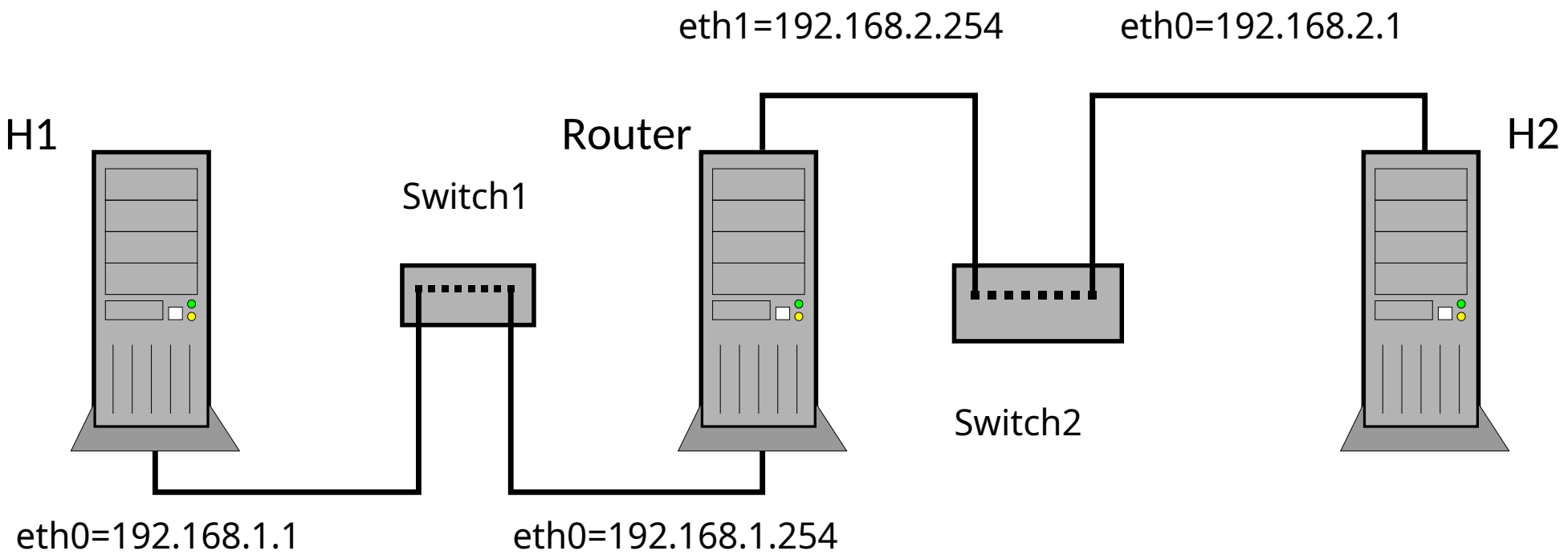
Routing su Linux

Scenario

- *Consideriamo la rete in figura*



LAN1 - 192.168.1.0
netmask 255.255.255.0

Ovvero **192.168.1.0/24**

LAN2 - 192.168.2.0
netmask 255.255.255.0

Ovvero 192.168.2.0/24

Scenario (2)

- Tre nodi
  - H1 :
    - eth0 = 192.168.1.1 netmask 255.255.255.0 (/24)
  - Router :
    - eth0 = 192.168.1.254 netmask 255.255.255.0 (/24)
    - eth1 = 192.168.2.254 netmask 255.255.255.0 (/24)
  - H2 :
    - eth0 = 192.168.2.1 netmask 255.255.255.0 (/24)

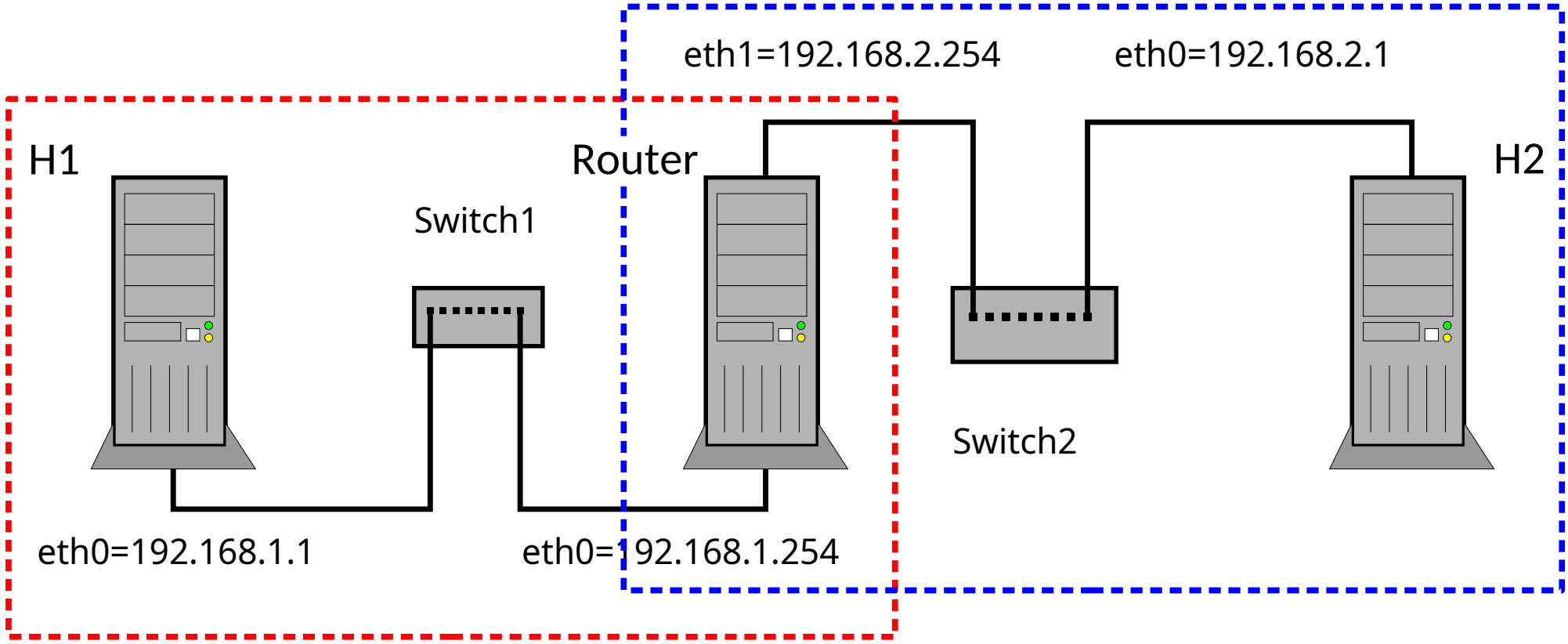
- ***Obiettivo****: far comunicare tutti i nodi*

Scenario (3)

- Usando una configurazione bridge, potremmo fare comunicare facilmente tutti i nodi, MA non rispetteremmo la configurazione richiesta
- Studio delle **due sottoreti:**
  - 192.168.1.0/24
    - HostMin: 192.168.1.1 ; HostMax: 192.168.1.254
  - 192.168.2.0/24
    - HostMin: 192.168.2.1 ; HostMax: 192.168.2.254
- Ogni nodo di una sottorete può comunicare a livello 2 solo con nodi appartenenti alla stessa sottorete. Per ogni altro nodo, è necessario ricorrere al **routing** dei pacchetti a livello IP.

Scenario (4)

*A livello 2 possiamo collegare H1 con Router e Router con H2, **ma non H1 e H2***



LAN1 - 192.168.1.0
netmask 255.255.255.0

LAN2 - 192.168.2.0
netmask 255.255.255.0

IP Forwarding e Routing

- Allo scopo di fare comunicare delle sottoreti, si individuano degli host che svolgono il ruolo di **router**, che **inoltrano** pacchetti da una sottorete a un'altra
  - l'inoltro dei pacchetti viene determinato in base all'indirizzo IP destinazione (livello 3 dello stack TCP/IP)
  - il router riceve un pacchetto con un indirizzo IP destinazione non suo e, invece di scartarlo (azione comune di un host), lo inoltra secondo **regole di routing**
  - ogni host deve conoscere quali sono i router a cui può inviare i pacchetti nel caso in cui i destinatari non facciano parte della sua sottorete, e li identifica con il termine di **gateway**

IP Forwarding e Routing (2)

- Problami da risolvere in configurazioni che comprendono multiple sottoreti:
  - ogni router deve sapere come raggiungere ogni rete, sia quelle alle quali è connesso, sia le altre
  - ogni host deve sapere a quali router (gateway) della propria sottorete deve inviare i pacchetti in base alla destinazione
- Ai nostri scopi, la configurazione (statica) degli host e dei gateway deve comprendere:
  - la definizione di **tabelle di routing** su tutti gli host
  - l'abilitazione dell'**ip forwarding** sugli host identificati come gateway

IP Forwarding su Linux

- La funzionalità di accettare pacchetti IP con destinazioni differenti e inoltrarli verso un destinatario viene chiamata **ip forwarding**, accessibile tramite il comando:

```
sysctl net.ipv4.ip_forward
```

- Se il parametro è **0** (default), l'ip forwarding è disabilitato. Per abilitare la funzionalità temporaneamente:

```
sysctl -w net.ipv4.ip_forward=1
```

- Altrimenti, per abilitarla permanentemente, impostare a **1** il campo **net.ipv4.ip_forward** nel file **/etc/sysctl.conf**. In questo caso, riavviare networking o usare:

```
sysctl -p /etc/sysctl.conf
```

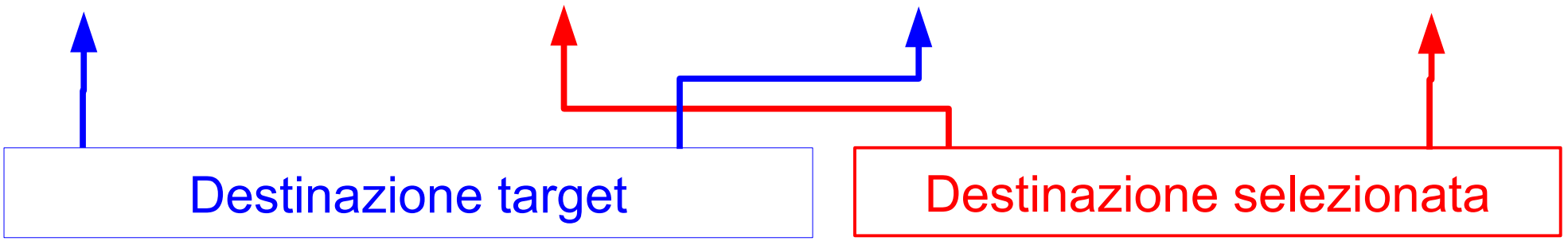
per rendere effettiva la modifica

Consultazione tabella di routing su Linux

Ogni host deve avere una tabella che raccoglie le regole di routing. Nei sistemi Linux, questa tabella è consultabile (e modificabile) tramite il comando **route** o **ip route show**:

```
# route -n # se non siamo root /sbin/route

Kernel IP routing table
Destination     Gateway         Genmask         Flags   Iface
192.168.1.0     *               255.255.255.0   U       eth1
155.185.48.128  *               255.255.255.192 U       eth0
192.168.2.5     192.168.1.254   255.255.255.255 UGH     eth1
0.0.0.0         155.185.54.190  0 0.0.0.0       UG      eth0
```



Consultazione tabella di routing su Linux (2)

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   Iface
192.168.1.0     *               255.255.255.0   U       eth1
155.185.48.128  *               255.255.255.192 U       eth0
192.168.2.5     192.168.1.254   255.255.255.255 UGH     eth1
0.0.0.0         155.185.54.190  0.0.0.0         UG      eth0
```

NB:

- la destinazione può essere una subnet, o un host
- è impostabile un default gateway per tutte le destinazioni target che non soddisfano nessuna delle destinazioni presenti nella tabella di routing (**NB**: ricordare che le destinazioni nella tabella sono definite da NetID + Netmask, e non solo da “Destination”)
  - Il default gateway è obbligatorio nel caso di raggiungibilità di Internet, sconsigliato nel caso di reti senza accessibilità

Consultazione tabella di routing su Linux (3)

Output corrispondente con comando **`ip route show`**

```
default via 155.185.54.190  dev eth1  \
     proto static  metric 1024

192.168.2.5 via 192.168.1.254  dev eth0  \
     proto static  metric 1024

192.168.1.0/24  dev eth0  proto kernel  \
     scope link  src 192.168.1.35

155.185.48.128/26  dev eth1  proto kernel  \
     scope link  src 155.185.48.147
```

Aggiungere regole di routing su Linux

Routing verso un host:

```
# route add -host <target> gw <gwaddr>
```

Routing verso una subnet:

```
# route add -net <target> gw <gwaddr>
```

Impostazione del default gateway:

```
# route add default gw <gwaddr>
```

Esempi:

```
# route add -host 192.161.4.1 gw 192.161.1.254
# route add -net 155.185.48.128 netmask \
   255.255.255.128 gw 192.161.1.254
# route add default gw 192.168.2.254
```

Rendere permanenti le tabelle di routing su Debian

- Si agisce sui blocchi delle interfacce in ***/etc/network/interfaces***
- Il default gateway può essere impostato tramite comando ***gateway***.
- Altre regole di routing possono essere impostate tramite direttiva ***post-up***, che esegue un comando in seguito all'attivazione dell'interfaccia.

Esempio:

```
iface eth0 inet static
      address 192.168.1.1
      netmask 255.255.255.0
      gateway 192.168.1.254
      post-up route add -net 192.168.2.0 \
         netmask 255.255.255.0 gw 192.168.1.253
```

Route con target ***reject***

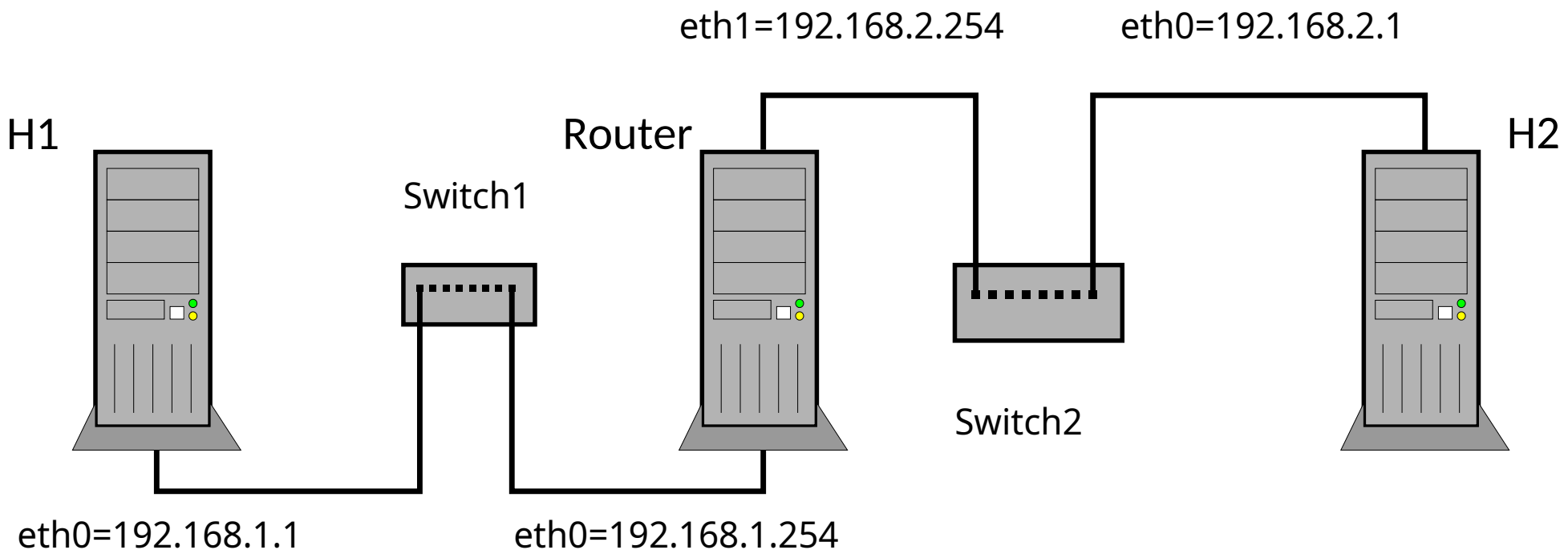
- Nel caso in cui un router sia referente per gestire un certo range di indirizzi, e che “sappia” che alcune sottoreti non siamo ancora impiegati, possiamo anche usare come target **reject**

  `route add {-host <>, -net <>}` ***`reject`***

  - Il router manderà un pacchetto ICMP di tipo host unreachable segnalando la non raggiungibilità della destinazione
    - Distinguere dall’errore locale *host unreachable* mostrato nel caso di timeout del protocollo ARP
  - Nota: questa regola non deve essere usata per “impedire” la raggiungibilità di reti esistenti, ma per segnalare reti o host non esistenti
    - Per impedire l’accesso vedremo i *firewall* verso la fine del corso

Risoluzione dello scenario

- Configurare correttamente la rete mostrate in precedenza



LAN1 - 192.168.1.0
netmask 255.255.255.0

LAN2 - 192.168.2.0
netmask 255.255.255.0

Risoluzione dello scenario (2)

- 2 subnet
  - LAN1: 192.168.1.0/24
  - LAN2: 192.168.2.0/24
- H2 fa parte di entrambe le subnet, non ha problemi a raggiungere gli altri due nodi
- H2 deve svolgere funzionalità di routing per permettere la comunicazione fra le due subnet
- *Soluzione*, considerando le interfacce di rete già configurate:
  - Abilitare l'**IP forward** su Router
  - Configurare la **tabella di routing** di H1 (H2) per raggiungere l'host H2 (H1) tramite Router. Possibile via:
    - routing verso l'host; routing verso la subnet, routing tramite default gateway

Risoluzione dello scenario (3)

IP Forward:

```
R # sysctl -w net.ipv4.ip_forward=1
```

Tabella di routing:

Routing basato su Subnet (migliore soluzione)

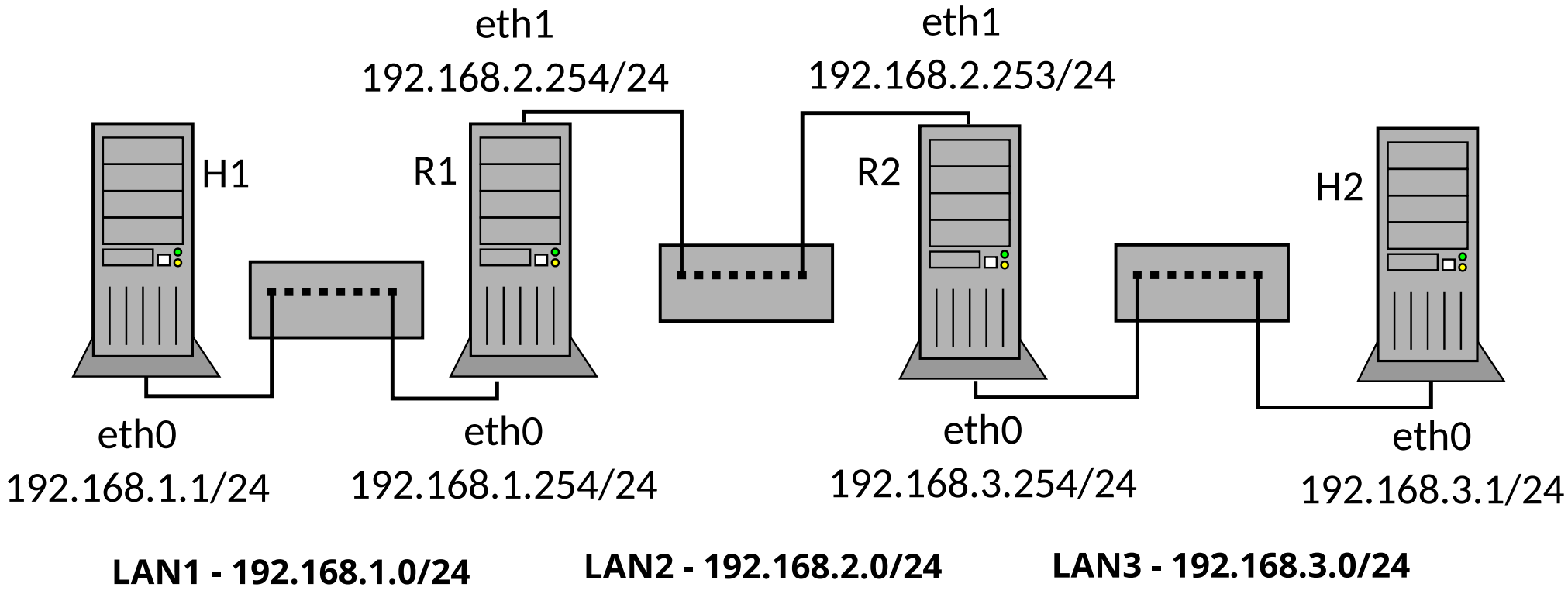
```
H1 # route add -net 192.168.2.0 netmask \
          255.255.255.0 gw 192.168.1.254
H2 # route add -net 192.168.1.0 netmask \
          255.255.255.0 gw 192.168.2.254
```

Routing basato su Host
(soluzione dimostrativa ma non realistica, nel "mondo reale" dovremmo creare un numero di regole pari al numero di host)

- H1 # **route add -host 192.168.2.1 gw 192.168.1.254**
- H2 # **route add -host 192.168.1.1 gw 192.168.2.254**

Estensione dello scenario (1)

- *Estendere la configurazione precedente per consentire la comunicazione fra i nodi della seguente rete*



Estensione dello scenario (1)
Svolgimento

È necessario inserire delle regole di routing:

- su H3, per raggiungere la rete locale e le altre reti

  H3# route add -net 192.168.2.0/24

- per raggiungibilità H2N sugli host della stessa rete

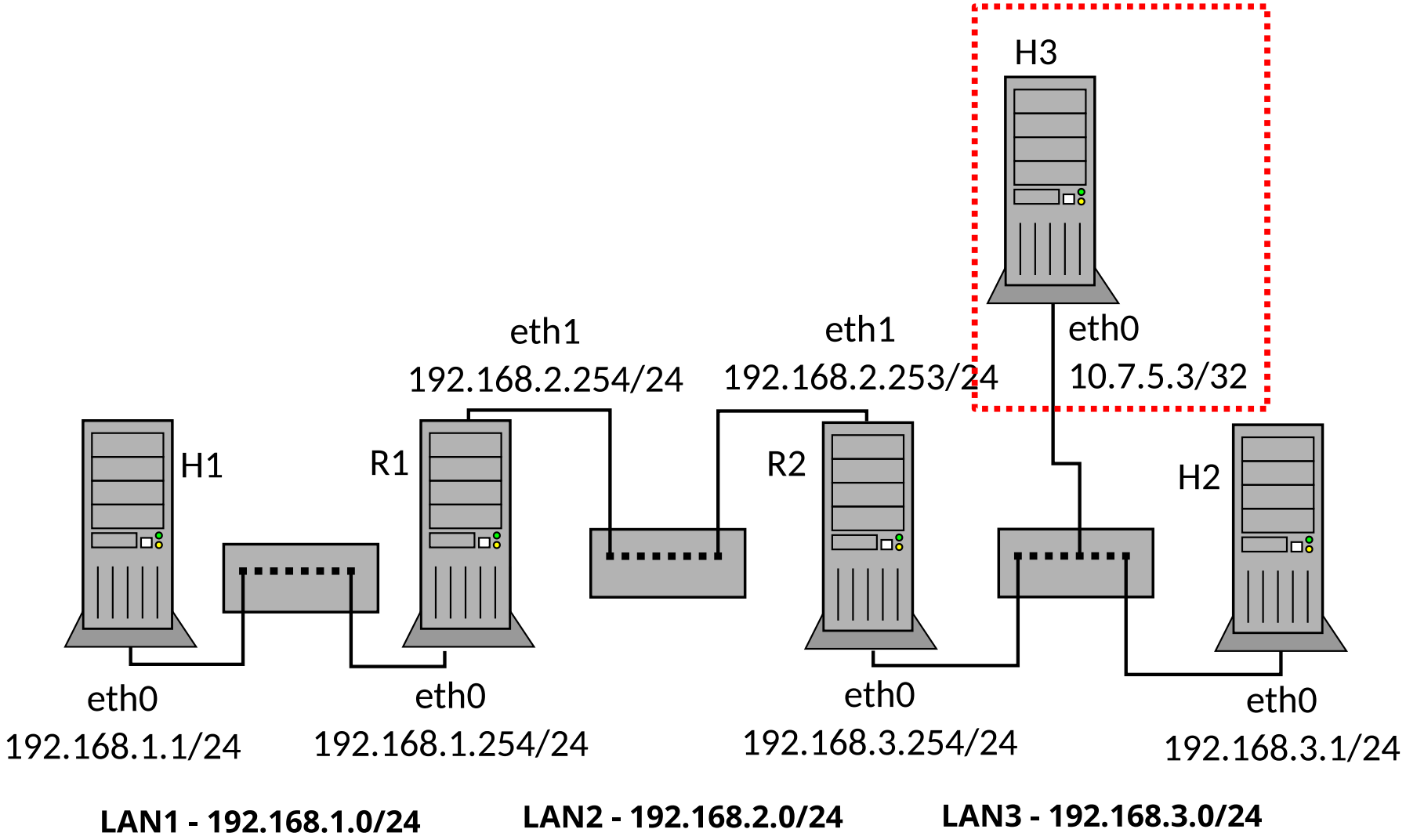
  H2# route add -host 10.7.5.3 dev eth0

  R2# route add -host 10.7.5.3 dev eth1

- *su tutti gli host dello stesso spazio di indirizzamento è necessario poi inserire delle regole ad-hoc per raggiungere quell'indirizzo tramite i dovuti router*
  - *Si rende evidente il motivo per cui è fondamentale sfruttare la logica di aggregazione degli indirizzi nelle reti IP*

Estensione dello scenario (2)

- *Estendere la rete immaginando di collegare un host a livello H2N con IP "al di fuori" delle LAN configurate*



Estensione dello scenario (1b)
Svolgimento

È necessario inserire delle regole di routing:

- su H3, per raggiungere la rete locale e le altre reti

```
H3# route add -net 192.168.3.0/24 dev eth0
H3# route add -net 192.168.2.0/24 gw 192.168.3.254
H3# route add -net 192.168.1.0/24 gw 192.168.3.254
```

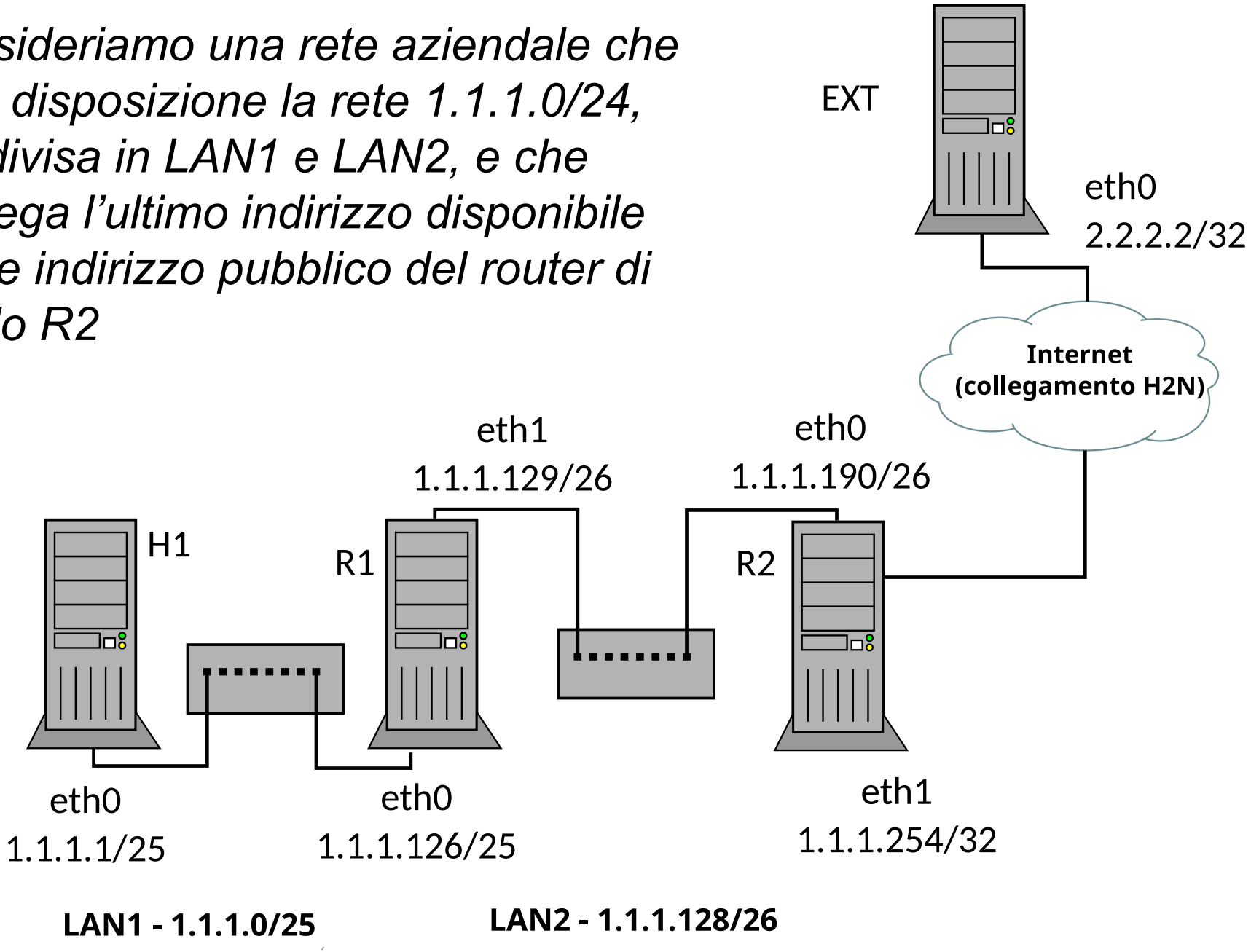
- per raggiungibilità H2N sugli host della stessa rete

```
H2# route add -host 10.7.5.3 dev eth0
R2# route add -host 10.7.5.3 dev eth1
```

- *su tutti gli host dello stesso spazio di indirizzamento è necessario poi inserire delle regole ad-hoc per raggiungere quell'indirizzo tramite i dovuti router*
  - *è evidente il motivo per cui è fondamentale sfruttare la logica di aggregazione degli indirizzi nelle reti IP*

Rete con accesso a Internet

Consideriamo una rete aziendale che ha a disposizione la rete 1.1.1.0/24, suddivisa in LAN1 e LAN2, e che impiega l'ultimo indirizzo disponibile come indirizzo pubblico del router di bordo R2



Rete con accesso a Internet (svolgimento – parte 1)

Notare che ora stiamo impiegando indirizzi IP pubblici (altrimenti servirebbe effettuare NATting, che vedremo nelle prossime lezioni)

Omettiamo configurazione degli indirizzi IP

Configuriamo default gateway per andare verso Internet, altrimenti regole specifiche a seconda delle reti da raggiungere

```
H1# route add default gw 1.1.1.126
R1# route add default gw 1.1.1.190
R2# route add -net 1.1.1.0/25 gw 1.1.1.129
```

Se consideriamo R2 referente per tutta la rete, e vogliamo indicare che la rete 1.1.1.192/26 non è utilizzata (il conflitto con l'indirizzo di R2 è solo apparente)

```
R2# route add -net 1.1.1.192/26 reject
```

Rete con accesso a Internet (svolgimento – parte 2)

“Simulazione” Internet H2N, in una rete reale potrebbe essere diverso
In questo caso potremmo considerare ext come il router dell’ISP/dell’AS

```
R2# route add -host 2.2.2.2 dev eth1
R2# route add default gw 2.2.2.2

ext# route add -host 1.1.1.254 dev eth0
ext# route add -net 1.1.1.0/24 gw 1.1.1.254
```