ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΚΑΘΗΓΗΤΗΣ: ΚΩΝΣΤΑΝΤΙΝΟΣ ΤΣΑΓΚΑΡΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ

«ΨΗΦΙΑΚΕΣ ΕΠΙΚΟΙΝΩΝΙΕΣ ΚΑΙ ΔΙΚΤΥΑ»

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΠΟΙΟΤΗΤΑΣ ΥΠΗΡΕΣΙΩΝ ΣΕ ΔΙΚΤΥΑ ΟΡΙΖΟΜΕΝΑ ΑΠΟ ΛΟΓΙΣΜΙΚΟ

ΥΠΟ
ΠΑΣΧΑΛΗΣ ΣΙΣΚΟΣ
ΜΕΤΑΠΤΥΧΙΑΚΟΣ ΦΟΙΤΗΤΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΘΗΝΑ 15/05/2018

Περιεχόμενα

Περιεχόμενα 2
ΠΕΡΙΛΗΨΗ 4
ABSTRACT 5
ΕΙΣΑΓΩΓΗ 6
ΓΕΝΙΚΟ ΜΕΡΟΣ 7
Κεφάλαιο 1 – Software Defined Networks 7
1.1 Software Defined Networks , τα δίκτυα του μέλλοντος 7
1.2 Αρχιτεκτονική 8
1.3 Πλεονεκτήματα των Software Defined Networks 9
1.4 Ιδιότητες 10
1.5 Σενάρια χρήσης (Use cases) 11
1.6 Εικονικοποίηση δικτυακών λειτουργιών – Network Functions Virtualization 12
1.7 Ο ρόλος του μηχανικού δικτύων στην εποχή των SDN 14
1.8 Τα SDN από διαφορετικές οπτικές γωνίες 15
ΚΕΦΑΛΑΙΟ 2 – SDN CONTROLLERS 17
2.1 – Controller, ο πυρήνας του δικτύου 17
2.2 – Ιδιόκτητοι και ελεύθερου κώδικα 18
2.3 The OpenDaylight Project 20
2.4 Αρχιτεκτονική της πλατφόρμας OpenDaylight 21
2.5 Αρχιτεκτονική των OSGi bundles 22
2.6 Ανάλυση των σημαντικότερων OSGi bundles 23
2.7 Restconf protocol & API 25
2.8 Apache Karaf 25
Κεφάλαιο 3 – Openflow 27
3.1 Ορισμός 27
3.2 Ιστορικό 27
3.3 Ανάπτυξη και εταιρίες - μέλη 28
3.4 Λειτουργία του Openflow 29
ΚΕΦΑΛΑΙΟ 4 – Εξομοιωτές και Προσομοιωτές Δικτύων 31
4.1 Εισαγωγή 31
4.2 Προσομοιωτές 31
4.3 Εξομοιωτές 32

4.4 Mininet Emulator 32
4.5 Λειτουργία του Mininet 34
4.6 Πλεονεκτήματα και Μειονεκτήματα 34
4.7 Εντολές και εφαρμογές 35
ΚΕΦΑΛΑΙΟ 5 – SDN και Quality of Service 37
5.1 Ποιότητα της Υπηρεσίας 37
5.2 Παραδοσιακές Προσεγγίσεις QoS 38
5.2 SDN QoS Frameworks 39
5.2.1 Συστήματα διαφύλαξης πόρων 40
5.2.2 Συστήματα δρομολόγησης ανά ροή 40
5.2.3 Συστήματα με έμφαση στην διαχείριση ουρών και τον προγραμματισμό πακέτων 41
5.2.4 Συστήματα επιβολής πολιτικών 41
ΕΙΔΙΚΟ ΜΕΡΟΣ 43
ΚΕΦΑΛΑΙΟ 1 – Περιβάλλον Ανάπτυξης 43
1.1 Απαιτούμενα συστήματα και εγκατάσταση 43
1.2 OpenDaylight Controller 43
1.3 Mininet 46
1.4 Openflow Command Line Tools 48
1.5 Postman 49
1.6 NetBeans IDE 50
ΚΕΦΑΛΑΙΟ 2 – SDN Application 51
2.1 Παρουσίαση της εφαρμογής 51
ΚΕΦΑΛΑΙΟ 3 – Quality of Service Tests and Measurements 60
3.1 Σενάριο με ένα switch και χρήση Monitor 60
3.2 Σενάριο σε δομή δένδρου με ύψος τέσσερα 63
3.3 Αποτελέσματα μετρήσεων 67
ΣΥΜΠΕΡΑΣΜΑΤΑ 68
Βιβλιογραφία 69
Εικόνες 70

ΠΕΡΙΛΗΨΗ

Σκοπός της διπλωματικής εργασίας είναι η μελέτη τον Δικτύων Οριζόμενων από Λογισμικό, αλλά και η δημιουργία λογισμικού σε γλώσσα προγραμματισμού Java που θα αλληλεπιδρά με διαχειριστή Δικτύων Οριζόμενων από Λογισμικό.

Στην εφαρμογή που αναπτύχθηκε εφαρμόζονται τεχνικές Ποιότητας της Υπηρεσίας και υπάρχουν δυνατότητες εμφάνισης, διαχείρισης και παραμετροποίησης των δικτυακών στοιχείων μέσω γραφικού περιβάλλοντος, αλλά και εξαγωγή τους σε αρχείο excel.

Στην εργασία παρατίθενται πληροφορίες για τα δίκτυα οριζόμενα από λογισμικό, τους διαχειριστές δικτύων οριζόμενων από λογισμικό, τον διαχειριστή OpenDayLight το πρωτόκολλο επικοινωνίας Openflow, την Ποιότητα της Υπηρεσίας και τον εξομοιωτή δικτύων Mininet. Αναλύεται η δομή, η αρχιτεκτονική και η αλληλεπίδραση των παραπάνω συστημάτων. Τέλος καταγράφονται εντολές που χρησιμοποιήθηκαν στα πλαίσια της εργασίας αλλά βασικά τμήματα και λειτουργίες της εφαρμογής.

ABSTRACT

The aim of this thesis is the study of Software Defined Networks and the development of a Java application that will interact with a Software Defined Network Controller.

The application exercises Quality of Service techniques and provides the ability to view, manage and configure network elements through a Graphical User Interface as also extract them to an excel file.

In the document there is information about Software Defined Networks, SDN controllers, OpenDayLight Controller, Openflow communication protocol, Quality of Service and Mininet network emulator. The structure, architecture and interaction of the above mentioned systems and frameworks is analyzed. Finally, there is a record of commands used during this exercise and key components and functions of the application.

ΕΙΣΑΓΩΓΗ

Ο σχεδιασμός και η διαχείριση των δικτύων γίνεται όλο και πιο καινοτόμος τα τελευταία χρόνια χάρη στη διείσδυση των οριζόμενων από το λογισμικό δικτύων. Η λογική της τεχνολογίας αυτής είναι στον διαχωρισμό του επιπέδου ελέγχου (control plane) από το επίπεδο δεδομένων (data plane). Στο επίπεδο δεδομένων γίνεται η προώθηση της κίνησης ενώ στο επίπεδο ελέγχου λαμβάνονται οι αποφάσεις για όλες τις λειτουργίες και την προώθηση.

Παράλληλα είναι τέτοια η δομή των SDN όπου μία συσκευή στο επίπεδο ελέγχου μπορεί να διαχειριστεί πολλά στοιχεία στο επίπεδο δεδομένων. Κάθε δικτυακό στοιχείο έχει πίνακες με κανόνες. Κάθε κανόνας ταυτίζεται με κάποιο τμήμα κίνησης και δρα πάνω σε αυτό. Ανάλογα με τους κανόνες που έχει εγκαταστήσει ο SDN Controller στο δικτυακό switch, το switch αυτό μπορεί να λειτουργήσει ως δρομολογητής, μεταγωγός, τείχος προστασίας ή οποιαδήποτε άλλη δικτυακή συσκευή.

Η τεχνολογία αυτή κέρδισε μεγάλη δημοτικότητα μετά το 2010. Όμως στην πραγματικότητα τα SDN ήρθαν για να καλύψουν μια παλαιότερη ανάγκη, αυτή των προγραμματιζόμενων δικτύων. Σχεδόν από την αρχή των δικτύων υπολογιστών αλλά και κυρίως με την ραγδαία ανάπτυξη του internet, υπήρχαν λόγοι ώστε η δομή των δικτύων να υποστεί δραστικές αλλαγές.

Τα δίκτυα υπολογιστών έχουν δύσκολη διαχείριση και είναι ανομοιογενείς οντότητες. Αποτελούνται από διάφορα μηχανήματα όπως δρομολογητές, μεταγωγούς, εξυπηρετητές εξισορρόπησης κίνησης , εξυπηρετητές λειτουργιών ασφαλείας όπως τείχη προστασίας και άλλα. Το λογισμικό λειτουργίας αυτών των μηχανημάτων είναι σύνθετο και στις περισσότερες περιπτώσεις κλειστό και ιδιόκτητο. Κάθε συσκευή πρέπει να παραμετροποιηθεί ξεχωριστά κάνοντας χρήση διαφορετικών πρωτοκόλλων και λογισμικού. Όλες αυτές οι δυσκολίες επιβραδύνουν την καινοτομία, αυξάνουν την πολυπλοκότητα αλλά και το κόστος εγκατάστασης και συντήρησης των δικτύων. Τα SDN εμφανίστηκαν ως μια ιδέα η οποία έχει σκοπό να βελτιώσει αδυναμίες και να αντιμετωπίσει ζητήματα που υπήρχαν από την αρχή των δικτύων υπολογιστών.

Εκτός όμως από τις προαναφερθείσες ιδιαιτερότητες υπήρχαν και άλλοι εξωτερικοί παράγοντες που διατέλεσαν στην αύξηση της δημοτικότητας που έχουν πάρει τα SDN τα τελευταία χρόνια. Η εξέλιξη του Openflow protocol και του Openflow API, η διαρκής ανάγκη για μείωση του κόστους και αύξηση της επεξεργαστικής ισχύος των δικτυακών συσκευών αλλά και η ανάπτυξη των γλωσσών προγραμματισμού όπως η Java είναι παράγοντες που έπαιξαν καταλυτικό ρόλο στην αύξηση της ροπής που έχουν τα SDN στη σημερινή αγορά.

ΓΕΝΙΚΟ ΜΕΡΟΣ

Κεφάλαιο 1 – Software Defined Networks

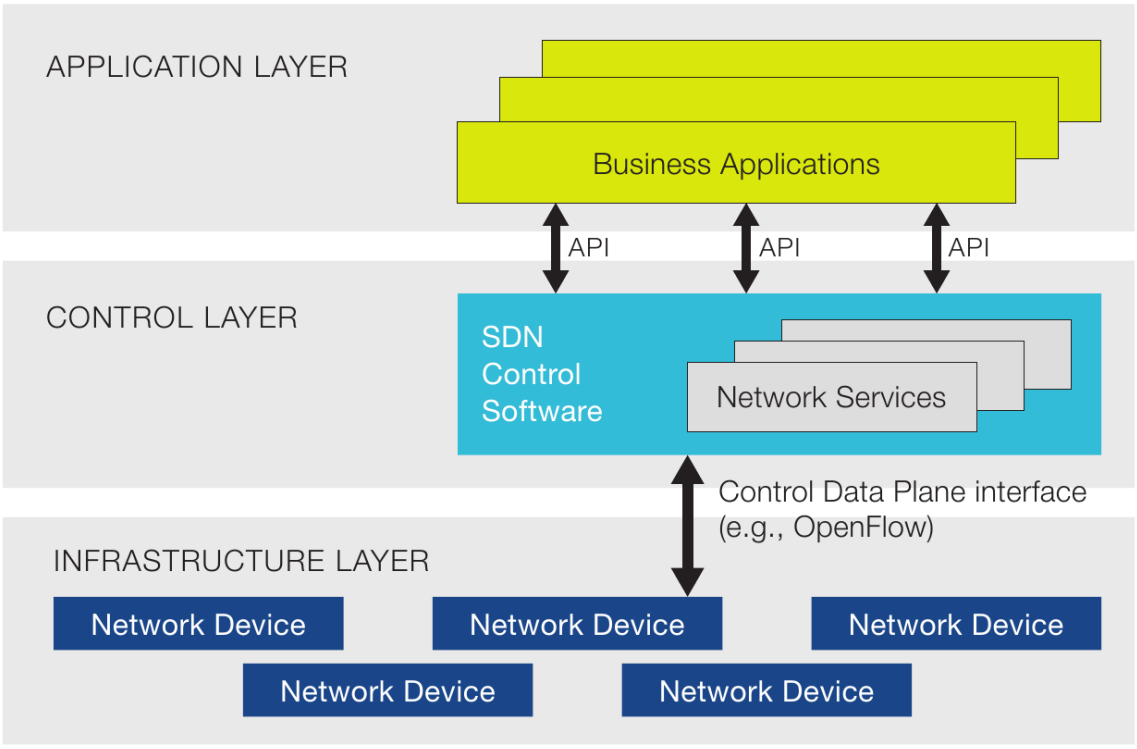
1.1 Software Defined Networks , τα δίκτυα του μέλλοντος

Τα Software Defined Networks θεωρούνται μια δραστική αλλαγή στο τομέα των επικοινωνιών. Είναι νούμερο ένα στη λίστα με τις τεχνολογίες που θα διαμορφώσουν τα δίκτυα και τα κέντρα δεδομένων του μέλλοντος. Επαναπροσδιορίζουν τη μορφή των δικτύων όπως τα γνωρίζουμε μέχρι σήμερα. Ο στόχος τους είναι η παραμετροποίηση και αλλαγή της δομής τους με χρήση λογισμικού χωρίς να απαιτείται τροποποίηση ή αναβάθμιση στην φυσική υπόσταση (hardware).

Η λογική της διαστρωμάτωσης των κλασικών μοντέλων, όπως του Open Systems Interconnection - OSI, διαφοροποιείται. Τα SDN βασίζονται στον διαχωρισμό του επιπέδου ελέγχου(control plane) και του επιπέδου δεδομένων(data plane) .

Η διαχείριση των δικτύων απαιτεί την υποστήριξη πολλών διαφορετικών λειτουργιών ταυτόχρονα, από τη δρομολόγηση μέχρι την παρακολούθηση την κίνησης, τον έλεγχο του επιπέδου πρόσβασης και την εξισορρόπηση της κίνησης. Τα SDN παρέχουν τη δυνατότητα στις εφαρμογές να έχουν διαχείριση των λειτουργιών αυτών, εγκαθιστώντας κανόνες επεξεργασίας πακέτων στα δικτυακά στοιχεία από τα οποία διέρχεται η κίνηση. Η διαχείριση του δικτύου μπορεί να απλοποιηθεί σε μεγάλο βαθμό, προσφέροντας στους μηχανικούς – προγραμματιστές καθολική εικόνα του δικτύου και άμεσο έλεγχο στη δομή του μέσω ενός ,σε λογικό επίπεδο, κεντρικού εργαλείου, τον διαχειριστή (controller).

1.2 Αρχιτεκτονική



Στην αρχιτεκτονική των SDN καθορίζεται πως ένα δικτυακό σύστημα μπορεί να σχεδιαστεί βασισμένο σε τεχνολογίες που στηρίζονται στο λογισμικό. Στο μοντέλο αυτό χρησιμοποιείται υλικό γενικού σκοπού και μειωμένου κόστους που μπορεί εύκολα να αντικατασταθεί με κάποιο παρόμοιο που να παρέχει τις ίδιες λειτουργίες. Η λογική στηρίζεται στον διαχωρισμό του επιπέδου ελέγχου με το επίπεδο δεδομένων.

Στα παραδοσιακά δίκτυα το επίπεδο ελέγχου και το επίπεδο δεδομένων βρίσκονται μαζί και εφαρμόζονται από το ίδιο ιδιόκτητο λογισμικό που παρέχεται από έναν ή περισσότερους προμηθευτές – εταιρείες. Η δομή των SDN μπορεί να διαχωριστεί σε τρία επίπεδα.

Στην κορυφή της στοίβας στο επίπεδο των εφαρμογών βρίσκονται προγράμματα τα οποία επικοινωνούν με τον διαχειριστή για να ζητήσουν πόρους και δεδομένα ή να μεταφέρουν αιτήματα για συμπεριφορές στο δίκτυο. Οι εφαρμογές αυτές μπορούν να δημιουργήσουν μια εικόνα του δικτύου συλλέγοντας δεδομένα από τον controller ώστε να τα συμπεριλάβουν για λήψη αποφάσεων. Οι εφαρμογές αυτές μπορούν να σχετίζονται με την ανάλυση δεδομένων, με τη διαχείριση του δικτύου, ακόμα και με μηχανισμούς ασφάλειας.

Στο ενδιάμεσο επίπεδο βρίσκεται ο SDN Controller. Είναι η λογική οντότητα που δέχεται εντολές και οδηγίες από το επίπεδο των εφαρμογών και τις εφαρμόζει στα κατώτερα στρώματα όπου βρίσκονται τα δικτυακά στοιχεία. Ακόμα παίρνει πληροφορίες για το δίκτυο από τις φυσικές συσκευές και τις μεταφέρει στις

εφαρμογές. Έχει μια συνολική αφηρημένη εικόνα του δικτύου και οι αποφάσεις που λαμβάνει είναι πιο «έξυπνες» καθώς έχει περισσότερα δεδομένα για την εξαγωγή τους. Η επικοινωνία γίνεται με το επίπεδο των εφαρμογών μέσω Application Programming Interfaces – APIs τα οποία αποκαλούνται northbound. Στην κάτω πλευρά του δικτύου επικοινωνεί με τις φυσικές συσκευές μέσω APIs που αποκαλούνται southbound. Το πιο γνωστό και διαδεδομένο από αυτά είναι το OpenFlow API που βασίζεται στο Openflow πρότυπο.

Στο τελευταίο στρώμα, στο επίπεδο δεδομένων, βρίσκονται οι δικτυακές φυσικές συσκευές που αναλαμβάνουν την επεξεργασία και την προώθηση των πακέτων. Με τα southbound APIs επικοινωνούν με τον controller μέσω κάποιου πρωτοκόλλου, όπως το OpenFlow, το οποίο πρέπει να υποστηρίζουν.

Ένα από τα μεγαλύτερα πλεονεκτήματα της αρχιτεκτονικής αυτής είναι ότι παρέχεται η δυνατότητα στις εφαρμογές να έχουν καθολική εικόνα του δικτύου. Η ιδιότητα αυτή κάνει τα δίκτυα εξυπνότερα καθώς μπορούν να αναλύσουν τη λειτουργία τους και να μεταδώσουν στις εφαρμογές δεδομένα σχετικά με τη δικτυακή δραστηριότητα σε πραγματικό χρόνο.

1.3 Πλεονεκτήματα των Software Defined Networks

Ο κλάδος των τηλεπικοινωνιών και της τεχνολογίας παρουσιάζει τα SDN ως εξαιρετική λύση στην αυξανόμενη πολυπλοκότητα και το κόστος διαχείρισης και λειτουργίας των δικτύων, που παραδοσιακά συνδέεται με την εφαρμογή νέων υπηρεσιών και τεχνολογιών.

Η διαφοροποίηση του επιπέδου ελέγχου με το επίπεδο δεδομένων παρέχει κεντρική , προγραμματιζόμενη διαχείριση των πόρων. Παράλληλα τα δίκτυα γίνονται πιο ευέλικτα και η ανάπτυξη τους είναι πιο εύκολη. Επίσης η ταχύτητα μεταφοράς δεδομένων αλλά και ο χρόνος για αποφάσεις δρομολόγησης μειώνεται σημαντικά. Δραστική είναι και η συμβολή στην ασφάλεια και τη διαχείριση ενέργειας των δικτύων.

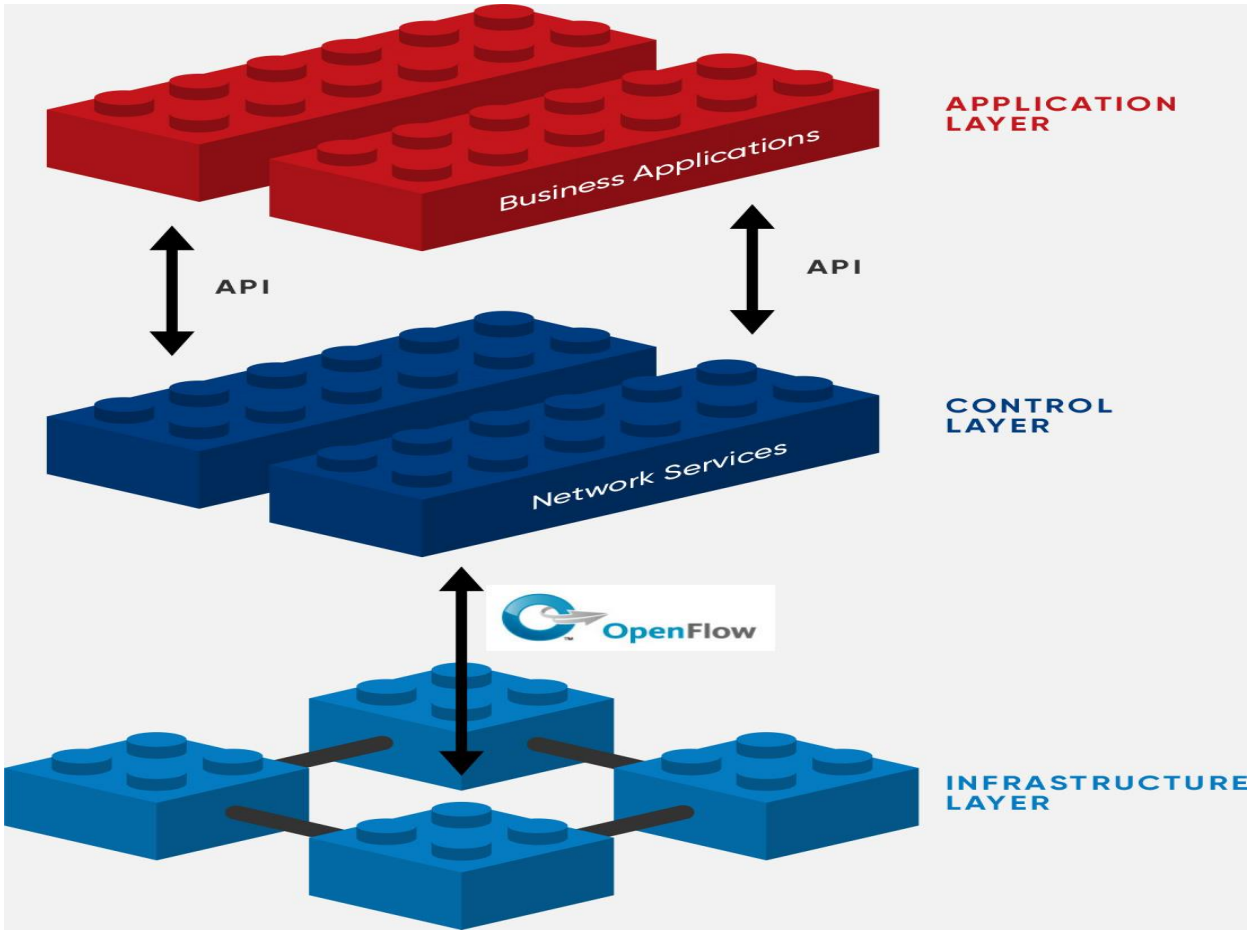
Η λογική είναι παρόμοια με την υποδομή που χρησιμοποιείται για εικονικούς εξυπηρετητές (virtual servers) και μέσα αποθήκευσης στα σύγχρονα κέντρα δεδομένων (data centers) .

Οι μηχανικοί/διαχειριστές δικτύων θα έχουν τη δυνατότητα να προσαρμόζονται άμεσα στις μονίμως τροποποιούμενες απαιτήσεις της αγοράς. Η κίνηση θα μπορεί να διαμορφώνεται από μια κεντρική κονσόλα διαχείρισης χωρίς να πρέπει να παραμετροποιηθεί ανεξάρτητα κάθε δικτυακό στοιχείο (switches, routers, etc.). Επίσης οι υπηρεσίες που θα παρέχονται θα είναι ανεξάρτητες με τις δυνατότητες του υλικού και των συσκευών που συνδέονται στο δίκτυο.

Το μεγαλύτερο ίσως όλων το πλεονεκτημάτων, είναι ότι ξεφεύγουμε τη λογική πως για κάθε λειτουργία χρειαζόμαστε διαφορετικό τύπο εξοπλισμού. Σε ένα δίκτυο υπολογιστών θα συναντήσουμε switches, multilayer switches, routers, firewalls, security/authentication servers, proxy servers, data servers, Nat translators και πολλές άλλες συσκευές. Ανάλογα με τη χρήση του εκάστοτε μηχανήματος

χρειαζόμαστε διαφορετικό υλικό αλλά και λογισμικό. Στο μοντέλο των SDN θα υπάρχουν switches γενικού σκοπού(general purpose) όπου ανάλογα με του κανόνες που θα τρέχουν θα μπορούν να λειτουργούν είτε ως δρομολογητές είτε ως π.χ. τείχος προστασίας. Ακόμα θα υπάρχει δυνατότητα μετατροπής από τη μία κατάσταση στην άλλη με απλή τροποποίηση της παραμετροποίησης.

1.4 Ιδιότητες



Η αρχιτεκτονική των SDN είναι :

- **Άμεσα προγραμματιζόμενη** διότι αποσπάται από συναρτήσεις προώθησης.
- **Ευέλικτη** καθώς ο διαχωρισμός του επιπέδου ελέγχου με το επίπεδο δεδομένων παρέχει τη δυνατότητα στους διαχειριστές δικτύου, να διαμορφώνουν δυναμικά την κίνηση σύμφωνα με τις μονίμως μεταβαλλόμενες ανάγκες.
- **Κεντρικώς διαχειριζόμενη** αφού η «ευφυΐα» του δικτύου είναι συσσωρευμένη στον SDN controller ο οποίος κατέχει συνολική εικόνα του δικτύου και εμφανίζεται στις εφαρμογές και τις μηχανές πολιτικής , σε λογικό επίπεδο, ως μοναδικός μεταγωγός (switch).
- **Παραμετροποιούμενη μέσω κώδικα.** Στα SDN οι διαχειριστές έχουν τη δυνατότητα να ελέγχουν, παραμετροποιούν, εφαρμόζουν μηχανισμούς

ασφαλείας και να βελτιστοποιούν τους δικτυακούς πόρους εύκολα μέσω δυναμικών, SDN προγραμμάτων τα οποία μπορούν να τροποποιηθούν καθώς δεν βασίζονται σε ιδιόκτητο λογισμικό.

- **Βασισμένη σε ανοιχτά πρότυπα και ουδέτερη σε προμηθευτές.** Ο σχεδιασμός του δικτύου απλοποιείται διότι οι λειτουργίες και οι οδηγίες παρέχονται από τον SDN controller μέσω ανοιχτών προτύπων και δε βασίζονται στις ιδιαιτερότητες των μηχανημάτων και των πρωτοκόλλων της εκάστοτε εταιρίας.

1.5 Σενάρια χρήσης (Use cases)

Ένα από τα πιο συχνά σενάρια χρήσης(use case) των SDN είναι αυτό του ελέγχου πρόσβασης σε ένα δίκτυο. Είτε είναι σε επίπεδο χρήστη είτε σε επίπεδο συσκευής υπάρχει δυνατότητα μέσω του controller να οριστούν από δικαιώματα μέχρι και όρια επιπέδου πρόσβασης και ποιότητα της υπηρεσίας. Έτσι μία συσκευή σε οποιαδήποτε τμήμα του δικτύου και να συνδεθεί θα ακολουθεί τους καθολικούς κανόνες που έχουν οριστεί από τον controller.

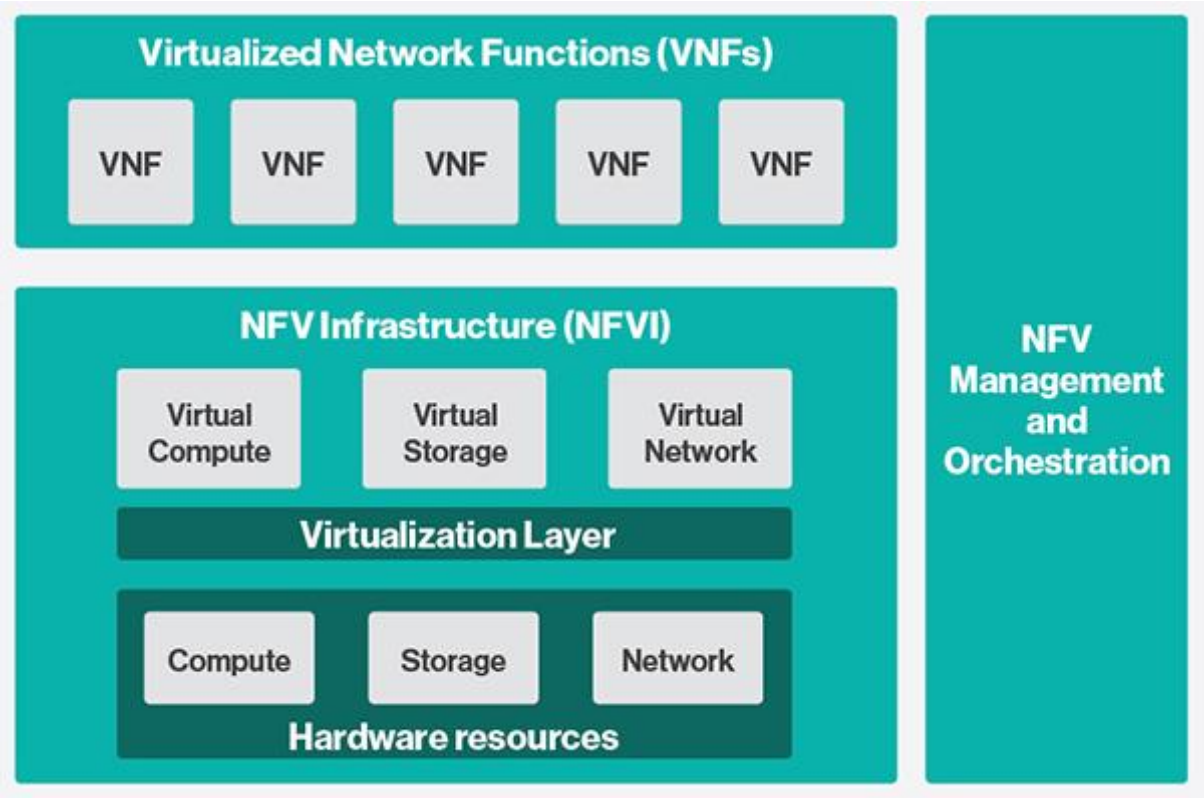
Η εικονική δημιουργία των λειτουργιών δικτύου(NFV – Networks Function Virtualization) είναι μία από σημαντικότερες εφαρμογές των SDN. Η δημιουργία ενός εικονικού χωρίς όρια δικτύου σε συνέχεια του φυσικού δικτύου επιτρέπει την εικονική τμηματοποίηση του. Ο διαχωρισμός αυτός του πραγματικού δικτύου σε πολλά μικρότερα εικονικά ανάλογα με τη λειτουργία, τις υπηρεσίες και τους μηχανισμούς ασφαλείας που εφαρμόζονται στο καθένα παρέχει μεγάλη ευελιξία στη σχεδίαση και τη διαχείριση. Ακόμα και αν τα εικονικά αυτά δίκτυα έχουν υλική υπόσταση σε διαφορετικά racks ή και κέντρα δεδομένων μέσω του NFV εμφανίζονται ως ένα δίκτυο.

Η εικονική δημιουργία στο δίκτυο του άκρου πελάτη είναι άλλο ένα σημείο όπου βρίσκουν εφαρμογή τα SDN. Η εφαρμογή μπορεί να γίνει μέσω δημιουργίας εικονικής πλατφόρμας στις εγκαταστάσεις του πελάτη είτε απορροφώντας τις λειτουργίες στον πυρήνα μέσω κεντρικής πλατφόρμας διαχείρισης που μπορεί να έχει φυσική υπόσταση σε τοπικό σημείο παρουσίας ή σε κέντρο δεδομένων.

Εφαρμογή με μεγάλο αντίκτυπο στην αγορά είναι αυτή της δυναμικής δημιουργίας συνδέσμων. Δυναμικά μπορούν να δημιουργηθούν δρόμοι μεταξύ κέντρων δεδομένων, εφαρμογών και εγκαταστάσεων επιχειρήσεων και παράλληλα να αποδοθεί συγκεκριμένη χωρητικότητα αλλά και να εφαρμοστεί ποιότητα της υπηρεσίας στους συνδέσμους αυτούς.

Τέλος τα SDN καλύπτουν ανάγκες των κέντρων δεδομένων καθώς παρέχουν τη δυνατότητα καλύτερης αξιοποίησης των πόρων. Λαμβάνοντας υπό όψιν το φορτίο μεταφοράς δεδομένων και την απαιτούμενη επεξεργαστική ισχύ μπορούμε να αυξήσουμε την αποδοτικότητα των εφαρμογών με σωστή παραμετροποίηση και υλοποίηση των δικτύων.

1.6 Εικονικοποίηση δικτυακών λειτουργιών – Network Functions Virtualization



Η εικονικοποίηση των δικτυακών λειτουργιών (NFV) , γνωστή και ως εικονική λειτουργία δικτύου(Virtual Network Function - VNF) παρέχει ένα νέο τρόπο σχεδιασμού και υλοποίησης των δικτυακών λειτουργιών. Στο κλασικό μοντέλο δικτύων που έχει εδραιωθεί σήμερα υπηρεσίες όπως η μετάφραση δικτυακών διευθύνσεων (Network Address Translation - NAT), τα τείχη προστασίας, μηχανισμοί ασφάλειας , το σύστημα ονόματος τομέα (Domain Name System - DNS) υλοποιούνται από ιδιόκτητο υλικό , το οποίο χρησιμοποιεί και αντίστοιχο λογισμικό.

Έχει σχεδιαστεί με τέτοιο τρόπο ώστε να δημιουργήσει όλα τα απαραίτητα δικτυακά τμήματα που απαιτούνται για να υπάρχει μια πλήρως εικονική και λειτουργική υποδομή, που θα συμπεριλαμβάνει εικονικούς εξυπηρετητές, μέσα αποθήκευσης, ακόμα και άλλα δίκτυα. Εκμεταλλεύεται κλασικές τεχνολογίες εικονικοποίησης υψηλής έντασης και δυνατοτήτων, το υλικό των μεταγωγών(switches) και των μέσων αποθήκευσης ώστε να δημιουργήσει εικονικά τις λειτουργίες δικτύου. Έχει πεδίο εφαρμογής σε οποιοδήποτε επίπεδο δεδομένων ή επίπεδο ελέγχου σε ενσύρματες αλλά και ασύρματες δικτυακές υποδομές.

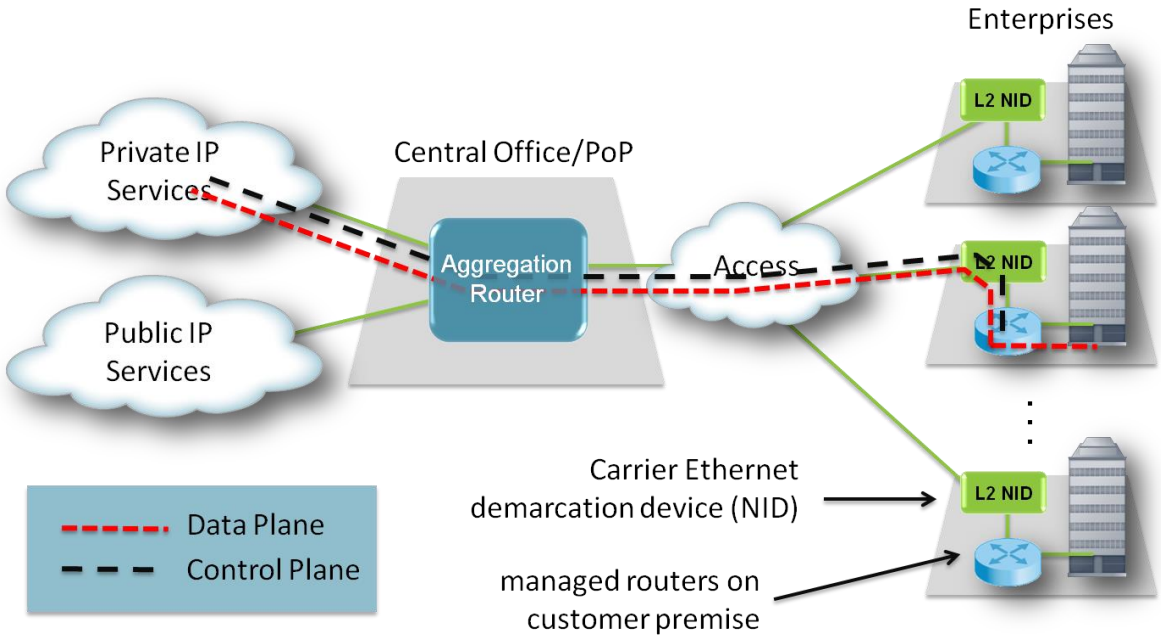
Το Ευρωπαϊκό Ινστιτούτο Τηλεπικοινωνιακών Προτύπων(European Telecommunications Standards Institute) έχει δημιουργήσει την ETSI Industry Specification Group for Network Functions Virtualization(ETSI ISG NFV). Η ομάδα αυτή είναι αρμόδια να αναπτύσσει τις απαιτήσεις και την αρχιτεκτονική για την εικονικοποίηση των δικτυακών λειτουργιών.

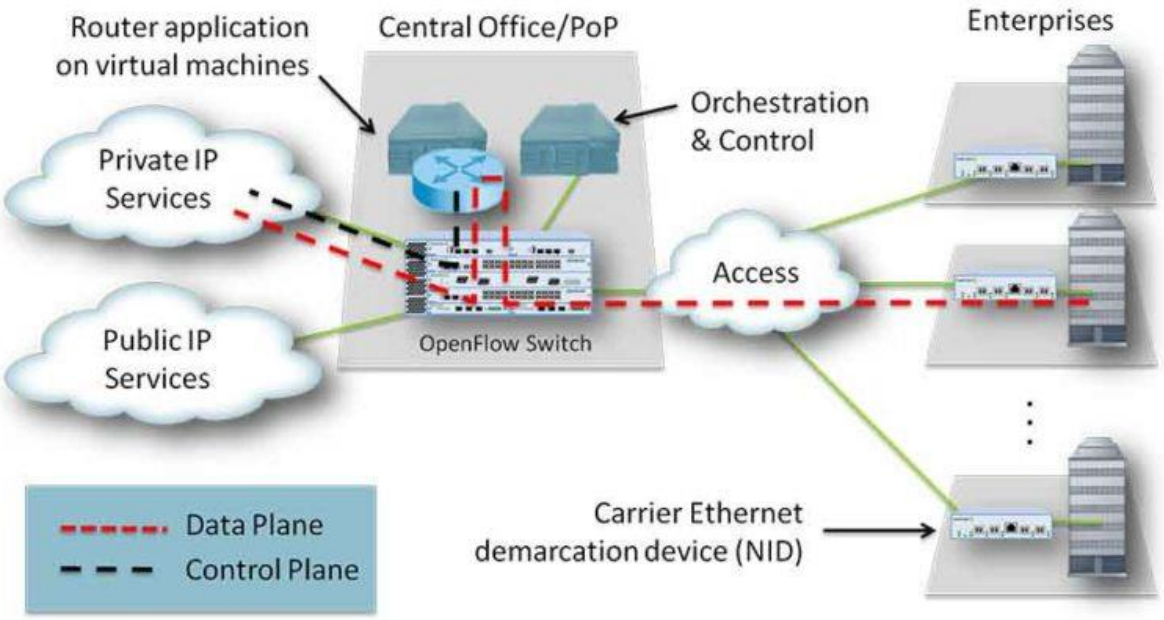
Στο μοντέλο των NFV οι υπηρεσίες δημιουργούνται πάνω σε εικονικά μηχανήματα που τρέχουν σε φθηνό hardware που μπορεί να αντικατασταθεί εύκολα. Οι ξεχωριστές δικτυακές λειτουργίες τρέχουν σε VNF και με μεγάλη ευκολία ένας διαχειριστής μπορεί να προσθέτει και να αφαιρεί εικονικές λειτουργίες ανάλογα με τις ανάγκες. Αν μια λειτουργία

απαιτεί μεγαλύτερο εύρος ζώνης ή επεξεργαστική ισχύ μπορούμε απλά να προσθέσουμε ένα επιπλέον εικονικό μηχάνημα. Αν ένας πελάτης ζητήσει μια καινούργια υπηρεσία , είναι εύκολο να δημιουργηθεί ένα καινούργιο εικονικό μηχάνημα χωρίς την παραμικρή αλλαγή στην υποδομή και το υλικό.

Η εικονικοποίηση των δικτύων με τη μεγάλη ευελιξία που παρέχει έχει στόχο να μειώσει το κεφάλαιο εγκατάστασης(Capital Expenditure = CapEx) καθώς το υλικό που πρέπει να αγοραστεί είναι φθηνότερο και η χρησιμοποίηση των πόρων είναι αποδοτική. Ακόμα μειώνονται και τα λειτουργικά έξοδα(Operational Expenditure - OpEx) καθώς ο χώρος, το ρεύμα και οι απαιτήσεις για ψύξη του εξοπλισμού των δικτυακών υπηρεσιών και λειτουργιών ελαττώνονται. Ακόμα μειώνεται ο χρόνος που απαιτείται ώστε να βγει μια υπηρεσία στην αγορά. Η μείωση αυτή μπορεί να αποφέρει μεγάλα έσοδα καθώς ο χρόνος που θα εκδοθεί εμπορικά μια υπηρεσία είναι κρίσιμος και μπορεί να επιστρέψει το κόστος της επένδυσης σε μικρότερο χρονικό διάστημα. Έτσι μειώνεται ακόμα περισσότερο το ρίσκο που σχετίζεται με τη δημιουργία και τη δοκιμή νέων υπηρεσιών. Τέλος υπάρχει μεγάλη ευκολία στην αύξηση και μείωση των αναγκών μιας επιχείρησης ή υπηρεσίας εφαρμόζοντας εύκολα και γρήγορα τη χρήση λογισμικού σε φθηνό υλικό.

Μεγάλη σύγχυση υπάρχει ανάμεσα στους όρους SDN και NFV. Πολλές φορές εξετάζονται ως μία οντότητα ενώ δεν είναι. Υπάρχει η δυνατότητα ξεχωριστής υλοποίησης SDN και NFV. Οι τεχνολογίες αυτές συμπληρώνουν η μία την άλλη και έχουν μεγαλύτερη αποδοτικότητα όταν συνεργάζονται. Έχουν παρόμοια λογική και στοχεύουν στη ευελιξία των δικτύων. Τα SDN υλοποιούν τον διαχωρισμό του επιπέδου δεδομένων με το επίπεδο ελέγχου ώστε να ελέγξουν και να κατευθύνουν την κίνηση. Η NFV αποσκοπεί στον διαχωρισμό των υπηρεσιών και τη μεταφορά τους από το υλικό σε εικονικά μηχανήματα.





1.7 Ο ρόλος του μηχανικού δικτύων στην εποχή των SDN

Πριν την εφαρμογή καινούργιων τεχνολογιών, στον τομέα των δικτύων και γενικότερα, υπάρχει μεγάλη συζήτηση μεταξύ των οργανισμών και εταιριών όσον αφορά την επένδυση της εφαρμογής αλλά και μεταξύ των επαγγελματιών που τους ανατίθεται το έργο της υλοποίησης.

Τα SDN δε θα μπορούσαν να αποτελέσουν εξαίρεση και ερωτήματα ως προς τις αλλαγές στην καθημερινότητα των μηχανικών δικτύων γεννιούνται. Η ιδέα των SDN υπόσχεται, μεταξύ άλλων, μείωση στο συνολικό κόστος, ευελιξία, ταχύτητα και αποδοτικότητα αλλά ενδεχομένως και απορρόφηση κάποιων καθημερινών λειτουργιών και κομμάτι της εργασίας των μηχανικών δικτύων. Παράλληλα επιβάλει και τη μάθηση αλλά και εξέλιξη νέων δεξιοτήτων , όπως και την αλλαγή στην εικόνα που έχουμε για τα δίκτυα μέχρι σήμερα. Όσο πιο γρήγορα εξαπλώνεται η εφαρμογή των SDN, τόσο πιο μεγάλος ρυθμός προσαρμογής με τις νέες τάσεις πρέπει να διατηρηθεί.

Στην «παραδοσιακή» διαχείριση δικτύων, οι μηχανικοί εφαρμόζουν τροποποίηση σε εξοπλισμό μέσω CLI(command line interface). Μικρές αλλαγές , όπως δημιουργία και τροποποίηση σε λίστες πρόσβασης (access lists), απαιτούν τη χειροκίνητη πληκτρολόγηση εντολών, έργο συνήθως χρονοβόρο αλλά και κουραστικό σε σύνθετα περιβάλλοντα. Τα SDN σχεδόν καταργούν τη διαδικασία αυτή, δίνοντας τη δυνατότητα οι αλλαγές αυτές να γίνουν μέσω κεντρικής διαχείρισης και σε γραφικό περιβάλλον. Τα αποτελέσματα που υπόσχονται είναι καθολική και ολοκληρωμένη εικόνα του δικτύου, διαθέσιμη χωρίς την απαίτηση εντολών σε κάθε ξεχωριστό μηχάνημα, αλλά και να κάνουν την καθημερινότητα του μηχανικού δικτύου ευκολότερη. Παράλληλα εισάγουν τα οφέλη των εφαρμογών και του προγραμματισμού στον κόσμο των δικτύων. Μέσω του αυτοματισμού και της

επαναχρησιμοποίησης του κώδικα, υπάρχει κέρδος σε χρόνο αποφεύγοντας επαναλαμβανόμενα χειροκίνητες διαδικασίες αλλά και εξάλειψη του παράγοντα ανθρώπινου λάθους.

Η επιτυχής αφομοίωση και εφαρμογή των SDN απαιτεί από την πλευρά των μηχανικών δικτύων την ανάπτυξη των ικανοτήτων τους στον προγραμματισμό και το scripting, σε συνδυασμό με τη χρήση του CLI. Αρχικά η ιδέα αυτή μπορεί να προκαλέσει μεγάλη σύγχυση καθώς η δουλειά του προγραμματιστή είναι σε μεγάλο βαθμό διαχωρισμένη από αυτή του μηχανικού δικτύων, με τα σημερινά δεδομένα. Όμως κάθε εργασία, ειδικά αυτές που σχετίζονται με τον τομέα της τεχνολογίας, απαιτεί την εξέλιξη και την ανάπτυξη νέων δεξιοτήτων.

Επίσης στα κλασικά μοντέλα είναι έτσι δομημένη η διαχείριση του δικτύου, ώστε να χρησιμοποιείται σύστημα αιτημάτων(tickets) και ενεργειών. Αυτού του είδους η δομή, προκαλεί μία , σε αρκετές περιπτώσεις, ασταμάτητη ροή της οποίας ο όγκος πρέπει να διαχειριστεί άμεσα. Το αποτέλεσμα είναι οι μηχανικοί δικτύων να αφιερώνουν τον περισσότερο χρόνο τους εξετάζοντας προβλήματα και αιτήματα στο δίκτυο, χωρίς να παρέχεται η ευκαιρία να αξιοποιήσουν τις δυνατότητες τους σε άλλους τομείς. Τα SDN υπόσχονται να μειώσουν αυτό τον όγκο εργασίας και να απελευθερώσουν χρόνο στους μηχανικούς ενθαρρύνοντας την καινοτομία, αλλά και παρέχοντας τους τη ευκαιρία να δημιουργήσουν νέες υπηρεσίες και να διεξαγάγουν έρευνα.

1.8 Τα SDN από διαφορετικές οπτικές γωνίες

Οι δυνατότητες και η έρευνα που σχετίζεται με SDN έχουν αδιαμφισβήτητα ανοδική τάση. Ακόμα υπάρχουν εξωγενείς παράγοντες ,όπως η ραγδαία αύξηση του αριθμού των κινητών συσκευών και η ανάπτυξη νέων τεχνολογιών και του NFV, που πιέζουν στη εδραίωση των SDN στην αγορά. Από τεχνολογικής άποψης η μετάβαση αυτή στη νέα γενιά δικτύων, μονάχα ωφέλιμη μπορεί να χαρακτηριστεί. Όμως εξετάζοντας τα SDN από διαφορετικές οπτικές γωνίες, παρατηρούνται εμπόδια τα οποία πρέπει να αντιμετωπιστούν ώστε η εφαρμογή τους να είναι καθολική.

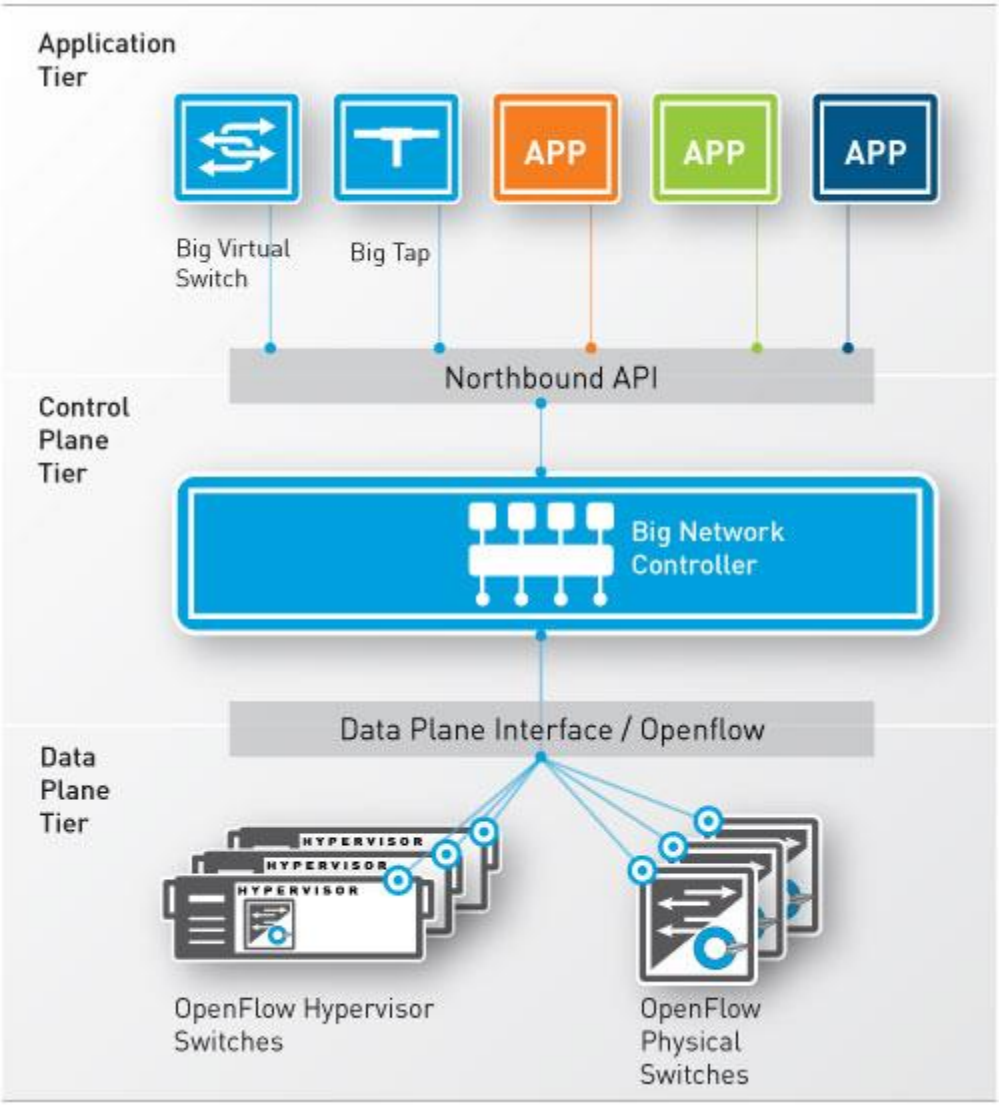
Όσον αφορά την επιχειρηματική οπτική, τα SDN αντιμετωπίζονται ως μια νέα , πολλά υποσχόμενη, τεχνολογία. Η δυνατότητα να μείνεις κοντά στις τεχνολογικές εξελίξεις παράλληλα με τη μείωση των λειτουργικών εξόδων, είναι δελεαστική. Ταυτόχρονα μειώνεται και το απαιτούμενο κεφάλαιο για εγκατάσταση δικτύων. Όμως η αλλαγή αυτή δεν έρχεται χωρίς ρίσκο καθώς οι εταιρίες οι οποίες έχουν υλοποιήσει υπηρεσίες και έχουν εφαρμόσει SDN τεχνικές σε δίκτυα παραγωγής είναι συγκριτικά λίγες.

Από την πλευρά των παρόχων αφενός υπάρχει ενθουσιασμός καθώς παρατηρούν τη δικτυακή κίνηση να αυξάνεται εκθετικά. Οι εφαρμογές γίνονται όλο και πιο απαιτητικές σε δικτυακούς πόρους. Η έξυπνη και αποδοτική εκμετάλλευση τους σε συνδυασμό με τη μείωση του κόστους εγκατάστασης και διαχείρισης, είναι παράγοντες που επηρεάζουν θετικά την εικόνα των παρόχων για τα SDN. Όμως η

δυναμική αξιοποίηση των πόρων και ο διαχωρισμός με προτεραιότητα διαφορετικών τύπων κίνησης, στο επίπεδο εφαρμογών είναι μια μεγάλη πρόκληση.

Από τη σκοπιά του πελάτη, υπάρχει μεγάλη σκεπτικότητα όσον αφορά τα SDN. Η κατάσταση αυτή δημιουργεί πίεση ώστε να βρεθούν λύσεις για να γίνουν τα δίκτυα πιο έξυπνα και να έχουν μεγαλύτερη γνώση των εφαρμογών. Τα SDN θέλουν να φέρουν τα δίκτυα πιο κοντά στις εφαρμογές κάνοντας τα προγραμματιζόμενα, ευέλικτα, δυναμικά και με τμηματική δομή. Όμως το κόστος υιοθέτησης είναι μεγάλο καθώς οι περισσότεροι δεν έχουν εγκαταστήσει αντίστοιχες λύσεις στην υποδομή τους. Υπάρχει πολύ μεγαλύτερη σιγουριά για τις παραδοσιακές λύσεις στις οποίες υπάρχει εμπειρία αλλά και δοκιμές δεκαετιών. Οι εφαρμογή τους σε δίκτυα παραγωγής, η διαδικασία μετάβασης, η απαίτηση ικανοτήτων προγραμματισμού και η μειωμένη υποστήριξη σε σχέση με τα μηχανήματα προμηθευτή, είναι μερικά από τα ζητήματα που προκαλούν προβληματισμό. Ως αποτέλεσμα η αβεβαιότητα αυτή για τη νέα γενιά δικτύων καθυστερεί την αποδοχή τους.

ΚΕΦΑΛΑΙΟ 2 – SDN CONTROLLERS



2.1 – Controller, ο πυρήνας του δικτύου

Στην αρχιτεκτονική των SDN ο διαχειριστής (controller) , που συχνά αποκαλείται και πλατφόρμα διαχειριστή(sdn controller platform) είναι ο «εγκέφαλος» του δικτύου. Αποτελεί την εφαρμογή που έχει καθολική εικόνα των δικτυακών στοιχείων και ως κεντρικό σημείο διαχείρισης λαμβάνει αποφάσεις για τη κίνηση και τις λειτουργίες του δικτύου. Στο κλασικό μοντέλο ένας μεταγωγός ή δρομολογητής μπορεί να πάρει αποφάσεις δρομολόγησης βάση κάποιων πρωτοκόλλων. Όμως οι αποφάσεις αυτές λαμβάνονται σύμφωνα με τα δεδομένα που έχει μία συσκευή αγνοώντας συμβάντα στο υπόλοιπο δίκτυο, όπως συμφορήσεις, πεσμένους συνδέσμους, λάθη, κτλ. Σε αντίθεση ο SDN Controller έχει πλήρη γνώση της κατάστασης όλης της τοπολογίας. Έτσι οι αποφάσεις που λαμβάνει είναι πιο σύνθετες καθώς η εξαγωγή συμπερασμάτων βασίζεται σε περισσότερα δεδομένα.

Ακόμα ελέγχει τις ροές στα switches επικοινωνώντας μαζί τους μέσω χαμηλού επιπέδου διεπαφών προγραμματισμού εφαρμογών(Application Programming

Interface - API) τα οποία αποκαλούνται Southbound APIs. Παράλληλα ελέγχει και τις υψηλού επιπέδου εφαρμογές μέσω των Northbound APIs με σκοπό να κάνει το δίκτυο «εξυπνότερο». Είναι το ενδιάμεσο στοιχείο ενώνει τις εφαρμογές τα κατώτερα τμήματα του δικτύου.

Ένας controller πρακτικά κατέχει ένα σύνολο ανεξάρτητων οντοτήτων όπου κάθε μία οντότητα εφαρμόζει και μία λειτουργία. Έτσι χρησιμοποιεί αυτές τις οντότητες στα δικτυακά στοιχεία που ελέγχει για να επιτύχει το επιθυμητό αποτέλεσμα. Παραδείγματα αυτών των λειτουργιών είναι η καταγραφή των συσκευών που συνδέονται στον controller, καταγραφή των ιδιοτήτων των συσκευών και συλλογή στατιστικών του δικτύου.

Μεγάλος ανταγωνισμός υπάρχει μεταξύ των εταιριών που επιθυμούν να υλοποιήσουν το δικό τους SDN Controller για το δικό τους υλικό. Επίσης υπάρχουν διαχειριστές ανοιχτού κώδικα οι οποίοι έχουν σχεδιαστεί για κάθε τύπο εξοπλισμού.

Ο πρώτος SDN controller που δημιουργήθηκε, παράλληλα με το Openflow, ήταν ο NOX. Σχεδιάστηκε από την εταιρεία Nicira Networks και το έτος 2008 δόθηκε δωρεάν στην κοινότητα των SDN και έγινε η βάση για αρκετούς από τους μετέπειτα διαχειριστές.

Έκτοτε έχουν δημιουργηθεί διάφοροι controllers από σχεδόν όλες τις μεγάλες εταιρίες δικτύων με αυτούς του ανοιχτού κώδικα να φαίνεται πως κερδίζουν την αγορά. Διαχειριστές όπως ο OpenDaylight, ο OpenContrail, ο Floodlight, ο Ryu και ο FlowVisor είναι από τους πιο διαδεδομένους. Έχουν προσελκύσει το ενδιαφέρον μεγάλων εταιριών και βασίζουν μεγάλο κομμάτι της ανάπτυξης αλλά και της υποστήριξης σε αυτές.

2.2 – Ιδιόκτητοι και ελεύθερου κώδικα

Μία από τις τάσεις που σχετίζονται με τα SDN είναι αυτή που αφορά προϊόντα με ιδιόκτητο λογισμικό. Αυτού του είδους η προσέγγιση δίνει έμφαση στον προγραμματισμό αλλά βάζει περιορισμούς στο κομμάτι της ελευθερίας καθώς εγκαθιστά ιδιόκτητα κομμάτια στην προγραμματιζόμενη υποδομή. Οι πελάτες έχουν εκδηλώσει ενδιαφέρον στο να γίνουν τα δίκτυα υπολογιστών πιο προγραμματιζόμενα ενώ η αγορά έχει αποφασίσει για την τοποθέτηση του OpenFlow και των SDN στο σύστημα. Παρότι όμως το OpenFlow και τα SDN είναι σημαντικές εξελίξεις, αυτό που πραγματικά ενδιαφέρει τις εταιρίες – πελάτες είναι να εξελιχθεί η εξυπνάδα των δικτύων ώστε να μπορούν να προγραμματίσουν τα δίκτυα γρήγορα και σύμφωνα με τις ανάγκες τους. Αυτό πρακτικά σημαίνει ότι αντί να εμπλέκονται σε παραμετροποίηση πρωτοκόλλων όπως το Border Gateway Protocol(BGP) και σε εγκαταστάσεις εικονικών τοπικών δικτύων (VLAN), να μπορούν να περάσουν στο δίκτυο την ανάγκη για σύνδεση μεταξύ δύο σημείων σύμφωνα με κάποια συμφωνία επιπέδου υπηρεσίας (Service Level Agreement - SLA). Το υφιστάμενο δίκτυο θα αναλάβει το έργο ώστε να εκτελεστεί η δικτυακή λειτουργία. Οι περισσότερες εταιρίες συμφωνούν στον ιδέα των προγραμματιζόμενων δικτύων. Ο τομέας στον οποία υπάρχει διχογνωμία είναι ο τρόπος με τον οποίο θα γίνει η εφαρμογή. Κάποιοι προμηθευτές υλικού επιθυμούν

να χρησιμοποιήσουν ιδιόκτητο λογισμικό για την επίτευξη του στόχου ενώ άλλοι εξετάζουν το ενδεχόμενο να συμβιβαστούν και να καταφύγουν σε λύσεις ανοιχτού κώδικα.

Αρκετοί προμηθευτές, έχουν βγάλει στην αγορά SDN controllers μαζί με υψηλού επιπέδου εφαρμογές λογισμικού, σαν κομμάτι του δικού τους προγραμματιζόμενου δικτυακού συστήματος. Η Cisco συγκεκριμένα έχει σε εμπορική διάθεση πλήθος προϊόντων για διαφορετικές περιπτώσεις. Αρχικά είχε ξεκινήσει με μία υβριδική προσέγγιση στα SDN, στην οποία το παραδοσιακό επίπεδο ελέγχου συνέχισε να υπάρχει, ενώ ένας εξωτερικός διαχειριστής έδινε δυνατότητα για προγραμματισμό και έλεγχο ροών μέσω εφαρμογών για συγκεκριμένες εταιρικές απαιτήσεις. Συνεχίζοντας στην ίδια λογική στηρίζει έμπρακτα την κοινότητα ανοιχτού κώδικα, συμμετέχοντας ενεργά στην ανάπτυξη του OpenFlow αλλά και του OpenDayLight Project καθώς είναι μία από τα τέσσερα πλατινένια μέλη – εταιρίες. Παράλληλα παρέχει συνδυασμό ιδιόκτητων λύσεων όπως το Cisco Open Networking Environment Platform Kit αλλά και ανοιχτού κώδικα όπως τον Cisco Open SDN Controller.

Στην άλλη πλευρά, οι υποστηρικτές της μη ιδιόκτητης και ελεύθερου κώδικα προσέγγισης πιστεύουν ότι ο βασικός σκοπός που πρέπει να επιτευχθεί είναι η ελεύθερη και πλήρως διαθέσιμη παροχή χωρίς να υπάρχουν κρυμμένα ιδιόκτητα τμήματα. Έτσι η ανεξάρτητη χρήση και ανάπτυξη των SDN θα έχει τη δυνατότητα να εξελιχθεί ακόμα περισσότερο.

Αρκετές εταιρίες υλικού βλέπουν με θετικό μάτι την ιδέα του open source ενώ άλλες είναι αρνητικές. Είναι όμως γεγονός ότι ο ανταγωνισμός στην αγορά τους πιέζει να εξετάσουν τμήματα επιλογές ελεύθερου λογισμικού καθώς πλήθος εταιριών έχουν τα περάσει τα ανοιχτά πρότυπα στην παραγωγή. Διαχειριστές και έργα όπως το OpenDaylight, ο controller OpenContrail, ο Floodlight και φυσικά το Openflow είναι πλήρως διαθέσιμα με άδεια ανοιχτού κώδικα. Εταιρίες όπως η Hewlett Packard και Big Switch ήταν από τις πρώτες που στήριξαν την open source κοινότητα διαθέτοντας εμπορικά λύσεις και hardware συμβατό με SDN εφαρμογές.

Αρκετές εταιρείες έχουν την αντίληψη ότι τα SDN μπορούν να εφαρμοστούν χωρίς τη χρήση του OpenFlow. Προϊόντα τους δεν βασίζονται στα χαμηλότερα επίπεδα του δικτύου και παρέχουν δυνατότητα προγραμματισμού για εικονικές και εύκολα κλιμακωτές υποδομές χωρίς το OpenFlow. Αυτού του είδους τα προϊόντα είναι ευκολότερο να εφαρμοστούν σε επιχειρήσεις και πελάτες που έχουν καταφύγει σε cloud υποδομές. Η λύση αυτή είναι ελκυστική και για εταιρίες που δε θέλουν να αντικαταστήσουν τον υπάρχον εξοπλισμό με OpenFlow συμβατά switches. Ακόμα οι υποστηρικτές του ιδιόκτητου μοντέλου δε θεωρούν αρκετά αποδοτική τη δημιουργία εικονικών δικτύων που η σύνδεση τους με το φυσικό δίκτυο είναι ασθενής. Πιστεύουν πως το ερώτημα αν τα switches λογισμικού μαζί με την ανωτέρου επιπέδου εικονική δικτυακή δομή, είναι ικανά να υποστηρίξουν περιβάλλοντα υψηλών απαιτήσεων δεν μπορεί να απαντηθεί με βάση τη γενική δυνατότητα κλιμάκωσης του προϊόντος αλλά εξαρτάται από κάθε συγκεκριμένη περίπτωση. Έτσι παρουσιάζουν ως κύριο επιχείρημα της προσέγγισης τους την έλλειψη επικοινωνίας και ανάδρασης μεταξύ του λογικού και του φυσικού επιπέδου.

Στο άλλο άκρο, η μη ιδιόκτητη και ανοιχτή προσέγγιση ασκεί κριτική στις εταιρίες που επιλέγουν να διατηρούν κρυφά και ιδιόκτητα τμήματα των δικτυακών λειτουργιών. Η δημιουργία εφαρμογών για τέτοιου είδους συσκευές είναι δύσκολη και πρέπει να λαμβάνει υπό όψιν διαφορετικές παραμέτρους ανάλογα με το λογισμικό της κάθε συσκευής αλλά και να προσαρμόζεται σε ενδεχόμενες αλλαγές όπως αναβαθμίσεις firmware. Η απελευθέρωση της καινοτομίας μαζί με την παγκόσμια προτυποποίηση είναι επιχειρήματα των υποστηρικτών της open source σχολής.

2.3 The OpenDaylight Project

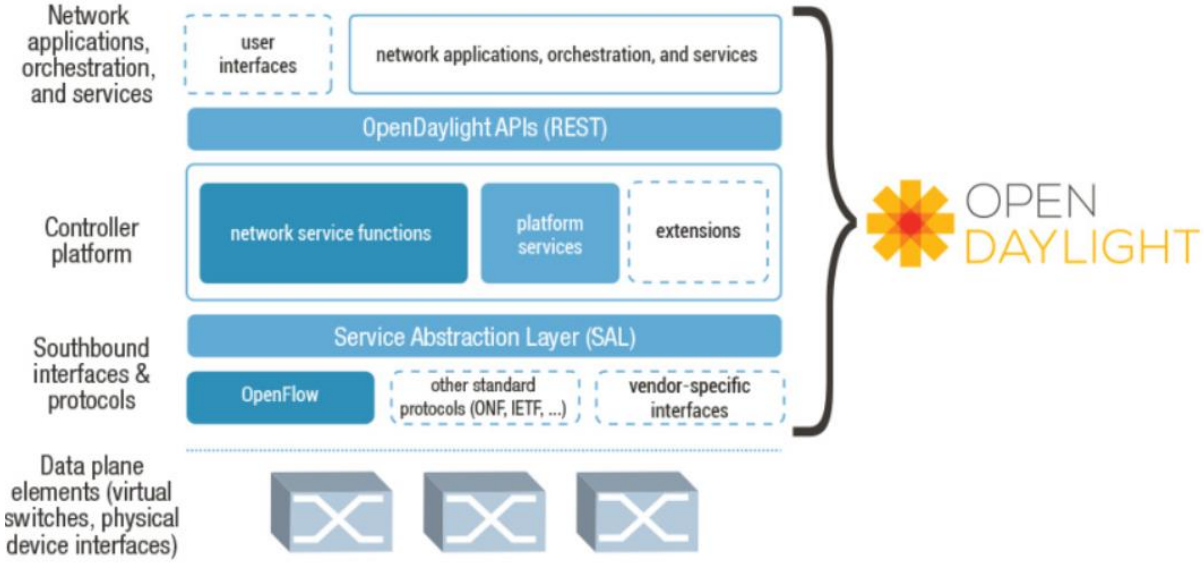
Ο οργανισμός OpenDaylight ιδρύθηκε το 2013, υπάγεται στο Linux Foundation και έχει ως σκοπό την παγκόσμια ανάπτυξη, διανομή αλλά και υιοθέτηση του OpenDaylight διαχειριστή και της πλατφόρμας του ως λύση για τα SDN. Με τη βοήθεια περισσότερων από χίλιους προγραμματιστές, τη στήριξη πάνω από πενήντα μεγάλων εταιριών και συνδρομητές που αγγίζουν το ένα δισεκατομμύριο αποτελεί ένα από τα μεγαλύτερα έργα ανοιχτού κώδικα στο κόσμο. Τα τελευταία χρόνια έχει καταφέρει να διαδοθεί σε πολύ μεγάλο βαθμό και να κερδίσει την υποστήριξη εταιριών κολοσσών στον τομέα των τηλεπικοινωνιών και της τεχνολογίας. Παρατηρώντας τη λίστα αυτή των εταιριών, γίνεται εμφανές το μέγεθος της προσπάθειας και της επένδυσης πάνω στο έργο, αλλά και δικαιολογείται ο υψηλός ρυθμός αύξησης της δημοτικότητας του. Ο ODL Controller είναι ένα Java Framework το οποίο ως Java Virtual Machine μπορεί να τρέξει σε οποιοδήποτε μηχάνημα υποστηρίζει Java.

Platinum

Gold

Silver

2.4 Αρχιτεκτονική της πλατφόρμας OpenDaylight



Η πλατφόρμα OpenDaylight έχει σχεδιαστεί με τμηματική δομή(modular) και είναι βασισμένη στην χρήση ενδιάμεσων plugin που καλύπτουν πληθώρα δικτυακών εφαρμογών και περιπτώσεων. Η δομή αυτή επιτυγχάνεται μέσω του Java OSGi(Open Service Gateway Initiative) framework. Ο controller αποτελείται από

διαφορετικές OSGi δέσμες οι οποίες συνεργάζονται για να παρέχουν την απαιτούμενη λειτουργικότητα. Οι δέσμες αυτές μπορούν να κατηγοριοποιηθούν ως εξής.

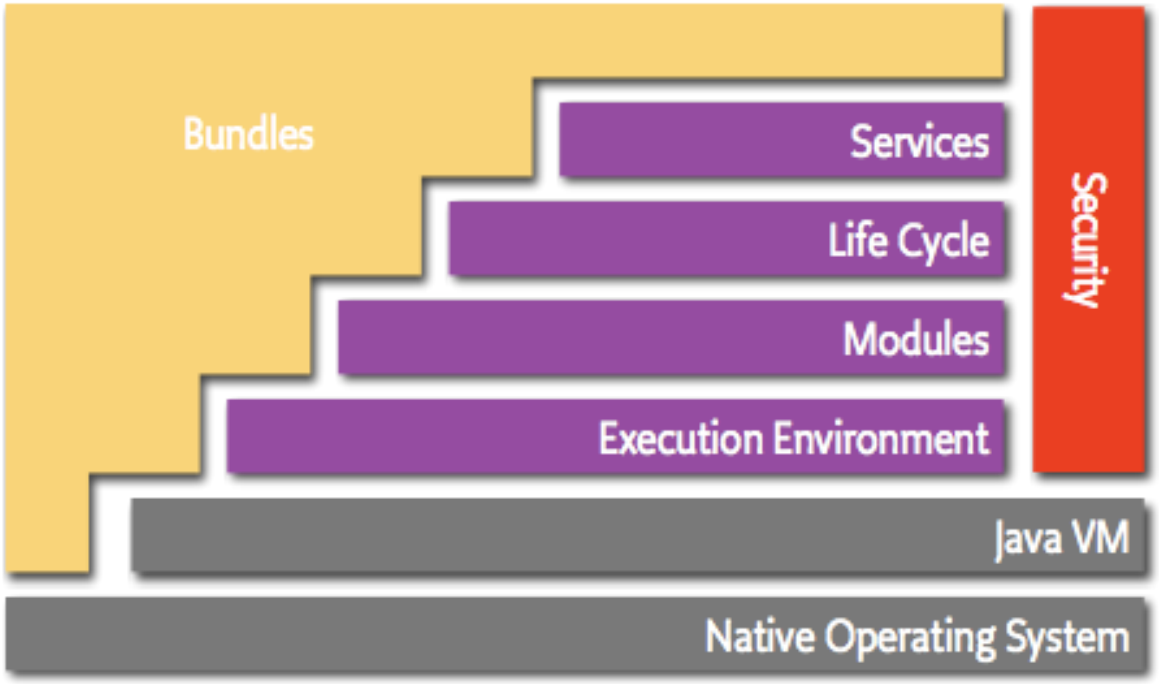
- Network Service Functional Modules
- NorthBound API Modules
- Service Abstraction Layer (SAL)
- SouthBound Plugins
- Application Modules

Η αρχιτεκτονική του Controller θα μπορούσε να διαχωριστεί λογικά σε πέντε επίπεδα σύμφωνα με τα OSGi bundles. Στο κατώτερο επίπεδο βρίσκονται τα Southbound Protocols και APIs, όπως το OpenFlow και ο ρόλος τους είναι να επικοινωνούν με τις δικτυακές συσκευές όπως τα switches. Στο αμέσως ανώτερο επίπεδο βρίσκεται το Service Abstraction Layer το οποίο είναι Model Driven (MD-SAL). Αποτελεί την σημαντικότερη λειτουργία της αρχιτεκτονικής και τον πυρήνα του OpenDaylight. Κάθε δικτυακή συσκευή και εφαρμογή αναπαριστάται ως ένα αντικείμενο, που αποκαλείται μοντέλο. Οι αλληλεπιδράσεις όλων αυτών των οντοτήτων γίνονται στο SAL. Μέσω της γλώσσας μοντελοποίησης Yang έχει σκοπό να «ενώσει» τα ανώτερα επίπεδα με τις δικτυακές συσκευές και να παρέχει λειτουργικότητα και ευελιξία. Στο παραπάνω στρώμα βρίσκεται το module που υπάρχουν συναρτήσεις δικτυακών υπηρεσιών και πάνω από αυτό τα Northbound APIs. Στο ανώτερο όλων των στρωμάτων είναι οι δικτυακές εφαρμογές και τα modules εφαρμογών.

Μπορεί να χρησιμοποιηθεί από μεγάλους οργανισμούς μέχρι απλούς χρήστες. Η τμηματική δομή του, επιτρέπει τη δημιουργία κώδικα και λειτουργιών και τη χρησιμοποίηση συναρτήσεων και υπηρεσιών άλλων προγραμματιστών. Λόγω της μεγάλης υποστήριξης των SDN standards βρίσκει εφαρμογή σε κάθε δίκτυο νέας γενιάς και μπορεί να καλύψει τις ανάγκες του.

2.5 Αρχιτεκτονική των OSGi bundles

Η τεχνολογία Open Services Gateway Initiative (OSGi) είναι ένα σύνολο προδιαγραφών για τον ορισμό ενός δυναμικού τμήματος (component) στη Java. Οι προδιαγραφές αυτές στοχεύουν σε ένα μοντέλο ανάπτυξης, όπου κάθε εφαρμογή δημιουργείται δυναμικά από τη συνεργασία διαφορετικών components. Η επικοινωνία μεταξύ τους γίνεται μέσω υπηρεσιών, που είναι συγκεκριμένα αντικείμενα και διαμοιράζονται μεταξύ των components. Με τον τρόπο αυτό οι εσωτερικές λεπτομέρειες υλοποίησης του κάθε component παραμένουν κρυφές, χωρίς να υπάρχει επίπτωση στη λειτουργικότητα των εφαρμογών.



Το OSGi έχει την παραπάνω δομή διαστρωμάτωσης. Τα bundles είναι τα components που δημιουργούν οι προγραμματιστές. Το επίπεδο των services είναι υπεύθυνο για την επικοινωνία μεταξύ των bundles με δυναμικό τρόπο. Το στρώμα life Cycle παρέχει API για την εγκατάσταση, εκκίνηση, τερματισμό, αναβάθμιση και απεγκατάσταση των bundles. Το στρώμα των Modules καθορίζει πως ένα bundle μπορεί εισάγει και εξάγει κώδικα. Το επίπεδο Security διαχειρίζεται όλες τις λειτουργίες που σχετίζονται με την ασφάλεια. Το Execution Environment ορίζει τις μεθόδους και κλάσεις που είναι διαθέσιμες σε κάθε περιβάλλον.

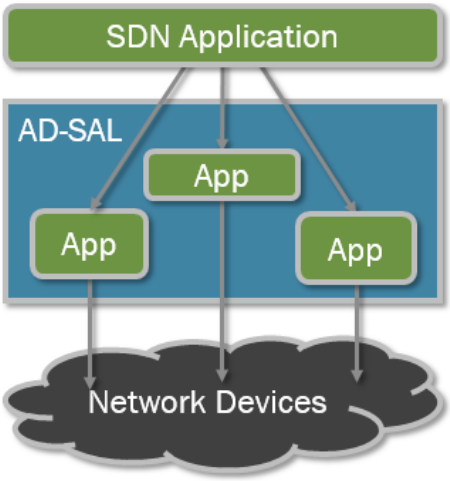
2.6 Ανάλυση των σημαντικότερων OSGi bundles

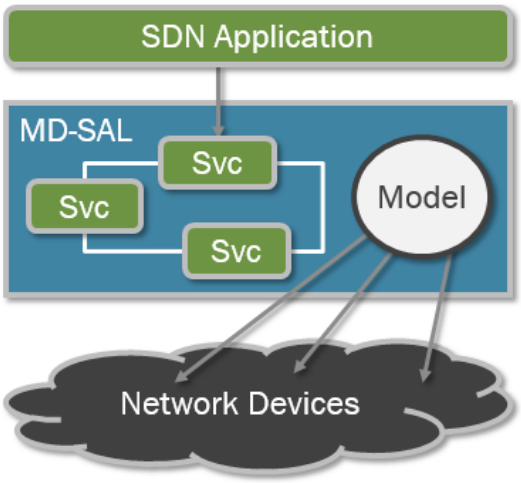
Κάθε SDN controller χρειάζεται τρόπο επικοινωνίας με τα δικτυακά στοιχεία. Για το λόγο αυτό χρειάζονται APIs/Plugins τα οποία αποκαλούνται Southbound. Έχουν πάρει την ονομασία αυτή καθώς επικοινωνούν με την «κάτω» πλευρά του δικτύου, εξετάζοντας το από το φυσικό επίπεδο προς το επίπεδο εφαρμογών. Το πιο διαδεδομένο Southbound API είναι το Openflow. Είναι ανοιχτού κώδικα και δείχνει να προηγείται σε μεγάλο βαθμό σε σχέση με τα υπόλοιπα πρωτόκολλα και APIs. Άλλη μία open source επιλογή αποτελεί το Network Configuration Protocol (NetConf) στο οποίο βασίζεται ο NetConf Client και χρησιμοποιεί Extensive Markup Language για ανταλλαγή δεδομένων. Ο OpenDaylight χρησιμοποιεί ως southbound plugin το NETCONF, το Open vSwitch Database(OVSDB) αλλά και το OpenDove. Επίσης υπάρχουν και αρκετά ιδιόκτητα πρωτόκολλα επικοινωνίας που δεν έχουν διαδοθεί στον ίδιο βαθμό με τα ελεύθερα αλλά και τα κλασικά πρωτόκολλα επικοινωνίας όπως το BGP.

Ο controller όμως πρέπει αν είναι εύκολα προσπελάσιμος και από τις εφαρμογές. Το κομμάτι της επικοινωνίας αυτής είναι κρίσιμο καθώς όλη η λογική των

SDN είναι δεμένη με τον αριθμό και τον τύπο εφαρμογών που μπορούν να υποστηρίξουν. Έτσι υπάρχει πληθώρα διεπαφών, ώστε να υποστηρίζουν κάθε είδους επικοινωνία με συγκεκριμένο τρόπο, προς το επίπεδο εφαρμογών. Αυτού του τύπου τα APIs αποκαλούνται Northbound και αποτελούν ίσως το πιο ουσιαστικό τμήμα των Controllers. Παραδείγματα τέτοιου είδους διεπαφών είναι τα Topology APIs, Bridge Domain APIs, Neutron APIs, Connection Manager APIs . Τα OSGi interfaces χρησιμοποιούνται για εφαρμογές που εκτελούνται στον ίδιο χώρο διευθύνσεων με τον controller ενώ για εφαρμογές σε διαφορετικό χώρο υπάρχουν REST APIs, όπως το RESTCONF. Ανάλογα με τα Interfaces που αξιοποιούν οι εφαρμογές, διαχωρίζονται σε εσωτερικές (internal) και εξωτερικές (external).

Η αρχιτεκτονική του OpenDaylight Controller περιλαμβάνει μια πολύ σημαντική λειτουργία, αυτή του επιπέδου αφηρημένες υπηρεσίας (Service Abstraction Layer - SAL). Στο μοντέλο αυτό υπάρχουν αντικείμενα τα οποία εκθέτουν λειτουργικότητα στα Northbound APIs και αποκαλούνται πάροχοι (providers). Επίσης υπάρχουν αντικείμενα τα οποία αξιοποιούν την λειτουργικότητα των παρόχων και αποκαλούνται καταναλωτές (consumers). Το Service Abstraction Layer είναι υπεύθυνο για τη μεταφορά των μηνυμάτων και δεδομένων μεταξύ των providers και consumers. Πρακτικά είναι το επίπεδο λειτουργίας που αναλαμβάνει την επικοινωνία μεταξύ των protocol plugins και των modules λειτουργιών δικτύου. Το κλασικό μοντέλο υλοποίησης του SAL ονομάζεται API Driven SAL (AD -SAL) και βασίζεται σε προγραμματιστικές διεπαφές. Στον OpenDaylight Controller χρησιμοποιείται ένα διαφορετικό καινοτόμο μοντέλο που αποκαλείται Model – Driven SAL. Στη προσέγγιση αυτή παρέχει τη δυνατότητα να ενωθούν λογικά τα Northbound με τα Southbound Interfaces και οι δομές δεδομένων που χρησιμοποιούνται σε διάφορες υπηρεσίες και οντότητες του SDN Controller. Η γλώσσα μοντελοποίησης που χρησιμοποιείται είναι η YANG. Όταν υπάρχει ανάγκη για δρομολόγηση μεταξύ ενός παρόχου και καταναλωτή το SAL API που θα την υλοποιήσει ορίζεται από μοντέλα, που θα ορίσουν το αντίστοιχο Plugin κατά την εκτέλεση. Στο μοντέλο AD-SAL οι αντίστοιχες προσαρμογές δεδομένων γίνονται κατά τη μεταγλώττιση. Σημαντική διαφορά μεταξύ των δύο υλοποιήσεων είναι ότι κατά το AD-SAL υπάρχει ένα αφιερωμένο στατικό plugin για κάθε Southbound ή Northbound interface ενώ στο MD-SAL υπάρχει ένα γενικό API το οποίο έχει πρόσβαση σε δεδομένα και λειτουργίες που ορίζονται σε μοντέλα.





2.7 Restconf protocol & API

Η διεπαφή Restconf αποτελεί ένα από τα Northbound APIs που εκθέτει ο OpenDaylight Controller ώστε να γίνει εφικτή και με εύκολο τρόπο η επικοινωνία με τα δικτυακά elements. Είναι ένα μέσο ώστε οι υψηλού επιπέδου εφαρμογές να μπορούν να αλληλεπιδράσουν με τα εσωτερικά τμήματα της δομής του Controller χωρίς να χρειάζεται να γνωρίζουν σε βάθος την αρχιτεκτονική και την υλοποίηση του.

Αποτελεί πρωτόκολλο επικοινωνίας που τρέχει πάνω από το HTTP(Hypertext Transfer Protocol) ώστε να αλληλεπιδράσει μια εξωτερική εφαρμογή με τα yang data models στον controller. Όπως λέει και το όνομα του ακολουθεί τις αρχές του REST(Representational State Transfer) σχεδιασμού. Αποτελεί προσχέδιο της IETF(Internet Engineering Task Force) όπου περιγράφεται πώς να προσδιοριστεί ένα YANG στοιχείο σε μία REST διεπαφή.

Σκοπός του είναι να παρέχει μια εύχρηστη διεπαφή. Κάθε ένα κομμάτι YANG εκτίθεται σε ένα URL και είναι εύκολα προσβάσιμο μέσω HTTP. Έτσι οι εξωτερικές εφαρμογές μπορούν να αλληλεπιδράσουν με τα δύο datastores του Controller(config, operational).

Το Restconf API «ακούει» στην πόρτα 8181 και υποστηρίζει τις κλασικές HTTP λειτουργίες όπως GET, PUT, POST, DELETE. Τα αιτήματα και οι απαντήσεις είναι σε JSON(JavaScript Object Notation) ή XML(Extensible Markup Language).

2.8 Apache Karaf

Το Apache Karaf είναι ένας ελαφρύς container που βασίζεται στην αρχιτεκτονική OSGi (Open Service Gateway Initiative) και υποστηρίζει την χρησιμοποίηση των OSGi εφαρμογών. Το μοντέλο OSGi έχει τμηματική δομή (modular) και κάθε κομμάτι (component) αποκαλείται bundle αλλά συχνά αναφέρεται και ως plug-in. Κατά την OSGi, τα bundles είναι συνήθως αλληλοεξαρτώμενα. Αυτό

σημαίνει ότι για να δημιουργηθεί μια OSGi εφαρμογή, στις περισσότερες των περιπτώσεων, απαιτείται η χρησιμοποίηση αρκετών bundles. Το Apache Karaf παρέχει ένα απλό και ευέλικτο τρόπο για τροφοδότηση των εφαρμογών με bundles. Στο Apache Karaf κάθε εφαρμογή αποκαλείται feature.

Μερικές από τις κύριες χρήσεις του karaf container είναι η εγκατάσταση features στον OpenDaylight Controller και η εκκίνηση του. Η εγκατάσταση γίνεται εύκολα μέσω του karaf cli με την εντολή feature:install <feature name> και δίνεται η δυνατότητα μαζικής εγκατάστασης. Π.χ. *feature:install odl-restconf odl-mdsal-apidocs.* Μέσω της εντολής *feature:list* εμφανίζονται τα διαθέσιμα features και με την παράμετρο –i τα εγκατεστημένα.
Πηγαίνοντας στον φάκελο που βρίσκεται ο Controller με την εντολή *./bin/karaf* γίνεται η εκκίνηση του OpenDaylight και βρισκόμαστε στο karaf CLI. Αρχικά δεν υπάρχει κανένα εγκατεστημένο feature. Αν θέλουμε να εκκινήσουμε τον controller διαγράφοντας εγκατεστημένα features προσθέτουμε την παράμετρο clean στην αρχική εντολή (*./bin/karaf clean*).

Κεφάλαιο 3 – Openflow

3.1 Ορισμός

Το Openflow θεωρείται το πρώτο πρότυπο των SDN. Η αρχική λειτουργία του ήταν να ορίσει ένα πρωτόκολλο για την επικοινωνία των Controllers και των δικτυακών συσκευών, όπως τα switches (φυσικά και εικονικά). Αποτελεί το σημείο επικοινωνίας των Southbound APIs με τις δικτυακές συσκευές, οι οποίες φυσικά πρέπει να υποστηρίζουν το πρωτόκολλο.

3.2 Ιστορικό

Το Openflow έγινε γνωστό στο ευρύ κοινό το έτος 2007. Η αρχική προσέγγιση ήταν να δημιουργηθεί ένας εναλλακτικός τρόπος δικτύωσης , ο οποίος θα ήταν ανοιχτού κώδικα και δωρεάν χωρίς να έχει τους περιορισμούς του ιδιόκτητου λογισμικού. Η ιδέα ήταν να παρέχει τη δυνατότητα στους μηχανικούς δικτύων να πειραματιστούν σε πραγματικά δίκτυα και εξοπλισμό με σκοπό την ανάπτυξη της καινοτομίας.
Στο αρχικό ερευνητικό στάδιο απευθύνθηκε σε πανεπιστήμια, προτρέποντας τους να το εφαρμόσουν στα δίκτυα τους. Η ιδέα της επικοινωνίας ετερογενών συσκευών με ομοιόμορφο τρόπο υποσχόταν να εξομαλύνει κάποιες από τις δυσκολίες των δικτύων υπολογιστών και να μας φέρει πιο κοντά σε αυτό που αντιλαμβανόμαστε σήμερα ως Software Defined Networking. Παράλληλα θα ενίσχυε την έρευνα καθώς θα εφαρμοζόταν πάνω σε εμπορικές συσκευές και πραγματικά δίκτυα χωρίς να χρειαστεί οι προμηθευτές να εκθέσουν τα «μυστικά» των υλοποιήσεων τους. Η αρχική προσέγγιση ήταν να υπάρχει πλήρης διαχωρισμός της «πραγματικής» κίνησης με την πειραματική, με χρήση διαφορετικών VLAN. Το πανεπιστήμιο του Stanford ήταν το πρώτο που δέχθηκε την πρόκληση και εφάρμοσε το Openflow στης υποδομές του.

Μέχρι σήμερα έχει γίνει μεγάλη πρόοδος και έχουν δημιουργηθεί πολλά πρωτόκολλα και Plugins για επικοινωνία των Controllers και των συσκευών. Μερικά από αυτά είναι τα OpenDove, LISP, NETCONF, OVSDB, BGP. Το Openflow έχει τη μεγαλύτερη χρήση και έχει διαδοθεί σε μεγάλο βαθμό. Το 2012 αρκετές εταιρίες όπως η Hewlett- Packard , η IBM, η Brocade και Juniper άρχισαν να υποστηρίζουν το πρωτόκολλο στις εμπορικές συσκευές τους. Σήμερα η λίστα είναι πολύ μεγαλύτερη και τα περισσότερα εμπορικά switches έχουν την υποδομή να το υποστηρίξουν. Η Google μάλιστα έχει υλοποιήσει μεγάλο κομμάτι του δικτύου τους ακολουθώντας τα πρότυπα των SDN και κάνοντας χρήση του Openflow.

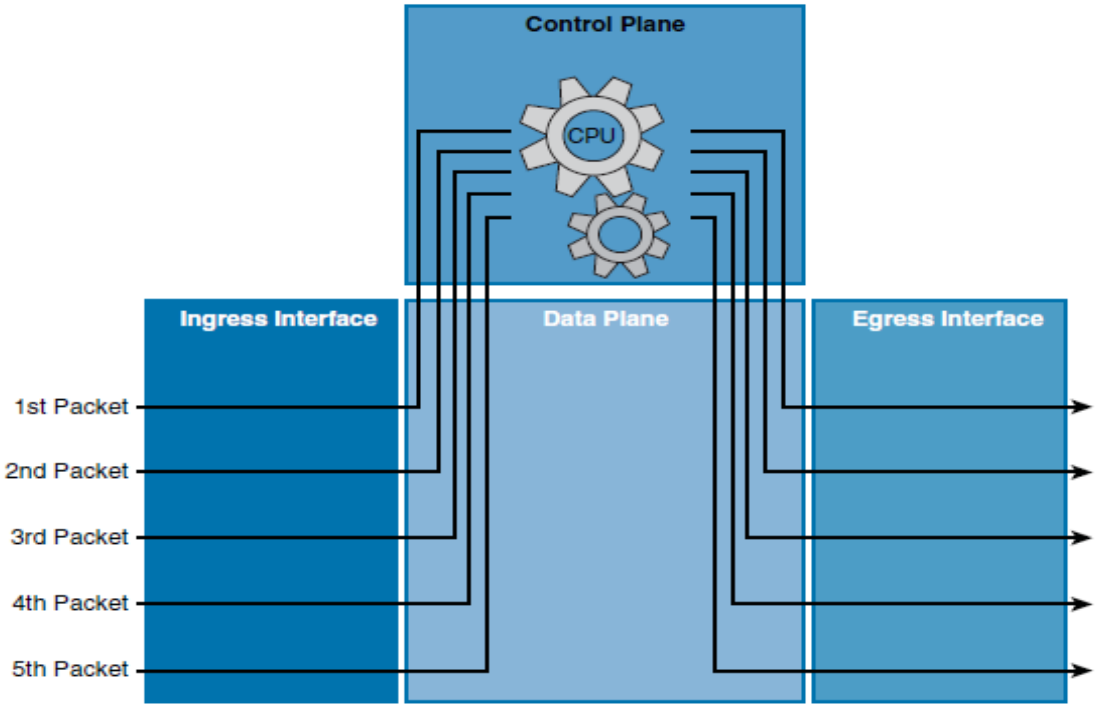
3.3 Ανάπτυξη και εταιρίες - μέλη

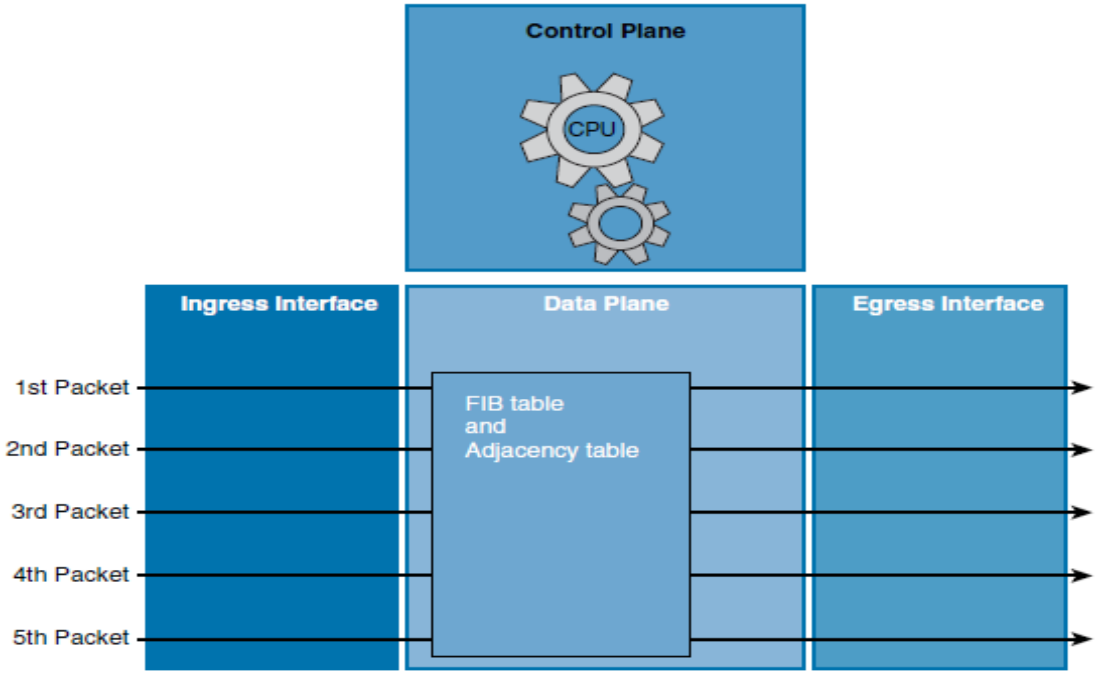
Το Openflow αναπτύσσεται από το Open Networking Foundation – ONF. Είναι ένα μη κερδοσκοπικό ίδρυμα που η κινητήριος δύναμη του είναι οι χρήστες του. Ο σκοπός του είναι η διάδοση και υιοθέτηση των SDN μέσα από ανάπτυξη ανοιχτών προτύπων. Το μεγαλύτερο έργο του είναι το Openflow, που είναι κοινώς αποδεκτό ως το πρώτο SDN πρότυπο. Ιδρύθηκε το 2011 από τις εταιρίες Facebook, Google, Microsoft, Verizon, Yahoo και Deutsche Telekom. Σήμερα η υποστήριξη στο ίδρυμα έχει πλαισιωθεί με περισσότερες εταιρίες και παρατίθεται σχετική λίστα.

3.4 Λειτουργία του Openflow

Το Openflow ακολουθεί τα πρότυπα των SDN εφαρμόζοντας τον διαχωρισμό του επιπέδου δεδομένων ή προώθησης με το επίπεδο ελέγχου. Σε ένα κλασικό switch οι λειτουργίες αυτές , όπως αποφάσεις δρομολόγησης, συμβαίνουν στο ίδιο hardware. Στα Openflow switches γίνεται μόνο προώθηση πακέτων. Αν χρειαστεί να ληφθεί οποιαδήποτε απόφαση υψηλού επιπέδου δρομολόγησης, για παράδειγμα να βρεθεί ο δρόμος για ένα δίκτυο για το οποίο δεν υπάρχουν κανόνες, τότε μέσω του Openflow protocol επικοινωνεί με τον SDN Controller για να του δώσει την απάντηση.

Τα switches έχουν ένα σύνολο από πίνακες με κανόνες. Κάθε κανόνας ταυτίζεται με συγκεκριμένη κίνηση, για παράδειγμα η Ip διεύθυνση προορισμού να ανήκει στο δίκτυο 10.1.0.0/16. Όταν ταυτιστεί η κίνηση με τον κανόνα, ανάλογα και με την προτεραιότητα που έχει, ακολουθούν οι δράσεις που έχουν οριστεί για τον κανόνα αυτό, π.χ. στείλε τα πακέτα στην φυσική πόρτα ένα ή πέταξε τα πακέτα. Οι κανόνες αυτοί ονομάζονται ροές (flows). Τα Openflow switches έχουν πίνακες ροών που ορίζουν την ενέργεια για κάθε πακέτο. Αν ένα πακέτο όταν εισέρχεται δεν έχει κάποια ροή που να ταυτίζεται μαζί του, συνήθως όταν είναι η πρώτη φορά, τότε γίνεται ερώτηση στον διαχειριστή για εισαγωγή αντίστοιχης ροής. Λόγω της μεγάλης διάδοσης του, έχει επικρατήσει οι Controllers που χρησιμοποιούν το πρωτόκολλο αυτό ως μέσο επικοινωνίας να ονομάζονται Openflow Controllers και τα switches ή network elements Openflow switches.


Control Plane
CPU
Ingress Interface
Data Plane
Egress Interface
1st Packet
2nd Packet
3rd Packet
4th Packet
5th Packet


Control Plane
CPU
Ingress Interface
Data Plane
Egress Interface
1st Packet
2nd Packet
3rd Packet
4th Packet
5th Packet
FIB table
and
Adjacency table

ΚΕΦΑΛΑΙΟ 4 – Εξομοιωτές και Προσομοιωτές Δικτύων

4.1 Εισαγωγή

Στον κόσμο των δικτύων, είναι γνωστό ότι μια εφαρμογή με λανθασμένη παραμετροποίηση μπορεί να προκαλέσει μεγάλη απώλεια χρόνου αλλά και χρημάτων. Τα δίκτυα όμως εξελίσσονται συνεχώς και οι αλλαγές στη δομή και τις εφαρμογές που λειτουργούν σε αυτά είναι καθημερινές. Ο καλύτερος τρόπος για να αποφευχθούν τέτοιου είδους ατυχήματα είναι μέσω τακτικών δοκιμών πάνω στις επερχόμενες αλλαγές. Είτε ο ζητούμενος στόχος είναι ο σχεδιασμός ενός νέου δικτύου ή η μεταφορά των υποδομών σε περιβάλλον cloud ή προσθήκη καινούργιου υλικού, κάθε βήμα των μετατροπών αυτών πρέπει να βασίζεται σε λεπτομερείς δοκιμές.

Όσον αφορά τις δικτυακές δοκιμές, οι όροι εξομοίωση και προσομοίωση χρησιμοποιούνται με την ίδια συχνότητα. Στις περισσότερες περιπτώσεις , όποια λέξη και αν χρησιμοποιηθεί το τελικό νόημα θα γίνει κατανοητό. Όμως υπάρχει μεγάλη διαφορά μεταξύ προσομοιωτών και εξομοιωτών, τόσο σε πρακτικό επίπεδο όσο και σε σημασιολογικό.

Ο προσομοιωτής έχει τη δυνατότητα να εκτελεί λειτουργίες ώστε να παρουσιάσει τη συμπεριφορά ενός δικτύου και τον τμημάτων του. Σε αντίθεση ο εξομοιωτής αντιγράφει τη συμπεριφορά των δικτύων με σκοπό να το αντικαταστήσει λειτουργικά.

Η έρευνα στην οποία στηρίχτηκε και εξακολουθεί να βασίζεται η μελέτη των SDN , στις περισσότερες των περιπτώσεων, χρησιμοποιεί τέτοιου είδους προγράμματα. Είναι πολύ δύσκολο να βρεθεί πρόσβαση για μελέτη σε πραγματικό δίκτυο και ακόμα πιο δύσκολο να υπάρχει δυνατότητα για πειραματισμούς.

4.2 Προσομοιωτές

Σε χαμηλό επίπεδο οι προσομοιωτές δικτύων χρησιμοποιούν μαθηματικές φόρμουλες για να δημιουργήσουν ένα θεωρητικό και εικονικό μοντέλο ενός δικτύου. Είναι λογισμικό που παρέχει λύσεις και ανάλογα με την εφαρμογή υπάρχουν διαφορετικοί τύποι. Αν και έχουν μεγαλύτερη χρησιμοποίηση σε ερευνητικό και ακαδημαϊκό επίπεδο, μπορούν να χρησιμοποιηθούν και ως εργαλεία δοκιμής για το σχεδιασμό και την ανάπτυξη ενός δικτύου. Μερικοί από τους πιο γνωστούς είναι οι ns - 2, ns - 3, OPNET, OMNeT++, NetSim, REAL, QualNet, J-Sim.

Προσομοιωτές, όπως ο ns – 3, χρησιμοποιούνται για την προσομοίωση δικτυακών πρωτοκόλλων και πρωτοκόλλων δρομολόγησης. Βασίζονται στην διακριτή προσομοίωση γεγονότων όπου χρονικά τοποθετεί τα γεγονότα σε ουρά και τα εκτελεί σαν ροή δεδομένων. Δίνει τη δυνατότητα στους μηχανικούς δικτύου να αξιολογήσουν ένα πειραματικό μοντέλο των δικτύων, συμπεριλαμβάνοντας την

τοπολογία και την ροή των εφαρμογών. Έτσι δημιουργώντας πληθώρα σεναρίων πραγματικής κίνησης , μπορούμε να εξάγουμε ένα υποθετικό συμπέρασμα για το πραγματικό δίκτυο. Παρότι οι δοκιμές αυτές μπορούν να γλιτώσουν χρόνο και χρήμα, οι προσομοιωτές έχουν τα μειονεκτήματα τους. Οι υψηλής πολυπλοκότητας λειτουργίες απαιτούν μεγάλο βαθμό εμπειρίας πάνω στα συγκεκριμένα εργαλεία ώστε η εξαγωγή συμπερασμάτων να είναι αξιόπιστη. Επίσης σε αρκετές περιπτώσεις δεν είναι αρκετά πρακτικοί καθώς είναι αδύνατο να προσομοιωθούν ορισμένα γεγονότα χωρίς να ληφθεί υπό όψη το φυσικό δίκτυο.

4.3 Εξομοιωτές

Οι εξομοιωτές δικτύου, συχνά αποκαλούμενοι και εξομοιωτές δικτύων μεγάλης εμβέλειας (WAN emulators), χρησιμοποιούνται για δοκιμαστικές μετρήσεις πραγματικών δικτύων. Οι συσκευές αυτές μπορούν να χρησιμοποιηθούν για σκοπούς όπως η διασφάλιση της ποιότητας, απόδειξη για συγκατάθεση ή βλαβοδιαχείριση. Είναι διαθέσιμοι ως λύσεις λογισμικού αλλά και υλικού. Δίνουν τη δυνατότητα στους μηχανικούς και τους σχεδιαστές δικτύων να μετρήσουν με ακρίβεια μετρικές των εφαρμογών, όπως χρόνοι ανταπόκρισης, τον καθαρό ρυθμό μεταφοράς δεδομένων γνωστό ως throughput και την εμπειρία της ποιότητας του τελικού χρήστη προτού προβούν σε αλλαγές ή προσθήκες στα συστήματα που διαχειρίζονται. Γνωστοί εξομοιωτές δικτύων είναι ο Mininet, ο CORE και ο EVE – NG.

Τοποθετώντας έναν εξομοιωτή μεταξύ δύο τμημάτων τοπικών δικτύων, μπορούμε να έχουμε ένα αντίγραφο της κίνησης μεταξύ εξυπηρετητή και πελάτη χωρίς την ανάγκη για πραγματικό router ή modem ή έστω και πραγματική κίνηση. Είναι εφικτό να παραμετροποιηθεί με τέτοιο τρόπο ώστε να χειραγωγήσει παραμέτρους κίνησης όπως απώλεια πακέτων, καθυστέρηση και διαφορά ελάχιστης με μέγιστη καθυστέρηση(jitter) . Η καθυστέρηση μπορεί τροποποιηθεί με ειδικές παραμέτρους ώστε να εξομοιώσει την κίνηση για μεγάλη αποστάσεις. Έτσι οι εφαρμογές ανταποκρίνονται σαν να υπήρχε φυσικός διαχωρισμός. Η απόδοση των εφαρμογών και η εμπειρία του τελικού χρήστη μπορούν να παρατηρηθούν, να δοκιμαστούν και να πιστοποιηθούν υπό τέτοιες συνθήκες σε πραγματικό χρόνο.

4.4 Mininet Emulator

Το Mininet παρέχει τη δυνατότητα δημιουργίας ενός ρεαλιστικού εικονικού δικτύου, τρέχοντας πραγματικούς πυρήνες (kernel), μεταγωγούς (switches) και κώδικα εφαρμογών σε ένα μηχάνημα. Το μηχάνημα αυτό μπορεί να είναι εικονικό(VM), σε cloud ή τοπικό και η δημιουργία του δικτύου διαρκεί μερικά δευτερόλεπτα ενώ απαιτεί μία μοναδική εντολή.

Λόγω της ευκολίας στην αλληλεπίδραση με το δίκτυο χρησιμοποιώντας το Mininet CLI (Command Line Interface) και API, στην παραμετροποίηση, στον διαμοιρασμό και την εφαρμογή του σε πραγματικό υλικό το Mininet μπορεί να χρησιμοποιηθεί στην ανάπτυξη , τη διδασκαλία και την έρευνα δικτύων. Υπάρχει ενεργή ομάδα ανάπτυξης και υποστήριξης και διατίθεται μέσω άδειας ανοιχτού κώδικα.

Το Mininet είναι ένας εξομοιωτής για τη δημιουργία δικτύων εικονικών τερματικών, μεταγωγών, διαχειριστών και συνδέσμων. Στα τερματικά εκτελείται τυπικό λογισμικό δικτύων Linux και οι μεταγωγοί υποστηρίζουν το πρωτόκολλο Openflow και μέγιστη ευελιξία στη δρομολόγηση μέσω του Software Defined Networking.

Υποστηρίζει την έρευνα, ανάπτυξη, μάθηση, δοκιμή , αναγνώριση σφαλμάτων και κάθε είδους λειτουργία. Μέσα σε μία τερματική συσκευή, όπως φορητό ή σταθερό υπολογιστή, παρέχονται όλα τα πλεονεκτήματα που θα έβρισκε κανείς αν είχε ένα πλήρες πειραματικό δίκτυο.

Το Mininet:

- Παρέχει ένα απλό και οικονομικό δικτυακό περιβάλλον δοκιμών για ανάπτυξη εφαρμογών συμβατών με το Openflow.
- Δίνει τη δυνατότητα σε ομάδα ατόμων να εργαστεί ταυτόχρονα και ανεξάρτητα στην ίδια τοπολογία.
- Υποστηρίζει δοκιμές παλινδρόμησης συστημάτων, τα οποία είναι επαναλαμβανόμενα και εύκολα αποθηκευόμενα.
- Επιτρέπει τις δοκιμές πολύπλοκων τοπολογιών, χωρίς την χρονοβόρα ανάγκη αλλαγών στη φυσική καλωδίωση.
- Περιλαμβάνει CLI για απασφαλμάτωση (debugging) και διεξαγωγή μακροσκελών δικτυακών ελέγχων, το οποίο έχει γνώση της τοπολογίας και του Openflow.
- Είναι εφικτή η δημιουργία οποιασδήποτε τοπολογίας αλλά έχει και βασική ομάδα τοπολογιών που μπορούν να παραμετροποιηθούν.
- Μπορεί να χρησιμοποιηθεί κατευθείαν χωρίς τον παραμικρό προγραμματισμό.
- Έχει όμως και εύκολη στη χρήση διεπαφή σε Python για δημιουργία και πειραματισμό δικτύων.

Το Mininet παρέχει έναν εύκολο τρόπο για διόρθωση της συμπεριφοράς συστημάτων και δοκιμές με διάφορες τοπολογίες. Ο μοναδικός περιορισμός δημιουργείται από τις δυνατότητες του υλικού και λογισμικού του μηχανήματος στο οποίο εκτελείται. Τρέχει «πραγματικό» κώδικα συμπεριλαμβάνοντας τις κλασικές δικτυακές εφαρμογές του λειτουργικού συστήματος Unix/Linux όπως και τον Linux kernel και τη δικτυακή στοίβα.

Λόγω των παραπάνω ο κώδικας που δημιουργείται και δοκιμάζεται στο περιβάλλον του Mininet, είτε είναι για έναν Openflow Controller, για switch ή τερματική συσκευή μπορεί να μεταφερθεί σε πραγματικό σύστημα με ελάχιστες αλλαγές για δοκιμές σε πραγματικό περιβάλλον και αξιολόγηση επίδοσης. Αυτό

σημαίνει πως ένας σχεδιασμός που λειτουργεί στο Mininet μπορεί απευθείας να μεταφερθεί σε πραγματικό υλικό για προώθηση με τους ρυθμούς της γραμμής.

4.5 Λειτουργία του Mininet

Σχεδόν κάθε λειτουργικό σύστημα εικονικοποιεί τους επεξεργαστικούς πόρους χρησιμοποιώντας την έννοια της τμηματοποίησης των διεργασιών (process abstraction). Το Mininet χρησιμοποιεί εικονικοποίηση βασισμένη σε διεργασίες για να τρέχει πολλαπλά (έχει δοκιμαστεί επιτυχώς μέχρι και με 4096) τερματικά και μεταγωγούς σε ένα μοναδικό πυρήνα λειτουργικού συστήματος. Μετά την έκδοση 2.2.25, το Linux υποστηρίζει τα network namespaces, λειτουργία ελαφριάς (σε δέσμευση πόρων) εικονικοποίησης που παρέχει ξεχωριστές διεργασίες με ξεχωριστές δικτυακές διεπαφές, πίνακες δρομολόγησης και πίνακες ARP (Address Resolution Protocol). Το Mininet μπορεί να δημιουργήσει πυρήνες ή OpenFlow μεταγωγούς, διαχειριστές για τα switches και τερματικές συσκευές για επικοινωνία πάνω στο προσομοιωμένο δίκτυο. Το Mininet ενώνει τους μεταγωγούς με τα τερματικά χρησιμοποιώντας εικονικά Ethernet (veth) ζευγάρια. Παρότι ,προς το παρόν, το Mininet στηρίζεται στον Linux kernel, μελλοντικά ενδέχεται να υποστηρίζει και άλλα λειτουργικά συστήματα. Τέλος, ο κώδικας του Mininet είναι σχεδόν εξολοκλήρου γραμμένος σε Python, με μικρή εξαίρεση ελάχιστες λειτουργίες σε C.

4.6 Πλεονεκτήματα και Μειονεκτήματα

Το Mininet συγκεντρώνεται λειτουργίες εξομοιωτών, προσομοιωτών και μηχανημάτων δοκιμών (hardware testbeds).

Συγκριτικά με άλλα παρόμοια συστήματα:

- Έχει γρηγορότερη εκκίνηση (μερικά δευτερόλεπτα).
- Προσφέρει μεγάλες δυνατότητες μεγέθυνσης του δικτύου (εκατοντάδες τερματικά και switches).
- Παρέχει μεγαλύτερο bandwidth
- Είναι εύκολο στην εγκατάσταση, υπάρχουν έτοιμα VMs που τρέχουν σε VMware ή VirtualBox για όλα τα λειτουργικά συστήματα.

Συγκριτικά με μηχανήματα δοκιμών:

- Είναι φθηνότερο και πάντα διαθέσιμο
- Είναι ευκολότερο και πιο γρήγορο να παραμετροποιηθεί αλλά και να κάνει επανεκκίνηση.

Στα μειονεκτήματα θα πρέπει να σημειωθεί ότι τα Mininet δίκτυα δεν μπορούν να ξεπεράσουν την διαθέσιμη επεξεργαστική ισχύ ή το εύρος ζώνης ενός μηχανήματος. Ακόμα δε μπορεί να τρέξει εφαρμογές ή Openflow switches που δεν είναι συμβατές με το λειτουργικό σύστημα Linux.

4.7 Εντολές και εφαρμογές

Στον εξομοιωτή Mininet έχουμε τη δυνατότητα να δημιουργήσουμε ένα εικονικό δίκτυο εύκολα και γρήγορα. Ανοίγοντας ένα τερματικό στο λειτουργικό σύστημα Linux και πληκτρολογώντας την εντολή ***sudo mn*** έχουμε την πιο απλή μορφή δικτύου με ένα switch και δύο hosts συνδεδεμένους σε αυτό. Παρακάτω καταγράφονται χρήσιμες εντολές με παραδείγματα που χρησιμοποιήθηκαν στα πλαίσια της εργασίας.

--controller=remote,ip=X.X.X.X

Ορίζει έναν απομακρυσμένο controller και την Ip διεύθυνση του. Αν ο controller βρίσκεται στο ίδιο μηχάνημα μπορεί να χρησιμοποιηθεί η localhost Ip, 127.0.0.1

--controller=remote,ip=127.0.0.1

--topo

Ορίζει την τοπολογία που θα δημιουργηθεί. Υπάρχουν έτοιμα μοντέλα όπως linear, tree αλλά και η δυνατότητα να δημιουργηθεί από python αρχείο για μεγαλύτερο έλεγχο.

--topo tree,3 (τοπολογία δυαδικού δέντρου με ύψος 3, 7 switches και 8 hosts)

--mac

Εξορισμού κάθε φορά που εκκινεί το Mininet δημιουργεί τυχαίες mac διευθύνσεις στους hosts. Αυτό μπορεί να δυσκολέψει και να καθυστερήσει αρκετά τους ελέγχους όπως και την εφαρμογή ροών στα switches. Με την παραπάνω εντολή οι τερματικές συσκευές παίρνουν μικρές και ευανάγνωστες τιμές. Π.χ. σε ένα δίκτυο με τέσσερις hosts θα δοθούν οι διευθύνσεις 00:00:00:00:00:01, 00:00:00:00:00:02, 00:00:00:00:00:03, 00:00:00:00:00:04.

--link

Παρέχει τη δυνατότητα ορισμού παραμέτρων στους συνδέσμους όπως χωρητικότητα και καθυστέρηση. Π.χ. η παράμετρος *–link tc,bw=100,delay=5ms* δημιουργεί συνδέσμους με χωρητικότητα 100 Mbps και καθυστέρηση πέντε χιλιοστών του δευτερολέπτου (5ms). Αξίζει να σημειωθεί ότι αν εκτελεστεί η εντολή ping μεταξύ δύο τερματικών στο ίδιο switch ο συνολικός χρόνος (Round Trop Time - RTT) θα είναι 20 ms καθώς τα πακέτα διέρχονται από τους συνδέσμους τέσσερις φορές.

--switch

Αναφέρεται ο τύπος του switch και το πρωτόκολλο που θα εκτελείται σε αυτό. --switch ovsk,protocols=OpenFlow13. Η παραπάνω εντολή δημιουργεί ένα switch τύπου Open vSwitch(OVS – Open Virtual Switch) το οποίο τρέχει το πρωτόκολλο OpenFlow στην έκδοση 1.3.

Συνδυασμός των παραπάνω εντολών χρησιμοποιείται σε μία γραμμή κατά την εκκίνηση του Mininet.

sudo mn --controller=remote,ip=127.0.0.1 --mac --link tc,bw=100 --topo tree,3 --switch ovsk,protocols=OpenFlow13

Μόλις δημιουργηθεί το Mininet δίκτυο εισερχόμαστε στο Mininet CLI το οποίο διαθέτει πληθώρα εντολών για την εμφάνιση των δικτυακών στοιχείων αλλά διεξαγωγή ελέγχων. Μερικές από τις πιο σημαντικές είναι οι:

nodes: Εμφανίζει τους κόμβους.

net: Εμφανίζει τους συνδέσμους.

ping: Δίνει τη δυνατότητα ελέγχου συνδεσιμότητας δύο σημείων μέσω της εντολής ping. Π.χ. *h1 ping h2*. Εκτελεί την εντολή ping μεταξύ του Host1 και Host2. Επιθυμητή η παράμετρος *–c XX* (όπου XX ο αριθμός των icmp πακέτων που θα στείλει, -c 10).

pingall: Εκτελεί την εντολή ping μεταξύ όλων των ζευγαριών στο δίκτυο.

iperf: Ελέγχει το διαθέσιμο bandwidth. Π.χ. *iperf h1 h2*

xterm: Δημιουργεί τερματικό παράθυρο σε κάθε host του δικτύου. Μπορεί να χρησιμοποιηθεί για συγκεκριμένους host. Π.χ. *xterm h1 h3*, ανοίγει τερματικό παράθυρο μόνο στους Host 1 και Host 3.

ΚΕΦΑΛΑΙΟ 5 – SDN και Quality of Service

5.1 Ποιότητα της Υπηρεσίας

Λαμβάνοντας υπό όψιν την σημερινή κατάσταση του Internet , οι τεχνικές και οι προσεγγίσεις που αφορούν την ποιότητα της υπηρεσίας έχουν αποδειχθεί περιοριστικές και υστερούν στον τομέα της ευελιξίας. Η ανάγκη για μια καλύτερη λύση, με μεγαλύτερες δυνατότητες κλιμάκωσης, που θα επιτρέπει την λεπτομερή ρύθμιση της δικτυακής κίνησης, γίνεται ολοένα και μεγαλύτερη. Όσον αφορά την QoS στα SDN , υπάρχουν αρκετά ολοκληρωμένα συστήματα που παρέχουν δυνατότητα παραμετροποίησης του δικτύου και υψηλού επιπέδου χειρισμού των πολιτικών.

Στην παραδοσιακή δικτυακή δομή υπάρχει μεγάλο πλήθος πρωτοκόλλων που καθορίζουν πως διαφορετικές τερματικές συσκευές σε διαφορετικά δίκτυα θα επικοινωνήσουν και θα συνδεθούν με αξιοπιστία. Πάραυτα, τα πρωτόκολλα αυτά τείνουν να έχουν σχεδιαστεί με τοπική προσέγγιση και για να επιλύουν πολύ συγκεκριμένα προβλήματα. Η προσέγγιση αυτή αυξάνει την πολυπλοκότητα των δικτύων και είναι ένας από τους μεγαλύτερους περιορισμούς γενικότερα για τα δίκτυα αλλά και για την εφαρμογή QoS. Λόγω των παραπάνω ζητημάτων τα σημερινά δίκτυα έχουν στατική δομή. Σχεδόν όλες οι αποφάσεις ελέγχου, λαμβάνονται σε διαφορετικές συσκευές που προωθούν τα δεδομένα. Το γεγονός αυτό αποτελεί μεγάλο εμπόδια για την QoS και την δυναμική φύση τους. Οι μηχανικοί δικτύων αναγκάζονται να διαχειρίζονται κάθε συσκευή ξεχωριστά, ακολουθώντας κανόνες και εντολές διαφορετικών προμηθευτών, τροποποιώντας παραμέτρους ώστε να υλοποιηθούν οι προκαθορισμένοι κανόνες και πολιτικές.

Αυτού του είδος η δομή είναι αδύνατο να προσαρμοστεί δυναμικά στις ανάγκες των εφαρμογών και των χρηστών, οι οποίες αλλάζουν συνέχεια.

Αγνοώντας τη λύση των SDN, η ποιότητα της υπηρεσίας βασίζεται σε συμφωνίες από άκρο σε άκρο μεταξύ των τερματικών συσκευών και σε συμφωνίες επιπέδου υπηρεσίας (Service Level Agreement - SLA) μεταξύ του χρήστη και του παρόχου. Παρότι το μοντέλο αυτό παρέχει σταθερότητα, δεν επιτρέπει υψηλού επιπέδου διαμόρφωση της κίνησης. Παράλληλα οι Voice over Ip (VoIP) ροές μαζί με τις ροές κίνησης απαιτούν παράδοση σε συγκεκριμένο χρόνο και απαιτούν ειδικό χειρισμό. Η αρχιτεκτονική των αποφάσεων ανά κόμβο (hop), είναι δύσκολο να παρατηρηθεί από άκρο σε άκρο, λόγω και των πολλών διαφορετικών λογισμικών που τρέχουν σε κάθε συσκευή. Δεν υπάρχει προτυποποιημένος τρόπος σήμανσης της κίνησης με προτεραιότητα και καθορισμού των πολιτικών και των περιορισμών, σε συνδυασμό με τον διαχωρισμό της κίνησης.

5.2 Παραδοσιακές Προσεγγίσεις QoS

Οι τεχνικές για την εφαρμογή της Ποιότητας της Υπηρεσίας μπορούν να διαχωριστούν σε δύο μεγάλες κατηγορίες. Στις λεγόμενες παραδοσιακές όπου κατατάσσονται προσεγγίσεις προτού τα SDN εμφανιστούν ως τεχνολογία και στις βασιζόμενες σε SDN QoS λύσεις.

Στην εποχή πριν τα SDN, έχουν επικρατήσει και προτυποποιηθεί δύο υποκατηγορίες. Η πρώτη των Ολοκληρωμένων Υπηρεσιών (Integrated Services – Int Serv) είναι μια καλοσχεδιασμένη και λεπτεπίλεπτη αρχιτεκτονική ελέγχου ανά ροή κίνησης. Η λειτουργικότητα της βασίζεται στη λογική ότι κάθε δικτυακή συσκευή ξεχωριστά (μεταγωγοί, δρομολογητές) πρέπει να διατηρεί και να δεσμεύει πόρους για κάθε ροή. Ο στόχος αυτός στην σημερινή δομή του διαδικτύου είναι δύσκολο να επιτευχθεί. Αρχικά οι πόροι (επεξεργαστική ισχύς, μνήμη ram) των δικτυακών συσκευών είναι πεπερασμένοι και απαγορεύουν την ταξινόμηση όλων των πιθανών ροών στη συσκευή. Επίσης η προσέγγιση αυτή, δεν έχει δυνατότητα κλιμάκωσης καθώς κάθε δρομολογητής στο μονοπάτι της ροής πρέπει να υποστηρίζει τις Integrated Services και να αποθηκεύει όλη την πληροφορία και τις πιθανές καταστάσεις των διαφορετικών ροών. Ο στόχος αυτός είναι δύσκολο να επιτευχθεί και έτσι η μέθοδος αυτή βρίσκει εφαρμογή σε μικρότερα δίκτυα.

Η δεύτερη γνωστή ως Διαχωρισμός Υπηρεσιών (Differentiated Services – Diff-Serv) είναι μια πιο «χοντροκομμένη» αρχιτεκτονική ελέγχου κίνησης. Βασίζεται στο πεδίο DS μεγέθους 8-bit (στην θέση του παλαιότερου Type of Service - TOS field ή byte) στην κεφαλίδα του πρωτοκόλλου IP (ip header). Το πεδίο αυτό υποστηρίζει 64 διαφορετικές κλάσεις κίνησης. Στη συνέχεια οι συμβατοί με το Diff-Serv routers αποφασίζουν για τη δρομολόγηση των πακέτων σύμφωνα με την κλάση αυτή. Ενώ η μέθοδος αυτή έχει εφαρμογή σε πολύ μεγάλα δίκτυα, καθώς μόνο μια σταθερά 64 κλάσεων πρέπει να διαφοροποιηθεί, είναι μια στατική προσέγγιση και δεν παρέχει τη δυνατότητα για λεπτομερή ρύθμιση της QoS σε διαφορετικές ροές. Ακόμα η ταξινόμηση της κίνησης και ο καθορισμός των πολιτικών γίνεται εντός του

αυτόνομου συστήματος (Autonomous System - AS). Αυτό σημαίνει ότι δεν μπορεί να υπάρξει εγγύηση από άκρο σε άκρο, παρά μόνο στην εμβέλεια της DiffServ περιοχής. Για να επιλυθεί το ζήτημα αυτό υπάρχει η πρόταση για την ύπαρξη ενός μεσάζοντα διαχειριστή εύρους ζώνης (Bandwidth Broker), ο οποίος έχει γνώση των ενεργών πολιτικών. Έτσι για τη διασφάλιση της από άκρο σε άκρο σωστής υπηρεσίας οι μεσάζοντες επικοινωνούν μεταξύ τους. Η προσέγγιση αυτή έρχεται πιο κοντά στη δομή των SDN , καθώς μεταφέρει τη λογική πιο κεντρικά στο δίκτυο. Μάλιστα αρκετές από τις προτεινόμενες ολοκληρωμένες λύσεις , επεκτείνουν την ιδέα της DiffServ στα SDN. Όμως λόγω της έλλειψης κεντρικού ελέγχου στη δομή του διαδικτύου, η DiffServ είναι αδύνατο να εφαρμοστεί παγκοσμίως καθώς δεν υπάρχει πρότυπο συνεργασίας των δρομολογητών που ανήκουν σε διαφορετικές εταιρίες – προμηθευτές.

5.2 SDN QoS Frameworks

Σε αντίθεση με τις προαναφερθείσες παραδοσιακές μεθόδους, οι βασισμένες στα SDN προσεγγίσεις αντιμετωπίζουν τα προβλήματα έχουν περιγραφεί. Λόγω των δυνατοτήτων που παρέχει ο SDN Controller , οι πολιτικές μπορούν να οριστούν σε όλο το δίκτυο χωρίς να υπάρχει η ανάγκη χαμηλού επιπέδου ρυθμίσεων σε κάθε συσκευή προώθησης ξεχωριστά. Το σύνολο το πολιτικών και το διαφορετικών κλάσεων για τις ροές δεν έχει όριο, επιτρέποντας την λεπτομερή ρύθμιση με βάση της ανάγκες του χρήστη ή της υπηρεσίας. Οι κανόνες μπορούν να οριστούν ανά ροή και ο Controller είναι υπεύθυνος να τις εφαρμόσει στα διαφορετικά δικτυακά στοιχεία.

Η πιο κοινή μέθοδος εφαρμογής των κανόνων είναι μέσω ενός είδους εικονικής τμηματοποίησης του εύρους ζώνης δικτύου (bandwidth slicing). Η μέθοδος αυτή μπορεί να κατηγοριοποιηθεί ως διαφύλαξη πόρων (resource reservation), όπου σε κάθε ροή ανατίθεται ένα κομμάτι της συνολικής χωρητικότητας μετάδοσης. Σε αντίθεση με τη λογική αυτή υπάρχουν συστήματα που εφαρμόζουν τη δυναμική δρομολόγηση ανά ροή, κατά την οποία δεν υπάρχει απευθείας ανάθεση πόρων στις ροές. Ακόμα υπάρχουν συστήματα τα οποία δίνουν ειδική έμφαση στην τοποθέτηση των πακέτων σε ουρές αλλά και στην επιβολή των QoS πολιτικών. Αρκετά από τα συστήματα αυτά απαιτούν κάποιου είδους δειγματοληψίας δεδομένων ώστε να διαχειρίζονται το δίκτυο. Αν υπάρχει δειγματοληψία για όλους τους κόμβους του δικτύου, πρέπει να ληφθεί υπό όψιν το εύρος ζώνης που δεσμεύεται για αυτό το σκοπό. Κάθε προσέγγιση έχει τα δικά της πλεονεκτήματα και μειονεκτήματα. Στο παρακάτω κομμάτι ακολουθεί ανάλυση των πιο γνωστών μεθόδων εφαρμογής QoS σε SDN λύσεις.

5.2.1 Συστήματα διαφύλαξης πόρων

Τα συστήματα διαφύλαξης πόρων (Resource Reservation Networks) αποτελούν μία από τις πιο κοινές λύσεις στην εφαρμογή QoS στα SDN δίκτυα. Έχουν υλοποιηθεί αρκετά συστήματα τα οποία εμπίπτουν στη κατηγορία αυτή. Η δομή τους συνήθως περιλαμβάνει ένα κομμάτι ταξινόμησης ροών και ένα διαμορφωτή ρυθμού βασισμένο στα SDN. Οι ταξινομητές διαβάζουν τα πεδία των πακέτων και επιχειρούν να αναθέσουν συγκεκριμένη προτεραιότητα στις εκάστοτε ροές βασισμένη στις πολιτικές που έχουν οριστεί στον διαχειριστή. Οι διαμορφωτές κίνησης από την πλευρά τους, εγκαθιστούν κανόνες διαφύλαξης πόρων ,στα συμβατά με το OpenFlow switches, σύμφωνα με την ταξινόμηση τους.

Το FlowQoS είναι ένα από αυτά τα συστήματα. Απευθύνεται σε μικρά δίκτυα και αξιοποιεί τα SDN για να παραμετροποιήσει οικιακούς δρομολογητές, σύμφωνα με πολιτικές διαμόρφωσης του ρυθμού της κίνησης που έχει ορίσει ο χρήστης.

Άλλο ένα σύστημα που αξιοποιεί παρόμοιες μεθοδολογίες είναι ένα τμήμα του έργου EuQoS. Αποτελεί λύση για μεγάλα δίκτυα και δίνει έμφαση στην παροχή QoS μέσω ροών σε εταιρικούς πελάτες μετά από αίτηση τους. Η ταξινόμηση και ο διαχωρισμός γίνονται μέσω του DS πεδίου και της ip διεύθυνσης προορισμού γίνεται η ταξινόμηση. Αυτή η προσέγγιση πλησιάζει αρκετά την παραδοσιακή DiffServ αλλά διαχωρίζει και τους κατόχους των ροών μέσω της ip διεύθυνσης τους.

5.2.2 Συστήματα δρομολόγησης ανά ροή

Η ιδέα των συστημάτων δρομολόγησης ανά ροή (Per-flow Routing Frameworks), στηρίζεται στον διαχωρισμό των ροών δεδομένων με τις ροές πολυμέσων. Παρόμοια και με την προηγούμενη κατηγορία, χρησιμοποιείται ταξινομητής. Όμως αντί να δεσμεύονται πόροι (με τη μορφή των bandwidth slices) σε κάθε συσκευή στο επίπεδο δεδομένων, ο διαχειριστής δυναμικά τοποθετεί υψηλής προτεραιότητας ροές σε δρόμους όπου μπορούν να εγγυηθούν Ποιότητα της Υπηρεσίας. Αυτή η τεχνική αξιοποιεί τη δυναμική δρομολόγηση. Παράλληλα οι κλασικές ροές δεδομένων δεν επηρεάζονται και ακολουθούν την ίδια δρομολόγηση. Εξετάζονται περιορισμοί όπως η καθυστέρηση μιας ροής πολυμέσων. Για να βρεθεί το μονοπάτι που μπορεί να διατηρήσει την QoS, η συμφόρηση του δικτύου εξετάζεται. Για παράδειγμα στο OpenQoS ένας σύνδεσμος θεωρείται πως έχει συμφόρηση αν η κίνηση ξεπερνά το 70% της χωρητικότητας του. Το μεγαλύτερο πλεονέκτημα σε σχέση με τη δέσμευση πόρων είναι η μειωμένη καθυστέρηση και απώλεια πακέτων για τις ροές που δεν ανήκουν σε QoS.

Αντιπροσωπευτικό παράδειγμα των συστημάτων αυτών είναι το OpenQoS. Παρέχει εγγύηση για την παράδοση κίνησης εφαρμογών και εστιάζει κυρίως στις

ροές πολυμέσων όπως η κίνηση VoIP και τα video streams. Το σύστημα βασίζεται στον Floodlight Controller.

5.2.3 Συστήματα με έμφαση στην διαχείριση ουρών και τον προγραμματισμό πακέτων

Το ζήτημα του προγραμματισμού πακέτων (packet scheduling) και της τοποθέτησης σε ουρές είναι σημαντικό. Στο πρότυπο OpenFlow 1.3 ορίζεται τοποθέτηση σε ουρές μέσω FIFO (First In First Out) των εισερχομένων πακέτων. Ο σχεδιασμός αυτός μπορεί να δημιουργήσει πρόβλημα σε κάποιες ροές, όπως ροές πολυμέσων, αν το δίκτυο έχει συμφόρηση. Για την αντιμετώπιση του ζητήματος αυτού έχει προταθεί στο σύστημα QosFlow, που στηρίζεται στην επανατοποθέτηση των πακέτων σε μια δεδομένη ουρά. Επιχειρεί να συστήσει τις δυνατότητες ελέγχου των Linux στα SDN δίκτυα. Αξιοποιεί διάφορους μηχανισμούς, με τον SFQ (Stochastic Fairness Queuing) να είναι από τους πιο αξιοσημείωτους. Κάθε ροή υψηλής προτεραιότητας εισάγεται σε ένα κουβά. Σε μετέπειτα στάδιο , με χρήση του αλγορίθμου round robin, τα πακέτα των ροών προωθούνται σε μία προσπάθεια διατήρησης της ισότητας.

Βέβαια υπάρχουν περιοριστικοί παράγοντες όπως ο μέγιστος αριθμός ροών ανά πόρτα μεταγωγού. Παρότι ο περιορισμός αυτός προέρχεται από το OpenFlow πρωτόκολλο και όχι από το σύστημα διαχείρισης QoS, πρέπει να ληφθεί υπό όψιν. Άλλο ένα μειονέκτημα είναι ότι απευθύνεται μόνο σε μηχανήματα τα οποία έχουν λειτουργικό Linux. Έτσι αυτή η λύση δε μπορεί να εφαρμοστεί στην τωρινή μορφή του διαδικτύου.

5.2.4 Συστήματα επιβολής πολιτικών

Ένα από τα μεγαλύτερα ζητήματα των προκαθορισμένων SLAs, από τη σκοπιά της QoS, είναι ότι με τη δεδομένη αρχιτεκτονική του διαδικτύου δεν υπάρχει ένας προτυποποιημένος και ευέλικτος τρόπος εφαρμογής τους. Οι περισσότερες από τις νέες τεχνολογίες που στοχεύουν στην διαφύλαξη των SLAs πάνω από το δίκτυο, είναι ιδιόκτητες.

Μέσω των SDN και του Controller είναι εφικτό να οριστούν δυναμικά SLAs και να τοποθετηθούν στο επίπεδο προώθησης μέσω των southbound APIs. Μία από τη προτεινόμενες λύσεις είναι το PolicyCop, βασισμένο στον Floodlight Controller.

Διαφέρει από τις μεθοδολογίες που χρησιμοποιούν το DiffServ καθώς δεν περιορίζεται από τις κλάσεις κίνησης και την χοντροκομμένη προσέγγιση στις λεπτομέρειες που τις ακολουθούν. Το σύστημα μπορεί να επεκταθεί εύκολα αφού έχει αρχιτεκτονική διαστρωμάτωσης. Κάθε επίπεδο επικοινωνεί με τα υπόλοιπα μέσω RESTful JSON APIs, οπότε υπάρχει και δυνατότητα προσθήκης στρωμάτων ή αντικατάστασης των υπαρχόντων. Ακόμα είναι εφικτό να δημιουργηθούν νέες

κλάσεις ροών, σε αντίθεση με τη στατική δομή του μοντέλου DiffServ. Άλλο είναι πλεονέκτημα είναι η ευελιξία αφού δεν υπάρχει δέσμευση με συγκεκριμένα συστήματα , εκτός από το OpenFlow.

ΕΙΔΙΚΟ ΜΕΡΟΣ

Στα πλαίσια της εργασίας δημιουργήθηκε εφαρμογή σε γλώσσα προγραμματισμού JAVA. Η εφαρμογή αλληλεπιδρά με τον OpenDaylight Controller μέσω του Restconf API. Μέσω γραφικού περιβάλλοντος υποστηρίζει τον έλεγχο και τη διαχείριση δικτύων που συνδέονται με τον OpenDaylight Controller, την εξαγωγή πληροφοριών του δικτύου σε αρχείο excel, την εισαγωγή και διαγραφή ροών, την εφαρμογή πολιτικών Quality of Service και την εισαγωγή monitors που μπορούν να εφαρμόσουν QoS μετά από κάποιο συμβάν (όπως συμφόρηση συνδέσμου).

Η ανάπτυξη έγινε σε λειτουργικό σύστημα Ubuntu Linux, όπου είχε εγκατασταθεί τοπικά ο OpenDaylight Controller, ο εξομοιωτής δικτύων Mininet και το NetBeans IDE. Ακόμα χρησιμοποιήθηκε το εργαλείο Postman για δοκιμές μέσω http requests προς το Restconf API.

ΚΕΦΑΛΑΙΟ 1 – Περιβάλλον Ανάπτυξης

1.1 Απαιτούμενα συστήματα και εγκατάσταση

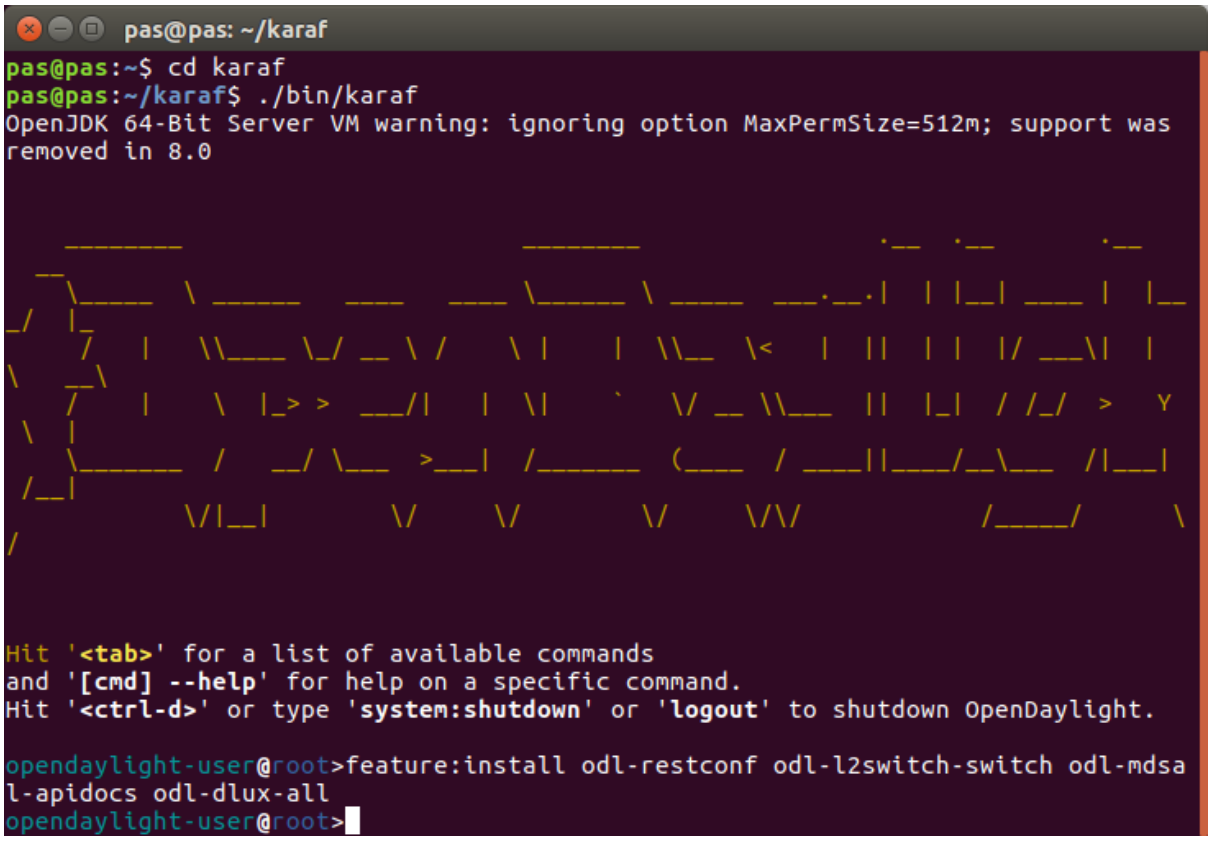
Τα δύο βασικότερα συστήματα για το περιβάλλον πειραματισμού και ανάπτυξης είναι ο OpenDaylight Controller και ο εξομοιωτής Mininet. Υπάρχουν αρκετά Virtual Machines που περιέχουν εγκατεστημένα τα παραπάνω προγράμματα. Στα πλαίσια της εργασίας προτιμήθηκε τοπική εγκατάσταση σε υπολογιστή με λειτουργικό σύστημα Ubuntu 16.04, για λόγους ταχύτητας και καλύτερης απόδοσης. Αξίζει να σημειωθεί ότι η χρήση λειτουργικού συστήματος Linux είναι υποχρεωτική ώστε να λειτουργήσει ο Mininet Emulator. Ακόμα ο ODL ως Java Framework , προϋποθέτει την ύπαρξη εγκατεστημένης έκδοσης Java στο μηχάνημα που θα εκτελεστεί αλλά και τον ορισμό της μεταβλητής περιβάλλοντος JAVA_HOME.

1.2 OpenDaylight Controller

Ο OpenDaylight Controller βρίσκεται πακεταρισμένος σε ένα karaf container. Η τεχνολογία αυτή επιτρέπει την γρήγορη εγκατάσταση και αλλαγή μεταξύ εκδόσεων καθώς όλα τα απαραίτητα προγράμματα βρίσκονται σε ένα φάκελο. Μπορούμε να κατεβάσουμε τις τελευταίες εκδόσεις από την επίσημη σελίδα του ODL https://www.opendaylight.org/technical-community/getting-started-for-developers/downloads-and-documentation αλλά και παλαιότερες στο https://nexus.opendaylight.org/content/repositories/opendaylight.release/org/opendaylight/integration/distribution-karaf/ . Στα πλαίσια της εργασίας χρησιμοποιήθηκε η

έκδοση Beryllium 0.4.0. Εφόσον αποσυμπιέσουμε το αρχείο που κατεβάσαμε αρκεί να τρέξουμε σε τερματικό παράθυρο την εντολή ***./bin/karaf*** μέσα στον φάκελο που βρίσκεται το karaf distribution.

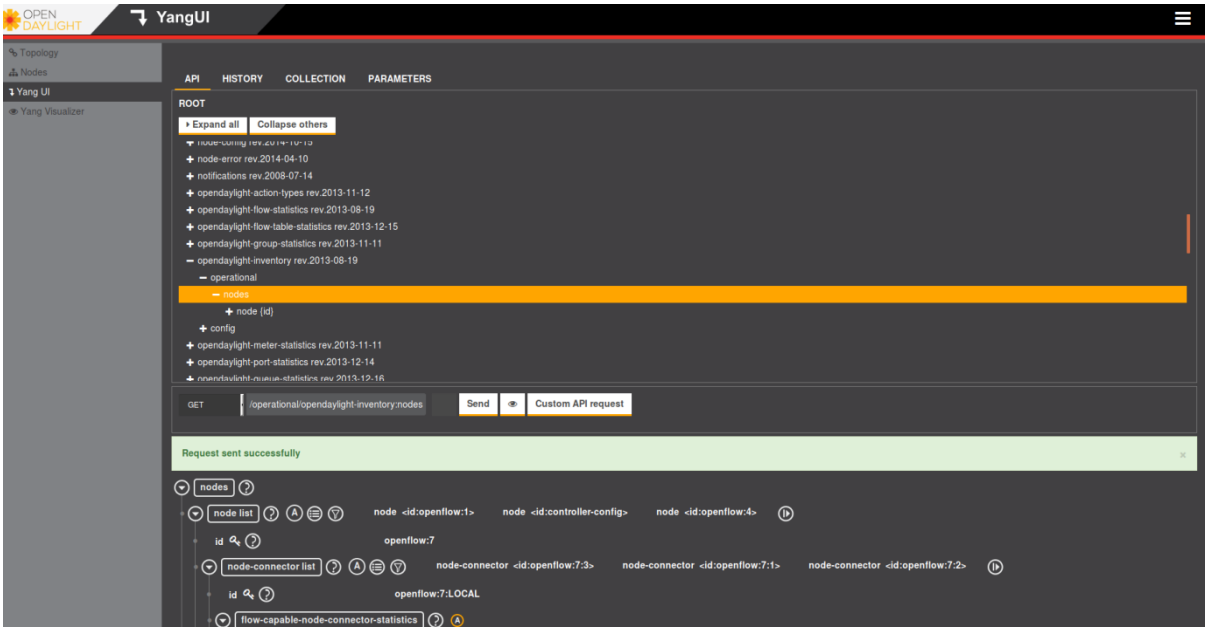
Μετά την εγκατάσταση μπορούμε να προσθέσουμε χρήσιμα components μέσω της κονσόλας karaf τα οποία αποκαλούνται features.



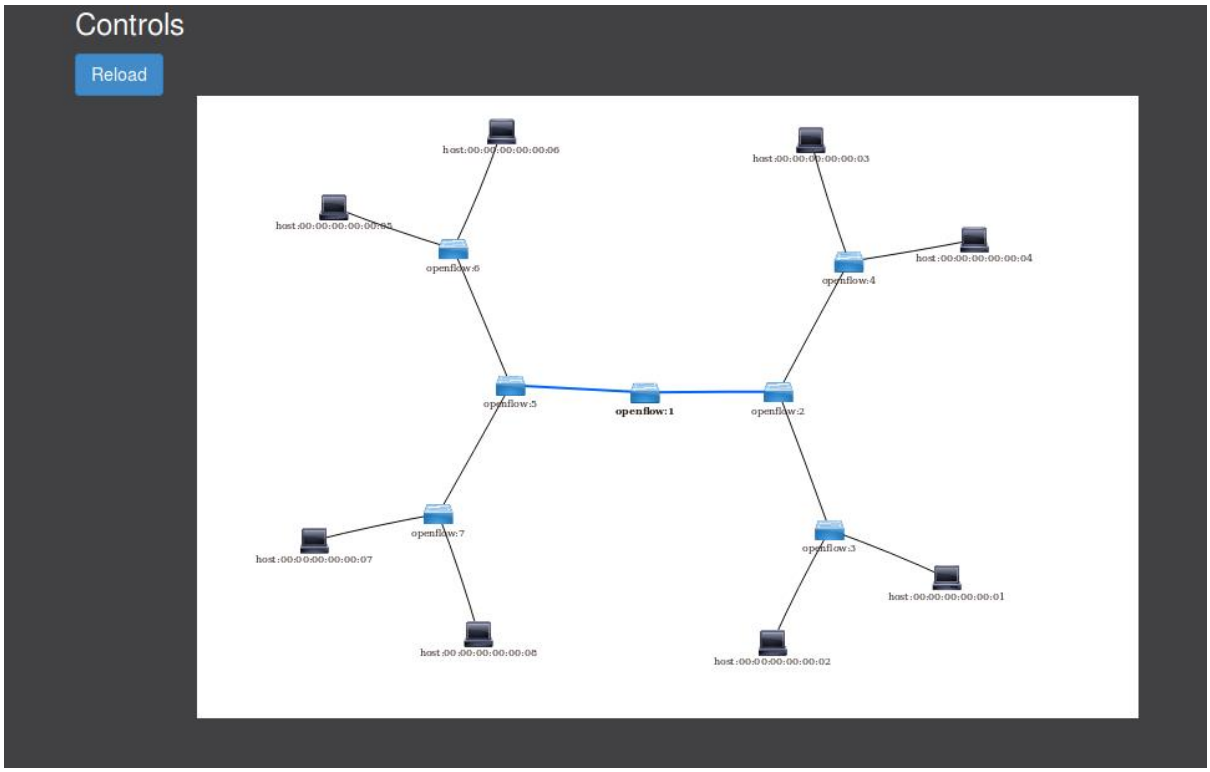
Για παράδειγμα το ***odl-restconf*** μας δίνει πρόσβαση στο Restconf API.

Το ***odl-l2switch-switch*** δίνει λειτουργικότητα στα SDN switches παρόμοια με αυτή ενός τυπικό layer 2 ethernet switch. Χωρίς την εγκατάσταση του συγκεκριμένου feature δε δρομολογείται κανένα πακέτο καθώς δεν υπάρχει κανένας κανόνας στα switch.

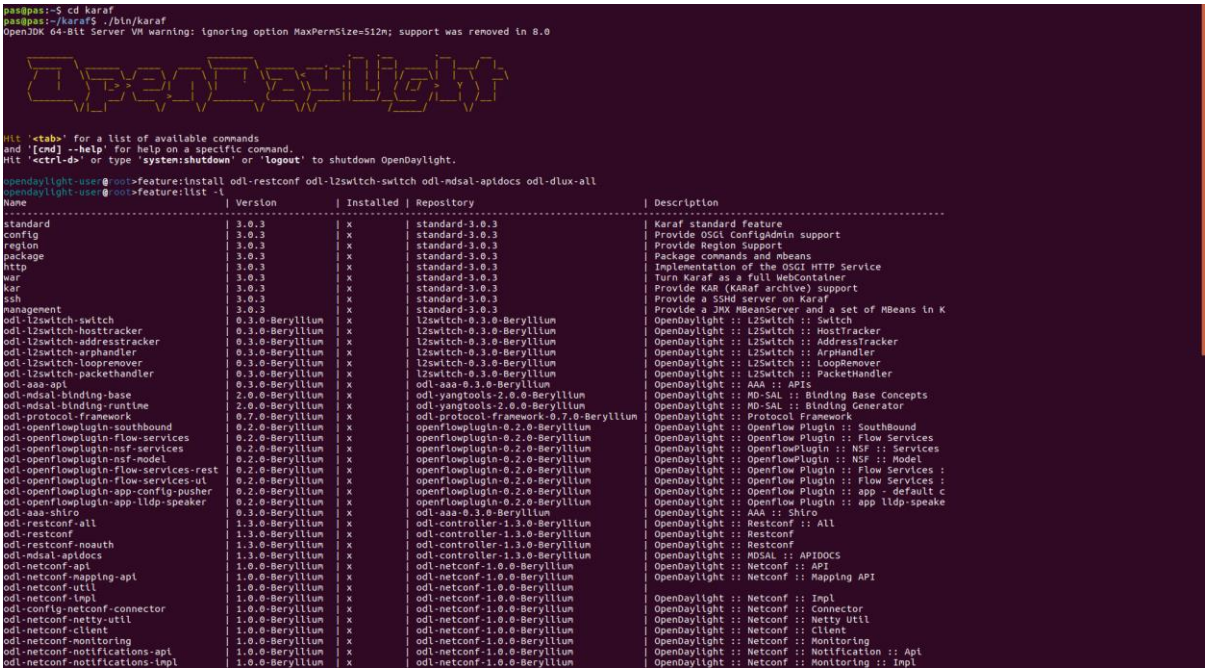
Μέσω του Yang API μπορούμε να δούμε τα υπάρχοντα models στον controller και να πάρουμε πληροφορίες για το δίκτυο μέσω γραφικού περιβάλλοντος. Το αντίστοιχο feature είναι το ***odl-mdsal-apidocs***.



Τέλος το DLUX API (***odl-dlux-all***) παρέχει γραφικό περιβάλλον για να δούμε στοιχεία του δικτύου, όπως την τοπολογία. Στο παρακάτω παράδειγμα βλέπουμε μία default τοπολογία δέντρου με ύψος τρία, όπου στο τελευταίο επίπεδο συνδέονται δύο τερματικές συσκευές σε κάθε switch.



Με την εντολή ***feature:list*** βλέπουμε όλα τα υπάρχοντα features που μπορούν να εγκατασταθούν ενώ με την ***feature:list –installed*** αναγράφονται τα εγκατεστημένα.



1.3 Mininet

Το Mininet είναι το εργαλείο που χρησιμοποιήθηκε για την δημιουργία ενός εικονικού δικτύου. Κατά την εντολή αρχικοποίησης από τερματικό παράθυρο δίνονται οι παράμετροι για το δίκτυο που θα δημιουργηθεί.

Για παράδειγμα στην παρακάτω εντολή ορίζεται η ip του controller (localhost λόγω τοπικής εγκατάστασης), τοπολογία δέντρου με ύψος τρία και η έκδοση του πρωτοκόλλου (1.3) OpenFlow που θα τρέχει στα switches. Εφόσον ορίζουμε τον controller κατά τη δημιουργία, ο OpenDaylight θα πρέπει να έχει ήδη ξεκινήσει τη λειτουργία του.

sudo mn --controller=remote,ip=127.0.0.1 --topo tree,3 --switch ovsk,protocols=OpenFlow13

```
pas@pas: ~
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6)
 (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['311 Mbits/sec', '318 Mbits/sec']
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['335 Mbits/sec', '344 Mbits/sec']
mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['296 Mbits/sec', '301 Mbits/sec']
mininet> xterm h1 h2
mininet>
```

Παρατηρούμε την δημιουργία των συνδέσμων, κόμβων και τερματικών του δικτύου. Με την εντολή ***pingall*** ελέγχεται η συνδεσιμότητα από και προς κάθε σημείο του δικτύου. Χρήσιμη εντολή, από το Mininet cli, για τον έλεγχο της χωρητικότητας των συνδέσμων είναι η ***iperf.*** Ακόμα υπάρχει δυνατότητα να ανοίξουμε τερματικό παράθυρο στους host με την εντολή ***xterm***. Μέσα από το τερματικό παράθυρο έχουμε όλες τις επιλογές που θα είχαμε σε ένα κανονικό υπολογιστή. Από τον έλεγχο ping μέχρι τη δημιουργία iperf server για τη δημιουργία κίνησης και πειραματισμό.

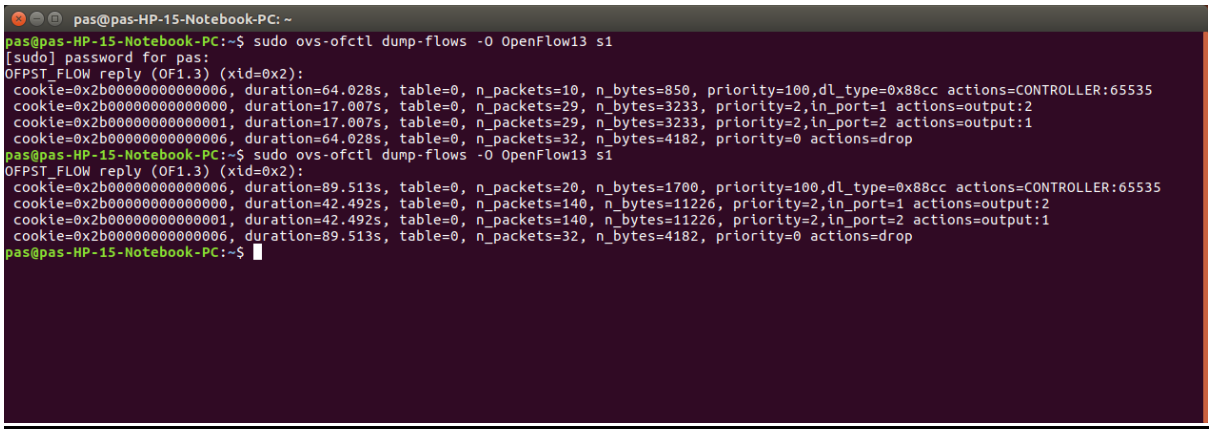
```
"Node: h2"
root@pas:~# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.573 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.314 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.314 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.303 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.274 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.325 ms
^C
--- 10.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5111ms
rtt min/avg/max/mdev = 0.274/0.350/0.573/0.102 ms
root@pas:~#
```

Είναι καλή πρακτική πριν τη δημιουργία καινούργιου δικτύου ή μετά την έξοδο από το Mininet cli να καθαρίζουμε την τοπολογία με την εντολή ***sudo mn –c***.

1.4 Openflow Command Line Tools

Τα switches που δημιουργούνται από το Mininet είναι εφικτό να παραμετροποιηθούν μέσω κονσόλας τερματικού. Με χρήση του command line εργαλείου ***ovs-ofctl*** και του ***ovs-vsctl*** utility έχουμε πλήρη δυνατότητα ελέγχου στα Open vSwitch. Το λεπτομερές manual μπορεί να βρεθεί στους παρακάτω συνδέσμους http://openvswitch.org/support/dist-docs/ovs-vsctl.8.txt και http://openvswitch.org/support/dist-docs/ovs-ofctl.8.txt. Ακολουθούν μερικές από τις εντολές που χρησιμοποιήθηκαν στα πλαίσια της εργασίας.

Με την παρακάτω εντολή μπορούμε να δούμε τις ροές που τρέχουν σε κάποιο switch ***sudo ovs-ofctl dump-flows*** με απαραίτητο όρισμα την OpenFlow έκδοση και το όνομα του switch. Π.χ. ***sudo ovs-ofctl dump-flows -O OpenFlow13 s1***.



Η διαγραφή των πολιτικών Quality of Service και των ουρών γίνεται με την παρακάτω εντολή: ***sudo ovs-vsctl -- --all destroy QoS -- --all destroy Queue***. Οι υπάρχουσες πολιτικές Quality of Service εμφανίζονται με την εντολή: ***sudo ovs-vsctl list qos*** ενώ οι υπάρχουσες ουρές με την εντολή: ***sudo ovs-vsctl list queue***. Ακόμα υπάρχει δυνατότητα εφαρμογής ποιότητας της υπηρεσίας και εισαγωγής ουρών στην εξερχόμενη κίνηση ενός switch interface. Ακολουθεί παράδειγμα όπου δημιουργείται QoS τύπου Tocken Bucket με μέγιστο ρυθμό μεταφοράς δεδομένων 100 Mbps και δύο ουρές, στην q1 min data rate = max data rate = 20mb/s, στην q2 min data rate = max data rate = 50mb/s.

sudo ovs-vsctl set port s1-eth1 qos=@min_bw -- --id=@min_bw create qos type=linux-htb other-config:max-rate=100000000 queues=1=@min_20mb,2=@min_50mb -- --id=@min_20mb create queue other-config:min-rate=20000000 other-config:max-rate=20000000 -- --id=@min_50mb create queue other-config:min-rate=50000000 other-config:max-rate=50000000

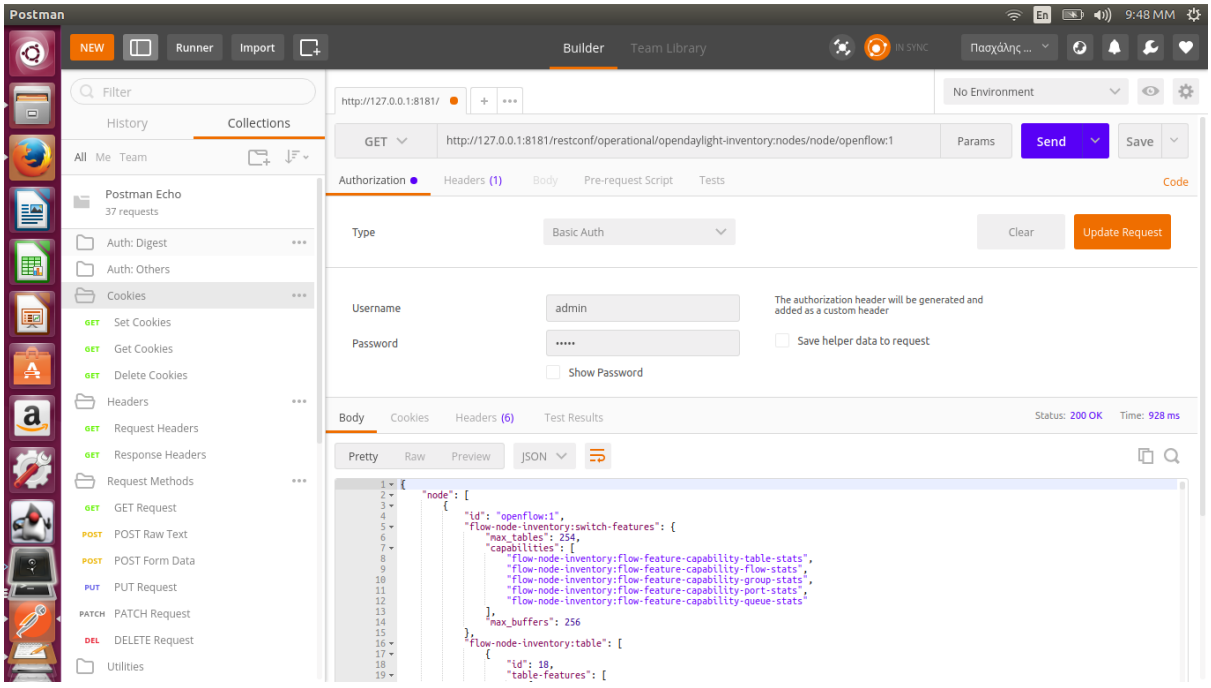
```
pas@pas-HP-15-Notebook-PC: ~
pas@pas-HP-15-Notebook-PC:~$ sudo ovs-vsctl set port s1-eth1 qos=@min_bw -- --id=@min_bw create qos type=linux-htb other-config:max-rate=10
0000000 queues=1=@min_20mb,2=@min_50mb -- --id=@min_20mb create queue other-config:min-rate=20000000 other-config:max-rate=20000000 -- --id
=@min_50mb create queue other-config:min-rate=50000000 other-config:max-rate=50000000
5fdb7e68-d69b-4b38-87d9-5cf8bdd977ee
f206605e-c5dd-42bf-b3b7-33c3946bc336
3dcff20b-fe6c-4a8a-81ae-2dc2ed774f94
pas@pas-HP-15-Notebook-PC:~$ sudo ovs-vsctl list qos
_uuid               : 5fdb7e68-d69b-4b38-87d9-5cf8bdd977ee
external_ids        : {}
other_config        : {max-rate="100000000"}
queues              : {1=f206605e-c5dd-42bf-b3b7-33c3946bc336, 2=3dcff20b-fe6c-4a8a-81ae-2dc2ed774f94}
type                : linux-htb
pas@pas-HP-15-Notebook-PC:~$ sudo ovs-vsctl list queue
_uuid               : f206605e-c5dd-42bf-b3b7-33c3946bc336
dscp                : []
external_ids        : {}
other_config        : {max-rate="20000000", min-rate="20000000"}

_uuid               : 3dcff20b-fe6c-4a8a-81ae-2dc2ed774f94
dscp                : []
external_ids        : {}
other_config        : {max-rate="50000000", min-rate="50000000"}
pas@pas-HP-15-Notebook-PC:~$
```

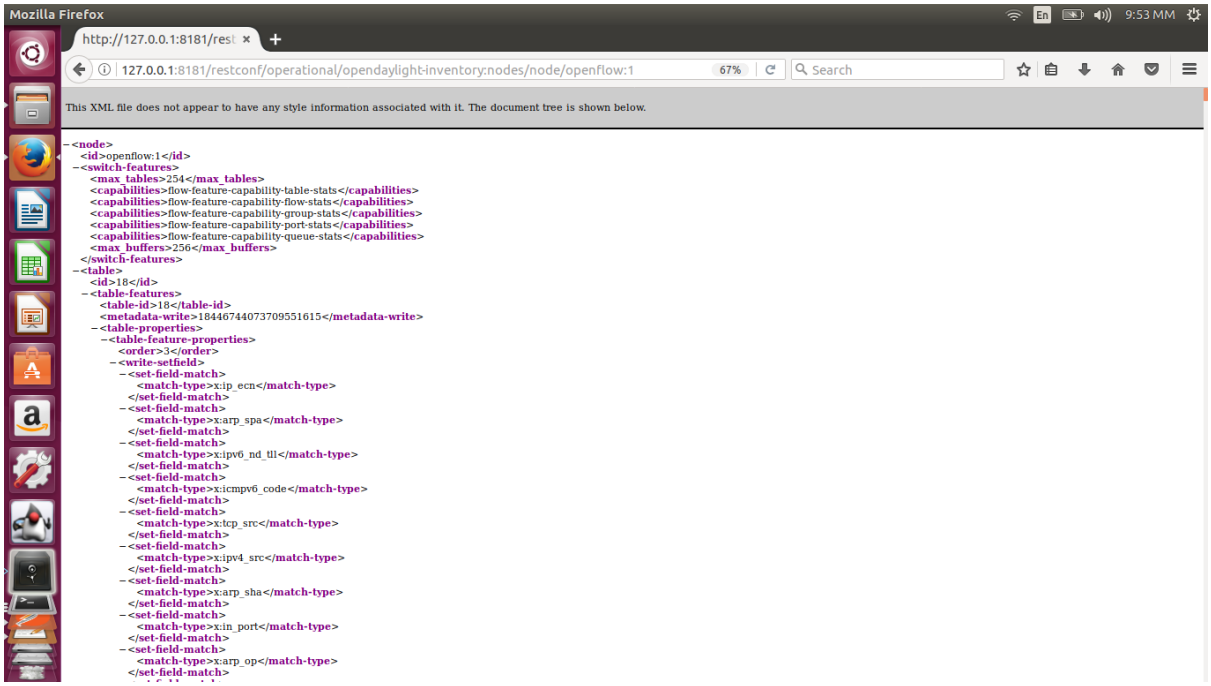
1.5 Postman

Το Postman είναι ένα πολύ χρήσιμο εργαλείο για την κατασκευή REST APIs, αλλά και για τον έλεγχο Http Requests μέσω από ένα πολύ φιλικό για το χρήστη γραφικό περιβάλλον. Καθώς στα πλαίσια της εργασίας χρησιμοποιήθηκε το Restconf API στο οποίο η επικοινωνία γίνεται μέσω http requests, το πρόγραμμα Postman βοήθησε σε μεγάλο βαθμό στον πειραματισμό των requests και μέσω των αποτελεσμάτων που λάβαμε στην τελική εφαρμογή.

Ακολουθεί παράδειγμα με get request στο Restconf Api στο url ***127.0.0.1:8181/restconf/operational/opendaylight-inventory-nodes/node/openflow:1***



Τα δεδομένα έχουν «εκτεθεί» στο url από το Restconf API και με το εργαλείο Postman εκτελούμε το get request. Το συγκεκριμένο url περιέχει πληροφορίες για τον κόμβο (switch) openflow:1



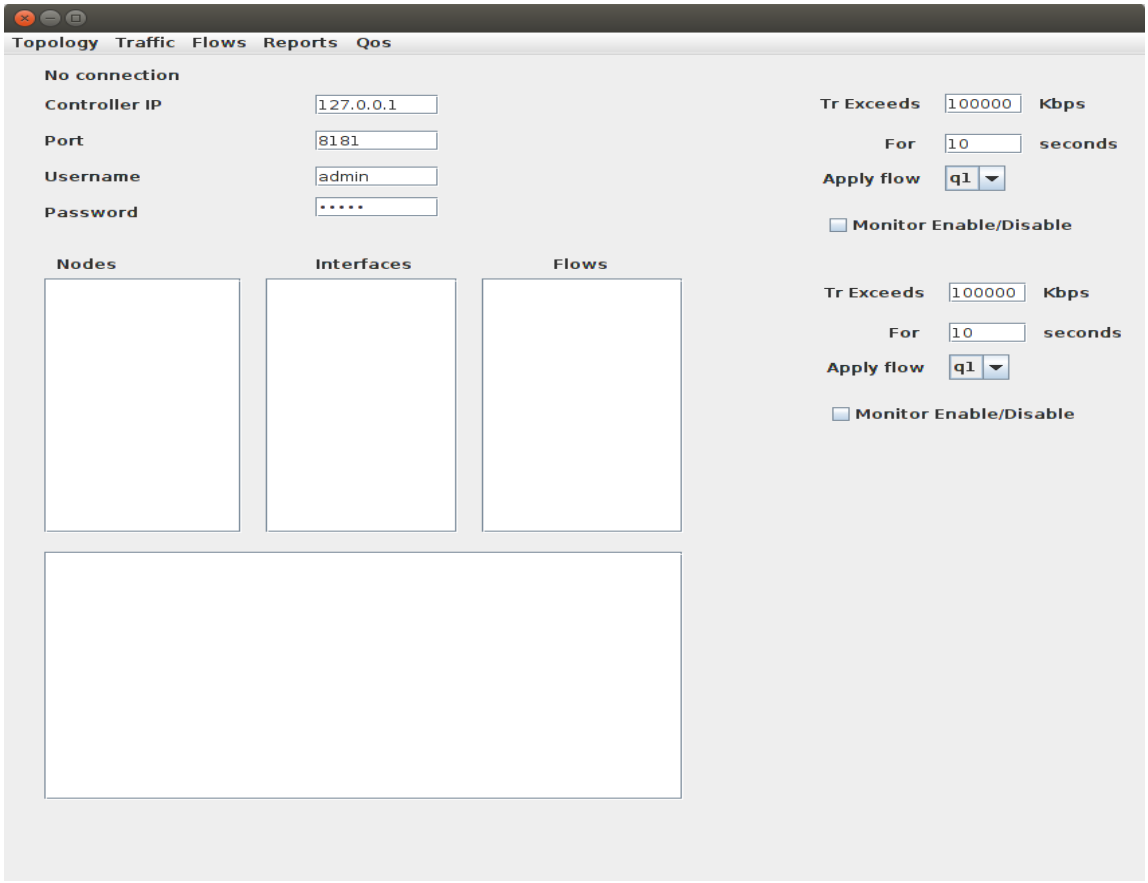
1.6 NetBeans IDE

Για την ανάπτυξη του πηγαίου κώδικα χρησιμοποιήθηκε το NetBeans 8.2 IDE (Integrated Development Environment). Μέσω του Java Swing API το γραφικό περιβάλλον της εφαρμογής δημιουργήθηκε με ευκολία και ταχύτητα.

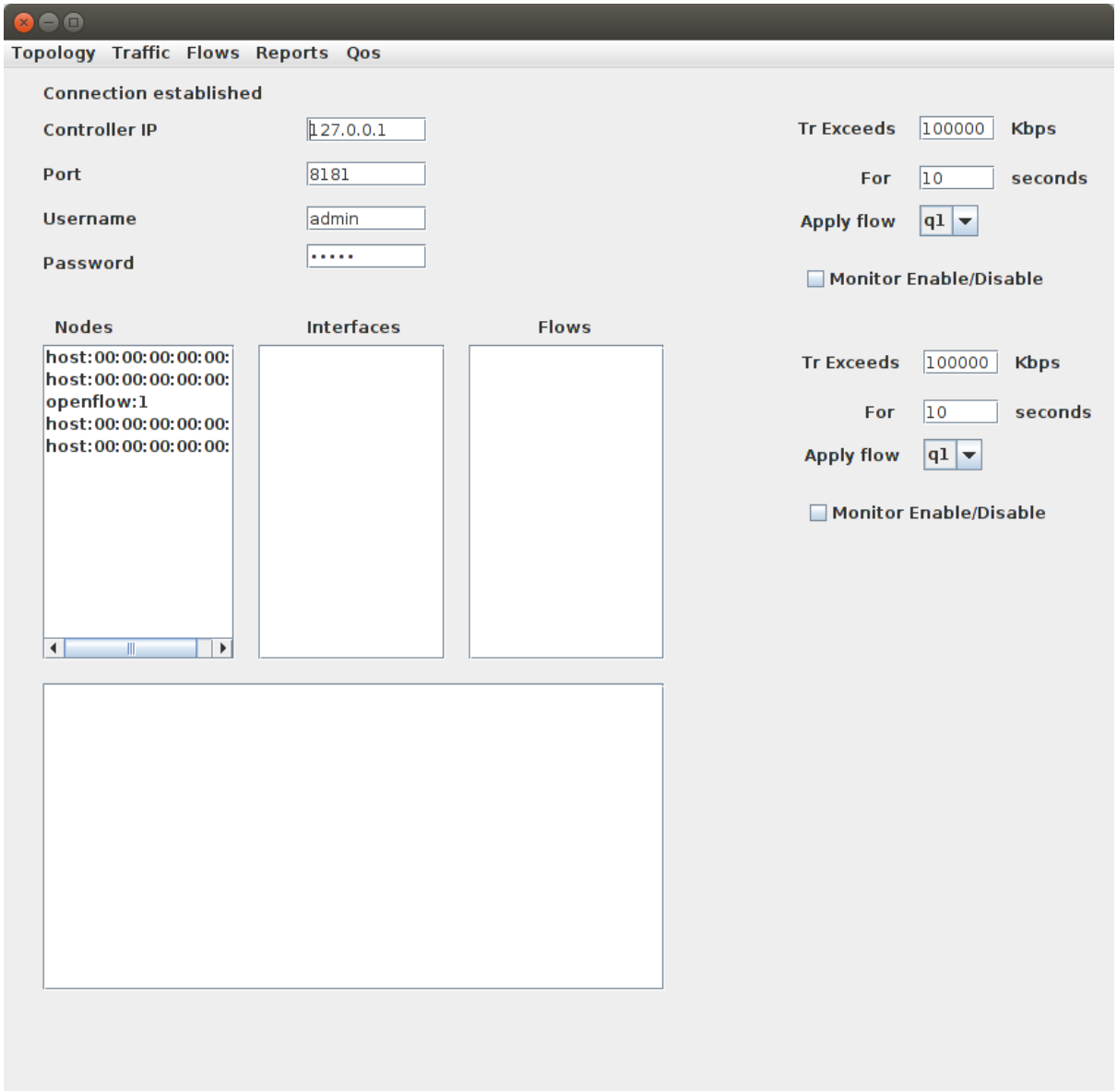
ΚΕΦΑΛΑΙΟ 2 – SDN Application

2.1 Παρουσίαση της εφαρμογής

Στα πλαίσια της εργασίας δημιουργήθηκε εφαρμογή που συνδέεται με τον SDN controller και εκτελεί πληθώρα λειτουργιών. Ο πηγαίος κώδικας βρίσκεται στο https://github.com/psiskos/java-sdn .Το γραφικό περιβάλλον είναι φτιαγμένο με το Java Swing API.

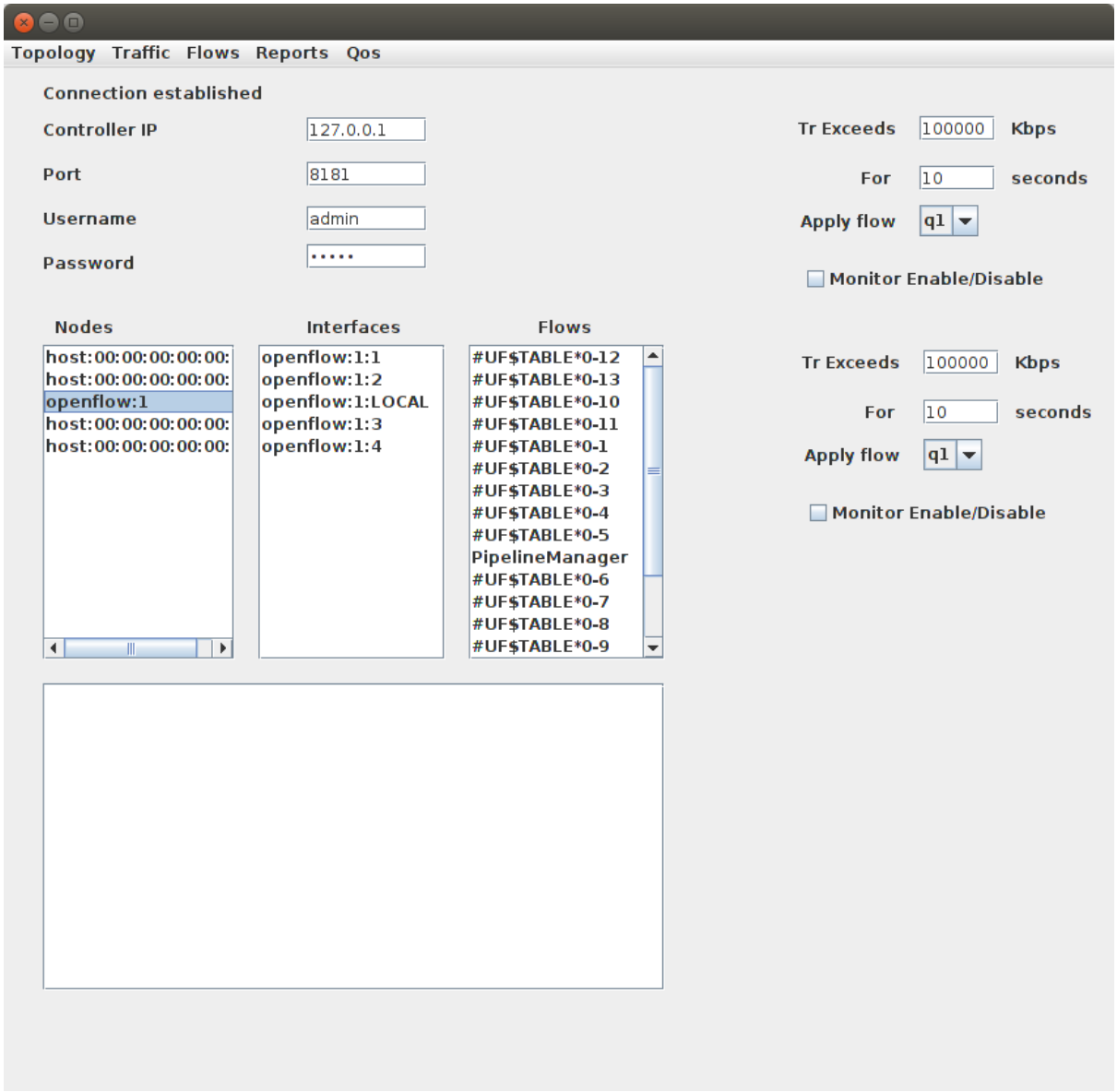


Για τη σύνδεση απαιτούνται η ip του controller, η πόρτα στην οποία τρέχει, username και password. Μόλις εισαχθούν τα παρακάτω στοιχεία ο χρήστης επιχειρεί σύνδεση πατώντας το Topology –> Connect από το menu. Αν η σύνδεση επιτευχθεί εμφανίζεται μήνυμα «Connection Established» και στην πρώτη λίστα οι hosts και τα switches που βρίσκονται συνδεδεμένα στον Controller. Μόλις πατηθεί το κουμπί γίνεται αρχικοποίηση κλάσεων και ένα http get request για την τοπολογία του δικτύου. Η απάντηση είναι ένα Json Object και μέσω parsing παίρνουμε την πληροφορία για τα switches , που αποκαλούνται nodes, και τους hosts.

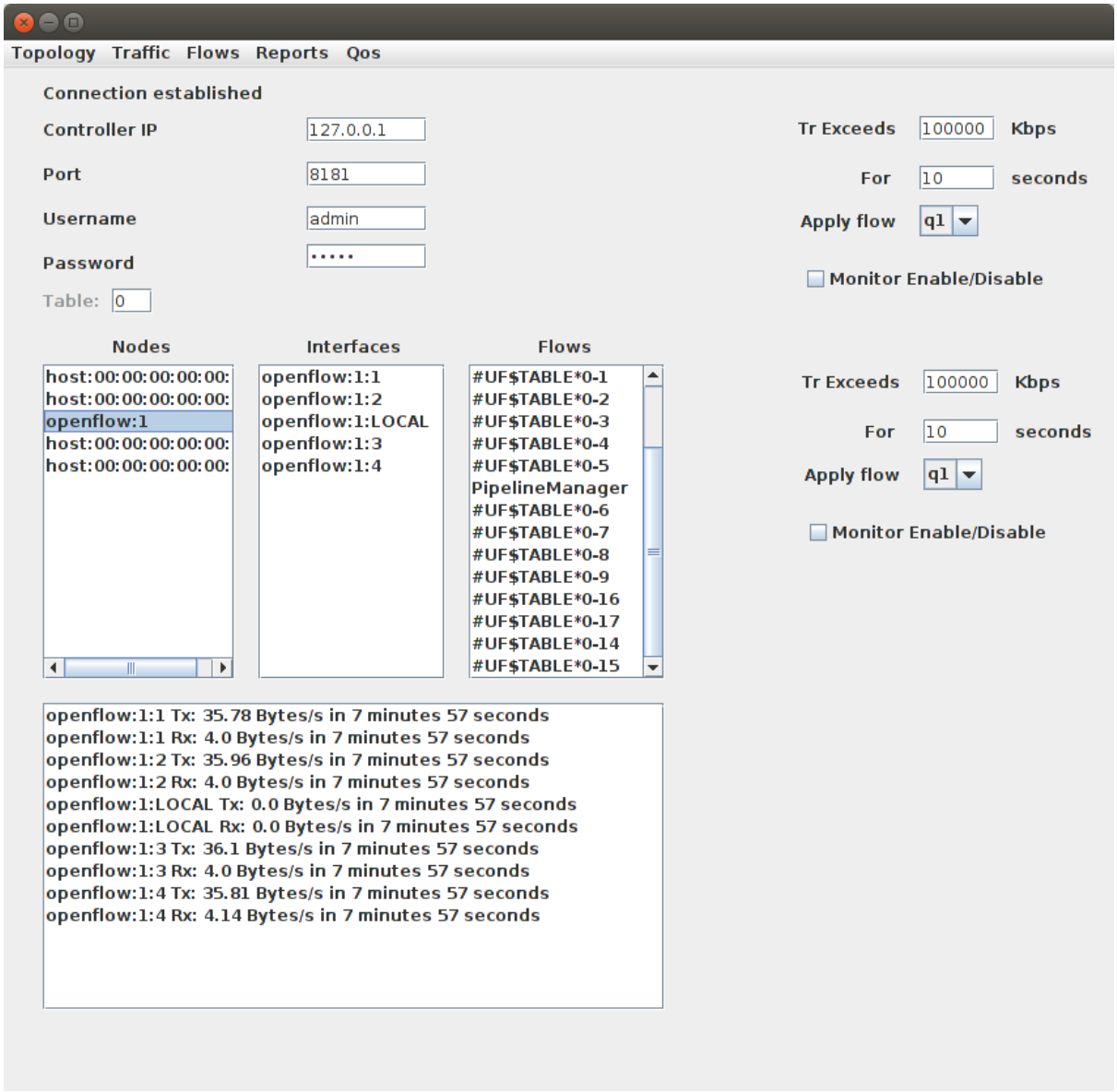


Με την επιλογή ενός host εμφανίζεται στην κάτω text area η mac address του και η ip διεύθυνση του. Με την επιλογή ενός switch εμφανίζονται οι ενεργές πόρτες του (interfaces), που αποκαλούνται node-connectors και οι ροές (flows) που υπάρχουν στο switch αυτό. Η διαδικασία είναι παρόμοια με την τοπολογία. Γίνεται get request στο url που περιέχει την πληροφορία των node-connectors για το συγκεκριμένο node. Κατόπιν γίνεται parsing στην απάντηση (json object) για την εξαγωγή της ζητούμενης πληροφορίας. Για παράδειγμα στο url υπάρχει η πληροφορία για το switch με id openflow:1

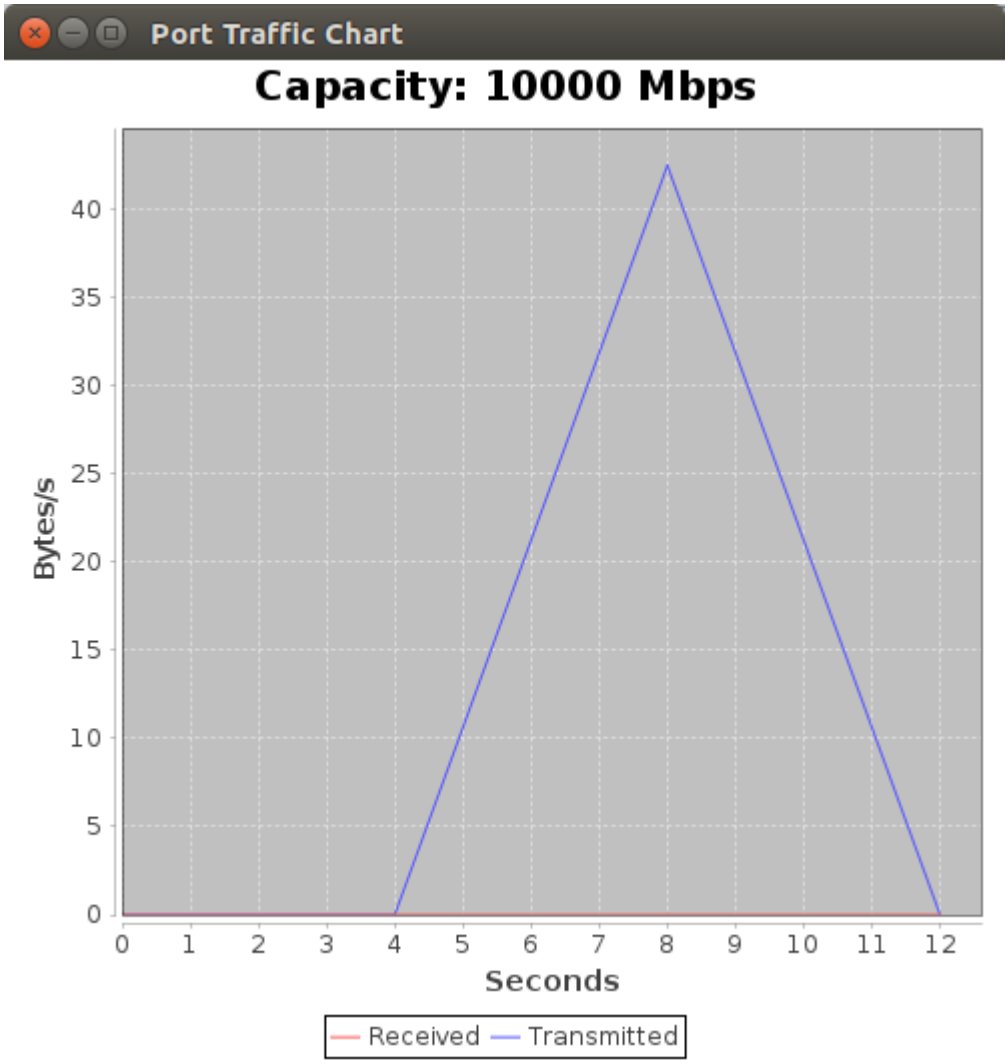
127.0.0.1:8181/restconf/operational/opendaylight-inventory-nodes/node/openflow:1



Στο Menu item Traffic υπάρχουν οι επιλογές Node Utilization και Interface Traffic. Με την επιλογή Node Utilization εμφανίζεται στην κάτω text area, ο μέσος όρος κίνησης του κάθε interface. Είναι το σύνολο των bytes που έχουν περάσει από το interface προς το χρόνο που είναι ενεργό. Στο περιβάλλον πειραματισμού , η μέτρηση γίνεται πρακτικά από τη στιγμή που το Mininet δημιουργεί το δίκτυο. Τα received και transmitted bytes/s εμφανίζονται χωριστά για κάθε πόρτα του switch. Η επιλογή αυτή παρέχει έναν εύκολο τρόπο εντοπισμού της πιο ενεργής πόρτας και ελέγχου αν κάποιο link παρουσιάζει ασυνήθιστα αυξημένη κίνηση.



Με την επιλογή Interface Traffic εμφανίζεται ένα νέο παράθυρο που δείχνει σε γραφική παράσταση την κίνηση ενός συγκεκριμένου interface, αυτό που είναι επιλεγμένο τη στιγμή που λαμβάνει χώρα το click event. Ο αλγόριθμος ζητάει επαναλαμβανόμενα από τον controller τον αριθμό των bytes που έχουν εισέλθει/εξέλθει. Αν δε εντοπιστεί διαφορά από την προηγούμενη μέτρηση, περιμένει (μέγιστο πέντε δευτερόλεπτα) και όταν αντιληφθεί αλλαγή εισάγει τα δεδομένα στο γράφημα. Έτσι αποφεύγονται λάθη σε περίπτωση που το πεδίο στο Restconf API δεν έχει ενημερωθεί τη στιγμή της μέτρησης. Επίσης αν η κίνηση ξεπεράσει το 85% της συνολικής χωρητικότητας, στη συγκεκριμένη περίπτωση είναι 10 Gb/s, εμφανίζεται κόκκινη γραμμή στο ύψος του 85% του bandwidth η οποία προειδοποιεί για συμφόρηση.



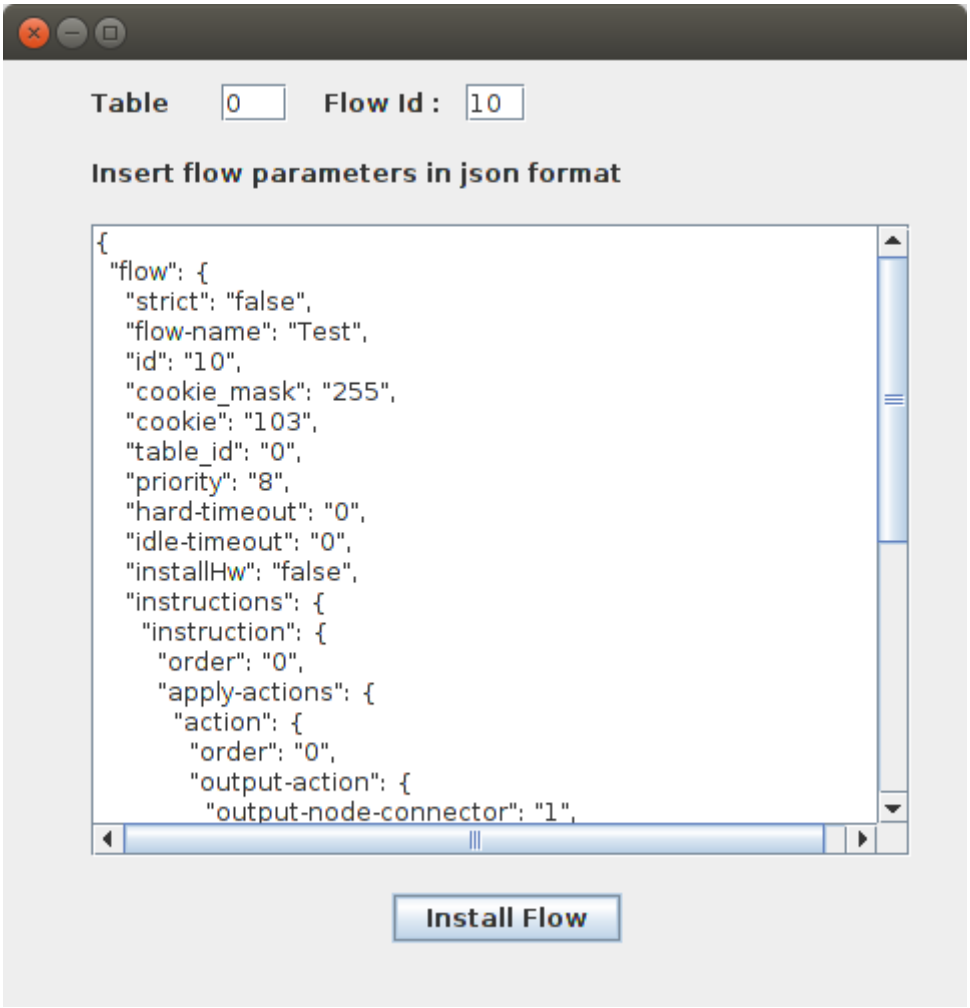
Στο αντικείμενο Flows του menu υπάρχουν οι επιλογές Drop Flow και Set flow. Το Drop Flow στέλνει ένα http delete request στο url που βρίσκεται το flow. Ο χρήστης πρέπει να έχει επιλεγμένο το flow που θέλει να διαγράψει από τη λίστα και να έχει ορίσει το table του flow. Εμφανίζεται μήνυμα για επιτυχημένο request ή στην αντίθετη περίπτωση το Error Code (π.χ. 400). Παρακάτω εμφανίζεται ένα τυπικό flow url στο Restconf API.

http://127.0.0.1:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/1

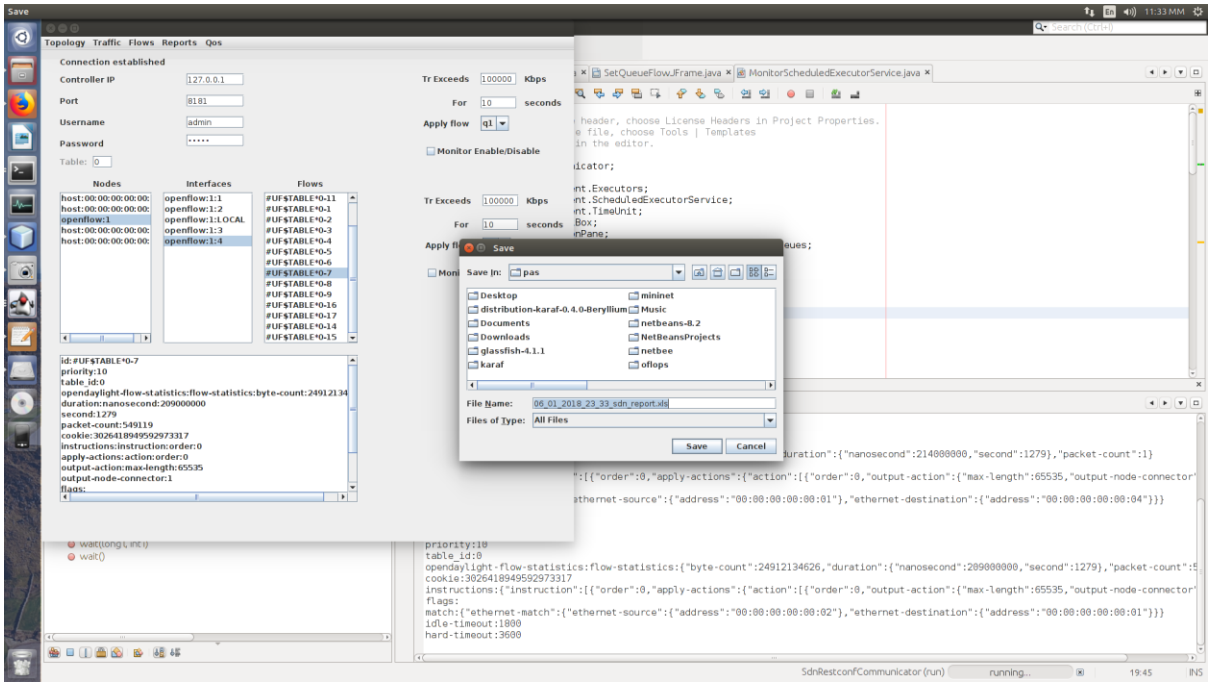
Στην επιλογή Set Flow έχουμε τη δυνατότητα να εισάγουμε ροές στον Controller και σε συνέχεια στις δικτυακές συσκευές. Εμφανίζεται ένα νέο παράθυρο στο οποίο πρέπει να εισάγουμε τον πίνακα (table) εισαγωγής της ροής, το όνομα της ροής (flow id) και τους κανόνες της ροής σε Json Format. Τα table id και flow id πρέπει να είναι ίδια με αυτά που ορίζονται στους κανόνες του flow. Η διαδικασία γίνεται μέσω http put request και λαμβάνουμε το response επιτυχίας η αποτυχίας. Αξίζει να σημειωθεί ότι ενώ το request μπορεί να είναι επιτυχημένο, αν το σώμα της ροής δεν ικανοποιεί κάποιους κανόνες του controller, το Restconf Url με τη ροή θα δημιουργηθεί αλλά ο κανόνας δε θα εισαχθεί στο δίκτυο, δηλαδή στο operational datastore.

Ο ODL έχει δύο datastores, το operational και το config. Στο Restconf API διαχωρίζονται αντίστοιχα τα url. (***127.0.0.1:8181/restconf/operational/ ,***

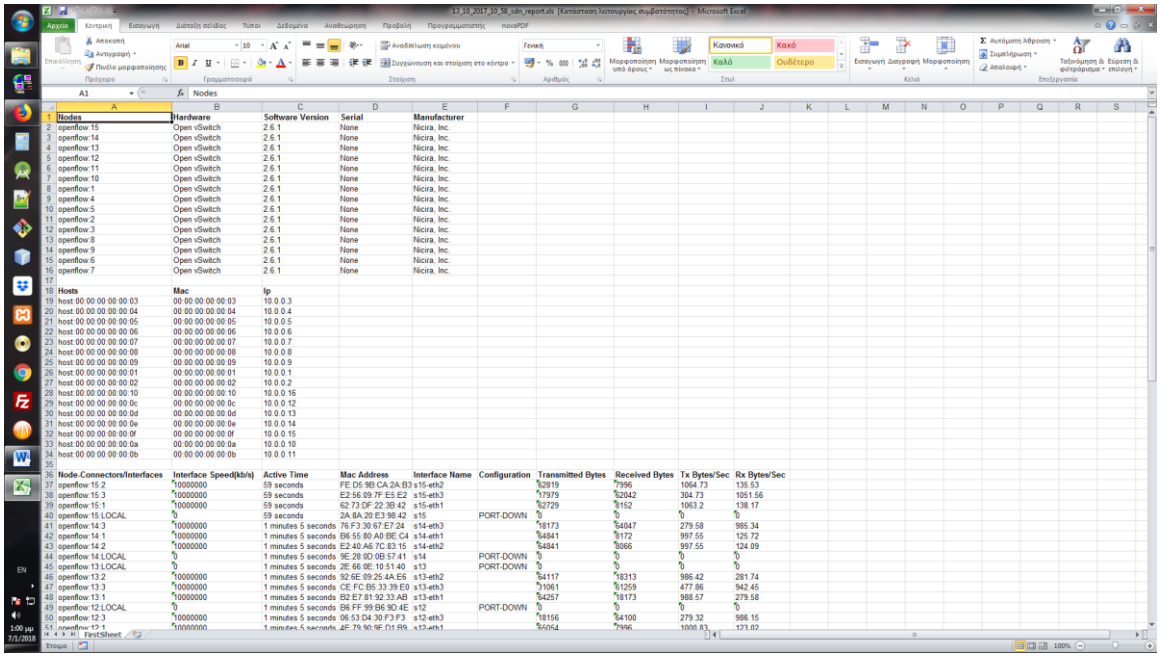
127.0.0.1:8181/restconf/config/).Για να λάβουμε δεδομένα στέλνουμε get requests στο operational datastore. Για να τροποποιήσουμε (εισαγωγή, διαγραφή) στέλνουμε το αντίστοιχο request στο config datastore. Συγκεκριμένα για εισαγωγή ροής κάνουμε put στο config url. Αν ο κανόνας πληρεί τις προϋποθέσεις θα εισαχθεί και στο operational datastore. Ένας εύκολος τρόπος ελέγχου είναι με την εντολή ***sudo ovs-ofctl dump-flows -O OpenFlow13 <switch name>*** από τερματική κονσόλα. Έτσι παρατηρούμε αν η ροή έχει «κατέβει» στο switch. Το εργαλείο Postman είναι ιδανικό για την αποθήκευση, τον πειραματισμό, την εισαγωγή και τον έλεγχο των ροών. Αν η ροή εισαχθεί σωστά θα εμφανίζεται και στη λίστα ροών του switch, μέσα από την εφαρμογή.



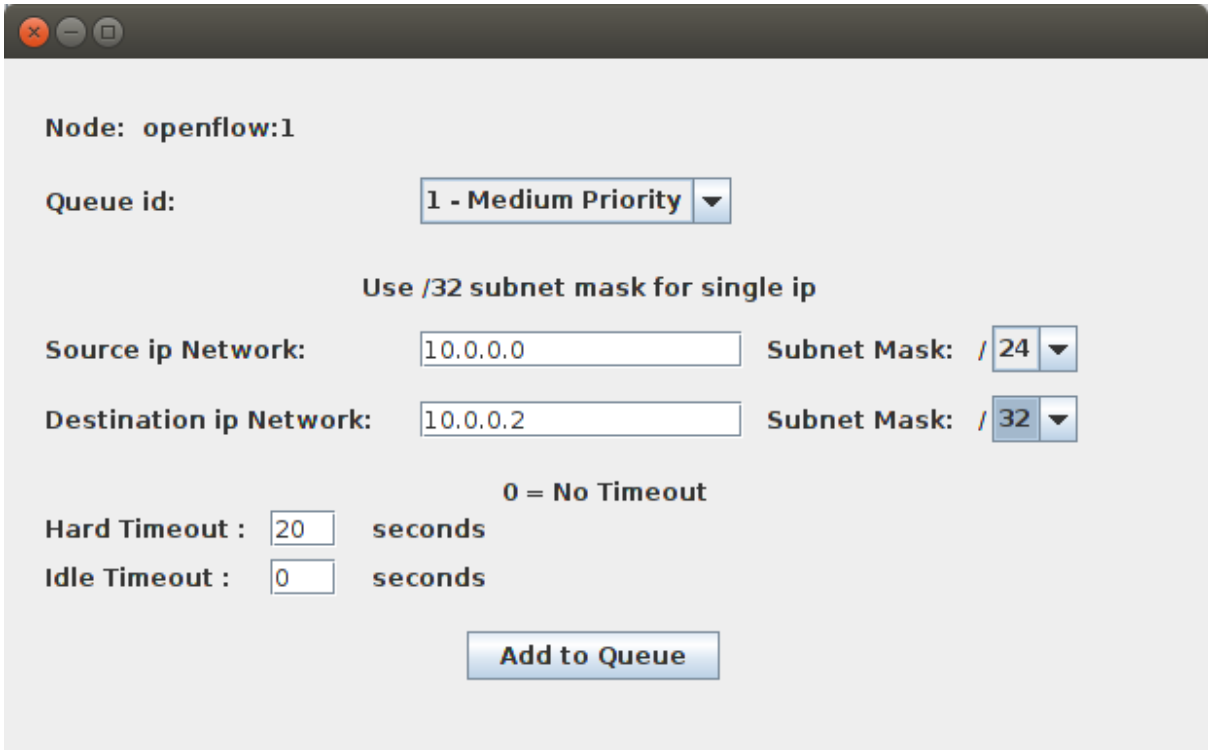
Από την επιλογή Reports > Excel Report έχουμε δυνατότητα αποθήκευσης διάφορων στοιχείων του δικτύου σε αρχείο excel. Εμφανίζεται παράθυρο επιλογής τοποθεσίας αποθήκευσης με προκαθορισμένο όνομα αρχείου βάση ημερομηνίας και ώρας.



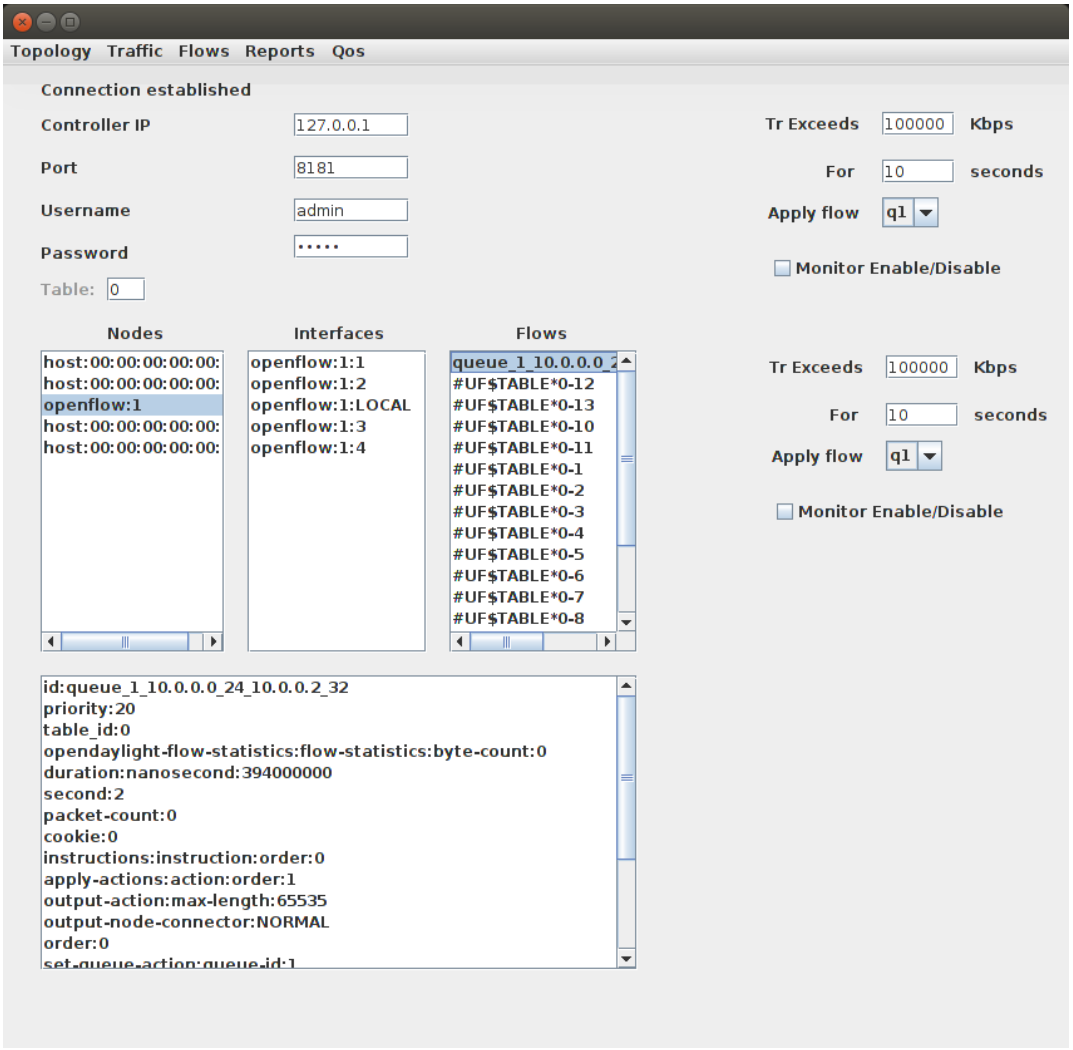
Το αρχείο excel έχει την παρακάτω μορφή.



Ακόμα από την επιλογή Qos έχουμε τη δυνατότητα εισαγωγής ροών που θα αξιοποιούν προκαθορισμένες ουρές. Η πολιτική Qos και οι ουρές πρέπει να έχουν περαστεί στο Open vSwitch. Στην εφαρμογή υπάρχει μοντέλο με δύο ουρές προτεραιότητας με σκοπό να δείξει τις δυνατότητες εφαρμογής QoS.



Εμφανίζεται το παραπάνω παράθυρο όπου έχουμε τη δυνατότητα εισαγωγής ροής για το τρέχων switch. Οι επιλογές είναι να μην εισαχθεί καθόλου σε QoS, να μπει στην ουρά ένα ή στην ουρά δύο. Στο παραπάνω παράδειγμα ο κανόνας ταυτίζει την κίνηση από το δίκτυο 10.0.0.0/24 προς την ip 10.0.0.2 και θα διαγραφεί μετά από 20 δευτερόλεπτα. Πατώντας το κουμπί «Add to Queue» λαμβάνουμε μήνυμα επιτυχίας ή αποτυχίας. Αν η ροή εισαχθεί σωστά θα πρέπει να εμφανίζεται στη λίστα ροών του switch.



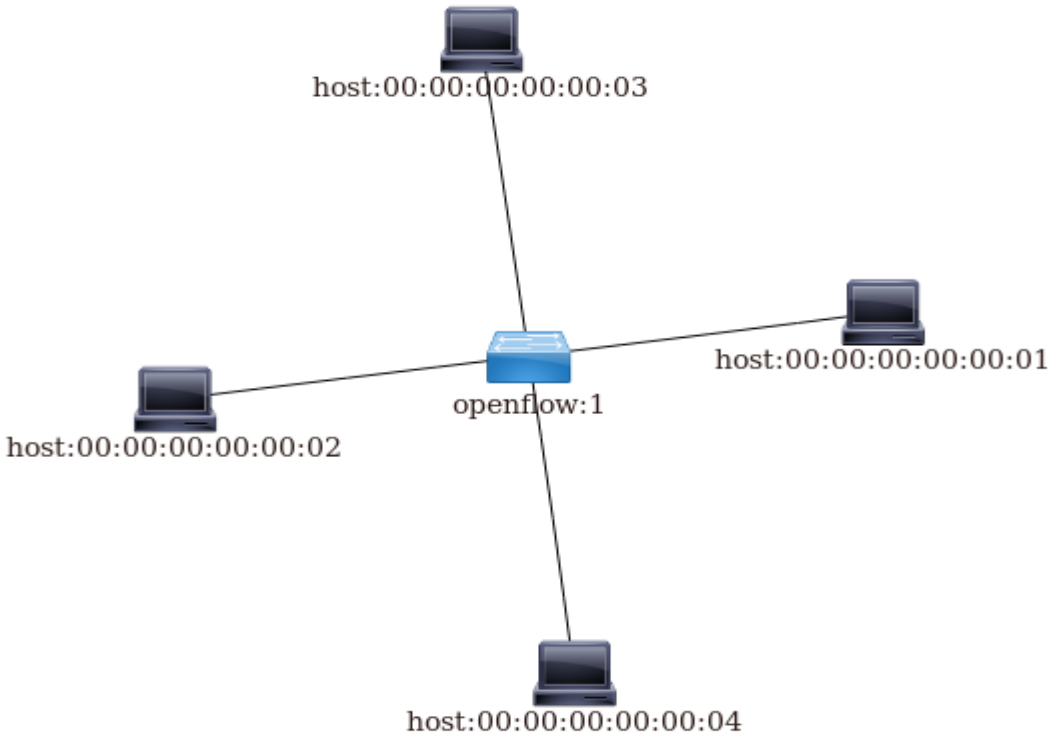
Τέλος υπάρχει δυνατότητα ορισμού ενός monitor σε interface για ενδεχόμενο congestion. Για παράδειγμα αν η κίνηση στο επιλεγμένο interface ξεπεράσει τα 100 Mb/s για 4 seconds, να εφαρμοστεί συγκεκριμένη ροή. Οι δυνατότητες της ροής είναι απεριόριστες, όπως π.χ. να μπει σε priority queue με guaranteed bit rate 50 mb/s η source ip 10.0.0.1 ή ακόμα η κίνηση που έχει source/destination ip τους servers του YouTube και του Netflix να μπει σε προτεραιότητα ή να γίνει διαχωρισμός tcp/udp κίνησης ή η κίνηση σε συγκεκριμένη πόρτα να πάρει προτεραιότητα. Στα πλαίσια της εφαρμογής υπάρχουν προκαθορισμένες ροές και ο χρήστης ορίζει τους κανόνες του monitor (bandwidth threshold, time in seconds). Μεγαλύτερη ανάλυση για σενάρια εφαρμογής γίνεται στο επόμενο κεφάλαιο με δοκιμές ροών και κίνησης.

ΚΕΦΑΛΑΙΟ 3 – Quality of Service Tests and Measurements

3.1 Σενάριο με ένα switch και χρήση Monitor

Στο περιβάλλον δοκιμών που υλοποιήθηκε η εργασία δημιουργήθηκε εικονικό δίκτυο μέσω του Mininet Emulator. Για την παραγωγή κίνησης χρησιμοποιήθηκε το εργαλείο iperf μέσα από τερματική κονσόλα στους εικονικούς hosts. Στα παρακάτω σενάρια δημιουργήθηκε iperf server σε ένα host και έγιναν δοκιμές μέτρησης throughput από διαφορετικά σημεία στο δίκτυο.

Στο πρώτο σενάριο έχουμε μια πολύ απλή δικτυακή τοπολογία. Στον host με mac 00:00:00:00:00:01 και ip 10.0.0.1 δημιουργείται tcp server που ακούει στην πόρτα 6000. ***iperf –s –p 6000 –i 1***

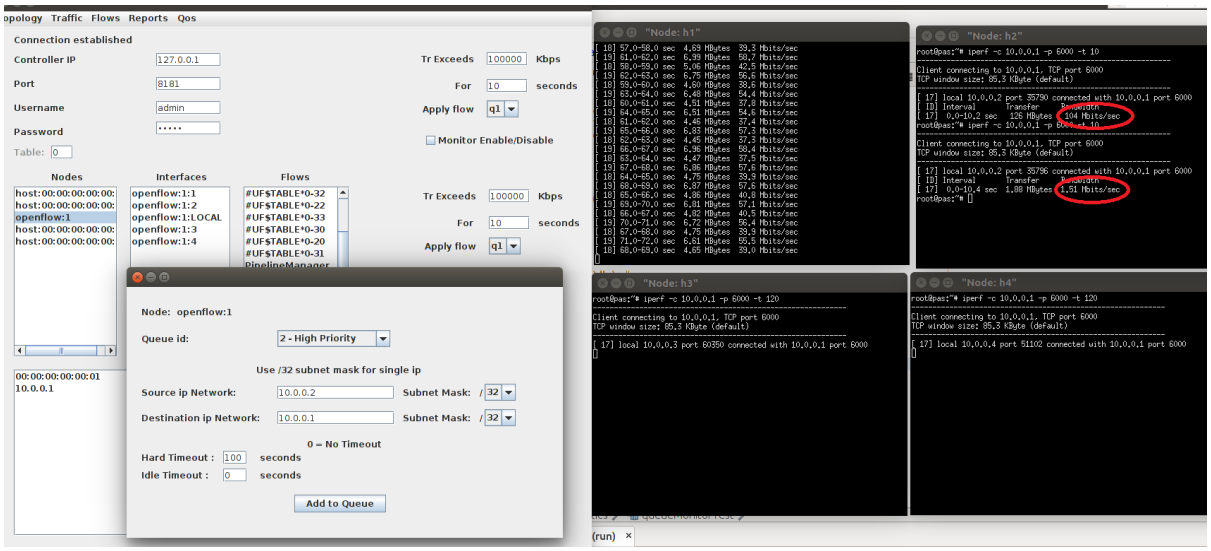


Στην πόρτα s1-eth1 εφαρμόζεται το qos και δύο queues με μέγιστο/ελάχιστο ρυθμό 20 Mb/s (Medium Priority) και 50 Mb/s (High Priority) αντίστοιχα.

```
pas@pas: ~
pas@pas:~$ sudo ovs-vsctl -- --all destroy QoS -- --all destroy Queue
[sudo] password for pas:
pas@pas:~$ sudo ovs-vsctl -- --all destroy QoS -- --all destroy Queue
pas@pas:~$ sudo ovs-vsctl list qos
pas@pas:~$ sudo ovs-vsctl list queue
pas@pas:~$ sudo ovs-vsctl set port s1-eth1 qos=@min_bw -- --id=@min_bw create qos type=linux-htb queues=1=@min_20mb,2=@min_50mb -- --id=@min_20mb create queue other-conf
ig:min-rate=20000000 other-config:max-rate=20000000 -- --id=@min_50mb create queue other-config:min-rate=50000000 other-config:max-rate=50000000
d3ccd33c-aebc-4dc5-a53e-41dac7141833
7853e69f-a44a-46b3-a086-55434206487e
27cb8d55-b9cf-4a0b-990a-c2893fea76ae
pas@pas:~$ sudo ovs-vsctl list queue
_uuid               : 27cb8d55-b9cf-4a0b-990a-c2893fea76ae
dscp                : []
external_ids        : {}
other_config        : {max-rate="50000000", min-rate="50000000"}

_uuid               : 7853e69f-a44a-46b3-a086-55434206487e
dscp                : []
external_ids        : {}
other_config        : {max-rate="20000000", min-rate="20000000"}
pas@pas:~$ sudo ovs-vsctl list qos
_uuid               : d3ccd33c-aebc-4dc5-a53e-41dac7141833
external_ids        : {}
other_config        : {}
queues              : {1=7853e69f-a44a-46b3-a086-55434206487e, 2=27cb8d55-b9cf-4a0b-990a-c2893fea76ae}
type                : linux-htb
pas@pas:~$
```

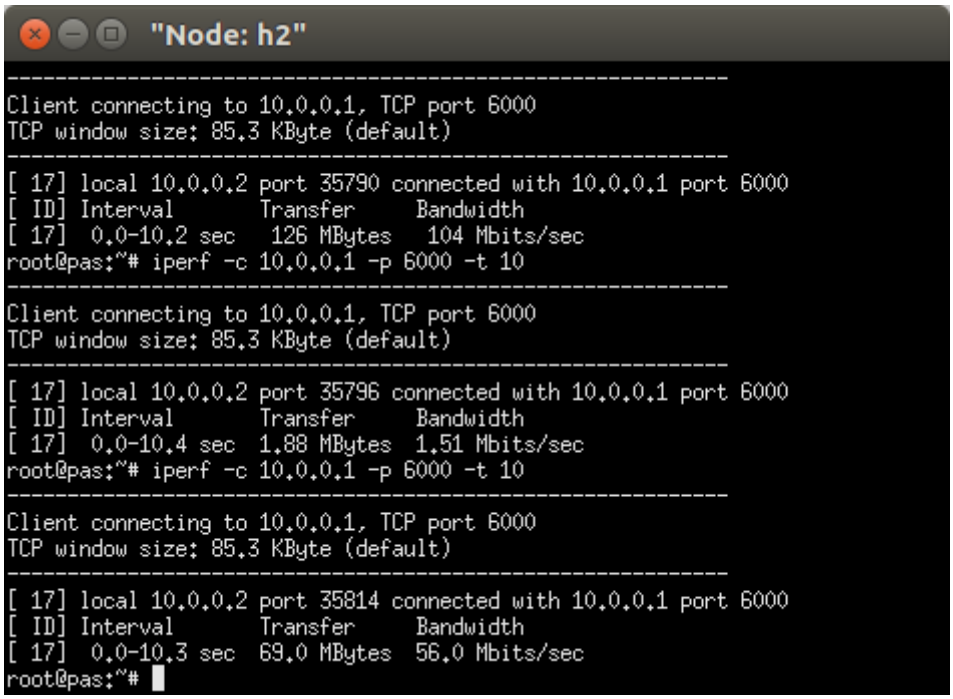
Κάνουμε μέτρηση του throughput από τον host 2 (10.0.0.2) προς τον server στην 10.0.0.1 χωρίς άλλη κίνηση στο δίκτυο (***iperf –c 10.0.0.1 –p 6000 –t 10)*** και παρατηρούμε data rate στα 100 Mb/s. Στη συνέχεια παράγουμε κίνηση από τους h3 και h4 αντίστοιχα ώστε να προκαλέσουμε συμφόρηση, χωρίς να έχει εισαχθεί η κίνηση σε ουρά προτεραιότητας. Παρατηρούμε ότι το data rate του h2 έχει πέσει στο 1,5 Mb/s.



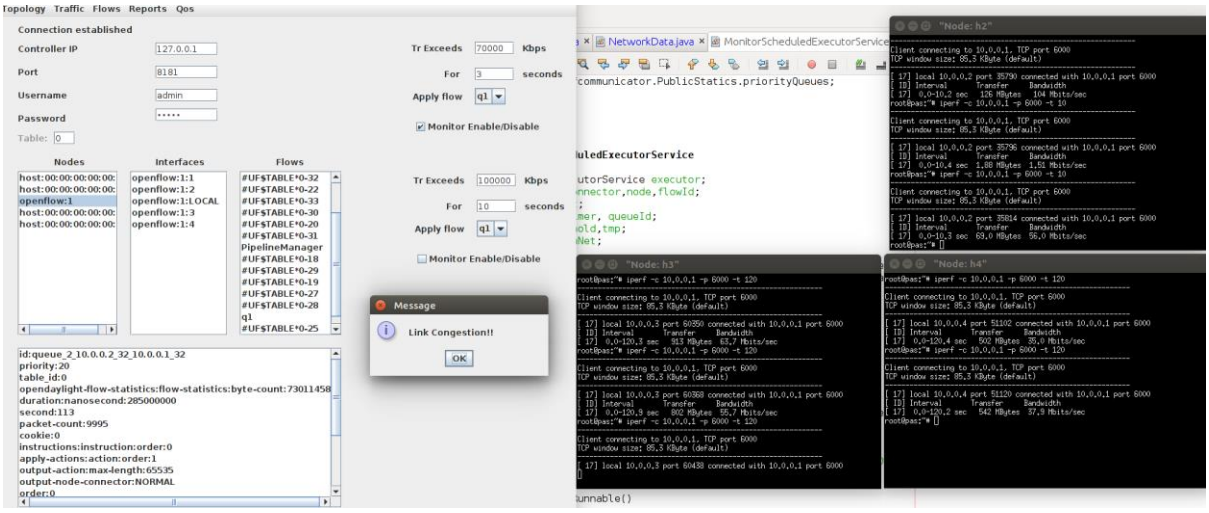
Στη συνέχεια εισάγουμε την κίνηση με source ip 10.0.0.2 και destination 10.0.0.1 στο queue 2 (50mb/s) μέσα από το interface της εφαρμογής. Ελέγχουμε την ροή στο switch και βλέπουμε ότι σε match της κίνησης εκτός από το NORMAL action, δηλαδή την δρομολόγηση της, γίνεται και εισαγωγή των πακέτων στο queue 2.

```
pas@pas: ~
 cookie=0x0, duration=395.656s, table=247, n_packets=14, n_bytes=1232, priority=0 actions=goto_table:248
 cookie=0x0, duration=395.656s, table=248, n_packets=14, n_bytes=1232, priority=0 actions=goto_table:249
 cookie=0x0, duration=395.656s, table=249, n_packets=14, n_bytes=1232, priority=0 actions=goto_table:250
 cookie=0x0, duration=395.656s, table=250, n_packets=14, n_bytes=1232, priority=0 actions=goto_table:251
 cookie=0x0, duration=395.656s, table=251, n_packets=14, n_bytes=1232, priority=0 actions=goto_table:252
 cookie=0x0, duration=395.657s, table=252, n_packets=14, n_bytes=1232, priority=0 actions=goto_table:253
 cookie=0x2b0000000000004, duration=391.766s, table=0, n_packets=12, n_bytes=840, priority=2,in_port=1 actions=output:2,output:3,output:4,CONTROLLER:65535
 cookie=0x2b0000000000005, duration=391.766s, table=0, n_packets=11, n_bytes=798, priority=2,in_port=2 actions=output:1,output:3,output:4,CONTROLLER:65535
 cookie=0x2b0000000000006, duration=391.766s, table=0, n_packets=11, n_bytes=798, priority=2,in_port=3 actions=output:1,output:2,output:4,CONTROLLER:65535
 cookie=0x2b0000000000007, duration=391.766s, table=0, n_packets=12, n_bytes=840, priority=2,in_port=4 actions=output:1,output:2,output:3,CONTROLLER:65535
 cookie=0x2a000000000000d, duration=378.795s, table=0, n_packets=10907, n_bytes=135221502, idle_timeout=1800, hard_timeout=3600, priority=10,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions
ut:1
 cookie=0x2a000000000000c, duration=378.795s, table=0, n_packets=10863, n_bytes=718258, idle_timeout=1800, hard_timeout=3600, priority=10,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02 actions=ou
2
 cookie=0x2a000000000000e, duration=378.785s, table=0, n_packets=122004, n_bytes=8369232, idle_timeout=1800, hard_timeout=3600, priority=10,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03 actions=
t:1
 cookie=0x2a000000000000f, duration=378.785s, table=0, n_packets=126724, n_bytes=969475908, idle_timeout=1800, hard_timeout=3600, priority=10,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01 actio
put:1
 cookie=0x2a0000000000010, duration=378.777s, table=0, n_packets=103850, n_bytes=7143472, idle_timeout=1800, hard_timeout=3600, priority=10,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:04 actions=
t:4
 cookie=0x2a0000000000011, duration=378.775s, table=0, n_packets=107471, n_bytes=534949250, idle_timeout=1800, hard_timeout=3600, priority=10,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:01 actio
put:1
 cookie=0x2a0000000000012, duration=378.774s, table=0, n_packets=2, n_bytes=140, idle_timeout=1800, hard_timeout=3600, priority=10,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03 actions=output:3
 cookie=0x2a0000000000013, duration=378.772s, table=0, n_packets=2, n_bytes=140, idle_timeout=1800, hard_timeout=3600, priority=10,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x2a0000000000014, duration=378.772s, table=0, n_packets=2, n_bytes=140, idle_timeout=1800, hard_timeout=3600, priority=10,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:04 actions=output:4
 cookie=0x2a0000000000015, duration=378.772s, table=0, n_packets=2, n_bytes=140, idle_timeout=1800, hard_timeout=3600, priority=10,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x2a0000000000016, duration=378.769s, table=0, n_packets=2, n_bytes=140, idle_timeout=1800, hard_timeout=3600, priority=10,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:04 actions=output:4
 cookie=0x2a0000000000017, duration=378.760s, table=0, n_packets=2, n_bytes=140, idle_timeout=1800, hard_timeout=3600, priority=10,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:03 actions=output:3
 cookie=0x0, duration=9.042s, table=0, n_packets=0, n_bytes=0, hard_timeout=120, priority=20,tcp,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=set_queue:2,NORMAL
 cookie=0x2b0000000000003, duration=394.040s, table=0, n_packets=0, n_bytes=0, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
pas@pas:~$
```

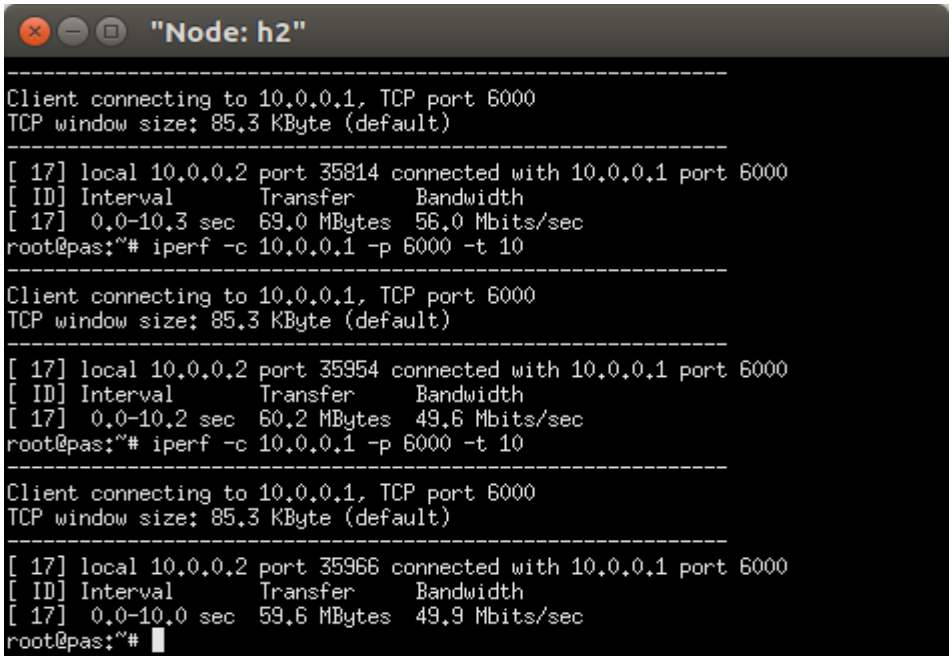
Επαναλαμβάνουμε τη μέτρηση throughput με κίνηση στο link και εντοπίζουμε ότι το data rate είναι πλέον στα 50mb/s.



Σβήνουμε τη συγκεκριμένη ροή και επαναλαμβάνουμε τις μετρήσεις χωρίς να κάνουμε χειροκίνητη εισαγωγή του flow αλλά με χρήση monitor. Στο monitor ορίζουμε ότι εάν και όταν η κίνηση στο interface openflow:1:1 (πρώτη πόρτα του switch 1) ξεπεράσει τα 70 mb/s για 3 δευτερόλεπτα, τότε να γίνει εισαγωγή του q1 στο Open vSwitch. Το q1 είναι προκαθορισμένο (βάζει στην ουρά 2, κίνηση με source 10.0.0.2, destination 10.0.0.1 για 50 seconds). Σε ιδανικό σχεδιασμό η λίστα με τα πιθανά flow θα βρισκόταν σε database που θα ενημερωνόταν δυναμικά. Με επιλογή του checkbox ***Monitor Enable/Disable*** ξεκινάει ένα thread που μετράει την κίνηση στο interface και σταματάει αν το monitor checkbox σταματήσει να είναι επιλεγμένο. Μπορούν να τρέχουν πολλαπλά monitors ταυτόχρονα με διαφορετικούς κανόνες και εισαγωγή διαφορετικών flows. Αν ικανοποιηθούν οι κανόνες εισάγεται η ροή στο switch και εμφανίζει παράθυρο μηνύματος.

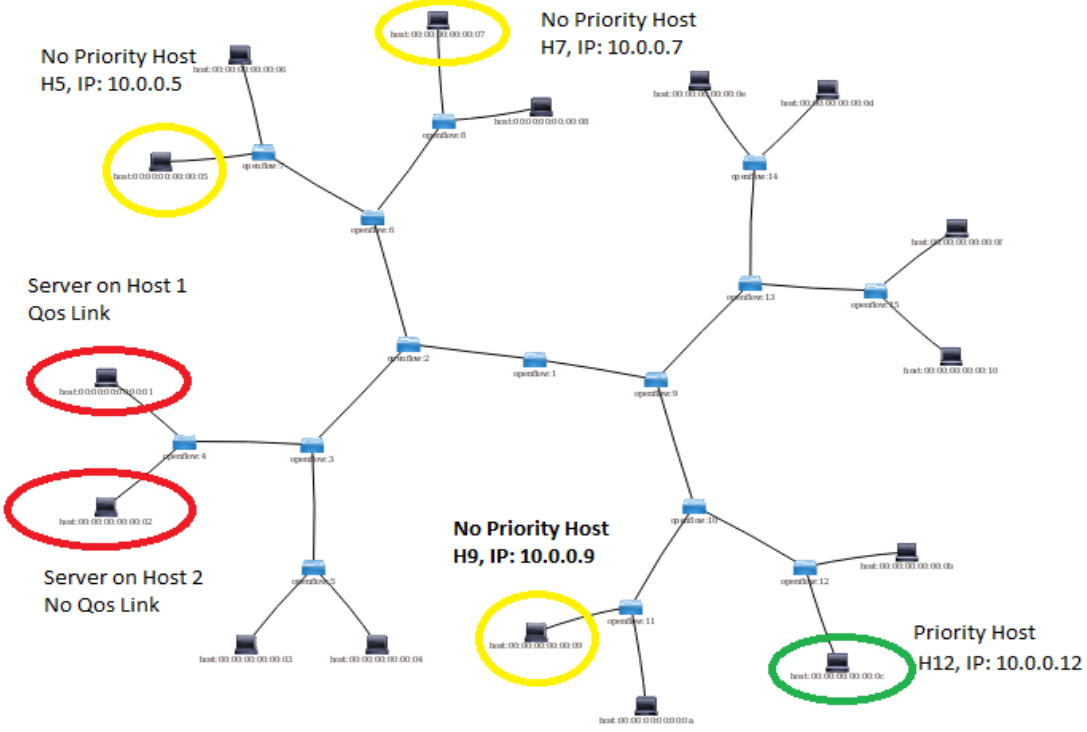


Παρατηρούμε σε όλες τις μετρήσεις ότι η κίνηση σε congested link, είτε με χειροκίνητη εισαγωγή, είτε μέσω του traffic monitor, δεν πέφτει κάτω από 50mb/s καθώς τόσος είναι ο ελάχιστος ρυθμός που έχει οριστεί στο link.

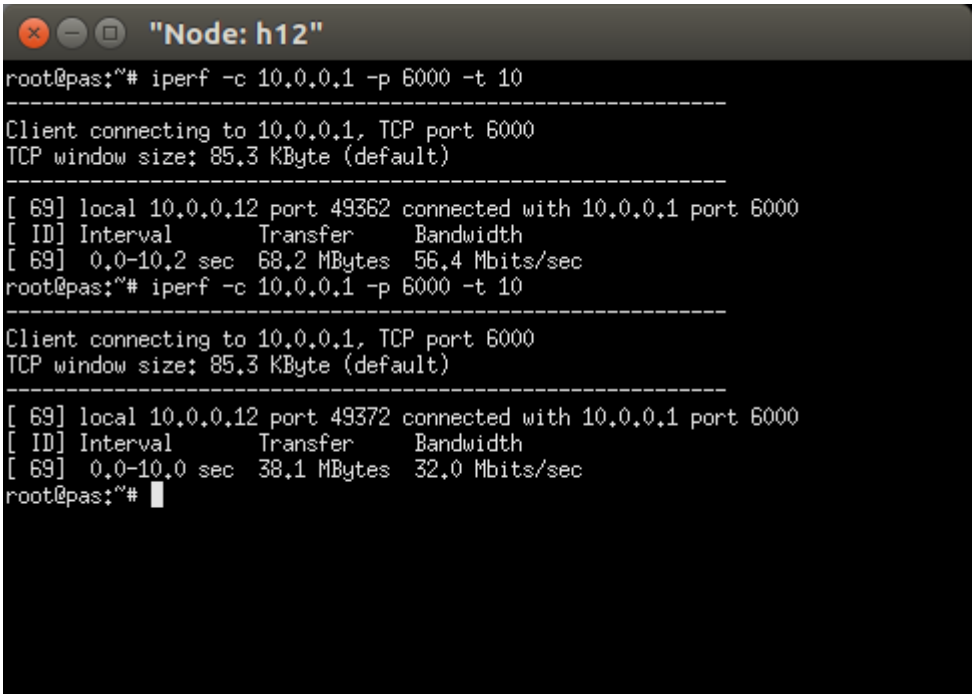


3.2 Σενάριο σε δομή δένδρου με ύψος τέσσερα

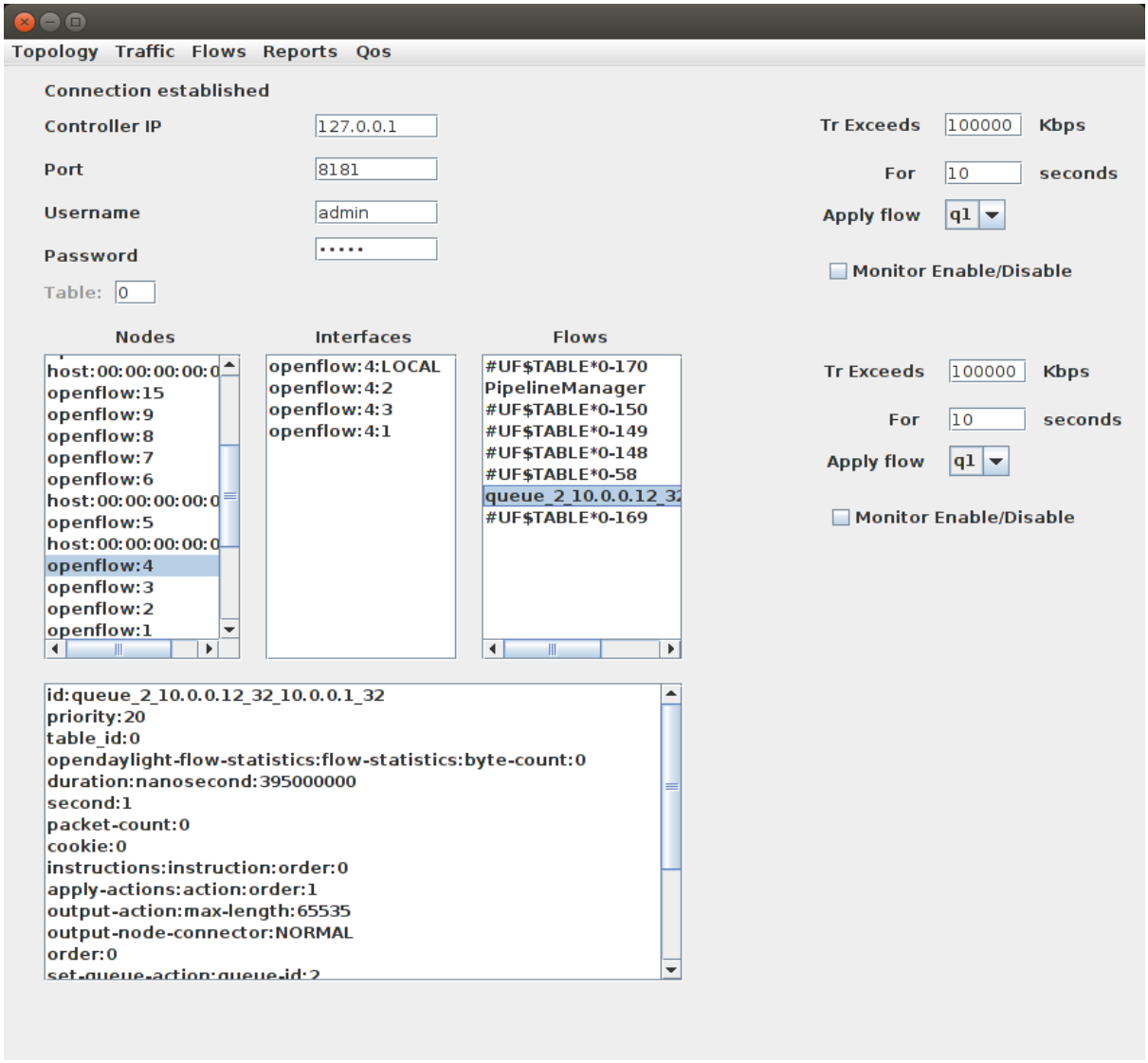
Στο παρακάτω σενάριο χρησιμοποιήθηκε παρόμοια μεθοδολογία σε μεγαλύτερο δίκτυο, με την κίνηση να προέρχεται από διαφορετικές τερματικές συσκευές από ξεχωριστά τμήματά του.



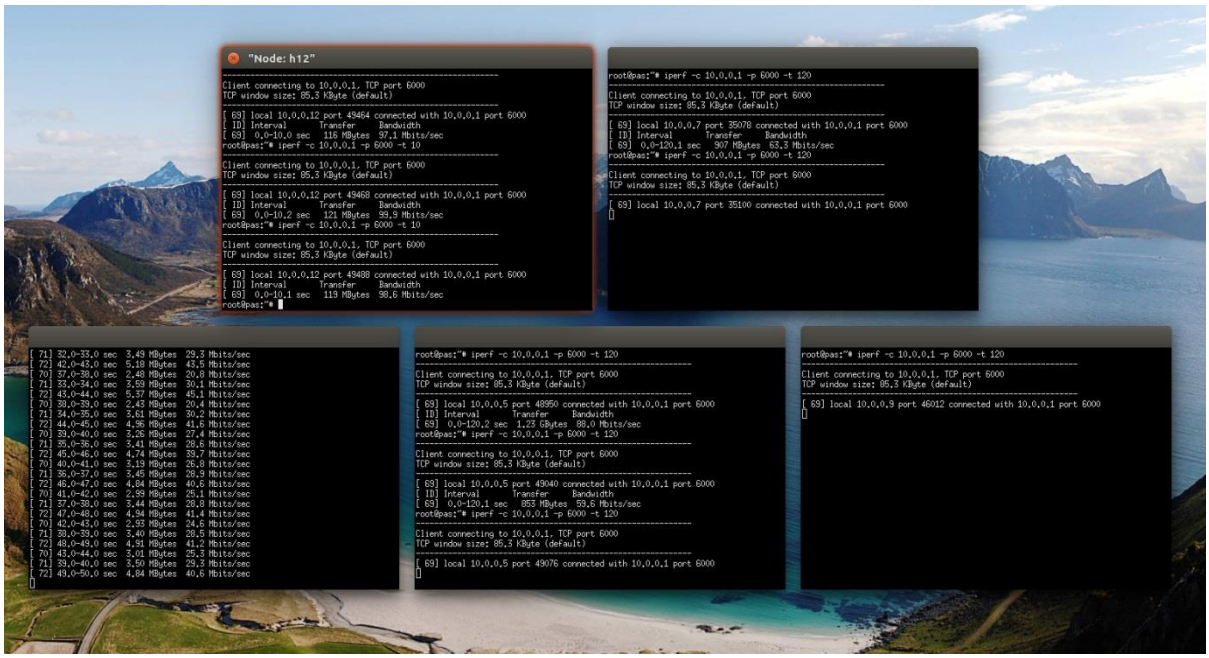
Ορίζουμε δύο priority queues με ελάχιστο/μέγιστο 20 Mb/s/50 Mb/s και 50 Mb/s/100 Mb/s. Δημιουργούμε δύο tcp servers στους hosts ένα και δύο. Πραγματοποιούμε μέτρηση από τον Host 12 όσο ο σύνδεσμος έχει κίνηση από τους host 5,7 και 9 χωρίς να έχει εφαρμοστεί QoS flow παρατηρώντας αυξομείωση και αστάθεια στο throughput.



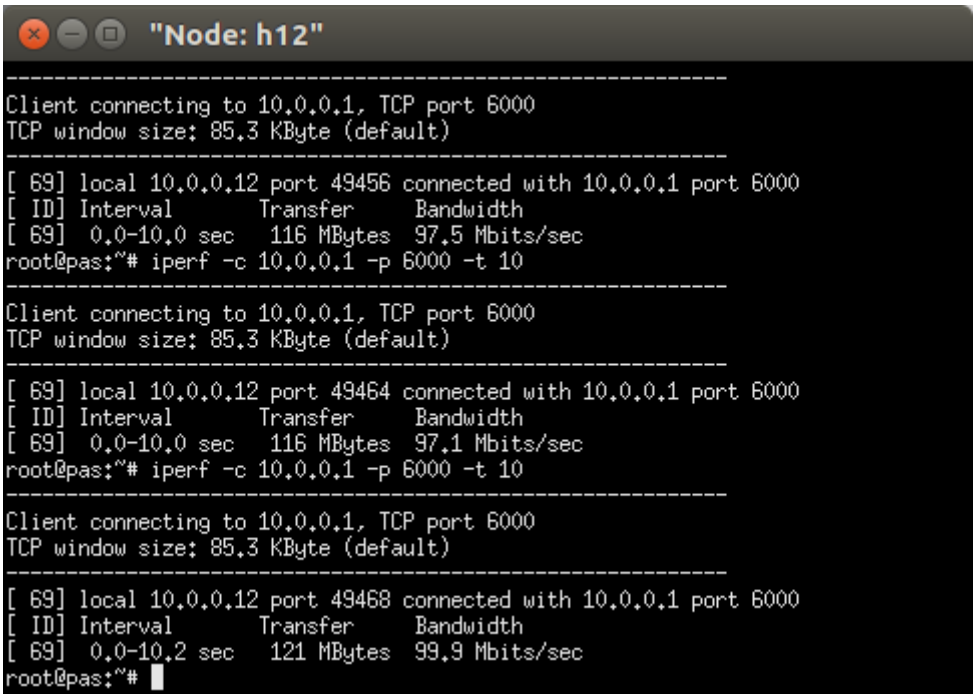
Δημιουργούμε ροή η οποία βάζει την κίνηση με source ip 10.0.0.12 και destination 10.0.0.1 στο queue 2 (min 50, max 100 Mb/s). Στην συγκεκριμένη τοπολογία ο host 1 βρίσκεται στο switch 4 (openflow:4), οπότε εκεί εισάγεται και η ροή.



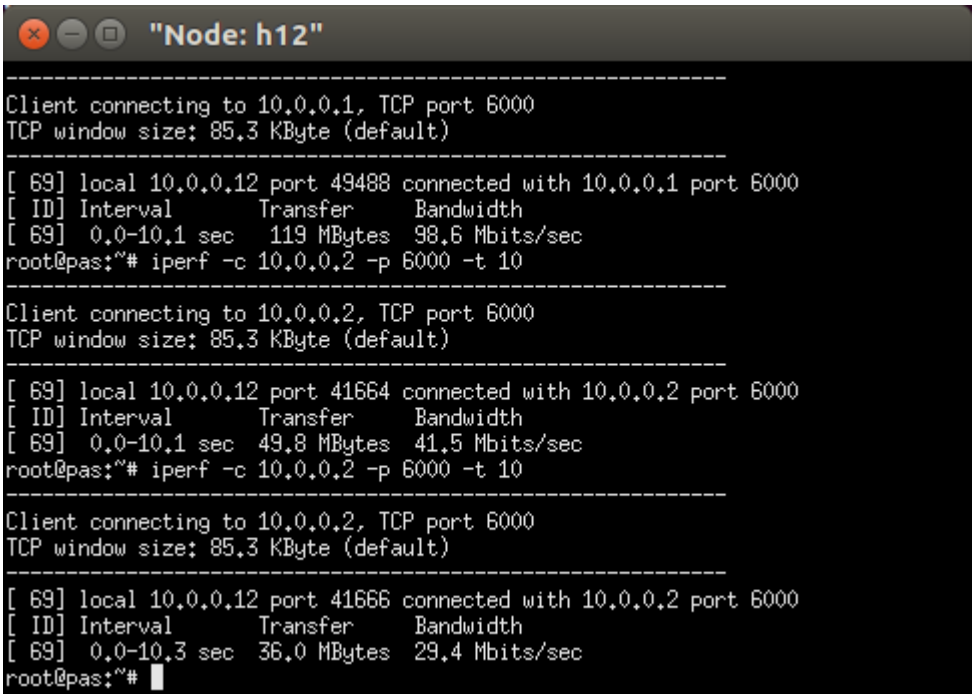
Πραγματοποιούμε πάλι τις ίδιες μετρήσεις ενώ υπάρχει κίνηση στο δίκτυο.



Βλέπουμε πλέον ότι ο host 12 πιάνει σχεδόν το μέγιστο data rate της ουράς στην οποία βρίσκεται.



Στη συνέχεια δρομολογούμε την κίνηση στον host 2 ο οποίος δε βρίσκεται σε QoS link και δεν έχει εφαρμοστεί κανένα priority. Όπως ήταν αναμενόμενο το throughput από τον host 12 είναι ασταθές και μικρότερο από το αντίστοιχο , προστατευόμενο από QoS, στον host 1.



3.3 Αποτελέσματα μετρήσεων

Από τις μετρήσεις και τις δοκιμές συμπεραίνουμε πως το συγκεκριμένο μοντέλο εφαρμογής Ποιότητας της Υπηρεσίας είναι πλήρως λειτουργικό. Απαραίτητη προϋπόθεση είναι να υπάρχουν κανόνες QoS και προκαθορισμένες ουρές στο OpenFlow switch στα σημεία συμφόρησης.

Εφόσον αυτή η συνθήκη έχει καλυφθεί, η ευελιξία των κανόνων που θα αξιοποιήσουν τις ουρές είναι απεριόριστη. Ακόμα τα flows που υλοποιούν την πολιτική εφαρμόζονται σε ολόκληρο το switch και όχι ανά interface. Η διαχείριση μπορεί να γίνεται δυναμικά μέσω αυτοματισμών και προγραμματισμού. Επίσης, οι αλλαγές στην ροή της κίνησης μπορούν να επιτευχθούν με ευκολία μέσω ενός http request αλλά και μέσω γραφικού περιβάλλοντος με μερικά κλικ. Τέλος, η δυνατότητα τοποθέτησης των trigger monitors δεν απαιτεί την χειροκίνητη αντίδραση του διαχειριστή δικτύου μετά από κάποιο ερέθισμα όπως στα παραδοσιακά δίκτυα (π.χ. SNMP messages).

ΣΥΜΠΕΡΑΣΜΑΤΑ

Μετά από μελέτη της τεχνολογίας των Software Defined Networks , την εξέλιξη αυτής τα τελευταία έτη, τη διερεύνηση μεθοδολογιών, πρωτοκόλλων και λογισμικού που είναι απαραίτητα για τα SDN και την υλοποίηση εφαρμογής για τα SDN δίκτυα καταλήγουμε στα εξής συμπεράσματα.

Τα SDN στοχεύουν σε μεγάλη αναδιαμόρφωση της δομής των δικτύων υπολογιστών, επιλύουν προβλήματα που υπήρχαν από τη δημιουργία τους και παρέχουν απόλυτη ευελιξία στην διαχείριση, εγκατάσταση και τροποποίηση τους. Επίσης μειώνουν το κόστος λειτουργίας μέσω προγραμματισμού αλλά και εγκατάστασης χρησιμοποιώντας φθηνότερο υλικό. Δεν ακυρώνουν την κλασική διαστρωμάτωση των δικτύων και των πρωτοκόλλων επικοινωνίας, αλλά διαχωρίζουν τη λογική από τη μεταφορά δεδομένων.

Υπάρχει δισταγμός ως προς την καθολική υλοποίηση τους, όπως και με όλες τις νέες τεχνολογίες, με την έλλειψη τεχνογνωσίας να αποτελεί ένα εμπόδιο προς αντιμετώπιση.

Μέσω του πρακτικού μέρους, αντιλαμβανόμαστε την ευκολία δημιουργίας λογισμικού, το οποίο θα εφαρμόζει υπηρεσίες στο δίκτυο. Τα APIs επικοινωνίας με τους Controllers παρέχουν πολλές επιλογές και είναι μεγάλη επένδυση αφού αποτελούν το σημείο επαφής των εξωτερικών εφαρμογών. Στο Restconf API ένα Http Request είναι αρκετό για τη διαμόρφωση της κίνησης και της δρομολόγησης σε οποιοδήποτε δικτυακό στοιχείο.

Ενώ τα APIs παρέχουν ευκολία στην παραμετροποίηση των δικτυακών συσκευών, πρέπει να αφιερωθεί αρκετός χρόνος στην κατανόηση της δομής και της αρχιτεκτονικής των SDN και του Controller που θα χρησιμοποιηθεί. Υπάρχει αρκετό υλικό διαθέσιμο, αλλά είναι ένα από τα κομμάτια που θα πρέπει να βελτιωθούν ώστε η εφαρμογή των SDN να εξαπλωθεί περισσότερο και με μεγαλύτερη ευκολία.

Τέλος, αξίζει να σημειωθεί ότι τα SDN δεν αποτελούν πλέον τεχνολογία του μέλλοντος αλλά του παρόντος καθώς έχουν ήδη μεγάλη διείσδυση στην αγορά, κυρίως στο εξωτερικό. Έχουν υλοποιηθεί πάρα πολλές εμπορικές λύσεις και εφαρμογές και τα μηχανήματα των μεγάλων προμηθευτών υποστηρίζουν το OpenFlow πρωτόκολλο. Το γεγονός ότι οι μεγαλύτεροι κατασκευαστές δικτυακού εξοπλισμού, επενδύουν στην δημιουργία ελεύθερου λογισμικού για SDN, δηλώνει τη σιγουριά τους πως η τεχνολογία αυτή είναι θέμα χρόνου να εφαρμοστεί καθολικά.

Βιβλιογραφία

Nick Feamster, Jennifer Rexford, Ellen Zegura. *The Road to SDN: An Intellectual History of Programmable Networks*, December 2013

Network Function Virtualization (NFV), October 2014
https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf

David Meyer, *OpenDaylight Introduction and Overview*
http://www.1-4-5.net/~dmm/talks/interop_nyc_2013.pdf

Openflow: Enabling Innovation in Campus Networks, March 2008
http://archive.openflow.org/documents/openflow-wp-latest.pdf

Bob Lantz, Brian O'Connor. *A Mininet-based Virtual Testbed for Distributed SDN Development*, August 2015

Chung-Sheng Li, Wanjiun Liao, Software Defined Networks, February 2013

Amin Tootoonchian, Sergey Gorbunov, Yashar Ganjali, Martin Casado, Rob Sherwood, On Controller Performance in Software-Defined Networks, 2012

Brandon Heller. *Reproducible network research with high-fidelity emulation*. Ph.D. Thesis, Stanford University, 2013.

Paul Goransson, Chuck Black *Software Defined Networks: A Comprehensive Approach*, May 2014

Christopher Monsanto, Joshua Reich, Nate Foster Composing Software Defined Networks, April 2013

Muhammad H. Raza, Shyamala C. Sivakumar, Ali Nafarieh, Bill Robertson A Comparison of Software Defined Network (SDN) Implementation Strategies, 2014

Atanas Mirchev, Survey of Concepts for QoS improvements via SDN, September 2015

Edited by Fei Hu, Network Innovation through OpenFlow and SDN

http://www.itu.int/en/ITU-T/sdn/Pages/default.aspx
https://www.opennetworking.org/sdn-definition/
https://www.opendaylight.org/
https://wiki.opendaylight.org/

https://www.sdxcentral.com/
http://searchsdn.techtarget.com/
http://docs.opendaylight.org/
https://nexus.opendaylight.org
http://ryu.readthedocs.io/en/latest/ofproto_ref.html
https://www.opennetworking.org/
https://www.cisco.com/
http://www.thetech.in
http://www.hp.com
http://sdntutorials.com
https://tools.ietf.org/
https://www.ietf.org/
https://www.networkworld.com
http://mininet.org/
http://openvswitch.org
https://gcn.com
https://www.bigswitch.com
http://www.nojitter.com
https://www.osgi.org

Εικόνες

https://fedcsis.org/2014/sdn.html
https://www.opennetworking.org/sdn-definition/
http://searchsdn.techtarget.com/
https://github.com/BRCDcomm/BVC/wiki/MD-SAL
https://wiki.opendaylight.org/
https://www.cisco.com/
https://www.sdxcentral.com/
https://www.bigswitch.com
https://www.canstockphoto.com/word-cloud-qos-19010668.html
https://www.osgi.org