Εργασία Διαχείρισης Δικτύων
Ετος 2020

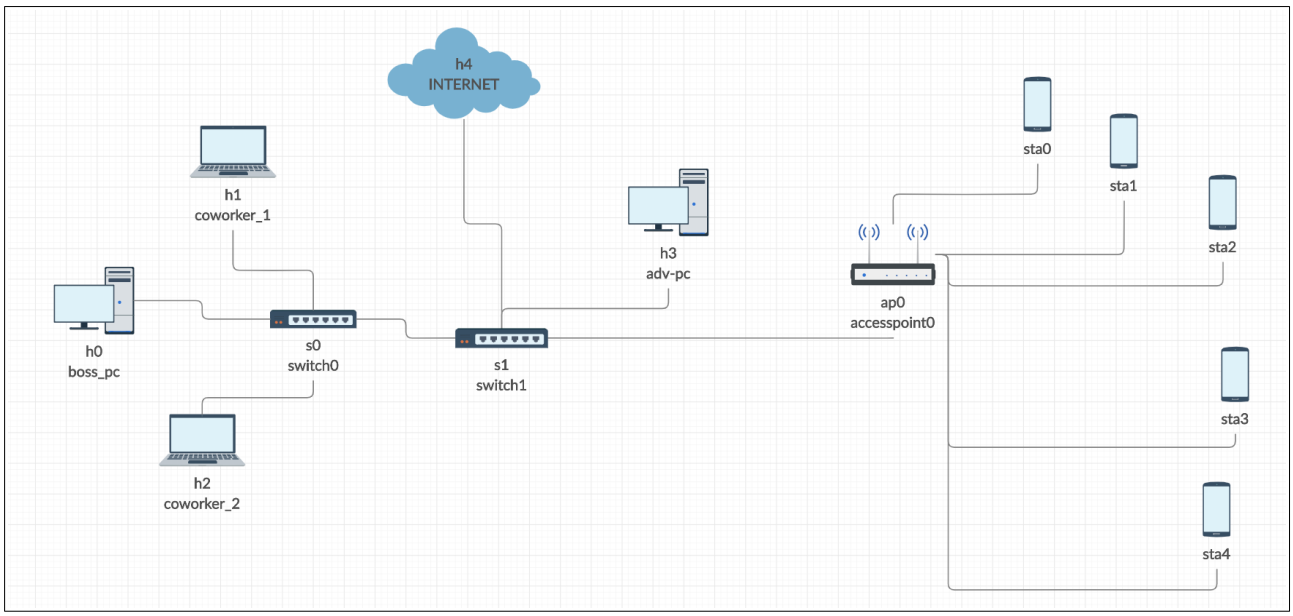
Τμήμα Πληροφορικής και Τηλεπικοινωνιών ΕΚΠΑ

Network Firewall and Failover ODL-Applications

Ομάδα:

1. Κονδύλης Γιάννης 1115201600072
2. Καραθάνος Αλέξανδρος 1115201400326
3. Εξοχίδης Γιώργος 1115201100069

1. Περιγραφή Ιδέας και Τοπολογίας



Η καφετέρια ΤαδεCafe έχει ένα δίκτυο από συσκευές

- → Οι συσκευές **host[h1, h2]** αποτελούν τους **υπολογιστές** των **υπαλλήλων** και η **h0** αυτόν του **αφεντικού** τους οι οποίες συνδέονται στο **switch s0**
- → Η συσκευή **host h4** πρόκειται για μια **αναπαράσταση** του **Διαδικτύου**
- → Η συσκευή **host h3** πρόκειται για έναν **Server** ο οποίος είναι υπεύθυνος για διαφημίσεις, back-ups, κλπ
- → Οι συσκευές **hosts[h3, h4]** συνδέονται μεταξύ τους μέσω του **switch s1** το οποίο συνδέεται με το **switch s0**
- → Οι **εικονικοί πελάτες** είναι συνδεδεμένοι στο δίκτυο μέσω των **κινητών τους συνσκευών stations [sta0, sta1, sta2, sta3, sta4]**
- → Οι **εικονικοί πελάτες** συνδέονται στο δίκτυο χρησιμοποιώντας το **access point ap0** , το οποίο αντιμετωπίζουμε σαν **public Wifi**, που συνδέεται στο **switch s1**

1. Οι **hosts[h0,h1,h2]** αποτελούν τους **businessHosts**
2. Οι **hosts[h0,h1,h2]** και οι **stations[sta0,sta1,sta2,sta3,sta4]** αποτελούν τους **clients**
3. Ο **host h3** είναι ο **server** της επιχείρησης

>Η εφαρμογή μας, υλοποιεί ενα FirewallServer με την χρήση του OpenDayLight ώστε να επιτρέπονται ή να απαγορεύονται συγκεκριμένες συνδέσεις

>Μέσω του Mininet-Wifi υλοποιούμε την Τοπολογία και εκτελούμε τα tests

Hosts' IPs		Hosts' MACs
h0	10.0.0.1	00:00:00:00:00:01
h1	10.0.0.2	00:00:00:00:00:02
h2	10.0.0.3	00:00:00:00:00:03
h3	10.0.0.4	00:00:00:00:00:04
h4	10.0.0.5	00:00:00:00:00:05

Stations' IPs		Stations' MACs
sta0	10.0.1.1	00:00:00:00:00:11
sta1	10.0.1.2	00:00:00:00:00:12
sta2	10.0.1.3	00:00:00:00:00:13
sta3	10.0.1.4	00:00:00:00:00:14
sta4	10.0.1.5	00:00:00:00:00:15

Switch' IPs		Switch' MACs	Switch' Dpid
s0	10.0.2.1	00:00:00:00:00:21	0000000000000001->1
s1	10.0.2.2	00:00:00:00:00:22	0000000000000002->2

Switch' IPs		Switch' MACs	Switch' Dpid
ap0	10.0.3.1	00:00:00:00:00:31	0000000000000065->101

>Να σημειωθεί ότι οι πίνακες **hosts,stations,switch,accesspoint** και όποιοι άλλοι προκείπτουν απο αυτούς ειναι **zerobased** όπως όλοι οι πίνακες στην Python ενώ οι **Ips,MACs,Dpid** είναι **onebased** για τεχνικούς λόγους IPs που τελειώνουν σε 0 είναι για broadcast

2. Εγκαταστάσεις

>Κατεβάσαμε το VM με το mininet-wifi προεγκατεστιμένο σε lubuntu
https://github.com/intrig-unicamp/mininet-wifi

>Εγκαταστήσαμε το OpenDaylight controller βαση των βημάτων του eclass

> Εγκαταστήσαμε VS code από όπου γραψαμε και τον κώδικα και εκτελούσαμε τα προγράμματα

>Εγκαταστήσαμε pip,python κλπ βαση του eclass

>Εγκατάσταση του postman για τα jason αρχεια τα request/get/post

3. Εκτέλεση

>Σε terminal (με το οποίο δεν θα ξανασχοληθούμε πέρα από το Ctrl+D για να το κλείσουμε στο τέλος) …./distribution-karaf-0.5.4-Boron-SR4/bin/karaf -of13 για να ανοίξει το OpenDaylight

>Σε άλλος terminal πλοηγούμαστε στον φάκελο .../network management/project1/mininet_topo και τρέχουμε την εντολή:

-sudo python topology.py

-sudo python topology.py --test (για να μπορέσουμε να εκτελέσουμε τα προκατ test)

+τρέχουμε τα τεστ πληκτρολογώντας το εκάστοτε νούμερο

+με exit ή e πηγαίνουμε στο CLI του mininet (όπου μπορούμε να κανουμε pingall για να εμφανιστεί πληρως η τοπολογια στο http://localhost:8181/index.html#/topology

και με Ctrl+D τερματίζουμε από το CLI

+ με c μπορούμε να εναλλάσουμε testmenu/CLI αν και εφόσον έχουμε καλέσει το πρόγραμμα με το όρισμα --test και δεν έχουμε κάνει exit

```
*** Starting network
+=============================================================================================+
|1)#ICMP BUSINESS<->BUSINESS    11)#TCP BUSINESS<->CUSTOMERS    21)#TCP CUSTOMERS<->INTERNET|
|2)#ICMP CUSTOMERS<->CUSTOMERS  12)#TCP BUSINESS<->SERVER       22)#TCP SERVER<->INTERNET   |
|3)#ICMP BUSINESS<->CUSTOMERS   13)#TCP CUSTOMERS<->SERVER      23)# CUSTOMERS<->SERVER     |
|4)#ICMP BUSINESS<->SERVER      14)#UDP BUSINESS<->BUSINESS     24)# BUSINESS<->BUSINESS    |
|5)#ICMP CUSTOMERS<->SERVER     15)#UDP CUSTOMERS<->CUSTOMERS                               |
|6)#HTTP BUSINESS<->INTERNET    16)#UDP BUSINESS<->CUSTOMERS                                |
|7)#HTTP CUSTOMERS<->INTERNET   17)#UDP BUSINESS<->SERVER                                   |
|8)#HTTP SERVER<->INTERNET      18)#UDP CUSTOMERS<->SERVER                                  |
|9)#TCP BUSINESS<->BUSINESS     19)#ICMP BUSINESS<->INTERNET                                |
|10)#TCP CUSTOMERS<->CUSTOMERS  20)#TCP BUSINESS<->INTERNET                                 |
+---------------------------------------------------------------------------------------------+
|   >Press the number of the test you want to examine (ex. 13 + ENTER )                      |
|   >Type exit to close this menu and open the mininet-wifi command line                     |
+=============================================================================================+
```

>Σε άλλος terminal πλοηγούμαστε στον φάκελο .../network management/project1/odl_controll και τρέχουμε την εντολή:

-python app.py

+με on/off και το νούμερο του flow το ενεργοποιούμε ή το απενεργοποιούμε
+με clear γίνονται diactivated ολα τα flows
+ενω με exit τερματίζει

```
Initializing application with arguments:
        Username: admin
        Password: admin
        Server socket: 127.0.0.1:8181

Initializing server with ip:127.0.0.1 and port:8181
+=========================================================================+
|1)#BLOCK ICMP - SWITCH 1                       7)#BLOCK CUSTOMERS<->SERVER     |
|2)#BLOCK TCP BUSINESS<->INTERNET - SWITCH 1    8)#BLOCK BUSINESS<->BUSINESS    |
|3)#BLOCK TCP CUSTOMERS<->INTERNET - SWITCH 2   9)                              |
|4)#BLOCK TCP SERVER<->INTERNET - SWITCH 2      10)                             |
|5)#BLOCK ICMP BUSINESS<->CUSTOMERS- SWITCH 2   11)                             |
|6)#BLOCK UDP BUSINESS<->CUSTOMERS- SWITCH 2    12)                             |
+-------------------------------------------------------------------------+
|   >Press the "on" or "off" + the number of the flow to activate/diactivate    |
|   >Type "exit" to close the app "clear" to diactivate all flows               |
+=========================================================================+
Type your command:
```

4. Some back-end

Τα τεστ πραγματοποιούνται με συνδιασμούς βασικών εντολών των Linux

Protocol	Command	Keyword to find
ICMP	ping -c 2	time
HTTP	python -m SimpleHTTPServer curl-m 2	<html>
TCP	iperf -s -p 5566 -i 1 iperf -c-t 1 -p 5566	connected
UDP	iperf -s -p 5566 -u -i 1 iperf -c -u -t 2 -p 5566	connected

Τα on/off flows υλοποιούνται μέσω των συναρτησεων addBlockAction και removeAction

Η addBlockAction κατασκευάζει ένα json μηνυμα που αποστέλεται στον controller στο αντίστοιχο endpoint και url για να μπλοκάρει την αντίστοιχη σύνδεση.
(έχει υλοποιηθεί και η AddAllowAction αλλά δεν χρησιμοποιήθηκε στα flows)

```
def addBlockAction(switch, table, flow_id, protocol = None, ip4_src = None,
ip4_dest = None, tcp_src_port = None, tcp_dest_port = None, udp_src_port =
None, udp_dest_port = None, mac_src = None, mac_dest = None, in_port = None,
out_port = None):
```

switch= σε ποιό switch θα επιδρασει ο κανόνας
table=σε ποιό switch θα βρίσκεται ο κανόνας
flow_id=το διακρητικό του κανόνα
protocol=ποιό πρωτόκολλο θα μπλοκαριστεί
ip4_src=η IP του αποστολέα-πηγή
ip4_dest=η IP του παραλήπτη-προορισμού
tcp_src_port=η TCP port του αποστολέα-πηγή
tcp_dest_port=η TCP port του παραλήπτη-προορισμού
udp_src_port=η UDP port του αποστολέα-πηγή
udp_dest_port=η UDP port του παραλήπτη-προορισμού
mac_src=η MAC του αποστολέα-πηγή
mac_dest=η MAC του παραλήπτη-προορισμού
in_port=η θύρα-interface εισόδου στο switch
out_port=η θύρα-interface εξόδου στο switch

5. ScreenShots and Testing

ScreenShots απο τα αποτελέσματα των εκτελέσεων των τεστ που ελέγχουν τους κανόνες θα βρειτε στον αντίστοιχο φάκελο **flows-test/flow_κ**

Δομή φακέλων:

OFF_flow_κ_test_λ.png	ScreenShots των test λ οταν το flow κ είναι απενεργοποιημένο
ON_flow_κ_test_λ.png	ScreenShots των test λ οταν το flow κ είναι ενεργοποιημένο
ON_flow_κ_command.png	ScreenShots της εντολής που ενεργοποιεί το flow κ

1ος κανόνας-flow:
Block all ICMP on switch 1
tests 1,3,4,19

2ος κανόνας-flow:
Block TCP(blocking TCP blocks HTTP) between Business hosts and Host 4(Internet) on switch 1

test 6,20

3ος κανόνας-flow:
Block TCP(blocking TCP blocks HTTP) betweem Customers stations and Host 4(Internet) on switch 2
test 7,21

4ος κανόνας-flow:
Block TCP(blocking TCP blocks HTTP) between Server stations and Host 4(Internet) on switch 2
test 8,22

5ος κανόνας-flow:
Block ICMP only between Business hosts and Customers stations on switch 2
test 3

6ος κανόνας-flow:
Block UDP only between Business hosts and Customers stations on switch 2
test 16

7ος κανόνας-flow:
Block any connection between Customers stations and Server on switch 2
test 23

8ος κανόνας-flow:
Block any connection between Business hosts on switch 1
test 24

Bug: η εντολή κόμβος.cmd(string) φαίνεται να μην αναγνωρίζει πάντα ολόκληρα τα string ή όλα τα ορίσματα. Αυτό έχει ως αποτέλεσμα όταν κάποια συνδεση γίνεται block η εντολή που την ελέγχει παρόλο που έχει timeout όρισμα να μην το διαβάζει και να παίρνει το default timeout
όταν τα flow μπλοκάρουν μια σύνδεση το testing για την συνδεση 1-1 θελει μερικά (2 περίπου) λεπτά

1. Περιγραφή Ιδέας και Τοπολογίας

Στη 2η εφαρμογή προσπαθήσαμε να υλοποιήσουμε την ιδέα του failover :

→ Οι συσκευές **host[h1, h2,h3,h4]** αποτελούν τους 3 servers και 1 client που συνδέονται σε ένα **switch s0**
→ Η συσκευή **host h1** πρόκειται για ένα node που θα στέλνει μηνύματα στους servers.
→ Οι συσκευές **host h2,h3,h4** είναι οι **Servers** υλοποιημένοι σαν webservers -http- .
→ Όλες οι συσκευές συνδέονται στο **switch s0**
→ Ο **h1** με curl στους servers επιβεβαιώνει ότι όλα λειτουργούν σωστά και υπάρχει μεταξύ όλων επικοινωνία με τα αντίστοιχα get, όπου σε κάθε server είναι διαφορετικό ώστε να είναι ευδιάκριτο στο debugging.

Τέλος το switch μέσω του odl κάνει τα αντίστοιχα redirects στους server που είναι online.

2. Εκτέλεση

Τρέχουμε αντίστοιχα τις εντολές:

-Σε terminal (με το οποίο δεν θα ξανασχοληθούμε πέρα από το Ctrl+D για να το κλείσουμε στο τέλος) …./distribution-karaf-0.5.4-Boron-SR4/bin/karaf -of13 για να ανοίξει το OpenDaylight

Σε άλλο terminal πλοηγούμαστε στον φάκελο .../network management/project2/mininet_topo και τρέχουμε την εντολή:
-sudo python topology.py
-python app.py

Στο μενού που εμφανίζεται έχουμε τις εξής λειτουργίες :

1. Start server 2
2. Start server 3
3. Start server 4
4. Stop server 2
5. Stop server 3
6. Stop server 4
7. Curl to server 2
8. Curl to server 3
9. Curl to server 4
10. Curl to cluster

Όπου cluster είναι μια εικονική ip με σκοπό τη λειτουργία του αυτόματου failover -αντί μέσω επιλογής- μέσω NAT όπου μηνύματα από το 10.0.1.1 θα γίνονται translate σε 10.0.1.2/3/4 και αντίστροφα, αλλά δεν προλάβαμε να υλοποιήσουμε σωστά την αντίστροφη επικοινωνία.