

Hálózatok építése és üzemeltetése

OpenFlow / POX gyakorlat

1. feladat:

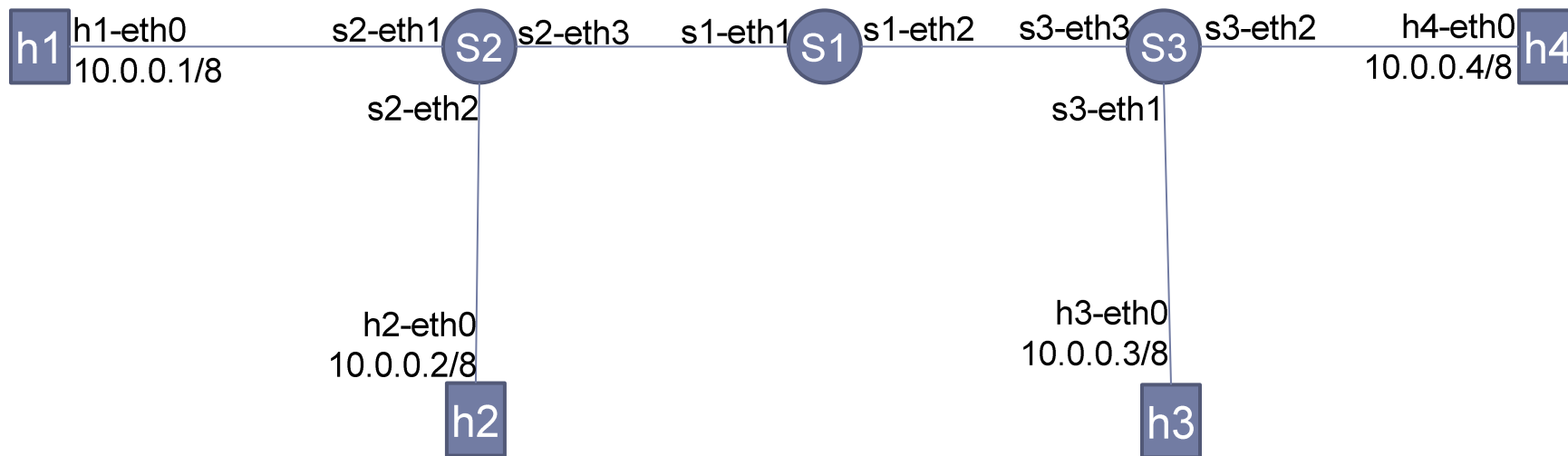
térképezzük fel az emulált topológiát

- ▶ `$ sudo -E mn --topo tree,depth=2 --controller=remote`
- ▶ **Használható parancsok:**
 - ▶ `mininet> net`
 - ▶ `mininet> dump`
 - ▶ `mininet> h1 ifconfig`
 - ▶ `mininet> ...`

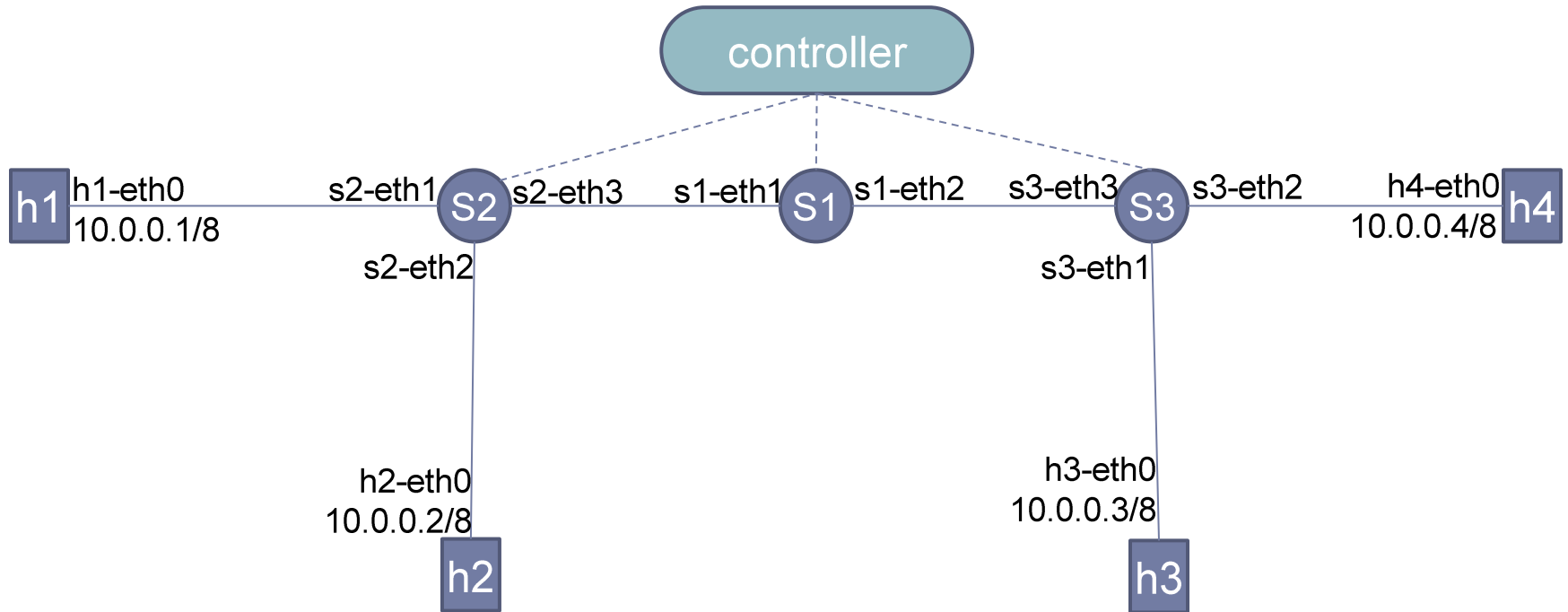
1. feladat:

térképezzük fel az emulált topológiát

- ▶ `$ sudo -E mn --topo tree,depth=2 --controller=remote`



1. feladat: térképezzük fel az emulált topológiát



2. feladat: forgalom megfigyelése

- ▶ h1: `~# ping -c 1 10.0.0.2`
- ▶ Figyeljük meg az s?-eth? interfészt. Meddig jutnak el az ARP kérések?

2. feladat: forgalom megfigyelése

- ▶ h1: `~# ping -c 1 10.0.0.2`
- ▶ Figyeljük meg az s?-eth? interfészt. Meddig jutnak el az ARP kérések?
 - ▶ ALT-F1: `~$ sudo wireshark &`
 - ▶ Mindössze három ARP csomag lesz a h1-s2 linken.
 - ▶ Hallgassuk bele a lo interfészbe is!
 - ▶ .

2. feladat: forgalom megfigyelése

- ▶ h1: `~# ping -c 1 10.0.0.2`
- ▶ Figyeljük meg az s?-eth? interfészt. Meddig jutnak el az ARP kérések?
 - ▶ ALT-F1: `~$ sudo wireshark &`
 - ▶ Mindössze három ARP csomag lesz a h1-s2 linken.
- ▶ Hallgassuk bele a lo interfészbe is!
 - ▶ A switchek próbálnak kapcsolódni a kontrollerhez.

POX ~~installálása~~, frissítése

```
git clone http://github.com/noxrepo/pox
```

```
cd ~/pox
```

```
git pull
```

```
git checkout eel
```


Egyszerű kontrolleralkalmazás

- ▶ `~/pox$./pox.py forwarding.hub`
- ▶ Most sikeres lesz az előző ping parancs, de hol lesz most adatforgalom?
 - ▶ (érdemes külön ablakokban tcpdumpot futtatni.)

Mi történik a kontrollerrel?

- ▶ Vizsgáljuk meg a **lo** forgalmát a kontroller indításakor?
 - ▶ Értelmezzük az üzeneteket!
- ▶ Hogy nézhetnek ki a folyamatáblák a switch-ekben?

Mi történik a kontrollerrel?

- ▶ Hogy nézhetnek ki a folyamtáblák a switch-ekben?
- ▶ Nézzük meg a folyamtáblákat!
 - ▶ `$ sudo ovs-ofctl show s2`
 - ▶ `$ sudo ovs-ofctl dump-flows s2`
 - ▶ `mininet> dpctl dump-flows`
 - ▶ `$ dpctl dump-flows tcp:localhost:6634 (vagy 35, 36)`

Egyszerű kontrolleralkalmazás 2.

- ▶ `~/pox$./pox.py log.level --DEBUG forwarding.hub --reactive`
- ▶ Most sikeres is lesz az előző ping parancs, de hogyan alakul most a **kontroll**forgalom?
- ▶ Mi lesz a folyamatáblákban? Miért?

Egyszerű kontrolleralkalmazás 2.

- ▶ `~/pox$./pox.py log.level --DEBUG forwarding.hub --reactive`
- ▶ Most sikeres is lesz az előző ping parancs, de hogyan alakul most a **kontroll**forgalom?
- ▶ Mi lesz a folyamatáblákban? Miért?
 - ▶ Semmi, mert a switchek minden csomagot felküldenek a kontrollernek, ami nem installál folyamatábla-bejegyzéseket, hanem minden egyes csomagra megmondja, hogy merre kell továbbítania a switchnek.

Érdekesebb folyamatábla bejegyzések

```
$ ./pox.py log.color log.level --DEBUG forwarding.l2_learning
```

```
h2:~$ nc -l 2222
```

```
h1:~$ date|nc 10.0.0.2 2222
```

```
mininet> dpctl dump-flows
```

Érdekesebb folyamatábla bejegyzések

```
$ ./pox.py log.color log.level --DEBUG forwarding.l2_learning
```

```
h2:~$ nc -p 2222
```

```
h1:~$ date|nc 10.0.0.2 2222
```

Érdekesebb statisztikai mezők

```
cookie=0x0, duration=2.27s, table=0, n_packets=5, n_bytes=367,  
idle_timeout=10, hard_timeout=30, idle_age=2, priority=65535,  
tcp,in_port=1,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,  
nw_src=10.0.0.1,nw_dst=10.0.0.2,nw_tos=0,tp_src=46359,tp_dst=2222  
actions=output:2
```

```
cookie=0x0, duration=2.267s, table=0, n_packets=3, n_bytes=206,  
idle_timeout=10, hard_timeout=30, idle_age=2, priority=65535,  
tcp,in_port=2,vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,  
nw_src=10.0.0.2,nw_dst=10.0.0.1,nw_tos=0,tp_src=2222,tp_dst=46359  
actions=output:1
```

Érdekesebb folyamatábla bejegyzések

```
$ ./pox.py log.color log.level --DEBUG forwarding.l2_learning
```

```
h2:~$ nc -p 2222
```

```
h1:~$ date|nc 10.0.0.2 2222
```

Érdekesebb illeszkedési mezők

```
cookie=0x0, duration=2.27s, table=0, n_packets=5, n_bytes=367,  
idle_timeout=10, hard_timeout=30, idle_age=2, priority=65535,  
tcp,in_port=1,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,  
nw_src=10.0.0.1,nw_dst=10.0.0.2,nw_tos=0,tp_src=46359,tp_dst=2222  
actions=output:2
```

```
cookie=0x0, duration=2.267s, table=0, n_packets=3, n_bytes=206,  
idle_timeout=10, hard_timeout=30, idle_age=2, priority=65535,  
tcp,in_port=2,vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,  
nw_src=10.0.0.2,nw_dst=10.0.0.1,nw_tos=0,tp_src=2222,tp_dst=46359  
actions=output:1
```


Érdekesebb folyamatábla bejegyzések

```
$ ./pox.py log.color log.level --DEBUG forwarding.l2_learning
```

```
h2:~$ nc -p 2222
```

```
h1:~$ date|nc 10.0.0.2 2222
```

Egyszerű akciólista

```
cookie=0x0, duration=2.27s, table=0, n_packets=5, n_bytes=367,  
idle_timeout=10, hard_timeout=30, idle_age=2, priority=65535,  
tcp,in_port=1,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,  
nw_src=10.0.0.1,nw_dst=10.0.0.2,nw_tos=0,tp_src=46359,tp_dst=2222  
actions=output:2
```

```
cookie=0x0, duration=2.267s, table=0, n_packets=3, n_bytes=206,  
idle_timeout=10, hard_timeout=30, idle_age=2, priority=65535,  
tcp,in_port=2,vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,  
nw_src=10.0.0.2,nw_dst=10.0.0.1,nw_tos=0,tp_src=2222,tp_dst=46359  
actions=output:1
```

-
- ▶ `sudo -E mn --mac --topo linear,k=5,n=1 --controller=remote --link=tc,delay=10ms`
 - ▶ `pox.py log.color log.level --DEBUG forwarding.l2_learning`
 - ▶ `mininet> h1 ping h5`

- ▶ sudo -E **mn** --mac --topo linear,k=5,n=1 --controller=remote --link=tc,delay=10ms
- ▶ **pox.py** log.color log.level --DEBUG forwarding.I2_learning

▶ mininet> h1 ping h5

PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.

64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=**459** ms

64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=**167** ms

64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=124 ms

64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=121 ms

^C

▶ Mitől nagyobb az első válaszidő?

▶ .

-
- ▶ `sudo -E mn --mac --topo linear,k=5,n=1 --controller=remote --link=tc,delay=10ms`
 - ▶ `pox.py log.color log.level --DEBUG forwarding.l2_learning`

▶ `mininet> h1 ping h5`

PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.

64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=**459** ms

64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=**167** ms

64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=124 ms

64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=121 ms

^C

- ▶ Mitől nagyobb az első válaszidő?
 - ▶ ARP + ping + kontrollerkommunikáció

```
pox.py log.color log.level --DEBUG forwarding.l2_learning proto.arp_responder --10.0.0.5=00:00:00:00:00:05 --  
10.0.0.1=00:00:00:00:00:01 py
```

▶ mininet> h1 ping h5

PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.

64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=232 ms

64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=186 ms

64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=121 ms

^C

▶ POX> arp

POX módosítása

- ▶ Soronként megadni az ARP táblát macerás.
Egyszerűbb is lehet az üzemeltetés, ha ismerjük a topológiát vagy algoritmikusan kikövetkeztethető a tábla.
- ▶ Jelenleg: $10.0.0.X \rightarrow 00:00:00:00:00:00:X$
- ▶ Írjuk át a arp_responder modult, hogy automatikusan válaszoljon a 10.0.0.0/24-es kérésekre!

POX, ethernet címek

- ▶ Írjuk át a `arp_responder` modult, hogy automatikusan válaszoljon a `10.0.0.0/24`-es kérésekre!
- ▶ `from pox.lib.addresses import EthAddr`
- ▶ **String → EthAddr:**
`mac = EthAddr("aa:bb:cc:dd:ee:ff")`
- ▶ **EthAddr → string:**
`s = str(mac)`

POX módosítása

- ▶ Írjuk át a arp_responder modult, hogy automatikusan válaszoljon a 10.0.0.0/24-es kérésekre!
- ▶ Jó lesz erre is?
 - ▶ sudo -E mn --mac --topo linear, **k=10**, n=1 --controller=remote --link=tc, delay=10ms

```
mininet@mininet-vm: ~/pox/pox/proto
mininet@mininet-vm:~/pox/pox/proto$ git diff
diff --git a/pox/proto/arp_responder.py b/pox/proto/arp_responder.py
index e8aa873..7787d41 100644
--- a/pox/proto/arp_responder.py
+++ b/pox/proto/arp_responder.py
@@ -30,6 +30,7 @@ import pox
 log = core.getLogger()

 from pox.lib.packet.ethernet import ethernet, ETHER_BROADCAST
+from pox.lib.addresses import EthAddr
 from pox.lib.packet.arp import arp
 from pox.lib.packet.vlan import vlan
 from pox.lib.addresses import IPAddr, EthAddr
@@ -205,7 +206,9 @@ class ARPResponder (object):
     if a.opcode == arp.REQUEST:
         # Maybe we can answer

-         if a.protodst in _arp_table:
+         prefix = '10.0.0.'
+         if a.protodst in _arp_table or \
+             str(a.protodst).startswith(prefix):
             # We have an answer...

             r = arp()
@@ -217,7 +220,12 @@ class ARPResponder (object):
     r.hwdst = a.hwsrc
     r.protodst = a.protosrc
     r.protosrc = a.protodst
-    mac = _arp_table[a.protodst].mac
+    if str(a.protodst).startswith(prefix):
+        end = str(a.protodst)[len(prefix):]
+        mac = EthAddr('00:00:00:00:00:%s' % end)
+        log.info('haha: %s' % end)
+    else:
+        mac = _arp_table[a.protodst].mac
     if mac is True:
         # Special case -- use ourself
         mac = event.connection.eth_addr
mininet@mininet-vm:~/pox/pox/proto$
```


forwarding.topo_proactive

- ▶ A debuggolást megkönnyíti a mininet --mac argumentuma, de a valóságban az IP-mac cím hozzárendelés előre nem ismert.
- ▶ De ha az IP címeket a DHCP szerver szisztematikusan osztja ki, akkor azt figyelembe lehet venni a routingnál.
- ▶ címkiosztási séma: **10.switchID.portNumber.x**
- ▶ Szorgalmi feladat: az implementáció megértése