

Περιεχόμενα

Ζητούμενο 1	4
Εργαστήριο 1	4
Υποερώτημα Α Υπηρεσία Υπολογιστικής Νέφους Okeanos.....	4
Υποερώτημα Β Απομακρυσμένη πρόσβαση σε μηχανές με το πρωτόκολλο SSH.....	4
Υποερώτημα Γ Εικονικές μηχανές με προεγκατεστημένο λογισμικό εξομοίωσης δικτύων SDN.	4
Υποερώτημα Δ Λογισμικό προσομοίωσης: Mininet.....	4
Εργαστήριο 2	5
Υποερώτημα Α Σύνδεση στη μηχανή που φιλοξενεί το λογισμικό προσομοίωσης.....	5
Υποερώτημα Β Λογισμικό προσομοίωσης mininet.....	5
Υποερώτημα Γ Δημιουργία δικτυώματος με έναν μεταγωγέα σε τοπολογία αστέρα.....	5
Υποερώτημα Δ Λειτουργία εικονικού μεταγωγέα OpenVswitch (OVS).....	6
Εργαστήριο 3	9
Υποερώτημα Α Γενικά περί Ροών.....	9
Υποερώτημα Β Ροές με στοιχεία 1ου επιπέδου (OSI physical layer).....	11
Υποερώτημα Γ Ροές με στοιχεία 2ου επιπέδου (OSI data-link layer).....	12
Υποερώτημα Δ Ροές με στοιχεία 3ου επιπέδου (OSI network layer).....	13
Υποερώτημα Ε Ροές με στοιχεία 4ου επιπέδου (OSI transport layer).....	15
Εργαστήριο 4	19
Υποερώτημα Α Συμμετοχή του Ελεγκτή OpenDaylight (ODL) στο δίκτυωμα.....	19
Ζητούμενο 2 Απάντηση.....	24
Ζητούμενο 3 Απάντηση.....	25

Ζητούμενο 1

Εργαστήριο 1

A. Υπηρεσία Υπολογιστικής Νέφους ~Οκεανος

Δημιουργήσαμε την μηχανή στην υπηρεσία Οκεανος. Του δώσαμε μια public IPv4.



B. Απομακρυσμένη πρόσβαση σε μηχανές με το πρωτόκολλο SSH

Για να συνδεθούμε στο VM του Οκεανος αρχικά ανοίγουμε το Windows Terminal σε WSL mode. Δεύτερον με την εντολή **sudo openvpn UNIWA-STUDENTS.opvn** και συμπληρώνοντας τα ακαδημαϊκά μας στοιχεία συνδεόμαστε στο VPN του ΠαΔΑ. Στην συνέχεια εκτελούμε την εντολή **ssh -i ~/.ssh/sdnKey user@83.212.79.210** και συνδεόμαστε στο VM. Ο λόγος που βάζουμε το **-i ~/.ssh/sdnKey** είναι γιατί δημιουργήσαμε ένα ζεύγος public-private key για αυξημένη ασφάλεια.

Γ. Εικονικές μηχανές με προεγκατεστημένο λογισμικό εξομοίωσης δικτύων SDN

Εκτελούμε όλες τις οδηγίες για την εγκατάσταση των εργαλείων. Στην συνέχεια όταν θέλουμε να δουλέψουμε εκτελούμε τα παρακάτω.

- Ενεργοποιούμε το VM με την εντολή **vboxmanage startvm -type headless "SDN Hub tutorial VM 64-bit with Docker"**
- Συνδεόμαστε στο VM με την εντολή **ssh -p 3022 ubuntu@127.0.0.1** και κωδικό **ubuntu**.

Δ. Λογισμικό προσομοίωσης: Mininet

Ελέγχουμε την διαθεσιμότητα του mininet.

```
ubuntu@sdnhubvm:~[23:49]$ sudo mn -h
```

```
Usage: mn [options]
```

```
(type mn -h for details) ...
```

Εργαστήριο 2

A. Σύνδεση στη μηχανή που φιλοξενεί το λογισμικό προσομοίωσης

Συνδεόμαστε στην κατάλληλη υποδομή.

B. Λογισμικό προσομοίωσης mininet

1. Ελέγξτε τη διαθεσιμότητα του λογισμικού mininet

```
sudo mn -h
```

```
ubuntu@sdnhubvm:~[01:00]$ sudo mn -h
Usage: mn [options]
(type mn -h for details)
```

The mn utility creates Mininet network from the command line. It can create parametrized topologies, invoke the Mininet CLI, and run tests.

Γ. Δημιουργία δικτυώματος με έναν μεταγωγέα σε τοπολογία αστέρα

1. Γενική εντολή δημιουργίας δικτυώματος με OpenVSwitch μεταγωγέα και Controller

Ερώτημα που δεν χρειάζεται απάντηση.

2. Δημιουργία δικτυώματος 3 hosts και OpenVSwitch μεταγωγέα σε τοπολογία αστέρα

```
sudo mn --arp --topo single,3 --mac --switch ovsk --controller remote
```

```
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

mininet>
Ελέγχουμε τους κόμβους.

mininet> nodes
available nodes are:
c0 h1 h2 h3 s1

Για να μάθουμε άλλες εντολές.
mininet> help
Documented commands (type help <topic>):...

3. Ελέγξτε τη διασυνδεσιμότητα σε επίπεδο δικτύου (L3) μεταξύ των hosts h1 και h2

mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4027ms

Αποτυχία, στέλνουμε ICMP echo request και δεν παραλαμβάνουμε κανένα ICMP echo reply. Αυτό συμβαίνει γιατί το flow table του s1 είναι άδειο.

Δ. Λειτουργία εικονικού μεταγωγέα OpenVswitch (OVS)

1. Εργαλείο ovs-ofctl

Με την εντολή αυτή ζητάμε από τον switch να μας δείξει τι γνωρίζει.

sudo ovs-ofctl show tcp:127.0.0.1:6634

OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS
ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst
mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
1(s1-eth1): addr:ee:d7:f6:77:d6:2a
config: 0
state: 0
current: 10GB-FD COPPER
speed: 10000 Mbps now, 0 Mbps max
2(s1-eth2): addr:ba:db:83:03:96:0c
config: 0
state: 0
current: 10GB-FD COPPER
speed: 10000 Mbps now, 0 Mbps max
3(s1-eth3): addr:4e:f0:62:45:c5:18

```
config: 0
state: 0
current: 10GB-FD COPPER
speed: 10000 Mbps now, 0 Mbps max
LOCAL(s1): addr:06:48:9c:4f:65:46
config: PORT_DOWN
state: LINK_DOWN
speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Βλέπουμε ότι το s1 αντιστοιχεί στην port 1, το s2 αντιστοιχεί το port 2 και στο s3 αντιστοιχεί το port 3. Ακόμα βλέπουμε την ταχύτητα των συνδέσεων καθώς και τις mac διευθύνσεις αυτών.

2. Εμφάνιση εγγραφών του πίνακα ροών του switch 1

Εμφανίζουμε τα flows του s1.

```
ubuntu@sdnhubvm:~[01:07]$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
```

Το flow table είναι άδειο.

3. Προσθήκη εγγραφών στον πίνακα ροών (flow table)

Προσθέτουμε flows.

```
ubuntu@sdnhubvm:~[01:09]$ sudo ovs-ofctl add-flow tcp:127.0.0.1:6634
in_port=1,action=output:2
ubuntu@sdnhubvm:~[01:09]$ sudo ovs-ofctl add-flow tcp:127.0.0.1:6634
in_port=2,action=output:1
```

Η πρώτη εντολή βάζει μια ροή στο switch s1 όπου ότι έρχεται από το port 1 να το προωθεί στο port 2. Όμοια και η δεύτερη αλλά ότι έρχεται στο port 2 το προωθεί στο port 1.

4. Επανελέγχος διασυνδεσιμότητας σε επίπεδο δικτύου (L3) μεταξύ των hosts h1 και h2

Κάνουμε ping από τον h1 στο h2.

```
mininet> h1 ping h2 -c3
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=4.04 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.18 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.444 ms
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 0.444/1.889/4.040/1.551 ms
```

Επιτυχία. Στείλαμε 3 ICMP echo request και πήραμε 3 ICMP echo reply. Αυτό γίνεται γιατί προσθέσαμε flows στο flow table και το switch γνωρίζει που να προωθήσει τα πακέτα. Να σημειωθεί ότι αφού δεν βάλαμε flows για το h3 τότε αυτό εξακολουθεί να είναι αποσυνδεδεμένο.

Εργαστήριο 3

A. Γενικά περί Ροών

1. Δημιουργήστε το ακόλουθο δικτύωμα μέσω mininet.

```
ubuntu@sdnhubvm:~[01:47]$ sudo mn --topo single,3 --mac --switch ovsk --  
controller=none
```

```
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2 h3  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1) (h3, s1)  
*** Configuring hosts  
h1 h2 h3  
*** Starting controller  
  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:
```

2. Αντιστοίχιση πορτών

Ελέγχουμε ποιο port αντιστοιχεί που.

```
mininet> sh ovs-ofctl show s1
```

```
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001  
n_tables:254, n_buffers:256  
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS  
ARP_MATCH_IP  
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst  
mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst  
1(s1-eth1): addr:32:01:8c:54:5d:89  
    config: 0  
    state: 0  
    current: 10GB-FD COPPER  
    speed: 10000 Mbps now, 0 Mbps max  
2(s1-eth2): addr:8a:35:b1:f1:9e:0e  
    config: 0  
    state: 0  
    current: 10GB-FD COPPER
```

```
speed: 10000 Mbps now, 0 Mbps max
3(s1-eth3): addr:a2:60:80:eb:cf:62
config: 0
state: 0
current: 10GB-FD COPPER
speed: 10000 Mbps now, 0 Mbps max
LOCAL(s1): addr:06:ec:f1:d7:f0:45
config: PORT_DOWN
state: LINK_DOWN
speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Κάνουμε pingall για να δούμε αν υπάρχει επικοινωνία μεταξύ των host.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
```

Αναμενόμενο αποτέλεσμα αφού το flow table είναι άδειο.

3. Δημιουργία ροής στον Πίνακα Ροών.

Προσθέτουμε ροή και ξανακάνουμε pingall.

```
mininet> sh ovs-ofctl add-flow s1 action=normal
```

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

Παρατηρούμε ότι υπάρχει επικοινωνία μεταξύ των host. Ελέγχουμε τα περιεχόμενα του flow table.

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=99.207s, table=0, n_packets=24, n_bytes=1680, idle_age=64,
actions=NORMAL
```

Το flow table του switch έχει μία εγγραφή. Το switch λειτουργεί σαν ένα απλό L2 switch αφού το βάλαμε σε τυπική λειτουργία προώθησης (normal).

4. Διαγραφή ροής από τον Πίνακα Ροών.

Διαγράφουμε τα περιεχόμενα του flow table και ελέγχουμε.

```
mininet> sh ovs-ofctl del-flows s1
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
```

B. Ροές με στοιχεία 1ου επιπέδου (OSI physical layer)

1. Δημιουργήστε το ακόλουθο δίκτυο μέσω mininet.

```
ubuntu@sdnhubvm:~[01:47]$ sudo mn --topo single,3 --mac --switch ovsk --controlle
```

Προσθέτουμε τις ροές και στις συνέχειες κάνουμε ping από τον host h1 τον host h2.

```
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=1,action=output:2
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=2,action=output:1
mininet> h1 ping h2 -c3
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=3.59 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.392 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.421 ms
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.392/1.469/3.594/1.502 ms
```

Παρατηρούμε ότι το ping ήταν επιτυχές. Στην συνέχεια βάζουμε την καινούργια ροή και κάνουμε pingall.

```
mininet> sh ovs-ofctl add-flow s1 priority=32768,action=drop
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
```

Βλέπουμε ότι επειδή αυτό το flow έχει μεγαλύτερη προτεραιότητα, όλα τα πακέτα θα γίνουν drop.

2. Αφαίρεση ροών που έχουν default παραμέτρους

Αφαιρούμε όλες τα flows με default παραμέτρους, δηλαδή στην συγκεκριμένη περίπτωση priority=32768.

```
mininet> sh ovs-ofctl del-flows s1 --strict
```

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
  cookie=0x0, duration=2861.676s, table=0, n_packets=10, n_bytes=756,
  idle_age=1248, priority=500,in_port=1 actions=output:2
  cookie=0x0, duration=2852.654s, table=0, n_packets=4, n_bytes=336, idle_age=2671,
  priority=500,in_port=2 actions=output:1
```

Και τέλος αφαιρούμε όλες τα υπόλοιπα flow.

```
mininet> sh ovs-ofctl del-flows s1
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
```

Γ. Ροές με στοιχεία 2ου επιπέδου (OSI data-link layer)

1. Προσθήκη ροών στον Πίνακα Ροών

Προσθέτουμε τα flows και εκτελούμε την pingall.

```
mininet> sh ovs-ofctl add-flow s1
dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,actions=output:2
mininet> sh ovs-ofctl add-flow s1
dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,actions=output:1
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
```

Παρατηρούμε ότι το pingall απέτυχε. Αυτό συμβαίνει γιατί το ICMP λειτουργεί στο επίπεδο 3 κατά OSI. Για να λειτουργήσει χρειάζεται τις mac τις οποίες τις μαθαίνει με την χρήση του ARP πρωτοκόλλου. Όμως σύμφωνα με το flow table που έχουμε, η κίνηση του ARP απορρίπτεται.

2. Ενεργοποίηση ARP πρωτοκόλλου

Δημιουργούμε flow για την κίνηση του ARP και στην συνέχεια ελέγχουμε το περιεχόμενο του flow table και εκτελούμε pingall.

```
mininet> sh ovs-ofctl add-flow s1 dl_type=0x806,nw_proto=1,action=flood
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
  cookie=0x0, duration=25.800s, table=0, n_packets=0, n_bytes=0, idle_age=25,
  dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02 actions=output:2
```

```

cookie=0x0, duration=18.879s, table=0, n_packets=0, n_bytes=0, idle_age=18,
dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions=output:1
mininet> sh ovs-ofctl add-flow s1 dl_type=0x806,nw_proto=1,action=flood
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X
h2 -> h1 X
h3 -> X X
*** Results: 66% dropped (2/6 received)

```

Βλέπουμε ότι το ping πέτυχε ανάμεσα στους h1 και h2 αλλά όχι στον h3 αφού δεν υπάρχει flow για αυτόν. Ακόμα αφαιρούμε όλες τις εγγραφές από το flow table.

```

mininet> sh ovs-ofctl del-flows s1
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):

```

Δ. Ροές με στοιχεία 3ου επιπέδου (OSI network layer)

1. Δικτυακές ρυθμίσεις

Βρίσκουμε τις ρυθμίσεις για κάθε host.

```

mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet  HWaddr 00:00:00:00:00:01
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::200:ff:fe00:1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:21 errors:0 dropped:0 overruns:0 frame:0
          TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1306 (1.3 KB)  TX bytes:1270 (1.2 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:112 (112.0 B)  TX bytes:112 (112.0 B)

mininet> h2 ifconfig
h2-eth0  Link encap:Ethernet  HWaddr 00:00:00:00:00:02
          inet addr:10.0.0.2  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::200:ff:fe00:2/64 Scope:Link

```

```
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:21 errors:0 dropped:0 overruns:0 frame:0
TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:1306 (1.3 KB) TX bytes:1270 (1.2 KB)
```

```
lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING MTU:65536 Metric:1
      RX packets:1 errors:0 dropped:0 overruns:0 frame:0
      TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:112 (112.0 B) TX bytes:112 (112.0 B)
```

```
mininet> h3 ifconfig
```

```
h3-eth0  Link encap:Ethernet HWaddr 00:00:00:00:00:03
          inet addr:10.0.0.3 Bcast:10.255.255.255 Mask:255.0.0.0
          inet6 addr: fe80::200:ff:fe00:3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:15 errors:0 dropped:0 overruns:0 frame:0
          TX packets:23 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:942 (942.0 B) TX bytes:1438 (1.4 KB)
```

```
lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING MTU:65536 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

2. Επικοινωνία στο 3ο επίπεδο με προτεραιότητα, βάσει κατηγοριοποίησης του μηχανισμού DiffServ.

Προσθέτουμε τα απαιτούμενα flow και στην συνέχεια δοκιμάζουμε την επικοινωνία με pingall.

```
mininet> sh ovs-ofctl add-flow s1
priority=500,dl_type=0x800,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=normal
mininet> sh ovs-ofctl add-flow s1
priority=800,dl_type=0x800,nw_src=10.0.0.3,nw_dst=10.0.0.0/24,actions=mod_nw_tos:
184,normal
```

Η επικοινωνία αποτυγχάνει. Το πρώτο flow έχει priority 500, αφορά κίνηση σε επίπεδο IP και συγκεκριμένα IPv4 πακέτα. Το δίκτυο προέλευσης και το δίκτυο προορισμού είναι το υπάρχον και το normal δηλώνει ότι θα λειτουργεί σαν ένα switch σε L2 κατά OSI. Το δεύτερο flow έχει ακριβώς τα ίδιο χαρακτηριστικά εκτός του action όπου δίνει προτεραιότητα σε ότι έρχεται από τον h3. Το 184 το χρησιμοποιούμε για να καθορίσουμε την τιμή του DSCP. Μετατοπίζοντας το 184 2 bits αριστερά προκύπτει το 46 όπου σε πεδίο TOS δηλώνει την μεγαλύτερη προτεραιότητα. Η επικοινωνία μεταξύ των host δεν θα είναι εφικτή καθώς πάλι δεν έχουμε επιτρέψει την ARP κίνηση.

2. Ενεργοποίηση ARP πρωτοκόλλου

Επιτρέπουμε την ARP κίνηση.

```
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.1,actions=output:1
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.2,actions=output:2
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.3,actions=output:3
```

Δοκιμάζουμε την επικοινωνία η οποία πετυχένει και τέλος καθαρίζουμε το flow table.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> sh ovs-ofctl del-flows s1
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
```

E. Ροές με στοιχεία 4ου επιπέδου (OSI transport layer)

1. Προσθήκη web server σε σταθμό της τοπολογίας

Φτιάχνουμε έναν web server που ακούει στην port 80.

```
mininet> h3 python -m SimpleHTTPServer 80 &
```

2. Ενεργοποίηση ARP πρωτοκόλλου

```
mininet> sh ovs-ofctl add-flow s1 arp,actions=normal
```

Η διαφορά των ξεχωριστών εντολών που χρησιμοποιήσαμε παραπάνω με την εντολή που κάνει χρήση του flooding είναι ότι εάν έχουμε πολλά για παράδειγμα ports τότε θα χρειαστούμε και πολλά flow.

3. Προσθήκη ροής για κίνηση TCP πρωτοκόλλου

Εκτελούμε την εντολή για την προσθήκη flow.

```
mininet> sh ovs-ofctl add-flow s1  
priority=500,dl_type=0x800,nw_proto=6,tp_dst=80,actions=output:3
```

Το flow έχει priority 500, αφορά επίπεδο IP, αφορά το πρωτόκολλο TCP με port προορισμού 80 και ότι αντιστοιχεί σε αυτά να προωθείτε στο port 3.

4. Προσθήκη ροής για τον σταθμό h3

Προσθέτουμε το παρακάτω flow. Ο λόγος είναι ότι με το προηγούμενο flow αφορούσε την κίνηση προς το h3. Για να δώσει απάντηση το h3 όμως χρειαζόμαστε και το παρακάτω flow.

```
mininet> sh ovs-ofctl add-flow s1 priority=800,ip,nw_src=10.0.0.3,actions=normal
```

5. Δοκιμή επικοινωνίας με τον web server

Δοκιμάζουμε την επικοινωνία και βλέπουμε ότι επιτυγχάνει. Με την εντολή curl ζητάμε από τον h3 να μας δώσει δεδομένα.

```
mininet> h1 curl h3  
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>  
<title>Directory listing for /</title>  
<body>  
<h2>Directory listing for /</h2>  
<hr>  
<ul>  
<li><a href=".bash_history">.bash_history</a>  
<li><a href=".bash_logout">.bash_logout</a>  
<li><a href=".bash_profile">.bash_profile</a>  
<li><a href=".bashrc">.bashrc</a>  
<li><a href=".cache/">.cache/</a>  
<li><a href=".config/">.config/</a>  
<li><a href=".dbus/">.dbus/</a>  
<li><a href=".dmrc">.dmrc</a>  
<li><a href=".eclipse/">.eclipse/</a>  
<li><a href=".gconf/">.gconf/</a>  
<li><a href=".gem/">.gem/</a>  
<li><a href=".gitconfig">.gitconfig</a>  
<li><a href=".gnome2/">.gnome2/</a>  
<li><a href=".gnome2_private/">.gnome2_private/</a>  
<li><a href=".gstreamer-0.10/">.gstreamer-0.10/</a>  
<li><a href=".ICEauthority">.ICEauthority</a>  
<li><a href=".irb-history">.irb-history</a>
```

- .karaf/
- .lessht
- .local/
- .m2/
- .mininet_history
- .mozilla/
- .oracle_jre_usage/
- .pip/
- .profile
- .pylint.d/
- .rnd
- .ssh/
- .thumbnails/
- .tmux.conf
- .vboxclient-clipboard.pid
- .vboxclient-display.pid
- .vboxclient-draganddrop.pid
- .vboxclient-seamless.pid
- .vim/
- .viminfo
- .viminfo.tmp
- .vimperatorrc
- .vimrc
- .watershed/
- .wireshark/
- .Xauthority
- .xscreensaver
- .xsession-errors
- .xsession-errors.old
- .zlogin
- .zshrc
- apache-maven-3.3.3
- cgi-bin/
- Desktop/
- Documents/
- Downloads/
- eclipse-workspace/
- floodlight/
- linc-config-generator/
- linc-oe/
- lorispack/
- mininet/
- Music/
- oflops/
- onos/
- openflow/

```
<li><a href="openvswitch/">openvswitch/</a>
<li><a href="Pictures/">Pictures/</a>
<li><a href="pox/">pox/</a>
<li><a href="Public/">Public/</a>
<li><a href="pyretic/">pyretic/</a>
<li><a href="ryu/">ryu/</a>
<li><a href="sdnhub.png">sdnhub.png</a>
<li><a href="SDNHub_Opendaylight_Tutorial/">SDNHub_Opendaylight_Tutorial/</a>
<li><a href="Templates/">Templates/</a>
<li><a href="trema/">trema/</a>
<li><a href="Videos/">Videos/</a>
</ul>
<hr>
</body>
</html>
```


Εργαστήριο 4

A. Συμμετοχή του Ελεγκτή OpenDaylight (ODL) στο δίκτυωμα

1. Ενεργοποιήστε τον Ελεγκτή ODL στην μηχανή VM.Okeanos.

Προσαρμόζουμε το JAVA_HOME.

```
nano ./distribution-karaf-0.4.4-Beryllium-SR4/bin/setenv
```

Έχουμε

```
export JAVA_HOME="/usr/lib/jvm/default-jre" # Location of Java installation
```

Ενεργοποιούμε τον ελεγκτή.

```
sudo ./distribution-karaf-0.4.4-Beryllium-SR4/bin/karaf
```

Ελέγχουμε ότι αναμένουν σύνδεση τα TCP ports 8181 και 6633.

```
user@snf-26789:~$ sudo lsof -i:8181
```

```
COMMAND PID USER  FD  TYPE DEVICE SIZE/OFF NODE NAME
```

```
java    6659 root  348u IPv6 313406   0t0  TCP *:8181 (LISTEN)
```

```
user@snf-26789:~$ sudo lsof -i:6633
```

```
COMMAND PID USER  FD  TYPE DEVICE SIZE/OFF NODE NAME
```

```
java    6659 root  611u IPv6 304882   0t0  TCP *:6633 (LISTEN)
```

Ελέγχουμε τις εγκατεστημένες δυνατότητες.

```
.opendaylight-user@root>feature:list -i
```

Name	Version	Installed	Repository	Description
------	---------	-----------	------------	-------------

standard	3.0.3	x	standard-3.0.3	Karaf standard feature
config	3.0.3	x	standard-3.0.3	Provide OSGi ConfigAdmin support
region	3.0.3	x	standard-3.0.3	Provide Region Support

package	3.0.3	x	standard-3.0.3	Package commands and mbeans
kar	3.0.3	x	standard-3.0.3	Provide KAR (KARaf archive) support
ssh	3.0.3	x	standard-3.0.3	Provide a SSHd server on Karaf
management	3.0.3	x	standard-3.0.3	Provide a JMX MBeanServer and a set of MBeans in K

Εγκαθιστούμε τα απαιτούμενα.

```
opendaylight-user@root>feature:install odl-mdsal-clustering
```

```
opendaylight-user@root>feature:install odl-restconf odl-l2switch-switch odl-l2switch-switch-ui odl-mdsal-apidocs odl-dlux-all
```

Refreshing bundles org.eclipse.persistence.moxy (122), org.eclipse.persistence.core (121), org.jboss.netty (159), com.google.guava (64)

Εκτελούμε τα απαιτούμενα.

```
user@snf-26789:~$ curl -u "admin:admin" http://83.212.79.210:8181/index.html
```

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="utf-8" />
```

```
<title>OpenDaylight Dlux</title>
```

Τέλος εγκαθιστούμε τα fglr, xserver-xorg-core, xserver-xorg και wireshark.


2. Δημιουργήστε το ακόλουθο δίκτυο μέσω mininet.

```
ubuntu@sdnhubvm:~[15:34]$ sudo mn --controller=remote,ip=83.212.79.210,port=6633
--topo single,6 --mac --switch ovsk
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

3. Χρήση γραφικής διεπαφής του ODL

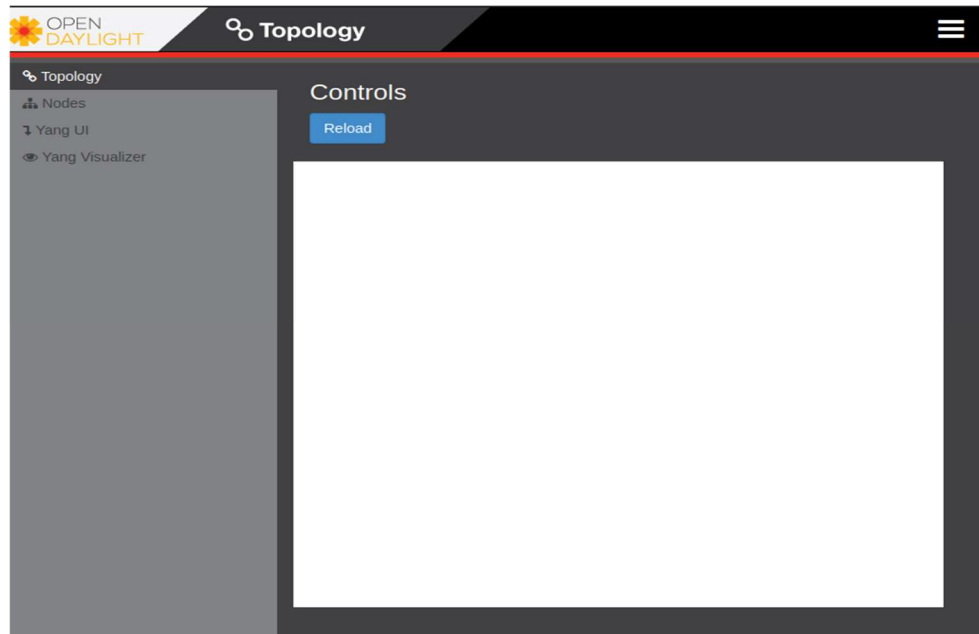
Πηγαίνουμε στην διεύθυνση <http://83.212.79.210:8181/index.html> και συνδεόμαστε με username admin και password admin.

Please Sign In

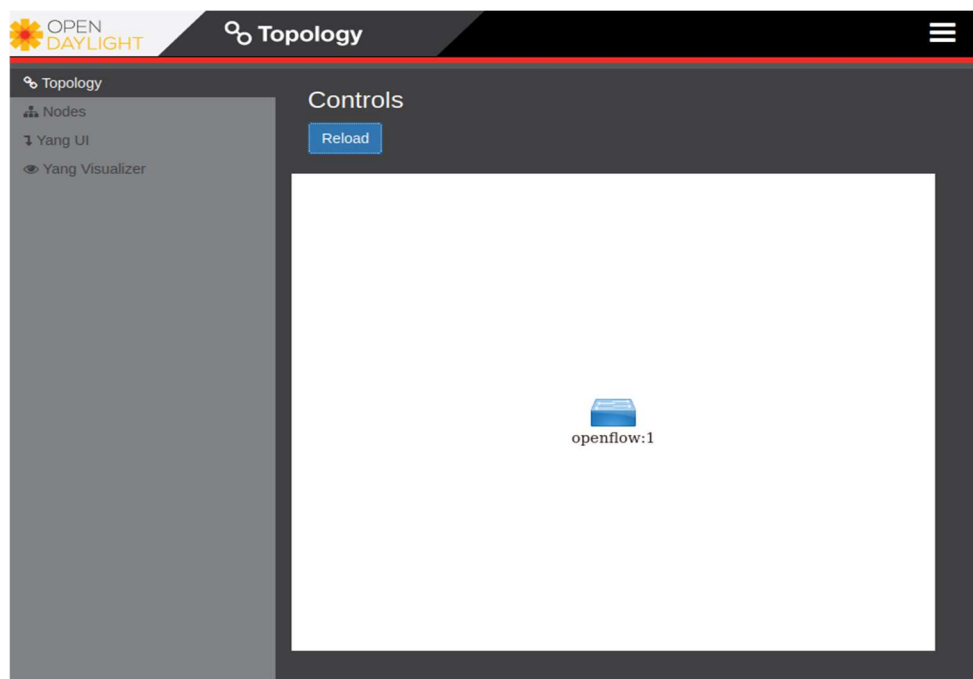
 OPEN
DAYLIGHT

☐ Remember Me

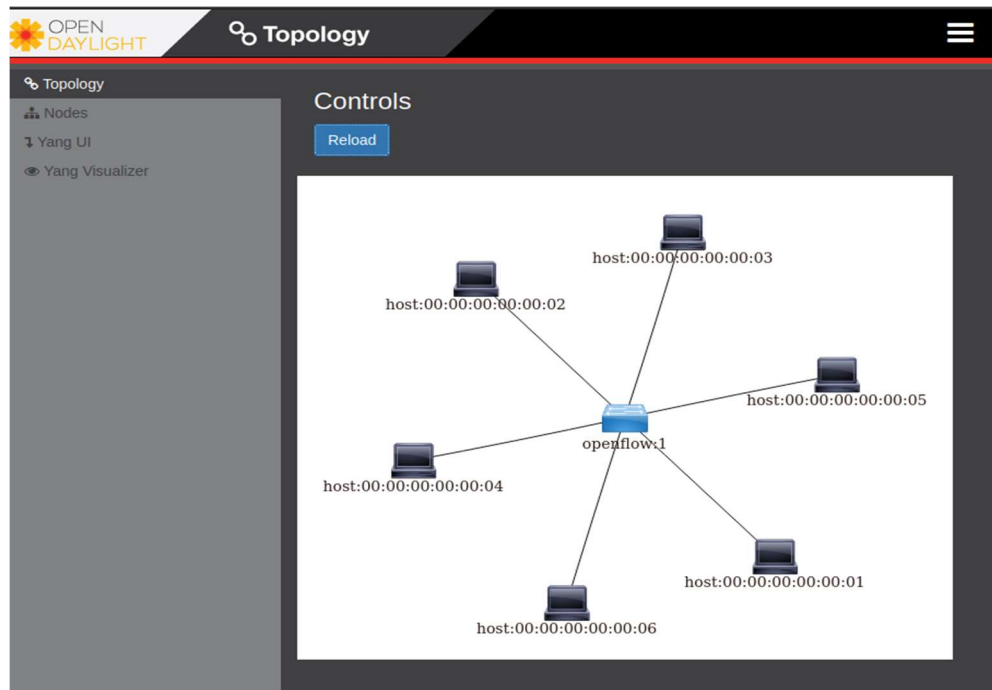
Login



Βλέπουμε το interface και πατάμε reload για να μας εμφανίσει το switch.



Τέλος αφού κάνουμε ringall πατάμε reload και βλέπουμε την τοπολογία.



Καταγράφουμε την ανταλλαγή των πακέτων στο wireshark και την αποθηκεύουμε στο capture.pcapng.

Ζητούμενο 2

Δημιουργούμε το δικτύωμα με την παρακάτω εντολή.

```
ubuntu@sdnhubvm:~[23:54]$ sudo mn --controller=none --topo linear,3 --mac --switch  
ovsk
```

```
*** Creating network
```

```
*** Adding controller
```

```
*** Adding hosts:
```

```
h1 h2 h3
```

```
*** Adding switches:
```

```
s1 s2 s3
```

```
*** Adding links:
```

```
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
```

```
*** Configuring hosts
```

```
h1 h2 h3
```

```
*** Starting controller
```

```
*** Starting 3 switches
```

```
s1 s2 s3 ...
```

```
*** Starting CLI:
```

Για να επικοινωνήσουν οι σταθμοί εκτελούμε τα παρακάτω. Αρχικά προσθέτουμε flows για την κίνηση στο L3 IP κατά OSI. Δεύτερον επιτρέπουμε την ARP κίνηση και τέλος ελέγχουμε την επικοινωνία με pingall η οποία επιτυγχάνει.

```
sh ovs-ofctl add-flow s1
```

```
priority=500,dl_type=0x800,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=normal
```

```
sh ovs-ofctl add-flow s2
```

```
priority=500,dl_type=0x800,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=normal
```

```
sh ovs-ofctl add-flow s3
```

```
priority=500,dl_type=0x800,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=normal
```

```
sh ovs-ofctl add-flow s1 arp,actions=normal
```

```
sh ovs-ofctl add-flow s2 arp,actions=normal
```

```
sh ovs-ofctl add-flow s3 arp,actions=normal
```

```
mininet> pingall
```

```
*** Ping: testing ping reachability
```

```
h1 -> h2 h3
```

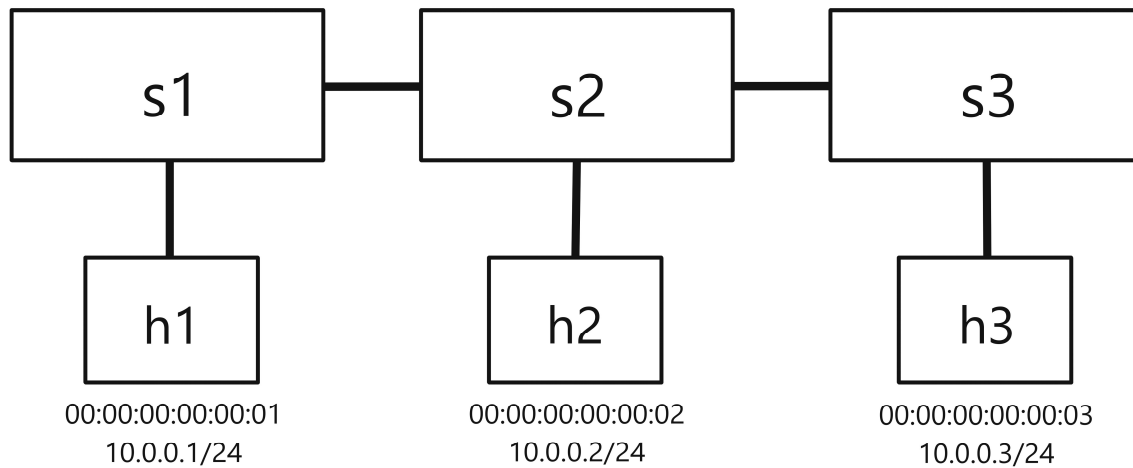
```
h2 -> h1 h3
```

```
h3 -> h1 h2
```

```
*** Results: 0% dropped (6/6 received)
```

Ζητούμενο 3

Παρουσιάζουμε το παραπάνω δικτύωμα ως σχήμα.



Η IPv4 του μηχανήματος στον Οκεανος που χρησιμοποιήθηκε είναι 83.212.79.210.