{ComNetsEmu}

SIMULAZIONE VXLAN

Konstantinos Zefkilis, Luca Pio Pierno

OBIETTIVO DEL PROGETTO

- Questo progetto dimostra come il protocollo VXLAN possa essere utilizzato per estendere reti di livello 2 su infrastrutture IP, attraverso una simulazione virtuale con Mininet e Open vSwitch (OVS).

- Mininet è stato usato per creare una rete virtuale con host e switch, mentre OVS ha configurato il tunneling VXLAN, consentendo l'incapsulamento dei pacchetti Ethernet in UDP per garantire la comunicazione tra gli switch.

- Infine, Wireshark è stato impiegato per catturare e analizzare i pacchetti, permettendo di visualizzare il VXLAN Network Identifier (VNI) e confermare la corretta trasmissione del traffico ICMP tra gli host nel tunnel VXLAN.

TOPOLOGIA DELLA RETE

Due switch OVS (s1 e s2) connessi da un tunnel VXLAN per permettere la comunicazione tra reti separate. Tre host (h1, h2, h3) configurati con diversi ruoli nella rete:

Lo switch h1

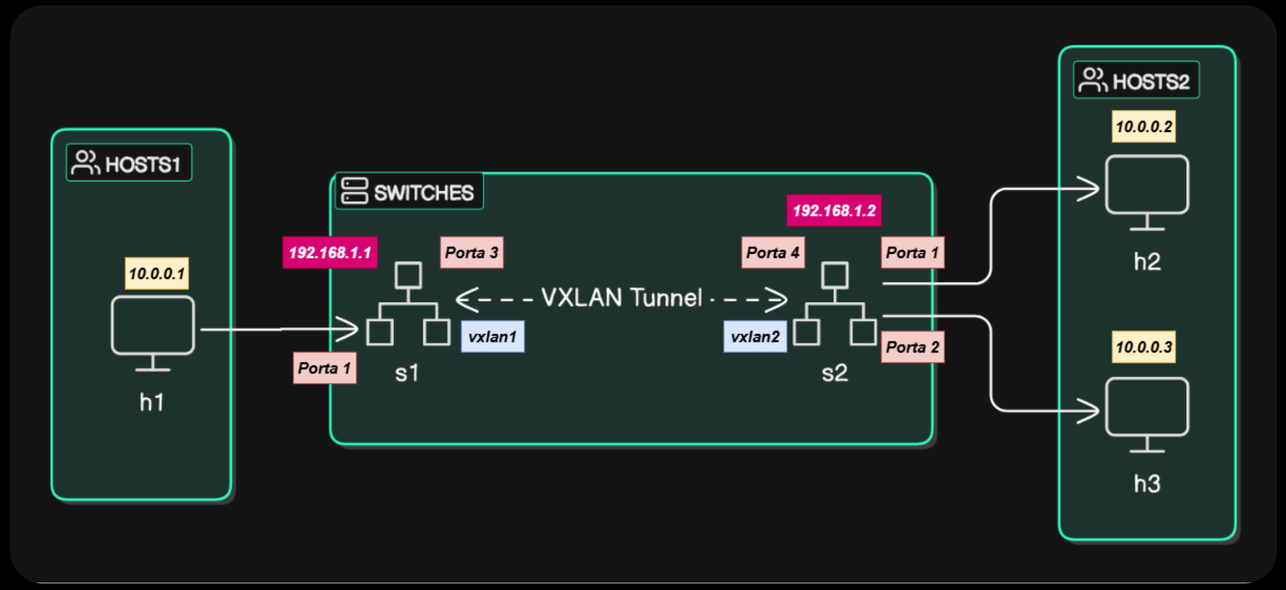
h1 è connesso direttamente a s1 e deve comunicare con gli host remoti tramite VXLAN.

Gli switch h2 e h3

h2 e h3 sono collegati allo switch s2, comunicando tra loro attraverso VLAN.

Comunicazione tra h1 e h2,h3

Il traffico tra h1 e h2/h3 viene incapsulato nel tunnel VXLAN, mentre h2 e h3 comunicano direttamente.




```python
# Creazione degli switch
s1 = self.addSwitch('s1')
s2 = self.addSwitch('s2')
```

```python
# Creazione degli host
h1 = self.addHost('h1', ip='10.0.0.1/24')
h2 = self.addHost('h2', ip='10.0.0.2/24')
h3 = self.addHost('h3', ip='10.0.0.3/24')
```

```python
# Collegamento degli host agli switch
self.addLink(h1, s1)
self.addLink(h2, s2)
self.addLink(h3, s2)
self.addLink(s1, s2)
```

IL TUNNEL VXLAN

Impostiamo gli indirizzi IP sulle interfacce dei due switch s1 e s2:

```
# Configurazione degli indirizzi IP degli switch
s1.cmd('ifconfig s1-eth2 192.168.1.1/24')
s2.cmd('ifconfig s2-eth3 192.168.1.2/24')
```

Le interfacce s1-eth2 e s2-eth3 degli switch vengono utilizzate per il tunneling VXLAN. Poiché VXLAN si basa sull'incapsulamento IP/UDP, è necessario configurare un indirizzo IP sugli switch per creare il tunnel.

Creazione e configuriamo il tunnel VXLAN tra due switch:

```
# Creazione delle interfacce VXLAN
s1.cmd('ovs-vsctl add-port s1 vxlan1 -- set interface vxlan1 type=vxlan \
       options:remote_ip=192.168.1.2 options:local_ip=192.168.1.1 options:key=1000')
s2.cmd('ovs-vsctl add-port s2 vxlan2 -- set interface vxlan2 type=vxlan \
       options:remote_ip=192.168.1.1 options:local_ip=192.168.1.2 options:key=1000')
```

- s1 crea un'interfaccia VXLAN chiamata vxlan1, destinata a 192.168.1.2 (lo switch s2).
- s2 crea un'interfaccia VXLAN chiamata vxlan2, destinata a 192.168.1.1 (lo switch s1).

Gestione del traffico tra h1 e la rete Vxlan

Definiamo le regole OpenFlow per s1 e s2, che determinano come lo switch deve gestire il traffico.

```
# Regole OpenFlow per s1
# Add flooding rules for broadcast/unknown traffic
s1.cmd('ovs-ofctl add-flow s1 "table=0,priority=1,in_port=1 actions=output:3"')
s1.cmd('ovs-ofctl add-flow s1 "table=0,priority=1,in_port=3 actions=output:1"')
```

```
# Gestione del traffico verso h1
 s2.cmd('ovs-ofctl add-flow s2 "table=0,priority=1,in_port=1,dl_dst=00:00:00:00:00:01 actions=output:4"')
 s2.cmd('ovs-ofctl add-flow s2 "table=0,priority=1,in_port=2,dl_dst=00:00:00:00:00:01 actions=output:4"')
```

- Se un pacchetto arriva da h1 (porta 1) → viene inviato nel tunnel VXLAN (porta 3).
- Se un pacchetto arriva dal tunnel VXLAN (porta 3) → viene inoltrato a h1 (porta 1).

- Se un pacchetto arriva su s2 dalla rete locale (h2 o h3) con destinazione h1, viene inoltrato nel tunnel VXLAN (porta 4).
- Solo i pacchetti diretti a h1 passano nel tunnel, evitando traffico inutile.

Comunicazione diretta tra h2 e h3, senza Vxlan

```
# Regole OpenFlow per s2
# Traffico locale tra h2 e h3 (priorità più alta)
s2.cmd('ovs-ofctl add-flow s2 "table=0,priority=2,in_port=1,dl_dst=00:00:00:00:00:03 actions=output:2"')
s2.cmd('ovs-ofctl add-flow s2 "table=0,priority=2,in_port=2,dl_dst=00:00:00:00:00:02 actions=output:1"')
```

- Se un pacchetto arriva dalla porta 1 con destinazione MAC 00:00:00:00:00:03 (h3), viene inoltrato alla porta 2.

- Se un pacchetto arriva dalla porta 2 con destinazione MAC 00:00:00:00:00:02 (h2), viene inoltrato alla porta 1.

ANALIZZIAMO I PACCHETTI

Analisi del traffico VXLAN (udp.port == 4789) con Wireshark

udp.port==4789

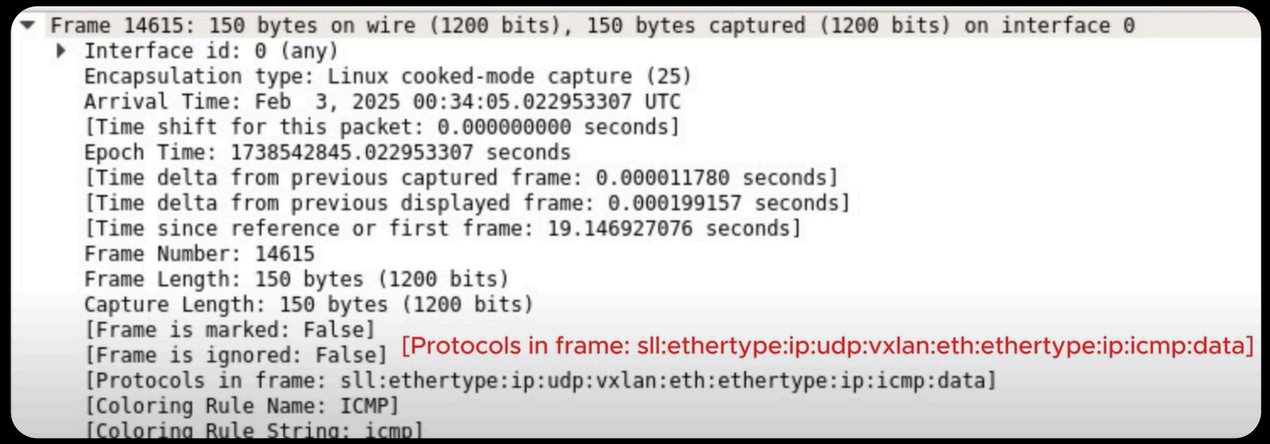
No.	Time	Source	Destination	Protocol	Length	Info
14605	19.146404336	00:00:00_00:00:01	Broadcast	ARP	94	Who has 10.0.0.2? Tell 10.0.0.1
14610	19.146727919	00:00:00_00:00:02	00:00:00_00:00:01	ARP	94	10.0.0.2 is at 00:00:00:00:00:02
14615	19.146927076	10.0.0.1	10.0.0.2	ICMP	150	Echo (ping) request id=0x2592, seq=1/256, ttl=64 (reply in 14620)
14620	19.147176218	10.0.0.2	10.0.0.1	ICMP	150	Echo (ping) reply id=0x2592, seq=1/256, ttl=64 (request in 14615)
14852	20.148283130	10.0.0.1	10.0.0.2	ICMP	150	Echo (ping) request id=0x2592, seq=2/512, ttl=64 (reply in 14857)
14857	20.148322946	10.0.0.2	10.0.0.1	ICMP	150	Echo (ping) reply id=0x2592, seq=2/512, ttl=64 (request in 14852)
16348	21.163893004	10.0.0.1	10.0.0.2	ICMP	150	Echo (ping) request id=0x2592, seq=3/768, ttl=64 (reply in 16353)
16353	21.163919311	10.0.0.2	10.0.0.1	ICMP	150	Echo (ping) reply id=0x2592, seq=3/768, ttl=64 (request in 16348)
17469	22.187956783	10.0.0.1	10.0.0.2	ICMP	150	Echo (ping) request id=0x2592, seq=4/1024, ttl=64 (reply in 17474)
17474	22.188012458	10.0.0.2	10.0.0.1	ICMP	150	Echo (ping) reply id=0x2592, seq=4/1024, ttl=64 (request in 17469)
18837	23.218313065	10.0.0.1	10.0.0.2	ICMP	150	Echo (ping) request id=0x2592, seq=5/1280, ttl=64 (reply in 18842)
18842	23.218363415	10.0.0.2	10.0.0.1	ICMP	150	Echo (ping) reply id=0x2592, seq=5/1280, ttl=64 (request in 18837)
19388	24.236189116	10.0.0.1	10.0.0.2	ICMP	150	Echo (ping) request id=0x2592, seq=6/1536, ttl=64 (reply in 19393)
19393	24.236238511	10.0.0.2	10.0.0.1	ICMP	150	Echo (ping) reply id=0x2592, seq=6/1536, ttl=64 (request in 19388)
19406	24.301022039	00:00:00_00:00:02	00:00:00_00:00:01	ARP	94	Who has 10.0.0.1? Tell 10.0.0.2
19411	24.301054150	00:00:00_00:00:01	00:00:00_00:00:02	ARP	94	10.0.0.1 is at 00:00:00:00:00:01
23631	29.164142461	fe80::200:ff:fe00:1	ff02::2	ICMPv6	122	Router Solicitation from 00:00:00:00:00:01

- La cattura mostra i pacchetti VXLAN filtrati su Wireshark tramite il comando udp.port==4789. Qui possiamo osservare il traffico incapsulato tra gli switch OVS che trasportano i dati tra gli host virtuali.

```
▼ Virtual eXtensible Local Area Network
  ▶ Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 1000
    Reserved: 0
```

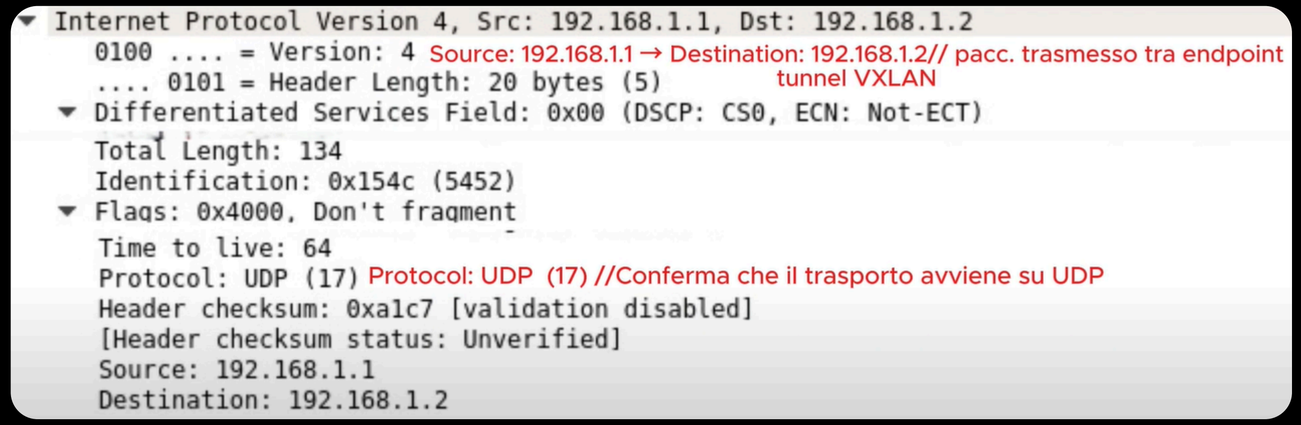
- L'header VXLAN mostra il campo VNI (VXLAN Network Identifier), che in questo caso è impostato a 1000.

Struttura del pacchetto incapsulato (IP + UDP + VXLAN + Ethernet + ICMP):



- Il pacchetto ICMP originale (ping) viene racchiuso in un frame Ethernet, quindi incapsulato in VXLAN, trasportato tramite UDP e infine inoltrato su un pacchetto IP esterno.

Pacchetto IP esterno: Trasporto del traffico VXLAN tra gli endpoint del tunnel



- Questo pacchetto IP mostra il traffico incapsulato nel tunnel VXLAN. Gli indirizzi IP sorgente e destinazione appartengono agli switch OVS, confermando che il pacchetto sta attraversando il tunnel.