Πανεπιστήμιο Θεσσαλίας, 2015-2016

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Διπλωματική Εργασία

Θέμα:

"Σχεδιασμός και υλοποίηση πρωτοκόλλου διαχείρισης φόρτου κίνησης σε OpenFlow δίκτυο"

"Design and implementation of traffic load balancing protocol in OpenFlow networks"

Ευσταθίου Ελισάβετ – Αποστολία

Επιβλέπων Καθηγητής:

Κοράκης Αθανάσιος (Επίκουρος Καθηγητής)

Συν επιβλέπων Καθηγητής:

Αργυρίου Αντώνιος (Λέκτορας Καθηγητής)

Βόλος, Οκτώμβριος 2015

Ευχαριστίες,

Με την εκπόνηση της παρούσας Διπλωματικής εργασίας, φέρνω εις πέρας τις προπτυχιακές μου σπουδές στο Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας.

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον Επίκουρο καθηγητή του Τμήματος Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών κ. Κοράκη Αθανάσιο, για την ανάθεση της παρούσας Διπλωματικής Εργασίας και για την ευκαιρία συνεργασίας με την ομάδα του NITlab που μου έχει προσφέρει. Από καρδίας θα ήθελα να ευχαριστήσω ολόκληρη την ομάδα του NITlab για την καθοδήγηση που μου έδωσε καθόλη τη διάρκεια της εκπόνησης της Διπλωματικής μου εργασίας. Ιδιαιτέρως να ευχαριστήσω τον Διδάκτορα του Τμήματος Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών Χούμα Κωνσταντίνο, τον Διδάκτορα Κερανίδη Ευστράτιο για την βοήθεια και τις υποδείξεις στην εκπόνηση της εργασίας, καθώς και την εμπιστοσύνη που έδειξαν στο πρόσωπό μου.

Ευχαριστώ θερμά την οικογένεια μου για την αμέριστη συμπαράσταση που μου παρείχε όλα αυτά τα χρόνια για την ολοκλήρωση των προπτυχιακών μου σπουδών. Ιδιαίτερα τους ευχαριστώ για την στήριξη και την εμπιστοσύνη που επέδειξαν στις επιλογές και στην κρίση μου.

Τέλος θα ήθελα να ευχαριστήσω όλους τους υπέροχους φίλους που απέκτησα εδώ στο Βόλο, και ιδιαίτερα την Τσακίρη Κωνσταντίνα, που πάντα βρίσκονταν δίπλα μου για να με στηρίξουν όχι μόνο στην εκπόνηση της Διπλωματικής μου εργασίας, αλλά καθ'όλη τη διάρκεια της φοίτησης μου στο Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών.

Αφιερωμένο στους γονείς μου, Ηλία και Ελευθερία.

Και στον αδερφό μου, Χρήστο.

Πίνακας Περιεχομένων

Περίληψη

Η παρούσα Διπλωματική εργασία πραγματεύεται τη σχεδίαση και την ανάπτυξη μηχανισμού εντοπισμού σημείων συμφόρησης και προσαρμογής του φόρτου κίνησης σε ετερογενή δίκτυα, στο επίπεδο προώθησης πακέτων (επίπεδο 2 του OSI). Σε τοπολογίες όπου υπάρχουν κοινά μονοπάτια δρομολόγησης των ροών που εισάγονται στο δίκτυο, χρησιμοποιούμε το λογισμικό Software Defined Networking – SDN με σκοπό να αναπτύξουμε αλγορίθμους που δίνουν την δυνατότητα ελέγχου όλων των ροών μέσα σε ένα δίκτυο (flow control) για να επιτευχγθεί δίκαιη κατανομή των πόρων και του φόρτου σε πραγματικό χρόνο.

Τα κοινά μονοπάτια δρομολόγησης τα οποία καταλήγουν σε δίκτυα χαμηλής χωρητικότητας τα οποία είναι ευάλωτα στις συνθήκες που επικρατούν κάθε φορά σε αυτά, οδηγούν σε υπερφόρτωση του δικτύου και μείωση της ποιότητας των παρεχόμενων υπηρεσιών των ζεύξεων που τα χρησιμοποιούν, καθώς σε αυτά παρατηρείται έντονα το φαινόμενο της συμφόρισης, με αποτέλεσμα να εμφανίζεται

- υπερφόρτωση των ενδιάμεσων συσκευών επιπέδου 2 όπως τα switches
- μόλυνση του δικτύου από πληροφορία που θα πεταχτεί τελικά στη διαδρομή
- σπατάλη τών πόρων του δικτύου

Επομένως εναπόκεινται σε μηχανισμούς ανωτέρου επιπέδου (επίπεδο δικτύου, επίπεδο μεταφοράς) η ανάπτυξη μηχανισμών ελέγχου σημείων συμφόρησης (congestion control) και ελέγχου όλων των ροών που εισέρχονται σε ένα δίκτυο (flow control). Αυτή ακριβώς είναι και η δουλεία των μηχανισμών που εντοπίζονται και εφαρμόζονται στο TCP πρωτόκολλο. Τέτοιοι μηχανισμοί όμως παραλείπονται σε πρωτόκολλα όπως το UDP που δεν μπορούν να εκτιμήσουν την πραγματική χωρητικότητα της συνολικής ζεύξης ανά πάσα στιγμή στο δίκτυο. Έρευνες δείχνουν όμως πως στο διαδίκτυο ένα μεγάλο ποσοστό της κίνησης που εισάγεται δημιουργείται από εφαρμογές που χρησιμοποιούν το UDP όπως είναι audio και video streaming, και αυτό το ποσοστό συνεχώς αυξάνεται. Για αυτό το λόγο στη διπλωματική εργασία εστιάσαμε στη δημιουργία μηχανισμών flow control για την κίνηση σε UDP.

Με την βοήθεια της τεχνολογίας των Software Defined Networks (SDN) και πρωτοκόλλων όπως το OpenFlow δημιουργήσαμε μηχανισμούς οι οποίοι προσαρμόζουν σε πραγματικό χρόνο το φόρτο που εισάγει κάθε ροή στο δίκτυο μέσω της εκτίμησης του πραγματικού throughput που μπορεί να επιτευγχθεί ανά πάσα χρονική στιγμή στην

τοπολογία. Στη συνέχεια βασιστήκαμε πάνω σε αυτόν και δημιουργήσαμε αλγορίθμους που εφαρμόζουν flow control για όλες τις ροές του δικτύου χρησιμοποιώντας ως παραμέτρους την ρυθμοαπόδοση μιας ροής και την ικανοποίηση που λαμβάνει ο χρήστης που δημιουργεί την ροή στο δίκτυο.

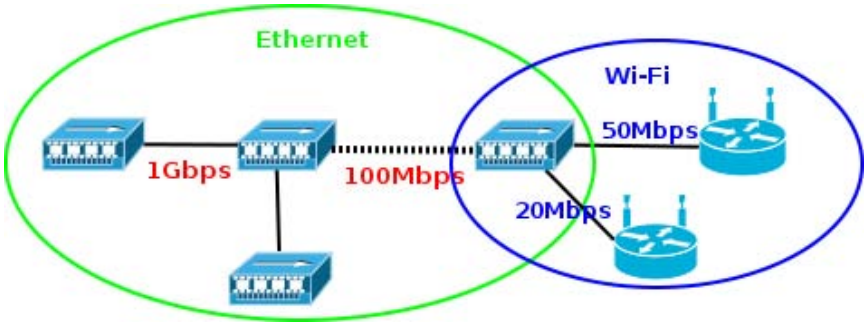
Λέξεις κλειδιά

Ετερογενή δίκτυα, Δικτύωση οριζόμενη από λογισμικό , Πρωτόκολλο Openflow , Συμφόριση , Δρομολόγηση , Μηχανισμός Ανάθεσης Ροών , Υπολογισμός ρυθμοαπόδοσης , Περιορισμός Ρυθμού ζεύξης, Χρήστες, SDN , UDP , TCP

1.Εισαγωγή

1.1 Διατύπωση του Προβλήματος

Το πρόβλημα που προσπαθήσαμε να αντιμετωπίσουμε στην παρούσα Διπλωματική εργασία, εμφανίζεται σε ετερογενή δίκτυα (1), δίκτυα δηλαδή που αποτελούνται από ενσύρματες ζεύξεις υψηλής χωρητικότητας και ζεύξεις χαμηλής χωρητικότητας όπως είναι οι ασύρματες. Σε τέτοιες τοπολογίες (**Σχήμα 1**) πρωτόκολλα όπως το UDP δεν μπορούν να εκτιμήσουν την πραγματική χωρητικότητα της συνολικής ζεύξεις που υπάρχει κάθε στιγμή σε κάθε σημείο του δικτύου με αποτέλεσμα να μειώνεται το quality of service (qos) πολλών εισερχόμενων ροών που χρησιμοποιούν ένα κοινό μονοπάτι δρομολόγησης όταν μία και μόνο ζεύξη υπερφορτώνει αυτό το κοινό μονοπάτι.



Σχήμα 1: Τοπολογία δικτύου με προβλήματα συμφόρισης

Τα θέματα που αναδύονται από μια τέτοια τοπολογία είναι:

- υπερφότωση ενδιάμεσων μεταγωγέων
- υπερφόρτωση ενδιάμεσων δρομολογητών
- σπατάλη πόρων δικτύου
- μόλυνση δικτύου από πακέτα που δεν καταλήγουν στον παραλήπτη
- λιμοκτονία κάποιων ροών του δικτύου

Παρακάτω εστιάζουμε σε περιπτώσεις όπου πολλαπλές ροές από διαφορετικούς χρήστες καταλήγουν μέσω μιας κοινής διαδρομής μέσω του ενσύρματου δικτύου σε ένα δρομολογητή – access point και μέσω αυτού χρησιμοποιούν κάποιο ασύρματο κανάλι του 802.11 ώστε να φτάσουν στους αντίστοιχους παραλήπτες. Επειδή όμως όπως γνωρίζουμε στο ασύρματο κανάλι η χωρητικότητα είναι περιορισμένη και επηρεάζεται από διάφορες συνθήκες, όπως είναι η απόσταση , εμπόδια , αριθμός χρηστών κ.λ.π , τελικά αρκετοί από τους παραλήπτες δεν καταφέρνουν να λάβουν τα πακέτα που προορίζονται για αυτούς και αντίστοιχα ο χρήστης δεν μπορεί να εξυπηρετηθεί. Έτσι υπάρχει περίπτωση κάποιοι χρήστες να καταχράζονται όλο το κανάλι και κάποιοι να μην μπορούν να το χρησιμοποιήσουν. Επομένως κάποιοι λιμοκτονούν ενώ κάποιοι άλλοι μεταδίδουν συνεχώς. Παρόλο που τα πακέτα τελικά πετιούνται στο ασύρματο κανάλι εφόσον δεν μπορούν να υποστηριχθούν, διασχίζουν όλο το μονοπάτι που περνάει από το ενσύρματο μέσο και δημιουργεί εκεί πρόβλημα σε άλλες ροές που μοιράζονται το ίδιο μονοπάτι καθώς δεν υπάρχει εκτίμησης της συνολικής χωρητικότητας μιας ζεύξης.

2. Επίπεδα λειτουργιών σε Μοντέλα Αναφοράς Αρχιτεκτονικής Δικτύου

Στις κλασσικές προτυποποιημένες αρχιτεκτονικές δικτύωσης, η κάθε δικτυακή συσκευή αποτελούσε μια αυτόνομη οντότητα. Εσωτερικά της, για λόγους οργάνωσης και απόδοσης, οι απαιτούμενοι μηχανισμοί ήταν σχεδιασμένοι και υλοποιημένοι σε διακριτές ομάδες με βασικότερες αυτές που αφορούσαν τις λειτουργίες προώθησης των δεδομένων **(forwarding/data plane)**, τις λειτουργίες διαχείρισης **(management)** καθώς και τις λειτουργίες ελέγχου **(control)**.

Οι μηχανισμοί που υλοποιούσαν τις προαναφερθείσες ομάδες λειτουργιών

χρησιμοποιούσαν ιδιοταγείς (proprietary) διεπαφές μεταξύ τους που αναπτύσσονταν από κάθε κατασκευαστή δικτυακών συσκευών χωριστά, χωρίς καμία δυνατότητα επιλογών, ειδικά στην περίπτωση της διεπαφής μεταξύ λειτουργιών ελέγχου και προώθησης δεδομένων. Το συγκεκριμένο γεγονός ήρθε να ανατρέψει η εμφάνιση αρχιτεκτονικών όπως είναι η ForCES [RFC3746] και το πρωτόκολλο OpenFlow, το οποίο αποτελεί και αντικείμενο μελέτης της διατριβής και αναλύεται στα βασικά του σημεία στην Παράγραφο 2.4.

2.1 Επίπεδο Προώθησης Δεδομένων (Forwarding/Data-plane)

Το επίπεδο προώθησης δεδομένων (2), παραδοσιακά, υλοποιείται τοπικά σε κάθε δικτυακή συσκευή και λειτουργεί με την ταχύτητα άφιξης των πακέτων (line-rate).

Μια βασική λειτουργία, που υλοποιείται σε δρομολογητές για παράδειγμα, είναι η προώθηση των πακέτων σε κάποια διεπαφή εξόδου (egress interface) βάσει κάποιων κανόνων δρομολόγησης (π.χ longest-prefix match) και ο ταυτόχρονος έλεγχος πλήρωσης κριτηρίων φιλτραρίσματος (Access Control Lists - ACLs).

Στο συγκεκριμένο επίπεδο υλοποιούνται επίσης λειτουργίες ελέγχου ροής πακέτων με χρήση ουρών (queue management) και προγραμματισμού πακέτων (packet scheduling). Το κύριο χαρακτηριστικό είναι ότι οι προαναφερθείσες λειτουργίες υλοποιούνται με χρήση εξειδικευμένου υλισμικού (hardware).

Αν και οι υλοποιήσεις του επιπέδου προώθησης πακέτου διαφέρουν ανά κατασκευαστή, η επικοινωνία μεταξύ συσκευών διαφορετικών κατασκευαστών είναι δυνατή λόγω των προτυποποιημένων πρωτοκόλλων (standardized) προώθησης δεδομένων (π.χ Ethernet, Internet Protocol).

2.2 Επίπεδο Ελέγχου (Control-Plane)

Το επίπεδο ελέγχου είναι υπεύθυνο να καθορίζει τη συμπεριφορά του επιπέδου προώθησης δεδομένων εφαρμόζοντας κατάλληλους αλγορίθμους. Η λειτουργία του επιπέδου μπορεί να είναι κεντρικοποιημένη (centralized) ή κατανεμημένη (distributed). Στην περίπτωση που είναι κεντρικοποιημένη, οι αποφάσεις λαμβάνονται σε ένα σημείο, αλλά αφορούν όλο το δίκτυο, ενώ στην περίπτωση της κατανεμημένης λειτουργίας του επιπέδου ελέγχου οι αλγόριθμοι που χρησιμοποιούνται εφαρμόζονται σε κάθε δικτυακή συσκευή που λαμβάνει μέρος στο επίπεδο ελέγχου.

Μια από τις κύριες λειτουργίες του επιπέδου ελέγχου είναι να υπολογίζει διαδρομές

δεδομένων συνδυάζοντας πληροφορίες από κάθε πρωτόκολλο δρομολόγησης, δημιουργώντας το Forwarding Information Base (FIB) που τελικά χρησιμοποιείται από το επίπεδο προώθησης δεδομένων.

Κατανεμημένο Επίπεδο Ελέγχου (Distributed Control-Plane)

Οι αλγόριθμοι κατανεμημένου ελέγχου απαιτούν την ανταλλαγή πληροφοριών μεταξύ των δικτυακών συσκευών που συμμετέχουν στον έλεγχο της προώθησης πακέτων. Τα πρωτόκολλα του κατανεμημένου επιπέδου ελέγχου περιγράφουν τη διαδικασία ανταλλαγής μηνυμάτων καθώς και τον αλγόριθμο που χρησιμοποιείται για την λήψη των αποφάσεων (**Σχήμα 2**) (3).

Για παράδειγμα τα μηνύματα ενημέρωσης (update messages) και η διαδικασία λήψης αποφάσεων δρομολόγησης του πρωτοκόλλου διατομεακής δρομολόγησης (inter-domain routing) Border Gateway Protocol (BGP) είναι χαρακτηριστικά στοιχεία του κατανεμημένου επιπέδου ελέγχου. Ενδεικτικά αναφέρουμε ότι στο συγκεκριμένο επίπεδο ανήκουν και οι ενημερώσεις για την κατάσταση διεπαφών (Link State Advertisements - LSAs) και ο αλγόριθμος Dijkstra που αποτελούν μέρος του πρωτοκόλλου Open Shortest Path First (OSPF).

Στην συντριπτική πλειοψηφία των δικτυακών συσκευών (π.χ. δρομολογητές) που το επίπεδο ελέγχου είναι κατανεμημένο, η επικοινωνία μεταξύ του επιπέδου ελέγχου της εκάστοτε συσκευής με το επίπεδο προώθησης δεδομένων γίνεται ακόμα και σήμερα μέσω ιδιοταγών (proprietary) διεπαφών. Η συγκεκριμένη χρήση ιδιοταγών διεπαφών καθιστά τις υλοποιήσεις των δικτυακών συσκευών ένα κλειστό σιλό τεχνολογιών ενός κατασκευαστή, μια προσέγγιση, που στον κόσμο των υπολογιστικών συστημάτων γενικής χρήσης έχει ξεπεραστεί εδώ και δυο δεκαετίες.

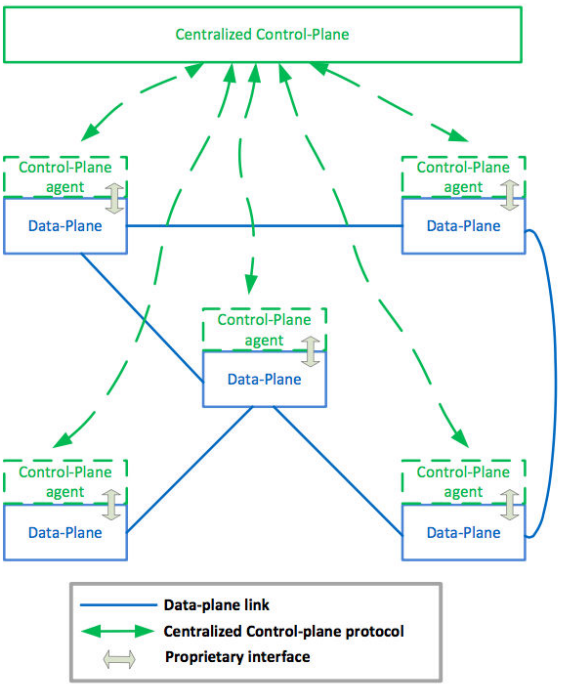
Σχήμα 2: Κατανεμημένο Επίπεδο Ελέγχου

Κεντρικοποιημένο Επίπεδο Ελέγχου (Centralized Control-Plane)

Την τελευταία δεκαετία, άρχισαν να κερδίζουν έδαφος προσεγγίσεις που προωθούσαν την αποσύζευξη (decoupling) του επιπέδου ελέγχου με αυτό της προώθησης δεδομένων (**Σχήμα 3**). Στην κυριαρχούσα κεντρικοποιημένη αρχιτεκτονική επιπέδου ελέγχου (4) υπάρχει μια κεντρική οντότητα που υπαγορεύει τον τρόπο προώθησης των πακέτων στο δίκτυο. Το δίκτυο προγραμματίζεται με την χρήση μιας ομάδας εντολών που έχει ως σκοπό την εισαγωγή ροών που χρησιμοποιούνται από το επίπεδο προώθησης πακέτων των συσκευών. Με αυτό τον τρόπο η διαχείριση και η λειτουργία των δικτυακών συσκευών γίνεται απλούστερη, σε αντιδιαστολή με αυτές των κεντρικών οντοτήτων ελέγχου που συγκεντρώνουν την πολυπλοκότητα.

Παράλληλα με το διαχωρισμό λειτουργιών ελέγχου και προώθησης δεδομένων άρχισε να προωθείται και ο ορισμός ανοιχτών/προτυποποιημένων (open/standardized) διεπαφών, κατά αντιστοιχία με αυτές που είχαν ήδη δημιουργηθεί στα λειτουργικά συστήματα υπολογιστών [Hjal00] [Pete00] [Kohl00]. Το πρώτο γενικευμένο πλαίσιο που

αναπτύχθηκε ήταν το Forwarding and Control Element Separation (ForCES) Framework, που όριζε την διαίρεση των δικτυακών οντοτήτων (network elements) σε οντότητες ελέγχου (control elements) και προώθησης δεδομένων (forwarding elements) καθώς και τις αναμεταξύ τους διεπαφές.



Σχήμα 3: Κεντρικοποιημένο Επίπεδο λέγχου

2.3 Επίπεδο Διαχείρισης (Managment-Plane)

Για το επίπεδο διαχείρισης δικτύου έχουν αναπτυχθεί πολλαπλά μοντέλα τα 30 τελευταία χρόνια από διαφορετικά σώματα τυποποίησης όπως είναι ο Διεθνής Οργανισμός Τυποποίησης (International Organization for Standardization - ISO), η Διεθνής Ένωση Τηλεπικοινωνιών (International Telecommunication Union - ITU) καθώς και η Ομάδα Έργου Μηχανίκευσης Ίντερνετ (Internet Engineering Task Force – IETF) (5).

3. Εργαλεία και Τεχνολογίες που χρησιμοποιήσαμε

3.1 Πρωτόκολλο OpenFlow (OpenFlow Protocol)

Το Openflow είναι ένα ανοιχτό standard που επιτρέπει στους ερευνητές να δοκιμάζουν νέα πρωτόκολλα και νέες τεχνολογιές. Συγκεκριμένα είναι ένα πρωτόκολλο επικοινωνίας που δίνει πρόσβαση στο forwarding plane ενός openflow switch (6).
Το Forwarding and Control Element Separation (ForCES) Framework είχε εισάγει την ιδέα του διαχωρισμού μεταξύ επιπέδου ελέγχου και επιπέδου προώθησης πακέτων. Στη συνέχεια το OpenFlow καθόρισε μια ολοκληρωμένη διεπαφή μεταξύ των δυο επιπέδων. Κάνοντας το συγκεκριμένο διαχωρισμό επέτρεψε την ταχύτερη ανάπτυξη των οριζόμενων από λογισμικό δικτύων (Software Defined Networking – SDN), που αποτελεί ένα από τα κύρια αντικείμενα ενασχόλησης του Open Networking Foundation (ONF).

Η αρχική ανάπτυξη του πρωτοκόλλου ξεκίνησε στον ακαδημαϊκό χώρο. Το ONF μετέπειτα ανέλαβε την εξέλιξη του πρωτοκόλλου και ολόκληρου του οικοσυστήματος δικτύωσης που αναπτύσσεται γύρω του, όπως οι δοκιμές και οι πιστοποιήσεις υλισμικού για τη συμβατότητά τους με το OpenFlow specification.

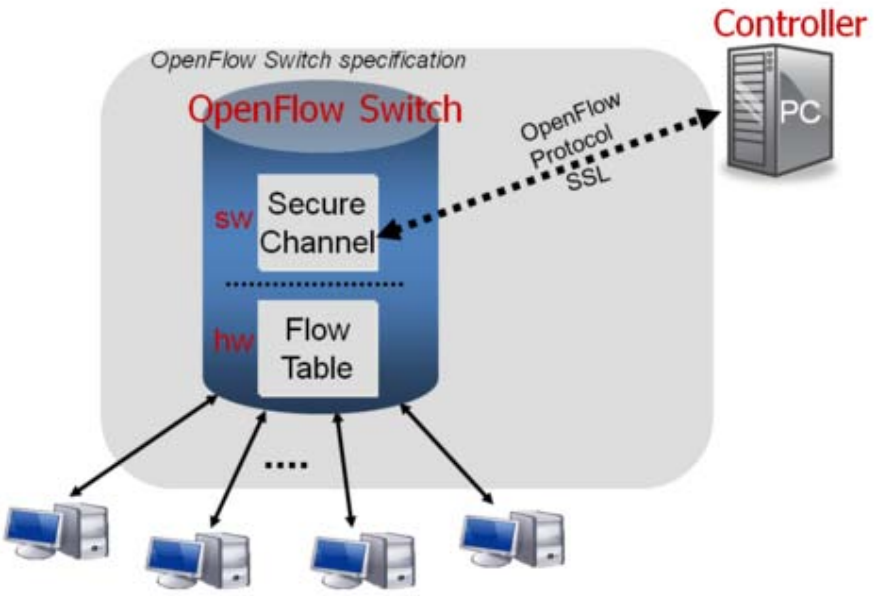
Το OpenFlow μπορεί να θεωρηθεί το ανάλογο του συνόλου εντολών (instruction set) ενός επεξεργαστή για δικτυακές συσκευές. Δίνει την δυνατότητα σε λογισμικό να προγραμματίσει τους δικτυακούς επεξεργαστές που συμμετέχουν στο επίπεδο προώθησης πακέτων, ανεξαρτήτως της εσωτερικής αρχιτεκτονικής των δικτυακών επεξεργαστών.

Ένα βασικό χαρακτηριστικό του είναι ότι όσες συσκευές είναι συμβατές μπορούν να υλοποιούν προώθηση πακέτων λαμβάνοντας υπόψη πολλαπλές επικεφαλίδες πρωτοκόλλων διαφορετικών επιπέδων της στοίβας πρωτοκόλλων (network protocol stack). Η συγκεκριμένη ιδιότητα επιτρέπει την προώθηση ροών (flow forwarding) με σύνθετα κριτήρια και όχι απλώς την προώθηση πακέτων (packet forwarding) βάσει των επικεφαλίδων ενός συγκεκριμένου πρωτοκόλλου (π.χ Internet Protocol). Η συγκεκριμένη δυνατότητα δινόταν μέχρι σήμερα από διάφορους κατασκευαστές, αλλά δεν υπήρχε κάποια ευρέως υιοθετημένη λύση.

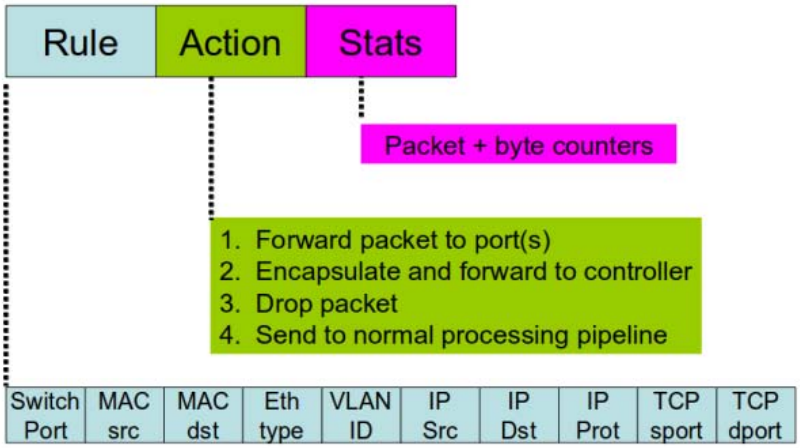
Τεχνική Περιγραφή OpenFlow (OpenFlow Specification)

Μια δικτυακή συσκευή του επιπέδου προώθησης δεδομένων που υποστηρίζει το OpenFlow protocol ονομάζεται **OpenFlow switch** (9), σύμφωνα με το OpenFlow Specification. Τα OpenFlow switches ελέγχονται μέσω του δικτυακού πρωτοκόλλου

OpenFlow protocol (7) από μια ανεξάρτητη συσκευή που ανήκει στο επίπεδο ελέγχου και ονομάζεται **OpenFlow Controller**, όπως απεικονίζεται στο (**Σχήμα 4**). Ο OpenFlow controller μπορεί να ελέγχει τον μηχανισμό προώθησης πακέτων που βρίσκεται υλοποιημένος μέσα στα OpenFlow switches καθορίζοντας τους κανόνες των πινάκων προώθησης που ονομάζονται **Flow tables** (**Σχήμα 5**). Σε περίπτωση που τα Flow tables είναι πολλαπλά, η προσπέλασή τους είναι μονόδρομη. Η προσπέλασή τους, δηλαδή, κατά την διαδικασία ταύτισης (matching) ενός πακέτου, γίνεται μόνο προς τα εμπρός, δίχως δυνατότητα μεταπήδησης από επόμενο flow table σε προηγούμενο.



Σχήμα 4: Ελεγχόμενο switch μέσω Openflow Controller



Σχήμα 5: Κανόνας μέσα στον Πίνακα Προώθησης ενός Openflow switch

Τα OpenFlow switches θα πρέπει να έχουν την ικανότητα να προωθούν Ethernet frames βάσει ενός συνόλου κανόνων που είναι αποθηκευμένοι σε ένα ή περισσότερα Flow tables. Τα βασικά στοιχεία κάθε εγγραφής (**Flow entry**) στον Flow table είναι:

- ένα πεδίο ταύτισης (**Match Field)**
- ένα σύνολο κανόνων (**Actions)** που καθορίζουν την τύχη του πακέτου
- μετρητές (**Counters**) που ανανεώνονται κάθε φορά που ένα πακέτο ταιριάζει στο πεδίο ταύτισης
- ένα πεδίο προτεραιότητας του flow entry (**Priority**) και ένα πεδίο χρονικού ορίου λήξεως της ισχύος του κανόνα (**Time-out**).

Ένας OpenFlow Controller μέσω του OpenFlow protocol μπορεί να εισάγει, αφαιρεί, ανανεώνει flow entries που βρίσκονται σε ένα διασυνδεδεμένο με αυτόν OpenFlow switch. Υπάρχει η δυνατότητα εισαγωγής νέων κανόνων προληπτικά (proactively), πριν την άφιξη δηλαδή κάποιου πακέτου που ταιριάζει στο εκάστοτε flow entry. Δίνεται επίσης η δυνατότητα χειρισμού ενός πακέτου που δεν αντιστοιχεί με κάποιον ήδη εγκαθιδρυμένο κανόνα, αφού το OpenFlow Switch έχει τη δυνατότητα να αποθηκεύσει προσωρινά ένα πακέτο και να ρωτήσει τον OpenFlow Controller για τον τρόπο χειρισμού του πακέτου. Η ερώτηση αυτή ουσιαστικά απαντάται με την εγκαθίδρυση ενός καινούριου OpenFlow entry από πλευράς OpenFlow Controller στο switch.

Τα Actions, που περιλαμβάνονται ως μέρος του κάθε Flow entry, δύνανται να υπαγορεύουν την έξοδο του πακέτου από μια πόρτα (forwarding), την απόρριψή του (dropping), την τροποποίηση (modification) κάποιων επικεφαλίδων πρωτοκόλλων που λαμβάνει υπόψη του το OpenFlow ως πεδία ταύτισης ή ακόμα και την μεταφορά της απόφασης σε κάποιο άλλο Flow table που ακολουθεί στην αλυσίδα των Flow tables.

3.2 Software Defined Networking (SDN)

Το SDN είναι μια αναδυόμενη αρχιτεκτονική που είναι δυναμική, εύχρηστη, αποδοτική, και προσαρμόσιμη, καθιστώντας την ιδανική για την υψηλού εύρους ζώνης, δυναμική φύση των σημερινών εφαρμογών (8). Αυτή η αρχιτεκτονική διαχωρίζει τον έλεγχο του δικτύου (control plane) από την προώθηση λειτουργιών (data plane) και δίνει την δυνατότητα στο δικτύου να γίνει άμεσα προγραμματιζόμενο και εισάγει την έννοια της

ευελιξίας στα δίκτυα. Το πρωτόκολλο OpenFlow που αναλύσαμε προηγουμένως είναι ένα θεμελιώδες στοιχείο για την οικοδόμηση του SDN. Η αρχιτεκτονική SDN είναι:

- Άμεσα προγραμματιζόμενη: Ο έλεγχος δικτύου είναι άμεσα προγραμματιζόμενος επειδή είναι αποσυνδεδεμένος από τις λειτουργίες προώθησης.
- Ευέλικτο: Επιτρέπει στους διαχειριστές να προσαρμόζουν δυναμικά το δίκτυο με βάση όλη την ροή της κυκλοφορίας ώστε να ανταποκρίνονται στις μεταβαλλόμενες ανάγκες.
- Κεντρική διαχείριση: Η νοημοσύνη του δικτύου είναι συγκεντρωμένη σε λογισμικό που βασίζεται σε ελεγκτές SDN (controllers SDN) που διατηρούν μια συνολική εικόνα του δικτύου.
- Προγραμματισμή ρύθμιση: Το SDN επιτρέπει στους διαχειριστές δικτύων την διαχείριση, ασφαλεια, και τη βελτιστοποίηση των πόρων του δικτύου πολύ γρήγορα μέσω δυναμικών, αυτοματοποιημένων προγράμματων SDN, τα οποία μπορούν να γράφουν οι ίδιοι, επειδή τα προγράμματα δεν εξαρτώνται από το ιδιόκτητο λογισμικό.
- Ανοικτά πρότυπα (Open Standards Based): Η υλοποιηθεί μέσω ανοικτών προτύπων, στο SDN απλοποιεί το σχεδιασμό και τη λειτουργία του δικτύου, επειδή οι οδηγίες πλέον παρέχονται από τους ελεγκτές SDN αντί των κατασκευαστών εξειδικευμένων συσκευών hardware ή πρωτοκόλλων.

3.3 Open vSwitch – Εικονικοί Μεταγωγείς

Το Open vSwitch είναι ένας πολυστρωματικός εικονικός διακόπτης - μεταγωγέας (12). Έχει σχεδιαστεί για να επιτρέψει τη μαζική αυτοματοποίηση του δικτύου μέσα από προγραμματιστικές επεκτάσεις, ενώ υποστηρίζει ακόμα τυποποιημένες διεπαφές διαχείρισης και πρωτόκολλα (π.χ. NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag). Επιπλέον, έχει σχεδιαστεί για να υποστηρίξει τη διανομή σε πολλούς φυσικούς εξυπηρετητές.

3.4 Testbed Experimentation

Το Testbed (10) είναι μια πειραματική πλατφόρμα δοκιμών (κλίνης δοκιμών) δηλαδή μια πλατφόρμα για τη διεξαγωγή αυστηρού, διαφανή και αναπαραγόμενου ελέγχου των επιστημονικών θεωριών, υπολογιστικών εργαλείων, νέων αλγορίθμων και τεχνολογιών. Ο όρος χρησιμοποιείται σε πολλούς επιστημονικούς κλάδους για να περιγράψει την πειραματική έρευνα και τις νέες πλατφόρμες ανάπτυξης προϊόντων και περιβάλλοντα.

Στον τομέα των Τηλεπικοινωνιών και Δικτύων χρησιμοποιούνται τα Testbeds για να αναπαραγάγουν τοπολογίες και προβλήματα σε πραγματικό περιβάλλον και σε συνθήκες που δεν ορίζονται μόνο από μαθηματικά μοντέλα και μεταβλητές. Με αυτό τον τρόπο δίνεται η δυνατότητα στους ερευνητές να τρέξουν τα πειράματα τους και να δοκιμάσουν νέους αλγορίθμους σε συνθήκες που ανταποκρίνονται στο πραγματικό διαδίκτυο.

NITOS Testbed

Στα πλαίσια της διπλωματικής μου εργασίας χρησιμοποίησαμε το NITOS Wireless Testbed (11) του εργαστηρίου NITLab που αποτελείται από ασύρματους και ενσύρματους κόμβους, βασιζόμενους σε ανοιχτό λογισμικό. Στους κόμβους δημιουργήσαμε τοπολογίες στις οποίες στη συνέχεια τρέξαμε τους αλγορίθμους και συγκεντρώσαμε τα αποτελέσματα των πειραμάτων. Επιλέξαμε να τρέξουμε τους αλγορίθμους μας στο Testbed για την υλοποίηση και την αξιολόγηση τους , αντί για κάποιον εξομοιωτή γιατί θέλαμε να τους εφαρμόσουμε σε πραγματικές συσκευές και σε πραγματικές συνθήκες δικτύου.

4. Πρωτόκολλα Επιπέδου Μεταφοράς

4.1 Πρωτόκολλο Ελέγχου Μετάδοσης - Transmission Control Protocol (TCP)

TCP / IP Περιγραφή

Το '"TCP/IP" (Transmission Control Protocol/Internet Protocol=Πρωτόκολλο Ελέγχου Μετάδοσης και πρωτόκολλο του Internet) (13) είναι μια συλλογή πρωτοκόλλων επικοινωνίας στα οποία βασίζεται το διαδίκτυο, αλλά και μεγάλο ποσοστό των εμπορικών δικτύων. Η ονομασία TCP/IP προέρχεται από τις συντομογραφίες των δυο κυριότερων πρωτοκόλλων που περιέχει το TCP ή Transmission Control Protocol *(Πρωτόκολλο Ελέγχου Μετάδοσης)* και το IP ή Internet Protocol *(Πρωτόκολλο Διαδικτύου)*. Είναι ένα πρωτόκολλο επιπέδου μεταφοράς που χρησιμοποιείται από τις εφαρμογές που απαιτούν εγγυημένη παράδοση.

Αυτή η συλλογή πρωτοκόλλων, όπως και πολλές άλλες άλλωστε, είναι οργανωμένη σε στρώματα ή επίπεδα (layers). Το καθένα τους απαντά σε συγκεκριμένα προβλήματα μεταφοράς δεδομένων και παρέχει μια καθορισμένη υπηρεσία στα υψηλότερα στρώματα. Τα ανώτερα επίπεδα είναι πιο κοντά στη λογική του χρήστη και εξετάζουν πιο αφηρημένα δεδομένα, στηριζόμενα σε πρωτόκολλα χαμηλότερων στρωμάτων για να μεταφράσουν δεδομένα σε μορφές που μπορούν να διαβιβαστούν με φυσικά μέσα.

Έλεγχος συμφόρησης TCP

Το TCP χρησιμοποιεί ένα παράθυρο συμφόρησης (congestion window) (14) στην πλευρά του αποστολέα ώστε να αποφύγει την κυκλοφοριακή συμφόρησης. Το παράθυρο συμφόρησης δείχνει τη μέγιστη ποσότητα δεδομένων που μπορεί να σταλεί σε μια σύνδεση χωρίς να αναγνωρίζεται. Το TCP ανιχνεύει συμφόρηση όταν αποτυγχάνει να λάβει μια επιβεβαίωση (Acknowledgment) για ένα πακέτο στην εκτίμηση του χρονικού ορίου. Σε μια τέτοια κατάσταση, μειώνει το παράθυρο συμφόρησης σε ένα μέγιστο μέγεθος τμήματος (MSS), και σύμφωνα με άλλες περιπτώσεις αυξάνει το παράθυρο συμφόρησης κατά ένα MSS. Υπάρχει επίσης ένα όριο στο παράθυρο συμφόρησης, το οποία έχει οριστεί ίσο με το ήμισυ του μεγέθους του παραθύρου συμφόρησης κατά το χρόνο εκ νέου εκπομπής.

Συγκεκριμένα χρησιμοποιεί μια εκτίμηση καθυστέρησης μετ' επιστροφής (rtt), για το προσαρμοστικό σύστημα παραθύρων που χρησιμοποιεί για να μεταδίδει τα δεδομένα αξιόπιστα πάνω από ένα αναξιόπιστο δίκτυο με διαφορετικό εύρος χρόνου.

4.2 User Datagram Protocol (UDP)

UDP Περιγραφή

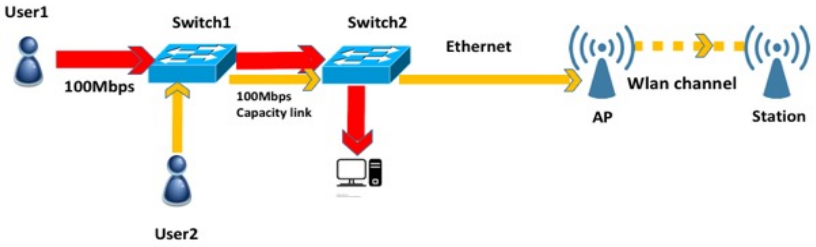
Το πρωτόκολλο User Datagram Protocol (UDP) (15) είναι ένα από τα βασικά πρωτόκολλα που χρησιμοποιούνται στο διαδίκτυο. Μία εναλλακτική ονομασία του πρωτοκόλλου είναι Universal Datagram Protocol. Διάφορα προγράμματα χρησιμοποιούν το πρωτόκολλο UDP για την αποστολή σύντομων μηνυμάτων (γνωστών και ως datagrams) από τον έναν υπολογιστή στον άλλον μέσα σε ένα δίκτυο υπολογιστών.

Ένα από τα κύρια χαρακτηριστικά του UDP είναι ότι δεν εγγυάται αξιόπιστη επικοινωνία. Τα πακέτα UDP που αποστέλλονται από έναν υπολογιστή μπορεί να φτάσουν στον παραλήπτη με λάθος σειρά, διπλά ή να μην φτάσουν καθόλου εάν το δίκτυο έχει μεγάλο φόρτο. Αντιθέτως, το πρωτόκολλο TCP διαθέτει όλους τους απαραίτητους μηχανισμούς ελέγχου και επιβολής της αξιοπιστίας και συνεπώς μπορεί να εγγυηθεί την αξιόπιστη επικοινωνία μεταξύ των υπολογιστών. Η έλλειψη των μηχανισμών αυτών από το πρωτόκολλο UDP το καθιστά αρκετά πιο γρήγορο και αποτελεσματικό, τουλάχιστον για τις εφαρμογές εκείνες που δεν απαιτούν αξιόπιστη επικοινωνία.

Οι εφαρμογές audio και video streaming χρησιμοποιούν κατά κόρον πακέτα UDP (16). Για τις εφαρμογές αυτές είναι πολύ σημαντικό τα πακέτα να παραδοθούν στον παραλήπτη σε σύντομο χρονικό διάστημα ούτως ώστε να μην υπάρχει διακοπή στην ροή του ήχου ή της εικόνας. Κατά συνέπεια προτιμάται το πρωτόκολλο UDP διότι είναι αρκετά γρήγορο, παρόλο που υπάρχει η πιθανότητα μερικά πακέτα UDP να χαθούν. Στην περίπτωση που χαθεί κάποιο πακέτο, οι εφαρμογές αυτές διαθέτουν ειδικούς μηχανισμούς διόρθωσης και παρεμβολής ούτως ώστε ο τελικός χρήστης να μην παρατηρεί καμία αλλοίωση ή διακοπή στην ροή του ήχου και της εικόνας λόγω του χαμένου πακέτου. Σε αντίθεση με το πρωτόκολλο TCP, το UDP υποστηρίζει broadcasting, δηλαδή την αποστολή ενός πακέτου σε όλους τους υπολογιστές ενός δικτύου, και multicasting, δηλαδή την αποστολή ενός πακέτου σε κάποιους συγκεκριμένους υπολογιστές ενός δικτύου. Η τελευταία δυνατότητα χρησιμοποιείται πολύ συχνά στις εφαρμογές audio και video streaming ούτως ώστε μία ροή ήχου ή εικόνας να μεταδίδεται ταυτόχρονα σε πολλούς συνδρομητές.

5. Τοπολογία σε περιβάλλον συμφόρησης (Underperformance in congested environment)

Για να μελετήσουμε το πρόβλημα στο οποίο αναφερθήκαμε στο Κεφάλαιο 1 Παράγραφο 1.1 της Εισαγωγής , αναπαραστήσαμε μια τοπολογία δικτύου στο Testbed χρησιμοποιώντας 6 κόμβους όπως φαίνεται στο παρακάτω Σχήμα **(Σχήμα 6).**



1

Σχήμα 6:Αναπαράσταση ετερογενούς δικτύου με πρόβλημα συμφόρησης

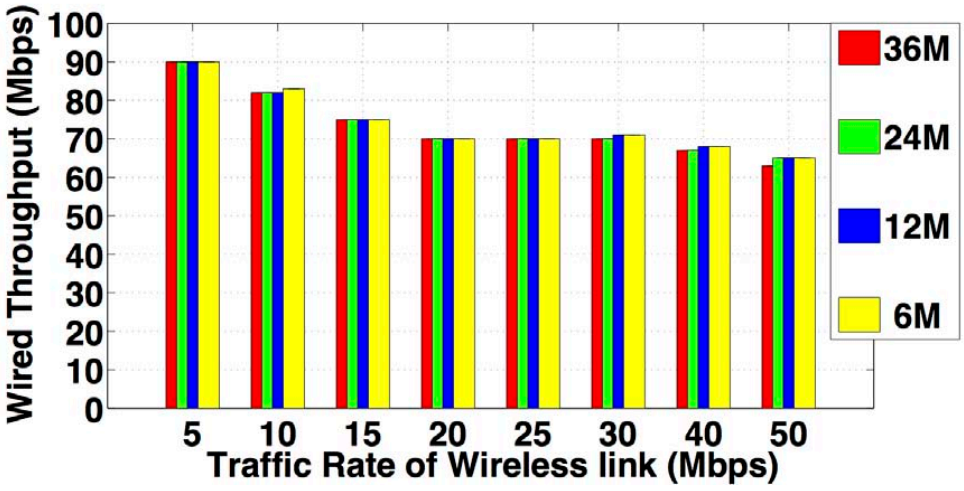
Περιγραφή Σεναρίου

Σε αυτή την τοπολογία δημιουργήσαμε το σενάριο ενός ετερογενούς δικτύου που αποτελείται από δύο μεταγωγείς που συνδέονται με Ethernet χωρητικότητας 100Mbps ενώ ο τελευταίος μεταγωγέας συνδέεται μέσω Ethernet με έναν κόμβο που λειτουργεί ως Access Point και μεταδίδει μέσω του ασύρματου καναλιού σε έναν ασύρματο χρήστη. Στο ασύρματο κανάλι χρησιμοποιήσαμε το 802.11 πρωτόκολλο στα 2.4GHz και για να δημιουργήσουμε διαφορετικές συνθήκες καναλιού χρησιμοποιήσαμε τέσσερα διαφορετικά Physical Rates (17) . Τα rates που χρησιμοποιήσαμε είναι:

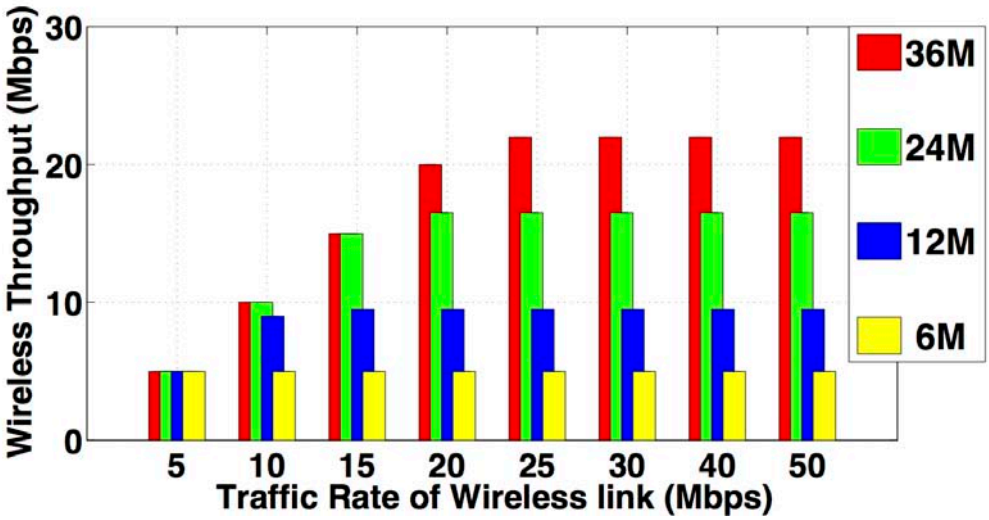
- 6Mbps
- 12Mbps
- 24Mbps
- 36Mbps

Για να εισάγουμε κίνηση , τοποθετήσαμε δυο χρήστες και τους συνδέσαμε στον πρώτο μεταγωγέα (switch1) μέσω του ενσύρματου. Ο πρώτος χρήστης (user1) εισάγει μια σταθερή **UDP** ροή στο δίκτυο, δλδ ο ρυθμός πακέτων που εισάγει είναι **100Mbps** (κόκκινο βελάκι) και στέλνει τα δεδομένα του στον δεύτερο μεταγωγέα (switch2) δλδ η διασύνδεση από άκρη σε άκρη είναι από τον χρήστη ένα του μεταγωγέα ένα έως τον μεταγωγέα δύο. Ενώ ο δεύτερος χρήστης εισάγει μια εναλλασσόμενη **UDP** ροή (κίτρινο βελάκι) η οποία διασχίζει όλο το μονοπάτι με τα διαφορετικά links για να φτάσει στον παραλήπτη του ασύρματου μέσου.

Για κάθε ένα από τα παραπάνω rates ο δεύτερος χρήστης (user 2) εισάγει διαφορετικό data rate , διαφορετικό **UDP** φόρτο κίνησης , ξεκινώντας από τα **5Mbps**, αυξάνοντας σταδιακά κατά 5Mbps και καταλήγοντας στα **50Mbps**. Σε κάθε ένα από αυτά τα σενάρια πήραμε τα αποτελέσματα της ρυθμοαπόδοσης (Troughput) που λαμβάνει κάθε παραλήπτης και στο ενσύρματο αλλά και στο ασύρματο μέσο. Οι τιμές από τα αποτελέσματα αυτά αναπαριστώνται στα παρακάτω γραφήματα.



Γράφημα 1:Troughput στον user1



Γράφημα 2:Troughput στον user2

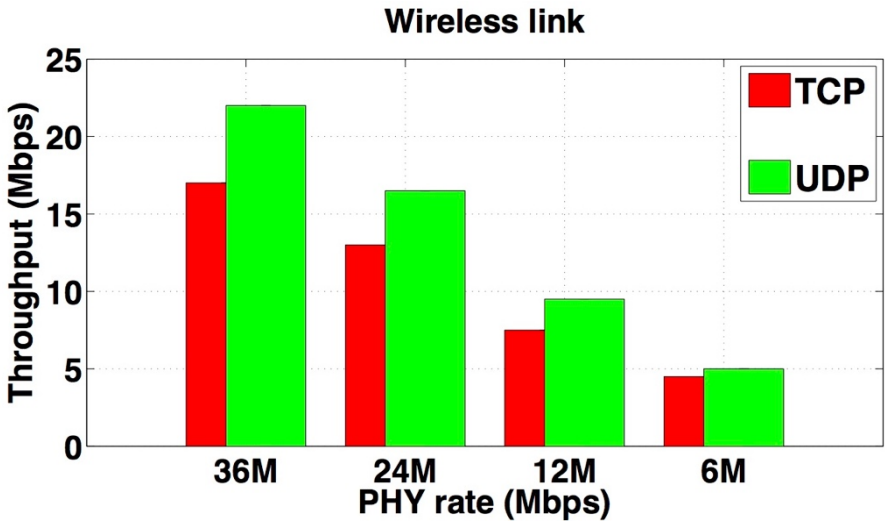
Το πρώρο γράφημα αναφέρεται στο ενσύρματο μονοπάτι και δείχνει την ρυθμοαπόδοση που πετυχαίνει η σταθερή ροή που εισάγει στο δίκτυο ο πρώτος χρήστης (user1) σε συνδιασμό με το data rate που εισσάγει ο δεύτερος χρήστης. Με διαφορετικά χρώματα δεξιά δείχνουμε τα διαφορετικά physical rates που χρησιμοποιήσαμε στα πειράματα.

Παρατηρώντας το γράφημα 2 συμπεραίνουμε ότι η απόδοση στο ασύρματο μέσο εξαρτάται άμεσα από τις συνθήκες του καναλιού , δλδ το physical rate που χρησιμοποιήσαμε. Συγκεκριμένα για το μικρότερο rate των 6Mbps το throughput του χρήστη 2 (user2) στο ασύρματο είναι σταθερά 5Mbps όσο κι αν είναι η ροή που εισάγει ο χρήστης 2. Ενώ αντίστοιχα στην καλύτερη περίπτωση των 36Mbps physical rate η μέγιστη απόδοση είναι 22Mbps. Επομένως αν πάρουμε την χειρότερη περίπτωση, όσα δεδομένα και να στείλει ο χρήστης 2, θα λαμβάνει πάντα το ίδιο throughput τελικά το οποίο είναι 5Mbps. Τα επιπλέον δεδομένα που εισέρχονται στο δίκτυο μέσω του χρήστη 2 διασχίζουν όλο το κοινό μονοπάτι από τον μεταγωγέα 1 μέχρι το Access Point για να καταλήξουν να πεταχθούν στο ασύρματο κανάλι. Αυτό δυστυχώς έχει αντίκτυπο στην απόδοση που λαμβάνει ο χρήστης 1 γιατί χρησιμοποιεί κι αυτός το ίδιο μονοπάτι και ανταγωνίζεται με τον χρήστη 2. Καθώς λοιπόν αυξάνεται το data rate του χρήστη 2 βλέπουμε να μειώνεται το throughput του χρήστη 1. Συγκεκριμένα στην χειρότερη περίπτωση των 6Mbps physical rate βλέπουμε μια σταδιακή πτώση του throughput του ενσύρματου, δλδ του χρήστη 1 από 100Mbps σε 65Mbps, 35% πτώση της απόδοσης όσο αυξάνεται το φόρτο που εισάγει ο χρήστης 2.

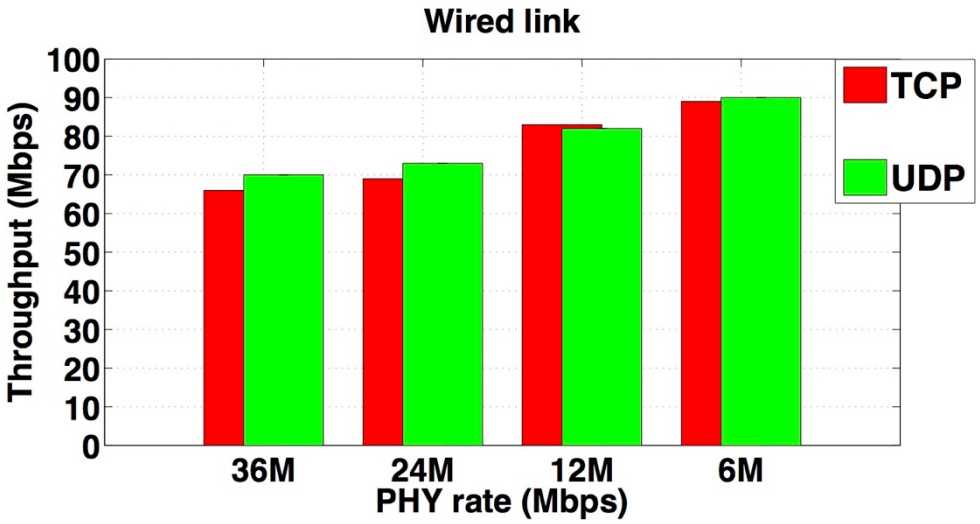
Επομένως είναι άσκοπο να εισάγουμε επιπλέον κίνηση στο δίκτυο (από 5 σε 50Mbps) όταν δεν βελτιώνεται η απόδοση του δικτύου καθώς τα πακέτα πετιούνται γιατί δεν μπορούν να υποστηριγχθούν από το link.

Το TCP λύνει αυτού του είδους τα πρόβλημα χρησιμοποιώντας τον μηχανισμό Ελέγχου Συμφόρησης ή Congestion Control που περιγράψαμε στην αρχή αυτού του κεφαλαίου. Εντοπίζει το πρόβλημα και προσαρμόζει ανάλογα τον ρυθμό των πακέτων που εισάγει μια ροή.

Στη συνέχεια συγκρίνουμε το TCP σε σχέση με την βέλτιστη απόδοση του UDP, δλδ δοκιμάζουμε όλα τα load στον χρήστη 2 με τα οποία πετυχαίνεται η καλύτερη απόδοση στο ασύρματο του χρήστη 1 και παίρνουμε τα παρακάτω διαγράμματα.



Γράφημα 3:Throughput στο ασύρματο(χρήστη2) για κάθε physical rate χρησιμοποιώντας tcp και udp



Γράφημα 4:Throughput στο ενσύρματο(χρήστη1) για κάθε physical rate χρησιμοποιώντας tcp και udp

Από τα διαγράμματα παρατηρούμε ότι το TCP είναι συντηρητικό στην προσαρμογή του φόρτου που εισάγει μια ροή, για αυτό και αποδίδει χαμηλότερα σε σχέση με το UDP (18). Το TCP εισάγει έως και 30% λιγότερο load από αυτό που μπορεί να υποστηριχθεί από το κανάλι, γιατί έτσι το εκτιμά με τον μηχανισμό flow και congestion control που ενεργοποιεί. Αναμενόμενο καθώς γνωρίζουμε ότι το TCP είναι ευάλωτο στις διακυμάνσεις του καναλιού.

Με αφορμή αυτό αλλά και το γεγονός πως η UDP κίνηση στο διαδίκτυο ολοένα και αυξάνεται, εμείς προσπαθήσαμε να ενεργοποιήσουμε έναν αυτόματο μηχανισμό flow control πάνω από το UDP.

6. Σχεδιασμός και Υλοποίηση Πρωτοκόλλου

6.1 Μηχανισμός Προσαρμογής φόρτου κίνησης UDP (Rate Adaptation Mechanism)

Η υλοποίηση βασίστηκε σε ένα μηχανισμό προσαρμογής του ρυθμού πακέτων που εισάγει κάθε ροή στο δίκτυο. Στη συνέχεια χρησιμοποιήσαμε αυτόν τον μηχανισμό για να σχεδιάσουμε τους αλγορίθμους που θα αναλύσουμε στη συνέχεια.

Με την τεχνολογία SDN και με το OpenFlow πρωτόκολλο καταφέραμε να διαχειριστούμε τη συμπεριφορά ενός δικτύου σχεδιάζοντας κατάλληλους **Controllers** (19) οι οποίοι

ελέγχουν την ροή εισαγωγής πακέτων στο δίκτυο ανά πάσα στιγμή εφόσον εντοπίσουν σημεία συμφόρησης. Προυπόθεση για αυτό το μηχανισμό είναι όλοι οι μεταγωγείς και οι δρομολογητές να επικοινωνούν με έναν ξεχωριστό controller.

Περιγραφή Μηχανισμού σε βήματα

1. Κάθε Switch/AP/Router κρατάει στατιστικά για όλα τα flows που εισέρχονται/εξέρχονται.

- Source/Destination Address
- Source/Destination IP
- Number of send packtes
- Number of send bytes

2. Υπολογίζει χρησιμοποιώντας τα στατιστικά, το throughput για κάθε flow του δικτύου το οποίο εισέρχεται/εξέρχεται στον controller.
3. Υπολογίζει σύμφωνα με τα στατιστικά το ρυθμό με τον οποίο εισέρχονται τα πακέτα μιας ροής (incoming) και τον ρυθμό με τον οποίο εξέρχονται τα πακέτα της ίδιας ροής (outgoing) .

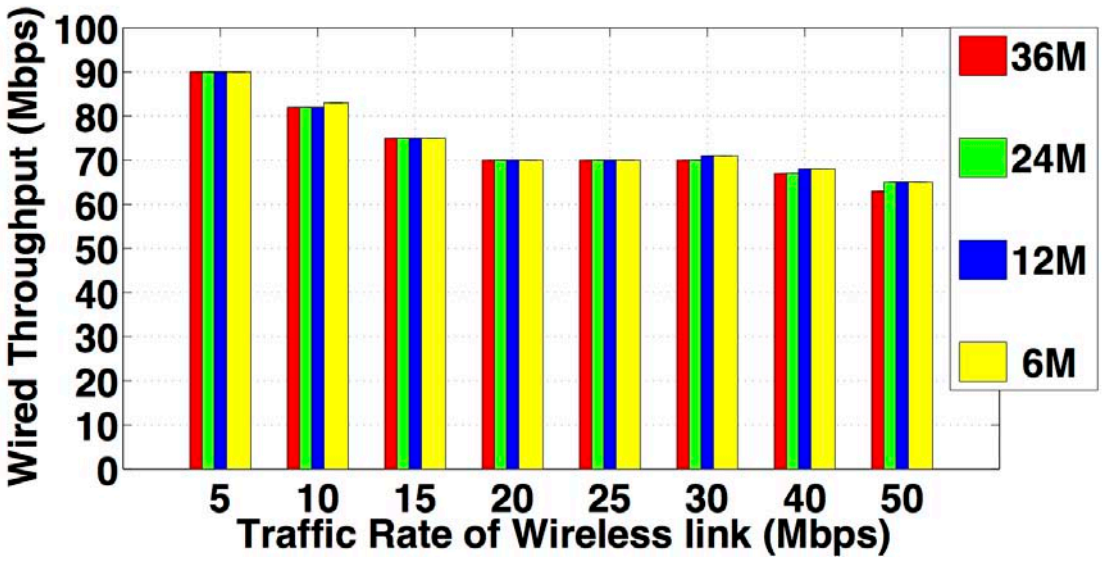
   If (outgoing<incoming-THR) at switch **X**

   - Ανιχνεύει το bottleneck
   - Ο controller του switch **X** ενημερώνει για το bottleneck τον controller του switch **Y**, το οποίο εισάγει την ροή που δημιουργεί το bottleneck.
   - Το switch **Y** προσαρμόζει το rate του flow που προκαλεί το bottleneck.

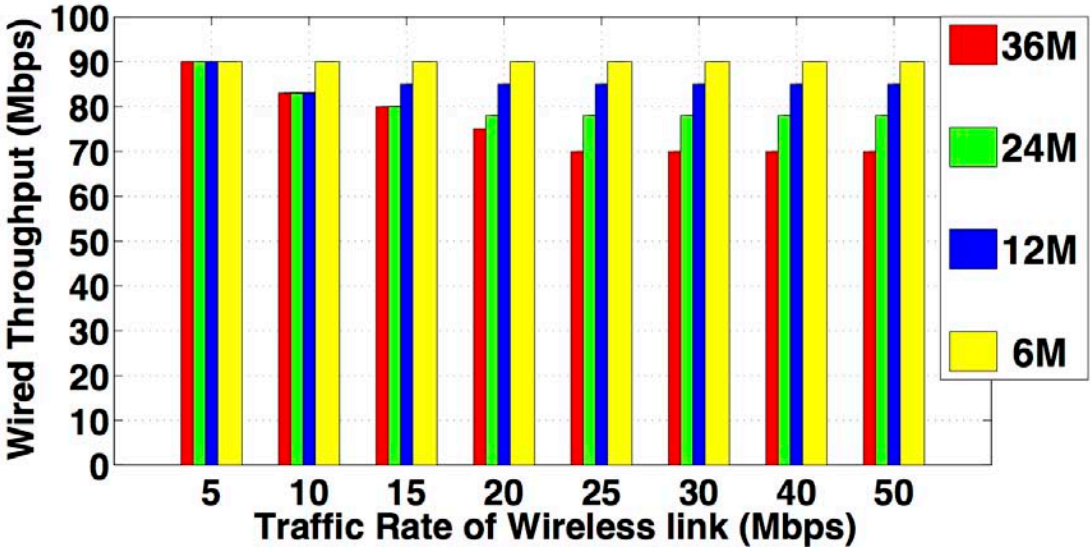
Για την προσαρμογή του rate της ροής που συμμετέχει ή προκαλεί το σημείο συμφόρησης εκμεταλλευτήκαμε τον μηχανισμό QoS που παρέχει το Openflow Πρωτόκολλο ώστε να διαμορφώσαμε τις ουρές μετάδοσης των διαφορετικών ροών που αντιστοιχούν στις πόρτες του εκάστοτε switch. Επομένως δημιουργήσαμε ουρές διαφορετικών προτεραιοτήτων για κάθε ροή ανάλογα με την εκτίμηση του λαμβανόμενου throughput για τη συγκεκριμένη ροή.

6.1.1 Αποτελέσματα μετά την ενεργοποίηση του Μηχανισμού

Στη συνέχεια συγκρίνουμε την απόδοση που πέτυχε το UDP στην τοπολογία του Σχήματος 6 στο Κεφάλαιο 5 χωρίς μηχανισμό flow control με την απόδοση που επιτεύγχθηκε μετά την ενεργοποίηση του μηχανισμού Adaptation Rate. Τα αποτελέσματα και στις δύο περιπτώσεις φαίνονται στα παρακάτω διαγράμματα.



Γράφημα5:Απόδοση του ενσύρματου στον χρήστη 1 πριν την ενεργοποίηση του μηχανισμού



Γράφημα6:Απόδοση του ενσύρματου στον χρήστη 1 μετά την ενεργοποίηση του μηχανισμού

Διαπιστώσαμε στο Κεφάλαιο 5 ότι η απόδοση του ενσύρματου κομματιού, συγκεκριμένα του χρήστη 1, μειώνεται καθώς αυξάνει το rate του ασύρματου flow, κάτι που φαίνεται και στο γράφημα 5 στη μείωση του ενσύρματου throughput.

Αντίθετα μετά την ενεργοποίηση του μηχανισμού προσαρμογής, βλέπουμε στο γράφημα 6 πως η απόδοση του ενσύρματου link ισούται με την μέγιστη που μπορεί να επιτευγχθεί καθώς παράλληλα λειτουργεί και το wireless flow.

Το κέρδος που έχουμε για κάθε physical rate αντίστοιχα είναι :

- 6 Mbps : 65 -> 90: **38.5%**
- 12 Mbps : 65 -> 85: **30.8%**
- 24 Mbps : 65 -> 78: **20%**
- 36 Mbps : 63 -> 70: **11%**

Παρακάτω χρησιμοποιούμε αυτόν τον μηχανισμό που περιγράψαμε σε αυτό το Κεφάλαιο για να υλοποιήσουμε διάφορες πολιτικές flow control για όλα τα flows του δικύτου και να αναπτύξουμε τους αλγορίθμους μας.

6.2 Minimum QoS Guarantee Algorithm

Για την ενεργοποίηση του έλεγχου ροής σε ένα δίκτυο χρησιμοποιήσαμε ως μητρική την απόδοση που λαμβάνει ο κάθε χρήστης στο δίκτυο, συγκεκριμένα χρησιμοποιήσαμε το throughput που λαμβάνει κάθε χρήστης. Έτσι υλοποιήσαμε έναν αλγόριθμο ο οποίος εγγυάται ένα ελάχιστο QoS για όλα τα flows του δικτύου και δίνει προτεραιότητα στα πιο μικρά flows ή καλύτερα στα πιο αδικημένα.

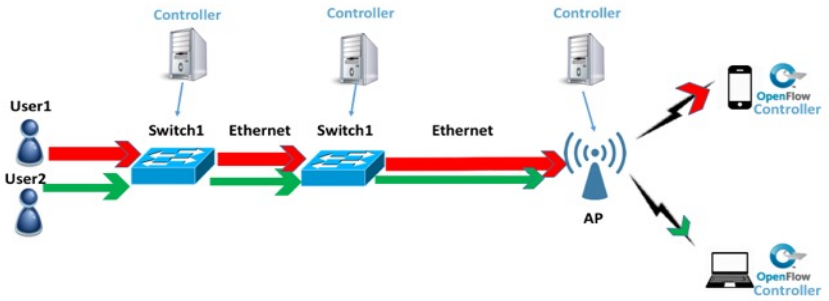
Ο μηχανισμός προσαρμογής από μόνος του δεν δίνει κάποιον έλεγχο στο που θα δοθεί το περισσότερο throughput στο δίκτυο ή ποιό flow θα ευνοηθεί τελικά περισσότερο. Όταν δεν υπάρχει κάποιος έλεγχος για αυτό, συνήθως οι ροές με τις μεγαλύτερες απαιτήσεις καταχράζονται όλο το δίκτυο και τους πόρους του με αποτέλεσμα να μην μένει διαθέσιμος χώρος και χρόνος να εξυπηρετηθού οι ροές με μικρότερες απαιτήσεις. Είναι όμως αυτό δίκαιο τελικά για όλο το δίκτυο;
Γενικά είναι δύσκολο να ορίσουμε την έννοια του δίκαιου, όπως στην πραγματική ζωή έτσι και στον κόσμο των δικτύων. Δίκαιο μπορεί να είναι να λαμβάνουν όλες οι ροές το ίδιο throughput ή την ίδια καθυστέρηση ή την ίδια ευχαρίστηση ή την ίδια μεταχείριση στο δίκτυο κ.τ.λ , ίσως αυτό είναι το πρώτο που μας έρχεται στο μυαλό όταν κάνουμε αυτή την ερώτηση στον εαυτό μας. Παρ'όλα αυτά όμως δίκαιο είναι και η χρήση του δικτύου να είναι ανάλογη με την χρήση ή την ανάγκη χρήσης του από το εκάστοτε flow/user ή ακόμα να γίνεται σύμφωνα με το ποσό που έχει πληρώσει ο χρήστης. Όλα αυτά είναι λογικά και για αυτό κάθε αλγόριθμος εστιάζει σε ένα από τα παραπάνω για να ελέγξει τις ροές στο δίκτυο.

Εμείς στον αλγόριθμο Minimum QoS επιλέξαμε να χρησιμοποιήσουμε το throughput / απόδοση που λαμβάνει κάθε χρήστης ως metric και να ορίσουμε ως δίκαιο τη δυνατότητα, το throughput όλων των χρηστών να μην πέφτει κάτω από ένα όριο / THRESHOLD_min που θέτει ο αλγόριθμος. Στη συνέχεια μπορούμε να δούμε τα βήματα που ακολουθεί αναλυτικά.

Περιγραφή Αλγορίθμου Minimum QoS σε βήματα

1. Ενεργοποιεί τον Rate Adaptation Mechanism για να εντοπίσει bottlenecks
2. Από όλα τα flows του δικτύου βρίσκει αυτό που έχει την μέγιστη απόδοση (Flow_max) και αυτό που έχει την ελάχιστη απόδοση (Flow_min)
3. While (Flow_min < THRESHOLD_min)
    a. Σταδιακά μειώνει το rate του Flow_max
    b. Υπολογίζει εκ νέου τα throughput όλων των flows
    c. If (|throughput(i) – throughput(j)| <= THRESHOLD_equal)
        i. Break;

Στο βήμα 3 του αλγορίθμου χρησιμοποιούμε ένα THRESHOLD_equal , συνήθως ίσο με το βήμα με το οποίο μειώνουμε σταδιακά το φόρτο που εισάγει ο χρήστης με την μεγαλύτερη απόδοση , για να αποφεύγουμε ατέρμων επαναλήψεις.

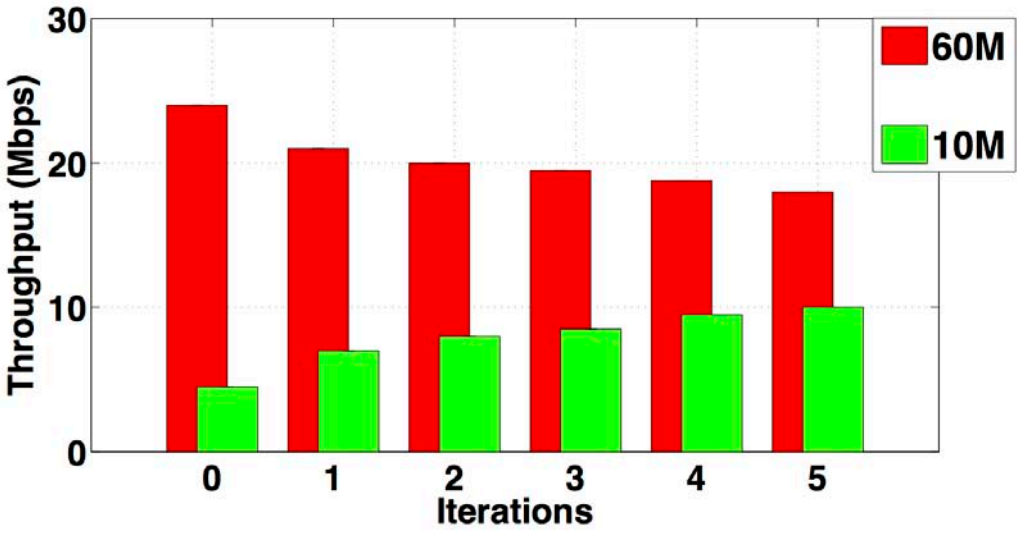


Σχήμα 7: Τοπολογία με 2 flows 60 kai 10 Mbps

Στο παραπάνω σχήμα παρατηρούμε ότι δυο κόμβοι λειτουργούν ως χρήστες και εισάγουν διαφορετικές ροές στο δίκτυο , οι οποίες διασχίζουν όλο το μονοπατι όπως φαίνεται από τα βελάκια και καταλήγουν μέσω ενός κόμβου – AP σε διαφορετικό device που χρησιμοποιεί το ασύρματο.

Κάθε συσκευή του δικτύου συνδέεται με έναν δικό της controller – ελεγκτή , ακόμα και τα device τρέχουν έναν controller για να έχουμε καλύτερη εκτίμηση για το throughput που λαμβάνει η συσκευή και να ενημερώνει τον controller του αντίστοιχου access point στο οποίο συνδέεται.

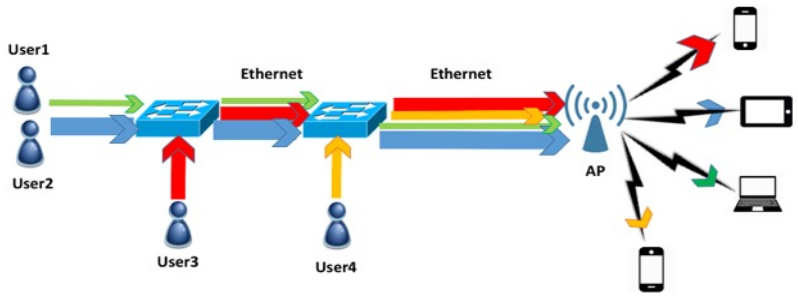
Στη συγκεκριμένη τοπολογία ο αλγόριθμος επιλέγει ως THRESHOLD_min τα 10 Mbps και ώς THRESHOLD_equal το 1 Mbps ενώ το bottleneck εμφανίζεται στο ασύρματο μέσο. Επομένως τρέχουμε τον αλγόριθμο και παίρνουμε τα αποτελέσμτα για την απόδοση του κάθε χρήστη.



Γράφημα 7:Απόδοση των 2 χρηστών

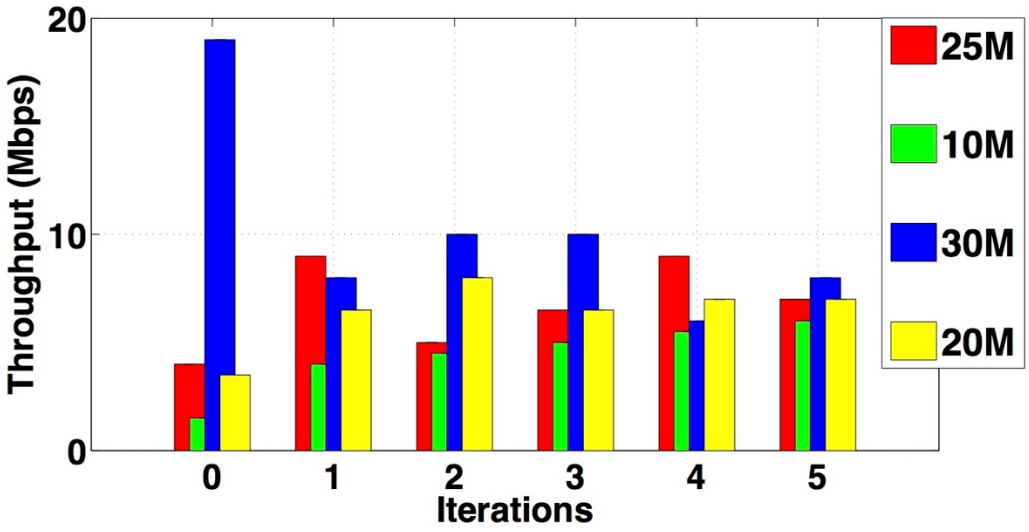
Στο γράφημα 7 βλέπουμε πως αρχικά ,πρίν τρέξουμε τον αλγόριθμο ο user 1 που εισάγει traffic 60 Mbps λαμβάνει throughput 25 Mbps και αφήνει ελεύθερα μόνο 3.5 Mbps για να χρησιμοποιήσει ο user 2. Συνολικά το ασύρματο κανάλι μπορεί να υποστηρίξει 28Mbps. Στην ουσία ο user 2 είναι αδικημένος από την απόδοση του δικτύου. Εφαρμόζωντας τον αλγόριθμο παρατηρούμε πως σταδιακά πλησιάζουν οι μπάρες του γραφήματος και καταλήγουν μετά απο πέντε επαναλήψεις να εγγυηθούν και στα 2 flows το ελάχιστο throughput των 10Mbps.

Στην ίδια τοπολογία πλέον εισάγουμε κι άλλους χρήστες οι οποίοι με τη σειρά του δημιουργούν νέες ροές με διαφορετική κίνηση η κάθε μία, οπότε πάμε στην παρακάτω τοπολογία του σχήματος 8.



Σχήμα 8: Τοπολογία με 4 flows 25, 10, 30 και 20 Mbps

Πάλι όλα τα switch, το AP και οι συσκευές συνδέονται με ξεχωριστούς controllers που ανταλλάσουν μηνύματα μεταξύ τους. Στη τοπολογία που βλέπουμε παραπάνω ο αλγόριθμος επιλέγει πάλι ως THRESHOLD_min τα 10 Mbps και ώς THRESHOLD_equal τα 2 Mbps ενώ το bottleneck εμφανίζεται όπως και πρίν στο ασύρματο μέσο. Και πάμε να δούμε τα αποτελέσματα που έδειξε ο αλγόριθμος στο παρακάτω γράφημα, στη νέα τοπολογία



Γράφημα 8:Απόδοση των 4 χρηστών

Πρίν την ενεργοποίηση του αλγορίθμου βλέπουμε πως το μεγαλύτερο κομμάτι του καναλιού το χρησιμοποιεί ο χρήστης 2 ο οποίος εισάγει την μπλέ κίνηση , όπως φαίνεται και στο σχήμα 8 αλλά και στο γράφημα 8, των 30 Mbps. Όλοι οι υπόλοιποι λαμβάνουν κάτι ανάμεσα απο το 1.5 και το 4 Mbps.

Καθώς λειτουργεί ο αλγόριθμος βλέπουμε να μειώνεται σταδιακά το throughput του χρήστη 2 και να συγκλίνουν όλες οι τιμές των throughput κοντά στο 6, και άρα να διαφέρουν μεταξύ τους ανά 2 Mbps όσο είναι δλδ το THRESHOLD_equal. Οπότε μετά τις πέντε επαναλήψεις ο αλγόριθμος τερματίζει χωρίς να έχει εγγυηθεί τα 10 Mbps σε όλα τα flows, έχει πετύχει όμως το καλύτερο δυνατό που μπορεί.
Συγκεκριμένα εγγυάται ό,τι πιο κοντά στο THRESHOLD_min μπορεί να επιτευγθεί.

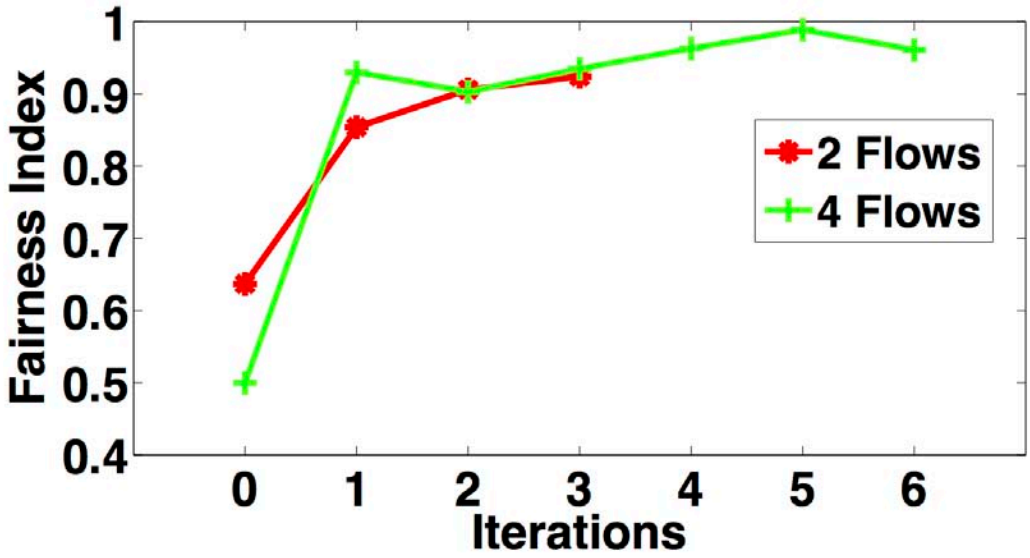
6.2.2 Jain's Fairness Index

Για να αξιολογήσουμε το σύστημά μας χρησιμοποιήσαμε ένα δείκτη ο οποίος δείχνει κατά πόσο οι πόροι του δικτύου μοιράζονται δίκαια ή όχι.

Συγκεκριμένα επιλέξαμε το Jain's Fairness Index (22) που καθορίζει κατά πόσο οι χρήστες ή οι εφαρμογές σε ένα δίκτυο λαμβάνουν ένα δίκαιο κομμάτι των πόρων αυτού του δικτύου, στην προκειμένη περίπτωση κατά πόσο το throughput που λαμβάνουν είναι το ίδιο. Ο τύπος του Jain's ορίζεται ως εξής :

$$\mathcal{J}(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} {x_i}^2}$$

Όπου για κάθε x_i θέτουμε το throughput του i χρήστη και n είναι ο συνολικός αριθμός των flows στο δίκτυο. Όπως διαπιστώνει κανείς εύκολα και από τον τύπο οι τιμές που λαμβάνει η συνάρτηση κυμαίνονται μεταξύ $\frac{1}{n}$ στην χειρότερη περίπτωση όπου όλα τα throughputs διαφέρουν μεταξύ τους και 1 στην καλύτερη περίπτωση όπου είναι ίδια.



Γράφημα 9:Jain's Index

Όπως παρατηρούμε καθώς αυξένεται ο αριθμός των επαναλήψεων η γραφική παράσταση, και στις 2 περιπτώσεις τοπολογιών, τείνει στο ένα, κάτι που περιμέναμε εφόσον ο αλγόριθμος οδηγεί σε σύγκλιση των throughputs των ροών.

6.3 Fairness Algorithm

Βασιζόμενοι στα αποτελέσματα του προηγούμενου αλγορίθμου συμπεραίνουμε πως είναι αρκετά δύσκολο ο αλγόριθμος να βρεί και να εγγυηθεί ένα κατώτατο όριο για το throughput ώστε να ικανοποιήσει όλα τα flows του δικτύου.

Επανερχόμενοι έτσι στο βασικό ερώτημα του τι είναι τελικά δίκαιο και αποδοτικό, σκεφτήκαμε πως υπάρχουν περιπτώσεις στις οποίες δε θέλουμε να λαμβάνουν όλοι η χρήστες το ίδιο throughput, για διάφορους λόγους (μερικούς αναφέραμε στην αρχή του κεφαλαίου) . Έτσι εστιάσαμε στην έννοια του «Proportional Fairness» (20) , δλδ στον ορισμό της δικαιοσύνης βασιζόμενη στην ικανοποίηση που λαμβάνει ένας χρήστης από την χρήση του μέσου. Για να συσχετίσουμε την έννοια της ικανοποίησης με ένα χρήστη χρησιμοποιήσαμε μια «utility function» (21) , η οποία μπορεί να θεωρηθεί ως ένας δείκτης ικανοποίησης του χρήστη όταν λαμβάνει ένα συγκεκριμένο ρυθμό πακέτων από το δίκτυο. Μια δεύτερη έννοια της συνάρτησης είναι πώς ανατίθεται σε ένα χρήστη με σκοπό να κατορθώσει να χρησιμοποιήσει ένα συγκεκριμένο ποσοστό του δικτύου. Τελικά με οποιαδήποτε έννοια κι αν χρησιμοποιηθεί , στην ουσία αυτό που κάνουμε

χρησιμοποιώντας αυτές τις συναρτήσεις είναι να μεγιστοποιούμε το «network utility» ή αλλιώς την ευχαρίστηση όλοκληρου του δικτύου.

Στον αλγόριθμο που θα δούμε στη συνέχεια επιλέξαμε ως «utility function» την συνάρτηση :

$$U(x_i) = w_i \log_{10}(x_i)$$

Όπου δείχνει την ικανοποίηση του χρήστη i όταν λαμβάνει throughput x . Ενώ το wi είναι ένα βάρος που αναθέτουμε σε κάθε χρήστη ώστε να πάρουμε το «weighted proportional fairness». Επί της ουσίας αυτά τα βάρη δηλώνουν το πόσο γρήγορα ή αργά ικανοποιείται ο χρήστης. Όπως όλες οι utility functions έτσι και η δική μας είναι εξ'ορισμού αύξουσα και καμπύλη – κοίλη.

Περνάμε τώρα στην υλοποίηση αυτού του αλγορίθμου ο οποίος προσπαθεί να μεγιστοποιήσει την συνολική ικανοποίηση όλων των χρηστών λαμβάνοντας υπόψιν τους περιορισμούς της χωρητικότητας των links στο δίκτυο.

Περιγραφή Αλγορίθμου Fairness σε βήματα

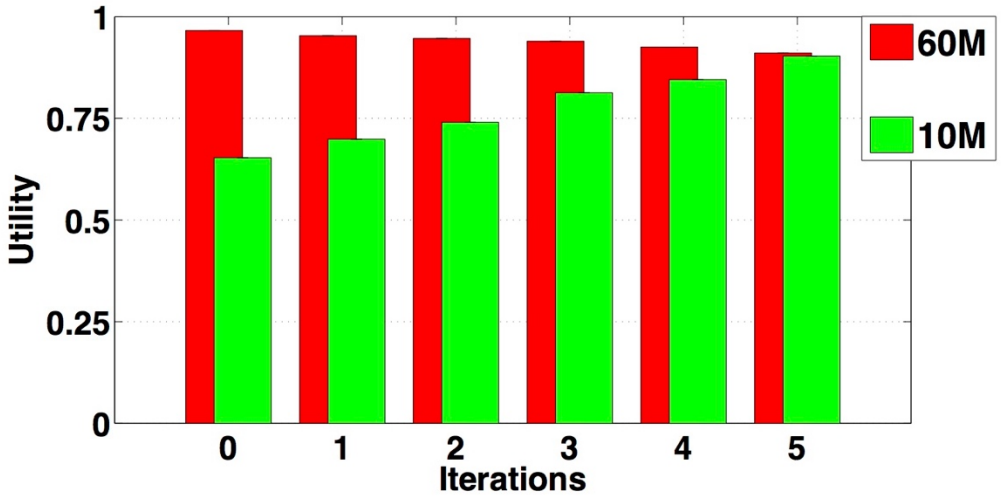
1. Ενεργοποιεί τον μηχανισμό εντοπισμού bottleneck
2. Για όλα τα flows εντοπίζει εκείνο με το μεγαλύτερο και το μικρότερο utility

   (Umax , Umin)
3. While (Umin < Umax)
   a. Σταδιακά μειώνει το rate του flow με το μεγαλύτερο utility
   b. Υπολογίζει εκ νέου τα utilities όλων των flows

6.3.1 Αποτελέσματα αλγορίθμου σε πραγματικές τοπολογίες

Χρησιμοποιώντας τις ίδιες τοπολογίες των σχημάτων 7 και 8 τρέξαμε τον νέο αλγόριθμο πλέον και συλλέξαμε τα αποτελέσματα που πέτυχε.

Για την τοπολογία του σχήματος 7 με τα δύο flows των δύο χρηστών έχουμε το νέο διάγραμμα :



Γράφημα 10:Utility των 2 χρηστών

Τα βάρη που αναθέσαμε στους δύο χρήστες είναι :

- w1 = 0.7 για τον χρήστη 1 (κόκκινη μπάρα)
- w2 = 1 για τον χρήστη 2 (πράσινη μπάρα)

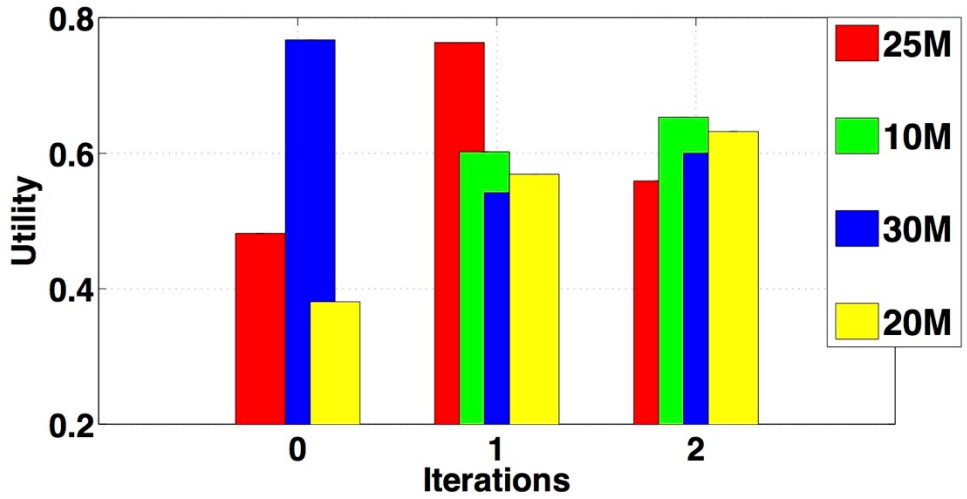
Από αυτό φαίνεται πώς πιο εύκολα ικανοποιείται ο χρήστης 2 από το throughput που λαμβάνει, σε σχέση με τον χρήστη 1. Μετά από πέντε επαναλήψεις ο αλγόριθμος τερμάτισε δίνοντας και στους δύο χρήστες την ίδια ικανοποίηση. Παρακάτω φαίνονται και τα throughput για τα οποία έχουμε τις συγκεκριμένες τιμές του utility.

60Mbps	log(24) = 1,3	log(23) = 1,36	log(22,5) = 1.35	log(22) = 1,34	log(21) =1,32	log(20)=1,3
f(u1)	**0.91**	**0.95**	**0.94**	**0.93**	**0.92**	**0.9**
10Mbps	log(4,5) = 0,65	log(5) = 0,69	log(5,5) = 0,74	log(6,5) = 0,81	log(7)=0.84	log(8) = 0,9
f(u2)	**0.65**	**0.69**	**0.74**	**0.81**	**0.84**	**0.9**

Πίνακας 1: Αποτελέσματα throughput και utility για 2 flows

Όπως βλέπουμε στον πίνακα στην τελευταία επανάληψη ο χρήστης 1 που εισάγει την κίνηση των 60 Mbps λαμβάνει τελικά throughput 20Mbps, ενώ ο χρήστης 2 που εισάγει την κίνηση των 10Mbps λαμβάνει 8Mbps. Αν ανατρέξουμε παραπάνω στα αποτελέσματα του Minimum QoS για τα ίδια flows θα δούμε πως το throughput που λάμβαναν ήταν 18 και 10 Mbps αντίστοιχα.

Για την τοπολογια του σχήματος 8 με τα τέσσερα διαφορετικά flows το νέο διάγραμμα είναι:



Γράφημα 11:Utility των 4 χρηστών

Για τα τέσσερα flows τα βάρη που χρησιμοποιήσαμε είναι:

- w1 = 0.8 για τα 25Mbps
- w2 = 1 για τα 10 Mbps
- w3 = 0.6 για 30 Mbps
- w4 = 0.7 για 20Mbps

Μόλις μετά από δύο επαναλήψεις τα utilities των flows παίρνουν σχεδόν ίδια τιμή χρησιμοποιώντας τα συγκεκριμένα βάρη για του χρήστες.

25Mbps	4	9	5

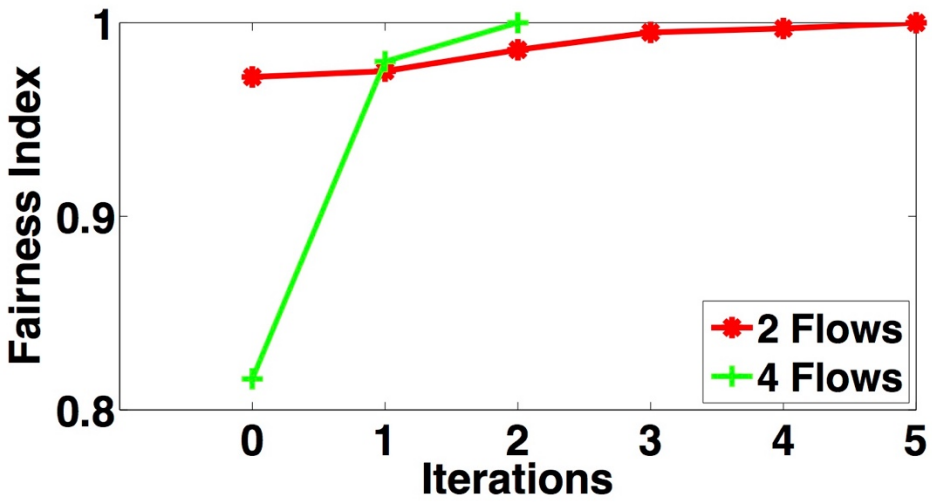
f(u1)	**0.48**	**0.76**	**0.6**
10Mbps	1,5	4	4,5
f(u2)	**0.17**	**0.6**	**0.6**
30Mbps	19	8	10
f(u3)	**0.76**	**0.54**	**0.6**
20Mbps	3,5	6,5	8
f(u4)	**0.38**	**0.56**	**0.6**
Jain	0.816	0.98	1
iteration	0	1	2

Πίνακας 2: Αποτελέσματα throughput και utility για 4 flows

Όπως βλέπουμε στον πίνακα τα τελικά throughputs είναι 5 , 4.5 , 10 και 8 αντίστοιχα για κάθε flow. Κι αν τα συγκρίνουμε με τα throughputs του προηγούμενο αλγόριθμο που λάμβαναν όλοι κάτι κοντά στο 6 στη συγκεκριμένη περίπτωση βλέπουμε μια πίο δίκαιη λύση σε συνάρτηση με το τι ζητάει ο χρήστης αρχικά.

6.3.2 Jain's Fairness Index

Βασιζόμενοι πάλι στον ίδιο δείκτη δίκαιης κατανομής πόρων δικτύου, χρησιμοποιούμε τον Jain's index για αξιολόγηση του αλγορίθμου. Ο τύπος είναι ο ίδιος με πρίν μόνο που τώρα βάζουμε στη θέση του xi το utility του εκάστοτε flow.



Γράφημα 12:Jain's Index

Η ίδια συμπεριφορά με πρίν, έτσι κι εδώ καθώς αυξάνεται ο αριθμός των επαναλήψεων του αλγορίθμου η γραφική παράσταση αγγίζει το 1 πλέον.

7. Κατευθύνσεις για συνέχιση της δουλειάς μας

Ολοκληρώνοντας θα παρουσιάσουμε κάποιες κατευθύνσεις για τη επέκταση της δουλειάς μας πάνω στον τομέα των Δικτύων και των Software Defined Networks.

Όπως παρατηρήσατε στο κεφάλαιο 6 της υλοποίησης, όλα τα πειράματα και οι μετρήσεις έγιναν χρησιμοποιώντας UDP κίνηση , τώρα θα θέλαμε να τρέξουμε στις ίδιες τοπολογίες με τα ίδια νούμερα, τους αλγορίθμους αλλά χρησιμοποιώντας TCP κίνηση αυτή τη φορά. Έτσι θα δούμε αν υπάρχουν διαφορές και αν πετυχαίνει κάτι καλύτερο το TCP πρωτόκολλο.

Η υλοποίηση του flow control σε ένα δίκτυο μπορεί να γίνει με πολλούς και διαφορετικούς τρόπους σύμφωνα με τον τρόπο που θέλουμε να κατευθύνουμε τα flows και τον ορισμό που ερμηνεύουμε την έννοια του «fairness».Εμείς εστιάσαμε μόνο σε δύο κριτήρια, το throughput και το utility, υπάρχουν όμως άπειρα metric που θα μπορούσαμε να βασιστούμε. Ενδεικτικά αναφέρουμε τα εξής:

- Άθροισμα των throughputs
- Άθροισμα των utilities
- Καθυστέρηση
- Ελαχιστοποίηση διακυμάνσεων του throughput ή της καθυστέρησης
- Ανθεκτικότητα σε εναλλασόμμεο περιβάλλον
- Rerouting schemes

Τέλος για να λάβουμε υπόψιν και τις περιπτώσεις όπου κάποιοι χρήστες χρειάζεται να λαμβάνουν μεγαλύτερο ποσοστό των πόρων του δίκτυου, επειδή για παράδειγμα έχουν

πληρώσει για να έχουν αυτό το δικαίωμα, θέλουμε να υλοποιήσουμε έναν αλγόριθμο όπου θα εισάγει προτεραιότητες στους χρήστες. Στη ουσία θα υπάρχει μία ιεράρχιση των ροών σε όλο το δίκτυο.

8. Αναφορές

1. https://en.wikipedia.org/wiki/Heterogeneous_network
2. https://en.wikipedia.org/wiki/Forwarding_plane
3. http://searchsdn.techtarget.com/definition/Distributed-Control-Plane-Architecture-DCPA
4. http://searchsdn.techtarget.com/tip/SDN-basics-Understanding-centralized-control-and-programmability
5. http://blog.ipspace.net/2013/08/management-control-and-data-planes-in.html
6. http://archive.openflow.org/wp/learnmore/
7. http://whatis.techtarget.com/definition/OpenFlow
8. https://www.opennetworking.org/sdn-resources/sdn-definition
9. http://searchsdn.techtarget.com/definition/OpenFlow-switch
10. https://en.wikipedia.org/wiki/Testbed
11. http://nitlab.inf.uth.gr/NITlab/index.php/component/users/?view=login
12. http://openvswitch.org/
13. https://el.wikipedia.org/wiki/Transmission_Control_Protocol
14. http://www.princeton.edu/~chiangm/ele539l6.pdf
15. https://el.wikipedia.org/wiki/UDP

16. https://en.wikipedia.org/wiki/User_Datagram_Protocol
17. http://www.cacs.louisiana.edu/~perkins/csce575/papers/80211_tutorial-veriwave.pdf
18. https://books.google.gr/books?id=jKltCQAAQBAJ&pg=PA179&lpg=PA179&dq=tcp+conservative+protocol&source=bl&ots=-lmqeQQMau&sig=HiFIPFNkn5w9Ikp1ndAr2T8pd08&hl=el&sa=X&ved=0CHAQ6AEwCWoVChMIhL_DxZ_DyAIVR7YaCh29kgaX#v=onepage&q=tcp%20conservative%20protocol&f=false
19. http://searchsdn.techtarget.com/definition/OpenFlow-controller
20. http://www.cs.helsinki.fi/u/ldaniel/mm_cn/lec1.2_cc_resource_allocation.pdf
21. http://www.ifp.illinois.edu/~srikant/ECE567/Fall09/lecture2-num-primal.pdf
22. https://en.wikipedia.org/wiki/Fairness_measure