ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μελέτη της αρχιτεκτονικής OpenDaylight Software Defined Networking και υλοποίηση παραμετροποίησης δικτυακών υπηρεσιών SDN με βάση την πλατφόρμα OpenDaylight

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΤΡΟΠΑΪΤΗ ΙΩΑΝΝΗ
(ΑΕΜ: 2528)

Επιβλέπων: **Νικολάου Σπυρίδων**
Λέκτορας

Καστοριά Δεκέμβριος - 2021

Η παρούσα σελίδα σκοπίμως παραμένει λευκή

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μελέτη της αρχιτεκτονικής OpenDaylight Software Defined Networking και υλοποίηση παραμετροποίησης δικτυακών υπηρεσιών SDN με βάση την πλατφόρμα OpenDaylight

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΤΡΟΠΑΪΤΗ ΙΩΑΝΝΗ

(ΑΕΜ: 2528)

Επιβλέπων: **Νικολάου Σπυρίδων**

Λέκτορας

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ημερομηνία εξέτασης

.....	.....	.....
Ον/μο Μέλους	Ον/μο Μέλους	Ον/μο Μέλους
Ιδιότητα Μέλους	Ιδιότητα Μέλους	Ιδιότητα Μέλους

Καστοριά Δεκέμβριος - 2021

Ευχαριστίες

Φτάνοντας στο τέλος των προπτυχιακών μου σπουδών στο Τμήμα Πληροφορικής του Πανεπιστήμιου Δυτικής Μακεδονίας, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κύριο Σπυρίδων Νικολάου, τόσο για την ανάθεση αυτής της πτυχιακής εργασίας η οποία με βοήθησε να εντρυφήσω ένα ενδιαφέρον σύγχρονο επιστημονικό πεδίο και να εμπλουτίσω περαιτέρω τις γνώσεις μου, όσο και για την συνεχή καθοδήγηση του καθ' όλη τη διάρκεια της εκπόνησής της.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου διότι χωρίς την βοήθεια και την στήριξή τους δεν θα είχα την δυνατότητα να ολοκληρώσω τις σπουδές μου.

Περίληψη

Τα δίκτυα καθοριζόμενα από λογισμικό (Software Defined Networks - SDNs), αποτελούν σημαντικό ερευνητικό πεδίο παγκοσμίως, τόσο για την ακαδημαϊκή-ερευνητική κοινότητα, αλλά και τη βιομηχανία ΤΠΕ, με στόχο τη μοντελοποίηση και ανάπτυξη σύγχρονων δικτύων ανεξάρτητης αρχιτεκτονικής δομής, που θα προσφέρουν υψηλότερη αποδοτικότητα, μεγαλύτερα ευελιξία και καλύτερη διαχείριση. Ήδη καινοτόμες ιδέες και προτάσεις σχετικών ερευνητικών έργων έχουν βρει πεδίο εφαρμογής και προσφέρουν αποτελεσματικές λύσεις. Λόγω της πολυπλοκότητας της αρχιτεκτονικής της παραδοσιακής δικτυακής υποδομής που διαθέτουν οι πάροχοι υπηρεσιών και τηλεπικοινωνιών, καθιστούν την υιοθέτηση των σύγχρονων και καινοτόμων τεχνολογιών δικτύωσης αργή και κοστοβόρα, με αποτέλεσμα να μην μπορεί να καλύψουν άμεσα τις ολοένα αυξανόμενες ανάγκες, ιδιαίτερα εν' όψει της εξάπλωσης του Διαδικτύου των Πραγμάτων (Internet of Things – IoT) και των εφαρμογών του σε όλους τους τομείς της καθημερινότητας.

Στόχος της εργασίας αυτής είναι η εκτεταμένη μελέτη των δικτύων που καθορίζονται από λογισμικό (SDN) με έμφαση στις δυνατότητες που παρέχουν τα πρωτόκολλα OpenFlow και ο ελεγκτής OpenDaylight. Επιπλέον παρουσιάζεται, βήμα προς βήμα, η διαδικασία ανάπτυξης τριών χαρακτηριστικών σεναρίων δικτύων καθοριζόμενων από λογισμικό, με τη χρήση του ελεγκτή OpenDaylight, σε περιβάλλον προσομοίωσης, με τη βοήθεια του εξομοιωτή Mininet, καθώς και η ανάλυση και αξιολόγηση των στατιστικών στοιχείων που προκύπτουν από τη μελέτη των σεναρίων αυτών.

Λέξεις Κλειδιά: Δίκτυα καθοριζόμενα από λογισμικό (SDN), Εικονικοποίηση Λειτουργιών Δικτύου (NFV), OpenDaylight, OpenFlow, ελεγκτής, μεταγωγέας, Mininet

Abstract

Software Defined Networks (SDNs) are an important research field, both for the academic-research community and the ICT industry, with the aim of modeling and developing systems of independent architectural structure, which will offer higher efficiency, greater flexibility, and better management. Innovative ideas and related proposals for research projects have already been tested and applied, offering effective and efficient solutions. Due to the complexity of the architecture of the traditional network infrastructure available to telecom service providers and organizations, they make the adoption of modern and innovative networking technologies slow and costly, and thus they cannot meet the growing needs, especially in view of Internet of Things (IoT) and its applications in all aspects of everyday life.

The aim of this work is the extensive study of Software-Defined Networks with emphasis on the capabilities provided by OpenFlow protocols and the OpenDaylight controller. In addition, the process of developing three characteristic SDN scenarios based on the OpenDaylight controller, in a simulation environment with the help of the Mininet emulator, is presented step by step, as well as the analysis and evaluation of statistical data derived from the study of these scenarios.

Keywords: Software Defined Network (SDN), Network Function Virtualization (NFV), OpenFlow, OpenDaylight, Controller, Switch, Mininet

Πίνακας Περιεχομένων

Εισαγωγή 1

1. Δικτύωση Software Defined Network 3

1.1 Ορισμός και βασικές αρχές του SDN 3

1.2 Αρχιτεκτονική SDN 4

1.2.1 Ανάλυση δομικών στοιχείων SDN 5

1.3 SDN Controllers 6

1.3.1 Τι είναι SDN Controllers 7

1.3.2 Είδη SDN Controllers 7

1.4 Πλεονεκτήματα χρήσης SDN 9

1.5 Εικονικοποίηση λειτουργιών δικτύου (Network Function Virtualization) 11

1.5.1 Τι είναι Network Function Virtualization και πως λειτουργεί 12

1.5.2 Αρχιτεκτονική Network Function Virtualization 13

1.5.3 Πλεονεκτήματα χρήσης Network Function Virtualization 13

1.5.4 Γιατί χρησιμοποιούμε Network Function Virtualization στις επιχειρήσεις 15

1.5.5 Διαφορές μεταξύ Network Function Virtualization και SDN 15

1.6 Υπολογιστική Νέφους (Cloud Computing) 16

1.6.1 Μοντέλο ανάπτυξης και μορφές υπολογιστικού νέφους 18

1.6.2 Πλεονεκτήματα και περιορισμοί υπολογιστικού νέφους 21

2. Το Πρωτόκολλο OpenFlow 24

2.1 Αρχιτεκτονική OpenFlow 24

2.1.1 OpenFlow Controller 24

2.1.2 OpenFlow Switch 25

2.1.3 OpenFlow Channel 29

2.2 OpenFlow πίνακες 30

2.2.1 Flow Tables 31

2.2.2 Group Tables 33

2.2.3 Meter Tables 35

2.3 OpenFlow μηνύματα 36

3. Ο ελεγκτής OpenDaylight SDN Controller 39

3.1 Αρχιτεκτονική OpenDaylight 41

3.2 Σύγκριση OpenDaylight με άλλους ελεγκτές 44

3.2.1 Προετοιμασία περιβάλλοντος δοκιμών αξιολόγησης 44

3.2.2 Αποτελέσματα ανάλυσης αξιολόγησης των πειραματικών δοκιμών 46

3.3 Χαρακτηριστικά και εφαρμογές OpenDaylight 49

3.3.1 DLUX 50

3.3.2 L2 Switch 51

3.3.3 OpenFlow plugin 52
4. Υλοποίηση προσομοίωσης σε Mininet εξομοιωτή αρχιτεκτονικής OpenDaylight SDN 54
4.1 Ανάπτυξη OpenDaylight σε VirtualBox 54
4.2 Υλοποίηση περιβάλλοντος ODL Karaf Distribution 55
4.2.1 Εγκατάσταση Java 55
4.2.2 Λήψη και εγκατάσταση ODL Controller 56
4.2.3 Εγκατάσταση χαρακτηριστικών ODL 57
4.3 Επισκόπηση του Mininet 57
4.3.1 Γιατί χρησιμοποιούμε το Mininet 58
4.3.2 Πλεονεκτήματα και περιορισμοί του Mininet 58
4.4 Προετοιμασία υλοποίησης προσομοίωσης σε Mininet 59
4.4.1 Εγκατάσταση Mininet VM 59
4.4.2 Εφαρμογές και ενσωματωμένα εργαλεία του Mininet 63
4.4.3 Βασικές εντολές και σημαντικές κλάσεις για τη δημιουργία τοπολογιών στο Mininet64
5. Ανάπτυξη σεναρίων προσομοίωσης OpenDaylight SDN 72
5.1 Υλοποίηση 1ου σεναρίου προσομοίωσης σε Linear OpenDaylight SDN Topology 72
5.2 Υλοποίηση 2ου σεναρίου προσομοίωσης σε Tree OpenDaylight SDN Topology 79
5.3 Υλοποίηση 3ου σεναρίου προσομοίωσης σε Mobility OpenDaylight SDN Topology 83
5.4 Ανάλυση αποτελεσμάτων προσομοίωσης της λειτουργίας δικτύου SDN με ODL Controller 91
5.4.1 Ανάλυση αποτελεσμάτων Linear Network Topology 91
Συμπεράσματα 94
Βιβλιογραφία 100
ΠΑΡΑΡΤΗΜΑ – Α 103
Α1. Προετοιμασία εικονικής μηχανής (Virtual Machine) για εγκατάσταση ODL Controller. 103
ΠΑΡΑΡΤΗΜΑ - Β 106
Β1. Παραγωγή Python κώδικα της Tree Network Topology μέσω Miniedit 106
Β2. Python script της Mobility Topology 107

ΛΙΣΤΑ ΕΙΚΟΝΩΝ

Εικόνα 1. Σχηματική αναπαράσταση SDN αρχιτεκτονικής 4
Εικόνα 2. Σχηματική αναπαράσταση εξέλιξης των SDN Controllers..... 6
Εικόνα 3. High-level NFV framework ETSI 12
Εικόνα 4. Σχηματική αναπαράσταση σύγκρισης NFV και SDN 15
Εικόνα 5. Σχηματική αναπαράσταση λειτουργίας Υπολογιστικού Νέφους..... 17
Εικόνα 6. Σχηματική αναπαράσταση υπηρεσιών Cloud Computing 23
Εικόνα 7. Τυπική αρχιτεκτονική OpenFlow switch 25
Εικόνα 8. Ροή πακέτου μέσω της διαδικασίας της διοχέτευσης 30
Εικόνα 9. Σχηματική αναπαράσταση ροής πακέτου μέσω της διαδικασίας της διοχέτευσης33
Εικόνα 10. Σχηματική αναπαράσταση γενικής αρχιτεκτονικής ODL με δομικά στοιχεία 41
Εικόνα 11. Σχηματική αναπαράσταση διάταξης πειράματος [26] 45
Εικόνα 12. Μέσος αριθμός αποκρίσεων ανά δευτερόλεπτο κάτω από μεταβαλλόμενο αριθμό μεταγωγέων (MACs=1000, threads=4) [26] 46
Εικόνα 13. Μέση καθυστέρηση κάτω από μεταβαλλόμενο αριθμό μεταγωγέων (MACs=1000, thread=4) [26] 47
Εικόνα 14. Αριθμός αποκρίσεων ανά δευτερόλεπτο κάτω από μεταβαλλόμενο αριθμό των MACs (s=16, threads=4) [26] 47
Εικόνα 15. Μέσος όρος καθυστέρησης με μεταβαλλόμενο αριθμό των MACs (s=16, threads=4) [26] 48
Εικόνα 16. Συνολική απόδοση ελεγκτή (MACs=1000, threads=4) [26] 48
Εικόνα 17. Συνολική απόδοση ελεγκτή (s=16, threads=4) [26] 49
Εικόνα 18. ODL Carbon χαρακτηριστικά 50
Εικόνα 19. Σχηματική αναπαράσταση DLUX Module 51
Εικόνα 20. Υλοποίηση OpenFlow σε ODL ελεγκτή [31] 53
Εικόνα 21. Επαλήθευση της εγκατάστασης της προεπιλεγμένης Mininet topology [36] 65
Εικόνα 22. Δημιουργία τοπολογίας με τη χρήση του MiniEdit [36] 65
Εικόνα 23. Χρήση εντολών nodes και net 66
Εικόνα 24. Χρήση εντολής dump 66
Εικόνα 25. Χρήση εντολής ping από τον h1 στον h2 67
Εικόνα 26. Πρόσβαση στις λεπτομέρειες του host h1..... 67
Εικόνα 27. Χρήση της εντολής pingall..... 67
Εικόνα 28. Διακοπή σύνδεσης ανάμεσα στο host h1 και switch s1..... 68
Εικόνα 29. Χρήση της εντολής sudo mn για τη δημιουργία τοπολογίας 68
Εικόνα 30. Δημιουργία προκαθορισμένης τοπολογίας στο Mininet..... 69
Εικόνα 31. Δημιουργία τοπολογίας και επικύρωση ελέγχων 69

Εικόνα 32. Έλεγχος της διαθεσιμότητας του host h2 από το host h1 70
Εικόνα 33. Δημιουργία τοπολογίας εικονικού δικτύου μέσω του CLI στο Mininet..... 73
Εικόνα 34. Είσοδος στον OpenDaylight controller..... 73
Εικόνα 35. Κεντρική σελίδα OpenDaylight DLUX..... 74
Εικόνα 36. Γενική σχηματική αναπαράσταση Linear Topology 74
Εικόνα 37. Σχηματική αναπαράσταση Yang Virtualizer 75
Εικόνα 38. Σχηματική αναπαράσταση σελίδας YANG UI 76
Εικόνα 39. Κεντρική σελίδα του Miniedit 79
Εικόνα 40. Υλοποίηση Tree Topology στο Miniedit..... 80
Εικόνα 41. Σχηματική αναπαράσταση Tree Topology στο OpenDaylight DLUX..... 80
Εικόνα 42. Αποθήκευση και εξαγωγή της τοπολογίας μας σε Python Script 82
Εικόνα 43. Δημιουργία της Tree Topology με την εντολή sudo python..... 83
Εικόνα 44. Έλεγχος συνδεσιμότητας μεταξύ των hosts..... 83
Εικόνα 45. Χρήση εντολής sudo python3 για τη δημιουργία της mobility topology 87
Εικόνα 46. Σχηματική αναπαράσταση mobility τοπολογίας στο χρόνο 2 second..... 88
Εικόνα 47. Σχηματική αναπαράσταση mobility τοπολογίας στο χρόνο 12 second..... 88
Εικόνα 48. Χρήση εντολής nodes.params για συλλογή πληροφοριών 89
Εικόνα 49. Top Ports και Topology μέσω του mininet-dashboard 92
Εικόνα 50. Σχηματική αναπαράσταση τοπολογίας Linear στο mininet-dashboard 92
Εικόνα 51. Χρήση iperf tool για τον έλεγχο του bandwidth μεταξύ των κεντρικών υπολογιστών 92
Εικόνα 52. Σχηματική αναπαράσταση κίνησης ροής μεταξύ h2 και h3 93
Εικόνα 53. Σχηματική αναπαράσταση κίνησης ροής μεταξύ h1 και h2 93
Εικόνα 54. Σχηματική αναπαράσταση κίνησης ροής μεταξύ h2 και h3 93
Εικόνα 55. Σχηματική αναπαράσταση διάταξης της Linear topology μέσω Trace Flow 93
Εικόνα 56. Top Port και Topology μέσω mininet-dashboard 94
Εικόνα 57. Σχηματική αναπαράσταση τοπολογίας Tree στο mininet-dashboard 94
Εικόνα 58. Σχηματική αναπαράσταση κίνησης ροής σε όλους τους κόμβους 95
Εικόνα 59. Σχηματική αναπαράσταση διάταξης της Treer topology μέσω Trace Flow 95

ΛΙΣΤΑ ΠΙΝΑΚΩΝ

Πίνακας 1. Σύγκριση δημοφιλών SDN ελεγκτών 9

Πίνακας 2. Δομή των Flow entries σε ένα Flow Table 31

Πίνακας 3. Βασικά δομικά στοιχεία μιας καταχώρησης ομάδας σε ένα Group Table 34

Εισαγωγή

Το SDN εμφανίστηκε ως ένα εναλλακτικό παράδειγμα υλοποίησης της διαδικασίας μεταγωγής και δρομολόγησης πακέτων δεδομένων που διεξάγεται από τα παραδοσιακά δίκτυα, επιτρέποντας το διαχωρισμό του επιπέδου ελέγχου (control plane) από το επίπεδο μεταγωγής των δεδομένων (data plane), μετατρέποντας έτσι τα δίκτυα υπολογιστών σε προγραμματιζόμενα. Αυτό καθιστά πιο εύκολη τη διαχείριση και τον έλεγχο σύνθετων τοπολογιών δικτύων, που περιέχουν μεγάλο πλήθος από δρομολογητές (Routers) και μεταγωγείς (Switches). Έτσι το SDN προσφέρει στον διαχειριστή τη δυνατότητα γενικού ελέγχου του δικτύου, ρυθμίζοντας ταυτόχρονα πολλές δικτυακές συσκευές, με μεγαλύτερη ευελιξία και ασφάλεια, σε σχέση με ένα παραδοσιακό δίκτυο.

Σήμερα έχουν αναπτυχθεί αρκετοί ελεγκτές για υλοποιήσεις δικτύων SDN, έχοντας ο καθένας τα δικά του χαρακτηριστικά πλεονεκτήματα και μειονεκτήματα και διαφορετικές γλώσσες προγραμματισμού των λειτουργιών τους. Στην παρούσα εργασία χρησιμοποιήθηκε ο SDN ελεγκτής OpenDaylight σε συνδυασμό με τον εξομοιωτή Mininet αναπτύσσοντας ένα σενάριο προσομοίωσης λειτουργίας δικτύου SDN.

Η εργασία αυτή αποτελείται από πέντε κεφάλαια τα οποία είναι διαμοιρασμένα έτσι ώστε ο αναγνώστης να κατανοήσει τόσο το θεωρητικό υπόβαθρο των δικτύων SDN, όπως τις βασικές αρχές και την αρχιτεκτονική τους και στη συνέχεια να εντρυφήσει στο τεχνολογικό υπόβαθρό τους, με έμφαση στις λειτουργίες του πρωτοκόλλου OpenFlow, αλλά και των κυριότερων διαθέσιμων ελεγκτών (SDN controllers).

Στο πρώτο κεφάλαιο παρουσιάζονται οι βασικές ιδέες για το SDN καθώς και η αρχιτεκτονική του. Επίσης γίνεται αναφορά στην τεχνολογία της εικονικοποίησης λειτουργιών δικτύωσης (Network Function Virtualization – NFV) αλλά και της υπολογιστικής νέφους (Cloud computing).

Στο δεύτερο κεφάλαιο αναλύεται το πρωτόκολλο OpenFlow, τόσο από τη σκοπιά της αρχιτεκτονικής του, όσο και από τα δομικά μέρη και τις βασικές λειτουργίες του. Παρουσιάζονται τα είδη των μηνυμάτων που ανταλλάσσονται

μέσω του πρωτοκόλλου OpenFlow, καθώς και οι αντίστοιχοι Πίνακες OpenFlow.

Στο τρίτο κεφάλαιο γίνεται επισκόπηση της αρχιτεκτονικής και των χαρακτηριστικών του ελεγκτή OpenDaylight, καθώς και συγκριτική παρουσίαση με άλλους ελεγκτές SDN, όπως Ryu, ONOS και Floodlight.

Στο τέταρτο κεφάλαιο παρουσιάζεται λεπτομερώς η διαδικασία προσομοίωσης της λειτουργίας δικτύου SDN με την εγκατάσταση και ανάπτυξη του ελεγκτή OpenDaylight σε μια εικονική μηχανή (VM).

Στο πέμπτο και τελευταίο κεφάλαιο υλοποιείται το πειραματικό μέρος της εργασίας με τη βοήθεια του εξομοιωτή Mininet και την αξιοποίηση χρήσιμων εργαλείων (Miniedit, Wireshark, ovs-ofctl, ovs-vsctl, iperf), καθώς και η παρουσίαση της στατιστικής ανάλυσης των αποτελεσμάτων της προσομοίωσης.

Τέλος, καταγράφονται τα συμπεράσματα και οι προκλήσεις που προκύπτουν από τη μελέτη του θέματος της παρούσας εργασίας.

1. Δικτύωση Software Defined Network

Το υλικό (Hardware) κυριαρχούσε στο κόσμο των δικτύων μέχρις ότου να αναδυθούν τα δίκτυα καθοριζόμενα από λογισμικό. Η προέλευση του SDN προέρχεται από την ερευνητική συνεργασία μεταξύ του Stanford University και του Barkeley University of California το 2008. Τα χρόνια της ίδρυσής του, το SDN είχε εξελιχθεί ως μία αξιόπιστη δικτυακή τεχνολογία ανεπτυγμένη από προμηθευτές και οργανισμούς όπως Cisco, VMware, Juniper, Pluribus και Big Switch. Το Open Networking Foundation (ONF) ιδρύθηκε το 2011 και ασχολήθηκε τόσο με την ανάπτυξη αυτής της νέας τεχνολογίας όσο και με την προώθησή της.

1.1 Ορισμός και βασικές αρχές του SDN

Ο επίσημος ορισμός για το τί ακριβώς σημαίνει Software Defined Network σύμφωνα με τον ONF[1] όπου ορίζει το SDN ως το φυσικό διαχωρισμό του επιπέδου ελέγχου (Control Plane) από το επίπεδο προώθησης (Forward Plane), όπου το επίπεδο ελέγχου μπορεί να διαχειρίζεται πολλές συσκευές.

Το SDN είναι μια αναδυόμενη αρχιτεκτονική που είναι δυναμική, διαχειρίσιμη, αποδοτική και προσαρμόσιμη, καθιστώντας την ιδανική για τις σημερινές εφαρμογές υψηλού εύρους ζώνης. Η συγκεκριμένη αρχιτεκτονική διαχωρίζει τον έλεγχο του δικτύου από τις λειτουργίες προώθησης επιτρέποντας τον έλεγχο του δικτύου σε άμεσα προγραμματιζόμενο και τον υπάρχον εξοπλισμό να διαχωριστεί από εφαρμογές και υπηρεσίες δικτύου. Η αρχιτεκτονική SDN είναι:

- Άμεσα προγραμματιζόμενη: Ο έλεγχος του δικτύου γίνεται από απόσταση μέσω προγραμματισμού και είναι διαχωρισμένος από τις λειτουργίες προώθησης.
- Ευέλικτη: Διαχωρίζοντας τον έλεγχο του δικτύου από τις λειτουργίες προώθησης, επιτρέπει στους διαχειριστές να προσαρμόζουν δυναμικά τη ροή κίνησης σε όλο το δίκτυο ώστε να αντιμετωπίζονται άμεσα οι μεταβαλλόμενες ανάγκες.
- Προγραμματιστικά ρυθμιζόμενη: Το SDN επιτρέπει στους διαχειριστές δικτύου να ρυθμίζουν, να διαχειρίζονται, να διασφαλίζουν και να βελτιστοποιούν άμεσα

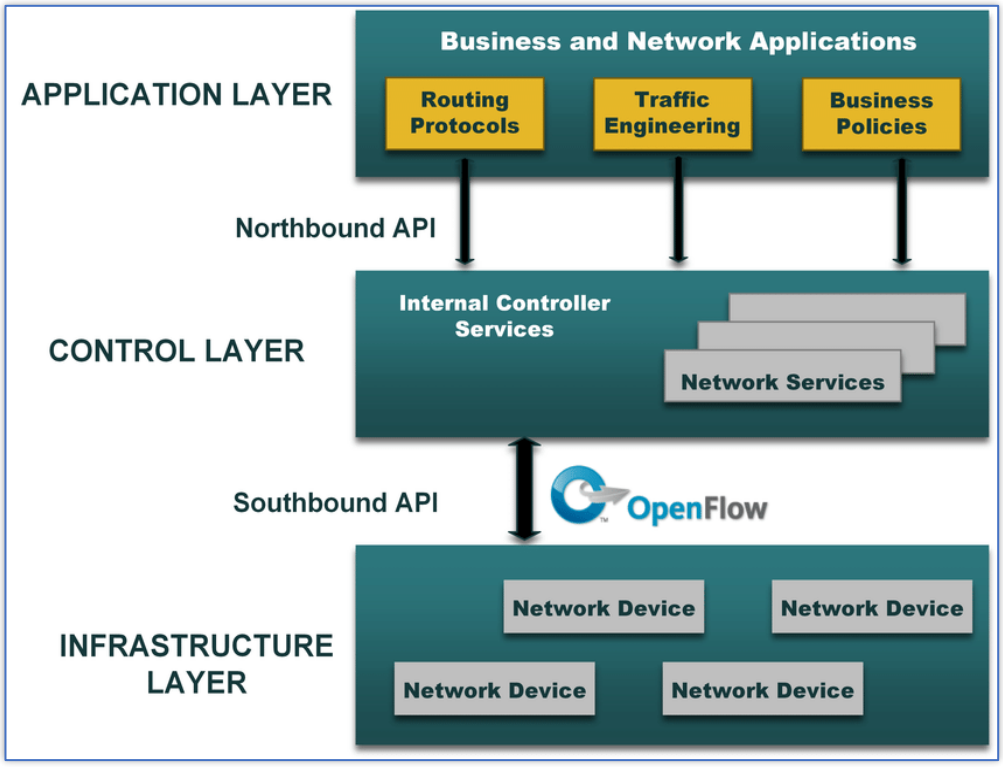
τους πόρους του δικτύου, μέσω αυτοματοποιημένων δυναμικά προγραμματιζόμενων λειτουργιών SDN, που βασίζονται σε λογισμικό ανοικτού κώδικα.

- Βασισμένη σε ουδέτερα πρότυπα και ουδέτερη σε προμηθευτές: Το SDN απλοποιεί το σχεδιασμό και τη λειτουργία του δικτύου, επειδή οι οδηγίες παρέχονται από τους SDN ελεγκτές και όχι από τις εκάστοτε συσκευές πολλαπλών προμηθευτών που απαιτούν την υποστήριξη πολλαπλών πρωτοκόλλων.

1.2 Αρχιτεκτονική SDN

Η αρχιτεκτονική ενός SDN δικτύου αποτελείται από τρία επίπεδα:

- Application layer (Επίπεδο εφαρμογών)
- Control layer (Επίπεδο ελέγχου)
- Infrastructure layer (Επίπεδο προώθησης δεδομένων)



Εικόνα 1. Σχηματική αναπαράσταση SDN αρχιτεκτονικής

Πηγή: https://www.researchgate.net/figure/Layered-architecture-of-the-SDN_fig1_311577105

1.2.1 Ανάλυση δομικών στοιχείων SDN

Application layer [2]**:** Στο επίπεδο αυτό οι SDN εφαρμογές είναι προγράμματα τα οποία επικοινωνούν με τους SDN Controllers μέσω των Application Programming Interfaces (APIs). Επίσης οι εφαρμογές μπορούν να έχουν γενική εικόνα του δικτύου συγκεντρώνοντας πληροφορίες από τον ελεγκτή για να λάβουν δραστικές αποφάσεις. Τέτοιες εφαρμογές μπορούν να συμπεριλάβουν διαχείριση δικτύου, ανάλυση στατιστικών ή εμπορικές εφαρμογές που χειρίζονται μεγάλα δεδομένα. Για παράδειγμα, μια εφαρμογή ανάλυσης στατιστικών στοιχείων θα μπορούσε να αναλάβει την αναγνώριση ύποπτης δραστηριότητας στο δίκτυο.

Control layer [2]**:** Ο SDN Controller είναι μία λογική οντότητα η οποία λαμβάνει οδηγίες από το Application layer και τις προωθεί στις δικτυακές συσκευές. Ο ελεγκτής λειτουργεί σαν γέφυρα ανάμεσα στο επίπεδο εφαρμογών και επίπεδο προώθησης δεδομένων λαμβάνοντας πληροφορίες για το δίκτυο από τις συσκευές υλικού και επικοινωνεί πίσω με τις SDN εφαρμογές έχοντας γενική εικόνα της κατάστασης του δικτύου.

Infrastructure layer [3]**:** Το επίπεδο προώθησης της κυκλοφορίας του δικτύου αποτελείται από ποικίλους εξοπλισμούς δικτύωσης όπως δικτυακούς μεταγωγείς (switches) και δρομολογητές (routers) στο κέντρο δεδομένων σχηματίζοντας έτσι το υπάρχον δίκτυο. Στο επίπεδο αυτό αναπαρίσταται η φυσική υπόσταση του δικτύου μέσω του Control layer.

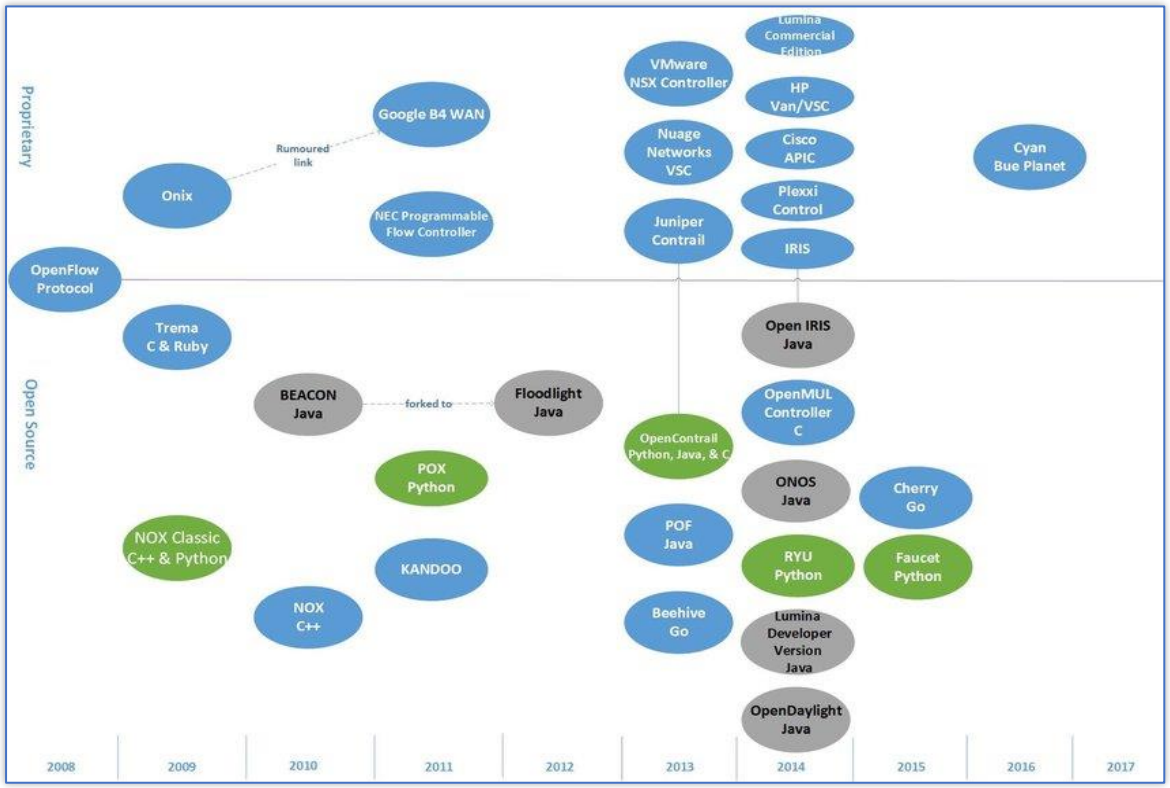
Northbound Interface [4] [5]**:** Σε μία SDN αρχιτεκτονική η βόρεια διεπαφή παρέχει επικοινωνία ανάμεσα στα υψηλότερου επιπέδου δομικά στοιχεία με τον SDN Controller. Οι εφαρμογές και οι υπηρεσίες που εκτελούνται στο δίκτυο επικοινωνούν με τον SDN Controller μέσω του Northbound Interface. Τα APIs ανάμεσα στον SDN Controller και το Application layer προσφέρουν μία πλατφόρμα που επιτρέπει στις εμπορικές εφαρμογές να λειτουργούν χωρίς να εξαρτώνται από τις λεπτομέρειες της υλοποίησής τους.

Southbound Interface [4] [6]**:** Σε μία SDN αρχιτεκτονική η νότια διεπαφή παρέχει επικοινωνία ανάμεσα στον SDN Controller με τα χαμηλότερου επιπέδου δομικά στοιχεία όπως μεταγωγείς και δρομολογητές για την ταυτοποίηση της

τοπολογίας του δικτύου, για τον καθορισμό ροών και εφαρμογής αιτημάτων μέσω του Northbound Interface. Υπάρχουν πολλά παραδείγματα για το Southbound APIs. Στην εργασία αυτή χρησιμοποιήθηκε το OpenFlow protocol που είναι το πιο βασικό για την επικοινωνία του Control plane με το Data plane το οποίο θα αναλυθεί στο κεφάλαιο 2.

1.3 SDN Controllers

Σήμερα υπάρχει μία μεγάλη γκάμα SDN Controllers που έχουν αναπτυχθεί και εφαρμοστεί τόσο στο βιομηχανικό τομέα όσο και στον ακαδημαϊκό. Πολλοί Controllers ξεκίνησαν ως ανοιχτού κώδικα projects συνεργαζόμενοι από εταιρίες δικτύων και οργανισμούς. Στη συνέχεια κάποιοι αγοράστηκαν από εταιρίες και μετατράπηκαν σε ιδιωτικά και εμπορικά projects εξυπηρετώντας συγκεκριμένους σκοπούς και λύσεις. Στο υποκεφάλαιο αυτό θα παρουσιάσουμε τι είναι SDN Controller καθώς και τούς πιο διάσημους SDN Controllers της αγοράς με τα χαρακτηριστικά και τις λειτουργίες τους.



Εικόνα 2. Σχηματική αναπαράσταση εξέλιξης των SDN Controllers

Πηγή: https://www.researchgate.net/figure/2-The-Evolution-of-SDN-Controllers-Frameworks_fig2_326222339

1.3.1 Τι είναι SDN Controllers

Σε μία SDN αρχιτεκτονική ο Controller είναι μία εφαρμογή που διαχειρίζεται τον έλεγχο ροής για την βέλτιστη διαχείριση του δικτύου και απόδοση εφαρμογής. Συγκεκριμένα ο SDN Controller κάνει έλεγχο του δικτύου τρέχοντας σε έναν server χρησιμοποιώντας πρωτόκολλα για τον έλεγχο ροής στους μεταγωγείς και δρομολογητές.

Οι SDN Controllers κατευθύνουν την κυκλοφορία του δικτύου μέσω πολιτικών προώθησης που ένας χειριστής δικτύου τοποθετεί εξοικονομώντας έτσι χειροκίνητες ρυθμίσεις για κάθε ξεχωριστή δικτυακή συσκευή. Απομακρύνοντας το επίπεδο ελέγχου από το υπόλοιπο υλικό του δικτύου και διαχειρίζοντας το ως λογισμικό, ο κεντροποιημένος Controller διευκολύνει την αυτοματοποιημένη διαχείριση του δικτύου και κάνει ευκολότερη την διαχείριση και ενσωμάτωση εμπορικών εφαρμογών. Στην πραγματικότητα ένας SDN Controller εξυπηρετεί διαφορετικά είδη λειτουργικού συστήματος (OS) για το δίκτυο.

Γενικά, ο Controller είναι ο πυρήνας μιας SDN δικτύωσης όπου είναι τοποθετημένος ανάμεσα στο επίπεδο προώθησης δεδομένων και το επίπεδο εφαρμογών. Έτσι οποιαδήποτε επικοινωνία ανάμεσα στα δύο επίπεδα περνάει από τον Controller

1.3.2 Είδη SDN Controllers

Μερικοί από τους πιο διάσημους SDN Controllers είναι:

- Trema
- NOX
- POX
- Floodlight
- ONOS
- Ryu
- OpenDaylight

Το **Trema** [7]είναι ένας ανοιχτού κώδικα ελεγκτής που χρησιμοποιείται για την ανάπτυξη του OpenFlow ελεγκτών για δίκτυα SDN σε γλώσσες Ruby και C. Η

διανομή αυτή περιέχει όλο το κώδικα πηγής του Trema που χρειάζεται κάποιος για να αναπτύξει τον δικό του OpenFlow Controller.

Το **NOX** [8]είναι μια ανοιχτού κώδικα ανάπτυξη ελεγκτή για SDN δίκτυα. Η πρώτη SDN τεχνολογία που πήρε πραγματική ονομαστική αναγνώριση ήταν το OpenFlow και στη συνέχεια αναπτύχθηκε το NOX στο Nicira Networks, ο πρώτος OpenFlow Controller. Το 2008 δωρίστηκε σε ερευνητικές κοινότητες και από τότε αποτελεί τη βάση για πολλά ερευνητικά projects υποστηρίζοντας C++ και το OpenFlow 1.0 API.

Το **POX** είναι και αυτό μια ανοιχτού κώδικα ανάπτυξη για βασιζόμενα σε Python SDN δίκτυα εφαρμογής ελέγχου, όπως οι OpenFlow SDN Controllers. Το POX, που επιτρέπει την ραγδαία ανάπτυξη και προτυποποίηση, γίνεται πιο ευρέως γνωστό project σε σχέση με το αδελφικό του το NOX.

Το **Floodlight** [9] είναι ένας SDN Controller βασιζόμενος σε Java (με την άδεια της Apache). Είναι μια από τις σημαντικές συνεισφορές του Big Switch Networks στις ανοιχτού κώδικα κοινότητες. Η αρχιτεκτονική του Floodlight είναι βασισμένη στο Big Network Controller (BNC) καθώς οι εφαρμογές του είναι γραμμένες από τον οποιονδήποτε προγραμματιστή. Τέλος, το Floodlight μπορεί εύκολα να βελτιωθεί και να αναπτυχθεί υποστηρίζοντας OpenFlow Switches 1.0, 1.3, 1.4.

Το **ONOS** (Open Network Operating System) [10] είναι ένα project ανοιχτού κώδικα που φιλοξενείται από το Linux Foundation σε γλώσσα Java. Στόχος αυτού του project είναι να δημιουργήσει SDN λειτουργικά συστήματα Telecommunications Service Providers (TSP) που είναι σχεδιασμένα για επεκτασιμότητα, υψηλή απόδοση και διαθεσιμότητα.

Το **Ryu** [11] [12] είναι ένας ανοιχτού κώδικα ελεγκτής για SDN δίκτυα βασιζόμενος σε Python. Επίσης είναι ένας ελεγκτής σχεδιασμένος για να αυξάνει την ευελιξία του δικτύου διευκολύνοντας τη διαχείριση και τη προσαρμογή της κυκλοφορίας . Τέλος, ο Ryu υποστηρίζει πολλά πρωτόκολλα για τη διαχείριση δικτυακών συσκευών, όπως το OpenFlow 1.0, 1.2, 1.3, 1.4, 1.5 και Nicira Extensions.

Το **OpenDaylight** είναι άλλος ένας ανοιχτού κώδικα SDN Controller που φιλοξενείται από το Linux Foundation. Αυτή τη στιγμή είναι από τους πιο γνωστούς

SDN Controllers. Είναι ένας ελεγκτής που υποστηρίζει πολλά πρωτόκολλα ταυτόχρονα δίνοντας έτσι τη δυνατότητα στους διαχειριστές να επιλύουν πολύπλοκα ζητήματα που προκύπτουν. Ο OpenDaylight υποστηρίζει το OpenFlow, Open VSwitch (OVS) Database (OVSDB), NETCONF, and BGP. Τέλος, ο OpenDaylight είναι εφαρμοσμένος αποκλειστικά σε λογισμικό καθώς υποστηρίζει το δικό του Java Virtual Machine (JVM) με αποτέλεσμα να μπορεί να αναπτυχθεί σε οποιοδήποτε λειτουργικό σύστημα που υποστηρίζει Java.

Πίνακας 1. Σύγκριση δημοφιλών SDN ελεγκτών

	Trema	NOX	POX	Floodlight	ONOS	Ryu	ODL
Interfaces	SB OpenFlow	SB OpenFlow	SB OpenFlow	SB OpenFlow NB Java & REST	SB OpenFlow, NETCOF	SB OpenFlow	SB OpenFlow
GUI	No	Yes	Yes	Web UI	Yes	Yes	Yes
REST API	No	No	No	Yes	Yes	Yes	Yes
Productivity	High	Medium	Medium	Medium	Medium	Medium	Medium
Open Source	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Documentation	Medium	Poor	Poor	Good	High	Medium	Medium
Language	C/Ruby	Python/C++	Python	Java	Java	Python	Java
Modularity	Medium	Medium	Medium	High	High	Medium	High
Platform Support	Linux	Linux	Linux, Mac OS & Windows	Linux, Mac & Windows	Linux & Mac OS	Linux	Linux
TLS Support	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OpenFlow Support	OF v1.0	OF v1.0	OF v1.0	OF v1.0	OF v 1.0-v1.5	OF v1.0, v1.2, v1.3, Nicira Extensions	OF v1.0

1.4 Πλεονεκτήματα χρήσης SDN

Τα βασικά πλεονεκτήματα που προσφέρει η χρήση των SDN δικτύων σε σχέση με τα παραδοσιακά δίκτυα [13] [14]:

- **Παροχή κεντρικού δικτύου**: Το SDN βοηθάει στην κεντροποιημένη διαχείριση και παροχή, προσφέροντας μια ενοποιημένη οπτική για ολόκληρο το δίκτυο. Επίσης το SDN έχει τη δυνατότητα να επιταχύνει την παράδοση

υπηρεσιών και να ενισχύει την ευελιξία παρέχοντας εικονικές και φυσικές δικτυακές συσκευές σε μια κεντρική τοποθεσία.

- **Μείωση λειτουργικού κόστους:** Πολλά προνόμια του SDN, όπως έχοντας μια αποτελεσματική διαχείριση, βελτιώσεις αξιοποίησης διακομιστών και βελτιωμένος έλεγχος εικονικοποίησης συμβάλλουν στην μείωση του λειτουργικού κόστους. Επειδή πολλά τακτικά ζητήματα διαχείρισης δικτύου μπορούν να γίνουν αυτόματα και κεντρικά το SDN συμβάλλει και στην αύξηση της εξοικονόμησης διαχείρισης.
- **Εξοικονόμηση υλικού και μειωμένες δαπάνες κεφαλαίου:** Το SDN συμβάλλει στην επαναφορά των κλασικών συσκευών και απλοποιεί την διαδικασία βελτιστοποίησης εμπορικών συσκευών. Ακολουθώντας τις οδηγίες του SDN ελεγκτή, το παλαιότερο υλικό μπορεί να επαναχρησιμοποιηθεί ενώ το υλικό με μικρό κόστος μπορεί να αξιοποιηθεί για καλυτέρα αποτελέσματα. Αυτή η διαδικασία επιτρέπει στις νέες συσκευές να γίνουν “white box” μεταγωγείς έχοντας την νοημοσύνη βασισμένη στον SDN ελεγκτή.
- **Αφαίρεση πόρων cloud:** Η χρήση του SDN για την αφαίρεση των πόρων του cloud βοηθά στην απλοποίηση της διαδικασίας της ενοποίησης των πόρων cloud. Ο ελεγκτής SDN μπορεί να διαχειρίζεται όλα τα δομικά μέρη του δικτύου που αποτελούνται από μεγάλες πλατφόρμες κέντρων δεδομένων.
- **Λεπτομερής προσέγγιση ασφάλειας:** Όταν υπάρχει μια κεντρική κονσόλα για τη διαχείριση του δικτύου, γίνεται ευκολότερη η παρακολούθηση και ο έλεγχος των χαρακτηριστικών ασφάλειας. Έτσι μειώνεται η εργασία με ή βασιζόμενη σε πολλαπλές εφαρμογές σε όλο το σύστημα του δικτύου. Η διαδικασία απλοποιείται έχοντας ένα κεντρικό σημείο και αξιοποιώντας το για καλύτερη προσέγγιση της ασφάλειας. Η ίδια κονσόλα μπορεί να χρησιμοποιηθεί για να διαμοιράζει πληροφορίες σε περίπτωση σχετικού συναγερμού ασφάλειας.
- **Αυτοματοποίηση:** Το σύνολο του αυτοματισμού που μπορεί να αξιοποιηθεί σε ένα SDN δίκτυο μπορεί να προσφέρει βοήθεια με πολλούς τρόπους. Είναι

ο καλύτερος τρόπος για επένδυση ταχύτητας στις συνολικές λειτουργίες του δικτύου. Σε αντίθεση με πριν, τα σημερινά δίκτυα δε χρειάζεται να αγωνίζονται για τη συνδεσιμότητα στο διαδίκτυο. Με το SDN είναι πιθανό να αλλάξουμε αυτόματες αποκρίσεις στο Cloud. Η διαδικασία δουλεύει εξίσου καλά σε περιβάλλοντα όπως SD-WAN δίκτυα σε επίπεδο επιχείρησης.

- **Συνεπής παράδοση περιεχομένου:** Τελευταίο και βασικό πλεονέκτημα των SDN δικτύων είναι η ικανότητα να χειρίζεται την ροή δεδομένων. Με το προαναφερθέν πλεονέκτημα γίνεται ευκολότερη η εφαρμογή του QoS (Quality of service) για το VoIP(Voice over Internet Protocol) και για την μεταφορά των πολυμέσων. Με το SDN η μετάδοση βίντεο σε υψηλή ποιότητα γίνεται πιο εύκολη από τότε που το SDN υποστηρίζει ανταπόκριση δικτύου, προσφέροντας βελτιωμένη εμπειρία.

1.5 Εικονικοποίηση λειτουργιών δικτύου (Network Function Virtualization)

Το European Telecommunications Standards Institute (ETSI), μια κοινοπραξία παροχών υπηρεσιών συμπεριλαμβανομένου το AT&T, China Mobile, BT Group, Deutsche Telekom και πολλών άλλων, παρουσίασαν πρώτοι την ιδέα για το πρότυπο του Network Function Virtualization στο OpenFlow World Congress το 2012. Οι πάροχοι αυτοί αναζητούν τρόπους για να επισπεύσουν την ανάπτυξη των υπηρεσιών δικτύου.

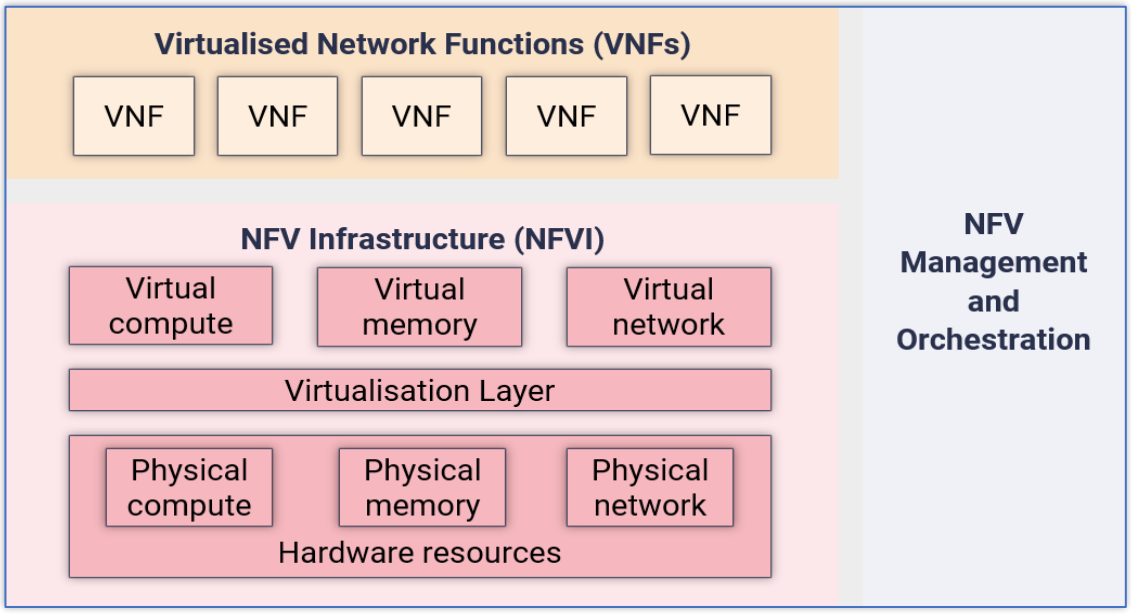
Παρουσιάζοντας νέες υπηρεσίες δικτύου συνήθιζε να είναι μια πολύπλοκη διαδικασία που απαιτούσε χώρο και δύναμη για τις επιπλέον συσκευές. Καθώς το κόστος της ενέργειας και του χώρου αυξάνεται και ο αριθμός ικανοτήτων των μηχανικών δικτύου του υλικού μειώνεται, η επιτροπή του ETSI στράφηκε στο Network Function Virtualization για να λύσει και τα δύο προβλήματα. Το NFV εξαλείφει την ανάγκη για φυσικό χώρο για συσκευές υλικού και δεν απαιτεί μεγάλη εμπειρία δικτύων για ρύθμιση και επισκευή.

Σήμερα, αρκετά ανοιχτού κώδικα projects εργάζονται για την ανάπτυξη του NFV προτύπου, συμπεριλαμβανομένου το ETSI, Open Platform for NFV, Open Network Automation Platform, Open Source Mano and MEF. Τόσοι πολλοί

διαφορετικοί οργανισμοί με ανταγωνιστικές προτάσεις για πρότυπα το έχουν κάνει απαιτητικό για τους παρόχους υπηρεσιών να αισθάνονται ασφαλής. Ακόμα μεγαλώνει η δημοτικότητα λόγω της γρήγορης εξάπλωσης της πολυπλοκότητας και των απαιτήσεων των δικτύων σήμερα. [15]

1.5.1 Τι είναι Network Function Virtualization και πως λειτουργεί

Το Network Function Virtualization (NFV) [15] [16] είναι ο διαχωρισμός των λειτουργιών του δικτύου από τις ιδιόκτητες συσκευές υλικού και διαχειρίζοντας τις συσκευές αυτές ως λογισμικό σε εικονικές μηχανές (Virtual Machines). Οι διαφορετικές λειτουργίες όπως τοίχος προστασίας(firewall), έλεγχος κυκλοφορίας(traffic control) και εικονική δρομολόγηση(virtual routing) ονομάζονται virtual network functions (VNFs).



Εικόνα 3. High-level NFV framework ETSI
Πηγή: https://stlpartners.com/telco_cloud/nfv-architectural-framework/

Το Network Function Virtualization αντικαθιστά τη λειτουργικότητα που παρέχεται από ξεχωριστά δικτυακά τμήματα υλικού (hardware). Αυτό σημαίνει ότι εικονικές μηχανές (VMs) τρέχουν λογισμικό το οποίο επιτυγχάνει τις ίδιες δικτυακές λειτουργίες με ένα παραδοσιακό δίκτυο. Η εξισορρόπηση φόρτου, δρομολόγηση και ασφάλεια τοίχου προστασίας εκτελούνται από λογισμικό αντί για υλικό. Ένας ελεγκτής SDN επιτρέπει στους μηχανικούς δικτύων να προγραμματίζουν όλα τα

τμήματα του εικονικού δικτύου αλλά αυτοματοποιεί και την παρακολούθηση του δικτύου.

1.5.2 Αρχιτεκτονική Network Function Virtualization

Το ETSI πρότεινε την NFV αρχιτεκτονική [17], η οποία έχει βοηθήσει στο καθορισμό της υλοποίησης των προτύπων του NFV. Κάθε δομικό στοιχείο της αρχιτεκτονικής αυτής βασίζεται στα πρότυπα αυτά για την προώθηση σταθερότητας και διαλειτουργικότητας. Μια NFV αρχιτεκτονική αποτελείται από τα εξής τμήματα:

- **Virtual Network Functions (VNFs):** Εφαρμογές λογισμικού που παράγουν δικτυακές λειτουργίες, όπως μοίρασμα αρχείων, ρυθμίσεις Internet Protocol (IP) και κατάλογος υπηρεσιών.
- **Network Functions Virtualization Infrastructure (NFVI):** Αποτελείται από τα δομικά στοιχεία-υπολογισμός, αποθήκευση, δικτύωση σε μια πλατφόρμα που υποστηρίζει λογισμικό ώστε να εκτελεί δικτυακές εφαρμογές.
- **Management, automation, and network orchestration (MANO):** Παρέχει το πλαίσιο για τη διαχείριση της δομής του NFV και για την παροχή νέων VNFs.

1.5.3 Πλεονεκτήματα χρήσης Network Function Virtualization

Οι λόγοι για τους οποίους οι οργανισμοί χρησιμοποιούν το NFV είναι:

- **Καλύτερη επικοινωνία και προσβασιμότητα επικοινωνιών:** Το NFV βελτιώνει δικτυακές λειτουργίες μεταμορφώνοντας πως οι σχεδιαστές δικτύου παράγουν δικτυακές υπηρεσίες. Η διαδικασία αυτή πραγματοποιείται με τη χρήση μιας αρχιτεκτονικά και δημιουργικά σχεδιασμένης μεθόδου συνδέοντας μεταξύ διαφορετικούς δικτυακούς κόμβους ώστε να παράγουν ένα επικοινωνιακό κανάλι που να μπορεί να παράγει ελεύθερη προσβάσιμη πληροφορία στους χρήστες.
- **Μειωμένο κόστος:** Μια από τις απηχήσεις του NFV πάνω στους δρομολογητές**,** τοίχους προστασίας και εξισορροπητή φόρτου είναι ότι δεν χρειάζεται ιδιοκτήτες δικτύου να αγοράσουν ειδικές συσκευές υλικού για να εκτελέσουν την εργασία τους ή την παραγωγή υπηρεσιών. Το προνόμιο αυτό

βοηθά ώστε να μειωθεί το κόστος των λειτουργικών δαπανών και επιτρέπει την εργασία να εκτελείται με λιγότερο πιθανά λειτουργικά ζητήματα.

- **Βελτιωμένη ευελιξία και επιτάχυνση χρόνου για αγορά νέων προϊόντων και ενημερώσεων:** Το NFV βοηθάει τους οργανισμούς να ενημερώνουν τις υποδομές λογισμικού όταν το δίκτυο απαιτεί αλλαγή. Καθώς επιχειρηματικές απαιτήσεις αλλάζουν και νέες ευκαιρίες στην αγορά ανοίγουν, το NFV βοηθάει τους οργανισμούς να προσαρμόζονται γρήγορα. Επειδή η υποδομή ενός δικτύου μπορεί να αλλάξει για καλύτερη υποστήριξη νέου προϊόντος, ο χρόνος περιόδου για αγορά μπορεί να μειωθεί.

- **Βελτιωμένη επεκτασιμότητα και διαχείριση πόρων:** Επειδή τα VMs έχουν εικονικοποιημένες υπηρεσίες, μπορούν να λαμβάνουν τμήματα από εικονικούς πόρους σε x86 συστήματα εξυπηρετητών επιτρέποντας έτσι πολλαπλά VMs να εκτελούνται από ένα μόνο εξυπηρετητή με καλύτερη κλιμάκωση, βασισμένα στους κατάλοιπους πόρους. Το πλεονέκτημα αυτό βοηθάει να κατευθύνει τους αχρησιμοποίητους πόρους εκεί που χρειάζονται και να ενισχύουν την αποτελεσματικότητα στα κέντρα δεδομένων με εικονικές υποδομές. Το NFV δίνει τη δυνατότητα για γρήγορη και εύκολη κλιμάκωση των πόρων του, βασισμένους στην εισερχόμενη κυκλοφορία και απαιτήσεις των πόρων.

  Όταν ένα κέντρο δεδομένων ή μια παρόμοια υποδομή είναι εικονικά μπορούν να κάνουν περισσότερα με λιγότερους πόρους καθώς ένας εξυπηρετητής μπορεί να εκτελεί διαφορετικά VNFs ταυτόχρονα παράγοντας την ίδια ποσότητα εργασίας. Επιτρέπει για μια αυξημένη χωρητικότητα φόρτου εργασίας ενώ μειώνονται τα αποτυπώματα του κέντρου δεδομένων, κατανάλωση δύναμης και ανάγκες ψύξης.

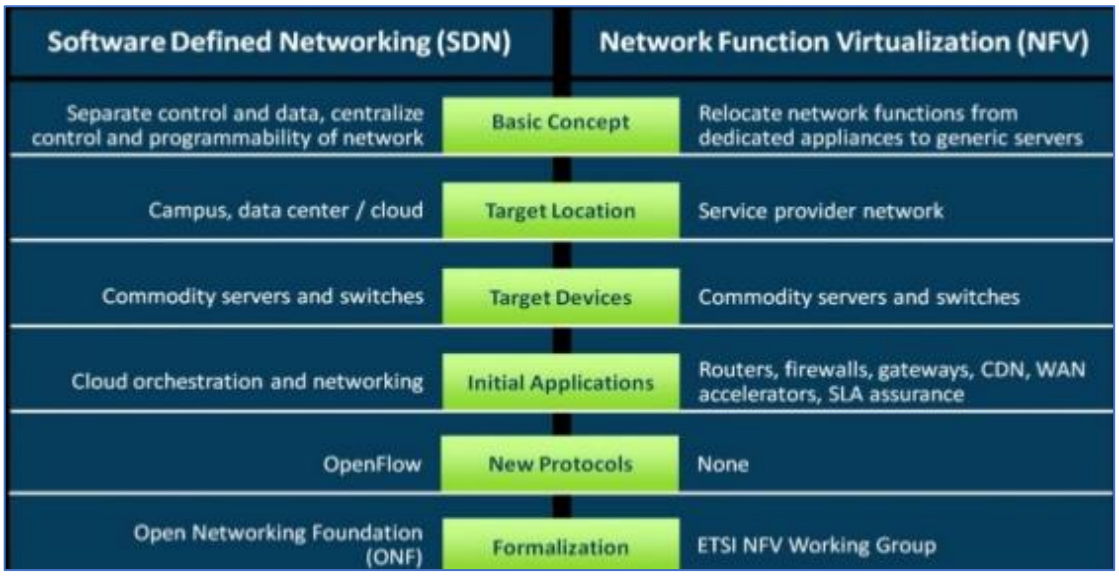
- **Μειωμένη λειτουργία των προμηθευτών:** Το μεγαλύτερο προνόμιο της εκτέλεσης των VNFs στο COTS (commercial off-the-shelf) hardware είναι ότι οι οργανισμοί δεν είναι αλυσιδωτά ιδιόκτητοι, σπαταλώντας πολύ χρόνο και εργασία για ανάπτυξη και ρυθμίσεις.

1.5.4 Γιατί χρησιμοποιούμε Network Function Virtualization στις επιχειρήσεις

Από το 2018, οι περισσότεροι προσωπικοί υπολογιστές και συσκευές κινητών είναι κατασκευασμένα στην x86 αρχιτεκτονική. Τα Virtualized Networking Components (VNCs) χρησιμοποιούνται από το NFV υποστηρίζουν μια ανεξάρτητου υλικού υποδομή. Εικονικοί πόροι, συμπεριλαμβανομένου και πόρων για υπολογισμό, αποθήκευση και λειτουργίες δικτύου, μπορούν να τοποθετηθούν σε x86 εξυπηρετητές και παρόμοιου τύπου COTS hardware. Επίσης το επίπεδο δεδομένων και το επίπεδο ελέγχου ανάμεσα στο κέντρο δεδομένων και στα εξωτερικά δίκτυα μπορούν να εικονικοποιηθούν με το NFV.

1.5.5 Διαφορές μεταξύ Network Function Virtualization και SDN

Το NFV και το SDN δεν εξαρτώνται το ένα από το άλλο αλλά έχουν ομοιότητες. Και οι δυο τεχνολογίες βασίζονται στην εικονικοποίηση και χρησιμοποιούν διαχωρισμό δικτύου αλλά διαφέρουν στο πως διαχωρίζουν λειτουργίες και αφαιρούν πόρους.



Εικόνα 4. Σχηματική αναπαράσταση σύγκρισης NFV και SDN

Πηγή: https://www.netmanias.com/en/post/blog/11022/sdn-nfv/difference-between-sdn-and-nfv-which-one-is-better

Το SDN διαχωρίζει τις λειτουργίες προώθησης του δικτύου από τις λειτουργίες ελέγχου του δικτύου με σκοπό την δημιουργία ενός δικτύου που είναι κεντρικά διαχειρίσιμο και προγραμματίσιμο. Το NFV διαχωρίζει τις λειτουργίες δικτύου από

το υλικό. Το NFV επίσης υποστηρίζει το SDN παρέχοντας την υποδομή στην οποία το SDN λογισμικό θα εκτελεστεί. Τέλος, το NFV και το SDN μπορούν να χρησιμοποιηθούν μαζί, εξαρτώντας από το τι θέλει να πετύχει ο καθένας, και οι δυο χρησιμοποιούν εμπορικά υλικά. Με το NFV και SDN μπορούμε να δημιουργήσουμε μια αρχιτεκτονική δικτύου που είναι περισσότερο ευέλικτη, προγραμματίσιμη και χρησιμοποιεί πόρους αποτελεσματικά.

1.6 Υπολογιστική Νέφους (Cloud Computing)

Η έννοια του Υπολογιστικού Νέφους (Cloud Computing) πρωτοεμφανίστηκε τη δεκαετία του 1950 σε εκπαιδευτικά ινστιτούτα και εταιρείες, και η χρήση του πραγματοποιούνταν από κεντρικούς υπολογιστές μεγάλων υπολογιστικών και αποθηκευτικών δυνατοτήτων. Οι χρήστες είχαν πρόσβαση σε αυτούς τους υπολογιστές μέσω τερματικών (dumb terminals), οι οποίοι δεν είχαν ούτε υπολογιστική ισχύ ούτε αποθηκευτικές ικανότητες. Ο όρος του Υπολογιστικού Νέφους έγινε ευρύτερα γνωστός τη δεκαετία του 1970, όταν η IBM και η Google αποφάσισαν να συνεργαστούν στο συγκεκριμένο τεχνολογικό πεδίο. Αρχικά η IBM παρουσίασε το λειτουργικό σύστημα εικονικών μηχανών (VM operating system), το οποίο παρείχε τη δυνατότητα να εργάζονται πολλές εικονικές μηχανές (virtual machines) στο ίδιο μηχάνημα. Κάθε εικονική μηχανή είναι μία αυτοτελής οντότητα που εκτελεί το δικό της λειτουργικό σύστημα και παρέχει υπολογιστικούς πόρους, όπως μια κεντρική μονάδα επεξεργασίας, μνήμης και μονάδες εισόδου-εξόδου.

Τη δεκαετία του 1980, η πρώτη τακτική χρήση των προσωπικών υπολογιστών συνοδεύτηκε από την υπόσχεση ότι οι χρήστες θα ήταν σε θέση να αποφασίζουν οι ίδιοι για το υπολογιστικό τους περιβάλλον. Η εμφάνιση του Διαδικτύου και του Παγκόσμιου Ιστού, εκτόξευσε τη φήμη του Υπολογιστικού Νέφους .

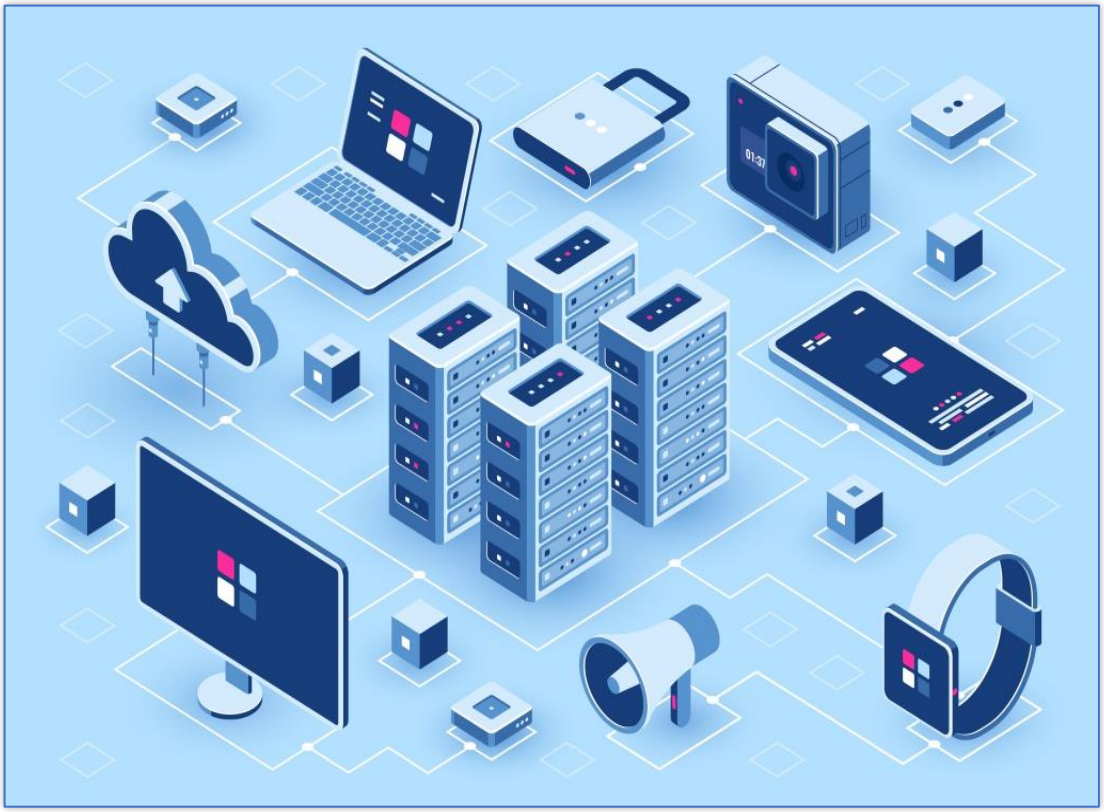
Τη δεκαετία του 1990 εμφανίστηκαν τα Εικονικά Ιδιωτικά Δίκτυα (Virtual Private Networks). Μέχρι τότε, οι εταιρείες τηλεπικοινωνιών υποστήριζαν τα κυκλώματα δεδομένων σημείο προς σημείο (point-to-point data circuits).

Μέχρι το 2000, κολοσσοί της Πληροφορικής όπως η Amazon, η Microsoft και η Google, ασχολήθηκαν με την ανάπτυξη και την παροχή υπηρεσιών Υπολογιστικού

Νέφους. Ακολούθησαν κάποια γεγονότα που θεωρούνται ευρέως ως ορόσημα στην ιστορία του Υπολογιστικού Νέφους.

Το 2006, η Amazon παρουσίασε το Ελαστικό Υπολογιστικό Νέφος (Elastic Compute Cloud(EC2)). Μία εμπορική υπηρεσία βασισμένη στο Παγκόσμιο Ιστό, που παρείχε στο χρήστη τη δυνατότητα να υλοποιεί εφαρμογές σε ενοικιαζόμενα μηχανήματα.

Το 2008, προωθήθηκε στην αγορά το Eucalyptus, η πρώτη πλατφόρμα ανοικτού κώδικα για ανάπτυξη ιδιωτικών σύννεφων (Private Clouds). Την ίδια χρονιά, η Google κυκλοφόρησε το Google App Engine, μια πλατφόρμα που υποστήριζε διάφορες υπηρεσίες του Υπολογιστικού Νέφους. Στη συνέχεια των εξελίξεων το 2011, η IBM παρουσίασε το IBM SmartCloud ενώ το 2012 κυκλοφόρησε το Oracle Cloud. Τέλος, η Microsoft ανακοίνωσε επίσης ότι θα εισαγάγει το Cloud Computing στην επόμενη μεγάλη ενημέρωση λύσεων Dynamics ERP και θα λειτουργούν μέσω της πλατφόρμας Windows Azure.



Εικόνα 5. Σχηματική αναπαράσταση λειτουργίας Υπολογιστικού Νέφους

Πηγή: https://www.techiexpert.com/7-trends-that-define-the-next-phase-of-cloud-computing/

Η Υπολογιστική Νέφους (Cloud Computing) [18] είναι μια τεχνολογία που χρησιμοποιεί το διαδίκτυο και κεντρικούς απομακρυσμένους εξυπηρετητές (servers) για την διατήρηση μεγάλου όγκου δεδομένων και εφαρμογών. Επιτρέπει στους καταναλωτές και στις επιχειρήσεις να χρησιμοποιούν εφαρμογές χωρίς να χρειάζονται εγκατάσταση αλλά και να έχουν πρόσβαση σε όλα τους τα αρχεία οποιαδήποτε στιγμή και σε οποιοδήποτε μέρος και αν βρίσκονται έχοντας μόνο έναν υπολογιστή συνδεδεμένο στο διαδίκτυο. Με άλλα λόγια, η Υπολογιστική Νέφους προσφέρει την Πληροφορική Τεχνολογία (Information Technology) ως υπηρεσία (IT- as-a-Service). Αντί να δημιουργήσει κάποιος ολόκληρη τεχνολογική υποδομή για να φιλοξενήσει βάσεις δεδομένων και λογισμικά, τρίτοι αναλαμβάνουν τη συγκεκριμένη φιλοξενία σε μεγάλες εκτάσεις με εξυπηρετητές.

1.6.1 Μοντέλο ανάπτυξης και μορφές υπολογιστικού νέφους

Υπάρχουν διαφορετικά μοντέλα για την ανάπτυξη μιας υπηρεσίας νέφους, καθένα από τα οποία είναι διαφορετικά μεταξύ τους. [19]

- **Public Cloud:** Το δημόσιο νέφος παρέχει τις υπηρεσίες σε εξυπηρετητές και αποθηκευτικούς χώρους μέσω διαδικτύου. Διατίθεται από ένα τρίτο φορέα παροχής υπηρεσιών νέφους, που χειρίζεται και ελέγχει όλο το υλικό, λογισμικό και γενικά την υποδομή. Οι πελάτες έχουν πρόσβαση στις υπηρεσίες μέσω λογαριασμών που μπορεί να είναι προσβάσιμοι από τον οποιονδήποτε.
- **Private Cloud:** Το ιδιωτικό νέφος είναι διατεθειμένο για συγκεκριμένους πελάτες, όπως μια επιχείρηση ή οργανισμός. Το κέντρο δεδομένων υπηρεσίας της εταιρίας μπορεί να φιλοξενήσει την υπηρεσία υπολογιστικού νέφους. Πολλά ιδιωτικά νέφη υπηρεσιών υπολογιστικής παρέχονται σε ιδιωτικά δίκτυα.
- **Hybrid Cloud:** Το υβριδικό νέφος είναι ένας συνδυασμός του δημόσιου και ιδιωτικού νέφους. Αυτός ο τύπος του μοντέλου επιτρέπει στο χρήστη περισσότερη ευελιξία και βοηθά στην βελτίωση της υποδομής του χρήστη και ασφάλεια.

Υπάρχουν τρεις βασικές μορφές υπηρεσιών Υπολογιστικού Νέφους όπως Infrastructure as a Service, Platform as a Service και Software as a Service. Κάθε μορφή Υπολογιστικού Νέφους παρέχει διαφορετικά επίπεδα ελέγχου, ευελιξίας και διαχείρισης ώστε να μπορεί ο καθένας να επιλέξει την κατάλληλη υπηρεσία για τις ανάγκες του.

- **Infrastructure as a Service (IaaS)** [20]**:** Είναι ένας τύπος υπηρεσίας υπολογιστικού νέφους που παρέχει την απαραίτητη υπολογιστική, αποθηκευτικό χώρο και δικτυακούς πόρους κατά απαίτηση. Μετακινώντας την υποδομή ενός οργανισμού σε μια IaaS λύση βοηθάει στη μείωση της συντήρησης των κέντρων δεδομένων, εξοικονόμηση χρημάτων για δαπάνες υλικού και κέρδος πραγματικού χρόνου. Οι λύσεις IaaS δίνουν ευελιξία για κλιμάκωση των IT πόρων, γρήγορη παροχή νέων εφαρμογών και αύξηση της αξιοπιστίας από την υπάρχουσα υποδομή. Επίσης με το IaaS παρακάμπτεται το κόστος και η πολυπλοκότητα για αγορά και διαχείριση φυσικών εξυπηρετητών και υποδομής κέντρων δεδομένων. Κάθε πόρος προσφέρεται ως ξεχωριστό τμήμα υπηρεσίας και μπορεί μόνο να αγοραστεί για ένα συγκεκριμένο πόρο για όσο χρονικό διάστημα χρειάζεται. Μια υπηρεσία παροχής υπολογιστικού νέφους όπως το Azure διαχειρίζεται την υποδομή καθώς ο χρήστης αγοράζει, εγκαθιστά, ρυθμίζει και διαχειρίζεται το δικό του λογισμικό, συμπεριλαμβανομένου λειτουργικό σύστημα, ενδιάμεσο λογισμικό και εφαρμογές.

  <u>Συνήθεις επιχειρησιακά σενάρια IaaS</u>

  - ΄Ελεγχος και ανάπτυξη
  - Αποθήκευση, αντίγραφο ασφαλείας και ανάκτηση
  - Εφαρμογές διαδικτύου
  - Μεγάλη απόδοση υπολογιστικής

  <u>Πλεονεκτήματα IaaS</u>

  - Μειώνει δαπάνες κεφαλαίου και βελτιώνει το κόστος
  - Αυξάνει την κλιμάκωση και απόδοση των IT φόρτου εργασίας (workload)
  - Αυξάνει την σταθερότητα, αξιοπιστία και υποστήριξη
  - Βελτιώνει την επιχειρηματική εξέλιξη και καταστροφή ανάκτησης

- Βελτιώνει την ασφάλεια
- Βοηθάει στην καινοτομία και την απόκτηση νέων εφαρμογών στους χρήστες γρηγορότερα

- **Platform as a Service (PaaS)** [20]**:** Είναι μια ολοκληρωμένη εφαρμογή και ανάπτυξη περιβάλλοντος σε νέφος, με πόρους που επιτρέπουν στο χρήστη να διαμοιράζει οτιδήποτε από μια απλή εφαρμογή βασισμένη σε νέφος σε πιο σύνθετες βασισμένες σε νέφος επιχειρηματικές εφαρμογές. Ο χρήστης αγοράζει τους πόρους που χρειάζεται από ένα πάροχο υπηρεσιών νέφους και αποκτά πρόσβαση μέσω μιας ασφαλούς σύνδεσης στο διαδίκτυο. Το PaaS περιλαμβάνει επίσης υποδομή όπως εξυπηρετητές, αποθηκευτικό χώρο και δικτύωση αλλά και ενδιάμεσο λογισμικό, εργαλεία ανάπτυξης, επιχειρηματική νοημοσύνη, συστήματα διαχείρισης βάσεων δεδομένων κ.α. Το PaaS είναι σχεδιασμένο για να υποστηρίζει τον απόλυτο κύκλο ζωής (lifecycle) της εφαρμογή διαδικτύου (ανάπτυξη, έλεγχος, διαχείριση και ενημέρωση). Τέλος, το PaaS επιτρέπει στο χρήστη να αποφύγει τις δαπάνες και την πολυπλοκότητα για αγορά και διαχείριση άδειας λογισμικού, την υποκείμενη υποδομή εφαρμογής και ενδιάμεσο λογισμικό, εργαλεία ανάπτυξης και άλλους πόρους. Ο χρήστης διαχειρίζεται τις εφαρμογές και τις υπηρεσίες που αναπτύσσει και ο πάροχος υπηρεσιών νέφους διαχειρίζεται όλα τα υπόλοιπα.

  <u>Συνήθη σενάρια PaaS</u>
  - Ανάπτυξη πλαισίου
  - Στατιστικά στοιχεία ή επιχειρησιακή νοημοσύνη

  <u>Πλεονεκτήματα PaaS</u>
  - Μείωση χρόνου προγραμματισμού
  - Προσθέτει εξελιγμένες δυνατότητες χωρίς να προσθέτει προσωπικό
  - Αναπτύσσεται για πολλαπλές πλατφόρμες ευκολότερα
  - Χρησιμοποιεί οικονομικά εργαλεία
  - Υποστηρίζει γεωγραφικά διαμοιρασμένες ομάδες ανάπτυξης
  - Αποτελεσματική διαχείριση κύκλου ζωής εφαρμογής

- **Software as a Service (SaaS**) [20]: Επιτρέπει στους χρήστες να συνδέονται και να χρησιμοποιούν εφαρμογές βασισμένες σε νέφος μέσω διαδικτύου. Κοινά παραδείγματα είναι το e-mail και εργαλεία office. Το SaaS παρέχει μια ολοκληρωμένη λύση λογισμικού που ο χρήστης μπορεί να αγοράσει από έναν πάροχο υπηρεσιών νέφους. Ο χρήστης νοικιάζει την χρήση μιας εφαρμογής για τον οργανισμό του και οι υπόλοιποι χρήστες συνδέονται στο διαδίκτυο μέσω ενός περιηγητή(browser). Όλη η υποκείμενη υποδομή, ενδιάμεσο λογισμικό, λογισμικό εφαρμογών και δεδομένα εφαρμογών είναι τοποθετημένα στο πάροχο υπηρεσιών του κέντρου δεδομένων. Ο πάροχος υπηρεσιών διαχειρίζεται το υλικό και το λογισμικό και με την κατάλληλή συμφωνία υπηρεσιών, επιβεβαιώνει την διαθεσιμότητα και την ασφάλεια της εφαρμογής και των δεδομένων.

  Το λογισμικό βασισμένο σε νέφος προσφέρει στις εταιρίες από όλους τους τομείς έναν αριθμό από πλεονεκτήματα, συμπεριλαμβανομένου την ικανότητα να χρησιμοποιεί λογισμικό από οποιαδήποτε συσκευή είτε μέσω ενσωματωμένης εφαρμογής ή μέσω περιηγητή. Ως αποτέλεσμα, οι χρήστες μπορούν να φροντίζουν και να τοποθετούν τους φακέλους τους σε άλλες συσκευές με εύκολο τρόπο.

  Συνήθη σενάρια SaaS

  - Βασισμένη σε διαδίκτυο e-mail υπηρεσία π.χ. Outlook, Yahoo!, Hotmail

  Πλεονεκτήματα SaaS

  - Αποκτά πρόσβαση σε εξεζητημένες εφαρμογές
  - Πληρωμή μόνο για ότι χρησιμοποιεί ο χρήστης
  - Χρήση ελεύθερου λογισμικού πελάτη
  - Κινητοποιεί εύκολα το εργατικό δυναμικό
  - Πρόσβαση στα δεδομένα της εφαρμογής από οπουδήποτε

1.6.2 Πλεονεκτήματα και περιορισμοί υπολογιστικού νέφους

Μέσω των υπηρεσιών της υπολογιστικής νέφους [19], οι χρήστες μπορούν να ελέγχουν τις ειδοποιήσεις των e-mail τους σε οποιοδήποτε υπολογιστή ακόμη και να αποθηκεύουν τους φακέλους τους χρησιμοποιώντας υπηρεσίες όπως

Dropbox και Google Drive. Επίσης οι υπηρεσίες υπολογιστικής νέφους δίνουν τη δυνατότητα στους χρήστες να δημιουργούν αντίγραφο ασφαλείας σε περίπτωση crash του σκληρού δίσκου.

Επιπλέον η υπολογιστική νέφους προσφέρει μεγάλη πιθανότητα κοστολογικού κέρδους στις επιχειρήσεις. Προτού η υπολογιστική νέφους γίνει πραγματική εναλλακτική, οι εταιρίες ήταν υποχρεωμένες να αγοράσουν, κατασκευάσουν και διατηρήσουν δαπανηρά τεχνολογίες διαχείρισης πληροφοριών και την υποδομή. Οι εταιρίες μπορούν να ανταλλάξουν δαπανηρά κέντρα δεδομένων και τμήματα IT για γρήγορες συνδέσεις δικτύου, όπου οι υπάλληλοι αλληλεπιδρούν με τα νέφη μέσω διαδικτύου για να ολοκληρώσουν τις εργασίες τους.

Τέλος, η δομή της υπολογιστικής νέφους επιτρέπει στο κάθε χρήστη να αποθηκεύει αποθηκευτικό χώρο στον υπολογιστή του. Επιπροσθέτως, δίνει τη δυνατότητα στους χρήστες να μπορούν να αναβαθμίζουν το λογισμικό τους γρηγορότερα καθώς οι εταιρίες λογισμικού μπορούν να προσφέρουν τα προϊόντα της μέσω διαδικτύου σε σχέση με τον παραδοσιακό τρόπο (π.χ. δισκέτες, USB flash).

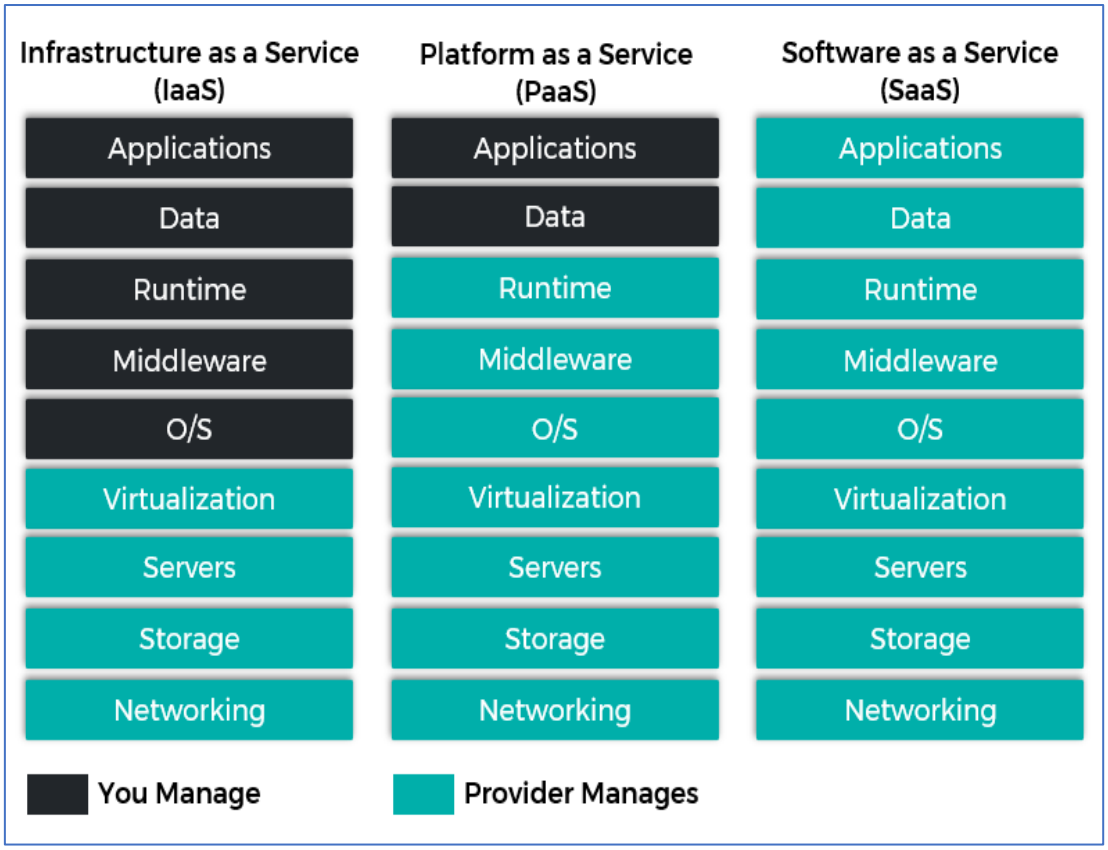
Ωστόσο, με την όλη ταχύτητα, τις αποδόσεις και καινοτομίες που ήρθαν με την εμφάνιση της υπολογιστικής νέφους, υπάρχουν φυσικά ρίσκα.

Η ασφάλεια πάντα αποτελούσε μεγάλη ανησυχία με το νέφος ειδικά όταν εισέρχεται σε ευαίσθητες ιατρικής φύσεως καταγραφές και πληροφορίες οικονομικών. Ενώ οι κανονισμοί αναγκάζουν τις υπηρεσίες της υπολογιστικής νέφους να ενισχύσουν την ασφάλεια τους και την προσαρμοστικότητα των μέτρων, παραμένει ένα τρέχων ζήτημα. Η κρυπτογράφηση προστατεύει σημαντικές πληροφορείς, εάν όμως το κλειδί κρυπτογράφησης χαθεί τα δεδομένα εξαφανίζονται.

Οι εξυπηρετητές που συντηρούνται από εταιρίες υπολογιστικής νέφους πιθανόν να πέσουν θύματα σε φυσικές καταστροφές, εσωτερικά σφάλματα και διακοπή ρεύματος. Η γεωγραφική έκταση της υπολογιστικής νέφους χωρίζει δυο μονοπάτια: Μια διακοπή ρεύματος στην Καλιφόρνια μπορεί να παραλύσει τους χρήστες στην Νέα Υόρκη, και μια εταιρία στο Τέξας να χάσει τα δεδομένα της εάν

κάτι την κύρια βάση του παρόχου να τερματίσει.

Καθώς με οποιαδήποτε τεχνολογία, υπάρχει μια καμπύλη εκμάθησης και για τους υπαλλήλους και για τους διαχειριστές. Αλλά καθώς και με πολλά άτομα να έχουν πρόσβαση και να χειρίζονται πληροφορίες μέσω μιας εισόδου, ακούσια λάθη μπορούν να διασχίσουν ολόκληρο το σύστημα.



Εικόνα 6. Σχηματική αναπαράσταση υπηρεσιών Cloud Computing

Πηγή: https://www.inap.com/blog/iaas-paas-saas-differences/

2. Το Πρωτόκολλο OpenFlow

Η αρχική ιδέα για το OpenFlow [21] ξεκίνησε στο Stanford University το 2008. Το Δεκέμβριο του 2009, η έκδοση 1.0 της προδιαγραφής του OpenFlow Switch κυκλοφόρησε. Από την ίδρυση του, το πρωτόκολλο αυτό διαχειρίζεται από τον ONF ενός μη κερδοσκοπικού οργανισμού αφιερωμένο στα ανοιχτά πρότυπα και την υιοθέτηση του SDN. Από τότε που κυκλοφόρησε το OpenFlow, πολλές εταιρίες και ανοιχτού κώδικα projects όπως το OpenDaylight υποστηρίζουν το OpenFlow και παρέχουν ακόμη OpenDaylight Controllers.

Το OpenFlow πρωτόκολλο θεωρείται ένα από τα πρώτα πρότυπα του SDN. Είναι αρχικά καθορισμένο το επικοινωνιακό πρωτόκολλο σε μια SDN αρχιτεκτονική που επιτρέπει στον SDN ελεγκτή την άμεση επικοινωνία με το επίπεδο προώθησης των δικτυακών συσκευών όπως μεταγωγείς και δρομολογητές, φυσικών ή εικονικών, ώστε να μπορεί να προσαρμόζεται καλύτερα στις μεταβαλλόμενες απαιτήσεις των επιχειρήσεων.

2.1 Αρχιτεκτονική OpenFlow

Η αρχιτεκτονική του OpenFlow αποτελεί μια σύνθεση τριών δομικών στοιχείων:

- OpenFlow ελεγκτής (Controller)
- OpenFlow μεταγωγέας (Switch)
- OpenFlow κανάλι (Channel)

2.1.1 OpenFlow Controller

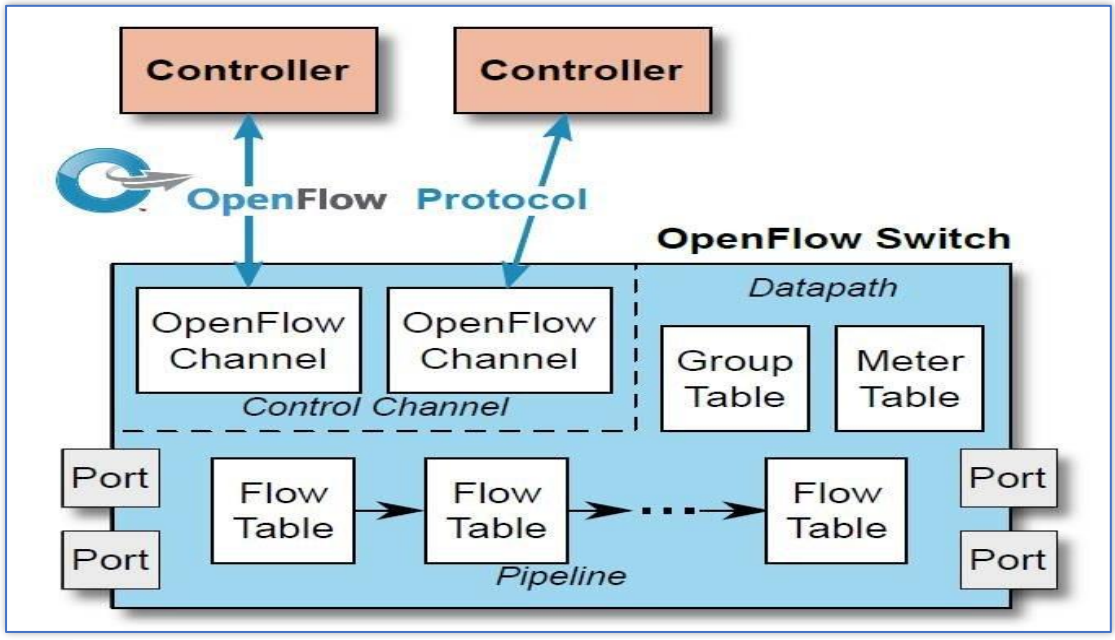
Ένας OpenFlow ελεγκτής είναι ένας τύπος SDN ελεγκτή που χρησιμοποιεί το OpenFlow πρωτόκολλο. Ο OpenFlow ελεγκτής χρησιμοποιεί το OpenFlow πρωτόκολλο για να συνδέσει και να διαμορφώσει δικτυακές συσκευές (δρομολογητές, μεταγωγείς) για να αποφασίσει το κατάλληλο μονοπάτι για την εφαρμογή κυκλοφορίας. Υπάρχουν επίσης και άλλα SDN πρωτόκολλα που ένας ελεγκτής μπορεί χρησιμοποιήσει όπως OpFlex, Yang και NetConf.

Οι SDN ελεγκτές μπορούν να απλοποιούν την διαχείριση του δικτύου,

μεταχειρίζοντας όλες τις επικοινωνίες ανάμεσα σε εφαρμογές και συσκευές για αποτελεσματική διαχείριση και τροποποίηση δικτυακών ροών για την αντιμετώπιση των μεταβαλλόμενων αναγκών. Όταν το επίπεδο ελέγχου του δικτύου είναι εφαρμοσμένο σε λογισμικό, οι διαχειριστές μπορούν να επιβλέπουν τη κυκλοφορία του δικτύου περισσότερο δυναμικά. Ένας SDN ελεγκτής μοιράζει πληροφορίες σε μεταγωγείς/δρομολογητές (μέσω southbound APIs) και εφαρμογές και επιχειρησιακή λογική (μέσω northbound APIs). Πιο συγκεκριμένα, ο OpenFlow ελεγκτής δημιουργεί ένα κεντρικό σημείο ελέγχου για την επίβλεψη ποικίλων OpenFlow ενεργών δικτυακών δομικών στοιχείων. Το OpenFlow πρωτόκολλο είναι σχεδιασμένο να αυξάνει την ευελιξία εξαλείφοντας ιδιόκτητα πρωτόκολλα από προμηθευτές υλικού.

2.1.2 OpenFlow Switch

Ένας μεταγωγέας OpenFlow [22] αποτελείται από έναν ή περισσότερους πίνακες ροών (flow tables), πίνακες ομαδοποίησης (group tables) και πίνακες μετρητές (meter tables), οι οποίοι εκτελούν αναζήτηση και προώθηση πακέτων, και ένα ή περισσότερα OpenFlow κανάλια σε έναν εξωτερικό ελεγκτή. Ο μεταγωγέας επικοινωνεί με τον ελεγκτή και ο ελεγκτής διαχειρίζεται τον μεταγωγέα μέσω του OpenFlow switch πρωτοκόλλου.



Εικόνα 7. Τυπική αρχιτεκτονική OpenFlow switch

Πηγή: https://www.researchgate.net/figure/OpenFlow-switch-atchitecture-An-OpenFlow-Switch-consists-of-one-or-more-flow-tables-and-a_fig4_320346909

Χρησιμοποιώντας το OpenFlow switch πρωτόκολλο, ο ελεγκτής μπορεί να προσθέσει, να ενημερώσει και να διαγράψει καταχωρήσεις ροής (flow entries) σε πίνακες ροής, διαδραστικά (ως απάντηση σε αιτήσεις πακέτων) ή προληπτικά. Κάθε πίνακας ροής στο μεταγωγέα περιέχει μια συλλογή καταχωρήσεων, κάθε μια αποτελείται από match fields, μετρητές (counters) και μια συλλογή οδηγιών για να εφαρμοστούν στα αντίστοιχα πακέτα.

Η αντιστοίχιση ξεκινάει στο πρώτο πίνακα ροής και μπορεί να συνεχίσει στους υπόλοιπους πίνακες ροής της διοχέτευσης (pipeline). Οι καταχωρήσεις ροών αντιστοιχίζουν τα πακέτα σε σειρά προτεραιότητας με την πρώτη αντίστοιχη καταχώρηση του σε κάθε πίνακα που χρησιμοποιείται. Σε περίπτωση που η αντίστοιχη καταχώρηση βρεθεί, οι οδηγίες που συνδέονται με συγκεκριμένες καταχωρήσεις ροής εκτελούνται. Εάν δε βρεθεί καμία αντιστοιχία σε ένα πίνακα ροής, το αποτέλεσμα εξαρτάται από τη διαμόρφωση του ελλιπούς πίνακα καταχώρησης ροής: για παράδειγμα, το πακέτο μπορεί να προωθηθεί στους ελεγκτές μέσω του OpenFlow καναλιού, να απορριφθεί ή να συνεχίσει στον επόμενο πίνακα ροής.

Οδηγίες που σχετίζονται με την εκάστοτε καταχώρηση ροής είτε περιέχουν λειτουργίες ή τροποποιούν την διαδικασία διοχέτευσης. Οι λειτουργίες περιγράφουν την προώθηση και την τροποποίηση των πακέτων, την επεξεργασία του πίνακα ροής. Οι οδηγίες που αναφέρονται στη διαδικασία της διοχέτευσης επιτρέπουν στα πακέτα να αποστέλλονται σε επόμενους πίνακες περαιτέρω επεξεργασία και επιτρέπει την ανταλλαγή πληροφορίας, με τη μορφή μεταδεδομένων (metadata), μεταξύ των πινάκων. Η διαδικασία της διοχέτευσης πινάκων σταματά όταν η συλλογή οδηγιών που σχετίζονται με μια αντίστοιχη καταχώρηση ροών δε προσδιορίζει έναν επόμενο πίνακα. Στο σημείο αυτό το πακέτο συνήθως τροποποιείται και προωθείται.

Οι καταχωρήσεις ροής μπορεί να προωθούνται σε μια θύρα η οποία συνήθως είναι φυσική (physical port) αλλά επίσης μπορεί να είναι λογική θύρα (logical port) ορισμένη από το μεταγωγέα ή μια κλειστή θύρα (reserved port). Οι κλειστές θύρες μπορούν να εκτελέσουν διαδικασίες προώθησης, όπως αποστολή στον ελεγκτή ή υπερχείλισης χρησιμοποιώντας μεθόδους χωρίς την χρήση του

OpenFlow, όπως δηλαδή μια συμβατική λειτουργία ενός μεταγωγέα.

Οι λειτουργίες σχετικά με τις καταχωρήσεις ροών, μπορούν επίσης να κατευθύνουν πακέτα σε πίνακες ομάδας (group tables), το οποίο ορίζει επιπλέον διαδικασία. Οι πίνακες ομάδας αναπαριστούν συλλογή λειτουργιών για υπερχείλιση (flooding), όπως επίσης περισσότερη πολύπλοκη προώθηση, παραδείγματος χάριν πολυδιόδευση πακέτων (multipath forwarding), γρήγορη επαναδρομολόγηση (fast reroute) ή συσσωμάτωση ζεύξεων (link aggregation). Ως ένα γενικό στρώμα εμμεσότητας, οι πίνακες ομάδας επιτρέπουν πολλαπλές καταχωρήσεις ροής να προωθούν σε ένα μοναδικό αναγνωριστικό (π.χ. προώθηση IP σε μία κοινή next hop).

Οι πίνακες ομάδας περιέχουν καταχωρήσεις ομάδας. Κάθε μια από αυτές περιέχει μια λίστα από action buckets με συγκεκριμένη σημασιολογία που εξαρτάται από τον τύπο της ομάδας. Οι λειτουργίες που περιέχονται σε ένα ή περισσότερα action buckets είναι εφαρμοσμένες σε πακέτα που αποστέλλονται σε πίνακες ομάδας.

Οι σχεδιαστές των μεταγωγέων μπορούν ελεύθερα να υλοποιήσουν τα εσωτερικά κατασκευαστικά στοιχεία με όποιο τρόπο επιθυμούν, υπό την προϋπόθεση ότι η κατάλληλη αντιστοίχιση και η σημασιολογία οδηγιών είναι προστατευμένα. Για παράδειγμα, ενώ μια καταχώρηση ροής μπορεί να χρησιμοποιεί όλους τους πίνακες ομάδας για να προωθήσει σε πολλαπλές θύρες, ένας σχεδιαστής μεταγωγέα μπορεί να επιλέξει να υλοποιήσει αυτό ως μοναδική μάσκα bit (bitmask) ανάμεσα στο πίνακα προώθησης υλικού.

Οι OpenFlow μεταγωγείς διακρίνονται σε δύο κατηγορίες: **OpenFlow-only** και **OpenFlow-hybrid**. Οι OpenFlow-only μεταγωγείς υποστηρίζουν μόνο την λειτουργία του OpenFlow πρωτοκόλλου, στους οποίους τα πακέτα επεξεργάζονται από την OpenFlow διασωλήνωση. Οι OpenFlow-hybrid μεταγωγείς υποστηρίζουν και τη λειτουργία του OpenFlow καθώς και του Ethernet πρωτοκόλλου. Οι μεταγωγείς αυτοί παρέχουν ένα μηχανισμό κατηγοριοποίησης εκτός του OpenFlow που δρομολογεί την κυκλοφορία είτε στην OpenFlow διασωλήνωση είτε στην παραδοσιακή διασωλήνωση. Για παράδειγμα, ένας μεταγωγέας μπορεί να

χρησιμοποιεί VLAN ετικέτα (tag) ή θύρα εισόδου του πακέτου για να αποφασίσει πότε να επεξεργασθεί το πακέτο χρησιμοποιώντας τη μια διασωλήνωση ή την άλλη, ή μπορεί να κατευθύνει όλα τα πακέτα στην OpenFlow διασωλήνωση.

Το OpenFlow ορίζει τρία είδη OpenFlow ports που υποστηρίζονται από το OpenFlow πρωτόκολλο όπως προαναφέρθηκε προηγουμένως σε ποιες θύρες μπορούν να προωθηθούν οι καταχωρήσεις ροής.

Φυσική πόρτα (Physical port): Μια φυσική πόρτα αντιστοιχεί σε μια διεπαφή υλικού του μεταγωγέα**.** Για παράδειγμα, σε ένα μεταγωγέα Ethernet, οι φυσικές πόρτες αντιστοιχούν μια προς μια τις διεπαφές Ethernet. Σε κάποιες αναπτύξεις, ο OpenFlow μεταγωγέας μπορεί να εικονικοποιηθεί πάνω από το μεταγωγέα υλικού. Στη περίπτωση αυτή, μια OpenFlow φυσική πόρτα μπορεί να απεικονίσει ένα εικονικό τμήμα της αντίστοιχης διεπαφής του υλικού του μεταγωγέα.

Λογική πόρτα (Logical port): Οι λογικές πόρτες δεν αντιστοιχούν απευθείας σε μια πόρτα υλικού του μεταγωγέα. Είναι υψηλότερου επιπέδου έννοιες που μπορεί να ορίζονται στο μεταγωγέα χρησιμοποιώντας ανεξάρτητες από το OpenFlow μεθόδους (π.χ. link aggregation groups, tunnels, loopback interfaces).

Κλεισμένη πόρτα (Reserved port): Οι κλειστές πόρτες ορίζουν γενική προώθηση λειτουργιών όπως αποστολή στον ελεγκτή, υπερχείλιση ή προώθηση χρησιμοποιώντας μεθόδους ανεξάρτητες από το OpenFlow. Ένας μεταγωγέας δεν απαιτεί να υποστηρίζει όλες τις κλειστές πόρτες, μόνο αυτές που απαιτούνται. Οι υποχρεωτικές πόρτες χωρίζονται σε πέντε κατηγορίες:

ALL: Αναπαριστά όλες τις πόρτες που ένας μεταγωγέας μπορεί να χρησιμοποιήσει για να προωθήσει ένα συγκεκριμένο πακέτο. Μπορούν να χρησιμοποιηθούν μόνο ως εξωτερικές πόρτες.

CONTROLLER: Αναπαριστά το κανάλι ελέγχου με τους ελεγκτές OpenFlow. Μπορεί να χρησιμοποιηθεί ως μια πόρτα εισόδου ή ως μια εξόδου.

TABLE: Αναπαριστά την αρχή της διοχέτευσης OpenFlow.

IN_PORT: Αναπαριστά την πόρτα εισόδου. Μπορεί να χρησιμοποιηθεί επίσης και ως πόρτα εξόδου στέλνοντας το πακέτο μέσω την πόρτας εισόδου που προήλθε.

ANY: Ειδική αξία χρησιμοποιείται σε μερικά OpenFlow αιτήματα όταν καμία πόρτα δεν έχει προσδιοριστεί. Κάποια αιτήματα OpenFlow περιέχουν μια αναφορά σε μια συγκεκριμένη πόρτα που το αίτημα εφαρμόζει**.** Χρησιμοποιώντας την ANY ως αριθμός πόρτας στα αιτήματα αυτά επιτρέπει πως αιτήματα να εφαρμόζονται σε οποιαδήποτε πόρτα. Μπορεί να χρησιμοποιηθεί είτε ως πόρτα εισόδου είτε ως εξόδου.

Επίσης υπάρχουν και οι προαιρετικές πόρτες οι οποίες χωρίζονται σε τρείς κατηγορίες:

LOCAL: Αναπαριστά τη στοίβα του τοπικού δικτυακού του μεταγωγέα και της διαχείρισης της στοίβας του. Μπορεί να χρησιμοποιηθεί ως πόρτα εισόδου ή ως πόρτα εξόδου. Η LOCAL πόρτα επιτρέπει απομακρυσμένες οντότητες να αλληλοεπιδρούν με το μεταγωγέα και των υπηρεσιών του διαδικτύου της μέσω του OpenFlow διαδικτύου.

NORMAL: Αναπαριστά τη προώθηση χρησιμοποιώντας την παραδοσιακή διοχέτευση του μεταγωγέα που είναι ανεξάρτητη από το OpenFlow. Μπορεί να χρησιμοποιηθεί ως πόρτα εξόδου και επεξεργάζεται τα πακέτα χρησιμοποιώντας την κανονική αρχιτεκτονική.

FLOOD: Αναπαριστά την υπερχείλιση χρησιμοποιώντας την παραδοσιακή διοχέτευση του μεταγωγέα που δεν βασίζεται στο OpenFlow πρωτόκολλο. Μπορεί να χρησιμοποιηθεί μόνο ως πόρτα εξόδου.

2.1.3 OpenFlow Channel

Το OpenFlow κανάλι είναι η διεπαφή που συνδέει κάθε OpenFlow μεταγωγέα με έναν OpenFlow ελεγκτή. Μέσω της διεπαφής αυτής, ο ελεγκτής ρυθμίζει και διαχειρίζεται τα πακέτα που ανταλλάσσει με τον μεταγωγέα. Το κανάλι αυτό συνήθως είναι κρυπτογραφημένο μέσω του TLS πρωτοκόλλου αλλά μπορεί να τρέχει απευθείας μέσω της TCP σύνδεσης.

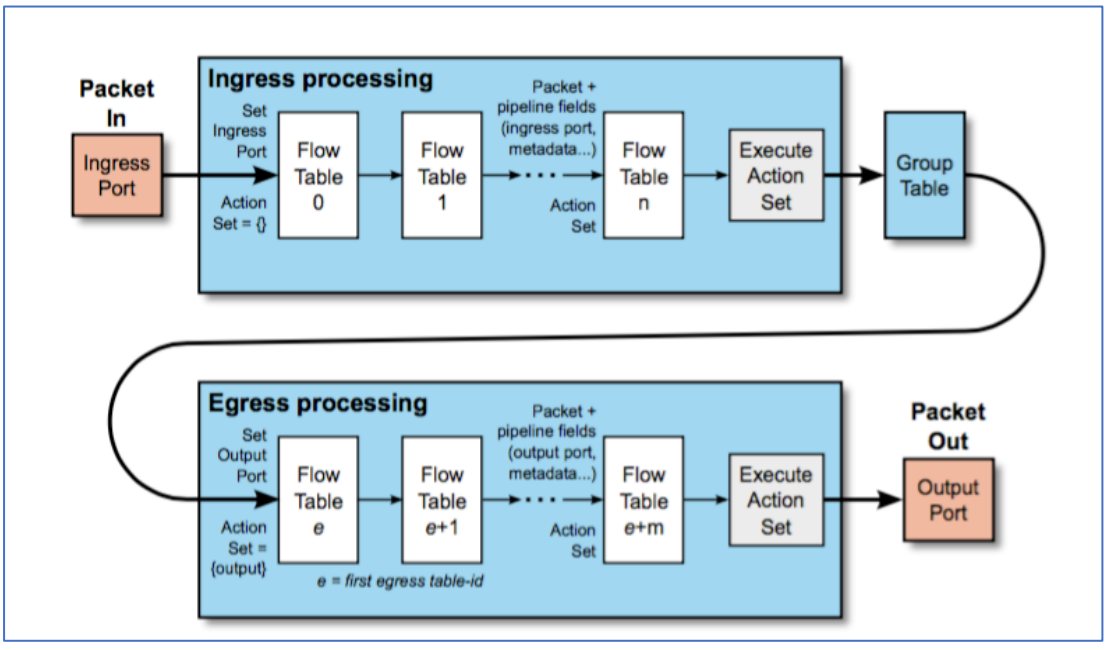
Όπως προαναφέρθηκε προηγουμένως, το OpenFlow κανάλι χρησιμοποιείται για να ανταλλάσσει OpenFlow μηνύματα ανάμεσα σε έναν OpenFlow μεταγωγέα και ένα OpenFlow ελεγκτή. Ένα τυπικό μήνυμα OpenFlow ελεγκτή διαχειρίζεται

πολλαπλά OpenFlow κανάλια, κάθε ένα για διαφορετικό OpenFlow μεταγωγέα. Ένας OpenFlow μεταγωγέας μπορεί να έχει ένα OpenFlow κανάλι σε ένα μοναδικό ελεγκτή, ή πολλαπλά κανάλια για αξιοπιστία, κάθε ένα για διαφορετικό ελεγκτή.

Ένας OpenFlow ελεγκτής τυπικά διαχειρίζεται έναν OpenFlow μεταγωγέα απομακρυσμένα μέσω ενός ή περισσότερων δικτύων. Το OpenFlow κανάλι είναι συνήθως ενσαρκωμένο ως μοναδική διαδικτυακή σύνδεση ανάμεσα στο μεταγωγέα και τον ελεγκτή, χρησιμοποιώντας το TLS πρωτόκολλο ή TCP σύνδεση. Εναλλακτικά, το OpenFlow κανάλι μπορεί να αποτελείται από πολλαπλές διαδικτυακές συνδέσεις για να αξιοποιήσει το παραλληλισμό. Ο OpenFlow μεταγωγέας πρέπει να είναι ικανός να δημιουργεί ένα OpenFlow κανάλι ξεκινώντας μια σύνδεση σε ένα OpenFlow μεταγωγέα. Μερικές υλοποιήσεις μεταγωγέων μπορεί προαιρετικά να επιτρέπουν σε ένα OpenFlow ελεγκτή να συνδεθεί με τον OpenFlow μεταγωγέα, στην περίπτωση αυτή ο μεταγωγέας περιορίζεται για να ασφαλίσει τις συνδέσεις εμποδίζοντας έτσι μη εξουσιοδοτημένες συνδέσεις.

2.2 OpenFlow πίνακες

Στην ενότητα αυτή περιγράφονται τα δομικά στοιχεία του πίνακα ροής, πίνακα ομάδας και πίνακα meter, μαζί με τους μηχανισμούς αντιστοίχισης και των λειτουργιών χειρισμού.



Εικόνα 8. Ροή πακέτου μέσω της διαδικασίας της διοχέτευσης

Πηγή: https://wiki.onosproject.org/display/ONOS/OpenFlow+1.5+Implementation

2.2.1 Flow Tables

Ένας πίνακας ροής αποτελείται από καταχωρήσεις ροών όπως φαίνεται στο παρακάτω πίνακα.

Πίνακας 2. Δομή των Flow entries σε ένα Flow Table

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Κάθε καταχώρηση του πίνακα ροής περιέχει:

- **Match fields:** Τα πεδία αντιστοίχισης πακέτων. Αυτά αποτελούνται από την πόρτα εισόδου και τις επικεφαλίδες των πακέτων, και προαιρετικά άλλα πεδία διοχέτευσης όπως μεταδεδομένα προσδιορισμένα από τον προηγούμενο πίνακα.
- **Priority**: Πεδίο αντιστοίχισης προτεραιότητας της καταχώρησης ροής.
- **Counters:** Ενημερώνονται όταν τα πακέτα είναι αντιστοιχισμένα.
- Instructions: Τροποποιούν μια συλλογή λειτουργιών ή διαδικασία διοχέτευσης.
- **Timeouts:** Μέγιστη ποσότητα χρόνου ή αδρανής χρόνος πριν την εκπνοή της ροής από το μεταγωγέα.
- **Cookie:** Αδιαφανής τύπος δεδομένων επιλεγμένος από τον ελεγκτή. Μπορεί να χρησιμοποιηθεί από τον ελεγκτή για να φιλτράρει καταχωρήσεις ροών επηρεασμένες από στατιστικές ροών, τροποποιήσεις ροών και αιτήματα ανίχνευσης ροών.
- **Flags:** Οι ετικέτες αλλάζουν τον τρόπο που διαχειρίζονται οι καταχωρήσεις ροών, για παράδειγμα η ετικέτα OFPFF_SEND_FLOW_REM προξενεί ροή απομακρυσμένων μηνυμάτων για αυτή την καταχώρηση ροής.

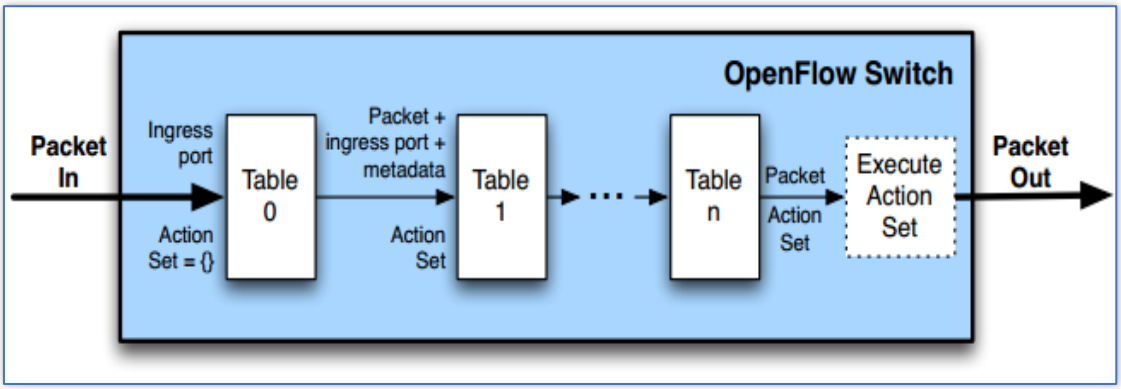
Η διαδικασία της διοχέτευσης OpenFlow ορίζει πόσα πακέτα αλληλοεπιδρούν με τους πίνακες ροής. Ένας OpenFlow μεταγωγέας απαιτεί να έχει τουλάχιστον μια καταχώρηση εισόδου και μπορεί προαιρετικά να έχει περισσότερους πίνακες ροής. Οι πίνακες ροής ενός OpenFlow μεταγωγέα είναι ακολουθιακά αριθμημένοι, ξεκινώντας από το μηδέν. Η διαδικασία της διοχέτευσης συμβαίνει σε δύο στάδια, διαδικασία εισόδου (ingress processing) και διαδικασία

εξόδου (egress processing). Ο διαχωρισμός των δύο σταδίων υποδεικνύεται από το πρώτο πίνακα εξόδου, όλοι οι ροών του πίνακα ροής, και οι υπόλοιποι πίνακες ροής εξόδου μπορεί να χρησιμοποιηθούν βασιζόμενοι στο αποτέλεσμα της αντιστοίχισης του πίνακα πίνακες με αριθμό χαμηλότερο του πρώτου πίνακα εξόδου πρέπει να χρησιμοποιούνται ως πίνακες εισόδου, και κανένας πίνακας με μεγαλύτερο αριθμό ή ίσο με το πρώτο πίνακα εξόδου μπορεί να χρησιμοποιηθεί ως πίνακας εισόδου. Η διαδικασία αυτή πάντα ξεκινά με την διαδικασία της εισόδου στο πρώτο πίνακα ροής. Το πακέτο πρέπει να είναι αντιστοιχισμένο με τις καταχωρήσεις ροών του πίνακα ροής 0. Η διαδικασία της εξόδου είναι προαιρετική, ένας μεταγωγέας μπορεί να μην υποστηρίζει πίνακες εξόδου ή μπορεί να μην είναι κατάλληλα διαμορφωμένος να τους χρησιμοποιεί. Σε περίπτωση που κανένας έγκυρος πίνακας εξόδου είναι ρυθμισμένος όπως ο πρώτος πίνακας εξόδου, το πακέτο πρέπει να επεξεργασθεί από τη θύρα εξόδου. Από την άλλη, σε περίπτωση που ένας έγκυρος πίνακας εξόδου είναι ρυθμισμένος όπως ο πρώτος πίνακας εξόδου, το πακέτο πρέπει να αντιστοιχηθεί με τις καταχωρήσεις ροής.

Όταν επεξεργασθεί από το πίνακα ροής, το πακέτο που έχει αντιστοιχηθεί με τις καταχωρήσεις του πίνακα ροής για την επιλογή μιας καταχώρησης ροής. Σε περίπτωση που μια καταχώρηση ροής βρεθεί, η συλλογή οδηγιών συμπεριλαμβανομένου και την καταχώρηση ροής εκτελείται. Αυτές οι οδηγίες μπορεί να κατευθύνουν το πακέτο σε ένα άλλο πίνακα ροής, όπου η ίδια διαδικασία επαναλαμβάνεται (εντολές Goto). Μια καταχώρηση ροής μπορεί μόνο να κατευθύνει ένα πακέτο σε ένα πίνακα ροής με αριθμό που είναι μεγαλύτερος από του δικού του πίνακα ροής αριθμό, καθώς η διαδικασία της διοχέτευσης μπορεί να λειτουργεί μόνο με την προώθηση και όχι προς τα πίσω. Εμφανώς, οι καταχωρήσεις ροών του τελευταίου πίνακα της διοχέτευσης δεν μπορούν να περιλαμβάνουν οδηγίες Goto. Σε περίπτωση που η αντιστοίχιση της καταχώρησης ροής δεν κατευθύνει τα πακέτα σε άλλο πίνακα ροής, η τρέχουσα διαδικασία της διοχέτευσης σταματά στο υπάρχον πίνακα, το πακέτο επεξεργάζεται με τις συνδεδεμένη συλλογή λειτουργιών και συνήθως προωθείται.

Σε περίπτωση που ένα πακέτο δεν αντιστοιχεί μια καταχώρηση ροής σε ένα πίνακα ροής, αυτό καλείται αστοχία πίνακα (table miss). Η συμπεριφορά σε ένα

πίνακα αστοχίας εξαρτάται από τη ρύθμιση του πίνακα. Οι οδηγίες που συμπεριλαμβάνονται σε ένα πίνακα αστοχίας καταχώρησης ροής σε ένα πίνακα ροής μπορούν με ευελιξία να προσδιορίσουν πως να επεξεργαστούν μη αντιστοιχισμένα πακέτα, χρήσιμες επιλογές συμπεριλαμβάνοντας την απόρριψη πακέτου, προώθηση σε άλλο πίνακα ή αποστολή στον ελεγκτή μέσω του καναλιού ελέγχου.



Εικόνα 9. Σχηματική αναπαράσταση ροής πακέτου μέσω της διαδικασίας της διοχέτευσης

Πηγή: https://suhu0426.github.io/Web/Presentation/20141118/index.html

2.2.2 Group Tables

Ένας πίνακας ομάδας αποτελείται από καταχωρήσεις ομάδας. Η ικανότητα για μια καταχώρηση ροής να υποδεικνύει σε μια ομάδα επιτρέπει στο OpenFlow να αναπαριστά επιπλέον μεθόδους προώθησης.

Κάθε καταχώρηση ομάδας αναγνωρίζεται από το αναγνωριστικό ομάδας της και περιέχει:

- **Group identifier:** Ένας 32 Bit μη προσημασμένος ακέραιος αριθμός που αποκλειστικά αναγνωρίζει την ομάδα πακέτου σε ένα OpenFlow μεταγωγέα.
- **Group type:** Τύπος της ομάδας που καθορίζει τη σημασιολογία της ομάδας.
- **Counters:** Μετρητές που φορτώνονται όταν τα πακέτα έχουν επεξεργασθεί από μια ομάδα.
- **Action buckets:** Μια ταξινομημένη λίστα από (εντολές) action buckets, όπου κάθε action bucket αποτελείται από μια συλλογή λειτουργιών για να εκτελεστεί και τις σχετικές παραμέτρους.

Πίνακας 3. Βασικά δομικά στοιχεία μιας καταχώρησης ομάδας σε ένα Group Table

Group identifier	Group type	Counters	Action buckets

Ένας μεταγωγέας δεν απαιτεί να υποστηρίζει όλα τα group types, μόνο όσα είναι υποχρεωτικά. Ο ελεγκτής μπορεί επίσης να αιτηθεί στο μεταγωγέα για ποια από τα προαιρετικά group types υποστηρίζει. Τα group types κατατάσσονται σε τέσσερις κατηγορίες:

- **Indirect (υποχρεωτικό):** Εκτελεί μια προκαθορισμένη εντολή σε αυτό το πίνακα ομάδας. Αυτή η ομάδα υποστηρίζει μόνο μια μοναδική εντολή. Επιτρέπει πολλαπλές καταχωρήσεις ροών ή πίνακες να υποδεικνύουν ένα κοινό αναγνωριστικό ομάδας, υποστηρίζοντας γρηγορότερα, και περισσότερο αποτελεσματική προσέγγιση. Αυτός ο τύπος ομάδας είναι αποτελεσματικά ίδιος για όλες τις ομάδες με μια εντολή. Είναι ο απλούστερος τύπος και συνεπώς οι μεταγωγείς θα μπορούν να υποστηρίζουν μεγαλύτερο αριθμό από αυτούς σε σχέση με άλλους τύπους ομάδας.

- **All (υποχρεωτικό):** Εκτελούνται όλες οι εντολές στην ομάδα. Αυτή η ομάδα χρησιμοποιείται για πολυεκπομπή (multicast) ή ευρυεκπομπή (broadcast) προώθησης. Το πακέτο είναι αποτελεσματικά κλωνοποιημένο και επεξεργάζεται για κάθε εντολή στο action bucket. Σε περίπτωση που μια εντολή κατευθύνει ένα πακέτο έξω από τη θύρα εισόδου, αυτό το πακέτο κλώνος απορρίπτεται. Εάν ο προγραμματιστής του ελεγκτή επιθυμεί να προωθήσει το πακέτο έξω από τη θύρα εισόδου, η ομάδα εντολών πρέπει να συμπεριλάβει μια επιπλέον εντολή εξόδου για την OFPP_IN_PORT κλεισμένη θύρα.

- **Select (προαιρετικό):** Εκτελεί μια εντολή στο πίνακα ομάδας**.** Τα πακέτα επεξεργάζονται από μια μοναδική εντολή στο πίνακα ομάδας, βασισμένα σε μια επιλογή αλγορίθμου μιας συσκευής. Όλη η ρύθμιση για την επιλογή αλγορίθμου είναι εξωτερικό για το OpenFlow. Η επιλογή αλγορίθμου πρέπει να εφαρμόζει ίσο μοίρασμα φόρτου και να μπορεί προαιρετικά να βασίζεται στα φορτία εντολών. Όταν μια θύρα προσδιορίζεται σε μια εντολή σε μια

επιλεγμένη ομάδα, ο μεταγωγέας μπορεί να περιορίζει την επιλογή εντολής για την υπόλοιπη συλλογή αντί να απορρίψει τα πακέτα που προορίζονται για αυτή τη θύρα.

- **Fast failover (προαιρετικό):** Εκτελείται η πρώτη εν ενεργεία εντολή. Κάθε συλλογή εντολών συσχετίζεται με μια συγκεκριμένη θύρα ή ομάδα που ελέγχει την ζωτικότητα της. Τα πακέτα αξιολογούνται με σκοπό να προσδιοριστούν από την ομάδα, και η πρώτη εντολή που συσχετίζεται με μια εν ενεργεία θύρα/ομάδα επιλέγεται. Αυτός ο τύπος ομάδας επιτρέπει στον μεταγωγέα να αλλάξει την προώθηση χωρίς να απαιτήσει το χρόνο μετάδοσης στον ελεγκτή. Σε περίπτωση που καμία εντολή δεν είναι σε εν ενεργεία, τα πακέτα απορρίπτονται. Αυτός ο τύπος ομάδας μπορεί να εφαρμόσει ένα μηχανισμό ζωτικότητας.

2.2.3 Meter Tables

Ένας πίνακας meter αποτελείται από meter καταχωρήσεις, ορίζοντας ανά ροή meters. Η ανά ροή meters επιτρέπουν στο OpenFlow να εφαρμόσει διάφορες απλές λειτουργίες του OpenFlow, όπως περιορισμός ροής πακέτων.

Κάθε καταχώρηση meter είναι αναγνωρισμένη από το δικό της meter αναγνωριστικό και περιλαμβάνει:

- **Meter identifier**: Ένας 32 Bit μη προσημασμένος αριθμός αναγνωριστικός του πίνακα meter.
- **Meter bands**: Μια λίστα από meter bands, όπου το κάθε meter band καθορίζει το ρυθμό και την επεξεργασία του πακέτου.
- **Counters**: Ενημερώνονται όταν τα πακέτα είναι επεξεργασμένα από ένα meter.

Πίνακας 4. Βασικά δομικά στοιχεία μιας καταχώρησης ροής σε ένα Meter Table

Meter identifier	Meter bands	Counters

Κάθε πίνακας Meter μπορεί να έχει ένα ή περισσότερα meter bands. Κάθε band

καθορίζει το ρυθμό στο οποίο το band εφαρμόζει και τον τρόπο που τα πακέτα πρέπει να επεξεργάζονται. Τα πακέτα επεξεργάζονται από ένα μοναδικό meter band βασισμένο στο τωρινό meter ρυθμό. Το meter εφαρμόζει στο meter band με το μεγαλύτερο ρυθμισμένο ρυθμό όπου είναι μικρότερος από τον τωρινό υπολογισμένο ρυθμό. Σε περίπτωση που ο τωρινός ρυθμός είναι χαμηλότερος από οποιοδήποτε καθορισμένο meter band ρυθμό, κανένα meter band εφαρμόζεται.

Κάθε meter band αναγνωρίζεται από το ρυθμό του και περιέχει:

- **Band type**: Ορίζει πόσα πακέτα έχουν επεξεργασθεί.
- **Rate**: Χρησιμοποιείται από το meter για να επιλέξει το meter band, ορίζει το χαμηλότερο ρυθμό στον οποίο το band μπορεί να εφαρμόσει.
- **Burst**: Ορίζει την πιστότητα του meter band.
- **Counters**: Ενημερώνονται όταν τα πακέτα έχουν επεξεργασθεί από ένα meter band.
- **Type specific arguments**: Κάποια band types έχουν προαιρετικά εγχειρήματα. Δεν υπάρχει band type που να απαιτείται από αυτή την προδιαγραφή. Ο ελεγκτής μπορεί να αιτηθεί στο μεταγωγέα για ποια από τα προαιρετικά meter band types υποστηρίζει:
  - **Drop** (προαιρετικό): Απόρριψη πακέτου.
  - **DSCP remark** (προαιρετικό): Αυξάνει την προτεραιότητα απόρριψης του DSCP πεδίου στο IP header του πακέτου.

2.3 OpenFlow μηνύματα

Το OpenFlow πρωτόκολλο υποστηρίζει τρία είδη μηνυμάτων, ελεγκτή-σε-μεταγωγέα (controller-to-switch), ασύγχρονα (asynchronous) και συμμετρικά (symmetric). Το καθένα υποστηρίζει πολλαπλούς υπο-τύπους. Τα μηνύματα ελεγκτή-σε-μεταγωγέα ξεκινούν από τον ελεγκτή και χρησιμοποιούνται ώστε να διαχειρίζονται ή να επιθεωρούν άμεσα την κατάσταση του μεταγωγέα. Τα ασύγχρονα μηνύματα ξεκινάνε από τον μεταγωγέα και χρησιμοποιούνται για να ενημερώνουν τον ελεγκτή για γεγονότα στο δίκτυο και αλλαγές στη κατάσταση του

μεταγωγέα. Τα συμμετρικά μηνύματα ξεκινάνε είτε από τον μεταγωγέα ή από τον ελεγκτή και αποστέλλονται χωρίς πρόσκληση.

Controller-to-Switch

Τα μηνύματα που ξεκινάνε από τον ελεγκτή και μπορεί να απαιτούν ή όχι μια απάντηση από τον μεταγωγέα. Ο ελεγκτής μπορεί να ζητήσει την ταυτότητα και τις βασικές δυνατότητες του μεταγωγέα στέλνοντας ένα αίτημα για τις λειτουργίες. Ο μεταγωγέας πρέπει να απαντήσει με μια χακτηριστική απάντηση για την ταυτότητα και τις βασικές δυνατότητες του μεταγωγέα. Αυτό συνήθως εμφανίζεται με την ίδρυση του OpenFlow καναλιού.

Asynchronous

Τα ασύγχρονα μηνύματα στέλνονται χωρίς ο ελεγκτής να προσελκύεται από ένα μεταγωγέα. Οι μεταγωγείς στέλνουν ασύγχρονα μηνύματα στους ελεγκτές για να δηλώσουν την άφιξη ενός πακέτου ή να δηλώσουν αλλαγή στη κατάσταση του μεταγωγέα. Οι κύριοι τύποι ασύγχρονων μηνυμάτων περιγράφονται παρακάτω:

- **Packet-in**: Μεταφέρει τον έλεγχο ενός πακέτου στον ελεγκτή. Για όλα τα πακέτα που προωθούνται στον ελεγκτή οι κλειστές θύρες χρησιμοποιούν μια καταχώρηση ροής ή ενός πίνακα-αστοχίας καταχώρηση ροής, ένα packet-in γεγονός πάντα στέλνεται στους ελεγκτές. Άλλη διαδικασία, όπως για παράδειγμα έλεγχος TTL, μπορεί επίσης να δημιουργήσει packet-in γεγονότα για να στείλει πακέτα στον ελεγκτή.

- **Flow-removed**: Πληροφορεί τον ελεγκτή για την απομάκρυνση μιας καταχώρησης ροής από ένα πίνακα ροής. Τα μηνύματα απομάκρυνση ροής (Flow-removed) στέλνονται μόνο για καταχωρήσεις ροών με OFPFF_SEND_FLOW_REM ετικέτα. Τα μηνύματα αυτά δημιουργούνται ως το αποτέλεσμα διαγραφής αιτήματος μιας καταχώρησης ελεγκτή ή λήξη διαδικασίας της ροής του ελεγκτή όταν ένα από τα χρονικά όρια της ροής υπερβεί.

- **Port-status**: Πληροφορεί τον ελεγκτή για μια αλλαγή σε μια θύρα. Ο μεταγωγέας αναμένει να στείλει μηνύματα port-status (κατάσταση-θύρας)

στους ελεγκτές ως διαμόρφωση θύρας ή αλλαγές κατάστασης θύρας.

- **Role-status**: Πληροφορεί τον ελεγκτή για αλλαγή του ρόλου του. Όταν ένας ελεγκτής εκλεγεί από μόνος του κύριος, ο μεταγωγέας αναμένει να στείλει μηνύματα role-status (κατάσταση ρόλου) στον προηγούμενο κύριο ελεγκτή.
- **Controller-status**: Πληροφορεί τον ελεγκτή όταν η κατάσταση ενός OpenFlow καναλιού αλλάξει. Ο μεταγωγέας στέλνει αυτά τα μηνύματα σε όλους τους ελεγκτές όταν η κατάσταση του OpenFlow καναλιού σε οποιεσδήποτε αλλαγές στους μεταγωγείς.
- **Flow-monitor**: Πληροφορεί των ελεγκτή για μια αλλαγή στο πίνακα ροής. Ένας ελεγκτής μπορεί να ορίσει μια συλλογή από επιτηρητές για να παρακολουθήσει αλλαγές στους πίνακες ροής.

Symmetric

Τα συμμετρικά μηνύματα στέλνονται χωρίς πρόσκληση, προς κάθε κατεύθυνση. Τα μηνύματα αυτά χωρίζονται σε τέσσερις κατηγορίες:

Hello: Τα μηνύματα Hello ανταλλάσσονται ανάμεσα στο μεταγωγέα και τον ελεγκτή όταν η επικοινωνία αρχίσει.

Echo: Τα μηνύματα Echo αίτησης/απάντησης μπορούν να σταλούν είτε από το μεταγωγέα ή από τον ελεγκτή, και πρέπει να επιστρέψουν μια echo απάντηση. Τα μηνύματα αυτά χρησιμοποιούνται για να επιβεβαιώσουν την ζωτικότητα της επικοινωνίας ενός ελεγκτή-μεταγωγέα, όπως και χρησιμοποιούνται για να μετρήσουν την καθυστέρηση ή το εύρος ζώνης τους.

Error: Τα μηνύματα Error χρησιμοποιούνται από τον μεταγωγέα ή τον ελεγκτή για να ενημερώσουν για προβλήματα για την άλλη πλευρά της επικοινωνίας. Συνήθως χρησιμοποιούνται από το μεταγωγέα για να δείξει ένα σφάλμα ενός αιτήματος που ξεκινάει από τον ελεγκτή.

Experimenter: Τα μηνύματα Experimenter παρέχουν ένα συγκεκριμένο τρόπο για τους OpenFlow μεταγωγείς για να προσφέρουν επιπλέον λειτουργικότητα μέσα στο χώρο του OpenFlow μηνύματος.

3. Ο ελεγκτής OpenDaylight SDN Controller

Το OpenDaylight Project (ODL) είναι ένα συνεργατικό έργο ανοιχτού κώδικα λογισμικό που φιλοξενείτε από το Linux Foundation. Το έργο χρησιμεύει ως πλατφόρμα για δικτύωση που καθορίζεται από λογισμικό (SDN) για ανοιχτή, κεντρική παρακολούθηση συσκευών δικτύου.

Στις 8 Απριλίου 2013 το Linux Foundation ανακοίνωσε την ίδρυση του OpenDaylight Project. Ο στόχος του έργου είναι να δημιουργήσει μια πλατφόρμα ανοιχτού κώδικα που διαχειρίζεται από τη κοινότητα και υποστηρίζεται από τη βιομηχανία για να επιταχύνει την υιοθέτηση και τη καινοτομία του SDN και του NFV. Τα ιδρυτικά μέλη του έργου αυτού ήταν η Big Switch Networks, Brocade, Cisco, Citrix, Ericsson, IBM, Juniper Networks, Microsoft, NEC, Red Hat και VMware. Έως τώρα υπάρχουν 13 κυκλοφορίες του ODL Controller που στηρίζουν το όνομά τους στον ατομικό αριθμό χημικών στοιχείων του περιοδικού πίνακα καθώς και τον αντίστοιχο αριθμού κυκλοφορίας τους: Hydrogen (Φεβρουάριος 2014), Helium (Οκτώβριος 2014), Lithium (Ιούνιος 2015), Beryllium (Φεβρουάριος 2016), Boron (Νοέμβριος 2016), Carbon (Ιούνιος 2017), Nitrogen (Σεπτέμβριος 2017), Oxygen (Μάρτιος 2018), Fluorine (Αύγουστος 2018), Neon (Μάρτιος 2019), Sodium (Σεπτέμβριος 2019), Magnesium (Μάρτιος 2020) και Aluminium (Σεπτέμβριος 2020).

Ο OpenDaylight Controller είναι αποκλειστικά εφαρμοσμένος σε λογισμικό και συντηρείται στο δικό του Java Virtual Machine (JVM) με αποτέλεσμα να μπορεί

να αναπτυχθεί σε οποιοδήποτε λειτουργικό σύστημα ή υλικό που υποστηρίζει Java. Ο ODL Controller κάνει επίσης χρήση των ακόλουθων εργαλείων [23]:

- **Maven**: Ο OpenDaylight χρησιμοποιεί το εργαλείο Maven για ευκολότερη αυτοματοποίηση της ανάπτυξης του κώδικα. Το Maven χρησιμοποιεί το pom.xml (Project Object Model) για να οργανώσει τις εξαρτήσεις ανάμεσα στο πακέτο και στη συνέχεια να περιγράψει τι πακέτα θα φορτωθούν και θα ξεκινήσουν.

- **OSGi container**: Το OSGi (Open Service Gateway Initiative) [24] [25] είναι ένα πλαίσιο, γνωστό και ως Dynamic Module System για Java, ορίζει μια αρχιτεκτονική για ανάπτυξη αρθρωτής εφαρμογής. Επιτρέπει τον διαχωρισμό των εφαρμογών σε πολλαπλά τμήματα ώστε να είναι ευκολότερη η διαχείριση των εξαρτήσεων ανάμεσα τους. Τα πακέτα OSGi είναι αρχεία τύπου .JAR με ένα MANIFEST.MF αρχείο περιέχοντας συγκεκριμένες OSGi κεφαλές (headers). Η πλατφόρμα του OSGi παρέχει ένα τρόπο να λαμβάνει ειδοποιήσεις για το πότε τα πακέτα είναι διαθέσιμα ή πότε αφαιρούνται από τη πλατφόρμα. Όταν ένα πακέτο είναι εξαρτημένο από τα άλλα πακέτα OSGi, πρώτα θα ξεκινήσουν αυτές οι εξαρτήσεις και στη συνέχεια το πακέτο από μόνο του, ειδάλλως το πακέτο δε θα ξεκινήσει. Ως αποτέλεσμα, ένας χρήστης μπορεί να ξεκινήσει, σταματήσει, εγκαταστήσει και απεγκαταστήσει τμήματα χωρίς να επηρεάσει τον περιέκτη. Τέλος θα εγκαταστήσουμε το Apache Karaf, μια πλατφόρμα που εκτελεί εφαρμογές που βασίζονται στο OSGi την οποία διαδικασία θα την αναλύσουμε λεπτομερώς στο επόμενο κεφάλαιο.

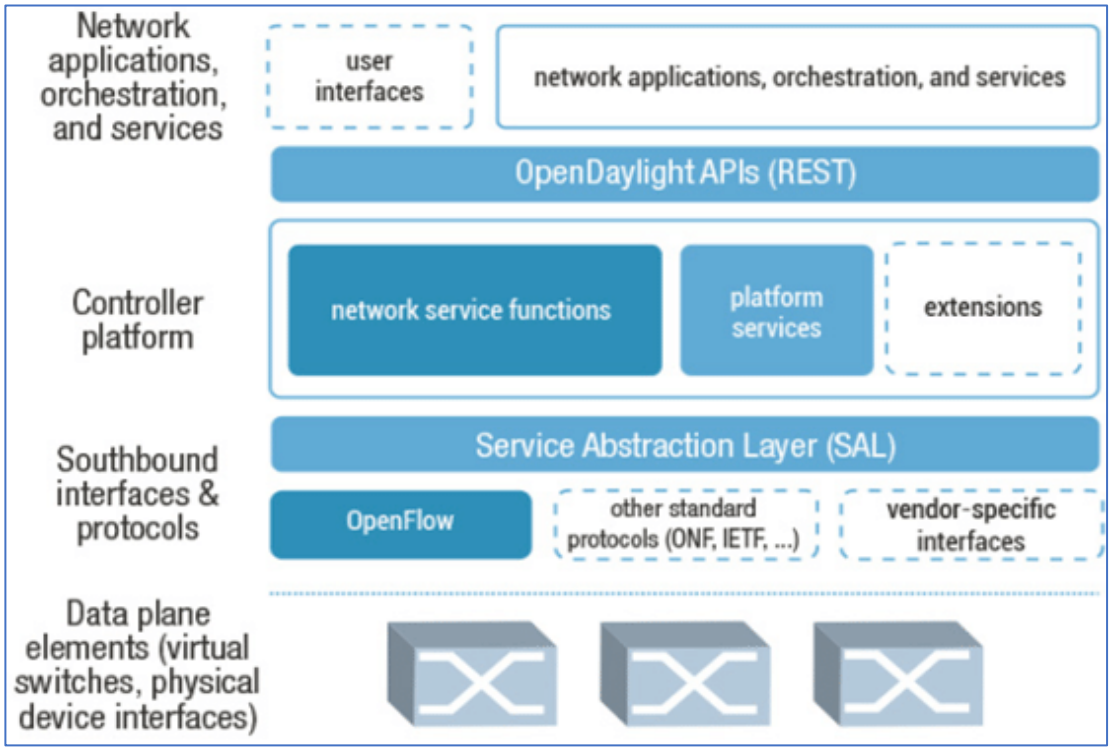
- **Java interfaces**: Οι Java interfaces χρησιμοποιούνται για περιστάσεις ακρόασης, προδιαγραφών και διαμόρφωση διάταξης. Αυτός είναι ο κύριος τρόπος με τον οποίο συγκεκριμένα πακέτα εφαρμόζουν υλοποίηση συνάρτησης (call-back) λειτουργίες για ζητήματα και επίσης να δείξουν ενημέρωση μιας συγκεκριμένης κατάστασης.

- **REST APIs**: Είναι northbound APIs όπως διαχειριστής τοπολογίας, ιχνηλάτης εξυπηρετητή, προγραμματιστής ροής και στατικής δρομολόγησης.

- **YANG**: Είναι συλλογή βιβλιοθηκών και εργαλείων παρέχοντας υποστήριξη για χρήση του YANG (Yet Another Next Generation) για Java εργασίες και εφαρμογές.

3.1 Αρχιτεκτονική OpenDaylight

Ο ODL υποστηρίζει μια επιστρωμένη αρχιτεκτονική με καθαρά ενσωματωμένα σημεία και APIs που επιτρέπουν στους τελικούς χρήστες και τους δικτυακούς προμηθευτές να συμμετέχουν στις δεξιότητες SDN ενός ODL. Το ακόλουθο διάγραμμα αναπαριστά την αρχιτεκτονική του ODL με τα δομικά του στοιχεία τα οποία θα αναλυθούν στη συνέχεια.



Εικόνα 10. Σχηματική αναπαράσταση γενικής αρχιτεκτονικής ODL με δομικά στοιχεία

Πηγή: https://www.researchgate.net/figure/The-simplified-architectural-framework-of-OpenDaylight-13_fig2_317057083

Controller platform: Ο ελεγκτής λειτουργεί σαν ενδιάμεσο λογισμικό στο οικοσύστημα του OpenDaylight. Είναι το πλαίσιο που ενώνει μαζί τις εφαρμογές απαιτώντας τις υπηρεσίες των δικτυακών συσκευών και των πρωτοκόλλων που επικοινωνούν με τις δικτυακές συσκευές για αφαίρεση υπηρεσιών. Ο ελεγκτής επιτρέπει οι εφαρμογές να είναι αγνωστικιστές για τις προδιαγραφές των δικτυακών

συσκευών, επιτρέποντας έτσι οι προγραμματιστές εφαρμογών να επικεντρώνονται στην ανάπτυξη της λειτουργικότητας της εφαρμογής.

Southbound protocols: Η νότια διεπαφή είναι ικανή να υποστηρίξει πολλαπλά πρωτόκολλα όπως για παράδειγμα το OpenFlow 1.0, OpenFlow 1.3, BGP-LS, LISP, SNMP και άλλα. Αυτές οι μονάδες είναι δυναμικά συνδεδεμένες σε ένα service abstraction layer (SAL), οι οποίες αποφασίζουν πως να εκπληρώσουν την αιτούμενη υπηρεσία ανεξάρτητα από το υπάρχον πρωτόκολλο που χρησιμοποιείται στον ελεγκτή και τις δικτυακές συσκευές.

Service Abstraction Layer: Στον OpenDaylight, το SAL είναι το κλειδί σχεδιασμού που επιτρέπει το διαχωρισμό των υπηρεσιών ανάμεσα στους καταναλωτές υπηρεσιών και τους κατασκευαστές. Το SAL λειτουργεί σαν μια μεγάλη εγγραφή υπηρεσιών που γνωστοποιείται από πολλές μονάδες και τις δεσμεύει σε εφαρμογές που τις απαιτούν. Οι μονάδες παρέχουν υπηρεσίες, ή οι κατασκευαστές, μπορούν να καταχωρήσουν τα APIs τους με την εγγραφή. Όταν μια εφαρμογή, ή ένας καταναλωτής, αιτούνται μια υπηρεσία μέσω ενός γενικού API, το SAL είναι υπεύθυνο για τη συναρμολόγηση του αιτήματος συνδέοντας κατασκευαστές και καταναλωτές σε μια σύμβαση, κανονίζεται και συντηρείται από το SAL. Το SAL έχει δυο διαφορετικούς αρχιτεκτονικά τρόπους για την εφαρμογή της εγγραφής: application-driven Sal και module-driven SAL.

Service Functions: Ο ελεγκτής έχει πολλές βασικές δικτυακές λειτουργίες που περιλαμβάνονται ως μέλος της βάσης αποστολής. Αυτό περιλαμβάνει υπηρεσίες για τον εντοπισμό τοπολογίας και διασποράς, ένα διαχειριστή προώθησης για την διαχείριση βασικών κανόνων προώθησης, και ένα διαχειριστή μεταγωγέα για την αναγνώριση στοιχείων δικτύου στην υπάρχουσα φυσική τοπολογία. Το SAL λειτουργεί ως μια ενεργή εγγραφή για τον κανονισμό συμβάσεων ανάμεσα στους παρόχους υπηρεσιών και τους καταναλωτές υπηρεσιών. Αυτές οι συμβάσεις είναι προνομιούχες από το SAL χωρίς κάποια άμεση εξάρτηση στα αντίστοιχα πρόσθετα. Για παράδειγμα, μια πρόσθετη υπηρεσία τοπολογίας είναι υπεύθυνη για την αναγνώριση κόμβων και μονοπατιών ανάμεσα τους για την δημιουργία ενός γραφήματος. Αυτό το πρόσθετο εκθέτει λειτουργίες που μπορούν να χρησιμοποιηθούν από μια εφαρμογή για να λάβει μια πλήρη εικόνα του φυσικού

στρώματος. Έκτοτε, η εφαρμογή μπορεί να χρησιμοποιήσει την υπηρεσία προγράμματος ροής για τη δημιουργία ροών σε όλες τις δικτυακές συσκευές για την εφαρμογή μιας λογικής end-to-end ροής.

Platform Services: Ο ελεγκτής πλατφόρμας από μόνος του περιέχει μια συλλογή από δυναμικά πρόσθετα τμήματα για την εκτέλεση αναγκαίων δικτυακών εργασιών. Επίσης για τις βασικές δικτυακές υπηρεσίες, υπηρεσίες πλατφόρμας και άλλες επεκτάσεις προστίθενται επίσης στον ελεγκτή πλατφόρμας για βελτιωμένη λειτουργικότητα του SDN. Μερικές από τις υπηρεσίες πλατφόρμας είναι (α) ένα Virtual Tenant Network (VTN) το οποίο παρέχει πολλαπλούς ενοικιαστές εικονικού δικτύου χρησιμοποιώντας το OpenFlow, (β) συγγένεια υπηρεσιών (affinity services) που εκθέτουν τα APIs για να εκφράσουν τις σχέσεις φόρτου εργασίας και επίπεδα υπηρεσίας, (γ) BGP-LS/PCEP για την υποστήριξη μηχανικής κίνησης με το BGP-LS (Border Gateway Protocol) και PCEP (Path Computation Element Protocol), (δ) ένα OpenStack Neutron για να παρέχει διαχείριση Neutron API για πολλαπλές υλοποιήσεις, (ε) μια πολιτική που βασίζεται σε ομάδα (GBP) όπου παρουσιάζει μια καινοτόμο ιδέα ομάδων των τελικών σημείων και αφηρημένες έννοιες πολιτικής που ελέγχουν την επικοινωνία ανάμεσα σε αυτές τις ομάδες, (ζ) Service Function Chaining (SFC), μια αλυσίδα λειτουργίας υπηρεσιών που παρέχει την δυνατότητα να ορίσει μια αλυσίδα από δικτυακές υπηρεσίες, (η) μια LISP υπηρεσία χαρτογράφησης που μπορεί να χρησιμοποιηθεί για την εφαρμογή εικονικών δικτύων, και (ι) ένα SDNi aggregator για τη συλλογή πληροφοριών, όπως η τοπολογία, στατιστικές και άλλα για την ενεργοποίηση επικοινωνίας δια SDN ελεγκτή.

Northbound ODL APIs: Η βασική αρχιτεκτονική αρχή του ODL ελεγκτή περιέχεται από εφαρμογές και αρθρωτές υπηρεσίες, διαχειριζόμενη από την εφαρμογή ενός SAL. Ο ελεγκτής εκθέτει ανοιχτά northbound APIs, τα οποία χρησιμοποιούνται από εφαρμογές. Ο ODL υποστηρίζει και το πλαίσιο OSGi και τα αμφίδρομα REST APIs στο βόρειο επίπεδο. Το πλαίσιο OSGi χρησιμοποιείται κυρίως από εφαρμογές που εκτελούνται στον ίδιο χώρο διεύθυνσης με τον ελεγκτή, ενώ το REST API χρησιμοποιείται από εφαρμογές που μπορούν να εκτελεστούν στην ιδιά μηχανή με τον ελεγκτή ή σε διαφορετική. Αυτές οι εφαρμογές αντιλαμβάνονται την επιχειρησιακή λογική και περιέχουν τους απαραίτητους αλγορίθμους.

ODL Applications: Το πάνω στρώμα του ODL αποτελείται από εφαρμογές επιχειρησιακής και δικτυακής λογικής όπου ελέγχουν και παρακολουθούν τη συμπεριφορά του δικτύου. Τέλος, οι βόρειες εφαρμογές περιέχουν περισσότερο πολύπλοκη ενορχήστρωση εφαρμογών όπου οδηγούν τη κίνηση του δικτύου σε εναρμονισμό με τις ανάγκες των περιβαλλόντων όπως το Cloud και το NFV.

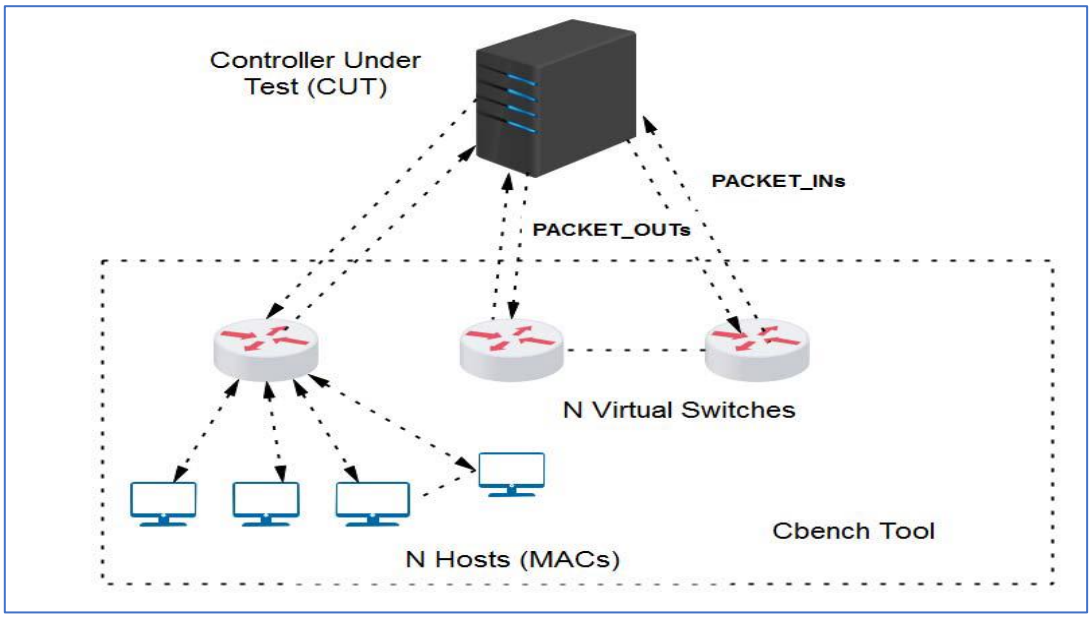
3.2 Σύγκριση OpenDaylight με άλλους ελεγκτές

Όπως έχουμε αναφέρει σε προηγούμενο κεφάλαιο, το SDN είναι μια αρχιτεκτονική που διαχωρίζει τη νοημοσύνη της δρομολόγησης από τις λειτουργίες προώθησης, χρησιμοποιώντας μια οντότητα που καλείται ελεγκτής. Ωστόσο, είναι ύψιστης σημασίας ότι η απόδοση του ελεγκτή είναι προτεραιότητα για την ανάπτυξη του. Η ραγδαία εμφάνιση πολλών νέων ελεγκτών στην ερευνητική κοινότητα θέτει δύσκολη την επιλογή του κατάλληλου ελεγκτή. Σύμφωνα με τους Mamushiane et. al. (2018) [26] η αξιολόγηση της απόδοσης περιλαμβάνει τον ODL με τους ελεγκτές ONOS, Ryu, Floodlight. Τα μετρικά τις απόδοσης περιλαμβάνονται από τη καθυστέρηση (latency) και τη παραγωγικότητα (throughput). Τέλος, ο κύριος στόχος είναι να διερευνήσουμε ποιος ελεγκτής αποδίδει την υψηλότερη παραγωγικότητα και την χαμηλότερη καθυστέρηση κάτω από διάφορους φόρτους εργασίας.

3.2.1 Προετοιμασία περιβάλλοντος δοκιμών αξιολόγησης

Η αξιολόγηση θα εκτελεστεί χρησιμοποιώντας το Cbench, ένα εργαλείο μέτρησης απόδοσης για την σύγκριση των προαναφερθέντων ελεγκτών που είναι συμβατοί με το πρωτόκολλο OpenFlow. Η διάταξη του πειράματος σκιαγραφείται στη παρακάτω εικόνα, όπου το εργαλείο Cbench χρησιμοποιείται για να προσομοιάσει διαφορετικό αριθμό μεταγωγέων (1, 4, 8, 12, 16, 20, 24, 28, και 32) όπου συνδέονται στον ελεγκτή (controller under test (CUT)), στέλνουν PACKET-IN μηνύματα και μετρούν τον αριθμό των αποκρίσεων (PACKET-OUTS) που λαμβάνονται ανά δευτερόλεπτο καθώς και τη καθυστέρηση. Ο μοναδικός αριθμός φυσικών διευθύνσεων (MAC addresses) παραμένει στις 1000 MACs ενώ μεταβάλλεται ο αριθμός των προσομοιωμένων μεταγωγέων. Η κάθε δοκιμή επαναλαμβάνεται 10 φορές και ένας μέσος όρος χρησιμοποιείται ως το αποτέλεσμα

για τις δύο καταστάσεις λειτουργίας (latency και throughput). Ο αριθμός των εργαζόμενων νημάτων (worker threads) κρατείται στα 4. Στόχος αυτής της διαδικασίας είναι να ερευνήσει την επίδραση του αυξανόμενου αριθμού των προσομοιωμένων μεταγωγέων στη νότιο απόδοση του ελεγκτή.



Εικόνα 11. Σχηματική αναπαράσταση διάταξης πειράματος [26]

Η δεύτερη δοκιμή εμπεριέχει μεταβαλλόμενο αριθμό φυσικών διευθύνσεων MACs (1k, 10k, 100k, 1000k, 10000k), ενώ διατηρεί σταθερό τον αριθμό των μεταγωγέων στους 16 και στις δυο καταστάσεις, καθυστέρησης και παραγωγικότητας. Η δοκιμή αυτή γίνεται για να καθορίσει τον αριθμό των τελικών εξυπηρετητών (end hosts) στην απόδοση του ελεγκτή. Η κάθε δοκιμή επαναλαμβάνεται 14 φορές, με κάθε μια από αυτές να διαρκεί 10 δευτερόλεπτα. Ο αριθμός των νημάτων διεργασιών (worker threads) διατηρείται στα 4. Το ακόλουθο παράδειγμα εντολής χρησιμοποιείται για την εκτέλεση των δοκιμών:

./cbench –c localhost –p 6633 –l 14 -m 10000 –M 1000 –s 8 –t

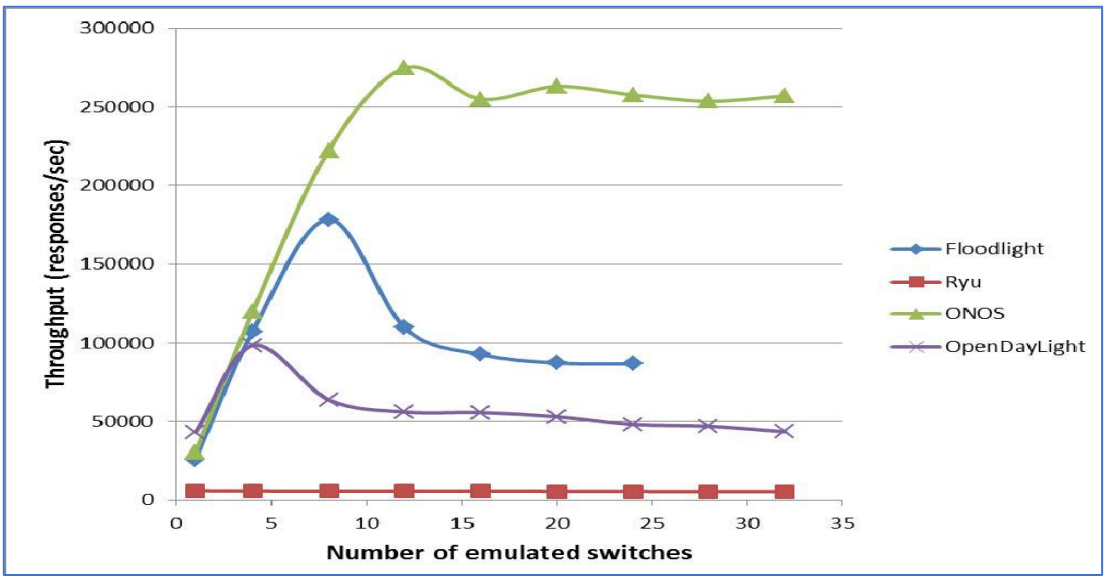
όπου οι παράμετροι των εντολών αναγνωρίζονται ως:

- c είναι ο ελεγκτής (IP ή hostname)
- p είναι το port number του ελεγκτή
- l είναι ο αριθμός των loops ανά εξέταση
- m δηλώνει το χρόνο της εξέτασης σε δευτερόλεπτα

- M είναι ο αριθμός των MAC διευθύνσεων ανά μεταγωγέα
- s είναι ο αριθμός των μεταγωγέων (switches)
- t σημαίνει ότι το Cbench εκτελείται σε κατάσταση παραγωγικότητας (throughput)

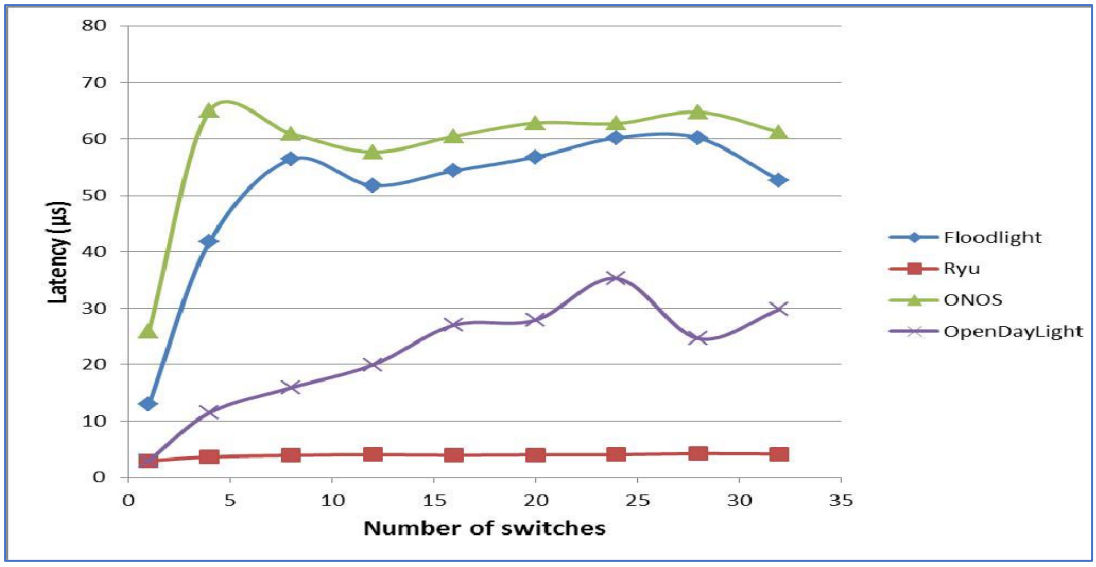
3.2.2 Αποτελέσματα ανάλυσης αξιολόγησης των πειραματικών δοκιμών

Throughput: Στην παρακάτω εικόνα παρουσιάζονται τα αποτελέσματα της αξιολόγησης της παραγωγικότητας, όπου ο Floodlight και ο ODL είναι δραστικά επηρεασμένοι από μια αύξηση στον αριθμό των ενεργών μεταγωγέων. Αυτό γίνεται επειδή έχοντας ένα μεγάλο αριθμό μεταγωγέων προκαλεί συναγωνισμό μεταξύ τους στο επίπεδο δεδομένων, γεγονός που απαιτεί μεγάλη επεξεργαστική ισχύ. Η παραγωγικότητα της απόδοσης του Ryu είναι η πιο χαμηλή και παραμένει σταθερά ανεξάρτητη από τον αριθμό των προσομοιωμένων μεταγωγέων. Ο ONOS εμφανίζει την καλύτερη απόδοση παραγωγικότητας.



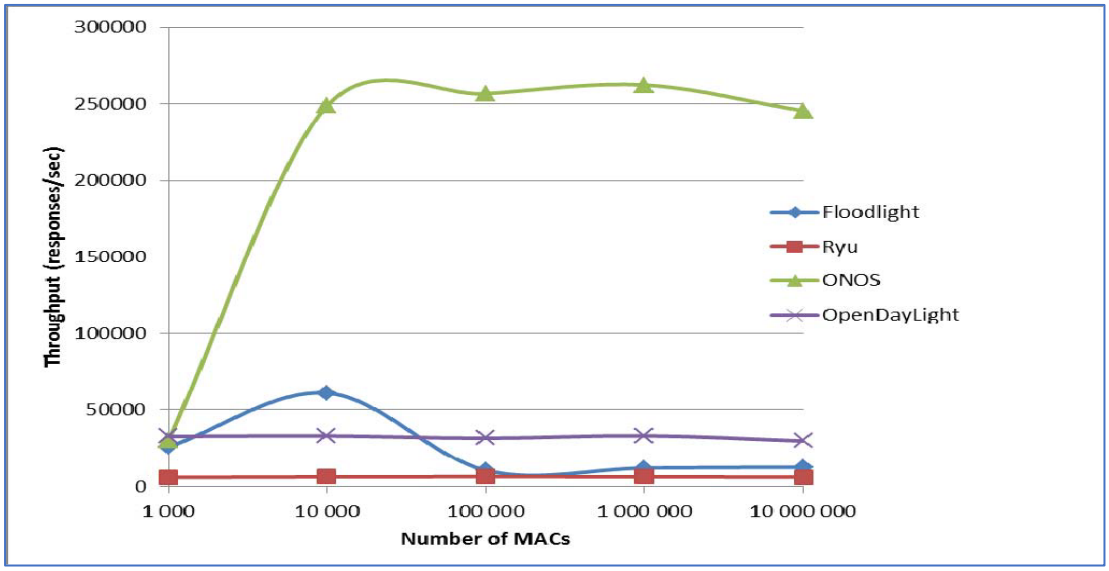
Εικόνα 12. Μέσος αριθμός αποκρίσεων ανά δευτερόλεπτο κάτω από μεταβαλλόμενο αριθμό μεταγωγέων (MACs=1000, threads=4) [26]

Latency: Όταν η λειτουργία είναι σε κατάσταση καθυστέρησης, ο Ryu και ο ODL παρουσιάζουν την καλύτερη καθυστέρηση. Ο ONOS και ο Floodlight δείχνουν τα χειρότερα αποτελέσματα καθυστέρησης καθώς ο αριθμός των μεταγωγέων αυξάνεται.



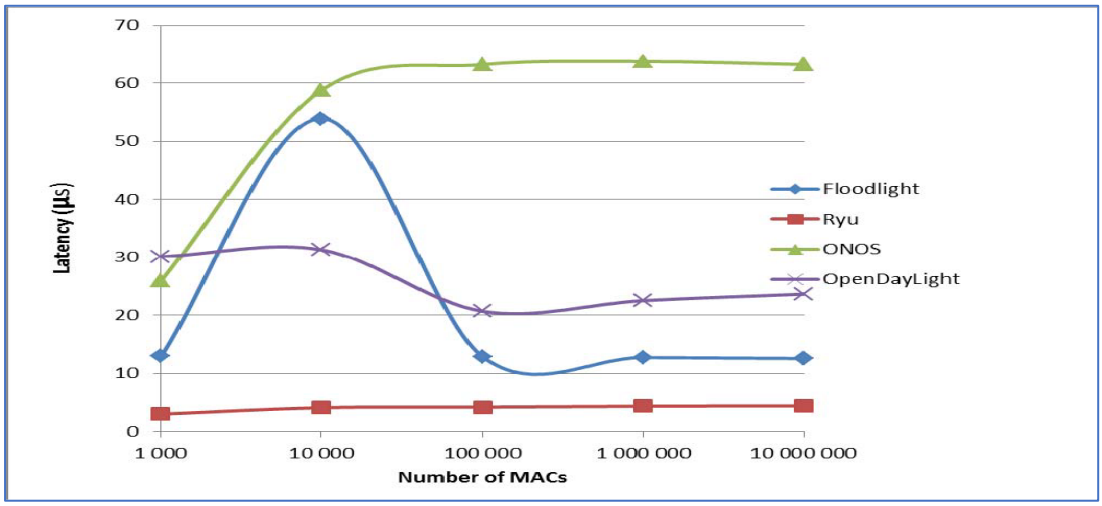
Εικόνα 13. Μέση καθυστέρηση κάτω από μεταβαλλόμενο αριθμό μεταγωγέων (MACs=1000, thread=4) [26]

Scalability: Όπως παρατηρούμε στην παρακάτω εικόνα, ο ODL, Ryu και Floodlight είναι σημαντικά επηρεασμένοι από το φόρτο εργασίας λόγω του μεγάλου αριθμού των MACs. Η απόδοση της παραγωγικότητας του ONOS είναι σταθερή ξεκινώντας από τους 10k μεταγωγείς. Αυτό συμβαίνει επειδή η εφαρμογή μεταγωγέα του ONOS διαχειρίζεται συναγωνισμό ανάμεσα στις MACs χωρίζοντας το πίνακα διεύθυνσης MAC του δικτύου ανάμεσα σε μια συλλογή κατακερματισμένων πινάκων (hash tables) που επιλέγονται από το κατακερματισμό της MAC διεύθυνσης.



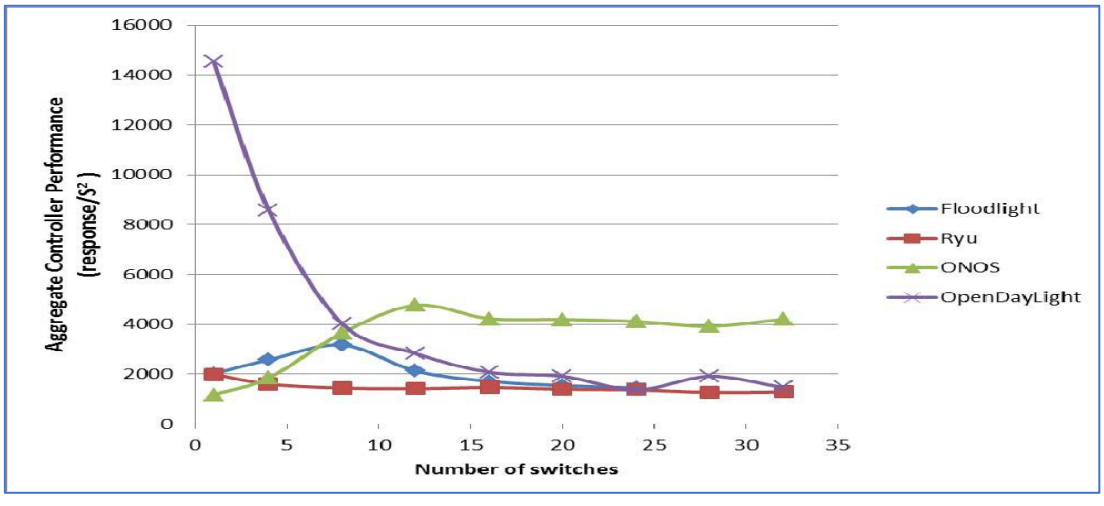
Εικόνα 14. Αριθμός αποκρίσεων ανά δευτερόλεπτο κάτω από μεταβαλλόμενο αριθμό των MACs (s=16, threads=4) [26]

Latency: Όπως φαίνεται στη παρακάτω εικόνα, όταν οι εξετάσεις εκτελούνται στη κατάσταση καθυστέρησης, ο ONOS παρουσιάζει τη χειρότερη απόδοση καθυστέρησης. Η υποβάθμιση της απόδοσης του Ryu είναι ασήμαντα μικρή. Η καθυστέρηση του ODL αυξάνεται με την άνοδο του φόρτου εργασίας και ο Floodlight παρουσιάζει καλύτερη απόδοση καθυστέρησης σε σύγκριση με τον ODL όταν ο αριθμός των MAC ρυθμίζεται στα 100k.



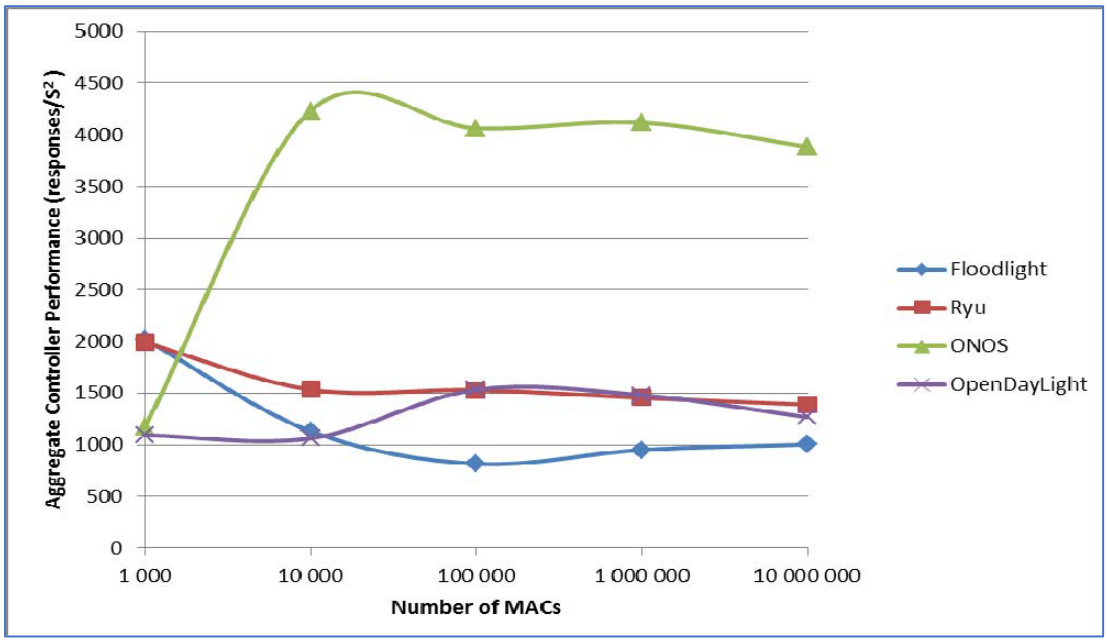
Εικόνα 15. Μέσος όρος καθυστέρησης με μεταβαλλόμενο αριθμό των MACs (s=16, threads=4) [26]

Aggregate performance: Ορίζουμε τα θετικά χαρακτηριστικά για τη συνολική απόδοση παίρνοντας μια αναλογία της παραγωγικότητας (responses/sec) σε καθυστέρηση (sec). Η ακόλουθη εικόνα παρουσιάζει τα αποτελέσματα απόδοσης εν όψει της καθυστέρησης και της παραγωγικότητας κάτω από μεταβαλλόμενο αριθμό μεταγωγέων. Το αποτέλεσμα δείχνει ότι ο ONOS έχει συνολικά την καλύτερη απόδοση καθώς το μέγεθος του επιπέδου δεδομένων αυξάνεται.



Εικόνα 16. Συνολική απόδοση ελεγκτή (MACs=1000, threads=4) [26]

Κάτω από μεταβαλλόμενο φόρτο εργασίας (MACs) όπως φαίνεται στη παρακάτω εικόνα ο ONOS παρουσιάζει εξαιρετική επεκτασιμότητα, ενώ ο ODL και ο Ryu συγκριτικά την ίδια απόδοση για 100.000 MACs και πάνω. Ο Floodlight έχει την χαμηλότερη συνολική απόδοση για μεγαλύτερο φόρτο εργασίας.

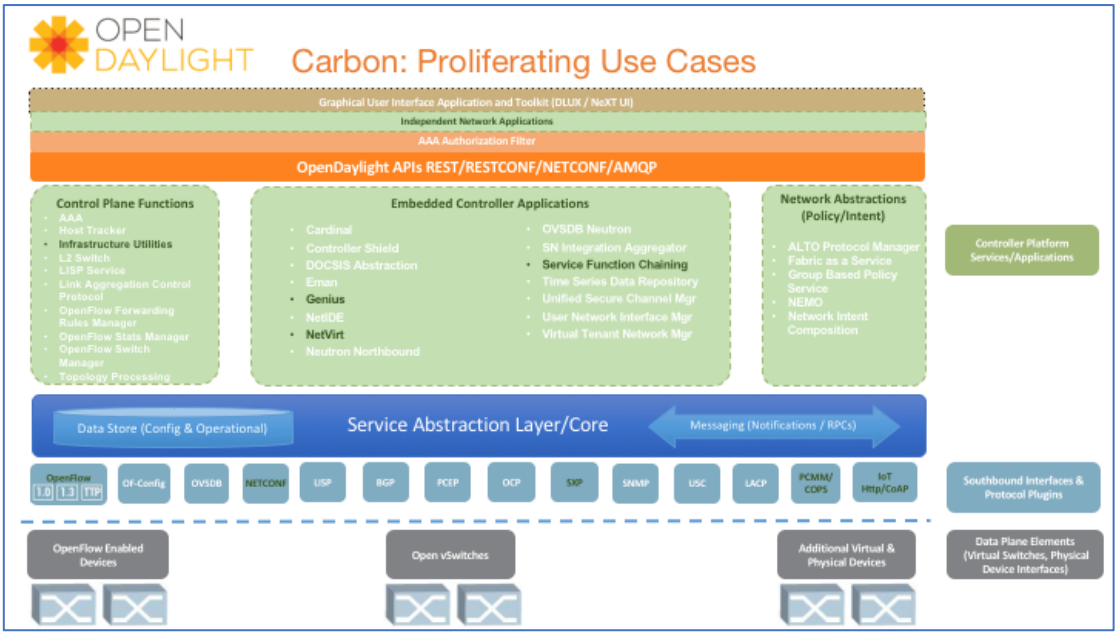


Εικόνα 17. Συνολική απόδοση ελεγκτή (s=16, threads=4) [26]

Στην παραπάνω συγκριτική αξιολόγηση, παρουσιάστηκαν τα θετικά χαρακτηριστικά και ελαττώματα των ελεγκτών (ODL, ONOS, Ryu και Floodlight). Από την ανάλυση αυτή συμπεραίνουμε ότι η υιοθέτηση του ODL είναι χαρακτηριστικά πιο πλούσια όσον αφορά την υποστήριξη διεπαφών από τους κατασκευαστές των ελεγκτών αυτών. Από την αξιολόγηση της απόδοσης, ο ONOS παρουσίασε την καλύτερη παραγωγικότητα ικανός να αποκριθεί σε αιτήματα άμεσα κάτω από διάφορο φόρτο κυκλοφορίας. Ωστόσο, σε κατάσταση καθυστέρησης ο Ryu δείχνει τα καλύτερα αποτελέσματα καθυστέρησης κάνοντας τον κατάλληλο για εφαρμογές ευάλωτες σε καθυστέρηση.

3.3 Χαρακτηριστικά και εφαρμογές OpenDaylight

Αυτή η ενότητα συνοψίζει τα υπάρχον χαρακτηριστικά και λειτουργίες του ODL [26] όπως φαίνονται στην παρακάτω εικόνα. Τα χαρακτηριστικά αυτά χρησιμοποιούνται δοκιμαστικά.



Εικόνα 18. ODL Carbon χαρακτηριστικά

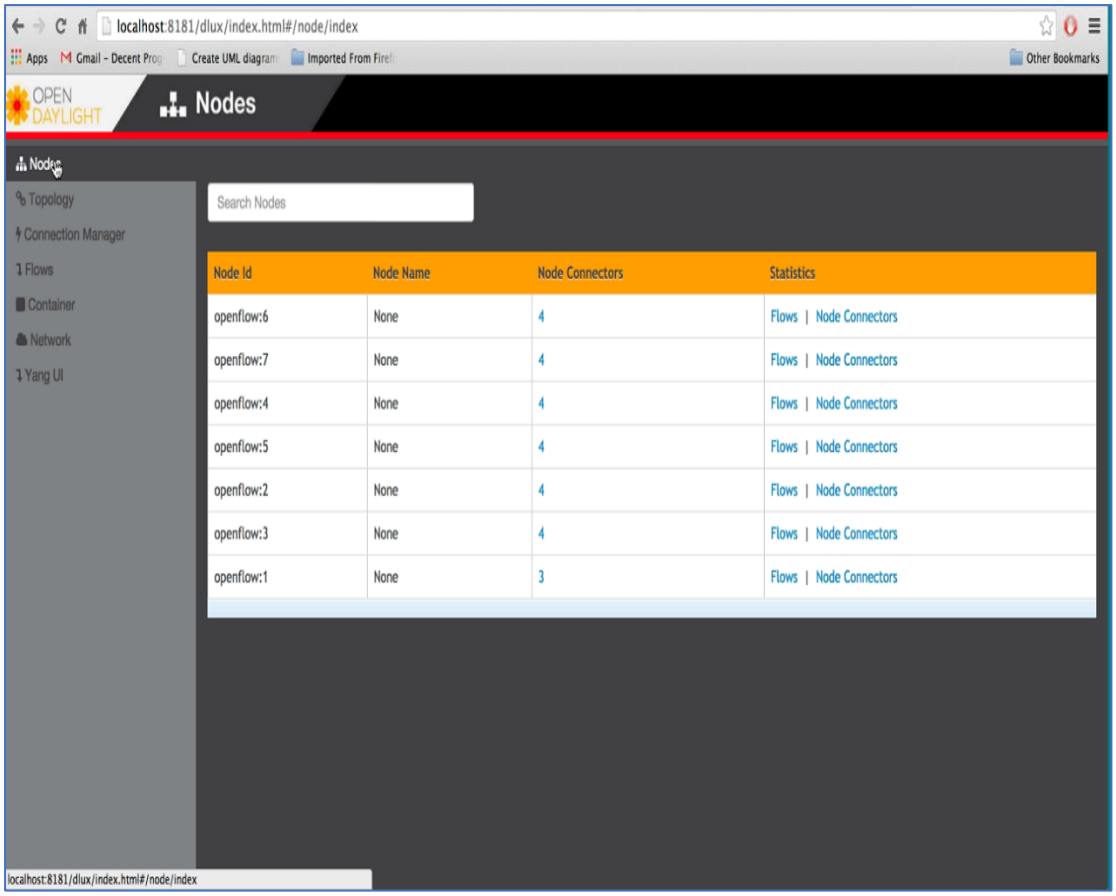
Πηγή: https://www.opendaylight.org/what-we-do/current-release/carbon

3.3.1 DLUX

Το OpenDaylight User Experience (DLUX) [28] είναι ένα βασισμένο σε JavaScript (JS) γραφικό περιβάλλον χρήστη όπου επικοινωνεί με την τελική υπηρεσία για να παρέχει μια σταθερή και φιλική προς τον χρήστη διεπαφή για να αλληλοεπιδρά με τα ODL projects και τον ελεγκτή βάσης. Το DLUX παρέχει έναν αριθμό από διάφορα χαρακτηριστικά, τα οποία ο χρήστης μπορεί να ενεργοποιήσει και να απενεργοποιήσει ξεχωριστά: [29]

- odl-dlux-core
- odl-dluxapps-nodes
- odl-dluxapps-topology
- odl-dluxapps-yangui
- odl-dluxapps-yangvisualizer
- odl-dluxapps-yangman

Για την πρόσβαση στο DLUX, μετά την εγκατάσταση της εφαρμογής ανοίγουμε έναν browser και πληκτρολογούμε την διεύθυνση URL *http://<your-karaf-ip>:8181/index.html* με username και password το συνθηματικό <<admin>>.



Εικόνα 19. Σχηματική αναπαράσταση DLUX Module

Πηγή: https://docs.opendaylight.org/en/stable-nitrogen/_images/dlux-login.png

Μετά την είσοδο στο DLUX, αν ενεργοποιήσουμε μόνο το χαρακτηριστικό *odl-dlux-core*, θα δούμε μόνο την εφαρμογή τοπολογίας διαθέσιμη στο αριστερό παράθυρο. Το DLUX έχει άλλες εφαρμογές όπως το node, yang UI οι οποίες δεν εμφανίζονται μέχρις ότου ενεργοποιήσουμε τα χαρακτηριστικά τους *odl-dluxapps-nodes* και *odl-dluxapps-yangui.* Τέλος, αν εγκαταστήσουμε την εφαρμογή στο DLUX θα προβληθεί στα αριστερά ένας πλοηγός μετά την ανανέωση του browser μας.

3.3.2 L2 Switch

Το L2 Switch [30] είναι χαρακτηριστικό ή τμήμα του ODL το οποίο παρέχει Layer2 switch λειτουργικότητα. Η αρχιτεκτονική του L2 Switch αποτελείται από:

- **Packet Handler**: Αποκωδικοποιεί τα εισερχόμενα πακέτα στον ελεγκτή και τα προωθεί κατάλληλα.
- **Loop Remover**: Αφαιρεί του βρόχους στο δίκτυο.
- **Arp Handler**: Διαχειρίζεται τα κωδικοποιημένα ARP πακέτα.

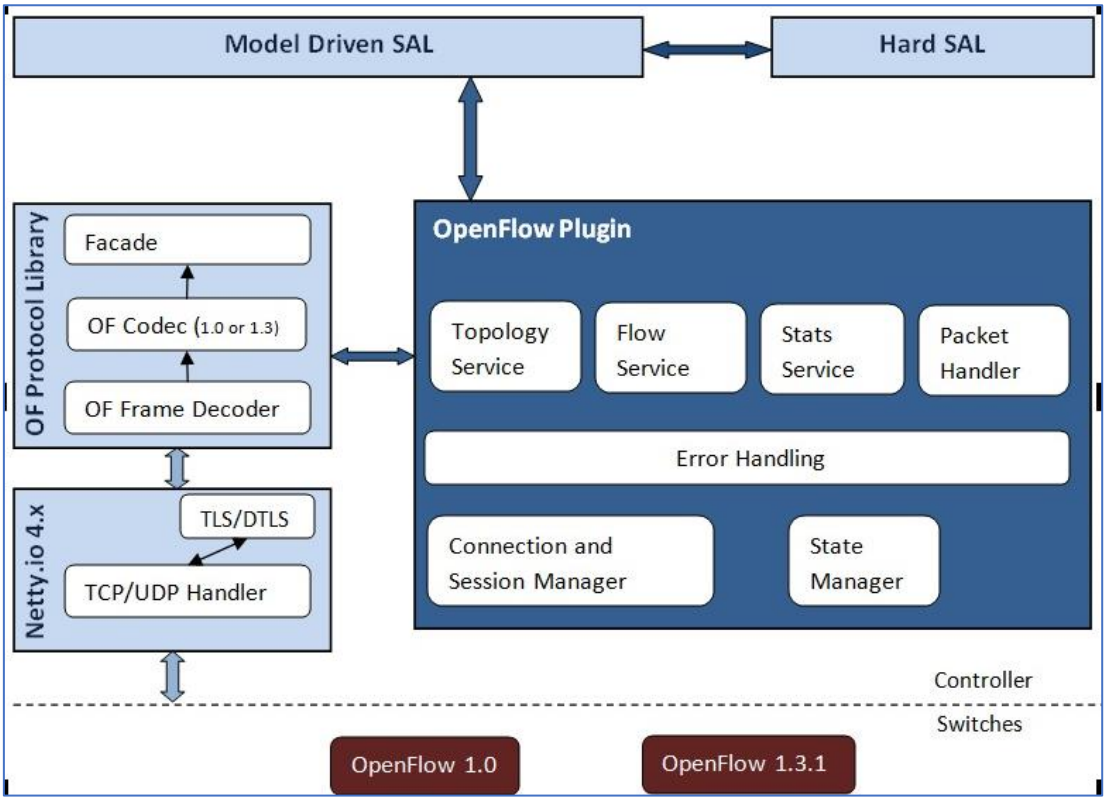
- **Address Tracker**: Ανακτά τις διευθύνσεις (MAC ή IP) των οντοτήτων του δικτύου.
- **Host Tracker**: Καταγράφει τις τοποθεσίες των hosts στο δίκτυο.
- **L2 Switch Main**: Εγκαθιστά ροές σε κάθε που βασίζεται στη κυκλοφορία του δικτύου.

3.3.3 OpenFlow plugin

Όπως έχει προαναφερθεί σε προηγούμενο κεφάλαιο, το OpenFlow είναι μια ορισμένη τυπική διεπαφή επικοινωνίας η οποία ενεργοποιεί επικοινωνία ανάμεσα στα επίπεδα ελέγχου και προώθησης σε μια SDN αρχιτεκτονική. Το OpenFlow plugin project [31] σκοπεύει να αναπτύξει ένα plugin για να υποστηρίξει υλοποιήσεις της προδιαγραφής OpenFlow καθώς αναπτύσσεται και εξελίσσεται. Συγκεκριμένα το project έχει αναπτύξει ένα plugin με στόχο να υποστηρίζει το OpenFlow 1.0 και 1.3x αλλά και μεταγενέστερες προδιαγραφές του OpenFlow. Το plugin αυτό, όπως και άλλα τμήματα του ODL, βασίζονται στο Model Driven Abstraction Layer (MD-SAL).

Το OpenFlow 1.3 plugin υποστηρίζει την ακόλουθη λειτουργικότητα:

- Μεταχείριση επικοινωνίας
- Διαχείριση συνεδρίας
- Διαχείριση κατάστασης
- Μεταχείριση σφάλματος
- Απεικόνιση λειτουργίας
- Ίδρυση επικοινωνίας που διαχειρίζεται από τη βιβλιοθήκη OpenFlow χρησιμοποιώντας ανοιχτού κώδικα netty.io βιβλιοθήκη
- Μεταχείριση μηνύματος
- Μεταχείριση γεγονότων και διάδοση στα ανώτερα επίπεδα
- Το plugin υποστηρίζει το MD-SAL και το Hard SAL
- Οπισθοδρομικά συμβατό με το OpenFlow 1.0



Εικόνα 20. Υλοποίηση OpenFlow σε ODL ελεγκτή [31]

4. Υλοποίηση προσομοίωσης σε Mininet εξομοιωτή αρχιτεκτονικής OpenDaylight SDN

4.1 Ανάπτυξη OpenDaylight σε VirtualBox

Σήμερα υπάρχει μια μεγάλη γκάμα από ελεύθερα λογισμικά εικονικών μηχανών (Virtual Machines). Στη παρούσα εργασία επιλέξαμε το VirtualBox για τη δοκιμή του πειράματος και την φιλοξενία του ODL ελεγκτή μαζί με το εργαλείο Mininet.

Σύντομη αναφορά για VMs και VirtualBox:

Μια εικονική μηχανή [32] είναι ένα πρόγραμμα που λειτουργεί ως εικονικός υπολογιστής. Λειτουργεί με το τρέχον λειτουργικό σύστημα μας, το λειτουργικό σύστημα “host”, και παρέχει εικονικό υλικό σε λειτουργικά συστήματα “guest”. Τα λειτουργικά συστήματα επισκεπτών τρέχουν σε παράθυρα στο λειτουργικό σύστημα του κεντρικού υπολογιστή μας, όπως ακριβώς οποιοδήποτε άλλο πρόγραμμα στον υπολογιστή μας. Το Oracle VM Virtual Box ή VirtualBox είναι υπερεπόπτης (hypervisor) ανοιχτού κώδικα για υπολογιστές x86 που αναπτύσσεται από την Oracle Corporation. Αναπτύχθηκε αρχικά από την Innotek GmbH και αποκτήθηκε από τη Sun Microsystems το 2008, η οποία εξαγοράστηκε το 2010 από την Oracle. Το VirtualBox είναι μια μεγάλη εφαρμογή ανοιχτού κώδικα που εκτελείται σε Windows, Linux και macOS. Ένα από τα πλεονεκτήματα του VirtualBox είναι ότι δεν υπάρχει εμπορική έκδοση και όλες του οι λειτουργίες είναι δωρεάν.

Ρυθμίσεις λειτουργικού συστήματος φιλοξενίας:

- Windows 10 pro
- Ενσωματωμένη μνήμη RAM 16,0 GB
- Επεξεργαστής (CPU) Intel(R) Core(TM) i5-6500 CPU @ 3.20 GHz
- Τύπος συστήματος 64-bit operating system, x64-based processor
- Μέγεθος δίσκου 465 GB

Ρυθμίσεις λογισμικού:

- Hypervisor: Oracle VM VirtualBox Manager version 6.0.24

- Λειτουργικό σύστημα: Ubuntu Desktop Image 20.04.2.0 LTS
- ODL version: Carbon (100 GB hard disk, 4 GB RAM, 2 CPUs)

4.2 Υλοποίηση περιβάλλοντος ODL Karaf Distribution

Στο κομμάτι αυτό θα παρουσιασθεί λεπτομερώς η εγκατάσταση, η υλοποίηση και η εκκίνηση του ODL controller ως Karaf distribution. Όπως έχει προαναφερθεί σε προηγούμενο κεφάλαιο, το Apache Karaf είναι μια πλατφόρμα βασισμένη στο OSGi πλαίσιο το οποίο παρέχει επιπλέον χαρακτηριστικά.

4.2.1 Εγκατάσταση Java

Όπως γνωρίζουμε ο ODL SDN controller είναι ένα βασισμένο σε Java πρόγραμμα, επομένως εγκαθιστούμε το περιβάλλον της Java με τις ακόλουθες εντολές [33]:

```
giannis@giannis-VirtualBox:~$ sudo apt-get install openjdk-8-jdk
```

Επαλήθευση της εγκατεστημένης Java Environment Kit (JDK) έκδοσης:

```
giannis@giannis-VirtualBox:~$ java -version
openjdk version "1.8.0_292"
OpenJDK Runtime Environment (build 1.8.0_292-8u292-b10-0ubuntu1~20.04-b10)
OpenJDK 64-Bit Server VM (build 25.292-b10, mixed mode)
```

Ρύθμιση της μεταβλητής περιβάλλοντος JAVA_HOME:

```
giannis@giannis-VirtualBox:~$ nano ~/.bashrc
```

Στη συνέχεια προσθέτουμε στο αρχείο .bashrc το προκαθορισμένο μονοπάτι (path) για τη Java:

```
  GNU nano 4.8                    /home/giannis/.bashrc
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

Αποθηκεύουμε και εκτελούμε το αρχείο .bashrc:

```
giannis@giannis-VirtualBox:~$ source ~/.bashrc
giannis@giannis-VirtualBox:~$ ls -a
.                .cache      .mozilla   .sudo_as_admin_successful
..               .config     Music      Templates
.bash_history    Desktop     Pictures   .vboxclient-clipboard.pid
.bash_logout     Documents   .profile   .vboxclient-display.pid
.bashrc          Downloads   Public     .vboxclient-draganddrop.pid
.bashrc.save     .gnupg      snap       .vboxclient-seamless.pid
.bashrc.save.1   .local      .ssh       Videos
```

Τέλος, επαληθεύουμε το προκαθορισμένο μονοπάτι της JAVA_HOME:

```
giannis@giannis-VirtualBox:~$ echo $JAVA_HOME
/usr/lib/jvm/java-8-openjdk-amd64
```

4.2.2 Λήψη και εγκατάσταση ODL Controller

Για τη λήψη του ODL λογισμικού έκδοσης Carbon από το https://docs.opendaylight.org/en/latest/downloads.html χρησιμοποιούμε την εντολή *wget* για τη λήψη του tar αρχείου:

```
giannis@giannis-VirtualBox:~$ wget https://nexus.opendaylight.org/content/repositories/public/org/openday
light/integration/distribution-karaf/0.6.4-Carbon/distribution-karaf-0.6.4-Carbon.tar.gz
```

Στη συνέχεια αποσυμπιέζουμε το αρχείο tar.:

```
giannis@giannis-VirtualBox:~$ tar -xvf distribution-karaf-0.6.4-Carbon.tar.gz
```

Για την εκκίνηση του ODL εκτελούμε την ακόλουθη εντολή μέσα στο φάκελο της δια νομής του πακέτου μας:
giannis@giannis-VirtualBox:~/distribution-karaf-0.6.4-Carbon$ **./bin/karaf**

```
100% [========================================================================]

Karaf started in 41s. Bundle stats: 331 active, 331 total

    ________                       ________                .__  .__       .__     __
    \_____  \ ______   ____   ____ \______ \ _____  ___.__.|  | |__| ____ |  |___/  |_
     /   |   \\____ \_/ __ \ /    \ |    |  \\__  \<   |  ||  | |  |/ ___\|  |  \   __\
    /    |    \  |_> >  ___/|   |  \|    `   \/ __ \\___  ||  |_|  / /_/  >   Y  \  |
    \_______  /   __/ \___  >___|  /_______  (____  / ____||____/__\___  /|___|  /__|
            \/|__|        \/     \/        \/     \/\/                 /_____/      \/


Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>
```

4.2.3 Εγκατάσταση χαρακτηριστικών ODL

Όταν ο ODL βρίσκεται σε κατάσταση λειτουργίας μπορούμε να εγκαταστήσουμε οποιοδήποτε χαρακτηριστικό. Οποιοδήποτε χαρακτηριστικό του ODL μπορεί να ενεργοποιηθεί από την ακόλουθη εντολή, όπου feature1 είναι το όνομα του χαρακτηριστικού:

opendaylight-user@root> **feature:install <feature1>**

Χρειάζεται να εγκαταστήσουμε μια φορά ένα χαρακτηριστικό. Αφού εγκατασταθεί αυτό το χαρακτηριστικό θα προστεθεί μόνιμα στον ελεγκτή και θα εκτελείται κάθε φορά που αυτός ανοίγει. Στη δοκιμή αυτή κάναμε χρήση των ακόλουθων χαρακτηριστικών:

- odl-restconf: Επιτρέπει τη πρόσβαση στο RESTCONF API.
- odl-l2switch-switch: Παρέχει δικτυακή λειτουργικότητα παρόμοια με ένα Ethernet switch.
- odl-mdsal-apidocs: Παρέχει πρόσβαση στο YANG API.
- odl-dlux-all: OpenDaylight γραφικό περιβάλλον χρήστη (OpenDaylight user experience) παρέχει φιλική διεπαφή για το χρήστη.

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

Για να εμφανίσουμε τη λίστα με τα διαθέσιμα χαρακτηριστικά, εκτελούμε την εντολή:

opendaylight-user@root> **feature:list**

Για να εμφανίσουμε τη λίστα με τα ήδη εγκατεστημένα χαρακτηριστικά, εκτελούμε την εντολή:

opendaylight-user@root> **feature:list --installed**

Τέλος, για τη παύση και την έξοδο από το περιβάλλον του ODL controller χρησιμοποιούμε το συνδυασμό <ctrl-d> ή πληκτρολογούμε <logout>.

4.3 Επισκόπηση του Mininet

Το Mininet [34] είναι ένας εξομοιωτής δικτύου ο οποίος δημιουργεί ένα

δίκτυο από εικονικά hosts, switches, controllers και links. Οι hosts εκτελούν τυπικό λογισμικό δικτύου σε περιβάλλον Linux και οι switches υποστηρίζουν το OpenFlow για υψηλά ευέλικτη δρομολόγηση για το SDN περιβάλλον. Το Mininet υποστηρίζει την έρευνα, την ανάπτυξη, τη μάθηση, τη πρωτοτυπία, τον έλεγχο και τον εντοπισμό σφαλμάτων, και οποιεσδήποτε άλλες εργασίες που θα μπορούσαν να επωφεληθούν από την ύπαρξη ενός ολοκληρωμένου πειραματικού δικτύου σε ένα φορητό ή άλλο υπολογιστή.

4.3.1 Γιατί χρησιμοποιούμε το Mininet

Ο σκοπός χρήσης του Mininet είναι η παροχή ενός απλού και οικονομικού δοκιμαστικού δικτύου για την ανάπτυξη εφαρμογών OpenFlow, επιτρέπει ταυτοχρόνως σε πολλούς προγραμματιστές να εργάζονται πάνω στην ίδια τοπολογία, υποστηρίζει δοκιμές παλινδρόμησης σε σύστημα επιπέδου και επιτρέπει τον έλεγχο πολύπλοκης τοπολογίας. Επίσης, περιλαμβάνει ένα command line interface (CLI) για την ενημέρωση της κατάστασης της τοπολογίας και του OpenFlow, για τον εντοπισμό σφαλμάτων ή την εκτέλεση δοκιμών όλου του δικτύου. Υποστηρίζει, επιπλέον, αυθαίρετες προσαρμοσμένες τοπολογίες και περιλαμβάνει μια βασική συλλογή από παραμετρικές τοπολογίες χωρίς τη χρήση προγραμματισμού. Τέλος, παρέχει ένα εύκολο και επεκτάσιμο Python API για την δημιουργία και το πειραματισμό του δικτύου.

4.3.2 Πλεονεκτήματα και περιορισμοί του Mininet

Το Mininet συνδυάζει πολλά από τα καλύτερα χαρακτηριστικά των εξομοιωτών, δοκιμές υλικού και προσομοιωτών. Συγκριτικά με προσεγγίσεις βασισμένες σε ένα πλήρες σύστημα εικονικοποίησης, το Mininet προσφέρει:

- Ταχύτερη εκκίνηση: Δευτερόλεπτα αντί για λεπτά.
- Μεγαλύτερη επεκτασιμότητα συσκευών: Εκατοντάδες hosts και switches έναντι μονοψήφιου αριθμού.
- Παρέχει μεγαλύτερο εύρος ζώνης: Συνήθως 2Gbps συνολικό εύρος ζώνης σε απλό υλικό.
- Εγκαθίσταται εύκολα: Ένα διαθέσιμο προσυσκευασμένο VM που εκτελείται σε VMware ή VirtualBox για Mac/Linux/Windows με ήδη εγκατεστημένα

εργαλεία όπως OpenFlow 1.0.

Σε σύγκριση με το υλικό δοκιμής, το Mininet:

- Είναι οικονομικό και πάντα διαθέσιμο.
- Γρήγορα επαναρυθμιζόμενο και επανακκινήσιμο.

Σε σύγκριση με τους προσομοιωτές, το Mininet:

- Εκτελεί πραγματικό, μη τροποποιημένο κώδικα συμπεριλαμβανομένου κώδικα εφαρμογής, κώδικα πυρήνα λειτουργικού συστήματος (OS kernel), και κώδικα επιπέδου ελέγχου (OpenFlow ελεγκτή και Open vSwitch κώδικα).
- Συνδέεται εύκολα με πραγματικά δίκτυα.
- Προσφέρει διαδραστική απόδοση.

Ωστόσο, υπάρχουν κάποιοι περιορισμοί όπου τα δίκτυα που βασίζονται στο Mininet δεν μπορούν να υπερβούν την υπολογιστική ισχύ (CPU) ή το διαθέσιμο εύρος ζώνης σε ένα μοναδικό εξυπηρετητή. Τέλος, το Mininet προς το παρόν δεν μπορεί να εκτελεί μη συμβατές με το Linux εφαρμογές ή OpenFlow switches.

4.4 Προετοιμασία υλοποίησης προσομοίωσης σε Mininet

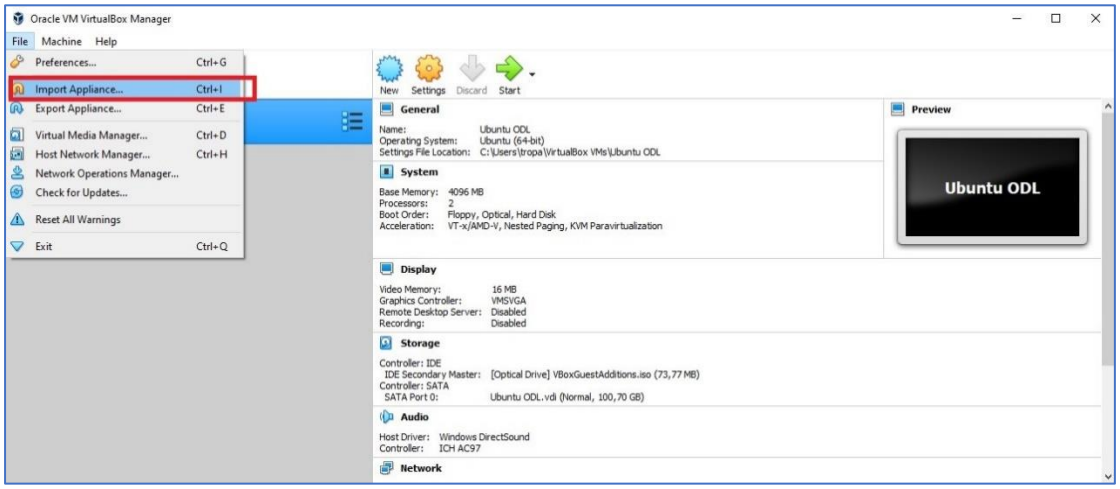
Για την εγκατάσταση του εργαλείου Mininet ώστε να δημιουργήσουμε μια τοπολογία εικονικού δικτύου, υπάρχουν τέσσερις επιλογές:

- Mininet VM Installation
- Native Installation from Source
- Installation from Packages
- Upgrading an existing Mininet Installation

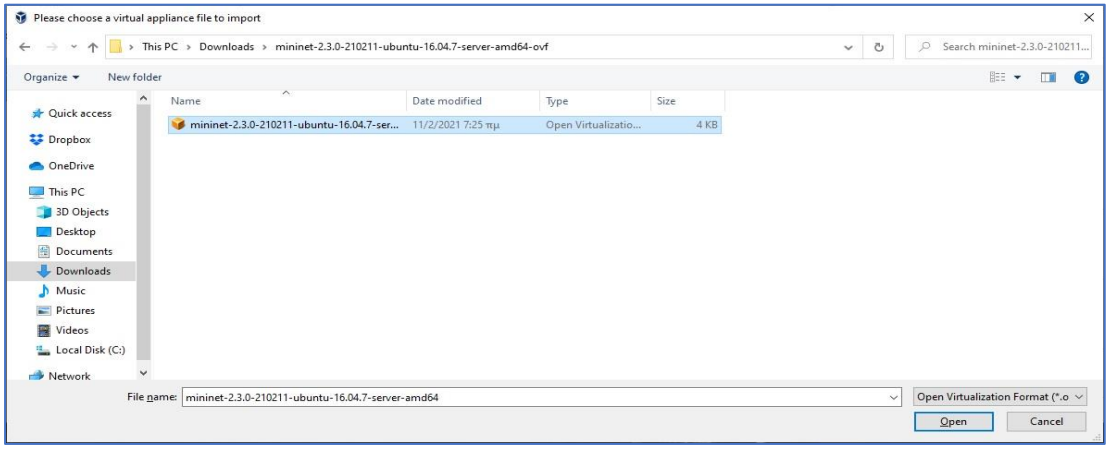
4.4.1 Εγκατάσταση Mininet VM

Για την εγκατάσταση του Mininet VM από το http://mininet.org/download/ επιλέξαμε την έκδοση mininet-2.3.0-210211-ubuntu-16.04.7-server-amd64-ovf. Το αρχείο είναι ένα συμπιεσμένο .ZIP αρχείο το οποίο περιέχει δυο αρχεία στο εσωτερικό του, όπου μετά το κατέβασμα και την αποσυμπίεση του αποθηκεύουμε στο σκληρό δίσκο. Στη συνέχεια θα παρουσιάσουμε βήμα προς βήμα την υλοποίηση του Mininet στο VirtualBox.

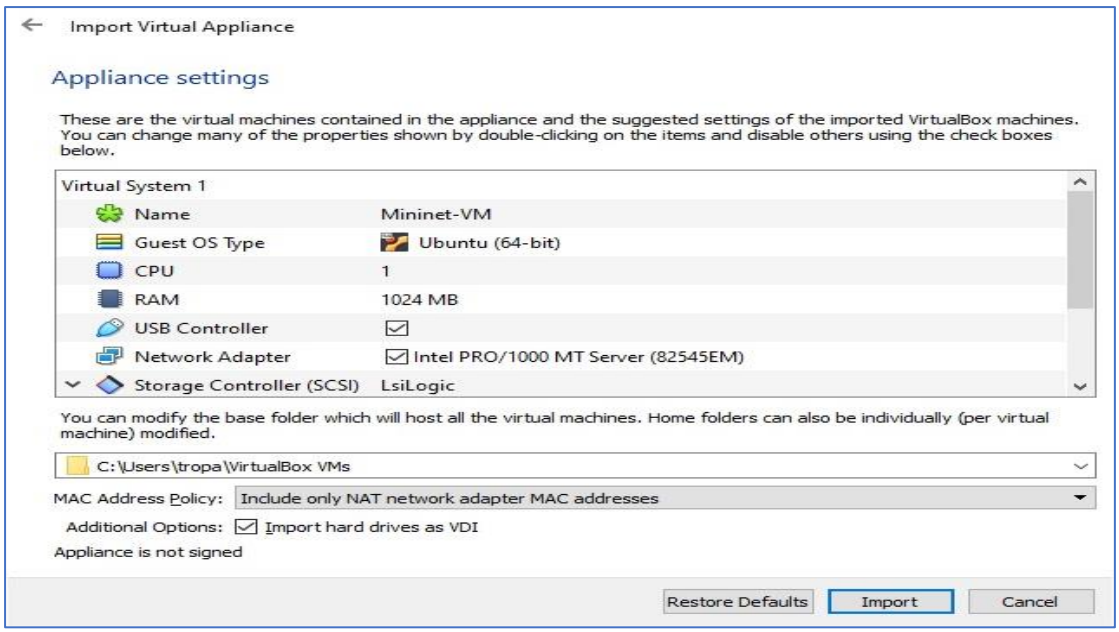
Βήμα 1: Όταν το VirtualBox βρίσκεται σε κατάσταση λειτουργίας επιλέγουμε από το κατάλογο File -> Import Appliance.



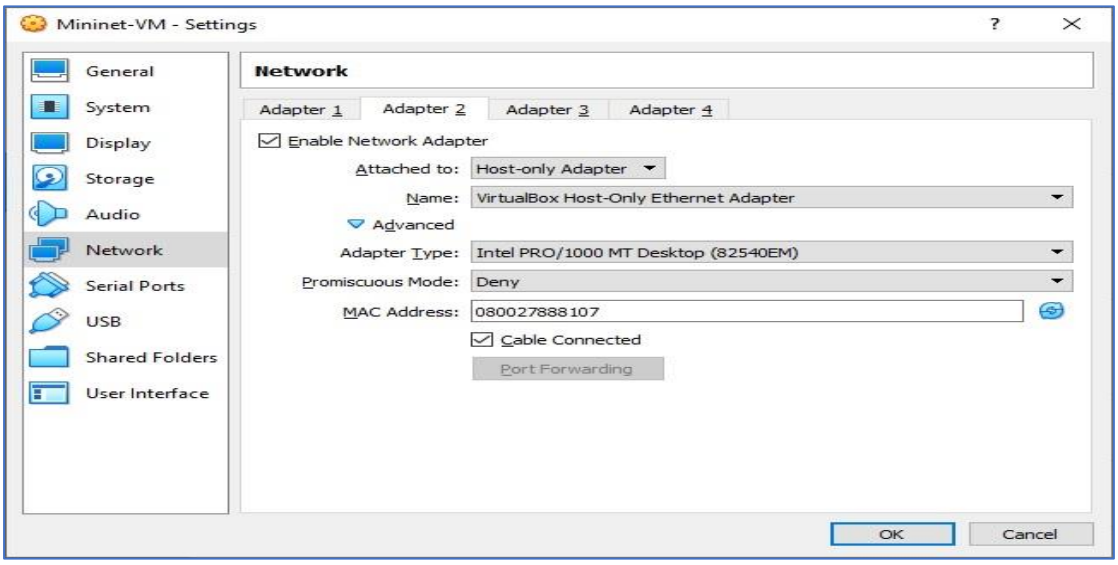
Βήμα 2: Στη συνέχεια εισάγουμε από το φάκελο Downloads το .ovf (Open Virtualization Format) αρχείο. Ανοίγοντας το αρχείο αυτό θα ξεπακετάρει και θα εισάγει μετά από μικρό χρονικό διάστημα το Mininet VM.



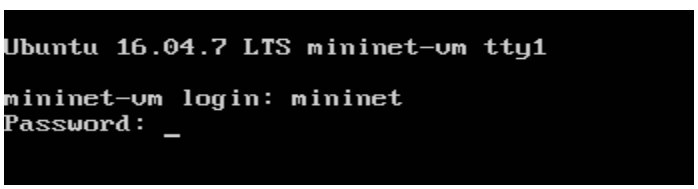
Βήμα 3: Στην ακόλουθη εικόνα φαίνεται η καθορισμένη μνήμη 1GB RAM, 1CPU και το φιλοξενούμενο OS Type Ubuntu (64 bit).



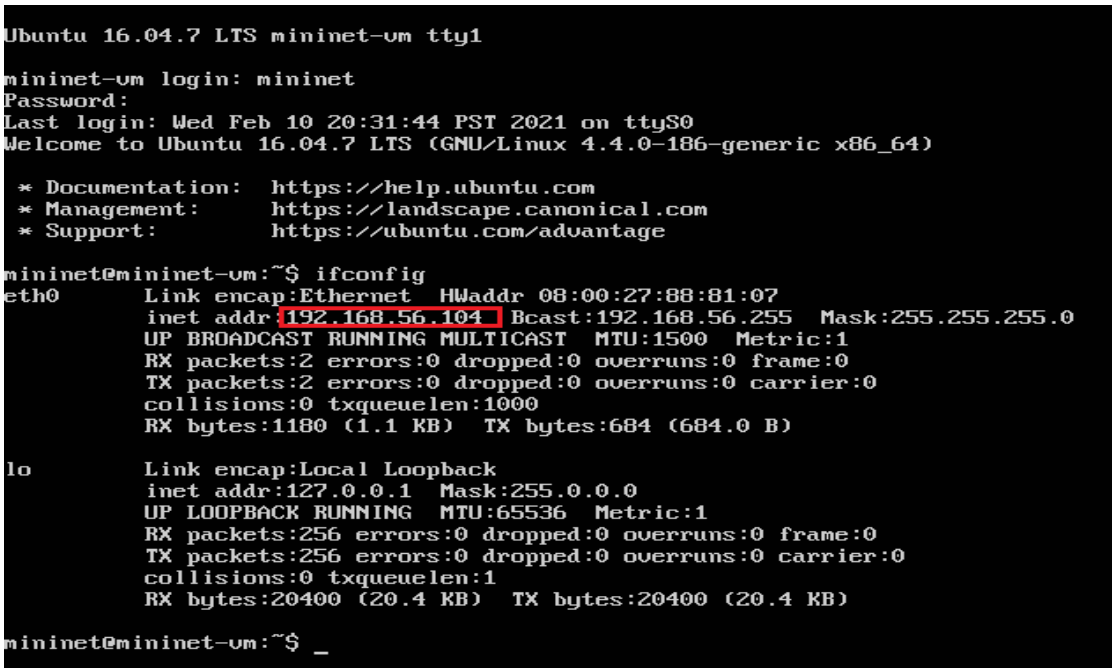
Βήμα 4: Όπως με τη διαρρύθμιση του ODL έτσι και στην ακόλουθη εικόνα ενεργοποιούμε το Network Adapter και επισυνάπτουμε το Host-only Adapter.



Όταν το Mininet VM βρίσκεται σε κατάσταση εκτέλεσης μετά από μικρό χρονικό διάστημα θα εμφανίσει ένα παράθυρο εισόδου με συνθηματικό για το UserID και password *"mininet".*



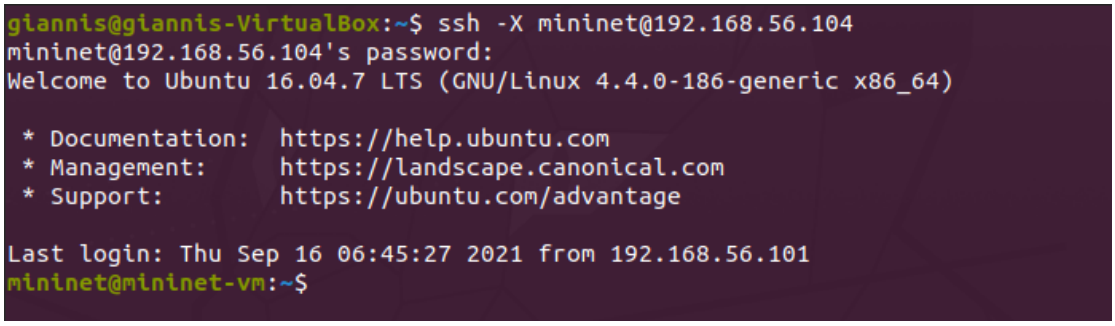
Στη συνέχεια, εκτελούμε την εντολή ifconfig για να δούμε τα διαθέσιμα interfaces στο Mininet VM.



Το Mininet VM βασίζεται στο Ubuntu Server 16.04.7, ο οποίος δε χρησιμοποιεί τα προβλέψιμα ονόματα των network interfaces όπως eth0s3 και eth0s8, οπότε βλέπουμε το eth0 και lo interfaces.

Διακρίνουμε ότι το eth0 είναι συνδεδεμένο με το host-only interface επειδή έχει την IP διεύθυνση 192.168.56.104 όπου είναι η διεύθυνση που καθορίστηκε από το VirtualBox host-only network DHCP server. Συνεπώς, η IP 192.168.56.104 είναι η διεύθυνση του Mininet VM που θα χρησιμοποιήσουμε για να έχουμε πρόσβαση στις εφαρμογές.

Τέλος, συνδέουμε το host computer μας με το Mininet VM μέσω του SSH (Secure Shell) και προωθούμε τον X server όπως φαίνεται στην ακόλουθη εικόνα.



4.4.2 Εφαρμογές και ενσωματωμένα εργαλεία του Mininet

Η διανομή του Mininet VM (2.3.0) που εγκαταστήσαμε φέρει ένα σύνολο από χαρακτηριστικά καθώς και πληθώρα από προεγκατεστημένες εφαρμογές και εργαλεία δικτύωσης τα οποία θα αναλυθούν στη συνέχεια:

Υποστήριξη Python 3: Η έκδοση Mininet 2.3.0 υποστηρίζει την Python 3, ωστόσο μπορεί να κάνει χρήση και της παλαιότερης έκδοσης Python 2.

Για την εγκατάσταση μιας Python 2 mn συντάσσουμε:

Sudo PYTHON=python2 mininet/util/install.sh -n # install Python 2 Mininet

Για την εγκατάσταση μιας Python 3 mn συντάσσουμε:

Sudo PYTHON=python3 mininet/util/install.sh -n # install Python 3 Mininet

Για να εξακριβώσουμε ποια έκδοση Python mn χρησιμοποιείται ως προεπιλογή:

echo py sys.version_info | sudo mn -v output
ή
sudo mn
...
mininet> py sys.version_info

Επίσης μια συλλογή από χρήσιμες εφαρμογές και εργαλεία έχουν υλοποιηθεί χρησιμοποιώντας το Mininet.

GUI - VND (Visual Network Description): Ένα εργαλείο GUI που επιτρέπει την αυτόματη δημιουργία του Mininet και OpenFlow Controllers Scripts.

GUI - MiniEdit: Είναι ένα εργαλείο ανοιχτού κώδικα για τη δημιουργία τοπολογίας. Δίνει την επιλογή στο χρήστη να αποθηκεύσει και να εξάγει την τοπολογία ως Mininet python script.

Mininet Topology Virtualizer: Ένα δικτυακό εργαλείο εικονικοποίησης http://mininet.spear.narmox.com/ της τοπολογίας που δημιουργείται από το Mininet. Χρησιμοποιεί τις εντολές dump και links στη κονσόλα του Mininet και στη συνέχεια κάνουμε επικόλληση το αποτέλεσμα στο εργαλείο και στη συνέχεια πατάμε το render graph.

Wireshark: Λογισμικό για την παρακολούθηση, τον εντοπισμό και την επίλυση

σφαλμάτων καθώς και για την ανάλυση πρωτοκόλλων του δικτύου.

Open vSwitch (OVS) [35]**:** Είναι μια εφαρμογή ανοιχτού κώδικα ενός διανεμημένου εικονικού μεταγωγέα πολλαπλών επιπέδων. Είναι σχεδιασμένη για να επιτρέπει ογκώδη αυτοματοποίηση του δικτύου μέσω προγραμματιστικής επέκτασης, ενώ υποστηρίζει τυπική διαχείριση διεπαφών και πρωτοκόλλων (όπως το NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag). Επιπλέον, είναι σχεδιασμένο να υποστηρίζει διανομή σε πολλαπλούς φυσικούς εξυπηρετητές παρόμοιοι με το διανεμημένο εικονικό μεταγωγέα του VMware vNetwork ή του Cisco Nexus 1000V. Η ακόλουθη εικόνα δείχνει ότι η έκδοση του switch που φιλοξενείται από το Mininet είναι η 2.5.9.

```
mininet@mininet-vm:~$ ovs-dpctl -V
ovs-dpctl (Open vSwitch) 2.5.9
Compiled Jan 28 2021 19:49:45
mininet@mininet-vm:~$
```

dpctl: Είναι ένα εργαλείο γραμμής εντολών (command line tool) το οποίο επιτρέπει τον έλεγχο του OpenFlow switch. Προσθέτει ροές στο πίνακα ροής, αιτείται για τα χαρακτηριστικά και τη κατάσταση του μεταγωγέα καθώς είναι υπεύθυνο και για άλλες ρυθμίσεις.

ovs-ofctl: Είναι ένα εργαλείο γραμμής εντολών για την επιτήρηση και την διαχείριση OpenFlow switches. Μπορεί επίσης να εμφανίσει την υπάρχουσα κατάσταση ενός OpenFlow switch (χαρακτηριστικά, διαρρύθμιση, ροές πίνακα, κ.α.).

ovs-vsctl: Ένα χρηστικό εργαλείο γραμμής εντολών για την αίτηση και τη διαρρύθμιση στον ovs-vswitchd.

iperf: Εργαλείο για τη μέτρηση και το συντονισμό απόδοσης του δικτύου.

4.4.3 Βασικές εντολές και σημαντικές κλάσεις για τη δημιουργία τοπολογιών στο Mininet

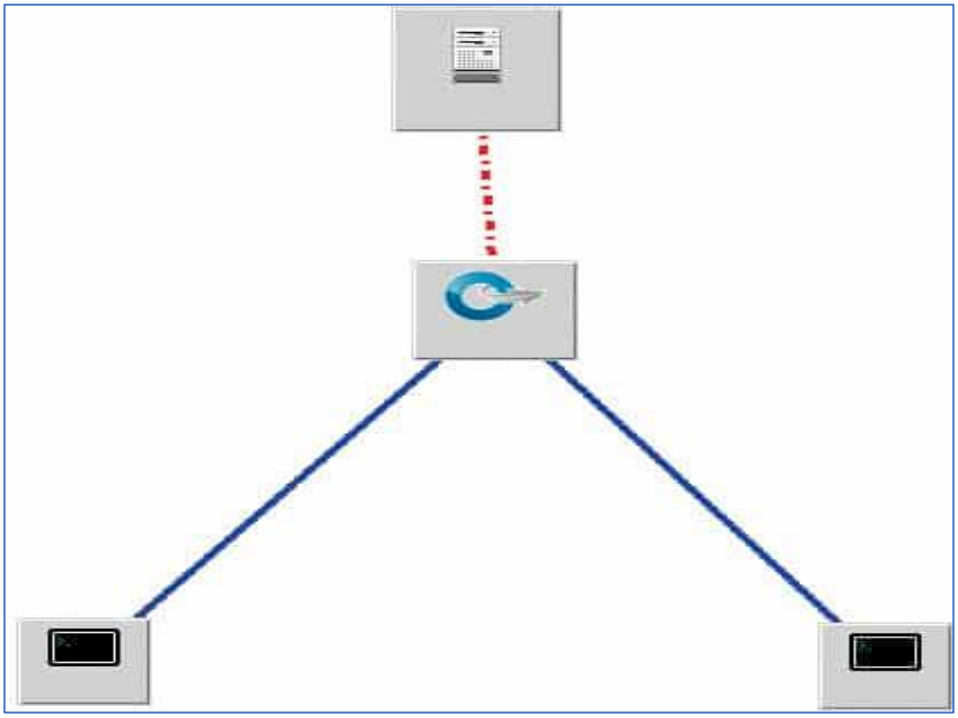
Στην ενότητα αυτή ακολουθούν μερικές βασικές εντολές για την διαμόρφωση και τον έλεγχο μιας τοπολογίας δικτύου SDN στον εξομοιωτή Mininet [36].

- Για την εγκατάσταση του Mininet, συντάσσουμε την ακόλουθη εντολή στο CLI:

  # apt-get install Mininet, to install the Mininet packages on your system

- Για να επαληθεύσουμε την εγκατάσταση του Mininet, δημιουργούμε μια προεπιλεγμένη τοπολογία στο CLI με την ακόλουθη εντολή: ***# mn***

Εικόνα 21. Επαλήθευση της εγκατάστασης της προεπιλεγμένης Mininet topology [36]



Εικόνα 22. Δημιουργία τοπολογίας με τη χρήση του MiniEdit [36]

- Για το άνοιγμα του Wireshark εκτελούμε την εντολή: ***$ sudo wireshark &***

- Με την ακόλουθη εντολή εμφανίζονται όλες οι βασικές εντολές: ***mininet> help***

- Η εντολή για την εμφάνιση των υπάρχων κόμβων του δικτύου είναι: ***mininet> nodes*** Η παραπάνω εντολή θα καταγράψει όλους τους υπάρχων κόμβους της τοπολογίας που δημιουργήθηκε (Εικόνα 23).

- Η εντολή για την εμφάνιση και την καταγραφή των υπάρχων συνδέσμων στο δίκτυο είναι: ***mininet> net***

Όπως φαίνεται παρακάτω (Εικόνα 23) η διεπαφή eth0 του host h1 είναι συνδεδεμένη με τη διεπαφή eth1 του switch1 και η διεπαφή eth0 του host2 είναι συνδεδεμένη με τη διεπαφή eth2 του switch2.

```
mininet> nodes
available nodes are:
h1 h2 s1
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
mininet>
```

Εικόνα 23. Χρήση εντολών nodes και net

- Η εντολή για την εμφάνιση των IP διευθύνσεων και των αναγνωριστικών διεργασίας των κόμβων είναι: ***mininet> dump***

Όπως φαίνεται στην ακόλουθη εικόνα, ο h1 έχει δεσμεύσει την IP 10.0.0.1 με process ID (pid) 1135 και ο h2 έχει δεσμεύσει την IP 10.0.0.2 με process ID (pid) 1140.

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1137>
<Host h2: h2-eth0:10.0.0.2 pid=1140>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=1146>
<Controller c0: 127.0.0.1:6653 pid=1130>
mininet>
```

Εικόνα 24. Χρήση εντολής dump

- Η εντολή με την οποία κάνουμε ping από ένα συγκεκριμένο κεντρικό υπολογιστή σε έναν άλλο κεντρικό υπολογιστή είναι: ***mininet> h1 ping h2***

Κατά των έλεγχο των πακέτων, βλέπουμε ότι το πρώτο ping διήρκησε περισσότερο

(0.516) σε σχέση με τα υπόλοιπα. Αυτό συμβαίνει επειδή οι πίνακες ARP, οι πίνακες MAC κ.α. έχουν αρχικοποιηθεί κατά τη διάρκεια του πρώτου ping.

```
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=0.516 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.104 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.096 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.088 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.098 ms
```

Εικόνα 25. Χρήση εντολής ping από τον h1 στον h2

- Η εντολή για την εμφάνιση των πληροφοριών διεύθυνσης των κόμβων είναι:

  mininet> h1 ifconfig -a

  Η εντολή αυτή θα εμφανίσει την IP address, Broadcast address και MAC address του host h1 όπως φαίνεται στην ακόλουθη εικόνα:

```
mininet> h1 ifconfig -a
h1-eth0   Link encap:Ethernet  HWaddr 1a:14:d3:1a:90:68
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  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:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          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:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet>
```

Εικόνα 26. Πρόσβαση στις λεπτομέρειες του host h1

- Η εντολή για τον έλεγχο της συνδεσιμότητας ανάμεσα στους κεντρικούς υπολογιστές είναι : ***mininet> pingall***

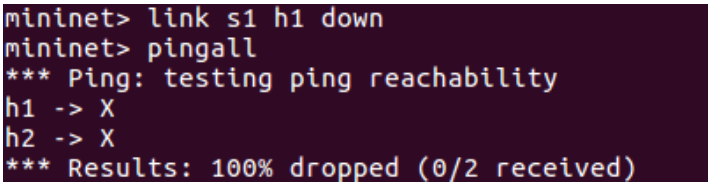
  Με την παραπάνω εντολή κάθε κεντρικός υπολογιστής στο δίκτυο θα κάνει ping σε κάθε άλλο κεντρικό υπολογιστή του δικτύου. Όπως φαίνεται στη παρακάτω εικόνα τα επιτυχημένα pings δείχνουν ότι όλοι οι σύνδεσμοι στο δίκτυο είναι ενεργοί.

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

Εικόνα 27. Χρήση της εντολής pingall

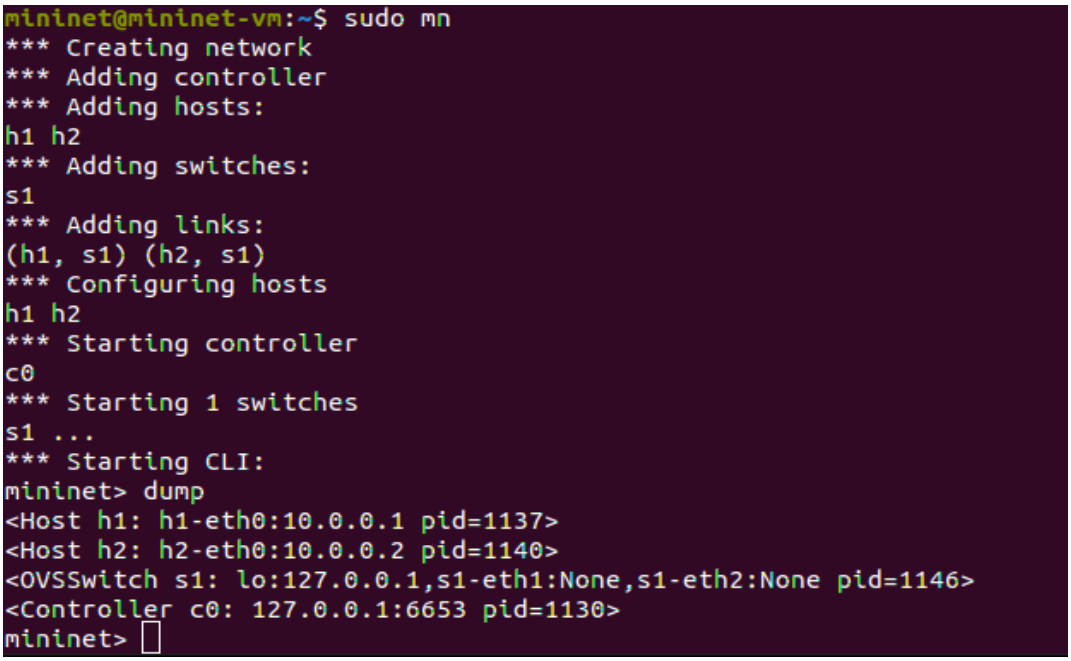
- Η εντολή για τη διακοπή μιας σύνδεσης είναι: ***mininet> link s1 h1 down***

  Η παραπάνω εντολή θα διακόψει τη σύνδεση ανάμεσα στο switch s1 και τον host h1. Αυτό το διαπιστώνουμε όταν εκτελέσουμε και πάλι την εντολή pingall θα δούμε ότι τα δυο pings είναι ανεπιτυχή όπως φαίνεται στην ακόλουθη εικόνα.



Εικόνα 28. Διακοπή σύνδεσης ανάμεσα στο host h1 και switch s1

- Η εντολή για τη δημιουργία μιας προεπιλεγμένης τοπολογίας είναι: ***$sudo mn***



Εικόνα 29. Χρήση της εντολής sudo mn για τη δημιουργία τοπολογίας

- Με την ακόλουθη εντολή δημιουργούμε μια προκαθορισμένη τοπολογία, αυξάνοντας τον αριθμό των κεντρικών υπολογιστών (hosts 3).

  #sudo mn –topo single,3

  Όπως φαίνεται στην ακόλουθη εικόνα έχει δημιουργηθεί μια τοπολογία που αποτελείται από έναν μεταγωγέα συνδεδεμένο με τρεις κεντρικούς υπολογιστές.

```
root@hp-HP-431-Notebook-PC:~# mn --topo single,3
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** 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:
mininet>
```

Εικόνα 30. Δημιουργία προκαθορισμένης τοπολογίας στο Mininet

- Η εντολή για την εκτέλεση παλινδρόμησης είναι: ***#sudo mn -- test pingpair***

  Η παραπάνω εντολή χρησιμοποιείται για τη δημιουργία ενός δικτύου με μια προκαθορισμένη τοπολογία, την εκτέλεση της λειτουργίας pingall και την διακοπή του δικτύου. Η εντολή αυτή χρησιμοποιείται για να ελέγξουμε πως λειτουργεί το Mininet.

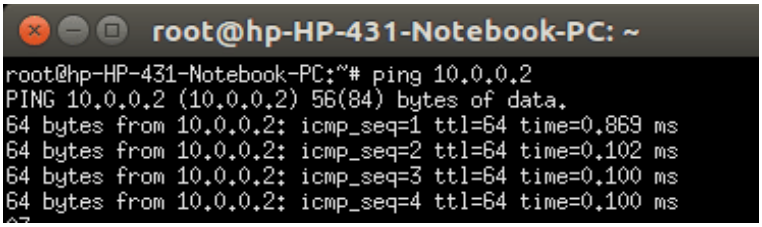
```
root@hp-HP-431-Notebook-PC:~# mn --test pingpair
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
```

Εικόνα 31. Δημιουργία τοπολογίας και επικύρωση ελέγχων

- Η εντολή για το άνοιγμα του xterm παραθύρου είναι: ***mininet> h1 xterm***

  Η παραπάνω εντολή χρησιμοποιείται για την εκκίνηση ενός εξωτερικού και ξεχωριστού τερματικού. Η εντολή ανοίγει ένα xterm παράθυρο, το οποίο είναι συγκεκριμένα για ένα κόμβο στο δίκτυο. Διάφορες συγκεκριμένες λειτουργίες μπορούν να εφαρμοστούν σε ένα κόμβο σε αυτό το παράθυρο. Όπως φαίνεται

στην ακόλουθη εικόνα το παράθυρο h1 xterm χρησιμοποιείται για να κάνει ping στο h2 (10.0.0.2).



Εικόνα 32. Έλεγχος της διαθεσιμότητας του host h2 από το host h1

- Η εντολή για παροχή μιας καθυστέρησης των 10 ms και την τοποθέτηση ενός εύρους ζώνης των 10Mbps σε όλους τους συνδέσμους είναι:
  #sudo mn –link tc,bw=10,delay=10ms

  Εάν η καθυστέρηση για κάθε σύνδεσμο είναι 10 ms, το round trip time (RTT) θα είναι περίπου 40 ms, αφού η ICMP αίτηση διασχίζει δυο συνδέσμους (ένα για το μεταγωγέα, ένα για το προορισμό) και η ICMP απάντηση διασχίζει δυο συνδέσμους για να επιστρέψει.

- Η εντολή για την έξοδο από το CLI του Mininet είναι: ***mininet> exit***

- Η εντολή για το καθαρισμό του Mininet σε περίπτωση δυσλειτουργίας είναι:
  mininet> sudo mn -c

Με τη βοήθεια της ενσωματωμένης γλώσσας προγραμματισμού Python καθώς και των διαθέσιμων παραμετροποιημένων δικτυακών τοπολογιών που διαθέτει το Mininet, μπορεί ο κάθε χρήστης να δημιουργήσει μια απλή, σύνθετη ή ευέλικτη τοπολογία ανάλογα με τις απαιτήσεις του. Στη συνέχεια παρουσιάζονται σημαντικές κλάσεις και μεταβλητές.

⇨ **Topo:** Η βασική κλάση για τις τοπολογίες στο Mininet.

⇨ **addSwitch() :** Προσθέτει ένα μεταγωγέα στη τοπολογία και επιστρέφει την ονομασία του.

⇨ **addHost() :** Προσθέτει ένα κεντρικό υπολογιστή στη τοπολογία και επιστρέφει την ονομασία του.

⇨ **addLink()** : Προσθέτει έναν αμφίδρομο σύνδεσμο από το κόμβο 1 στο κόμβο 2.

- **Mininet:** Η κύρια κλάση για τη δημιουργία και τη διαχείριση του δικτύου.
- **start() :** Ενεργοποίηση λειτουργίας του δικτύου.
- **stop()** : Διακοπή λειτουργίας του δικτύου.
- **pingAll() :** Έλεγχος συνδεσιμότητας των κόμβων εκτελώντας ping μεταξύ όλων των κόμβων.

5. Ανάπτυξη σεναρίων προσομοίωσης OpenDaylight SDN

Στο κεφάλαιο αυτό θα παρουσιασθούν τρεις διαφορετικές προσεγγίσεις σεναρίων σε περιβάλλον εικονικού δικτύου με τη χρήση του εξομοιωτή Mininet και Mininet-WiFi, σε συνεργασία με τον ODL controller. Για να επιτευχθεί αυτό πρέπει οι δυο εικονικές μηχανές να βρίσκονται σε κατάσταση λειτουργίας (running state). Η υλοποίηση του πρώτου σεναρίου προσομοίωσης αφορά απλή διαμόρφωση δικτύωσης SDN με διάταξη γραμμικής (Linear) τοπολογίας αποτελούμενη από τον ODL controller, τρία switches και τρεις hosts. Η υλοποίηση του δεύτερου σεναρίου προσομοίωσης αφορά διαμόρφωση δικτύωσης SDN με διάταξη τοπολογίας δένδρου (Tree) αποτελούμενη από τον ODL controller, τρία switches και τέσσερις hosts. Τέλος, η υλοποίηση του τρίτου σεναρίου προσομοίωσης αφορά ασύρματη τοπολογία δικτύωσης SDN με κινητικότητα των τερματικών σταθμών (Stations), που περιλαμβάνει τον ODL controller, δύο Access Points και δύο κινητούς τερματικούς σταθμούς.

5.1 Υλοποίηση 1ου σεναρίου προσομοίωσης σε Linear OpenDaylight SDN Topology

Για να δημιουργήσουμε μια εικονική τοπολογία δικτύου SDN με διάταξη γραμμικής (Linear) τοπολογίας, συντάσσουμε την εντολή:

```
mininet@mininet-vm:~$ sudo mn --topo linear, 3 --mac --controller=remote,
ip=192.168.56.101, port=6633 --switch ovs, protocols=OpenFlow13
```

όπου

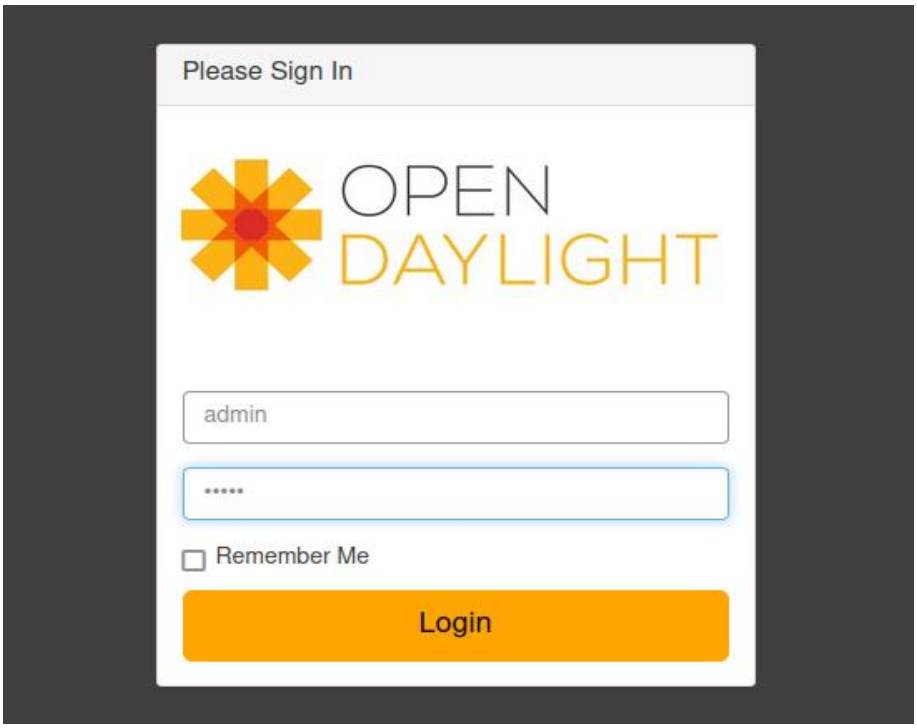
- **topo:** Η linear ορίζει μια τοπολογία αποτελούμενη από τρία switches όπου το κάθε switch είναι συνδεδεμένο με ένα host.
- **mac:** Κατανέμει τις MAC addresses για κάθε host σύμφωνα με την IP διεύθυνση τους.
- **ip**: Ορίζει την IP address για τον απομακρυσμένο ελεγκτή (192.169.56.101).
- **controller:** Είναι η IP address του ODL όπου οι εικονικοί μεταγωγείς είναι συνδεδεμένοι.
- **switch**: Παράμετρος για την ταυτοποίηση του τύπου του switch (OVS).
- **protocols:** Ο switch είναι συμβατός με την έκδοση του πρωτοκόλλου OpenFlow13.

Όταν το εικονικό δίκτυο υλοποιηθεί, εκτελούμε την εντολή **pingall** για να ελέγξουμε αν υπάρχει επικοινωνία ανάμεσα στους hosts.

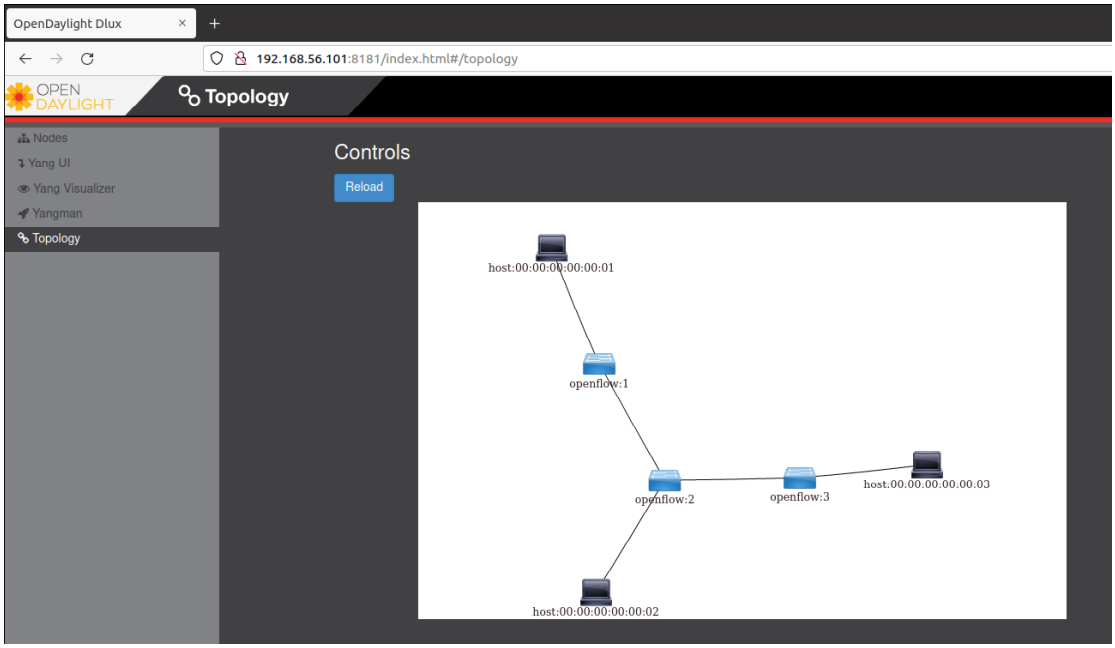
```
mininet@mininet-vm:~$ sudo mn --topo linear,3 --mac --controller=remote,ip=192.1
68.56.101,port=6633 --switch ovs,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

Εικόνα 33. Δημιουργία τοπολογίας εικονικού δικτύου μέσω του CLI στο Mininet

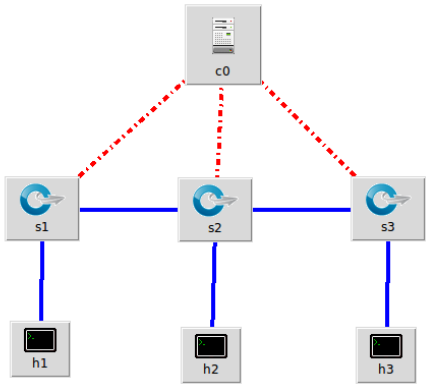
Στη συνέχεια ανοίγουμε έναν περιηγητή και χρησιμοποιούμε το web user interface του OpenDaylight εισάγοντας το URL http://192.168.56.101:8181/index.html.



Εικόνα 34. Είσοδος στον OpenDaylight controller



Εικόνα 35. Κεντρική σελίδα OpenDaylight DLUX

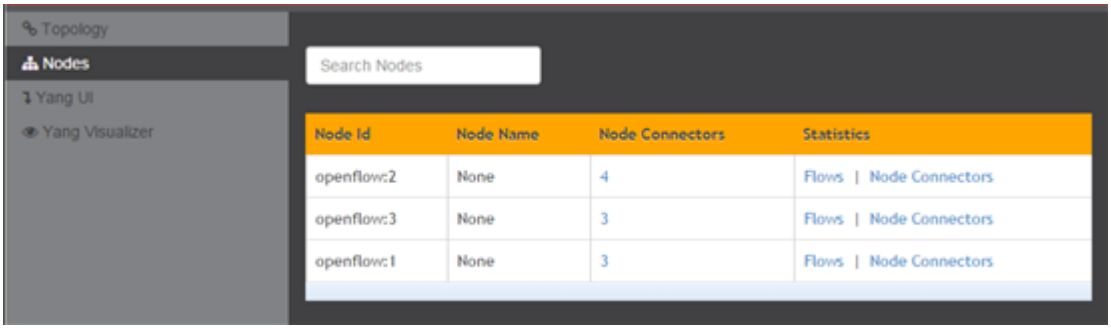


Εικόνα 36. Γενική σχηματική αναπαράσταση Linear Topology

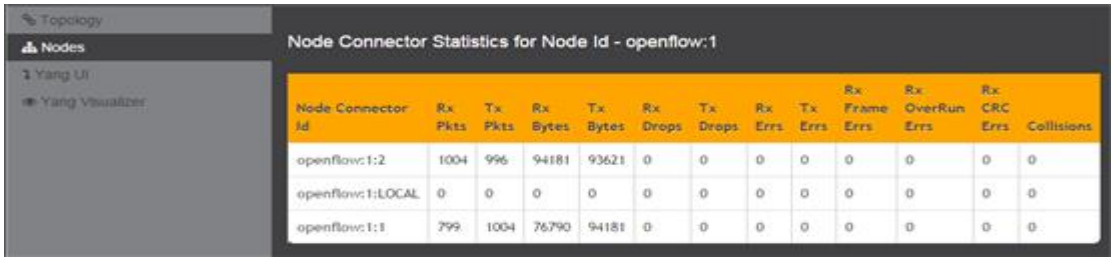
Ο OpenDaylight DLUX παρέχει στο χρήστη ορατότητα για τη κατάσταση της τοπολογίας του δικτύου καθώς και χρήσιμες πληροφορίες για τις στατιστικές του δικτύου.

Συνεχίζοντας την πλοήγηση μέσω του graphical user interface (GUI) παρατηρούμε πάνω αριστερά μια λίστα με τα χαρακτηριστικά του OpenDaylight που έχουμε εγκαταστήσει προηγουμένως.

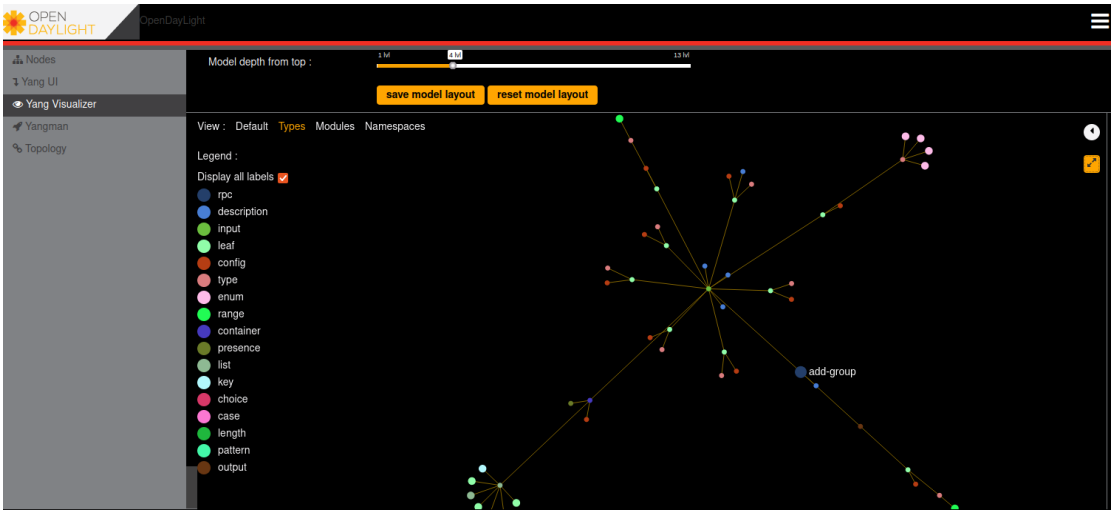
Από την επιλογή Nodes μπορούμε να βλέπουμε πληροφορίες για το κάθε switch:



Στη συνέχεια ανοίγοντας τον σύνδεσμο Node Connectors βλέπουμε σε κάθε σειρά πληροφορίες για κάθε port στο εκάστοτε switch:



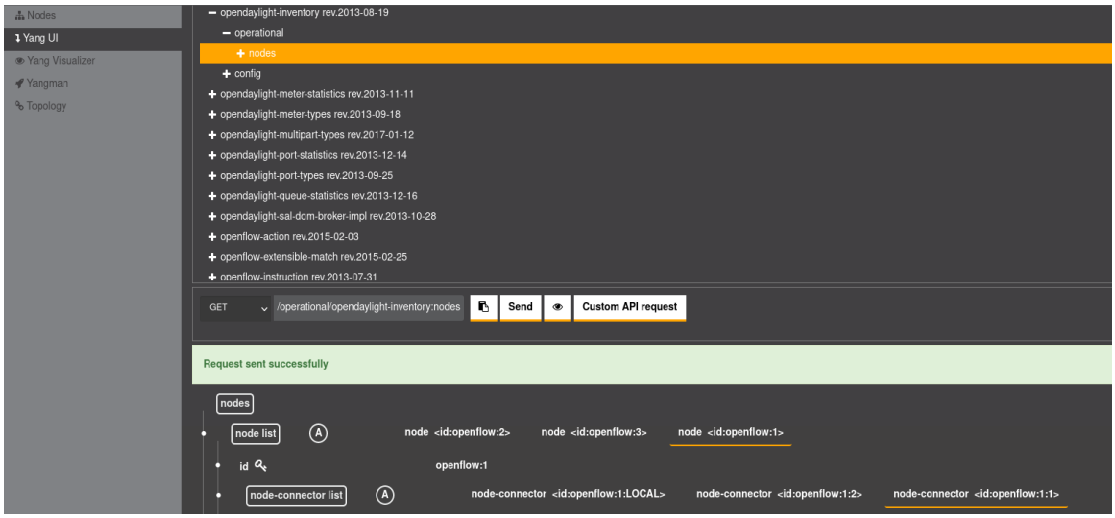
Άλλο ένα επιπλέον χαρακτηριστικό του ODL controller είναι το Yang Virtualizer όπου απεικονίζονται όλοι οι τύποι δεδομένων (data types) στην τοπολογία μας:



Εικόνα 37. Σχηματική αναπαράσταση Yang Virtualizer

Ένα από τα πιο σημαντικά χαρακτηριστικά του OpenDaylight DLUX είναι το YANG UI. Το YANG είναι μια δομή μοντέλου δεδομένων. Επιλέγουμε το YANG UI από τη λίστα χαρακτηριστικών στο αριστερό τμήμα. Το δεξί τμήμα χωρίζεται σε δύο μέρη. Στη κορυφή του YANG (εικόνα 38) εμφανίζονται όλα τα APIs και subAPIs σε σχηματισμό δένδρου και όλες οι διαθέσιμες λειτουργίες (GET, POST, PUT και DELETE). Οι λειτουργία GET παίρνει δεδομένα από τον ODL, οι λειτουργίες PUT και POST στέλνουν δεδομένα στον ODL για αποθήκευση και τέλος η λειτουργία DELETE

στέλνει δεδομένα στον ODL για διαγραφή. Στο κάτω τμήμα της σελίδας του YANG εμφανίζονται οι καταχωρήσεις σύμφωνα με το επιλεγμένο subAPI.



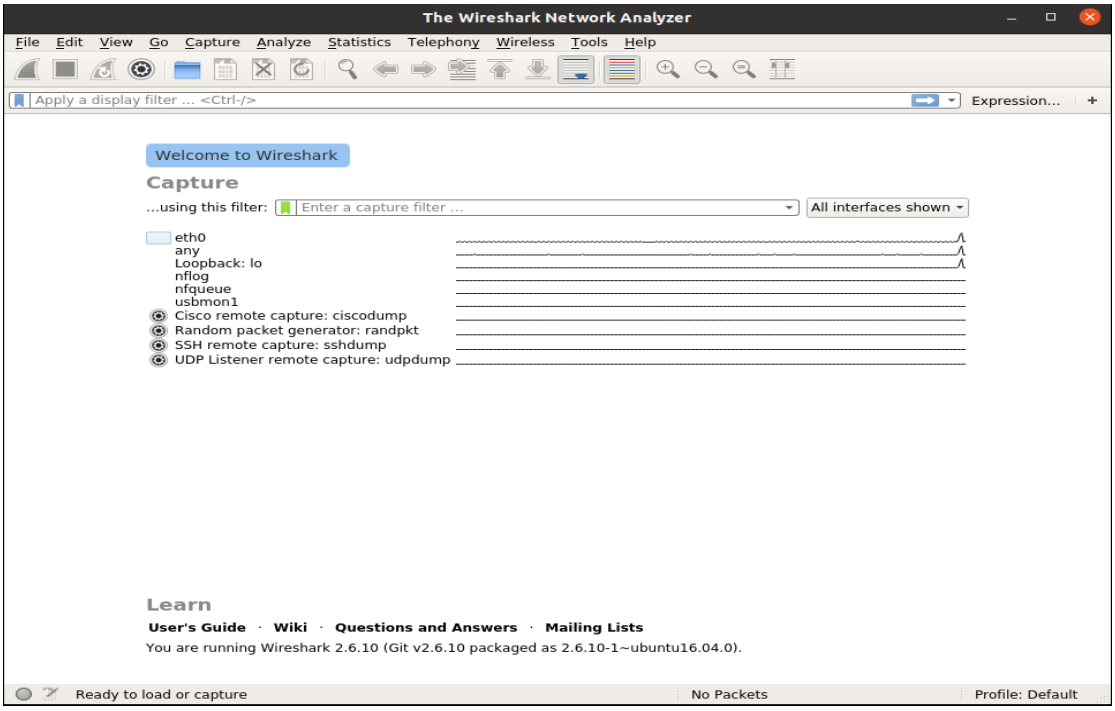
Εικόνα 38. Σχηματική αναπαράσταση σελίδας YANG UI

Εμβαθύνοντας περισσότερο στο πώς λειτουργούν οι SDN controllers και τα switches, θα παρουσιάσουμε στη συνέχεια πώς γίνεται η ανταλλαγή των OpenFlow μηνυμάτων ανάμεσα στον controller και τα switches στο SDN δίκτυο. Ο πιο εύκολος τρόπος να δούμε τα OpenFlow μηνύματα είναι να χρησιμοποιήσουμε το εργαλείο Wireshark στο Mininet VM και να δεσμεύσουμε δεδομένα στη διεπαφή που είναι συνδεδεμένη με το host-only network (eth0).

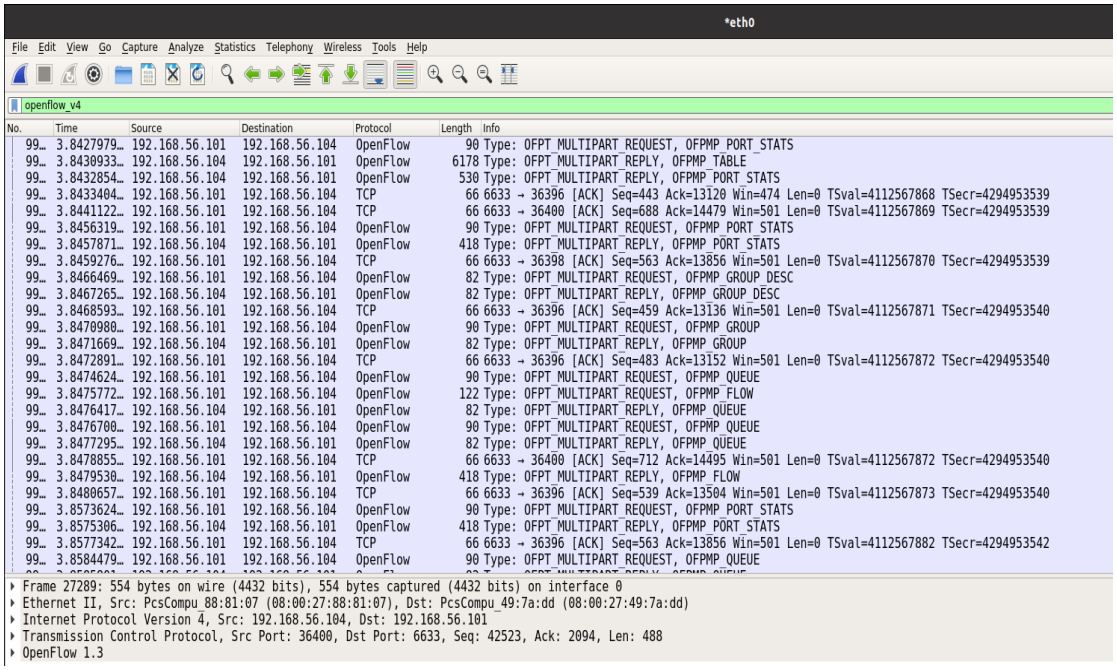
Ανοίγοντας και πάλι ένα νέο τερματικό παράθυρο συνδεόμαστε με το Mininet VM μέσω του SSH πρωτοκόλλου και εν συνεχεία εκτελούμε την ακόλουθη εντολή για την εκκίνηση του Wireshark:

```
giannis@giannis-VirtualBox:~$ ssh -X mininet@192.168.56.104
mininet@192.168.56.104's password:
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.4.0-186-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Last login: Tue Oct  5 12:06:32 2021 from 192.168.56.101
mininet@mininet-vm:~$ sudo wireshark &
```

Ακολούθως μετά από λίγα δευτερόλεπτα ανοίγει αυτόματα η εφαρμογή του Wireshark:



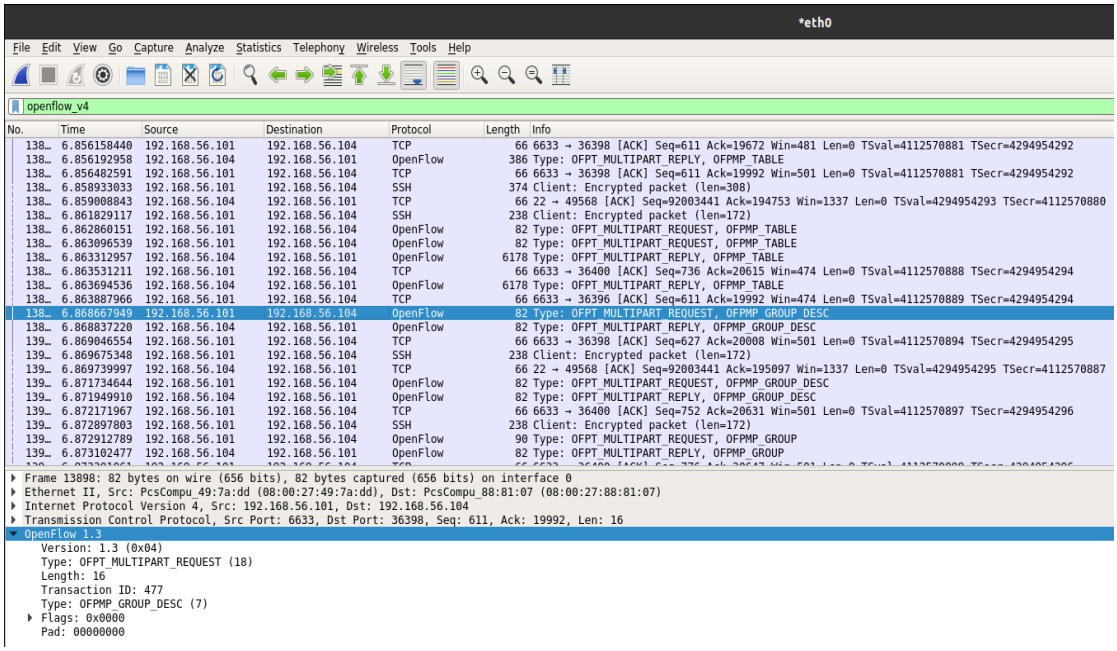
Διαμορφώνουμε ένα φίλτρο εμφάνισης των OpenFlow μηνυμάτων με τη λέξη “openflow_v4” και στη συνέχεια το πλήκτρο apply:



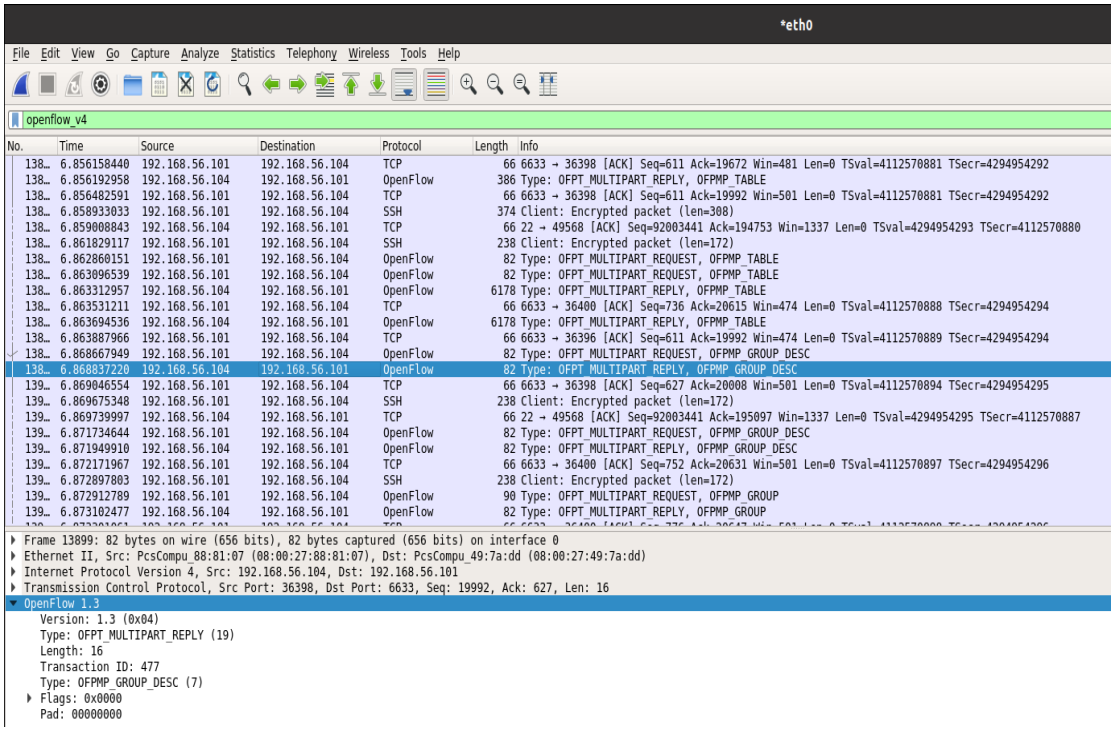
Όπως βλέπουμε από την παραπάνω εικόνα, το Wireshark επιβεβαιώνει ότι υπάρχει επικοινωνία μεταξύ του ODL controller (192.168.56.101) και της εικονικής τοπολογίας δικτύου από το Mininet (192.168.56.104). Τα πακέτα που ανταλλάσσονται είναι τύπου OFPT_MULTIPART.

Η επικοινωνία αρχίζει κατά τη διάρκεια της τριμερούς χειραψίας TCP, όπου ο ODL controller στέλνει αίτημα σε κάθε switch και κατά συνέπεια το κάθε switch απαντά.

Τα πακέτα OFPT_MULTIPART_REQUEST αποτελούνται από τα εξής χαρακτηριστικά: Version, Type, Length και Transaction_ID.



Τα πακέτα OFPT_MULTIPART_REPLY αποτελούνται επίσης από τα χαρακτηριστικά: Version, Type, Length και Transaction_ID.



5.2 Υλοποίηση 2ου σεναρίου προσομοίωσης σε Tree OpenDaylight SDN Topology

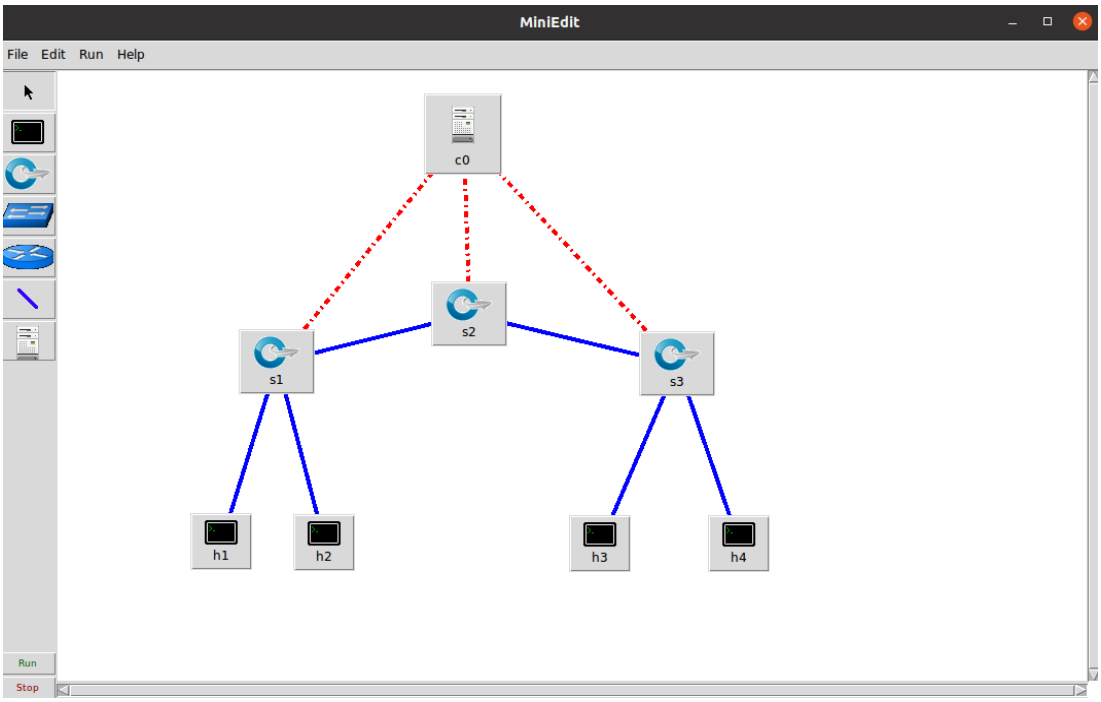
Για τη δημιουργία μιας Tree Network Topology θα προσεγγίσουμε διαφορετικά το πείραμά μας στη περίπτωση αυτή, αξιοποιώντας το ενσωματωμένο εργαλείο **Miniedit** του Mininet.

Εφόσον έχουμε κάνει τις κατάλληλες διαρρυθμίσεις, όπως έχουμε υποδείξει, από το CLI του Mininet εκτελούμε τις ακόλουθες εντολές για την εκκίνηση του Miniedit:

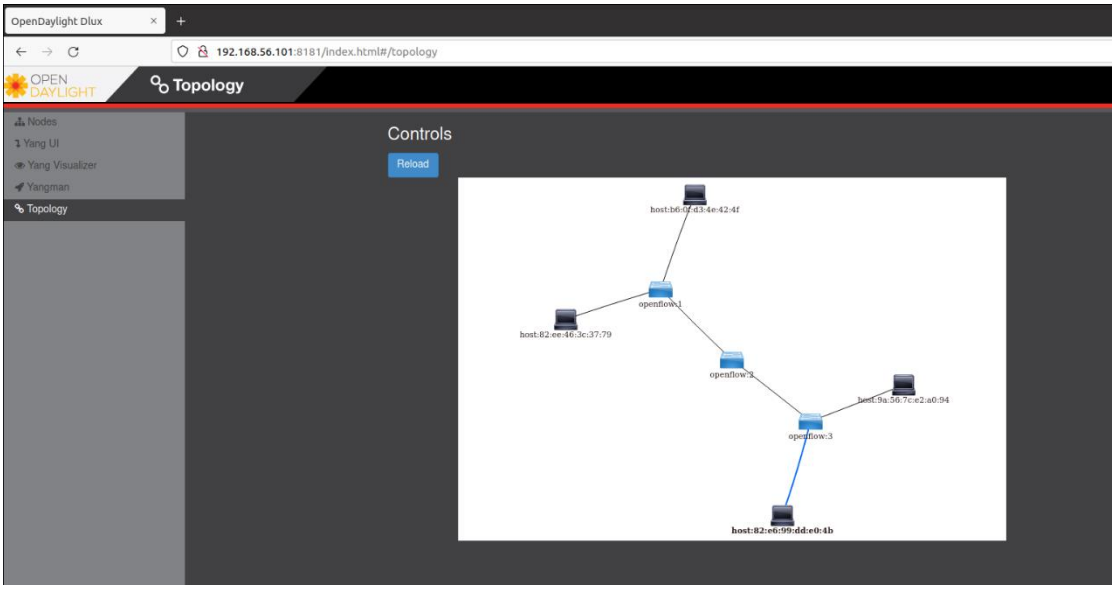
```
mininet@mininet-vm:~$ cd mininet
mininet@mininet-vm:~/mininet$ cd examples
mininet@mininet-vm:~/mininet/examples$ python ./miniedit.py
```

Αφού εισέλθουμε στο Miniedit είμαστε έτοιμοι να υλοποιήσουμε την ζητούμενη Tree Network Topology.

Εικόνα 39. Κεντρική σελίδα του Miniedit



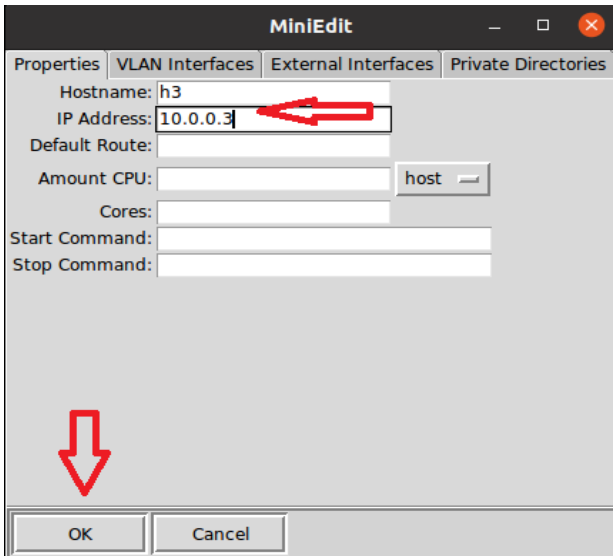
Εικόνα 40. Υλοποίηση Tree Topology στο Miniedit

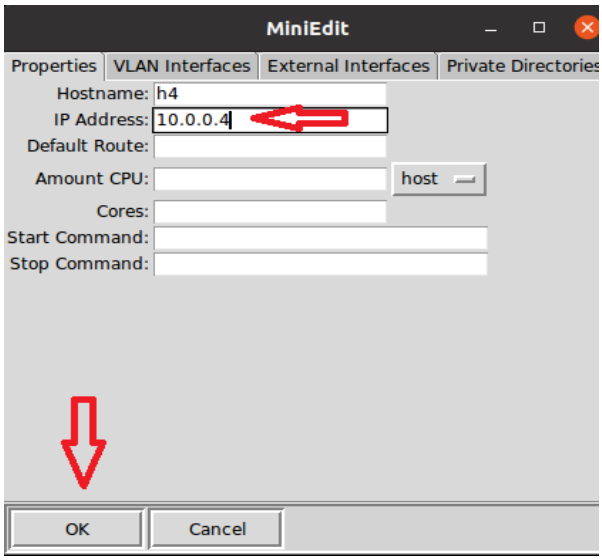


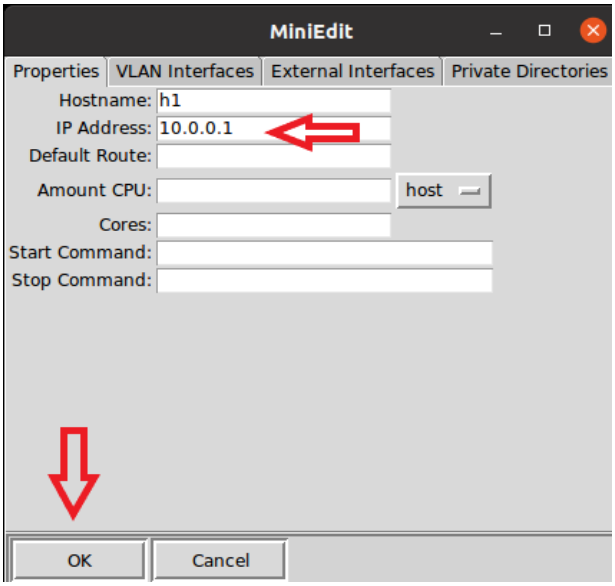
Εικόνα 41. Σχηματική αναπαράσταση Tree Topology στο OpenDaylight DLUX

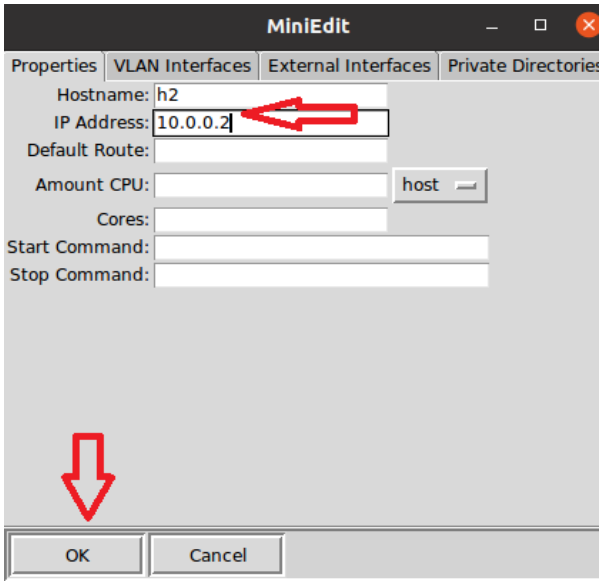
Στη συνέχεια θα παρουσιάσουμε βήμα προς βήμα την κατάλληλη ρύθμιση του εκάστοτε host και του controller για την σωστή λειτουργία του δικτύου μας.

Βήμα 1: Απόδοση IP Address σε κάθε host.

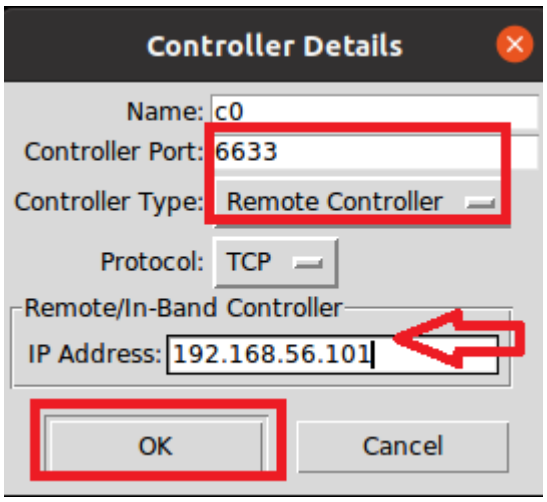




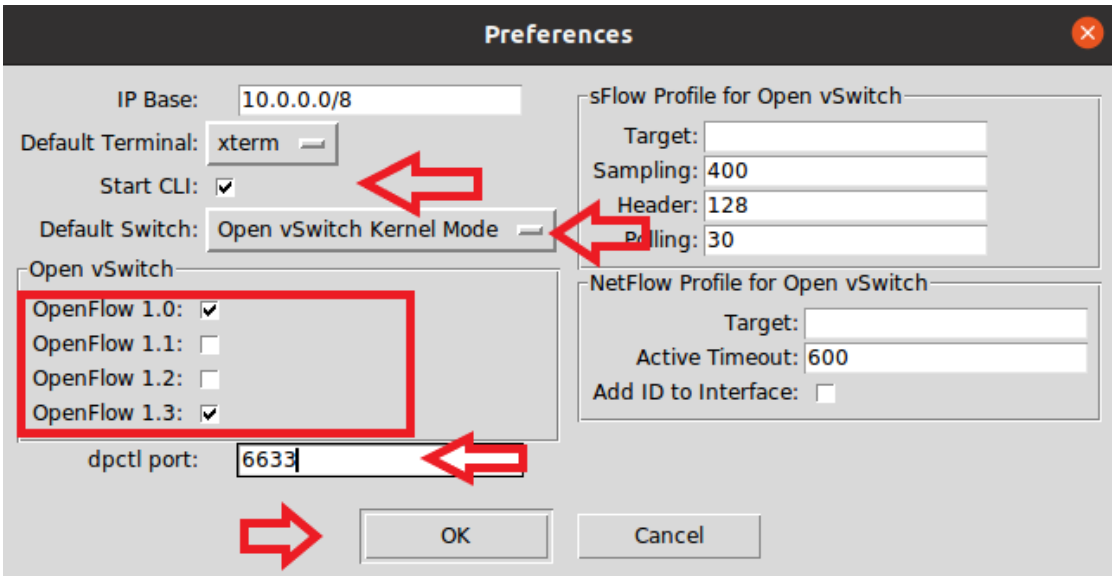




Βήμα 2: Διαρρύθμιση του controller.

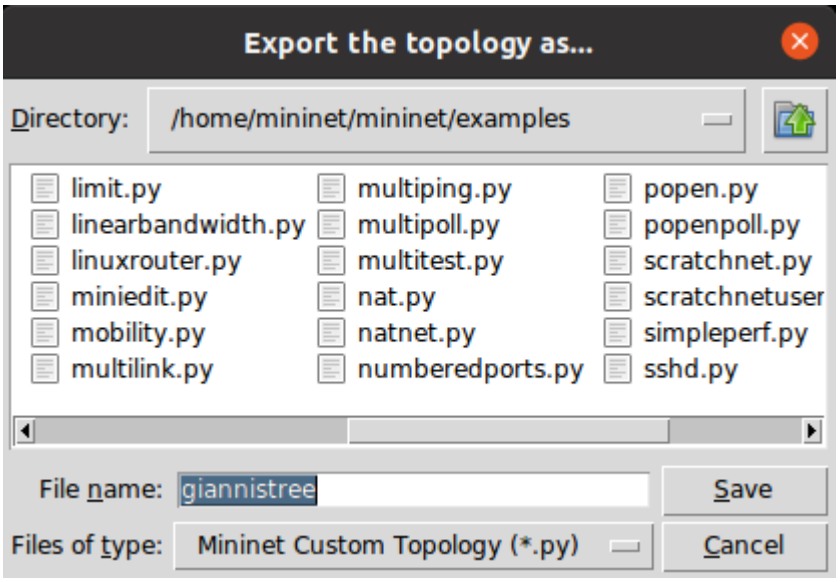


Βήμα 3: Διαρρύθμιση των Preferences.



Τέλος, εκτελούμε από τις επιλογές κάτω αριστερά **Run** για την εκκίνηση της λειτουργίας της τοπολογίας του δικτύου μας.

Στη συνέχεια αυτού του πειράματος, επιλέγουμε από το εργαλείο Miniedit File -> Export Level 2 Script και αποθηκεύουμε τον κώδικα (ΠΑΡΑΡΤΗΜΑ -Β -Β1) της τοπολογίας μας στο ακόλουθο μονοπάτι : /home/mininet/mininet/examples.



Εικόνα 42. Αποθήκευση και εξαγωγή της τοπολογίας μας σε Python Script

Για να θέσουμε σε λειτουργία το δίκτυο που δημιουργήσαμε εκτελούμε:

mininet@mininet-vm:~/mininet/examples$ ***sudo python giannistree.py***

```
mininet@mininet-vm:~/mininet/examples$ sudo python giannistree.py
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h1 h2 h3 h4
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet>
```

Εικόνα 43. Δημιουργία της Tree Topology με την εντολή sudo python

Τελευταίο βήμα για την επαλήθευση της σωστής λειτουργίας της τοπολογίας μας είναι να ελέγξουμε την συνδεσιμότητα μεταξύ των κεντρικών υπολογιστών.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

Εικόνα 44. Έλεγχος συνδεσιμότητας μεταξύ των hosts

5.3 Υλοποίηση 3ου σεναρίου προσομοίωσης σε Mobility OpenDaylight SDN Topology

Για την υλοποίηση του τελευταίου σεναρίου μας θα κινηθούμε διαφορετικά συγκριτικά με τα προηγούμενα σενάρια, καθώς θα δημιουργήσουμε μια πιο σύνθετη τοπολογία με τη συνεργασία του εξομοιωτή **Mininet-WiFi.** Για την επίτευξη αυτού του σεναρίου θα πρέπει πρωτίστως να εγκαταστήσουμε τον εξομοιωτή δικτύου Mininet-WiFi όπου θα αναλύσουμε στη συνέχεια.

<u>Σύντομη προεπισκόπηση Mininet-WiFi:</u>

Το Mininet-WiFi είναι ένα βελτιωμένο λογισμικό του Mininet για ασύρματο περιβάλλον (Wireless Environment). Υποστηρίζει ασύρματους σταθμούς (Wi-Fi

Stations) και σημεία πρόσβασης (Access Points). Το σημείο που το διαφοροποιεί είναι ότι προσθέτει ασύρματα χαρακτηριστικά (wifi features) και ο χρήστης μπορεί να εργαστεί όπως και με το Mininet.

Εγκατάσταση Mininet-WiFi:

Για την εγκατάσταση του Mininet-WiFi εκτελούμε κατά σειρά τις ακόλουθες εντολές.

Βήμα 1: $ sudo apt-get install git

Βήμα 2: $ git clone https://github.com/intrig-unicamp/mininet-wifi

Βήμα 3: $ cd mininet-wifi

Βήμα 4: $ sudo util/install.sh -Wlnfv

όπου

-W: wireless dependencies
-l: wmediumd
-n: mininet-wifi dependencies
-f: OpenFlow
-v: OpenvSwitch

Οι παρακάτω εικόνες αναπαριστούν τη διαδικασία εγκατάστασης του εξομοιωτή Mininet-Wi-Fi για ασύρματα δίκτυα καθοριζόμενα από λογισμικό (Software-Defined Wireless Networks):

```
giannis@giannis-VirtualBox:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi libgstreamer-plugins-bad1.0-0 libllvm11
  libva-wayland2
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs
  git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 5465 kB of archives.
After this operation, 38,4 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

```
giannis@giannis-VirtualBox:~$ git clone https://github.com/intrig-unicamp/mininet-wifi
Cloning into 'mininet-wifi'...
remote: Enumerating objects: 25095, done.
remote: Counting objects: 100% (444/444), done.
remote: Compressing objects: 100% (303/303), done.
remote: Total 25095 (delta 283), reused 267 (delta 141), pack-reused 24651
Receiving objects: 100% (25095/25095), 22.65 MiB | 1.89 MiB/s, done.
Resolving deltas: 100% (17565/17565), done.
giannis@giannis-VirtualBox:~$ cd mininet-wifi
giannis@giannis-VirtualBox:~/mininet-wifi$ sudo util/install.sh -Wlnfv
```

Χρήσιμες εντολές Mininet-WiFi:

Για την εκκίνηση μιας minimal topology εκτελούμε στο CLI:

$sudo mn --wifi

Για την εμφάνιση επιλογών εκτελούμε:

mininet-wifi> help

Για την εμφάνιση των κόμβων εκτελούμε:

mininet-wifi> nodes

Σε περίπτωση που το πρώτο string που πληκτρολογούμε στο Mininet-Wifi CLI είναι Station, AP ή Controller, η εντολή εκτελείται σε αυτό το κόμβο. Εκτελούμε σε ένα Station για να εμφανίσουμε τις διεπαφές sta-wlan0 και loopback(lo):

mininet-wifi> sta1 ifconfig -a

Αντιθέτως, η εντολή για ένα ap:

mininet-wifi> ap1 ifconfig -a

Για τη συλλογή πληροφοριών από το nodes.params:

py sta1.params

Για τη συλλογή πληροφοριών από τις διεπαφές ασύρματου δικτύου:

py sta1.wintfs

Για τον έλεγχο συνδεσιμότητας μεταξύ των stations εκτελούμε:

mininet-wifi> sta1 ping -c1 sta2

Για τη δημιουργία ενσύρματης σύνδεσης μεταξύ ενός station και ενός access point:

from mininet.link import TCLink

..

..

net.addLink(sta1, ap1, cls=TCLink)

Για την εκτέλεση ελέγχου παλινδρόμησης (regression test) εκτελούμε:

$ sudo mn --wifi --test pingpair

Με την παραπάνω εντολή δημιουργείται μια minimal topology αποτελούμενη από έναν OpenFlow reference controller, εκτελείται ένας all-pairs-ping έλεγχος και στη συνέχεια διακόπτεται η τοπολογία και ο controller.

Ένα ακόμη χρήσιμος έλεγχος είναι το iperf:

$ sudo mn --wifi --test iperf

Με την παραπάνω εντολή δημιουργείται η ίδια Mininet-WiFi τοπολογία, εκτελείται ένας iperf server στον έναν host, εκτελείται ένας iperf client σε ένα δεύτερο host και αναλύεται το πετυχημένο εύρος ζώνης.

Για την σχεδίαση γραφήματος καλούμε την ακόλουθη συνάρτηση:

net.plotGraph()

Όταν δημιουργούμε ένα σύνδεσμο ανάμεσα σε δυο APs ταυτόχρονα δημιουργείται ένας σταθερός σύνδεσμος μεταξύ τους. Ωστόσο, εάν θέλουμε να εξατομικεύσουμε το στυλ γραμμής (line style) εκτελούμε:

net.addLink(ap1, ap2, ls='.')

Για τη δημιουργία πολλαπλών SSIDs πάνω από ένα single AP εκτελούμε:

ap1 = net.addAccessPoint('ap1', vssids='ssid1,ssid2,ssid3,ssid4',

ssid='ssid', mode='g', channel='1')

Για να θέσουμε την τοποθεσία σε ένα κόμβο εκτελούμε:

mininet-wifi> py sta1.setPosition('10,10,0') # x=10, y=10, z=0

Για να επιβεβαιώσουμε την τοποθεσία εκτελούμε:

mininet-wifi> py sta1.position

Για την αποσύνδεση μεταξύ του συνδέσμου sta1 από το ap1 εκτελούμε:

mininet-wifi> link ap1 sta1 down

Για την επανασύνδεση του συνδέσμου εκτελούμε ακολούθως:

mininet-wifi> link ap1 sta1 up

Για την εμφάνιση του xterm για το sta1 και sta2:

mininet-wifi> xterm sta1 sta2

Για τη διακοπή της λειτουργίας ενός AP εκτελούμε:

mininet-wifi> py ap1.stop_()

Για την επαναφορά της λειτουργίας του εκτελούμε:

mininet-wifi> py ap1.start_()

Για τον τερματισμό της εξομοίωσης εκτελούμε:

mininet-wifi> stop

Για την επανεκκίνηση της εξομοίωσης εκτελούμε:

mininet-wifi> start

Πριν μεταβούμε στην υλοποίηση της τοπολογίας μας θα αναφέρουμε ονομαστικά τα μοντέλα mobility που υποστηρίζει το Mininet-WiFi τα οποία κατηγοριοποιούνται σε:

- RandomWalk
- TruncatedLevyWalk
- RandomDirection
- RandomWayPoint
- GaussMarkov
- ReferencePoint
- TimeVariantCommunity

Για τη διαρρύθμιση των mobility μοντέλων συντάσσουμε:

net.setMobilityModel(time=0, model='RandomDirection' max_x=100, max_y=100, seed=20)

όπου

time: Ο χρόνος (in seconds) στον οποίο η τοπολογία mobility ξεκινά.

model: mobility model

max_x: Μέγιστη τιμή x

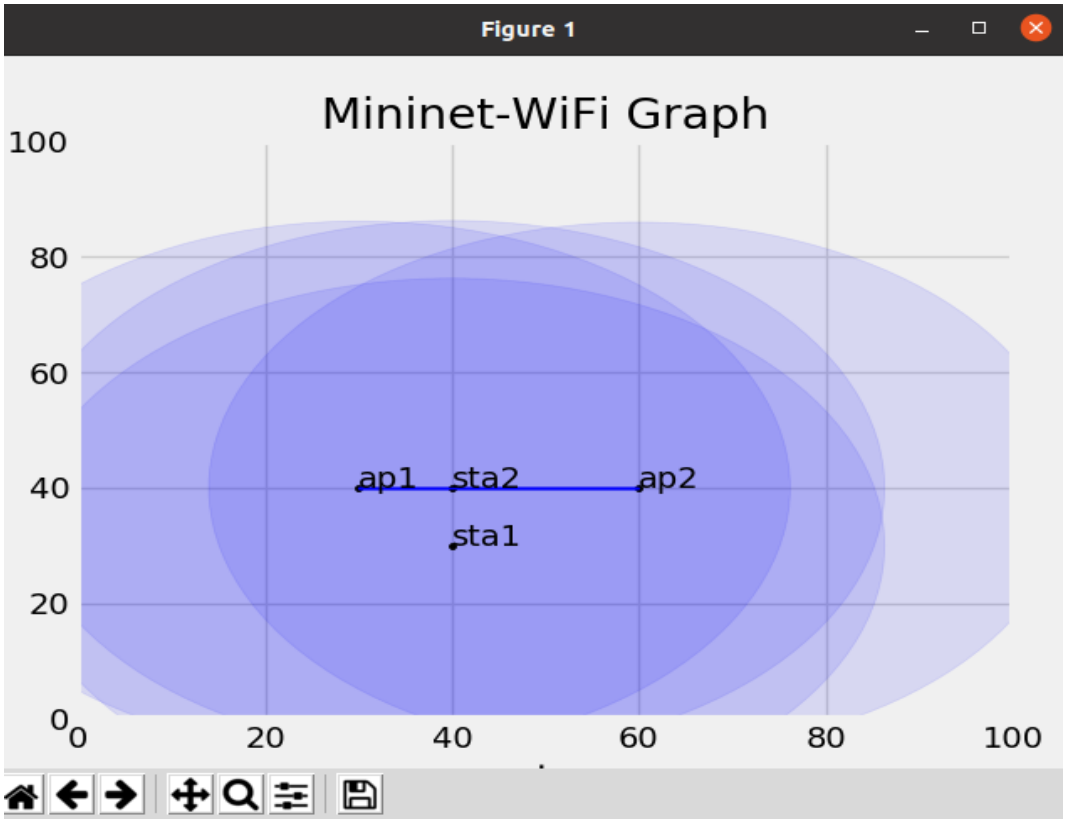
min_x: Ελάχιστη τιμή x

seed: Ορίζει μια αρχική τιμή για μια τυχαία ψευδοσειρά.

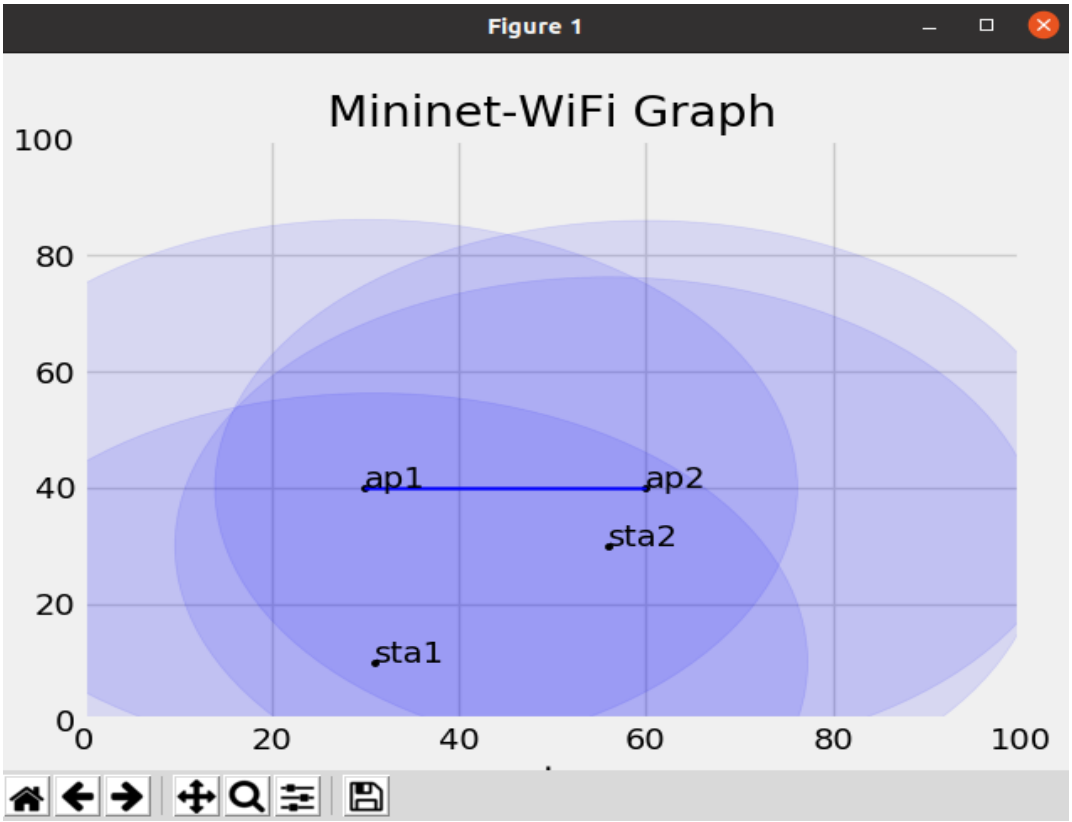
Τέλος, για την δημιουργία του σεναρίου της mobility (ΠΑΡΑΡΤΗΜΑ Β -Β2) τοπολογίας θα πρέπει να επισημάνουμε ότι το Mininet-WiFi υποστηρίζει την Python3, επομένως για την υλοποίηση της συντάσσουμε την εντολή που φαίνεται στην παρακάτω εικόνα:

```
giannis@giannis-VirtualBox:~$ cd mininet-wifi/examples
giannis@giannis-VirtualBox:~/mininet-wifi/examples$ sudo python3 mobility.py
[sudo] password for giannis:
*** Creating nodes
*** Configuring propagation model
*** Configuring wifi nodes
*** Associating and Creating links
*** Starting network
*** Running CLI
*** Starting CLI:
mininet-wifi>
```

Εικόνα 45. Χρήση εντολής sudo python3 για τη δημιουργία της mobility topology



Εικόνα 46. Σχηματική αναπαράσταση mobility τοπολογίας στο χρόνο 2 second



Εικόνα 47. Σχηματική αναπαράσταση mobility τοπολογίας στο χρόνο 12 second

Για να δούμε συνολικά τις πληροφορίες για τους κόμβους sta1 και sta2 συντάσσουμε την εντολή που αναπαρίστανται παρακάτω (Εικόνα 47).

```
mininet-wifi> py sta1.params
{'ip': '10.0.0.2/8', 'ip6': '2001:0:0:0:0:0:0:1/64', 'channel': 1, 'band': 20, 'freq': 2
.4, 'mode': 'g', 'mac': '00:00:00:00:00:02', 'wlan': ['sta1-wlan0'], 'initPos': (40.0, 3
0.0, 0.0), 'finPos': (31.0, 10.0, 0.0)}
mininet-wifi> py sta2.params
{'ip': '10.0.0.3/8', 'ip6': '2001:0:0:0:0:0:0:2/64', 'channel': 1, 'band': 20, 'freq': 2
.4, 'mode': 'g', 'mac': '00:00:00:00:00:03', 'wlan': ['sta2-wlan0'], 'initPos': (40.0, 4
0.0, 0.0), 'finPos': (55.0, 31.0, 0.0)}
mininet-wifi>
```

Εικόνα 48. Χρήση εντολής nodes.params για συλλογή πληροφοριών

Κατόπιν θα εμφανίσουμε ένα xterm window για τους εκάστοτε κόμβους sta1(10.0.0.2) και sta2(10.0.0.3) καθότι στη συνέχεια θα εκτελέσουμε ping από τον ένα κόμβο στον άλλο για να παρακολουθήσουμε την αποστολή των icmp μηνυμάτων αλλά και τη χρονική διάρκεια της.

```
"Node: sta1"
root@giannis-VirtualBox:/home/giannis/mininet-wifi/examples# ping 10.0.0.2
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=0.027 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.023 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.039 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.027 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.023 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4086ms
rtt min/avg/max/mdev = 0.023/0.027/0.039/0.005 ms
root@giannis-VirtualBox:/home/giannis/mininet-wifi/examples#
```

```
"Node: sta2"
root@giannis-VirtualBox:/home/giannis/mininet-wifi/examples# ping 10.0.0.2
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=7.17 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=5.60 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=6.18 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=5.80 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=6.25 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 5.598/6.200/7.173/0.542 ms
root@giannis-VirtualBox:/home/giannis/mininet-wifi/examples#
```

Στη συνέχεια του πειράματος σε ένα νέο τερματικό παράθυρο θα χρησιμοποιήσουμε την ακόλουθη εντολή του Open vSwitch για να μας εμφανίσει μια σύντομη σύνοψη των περιεχομένων της βάσης δεδομένων:

```
giannis@giannis-VirtualBox:~$ sudo ovs-vsctl show
[sudo] password for giannis:
0c8b1d17-320f-44f4-ab71-f9f5246a3394
    Bridge ap1
        Controller "tcp:127.0.0.1:6653"
            is_connected: true
        fail_mode: secure
        Port ap1
            Interface ap1
                type: internal
        Port ap1-eth3
            Interface ap1-eth3
        Port ap1-eth2
            Interface ap1-eth2
        Port ap1-wlan1
            Interface ap1-wlan1
    Bridge ap2
        Controller "tcp:127.0.0.1:6653"
            is_connected: true
        fail_mode: secure
        Port ap2-eth3
            Interface ap2-eth3
        Port ap2-wlan1
            Interface ap2-wlan1
        Port ap2-eth2
            Interface ap2-eth2
        Port ap2
            Interface ap2
                type: internal
    ovs_version: "2.13.3"
giannis@giannis-VirtualBox:~$
```

Κλείνοντας για να εμφανίσουμε την ύπαρξη ροών στους κόμβους ap1 και ap2 εκτελούμε:

```
giannis@giannis-VirtualBox:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows ap1
 cookie=0x0, duration=3.210s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, reset
_counts priority=65535,arp,in_port="ap1-wlan1",vlan_tci=0x0000/0x1fff,dl_src=00:00:00
:00:00:02,dl_dst=00:00:00:00:00:01,arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_op=2 actions
=output:"ap1-eth2"
 cookie=0x0, duration=3.189s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, reset
_counts priority=65535,arp,in_port="ap1-eth3",vlan_tci=0x0000/0x1fff,dl_src=00:00:00:
00:00:03,dl_dst=00:00:00:00:00:01,arp_spa=10.0.0.3,arp_tpa=10.0.0.1,arp_op=2 actions=
output:"ap1-eth2"
 cookie=0x0, duration=3.163s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, reset
_counts priority=65535,arp,in_port="ap1-eth3",vlan_tci=0x0000/0x1fff,dl_src=00:00:00:
00:00:03,dl_dst=00:00:00:00:00:02,arp_spa=10.0.0.3,arp_tpa=10.0.0.2,arp_op=2 actions=
output:"ap1-wlan1"
 cookie=0x0, duration=3.209s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, reset
_counts priority=65535,icmp,in_port="ap1-eth2",vlan_tci=0x0000/0x1fff,dl_src=00:00:00
:00:00:01,dl_dst=00:00:00:00:00:02,nw_src=10.0.0.1,nw_dst=10.0.0.2,nw_tos=0,icmp_type
=8,icmp_code=0 actions=output:"ap1-wlan1"
```

```
giannis@giannis-VirtualBox:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows ap2
 cookie=0x0, duration=27.783s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, rese
t_counts priority=65535,arp,in_port="ap2-wlan1",vlan_tci=0x0000/0x1fff,dl_src=00:00:0
0:00:00:03,dl_dst=00:00:00:00:00:01,arp_spa=10.0.0.3,arp_tpa=10.0.0.1,arp_op=2 action
s=output:"ap2-eth3"
 cookie=0x0, duration=27.756s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, rese
t_counts priority=65535,arp,in_port="ap2-wlan1",vlan_tci=0x0000/0x1fff,dl_src=00:00:0
0:00:00:03,dl_dst=00:00:00:00:00:02,arp_spa=10.0.0.3,arp_tpa=10.0.0.2,arp_op=2 action
s=output:"ap2-eth3"
 cookie=0x0, duration=22.529s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, rese
t_counts priority=65535,arp,in_port="ap2-wlan1",vlan_tci=0x0000/0x1fff,dl_src=00:00:0
0:00:00:03,dl_dst=00:00:00:00:00:02,arp_spa=10.0.0.3,arp_tpa=10.0.0.2,arp_op=1 action
s=output:"ap2-eth3"
 cookie=0x0, duration=22.529s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, rese
t_counts priority=65535,arp,in_port="ap2-wlan1",vlan_tci=0x0000/0x1fff,dl_src=00:00:0
0:00:00:03,dl_dst=00:00:00:00:00:01,arp_spa=10.0.0.3,arp_tpa=10.0.0.1,arp_op=1 action
s=output:"ap2-eth3"
```

5.4 Ανάλυση αποτελεσμάτων προσομοίωσης της λειτουργίας δικτύου SDN με ODL Controller

Για την αποτύπωση της απόδοσης των αποτελεσμάτων των παραπάνω σεναρίων θα χρησιμοποιήσουμε την ασύγχρονη τεχνολογία στατιστικής ανάλυσης **sFlow-RT** [37]. Ο μηχανισμός ανάλυσης sFlow-RT παρέχει παρακολούθηση σε πραγματικό χρόνο (real-time) σεναρίων SDN δικτύωσης, καθώς επίσης λαμβάνει μια συνεχή ροή τηλεμετρίας, μέσω του REST API, από τους sFlow Agents που είναι ενσωματωμένοι σε δικτυακές συσκευές.

Στην ανάλυση αυτή θα εξετασθούν δύο τοπολογίες (Linear,Tree), όπου για την επιτυχή ολοκλήρωση της πρέπει να είναι σε κατάσταση λειτουργίας παράλληλα ο ODL controller, ο sFlow-RT server και το Mininet VM. Για να εκκινήσουμε το εργαλείο sFlow-RT θα χρειαστεί να εκτελέσουμε την ακόλουθη εντολή:

./sflow-rt/starts.sh

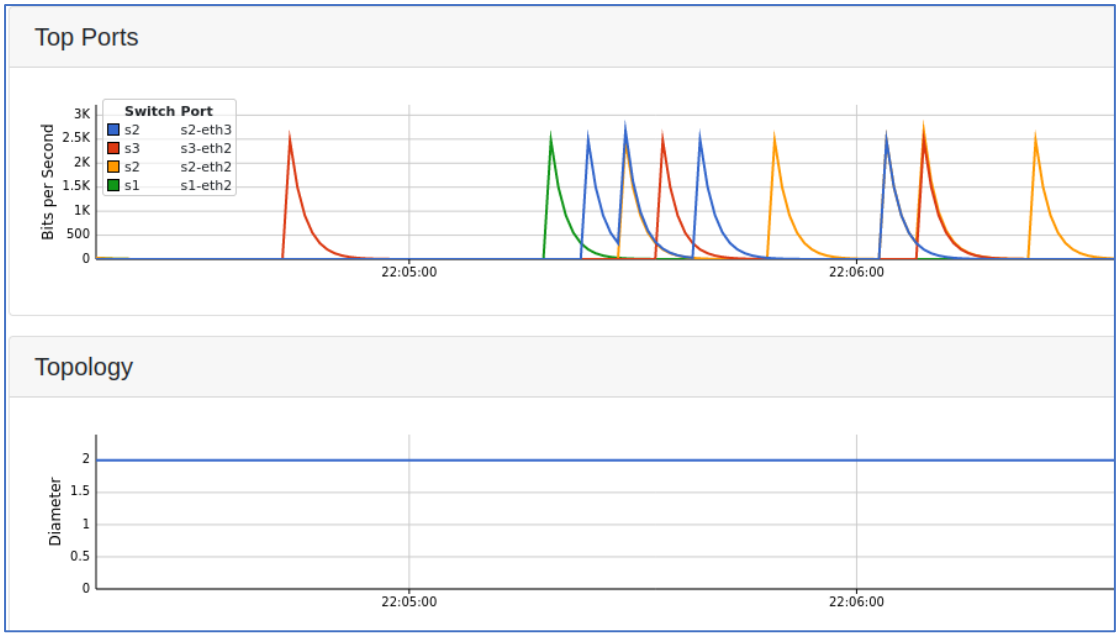
5.4.1 Ανάλυση αποτελεσμάτων Linear Network Topology

Προϋποθέτοντας ότι έχουν γίνει οι κατάλληλες διαρρυθμίσεις, για την ανάλυση αυτού του σεναρίου σε ένα νέο τερματικό παράθυρο θα πρέπει αρχικά να εκτελέσουμε την ακόλουθη εντολή για την εκκίνηση της Linear Topology:

giannis@giannis-VirtualBox:~$ ***sudo mn --custom sflow-rt/extras/sflow.py --topo linear,3 --mac --controller=remote,ip=192.168.56.101,port=6633 --switch ovs,protocols=OpenFlow13***

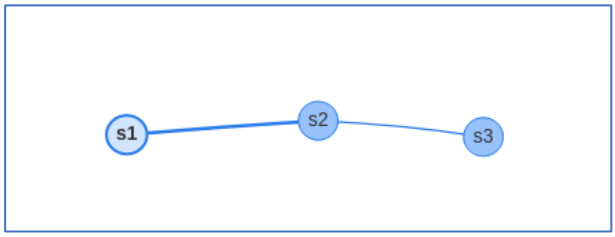
Κατόπιν ανοίγουμε ένα web browser και συνδεόμαστε με τον ακόλουθο σύνδεσμο http://192.168.56.101:8008/html/index.html#status για να αλληλοεπιδράσουμε με το REST API.

Μέσω της εγκατεστημένης εφαρμογής **mininet-dashboard** του εργαλείου sFlow-RT μπορούμε να παρατηρήσουμε σε πραγματικό χρόνο την κίνηση των ports σε Bps καθώς και την συμπεριφορά της τοπολογίας μας (Εικόνα49).



Εικόνα 49. Top Ports και Topology μέσω του mininet-dashboard

Από την παραπάνω εικόνα μπορούμε να αντιληφθούμε ότι η κίνηση ανάμεσα στα switch ports είναι αρμονική και πως η τοπολογία αναπαρίσταται γραμμική.



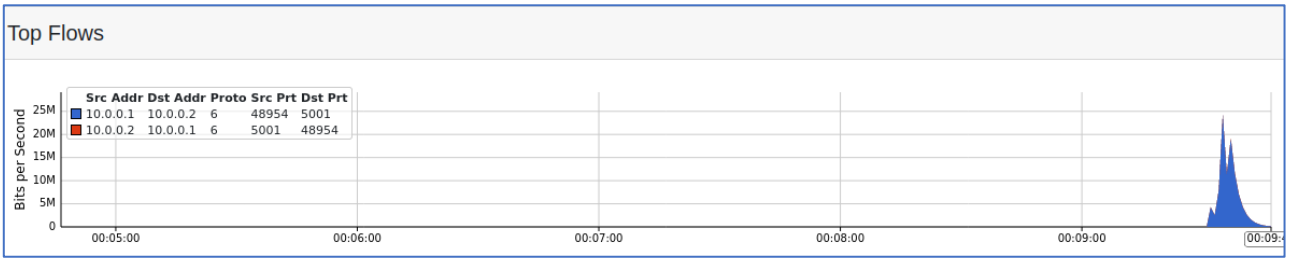
Εικόνα 50. Σχηματική αναπαράσταση τοπολογίας Linear στο mininet-dashboard

Στη συνέχεια κάνοντας χρήση του εργαλείου **iperf** θα ελέγξουμε την απόδοση και το εύρος ζώνης του δικτύου μας, όπου τα αποτελέσματα θα απεικονιστούν στα charts του mininet-dashboard.

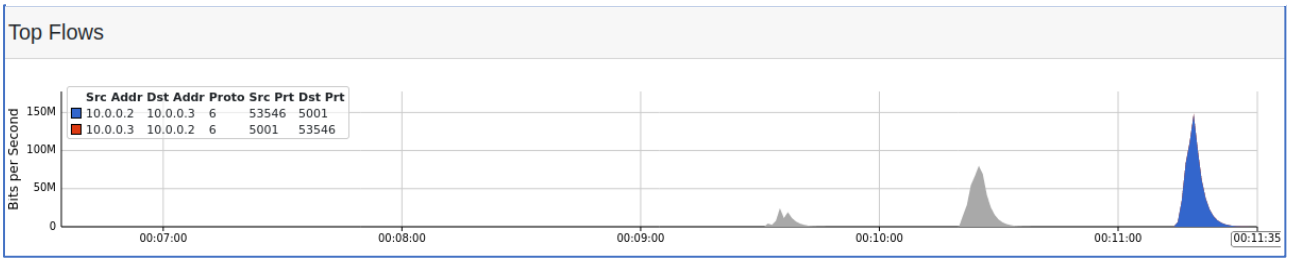
```
mininet-wifi> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['2.42 Gbits/sec', '2.43 Gbits/sec']
mininet-wifi> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['3.34 Gbits/sec', '3.34 Gbits/sec']
mininet-wifi> iperf h2 h3
*** Iperf: testing TCP bandwidth between h2 and h3
*** Results: ['3.49 Gbits/sec', '3.51 Gbits/sec']
```

Εικόνα 51. Χρήση iperf tool για τον έλεγχο του bandwidth μεταξύ των κεντρικών υπολογιστών

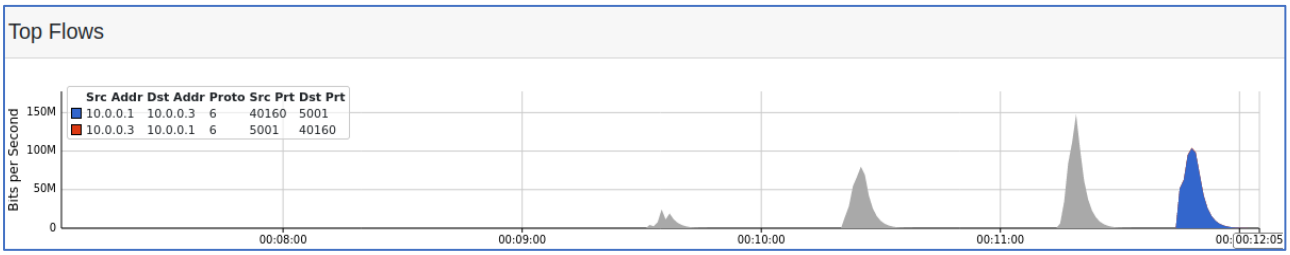
Από τις ακόλουθες εικόνες μπορούμε να παρατηρούμε την κίνηση ροής ανάμεσα στους κεντρικούς υπολογιστές σε bits per second.



Εικόνα 53. Σχηματική αναπαράσταση κίνησης ροής μεταξύ h1 και h2

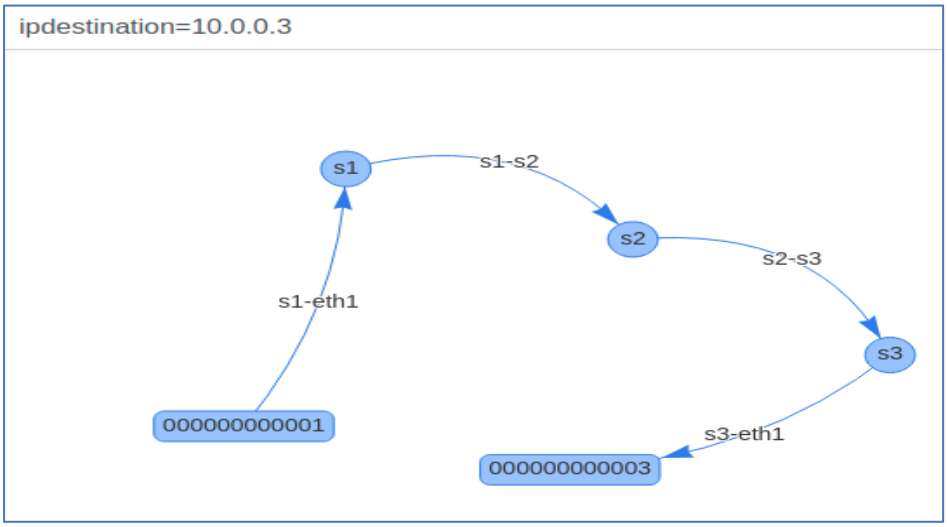


Εικόνα 52. Σχηματική αναπαράσταση κίνησης ροής μεταξύ h2 και h3



Εικόνα 54. Σχηματική αναπαράσταση κίνησης ροής μεταξύ h2 και h3

Τέλος, μια ακόμη πηγή πληροφορίας είναι η εφαρμογή Trace Flow του sFlow-RT όπου μπορούμε να έχουμε επίγνωση της κατάστασης της λειτουργίας του δικτύου μας σε πραγματικό χρόνο (Εικόνα 55).



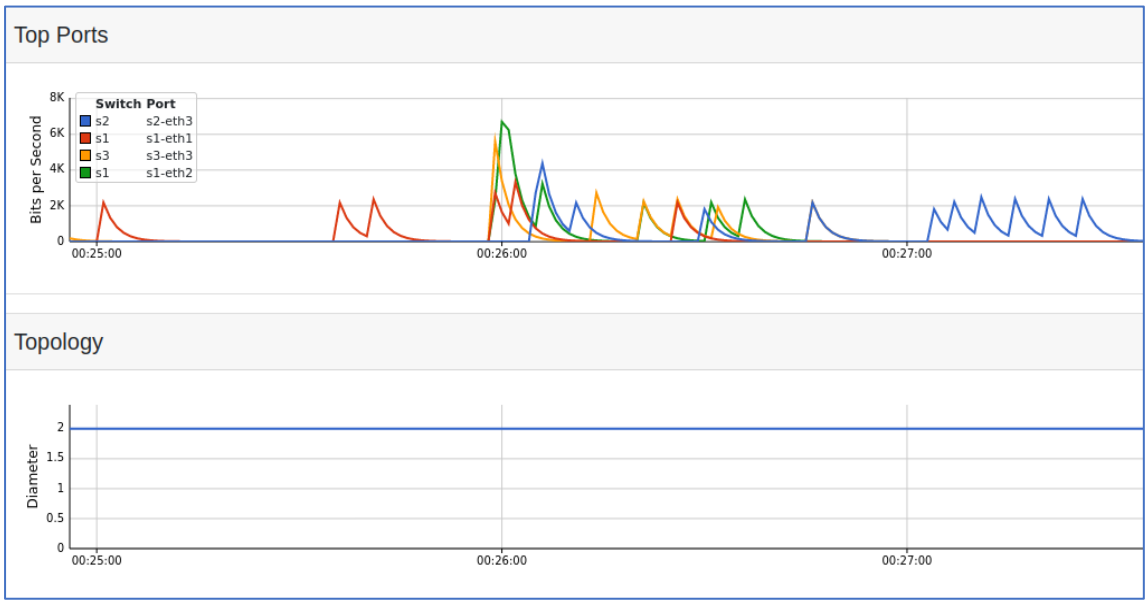
Εικόνα 55. Σχηματική αναπαράσταση διάταξης της Linear topology μέσω Trace Flow

5.4.2 Ανάλυση αποτελεσμάτων Tree Network Topology

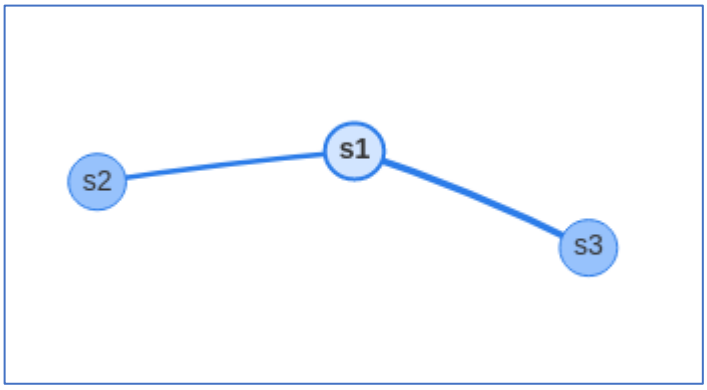
Για να αντλήσουμε αποτελέσματα από μια Tree Topology θα πρέπει για την εκκίνηση της να εκτελέσουμε:

giannis@giannis-VirtualBox:~$ ***sudo mn --custom sflow-rt/extras/sflow.py --topo tree,2,2 --mac --controller=remote,ip=192.168.56.101,port=6633 --switch ovs,protocols=OpenFlow13***

Από την υλοποίηση της παραπάνω διάταξης παράχθηκαν τα ακόλουθα διαγράμματα:



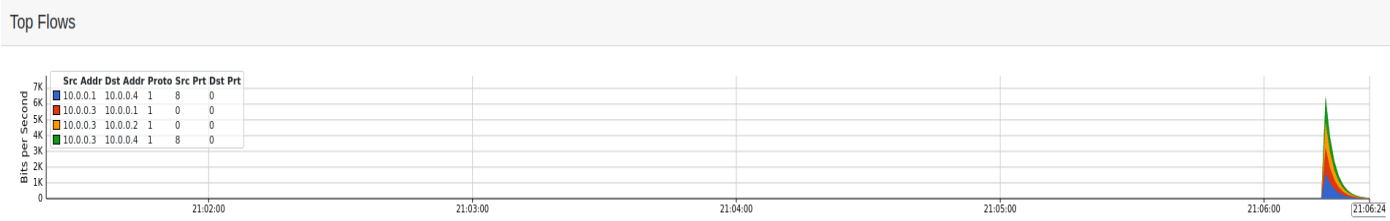
Εικόνα 56. Top Port και Topology μέσω mininet-dashboard



Εικόνα 57. Σχηματική αναπαράσταση τοπολογίας Tree στο mininet-dashboard

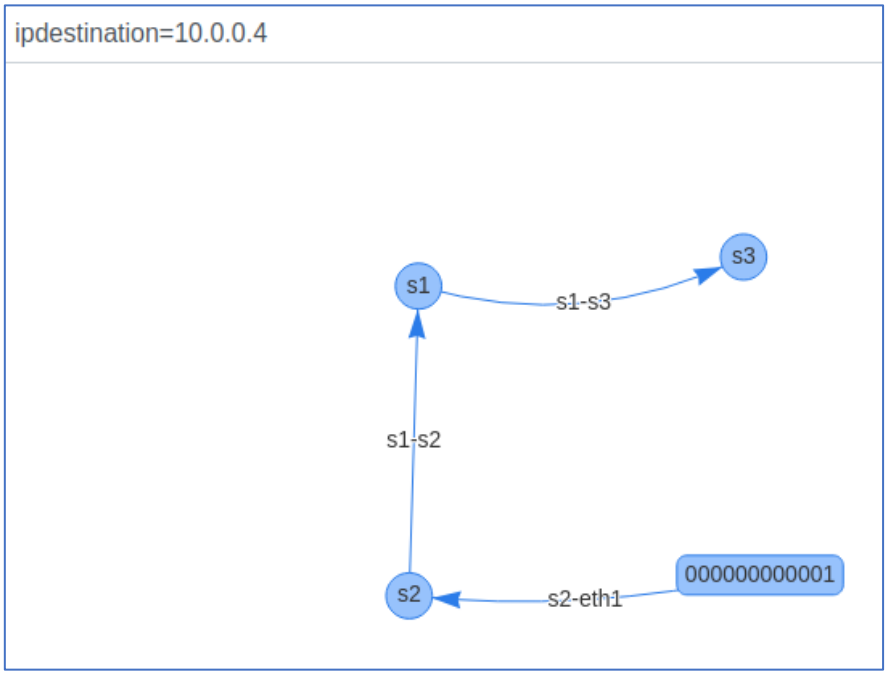
Στη συνέχεια του πειράματος αυτού θα εκτελέσουμε την εντολή **pingall** για να καταγράψουμε πληροφορίες για την κίνηση ροών.

Από την παραπάνω εκτέλεση έχουμε τα ακόλουθα αποτελέσματα:



Εικόνα 58. Σχηματική αναπαράσταση κίνησης ροής σε όλους τους κόμβους

Τέλος, όμοια με την προηγούμενη ανάλυση σεναρίου θα χρησιμοποιηθεί το εργαλείο Trace Flow για να επαληθεύσουμε την κατάσταση της λειτουργίας του δικτύου μας.



Εικόνα 59. Σχηματική αναπαράσταση διάταξης της Treer topology μέσω Trace Flow

Συμπέρασμα καταγραφής ανάλυσης σεναρίων:

Κάνοντας σύγκριση των γραφημάτων που προέκυψαν από την υλοποίηση των παραπάνω τοπολογιών Linear και Tree, μπορούμε να συμπεράνουμε πως και οι δυο διατάξεις ανταποκρίνονται με παρόμοιο τρόπο στην καταγραφή που πραγματοποιήθηκε, καθώς παρουσιάζουν αποδοτική διαχείριση και εξισορρόπηση της κίνησης των ροών δεδομένων (flows) στις θύρες (ports) των OpenFlow Switches.

Συμπεράσματα

Η αρχιτεκτονική του Software Defined Network παρέχει στους μηχανικούς δικτυών άφθονη επίβλεψη και έλεγχο των δυνατοτήτων, διαχωρίζοντας το επίπεδο ελέγχου από το επίπεδο προώθησης. Ο κεντροποιημένος έλεγχος του δικτύου και το πρωτόκολλο OpenFlow το οποίο είναι συμβατό με την πλειοψηφία των δομικών στοιχείων του SDN, κάνουν το δίκτυο προγραμματιζόμενο, άμεσα ρυθμιζόμενο και διαχειρίσιμο.

Αναμφισβήτητα, το μέλλον των δικτύων ανεξαρτητοποιείται από το υλικό και οδεύει προς το λογισμικό μετατρέποντας έτσι τα συμβατικά δίκτυα σε ευέλικτες και προγραμματιζόμενες πλατφόρμες. Με το τρόπο αυτό καλύπτονται οι απαιτήσεις των χρηστών και των επιχειρήσεων, μειώνοντας ταυτόχρονα το λειτουργικό κόστος μέσω της βελτιστοποίησης της χρήσης των πόρων, της απλοποίησης των λειτουργιών και μέσω της εικονικοποίησης για την υποστήριξη αυτοματοποιημένων και ασφαλών περιβαλλόντων νέφους. Παράλληλα μειώνονται και οι δαπάνες για αγορά εξοπλισμού αφού δεν απαιτείται η ανάγκη για αγορά δικτυακών συσκευών (δρομολογητές κτλ.) αλλά ο εφοδιασμός απλών υπολογιστών.

Πρόκειται για μια τεχνολογία η οποία φέρει την επανάσταση στο χώρο της βιομηχανίας των δικτύων και των τηλεπικοινωνιών. Με τα πολλά πλεονεκτήματα που προσφέρει το SDN, σε συνδυασμό με την ολοένα αυξανόμενη απαίτηση για ευελιξία του δικτύου και με την εξέλιξη των τεχνολογιών της πληροφορικής, αναμένεται να κατασταλεί ως το πρότυπο στο κόσμο των δικτύων.

Συμπεράσματα υλοποίησης

Στη μελέτη αυτή υλοποιήθηκαν τρία είδη τοπολογιών, Linear, Tree και Mobility. Η Linear τοπολογία προσφέρει εύκολο έλεγχο και συντήρηση, καθώς οι μηχανικοί μπορούν εύκολα να εντοπίσουν και να επιδιορθώσουν ένα πιθανό σφάλμα. Η Tree τοπολογία είναι απλή και ισορροπημένη, απαιτεί απλή διαχείριση και ρύθμιση ωστόσο μπορεί εύκολα να γίνει σύνθετη προσθέτοντας επιπλέον συσκευές δημιουργώντας έτσι μικρότερα υποδίκτυα σχηματίζοντας τοπολογίες αστέρα (Star). Τέλος, η τοπολογία Mobility είναι πιο εξειδικευμένη και σύνθετη, αφού αντί για switches έχουμε Access Points (AP) και αντί για κεντρικούς

υπολογιστές έχουμε κινούμενους σταθμούς (Stations). Ως ανάλυση αποτελεσμάτων απόδοσης τοπολογιών εξετάστηκαν η Linear και η Tree, με τη συνεργασία του εργαλείου sFLow-RT όπου είχαμε παρακολούθηση σε πραγματικό χρόνο της κατάστασης της τοπολογίας του δικτύου μας καθώς και την παροχή χρήσιμων πληροφοριών μέσω των εγκατεστημένων εφαρμογών του sFlow.

Μελλοντικές προκλήσεις

Η αυξημένη κατανάλωση των υπηρεσιών των πολυμέσων και η ανάγκη για υψηλή ποιότητα υπηρεσιών (Quality of Service) από τους χρήστες έχει δώσει έναυσμα για θεμελιώδη αλλαγή στο τρόπο που διαχειριζόμαστε το δίκτυο όσον αφορά την άντληση, το διαχωρισμό και τη σχεδίαση της προώθησης, του ελέγχου και της διαχείρισης των υπηρεσιών. Για το λόγο αυτό η βιομηχανία και η πανεπιστημιακή κοινότητα έχουν ενστερνιστεί την επόμενη γενιά κινητού δικτύου 5G, ικανή να υποστηρίξει τις απαιτήσεις των υπηρεσιών των εφαρμογών. Για να συνειδητοποιήσουμε την εικόνα του 5G δικτύου, το φυσικό δίκτυο πρέπει να τεμαχιστεί σε πολλαπλά απομονωμένα λογικά δίκτυα μεταβαλλόμενου μεγέθους και δομής, τα οποία είναι ειδικά για διαφορετικούς τύπους υπηρεσιών που βασίζονται σε απαιτήσεις με διαφορετικά χαρακτηριστικά (IoT συσκευές, smartphones, αυτόνομα αυτοκίνητα κ.α.). Οι τεχνολογίες SDN και NFV στα 5G δίκτυα αναμένεται να καλύψουν το κενό του προγραμματιζόμενου ελέγχου και διαχείρισης των πόρων του δικτύου.

5G AND SDN

Η κεντρική ιδέα πίσω από το SDN είναι ο διαχωρισμός του επιπέδου ελέγχου από το δίκτυο υλικού και η ενεργοποίηση εξωτερικού ελέγχου των δεδομένων μέσω μιας λογικής οντότητας λογισμικού τον controller. Ο controller, ο οποίος διαχειρίζεται τον έλεγχο ροής των πακέτων για την δημιουργία δικτυακής νοημοσύνης, είναι τοποθετημένος στις δικτυακές συσκευές και τις εφαρμογές. Στην αρχιτεκτονική αυτή ο έλεγχος του δικτύου γίνεται προγραμματίσιμος. Με τον controller, οι διαχειριστές μπορούν εύκολα να διαχειριστούν το δίκτυο 5G και να παράγουν νέες υπηρεσίες ή αλλαγές. Ένα ευφυές, εικονικοποιημένο και

προγραμματίσιμο δίκτυο 5G επιτρέπει στις εταιρίες να καινοτομήσουν, τόσο στη παροχή λειτουργιών όσο και στη παροχή υπηρεσιών. Έτσι οι εταιρίες θα μπορούν να προσφέρουν νέες υπηρεσίες κατά απαίτηση, βελτιώνοντας την συνολική αποτελεσματικότητα. Η προγραμματιστικότητα του 5G SDN δικτύου επιτρέπει νέα μοντέλα επιχειρήσεων και οδηγεί στην αύξηση εισοδήματος. Η αρχιτεκτονική του δικτύου 5G SDN είναι δυναμική, υψηλά διαχειρίσιμη και οικονομικά αποτελεσματική, κάνοντας την ιδανική για τη δυναμική και υψηλού εύρους ζώνης φύση των περιπτώσεων 5G.

5G AND NFV

Η τεχνολογία NFV είναι το κλειδί δημιουργίας της εισερχόμενης 5G υποδομής βοηθώντας να εικονικοποιεί τις διάφορες συσκευές του δικτύου. Στο 5G, η τεχνολογία NFV επιτρέπει τον τεμαχισμό του δικτύου, μια εικονική αρχιτεκτονική δικτύου που θα επιτρέπει πολλαπλά εικονικά δίκτυα να δημιουργηθούν πάνω σε μια κοινόχρηστη φυσική υποδομή. Τα εικονικά δίκτυα μπορούν να προσαρμοστούν για να καλύψουν τις ανάγκες των εφαρμογών, των υπηρεσιών, των συσκευών, των χρηστών και των χειριστών. Στο 5G, το NFV θα μπορεί επίσης να επιτρέπει το διαμοιρασμένο νέφος, βοηθώντας στη δημιουργία ευέλικτων και προγραμματίσιμων δικτύων για τις ανάγκες του αύριο.

Βιβλιογραφία

[1] O. N. Foundation, "Open Network Foundation," 2011. [Online]. Available: https://opennetworking.org/sdn-definition/.

[2] S. Studios, "SDxCentral," 13 Μάρτιος 2015. [Online]. Available: https://www.sdxcentral.com/networking/sdn/definitions/inside-sdn-architecture/.

[3] T. Thakur, "HowtoForge," [Online]. Available: https://www.howtoforge.com/tutorial/software-defined-networking-sdn-architecture-and-role-of-openflow/.

[4] "HowtoForge," [Online]. Available: https://www.howtoforge.com/tutorial/software-defined-networking-sdn-architecture-and-role-of-openflow/.

[5] M. Lessing, "SDxCentral Studios," Οκτώμβριος 2019. [Online]. Available: https://www.sdxcentral.com/networking/sdn/definitions/north-bound-interfaces-api/.

[6] M. Lessing, "SDxCentral Studios," Σεπτέμβριος 2019. [Online]. Available: https://www.sdxcentral.com/networking/sdn/definitions/southbound-interface-api/.

[7] "SDxCentral," [Online]. Available: https://www.sdxcentral.com/directory/nec/trema/.

[8] "Nox (platform)," Μάϊος 2015. [Online]. Available: https://en.wikipedia.org/wiki/Nox_(platform).

[9] S. Rao, "The New Stack," 6 Ιανουάριος 2015. [Online]. Available: https://thenewstack.io/sdn-series-part-v-floodlight/.

[10] [Online]. Available: https://en.wikipedia.org/wiki/ONOS.

[11] "SDxCentral Studios," 23 Ιουνίου 2016. [Online]. Available: https://www.sdxcentral.com/networking/sdn/definitions/what-is-ryu-controller/.

[12] [Online]. Available: https://ryu-sdn.org/.

[13] C. Francis, "QuickStart," 19 Φεβρουάριος 2019. [Online]. Available: https://www.quickstart.com/blog/8-advantages-of-software-defined-marketing/.

[14] "ibm," [Online]. Available: https://www.ibm.com/services/network/sdn-versus-traditional-networking.

[15] "VMware," [Online]. Available: https://www.vmware.com/topics/glossary/content/network-functions-virtualization-nfv.

[16] "SDxCentral Studios," 26 Αυγούστου 2013. [Online]. Available: https://www.sdxcentral.com/networking/nfv/definitions/whats-network-functions-virtualization-nfv/.

[17] "redhat," [Online]. Available: https://www.redhat.com/en/topics/virtualization/what-is-nfv.

[18] Ε. Αλμετίδου. [Online]. Available: https://sites.google.com/site/cloudintime/.

[19] J. FRANKENFIELD, "Investopedia," 28 Ιούλιος 2020. [Online]. Available:

https://www.investopedia.com/terms/c/cloud-computing.asp.

[20] "azure.microsoft.com," [Online]. Available: https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/#cloud-computing-models.

[21] C. Craven, "SDxCentral Studios," Νοέμβριος 2020. [Online]. Available: https://www.sdxcentral.com/networking/sdn/definitions/what-is-openflow/.

[22] "The Open Networking Foundation," 26 Μαρτίου 2015. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

[23] [Online]. Available: https://docs.opendaylight.org/en/stable-silicon/user-guide/opendaylight-controller-overview.html.

[24] Baeldung, "Baeldung," 14 Αύγουστος 2019. [Online]. Available: https://www.baeldung.com/osgi.

[25] S. Patil, "InfoWorld," 4 Μάρτιος 2008. [Online]. Available: https://www.infoworld.com/article/2077837/java-se-hello-osgi-part-1-bundles-for-beginners.html.

[26] A. L. S. D. Lusani Mamushiane, 3-5 Μάιος 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8361694. [Accessed 21 Μάιος 2018].

[27] "LFNETWORKING," 31 Μαρτίου 2020. [Online]. Available: https://www.lfnetworking.org/blog/2020/03/31/odl-magnesium-new-projects-new-service-provider-features-and-loads-of-performance-improvements/.

[28] [Online]. Available: https://github.com/opendaylight/dlux.

[29] "OpenDaylight," 2016-2018. [Online]. Available: https://docs.opendaylight.org/en/stable-nitrogen/getting-started-guide/common-features/dlux.html.

[30] "OpenDaylight," [Online]. Available: https://docs.opendaylight.org/en/stable-fluorine/user-guide/l2switch-user-guide.html.

[31] "OpenDaylight Documentation," [Online]. Available: https://docs.opendaylight.org/en/stable-neon/user-guide/openflow-plugin-project-user-guide.html.

[32] [Online]. Available: https://el.tipsandtrics.com/what-is-virtual-machine-767992.

[33] "brianlinkletter," 28 Φεβρουάριος 2016. [Online]. Available: https://www.brianlinkletter.com/2016/02/using-the-opendaylight-sdn-controller-with-the-mininet-network-emulator/.

[34] "Mininet," [Online]. Available: http://mininet.org/overview/.

[35] "OpenvSwitch," [Online]. Available: https://www.openvswitch.org/.

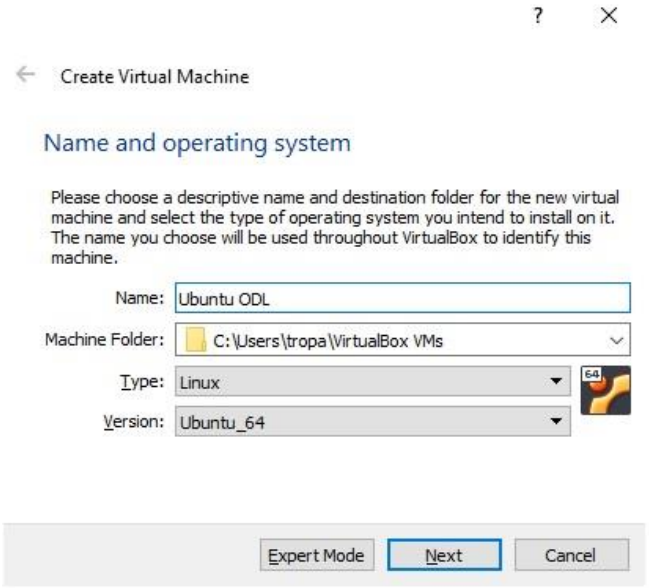
[36] K.-V. -. R. S. -. T-Subashri, "OpenSourceForU," 7 Απρίλιος 2017. [Online]. Available: https://www.opensourceforu.com/2017/04/beginners-guide-mininet/.

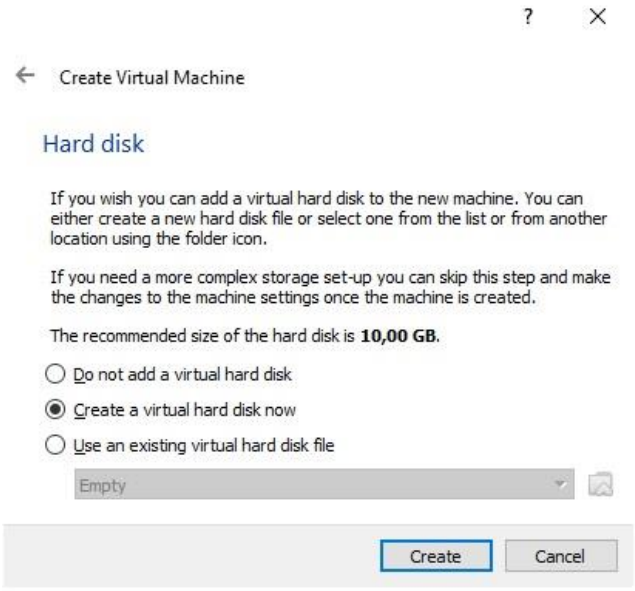
[37] "inMon," [Online]. Available: https://inmon.com/products/sFlow-RT.php.

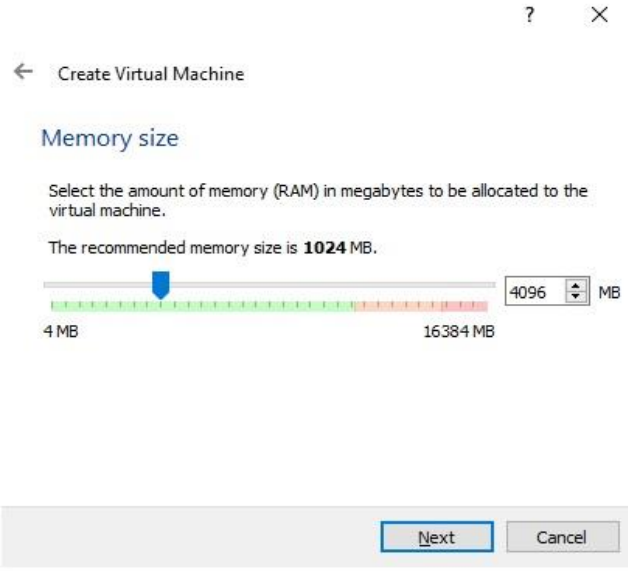
ΠΑΡΑΡΤΗΜΑ - Α

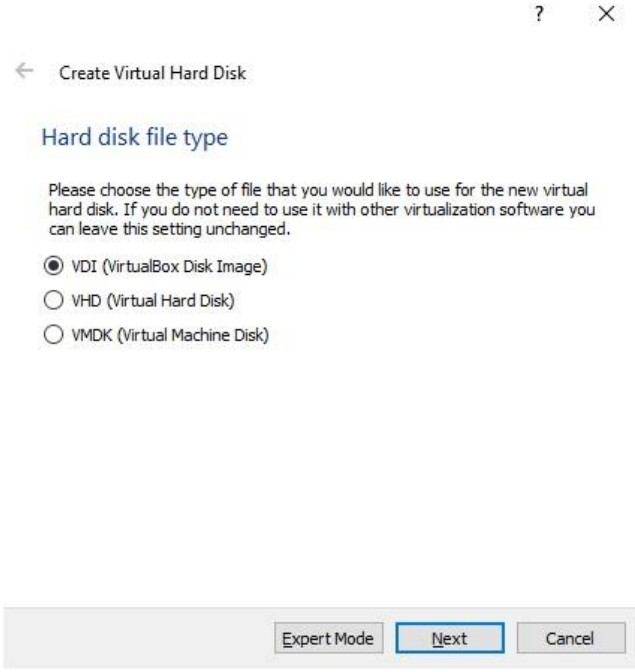
Α1. Προετοιμασία εικονικής μηχανής (Virtual Machine) για εγκατάσταση ODL Controller

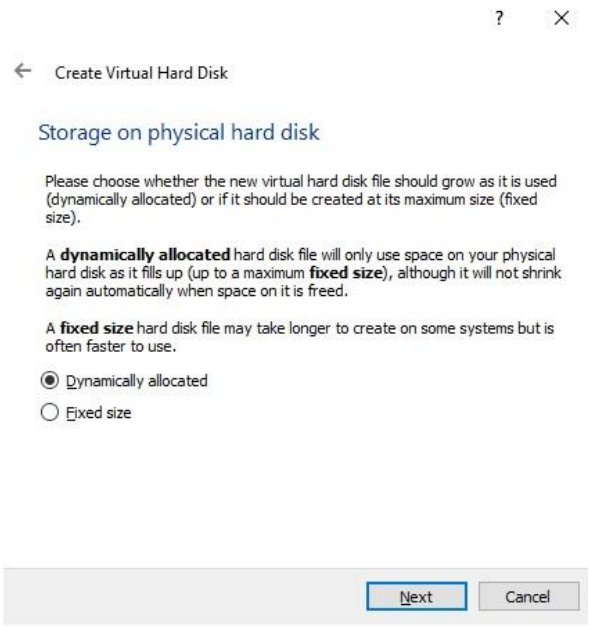
Οι ακόλουθες εικόνες αναπαριστούν βήμα προς βήμα την ανάπτυξη του OpenDaylight σε μια εικονική μηχανή.

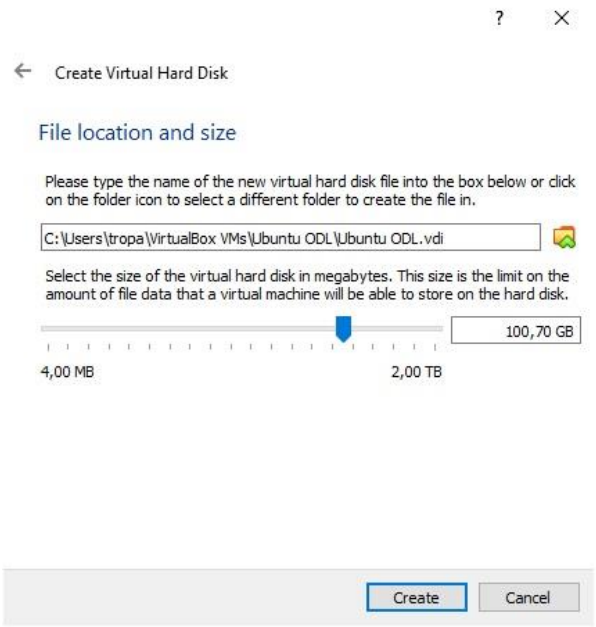


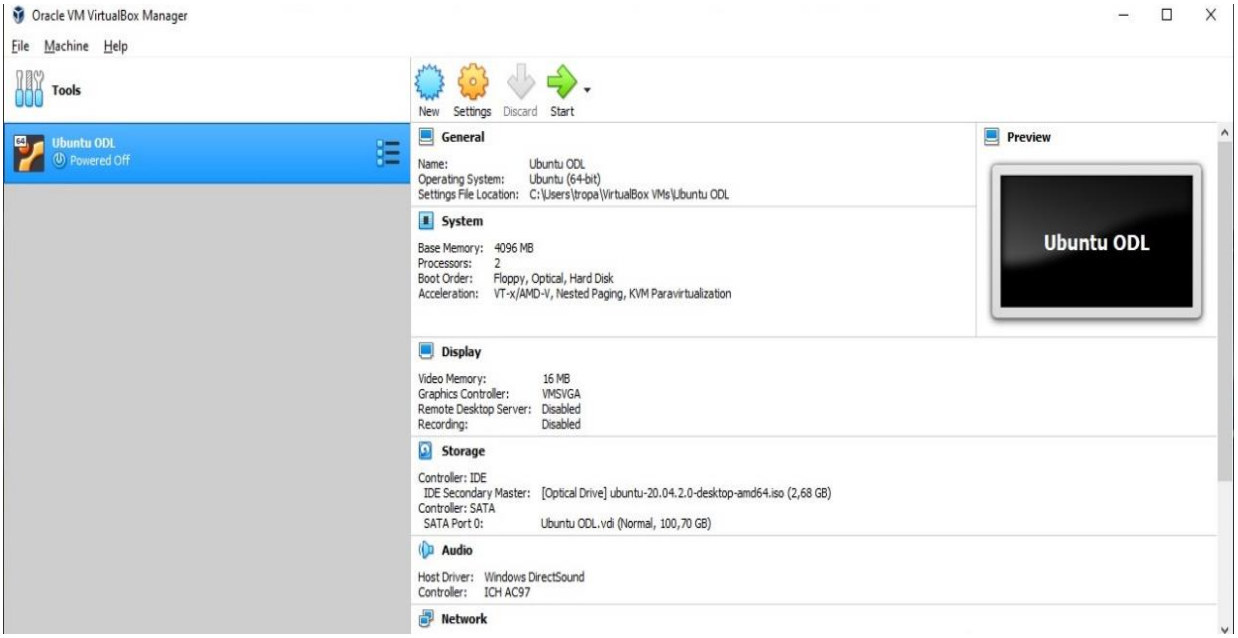


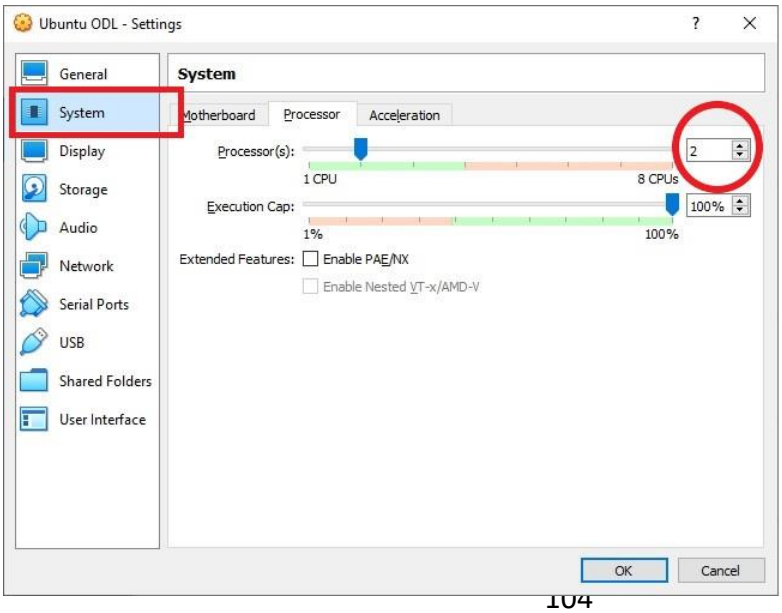


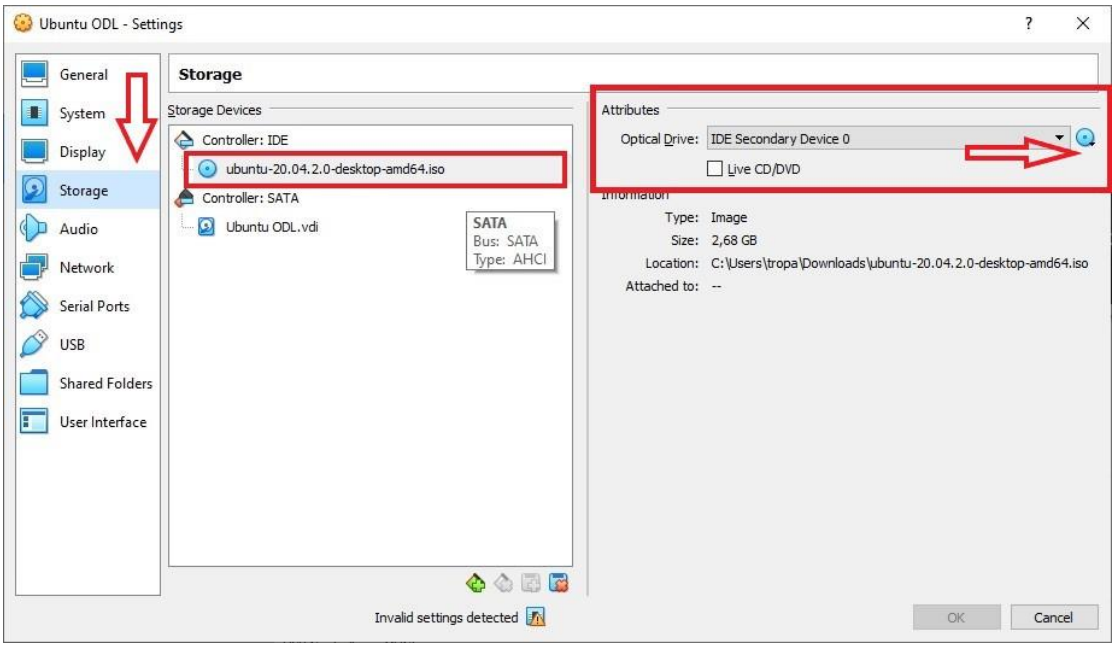


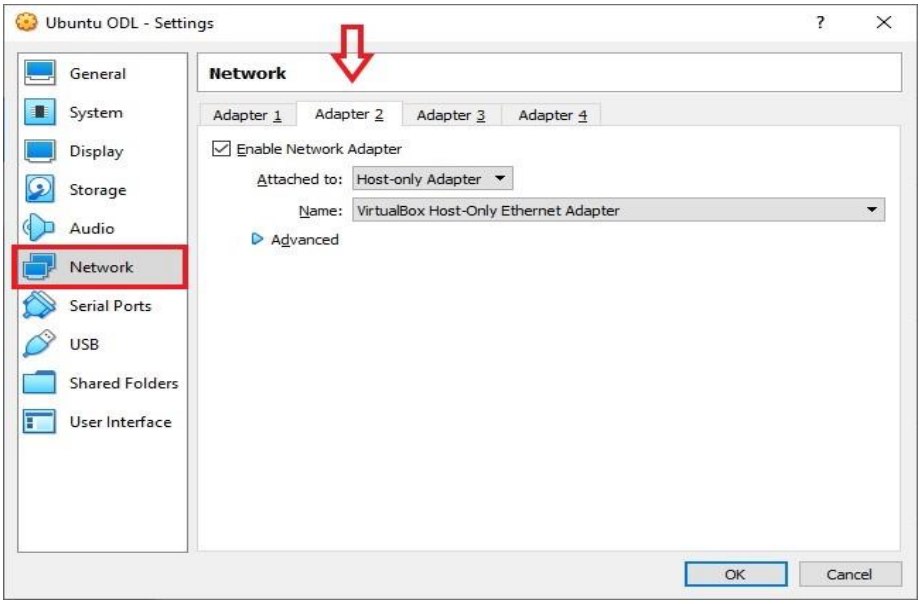

Create Virtual Hard Disk
Storage on physical hard disk
Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).
A dynamically allocated hard disk file will only use space on your physical hard disk as it fills up (up to a maximum fixed size), although it will not shrink again automatically when space on it is freed.
A fixed size hard disk file may take longer to create on some systems but is often faster to use.
Dynamically allocated
Fixed size
Next
Cancel

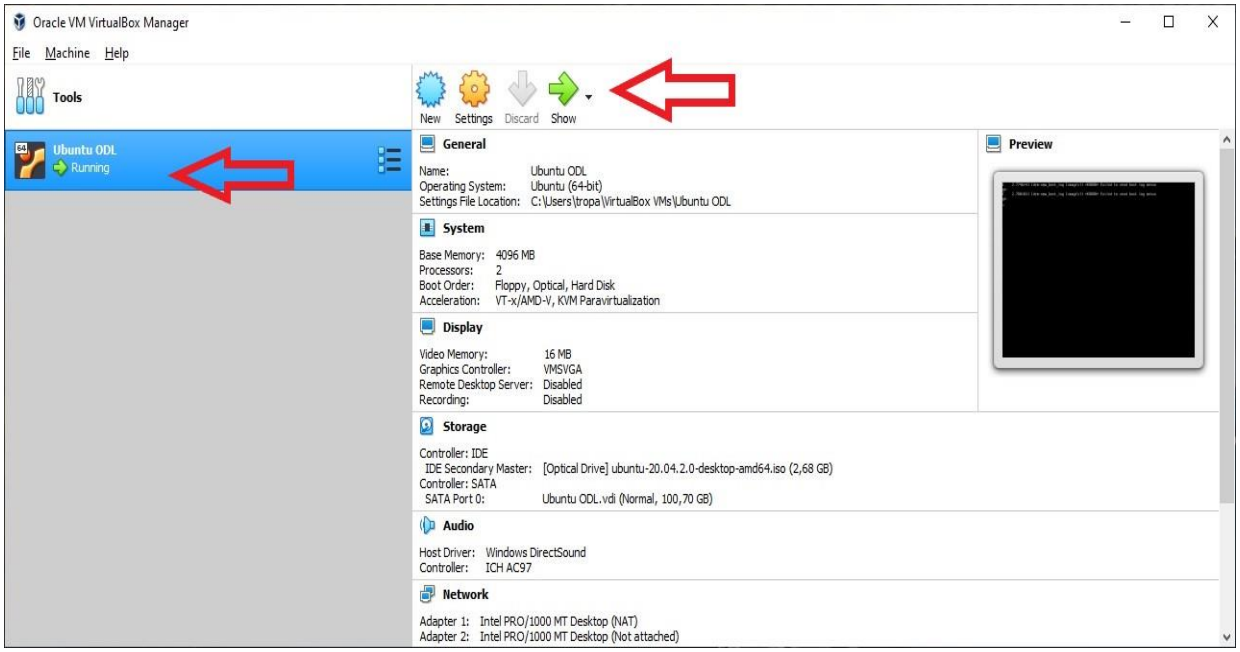

Create Virtual Hard Disk
File location and size
Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.
C:\Users\tropa\VirtualBox VMs\Ubuntu ODL\Ubuntu ODL.vdi
Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.
100,70 GB
4,00 MB
2,00 TB
Create
Cancel


Oracle VM VirtualBox Manager
File Machine Help
Tools
Ubuntu ODL
Powered Off
New Settings Discard Start
General
Name: Ubuntu ODL
Operating System: Ubuntu (64-bit)
Settings File Location: C:\Users\tropa\VirtualBox VMs\Ubuntu ODL
System
Base Memory: 4096 MB
Processors: 2
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization
Preview
Ubuntu ODL
Display
Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled
Storage
Controller: IDE
IDE Secondary Master: [Optical Drive] ubuntu-20.04.2.0-desktop-amd64.iso (2,68 GB)
Controller: SATA
SATA Port 0: Ubuntu ODL.vdi (Normal, 100,70 GB)
Audio
Host Driver: Windows DirectSound
Controller: ICH AC97
Network


Ubuntu ODL - Settings
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
User Interface
System
Motherboard
Processor
Acceleration
Processor(s):
1 CPU
8 CPUs
2
Execution Cap:
1%
100%
100%
Extended Features:
Enable PAE/NX
Enable Nested VT-x/AMD-V
OK
Cancel


Ubuntu ODL - Settings
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
User Interface
Storage
Storage Devices
Controller: IDE
ubuntu-20.04.2.0-desktop-amd64.iso
Controller: SATA
Ubuntu ODL.vdi
SATA
Bus: SATA
Type: AHCI
Attributes
Optical Drive: IDE Secondary Device 0
Live CD/DVD
Information
Type: Image
Size: 2,68 GB
Location: C:\Users\tropa\Downloads\ubuntu-20.04.2.0-desktop-amd64.iso
Attached to: --
Invalid settings detected
OK
Cancel


Ubuntu ODL - Settings
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
User Interface
Network
Adapter 1
Adapter 2
Adapter 3
Adapter 4
Enable Network Adapter
Attached to: Host-only Adapter
Name: VirtualBox Host-Only Ethernet Adapter
Advanced
OK
Cancel


Oracle VM VirtualBox Manager
File Machine Help
Tools
Ubuntu ODL
Running
New Settings Discard Show
General
Name: Ubuntu ODL
Operating System: Ubuntu (64-bit)
Settings File Location: C:\Users\tropa\VirtualBox VMs\Ubuntu ODL
System
Base Memory: 4096 MB
Processors: 2
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, KVM Paravirtualization
Display
Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled
Storage
Controller: IDE
IDE Secondary Master: [Optical Drive] ubuntu-20.04.2.0-desktop-amd64.iso (2,68 GB)
Controller: SATA
SATA Port 0: Ubuntu ODL.vdi (Normal, 100,70 GB)
Audio
Host Driver: Windows DirectSound
Controller: ICH AC97
Network
Adapter 1: Intel PRO/1000 MT Desktop (NAT)
Adapter 2: Intel PRO/1000 MT Desktop (Not attached)
Preview

ΠΑΡΑΡΤΗΜΑ - Β

Β1. Παραγωγή Python κώδικα της Tree Network Topology μέσω Miniedit

```
#!/usr/bin/env python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

def myNetwork():

  net = Mininet( topo=None,
             listenPort=6633,
             build=False,
             ipBase='10.0.0.0/8')

   info( '*** Adding controller\n' )
   c0=net.addController(name='c0',
   controller=RemoteController,
   ip='192.168.56.101',
   protocol='tcp',
   port=6633)

   info( '*** Add switches\n')
   s3 = net.addSwitch('s3', cls=OVSKernelSwitch)
   s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
   s1 = net.addSwitch('s1', cls=OVSKernelSwitch)

   info( '*** Add hosts\n')
   h1 = net.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
   h2 = net.addHost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)
   h3 = net.addHost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
   h4 = net.addHost('h4', cls=Host, ip='10.0.0.4', defaultRoute=None)

   info( '*** Add links\n')
   net.addLink(s1, h1)
```

```
    net.addLink(s1, h2)
    net.addLink(s1, s2)
    net.addLink(s2, s3)
    net.addLink(s3, h3)
    net.addLink(s3, h4)

    info( '*** Starting network\n')
    net.build()
    info( '*** Starting controllers\n')
    for controller in net.controllers:
    controller.start()

    info( '*** Starting switches\n')
    net.get('s3').start([c0])
    net.get('s2').start([c0])
    net.get('s1').start([c0])

    info( '*** Post configure switches and hosts\n')

    CLI(net)
    net.stop()

if __name__ == '__main__':
   setLogLevel( 'info' )
   myNetwork()
```

B2. Python script της Mobility Topology

```
#!/usr/bin/python

'Setting the position of nodes and providing mobility'

import sys

from mininet.log import setLogLevel, info
from mn_wifi.cli import CLI
from mn_wifi.net import Mininet_wifi

def topology(args):
   "Create a network."
   net = Mininet_wifi()
   info("*** Creating nodes\n")
```

```
h1 = net.addHost('h1', mac='00:00:00:00:00:01', ip='10.0.0.1/8')
sta1 = net.addStation('sta1', mac='00:00:00:00:00:02', ip='10.0.0.2/8')
sta2 = net.addStation('sta2', mac='00:00:00:00:00:03', ip='10.0.0.3/8')
ap1 = net.addAccessPoint('ap1', ssid='new-ssid', mode='g', channel='1',
                         position='30,40,0')
ap2 = net.addAccessPoint('ap2', ssid='new-ssid', mode='g', channel='5',
                         position='60,40,0') // Προσθήκη δεύτερου Access Point
c1 = net.addController('c1')

info("*** Configuring propagation model\n")
net.setPropagationModel(model="logDistance", exp=4.5)

info("*** Configuring wifi nodes\n")
net.configureWifiNodes()

info("*** Associating and Creating links\n")
net.addLink(ap1, h1)
net.addLink(ap2, h1) // Δημιουργία συνδέσμων
net.addLink(ap1, ap2) //
net.addLink(ap1, sta1) //
net.addLink(ap2, sta2) //

if '-p' not in args:
    net.plotGraph(max_x=100, max_y=100) // Ορίζουμε μέγιστη τιμή των x, y=100

if '-c' in args:
    sta1.coord = ['40.0,30.0,0.0', '31.0,10.0,0.0', '31.0,30.0,0.0']
    sta2.coord = ['40.0,40.0,0.0', '55.0,31.0,0.0', '55.0,81.0,0.0']

net.startMobility(time=0, mob_rep=1, reverse=False)

p1, p2, p3, p4 = dict(), dict(), dict(), dict()
if '-c' not in args:
    p1 = {'position': '40.0,30.0,0.0'}
    p2 = {'position': '40.0,40.0,0.0'}
    p3 = {'position': '31.0,10.0,0.0'}
    p4 = {'position': '55.0,31.0,0.0'}

net.mobility(sta1, 'start', time=2, **p1) //Εκκίνηση sta1 σε χρόνο 2 second
net.mobility(sta2, 'start', time=2, **p2) //Εκκίνηση sta2 σε χρόνο 2second
net.mobility(sta1, 'stop', time=12, **p3) //Τερματισμός sta1 σε χρόνο 12
```

```
    net.mobility(sta2, 'stop', time=22, **p4) //Τερματισμός sta2 σε χρόνο 22 second
    net.stopMobility(time=23)

    info("*** Starting network\n")
    net.build()
    c1.start()
    ap1.start([c1])
    ap2.start([c1])

    info("*** Running CLI\n")
    CLI(net)

    info("*** Stopping network\n")
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    topology(sys.argv)
```