Network & System Defense

PROGETTO #2

Matteo Chiacchia
0300177

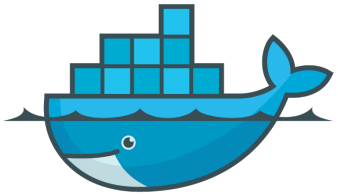
matteoch99@gmail.com

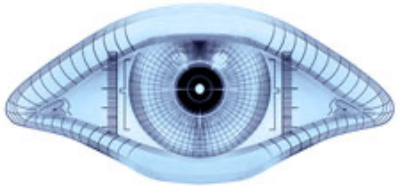
Introduzione

Topics

- MPLS/BGP VPN
- MACSec
- Firewall
- OpenVPN
- AVs

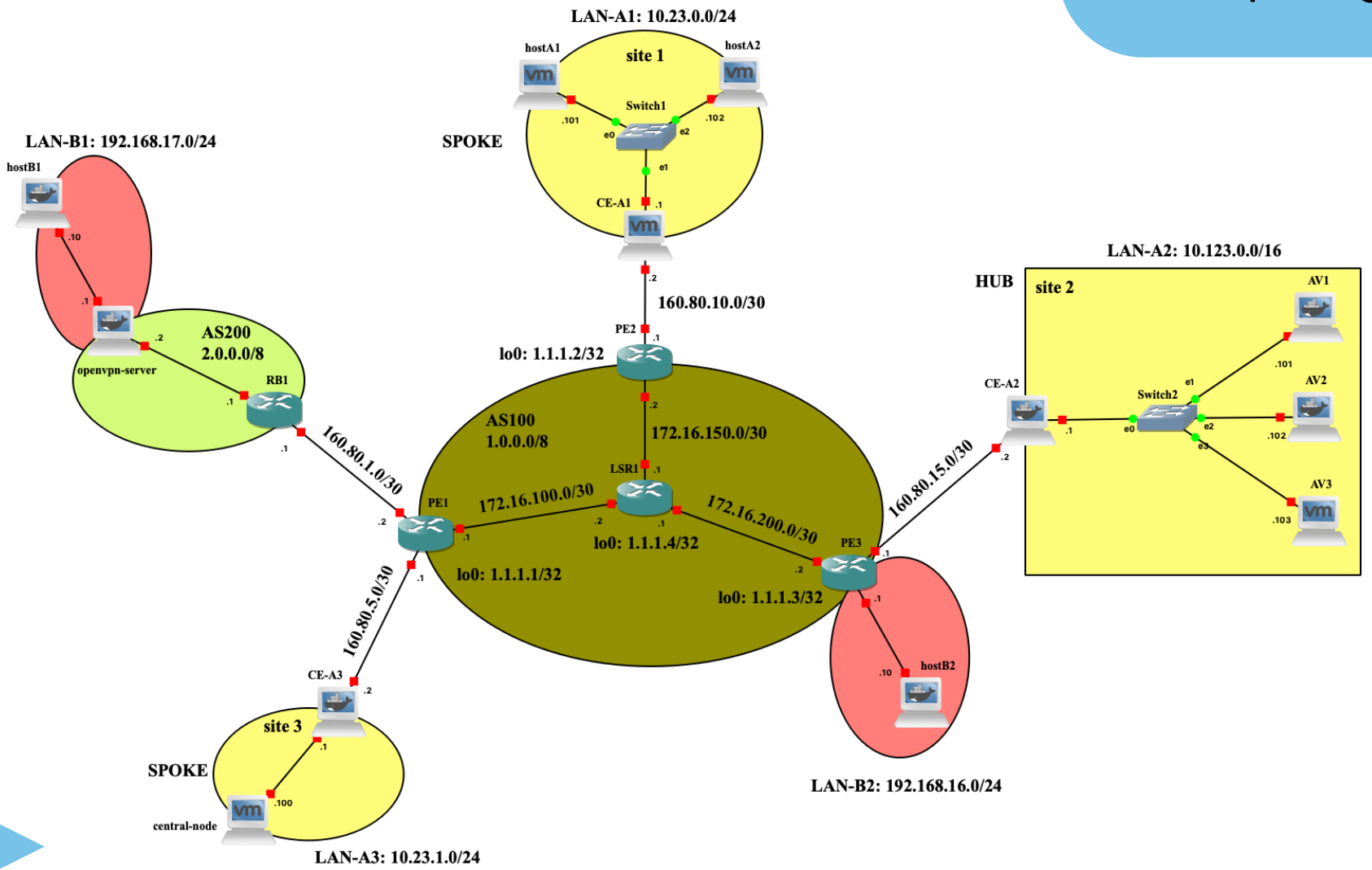
Software





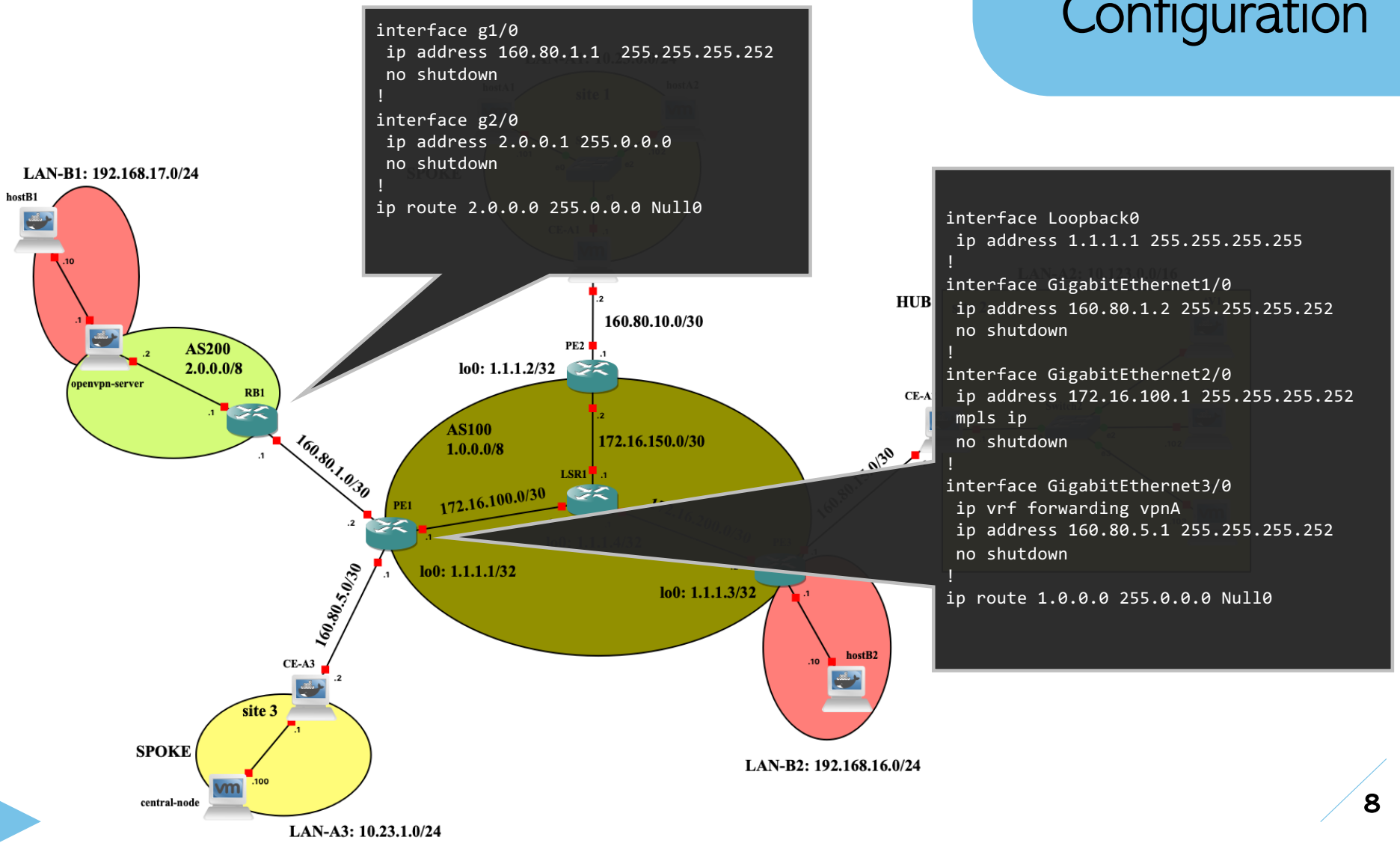
Topologia

Topologia

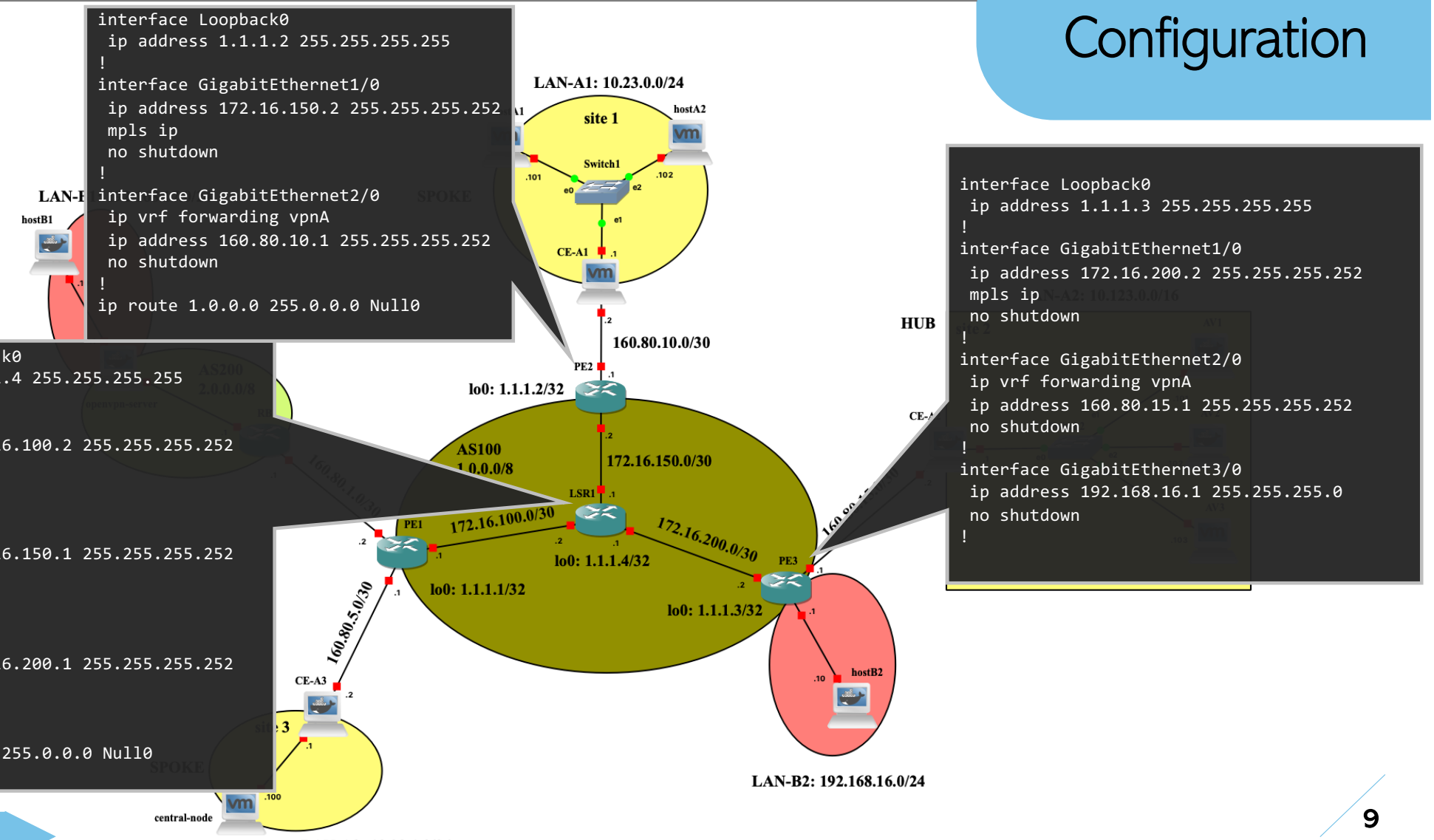

LAN-A1: 10.23.0.0/24
hostA1
site 1
hostA2
Switch1
.101
.102
e0
e2
e1
SPOKE
CE-A1
.1
.2
160.80.10.0/30
PE2
lo0: 1.1.1.2/32
LAN-B1: 192.168.17.0/24
hostB1
.10
.1
.2
AS200
2.0.0.0/8
openvpn-server
RB1
.1
.1
160.80.1.0/30
AS100
1.0.0.0/8
172.16.150.0/30
LSR1
.1
.2
PE1
172.16.100.0/30
172.16.200.0/30
.2
.1
.1
.2
lo0: 1.1.1.4/32
lo0: 1.1.1.1/32
lo0: 1.1.1.3/32
160.80.5.0/30
.1
.2
CE-A3
site 3
SPOKE
.1
.100
central-node
LAN-A3: 10.23.1.0/24
160.80.15.0/30
.2
.1
PE3
.1
.10
hostB2
LAN-B2: 192.168.16.0/24
LAN-A2: 10.123.0.0/16
HUB
site 2
CE-A2
.1
e0
Switch2
e1
e2
e3
AV1
.101
AV2
.102
AV3
.103

Network

Configuration



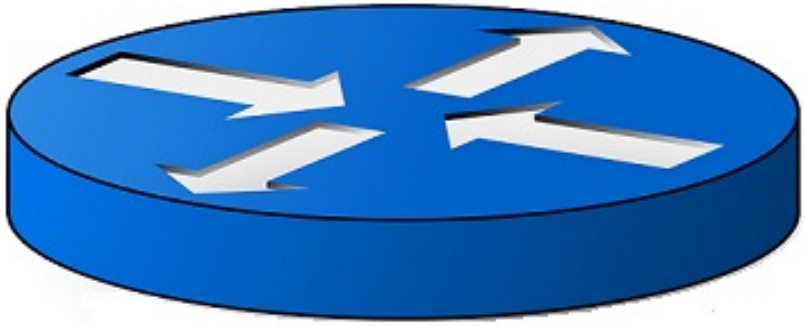
Configuration



Network

MPLS/BGP VPN

- Protocolli di routing:
  - *OSPF*
  - *BGP*
    - *eBGP*
    - *iBGP*
  - *MPLS*

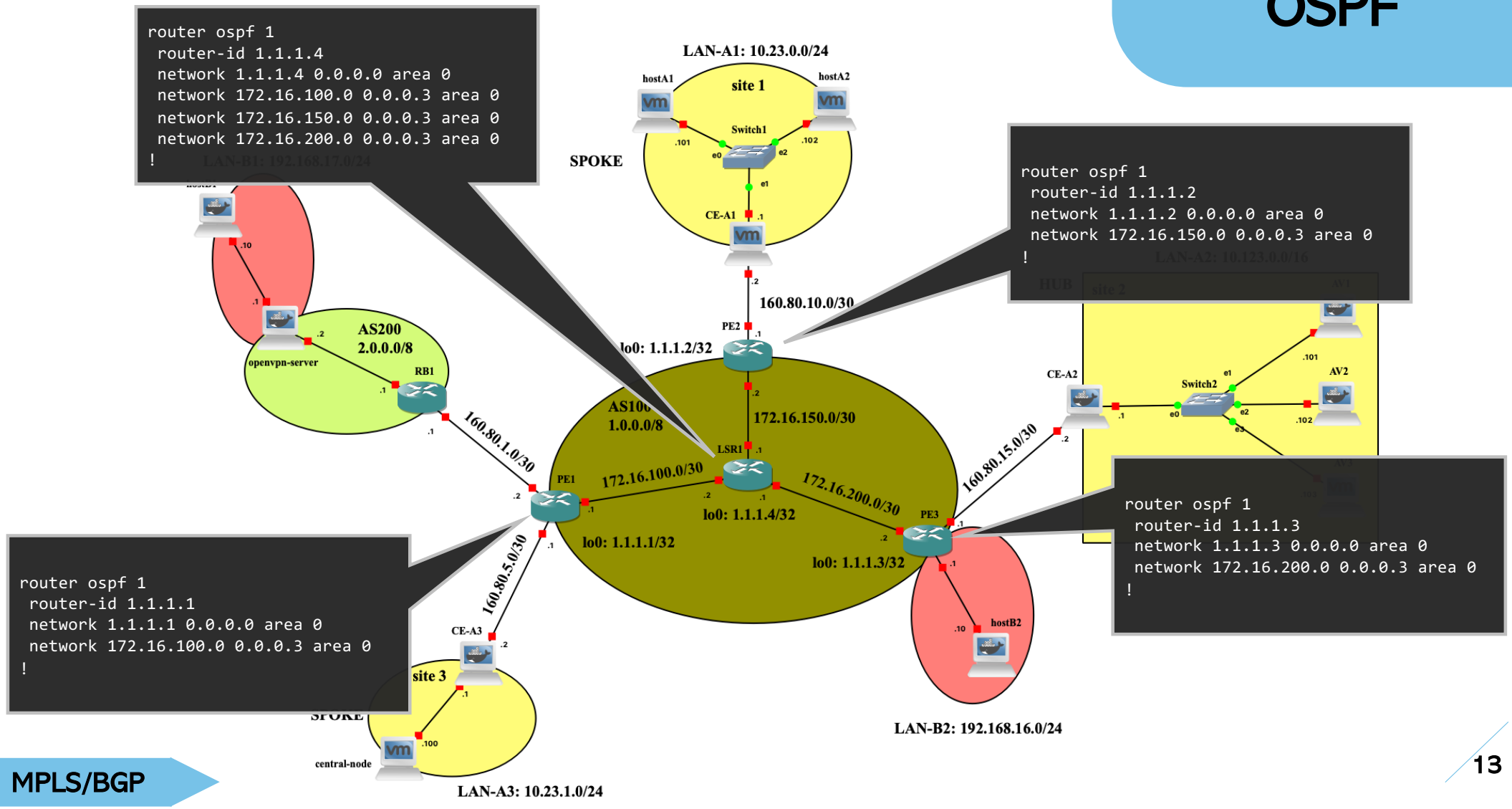


❑ OSPF: Open Shortest Path First

- ❖ Algoritmo di *routing* basato su *Link-State* per conoscere la topologia della rete e calcolare i percorsi migliori.
- ❖ Utilizza il *flooding* di informazioni.
- ❖ *Algoritmo di Dijkstra* per la determinazione del percorso a costo minimo *INTRA-AS*.
- ❖ Utilizzo specifico
  - Apprendimento delle rotte dell'AS100 da parte dei *PEs*.
  - Configurato con interfacce di ***Loopback***.

Link state è un tipo di protocollo di routing in cui la topologia dell'intera rete e tutti i *costi* dei collegamenti sono noti ai router di un certo AS.

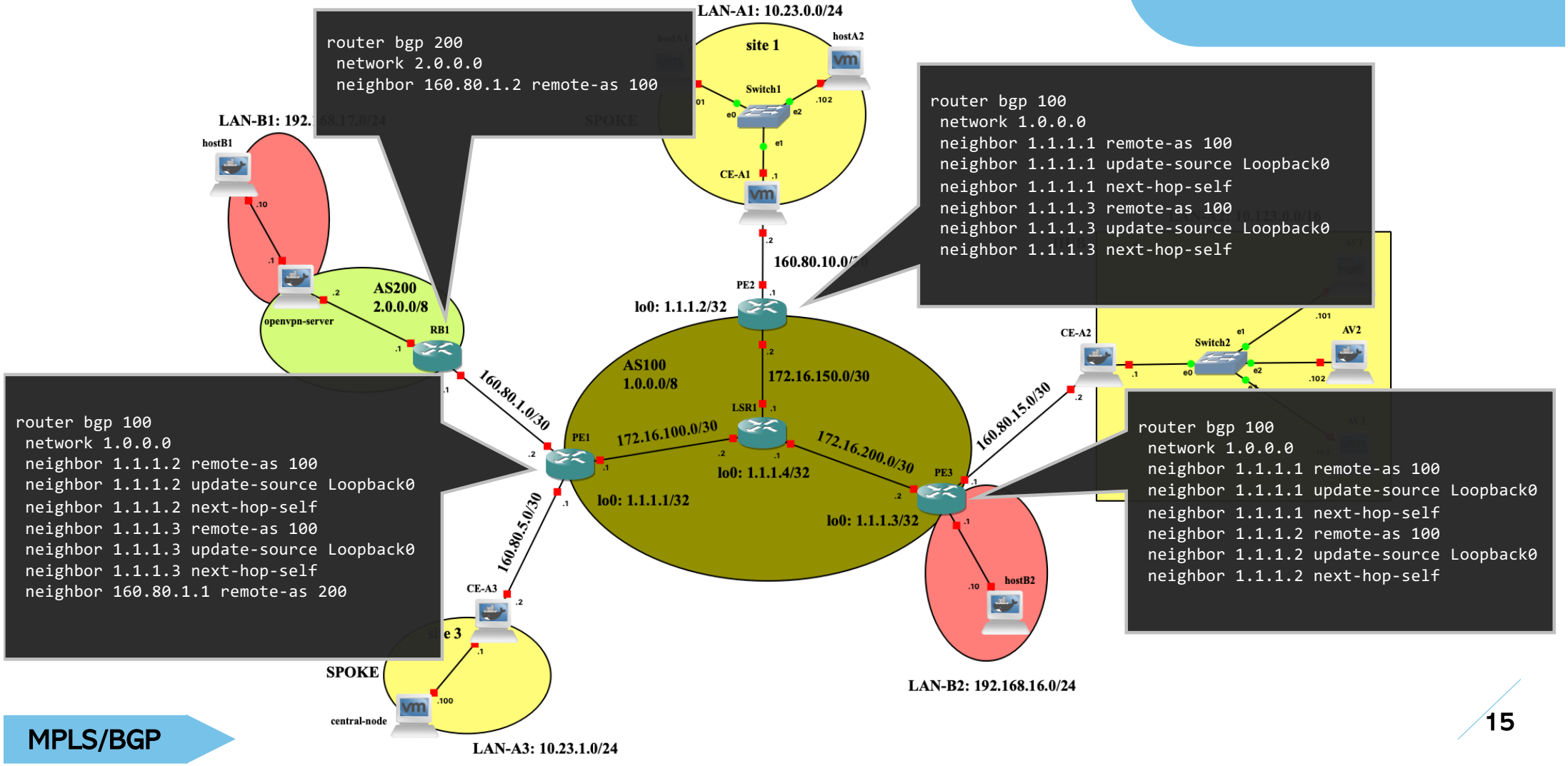
OSPF


router ospf 1
router-id 1.1.1.4
network 1.1.1.4 0.0.0.0 area 0
network 172.16.100.0 0.0.0.3 area 0
network 172.16.150.0 0.0.0.3 area 0
network 172.16.200.0 0.0.0.3 area 0
!
router ospf 1
router-id 1.1.1.2
network 1.1.1.2 0.0.0.0 area 0
network 172.16.150.0 0.0.0.3 area 0
!
router ospf 1
router-id 1.1.1.1
network 1.1.1.1 0.0.0.0 area 0
network 172.16.100.0 0.0.0.3 area 0
!
router ospf 1
router-id 1.1.1.3
network 1.1.1.3 0.0.0.0 area 0
network 172.16.200.0 0.0.0.3 area 0
!
LAN-A1: 10.23.0.0/24
site 1
hostA1
hostA2
Switch1
SPOKE
CE-A1
160.80.10.0/30
PE2
lo0: 1.1.1.2/32
AS200
2.0.0.0/8
openvpn-server
RB1
AS100
1.0.0.0/8
172.16.150.0/30
LSR1
160.80.1.0/30
PE1
172.16.100.0/30
172.16.200.0/30
PE3
lo0: 1.1.1.4/32
lo0: 1.1.1.1/32
lo0: 1.1.1.3/32
160.80.15.0/30
CE-A2
Switch2
AV2
HUB
site 2
160.80.5.0/30
CE-A3
site 3
central-node
hostB2
LAN-B2: 192.168.16.0/24
LAN-A3: 10.23.1.0/24

❑ BGP: Border Gateway Protocol

- ❖ Connette diversi *AS.*
- ❖ Scambio informazioni su rotte per raggiungibilità.
- ❖ **iBGP** per la conoscenza all'interno dell'AS delle rotte esterne.
- ❖ Utilizzo specifico
  - Apprendimento delle rotte tra *AS100* e *AS200*.
  - *iBGP* configurato con interfacce di ***Loopback.***
  - Rete *Full Mesh*: ogni router *iBGP* collegato (logicamente) a ogni altro router *iBGP* all'interno dell'AS100.

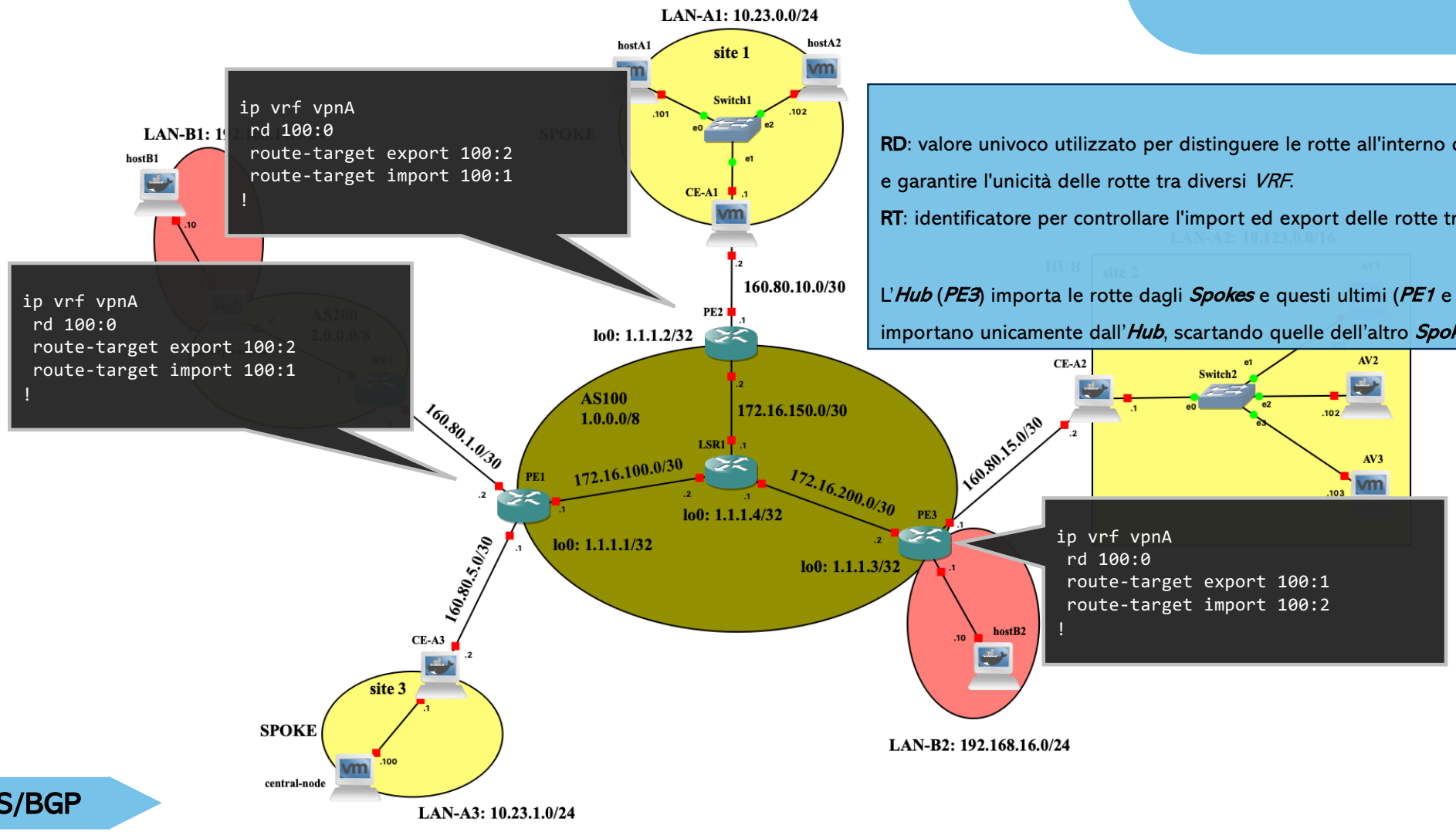
BGP


router bgp 200
network 2.0.0.0
neighbor 160.80.1.2 remote-as 100
LAN-A1: 10.23.0.0/24
site 1
hostA2
Switch1
CE-A1
router bgp 100
network 1.0.0.0
neighbor 1.1.1.1 remote-as 100
neighbor 1.1.1.1 update-source Loopback0
neighbor 1.1.1.1 next-hop-self
neighbor 1.1.1.3 remote-as 100
neighbor 1.1.1.3 update-source Loopback0
neighbor 1.1.1.3 next-hop-self
LAN-B1: 192.
hostB1
AS200
2.0.0.0/8
openvpn-server
RB1
160.80.10.0/
PE2
lo0: 1.1.1.2/32
AS100
1.0.0.0/8
172.16.150.0/30
LSR1
CE-A2
Switch2
AV2
router bgp 100
network 1.0.0.0
neighbor 1.1.1.2 remote-as 100
neighbor 1.1.1.2 update-source Loopback0
neighbor 1.1.1.2 next-hop-self
neighbor 1.1.1.3 remote-as 100
neighbor 1.1.1.3 update-source Loopback0
neighbor 1.1.1.3 next-hop-self
neighbor 160.80.1.1 remote-as 200
160.80.1.0/30
PE1
172.16.100.0/30
172.16.200.0/30
lo0: 1.1.1.4/32
lo0: 1.1.1.1/32
lo0: 1.1.1.3/32
PE3
160.80.15.0/30
router bgp 100
network 1.0.0.0
neighbor 1.1.1.1 remote-as 100
neighbor 1.1.1.1 update-source Loopback0
neighbor 1.1.1.1 next-hop-self
neighbor 1.1.1.2 remote-as 100
neighbor 1.1.1.2 update-source Loopback0
neighbor 1.1.1.2 next-hop-self
160.80.5.0/30
CE-A3
hostB2
SPOKE
central-node
LAN-A3: 10.23.1.0/24
LAN-B2: 192.168.16.0/24

❑ MPLS: Multi-Protocol Label Switching

- ❖ Utilizzo di ***Label*** per il *forwarding*
- ❖ Insieme a *BGP* si crea una ***VPN Intra-AS***
- ❖ Invio messaggi *MP-iBGP* per sincronizzazione tabelle ***VRF***
- ❖ Utilizzo specifico
  - Creazione di una *VPN intra-AS* per *LAN-A1*, *LAN-A2* e *LAN-A3*
  - Topologia *Hub e Spokes*
- ❖ Comandi per configurazione delle interfacce dei *router 7200*:
  - ➢ *mpls ip:* utilizzato per abilitare il forwarding *MPLS* (slide 8)
  - ➢ *ip vrf forwarding vpnA* : permette il forwarding del traffico della *VPN* verso i vari *CEs* (slide 8)

MPLS rd:rt

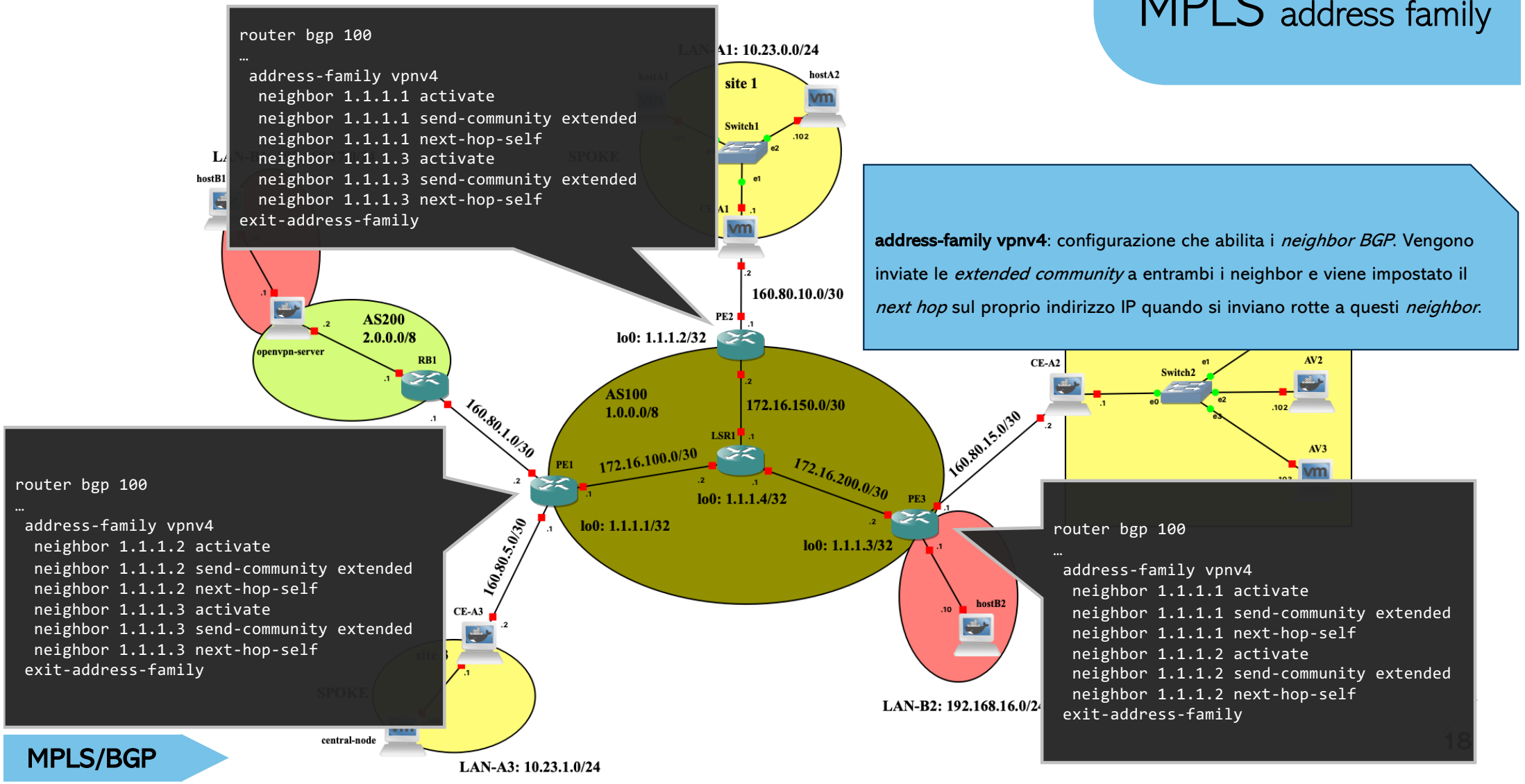


RD: valore univoco utilizzato per distinguere le rotte all'interno di un *VRF* e garantire l'unicità delle rotte tra diversi *VRF*.

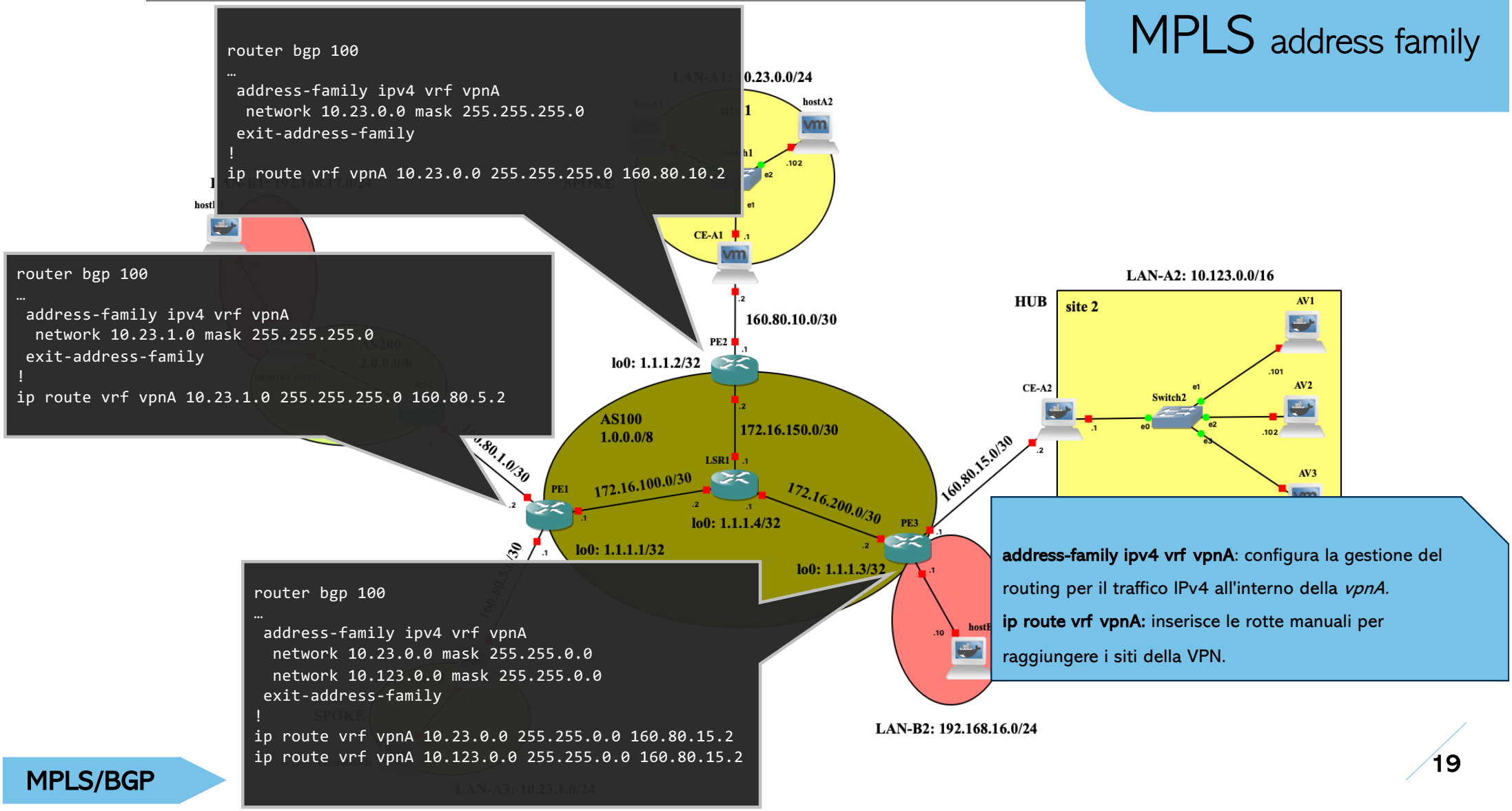
RT: identificatore per controllare l'import ed export delle rotte tra i *VRF*.

L'***Hub*** (***PE3***) importa le rotte dagli ***Spokes*** e questi ultimi (***PE1*** e ***PE2***) le importano unicamente dall'***Hub***, scartando quelle dell'altro ***Spoke***.

MPLS address family



MPLS address family



MacSec

MacSec & MKA

❑ MACsec: Media Access Control Security

- ❖ Siurezza a livello *MAC* nelle reti *LAN*
- ❖ *Encryption, frame integrity …*

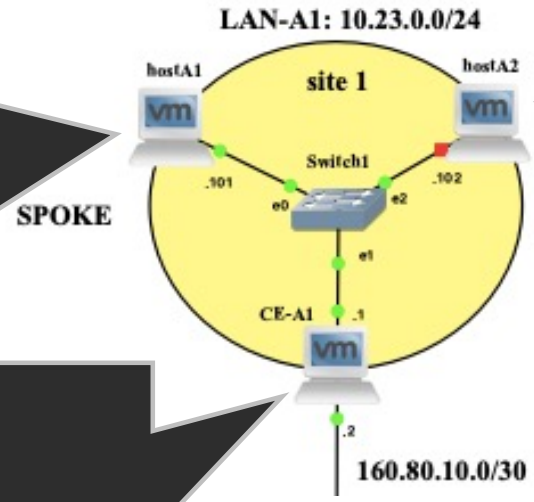
❑ MKA: MACsec Key Agreement

- ❖ ***SAK***: *MACsec Secure Association Keys*
- ❖ ***CAK***: *Connectivity Association Key*
- ❖ Utilizzo specifico
  - ➢ Sicurezza nei collegamenti *Ethernet* nella LAN-A1
  - ➢ *Static CAK mode*

MKA

```
export MKA_CAK=00112233445566778899aabbccddeeff
export MKA_CKN=000011112222333344445555666677…
```

```
nmcli connection add type macsec \
 con-name macsec-conf \
 ifname macsec0 \
 connection.autoconnect no \
 macsec.parent ens33 \
 macsec.mode psk \
 macsec.mka-cak $MKA_CKA \
 macsec.mka-cak-flags 0 \
 macsec.mka-ckn $MKA_CKN \
 ipv4.method manual ipv4.addresses 10.23.0.101/24

nmcli connection up macsec-conf
```




```
nmcli connection add type macsec \
 con-name macsec-conf \
 ifname macsec0 \
 connection.autoconnect no \
 macsec.parent ens33 \
 macsec.mode psk \
 macsec.mka-cak $MKA_CKA \
 macsec.mka-cak-flags 0 \
 macsec.mka-ckn $MKA_CKN \
 ipv4.method manual ipv4.addresses 10.23.0.102/24

nmcli connection up macsec-conf
```

```
nmcli connection add type macsec \
 con-name macsec-conf \
 ifname macsec0 \
 connection.autoconnect no \
 macsec.parent ens37 \
 macsec.mode psk \
 macsec.mka-cak $MKA_CKA \
 macsec.mka-cak-flags 0 \
 macsec.mka-ckn $MKA_CKN \
 ipv4.method manual ipv4.addresses 10.23.0.1/24

nmcli connection up macsec-conf
```

macsec0: interfaccia virtuale sopra interfaccia fisica (*ens33* o *ens37*)

Firewalls

Firewall & iptables

❑ Firewall

- ❖ Regolazione del traffico in ingresso/uscita
- ❖ Definizione delle regole di ACCEPT/DROP

❑ iptables

- ❖ Configurazione di *NetFilter*
- ❖ Gestione dei pacchetti in ingresso/uscita in base a:
  - ➢ Interfacce ingresso/uscita
  - ➢ Porte sorgente/destinazione
  - ➢ Protocolli
  - ➢ Indirizzi Ipv4 sorgente/destinazione

Firewalls richiesti

❑ CE-A1

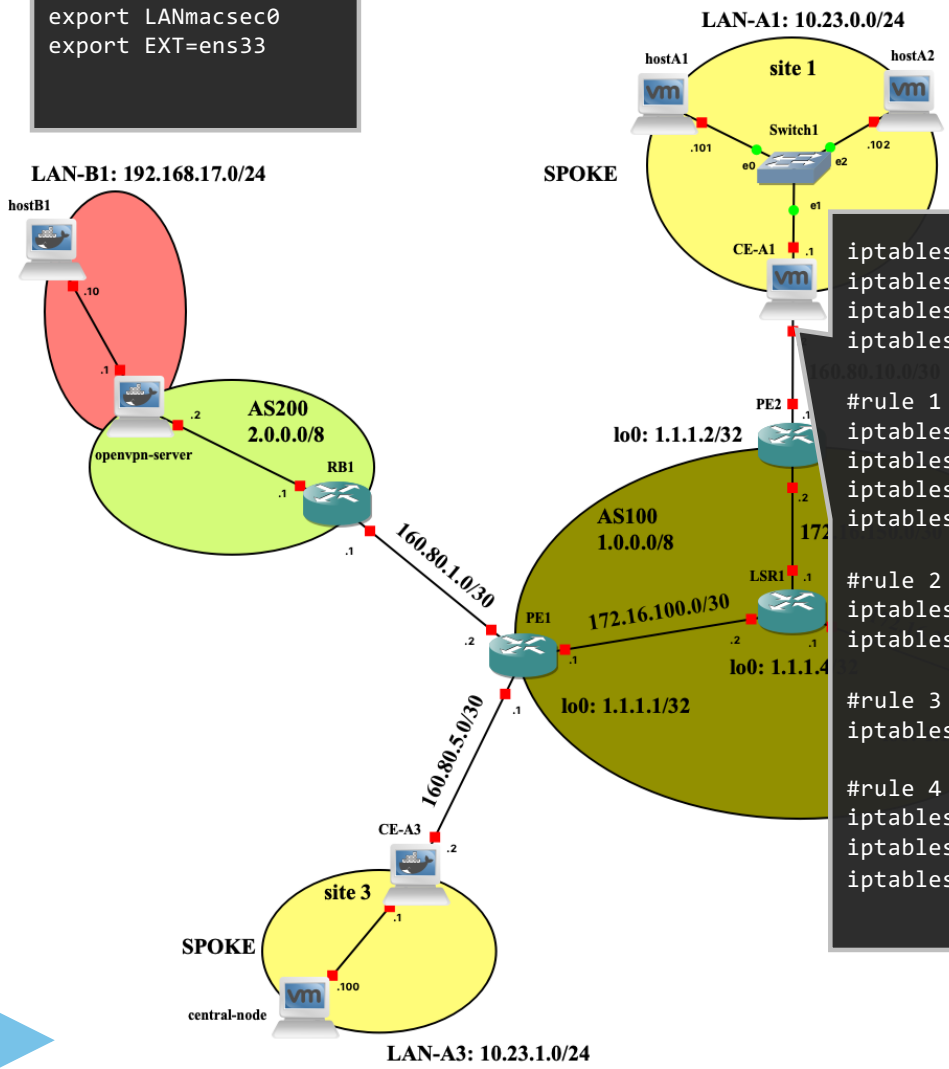
- ❖ Permettere il traffico tra la *LAN* e la rete esterna solo se iniziato dalla LAN, con **SNAT**
- ❖ Negare tutto il traffico verso il *GW*, tranne ***SSH*** e ***ICMP***, solo se iniziato dalla *LAN*.
- ❖ Permettere il traffico dal *GW* verso qualsiasi destinazione (e pacchetti di risposta correlati).
- ❖ Permettere il forwarding con ***DNAT*** verso *hostA1* e *hostA2* sono per l'**HTTP service**.

❑ CE-2

- ❖ Permettere la comunicazione bidirezionale end-to-end tra il *central-node* e gli *AVs* e negare tutto il resto.

CE-A1 Firewall

```
export LANmacsec0
export EXT=ens33
```

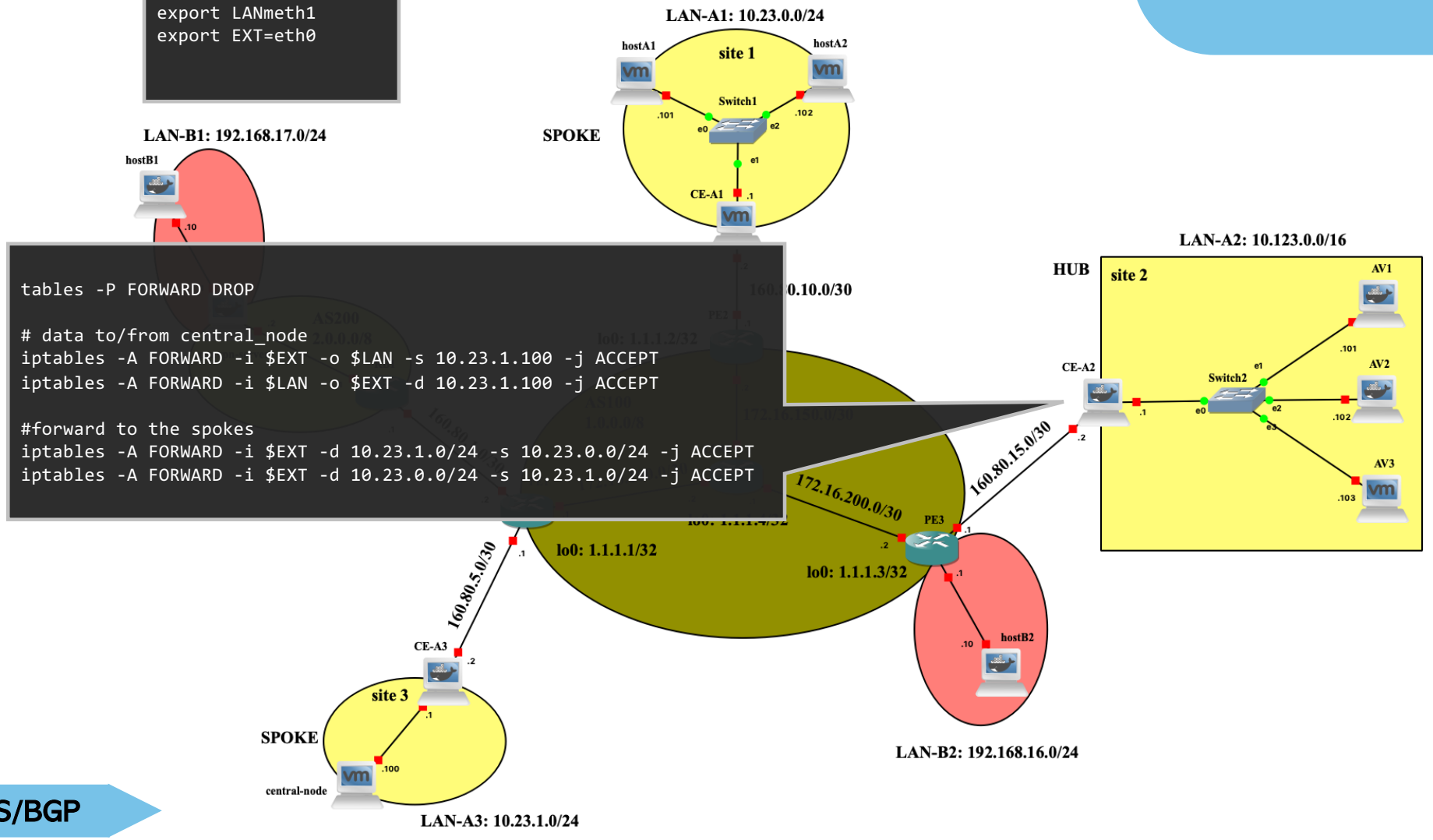



```
iptables -F
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT

#rule 1
iptables -A FORWARD -i $LAN -o $EXT -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED
iptables -t nat -A POSTROUTING -o $EXT -s 10.23.0.101 -j SNAT --to 10.23.0.10
iptables -t nat -A POSTROUTING -o $EXT -s 10.23.0.102 -j SNAT --to 10.23.0.20

#rule 2
iptables -A INPUT -i $LAN -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -i $LAN -p icmp -j ACCEPT

#rule 3
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

#rule 4
iptables -A FORWARD -i $EXT -o $LAN -p tcp --dport 80 -j ACCEPT
iptables -t nat -A PREROUTING -i $EXT -d 10.23.0.10 -j DNAT --to 10.23.0.101
iptables -t nat -A PREROUTING -i $EXT -d 10.23.0.20 -j DNAT --to 10.23.0.102
```

LAN-B2: 192.168.16.0/24

CE-A1 Firewall

```
export LANmeth1
export EXT=eth0
```

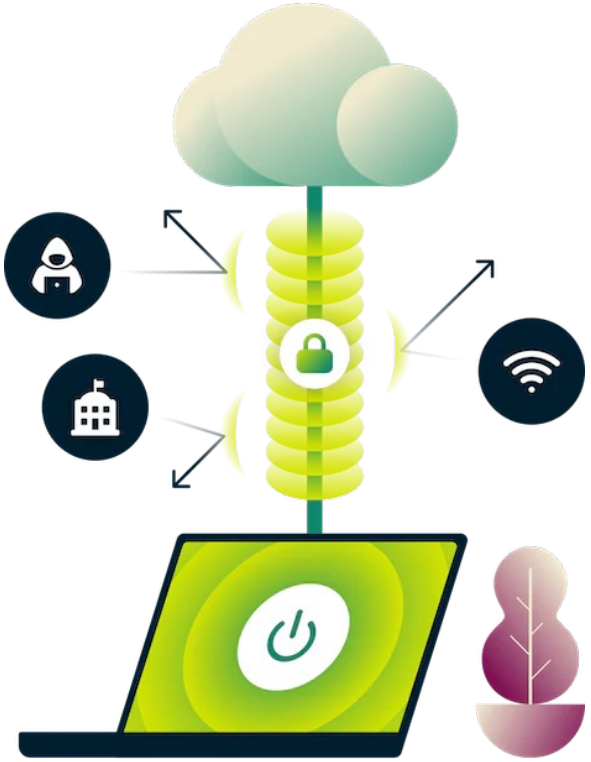



```
tables -P FORWARD DROP

# data to/from central_node
iptables -A FORWARD -i $EXT -o $LAN -s 10.23.1.100 -j ACCEPT
iptables -A FORWARD -i $LAN -o $EXT -d 10.23.1.100 -j ACCEPT

#forward to the spokes
iptables -A FORWARD -i $EXT -d 10.23.1.0/24 -s 10.23.0.0/24 -j ACCEPT
iptables -A FORWARD -i $EXT -d 10.23.0.0/24 -s 10.23.1.0/24 -j ACCEPT
```

OPENVPN

❑ OPENVPN

- ❖ Software open-source per la creazione di reti private virtuali (VPN).
- ❖ Fornisce un tunnel crittografato tra client e server, consentendo la trasmissione sicura dei dati su reti non sicure.

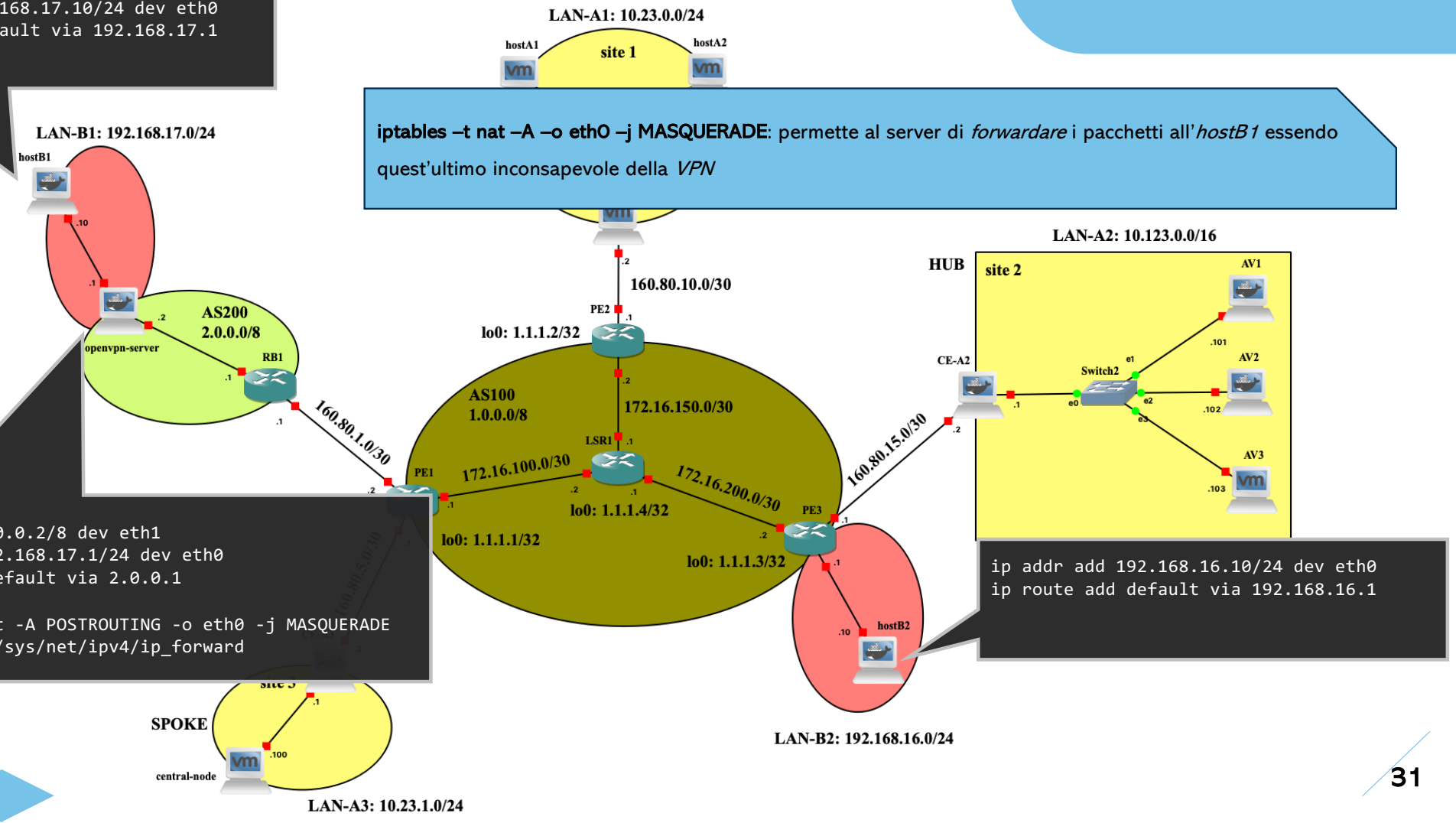


❑ *Configurazione*

- ❖ Configurazione della *Certificate Authority (CA):*
  - ➢ Creazione di una *CA* con ***OpenSSL***.
- ❖ Generazione e firma delle chiavi:
  - ✓ Generazione delle coppie di chiavi pubbliche/ private per il server *OpenVPN* e per gli *host* client.
  - ✓ Firma delle chiavi client utilizzando la CA per autenticazione e verifica.
- ❖ Configurazione della connessione *OPENVPN*
  - ✓ `openvpn server.ovpn`
  - ✓ `openvpn hostB2.ovpn`

Lan-B2 Network

```
ip addr add 192.168.17.10/24 dev eth0
ip route add default via 192.168.17.1
```



iptables –t nat –A –o eth0 –j MASQUERADE: permette al server di *forwardare* i pacchetti all'*hostB1* essendo quest'ultimo inconsapevole della *VPN*

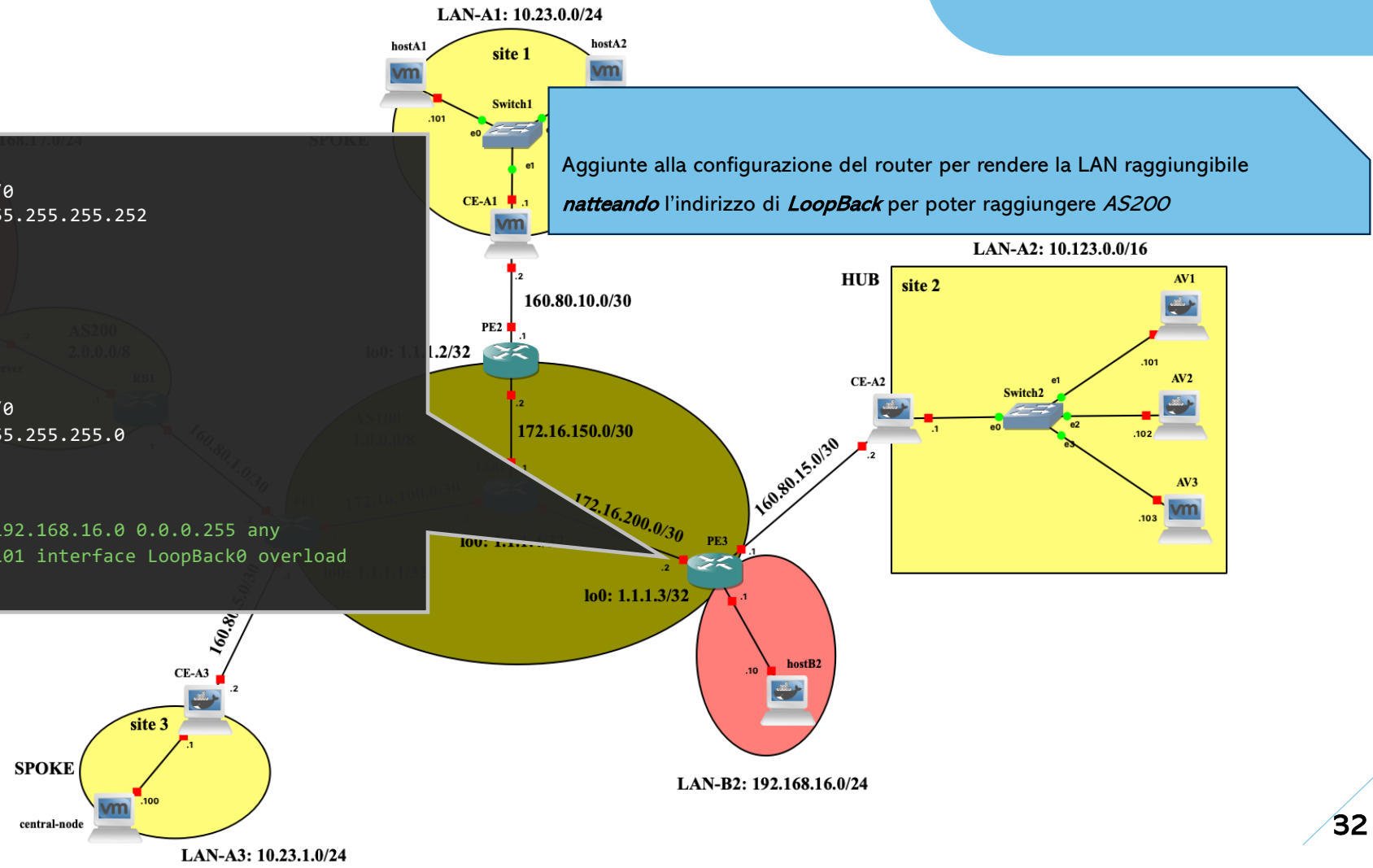
```
ip addr add 2.0.0.2/8 dev eth1
ip addr add 192.168.17.1/24 dev eth0
ip route add default via 2.0.0.1

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
ip addr add 192.168.16.10/24 dev eth0
ip route add default via 192.168.16.1
```

Lan-B2 Network

```
interface GigabitEthernet1/0
 ip address 172.16.200.2 255.255.255.252
 ip nat outside
 mpls ip
!
.
.
.
!
interface GigabitEthernet3/0
 ip address 192.168.16.1 255.255.255.0
 ip nat inside
!

access-list 101 permit ip 192.168.16.0 0.0.0.255 any
ip nat inside source list 101 interface LoopBack0 overload
```

Aggiunte alla configurazione del router per rendere la LAN raggiungibile *natteando* l'indirizzo di *LoopBack* per poter raggiungere *AS200*



CA & Keys

❖ Creazione del certificato e delle chiavi pubbliche e private

- openSSL, easyRSA

1. Creazione del certificato e della chiave privata della CA
2. Creazione del certificato e della chiave privata del Server
3. Creazione dei parametri Diffie-Hellman
4. Creazione del certificato e della chiave privata del client

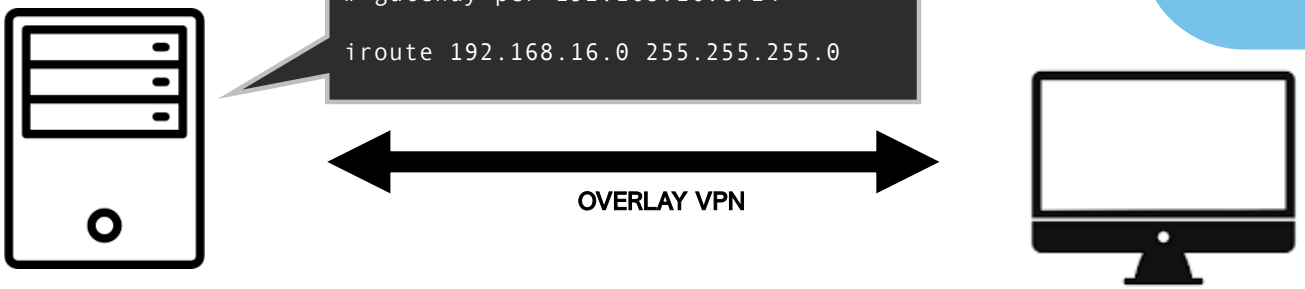
```
# ca and key build

cd /usr/share/easy-rsa
cp openssl-1.0.0.cnf openssl.cnf
. ./vars
./clean-all
./build-ca
./build-key-server server
./build-dh
./build-key hostB2
```

OPENVPN

OPENVPN config

/gns3volumes/share_openvpn/keys/ccd/**hostB2**

```
# Permette al server di agire da
# gateway per 192.168.16.0/24

iroute 192.168.16.0 255.255.255.0
```



/gns3volumes/share_openvpn/keys/**server.ovpn**

```
port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key
dh dh2048.pem
server 192.168.100.0 255.255.255.0
push "route 192.168.17.0 255.255.255.0"
route 192.168.16.0 255.255.255.0
client-config-dir ccd
keepalive 10 120
cipher AES-256-CBC
```

/gns3volumes/share_openvpn/**hostB2.ovpn**

```
client
dev tun
proto udp
remote 2.0.0.2 1194
resolv-retry infinite
ca ca.crt
cert hostB2.crt
key hostB2.key
remote-cert-tls server
cipher AES-256-CBC
```

Antivirus

❑ *Antivirus*

- ❖ Software per rilevare, prevenire e rimuovere *malware*
- ❖ Come?
  - ➢ Scansione dei file (e.g .elf, .exe)
  - ➢ Monitoraggio in tempo reale
  - ➢ Rilevazione delle ***signatures***
  - ➢ …
- ❖ ***Antivirus*** utilizzati
  - ➢ *ClamAV*
  - ➢ *Loki*
  - ➢ *RKHunter*

❑ *ClamAV*

- ❖ *https://docs.clamav.net*
- ❖ *Toolkit antivirus open source (GPLv2)* progettato specificamente per la scansione delle email sui gateway di posta.
  - Adattabile per ogni tipo di *file*
- ❖ Fornisce diverse utility: un demone *multithreaded* flessibile e scalabile, uno scanner a riga di comando e uno strumento avanzato per gli aggiornamenti automatici del database delle *signature*
- ❖ Altamente flessibile e scalabile, adatto per l'implementazione su larga scala.

❑ *Loki*

- *https://github.com/Neo23x0/Loki*
- open-source IOC and **YARA** scanner
- Scritto in *Python*
- ***YARA***
  - multi-piattaforma che può essere eseguita su Windows, Linux e Mac OS X. Utilizzato attraverso CLI o tramite API Python utilizzando l'estensione ***yara-python***.
  - Strumento per l'identificazione e la classificazione di campioni di malware
  - Permette di creare descrizioni di famiglie di malware basate su pattern testuali o binari

❑ *RKHunter*

- ❖ https://rkhunter.sourceforge.net
- ❖ Esegue una scansione dei file di sistema alla ricerca di anomalie o firme associate ai ***rootkit*** noti. Questo controllo può rilevare modifiche non autorizzate o indicatori di compromissione nel sistema.
- ❖ ***Rootkit***
  - Forme di ***malware*** sofisticate che mirano ad ottenere accesso non autorizzato a un sistema informatico
  - Sfruttano le vulnerabilità di sicurezza per nascondersi e operare in modo invisibile agli utenti e agli strumenti di sicurezza.

Codice

```
echo "Clamav, Waiting for a new file to analyze!"
python3 receive.py malware 1234

clamscan malware > log1.log
nc -q 10 10.23.1.100 1111 < log1.log
rm malware
```

Clamav è in grado di eseguire la scansione direttamente sul file.

Loki esegue la scansione sulle directory, di conseguenze il file viene spostato nel directory *target* della scansione.

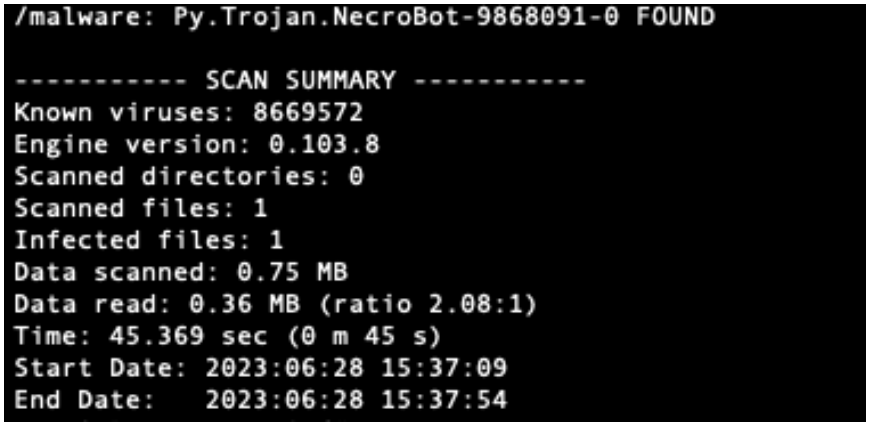
```
echo "Loki, Waiting for a new file to analyze!"
python3 receive.py malware 1234
mv malware /mw_to_scan
python3 loki.py -p /mw_to_scan > log2.log

python3 send.py log2.log 10.23.1.100 2222
rm /mw_to_scan/malware
```

```
echo "RKHunter, Waiting for a new file to analyze!"
python3 receive.py malware 1234

sudo chmod +x malware
./malware &
sleep 5

sudo rkhunter -c --rwo --sk --summary > log3.log
python3 send.py log3.log 10.23.1.100 3333
```

RKHunter controlla i processi in esecuzione nel sistema, di conseguenza, prima della scansione, il *file* sospetto viene eseguito.

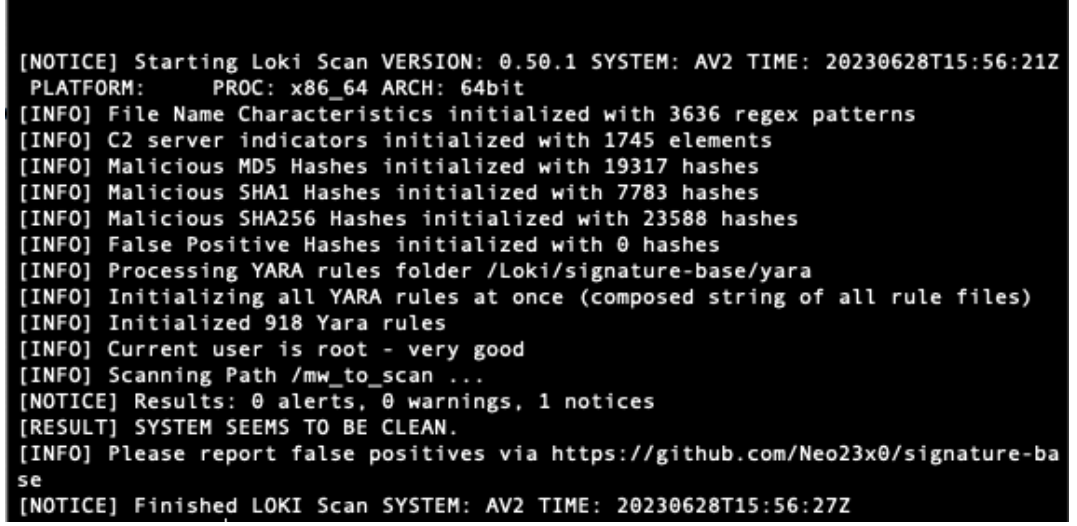
Risultati

Clamav è in grado di effettuare un confronto con il suo *database* di 8669572 virus conosciuto e riesce facilmente a classificare un file come *»infected».* Questo *AV* è, infatti, il più consigliato dagli esperti di *Cybersecurity* per via sia della sua natura *OpenSource* che per la sua forte flessibilità



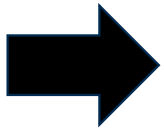
Risultati

Loki, a differenza di *Clamav*, non è in grado di classificare lo stesso *file* come malevolo. Questo probabilmente perché l'AV in questione è obsoleto. Esiste una sua nuova versione: ***Thor*** (https://www.nextron-systems.com/thor-lite/)



Risultati

RKHunter viene eseguito all'interno di una *sandbox* in modo tale da evitare (o quantomeno limitare) il fatto che eventuali *rootkit* possano estendersi nel sistema. Si è utilizzato come ambiente di esecuzione *Linux Lite.* Dai vari output si nota che l'AV tende a sovrastimare le possibili minacce.



```
System checks summary
=====================

File properties checks...
    Files checked: 147
    Suspect files: 6

Rootkit checks...
    Rootkits checked : 480
    Possible rootkits: 2
    Rootkit names    : Spam tool component

Applications checks...
    All checks skipped

The system checks took: 4 minutes and 9 seconds
```

Considerazioni

- I ***malware*** sono stati scaricati da librerie *Open Source*
  - https://github.com/Pyran1/MalwareDatabase/
  - https://github.com/MalwareSamples/Linux-Malware-Samples
  - https://www.vx-underground.org
  - https://bazaar.abuse.ch/browse/
- Sulla *GNS3 VM* non c'è possibilità di effettuare *snapshot*
  - Motivo per la scelta di *Linux Lite* su *VMWare* per l'esecuzione e l'analisi dei *rootkit*
- Altro motivo per l'utilizzo di ***Linux Lite*** su ***VMWare***, piuttosto che un *container Docker,* per l'esecuzione di ***RKHunter***, è dato dal fatto che i *rootkit* sono progettati per operare a livello di sistema e manipolare il *kernel* o i componenti dell'OS.

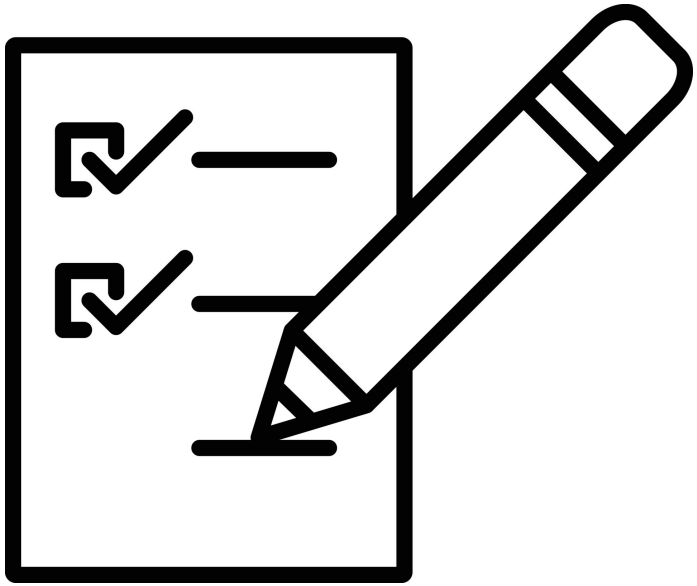
Considerazioni

- Sono stati testati anche altri *Antivirus,* in particolare:
  - ***Maldetect*** (https://github.com/waja/maldetect)
  - ***Multiscanner*** (https://github.com/mitre/multiscanner)
  - ***Kicomav*** (https://www.kicomav.com)
  - ***Chkrootkit*** (https://www.chkrootkit.org)
- Non utilizzati a causa della presenza di alcuni problemi di configurazione e compatibilità oppure di analogia nel funzionamento.
  - *Maldetect* utilizza *Clamav.*
  - *ChkRootkit* ha lo stesso obiettivo di *RKHunter.*

Test

Test effettuati

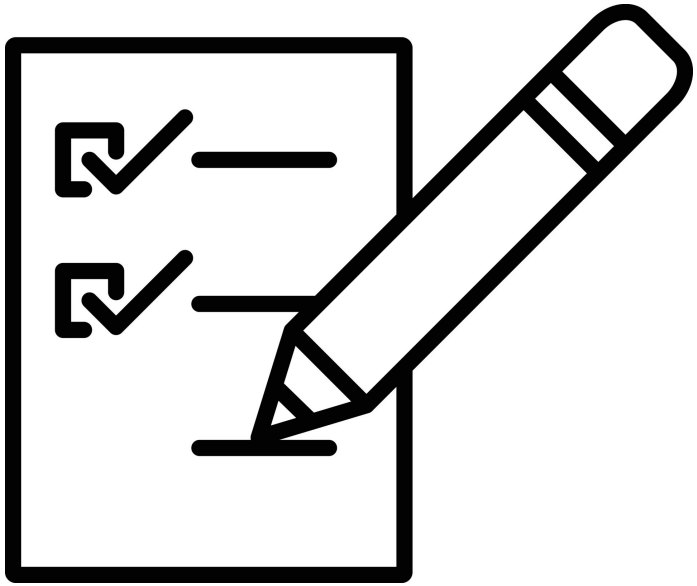
- Funzionamento corretto della configurazione *Hub & Spoke* della *VPN MPLS/BGP*
  1. *Label switching*
  2. *Spoke to Spoke communication*
- Funzionamento corretto della configurazione di *MacSec Key Agreement*
  1. Scambio chiavi
  2. Verifica di incapsulamento all'interno di *frame MacSec*



N.B: i test sono stati eseguiti con tutte le configurazioni attive, compresi i firewall.

Test effettuati

- Funzionamento corretto della configurazione dei *Firewall*
  1. *Ping hostA1* dall'esterno.
  2. *Ping CE-A3 da LAN-A1*
  3. Verifica che unica porta *TCP* aperta in hostA1 è 80.
  4. *LAB-B2* raggiungibile solo da *central-node.*
- Funzionamento corretto della configurazione di *OpenVPN*
  1. *TLS handshake.*
  2. *LAN-B1* raggiungibile, tramite *OpenVpn,* da *LAN-B2.*



Test

N.B: i test sono stati eseguiti con tutte le configurazioni attive, compresi i firewall.

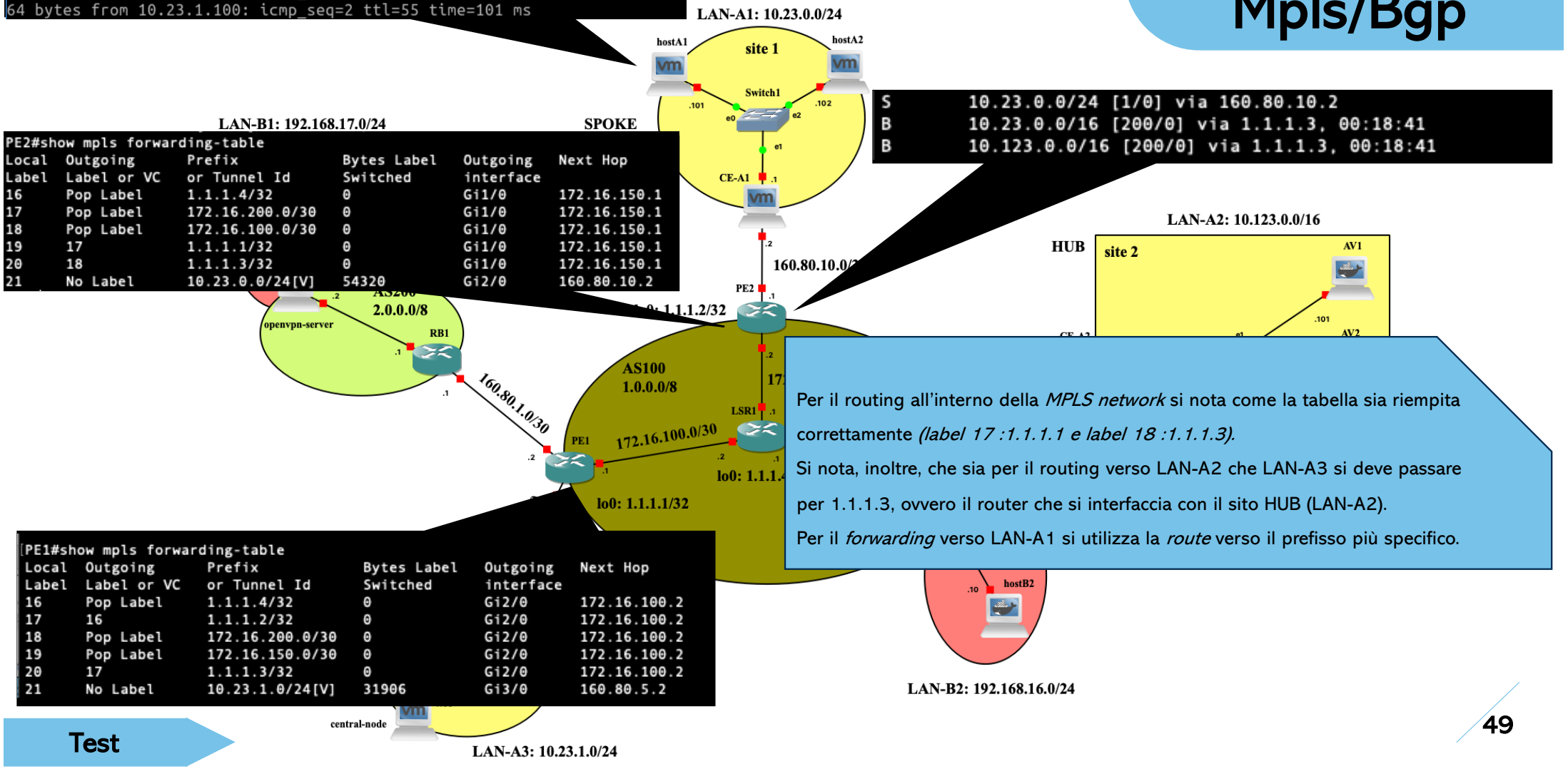
Test Vpn Mpls/Bgp

```
mattintosh@mattintosh-vmwarevirtualplatform:~/Desktop$ ping 10.23.1.100
PING 10.23.1.100 (10.23.1.100) 56(84) bytes of data.
64 bytes from 10.23.1.100: icmp_seq=1 ttl=55 time=127 ms
64 bytes from 10.23.1.100: icmp_seq=2 ttl=55 time=101 ms
```

```
S       10.23.0.0/24 [1/0] via 160.80.10.2
B       10.23.0.0/16 [200/0] via 1.1.1.3, 00:18:41
B       10.123.0.0/16 [200/0] via 1.1.1.3, 00:18:41
```

```
PE2#show mpls forwarding-table
Local  Outgoing      Prefix           Bytes Label   Outgoing    Next Hop
Label  Label or VC   or Tunnel Id     Switched      interface
16     Pop Label     1.1.1.4/32       0             Gi1/0       172.16.150.1
17     Pop Label     172.16.200.0/30  0             Gi1/0       172.16.150.1
18     Pop Label     172.16.100.0/30  0             Gi1/0       172.16.150.1
19     17            1.1.1.1/32       0             Gi1/0       172.16.150.1
20     18            1.1.1.3/32       0             Gi1/0       172.16.150.1
21     No Label      10.23.0.0/24[V]  54320         Gi2/0       160.80.10.2
```

```
PE1#show mpls forwarding-table
Local  Outgoing      Prefix           Bytes Label   Outgoing    Next Hop
Label  Label or VC   or Tunnel Id     Switched      interface
16     Pop Label     1.1.1.4/32       0             Gi2/0       172.16.100.2
17     16            1.1.1.2/32       0             Gi2/0       172.16.100.2
18     Pop Label     172.16.200.0/30  0             Gi2/0       172.16.100.2
19     Pop Label     172.16.150.0/30  0             Gi2/0       172.16.100.2
20     17            1.1.1.3/32       0             Gi2/0       172.16.100.2
21     No Label      10.23.1.0/24[V]  31906         Gi3/0       160.80.5.2
```



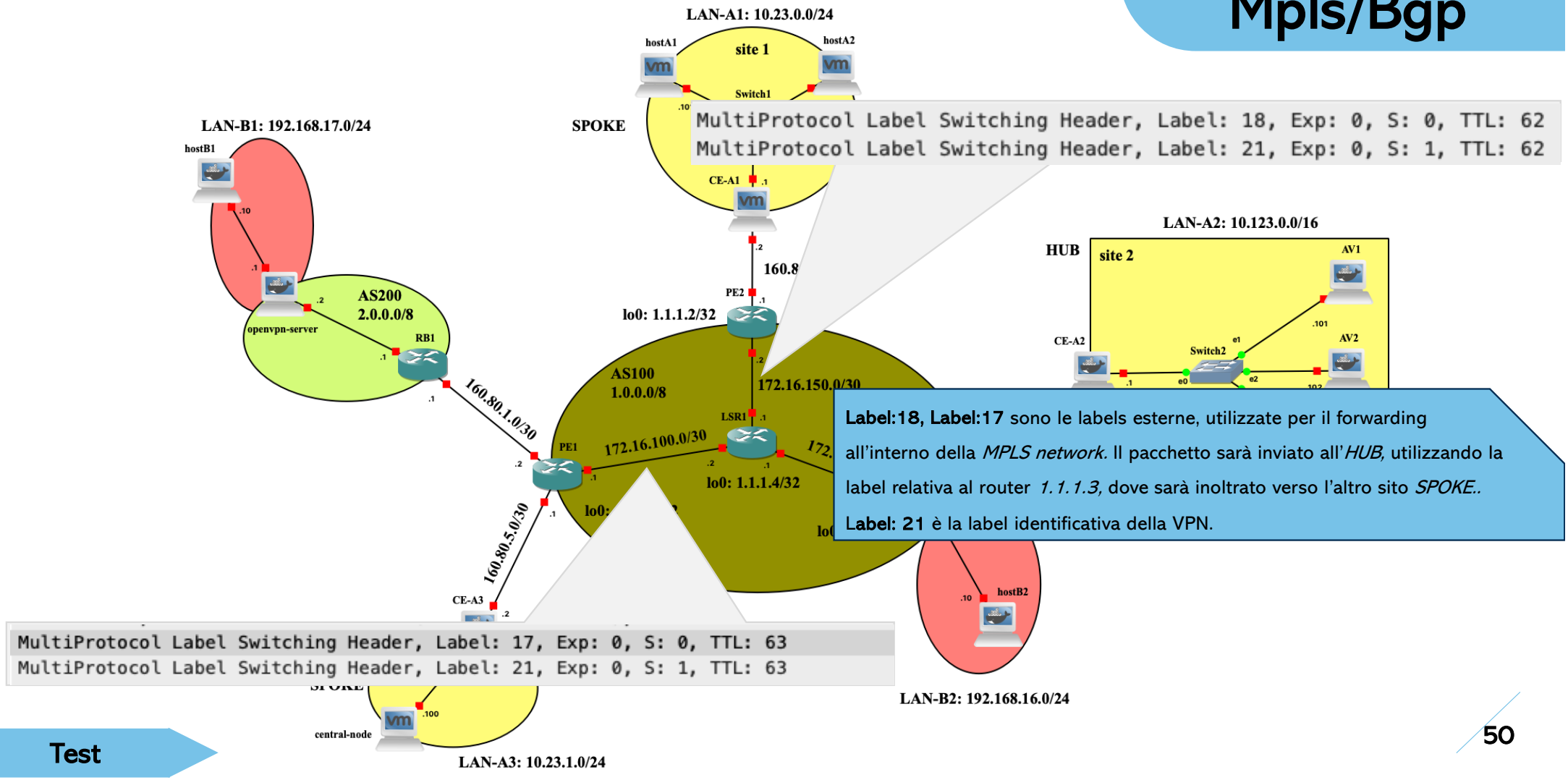
Per il routing all'interno della *MPLS network* si nota come la tabella sia riempita correttamente *(label 17 :1.1.1.1 e label 18 :1.1.1.3).*

Si nota, inoltre, che sia per il routing verso LAN-A2 che LAN-A3 si deve passare per 1.1.1.3, ovvero il router che si interfaccia con il sito HUB (LAN-A2).

Per il *forwarding* verso LAN-A1 si utilizza la *route* verso il prefisso più specifico.

Test

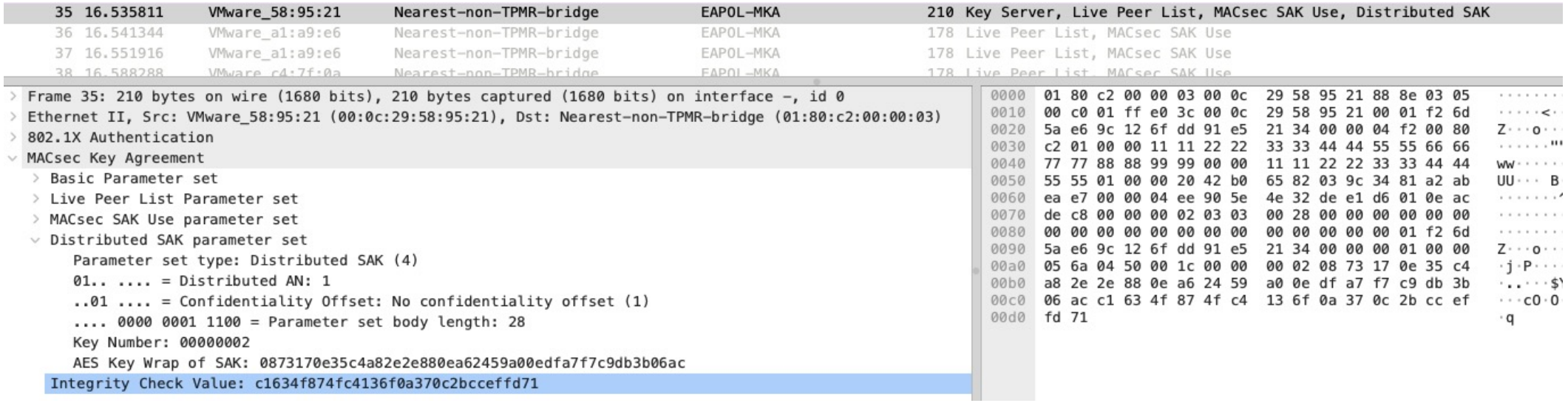
Test Vpn Mpls/Bgp



Label:18, Label:17 sono le labels esterne, utilizzate per il forwarding all'interno della *MPLS network.* Il pacchetto sarà inviato all'*HUB,* utilizzando la label relativa al router *1.1.1.3,* dove sarà inoltrato verso l'altro sito *SPOKE..*

Label: 21 è la label identificativa della VPN.

Test MKA

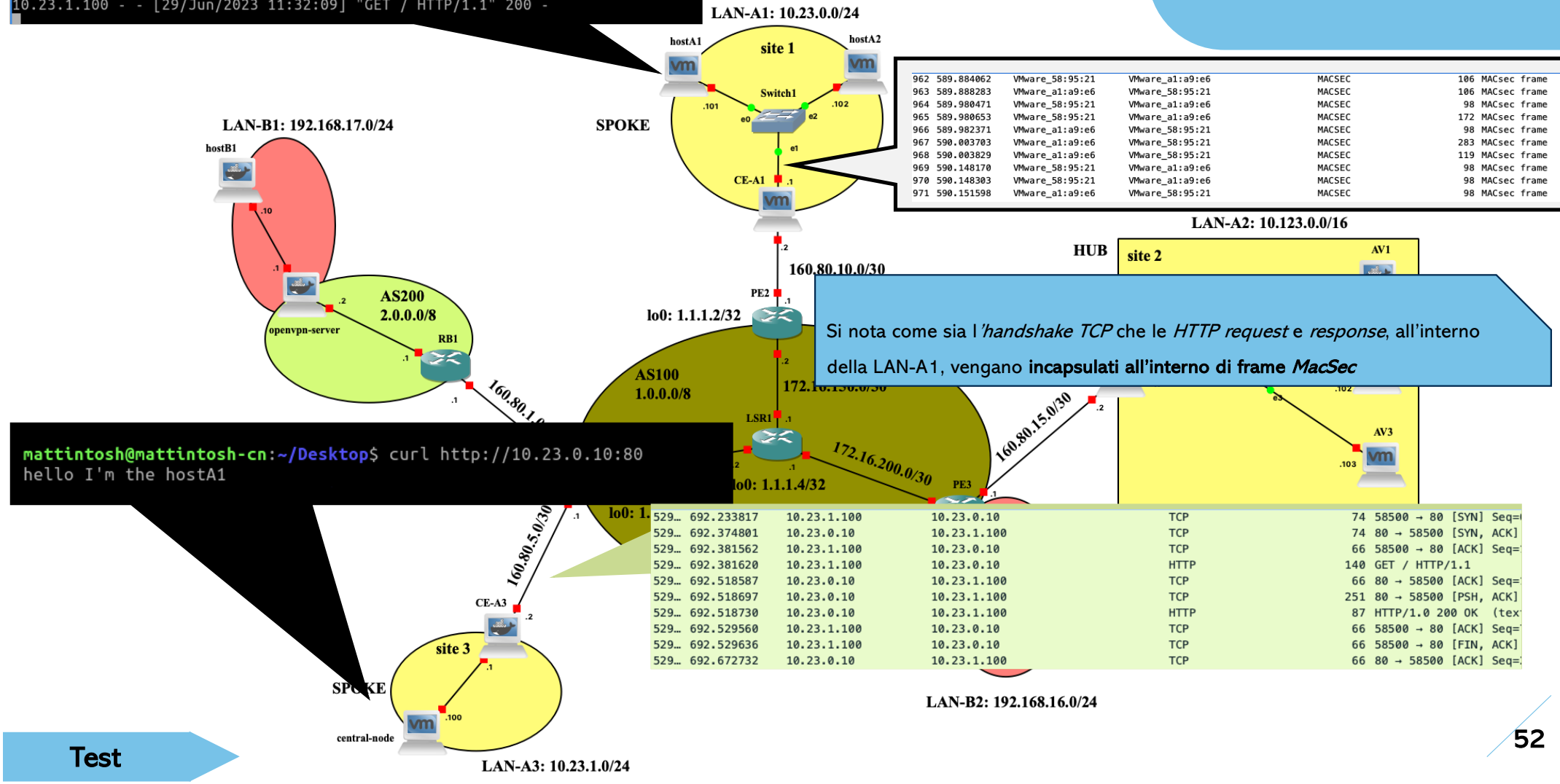


- **MACsec SAK Use parameter set**: Set di parametri che specifica l'uso della SAK.
- **Distributed SAK parameter set**: Set di parametri che indica l'uso di una SAK distribuita.
- **AES Key Wrap of SAK**: Chiave di crittografia *AES* utilizzata per il *wrapping* della SAK.
- **Integrity Check Value**: Valore utilizzato per il controllo di integrità dei dati.

N.B: AES Key Wrap è un algoritmo di crittografia che viene utilizzato per proteggere una chiave di crittografia simmetrica.

Test MKA

```
mattintosh@mattintosh-vmwarevirtualplatform:~/Desktop/http_test$ sudo python3
 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.23.1.100 - - [29/Jun/2023 11:32:09] "GET / HTTP/1.1" 200 -
```



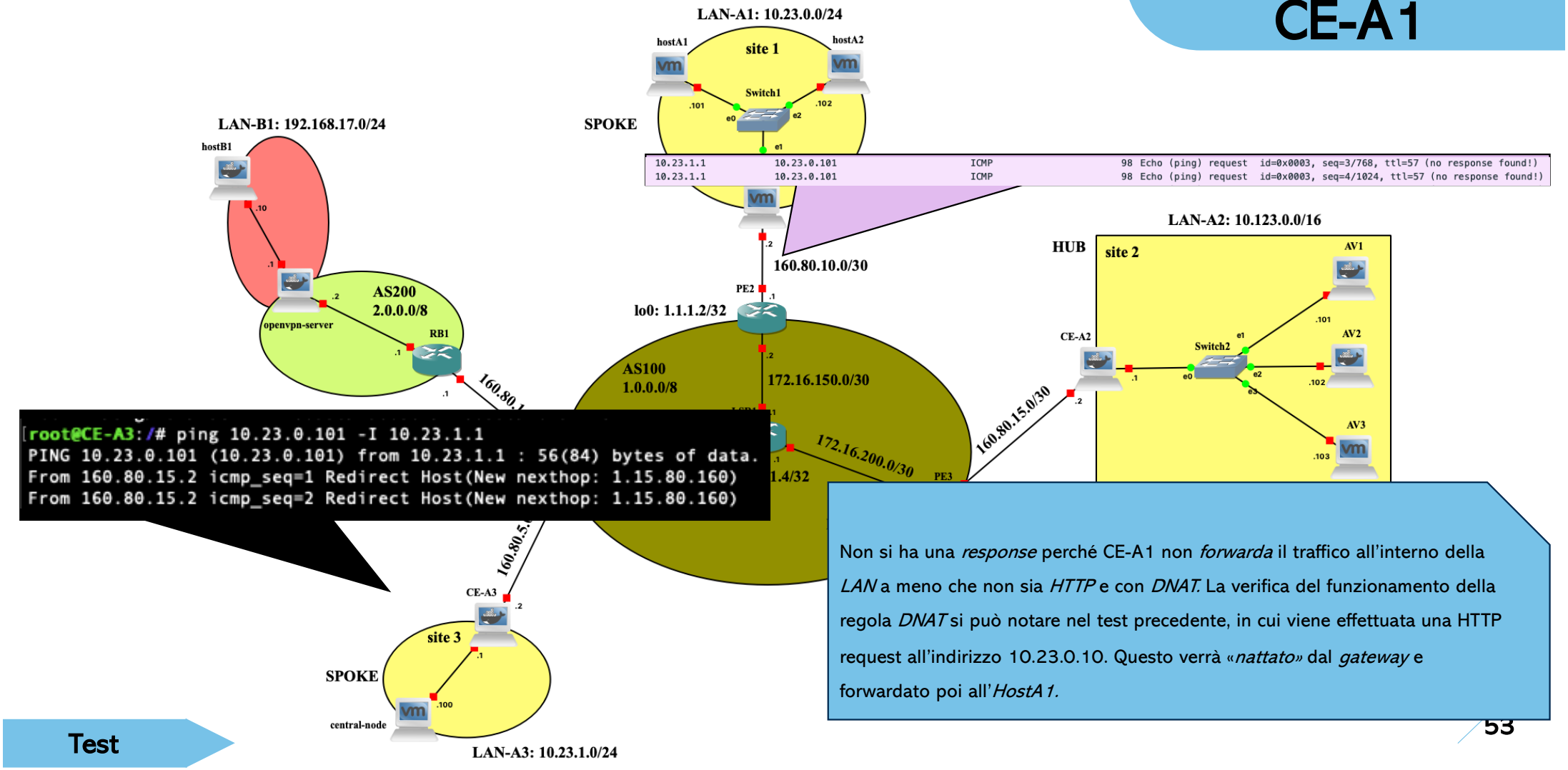
962	589.884062	VMware_58:95:21	VMware_a1:a9:e6	MACSEC	106 MACsec frame
963	589.888283	VMware_a1:a9:e6	VMware_58:95:21	MACSEC	106 MACsec frame
964	589.980471	VMware_58:95:21	VMware_a1:a9:e6	MACSEC	98 MACsec frame
965	589.980653	VMware_58:95:21	VMware_a1:a9:e6	MACSEC	172 MACsec frame
966	589.982371	VMware_a1:a9:e6	VMware_58:95:21	MACSEC	98 MACsec frame
967	590.003703	VMware_a1:a9:e6	VMware_58:95:21	MACSEC	283 MACsec frame
968	590.003829	VMware_a1:a9:e6	VMware_58:95:21	MACSEC	119 MACsec frame
969	590.148170	VMware_58:95:21	VMware_a1:a9:e6	MACSEC	98 MACsec frame
970	590.148303	VMware_a1:a9:e6	VMware_58:95:21	MACSEC	98 MACsec frame
971	590.151598	VMware_a1:a9:e6	VMware_58:95:21	MACSEC	98 MACsec frame

Si nota come sia l'*handshake TCP* che le *HTTP request* e *response*, all'interno della LAN-A1, vengano **incapsulati all'interno di frame *MacSec***

```
mattintosh@mattintosh-cn:~/Desktop$ curl http://10.23.0.10:80
hello I'm the hostA1
```

529…	692.233817	10.23.1.100	10.23.0.10	TCP	74 58500 → 80 [SYN] Seq=
529…	692.374801	10.23.0.10	10.23.1.100	TCP	74 80 → 58500 [SYN, ACK]
529…	692.381562	10.23.1.100	10.23.0.10	TCP	66 58500 → 80 [ACK] Seq=
529…	692.381620	10.23.1.100	10.23.0.10	HTTP	140 GET / HTTP/1.1
529…	692.518587	10.23.0.10	10.23.1.100	TCP	66 80 → 58500 [ACK] Seq=
529…	692.518697	10.23.0.10	10.23.1.100	TCP	251 80 → 58500 [PSH, ACK]
529…	692.518730	10.23.0.10	10.23.1.100	HTTP	87 HTTP/1.0 200 OK (tex
529…	692.529560	10.23.1.100	10.23.0.10	TCP	66 58500 → 80 [ACK] Seq=
529…	692.529636	10.23.1.100	10.23.0.10	TCP	66 58500 → 80 [FIN, ACK]
529…	692.672732	10.23.0.10	10.23.1.100	TCP	66 80 → 58500 [ACK] Seq=

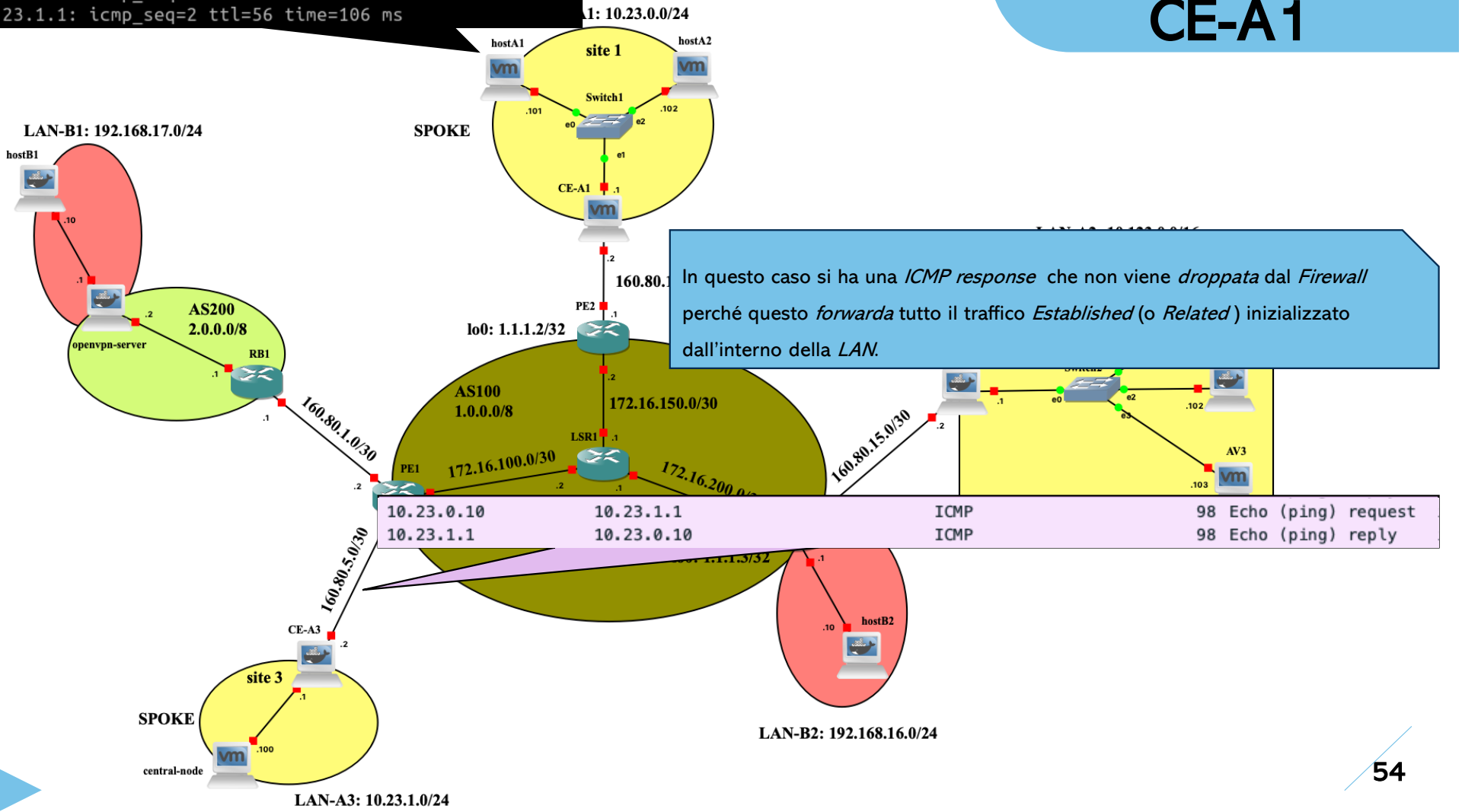
Test Firewall CE-A1



Non si ha una *response* perché CE-A1 non *forwarda* il traffico all'interno della *LAN* a meno che non sia *HTTP* e con *DNAT.* La verifica del funzionamento della regola *DNAT* si può notare nel test precedente, in cui viene effettuata una HTTP request all'indirizzo 10.23.0.10. Questo verrà «*nattato*» dal *gateway* e forwardato poi all'*HostA1.*

Test Firewall CE-A1

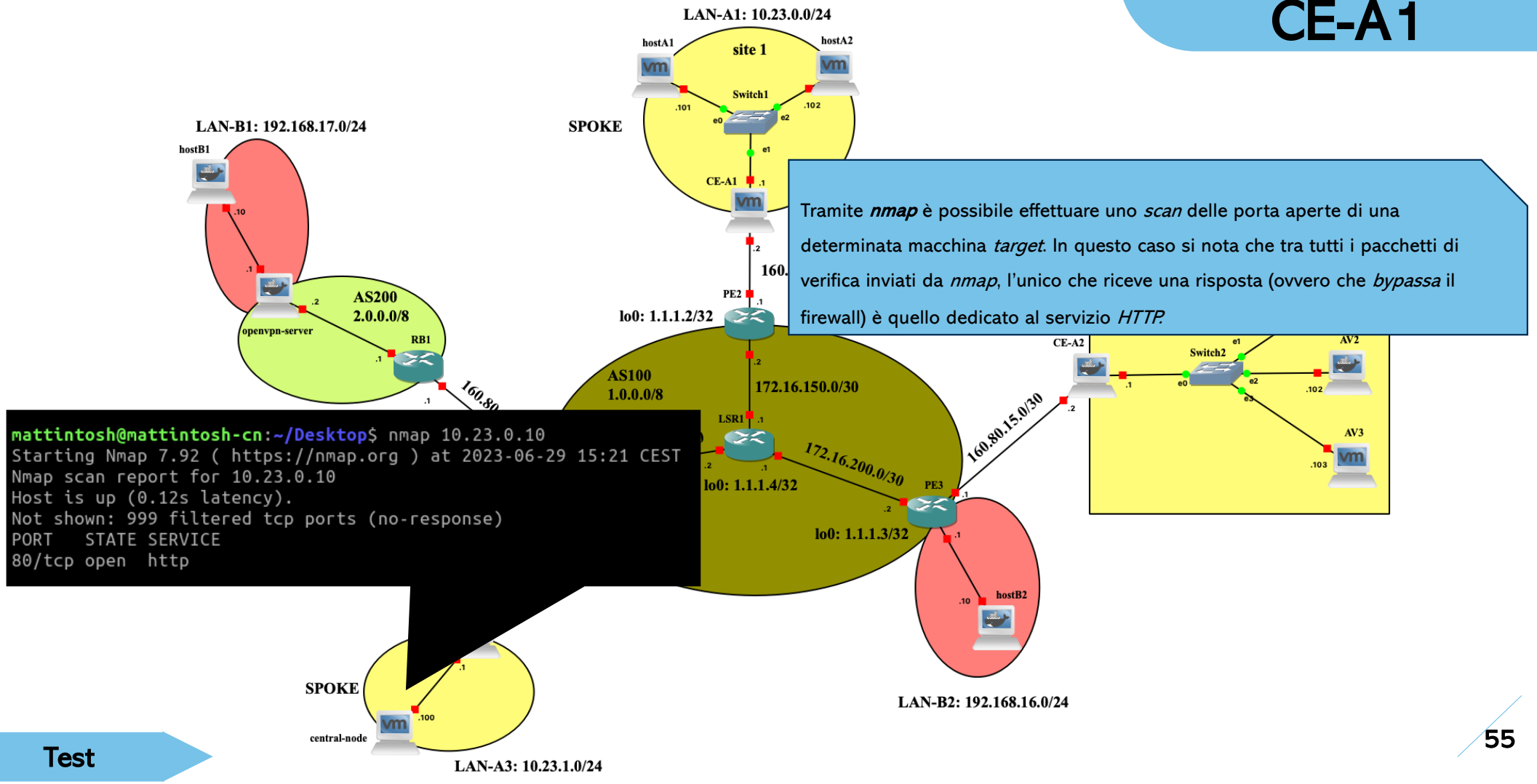
```
mattintosh@mattintosh-vmwarevirtualplatform:~/Desktop$ ping 10.23.1.1
PING 10.23.1.1 (10.23.1.1) 56(84) bytes of data.
64 bytes from 10.23.1.1: icmp_seq=1 ttl=56 time=126 ms
64 bytes from 10.23.1.1: icmp_seq=2 ttl=56 time=106 ms
```



10.23.0.10	10.23.1.1	ICMP	98 Echo (ping) request
10.23.1.1	10.23.0.10	ICMP	98 Echo (ping) reply

In questo caso si ha una *ICMP response* che non viene *droppata* dal *Firewall* perché questo *forwarda* tutto il traffico *Established* (o *Related*) inizializzato dall'interno della *LAN*.

Test Firewall CE-A1

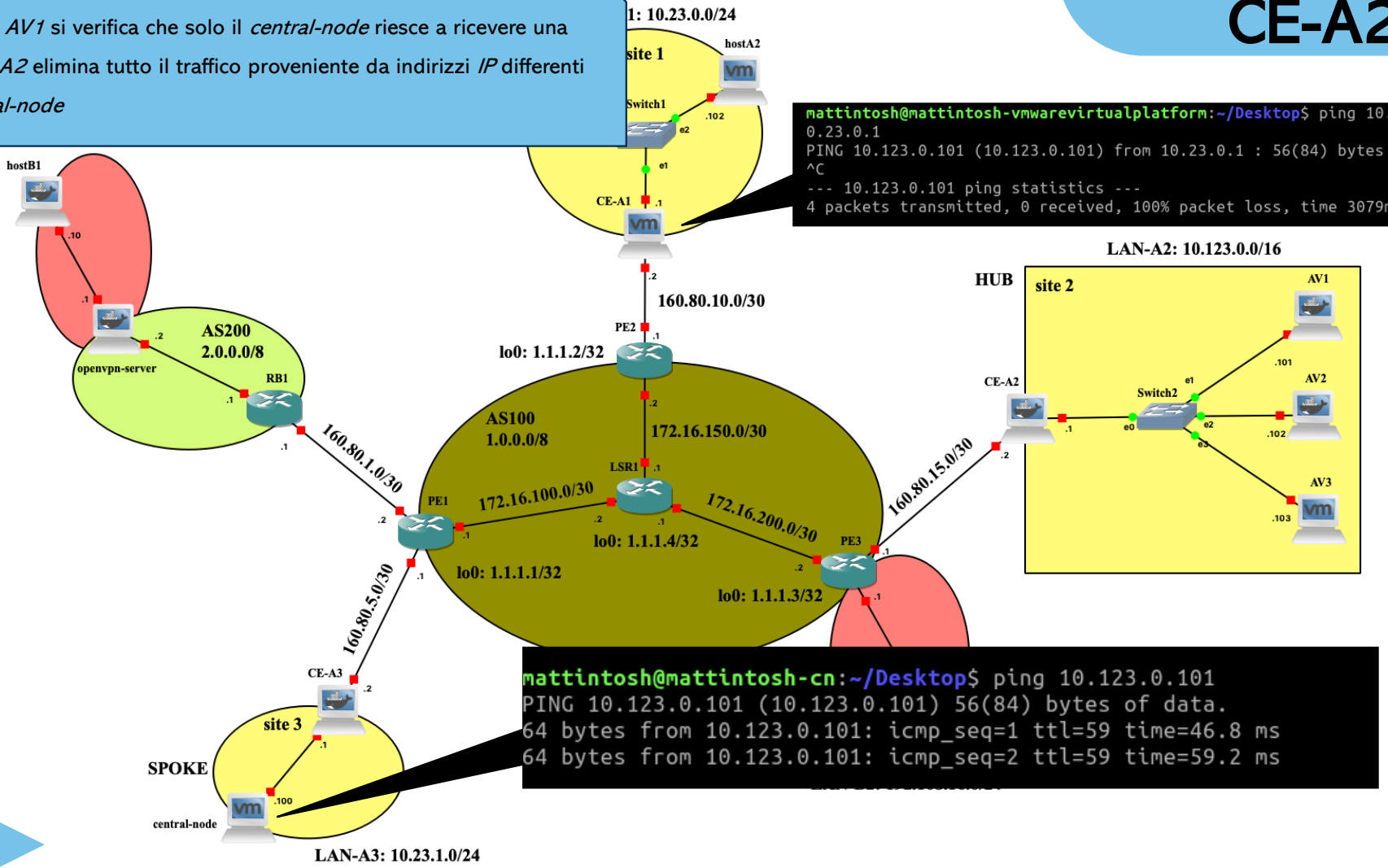



```
mattintosh@mattintosh-cn:~/Desktop$ nmap 10.23.0.10
Starting Nmap 7.92 ( https://nmap.org ) at 2023-06-29 15:21 CEST
Nmap scan report for 10.23.0.10
Host is up (0.12s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT   STATE SERVICE
80/tcp open  http
```

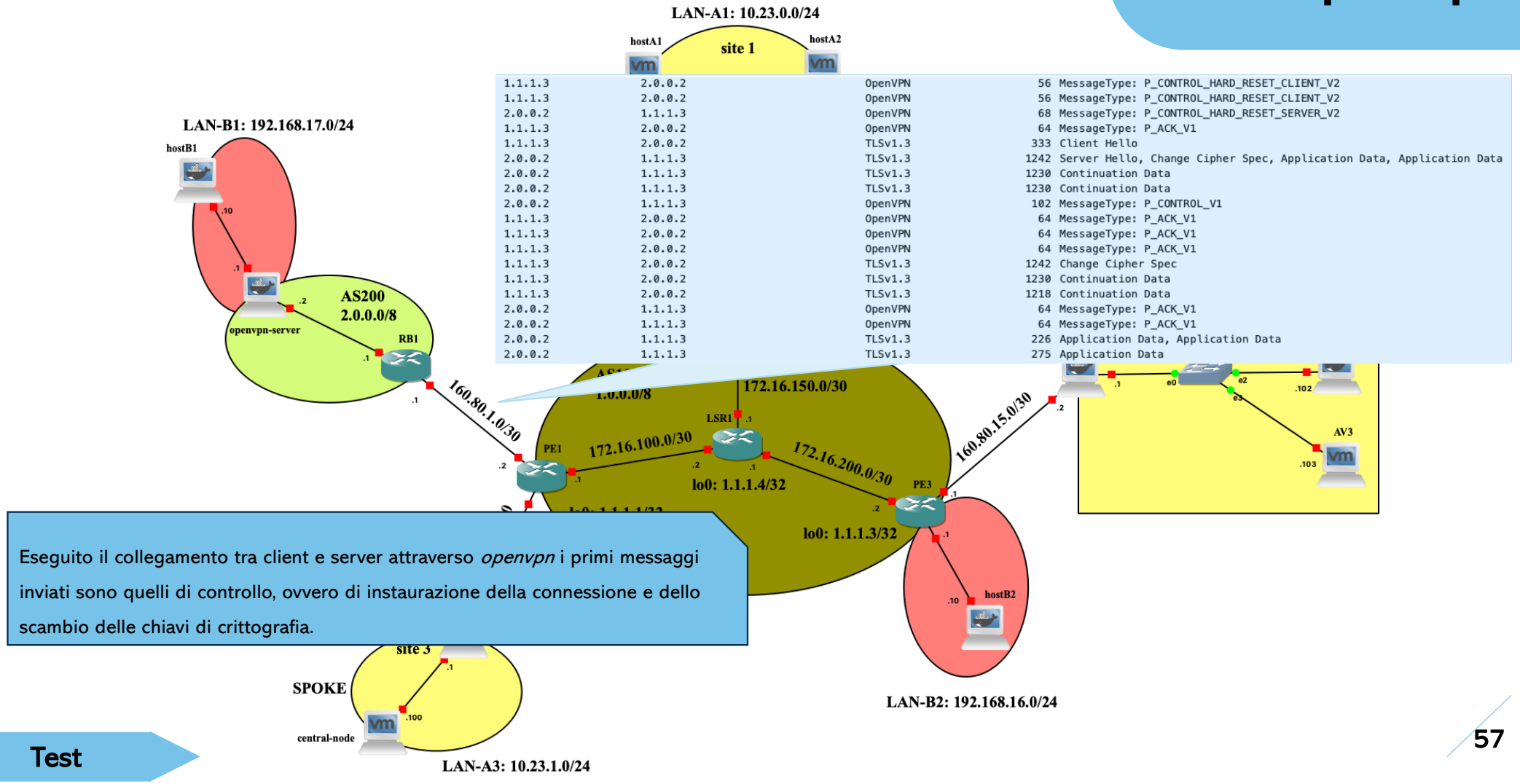
Tramite ***nmap*** è possibile effettuare uno *scan* delle porta aperte di una determinata macchina *target*. In questo caso si nota che tra tutti i pacchetti di verifica inviati da *nmap*, l'unico che riceve una risposta (ovvero che *bypassa* il firewall) è quello dedicato al servizio *HTTP*.

Test Firewall CE-A2

Provando a pingare *AV1* si verifica che solo il *central-node* riesce a ricevere una risposta. Infatti, *CE-A2* elimina tutto il traffico proveniente da indirizzi *IP* differenti da quello del *central-node*



Test OpenVpn



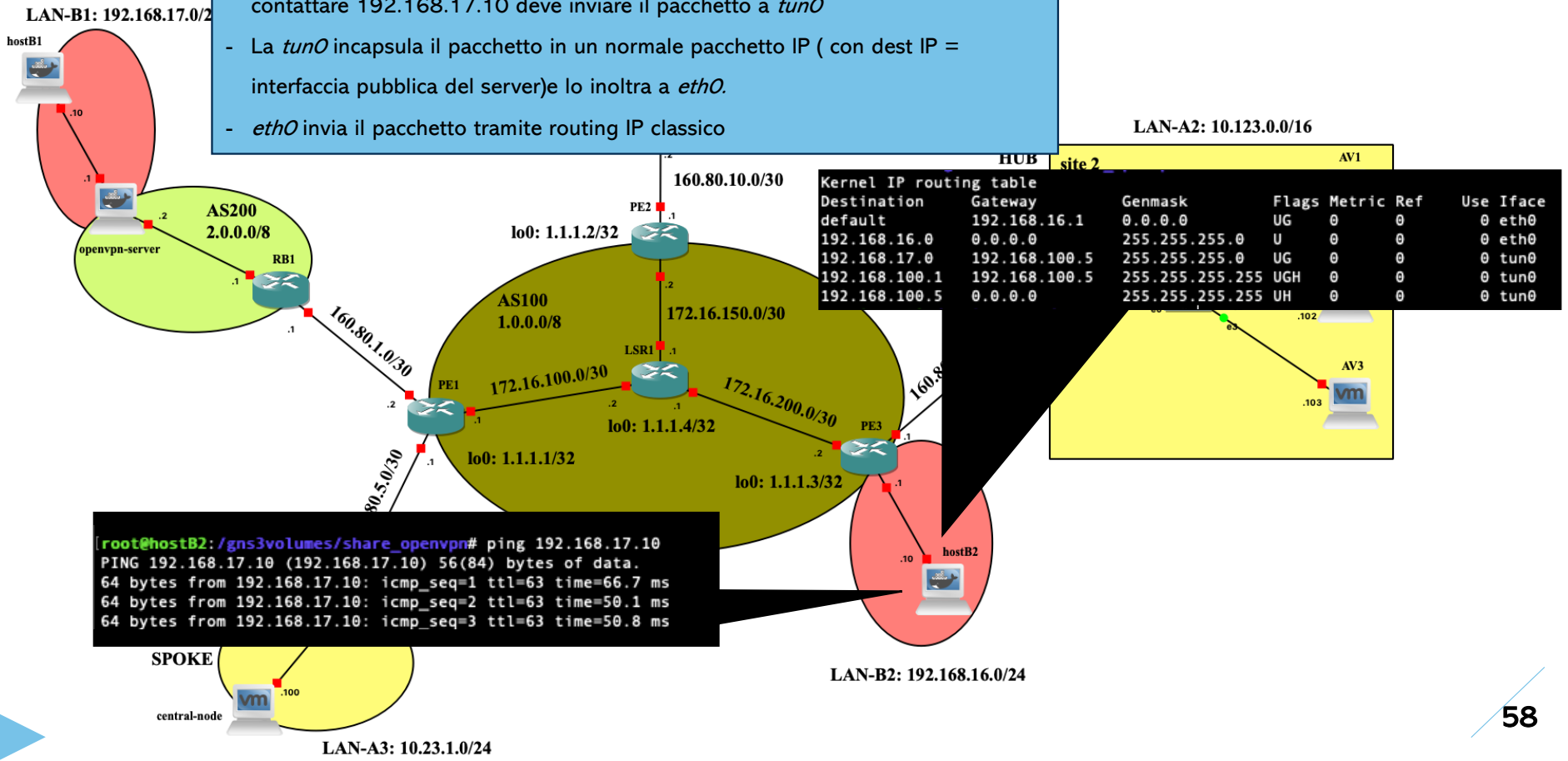
1.1.1.3	2.0.0.2	OpenVPN	56	MessageType: P_CONTROL_HARD_RESET_CLIENT_V2
1.1.1.3	2.0.0.2	OpenVPN	56	MessageType: P_CONTROL_HARD_RESET_CLIENT_V2
2.0.0.2	1.1.1.3	OpenVPN	68	MessageType: P_CONTROL_HARD_RESET_SERVER_V2
1.1.1.3	2.0.0.2	OpenVPN	64	MessageType: P_ACK_V1
1.1.1.3	2.0.0.2	TLSv1.3	333	Client Hello
2.0.0.2	1.1.1.3	TLSv1.3	1242	Server Hello, Change Cipher Spec, Application Data, Application Data
2.0.0.2	1.1.1.3	TLSv1.3	1230	Continuation Data
2.0.0.2	1.1.1.3	TLSv1.3	1230	Continuation Data
2.0.0.2	1.1.1.3	OpenVPN	102	MessageType: P_CONTROL_V1
1.1.1.3	2.0.0.2	OpenVPN	64	MessageType: P_ACK_V1
1.1.1.3	2.0.0.2	OpenVPN	64	MessageType: P_ACK_V1
1.1.1.3	2.0.0.2	OpenVPN	64	MessageType: P_ACK_V1
1.1.1.3	2.0.0.2	TLSv1.3	1242	Change Cipher Spec
1.1.1.3	2.0.0.2	TLSv1.3	1230	Continuation Data
1.1.1.3	2.0.0.2	TLSv1.3	1218	Continuation Data
2.0.0.2	1.1.1.3	OpenVPN	64	MessageType: P_ACK_V1
2.0.0.2	1.1.1.3	OpenVPN	64	MessageType: P_ACK_V1
2.0.0.2	1.1.1.3	TLSv1.3	226	Application Data, Application Data
2.0.0.2	1.1.1.3	TLSv1.3	275	Application Data

Eseguito il collegamento tra client e server attraverso *openvpn* i primi messaggi inviati sono quelli di controllo, ovvero di instaurazione della connessione e dello scambio delle chiavi di crittografia.

Test OpenVpn

Il ping verso *hostB1* ha successo e il routing della richiesta *ICMP* avviene nel seguente modo:

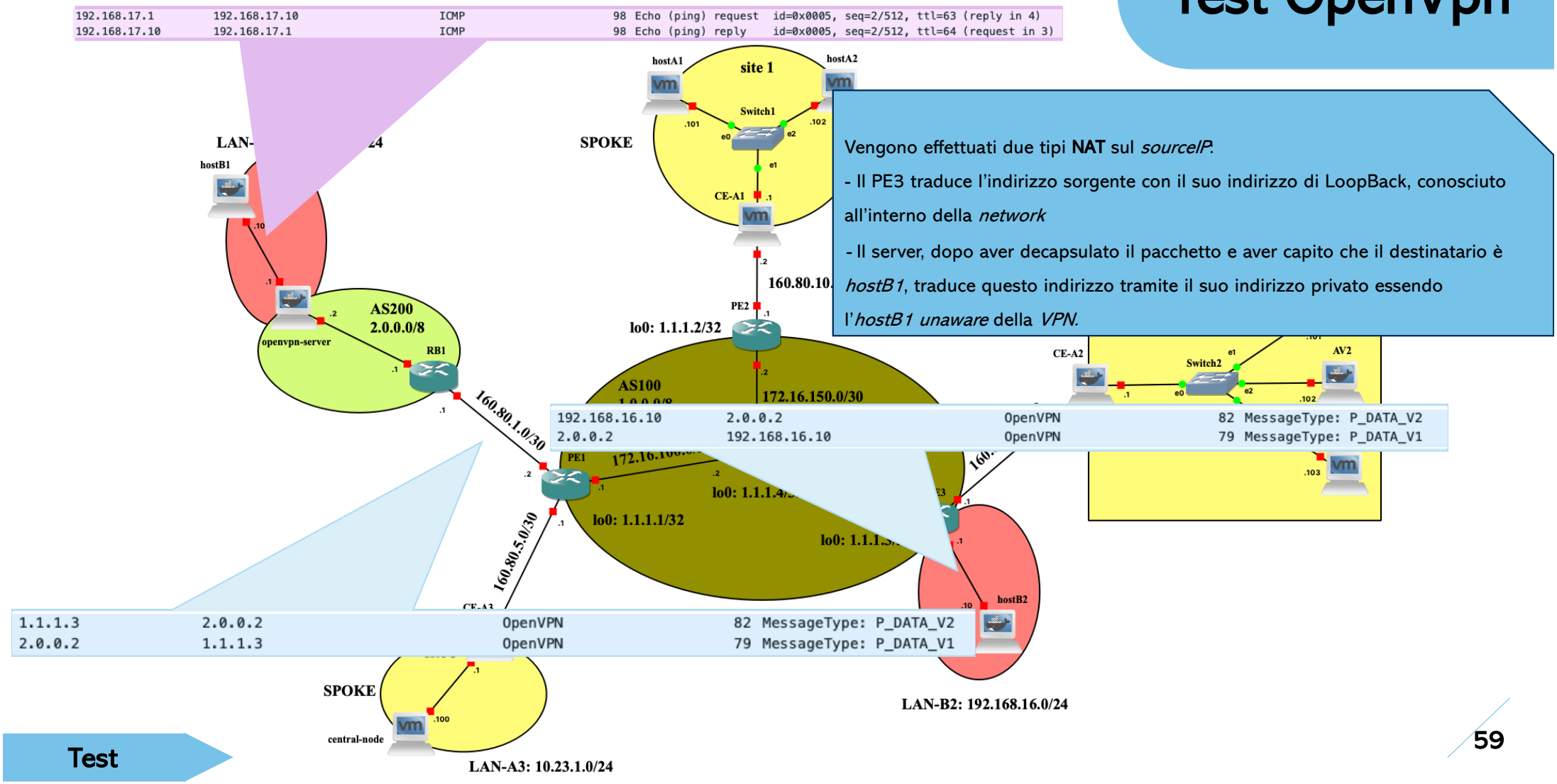
- hostB2 effettua un lookup delle tabelle di routing e scopre che per contattare 192.168.17.10 deve inviare il pacchetto a *tunO*
- La *tunO* incapsula il pacchetto in un normale pacchetto IP (con dest IP = interfaccia pubblica del server)e lo inoltra a *ethO.*
- *ethO* invia il pacchetto tramite routing IP classico




```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.16.1    0.0.0.0         UG    0      0        0 eth0
192.168.16.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.17.0    192.168.100.5   255.255.255.0   UG    0      0        0 tun0
192.168.100.1   192.168.100.5   255.255.255.255 UGH   0      0        0 tun0
192.168.100.5   0.0.0.0         255.255.255.255 UH    0      0        0 tun0
```

```
root@hostB2:/gns3volumes/share_openvpn# ping 192.168.17.10
PING 192.168.17.10 (192.168.17.10) 56(84) bytes of data.
64 bytes from 192.168.17.10: icmp_seq=1 ttl=63 time=66.7 ms
64 bytes from 192.168.17.10: icmp_seq=2 ttl=63 time=50.1 ms
64 bytes from 192.168.17.10: icmp_seq=3 ttl=63 time=50.8 ms
```

Test OpenVpn



Vengono effettuati due tipi **NAT** sul *sourceIP*.

- Il PE3 traduce l'indirizzo sorgente con il suo indirizzo di LoopBack, conosciuto all'interno della *network*
- Il server, dopo aver decapsulato il pacchetto e aver capito che il destinatario è *hostB1*, traduce questo indirizzo tramite il suo indirizzo privato essendo l'*hostB1 unaware* della *VPN*.

Conclusioni

Conclusioni

- Problemi riscontrati
  - I file di grandi dimensioni non riuscivano ad essere inviati tramite *netcat*. Venivano infatti effettuate numerose *TCP retrasmissions* dovute a congestioni della rete che hanno portato quindi alla chiusura della connessione.
    - Da prove effettuate un *MTU* da 512 bytes risulta essere ottimale nell'ottenimento di una percentuale di *packet loss* prossima allo 0.
- VMs & Containers utilizzati
  - ***Docker** : chiacchius/openvpn, chiacchius/cust_edge, chiacchius/av1_clamav, chiacchius/av2_loki*
  - ***Lubuntu**: hostA1, hostA2, CE-A1, central-node*
  - ***Linux-Lite**: av3*