ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΥΠΗΡΕΣΙΕΣ ΔΙΑΔΙΚΤΥΟΥ

Εισαγωγή στο **Mininet** *Network Emulator*

Άννα Κεφάλα

Ακολουθούν οδηγίες πρακτικής εξάσκησης με το Mininet σύμφωνα με την παρουσίαση του εργαλείου στα πλαίσια του μαθήματος.

Εκκίνηση του Mininet με την ελάχιστη δικτυακή τοπολογία

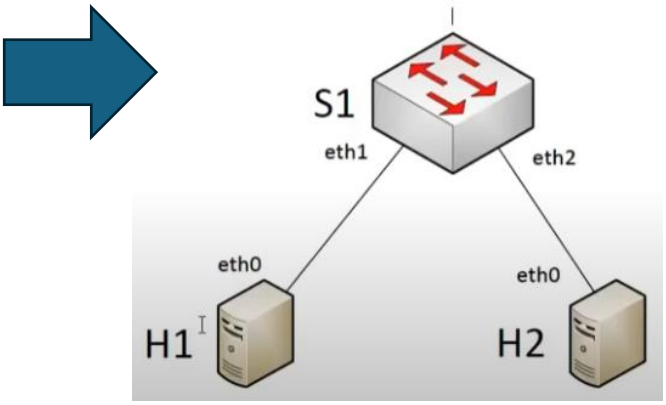
Ανοίγουμε ένα τερματικό και ξεκινάμε το Mininet:

```
$ sudo mn
```

Τοπολογία: 2 hosts, 1 openflow kernel switch, 1 basic openflow reference controller.

Χρησιμοποιώντας το Python API μπορούμε να υλοποιήσουμε διαφορετικές τοπολογίες.

```
csuser@csuser-virtualbox:/$ sudo mn
[sudo] password for csuser:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```




```
mininet> help
```
(μπορούμε να δούμε τις εντολές που παρέχονται στο CLI)

```
mininet> help

Documented commands (type help <topic>):
========================================
EOF    gterm  iperfudp  nodes        pingpair      py      switch  xterm
dpctl  help   link      noecho       pingpairfull  quit    time
dump   intfs  links     pingall      ports         sh      wait
exit   iperf  net       pingallfull  px            source  x

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2
```

`mininet> nodes` *(εμφανίζει τα nodes της τοπολογίας)*

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
```

`mininet> dump` *(εμφανίζει περισσότερες λεπτομέρειες για τους hosts, IP, pid, πλήρες όνομα)*

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=2214>
<Host h2: h2-eth0:10.0.0.2 pid=2216>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=2221>
<OVSController c0: 127.0.0.1:6653 pid=2207>
```

`mininet> net` *(εμφανίζει τις δικτυακές συνδέσεις μεταξύ των hosts: virtual ethernet ζεύγη μεταξύ hosts και switch links)*

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
```

Μπορούμε να δώσουμε εντολές στους **hosts** μέσω του Mininet CLI. Ελέγχουμε την επικοινωνία μεταξύ των hosts στέλνοντας ICMP πακέτα (ping):

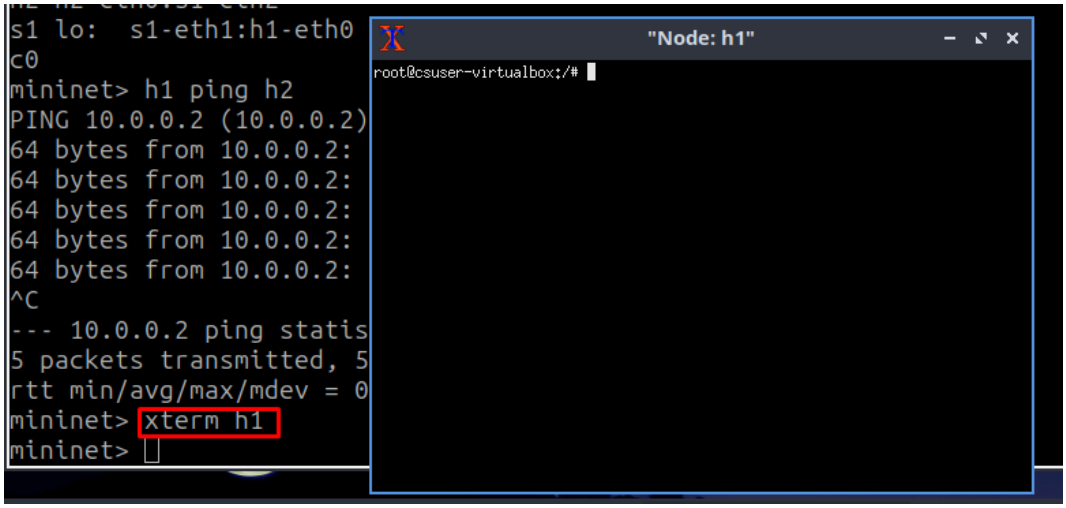
`mininet> h1 ping h2`

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=3.65 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.310 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.058 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.067 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.056 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4052ms
rtt min/avg/max/mdev = 0.056/0.829/3.654/1.415 ms
```

μεγαλύτερο RTT για το **1ο πακέτο**, γιατί θα πρέπει να ρωτήσει τον controller τί να κάνει με αυτό το flow. Για τα επόμενα πακέτα, έχει κρατήσει στην cache του το flow (rule), οπότε γίνεται κατευθείαν το ping στον h2.

`mininet> xterm h1` *(ανοίγει extra terminal στον h1)*

`h1> ping 10.0.0.2` *(ping στον h2, δεν τον γνωρίζει ως h2, → πρέπει να χρησιμοποιήσω την IP)*



`mininet> pingall` *(ping all host pairs)*

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

`mininet> iperf` *(έλεγχος -απόδοσης- tcp bandwidth μεταξύ των hosts)*

`mininet> iperfudp` *(έλεγχος -απόδοσης- udp bandwidth μεταξύ των hosts)*

```
mininet> iperfudp
*** Iperf: testing UDP bandwidth between h1 and h2
*** Results: ['10M', '10.5 Mbits/sec', '10.5 Mbits/sec']
```

`mininet> exit` *(stops topology και exits mininet)*

```
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 1 terms
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 2565.699 seconds
```

$ sudo mn -c *(cleanup after exiting mininet)*

Εκκίνηση του mininet εισάγοντας καθυστέρηση (latency) στο δίκτυο

$ sudo mn --link tc,bw=10,delay=10ms *(bandwidth: 10Mbps, delay: 10msec)*

```
csuser@csuser-virtualbox:/$ sudo mn --link tc,bw=10,delay=10ms
[sudo] password for csuser:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (h1, s1) (10.00Mbit 10ms delay) (10.00Mbit 10ms dela
y) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(10.00Mbit 10ms delay) (10.00Mbit 10ms delay)
*** Starting CLI:
mininet>
```

Αν τρέξουμε πάλι το **iperfudp** θα φανεί το latency του δικτύου:

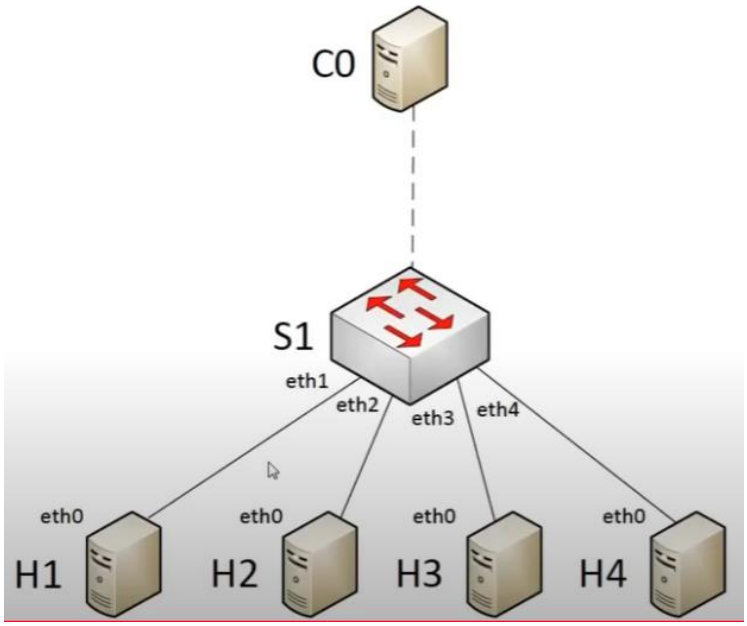
```
mininet> iperfudp
*** Iperf: testing UDP bandwidth between h1 and h2
*** Results: ['10M', '9.67 Mbits/sec', '9.67 Mbits/sec']
```

Είναι διαθέσιμες και **άλλες τοπολογίες** μέσω του Mininet CLI:

$ sudo mn --topo=single,4

Τοπολογία δικτύου: 1 switch, 4 hosts

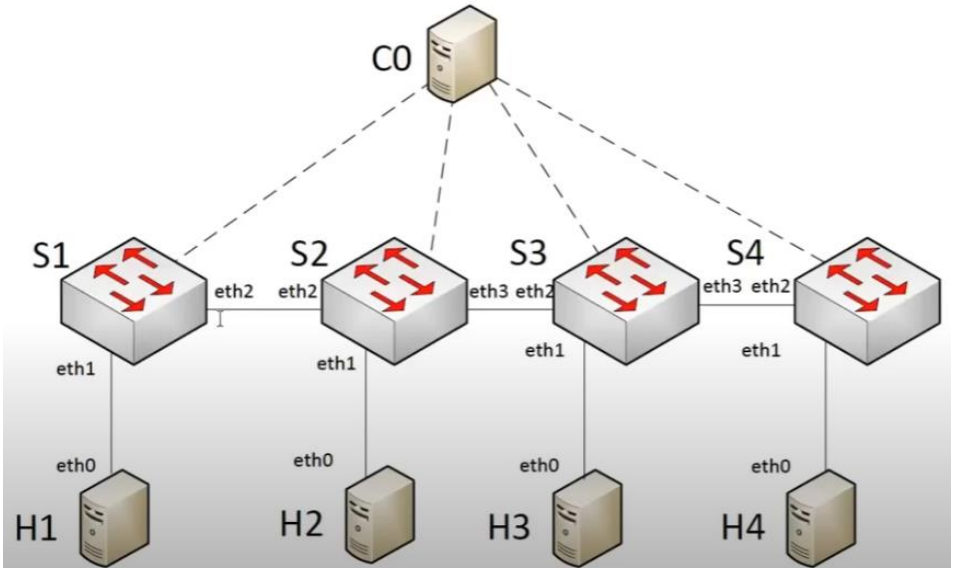
```
csuser@csuser-virtualbox:/$ sudo mn --topo=single,4
[sudo] password for csuser:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```



$ sudo mn --topo=linear,4

Τοπολογία δικτύου: 4 switches, 4 hosts, ζεύγη host/switch και linear connections μεταξύ switches

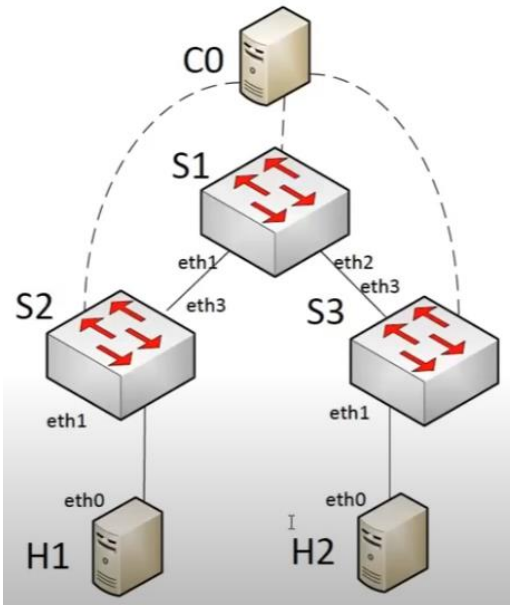
```
csuser@csuser-virtualbox:/$ sudo mn --topo=linear,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s2, s1) (s3, s2) (s4, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet>
```



$ sudo mn --topo=tree,2,2

Τοπολογία δικτύου: tree topology, depth:2 and fanout:2, 2 επίπεδα switches, 2 switches στο 2ο επίπεδο

```
csuser@csuser-virtualbox:/$ sudo mn --topo=tree,2,2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```



Εντελώς customized τοπολογίες με χρήση του Python API. Δύο τρόποι εκτέλεσης της τοπολογίας δικτύου στο Mininet.

1. Δημιουργία μίας custom τοπολογίας σε python script και «εκτέλεση» της τοπολογίας μέσα στο Mininet:

   ```
   $ sudo mn --custom myscript.py --topo mytopo
   ```

2. Δημιουργία μίας custom τοπολογίας σε python script και «εκτέλεση» κατευθείαν του python script χωρίς την εντολή “mn”:

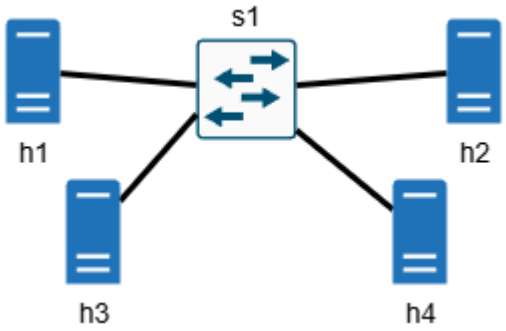
   ```
   $ python3 myscript.py
   ```

Δεν χρειάζονται ιδιαίτερες γνώσεις python, αλλά τροποποιήσεις σε python scripts.

Χρήση του OpenFlow: μπορούμε να χρησιμοποιήσουμε το OpenFlow για να προγραμματίσουμε τα switches και να δημιουργήσουμε εξειδικευμένες λειτουργίες δικτύωσης.

Παράδειγμα 1

Δίνεται ο σκελετός ενός python script μιας απλής τοπολογίας (part1-skeleton.py), θέλουμε να το τροποποιήσουμε ώστε να υλοποιήσουμε την ακόλουθη τοπολογία:



Έστω part1.py το python script που υλοποιεί την παραπάνω τοπολογία.

$ sudo mn --custom part1.py --topo part1

```
csuser@csuser-virtualbox:/media/sf_VMsharedFiles$ sudo mn --custom part1.py --topo part1
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

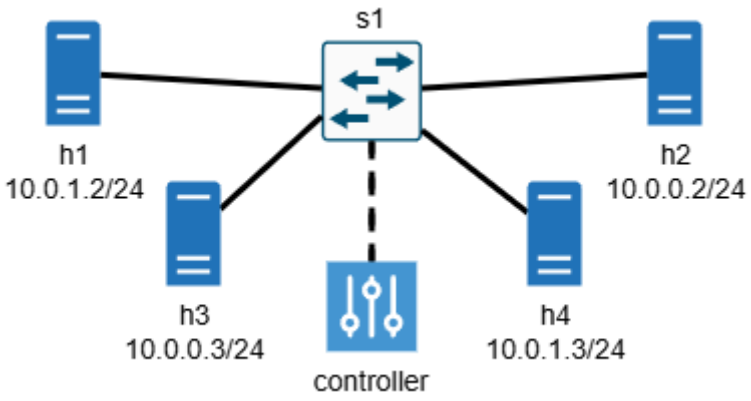
Εναλλακτικός τρόπος εκτέλεσης της τοπολογίας:

$ sudo python3 part1.py

Παράδειγμα 2

Θα προσθέσουμε στην τοπολογία του part1 ένα **POX** (python) **controller**, ο οποίος θα στέλνει μέσω OpenFlow εντολές στο switch, υλοποιώντας ένα απλό **Firewall**.

Ο remote controller θα δέχεται μηνύματα στη default IP loopback address 127.0.0.1, στο port 6653. Θα υλοποιήσουμε την ακόλουθη τοπολογία:



Σημείωση: h1 και h4 στο ίδιο subnet, h2 και h3 στο ίδιο subnet.

part2.py: προσδιορίζουμε την τοπολογία που θέλουμε να υλοποιηθεί

part2controller.py: POX controller στον οποίο θα υλοποιηθεί η λειτουργία του firewall

Οι **κανόνες** που θα πρέπει να υλοποιούνται στο firewall είναι οι εξής:

src ip	dst ip	protocol	action
any ipv4	any ipv4	icmp	accept
any	any	arp	accept
any ipv4	any ipv4	-	drop

Δηλαδή το firewall θα πρέπει να επιτρέπει όλη την κίνηση ARP και ICMP να περνάει. Οποιαδήποτε άλλη κίνηση θα γίνεται drop. Τα flow tables κάνουν match το flow/rule με την μεγαλύτερη προτεραιότητα πρώτα. Θα πρέπει όταν δημιουργείται ένα flow στον controller να γίνεται και "install" στο switch έτσι ώστε να «θυμάται» τον κανόνα και να μην ρωτάει κάθε φορά τον controller.

Για να τρέξει σωστά ο **POX controller** πρέπει όταν ξεκινάει να προσδιορίζεται το port στο οποίο δέχεται μηνύματα. Το POX ψάχνει τα source files στο ~/pox/pox, βάζουμε τα python scripts στο ~/pox/pox/misc (μέσα στα αρχεία του csuser). Το misc θα πρέπει να συμπεριλαμβάνεται στο όνομα του python script.

```
$ sudo ~/pox/pox.py openflow.of_01 --port=6653 misc.part2controller
```
(χωρίς .py)

```
$ sudo python3 part2.py
```
(σε διαφορετικό terminal)

Ο POX controller θα τρέξει σε διαφορετικό -remote- terminal.

```
mininet> dump
```

```
mininet> h1 ping h2
```
(διαφορετικά subnets)

```
mininet> h1 ping h2
ping: connect: Network is unreachable
```

```
mininet> h1 ping h4
```
(same subnet)

```
mininet> h1 ping h4
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.
64 bytes from 10.0.1.3: icmp_seq=1 ttl=64 time=0.387 ms
64 bytes from 10.0.1.3: icmp_seq=2 ttl=64 time=0.077 ms
64 bytes from 10.0.1.3: icmp_seq=3 ttl=64 time=0.073 ms
^C
--- 10.0.1.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2068ms
rtt min/avg/max/mdev = 0.073/0.179/0.387/0.147 ms
```