Πτυχιακή Εργασία

Στο πλαίσιο του Προγράμματος Μεταπτυχιακών Σπουδών

«Τεχνολογίες και Συστήματα Ευρυζωνικών Εφαρμογών και Υπηρεσιών»

«Software Defined Networking»

Ιωάννης Χριστοδουλόπουλος
Α.Μ.: Μ9

Επιβλέποντες:

Καθ. Βασίλης Τριανταφύλλου

Επίκ. Καθ. Μιχάλης Παρασκευάς

Αντίρριο, 2016

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Αντίρριο, Ημερομηνία

14/06/2016

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Τριανταφύλλου Βασίλειος

2. Παρασκευάς Μιχαήλ

3. Ασαρίδης Ηλίας

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε κατά το ακαδημαϊκό έτος 2015-2016, υπό την επίβλεψη των καθηγητών Βασίλη Τριανταφύλλου και Μιχάλη Παρασκευά, του Τμήματος Μηχανικών Πληροφορικής του ΤΕΙ Δυτικής Ελλάδας, στους οποίους οφείλω ιδιαίτερες ευχαριστίες για την εμπιστοσύνη που μου έδειξαν αναθέτοντάς μου την παρούσα εργασία, αλλά και τη δυνατότητα που μου έδωσαν να ασχοληθώ με την συγκεκριμένη θεματολογία. Θα ήθελα να ευχαριστήσω, επίσης, την οικογένειά μου και τους φίλους μου για την στήριξή τους, καθ' όλη την διάρκεια των σπουδών μου στο Πρόγραμμα Μεταπτυχιακών Σπουδών.

Πίνακας περιεχομένων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Συντμήσεις Όρων

ACL	Access Control List
API	Application Programming Interface
BNC	Big Network Controller
DCNs	Data-Center Networks
DoS	Denial of Service
ESG	Enterprise Strategy Group
FAWG	Forwarding Abstractions Working Group
FIB	Forwarding Information Base
GRE	Generic Routing Encapsulation
ICMP	Internet Control Message Protocol
KVM	Kernel-based Virtual Machine
LLDP	Link Layer Discovery Protocol
MAC	Media Access Control
NaaS	Network as a Service
NAT	Network Address Translation
NOS	Network Operating System
OAM	network Operations, Administration, Management
ONF	Open Networking Foundation
OVS	OpenVSwitch
POF	Protocol Oblivious Forwarding
PXE	Preboot Execution Environment
QoS	Quality of Service
SDN	Software Defined Networking
SiBF	Switching with in-packet Bloom Filters
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLV	Type/Length/Value
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VM	Virtual Machine
Π.Σ.Δ.	Πανελλήνιο Σχολικό Δίκτυο

Πρόλογος

Το Software Defined Networking αποτελεί μία διαφορετική προσέγγιση του τρόπου λειτουργίας των δικτύων. Με τον γνωστό κλασσικό τρόπο τα δίκτυα ενσωματώνουν τόσο το φυσικό επίπεδο (σηματοδοσία) όσο και τη λογική για τον τρόπο ανταλλαγής πακέτων. Αυτό αποσκοπεί στη μεγιστοποίηση της ταχύτητας αποστολής των δεδομένων, μιας και προκαθορισμένες διαδρομές δεν επέβαλαν καθυστερήσεις.

Καθώς η πολυπλοκότητα των δικτύων αυξάνεται και ο εξοπλισμός γίνεται ολοένα και ακριβότερος, οι περιπτώσεις υποχρησιμοποίησης (underutilization) των δικτυακών πόρων ενίσχυσαν την αναζήτηση για έναν διαφορετικό τρόπο χρήσης του φυσικού επιπέδου επικοινωνίας. Έναν τρόπο που θα επέτρεπε την από κοινού χρήση του υλικού από διαφορετικές ομάδες χρηστών και την ευέλικτη παραχώρηση επιπλέον πόρων σε εκείνες στις ομάδες με αυξημένες ανάγκες ανταλλαγής δεδομένων.

Η επίλυση του παραπάνω προβλήματος θα ερχόταν μόνο με το διαχωρισμό του επιπέδου λήψης αποφάσεων σε σχέση με το φυσικό επίπεδο σηματοδοσίας. Επίσης, η ευελιξία ισοδυναμεί με την εύκολη αλλαγή της συμπεριφοράς του εξοπλισμού, μιας συμπεριφοράς που βασίζεται σε μετρικές χρήσης των καναλιών επικοινωνίας ώστε να αποφασίζει τον απαιτούμενο ρυθμό διακίνησης δεδομένων. Η προσπάθεια αυτή χαρακτηρίζεται σήμερα με τον όρο **Software Defined Networking**.

Παρόμοιες λύσεις έχουν δοθεί και για άλλους πόρους από το πεδίο της Πληροφορικής. Για παράδειγμα, επεξεργαστική ισχύς, μαγνητικά αποθηκευτικά μέσα καθώς και μνήμες τυχαίας προσπέλασης, αποτελούν στοιχεία τα οποία μπορούν να διαχειριστούν ανάλογα με τις ανάγκες των χρηστών. Κάτι τέτοιο αποτέλεσε τη βασική ιδέα από την εποχή των mainframes μέχρι και το Virtualization και το Cloud Computing.

Ακολουθώντας λοιπόν τα παραδείγματα αυτά, το software defined networking ορίζει ροές δεδομένων ανάλογα με τις υπάρχουσες ανάγκες και τον διαθέσιμο εξοπλισμό επί του οποίου λειτουργεί. Πρόκειται για ένα αφαιρετικό μηχανισμό που επιτρέπει τον χειρισμό των καναλιών επικοινωνίας από το λογισμικό και όχι από το ίδιο το υλικό.

Abstract

The Software Defined Networking is a different approach to the way network equipment operation. Until recently, network devices embody both the physical layer (signaling) and the logic on how to exchange packets. This was intended to maximize the upload speed, and a predetermined path are not imposed delays.

As the complexity of networks is increasing and the equipment becomes increasingly expensive, cases of underutilization of network resources strengthened the search for a different way of using the natural level of communication. A way that would allow the sharing of material from different user groups and the flexible allocation of additional resources to those groups with higher data exchange requirements.

Solving the above problem would come only by separating the decision level in relation to the physical signaling level. Furthermore, flexibility means easy change of equipment behavior, a behavior-based metrics use of communication channels in order to decide the required data traffic rate. This effort is now termed as **Software Defined Networking**.

Similar solutions have been given to other resources in the field of Information Technology. Processing power, magnetic storage media, as well as random access memory, a factor which may be available depending on the needs of users. This was the main idea of the area of mainframes up to Virtualization and Cloud Computing.

So following these examples, the software defined networking provides data streams according to existing needs and available equipment on which it operates. It is an abstraction mechanism allows the handling of communication channels by the software and not by the material itself.

Σκοπός της διπλωματικής

Το κύριο πεδίο εφαρμογής της παρούσας μεταπτυχιακής εργασίας είναι η θεωρητική μελέτη και υλοποίηση του Software-Defined Networking (SDN). Το SDN είναι μια αναδυόμενη αρχιτεκτονική, που είναι δυναμική, εύχρηστη, αποδοτική και ευπροσάρμοστη, ιδανική για υψηλού εύρους ζώνης εφαρμογές. Αυτή η αρχιτεκτονική αποσυνδέει τον έλεγχο του δικτύου από τις συσκευές προώθησης, επιτρέποντας στον έλεγχο του δικτύου να γίνει άμεσα προγραμματιζόμενος και σχετικός της υποδομής που πρέπει να υλοποιήσει τις εφαρμογές και τις υπηρεσίες δικτύου. Το πρωτόκολλο OpenFlow™ είναι ένα θεμελιώδες στοιχείο για την οικοδόμηση λύσεων SDN.

Για να επιτευχθεί αυτό θα παρουσιάσουμε πρώτα ένα θεωρητικό υπόβαθρο για τις τεχνολογίες SDN & OpenFlow™, θα γίνει βιβλιογραφική επισκόπηση (παρουσίαση των θεωρητικών και ερευνητικών (State of the Art) γνώσεων του αντικειμένου) και θα δείξουμε πώς το SDN λειτουργεί, ιδιαίτερα σε περιβάλλον σχολικών δικτύων.

Το δεύτερο στάδιο αποτελεί στην προσομοίωση ενός μικρού περιβάλλοντος SDN σε ένα εργαλείο προσομοιωτή. Στη συνέχεια θα ακολουθήσει προσομοίωση ενός πραγματικού περιβάλλοντος SDN με βάση ένα εικονικό περιβάλλον δικτύου, όπου είναι μέρος από το Πανελλήνιο Σχολικό Δίκτυο (Π.Σ.Δ.). Τέλος, θα γίνει η τροφοδότηση ροών κίνησης στο δίκτυο του Π.Σ.Δ. καθώς και ένα σενάριο για το πως μπορούμε να προσφέρουμε ποιότητα υπηρεσίας (QoS) με υλοποίηση SDN, το οποίο αποτελεί ένα ανοικτό ερευνητικό πρόβλημα.

Δομή κειμένου

Η εργασία απαρτίζεται από επτά συνολικά κεφάλαια. Στο παρόν κεφάλαιο γίνεται η πρώτη εισαγωγή και μια επεξήγηση της διάρθρωσης της εργασίας.

Στο πρώτο κεφάλαιο γίνεται μια αναλυτική περιγραφή για τα Software Defined Networks (SDN's), την τεχνολογία, την ιστορία, τα στοιχεία που απαρτίζουν τα SDN's καθώς και το τι ισχύει σήμερα σε αυτά τα δίκτυα (State of the Art).

Στο δεύτερο κεφάλαιο γίνεται μια περιγραφή του Openflow δίνοντας την αρχιτεκτονική του, παρουσιάζοντας το OF Switch, τις επικοινωνίες που υπάρχουν, το μηχανισμό προώθησης των πακέτων, τα χαρακτηριστικά και τέλος τους πιο δημοφιλείς ελεγκτές (Openflow Controllers) για τον τρόπο διαχείρισης του δικτύου.

Μια πιο εις βάθος ανάλυση των δικτυακών εξομοιωτών SDN που υπάρχουν γίνεται στο τρίτο κεφάλαιο. Η παρουσία δύο εξομοιωτών του ESTINET και του MININET ως προς την υλοποίηση δικτυακών τοπολογιών και υποστήριξης του Openflow δίνει μια δυνατότητα επιλογής για το ποιο είναι κατάλληλο για την υλοποίηση της διπλωματικής.

Στο τέταρτο κεφάλαιο έχουμε την περιγραφή των εργαλείων του Openflow Controller, του δικτυακού εξομοιωτή, του περιβάλλοντος εικονικοποίησης (Virtualbox) και του εικονικού μεταγωγέα (OpenVSwitch) που έχουμε επιλέξει για την υλοποίηση.

Τέλος, στα τρία κεφάλαια που υπολείπονται έχουμε την Αναφορά SDN σε περιβάλλοντα Σχολικών δικτύων, την Προσομοίωση Νησίδας SDN (πειραματικό στάδιο), την Προσομοίωση πραγματικού δικτύου (μέρος από το Π.Σ.Δ.) με την Τροφοδότηση πραγματικών flows στο παραπάνω Δίκτυο καθώς και την υλοποίηση του ανοικτού ερευνητικού προβλήματος για σενάρια QoS στα δίκτυα SDN με τα συμπεράσματα που προκύπτουν.

Κεφάλαιο 1:
Software-Defined Networking

1.1 Εισαγωγή

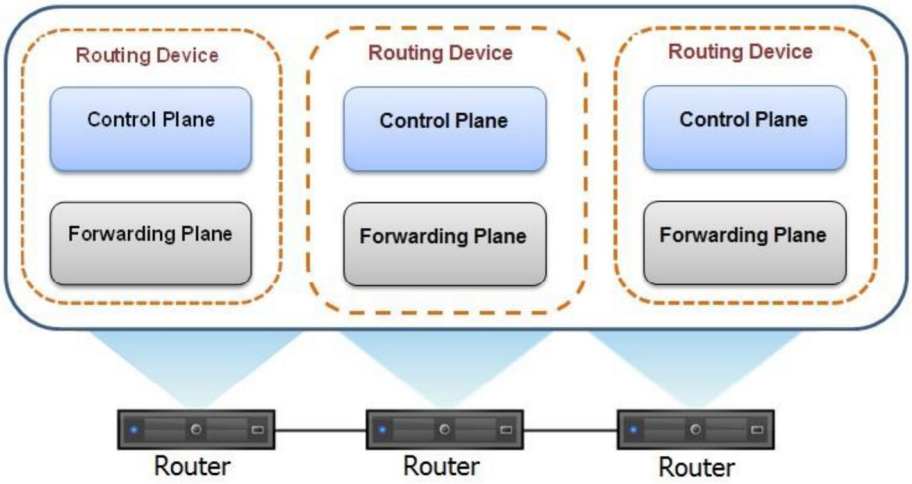
Σύμφωνα με τον επίσημο ορισμό από το Open Networking Foundation *«Το Software Defined Networking (SDN) είναι μια αναδυόμενη αρχιτεκτονική δικτύου, όπου το δίκτυο ελέγχου είναι διαχωρισμένο από το δίκτυο προώθησης και είναι άμεσα προγραμματιζόμενο. Αυτή η μετανάστευση του ελέγχου, πρώην συνδεδεμένη σε μεμονωμένες συσκευές δικτύου (δρομολογητές, μεταγωγείς), είναι προσιτή σε υπολογιστικές συσκευές επιτρέποντας στην υποκείμενη υποδομή να αντλήσει τις εφαρμογές και τις υπηρεσίες δικτύου, το οποίο μπορεί να αντιμετωπίσει το δίκτυο ως μία λογική ή εικονική οντότητα».* [1]

Ένας άλλος ορισμός του SDN είναι ο ακόλουθος: *«Ο φυσικός διαχωρισμός του επιπέδου ελέγχου του δικτύου από το επίπεδο προώθησης, και όπου το επίπεδο ελέγχου ελέγχει αρκετές συσκευές»*. [2]

1.2 Παραδοσιακές αρχιτεκτονικές δικτύων

Τα δίκτυα που γνωρίζουμε σήμερα είναι απομονωμένα και διαχωρίζονται με φυσικό τρόπο για να καλύψουν τις ανάγκες της βιομηχανίας, των παρόχων υπηρεσιών, οργανισμών και τελικών χρηστών. Στα δίκτυα αυτά τόσο το επίπεδο ελέγχου (control plane) όσο και το επίπεδο δεδομένων (data plane) υλοποιούνται στην ίδια συσκευή, όπως φαίνεται στην Εικόνα 1.

Παρά το γεγονός ότι η παραδοσιακή αρχιτεκτονική δικτύου έχει ανταπεξέλθει με αξιοθαύμαστο τρόπο στην εξέλιξη της δικτυακής τεχνολογίας, είναι δύσκολο, αν όχι αδύνατο, στον σημερινό κόσμο τα παραδοσιακά δίκτυα να ανταποκριθούν στις νέες εικονικές απαιτήσεις. Με περιορισμένους ή κατ' αποκοπή προϋπολογισμούς τα τμήματα πληροφορικής των επιχειρήσεων αναζητούν την εικονικοποίηση (virtualization) των περισσοτέρων από τους κεντρικούς υπολογιστές (servers) τους. Μια τέτοια διαδικασία είναι δύσκολη, δεδομένου ότι η ζήτηση τόσο για την εφαρμογή όσο και για την κινητικότητα των χρηστών αυξάνεται ραγδαία.



Εικόνα 1: Αρχιτεκτονική Παραδοσιακών Δικτύων

1.3 Όρια παραδοσιακών δικτύων

Καθώς οι υφιστάμενες παραδοσιακές αρχιτεκτονικές δικτύου δεν χτίστηκαν κατά τρόπο που να ανταποκρίνονται στις σημερινές απαιτήσεις τελικών χρηστών, πάροχων υπηρεσιών και επιχειρήσεων, έχουν προκύψει μερικοί περιορισμοί της παραδοσιακής αρχιτεκτονικής δικτύου, όπως [1]:

- **Πολυπλοκότητα διαχείρισης**: Οι υφιστάμενες τεχνολογίες δικτύων υπολογιστών είναι στηριγμένες σε ένα σύνολο πρωτοκόλλων δρομολόγησης που κατασκευάστηκαν για να συνδέσουν τους hosts (τελικούς χρήστες) κατά τρόπο αξιόπιστο και με υψηλές ταχύτητες ακόμα και σε μεγάλες αποστάσεις και σε ετερογενή δίκτυα. Προκειμένου, να καλυφθούν απαιτήσεις όπως η υψηλή διαθεσιμότητα, η ασφάλεια και η εκτεταμένη συνδεσιμότητα, κατά τη διάρκεια των τελευταίων δεκαετιών, τα πρωτόκολλα έχουν σχεδιαστεί με πολλούς τρόπους που οδηγούν στη λύση των συγκεκριμένων προβλημάτων, χωρίς να λάβουν υπόψη άλλα προβλήματα που πιθανά δημιουργούν. Μία τέτοια προσέγγιση έχει οδηγήσει σε ένα από τα κύρια προβλήματα που πρέπει να αντιμετωπίσουν οι διαχειριστές δικτύου σήμερα, όπως η πολυπλοκότητα διαχείρισης των δικτύων. Ένα παράδειγμα είναι ότι η προσθήκη ή η αφαίρεση μιας συσκευής σε ένα δίκτυο μπορεί να έχει μεγάλο φόρτο εργασίας για τους διαχειριστές δικτύων, επειδή διάφορα μέρη του δικτύου πρέπει να μετατραπούν, όπως λίστες πρόσβασης (Access Control Lists, ACL), εικονικά τοπικά δίκτυα (VLAN's), ποιότητα των υπηρεσιών (QoS), πρωτοκόλλα δρομολόγησης και τοπολογίες δικτύων.

  Εκτός από τα παραπάνω, η συμβατότητα των εκδόσεων του εξοπλισμού και του λογισμικού πρέπει να εξεταστεί πριν γίνει οποιαδήποτε τροποποίηση στο δίκτυο. Κατά συνέπεια, οι διαχειριστές δικτύων προτιμούν να κρατούν στατικό το δίκτυό τους, προκειμένου να αποφευχθεί ή να ελαχιστοποιηθεί ο χρόνος διακοπής υπηρεσιών που μπορεί να προκληθούν από οποιαδήποτε

αλλαγή. Η φύση του στατικού σχεδιασμού δικτύων περιορίζει τη δυναμική φύση εικονικοποίησης κεντρικών υπολογιστών (server virtualization), η οποία αυξάνει συνέχεια τον αριθμό των εξυπηρετητών που χρειάζονται συνδεσιμότητα.

- **Δυσκολία εφαρμογής πολιτικών**: Προκειμένου να διατηρήσουν μια πολιτική οι διαχειριστές δικτύου θα χρειαστεί να ρυθμίσουν αρκετούς δρομολογητές (routers), μεταγωγείς (switches) και μηχανισμούς. Στην εικονικοποίηση, κάθε φορά που ο διαχειριστής προσθέτει μία εικονική μηχανή (Virtual Machine – VM) στο δίκτυο, συνήθως καταλαμβάνει ώρες, αν όχι ημέρες, για να ρυθμίσει τις λίστες πρόσβασης (ACL) σε όλο το δίκτυο. Με το είδος του δικτύου και τη πολυπλοκότητα που διατρέχουμε σήμερα, έχει γίνει πολύ δύσκολο για τους διαχειριστές δικτύου να διατηρούν ένα σταθερό περιβάλλον για τα δικαιώματα πρόσβασης, την ποιότητα των υπηρεσιών και της ασφάλειας.

- **Θέματα Επεκτασιμότητας**: Καθώς τα κέντρα δεδομένων (data centers) έχουν μια υψηλή ζήτηση που αυξάνεται γρήγορα, το δίκτυο πρέπει να αυξάνεται με την ίδια ταχύτητα. Αλλά στην πραγματικότητα τα δίκτυα έχουν γίνει εξαιρετικά σύνθετα εξαιτίας της προσθήκης των εκατοντάδων εάν όχι χιλιάδων δρομολογητών, μεταγωγέων και firewalls που απαιτούνται για να είναι εύχρηστα και διαμορφώσιμα. Οι διαχειριστές δικτύων έχουν στηριχθεί σε μεγάλο ποσοστό στο υψηλό διαθέσιμο εύρος ζώνης (bandwidth) προκειμένου να επεκτείνουν τα επιχειρηματικά δίκτυα, αλλά με εικονικοποίηση των data centers η κίνηση των δεδομένων έχει γίνει ιδιαίτερα δυναμική, γεγονός που οδήγησε στη δυσκολία πρόβλεψης των μονοπατιών. Οι μεγάλες εταιρείες, όπως Amazon, eBay και Facebook αντιμετωπίζουν ακόμα περισσότερα ζητήματα επεκτασιμότητας. Τέτοια μεγάλη επεκτασιμότητα δικτύων είναι αδύνατη να ρυθμιστεί χειροκίνητα.

- **Εξάρτηση από τον προμηθευτή**: Οι φορείς και οι επιχειρήσεις επιδιώκουν να παρουσιάσουν νέες δυνατότητες και υπηρεσίες γρήγορα ως απάντηση στις μεταβαλλόμενες επιχειρησιακές ανάγκες ή στις απαιτήσεις των χρηστών τους. Εντούτοις, η δυνατότητά τους να αποκριθούν εμποδίζεται από τους κύκλους παραγωγής των προμηθευτών προϊόντων δικτυακού εξοπλισμού, όπου οι κύκλοι μπορούν να κυμαίνονται σε τρία ή περισσότερα έτη. Η έλλειψη τυποποιήσεων (standards) και ανοικτών διεπαφών (open interfaces) περιορίζουν τη δυνατότητα των παρόχων δικτυακών υπηρεσιών να προσαρμόσουν τα δίκτυα στα μεμονωμένα περιβάλλοντά τους.

  Αυτός ο κακός συνδυασμός μεταξύ των απαιτήσεων της αγοράς και των ικανοτήτων των δικτυών έχει φέρει τη βιομηχανία σε ένα σημείο αιχμής. Ως απάντηση, η βιομηχανία έχει δημιουργήσει το Software-Defined Networking (SDN) και αναπτύσσει τα σχετικά πρότυπα.

1.4 Εισαγωγή στην τεχνολογία SDN

Ο κατανεμημένος έλεγχος και τα δικτυακά πρωτόκολλα μεταφοράς τρέχουν σε δρομολογητές (routers) και μεταγωγείς (switches), επιτρέποντας έτσι στα δικτυακά πακέτα να ταξιδεύουν μέσα στον δικτυακό κόσμο [3]. Παρόλο της ευρείας τους αποδοχής τα παραδοσιακά IP δίκτυα είναι σύνθετα και δύσκολα στην διαχείρισή τους. Για να μπορέσουμε να επιτύχουν μια αποδοτική πολιτική δικτύου οι πάροχοι του δικτύου θα πρέπει να ρυθμίσουν ξεχωριστά και με ειδικές εντολές την κάθε δικτυακή συσκευή. Αυτό τους οδηγεί σε μια πολυπλοκότητα ρυθμίσεων, κάνοντας έτσι το δίκτυο ευάλωτο σε λάθη καθώς και στην αναγκαιότητα επανεκκίνησης των συσκευών για να φορτώσουν τις αλλαγές των ρυθμίσεων. Έτσι τα τωρινά IP δίκτυα δεν επιτρέπουν τις άμεσες αλλαγές ρυθμίσεων και εφαρμογής τους από τις δικτυακές συσκευές.

Το επίπεδο ελέγχου (control plane), το οποίο είναι υπεύθυνο για την διαχείριση της δικτυακής κίνησης και το επίπεδο δεδομένων (data plane), το οποίο προωθεί την κίνηση σύμφωνα με τις αποφάσεις του επιπέδου ελέγχου είναι ενσωματωμένα στις δικτυακές συσκευές, κάνοντας έτσι τη δικτυακή υποδομή να μην είναι ευέλικτη σε αλλαγές και να μην μπορούμε εύκολα να την εξελίξουμε.

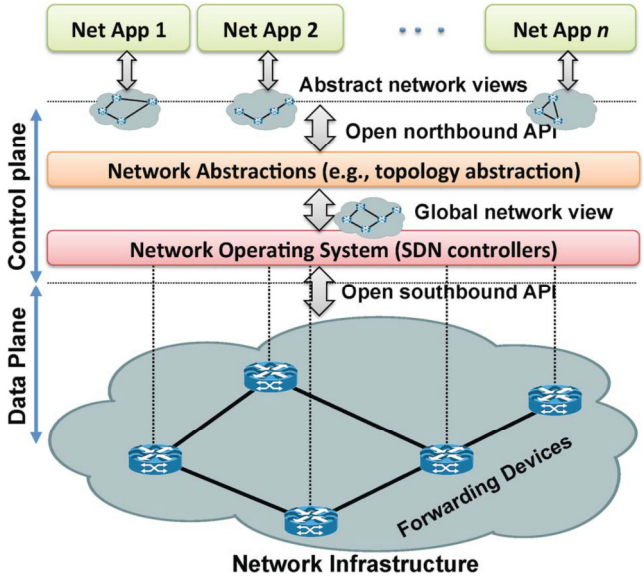
Η τεχνολογία Software-Defined Networking (SDN) προσφέρει την αλλαγή των περιορισμών των τωρινών δικτυακών υποδομών, δηλαδή τον διαχωρισμό της λογικής ελέγχου του δικτύου από τις συσκευές των δρομολογητών και των μεταγωγέων, όπου προωθούν την κίνηση των πακέτων. Αυτός ο διαχωρισμός κάνει απλές τις συσκευές προώθησης, καθώς η λογική ελέγχου υλοποιείται σε έναν κεντρικό ελεγκτή (controller), με αποτέλεσμα την εύκολη υλοποίηση πολιτικών, ρυθμίσεων και εξέλιξης του δικτύου. Ο διαχωρισμός αυτός πραγματοποιείται από μια προγραμματιζόμενη διεπαφή (interface) μεταξύ των μεταγωγέων και του SDN ελεγκτή. Ο ελεγκτής επιτρέπει τον έλεγχο της κατάστασης των στοιχείων που αποτελούν το επίπεδο δεδομένων μέσω μίας προγραμματιζόμενης διεπαφής εφαρμογής (API), π.χ. του OpenFlow.

Η τεχνολογία του SDN και του Openflow ξεκίνησε αρχικά πειραματικά από την ακαδημαϊκή κοινότητα, αλλά υποστηρίχτηκε γρήγορα από μεγάλες επιχειρήσεις, όπως Google, Facebook, Yahoo, Microsoft, Verizon και Deutsche Telekom, ιδρύοντας έτσι το Open Networking Foundation (ONF) με μοναδικό σκοπό του την εξέλιξη, την προώθηση και την αποδοχή του SDN.

Όπως ορίζεται το SDN αναφέρεται ως μια δικτυακή αρχιτεκτονική που η προώθηση κίνησης στο επίπεδο δεδομένων είναι διαχειρίσιμη από ένα απομακρυσμένο επίπεδο ελέγχου, διαχωρισμένο από τις συσκευές, Εικόνα 2.

Η αρχιτεκτονική του SDN αποτελείται από 4 πυλώνες:

1. Το επίπεδο ελέγχου είναι διαχωρισμένο από το επίπεδο δεδομένων .
2. Οι αποφάσεις προώθησης γίνονται βάση ροών και όχι με βάση τον προορισμό τους.
3. Η λογική ελέγχου βρίσκεται σε μία εξωτερική οντότητα που ονομάζεται SDN Controller ή Network Operating System (NOS).
4. Το δίκτυο είναι προγραμματιζόμενο διαμέσου εφαρμογών που τρέχουν στον Controller.



Εικόνα 2: Αρχιτεκτονική SDN

Σε ότι αφορά την δικτυακή αρχιτεκτονική, το SDN αναμένεται να αλλάξει τον τρόπο που οι μηχανικοί και οι σχεδιαστές δικτυών χτίζουν και ενεργοποιούν τα δίκτυά τους. Με την εισαγωγή του SDN, τα δίκτυα έχουν γίνει ανοικτά πρότυπα, μη εξειδικευμένα σε υλικό – λογισμικό και εύκολα στον προγραμματισμό και στην διαχείρισή τους. Το SDN θα δώσει στις επιχειρήσεις και στους παρόχους καλύτερο και περισσότερο έλεγχο των δικτύων τους, επιτρέποντας προσαρμογή και βελτιστοποίηση ώστε να μειωθεί το γενικό κόστος των δικτύων.

Μερικά από τα κύρια οφέλη του SDN είναι [4]:

- **Απλότητα στη διαχείριση του δικτύου**: Με το SDN το δίκτυο μπορεί να αντιμετωπισθεί και να ρυθμιστεί ως ένας ενιαίος κόμβος που θα μεταφέρει σε αφαιρετικό επίπεδο τις απαιτητικές εργασίες διαχείρισης του δικτύου, για να γίνει εύκολη η διαχείριση μέσα από τις διεπαφές (interfaces).

- **Γρήγορη ανάπτυξη υπηρεσιών**: Τα νέα χαρακτηριστικά του δικτύου και οι εφαρμογές μπορούν να αναπτυχθούν κατά τρόπο γρήγορο εντός μερικών ωρών αντί πολλών ημερών.

- **Αυτοματοποίηση ρυθμίσεων**: Χειροκίνητες εργασίες διαμόρφωσης, όπως η ανάθεση VLAN και η ρύθμιση QoS μπορούν να γίνουν αυτόματα.

- **Εικονικοποίηση δικτύου (Network Virtualization)**: Από τη στιγμή που η εικονικοποίηση των εξυπηρετητών (servers) και των συσκευών αποθήκευσης (storages) έχει αναπτυχθεί πριν από τα δίκτυα μπορεί να ωφελήσει και στην εικονικοποίηση του SDN.

- **Μείωση της λειτουργικής δαπάνης**: Με όφελος από την αυτοματοποίηση της ανάπτυξης του δικτύου, μία αλλαγή σε αυτό δεν ήταν ποτέ πιο εύκολη

διαδικασία και ως αποτέλεσμα μειώνονται τα κόστη της λειτουργίας του δικτύου.

1.5 Ιστορική εξέλιξη της τεχνολογίας SDN

Το 1980 υπήρχε η έννοια του κεντρικοποιημένου ελέγχου του δικτύου καθώς τα δεδομένα και ο έλεγχος γίνονταν μέσα από το ίδιο κανάλι, υπήρχαν ειδικές συχνότητες για αρχικοποίηση της γραμμής ή για τη δρομολόγηση της τηλεφωνικής κλήσης. Αργότερα, το 1981 η AT&T διαχώρισε το επίπεδο Δεδομένων από το επίπεδο Ελέγχου μέσω μίας οντότητας που ονομαζόταν Network Point και από εκεί μπορούσαν εύκολα να καταλάβουν αν η γραμμή ήταν δεσμευμένη.

Το 1990 εμφανίστηκαν τα Active Networks σαν μία έννοια των προγραμματιζόμενων δικτύων (Programmable Networks), όπου αποτελούνταν από μεταγωγείς που έκαναν κάποιους υπολογισμούς στα πακέτα, όπως το ίχνος (trace) σε κάθε δρομολογητή, firewalls, proxies και κάποιες εφαρμογές. Έρευνα της DARPA το 1994-95 [5] βρίσκει προβλήματα στα δίκτυα αυτά, όπως δυσκολία προσαρμογής σε νέες τεχνολογίες, μικρή απόδοση λόγω των πολλών επιπέδων πρωτοκόλλων καθώς και δυσκολίες σε νέες υπηρεσίες. Ήταν κάτι καινοτόμο και πήρε 10 χρόνια για να γίνει πρότυπο, να προμηθευτεί η αγορά, να υλοποιηθεί και να αναπτυχθεί. Αποτελείται από active nodes που δίνουν τη δυνατότητα στους δρομολογητές να δίνουν στην υποδομή του δικτύου νέες υπηρεσίες κάτι που το κάνει να μοιάζει στα Software Defined Networks (SDN's). Έχει δύο προσεγγίσεις, το Capsules, όπου το κάθε μήνυμα είναι πρόγραμμα και αξιολογείται από το Node που μεταφέρει τα πακέτα και τα Programmable Switches όπου συναρτήσεις τρέχουν στους δρομολογητές και τα πακέτα δρομολογούνται μέσω των προγραμματιζόμενων κόμβων καθώς το πρόγραμμα που τρέχει για τη δρομολόγηση του πακέτου εξαρτάται από το περιεχόμενο στο header του πακέτου. Η τεχνολογία αυτή είχε κάποια μειονεκτήματα, όπως δεν ήταν καθαρή η έννοια του χρόνου, είχε ακριβό υλικό, υπήρχαν λάθη στη ασφάλεια, αφού τα πακέτα περιέχουν τον κώδικα του προγράμματος και στη διαλειτουργικότητα του δικτύου, δηλαδή στην ικανότητα να στέλνονται και να λαμβάνονται πακέτα συνεχώς από το δίκτυο για QoS του χρήστη χωρίς να δημιουργούνται προβλήματα στο δίκτυο. Όλα αυτά τα προβλήματα οδήγησαν στο συμπέρασμα ότι έπρεπε να υπάρχει μία υλοποίηση, όπου θα ήταν συμβατή με τη προς τα πίσω τεχνολογία και να λειτουργεί με την ίδια υποδομή. Τη λύση αυτή μας δίνει το πρωτόκολλο του Openflow.

Επίσης, το 1990 αναπτύσσεται η ιδέα του Network Virtualization, όπου μας δίνει την αναπαράσταση μίας ή περισσοτέρων δικτυακών τοπολογιών από την ίδια δικτυακή υποδομή. Σαν τέτοιες περιπτώσεις βρίσκουμε τα VLAN's, Tempest, Vini, Cabo, VMWare και Nicira [6]. Μας δίνουν πολλά πλεονεκτήματα, όπως διαμοιρασμό των πόρων καθώς πολλοί δρομολογητές λειτουργούν πάνω σε μία πλατφόρμα και χρησιμοποιούν τη CPU, RAM, Forwarding Tables και Bandwidth καθώς και μία δυνατότητα προσωποποίησης στην οποία μπορούμε να έχουμε το δικό μας λογισμικό για τις πολιτικές δρομολόγησης (routing) και προώθησης (forwarding) μέσω CPU γενικού σκοπού και Network processors & FPGA's για το επίπεδο Δεδομένων.

Όλα αυτά τα στοιχεία αποτελούν μία κληρονομιά για το SDN γιατί:

- Διαχωρίζονται οι υπηρεσίες από την υποδομή του δικτύου.
- Μπορούμε να έχουμε πολλαπλούς Controllers σε ένα Switch.
- Μπορούμε να έχουμε πολλές λογικές τοπολογίες.

Για να γίνει όμως ο έλεγχος σε ένα δίκτυο μεταγωγής πακέτων θα πρέπει να έχουμε τον διαχωρισμό του επιπέδου Ελέγχου (Control Plane) από το επίπεδο Δεδομένων (Data Plane).

Το επιπέδου Ελέγχου (Control Plane) αποτελεί τη λογική για τον έλεγχο της συμπεριφοράς προώθησης των πακέτων, π.χ. routing πρωτόκολλα, φίλτρα, firewalls, NAT, security, load balancer.

Το επίπεδο Δεδομένων (Data Plane) είναι υπεύθυνο για τη προώθηση της κίνησης σύμφωνα με τη λογική του επιπέδου Ελέγχου, π.χ. ip forwarding – Layer 2 Switching.

Ο διαχωρισμός των επιπέδων αυτών μας οδηγεί σε κάποια πλεονεκτήματα:

- Ταχύτερη καινοτομία: Η λογική του ελέγχου δεν είναι δεμένη με το hardware και έτσι έχουμε την ανεξάρτητη εξέλιξη και ανάπτυξη του λογισμικού του Controller του δικτύου σε σχέση με την υποδομή του δικτύου.
- Ευρεία απεικόνιση του δικτύου: Μπορούμε να συμπεράνουμε ευκολότερα τη συμπεριφορά του δικτύου, επιτρέποντας έτσι τον έλεγχο του δικτύου από ένα high-level πρόγραμμα που βλέπουμε το δίκτυο, την αποσφαλμάτωσή του και το τσεκάρισμα των λειτουργιών του.
- Ευκαμψία (flexible): Παρουσίαση των νέων υπηρεσιών πιο εύκολα.

Η ανάπτυξη του ελέγχου στα δίκτυα μεταγωγής πακέτων ξεκίνησε από το 2003. Εκεί το FORCES - [7] χρησιμοποιεί πρωτόκολλα για πολλαπλά στοιχεία ελέγχου (Control Elements) και στοιχεία προώθησης της κίνησης (Forwarding Elementes) δίνει μία ιδέα σαν το Openflow αλλά χρειάζεται να γίνει πρότυπο, να υιοθετηθεί και να δημιουργηθεί νέο υλικό.

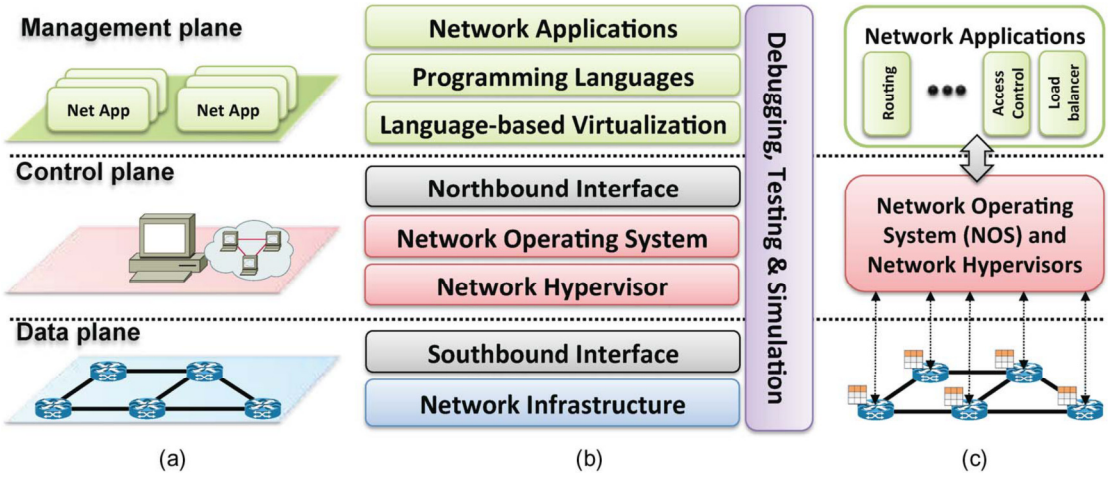
Το 2004 υπήρξε η ιδέα να χρησιμοποιηθούν υπάρχοντα πρωτόκολλα για τον υπολογισμό των διαδρομών με σκοπό την προώθηση των πακέτων και ονομάζεται Routing Control Platform [7]. Εδώ, ο υπολογισμός γίνεται σε ένα σημείο του Αυτόνομου Συστήματος (Autonomous System) και όταν βρεθούν οι διαδρομές διαφημίζονται μέσω του πρωτοκόλλου iBGP, αλλά ο έλεγχος περιορίζεται από την υποστήριξη που έχουν τα υπάρχοντα πρωτόκολλα.

Το 2007 δημιουργήθηκε το Ethane [7] και λειτουργεί ως αρχιτεκτονική δικτύου για επιχειρήσεις και έχει ως βάση τις άμεσες εκτελούμενες πολιτικές δικτύου. Ο Domain Controller υπολογίζει τα flow tables με βάση τις πολιτικές που υπάρχουν αλλά χρειάζονται τροποποιημένα switches, όπως OpenWrt- NetFPGA και Linux που να υποστηρίζουν το Ethane.

Το 2008 έχουμε την παρουσίαση του Openflow. Ο controller ομιλεί με τους forwarding tables του switch και αυτό που χρειάζεται είναι να ανοίξουν οι κατασκευαστές μία διεπαφή (interface) έτσι ώστε να δέχεται ο μεταγωγέας τις ροές κίνησης από τον ελεγκτή.

1.6 Αρχιτεκτονική SDN

Μια αρχιτεκτονική SDN μπορεί να περιγραφεί ως μια σύνθεση των διαφορετικών στρωμάτων δικτύου, όπως φαίνεται στην Εικόνα 3(b). Κάθε στρώμα έχει τις δικές του συγκεκριμένες λειτουργίες, όπως southbound API, NOSs, northbound API, network applications και άλλα [3].



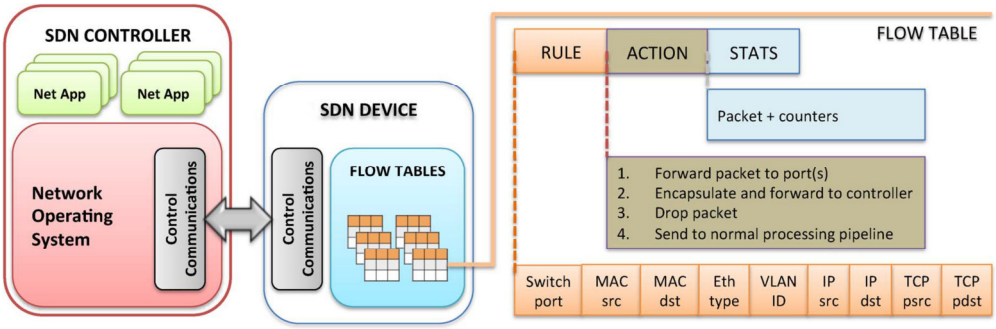
Εικόνα 3: Software-Defined Networking αρχιτεκτονική σε επίπεδα (a), στρώματα (b) και σχεδιασμό συστήματος (c)

Η Εικόνα 3(a) παρουσιάζει τη γενική αρχιτεκτονική του SDN τριών επιπέδων. Τα στρώματα του SDN φαίνονται στην Εικόνα 3(b) και μία απεικόνιση της αρχιτεκτονικής σχεδιασμού συστήματος έχουμε στην Εικόνα 3(c).

Στις παρακάτω ενότητες παρουσιάζονται συνοπτικά τα επίπεδα που προκύπτουν από την Εικόνα **3**(b).

1.6.1 Υποδομή SDN δικτύου

Η υποδομή του SDN είναι σχεδόν ίδια με αυτή των παραδοσιακών δικτύων, δηλαδή αποτελείται από δικτυακό εξοπλισμό (δρομολογητές, μεταγωγείς, κλπ). Η κυριότερη διαφορά βρίσκεται στο γεγονός ότι οι φυσικές συσκευές είναι απλά στοιχεία προώθησης της κίνησης χωρίς να έχουν κάποιο ενσωματωμένο λογισμικό για τον έλεγχο, ούτε λαμβάνουν αποφάσεις προώθησης των πακέτων. Η «εξυπνάδα» του δικτύου βρίσκεται σε ένα κεντρικό σύστημα ελέγχου το οποίο ονομάζεται Network Operating System (NOS). Αυτά τα δίκτυα βρίσκονται πάνω από κάποιες διεπαφές (Openflow), οι οποίες επιτρέπουν τον δυναμικό προγραμματισμό ετερογενών στοιχείων προώθησης κάτι που ήταν δύσκολο μέχρι τώρα.



Εικόνα 4: Συσκευές SDN που υποστηρίζουν OpenFlow

Σε μια αρχιτεκτονική SDN / OpenFlow υπάρχουν δύο κύρια στοιχεία, οι ελεγκτές (SDN Controller) και οι συσκευές προώθησης (SDN device), όπως φαίνεται στην Εικόνα 4. Μια συσκευή του επιπέδου δεδομένων είναι ένα υλικό ή στοιχείο λογισμικού που ειδικεύεται στην προώθηση των πακέτων, ενώ ελεγκτής είναι μια στοίβα λογισμικού (ο «εγκέφαλος του δικτύου») που τρέχει σε ένα μηχάνημα. Μια OpenFlow enabled συσκευή προώθησης βασίζεται σε έναν αγωγό που έχει πίνακες-ροής, όπου κάθε εγγραφή ενός πίνακα-ροής εξυπηρετεί τρεις λειτουργίες:

1. Να αντιστοιχηθούν οι ροές με τους κανόνες
2. Ενέργειες που πρέπει να εκτελεστούν όταν έχουμε αντιστοιχία των πακέτων
3. Μετρητές που κρατούν στατιστικά στοιχεία των πακέτα.

Αυτό το υψηλού επιπέδου και απλοποιημένο μοντέλο που προέρχεται από το OpenFlow είναι σήμερα ο πιο διαδεδομένος σχεδιασμός του SDN για τις συσκευές στο επίπεδο δεδομένων. Παρ 'όλα αυτά, υπάρχουν και άλλες προδιαγραφές για τις συσκευές προώθησης που υποστηρίζουν SDN, όπως οι POF και negotiable datapaht models (NDMs) από τις ONF Forwarding Abstractions Working Group (FAWG) [3].

Μέσα σε μια συσκευή OpenFlow ένα μονοπάτι καθορίζεται μέσα από μια αλληλουχία πινάκων-ροής για το πώς πρέπει να αντιμετωπιστούν τα πακέτα. Όταν φθάνει ένα νέο πακέτο, η διαδικασία αναζήτησης ξεκινάει από τον πρώτο πίνακα και τελειώνει είτε με μία αντιστοίχιση σε έναν από τους πίνακες της αλληλουχίας ή με μια αστοχία (όταν δεν υπάρχει κανόνας για το πακέτο). Ένας κανόνας-ροής μπορεί να καθορίζεται από το συνδυασμό διαφορετικών πεδίων που ταιριάζουν, όπως απεικονίζεται στην Εικόνα 4. Εάν δεν υπάρχει κανένας κανόνας, το πακέτο θα απορριφθεί. Ωστόσο, η κοινή πρακτική είναι να εγκαταστήσουμε έναν προεπιλεγμένο κανόνα που λέει στον μεταγωγέα να στείλει το πακέτο στον ελεγκτή OpenFlow ή ακόμα και σε μεταγωγείς που δεν υποστηρίζουν OpenFlow. Οι προτεραιότητες των κανόνων ακολουθούν τη φυσική αλληλουχία του αριθμού των πινάκων και της τάξης των σειρών σε έναν πίνακα-ροής. Οι δυνατές ενέργειες που μπορεί να δεχτεί ένα πακέτο είναι:

1. Να προωθηθεί το πακέτο στην εξερχόμενη πόρτα (ες)
2. Να ενθυλακωθεί το πακέτο και να προωθηθεί στον ελεγκτή
3. Να απορριφθεί το πακέτο
4. Να σταλεί με την κανονική αλληλουχία της επεξεργασίας

5. Να σταλεί σε έναν επόμενο πίνακα-ροής ή σε ειδικούς πίνακες, όπως ομάδες ή πίνακες-μέτρησης που εισάγονται στην τελευταία έκδοση του πρωτοκόλλου OpenFlow.

1.6.2 Southbound Interfaces

Το Southbound Interfaces (ή Southbound APIs) είναι διασυνδεδεμένες γέφυρες ανάμεσα στον έλεγχο και στα στοιχεία προώθησης που επιτρέπουν σε ένα SDN δίκτυο να επικοινωνήσει με συστατικά κατώτερων επιπέδων. Σαν ένα κεντρικό στοιχείο στο σχεδιασμό του SDN, τα Southbound APIs αποτελούν ένα φραγμό για την εισαγωγή και αποδοχή οποιασδήποτε νέας τεχνολογίας στη δικτύωση. Υπό αυτό το πρίσμα, η εμφάνιση των SDN προτάσεων για τα Southbound API, όπως το OpenFlow θεωρείται ευπρόσδεκτο από πολλούς στον κλάδο.

Αυτά τα πρότυπα προώθησης της διαλειτουργικότητας, επιτρέπουν η ανάπτυξη των συσκευών δικτύου να γίνεται ανταγωνιστική. Αυτό έχει ήδη αποδειχθεί από την διαλειτουργικότητα μεταξύ των OpenFlow-enabled συσκευών από διαφορετικούς προμηθευτές.

Το OpenFlow είναι η πιο ευρέως αποδεκτή λύση και αναπτύσσεται σαν ανοικτό πρότυπο Southbound API για το SDN. Παρέχει μια κοινή προδιαγραφή για τις OpenFlow-enabled συσκευές προώθησης, καθώς και για την επικοινωνία μέσω ενός καναλιού μεταξύ των συσκευών του επιπέδου δεδομένων και του επίπεδου ελέγχου (π.χ. μεταγωγείς και ελεγκτές).

Το πρωτόκολλο OpenFlow παρέχει τις ακόλουθες τρεις πηγές πληροφόρησης στο NOSs:

- Μηνύματα βασισμένα στα γεγονότα αποστέλλονται από συσκευές προώθησης στον ελεγκτή όταν μία σύνδεση ή μία πόρτα έχει αλλαγή.
- Στατιστικά στοιχεία ροών που δημιουργούνται από τις συσκευές προώθησης στέλνονται και συλλέγονται από τον ελεγκτή.
- Μηνύματα τύπου packet-in αποστέλλονται από τις συσκευές προώθησης στον ελεγκτή όταν δεν γνωρίζουν τι πρέπει να κάνουν με τη νέα εισερχόμενη ροή ή επειδή υπάρχει ενέργεια τύπου «send to controller» αντιστοιχισμένη στην είσοδο του πίνακα-ροής.

Αυτές οι πληροφορίες είναι βασικές για να παρέχουν πληροφορίες του επιπέδου-ροής στο NOS. Παρ’ όλα αυτά, υπάρχουν και άλλες προδιαγραφές για την υλοποίηση Southbound API’s, όπως ForCES, Open vSwitch Database, POF, OpFLEX, OpenState, κλπ [3].

1.6.3 Network Hypervisors

Ένα από τα ενδιαφέροντα χαρακτηριστικά των τεχνολογιών εικονικοποίησης (virtualization) σήμερα, είναι το γεγονός ότι οι εικονικές μηχανές μπορούν εύκολα να μεταφερθούν από ένα φυσικό server στον άλλο και ότι μπορούν να δημιουργηθούν ή και να διαγραφούν κατά απαίτηση, επιτρέποντας έτσι την ευέλικτη και εύκολη διαχείρισή τους.

Παρά από τις μεγάλες προόδους στην εικονικοποίηση υπολογιστών και των στοιχείων αποθήκευσης, το δίκτυο εξακολουθεί να παραμένει στατικό, ρυθμιζόμενο στην εικονικοποίηση με ένα τρόπο box-by-box, δηλαδή την κάθε μία δικτυακή συσκευή ανεξάρτητα από τις υπόλοιπες. Οι κύριες απαιτήσεις του δικτύου αφορούν την τοπολογία του δικτύου και τον χώρο διευθύνσεων. Διαφορετικό φόρτο εργασίας απαιτούν διαφορετικές τοπολογίες δικτύου και υπηρεσίες, όπως υπηρεσίες επιπέδου L2 ή L3, ή ακόμη περισσότερο σύνθετες υπηρεσίες επιπέδων L4-L7 για προηγμένη λειτουργικότητα. Επί του παρόντος, είναι πολύ δύσκολο για μια ενιαία φυσική τοπολογία να υποστηρίξει τις ποικίλες απαιτήσεις των εφαρμογών και υπηρεσιών.

Ομοίως, ο χώρος ip διευθύνσεων είναι δύσκολο να αλλάξει στα τρέχοντα δίκτυα. Ως εκ τούτου, είναι δύσκολο να κρατήσουμε τις αρχικές δικτυακές ρυθμίσεις ένος μισθωτή, αφού οι εικονικές μηχανές δεν μπορούν να μεταναστεύσουν σε αυθαίρετες τοποθεσίες και το σχήμα των ip διευθύνσεων να είναι σταθερό και δύσκολο να αλλάξει. Για παράδειγμα, το IPv6 δεν μπορεί να χρησιμοποιηθεί από τις εικονικές μηχανές (VMs) του μισθωτή, εφόσον το φυσικό επίπεδο που αποτελείται από συσκευές προώθησης υποστηρίζει μόνο IPv4.

Για να παρέχεται πλήρη εικονικοποίηση, το δίκτυο θα πρέπει να παρέχει παρόμοιες ιδιότητες με το υπολογιστικό στρώμα. Η δικτυακή υποδομή θα πρέπει να είναι σε θέση να υποστηρίζει αυθαίρετα τις τοπολογίες δικτύου και τα σχήματα των διευθύνσεων. Κάθε μισθωτής θα πρέπει να έχει τη δυνατότητα να ρυθμίζει τους υπολογιστικούς κόμβους και το δίκτυο ταυτόχρονα. Η μετανάστευση των hosts θα πρέπει να ενεργοποιήσει αυτόματα τη μετανάστευση των αντίστοιχων εικονικές θυρών του δικτύου. Θα μπορούσε κανείς να σκεφτεί ότι μακρινές τεχνικές virtualization, όπως τα VLANs (εικονοποίηση L2 τομέα), το NAT (εικονοποίηση χώρου διευθύνσεων IP) και το MPLS (εικονοποίηση διαδρομής) είναι αρκετά για να παρέχουν πλήρη και αυτοματοποιημένη εικονικοποίηση δικτύου. Ωστόσο, αυτές οι τεχνολογίες είναι προσαρμοσμένες σε ρυθμίσεις box-by-box, δηλαδή, δεν υπάρχει μία ενιαία ενωτική αφαίρεση για να ρυθμιστεί (ή να επαναρυθμιστεί) το δίκτυο με ένα σφαιρικό τρόπο. Κατά συνέπεια, η δικτυακή τροφοδοσία μπορεί να πάρει μήνες, ενώ η υπολογιστική τροφοδοσία διαρκεί μόνο λίγα λεπτά.

1.6.4 Network Operating Systems (NOS)/Controllers

Τα παραδοσιακά λειτουργικά συστήματα παρέχουν αφαιρέσεις (π.χ., υψηλού επιπέδου APIs) για την πρόσβαση σε χαμηλότερου επίπεδου συσκευές, τη διαχείριση και την ταυτόχρονη πρόσβαση σε υποκειμενικούς πόρους (π.χ., μονάδα σκληρού δίσκου, προσαρμογέα δικτύου, CPU, μνήμη), και παρέχουν μηχανισμούς προστασίας για την ασφάλεια. Αυτές οι λειτουργίες και οι πόροι είναι οι κύριοι μοχλοί για την αύξηση της παραγωγικότητας, κάνοντας τη χρήση του συστήματος και των εφαρμογών πιο εύκολη για τους προγραμματιστές. Στην ευρεία χρήση τους έχει συμβάλει η εξέλιξη των διαφόρων συστημάτων (π.χ. γλώσσες προγραμματισμού) και η ανάπτυξη πολλών εφαρμογών. Σε αντίθεση, τα δίκτυα μέχρι στιγμής διαχειρίζονται και ρυθμίζονται χρησιμοποιώντας χαμηλού επίπεδου σύνολα οδηγιών, προσαρμοσμένα στο υλικό από κλειστά λειτουργικά συστήματα NOSs (π.χ. Cisco IOS και Juniper Junos). Επιπλέον, η ιδέα των λειτουργικών

συστημάτων να αφαιρεί χαρακτηριστικά συγκεκριμένων συσκευών και να παρέχει, με διαφανή τρόπο, κοινές λειτουργίες είναι κάτι που ακόμα απουσιάζει από τα δίκτυα. Για παράδειγμα, σήμερα οι σχεδιαστές των πρωτοκόλλων δρομολόγησης πρέπει να ασχοληθούν με πολύπλοκους κατανεμημένους αλγορίθμους κατά την επίλυση προβλημάτων δικτύωσης. Έτσι οι επαγγελματίες δικτύων λύνουν τα ίδια προβλήματα ξανά και ξανά.

Το SDN έχει υποσχεθεί να διευκολύνει τη διαχείριση του δικτύου και να ελαφρύνει το βάρος της επίλυσης των προβλημάτων διαχείρισης, μέσω της δικτύωσης του λογικά δοσμένου κεντρικού ελέγχου που προσφέρεται από ένα NOS. Όπως και με τα παραδοσιακά λειτουργικά συστήματα, η κρίσιμη αξία του NOS είναι να παρέχει αφαιρέσεις, βασικές υπηρεσίες και κοινά API για τους προγραμματιστές. Κάποιες υπηρεσίες που παρέχονται από το NOS είναι η γενική λειτουργία της κατάστασης του δικτύου και πληροφορίες σχετικές με τη δικτυακή τοπολογία, η αναζήτηση συσκευών και η κατανομή των δικτυακών ρυθμίσεων. Με το NOS για παράδειγμα, ο προγραμματιστής δεν θα χρειάζεται πλέον να νοιάζεται για τις λεπτομέρειες χαμηλού επιπέδου της διανομής δεδομένων μεταξύ των στοιχείων δρομολόγησης ώστε να καθορίσει τις πολιτικές του δικτύου. Τέτοια συστήματα μπορούν αναμφισβήτητα να δημιουργήσουν ένα νέο περιβάλλον που μπορούν να ενισχύσουν την καινοτομία σε ένα ταχύτερο ρυθμό, μειώνοντας την εν γένη πολυπλοκότητα της δημιουργίας νέων πρωτοκόλλων δικτύου και δικτυακών εφαρμογών. Παρόμοια με ένα παραδοσιακό λειτουργικό σύστημα, η πλατφόρμα ελέγχου αφαιρεί το χαμηλότερο επίπεδο λεπτομερειών της σύνδεσης και αλληλεπιδρά με τις συσκευές προώθησης (δηλαδή, την υλοποίηση πολιτικών δικτύου).

1.6.5 Northbound Interfaces

Οι διασυνδέσεις Northbound και Southbound αποτελούν τις δύο βασικές αφαιρέσεις του συστήματος SDN, που επιτρέπουν την επικοινωνία με τα ανώτερα και τα κατώτερα επίπεδα δικτύου, αντίστοιχα.

Ενώ η διεπαφή Southbound έχει ήδη μία ευρέως αποδεκτή πρόταση (OpenFlow), μια κοινή διεπαφή Northbound εξακολουθεί να είναι ένα ανοικτό ζήτημα. Αυτή τη στιγμή δεν μπορεί να οριστεί ένα πρότυπο για διεπαφή Northbound, καθώς οι περιπτώσεις χρήσης εξακολουθούν να είναι σε επεξεργασία. Εν πάση περιπτώσει, πρέπει να αναμένεται μια κοινή διεπαφή Northbound να προκύψει αφού το SDN εξελίσσεται. Μια αφαίρεση που θα επιτρέπει στις εφαρμογές δικτύου να μην εξαρτάται από τις συγκεκριμένες υλοποιήσεις, είναι σημαντικό να διερευνηθεί πλήρως για το SDN. Η διεπαφή Northbound είναι ως επί το πλείστον ένα σύστημα λογισμικού και δεν είναι ένα υλικό, όπως είναι η περίπτωση του Southbound API. Η υλοποίηση είναι συνήθως ένας πρωτοποριακός οδηγός, ενώ τα πρότυπα που θα προκύψουν αργότερα και οδηγούν ουσιαστικά την ευρεία υιοθέτηση τους. Εντούτοις, αρχικά και ελάχιστα πρότυπα για τις διεπαφές Southbound μπορούν ακόμα να διαδραματίσουν έναν σημαντικό ρόλο για το μέλλον του SDN. Η εμπειρία από την ανάπτυξη των διαφορετικών ελεγκτών θα είναι βεβαίως η βάση για μια κοινή εφαρμογή επιπέδου διεπαφής.

1.6.6 Language-Based Virtualization

Δύο βασικά χαρακτηριστικά των λύσεων virtualization είναι η ικανότητα να εκφράζουν αυτονομία και να επιτρέπουν διαφορετικά επίπεδα αφαίρεσης, ενώ συνεχίζονται να εγγυούνται οι επιθυμητές ιδιότητες, όπως η προστασία. Για παράδειγμα, οι τεχνικές virtualization μπορούν να επιτρέψουν διαφορετικές όψεις σε μία ενιαίου φυσικής υποδομής. Για παράδειγμα, ένα εικονικό «μεγάλο switch» θα μπορούσε να αποτελέσει ένα συνδυασμό πολλών συσκευών προώθησης. Αυτό απλοποιεί εγγενώς την ανάπτυξη εφαρμογών, καθώς δεν χρειάζεται να σκεφτούμε για την ακολουθία των μεταγωγέων, όπου οι κανόνες προώθησης πρέπει να εγκατασταθούν, αλλά μάλλον να δούμε το δίκτυο ως ένα απλό «μεγάλο switch». Τέτοιου είδους αφαιρέσεις απλοποιούν σημαντικά την ανάπτυξη και την εγκατάσταση των σύνθετων δικτυακών εφαρμογών, όπως οι προηγμένες υπηρεσίες που σχετίζονται με την ασφάλεια.

1.6.7 Programming Languages

Οι γλώσσες προγραμματισμού έχουν πολλαπλασιαστεί στο πέρασμα των δεκαετιών. Τόσο τα πανεπιστήμια όσο και η βιομηχανία έχουν συμμετάσχει στην εξέλιξη των χαμηλού επιπέδου γλωσσών μηχανής για συγκεκριμένα υλικά, όπως συναρμολόγηση για αρχιτεκτονικές x86, και των υψηλού επιπέδου ισχυρών γλωσσών προγραμματισμού, όπως η Java και Python. Οι εξελίξεις προς πιο φορητό και επαναχρησιμοποιήσιμο κώδικα έχουν οδηγήσει σε σημαντική στροφή τη βιομηχανία ηλεκτρονικών υπολογιστών.

Ομοίως, η δυνατότητα προγραμματισμού στα δίκτυα έχει αρχίσει να μετακινείται από τις γλώσσες μηχανής χαμηλού επιπέδου, όπως OpenFlow (αντίστοιχη της «assembly») στις γλώσσες προγραμματισμού υψηλού επιπέδου. Γλώσσες μηχανής όπως οι OpenFlow και POF μιμούνται τη συμπεριφορά των συσκευών προώθησης, αναγκάζοντας τους προγραμματιστές να ξοδεύουν πάρα πολύ χρόνο σε χαμηλού επιπέδου λεπτομέρειες αντί να λύσουν το πρόβλημα.

Πρώτα τα προγράμματα OpenFlow πρέπει να ασχοληθούν με τις λεπτομέρειες της συμπεριφοράς του υλικού, όπως η επικάλυψη των κανόνων, η προτεραιότητα των κανόνων και για τους κανόνες-ροής που εγκαθίστανται στο επίπεδο δεδομένων. Η χρήση των γλωσσών χαμηλού επιπέδου καθιστά δύσκολη την επαναχρησιμοποίηση λογισμικού και οδηγεί σε μια διαδικασία περισσότερο επιρρεπή σε λάθη ανάπτυξης.

Οι αφαιρέσεις που παρέχονται από τις γλώσσες προγραμματισμού υψηλού επιπέδου μπορεί να βοηθήσουν σημαντικά στην αντιμετώπιση πολλών προκλήσεων. Στο SDN μπορούν να σχεδιαστούν και να χρησιμοποιηθούν γλώσσες προγραμματισμού υψηλού επιπέδου για:

1) Να δημιουργηθεί υψηλότερο επίπεδο αφαίρεσης για την απλούστευση των διεργασιών προγραμματισμού των συσκευών προώθησης.
2) Να επιτρέπει πιο παραγωγικά και εστιασμένα στο πρόβλημα περιβάλλοντα για τους προγραμματιστές λογισμικού του δικτύου.
3) Να προωθήσει αρθρωτό λογισμικό και επαναχρησιμοποίηση του κώδικα στο επίπεδο ελέγχου του δικτύου.

4) Τη συμβολή στην ανάπτυξη του virtualization για το δίκτυο.

1.6.8 Network Applications

Οι εφαρμογές των δικτύων μπορούν να θεωρηθούν ως το «μυαλό του δικτύου». Μπορούν να εφαρμόσουν τη λογική ελέγχου που θα μεταφράζονται σε εντολές για να εγκατασταθούν στο επίπεδο δεδομένων, καθορίζοντας την συμπεριφορά των συσκευών προώθησης.

Ως παράδειγμα μπορούμε να δούμε μία απλή εφαρμογή δρομολόγησης. Η λογική της εφαρμογής αυτής είναι να ορίσει τη διαδρομή μέσω της οποίας τα πακέτα θα εισρέουν από το σημείο Α στο σημείο Β. Για την επίτευξη αυτού του στόχου, η αίτηση δρομολόγησης βασίζεται στην τοπολογία για να αποφασίσει σχετικά με την πορεία που θα χρησιμοποιηθεί και να δώσει εντολή στον ελεγκτή να εγκαταστήσει τους αντίστοιχους κανόνες προώθησης σε κάθε συσκευή προώθησης με την επιλεγμένη διαδρομή, από το Α στο Β.

Το SDN μπορεί να αναπτυχθεί σε οποιοδήποτε παραδοσιακό περιβάλλον δικτύου, από το σπίτι μέχρι τα επιχειρησιακά δίκτυα των Data Centers και τα σημεία ανταλλαγής Διαδικτύου (Internet exchange points). Μία τέτοια ποικιλία περιβαλλόντων οδήγησε σε ένα ευρύ φάσμα εφαρμογών δικτύου. Οι υπάρχουσες εφαρμογές δικτύου επιτελούν παραδοσιακές λειτουργίες, όπως η δρομολόγηση, η εξισορρόπηση φορτίου (Load Balancing), οι πολιτικές ασφαλείας, αλλά και να διερευνηθούν νέες προσεγγίσεις, όπως η μείωση κατανάλωσης ενέργειας (ενεργοαποδοτικά δίκτυα). Άλλα παραδείγματα περιλαμβάνουν σενάρια ανάκτησης από βλάβη (fail-over), αξιοπιστία λειτουργιών στο επίπεδο των δεδομένων, end-to-end QoS, virtualization του δικτύου και διαχείριση της κινητικότητας σε ασύρματα δίκτυα. Η ποικιλία των εφαρμογών δικτύου σε συνδυασμό με την ανάπτυξη πραγματικών περιπτώσεων χρήσης, αναμένονται να είναι από τις μεγάλες δυνάμεις για την προώθηση και την ευρεία υιοθέτηση των SDNs. Παρά τη μεγάλη ποικιλία των περιπτώσεων χρήσης, οι περισσότερες εφαρμογές SDN μπορούν να ομαδοποιηθούν σε μία από τις έξι κατηγορίες:

1. Μηχανισμός δικτυακής κίνησης
2. Κινητικότητα χρηστών και Ασύρματη σύνδεση
3. Μετρήσεις και παρακολούθηση δικτύου
4. Ασφάλεια και αξιοπιστία δικτύου
5. Δικτύωση Κέντρου Δεδομένων
6. Κατασκευή online καταστημάτων που προσφέρουν έτοιμες εφαρμογές SDN

1.7 Εξέλιξη της τεχνολογίας (State of the Art)

Το SDN είναι μία πολλά υποσχόμενη προσέγγιση που μπορεί να ξεπεράσει τις προκλήσεις που αντιμετωπίζουν οι υπηρεσίες cloud computing και συγκεκριμένα οι υπηρεσίες των δικτύων Κέντρων Δεδομένων (data-center networks ή DCNs) και του Δικτύου ως υπηρεσία (Network as a Service ή NaaS) [8]. Ως εκ τούτου, το ακόλουθο τμήμα υπογραμμίζει τη σημασία του SDN σε αυτές τις υπηρεσίες και περιγράφει τις διάφορες εφαρμογές του.

1.7.1 Data-Center Δίκτυα

Η κλίμακα και η πολυπλοκότητα του κέντρου των δικτύων δεδομένων (DCNS) προσεγγίζει το όριο των παραδοσιακών εξοπλισμών δικτύωσης και λειτουργιών IT. Επί του παρόντος, η υποδομή του κέντρου των δικτύων δεδομένων υφίσταται τεράστιες και ραγδαίες αλλαγές. Το Enterprise Strategy Group (ESG) [8] έχει καθορίσει τους λόγους που προκάλεσαν αυτές τις αλλαγές και τις συνοψίζει ως εξής:

- **Επιθετική συμμαχία στον τομέα της στατιστικής έρευνας Data Centers**. Το ESG δείχνει ότι το 63% των επιχειρήσεων που συμμετείχαν σε δημοσκόπηση σχεδιάζουν τη συγχώνευση των κέντρων δεδομένων τους.

- **Προοδευτική χρήση της τεχνολογίας virtualization**. Μεγάλες επιχειρήσεις, όπως οι Citrix, Microsoft και VMware αναπτύσσουν τεχνολογίες για virtualization. Επιπλέον, άλλοι οργανισμοί είναι πρόθυμοι να εισάγουν νέες πρωτοβουλίες για τις υποδομές τους που χρησιμοποιούν τις έννοιες της τεχνολογίας virtualization. Κατά συνέπεια, απαιτείται συμπαγής ολοκλήρωση μεταξύ της φυσικής υποδομής, των virtual servers και των δικτύων.

- **Η ανάπτυξη των Web εφαρμογών χρησιμοποιείται ευρέως σε πολλούς οργανισμούς**. Επιπλέον, αυτές οι εφαρμογές χρησιμοποιούν server-to-server επικοινωνία, επειδή βασίζονται σε server βαθμίδας x86 αρχιτεκτονικής οριζόντιας κλιμάκωσης. Ως εκ τούτου, τα κέντρα δεδομένων πρέπει να συγκρατήσουν την αύξηση της εσωτερικής κυκλοφορίας λόγω της μαζικής ανάπτυξης τέτοιου τύπου εφαρμογών. Επειδή, η δυναμική κλιμάκωση των δεδομένων δικτύων βασίζεται σε στατικές συσκευές δικτύου (Ethernet και συνδέσεις πακέτων IP), οι ομάδες IT αντιμετώπισαν ένα κενό ασυνέχειας κατά την εφαρμογή των επεκτάσιμων δεδομένων δικτύου.

Το ESG περιγράφει τις κύριες προκλήσεις του δικτύου ως εξής:

- **Δίκτυο Τμηματοποίησης και Ασφάλειας**. Σήμερα, η τμηματοποίηση DCN βασίζεται σε ένα συνδυασμό των VLANs, υποδικτύων IP, λίστες ελέγχου πρόσβασης των συσκευών (ACL) και σε κανόνες προστασίας που έχουν διατηρηθεί για χρόνια.

- **Μηχανική κίνηση**. Κάθε κυκλοφοριακή συμφόρηση ή αστοχία υλικού θα επηρεάσει την απόδοση και την καθυστέρηση όλων των άλλων συσκευών, λόγω της κίνησης του δικτύου που ακολουθεί σταθερή πορεία. Επιπλέον, η ανάπτυξη των VMs και των εικονικών servers σε πρόσφατη DCNS προσθέτει μια συμπληρωματική επιβάρυνση για τις επιδόσεις του δικτύου.

- **Network Provisioning και διαμόρφωσης**. Αν και οι virtual servers που προβλέπονται από τα εργαλεία cloud ενορχήστρωσης, οι πολιτικές των

δεδομένων του κέντρου εξοπλισμού και του ελέγχου πρέπει να συσταθούν με βάση τη συσκευή-προς-συσκευή ή τη ροή-σε-ροή, και θα πρέπει να διαχειρίζονται από ετερογενή δίκτυα με πολλαπλά συστήματα διαχείρισης. Ακόμα κι αν το λογισμικό διαχείρισης του δικτύου μπορεί να βοηθήσει σε αυτό το στάδιο, οι αλλαγές διαμόρφωσης του δικτύου παραμένουν μία επίπονη διαδικασία που θα πρέπει να γίνουν στο επίπεδο συνδέσμων του δικτύου.

Τελικά, η ασυνέχεια DCN θα είναι μια απειλή για τις επιχειρηματικές δραστηριότητες, διότι μπορεί να προκαλέσει υποβάθμιση του επιπέδου των υπηρεσιών της, καθυστερήσεις επιχειρηματικών πρωτοβουλιών καθώς και αύξηση του κόστους λειτουργίας της. Παρά το γεγονός ότι οι πωλητές δικτύωσης έχουν ξεκινήσει κάποιες καινοτομίες, όπως οικοδόμηση δικτύων και αρχιτεκτονικές σύγκλισης διορθώνοντας τα προβλήματα στην υποδομή DCN, δεν δημιουργούν προβλήματα σε ετερογενή δίκτυα. Παρ 'όλα αυτά, το πρότυπο του SDN είναι μια λύση για την επίλυση αυτών των προκλήσεων στις ρυθμίσεις DCN.

1.7.2 Δίκτυο ως υπηρεσία

Η αρχιτεκτονική προσανατολισμένη στις υπηρεσίες (Service Oriented Architecture ή SOA), είναι η ιδέα της οικοδόμησης ενός λογισμικού συστήματος που βασίζεται σε πολλαπλές ολοκληρωμένες λογικές μονάδες. Αυτές οι μονάδες είναι γνωστές ως υπηρεσίες και επιτρέπουν την καλύτερη κατασκευή και –διαχείριση λύσεων, για την επίλυση μεγάλων προβλημάτων, σε διάφορους τομείς. Η αρχιτεκτονική εξαρτάται από τις υπηρεσίες που χρησιμοποιούνται από την οντότητα υπηρεσίας του χρήστη.

1.7.3 SDN προκλήσεις και υπάρχουσες λύσεις

Παρά το γεγονός ότι το SDN είναι μια λύση για τους παρόχους και τις επιχειρήσεις, αντιμετωπίζει ορισμένες προκλήσεις που θα μπορούσαν να επηρεάσουν την απόδοσή του και την εφαρμογή του στα ασύρματα δίκτυα. Μία λίστα SDN προκλήσεων και μερικές από τις υπάρχουσες λύσεις τους αναφέρονται παρακάτω.

1.7.3.1 Αξιοπιστία

Ο ελεγκτής SDN πρέπει να ρυθμίζεται και να επικυρώνει τοπολογίες δικτύου για την πρόληψη της χρήσης των σφαλμάτων και για την αύξηση της διαθεσιμότητας του δικτύου. Ωστόσο, αυτή η νοημοσύνη μπορεί να ανασταλεί εξαιτίας του προβλήματος brain-split [8][αναφορά] που καθιστά τον ελεγκτή ένα σημείο αποτυχίας.

Στα παραδοσιακά δίκτυα, όταν μία ή περισσότερες συσκευές δικτύου αποτύχει, η κίνηση στο δίκτυο δρομολογείται είτε μέσω εναλλακτικών ροών είτε κοντά σε κόμβους ή συσκευές για τη διατήρηση της συνεχούς ροής κίνησης. Ωστόσο, στην αρχιτεκτονική κεντρικού ελεγκτή (SDN) και στην απουσία ενός εν’ αναμονή (stand-by) ελεγκτή, μόνο ένας κεντρικός ελεγκτής είναι υπεύθυνος για το σύνολο του δικτύου. Αν αυτός ο ελεγκτής αποτύχει, τότε το σύνολο του δικτύου μπορεί να

καταρρεύσει. Για την αντιμετώπιση αυτής της πρόκλησης, οι διαχειριστές του δικτύου θα πρέπει να επικεντρωθούν στην αξιοποίηση των κύριων λειτουργιών του ελεγκτή που μπορεί να αυξήσει την αξιοπιστία του δικτύου. Σε περίπτωση αποτυχίας της διαδρομής, ο ελεγκτής SDN θα πρέπει να έχει τη δυνατότητα να υποστηρίξει λύσεις πολλαπλών διαδρομών ή γρήγορης κίνησης της δρομολόγησής τους σε ενεργές συνδέσεις.

Εάν ο ελεγκτής υποστηρίζει τεχνολογίες, όπως Virtual Router Redundancy Protocol (VRRP) και Multi-Chassis Link Aggregation Group (MC-LAG), αυτά θα μπορούσαν να συμβάλουν στην αύξηση της διαθεσιμότητας του δικτύου. Σε περίπτωση αποτυχίας του ελεγκτή, είναι σημαντικό ότι ο ελεγκτής μπορεί να επιτρέψει την ομαδοποίηση δύο ή περισσότερων ελεγκτών SDN σε ενεργή κατάσταση αναμονής. Ωστόσο, πρέπει να διατηρηθεί ο συγχρονισμός μνήμης μεταξύ των ενεργών και των stand-by ελεγκτών.

Σε περίπτωση αποτυχίας του ελεγκτή η κεντρική αρχιτεκτονική θα διακόψει τις αιτήσεις κίνησης και τη ροή στο δίκτυο. Συγκεκριμένα, προτείνεται μια κατανεμημένη αρχιτεκτονική, SiBF (Switching with in-packet Bloom filters) [9], η οποία αποτελείται από πολλούς διαχειριστές τοποθετημένοι σε συστοιχία (RMS), που ενεργούν ως ελεγκτές. Κατά συνέπεια, όταν ο κεντρικός ελεγκτής αποτύχει, οι αιτήσεις ροής γίνονται από έναν άλλο stand-by ελεγκτή (RM) έως ότου ο κεντρικός ελεγκτής επανέλθει. Σε περίπτωση αποτυχίας ενός μεταγωγέα, η SiBF εγκαθιστά νέες χαρτογραφήσεις (νέες καταχωρήσεις back-up ροών) στους μεταγωγείς για κάθε ενεργή είσοδο. Τα πακέτα θα πρέπει να κατευθύνονται προς τους προορισμούς τους σχετικά με τις εναλλακτικές διαδρομές που υποδεικνύονται από τις back-up καταχωρήσεις.

Η ολοκλήρωση μεταξύ των ελεύθερων πολλαπλών διαδρομών (multipathing) και ο έλεγχος συμφόρησης βασίζεται σε μια δυναμική προσέγγιση multipathing εξισορρόπησης φόρτου που τρέχει ένα κατανεμημένο αλγόριθμο σε περίπτωση αποτυχίας του ελεγκτή. Ο αλγόριθμος ενημερώνει τους μεταγωγείς με τυχόν αλλαγές στο «φορτίο μονοπατιού» για τις συναφείς διαδρομές σε περιπτώσεις κυκλοφοριακής συμφόρησης και της ανισορροπίας φορτίου.

1.7.3.2 Επεκτασιμότητα

Η αποσύνδεση μεταξύ των επιπέδων των δεδομένων και του ελέγχου διακρίνει τα δίκτυα SDN από ένα παραδοσιακό δίκτυο. Στο SDN, τα δύο επίπεδα μπορούν να εξελιχθούν ανεξάρτητα και αυτή η συγκεντρωτική άποψη του δικτύου επιταχύνει τις αλλαγές στο επίπεδο του ελέγχου. Ωστόσο, η αποσύνδεση έχει και τα δικά της μειονεκτήματα.

Εκτός από την πολυπλοκότητα προσδιορισμού ενός προτύπου API μεταξύ των δύο επιπέδων, ενδέχεται να προκύψουν και περιορισμοί επεκτασιμότητας. Έτσι, καταλήγουμε στο συμπέρασμα ότι «όταν το δίκτυο κλιμακώνεται μέχρι τον μέγιστο αριθμό μεταγωγέων και τον μέγιστο αριθμό των τελικών χρηστών, ο ελεγκτής SDN μπορεί να γίνει ένα βασικό εμπόδιο».

Δεδομένου ότι το εύρος ζώνης και ο αριθμός των μεταγωγέων και των ροών αυξάνεται, περισσότερες αιτήσεις θα πρέπει να βρίσκονται στην ουρά στον ελεγκτή,

ο οποίος μπορεί να μην είναι σε θέση να τα χειριστεί όλα. Μελέτες σε έναν ελεγκτή SDN (NOX) έχουν δείξει ότι μπορεί να χειριστεί 30.000 αιτήματα / δευτερόλεπτο. Αυτό μπορεί να είναι επαρκές για τις επιχειρήσεις και για τα πανεπιστημιακά δίκτυα, αλλά είναι μια δυσχέρεια για τα δίκτυα των κέντρων δεδομένων με υψηλές ταχύτητες ροής. Επιπλέον, εκτιμάται ότι ένα μεγάλο κέντρο δεδομένων που αποτελείται από 2 εκατομμύρια εικονικές μηχανές μπορεί να δημιουργήσουν 20 εκατομμύρια ροές ανά δευτερόλεπτο. Ωστόσο, οι τρέχοντες ελεγκτές μπορούν να υποστηρίξουν περίπου 10^5 ροές ανά δευτερόλεπτο στη βέλτιστη περίπτωση. Εκτός από την υπερφόρτωση του ελεγκτή, η διαδικασία εγκατάστασης ροής μπορεί να επιβάλει περιορισμούς στην επεκτασιμότητα του δικτύου. Η ρύθμιση της ροής αποτελείται από τέσσερα βήματα:

1. Ένα πακέτο φτάνει σε ένα μεταγωγέα και δεν ταιριάζει με καμία εγγραφή ροής.
2. Ο μεταγωγέας στέλνει ένα αίτημα στον ελεγκτή για να πάρει οδηγίες για το πώς θα προωθήσει το πακέτο.
3. Ο ελεγκτής στέλνει μια νέα καταχώρηση ροής με νέους κανόνες προώθησης πίσω στο μεταγωγέα.
4. Ο μεταγωγέας ενημερώνει τις εγγραφές στον πίνακα ροής.

Η απόδοση της διαδικασίας εγκατάστασης ροής εξαρτάται από τους πόρους του μεταγωγέα (CPU, μνήμη, κλπ) και την απόδοση του λογισμικού του ελεγκτή. Ο χρόνος ενημέρωσης της βάσης πληροφοριών προώθησης του μεταγωγέα (FIB) εισάγει καθυστέρηση στη δημιουργία κάθε νέας ροής. Πρόσφατα σημεία αναφοράς σε ελεγκτές SDN και μεταγωγείς έδειξαν ότι ο ελεγκτής θα μπορούσε να ανταποκριθεί σε ένα αίτημα εγκατάστασης ροής μέσα σε ένα χιλιοστό του δευτερολέπτου, ενώ οι μεταγωγείς υλικού θα μπορούσαν να υποστηρίξουν μερικές χιλιάδες εγκαταστάσεις ανά δευτερόλεπτο με μια καθυστέρηση κάτω των 10 δευτερολέπτων στην καλύτερη περίπτωση. Καθυστερήσεις εγκατάστασης ροής μπορούν να αποτελέσουν μια πρόκληση στην επεκτασιμότητα της δικτύωσης. Επιπλέον, το δίκτυο μεταδίδει παραπάνω πληροφορίες και η διάδοση της ροής καταχωρήσεων στους πίνακες περιορίζουν την επεκτασιμότητα των δικτύων SDN. Η πλατφόρμα SDN μπορεί να προκαλέσει περιορισμένη ορατότητα της κίνησης του δικτύου, καθιστώντας σχεδόν αδύνατη την αντιμετώπιση προβλημάτων. Πριν από τα SDN, μια ομάδα διαχειριστών του δικτύου θα μπορούσε γρήγορα να εντοπίσει, για παράδειγμα, ότι ένα αντίγραφο ασφαλείας του δικτύου επιβραδύνει την ταχύτητα του δικτύου. Η λύση θα ήταν τότε να αναδιαρθρώσει το αντίγραφο ασφαλείας σε έναν χρόνο λιγότερο πολυσύχναστο.

Δυστυχώς, στα SDN, είναι ορατή μόνο η πηγή σήραγγας (tunnel) και το τελικό σημείο του tunnel με τύπο κυκλοφορίας User Datagram Protocol (UDP), αλλά κυρίως, δεν μπορεί κανείς να δει ποιος χρησιμοποιεί το tunnel. Δεν υπάρχει κανένας τρόπος για να διαπιστωθεί αν το πρόβλημα είναι η διαδικασία αντιγραφής, το σύστημα ηλεκτρονικού ταχυδρομείου ή κάτι άλλο. Για παράδειγμα μία επικοινωνία ομιλίας είναι θωρακισμένη από άποψη του UDP tunnel, πράγμα που σημαίνει πως όταν η κυκλοφορία επιβραδύνεται οι χρήστες διαμαρτύρονται και έτσι εντοπίζεται η περιοχή του προβλήματος στο δίκτυο. Με αυτή την απώλεια ορατότητας, η αντιμετώπιση προβλημάτων παρεμποδίζεται, οι περιορισμοί επεκτασιμότητας αναδύονται, και οι καθυστερήσεις στην ανάλυση μπορεί να γίνουν

ζημιογόνες για την επιχείρηση. Προκειμένου να ελαχιστοποιηθεί ο πολλαπλασιασμός των καταχωρήσεων ροής, ο ελεγκτής θα πρέπει να χρησιμοποιεί κεφαλίδες στο κεντρικό δίκτυο. Οι καταχωρήσεις ροής θα είναι είσοδοι και έξοδοι στους μεταγωγείς.

Η βελτιωμένη επεκτασιμότητα του δικτύου μπορεί επίσης να εξασφαλιστεί με τη χρήση VM και της εικονικής μετανάστευσης αποθήκευσης. Μία άλλη λύση για την επεκτασιμότητα που προτείνεται είναι η «DIFANE». Αυτή είναι μια κατανεμημένη αρχιτεκτονική ροής διαχείρισης που μπορεί να κλιμακωθεί μέχρι να ικανοποιηθούν οι απαιτήσεις (δηλ. μεγάλο αριθμό τελικών χρηστών, ροών, και κανόνων) των μεγάλων δικτύων.

Μια βιώσιμη λύση στις προκλήσεις της κλιμάκωσης προτείνεται με την «CORONET» αρχιτεκτονική SDN, η οποία είναι ανεκτική σε σφάλματα και επεκτάσιμη σε μεγάλα δίκτυα λόγω του μηχανισμού VLAN που μπορεί να εγκατασταθεί σε έναν τοπικό μεταγωγέα. Το CORONET έχει γρήγορη ανάκαμψη από το μεταγωγέα ή τη σύνδεση αποτυχίας, υποστηρίζει την επεκτασιμότητα δικτύων, χρησιμοποιεί εναλλακτικές τεχνικές δρομολόγησης πολλαπλών διαδρομών, λειτουργεί με οποιαδήποτε τοπολογία του δικτύου και χρησιμοποιεί έναν κεντρικό ελεγκτή για να προωθήσει τα πακέτα. Αποτελείται από μονάδες υπεύθυνες για την ανακάλυψη της τοπολογίας, το σχεδιασμό της διαδρομής, την εκχώρηση της κυκλοφορίας, και τον υπολογισμό της διαδρομής συντομότερης διαδρομής (αλγόριθμος Dijkstra). Το κύριο χαρακτηριστικό του CORONET είναι η χρήση των VLANs, η οποία μπορεί να απλοποιήσει την προώθηση των πακέτων, να ελαχιστοποιήσει τον αριθμό των κανόνων ροών και να υποστηρίξει ιδιότητες κλιμάκωσης.

Σε μια άλλη λύση, ονομαζόμενη «DevoFlow», προβλέπει ότι οι μικρο-ροές διαχειρίζονται από το επίπεδο των δεδομένων και οι πιο μαζικές ροές από τον ελεγκτή, πράγμα που σημαίνει ότι το φορτίο του ελεγκτή θα μειωθεί και η επεκτασιμότητα του δικτύου θα μεγιστοποιηθεί. Αυτή η προσέγγιση ελαχιστοποιεί το κόστος του ελεγκτή για την ορατότητα που συνδέεται με κάθε ρύθμιση ροής και μειώνει την επίδραση των επιπλέων προγραμματιζόμενων ροών, ενισχύοντας έτσι την απόδοση του δικτύου και την επεκτασιμότητά του.

Τέλος, ακόμα ένα κλιμακούμενο πλαίσιο ελέγχου SDN, το McNettle, το οποίο εκτελείται σε πολυπύρηνους διακομιστές που χρησιμοποιούν κοινή μνήμη και βασίζονται στην τεχνολογία Nettle. Πειράματα έδειξαν ότι το McNettle θα μπορούσε να τροφοδοτήσει 5000 μεταγωγείς με έναν μόνο ελεγκτή με 46 πυρήνες και θα μπορούσε να χειριστεί 14 εκατομμύρια ροές ανά δευτερόλεπτο με καθυστέρηση κάτω από 200 μS για ελαφρά φορτία και 10 ms για φορτία που αποτελούνται μέχρι και 5000 μεταγωγείς.

Κεφάλαιο 2:
Η τεχνολογία OpenFlow

2.1 Εισαγωγή

Τα τελευταία χρόνια η τεχνολογία δικτύωσης έχει βελτιωθεί σε μεγάλη κλίμακα και με καινοτόμους μετασχηματισμούς όσον αφορά την ταχύτητα, την αξιοπιστία και την ασφάλεια του δικτύου [10]. Στο φυσικό στρώμα, οι συσκευές δικτύωσης έχουν εξελιχθεί παρέχοντας μεγάλη χωρητικότητα στις συνδέσεις, έχουν βελτιωθεί από άποψη υπολογιστικής δύναμης και ποικιλίας εφαρμογών, προσφέροντας εργαλεία για να παρακολουθούμε εύκολα τις λειτουργίες τους. Ωστόσο, το δίκτυο στη δομή του δεν έχει δει μεγάλη αλλαγή από τις αρχές της εμφάνισής του.

Στην υπάρχουσα υποδομή, οι εργασίες που αποτελούν τη γενική λειτουργία του δικτύου όπως η δρομολόγηση και η μεταγωγή πακέτων ή οι αποφάσεις πρόσβασης στο δίκτυο αντιμετωπίζονται από τις συσκευές δικτύων των διαφορετικών κατασκευαστών τρέχοντας απλώς ένα διαφορετικό λογισμικό τύπου «firmware». Αυτό δεν παρέχει αρκετό χώρο για νέες ερευνητικές ιδέες, όπως νέοι αλγόριθμοι δρομολόγησης που εξετάζονται στα ευρείας κλίμακα πραγματικά δίκτυα. Επιπλέον, οποιαδήποτε προσπάθεια για νέες πειραματικές ιδέες πέρα από το τρέχων δίκτυο μπορεί να καταλήξει σε κάποιο βαθμό σε αποτυχία του δικτύου, οδηγώντας την υποδομή των δικτύων να είναι στατική, άκαμπτη, και να μην έχει προσελκύσει σημαντικές καινοτομίες [11].

Το OpenFlow είναι μια προσέγγιση που αντιμετωπίζει το πρόβλημα αυτό, επειδή οι διαχειριστές δικτύων μπορούν να εφαρμόσουν και να ελέγξουν τα στοιχεία που θέλουν στο λογισμικό, χωρίς να περιμένουν τους κατασκευαστές να τα ενσωματώσουν σε μία νέα έκδοση του firmware. Επιπλέον, επιτρέπει στους κατασκευαστές να δώσουν πρόσβαση σε ερευνητές όσον αφορά στον εξοπλισμό τους με έναν ενοποιημένο τρόπο χωρίς το άνοιγμα των προϊόντων τους. Επομένως, οι ερευνητές είναι σε θέση να πραγματοποιήσουν πειράματα με νέα πρωτόκολλα σε ένα πραγματικό δίκτυο χωρίς να επηρεάζουν την κυκλοφορία του παραγωγικού δικτύου.

Το OpenFlow χρησιμοποιεί πίνακες-ροής (flow-tables) που είναι παρόμοιοι με τους πίνακες lookup (lookup tables), στους σύγχρονους Ethernet δρομολογητές και μεταγωγείς. Αυτοί οι πίνακες-ροής μπορούν να εφαρμόσουν firewalls, NAT, QoS ή να συλλέξουν στατιστικά για τη διαχείριση των δικτύων χωρίς να παρεμβάλλονται οι κατασκευαστές. Επίσης, οι πίνακες-ροής περιέχουν τους κανόνες αντιστοιχιών/δράσης (match/action rules) που μπορούν να δημιουργηθούν και να τροποποιηθούν από έναν κεντρικοποιημένο ελεγκτή (centralized controller). Ο ελεγκτής προσφέρει έναν προγραμματιστικό έλεγχο των ροών στον διαχειριστή του

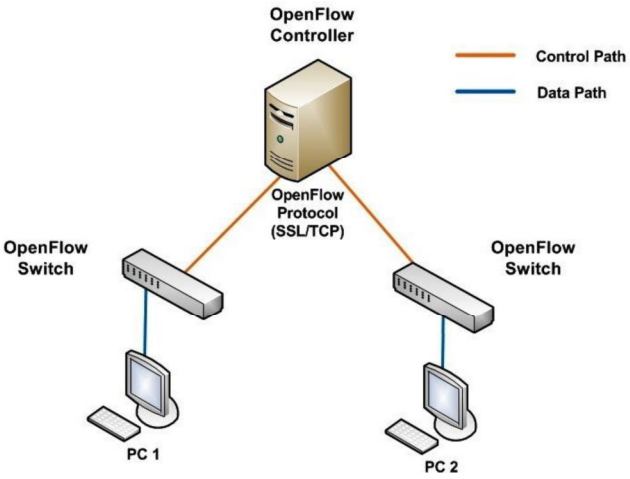
δικτύου ώστε να καθορίσει μια συγκεκριμένη διαδρομή από την πηγή στον προορισμό της χρησιμοποιώντας επεξεργασία βασισμένη στην ροή προώθησης των πακέτων. Μειώνει τις δαπάνες κατανάλωσης ισχύος και διαχείρισης με την εξάλειψη της επεξεργασίας των πακέτων στον δρομολογητή, αφού πλέον οι διαδρομές πακέτων καθορίζονται από τον κεντρικοποιημένο ελεγκτή.

2.2 Αρχιτεκτονική OpenFlow

Η δικτυακή αρχιτεκτονική του OpenFlow αποτελείται από τρεις βασικές έννοιες:

1. OpenFlow-συμβατούς μεταγωγείς που συνθέτουν το επίπεδο δεδομένων.
2. Το επίπεδο ελέγχου αποτελείται από έναν ή περισσότερους OpenFlow ελεγκτές.
3. Ένα ασφαλές κανάλι ελέγχου συνδέει τους μεταγωγείς με το επίπεδο ελέγχου.

Οι μεταγωγείς επικοινωνούν μεταξύ τους αλλά και με τους τελικούς χρήστες (hosts) χρησιμοποιώντας το λογισμικό διαδρομής δεδομένων (data path) που παρέχεται και ο ελεγκτής επικοινωνεί με τους μεταγωγείς χρησιμοποιώντας την διαδρομή ελέγχου όπως φαίνεται παρακάτω, Εικόνα 5.



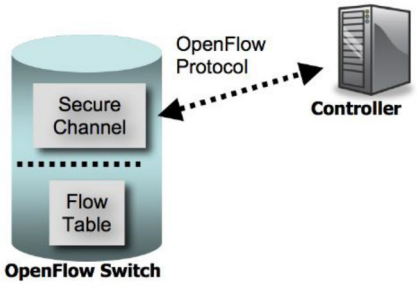
Εικόνα 5: Η δικτυακή αρχιτεκτονική OpenFlow

Η σύνδεση μεταξύ του ελεγκτή OpenFlow και του μεταγωγέα εξασφαλίζεται μέσω ενός Ασφαλούς Καναλιού (Secure Chanel) χρησιμοποιώντας κρυπτογραφικά πρωτόκολλα SSL ή TLS, όπου ο μεταγωγέας και ο ελεγκτής επικυρώνονται αμοιβαία με την ανταλλαγή πιστοποιητικών που υπογράφονται με ιδιωτικό κλειδί και από τις δύο πλευρές. Αν και αυτός είναι ένας πολύ ισχυρός αλγόριθμος ασφάλειας, ο ελεγκτής μπορεί να είναι τρωτός σε επιθέσεις άρνησης υπηρεσιών (DoS), ή σε επίθεση Man in the middle, επομένως πρέπει να εφαρμοστούν κατάλληλες πρακτικές ασφάλειας για να αποτραπούν τέτοιου είδους επιθέσεις.

2.3 OpenFlow Switch

Ένας μεταγωγέας OpenFlow αποτελείται από έναν πίνακα-ροής (flow table), ο οποίος εκτελεί διαδικασίες αναζήτησης και προώθησης στα πακέτα, και από ένα ασφαλές κανάλι με έναν εξωτερικό ελεγκτή, όπως φαίνεται στην Εικόνα 6. Ο ελεγκτής διαχειρίζεται τον μεταγωγέα μέσα από το ασφαλές κανάλι χρησιμοποιώντας το πρωτόκολλο OpenFlow. [12]

Ο πίνακας-ροής περιέχει ένα σύνολο ροών-εγγραφής (flow entries) που έχουν τις τιμές κεφαλίδων των πακέτων, τους μετρητές δραστηριότητας, και ένα σύνολο μηδενικών ή περισσότερων ισχύων ενεργειών για το ταίριασμα των πακέτων. Όλα τα πακέτα υποβάλλονται σε επεξεργασία από τον μεταγωγέα και συγκρίνονται με τον πίνακα-ροής. Εάν βρεθεί μία εγγραφή που ταιριάζει, τότε εκτελούνται στο πακέτο οποιεσδήποτε ενέργειες που ισχύουν για την συγκεκριμένη εγγραφή (π.χ. μία ενέργεια μπορεί να είναι η προώθηση του πακέτου σε μία συγκεκριμένη πόρτα εξόδου). Εάν δεν βρεθεί καμία εγγραφή που να ταιριάζει, τότε το πακέτο διαβιβάζεται στον ελεγκτή μέσα από το ασφαλές κανάλι. Ο ελεγκτής είναι υπεύθυνος να χειρίζεται τα πακέτα χωρίς έγκυρες ροές-εγγραφής και να διαχειρίζεται τον πίνακα-ροής του μεταγωγέα με την προσθήκη και την αφαίρεση ροών-εγγραφής.



Εικόνα 6: OpenFlow enabled switch OF v1.0 specification **[12]**

2.4 Flow Table

Ένας μεταγωγέας στο OpenFlow δίκτυο έχει έναν ή περισσότερους πίνακες-ροής που περιλαμβάνουν ένα σύνολο από εγγραφές αποτελούμενες από πεδία κεφαλίδας (Header Fields), μετρητές (Counters) και ενέργειες (Actions) όπως φαίνεται στον

Πίνακας **1**:

Header Fields	Counters	Actions

Πίνακας 1: Ροή-εγγραφής (flow entry)

όπου :

- Header Fields (πεδία κεφαλίδας) για το ταίριασμα σε σύγκριση με τα πακέτα
- Counters (μετρητές), που αυξάνονται κατά το ταίριασμα των πακέτων
- Actions (ενέργειες), που εφαρμόζονται αφού έχουμε ταίριασμα πακέτου

2.4.1 Header Fields

Τα πεδία κεφαλίδας (Header Fields) αποτελούνται από διάφορα πεδία όπως αναφέρονται στον παρακάτω

Πίνακας **2**.

Incoming switch port	IEEE 802.3 Ethernet source address	IEEE 802.3 Ethernet destination address	IEEE 802.3 Ethernet type	IEEE 802.1Q VLAN ID	IEEE 802.1Q VLAN priority	IP source address	IP destination address	IP proto field	IP Type Of Service (TOS) bits	TCP/UDP source ports	TCP/UDP destination ports

Πίνακας 2: Πεδία από τα πακέτα που χρησιμοποιούνται για το ταίριασμα ενάντια στις ροές-εγγραφής

Τα εισερχόμενα πακέτα μπορούν να αντιπαραβληθούν με διάφορα πεδία του πακέτου από τα στρώματα του μοντέλου OSI, που κυμαίνονται από τη σύνδεση δεδομένων (data link) έως στο επίπεδο μεταφοράς (transport layer), καθώς και στην εισερχόμενη θύρα του μεταγωγέα. Στη περίπτωση που θέλουμε να ταιριάξουν όλα τα πεδία του πακέτου, υπάρχει μία τιμή τύπου «ANY» που μπορεί να χρησιμοποιηθεί στον πίνακα-ροής.

2.4.2 Counters

Τα πρότυπα OpenFlow επιτρέπουν στον μεταγωγέα να εκθέσει στατιστικά στοιχεία μέσω μετρητών. Οι μετρητές αποτελούνται από πολλαπλές μεταβλητές ανά πίνακα (Table), ροή (Flow), πόρτα (Port), ουρά (Queue), όπως αναφέρονται στον παρακάτω

Πίνακας **3**.

per Table		**per Flow**		**per Port**		**per Queue**	
Counters	Bits	Counters	Bits	Counters	Bits	Counters	Bits
Active entries	32	Received packets	64	Received packets	64	Transmit packets	64
Packet Lookup	64	Received Bytes	64	Transmitted packets	64	Transmit bytes	64
Packet Matches	64	Duration (seconds)	32	Received Bytes	64	Transmit overrun errors	64

-	Duration (nano seconds)	32	Transmitted bytes	64	-
	-		Receive Drops	64	
			Transmit Drops	64	
			Receive Errors	64	
			Transmit Errors	64	
			Receive Frame Alignment Errors	64	
			Receive Overrun errors	64	
			Receive CRC Errors	64	
			Collisions	64	

Πίνακας 3: Λίστα από τους μετρητές που χρησιμοποιούνται για τα στατιστικά των μηνυμάτων **[12]**

2.4.3 Actions

Αν ένα πακέτο εισόδου ταιριάζει με ένα από τα πεδία ταιριάσματος στον πίνακα-ροής, τότε στο συγκεκριμένο πακέτο θα εφαρμοστεί η ενέργεια που περιγράφεται στο πεδίο «Actions» του πίνακα-ροής. Ο μεταγωγέας OpenFlow υποστηρίζει τη προώθηση του πακέτου σε μία φυσική πόρτα. Επιπλέον, υπάρχουν εικονικές πόρτες που καθορίζονται από την τυποποίηση του OpenFlow ως ειδικοί στόχοι που τα πακέτα μπορούν να προωθηθούν.

Οι ενέργειες διαχωρίζονται σε «υποχρεωτικές» (required) και «προαιρετικές» (optional). Οι «required» ενέργειες πρέπει να υποστηρίζονται από όλους τους μεταγωγείς που είναι συμβατοί με το OpenFlow και οι «optional» ενέργειες οι οποίες έχουν αποδειχθεί ότι είναι χρήσιμες δεν είναι απαραίτητο να υποστηρίζονται τους μεταγωγείς.

Υποχρεωτική ενέργεια προώθησης «required»:

- **ALL**: Στέλνει το πακέτο σε όλες τις διεπαφές, χωρίς να περιλαμβάνει την εισερχόμενη διεπαφή.
- **CONTROLLER**: Ενσωματώνει και στέλνει το πακέτο στον ελεγκτή.
- **LOCAL**: Στέλνει το πακέτο στους μεταγωγείς τις τοπικής στοίβας δικτύωσης.
- **TABLE**: Υλοποιεί τις ενέργειες στον πίνακα-ροής. Μόνο για μηνύματα «packet-out».
- **IN PORT**: Στέλνει το πακέτο στη εισερχόμενη πόρτα.

Προαιρετική ενέργεια προώθησης «optional»:

- **Normal**: Προωθεί το πακέτο με τις παραδοσιακές μεθόδους προώθησης, π.χ. L2, VLAN, και L3 επεξεργασία.
- **Flood**: Πλημμυρίζει το πακέτο κατά μήκος του ελάχιστου spanning tree, χωρίς να περιλαμβάνει την εισερχόμενη διεπαφή.

Εκτός από την ενέργεια της προώθησης σε ένα πίνακα-ροής υπάρχουν και οι παρακάτω ενέργειες [10]:

- **DROP**: Μια «required» ενέργεια που υποδεικνύεται από μια κενή λίστα ενεργειών. Όλα τα πακέτα που ταιριάζουν σε μια κενή λίστα ενεργειών απορρίπτονται.
- **Enqueue**: Είναι «optional» ενέργεια και μπορεί να χρησιμοποιηθεί για να θέσει τα πακέτα σε μια ουρά η οποία συνδέεται σε μία πόρτα για την παροχή Quality of service (QoS).
- **Modify-field**: Είναι «optional» ενέργεια χρησιμοποιείται για να τροποποιήσει ένα ειδικό πεδίο κεφαλίδας για το εισερχόμενο πακέτο, όπως:
    - Ορισμός VLAN ID και προτεραιοτήτων.
    - Αφαίρεση VLAN κεφαλίδας.
    - Τροποποίηση Ethernet MAC διεύθυνσης πηγής και προορισμού.
    - Τροποποίηση IP διεύθυνσης πηγής και προορισμού.
    - Τροποποίηση IP TOS bits
    - Τροποποίηση θυρών πηγής και προορισμού του επιπέδου Μεταφοράς (transport layer).

2.5 Επικοινωνίες

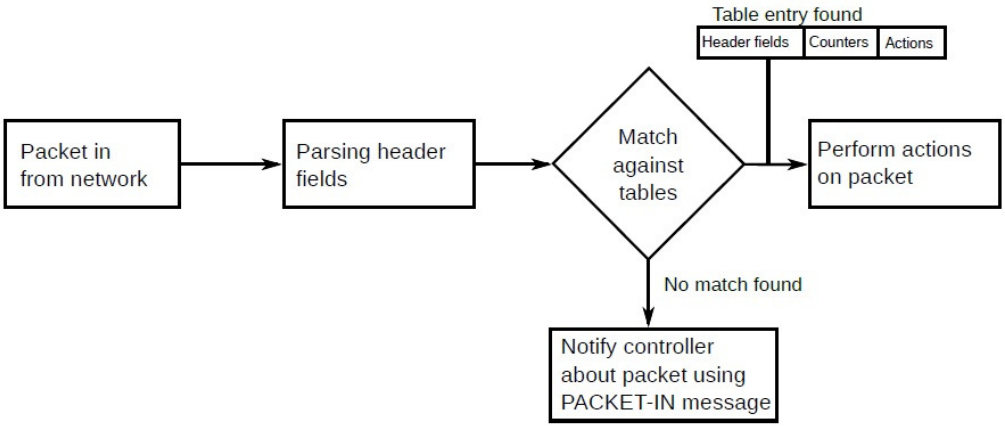
Υπάρχουν τρεις κατηγορίες επικοινωνίας στο πρωτόκολλο OpenFlow [12]:

- **Controller-to-Switch**: η επικοινωνία αυτή είναι υπεύθυνη για την ανίχνευση χαρακτηριστικών, ρυθμίσεων, προγραμματισμό του μεταγωγέα και ανάκτηση πληροφοριών (τύποι μηνυμάτων: Features, Configuration, Modify-State, Read-State, Send-Packet, Barrier).
- **Asynchronous**: επικοινωνία με πρωτοβουλία του συμβατού OpenFlow μεταγωγέα χωρίς καμία παρακίνηση από τον ελεγκτή. Χρησιμοποιείται για να ενημερώνει τον ελεγκτή για αφίξεις πακέτων, αλλαγές κατάστασης του μεταγωγέα και λάθη (τύποι μηνυμάτων: Packet-in, Flow-Removed, Port-status, Error).
- **Symmetric**: επικοινωνία για να δούμε αν το κανάλι ελέγχου είναι ενεργό και διαθέσιμο (τύποι μηνυμάτων: Hello, Echo, Vendor).

2.6 Μηχανισμός Προώθησης Πακέτων

Σε ένα OpenFlow δίκτυο όταν ο μεταγωγέας λαμβάνει ένα πακέτο, αναλύει το πεδίο κεφαλίδας και ελέγχει αν ταιριάζει στους κανόνες του πίνακα-ροής. Αν υπάρχει ένα ταίριασμα, τότε υλοποιείται η ενέργεια από τον πίνακα-ροής. Αν τα πακέτα ταιριάζουν σε περισσότερους από έναν κανόνες, τότε τα πακέτα αντιπαραβάλλονται με μια συγκεκριμένη εγγραφή-ροής με βάση ιεραρχημένες

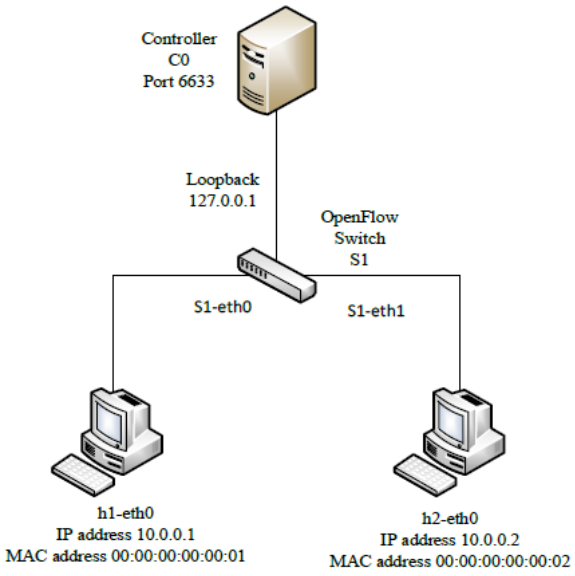
προτεραιότητες, δηλαδή επιλέγεται η εγγραφή-ροής με την υψηλότερη προτεραιότητα. Στη συνέχεια, ο μεταγωγέας ενημερώνει τους μετρητές του εν λόγω πίνακα-ροής. Τέλος, ο μεταγωγέας προωθεί το πακέτο σε μια πόρτα εξόδου. Αν το εισερχόμενο πακέτο δεν ταιριάζει με καμία εγγραφή-ροής στον πίνακα-ροής, ο μεταγωγέας θα προωθήσει το πακέτο στον ελεγκτή για να υπολογίσει την λογική που θα πρέπει να υλοποιηθεί στο πακέτο και στα παρόμοια μελλοντικά πακέτα. Η διαδικασία μηχανισμού της προώθηση των πακέτων απεικονίζεται στο διάγραμμα ροής της Εικόνα 7.



Εικόνα 7: Μηχανισμός Προώθησης Πακέτων του OpenFlow **[13]**

2.7 Επίδειξη μηνυμάτων που ανταλλάσσονται στο OpenFlow δίκτυο

Για να δείξουμε τα μηνύματα που ανταλλάσσονται σε ένα πραγματικό δίκτυο OpenFlow [10], μπορούμε να χρησιμοποιήσουμε τον δικτυακό εξομοιωτή Mininet [14]. Με αυτόν θα εξομοιώσουμε δύο τελικούς χρήστες (hosts) συνδεδεμένους με ένα μεταγωγέα που συνδέεται σε έναν ελεγκτή, όπως φαίνεται και στην Εικόνα 8.

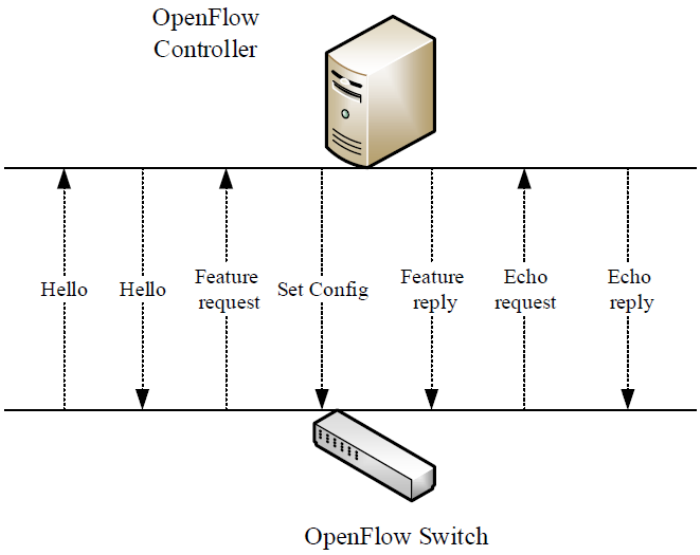


Εικόνα 8: Δικτυακή τοπολογία από το Mininet

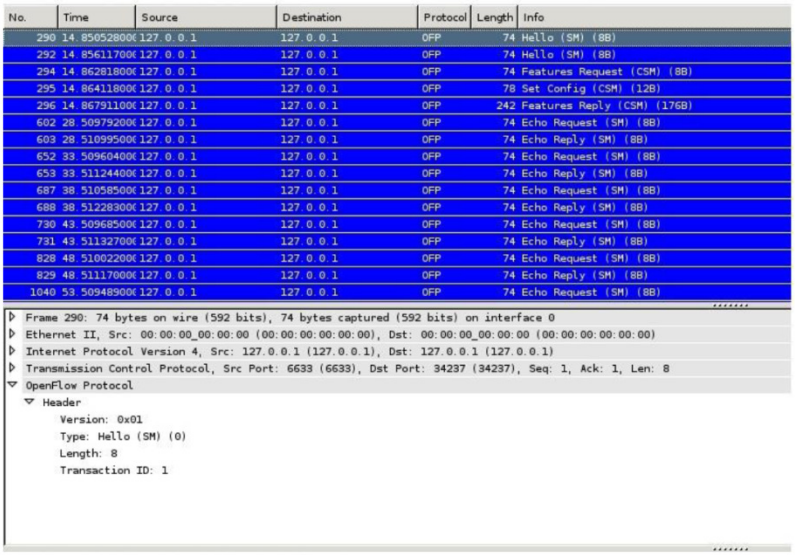
Γι' αυτή την επίδειξη πρέπει να εξηγήσουμε την εγκατάσταση της σύνδεσης μεταξύ Μεταγωγέα - Ελεγκτή, και την επικοινωνία host-to-host μέσω του μεταγωγέα και ελεγκτή OpenFlow.

2.7.1 Δημιουργία μηνυμάτων μεταξύ μεταγωγέα και ελεγκτή

Όταν ένας μεταγωγέας συνδέεται σε ένα δίκτυο OpenFlow εγκαθιδρύει μια σύνδεση TCP με τη διεύθυνση IP του ελεγκτή (Loopback interface 127.0.0.1), σε μια προεπιλεγμένη πόρτα με αριθμό 6633. Μετά τη διαδικασία αυτή οι δύο πλευρές αρχίζουν να ανταλλάσσουν μηνύματα Hello, που περιλαμβάνουν τον μεγαλύτερο αριθμό έκδοσης του OpenFlow που υποστηρίζουν. Ακολουθεί το μήνυμα Feature request το οποίο αποστέλλεται από τον ελεγκτή για να μάθει ποιες πόρτες είναι διαθέσιμες στον μεταγωγέα, ο οποίος με τη σειρά του απαντά με μήνυμα Feature reply που περιέχει μια λίστα με τις πόρτες, την ταχύτητα των πορτών, και τους υποστηριζόμενους πίνακες και ενέργειες. Το μήνυμα SET config αποστέλλεται στη συνέχεια από τον ελεγκτή στον μεταγωγέα για να ρωτήσει αν θα λήξει τη ροή. Τέλος, μηνύματα echo request και echo reply αποστέλλονται συχνά μεταξύ του ελεγκτή και του μεταγωγέα για να ανταλλάξουν πληροφορίες σχετικά με το εύρος ζώνης, τις καθυστερήσεις και για να κρατάνε «ζωντανή» τη σύνδεσή τους. Η διαδικασία αυτή απεικονίζεται παρακάτω στην Εικόνα 9 και στην Εικόνα 10.



Εικόνα 9: Μηνύματα επικοινωνίας μεταξύ
του OpenFlow μεταγωγέα και του OpenFlow ελεγκτή



Εικόνα 10: Απεικόνιση της επικοινωνίας μεταξύ OpenFlow μεταγωγέα και ελεγκτή

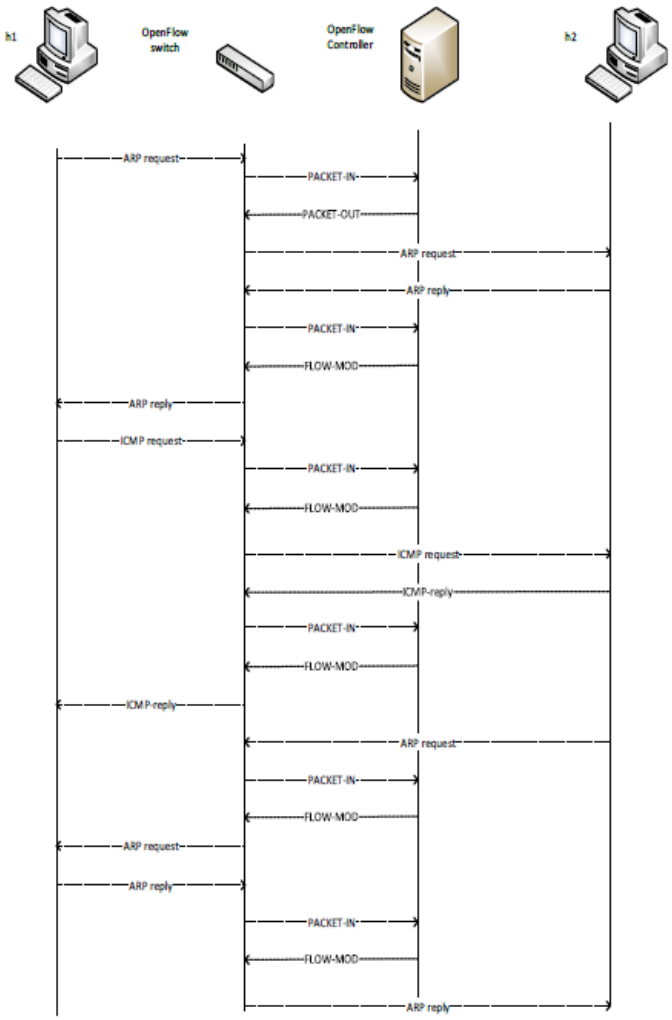
2.7.2 Μηνύματα που ανταλλάσσονται μεταξύ δύο Hosts

Για να δείξουμε πως γίνεται η σύνδεση host-to-host σε ένα OpenFlow δίκτυο, θα χρησιμοποιήσουμε το εργαλείο του Ping για να στείλουμε ICMP πακέτα από τον host1 (h1) στον host2 (h2) και το αντίστροφο.

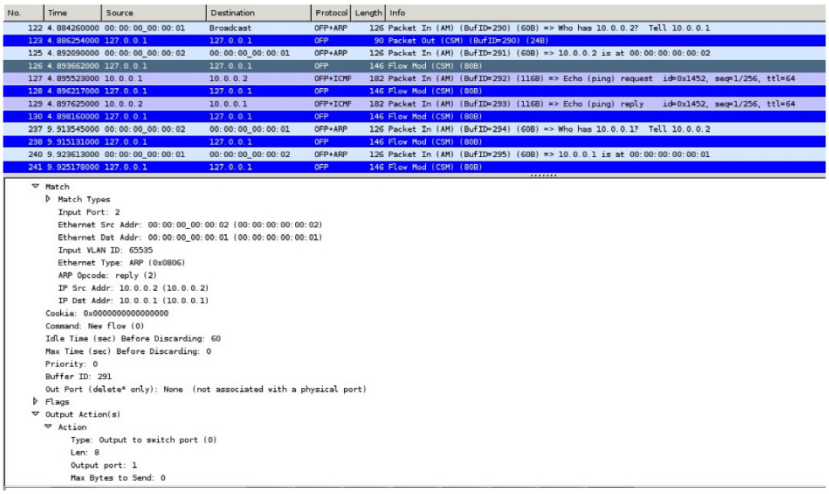
Η διαδικασία ξεκινά όταν ο h1 στέλνει ένα αίτημα ARP στον μεταγωγέα, ζητώντας να μάθει τη MAC διεύθυνση του h2. Ο μεταγωγέας δεν γνωρίζει πώς να διαχειριστεί το πακέτο και έτσι στέλνει το πακέτο ως μήνυμα PACKET-IN στον ελεγκτή. Ο ελεγκτής απαντά με ένα μήνυμα PACKET-OUT, που έχει μια ενέργεια η οποία οδηγεί τον μεταγωγέα να στείλει το πακέτο σε όλες τις πόρτες του εκτός από τη πόρτα εισόδου, περιμένοντας απάντηση στο αίτημα του. Όταν ο h2 απαντήσει στο αίτημα, ο μεταγωγέας στέλνει την απάντηση στον ελεγκτή επειδή δεν έχει καμία γνώση για το που θα προωθήσει το πακέτο. Όταν ο ελεγκτής λάβει την απάντηση ARP, στέλνει μήνυμα FLOW-MOD για να εγκαταστήσει ο μεταγωγέας μια νέα εγγραφή-ροής, η οποία θα χρησιμοποιηθεί στο μέλλον για τις ARP απαντήσεις από τον h2 και οι οποίες προωθούνται απευθείας από τον μεταγωγέα, χωρίς να κοινοποιούνται στον ελεγκτή. Η ίδια διαδικασία συμβαίνει όταν ο h1 στέλνει ICMP αίτηση / απάντηση και όταν ο h2 στέλνει ένα αίτημα ARP για να μάθει τη MAC διεύθυνση του h1 έχοντας ως συνέπεια την απάντηση ARP. Στο τέλος, πέντε νέες εγγραφές-ροής θα εγκατασταθούν στον πίνακα-ροής του μεταγωγέα από τον ελεγκτή OpenFlow, όπως φαίνεται και στην

Εικόνα **11** και στην

Εικόνα **12**.



Εικόνα 11: Διαδικασία Ping μεταξύ h1 και h2



Εικόνα 12: Πακέτα που ανταλλάζονται μεταξύ του h1 και του h2

2.8 Προδιαγραφές εκδόσεων OpenFlow

Εκτός από την έκδοση 1.0 του OpenFlow που περιγράφουμε στις προηγούμενες παραγράφους υπάρχουν και άλλες εκδόσεις που θα περιγράψουμε περιληπτικά σε σχέση με τις διαφορές ως προς την έκδοση 1.0 [13].

2.8.1 OpenFlow 1.1

Το OpenFlow 1.1 [15] κυκλοφόρησε τον Φεβρουάριο του 2011. Περιέχει σημαντικές αλλαγές σε σύγκριση με το OpenFlow 1.0. Για παράδειγμα, η επεξεργασία πακέτων λειτουργεί διαφορετικά. Στην έκδοση 1.1 τα πακέτα διεκπεραιώνονται μέσω αγωγού (pipiline) πολλαπλών πινάκων-ροής. Οι δύο κύριες αλλαγές είναι ένας αγωγός (pipiline)αποτελούμενος από πολλαπλούς πίνακες-ροής και έναν πίνακα-ομάδας.

2.8.2 OpenFlow 1.2

Το OpenFlow 1.2 [16] κυκλοφόρησε τον Δεκέμβριο του 2011. Έρχεται με εκτεταμένη υποστήριξη του πρωτοκόλλου σε σχέση με το IPv6. Το OpenFlow 1.2 μπορεί να ταιριάξει τις IPv6 διευθύνσεις πηγής και προορισμού, αριθμό πρωτοκόλλου, flow label, την κλάση της κίνησης στα διάφορα πεδία του ICMPv6. Επίσης, μπορεί ένας μεταγωγέας να συνδεθεί σε περισσότερους από έναν ελεγκτές.

2.8.3 OpenFlow 1.3

Το OpenFlow 1.3 [17], εισάγει νέες δυνατότητες για την παρακολούθηση, τις υπηρεσίες και τη διαχείριση (Monitoring Operations, Management-OAM). Για το σκοπό αυτό προστίθεται ένας πίνακας-μετρητής (Meter-table) στην αρχιτεκτονική του μεταγωγέα. Ο μετρητής συνδέεται άμεσα με μία εγγραφή του πίνακα-ροής από το αναγνωριστικό του μετρητή και μετρά το ποσοστό των πακέτων που έχουν ανατεθεί. Μια ζώνη-μετρητών (Meter-band) μπορεί να χρησιμοποιηθεί για να περιορίσει το σχετικό πακέτο ή το ρυθμό δεδομένων απορρίψεων των πακέτων όταν γίνεται υπέρβαση από ένα συγκεκριμένο ποσοστό. Αντί να απορρίπτει τα πακέτα, μια ζώνη-μετρητών μπορεί προαιρετικά να επαναχρωματίσει τα πακέτα με τροποποίηση του πεδίου των διαφοροποιημένων υπηρεσιών (DS field). Έτσι, απλά ή πολύπλοκα πλαίσια QoS μπορούν να υλοποιηθούν με το OpenFlow 1.3 και στις επόμενες εκδόσεις του.

2.8.4 OpenFlow 1.4

Το OpenFlow 1.4 [18] κυκλοφόρησε τον Οκτώβριο του 2013. Η ONF βελτίωσε την υποστήριξη για το OpenFlow Extensible Match (OXM). Προστέθηκαν TLV δομές για τις πόρτες, τους πίνακες και τις ουρές στο πρωτόκολλο, και δύσκολα κωδικοποιημένα τμήματα των προηγούμενων εκδόσεων έχουν πλέον αντικατασταθεί από νέες δομές TLV. Η ρύθμιση των οπτικών θυρών είναι πλέον δυνατή. Επιπλέον, οι ελεγκτές μπορούν να στείλουν μηνύματα ελέγχου σε μια

ενιαία δέσμη μηνυμάτων στους μεταγωγείς. Συμπεριλαμβάνονται επίσης, μικρές βελτιώσεις στους πίνακες-ομάδας και δυνατότητες παρακολούθησης.

2.9 OpenFlow Controllers

Ο ελεγκτής [10] είναι ο πυρήνας και το κύριο μέρος της λειτουργικού συστήματος του δικτύου (NOS) στο SDN. Είναι υπεύθυνος για το χειρισμό του πίνακα-ροών του μεταγωγέα, καθώς και για την επικοινωνία μεταξύ των εφαρμογών και των συσκευών του δικτύου χρησιμοποιώντας το πρωτόκολλο OpenFlow.

Οι ελεγκτές μπορούν να ταξινομηθούν σε δύο κύριες κατηγορίες:

1. Ανοικτού κώδικα, μεμονωμένοι ελεγκτές.
2. Εμπορικοί, κλειστού κώδικα, κατανεμημένοι ελεγκτές.

Οι ελεγκτές ανοικτού κώδικα είναι διαθέσιμοι για την έρευνα και την ανάπτυξη, αναπαριστώνται σαν μεμονωμένοι ελεγκτές με τη δυνατότητα ανάπτυξης διάφορων API’s για την υλοποίηση συγκεκριμένων διεργασιών. Υπάρχουν πολλοί Open source OpenFlow ελεγκτές, με κύρια μεταξύ τους διαφορά τη γλώσσα προγραμματισμού που είναι γραμμένοι.

Παρακάτω είναι μια λίστα για τους ελεγκτές ανοικτού κώδικα βασισμένη στη γλώσσα προγραμματισμού τους [19]:

- **C** : Trema (επίσης σε Ruby) και MUL
- **C++** : NOX (επίσης σε Python)
- **Java** : Beacon, Floodlight, Maestro και OpenDaylight [20]
- **Python** : POX και RYU.

Οι κατανεμημένοι ελεγκτές είναι σε θέση να λειτουργούν και να ελέγχουν το δίκτυο μέσω πολλαπλών ελεγκτών. Με την υλοποίηση αυτή, τα οφέλη που έχουν είναι επιπλέον αφαίρεση των επιπέδων στο επίπεδο ελέγχου και στην ανοχή σε σφάλματα. Μερικοί τέτοιοι ελεγκτές είναι: Onix [31], από Nicira Networks, IRIS [42], από την ερευνητική ομάδα του ETRI, Big Network Controller από Big Switch Networks και Programmable Flow από την NEC. Οι ελεγκτές Onix και IRIS έχουν την πρόσθετη δυνατότητα κλιμάκωσης απόδοσης με την προσθήκη επιπλέον ελεγκτών μέσα σε ένα συγκρότημα ελεγκτή (Controller Cluster).

Κεφάλαιο 3:
Κορυφαίοι δικτυακοί εξομοιωτές SDN

3.1 Εισαγωγή

Στο κομμάτι των εξομοιωτών δικτύων για το SDN χρησιμοποιούνται δύο εργαλεία που μπορούν να προσομοιώνουν ή να μιμηθούν ένα δίκτυο OpenFlow [21]. Ένα από αυτά είναι το EstiNet [22], το οποίο μπορεί να χρησιμοποιηθεί ως ένας προσομοιωτής ή εξομοιωτής και το άλλο είναι ο εξομοιωτής Mininet [23], [14].

Μια καλή ιδιότητα του EstiNet είναι ότι χρησιμοποιεί μία μεθοδολογία ονομαζόμενη «kernel re-entering» [24], [25] για να ενεργοποιήσει μη τροποποιημένες πραγματικές εφαρμογές προγραμμάτων που τρέχουν στους προσομοιωμένους τελικούς χρήστες (hosts). Λόγω αυτής της δυνατότητας, τα αποτελέσματα της προσομοίωσης του EstiNet προσομοιωτή είναι τόσο ακριβή όσο και τα αποτελέσματα που θα προέκυπταν από έναν εξομοιωτή. Έτσι, ο προσομοιωτής EstiNet δεν έχει μόνο τα πολλά καλά πλεονεκτήματα της προσομοίωσης, η οποία χρησιμοποιεί το δικό της ρολόι προσομοίωσης για να ελέγξει τη σειρά εκτέλεσης των γεγονότων προσομοίωσης, αλλά δημιουργεί επίσης και πολύ ακριβή αποτελέσματα.

Σε ένα OpenFlow δίκτυο προσομοιωμένο με το EstiNet, μπορεί να εκτελεστούν απευθείας προγράμματα OpenFlow ελεγκτών σε έναν προσομοιωμένο τελικό χρήστη (host) για τον έλεγχο των προσομοιωμένων μεταγωγέων OpenFlow χωρίς καμία τροποποίηση, όπως το NOX / POX [26], Ryu [27], και Floodlight [28].

Αναφορικά με τον EstiNet εξομοιωτή, όπως κάθε εξομοιωτής αυτό που πρέπει να κάνει είναι να εκτελέσει την εξομοίωση σε πραγματικό χρόνο και να μπορεί να επιτρέψει στο πρόγραμμα του ελεγκτή να εκτελεστεί σε ένα εξωτερικό μηχάνημα για να ελέγχει τους εξομοιωμένους μεταγωγείς OpenFlow.

Εξαιτίας της χρήσης της μεθοδολογίας «kernel re-entering» ωστόσο, ο EstiNet εξομοιωτής μπορεί επίσης να επιτρέψει το πρόγραμμα του ελεγκτή και τον εξομοιωμένο μεταγωγέα OpenFlow να εκτελούνται στο ίδιο μηχάνημα.

Ο Mininet εξομοιωτής είναι μια φθηνή και γρήγορα διαμορφώσιμη πλατφόρμα δοκιμών του δικτύου. Μέχρι στιγμής είναι το πιο γνωστό εργαλείο υποστήριξης του ερευνητικού δικτύου SDN OpenFlow, όπως παρατηρείται από το συνέδριο του ONS (Office for National Statistics) το 2013. Το Mininet χρησιμοποιεί virtual hosts, μεταγωγείς και συνδέσεις για να δημιουργήσει ένα δίκτυο σε έναν ενιαίο πυρήνα του λειτουργικού συστήματος, χρησιμοποιώντας την πραγματική στοίβα δικτύου για να επεξεργαστεί τα πακέτα και για να συνδεθεί σε πραγματικά δίκτυα.

Επιπλέον, δικτυακές εφαρμογές τύπου Unix / Linux-based έχουν επίσης τη δυνατότητα να τρέξουν στους virtual hosts. Σε ένα δίκτυο OpenFlow εξομοιωμένο από το Mininet, ένας πραγματικός OpenFlow ελεγκτής μπορεί να εκτελεστεί σε ένα εξωτερικό μηχάνημα ή στο ίδιο μηχάνημα όπου εξομοιώνονται οι virtual hosts.

3.2 EstiNet

Το «kernel re-entering» [24], [25], η μοναδική μεθοδολογία που χρησιμοποιείται από το EstiNet, υλοποιείται με τη χρήση tunneling του δικτύου διεπαφής για να συλλάβει τα πακέτα που αποστέλλονται κάτω από το επίπεδο IP στον πυρήνα του Linux και να τα στείλει στη μηχανή προσομοίωσης.

Εάν μία εφαρμογή τρέχει στον host 1 και στέλνει ένα πακέτο σε μία εφαρμογή που τρέχει στον host 2, το πακέτο θα περάσει από πραγματικό socket / TCP / IP επίπεδου στον πυρήνα του Linux και θα πάει σε μια tunnel διεπαφή που συνδέεται με την μηχανή προσομοίωσης. Εντός της μηχανής προσομοίωσης, κάθε host έχει τη δική του προσομοιωμένη στοίβα πρωτοκόλλων που περιέχει τον έλεγχο πρόσβασης στο μέσο (MAC), το φυσικό στρώμα, και άλλα επίπεδα πρωτοκόλλου κάτω από το IP επίπεδο. Αυτά τα επίπεδα προσομοιώνουν πολλές επιδράσεις όπως η καθυστέρηση διασύνδεσης (link delay), το εύρος ζώνης (link Bandwidth), το link down time, κ.λ.π. Στη συνέχεια εφαρμόζει τις επιδράσεις αυτές στον host 1, το πακέτο αποστέλλεται στο φυσικό επίπεδο του host 2 και θα επεξεργαστούν από τη στοίβα πρωτοκόλλων του host 2 μέσα στην μηχανή προσομοίωσης. Μετά από αυτό, το πακέτο στέλνεται έξω από την μηχανή προσομοίωσης σε μια άλλη tunnel διεπαφή που συνδέει το πρωτόκολλο IP με τον πυρήνα του Linux. Από εκείνο το σημείο και μετά το πακέτο θα περάσει από επεξεργασία των επιπέδων IP / TCP / socket του πυρήνα του Linux μέχρι να φθάσει στην εφαρμογή που τρέχει στον host 2.

Με τη μεθοδολογία KR (kernel Re-entering), το EstiNet επιτρέπει όχι μόνο σε πραγματικές εφαρμογές να τρέχουν σε προσομοιωμένους τελικούς χρήστες (hosts), αλλά επίσης να χρησιμοποιεί το πραγματικό πρωτόκολλο TCP / IP στον πυρήνα του Linux για να ρυθμίσει τις μεταξύ τους TCP συνδέσεις. Σημειώνεται ότι σε ένα πραγματικό δίκτυο OpenFlow κάθε μεταγωγέας OpenFlow πρέπει να δημιουργήσει μια σύνδεση TCP με έναν πραγματικό ελεγκτή OpenFlow. Με τις δυνατότητες που προσφέρει η μεθοδολογία KR, σε ένα προσομοιωμένο OpenFlow δίκτυο στο EstiNet, κάθε προσομοίωση OpenFlow μεταγωγέα μπορεί να εγκαθιδρύσει μια πραγματική σύνδεση TCP με έναν πραγματικό ελεγκτή OpenFlow και η επεξεργασία των πρωτοκόλλων μεταξύ του OpenFlow ελεγκτή και του OpenFlow μεταγωγέα είναι ακριβώς το ίδιο όπως με την επεξεργασία των πρωτοκόλλων για τη ρύθμιση μίας TCP σύνδεσης μεταξύ 2 πραγματικών τελικών χρηστών (hosts).

3.3 Mininet

Το Mininet [23] δημιουργεί virtual hosts χρησιμοποιώντας μια διαδικασία που βασίζεται στη μέθοδο του virtualization και του δικτυακού μηχανισμού namespace, η οποία αποτελεί χαρακτηριστικό που υποστηρίζεται από την έκδοση 2.2.26 του Linux, για να διαχωρίσει τις διασυνδέσεις δικτύου, τους πίνακες δρομολόγησης και

τους πίνακες ARP των διαφορετικών virtual hosts. Οι εικονικοί μεταγωγείς στο Mininet είναι είδος λογισμικού OpenFlow μεταγωγέα που ονομάζεται "Open vSwitch" [29].

Οι διασυνδέσεις μεταξύ των εικονικών host και των εικονικών μεταγωγέων υλοποιείται με τη χρήση εικονικών ζευγών Ethernet που παρέχονται από τον πυρήνα του Linux. Εάν ένα πακέτο στέλνεται από μια εφαρμογή που εκτελείται στον host 1 σε μια άλλη εφαρμογή που εκτελείται στον host 2, θα επεξεργάζονται από τη πραγματική δικτυακή στοίβα των πρωτοκόλλων στον πυρήνα του Linux.

Σε ένα δίκτυο OpenFlow που προσομοιώνεται στο Mininet, οι εικονικοί μεταγωγείς πρέπει να ρυθμίζουν τις συνδέσεις TCP σε έναν πραγματικό OpenFlow ελεγκτή, ο οποίος μπορεί να εκτελείται σε έναν εικονικό host ή σε ένα εξωτερικό μηχάνημα. Ωστόσο, επειδή στο Mininet οι κύκλοι της CPU πρέπει να διαμοιράζονται σε όλα τα virtual hosts, virtual switches και στον ελεγκτή που εκτελείται στον ενιαίο πυρήνα του λειτουργικού συστήματος, ο χρονοπρογραμματισμός για τις εργασίες αυτές δεν μπορεί να ελεγχθεί με ακρίβεια από τον χρονοπρογραμματιστή της CPU στον πυρήνα του Linux. Τα αποτελέσματα που προκύπτουν από τον Mininet εξομοιωτή δεν μπορούν να επαναληφθούν και μερικές φορές μπορεί να διαφέρουν από τα σωστά αποτελέσματα.

Κεφάλαιο 4:
Επιλογή των Εργαλείων

4.1 Floodlight OpenFlow Controller

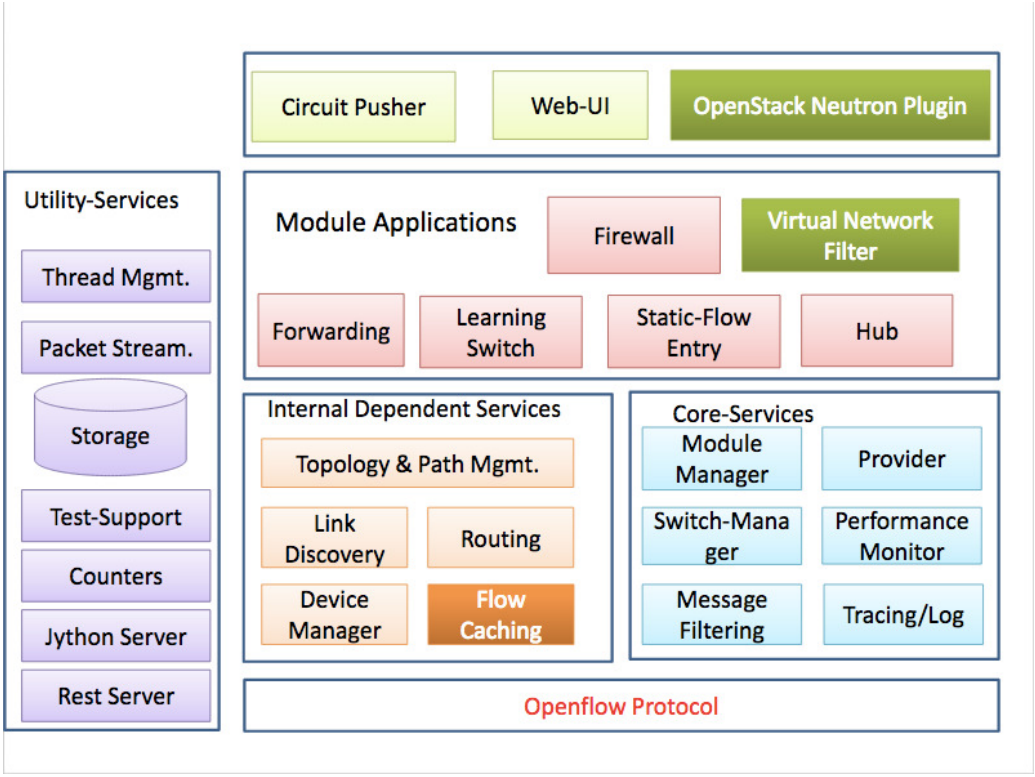
Ο Floodlight ελεγκτής είναι βασισμένος στη Java όπως ο Beacon, αλλά υιοθετεί μια διαφορετική αρχιτεκτονική και τρόπο λειτουργίας [30]. Έχει αυξηθεί αλματωδώς να είναι ένα από τους πιο δημοφιλής open source SDN ελεγκτές με περισσότερες από 15.000 λήψεις. Επίσης, αναγνωρίζεται ως καλύτερος σε σχέση με τα χαρακτηριστικά και τις επιδόσεις.

Ο Floodlight Controller έχει άδεια από την Apache και ένα χαρακτηριστικό του είναι ότι έχει συνεισφέρει σημαντικά στην εξέλιξή του η εταιρία Big Switch Networks στην κοινότητα του ανοικτού λογισμικού. Η αρχιτεκτονική του Floodlight βασίζεται σε έναν Big Network Controller (BNC), που είναι μία εμπορική προσφορά της εταιρείας. Οι εφαρμογές που έχουν γραφτεί για τον Floodlight Controller από κάθε προγραμματιστή μπορεί να διατεθούν για πιστοποίηση και για εξειδίκευση στον BNC. Όπως και πολλοί άλλοι ελεγκτές, όταν εκτελούμε τον Floodlight, δύο διαδικασίες γίνονται ενεργές, η Northbound και η Southbound. Δηλαδή, όταν εκτελούμε τον Floodlight, θα εκτελεστεί ο ελεγκτής και το σύνολο των διαμορφωμένων εφαρμογών module. Τα Northbound REST APIs εκθέτονται από όλα τα modules που τρέχουν και είναι διαθέσιμα μέσω συγκεκριμένων REST θύρων (port). Κάθε module μπορεί να αλληλεπιδράσει (ανάκτηση πληροφοριών και υπηρεσίες επίκλησης) με τον ελεγκτή από την αποστολή εντολών http REST. Από την άλλη πλευρά, στο Southbound, το module που παρέχεται από τον Floodlight, θα ξεκινήσει την ακρόαση των OpenFlow-καθοριζομένων TCP-port στο δίκτυο σχετικά με τις συνδέσεις από τους μεταγωγείς OpenFlow.

4.1.1 Floodlight Αρχιτεκτονική

Ο όρος «modular architecture» χρησιμοποιείται για να περιγράψει την αρχιτεκτονική του Floodlight Controller, που φαίνεται στην Εικόνα 13. Η βασική αρχιτεκτονική περιλαμβάνει διάφορα modules, όπως διαχείριση τοπολογίας, διαχείριση συσκευών / τερματικού-σταθμού, υπολογισμό μονοπατιού / διαδρομής, υποδομή για πρόσβαση στο διαδίκτυο (διαχείριση), αποθήκευση μετρητών (μετρητές OpenFlow), και ένα σύστημα αποθήκευσης κατάστασης (topology management, device/end-station management, path/route computation, infrastructure for web access (management), counter store (OpenFlow counters), and a state storage system) που είναι καλά δεμένα από ένα σύστημα module-

διαχείρισης. Παρακάτω θα περιγράψουμε μερικά από τα σημαντικά στοιχεία της αρχιτεκτονικής του ελεγκτή.



Εικόνα 13: Αρχιτεκτονική Floodlight controller

4.1.2 Εφαρμογές βασισμένες στο Rest-API

Ο Floodlight έρχεται με μερικές εφαρμογές που χρησιμοποιούν το εκτεθειμένο REST APIs. Το Κύκλωμα Pusher χρησιμοποιεί Floodlight Rest APIs για να δημιουργήσει μια διαδρομή μεταξύ δύο οποιωνδήποτε IP διευθυνσιοδοτήσεων συσκευών προσθέτοντας καταχωρήσεις-ροών σε όλους τους μεταγωγείς που αποτελούν τη διαδρομή. Εκτός από το κύκλωμα Pusher, ο Floodlight μπορεί να λειτουργήσει ως backend δίκτυο για το OpenStack χρησιμοποιώντας ένα Neutron plugin. Υπάρχουν δύο κύρια συστατικά για αυτή τη λύση:

- Ένας RestProxy για να πραγματοποιήσει τη συνδεσιμότητα μεταξύ Floodlight Controller και OpenStack Neutron.
- Ένα VirtualNetworkFIlter για να πραγματοποιήσει τα Neutron APIs. Το module VirtualNetworkFilter, υλοποιεί την L2-διεύθυνση, δεν εξαρτάται από την OpenStack-Neutron, και μπορεί να ενεργοποιηθεί μέσω ενός αρχείου ρυθμίσεων. Ωστόσο, το plugin RestProxy σχεδιάστηκε για να τρέξει ως μέρος της υπηρεσίας OpenStack Neutron.

4.1.3 Εφαρμογές Built-In Module

Ο Floodlight περιλαμβάνει πολλαπλές εφαρμογές. Στη συνέχεια θα συνοψίσουμε μερικές. Συνήθως, αυτές οι εφαρμογές είναι εξαιρετικές σε έναν προγραμματιστή για να κατανοήσει τη χρήση των APIs, εάν ενδιαφέρεται για την ανάπτυξη SDN-εφαρμογών. Αυτό ισχύει για όλους τους ελεγκτές SDN, και όχι μόνο στον Floodlight.

Η εφαρμογή forwarding, όπως υποδηλώνει το όνομά της, διαβιβάζει τα πακέτα μεταξύ δύο συσκευών που μπορούν να συνδεθούν μέσω OpenFlow (ή μη-OpenFlow) μεταγωγέων. Η εφαρμογή learning-switch, εδώ ο μεταγωγέας θα εξετάσει κάθε πακέτο για να κάνει χαρτογράφηση πόρτας-προέλευσης (source-port mapping). Στη συνέχεια, η διεύθυνση MAC προέλευσης θα πρέπει να συνδέεται με την πόρτα. Εάν ο προορισμός του πακέτου έχει ήδη συσχετιστεί με κάποια πόρτα, το πακέτο θα σταλεί στην πόρτα, αλλιώς θα μπορούσε να έχει έξοδο σε όλες τις πόρτες του μεταγωγέα. Η εφαρμογή hub στέλνει ακριβώς κάθε εισερχόμενο πακέτο σε όλες τις άλλες ενεργές πόρτες. Η εφαρμογή static-flow entry pusher απλά προσθέτει μια καταχώρηση ροής OpenFlow (Match+Action) σε οποιοδήποτε ή συγκεκριμένο μεταγωγέα, χρησιμοποιώντας flow-mod μήνυματα του OpenFlow. Η VirtualNetworkFilter εφαρμογή, όπως περιγράφεται παραπάνω είναι μια MAC-Address βασισμένη σε μία εφαρμογή που χωρίζει το δίκτυο σε τμήματα. Τέλος, υπάρχει μια εφαρμογή firewall για να εφαρμόσει λίστες ελέγχου πρόσβασης (ACL - Access Control List), που δεν είναι τίποτα άλλο, αλλά ένα σύνολο συνθηκών για τον έλεγχο (να επιτρέψει ή να αρνηθεί) τη ροή της κυκλοφορίας με βάση συγκεκριμένο σύνολο πολιτικών.

4.1.4 Υπηρεσίες Core, Internal and Utility

Ένα σημαντικό σημείο που πρέπει να σημειωθεί για κάθε ελεγκτή SDN είναι ότι οι ενσωματωμένες υπηρεσίες καθορίζουν την ικανότητα του ελεγκτή και αυτές οι υπηρεσίες είναι αυτές που χρησιμοποιούνται από τις northbound εφαρμογές. Έτσι ο Floodlight περιλαμβάνει τις υπηρεσίες που κυμαίνονται από την ανακάλυψη της κατάστασης του δικτύου και τα συμβάντα για την ενεργοποίηση μίας επικοινωνίας μεταγωγέων υποστηρίζοντας ικανότητες όπως η αποθήκευση, νήματα και web-UI. Παρακάτω στον

Πίνακας **4** περιγράφονται μερικές από αυτές τις υπηρεσίες.

Όνομα Υπηρεσίας	**Περιγραφή**
Floodlight-Provider	Διαχειρίζεται την ασφαλή σύνδεση των μεταγωγέων. Ο Floodlight Provider είναι ένα module που είναι υπεύθυνο για τη μετάφραση των ληφθέντων μηνυμάτων OpenFlow σε γεγονότα, τα οποία μπορούν να υποβληθούν σε επεξεργασία από άλλα module. Επιπλέον, εξασφαλίζει τη διάταξη των ληφθέντων μηνυμάτων, πριν περάσουν στα module που πρέπει να

	πάνε τα μηνύματα. [IOFMessageListener]
Device-Manager	Η υπηρεσία Device Manager παρακολουθεί τις συσκευές ή τερματικούς-σταθμούς μέσω αιτημάτων PacketIn. Παρά τις πληροφορίες (MAC, VLAN, κλπ) που υπάρχουν στο PacketIn, μπορεί επίσης να μάθει σε ποια πόρτα ποιου μεταγωγέα είναι συνδεδεμένη μία συσκευή.
Link Discovery Manager	Η υπηρεσία Link Discovery Manager ανακαλύπτει τις διασυνδέσεις. Χρησιμοποιεί LLDPs για να ανιχνεύσει διασυνδέσεις, δηλαδή όταν μία σύνδεση είναι έτοιμη για εγκαθίδρυση μεταξύ δύο μεταγωγέων εάν σταλεί ένα LLDP από μία πόρτα του ενός μεταγωγέα και ληθφεί σε μία πόρτα του άλλου μεταγωγέα τότε η σύνδεση ανιχνεύεται.
Topology-Service	Η υπηρεσία Topology-Service υπολογίζει τις τοπολογίες με βάση τις πληροφορίες από το Link-Discovery-Manager. Εδώ, χρησιμοποιείται ένας όρος OpenFlow island για να δηλώσει μια ομάδα συνδεδεμένων OpenFlow-μεταγωγέων που διαχειρίζονται από τον ίδιο SDN-ελεγκτή (Floodlight). Επίσης, τα «νησιά» μπορούν να διασυνδεθούν χρησιμοποιώντας μεταγωγείς μη-OpenFlow στο ίδιο στρώμα 2 του τομέα.
Flow Cache	Το module αυτό υπάρχει για να ενθαρρύνει τους προγραμματιστές εφαρμογών για να εφαρμόσουν λύσεις ανάλογα με τις ανάγκες τους, με το χειρισμό μιας σειράς από γεγονότα από το δίκτυο με το δικό τους τρόπο.
Packet Streamer	Η υπηρεσία Packet Streamer μπορεί να διαβιβάσει τα πακέτα OpenFlow σε οποιαδήποτε συνδεδεμένη συσκευή παρακολούθησης στο δίκτυο. Παρέχει μια διεπαφή για να ορίσει (με όρους ενός ή πολλαπλών πεδίων στο πακέτο) τα ενδιαφερόμενα μηνύματα OpenFlow και συνήθως αποκαλείται ως ένα φίλτρο.
Memory Storage Source	Η υπηρεσία MemoryStorageSource αποτελεί ‘in-memory’ storage source - αποθήκευση των κοινών δεδομένων. Τα modules του Floodlight που εξαρτώνται από αυτό μπορούν να δημιουργήσουν / διαγράψουν / τροποποιήσουν τα δεδομένα στην προέλευση αποθήκευσης μνήμης.
ThreadPool	Η υπηρεσία ThreadPool είναι module του Floodlight που είναι «περιτύλιγμα» για μία υπηρεσία Scheduled Executor Service της Java. Μπορεί να χρησιμοποιηθεί για τη δημιουργία νημάτων που μπορούν να τρέξουν σε συγκεκριμένες ώρες ή σε περιοδικό χρόνο.

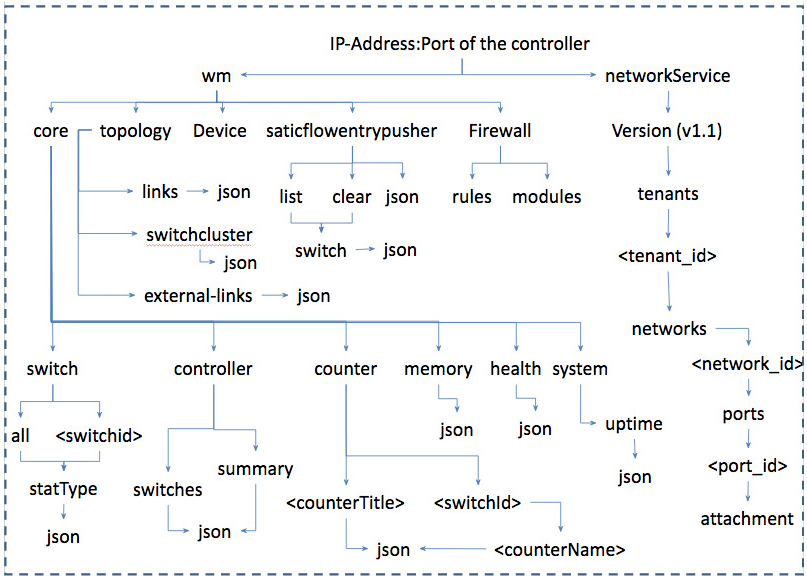
Πίνακας 4: Υπηρεσίες Floodlight

4.1.5 Floodlight Controller RestAPIs

Ο Fllodlight περιλαμβάνει ένα διακομιστή RestAPI, που χρησιμοποιεί τη βιβλιοθήκη Restlets. Με την restlets, κάθε module που αναπτύχθηκε μπορεί να εκθέσει πρόσθετες REST APIs μέσω μιας υπηρεσίας IRestAPI (συνήθως), τα modules που εξαρτώνται από τον διακομιστή REST εκθέτουν τα APIs από την εφαρμογή RestletRoutable σε μια κλάση. Ο ίδιος ο ελεγκτής παρουσιάζει μια σειρά επεκτάσιμων REST APIs για να ορίσει διάφορους τύπους πληροφοριών. Το REST API (Εικόνα 14) είναι η συνιστώμενη διεπαφή για την ανάπτυξη εφαρμογών που χρησιμοποιούν τις υποστηριζόμενες λειτουργίες του Floodlight.

Όταν ο Floodlight είναι σε λειτουργία, μπορεί κανείς να χρησιμοποιήσει τα API που υποστηρίζονται ήδη στον ελεγκτή ή να συνοψίζει τις διάφορες λειτουργίες που υποστηρίζονται από τον Floodlight. Για παράδειγμα, η παρακάτω εντολή curl διαβάζει τους μεταγωγείς που συνδέονται με τον ελεγκτή(Controller) με IP 10.0.0.1.

- curl http://10.0.0.1:8080/wm/core/controller/switches/json



Εικόνα 14: Δομή Floodlight REST API

4.2 Ο δικτυακός εξομοιωτής Mininet

Το Mininet [14], [31] είναι ένας δικτυακός εξομοιωτής. Έχει την δυνατότητα να εκτελεί ταυτόχρονα ένα σύνολο από τερματικά, δρομολογητές, μεταγωγείς ethernet άλλα και των αντίστοιχων συνδέσμων σε ένα ενιαίο Linux Kernel (πυρήνα). Χρησιμοποιεί την τεχνολογία της εικονοποίησης ώστε να μπορεί να είναι ένα ενιαίο σύστημα και να προσομοιώνεται ως ένα πλήρες δίκτυο, χρησιμοποιώντας το ίδιο σύστημα πυρήνα και με τους ίδιους κωδικούς χρήστη. Το κάθε εικονικό τερματικό

στο mininet λειτουργεί σαν ένα πραγματικό τερματικό. Επιπλέον, παρέχεται η δυνατότητα ασφαλούς σύνδεσης (τύπου SSH) στο τερματικό και η εκτέλεση οποιοδήποτε προγράμματος (με την προϋπόθεση ότι αυτό είναι εγκατεστημένο στο σύστημα Linux). Τα προγράμματα που εκτελούνται μπορούν να αποστείλουν πακέτα μεταξύ των τερματικών καθώς και να αναγνωρίζουν την σύνδεση μεταξύ τους ως διεπαφές τύπου Ethernet. Η αποστολή των πακέτων πραγματοποιείται με δεδομένη ταχύτητα σύνδεσης και την απαιτούμενη καθυστέρηση. Τα πακέτα επεξεργάζονται από συσκευές που λειτουργούν ως μεταγωγείς (Ethernet swtiches, routers) με δεδομένο χρόνο σε ουρές αναμονής. Όταν δύο προγράμματα, όπως για παράδειγμα το iperf (το οποίο μετράει την χωρητικότητα της γραμμής μεταξύ δύο σημείων) μεταξύ ενός πελάτη (client) και ενός διακομιστή (server) επικοινωνούν μέσω Mininet, η μετρούμενη απόδοση θα πρέπει να είναι κοινή με αυτή των δύο φυσικών μηχανών.

Εν συντομία, στο Mininet τα τερματικά, οι δρομολογητές, οι μεταγωγείς, οι ελεγκτές και οι συνδέσεις δημιουργούνται με τη χρήση λογισμικού (software) και όχι υλικού (hardware). Είναι δυνατόν η δημιουργία ενός δικτύου Mininet παρόμοιο με ένα πραγματικό δίκτυο που βασίζεται σε hardware ή η δημιουργία ενός δικτύου hardware παρόμοιο με αυτό του Mininet, τα οποία να εκτελούν τον ίδιο δυαδικό κώδικα και εφαρμογές στην κάθε πλατφόρμα.

Το Mininet είναι ένα εύχρηστο και αξιόπιστο εργαλείο στην προσομοίωση δικτύων έχοντας πολλά πλεονεκτήματα όπως:

- Είναι γρήγορο: μπορεί να ξεκινήσει ένα απλό δίκτυο σε μερικά δευτερόλεπτα, πράγμα που το κάνει να είναι γρήγορο στο να εκτελείται, να τροποποιείται και να αποσφαλματώνεται.
- Μπορούμε να δημιουργούμε δικές μας τοπολογίες: με έναν μεταγωγέα, μεγάλες τοπολογίες σαν το Internet, data center, κ.λ.π.
- Μπορούμε να εκτελέσουμε πραγματικά προγράμματα: ότι τρέχει σε Linux είναι στη διάθεσή μας, από web servers στο TCP, έως εργαλεία παρακολούθησης, όπως το Wireshark.
- Μπορούμε να τροποποιήσουμε την προώθηση των πακέτων: Οι μεταγωγείς-Mininet μπορούν να προγραμματιστούν χρησιμοποιώντας OpenFlow πρωτόκολλο.
- Μπορούμε να το εκτελέσουμε σε οποιαδήποτε υπολογιστή, διακομιστή, εικονική μηχανή ή ακόμα και σε τεχνολογία τύπου cloud (νεφοϋπολογιστική).
- Τα αποτελέσματα του λογισμικού μπορούν να αναπαραχθούν από οποιοδήποτε χρήστη καθώς το μόνο που απαιτείται είναι η εκτέλεση του ίδιου κώδικα στο αντίστοιχο τερματικό.
- Μπορούμε να το χρησιμοποιούμε εύκολα: για τη δημιουργία και την εκτέλεση πειραμάτων στο Mininet απαιτείται προγραμματισμός σε γλώσσα Python.
- Αποτελεί έργο ανοιχτού κώδικα και βρίσκεται υπό ενεργή ανάπτυξη: η κοινότητα του Mininet αποτελείται από χρήστες και προγραμματιστές και μπορούν να συμβάλλουν στην επίλυση οποιουδήποτε προβλήματος που μπορεί να αντιμετωπίσει ο εκάστοτε χρήστης.

Πέρα όμως από τα πλεονεκτήματα, έχει και ορισμένους περιορισμούς [23]. Ο κυριότερος περιορισμός του Mininet είναι μια έλλειψη απόδοσης στα υψηλά φορτία. Οι πόροι της CPU πολλαπλασιάζονται γρήγορα από τον χρονοπρογραμματιστή του Linux, ο οποίος δεν παρέχει καμία εγγύηση ότι ένας τελικός χρήστης (host) που είναι έτοιμος να στείλει ένα πακέτο θα χρονοπρογραμματιστεί αμέσως ή ότι όλοι οι μεταγωγείς θα προωθούν με τον ίδιο ρυθμό. Επιπλέον, η λογισμική προώθηση των πακέτων δεν ταιριάζει με αυτή του υλικού. Η διαδικασία αναζήτησης (lookup) ενός λογισμικού πίνακα αποτελεί O(n) γραμμική χρήση της μνήμης και δεν μπορεί να πλησιάσει το O(1) άμεσης χρήσης της μνήμης της διαδικασίας αναζήτησης από ένα υλικό που διαθέτει επιταχυνόμενη μνήμη τύπου TCAM, προκαλώντας έτσι τον ρυθμό της προώθησης των πακέτων να πέφτει σε μεγάλου μεγέθους πίνακες.

Για να επιβάλει τα όρια του εύρους ζώνης και την ποιότητα της υπηρεσίας (QoS) σε μια σύνδεση, χρησιμοποιείται το πρόγραμμα ελέγχου κίνησης (TC) του Linux. Η μερική εικονικοποίηση του Mininet προσεγγίζει επίσης περιορισμούς στο τι μπορεί να κάνει. Δεν μπορεί να χειριστεί διαφορετικούς πυρήνες ταυτόχρονα. Όλοι οι hosts μοιράζονται το ίδιο σύστημα αρχείων, αν και αυτό μπορεί να αλλάξει με τη χρησιμοποίηση της chroot εντολής. Οι hosts δεν μπορούν να μεταναστεύσουν (migrated) ενεργά όπως τα VMs. Αυτοί οι περιορισμοί είναι ένα λογικό tradeoff για τις δυνατότητες να δοκιμαστούν ιδέες σε μεγάλη κλίμακα.

4.2.1 Τοπολογίες στο Mininet

Το Mininet υποστηρίζει τη δημιουργία παραμετροποιήσιμων τοπολογιών. Με την δημιουργία του αντίστοιχου κώδικα Python, παρέχεται η δυνατότητα δημιουργίας ευέλικτης τοπολογίας η οποία μπορεί να διαμορφωθεί με βάση τις παραμέτρους που εντάσσονται στον κώδικα, και μπορεί να επαναχρησιμοποιηθεί σε πολλαπλά πειράματα.

Για παράδειγμα, παρακάτω (

Εικόνα **15**) παρουσιάζεται μία τοπολογία δικτύου, η οποία αποτελείται από έναν καθορισμένο αριθμό χρηστών (hosts) που συνδέονται με έναν μεταγωγέα.

```
#!/usr/bin/python

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.util import dumpNodeConnections
from mininet.log import setLogLevel

class SingleSwitchTopo(Topo):
    "Single switch connected to n hosts."
    def build(self, n=2):
        switch = self.addSwitch('s1')
        # Python's range(N) generates 0..N-1
        for h in range(n):
            host = self.addHost('h%s' % (h + 1))
            self.addLink(host, switch)

def simpleTest():
    "Create and test a simple network"
    topo = SingleSwitchTopo(n=4)
    net = Mininet(topo)
    net.start()
    print "Dumping host connections"
    dumpNodeConnections(net.hosts)
    print "Testing network connectivity"
    net.pingAll()
    net.stop()

if __name__ == '__main__':
    # Tell mininet to print useful information
    setLogLevel('info')
    simpleTest()
```

Εικόνα 15: Κώδικας τοπολογίας Mininet

Οι κλάσεις και οι συναρτήσεις που χρησιμοποιήθηκαν στον κώδικα της αναλύονται παρακάτω:

- Topo: η βασική κλάση που χρησιμοποιείται στις τοπολογίες Mininet.
- addSwitch(): προσθέτει έναν μεταγωγέα στην τοπολογία και επιστρέφει την ονομασία του μεταγωγέα.
- addHost(): προσθέτει τερματικό στην τοπολογία και επιστρέφει την ονομασία του.
- addLink(): προσθέτει μια αμφίδρομη σύνδεση στην τοπολογία.(Οι σύνδεσεις στο Mininet είναι αμφίδρομες, εκτός αν αναφέρεται διαφορετικά.)
- Mininet: κύρια κατηγορία για την δημιουργία και τη διαχείριση του δικτύου.
- start(): ενεργοποιεί την λειτουργία του δικτύου.
- pingAll(): ελέγχει τη συνδεσιμότητα των τερματικών εκτελώντας διαδοχικά αιτήσεις ping μεταξύ των κόμβων.
- stop(): τερματίζει τη λειτουργία του δικτύου.
- net.hosts: επιστρέφει την ονομασία όλων των κόμβων.
- dumpNodeConnections(): απορρίπτει συνδέσεις προς/από ένα σύνολο κόμβων.
- setLogLevel('info' | 'debug' | 'output'): ορίζει το προκαθορισμένο επίπεδο εξόδου του Mininet's 'info' που είναι προτεινόμενο γιατί παρέχει χρήσιμες πληροφορίες.

4.2.2 Ρύθμιση παραμέτρων απόδοσης

Εκτός από τις βασικές λειτουργίες της δικτύωσης, το Mininet παρέχει δυνατότητα ρύθμισης της απόδοσης και απομόνωσης ορισμένων χαρακτηριστικών, μέσω των κλάσεων CPULimitedHost και TCLink. Οι πιο σημαντικές μέθοδοι και παράμετροι που χρησιμοποιούνται δίνονται παρακάτω:

- self.addHost(name, cpu=f): Με τη χρήση αυτής της εντολής επιτρέπεται ο ορισμός του ποσοστού της συνολικής CPU του συστήματος που θα χρησιμοποιήσει ο εικονικός Host.
- self.addLink(node1, node2, bw=10, delay='5ms', max_queue_size=1000, loss=10, use_htb=True): Δημιουργεί σύνδεση διπλής κατεύθυνσης μεταξύ δύο κόμβων με συγκεκριμένα χαρακτηριστικά, όπως εύρος ζώνης, καθυστέρηση, ανεκτικότητα απωλειών πακέτων, μέγιστο μέγεθος της ουράς αναμονής τα 1000 πακέτα. Η παράμετρος bw εκφράζεται σε Mb/s, ενώ η delay ακολουθείται με την αντίστοιχη μονάδα χρόνου (s, ms, us). Αντίθετα η παράμετρος loos εκφράζεται σε ποσοστό επι τοις εκατό.

4.2.3 Εκτέλεση προγραμμάτων στους Hosts και μέθοδοι παραμετροποίησής τους

Η εκτέλεση προγραμμάτων στα τερματικά αποτελεί το πιο αξιοσημείωτο γεγονός κατά την εκτέλεση των πειραμάτων [31], έτσι ώστε να μπορούν να υποστηριχτούν επιπλέον εντολές από τις συνηθισμένες εντολές τύπου pingAll () και iperf (). Η διαδικασία αυτή υποστηρίζεται από το λογισμικό Mininet.

Κάθε τερματικό στο Mininet αποτελεί ουσιαστικά ένα κέλυφος τύπου bash που συνδέεται με μία ή και περισσότερες διεπαφές δικτύου με αποτέλεσμα να υποστηρίζει την εκτέλεση εντολών τύπου bash. Για τον λόγο αυτό για την επικοινωνία με κάθε τερματικό χρησιμοποιείται κυρίως μέθοδος τύπου CMD.

Για την εκτέλεση μίας εντολής από κάποιον ξενιστή και την αποτύπωση του αποτελέσματος, μέσω μεθόδου cmd, χρησιμοποιείται ο παρακάτω κώδικας:

```
h1 = net.get('h1')
result = h1.cmd('ifconfig')
print result
```

Οι hosts στο mininet παρέχουν μια σειρά από μεθόδους που συμβάλλουν στην ευκολία της διαμόρφωση του δικτύου.

- IP(): επιστρέφει τη διεύθυνση IP του τερματικού ή κάποιας συγκεκριμένης διεπαφής.
- MAC(): επιστρέφει τη διεύθυνση MAC του τερματικού ή κάποιας συγκεκριμένης διεπαφής.
- setARP(): δημιουργεί εγγραφή static ARP στην ARP cache του τερματικού.

- setIP(): ρύθμιση συγκεκριμένης διεύθυνσης IP για κάποιο τερματικό ή διεπαφή.
- setMAC(): ρύθμιση συγκεκριμένης διεύθυνσης IP για κάποιο τερματικό ή διεπαφή.

4.2.4 Διαμοιραζόμενο Σύστημα Αρχείων

Οι εικονικοί ξενιστές του Mininet διαμοιράζονται από προεπιλογή τους φακέλων root του συστήματος του υποκείμενου διακομιστή. Αντίθετα, η δημιουργία νέου ξεχωριστού συστήματος (file system) αποτελεί χρονοβόρα διαδικασία και ιδιαίτερα δύσκολη [14], [31].

Η κοινή χρήση των αρχείων του συστήματος παρέχει το πλεονέκτημα ότι δεν θα χρειαστεί η αντιγραφή των δεδομένων μεταξύ των ξενιστών καθώς έχουν ήδη δημιουργηθεί. Το γεγονός αυτό όμως, έχει και ένα σημαντικό μειονέκτημα. Σε περίπτωση που απαιτείται ειδική διαμόρφωση για κάποιο πρόγραμμα (πχ. Httpd), απαιτείται η δημιουργία νέων αρχείων ρύθμισης παραμέτρων για κάθε host. Επιπλέον, δημιουργείται και ο κίνδυνος συγκρούσεων αρχείων, σε περίπτωση που δημιουργηθεί στον ίδιο κατάλογο το ίδιο αρχείο.

4.2.5 Mininet CLI

Το Mininet περιλαμβάνει περιβάλλον γραμμής εντολών (Command Line Interface) που μπορούν να λειτουργήσει και σε ένα δίκτυο. Παρέχει μια ποικιλία από χρήσιμες εντολές, καθώς και τη δυνατότητα εμφάνισης παραθύρων xterm για την εκτέλεση εντολών σε επιμέρους κόμβους του δικτύου σας.

4.2.6 Mininet API

Καλούμε API ή Διεπαφή Προγραμματισμού Εφαρμογών γνωστή και ως Διασύνδεση Προγραμματισμού Εφαρμογών, τη διεπαφή των προγραμματιστικών διαδικασιών που ένα λειτουργικό σύστημα, βιβλιοθήκη ή εφαρμογή παρέχει προκειμένου να επιτρέπει να γίνονται προς αυτό αιτήσεις από άλλα προγράμματα ή ανταλλαγή δεδομένων [14], [31].

Στις προηγούμενες παραγράφους δόθηκε μια σειρά από κλάσεις της Python που περιλαμβάνονται στο API του Mininet, συμπεριλαμβανομένου των κλάσεων Topo, Mininet, Host, Switch, Link και των υποκατηγοριών τους. Για την απλοποίηση και τη διευκόλυνση του προγραμματισμού, οι κλάσεις χωρίζονται σε τρείς κατηγορίες-επίπεδα: υψηλού επιπέδου API, μεσαίου επιπέδου API και χαμηλού επιπέδου API.

- **Χαμηλού επιπέδου API**: Αποτελείται από τις κύριες κλάσεις που αφορούν τους κόμβους και τις συνδέσεις (όπως Host, Switch, και Link και τις υποκατηγορίες τους), οι οποίες χρησιμοποιούνται για τη δημιουργία δικτύου. Ο τρόπος δημιουργίας δικτύου μόνο με κλάσεις του επιπέδου αυτού αποτελεί διαδικασία ιδιαίτερα δύσχρηστη.
- **Μεσαίου επιπέδου API**: Προσθέτει αντικείμενα τύπου Mininet, τα οποία χρησιμεύουν ως επιπρόσθετα στοιχεία και ρυθμίσεις σε κόμβους και συνδέσεις. Παρέχει μία σειρά από μεθόδους (όπως addHost(), addSwitch(),

και addLink()) για την πρόσθεση κόμβων και συνδέσεων στον δίκτυο, καθώς και για τη διαμόρφωση του δικτύου, την εκκίνηση και τον τερματισμό του (start(), stop()).

- **Υψηλού επιπέδου API**: Παρέχει πρόσθετες επιλογές ρύθμισης της τοπολογίας. Μέσω της κλάσης topo προσφέρει τη δυνατότητα δημιουργίας υποδείγματος τοπολογίας που μπορεί να παραμετροποιηθεί και να επαναχρησιμοποιηθεί. Τα υποδείγματα του τύπου αυτού ορίζονται μέσω της εντολής mn (μέσω του ορίσματος -- custom option) και εκτελούνται από την γραμμή εντολών.

Για τον έλεγχο των κόμβων και των μεταγωγέων χρησιμοποιείται το API χαμηλού επιπέδου. Αντιθέτως, για την εκκίνηση ή τη διακοπή της λειτουργίας ενός δικτύου χρησιμοποιείται το API μεσαίου επιπέδου. Η δημιουργία ενός πλήρους και λεπτομερέστατου δικτύου μπορεί να πραγματοποιηθεί με την χρήση οποιουδήποτε επιπέδου API, αλλά συνήθως επιλέγεται το μεσαίο και το υψηλό επίπεδο εξαιτίας των κλάσεων που περιέχουν και διευκολύνουν αρκετά τη δημιουργία του.

4.2.7 Εργαλεία μέτρησης

Το Mininet περιέχει εντολές-εργαλεία που καταγράφουν μετρήσεις και συμβάλουν στον έλεγχο του δικτύου και στη διαδικασία αποσφαλμάτωσης του. Μερικά από αυτά είναι:

- Εύρος ζώνης (bmw-ng, ethstats)
- Καθυστέρηση (μέσω της εντολής ping)
- Ουρές αναμονής (μέσω της εντολής tc που περιλαμβάνεται στην κλάση monitor.py)
- Στατιστικά TCP (tcp_probe)
- Χρήση CPU (top, cpuacct)

4.2.8 Mininet OpenFlow Ελεγκτές και SDN

Ένα από τα πιο ισχυρά και χρήσιμα χαρακτηριστικά του Mininet είναι ότι χρησιμοποιεί Software-Defined Networking. Με τη χρήση του OpenFlow πρωτοκόλλου επιτρέπεται ο προγραμματισμός των μεταγωγέων έτσι ώστε να μπορούν να λαμβάνουν αποφάσεις για τα πακέτα που εισέρχονται σε αυτούς.

4.2.8.1 OpenFlow Ελεγκτές (Controllers)

Εάν τρέξουμε την εντολή mn στο mininet θα χρησιμοποιηθεί σαν προεπιλεγμένος ελεγκτής ο ελεγκτής τύπου ovsc. Η αντίστοιχη ισοδύναμη εντολή είναι:

```
$ sudo mn --controller ovsc
```

Ο ελεγκτής αυτού του τύπου υλοποιεί ένα απλό μεταγωγέα μάθησης Ethernet και υποστηρίζει έως και δεκαέξι επιμέρους μεταγωγείς.

Κατά την κατασκευή ενός σεναρίου (Script) όταν εκτελεστεί η κλάση Mininet() θα πρέπει να έχει οριστεί και η αντίστοιχη κλάση του ελεγκτή. Σε περίπτωση που δεν έχει δηλωθεί από τον χρήστη κάποια συγκεκριμένη κλάση ελεγκτή, τότε καλείται η προεπιλεγμένη κλάση Controller() με αποτέλεσμα τη δημιουργία ελεγκτών τύπου Stanford OpenFlow.

Αντιθέτως παρέχεται και η δυνατότητα χρησιμοποιήσεις διαφορετικού τύπου ελεγκτή ανάλογα με τις ανάγκες της εφαρμογής. Ο χρήστης μπορεί να δημιουργήσει μία υποκατηγορία Controller() και να την μεταφέρει στα αρχεία του συστήματος του Mininet.

4.2.8.2 Απομακρυσμένοι ελεγκτές OpenFlow

Το Mininet παρέχει τη δυνατότητα σύνδεσης με ελεγκτή που εκτελείται σε κάποιο τερματικό στο τοπικό δίκτυο ή σε μία εικονική μηχανή ή ακόμα και στον ίδιο τον υπολογιστή μας.

Η κλάση RemoteController() λειτουργεί ως διαμεσολαβητής για έναν ελεγκτή ο οποίος μπορεί να λειτουργεί σε οποιοδήποτε σημείο του δικτύου ελέγχου, με την διαφορά ότι η έναρξη και η διακοπή της λειτουργίας του θα πρέπει να γίνει με τρόπο χειροκίνητο ή με κάποιο μηχανισμό που δεν ελέγχεται από το Mininet.

Παρακάτω δίνεται ένα παράδειγμα της χρήσης της κλάσης RemoteController() :

```
from functools import partial

net = Mininet(topo=topo, controller=partial(RemoteController,
ip='127.0.0.1', port=6633))
```

Όπως φαίνεται και στα παραδείγματα ο ελεγκτής, αποτελεί συνάρτηση δόμησης και όχι αντικείμενο. Παρέχεται η δυνατότητα δημιουργίας συνάρτησης δόμησης εν σειρά χρησιμοποιώντας το όρισμα partial ή lambda ή δημιουργώντας μία συνάρτηση που θα παίρνει ορίσματα και θα επιστρέφει τον ελεγκτή ως αντικείμενο. Τέλος δίνεται και η δυνατότητα εισαγωγής του ελεγκτή και ως κλάση (υποκατηγορία της κλάσης RemoteController()).

4.3 Oracle VM VirtualBox

Το VirtualBox είναι μια εφαρμογή εικονοποίησης επάνω σε πολλές πλατφόρμες (virtualization cross-platform). Αυτό σημαίνει ότι μπορούμε να το εγκαταστήσουμε σε υπολογιστές, που βασίζονται σε αρχιτεκτονικές Intel ή AMD, είτε πρόκειται για Windows, Mac, Linux ή λειτουργικά συστήματα τύπου Solaris. Επεκτείνει τις δυνατότητες του υπολογιστή μας, έτσι ώστε αυτός να μπορεί να τρέξει πολλαπλά λειτουργικά συστήματα (μέσα σε πολλαπλές εικονικές μηχανές) ταυτόχρονα. Έτσι, για παράδειγμα, μπορούμε να εκτελέσουμε Windows και Linux σε έναν Mac υπολογιστή ή να εκτελέσουμε τα Windows Server 2008 σε Linux server ή να τρέξουμε Linux σε Windows PC, και ούτω καθεξής, όλα μαζί με τις εφαρμογές που έχουμε ήδη εγκατεστημένες. Μπορούμε να εγκαταστήσουμε και να εκτελέσουμε όσες εικονικές μηχανές όπως επιθυμούμε, αφού τα μόνα πρακτικά όρια είναι ο

χώρος στο δίσκο και η μνήμη. Παρά την απλότητά του είναι ένα πολύ ισχυρό πρόγραμμα και μπορεί να τρέξει παντού, από μικρά ενσωματωμένα συστήματα ή μηχανήματα desktop κατηγορίας, αλλά επίσης να χρησιμοποιηθεί στην ανάπτυξη datacenter ή και ακόμη Cloud περιβάλλοντος.

4.3.1 Χρησιμότητα εικονικοποίησης (virtualization)

Οι τεχνικές και τα χαρακτηριστικά που παρέχει το VirtualBox είναι χρήσιμα για διάφορα σενάρια, όπως:

- **Ταυτόχρονη εκτέλεση πολλαπλών λειτουργικών συστημάτων**. Το VirtualBox επιτρέπει την εκτέλεση περισσότερων από ένα λειτουργικό σύστημα. Με αυτό τον τρόπο μπορούμε να εκτελέσουμε λογισμικά που είναι γραμμένα για έναν τύπο λειτουργικού συστήματος σε έναν άλλο (για παράδειγμα, το λογισμικό των Windows σε Linux ή Mac), χωρίς να χρειάζεται να κάνουμε επανεκκίνηση για να το χρησιμοποιήσουμε. Δεδομένου ότι μπορούμε να ρυθμίσουμε τι είδους «εικονικό» υλικό θα χρησιμοποιήσουμε για κάθε λειτουργικό σύστημα, μπορούμε να εγκαταστήσουμε ένα παλιό λειτουργικό σύστημα όπως DOS ή OS/2, ακόμα και αν το υλικό του πραγματικού υπολογιστή δεν υποστηρίζεται πλέον από το εν λόγω λειτουργικό σύστημα.
- **Ευκολότερη εγκατάσταση λογισμικού**. Οι προμηθευτές λογισμικού μπορούν να χρησιμοποιήσουν εικονικές μηχανές για ολόκληρες διαμορφώσεις λογισμικού. Για παράδειγμα, η εγκατάσταση ενός διακομιστή αλληλογραφίας σε μια πραγματική μηχανή μπορεί να είναι μια επίπονη εργασία. Με το VirtualBox μία τέτοια επίπονη εργασία (συχνά ονομάζεται "συσκευή") μπορεί να είναι μέσα σε μια εικονική μηχανή, άρα η εγκατάσταση και η λειτουργία ενός διακομιστή αλληλογραφίας γίνεται τόσο εύκολη όσο εισάγοντας μια τέτοια συσκευή στο VirtualBox.
- **Δοκιμές και αποκατάσταση μετά από καταστροφή**. Μόλις εγκατασταθεί μια εικονική μηχανή, ο εικονικός σκληρός δίσκος της μπορεί να θεωρηθεί ως ένα «κιβώτιο/container» που μπορεί αυθαίρετα να παγώσει, να ξυπνήσει, να αντιγραφεί ή και να δημιουργήσει αντίγραφα ασφαλείας. Με τη χρήση ενός άλλου χαρακτηριστικού του VirtualBox που ονομάζεται «στιγμιότυπα/ snapshots», μπορεί κανείς να σώσει μια συγκεκριμένη κατάσταση μιας εικονικής μηχανής και να επανέλθει σε αυτή την κατάσταση εάν είναι απαραίτητο. Με αυτό τον τρόπο, κάποιος μπορεί να πειραματιστεί με ένα ελεύθερο υπολογιστικό περιβάλλον. Αν κάτι πάει στραβά (π.χ. μετά την εγκατάσταση δεν λειτουργεί σωστά το λογισμικό ή μολυνθεί το λογισμικό από έναν ιό), μπορεί κανείς εύκολα να επιστρέψει σε ένα προηγούμενο στιγμιότυπο και να αποφευχθεί η ανάγκη δημιουργίας συχνών αντιγράφων ασφαλείας καθώς και η αποκατάσταση τους. Μπορεί να δημιουργηθεί οποιοσδήποτε αριθμός των στιγμιότυπων καθώς και να διαγραφούν στιγμιότυπα ενώ ένα VM τρέχει για να ανακτήσουμε χώρο στο δίσκο.
- **Υποδομή ενοποίησης**. Το Virtualization μπορεί να μειώσει σημαντικά το κόστος του υλικού και της ηλεκτρικής ενέργειας. Τις περισσότερες φορές, οι υπολογιστές σήμερα χρησιμοποιούν μόνο ένα κλάσμα των δυνατοτήτων

τους και τρέχουν με χαμηλό μέσο όρο τα φορτία του συστήματος. Με αυτόν τον τρόπο ένα μεγάλο μέρος των πόρων του υλικού, καθώς και της ηλεκτρικής ενέργειας, μπορεί να σπαταλιέται . Έτσι, αντί να έχουμε πολλούς τέτοιους φυσικούς υπολογιστές που χρησιμοποιούνται εν μέρει μόνο, μπορεί κανείς να ενοποιήσει πολλές εικονικές μηχανές σε λίγους ισχυρούς υπολογιστές και να εξισορροπήσει τα φορτία μεταξύ τους.

4.3.2 Ορολογία

Host operating system (host OS). Είναι το λειτουργικό σύστημα του φυσικού υπολογιστή στον οποίο έχει εγκατασταθεί το VirtualBox.

Guest operating system (guest OS). Είναι το λειτουργικό σύστημα που εκτελείται στο εσωτερικό της εικονικής μηχανής. Θεωρητικά το VirtualBox τρέχει οποιοδήποτε λειτουργικό σύστημα x86 (DOS, Windows, OS / 2, FreeBSD, OpenBSD), αλλά για να επιτευχθεί σχεδόν μητρική απόδοση του κώδικα του επισκέπτη (guest) στον υπολογιστή μας, θα έπρεπε να γίνουν πολλές βελτιστοποιήσεις που είναι ειδικές σε ορισμένα λειτουργικά συστήματα.

Virtual machine (VM). Το λειτουργικό σύστημα επισκεπτών τρέχει σε μία εικονική μηχανή (VM). Κανονικά, ένα VM θα εμφανίζεται ως ένα παράθυρο στην επιφάνεια εργασίας του υπολογιστή μας. Σε ένα πιο αφηρημένο τρόπο, στο εσωτερικό, το VirtualBox λειτουργεί ένα VM ως ένα σύνολο παραμέτρων που καθορίζουν τη συμπεριφορά του. Περιλαμβάνουν τις ρυθμίσεις υλικού (πόση μνήμη θα πρέπει να έχει το VM, τι σκληρούς δίσκους VirtualBox πρέπει να εικονικοποιήσει μέσω των αρχείων κοντέινερ, εάν κάποιο CD είναι τοποθετημένο κλπ) καθώς και πληροφορίες κατάστασης (εάν το VM εκτελείται αυτή τη στιγμή, εάν σώθηκε, εάν υπάρχουν τα στιγμιότυπα του κ.λπ.). Οι ρυθμίσεις αυτές αντικατοπτρίζονται επίσης στο παράθυρο του Διαχειριστή του VirtualBox το οποίο εμφανίζει τις παραπάνω λεπτομέρειες.

4.3.3 Επισκόπηση Χαρακτηριστικών

Παρακάτω παρατίθεται μια σύντομη περιγραφή των κύριων χαρακτηριστικών του VirtualBox όπως:

- **Φορητότητα**. Το VirtualBox τρέχει σε ένα μεγάλο αριθμό λειτουργικών συστημάτων υποδοχής 32-bit και 64-bit αρχιτεκτονικών και καλείται "hosted" hypervisor (μερικές φορές αναφέρεται ως hypervisor "τύπου 2"), γιατί απαιτεί ένα υπάρχον λειτουργικό σύστημα που θα εγκατασταθεί. Μπορεί έτσι να τρέξει παράλληλα με τις υπάρχουσες εφαρμογές σε έναν κεντρικό υπολογιστή. Λόγω του ότι είναι λειτουργικά πανομοιότυπο σε όλες τις πλατφόρμες και χρησιμοποιεί τις ίδιες μορφές αρχείων και εικόνων, επιτρέπει την εκτέλεση των εικονικών μηχανών που δημιουργήθηκαν σε έναν υπολογιστή σε έναν άλλο με διαφορετικό λειτουργικό σύστημα. Για παράδειγμα, μπορούμε να δημιουργήσουμε μια εικονική μηχανή στα Windows και στη συνέχεια να την εκτελέσουμε από το Linux.

- **Δεν απαιτείται εικονικοποίηση υλικού**. Για πολλά σενάρια το VirtualBox δεν απαιτεί τις δυνατότητες επεξεργαστή όπως η Intel VT-x ή AMD-V να είναι ενσωματωμένες σε ένα νεότερο υλικό επεξεργαστή. Σε αντίθεση με πολλές άλλες λύσεις virtualization, μπορούμε να χρησιμοποιήσουμε το VirtualBox ακόμα και σε παλιότερο υλικό, που αυτά τα χαρακτηριστικά δεν υπάρχουν.
- **Μεγάλη υποστήριξη υλικού**. Το VirtualBox υποστηρίζει:
    - **Guest multiprocessing (SMP).** Το VirtualBox μπορεί να παρουσιάσει έως και 32 εικονικές CPUs σε κάθε εικονική μηχανή, ανεξάρτητα από το πλήθος των πυρήνων CPU που διαθέτει το σύστημα.
    - **Υποστήριξη USB συσκευής**. Το VirtualBox υλοποιεί έναν εικονικό ελεγκτή USB και επιτρέπει να συνδέσουμε συσκευές USB στις εικονικές μηχανές χωρίς να χρειάζεται να εγκαταστήσουμε το ειδικό προγράμματα οδήγησης του USB.
    - **Συμβατότητα Υλικού**. Το VirtualBox εικονικοποιεί μια μεγάλη σειρά από εικονικές συσκευές, μεταξύ των οποίων πολλές συσκευές που συνήθως παρέχονται από άλλες πλατφόρμες virtualization. Αυτό περιλαμβάνει IDE, SCSI και SATA ελεγκτές σκληρών δίσκων, πολλές εικονικές κάρτες δικτύου και κάρτες ήχου, εικονικές σειριακές θύρες και παράλληλες.
    - **Ενσωματωμένη υποστήριξη iSCSI**. Αυτό το μοναδικό χαρακτηριστικό επιτρέπει τη σύνδεση σε μια εικονική μηχανή απευθείας σε έναν server αποθήκευσης iSCSI, χωρίς να περάσει από το σύστημα που τρέχει το virtualbox. Το VM έχει πρόσβαση στο iSCSI απευθείας, χωρίς την επιπλέον επιβάρυνση που απαιτείται για την εικονικοποίηση σκληρών δίσκων στα αρχεία κοντέινερ.
    - **PXE εκκίνηση μέσω δικτύου**. Οι ολοκληρωμένες εικονικές κάρτες δικτύου του VirtualBox υποστηρίζουν πλήρως απομακρυσμένη εκκίνηση μέσω του Execution Environment Preboot (PXE).
- **Δημιουργία πολλαπλών στιγμιότυπων (snapshots)**. Το VirtualBox μπορεί να σώσει αυθαίρετα στιγμιότυπα της κατάστασης μίας εικονικής μηχανής. Μπορούμε να επαναφέρουμε την εικονική μηχανή με οποιοδήποτε στιγμιότυπο και να αρχίσουμε μια εναλλακτική διαμόρφωση του VM από εκεί που είναι το στιγμιότυπο, δημιουργώντας έτσι ένα δενδρικό σύστημα αποτελούμενο από στιγμιότυπα.
- **Ομάδες VM**. Το VirtualBox παρέχει ένα ομαδικό χαρακτηριστικό που επιτρέπει στον χρήστη να οργανώνει και να ελέγχει τις εικονικές μηχανές συλλογικά, καθώς και μεμονωμένα. Εκτός από τις βασικές ομάδες, είναι επίσης δυνατό για κάθε VM να ομαδοποιείται σε περισσότερες από μία ομάδες. Σε γενικές γραμμές, οι λειτουργίες που μπορούν να εκτελεστούν σε ομάδες είναι οι ίδιες με αυτές που μπορεί να εφαρμοστεί στα μεμονωμένα VMs, δηλαδή Έναρξη, Παύση, Επαναφορά, Κλείσιμο, Απόρριψη αποθηκευμένης κατάστασης, Εμφάνιση στο σύστημα αρχείων και Ταξινόμηση.

4.3.4 Ρυθμίσεις δικτύου στο Virtualbox

Το Virtualbox μας δίνει την επιλογή από το παράθυρο δικτύου στις ρυθμίσεις της εικονικής μηχανής να επιλέξουμε τον τρόπο δικτύωσης που επιθυμούμε καθώς και τον τύπο του προσαρμογέα δικτύου. Παρακάτω ακολουθούν μερικοί τρόποι δικτύωσης:

- **Not attached**: Στη συγκεκριμένη λειτουργία η εικονική μηχανή θεωρεί ότι διαθέτει μία κάρτα δικτύου χωρίς όμως να υπάρχει κάποιου είδους συνδεσιμότητα, δηλαδή να μην υπάρχει κάποιο καλώδιο να τη συνδέσει με το δίκτυο.
- **Network Address Translation (NAT)**: Κατά τη λειτουργία αυτή επιτρέπονται όλες εκείνες οι διαδικασίες που επιτρέπουν στο εικονικό μηχάνημα πρόσβαση στο Internet καθώς και κάποιες απλές λειτουργίες file sharing.
- **Bridged Networking**: Στη λειτουργία αυτή η εικονική κάρτα δικτύου συνδέεται με μία κάρτα δικτύου του φυσικού μηχανήματος, παρακάμπτοντας το network stack του συστήματος.
- **Internal Networking**: Χρησιμοποιείται για τη δημιουργία δικτύων μεταξύ των εικονικών μηχανών απομονωμένων από το φυσικό μηχάνημα και τις εφαρμογές που διαχειρίζεται.
- **Host-only networking**: Η συγκεκριμένη λειτουργία επιλέγεται για τη δημιουργία ενός δικτύου αποτελούμενου από τον υπολογιστή που τρέχουμε το virtualbox και όσα εικονικά μηχανήματα επιλεγούν τη λειτουργία αυτή.

4.4 Open vSwitch

4.4.1 Εισαγωγή

Τα δίκτυα σε εικονικοποιημένα περιβάλλοντα παρουσιάζουν νέες ευκαιρίες αλλά και προβλήματα. Ωστόσο, το τυπικό διαδικτυακό μοντέλο σε αυτά τα περιβάλλοντα αποτελείται από τον κλασσικό L2 μεταγωγέα ή L3 δρομολογητή σε επίπεδο hypervisor ή στο επίπεδο διαχείρισης υλικού του εικονικού περιβάλλοντος. Αυτά τα εικονικά δικτυακά στοιχεία διαχειρίζονται την επικοινωνία μεταξύ των εικονικών μηχανών που βρίσκονται στην ίδια φυσική τοποθεσία, καθώς και την επικοινωνία με την φυσική NIC. Είναι υλοποιημένα σε software και συνήθως τοποθετούνται στο πλαίσιο του host.

Το Open vSwitch είναι ένας λογικός δικτυακός μεταγωγέας (vswitch) ειδικά προορισμένος για εικονικά περιβάλλοντα. Η διαφορά του σε σχέση με άλλες παρόμοιες προσεγγίσεις έγκειται στο γεγονός ότι παρέχει μία εξωτερική διεπαφή (interface) για τον έλεγχο της συμπεριφοράς προώθησης, ώστε να υποστηρίξει λειτουργίες QoS, tunneling και filtering κανόνων. Επίσης, υποστηρίζει μία απομακρυσμένη διεπαφή που επιτρέπει τη μετανάστευση (migration) του παραμετροποιημένου στιγμιότυπου (configuration state). Επιπλέον, η υλοποίησή του παρέχει μια ευέλικτη μηχανή προώθησης, βασισμένη σε πίνακες, που μπορεί να χρησιμοποιηθεί για τη λογική τμηματοποιήση (partition) του επιπέδου προώθησης (forwarding plane). Τέλος, έχει τη δυνατότητα συνεργασίας με τα περισσότερα Linux-based περιβάλλοντα εικονικοποίησης, όπως Xen, XenServer, KVM και QEMU.

Το βασικότερο πλεονέκτημα του συγκεκριμένου λογισμικού είναι ότι παρέχει στο χρήστη interfaces για τη διαχείριση και τον έλεγχο όλων των λειτουργιών και της κατάστασης του δικτύου κατά τη χρήση του. Τα interfaces αυτά αναλύονται ακολούθως:

- **Configuration Interface**: Μέσω αυτού ο χρήστης μπορεί να ελέγχει τυχόν αλλαγές στην κατάσταση μέσω μιας διαδικασίας (process), η οποία διαβάζει και καταγράφει τις αλλαγές αυτές μέσω ζευγαριών τιμών-κλειδιών. Ο χρήστης μπορεί επίσης να εφαρμόσει πολιτικές ποιότητας της υπηρεσίας (QoS), να ελέγξει τη ροή του δικτύου (NetFlow) και να δεσμεύει ή να αποδεσμεύει interfaces για τη βέλτιστη απόδοση και διαθεσιμότητα του δικτύου.
- **Forwarding Path**: Σε αντίθεση με τους φυσικούς μεταγωγείς, το Open vSwitch παρέχει τη δυνατότητα για παρέμβαση στους πίνακες προώθησης (forwarding tables) και συνεπώς στον προσδιορισμό μιας συγκεκριμένης λειτουργίας των πακέτων (απόρριψη, προώθηση σε συγκεκριμένη πόρτα, ενσωμάτωση) ανάλογα με το είδος της επικεφαλίδας τους (header).
- **Connectivity management**: Μέσω του συγκεκριμένου interface παρέχεται η δυνατότητα στο χρήστη να διαχειρίζεται την τοπολογία του δικτύου του. Αυτό περιλαμβάνει διαδικασίες δημιουργίας εικονικών μεταγωγέων, διαχείρισης της συνδεσιμότητας των εικονικών interfaces, καθώς και διαχείρισης των φυσικών interfaces του host.

Μερικοί από τους τρόπους που μπορεί να χρησιμοποιηθεί το Open vSwitch στα εικονικοποιημένα περιβάλλοντα είναι:

- **Centralized Management:** Το Open vSwitch όπως αναφέρθηκε παρέχει τη δυνατότητα ελέγχου των λειτουργιών του μέσω των interfaces που παρέχει. Ταυτόχρονα είναι εφικτό να δημιουργηθούν εικονικές συνδέσεις μεταξύ των interfaces αυτών σε δύο διαφορετικά φυσικά μηχανήματα. Γίνεται λοιπόν κατανοητό ότι μπορεί να υπάρξει κοινή διαχείριση όλων των μηχανημάτων που διαθέτουν το Open vSwitch από μία κεντρική μονάδα.
- **Virtual Private Networks:** Όπως ακριβώς στα φυσικά δίκτυα, μία ομάδα από μηχανήματα μπορεί να συνδέονται μεταξύ τους μέσω ενός δεσμευμένου δικτύου, έτσι και στα εικονικοποιημένα περιβάλλοντα οι εικονικές μηχανές μπορούν να δημιουργήσουν ένα εικονικό δεσμευμένο δίκτυο ένα επίπεδο πάνω από το φυσικό επίπεδο. Το Open vSwitch υποστηρίζει λειτουργίες VLAN και GRE tunneling. Πιο συγκεκριμένα το VLAN χρησιμοποιείται για μικρού μεγέθους εγκαταστάσεις δικτύων, το GRE χρησιμοποιείται για την αντιμετώπιση μεγάλου μεγέθους εγκαταστάσεων.
- **Δυνατότητα μετακίνησης μεταξύ διαφορετικών υποδικτύων:** Ένα από τα προβλήματα στις πλατφόρμες εικονικοποίησης που αντιμετωπίζει το Open vSwitch, είναι η δυνατότητα αντιγραφής και μετακίνησης των ιδιοτήτων των εικονικών δικτύων σε ένα νέο υποδίκτυο χωρίς να χρειάζεται να γίνει στο ίδιο υποδίκτυο όπως γίνεται από τις υπάρχουσες λύσεις.

4.4.2 Αρχιτεκτονική

Το Open vSwitch είναι λογισμικό που τοποθετείται στον επίπεδο του management domain (hypervisor ή σε άλλες περιπτώσεις host kernel space). Παρέχει συνδεσιμότητα μεταξύ των εικονικών μηχανών και των φυσικών interfaces. Υλοποιεί το τυπικό Ethernet switching με δυνατότητες για VLAN, RSPAN και βασικά ACL. Μπορεί να χρησιμοποιηθεί και αυτόνομα, σαν ένας standard L2 μεταγωγέας. Για να υποστηρίξει όμως, την σύνδεση με εικονικά περιβάλλοντα παρέχει διεπαφές για την διαχείριση του forwarding state και managing configuration state στο περιβάλλον εκτέλεσης.

Κάτω από το Open vSwitch βρίσκεται ένα flow-table forwarding model, παρόμοιο με αυτό που χρησιμοποιείται από το OpenFlow. Το rule-based forwarding αποσκοπεί στην επίτευξη ενός σχεδόν αυθαίρετου logical partitioning των διαδικασιών του forwarding. Πιο συγκεκριμένα, επιτρέπει την σύνδεση μεταξύ του δικτυακού configuration state και των διαδικασιών του forwarding με ένα υποσύνολο της κίνησης, είτε από ένα VM, είτε ένα σύνολο από VMs.

Το Open vSwitch συμπεριφέρεται, όπως ένας παραδοσιακός φυσικός μεταγωγέας μέσα στο επίπεδο της εικονικοποίησης. Κάθε στιγμιότυπο διαχειρίζεται ξεχωριστά διαμέσου των διεπαφών διαχείρισης, παρέχοντας ορατότητα και έλεγχο πάνω σε inter-VM επικοινωνίες, που είναι ορατές στο first hop του φυσικού μεταγωγέα. Ωστόσο, η συμπερίληψη των interfaces για καθολικό managing configuration και forwarding state επιτρέπει την κατανομή των λειτουργιών του μεταγωγέα σε πολλαπλούς servers, αποσυνδέοντας έτσι αποτελεσματικά την λογική δικτυακή τοπολογία από την φυσική. Για παράδειγμα, μια απομακρυσμένη διαδικασία, εφόσον είναι ενσωματωμένη στην πλατφόρμα ελέγχου του virtualization, μπορεί να κάνει migrate το network configuration state ταυτόχρονα με τα VMs, όπως αυτά μετακινούνται μεταξύ των φυσικών servers.

Επιπρόσθετα, η δυνατότητα της διαχείρισης του forwarding table διαμέσου ενός εξωτερικού interface επιτρέπει στην χαμηλού επιπέδου flow state να κάνει migrate μαζί με το VM. Αυτό θα ήταν χρήσιμο για την μεταφορά των υπαρχόντων flow counters και ACLs. Επίσης επιτρέπει το migration κανόνων που αφορούν το tunneling, δυνατότητα χρήσιμη στο migration μεταξύ διαφορετικών υποδικτύων IP.

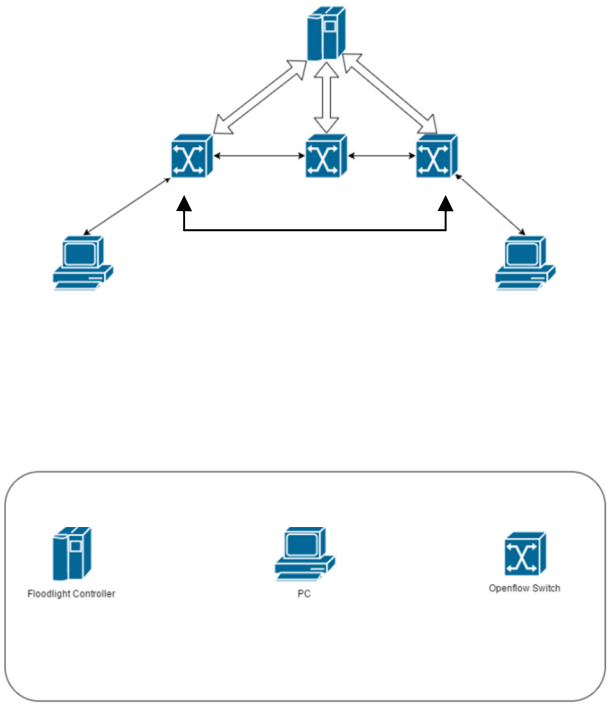
Κεφάλαιο 5:
SDN σε Περιβάλλον Σχολικών Δικτύων

Μετά από έρευνα στις ιστοσελίδες παροχής μηχανών αναζήτησης, όπως Google και Bing, δεν βρέθηκε κάποιο έγγραφο προσανατολισμένο στη χρήση του Software Defined Networking στα περιβάλλοντα Σχολικών Δικτύων. Αυτό, έχει ως αποτέλεσμα την ερευνητική προσέγγιση της υλοποίησης του SDN πάνω στο Πανελλήνιο Σχολικό Δίκτυο και μάλιστα στο πιο χαμηλό του επίπεδο, δηλαδή στη περίπτωση δικτύωσης ενός σχολείου.

5.1 Προσομοίωση Νησίδας SDN (πειραματικό στάδιο)

Στα προηγούμενα κεφάλαια παρουσιάστηκαν τα εργαλεία τα οποία μπορούν να χρησιμοποιηθούν για να υλοποιήσουν μία νησίδα SDN. Έτσι, θα ξεκινήσουμε την δημιουργία ενός εργαστηρίου OpenFlow βασισμένο στο Mininet και σαν ελεγκτή SDN των μεταγωγέων (switches) θα χρησιμοποιηθεί ο Floodlight. Στην παρακάτω

Εικόνα **16** φαίνεται η τοπολογία του δικτύου που θα προσομοιώσουμε.



Εικόνα 16: Τοπολογία Δικτύου Προσομοίωσης

5.2 Οδηγίες Εγκατάστασης

Ξεκινώντας το πειραματικό στάδιο για την προσομοίωση μίας νησίδας SDN θα πρέπει να αναφερθούν μερικοί τρόποι προσομοίωσης. Ως εκ τούτου προκύπτουν οι υλοποιήσεις σε εικονικές μηχανές (Virtual Machines - VM), που μπορούμε να τις παραμετροποιήσουμε μέσω ενός κρυπτογραφικού πρωτοκόλλου δικτύου Secure Shell ή SSH. Αυτό, για να επιτραπούν η απομακρυσμένη σύνδεση και άλλες υπηρεσίες δικτύου με ασφάλεια στις εικονικές μηχανές, μέσω ενός μη ασφαλούς δικτύου και οι υλοποιήσεις σε υπολογιστές ο οποίοι στο παρών λειτουργικό τους σύστημα τρέχουν τα προγράμματα για την προσομοίωση. Προτείνονται οι υλοποιήσεις σε εικονικές μηχανές γιατί είναι βέβαιο ότι η εγκατάσταση έχει γίνει σωστά και βέβαια δεν επηρεάζουμε τη λειτουργία του υπολογιστή που διαθέτουμε. Τέτοιες λύσεις για προσομοίωση SDN μπορούν να βρεθούν στις ακόλουθες ιστοσελίδες :

- http://sdnhub.org/tutorials/
- http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial

και άλλες σε διάφορες ιστοσελίδες που προφέρονται από μία αναζήτηση στη Google. Στο παρόν κεφάλαιο θα ακολουθήσουμε την υλοποίηση στο παρών λειτουργικό μας.

Στο μέρος αυτό θα διατυπωθούν μερικές παραδοχές για να ξεκινήσουμε το πείραμα μας. Ο υπολογιστής έχει εγκατεστημένο λειτουργικό σύστημα Linux Ubuntu έκδοση 14.04.3-desktop και τύπου 64-bit (amd64) και δικαιώματα απλού χρήστη. Για να εγκαταστήσουμε τα απαραίτητα προγράμματα θα χρησιμοποιηθεί το Terminal των Ubuntu (κονσόλα) έχοντας όμως δικαιώματα Super User (διαχειριστή) δίνοντας απλώς στο terminal του Linux (κονσόλα) την εντολή sudo –i για να γίνουμε διαχειριστές του συστήματος.

5.2.1 Mininet

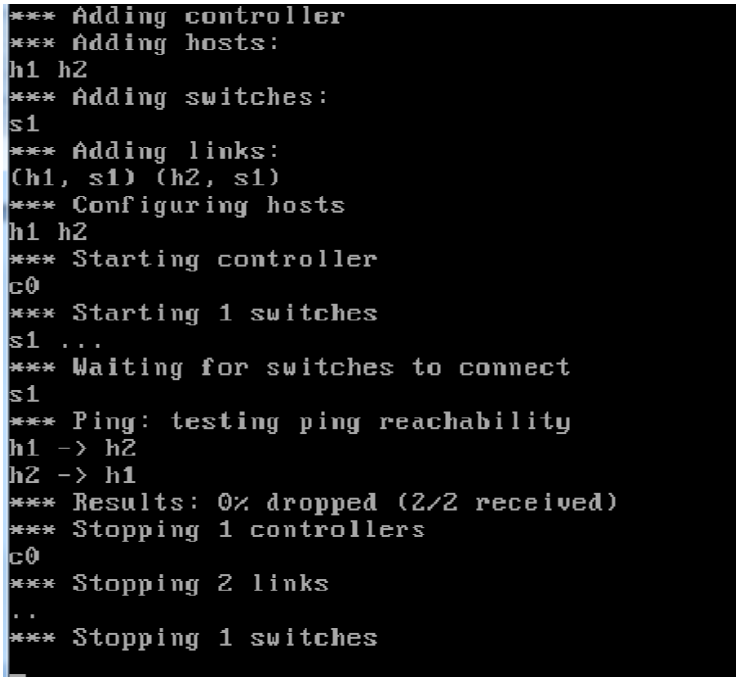
Αφού έχουμε ανοίξει το terminal του Linux (κονσόλα) χρησιμοποιούμε τις ακόλουθες εντολές για να εγκαταστήσουμε το Mininet:

```
git clone git://github.com/mininet/mininet
cd mininet
git tag
git checkout -b 2.2.1 2.2.1
cd util
install.sh –a
```

Αφού η εγκατάσταση έχει ολοκληρωθεί μπορούμε να δοκιμάσουμε την καλή λειτουργία του mininet με την ακόλουθη εντολή:

```
sudo mn --test pingall
```

και θα προκύψουν (Εικόνα 17) στην κονσόλα τα εξής:



Εικόνα 17: Αποτέλεσμα κονσόλας της εντολής mn --test pingall

5.2.2 Floodlight Controller

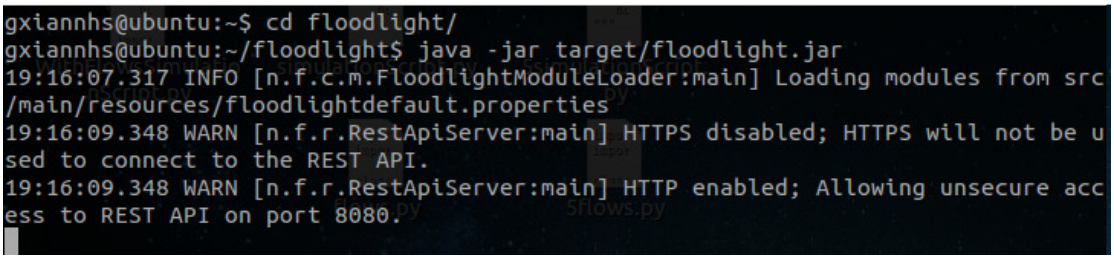
Αφού έχουμε ανοίξει το terminal του Linux (κονσόλα) χρησιμοποιούμε τις ακόλουθες εντολές για να εγκαταστήσουμε τον Floodlight controller:

```
git clone git://github.com/floodlight/floodlight.git
cd floodlight
ant
mkdir /var/lib/floodlight
chmod 777 /var/lib/floodlight
```

Αφού η εγκατάσταση έχει ολοκληρωθεί μπορούμε να ξεκινήσουμε την λειτουργία του Floodlight controller με την ακόλουθη εντολή:

```
java -jar target/floodlight.jar
```

και θα αρχίσουν να προβάλλονται και να εκτυπώνονται τα αποτελέσματα της αποσφαλμάτωσης στην κονσόλα (Εικόνα 18).



Εικόνα 18: Αποτέλεσμα κονσόλας της εντολής java -jar target/floodlight.jar

5.3 Προσομοίωση

Αφού έχουν εγκατασταθεί σωστά τα εργαλεία θα συνεχίσουμε με τη διαδικασία της προσομοίωσης.

Αρχικά, θα ξεκινήσουμε τον ελεγκτή SDN όπως ξέρουμε από τα βήματα της εγκατάστασης (Εικόνα 18) και καθώς προχωρούν τα μηνύματα στην κονσόλα μόλις εμφανιστεί η πληροφορία *INFO [n.f.c.i.OFSwitchManager:main] Listening for switch connections on 0.0.0.0/0.0.0.0:6653* (

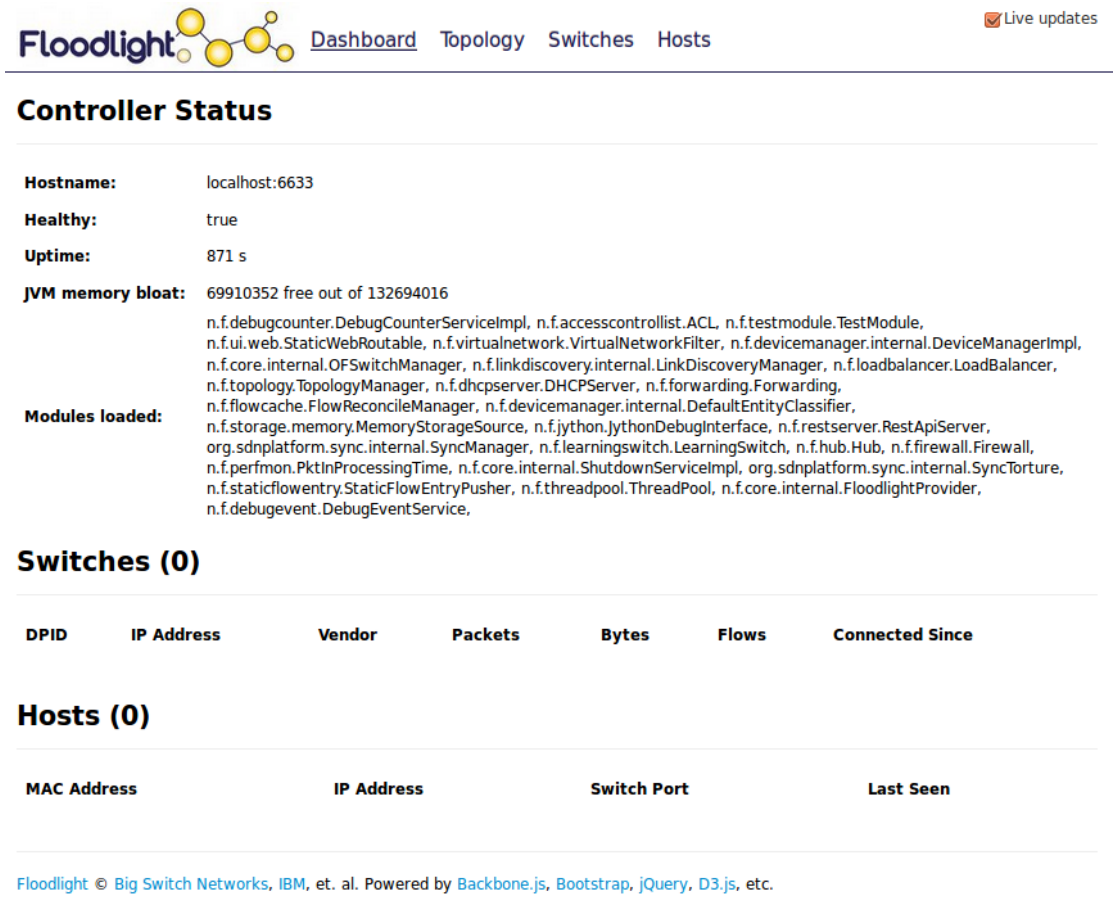
Εικόνα **19**), ο SDN controller είναι προσβάσιμος μέσω φυλλομετρητή (browser) στη διεύθυνση:

- http://127.0.0.1:8080/ui/index.html

όπως φαίνεται στην Εικόνα 20.

```
19:16:16.910 INFO [n.f.f.Forwarding:main] Default hard timeout not configured. U
sing 0.
19:16:16.910 INFO [n.f.f.Forwarding:main] Default idle timeout not configured. U
sing 5.
19:16:16.910 INFO [n.f.f.Forwarding:main] Default priority not configured. Using
 1.
19:16:16.910 INFO [n.f.f.Forwarding:main] Default flags will be empty.
19:16:16.910 INFO [n.f.f.Forwarding:main] Default flow matches set to: VLAN=true
, MAC=true, IP=true, TPPT=true
19:16:16.910 INFO [n.f.f.Forwarding:main] Not flooding ARP packets. ARP flows wi
ll be inserted for known destinations
19:16:17.293 INFO [o.s.s.i.c.FallbackCCProvider:main] Cluster not yet configured
; using fallback local configuration
19:16:17.293 INFO [o.s.s.i.SyncManager:main] [32767] Updating sync configuration
 ClusterConfig [allNodes={32767=Node [hostname=localhost, port=6642, nodeId=3276
7, domainId=32767]}, authScheme=CHALLENGE_RESPONSE, keyStorePath=/etc/floodlight
/auth_credentials.jceks, keyStorePassword is unset]
19:16:17.486 INFO [o.s.s.i.r.RPCService:main] Listening for internal floodlight
RPC on localhost/127.0.0.1:6642
19:16:17.585 INFO [n.f.c.i.OFSwitchManager:main] Listening for switch connection
s on 0.0.0.0/0.0.0.0:6653
19:16:17.595 INFO [n.f.l.i.LinkDiscoveryManager:main] Setting autoportfast featu
re to OFF
```

Εικόνα 19: Πληροφορία για σύνδεση στον ελεγκτή SDN



Εικόνα 20: Floodlight Controller

Σε αυτό το σημείο θα χρησιμοποιηθεί το Python API που προσφέρεται από το mininet και δημιουργούμε το παρακάτω πρόγραμμα σε γλώσσα python ώστε να μην χρειάζεται να τρέχουμε εντολές στο mininet CLI.

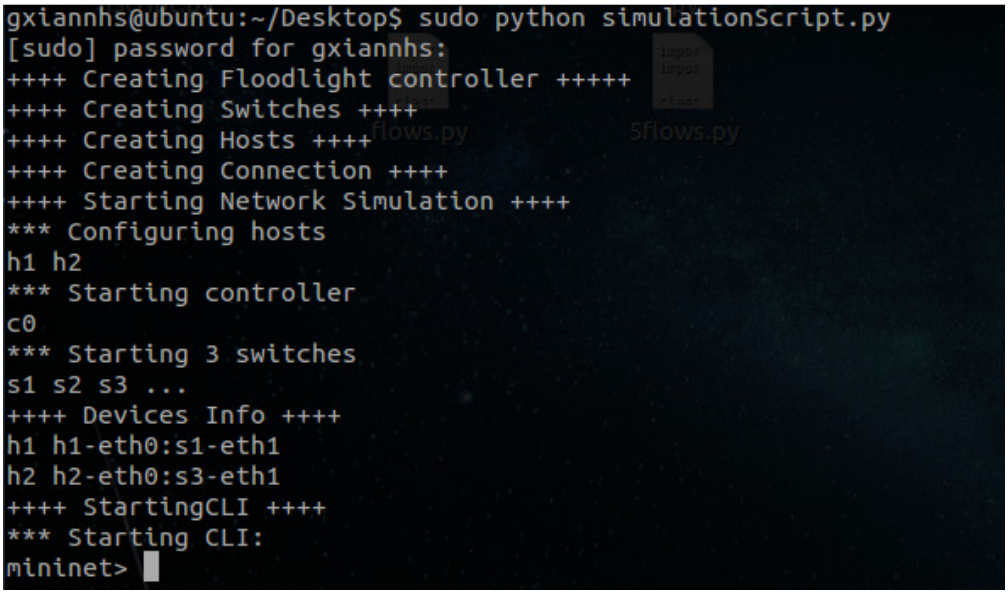
```
"""Custom topology
 host --- switch --- switch --- switch --- host
"""

from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.net import Mininet
from mininet.node import OVSSwitch, Controller, RemoteController
from mininet.util import dumpNodeConnections
```

```
def myNet():


   "Create a network with 3 switches and 2 pcs"
   net = Mininet(controller=Controller)


   info('++++ Creating Floodlight controller +++++\n')
   net.addController('c0',controller=RemoteController,
ip="192.168.2.6", port=6653)


   info('++++ Creating Switches ++++\n')
   s1 = net.addSwitch('s1', protocols='OpenFlow13')
   s2 = net.addSwitch('s2', protocols='OpenFlow13')
   s3 = net.addSwitch('s3', protocols='OpenFlow13')


   info('++++ Creating Hosts ++++\n')
   h1 = net.addHost('h1', ip="10.0.1.1/24")
   h2 = net.addHost('h2', ip="10.0.1.2/24")


   info('++++ Creating Connection ++++\n')
   net.addLink(h1, s1)
   net.addLink(h2, s3)
   net.addLink(s1, s2)
   net.addLink(s2, s3)
   net.addLink(s1, s3)


   info('++++ Starting Network Simulation ++++\n')
   net.start()


   info('++++ Devices Info ++++\n')
   dumpNodeConnections(net.hosts)


   info('++++ StartingCLI ++++\n')
   CLI(net)


   info('++++ Stopping Network ++++\n')
   net.stop()


setLogLevel('info')
myNet()
```

Το παραπάνω πρόγραμμα δημιουργεί μία τοπολογία στο Mininet αποτελούμενη από 2 τελικούς χρήστες (hosts) και 3 μεταγωγείς (switches). Μπορούμε να το τρέξουμε από την κονσόλα με τον εντολή:

```
sudo python simulationScript.py
```

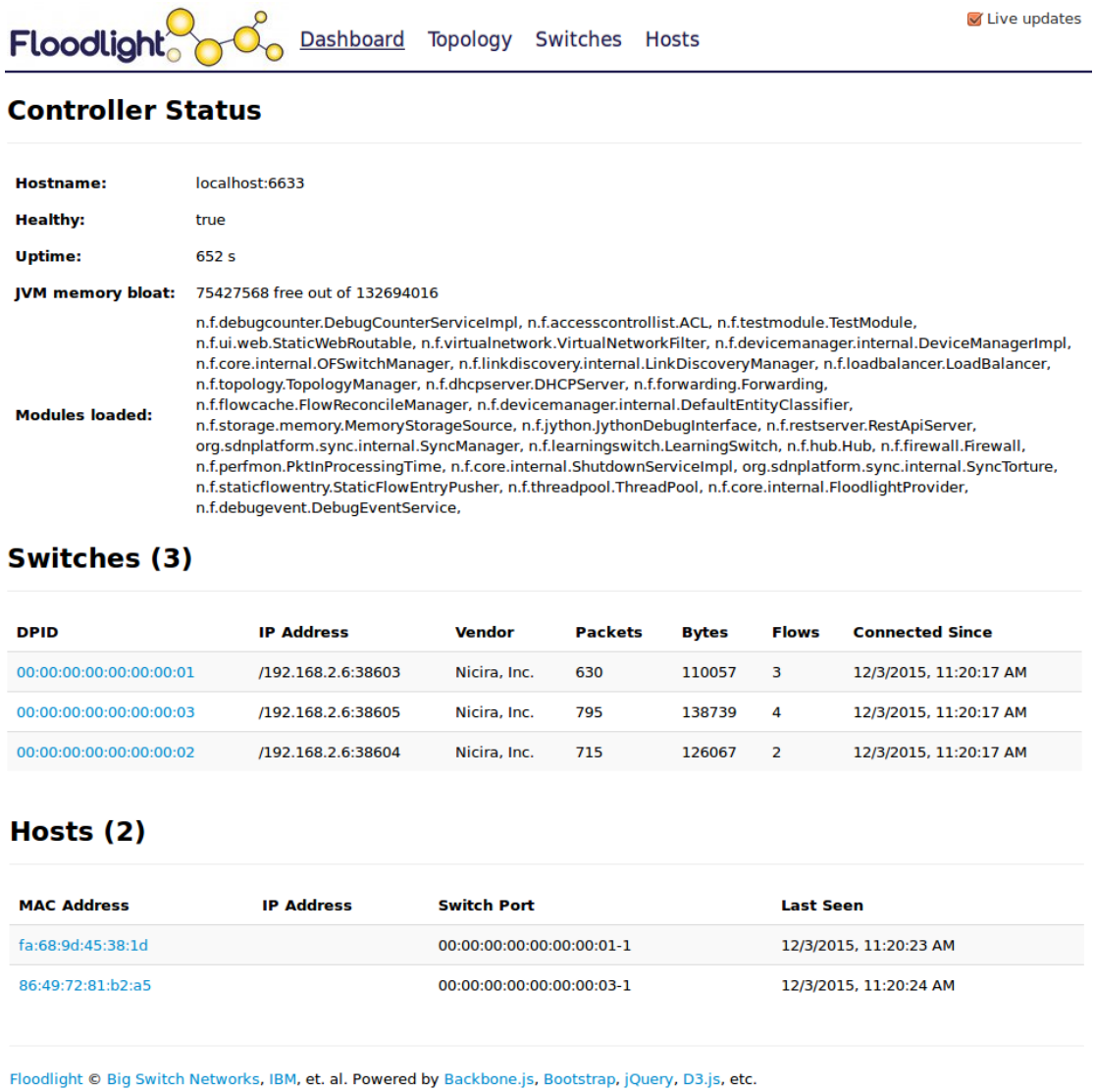
και θα προκύψουν (Εικόνα 21) τα εξής:



Εικόνα 21: Αποτελέσματα κονσόλας της εντολής sudo python simulationScript.py

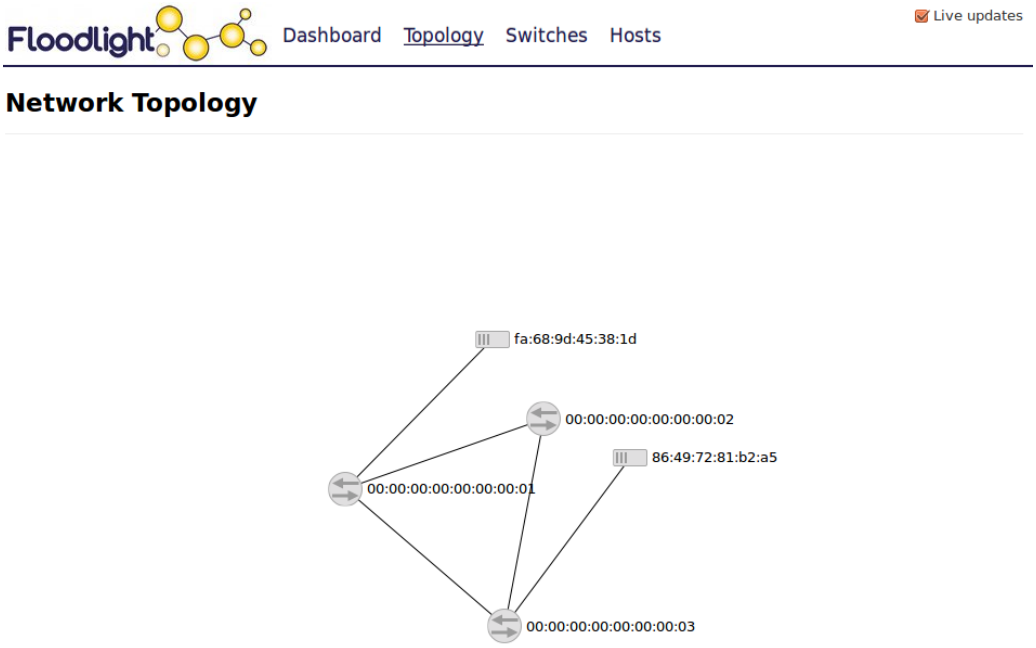
Μετά την εκτέλεση του python script μπορούμε να δούμε στον φυλλομετρητή / browser τις πληροφορίες που προκύπτουν από το παραπάνω πρόγραμμα δηλαδή τους 2 τελικούς χρήστες (hosts) και οι 3 μεταγωγείς (switches) όπως φαίνεται και στην

Εικόνα **22**:

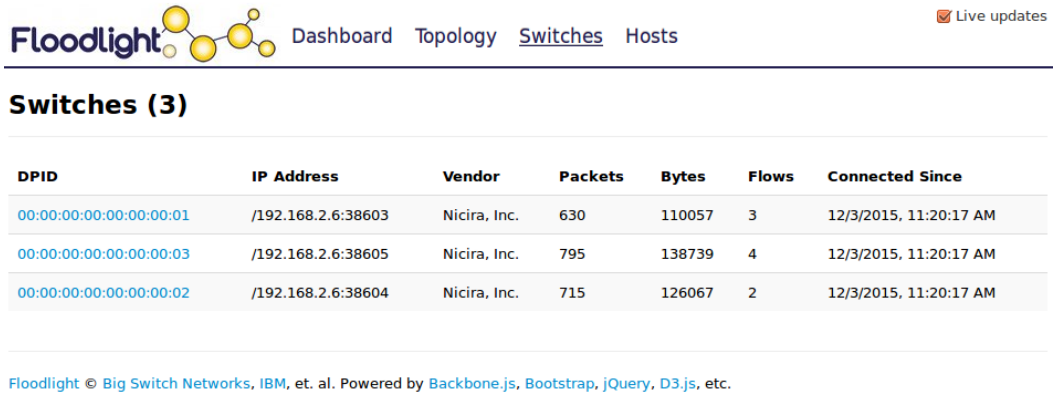


Εικόνα 22: Floodlight controller Dashboard

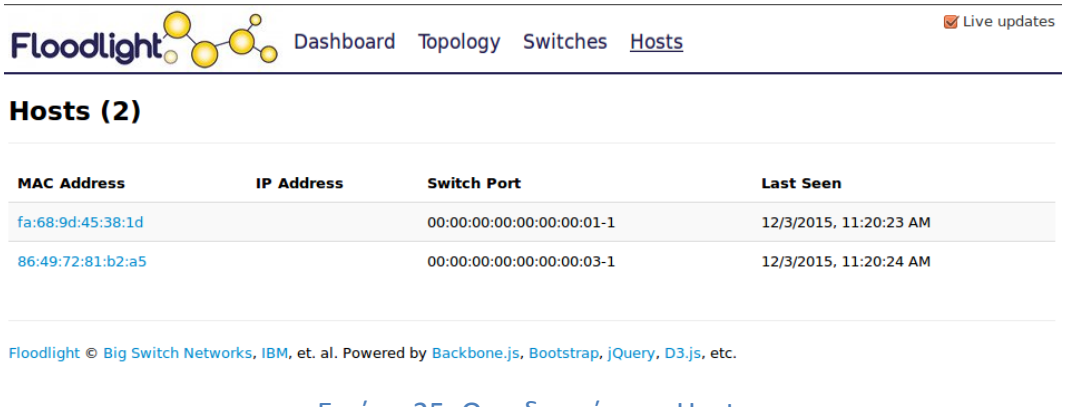
Κάνοντας μία περιήγηση μέσα στον φυλλομετρητή μπορούμε να δούμε στην καρτέλα “Topology” την τοπολογία του δικτύου, η οποία επιβεβαιώνει την ύπαρξη των επιθυμητών διαδρομών (Εικόνα 23), στην καρτέλα “Switches” τους μεταγωγείς του δικτύου, η οποία επιβεβαιώνει την ύπαρξη των επιθυμητών μεταγωγέων (Εικόνα 24), στην καρτέλα “Hosts” τους hosts του δικτύου, η οποία επιβεβαιώνει την ύπαρξη των επιθυμητών hosts (Εικόνα 25) και τις πληροφορίες τους εάν επιλέξουμε τον κάθε host με κλικ στην διεύθυνση MAC υλικού του (Εικόνα 26).



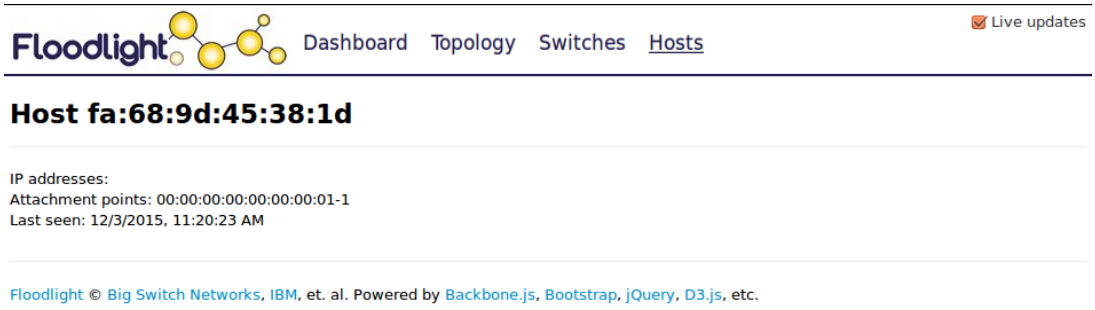
Εικόνα 23: Τοπολογία Δικτύου



Εικόνα 24: Μεταγωγείς – Switches



Εικόνα 25: Οικοδεσπότες – Hosts



Εικόνα 26: Πληροφορίες Host

Σε αυτό το σημείο το δίκτυο δεν έχει κάποια επικοινωνία, δηλαδή οι 2 τελικοί χρήστες (hosts) δεν επικοινωνούν μεταξύ τους και αυτό είναι λογικό σύμφωνα με το θεωρητικό υπόβαθρο που έχει καλυφθεί στα πρώτα κεφάλαια, επιβεβαιώνοντας το ότι οι μεταγωγείς (switches) δεν έχουν κάποια λογική αφού το Επίπεδο Ελέγχου το έχει ο SDN ελεγκτής και αυτά έχουν το Επίπεδο Δεδομένων.

Για να επικοινωνήσουν οι 2 τελικοί χρήστες (hosts) θα πρέπει να δώσουμε τις ροές της κίνησης (flows) στα switches μέσω του Controller (ελεγκτή SDN). Αυτό μπορεί να γίνει με τη εντολής curl, που μπορούμε να τη δώσουμε στην κονσόλα και θα χρησιμοποιήσουμε REST API στον Controller για να μπορέσουμε να περάσουμε τις ροές στους μεταγωγείς. Το Static Flow Pusher είναι αυτό που μας επιτρέπει να εισάγουμε χειροκίνητα ροές σε ένα δίκτυο OpenFlow και είναι προσβάσιμο μέσω ενός REST API. Για να προστεθεί μια στατική ροή, ο χρήστης πρέπει να καθορίσει τη ροή σε μορφή JSON. Για παράδειγμα, για να εισαγάγουμε μια ροή στο switch 1 που λαμβάνει τα πακέτα από τη θύρα 1 και ορίζει εξόδους τους τη θύρα 2, μπορούμε να συνθέσουμε το string JSON και απλά να χρησιμοποιήσετε μια εντολή curl για να στείλουμε το αίτημα POST μέσω HTTP στον ελεγκτή όπως φαίνεται παρακάτω:

```
curl -d '{"switch": "00:00:00:00:00:00:00:01", "name":"flow-mod-1",
"cookie":"0", "priority":"32768", "in_port":"1", "active":"true",
"actions":"output=2"}'
http://<controller_ip>:8080/wm/staticflowpusher/json
```

Για να μην χρειάζεται να τρέχουμε τις εντολές αυτές ξεχωριστά κάθε φορά στην κονσόλα δημιουργούμε το παρακάτω πρόγραμμα σε γλώσσα python ώστε να κάνει την εισαγωγή των ροών αυτόματα.

```
import httplib
import json
```

```
class StaticFlowPusher(object):

 def __init__(self, server):
 self.server = server

 def get(self, data):
 ret = self.rest_call({}, 'GET')
 return json.loads(ret[2])

 def set(self, data):
 ret = self.rest_call(data, 'POST')
 return ret[0] == 200

 def remove(self, objtype, data):
 ret = self.rest_call(data, 'DELETE')
 return ret[0] == 200

 def rest_call(self, data, action):
 path = '/wm/staticflowpusher/json'
 headers = {
 'Content-type': 'application/json',
 'Accept': 'application/json',
 }
 body = json.dumps(data)
 conn = httplib.HTTPConnection(self.server, 8080)
 conn.request(action, path, body, headers)
 response = conn.getresponse()
 ret = (response.status, response.reason, response.read())
 print ret
 conn.close()
 return ret

pusher = StaticFlowPusher('192.168.2.6')

flow1 = {
       "switch": "00:00:00:00:00:00:00:01",
       "name":"flow-mod-1-1",
       "cookie":"0",
       "priority":"1",
```

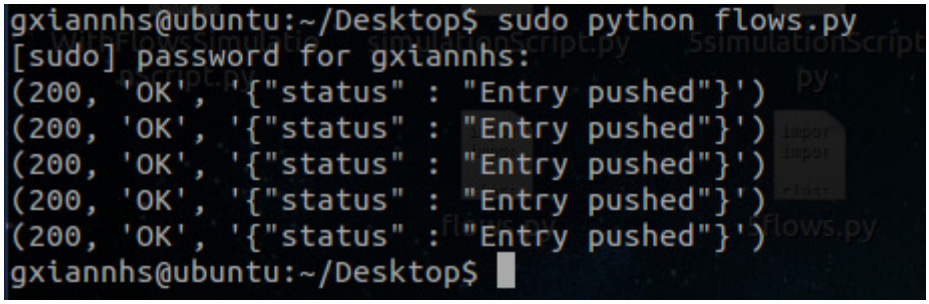
```
       "in_port":"1",
       "src-ip":"10.0.1.1",
       "dst-ip":"10.0.1.2",
       "active":"true",
       "actions":"output=2"
 }

flow2 = {
       "switch": "00:00:00:00:00:00:00:02",
       "name":"flow-mod-2-1",
       "cookie":"0",
       "priority":"1",
       "in_port":"1",
       "src-ip":"10.0.1.1",
       "dst-ip":"10.0.1.2",
       "active":"true",
       "actions":"output=2"
 }

flow3 = {
       "switch": "00:00:00:00:00:00:00:03",
       "name":"flow-mod-3-1",
       "cookie":"0",
       "priority":"1",
       "in_port":"2",
       "src-ip":"10.0.1.1",
       "dst-ip":"10.0.1.2",
       "active":"true",
       "actions":"output=1"
 }

flow4 = {
       "switch": "00:00:00:00:00:00:00:03",
       "name":"flow-mod-3-2",
       "cookie":"0",
       "priority":"1",
       "in_port":"1",
       "src-ip":"10.0.1.2",
       "dst-ip":"10.0.1.1",
```

```
        "active":"true",
        "actions":"output=3"
 }

flow5 = {
        "switch": "00:00:00:00:00:00:00:01",
        "name":"flow-mod-1-2",
        "cookie":"0",
        "priority":"1",
        "in_port":"3",
        "src-ip":"10.0.1.2",
        "dst-ip":"10.0.1.1",
        "active":"true",
        "actions":"output=1"
 }

pusher.set(flow1)
pusher.set(flow2)
pusher.set(flow3)
pusher.set(flow4)
pusher.set(flow5)
```

Το παραπάνω πρόγραμμα με την εντολή *pusher=StaticFlowPusher ('192.168.2.6')*, όπου το 192.168.2.6 αποτελεί τη ip διεύθυνση του υπολογιστή που τρέχει τον ελεγκτή θα στείλει τις ροές στα 3 switches (μεταγωγείς) που αποτελούν την τοπολογία και έτσι θα επικοινωνούν οι 2 hosts. Μπορούμε να το τρέξουμε από την κονσόλα με τον εντολή:

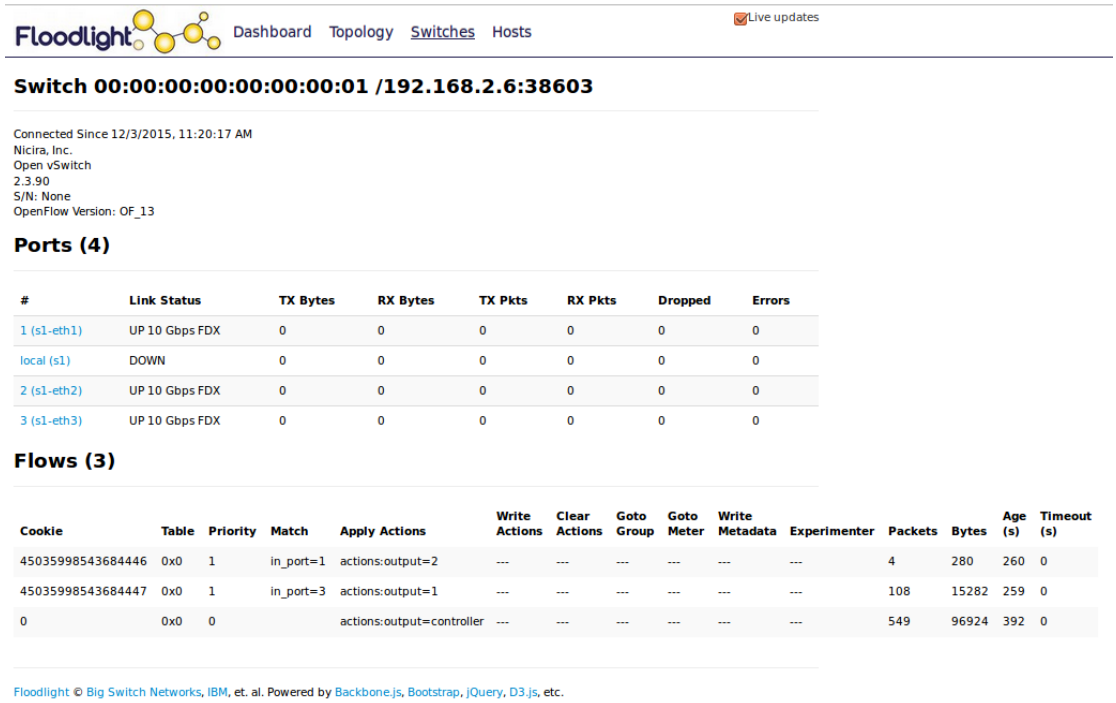
```
sudo python flows.py
```

και θα προκύψουν (Εικόνα 27) στην κονσόλα τα εξής:



Εικόνα 27:Αποτελέσματα κονσόλας της εντολής sudo python flows.py

Στη συνέχεια για να επιβεβαιώσουμε την εισαγωγή των ροών στον ελεγκτή αρκεί να μεταβούμε στην καρτέλα “Switches” η οποία επιβεβαιώνει την ύπαρξη των επιθυμητών μεταγωγέων και να δούμε τις πληροφορίες τους εάν επιλέξουμε το κάθε switch με κλικ στη διεύθυνση MAC υλικού του (Εικόνα 28). Από αυτές τις πληροφορίες προκύπτουν το όνομα του switch, το πότε συνδέθηκε στο δίκτυο, το πρωτόκολλο OpenFlow που τρέχει, την κατάσταση των θυρών που έχει συνδεδεμένες και τις ροές που υποστηρίζει.



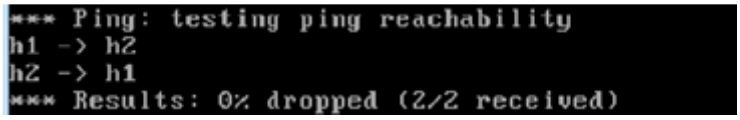
Εικόνα 28: Πληροφορίες Switch

Στην ενότητα των flows βλέπουμε τις 2 ροές που δώσαμε μέσω static flow pusher και μία ροή η οποία δεν έχει cookie. Αυτή είναι η ροή που στέλνει μηνύματα τύπου PACKET_IN από τον μεταγωγέα στον ελεγκτή.

Επιβεβαιώνουμε και τη σωστή χρήση των ροών αφού τρέξουμε στο Mininet CLI την εντολή:

```
pingall
```

και θα προκύψουν (Εικόνα 29) στην κονσόλα τα εξής:



Εικόνα 29: Αποτελέσματα κονσόλας της εντολής mininet CLI-> pingall

Σε αυτό το σημείο έχει τελειώσει το πειραματικό στάδιο έχοντας προσφέρει μία λεπτομερή αναφορά ενός εργαστηρίου OpenFlow βασισμένο στο Mininet και πως αυτό αλληλεπιδρά με έναν απομακρυσμένο ελεγκτή SDN (Floodlight).

Κεφάλαιο 6:
Προσομοίωση πραγματικού σχολικού δικτύου

Ως μέρος από το Πανελλήνιο Σχολικό Δίκτυο θα υλοποιήσουμε σε περιβάλλον εικονικοποίησης την περίπτωση των σχολικών εργαστηρίων. Αυτά γενικά βρίσκονται στο ίδιο δίκτυο με το γραφείο του Διευθυντή και των εκπαιδευτικών.

6.1 Επιλογή εργαλείων

Για την προσομοίωση του πραγματικού δικτύου θα χρειαστούμε κάποια εργαλεία που θα μας δώσουν τη δυνατότητα να υλοποιήσουμε το δίκτυο. Για αυτό το λόγο θα γίνει η χρήση του Virtualbox, που είναι ένα περιβάλλον που μας επιτρέπει να δημιουργήσουμε εινονικές μηχανές και να περιγράψουμε το πείραμα της προσομοίωσης. Επίσης, για την περίπτωση του λογικού μεταγωγέα θα επιλέξουμε το Open vSwitch, που έχουμε αναλύσει στο κεφάλαιο «Επιλογή των εργαλείων». Επίσης, θα χρησιμοποιήσουμε το floodlight controller σαν ελεγκτή του λογικού μεταγωγέα πού έχουμε αναφέρει στο προηγούμενο κεφάλαιο.

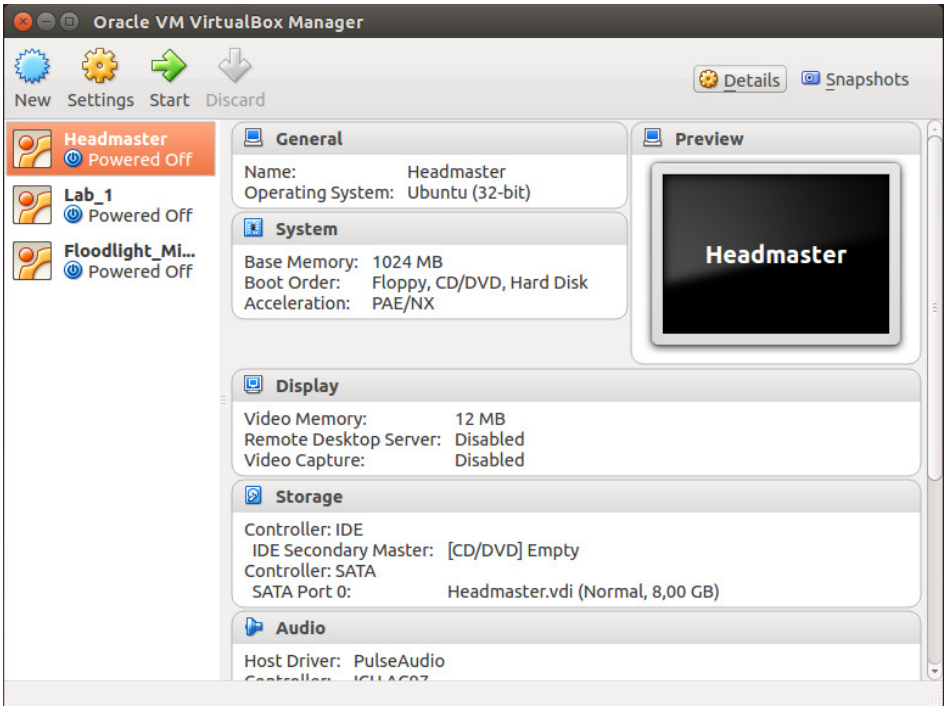
6.2 Οδηγίες Εγκατάστασης

6.2.1 Virtualbox

Για να εγκαταστήσουμε το Virtualbox στα Ubuntu θα πρέπει να ανοίξουμε τη κονσόλα των linux και να εκτελέσουμε τις ακόλουθες εντολές:

- sudo sh -c 'echo "deb http://download.virtualbox.org/virtualbox/debian trusty contrib" >> /etc/apt/sources.list'
- wget http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc -O- | sudo apt-key add –
- sudo apt-get update
- sudo apt-get install virtualbox-4.3
- sudo apt-get install virtualbox-dkms

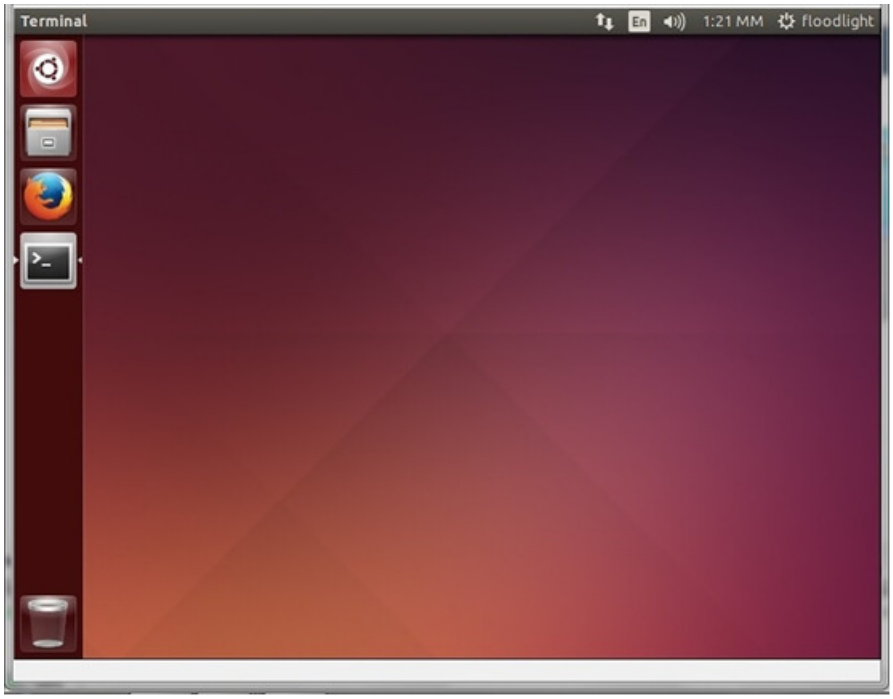
Αφού γίνει σωστά η εγκατάσταση μπορούμε να το εκτελέσουμε και να έχουμε την εξής εικόνα:



Για τη δημιουργία μίας καινούριας εικονικής μηχανής πατάμε το New πάνω αριστερά και ακολουθούμε τα απαραίτητα βήματα για την ολοκλήρωση της διαδικασίας.

6.2.2 Floodlight Controller

Ακολουθεί η περιγραφή των βημάτων εγκατάστασης και εκτέλεσης του Floodlight Controller.



Αφού έχουμε ανοίξει το terminal του Linux (κονσόλα) χρησιμοποιούμε τις ακόλουθες εντολές για να εγκαταστήσουμε τον Floodlight controller:

- git clone git://github.com/floodlight/floodlight.git
- cd floodlight
- ant
- mkdir /var/lib/floodlight
- chmod 777 /var/lib/floodlight

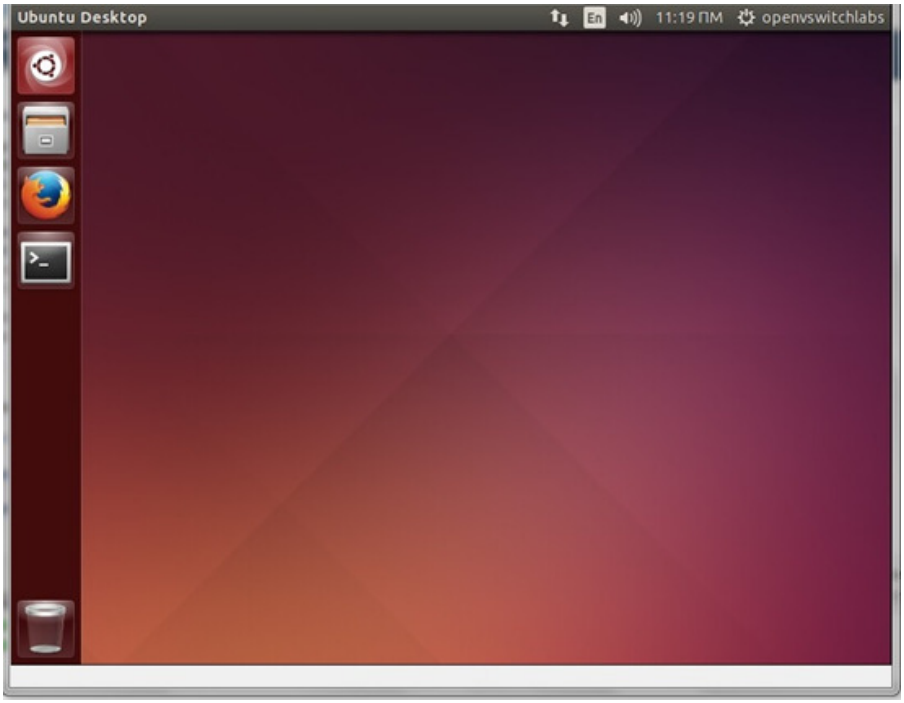
Αφού η εγκατάσταση έχει ολοκληρωθεί μπορούμε να ξεκινήσουμε τη λειτουργία του Floodlight controller με την ακόλουθη εντολή:

- java -jar target/floodlight.jar

```
floodlight@floodlight:~$ cd floodlight/
floodlight@floodlight:~/floodlight$ java -jar target/floodlight.jar
13:21:38.500 INFO [n.f.c.m.FloodlightModuleLoader:main] Loading modules from src
/main/resources/floodlightdefault.properties
13:21:39.026 WARN [n.f.r.RestApiServer:main] HTTPS disabled; HTTPS will not be u
sed to connect to the REST API.
13:21:39.026 WARN [n.f.r.RestApiServer:main] HTTP enabled; Allowing unsecure acc
ess to REST API on port 8080.
```

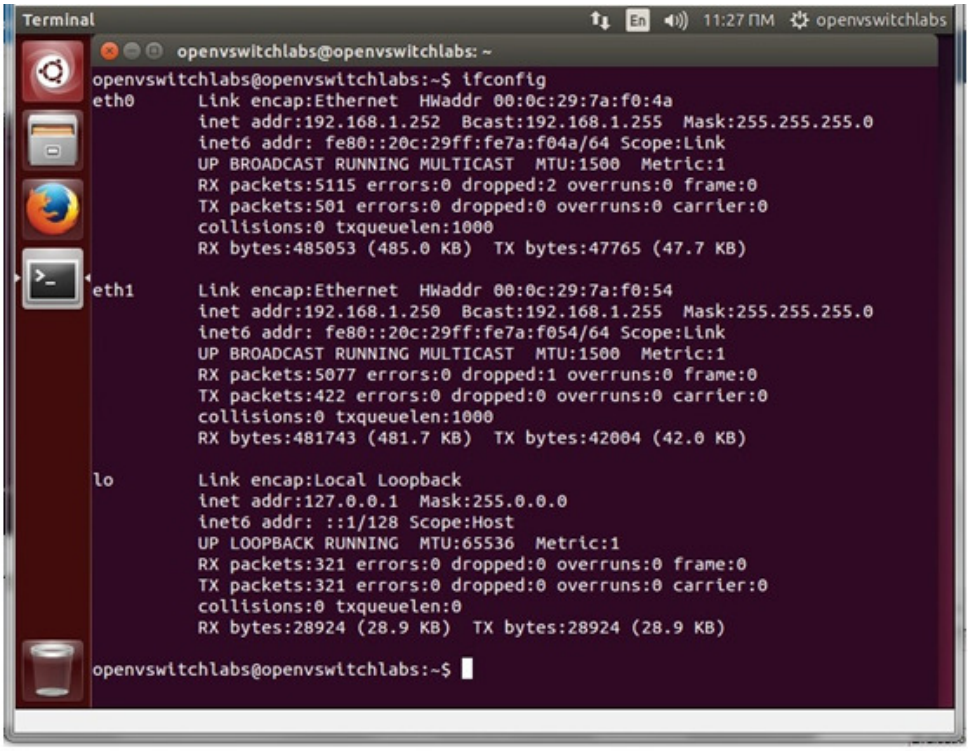
6.2.3 OpenVSwitch

Αφού έχουμε μπει στη αρχική οθόνη των ubuntu θα δείξουμε τα βήματα εγκατάστασης και θα εκτελέσουμε το OpenVSwitch.



Θα δούμε τις ρυθμίσεις των του δικτύου του με την εντολή:

- ifconfig



Εδώ θα αναφέρουμε ότι έχει 2 κάρτες δικτύου και θα χρησιμοποιήσουμε την κάρτα eth0 για το SDN και τη κάρτα eth1 για να βγαίνουμε στο internet.

Για την εγκατάστασή του OpenVSwitch χρειάζονται πρώτα να καλύψουμε κάποιες εξαρτήσεις του linux και έτσι προβαίνουμε στις παρακάτω εντολές:

Πρώτα εκτελούμε την παρακάτω εντολή με σκοπό την ενημέρωση των τρεχόντων λιστών του λογισμικού που έχουμε για να μπορέσουμε να εγκαταστήσουμε τις τελευταίες εκδόσεις των προγραμμάτων που θα χρειαστούμε.

- apt-get update

```
openvswitchlabs@openvswitchlabs:~$ sudo apt-get update
[sudo] password for openvswitchlabs:
Ign http://gr.archive.ubuntu.com trusty InRelease
Get:1 http://gr.archive.ubuntu.com trusty-updates InRelease [64,4 kB]
Ign http://extras.ubuntu.com trusty InRelease
Get:2 http://security.ubuntu.com trusty-security InRelease [64,4 kB]
Hit http://gr.archive.ubuntu.com trusty-backports InRelease
Get:3 http://extras.ubuntu.com trusty Release.gpg [72 B]
Hit http://gr.archive.ubuntu.com trusty Release.gpg
Hit http://gr.archive.ubuntu.com trusty Release
Get:4 http://extras.ubuntu.com trusty Release [11,9 kB]
Get:5 http://gr.archive.ubuntu.com trusty-updates/main Sources [247 kB]
Get:6 http://security.ubuntu.com trusty-security/main Sources [100 kB]
Get:7 http://extras.ubuntu.com trusty/main Sources [14 B]
Get:8 http://gr.archive.ubuntu.com trusty-updates/restricted Sources [5359 B]
Get:9 http://gr.archive.ubuntu.com trusty-updates/universe Sources [144 kB]
Get:10 http://extras.ubuntu.com trusty/main amd64 Packages [14 B]
Get:11 http://security.ubuntu.com trusty-security/restricted Sources [4035 B]
Get:12 http://security.ubuntu.com trusty-security/universe Sources [31,9 kB]
Get:13 http://extras.ubuntu.com trusty/main i386 Packages [14 B]
Get:14 http://gr.archive.ubuntu.com trusty-updates/multiverse Sources [5162 B]
Get:15 http://gr.archive.ubuntu.com trusty-updates/main amd64 Packages [664 kB]
Get:16 http://security.ubuntu.com trusty-security/multiverse Sources [2345 B]
Get:17 http://gr.archive.ubuntu.com trusty-updates/restricted amd64 Packages [15
,9 kB]
Get:18 http://security.ubuntu.com trusty-security/main amd64 Packages [381 kB]
79% [Waiting for headers] [18 Packages 15,5 kB/381 kB 4%] [Waiting for headers]
```

Στη συνέχεια θα εγκαταστήσουμε τις υπόλοιπες εξαρτήσεις.

- apt-get install -y git python-simplejson python-qt4 python-twisted-conch automake autoconf gcc uml-utilities libtool build-essential git pkg-config

```
openvswitchlabs@openvswitchlabs:~$ sudo apt-get install python-simplejson python
-qt4 python-twisted-conch automake autoconf gcc uml-utilities libtool build-esse
ntial pkg-config
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc is already the newest version.
pkg-config is already the newest version.
pkg-config set to manually installed.
python-qt4 is already the newest version.
python-qt4 set to manually installed.
The following extra packages will be installed:
  autotools-dev dpkg-dev fakeroot g++ g++-4.8 libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libdpkg-perl libfakeroot
  libjs-jquery libltdl-dev libsigsegv2 libstdc++-4.8-dev m4 python-pyasn1
Suggested packages:
  autoconf2.13 autoconf-archive gnu-standards autoconf-doc debian-keyring
  g++-multilib g++-4.8-multilib gcc-4.8-doc libstdc++6-4.8-dbg
  javascript-common libtool-doc libstdc++-4.8-doc automaken gfortran
  fortran95-compiler gcj-jdk user-mode-linux
The following NEW packages will be installed:
  autoconf automake autotools-dev build-essential dpkg-dev fakeroot g++
  g++-4.8 libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libfakeroot libjs-jquery libltdl-dev libsigsegv2
  libstdc++-4.8-dev libtool m4 python-pyasn1 python-simplejson
  python-twisted-conch uml-utilities
The following packages will be upgraded:
  libdpkg-perl
1 upgraded, 22 newly installed, 0 to remove and 203 not upgraded.
Need to get 19,0 MB of archives.
After this operation, 48,4 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

- linux-headers-`uname -r`

```
openvswitchlabs@openvswitchlabs:~$ uname -r
3.19.0-25-generic
openvswitchlabs@openvswitchlabs:~$ sudo apt-get install -y linux-headers-`uname
-r`
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Αφού έχουμε ενημερώσει το σύστημά μας, θα κατεβάσουμε το OpenVSwitch με την ακόλουθη εντολή:

- wget http://openvswitch.org/releases/openvswitch-2.4.0.tar.gz

και θα αποσυμπιέσουμε το αρχείο που κατεβάσαμε με την εντολή:

- tar -xzvf openvswitch-2.4.0.tar.gz

```
openvswitchlabs@openvswitchlabs:~$ wget http://openvswitch.org/releases/openvswi
tch-2.4.0.tar.gz
--2015-12-05 12:15:11--  http://openvswitch.org/releases/openvswitch-2.4.0.tar.g
z
Resolving openvswitch.org (openvswitch.org)... 69.56.251.103
Connecting to openvswitch.org (openvswitch.org)|69.56.251.103|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4182154 (4,0M) [application/x-gzip]
Saving to: 'openvswitch-2.4.0.tar.gz'

100%[======================================>] 4182154     1,09MB/s   in 3,7s

2015-12-05 12:15:16 (1,09 MB/s) - 'openvswitch-2.4.0.tar.gz' saved [4182154/4182
154]

openvswitchlabs@openvswitchlabs:~$ ls
Desktop    Downloads         Music                    Pictures  Templates
Documents  examples.desktop  openvswitch-2.4.0.tar.gz  Public    Videos
openvswitchlabs@openvswitchlabs:~$ tar -xzvf openvswitch-2.4.0.tar.gz
```

Αφού τελειώσει η διαδικασία της αποσυμπίεσης θα μπούμε στο φάκελο με την εξής εντολή:

- cd openvswitch

και θα συνεχίσουμε με την εγκατάσταση του OpenVSwitch. Παρακάτω δίνονται οι εντολές για την εγκατάστασή του:

- ./boot.sh
- ./configure --with-linux=/lib/modules/`uname -r`/build
- sudo –i (από αυτό το σημείο θα πρέπει να έχουμε δικαιώματα Super User στο Linux και έτσι θα χρησιμοποιήσουμε την εντολή sudo –i για να γίνουμε root χρήστης.)
- make && make install

```
openvswitchlabs@openvswitchlabs:~$ cd openvswitch-2.4.0/
openvswitchlabs@openvswitchlabs:~/openvswitch-2.4.0$ ./boot.sh
libtoolize: putting auxiliary files in AC_CONFIG_AUX_DIR, `build-aux'.
libtoolize: copying file `build-aux/ltmain.sh'
libtoolize: putting macros in AC_CONFIG_MACRO_DIR, `m4'.
libtoolize: copying file `m4/libtool.m4'
libtoolize: copying file `m4/ltoptions.m4'
libtoolize: copying file `m4/ltsugar.m4'
libtoolize: copying file `m4/ltversion.m4'
libtoolize: copying file `m4/lt~obsolete.m4'
openvswitchlabs@openvswitchlabs:~/openvswitch-2.4.0$ uname -r
3.19.0-25-generic
openvswitchlabs@openvswitchlabs:~/openvswitch-2.4.0$ sudo ./configure --with-lin
ux=/lib/modules/`uname -r`/build
```

```
openvswitchlabs@openvswitchlabs:~$ sudo -i
root@openvswitchlabs:~# cd /home/openvswitchlabs/openvswitch-2.4.0/
root@openvswitchlabs:/home/openvswitchlabs/openvswitch-2.4.0# make && make insta
ll
 • • •
make[3]: Leaving directory `/home/openvswitchlabs/openvswitch-2.4.0'
make[2]: Leaving directory `/home/openvswitchlabs/openvswitch-2.4.0'
make[1]: Leaving directory `/home/openvswitchlabs/openvswitch-2.4.0'
root@openvswitchlabs:/home/openvswitchlabs/openvswitch-2.4.0#
```

Θα φορτώσουμε το OpenVSwitch στον πυρήνα του Linux με τις εξής εντολές:

- cd datapath/linux
- modprobe openvswitch

```
root@openvswitchlabs:/home/openvswitchlabs/openvswitch-2.4.0# cd datapath/linux
root@openvswitchlabs:/home/openvswitchlabs/openvswitch-2.4.0/datapath/linux# mod
probe openvswitch
root@openvswitchlabs:/home/openvswitchlabs/openvswitch-2.4.0/datapath/linux#
```

Θα βεβαιώσουμε ότι τρέχει με την εντολή:

- lsmod | grep openvswitch

και θα χρειαστεί να δημιουργήσουμε ένα αρχείο και έναν φάκελο που είναι απαραίτητα για το OpenVSwitch με τις παρακάτω εντολές:

- touch /usr/local/etc/ovs-vswitchd.conf
- mkdir -p /usr/local/etc/openvswitch

και στη συνέχεια θα δημιουργήσουμε ένα αρχείο στον φάκελο του OpenVSwitch με τις παρακάτω εντολές:

- cd ../..
- ovsdb-tool create /usr/local/etc/openvswitch/conf.dbvswitchd/vswitch.ovsschema

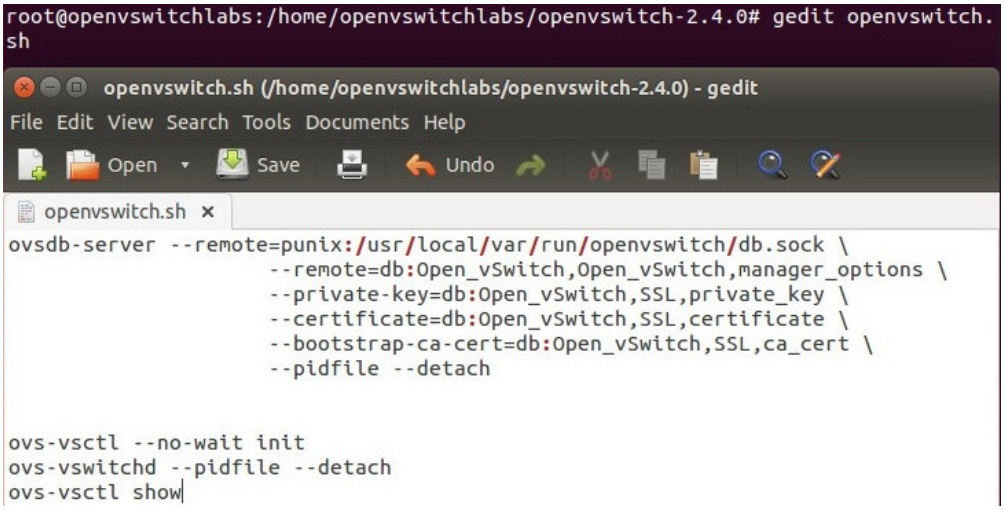
```
root@openvswitchlabs:/home/openvswitchlabs/openvswitch-2.4.0/datapath/linux# lsm
od | grep openvswitch
openvswitch            81920  0
libcrc32c              16384  1 openvswitch
root@openvswitchlabs:/home/openvswitchlabs/openvswitch-2.4.0/datapath/linux# tou
ch /usr/local/etc/ovs-vswitchd.conf
root@openvswitchlabs:/home/openvswitchlabs/openvswitch-2.4.0/datapath/linux# mkd
i
mkdir        mkdiskimage
root@openvswitchlabs:/home/openvswitchlabs/openvswitch-2.4.0/datapath/linux# mkd
i
mkdir        mkdiskimage
root@openvswitchlabs:/home/openvswitchlabs/openvswitch-2.4.0/datapath/linux# mkd
ir -p /usr/local/etc/openvswitch
root@openvswitchlabs:/home/openvswitchlabs/openvswitch-2.4.0/datapath/linux# cd
../..
root@openvswitchlabs:/home/openvswitchlabs/openvswitch-2.4.0# ovsdb-tool create
/usr/local/etc/openvswitch/conf.db vswitchd/vswitch.ovsschema
```

Θα δημιουργήσουμε το αρχείο openvswitch.sh στο φάκελο του OpenVSwitch:

- gedit openvswitch.sh

και στη συνέχεια θα αποθηκεύσουμε το αρχείο αφού πρώτα γράψουμε τα εξής:

- ovsdb-server --remote=punix:/usr/local/var/run/openvswitch/db.sock \
  --remote=db:Open_vSwitch,Open_vSwitch,manager_options \
  --private-key=db:Open_vSwitch,SSL,private_key \
  --certificate=db:Open_vSwitch,SSL,certificate \
  --bootstrap-ca-cert=db:Open_vSwitch,SSL,ca_cert \
  --pidfile --detach
- ovs-vsctl --no-wait init
- ovs-vswitchd --pidfile --detach
- ovs-vsctl show

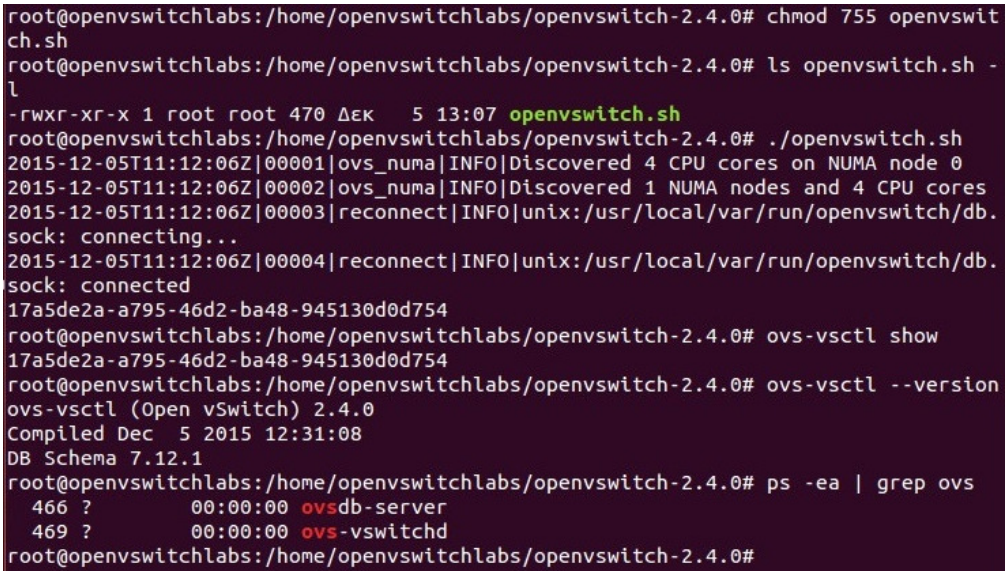


Στη συνέχεια θα αλλάξουμε την άδεια που έχει ο χρήστης μας για το αυτό και θα το εκτελέσουμε:

- chmod 755 openvswitch.sh
- ls openvswitch.sh -l
- ./openvswitch.sh

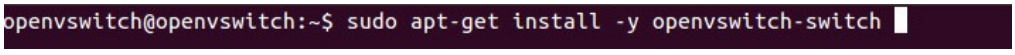
Αφού μας εμφανίσει τις πληροφορίες, μπορούμε να δούμε την κατάσταση του OpenVSwitch, την έκδοσή του και ποιες διεργασίες τρέχει με τις ακόλουθες εντολές:

- ovs-vsctl show
- ovs-vsctl --version
- ps -ea | grep ovs



Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε το γενικό πακέτο του openvswitch που υποστηρίζεται από τα repository που υπάρχουν στα Linux εκτελόντας την εντολή:

- apt-get install openvswitch-switch



και έχουμε την λειτουργία του openvswitch χωρίς να εγκαθιστούμε βήμα-βήμα τα πακέτα που χρειάζονται.

6.3 Προσομοίωση

Αφού έχουν εγκατασταθεί σωστά τα εργαλεία θα ακολουθήσουμε στην διαδικασία της προσομοίωσης.

6.3.1 Περιγραφή βημάτων Προσομοίωσης

Έχοντας εγκαταστήσει το openvswitch και τον floodlight controller ανοίγουμε το terminal του linux και δίνουμε τις εξής εντολές:

Για να ξεκινήσουμε τον floodlight controller ανοίγουμε τον φάκελο που είναι εγκατεστημένος από το terminal του linux και δίνουμε την εντολή:

- java –jar target/floodlight.jar

```
gxiannhs@ubuntu:~/floodlight$ sudo java -jar target/floodlight.jar
17:01:45.235 INFO [n.f.c.m.FloodlightModuleLoader:main] Loading modules from src
/main/resources/floodlightdefault.properties
17:01:45.736 WARN [n.f.r.RestApiServer:main] HTTPS disabled; HTTPS will not be u
sed to connect to the REST API.
17:01:45.737 WARN [n.f.r.RestApiServer:main] HTTP enabled; Allowing unsecure acc
ess to REST API on port 8080.
```

και στη συνέχεια θα ανοίξουμε ένα άλλο terminal και θα ξεκινήσουμε τις ρυθμίσεις για το openvswitch.

- ovs-vsctl add-br sdnbr
- ovs-vsctl add-port sdnbr wlan0
- ifconfig wlan0 0 up
- ifconfig sdnbr 192.168.2.3 netmask 255.255.255.0 up
- ovs-vsctl set-fail-mode sdnbr secure
- ovs-vsctl set-controller sdnbr tcp:192.168.2.3:6653
- ovs-vsctl show

```
gxiannhs@ubuntu:~$ sudo ovs-vsctl add-br sdnbr
gxiannhs@ubuntu:~$ sudo ovs-vsctl add-port sdnbr wla0
gxiannhs@ubuntu:~$ sudo ifconfig wlan0 0 up
gxiannhs@ubuntu:~$ sudo ifconfig sdnbr 192.168.2.3 netmask 255.255.255.0 up
gxiannhs@ubuntu:~$ sudo ovs-vsctl set-fail-mode sdnbr secure
gxiannhs@ubuntu:~$ sudo ovs-vsctl set-controller sdnbr tcp:192.168.2.3:6653
gxiannhs@ubuntu:~$ sudo ovs-vsctl show
21b481bc-a8e3-43f4-9fd3-4d3581257c47
    Bridge sdnbr
        Controller "tcp:192.168.2.3:6653"
            is_connected: true
        fail_mode: secure
        Port sdnbr
            Interface sdnbr
                type: internal
        Port "wla0"
            Interface "wla0"
    ovs_version: "2.0.1"
```

Από τη στιγμή αυτή μπορούμε να δούμε τη κατάσταση του openvswitch από το φυλλομετρητή στη διεύθυνση http://127.0.0.1:8080/ui/index.html

Switches (1)

DPID	IP Address	Vendor	Packets	Bytes	Flows	Connected Since
00:00:16:d8:cc:fb:04:42	/192.168.2.3:50033	Nicira, Inc.	0	0	0	17/3/2016, 5:03:53 μ.μ.

Θα προετοιμάσουμε τις εικονικές διεπαφές για το virtualbox ώστε να συνδέσουμε τις εικονικές μηχανές.

- ip tuntap add mode tap vnet1
- ip link set vnet1 up
- ovs-vsctl add-port sdnbr vnet1
- ifconfig sdnbr up
- ovs-vsctl show
- ip link

```
gxiannhs@ubuntu:~$ sudo ip tuntap add mode tap vnet1
gxiannhs@ubuntu:~$ sudo ip link set vnet1 up
gxiannhs@ubuntu:~$ sudo ovs-vsctl add-port sdnbr vnet1
gxiannhs@ubuntu:~$ sudo ifconfig sdnbr up
gxiannhs@ubuntu:~$ sudo ovs-vsctl show
21b481bc-a8e3-43f4-9fd3-4d3581257c47
    Bridge sdnbr
        Controller "tcp:192.168.2.3:6653"
            is_connected: true
        fail_mode: secure
        Port sdnbr
            Interface sdnbr
                type: internal
        Port "wla0"
            Interface "wla0"
        Port "vnet1"
            Interface "vnet1"
    ovs_version: "2.0.1"
gxiannhs@ubuntu:~$ sudo ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
 group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mo
de DEFAULT group default qlen 1000
    link/ether e8:03:9a:35:a8:4b brd ff:ff:ff:ff:ff:ff
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DORM
ANT group default qlen 1000
    link/ether b8:03:05:07:b2:2d brd ff:ff:ff:ff:ff:ff
4: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
 group default
    link/ether ae:4d:5f:9d:eb:1f brd ff:ff:ff:ff:ff:ff
6: sdnbr: <BROADCAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFA
ULT group default
    link/ether 16:d8:cc:fb:04:42 brd ff:ff:ff:ff:ff:ff
7: vnet1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast master o
vs-system state DOWN mode DEFAULT group default qlen 500
    link/ether a6:32:ee:f3:31:b6 brd ff:ff:ff:ff:ff:ff
gxiannhs@ubuntu:~$
```

- ifconfig

```
vnet1     Link encap:Ethernet  HWaddr a6:32:ee:f3:31:b6
          UP BROADCAST 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:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- ip tuntap add mode tap vnet2
- ip link set vnet2 up

- ovs-vsctl add-port sdnbr vnet2
- ifconfig sdnbr up
- ovs-vsctl show

```
gxiannhs@ubuntu:~$ sudo ip tuntap add mode tap vnet2
gxiannhs@ubuntu:~$ sudo ip link set vnet2 up
gxiannhs@ubuntu:~$ sudo ovs-vsctl add-port sdnbr vnet2
gxiannhs@ubuntu:~$ sudo ifconfig sdnbr up
gxiannhs@ubuntu:~$ sudo ovs-vsctl show
21b481bc-a8e3-43f4-9fd3-4d3581257c47
    Bridge sdnbr
        Controller "tcp:192.168.2.3:6653"
            is_connected: true
        fail_mode: secure
        Port sdnbr
            Interface sdnbr
                type: internal
        Port "wla0"
            Interface "wla0"
        Port "vnet2"
            Interface "vnet2"
        Port "vnet1"
            Interface "vnet1"
    ovs_version: "2.0.1"
```

- ip link

```
gxiannhs@ubuntu:~$ sudo ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
 group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mo
de DEFAULT group default qlen 1000
    link/ether e8:03:9a:35:a8:4b brd ff:ff:ff:ff:ff:ff
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DORM
ANT group default qlen 1000
    link/ether b8:03:05:07:b2:2d brd ff:ff:ff:ff:ff:ff
4: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
 group default
    link/ether ae:4d:5f:9d:eb:1f brd ff:ff:ff:ff:ff:ff
6: sdnbr: <BROADCAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFA
ULT group default
    link/ether 16:d8:cc:fb:04:42 brd ff:ff:ff:ff:ff:ff
7: vnet1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast master o
vs-system state DOWN mode DEFAULT group default qlen 500
    link/ether a6:32:ee:f3:31:b6 brd ff:ff:ff:ff:ff:ff
8: vnet2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast master o
vs-system state DOWN mode DEFAULT group default qlen 500
    link/ether e2:ae:04:3e:6d:32 brd ff:ff:ff:ff:ff:ff
```

- ifconfig

```
vnet2     Link encap:Ethernet  HWaddr e2:ae:04:3e:6d:32
          UP BROADCAST 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:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- ovs-ofctl dump-flows sdnbr

```
gxiannhs@ubuntu:~$ sudo ovs-ofctl dump-flows sdnbr
NXST_FLOW reply (xid=0x4):
```

Επιβεβαιώνουμε ότι το openvswitch δεν έχει καμία εγγραφή στο πίνακα ροών.

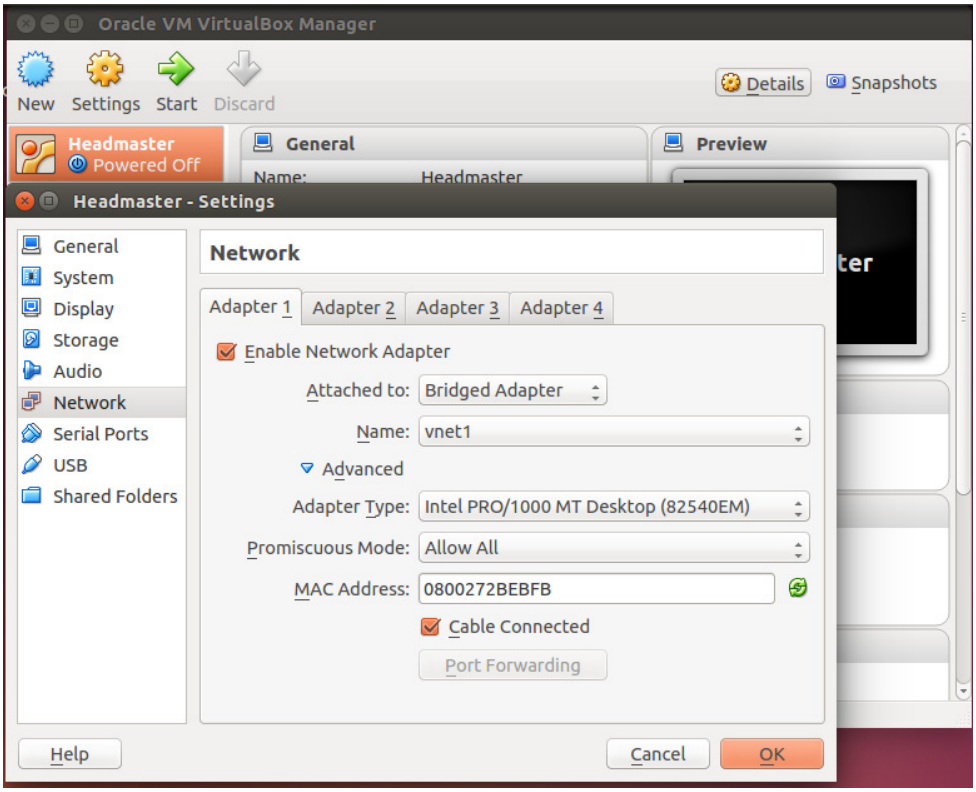
Για τη δημιουργία των παραπάνω βημάτων μπορούμε να κατασκευάσουμε ένα bash script όπου θα εκτελεί αυτόματα τις εντολές χωρίς να χρειάζεται να τις γράφουμε.

```
echo "Create SDN Bridge in OpenVSwitch"
sudo ovs-vsctl add-br sdnbr
sleep 2
echo "Attach Ehternet Port to SDN Bridge"
sudo ovs-vsctl add-port sdnbr wlan0
sleep 2
echo "Delete settings from Ethernet Port"
sudo ifconfig wlan0 0 up
sleep 2
echo "Setup ip address to SDN Bridge"
sudo ifconfig sdnbr 192.168.2.3 netmask 255.255.255.0 up
sleep 2
echo "Set SDN Bridge in secure mode"
sudo ovs-vsctl set-fail-mode sdnbr secure
sleep 2
echo "Attach Floodlight controller to SDN Bridge"
sudo ovs-vsctl set-controller sdnbr tcp:192.168.2.3:6653
echo "Show OpenVSwitch control settings"
sudo ovs-vsctl show
sleep 4
echo "Create virtual tap interface vnet1"
sudo ip tuntap add mode tap vnet1
sleep 2
echo "Setting up vnet1"
sudo ip link set vnet1 up
sleep 2
echo "Add the vnet1 to SDN Bridge"
sudo ovs-vsctl add-port sdnbr vnet1
sleep 2
echo "Restart SDN Bridge"
sudo ifconfig sdnbr up
echo "Show OpenVSwitch control settings"
sudo ovs-vsctl show
sleep 4
echo "Show Links that exists in the system"
ip link
```

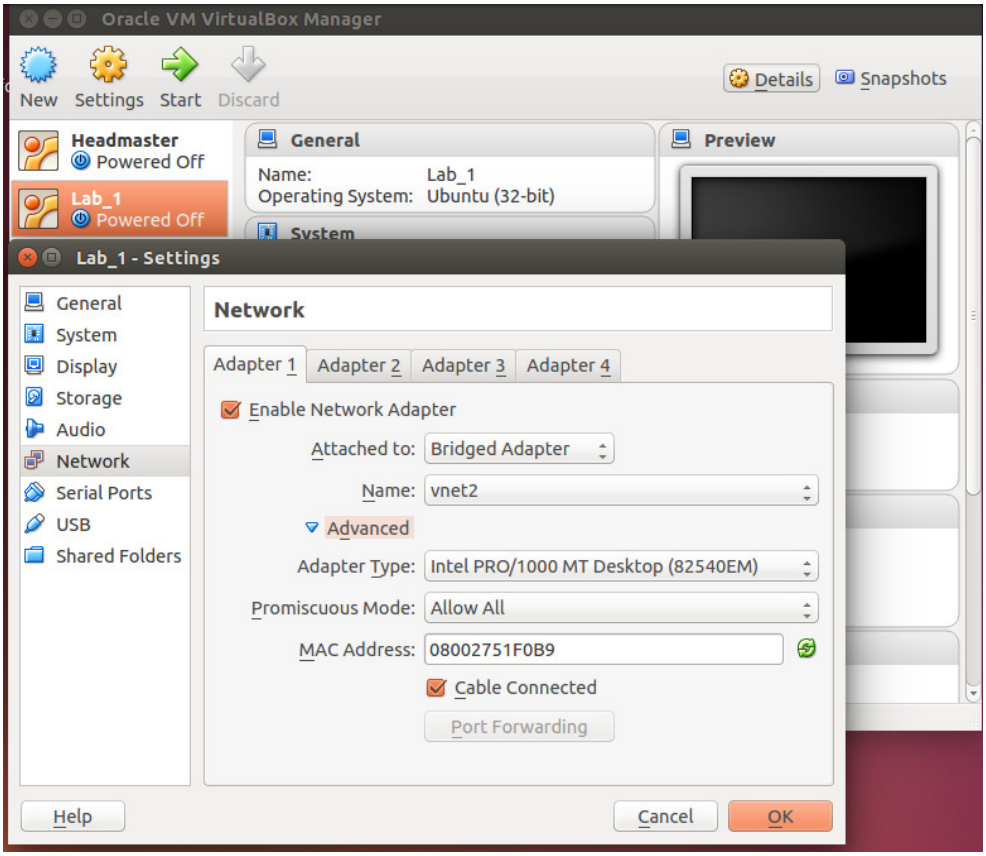
```
sleep 4
echo "Show Interfaces"
ifconfig
sleep 2
echo "Create virtual tap interface vnet2"
sudo ip tuntap add mode tap vnet2
sleep 2
echo "Setting up vnet2"
sudo ip link set vnet2 up
sleep 2
echo "Add the vnet2 to SDN Bridge"
sudo ovs-vsctl add-port sdnbr vnet2
sleep 2
echo "Restart SDN Bridge"
sudo ifconfig sdnbr up
echo "Show OpenVSwitch control settings"
sudo ovs-vsctl show
sleep 4
echo "Show Links that exists in the system"
ip link
sleep 4
echo "Show Interfaces"
ifconfig
sleep 2
echo "Show Flows of SDN Bridge"
sudo ovs-ofctl dump-flows sdnbr
```

Στη συνέχεια θα συνδέσουμε τις δύο εικονικές μηχανές στις δύο εικονικές διεπαφές που δημιουργήσαμε στα προηγούμενα βήματα.

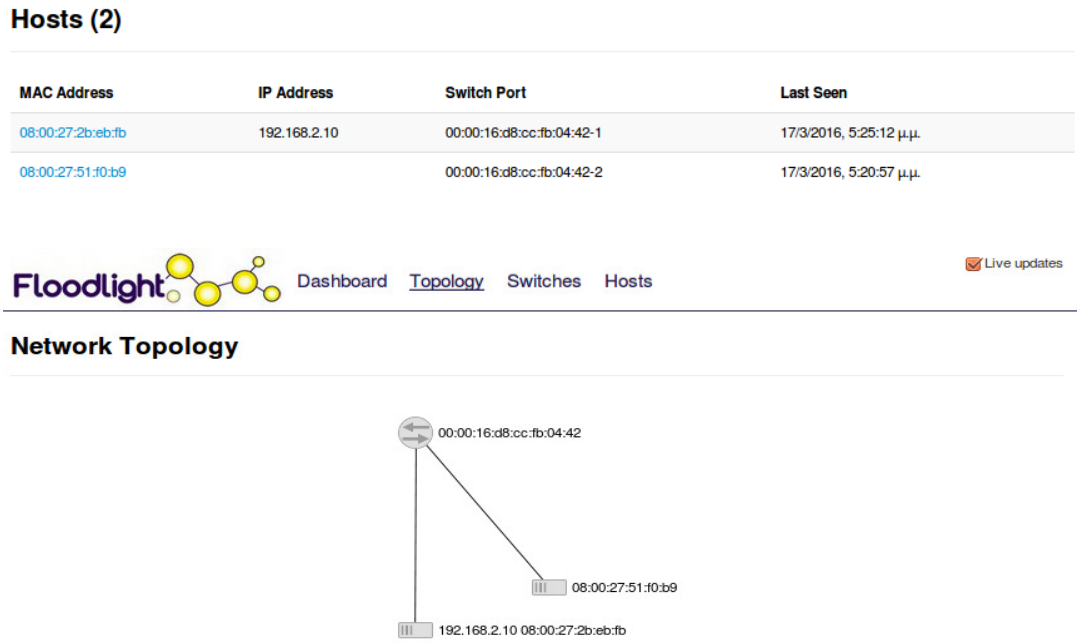
Αφού ανοίξουμε το Virtualbox επιλέγουμε την εικονική μηχανή “Headmaster” και πατάμε το κουμπί “Settings” στο παράθυρο που μας ανοίγει επιλέγουμε την καρτέλα “Network” από το πλάγιο μενού και στο πεδίο “Attached to” επιλέγουμε “Bridged Adapter”, στο πεδίο “Name” επιλέγουμε “vnet1” και στο πεδίο “Promiscuous Mode” των “Advanced” επιλογών επιλέγουμε το “Allow All”.



Στη συνέχεια επιλέγουμε την εικονική μηχανή “Lab1” και πατάμε το κουμπί “Settings” στο παράθυρο που μας ανοίγει επιλέγουμε την καρτέλα “Network” από το πλάγιο μενού και στο πεδίο “Attached to” επιλέγουμε “Bridged Adapter”, στο πεδίο “Name” επιλέγουμε “vnet2” και στο πεδίο “Promiscuous Mode” των “Advanced” επιλογών επιλέγουμε το “Allow All”.



Μετά ανοίγουμε το φυλλομετρητή στη διεύθυνση http://127.0.0.1:8080/ui/index.html και επιβεβαιώνουμε την σύνδεση των εικονικών μηχανών στο openvswitch και την τοπολογία του δικτύου.



6.3.2 Τροφοδότηση πραγματικών flows στο παραπάνω Δίκτυο

Σε συνέχεια του προηγούμενου κεφαλαίου αφού έχουμε υλοποιήσει το πραγματικό δίκτυο για να του δώσουμε την λειτουργικότητα ώστε να είναι εφικτή η επικοινωνία με το internet θα πρέπει να ορίσουμε τα flows όπου θα τα στείλουμε στον μεταγωγέα μέσω του ελεγκτή του.

Για να διασυνδέσουμε τις δύο εικονικές μηχανές και να επικοινωνούν μεταξύ τους θα δώσουμε στο openvswitch δύο ροές κίνησης μέσω του controller:

- curl –s –d ‘{“switch”:”00:00:16:d8:cc:fb:04:42”,”name”:”flow-1”,”priority”:”1”,”in_port”:”1”,”src-ip”:”192.168.2.10”,”dst-ip”:”192.168.2.11”,”actions”:”output=2”}’ http://192.168.2.3:8080/wm/staticflowpusher/json
- curl –s –d ‘{“switch”:” 00:00:16:d8:cc:fb:04:42”,”name”:”flow-2”,”priority”:”2”,”in_port”:”2”,”src-ip”:”192.168.2.11”,”dst-ip”:”192.168.2.10”,”actions”:”output=3”}’ http://192.168.2.3:8080/wm/staticflowpusher/json

```
gxiannhs@ubuntu:~$ curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
1","priority":"1","in_port":"1","src-ip":"192.168.2.10","dst-ip":"192.168.2.11",
"actions":"output=2"}' http://192.168.2.3:8080/wm/staticflowpusher/json
{"status" : "Entry pushed"}gxiannhs@ubuntu:~$
gxiannhs@ubuntu:~$ curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
2","priority":"2","in_port":"2","src-ip":"192.168.2.11","dst-ip":"192.168.2.10",
"actions":"output=1"}' http://192.168.2.3:8080/wm/staticflowpusher/json
{"status" : "Entry pushed"}gxiannhs@ubuntu:~$
```

Επίσης, μπορούμε να εκτελέσουμε το bash script για την εισαγωγή των ροών αυτόματα.

```
echo "Create Flow for packets from vnet1 to vnet2"
```

```
curl –s –d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
1","priority":"1","in_port":"1","src-ip":"192.168.2.10","dst-
ip":"192.168.2.11","actions":"output=2"}'
http://192.168.2.3:8080/wm/staticflowpusher/json

sleep 2

echo "Create Flow for packets from vnet2 to vnet1"

curl –s –d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
2","priority":"2","in_port":"2","src-ip":"192.168.2.11","dst-
ip":"192.168.2.10","actions":"output=1"}'
http://192.168.2.3:8080/wm/staticflowpusher/json
```

Τώρα μπορούμε να εκτελέσουμε ping ανάμεσα στις δύο εικονικές μηχανές και επίσης μπορούμε να δούμε στο Dashboard του floodlight controller τις ροές κίνησης που έχει το openvswitch όπως και να εκτελέσουμε την εντολή:

- ovs-ofctl dump-flows sdnbr

```
gxiannhs@ubuntu:~$ sudo ovs-ofctl dump-flows sdnbr
NXST_FLOW reply (xid=0x4):
 cookie=0xa00000ec6a7509, duration=20.396s, table=0, n_packets=16, n_bytes=960,
idle_age=2, priority=1,in_port=1 actions=output:2
 cookie=0xa00000ec6a750a, duration=2.416s, table=0, n_packets=0, n_bytes=0, idle
_age=2, priority=2,in_port=2 actions=output:1
```

Floodlight Dashboard Topology Switches Hosts Live updates

Switch 00:00:16:d8:cc:fb:04:42 /192.168.2.3:50033

Connected Since 17/3/2016, 5:03:53 μ.μ.
Nicira, Inc.
Open vSwitch
2.0.1
S/N: None
OpenFlow Version: OF_10

Ports (0)

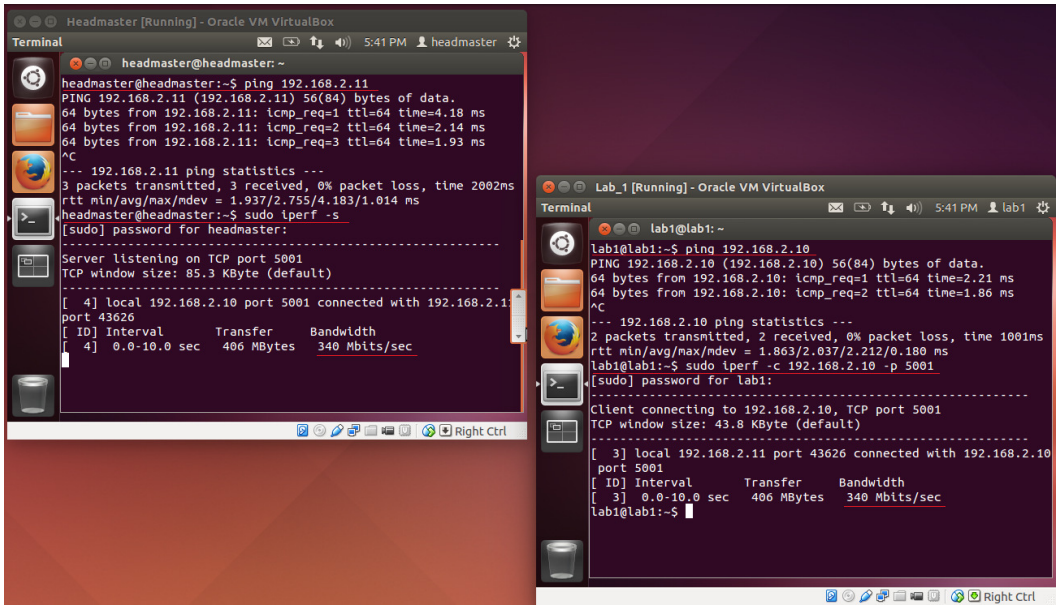
#	Link Status	TX Bytes	RX Bytes	TX Pkts	RX Pkts	Dropped	Errors

Flows (2)

Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)	Timeout (s)
45036000240104713	0x0	1	in_port=1	actions:output=2	n/a	n/a	n/a	n/a	n/a	n/a	157	9610	246	0
45036000240104714	0x0	2	in_port=2	actions:output=1	n/a	n/a	n/a	n/a	n/a	n/a	6	550	228	0

Μπορούμε να ελέγξουμε το bandwidth ανάμεσα στις δύο εικονικές μηχανές με τις εντολές:

- iperf –s (στη μία εικονική μηχανή)
- iperf –c 192.168.2.10 –p 5001



6.3.3 Αντιμετώπιση Ανοικτού Ερευνητικού Προβλήματος

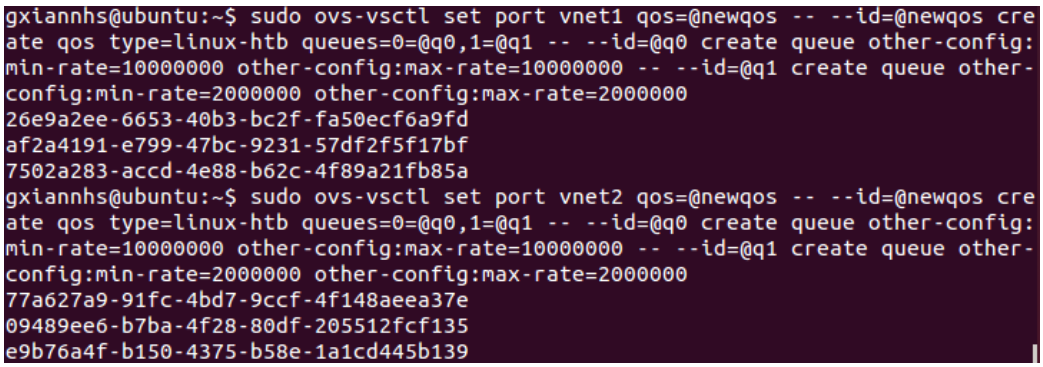
Σε αυτό το σημείο το δίκτυο έχουμε υλοποιήσει την λειτουργία ενός πραγματικού δικτύου χωρίς να επέμβουμε στην λογική του. Παρατηρώντας το παραπάνω δίκτυο θα μπορούσαμε να εφαρμόσουμε κάποια flows ώστε να είναι απαραίτητη η χρήση του SDN. Έτσι καταλήγουμε σε ένα ανοικτό πρόβλημα που δεν έχει κάποια υπάρχουσα λύση με βάση την έρευνα που ακολουθήθηκε στο κεφάλαιο 6. Προσδιορίζοντας το πρόβλημα αυτό είναι ότι δεν υπάρχει κάποια λογική ποιότητας της υπηρεσίας (QoS) στο δίκτυο αυτό και έτσι θα προσπαθήσουμε να το αντιμετωπίσουμε.

Ένα σενάριο χρήσης στο δίκτυο αυτό θα ήταν τι θα γινόταν στην περίπτωση που τα εργαστήρια έκαναν υπερβολική χρήση της γραμμής του internet και π.χ. δεν θα μπορούσε να έχει μία ποιοτική τηλεδιάσκεψη το γραφείο του Διευθυντή. Η απάντηση σε αυτό το σενάριο είναι δημιουργήσουμε ουρές κίνησης στον μεταγωγέα και με αυτό τον τρόπο θα μειώσουμε την κίνηση στο κανάλι επικοινωνίας δίνοντας ροές-κίνησης μέσω του ελεγκτή για να οδηγηθούν τα πακέτα δεδομένων στις ουρές κίνησης με αποτέλεσμα να μειώσουμε το εύρος ζώνης στο κανάλι επικοινωνίας.

Δημιουργία κανόνων ποιότητας της υπηρεσίας (QoS) και ουρών στο openvswitch σε κάθε εικονική διεπαφή.

- ovs-vsctl set port vnet1 qos=@newqos -- --id=@newqos create qos type=linux-htb queue=0=@q0,1=@q1 -- --id=@q0 create queue other-config:min-rate=10000000 other-config:max-rate=10000000 -- --id=@q1 create queue other-config:min-rate=2000000 other-config:max-rate=2000000
- ovs-vsctl set port vnet2 qos=@newqos -- --id=@newqos create qos type=linux-htb queue=0=@q0,1=@q1 -- --id=@q0 create queue other-config:min-rate=10000000 other-config:max-rate=10000000 -- --id=@q1

create queue other-config:min-rate=2000000 other-config:max-rate=2000000



Επίσης, μπορούμε να εκτελέσουμε το bash script για την εισαγωγή των ουρών αυτόματα.

```
echo "Create 2 QoS at vnet1 -> 0(default):10Mbps, 1:2Mbps"

sudo ovs-vsctl set port vnet1 qos=@newqos -- --id=@newqos create
qos type=linux-htb queues=0=@q0,1=@q1 -- --id=@q0 create queue
other-config:min-rate=10000000 other-config:max-rate=10000000 --
--id=@q1 create queue other-config:min-rate=2000000 other-
config:max-rate=2000000

sleep 3

echo "Create 2 QoS at vnet2 -> 0(default):10Mbps, 1:2Mbps"

sudo ovs-vsctl set port vnet2 qos=@newqos -- --id=@newqos create
qos type=linux-htb queues=0=@q0,1=@q1 -- --id=@q0 create queue
other-config:min-rate=10000000 other-config:max-rate=10000000 --
--id=@q1 create queue other-config:min-rate=2000000 other-
config:max-rate=2000000

sleep 3

echo "Show Created QoS"

sudo ovs-vsctl list qos

sleep 5

echo "Show Created Queue"

sudo ovs-vsctl list queue
```

Μπορούμε να ελέγξουμε τις ουρές κίνησης εκτελούμε:

- ovs-vsctl list qos

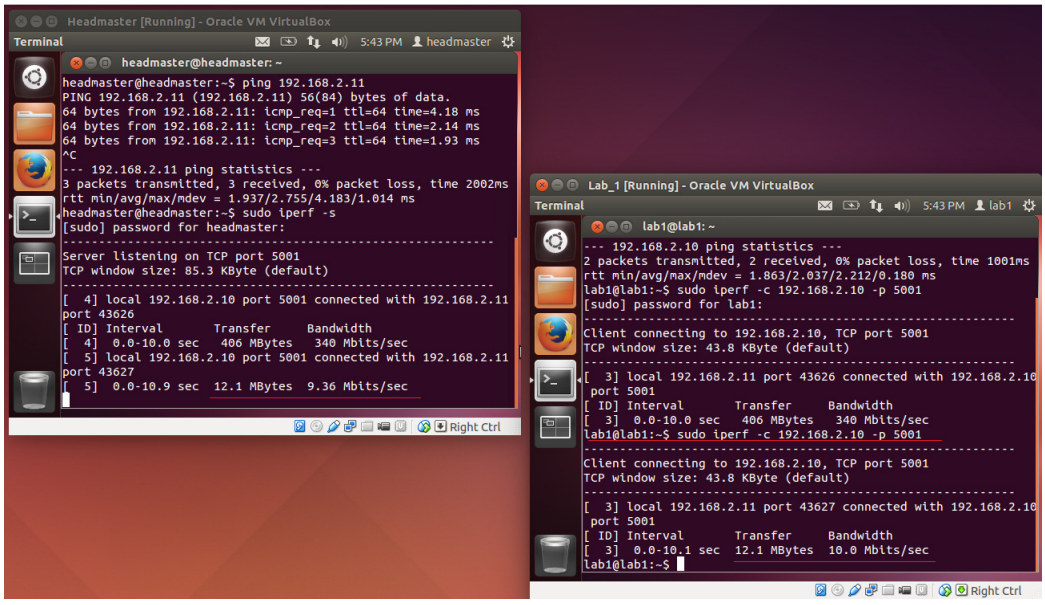
```
gxiannhs@ubuntu:~$ sudo ovs-vsctl list qos
_uuid               : 77a627a9-91fc-4bd7-9ccf-4f148aeea37e
external_ids        : {}
other_config        : {}
queues              : {0=09489ee6-b7ba-4f28-80df-205512fcf135, 1=e9b76a4f-b150-4
375-b58e-1a1cd445b139}
type                : linux-htb

_uuid               : 26e9a2ee-6653-40b3-bc2f-fa50ecf6a9fd
external_ids        : {}
other_config        : {}
queues              : {0=af2a4191-e799-47bc-9231-57df2f5f17bf, 1=7502a283-accd-4
e88-b62c-4f89a21fb85a}
type                : linux-htb
```

- ovs-vsctl list queue

```
gxiannhs@ubuntu:~$ sudo ovs-vsctl list queue
_uuid               : af2a4191-e799-47bc-9231-57df2f5f17bf
dscp                : []
external_ids        : {}
other_config        : {max-rate="10000000", min-rate="10000000"}

_uuid               : 09489ee6-b7ba-4f28-80df-205512fcf135
dscp                : []
external_ids        : {}
other_config        : {max-rate="10000000", min-rate="10000000"}

_uuid               : e9b76a4f-b150-4375-b58e-1a1cd445b139
dscp                : []
external_ids        : {}
other_config        : {max-rate="2000000", min-rate="2000000"}

_uuid               : 7502a283-accd-4e88-b62c-4f89a21fb85a
dscp                : []
external_ids        : {}
other_config        : {max-rate="2000000", min-rate="2000000"}
```

Μπορούμε να ελέγξουμε το bandwidth ανάμεσα στις δύο εικονικές μηχανές με τις εντολές:

- iperf –s (στη μία εικονική μηχανή)
- iperf –c 192.168.2.10 –p 5001



Σε αυτό το σημείο ισχύουν οι ροές κίνησης που έχουμε δώσει παραπάνω και μειώνουμε το bandwidth στα 10Mbps/sec. Θα αλλάξουμε τις ουρές κίνησης για να μειώσουμε το bandwidth στα 2Mbps/sec.

Πρώτα θα σβήσουμε τις υπάρχουσες ροές με την εντολή:

- ovs-ofctl del-flows sdnbr

```
gxiannhs@ubuntu:~$ sudo ovs-ofctl del-flows sdnbr
```

Επιβεβαιώνουμε ότι οι ροές σβήστηκαν ανοίγοντας τον φυλλομετρητή καθώς και με την εντολή:

- ovs-ofctl dump-flows sdnbr

```
gxiannhs@ubuntu:~$ sudo ovs-ofctl dump-flows sdnbr
NXST_FLOW reply (xid=0x4):
```

Switches (1)

DPID	IP Address	Vendor	Packets	Bytes	Flows	Connected Since
00:00:16:d8:cc:fb:04:42	/192.168.2.3:50033	Nicira, Inc.	0	0	0	17/3/2016, 5:03:53 μ.μ.

Για να διασυνδέσουμε τις δύο εικονικές μηχανές και να επικοινωνούν μεταξύ τους θα δώσουμε στο openvswitch δύο ροές κίνησης μέσω του controller αλλά θα δώσουμε σαν έξοδο την δεύτερη ουρά κίνησης για να μειώσουμε το bandwidth.

- curl –s –d ‘{“switch”:” 00:00:16:d8:cc:fb:04:42”,”name”:”flow-1”,”priority”:”1”,”in_port”:”1”,”src-ip”:”192.168.2.10”,”dst-ip”:”192.168.2.11”,”actions”:”enqueue=2:1”}’ http://192.168.2.3:8080/wm/staticflowpusher/json
- curl –s –d ‘{“switch”:” 00:00:16:d8:cc:fb:04:42”,”name”:”flow-2”,”priority”:”2”,”in_port”:”2”,”src-ip”:”192.168.2.11”,”dst-ip”:”192.168.2.10”,”actions”:” enqueue =1:1”}’ http://192.168.2.3:8080/wm/staticflowpusher/json

```
gxiannhs@ubuntu:~$ curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
1","priority":"1","in_port":"1","src-ip":"192.168.2.10","dst-ip":"192.168.2.11",
"actions":"enqueue=2:1"}' http://192.168.2.3:8080/wm/staticflowpusher/json
{"status" : "Entry pushed"}gxiannhs@ubuntu:~$
gxiannhs@ubuntu:~$ curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
2","priority":"2","in_port":"2","src-ip":"192.168.2.11","dst-ip":"192.168.2.10",
"actions":"enqueue=1:1"}' http://192.168.2.3:8080/wm/staticflowpusher/json
{"status" : "Entry pushed"}gxiannhs@ubuntu:~$
```

Επίσης, μπορούμε να εκτελέσουμε το bash script για την εισαγωγή των ροών αυτόματα.

```
echo "Create Flow for packets from vnet1 to vnet2"

curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
1","priority":"1","in_port":"1","src-ip":"192.168.2.10","dst-
ip":"192.168.2.11","actions":"enqueue=2:1"}'
http://192.168.2.3:8080/wm/staticflowpusher/json

sleep 2

echo "Create Flow for packets from vnet2 to vnet1"

curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
2","priority":"2","in_port":"2","src-ip":"192.168.2.11","dst-
ip":"192.168.2.10","actions":"enqueue=1:1"}'
http://192.168.2.3:8080/wm/staticflowpusher/json
```

μπορούμε να δούμε στο Dashboard του floodlight controller τις ροές κίνησης που έχει το openvswitch όπως και να εκτελέσουμε την εντολή:

- ovs-ofctl dump-flows sdnbr

```
gxiannhs@ubuntu:~$ sudo ovs-ofctl dump-flows sdnbr
NXST_FLOW reply (xid=0x4):
 cookie=0xa00000ec6a7509, duration=21.573s, table=0, n_packets=17, n_bytes=1020,
 idle_age=1, priority=1,in_port=1 actions=enqueue:2q1
 cookie=0xa00000ec6a750a, duration=3.07s, table=0, n_packets=0, n_bytes=0, idle_
age=3, priority=2,in_port=2 actions=enqueue:1q1
```

Floodlight Dashboard Topology Switches Hosts Live updates

Switch 00:00:16:d8:cc:fb:04:42 /192.168.2.3:50033

Connected Since 17/3/2016, 5:03:53 μ.μ.
Nicira, Inc.
Open vSwitch
2.0.1
S/N: None
OpenFlow Version: OF_10

Ports (0)

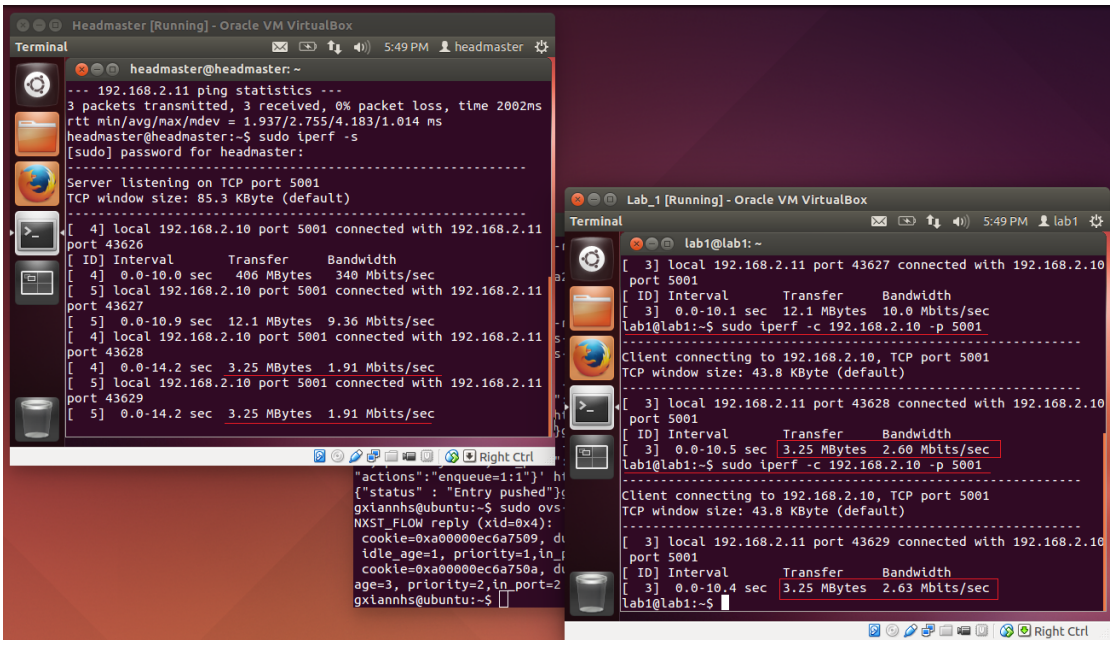
#	Link Status	TX Bytes	RX Bytes	TX Pkts	RX Pkts	Dropped	Errors

Flows (2)

Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)	Timeout (s)
45036000240104713	0x0	1	in_port=1	actions:enqueue=2	n/a	n/a	n/a	n/a	n/a	n/a	31	1860	45	0
45036000240104714	0x0	2	in_port=2	actions:enqueue=1	n/a	n/a	n/a	n/a	n/a	n/a	0	0	27	0

Μπορούμε να τσεκάρουμε το bandwidth ανάμεσα στις δύο εικονικές μηχανές με τις εντολές:

- iperf –s (στη μία εικονική μηχανή)
- iperf –c 192.168.2.10 –p 5001



Σε αυτό το σημείο ισχύουν οι ροές κίνησης που έχουμε δώσει παραπάνω και μειώνουμε το bandwidth στα 2Mbps/sec.

Έτσι με αυτόν τον τρόπο περιορίζουμε την κίνηση στο δίκτυο και μπορούμε να δώσουμε το απαιτούμενο bandwidth για μία ποιοτική τηλεδιάσκεψη.

6.3.4 Σενάριο Επεκτασιμότητας

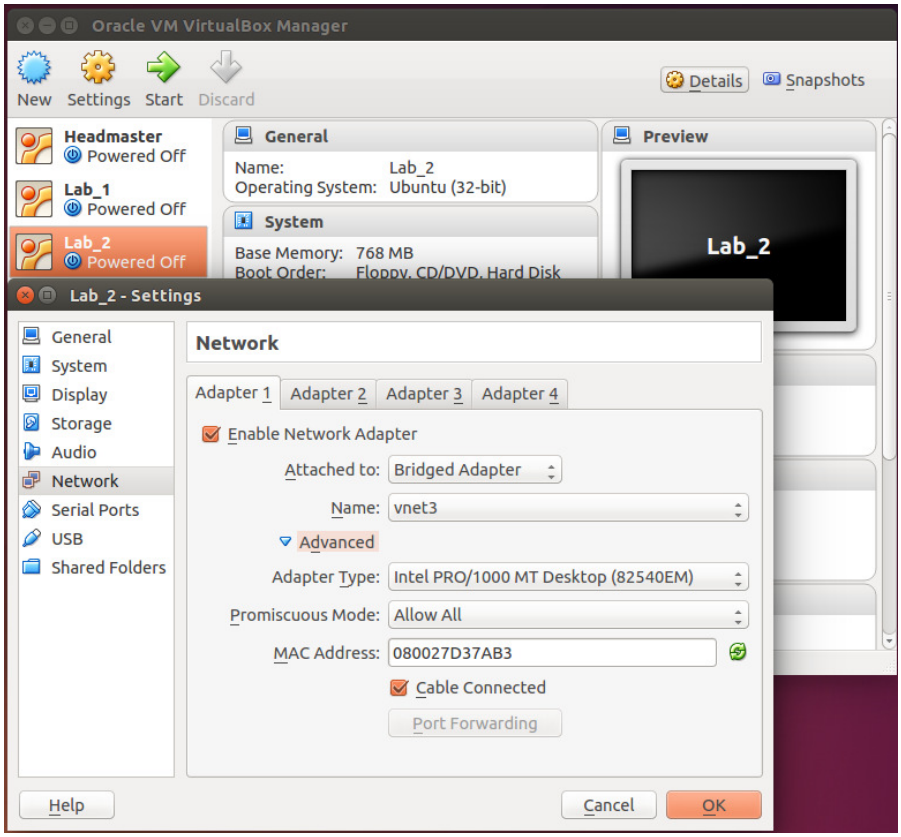
Αφού έχουμε το παραπάνω δίκτυο πλήρως υλοποιημένο ας υποθέσουμε ότι θέλουμε να προσθέσουμε μία επιπλέον εικονική μηχανή. Τα βήματα που θα πρέπει να ακολουθήσουμε θα πρέπει να είναι:

1. Δημιουργία επιπλέον μίας εικονικής διεπαφής στον εικονικό μεταγωγέα (Open vSwitch).
2. Δημιουργία μίας εικονικής μηχανής στο περιβάλλον εικονικοποίησης (Virtualbox).
3. Σύνδεση δικτύου της εικονικής μηχανής με την εικονική διεπαφή.
4. Δημιουργία ουρών κίνησης για την εικονική διεπαφή.
5. Ορισμός ροών-κίνησης για την επικοινωνία μεταξύ των εικονικών μηχανών.
6. Ορισμός ροών-κίνησης για την επικοινωνία μεταξύ των εικονικών μηχανών μέσω των ουρών κίνησης για υλοποίηση σεναρίου QoS – ποιότητα της υπηρεσίας.

Βέβαια αυτά θα είναι τα βήματα που θα πρέπει να ακολουθήσουμε στην περίπτωση που θέλουμε να προσθέσουμε όσες εικονικές μηχανές θέλουμε.

Στη συνέχεια θα αναλύσουμε την περίπτωση που θα επεκτείνουμε το δίκτυο κατά μία παραπάνω εικονική μηχανή και θα παραθέσουμε ποιες εντολές θα πρέπει να εκτελέσουμε για την υλοποίηση του σεναρίου μας:

1. Για τη δημιουργία της εικονικής διεπαφής στο Open vSwitch θα εκτελέσουμε τις ακόλουθες εντολές στην κονσόλα:
    - ip tuntap add mode tap vnet3
    - ip link set vnet3 up
    - ovs-vsctl add-port sdnbr vnet3
    - ifconfig sdnbr up
    - ovs-vsctl show
    - ip link
2. Για τη δημιουργία μίας καινούριας εικονικής μηχανής πατάμε το New πάνω αριστερά στο παράθυρο διαλόγου του Virtualbox και ακολουθούμε τα απαραίτητα βήματα για την ολοκλήρωση της διαδικασίας.
3. Για τη σύνδεση ακολουθούμε την εξής εικόνα:



- και θα αναθέσουμε ως IP διεύθυνση της εικονικής μηχανής το 192.168.2.12 μόλις ανοίξει το περιβάλλον του λειτουργικού της

4. Για τη δημιουργία ουρών κίνησης της εικονικής διεπαφής θα εκτελέσουμε την ακόλουθη εντολή στην κονσόλα:
    - sudo ovs-vsctl set port vnet3 qos=@newqos -- --id=@newqos create qos type=linux-htb queues=0=@q0,1=@q1 -- --id=@q0 create queue other-config:min-rate=10000000 other-config:max-rate=10000000 -- --id=@q1 create queue other-config:min-rate=2000000 other-config:max-rate=200000
5. Για τον ορισμό ροών-κίνησης για την επικοινωνία μεταξύ των εικονικών μηχανών θα εκτελέσουμε τις ακόλουθες εντολές (με έντονα γράμματα αποτυπώνονται οι αλλαγές σε σχέση με το προηγούμενο σενάριο και στο περιεχόμενο των παρενθέσεων η κίνηση των διεπαφών):
    - curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-1","priority":"1","in_port":"1","src-ip":"192.168.2.10","dst-ip":"192.168.2.11","actions":"output=2"}' http://192.168.2.3:8080/wm/staticflowpusher/json (vnet1 -> vnet2)
    - curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-2","priority":"2","in_port":"2","src-ip":"192.168.2.11","dst-ip":"192.168.2.10","actions":"output=1"}' http://192.168.2.3:8080/wm/staticflowpusher/json (vnet2 -> vnet1)
    - **curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-3","priority":"3","in_port":"1","src-ip":"192.168.2.10","dst-ip":"192.168.2.12","actions":"output=3"}' http://192.168.2.3:8080/wm/staticflowpusher/json** (vnet1 -> vnet3)

- **curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-4","priority":"4","in_port":"3","src-ip":"192.168.2.12","dst-ip":"192.168.2.10","actions":"output=1"}' http://192.168.2.3:8080/wm/staticflowpusher/json** (vnet3 -> vnet1)
- **curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-5","priority":"5","in_port":"2","src-ip":"192.168.2.11","dst-ip":"192.168.2.12","actions":"output=3"}' http://192.168.2.3:8080/wm/staticflowpusher/json** (vnet2 -> vnet3)
- **curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-6","priority":"6","in_port":"3","src-ip":"192.168.2.12","dst-ip":"192.168.2.11","actions":"output=2"}' http://192.168.2.3:8080/wm/staticflowpusher/json** (vnet3 -> vnet2)

6. Για τον ορισμό ροών-κίνησης για την επικοινωνία μεταξύ των εικονικών μηχανών μέσω των ουρών κίνησης για υλοποίηση σεναρίου QoS – ποιότητα της υπηρεσίας (με έντονα γράμματα αποτυπώνονται οι αλλαγές σε σχέση με το προηγούμενο σενάριο και στο περιεχόμενο των παρενθέσεων η κίνηση των διεπαφών):
    - curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-1","priority":"1","in_port":"1","src-ip":"192.168.2.10","dst-ip":"192.168.2.11","actions":"enqueue=2:1"}' http://192.168.2.3:8080/wm/staticflowpusher/json (vnet1 -> vnet2)
    - curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-2","priority":"2","in_port":"2","src-ip":"192.168.2.11","dst-ip":"192.168.2.10","actions":"enqueue=1:1"}' http://192.168.2.3:8080/wm/staticflowpusher/json (vnet2 -> vnet1)
    - **curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-3","priority":"3","in_port":"1","src-ip":"192.168.2.10","dst-ip":"192.168.2.12","actions":"enqueue=3:1"}' http://192.168.2.3:8080/wm/staticflowpusher/json** (vnet1 -> vnet3)
    - **curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-4","priority":"4","in_port":"3","src-ip":"192.168.2.12","dst-ip":"192.168.2.10","actions":"enqueue=1:1"}' http://192.168.2.3:8080/wm/staticflowpusher/json** (vnet3 -> vnet1)
    - **curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-5","priority":"5","in_port":"2","src-ip":"192.168.2.11","dst-ip":"192.168.2.12","actions":"enqueue=3:1"}' http://192.168.2.3:8080/wm/staticflowpusher/json** (vnet2 -> vnet3)
    - **curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-6","priority":"6","in_port":"3","src-ip":"192.168.2.12","dst-ip":"192.168.2.11","actions":"enqueue=2:1"}' http://192.168.2.3:8080/wm/staticflowpusher/json** (vnet3 -> vnet2)

Τα ακόλουθα script μας βοηθούν από το να γράφουμε και να εκτελούμε μία-μία τις παραπάνω εντολές.

Script βήματος 1

```
echo "Create virtual tap interface vnet3"
sudo ip tuntap add mode tap vnet3
sleep 2
echo "Setting up vnet3"
sudo ip link set vnet3 up
sleep 2
echo "Add the vnet3 to SDN Bridge"
sudo ovs-vsctl add-port sdnbr vnet3
sleep 2
echo "Restart SDN Bridge"
sudo ifconfig sdnbr up
echo "Show OpenVSwitch control settings"
sudo ovs-vsctl show
sleep 4
echo "Show Links that exists in the system"
ip link
sleep 4
echo "Show Interfaces"
ifconfig
sleep 2
echo "Show Flows of SDN Bridge"
sudo ovs-ofctl dump-flows sdnbr
```

Script βήματος 4

```
echo "Create 2 QoS at vnet3 -> 0(default):10Mbps, 1:2Mbps"
sudo ovs-vsctl set port vnet3 qos=@newqos -- --id=@newqos create
qos type=linux-htb queues=0=@q0,1=@q1 -- --id=@q0 create queue
other-config:min-rate=10000000 other-config:max-rate=10000000 --
--id=@q1 create queue other-config:min-rate=2000000 other-
config:max-rate=2000000
sleep 3
echo "Show Created QoS"
sudo ovs-vsctl list qos
sleep 5
echo "Show Created Queue"
sudo ovs-vsctl list queue
```

Script βήματος 5

```
echo "Create Flow for packets from vnet1 to vnet2"

curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
1","priority":"1","in_port":"1","src-ip":"192.168.2.10","dst-
ip":"192.168.2.11","actions":"output=2"}'
http://192.168.2.3:8080/wm/staticflowpusher/json

sleep 2

echo "Create Flow for packets from vnet2 to vnet1"

curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
2","priority":"2","in_port":"2","src-ip":"192.168.2.11","dst-
ip":"192.168.2.10","actions":"output=1"}'
http://192.168.2.3:8080/wm/staticflowpusher/json

sleep 2

echo "Create Flow for packets from vnet1 to vnet3"

curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
3","priority":"3","in_port":"1","src-ip":"192.168.2.10","dst-
ip":"192.168.2.12","actions":"output=3"}'
http://192.168.2.3:8080/wm/staticflowpusher/json

sleep 2

echo "Create Flow for packets from vnet3 to vnet1"

curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
4","priority":"4","in_port":"3","src-ip":"192.168.2.12","dst-
ip":"192.168.2.10","actions":"output=1"}'
http://192.168.2.3:8080/wm/staticflowpusher/json

sleep 2

echo "Create Flow for packets from vnet2 to vnet3"

curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
5","priority":"5","in_port":"2","src-ip":"192.168.2.11","dst-
ip":"192.168.2.12","actions":"output=3"}'
http://192.168.2.3:8080/wm/staticflowpusher/json

sleep 2

echo "Create Flow for packets from vnet3 to vnet2"

curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
6","priority":"6","in_port":"3","src-ip":"192.168.2.12","dst-
ip":"192.168.2.11","actions":"output=2"}'
http://192.168.2.3:8080/wm/staticflowpusher/json
```

Script βήματος 6

```
echo "Create Flow for packets from vnet1 to vnet2"

curl -s -d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
1","priority":"1","in_port":"1","src-ip":"192.168.2.10","dst-
ip":"192.168.2.11","actions":"enqueue=2:1"}'
http://192.168.2.3:8080/wm/staticflowpusher/json

sleep 2

echo "Create Flow for packets from vnet2 to vnet1"
```

```
curl –s –d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
2","priority":"2","in_port":"2","src-ip":"192.168.2.11","dst-
ip":"192.168.2.10","actions":"enqueue=1:1"}'
http://192.168.2.3:8080/wm/staticflowpusher/json

sleep 2

echo "Create Flow for packets from vnet1 to vnet3"

curl –s –d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
3","priority":"3","in_port":"1","src-ip":"192.168.2.10","dst-
ip":"192.168.2.12","actions":"enqueue=3:1"}'
http://192.168.2.3:8080/wm/staticflowpusher/json

sleep 2

echo "Create Flow for packets from vnet3 to vnet1"

curl –s –d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
4","priority":"4","in_port":"3","src-ip":"192.168.2.12","dst-
ip":"192.168.2.10","actions":"enqueue=1:1"}'
http://192.168.2.3:8080/wm/staticflowpusher/json

sleep 2

echo "Create Flow for packets from vnet2 to vnet3"

curl –s –d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
5","priority":"5","in_port":"2","src-ip":"192.168.2.11","dst-
ip":"192.168.2.12","actions":"enqueue=3:1"}'
http://192.168.2.3:8080/wm/staticflowpusher/json

sleep 2

echo "Create Flow for packets from vnet3 to vnet2"

curl –s –d '{"switch":"00:00:16:d8:cc:fb:04:42","name":"flow-
6","priority":"6","in_port":"3","src-ip":"192.168.2.12","dst-
ip":"192.168.2.11","actions":"enqueue=2:1"}'
http://192.168.2.3:8080/wm/staticflowpusher/json
```

Κεφάλαιο 7: Συμπεράσματα

Στην παρούσα πτυχιακή εργασία παρουσιάσαμε αναλυτικά το θεωρικό υπόβαθρο για τις τεχνολογίες του Software-Defined Networking καθώς και του OpenFlow παρουσιάζοντας τις λεπτομέρειες ξεχωριστά για την κάθε τεχνολογία. Δώσαμε βάση στη εξομοίωση του SDN κάνοντας έρευνα για τους κορυφαίους δικτυακούς εξομοιωτές που υπάρχουν και επιλέξαμε τον καλύτερο σε σχέση με την ανάγκη που υπήρχε για την πτυχιακή. Κάναμε ανάλυση των εργαλείων που θα χρησιμοποιούσαμε για την υλοποίηση, προσφέροντας μεγαλύτερη εξοικείωση με τα εργαλεία.

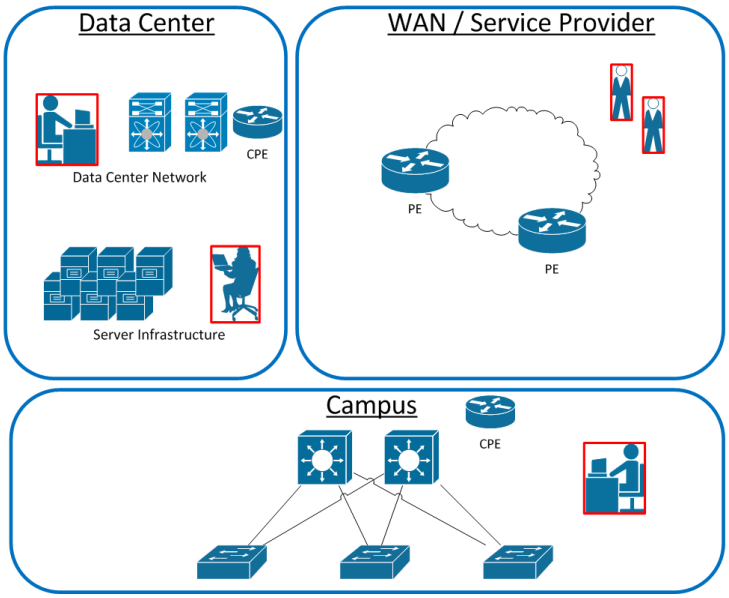
Πέρα όμως των θεωρικών πληροφοριών που μας εμβάθυναν στην τεχνολογία που μελετάμε, παρουσιάσαμε την υλοποίηση του SDN σε δύο στάδια. Το πρώτο ήταν πειραματικό και είχε σκοπό την εξοικείωση με τα συστήματα υλοποίησης καθώς και να κατανοήσουμε πως λειτουργεί στην πράξη η θεωρία γι’ αυτό την παρουσιάστηκε η διαδικασία εγκατάστασης και η διαδικασία προσομοίωσης σε ένα δίκτυο αποτελούμενο από 2 τελικούς χρήστες και 3 μεταγωγείς openflow που συνδεόταν μεταξύ τους για την επικοινωνία των τελικών χρηστών με την τεχνολογία του SDN. Το δεύτερο στάδιο είναι με τη χρήση του SDN να προσομοιάσουμε ένα πραγματικό δίκτυο σε ένα περιβάλλον εικονικοποίησης (virtualization), το οποίο αναφέρεται στην τοπολογία ενός σχολικού δικτύου παίρνοντας πληροφορίες από το Πανελλήνιο Σχολικό Δίκτυο. Προσεγγίσαμε με επιτυχία την υλοποίηση ενός τέτοιου δικτύου με την χρήση του SDN και αναλογιζόμενοι την παροχή ποιότητας των υπηρεσιών (QoS) στους τελικούς χρήστες, υλοποιήσαμε μετά από ερευνητική ανάλυση μία πολιτική δικτύου που μας προσφέρει QoS.

7.1 Χρησιμότητα

Η σημασία ύπαρξης της ποιότητας της υπηρεσίας (QoS) μέσα σε ένα περιβάλλον δικτύου είναι αρκετά σημαντική και χρήσιμη. Για αυτό το λόγο την θεωρήσαμε ως μία περίπτωση χρήσης της υλοποίησης του SDN γιατί μπορεί να λύσει αποτελεσματικά τα προβλήματα QoS που υπάρχουν σε έναν μεγάλο αριθμό δικτύων.

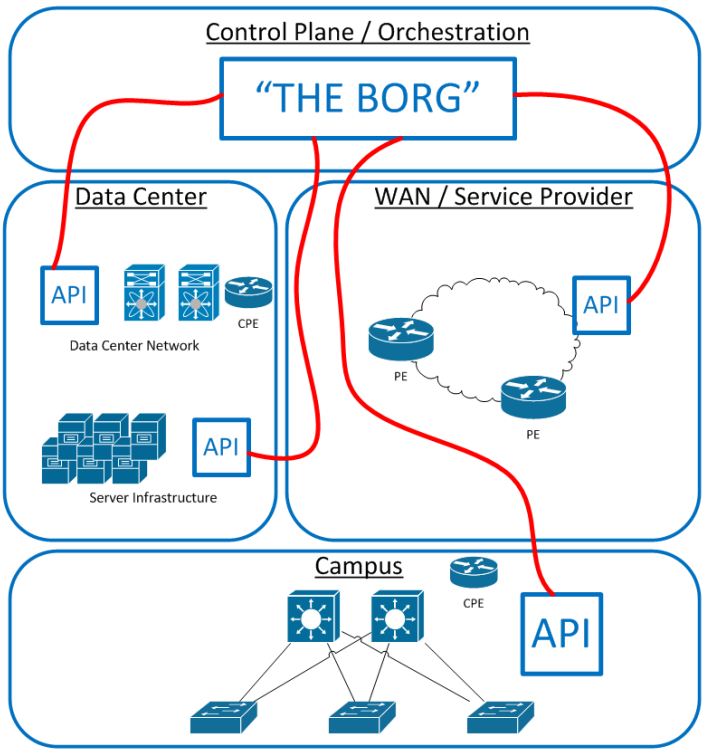
Το δίκτυο μέσα σε μία μικρή ή μεγάλη επιχείρηση προσεγγίζεται ως κάτι ξεχωριστό και υποστηρίζεται από άτομα που διαθέτουν συγκεκριμένες γνώσεις και δυνατότητες για να κάνουν εφαρμογή της τεχνολογίας αυτής μέσα στην επιχείρηση. Βέβαια εάν υπάρχει στο δίκτυο αυτό ένα Data Center, μία υποδομή για Εξυπηρετητές/Servers, μία υποδομή για αποθήκευση δεδομένων, κτλ θα

απαιτούνταν και άλλα άτομα από αυτά που διαχειρίζονται το παραπάνω δίκτυο λόγω, των διαφορετικών γνώσεων και δυνατοτήτων του κάθε ανθρώπου. Όμως για την υλοποίηση μίας πολιτικής QoS μέσα στην επιχείρηση όλα τα παραπάνω έχουν συγκεκριμένο ρόλο και πρέπει να ληφθούν υπόψη.



Εικόνα 30: Τεχνικοί Δικτύων

Αυτά βέβαια ισχύουν αναλογιζόμενοι ότι η παραπάνω τεχνολογία βρίσκεται οργανωμένη μέσα στην επιχείρηση ή οργανισμούς (Data Center, Παροχών Τηλεπικοινωνίας, Πανεπιστημιακά Δίκτυα, Σχολικά Δίκτυα, κτλ) και όχι κάπου σαν λύση που παρέχεται από κάποιες εταιρίες που εφαρμόζουν τέτοιες υλοποιήσεις τεχνολογίας, όπου οι τεχνικοί είναι παρόν μέχρι να υλοποιήσουν το δίκτυο.



Εικόνα 31: Χρησιμότητα API

Ως αποτέλεσμα των παραπάνω είναι μία πολιτική ποιότητας της υπηρεσίας (QoS) να είναι δύσκολη διαδικασία. Το SDN όμως προσφέρει πολλά οφέλη λύνοντας τα παραπάνω κύρια προβλήματα και υλοποιεί σενάρια QoS εύκολα από ένα κεντρικό επίπεδο ελέγχου γιατί δίνει πραγματική αλληλεπίδραση ρυθμίσεων σε κάθε κόμβο του δικτύου, όπως δρομολογητές, μεταγωγείς, εξυπηρετητές, εξισορρόπησης φορτίου, firewalls, κ.λ.π.

Φυσικά, το SDN δεν είναι η μοναδική λύση για όλα τα προβλήματα στο χώρο των δικτύων. Σίγουρα θα βοηθήσει στην ανάγκη λιγότερων ρυθμίσεων μέσα από τις κονσόλες (CLI) των κόμβων του δικτύου, αλλά η στήριξη από οργανισμούς και η μετάβασή τους σε μοντέλα SDN θα βοηθήσουν στην εξέλιξή του καθώς θα ανακαλύπτονται μία-μία περισσότερες λειτουργίες του.

Βιβλιογραφία

[1] Open Networking Foundation, «Software-defined networking: The new norm,» ONF White Paper, Apr. 2012.

[2] Open Networking Foundation, «Software-Defined Networking (SDN) Definition,» 25 May 2014. [Ηλεκτρονικό]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition.

[3] «Software-Defined Networking: A Comprehensive Survey,» σε *Proceedings of the IEEE*, Contributed Papers, 2015.

[4] EXTREME NETWORKS, «Different Take on SDN,» 2014.

[5] N. Feamster, «https://www.youtube.com,» M1.2 SDN History: Programmable Networks, 2014. [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=CFNTzni1FhA&list=PLpherdrLyny8YN4M24iRJBMCXkLcGbmhY&index=3.

[6] N. Feamster, «https://www.youtube.com,» M1.3 SDN History: Network Virtualization, 2014. [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=vsbyyNFg5BM&list=PLpherdrLyny8YN4M24iRJBMCXkLcGbmhY&index=4.

[7] N. Feamster, «https://www.youtube.com,» M1.4: SDN History: Control of Packet-Switched Networks, 2014. [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=AFj-ZIuAGwo&list=PLpherdrLyny8YN4M24iRJBMCXkLcGbmhY&index=5.

[8] «Software-Defined Networking: State of the Art and Research Challenges,» 2014.

[9] C. E. R. M. F. M. Carlos A. B. Macapuna, «In-packet Bloom filter based data center networking with distributed OpenFlow controllers,» 2010.

[10] A. Sonba και H. Abdalkreim, *Performance Comparison Of the state of the art Openflow Controllers,* 2014.

[11] G. Kontesidou και K. Zarifis, *Openflow Virtual Networking: A Flow-Based Network Virtualization Architecture,* 2009.

[12] B. Heller, OpenFlow Switch Specifications version 1.0.0, 2009.

[13] W. Braun και M. Menth, «Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices,» 2014.

[14] «Mininet – An Instant Virtual Network on your Laptop (or other PC),» [Ηλεκτρονικό]. Available: http://mininet.org/ and https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.

[15] B. Heller, OpenFlow Switch Specifications version 1.1.0, 2011.

[16] Open Networking Foundation, OpenFlow Switch Specifications version 1.2.0, 2011.

[17] Open Networking Foundation, OpenFlow Switch Specifications version 1.3.0, 2013.

[18] Open Networking Foundation, OpenFlow Switch Specifications version 1.4.0, 2014.

[19] «List of OpenFlow Software Projects,» [Ηλεκτρονικό]. Available:

https://yuba.stanford.edu/~casado/of-sw.html.

[20] «Opendaylight Controller,» [Ηλεκτρονικό]. Available: https://www.opendaylight.org/.

[21] S.-Y. Wang, «Comparison of SDN OpenFlow Network Simulator and Emulators: EstiNet vs. Mininet,» 2014.

[22] S. Wang, C. Chou και C. Yang, «EstiNet 8.0 OpenFlow Network Simulator and Emulator,» IEEE Communication Magazine, Vol. 51, Issue 9, 2013.

[23] B. Lantz, B. Heller και N. McKeown, «A Network in a Laptop: Rapid Prototyping for Software-Defined Networks,» 2010.

[24] S. Wang και H. Kung, «A Simple Methodology for Constructing Extensible and High-Fidelity TCP/IP Network Simulator,» 1999.

[25] S. Wang, C. Chou και C. Lin, «The Design and Implementation of the NCTUns Network Simulation Engine,» σε *Simulation Modelling Practice and Theory*, 2007.

[26] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown και S. Shenker, «NOX: towards an Operating System for Networks,» σε *ACM SIGCOMM Computer Communication Review, Volume 38 Issue 3*, 2008.

[27] «Ryu, a component-based software defined networking framework,» [Ηλεκτρονικό]. Available: http://osrg.github.io/ryu/.

[28] «Floodlight OpenFlow controller,» [Ηλεκτρονικό]. Available: http://www.projectfloodlight.org/floodlight/.

[29] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado και S. Shenker, «Extending Networking into the Virtualization Layer,» 2009.

[30] S. RAO, «SDN Series Part Five: Floodlight, an OpenFlow Controller,» 01 2015. [Ηλεκτρονικό]. Available: http://thenewstack.io/sdn-series-part-v-floodlight/.

[31] Ρ. ΧΡΙΣΤΟΦΟΡΟΥ, «Ανάπτυξη λογισμικού με χρήση ελεγκτή OpenFlow (OF controller) και υλοποίηση μηχανισμών δρομολόγησης πακέτων,» 2014.

[32] «https://www.youtube.com,» [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=vsbyyNFg5BM&list=PLpherdrLyny8YN4M24iRJBMCXkLcGbmhY&index=4.