Διαχείριση Δικτύων – Ευφυή Δίκτυα

ΟΝΟΜΑ: ΠΑΠΑΔΟΠΟΥΛΛΟΣ ΜΙΧΑΛΗΣ
Α.Μ: 03114702
ΟΝΟΜΑ: ΚΑΡΔΑΡΗΣ ΧΑΡΑΛΑΜΠΟΣ
Α.Μ: 03114074

ΟΜΑΔΑ: netmg034

5η Άσκηση

Χρήσιμες Παραπομπές:

I. **OpenFlow Switch Specification Version 1.0.0 (Wire Protocol 0x01)**
https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf

II. **Mininet (An Instant Virtual Network on your PC)**
http://mininet.org/

III. **RYU OpenFlow Controller**
https://osrg.github.io/ryu/

Άσκηση

1) ΔΗΜΙΟΥΡΓΙΑ ΕΙΚΟΝΙΚΗΣ ΜΗΧΑΝΗΣ ΚΑΙ ΣΥΝΔΕΣΗ ΣΕ ΑΥΤΗΝ

Δημιουργήστε εικονική μηχανή στην υπηρεσία ~okeanos (okeanos.grnet.gr) ακολουθώντας τις οδηγίες που θα βρείτε στον σύνδεσμο:
https://okeanos.grnet.gr/support/user-guide/cyclades-how-to-create-a-vm/

Το image του λειτουργικού συστήματος που θα επιλέξετε για την εικονική μηχανή σας λέγεται **«netman2017-2018»** και βρίσκεται στην κατηγορία (image type) που ονομάζεται **“Public”.**
Στο βήμα που σας δίνεται η δυνατότητα για την χρήση Public IPv4 επιλέγετε ότι θέλετε να κάνετε χρήση.
Όταν δημιουργηθεί η εικονική μηχανή **θα πρέπει να σημειώσετε το password πρόσβασης** σε αυτό, που θα δημιουργήσει η υπηρεσία ~okeanos και θα σας παρουσιάσει σε παράθυρο του φυλλομετρητή. Εναλλακτικά μπορείτε να προσθέσετε το public key που δημιουργήσατε στην άσκηση 4 (το κλειδί πρέπει να προστεθεί πριν δημιουργηθεί το μηχάνημα).

Τέλος, συνδεθείτε με ssh client (putty) στην Public IP της εικονικής μηχανής, με **username “netman”.**

2) ΕΚΚΙΝΗΣΗ OPENFLOW CONTROLLER
Εκτελέστε τις παρακάτω εντολές για να ξεκινήσει ο RYU OpenFlow controller στην TCP port 6633. Η λογική με την οποία θα εγκαθιδρύει ο OpenFlow controller flow-rules μέσα στα OpenFlow switches ορίζεται βάσει της εφαρμογής που θα χρησιμοποιηθεί.

Συγκεκριμένα θα χρησιμοποιηθεί η εφαρμογή ***simple_switch.py*** η οποία «προσομοιώνει» την λειτουργία mac learning ενός συμβατικού L2 μεταγωγέα.

cd /home/netman/ryu/
python ./bin/ryu-manager ./ryu/app/simple_switch.py

Προαιρετικά μπορείτε να χρησιμοποιήσετε στο πέρας της προηγούμενης εντολής τη flag --verbose για να δείτε αναλυτικότερα την εκτέλεση της εφαρμογής.

3) ΕΚΚΙΝΗΣΗ ΕΞΟΜΟΙΩΤΗ MININET
Η παράμετρος «--controller remote,ip=127.0.0.1,port=6633» επιβάλλει ότι ο έλεγχος των OpenFlow switches θα γίνεται από ένα κεντρικό controller (βλ. βήμα 2).

sudo mn --topo single,3 --mac --switch ovsk --controller remote,ip=127.0.0.1,port=6633

Η τοπολογία που θα δημιουργηθεί είναι η ακόλουθη:

```
          --- host1
switch1   --- host2
          --- host3
```

ΣΗΜΕΙΩΣΗ: Σε περίπτωση που πάρετε το ακόλουθο μήνυμα κατά την εκτέλεση της εντολής ***mn*** :

ovs-vsctl exited with code -14
**** Error connecting to ovs-db with ovs-vsctl*

εκτελέστε την εντολή ***service openvswitch-switch start*** και στη συνέχεια εκτελέστε εκ νέου την εντολή για την έναρξη του mininet.

switch features ev
version=0x1,msg_type=0x6,msg_len=0xe0,xid=0x6120c868,OFPSwitchFeatures(actions=4095,capabilities=199,datapath_id=1,n_buffers=256,n_tables=254,ports={1: OFPPhyPort(port_no=1,hw_addr='6a:50:18:a0:8a:25',name='s1-eth1',config=0,state=0,curr=192,advertised=0,supported=0,peer=0), 2: OFPPhyPort(port_no=2,hw_addr='96:70:7c:f1:4e:2e',name='s1-eth2',config=0,state=0,curr=192,advertised=0,supported=0,peer=0), 3: OFPPhyPort(port_no=3,hw_addr='1a:bc:15:d0:c5:b8',name='s1-eth3',config=0,state=0,curr=192,advertised=0,supported=0,peer=0), 65534:

OFPPhyPort(port_no=65534,hw_addr='16:59:14:f
8:d6:4e',name='s1',config=1,state=1,curr=0,advertised=0,supported=0,peer=0)
})

4) (A) Εκτελώντας την εντολή «s1 ovs-vsctl show» στο mininet cli (*mininet>*) μπορείτε να βρείτε ότι τα OpenFlow switches ακούν στην TCP πόρτα 663X (π.χ. ptcp:6634). Καταγράψτε σε ποια TCP πόρτα ακούει το κάθε OpenFlow switch και πόσες πόρτες έχει.

```
mininet> s1 ovs-vsctl show
ecf5997b-e954-4592-826a-c2fd43bf9093
    Bridge "s1"
        Controller "ptcp:6654"
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        fail_mode: secure
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth3"
            Interface "s1-eth3"
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1-eth2"
            Interface "s1-eth2"
    ovs_version: "2.3.0"
```

```
mininet> ports
s1 lo:0 s1-eth1:1 s1-eth2:2 s1-eth3:3
```

(B) Καταγράψτε τις IP διευθύνσεις των hosts.

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

c0 → Controller
h1 → Host #1
h2 → Host #2
h3 → Host #3
s1 → Switch

```
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
h3-eth0<->s1-eth3 (OK OK)
```

```
mininet> s1 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UP group default qlen 1000
    link/ether aa:00:03:39:ed:d7 brd ff:ff:ff:ff:ff:ff
    inet6 2001:648:2ffc:1225:a800:3ff:fe39:edd7/64 scope global mngtmpaddr
dynamic
       valid_lft forever preferred_lft forever
    inet6 fe80::a800:3ff:fe39:edd7/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UP group default qlen 1000
    link/ether aa:0c:f3:46:de:69 brd ff:ff:ff:ff:ff:ff
    inet 83.212.98.34/21 brd 83.212.103.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::a80c:f3ff:fe46:de69/64 scope link
       valid_lft forever preferred_lft forever
4: s1-eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast master ovs-system state UP group default qlen 1000
    link/ether 4e:dc:87:8f:e8:9f brd ff:ff:ff:ff:ff:ff
    inet6 fe80::4cdc:87ff:fe8f:e89f/64 scope link
       valid_lft forever preferred_lft forever
5: s1-eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast master ovs-system state UP group default qlen 1000
    link/ether 6a:24:af:37:92:ab brd ff:ff:ff:ff:ff:ff
    inet6 fe80::6824:afff:fe37:92ab/64 scope link
       valid_lft forever preferred_lft forever
6: s1-eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast master ovs-system state UP group default qlen 1000
    link/ether a6:7a:45:fe:b5:04 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::a47a:45ff:fefe:b504/64 scope link
       valid_lft forever preferred_lft forever
7: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state
DOWN group default
    link/ether 0e:71:ae:c1:05:21 brd ff:ff:ff:ff:ff:ff
8: s1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
group default
    link/ether 62:9b:50:de:eb:4c brd ff:ff:ff:ff:ff:ff
```

```
mininet> px print h1.cmd('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:50 errors:0 dropped:0 overruns:0 frame:0
          TX packets:37 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3840 (3.7 KiB)
              TX bytes:2850 (2.7 KiB)

mininet> px print h2.cmd('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:51 errors:0 dropped:0 overruns:0 frame:0
          TX packets:37 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3918 (3.8 KiB)  TX bytes:2850 (2.7 KiB)

mininet> px print h3.cmd('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:46 errors:0 dropped:0 overruns:0 frame:0
          TX packets:33 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3448 (3.3 KiB)  TX bytes:2458 (2.4 KiB)
```

5) Στο mininet cli εκτελέστε την εντολή "*h1 ping –c 3 h2*" και αφού τερματίσει, εκτελέστε εκ νέου άμεσα την εντολή "*s1 dpctl dump-flows tcp:127.0.0.1:6654*".

```
mininet> h1 ping -c3 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=0.538 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.140 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.141 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.140/0.273/0.538/0.187 ms

mininet> s1 dpctl dump-flows tcp:127.0.0.1:6654
stats_reply (xid=0x25e4df64): flags=none type=1(flow)
  cookie=0, duration_sec=41s, duration_nsec=842000000s, table_id=0,
priority=32768, n_packets=4, n_bytes=336,
idle_timeout=0,hard_timeout=0,in_port=2,dl_dst=00:00:00:00:00:01,actions=
output:1
  cookie=0, duration_sec=41s, duration_nsec=834000000s, table_id=0,
priority=32768, n_packets=3, n_bytes=238,
idle_timeout=0,hard_timeout=0,in_port=1,dl_dst=00:00:00:00:00:02,actions=
output:2
```

(A) Ερμηνεύστε τα αποτελέσματα της συγκεκριμένης εντολής με την βοήθεια της παραπομπής (I), εξετάζοντας μόνο τα flows που αφορούν τους hosts.

Βλεπουμε στο dump τα εξης:

n_packets=4 →
in_port=2 → Input Switch Port
dl_dst=00:00:00:00:00:01 → Host's #1 Data Link Destination Address (Ethernet/MAC Address)
actions=output:1 → Matching packet, must be forwarded to port #1

n_packets=3 →
in_port=1 → Input Switch Port
dl_dst=00:00:00:00:00:02 → Host's #2 MAC Address
actions=output:2 → Matching packet, must be forwarded to port #2

(Β) Τερματίστε την εφαρμογή του RYU Controller (Ctrl+C) και δοκιμάστε να επαναλάβετε την εντολή του προηγούμενου βήματος. Δοκιμάστε να εκτελέσετε την εντολή "*h1 ping –c 3 h3*" . Τι παρατηρείτε;

```
mininet> h1 ping -c3 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable

--- 10.0.0.3 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2015ms
pipe 3
```

Παρατηρούμε ότι παίρνουμε μήνυμα σφάλματος destination host unreachable στην περίπτωση του ping στον νέο host h3. Αντίθετα στον host h2 δεν παρατηρείται πρόβλημα. Αυτό συμβαίνει επειδή είναι γνωστή πλέον η διεύθυνση MAC του h2 από την προηγούμενη εκτέλεση της εντολής όταν είχαμε ενεργοποιημένο τον controller. Αντίθετα, μετά δεν έχουμε controller και άρα δεν μπορεί να γίνει το MAC resolution.

(Γ) Εκτελέστε εκ νέου την εφαρμογή simple_switch.py και εκτελέστε την εντολή pingall στο mininet.Κλείστε και πάλι την εφαρμογή. Τι παρατηρείτε στο flow table του switch.

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

```
RYU:
EVENT ofp_event->SimpleSwitch EventOFPPacketIn
packet in 1 00:00:00:00:00:01 ff:ff:ff:ff:ff:ff 1
EVENT ofp_event->SimpleSwitch EventOFPPacketIn
packet in 1 00:00:00:00:00:03 00:00:00:00:00:01 3
EVENT ofp_event->SimpleSwitch EventOFPPacketIn
packet in 1 00:00:00:00:00:01 00:00:00:00:00:03 1
EVENT ofp_event->SimpleSwitch EventOFPPacketIn
packet in 1 00:00:00:00:00:02 ff:ff:ff:ff:ff:ff 2
EVENT ofp_event->SimpleSwitch EventOFPPacketIn
packet in 1 00:00:00:00:00:03 00:00:00:00:00:02 3
EVENT ofp_event->SimpleSwitch EventOFPPacketIn
packet in 1 00:00:00:00:00:02 00:00:00:00:00:03 2
```

Παρατηρούμε την διαδικασία MAC resolution μεταξύ των κόμβων που δεν έχουν ανταλλάξει μήνυμα μέχρι τώρα. Συγκεκριμένα, ο h1 ρωτάει για την MAC του h3 και μετά του κάνει ping και ακολούθως αντίστοιχα ο h2 ρωτάει για την MAC του h3. Οι υπόλοιπες MAC είναι γνωστές στους κόμβους ανά ζεύγη οπότε έχουμε άμεση αποστολή των ICMP πακέτων στον αντίστοιχο κόμβο.

```
mininet> s1 dpctl dump-flows tcp:127.0.0.1:6654
stats_reply (xid=0x9810b087): flags=none type=1(flow)

  cookie=0, duration_sec=165s, duration_nsec=695000000s, table_id=0,
priority=32768, n_packets=3, n_bytes=238,
idle_timeout=0,hard_timeout=0,in_port=3,dl_dst=00:00:00:00:00:02,actions=
output:2

  cookie=0, duration_sec=1729s, duration_nsec=768000000s, table_id=0,
priority=32768, n_packets=15,
n_bytes=1246,
idle_timeout=0,hard_timeout=0,in_port=2,dl_dst=00:00:00:00:00:01,actions=
output:1

  cookie=0, duration_sec=165s, duration_nsec=724000000s, table_id=0,
priority=32768, n_packets=3, n_bytes=238,
idle_timeout=0,hard_timeout=0,in_port=3,dl_dst=00:00:00:00:00:01,actions=
output:1

  cookie=0, duration_sec=165s, duration_nsec=689000000s, table_id=0,
priority=32768, n_packets=2, n_bytes=140,
idle_timeout=0,hard_timeout=0,in_port=2,dl_dst=00:00:00:00:00:03,actions=
output:3

  cookie=0, duration_sec=165s, duration_nsec=718000000s, table_id=0,
priority=32768, n_packets=2, n_bytes=140,
idle_timeout=0,hard_timeout=0,in_port=1,dl_dst=00:00:00:00:00:03,actions=
output:3

  cookie=0, duration_sec=1729s, duration_nsec=760000000s, table_id=0,
priority=32768, n_packets=14,
n_bytes=1148,
idle_timeout=0,hard_timeout=0,in_port=1,dl_dst=00:00:00:00:00:02,actions=
output:2
```

Το flow-table του switch δείχνει την εγκαθίδρυση 6 κανόνων για επικοινωνία μεταξύ των hosts. 1 κανόνα για κάθε διατεταγμένο ζεύξος hosts.

6) Στη συνέχεια της άσκησης θα χρησιμοποιηθεί η εφαρμογή που παρέχει REST API για τον RYU. Μέσω της εφαρμογής αυτής είναι δυνατή η εισαγωγή, διαγραφή και προβολή OpenFlow Rules με χρήση του πρωτοκόλλου HTTP. Για την εκτέλεση της εφαρμογής ανοίξτε ένα νέο ssh session συνδεθείτε στο εικονικό μηχάνημα και εκτελέστε τις παρακάτω εντολές:

cd /home/netman/ryu/
python ./bin/ryu-manager ./ryu/app/ofctl_rest.py

Δημιουργείστε ακόμα ένα ssh session. Χρησιμοποιώντας πληροφορία που θα βρείτε στο *http://ryu.readthedocs.io/en/latest/app/ofctl_rest.html*

(A) Να βρεθούν και να καταγραφούν τα flows στο OpenFlow table του switch.

GET 127.0.0.1:8080/stats/flow/1


```
{
   "1": [
      {
         "actions": [
            "OUTPUT:2"
         ],
         "idle_timeout": 0,
         "cookie": 0,
         "packet_count": 7,
         "hard_timeout": 0,
         "byte_count": 518,
         "duration_sec": 12555,
         "duration_nsec": 576000000,
         "priority": 32768,
         "length": 96,
         "flags": 1,
         "table_id": 0,
         "match": {
            "dl_dst": "00:00:00:00:00:02",
            "in_port": 3
         }
      },
```

```
{
    "actions": [
        "OUTPUT:1"
    ],
    "idle_timeout": 0,
    "cookie": 0,
    "packet_count": 19,
    "hard_timeout": 0,
    "byte_count": 1526,
    "duration_sec": 14119,
    "duration_nsec": 649000000,
    "priority": 32768,
    "length": 96,
    "flags": 1,
    "table_id": 0,
    "match": {
        "dl_dst": "00:00:00:00:00:01",
        "in_port": 2
    }
},
{
    "actions": [
        "OUTPUT:1"
    ],
    "idle_timeout": 0,
    "cookie": 0,
    "packet_count": 7,
    "hard_timeout": 0,
    "byte_count": 518,
    "duration_sec": 12555,
    "duration_nsec": 605000000,
    "priority": 32768,
    "length": 96,
    "flags": 1,
    "table_id": 0,
    "match": {
        "dl_dst": "00:00:00:00:00:01",
        "in_port": 3
    }
},
```

```
{
    "actions": [
        "OUTPUT:3"
    ],
    "idle_timeout": 0,
    "cookie": 0,
    "packet_count": 6,
    "hard_timeout": 0,
    "byte_count": 420,
    "duration_sec": 12555,
    "duration_nsec": 570000000,
    "priority": 32768,
    "length": 96,
    "flags": 1,
    "table_id": 0,
    "match": {
        "dl_dst": "00:00:00:00:00:03",
        "in_port": 2
    }
},
{
    "actions": [
        "OUTPUT:3"
    ],
    "idle_timeout": 0,
    "cookie": 0,
    "packet_count": 6,
    "hard_timeout": 0,
    "byte_count": 420,
    "duration_sec": 12555,
    "duration_nsec": 599000000,
    "priority": 32768,
    "length": 96,
    "flags": 1,
    "table_id": 0,
    "match": {
        "dl_dst": "00:00:00:00:00:03",
        "in_port": 1
    }
},
```

```
        {
            "actions": [
                "OUTPUT:2"
            ],
            "idle_timeout": 0,
            "cookie": 0,
            "packet_count": 18,
            "hard_timeout": 0,
            "byte_count": 1428,
            "duration_sec": 14119,
            "duration_nsec": 641000000,
            "priority": 32768,
            "length": 96,
            "flags": 1,
            "table_id": 0,
            "match": {
                "dl_dst": "00:00:00:00:00:02",
                "in_port": 1
            }
        }
    ]
}
```

(Β) Να εγκατασταθεί κανόνας που να απαγορεύει την επικοινωνία μεταξύ των h1και h3 για 30 δευτερόλεπτα ανεξάρτητα από το αν υπάρχει ανταλλαγή πακέτων μεταξύ τους (**Προσοχή** στα πεδία priority, timeouts, πεδίο action που θα χρησιμοποιήσετε).

When two rules match the same flow, only the one with highest priority is applied.

Ο κανόνας που εγκαθιστούμε κάνει drop τα πακέτα από τον h1 στον h3 και άρα καθιστά αδύνατη την επικοινωνία, η οποία σε κάθε περίπτωση απαιτεί queries και responsies και από τις δύο πλευρές.

```
curl -X POST -d '{
        "dpid": 1,
        "cookie": 1,
        "cookie_mask": 1,
        "table_id": 0,
        "idle_timeout": 30,
        "hard_timeout": 30,
        "priority": 32769,
        "flags": 1,
        "match":{
                "in_port":1
        },
        "match": {
                "in_port": 1,    # input port
                "dl_src": "00:00:00:00:00:01",  # src MAC
                "dl_dst": "00:00:00:00:00:03"   # dst MAC
        },
        "actions": []
 }' http://localhost:8080/stats/flowentry/add
```

Η default action είναι “DROP”

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

```
$ curl -sX POST \
     http://localhost:8080/stats/flowentry/add -d \
     @rule.json
```

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X
h2 -> h1 h3
h3 -> X h2
*** Results: 33% dropped (4/6 received)
```