
ΔΙΑΧΕΙΡΗΣΗ ΔΙΚΤΥΩΝ – ΤΕΛΙΚΗ ΕΡΓΑΣΙΑ

*ΠΡΟΣΟΜΟΙΩΣΗ ΔΙΚΤΥΟΥ LAN, ΠΡΟΣΘΗΚΗ FIREWALL ΚΑΙ ΕΥΡΕΣΗ ΒΕΛΤΙΣΤΩΝ
ΔΙΑΔΡΟΜΩΝ*

ΟΜΑΔΑ:

ΒΛΑΧΟΥ ΜΕΛΙΝΑ-ΜΑΡΙΑ,

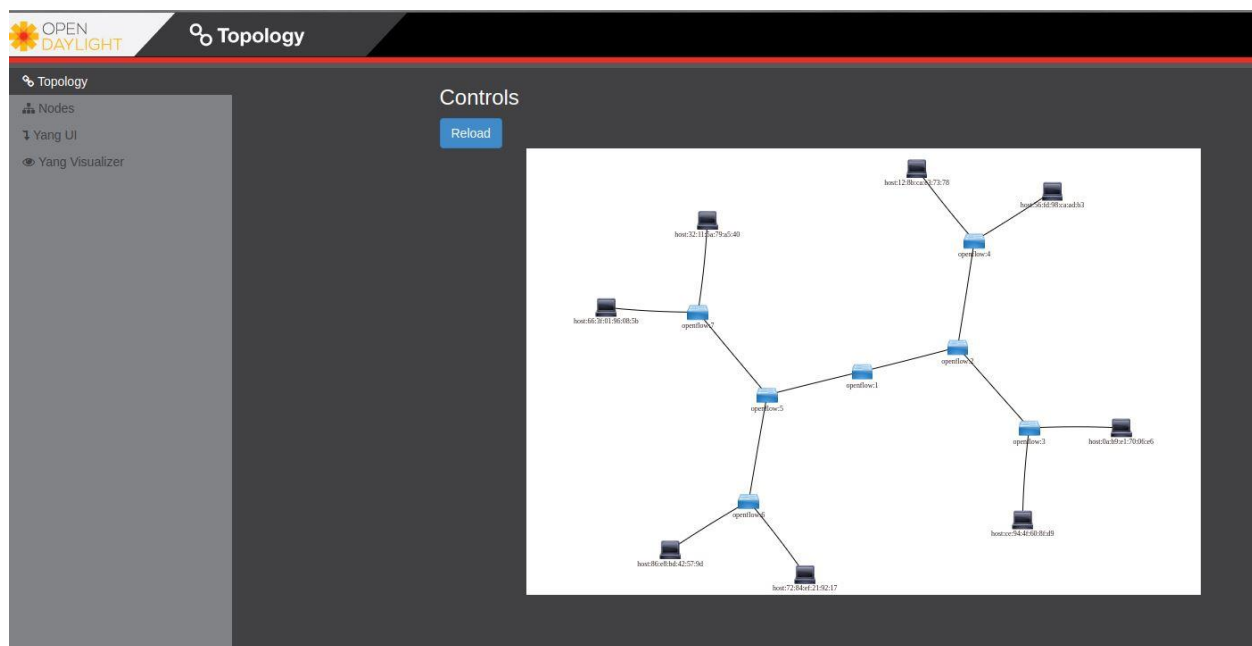
ΚΑΡΑΝΙΚΟΛΑΣ ΘΑΝΑΣΗΣ,

ΤΑΒΟΥΛΑΡΗΣ ΦΩΤΗΣ-ΕΚΤΟΡΑΣ

Εισαγωγή:

Στην εργασία αυτή υλοποιείται η προσωμοίωση ενός δικτύου LAN το οποίο αποτελείται από εικονικούς hosts και switches. Οι τοπολογίες του δικτύου μας δημιουργούνται στα αρχεία topology.py και topology2.py μέσω του εργαλείου mininet και οπτικοποιούνται μέσω του εργαλείου OpenDayLight. Υπάρχουν δυο τοπολογίες, μια για το κάθε application που υλοποιήθηκε. Τα δυο applications είναι τα Firewall και εύρεση βελτιστής διαδρομής με τον αλγόριθμο Bellman Ford όπου αναλύονται παρακάτω.

- Η 1^η τοπολογία (topology.py) στην οποία εφαρμόζεται το Firewall αποτελείται από 8 κομβούς (hosts) που ο καθένας έχει την δική του IP της μορφής '10.0.0.X' και από 7 μεταγωγείς (switches) οι οποίοι οργανώνονται σε 2 layers. Αν ανοίξουμε το OpenDayLight (<http://localhost:8181/index.html#/topology>) αφού έχουμε τρέξει το topology.py θα δούμε το παρακάτω:



- Η 2^η (topology2.py) αποτελείται από 2 hosts και 7 switches.

Οδηγίες για εκτέλεση:

Απαραίτητη προϋποθεση είναι να υπάρχουν εγκατεστημένα τα Mininet, OpenDayLight, Python 2.7 τα οποία εγκαταστήσαμε σύμφωνα με τις οδηγίες στις διαφάνειες.

Για την 1^η τοπολογία και το firewall:

- To start the firewall: `./run.sh`
- To create the topology: `sudo python topology.py`
- To start OpenDayLight: `./bin/karaf -of13` (inside distribution-karaf-0.5.4-Boron-SR4/)

Για την 2^η τοπολογία και το Bellman Ford:

- To start the Bellman Ford algorithm: `./bfrun.sh`
- To create the topology: `sudo python topology2.py`
- To start OpenDayLight: `./bin/karaf -of13` (inside distribution-karaf-0.5.4-Boron-SR4/)

Εαν επιθυμείται να ξανατρεξει καποια τοπολογια προτεινουμε ενδιαμεσα :

- `./clean.sh`

Εχοντας ξεκινήσει τις δυο τοπολογίες για να ελέγξουμε ότι λειτουργούν μπορούμε να κάνουμε τα εξής:

mininet> pingall (διαπιστώνουμε ποιο host επικοινωνεί με ποιο)

mininet> h1 ping h2 (διαπιστώνουμε αν το h1 μπορεί να στείλει πακέτα στο h2 για παραδειγμα, αντιστοίχα και για άλλα hosts)

- Στην περίπτωση του firewall θα δούμε ότι (σύμφωνα με κανόνες που έχουμε ορίσει) κάποια hosts δεν επικοινωνούν με κάποια άλλα γιατί θεωρούνται ως κακοβούλα και δημιουργείται αμοιβαίος αποκλεισμός τους. Οποτε με την πρώτη εντολή θα βλέπαμε ένα X σε αυτά τα hosts και με την δεύτερη εφόσον το h1 με h2 είναι αποκλεισμένα, θα βλέπαμε ότι δεν παραδωθήκε κανένα πακέτο. Όλα αυτά παραθετονται με φωτογραφίες παρακατω.
- Στην περίπτωση του Bellman Ford θα αρχίσουμε να βλέπουμε πακέτα απ το ένα host στο άλλο και θα εκτυπωθεί η βελτιστη διαδρομή τόσο απο το 1 στο 2 όσο και απο το 2 στο 1

Αυτο που επιτυγχανουμε ειναι σε καθε περιπτωση οσοι hosts επιθυμουμε να μπορουν να στελνουν πακετα μηνυματων μεταξυ τους με την βοηθεια των μεταγωγων που βρισκονται αναμεσα τους και παιρνουν το μηνυμα απο τη θυρα εξοδου του αποστοlea και το μεταφερουν στην θυρα εισοδου του παραληπτη

APPLICATION 1- FIREWALL

Το firewall ειναι ενα τοιχος προστασιας που υπαρχει σε καθε host ωστε να προστατευει τον «υπολογιστη» απο κακοβουλους χρηστες. Η δουλεια του ειναι με προκαθορισμενα κριτηρια να επιτρεπει η να απορριπτει(μπλοκαρει) χρηστες που προσπαθουν να επικοινωνησουν με τον καθε host. Ολες οι πληροφοριες (μηνυματα) που εισερχονται η εξερχονται απο ενα δικτυο περνουν μεσα απο αυτο το τοιχος προστασιας.

Στη συνεχεια βλεπουμε πως το δικο μας firewall εχει αποτρεψει την επικοινωνια καποιων hosts συμφωνα με τους κανονες που εχουμε θεσει(οι οποιοι λαμβανονται απ το firewall μεσω του firewall_rules.csv):

1	10.0.0.1	10.0.0.2
2	10.0.0.2	10.0.0.4
3	10.0.0.5	10.0.0.8
4	10.0.0.6	10.0.0.7

Μετα την εκτελεση του run.sh οι κανονες αυτοι φαινονται ως εξης:

```
INFO:openflow.of_01:[00-00-00-00-00-01 5] connected
DEBUG:forwarding.l2_learning:Connection [00-00-00-00-00-01 5]
INFO:misc.firewall:Adding rule: source 10.0.0.1 - destination 10.0.0.2
INFO:misc.firewall:Adding rule: source 10.0.0.2 - destination 10.0.0.4
INFO:misc.firewall:Adding rule: source 10.0.0.5 - destination 10.0.0.8
INFO:misc.firewall:Adding rule: source 10.0.0.6 - destination 10.0.0.7
```

Εχουμε αποτρεψει την επικοινωνια δηλαδη μεταξυ: **h1-h2, h2-h4, h5-h8, h6-h7**

Κανοντας pingall εχουμε:

```
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(h1, s3) (h2, s3) (h3, s4) (h4, s4) (h5, s6) (h6, s6) (h7, s7) (h8, s7) (s1, s2) (s1, s5) (s2, s3) (s2, s4) (s5, s6) (s5, s7)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*Help* Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 h4 h5 h6 h7 h8
h2 -> X h3 X h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 X h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 X
h6 -> h1 h2 h3 h4 h5 X h8
h7 -> h1 h2 h3 h4 h5 X h8
h8 -> h1 h2 h3 h4 X h6 h7
*** Results: 14% dropped (48/56 received)
mininet>
```

οπου παρατηρουμε στα αποκλεισμενα hosts ενα X στη θεση τους αντιστοιχα. Αν στη συνεχεια δοκιμασουμε να κανουμε ping στα 1 και 2 στελνοντας 3 πακετα θα δουμε:

```
mininet> h1 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2053ms
```

Οτι δηλαδη κανενα πακετο δε μπορεσε να φτασει απο το 1 στο 2 οπως και θα επρεπε. Συμπεραινουμε λοιπον τη σωστη λειτουργια του firewall.

[APPLICATION 2- SHORTEST PATH WITH BELLMAN FORD](#)

Η εφαρμογη αυτη βρισκε την συντομοτερη διαδρομη απο εναν host1 σε εναν host2 μεταξυ των μεταγωγων που παρεμβалονται αναμεσα τους συμφωνα με τον αλγοριθμο Bellman Ford και σε καθε βημα αποθηκευει τα switches και ports τους ωστε να βρει το επιθυμητο path.

Αρχικά, κάνοντας pingall επιβεβαιώνουμε ότι η επικοινωνία έχει πετύχει:

```
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(h1, s1) (s1, s2) (s1, s3) (s2, s4) (s2, s5) (s3, s6) (s3, s7) (s7, h2)
(s7, s5)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

Μετα την εκτέλεση του bellmanFord.py βλέπουμε στην οθόνη τη συντομότερη διαδρομή όπως φαίνεται παρακάτω:

```
~~~~~
[01 - 07]
Path:
[00-00-00-00-00-01, 00-00-00-00-00-03, 00-00-00-00-00-07]
~~~~~
```

Εδώ λοιπόν βλέπουμε ότι η διαδρομή που επέλεξε ο αλγόριθμος είναι η **s1-s3-s7**

Όπου σύμφωνα με το διαγράμμα της τοπολογίας που έχουμε φτιάξει είναι σωστό:

