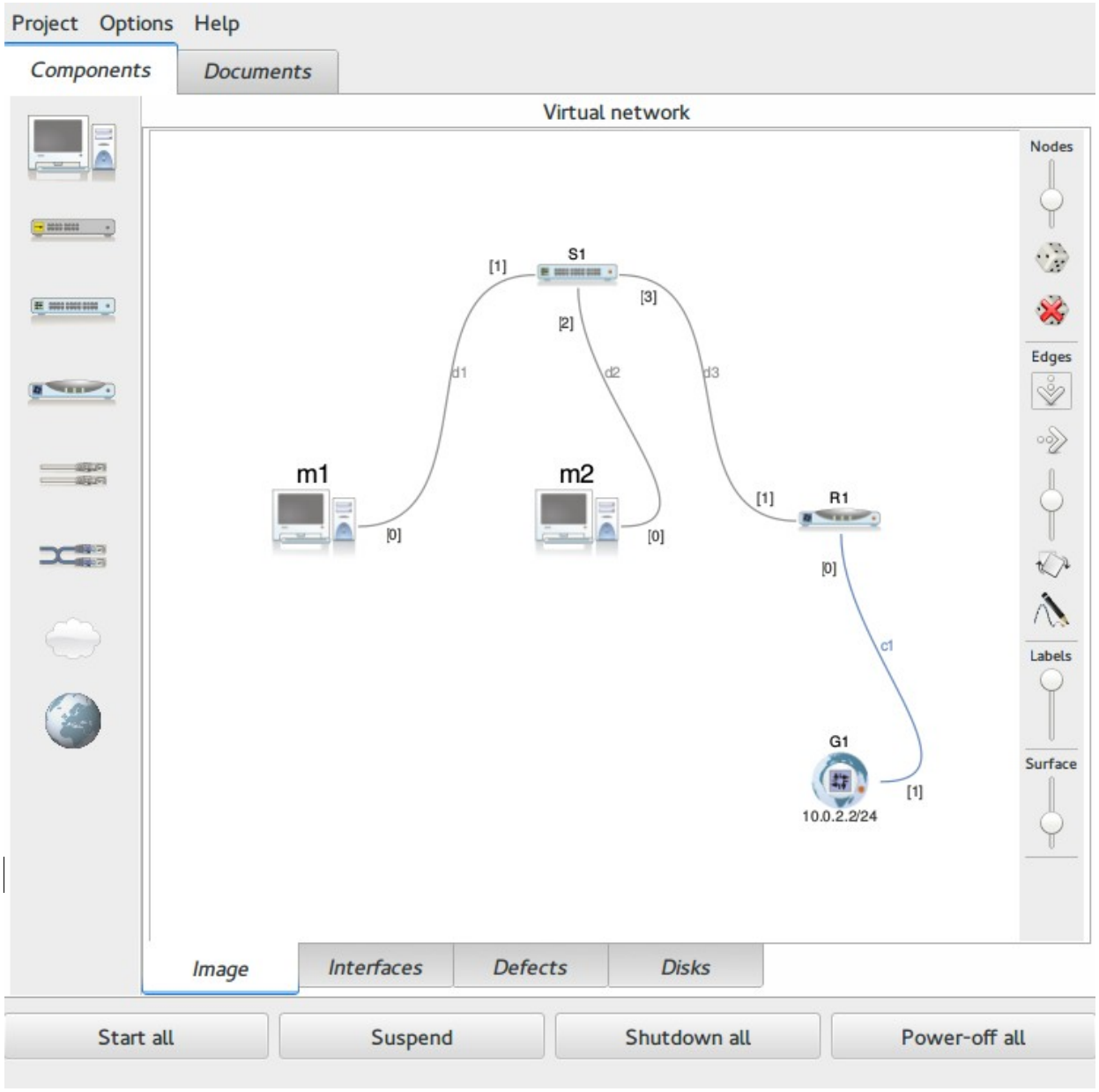
Marionnet è un software per la simulazione di reti di calcolatori.

Offre un frontend grafico per facilitare il deployment delle reti, basato su UML ed altri tools (ad esempio, vde e wirefilter), includendo:

- host di rete
- hub e switch
- router
- gateway Internet

Durante il corso useremo Marionnet per definire reti complesse, e configureremo i componenti delle reti in modo del tutto simile a reti reali.

http://www.marionnet.org

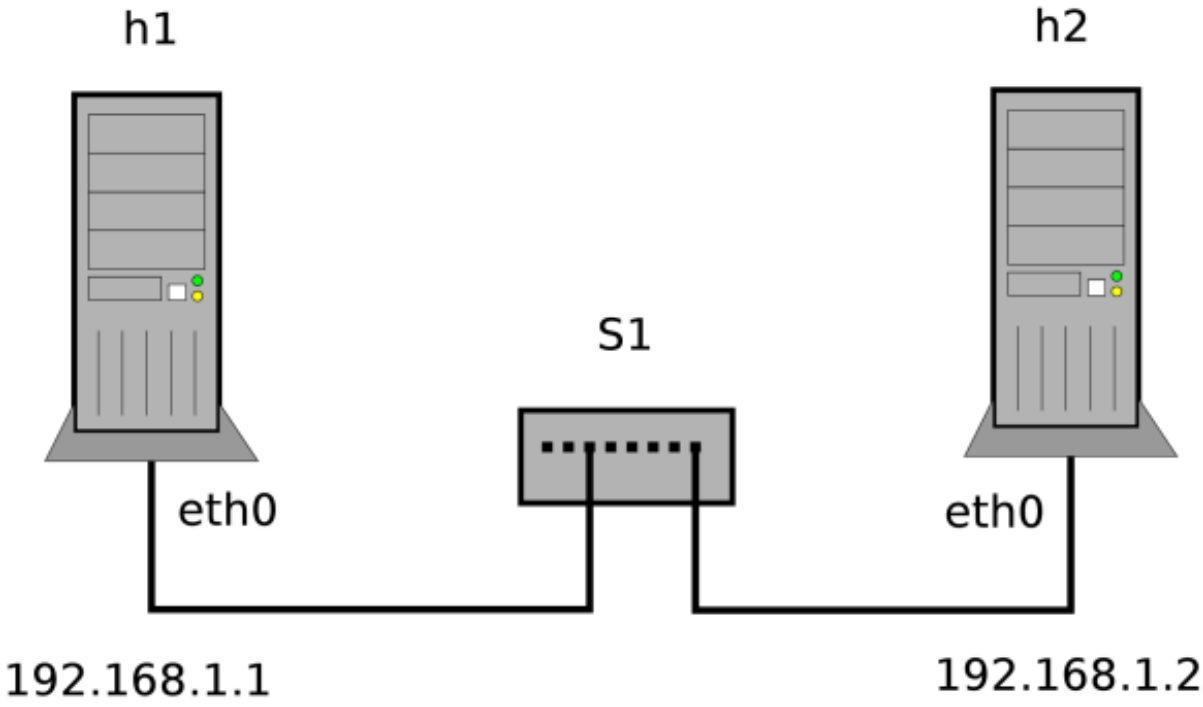


Note: utilizzo Marionnet durante il corso

- Utilizzando un computer del laboratorio al FIM (vedere slide introduttive corso per collegamento remoto):
  - Sono presenti due sistemi: quello più aggiornato e consigliato è **aapar2**, mentre quello chiamato **apar1**
- Utilizzando un sistema locale scaricabile dai link presenti su Moodle:
  - **Si consiglia/raccomanda l'uso della macchina virtuale**
    - Ci sono diversi formati di dischi a seconda del sistema del sistema di virtualizzazione
    - In questo caso è presente solo il sistema **aapar2**
  - A vostro rischio e pericolo, è presente un archivio per installare **aapar2** su un sistema nativo linux (non documentato, potete provare a ispezionare gli script per capirne il funzionamento)
- **Nota:** attualmente i due sistemi **apaar2** presenti nei laboratori e nelle macchine virtuali non sono compatibili fra di loro (non si può aprire un progetto salvato con una macchina virtuale tramite il sistema del laboratorio, e viceversa)

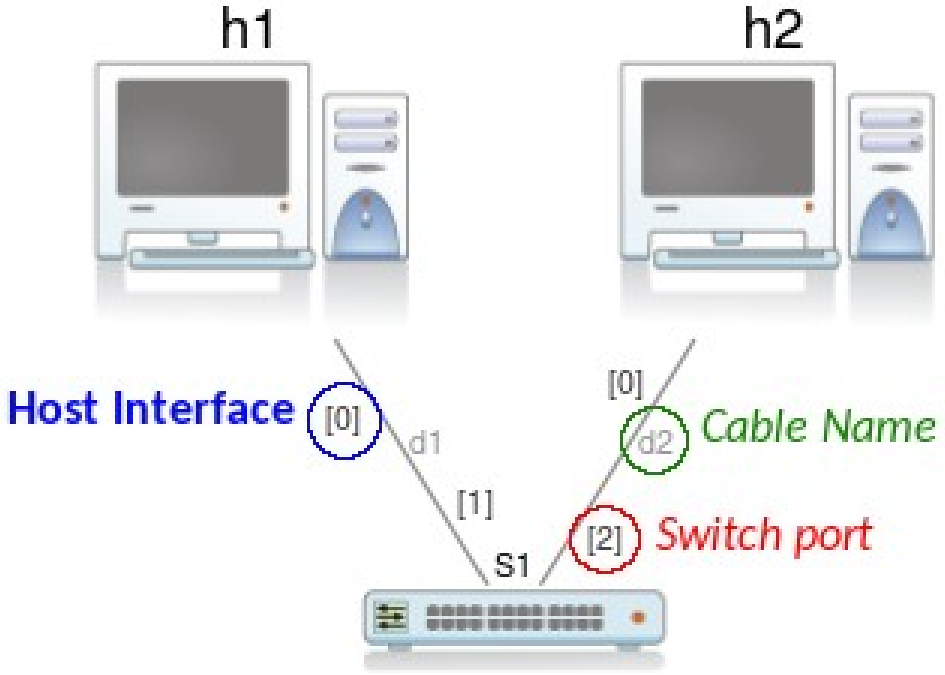
Prima configurazione di rete

- Collegare in rete due nodi tramite uno switch.



- **Obiettivo**: far comunicare i due nodi utilizzando gli indirizzi IP e gli hostname mostrati in figura

Creazione della rete con Marionnet

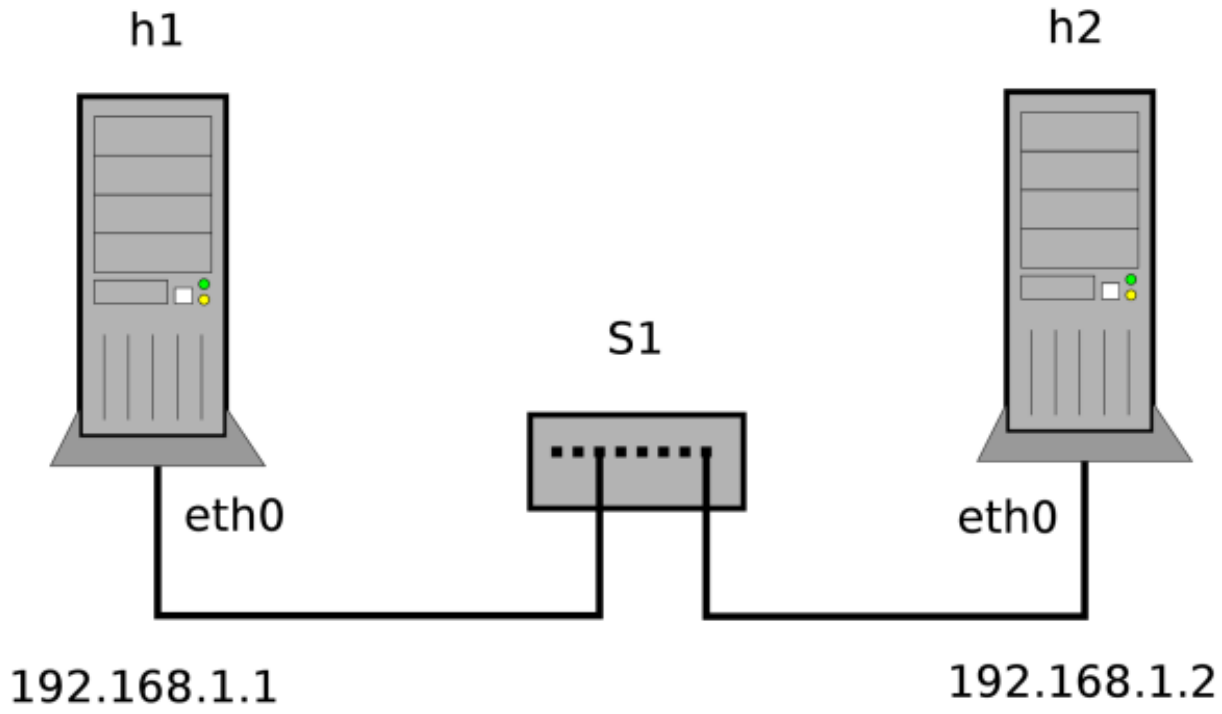


Nota: attenzione alla tipologia dei cavi utilizzati

- ***cavi incrociati (crossover)***: connessioni fra dispositivi di rete dello stesso livello dello stack (e.g., switch-switch, host-host)
- ***cavi dritti (straight)***: connessioni fra dispositivi di livello differente (e.g., switch-host)

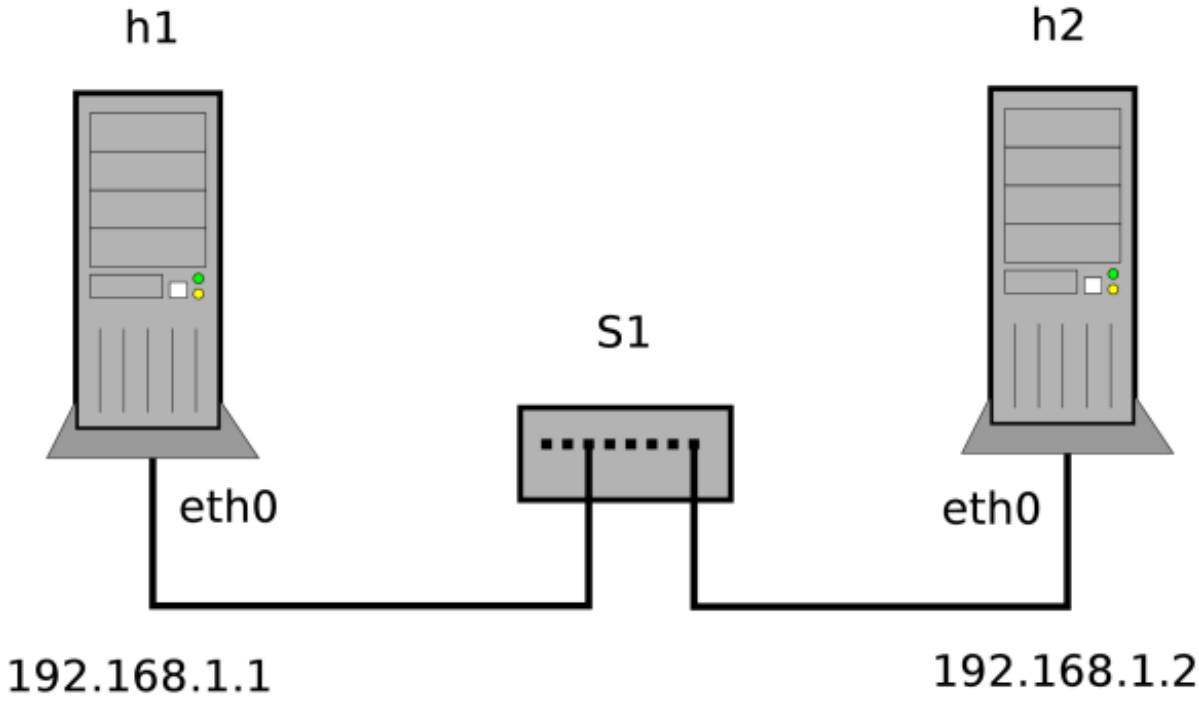
Obiettivi esercitazione

1) **Configurazione della rete**
2) Test di connettività senza configurare i nodi



Configurazione della rete

1) Configurazione dei nodi:
   - impostazione del proprio hostname
   - configurazione della corrispondenza hostname/indirizzo IP degli altri host della rete
   - configurazione del proprio indirizzo IP
   - test di connettività



Hostname

- Per assegnare temporaneamente l'hostname usare il comando:

  `hostname <nome>`

  (invocato senza parametri stampa in output l'hostname attuale)

- Per assegnare in modo permanente l'hostname della macchina che stiamo configurando, editare il file **`/etc/hostname`** in modo che contenga il nome del nodo

Nota Marionnet: all'avvio di un nodo Marionnet configura automaticamente l'hostname impostato nell'interfaccia grafica, per cui ogni modifica effettuata sul file `/etc/hostname` verrà sovrascritta

Nomi degli host della rete

- Inserire le giuste informazioni di rete per risolvere i nomi
  - si agisce sul file **`/etc/hosts`**
  - nel nostro caso:

```
127.0.0.1       localhost
192.168.1.1     h1
192.168.1.2     h2
```

Note:

- ricordare che il file viene letto dal sistema dall'alto al basso, per cui ogni conflitto viene risolto dando precedenza alla **prima** corrispondenza trovata
- le modalità e l'ordine in cui vengono risolti gli hostname è gestito dal ***Name Service Switch*** (`/etc/nsswitch.conf`)

Impostazioni interfaccia di rete: Configurazione permanente tramite ***interfaces*** (1)

Configurare permanentemente le interfacce di rete del sistema:

1) la configurazione delle interfacce deve essere memorizzata in **/etc/network/interfaces**
   - è possibile anche utilizzare file di configurazione indipendenti in **/etc/network/interfaces.d** (direttiva ***source*** nel file ***interfaces***)
2) nel caso sia in esecuzione il servizio **NetworkManager**, utilizzare il tool **nmcli**
   - sovrascrive le regole presenti in **/etc/network/interfaces**
   - è quasi sempre presente nei sistemi desktop, utilizzato in questo caso tramite un'interfaccia grafica, ma è molto diffuso anche in ambienti server per configurazioni avanzate

Impostazioni interfaccia di rete: Configurazione permanente tramite ***interfaces*** (2)

- Direttiva **`iface`** nel file /etc/network/interfaces
  - `<modalità>`
    - `dhcp` = inizializzazione dell'interfaccia in modo automatico col protocollo dhcp
    - `loopback` = interfaccia locale (127.0.0.1)
    - **`static`** = seguono parametri per configurare l'interfaccia
  - **Configurazione statica di un'interfaccia**
    - `address` = indirizzo IP dell'interfaccia (192.168.X.Y)
    - `netmask` = 255.255.255.0
    - `network` = indirizzo della rete (192.168.X.0)
    - `broadcast` = indirizzo di broadcast (192.168.X.255)
    - `gateway` = **default gateway**, da utilizzare nel caso la configurazione di rete lo richieda

Impostazioni interfaccia di rete: Configurazione permanente tramite ***interfaces*** (3)

```
# File di configurazione /etc/network/interfaces

auto eth0

iface eth0 inet static
        address 192.168.1.1
```

Impostazioni interfaccia di rete: Applicazione delle modifiche permanenti con *ifupdown*

All'avvio del sistema, se non è presente la direttiva **auto** nei file di configurazione opportuni, l'interfaccia di rete sarà spenta.

Per attivare l'interfaccia configurandola utilizzando i file di configurazione, utilizzare i comandi **ifup** e **ifdown:**

```
# ifdown <iface>
```
← disattiva l'interfaccia

```
# ifup <iface>
```
← attiva l'interfaccia

Ad esempio:

```
# ifdown eth0
# ifup eth0
# ifdown eth0 && ifup eth0
```

Note:

- l'opzione **-a** permette di agire su tutte le interfacce *configurate*.
- in caso di errori nei file di configurazione, il comando **ifup** fallirà

Impostazioni interfaccia di rete: Consultare la configurazione attuale

Usare il comando **`ifconfig`** (deprecato, ma usato ancora spesso)

```
$ ifconfig [-a] [<name>]
```

Se non viene specificata nessuna interfaccia, vengono mostrate tutte le interfacce attive. Usando l'opzione **`-a`** vengono visualizzate anche le interfacce non attive.

Oppure **`ip`** (suite ***iproute2***):

```
$ ip addr show [dev <name>]
```

NOTA: per controllare lo stato dei un'interfaccia (così come di qualsiasi altra configurazione), non controllare il file della configurazione!

6. Verificare il funzionamento

Per verificare la connettività fra due host esistono diverse possibilità che sfruttano diversi protocolli ai vari livelli dello stack TCP/IP.

Il più conosciuto è **`ping`**, che utilizza il protocollo di supporto **ICMP**, quindi di default si basa sul **livello 3** (ci torneremo in dettaglio in futuro).

Un altro tool disponibile su Linux è **`arping`**, che verifica la presenza di un indirizzo IP a **livello 2** (H2N) per mezzo di richieste **ARP**.

arping - esempio

```
# arping -i eth0 192.168.1.254

ARPING 192.168.1.254
60 bytes from 00:01:90:ea:af:a1 (192.168.1.2): index=0 time=1.001 sec
60 bytes from 00:01:90:ea:af:a1 (192.168.1.2): index=1 time=1.001 sec
60 bytes from 00:01:90:ea:af:a1 (192.168.1.2): index=2 time=1.001 sec
60 bytes from 00:01:90:ea:af:a1 (192.168.1.2): index=3 time=877.952 msec
60 bytes from 00:01:90:ea:af:a1 (192.168.1.2): index=4 time=949.535 msec
60 bytes from 00:01:90:ea:af:a1 (192.168.1.2): index=5 time=1.001 sec
60 bytes from 00:01:90:ea:af:a1 (192.168.1.2): index=6 time=1.001 sec
60 bytes from 00:01:90:ea:af:a1 (192.168.1.2): index=7 time=1.001 sec
60 bytes from 00:01:90:ea:af:a1 (192.168.1.2): index=8 time=1.001 sec
60 bytes from 00:01:90:ea:af:a1 (192.168.1.2): index=9 time=1.001 sec
```

`ping`

- Invia una successione di pacchetti ICMP **ECHO_REQUEST** e attende la relativa risposta **ECHO_REPLY** *(ci torneremo analizzando il livello 3 dello stack TCP/IP)*
- Misura il tempo che intercorre tra l'invio e la ricezione di ogni pacchetto e riporta semplici statistiche
- Il comando **ping** fa parte della dotazione standard di tutte le macchine sia Unix che Windows anche se con sintassi leggermente diverse.
- L'indirizzo **localhost** viene risolto in un indirizzo IP normale che dall'output è 127.0.0.1
- Ulteriori informazioni: manuale in linea di **ping** (sui sistemi *nix: **man ping**)

Analisi della tabella ARP

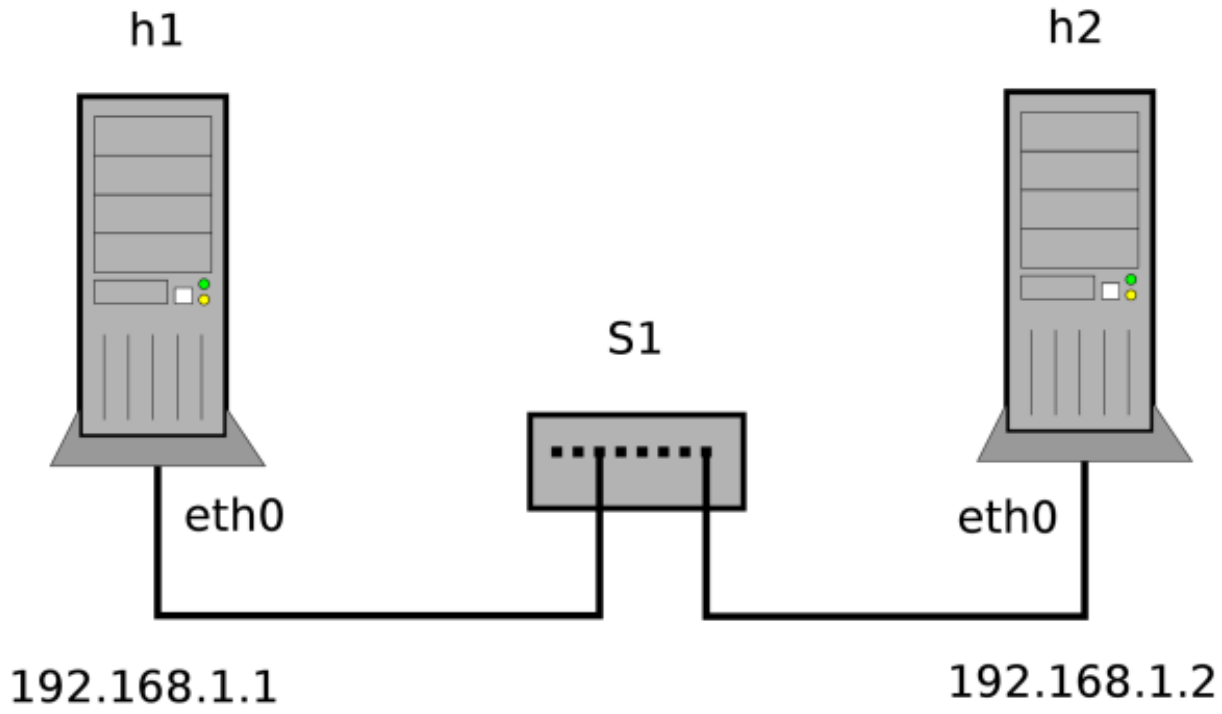
- comandi **arp** e **ip neigh**

```
(h1) $ ping 192.168.1.1
(h1) # arp
Address          HWtype  HWaddress           Flags Mask  Iface
192.168.1.1      ether   02:04:06:64:66:db   C           eth0

(h1) # ip neigh
192.168.1.1 dev eth0 lladdr 02:04:06:64:66:db REACHABLE
```

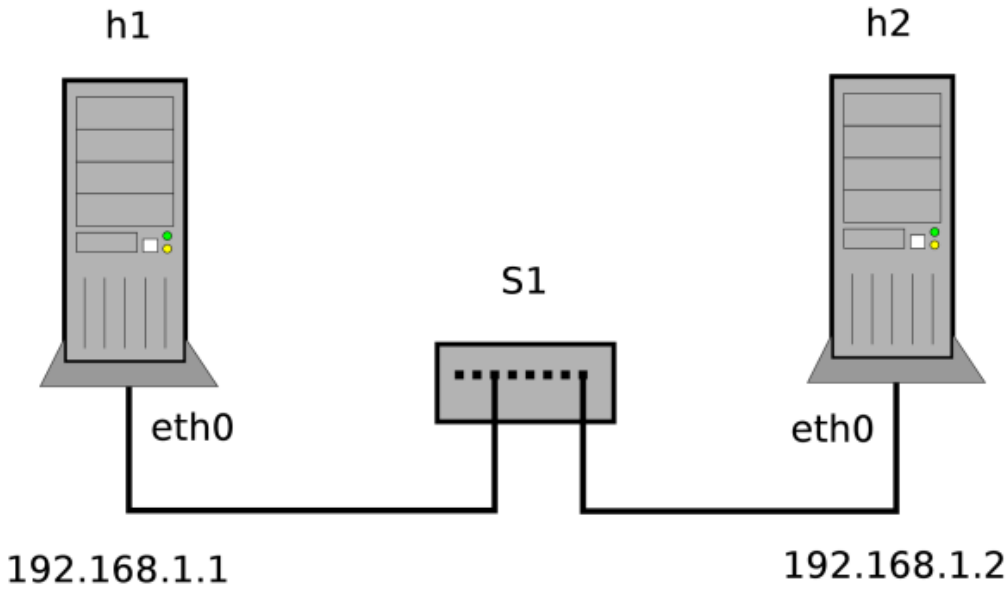
Obiettivi esercitazione

1) Configurazione dei nodi
2) **Test di connettività senza configurare i nodi**



Test connessioni di rete senza configurazione degli indirizzi

1) Test delle connessioni di rete senza configurazione dei nodi:
   - attivare le interfacce di entrambi i nodi senza alcun indirizzo IP
     - generare traffico da un nodo (h1)
     - ascoltare il traffico dall'altro nodo (h2)
   - configurare solo l'indirizzo IP di un nodo (h2)
     - verificare la corretta configurazione dell'indirizzo dall'altro nodo (h1) utilizzando ***arping***



Test connessioni di rete senza configurazione degli indirizzi IP

Non abbiamo assegnato alcun indirizzo IP, quindi non possiamo utilizzare protocolli che si affidano al livello di rete dello stack TCP/IP.

Possiamo utilizzare protocolli del livello **H2N**, in particolare il protocollo **ARP** di **Ethernet**, grazie al tool ***arping***.

Possiamo intercettare (o “sniffare”) il traffico che passa da un'interfaccia di rete tramite il tool ***tcpdump***.

Protocollo ARP

- Le reti utilizzano gli indirizzi IP come elemento identificativo della sorgente dei dati
- Tuttavia a livello più basso le schede di rete ragionano in termini di indirizzi MAC
- Serve un sistema per stabilire una corrispondenza tra IP e MAC address
- Uso di protocolli appositi (ARP = Address Resolution Protocol) per interrogare le schede di rete remote al fine di conoscere l'indirizzo MAC corrispondente ad un dato indirizzo IP
- Uso di una **cache** locale per conservare le risoluzione ARP fatte di recente

Test connettività senza configurazione degli indirizzi

```
(h1) # ifconfig eth0 up
(h1) # arping -0Bi eth0
```

← attiviamo l'interfaccia ***eth0*** di **h1**
← generiamo richieste ***arp***

```
ARPING 255.255.255.255
Timeout
Timeout
Timeout
Timeout
Timeout
```

```
(h2) # ifconfig eth0 up
(h2) # tcpdump -ni eth0 arp
```

← attiviamo l'interfaccia ***eth0*** di ***h2***
← "sniffiamo" il traffico ***arp*** su ***eth0***

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
15:45:20.071611 ARP, Request who-has 255.255.255.255 tell 0.0.0.0, length 28
15:45:21.080772 ARP, Request who-has 255.255.255.255 tell 0.0.0.0, length 28
```

Note:
- utilizzare ***tcpdump*** con l'opzione **-e** per visuallizare anche gli indirizzi MAC.
- utilizzare ***tmux*** per effettuare il multiplexing di più terminali

arping

 # arping [-0] [-i <iface>] {-B,<ip_addr>}

- Di default, invia una successione di richieste ARP (in broadcast) per un determinato indirizzo IP.
- **NB**: il protocollo ARP agisce a livello 2, quindi arping permette di raggiungere solo gli host che fanno parte dello stesso dominio di broadcast
- Funziona anche senza avere configurato l'indirizzo IP dell'interfaccia (ma l'interfaccia deve comunque essere attiva):
  - in questo caso utilizzare il parametro opzionale **-0**
- Altre opzioni:
  - **-B**: esegue ARP cercando di risolvere l'indirizzo IP 255.255.255.255
  - **-i:** forza l'utilizzo di un'interfaccia di rete. Senza l'opzione, arping cerca di “indovinare” alla configurazione delle interfacce.

Sniffing dei pacchetti: Tcpdump (1)

```
tcpdump [ -adeflnNOpqRStuvxX ]  [ -c count ]
                [ -C file_size ]  [ -F file ]
                [ -i interface ]  [ -m module ]  [ -r file ]
                [ -s snaplen ]  [ -T type ]  [ -w file ]
                [ -E algo:secret ]  [ expression ]
```

Alcune opzioni:

- **-c** Termina dopo aver ricevuto count pacchetti
- **-e** Visualizza gli header a livello DATA LINK
- **-E** Utilizza algo:secret per decifrare i pacchetti IPsec
- **-i** Rimane in ascolto sull'interfaccia di rete indicata
- **-n** Non converte gli indirizzi (IP, di porta…) in nomi
- **-r** Legge i pacchetti dal file file
- **-X** Visualizza i pacchetti in formato hex e ascii

Tcpdump (2)

- Tcpdump stampa a video l'header dei pacchetti in transito su una LAN che corrispondono alle caratteristiche indicate in expression
- Quando Tcpdump termina la sua esecuzione fornisce un report riguardante il numero di pacchetti ricevuti ed il numero di pacchetti scartati dal kernel.

```
244 packets received by filter
76 packets dropped by kernel
```

Tcpdump - Espressioni

Con le expression si definiscono i criteri qualitativi coi quali scegliere i pacchetti da visualizzare.

Le expression consistono in una o più primitive precedute da "**qualificatori**":

- **type** → host, net, port

  Esempio: 'host 155.185.54.156', 'port 22', ecc.

- **dir** → src, dst, src or dst

  Esempio: 'src 155.185.54.156'

- **proto →** ether, fddi, tr, ip, ip6, arp, rarp, decnet, tcp, udp

  Esempio: 'tcp port 21', 'arp net 155.185.54'

Test con solo un host configurato (1)

Configuriamo l'indirizzo IP dell'interfaccia ***eth0*** di ***h2***

```
(h2) # ifconfig eth0 192.168.1.1
```

Testiamo se ***h2*** risponde correttamente utilizzando ***arping*** da ***h1***

```
(h1) # arping -0i eth0 192.168.1.1
ARPING 192.168.1.1
42 bytes from 02:04:06:0f:47:dc (192.168.1.1): index=0 time=1.011 sec
42 bytes from 02:04:06:0f:47:dc (192.168.1.1): index=1 time=1.008 sec
42 bytes from 02:04:06:0f:47:dc (192.168.1.1): index=2 time=1.010 sec
```

Nota:

- non utilizziamo più l'opzione ***-B*** perchè specifichiamo l'indirizzo IP da risolvere

Test con solo un host configurato (2)

Analizzando il traffico su ***h2*** possiamo verificare che ora h2 sta rispondendo

```
(h2) # tcpdump -ni eth0 arp

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
08:04:55.933331 ARP, Request who-has 192.168.1.1 tell 0.0.0.0, length 28
08:04:55.933365 ARP, Reply 192.168.1.1 is-at 02:04:06:0f:47:dc, length 28
08:04:56.943018 ARP, Request who-has 192.168.1.1 tell 0.0.0.0, length 28
08:04:56.943055 ARP, Reply 192.168.1.1 is-at 02:04:06:0f:47:dc, length 28
```

Troubleshooting

Se qualcosa **non funziona**

NON CONTROLLARE (SUBITO) IL FILE DI CONFIGURAZIONE DELLE INTERFACCE

MA UTILIZZARE I COMANDI OPPORTUNI PER VERIFICARE LO STATO ATTUALE DEL SISTEMA

Solo **dopo aver verificato** qual è lo stato attuale del sistema andiamo a cercare cosa è stato a causare l'errore.

Troubleshooting:
Correggere errori di configurazione

In caso di errori di configurazione, le direttive **ifup** e **ifdown** potrebbero non riuscire ad attivare e disattivare correttamente le interfacce.

In certe situazioni potremmo non essere in grado nè di attivare nè di disattivare un'interfaccia, perchè in uno stato inconsistente.

Workflow suggerito:

ifup → *ERRORE* → *controllare la configurazione dell'iterfaccia*

- se l'interfaccia è parzialmente configurata, è probabile che non riusciremo a disattivarla utilizzando ***ifdown***:

  → ***deconfigurare l'interfaccia (rimuovere gli indirizzi assegnati)*** →

→ ***ifdown (o disattivare interfaccia con altro comando)*** → ***correzione configurazione*** → ***ifup***

oppure

→ ***configurazione temporanea***: verifichiamo se la nostra configurazione introduce dei conflitti utilizzando i comandi di configurazione temporanea.

Troubleshooting:
Riavvio servizi di rete

In caso di **conflitti "critici"**, possiamo riavviare i servizi di rete:

```
# systemctl restart networking
```

Durante il riavvio, il sistema eseguirà tutte le operazioni svolte durante la fase di boot del sistema, **inclusa la lettura dei file di configurazione.**

Se ci sono degli **errori nei file di configurazione**, potremmo ottenere un sistema inconsistente anche dopo il riavvio dei serivizi!

Quindi, preferire la disattivazione e riattivazione specifica di singole interfacce per capire dove stiamo sbagliando.

Ricordare inoltre che networking è un servizio del sistema operativo, e alcune opzioni di configurazione che vengono gestite dal kernel o da altri servizi non sono ovviamente interessate (ad esempio, vedremo ip forwarding)

Impostazioni interfaccia di rete: Configurazione temporanea

I comandi **ifconfig** e **ip** possono essere utilizzati anche per configurare le interfacce di rete senza utilizzare alcun file.

In questo caso, la configurazione sarà **temporanea** e sarà persa al riavvio della macchina.

Configurazione di un indirizzo IP (sintassi più semplice, la complicheremo nelle prossime lezioni):

```
# ifconfig <iface> <ip-address> [up]
```

oppure:

```
# ip addr {add,change,replace} dev <iface> <ip-address>
```

Note: il comando **ip** è stato pensato per gestire facilmente diversi indirizzi IP associati alla stessa interfaccia di rete. Per questo motivo è necessario scegliere sempre l'azione opportuna (add, change, replace).

Inoltre, **ip** non configura automaticamente la subnet in base alla classe dell'indirizzo configurato (ci torneremo in seguito).

Troubleshooting: Stato delle interfacce di rete

In questi casi, ci sono due opzioni principali:

1) rimuovere le direttiva **auto**, per attivare manualmente le interfacce e verificare passo-passo il loro comportamento
2) **utilizzare dei comandi che non consultano i file di configurazione incriminati**

Attivazione/disattivazione dell'interfaccia di rete senza leggere alcun file di configurazione:

```
# ifconfig <iface> {up,down}
```

oppure:

```
# ip link set dev <iface> {up,down}
```

Troubleshooting: Deconfigurare un'interfaccia

Per rimuovere l'assegnazione di un indirizzo utilizzare:

```
# ifconfig <iface> 0
```

Oppure:

```
# ip addr flush dev <iface>
```

Oppure (in caso di indirizzi multipli, per cancellarne solo uno)

```
# ip addr del <address> dev <iface>
```

Sistemi alternativi per la configurazione di rete

- In questo corso utilizzeremo il sistema ***ifupdown*** per la configurazione delle reti, gestito tramite file di configurazione ***interfaces*** per le configurazioni permanenti
  - Sono sistemi *formalmente* deprecati, ma *de-facto* ancora presenti su larga parte dei sistemi Linux – e ricevono ancora aggiornamenti
- Come alternativa, a volte vedremo la più moderna suite ***iproute2***
  - Il comando ***ip*** fa parte di questa suite
- Non utilizzeremo ***NetworkManager***, presente tramite un configuratore grafico in distribuzioni con desktop environment installato (e.g., Gnome), ma gestibile da linea di comando tramite il comando ***nmcli***. Suggerito per configurazioni complesse (e.g., failover, vpn, proxy)
- Altri sistemi per la configurazione di rete noti (Linux):
- ***Network-scripts*** *(Red-hat, CentOS, …)*, ***connman*** *(Linux embedded)*, ***systemd-networkd*** *(presente con systemd)*, ***netplan*** (Ubuntu server)
- Nota: alcuni di questi sistemi sono presenti nelle attuali macchine UML su marionnet, potete attivarli con systemctl e sperimentare (cercate documentazione online)

Esempi NetworkManager

Se attivate il servizio di default viene attivato un client dhcp

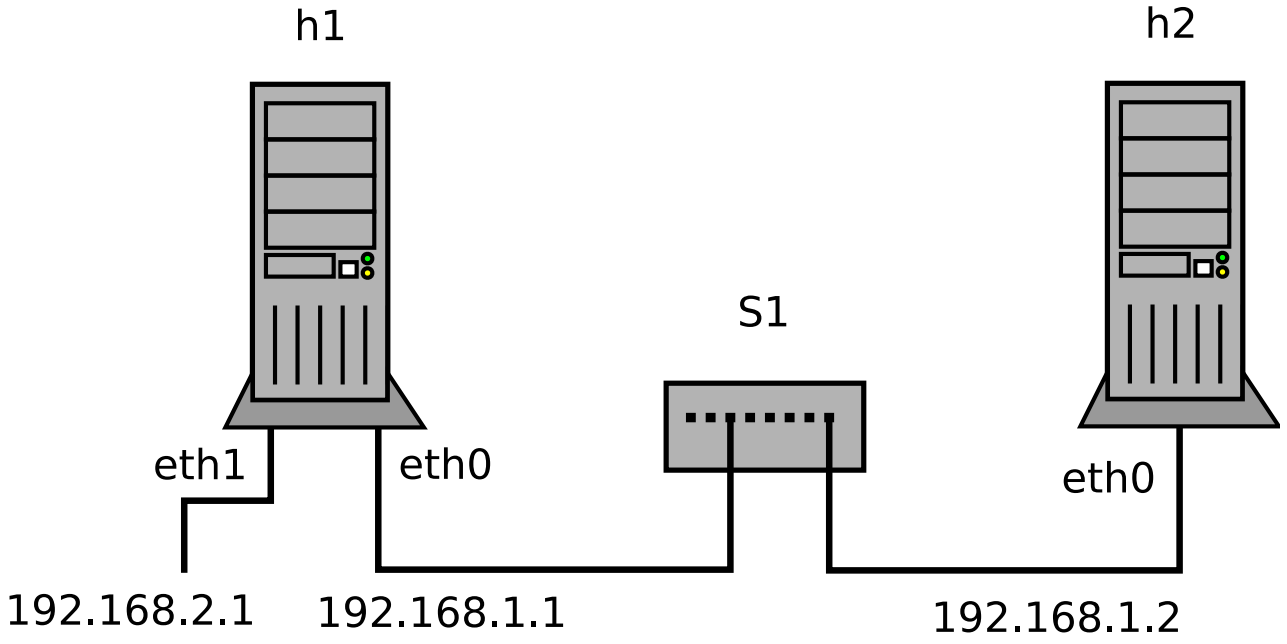
```
# systemctl enable --now NetworkManager
```

Potete modificare la configurazione con un ip statico con un comando come il seguente (per assegnare a eth0 ip 192.168.0.1/24):

```
# nmcli connection modify Wired Connection 1 \
    ipv4.method manual ipv4.address 192.168.0.1/24
```

Esercizio di approfondimento (1)

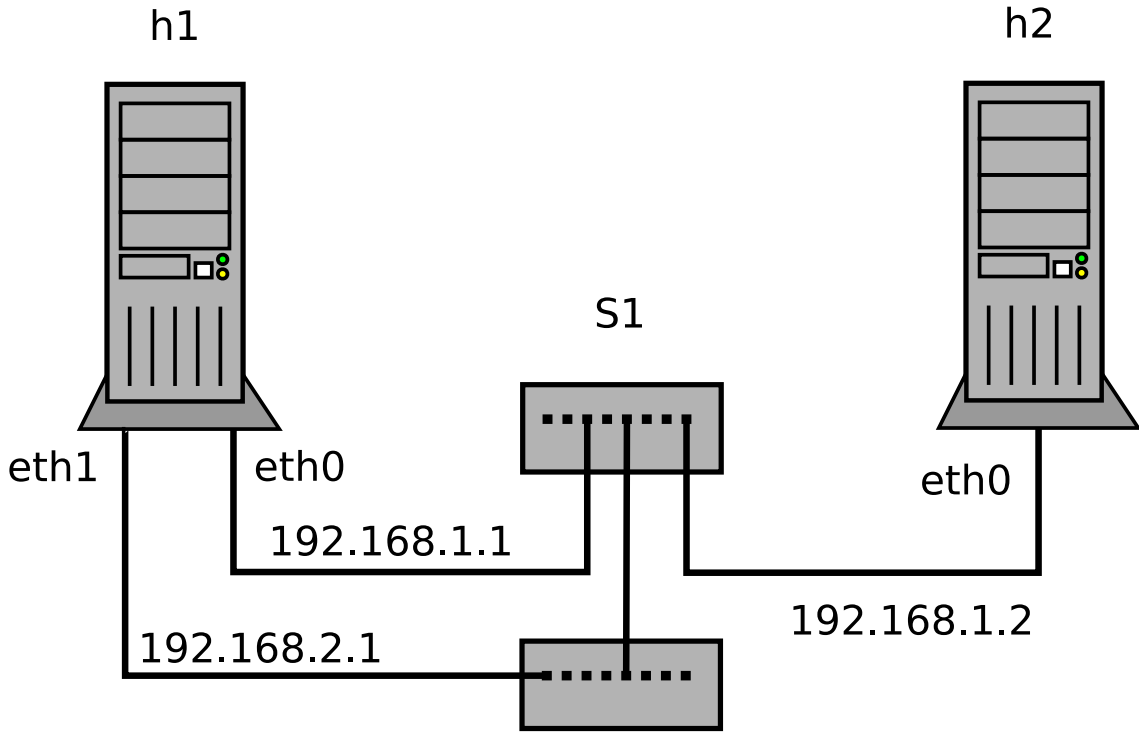
- Provare ad aggiungere un'interfaccia di rete ad ***h1***:
  - lasciare scollegata l'interfaccia
  - configurare un indirizzo IP (192.168.2.1)
  - cercare di risolvere il nuovo indirizzo assegnato tramite ***arping*** da ***h2***



Che risultato ci aspettiamo?

Esercizio di approfondimento (2)

- Provare ora collegando la nuova interfaccia allo stesso switch (o a un secondo switch collegato al primo, per ottenere uno schema migliore in Marionnet)
  - cercare di risolvere il nuovo indirizzo assegnato tramite ***arping*** da ***h2***



Anche considerando il risultato ottenuto in precedenza, che risultato ci aspettiamo?