

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΣΧΟΛΗ ΕΠΙΣΤΗΜΗΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
Τμήμα Πληροφορικής



Πτυχιακή Εργασία

Μελέτη Λειτουργίας SDN

Δικτυακών Περιβάλλοντων

Γεώργιος Σιδεράς Π2017164

Επιβλέπων: Κωνσταντίνος Οικονόμου

16 Ιουνίου 2021

Επιβλέπων

Κωνσταντίνος Οικονόμου, Καθηγητής,
Ιόνιο Πανεπιστήμιο

Τριμελής Επιτροπή

Κωνσταντίνος Οικονόμου, Καθηγητής,
Ιόνιο Πανεπιστήμιο

Θεόδωρος Ανδρόνικος, Αναπληρωτής Καθηγητής,
Ιόνιο Πανεπιστήμιο

Μιχαήλ Στεφανιδάκης, Αναπληρωτής Καθηγητής,
Ιόνιο Πανεπιστήμιο

Αυτή η προσπάθεια αφιερώνεται στους καθηγητές και συμφοιτητές του Τμήματος Πληροφορικής του Ιονίου Πανεπιστημίου που με βοήθησαν αλλά και στους γονείς μου, που μου παρείχαν όλα τα μέσα για να πετύχω τον σκοπό μου.

Περίληψη

Η χρήση του όρου Software-Defined Networking (SDN) είναι πλέον μια καθοριστική τεχνολογία για το μέλλον των δικτύων επικοινωνίας. Πιο συγκεκριμένα, αποτελεί μια διαφορετική προσέγγιση λειτουργίας δικτύων από αυτήν που χρησιμοποιούν τα παραδοσιακά. Στόχος της είναι να επιτρέψει στους διαχειριστές των δικτύων να έχουν άμεση πρόσβαση και περισσότερες ενέργειες σχετικά με τον πίνακα ροής των πακέτων του δικτύου και συνεπώς, με τη δρομολόγηση τους. Η χρήση της SDN τεχνολογίας στα σημερινά δίκτυα αυξάνεται λόγω της αυξημένης πολυπλοκότητας και της υποχρησιμοποίησης των δικτυακών πόρων. Μερικές από τις χαρακτηριστικές εφαρμογές του SDN είναι σε αλλαγές των μοτίβων δρομολόγησης σε δίκτυα διομότιμων υπολογιστών (p2p), στις υπηρεσίες νέφους αλλά και σε δίκτυα με μεγάλη χρήση σε μικρο-συσσκευές ή IoT.

Για να πραγματοποιηθεί η εφαρμογή του SDN γίνεται διαχωρισμός του φυσικού επιπέδου μεταφοράς δεδομένων με το επίπεδο λήψης αποφάσεων. Πιο συγκεκριμένα, γίνεται τοποθέτηση ενός ή περισσότερων κεντρικών οντοτήτων οι οποίες, έχουν την πλήρη εικόνα του δικτύου, και παίρνουν τις αποφάσεις για τη δρομολόγηση. Αυτό γίνεται μέσω της τροποποίησης του πίνακα-ροής που έχουν οι συσκευές αναμετάδοσης των πακέτων στο δίκτυο, όπως οι μεταγωγείς. Συμπεραίνοντας, πρόκειται για έναν μηχανισμό που επιτρέπει τη λήψη αποφάσεων για τα κανάλια επικοινωνίας του δικτύου με τη χρήση του λογισμικού και όχι του υλικού.

Παρ' όλα αυτά, το SDN είναι η λογική που χρησιμοποιείται στα δίκτυα για να γίνει ο διαχωρισμός των δυο καναλιών. Για να εφαρμοστεί αυτό στην πράξη χρειάζεται ένα σύνολο από κανόνες και διαδικασίες ή αλλιώς ένα πρωτόκολλο επικοινωνίας, το οποίο θα ακολουθούν οι συσκευές του δικτύου. Επομένως, ένα από αυτά τα πρωτόκολλα είναι το OpenFlow. Διευκρινίζοντας, έχει τη δυνατότητα να εφαρμοστεί στους περισσότερους κλασσικούς μεταγωγείς (switches) της αγοράς αλλά και να δημιουργηθεί εξ αρχής σε έναν. Επιπροσθέτως, η κεντρική οντότητα που πρέπει να επικοινωνούν όλοι οι μεταγωγείς ονομάζεται ελεγκτής (Controller) και υπάρχουν αρκετά έτοιμα λογισμικά που τρέχουν συγκεκριμένες εφαρμογές ή κάποια α-

νοικτού λογισμικού που επιτρέπουν στον διαχειριστή του δικτύου να τρέξει τις δικές του εφαρμογές. Το συγκεκριμένο πρωτόκολλο διαχειρίζεται και επεκτείνεται από τον οργανισμό Open Networking Foundation(ONF) για να καλύπτει όλο και περισσότερες ανάγκες.

Σε αυτήν την εργασία προτείνεται η αντιμετώπιση ενός προβλήματος ισορρόπησης του φόρτου σε ένα δίκτυο. Για αυτό τον σκοπό, προσομοιώνονται δύο εικονικά δίκτυα με τη χρήση του προγράμματος OMNeT++ και την εγκατάσταση επιπρόσθετων πακέτων όπως του INET και του OpenFlow. Το ένα δίκτυο κάνει χρήση του πρωτοκόλλου επικοινωνίας OpenFlow, ενώ το άλλο είναι ένα παραδοσιακό δίκτυο. Για να δημιουργηθεί μεγάλη κίνηση στο δίκτυο, γίνεται η εισαγωγή πολλών χρηστών, και έπειτα γίνονται πειραματισμοί για διάφορες τιμές όπως το data rate, η καθυστέρηση της διάδοσης της γραμμής και το μέγεθος των αρχείων που αποστέλλονται. Στη συνέχεια, αναλύονται τα προτερήματα της αναβάθμισης ενός δικτύου σε SDN και γιατί να γίνει αυτή η αλλαγή. Τέλος, παρουσιάζονται τα αποτελέσματα και τα διαγράμματα των πειραμάτων, καθώς επίσης η σύγκριση της απόδοσης αυτών των δύο δικτύων.

Λέξεις - Κλειδιά (SDN, OpenFlow, OMNeT++, Load Balancing)

Abstract

The usage of term Software-Defined Networking (SDN), is now a well-defined technology for the future of communication networks. More specifically, it is considered a different approach in the use of network rather than the traditional one. It aims to the allowance of the network administrators to take control of the actions regarding the flows of the packets and therefore, the routing of the network. The use of the SDN technology in today's networks, is growing increasingly, because of the complexity and the under-utilization of the network's resources. Some notably applications of SDN are the change of pattern in routing of peer-to-peer networks, cloud services or in networks based on Internet of Things (IoT).

In order for the SDN to be applied in a network, the separation of the data plane and the control plane must take place. For instance, one or more central entities are placed in the network that can have the full picture of the network and can take decisions about the routing of each packet through the flow-entries of the switches. In conclusion, it is about a mechanism that can control the flow of all channels in the network with the use of added software and not the already existing hardware.

Nonetheless, SDN is the idea that is been used in networks for the separation of the control and data plane. In order for this to work in real life, some rules and actions must be applied in the network, in other words, a communication protocol must be set. Hence, one of these protocols is OpenFlow. Subsequently, it can be installed in almost all normal switches that can be found in the market or to get one that has it already build in. Also, the central entity is called a Controller and can be acquired with ready software for specific operations or to get one with open source so that networks administrators can use their own code. This specific protocol is being managed and upgraded from the Open Networking Foundation (ONF) in order to satisfy the needs of more people every day.

In this study, the suggestion of how to solve a load balancing problem in a network is

been presented. For this, two virtual networks are being simulated using the OMNeT++ program with the addition of INET and OpenFlow packages. One network is using the OpenFlow protocol with the extra placement of a controller and the other is a traditional network. For the purpose of big load in the routes of the network, many users are been placed as well as values as data rate, link delay and file size are been changed for different experiments. Moreover, the advantage of using and SDN are being analyzed, and why the upgrade to SDN is essential. In conclusion, all the results and charts are presented, as well as the comparison of efficiencies of each network.

Key Words (SDN, OpenFlow, OMNeT++, Load Balancing)

Πρόλογος και Ευχαριστίες

Η παρούσα πτυχιακή εργασία εκπονήθηκε κατά την περίοδο του έτους 2020 - 2021 στα πλαίσια της ολοκλήρωσης των σπουδών της σχολής. Η εργασία πραγματοποιήθηκε υπό την επίβλεψη του κ. Κωνσταντίνου Οικονόμου, Καθηγητή του τμήματος 'Πληροφορικής' του Ιονίου Πανεπιστημίου. Αντικείμενο της εργασίας αποτελεί η μελέτη των SDN δικτυακών περιβάλλοντων και η προσομοίωση τους με το πρόγραμμα προσομοίωσης δικτύων OMNeT++.

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω όλους όσους συνέβαλαν στην εκπόνηση της πτυχιακής μου εργασίας. Οφείλω να εκφράσω τις θερμές μου ευχαριστίες, προς τον επιβλέποντα της εργασίας, Καθηγητή κ. Κωνσταντίνο Οικονόμου, για την καθοδήγησή του, και την πολύτιμη βοήθεια που προσέφερε σε κάθε στάδιο εκπόνησης της διατριβής μου. Επίσης, τον κ. Παπαμιχαήλ Αστέριο, υποψήφιος μεταπτυχιακού διπλώματος του Ιονίου Πανεπιστημίου, για την καθοδήγησή του σχετικά με το τεχνικό μέρος της εργασίας.

Επίσης, θα ήθελα αν ευχαριστήσω τους συναδέλφους μου, Σοφρονία Φουτσιτζή και Νέμανια Γιέβτιτς για την απαραίτητη βοήθεια, όπου αυτή χρειάστηκε. Τέλος, ευχαριστώ θερμά την οικογένειά μου, την κοπέλα μου και τους φίλους μου για τη συμπαράσταση που μου έδειξαν κατά τη διάρκεια εκπόνησης αυτής της πτυχιακής εργασίας.

Περιεχόμενα

A´	Εισαγωγικά Στοιχεία	1
A´.1	ΕΙΣΑΓΩΓΗ	1
A´.2	ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΕΡΕΥΝΑ	2
A´.3	ΑΠΟΤΕΛΕΣΜΑΤΑ ΚΑΙ ΣΥΝΕΙΣΦΟΡΑ	3
A´.4	ΔΟΜΗ ΕΡΓΑΣΙΑΣ	3
B´	Περιγραφή της SDN Λειτουργίας	5
B´.1	ΟΡΙΣΜΟΣ	5
B´.2	ΙΣΤΟΡΙΑ ΤΟΥ SDN	6
B´.3	ΧΡΗΣΙΜΟΤΗΤΑ ΚΑΙ ΕΦΑΡΜΟΓΕΣ	7
B´.3.1	Χρησιμότητα	7
B´.3.2	Εφαρμογές	9
B´.4	ΔΟΜΗ ΤΟΥ SDN	11
Γ´	Πρωτόκολλο Επικοινωνίας OpenFlow	14
Γ´.1	ΕΙΣΑΓΩΓΗ	14
Γ´.2	ΔΗΜΙΟΥΡΓΙΑ ΚΑΙ ΠΟΡΕΙΑ ΑΝΑΠΤΥΞΗΣ	15
Γ´.3	OPENFLOW ΜΕΤΑΓΩΓΕΙΣ	17
Γ´.3.1	Γενικά	17
Γ´.3.2	Αφιερωμένοι OpenFlow Μεταγωγείς	18
Γ´.3.3	Ενεργοποιημένοι OpenFlow Μεταγωγείς	19
Γ´.4	ΠΙΝΑΚΑΣ ΡΟΗΣ	20
Γ´.5	ΣΥΝΔΕΣΗ ΚΑΙ ΜΗΧΑΝΙΣΜΟΣ ΠΡΟΩΘΗΣΗΣ ΜΗΝΥΜΑΤΩΝ	20
Γ´.5.1	Αποστολή Πακέτων μέσω TCP/IP	20

Γ'.5.2	Αποστολή Πακέτων με OpenFlow	22
Γ'.6	OPENFLOW CONTROLLERS	23
Γ'.7	ΠΡΟΔΙΑΓΡΑΦΕΣ ΕΚΔΟΣΕΩΝ	25
Γ'.7.1	OpenFlow 1.1	25
Γ'.7.2	OpenFlow 1.2	25
Γ'.7.3	OpenFlow 1.3	26
Γ'.7.4	OpenFlow 1.4	26
Γ'.7.5	OpenFlow 1.5	26
Δ'	Πειραματισμοί με OMNeT++	28
Δ'.1	ΚΥΡΙΑ ΙΔΕΑ	28
Δ'.2	ΠΕΡΙΒΑΛΛΟΝ ΠΡΟΣΟΜΙΩΣΗΣ	29
Δ'.2.1	OMNeT++	29
Δ'.2.2	INET	29
Δ'.2.3	OpenFlow	30
Δ'.2.4	Εγκατάσταση	30
Δ'.3	ΤΟΠΟΛΟΓΙΑ ΚΑΙ ΜΕΤΑΒΛΗΤΕΣ	31
Δ'.3.1	Τοπολογία	31
Δ'.3.2	Μεταβλητές	31
Δ'.4	ΠΑΡΑΤΗΡΗΣΕΙΣ / ΠΡΟΒΛΗΜΑΤΑ ΠΡΟΣΟΜΙΩΣΕΩΝ	32
Ε'	Αποτελέσματα Προσομοιώσεων	34
Ε'.1	ΠΕΙΡΑΜΑΤΑ / ΠΡΟΣΟΜΙΩΣΕΙΣ	34
Ε'.1.1	Πρώτο Πείραμα (σταθερή η χρονική καθυστέρηση της διάδοσης και το Data Rate)	34
Ε'.1.2	Δεύτερο Πείραμα (Σταθερός ο αριθμός των χρηστών και του Data Rate)	34
Ε'.1.3	Τρίτο Πείραμα (Σταθερός ο αριθμός των χρηστών και της χρονικής καθυστέρησης της διάδοσης)	35
Ε'.1.4	Τέταρτο Πείραμα (Διαφορά της απόδοσης σε σύγκριση με το μέγεθος του αρχείου)	35
Ε'.1.5	Πέμπτο Πείραμα (Απόδοση της εφαρμογής LLDPBalancedHop)	35
Ε'.2	ΑΠΟΤΕΛΕΣΜΑΤΑ	35

Ε'.2.1	Συλλογή αποτελεσμάτων	35
Ε'.2.2	Αποτελέσματα πρώτου πειράματος	36
Ε'.2.3	Αποτελέσματα δεύτερου πειράματος	36
Ε'.2.4	Αποτελέσματα τρίτου πειράματος	37
Ε'.2.5	Αποτελέσματα τέταρτου πειράματος	38
Ε'.2.6	Αποτελέσματα πέμπτου πειράματος	39
Ε'.2.7	Σχολιασμός των αποτελεσμάτων	39
Γ'	Συμπεράσματα	44
Γ'.1	ΣΤΥΝΟΨΗ	44
Γ'.2	ΣΤΥΜΠΕΡΑΣΜΑΤΑ	45
Γ'.3	ΣΤΥΝΕΙΣΦΟΡΑ	46
	Βιβλιογραφία	46
	Παράρτημα	54
	Συντμήσεις	68
	Γλωσσάρι Ξενικών Όρων	70

Κατάλογος Σχημάτων

B'.1	Παραδοσιακό Δίκτυο vs SDN.	6
B'.2	Δίκτυο πελάτη-εξυπηρετητή με δίκτυο peer-to-peer.	8
B'.3	RAN και Core Δίκτυα.	10
B'.4	Δομή SDN.	12
Γ'.5	Απλή επίδειξη ενός OpenFlow δικτύου.	15
Γ'.6	Χρονοδιάγραμμα της πορείας ανάπτυξης του OpenFlow.	16
Γ'.7	Εσωτερική δομή ενός OpenFlow μεταγωγέα.	18
Γ'.8	Η είσοδος μιας εγγραφής στον πίνακα-ροής και οι επικεφαλίδες της.	21
Γ'.9	Αναλυτικό παράδειγμα ανταλλαγής μηνυμάτων για την εγκαθίδρυση σύνδεσης.	22
Γ'.10	Αναλυτικό παράδειγμα ανταλλαγής μηνυμάτων για τον τερματισμό σύνδεσης.	23
Γ'.11	Αναλυτικό παράδειγμα ανταλλαγής μηνυμάτων για την εγκαθίδρυση TCP σύνδεσης του μεταγωγέα με τον Controller.	24
Γ'.12	Αναλυτικό παράδειγμα ανταλλαγής μηνυμάτων για την εγκαθίδρυση TCP σύνδεσης του χρήστη με τον εξυπηρετητή σε OpenFlow δίκτυο.	27
Δ'.13	Στήσιμο του Περιβάλλοντος Προσομοίωσης.	30
Δ'.14	Τοπολογία παραδοσιακού Δικτύου.	31
Δ'.15	Τοπολογία SDN Δικτύου.	32
Ε'.16	Σύγκριση των δικτύων με βάση το πλήθος των χρηστών.	36
Ε'.17	Σύγκριση των δικτύων με βάση τη χρονική καθυστέρηση για 10Mbps.	37
Ε'.18	Σύγκριση των δικτύων με βάση το data rate.	38
Ε'.19	Σύγκριση των δικτύων με βάση το μέγεθος του αρχείου για data rate 20Mbps και αριθμός χρηστών 10.	39

Ε'.20 Σύγκριση των δικτύων με βάση το μέγεθος του αρχείου για <i>data rate 20Mbps</i> και αριθμός χρηστών 20.	40
Ε'.21 Σύγκριση των δικτύων με βάση το μέγεθος του αρχείου για <i>data rate 10Mbps</i> και αριθμός χρηστών 10.	41
Ε'.22 Σύγκριση των δικτύων με βάση το μέγεθος του αρχείου για <i>data rate 10Mbps</i> και αριθμός χρηστών 20.	42
Ε'.23 Χρήση όλων των καναλιών σε μια προσομοίωση.	42
Ε'.24 Χρήση των τριών μονοπατιών για τις διάφορες τιμές του πλήθους των χρηστών.	43

Κεφάλαιο Α΄

Εισαγωγικά Στοιχεία

ΕΚΙΝΩΝΤΑΣ την ανάγνωση της παρούσας πτυχιακής εργασίας, γίνεται μια εισαγωγή στο θέμα και τους λόγους για τους οποίους επιλέχθηκε. Έπειτα, αναφέρεται η βιβλιογραφική έρευνα που προηγήθηκε για την απόκτηση των κατάλληλων γνώσεων, ώστε να ολοκληρωθεί η εκπόνηση της εργασίας. Επίσης, παρουσιάζεται μια αρχική εικόνα των αποτελεσμάτων και της δομής της.

Α΄.1 Εισαγωγή

Στις μέρες μας, οι απαιτήσεις των δικτυακών πόρων αυξάνονται ραγδαία. Οι διαχειριστές τους δεν μπορούν να χρησιμοποιήσουν την πλήρη δύναμη των δικτύων, λόγω της παλαιάς τους δομής. Αυτό μπορεί να αποφευχθεί με τη χρήση μιας τεχνολογίας που ονομάζεται Software-Defined Networking ή SDN [26]. Πιο συγκεκριμένα, είναι μια τεχνολογία που διαχωρίζει το κανάλι μετάδοσης δεδομένων από το κανάλι λήψης αποφάσεων δρομολόγησής τους.

Συνεπώς, το θέμα που επιλέχθηκε, είναι η παρουσίαση αυτής της τεχνολογίας, με όρους κατανοητούς από τον αναγνώστη, ώστε να εισαχθεί στις πιο πολύπλοκες έννοιες και να διευκρινιστεί ο τρόπος χρήσης της. Αναλυτικότερα, για να δημιουργηθεί ένα δίκτυο που βασίζεται στην παραπάνω τεχνολογία χρειάζεται τρία βασικά βήματα. Αρχικά, η τοποθέτηση μιας κεντρικής οντότητας, η οποία θα έχει την πλήρη εικόνα του δικτύου και θα μπορεί να παίρνει αποφάσεις για αυτήν. Κατόπιν, γίνεται η αντικατάσταση ή η αναβάθμιση των μεταγωγών του δικτύου ώστε να μπορούν να υποστηρίξουν κάποιο πρωτόκολλο επικοινωνίας μεταξύ της κεντρικής οντότητας. Τέλος, πρέπει να εισαχθεί αυτό το πρωτόκολλο στο δίκτυο σύμφωνα με

το οποίο θα συνδέονται μεταξύ τους όλες οι συσκευές. Ένα από τα πιο γνωστά για τη χρήση της SDN τεχνολογίας είναι το OpenFlow [38] το οποίο και θα παρουσιαστεί αναλυτικά στην παρούσα εργασία.

Το ενδιαφέρον υλοποίησης της συγκεκριμένης εργασίας οφείλεται στο γεγονός εκμάθησης αυτών των τεχνολογιών, ώστε να μπορούν να εφαρμοστούν σε ένα αληθινό περιβάλλον. Παρά το γεγονός ότι αυτό δεν γίνεται να υλοποιηθεί στην παρούσα πτυχιακή, εκτελέστηκαν μερικά πειράματα σύγκρισης εικονικών δικτύων σε περιβάλλον προσομοίωσης. Τα δύο δίκτυα διακρίνονται σε ένα παραδοσιακό δίκτυο και σε ένα δίκτυο που εφαρμόζει την τεχνολογία του SDN. Στη συνέχεια, δημιουργείται υπερβολική κίνηση στο δίκτυο ώστε να μπορέσει να εφαρμοστεί μια εφαρμογή ισορρόπησης του φόρτου στο δίκτυο με SDN. Τέλος, παρουσιάζονται οι διαφορές στις αποδόσεις των δυο δικτύων.

Α'.2 Βιβλιογραφική Έρευνα

Από την έρευνα που έγινε σε βιβλιογραφία δε βρέθηκε η υλοποίηση των ίδιων πειραμάτων μέσω του προγράμματος προσομοίωσης OMNeT++ [57] (Θα γίνει αναλυτική του παρουσίαση στο Κεφάλαιο Δ'). Δηλαδή σύγκριση δύο δικτύων, ενός παραδοσιακού και ενός SDN που χρησιμοποιεί μιας εφαρμογή ισορρόπησης φόρτου.

Παρόλα αυτά, σαν βάση αυτής της εργασίας ήταν το άρθρο που αναφέρεται στην χρήση του SDN με το OMNeT++ [17]. Στο συγκεκριμένο άρθρο, παρουσιάστηκε ένα πακέτο, δηλαδή ένα σύνολο από κομμάτια κώδικα, που λειτουργούν μεταξύ τους, για να υλοποιήσουν το πρωτόκολλο OpenFlow. Καθώς επίσης και η υλοποίηση των κατάλληλων κόμβων για να υπάρξει ενός τέτοιου είδους δίκτυο. Για την ακρίβεια, τους Controllers και τους μεταγωγείς που υποστηρίζουν OpenFlow πρωτόκολλο. Στόχος τους ήταν η κατάλληλη τοποθέτηση ενός Controller σε ένα μεγάλο δίκτυο που καλύπτει την έκταση μιας ηπείρου. Με την προϋπόθεση, να μην υπάρχει μείωση στην απόδοση κατά τη διάρκεια επικοινωνίας του Controller με τους μεταγωγείς.

Κάποιες ακόμα βιβλιογραφίες που βοήθησαν στην υλοποίηση αυτής της εργασίας, αναφέρθηκαν στη γενική εκτίμηση απόδοσης των OpenFlow δικτύων στο περιβάλλον του OMNeT++. Όπως για παράδειγμα, ο χρόνος μετάδοσης μετ' επιστροφής (RTT) για την αποστολή δεδομένων [10, 6]. Επιπροσθέτως, συνέβαλαν στην ανάλυση των μαθηματικών που χρειάζονται για να γίνει αποτελεσματική ισορρόπηση φόρτου σε κέντρα δεδομένων με τη χρήση του SDN [33].

Ακόμα μια βιβλιογραφία, παρουσίαζε για το κατάλληλο δέντρο έκτασης (spanning tree), ώστε να έχει τον καλύτερο RTT στην περίπτωση που υπάρχουν παραπάνω από ένας Controller [29].

Συνοψίζοντας, για την εκτέλεση των παρόντων πειραμάτων έγινε ένας συνδυασμός από την παραπάνω βιβλιογραφία. Δηλαδή, συνδυάστηκε η ισορρόπηση φόρτου σε διάφορα μονοπάτια, το πακέτο από το βιβλίο για την προσομοίωση OpenFlow δικτύων στο OMNeT++ και τέλος, η ιδέα σύγκρισης TCP σύνδεσης μεταξύ ενός παραδοσιακού δικτύου και ενός με SDN.

Α΄.3 Αποτελέσματα και Συνεισφορά

Κατά την ολοκλήρωση των πειραμάτων, συλλέχθηκαν αρκετά αποτελέσματα και ανεγέρθηκαν ορισμένα συμπεράσματα. Με άλλα λόγια, συγκεντρώθηκαν δεδομένα όπως ο RTT, πόσα πακέτα μεταδόθηκαν από μια συσκευή ανά δευτερόλεπτο, οι συνολικές συγκρούσεις που έγιναν και πολλές ακόμα. Πιο σημαντικές όμως, ήταν ο συνολικός χρόνος μετάδοσης ενός συγκεκριμένου μεγέθους αρχείου και η χρησιμοποίηση του κάθε μονοπατιού. Χάρη σε αυτά τα αποτελέσματα, φάνηκε η βελτίωση που παρέχει η προσθήκη της τεχνολογίας του SDN σε ένα δίκτυο μιας και η χρησιμοποίηση των μονοπατιών μοιράστηκε και ο συνολικός χρόνος μειώθηκε.

Κατά συνέπεια, διαβάζοντας αυτήν την εργασία μπορεί κάποιος να μάθει πως μπορεί να προσομοιώσει εικονικά δίκτυα με την προσθήκη του OpenFlow και να κάνει πιο πολύπλοκα πειράματα. Επίσης, παρουσιάστηκε η λειτουργικότητα που είχε μια εφαρμογή ισορρόπησης φόρτου στο OMNeT++, οι δυσκολίες που μπορούν να προκύψουν και πως να παραλειφθούν.

Α΄.4 Δομή Εργασίας

Η παρούσα πτυχιακή εργασία συντάχθηκε με τρόπο τέτοιο ώστε να γίνει ευκόλως κατανοητή από τον αναγνώστη. Ακολουθήθηκε μια σειρά κατά την οποία από πιο γενικές έννοιες, έγινε μετάβαση σε πιο ειδικές. Τέλος, παρουσιάστηκαν λεπτομερώς τα πειράματα και επεξηγήθηκαν τα συμπεράσματά τους. Παρακάτω, αναφέρεται η αναλυτική της δομή.

Στην αρχή, έγινε αναφορά στη γενική τεχνολογία που στοχεύει αυτή η πτυχιακή. Δηλαδή, στα δίκτυα που εφαρμόζουν SDN για να βελτιώσουν την απόδοσή τους. Αυτό συμβαίνει χωρίζοντας τη μετάδοση των δεδομένων, από τη λήψη αποφάσεων για αυτή. Γίνεται μια σύντομη παρουσίαση της ιστορίας που έχει το SDN, η χρησιμότητα της εφαρμογής στα μοντέρνα δίκτυα,

που εφαρμόζεται, και τέλος η αναλυτική δομή του.

Ύστερα, αναλύθηκε ένα απο τα πιο διαδεδομένα πρωτόκολλα επικοινωνίας SDN δικτύων, δηλαδή το OpenFlow. Αναλυτικότερα, γίνεται αναφορά στη δημιουργία και στην πορεία ανάπτυξης του πρωτοκόλλου. Μέτα, πως αυτό εφαρμόζεται στους μεταγωγείς και στους Controllers, αλλά και ο πίνακας-ροής που περιέχει πληροφορίες για τη μεταφορά του κάθε πακέτου. Συμπληρωματικά, γίνεται μια αναφορά στο πως γίνεται η μεταφορά πακέτων σε μια παραδοσιακή TCP σύνδεση και στη συνέχεια σε μια OpenFlow σύνδεση. Στο τελευταίο υποκεφάλαιο, αναφέρονται οι διάφορες προσθήκες στην εκάστοτε έκδοση του OpenFlow.

Κατόπιν, παρουσιάστηκε ο τρόπος με τον οποίο έγιναν τα πειράματα και το πρόγραμμα προσομοίωσης τους. Αυτό έγινε με την εξής σειρά, πρώτα η κύρια ιδέα των πειραμάτων, μετά πως γίνεται η εγκατάσταση και η χρήση του περιβάλλοντος προσομοίωσης. Στη συνέχεια, εμφανίζεται η τοπολογία και οι χρήσιμες μεταβλητές και τέλος, παρουσιάστηκαν κάποιες παρατηρήσεις και προβλήματα που αντιμετωπίστηκαν.

Τέλος, παρουσιάζονται τα διάφορα πειράματα που εκτελέστηκαν αλλά και τα αποτελέσματα τους, μαζί με έναν σύντομο σχολιασμό τους. Συνοψίζοντας, ακολουθεί μια ανακεφαλαίωση όσων προηγούνταν, τα συμπεράσματα και η συνεισφορά της παρούσας πτυχιακής.

Κεφάλαιο Β΄

Περιγραφή της SDN Λειτουργίας

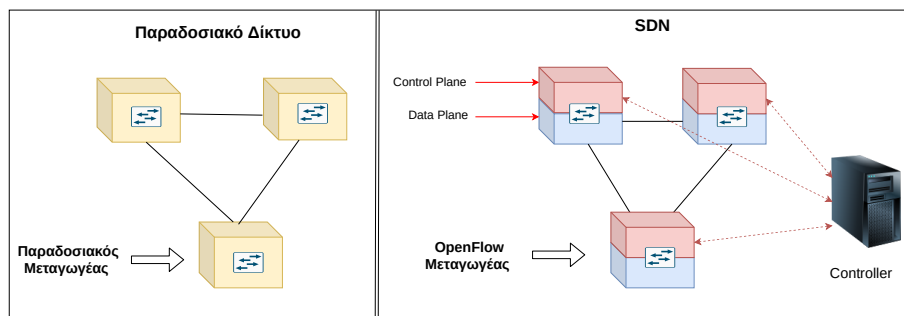
Σε αυτό το κεφάλαιο θα παρουσιαστεί ο ορισμός και η ανάλυση του software-defined networking, γνωστό και ως SDN. Θα γίνει μια σύντομη απόδοση ορισμού, στην συνέχεια θα γίνει αναφορά στη χρονολογική του εξέλιξη. Έπειτα, μερικές από τις εφαρμογές που χρησιμοποιούν SDN και τέλος η δομή του.

Β΄.1 Ορισμός

Με τον όρο SDN ή στα ελληνικά, δίκτυα βασισμένα σε λειτουργικό, αναφερόμαστε στη γενικότερη ιδέα υλοποίησης ενός δικτύου το οποίο είναι πλήρως προγραμματιζόμενο και στοχεύει στην απόδοση, στην ευελιξία και στον έλεγχο για σφάλματα σε σχέση με τη διαχείριση ενός παραδοσιακού δικτύου [7]. Αυτό πραγματοποιείται με τον διαχωρισμό της διαχείρισης από την προώθηση των δεδομένων του με σκοπό τη βελτίωσή του [39]. Πιο συγκεκριμένα, προσφέρονται τα οφέλη του συνδυασμού των “system virtualization” και “cloud computing” καθώς παρέχεται μια καθαρή εικόνα του δικτύου με κεντρικό ευφυή έλεγχο [7]. Αυτός ο έλεγχος πραγματοποιείται στο “Control Plane” από τον εγκέφαλο του δικτύου, ο οποίος ονομάζεται “Controller”, και μπορεί να είναι παραπάνω από ένας [21, 48]. Αντιθέτως, σύμφωνα με τις εντολές του Controller, η προώθηση των δεδομένων πραγματοποιείται στο “Data Plane”.

Το πιο ευρέως διαδεδομένο παράδειγμα του SDN είναι το OpenFlow, το οποίο θέτει ορισμένους κανόνες στους μεταγωγείς του δικτύου και στη διαχείριση των πακέτων και των διαδρομών [38]. Επίσης σε συνδυασμό με το “Hyper-Flow”, ένα “Openflow” δίκτυο μπορεί να χωριστεί σε επιμέρους πεδία με δικό τους Controller το κάθε ένα, ώστε να υπάρχει

ισορροπία στον φόρτο του δικτύου και στην ελαστικότητα [53]. Στο σχήμα Β'.1 φαίνονται οι διαφορές μεταξύ ενός παραδοσιακού δικτύου και ενός που κάνει χρήση του SDN. Όπως μπορεί να παρατηρηθεί, στο παραδοσιακό δίκτυο, οι μεταγωγείς επικοινωνούν μεταξύ τους χωρίς να εμπλέκεται κάπως ο Controller. Απο την άλλη, στο SDN, οι μεταγωγείς έχουν χωριστεί στην μέση ξεχωρίζοντας το Data Plane απο το Control Plane. Καθώς επίσης, δέχονται εντολές μέσω του Control Plane απο τον Controller.



Σχήμα Β'.1: Παραδοσιακό Δίκτυο vs SDN.

Β'.2 Ιστορία του SDN

Ο στόχος του SDN είναι να προσθέσει καινοτομίες σε ένα συνηθισμένο δίκτυο και να κάνει γενικά πιο απλουστευμένη τη διαχείρισή του. Αυτή η τεχνολογία δεν είναι απαραίτητα καινούργια μιας και ο προγραμματισμός ενός τέτοιου δικτύου είχε επιχειρηθεί με το Active Networking στη δεκαετία του 1990 [15, 54]. Επίσης, η πρώτη φορά που διαχωρίστηκε το “Control Plane” ήταν από την “ForCES Working Group (WG)” στη δεκαετία του 2000 [15, 54].

Έπειτα, στα μέσα τις δεκαετίας του 2000, οι επιστήμονες του χώρου εντυπωσιάστηκαν από τις δυνατότητες που θα προσέφερε η ιδέα ενός SDN και προσπάθησαν να το υλοποιήσουν σε πολύ μεγαλύτερες κλίμακες [15]. Πιο συγκεκριμένα, το επιχείρησαν σε επιταχυντές και τηλεσκόπια αλλά δεν τα κατάφεραν, οπότε έστρεψαν την προσοχή τους σε δίκτυα που χρησιμοποιούν οι πανεπιστημιούπολεις [38]. Στη συνέχεια, προσπάθησαν να υλοποιήσουν ένα όραμα όπου θα υπάρχουν πλήρως προγραμματιζόμενα δίκτυα αλλά επίσης θα μπορούν να αναπτυχθούν στο πραγματικό κόσμο [15]. Αυτός ο στόχος επιτεύχθηκε με τη δημιουργία του OpenFlow [38], το οποίο μπορούσε να δουλέψει πάνω στους ήδη υπάρχοντες μεταγωγείς και έδινε περισσότερες δυνατότητες στον έλεγχο ενός δικτύου από οτιδήποτε άλλο εκείνη την εποχή. Αμέσως

μετά, ακολούθησε η δημιουργία των OpenFlow API με την κατασκευή πλατφόρμας ελέγχου (Controller platform) τύπου NOX [18] οι οποίες βοηθούσαν στην προσθήκη νέων εφαρμογών ελέγχου.

Καθώς η κλίμακα του μεγέθους των δικτύων αυξάνεται, δημιουργείται η ανάγκη για χρήση περισσότερων Controllers. Για να πετύχει αυτό, θα πρέπει όλοι οι Controllers να δουλεύουν σαν έναν κεντρικό λογικό Controller και αυτό πραγματοποιείται με τον Onix Controller [30], ο οποίος εφάρμοσε πρώτη φορά αυτήν την ιδέα [15]. Πιο πρόσφατα, το σύστημα ONOS [8] επιτρέπει στους "Open Source Controllers" να χρησιμοποιούν ήδη υπάρχουσες εφαρμογές με σκοπό τη συντήρηση και μια κοινή εικόνα της τοπολογίας του δικτύου σε όλους τους Controllers [15].

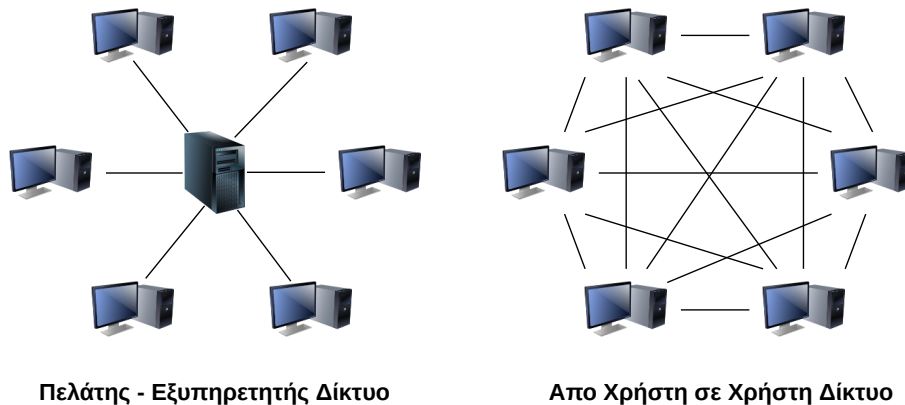
Τέλος, ως επέκταση του SDN, έρχεται το CORD [43], μια τεχνολογία όπου οι διαχειριστές των δικτύων θα μπορούν να βελτιώσουν τα δίκτυά σε διάφορους τομείς. Πιο συγκεκριμένα, στην αποδοτικότητα, στον προγραμματισμό, στην ελαστικότητα, στις μεγάλες απαιτήσεις εύρους ζώνης αλλά και στο να δημιουργήσουν νέες ροές εσόδων με καινοτόμες υπηρεσίες [43]. Αυτό επιτυγχάνεται με τον συνδυασμό τριών λειτουργιών. Αρχικά με το SDN, δηλαδή τον διαχωρισμό του Data Plane από το Control Plane. Έπειτα με το NVF [24, 34] που μεταφέρει το Data Plane από τις εφαρμογές του υλικού (Hardware) σε εικονικά μηχανήματα. Και τέλος του νέφους (Cloud) [20], το οποίο βοηθάει στην κατασκευή επεκτάσιμων υπηρεσιών [43].

Β'.3 Χρησιμότητα και εφαρμογές

Β'.3.1 Χρησιμότητα

Με την προσθήκη νέων τεχνολογιών, την ανάπτυξη των ήδη υπάρχουσων και την αύξηση της πολυπλοκότητας σε ένα δίκτυο, δημιουργείται η ανάγκη για ένα πιο προσαρμόσιμο και έξυπνο δίκτυο. Πιο συγκεκριμένα, η χρήση όλο και περισσότερων κινητών συσκευών και τεχνολογιών όπως ο υπολογισμός και η οπτικοποίηση με τη χρήση νέφους (Cloud), επηρεάζουν τα μοτίβα μεταφοράς των δεδομένων [45]. Πλέον, χρειάζεται ο σχεδιασμός ενός δικτύου με ανάγκες για δυναμικό υπολογισμό και αποθήκευση μεγάλων δεδομένων για κέντρα δεδομένων επιχειρήσεων και πανεπιστημιακών [40]. Επίσης, θα ήταν πολύ δύσκολο οι διαχειριστές αυτών των δικτύων να ρυθμίζουν πολλαπλές μεμονωμένες συσκευές κάθε φορά που γίνεται μια μικρή αλλαγή στο δίκτυο [45]. Μερικές από τις νέες τάσεις που οδηγούν σε αλλαγή των σύγχρονων δικτύων είναι :

- Οι αλλαγές στα μοτίβα δρομολόγησης [3]. Δηλαδή, όταν δεν υπήρχε πολλή κίνηση στο διαδίκτυο ή γενικότερα σε μικρά δίκτυα γινόταν η χρήση της επικοινωνίας πελάτη με εξυπηρετητή. Σε αντίθεση με τα σημερινά πρότυπα που γίνεται περισσότερο η χρήση του δικτύου διομότιμων υπολογιστών ή peer-to-peer(p2p), η οποία είναι αρκετά πιο βολική αλλά πιο περίπλοκη στη δρομολόγηση.



Σχήμα Β'.2: Δίκτυο πελάτη-εξυπηρετητή με δίκτυο peer-to-peer.

- Η μεγάλη χρήση των μικρο-συσκευών ή 'Internet of Things' (IoT). Με την εξέλιξη της τεχνολογίας όλο και περισσότερα δίκτυα ενσωματώνουν μικρές συσκευές του τύπου έξυπνα κινητά, ασύρματοι εκτυπωτές, κάμερες και διαφόρων ειδών μετρητές. Όλες αυτές οι συσκευές εκμεταλλεύονται πόρους του δικτύου για να επικοινωνήσουν, καθώς χρειάζονται ειδική μεταχείριση ώστε να παραμείνουν τα δεδομένα αναλλοίωτα και ασφαλή.
- Η έγερση των υπηρεσιών νέφους. Ειδικότερα, αυξήθηκαν οι απαιτήσεις για άμεση χρήση πόρων πληροφορικής σε ιδιωτικές αλλά και σε δημόσιες υπηρεσίες νέφους. Αυτή η διάδοση των πόρων θα πρέπει να γίνεται πάντα σε περιβάλλον με αυξημένη ασφάλεια, ταχύτητα, με υψηλό έλεγχο και ευχρηστία για την αναδιαμόρφωση και την αλλαγή σε άμεσο χρονικό διάστημα [58].
- Η μετάδοση μεγάλων δεδομένων. Τα περισσότερα κέντρα δεδομένων, έχουν πρόσβαση σε πολύ μεγάλο όγκο δεδομένων ή σετ δεδομένων, τα οποία πρέπει να τα διαμοιράσουν και να τα επεξεργαστούν από πολλούς εξυπηρετητές. Οι εξυπηρετητές αυτοί είναι δι-

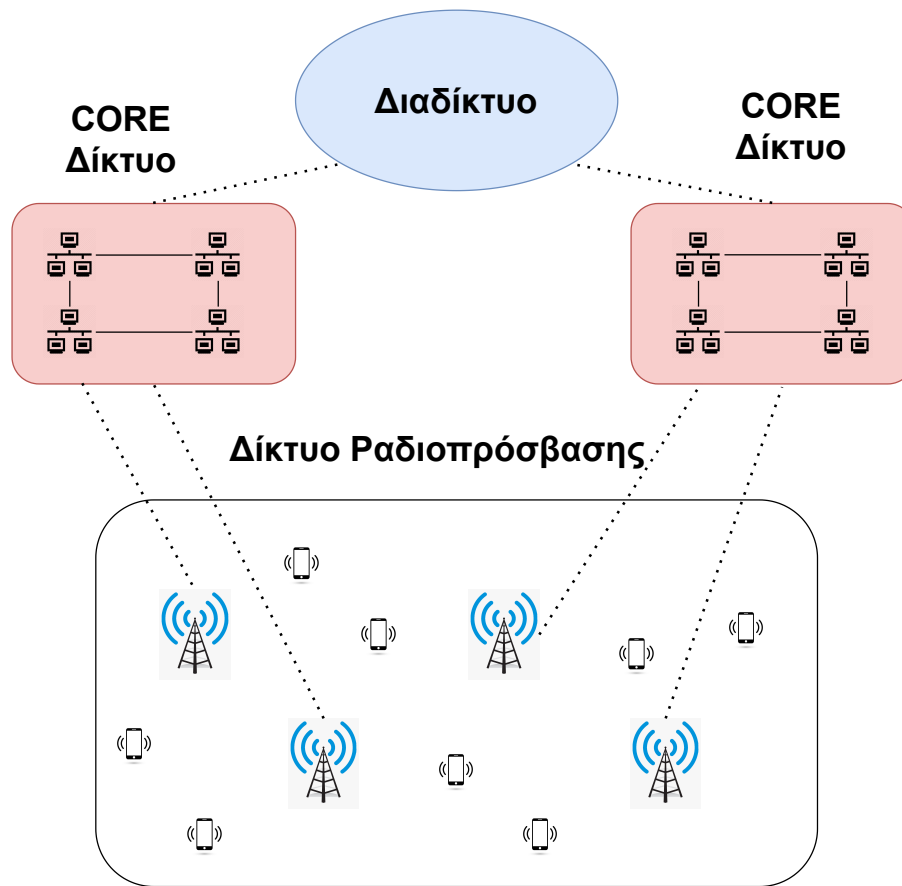
άσπαρτοι σε διάφορα μέρη του κόσμου και πρέπει να συγχρονίζονται άμεσα χωρίς προβλήματα. Επομένως, οι διαχειριστές αυτών των δικτύων θα πρέπει να είναι μονίμως σε θέση να τα επεκτείνουν για να καλύψουν τις ανάγκες χωρητικότητας [58, 5].

Συμπεραίνοντας, πολλές από τις δυσκολίες που αντιμετωπίζουν οι διαχειριστές δικτύων με τις νέες τεχνολογικές απαιτήσεις μπορούν να αντιμετωπιστούν με τη χρήση του SDN. Αυτό σημαίνει ότι η μεταβολή του παραδοσιακού δικτύου σε SDN θα επιλύσει προβλήματα όπως η επεκτασιμότητα, ο συγχρονισμός, η ταχύτητα και η ασφάλεια.

Β'.3.2 Εφαρμογές

Η χρησιμότητα του SDN είναι εμφανής και έχουν υπάρξει αρκετές εφαρμογές και περιπτώσεις στις οποίες χρησιμοποιείται. Αρχικά υπάρχει για SDMN (Software Defined Mobile Networking) [46], δηλαδή μια προσπάθεια σχεδίασης κινητών δικτύων [42, 35]. Όλα τα πρωτόκολλα υλοποιούνται σε λογισμικό, οπότε θα λειτουργούν σε όλες της γενικής χρήσης συσκευές για την επικοινωνία διαφόρων τοπικών δικτύων με δίκτυα ραδιοπρόσβασης [12], όπως φαίνεται στο Σχήμα Β'.3. Έπειτα, υπάρχει η χρήση σαν SD-WAN (Software Defined - Wireless Area Network) δηλαδή για ασύρματα δίκτυα. Σε μεγάλα ασύρματα δίκτυα υπάρχει πρόβλημα με την αποστολή πακέτων από κόμβο σε κόμβο, διότι υπάρχει πτώση της απόδοσης και μεγάλος όγκος εισόδου στον πίνακα προορισμών (flow table) [37]. Συνεπώς, γίνεται χρήση της ειδικής MPLS ετικέτας στα πακέτα, η οποία περιέχει πληροφορίες δρομολόγησης. Πιο συγκεκριμένα, σημειώνεται μια φορά το μονοπάτι που πρέπει να ακολουθήσουν τα πακέτα και με την τεχνική "Source Routing", που υποστηρίζει το SD-WAN γίνεται αποδοτικότερη η μεταφορά τους [37]. Τέλος υπάρχει το SD-LAN το οποίο ακολουθεί τις αρχές του SDN αλλά σε ένα τοπικό δίκτυο. Αυτό συμβαίνει για να καλύψει θέματα ασφάλειας και επιπλέον εφαρμογές στον έλεγχο του δικτύου, σε ενσύρματη, αλλά και ασύρματη σύνδεση.

Σειρά έχει η αναφορά σε συγκεκριμένες εφαρμογές του SDN σε θέματα ασφάλειας [52]. Αρκετές από τις προσπάθειες που υλοποιήθηκαν σε εφαρμογές του Controller για να αυξήσουν την ασφάλεια σε ένα δίκτυο είχαν διαφορετικές κατευθύνσεις. Πιο συγκεκριμένα, ανίχνευση και μείωση των επιθέσεων τύπου κατανεμημένη άρνηση πόρων και υπηρεσιών "Distributed Denial of Service (DDoS)" [9, 16]. Επιθέσεις τύπου botnet [14], δηλαδή πολλές συσκευές συνδεδεμένες στο διαδίκτυο, όπου στην κάθε μια λειτουργεί ένα ή παραπάνω μποτ, ή και επιθέσεις τύπου σκουληκιών [27]. Η ιδέα για την αντιμετώπιση αυτών των επιθέσεων βασίζεται στις αρχές του πρωτοκόλλου OpenFlow για τη συλλογή δεδομένων από το Data Plane. Έ-



Σχήμα Β'3: RAN και Core Δίκτυα.

πειτα, εφαρμόζονται αλγόριθμοι κατάταξης (Classification), οι οποίοι ανιχνεύουν αλλαγές στο δίκτυο και στην συνέχεια ο Controller επιλέγει τι θα κάνει με αυτές.

Ένα άλλο είδος εφαρμογής για την ενίσχυση ασφάλειας του δικτύου είναι τεχνικές αλγορίθμων κινούμενου στόχου. Ειδικότερα, να αλλάζουν οι κυρίες μεταβλητές κλειδιά για το δίκτυο ή να παραμένουν κρυφές, ώστε να δυσκολεύεται ο επιτιθέμενος. Τέτοιες τεχνικές άμυνας είναι αδύνατον να εφαρμοστούν σε ένα παραδοσιακό δίκτυο χωρίς να υπάρχει ένας κεντρικός εγκέφαλος, ο οποίος θα παίρνει και θα συντονίζει αυτές τις αποφάσεις. Αλλά με τη χρήση ενός Controller μπορεί να κάνει διάφορες τεχνικές, όπως να φτιάχνει εικονικές διευθύνσεις IP κρύβοντας τις πραγματικές και κάνοντας συσχετίσεις [23]. Ένα ακόμα παράδειγμα θα ήταν να δείχνει ότι υπάρχουν ψευδείς ανοιχτές πόρτες σε διάφορες συσκευές του δικτύου για να προσθέτει πολυπλοκότητα στον επιτιθέμενο [28].

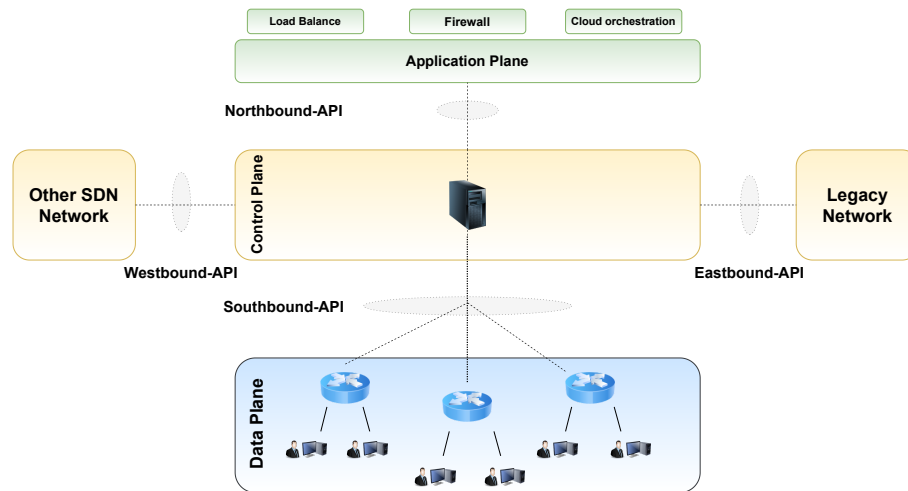
Δε θα έπρεπε να παραληφθεί η αξία που έχουν στην ασφάλεια εφαρμογές, όπως το FlowVisor [49] και το FlowChecker [4]. Αναλυτικότερα, το FlowVisor χρησιμοποιεί ένα μόνο επίπεδο προώθησης υλικού, που μοιράζεται πολλαπλά διαχωριζόμενα λογικά δίκτυα. Αυτό το υλικό χρησιμοποιείται για την παραγωγή, την ανάπτυξη, τη διαχωριζόμενη επιτήρηση, τη διαμόρφωση καθώς και την κίνηση του διαδικτύου. Το καθένα από τα παραπάνω θα έχει τη δικιά του λογική τοπολογία, η οποία ονομάζεται φέτα (slice). Σε συνδυασμό με την προηγούμενη τεχνολογία, το FlowChecker εφαρμόζει τους κανόνες του πρωτοκόλλου OpenFlow, που έχουν τεθεί από τους χρήστες της ανάλογης φέτας [49].

Ένα ακόμα πεδίο, το οποίο βελτιώνεται αρκετά από την εφαρμογή του SDN είναι η μεταφορά μεγάλων γκρουπ δεδομένων. Στα μεγάλα κέντρα δεδομένων, όταν πολλοί χρήστες ζητούν ένα γκρουπ δεδομένων, ο αποστολέας θα πρέπει να στείλει αντίγραφα σε όλους του παραλήπτες. Οι προϋπάρχουσες τεχνολογίες δεν είναι πολύ αποτελεσματικές σε μεγάλη κλίμακα δικτύου ή σε περιορισμένο εύρος ζώνης [11]. Οι μεταγωγείς που χρησιμοποιούν SDN μπορούν να αντιμετωπίσουν τέτοιου είδους προβλήματα. Πιο συγκεκριμένα, η τοποθέτηση ενός Controller στο δίκτυο θα βοηθήσει να αποφευχθούν μονοπάτια τα οποία έχουν πολύ φόρτο μεταφοράς και να μειωθεί η συσσώρευση. Έτσι δημιουργούνται οι κατάλληλες τοπολογίες που χρησιμοποιούν καλύτερα το εύρος ζώνης του δικτύου με τεχνολογίες όπως το DCCast [61] και το QuickCast [41].

B'.4 Δομή του SDN

Η βάση της τεχνολογίας SDN συνίσταται από τουλάχιστον έναν Controller, κάποιου είδους εφαρμογές που τρέχουν στον Controller, υλικό δρομολόγησης πακέτων γενικής χρήσης που να υποστηρίζουν την τεχνολογία του SDN. Ο Controller μπορεί να είναι μια λογική κεντρικοποιημένη οντότητα, που υλοποιεί τις εφαρμογές του επιπέδου εφαρμογών, άλλα και συλλέγει στατιστικές πληροφορίες και γεγονότα που συμβαίνουν στο δίκτυο. Το υλικό που θα υποστηρίζει το επίπεδο μεταφοράς πρέπει να αποτελείται από μεταγωγείς, δρομολογητές ή και σημεία πρόσβασης με την προϋπόθεση ότι όλα υποστηρίζουν κάποιο πρωτόκολλο επικοινωνίας για SDN, όπως το OpenFlow. Εμβανύνοντας, οι τέσσερις κύριες διεπαφές που χαρακτηρίζουν το SDN είναι το Southbound-API, το Northbound-API, το Westbound-API και το Eastbound-API [26].

- Το Southbound-API παρουσιάζει τη διεπαφή που έχει ο Controller με το Data Plane



Σχήμα Β'.4: Δομή SDN.

[47]. Δηλαδή η διεπαφή που μεταφέρει τις εντολές της κεντριοποιημένης ιδέας του SDN στο υπόλοιπο δίκτυο [26]. Αυτή ακολουθείται από κάποιο σετ κανόνων και προτύπων, όπως για παράδειγμα το IETF ForCES [15] και το OpenFlow [38].

- Το Northbound-API επιτρέπει στο SDN να δέχεται την ανταλλαγή πληροφοριών με τις εφαρμογές που τρέχουν πάνω στο δίκτυο [26, 60]. Συνδέει τον Controller με το Application Control Plane, και μπορούν να υπάρχουν παραπάνω από μια εφαρμογές και για διάφορους κόμβους η κάθε μια [47]. Πιο συγκεκριμένα, εφαρμογές όπως κάποιο Firewall, κάποιο πρόγραμμα που να επιλέγει το βέλτιστο μονοπάτι για να μην υπάρχει συμφόρηση, ή κάποια εφαρμογή συγχρονισμού νέφους.
- Το Westbound-API το οποίο συνδέει πολλαπλούς Controllers από διαφορετικά SDN δίκτυα [25]. Επιτρέπει την ανταλλαγή της κατάστασης του κάθε δικτύου που μπορεί να επηρεάσουν τις αποφάσεις ενός άλλου Controller, καθώς επίσης και την αδιάκοπη διευθέτηση της ροής των δικτύων σε πολλαπλούς τομείς [26].
- Το Eastbound-API το οποίο συμβάλει στην επικοινωνία με άλλα δίκτυα, τα οποία δεν υποστηρίζουν SDN [26]. Ένα συνηθισμένο παράδειγμα χρήσης αυτής της διεπαφής είναι οι ετικέτες MPLS [13]. Η υλοποίηση αυτής της διεπαφής βασίζεται στη μεριά του δικτύου που δε χρησιμοποιεί τεχνολογία SDN. Ιδανικά χρειάζεται μια τεχνολογία μετάφρασης μεταξύ των δυο διαφορετικών δικτύων, ώστε αυτές οι τοπολογίες να είναι συμβατές μεταξύ τους [26]. Για παράδειγμα, το SDN δίκτυο θα έπρεπε να μπορεί να χρη-

σιμοποιήσει τα πρωτόκολλα δρομολόγησης του παραδοσιακού δικτύου ή των ετικετών MPLS.

Στο SDN το Control Plane μπορεί να έχει κεντρικοποιημένη, ιεραρχική ή κατανεμημένη δομή. Η αρχική ιδέα του SDN, είναι να υπάρχει μια κεντρική οντότητα η οποία θα έχει την πλήρη εικόνα του δικτύου και θα μπορεί να πάρει αποφάσεις. Αν και αυτή είναι η πιο λογική υλοποίηση, αντιμετωπίζει προβλήματα όσο το μέγεθος ή η δυναμική του δικτύου κλιμακώνεται. Προς την επίλυση αυτών των προβλημάτων, παρουσιάζεται η ιεραρχική και η κατανεμημένη τοποθέτηση παραπάνω του ενός Controller. Στην ιεραρχική δομή, ο κάθε Controller ελέγχει ένα συγκεκριμένο μέρος του δικτύου για συγκεκριμένες αποφάσεις, ενώ για όλο το δίκτυο υπάρχει ένας ριζικός (root) Controller [19, 2]. Από την άλλη μεριά στην κατανεμημένη τοποθέτηση, ο κάθε Controller ελέγχει το τοπικό του δίκτυο και έπειτα οι Controllers ανταλλάζουν μηνύματα συγχρονισμού μεταξύ για να εμπλουτίσουν τις γνώσεις τους [53, 31, 55].

Παρόλα αυτά, η τοποθέτηση παραπάνω από έναν Controller φέρνει και το πρόβλημα του κατάλληλου αριθμού και τοποθέτησης τους. Ένα από τα πιο γνωστά προβλήματα στα πλαίσια ενός μεγάλου δικτύου είναι της αναδρομικής καθυστέρησης μεταξύ των Controller και των συσκευών του [50]. Δε θα έπρεπε να παραλειφθούν προβλήματα όπως η αξιοπιστία μονοπατιού [22], η ανοχή σε σφάλματα [44] και οι απαιτήσεις των εφαρμογών [56].

Η προωθητική ροή του SDN βασίζεται σε OpenFlow μεταγωγείς για την αποθήκευση των flow-entries, οι οποίοι χρησιμοποιούν μια τεχνολογία μνήμης που ονομάζεται Ternary Content Addressable Memory (TCAM) [36]. Χαρακτηριστικό της είναι η ταυτόχρονη πρόσβαση της μνήμης στην ταχύτητα της γραμμής, η οποία είναι ανεπηρεάστη από το πλήθος των αποθηκευμένων εισαγωγών [32]. Όταν ένας μεταγωγέας λάβει ένα πακέτο και πρέπει να το προωθήσει, έχει τρεις τρόπους που μπορεί να ακολουθήσει. Είναι ο προληπτικός, ο αντιδραστικός και ο υβριδικός. Επεξηγηματικά, ο αντιδραστικός, κάθε φορά που δέχεται ένα πακέτο, ελέγχει το flow entry και αν δεν υπάρχουν οδηγίες ως προς την προώθηση του ρωτάει τον Controller. Ενώ στην προληπτικό υπάρχουν όλα τα αποδεκτά είδη μηνυμάτων αποθηκευμένα στον Controller και δεν υπάρχει επιπλέον καθυστέρηση στην προώθηση τους. Δηλαδή, γίνεται έλεγχος στον flow entry και αν υπάρχει εγγραφή ροής ακολουθεί την διαδικασία, αλλιώς δεν το προωθεί. Τέλος, στον υβριδικό τρόπο προϋπάρχουν στο flow entry όλες οι διαδικασίες, αλλά αν δεν υφίστανται, τότε ρωτάει τον Controller [51].

Κεφάλαιο Γ΄

Πρωτόκολλο Επικοινωνίας OpenFlow

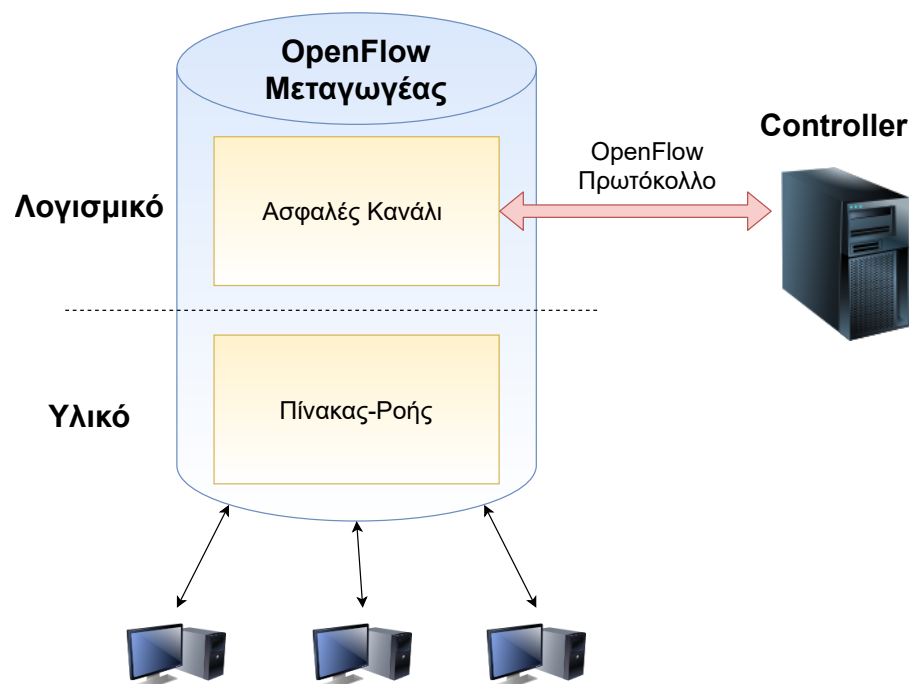
Σε αυτό το κεφάλαιο θα γίνει ειδικότερη αναφορά του SDN πάνω στο πρωτόκολλο επικοινωνίας ονόματι OpenFlow. Θα παρουσιαστεί η ιστορία και η ανάπτυξη του πρωτοκόλλου, το υλικό που είναι απαραίτητο για τη χρήση του, καθώς επίσης και οι προδιαγραφές του.

Γ΄.1 Εισαγωγή

Η ιδέα του SDN βασίζεται στην επικοινωνία ενός κεντρικού υπολογιστή με τους επιμέρους ενδιαμέσους κόμβους. Στόχος του κεντρικού υπολογιστή είναι να παίρνει όλες τις αποφάσεις και να επιτυγχάνει μια υψηλής ευελιξίας διαμόρφωση στη δρομολόγηση των πακέτων, ώστε να υπάρξει βελτίωση στη χρησιμοποίηση των διαθέσιμων δικτυακών πόρων. Στην περίπτωση κατά την οποία υπάρχει ένα τοπικό ενσύρματο δίκτυο οι ενδιαμέσοι κόμβοι είναι οι μεταγωγείς ή αλλιώς switches. Η επικοινωνία μεταξύ του κεντρικού υπολογιστή και των μεταγωγών γίνεται με το κανάλι λήψεων αποφάσεων και πάνω σε αυτό εφαρμόζεται το πρωτόκολλο OpenFlow.

Το Openflow προσφέρει ευελιξία στους χρήστες του, ώστε να μπορούν να προγραμματίσουν το πίνακα με τις ροές δεδομένων (flow table) των μεταγωγών ή των δρομολογητών. Έτσι, οι διαχειριστές των δικτύων μπορούν να επιλέξουν τις διαδρομές και τις διαδικασίες που θέλουν συγκεκριμένα πακέτα να ακολουθήσουν. Αυτό συνεπάγεται, ότι με τη χρήση του OpenFlow μπορούν να εφαρμοστούν νέα πρωτόκολλα δρομολόγησης, μοντέλα ασφαλείας, διαφορετικά σχήματα διευθύνσεων, αλλά μέχρι και παραλλαγές στον παραδοσιακό τρόπο ανταλλαγής μηνυμάτων με τις IP διευθύνσεις [38].

Ο τρόπος με τον οποίο λειτουργεί το OpenFlow περιλαμβάνει την ακόλουθη διαδικασία. Όταν ένα μήνυμα φτάσει στον μεταγωγέα, ελέγχει τον πίνακα-ροής του για την ύπαρξη μιας προγραμματισμένης εντολής για το συγκεκριμένο μήνυμα. Αυτό γίνεται στο υλικό του μεταγωγέα, και στην συνέχεια στο λογισμικό του δημιουργείται ένα ασφαλές κανάλι επικοινωνίας με τον Controller όπως φαίνεται στο Σχήμα Γ'.5. Τέλος, αν δεν υπάρχει κάποια προγραμματισμένη εντολή, ρωτάει τον Controller μέσω αυτού του καναλιού, το οποίο υποστηρίζει το πρωτόκολλο OpenFlow.



Σχήμα Γ'.5: Απλή επίδειξη ενός OpenFlow δικτύου.

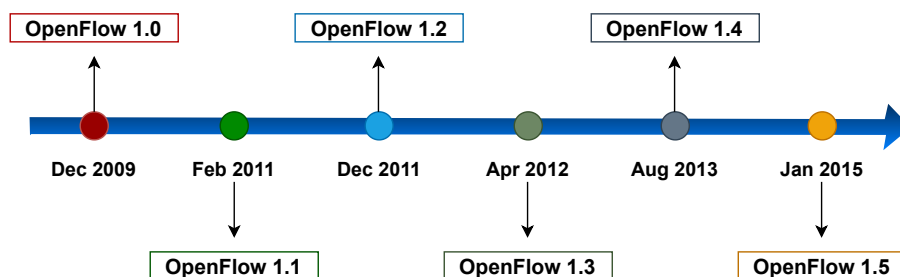
Γ'.2 Δημιουργία και πορεία ανάπτυξης

Αρχικά, οι προδιαγραφές του OpenFlow διαχειρίζονταν από μια ανοικτή ομάδα ατόμων που βρίσκονταν στο πανεπιστήμιο του Stanford [38]. Έπειτα, το 2011 ιδρύθηκε ο Open Networking Foundation (ONF), ο οποίος στόχευε να επεκτείνει το OpenFlow και να είναι το κύριο πρωτόκολλο υποστήριξης του SDN. Για την επέκτασή του, καθόρισε αρχικά μια διαδικασία που επιτρέπει στις εταιρίες μέλη του ONF να συμμετέχουν στην εξέλιξη του χωρίς να χάσει

την αρχική του ορμή [54]. Ωστόσο, ήταν απαραίτητο να διατηρηθεί η ευελιξία και ο δυναμισμός της προηγούμενης ανάπτυξης.

Η διαδικασία επεκτασιμότητας περιέχει τέσσερις φάσεις. Την παρουσίαση της αρχικής πρότασης, την περίοδο επώασης, τη φάση δημιουργίας πρωτοτύπου και τέλος μια ανοικτή παρουσίαση από τον ONF. Κατά τη διάρκεια της αρχικής πρότασης, παρουσιάστηκε η χρήση της περίπτωσης και ένα σετ από τεχνικές λύσεις υλοποίησης. Η φάση της επώασης είναι αυτή κατά την οποία το κείμενο για πρόταση των κανονικών προδιαγραφών εξετάζεται και εγκρίνεται. Στην προτελευταία φάση, ο πρωτότυπος ανοικτός κώδικας της πρότασης, υλοποιείται, εξετάζεται και εγκρίνεται. Τέλος, γίνονται οι αξιολογήσεις του πρωτοκόλλου από άλλους εξωτερικούς οργανισμούς και δημοσιεύεται.

Αφού ιδρύθηκε και ανέλαβε το OpenFlow πρωτόκολλο ο ONF δημοσιεύτηκε η έκδοση 1.1 στις 28 Φεβρουαρίου του 2011. Έπειτα ακολούθησε η έκδοση 1.2 τον Δεκέμβριο του 2012 που ενέκρινε το διοικητικό συμβούλιο του ONF, και τη δημοσίευσε τον Φεβρουάριο του ίδιου έτους. Στη συνέχεια, ακολούθησαν οι εκδόσεις 1.3 και 1.4 τον Απριλίου του 2012 και Αυγούστου του 2013 αντίστοιχα. Σήμερα κυκλοφορεί η έκδοση 1.5.1 που δημοσιεύτηκε τον Μάρτιο του 2015 αν και έχει δημιουργηθεί και η έκδοση 1.6 αλλά είναι προσβάσιμη μόνο για τα μέλη του ONF. Για την καλύτερη χρονική απεικόνιση υπάρχει και το σχήμα Γ'.6.



Σχήμα Γ'.6: Χρονοδιάγραμμα της πορείας ανάπτυξης του OpenFlow.

Μερικές ακόμα σημαντικές στιγμές για την πρόοδο του πρωτοκόλλου OpenFlow είναι οι εξής:

- Τον Μάιο του 2011, οι εταιρίες Marvell και Larch Networks ανακοίνωσαν ότι πλέον υποστηρίζουν λύσεις με OpenFlow βασισμένες στον δικτυακό έλεγχο στοίβας της Marvell και στους επεξεργαστές τσέπης της οικογένειας Presta.
- Το πανεπιστήμιο της Indiana τον Μάιο του 2011 ίδρυσε εργαστήριο λειτουργικότητας για να ελέγξει σε συνεργασία με τον ONF, πόσο καλά δουλεύουν προϊόντα SDN με το

OpenFlow από διάφορους πωλητές.

- Τον Ιούνιο του 2012 η Infoblox δημοσίευσε το LINC. Ένας μεταγωγέας ο οποίος είναι συμβατός με τις εκδόσεις 1.2 και 1.3 του πρωτοκόλλου OpenFlow.
- Το Φεβρουάριο του 2012, η Big Switch Networks κυκλοφόρησε το πρότζεκτ Floodlight [59], το οποίο είναι ένας OpenFlow Controller αναγνωρισμένος από την Apache, που βασίζεται σε ανοικτού κωδικά λογισμικό. Επίσης, τον Νοέμβριο του ίδιου έτους ανακοίνωσε την SDN σουίτα βασιζόμενη στο OpenFlow, η οποία περιέχει εμπορικό Controller και εικονικούς μεταγωγείς.
- Η εταιρία της HP άρχισε να υποστηρίζει το Openflow σε 16 από τους ethernet μεταγωγείς της.
- Τον Απρίλιο του 2012 η Google ανακοίνωσε πως λειτουργεί κάτω από τη διαχείριση του πρωτοκόλλου OpenFlow με σημαντικές βελτιώσεις στην απόδοση.
- Τέλος, τον Ιανουάριο του 2013 η NEC αποκάλυψε ένα εικονικό μεταγωγέα για τον “Microsoft’s Windows Server 2012 Hyper-V hypervisor”, που σχεδιάστηκε για να φέρει τη χρήση του SDN και την οπτικοποίηση των δικτύων με πρωτόκολλο του OpenFlow στο περιβάλλον της Microsoft.

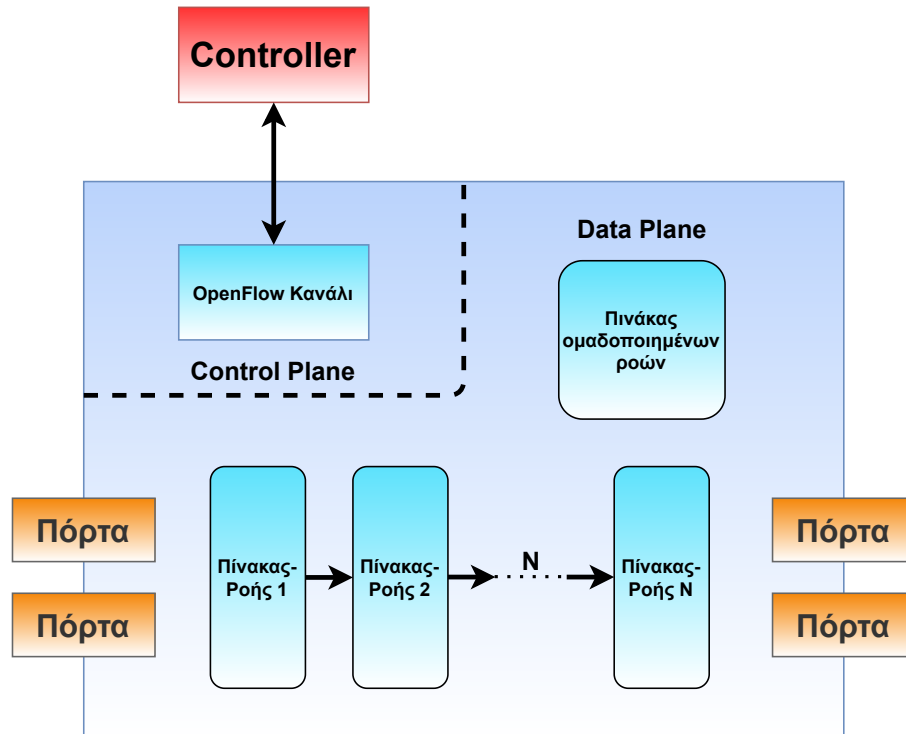
Γ'.3 OpenFlow Μεταγωγείς

Γ'.3.1 Γενικά

Οι OpenFlow μεταγωγείς εκμεταλλεύονται ένα συγκεκριμένο χαρακτηριστικό, που έχουν σχεδόν όλοι οι παραδοσιακοί Ethernet μεταγωγείς και δρομολογητές. Την ύπαρξη του πίνακα-ροής (flow-table), από τον οποίο συλλέγονται στατιστικά. Επομένως, ο διαχειριστής του δικτύου μπορεί να ελέγχει αυτές τις ροές, ώστε να ρυθμίζει τα μονοπάτια που θέλει να ακολουθεί το κάθε πακέτο.

Η διαδρομή που ακολουθούν τα δεδομένα στους OpenFlow μεταγωγείς είναι η είσοδος στον πίνακα-ροής και έπειτα η συσχέτιση της με μια ενέργεια που πρέπει να εκτελεστεί. Πιο συγκεκριμένα, ένας OpenFlow μεταγωγέας αποτελείται από τουλάχιστον τρία μέρη: (1) έναν πίνακα-ροής, με μια ενέργεια να σχετίζεται με κάθε είσοδο ροής, (2) ένα ασφαλές κανάλι, το οποίο συνδέει τον μεταγωγέα με τον Controller του δικτύου, το οποίο θα επιτρέπει την

αποστολή εντολών και πακέτων χρησιμοποιώντας το (3) πρωτόκολλο του OpenFlow, δηλαδή έναν τυπικό τρόπο για να επικοινωνούν μεταγωγείς με Controllers.



Σχήμα Γ'.7: Εσωτερική δομή ενός OpenFlow μεταγωγέα.

Δε θα έπρεπε να παραληφθεί ότι οι OpenFlow μεταγωγείς χωρίζονται σε δυο κατηγορίες. Τους αφιερωμένους (dedicated) OpenFlow μεταγωγείς, οι οποίοι δεν υποστηρίζουν κανονική επεξεργασία επιπέδου OSI 2 και 3. Επιπλέον, τους ενεργοποιημένους με OpenFlow μεταγωγείς, όπως οι παραδοσιακοί γενικής χρήσης εμπορεύσιμοι μεταγωγείς και δρομολογητές, στους οποίους έχουν προστεθεί στη συνέχεια οι διεπαφές και το πρωτόκολλο του OpenFlow.

Γ'.3.2 Αφιερωμένοι OpenFlow Μεταγωγείς

Η λειτουργία ενός αφιερωμένου OpenFlow μεταγωγέα είναι να προωθεί πακέτα στις κατάλληλες πόρτες βάση των εντολών που έχει λάβει από το λογισμικό του Controller. Με αυτήν τη λογική θα μπορούσε να υπάρχει ένας κανόνας ορισμένος από τον Controller, ο οποίος όταν λαμβάνει πακέτο από μια συγκεκριμένη MAC διεύθυνση θα το στέλνει σε μια συγκεκριμένη

πόρτα. Ανάλογα δηλαδή με τις εντολές που έχει στον πίνακα-ροής, κάθε εισαγωγή έχει μια από τις εξής βασικές εντολές:

1. Προώθησε τα πακέτα αυτής της ροής σε μια δοσμένη πόρτα (ή πόρτες). Με αυτήν την εντολή δρομολογούνται τα πακέτα στο δίκτυο, συνήθως χωρίς κάποια καθυστέρηση.
2. Περιλάμβανε και προώθησε το πακέτο αυτής της ροής στον Controller. Με αυτήν την εντολή τα πακέτα στέλνονται με ασφαλή σύνδεση στον Controller, ο οποίος αποφασίζει τι θα πρέπει να κάνει ο μεταγωγέας με αυτού του είδους τη ροή. Αν χρειαστεί μπορεί να τη συμπεριλάβει στον πίνακα-ροής του μεταγωγέα για να μην χρειαστεί ξανά να επεξεργαστεί τέτοιου είδους ροές. Διαφορετικά όλες οι ροές τέτοιου είδους θα αποσταλούν στον Controller.
3. Μην προωθήσεις πουθενά αυτής της ροής το πακέτο. Τέτοιου είδους εντολές χρησιμοποιούνται συνήθως για πρωτόκολλα ασφάλειας, για να εξαλείφουν ύποπτες επικοινωνίες στον δίκτυο.

Γ'.3.3 Ενεργοποιημένοι OpenFlow Μεταγωγείς

Οι παραδοσιακοί μεταγωγείς και δρομολογητές μπορούν να μετατραπούν σε ενεργοποιημένους OpenFlow μεταγωγείς. Αυτό είναι εφικτό με την ενίσχυσή τους με τα χαρακτηριστικά του OpenFlow, δηλαδή με την προσθήκη του πίνακα-ροής, ασφαλές κανάλι επικοινωνίας και το πρωτόκολλο του OpenFlow. Ουσιαστικά ο πίνακας ροής του υλικού θα επαναχρησιμοποιηθεί για της ανάγκες του OpenFlow. Επιπροσθέτως, το ασφαλές κανάλι επικοινωνίας και το πρωτόκολλο του θα υλοποιηθούν στο λογισμικό OpenFlow του μεταγωγέα. Στόχος αυτής της υλοποίησης είναι να μπορούν να γίνουν πειράματα παράλληλα με την παραδοσιακή χρήση του δικτύου, χωρίς να γίνει κάποια παρεμβολή στην ομαλή λειτουργία του δικτύου. Αυτό μπορεί να επιτευχθεί με τον διαχωρισμό της πειραματικής κυκλοφορίας χάρη στην περαιτέρω χρήση του πίνακα ροής, από την παραδοσιακή κυκλοφορία, η οποία γίνεται στο επίπεδο 2 και 3 του OSI με τη χρήση ενός αγωγού (pipeline) του μεταγωγέα.

Στην πράξη αυτό γίνεται με δυο τρόπους:

1. Με την προσθήκη μιας τέταρτης βασικής εντολής, η οποία θα προωθεί τα πακέτα της συγκεκριμένης ροής μέσω ενός αγωγού στην παραδοσιακή κίνηση.

2. Ο άλλος τρόπος είναι να προσδιοριστούν ξεχωριστά ζευγάρια εικονικών δικτύων VLAN's. Ένα για τα πειράματα και το άλλο για την παραδοσιακή κυκλοφορία. Με τον τρόπο μπορούν να παρατηρούνται οι δυνατότητες και οι διαφορές των δυο μεθόδων.

Γ'.4 Πίνακας Ροής

Κάθε OpenFlow μεταγωγέας έχει τουλάχιστον έναν πίνακα-ροής των πακέτων. Κάθε φορά που κάποιος μεταγωγέας δέχεται ένα πακέτο, ελέγχει τον πίνακα-ροής για να μάθει πως να χειριστεί το πακέτο που έλαβε. Αν δεν υπάρχει κάποια σχετική εγγραφή στον πίνακα, τότε ρωτάει τον Controller. Όπως προαναφέρθηκε στο κεφάλαιο Β'.4, αυτές οι εγγραφές μπορούν να έχουν εισαχθεί στον πίνακα προληπτικά, είτε να εισαχθούν, αφού απαντήσει ο Controller την εντολή για τη συγκεκριμένη ροή, δηλαδή αντιδραστικά.

Ο πίνακας-ροής, περιέχει τρία πεδία: (1) μια προσδιοριστική επικεφαλίδα του εκάστοτε πακέτου, (2) την εντολή που πρέπει να εκτελεστεί για τη συγκεκριμένη ροή και (3) τα στατιστικά. Τα οποία εμπεριέχουν πληροφορίες, όπως τον αριθμό των πακέτων, τα bytes της κάθε ροής και ο χρόνος που πέρασε από την τελευταία φορά που εμφανίστηκε αυτή η ροή. Το τελευταίο σκοπεύει στη διαγραφή ροών που δε χρησιμοποιούνται πια.

Στη γενιά μεταγωγών με ονομασία "Τύπος-0", η επικεφαλίδα της κάθε ροής αποτελούταν από 10 πεδία όπως στο Σχήμα Γ'.8. Αναλυτικότερα, περιέχουν πληροφορίες όπως, την πόρτα εισόδου του μεταγωγέα, την εικονική IP (αν υπάρχει). Έπειτα για το φυσικό επίπεδο και το επίπεδο σύνδεσης δεδομένων (data link) υπάρχει Ethernet διεύθυνση αποστολεά-παραλήπτη και τύπος. Στη συνέχεια για το επίπεδο δικτύου, υπάρχει IP διεύθυνση αποστολεά-παραλήπτη και το πρωτοκόλλο (IPv4 ή IPv6). Για το επίπεδο μεταφοράς, μόνο διεύθυνση αποστολεά-παραλήπτη. Αν η ροή είναι τύπου TCP τότε θα πρέπει να συμπεριληφθούν και τα δέκα πεδία, ενώ αν είναι τύπου IP μπορούν να παραληφθούν τα δύο τελευταία. Τέλος, αν υπάρχει ανάγκη να αποσταλεί ένα πακέτο σε όλους, υπάρχει η επιλογή χρήσης του αστερίσκου (*).

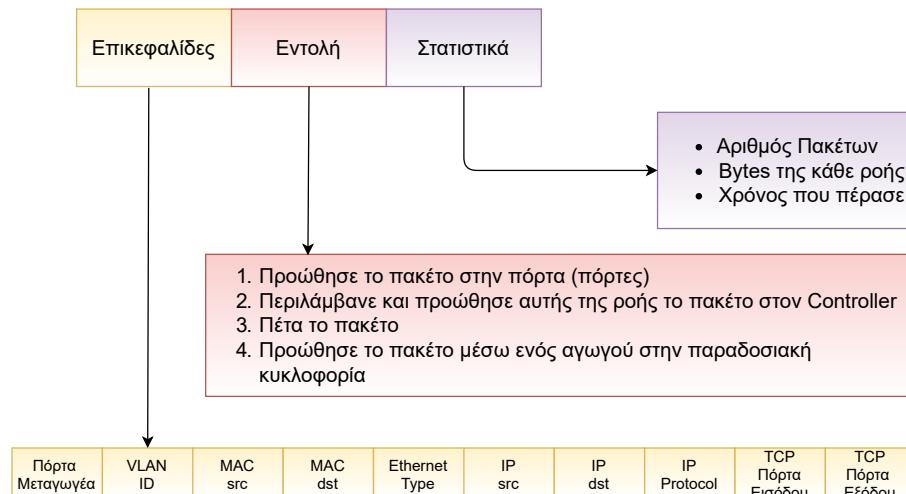
Γ'.5 Σύνδεση και μηχανισμός προώθησης μηνυμάτων

Γ'.5.1 Αποστολή Πακέτων μέσω TCP/IP

Σε αυτό το σημείο αξίζει να γίνει μια σύντομη αναφορά στον τρόπο με τον οποίο λειτουργεί η αποστολή πακέτων μέσω μιας TCP/IP σύνδεσης. Όταν γίνεται η μετάδοση μέσω ενός

Είσοδος Εγγραφής στον Πίνακα-Ροής

OpenFlow μεταγωγέας "Τύπου-0"



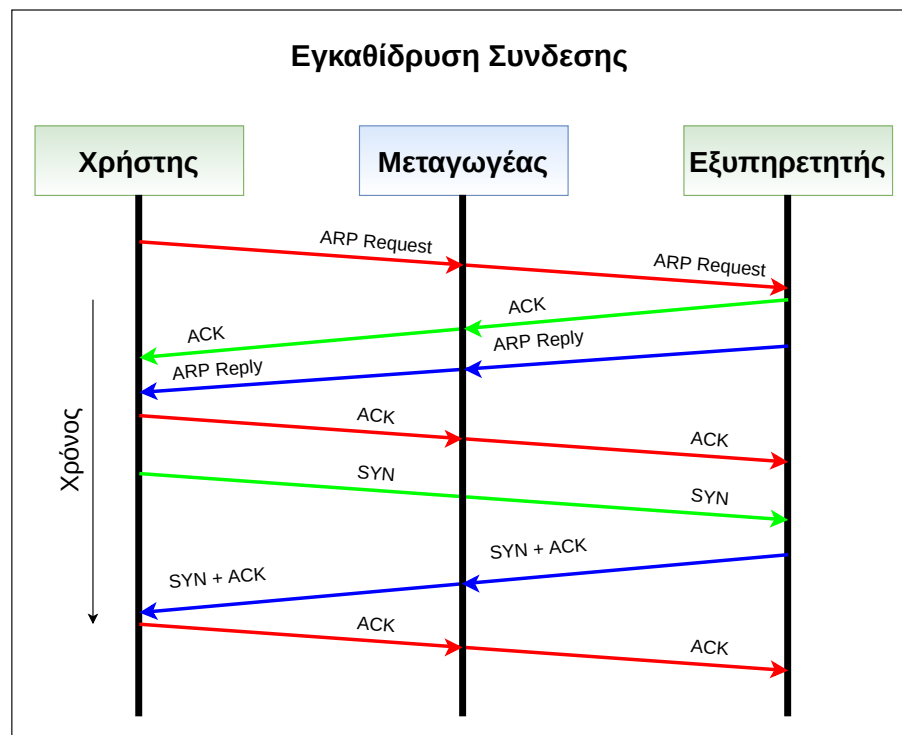
Σχήμα Γ'.8: Η είσοδος μιας εγγραφής στον πίνακα-ροής και οι επικεφαλίδες της.

Ethernet δικτύου και το πακέτο φτάσει στον παραδοσιακό μεταγωγέα, θα ελεγχξει αν η MAC διεύθυνση παραλήπτη υπάρχει στον πίνακα με της διευθύνσεις MAC. Αν υπάρχει, θα γίνει συσχέτιση με την κατάλληλη πόρτα του μεταγωγέα και θα αποστείλει το πακέτο.

Για την επικοινωνία ενός χρήστη με έναν εξυπηρετητή, η εγκαθίδρυση της TCP σύνδεσης απαιτεί μια τριών βημάτων χειραψία και για τον τερματισμό μια τεσσάρων βημάτων.

Για την εγκαθίδρυση, αφού γίνει η συσχέτιση της IP με τη MAC διεύθυνση σύμφωνα με το πρωτόκολλο ARP, ο χρήστης στέλνει μήνυμα συγχρονισμού στον εξυπηρετητή. Αυτός απαντάει με ένα μήνυμα επιβεβαίωσης συγχρονισμού. Τέλος, επιβεβαιώνει ο χρήστης και η εγκαθίδρυση της σύνδεσης έχει ολοκληρωθεί και μπορεί πλέον αν αρχίσει η αποστολή των πακέτων.

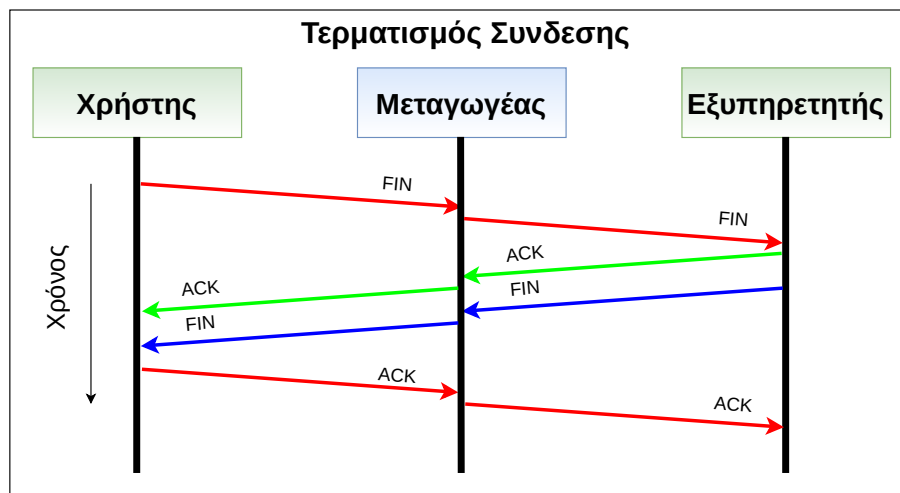
Για τον τερματισμό της σύνδεσης, ο χρήστης αποστέλλει ένα μήνυμα τερματισμού, στο οποίο ο εξυπηρετητής απαντάει με δυο διαδοχικά μηνύματα. Ένα μήνυμα επιβεβαίωσης και όταν τερματίσουν οι διαδικασίες του εξυπηρετητή, τότε στέλνει και ένα μήνυμα επιβεβαίωσης τερματισμού. Τέλος, η σύνδεση λήγει με την τελική επιβεβαίωση του χρήστη προς τον εξυπηρετητή.



Σχήμα Γ'.9: Αναλυτικό παράδειγμα ανταλλαγής μηνυμάτων για την εγκαθίδρυση σύνδεσης.

Γ'.5.2 Αποστολή Πακέτων με OpenFlow

Στην περίπτωση του OpenFlow, η διαδικασία εγκαθίδρυσης σύνδεσης και του τερματισμού της διαφέρουν λίγο από την απλή περίπτωση του TCP. Πιο συγκεκριμένα, πρέπει πρώτα να συνδεθούν όλοι οι μεταγωγείς με τον Controller. Αυτό γίνεται με την παραδοσιακή σύνδεση TCP μέσω ενός ασφαλούς καναλιού, αλλά με την προσθήκη μερικών ακόμα μηνυμάτων. Επομένως, γίνεται η συσχέτιση των IP και των MAC διευθύνσεων με το ARP, και η παραδοσιακή TCP σύνδεση με τη χειραψία τριών βημάτων. Στη συνέχεια ο μεταγωγέας άλλα και ο κεντρικός υπολογιστής ανταλλάσσουν 'Hello' μηνύματα, για να αναγνωρίσουν ποια είναι η μεγαλύτερη έκδοση OpenFlow που χρησιμοποιεί το κάθε ένα, με μηνύματα επιβεβαίωσης ενδιάμεσα. Στη συνέχεια, ο Controller στέλνει ένα 'Features Request' μήνυμα, για να μάθει τις ικανότητες που έχουν οι μεταγωγείς και αυτοί απαντούν με ένα 'Features Response', και σε αυτή την περίπτωση πάλι υπάρχουν μηνύματα επιβεβαίωσης ενδιάμεσα. Τέλος, ο Controller αποστέλλει ένα "flow-mod" μήνυμα για να εισάγει της κατάλληλες εγγραφές στον πίνακα-ροής του μεταγωγέα, και η σύνδεση τελειώνει με την επιβεβαίωση του μεταγωγέα. Υπάρχει αναλυτικό

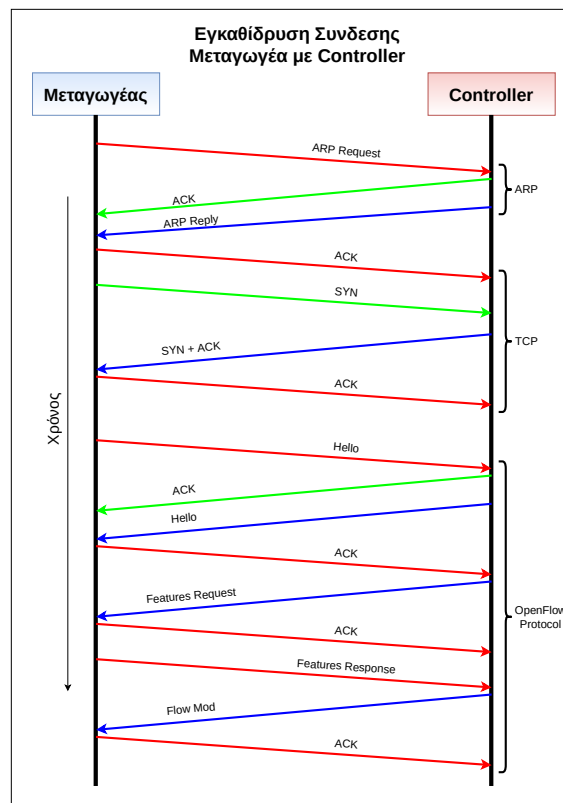


Σχήμα Γ'.10: Αναλυτικό παράδειγμα ανταλλαγής μηνυμάτων για τον τερματισμό σύνδεσης.

παράδειγμα στο Σχήμα Γ'.11. Μετά ο χρήστης ακολουθεί την παραδοσιακή σύνδεση TCP με τον εξυπηρετητή, μόνο που κάθε φορά που λαμβάνει ένας μεταγωγέας μηνύματα τύπου arp, ack, syn, ελέγχει το πίνακα-ροής και αν δεν υπάρχει κάποια εντολή τότε πρέπει να πάρει απάντηση από τον Controller. Αυτό γίνεται με τα μηνύματα 'Packet-In' και 'Packet-Out' που ανταλλάσσουν μεταξύ τους, όπως φαίνεται στο Σχήμα Γ'.12. Ο μεταγωγέας στέλνει ένα μήνυμα 'Packet-In' στον κεντρικό υπολογιστή μαζί με ένα κομμάτι της επικεφαλίδας του μηνύματος που έλαβε ή όλο το πακέτο και περιμένει απάντηση από τον Controller. Στη συνέχεια, αυτός απαντάει με ένα μήνυμα 'Packet-Out' στον μεταγωγέα και που να το προωθήσει αυτός. Στην περίπτωση που ο Controller επιθυμεί να αλλάξει τον πίνακα-ροής του μεταγωγέα, στέλνει ένα 'Flow-Mod' μήνυμα. Έτσι όταν ο μεταγωγέας λάβει ίδιας ροής μήνυμα θα ξέρει πως να το διαχειριστεί χωρίς να στείλει "Packet-In" στον Controller. Για τον τερματισμό της σύνδεσης ακολουθείται το ίδιο μοτίβο με την εγκαθίδρυση σύνδεσης.

Γ'.6 OpenFlow Controllers

Με την έννοια Controller, αναφερόμαστε σε μια κεντρική οντότητα η οποία έχει την πλήρη εικόνα του δικτύου και μπορεί να παίρνει αποφασίσεις για τη βελτίωση του δικτύου. Αποφάσεις, για να προσθέτει, αφαιρεί ροές δεδομένων από τους πίνακες των μεταγωγών, ανάλογα με το εκάστοτε πείραμα που χρειάζεται να υλοποιηθεί στο κάθε δίκτυο. Πιο συγκεκριμένα, θα



Σχήμα Γ'.11: Αναλυτικό παράδειγμα ανταλλαγής μηνυμάτων για την εγκαθίδρυση TCP σύνδεσης του μεταγωγέα με τον Controller.

μπορούσε να είναι ένας προσωπικός υπολογιστής ο οποίος τρέχει ένα πρόγραμμα για να γίνεται η σύνδεση ενός καινούργιου υπολογιστή στο δίκτυο.

Ένα πιο πολύπλοκο παράδειγμα θα ήταν ένας πιο αναπτυγμένος υπολογιστής, ο οποίος θα μπορούσε να παίρνει αποφάσεις για να διαγράφει και να προσθέτει ροές δεδομένων στους πίνακες των μεταγωγών. Αλλά και ακόμα πιο δύσκολο, τοποθετώντας περισσότερους από έναν Controller σε ένα δίκτυο, στους οποίους μπορούν να έχουν πρόσβαση διάφορα άτομα με διαφορετικά δικαιώματα και άδειες ο κάθε ένας. Ως αποτέλεσμα, ανάλογα με τα δικαιώματα που θα έχει ο κάθε χρήστης θα μπορεί να απορρίπτει ή να προωθεί την αποστολή συγκεκριμένων μηνυμάτων από το δίκτυο.

Οι Controllers μπορούν να ταξινομηθούν κυρίως σε δυο κατηγορίες. Αυτοί που είναι ανοικτού κώδικα, και μπορούν οι χρήστες να γράψουν δικό τους κώδικα για διάφορες εφαρμογές που θέλουν να εφαρμόσουν στο δίκτυο τους. Και οι εμπορικοί Controller, που έχουν έτοιμες

γραμμένες εφαρμογές, υπεύθυνες για μια συγκεκριμένη λειτουργία, οι οποίοι είναι κλειστού κώδικα. Οι Controllers, ανοικτού κώδικα χρησιμοποιούνται πιο πολύ για πειραματικούς σκοπούς και υπάρχουν αρκετοί έτοιμοι με ανοικτό κώδικα για αυτόν τον σκοπό, με κύρια διαφορά τους τη γλώσσα που είναι γραμμένος ο κώδικας τους. Μερικοί από τους πιο γνωστούς με βάση τη γλώσσα προγραμματισμού τους είναι:

- Σε C : Trema (επίσης σε Ruby), MUL και ovs-controller.
- Σε C++ : NOX (επίσης σε Python).
- Σε Java : Beacon, Floodlight, Maestro, OpenDaylight, Iris, Jaxon.
- Σε Python : POX και RYU.
- Σε JavaScript : NodeFlow.

Γ'.7 Προδιαγραφές εκδόσεων

Παρακάτω θα γίνει μια σύντομη αναφορά των πιο χρήσιμων προσθηκών σε κάθε έκδοση του πρωτοκόλλου. Οι πληροφορίες είναι διαθέσιμες στην επίσημη ιστοσελίδα του ONF.

Γ'.7.1 OpenFlow 1.1

Με τη δημοσίευση της έκδοσης 1.1 τον Φεβρουάριο του 2011, αυξήθηκε ο αριθμός των επικεφαλίδων από 12 σε 15. Επιπλέον, προστέθηκαν χαρακτηριστικά όπως η χρήση των πολλαπλών πινάκων για της ροές δεδομένων. Προστέθηκε ο πίνακας ομαδοποιημένων ροών, ώστε να εφαρμόζεται μια εντολή σε μια ομάδα από ροές που στοχεύουν στην ίδια λειτουργία και η πλήρης υποστήριξη των εικονικών δικτύων VLANs, καθώς επίσης και των ετικετών MPLS.

Γ'.7.2 OpenFlow 1.2

Τον Δεκέμβριο του 2011 οι επικεφαλίδες αυξήθηκαν στις 36 και προστέθηκαν επιπλέον χαρακτηριστικά, η υποστήριξη του IPv6 πρωτοκόλλου, αλλά και η χρήση πολλαπλών Controller. Μια λειτουργία που βοήθησε πολύ σε θέματα όπου ένας Controller μπορεί να αποτύγχανε να επικοινωνήσει ή όταν το δίκτυο ήταν πολύ μεγάλο για να διαχειρίζεται όλες της ροές μόνο

ένας. Επίσης, έγινε προσθήκη νέων πεδίων για τους κανόνες με το OpenFlow Extensible Match (OXM), βασισμένο σε δομές τύπου type-length-value (TLV).

Γ'.7.3 OpenFlow 1.3

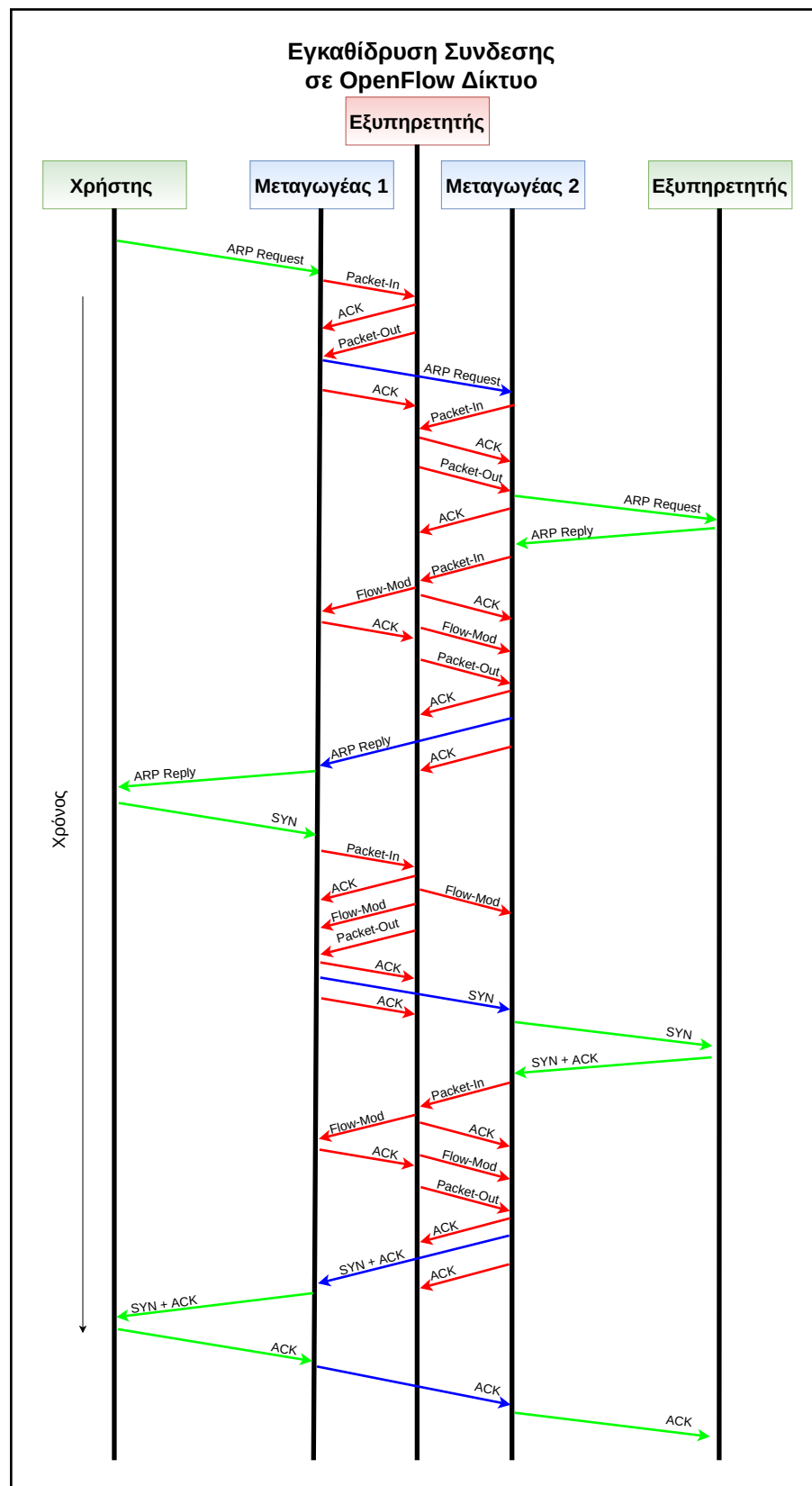
Η έκδοση 1.3 του OpenFlow δημοσιεύτηκε τον Ιούνιο του 2012 με την προσθήκη τεσσάρων ακόμα επικεφαλίδων, φτάνοντας το σύνολο τις 40. Κάποια από τα κύρια χαρακτηριστικά που εισήχθησαν στον πρωτόκολλο, είναι η δυνατότητα για παρακολούθηση των υπηρεσιών και της διαχείρισης με την προσθήκη των μετρικών ή αλλιώς πίνακας μετρητής (meter-table). Επιπλέον, προστέθηκε η αποτυχία ταύτισης με κάποια ροή από τον πίνακα γεγονός που προσδίδει περισσότερη ευκαμψία. Επιπρόσθετα, βελτιώθηκε η επικοινωνία μεταξύ των μεταγωγών, καθώς επιτράπηκαν παράλληλα κανάλια επικοινωνίας με τον Controller.

Γ'.7.4 OpenFlow 1.4

Έπειτα δημοσιεύτηκε η έκδοση 1.4 τον Δεκέμβριο του 2013 και το πρωτόκολλο υποστηρίζει μέχρι και 41 επικεφαλίδες. Πλέον υποστηρίζει οπτικές πόρτες, με την προσθήκη των TLV δομών σε αυτές. Έγινε περαιτέρω ανάπτυξη του συγχρονισμού των μεταγωγών. Επίσης, βελτιώθηκε η υποστήριξη των OXM, αλλά και ο συγχρονισμός των πινάκων που βοήθησε σε προβλήματα όπου οι πίνακες χρειάζονταν να είναι μεγάλοι. Τέλος, ορίστηκε ως προκαθορισμένη πόρτα η 6653.

Γ'.7.5 OpenFlow 1.5

Η τελευταία δημοσιευμένη έκδοση είναι η 1.5 και βγήκε τον Δεκέμβρη του 2014. Προστέθηκε η επεξεργασία δεδομένων στις πόρτες εξόδου. Τέλος, προστέθηκε η αναγνώριση διαφόρων τύπων πακέτων από τους αγωγούς. Πιο συγκεκριμένα, μέχρι τώρα αναγνώριζε μόνο Ethernet πακέτα, άλλα με την τελευταία έκδοση αναγνωρίζει και πακέτα IP ή PPP.



Σχήμα Γ'.12: Αναλυτικό παράδειγμα ανταλλαγής μηνυμάτων για την εγκαθίδρυση TCP σύνδεσης του χρήστη με τον εξυπηρετητή σε OpenFlow δίκτυο.

Κεφάλαιο Δ΄

Πειραματισμοί με OMNeT++

Το παρακάτω κεφάλαιο αναλύει το περιβάλλον προσομοίωσης που εκτελέστηκαν τα πειράματα και ο τρόπος με τον οποίο έγινε η εγκατάσταση του. Επιπλέον παρουσιάζεται η τοπολογία των δικτύων και οι μεταβλητές που απαρτίζουν τα πειράματα.

Δ΄.1 Κύρια ιδέα

Στόχος είναι η σύγκριση απόδοσης δύο εικονικών δικτύων, όπου στο ένα γίνεται χρήση του SDN ενώ το άλλο είναι ένα παραδοσιακό δίκτυο. Σαν απόδοση δικτύου παρουσιάζεται η διαφορά στον συνολικό χρόνο που χρειάζεται το κάθε δίκτυο, για την αποστολή κάποιων αρχείων με βάση κάποια εφαρμογή που τρέχει ο Controller. Ουσιαστικότερα, στόχος είναι να παρουσιάσουν τα πιθανά οφέλη που θα έχει η επιρροή του Controller σε ένα δίκτυο. Αναλυτικότερα, γίνεται παρουσίαση του πρωτοκόλλου του OpenFlow προσομοιωμένο με το προγραμματιστικό περιβάλλον του OMNeT++ σε ένα απλό LAN Ethernet δίκτυο με τη χρήση των πακέτων του INET. Αρχικά, εδραιώνεται μια σύνδεση TCP και σε όλα τα πειράματα εκτός το τελευταίο, αποστέλλονται αρχεία συνολικού μεγέθους 20MiB¹ σε υποδεέστερα πακέτα από έναν αριθμό χρηστών προς έναν δέκτη. Αυτή η σύνδεση πραγματοποιείται σε μία τοπολογία με τρία διαφορετικά μονοπάτια, που οδηγούν από τους χρήστες προς τον δέκτη, συνδεδεμένα με πολλαπλά switches σε σειρά. Η εφαρμογή που τρέχει ο Controller, είναι μία από τις έτοιμες

¹Η μονάδα μέτρησης MiB ή αλλιώς mebibyte είναι ίση με 2^{20} ή 1.048.567 bytes. Ενώ ένα megabyte είναι ίσο με 10^6 ή 1.000.000 bytes. Η διαφορά είναι πολύ μικρή και για αυτό συχνά συγχέονται.

εφαρμογές του OpenFlow στο OMNeT++ και ονομάζεται “LLDPBalancedMinHop”. Η συγκεκριμένη εφαρμογή επεκτείνει την εφαρμογή “LLDPForwarding²”, και στόχος της είναι να ισοροποήσει τον φόρτο του δικτύου. Δηλαδή, εκτός από τη δρομολόγηση των πακέτων μέσω της πιο σύντομης διαδρομής βασισμένη στον αλγόριθμο του Dijkstra, προσπαθεί να ισορροπήσει το φόρτο της τοπολογίας σε περισσότερα από ένα συντομότερα μονοπάτια.

Δ'.2 Περιβάλλον Προσομοίωσης

Δ'.2.1 OMNeT++

Για την προσομοίωση των δικτύων και την εκτέλεση των πειραμάτων έγινε χρήση του προγράμματος Objective Modular Network Testbed in C++ (OMNeT++) [57]. Πιο συγκεκριμένα, είναι ένα δομοστοιχειωτό (modular) πρόγραμμα βασισμένο σε πλαίσιο (framework) και βιβλιοθήκες προσομοίωσης της γλώσσα προγραμματισμού C, κυρίως για προσομοιώσεις δικτυακών περιβαλλόντων. Χρησιμοποιείται για μη εμπορικές χρήσεις, συνήθως σε ακαδημαϊκό επίπεδο, για διδασκαλία και εργασίες. Το πρόγραμμα από μόνο του, δεν περιέχει έτοιμα πακέτα για την υλοποίηση δικτυακών πρωτοκόλλων όπως IP ή HTTP, αλλά υπάρχουν εξωτερικά πακέτα που μπορούν να εγκατασταθούν για να προσθέσουν περισσότερες λειτουργίες. Είναι γραμμένο σε γλώσσα προγραμματισμού C++, υποστηρίζει λειτουργικά συστήματα όπως Windows, Linux αλλά και macOS και η τελευταία σταθερή έκδοση είναι η 5.6.2.

Δ'.2.2 INET

Το INET είναι ένα πακέτο ανοικτού κωδικά, με διάφορες βιβλιοθήκες για να εμπλουτίσει το περιβάλλον προσομοίωσης του OMNeT++. Παρέχει πρωτόκολλα και άλλα μοντέλα για να προσομοιωθούν δίκτυα επικοινωνίας. Ειδικότερα, είναι χρήσιμο για τη δοκιμή νέων πρωτοκόλλων ή κάποιο εξωτικό σεναρίων.

Παρέχει μοντέλα για την προσομοίωση του διαδικτύου, όπως το TCP, UDP και IPv4, ασύρματα και ενσύρματα μοντέλα για το επίπεδο ζεύξης, όπως Ethernet, PPP και IEEE 802.11 και πολλά ακόμα. Επιπλέον, είναι η βάση για την συνεργασία άλλων πακέτων του OMNeT++ όπως το OpenFlow.

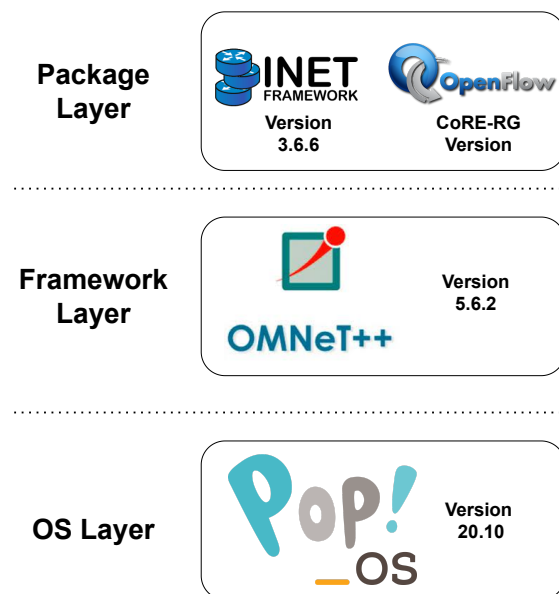
²Στόχος της εφαρμογής 'LLDPForwarding' είναι να χρησιμοποιεί της πληροφορίες που έχει για την τοπολογία και να προωθεί τα μηνύματα από το πιο σύντομο μονοπάτι

Δ'.2.3 OpenFlow

Η σουίτα OpenFlow του OMNeT++ επεκτείνει τις δυνατότητες του OMNeT++ σε συνεργασία με το πακέτο INET, στην υλοποίηση δικτύων με δυνατότητες του πρωτοκόλλου OpenFlow. Πιο συγκεκριμένα παρέχει SDN Controller, μεταγωγείς που υποστηρίζουν OpenFlow επικοινωνία, καθώς επίσης και κάποιες βασικές εφαρμογές για Controller. Αυτές οι εφαρμογές ειδικεύονται σε λειτουργίες, όπως ανακάλυψη τοπολογίας, διαφόρους μηχανισμούς προώθησης των πακέτων και ARP proxy.

Δ'.2.4 Εγκατάσταση

Για να υλοποιηθούν τα πειράματα προσομοίωσης, έγινε εγκατάσταση των εργαλείων που προαναφέρθηκαν, σε λογισμικό Linux με διανομέα το popOS. Πιο συγκεκριμένα, εγκαταστάθηκε το OMNeT++ έκδοση 5.6.2, έπειτα έγινε εισαγωγή των πακέτων INET 3.6.6 και OpenFlow από αποθετήριο κώδικα του Github της CoRE-RG.

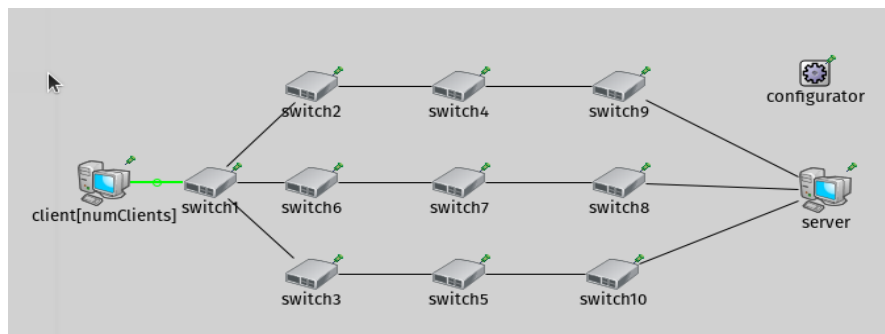


Σχήμα Δ'.13: Στήσιμο του Περιβάλλοντος Προσομοίωσης.

Δ'.3 Τοπολογία και Μεταβλητές

Δ'.3.1 Τοπολογία

Η τοπολογία του παραδοσιακού δικτύου ακολουθεί την εξής δομή. Αρχικά, όλοι οι χρήστες είναι συνδεδεμένοι με τον πρώτο μεταγωγέα, και αυτός συνεχίζει σε τρία μονοπάτια με τρεις μεταγωγείς το κάθε ένα. Τέλος οι τρεις τελευταίοι μεταγωγείς συνδέονται με τον δέκτη, όπως φαίνεται και στο Σχήμα Δ'.14. Αντίθετα, η SDN τοπολογία επεκτείνει την προηγούμενη, προσθέτοντας έναν Controller, ο οποίος επικοινωνεί με όλους τους μεταγωγείς, πιο συγκεκριμένα, αυτό είναι το Control Plane όπως φαίνεται στο Σχήμα Δ'.15. Ο κώδικας απο τα τελεία `ned`³ αρχεία θα προστεθούν στο τέλος, στο Παράρτημα Γ'.3.



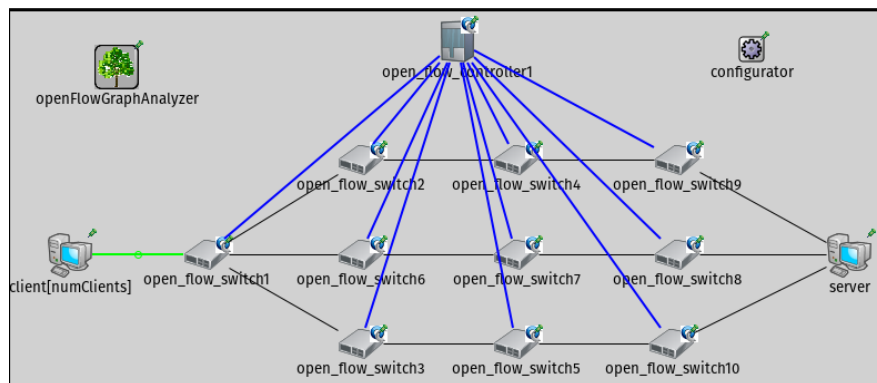
Σχήμα Δ'.14: Τοπολογία παραδοσιακού Δικτύου.

Δ'.3.2 Μεταβλητές

Μερικές από τις μεταβλητές που τροποποιούνται για την ανάλυσή των αποτελεσμάτων είναι:

- Ο αριθμός των χρηστών που είναι συνδεδεμένοι με τον πρώτο μεταγωγέα. Ο κώδικας έχει ρυθμιστεί έτσι ώστε όλοι οι χρήστες να ξεκινήσουν την αποστολή των αρχείων την ίδια χρονική στιγμή. Επομένως, όσο μεγαλύτερος είναι ο αριθμός των χρηστών, τόσα πιο πολλά κομμάτια (segments) του αρχείου αποστέλλονται κάθε χρονική στιγμή, και έτσι υπάρχει μεγαλύτερη επιβάρυνση στο δίκτυο.

³Είναι ένα αρχείο που υποστηρίζει το OMNeT++ για να παρουσιάζει το δίκτυο. Η πλήρης ονομασία τους είναι Network Description File



Σχήμα Δ'.15: Τοπολογία SDN Δικτύου.

- Η χρονική καθυστέρηση που έχει η γραμμή διάδοσης. Δηλαδή, ο χρόνος που απαιτείται για να διανύσει το σήμα την απόσταση μιας ζεύξης. Συνήθως η μονάδα μέτρησης είναι σε mili-seconds (ms).
- Ο ρυθμός με τον οποίο μεταδίδονται τα bits δεδομένων, που από εδώ και πέρα θα αναφέρεται ως data rate.
- Το μέγεθος του αρχείου που θα αποστείλει ο κάθε χρήστης.

Αναλυτικότερα, το data rate μεταβάλλεται από 10 μέχρι 100 Mbps με βήμα 10. Η χρονική καθυστέρηση της διάδοσης μεταβάλλεται από μονάδα μέτρησης uni-seconds (us) σε mili-seconds (ms) και ο αριθμός των χρηστών από 5 μέχρι 20 με βήμα 5. Επιπλέον, το μέγεθος αρχείων μεταβάλλονται από 20Mib μέχρι 100Mib με βήμα 20.

Δ'.4 Παρατηρήσεις / Προβλήματα Προσομοιώσεων

Αν και το OMNeT++ είναι ένα χρήσιμο πρόγραμμα για την προσομοίωση δικτύων, τα έτοιμα πακέτα που υπάρχουν για τη χρήση του OpenFlow, αντιμετωπίζουν κάποια προβλήματα. Για αρχή, αν κάποιος θέλει να παρατηρήσει την απόδοση ενός SDN δικτύου σε περιπτώσεις όπου υπάρχει μεγάλη καθυστέρηση στη γραμμή διάδοσης δεν μπορεί. Αν ξεπεράσει τιμές όπως 25 ms η προσομοίωση σταματάει χωρίς κάποιο ανάλογο μήνυμα.

Επίσης, η ανάθεση των IP διευθύνσεων δε γίνεται αυτόματα. Χρειάζεται να γίνει χειροκίνητη

ανάθεση σε κάθε module, καθώς επίσης και ο διαχωρισμός των IP διευθύνσεων του Control Plane από το Data Plane μέσω ενός αρχείου "XML". Διευκρινίζοντας, θα πρέπει ο χρήστης να προσδιορίσει ένα εύρος τιμών διευθύνσεων IP όπου θα στέλνουν οι μεταγωγείς ως Control Plane και το αντίστοιχο για το Data Plane. Στην συνέχεια, πρέπει να γίνει εισαγωγή ενός configurator στο δίκτυο για να κάνει τις αναθέσεις το διευθύνσεων απο το "XML" αρχείο. Επιπροσθέτως, για να μπορούν να επιλεχθούν τιμές του data rate διάφορες του 10 και του 100 πρέπει να γίνει πάλι χειροκίνητα. Πιο συγκεκριμένα, πρέπει να τροποποιηθεί ο κώδικας του Ethernet ώστε να υποστηρίζει επιπλέον τιμές. Όλες οι αλλαγές που έγιναν στα έτοιμα πακέτα για τις προσομοιώσεις, παρουσιάζονται στο Παράρτημα Γ'.3.

Μια ακόμα παρατήρηση, είναι ότι υπάρχει επιλογή για το πότε να ξεκινήσουν τη σύνδεση μεταξύ τους οι μεταγωγείς και ο Controller. Στα παρακάτω πειράματα, έχει επιλεγθεί να γίνεται η σύνδεση στο 0 δευτερόλεπτο, ώστε μέχρι το 1 να έχει τελειώσει και να ξεκινήσουν ταυτόχρονα οι χρήστες την έναρξη TCP χειραψίας και στα δύο δίκτυα. Επιλέχθηκε να γίνει στο 0 δευτερόλεπτο η σύνδεση, για να μην υπάρχει προβάδισμα στο παραδοσιακό δίκτυο, διότι δε μας αφορά ο χρόνος σύνδεσης των μεταγωγών με το Controller.

Παρά όλες τις δυσκολίες που προβάλλει το πρόγραμμα, είναι πολύ χρήσιμο το γεγονός ότι υπάρχει απεικόνιση του δικτύου. Επίσης, υπάρχει η επιλογή να παρακολουθήσεις ζωντανά τη μεταφορά όλων των πακέτων που ανταλλάσσονται, αλλά και να αυξήσεις την ταχύτητα προσομοίωσης, ώστε τα αποτελέσματα να παραχθούν πιο γρήγορα.

Κεφάλαιο Ε΄

Αποτελέσματα Προσομοιώσεων

Ως συνέχεια του προηγούμενου κεφαλαίου, θα παρουσιάσονται τα διάφορα πειράματα που εκτελέστηκαν και τα αποτελέσματά τους. Επίσης υπάρχουν διαγράμματα παρουσίασης των αποτελεσμάτων και σχολιασμός τους.

Ε΄.1 Πειράματα / Προσομοιώσεις

Ε΄.1.1 Πρώτο Πείραμα (σταθερή η χρονική καθυστέρηση της διάδοσης και το Data Rate)

Στο πρώτο πείραμα παρατηρείται η διαφορά των δύο δικτύων, στον συνολικό χρόνο που χρειάστηκε για να γίνει η αποστολή των αρχείων, καθώς μεταβάλλεται μόνο ο αριθμός των χρηστών κάθε φορά. Το data rate και ο χρόνος καθυστέρησης της διάδοσης της γραμμής παραμένουν σταθερά με τιμές 10Mbps και 1ms αντίστοιχα. Επομένως, θα παρουσιαστεί η απόδοση του κάθε δικτύου ανάλογα με το πόσοι ταυτόχρονοι χρήστες θέλουν να αποστείλουν αρχεία.

Ε΄.1.2 Δεύτερο Πείραμα (Σταθερός ο αριθμός των χρηστών και του Data Rate)

Στο δεύτερο πείραμα παρατηρείται η απόδοση των δύο δικτύων για τις διάφορες τιμές της χρονικής καθυστέρησης της διάδοσης της γραμμής. Επίσης, φάνηκε ότι καλύτερη απόδοση και λειτουργία είχαν τα δίκτυα για 10 χρήστες, οπότε ο αριθμός των χρηστών έμεινε σταθερός, καθώς το data rate έμεινε σταθερό στα 10 Mbps.

Ε'.1.3 Τρίτο Πείραμα (Σταθερός ο αριθμός των χρηστών και της χρονικής καθυστέρησης της διάδοσης)

Με το τρίτο πείραμα, συγκρίνονται τα δύο δίκτυα για τις διάφορες τιμές του data rate. Οι τιμές αυξάνονται ανά 10 από 10 Mbps μέχρι και 100 Mbps, ενώ ο αριθμός των χρηστών μένει σταθερός στο 10 και η χρονική καθυστέρηση της διάδοσης της γραμμής στο 1ms.

Ε'.1.4 Τέταρτο Πείραμα (Διαφορά της απόδοσης σε σύγκριση με το μέγεθος του αρχείου)

Σε αυτό το πείραμα, συγκρίνεται η διαφορά απόδοσης μεταξύ των δύο δικτύων για τις διάφορες τιμές μεγέθους του αρχείου που θα αποστείλει ο κάθε χρήστης. Η χρονική καθυστέρηση της διάδοσης της γραμμής έμεινε σε όλες τις περιπτώσεις σταθερή στο 1us. Αντιθέτως, σε κάθε υποπείραμα, εκτός του μεγέθους του αρχείου, μεταβαλλόταν το data rate από 10 σε 20 Mbps και ο αριθμός των χρηστών που βρίσκονται στο δίκτυο, από 10 σε 20 χρήστες ταυτόχρονα.

Ε'.1.5 Πέμπτο Πείραμα (Απόδοση της εφαρμογής LLDPBalanced-Hop)

Στόχος του πέμπτου πειράματος είναι να παρουσιαστεί η απόδοση της εφαρμογής “LLDP-BalancedMinHop”, που χρησιμοποιεί ο Controller στο πακέτο του OpenFlow για το OM-NeT++. Πιο συγκεκριμένα, παρατηρείται η χρησιμοποίηση (utilization) της κάθε διαδρομής σε μια τοπολογία η οποία έχει μεγάλο φόρτο. Συνεπώς, γίνεται σύγκριση της αξιοποίησης των ελεύθερων μονοπατιών σε περιπτώσεις, όπου υπάρχει καθυστέρηση στην αποστολή των πακέτων, μέσω ενός συγκεκριμένου μονοπατιού. Εμφανίζονται οι διαφορές που έχουν τα δύο δίκτυα, για κάθε μία από τις τρεις διαδρομές, αλλά και από κόμβο σε κόμβο.

Ε'.2 Αποτελέσματα

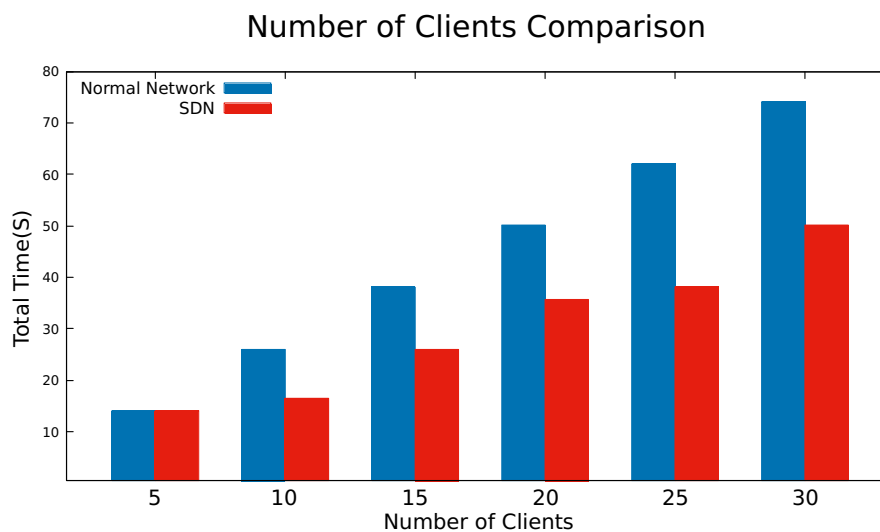
Ε'.2.1 Συλλογή αποτελεσμάτων

Για να γίνει η συλλογή των αποτελεσμάτων προσθέτονται κάποιες μεταβλητές στο αρχείο τελεία ini του κάθε δικτύου. Επιπλέον, στην περίπτωση του SDN προστίθεται ένα module ονόματι ‘OpenFlowGraphAnalyzer’, το οποίο κρατάει κάποιες έξτρα μεταβλητές σχετικές με

τον Controller. Στη συνέχεια, τα δεδομένα που παράγει το OMNeT++ είναι σε διανυσματική (vectors), και βαθμωτή (scalar) μορφή, από τα οποία έγινε εξόρυξη των πιο χρήσιμων μεταβλητών σε ένα αρχείο τελεία csv. Τέλος η ανάλυση και η δημιουργία γραφημάτων έγινε με τη χρήση του Gnuplot.

Ε'.2.2 Αποτελέσματα πρώτου πειράματος

Μετά από 12 προσομοιώσεις, παρατηρείται ότι και στις 6 τιμές του πλήθους των χρηστών, το δίκτυο που είναι βασισμένο σε SDN, ολοκληρώνει την αποστολή των πακέτων, σε λιγότερο χρόνο εκτός της πρώτης περίπτωσης. Αυτό συμβαίνει διότι ο αριθμός των χρηστών δεν είναι αρκετός για να δημιουργήσει αρκετό φόρτο στο δίκτυο, και να πυροδοτήσει την έναρξη της εφαρμογής του LLDPBalancedMinHop. Επομένως, αν δημιουργηθεί αρκετός φόρτος στο δίκτυο, η χρήση του SDN, ανεξάρτητα από το πλήθος των χρηστών, είναι πιο αποτελεσματική από το παραδοσιακό δίκτυο.

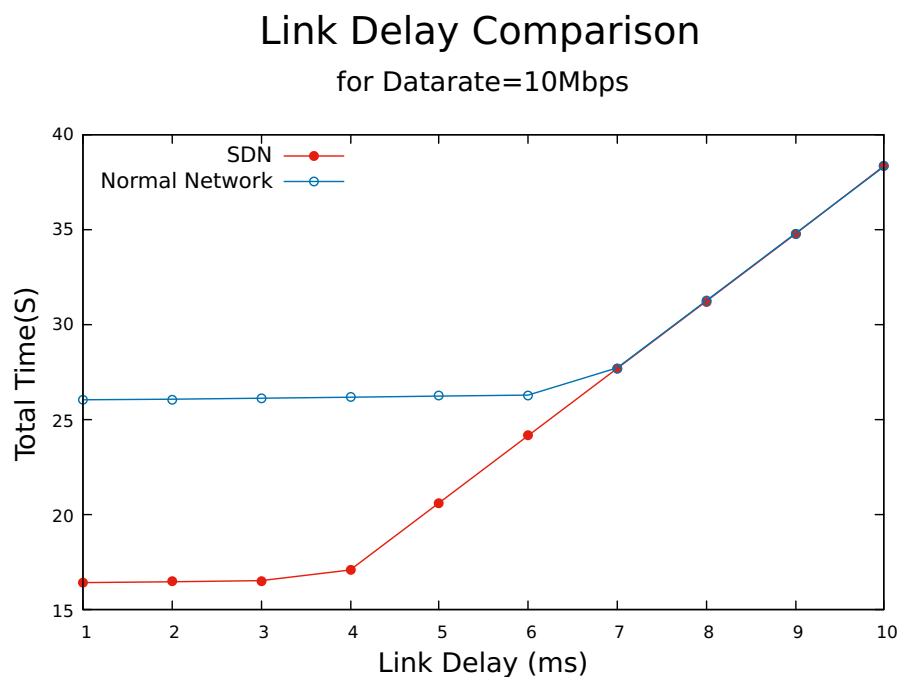


Σχήμα Ε'.16: Σύγκριση των δικτύων με βάση το πλήθος των χρηστών.

Ε'.2.3 Αποτελέσματα δεύτερου πειράματος

Σε αυτό το πείραμα έγιναν στο σύνολο 10 προσομοιώσεις. Οι μισές για data rate 10Mbps και οι άλλες για 100Mbps. Για την τιμή των 10 Mbps, παρατηρείται ότι από την τιμή 1ms

μέχρι τα 4ms το SDN δίκτυο αποδίδει καλύτερα με εμφανή αποτελέσματα, ενώ μέχρι τα 7ms αυτή η διαφορά συγχλίνει, και μετά τα 7ms εκτελούν την αποστολή του αρχείου στον ίδιο χρόνο. Αυτό συμβαίνει διότι όταν το data rate παραμένει σταθερό μέχρι τα 10Mbps και η χρονική καθυστέρηση διάδοσης της γραμμής αυξηθεί πάνω από ένα όριο, (στη συγκεκριμένη περίπτωση τα 7ms), ακόμα και αν λειτουργεί η εφαρμογή για να μεταφέρονται τα πακέτα και από άλλα μονοπάτια, λόγω της υπερβολικής καθυστέρησης εκτελείται ο ίδιος χρόνος για την αποστολή του αρχείου. Για data rate 100 Mbps η διαφορά του συνολικού χρόνου εκτέλεσης των δύο δικτύων είναι πολύ μικρή, οπότε παραλείφθηκε από τα διαγράμματα.

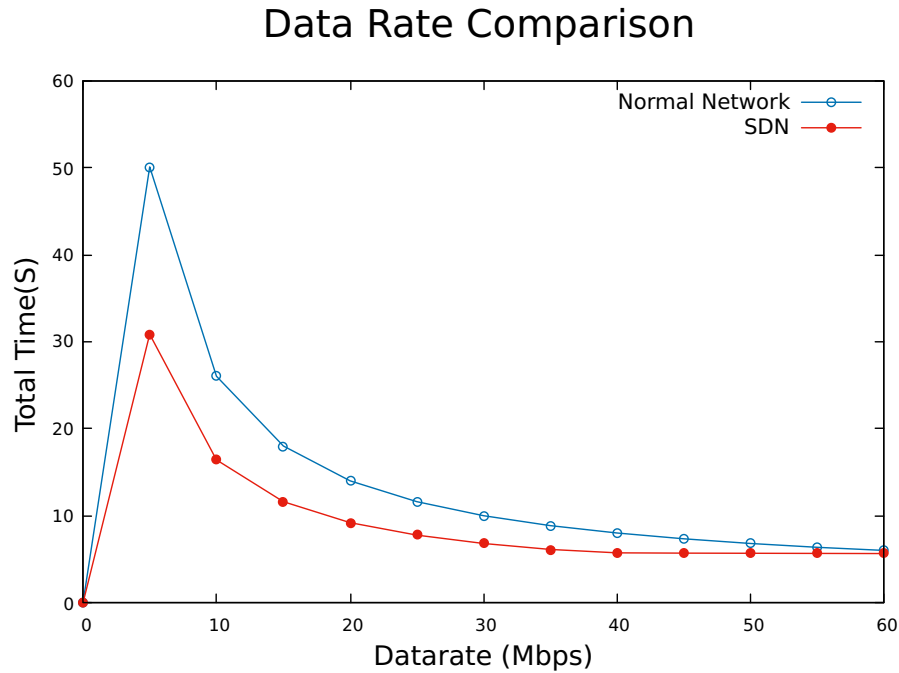


Σχήμα Ε'.17: Σύγκριση των δικτύων με βάση τη χρονική καθυστέρηση για 10Mbps.

Ε'.2.4 Αποτελέσματα τρίτου πειράματος

Σε αυτή την περίπτωση παρατηρούμε για αριθμό των χρηστών ίσο με 10 και χρονική καθυστέρηση της γραμμής ίση με 1μs, τις διαφορές των δύο δικτύων καθώς το data rate αυξάνεται. Πραγματοποιήθηκαν 24 προσομοιώσεις για τιμές από 5 μέχρι 60 Mbps ανά 5. Από αυτό το πείραμα είναι εμφανές ότι με τη χρήση του SDN ο συνολικός χρόνος είναι γενικά καλύτερος, καθώς επίσης ότι για μικρότερες τιμές του ρυθμού μετάδοσης, η διαφορά είναι μεγαλύτερη σε

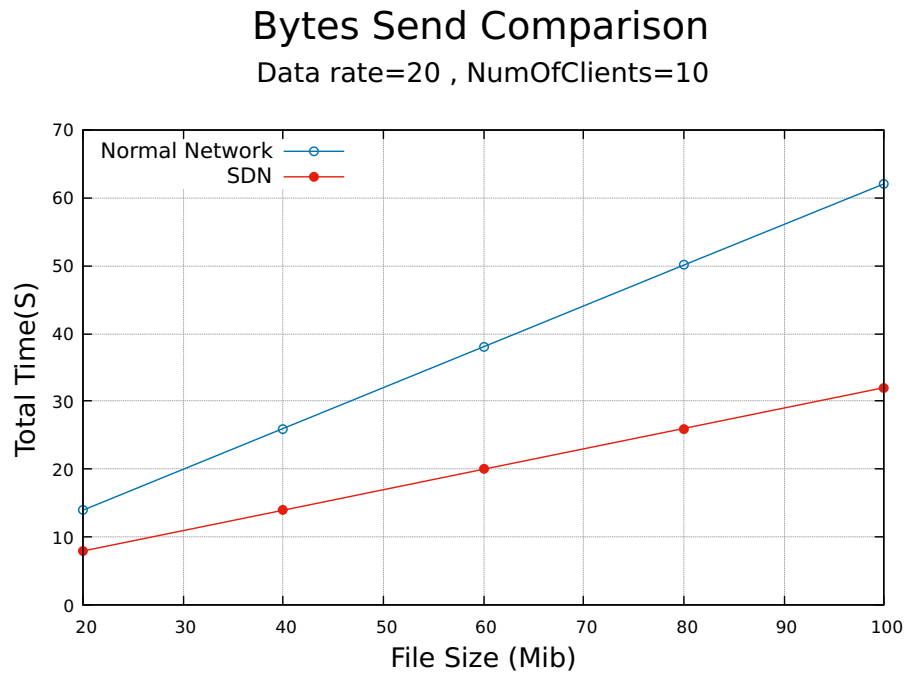
σχέση με όταν έχουμε μεγαλύτερες τιμές. Αυτό έγκειται στο γεγονός, ότι όταν το δίκτυο έχει καλύτερο data rate, τα πακέτα αποστέλλονται σχεδόν στον ίδιο χρόνο και από τη μια μόνο διαδρομή.



Σχήμα Ε'.18: Σύγκριση των δικτύων με βάση το data rate.

Ε'.2.5 Αποτελέσματα τέταρτου πειράματος

Στο τέταρτο πείραμα τα αποτελέσματα είναι αρκετά πιο ευδιάκριτα. Αρχικά, εκτελέστηκαν 4 υποπειράματα, και συνεπώς 8 προσομοιώσεις. Δύο προσομοιώσεις για σταθερό το data rate στα 20 Mbps και μεταβαλλόταν μόνο αριθμός των ταυτόχρονων χρηστών από 10 σε 20. Ακολούθως, στα επόμενα δύο μειώθηκε το data rate στα 10 Mbps και για τα δύο, αλλά μεταβαλλόταν πάλι ο αριθμός των ταυτόχρονων χρηστών από 10 σε 20. Σύμφωνα με τα διαγράμματα, παρατηρείται ότι και στα τέσσερα, η χρήση του SDN αποδίδει καλύτερα με σχεδόν το μισό χρόνο αποστολής κάθε φορά. Επιπλέον, συμπεραίνεται ότι όσο αυξάνεται το μέγεθος του αρχείου αποστολής, τόσο μεγαλύτερη είναι η διαφορά στην απόδοση μεταξύ των δύο δικτύων.



Σχήμα Ε'.19: Σύγκριση των δικτύων με βάση το μέγεθος του αρχείου για data rate 20Mbps και αριθμός χρηστών 10.

Ε'.2.6 Αποτελέσματα πέμπτου πειράματος

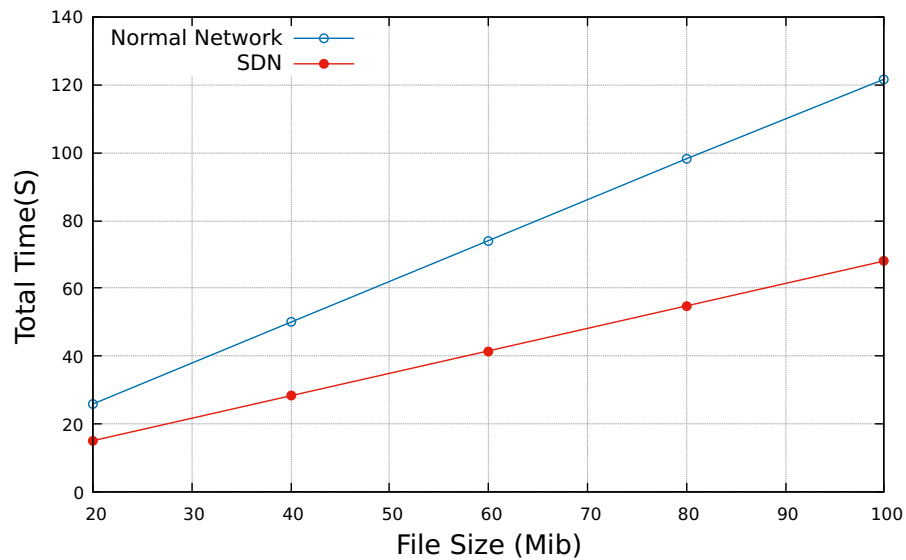
Στο τελευταίο πείραμα βλέπουμε για μια μόνο προσομοίωση, και για τα δύο δίκτυα, την χρήση που έγινε για το κάθε μονοπάτι. Το πείραμα έγινε για τις τιμές όπου ο αριθμός των χρηστών είναι 5, η χρονική καθυστέρηση 1ms και το data rate 10Mbps. Στο πρώτο διάγραμμα παρουσιάζεται πόσο τοις εκατό χρήση έγινε σε κάθε σύνδεση μέσα στις δύο τοπολογίες, δηλαδή για κάθε κόμβο. Ενώ στο δεύτερο, συγκρίνονται ανά τριάδες, η χρήση των τριών μονοπατιών σε κάθε δίκτυο, για τις διάφορες τιμές του πλήθους των χρηστών.

Ε'.2.7 Σχολιασμός των αποτελεσμάτων

Λαμβάνοντας υπόψη τα παραπάνω πειράματα που υλοποιήθηκαν, η χρήση μιας εφαρμογής όπως το “LLDPBalancedMinHop” μείωσαν τον συνολικό χρόνο αποστολής κάποιων αρχείων. Αυτό έγινε ισορροπώντας το φόρτο του δικτύου σε παραπάνω από ένα μονοπάτι, όταν υπήρχε η κατάλληλη συμφόρηση, ώστε να ενεργοποιηθεί ο αλγόριθμος του “LLDPBalancedMinHop”.

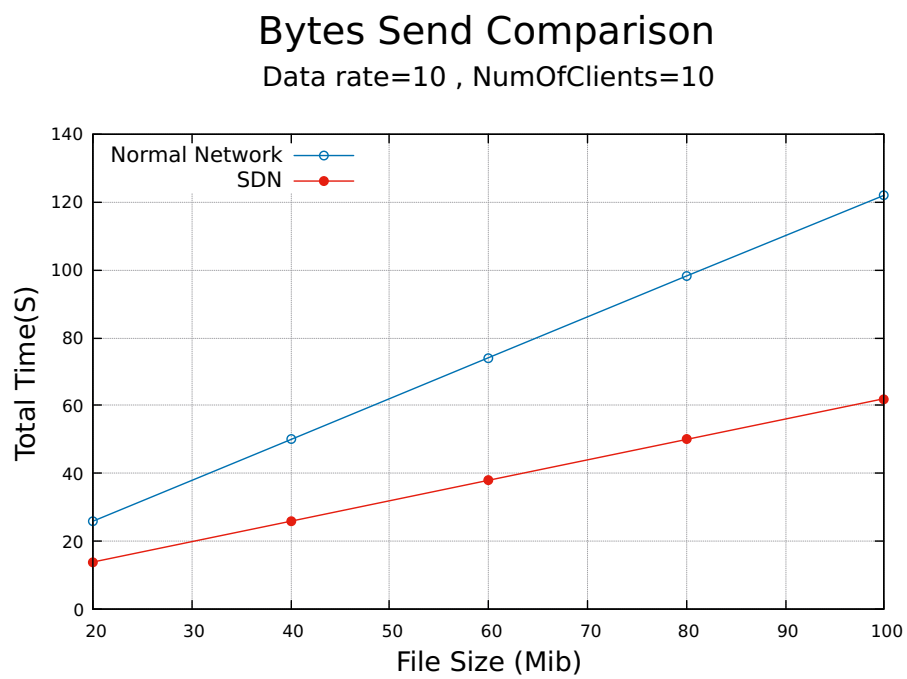
Bytes Send Comparison

Data rate=20 , NumOfClients=20

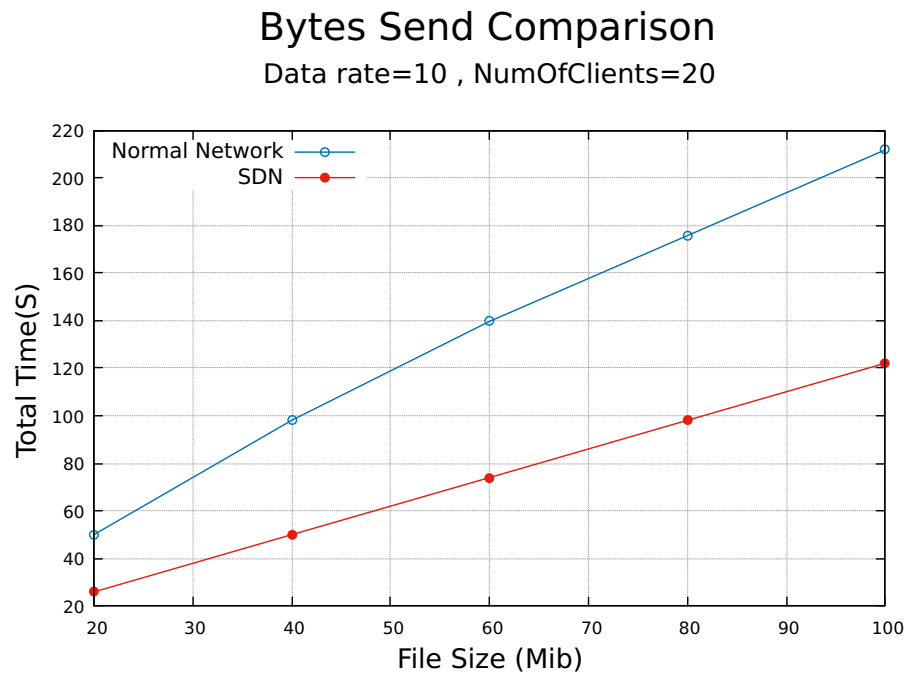


Σχήμα Ε'.20: Σύγκριση των δικτύων με βάση το μέγεθος του αρχείου για data rate 20Mbps και αριθμός χρηστών 20.

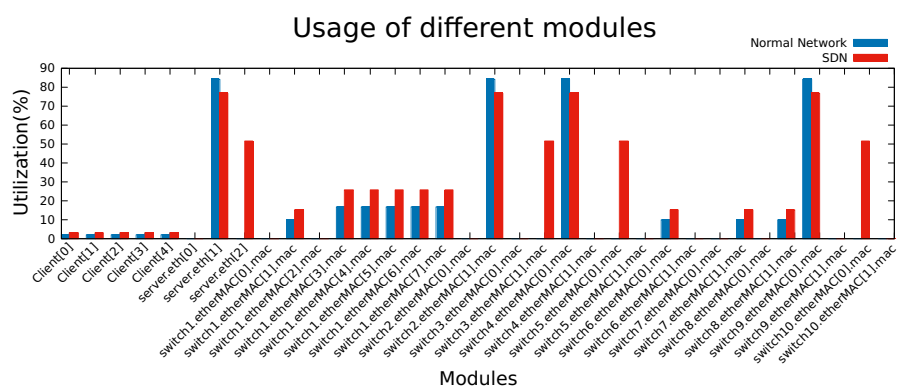
Επομένως, το δίκτυο που κάνει χρήση του SDN εκτελεί την αποστολή των πακέτων σε λιγότερο χρόνο ανεξαρτήτως του αριθμού των χρηστών, εκτός και αν αυτός είναι ελάχιστος και δεν προκαλεί τον κατάλληλο φόρτο στο δίκτυο, ώστε να ενεργοποιηθεί η εφαρμογή που τρέχει στον Controller. Επιπροσθέτως είναι πιο γρήγορο από το παραδοσιακό για αρκετές τιμές του data rate, εκτός την περίπτωση όπου τα πακέτα μεταδίδονται τόσο γρήγορα, ώστε στη συγκεκριμένη τοπολογία να μην επηρεάζει τον συνολικό χρόνο εκτέλεσης. Παρομοίως δουλεύει και με την καθυστέρηση της διάδοσης, δηλαδή αν αυξηθεί πολύ, τότε τα πακέτα κάνουν τον ίδιο χρόνο είτε δρομολογούνται όλα από ένα μονοπάτι, είτε από πολλά. Αυτό συμβαίνει διότι με τον τρόπο με τον οποίο είναι υλοποιημένη η αποστολή TCP αρχείων με το INET στέλνει κάθε φορά μέχρι 14 πακέτα και μετά περιμένει τις επιβεβαιώσεις ACK τους μέχρι να συνεχίσει την αποστολή. Συνεπώς, ακόμα και αν τα στείλει από διαφορετικά μονοπάτια θα φτάσουν στον ίδιο χρόνο στον εξυπηρετητή και αυτός με τη σειρά του θα στείλει της επιβεβαιώσεις στον ίδιο χρόνο που θα τις έστειλε αν λάμβανε τα πακέτα από ένα μονοπάτι.



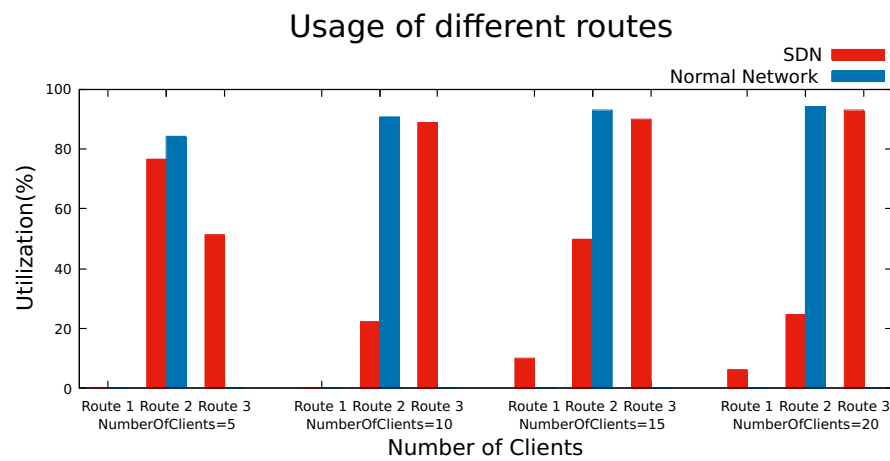
Σχήμα Ε'.21: Σύγκριση των δικτύων με βάση το μέγεθος του αρχείο για data rate 10Mbps και αριθμός χρηστών 10.



Σχήμα Ε'.22: Σύγκριση των δικτύων με βάση το μέγεθος του αρχείου για data rate 10Mbps και αριθμός χρηστών 20.



Σχήμα Ε'.23: Χρήση όλων των καναλιών σε μια προσομοίωση.



Σχήμα Ε'.24: Χρήση των τριών μονοπατιών για τις διάφορες τιμές του πλήθους των χρηστών.

Κεφάλαιο Γ΄

Συμπεράσματα

Με το τέλος της παρούσας πτυχιακής εργασίας, συνοψίζεται η χρησιμότητα του SDN, η δουλειά του OpenFlow σε αυτό και τα συμπεράσματα που υπάρχουν για την χρήση τους στο περιβάλλον του OMNeT++.

Γ΄.1 Σύνοψη

Ξεκίνησε αυτή η εργασία, δηλώνοντας ότι οι διαχειριστές των δικτύων χρησιμοποιώντας την τεχνολογία του SDN θα μπορέσουν να βελτιώσουν την απόδοση των δικτύων τους. Για να επιτευχθεί αυτό, θα πρέπει να αναβαθμιστεί η παλαιά τους δομή ώστε να μπορούν να ανταπεξέλθουν στις ραγδαίες απαιτήσεις των δικτυακών πόρων. Αναλύοντας, με το SDN, δηλαδή μια τεχνολογία που διαχωρίζει το κανάλι μετάδοσης των δεδομένων με το κανάλι λήψης αποφάσεων δρομολόγησης, θα τους βοηθήσει να εκμεταλλευτούν καλύτερα την ισχύ του δικτύου τους. Δηλαδή, με την προσθήκη ενός ή περισσότερων Controller για να γίνεται η λήψη αυτών των αποφάσεων και την αναβάθμιση των συσκευών μετάδοσης των δεδομένων, όπως ένας μεταγωγέας, σε κάποιες αντίστοιχες που να υποστηρίζουν κάποιο πρωτόκολλο επικοινωνίας για SDN.

Για να λειτουργήσει ένα SDN δίκτυο πρέπει όλες οι συσκευές που το απαρτίζουν να ακολουθούν τους ίδιους κανόνες και τις ίδιες διαδικασίες ανταλλαγής μηνυμάτων. Αυτό πραγματοποιείται με τη χρήση του πρωτοκόλλου OpenFlow, το οποίο έχει δημιουργηθεί και συντηρείται από τον ONF. Σύμφωνα με αυτό το πρωτόκολλο, υπάρχουν κάποια συνήθη μηνύματα που ανταλλάζουν μεταξύ τους ο Controller και οι μεταγωγείς του δικτύου. Μηνύματα όπως το “Packet-In” για

να ρωτήσει ο μεταγωγέας τον Controller και το “Packet-Out” για να στείλει ο Controller την ανάλογη εντολή. Με αυτόν τον τρόπο επιτρέπεται η ανανέωση των πινάκων-ροής των μεταγωγών ώστε να προσαρμόζονται κατάλληλα στις απαιτήσεις κάθε περίπτωσης.

Η εφαρμογή του SDN με το πρωτόκολλο του OpenFlow σε ένα δίκτυο από μόνα τους, δεν αυξάνουν κάπως την απόδοση. Για να υπάρξει κάποια απόδοση πρέπει να εφαρμοστούν οι κατάλληλες εφαρμογές, που θα βελτίωναν το εκάστοτε δίκτυο. Αν κάποιο δίκτυο δέχεται συχνά επιθέσεις τύπου DDoS, μπορεί να προγραμματιστεί έτσι ώστε να απορρίπτει ύποπτα πακέτα. Ένα άλλο παράδειγμα θα ήταν ο υπερβολικός φόρτος ενός δικτύου, διότι όλα τα πακέτα δρομολογούνται από το ίδιο μονοπάτι. Ενός τέτοιου είδους πρόβλημα, παρουσιάστηκε σε αυτή την πτυχιακή.

Για να γίνει η προσομοίωση ενός δικτύου που αντιμετωπίζει πρόβλημα με την ισορρόπηση του φόρτου, χρησιμοποιήθηκε το πρόγραμμα προσομοίωσης δικτύων OMNeT++. Στη συνέχεια, δημιουργήθηκε το ίδιο δίκτυο με τη χρήση του OpenFlow και μια εφαρμογή που τρέχει στον Controller με στόχο την ισορρόπηση φόρτου των διαφόρων μονοπατιών. Τέλος, έγινε η σύγκριση αυτών των δύο δικτύων ως προς τον χρόνο που χρειάστηκε το κάθε ένα για την αποστολή κάποιων αρχείων.

Γ'.2 Συμπεράσματα

Είναι πιστευτό, ότι με την παρούσα μελέτη, ενισχύεται η ακαδημαϊκή κατανόηση των παραγόντων που απαρτίζουν το SDN. Από τις προσομοιώσεις και τα πειράματα που έγιναν, μπορεί κανείς να συμπεράνει ότι η επιρροή ενός Controller σε μια τοπολογία μπορεί να φανεί χρήσιμη, ανάλογα με τις ανάγκες που έχει το κάθε δίκτυο. Πιο συγκεκριμένα, υπάρχουν πολλά σενάρια όπου το δίκτυο από μόνο του αδυνατεί να προσαρμοστεί στις απαιτήσεις των χρηστών και χρειάζεται η αναβάθμιση του σε SDN.

Στο συγκεκριμένο παράδειγμα που παρουσιάστηκε, η απόδοση του δικτύου δεν ήταν πολύ καλή, αρχικά λόγω των πολλών χρηστών και έπειτα λόγω της ταχύτητας του. Το δεύτερο, οφείλεται στην καθυστέρηση της διάδοσης της γραμμής ή στο χαμηλό data rate. Αυτό είχε ως αποτέλεσμα, στην ύπαρξη υπερβολικά μεγάλου φόρτου σε ένα μονοπάτι. Η αιτία είναι, ότι ο μεταγωγέας επέλεγε την ίδια δρομολόγηση των πακέτων, ανεξάρτητα του φόρτου. Επομένως, μια εφαρμογή ισορρόπησης φόρτου όπως το ‘LLDPBalancedMinHop’ μπορεί να κάνει τη δρομολόγηση των πακέτων καλύτερη όπως παρατηρήθηκε από τα διαγράμματα των πειραμάτων.

Γ'.3 Συνεισφορά

Η παρούσα πτυχιακή εργασία, οδηγεί στα εμφανή οφέλη δρομολόγησης που μπορεί να επιφέρει η τοποθέτηση ενός Controller σε ένα δίκτυο. Αρχικά, φανερώνεται η χρησιμότητα των SDN δικτύων όταν ο διαχειριστής τους θέλει να τροποποιήσει την δρομολόγηση των πακέτων με στόχο την βελτίωση τους. Έπειτα παρουσιάστηκαν με άθρονα σχήματα οι προδιαγραφές και ο τρόπος χρήσης του πρωτοκόλλου OpenFlow. Τέλος, παρουσιάστηκε πως μπορούν αυτές οι τεχνολογίες να υλοποιηθούν για πειραματικούς σκοπούς στο OMNeT++. Ειδικότερα, έγινε επίδειξη της απόδοσης ενός δικτύου με εφαρμογές ισοροπικής φόρτου στο OMNeT++ με γραφήματα και αποτελέσματα.

Βιβλιογραφία

- [1]
- [2] Reaz Ahmed and Raouf Boutaba. Design considerations for managing wide area software defined networks. *IEEE Communications Magazine*, 52(7):116–123, 2014.
- [3] Ian F Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, and Wu Chou. A roadmap for traffic engineering in sdn-openflow networks. *Computer Networks*, 71:1–30, 2014.
- [4] Ehab Al-Shaer and Saeed Al-Haj. Flowchecker: Configuration analysis and verification of federated openflow infrastructures. In *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration*, pages 37–44, 2010.
- [5] Mohammed A Alqarni. Benefits of sdn for big data applications. In *2017 14th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT)*, pages 74–77. IEEE, 2017.
- [6] Ameen Banjar, Pakawat Pupatwibul, Robin Braun, and Bruce Moulton. Analysing the performance of the openflow standard for software-defined networking using the omnet++ network simulator. In *2014 Asia-Pacific Conference on Computer Aided System Engineering (APCASE)*, pages 31–37. IEEE, 2014.
- [7] Kamal Benzekki, Abdeslam El Fergougui, and Abdelbaki Elbelrhiti Elalaoui. Software-defined networking (sdn): a survey. *Security and communication networks*, 9(18):5803–5833, 2016.
- [8] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O’Connor, Pavlin Radoslavov, William Snow, et al. Onos: towards an open, distributed sdn os. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 1–6, 2014.

- [9] Rodrigo Braga, Edjard Mota, and Alexandre Passito. Lightweight ddos flooding attack detection using nox/openflow. In *IEEE Local Computer Network Conference*, pages 408–415. IEEE, 2010.
- [10] RM Braun, P Pupatwibul, and A Banjar. Performance evaluation of tcp/ip vs. openflow in inet framework using omnet++, and implementation of intelligent computational model to provide autonomous behavior. In *The Asian Conference on Technology, Information & Society 2014 (ACTIS 2014)*. IAFOR International Academic Forum, 2014.
- [11] Jiaxin Cao, Chuanxiong Guo, Guohan Lu, Yongqiang Xiong, Yixin Zheng, Yongguang Zhang, Yibo Zhu, Chen Chen, and Ye Tian. Datacast: A scalable and efficient reliable group data delivery service for data centers. *IEEE Journal on Selected Areas in Communications*, 31(12):2632–2645, 2013.
- [12] Jose Costa-Requena, Jesús Llorente Santos, Vicent Ferrer Guasch, Kimmo Ahokas, Gopika Premsankar, Sakari Luukkainen, Oscar López Pérez, Mikel Uriarte Itzazelaia, Ijaz Ahmad, Madhusanka Liyanage, Mika Ylianttila, and Edgardo Montes de Oca. Sdn and nfv integration in generalized mobile network architecture. In *2015 European Conference on Networks and Communications (EuCNC)*, pages 154–158, 2015.
- [13] Alisa Devlic, Wolfgang John, and Pontus Sköldström. Carrier-grade network management extensions to the sdn framework. In *Proc. 8th Swedish National Computer Networking Wksp. SNCNW 2012*, 2012.
- [14] Nick Feamster. Outsourcing home network security. In *Proceedings of the 2010 ACM SIGCOMM workshop on Home networks*, pages 37–42, 2010.
- [15] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to sdn: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review*, 44(2):87–98, 2014.
- [16] Kostas Giotis, Christos Argyropoulos, Georgios Androulidakis, Dimitrios Kalogeras, and Vasilis Maglaris. Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. *Computer Networks*, 62:122–136, 2014.

- [17] Nicholas Gray, Thomas Zinner, Steffen Gebert, and Phuoc Tran-Gia. Simulation framework for distributed sdn-controller architectures in omnet++. In *International Conference on Mobile Networks and Management*, pages 3–18. Springer, 2016.
- [18] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKown, and Scott Shenker. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, 2008.
- [19] Soheil Hassas Yeganeh and Yashar Ganjali. Kandoo: a framework for efficient and scalable offloading of control applications. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 19–24, 2012.
- [20] Brian Hayes. Cloud computing, 2008.
- [21] David Hock, Matthias Hartmann, Steffen Gebert, Michael Jarschel, Thomas Zinner, and Phuoc Tran-Gia. Pareto-optimal resilient controller placement in sdn-based core networks. In *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*, pages 1–9. IEEE, 2013.
- [22] Yan-Nan Hu, Wen-Dong Wang, Xiang-Yang Gong, Xi-Rong Que, and Shi-Duan Cheng. On the placement of controllers in software-defined networks. *The Journal of China Universities of Posts and Telecommunications*, 19:92–171, 2012.
- [23] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. Openflow random host mutation: transparent moving target defense using software defined networking. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 127–132, 2012.
- [24] Raj Jain and Subharthi Paul. Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, 51(11):24–31, 2013.
- [25] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. B4: Experience with a globally-deployed software defined wan. *ACM SIGCOMM Computer Communication Review*, 43(4):3–14, 2013.
- [26] Michael Jarschel, Thomas Zinner, Tobias Hofffeld, Phuoc Tran-Gia, and Wolfgang Kellerer. Interfaces, attributes, and use cases: A compass for sdn. *IEEE Communications Magazine*, 52(6):210–217, 2014.

- [27] Ruofan Jin and Bing Wang. Malware detection for mobile devices using software-defined networking. In *2013 Second GENI Research and Educational Experiment Workshop*, pages 81–88. IEEE, 2013.
- [28] Panos Kampanakis, Harry Perros, and Tsegereda Beyene. Sdn-based solutions for moving target defense network protection. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6. IEEE, 2014.
- [29] Dominik Klein and Michael Jarschel. An openflow extension for the omnet++ inet framework. In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, pages 322–329, 2013.
- [30] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, et al. Onix: A distributed control platform for large-scale production networks. In *OSDI*, volume 10, pages 1–6, 2010.
- [31] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, et al. Onix: A distributed control platform for large-scale production networks. In *OSDI*, volume 10, pages 1–6, 2010.
- [32] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2014.
- [33] Yuan-Liang Lan, Kuochen Wang, and Yi-Huai Hsu. Dynamic load-balanced path optimization in sdn-based data center networks. In *2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pages 1–6. IEEE, 2016.
- [34] Jie Li, Eitan Altman, and Corinne Touati. A general sdn-based iot framework with nvf implementation. *ZTE communications*, 13(3):42–45, 2015.
- [35] Madhusanka Liyanage, Andrei Gurtov, and Mika Ylianttila. *Software defined mobile networks (SDMN): beyond LTE network architecture*. John Wiley & Sons, 2015.

- [36] Meilian Lu, Wei Deng, and Yan Shi. Tf-idle timeout: Improving efficiency of tcam in sdn by dynamically adjusting flow entry lifecycle. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002681–002686, 2016.
- [37] Majdoub Manel, Youssef Habib, et al. An efficient mpls-based source routing scheme in software-defined wide area networks (sd-wan). In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 1205–1211. IEEE, 2017.
- [38] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, mar 2008.
- [39] Ahmadreza Montazerolghaem. Software-defined load-balanced data center: design, implementation and performance analysis. *Cluster Computing*, pages 1–20, 2020.
- [40] Ahmadreza Montazerolghaem, Mohammad Hossein Yaghmaee Moghaddam, and Alberto Leon-Garcia. Opensip: Toward software-defined sip networking. *IEEE Transactions on Network and Service Management*, 15(1):184–199, 2017.
- [41] Mohammad Noormohammadpour, Cauligi S. Raghavendra, Srikanth Kandula, and Sriram Rao. Quickcast: Fast and efficient inter-datacenter transfers using forwarding tree cohorts. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 225–233, 2018.
- [42] Kostas Pentikousis, Yan Wang, and Weihua Hu. Mobileflow: Toward software-defined mobile networks. *IEEE Communications Magazine*, 51(7):44–53, 2013.
- [43] Larry Peterson, Ali Al-Shabibi, Tom Anshutz, Scott Baker, Andy Bavier, Saurav Das, Jonathan Hart, Guru Palukar, and William Snow. Central office re-architected as a data center. *IEEE Communications Magazine*, 54(10):96–101, 2016.
- [44] Francisco Javier Ros and Pedro Miguel Ruiz. Five nines of southbound reliability in software-defined networks. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 31–36, 2014.

- [45] Shiva Rowshanrad, Sahar Namvarasl, Vajihe Abdi, Maryam Hajizadeh, and Manijeh Keshtgary. A survey on sdn, the future of networking. *Journal of Advanced Computer Science & Technology*, 3(2):232–248, 2014.
- [46] Mukhald Salih. *Software defined networking integration with mobile network towards scalable and programmable core*. PhD thesis, Brunel University London, 2019.
- [47] Sakir Sezer, Sandra Scott-Hayward, Pushpinder Kaur Chouhan, Barbara Fraser, David Lake, Jim Finnegan, Niel Viljoen, Marc Miller, and Navneet Rao. Are we ready for sdn? implementation challenges for software-defined networks. *IEEE Communications Magazine*, 51(7):36–43, 2013.
- [48] Alexander Shalimov, Dmitry Zuikov, Daria Zimarina, Vasily Pashkov, and Ruslan Smeliansky. Advanced study of sdn/openflow controllers. In *Proceedings of the 9th central & eastern european software engineering conference in russia*, pages 1–6, 2013.
- [49] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru Parulkar. Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, 1:132, 2009.
- [50] Rob Sherwood, N Mckeown, et al. The controller placement problem [c]. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, pages 7–12. Citeseer, 2012.
- [51] Yash Sinha, K Haribabu, et al. A survey: Hybrid sdn. *Journal of Network and Computer Applications*, 100:35–55, 2017.
- [52] Marco Tiloca, Alexandra Stagkopolou, and Gianluca Dini. Performance and security evaluation of sdn networks in omnet++/inet. *arXiv preprint arXiv:1609.04554*, 2016.
- [53] Amin Tootoonchian and Yashar Ganjali. Hyperflow: A distributed control plane for openflow. In *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, volume 3, 2010.
- [54] Jean Tourrilhes, Puneet Sharma, Sujata Banerjee, and Justin Pettit. The evolution of sdn and openflow: a standards perspective. *IEEE Computer Society*, 47(11):22–29, 2014.

- [55] Daphne Tuncer, Marinos Charalambides, Stuart Clayman, and George Pavlou. Adaptive resource management and control in software defined networks. *IEEE Transactions on Network and Service Management*, 12(1):18–33, 2015.
- [56] Daphne Tuncer, Marinos Charalambides, Stuart Clayman, and George Pavlou. On the placement of management and control functionality in software defined networks. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 360–365. IEEE, 2015.
- [57] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–10, 2008.
- [58] Cleverton Vicentini, Altair Santin, Eduardo Viegas, and Vilmar Abreu. Sdn-based and multitenant-aware resource provisioning mechanism for cloud-based big data streaming. *Journal of Network and Computer Applications*, 126:133–149, 2019.
- [59] Ryan Wallner and Robert Cannistra. An sdn approach: quality of service using big switch’s floodlight open-source controller. *Proceedings of the Asia-Pacific Advanced Network*, 35(14-19):10–7125, 2013.
- [60] Jia-Si Weng, Jian Weng, Yue Zhang, Weiqi Luo, and Weiming Lan. Benbi: Scalable and dynamic access control on the northbound interface of sdn-based vanet. *IEEE Transactions on Vehicular Technology*, 68(1):822–831, 2018.
- [61] Tingwei Zhu, Fang Wang, Yu Hua, Dan Feng, Yong Wan, Qingyu Shi, and Yanwen Xie. Mctcp: Congestion-aware and robust multicast tcp in software-defined networks. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, pages 1–10, 2016.

Παράρτημα

Σε αυτό το κεφάλαιο θα παρουσιαστούν τα συνοδευτικά αρχεία της εργασίας. Πιο συγκεκριμένα, τα αρχεία του δικτύου που παρουσιάζουν την τοπολογία των δικτύων, τα αρχεία με τις ρυθμίσεις και τις μεταβλητές. Τέλος, οι αλλαγές που έγιναν στον κωδικά του πακέτου του OpenFlow ώστε να υλοποιηθούν τα πειράματα της εργασίας.

Ο κώδικας του τελεία ned αρχείο για το παραδοσιακό δίκτυο.

```
import inet.networklayer.configurator.ipv4.IPv4NetworkConfigurator;
import inet.node.inet.StandardHost;
import openflow.openflow.controller.Open_Flow_Controller;
import inet.common.misc.ThruputMeteringChannel;
import inet.node.ethernet.EtherSwitch;
import openflow.utility.StaticSpanningTree;
import openflow.utility.ControllerInvolvementFilter;

network Test2_normal
{
    parameters:
        int numClients = default(0); //Default number of total clients
        @display("bgb=1618.3875,831.1912");
    types:
        channel ethernetline extends ThruputMeteringChannel
        {
            delay = 1ms; //Link Delay
            datarate = 10Mbps; //Data rate
```

```

        thruputDisplayFormat = "u"; //Display utilization
    }
submodules:
    configurator: IPv4NetworkConfigurator { //For IP assignment
        @display("p=1386.81,257.4925");
    }
    client[numClients]: StandardHost { //Clinets in the Network
        @display("p=160.1725,450.105,ri,130");
    }
    server: StandardHost { //Server
        @display("p=1403.03,462.27002");
    }
    //Addition of a second Server
    //server2: StandardHost {
    //@display("p=522,200");
    //}
    //00
    switch1: EtherSwitch {
        @display("p=454.16,448.0775");
    }
    //01
    switch2: EtherSwitch {
        @display("p=610.2775,277.76752");
    }
    //02
    switch3: EtherSwitch {
        @display("p=610.2775,610.2775");
    }
    //03
    switch4: EtherSwitch {
        @display("p=837.3575,277.76752");
    }
    //04
    switch5: EtherSwitch {

```

```

        @display("p=837.3575,610.2775");
    }
    //05
    switch6: EtherSwitch {
        @display("p=610.2775,448.0775");
    }
    //06
    switch7: EtherSwitch {
        @display("p=837.3575,448.0775");
    }
    //07
    switch8: EtherSwitch {
        @display("p=1084.7125,448.0775");
    }
    //08
    switch9: EtherSwitch {
        @display("p=1084.7125,277.76752");
    }
    //09
    switch10: EtherSwitch {
        @display("p=1074.5751,610.2775");
    }
connections:

    //Data Plane
    //Between Switches
    switch2.ethg++ <--> ethernetline <--> switch4.ethg++;
    switch6.ethg++ <--> ethernetline <--> switch7.ethg++;
    switch3.ethg++ <--> ethernetline <--> switch5.ethg++;
    switch4.ethg++ <--> ethernetline <--> switch9.ethg++;
    switch5.ethg++ <--> ethernetline <--> switch10.ethg++;
    switch7.ethg++ <--> ethernetline <--> switch8.ethg++;
    switch1.ethg++ <--> ethernetline <--> switch3.ethg++;
    switch1.ethg++ <--> ethernetline <--> switch6.ethg++;

```

```

switch1.ethg++ <--> ethernetline <--> switch2.ethg++;

for i=0..(numClients)-1 {
  client[i].ethg++ <--> ethernetline { @display("ls=green,2"); }
  <--> switch1.ethg++;
}

server.ethg++ <--> ethernetline <--> switch8.ethg++;
server.ethg++ <--> ethernetline <--> switch9.ethg++;
server.ethg++ <--> ethernetline <--> switch10.ethg++;
}

```

Ο κώδικας του τελεία ned αρχείο για το SDN.

```

import inet.networklayer.configurator.ipv4.Ipv4NetworkConfigurator;
import inet.node.inet.StandardHost;
import openflow.openflow.controller.Open_Flow_Controller;
import openflow.openflow.switch.Open_Flow_Switch;
import inet.common.misc.ThruputMeteringChannel;
import inet.node.ethernet.EtherSwitch;
import openflow.utility.OpenFlowGraphAnalyzer;
import openflow.utility.StaticSpanningTree;
import openflow.utility.ControllerInvolvementFilter;

network Test2
{
  parameters:
    int numClients = default(0); //Default number of total clients
    @display("bgb=1618.3875,831.1912");
  types:
    channel ethernetline extends ThruputMeteringChannel
    {
      delay = 1ms; //Link Delay
    }
}

```

```

        datarate = 10Mbps; //Data Rate
        thruputDisplayFormat = "u"; //Display utilization
    }
submodules:
    configurator: IPv4NetworkConfigurator { //For IP assignment
        @display("p=1354.37,68.935");
    }
    client[numClients]: StandardHost { //Clinets in the Network
        @display("p=40.55,444.0225,ri,70");
    }
    server: StandardHost { //Server
        @display("p=1532.79,444.0225");
    }
    //Addition of second Server
    //server2: StandardHost {
    //@display("p=522,200");
    //}
    //00
    open_flow_switch1: Open_Flow_Switch {
        @display("p=358.8675,441.995;i2=OpenFlow-Logo-small");
    }
    //01
    open_flow_switch2: Open_Flow_Switch {
        @display("p=640.69,269.6575;i2=OpenFlow-Logo-small");
    }
    //02
    open_flow_switch3: Open_Flow_Switch {
        @display("p=640.69,604.195;i2=OpenFlow-Logo-small");
    }
    //03
    open_flow_switch4: Open_Flow_Switch {
        @display("p=924.54004,269.6575;i2=OpenFlow-Logo-small");
    }
    //04

```

```

open_flow_switch5: Open_Flow_Switch {
    @display("p=924.54004,604.195;i2=OpenFlow-Logo-small");
}
//05
open_flow_switch6: Open_Flow_Switch {
    @display("p=640.69,441.995;i2=OpenFlow-Logo-small");
}
//06
open_flow_switch7: Open_Flow_Switch {
    @display("p=924.54004,441.995;i2=OpenFlow-Logo-small");
}
//07
open_flow_switch8: Open_Flow_Switch {
    @display("p=1218.5275,441.995;i2=OpenFlow-Logo-small");
}
//08
open_flow_switch9: Open_Flow_Switch {
    @display("p=1218.5275,269.6575;i2=OpenFlow-Logo-small");
}
//09
open_flow_switch10: Open_Flow_Switch {
    @display("p=1208.39,604.195;i2=OpenFlow-Logo-small");
}
open_flow_controller1: Open_Flow_Controller {
    @display("p=817.0825,54.7425;i2=OpenFlow-Logo-small");
}
openFlowGraphAnalyzer: OpenFlowGraphAnalyzer { //For analyzing more data
    @display("p=194.64,101.375");
}
connections:

//Control Plane
open_flow_switch1.gateControlPlane++ <--> ethernetline
{ @display("ls=blue,2"); } <--> open_flow_controller1.ethg++;

```

```

open_flow_switch2.gateControlPlane++ <--> ethernetline
{ @display("ls=blue,2"); } <--> open_flow_controller1.ethg++;
open_flow_switch3.gateControlPlane++ <--> ethernetline
{ @display("ls=blue,2"); } <--> open_flow_controller1.ethg++;
open_flow_switch4.gateControlPlane++ <--> ethernetline
{ @display("ls=blue,2"); } <--> open_flow_controller1.ethg++;
open_flow_switch6.gateControlPlane++ <--> ethernetline
{ @display("ls=blue,2"); } <--> open_flow_controller1.ethg++;
open_flow_switch5.gateControlPlane++ <--> ethernetline
{ @display("ls=blue,2"); } <--> open_flow_controller1.ethg++;
open_flow_switch7.gateControlPlane++ <--> ethernetline
{ @display("ls=blue,2"); } <--> open_flow_controller1.ethg++;
open_flow_switch8.gateControlPlane++ <--> ethernetline
{ @display("ls=blue,2"); } <--> open_flow_controller1.ethg++;
open_flow_switch9.gateControlPlane++ <--> ethernetline
{ @display("ls=blue,2"); } <--> open_flow_controller1.ethg++;
open_flow_switch10.gateControlPlane++ <--> ethernetline
{ @display("ls=blue,2"); } <--> open_flow_controller1.ethg++;

```

```
//Data Plane
```

```
//Between Switches
```

```

open_flow_switch2.gateDataPlane++ <--> ethernetline <-->
open_flow_switch4.gateDataPlane++;
open_flow_switch6.gateDataPlane++ <--> ethernetline <-->
open_flow_switch7.gateDataPlane++;
open_flow_switch3.gateDataPlane++ <--> ethernetline <-->
open_flow_switch5.gateDataPlane++;
open_flow_switch4.gateDataPlane++ <--> ethernetline <-->
open_flow_switch9.gateDataPlane++;
open_flow_switch5.gateDataPlane++ <--> ethernetline <-->
open_flow_switch10.gateDataPlane++;
open_flow_switch7.gateDataPlane++ <--> ethernetline <-->
open_flow_switch8.gateDataPlane++;
open_flow_switch1.gateDataPlane++ <--> ethernetline <-->

```

```

open_flow_switch3.gateDataPlane++;
open_flow_switch1.gateDataPlane++ <--> ethernetline <-->
open_flow_switch6.gateDataPlane++;
open_flow_switch1.gateDataPlane++ <--> ethernetline <-->
open_flow_switch2.gateDataPlane++;

//Between Clients
for i=0..(numClients)-1 {
client[i].ethg++ <--> ethernetline { @display("ls=green,2"); }
<--> open_flow_switch1.gateDataPlane++;
}

//Between server
server.ethg++ <--> ethernetline <-->
open_flow_switch8.gateDataPlane++;
server.ethg++ <--> ethernetline <-->
open_flow_switch9.gateDataPlane++;
server.ethg++ <--> ethernetline <-->
open_flow_switch10.gateDataPlane++;
}

```

Ο κώδικας του τελεία ini αρχείο για το παραδοσιακό δίκτυο.

```

[General]
network = Test2_normal
description = "Normal Network with multiple routes"

debug-on-errors = true //For live debugging
#Creating recording vector file
output-vector-file = ${resultdir}/${inifile}/${inifile}-${runnumber}.vec
#Creating recording scalar file
output-scalar-file = ${resultdir}/${inifile}/${inifile}-${runnumber}.sca

```

```

sim-time-limit = 200s //Simulation limit time

#Configurator not involving in routing descisions
*.configurator.optimizeRoutes = false

#Give options for the total number of clients
*.numClients = (${NoClients = 5, 10, 15, 20, 25, 30})

**.ppp[*].queue.frameCapacity = 100 # in routers

#Host params
*.server.numTcpApps = 1 //Number of tcp applications for the server
*.server.tcpApp[0].typename = "TCPSinkApp" //Only receiving
*.server.tcpApp[0].localPort = 1000 //Local port for the server
*.client*.numTcpApps = 1 //Number of tcp applications for the clients
*.client*.tcpApp[0].typename = "TCPSessionApp" //Only sending
*.client*.tcpApp[0].connectAddress = "server" //Name of the module to connect
*.client*.tcpApp[0].connectPort = 1000 //Connect port
*.client*.tcpApp[0].tOpen = 1s //When to start the connection
*.client*.tcpApp[0].tSend = 2s //When to start sending
*.client*.tcpApp[0].sendBytes = 20Mib //File size
*.client*.tcpApp[0].tClose = 0s //Don't close connection if not finished
**.arp.cacheTimeout = 200s //When arp cache timeout

# For the recording
**.queueSize.result-recording-modes = +vector,+stats
**.bufferSize.result-recording-modes = +vector,+stats
**.waitingTime.result-recording-modes = +vector,+stats
**.tcpApp[1].connectionStarted.result-recording-modes = +vector,+stats
**.tcpApp[1].connectionEstablished.result-recording-modes = +vector,+stats
**.tcpApp[1].connectionFinished.result-recording-modes = +vector,+stats
**.tcpApp[1].transmittedBytes.result-recording-modes = +vector,+stats

**.packets.result-recording-modes = +count

```

```

**.packetBytes.result-recording-modes = +sum
**.packets*.scalar-recording = true
**.packetBytes*.scalar-recording = true

**.bufferSize*.vector-recording = true
**.queueSize*.vector-recording = true
**.waitingTime*.vector-recording = true

**.flowTable*.scalar-recording = true

**.tcpApp[1].connectionStarted*.vector-recording = true
**.tcpApp[1].connectionEstablished*.vector-recording = true
**.tcpApp[1].connectionFinished*.vector-recording = true
**.tcpApp[1].transmittedBytes*.vector-recording = true

**.vector-recording = true
**.scalar-recording = true

```

Ο κώδικας του τελεία ini αρχείο για το SDN.

```

[General]
network = Test2
description = "SDN Network with multiple routes"

debug-on-errors = true //For live debugging
#Creating recording vector file
output-vector-file = ${resultdir}/${inifile}/${inifile}-${runnumber}.vec
#Creating recording scalar file
output-scalar-file = ${resultdir}/${inifile}/${inifile}-${runnumber}.sca

sim-time-limit = 200s //Simulation limit time

```

```

# configure control and data plane interfaces separately
*.configurator.config = xmldoc("ipv4config.xml")

*.controllerInvolvementFilter.module-eventlog-recording = true //record more data

#Give options for the total number of clients
*.numClients = (${NoClients = 5, 10, 15, 20, 25, 30})

##openflow params
#Name of the controller to connect at
**.open_flow_switch*.OF_Switch.connectAddress = "open_flow_controller1"
**.open_flow_switch*.OF_Switch.connectAt = 0s //When to connect
**.OF_Switch.bufferCapacity = 3712 //Switch's buffer capacity
**.open_flow_controller*.serviceTime = 0.000005556ss //Controllers service time
**.ppp[*].queue.frameCapacity = 10 # in routers
**.open_flow_switch*.tcp.mss = 800 //Maximum segment size

#Host params
*.server.numTcpApps = 1 //Number of tcp applications for the server
*.server.tcpApp[0].typename = "TCPSinkApp" //Only receiving
*.server.tcpApp[0].localPort = 1000 //Local port for the server
*.client*.numTcpApps = 1 //Number of tcp applications for the clients
*.client*.tcpApp[0].typename = "TCPSessionApp" //Only sending
*.client*.tcpApp[0].connectAddress = "server" //Name of the module to connect
*.client*.tcpApp[0].connectPort = 1000 //Connect port
*.client*.tcpApp[0].tOpen = 1s //When to start the connection
*.client*.tcpApp[0].tSend = 2s //When to start sending
*.client*.tcpApp[0].sendBytes = 20Mib //File size
*.client*.tcpApp[0].tClose = 0s //Don't close connection if not finished

#For displaying interface addresses
*.client[*].interfaceTable.displayAddresses = true
*.server.interfaceTable.displayAddresses = true

```

```

#Controller Params
#Number of Controller's applications
**.open_flow_controller*.numControllerApps = 2
#For balancing load
**.open_flow_controller*.controllerApps[0].typename = "LLDPBalancedMinHop"
**.open_flow_controller*.controllerApps[0].flowModHardTimeOut = 0
**.open_flow_controller*.controllerApps[0].flowModIdleTimeOut = 1
**.open_flow_controller*.controllerApps[0].printMibGraph = true
#For topology
**.open_flow_controller*.controllerApps[1].typename = "LLDPAgent"
**.open_flow_controller*.controllerApps[1].flowModIdleTimeOut = 1
**.open_flow_controller*.controllerApps[1].flowModHardTimeOut = 0

**.arp.cacheTimeout = 200s //When arp cache timeout

# For the recording
**.queueSize.result-recording-modes = +vector,+stats
**.bufferSize.result-recording-modes = +vector,+stats
**.waitingTime.result-recording-modes = +vector,+stats
**.tcpApp[1].connectionStarted.result-recording-modes = +vector,+stats
**.tcpApp[1].connectionEstablished.result-recording-modes = +vector,+stats
**.tcpApp[1].connectionFinished.result-recording-modes = +vector,+stats
**.tcpApp[1].transmittedBytes.result-recording-modes = +vector,+stats

**.packets.result-recording-modes = +count
**.packetBytes.result-recording-modes = +sum
**.packets*.scalar-recording = true
**.packetBytes*.scalar-recording = true

**.numPacketIn*.scalar-recording = true

**.bufferSize*.vector-recording = true
**.queueSize*.vector-recording = true
**.waitingTime*.vector-recording = true

```

```

**.flowTable**.scalar-recording = true

**.tcpApp[1].connectionStarted*.vector-recording = true
**.tcpApp[1].connectionEstablished*.vector-recording = true
**.tcpApp[1].connectionFinished*.vector-recording = true
**.tcpApp[1].transmittedBytes*.vector-recording = true

**.controllerApps[*].*.scalar-recording = true

**.vector-recording = true
**.scalar-recording = true

```

Αλλαγές στο πακέτο του INET για περισσότερες τιμές του data rate.

In order to add different values for data rate, other than 10Mbps, follow these steps:

- 1) Add the following line to file Ethernet.h

```

#define TWENTY_MEGABIT_ETHERNET_TXRATE 20000000.0
/* 20 Mbit/sec (in bits/s) */

```

after the line

```

#define ETHERNET_TXRATE 10000000.0
/* 20 Mbit/sec (in bits/s) */

```

- 2) In file EtherMACBase.h change the following line from

```

NUM_OF_ETHERDESCRS = 6

```

To

```
NUM_OF_ETHERDESCRS = 7
```

3) In file EtherMACBase.cc

After this

```
{ETHERNET_TXRATE,  
 0.5 / ETHERNET_TXRATE,  
 MIN_ETHERNET_FRAME_BYTES,  
 0,  
 0,  
 MIN_ETHERNET_FRAME_BYTES,  
 MIN_ETHERNET_FRAME_BYTES,  
 512 / ETHERNET_TXRATE,  
 2500 /*m*/ / SPEED_OF_LIGHT_IN_CABLE},
```

Add that

```
{TWENTY_MEGABIT_ETHERNET_TXRATE,  
 0.5 / TWENTY_MEGABIT_ETHERNET_TXRATE,  
 MIN_ETHERNET_FRAME_BYTES,  
 0,  
 0,  
 -1, // half-duplex is not supported  
 0,  
 0.0,  
 0.0},
```

Συντμήσεις

ONF	Open Networking Foundation
SDN	Software-defined Networking
SDMN	Software-defined Mobile Networking
SD-WAN	Software-defined Wireless Area Network
SD-LAN	Software-defined Local Area Network
VLAN	Virtual Local Area Network
RAN	Radio Access Network
DDoS	Distributed Denial of Service
NVF	Network Function Virtualization
API	Application Programming Interface
IoT	Internet of Things
P2P	Peer-to-Peer
PPP	Point-to-Point Protocol
MAC	Media Access Control

IP	Internet Protocol
TCP	Transport Control Protocol
UDP	User Datagram Protocol
OSI	Open Systems Interconnection
MPLS	Multiprotocol Label Switching
CORD	Central Office Re-architected as a Datacenter
OXM	OpenFlow Extensible Match
TLV	Type Length Value
OMNeT++	Objective Modular Network Testbed in C++
Mbps	Mega bits per second
RTT	Round Trip Time
ned file	Network Description File
ini file	Initialization File

Γλωσσάρι Ξενικών Όρων

Software-defined Network	Δίκτυο βασισμένο σε λογισμικό
Data Plane	Επίπεδο δεδομένων
Control Plane	Επίπεδο λήψης αποφάσεων
ARP	Πρωτόκολλο επίλυσης διευθύνσεων
Forwarding	Προώθηση
Flow-Table	Πίνακα-Ροής
Header	Κεφαλίδα
Physical Layer	Φυσικό Επίπεδο
Router	Δρομολογητής
Switch	Μεταγωγέας
Client	Πελάτης/Χρήστης

Server	Εξυπηρετητής
Controller	Ελεγκτής / Διαχειριστής
Software	Λογισμικό
Hardware	Υλικό
Modular	Δομοστοιχειωτό
Round Trip Time	Χρόνος μετάδοσης μετ' επιστροφής