Πτυχιακή Εργασία

«Software Defined Networking»

Κακαβάς Θάνος Α.Μ.:1394 &

Κρητικοπούλου Ελπίδα-Μαρία Α.Μ:1953

Επιβλέπων Καθηγητής: Μιχάλης Παρασκευάς

Αντίρριο, Οκτώμβριος 2018

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Αντίρριο, 29/10/2018

«Software Defined Networking»

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Παρασκευάς Μιχάλης (εισηγητής)

2. Κίτσος Πάρης

3. Τσακανίκας Βασίλης

Ευχαριστίες

Θέλουμε να ευχαριστήσουμε τον καθηγητή μας κ. Μιχαήλ Παρασκευά για την επίβλεψη της παρούσας πτυχιακής εργασίας.

Επίσης θέλουμε να ευχαριστήσουμε τις οικογενειές μας για την στήριξη, τις συμβουλές τους και την καθοδήγησή τους καθ' όλη την διάρκεια των σπουδών μας.

Συντομογραφίες

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

Πίνακας Περιεχόμενων

Πίνακας σχημάτων

Πίνακας εικόνων

Πίνακας πινάκων

Περίληψη

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

Software-based Networking Technology (SDN) is a computer networking approach that allows network administrators to initialize, control, change, and manage network behavior dynamically through open interfaces, and to provide lower-level functionality. SDN technology aims to address the fact that the static architecture of traditional networks does not support dynamic, scalable computing and storage needs of more modern computing environments such as data centers. This is done by disconnecting the decision-making system (SDN controller or control level) from the underlying systems that promote traffic to the selected destination (the data layer) SD-WAN is a broad network (WAN) that is managed using the network-defined principles defined by the software.

The primary responsibility of SD-WAN is to reduce WAN cost by using more affordable and commercially available leased lines as an alternative or partial replacement of the more expensive MPLS lines. Control and management are separated from the hardware with centralized controllers that make it easier to configure and manage.

In a true OpenFlow network, each OpenFlow switch has to create a TCP connection with a real OpenFlow controller. With the capabilities offered by the KR methodology, in a simulated OpenFlow network at EstiNet, any OpenFlow switch simulation can establish a true TCP connection with a real OpenFlow controller and the processing of protocols between the OpenFlow controller and the OpenFlow switch is exactly the same such as by processing protocols to set up a TCP connection between 2 real end users (hosts).

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 επιχείρησης. Τέλος, θα γίνει η τροφοδότηση ροών κίνησης στο δίκτυο της επιχείρησης, 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 ικονοποίησης (Virtualbox) και του εικονικού μεταγωγέα (OpenVSwitch) που έχουμε επιλέξει για την υλοποίηση.

Τέλος, στα τρία κεφάλαια που υπολείπονται έχουμε την Αναφορά SDN σε περιβάλλον δικτύου επιχείρησης, την Προσομοίωση Νησίδας SDN (πειραματικό στάδιο), την Προσομοίωση πραγματικού δικτύου 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: Δικτύωση Καθοριζόμενη από Λογισμικό (Software-Defined Networking)

Η τεχνολογία SDN συσχετίζεται συνήθως με το πρωτόκολλο OpenFlow (για την απομακρυσμένη επικοινωνία με στοιχεία επιπέδου δικτύου για τον προσδιορισμό της διαδρομής των πακέτων δικτύου σε όλους τους διακόπτες δικτύου) από την εμφάνιση του τελευταίου το 2011 (Halepidis et al., 2015). Από το 2012 πολλές εταιρείες έχουν απομακρυνθεί από το OpenFlow και έχουν υιοθετήσει διαφορετικές τεχνικές. Αυτές περιλαμβάνουν το Open Network Environment της Cisco Systems και την πλατφόρμα εικονικοποίησης δικτύου της Nicira. Το σύστημα SD-WAN εφαρμόζει παρόμοια τεχνολογία σε ένα δίκτυο ευρείας περιοχής (WAN) (Lerenr, 2015). (Halepidis et al., 2015).

1.1 Ιστορικό

Ένα από τα πρώτα έργα SDN ήταν το GeoPlex της AT&T (AZIZ and Okamura, 2016). Τα μέλη του προγράμματος AT&T LabsGeoplex, Micha Lerner, George Vanecek, Nino Vidovic και Dado Vrsalovic αξιοποίησαν τα API του δικτύου και τις δυναμικές πτυχές της γλώσσας Java ως μέσο υλοποίησης των μεσισμικών δικτύων (middleware). «Το GeoPlex δεν είναι ένα λειτουργικό σύστημα, ούτε επιχειρεί να ανταγωνιστεί κάποιο. Είναι ένα μεσισμικό λογισμικό δικτύωσης που χρησιμοποιεί ένα ή περισσότερα λειτουργικά συστήματα που εκτελούνται σε υπολογιστές συνδεδεμένους στο Διαδίκτυο. Το GeoPlex είναι μια πλατφόρμα υπηρεσιών που διαχειρίζεται δίκτυα και διαδικτυακές υπηρεσίες. Χαρτογραφεί όλες τις δραστηριότητες δικτύου IP σε μία ή περισσότερες υπηρεσίες» (Jadeja and Modi, 2012).

Το GeoPlex δεν ασχολήθηκε με τα λειτουργικά συστήματα που λειτουργούν με διακόπτες υλικού δικτύωσης και δρομολογητές. Η AT&T ήθελε έναν «διακόπτη λογισμικού» που θα μπορούσε να αναδιαμορφώσει τους φυσικούς διακόπτες στο

δίκτυο και να τους φορτώσει με νέες υπηρεσίες από ένα σύστημα υποστήριξης λειτουργιών (OSS). Ωστόσο, όταν παρέχονται οι υπηρεσίες το GeoPlex δεν μπόρεσε να φτάσει βαθιά στις φυσικές συσκευές για να εκτελέσουν την αναδιάρθρωση. Τα λειτουργικά συστήματα που λειτουργούν σε δικτυωμένες συσκευές στο φυσικό δίκτυο, επομένως, αποτέλεσαν εμπόδιο στην παράδοση υπηρεσιών τύπου SDN.

Το 1998, ο Mark Medovich, επιστήμονας της Sun Microsystems και της Javasoft, εγκατέλειψε την Sun για να ξεκινήσει την WebSprocket μια νεοφυή εταιρεία παραγωγής διακόπτη λογισμικού στην Silicon Valley. Η Medovich σχεδίασε ένα νέο λειτουργικό σύστημα δικτύου και ένα αντικειμενοστρεφές δομημένο μοντέλο χρόνου εκτέλεσης που μπορούσε να τροποποιηθεί από έναν δικτυακό μεταγλωττιστή και έναν φορτωτή κλάσης σε πραγματικό χρόνο. Με αυτήν την προσέγγιση, οι εφαρμογές θα μπορούσαν να γραφτούν με νήματα σε Java που απέκτησαν τις κλάσεις του πυρήνα, του δικτύου και της συσκευής του WebSprocket και αργότερα τροποποιήθηκαν από τον δικτυωμένο μεταγλωττιστή / φορτωτή κλάσης. Η πλατφόρμα του WebSprocket σχεδιάστηκε έτσι ώστε οι συσκευές να έχουν την δυνατότητα να δημιουργούν σε στοίβες δικτύου, διασυνδέσεις και πρωτόκολλα ως πολλαπλά νήματα (Cheung, Thomas and Patrick, 2010).

Τον Ιούλιο του 2000, η WebSprocket κυκλοφόρησε το VMFoundry, έναν μεταγλωττιστή εκτέλεσης Java και το VMServer, έναν μεταγλωττιστή δικτυωμένων συσκευών / φορτωτή κλάσης διακομιστή εφαρμογών (Boon, 2017). Στις προσαρμοσμένες δικτυωμένες συσκευές υπήρχαν προεγκατεστημένες εικόνες που δημιουργήθηκαν από την VMFoundry και στην συνέχεια αναπτύχθηκαν στο δίκτυο και συνδέθηκαν με τον VMServer μέσω του User Datagram Protocol (UDP) ή του Transmission Control Protocol (TCP), το οποίο θα μπορούσε προληπτικά ή αντιδραστικά να φορτώσει ή να επεκτείνει τις μεθόδους του πρωτοκόλλου του δικτύου και τις κλάσεις στο στοχευόμενο σύστημα.

Επομένως, η έκδοση του SDN του WebSprocket δεν περιοριζόταν σε μια σειρά περιορισμένων ενεργειών που διαχειρίζεται ένας ελεγκτής SDN. Αντίθετα, το επίπεδο ελέγχου του WebSprocket περιείχε κώδικα που θα μπορούσε να αλλάξει, να παρακάμψει, να επεκτείνει ή να ενισχύσει τα πρωτόκολλα δικτύου σε

λειτουργικά δικτυωμένα συστήματα (MaacCaw, 2011). Ο Bill Yount (του δικτύου του πανεπιστημίου Stanford) επισκέφθηκε το εργαστήριο Sunnyvale της WebSprocket για να παρακολουθήσει μία επίδειξη και εξέφρασε ενθουσιασμό για την όλη ιδέα, ειδικά για το VMServer (SDN Controller) και προφητικά δήλωσε ότι η τεχνολογία SDN (της WebSprocket) είναι «10 χρόνια μπροστά από την εποχή της». Το καλοκαίρι του 2000, οι μηχανικοί έρευνας της Ericsson εντόπισαν μία άμεση ανάγκη και επισκέφθηκαν την WebSprocket για να σχεδιάσουν και να αρχικοποιήσουν τα χαρακτηριστικά ενός διακόπτη λογισμικού επόμενης γενιάς, κάνοντας τα πρώτα βήματα για την κατασκευή του πρώτου εμπορικού διακόπτη λογισμικού στον κόσμο.

Κάποια στιγμή κατά την διάρκεια του 2000, ο Όμιλος Gartner εισήγαγε το «Supranet», μία σύντηξη του φυσικού και του ψηφιακού (εικονικού) κόσμου ως ένα «διαδίκτυο των πραγμάτων» και τον Οκτώβριο του 2000 η ομάδα Gartner επέλεξε το WebSprocket ως μία από τις κορυφαίες αναδυόμενες τεχνολογίες (Van Osch and Avital, 2010).

Στις αρχές του 2001, η Ericsson και η WebSprocket συνήψαν σύμβαση για την δημιουργία του πρώτου εμπορικού διακόπτη λογισμικού. Δημιουργήθηκε μια διεθνής κοινοπραξία για την ανάπτυξη προτύπων για το Supranet. Τον Μάρτιο του 2001, ο Kurt Dewitt, Πρόεδρος της Consortium Supranet και Διευθυντής Επιχειρηματικής Ανάπτυξης της Διεύθυνσης Ευρυζωνικών Δεδομένων και Οπτικών Δικτύων της Ericsson, ανακοίνωσε την επιλογή του WebSprocket ως την τεχνολογία ενεργοποίησης του Supranet Transaction Server (STS), ενός ολοκληρωμένου πλαισίου για την παροχή οποιασδήποτε δικτυακής υπηρεσίας (Sandri, 2015).

Τον Απρίλιο και τον Μάιο του 2001, το Πανεπιστήμιο του Οχάιο και η OARnet διεξήγαγαν την πρώτη δοκιμή του SDN και ανέπτυξαν την πρώτη πρακτική χρήση του SDN για το Internet2. Μετά την επιτυχή ολοκλήρωση των δοκιμών, η OARnet εξέδωσε την ακόλουθη δήλωση στις 8 Μαΐου 2001:

«Έχουμε παρακολουθήσει το πρώτο επιτυχημένο βήμα στην εκπλήρωση έξυπνων, διαλειτουργικών δικτύων μέσω της ανάπτυξης του Supranet Transaction Server. Η τεχνολογία ολοκληρώθηκε καθώς ένα νέο σύνολο οδηγιών μεταδόθηκε

δυναμικά σε όλο το δίκτυο, αλλάζοντας την συμπεριφορά του αιτούντος υπολογιστή. Δεν υπήρχε ανάγκη να καταργηθεί οποιοδήποτε μέρος του συστήματος και δεν υπήρξε διακοπή της υπηρεσίας. Η δοκιμή μας θα συνεχιστεί και αναμένουμε περαιτέρω πρόοδο της επόμενης γενιάς Διαδικτύου μέσω της συνεργασίας μας με την Websprocket», δήλωσε ο Pankaj Shah (Διευθύνων Σύμβουλος της OARnet) (Kim and Feamster, 2013).

Η αγορά τηλεπικοινωνιών ξεφούσκωσε το 2001 και το πρόγραμμα ανάπτυξης του διακόπτη λογισμικού της Ericsson ήλθε στο τέλος του, περιορίζοντας έτσι την μοναδική εμπορική προσπάθεια E&A στην τεχνολογία διακόπτη λογισμικού SDN για τότε.

Η δικτύωση που καθορίζεται από το λογισμικό συνεχίστηκε με εργασίες που πραγματοποιήθηκαν το 2003 από τους Bob Burke και Zac Carman για την ανάπτυξη της αίτησης για δίπλωμα ευρεσιτεχνίας για το Network Delivery Control Network, που τελικά εκδόθηκε ως δύο διπλώματα ευρεσιτεχνίας των Η.Π.Α. (Murphy et al., 2010): 8.122.128 και 8.799.468 Σε αυτήν την ιδέα, η τεχνολογία SDN που ονομάστηκε αρχιτεκτονική προτίμησης υπηρεσίας (SPA) στην πατέντα τους, περιγράφηκε ως μία συλλογή τεχνικών υπολογιστικής ενσωμάτωσης που χρησιμοποιούνται για τον έλεγχο της λειτουργίας των στοιχείων δικτύου, δηλαδή των διακομιστών περιεχομένου, των δρομολογητών, διακοπτών και πυλών, την προστασία του περιεχομένου από την κλοπή (P2P) ή την ανεπιθύμητη παρακολούθηση και την αποτελεσματική παράδοση περιεχομένου για πληρωμένες υπηρεσίες. Η CableLabs καθόρισε αργότερα το Digital Cable και το CableCARD χρησιμοποιώντας αυτό που τώρα γνωρίζουμε ως SDN, το οποίο έκανε το ντεμπούτο του το 2007. Η τεχνολογία SDN προχώρησε και πάλι στην εργασία που έγινε στο πανεπιστήμιο Berkeley και στο Πανεπιστήμιο του Στάνφορντ γύρω στο 2008. Το Ίδρυμα Open Networking ιδρύθηκε το 2011 για να προωθήσει το SDN και το OpenFlow (Shenker, 2012).

Στην έκθεση Interop και Tech Field 2014, η δικτύωση που καθορίζεται από το λογισμικό παρουσιάστηκε από την Avaya χρησιμοποιώντας την συντομότερη πορεία γεφύρωσης και το OpenStackas μια αυτοματοποιημένη πανεπιστημιούπολη, επεκτείνοντας την αυτοματοποίηση από το κέντρο δεδομένων στην τελική συσκευή,

αφαιρώντας την μην αυτόματη παροχή από την υπηρεσία (Infotechlead, 2014). Μέχρι το 2016, κάποιοι στον κλάδο θεωρούσαν ότι η τεχνολογία SDN είχε μετατραπεί σε έναν ασήμαντο όρο του μάρκετινγκ (Suffolk, 2012).

1.2 Η ιδέα

Η δικτύωση που καθορίζεται από το λογισμικό (SDN) είναι μια αρχιτεκτονική που χαρακτηρίζεται δυναμική, διαχειρίσιμη, οικονομικά αποδοτική και προσαρμόσιμη, επιδιώκοντας να είναι κατάλληλη για την δυναμική φύση των σημερινών εφαρμογών. Οι αρχιτεκτονικές SDN αποσυνδέουν τις λειτουργίες ελέγχου και προώθησης του δικτύου, επιτρέποντας τον άμεσο προγραμματισμό του ελέγχου του δικτύου και της υποκείμενης υποδομής από τις εφαρμογές και τις υπηρεσίες δικτύου (ONF, 2017).

Το πρωτόκολλο OpenFlow μπορεί να χρησιμοποιηθεί σε τεχνολογίες SDN. Η αρχιτεκτονική SDN είναι:

- Άμεσα προγραμματιζόμενη: Ο έλεγχος δικτύου είναι άμεσα προγραμματιζόμενος επειδή αποσυνδέεται από τις λειτουργίες προώθησης.
- Ευκίνητη: Η αφαίρεση του ελέγχου από την προώθηση επιτρέπει στους διαχειριστές να προσαρμόζουν δυναμικά την ροή της κυκλοφορίας σε όλο το δίκτυο για να ανταποκρίνεται στις μεταβαλλόμενες ανάγκες.
- Κεντρικά διαχειριζόμενη: Η νοημοσύνη του δικτύου (λογικά) συγκεντρώνεται σε ελεγκτές SDN που βασίζονται σε λογισμικό και διατηρούν μια συνολική εικόνα του δικτύου, η οποία εμφανίζεται σε εφαρμογές και μηχανές πολιτικής ως ένας και μόνος λογικός διακόπτης.
- Ρυθμιζόμενη με προγραμματισμό: Η τεχνολογία SDN επιτρέπει στους διαχειριστές δικτύου να ρυθμίζουν, να διαχειρίζονται, να ασφαλίζουν και να βελτιστοποιούν πολύ γρήγορα τους πόρους του δικτύου μέσω δυναμικών αυτοματοποιημένων προγραμμάτων SDN, τα οποία μπορούν να γράψουν οι ίδιοι, επειδή τα προγράμματα δεν εξαρτώνται από ιδιόκτητο λογισμικό.

- Βασίζεται σε ανοικτά πρότυπα και είναι ανεξάρτητη πωλητών: Όταν εφαρμόζεται μέσω ανοικτών προτύπων, η τεχνολογία SDN απλοποιεί τον σχεδιασμό και την λειτουργία του δικτύου, διότι οι οδηγίες παρέχονται από τους ελεγκτές SDN αντί της ύπαρξης πολλαπλών συσκευών και πρωτοκόλλων από συγκεκριμένους πωλητές.

1.3 Η ανάγκη για μια νέα αρχιτεκτονική δικτύου

Η έκρηξη των κινητών συσκευών και περιεχομένου, η εικονικοποίηση των διακομιστών και η εμφάνιση των υπηρεσιών νέφους (cloud) συγκαταλέγονται στις τάσεις που οδηγούν την βιομηχανία δικτύωσης στο να επανεξετάσει τις παραδοσιακές αρχιτεκτονικές δικτύων (ONF, 2017). Πολλά συμβατικά δίκτυα είναι ιεραρχικά, χτισμένα με σειρές διακοπτών Ethernet τοποθετημένους σε δομή δέντρου. Αυτός ο σχεδιασμός είχε νόημα όταν κυριαρχούσε το υπολογιστικό μοντέλο πελάτης-διακομιστής, αλλά μια τέτοια στατική αρχιτεκτονική δεν είναι κατάλληλη για τις δυναμικές ανάγκες υπολογιστικής και αποθήκευσης των σημερινών κέντρων δεδομένων επιχειρήσεων, πανεπιστημιουπόλεων και φορέων (Montazerolghaem, Yaghmaee and Leon-Garcia, 2017). Ορισμένες από τις βασικές τάσεις υπολογιστικής που οδήγησαν στην ανάγκη για ένα νέο μοντέλο δικτύου περιλαμβάνουν:

Αλλαγή προτύπων κυκλοφορίας: Στο εσωτερικό του κέντρου δεδομένων της επιχείρησης, τα πρότυπα κυκλοφορίας έχουν αλλάξει σημαντικά. Σε αντίθεση με τις εφαρμογές πελάτη-διακομιστή όπου ο κύριος όγκος επικοινωνίας συμβαίνει μεταξύ ενός πελάτη και ενός διακομιστή, οι σημερινές εφαρμογές έχουν πρόσβαση σε διαφορετικές βάσεις δεδομένων και διακομιστές, δημιουργώντας μια αναταραχή της κυκλοφορίας μηχανής προς μηχανή του μοντέλου «ανατολής-δύσης» πριν επιστρέψουν τα δεδομένα στον τελικό χρήστη στο κλασικό μοντέλο κυκλοφορίας «βορρά-νότου». Ταυτόχρονα, οι χρήστες αλλάζουν τα πρότυπα κυκλοφορίας δικτύου καθώς πιέζουν για πρόσβαση σε εταιρικό περιεχόμενο και εφαρμογές από οποιοδήποτε τύπο συσκευής (συμπεριλαμβανομένων των δικών τους), συνδεόμενοι από οπουδήποτε και ανά πάσα στιγμή. Τέλος, πολλοί διαχειριστές κέντρων

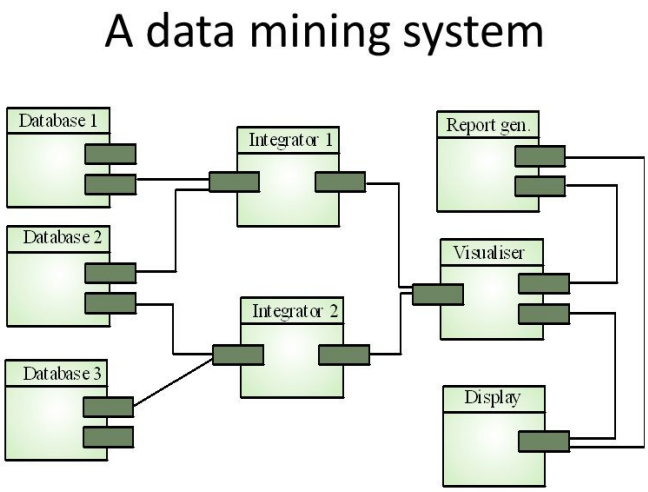
δεδομένων επιχειρήσεων σκέφτονται ένα μοντέλο υπολογιστικής χρησιμότητας, το οποίο μπορεί να περιλαμβάνει ένα ιδιωτικό νέφος, ένα δημόσιο νέφος ή κάποιο μίγμα και των δύο, με αποτέλεσμα την πρόσθετη κυκλοφορία σε όλο το δίκτυο ευρείας περιοχής.

Η «καταναλωτικοποίηση της τεχνολογίας της πληροφορίας»: Οι χρήστες χρησιμοποιούν ολοένα και περισσότερο προσωπικές κινητές συσκευές, όπως τα smartphones, τα tablet και τα notebooks για πρόσβαση στο εταιρικό δίκτυο. Η τεχνολογία της πληροφορίας υφίσταται πίεση για να φιλοξενήσει αυτές τις προσωπικές συσκευές με ραφιναρισμένο τρόπο, προστατεύοντας παράλληλα τα εταιρικά δεδομένα και την πνευματική ιδιοκτησία και ικανοποιώντας τις εντολές συμμόρφωσης.

Η άνοδος των υπηρεσιών νέφους: Οι επιχειρήσεις έχουν αγκαλιάσει με ενθουσιασμό τόσο τις δημόσιες όσο και τις ιδιωτικές υπηρεσίες νέφους, με αποτέλεσμα την άνευ προηγουμένου ανάπτυξη αυτών των υπηρεσιών. Οι επιχειρησιακές μονάδες επιθυμούν τώρα την ευκινησία να έχουν πρόσβαση σε εφαρμογές, υποδομές και άλλους πόρους πληροφορικής κατόπιν αιτήματος και κατ' επιλογήν. Για να αυξηθεί η πολυπλοκότητα, ο σχεδιασμός της τεχνολογίας της πληροφορίας για υπηρεσίες νέφους θα πρέπει να γίνει σε ένα περιβάλλον αυξημένων απαιτήσεων ασφάλειας, συμμόρφωσης και ελέγχου, μαζί με αναδιοργανώσεις επιχειρήσεων, ενοποιήσεις επιχειρήσεων και συγχωνεύσεις που μπορούν να αλλάξουν τις παραδοχές εν μία νυκτί. Η παροχή υπηρεσιών αυτοεξυπηρέτησης, είτε σε ιδιωτικό είτε σε δημόσιο νέφος, απαιτεί ελαστική επέκταση της υπολογιστικής, της αποθήκευσης και των πόρων δικτύου, ιδανικά από μια κοινή οπτική γωνία και με μια κοινή σουίτα εργαλείων.

Τα μεγάλα δεδομένα σημαίνουν περισσότερο εύρος ζώνης: Η διαχείριση των σημερινών μεγάλων δεδομένων ή μεγάλων συνόλων δεδομένων απαιτεί μαζική παράλληλη επεξεργασία σε χιλιάδες εξυπηρετητές, οι οποίοι χρειάζονται άμεσες συνδέσεις μεταξύ τους. Η άνοδος των μεγάλων συνόλων δεδομένων τροφοδοτεί την συνεχή ζήτηση για πρόσθετη χωρητικότητα δικτύου στο κέντρο δεδομένων. Οι φορείς εκμετάλλευσης μεγάλων δικτύων κέντρων δεδομένων αντιμετωπίζουν το δύσκολο έργο της επέκτασης του δικτύου σε μέγεθος που δεν μπορούσαν καν να

φανταστούν, διατηρώντας την σύνδεση τύπου οποιοσδήποτε-προς-οποιονδήποτε χωρίς υψηλό κόστος.



Σχήμα 1.1: Αρχιτεκτονικά συστατικά στοιχεία
Εικόνα αρχιτεκτονικής δικτύωσης που καθορίζεται από το λογισμικό

Πηγή: Bailey et al., 2013

Τα παρακάτω παρουσιάζουν τα αρχιτεκτονικά συστατικά στοιχεία (Bailey et al., 2013):

Εφαρμογή SDN: Οι εφαρμογές SDN είναι προγράμματα που επικοινωνούν ρητά, άμεσα και προγραμματικά τις απαιτήσεις δικτύου τους και την επιθυμητή συμπεριφορά δικτύου στον ελεγκτή SDN μέσω μιας βόρειας διεπαφής (NBI). Επιπλέον, μπορούν να καταναλώνουν μια αφηρημένη άποψη του δικτύου για εσωτερικούς σκοπούς λήψης αποφάσεων. Μια εφαρμογή SDN αποτελείται από μία λογική εφαρμογής SDN και έναν ή περισσότερους οδηγούς NBI. Οι εφαρμογές SDN μπορούν να εκθέσουν οι ίδιες ένα άλλο επίπεδο αφηρημένου ελέγχου δικτύου, προσφέροντας έτσι μία ή περισσότερες βόρειες διεπαφές υψηλότερου επιπέδου μέσω αντίστοιχων πρακτόρων βόρειας διεπαφής.

Ελεγκτής SDN: Ο ελεγκτής SDN είναι μια λογικά κεντική οντότητα που είναι υπεύθυνη για α) την μετάφραση των απαιτήσεων από το επίπεδο της εφαρμογής SDN μέχρι τα SDN Datapaths και β) την παροχή των εφαρμογών SDN με μια

αφηρημένη άποψη του δικτύου (που μπορεί να περιλαμβάνει στατιστικά στοιχεία και γεγονότα). Ένας ελεγκτής SDN αποτελείται από έναν ή περισσότερους πράκτορες NBI, την Λογική Ελέγχου SDN και τον οδηγό Ελέγχου Διεπαφής Δεδομένων-Επιπέδου (CDPI). Ο ορισμός ως μια λογικά κεντρική οντότητα ούτε προδιαγράφει ούτε αποκλείει λεπτομέρειες υλοποίησης όπως την ομοσπονδία πολλών ελεγκτών, την ιεραρχική σύνδεση των ελεγκτών, τις διεπαφές επικοινωνίας μεταξύ των ελεγκτών, ούτε την εικονικοποίηση ή τον τεμαχισμό των πόρων του δικτύου.

SDN Datapath: Το Datapath SDN είναι μια λογική συσκευή δικτύου που εκθέτει την ορατότητα και τον απαράβατο έλεγχο των διαφημιζόμενων δυνατοτήτων προώθησης και επεξεργασίας δεδομένων. Η λογική αναπαράσταση μπορεί να περιλαμβάνει το σύνολο ή ένα υποσύνολο των φυσικών πόρων του υποστρώματος. Ένα SDN Datapath περιλαμβάνει έναν πράκτορα CDPI και ένα σύνολο από μία ή περισσότερες μηχανές προώθησης της κυκλοφορίας και μηδέν ή περισσότερες λειτουργίες επεξεργασίας κυκλοφορίας. Αυτές οι μηχανές και λειτουργίες μπορεί να περιλαμβάνουν την απλή προώθηση μεταξύ των εξωτερικών διεπαφών του datapath ή την εσωτερική λειτουργία επεξεργασίας ή τερματισμού της κυκλοφορίας. Ένα ή περισσότερα SDN Datapaths μπορεί να περιέχονται σε ένα μόνο στοιχείο δικτύου (φυσικό) - ένας ολοκληρωμένος φυσικός συνδυασμός πόρων επικοινωνίας τους οποίους διαχειρίζεται ως μονάδα. Ένα Datapath SDN μπορεί επίσης να οριστεί σε πολλαπλά φυσικά στοιχεία δικτύου. Αυτός ο λογικός ορισμός ούτε προδιαγράφει ούτε αποκλείει λεπτομέρειες υλοποίησης όπως την λογική προς την φυσική χαρτογράφηση, την διαχείριση των κοινών φυσικών πόρων, την εικονικοποίηση ή τον τεμαχισμό του SDN Datapath, την διαλειτουργικότητα με μη SDN δικτύωση ή την λειτουργία επεξεργασίας δεδομένων, η οποία μπορεί να περιλαμβάνει επίπεδο OSI με 4-7 λειτουργίες.

Έλεγχος SDN σε διεπαφή δεδομένων-επιπέδου (CDPI): Το SDP CDPI είναι η διεπαφή που ορίζεται μεταξύ ενός ελεγκτή SDN και ενός Datapath SDN, το οποίο παρέχει τουλάχιστον: α) προγραμματικό έλεγχο όλων των λειτουργιών προώθησης, β) διαφήμιση δυνατοτήτων, γ) αναφορά στατιστικών στοιχείων και δ) ειδοποίηση

συμβάντος. Μία αξία του SDN είναι η προσδοκία ότι το CDPI θα υλοποιηθεί με ανοικτό και διαλειτουργικό τρόπο και ανεξάρτητα από τον πωλητή.

Διασυνδέσεις SDN Northbound (NBI): Τα SDN NBIs είναι διασυνδέσεις μεταξύ εφαρμογών SDN και ελεγκτών SDN και συνήθως παρέχουν αφηρημένη ορατότητα δικτύου και επιτρέπουν την άμεση έκφραση της συμπεριφοράς και των απαιτήσεων του δικτύου. Αυτό μπορεί να συμβεί σε οποιοδήποτε επίπεδο αφαίρεσης (γεωγραφικό πλάτος) και σε διαφορετικά σύνολα λειτουργικότητας (γεωγραφικό μήκος). Μια τιμή του SDN έγκειται στην προσδοκία ότι οι διεπαφές αυτές θα υλοποιηθούν με ανοικτό και διαλειτουργικό τρόπο και ανεξάρτητα από τον πωλητή.

Κεντρική - Ιεραρχική - Κατανεμημένη: Η υλοποίηση του επιπέδου ελέγχου του SDN μπορεί να ακολουθήσει έναν κεντρικό, ιεραρχικό ή αποκεντρωμένο σχεδιασμό. Οι αρχικές προτάσεις του επιπέδου ελέγχου του SDN επικεντρώθηκαν σε μια κεντρική λύση, όπου μια ενιαία οντότητα ελέγχου έχει μια συνολική εικόνα του δικτύου. Ενώ αυτή η λύση απλοποιεί την εφαρμογή της λογικής του ελέγχου, έχει περιορισμούς σε επίπεδο επεκτασιμότητας καθώς αυξάνεται το μέγεθος και η δυναμική του δικτύου. Για να ξεπεραστούν αυτοί οι περιορισμοί, έχουν προταθεί διάφορες προσεγγίσεις στην βιβλιογραφία που εμπίπτουν σε δύο κατηγορίες, τις ιεραρχικές και τις πλήρως κατανεμημένες προσεγγίσεις. Στις ιεραρχικές λύσεις (Hassas Yeganeh and Ganjali, 2012; Ahmed and Boutaba, 2014), οι κατανεμημένοι ελεγκτές λειτουργούν σε μια διαχωρισμένη προβολή δικτύου, ενώ οι αποφάσεις που απαιτούν γνώσεις από όλο το δίκτυο λαμβάνονται από έναν λογικό κεντρικό ελεγκτή ρίζας. Στις κατανεμημένες προσεγγίσεις, οι ελεγκτές (Koponen et al., 2010; Tuncer et al., 2015) λειτουργούν με την τοπική τους οπτική γωνία ή μπορούν να ανταλλάσσουν μηνύματα συγχρονισμού για να ενισχύσουν τις γνώσεις τους. Οι κατανεμημένες λύσεις είναι πιο κατάλληλες για υποστήριξη προσαρμοστικών εφαρμογών SDN.

Τοποθέτηση ελεγκτή: Ένα βασικό ζήτημα κατά τον σχεδιασμό ενός κατανεμημένου επιπέδου ελέγχου SDN είναι να αποφασιστεί ο αριθμός και η τοποθέτηση των οντοτήτων ελέγχου. Μια σημαντική παράμετρος που πρέπει να ληφθεί υπόψη είναι η καθυστέρηση της διάδοσης μεταξύ των ελεγκτών και των συσκευών δικτύου (Heller, Sherwood and McKeown, 2012), ιδίως στο πλαίσιο μεγάλων δικτύων. Άλλοι

στόχοι που εξετάστηκαν περιλαμβάνουν την αξιοπιστία της διαδρομής ελέγχου, την ανοχή σφάλματος και τις απαιτήσεις τηε εφαρμογής (ros and Ruiz 2014; Tuncer et al., 2015).

Προορατική έναντι Αντδραστικής έναντι Υβριδικής λειτουργίας (Salisbury, 2013): Το OpenFlow χρησιμοποιεί πίνακες TCAM για την δρομολόγηση ακολουθιών πακέτων (ροές). Εάν οι ροές φθάνουν σε έναν διακόπτη, πραγματοποιείται μια αναζήτηση πίνακα ροής. Ανάλογα με την υλοποίηση του πίνακα ροής, αυτό γίνεται σε έναν πίνακα ροής λογισμικού εάν χρησιμοποιείται vSwitch ή σε ένα ASIC εάν υλοποιείται σε υλικό. Στην περίπτωση που δεν βρεθεί αντίστοιχη ροή, αποστέλλεται αίτημα στον ελεγκτή για περαιτέρω οδηγίες. Η διαχείριση του αιτήματος γίνεται με έναν από τρεις διαφορετικούς τρόπους. Σε περίπτωση αντιδραστικής λειτουργίας ο ελεγκτής λειτουργεί μετά από αυτά τα αιτήματα και δημιουργεί και εγκαθιστά έναν κανόνα στον πίνακα ροής για το αντίστοιχο πακέτο, εάν είναι απαραίτητο. Στην προορατική λειτουργία, ο ελεγκτής συμπληρώνει τις καταχωρήσεις του πίνακα ροής για όλες τις πιθανές αντιστοιχίσεις κυκλοφορίας που είναι δυνατές για αυτόν τον διακόπτη εκ των προτέρων. Αυτή η λειτουργία μπορεί να συγκριθεί με τις τυπικές καταχωρήσεις πίνακα δρομολόγησης σήμερα, όπου όλες οι στατικές καταχωρήσεις εγκαθίστανται από πριν. Μετά από αυτό δεν αποστέλλεται αίτημα στον ελεγκτή αφού όλες οι εισερχόμενες ροές θα βρουν μια αντίστοιχη καταχώρηση. Ένα σημαντικό πλεονέκτημα στην προορατική λειτουργία είναι ότι όλα τα πακέτα προωθούνται με ρυθμό γραμμής (λαμβάνοντας υπόψη όλες τις καταχωρήσεις του πίνακα ροής στο TCAM) και δεν προστίθεται καμία καθυστέρηση. Η τρίτη λειτουργία, η υβριδική, ακολουθεί την ευελιξία μιας αντιδραστικής λειτουργίας για ένα σύνολο κυκλοφορίας και την προώθηση χαμηλής καθυστέρησης (προορατική λειτουργία) για το υπόλοιπο της κυκλοφορίας.

1.4 Εφαρμογές

Η SDMN είναι μια προσέγγιση στον σχεδιασμό δικτύωσης στην κινητή τεχνολογία όπου όλες οι λειτουργίες που σχετίζονται με το πρωτόκολλο υλοποιούνται στο λογισμικό, μεγιστοποιώντας την χρήση του γενικού και βασικού υλικού και του λογισμικού τόσο στο βασικό δίκτυο, όσο και στο δίκτυο της

ασύρματης πρόσβασης (Costa-Requena et al., 2015). Προτείνεται ως επέκταση του υποδείγματος SDN για την ενσωμάτωση λειτουργικοτήτων που σχετίζονται με το δίκτυο της κινητής τεχνολογίας (Liyanage, Ylianttila and Gurtov, 2014). (Butler, 2017).

Το SD-LAN είναι ένα τοπικό δίκτυο (LAN) που βασίζεται στις αρχές της δικτύωσης που καθορίζεται από το λογισμικό, αν και υπάρχουν σημαντικές διαφορές στην τοπολογία, την ασφάλεια δικτύου, την ορατότητα και τον έλεγχο της εφαρμογής, την διαχείριση και την ποιότητα των υπηρεσιών. Το SD-LAN αποσυνδέει την διαχείριση ελέγχου και τα επίπεδα δεδομένων επιτρέποντας να αναπτυχθεί μια αρχιτεκτονική που καθοδηγείται από την πολιτική για τα ενσύρματα και ασύρματα LAN. Τα δίκτυα SD-LAN χαρακτηρίζονται από την χρήση ενός συστήματος διαχείρισης νέφους και ασύρματης σύνδεσης χωρίς την παρουσία ενός φυσικού ελεγκτή (Schulz-Zander, 2016).

1.5 Ασφάλεια χρησιμοποιώντας το πρότυπο SDN

Η αρχιτεκτονική SDN μπορεί να ενεργοποιεί, να διευκολύνει ή να ενισχύει εφαρμογές ασφάλειας που σχετίζονται με το δίκτυο, λόγω της κεντρικής όψης του δικτύου από τον ελεγκτή και της ικανότητάς του να επαναπρογραμματίζει το επίπεδο δεδομένων ανά πάσα στιγμή. Ενώ η ασφάλεια της αρχιτεκτονικής SDN παραμένει ένα ανοιχτό ερώτημα που έχει ήδη μελετηθεί κάποιες φορές από την ερευνητική κοινότητα, οι ακόλουθες παράγραφοι εστιάζονται μόνο στις εφαρμογές ασφάλειας που έγιναν δυνατές ή επανεξετάστηκαν χρησιμοποιώντας το SDN.

Αρκετές ερευνητικές εργασίες για το SDN έχουν ήδη διερευνήσει εφαρμογές ασφάλειας που βασίζονται στον ελεγκτή SDN, έχοντας κατά νου διαφορετικούς στόχους. Η ανίχνευση και ο μετριασμός της κατανεμημένης άρνησης εξυπηρέτησης (DDoS), όπως και η διάδοση botnet (Feamster, 2010) και worm (Jin and Wang, 2013), είναι μερικές συγκεκριμένες περιπτώσεις χρήσης τέτοιων εφαρμογών. Βασικά, η ιδέα συνίσταται στην περιοδική συλλογή στατιστικών του δικτύου από το επίπεδο προώθησης του δικτύου με τυποποιημένο τρόπο (π.χ. με την χρήση του Openflow) και στην συνέχεια στην εφαρμογή αλγορίθμων ταξινόμησης σε αυτά τα στατιστικά στοιχεία για την ανίχνευση οποιωνδήποτε ανωμαλιών στο δίκτυο. Εάν

ανιχνευθεί κάποια ανωμαλία, η εφαρμογή καθοδηγεί τον ελεγκτή στο πώς να επαναπρογραμματίσει το επίπεδο δεδομένων προκειμένου να μετριάσει την ανωμαλία.

Ένα άλλο είδος εφαρμογής ασφάλειας χρησιμοποιεί τον ελεγκτή SDN εφαρμόζοντας ορισμένους αλγόριθμους άμυνας κινούμενου στόχου (MTD). Οι αλγόριθμοι MTD συνήθως χρησιμοποιούνται για να κάνουν οποιαδήποτε επίθεση σε ένα συγκεκριμένο σύστημα ή δίκτυο δυσκολότερη από το συνηθισμένο, κρύβοντας περιοδικά ή αλλάζοντας τις βασικές ιδιότητες αυτού του συστήματος ή δικτύου. Στα παραδοσιακά δίκτυα, η εφαρμογή αλγορίθμων MTD δεν αποτελεί καθόλου τετριμμένο καθήκον, δεδομένου ότι είναι δύσκολο να δημιουργηθεί μια κεντρική αρχή ικανή να προσδιορίσει - για κάθε τμήμα του συστήματος που πρέπει να προστατευθεί - ποιες βασικές ιδιότητες κρύβονται ή αλλάζουν. Σε ένα δίκτυο SDN, οι εργασίες αυτές καθίστανται πιο απλές χάρη στην κεντρική θέση του ελεγκτή. Μια εφαρμογή μπορεί για παράδειγμα να εκχωρεί περιοδικά εικονικές IP σε κεντρικούς υπολογιστές μέσα στο δίκτυο και στην συνέχεια η ψαρτογράφηση της εικονικής IP / πραγματικής IP εκτελείται από τον ελεγκτή (Jafarian, Al-Shaer and Duan, 2012). Μια άλλη εφαρμογή μπορεί να προσομοιώνει ορισμένες ψεύτικες ανοικτές / κλειστές / φιλτραρισμένες θύρες σε τυχαίους υποδοχείς στο δίκτυο προκειμένου να προσθέσει σημαντικό θόρυβο κατά την φάση της αναγνώρισης (π.χ. σάρωση) που εκτελείται από έναν εισβολέα (Kampanakis, Perros and Beyene, 2014).

Πρόσθετη αξία σχετικά με την ασφάλεια στα δίκτυα με δυνατότητα SDN μπορεί επίσης να αποκτηθεί χρησιμοποιώντας τα FlowVisor (Jain and Paul, 2013) και FlowChecker (Al-Shaer and Al-Haj, 2010) αντίστοιχα. Το πρώτο προσπαθεί χρήσης ενός ενιαίου επιπέδου υλικού προώθησης που μοιράζεται πολλά διαφορετικά λογικά δίκτυα. Σε αυτήν την προσέγγιση, οι ίδιοι πόροι υλικού μπορούν να χρησιμοποιηθούν για σκοπούς παραγωγής και ανάπτυξης καθώς και για τον διαχωρισμό της παρακολούθησης, της διαμόρφωσης και της διαδικτυακής κυκλοφορίας, όπου κάθε σενάριο μπορεί να έχει την δική του λογική τοπολογία που ονομάζεται τεμάχιο (slice). Σε συνδυασμό με αυτήν την προσέγγιση, το FlowChecker (Jain and Paul, 2013) αντιλαμβάνεται την επικύρωση των νέων κανόνων OpenFlow που αναπτύσσονται από τους χρήστες χρησιμοποιώντας το δικό τους τεμάχιο.

Οι εφαρμογές ελεγκτή SDN αναπτύσσονται κυρίως σε μεγάλης κλίμακας σενάρια, γεγονός που απαιτεί εκτεταμένους ελέγχους πιθανών σφαλμάτων προγραμματισμού. Ένα σύστημα που μπορεί να το κάνει αυτό ονομάζεται NICE και περιγράφηκε το 2012 (Canini et al., 2012). Με την εισαγωγή μιας γενικής αρχιτεκτονικής ασφάλειας απαιτείται μια ολοκληρωμένη και παρατεταμένη προσέγγιση της SDN. Από τότε που εισήχθη, οι σχεδιαστές εξετάζουν τους πιθανούς τρόπους για να εξασφαλίσουν αρχιτεκτονική SDN που δεν θέτει σε κίνδυνο την επεκτασιμότητα. Μια αρχιτεκτονική που ονομάζεται Αρχιτεκτονική Ασφάλειας SN-SECA (SDN + NFV) (Bernardo and Chua, 2015).

1.6 Ομαδική παράδοση δεδομένων χρησιμοποιώντας SDN

Κατανεμημένες εφαρμογές που λειτουργούν σε κέντρα δεδομένων αναπαράγουν συνήθως δεδομένα για σκοπούς συγχρονισμού, ανοχής στα σφάλματα, εξισορρόπησης φορτίου και για να φέρουν τα δεδομένα πιο κοντά στους χρήστες (το οποίο μειώνει την λανθάνουσα κατάσταση για τους χρήστες και αυξάνει την αντιληπτή απόδοση). Επίσης, πολλές εφαρμογές, όπως το Hadoop, αναπαράγουν δεδομένα σε ένα κέντρο δεδομένων σε πολλαπλά racks για να αυξηθεί η ανοχή στο σφάλμα και να διευκολυνθεί η ανάκτηση δεδομένων. Όλες αυτές οι λειτουργίες απαιτούν την παράδοση δεδομένων από μία μηχανή ή ένα κέντρο δεδομένων σε πολλαπλές μηχανές ή κέντρα δεδομένων. Η διαδικασία της αξιόπιστης παράδοσης δεδομένων από ένα μηχάνημα σε πολλαπλές μηχανές αναφέρεται ως Αξιόπιστη Ομαδική Παράδοση Δεδομένων (RGDD).

Οι διακόπτες SDN μπορούν να χρησιμοποιηθούν για την περίπτωση RGDD μέσω της εγκατάστασης κανόνων που επιτρέπουν την προώθηση σε πολλαπλές εξερχόμενες θύρες. Για παράδειγμα, το OpenFlow παρέχει υποστήριξη για ομαδικούς πίνακες από την έκδοση 1.1 που καθιστά το παραπάνω εφικτό (Heller, 2009). Χρησιμοποιώντας το SDN, ένας κεντρικός ελεγκτής μπορεί να ρυθμίσει προσεκτικά και έξυπνα τα δέντρα προώθησης για το RGDD. Τέτοια δέντρα μπορούν να κατασκευαστούν με προσοχή στην κατάσταση συμφόρησης / φορτίου του δικτύου για βελτίωση της απόδοσης (Zhu et al., 2016). Για παράδειγμα, το MCTCP είναι ένα σχέδιο για την παράδοση σε πολλούς κόμβους μέσα σε κέντρα δεδομένων

που βασίζονται σε κανονικές και δομημένες τοπολογίες και το DCCast αποτελεί μία παρόμοια προσέγγιση για την παράδοση σε όλα τα κέντρα δεδομένων (Noormohammadpour et al., 2017).

1.7 Ελεγκτές SDN

Υπάρχει ένα σύνολο από ελεγκτές SDN που έχουν αναπτυχθεί με διαφορετικές σκοπούς και στο συγκεκριμένο υποκέφαλαιο θα αναφερθούμε σε χαρακτηριστικές περιπτώσεις περιγράφοντας την δομή τους και τις λειτουργίες τους SDN τεχνολογία, τα δίκτυα στην νέα εποχή.

1.7.1 Openflow Reference Controller

Ο openflow reference controller γνωστός και OVS controller αποτελεί τον πιο απλό controller που διανέμεται μαζί με το Openflow. Σκοπός είναι να προσφέρει απλές εφαρμογές που σχετίζονται με το Openflow πρωτόκολλο όπως την διαχείριση μεγάλο αριθμό openflow μεταγωγών , και την δημιουργία απλών εγγραφών στους Openflow πίνακες. (Christodoulopoulos et al., 2012)

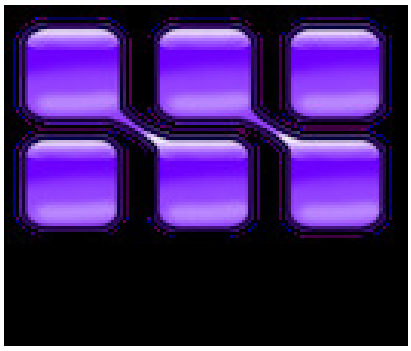
1.7.2 Nox Controller

Ο Nox αποτελεί έναν από τους κύριους Openflow Controller. Βασίζεται στην ιδέα του δικτυακού λειτουργικού συστήματος , όπου αποτελεί την βάση πάνω στην οποία αναπτύσσονται οι διάφορες λειτουργίες του δικτύου. Με αυτόν τον τρόπο , λειτουργίες ελέγχου όπως δρομολόγηση , έλεγχος κυκλοφορίας , spanning tree κλπ προσφέρονται ως εφαρμογές οι οποίες είναι εγκατεστημένες στο δικτυακό λειτουργικό σύστημα. (Tavakoli et al., 2009).

1.7.3 Pox controller

Ο Pox controller αποτελεί την εξέλιξη μίας διεπαφής Openflow που είχε αναπτυχθεί σε γλώσσα python για τον Nox controller και στην συνέχεια αποσύρθηκε, και αποτέλεσε ένα βολικό μονοπάτι για την εισαγωγή στην ανάπτυξη SDN εφαρμογών.

Αποτελεί τον controller που θα χρησιμοποιήσουμε για να αναπτύξουμε και την δική μας εφαρμογή στο πλαίσιο της συγκεκριμένης εργασίας μας (Prete et al., 2014).



Σχήμα 1.2:Pox Controller
Πηγή: Koponen et al., 2010

1.7.4 Onix Controller

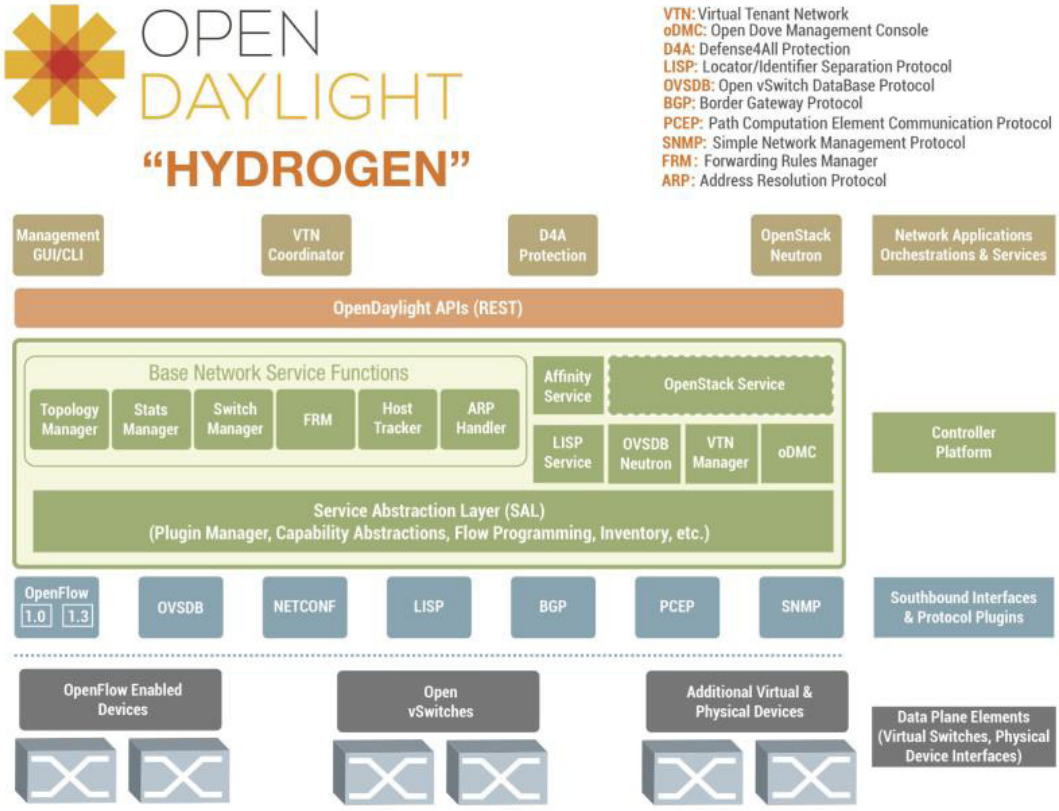
Ο Onix είναι ένας controller που είναι εγκατεστημένος και λειτουργεί σε ένα σύνολο από φυσικούς server όπου σε κάθε server μπορεί να τρέχει ταυτόχρονα ο controller. Ο controller προσφέρει μία προγραμματιστική πρόσβαση στην τοπολογία του δικτύου, όπου το σύνολο το server ενημερώνεται ταυτόχρονα για την κατάστασή του (Koponen et al., 2010).

1.7.5 Beacon και Opendaylight Controller

Ο Beacon είναι ένας controller που βασίζεται στην java κάτι που του επιτρέπει να τρέχει σε διαφορετικές πλατφόρμες , ακόμη και σε android. Ένα από τα μεγάλα του πλεονεκτήματα είναι η δυναμική του φύση. Έχει την δυνατότητα να εκτελεί, σταματάει και να μπορεί να εγκαταστήσει εφαρμογές ακόμη και την στιγμή που ο controller λειτουργεί σε αντίθεση με τους προαναφερόμενους.

1.7.6 Opendaylight project

Το Opendaylight project είναι ένα σύνολο από ερευνητικά έργα που σχετίζονται με τα SDN δίκτυα και αναπτύσσονται υπό την επίβλεψη του Linux foundation με την συμμετοχή των μεγαλύτερων εταιριών δικτύων στον κόσμο. Στόχος του δεν είναι να δημιουργήσει πρότυπα, αλλά εφαρμόσιμες λειτουργίες. Στο κέντρο των έργων βρίσκεται ο SDN controller, που μπορεί να λειτουργεί σε οποιοδήποτε υλικό ή λογισμικό υποστηρίζει java και περιέχει ένα σύνολο από πακέτα λογισμικού που εκτελούν διάφορες εφαρμογές στις προς νότο, και προς βορρά διεπαφές. Στην παρακάτω φωτογραφία εμφανίζεται το σύνολο των έργων που ερευνώνται στο πλαίσιο του Opendaylight.



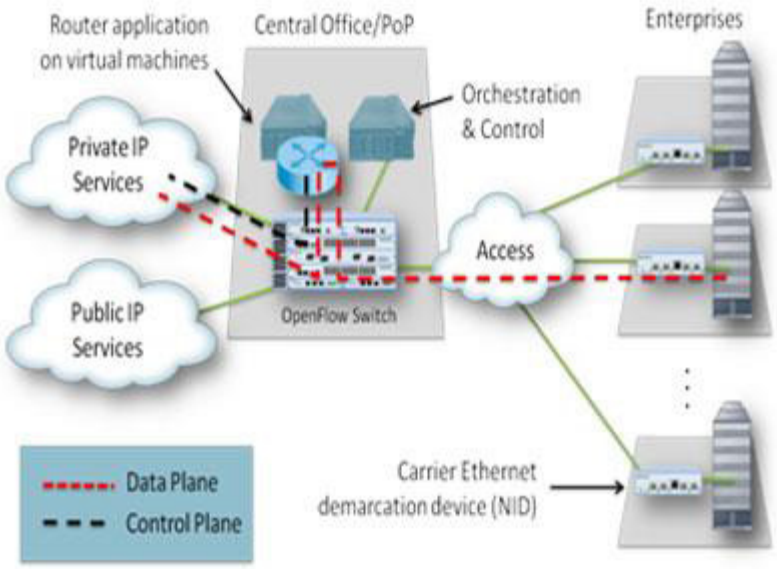
Σχήμα 1.3:OpenDaylight framework
Πηγή: Marschke, Doyle and Moyer, 2015

1.8 Δρομολόγηση ή μεταγωγή;

Συνηθίζουμε να διαχωρίζουμε τον δικτυακό εξοπλισμό , σε δρομολογητές και μεταγωγείς ανάλογα το επίπεδο του OSI βάση του οποίου αποφασίζουν για την προώθηση των πακέτων (Marschke, Doyle and Moyer, 2015). Οι δρομολογητές χρησιμοποιούν διευθύνσεις επιπέδου τρία (IP πρωτόκολλο) ώστε βάση των αντίστοιχων πρωτοκόλλων που χρησιμοποιούνται να δημιουργηθούν και οι αντίστοιχοι πίνακες δρομολόγησης(routing tables), και ενώ οι μεταγωγείς σε επίπεδο δύο(Ethernet πρωτόκολλο).

1.9 SDN και Network Function Virtualization

Η Network functions Virtualization(NFV) τεχνολογία (γνωστή και ως virtual network function(VNF) προσφέρει έναν καινούριο τρόπο, σχεδίασης και ανάπτυξης δικτυακών υπηρεσιών. Η NFV διαχωρίζει δικτυακές λειτουργίες όπως το network address translation(NAT) το domain name service(DNS) , το firewalling κλπ από τον υλικό ώστε να μπορούν να υλοποιηθούν ως λογισμικό. Έχει σχεδιαστεί με σκοπό την ενσωμάτωση δικτυακών λειτουργιών σε πλήρως εικονοποιημένες (virtual) τοπολογίες όπως virtual server , storages ακόμη και άλλα δίκτυα. Αξιοποιεί διαθέσιμες τεχνολογίες εικονοποίησης για την διάθεση εικονοποιημένων δικτυακών εφαρμογών. Βρίσκει εφαρμογή σε όλες τις λειτουργίες είτε του control plane είτε του data plane σε ενσύρματα και ασύρματα δίκτυα.



Σχήμα 1.1:Αρχιτεκτονική NFV
Πηγή: Marschke, Doyle and Moyer, 2015

1.10 Επίλογος

Στο συγκεκριμένο κεφάλαιο παρουσιάσαμε τις βασικές έννοιες και τα δομικά στοιχεία που απαρτίζουν τα SDN δίκτυα. Αναφερθήκαμε στα πλεονεκτήματά τους , σε ορισμένες χαρακτηριστικές περιπτώσεις controller, καθώς και αποσαφηνίσαμε ορισμένες έννοιες με τις οποίες σχετίζονται.

Κεφάλαιο 2: Η τεχνολογία OpenFlow

2.1 Εισαγωγή

Το OpenFlow επιτρέπει στους ελεγκτές δικτύου να καθορίζουν τη διαδρομή των πακέτων δικτύου σε ένα δίκτυο διακοπτών. Οι ελεγκτές διακρίνονται από τους διακόπτες. Αυτός ο διαχωρισμός του ελέγχου από την προώθηση επιτρέπει την πιο εξελιγμένη διαχείριση της κυκλοφορίας από ό, τι είναι εφικτό χρησιμοποιώντας λίστες ελέγχου πρόσβασης (ACL) και πρωτόκολλα δρομολόγησης. Επίσης, το OpenFlow επιτρέπει τη διαχείριση απομακρυσμένων από διαφορετικούς προμηθευτές - συχνά με τις δικές τους ιδιόκτητες διεπαφές και γλώσσες δέσμης ενεργειών - χρησιμοποιώντας ένα απλό, ανοιχτό πρωτόκολλο. Οι εφευρέτες του πρωτοκόλλου θεωρούν το OpenFlow έναν παράγοντα που επιτρέπει τη δημιουργία δικτύων καθορισμένων από λογισμικό (SDN).

Το OpenFlow επιτρέπει την απομακρυσμένη διαχείριση των πινάκων προώθησης πακέτων ενός μεταγωγέα στρώματος 3, προσθέτοντας, τροποποιώντας και καταργώντας κανόνες και ενέργειες αντιστοίχισης πακέτων. Με αυτό τον τρόπο, οι αποφάσεις δρομολόγησης μπορούν να γίνονται περιοδικά ή ad hoc από τον ελεγκτή και να μεταφράζονται σε κανόνες και ενέργειες με ρυθμιζόμενη διάρκεια ζωής, οι οποίες στη συνέχεια αναπτύσσονται στον πίνακα ροής ενός διακόπτη, αφήνοντας την πραγματική προώθηση των αντιστοιχισμένων πακέτων στον διακόπτη σε ταχύτητα καλωδίου τη διάρκεια αυτών των κανόνων. Τα πακέτα που είναι ασύγκριτα από τον διακόπτη μπορούν να προωθηθούν στον ελεγκτή.

Ο ελεγκτής μπορεί στη συνέχεια να αποφασίσει να τροποποιήσει τους υπάρχοντες κανόνες πίνακα ροής σε έναν ή περισσότερους διακόπτες ή να εφαρμόσει νέους κανόνες, ώστε να αποφευχθεί μια δομική ροή κίνησης μεταξύ διακόπτη και ελεγκτή. Μπορεί ακόμη και να αποφασίσει να προωθήσει την ίδια την κυκλοφορία, υπό την προϋπόθεση ότι έχει πει στον διακόπτη να μεταφέρει ολόκληρα πακέτα αντί μόνο στην επικεφαλίδα τους. Το πρωτόκολλο OpenFlow τοποθετείται πάνω από το Πρωτόκολλο Ελέγχου Μεταφοράς (TCP) και προδιαγράφει τη χρήση του TLS (Transport Layer Security). (Araniti et al., 2014, Touch et al., 2013).

2.2 History

Το OpenFing Foundation (ONF), ένας οργανισμός που καθοδηγείται από χρήστες και διαχειρίζεται το πρότυπο OpenFlow (Shin, Nam και Kim, 2012), αφιερωμένο στην προώθηση και υιοθέτηση της δικτύωσης που καθορίζεται από το λογισμικό (Tootoonchian et al., 2012). Το ONF ορίζει το OpenFlow ως την πρώτη τυπική διεπαφή επικοινωνίας που ορίζεται μεταξύ των επιπέδων ελέγχου και προώθησης μιας αρχιτεκτονικής SDN. Το OpenFlow επιτρέπει την άμεση πρόσβαση και το χειρισμό του επιπέδου προώθησης των συσκευών δικτύου, όπως οι διακόπτες και οι δρομολογητές, τόσο φυσικοί όσο και εικονικοί (βασισμένοι σε hypervisor).

Είναι η απουσία ανοιχτής διεπαφής στο επίπεδο προώθησης που οδήγησε στο χαρακτηρισμό των σημερινών συσκευών δικτύωσης ως μονολιθικών, κλειστών και κεντρικών υπολογιστών. Ένα πρωτόκολλο όπως το OpenFlow είναι απαραίτητο για να μεταφερθεί ο έλεγχος του δικτύου από ιδιοκτήτες διακόπτες δικτύου και σε λογισμικό ελέγχου που είναι ανοιχτού κώδικα και τοπικά διαχειριζόμενη (Sezer et al., 2013).

Ορισμένοι διανομείς δικτύων και δρομολογητών έχουν ανακοινώσει την πρόθεσή τους να αθληθούν ή να μεταφέρουν σπορ διακόπτες για το OpenFlow, συμπεριλαμβανομένων των Alcatel-Lucent (Simon, 2015), Big Switch Networks (Costanzo κ.ά., 2013), Brocade Communications (Azodolmolky, Wieder (Yanni), Radisys (Dolan et al., 2012), Arista Networks, Pica8, NoviFlow, Huawei, Cisco, Dell EMC, Extreme Networks, IBM, Juniper Networks, Digisol, Larch Networks, Hewlett-Packard, NEC και MikroTik (Rezaei et αϊ., 2017). Ορισμένες υλοποιήσεις επιπέδου ελέγχου δικτύου χρησιμοποιούν το πρωτόκολλο διαχείρισης των στοιχείων προώθησης δικτύου (Yeganeh, Tootoonchian και Ganjali, 2013). Το OpenFlow χρησιμοποιείται κυρίως μεταξύ του διακόπτη και του ελεγκτή σε ασφαλές κανάλι.

Ένας αρκετά εκτεταμένος κατάλογος των προϊόντων που σχετίζονται με το OpenFlow μπορεί να βρεθεί στον ιστότοπο ONF και στην ιστοσελίδα του SDNCentral.

2.3 Development

Η έκδοση 1.1 του πρωτοκόλλου OpenFlow κυκλοφόρησε στις 28 Φεβρουαρίου 2011 και η νέα ανάπτυξη του προτύπου διοργανώθηκε από το Ίδρυμα Open Networking (ONF) (Franchi, Poggi και Tomaiuolo, 2013). Τον Δεκέμβριο του 2011, το διοικητικό συμβούλιο ONF ενέκρινε την έκδοση 1.2 του OpenFlow και το δημοσίευσε τον Φεβρουάριο του 2012 (Sonkoly et al., 2012). Η τρέχουσα έκδοση του OpenFlow είναι 1,4 (Akyidiz et al., 2014).

Τον Μάιο του 2011, οι Marvell και Larch Networks ανακοίνωσαν τη διαθεσιμότητα μιας λύσης switching με δυνατότητα OpenFlow, βασισμένης στη στοίβα ελέγχου δικτύου Marvell και στην οικογένεια επεξεργαστών πακέτων Prestera (Khan et al., 2013, Hu, 2014).

Το Πανεπιστήμιο της Ιντιάνα, τον Μάιο του 2011, ξεκίνησε ένα Εργαστήριο Διαλειτουργικότητας SDN σε συνεργασία με το Ίδρυμα Ανοικτού Δικτύου (Open Networking Foundation) για να εξετάσει πόσο καλά συνεργάζονται τα προϊόντα που ορίζονται από το λογισμικό και το OpenFlow (Lopez et al., 2015). Τον Ιούνιο του 2012, το Infoblox κυκλοφόρησε το LINC, έναν ανοιχτό κώδικα OpenFlow 1.2 και 1.3 που είναι συμβατός με το λογισμικό (Kirichek et al., 2016). Τον Φεβρουάριο του 2012, η Big Switch Networks κυκλοφόρησε το Project Floodlight, ένα λογισμικό ανοιχτού κώδικα με Open Source OpenFlow Controller (Erickson, 2013) και ανακοίνωσε την έκδοση SDN Suite της OpenFlow το Νοέμβριο του ίδιου έτους, και παρακολουθήστε εφαρμογές παρακολούθησης (van der Meer et al., 2013).

Τον Φεβρουάριο του 2012, η HP δήλωσε ότι αποτελεί αθλητικό πρότυπο σε 16 από τα προϊόντα Ethernet Switch (Blendin, 2013). Τον Απρίλιο του 2012, ο Urs Hölzle της Google περιέγραψε τον τρόπο με τον οποίο το εσωτερικό δίκτυο της εταιρείας είχε ανασχεδιαστεί πλήρως τα δύο προηγούμενα χρόνια, προκειμένου να λειτουργήσει υπό το OpenFlow με σημαντική βελτίωση της απόδοσης (Levy, 2012).

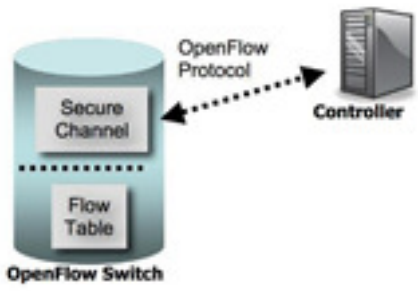
Τον Ιανουάριο του 2013, η NEC αποκάλυψε έναν εικονικό διακόπτη για τον hypervisor Hyper-V του Windows Server 2012 Hyper-V, ο οποίος έχει σχεδιαστεί για να φέρει σε λειτουργία περιβάλλοντα Microsoft (Leitner, 2015).

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 (Heller, 2009).

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 (Antikainen, Aura and Sarela, 2014).



Εικόνα 2.1: OpenFlow enabled switch OF v1.0 specification
Πηγή: Heller (2009)

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 2.1: Ροή-εγγραφής (flow entry)
2-1όπου :

Header Fields	Counters	Actions

Πηγή: (Guo et al., 2015)

- 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 (Banerjee and Kannan, 2014).

Πίνακας 2.2: Πεδία από τα πακέτα που χρησιμοποιούνται για το ταίριασμα ενάντια στις ροές-εγγραφής

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

Πηγή: (Guo et al., 2015)

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 2.3: Λίστα από τους μετρητές που χρησιμοποιούνται για τα στατιστικά των μηνυμάτων

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		

Source: Heller (2009)

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 (Vissicchio and Cittadini, 2016).

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 (Sonba and Abdalkreim, 2014):

- 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. ο Αφαίρεση VLAN κεφαλίδας.
  - Τροποποίηση Ethernet MAC διεύθυνσης πηγής και προορισμού. ο Τροποποίηση IP διεύθυνσης πηγής και προορισμού.
  - Τροποποίηση IP TOS bits
  - Τροποποίηση θυρών πηγής και προορισμού του επιπέδου Μεταφοράς (transport layer).

2.5 Επικοινωνίες

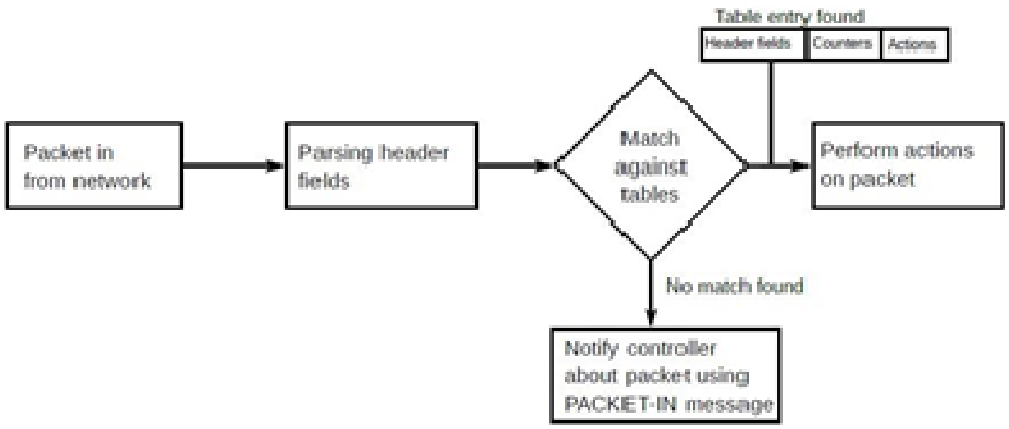
Υπάρχουν τρεις κατηγορίες επικοινωνίας στο πρωτόκολλο OpenFlow (Heller, 2009):

- 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.

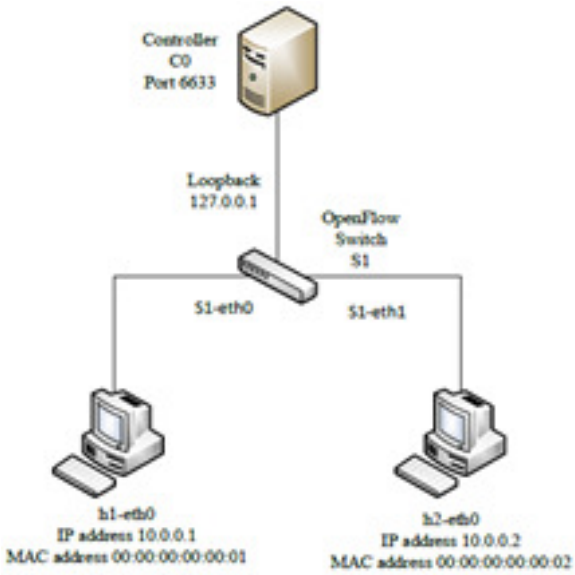
Στη συνέχεια, ο μεταγωγέας ενημερώνει τους μετρητές του εν λόγω πίνακα-ροής. Τέλος, ο μεταγωγέας προωθεί το πακέτο σε μια πόρτα εξόδου. Αν το εισερχόμενο πακέτο δεν ταιριάζει με καμία εγγραφή-ροής στον πίνακα-ροής, ο μεταγωγέας θα προωθήσει το πακέτο στον ελεγκτή για να υπολογίσει την λογική που θα πρέπει να υλοποιηθεί στο πακέτο και στα παρόμοια μελλοντικά πακέτα. Η διαδικασία μηχανισμού της προώθηση των πακέτων απεικονίζεται στο διάγραμμα ροής της Εικόνας 2.2 (Lee et al., 2014).



Εικόνα 2.2: Μηχανισμός Προώθησης Πακέτων του OpenFlow
Πηγή: Braun and Menth (2014)

2.7 Επίδειξη μηνυμάτων που ανταλλάσσονται στο OpenFlow δίκτυο

Για να δείξουμε τα μηνύματα που ανταλλάσσονται σε ένα πραγματικό δίκτυο OpenFlow (Sonba and Abdalkreim, 2014), μπορούμε να χρησιμοποιήσουμε τον δικτυακό εξομοιωτή Mininet (Team, 2012). Με αυτόν θα εξομοιώσουμε δύο τελικούς χρήστες (hosts) συνδεδεμένους με ένα μεταγωγέα που συνδέεται σε έναν ελεγκτή, όπως φαίνεται και στην Εικόνα που ακολουθεί.



Εικόνα 2.3: Δικτυακή τοπολογία από το Mininet
Πηγή: Lopez et al., 2015

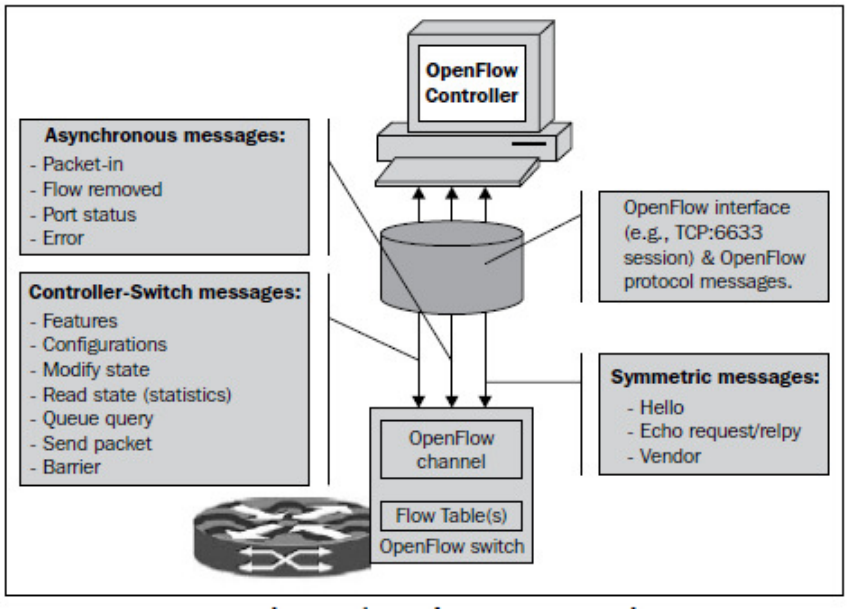
Γι' αυτή την επίδειξη πρέπει να εξηγήσουμε την εγκατάσταση της σύνδεσης μεταξύ Μεταγωγέα - Ελεγκτή, και την επικοινωνία host-to-host μέσω του μεταγωγέα και ελεγκτή OpenFlow (Lopez et al., 2015).

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 (Prete et al., 2014).



Εικόνα 2.4: Μηνύματα επικοινωνίας μεταξύ του OpenFlow μεταγωγέα και του OpenFlow ελεγκτή
Πηγή: Lopez et al., 2015

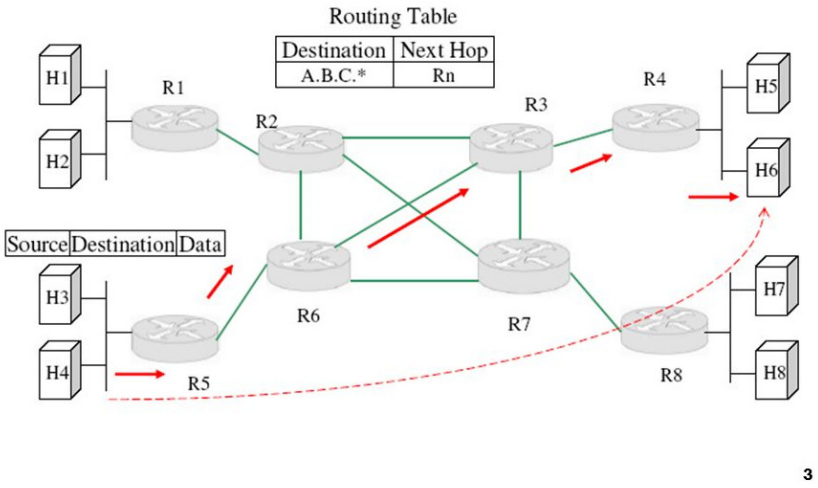
Εικόνα 2.5: Απεικόνιση της επικοινωνίας μεταξύ OpenFlow μεταγωγέα και ελεγκτή
Πηγή: Lopez et al., 2015

2.7.2 Μηνύματα που ανταλλάσσονται μεταξύ δύο Hosts

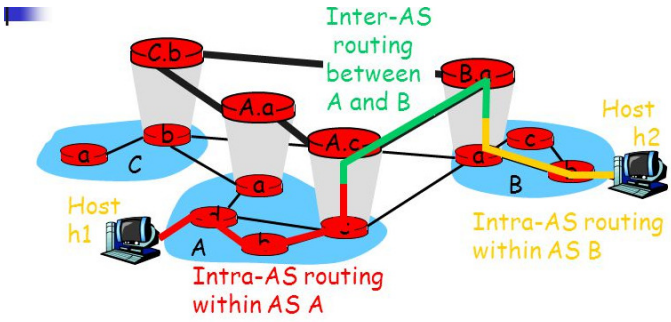
Για να δείξουμε πως γίνεται η σύνδεση host-to-host σε ένα OpenFlow δίκτυο, θα χρησιμοποιήσουμε το εργαλείο του Ping για να στείλουμε ICMP πακέτα από τον host1 (h1) στον host2 (h2) και το αντίστροφο.

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 παρακάτω



Εικόνα 2.6: Διαδικασία Ping μεταξύ h1 και h2
Πηγή: Lopez et al., 2015



Εικόνα 2.7: Πακέτα που ανταλλάζονται μεταξύ του h1 και του h2
Πηγή: Lopez et al., 2015

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 (Braun and Menth, 2014) .

2.8.1 OpenFlow 1.1

Το OpenFlow 1.1 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 (Heller, 2011).

2.8.2 OpenFlow 1.2

Το OpenFlow 1.2 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 κυκλοφορίαςς στα διάφορα πεδία του ICMPv6. Επίσης, μπορεί ένας μεταγωγέας να συνδεθεί σε περισσότερους από έναν ελεγκτές (Open Networking Foundation, 2011).

2.8.3 OpenFlow 1.3

Το OpenFlow 1.3, 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 (Open Networking Foundation, 2013).

2.8.4 OpenFlow 1.4

Το OpenFlow 1.4 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 (Open Networking Foundation, 2014).

2.9 OpenFlow Controllers

Ο ελεγκτής 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 (Sonba and Abdalkreim, 2014).

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 (List of OpenFlow Software Projects, 2015):

- C : Trema (επίσης σε Ruby) και MUL
- C++ : NOX (επίσης σε Python)
- Java : Beacon, Floodlight, Maestro και OpenDaylight
- 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 (Opendailight Cntroller, 2016). Μερικοί τέτοιοι ελεγκτές είναι: Onix (Χριστοφόρου, 2014), από Nicira Networks, IRIS, 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) (Turull, Hidell and Sjodin, 2014).

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 (Wang, 2014). Ένα από αυτά είναι το EstiNet (Wang, Chou and Yang, 2013), το οποίο μπορεί να χρησιμοποιηθεί ως ένας προσομοιωτής ή εξομοιωτής και το άλλο είναι ο εξομοιωτής Mininet (Lantz, Heller and McKeown, 2010; Team, 2012).

Μια καλή ιδιότητα του EstiNet είναι ότι χρησιμοποιεί μία μεθοδολογία ονομαζόμενη «kernel re-entering» 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 (Wang and Kung, 1999; Wang, Chou and Lin, 2007).

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 (Gude et al., 2008), Ryu (2013), και Floodlight (Project Foodlight, 2017).

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 (Nunes et al., 2014).

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» 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 (Wang and Kung, 1999; Wang, Chou and Lin, 2007).

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 (Wang, 2014).

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.

3.3 Mininet

Το Mininet (Lantz, Heller and McKeown, 2010) 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" (Pfaff et al., 2009).

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 (De Oliveira et al., 2014).

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 (Rao, 2014). 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.

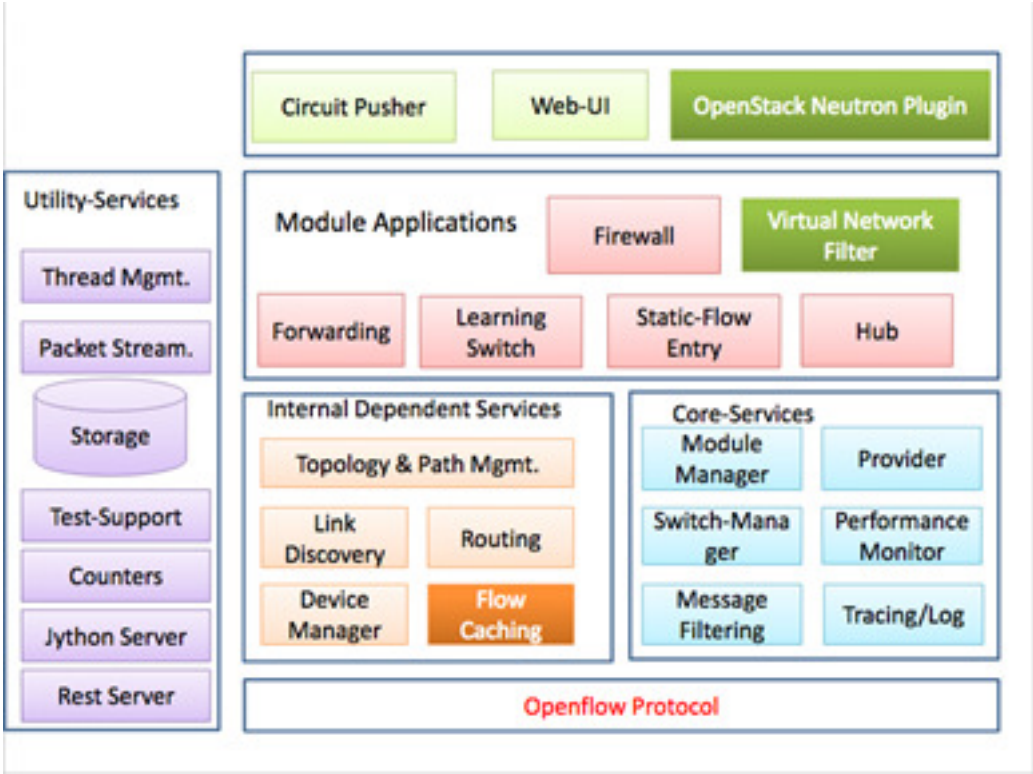
Ο Floodlight Controller έχει άδεια από την Apache και ένα χαρακτηριστικό του είναι ότι έχει συνεισφέρει σημαντικά στην εξέλιξή του η εταιρία Big Switch Networks στην κοινότητα του ανοικτού λογισμικού. Η αρχιτεκτονική του Floodlight βασίζεται σε έναν Big Network Controller (BNC), που είναι μία εμπορική 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 (Wang, 2014).

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 (Park and Choi, 2014).



Εικόνα 4.1: 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 (Ong et al., 2015):

- 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 (Tajima, 2016).

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 πίνακα 4 περιγράφονται μερικές από αυτές τις υπηρεσίες.

Πίνακας 4.1: Υπηρεσίες Floodlight

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.
	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

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 ληφθεί
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
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
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
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 /
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

Πηγή: Tajima, 2016

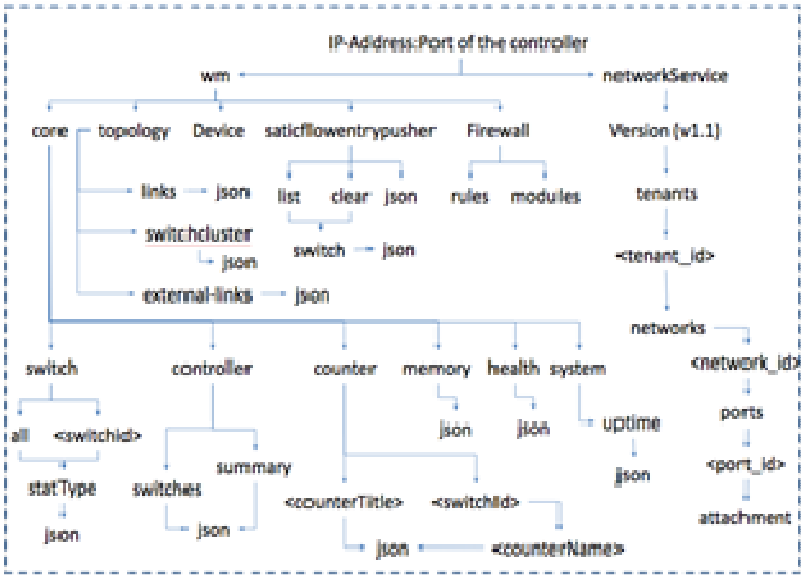
4.1.5 Floodlight Controller RestAPIs

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 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 4.2: Δομή Floodlight REST API
Tajima, 2016

4.2 Ο δικτυακός εξομοιωτής Mininet

Το Mininet 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 (Team, 2012; Χριστοφόρου, 2014).

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 (De Oliveira et al., 2014):

- 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 (Lantz, Heller and McKeown, 2010). 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.

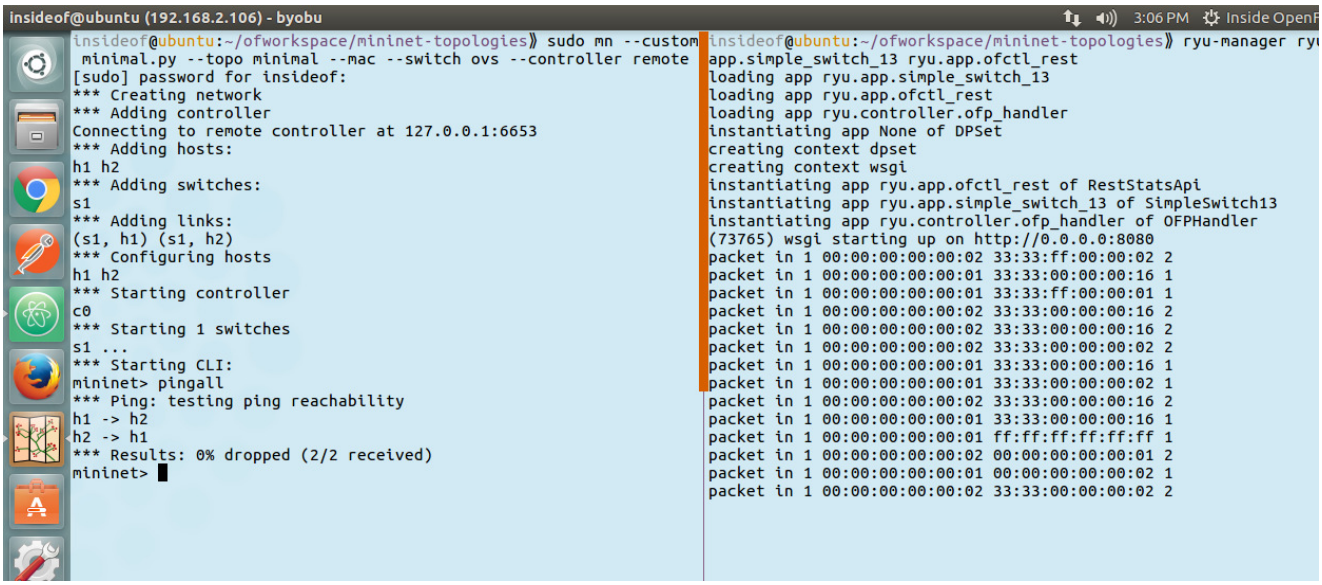
Για να επιβάλει τα όρια του εύρους ζώνης και την ποιότητα της υπηρεσίας (QoS) σε μια σύνδεση, χρησιμοποιείται το πρόγραμμα ελέγχου κυκλοφορίας (TC) του

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 (Wang, 2014).

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



Εικόνα 4.3: Κώδικας τοπολογίας Mininet
Πηγή: Tajima, 2016

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, έτσι ώστε να μπορούν να υποστηριχτούν επιπλέον εντολές από τις συνηθισμένες εντολές τύπου pingAll () και iperf (). Η διαδικασία αυτή υποστηρίζεται από το λογισμικό Mininet (Χριστοφόρου, 2014).

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 (Team, 2012; Χριστοφόρου, 2014).

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 (Team, 2012; Χριστοφόρου, 2014).

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 (Wang, 2014).

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.

Κκατά 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 (Casellas et al., 2014). 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 (Langone, Key and Alder, 2015).

4.3.1 Χρησιμότητα εικονικοποίησης (virtualization)

Οι τεχνικές και τα χαρακτηριστικά που παρέχει το VirtualBox είναι χρήσιμα για διάφορα σενάρια, όπως (Kuhn, Kim and Lopuz, 2015):

- 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 (Kuhn, Kim and Lopuz, 2015):

- 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 απαιτεί 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 (Pfaff et al., 2015).

- 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.

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 (Christodoulopoulos et al., 2012)

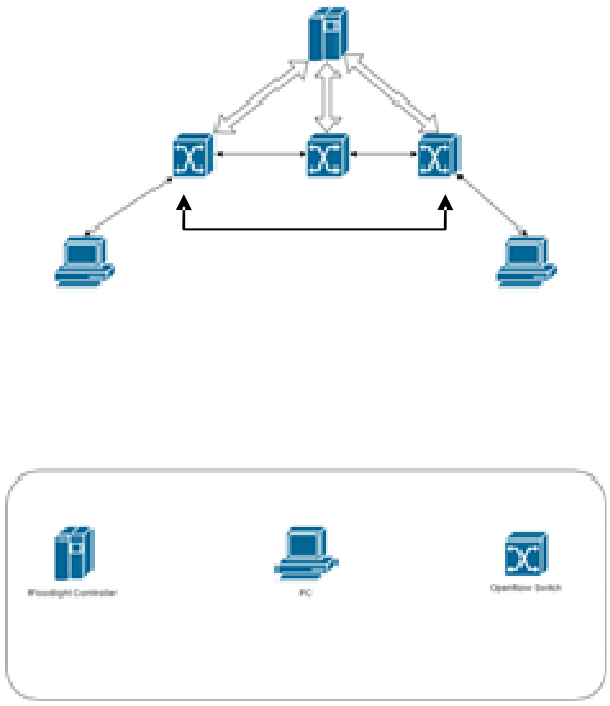
Επιπρόσθετα, η δυνατότητα της διαχείρισης του forwarding table διαμέσου ενός εξωτερικού interface επιτρέπει στην χαμηλού επιπέδου flow state να κάνει migrate μαζί με το VM. Αυτό θα ήταν χρήσιμο για την μεταφορά των υπαρχόντων flow counters και ACLs. Επίσης επιτρέπει το migration κανόνων που αφορούν το tunneling, δυνατότητα χρήσιμη στο migration μεταξύ διαφορετικών υποδικτύων IP (Mian et al., 2014).

Κεφάλαιο 5: SDN σε Περιβάλλον Επιχείρησης

5.1 Προσομοίωση Νησίδας SDN (πειραματικό στάδιο)

Σύμφωνα με τα προαναφερθέντα έγινε παρουσίαση των εργαλείων που είναι δυνατόν να έχουν χρήση ως προς την υλοποίηση ενός SDN.

Οπότε θα καταγραφεί η μελέτη ενός εργαστηρίου OpenFlow που θα βασίζεται στο Mininet και ως ελεγκτής SDN θα είναι ο Floodlight. Στην Εικόνα φάινεται η τοπολογία του δικτύου που θα προσομοιωθεί(Jeong et al., 2014).



Εικόνα 5.1: Τοπολογία Δικτύου Προσομοίωσης
Πηγή: Jeong et al., 2014

5.2 Οδηγίες Εγκατάστασης

Με την εκκίνηση του πειραματικού σταδίου για την προσομοίωση μίας νησίδας SDN θα γίνει λόγος για κάποιους τ΄ροπους προσομοίωσης. Προκύπτουν οι υλοποιήσεις σε εικονικές μηχανές (Virtual Machines - VM), που κρυπτογραφούνται από το SSH. Προτείνονται οι υλοποιήσεις σε εικονικές μηχανές μέσα από ιστοσλίδες όπως η http://sdnhub.org/tutorials/και http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial

Με το υπολογιστικό σύστημα Linux Ubuntu έκδοση 14.04.3-desktop και τύπου 64-bit (amd64) μπαίνουν τα νέα προγράμματα μέσα από το Terminal των Ubuntu με δικαιώματα (διαχειριστή) δίνοντας στη (κονσόλα) την εντολή sudo –i .

5.2.1 Mininet

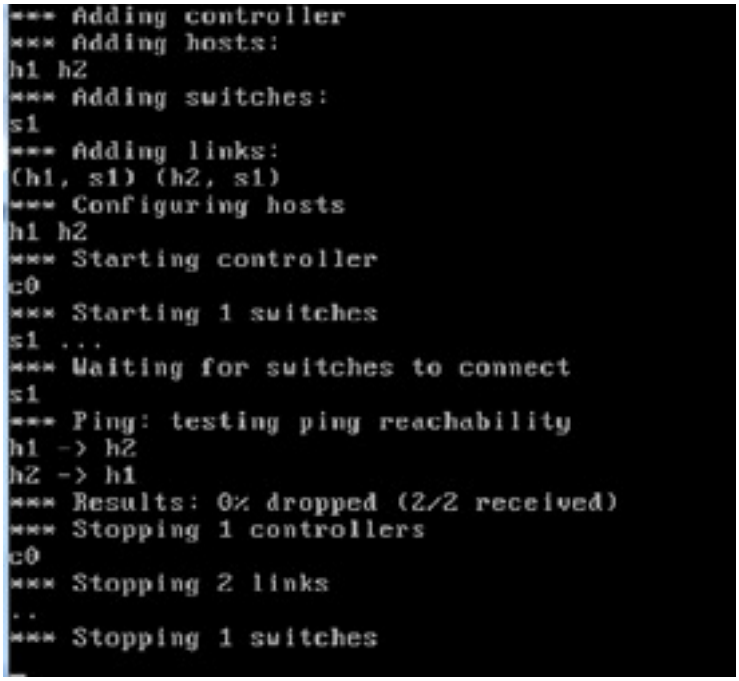
Με το άνοιγμα το terminal του Linux (κονσόλα) επιλέγουνται οι παρακάτω εντολές για εγκατάσταση του Mininet:

```
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
```

Όταν ολοκληρωθεί η εγκατάσταση δοκιμάζουμε την καλή λειτουργία του mininet με την ακόλουθη εντολή:

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

και θα προκύψουν στην κονσόλα τα εξής:



Εικόνα 5.2: Αποτέλεσμα κονσόλας της εντολής mn --test pingall
Πηγή: Jeong et al., 2014

5.2.2 Floodlight Controller

Με το άνοιγμα του terminal του Linux χρησιμοποιούμε τις παρακάτω εντολές για να μπει τον Floodlight controller:

git clone git://github.com/floodlight/floodlight.git

cd floodlight

ant

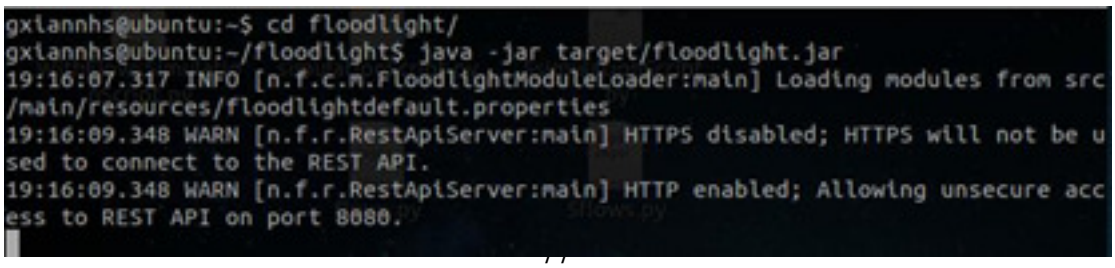
mkdir /var/lib/floodlight

chmod 777 /var/lib/floodlight

Ξεκινάει η λειτουργία του Floodlight controller με την εντολή:

java -jar target/floodlight.jar

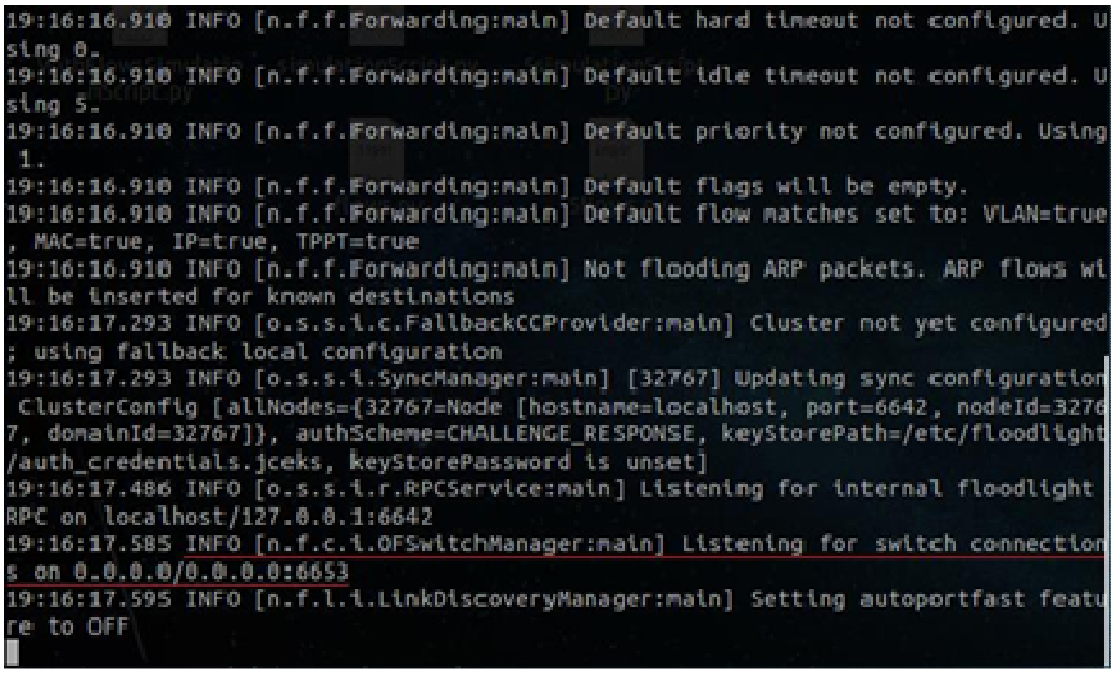
και θα αρχίσουν να εκτυπώνονται τα πορίσματα της αποσφαλμάτωσης στην κονσόλα



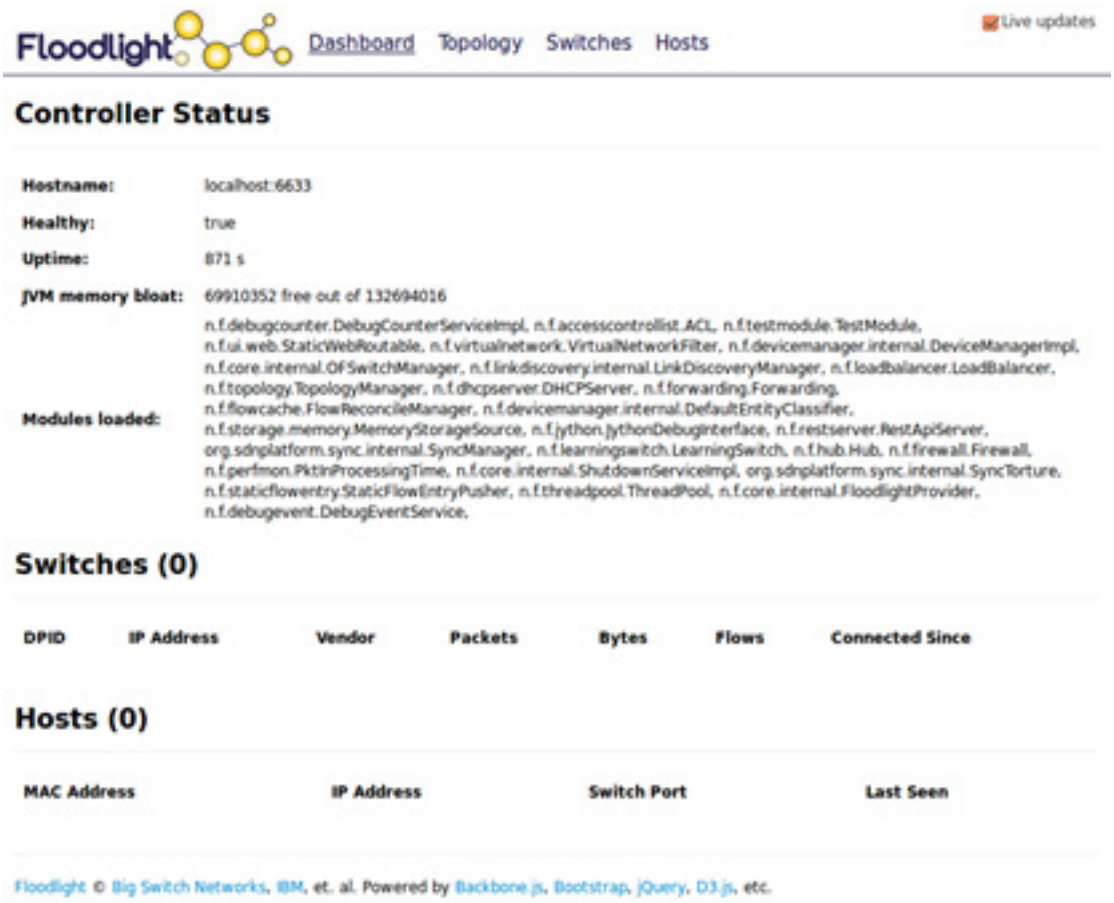
Εικόνα 5.3: Αποτέλεσμα κονσόλας της εντολής java -jar target/floodlight.jar
Πηγή: Jeong et al., 2014

5.3 Προσομοίωση

Με την εγκατάσταση των εργαλείων θα συνεχίσουμε με τη διαδικασία της προσομοίωσης. Αρχικά, θα ξεκινήσουμε τον ελεγκτή SDN όπως ξέρουμε από τα βήματα της εγκατάστασης (Εικόνα 5.2) και ενώ 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 5.3), ο SDN controller είναι προσβάσιμος μέσω φυλλομετρητή (browser) στη διεύθυνση: http://127.0.0.1:8080/ui/index.html όπως φαίνεται στην Εικόνα 5.4.



Εικόνα 5.4: Πληροφορία για σύνδεση στον ελεγκτή SDN
Πηγή: Jeong et al., 2014



Εικόνα 5.5: Floodlight Controller
Πηγή: Jeong et al., 2014

Με τη χρήση του Python API που προσφέρεται από το mininet και δημιουργούμε το κάτωθι πρόγραμμα σε γλώσσα python προκειμένου να μην πρέπει 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()
```

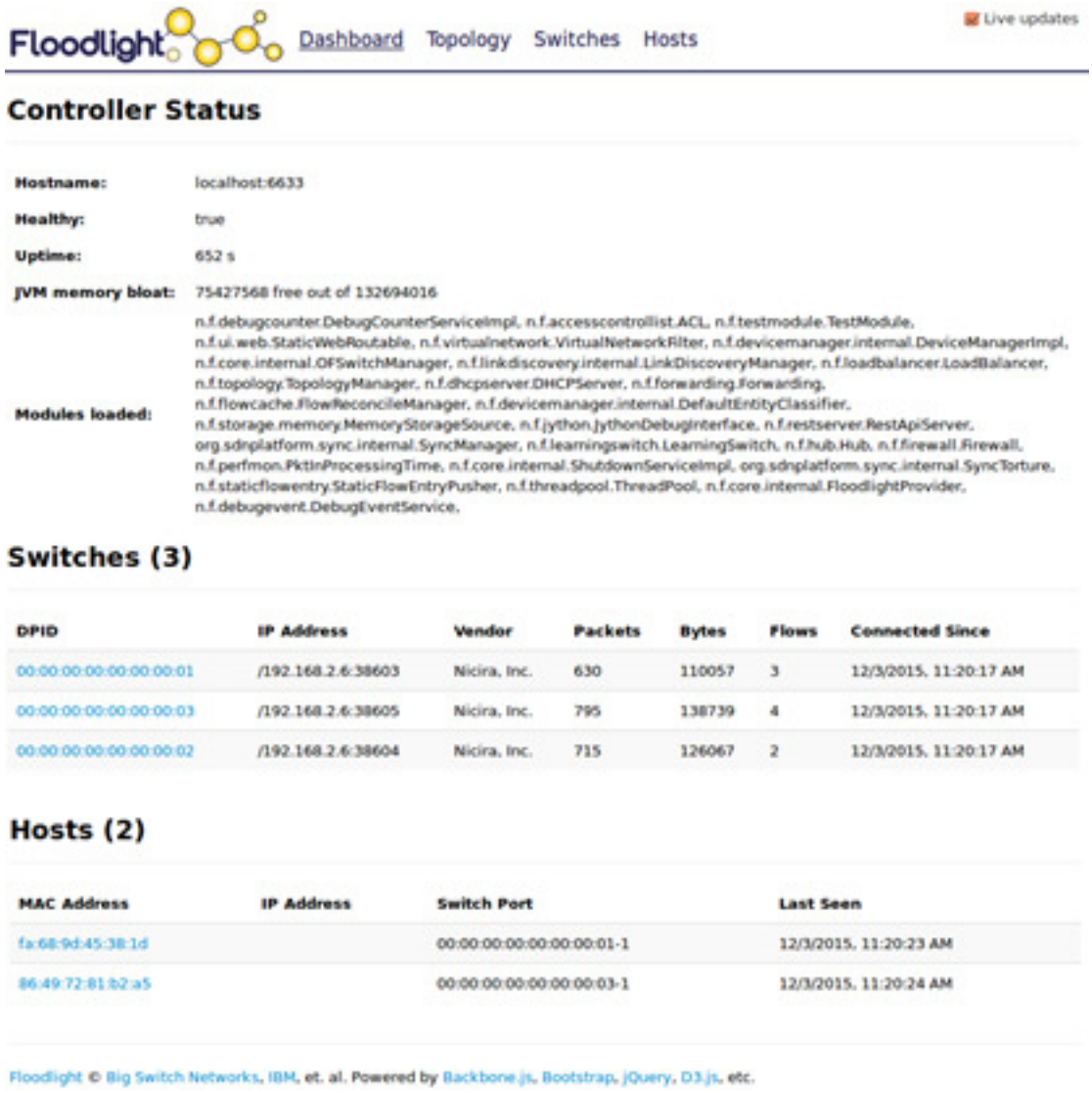
Το προαναφερθέν πρόγραμμα κάνει μία τοπολογία στο Mininet αποτελούμενη από 2 τελικούς χρήστες (hosts) και 3 μεταγωγείς (switches). Είναι δυνατόν να το τρέξουμε από την κονσόλα με τον εντολή:

sudo python simulationScript.py

και θα προκύψουν (Εικόνα 5.6) τα εξής:

Εικόνα 5.6: Πορίσματα κονσόλας της εντολής sudo python simulationScript.py
Πηγή: Jeong et al., 2014

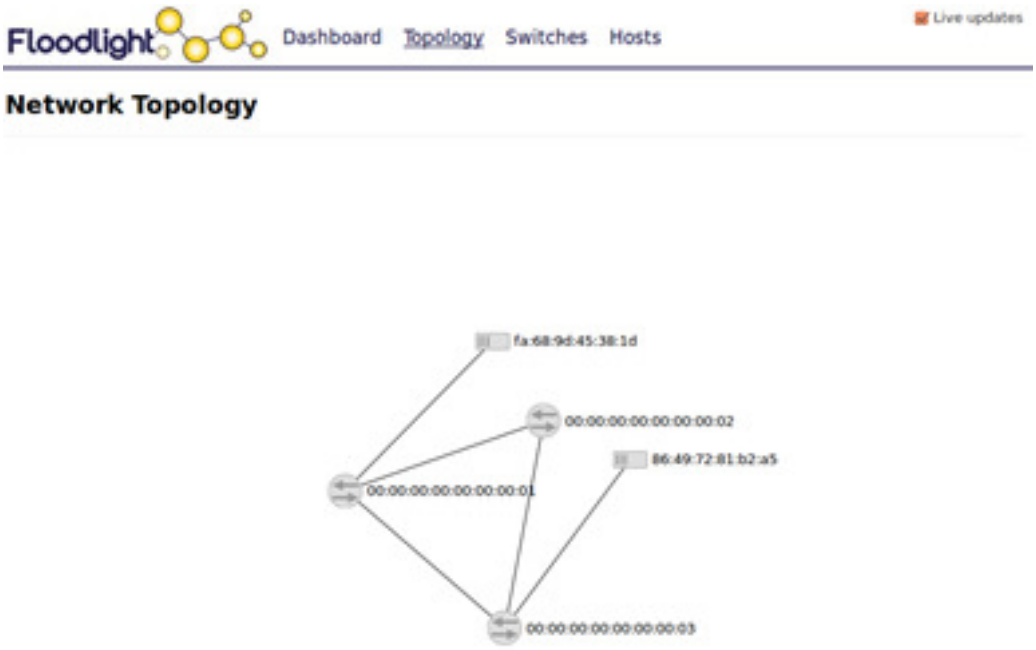
Μετά την εκτέλεση του python script μπορούμε να δούμε στον φυλλομετρητή / browser τις πληροφορίες που προκύπτουν από το παραπάνω πρόγραμμα δηλαδή τους 2 τελικούς χρήστες (hosts) και οι 3 μεταγωγείς (switches) όπως παρουσιάζεται και στην Εικόνα 5.7:



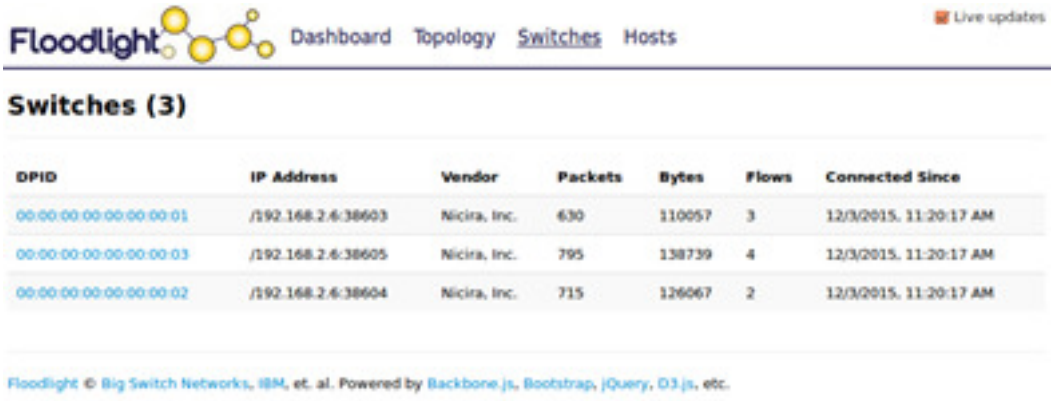
Εικόνα 5.7: Floodlight controller Dashboard
Πηγή: Jeong et al., 2014

Κατόπιν περιήγησης στον φυλλομετρητή δίνουμε στην καρτέλα “Topology” την τοπολογία του δικτύου, που επιβεβαιώνει την ύπαρξη των επιθυμητών διαδρομών στην καρτέλα “Switches” τους μεταγωγείς του δικτύου, που επιβεβαιώνει την παρουσία των επιθυμητών μεταγωγέων (Εικόνα 24), στην καρτέλα “Hosts” τους hosts του δικτύου, που επιβεβαιώνει την ύπαρξη των επιθυμητών

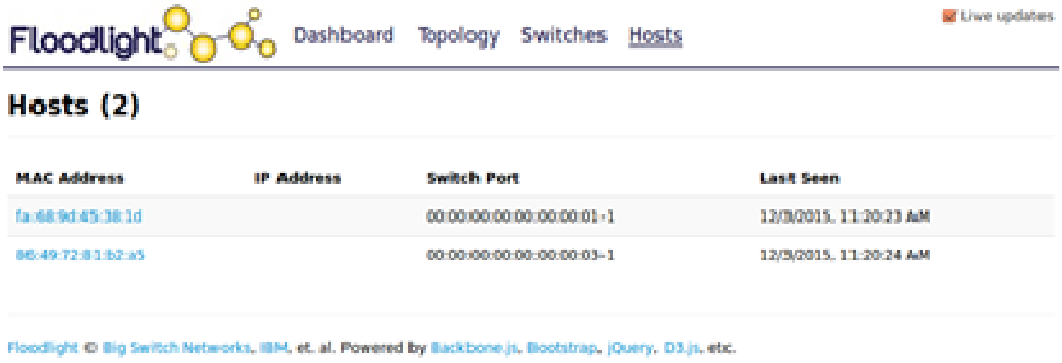
hosts (Εικόνα 5.10) και τις πληροφορίες τους εάν επιλέξουμε τον κάθε host με κλικ στην διεύθυνση MAC υλικού του (Εικόνα 5.11).



Εικόνα 5.8: Τοπολογία Δικτύου
Πηγή: Jeong et al., 2014



Εικόνα 5.9: Μεταγωγείς – Switches
Πηγή: Jeong et al., 2014



Εικόνα 5.10: Οικοδεσπότες – Hosts

Εικόνα 5.11: 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)

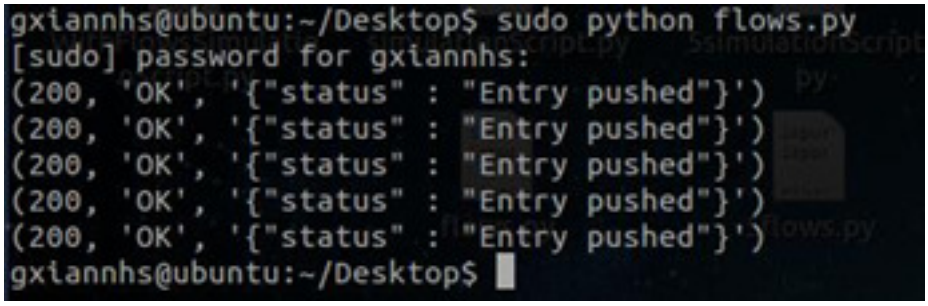
pusher.set(flow4)

pusher.set(flow5)

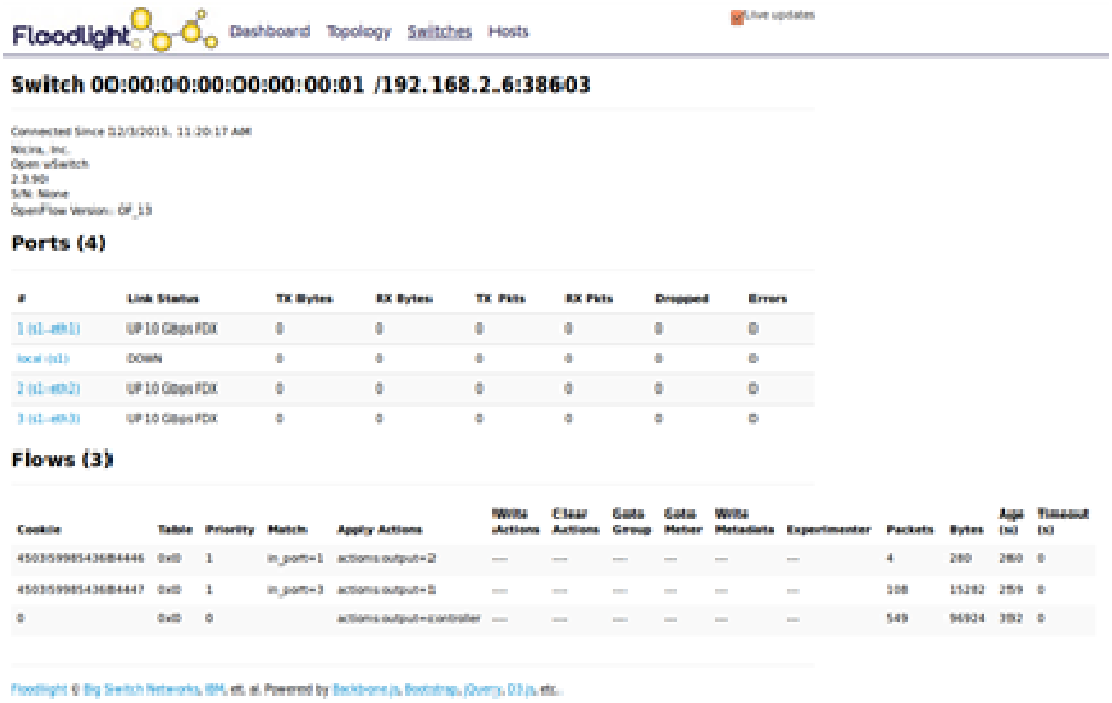
Το πιο πάνω πρόγραμμα με την εντολή pusher=StaticFlowPusher ('192.168.2.6'), όπου το 192.168.2.6 αποτελεί τη ip διεύθυνση του υπολογιστή που τρέχει τον ελεγκτή θα στείλει τις ροές στα 3 switches που αποτελούν την τοπολογία και έτσι θα επικοινωνούν οι 2 hosts. Είναι δυνατόν να το τρέξουμε από την κονσόλα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

sudo python flows.py

και θα προκύψουν στην κονσόλα τα εξής:



Στη συνέχεια για να επιβεβαιωθεί η εισαγωγή των ροών στον ελεγκτή πρέπει να γίνει μετάβαση 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 Από αυτά τα στοιχεία προκύπτει το όνομα του switch, το πότε συνδέθηκε στο δίκτυο, το πρωτόκολλο OpenFlow, η κατάσταση των θυρών και οι ροές που υποστηρίζει.



Εικόνα 5.12: Πληροφορίες Switch
Πηγή: Jeong et al., 2014

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 στην κονσόλα τα εξής:

Έτσι τελειώνει το πειραματικό στάδιο μέσα από αναλυτική αναφορά ενός εργαστηρίου OpenFlow που βασίζεται στο Mininet και πως αυτό αλληλοσχετίζεται με έναν απομακρυσμένο ελεγκτή SDN (Floodlight).

Κεφάλαιο 6: Προσομοίωση πραγματικού δικτύου Επιχείρησης

Ως πρακτικό μέρος θα υλοποιήσουμε σε περιβάλλον εικονικοποίησης την περίπτωση μίας επιχείρησης με τα τμήματα της Διεύθυνσης, της Παραγωγής και του Λογιστηρίου. Αυτά γενικά βρίσκονται στο ίδιο δίκτυο και θέλουμε να έχουμε την επικοινωνία του τμήματος της Διεύθυνσης με το τμήμα του Λογιστηρίου και το τμήμα της Παραγωγής. Ο κύριος σκοπός των πειραμάτων που ακολουθούν είναι η προσομίωση ενός πραγματικού δικτύου μίας επιχειρήσης στη πραγματικότητα με τη χρήση του SDN. Μέσα από το πείραμα θα δούμε ότι δεν αρκεί μόνο να συνδέσουμε τις συσκευές μας σε έναν μεταγωγέα για να έχουμε επικοινωνία αλλά θα χρειαστεί να δημιουργήθουν ροές δεδομένων οι οποίες είναι υπεύθυνες να υλοποιήσουν την επικοινωνία μεταξύ των συσκευών. Αφού έχουμε την πλήρη επικοινωνία των συσκευών που αποτελούν και τα διαφορετικά τμήματα μίας επιχείρησης έχουμε ολοκληρώσει ουσιαστικά το αρχικό μας πείραμα. Κατ' επέκταση σε μελλοντικά σενάρια χρήσης του δικτύο και με τη χρήση του κεντρικού ελεγκτή που έχουμε στο δίκτυο θα μπορέσουμε με ευκολία να υλοποιήσουμε σενάρια Ποιότητας της Υπηρεσίας (QoS), Ασφάλειας, Vlans, Firewalls, κτλ.

[1]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 OpenvSwitch, που έχουμε αναλύσει στο κεφάλαιο «Επιλογή των εργαλείων». Επίσης, θα χρησιμοποιήσουμε το floodlight controller σαν ελεγκτή του λογικού μεταγωγέα που έχουμε αναφέρει στο προηγούμενο κεφάλαιο.

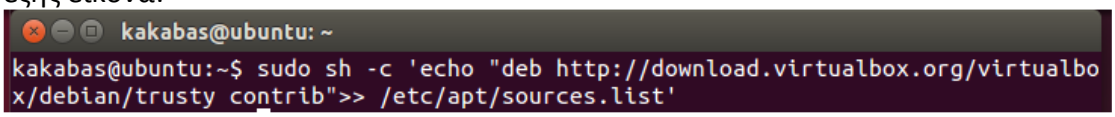
6.2 Οδηγίες Εγκατάστασης

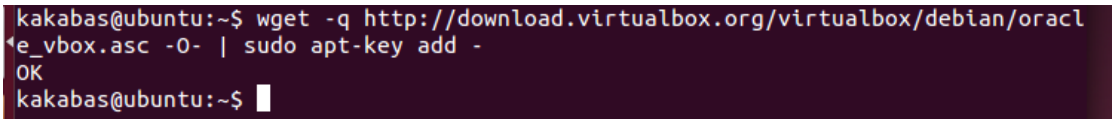
6.2.1 Virtualbox

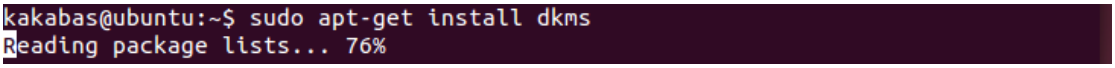
Για να εγκαταστήσουμε το Virtualbox στα Ubuntu θα πρέπει να ανοίξουμε τη κονσόλα των linux και να εκτελέσουμε τις ακόλουθες εντολές:

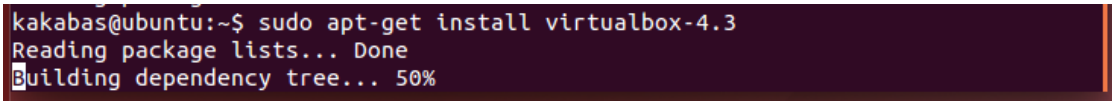
- sudo sh -c 'echo "deb http://download.virtualbox.org/virtualbox/debian trusty contrib" >> /etc/apt/sources.list'
- wget http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc -O- | sudo apt-key add –
- sudo apt-get install dkms
- sudo apt-get install virtualbox-4.3

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

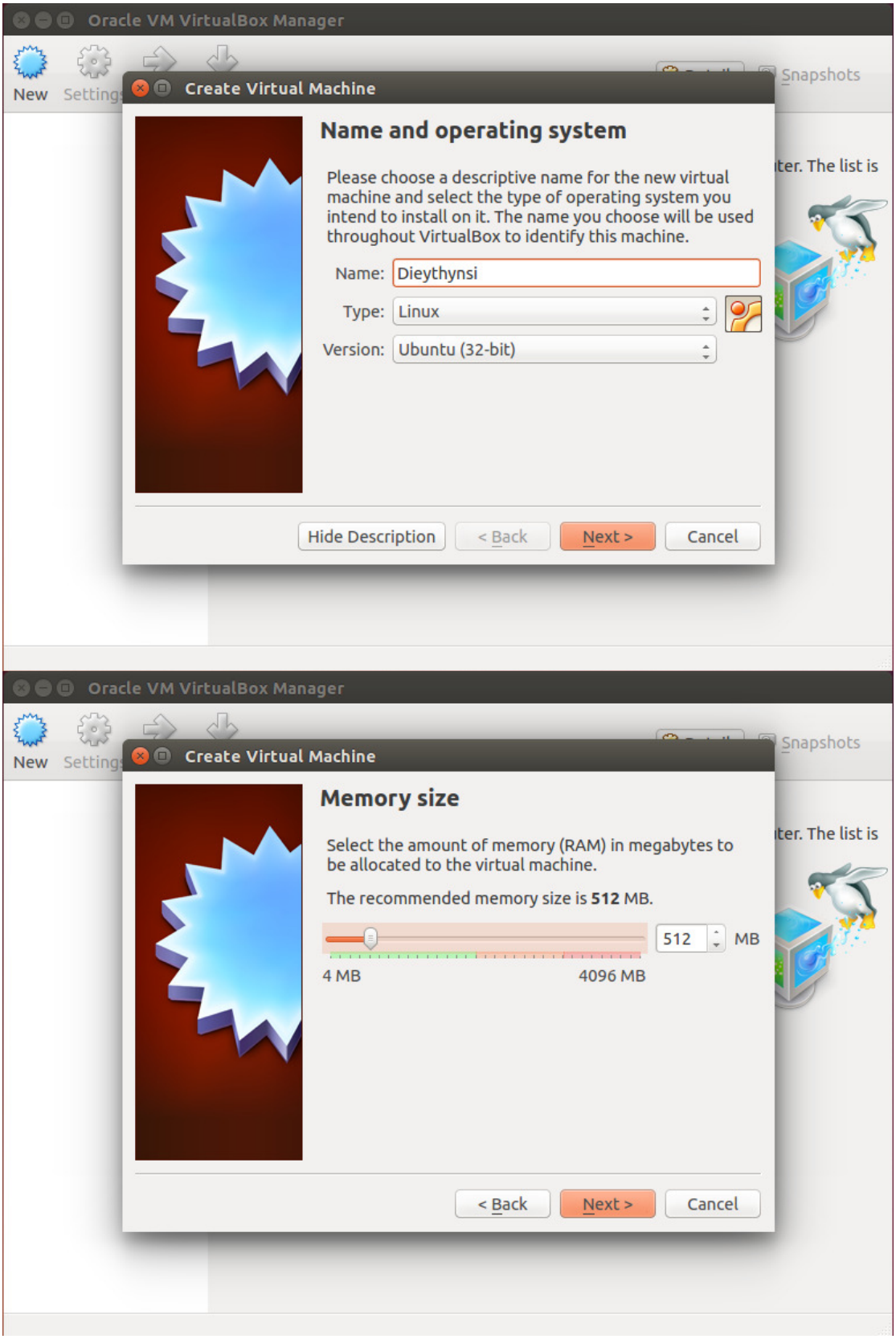


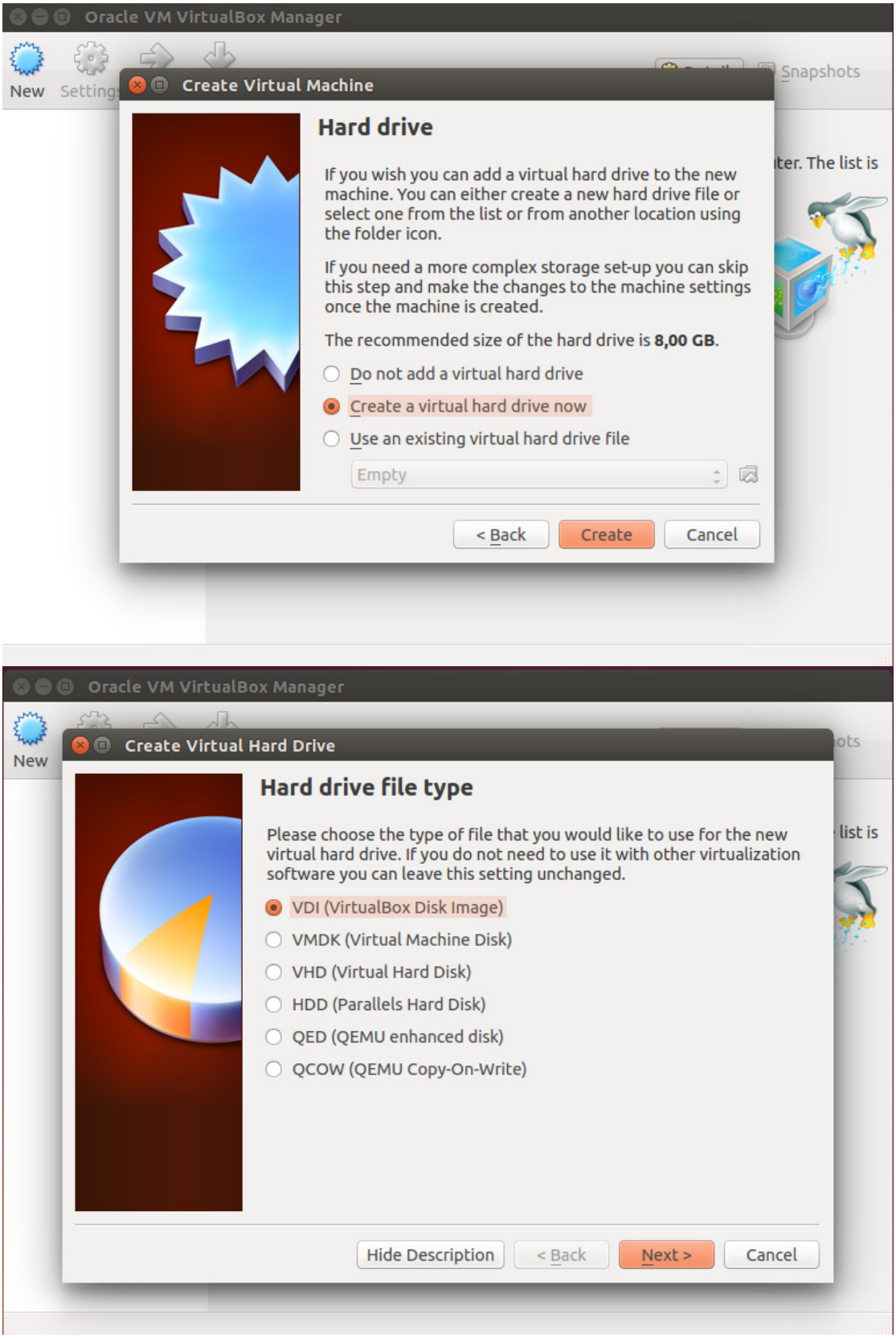


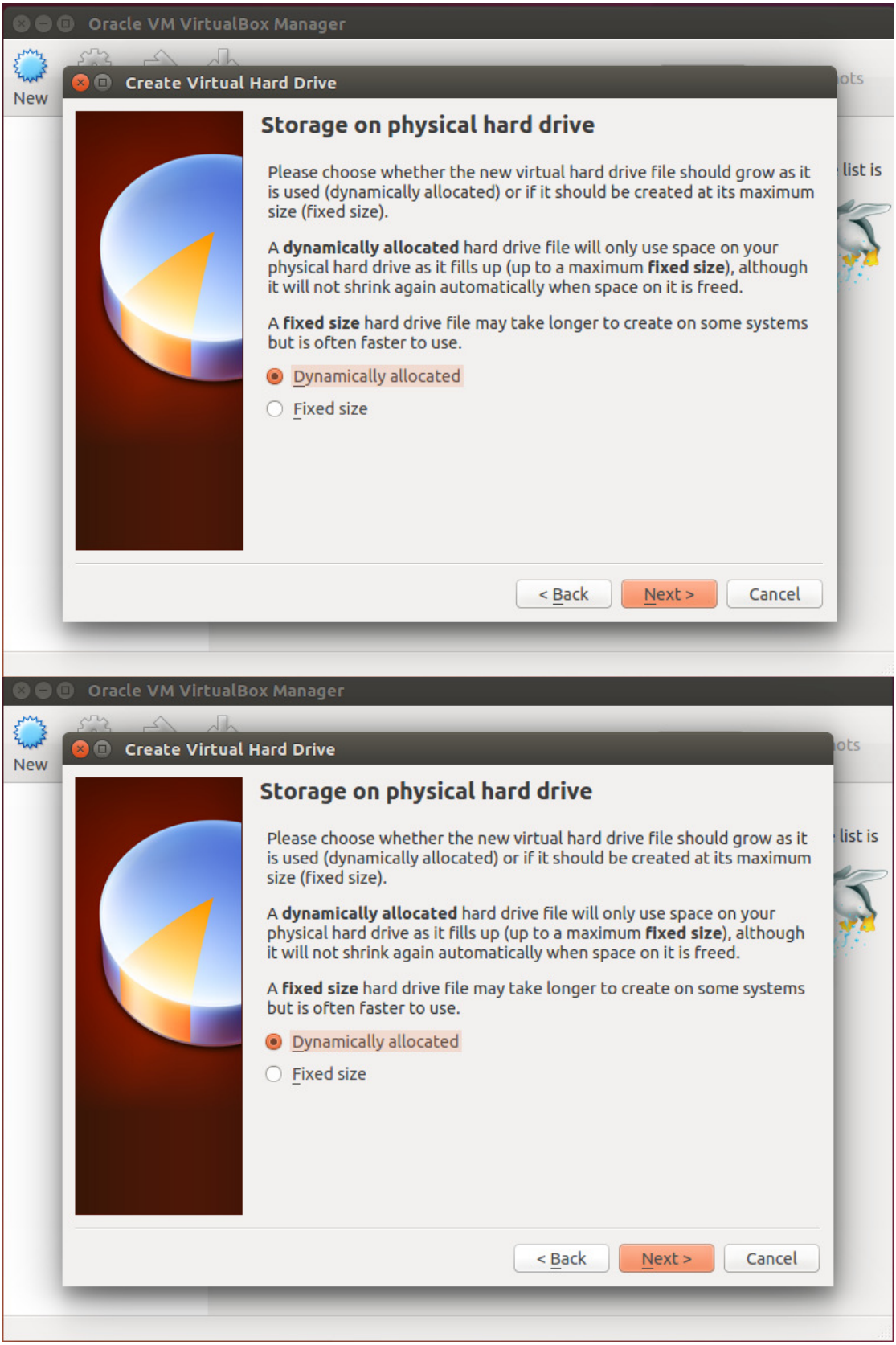


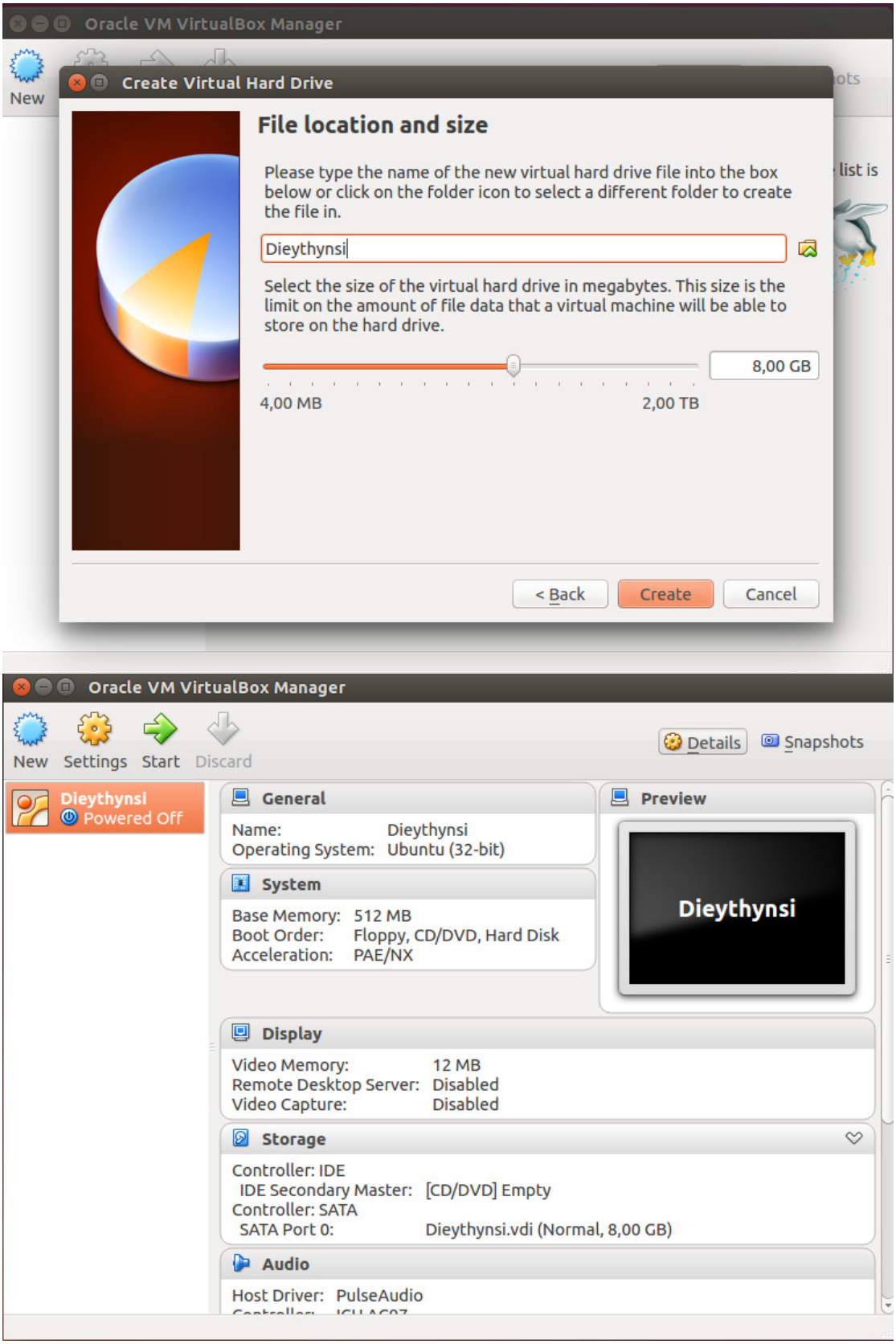


Για τη δημιουργία μίας καινούριας εικονικής μηχανής πατάμε το New πάνω αριστερά και ακολουθούμε τα απαραίτητα βήματα για την ολοκλήρωση της διαδικασίας που περιγράφονται από τις εικόνες.


Oracle VM VirtualBox Manager
New
Settings
Snapshots
Create Virtual Machine
Name and operating system
Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.
Name: Dieythynsi
Type: Linux
Version: Ubuntu (32-bit)
Hide Description
< Back
Next >
Cancel
Oracle VM VirtualBox Manager
New
Settings
Snapshots
Create Virtual Machine
Memory size
Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.
The recommended memory size is 512 MB.
512
MB
4 MB
4096 MB
< Back
Next >
Cancel

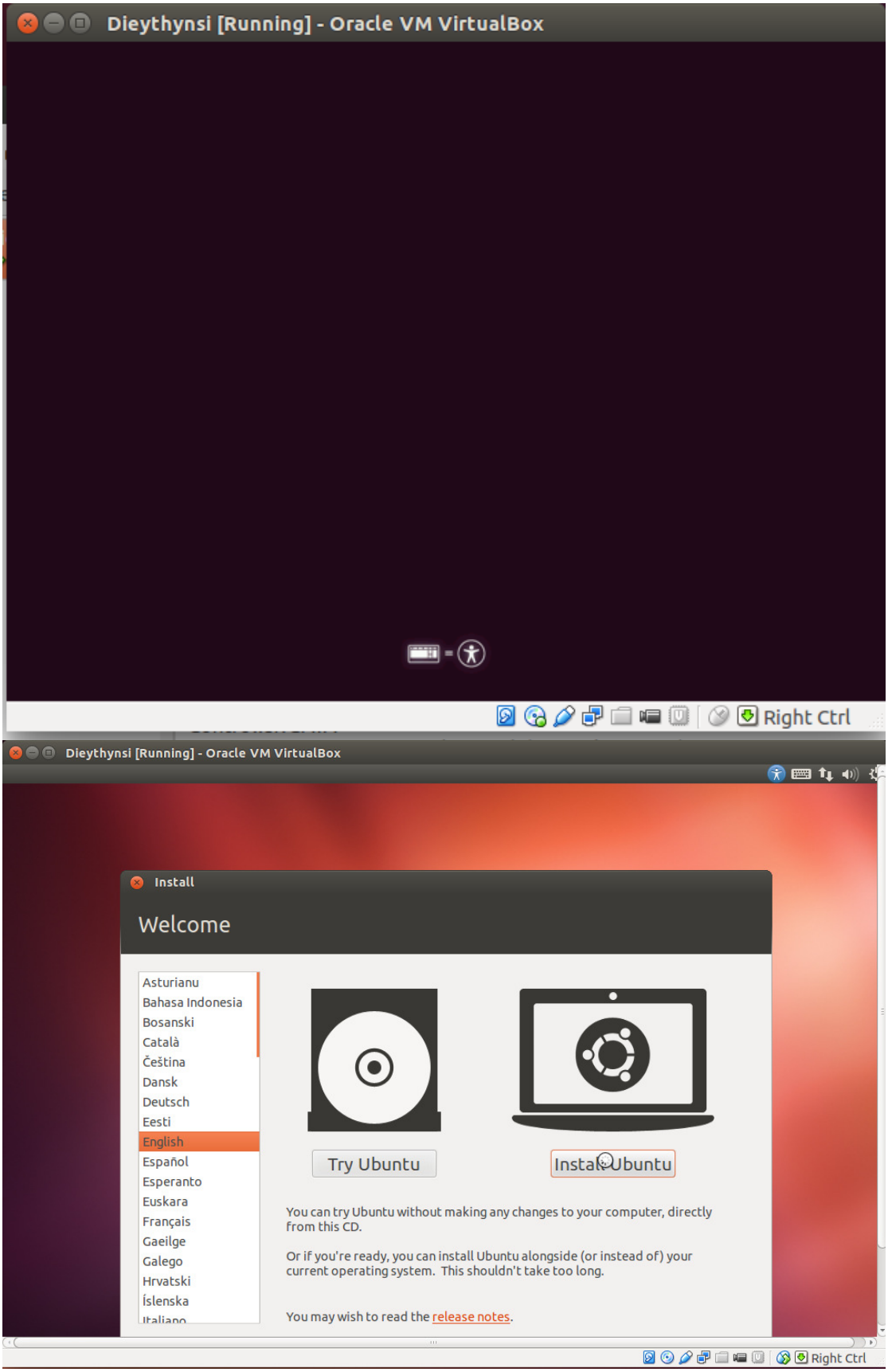

Oracle VM VirtualBox Manager
New
Snapshots
Create Virtual Machine
Hard drive
If you wish you can add a virtual hard drive to the new machine. You can either create a new hard drive file or select one from the list or from another location using the folder icon.
If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.
The recommended size of the hard drive is 8,00 GB.
Do not add a virtual hard drive
Create a virtual hard drive now
Use an existing virtual hard drive file
Empty
< Back
Create
Cancel
Oracle VM VirtualBox Manager
New
Create Virtual Hard Drive
Hard drive file type
Please choose the type of file that you would like to use for the new virtual hard drive. If you do not need to use it with other virtualization software you can leave this setting unchanged.
VDI (VirtualBox Disk Image)
VMDK (Virtual Machine Disk)
VHD (Virtual Hard Disk)
HDD (Parallels Hard Disk)
QED (QEMU enhanced disk)
QCOW (QEMU Copy-On-Write)
Hide Description
< Back
Next >
Cancel

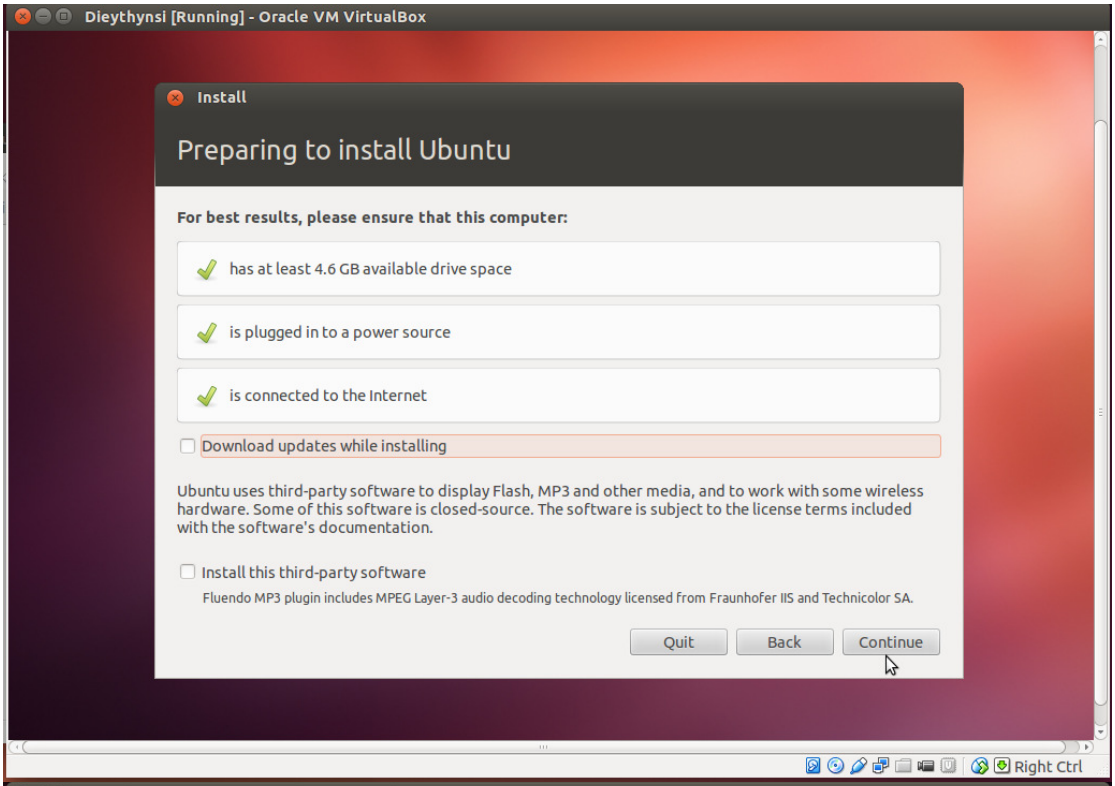

Oracle VM VirtualBox Manager
New
Create Virtual Hard Drive
Storage on physical hard drive
Please choose whether the new virtual hard drive file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).
A dynamically allocated hard drive file will only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

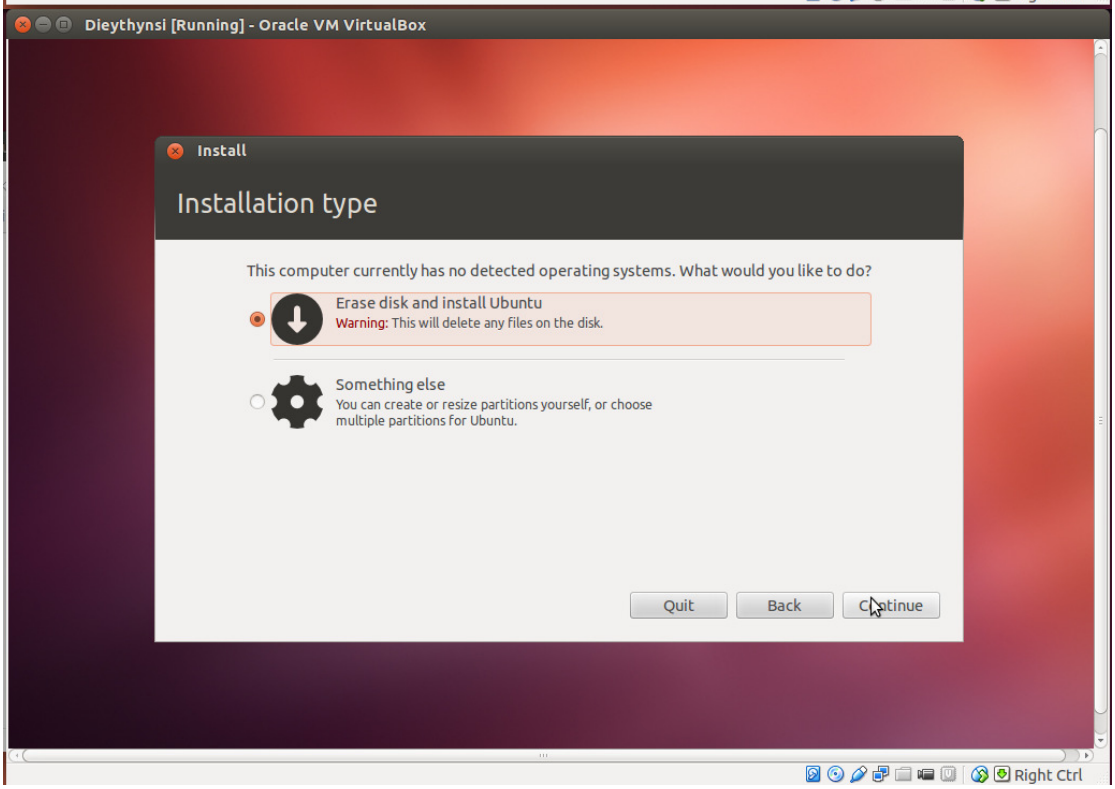


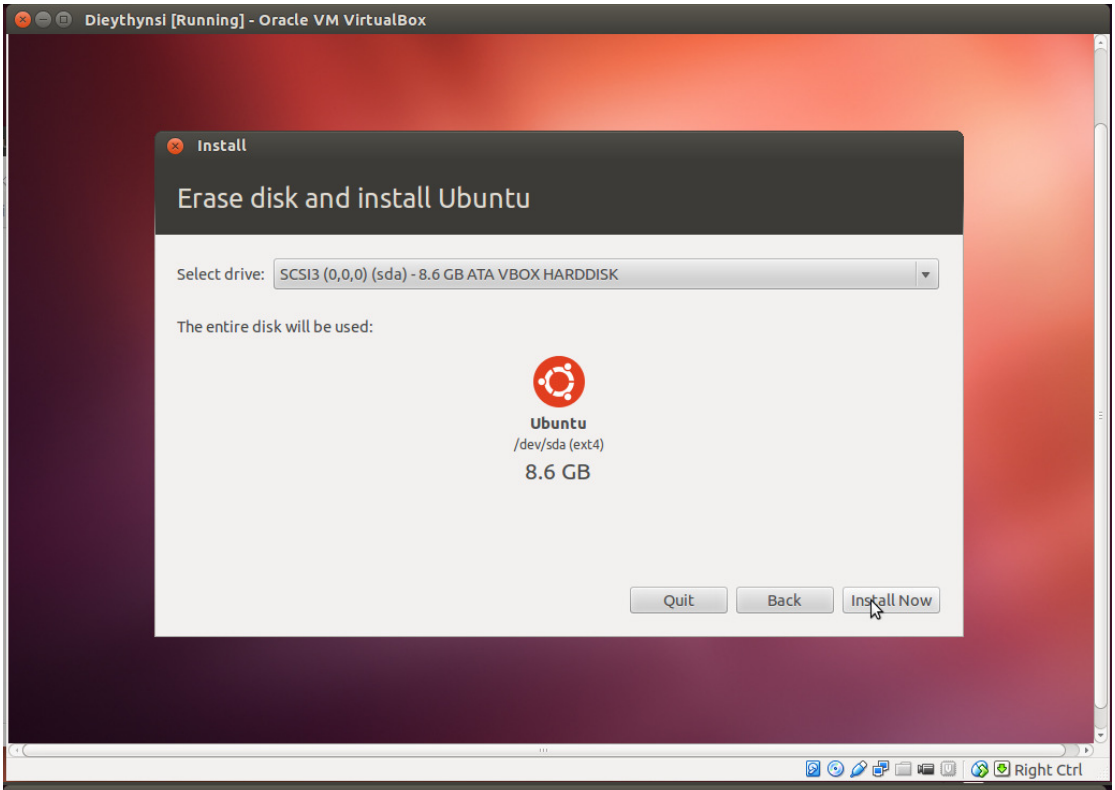
Μόλις ολοκληρωθεί η δημιουργία της εικονικής μηχανής θα περάσουμε ως λειτουργικό σύστημα το LinuxUbuntu 12.04 ως κύριο λειτουργικό σύστημα των τμημάτων της επιχείρησης. Θα το χρησιμοποιήσουμε για να μπορέσουμε να

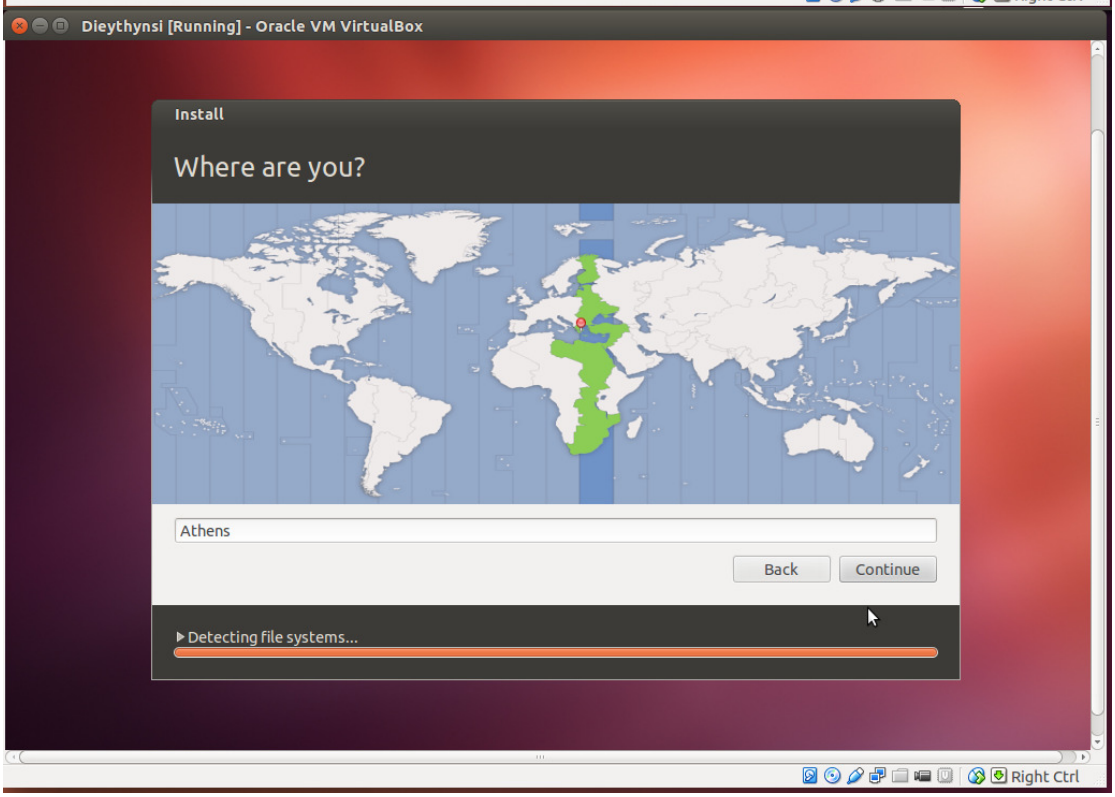
επιβεβαιώσουμε την επικοινωνία μεταξύ των υπόλοιπων εικονικών μηχανών που αποτελούν τα τμήματα για το σενάριο που υλοποιούμε.
Οπότε ακολουθεί η εγκατάσταση του λειτουργικού συστήματος Linux Ubuntu 12.04 στην εικονική μηχανή του τμήματος της Διεύθυνσης.

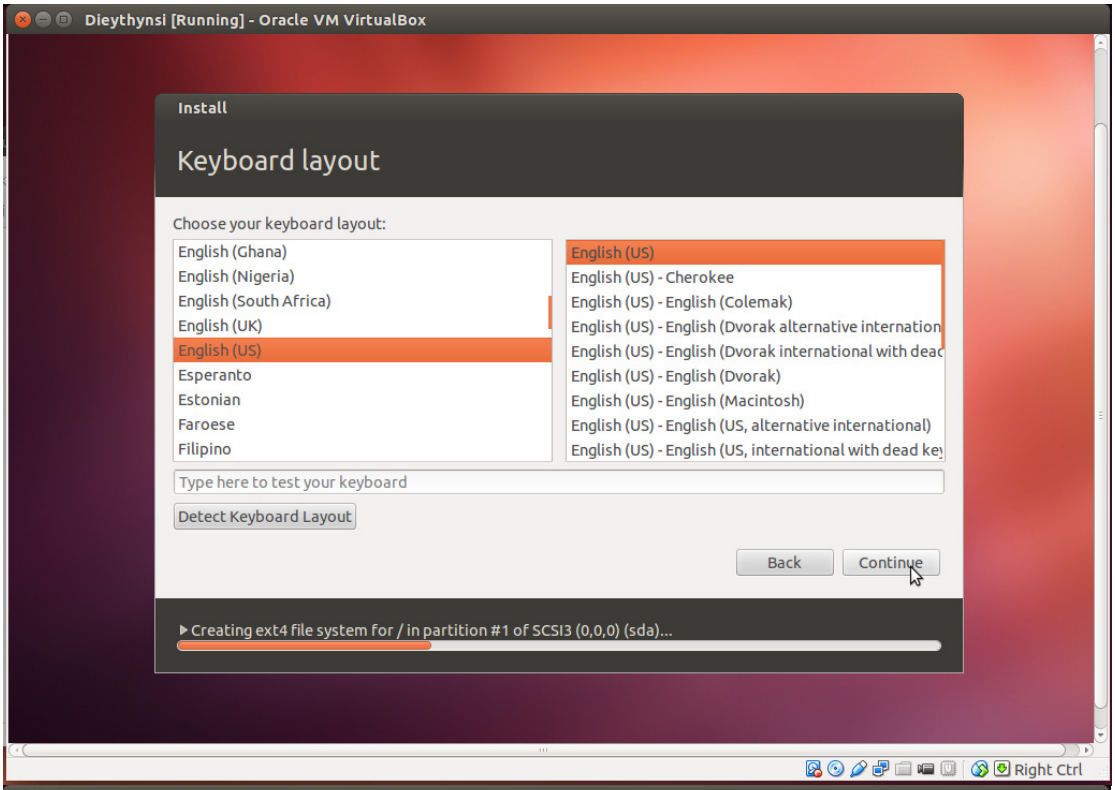

Dieythynsi [Running] - Oracle VM VirtualBox
Right Ctrl
Dieythynsi [Running] - Oracle VM VirtualBox
Install
Welcome
Asturianu
Bahasa Indonesia
Bosanski
Català
Čeština
Dansk
Deutsch
Eesti
English
Español
Esperanto
Euskara
Français
Gaeilge
Galego
Hrvatski
Íslenska
Italiano
Try Ubuntu
Install Ubuntu
You can try Ubuntu without making any changes to your computer, directly from this CD.
Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.
You may wish to read the release notes.
Right Ctrl

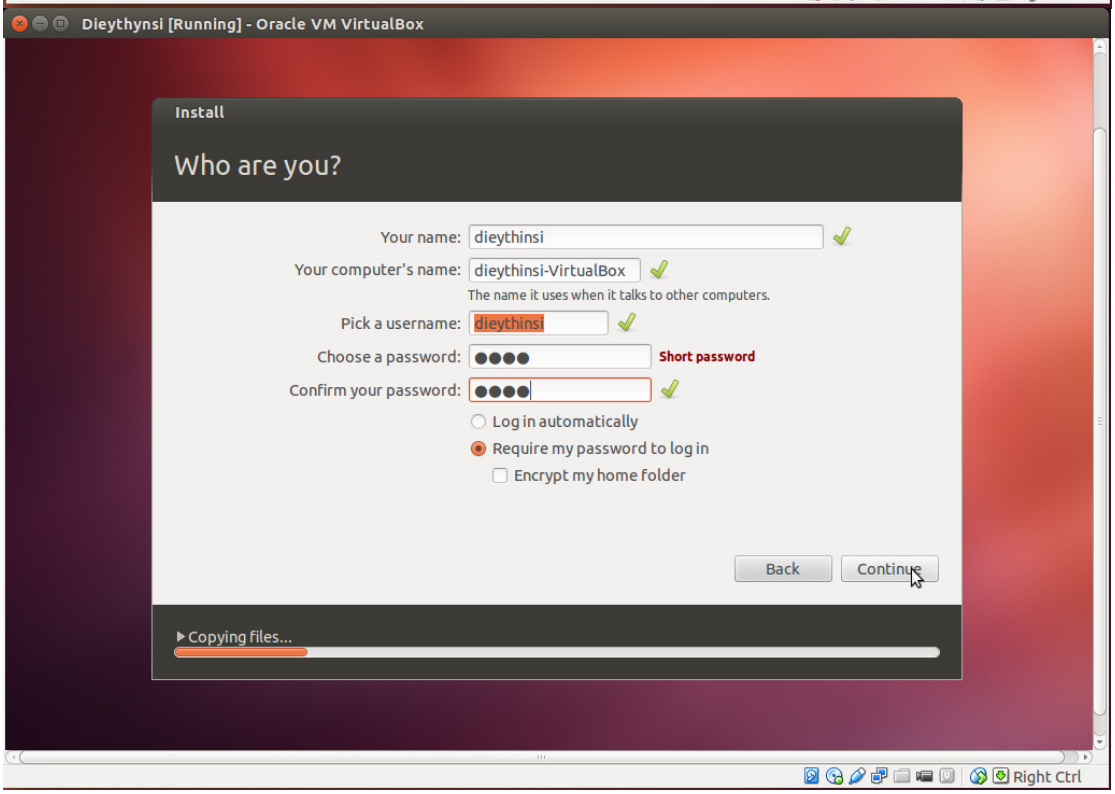

Dieythynsi [Running] - Oracle VM VirtualBox
Install
Preparing to install Ubuntu
For best results, please ensure that this computer:
has at least 4.6 GB available drive space
is plugged in to a power source
is connected to the Internet
Download updates while installing
Ubuntu uses third-party software to display Flash, MP3 and other media, and to work with some wireless hardware. Some of this software is closed-source. The software is subject to the license terms included with the software's documentation.
Install this third-party software
Fluendo MP3 plugin includes MPEG Layer-3 audio decoding technology licensed from Fraunhofer IIS and Technicolor SA.
Quit
Back
Continue
Right Ctrl

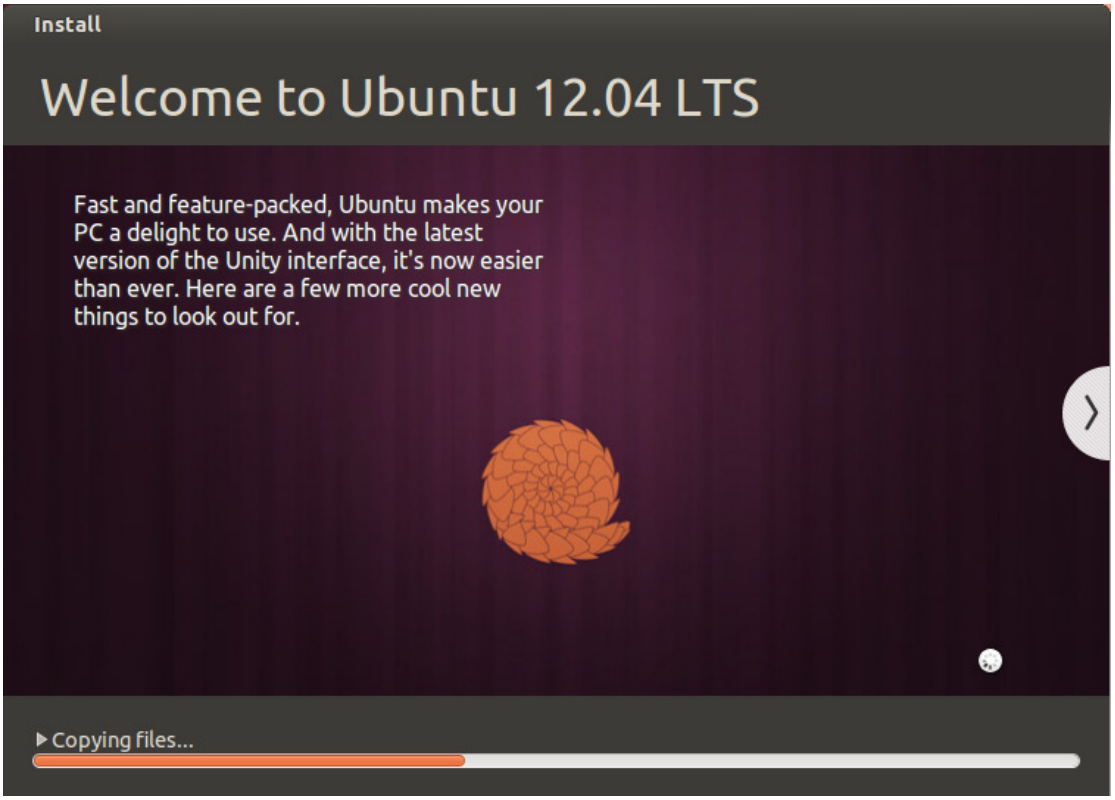

Dieythynsi [Running] - Oracle VM VirtualBox
Install
Installation type
This computer currently has no detected operating systems. What would you like to do?
Erase disk and install Ubuntu
Warning: This will delete any files on the disk.
Something else
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.
Quit
Back
Continue
Right Ctrl


Dieythynsi [Running] - Oracle VM VirtualBox
Install
Erase disk and install Ubuntu
Select drive:
SCSI3 (0,0,0) (sda) - 8.6 GB ATA VBOX HARDDISK
The entire disk will be used:
Ubuntu
/dev/sda (ext4)
8.6 GB
Quit
Back
Install Now
Right Ctrl

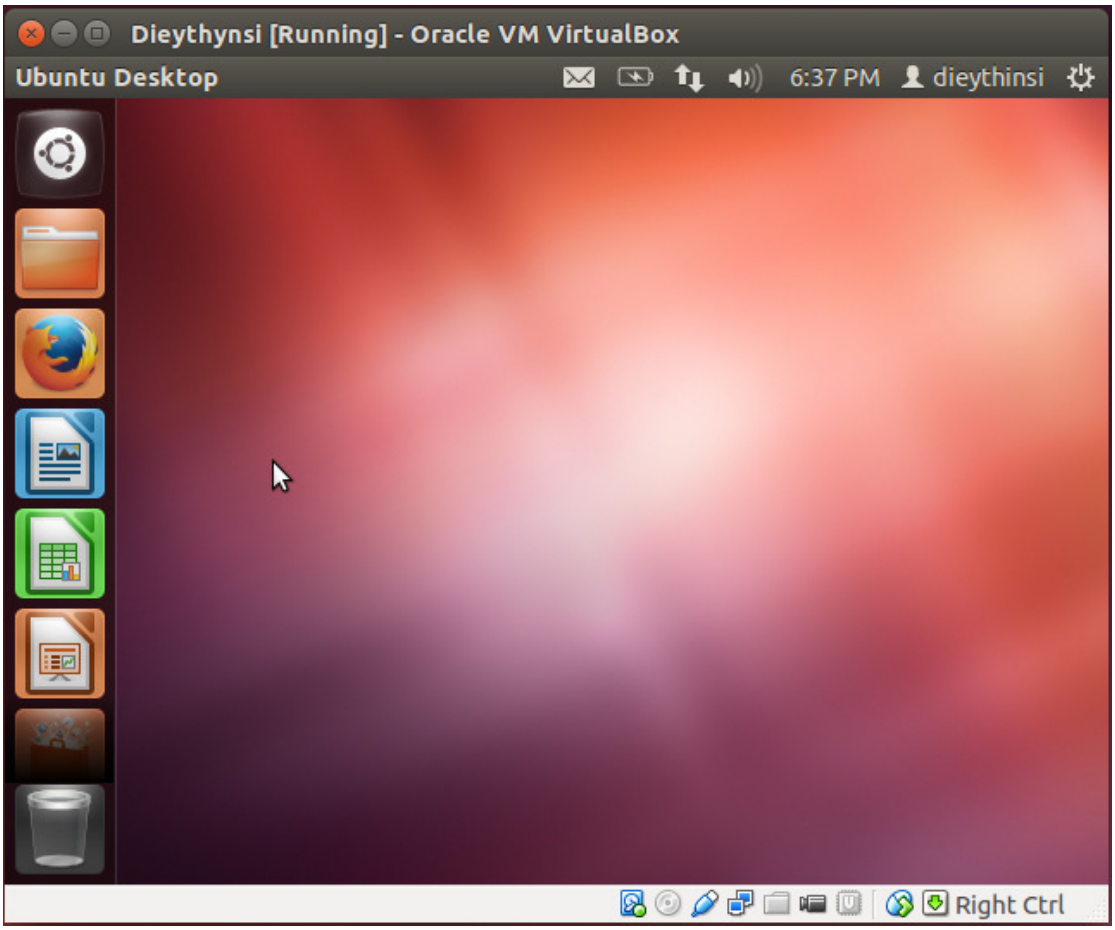

Dieythynsi [Running] - Oracle VM VirtualBox
Install
Where are you?
Athens
Back
Continue
Detecting file systems...
Right Ctrl


Dieythynsi [Running] - Oracle VM VirtualBox
Install
Keyboard layout
Choose your keyboard layout:
English (Ghana)
English (Nigeria)
English (South Africa)
English (UK)
English (US)
Esperanto
Estonian
Faroese
Filipino
English (US)
English (US) - Cherokee
English (US) - English (Colemak)
English (US) - English (Dvorak alternative internation
English (US) - English (Dvorak international with deac
English (US) - English (Dvorak)
English (US) - English (Macintosh)
English (US) - English (US, alternative international)
English (US) - English (US, international with dead ke
Type here to test your keyboard
Detect Keyboard Layout
Back
Continue
Creating ext4 file system for / in partition #1 of SCSI3 (0,0,0) (sda)...
Right Ctrl


Dieythynsi [Running] - Oracle VM VirtualBox
Install
Who are you?
Your name:
dieythinsi
Your computer's name:
dieythinsi-VirtualBox
The name it uses when it talks to other computers.
Pick a username:
dieythinsi
Choose a password:
Short password
Confirm your password:
Log in automatically
Require my password to log in
Encrypt my home folder
Back
Continue
Copying files...
Right Ctrl



Μόλις εγκατασταθεί το λειτουργικό σύστημα θα μας εμφανίσει την παρακάτω εικόνα:



Με την ίδια διαδικασία θα δημιουργήσουμε και τις εικονικές μηχανές για τα τμήματα της Παραγωγής και του Λογιστηρίου.

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:
Πριν την εγκατάσταση του Floodlight θα πρέπει να εγκαταστήσουμε κάποια βασικά πακέτα που χρειάζονται στο Linux:

- sudo apt-get install build-essential ant maven python-dev

```
kakabas@ubuntu:~$ sudo apt-get install build-essential ant maven python-dev
```

Στη συνέχεια κατεβάζουμε τον ελεγκτή floodlight από το repository και προχωράμε στην εγκατάστασή του:

- git clone git://github.com/floodlight/floodlight.git
- cd floodlight
- sudo apt-get install oracle-java8-installer
- git submodule init
- git submodule update
- ant
- mkdir /var/lib/floodlight
- sudochmod 777 /var/lib/floodlight

```
kakabas@ubuntu:~$ git clone git://github.com/floodlight/floodlight.git
Cloning into 'floodlight'...
remote: Counting objects: 52241, done.
remote: Compressing objects: 100% (46/46), done.
remote: Total 52241 (delta 49), reused 51 (delta 41), pack-reused 52146
Receiving objects: 100% (52241/52241), 383.60 MiB | 1.69 MiB/s, done.
Resolving deltas: 100% (33953/33953), done.
Checking connectivity... done.
Checking out files: 100% (800/800), done.
kakabas@ubuntu:~$ cd floodlight/
kakabas@ubuntu:~/floodlight$
kakabas@ubuntu:~/floodlight$ sudo apt-get install oracle-java8-installer
^Ckakabas@ubuntu:~/floodlight$ git submodule update
^Ckakabas@ubuntu:~/floodlight$ git submodule init
kakabas@ubuntu:~/floodlight$ ant
kakabas@ubuntu:~/floodlight$ mkdir /var/lib/floodlight/
kakabas@ubuntu:~/floodlight$ sudo 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

```
kakabas@ubuntu: ~/floodlight
kakabas@ubuntu:~/floodlight$ sudo java -jar target/floodlight.jar
```

Και επιβεβαιώνουμε την λειτουργία του εστιάζοντας στο “server on port 8080” όπως φαίνεται και στην εικόνα παρακάτω:

```
kakabas@ubuntu: ~/floodlight
2018-06-15 00:52:50.907 INFO  [n.f.h.HAController] Configuration parameters: {se
rverPort=127.0.0.1:4242, nodeid=1} 1
2018-06-15 00:52:50.977 INFO  [o.s.s.i.SyncManager] [1] Updating sync configurat
ion ClusterConfig [allNodes={1=Node [hostname=192.168.56.1, port=6642, nodeId=1,
 domainId=1], 2=Node [hostname=192.168.56.1, port=6643, nodeId=2, domainId=1], 3
=Node [hostname=192.168.56.1, port=6644, nodeId=3, domainId=1], 4=Node [hostname
=192.168.56.1, port=6645, nodeId=4, domainId=1]}, authScheme=CHALLENGE_RESPONSE,
 keyStorePath=/etc/floodlight/myKey.jceks, keyStorePassword is set]
2018-06-15 00:52:51.159 INFO  [o.s.s.i.r.RPCService] Listening for internal floo
dlight RPC on 0.0.0.0/0.0.0.0:6642
2018-06-15 00:52:51.235 INFO  [n.f.h.HAController] LDHAWorker is starting...
2018-06-15 00:52:51.238 INFO  [n.f.h.HAController] TopoHAWorker is starting...
2018-06-15 00:52:51.356 INFO  [n.f.h.AsyncElection] [AsyncElection] Priorities a
re not set.
2018-06-15 00:52:51.370 INFO  [n.f.h.HAController] HAController is starting...
2018-06-15 00:52:51.385 INFO  [n.f.h.ControllerLogic] [ControllerLogic] Running.
..
2018-06-15 00:52:51.392 INFO  [n.f.h.HAServer] Starting HAServer...
2018-06-15 00:52:51.654 INFO  [o.r.C.I.Server] Starting the Simple [HTTP/1.1] se
rver on port 8080
2018-06-15 00:52:51.654 INFO  [org.restlet] Starting net.floodlightcontroller.re
stserver.RestApiServer$RestApplication application
2018-06-15 00:52:56.865 INFO  [n.f.j.JythonServer] Starting DebugServer on :6655
```

6.2.3 OpenVSwitch

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

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.

- sudo apt-get update

```
kakabas@ubuntu:~$ sudo apt-get update
```

Στη συνέχεια θα εγκαταστήσουμε τις υπόλοιπες εξαρτήσεις.

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

```
kakabas@ubuntu:~$ sudo apt-get install -y git python-simplejson python-qt4 pytho
n-twisted-conch automake autoconf gcc uml-utilities libtool build-essential git
pkg-config
```

- sudo apt-get install –y linux-headers-`uname -r`

```
kakabas@ubuntu:~$ sudo apt-get install -y linux-headers-`uname -r`
```

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

```
kakabas@ubuntu:~$ 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

```
kakabas@ubuntu:~$ 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

```
kakabas@ubuntu:~$ cd openvswitch-2.4.0/
kakabas@ubuntu:~/openvswitch-2.4.0$ ./boot.sh
kakabas@ubuntu:~/openvswitch-2.4.0$ ./configure --with-linux=/lib/modules/`uname
-r`/build
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo -i
```

```
root@ubuntu:~# make & make install
```

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

- cd datapath/linux
- modprobe openvswitch

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@ubuntu: /home/kakabas/openvswitch-2.4.0
show utilities/bugtool/ovs-bugtool-coverage-show utilities/bugtool/ovs-bugtool-f
db-show utilities/bugtool/ovs-bugtool-lacp-show utilities/bugtool/ovs-bugtool-li
st-dbs utilities/bugtool/ovs-bugtool-memory-show utilities/bugtool/ovs-bugtool-t
c-class-show utilities/bugtool/ovs-bugtool-vsctl-show utilities/bugtool/ovs-bugt
ool-ovsdb-dump utilities/bugtool/ovs-bugtool-daemons-ver utilities/bugtool/ovs-b
ugtool-ovs-ofctl-show utilities/bugtool/ovs-bugtool-ovs-ofctl-dump-flows utiliti
es/bugtool/ovs-bugtool-ovs-appctl-dpif utilities/bugtool/ovs-bugtool-bond-show v
tep/ovs-vtep '/usr/local/share/openvswitch/scripts'
make[3]: Leaving directory `/home/kakabas/openvswitch-2.4.0'
make[2]: Leaving directory `/home/kakabas/openvswitch-2.4.0'
make[1]: Leaving directory `/home/kakabas/openvswitch-2.4.0'
root@ubuntu:/home/kakabas/openvswitch-2.4.0# cd datapath/linux
root@ubuntu:/home/kakabas/openvswitch-2.4.0/datapath/linux# modprobe openvswitch
root@ubuntu:/home/kakabas/openvswitch-2.4.0/datapath/linux# lsmod | grep openvsw
itch
openvswitch            81920  0
libcrc32c              16384  1 openvswitch
root@ubuntu:/home/kakabas/openvswitch-2.4.0/datapath/linux# touch /usr/local/etc
/ovs-vswitch.conf
root@ubuntu:/home/kakabas/openvswitch-2.4.0/datapath/linux# mkdir -p /usr/local/
etc/openvswitch
root@ubuntu:/home/kakabas/openvswitch-2.4.0/datapath/linux# cd ../..
root@ubuntu:/home/kakabas/openvswitch-2.4.0# ovsdb-tool create /usr/local/etc/op
envswitch/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

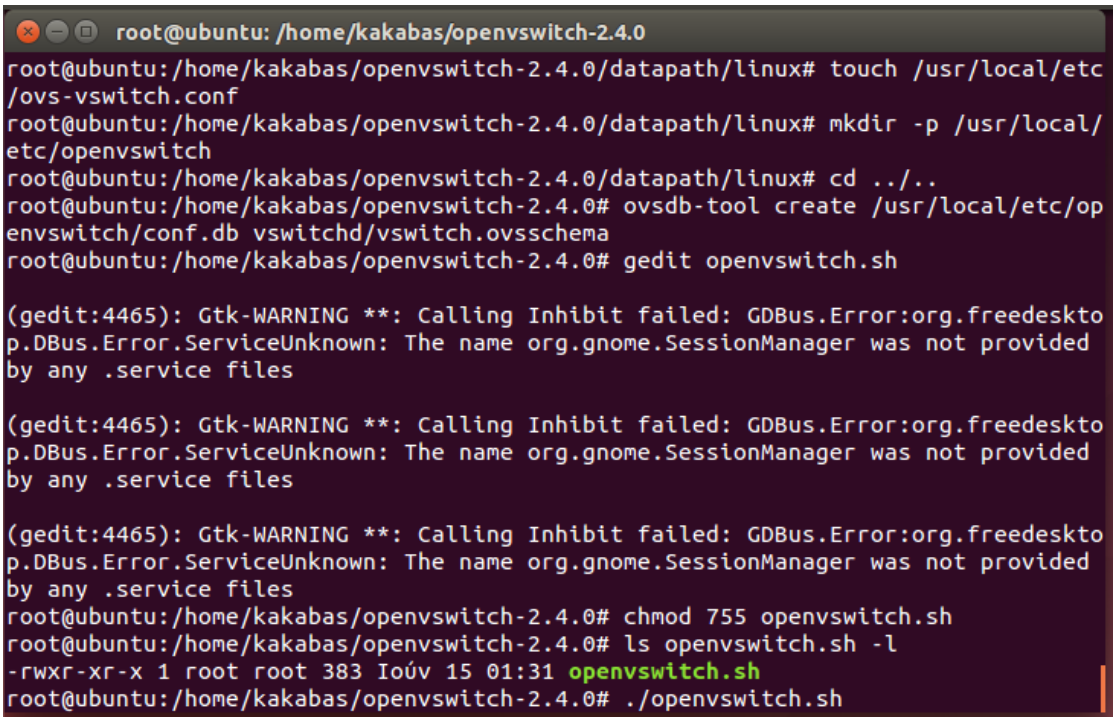
  ovs-vsctl --no-wait init
  ovs-vswitchd --pidfile --detach
  ovs-vsctl show

```
openvswitch.sh ×
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
```

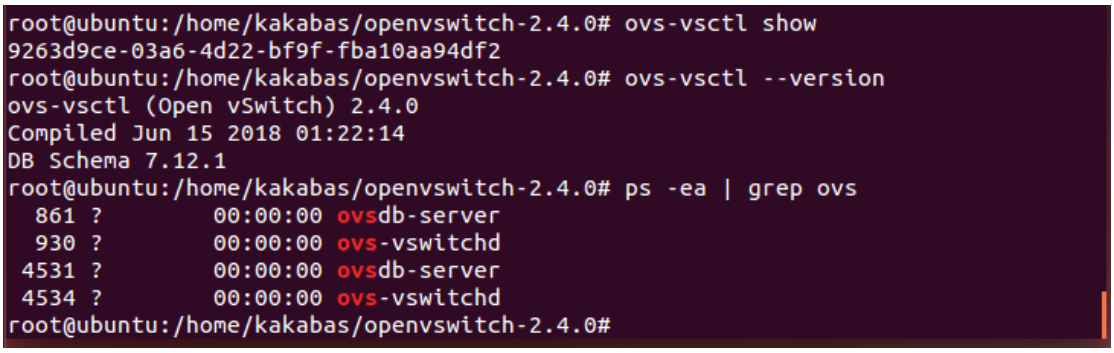
Στη συνέχεια θα αλλάξουμε την άδεια που έχει ο χρήστης μας για το αρχείο και θα το εκτελέσουμε:

- 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



Κεφάλαιο 7: Προσομοίωση

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

7.1 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 floodlightcontroller ανοίγουμε τον φάκελο που είναι εγκατεστημένος από το terminal του linux και δίνουμε την εντολή:

- sudo java –jar target/floodlight.jar

```
kakabas@ubuntu:~/floodlight$ sudo java -jar target/floodlight.jar
```

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

- sudo vs-vsctl add-br sdnkakabas
- sudo ovs-vsctl add-port sdnkakabas wlan0
- sudo ifconfig wlan0 0 up

```
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-vsctl add-br sdnkakabas
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-vsctl add-port sdnkakabas wlan0
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ifconfig wlan0 up
```

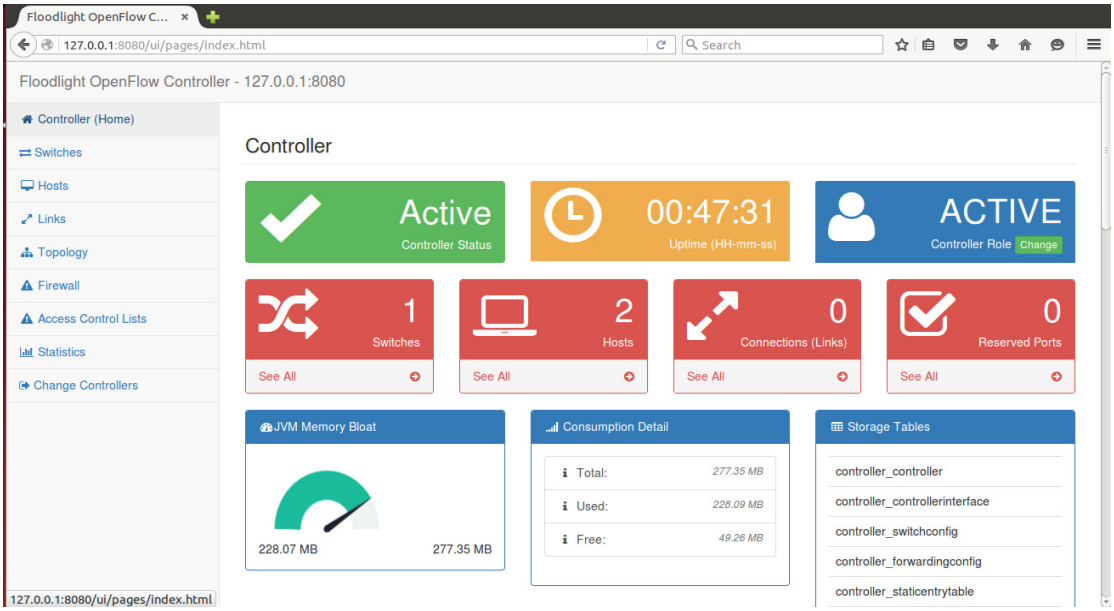
- sudo ifconfig sdnkakabas 192.168.43.99 netmask 255.255.255.0 up

```
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ifconfig sdnkakabas 192.168.43.99 netma
sk 255.255.255.0 up
```

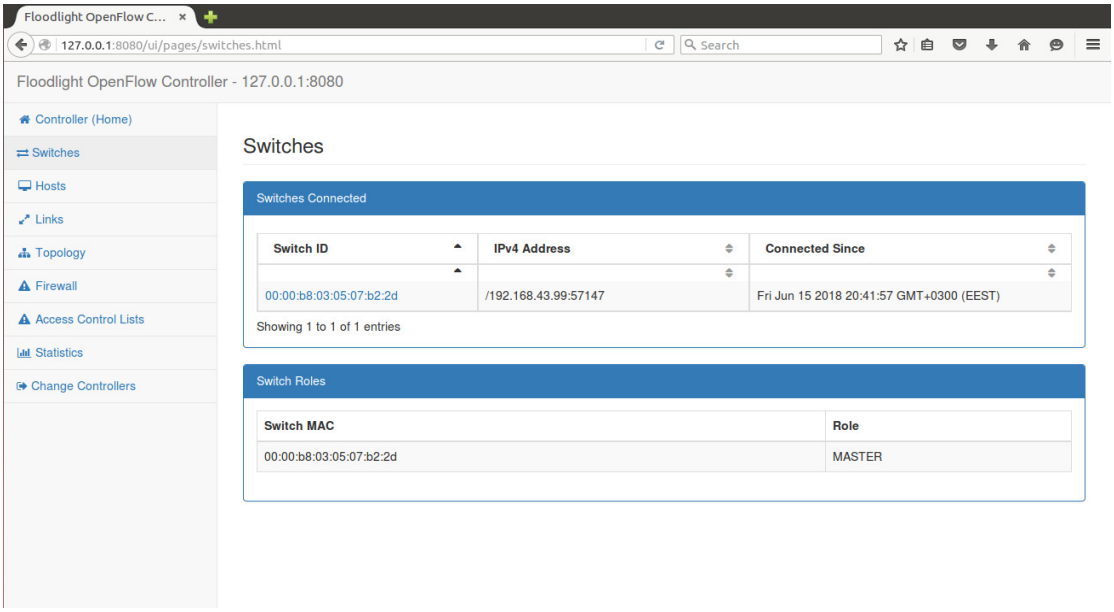
- sudo ovs-vsctl set-fail-mode sdnkakabas secure
- sudo ovs-vsctl set-controller sdnkakabas tcp:192.168.43.99:6653
- sudo ovs-vsctl show

```
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-vsctl set-fail-mode sdnkakabas secu
re
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-vsctl set-controller sdnkakabas tcp
:192.168.43.99:6653
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-vsctl show
9263d9ce-03a6-4d22-bf9f-fba10aa94df2
    Bridge sdnkakabas
        Controller "tcp:192.168.43.99:6653"
            is_connected: true
        fail_mode: secure
```

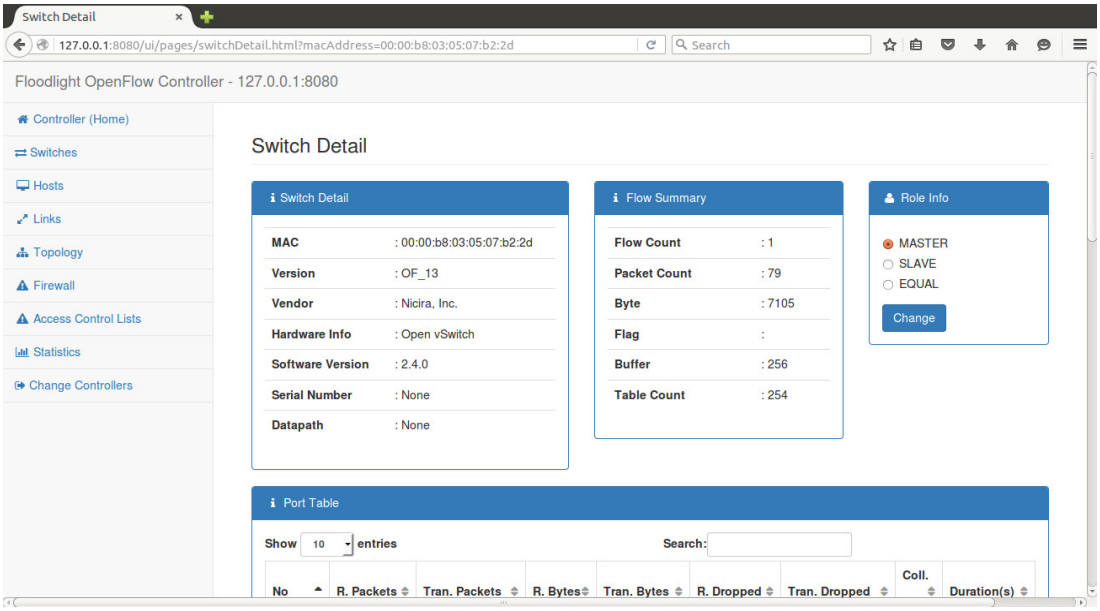
Από τη στιγμή αυτή μπορούμε να δούμε τη κατάσταση του openvswitch από το φυλλομετρητή στη διεύθυνση http://127.0.0.1:8080/ui/index.html



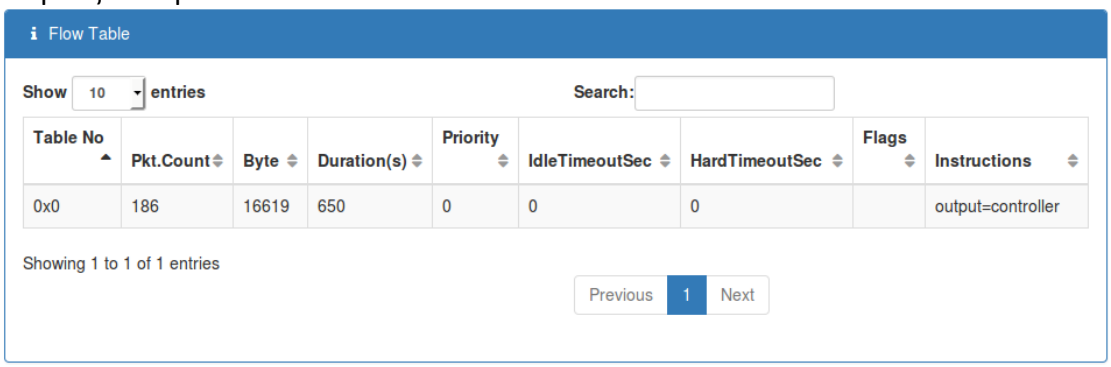
Από την παραπάνω εικόνα βλέπουμε ότι ο ελεγκτής Floodlight είναι ενεργός καθώς και ότι έχει συνδεθεί σε αυτόν ένας μεταγωγές (switch) κάνοντας κλικ στο “See All” από την ενότητα Switches μας παραπέμπει το σύστημα στη σελίδα που δείχνει τα switches:



Μπορούμε να δούμε και περισσότερες πληροφορίες, όπως την έκδοση OpenFlow που τρέχει, τις ροές δεδομένων που περιέχει κτλ, για το switch αρκεί να κάνουμε κλικ στο σύνδεσμο με τη διεύθυνση της κάρτας δικτύου:



Οι ροές δεδομένων:



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

- sudo ip tuntap add mode tap ovs1
- sudo ip link set ovs1 up
- sudo ovs-vsctl add-port sdnkakabas ovs1
- sudo ifconfig sdnkakabas up
- sudo ovs-vsctl show

```
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ip tuntap add mode tap ovs1
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ip link set ovs1 up
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-vsctl add-port sdnkakabas ovs1
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ifconfig sdnkakabas up
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-vsctl show
9263d9ce-03a6-4d22-bf9f-fba10aa94df2
    Bridge sdnkakabas
        Controller "tcp:192.168.43.99:6653"
            is_connected: true
        fail_mode: secure
        Port sdnkakabas
            Interface sdnkakabas
                type: internal
        Port "wlan0"
            Interface "wlan0"
        Port "ovs1"
            Interface "ovs1"
kakabas@ubuntu:~/openvswitch-2.4.0$
```

- ip link

```
12: ovs1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast master o
vs-system state DOWN mode DEFAULT group default qlen 500
    link/ether 0e:3e:82:e8:ab:d9 brd ff:ff:ff:ff:ff:ff
kakabas@ubuntu:~/openvswitch-2.4.0$
```

- if config

```
ovs1      Link encap:Ethernet  HWaddr 0e:3e:82:e8:ab:d9
          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)

sdnkakabas Link encap:Ethernet  HWaddr b8:03:05:07:b2:2d
          inet addr:192.168.43.99  Bcast:192.168.43.255  Mask:255.255.255.0
          inet6 addr: fe80::ba03:5ff:fe07:b22d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:156 errors:0 dropped:32 overruns:0 frame:0
          TX packets:174 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:11379 (11.3 KB)  TX bytes:17345 (17.3 KB)
```

- sudo ip tuntap add mode tap ovs2
- sudo ip link set ovs2 up
- sudo ovs-vsctl add-port sdnkakabas ovs2
- sudo ifconfig sdnkakabas up
- sudo ovs-vsctl show

```
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ip tuntap add mode tap ovs2
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ip link set ovs2 up
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-vsctl add-port sdnkakabas ovs2
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ifconfig sdnkakabas up
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-vsctl show
9263d9ce-03a6-4d22-bf9f-fba10aa94df2
    Bridge sdnkakabas
        Controller "tcp:192.168.43.99:6653"
            is_connected: true
        fail_mode: secure
        Port sdnkakabas
            Interface sdnkakabas
                type: internal
        Port "wlan0"
            Interface "wlan0"
        Port "ovs2"
            Interface "ovs2"
        Port "ovs1"
            Interface "ovs1"
kakabas@ubuntu:~/openvswitch-2.4.0$
```

- iplink

```
13: ovs2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast master o
vs-system state DOWN mode DEFAULT group default qlen 500
    link/ether 16:3b:8b:9c:14:d1 brd ff:ff:ff:ff:ff:ff
kakabas@ubuntu:~/openvswitch-2.4.0$
```

- ifconfig

```
ovs2      Link encap:Ethernet  HWaddr 16:3b:8b:9c:14:d1
          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)
```

- sudo iptuntap add mode tap ovs3
- sudo ip link set ovs3 up

- sudo ovs-vsctl add-port sdnkakabas ovs3
- sudo ifconfigsdnkakabas up
- sudo ovs-vsctl show

```
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ip tuntap add mode tap ovs3
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ip link set ovs3 up
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-vsctl add-port sdnkakabas ovs3
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ifconfig sdnkakabas up
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-vsctl show
9263d9ce-03a6-4d22-bf9f-fba10aa94df2
    Bridge sdnkakabas
        Controller "tcp:192.168.43.99:6653"
            is_connected: true
        fail_mode: secure
        Port sdnkakabas
            Interface sdnkakabas
                type: internal
        Port "wlan0"
            Interface "wlan0"
        Port "ovs3"
            Interface "ovs3"
        Port "ovs2"
            Interface "ovs2"
        Port "ovs1"
            Interface "ovs1"
kakabas@ubuntu:~/openvswitch-2.4.0$
```

- iplink

```
14: ovs3: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast master o
vs-system state DOWN mode DEFAULT group default qlen 500
    link/ether 1e:aa:ff:48:96:5c brd ff:ff:ff:ff:ff:ff
kakabas@ubuntu:~/openvswitch-2.4.0$
```

- ifconfig

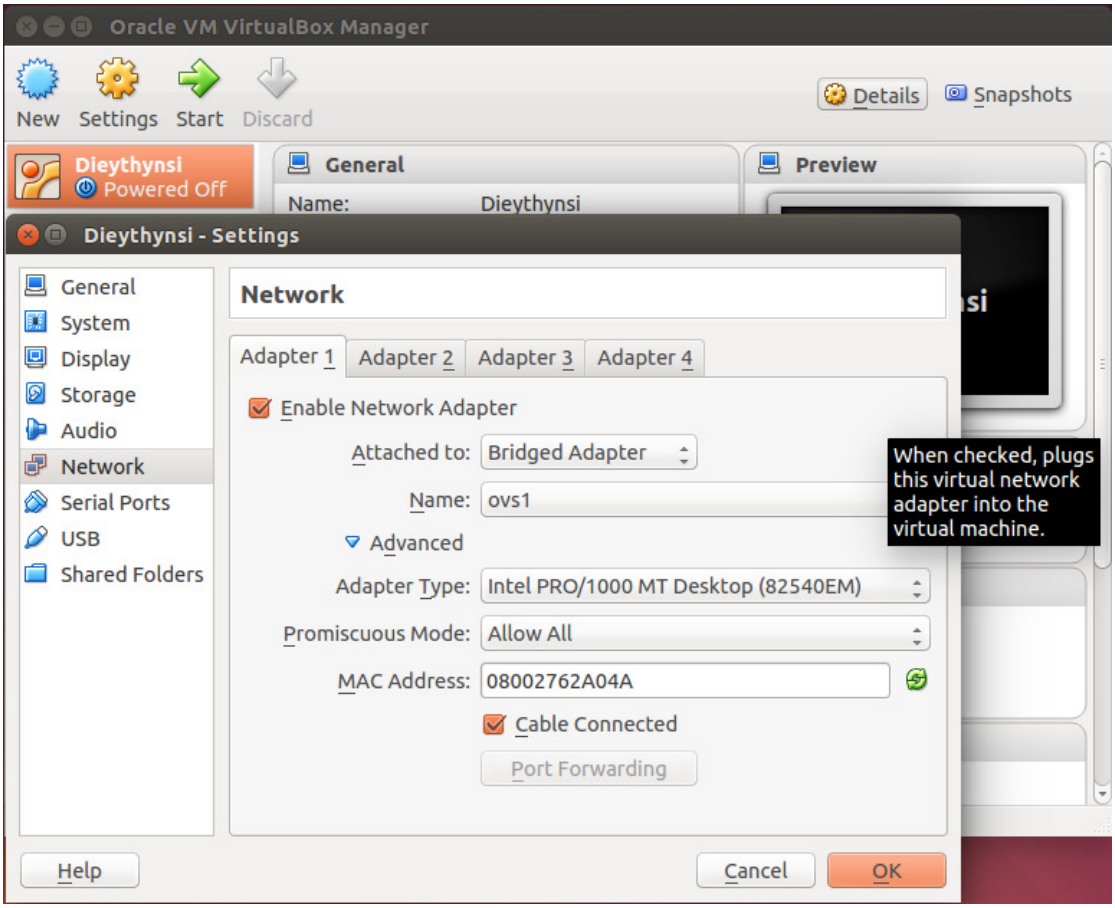
```
ovs3      Link encap:Ethernet  HWaddr 1e:aa:ff:48:96:5c
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- ovs-ofctl dump-flows sdnkakabas

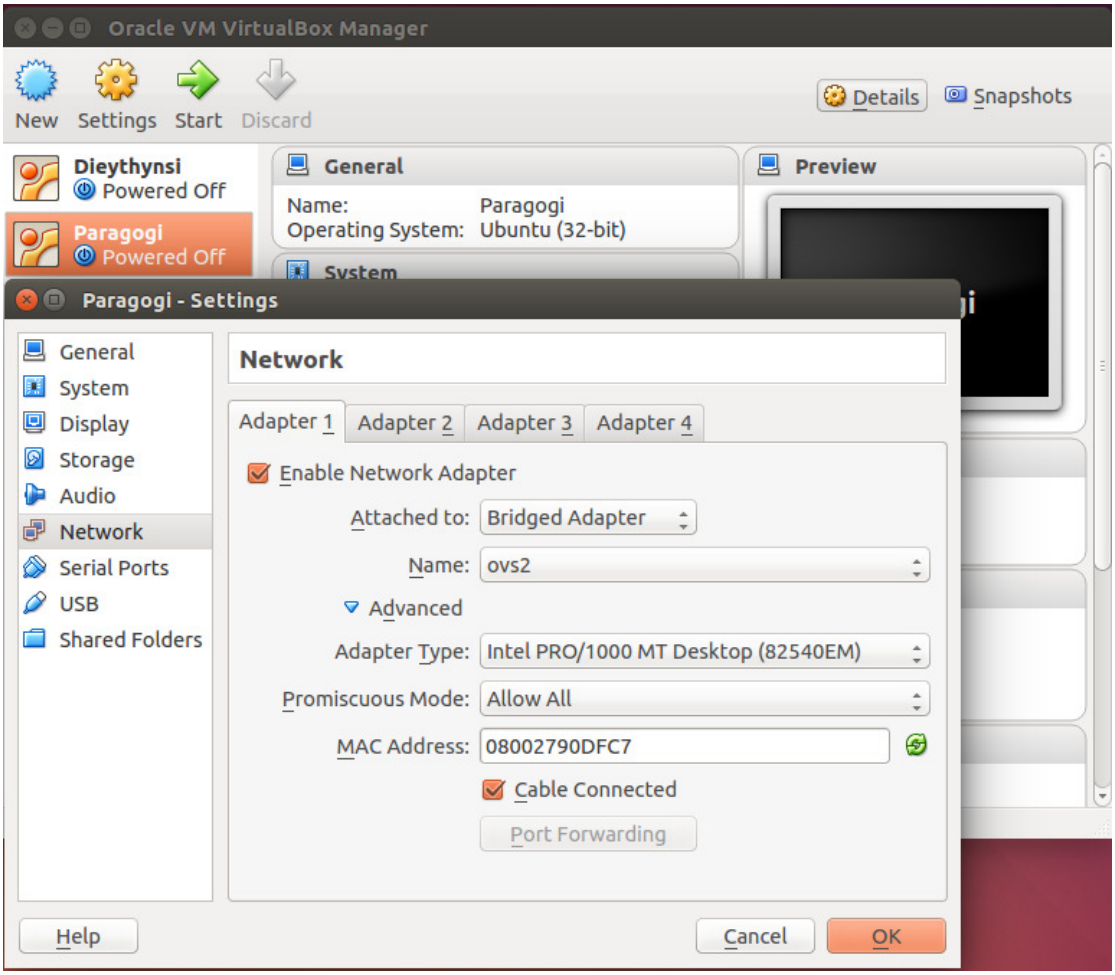
```
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-vsctl dump-flows sdnkakabas
ovs-vsctl: unknown command 'dump-flows'; use --help for help
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-ofctl dump-flows sdnkakabas
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=715.724s, table=0, n_packets=202, n_bytes=18267, idle_age=
44, priority=0 actions=CONTROLLER:65535
kakabas@ubuntu:~/openvswitch-2.4.0$
```

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

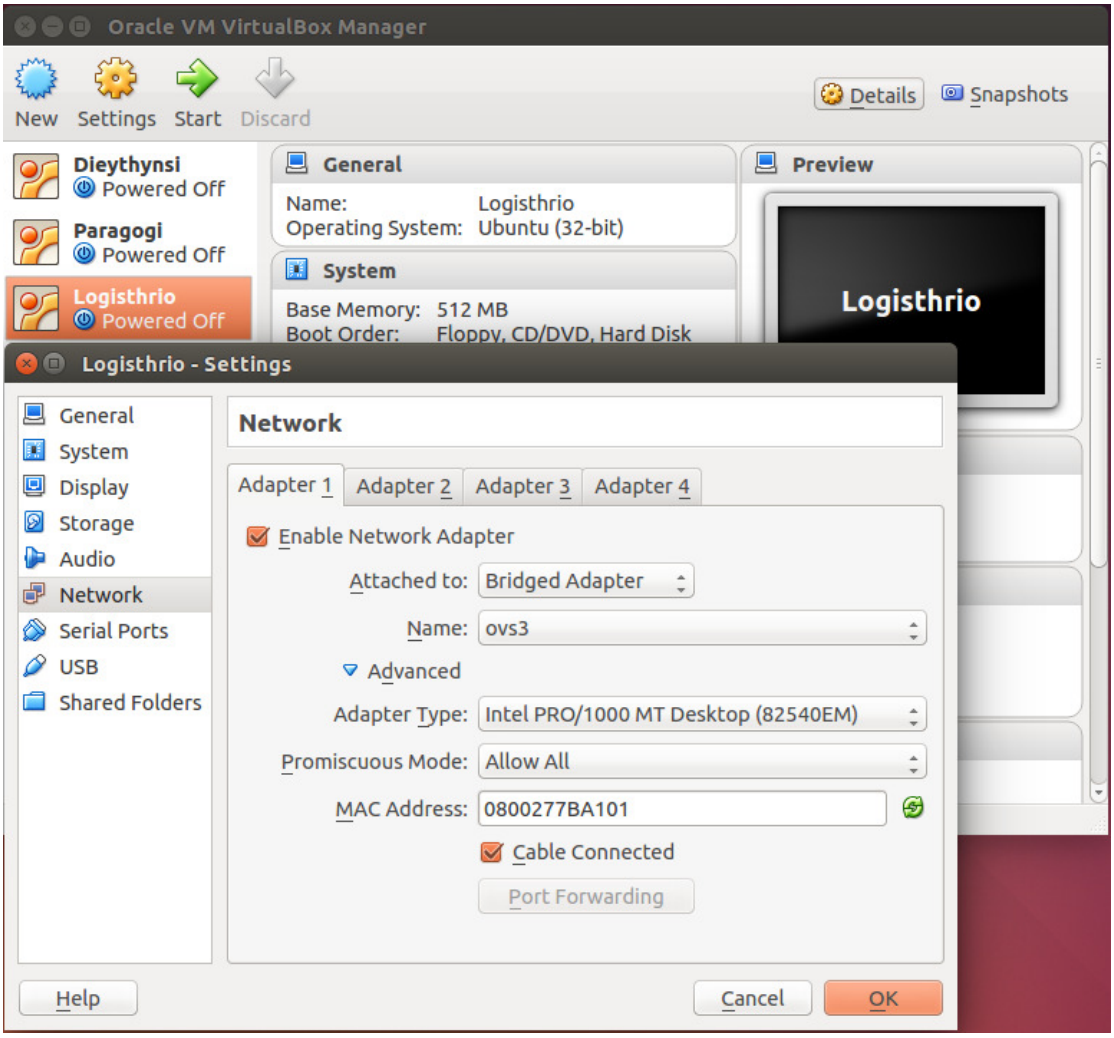
Αφού ανοίξουμε το Virtualbox επιλέγουμε την εικονική μηχανή “Dieythynsi” και πατάμε το κουμπί “Settings” στο παράθυρο που μας ανοίγει επιλέγουμε την καρτέλα “Network” από το πλάγιο μενού και στο πεδίο “Attached to” επιλέγουμε “Bridged Adapter”, στο πεδίο “Name” επιλέγουμε “ovs1” και στο πεδίο “Promiscuous Mode” των “Advanced” επιλογών επιλέγουμε το “Allow All”.



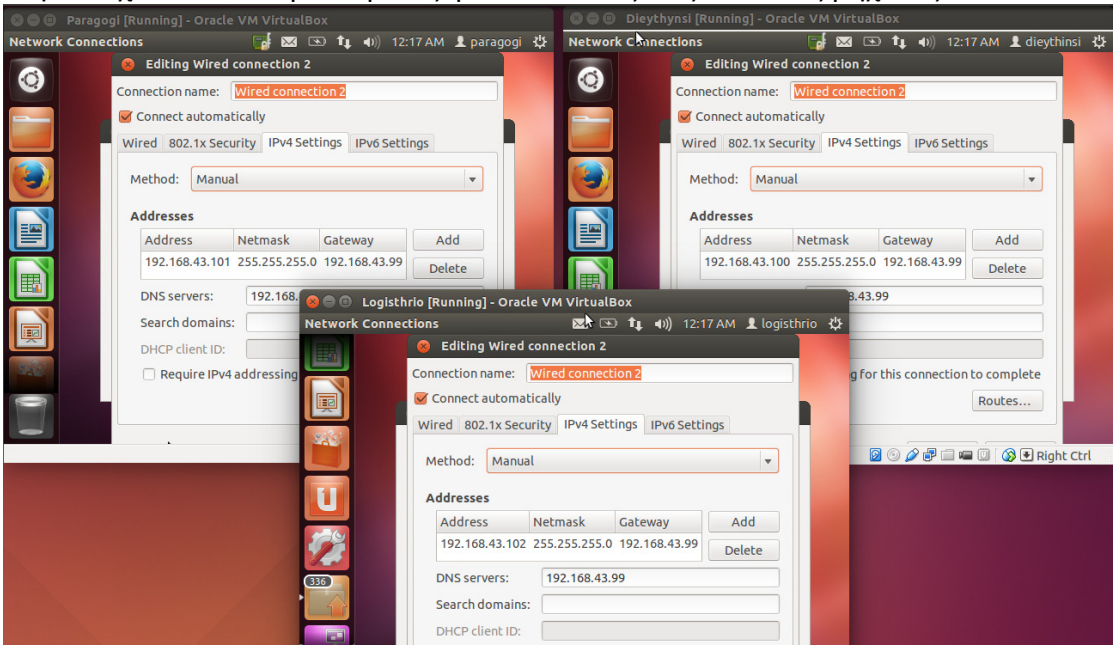
Στη συνέχεια επιλέγουμε την εικονική μηχανή “Paragogi” και πατάμε το κουμπί “Settings” στο παράθυρο που μας ανοίγει επιλέγουμε την καρτέλα “Network” από το πλάγιο μενού και στο πεδίο “Attached to” επιλέγουμε “Bridged Adapter”, στο πεδίο “Name” επιλέγουμε “ovs2” και στο πεδίο “Promiscuous Mode” των “Advanced” επιλογών επιλέγουμε το “Allow All”.



Στη συνέχεια επιλέγουμε την εικονική μηχανή “Logistirio” 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 “ovs3” και στο πεδίο “Promiscuous Mode” των “Advanced” επιλογών επιλέγουμε το “Allow All”.

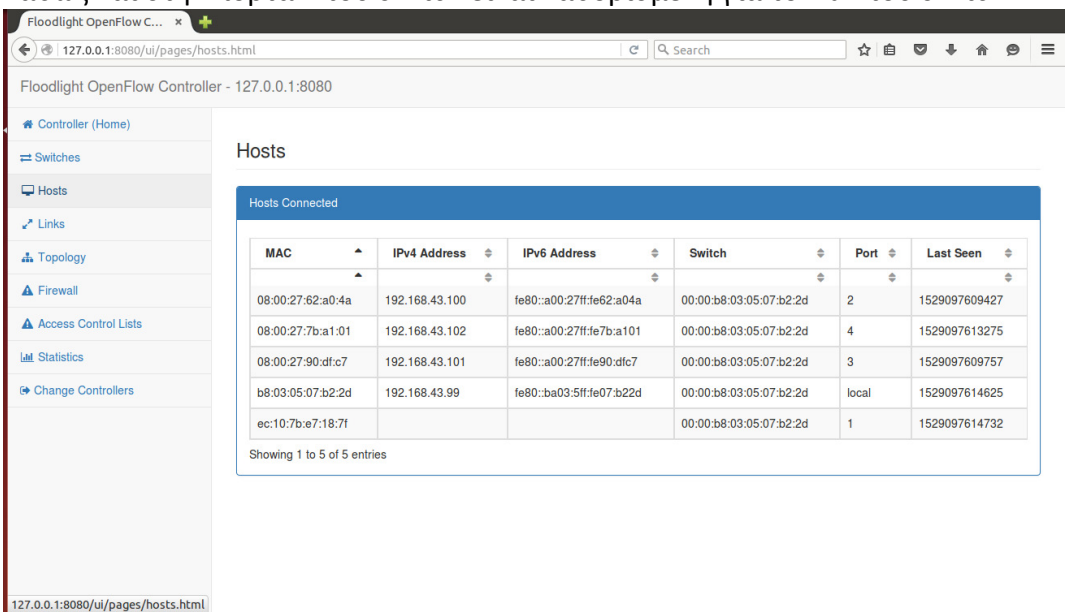


Στη συνέχεια θα καθορίσουμε τις ip διευθύνσεις στις εικονικές μηχανές:

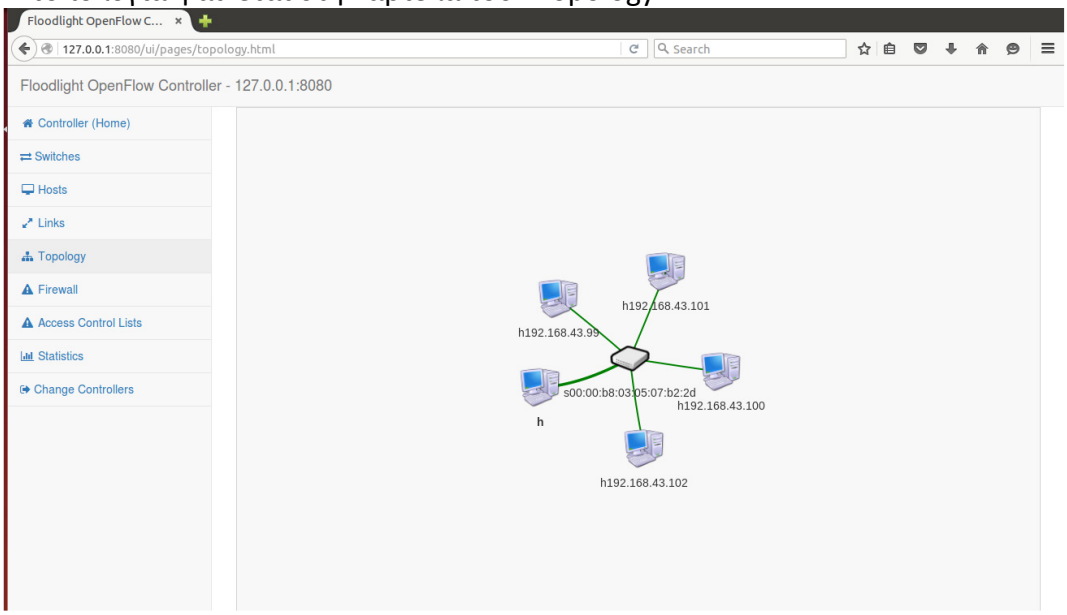


Μετά ανοίγουμε το φυλλομετρητή στη διεύθυνση http://127.0.0.1:8080/ui/index.html και επιβεβαιώνουμε την σύνδεση των εικονικών μηχανών στο openvswitch και την τοπολογία του δικτύου.

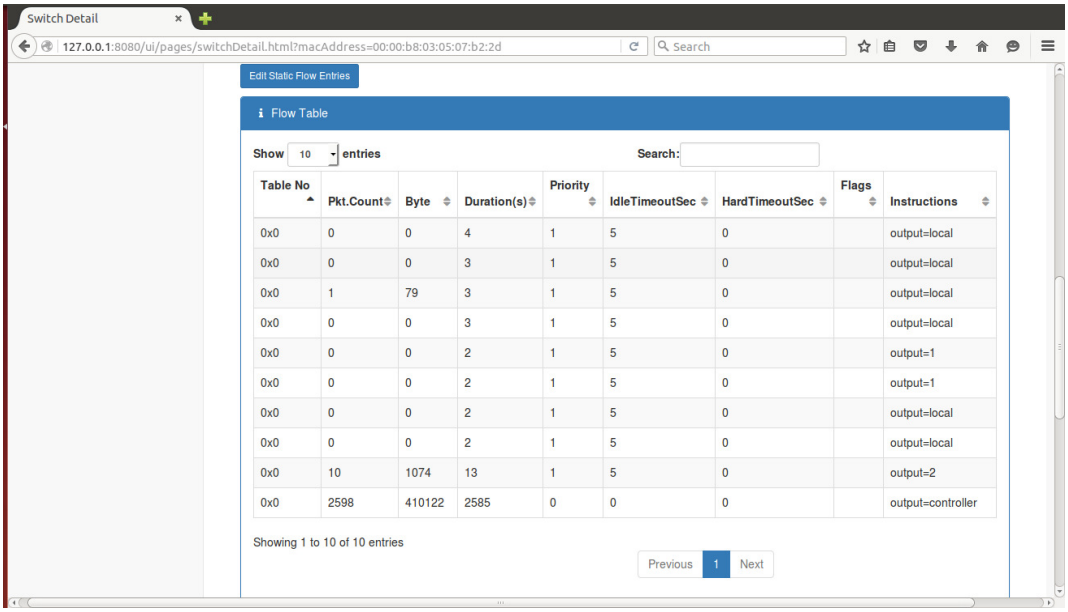
Οι συνδεδεμένες μηχανές φαίνονται στη καρτέλα του “Hosts” με την ip που έχουν. Όπως προκύπτει από το δίκτυο που έχουμε δημιουργήσει βλέπουμε την ip διεύθυνση του τμήματος της Διεύθυνσης με την ip: 192.168.43.100 να είναι συνδεδεμένο με την πόρτα 2 του Switch, την ip διεύθυνση του τμήματος της Παραγωγής με την ip: 192.168.43.101 να είναι συνδεδεμένο με την πόρτα 3 του Switch, την ip διεύθυνση του τμήματος του Λογιστηρίου με την ip: 192.168.43.102 να είναι συνδεδεμένο με την πόρτα 4 του Switch. Στο switch είναι επίσης συνδεδεμένη τοπικά (local) η επαφή διαχείρισης του μεταγωγέα ip: 192.168.43.99 καθώς και στην πόρτα 1 του switch είναι καθορισμένη για το vlan του switch.



Η τοπολογία φαίνεται στη καρτέλα του “Topology”.



Οι ροές δεδομένων προστίθενται αυτόματα από τον ελεγκτή στο switch για την τοπολογία που έχουμε δημιουργήσει, οπότε μπορούμε να δούμε τις ροές από τον πίνακα ροών του μεταγωγέα (switch).



Μπορούμε να επιβεβαιώσουμε τις ροές επικοινωνίας του switch και από το τερματικό του Linux με την εξής εντολή:

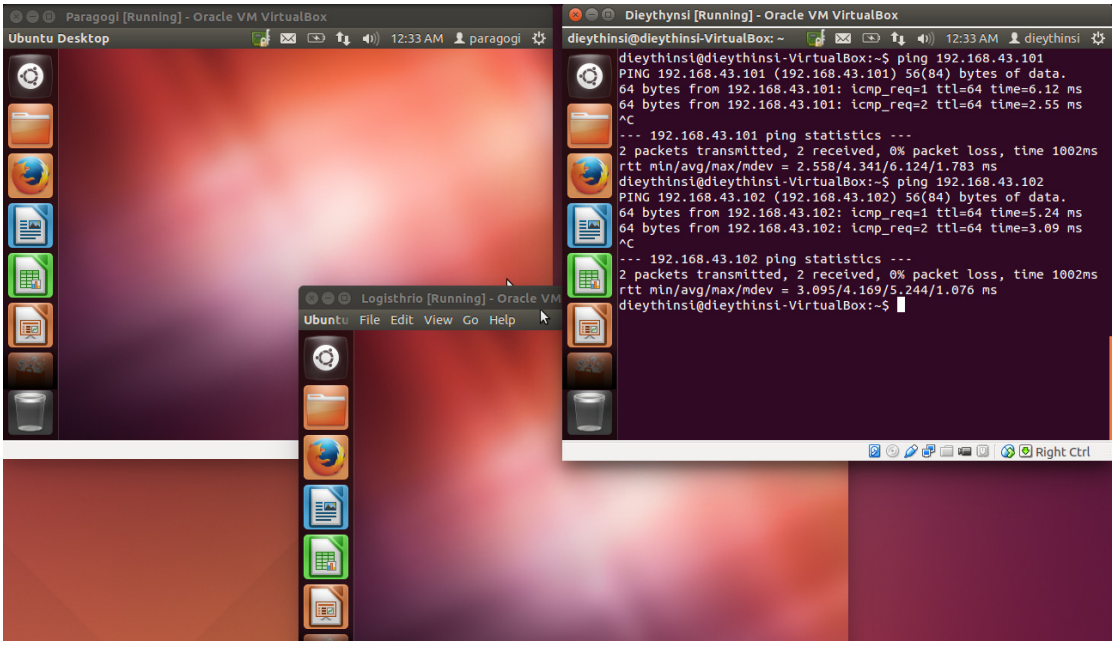
- ovs-ofctl dump-flows sdnkakabas

```
kakabas@ubuntu: ~/openvswitch-2.4.0
kakabas@ubuntu:~/openvswitch-2.4.0$ sudo ovs-ofctl dump-flows sdnkakabas
NXST_FLOW reply (xid=0x4):
 cookie=0x2000055a000000, duration=9.398s, table=0, n_packets=1, n_bytes=82, idl
e_timeout=5, idle_age=4, priority=1,udp,in_port=3,dl_src=08:00:27:90:df:c7,dl_ds
t=b8:03:05:07:b2:2d,nw_src=192.168.43.101,nw_dst=192.168.43.99,tp_src=51688,tp_d
st=53 actions=LOCAL
 cookie=0x2000055c000000, duration=5.808s, table=0, n_packets=1, n_bytes=82, idl
e_timeout=5, idle_age=0, priority=1,udp,in_port=4,dl_src=08:00:27:7b:a1:01,dl_ds
t=b8:03:05:07:b2:2d,nw_src=192.168.43.102,nw_dst=192.168.43.99,tp_src=41798,tp_d
st=53 actions=LOCAL
 cookie=0x2000055b000000, duration=9.395s, table=0, n_packets=1, n_bytes=110, id
le_timeout=5, idle_age=4, priority=1,ip,in_port=LOCAL,dl_src=b8:03:05:07:b2:2d,d
l_dst=08:00:27:90:df:c7,nw_src=192.168.43.99,nw_dst=192.168.43.101 actions=outpu
t:3
 cookie=0x2000055d000000, duration=5.806s, table=0, n_packets=1, n_bytes=110, id
le_timeout=5, idle_age=0, priority=1,ip,in_port=LOCAL,dl_src=b8:03:05:07:b2:2d,d
l_dst=08:00:27:7b:a1:01,nw_src=192.168.43.99,nw_dst=192.168.43.102 actions=outpu
t:4
 cookie=0x2000055e000000, duration=0.802s, table=0, n_packets=0, n_bytes=0, idle
_timeout=5, idle_age=0, priority=1,arp,in_port=4,dl_src=08:00:27:7b:a1:01,dl_dst
=b8:03:05:07:b2:2d actions=LOCAL
 cookie=0x2000055f000000, duration=0.800s, table=0, n_packets=0, n_bytes=0, idle
_timeout=5, idle_age=0, priority=1,arp,in_port=LOCAL,dl_src=b8:03:05:07:b2:2d,dl
_dst=08:00:27:7b:a1:01 actions=output:4
 cookie=0x0, duration=2541.457s, table=0, n_packets=2578, n_bytes=408275, idle_a
ge=0, priority=0 actions=CONTROLLER:65535
kakabas@ubuntu:~/openvswitch-2.4.0$
```

7.2 Έλεγχος Επικοινωνίας

Μετά από την επιβεβαίωση των ροών δεδομένων (Flows) μπορούμε να ελέγξουμε την επικοινωνία των τμημάτων της επιχείρησης.

Στην παρακάτω εικόνα βλέπουμε την επικοινωνία που υπάρχει μεταξύ των τμημάτων της Διεύθυνσης, της Παραγωγής και του Λογιστηρίου.



7.3 Συμπεράσματα

Μέσα από το πρακτικό κομμάτι είδαμε να επιβεβαιώνεται η θεωρία του Software Defined Networking με την χρήση του Openflow που ουσιαστικά οι ροές δεδομένων που περάσανε στο πίνακα του μεταγωγέα επιτρέψανε την επικοινωνία μεταξύ των τμημάτων της επιχείρησης που είχαμε σαν περίπτωση χρήσης του SDN. Πλέον μέσω του προγραμματιζόμενου μεταγωγέα μπορούμε εύκολα να μεταβάλουμε την τοπολογία και την ροή των δεδομένων από ένα κεντρικό σύστημα διαχείρισης που στη παρούσα εργασία είναι ο ελεγκτής Floodlight στο δίκτυο που έχουμε. Αυτό μας δίνει την δυνατότητα να ελέγχουμε το δίκτυο και την κατάστασή του, να μπορούμε να εφαρμόσουμε τεχνικές Ποιότητα της Υπηρεσίας (QoS), Ασφάλειας και πολλές εφαρμογές δικτύων.

Κεφάλαιο 8: Αξιολόγηση αποτλεσμάτων - 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.

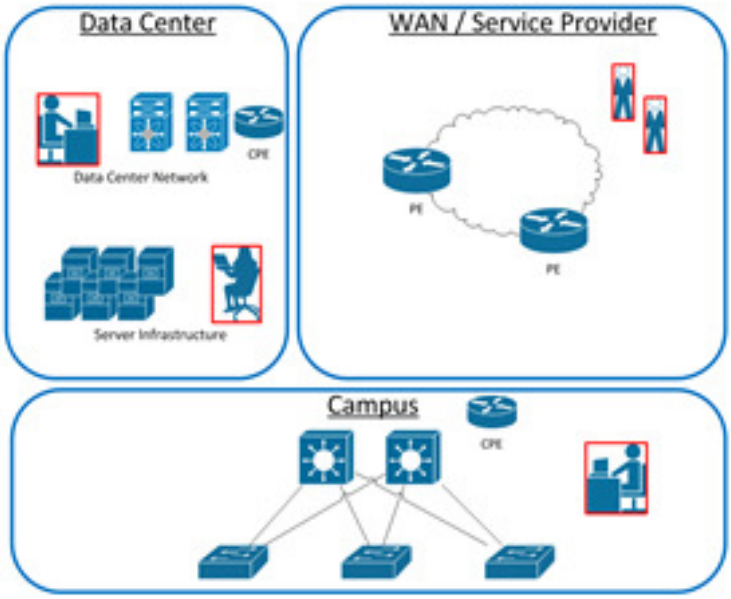
Πέρα όμως των θεωρικών πληροφοριών που μας εμβάθυναν στην τεχνολογία που μελετάμε, παρουσιάσαμε την υλοποίηση του SDN σε δύο στάδια. Το πρώτο ήταν πειραματικό και είχε σκοπό την εξοικείωση με τα συστήματα υλοποίησης καθώς και να κατανοήσουμε πως λειτουργεί στην πράξη η θεωρία γι' αυτό την παρουσιάστηκε 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 μίας επιχείρησης που περιλαμβάνει τρία τμήματα (Διεύθυνσης, Παραγωγής, Λογιστηρίου) και πως αυτά αλληλεπιδρούν μεταξύ τους παίρνοντας πληροφορίες από μία μικρή μονάδα παραγωγής. Προσεγγίσαμε με επιτυχία την υλοποίηση ενός τέτοιου δικτύου με την χρήση του SDN και αναλογιζόμενοι τα πλεονεκτήματα που έχει, έχουμε σχεδιάσει έτσι το δίκτυό μας ώστε να είναι ευέλικτο σε οποιαδήποτε σενάριο θα μπορούσαμε στο μέλλον να υλοποιήσουμε. Περιπτώσεις χρήσης θα μπορούσαν να είναι η εφαρμογή κανόνων Ποιότητα της Υπηρεσίας, Ασφάλειας αλλά και απλή χρήση VLAN έτσι ώστε κάποια από τα επιμέρους δίκτυα των τμημάτων της επιχείρησης να μην μπορούν να επικοινωνούν μεταξύ τους.

8.1 Χρησιμότητα

Η σημασία ύπαρξης της ευελιξίας των σεναρίων που μπορούμε να έχουμε μέσα σε ένα περιβάλλον δικτύου είναι αρκετά σημαντική και χρήσιμη. Για αυτό το λόγο θεωρήσαμε σημαντική την περίπτωση χρήσης του SDN γιατί μπορεί να λύσει αποτελεσματικά τα προβλήματα QoS, Ασφαλείας, κτλ που υπάρχουν μέσα σε ένα δικτύο.

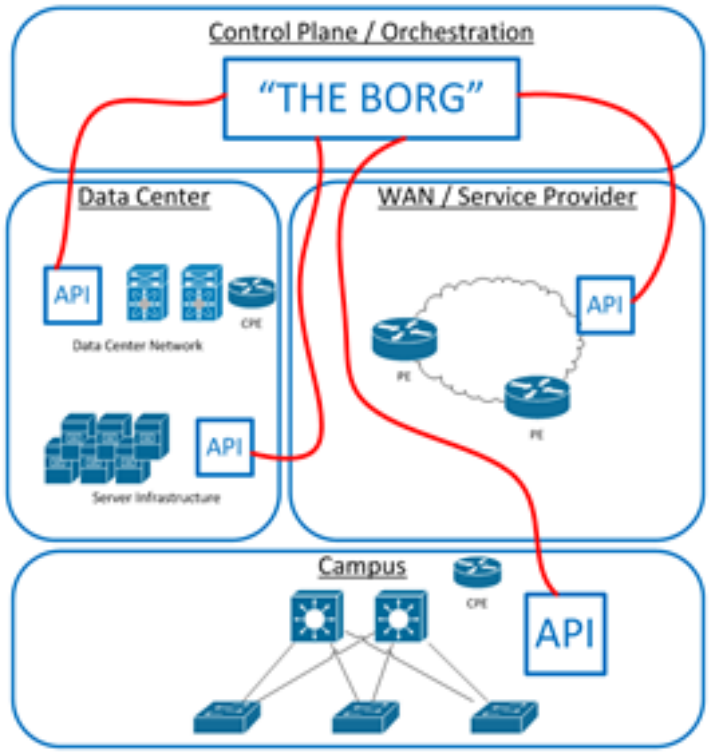
Το δίκτυο μέσα σε μία μικρή ή μεγάλη επιχείρηση προσεγγίζεται ως κάτι ξεχωριστό και υποστηρίζεται από άτομα που διαθέτουν συγκεκριμένες γνώσεις και δυνατότητες για να κάνουν εφαρμογή της τεχνολογίας αυτής μέσα στην επιχείρηση.

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 παραδείγματος χάρη για την υλοποίηση μίας πολιτικής QoS μέσα στην επιχείρηση όλα τα παραπάνω έχουν συγκεκριμένο ρόλο και πρέπει να ληφθούν υπόψη.



Εικόνα 8.1: Τεχνικοί Δικτύων
Πηγή: Jeong et al., 2014

Αυτά βέβαια ισχύουν αναλογιζόμενοι ότι η παραπάνω τεχνολογία βρίσκεται οργανωμένη μέσα στην επιχείρηση ή οργανισμούς (Data Center, Παροχών Τηλεπικοινωνίας, Πανεπιστημιακά Δίκτυα, Επιχειρήσεις, 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 8.2: Χρησιμότητα API
Πηγή: Jeong et al., 2014

Ως αποτέλεσμα των παραπάνω είναι ότι μία πολιτική ποιότητας της υπηρεσίας (QoS) για παράδειγμα μπορεί τελικά να μην είναι μία 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

Ahmed, R., & Boutaba, R. (2014). Design considerations for managing wide area software defined networks. IEEE Communications Magazine, 52(7), 116-123.

Akyildiz, I. F., Lee, A., Wang, P., Luo, M., & Chou, W. (2014). A roadmap for traffic engineering in SDN-OpenFlow networks. Computer Networks, 71, 1-30.

Al-Shaer, E., & Al-Haj, S. (2010, October). FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures. In Proceedings of the 3rd ACM workshop on Assurable and usable security configuration (pp. 37-44). ACM.

Antikainen, M., Aura, T., & Särelä, M. (2014, October). Spook in your network: Attacking an sdn with a compromised openflow switch. In Nordic Conference on Secure IT Systems (pp. 229-244). Springer, Cham.

Araniti, G., Cosmas, J., Iera, A., Molinaro, A., Morabito, R., & Orsino, A. (2014, June). OpenFlow over wireless networks: Performance analysis. In Broadband Multimedia Systems and Broadcasting (BMSB), 2014 IEEE International Symposium on(pp. 1-5). IEEE.

AZIZ, M. Z. B. A., & Okamura, K. (2016). An Analysis of Botnet Attack for SMTP Server using Software Define Network (SDN). Proceedings of the Asia-Pacific Advanced Network, 42, 21-26.

Azodolmolky, S., Wieder, P., & Yahyapour, R. (2013). Cloud computing networking: challenges and opportunities for innovations. IEEE Communications Magazine, 51(7), 54-62.

Bailey, S., Bansal, D., Dunbar, L., Hood, D., Kis, Z. L., MackCrane, B., ... & Schaller, S. (2013). SDN Architecture Overview. Open Networking Foundation, Ver, 1.

Banerjee, S., & Kannan, K. (2014, November). Tag-in-tag: Efficient flow table management in sdn switches. In Network and Service Management (CNSM), 2014 10th International Conference on (pp. 109-117). IEEE.

Bates, A., Butler, K., Haeberlen, A., Sherr, M., & Zhou, W. (2014, February). Let SDN be your eyes: Secure forensics in data center networks. In Proceedings of the NDSS workshop on security of emerging network technologies (SENT'14).

Bernardo, D. V., & Chua, B. B. (2015, March). Introduction and analysis of SDN and NFV security architecture (SN-SECA). In Advanced Information Networking

and Applications (AINA), 2015 IEEE 29th International Conference on (pp. 796-801). IEEE.

Bernardos, C. J., De La Oliva, A., Serrano, P., Banchs, A., Contreras, L. M., Jin, H., & Zúñiga, J. C. (2014). An architecture for software defined wireless networking. IEEE wireless communications, 21(3), 52-61.

Blendin, J. (2013). Cross-layer optimization of peer-to-peer video streaming in OpenFlow-based ISP networks (Doctoral dissertation, Diploma Thesis, Technische Universität Darmstadt, http://www. ps. tu-darmstadt. de/fileadmin/publications/Ble13. pdf).

Blenk, A., Basta, A., Zerwas, J., Reisslein, M., & Kellerer, W. (2016). Control plane latency with sdn network hypervisors: The cost of virtualization. IEEE Transactions on Network and Service Management, 13(3), 366-380.

Boon, J. M. (2017). Why do innovation specialist firms engage with Open Source projects? (Master's thesis, University of Twente).

Borges, L. M., Velez, F. J., & Lebres, A. S. (2014). Survey on the characterization and classification of wireless sensor network applications. IEEE Communications Surveys & Tutorials, 16(4), 1860-1890.

Braun, W., & Menth, M. (2014). Software-defined networking using OpenFlow: Protocols, applications and architectural design choices. Future Internet, 6(2), 302-336.

Butler, B., (2017) SD-WAN: What is it and why you'll use it one day, Networkworld [online] available at: https://www.networkworld.com/article/3031279/sd-wan/sd-wan-what-it-is-and-why-you-ll-use-it-one-day.html [access 28-10-2017]

Canini, M., Venzano, D., Peresini, P., Kostic, D., & Rexford, J. (2012). A NICE way to test OpenFlow applications. In Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI) (No. EPFL-CONF-170618).

Carlos, A. M., Rothenberg, C. E., & Maurício, F. M. (2010, December). In-packet Bloom filter based data center networking with distributed OpenFlow controllers. In GLOBECOM Workshops (GC Wkshps), 2010 IEEE (pp. 584-588). IEEE.

Christodoulopoulos, K., Palkopoulou, E., Angelou, M., Klonidis, D., Klekamp, A., Buchali, F., ... & Tomkos, I. (2012). Quantifying spectrum, cost, and energy

efficiency in fixed-grid and flex-grid networks. *IEEE/OSA Journal of Optical Communications and Networking*, *4*(11), B42-B51.

Casellas, R., Muñoz, R., Martínez, R., Vilalta, R., Liu, L., Tsuritani, T., ... & Fernández-Palacios, J. P. (2014, March). SDN based provisioning orchestration of OpenFlow/GMPLS flexi-grid networks with a stateful hierarchical PCE. In Optical Fiber Communications Conference and Exhibition (OFC), 2014 (pp. 1-3). IEEE.

Cass, S. (2015). The 2015 top ten programming languages. IEEE Spectrum, July, 20.

Chen, X., & Wu, T. (2017, January). Towards the Semantic Web Based Northbound Interface for SDN Resource Management. In Semantic Computing (ICSC), 2017 IEEE 11th International Conference on (pp. 40-47). IEEE.

Chen, Z., Dong, W., Li, H., Zhang, P., Chen, X., & Cao, J. (2014). Collaborative network security in multi-tenant data center for cloud computing. Tsinghua Science and Technology, 19(1), 82-94.

Cheung, O., Thomas, D., & Patrick, S. (2010). New approaches to e-reserve: Linking, sharing and streaming. Elsevier.

Costanzo, S. Next Generation Networking with SDN and NFV.

Costa-Requena, J., Santos, J. L., Guasch, V. F., Ahokas, K., Premsankar, G., Luukkainen, S., ... & Ylianttila, M. (2015, June). SDN and NFV integration in generalized mobile network architecture. In Networks and Communications (EuCNC), 2015 European Conference on (pp. 154-158). IEEE.

Curtis, M., Liljenstolpe, C., Pollitt, A., Dupre, M. G. H., & Harrison, E. P. (2015). U.S. Patent Application No. 14/838,928.

De Oliveira, R. L. S., Shinoda, A. A., Schweitzer, C. M., & Prete, L. R. (2014, June). Using mininet for emulation and prototyping software-defined networks. In Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on (pp. 1-6). IEEE.

Dogar, F. R., Karagiannis, T., Ballani, H., & Rowstron, A. (2014, August). Decentralized task-aware scheduling for data center networks. In ACM SIGCOMM Computer Communication Review(Vol. 44, No. 4, pp. 431-442). ACM.

Dolan, M. J., Davidson, E. M., Kockar, I., Ault, G. W., & McArthur, S. D. (2012). Distribution power flow management utilizing an online optimal power flow technique. IEEE Transactions on Power Systems, 27(2), 790-799.

Doverspike, R., Clapp, G., Douyon, P., Freimuth, D. M., Gullapalli, K., Han, B., ... & Pastor, J. (2015). Using SDN technology to enable cost-effective bandwidth-on-demand for cloud services. Journal of Optical Communications and Networking, 7(2), A326-A334.

Erickson, D. (2013, August). The beacon openflow controller. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (pp. 13-18). ACM.

Extreme Networks (2014) «Different Take on SDN»

Feamster, N. (2010, September). Outsourcing home network security. In Proceedings of the 2010 ACM SIGCOMM workshop on Home networks (pp. 37-42). ACM.

Feamster, N., (2014) M1.3 SDN History: Network Virtualization, 2014. [Online]. Available at: https://www.youtube.com/watch?v=vsbyyNFg5BM&list=PLpherdrLyny8YN4M24iRJB MCXkLcGbmhY&index=4.

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

Feamster, N., Rexford, J., & Zegura, E. (2014). The road to SDN: an intellectual history of programmable networks. ACM SIGCOMM Computer Communication Review, 44(2), 87-98.

Franchi, E., Poggi, A., & Tomaiuolo, M. (2013). Open social networking for online collaboration. International Journal of e-Collaboration (IJeC), 9(3), 50-68.

Fundation, O. N. (2012). Software-defined networking: The new norm for networks. ONF White Paper, 2, 2-6.

Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., & Shenker, S. (2008). NOX: towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 38(3), 105-110.

Guo, Z., Xu, Y., Cello, M., Zhang, J., Wang, Z., Liu, M., & Chao, H. J. (2015). JumpFlow: Reducing flow table usage in software-defined networks. Computer Networks, 92, 300-315.

Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D. C., & Gayraud, T. (2014). Software-defined networking: Challenges and research opportunities for future internet. Computer Networks, 75, 453-471.

Haleplidis, E., Pentikousis, K., Denazis, S., Salim, J. H., Meyer, D., & Koufopavlou, O. (2015). Software-defined networking (SDN): Layers and architecture terminology (No. RFC 7426).

Hammadi, A., & Mhamdi, L. (2014). A survey on architectures and energy efficiency in data center networks. Computer Communications, 40, 1-21.

Harans M., (2016) 16 Hot networking products putting the Sizzle in SD-WAN, CRN [online] available at: http://www.crn.com/slide-shows/networking/300082325/16-hot-networking-products-putting-the-sizzle-in-sd-wan.htm?itc=hp_slideshow [access 29-10-2017]

Hassas Yeganeh, S., & Ganjali, Y. (2012, August). Kandoo: a framework for efficient and scalable offloading of control applications. In Proceedings of the first workshop on Hot topics in software defined networks (pp. 19-24). ACM.

Heller, B. (2009) Openflow switch specification, version 1.0. 0. Wire. December.

Heller, B. (2011) Openflow switch specification, version 1.0. 0. Wire. December.

Heller, B., Sherwood, R., & McKeown, N. (2012, August). The controller placement problem. In Proceedings of the first workshop on Hot topics in software defined networks (pp. 7-12). ACM.

Hong, S., Xu, L., Wang, H., & Gu, G. (2015, February). Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures. In NDSS.

Hu, F. (Ed.). (2014). Network Innovation through OpenFlow and SDN: Principles and Design. CRC Press.

Huang, S., Griffioen, J., & Calvert, K. L. (2014). Network hypervisors: Enhancing sdn infrastructure. Computer Communications, 46, 87-96.

Infotechlead (2014) Interop 2014: Avaya to showcase Automated Campus part of SDN initiative [online] available at: http://www.infotechlead.com/mobility/interop-2014-avaya-showcase-automated-campus-part-sdn-initiative-21223 [access 29-10-2017]

Jackson, C., Nejabati, R., Agraz, F., Pagès, A., Galili, M., Spadaro, S., & Simeonidou, D. (2017, March). Demonstration of the benefits of SDN technology for all-

optical data centre virtualisation. In Optical Fiber Communications Conference and Exhibition (OFC), 2017 (pp. 1-2). IEEE.

Jackson, E. J., Walls, M., Panda, A., Pettit, J., Pfaff, B., Rajahalme, J., ... & Shenker, S. (2016, June). SoftFlow: A Middlebox Architecture for Open vSwitch. In USENIX Annual Technical Conference (pp. 15-28).

Jadeja, Y., & Modi, K. (2012, March). Cloud computing-concepts, architecture and challenges. In Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on (pp. 877-880). IEEE.

Jafarian, J. H., Al-Shaer, E., & Duan, Q. (2012, August). Openflow random host mutation: transparent moving target defense using software defined networking. In Proceedings of the first workshop on Hot topics in software defined networks (pp. 127-132). ACM.

Jain, R., & Paul, S. (2013). Network virtualization and software defined networking for cloud computing: a survey. IEEE Communications Magazine, 51(11), 24-31.

Jammal, M., Singh, T., Shami, A., Asal, R., & Li, Y. (2014). Software defined networking: State of the art and research challenges. Computer Networks, 72, 74-98.

Jeong, C., Ha, T., Narantuya, J., Lim, H., & Kim, J. (2014, October). Scalable network intrusion detection on virtual SDN environment. In Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on (pp. 264-265). IEEE.

Jin, R., & Wang, B. (2013, March). Malware detection for mobile devices using software-defined networking. In Research and Educational Experiment Workshop (GREE), 2013 Second GENI (pp. 81-88). IEEE.

Kampanakis, P., Perros, H., & Beyene, T. (2014, June). SDN-based solutions for moving target defense network protection. In World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a (pp. 1-6). IEEE.

Khan, H., Khayam, S. A., Rajarajan, M., Golubchik, L., & Orr, M. (2013). Wirespeed, privacy-preserving P2P traffic detection on commodity switches. under submission.

Kim, H., & Feamster, N. (2013). Improving network management with software defined networking. IEEE Communications Magazine, 51(2), 114-119.

Kirichek, R., Vladyko, A., Zakharov, M., & Koucheryavy, A. (2016, January). Model networks for internet of things and SDN. In Advanced Communication Technology (ICACT), 2016 18th International Conference on (pp. 76-79). IEEE.

Kontesidou, G., & Zarifis, K. (2009). Openflow Virtual Networking: A Flow-Based Network VirtualizationArchitecture.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., ... & Shenker, S. (2010, October). Onix: A distributed control platform for large-scale production networks. In OSDI (Vol. 10, pp. 1-6).

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), 14-76.

Kuhn, D., Kim, C., & Lopuz, B. (2015). VirtualBox for Oracle. In Linux and Solaris Recipes for Oracle DBAs (pp. 325-344). Apress.

Langone, J., Key, S., & Alder, U. S. (2015). Oracle announces beta for VM Virtualbox 5.0.

Lantz, B., Heller, B., & McKeown, N. (2010, October). A network in a laptop: rapid prototyping for software-defined networks. In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (p. 19). ACM.

Lara, A., Kolasani, A., & Ramamurthy, B. (2014). Network innovation using openflow: A survey. IEEE communications surveys & tutorials, 16(1), 493-512.

Lee, Y., Villar, M. T., Artigues, A., & Beamer, L. J. (2014). Promotion of enzyme flexibility by dephosphorylation and coupling to the catalytic mechanism of a phosphohexomutase. Journal of Biological Chemistry, 289(8), 4674-4682.

Leitner, M. (2015). Přehled dostupných nástrojů pro kontrolní vrstvu softwarově definovaných sítí (Doctoral dissertation, Masarykova univerzita, Fakulta informatiky).

Lerenr, A., (2015) Predicting SD-WAN Adoption [online] available at: https://blogs.gartner.com/andrew-lerner/2015/12/15/predicting-sd-wan-adoption/ [access 26-10-2017]

Levy, S. (2012). Going with the flow: Google's secret switch to the next wave of networking. Wired, April, 17.

Lin, Y. D., Lin, P. C., Yeh, C. H., Wang, Y. C., & Lai, Y. C. (2015). An extended SDN architecture for network function virtualization with a case study on intrusion prevention. IEEE Network, 29(3), 48-53.

List of OpenFlow Software Projects [Online]. Available at: https://yuba.stanford.edu/~casado/of-sw.html.

Liyanage, M., Ylianttila, M., & Gurtov, A. (2014, June). Securing the control channel of software-defined mobile networks. In World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a (pp. 1-6). IEEE.

Lopez, V., Miguel, L., Foster, J., Silva, H., Blair, L., Marsella, J., ... & Syed, S. (2015, March). Demonstration of SDN orchestration in optical multi-vendor scenarios. In Optical Fiber Communication Conference (pp. Th2A-41). Optical Society of America.

Lopez, V., Miguel, L., Foster, J., Silva, H., Blair, L., Marsella, J., ... & Syed, S. (2015, March). Demonstration of SDN orchestration in optical multi-vendor scenarios. In Optical Fiber Communication Conference (pp. Th2A-41). Optical Society of America.

Lu, X., Wang, P., Niyato, D., Kim, D. I., & Han, Z. (2016). Wireless charging technologies: Fundamentals, standards, and network applications. IEEE Communications Surveys & Tutorials, 18(2), 1413-1452.

MacCaw, A. (2011). JavaScript Web Applications: JQuery Developers' Guide to Moving State to the Client. " O'Reilly Media, Inc.".

Marschke, D., Doyle, J., & Moyer, P. (2015). Software Defined Networking (SDN): Anatomy of OpenFlow Volume I (Vol. 1). Lulu. com.

Mian, A. N., Mamoon, A., Khan, R., & Anjum, A. (2014, December). Effects of virtualization on network and processor performance using open vswitch and xen server. In Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on (pp. 762-767). IEEE.

Montazerolghaem, A., Yaghmaee, M. H., & Leon-Garcia, A. (2017). OpenSIP: Toward Software-Defined SIP Networking. IEEE Transactions on Network and Service Management.

Munir, A., Baig, G., Irteza, S. M., Qazi, I. A., Liu, A. X., & Dogar, F. R. (2015). Friends, not foes: synthesizing existing transport strategies for data center

networks. ACM SIGCOMM Computer Communication Review, 44(4), 491-502.

Murphy, R., McLaughlin, J., Williams, D., & Kruzeniski, M. (2010). U.S. Patent No. D626,133. Washington, DC: U.S. Patent and Trademark Office.

Noormohammadpour, M., Raghavendra, C. S., Rao, S., Kandula, S., & Noormohammadpour, P. D. F. (2017). DCCast: Efficient Point to Multipoint Transfers Across Datacenters.

Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Communications Surveys & Tutorials, 16(3), 1617-1634.

ONF (2017) Software-Defined Networking (SDN) Definition [online] available at:https://www.opennetworking.org/sdn-definition/ [access 30-10-2017]

Ong, S. P., Cholia, S., Jain, A., Brafman, M., Gunter, D., Ceder, G., & Persson, K. A. (2015). The Materials Application Programming Interface (API): A simple, flexible and efficient API for materials data based on REpresentational State Transfer (REST) principles. Computational Materials Science, 97, 209-215.

Open Networking Foundation (2011) OpenFlow Switch Specifications version 1.2.0, 2011.

Open Networking Foundation (2013) OpenFlow Switch Specifications version 1.3.0, 2013.

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

Open Networking Foundation (2014) OpenFlow Switch Specifications version 1.4.0, 2014.

Opendaylight Controller (2016) [Online]. Available at: https://www.opendaylight.org/.

Park, H. K., & Choi, J. Y. (2014). U.S. Patent No. D699,386. Washington, DC: U.S. Patent and Trademark Office.

Pate, P. (2015). NFV and SDN: What's the Difference.

Pfaff, B., Pettit, J., Amidon, K., Casado, M., Koponen, T., & Shenker, S. (2009, October). Extending Networking into the Virtualization Layer. In Hotnets.

Pfaff, B., Pettit, J., Koponen, T., Jackson, E. J., Zhou, A., Rajahalme, J., ... & Amidon, K. (2015, May). The Design and Implementation of Open vSwitch. In NSDI (pp. 117-130).

Ports, D. R., Li, J., Liu, V., Sharma, N. K., & Krishnamurthy, A. (2015, March). Designing Distributed Systems Using Approximate Synchrony in Data Center Networks. In NSDI (pp. 43-57).

Prete, L. R., Schweitzer, C. M., Shinoda, A. A., & de Oliveira, R. L. S. (2014, June). Simulation in an SDN network scenario using the POX Controller. In Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on (pp. 1-6). IEEE.

Projext Foodlight (2017) «Floodlight OpenFlow controller,» [Online]. Available at: http://www.projectfloodlight.org/floodlight/.

Rao, S. K. (2014). SDN and its use-cases-NV and NFV. Network, 2, H6.

Rezaei, A. A., Mohammadifar, R., Lotfi, E., Khosravi, A., & Nahavandi, S. (2017, July). A heterogeneous defense method using fuzzy decision making. In Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on (pp. 1-8). IEEE.

Ros, F. J., & Ruiz, P. M. (2014, August). Five nines of southbound reliability in software-defined networks. In Proceedings of the third workshop on Hot topics in software defined networking (pp. 31-36). ACM.

Ryu, A. (2013). Component-based Software-defined Networking Framework.

Salisbury B., (2013) OpenFlow: Proactive vs Reactive Flows [online] available at: http://networkstatic.net/openflow-proactive-vs-reactive-flows/ [access 30-10-2017]

Sandri, M. (2015). MultiFlow: uma solução para distribuição de subfluxos MPTCP em Redes OpenFlow.

Schulz-Zander, J. (2016). Realizing software-defined wireless networks.

Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., ... & Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. IEEE Communications Magazine, 51(7), 36-43.

Shenker, S. (2012). Gentle Introduction to Software-Defined Networking. Technion lecture". YouTube.

Shin, M. K., Nam, K. H., & Kim, H. J. (2012, October). Software-defined networking (SDN): A reference architecture and open APIs. In ICT Convergence (ICTC), 2012 International Conference on (pp. 360-361). IEEE.

Shin, S., Song, Y., Lee, T., Lee, S., Chung, J., Porras, P., ... & Kang, B. B. (2014, November). Rosemary: A robust, secure, and high-performance network operating system. In Proceedings of the 2014 ACM SIGSAC conference on computer and communications security (pp. 78-89). ACM.

Simon, G. (2015). Massive Interactive Multimedia Services over the Internet (Doctoral dissertation, PhD Thesis, Apr 2015, Habilitation á diriger des recherches de l'Université Rennes 1).

Sonba, A., & Abdalkreim, H. (2014). Performance Comparison Of the state of the art Openflow Controllers.

Sonkoly, B., Gulyás, A., Németh, F., Czentye, J., Kurucz, K., Novák, B., & Vaszkun, G. (2012, October). OpenFlow virtualization framework with advanced capabilities. In Software Defined Networking (EWSDN), 2012 European Workshop on(pp. 18-23). IEEE.

Suffolk, J. (2012). Cyber Security Perspectives. Huawei paper, 19.

Tajima, M. (2016). U.S. Patent No. 9,478,213. Washington, DC: U.S. Patent and Trademark Office.

Tardo, J. J. (2014). U.S. Patent Application No. 14/265,533.

Tavakoli, A., Casado, M., Koponen, T., & Shenker, S. (2009, October). Applying NOX to the Datacenter. In HotNets.

Team, M. (2012). Mininet: An instant virtual network on your laptop (or other PC). 2015-11-15]. http://mininet. org.

Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., & Sherwood, R. (2012). On Controller Performance in Software-Defined Networks. Hot-ICE, 12, 1-6.

Touch, J., Kojo, M., Lear, E., Mankin, A., Ono, K., Stiemerling, M., & Eggert, L. (2013). Service name and transport protocol port number registry. The Internet Assigned Numbers Authority (IANA).

Truong, N. B., Lee, G. M., & Ghamri-Doudane, Y. (2015, May). Software defined networking-based vehicular adhoc network with fog computing. In Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on (pp. 1202-1207). IEEE.

Tuncer, D., Charalambides, M., Clayman, S., & Pavlou, G. (2015). Adaptive resource management and control in software defined networks. IEEE Transactions on Network and Service Management, 12(1), 18-33.

Tuncer, D., Charalambides, M., Clayman, S., & Pavlou, G. (2015, November). On the placement of management and control functionality in software defined networks. In Network and Service Management (CNSM), 2015 11th International Conference on (pp. 360-365). IEEE.

Turull, D., Hidell, M., & Sjödin, P. (2014, July). Performance evaluation of openflow controllers for network virtualization. In High Performance Switching and Routing (HPSR), 2014 IEEE 15th International Conference on (pp. 50-56). IEEE.

van der Meer, S., Grasa, E., Gastón, B., Crotty, M., Barron, J., Roig, V. A., ... & Day, J. Challenges for Managing the Disappearing Network–beyond SDN.

Van Osch, W., & Avital, M. (2010, August). From Green IT to Sustainable Innovation. In AMCIS (p. 490).

Vissicchio, S., & Cittadini, L. (2016, April). Flip the (flow) table: Fast lightweight policy-preserving sdn updates. In Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on (pp. 1-9). IEEE.

Wang, S. Y. (2014, June). Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet. In Computers and Communication (ISCC), 2014 IEEE Symposium on (pp. 1-6). IEEE.

Wang, S. Y., & Kung, H. T. (1999, March). A simple methodology for constructing extensible and high-fidelity TCP/IP network simulators. In INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE (Vol. 3, pp. 1134-1143). IEEE.

Wang, S. Y., Chou, C. L., & Lin, C. C. (2007). The design and implementation of the NCTUns network simulation engine. Simulation Modelling Practice and Theory, 15(1), 57-81.

Wang, S. Y., Chou, C. L., & Yang, C. M. (2013). EstiNet openflow network simulator and emulator. IEEE Communications Magazine, 51(9), 110-117.

Wexelblat, R. L. (Ed.). (2014). History of programming languages. Academic Press.

Xia, W., Wen, Y., Foh, C. H., Niyato, D., & Xie, H. (2015). A survey on software-defined networking. IEEE Communications Surveys & Tutorials, 17(1), 27-51.

Yazici, V., Sunay, M. O., & Ercan, A. O. (2014). Controlling a software-defined network via distributed controllers. arXiv preprint arXiv:1401.7651.

Yeganeh, S. H., Tootoonchian, A., & Ganjali, Y. (2013). On scalability of software-defined networking. IEEE Communications Magazine, 51(2), 136-141.

Zhou, Q., Wang, C. X., McLaughlin, S., & Zhou, X. (2015). Network virtualization and resource description in software-defined wireless networks. IEEE Communications Magazine, 53(11), 110-117.

Zhu, T., Wang, F., Hua, Y., Feng, D., Wan, Y., Shi, Q., & Xie, Y. (2016, June). MCTCP: Congestion-aware and robust multicast TCP in Software-Defined networks. In Quality of Service (IWQoS), 2016 IEEE/ACM 24th International Symposium on (pp. 1-10). IEEE.

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

Install Virtual Box (2008) [Online]. Available at: https://ubuntuforums.org/showthread.php?t=2247263 .

Floodlight Controller (2018) [Online]. Available at: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343544/Installation+Guide#InstallationGuide-FloodlightMasterandAbove.

Open vSwitch on Linux, FreeBSD and NetBSD (2016) [Online]. Available at: http://docs.openvswitch.org/en/latest/intro/install/general/.