

**PARTE C**

**VLAN**

## **Modulo 1: VLAN**

# LAN Tradizionale

- **Gli host sono aggregati “fisicamente” mediante dispositivi di rete, quali hub, switch e router**

Hub: non differenziano il dominio di collisione né il dominio di broadcast

Switch: differenziano il dominio di collisione ma non il dominio di broadcast

Router: differenziano sia il dominio di collisione sia il dominio di broadcast

# *Virtual Local Area Network*

- **Mediante le VLAN gli host possono essere raggruppati "logicamente"**

dipartimento, applicazioni che eseguono, funzioni, livello di riservatezza, ...

## **Facilità di gestione**

Invece di ricablare ad ogni spostamento, si modifica la configurazione dei dispositivi di rete

## **Isolamento**

Comunicazione diretta impedita tra VLAN diverse

## **Performance**

Il traffico di broadcast è limitato agli host della VLAN

# *Implementazione*

- **Tecnicamente creare una VLAN equivale a creare un dominio di broadcast separato**

Gli host che si trovano all'interno della VLAN possono comunicare direttamente

Gli host che si trovano in VLAN differenti possono comunicare mediante l'intermediazione di un dispositivo di rete (router)

**Funziona perché per risolvere un indirizzo IP il protocollo ARP individua l'indirizzo di destinazione MAC mediante broadcast**

La richiesta broadcast arriverà solo agli host facenti parte della stessa VLAN

# ***Bridge VLAN-aware***

- **Access list definiscono quali porte possono ricevere/inviare frame da/verso le diverse VLAN**

Un frame in arrivo al bridge viene etichettato con l'identificatore numerico della VLAN

Viene inoltrato solo sulle porte che possono accedere alla relativa VLAN

**L'assegnazione della VLAN può avvenire in base a diverse proprietà del pacchetto**

livello 1 - porta di ingresso

livello 2 - mac address del frame  
ma anche a livello 3, 4

# Collegamento al bridge

- **Access link**

Usato da dispositivi o segmenti di rete **VLAN-unaware**. Il tagging e l'untagging sono eseguiti in modo trasparente dal bridge

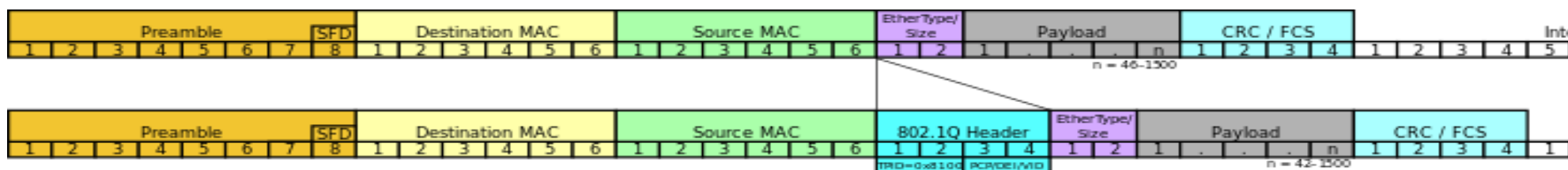
- **Trunk link**

Tutti i dispositivi connessi a questo link sono **VLAN-aware** e sono in grado di interpretare il tag VLAN presente nei frame [i.e. IEEE 802.1q]

- **Hybrid link**

Un link sulla quale possono essere connessi entrambi i tipi di dispositivi.

# IEEE 802.1q



- **Non incapsula il frame ethernet ma aggiunge un campo**

**Viene aggiunto un identificativo di 12bit**

4094 possibili VLAN (0 e 4095 riservati)

**TPID = 0x8100 per distinguersi da un normale frame ethernet**



## **Modulo 2: VLAN con Linux e vde\_switch**

# ***VLAN trunk su Linux***

- **Linux supporta la creazione di interfacce virtuali che gestiscono il tagging e l'untagging dei frame per una particolare VLAN**

## **Si crea un'interfaccia “virtuale” figlia di una fisica**

All'interfaccia virtuale arriveranno solo i pacchetti per la VLAN relativa ricevuti dall'interfaccia fisica

I pacchetti inviati tramite l'interfaccia virtuale avranno il tag VLAN aggiunto automaticamente e saranno inviati sull'interfaccia fisica

- ***Approfondiremo il discorso delle interfacce virtuali parlando di IP***

# VLAN trunk su Linux

## Creazione dell'interfaccia VLAN

```
ip link add link <physif> <virtif> \  
    type vlan id <N>
```

*oppure,*

```
vconfig add <physif> <N>
```

## Rimozione

```
ip link del <virtif>
```

*oppure,*

```
vconfig rem <physif>.<N>
```

L'interfaccia *<virtif>* si configura con i tool standard.

**NB:** l'interfaccia create con i precedenti comandi sarà *temporanea*.

# VLAN trunk su Linux

- Per rendere le modifiche permanenti (in Debian), è necessario inserire nel file `/etc/network/interfaces` un'interfaccia del tipo `<physif>.<N>`

I parametri sono gli stessi necessari per la configurazione di una classica interfaccia Ethernet. Ad esempio:

```
auto <physif>.<N>  
iface <physif>.<N> inet static  
    address <ip_address>  
    netmask <netmask>  
    gateway <ip_addr_gateway>
```

# VLAN trunk su Linux

Se l'interfaccia vlan è creata con *ip* il nome può non riflettere il vlan id, ad esempio:

```
ip link add link <iface> <virtf> type vlan id <id>
```

Per recuperare il vlan id associato ad una interfaccia <virtif> si può utilizzare il seguente comando:

```
grep VID /proc/net/vlan/<virtif>
```

Ad esempio:

```
# ip link add link eth0 pippo type vlan id 10  
# grep VID /proc/net/vlan/pippo  
pippo  VID: 10    REORDER_HDR: 1  dev->priv_flags: 1
```

# *vde\_switch*

- VDE = Virtual Distributed Ethernet <http://vde.sourceforge.net/>

## **Progetto Virtual Square (<http://wiki.v2.cs.unibo.it>)**

vde\_switch mette a disposizione funzionalità utili per la virtualizzazione di una rete LAN avanzata, configurabili tramite un terminale

supporto VLAN, bridge, STP, altro...

distribuito (switch su diverse macchine host )

modulare

compatibile con uml e uml\_switch

# Console di vde\_switch

Comandi principali di vde\_switch che useremo:

**port** : gestione delle porte

**vlan** : gestione delle VLAN

**hash** : gestione dell'hash table dello switch

***help [comando] è vostro amico!***

**NB:** altri comandi potrebbero essere disponibili in seguito al caricamento di plugin (e.g. vedi traffic sniffing con vde\_switch con pdump)

# Configurazione delle VLAN su vde\_switch

Creazione di due VLAN sugli switch

vlan/create **vlan\_number**

Impostare la VLAN per ogni porta a cui è collegato un host

port/setvlan **port\_number** **vlan\_number**

Aggiungere la porta collegata all'altro switch alla VLAN

vlan/addport **vlan\_number** **port\_number**

**NB:** differenza fra `port/setvlan` e `vlan/addport`?

Impostano rispettivamente VLAN untagged e tagged



# Esempio prompt di vde\_switch: help

```
vde$ help
```

```
0000 DATA END WITH '.'
```

```
COMMAND PATH
```

```
SYNTAX
```

```
HELP
```

```
-----
```

```
-----
```

```
-----
```

```
ds
```

```
=====
```

```
DATA SOCKET MENU
```

```
ds/showinfo
```

```
show ds info
```

```
help
```

```
[arg]
```

```
Help (limited to arg when specified)
```

```
logout
```

```
logout from this mgmt terminal
```

```
shutdown
```

```
shutdown of the switch
```

```
showinfo
```

```
show switch version and info
```

```
load
```


```
path
```

```
load a configuration script
```

```
[...]
```

# Esempio prompt di vde\_switch: VLAN

```
• vde$ vlan/print
0000 DATA END WITH '.'
VLAN 0000
-- Port 0001 tagged=0 active=1 status=Forwarding
-- Port 0002 tagged=0 active=1 status=Forwarding
VLAN 0042
-- Port 0002 tagged=1 active=1 status=Forwarding
-- Port 0003 tagged=0 active=1 status=Forwarding
• 1000 Success
```



Tagged = 1 : *Trunked Link*, accetta solo pacchetti taggati IEEE 802.1q  
Tagged = 0 : *Access Link*, accetta anche pacchetti non taggati, applica il tag a quelli non taggati

# Esempio prompt di vde\_switch: hash table


```
vde$ hash/print
```

```
0000 DATA END WITH '.'
```

```
Hash: 0024 Addr: 6a:9b:43:98:97:27 VLAN 0000 to port: 002 age 1  
secs
```

```
Hash: 0114 Addr: 12:42:12:f0:65:c3 VLAN 0000 to port: 001 age 1  
secs
```

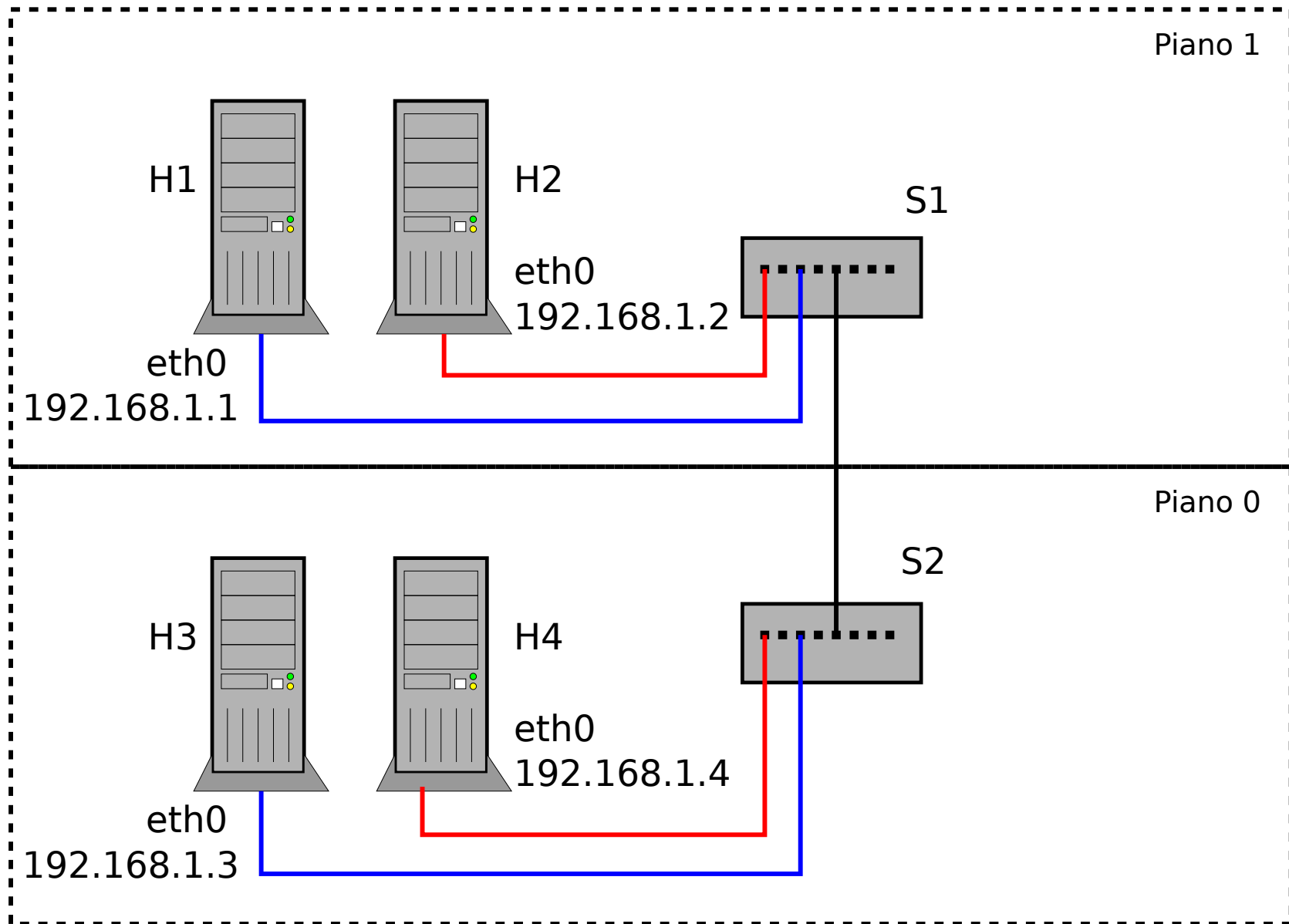
```
•  
1000 Success
```



Il collegamento fra MAC e porta è necessario per conoscere a che porte sono collegati i nodi uml quando ci si collega con stack standard uml (...eth0=daemon,...)

## **Modulo 3: Primo esercizio**

# Esercizio 1: VLAN Access link



# VLAN Access link

- **switch s1**

H1: 192.168.1.1

H2: 192.168.1.2

- **switch s2**

H3: 192.168.1.3

H4: 192.168.1.4

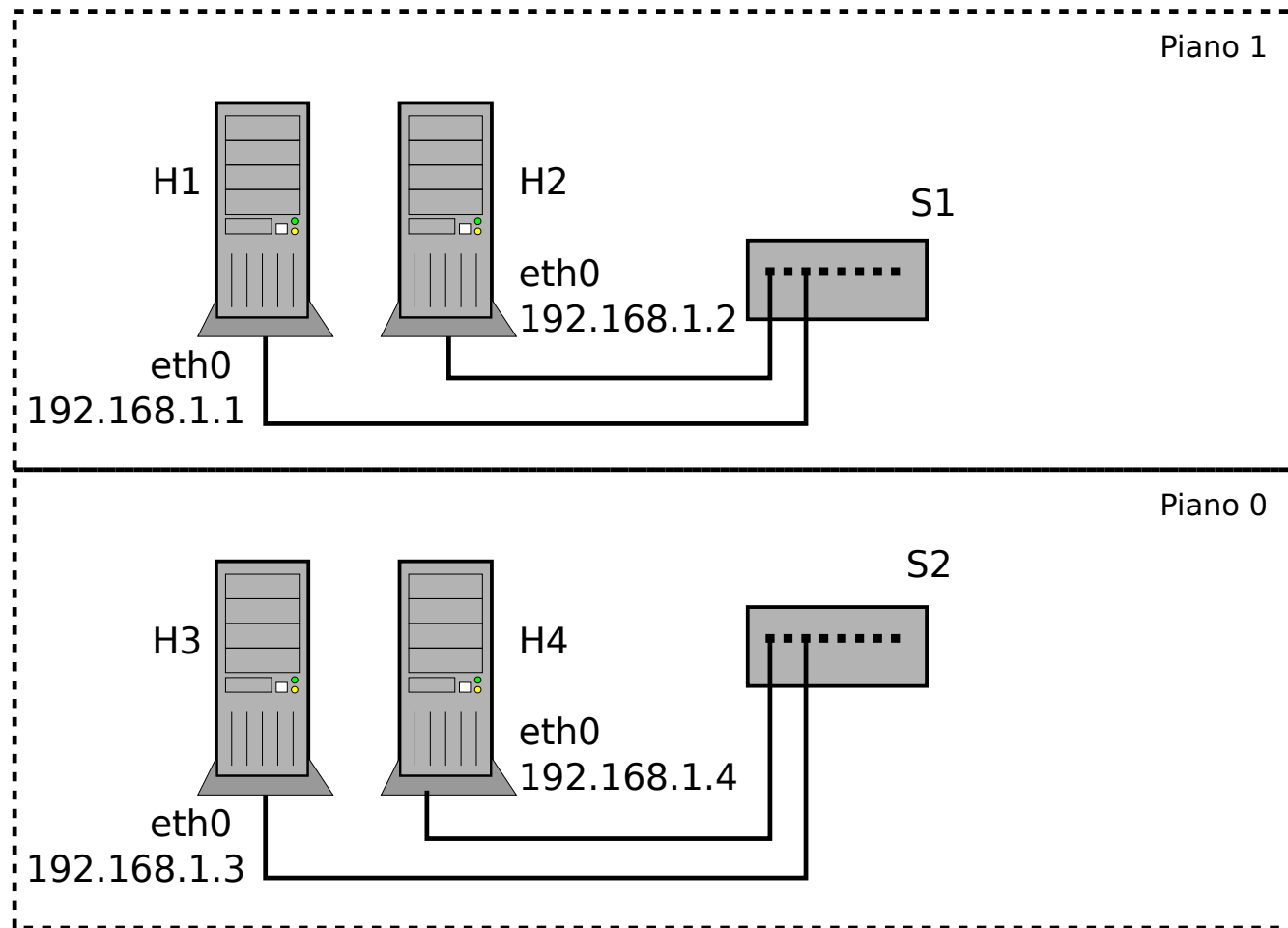
**Checkpoint 1:** creare due LAN *“classiche”*, separate tra loro  
=> H1 comunica con H2, e H3 con H4

**Checkpoint 2:** mettere in comunicazione le due LAN  
=> Tutti i nodi comunicano fra loro

**Obiettivo finale:** creare due VLAN *“trasversali”* alle LAN originali

=> H1 comunica solo con H3, H2 comunica solo con H4

# Checkpoint 1



- **4 Host**
- **2 Switch VDE**
- **Solo gli host collegati allo stesso switch comunicano**

# ***Checkpoint 1: possibile soluzione***

- **Decidere a che porte degli switch collegare i nodi della rete:**

**H1 collegato alla porta 1 dello switch 1**

**H2 collegato alla porta 2 dello switch 1**

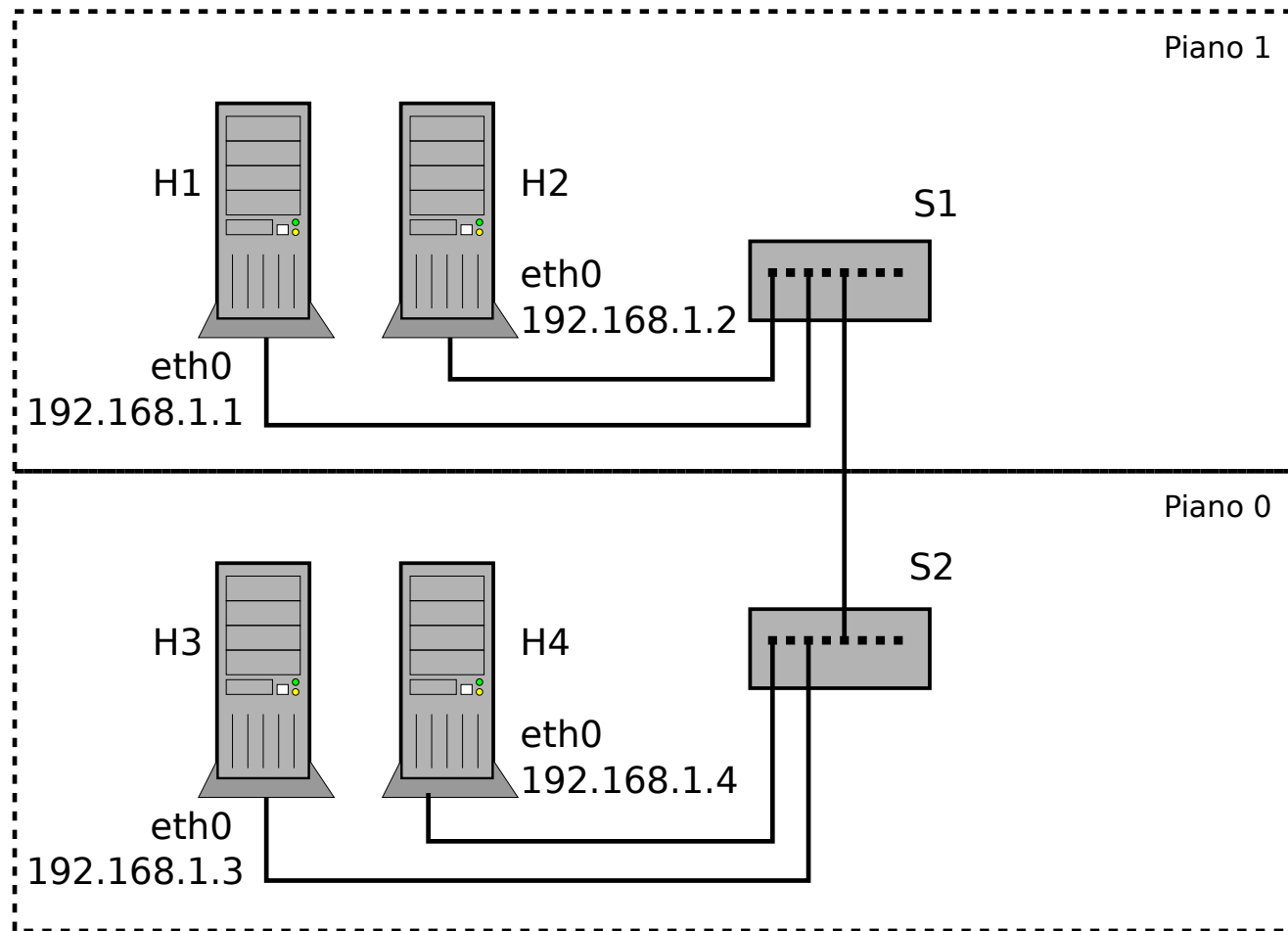
**H3 collegato alla porta 1 dello switch 2**

**H4 collegato alla porta 2 dello switch 2**

*La configurazione dei nomi degli host (opzionale) e delle interfacce di rete si esegue secondo le modalità viste nel precedente laboratorio*



# Checkpoint 2



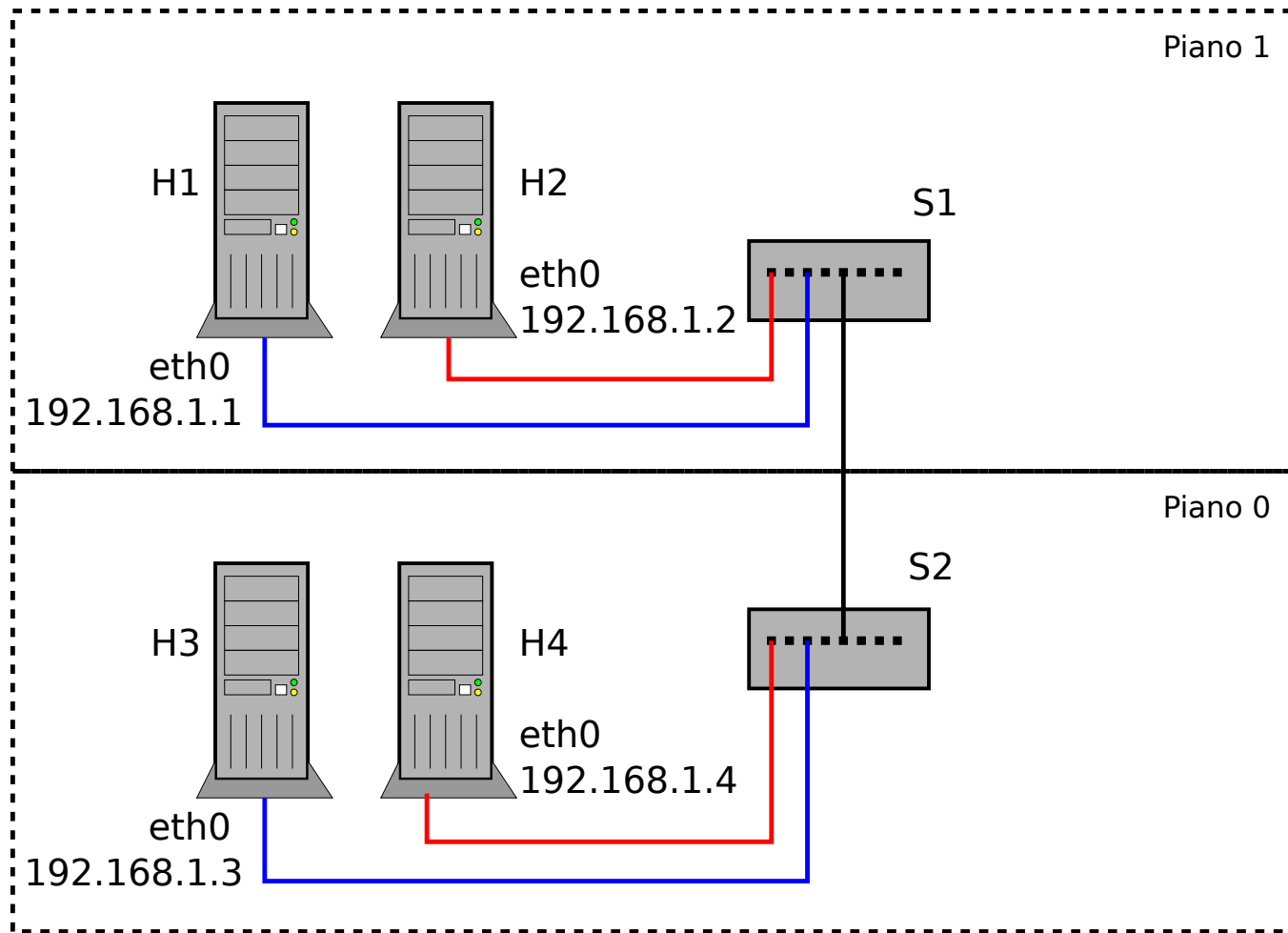
- **4 Host**
- **2 Switch VDE collegati**
- **Tutti i nodi comunica no fra loro**

***Target: collegare fra loro i due switch vde***  
***Nota: utilizzare un cavo cross!***

## ***Checkpoint 2: possibile soluzione***

- **Decidere che porte impiegare su ciascuno dei due switch**
- **Il collegamento fra gli switch viene effettuato impiegando la porta 3 su entrambi**
- **Verificare che tutti i nodi possano comunicare tra loro**

# Checkpoint 3



- 4 Host
- 2 Switch VDE collegati
- 2 VLAN
- I PC pingano solo all'interno della propria VLAN

**Target: configurare le VLAN sugli switch vde**

# Checkpoint 3: possibile soluzione

- Decidere che tag associare a ciascuna VLAN:
- La VLAN che collega i nodi H1 e H3 ha tag 10
- La VLAN che collega i nodi H2 e H4 ha tag 20
- Date le scelte implementative precedenti, possiamo eseguire su entrambi gli switch i medesimi comandi:

```
vlan/create 10  
vlan/create 20
```

→ Creazione VLAN 10

→ Creazione VLAN 20

```
port/setvlan 1 10  
port/setvlan 2 20
```

→ VLAN untagged 10 per la porta 1

→ VLAN untagged 20 per la porta 2

```
vlan/addport 10 3  
vlan/addport 20 3
```

→ VLAN tagged 10 per la porta 3

→ VLAN tagged 20 per la porta 3

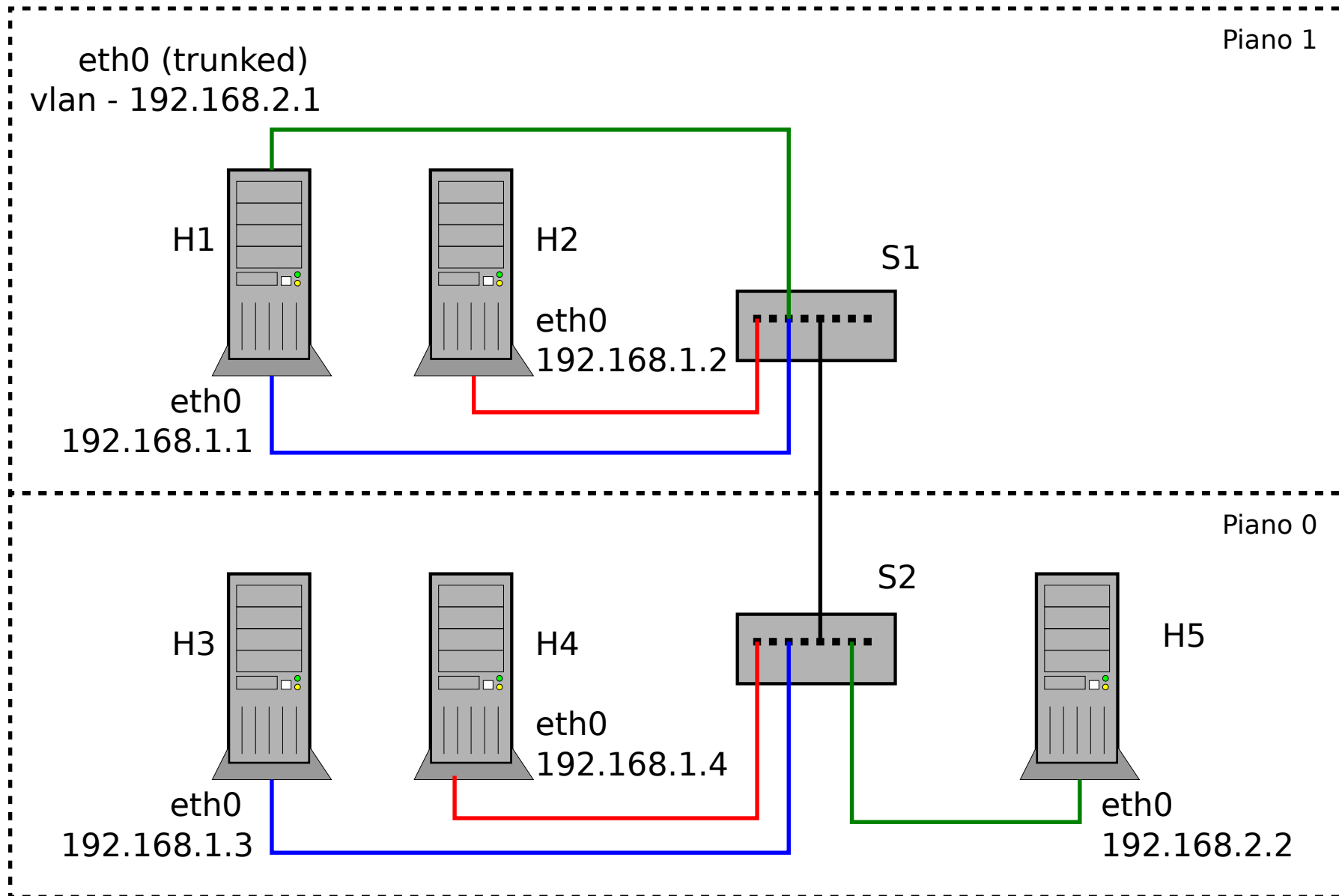
# Osservazioni

- Analizzare come il traffico Ethernet broadcast viene inoltrato dagli switch tramite i tool *arping* e *tcpdump* (eventualmente utilizzare anche i “led” degli switch di Marionnet che mostrano l'attività di rete delle porte). Ad esempio:
- come cambia l'inoltro del traffico tra il primo e il secondo checkpoint?
- perchè non possiamo utilizzare *ping* per testare il corretto funzionamento delle VLAN?
- Perchè dobbiamo utilizzare obbligatoriamente collegamenti *trunk link* fra gli switch?

## **Modulo 4: Secondo esercizio**

# Esercizio 2: VLAN trunk

Aggiungere un host H5 al secondo switch, che comunichi tramite VLAN solo con H1.



# ***VLAN trunk: possibile soluzione***

- **Si decide che la VLAN aggiuntiva usa tag 30, e H5 è collegato alla porta 4 di s2**
- **Occorre:**
- **Configurare H5**
- **Necessario aggiungere un'interfaccia virtuale di tipo VLAN su H1**
- **Configurare i due switch per propagare correttamente i frame della vlan 30**



# *VLAN trunk: possibile soluzione*

**Necessario configurare i due switch in modo analogo rispetto a quanto fatto prima**

```
switch1:  vlan/create 30  
            vlan/addport 30 1  
            vlan/addport 30 3
```

```
switch2:  vlan/create 30  
            port/setvlan 4 30  
            vlan/addport 30 3
```

**NB:** la porta 1 di s1 sarà configurata in modalità trunk (tagged), mentre la 3 di s2 in modalità access link (untagged)

# VLAN trunk: possibile soluzione (3)

**Necessario aggiungere un'interfaccia virtuale di tipo VLAN su H1**

**Configurazione on-the-fly (usando ip):**

```
ip link add link eth0 vlan30 type vlan id 30
ip address add 192.168.2.1/24 dev vlan30
ip link set dev vlan30 up
```

**Configurazione permanente, aggiungere a /etc/network/interfaces:**

```
auto eth0.30
iface eth0.30 inet static
    address 192.168.2.1
    netmask 255.255.255.0
```

**NB: in entrambi i casi si assume che eth0 sia già up**

- **Ci sono differenze a sniffare il traffico sull'interfaccia di rete fisica e su quella virtuale configurata per la VLAN sull'host H1?**
- **Provare ad analizzare il traffico usando anche il filtro vlan [id] su host configurati con collegamenti trunk. Cosa cambia?**
- **Provare a utilizzare il comando ping fra due host in VLAN separate. Che risultato si ottiene e perchè? Che tipo di traffico stiamo generando?**

- **Proviamo a contattare h1 da h5**
  - # arping -i eth0 192.168.1.1
  - # ping 192.168.1.1
- **Che differenze osserviamo?**
- **Perché ci sono queste differenze?**  
[suggerimenti]
  - Ricordarsi quanto è stato accennato sul routing IP
  - Ricordarsi del comportamento di arp rispetto a interfacce dello stesso host