



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

SDN & NETWORK SLICING

ΠΑΠΑΛΑΤΟΣ ΣΩΣΗΠΑΤΡΟΣ

ΕΠΙΒΛΕΠΩΝ: Αν. Καθ. Καρκαζής Παναγιώτης

ΑΘΗΝΑ

ΙΟΥΝΙΟΣ 2022

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

SDN & Network Slicing

Παπαδάτος Σωσίπατρος

ΑΜ: 161189

ΕΠΙΒΛΕΠΟΝΤΕΣ: Καρκαζής Πανγιώτης

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία θα αναλυθεί ο μηχανισμός τεμαχισμού δικτύου. Αρχικά θα γίνει μία ιστορική αναδρομή στα ασύρματα δίκτυα (1G-5G). Φτάνοντας στο 5G θα αναλυθεί η αρχιτεκτονική του και θα αναφερθούν τα βασικότερα πλεονεκτήματα που μας δίνει αυτό. Στην συνέχεια θα αναλυθεί σε βάθος η έννοια του Network Slicing. Επίσης, θα γίνει αναφορά στα πλεονεκτήματα, τις προκλήσεις του τεμαχισμού δικτύου και στο network slicing framework. Τελειώνοντας με το network slicing, και μπαίνοντας στο επόμενο κεφάλαιο, θα αναλυθεί το SDN, και θα δοθεί έμφαση στους SDN controllers Flowvisor, OpenDaylight και Onos παρουσιάζοντας το τρόπο εγκατάστασής τους και τον τρόπο λειτουργίας τους. Ακόμη θα γίνει μία αναφορά στο πρωτόκολλο OpenFlow. Τέλος, θα παρουσιαστεί ο τρόπος με τον οποίο ένα εικονικό δίκτυο μπορεί να τεμαχιστεί σε slices με την χρήση του SDN Controller Flowvisor, του Mininet και του MiniEdit. Το πρώτο μέρος θα αφορά ένα παράδειγμα κατά το οποίο χωρίζεται ένα εικονικό δίκτυο σε δύο διαφορετικά slices, με αποτέλεσμα η επικοινωνία μεταξύ των host να είναι περιορισμένη αναλόγως με το slice στο οποίο ανήκουν. Το δεύτερο μέρος θα αφορά ένα πιο ρεαλιστικό παράδειγμα, κατά το οποίο θα δημιουργηθούν δύο slice. Το ένα θα αφορά την αποστολή ενός video και το άλλο την κίνηση για non-video. Αποτέλεσμα του παραδείγματος αυτού, θα είναι κάθε κίνηση που αφορά την αποστολή video, να πηγαίνει από το slice στο οποίο θα υπάρχει μεγαλύτερο bandwidth, ενώ οποιαδήποτε άλλη κίνηση θα πηγαίνει από το το slice που είναι για non-video. Με αυτόν τον τρόπο επιτυγχάνουμε την διαμόρφωση του υποκείμενου δικτύου με βάση τις απαιτήσεις της εκάστοτε εφαρμογής και την εξασφάλιση του επιθυμητού επιπέδου ποιότητας υπηρεσιών.

ABSTRACT

In this thesis the network slicing mechanism will be analyzed. Firstly, a historical review of wireless networks(1G-5G) will take place. Next, the architecture of 5G will be analyzed and the main advantages that this gives us will be mentioned. Then the concept of Network Slicing will be analyzed in depth. Also, the advantages, challenges of network slicing and the network slicing framework will be mentioned. Finishing with network slicing, and entering the next chapter, SDN will be analyzed, and the SDN controller Flowvisor, OpenDaylight and Onos will be emphasized by showing how they are deployed and how they work. Furthermore, a reference to OpenFlow protocol will be made. Finally, there will be an experiment consisting of two different parts, showing how a virtual network can be sliced using the SDN controller Flowvisor, Mininet and MiniEdit. In the first part present an example where a virtual network is divided into two different slices, resulting in communication between hosts being restricted depending on the slice they belong to. The second part, will be about a more realistic example, in which two slices will be created. One will involve sending a video and the other will involve non-video traffic. As a result of this, the traffic that involves video uses the network slice with highest bandwidth, while any other traffic will uses the slice with the lower bandwidth. Thus, we managed to configure the network basen on the application, and provide the required QoS level.

EYXΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω τις ευχαριστίες και την ευγνωμοσύνη μου επιβλέποντα καθηγητή κ. Παναγιώτη Καρκαζή, για την πολύτιμη βοήθεια, τις συμβουλές που μου έδωσε, και τον χρόνο που διέθεσε για την διεκπεραίωση της πτυχιακής. Ακόμη, θα ήθελα να ευχαριστήσω την οικογένεια μου για την στήριξη που είχα καθ' όλη την διάρκεια των σπουδών μου.

Περιεχόμενα

Περιεχόμενα	10
Κεφάλαιο 1. Εισαγωγή	13
Κεφάλαιο 2. Δίκτυα Κινητών Επικοινωνιών 1G-5G.....	14
2.1 Ιστορική αναδρομή ασύρματων δικτύων έως και σήμερα	14
2.2 Αρχιτεκτονική 5G.....	17
Κεφάλαιο 3. Network Slicing.....	20
3.1 Ορισμός Network Slicing	20
3.2 Πλεονεκτήματα στον Τεμαχισμού Δικτύου	20
3.3 Προκλήσεις.....	21
3.3.1 Κοινή χρήση πόρων.....	21
3.3.2 Δημιουργία και διαχείριση δυναμικών τμημάτων.....	21
3.3.3 Απομόνωση μεταξύ τμημάτων δικτύου.....	22
3.3.4 Διαχείριση της κινητικότητας κατά το network slicing.....	22
3.3.5 Ασφάλεια κατά το network slicing.....	23
3.3.6 Ασύρματη εικονικοποίηση πόρων.....	23
3.3.7 Αλγοριθμικές πτυχές της κατανομής των πόρων.....	23
3.4 Network slicing framework	24
Κεφάλαιο 4. Software Defined Networks	27
4.1 Ορισμός SDN	27
4.2 Η αρχιτεκτονική	27
4.3 Επικοινωνία SDN.....	28
4.4 OpenFlow protocol	29
4.4.1 Πλεονεκτήματα OpenFlow πρωτοκόλλου	31
4.5 SDN controllers.....	31
4.6 Mininet	32
4.6.1 Εγκατάσταση Mininet.....	32
4.7 OpenDaylight.....	38
4.7.1 Αρχιτεκτονική OpenDaylight.....	38
4.7.2 Μελέτη περίπτωσης.....	40
4.7.3 REST API.....	41
4.7.4 Εγκατάσταση-Παρουσίαση OpenDaylight.....	42
4.8 ONOS.....	50
4.8.1 Αρχιτεκτονική ONOS.....	51
4.8.2 Λειτουργία ONOS.....	53
4.8.3 Docker.....	54

4.8.4 Εγκατάσταση-Παρουσίαση ONOS	54
4.9 Flowvisor	61
4.9.1 Flowvisor – SDN	61
4.9.2 Εγκατάσταση Flowvisor.....	61
Κεφάλαιο 5. Επίδειξη Network Slicing.....	66
Μέρος 1ο.....	66
Μέρος 2ο.....	78
Κεφάλαιο 6. Συμπεράσματα	84

1. Εισαγωγή

Η ταχεία ανάπτυξη και οι καινοτομίες των τεχνολογιών κινητής δικτύωσης, μας οδήγησαν σε μία νέα εποχή των κινητών επικοινωνιών, σε αυτήν του 5G.

Το 5G είναι η πέμπτη γενιά της τεχνολογίας ευρυζωνικών κυψελοειδών δικτύων, και διαδέχθηκε το 4G(LTE/WiMax), 3G(UMTS) και το 2G(GSM). Ο κύριος στόχος του 5G είναι ο υψηλός ρυθμός μετάδοσης δεδομένων, η ελάχιστη έως και μηδαμινή καθυστέρηση, η μεγαλύτερη γεωγραφική κάλυψη σε σχέση με τους προκατόχους του, η ασφάλεια και η αξιοπιστία. Μία μεγάλη προσθήκη στα 5G δίκτυα, είναι η υποστήριξη του Internet of Things (IoT).

Το 5G δίνει τεράστια έμφαση στην ικανοποίηση διαφόρων απαιτήσεων ποιότητας των υπηρεσιών που παρέχονται (Quality of Service - QoS) σε διαφορετικά σενάρια εφαρμογών, πχ όσον αφορά το ρυθμό μετάδοσης δεδομένων και την καθυστέρηση. Μία πρόκληση που το 5G έχει αντιμετωπίσει επάξια ως τώρα, είναι ο υψηλός ρυθμός μετάδοσης που προσφέρει σε οποιοδήποτε σημείο και αν βρίσκεται ο χρήστης, ακόμη και αν είναι κοντά στις άκρες των κυψελών.

Οι ταχύτητες που προσφέρει το 5G είναι πολύ μεγάλες, και κυμαίνονται από 1Gbps σε 10 Gbps¹, σε αντίθεση με την τεχνολογία προηγούμενης γενιάς 4G LTE η οποία μπορεί να φτάσει στο μέγιστο ταχύτητες της τάξης του 1Gbit/s.² Η ραγδαία αυτή αύξηση που προσφέρει το 5G στην ταχύτητα, αυτόματα συμβάλει στην εξυπηρέτηση υπηρεσιών και εφαρμογών που χρειάζονται bandwidth τέτοιου βεληνεκού, η οποίες πριν, δεν μπορούσαν να εξυπηρετηθούν αποδοτικά. Ακόμη, η μείωση της καθυστέρησης, δηλαδή του χρόνου ανάμεσα στην αποστολή αιτήματος στο δίκτυο και της λήψης απάντησης από αυτό, είναι κρίσιμη για την ανάπτυξη εφαρμογών που βασίζονται στην γρήγορη απόκριση του δικτύου για τις λειτουργίες τους.

Μία από τις τεχνολογίες που εφαρμόζονται στα 5G δίκτυα, είναι το Network Slicing. Σύμφωνα με αυτό, πραγματοποιείται τεμαχισμός του δικτύου, με σκοπό την δημιουργία εικονικών υποδομών δικτύου, κάθε μία από τις οποίες είναι εξειδικευμένη στην ικανοποίηση συγκεκριμένου τύπου υπηρεσίας πχ IoT με αισθητήρες μέτρησης ρύπων στους δρόμους των πόλεων. Κάθε τεμάχιο λοιπόν που δημιουργείτε, προσαρμόζεται κατάλληλα για τον τύπο υπηρεσίας που απαιτείται.

Στα κεφάλαια τα οποία ακολουθούν, περιγράφεται το 5G δίκτυο και η αρχιτεκτονική του. Γίνεται λεπτομερής ανάλυση σε ότι αφορά το Network Slicing, και στους μηχανισμούς που συμβάλουν για την επίτευξή του (SDN controllers, OpenFlow protocol, Mininet, MiniEdit, VM). Τέλος, γίνεται μία παρουσίαση δύο περιπτώσεων τεμαχισμού εικονικού δικτύου.

¹ <https://mse238blog.stanford.edu/2017/07/ssound/1g-2g-5g-the-evolution-of-the-gs/>

² <https://www.verizon.com/about/news/what-4g-lte-and-why-it-matters>

2. Δίκτυα Κινητών Επικοινωνιών 1G-5G

Την 1^η Δεκεμβρίου του 2018, η Νότια Κορέα έγινε η πρώτη χώρα παγκοσμίως που προσέφερε το 5G, και είναι δίκαιο να πούμε ότι η κινητή βιομηχανία έχει σημειώσει εκπληκτική πρόοδο από το όχι και τόσο μακρινό 1973 όπου πραγματοποιήθηκε η πρώτη τηλεφωνική κλήση μέσω κινητού.

Η τεχνολογία ασύρματης επικοινωνίας έχει εξελιχθεί εδώ και αρκετές δεκαετίες. Ξεκινώντας από το τότε επαναστατικό 1G μέχρι και το 5G του σήμερα.

Τί έχει όμως αλλάξει πραγματικά, και γιατί ήταν απαραίτητη με το πέρασ το χρόνων η μετάβαση από την μία γενιά στην άλλη;

2.1 Ιστορική αναδρομή ασύρματων δικτύων έως και σήμερα

1G Εκεί που ξεκίνησαν όλα.

Η πρώτη γενιά δικτύων κινητής τηλεφωνίας, κυκλοφόρησε από την Nippon Telegraph στο Τόκιο το 1979. Έως και το 1984, η NTT είχε κυκλοφορήσει το 1G προκειμένου να καλύψει ολόκληρη την Ιαπωνία.³

Τέσσερα χρόνια μετά την κυκλοφορία του 1G, το 1983, οι ΗΠΑ ενέκριναν τις λειτουργίες του, και το DynaTAC της Motorola έγινε ένα από τα πρώτα κινητά τηλέφωνα που γνώρισα ευρεία χρήση στις ΗΠΑ.

Η τεχνολογία αυτή όμως, είχε και κάποια μειονεκτήματα. Το 1G είχε δημιουργηθεί μόνο για φωνητική επικοινωνία, και επομένως δεν επέτρεπε την μετάδοση δεδομένων. Η κάλυψη ήταν κακή και η ποιότητα του ήχου σε πολύ χαμηλά επίπεδα. Δεν υπήρχε υποστήριξη περιαγωγής μεταξύ διαφόρων χειριστών και δεν υπήρχε συμβατότητα μεταξύ των συστημάτων που λειτουργούσαν σε διαφορετικές περιοχές συχνοτήτων. Όμως το σημαντικότερο μειονέκτημα ήταν η ασφάλεια των κλήσεων. Δεν υπήρχε κρυπτογράφηση κατά τις κλήσεις, με αποτέλεσμα οποιοσδήποτε είχε έναν σαρωτή ραδιοφώνου, να έχει την δυνατότητα να ακούσει τις κλήσεις.

Ωστόσο, παρά τα μειονεκτήματα που υπήρχαν το DynaTAC, κατάφερε να συγκεντρώσει 20 εκατομμύρια⁴ συνδρομητές παγκοσμίως έως και το 1990.

Η επιτυχία του 1G άνοιξε τον δρόμο για την δεύτερη γενιά, το 2G.

2G Η πολιτιστική Επανάσταση

Το 2G, κυκλοφόρησε με το πρότυπο GSM στην Φιλανδία το 1991. Οι κλήσεις, για πρώτη φορά, μπορούσαν να κρυπτογραφηθούν και οι φωνητικές κλήσεις ήταν πολύ πιο καθαρές σε σχέση με το 1G. Τα συστήματα 2G είχαν σημαντικά πιο αποτελεσματικό φάσμα, επιτρέποντας πολύ μεγαλύτερα επίπεδα ασύρματης

³ <https://www.brainbridge.be/en/blog/1g-5g-brief-history-evolution-mobile-standards>

διείσδυσης. Επίσης, εισήγαγε υπηρεσίες δεδομένων για κινητά, ξεκινώντας με μηνύματα SMS. Όλα τα SMS που αποστέλλονται είναι ψηφιακά κρυπτογραφημένα, επιτρέποντας την μεταφορά δεδομένων με τέτοιο τρόπο ώστε μόνο ο προοριζόμενος δέκτης να μπορεί να τα λάβει και να τα διαβάσει.

Το αναλογικό παρελθόν του 1G, έδωσε την θέση του στο ψηφιακό μέλλον που παρουσιάζει το 2G. Αυτό οδήγησε σε μαζική υιοθέτηση τόσο από καταναλωτές όσο και από επιχειρήσεις σε πρωτοφανή για τα τότε δεδομένα κλίμακα.

Οι ταχύτητες μεταφοράς αρχικά ήταν μόνο περίπου⁵ 9,6kbit/s. Οι φορείς ωστόσο έσπευσαν να επενδύσουν σε νέες υποδομές, όπως πύργους κινητής τηλεφωνίας. Μέχρι και το τέλος της εποχής του 2G, ταχύτητες⁶ 40kbit/s. ήταν επιτεύξιμες και οι συνδέσεις EDGE προσέφεραν ταχύτητες έως και⁷ 500 kbit/s.

Το 2G έφερε την επανάσταση στα ασύρματα δίκτυα, και άλλαξε τον κόσμο για πάντα.

3G Η επανάσταση της Μεταγωγής πακέτων

Το 3G κυκλοφόρησε από την NTT DoCoMo το 2001 βασίστηκε στις τεχνολογίες CDMA2000 και EDGE και είχε στόχο να τυποποιήσει το πρωτόκολλο δικτύου που χρησιμοποιούν οι προμηθευτές. Με άλλα λόγια, οι χρήστες μπορούσαν να έχουν πρόσβαση σε δεδομένα από οποιαδήποτε τοποθεσία στον κόσμο, καθώς τυποποιήθηκαν τα πακέτα δεδομένων μέσω των οποίων έγινε εφικτή η σύνδεση στο διαδίκτυο.

Το 3G είχε δυνατότητες ταχύτητας έως και 2mbps. Έδωσε την δυνατότητα στα smartphones να παρέχουν ταχύτερη επικοινωνία, να στέλνουν/λαμβάνουν email και μηνύματα, να παρέχουν γρήγορη περιήγηση στον ιστό και γενικότερα μεγαλύτερη ασφάλεια.

Οι αυξημένες δυνατότητες μεταφοράς δεδομένων του 3G οδήγησαν στην άνοδο νέων υπηρεσιών όπως η τηλεδιάσκεψη, η ροή βίντεο και η φωνή μέσω IP(πχ Skype). Το 2002, κυκλοφόρησε το Blackberry και πολλά από τα ισχυρά χαρακτηριστικά του έγιναν δυνατά χάρη στο 3G.

Το πικ όμως της εποχής του 3G ήρθε με το λανσάρισμα του iPhone το 2007, πράγμα που σήμαινε ότι η δυνατότητα δικτύου επρόκειτο να διευρυνθεί όσο ποτέ άλλοτε.

4G Η εποχή του streaming

Το 4G αναπτύχθηκε πρώτη φορά στην Σουηδία και στην Νορβηγία το 2009 ως το πρότυπο 4G Long Term Evolution(LTE). Ξεκινώντας από τουλάχιστον 12,5 Mbps, το 4G παρείχε βίντεο υψηλής ποιότητας, γρήγορη πρόσβαση στον ιστό, βίντεο HD και βιντεοδιάσκεψη HQ.

Κατά την μετάβαση από το 2G στο 3G, οι χρήστες αρκεί να άλλαζαν κάρτα sim. Τώρα, με το 4G ήταν απαραίτητη η αλλαγή συσκευών, οι οποίες είχαν σχεδιαστεί

⁵ <https://www.brainbridge.be/en/blog/1g-5g-brief-history-evolution-mobile-standards>

⁶ <https://www.brainbridge.be/en/blog/1g-5g-brief-history-evolution-mobile-standards>

⁷ <https://www.brainbridge.be/en/blog/1g-5g-brief-history-evolution-mobile-standards>

ειδικά για να υποστηρίξουν 4G. Είναι προφανές πως εταιρίες κινητής τηλεφωνίας αύξησαν κατακόρυφα τα κέρδη τους, παράγοντας συσκευές που υποστήριζαν το 4G.

Μπορεί το 4G να ήταν το τρέχον πρότυπο σε όλο τον κόσμο, ωστόσο ορισμένες περιοχές είχαν τεράστια προβλήματα δικτύου με αποτέλεσμα να έχουν χαμηλή διείσδυση 4G LTE.

4G LTE Η εποχή του streaming

Το 4G LTE είναι ένας όρος που χρησιμοποιείται για να περιγράψει τον τύπο ασύρματης τεχνολογίας. Το 4G LTE είναι συντομογραφία του 4G long-term evolution. Επομένως αναφέρεται σε 2 όρους συνδυασμένους. Αρχικά, το 4G αντιπροσωπεύει όπως είναι προφανές την τέταρτη γενιά κινητής τηλεφωνίας. Όσο για το long-term evolution (LTE), είναι η βιομηχανική ορολογία που χρησιμοποιείται για να περιγράψει τον συγκεκριμένο τύπο 4G που προσφέρει την ταχύτερη απόδοση διαδικτύου για κινητά. Το 4G LTE είναι 10 φορές πιο γρήγορο από ό,τι το 3G.⁸

5G Η εποχή του Διαδικτύου των Αντικειμένων (Internet of Things - IoT)

Το 5G είναι η πέμπτη γενιά ασύρματης τεχνολογίας. Παρέχει υψηλότερη ταχύτητα, χαμηλότερη καθυστέρηση και μεγαλύτερη χωρητικότητα από τα δίκτυα 4G LTE. Είναι μία από τις πιο γρήγορες και πιο ισχυρές τεχνολογίες που έχει δει ποτέ ο άνθρωπος.

Σύμφωνα με την ρυθμιστική αρχή Ofcom, το 2025 θα χρησιμοποιούμε 13 φορές περισσότερα δεδομένα απ' ό,τι σήμερα. Σήμερα υπάρχουν 7 δισεκατομμύρια συσκευές συνδεδεμένες στο διαδίκτυο παγκοσμίως, ενώ έως και το 2025 αναμένεται οι συσκευές αυτές να εκτοξευθούν στις 21 δισεκατομμύρια. Πολλές από αυτές θα είναι πράγματα που τροφοδοτούν και παρακολουθούν τα σπίτια, τις υποδομές της πόλης, τις μεταφορές και πολλά άλλα περνώντας έτσι στην Internet of Things εποχή.

Η μεγάλη υπόθεση του 5G λοιπόν, είναι το Internet of Things(IoT). Το IoT είναι η ταχέως αναπτυσσόμενη συλλογή «πράξεων» που βασίζονται στο διαδίκτυο για την συλλογή, την κοινή χρήση και την μετάδοση δεδομένων. Καθώς η τεχνολογία προχωρά, όλο και περισσότερα «πράγματα» εντάσσονται στο σύμπαν του IoT. Ο αριθμός των συσκευών IoT ολοένα και αυξάνονται και μεγάλος μέρος αυτής της αύξησης αναμένεται να επιτραπεί από την εκτεταμένη συνδεσιμότητα που θα παρέχουν τελικά τα δίκτυα 5G.

Το 5G ωστόσο, δεν είναι σημαντικό μόνο επειδή έχει την δυνατότητα να υποστηρίξει εκατομμύρια συσκευές σε εξαιρετικά γρήγορες ταχύτητες, αλλά και επειδή έχει την δυνατότητα να μεταμορφώσει και την ζωή των ανθρώπων σε όλο τον κόσμο.

Παρέχει βελτίωση της προσβασιμότητας. Για παράδειγμα σημαντικές προόδους στην τεχνολογία αυτόνομων οχημάτων είναι δυνατές με το 5G, δημιουργώντας την δυνατότητα στους ανθρώπους να έχουν νέα επίπεδα προσωπικής και

⁸ <https://www.verizon.com/about/news/what-4g-lte-and-why-it-matters>

επαγγελματικής ελευθερίας. Οι συνδεδεμένες συσκευές μπορούν να βοηθήσουν και στην αυτοματοποίηση των εργασιών γύρω από το σπίτι, κάτι που μπορεί όχι μόνο να βελτιώσει την προσωπική άνεση αλλά και να βοηθήσει όσους χρειάζονται βοήθεια με τις καθημερινές εργασίες.

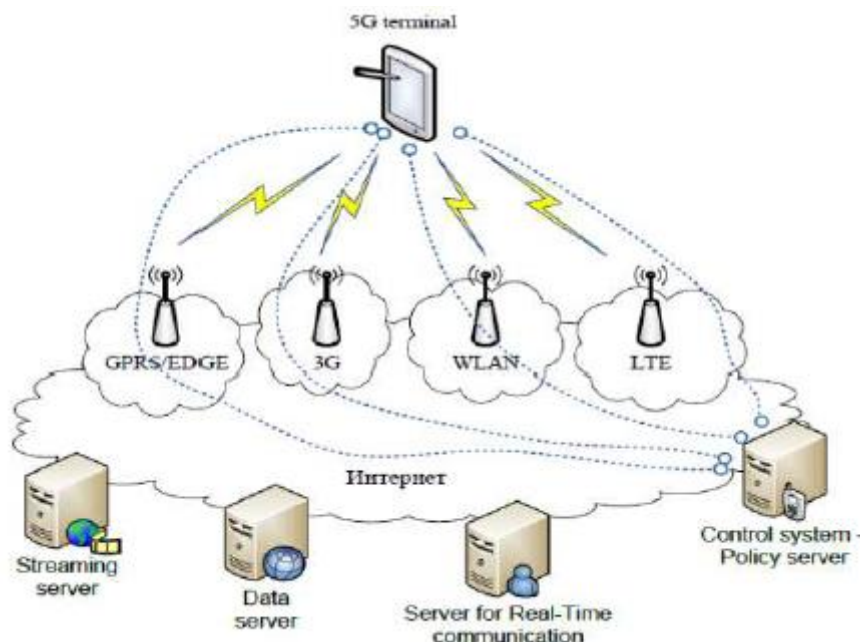
Το 5G ακόμη συμβάλει στην βελτίωση της ασφάλειας, της υγείας και της προστασίας. Η πρόσβαση στο 5G υπόσχεται να βελτιώσει τις υπηρεσίες υψίστης σημασίας που επηρεάζουν την ασφάλεια και την προστασία των υπηρεσιών σήμερα. Οι ευκαιρίες περιλαμβάνουν τις έξυπνες πόλεις, την δυνατότητα για εξ αποστάσεως χειρουργική επέμβαση, καλύτερο έλεγχο της κυκλοφορίας, και πολλές άλλες εφαρμογές που εξαρτώνται από τον σχεδόν στιγμιαίο χρόνο απόκρισης.

2.2 Αρχιτεκτονική 5G

Η αρχιτεκτονική του 5G είναι πολύ προηγμένη, τα στοιχεία δικτύου και τα διάφορα τερματικά του, είναι χαρακτηριστικά αναβαθμισμένα ώστε να μπορούν να αντιμετωπίσουν οποιαδήποτε κατάσταση.

Η δυνατότητα της αναβάθμισης βασίζεται στην τεχνολογία ραδιοπρόσβασης η οποία περιλαμβάνει πολλά σημαντικά χαρακτηριστικά, όπως η ικανότητα των συσκευών να αναγνωρίζουν την γεωγραφική τοποθεσία, τον καιρό, την θερμοκρασία.

Όπως δείχνει η παρακάτω εικόνα, το 5G είναι εξ ολοκλήρου βασισμένο στο IP μοντέλο και σχεδιασμένο για ασύρματα και δίκτυα κινητής τηλεφωνίας.

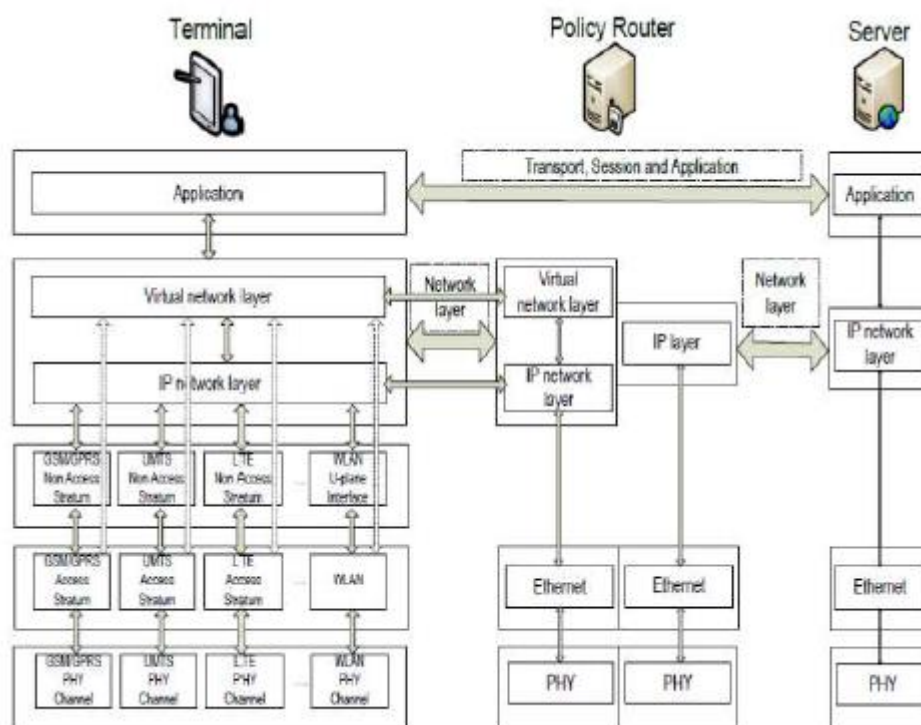


Εικόνα 2.1: Αρχιτεκτονική 5G(https://www.tutorialspoint.com/5g/5g_architecture.htm)

Το σύστημα αποτελείται από ένα κύριο τερματικό χρήστη και έναν αριθμό ανεξάρτητων και αυτόνομων τεχνολογιών ραδιοπρόσβασης. Κάθε μία από τις ραδιοφωνικές τεχνολογίες θεωρείται ως μία σύνδεση IP στον έξω κόσμο του Διαδικτύου. Η τεχνολογία IP έχει σχεδιαστεί αποκλειστικά για να διασφαλίζει επαρκή δεδομένα ελέγχου για την κατάλληλη δρομολόγηση πακέτων IP που

σχετίζονται με συγκεκριμένες συνδέσεις εφαρμογής, πχ συνεδρίες μεταξύ εφαρμογών πελάτη και διακομιστών σε κάποιο σημείο του Διαδικτύου.

Για να γίνει προσβάσιμη η δρομολόγηση των πακέτων, θα πρέπει να καθοριστεί σύμφωνα με τις δεδομένες πολιτικές του χρήστη, όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 2.2: Πολιτικές δρομολόγησης πακέτων(https://www.tutorialspoint.com/5g/5g_architecture.htm)

2.2 Βασικά πλεονεκτήματα 5G

Τα 5G δίκτυα διαθέτουν χαμηλότερη καθυστέρηση, μεγαλύτερη χωρητικότητα και αυξημένο εύρος ζώνης σε σύγκριση με το 4G. Αυτές οι βελτιώσεις δικτύου θα έχουν εκτεταμένες επιπτώσεις στον τρόπο ζωής, εργασίας και παιχνιδιού των ανθρώπων σε όλο τον κόσμο.

Παρακάτω αναλύονται τα βασικότερα πλεονεκτήματα του 5G.

Αναβαθμίσεις Ταχύτητας

Οι προβλεπόμενες ταχύτητες έως και 10Gbps⁹ αντιπροσωπεύουν έως και 100x¹⁰ αύξηση σε σύγκριση με αυτές στο 4G. Οι βελτιώσεις στην ταχύτητα θα οδηγήσουν σε συναρπαστικές δυνατότητες για τους χρήστες. Η μεταφορά για παράδειγμα μίας ταινίας HD σε μέγιστες ταχύτητες λήψης θα φτάσεις από επτά

⁹ <https://www.intel.com/content/www/us/en/wireless-network/5g-benefits-features.html>

¹⁰ <https://www.intel.com/content/www/us/en/wireless-network/5g-benefits-features.html>

λεπτά σε μόλις έξι δευτερόλεπτα. Αυτό συνεπάγεται και σε τεράστια εξοικονόμηση χρόνου.

Ακόμη, τα δίκτυα 5G θα μπορούν να δώσουν μια ισχυρά εναλλακτική λύση για γρήγορες ευρυζωνικές συνδέσεις, ανάμεσα σε επιχειρήσεις.

Χαμηλή Καθυστέρηση

Ένας από τους στόχους κάθε ασύρματης γενιάς, ήταν η μείωση της καθυστέρησης από την στιγμή που ένα σήμα θα μεταβεί από την πηγή του στον δέκτη, και μετά πάλι πίσω. Τα 5G έχουν ακόμη χαμηλότερη καθυστέρηση από του 4G, με την μετ' επιστροφής μετάδοση δεδομένων να διαρκεί λιγότερο από 5 χιλιοστά του δευτερολέπτου.

Ο λανθάνοντας χρόνος 5G θα είναι ταχύτερος από την ανθρώπινη οπτική επεξεργασία, καθιστώντας δυνατό τον έλεγχο συσκευών εξ αποστάσεως σε σχεδόν πραγματικό χρόνο. Η ταχύτητα ανθρώπινης αντίδρασης θα γίνει ο περιοριστικός παράγοντας για απομακρυσμένες εφαρμογές που χρησιμοποιούν 5G και IoT—και πολλές νέες εφαρμογές θα περιλαμβάνουν επικοινωνία μηχανής με μηχανή που δεν περιορίζεται από το πόσο γρήγορα μπορούν να ανταποκριθούν οι άνθρωποι.

Ενισχυμένη Χωρητικότητα

Το 5G προσφέρει έως και 1000 φορές¹¹ περισσότερη χωρητικότητα από το 4G, δημιουργώντας πρόσφορο έδαφος για ανάπτυξη IoT. Το 5G και το IoT ταιριάζουν απόλυτα με στόχο να επαναπροσδιορίσουν τον τρόπο χρήσης ασύρματων δικτύων, καθώς και του διαδικτύου στο σύνολό του. Με χωρητικότητα για εκατοντάδες συσκευές που επικοινωνούν μεταξύ τους, νέες εφαρμογές σε κάθε κλάδο, θα ανθίσουν.

Τα έξυπνα σπίτια και οι πόλεις, θα κάνουν ένα τεράστιο άλμα προς τα εμπρός στο μέλλον του 5G. Η τεχνητή νοημοσύνη θα μεταφερθεί σε μέρη που δεν είχε ξαναβρεθεί με υπολογιστές αιχμής.

Αυξημένος Εύρος Ζώνης

Ο συνδυασμός αυξημένης ταχύτητας και χωρητικότητας δικτύου στα δίκτυα 5G θα δημιουργήσει τη δυνατότητα για μετάδοση μεγαλύτερων ποσοτήτων δεδομένων από ό,τι ήταν δυνατό με τα δίκτυα 4G LTE.

Τα δίκτυα 5G έχουν σχεδιαστεί διαφορετικά από τα παραδοσιακά δίκτυα 4G, επιτρέποντας μεγαλύτερη βελτιστοποίηση της κυκλοφορίας του δικτύου και ομαλό χειρισμό των αιχμών χρήσης.

¹¹ <https://www.intel.com/content/www/us/en/wireless-network/5g-benefits-features.html>

3. Network Slicing

3.1 Ορισμός Network Slicing

Το network slicing είναι μία μέθοδος δημιουργίας πολλαπλών λογικών και εικονικών δικτύων μέσω μιας κοινής υποδομής πολλαπλών τομέων. Χρησιμοποιώντας την δικτύωση η οποία καθορίζεται από λογισμικό SDN (θα αναλυθεί στο επόμενο κεφάλαιο), την εικονικοποίηση λειτουργιών δικτύου NFV (θα αναλυθεί παρακάτω), την ενορχήστρωση, τα αναλυτικά στοιχεία και τον αυτοματισμό, οι χειριστές δικτύων κινητής τηλεφωνίας έχουν την δυνατότητα να δημιουργήσουν γρήγορα slices δικτύου τα οποία θα μπορούν να υποστηρίξουν μια συγκεκριμένη εφαρμογή, υπηρεσία, σύνολο χρηστών ή δίκτυο. Τα slices δικτύου, μπορούν να εκτείνονται σε πολλούς τομείς δικτύου, συμπεριλαμβανομένης της πρόσβασης, του πυρήνα και της μεταφοράς, και να αναπτυχθούν σε πολλούς χειριστές.

Το Network Slicing λοιπόν, είναι μία από τις πιο σημαντικές τεχνολογίες στο 5G. Υποστηρίζει νέες υπηρεσίες με πολύ διαφορετικές απαιτήσεις, από ένα συνδεδεμένο όχημα μέχρι μια φωνητική κλήση, οι οποίες απαιτούν διαφορετική απόδοση, καθυστέρηση και αξιοπιστία.

Τεμαχίζοντας το δίκτυο, δημιουργούνται slices τα οποία έχουν την δική τους αρχιτεκτονική, διαχείριση και ασφάλεια για την υποστήριξη μίας συγκεκριμένης περίπτωσης χρήσης.

Οι απαιτήσεις που έχει ένας χρήστης ενός δικτύου πολλές φορές είναι αρκετά διαφορετικές σε σχέση με κάποιον άλλον. Ο τεμαχισμός στο δίκτυο, δίνει μία λύση καθώς είναι μια υπηρεσία η οποία προσαρμόζει το δίκτυο (το slice) στις απαιτήσεις του χρήστη.

Για να γίνει πιο κατανοητή η αναγκαιότητα και η σημασία του network slicing, δίνεται το παρακάτω παράδειγμα. Έστω ότι σε ένα δίκτυο υπάρχουν 4 χρήστες. Ο ένας θέλει να στείλει ένα βίντεο υψηλής ανάλυσης, ο δεύτερος θέλει να κατεβάσει μία ταινία, ο τρίτος να κάνει ένα live streaming, και ο τέταρτος θέλει να κάνει βιντεοκλήση. Χωρίς τον τεμαχισμό του δικτύου, θα υπήρχε μεγάλη καθυστέρηση ανάμεσα σε κάθε ενέργεια, και η λειτουργία του δικτύου δεν θα ήταν όσο αποδοτική θα έπρεπε. Τεμαχίζοντας το δίκτυο, ένα slice θα παρέχει την υπηρεσία του πρώτου χρήστη, ένα δεύτερο slice την υπηρεσία του δεύτερου χρήστη κ.ο.κ. Έτσι λοιπόν, δημιουργείται ένα slice για κάθε λειτουργία, με αποτέλεσμα η λειτουργία του ενός να μην επηρεάζει την απόδοση του άλλου. Κάθε slice ουσιαστικά είναι ξεχωριστό από το άλλο.

Επομένως, μπορεί να υπάρχει μεγάλη ποικιλία στις απαιτήσεις των χρηστών, ωστόσο με το network slicing μπορούν να εξυπηρετηθούν όλοι αποδοτικά.

3.2 Πλεονεκτήματα στον Τεμαχισμού Δικτύου

Όπως αναφέρθηκε και παραπάνω, το network slicing είναι μία πολύ σημαντική τεχνολογία για το 5G. Ποια όμως είναι τα πλεονεκτήματα και το όφελος του τεμαχισμού δικτύου; Παρακάτω γίνεται μία αναφορά στα πλεονεκτήματα.

- Ένα ενιαίο δίκτυο μπορεί να χρησιμοποιηθεί για να προσφέρει διάφορες υπηρεσίες με βάση τις απαιτήσεις του χρήστη και διάφορες περιπτώσεις χρήσης.

- Οι χειριστές του δικτύου μπορούν να καταναείμουν τη σωστή ποσότητα των απαιτούμενων πόρων ανάλογα με το slice δικτύου. Ως εκ τούτου, βοηθά στην αποτελεσματική και αποδοτική χρήση των πόρων. Ένα slice δικτύου, μπορεί να σχεδιαστεί για να παρέχει χαμηλό λανθάνοντα χρόνο και χαμηλό ρυθμό δεδομένων, ενώ το άλλο slice μπορεί να διαμορφωθεί ώστε να παρέχει υψηλή απόδοση.
- Βελτιώνει κατά πολύ την λειτουργική αποτελεσματικότητα και τον χρόνο διάθεσης στην αγορά για την παροχή υπηρεσιών δικτύου 5G.
- Ξεπερνά όλα τα μειονεκτήματα του “DiffServ”(μοντέλο για την παροχή QoS στο διαδίκτυο, διαφοροποιώντας την κίνηση) που είναι η πιο δημοφιλής λύση στο QoS.
- Μέσω διαφόρων μηχανισμών που διαθέτει η τεχνική τεμαχισμού δικτύου, διαμελίζοντας το δίκτυο σε πολλαπλά αυτόνομα slices έχει ως όφελος ένα πρόβλημα σε ένα συγκεκριμένο slice δικτύου, να μην επηρεάσει τα υπόλοιπα slices μέσα στο δίκτυο. Έτσι υπάρχει η απαιτούμενη προστασία ανάμεσα στα slices μέσα στο δίκτυο.
- Μία από τις κυριότερες ικανότητες του τεμαχισμού δικτύου είναι η επεκτασιμότητα, καθώς υπάρχει η δυνατότητα να προστεθούν επιπλέον λειτουργίες και χαρακτηριστικά σε ένα slice δικτύου μέσω μιας αλλαγής της υπάρχουσας λειτουργίας του εικονικού δικτύου, με ελάχιστες πιθανότητες να επηρεαστεί οποιαδήποτε υπάρχουσα λειτουργία του παρών δικτύου. Αποτέλεσμα αυτού είναι να αναπτύσσεται το δίκτυο με γρήγορους ρυθμούς με μηδαμινά εμπόδια.

3.3 Προκλήσεις

3.3.1 Κοινή χρήση πόρων.

Τα δίκτυα κινητής τηλεφωνίας έχουν διαπιστώσει ότι η κατανομή των πόρων μεταξύ των slices αποτελεί βασικό ζήτημα. Η κοινή χρήση πόρων μπορεί να υλοποιηθεί με στατική κοινή χρήση κατάτμησης ή ελαστικά δυναμική κοινή χρήση. Λόγω των δυναμικών χαρακτηριστικών του φόρτου δικτύου, η δυναμική κατανομή των πόρων μεταξύ των slices καθιστά την χρήση των πόρων πιο αποτελεσματική. Υπάρχουν ωστόσο ορισμένα συγκεκριμένα ζητήματα που πρέπει να επιλυθούν όσον αφορά την κατανομή των πόρων. Για παράδειγμα τα ασύρματα κανάλια επικοινωνίας μπορούν να μοιραστούν μεταξύ των τμημάτων του ασύρματου δικτύου πρόσβασης(Radio Access Network RAN). Απαιτείται επομένως ένας καλός μηχανισμός διαχείρισης για την κατανομή των ραδιοφωνικών πόρων μεταξύ των άλλων τμημάτων. Επιπλέον, πρέπει να εξεταστεί το ενδεχόμενο επιμερισμού των υπολογιστικών πόρων και άλλων πόρων. Παρόλο που η κατανομή των πόρων αποφέρει οφέλη στους παρόχους υποδομών, δημιουργεί επίσης περισσότερες προκλήσεις σε άλλα προβλήματα, όπως η απομόνωση των slice.

3.3.2 Δημιουργία και διαχείριση δυναμικών τμημάτων

Μία από τις ανησυχίες για τους μελλοντικούς φορείς εκμετάλλευσης δικτύων 5G είναι ο τρόπος κατανομής των πόρων του δικτύου για την μεγιστοποίηση των οφελών τους. Στο πλαίσιο αυτό, η διαχείριση του κύκλου ζωής των slices του δικτύου είναι ένα κρίσιμο πρόβλημα που πρέπει να επιλυθεί. Προκειμένου να καλυφθεί ο μέγιστος αριθμός διαφοροποιημένων αιτημάτων παροχής υπηρεσιών, οι φορείς εκμετάλλευσης δικτύων 5G πρέπει να αναπτύξουν εικονικές λειτουργίες

δικτύου και να καταναείμουν πόρους δικτύου γρήγορα για να δημιουργηθούν τα slices δικτύου(network slices). Επιπλέον θα πρέπει να είναι σε θέση να κλιμακώνουν τα slice δυναμικά σύμφωνα με το μεταβλητό φορτίο υπηρεσίας.

Αντίστοιχα, η τεχνική του network slicing θα πρέπει να εξετάζει τον τρόπο ανοίγματος μερικών αδειών σε κάθε slice για να το διαμορφώσει και να το διαχειριστεί χωρίς να εγείρει ζητήματα ασφάλειας. Τέλος, η διαχείριση του τεμαχισμού του δικτύου, θα πρέπει να εφαρμόζεται με αυτοματοποιημένο τρόπο, ώστε να αποφεύγονται οι χειροκίνητες προσπάθειες και τα σφάλματα.

3.3.3 Απομόνωση μεταξύ τμημάτων δικτύου

Κάθε υπηρεσία στο δίκτυο 5G, έχει και την δική της μοναδική απαίτηση. Επομένως οι απαιτήσεις από υπηρεσία σε υπηρεσία, μπορεί να διαφέρουν. Κατά συνέπεια, απαιτούνται αποκλειστικοί πόροι εικονικού δικτύου για την διασφάλιση της ποιότητας των υπηρεσιών σε κάθε slice. Αυτό, απαιτεί τα slice να είναι απομονωμένα το ένα από το άλλο.

Η απομόνωση μεταξύ των τμημάτων δικτύου, μπορεί να επιτευχθεί μέσω απομόνωσης επιπέδου δεδομένων και απομόνωσης επιπέδου ελέγχου. Σε γενικές γραμμές, η λειτουργία ελέγχου του slice μπορεί να μοιραστεί μεταξύ των slices, ενώ για ορισμένες υπηρεσίες, όπως οι επικοινωνίες κρίσιμης σημασίας για την αποστολής, το slice χρειάζεται την δική του λειτουργία ελέγχου.

Επιπλέον, η αποτελεσματική απομόνωση των τμημάτων δικτύου μπορεί να διασφαλίσει ότι η αστοχία ή η επίθεση ασφάλειας σε ένα slice δεν επηρεάζει την λειτουργία των άλλων slice. Ως εκ τούτου, ο μηχανισμός απομόνωσης των slices αποτελεί βασική πρόκληση για την υλοποίηση του τεμαχισμού δικτύου.

3.3.4 Διαχείριση της κινητικότητας κατά το network slicing

Οι πτυχές της κινητικότητας, όπως η απρόσκοπτη μεταβίβαση και διαχείριση των παρεμβολών, δημιουργούν νέες προκλήσεις για τον τεμαχισμό δικτύων. Η ταχεία μεταβίβαση της κινητικότητας είναι ζωτικής σημασίας για τις υπηρεσίες σε πραγματικό χρόνο, οι οποίες έχουν άμεση επίδραση στην ποιότητα των υπηρεσιών. Η υποστήριξη της κινητικότητας για το network slicing είναι προαιρετική για ορισμένα συγκεκριμένα network slices. Για παράδειγμα, τα slices δικτύου που εξυπηρετούν βιομηχανικό έλεγχο δεν χρειάζονται λειτουργίες διαχείρισης της κινητικότητας λόγω της σταθερής θέσης των συσκευών.

Ωστόσο, τα τμήματα δικτύου που χρειάζονται διαχείριση της κινητικότητας διαφέρουν ως προς τις απαιτήσεις κινητικότητας. Για παράδειγμα, η απαίτηση κινητικότητας για το τμήμα κινητών ευρυζωνικών συνδέσεων διαφέρει από το τμήμα για τις αυτοματοποιημένες υπηρεσίες οδήγησης.

Ως εκ τούτου, ο σχεδιασμός ενός πρωτοκόλλου διαχείρισης της κινητικότητας με προσανατολισμό σε slice είναι απαραίτητος για την αντιμετώπιση των προκλήσεων της κινητικότητας κατά τον τεμαχισμό δικτύου.

3.3.5 Ασφάλεια κατά το network slicing

Η ασφάλεια κατά τον τεμαχισμό δικτύου είναι ένα κρίσιμο πρόβλημα που πρέπει να αντιμετωπιστεί λόγω της κατανομής των πόρων μεταξύ των slices. Τα slices δικτύου που εξυπηρετούν διαφορετικούς τύπους υπηρεσιών ενδέχεται να έχουν διαφορετικά επίπεδα απαιτήσεων πολιτικής ασφάλειας. Επομένως, κατά τον σχεδιασμό των πρωτοκόλλων ασφάλειας για τον τεμαχισμό του δικτύου, είναι απαραίτητο να ληφθεί υπόψη ο αντίκτυπος σε άλλα slice και στο σύνολο όλου του συστήματος του δικτύου.

Επιπλέον, τα ζητήματα ασφάλειας γίνονται πιο περίπλοκα όταν εφαρμόζεται το network slicing στην υποδομή πολλαπλών τομέων. Πρέπει να σχεδιαστούν μηχανισμοί συντονισμού της πολιτικής ασφάλειας μεταξύ διαφόρων τομεακών υποδομών.

3.3.6 Εικονικοποίηση πόρων

Η εικονικοποίηση είναι η πιο σημαντική τεχνολογία για την εφαρμογή του network slicing. Η τεχνολογία εικονικοποίησης έχει εξελιχθεί τα τελευταία 20 χρόνια και έχει εφαρμοστεί κυρίως σε ενσύρματα δίκτυα. Πολλή δουλειά έχει επικεντρωθεί σε εικονικοποίηση δικτύων. Ωστόσο, η ασύρματη επικοινωνία είναι πιο περίπλοκη, καθώς οι ασύρματες συνδέσεις ποικίλλουν χρονικά και είναι ευάλωτες σε παρεμβολές.

Ορισμένες από τις μεθόδους εικονικοποίησης σε ενσύρματο δίκτυο δεν μπορούν να χρησιμοποιηθούν άμεσα για τους ασύρματους ομολόγους τους. Ως εκ τούτου, ο σχεδιασμός νέων μηχανισμών εικονικοποίησης για την επίτευξη της κοινής χρήσης των πόρων ασύρματου φάσματος και της εικονικοποίησης των σταθμών βάσης διαδραματίζει ζωτικό ρόλο στην εφαρμογή του τεμαχισμού RAN.

3.3.7 Αλγοριθμικές πτυχές της κατανομής των πόρων

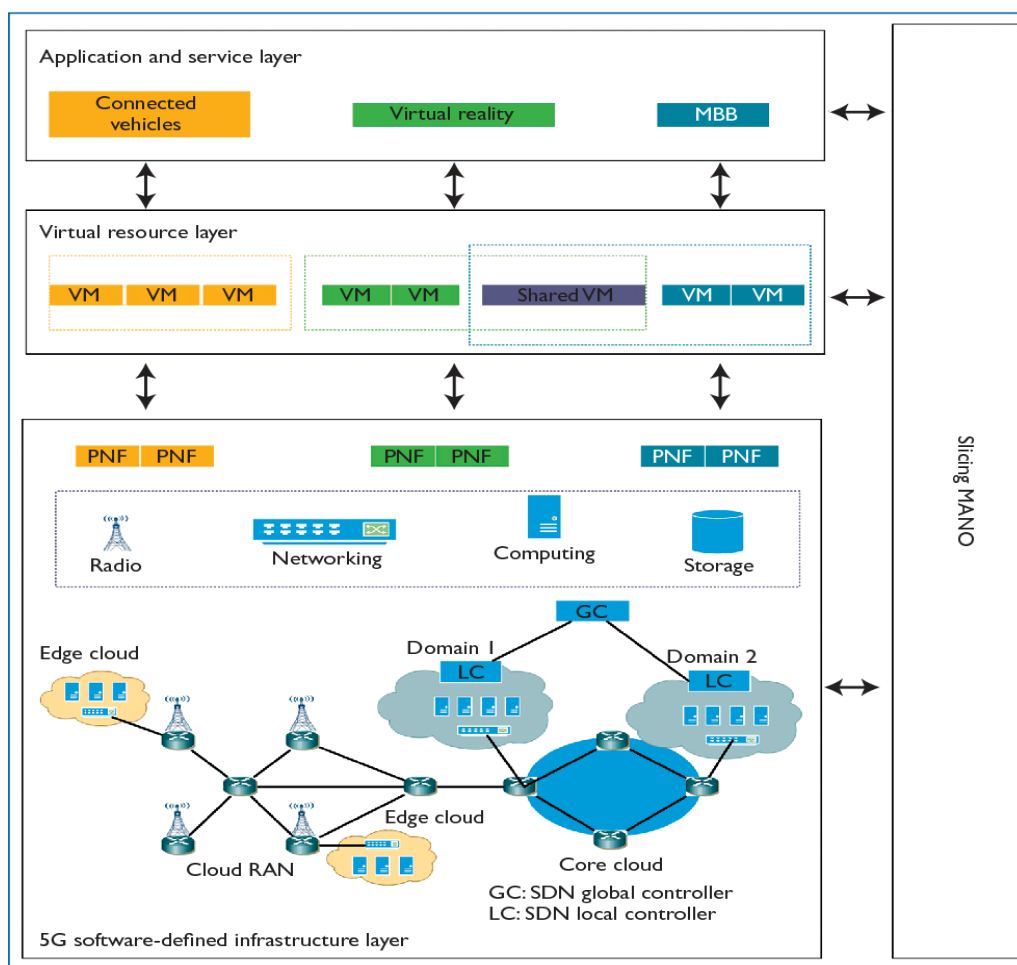
Ο τρόπος αποτελεσματικής κατανομής των πόρων σε slice είναι ένα δύσκολο πρόβλημα που πρέπει να αντιμετωπιστεί. Το πρόβλημα κατανομής πόρων τομής, είναι παρόμοιο με το πρόβλημα ενσωμάτωσης εικονικού δικτύου. Η διαμόρφωση αυτού του προβλήματος μπορεί να χρησιμοποιήσει μαθηματικές μεθόδους στην επιχειρησιακή έρευνα όπως ο ακέραιος γραμμικός προγραμματισμός.

Οι αλγόριθμοι για την επίλυση του προβλήματος μπορεί να υιοθετήσουν διαφορετικές στρατηγικές ανάλογα με το μέγεθος ενός slice, δηλαδή τον αριθμό των εικονικών συνδέσμων και των εικονικών κόμβων.

Η κατανομή των πόρων για slice μικρής κλίμακας μπορεί να επιλυθεί με ακριβείς αλγορίθμους, ενώ για τα slice μεγάλης κλίμακας οι ευρετικές ή μετα-ευρετικές στρατηγικές είναι πιο αποτελεσματικές. Αξίζει να σημειωθεί ότι, λόγω του δυναμικού χαρακτηριστικού του δικτύου 5G, πρέπει να σχεδιάσουμε πρακτικούς αλγορίθμους κατανομής πόρων με την ικανότητα να αναδιαμορφώνουμε και να μεταναστεύουμε πόρους σε slice.

3.4 Network slicing framework

Παρακάτω θα παρουσιαστεί το πλαίσιο κοπής δικτύου (network slicing framework) τριών επιπέδων για τα δίκτυα 5G όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 3.1: Network Slicing Framework 3
 επιπέδων (https://www.cse.wustl.edu/~jain/papers/ftp/slic_ic.pdf)

Το προτεινόμενο framework υποστηρίζει την ευέλικτη ανάπτυξη και διαχείριση ποικίλων εφαρμογών δικτύου σε μία κοινή υποδομή. Το framework περιέχει επίπεδο υποδομής (5G-SDI) που ορίζεται από λογισμικό 5G, εικονικό επίπεδο πόρων, επίπεδο εφαρμογής και υπηρεσίας και λειτουργικό στοιχείο MANO τεμαχισμού.

Στην εικόνα, φαίνονται τρία διαφορετικά χρώματα τα οποία υποδεικνύουν τρία slice δικτύου για τα συνδεδεμένα οχήματα, την εικονική πραγματικότητα και την κινητή ευρυζωνικότητα (Mobile Broadband - MBB). Το κατώτερο στρώμα είναι το στρώμα υποδομής 5G, το οποίο αποτελείται από υποδομές καθοριζόμενες από λογισμικό 5G πολλαπλών τομέων με έλεγχο και διαχείριση βάσει λογισμικού.

Η χρήση της προσέγγισης που καθορίζεται από το λογισμικό αποφέρει τα ακόλουθα οφέλη στο framework: 1) Παρέχει άντληση πόρων και επιτρέπει την δυναμική κατανομή των πόρων. 2) Η εικονικοποίηση των πόρων υποδομής μπορεί να διαμορφώσει με ευελιξία συγκεκριμένα για κάθε υπηρεσία slice. 3)

Τόσο ο πάροχος του δικτύου όσο και ο χρήστης του slice μπορούν να διαχειρίζονται εικονικούς πόρους.

Ο σχεδιασμός του SDN controller στο framework χρησιμοποιεί την ιεραρχική μέθοδο. Κάθε domain διαθέτει έναν τοπικό ελεγκτή και αυτοί οι τοπικοί ελεγκτές συγκλίνουν στον κεντρικό global controller. Μερικά παραδείγματα των ελεγκτών 5G-SDI είναι το OpenDaylight, Flowvisor, FoodLight, και το ONOS.

Τεχνολογίες που βασίζονται στο υπολογιστικό νέφος, όπως το cloud RAN και το MEC, μπορούν να εισαχθούν ώστε να καταστεί δυνατή η δημιουργία υποδομής 5G στο νέφος. Οι εικονικές λειτουργίες δικτύου(NFV) χαμηλού πλάτους όπως το slice για συνδεδεμένα οχήματα μπορούν να τοποθετηθούν σε σύννεφα άκρων (edge clouds), ενώ οι NFV σε slice MBB μπορούν να τροφοδοτηθούν σε διακομιστές εμπορευμάτων στο κεντρικό cloud.

Το επίπεδο εικονικών πόρων παρέχει όλους του εικονικούς πόρους που απαιτούνται για τις τομές δικτύου, όπως το ραδιόφωνο, η πληροφορική, η αποθήκευση, το εύρος ζώνης δικτύου και άλλα. Οι εικονικοί πόροι και τα NFV λειτουργούν ως εικονική μηχανή (VM).

Αυτοί οι πόροι είναι στην πραγματικότητα μια αφαίρεση των υποκείμενων φυσικών πόρων που διαχειρίζονται στο 5G-SDI και μπορούν να μοιραστούν μεταξύ slices, για παράδειγμα, το παράδειγμα VM μοιράζεται μεταξύ του τμήματος εικονικής πραγματικότητας και του τμήματος MBB στην εικόνα. Επιπλέον, αυτοί οι πόροι μπορούν να χρησιμοποιηθούν κατά παραγγελία από τμήματα δικτύου, γεγονός που ενισχύει την χρήση πόρων δικτύου.

Οι ετερογενείς περιπτώσεις σέρβις, όπως τα συνδεδεμένα οχήματα, η εικονική πραγματικότητα και το MBB, αποτελούν το επίπεδο υπηρεσίας. Αυτές οι περιπτώσεις σέρβις ανήκουν σε πολλούς ενοικιαστές. Ωστόσο, ένας ενοικιαστής μπορεί ακόμη να έχει διαφόρους τύπους υπηρεσιών.

Η παρουσία σέρβις ενημερώνει την MANO για τις απαιτήσεις σέρβις, οι οποίες στην συνέχεια αντιστοιχίζονται στο αντίστοιχο slice δικτύου. Ο δυναμικός χαρακτήρας των υπηρεσιών επιβάλλει την κλιμάκωση των τμημάτων του δικτύου κατά την διάρκεια της εκτέλεσης. Οι υπηρεσίες που έχουν παρόμοιες απαιτήσεις μπορεί να μοιράζονται ένα slice, και οι υπηρεσίες με διαφορετικές απαιτήσεις υπηρεσιών μπορούν να μοιράζονται εν μέρει τα slices.

Η λειτουργική συνιστώσα διαχείρισης και ενορχήστρωσης του τεμαχισμού (MANO) αποτελεί βασικό μέρος του πλαισίου τεμαχισμού δικτύου. Διαχειρίζεται και ενορχηστρώνει τους φυσικούς πόρους, τους εικονικούς πόρους και τις τομές δικτύου με τρόπο που θα ενσωματωθεί το NFV-MANO. Το slicing MANO μπορεί να έχει δευτερεύουσες λειτουργίες όπως διαχειριστής υποδομής, διαχειριστής NFV και ενορχηστρωτής slice. Τα καθήκοντα του τεμαχισμού MANO περιλαμβάνουν: 1)Την δημιουργία και την διαχείριση στιγμιότυπων VM με τη χρήση πόρων υποδομής. 2)Χαρτογράφηση λειτουργιών δικτύου σε εικονικούς πόρους και σύνδεση λειτουργιών δικτύου για την δημιουργία αλυσίδων υπηρεσιών. 3)Διαχείριση του κύκλου ζωής των τμημάτων δικτύου με αλληλεπίδραση με το επίπεδο εφαρμογής και υπηρεσίας.

Στην πραγματικότητα η διαχείριση των τμημάτων δικτύου μπορεί να πραγματοποιείται σε δύο επίπεδα, τα οποία είναι η διαχείριση μεταξύ των

τμημάτων και η διαχείριση εντός των τμημάτων. Η διαχείριση μεταξύ των slice εκτελείται από τον ενορχηστρωτή των slice, ενώ η διαχείριση εντός των slice υλοποιείται εκτελώντας μια λειτουργία εικονικού διαχειριστή ως μέρος της τομής.

4. Software Defined Networks

4.1 Ορισμός SDN

Το SDN είναι μια προσέγγιση αρχιτεκτονικής δικτύου που επιτρέπει στο δίκτυο, να ελέγχεται έξυπνα και κεντρικά ή να προγραμματίζεται χρησιμοποιώντας εφαρμογές λογισμικού. Αυτό, βοηθά τους χειριστές να διαχειρίζονται ολόκληρο το δίκτυο με συνέπεια και ολιστική, ανεξάρτητα από την υποκείμενη τεχνολογία λογισμικού.

Το SDN επιτρέπει τον προγραμματισμό της συμπεριφοράς του δικτύου με κεντρικά ελεγχόμενο τρόπο μέσω εφαρμογών λογισμικού που χρησιμοποιούν ανοιχτά API. Ανοίγοντας παραδοσιακά κλειστές πλατφόρμες δικτύου και εφαρμόζοντας ένα κοινό επίπεδο ελέγχου SDN, οι χειριστές μπορούν να διαχειρίζονται ολόκληρο το δίκτυο και τις συσκευές του με συνέπεια, ανεξάρτητα από την πολυπλοκότητα της υποκείμενης τεχνολογίας δικτύου.

Η δικτύωση λοιπόν οριζόμενη από το λογισμικό (SDN) μεταφέρει τις περισσότερες από τις λειτουργίες του επιπέδου ελέγχου σε έναν σχετικό ελεγκτή(controller), ο οποίος βρίσκεται έξω από το στοιχείο. Για να μπορεί ο εξωτερικός controller να τροποποιεί και να εποπτεύει το στοιχείο δικτύου, προστίθεται μια επιπλέον υπομονάδα ελέγχου. Η υπομονάδα, αντί να παρέχει μια εξωτερική διασύνδεση για ανθρώπους-διαχειριστές, ιδέα είναι να επιτρέπεται στο λογισμικό διαχείρισης που εκτελείται στον εξωτερικό ελεγκτή να διευθετεί πίνακες προώθησης στο επίπεδο δεδομένων.

4.2 Η αρχιτεκτονική SDN

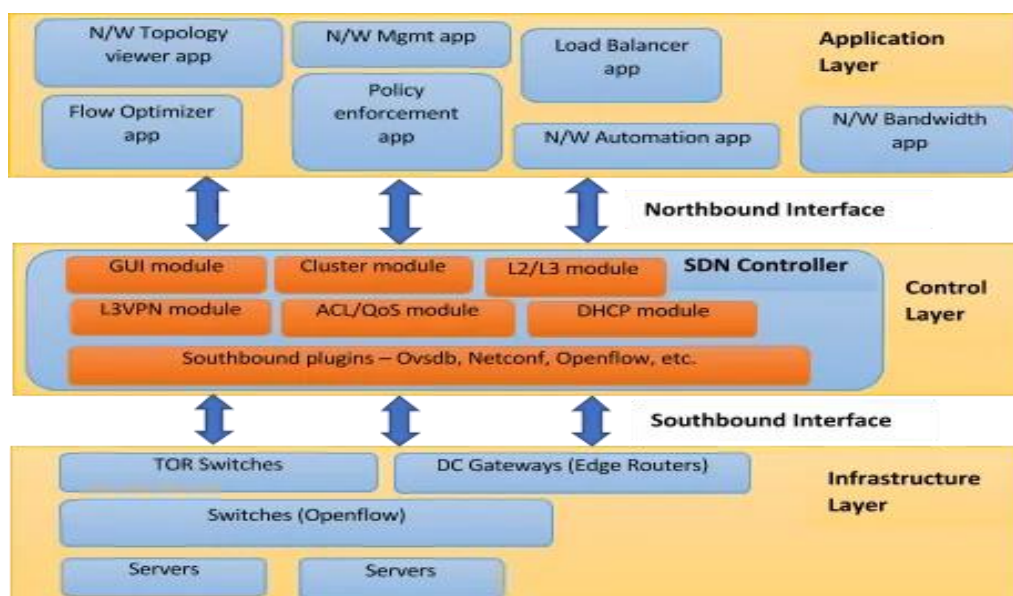
Η αρχιτεκτονική του SDN, μπορεί να αναγνωριστεί σε τέσσερα βασικά χαρακτηριστικά.

- Βασιζόμενη σε ροή προώθησης. Η προώθηση πακέτων από μεταγωγείς που ελέγχονται από SDN μπορεί να βασίζεται σε οποιοδήποτε αριθμό τιμών πεδίου κεφαλίδας, που βρίσκονται μέσα στην κεφαλίδα επιπέδου μεταφοράς, επιπέδου δικτύου ή επιπέδου ζεύξης. Αυτό έρχεται σε αντίθεση με την παραδοσιακή προώθηση που επιτυγχάνονταν μέσω δρομολογητών προώθησης κατά την οποία η προώθηση βασιζόταν αποκλειστικά στην διεύθυνση προορισμού IP ενός δεδομενογράμματος. Τώρα είναι δουλειά του SDN επιπέδου ελέγχου να υπολογίζει, να διαχειρίζεται και να εγκαθιστά καταχωρίσεις πίνακα ροής σε όλους τους μεταγωγείς του δικτύου.
- Διαχωρισμός επιπέδου δεδομένων και επιπέδου ελέγχου. Το επίπεδο δεδομένων αποτελείται απ' τους μεταγωγείς δικτύου, που εκτελούν τους κανόνες ταιριάσματος και ενέργειας μέσα στους πίνακες ροής τους. Το επίπεδο ελέγχου αποτελείται από εξυπηρετητές και λογισμικό, που καθορίζουν και διαχειρίζονται τους πίνακες ροής των μεταγωγέων.
- Λειτουργίες ελέγχου δικτύου. Το SDN(Software defined network) υλοποιείται με λογισμικό. Το λογισμικό αυτό εκτελείται σε εξυπηρετητές, που είναι διακριτοί και

απομακρυσμένοι απ' τους μεταγωγείς του δικτύου. Το επίπεδο ελέγχου, αποτελείται από δύο συστατικά. Από έναν ελεγκτή SDN και ένα σύνολο ελεγχόμενων από το δίκτυο εφαρμογών. Ο controller, διατηρεί ακριβείς πληροφορίες κατάστασης δικτύου τις οποίες παρέχει στις εφαρμογές ελέγχου δικτύου, που εκτελούνται στο επίπεδο ελέγχου, καθώς επίσης παρέχει και τον τρόπο μέσω του οποίου αυτές οι εφαρμογές μπορούν να παρακολουθούν, να προγραμματίζουν και να ελέγχουν τις συσκευές του υποκείμενου δικτύου.

- Ένα προγραμματιζόμενο δίκτυο. Το δίκτυο προγραμματίζεται μέσω των εφαρμογών ελέγχου δικτύου, που εκτελούνται στο επίπεδο ελέγχου. Οι εφαρμογές αυτές αντιπροσωπεύουν τον εγκέφαλο του SDN, χρησιμοποιώντας τις API, που παρέχονται απ' τον ελεγκτή SDN, για να καθορίσουν και να ελέγξουν το επίπεδο δεδομένων σε συσκευές δικτύου.

Βάση των παραπάνω χαρακτηριστικών, είναι προφανές ότι το SDN αποτελεί ένα σημαντικό διαχωριστή της λειτουργικότητας του δικτύου. Οι μεταγωγείς επιπέδου δεδομένων, οι SDN controller, και οι εφαρμογές ελέγχου δικτύου είναι ξεχωριστές οντότητες, που μπορούν να παρέχονται από διαφορετικούς προμηθευτές και οργανισμούς.



Εικόνα 4.1: Αρχιτεκτονική SDN(<https://www.howtoforge.com/tutorial/software-defined-networking-sdn-architecture-and-role-of-openflow/>)

4.3 Επικοινωνία SDN

Ένα σύστημα SDN πρέπει να χειρίζεται δύο νέα πρωτόκολλα επικοινωνίας. 1)Επικοινωνία controller-στοιχείου. 2)Επικοινωνία μεταξύ controllers.

Το SDN ακολουθεί το συνολικό υπόδειγμα που εφαρμόστηκε για πρώτη φορά από το SNMP. Στο SDN, τα πρωτόκολλα διαχείρισης λειτουργούν στο επίπεδο Εφαρμογών. Με λίγα λόγια, το SDN υποθέτει ότι κάθε εξωτερικός controller και κάθε στοιχείο δικτύου διαθέτουν μια συμβατική στοίβα TCP/IP την οποία μπορεί να χρησιμοποιεί το λογισμικό διαχείρισης για να επικοινωνεί με στοιχεία δικτύου στην περιοχή SDN, καθώς και με άλλους SDN controllers. Το SDN υιοθετεί επίσης την προσέγγιση SNMP όσον αφορά τη φυσική συνδεσιμότητα. Αντί να χρησιμοποιεί ένα ξεχωριστό δίκτυο ελέγχου για να μεταφέρει την κυκλοφορία

διαχείρισης, την στέλνει μέσω του ίδιου δικτύου παραγωγής που μεταφέρει την κυκλοφορία δεδομένων.

Η χρήση ωστόσο του δικτύου παραγωγής για την κυκλοφορία διαχείρισης ενέχει έναν κίνδυνο. Ένα πρόβλημα στο δίκτυο παραγωγής μπορεί να εμποδίσει έναν SDN controller να το διορθώσει. Οι υποστηρικτές του SDN όμως, λένε ότι τα πιθανά προβλήματα δεν θα είναι τόσο σοβαρά όσο στο SNMP. Ένας controller SDN που έχει αποκοπεί από το δίκτυο θα είναι σε θέση να συνεχίσει να λειτουργεί αυτόνομα, ενώ ένας άνθρωπος διαχειριστής δεν θα μπορεί να διαχειρίζεται στοιχεία αν το δίκτυο μεταξύ του ίδιου και των στοιχείων δεν λειτουργεί. Το SDN σε έναν απομονωμένο controller μπορεί να αξιολογήσει την κατάσταση, και να προσπαθήσει να ορίσει εναλλακτικές διαδρομές. Επιπλέον οι υπέρμαχοι του SDN υποστηρίζουν ότι το λογισμικό που εκτελείται στους controllers λειτουργεί πιο γρήγορα από έναν άνθρωπο-διαχειριστή.

4.4 OpenFlow protocol

Προκειμένου να επιτευχθεί η επικοινωνία ανάμεσα στον SDN controller και ενός στοιχείου στο δίκτυο, απαιτείται η χρήση ενός πρωτοκόλλου. Το πρωτόκολλο το οποίο χρησιμοποιείται είναι το OpenFlow πρωτόκολλο. Μέσω του πρωτοκόλλου αυτού, καθορίζονται τα εξής: 1)Υπόδειγμα επικοινωνίας 2)Ορισμός και ταξινόμηση στοιχείων 3)Μορφή μηνυμάτων.

Το OpenFlow, χρησιμοποιεί ένα υπόδειγμα συνδεσμικής επικοινωνίας. Το πρωτόκολλο, είναι αυτό το οποίο καθιστά ασφαλής την επικοινωνία, δημιουργώντας ένα ασφαλές κανάλι. Ο controller ανοίγει μία σύνδεση TCP με SDN που εκτελείται σε ένα στοιχείο δικτύου. Ο πιο σύνηθες μηχανισμός για την επίτευξη της ασφαλούς επικοινωνίας, είναι η χρήση SSL(κρυπτογραφικό πρωτόκολλο προκειμένου να παρέχει ασφαλής επικοινωνία).

Το πρωτόκολλο επίσης επικεντρώνεται στην προώθηση πακέτων και χρησιμοποιεί ένα μοντέλο πίνακα ροής, ο οποίος περιέχει ένα σύνολο μοτίβων και μία ενέργεια για κάθε μοτίβο. Φτάνοντας ένα πακέτο το υλικό του επιπέδου δεδομένων εκτελεί μια διαδικασία ταιριάσματος μοτίβου και εφαρμόζει την ενέργεια που αντιστοιχεί στο μοτίβο το οποίο ταιριάζει.

Επιπλέον, το OpenFlow καθορίζει τόσο την μορφή μηνυμάτων όσο και την σημασία κάθε πεδίου. Ειδικότερα, για να εξασφαλιστεί η διαλειτουργικότητα σε διάφορες αρχιτεκτονικές, το OpenFlow ορίζει ότι όλες οι ακέραιες τιμές αναπαρίστανται με σειρά μεγάλου άκρου.

Ανάμεσα στα σημαντικότερα μηνύματα που ρέουν απ' τον ελεγκτή προς τον ελεγχόμενο μεταγωγέα είναι τα εξής:

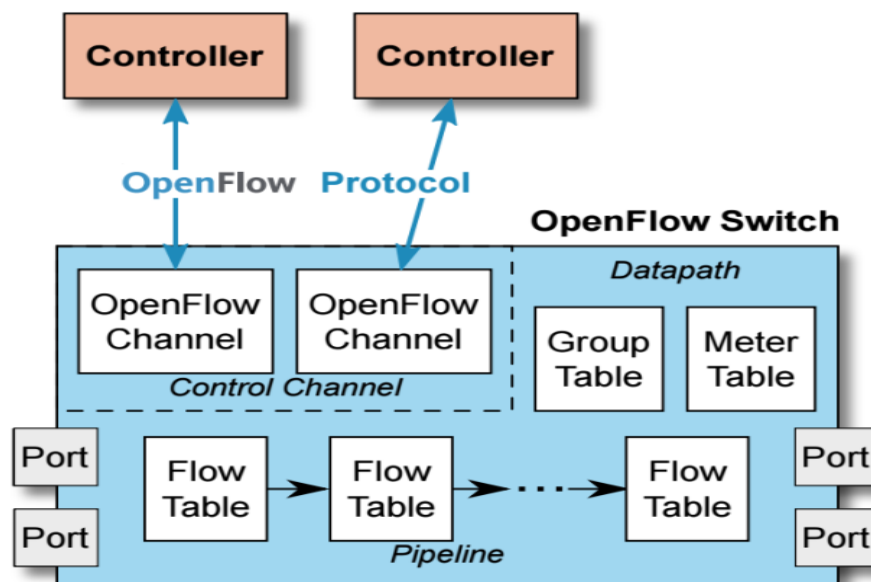
- Παραμετροποίηση. Επιτρέπει στον controller να υποβάλει ερωτήματα και να θέτει τις παραμέτρους παραμετροποίησης ενός μεταγωγέα.
- Τροποποίηση κατάστασης. Χρησιμοποιείται από έναν controller για να προσθέτει/διαγράφει ή τροποποιεί καταχωρίσεις μέσα στον πίνακα ροής του μεταγωγέα και να θέτει ιδιότητες θυρών μεταγωγέα.

- Κατάσταση Ανάγνωσης. Χρησιμοποιείται από έναν controller για να συλλέγει στατιστικά στοιχεία και τιμές μετρητών από τον πίνακα ροής και τις θύρες του μεταγωγέα.
- Αποστολή Πακέτου. Χρησιμοποιείται από τον controller για να στέλνει ένα συγκεκριμένο πακέτο σε μια καθορισμένη θύρα στον ελεγχόμενο μεταγωγέα. Το μήνυμα περιέχει το πακέτο, που θα σταλεί μέσα στο ωφέλιμο φορτίο του.

Ανάμεσα στα μηνύματα που ρέουν από τον ελεγχόμενο από SDN μεταγωγέα προς τον controller είναι τα εξής:

- Αφαίρεση Ροής. Πληροφορεί τον controller ότι μια καταχώριση του πίνακα ροής έχει αφαιρεθεί.
- Κατάσταση Θύρας. Χρησιμοποιείται από έναν μεταγωγέα για να πληροφορεί τον ελεγκτή για μια αλλαγή στην κατάσταση θύρας.
- Εισερχόμενο πακέτο. Όταν ένα πακέτο που φθάνει σε μια θύρα ενός μεταγωγέα δεν ταιριάζει με μια καταχώριση του πίνακα ροής, στέλνεται στον controller για πρόσθετη επεξεργασία. Τα πακέτα που ταιριάζουν μπορούν να στέλνονται επίσης στον controller, ως μια ενέργεια, που γίνεται στην περίπτωση ταιριάσματος. Το μήνυμα εισερχόμενου πακέτου χρησιμοποιείται για αποστολή τέτοιων πακέτων στον controller.

Τα δομικά μέρη του πρωτοκόλλου είναι ο OpenFlow controller και οι OpenFlow μεταγωγείς, οι οποίοι αποτελούνται από πόρτες, πίνακα ροών και ασφαλές κανάλι OpenFlow.



Εικόνα 4.2: Δομικά μέρη OpenFlow protocol(<https://www.cables-solutions.com/whats-openflow-switch-how-it-works.html>)

4.4.1 Πλεονεκτήματα OpenFlow πρωτοκόλλου

Τα κυριότερα πλεονεκτήματα στην χρήση του πρωτοκόλλου OpenFlow και του SDN είναι τα εξής:

- Το SDN επιτρέπει τον διαχωρισμό επιπέδου ελέγχου και του επιπέδου δεδομένων, πράγμα που σημαίνει ότι οι μεταγωγείς μπορούν να χρησιμοποιήσουν όλους τους πόρους υλικού για απλή προώθηση δεδομένων αντί για υπολογιστικές διαδρομές.
- Το OpenFlow παρέχει έναν εύκολο τρόπο επικοινωνίας μεταξύ controller και μεταγωγέα, που υλοποιείται εύκολα σε ένα υπάρχον δίκτυο.
- Οι περισσότερες τρέχουσες συσκευές υποστηρίζουν OpenFlow το οποίο μπορεί να ενεργοποιηθεί εύκολα, και να χρησιμοποιηθεί για μετάβαση στο SDN.
- Το OpenFlow παρέχει ασφάλεια με σύνδεση TLS για την αποφυγή κατασκοπείας και επιθέσεων DoS στον controller ή/και στο δίκτυο.
- Το OpenFlow δεν αλλάζει την διαμόρφωση ενός switch. Απλώς ενημερώνει τους πίνακες ροής, οι οποίοι ορίζουν την διαδρομή για ένα πακέτο.

4.5 SDN controllers

Ένας SDN controller διαχειρίζεται τον έλεγχο ροής προς τα switches/routers και τις εφαρμογές και την επιχειρηματική λογική για την ανάπτυξη ευφυών δικτύων. Οι SDN controllers, παγιώνουν και μεσολαβούν μεταξύ διαφορετικών τομέων ελεγκτών χρησιμοποιώντας κοινές διεπαφές εφαρμογών. Οι περισσότεροι ελεγκτές, βασίζονται στο πρωτόκολλο OpenFlow η περιγραφή του οποίου έγινε παραπάνω.

Στην συνέχεια θα αναλυθούν οι SDN controllers OpenDaylight, ONOS, και FlowVisor. Όσον αφορά τον τελευταίο, θα γίνει και μία επίδειξη για τον τρόπο λειτουργίας του σε ένα virtual network, και το πως μέσω του συγκεκριμένου controller μπορεί να τεμαχιστεί ένα δίκτυο.

Για την παρουσίαση των controllers απαιτείται η χρήση του Virtual Machine. Ο server που χρησιμοποιώ είναι ο ubuntu 18.0.4 τον οποίο μπορείτε να βρείτε εδώ <https://ubuntu.com/download/alternative-downloads>.

Οι τοπολογίες που θα φτιάξω θα γίνουν μέσω του mininet, το οποίο στην συνέχεια συνδέεται με τους controller. Το mininet <http://mininet.org/> διατίθεται στο github με λεπτομέρειες για τον τρόπο εγκατάστασής του. Θα παρουσιαστεί αμέσως μετά ο τρόπος εγκατάσταση του mininet.

<https://github.com/mininet/mininet/releases/tag/2.2.2>

Για μεγαλύτερη ευκολία, οτιδήποτε κάνω στην εικονική μηχανή, το κάνω μέσω SSH σύνδεσης. Αυτό, προϋποθέτει κατά την δημιουργία κάθε εικονικής μηχανής, να ενεργοποιηθεί μία δεύτερη κάρτα δικτύου. (Θα το δείξω κατά την εγκατάσταση του ODL).

Για την σύνδεση SSH χρησιμοποιώ την εφαρμογή PUTTY.

4.6 Mininet

Για την παρουσίαση των controller, που θα γίνει στην συνέχεια, είναι απαραίτητο να υπάρχει η εικονική μηχανή που περιέχει το Mininet. Μέσω του mininet, έχουμε την δυνατότητα να δημιουργήσουμε ένα ρεαλιστικό εικονικό δίκτυο, το οποίο εκτελεί πραγματικό kernel, switches και κώδικα εφαρμογής, σε μία μοναδική μηχανή (VM, cloud ή native), πολύ γρήγορα και με μία μόνο εντολή.

Επειδή μέσω του mininet μπορούμε εύκολα να αλληλεπιδράσουμε με το δίκτυο χρησιμοποιώντας το Mininet CLI (και API), να το προσαρμόσουμε, να το μοιραστούμε με άλλους ή να το αναπτύξουμε σε πραγματικό hardware, είναι χρήσιμο για ανάπτυξη εικονικών δικτύων.

Το Mininet, είναι ένας εξαιρετικός τρόπος ανάπτυξης, κοινής χρήσης και πειραματισμού με συστήματα δικτύωσης που καθορίζονται από το λογισμικό (SDN), χρησιμοποιώντας OpenFlow και P4.

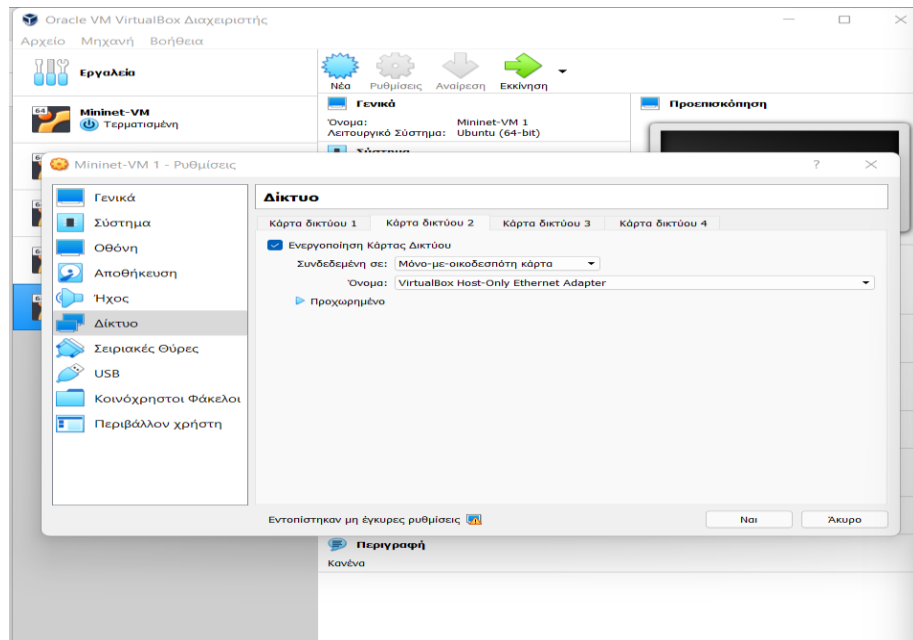
Το Mininet, αναπτύσσεται και υποστηρίζεται ενεργά και κυκλοφορεί με άδεια BSD Open Source.

4.6.1 Εγκατάσταση Mininet

Κατεβάζουμε ένα έτοιμο image που περιέχει το Mininet. Το αρχείο βρίσκεται στο <https://github.com/mininet/mininet/releases/tag/2.2.2> . Το αρχείο το οποίο θέλουμε είναι ανάλογα τον υπολογιστή μας, το `mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip` ή το `mininet-2.2.2-170321-ubuntu-14.04.4-server-i386.zip`

Στην συνέχεια εφόσον κατέβει, το αποσυμπιέζουμε και ανοίγουμε το αρχείο το οποίο έχει κατάληξη `.ovf` . Προϋπόθεση για να ανοίξει το αρχείο, είναι να έχουμε εγκατεστημένο στον υπολογιστή μας το Virtual Box.

Επιλέγουμε το αρχείο, και ανοίγει το παράθυρο στο Virtual Box για την εισαγωγή της εικονική μηχανής. Στο πλαίσιο το οποίο μας εμφάνισε, πατάμε στο **εισαγωγή**. Στην συνέχεια, έχοντας επιλέξει την εικονική μηχανή, πάμε στις *Ρυθμίσεις* -> *Δίκτυο* -> *Κάρτα δικτύου 2*. Την ενεργοποιούμε και επιλέγουμε *Συνδεδεμένη σε: Μόνο-με-οικοδεσπότη κάρτα*.



Εικόνα 4.3: Ενεργοποίηση Κάρτας Δικτύου 2

Στην συνέχεια επιλέγουμε **ΝΑΙ** και έπειτα εκκινούμε την μηχανή μας.

Κατά την εκκίνηση θα ζητηθεί username και password.

Username: mininet

Password: mininet

Επειδή το τερματικό που έχουμε δεν είναι εύχρηστο, δημιουργούμε μία σύνδεση SSH μέσω του Putty. Για να πραγματοποιηθεί αυτή η σύνδεση, χρειαζόμαστε την IP της δεύτερης κάρτας δικτύου την οποία ενεργοποιήσαμε. Προκειμένου να βρούμε την IP, δίνουμε την εντολή:

```
mininet@mininet-vm:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:06:ef:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.112/24 brd 192.168.56.255 scope global eth0
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:32:48:f9 brd ff:ff:ff:ff:ff:ff
mininet@mininet-vm:~$
```

Εικόνα 4.4: Εύρεση Διεύθυνσης IP

Είναι απαραίτητο να ρυθμιστεί το eth1 έτσι ώστε να παίρνει αυτόματα IP.

Αυτό θα γίνει ως εξής:

Ανοίγουμε το αρχείο /etc/network/interfaces

```
sudo vi /etc/network/interfaces
```

Προσθέτουμε τα εξής:

```
auto eth1
iface eth1 inet dhcp
```

Για να γράψουμε στο αρχείο πατάμε το γράμμα “i” , και αφού γράψουμε, προκειμένου να το αποθηκεύσουμε πατάμε το ESC, και στην συνέχεια τα :wq και enter.

Έπειτα, για να μην κάνουμε επανεκκίνηση και να ανατεθεί διεύθυνση στο eth1, δίνουμε την εντολή:

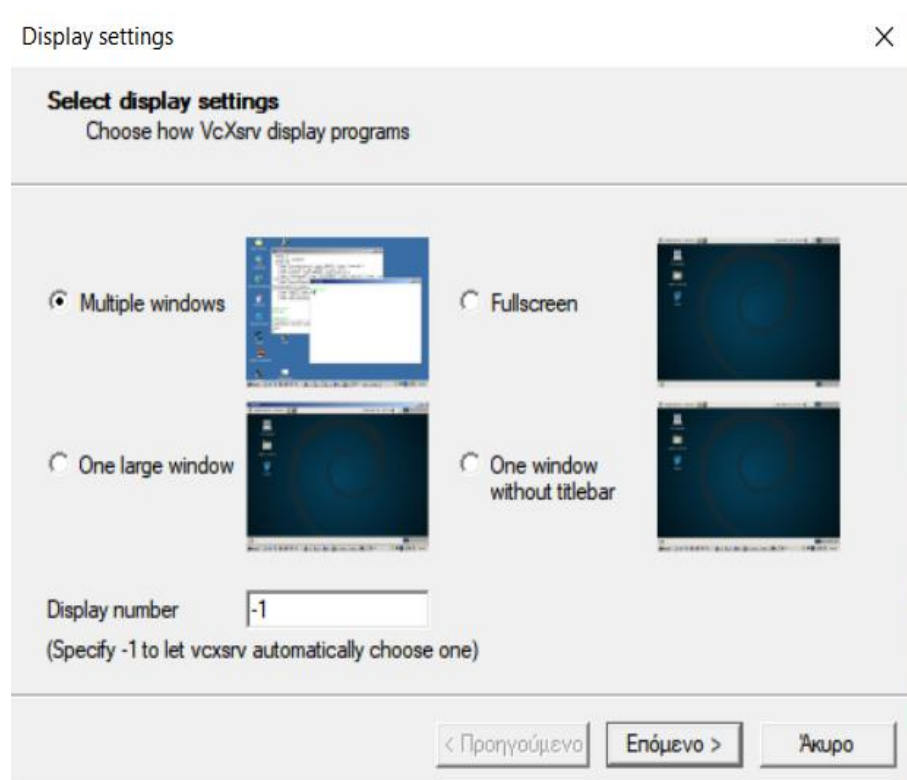
```
sudo dhclient eth1
```

Μέσω του mininet, μπορούμε να χρησιμοποιήσουμε παραθυρικές εφαρμογές, όπως είναι το MiniEdit (που θα μας χρησιμεύσει στην συνέχεια). Για μπορέσουμε να τις χρησιμοποιήσουμε, κατεβάζουμε έναν ειδικό server που υποστηρίζει το Mininet.

Ένας από αυτούς είναι ο Xming X Server for Windows.

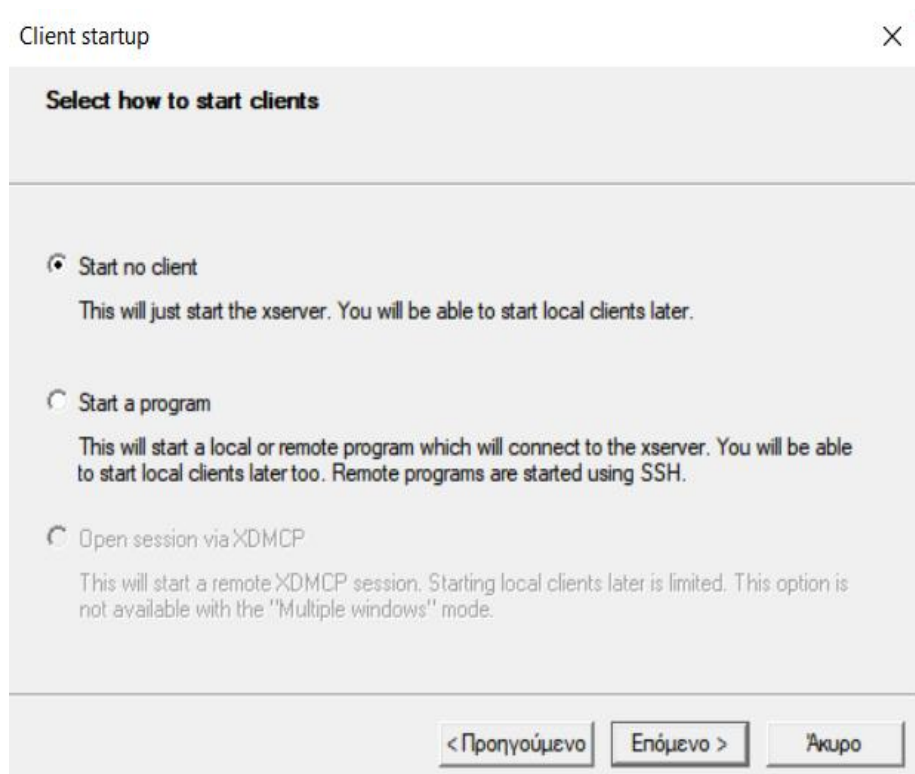
<https://sourceforge.net/projects/xming/>

Όποτε χρειαστούμε λοιπόν να χρησιμοποιήσουμε παραθυρικές εφαρμογές, θα πρέπει πρώτα να τρέξουμε τον server αυτόν.



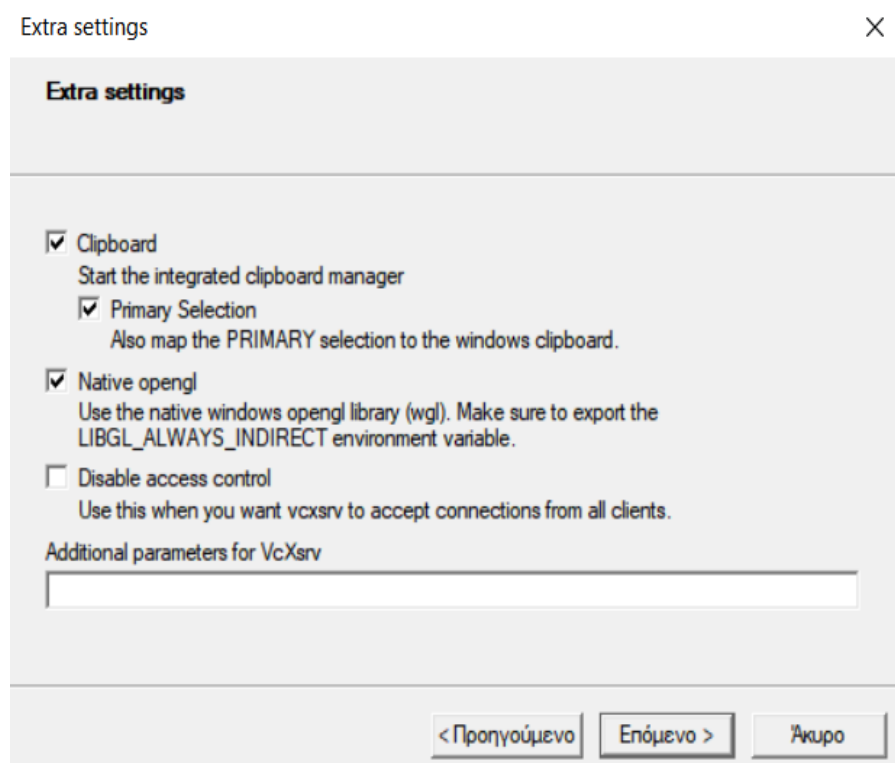
Εικόνα 4.5: Εκκίνηση XLaunch Server

Επόμενο >.



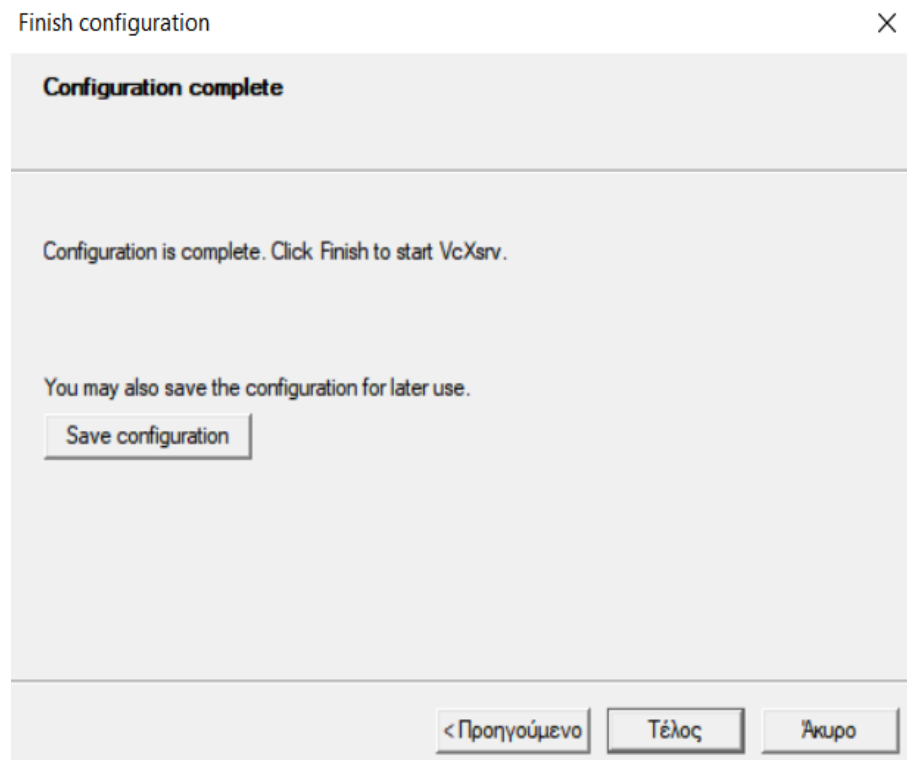
Εικόνα 4.6: Ενεργοποίηση XLaunch Server

Επόμενο >.



Εικόνα 4.7: Ενεργοποίηση XLaunch Server

Επόμενο >.



Εικόνα 4.8: Ενεργοποίηση XLaunch Server

Τέλος.

Η ένδειξη ότι ο server έχει ξεκινήσει, θα φανεί κάτω δεξιά στην οθόνη.



Εικόνα 4.9: Ένδειξη Ενεργοποίησης XLaunch Server

Προκειμένου να σταματήσουμε τον server, κάνουμε δεξί κλικ στο εικονίδιο, και επιλέγουμε το **Exit**.

Για παράδειγμα, έχοντας εκκινήσει τον server, μπορούμε να ανοίξουμε το MiniEdit. Εάν ο server δεν τρέχει, τότε παίρνουμε το παρακάτω σφάλμα:

```
mininet@mininet-vm: ~
mininet@mininet-vm:~$ sudo ~/mininet/mininet/examples/miniedit.py
MiniEdit running against Mininet 2.2.2
PuTTY X11 proxy: unable to connect to forwarded X server: Network error: Connection refused
PuTTY X11 proxy: unable to connect to forwarded X server: Network error: Connection refused
Traceback (most recent call last):
  File "/home/mininet/mininet/mininet/examples/miniedit.py", line 3579, in <module>
    app = MiniEdit()
  File "/home/mininet/mininet/mininet/examples/miniedit.py", line 1110, in __init__
    Frame.__init__(self, parent)
  File "/usr/lib/python2.7/lib-tk/Tkinter.py", line 2537, in __init__
    Widget.__init__(self, master, 'frame', cnf, {}, extra)
  File "/usr/lib/python2.7/lib-tk/Tkinter.py", line 2049, in __init__
    BaseWidget.setup(self, master, cnf)
  File "/usr/lib/python2.7/lib-tk/Tkinter.py", line 2024, in setup
    default_root = Tk()
  File "/usr/lib/python2.7/lib-tk/Tkinter.py", line 1767, in __init__
    self.tk = _tkinter.create(screenName, baseName, className, interactive, wantobjects, useTk, sync, use)
_tkinter.TclError: couldn't connect to display "localhost:10.0"
mininet@mininet-vm:~$
```

Εικόνα 4.10: Σφάλμα Εκκίνησης MiniEdit χωρίς τον XLaunch

Δίνουμε την εντολή:

```
sudo ~/mininet/mininet/examples/miniedit.py
```

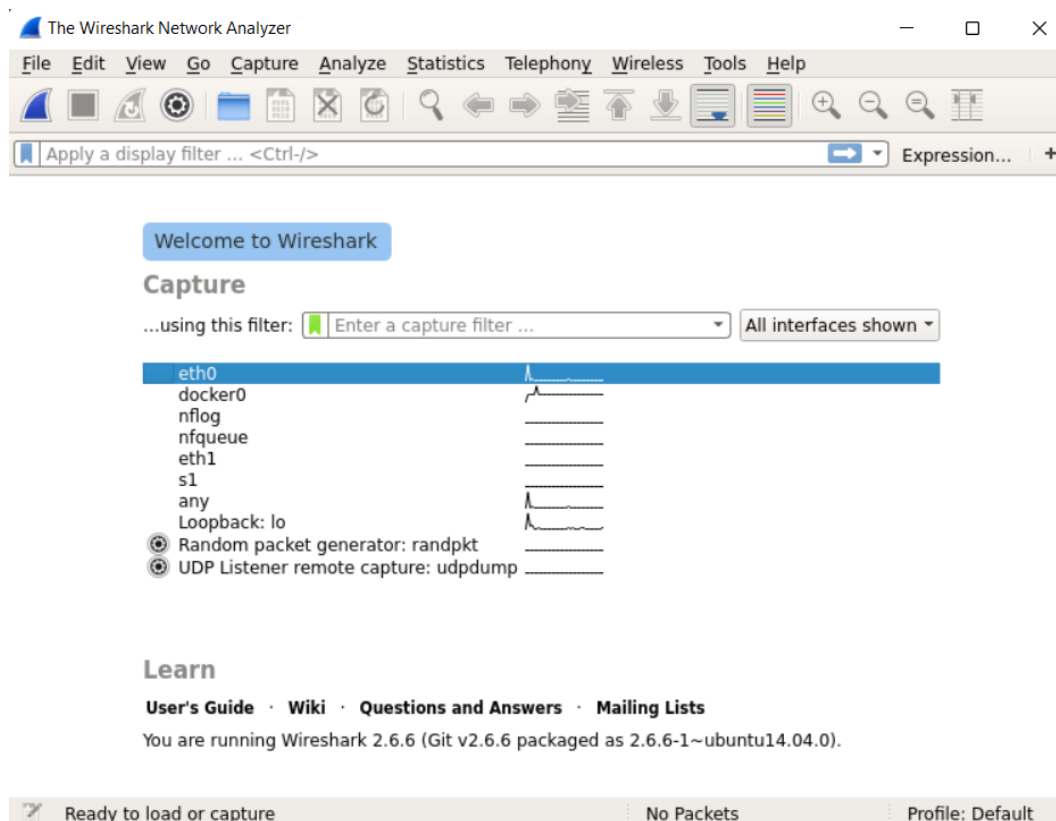
Αυτό που παίρνουμε, είναι το περιβάλλον του MiniEdit.

**Εικόνα 4.11: Περιβάλλον MiniEdit**

Μία άλλη χρήσιμη παραθυρική εφαρμογή που μπορούμε να πάρουμε μέσω του Mininet, είναι το Wireshark. Για να το ανοίξουμε, δίνουμε την ακόλουθη εντολή:

```
sudo wireshark &
```

Το αποτέλεσμα είναι:



Εικόνα 4.12: Περιβάλλον Wireshark

Το Miniedit, θα μας χρειαστεί στην συνέχεια που θα γίνει η παρουσίαση του τεμαχισμού δικτύου.

4.7 OpenDaylight

Το OpenDaylight είναι μία ανοιχτή πλατφόρμα για την προσαρμογή και την αυτοματοποίηση δικτύων οποιουδήποτε μεγέθους και κλίμακας. Προέκυψε λόγω του SDN, με σαφή εστίαση στην δυνατότητα προγραμματισμού δικτύου. Από την αρχή, σχεδιάστηκε ως βάση για εμπορικές λύσεις που αντιμετωπίζουν ποικίλες περιπτώσεις χρήσης σε υπάρχοντα περιβάλλοντα δικτύου. Υποστηρίζει το OpenFlow protocol καθώς και άλλα πρωτόκολλα και τεχνολογίες SDN αρχιτεκτονικής εφόσον μπορεί να παραμετροποιηθεί καταλλήλως από τον διαχειριστή.

Το OpenDaylight είναι η πιο ευρέως διαδεδομένη πλατφόρμα ελεγκτών SDN ανοιχτού κώδικα και διαθέτει 13 εκδόσεις, 1000+ συγγραφείς, 100K+ δεσμεύσεις και εξουσιοδοτεί δίκτυα 1B+ παγκοσμίως συνδρομητών.

4.7.1 Αρχιτεκτονική OpenDaylight Model-Driven

Ο πυρήνας του OpenDaylight είναι το Model-Driven Service Abstraction Layer (MD-SAL). Στο OpenDaylight, οι υποκείμενες συσκευές δικτύου και οι εφαρμογές δικτύου αναπαριστώνται όλες ως αντικείμενα ή μοντέλα, των οποίων οι αλληλεπιδράσεις υποβάλλονται σε επεξεργασία εντός του SAL.

Το Service Abstraction Layer (SAL) είναι ένας μηχανισμός ανταλλαγής δεδομένων και προσαρμογής μεταξύ των μοντέλων YANG που αντιπροσωπεύουν συσκευές και εφαρμογές δικτύου. Τα μοντέλα YANG παρέχουν γενικευμένες περιγραφές των δυνατοτήτων μιας συσκευής ή εφαρμογής, χωρίς να απαιτείται να γνωρίζουν τις συγκεκριμένες λεπτομέρειες εφαρμογής της άλλης. Εντός του SAL, τα μοντέλα ορίζονται απλά από τους αντίστοιχους ρόλους τους σε μια δεδομένη αλληλεπίδραση. Ένα μοντέλο παραγωγού εφαρμόζει ένα API και παρέχει τα δεδομένα του API. Ένα μοντέλο καταναλωτή, χρησιμοποιεί το API και καταναλώνει τα δεδομένα του API.

Ενώ οι όροι “northbound” και “southbound” παρέχουν την άποψη ενός μηχανικού δικτύου για το SAL, οι όροι “καταναλωτής” και “παραγωγός” είναι ακριβέστερες περιγραφές των αλληλεπιδράσεων εντός του SAL. Το SAL, ταιριάζει με τους παραγωγούς και τους καταναλωτές από τις αποθήκες δεδομένων του και ανταλλάσσει πληροφορίες. Ένας καταναλωτής μπορεί να βρει έναν πάροχο για τον οποίο ενδιαφέρεται. Ένας παραγωγός μπορεί να παράγει ειδοποιήσεις, ενώ ένας καταναλωτής μπορεί να λαμβάνει ειδοποιήσεις και να εκδίδει RPC για την λήψη δεδομένων από τους παρόχους. Ένας παραγωγός μπορεί να εισάγει δεδομένα στην αποθήκευση του SAL. Ένας καταναλωτής μπορεί να διαβάσει δεδομένα από την αποθήκευση του SAL. Τέλος, ένας παραγωγός εφαρμόζει ένα API και παρέχει τα δεδομένα του API, ενώ ένας καταναλωτής χρησιμοποιεί το API και καταναλώνει τα δεδομένα του API.

Modular and Multiprotocol

Η πλατφόρμα OpenDaylight (ODL) έχει σχεδιαστεί για να παρέχει στους μεταγενέστερους χρήστε και παρόχους λύσεων, την μέγιστη ευελιξία στην κατασκευή ενός ελεγκτή που θα ανταποκρίνεται στις ανάγκες τους. Ο αρθρωτός σχεδιασμός της πλατφόρμας ODL επιτρέπει σε οποιονδήποτε στο οικοσύστημα ODL να αξιοποιήσει υπηρεσίες που δημιουργήθηκαν από άλλους, να γράψει και να ενσωματώσει την δική του και να μοιραστεί το έργο του με άλλους. Το ODL περιλαμβάνει υποστήριξη για το ευρύτερο σύνολο πρωτοκόλλων σε οποιαδήποτε πλατφόρμα SDN – OpenFlow, OVSDB, NETCONF, BGP και πολλά άλλα, που βελτιώνουν την προγραμματισσιμότητα των σύγχρονων δικτύων και επιλύουν μια σειρά αναγκών των χρηστών.

Τα southbound πρωτόκολλα και οι control plane services, που αγκυρώνονται από το MD-SAL, μπορούν να επιλέγουν ή να γραφούν μεμονωμένα και να συσκευαστούν μαζί σύμφωνα με τις απαιτήσεις μια συγκεκριμένης περίπτωσης χρήσης. Ένα πακέτο ελεγκτή είναι δομημένο γύρω από τέσσερα βασικά συστατικά. Σε αυτό, ο προγραμματιστής λύσεων προσθέτει μία σχετική ομάδα πρόσθετων southbound πρωτοκόλλων, τις περισσότερες ή όλες από τις τυπικές λειτουργίες του επιπέδου ελέγχου και ορισμένες επιλεγμένες εφαρμογές ενσωματωμένου και εξωτερικού ελεγκτή, τις οποίες διαχειρίζεται η πολιτική.

Το ODL είναι ο κύριος τύπος για την ανάπτυξη και δοκιμή διαφορετικών προσεγγίσεων στον πολιτική και την πρόσθεση, όπως το ALTO, το Group Based Policy και το Network Intent Composition. Συνεργάζεται στενά με διάφορους βιομηχανικούς ομίλους, όπως το Open Networking Foundation και το IETF για να εξεταστούν και να δοκιμαστούν διαφορετικές προσεγγίσεις.

Καθένα από τα παραπάνω εξαρτήματα, απομονώνεται ως χαρακτηριστικό του Karaf, για να διασφαλιστεί ότι η νέα εργασία δεν παρεμβαίνει σε ώριμο, δοκιμασμένο κώδικα. Το ODL χρησιμοποιεί το OSGi και το Maven για να δημιουργήσει ένα πακέτο που διαχειρίζεται αυτές τις λειτουργίες Karaf και τις αλληλεπιδράσεις τους. Αυτό το αρθρωτό πλαίσιο επιτρέπει στους προγραμματιστές και τους χρήστες να:

- Εγκαταστήσουν μόνο τα πρωτόκολλα και τις υπηρεσίες που χρειάζονται.
- Συνδυάσουν πολλαπλές υπηρεσίες και πρωτόκολλα για την επίλυση πιο σύνθετων προβλημάτων καθώς προκύπτουν ανάγκες.
- Εξελίσσονται οι δυνατότητες της πλατφόρμας ανοιχτού κώδικα σταδιακά και συνεργατικά.
- Αναπτύξουν γρήγορα προσαρμοσμένες λειτουργίες προστιθέμενης αξίας για εξαιρετικά εξειδικευμένες περιπτώσεις χρήσης, αξιοποιώντας μια κοινή πλατφόρμα που μοιράζεται σε όλο τον κλάδο.

S3P: Security, Scalability, Stability and Performance

Το ODL παρέχει συνεχείς βελτιώσεις σε όλα τα έργα του στους τομείς της ασφάλειας, της επεκτασιμότητας, της σταθερότητας και της απόδοσης (S3P). Το ODL συνεντάσσεται επίσης με την OPNFV για την υποστήριξη ενός έργου Δοκιμών Απόδοσης Ελεγκτή που θα δημιουργούσε δοκιμές απόδοσης σε όλο τον κλάδο για ελεγκτές SDN σε ρεαλιστικές, μεγάλες, αυτοματοποιημένες αναπτύξεις

Η ασφάλεια αποτελεί βασικό τομέα εστίασης για το ODL. Η πλατφόρμα παρέχει ένα πλαίσιο για τον έλεγχο ταυτότητας, πρόσβασης και χρήσης υπηρεσιών (Authentication, Authorization and Accounting (AAA)) καθώς και την αυτόματη ανακάλυψη και την ασφάλεια των συσκευών δικτύου και των ελεγκτών.

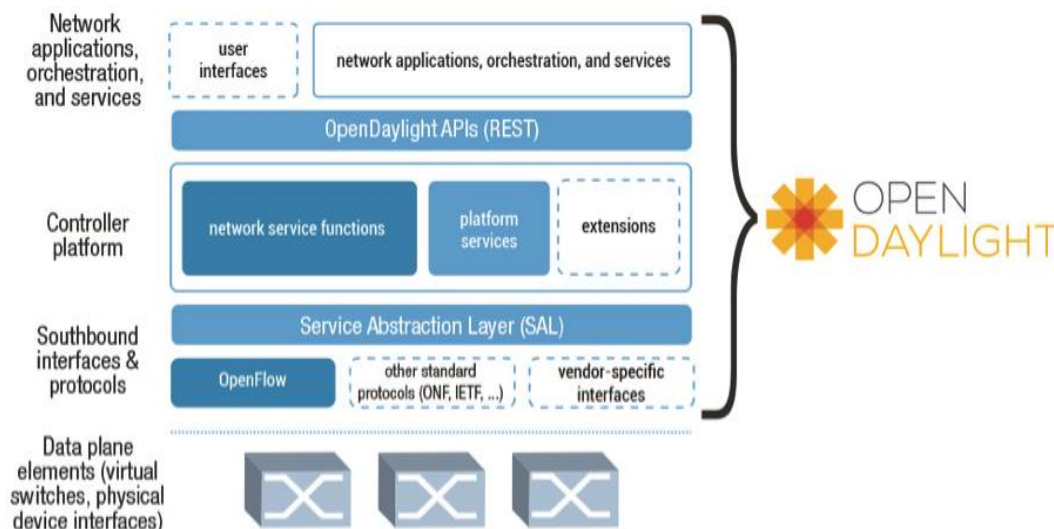
Γενικότερα, το λογισμικό ανοιχτού κώδικα έχει σημαντικά πλεονεκτήματα όσον αφορά την ασφάλεια. Ο καθένας μπορεί να βρει και να αναφέρει τρωτά σημεία. Η κοινότητα του ODL βασίζεται σε ένα ευρύ φάσμα εμπειρογνομόνων και προγραμματιστών σε όλες τις εταιρίες για να συζητούν και να διορθώνουν τρωτά σημεία.

4.7.2 Μελέτη περίπτωσης

Εφαρμογές υπηρεσίας δικτύου είναι οι εφαρμογές που καθορίζουν πώς η προώθηση και οι άλλες υπηρεσίες επιπέδου δεδομένων, όπως οι υπηρεσίες firewall και ισοστάθμισης φορτίου, επιτυγχάνονται μέσα σε ελεγχόμενους μεταγωγείς. Ο ODL έχει δύο διεπαφές, μέσω των οποίων οι εφαρμογές μπορούν να επικοινωνούν με γηγενείς υπηρεσίες ελεγκτή και μεταξύ τους. Οι εξωτερικές εφαρμογές επικοινωνούν με ενότητες ελεγκτή χρησιμοποιώντας μια REST API αίτησης-απόκρισης, που εκτελείται επί HTTP. Οι εσωτερικές εφαρμογές επικοινωνούν μεταξύ τους μέσω του επιπέδου SAL. Η επιλογή για το αν μια εφαρμογή ελεγκτή υλοποιείται εξωτερικά ή εσωτερικά ανήκει στον σχεδιασμό της εφαρμογής.

Οι βασικές λειτουργίες υπηρεσίας δικτύου του ODL είναι η καρδιά του ελεγκτή και αντιστοιχούν με τις δυνατότητες διαχείρισης κατάστασης για όλο το δίκτυο. Το

SAL είναι το νευρικό σύστημα του ελεγκτή, επιτρέποντας σε συστατικά του ελεγκτή και σε εφαρμογές να καλούν τις υπηρεσίες των άλλων συστατικών και να εγγράφονται για να πληροφορούνται για συμβάντα που παράγουν. Επίσης παράγει μια ομοιόμορφη αφαιρετική διεπαφή, προς τα συγκεκριμένα υποκείμενα πρωτόκολλα επικοινωνιών, μέσα στο επίπεδο επικοινωνίας, τα οποία περιλαμβάνουν το OpenFlow και το SNMP. Το OVSDB πρωτόκολλο, είναι ένα πρωτόκολλο που χρησιμοποιείται για την διαχείριση μεταγωγής δεδομένων, μια σημαντική περιοχή εφαρμογής της τεχνολογίας SDN.



Εικόνα 4.13: Αρχιτεκτονική OpenDaylight(<https://www.embedded.com/network-firms-form-software-defined-networking-.opendaylight-project/>)

4.7.3 REST API

Το REST API είναι μία αρχιτεκτονική η οποία χρησιμοποιείται στις διαδικτυακές εφαρμογές. Το REST προσφέρει στα συστήματα που βασίζονται σε αυτή την αρχιτεκτονική, καλή απόδοση, αξιοπιστία και δυνατότητα επέκταση ως προς το πλήθος των χρηστών που μπορούν να το υποστηρίξουν.

Τα RestFul συστήματα επικοινωνούν μέσω HTTP πρωτοκόλλου μέσω των μεθόδων GET, PUT, POST και DELETE οι οποίες χρησιμοποιούνται για να σταλούν δεδομένα σε remote εξυπηρετητές. Η αρχιτεκτονική REST έχει κάποια βασικά πλεονεκτήματα έναντι άλλων παρόμοιων αρχιτεκτονικών:

- Αξιοποιείται η cache memory.
- Είναι stateless πρωτόκολλο, επομένως η αίτηση ενός πελάτη με την αντίστοιχη απάντηση είναι ανεξάρτητες.
- Αξιοποιείται ένα πολυεπίπεδο σύστημα.
- Αξιοποιείται μια ενιαία διεπαφή.

Η REST αρχιτεκτονική χρησιμοποιεί το RESTCONF πρωτόκολλο, το οποίο δίνει την δυνατότητα πρόσβασης σε δομές δεδομένων στον ελεγκτή. Τα είδη των δομών δεδομένων είναι:

- Config: Περιέχει πληροφορίες που προστίθενται μέσω ελεγκτή.

- Operational: Περιέχει πληροφορίες που προστίθενται μέσω του δικτύου.

Η REST αρχιτεκτονική είναι αυτή η οποία χρησιμοποιείται από τον ODL και επομένως προσφέρει όσα προαναφέρθηκαν.

Κάθε αίτηση που γίνεται μέσω HTTP μεθόδου, πρέπει να γίνεται στο αντίστοιχο URI, το οποίο έχει την μορφή:

<http://<odl-IP>:8181/restconf/operational/<path>/>

<http://<odliIP>:8181/restconf/config/<path>/>

4.7.4 Εγκατάσταση-Παρουσίαση OpenDaylight

Ανοίγουμε το VirtualBox.

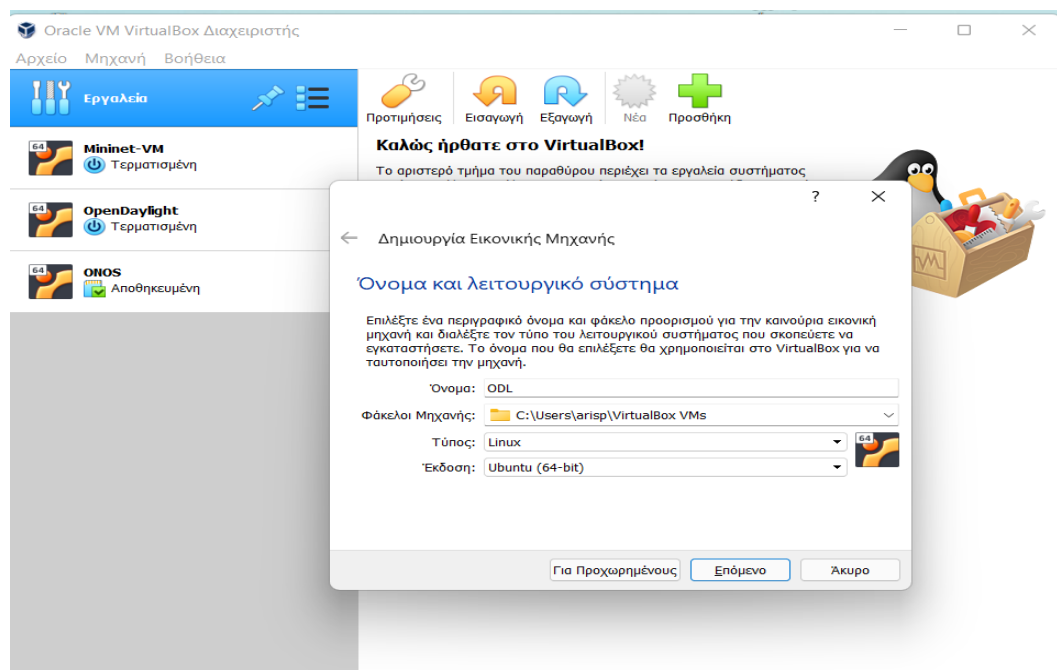
Πάμε στην επιλογή Νέα.

Δίνουμε το όνομα που θέλουμε να έχει η εικονική μηχανή μας.

Τύπος: Linux

Έκδοση: Ubuntu(64 bit) ή Ubuntu(32 bit)

Και πατάμε επόμενο.



Εικόνα 4.14: Ρυθμίσεις Ονόματος και Λειτουργικού Συστήματος VM

Μέγεθος μνήμης έχει ρυθμιστεί στα 1024MB. Μπορούμε αν θέλουμε να το αυξομειώσουμε.

Σκληρός δίσκος: Δημιουργήστε έναν εικονικό σκληρό δίσκο τώρα -> Δημιουργία

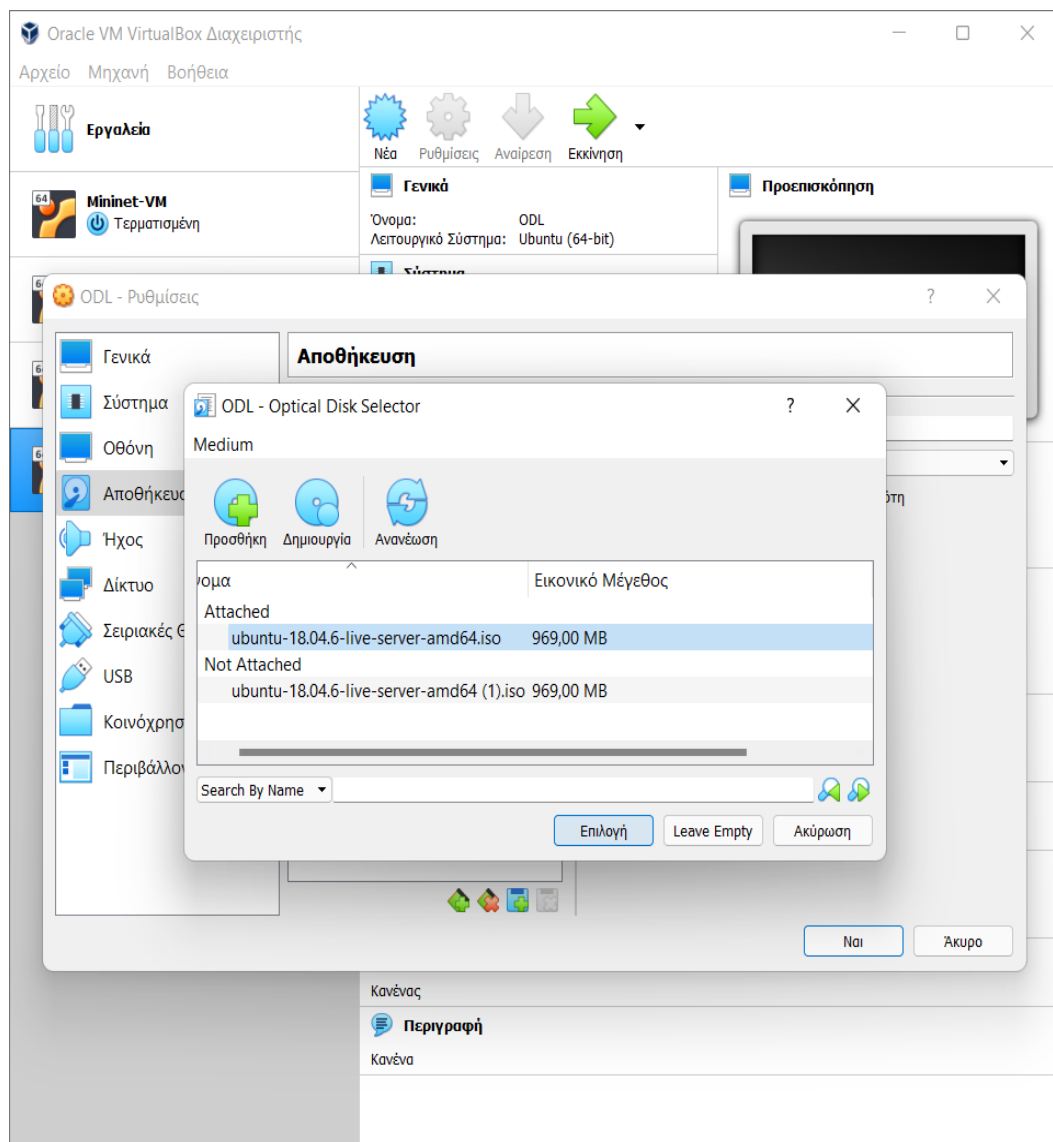
Τύπος αρχείου σκληρού δίσκου: VDI ->Επόμενο

Αποθήκευση σε πραγματικό σκληρό δίσκο: Δυναμική εκχώρηση -> Επόμενο

Θέση αρχείου και μέγεθος: Επιλέγουμε που θέλουμε να αποθηκευτεί. Το μέγεθος έχει οριστεί στα 10GB, ωστόσο μπορούμε να το αυξομειώσουμε. ->Δημιουργήστε.

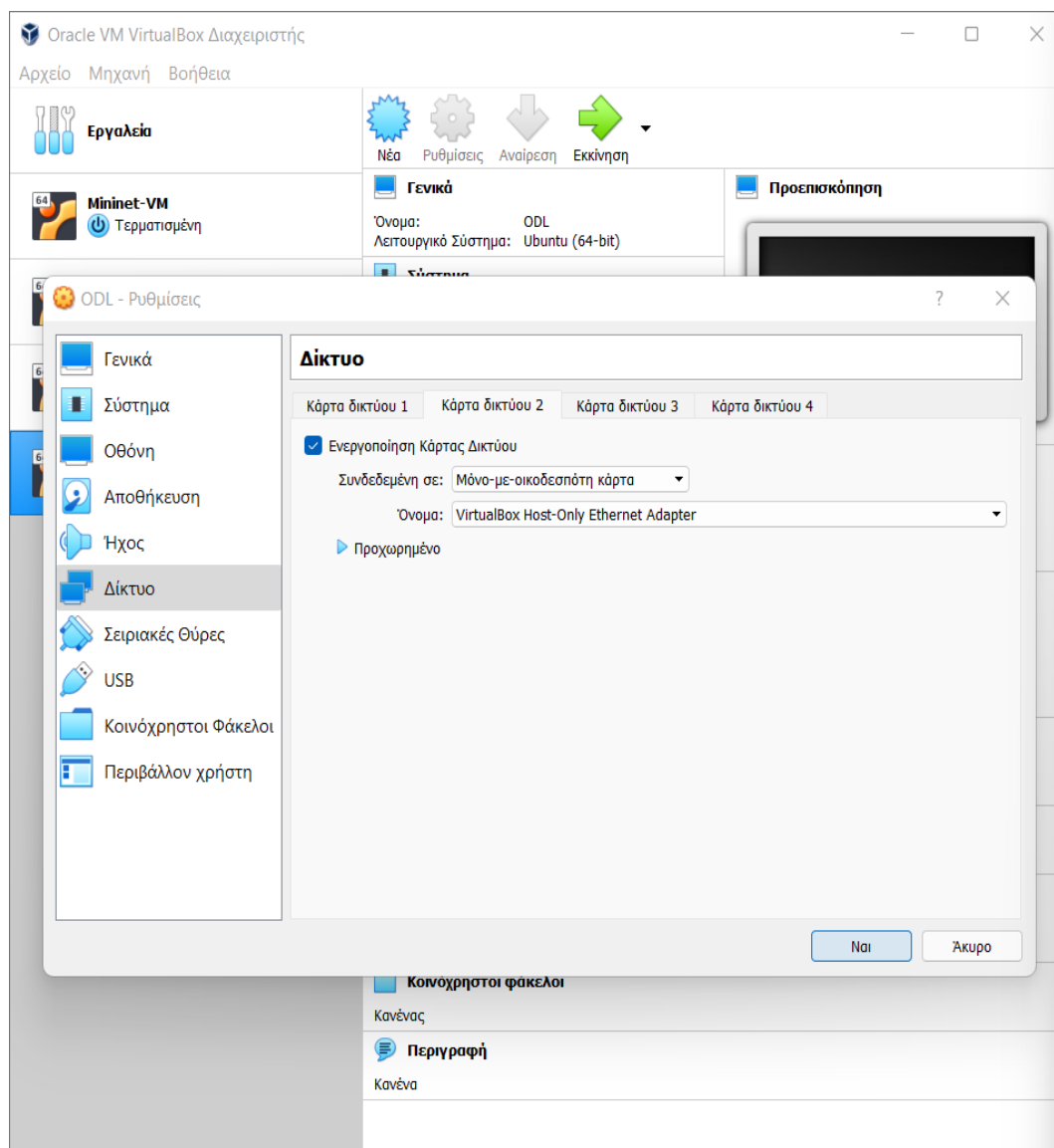
Έχοντας δημιουργηθεί η εικονική μηχανή, την επιλέγουμε και πάμε στις Ρυθμίσεις.

Ρυθμίσεις->Αποθήκευση. Στο Ελεγκτής:IDE επιλέγουμε το πρώτο εικονίδιο στα δεξιά του: Πρόσθεση οπτικού οδηγού, και εκεί μας έχει να επιλέξουμε τον server που έχουμε κατεβάσει. Τον επιλέγουμε και πατάμε Επιλογή->Ναι



Εικόνα 4.15: Πρόσθεση Οπτικού Οδηγού

Επιλέγουμε εκ νέου την εικονική μηχανή, και πάμε Ρυθμίσεις->Δίκτυο->Κάρτα δικτύου 2 και ενεργοποιούμε την κάρτα. Στην συνέχεια στο πλαίσιο που λέει Συνδεδεμένη σε: Επιλέγουμε *Μόνο-με-οικοδεσπότη κάρτα*. Πατάμε Ναι.



Εικόνα 4.16: Ρυθμίσεις Κάρτας Δικτύου 2

Η εικονική μηχανή είναι έτοιμη, και μπορούμε να την εκκινήσουμε.

Εκκινώντας την μηχανή, υπάρχουν κάποιες ρυθμίσεις που ζητείται να γίνουν.

Την ρυθμίζουμε στα μέτρα μας, αλλά πρέπει να αποθηκεύσουμε κάπου την IP της δεύτερης κάρτα δικτύου η οποία εμφανίζεται στο πλαίσιο Network connections. Η IP αυτή θα χρησιμεύσει για την σύνδεση με το PUTTY. Ωστόσο, αν δεν την αποθηκεύσουμε κάπου, μπορούμε να την βρούμε εφόσον έχει δημιουργηθεί η μηχανή μας, δίνοντας την εντολή: `ip addr`.

Στην συνέχεια στο πλαίσιο SSH Setup, επιλέγουμε να γίνει Install OpenSSH server.

Εφόσον έχουν γίνει τα παραπάνω, μετά από μερικά λεπτά που χρειάζονται για να γίνουν οι κατάλληλες ρυθμίσεις από το σύστημα, το virtual machine είναι έτοιμο.

Για την εγκατάσταση του ODL:

JAVA:

Πρέπει να γίνει εγκατάσταση του JAVA 8 JRE:

```
sudo apt-get -y install openjdk-8-jre
```

Στην συνέχεια ανοίγουμε το bashrc αρχείο.

```
vim ~/.bashrc
```

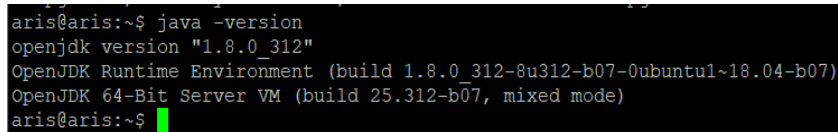
Και προσθέτουμε την παρακάτω γραμμή.

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
```

Αποθηκεύουμε το αρχείο.

```
source ~/.bashrc
```

Μπορούμε να δούμε αν όλα πήγαν καλά δίνοντας την εντολή: `java -version`, η οποία θα μας δώσει ως αποτέλεσμα την version η οποία εγκαταστάθηκε.



```
aris@aris:~$ java -version
openjdk version "1.8.0_312"
OpenJDK Runtime Environment (build 1.8.0_312-8u312-b07-0ubuntu1~18.04-b07)
OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)
aris@aris:~$
```

Εικόνα 4.17: Java Version

OpenDaylight:

Εφόσον έχει εγκατασταθεί σωστά η JAVA 8 JRE μπορούμε να ξεκινήσουμε για την εγκατάσταση του controller.

Κατεβάζουμε το ODL συμπιεσμένο εδώ.

wget

<https://nexus.opendaylight.org/content/groups/public/org/.opendaylight/integration/distribution-karaf/0.5.0-Boron/distribution-karaf-0.5.0-Boron.zip>

Στην συνέχεια για να το κάνουμε unzip, εγκαθιστούμε πρώτα το unzip.

```
sudo apt-get install unzip
```

Και κάνουμε μετά unzip το αρχείο μας.

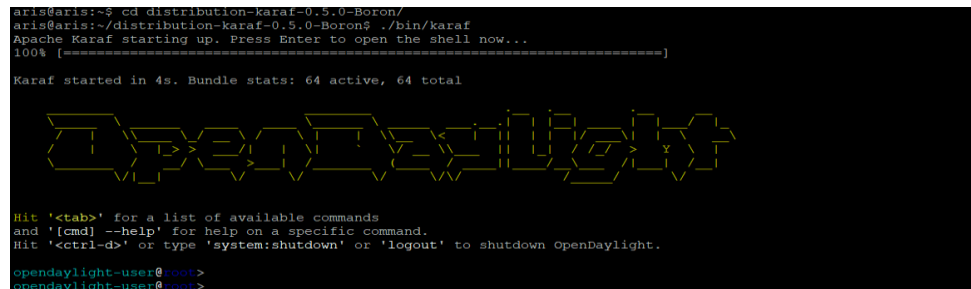
```
unzip distribution-karaf-0.5.0-Boron.zip
```

Και τώρα μπορούμε να εκκινήσουμε τον controller.

```
cd distribution-karaf-0.5.0-Boron/
```

```
./bin/karaf
```

Εφόσον όλα έχουν γίνει όπως πρέπει, ο controller έχει ξεκινήσει και βρισκόμαστε εδώ.

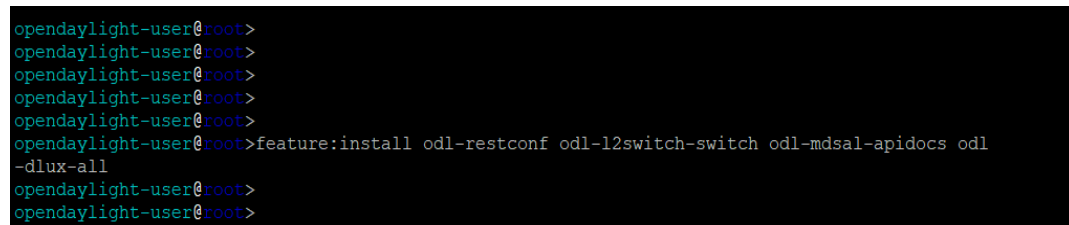


Εικόνα 4.18: Εκκίνηση OpenDaylight στο Τερματικό

Θα πρέπει να προσθέσουμε κάποια features τα οποία είναι απαραίτητα για την επίδειξη του ODL.

Δίνουμε την εντολή:

```
feature:install odl-restconf odl-l2switch-switch odl-mdsal-apidocs odl-dlux-all
```



Εικόνα 4.19: Εγκατάσταση Πρόσθετων Features στον OpenDaylight

Περιμένουμε λίγα δευτερόλεπτα.

Τώρα μπορούμε να αποκτήσουμε πρόσβαση στον controller μέσω του browser μας.

Η πρόσβαση γίνεται μέσω του URL:

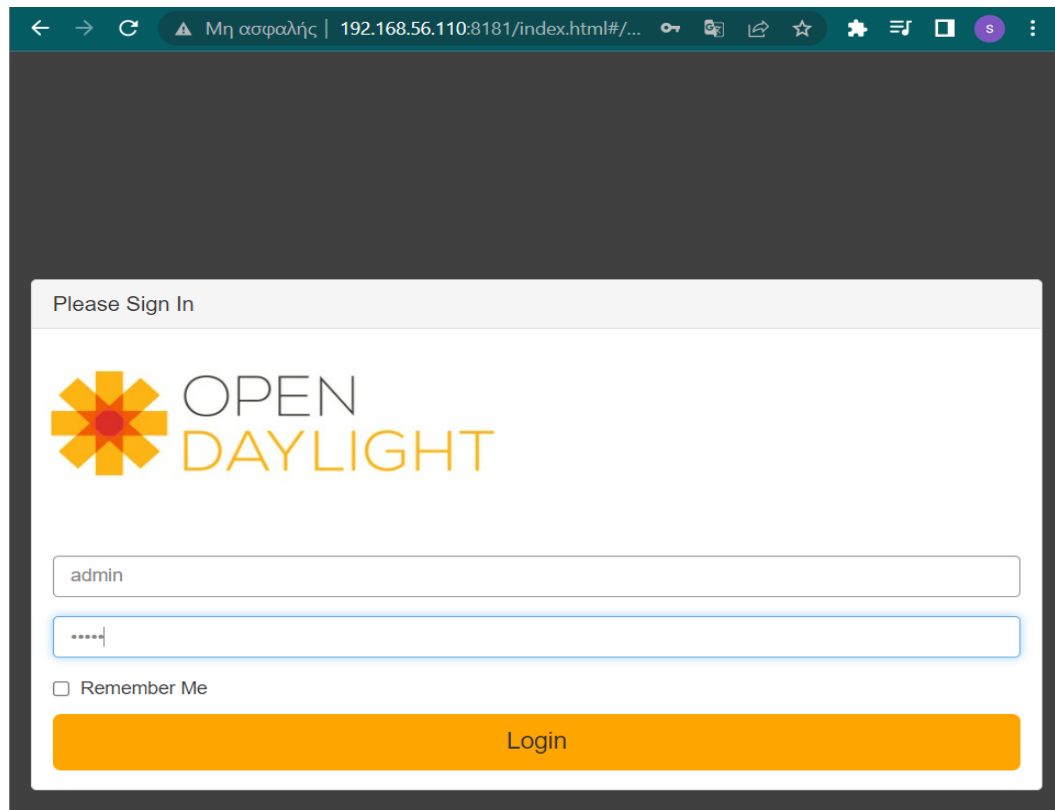
<http://ip-address-of-opendaylight-:8181/index.html#/login>

Όπου ip-address-of-opendaylight είναι η IP της δεύτερης κάρτα δικτύου, η οποία χρησιμοποιείται και για την σύνδεση μέσω PUTTY.

Εάν όλα πάνε καλά, ήμαστε έτοιμοι να συνδεθούμε.

Username: admin

Password: admin



Εικόνα 4.20: Σύνδεση στο Περιβάλλον του ODL

Αφού συνδεθούμε, βλέπουμε το πλαίσιο στο οποίο θα εμφανιστεί η τοπολογία μας, το οποίο είναι κενό αυτήν την στιγμή. Αυτό είναι λογικό, αφού μέχρι στιγμής δεν έχουμε δημιουργήσει κάποια τοπολογία.

Mininet

Εκκινούμε την εικονική μηχανή του Mininet.

Δημιουργούμε μία απλή τοπολογία η οποία θα έχει 2 virtual switches που υποστηρίζουν το OpenFlow, 2 virtual hosts και θα χρησιμοποιεί τον ODL ως remote SDN controller.

Δίνουμε επομένως την εντολή:

```
sudo mn --topo linear,2 --mac --controller=remote,ip=192.168.56.110,port=6633 --switch ovs,protocols=OpenFlow10
```

Όπου ως ip ορίζουμε την ip του ODL προκειμένου να γίνει η σύνδεση.

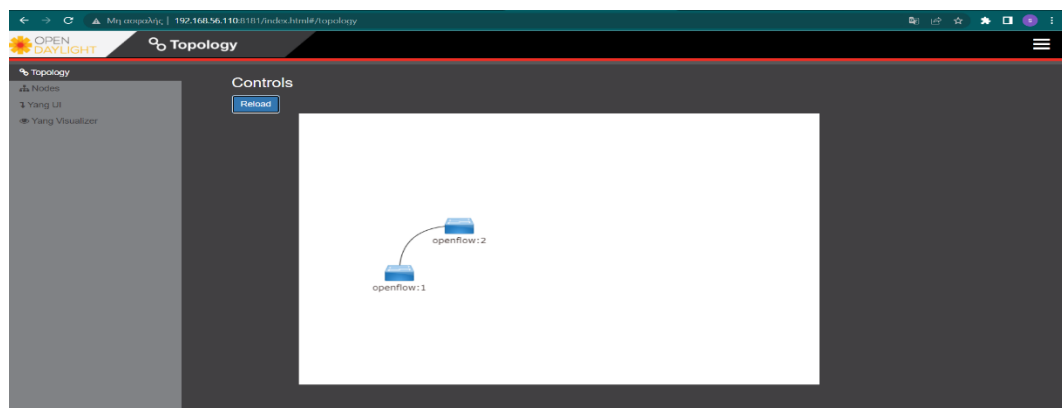
```

mininet@mininet-vm: ~
mininet@mininet-vm:~$ sudo mn --topo linear,2 --mac --controller=remote,ip=192.1
68.56.110,port=6633 --switch ovs,protocols=OpenFlow10
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s2) (s2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet>

```

Εικόνα 4.21: Δημιουργία Τοπολογία στο Mininet

Τώρα πάμε στην τοπολογία μας στο ODL και κάνουμε ανανέωση. Εμφανίστηκαν τα switches συνδεδεμένα μεταξύ τους.



Εικόνα 4.22: Τοπολογία μέσω του OpenDaylight

Προκειμένου να ελέγξουμε την λειτουργία της τοπολογίας, πάμε στο Mininet και δίνουμε την εντολή **pingall** για να δούμε αν οι hosts επικοινωνούν μεταξύ τους σωστά.

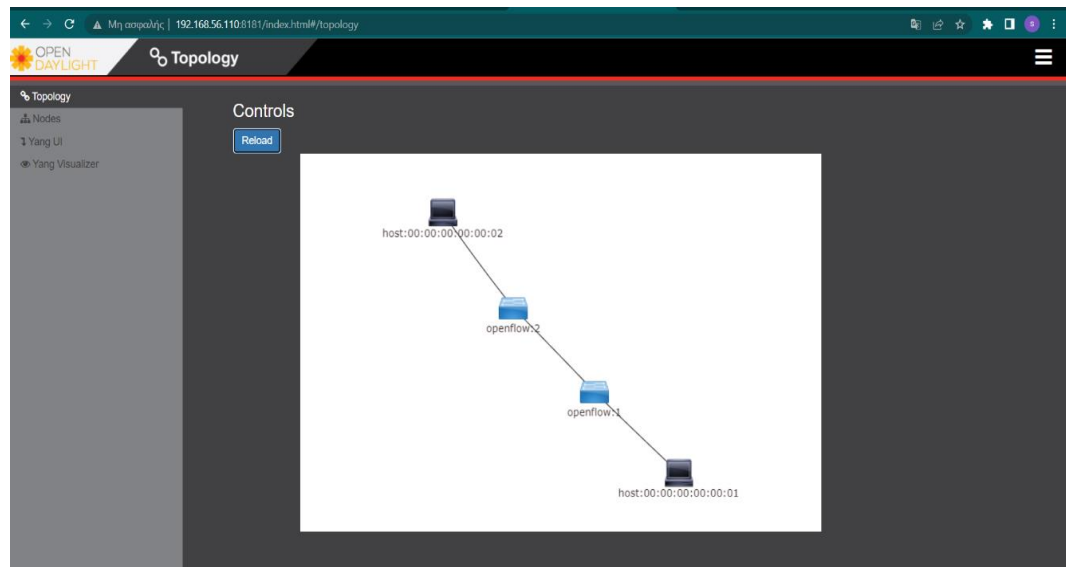
```

mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>

```

Εικόνα 4.23: Αποτέλεσμα εντολής pingall

Εφόσον οι host επικοινωνούν μεταξύ τους, πάμε πάλι στο ODL και κάνουμε ανανέωση την τοπολογία. Τώρα εμφανίζεται ολόκληρο το δίκτυο μας.



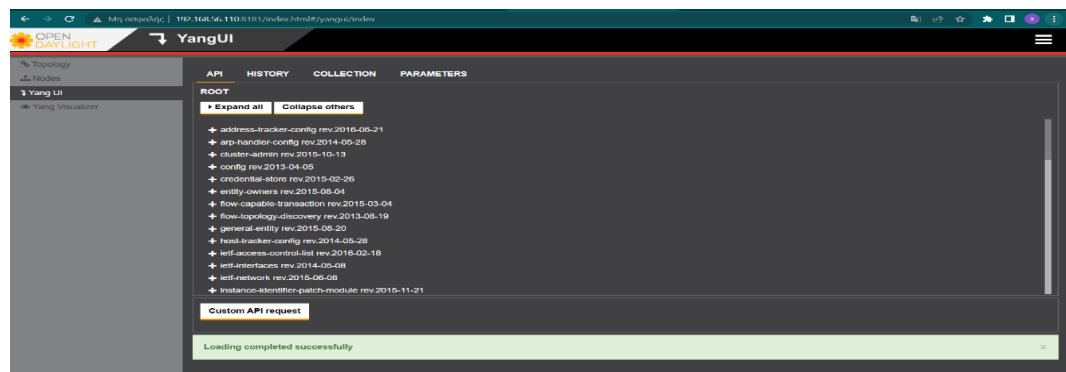
Εικόνα 4.24: Εμφάνιση Τοπολογία στο Περιβάλλον του ODL

YANG UI

Μπορούμε να αποκτήσουμε πρόσβαση στο YANG UI από το ODL.

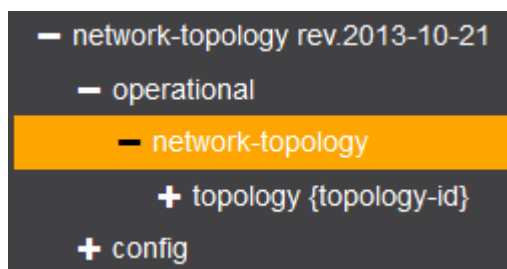
Αριστερά στην οθόνη πατάμε στο YANG UI.

Εμφανίζονται πολλές επιλογές. Θα αποκτήσουμε πρόσβαση στην τοπολογία μας μέσω του Yang.



Εικόνα 4.25: YangUI OpenDaylight

Επιλέγουμε το network topology.



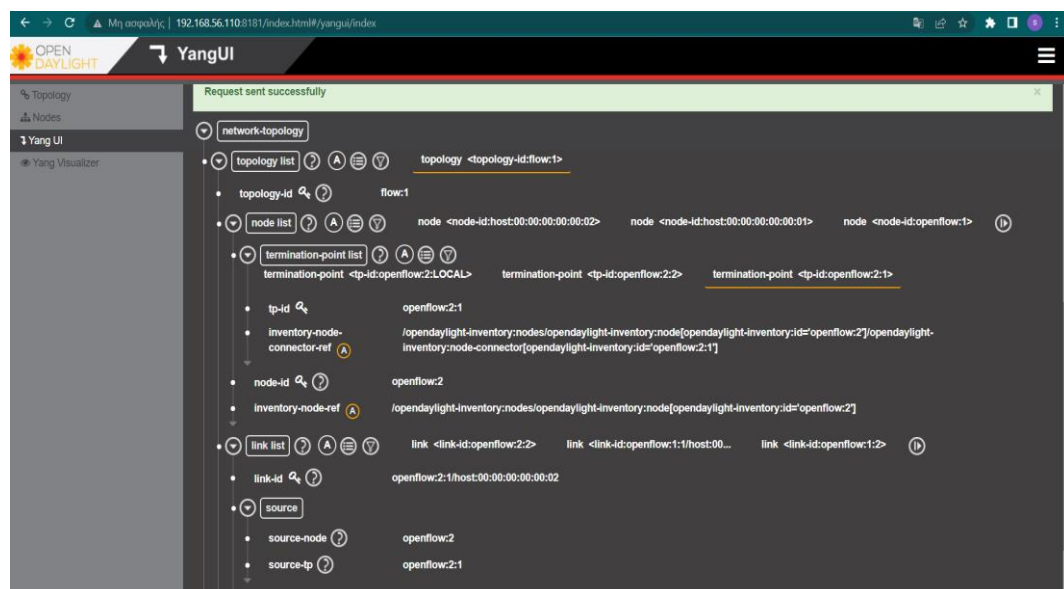
Εικόνα 4.26: Επιλογή network-topology

Το YANG εμφανίζει το CONFREST API URL το οποίο θα μας δώσει πρόσβαση στις πληροφορίες της τοπολογίας.



Εικόνα 4.27: CONFREST API URL

Πατάμε στο Send και βλέπουμε την λειτουργία του δικτύου μας.



Εικόνα 4.28: Εμφάνιση Λειτουργία Δικτύου μέσω του YANG UI

4.8 ONOS

Το Open Network Operating System (ONOS), είναι ένα λειτουργικό σύστημα που έχει σχεδιαστεί για να βοηθήσει τους παρόχους υπηρεσιών δικτύου να κατασκευάσουν δίκτυα επιπέδου μεταφορέα, σχεδιασμένα για υψηλή επεκτασιμότητα, διαθεσιμότητα και απόδοση. Παρόλο που έχει σχεδιαστεί ειδικά για την αντιμετώπιση των αναγκών των παρόχων υπηρεσιών, το ONOS μπορεί επίσης να λειτουργήσει ως επίπεδο ελέγχου δικτύου (SDN) για τοπικά δίκτυα (LAN), και δίκτυα κέντρων δεδομένων.

Ιστορικά, το ONOS, κυκλοφόρησε τον πηγαίο κώδικα ONOS, γραμμένο σε JAVA, στην κοινότητα ανοιχτού κώδικα τον Δεκέμβριο του 2014. Είναι ένας σχετικά νέος controller. Τον Οκτώβριο του 2015, εντάχθηκε στο ίδρυμα Linux ως ένα συνεργατικό έργο ανοιχτού κώδικα Linux. Όπως και με τα περισσότερα έργα

ανοιχτού κώδικα, το ONOS έχει μια σελίδα GitHub όπου οι συνεργάτες μπορούν να συνεισφέρουν αλλαγές στον κώδικα. Μεταξύ των παρόχων υπηρεσιών που συμβάλλουν στην πρωτοβουλία ONOS, είναι η AT&T, η NTT Communications και η SK Telecom. Οι προμηθευτές που συνεισφέρουν στο ONOS περιλαμβάνουν τις Cisco, Ericsson, Intel, NEC, Ciena και Huawei.

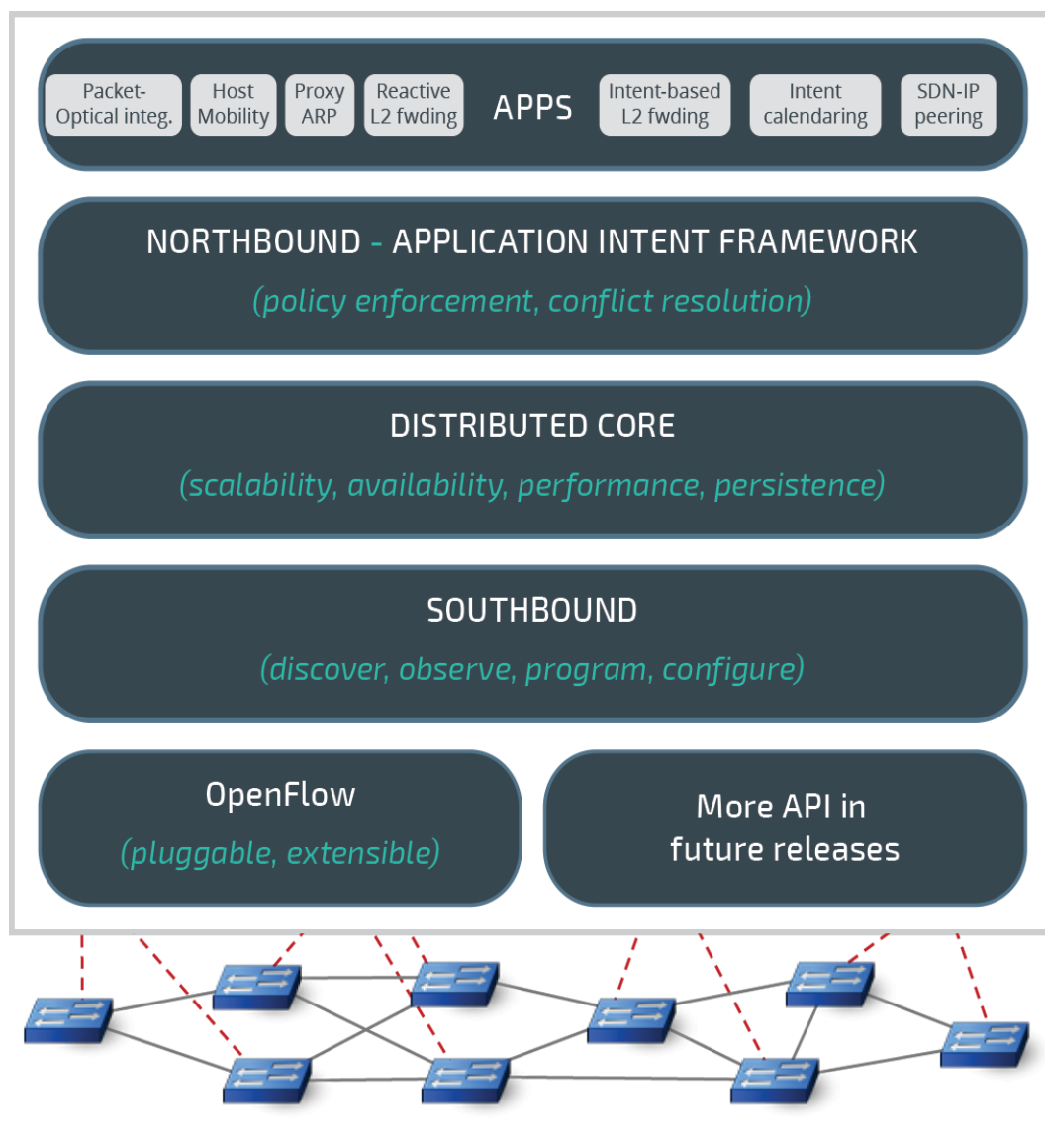
Οι συνεργάτες ON.Lab και ONOS, έχουν βρει περιπτώσεις πολλαπλής χρήσης για το λειτουργικό σύστημα. Ένα από τα πιο γνωστά, είναι το ON.Lab's Central Office Re-architecture as a Datacenter (CORD). Το CORD, δημιουργήθηκε για να μετατρέψει τα κεντρικά γραφεία φορέων τηλεπικοινωνιών σε πιο κλιμακούμενα και ευέλικτα περιβάλλοντα, παρόμοια με τα next-generation data centers. Επιτυγχάνει αυτόν τον μετασχηματισμό-για παράδειγμα-μέσω εικονικών λειτουργιών δικτύου και εξοπλισμού εγκαταστάσεων πελατών.

Το ONOS λοιπόν είναι ένα από τα συστήματα λογισμικού ανοιχτού κώδικα που χρησιμοποιούνται για την διαχείριση του CORD.

4.8.1 Αρχιτεκτονική ONOS

Το ONOS έχει σχεδιαστεί ως αρχιτεκτονική τριών επιπέδων.

- Το επίπεδο 1 αποτελείται από ενότητες που σχετίζονται με πρωτόκολλα που επικοινωνούν με τις συσκευές δικτύου (Southbound).
- Το επίπεδο 2 αποτελείται από τον πυρήνα του ONOS και παρέχει κατάσταση δικτύου χωρίς να βασίζεται σε κάποιο συγκεκριμένο πρωτόκολλο.
- Το επίπεδο 3 αποτελείται από τις ONOS εφαρμογές οι οποίες χρησιμοποιούν πληροφορίες κατάστασης δικτύου που παρουσιάζονται στο επίπεδο 2.



Εικόνα 4.29: Αρχιτεκτονική ONOS(<https://aptira.com/comparison-of-software-defined-networking-sdn-controllers-part-2-open-network-operating-system-onos/>)

- Βόρειες αφαιρέσεις και πρωτόκολλα: Ένα μοναδικό χαρακτηριστικό του ONOS είναι το πλαίσιο εργασίας πρόθεσης, το οποίο επιτρέπει σε μια εφαρμογή να ζητά μια υπηρεσία υψηλού επιπέδου, χωρίς να χρειάζεται να γνωρίζει τις λεπτομέρειες περί του πως εκτελείται αυτή η υπηρεσία. Παρέχονται πληροφορίες κατάστασης στις εφαρμογές ελέγχου δικτύου δια μέσου της βόρεια API είτε συγχρονισμένα, είτε ασύγχρονα.
- Κατανεμημένος πυρήνας: Η κατάσταση των ζεύξεων των υπολογιστών και των συσκευών του δικτύου τηρείται στον κατανεμημένο πυρήνα του ONOS. Το ONOS υλοποιείται ως μια υπηρεσία επάνω σε ένα σύνολο διασυνδεδεμένων εξυπηρετητών, όπου κάθε εξυπηρετητής εκτελεί ένα πανομοιότυπο αντίγραφο του λογισμικού ONOS. Όλο και περισσότεροι εξυπηρετητές προσφέρουν μια αυξημένη χωρητικότητα υπηρεσίας. Ο πυρήνας του ONOS, παρέχει τους μηχανισμούς για αντιγραφή υπηρεσιών και συντονισμό ανάμεσα σε στιγμιότυπα, παρέχοντας στις

εφαρμογές, που είναι από πάνω του και στις συσκευές δικτύου που είναι από κάτω του, την αφαίρεση των λογικά κεντριοποιημένων υπηρεσιών πυρήνα.

- Νότιες αφαιρέσεις και πρωτόκολλα: Οι νότιες αφαιρέσεις καλύπτουν την ετερογένεια των υποκείμενων υπολογιστών, ζεύξεων, μεταγωγέων και πρωτοκόλλων, επιτρέποντας στον κατανεμημένο πυρήνα να μην έχει γνώση τόσο για τις συσκευές, όσο και για τις υπηρεσίες.

4.8.2 Λειτουργία ONOS

Πώς όμως λειτουργεί ο controller; Ο πυρήνας ONOS βασίζεται στην αρθρωτή αρχιτεκτονική, σε αντίθεση με ένα ολοκληρωμένο σύστημα που θολώνει την διαίρεση μεταξύ των συστατικών του. Αυτή η αρθρωτότητα διατηρεί τις ροές εργασίας Βορρά-Νότου διαχωρισμένες από τις ροές εργασίας Ανατολής-Δύσης, ενώ επιτρέπει επίσης την ευκολότερη προσαρμογή για ολόκληρο το σύστημα.

Επειδή οι πάροχοι υπηρεσιών, απαιτούν τη δυνατότητα κλιμάκωσης των δικτύων τους, ο ελεγκτής ONOS μπορεί να κλιμακωθεί για να φιλοξενήσει ένα φυσικά κατανεμημένο σύστημα συσκευών. Αυτό, επιτρέπει στους παρόχους υπηρεσιών να προσθέτουν νέους διακόπτες ή εξαρτήματα, χωρίς να διαταράσσουν το υπόλοιπο σύστημα. Ακόμη, η κατανεμημένη αρχιτεκτονική μειώνει την αστοχία του δικτύου, καθώς πανομοιότυπες εμφανίσεις μπορούν να εντοπιστούν εκεί όπου μια άλλη αστοχία έχει εμφανιστεί. Με την σειρά του αυτό, οδηγεί σε υψηλή διαθεσιμότητα.

Ενώ ο πυρήνας ONOS διανέμεται για να παρέχει προσβασιμότητα σε κάθε συσκευή μεταγωγής δικτύου, ο ελεγκτής ONOS παραμένει λογικά συγκεντρωμένος και οι ξεχωριστές υποδιαιρέσεις ή περιπτώσεις στην πλήρη αρχιτεκτονική ONOS μπορούν να θεωρηθούν και να προσπελαστούν ως ένα ενιαίο σύστημα.

Το ONOS, για την συνολική ορατότητα και διαχείριση του συστήματος, παρέχει μια σχετικά απλή GUI. Επιπλέον, το ONOS διαθέτει διεπαφές προγράμματος εφαρμογών (API) με κατεύθυνση βόρεια και νότια, οι οποίες βασίζονται στην αφαίρεση για την πρόληψη της διαμόρφωσης και του κλειδώματος πρωτοκόλλων για εφαρμογές και συσκευές ανίστοιχα.

Για αλληλεπιδράσεις προς τον βορρά, το ONOS χρησιμοποιεί το Intent Framework subsystem, το οποίο επιτρέπει στις εφαρμογές να καθορίσουν τι χρειάζονται από το σύστημα, για παράδειγμα αν μία εφαρμογή χρειάζεται περισσότερο bandwidth. Μόλις μια εφαρμογή δηλώσει τι χρειάζεται, το σύστημα λειτουργεί για να ρυθμιστεί ανάλογα.

Το ONOS έχει σχεδιαστεί για να υποστηρίζει μια εισροή περίπου ενός εκατομμυρίου αιτημάτων πρόθεσης εφαρμογής ανά δευτερόλεπτο. Αυτή η δυνατότητα διατηρεί την υψηλή απόδοση του συστήματος για αιτήματα εφαρμογής και τον χαμηλό χρόνο αναμονής που χρειάζονται οι πάροχοι υπηρεσιών.

Κάθε περίπτωση ONOS αλληλεπιδρά με το περιβάλλον του δικτύου και τις συσκευές ενός southbound API, που επικοινωνεί με στοιχεία χαμηλότερου επιπέδου. Ο πυρήνας ανακαλύπτει ποια πρωτόκολλα μπορούν να χρησιμοποιηθούν για να αλληλεπιδράσουν με την συσκευή, και το southbound API χρησιμοποιεί αυτό το πρωτόκολλο για να αλληλεπιδράσει με την συσκευή. Αυτό το API είναι αφηρημένο, έτσι το ONOS παραμένει ανεπηρέαστο από συγκεκριμένα πρωτόκολλα όπως το OpenFlow, το NETCONF, ή διεπαφές γραμμής εντολών.

4.8.3 Docker

Το Docker είναι μία πλατφόρμα ανοιχτού κώδικα. Επιτρέπει στους προγραμματιστές να συσκευάζουν εφαρμογές σε δοχεία-τυποποιημένα(containers-standardized) εκτελέσιμα αρχεία που συνδυάζουν μετάβαση στο περιεχόμενο πηγαίου κώδικα εφαρμογής με τις βιβλιοθήκες του λειτουργικού συστήματος και τις εξαρτήσεις που απαιτούνται για την εκτέλεση αυτού του κώδικα σε οποιοδήποτε περιβάλλον.

Τα containers απλοποιούν την παράδοση των κατανεμημένων εφαρμογών και έχουν γίνει όλο και πιο δημοφιλή καθώς οι οργανισμοί στρέφονται προς την ανάπτυξη του νέφους και τα υβριδικά περιβάλλοντα πολλαπλών νεφών. Η δημιουργία containers, μπορεί γίνει και χωρίς την χρήση του Docker, όμως το Docker κάνει ευκολότερη, απλούστερη και ασφαλέστερη την κατασκευή, την ανάπτυξη και την διαχείριση των containers.

Το Docker στην ουσία είναι μία εργαλειοθήκη που επιτρέπει στους προγραμματιστές να κατασκευάζουν, να αναπτύσσουν, να εκτελούν και να ενημερώνουν τα containers χρησιμοποιώντας απλές εντολές και αυτοματισμό εξοικονόμησης εργασίας μέσω ενός μόνο API.

Εάν υπάρχει παλαιότερη έκδοση του docker, τότε την κάνουμε uninstall:

```
sudo apt-get remove docker
```

Η εγκατάσταση σε περιβάλλον Linux είναι πολύ εύκολη και γίνεται ως εξής:

1. `sudo apt-get update` Ενημερώνει το σύστημα
2. `sudo apt-get -y install docker.io` Κατεβάζει το πακέτο docker.io
3. `sudo ln -sf /usr/bin/docker.io /usr/local/bin/docker` Δημιουργεί έναν συμβολικό σύνδεσμο όπου είναι εγκατεστημένα τα αρχεία του πακέτου docker io στον κατάλογο /usr/local/bin/docker, ώστε ένας χρήστης linux να μπορεί να εκτελέσει το docker cli εκτελώντας το docker.

4.8.4 Εγκατάσταση-Παρουσίαση ONOS

Στην συνέχεια θα γίνει μία παρουσίαση για τον τρόπο εγκατάστασης του ONOS χρησιμοποιώντας το Docker, καθώς επίσης και για το πως γίνεται ο έλεγχος μίας τοπολογίας μέσω του Mininet μέσω του ONOS controller.

Ανοίγουμε την εικονική μηχανή που έχουμε εγκαταστήσει για το Mininet.

Κάνουμε update στο σύστημα.

```
sudo apt-get update
```

Έχουμε εγκαταστήσει το Docker, όπως αναφέρθηκε στο 4.7.3.

Κατεβάζουμε το onos.

```
sudo docker pull onosproject/onos
```

Για να αποκτήσουμε πρόσβαση στα ONOS CLI, GUI, NETCONF και Debugger ports δίνουμε την παρακάτω εντολή.

```
sudo docker run -t -d -p 8181:8181 -p 8101:8101 -p 5005:5005 -p 830:830 --name onos onosproject/onos
```

Για να βεβαιωθούμε για την ορθή εγκατάσταση.

```
sudo docker ps
```

Αν όλα πήγαν καλά, θα πρέπει να εμφανιστεί κάτι παρόμοιο:

```
mininet@mininet-vm:~$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	NAMES
b4f3f8d71531	onosproject/onos:latest	"/bin/onos-service	11 seconds ago	onos
	Up 10 seconds	0.0.0.0:830->830/tcp, 6640/tcp, 0.0.0.0:5005->5005/tcp, 0.0.0.0:8101->8101/tcp, 6653/tcp, 0.0.0.0:8181->8181/tcp, 9876/tcp		

4.30: Λεπτομέρειες Εγκατεστημένου docker

Μπορούμε να δούμε ότι το ONOS τρέχει και ότι το UI είναι προσβάσιμο δίνοντας την εντολή:

```
wget -O - http://localhost:8181/onos/ui > /dev/null
```

Όπου localhost είναι η IP για το Mininet. Για να βρούμε την IP μπορούμε να γράψουμε:

```
ip addr | grep eth0
```

Για να δούμε το περιβάλλον του ONOS, ανοίγουμε τον browser και δίνουμε το URL:

<http://localhost:8181/onos/ui>

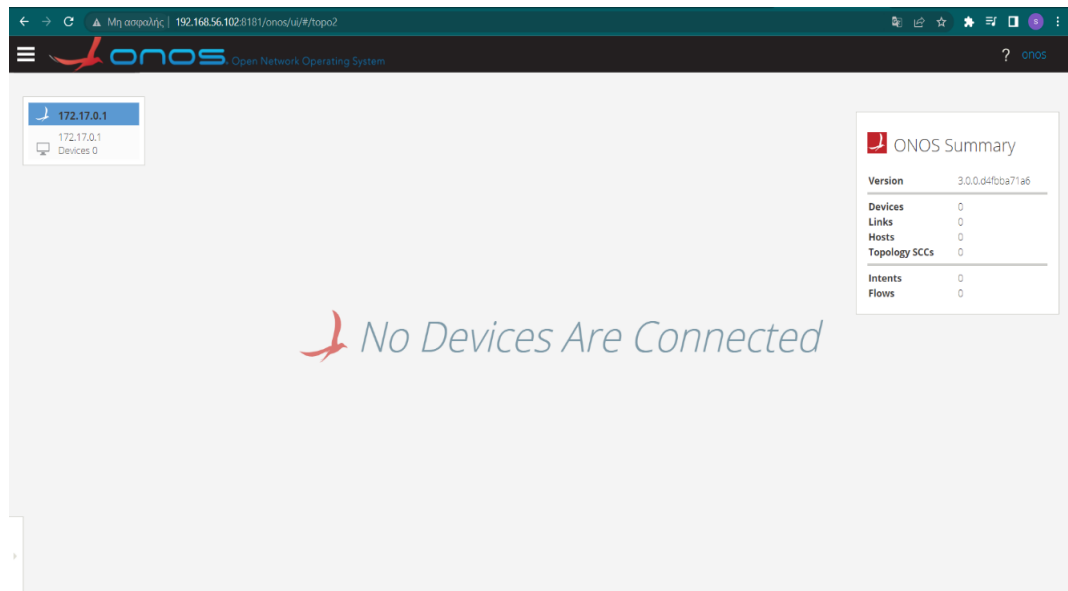
User: onos

Password: rocks



Εικόνα 4.31: Σύνδεση στο Περιβάλλον του ONOS

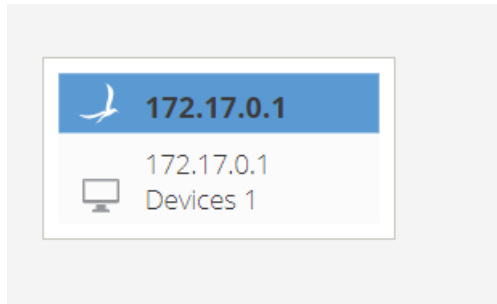
Έχοντας συνδεθεί, μπορούμε να δούμε το περιβάλλον του ONOS και την τοπολογία μας. Αυτή τη στιγμή, δεν υπάρχει κάποια τοπολογία από την στιγμή που δεν έχουμε δημιουργήσει μία.



Εικόνα 4.32: Περιβάλλον ONOS

Μπορούμε να διαμορφώσουμε το ONOS και μέσω του CLI. Θα χρησιμοποιήσουμε το ssh για να αποκτήσουμε πρόσβαση στο ONOS, δίνοντας την ακόλουθη εντολή: `ssh -p 8101 karaf@<ONOS_IP>`

Το ONOS_IP το βρίσκουμε πάνω αριστερά στο GUI του ONOS.



Εικόνα 4.33: Εύρεση Διεύθυνσης IP στο ONOS

Εάν εμφανιστεί το παρακάτω warning:

```
mininet@mininet-vm:~$ ssh -p 8101 karaf@172.17.0.1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
68:3b:01:dc:e6:8b:51:74:7c:e6:02:7d:2f:03:09:f0.
Please contact your system administrator.
Add correct host key in /home/mininet/.ssh/known_hosts to get rid of this message.
Offending RSA key in /home/mininet/.ssh/known_hosts:1
  remove with: ssh-keygen -f "/home/mininet/.ssh/known_hosts" -R [172.17.0.1]:8101
RSA host key for [172.17.0.1]:8101 has changed and you have requested strict checking.
Host key verification failed.
```

Εικόνα 4.34: Warning κατά την Εγκατάσταση του ONOS

Μπορούμε να το ξεπεράσουμε δίνοντας την εντολή:

```
rm .ssh/known_hosts
```

Μόλις δώσουμε την εντολή αυτή θα μας ζητήσει έναν κωδικό. Ο κωδικός είναι: **karaf**

```
Welcome to Open Network Operating System (ONOS)!

  ONOS

Documentation: wiki.onosproject.org
Tutorials:    tutorials.onosproject.org
Mailing lists: lists.onosproject.org

Come help out! Find out how at: contribute.onosproject.org

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'logout' to exit ONOS session.

karaf@root > █
```

Εικόνα 4.35: Εκκίνηση ONOS στο Τερματικό

Από το σημείο αυτό, μπορούμε να διαμορφώσουμε τον controller. Με την εντολή `apps -s` εμφανίζουμε όλες τις εφαρμογές που υποστηρίζει ο controller.

Μπορούμε επίσης να ενεργοποιήσουμε το open flow application δίνοντας την εντολή:

```
app activate org.onosproject.openflow
```

```
karaf@root > app activate org.onosproject.openflow
Activated org.onosproject.openflow
karaf@root > █
```

Εικόνα 4.36: Ενεργοποίηση OpenFlow Application

Στην συνέχεια, πρέπει να κάνουμε Install το feature onos-apps-fwd το οποίο επιτυγχάνεται δίνοντας την εντολή:

```
feature:install onos-apps-fwd
```

Μπορούμε επίσης να δούμε ποιες εφαρμογές είναι ενεργοποιημένες με την εντολή:

```
apps -s -a
```

```
karaf@root > apps -s -a
* 3 org.onosproject.gui2          3.0.0.SNAPSHOT ONOS GUI2
* 4 org.onosproject.drivers       3.0.0.SNAPSHOT Default Drivers
* 31 org.onosproject.optical-model 3.0.0.SNAPSHOT Optical Network Model
* 47 org.onosproject.hostprovider 3.0.0.SNAPSHOT Host Location Provider
* 48 org.onosproject.lldpprovider 3.0.0.SNAPSHOT LLDP Link Provider
* 49 org.onosproject.openflow-base 3.0.0.SNAPSHOT OpenFlow Base Provider
* 50 org.onosproject.openflow     3.0.0.SNAPSHOT OpenFlow Provider Suite
karaf@root > █
```

Εικόνα 4.37: Ενεργοποιημένες Εφαρμογές

Ακόμη, τις εφαρμογές μπορούμε να τις βρούμε από το GUI το ONOS, επιλέγοντας πάνω αριστερά τις τρεις γραμμές που υπάρχουν, και στην συνέχεια επιλέγοντας το Applications.

	Title	App ID	Version	Category	Origin
✓	Default Drivers	org.onosproject.drivers	3.0.0.SNAPSHOT	Drivers	ONOS Community
✓	Host Location Provider	org.onosproject.hostprovider	3.0.0.SNAPSHOT	Provider	ONOS Community
✓	LLDP Link Provider	org.onosproject.lldpprovider	3.0.0.SNAPSHOT	Provider	ONOS Community
✓	ONOS GUI2	org.onosproject.gui2	3.0.0.SNAPSHOT	Graphical User Interface	ONOS Community
✓	OpenFlow Base Provider	org.onosproject.openflow-base	3.0.0.SNAPSHOT	Provider	ONOS Community
✓	OpenFlow Provider Suite	org.onosproject.openflow	3.0.0.SNAPSHOT	Provider	ONOS Community
✓	Optical Network Model	org.onosproject.optical-model	3.0.0.SNAPSHOT	Optical	ONOS Community
■	Access Control Lists	org.onosproject.adl	3.0.0.SNAPSHOT	Security	ONOS Community
■	Arista Drivers	org.onosproject.drivers.arista	3.0.0.SNAPSHOT	Drivers	ONOS Community
■	Artemis	org.onosproject.artemis	3.0.0.SNAPSHOT	Monitoring	ONOS Community
■	BGP Router	org.onosproject.bgprouter	3.0.0.SNAPSHOT	Traffic Engineering	ONOS Community
■	BMV2 Drivers	org.onosproject.drivers.bmv2	3.0.0.SNAPSHOT	Drivers	ONOS Community
■	Barefoot Drivers	org.onosproject.drivers.barefoot	3.0.0.SNAPSHOT	Drivers	ONOS Community
■	Basic Optical Drivers	org.onosproject.drivers.optical	3.0.0.SNAPSHOT	Drivers	ONOS Community
■	Basic Pipelines	org.onosproject.pipelines.basic	3.0.0.SNAPSHOT	Pipeline	ONOS Community
■	CORD Support	org.onosproject.cord-support	3.0.0.SNAPSHOT	Integration	ONOS Community
■	Castor	org.onosproject.castor	3.0.0.SNAPSHOT	Utility	ONOS Community

Εικόνα 4.38: Ενεργοποιημένες Εφαρμογές μέσω του Περιβάλλον του ONOS

Προκειμένου να δούμε πως συνδέεται ο controller με την τοπολογία που θα φτιάξουμε, ανοίγουμε μία νέα σύνδεση στο mininet, χωρίς να κλείσουμε την ήδη ανοιχτή σύνδεση που έχουμε εκκινήσει τον controller.

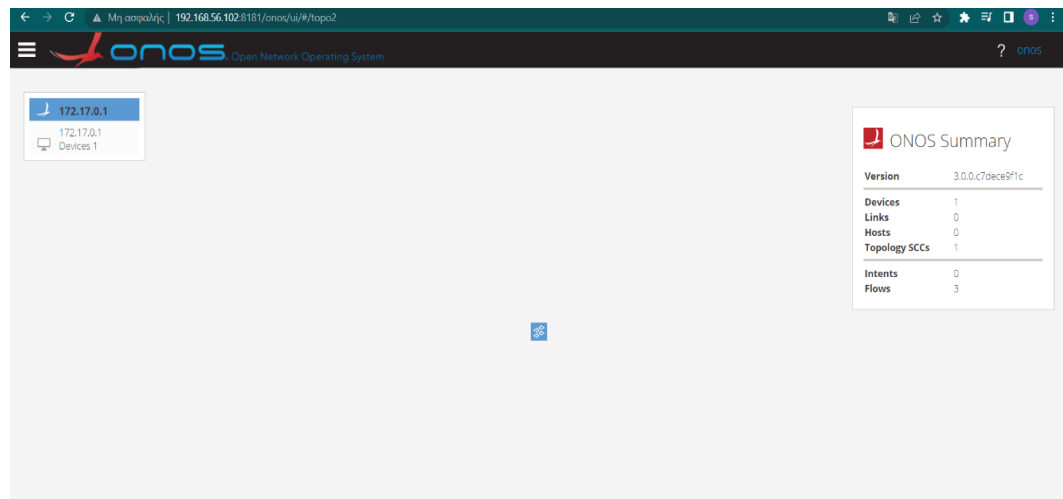
Στην νέα αυτή σύνδεση, προκειμένου να δημιουργηθεί μία απλή τοπολογία, δίνουμε την εντολή:

```
sudo mn --controller remote,ip=<ONOS_IP>
```

Όπου το ONOS_IP θα το βρούμε στο GUI του ONOS, πάνω αριστερά.

Η τοπολογία αυτή περιέχει 2 Hosts, ένα switch και σύνδεση με τον controller ONOS.

Αφού δημιουργηθεί σωστά, θα πρέπει να έχει εμφανιστεί το switch στην οθόνη.



Εικόνα 4.39: Τοπολογία στο Περιβάλλον του ONOS

Στην συνέχεια κάνουμε Ping τους Hosts μέσω του mininet, για να δούμε αν επικοινωνούν σωστά.

```
mininet@mininet-vm:~$ sudo mn --controller remote,ip=172.17.0.1
*** Creating network
*** Adding controller
Connecting to remote controller at 172.17.0.1:6653
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

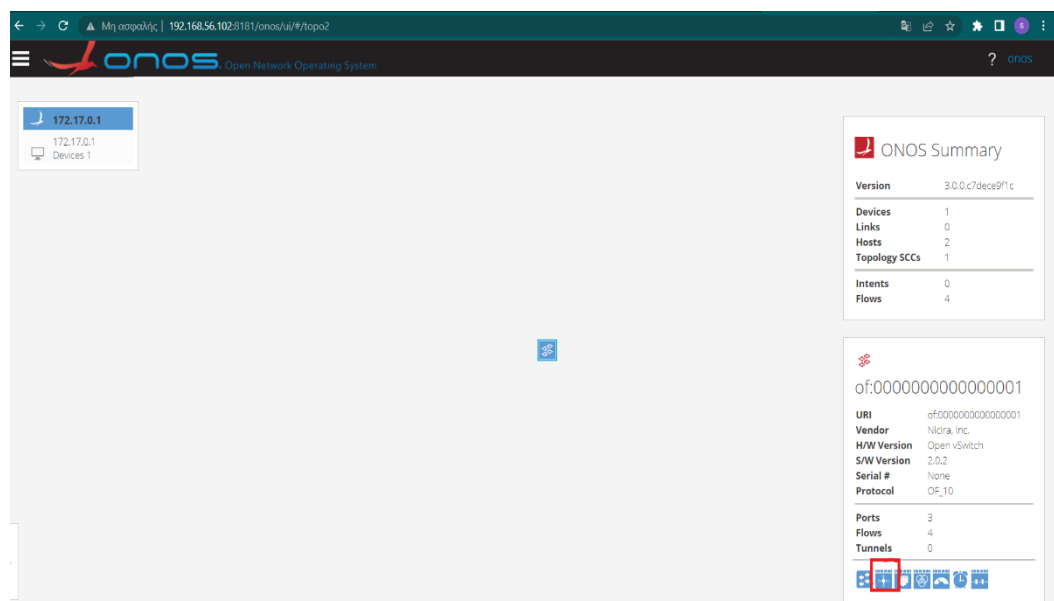
Εικόνα 4.40: Δημιουργία Τοπολογίας μέσω του Mininet

Εφόσον οι hosts επικοινωνούν μεταξύ τους, σημαίνει ότι ο controller λειτουργεί κανονικά.

Πηγαίνοντας τώρα στο GUI του ONOS, βλέπουμε ότι υπάρχει μία συσκευή συνδεδεμένη με τον controller, το switch s1.

Κάνοντας κλικ πάνω στο switch εμφανίζεται ένα πλαίσιο κάτω δεξιά. Μέσω αυτού, υπάρχει η δυνατότητα να ελέγξουμε τα flows της τοπολογίας, να δούμε λεπτομέρειες τις συσκευής που είναι συνδεδεμένη με τον controller, να δούμε τα group που υπάρχουν για την συσκευή (στην προκειμένη τοπολογία δεν υπάρχουν group), να ελέγξουμε για τα alarms της τοπολογίας, και να δούμε το pipeconf της συσκευής, κάνοντας διπλό κλικ σε κάθε εικονίδιο που εμφανίζεται στο πλαίσιο αντίστοιχα.

Για παράδειγμα, επιλέγοντας το εικονίδιο για τα flow, εμφανίζονται στην οθόνη:



Εικόνα 4.41: Επιλογή Εικονιδίου flow

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	2.255	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	0	2.255	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	8	2.255	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	10	2.255	5	0	ETH_TYPE:ip4	imm[OUTPUT:CONTROLLER], cleared:true	*core

Εικόνα 4.42: Αποτέλεσμα flow

4.9 Flowvisor

Το Flowvisor είναι ένας ειδικός ελεγκτής SDN που στοχεύει στον διαχωρισμό ενός φυσικού SDN σε πολλαπλά εικονικά slice, και λειτουργεί ως ελεγκτής για τα switches και ως OpenFlow switch για τους controllers. Αναπτύχθηκε και χρησιμοποιήθηκε στο δίκτυο παραγωγής του Πανεπιστημίου του Στάνφορντ από το OpenNetworkingLab μεταξύ 2009 και 2013.

Το Flowvisor είναι ένας controller OpenFlow ειδικού σκοπού, ο οποίος λειτουργεί ως διαφανής διαμεσολαβητής μεταξύ των OpenFlow switches και των πολλαπλών OpenFlow controllers. Δημιουργεί slices στο δίκτυο, καθένα εκ των οποίων τον έλεγχο τον αναθέτει σε διαφορετικό controller. Τα slices μπορούν να οριστούν από οποιονδήποτε συνδυασμό switch ports (layer 1), διεύθυνσης ή τύπου src/dst ethernet (layer 2), διεύθυνσης ή τύπου src/dst IP (layer 3), και θύρας src/dst TCP/UDP ή κωδικού τύπου ICMP (layer 4). Ακόμη το Flowvisor επιβάλλει απομόνωση μεταξύ των slices, επομένως ένα slice δεν μπορεί να ελέγξει την κυκλοφορία του άλλου.

Αν και εξακολουθεί να είναι αποτελεσματικό και λειτουργικό, το Flowvisor έχει σταματήσει να συντηρείται πλέον και δεν θα πρέπει να λειτουργεί σε δίκτυα παραγωγής για τρεις κύριους λόγους:

- 1) Υποστηρίζει μόνο το OpenFlow 1.0, ενώ το OpenFlow 1.5 έχει κυκλοφορήσει ήδη εδώ και 2 χρόνια.
- 2) Υπάρχουν αρκετά τρωτά σημεία σχετικά με την απομόνωση των slices.
- 3) Στερείται επεκτασιμότητας καθώς οι συνιστώμενες απαιτήσεις για την λειτουργία του Flowvisor είναι 4 πυρήνες επεξεργαστή και 4GB Java heap.

Ωστόσο το Flowvisor έχει χρησιμοποιηθεί από το 2009 και εξακολουθεί να χρησιμοποιείται σε πολλαπλά ερευνητικά δίκτυα όπου οι αδυναμίες απομόνωσης των slices μπορεί να μην αποτελούν μεγάλη ανησυχία.

4.9.1 Flowvisor – SDN

Το Flowvisor, στην λειτουργία του SDN λειτουργεί ανάμεσα από το επίπεδο υποδομής και το επίπεδο ελέγχου, και παρέχει όλες τις λειτουργίες εικονικοποίησης που χρειάζονται. Το Flowvisor είναι αυτό στο οποίο θα ανατεθούν οι πόροι που χρειάζονται προκειμένου να δημιουργούν τα slice στο δίκτυο. Αυτό λοιπόν που συμβαίνει μετά την δημιουργία των slice, είναι οι SDN controller να έχουν γνώση μόνο για τα slice τα οποία έχει δημιουργήσει το flowvisor, επομένως, κάθε φορά που θέλουν να στείλουν ένα μήνυμα ελέγχου μέσω του OpenFlow, αυτό προωθείται στο flowvisor οποίος με την σειρά του προωθεί τα μηνύματα αυτά στις συσκευές που είναι οι προορισμοί. Η ίδια διαδικασία ισχύει και για το αντίστροφο. Εάν θέλει δηλαδή μία συσκευή να μάθει για ένα πακέτο, υποβάλλοντας ερώτημα, τότε το ερώτημα αυτό προωθείται στο flowvisor το οποίο με την σειρά του το προωθεί στον κατάλληλο controller.

4.9.2 Εγκατάσταση Flowvisor

Εκκινούμε την εικονική μηχανή του Mininet που εγκαταστήσαμε στο 4.6.1 .

Κάνουμε ένα update στην μηχανή μας.

```
sudo apt-get update
```

Εγκαθιστούμε το openjdk που υποστηρίζει το Flowvisor. Αυτό είναι το openjdk6.

```
sudo apt-get install openjdk-6-jdk
```

Στην συνέχεια πρέπει να κατεβάσουμε το ant. Το ant, είναι μια βιβλιοθήκη Java και ένα εργαλείο γραμμής εντολών, η αποστολή του οποίου είναι να οδηγεί διαδικασίες που περιγράφονται σε αρχεία δόμησης ως στόχοι και σημεία επέκτασης που εξαρτώνται το ένα από το άλλο. Η κύρια γνωστή χρήση του ant είναι η κατασκευή εφαρμογών Java. Το ant παρέχει έναν αριθμό ενσωματωμένων εργασιών που επιτρέπουν την μεταγλώττιση, την συναρμολόγηση, την δοκιμή και την εκτέλεση εφαρμογών Java. Ακόμη μπορεί να χρησιμοποιηθεί αποτελεσματικά για τη δημιουργία εφαρμογών εκτός Java, για παράδειγμα εφαρμογών C και C++.

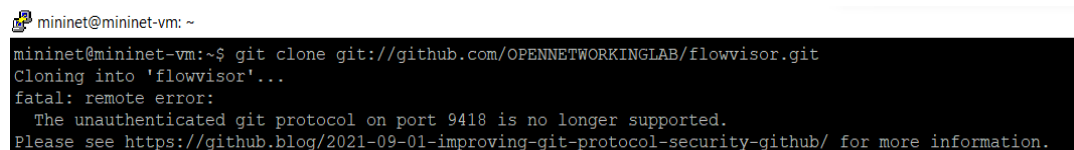
Γενικότερα το ant μπορεί να χρησιμοποιηθεί για να πιλοτάρει οποιοδήποτε είδος διαδικασίας που μπορεί να περιγραφεί από την άποψη των στόχων και των καθηκόντων.

```
sudo apt-get install ant
```

Κατεβάζουμε τον πηγαίο κώδικα του Flowvisor.

```
git clone git://github.com/OPENNETWORKINGLAB/flowvisor.git
```

Εάν εμφανιστεί το παρακάτω error:



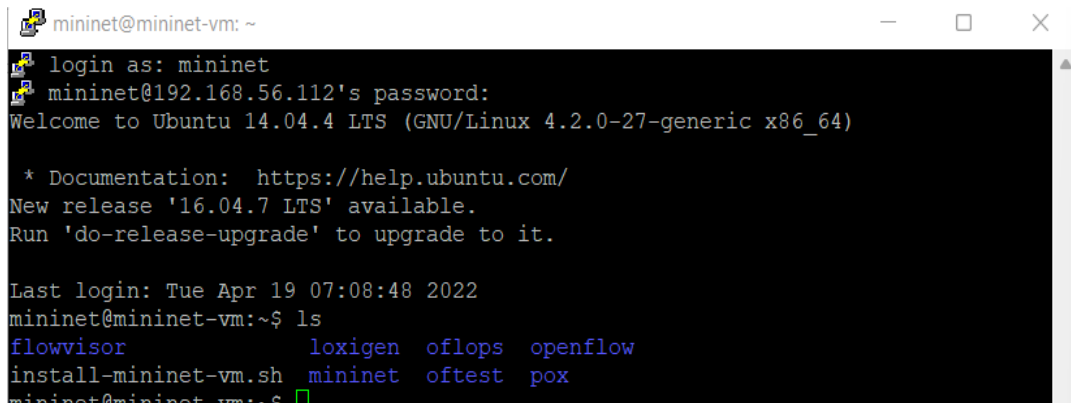
```
mininet@mininet-vm: ~
mininet@mininet-vm:~$ git clone git://github.com/OPENNETWORKINGLAB/flowvisor.git
Cloning into 'flowvisor'...
fatal: remote error:
The unauthenticated git protocol on port 9418 is no longer supported.
Please see https://github.blog/2021-09-01-improving-git-protocol-security-github/ for more information.
```

Εικόνα 4.43: Error στο Mininet

τότε θα πρέπει να γίνει μία αλλαγή στο πρωτόκολλο. Το error αυτό οφείλεται στην ασφάλεια που έχει το git protocol στο github. Ο ευκολότερος τρόπος για να επιλυθεί το error αυτό και να προχωρήσουμε είναι να μεταβούμε από το git στο HTTPS. Αυτό γίνεται με την παρακάτω εντολή.

```
git config --global url.https://.insteadOf git://
```

Εφόσον έχουν γίνει σωστά τα παραπάνω, δίνουμε την εντολή **ls** για να δούμε αν κατέβηκε το αρχείο που θέλουμε.



```

mininet@mininet-vm: ~
login as: mininet
mininet@192.168.56.112's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Apr 19 07:08:48 2022
mininet@mininet-vm:~$ ls
flowvisor      loxigen  oflops  openflow
install-mininet-vm.sh  mininet  oftest  pox
mininet@mininet-vm:~$

```

Εικόνα 4.44: Αποτέλεσμα Εντολής ls

Θα πρέπει να εμφανίζεται ο φάκελος **flowvisor**.

Αφού εμφανίζεται, μπαίνουμε στον φάκελο του flowvisor δίνοντας την εντολή.

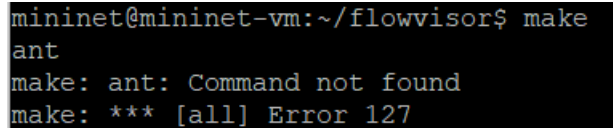
```
cd flowvisor
```

Μέσα στον φάκελο αυτόν, υπάρχει ο πηγαίος κώδικας προκειμένου να εγκατασταθεί το flowvisor. Για να τον εγκαταστήσουμε δίνουμε τις εντολές:

```
make
```

```
sudo make install
```

ΠΡΟΣΟΧΗ!! Εάν δεν έχει εγκατασταθεί σωστά το **ant**, θα εμφανιστεί το παρακάτω error:



```

mininet@mininet-vm:~/flowvisor$ make
ant
make: ant: Command not found
make: *** [all] Error 127

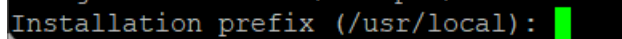
```

Εικόνα 4.45: Error ant

Επομένως θα πρέπει να εγκαταστήσουμε το ant ξανά.

Όσο γίνεται η εγκατάσταση ζητείται να δώσουμε κάποιες τιμές.

Αρχικά ζητείται το εξής:

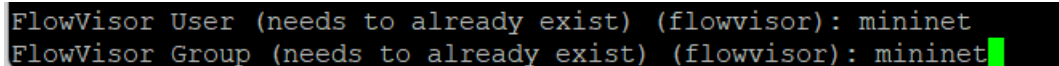


```
Installation prefix (/usr/local):
```

Εικόνα 4.46: Ορισμός /usr/local

στο οποίο απλά πατάμε Enter.

Στην συνέχεια ζητείται να ορίσουμε το Flowvisor Users και το Flowvisor Group, τα οποία θα πάρουν και τα 2 τις τιμές mininet.



```

FlowVisor User (needs to already exist) (flowvisor): mininet
FlowVisor Group (needs to already exist) (flowvisor): mininet

```

Εικόνα 4.47: Ορισμός Flowvisor Users/Group

Έπειτα ζητείται να ορίσουμε κατάλογο εγκατάστασης, στο οποίο πατάμε Enter για να παραμείνει ο προκαθορισμένος.

```
Install to different root directory ()
```

Εικόνα 4.48: Ορισμός Καταλόγου Εγκατάστασης

Τέλος, ζητείται να δώσουμε έναν password. Μπορούμε να βάλουμε οποιοδήποτε θέλουμε. Στην προκειμένη περίπτωση εγώ τον άφησα κενό πατώντας Enter.

```
Enter password for account 'fvadmin' on the flowvisor:
```

Εικόνα 4.49: Ορισμός Password

Συνεχίζουμε ορίζοντας τον ιδιοκτήτη με τα δικαιώματα του. Δίνουμε τις εντολές:

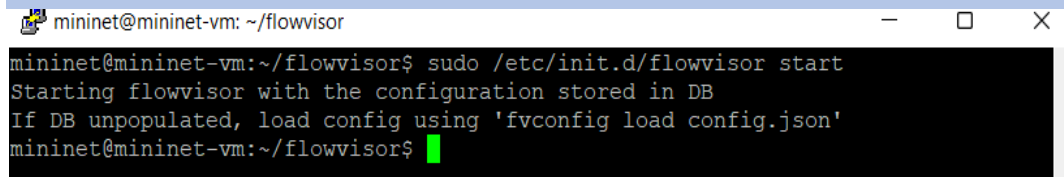
```
sudo chown mininet:mininet -R /usr/local/share/db
sudo chmod -R 777 /usr/local/share/db
```

Κατεβάζουμε το configuration file.

```
sudo fvconfig load /etc/flowvisor/config.json
```

Εφόσον όλα τα παραπάνω έχουν γίνει σωστά, είμαστε σε θέση να εκκινήσουμε το Flowvisor:

```
sudo /etc/init.d/flowvisor start
```



```
mininet@mininet-vm: ~/flowvisor$ sudo /etc/init.d/flowvisor start
Starting flowvisor with the configuration stored in DB
If DB unpopulated, load config using 'fvconfig load config.json'
mininet@mininet-vm: ~/flowvisor$
```

Εικόνα 4.50: Εκκίνηση FlowVisor controller

Ενεργοποιούμε τον controller της τοπολογίας (εφόσον έχει φτιαχτεί):

```
fvctl set-config --enable-topo-ctrl
```

Ακόμη, μπορούμε να δούμε το configuration με την εντολή:

```
fvctl get-config
```

θα ζητηθεί ένας κωδικός, ο οποίος είναι αυτός που ορίσαμε κατά το Installation.

Αποτέλεσμα της εντολής αυτής θα πρέπει να είναι κάτι τέτοιο:


```

mininet@mininet-vm:~/flowvisor$ fvctl get-config
Password:
{
  "api_jetty_webserver_port": 8081,
  "api_webserver_port": 8080,
  "checkpointing": false,
  "config_name": "default",
  "db_version": "2",
  "enable-topo-ctrl": false,
  "flood-perm": {
    "dpid": "all",
    "slice-name": "fvadmin"
  },
  "flow-stats-cache": 30,
  "flowmod-limit": {
    "fvadmin": {
      "any": null
    }
  },
  "host": "localhost",
  "log_facility": "LOG_LOCAL7",
  "log_ident": "flowvisor",
  "logging": "NOTE",
  "stats-desc": false,
  "track-flows": false,
  "version": "flowvisor-1.4.0"
}
mininet@mininet-vm:~/flowvisor$ █

```

Εικόνα 4.51: Αποτέλεσμα configuration του flowvisor

Όταν θέλουμε να τερματίσουμε το Flowvisor δίνουμε την εντολή:

```
sudo /etc/init.d/flowvisor stop
```

Υπάρχει και η δυνατότητα να τον επανεκκινήσουμε με την εντολή:

```
sudo /etc/init.d/flowvisor restart
```

5. Επίδειξη Network Slicing

Προκειμένου να πραγματοποιηθεί η επίδειξη του Network Slicing σε μία εικονική τοπολογία που θα δημιουργήσουμε, θα χρησιμοποιήσουμε το Mininet, MiniEdit, τον controller Flowvisor, τον server XLaunch και το Putty για τις συνδέσεις SSH. Η εγκατάσταση των παραπάνω εργαλείων, έχει παρουσιαστεί αναλυτικά στα προηγούμενα κεφάλαια.

Αρχικά εκκινούμε την εικονική μηχανή του Mininet στο οποίο έχουμε ήδη εγκαταστήσει το Flowvisor. Εφόσον γίνει η εκκίνηση, για ευκολία έχω συνεχίσει δημιουργώντας σύνδεση μέσω του Putty. Επίσης εκκινούμε τον Server XLaunch. Όπως έχει αναφερθεί και στα προηγούμενα κεφάλαια, ο server αυτός είναι απαραίτητος για να μπορούμε να τρέξουμε παραθυριακές εφαρμογές μέσω της εικονικής μηχανής. Συγκεκριμένα θα τον χρειαστούμε για το MiniEdit.

Εάν έχει εγκατασταθεί σωστά ο server και τον εκκινήσουμε, θα πρέπει να εμφανιστεί η παρακάτω ένδειξη κάτω δεξιά στην οθόνη στην γραμμή εργασιών.



Εικόνα 5.1: Ένδειξη Ενεργοποίησης XLaunch Server

Μέρος 1^ο

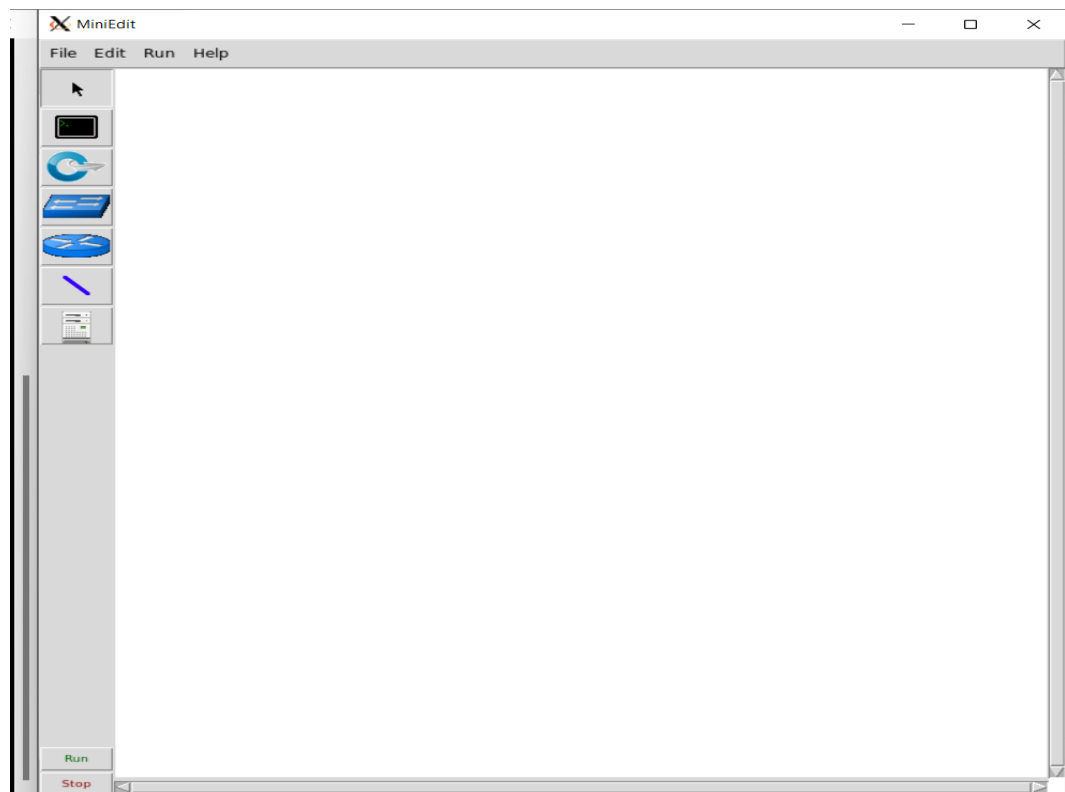
Σε αυτό το παράδειγμα θα δείξουμε πως λειτουργεί το network slicing σε φυσικό επίπεδο. Η τοπολογία που θα δημιουργήσουμε, θα έχει τέσσερις hosts και αποτέλεσμα του network slicing θα είναι η περιορισμένη επικοινωνία μεταξύ αυτών. Αυτό θα επιτευχθεί χωρίζοντας την τοπολογία σε δύο διαφορετικά slices. Έτσι ο host h1 θα επικοινωνεί μόνο με τον host h3 και αντίστοιχα ο host h2 με τον host h4.

Τώρα πρέπει να δημιουργήσουμε την εικονική τοπολογία στην οποία θα πραγματοποιήσουμε τον τεμαχισμό. Η εικονική τοπολογία θα σχεδιαστεί μέσω του εργαλείου MiniEdit.

Για να ανοίξουμε το MiniEdit, στην εικονική μηχανή του Mininet δίνουμε την εντολή:

```
sudo ~/mininet/mininet/examples/miniedit.py
```

Θα πρέπει να εμφανιστεί το περιβάλλον του MiniEdit.

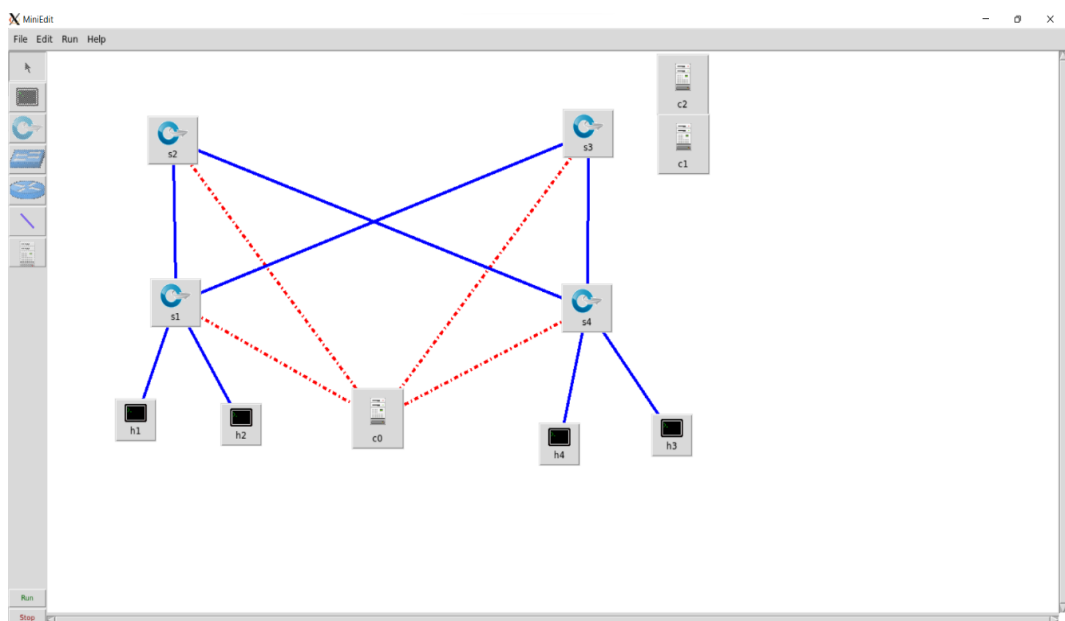


Εικόνα 5.2: Περιβάλλον MiniEdit

Εάν δεν έχουμε εκκινήσει τον server XLaunch, τότε η εντολή δεν θα τρέξει.

Στο περιβάλλον του MiniEdit θα δημιουργήσουμε την εικονική τοπολογία. Αριστερά υπάρχουν οι επιλογές για την εισαγωγή Host, Switch, LegacySwitch, LegacyRouter, NetLink, Controller.

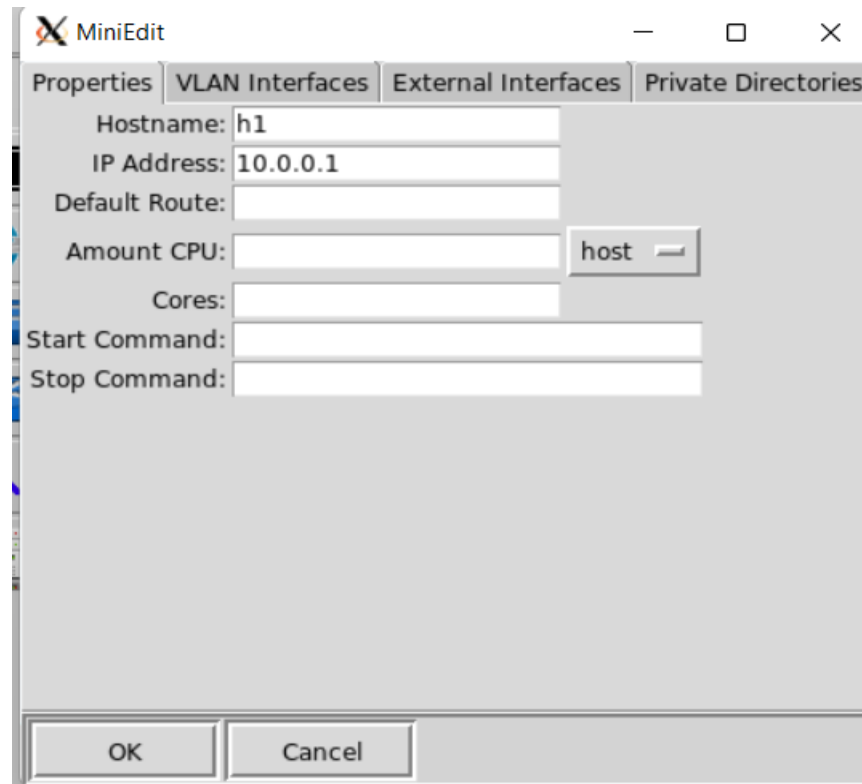
Η τοπολογία που δημιουργούμε περιέχει 4 Hosts, 3 controllers, 4 switches.



Εικόνα 5.3: Τοπολογία στο MiniEdit

Σε κάθε ένα από τα εργαλεία που χρησιμοποιούμε για την τοπολογία θα πρέπει να γίνουν κάποιες ρυθμίσεις.

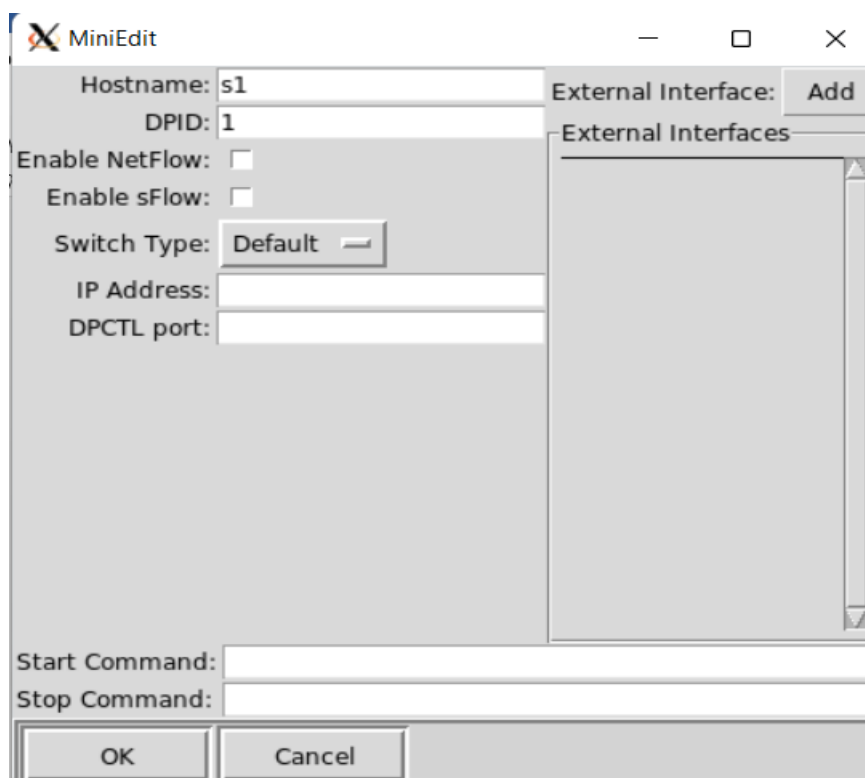
Ξεκινώντας με τα Hosts, πάμε τον κέρσορα πάνω σε κάθε Host, κρατάμε το δεξί κλικ πατημένο, και επιλέγουμε Properties.



Εικόνα 5.4: Ρυθμίσεις h1

Σε κάθε host ορίζουμε ένα HostName και την IP Address του. Στον πρώτο host δίνουμε ως hostname το h1, και ως IP Address 10.0.0.1. Στον δεύτερο host ως hostname το h2, και ως IP Address 10.0.0.2 και παρόμοια στον h3 και h4.

Όσον αφορά τα switches, πάμε τον κέρσορα πάνω σε κάθε switch, και επιλέγουμε το properties.



Εικόνα 5.5: Ρυθμίσεις h2

Για το πρώτο switch για παράδειγμα ορίζουμε ως Hostname το s1 και DPID το 1.

Για το δεύτερο switch ως Hostname το s2 και DPID το 2. Παρόμοια για τα s3 και s4.

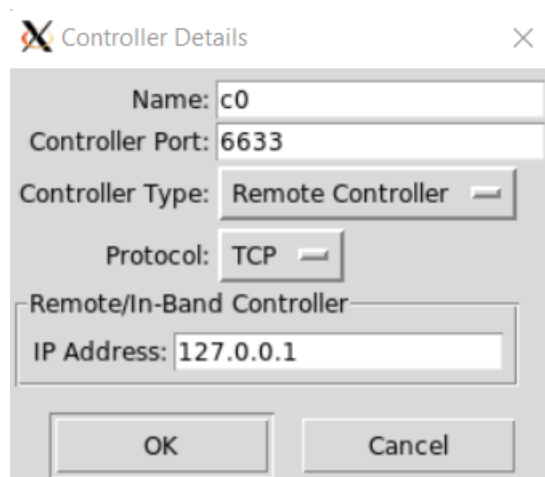
Σε κάθε link μπορούμε να ορίσουμε τα Bandwidth, Delay, Loss, Max Queue size, jitter, και Speedup. Τα link τα οποία μας ενδιαφέρουν και θα ρυθμίσουμε είναι τα: s1-s2, s1-s3, s2-s4, s4-s3.

Συγκεκριμένα, στα link s1-s3 και s3-s4 η μόνη ρύθμιση που θα γίνει, είναι να δώσουμε στο Bandwidth τιμή 10Mbit.

Ομοίως, στα link s1-s2 και s2-s4 θα δώσουμε στο Bandwidth τιμή 1Mbit.

Όσον αφορά τους controllers, για τον controller ο οποίος συνδέεται με τα switches, δίνουμε ως Name το c0, ως controller port την 6633. Στο controller type επιλέγουμε το remote controller. Protocol επιλέγουμε TCP, και IP Address δίνουμε την 127.0.0.1.

Οι παραπάνω ρυθμίσεις έγιναν έτσι, διότι ο συγκεκριμένος controller είναι αυτός που θα κάνει τον τεμαχισμό και στην ουσία, είναι ο flowvisor sdn controller. Για τον λόγο αυτόν, έχει οριστεί και ως Remote Controller.



The screenshot shows a 'Controller Details' dialog box with the following fields and values:

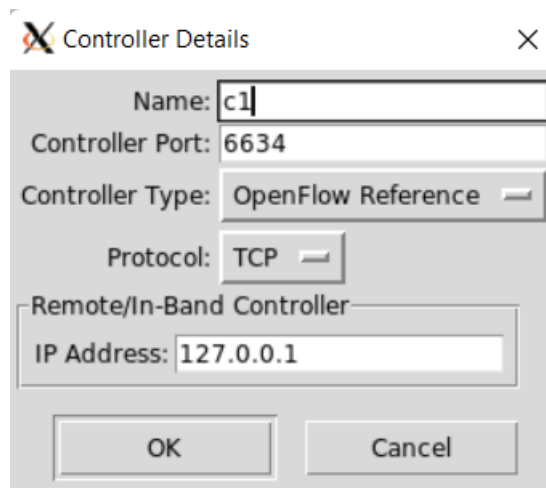
- Name: c0
- Controller Port: 6633
- Controller Type: Remote Controller
- Protocol: TCP
- Remote/In-Band Controller: (unchecked)
- IP Address: 127.0.0.1

At the bottom are 'OK' and 'Cancel' buttons.

Εικόνα 5.6: Ρυθμίσεις c0

Οι άλλοι 2 controllers θα επιβλέπουν τα slice που έχει δημιουργήσει ο προηγούμενος controller. Επομένως δουλειά των 2 αυτών controller είναι η προώθηση των πακέτων.

Οι ρυθμίσεις που έχουν γίνει για αυτούς είναι:

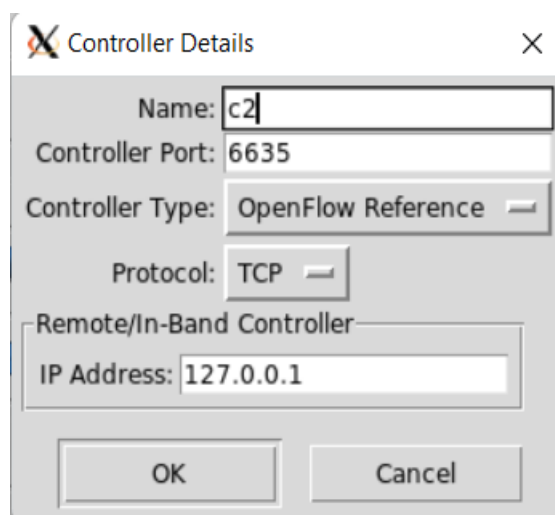


The screenshot shows a 'Controller Details' dialog box with the following fields and values:

- Name: c1
- Controller Port: 6634
- Controller Type: OpenFlow Reference
- Protocol: TCP
- Remote/In-Band Controller: (unchecked)
- IP Address: 127.0.0.1

At the bottom are 'OK' and 'Cancel' buttons.

Εικόνα 5.7: Ρυθμίσεις c1

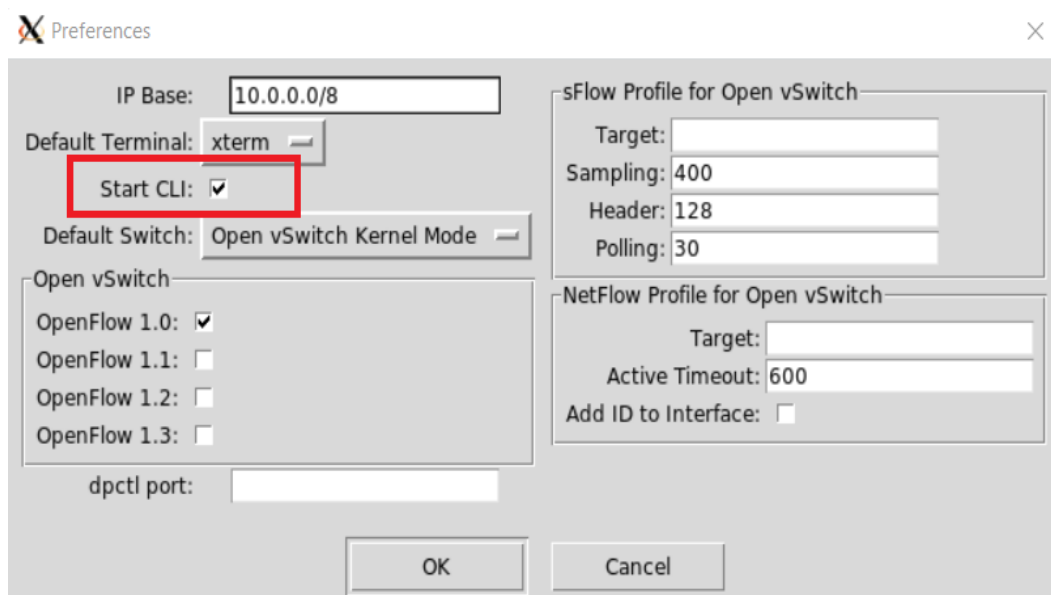


Εικόνα 5.8: Ρυθμίσεις c2

Μπορούμε αν θέλουμε να αποθηκεύσουμε την τοπολογία πηγαίνοντας πάνω αριστερά στο File->Save και να δώσουμε όνομα με κατάληξη είτε .mn ή .py .

Στο συγκεκριμένο παράδειγμα, αυτό που θέλουμε να πετύχουμε είναι να χωρίσουμε την τοπολογία στην μέση, δημιουργώντας 2 slices, και μέσω των 2 αυτών slice να επικοινωνούν μεταξύ τους οι h1-h3 και h2-h4. Επομένως θέλουμε να απομονώσουμε τις 2 αυτές συνδέσεις, και να μην μπορούν να επικοινωνήσουν με κανέναν άλλον host. Για παράδειγμα ο h1 δεν θα μπορεί να επικοινωνήσει με τον h2 ή τον h4, κοκ.

Πριν ξεκινήσουμε να χωρίσουμε την τοπολογία με την βοήθεια του flowvisor, θα πρέπει να την κάνουμε Run. Πάμε πρώτα στο περιβάλλον του MiniEdit, στο Edit->Preferences και επιλέγουμε το Start CLI.



Εικόνα 5.9: Επιλογή Start CLI

Πατάμε OK και στην συνέχεια επιλέγουμε κάτω αριστερά το Run. Στο παράθυρο του Mininet, θα πρέπει να έχει εμφανιστεί κάτι τέτοιο:

```
NOTE: PLEASE REMEMBER TO EXIT THE CLI BEFORE YOU PRESS THE STOP BUTTON. Not exiting will prevent MiniEdit from quitting and will prevent you from starting the network again during this session.

*** Starting CLI:
mininet>
```

Εικόνα 5.10: Mininet Starting CLI

Όπως φαίνεται και στο μήνυμα το οποίο έχει εμφανιστεί, πριν σταματήσουμε την τοπολογία, θα πρέπει πρώτα να δώσουμε την εντολή **exit** στο Mininet.

Στο σημείο αυτό θα πρέπει να τεμαχίσουμε το δίκτυο μέσω του flowvisor controller. Ξεκινάμε μία νέα σύνδεση mininet, καθώς στην πρώτη που έχουμε, βρισκόμαστε στο CLI της τοπολογίας. Αρχικά εκκινούμε το flowvisor. Αυτό θα γίνει δίνοντας την εντολή:

```
sudo /etc/init.d/flowvisor start
```

Το flowvisor τώρα έχει ξεκινήσει. Μπορούμε να δούμε την παραμετροποίηση του, δίνοντας την εντολή:

```
fvctl get-config
```

Ή την εντολή:

```
fvctl -f /dev/null get-config
```

εάν δεν θέλουμε συνέχεια να μας ζητάει το password.

Για να προχωρήσουμε, θα πρέπει να ενεργοποιήσουμε την τοπολογία του flowvisor. Μπορούμε να δούμε εάν είναι ήδη ενεργοποιημένη ελέγχοντας στα αποτελέσματα της παραπάνω εντολής, εάν το πεδίο enable-topo-ctrl έχει την τιμή true. Εάν δεν την έχει τότε την ορίζουμε εμείς δίνοντας την εντολή:

```
fvctl -f /dev/null set-config --enable-topo-ctrl
```

και κάνοντας επανεκκίνηση το flowvisor:

```
sudo /etc/init.d/flowvisor restart
```

```
mininet@mininet-vm:~$ fvctl get-config
Password:
{
  "api_jetty_webserver_port": 8081,
  "api_webserver_port": 8080,
  "checkpointing": false,
  "config_name": "default",
  "db_version": "2",
  "enable-topo-ctrl": true,
```

Εικόνα 5.11: Ένδειξη Ενεργοποίησης τοπολογίας flowvisor

Τώρα θα πρέπει να χωρίσουμε το δίκτυο μας σε 2 slice, όπως έχει αναλυθεί παραπάνω. Ο χωρισμός των slice θα γίνει βάση των πορτών των switch, η οποίες

έχουν πάρει τις τιμές τους ανάλογα με το πώς έχουν γίνει οι συνδέσεις. Προκειμένου να δούμε τις πόρτες κάθε switch, αρκεί να δώσουμε την εντολή:

```
links
```

στο παράθυρο του mininet στο οποίο τρέχει το CLI της τοπολογίας.

Το αποτέλεσμα θα πρέπει να είναι κάτι τέτοιο:

```
mininet> links
s1-eth1<->s2-eth1 (OK OK)
s1-eth2<->s3-eth1 (OK OK)
s2-eth2<->s4-eth1 (OK OK)
s3-eth2<->s4-eth2 (OK OK)
s1-eth3<->h1-eth0 (OK OK)
s1-eth4<->h2-eth0 (OK OK)
s4-eth3<->h3-eth0 (OK OK)
s4-eth4<->h4-eth0 (OK OK)
mininet> █
```

Εικόνα 5.12: Αποτέλεσμα links εντολής

Εάν υπάρχουν ήδη κάποια slice, μπορούμε να το ελέγξουμε με την εντολή:

```
fvctl -f /dev/null list-slices
```

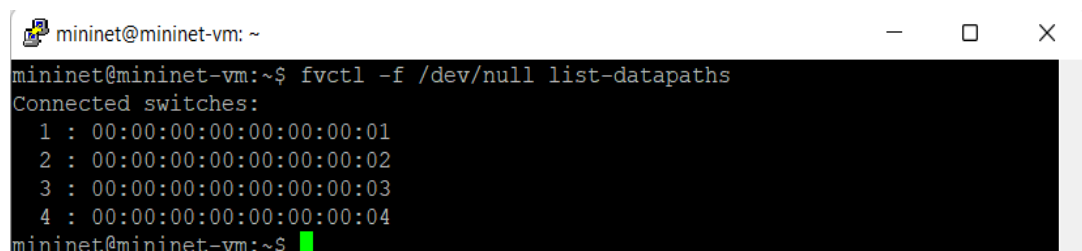
Μπορούμε επίσης να ελέγξουμε εάν υπάρχουν ήδη flowspace με την εντολή:

```
fvctl -f /dev/null list-flowspace
```

Πρέπει να βεβαιωθούμε ότι όλα τα switches έχουν συνδεθεί και τρέχουν. Αυτό θα το δούμε με την εντολή:

```
fvctl -f /dev/null list-datapaths
```

Το αποτέλεσμα που θα πρέπει να πάρουμε θα είναι αυτό:



```
mininet@mininet-vm: ~
mininet@mininet-vm:~$ fvctl -f /dev/null list-datapaths
Connected switches:
 1 : 00:00:00:00:00:00:00:01
 2 : 00:00:00:00:00:00:00:02
 3 : 00:00:00:00:00:00:00:03
 4 : 00:00:00:00:00:00:00:04
mininet@mininet-vm:~$ █
```

Εικόνα 5.13: Αποτέλεσμα εντολής `fvctl -f /dev/null list-datapaths`

Ακόμη πρέπει να βεβαιωθούμε ότι όλα τα Links είναι ενεργά και τρέχουν. Αυτό θα γίνει με την εντολή:

```
fvctl -f /dev/null list-links
```

Και το αποτέλεσμα θα πρέπει να είναι αυτό:

```

mininet@mininet-vm: ~
mininet@mininet-vm:~$ fvctl -f /dev/null list-links
[
  {
    "dstDPID": "00:00:00:00:00:00:00:01",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:02",
    "srcPort": "1"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:02",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:01",
    "srcPort": "1"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:04",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:02",
    "srcPort": "2"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:03",
    "dstPort": "2",
    "srcDPID": "00:00:00:00:00:00:00:04",
    "srcPort": "2"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:01",
    "dstPort": "2",
    "srcDPID": "00:00:00:00:00:00:00:03",
    "srcPort": "1"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:04",
    "dstPort": "2",
    "srcDPID": "00:00:00:00:00:00:00:03",
    "srcPort": "2"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:02",
    "dstPort": "2",
    "srcDPID": "00:00:00:00:00:00:00:04",
    "srcPort": "1"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:03",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:01",
    "srcPort": "2"
  }
]

```

Εικόνα 5.14: Αποτέλεσμα εντολής *fvctl -f /dev/null list-links*

Αυτό που θα κάνουμε τώρα, είναι να ορίσουμε τα 2 slice της τοπολογίας. Η δημιουργία ενός slice γίνεται με την εντολή:

```
fvctl add-slice <slicename><controller-url><admin-mail>
```

όπου *slicename* είναι το όνομα που θα δώσουμε στο slice, *controller-url* είναι διεύθυνση του controller που θα έχει το slice και έχει την μορφή *tcp:localhost:port*, και *admin-mail* το email που θα χρησιμοποιηθεί σε περίπτωση προβλήματος. Όπως αναφέρθηκε και στην περιγραφή του demo, το ένα slice θα ανατεθεί στον controller c1, και το άλλο στον c2. Για να δημιουργήσουμε επομένως το πρώτο slice, δίνουμε την εντολή:

```
fvctl -f /dev/null add-slice upperslice tcp:localhost:6634 abc@upperslice
```

Για το δεύτερο slice δίνουμε την εντολή:

```
fvctl -f /dev/null add-slice loweslice tcp:localhost:6635 bcd@lowerslice
```

Ελέγχουμε εάν η δημιουργία έγινε σωστά.

```
fvctl -f /dev/null list-slices
```

Το αποτέλεσμα θα πρέπει να είναι αυτό:

```
mininet@mininet-vm:~$ fvctl list-slices
Password:
Configured slices:
fvadmin      --> enabled
upperslice   --> enabled
lowerslice   --> enabled
```

Εικόνα 5.15: Αποτέλεσμα εντολής *fvctl list-slices*

Μετά την δημιουργία των slices, θα πρέπει να δημιουργήσουμε και τα flowspaces. Αυτό θα γίνει με την εντολή:

```
fvctl add-flowspace <flowspace-name><dpid><priority><match><slice-perm>
```

κατά τα οποία, flowspace-name θα είναι το όνομα που θα δώσουμε στο flowspace. dpid αφορά το switch στο οποίο θέλουμε να αναφερθούμε -το dpid το είχαμε ορίσει κατά την δημιουργία της τοπολογίας για κάθε switch-. Priority αφορά την προτεραιότητα που καθορίζεται όταν ένα πακέτο απασχολεί περισσότερα από ένα flowspace. Το match αφορά flow ή flows, τα οποία περιλαμβάνουν εκχωρήσεις μορφής πεδίου=τιμή τα οποία διαχωρίζονται με κόμμα. Το slice-perm είναι μία λίστα από slices διαχωρισμένα από κόμματα η οποία έχει τον έλεγχο για ένα συγκεκριμένο flowspace, με την μορφή slice_name=value. Το value που θα μπορεί να πάρει κάθε φορά, είναι DELEGATE, READ και WRITE, και προσδιορίζονται με τις τιμές: DELEGATE=1, READ=2, WRITE=4. Η δημιουργία των flowspaces είναι απαραίτητη, καθώς μέσω αυτών αντιστοιχίζονται οι κινήσεις στις πόρτες τως switch.

Στο παράθυρο του mininet, μπορούμε να δούμε κι άλλες πληροφορίες για την δημιουργία των flowspaces, δίνοντας την εντολή:

```
fvctl add-flowspace -h
```

Ξεκινάμε δημιουργώντας το πρώτο flowspace, το οποίο αφορά το switch s1 και την πόρτα 1. Το switch s1, ανήκει μισό στο upperslice, και μισό στο lowerslice. Το flowspace που θα δημιουργηθεί τώρα, αφορά το μέρος του switch που ανήκει στο upperslice. Ως flowspace_name θα δώσουμε το όνομα dpid1port1, για να είναι εύκολα αντιληπτό ότι αναφερόμαστε στο switch1 (s1) και στην πόρτα 1.

```
fvctl -f /dev/null add-flowspace dpid1port1 1 1 in_port=1 upperslice=7
```

Συνεχίζουμε δημιουργώντας το flow-space που αφορά το switch s1 στην σύνδεσή του με τον host h1. Η πόρτα που θα αναφερθούμε είναι η 3, και το slice το οποίο μας ενδιαφέρει είναι το upperslice.

```
fvctl -f /dev/null add-flow-space dpid1port3 1 1 in_port=3 upperslice=7
```

Το switch s2, ανήκει στο upperslice εξ ολοκλήρου. Όταν ένα switch ανήκει μόνο σε ένα slice, στην θέση των κριτηρίων μπαίνει η λέξη any.

```
fvctl -f /dev/null add-flow-space dpid2 2 1 any upperslice=7
```

Πηγαίνοντας τώρα στο switch s4, ομοίως με το switch s1, το μισό ανήκει στο upperslice και το άλλο μισό στο lowerslice. Ξεκινώντας για την πόρτα 1 η οποία ανήκει στο upperslice δημιουργούμε το flow-space.

```
fvctl -f /dev/null add-flow-space dpid4port1 4 1 in_port=1 upperslice=7
```

Και για την πόρτα 3 η οποία ανήκει στο upperslice επίσης:

```
fvctl -f /dev/null add-flow-space dpid4port3 4 1 in_port=3 upperslice=7
```

Τώρα, θα δημιουργήσουμε τα flow-space που αφορούν το lowerslice. Συγκεκριμένα ξεκινάμε με την δημιουργία του flow-space που αφορά το switch s1 και την πόρτα 4.

```
fvctl -f /dev/null add-flow-space dpid1port4 1 1 in_port=4 lowerslice=7
```

Στην συνέχεια για την πόρτα 2.

```
fvctl -f /dev/null add-flow-space dpid1port2 1 1 in_port=2 lowerslice=7
```

Το switch s3, ομοίως με το switch s2, ανήκει ολόκληρο στο lowerslice. Επομένως:

```
fvctl -f /dev/null add-flow-space dpid3 3 1 any lowerslice=7
```

Τελευταίο που έχει μείνει, είναι το switch s4 και συγκεκριμένα οι πόρτες 2 και 4 η οποίες ανήκουν στο lowerslice.

```
fvctl -f /dev/null add-flow-space dpid4port2 4 1 in_port=2 lowerslice=7
```

```
fvctl -f /dev/null add-flow-space dpid4port4 4 1 in_port=4 lowerslice=7
```

Εφόσον τα flow-space έχουν δημιουργηθεί σωστά, μπορούμε να τα ελέγξουμε με την παρακάτω εντολή:

```
fvctl -f /dev/null list-flow-space
```

Και το αποτέλεσμα που θα πάρουμε είναι αυτό:

```
mininet@mininet-vm:~$ fvctl -f /dev/null list-flowspace
Configured Flow entries:
{"force-enqueue": -1, "name": "dpid1port3", "slice-action": [{"slice-name": "upper", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01", "id": 70, "match": {"wildcards": 4194302, "in_port": 3}}
{"force-enqueue": -1, "name": "dpid1port3", "slice-action": [{"slice-name": "upper", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01", "id": 71, "match": {"wildcards": 4194302, "in_port": 3}}
{"force-enqueue": -1, "name": "dpid1port1", "slice-action": [{"slice-name": "upper", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01", "id": 72, "match": {"wildcards": 4194302, "in_port": 1}}
{"force-enqueue": -1, "name": "dpid2", "slice-action": [{"slice-name": "upper", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:02", "id": 73, "match": {"wildcards": 4194303}}
{"force-enqueue": -1, "name": "dpid4port1", "slice-action": [{"slice-name": "upper", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:04", "id": 74, "match": {"wildcards": 4194302, "in_port": 1}}
{"force-enqueue": -1, "name": "dpid4port3", "slice-action": [{"slice-name": "upper", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:04", "id": 75, "match": {"wildcards": 4194302, "in_port": 3}}
{"force-enqueue": -1, "name": "dpid1port4", "slice-action": [{"slice-name": "lower", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01", "id": 76, "match": {"wildcards": 4194302, "in_port": 4}}
{"force-enqueue": -1, "name": "dpid1port2", "slice-action": [{"slice-name": "lower", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01", "id": 77, "match": {"wildcards": 4194302, "in_port": 2}}
{"force-enqueue": -1, "name": "dpid3", "slice-action": [{"slice-name": "lower", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:03", "id": 78, "match": {"wildcards": 4194303}}
{"force-enqueue": -1, "name": "dpid4port2", "slice-action": [{"slice-name": "lower", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:04", "id": 79, "match": {"wildcards": 4194302, "in_port": 2}}
{"force-enqueue": -1, "name": "dpid4port4", "slice-action": [{"slice-name": "lower", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:04", "id": 80, "match": {"wildcards": 4194302, "in_port": 4}}
mininet@mininet-vm:~$
```

Εικόνα 5.16: Αποτέλεσμα εντολής *fvctl -f /dev/null list-flowspace*

Προκειμένου να ελέγξουμε εάν όλη η παραμετροποίηση του δικτύου μας έχει γίνει με επιτυχία, θα πρέπει να δούμε τον τρόπο με τον οποίο επικοινωνούν μεταξύ τους οι hosts. Αυτό θα γίνει δίνοντας την εντολή **pingall** στο παράθυρο του mininet, στο CLI. Πριν από αυτό όμως πρέπει να είμαστε βέβαιοι, πως το flowvisor τρέχει. Εάν δεν το έχουμε ξεκινήσει, η απάντηση που θα πάρουμε από την παραπάνω εντολή, θα είναι αυτή:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X X X
h4 -> X X X
h3 -> X X X
*** Results: 100% dropped (0/12 received)
mininet>
```

Εικόνα 5.17: Αποτέλεσμα εντολής *pingall*

Είναι προφανές πως δεν υπάρχει επικοινωνία μεταξύ των host, χωρίς την ενεργοποίηση του flowvisor.

Εφόσον λοιπόν έχουμε ξεκινήσει το flowvisor (με την εντολή **sudo /etc/init.d/flowvisor start**) κάνουμε pingall, και το αποτέλεσμα που παίρνουμε είναι:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X h3
h2 -> X h4 X
h4 -> X h2 X
h3 -> h1 X X
*** Results: 66% dropped (4/12 received)
mininet>
```

Εικόνα 5.18: Αποτέλεσμα εντολής *pingall*

Βλέπουμε λοιπόν, πως υπάρχει επικοινωνία μεταξύ των h1-h3 και h2-h4. Αυτό είναι που θέλαμε να πετύχουμε με τον τεμαχισμό του δικτύου μας. Επομένως η παραμετροποίηση -ο τεμαχισμός- του δικτύου έχει γίνει σωστά.

Μέρος 2ο

Στο προηγούμενο παράδειγμα, δείξαμε πως μπορεί να επιτευχθεί το network slicing σε φυσικό επίπεδο. Όμως αυτό δεν είναι ένα ρεαλιστικό παράδειγμα, και δεν αποδεικνύεται η ουσία του network slicing. Βάση της παραπάνω τοπολογίας, θα δημιουργήσουμε ένα νέο παράδειγμα.

Αρχικά, θα πρέπει να διαγράψουμε τα slices που δημιουργήσαμε στο πρώτο παράδειγμα. Αυτό θα γίνει με την εντολή:

```
fvctl -f /dev/null remove-slice upperslice
fvctl -f /dev/null remove-slice lowerslice
```

Στο παράδειγμα αυτό θα δημιουργήσουμε 2 μονοπάτια. Ένα με υψηλό bandwidth και ένα με χαμηλό. Για να γίνει το παράδειγμα πιο ρεαλιστικό, θεωρούμε πως έχουμε κίνηση για την αποστολή video, και κίνηση που είναι για non-video. Όπως είναι λογικό, θέλουμε κάθε κίνηση που αφορά το video, να την κάνουμε να περνάει από το μονοπάτι με το υψηλό bandwidth. Με τον τρόπο αυτό, υποθέτοντας πως έχουμε συνδέσεις Skype για παράδειγμα, αυτές δεν θα επηρεάζουν το μονοπάτι στο οποίο θα αποστέλλεται το video.

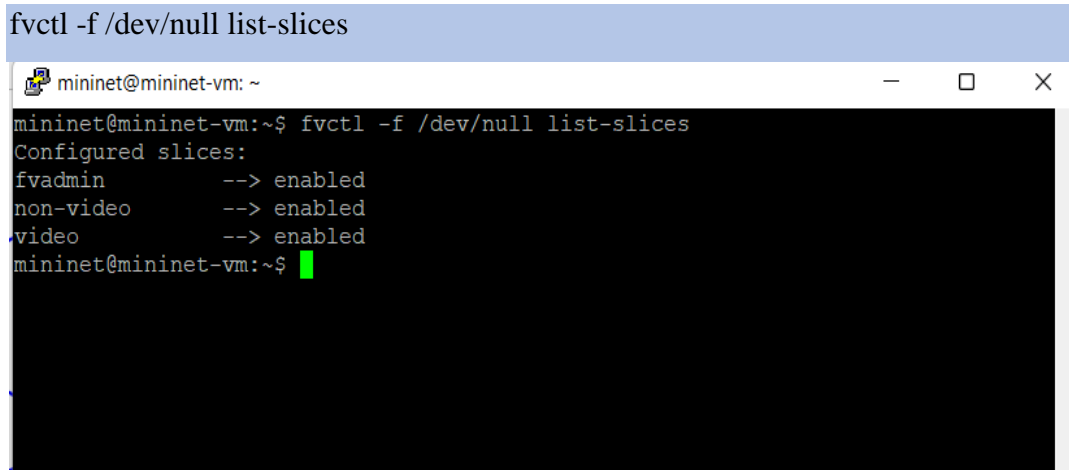
Προκειμένου να επιτευχθεί το παραπάνω, θα δημιουργήσουμε 2 slice, τα οποία θα τα ονομάσουμε video και non-video. Στον controller c1 με port 6634 θα ανατεθεί η κίνηση του non-video, ενώ στον c2 με port 6635 θα ανατεθεί η κίνηση του video. Η κίνηση θα ανατίθεται βάση των προτεραιοτήτων που θα ορίσουμε. Συγκεκριμένα τα

Η δημιουργία των slice, θα γίνει ως εξής:

```
fvctl -f /dev/null add-slice non-video tcp:localhost:6634 admin@non-video
fvctl -f /dev/null add-slice video tcp:localhost:6635 admin@video
```

Βλέπουμε πως τα slice δημιουργήθηκαν με την εντολή:

```
fvctl -f /dev/null list-slices
```



```
mininet@mininet-vm:~$ fvctl -f /dev/null list-slices
Configured slices:
fvadmin      --> enabled
non-video    --> enabled
video        --> enabled
mininet@mininet-vm:~$
```

Εικόνα 5.19: Αποτέλεσμα εντολής *fvctl -f /dev/nul list-slices*

Επόμενο βήμα που πρέπει να γίνει, είναι η δημιουργία των flowspace. Ξεκινώντας με το πρώτο switch (s1), πρέπει να αντιστοιχήσουμε τις κινήσεις για κάθε πόρτα. Η πόρτα 4, ανήκει και στα 2 slice. Πρέπει επομένως να την παραμετροποιήσουμε ανάλογα με τον προορισμό. Εάν δηλαδή από την πόρτα 4, φεύγουν δεδομένα προς τον h1, ή έρχονται δεδομένα από αυτόν, αυτομάτως αναφερόμαστε σε video traffic. Επομένως θα πρέπει η κίνηση να ανατίθεται στο slice που αφορά το video. Αυτό γίνεται με τις εντολές:

```
fvctl -f /dev/null add-flowspace dpid1port4video-src 1 10
in_port=4,nw_src=10.0.0.1 video=7

fvctl -f /dev/null add-flowspace dpid1port4video-dst 1 10
in_port=4,nw_dst=10.0.0.1 video=7

fvctl -f /dev/null add-flowspace dpid1port4non-video 1 1 in_port=4 non-video=7
```

Η πρώτη εντολή, αφορά την περίπτωση που ο h1 είναι source και η πόρτα 4 του s1 είναι destination. Η δεύτερη εντολή αφορά την αντίθετη περίπτωση. Η τρίτη εντολή αφορά όλα τα υπόλοιπα πακέτα για την πόρτα 4 του s1, τα οποία ανατίθενται στο slice non-video.

Τώρα πρέπει να ορίσουμε την κίνηση για τις υπόλοιπες πόρτες του s1. Οι εντολές που χρειάζονται είναι:

```
fvctl -f /dev/null add-flowspace dpid1port3video 1 1 in_port=3 video=7

fvctl -f /dev/null add-flowspace dpid1port1non-video 1 1 in_port=1 non-video=7

fvctl -f /dev/null add-flowspace dpid1port2video 1 1 in_port=2 video=7
```

Η πόρτα 1 ανήκει στο non-video καθώς συνδέεται με το s2. Η πόρτα 2 και η πόρτα 3 ανήκουν στο video, καθώς ανήκουν στον h1 και στο s3 αντίστοιχα.

Το switch s2 ανήκει ολόκληρο στο non-video slice:

```
fvctl -f /dev/null add-flowspace dpid2non-video 2 1 any non-video=7
```

Ενώ το s3 ανήκει ολόκληρο στο video slice:

```
fvctl -f /dev/null add-flowspace dpid3video 3 1 any video=7
```

Στην συνέχεια παραμετροποιούμε τις πόρτες του switch 4 (s4). Η πόρτα 3, αφορά και το video και το non-video. Επομένως:

```
fvctl -f /dev/null add-flowspace dpid4port3non-video 4 1 in_port=3 non-video=7
fvctl -f /dev/null add-flowspace dpid4port3video-src 4 10 in_port=3,nw_src=10.0.0.1 video=7
fvctl -f /dev/null add-flowspace dpid4port3video-dst 4 10 in_port=3,nw_dst=10.0.0.1 video=7
```

Παρόμοια με την πόρτα 3, έτσι και η 4 ανήκει και στο video και στο non-video. Άρα:

```
fvctl -f /dev/null add-flowspace dpid4port3video-src 4 10 in_port=3,nw_src=10.0.0.1 video=7
fvctl -f /dev/null add-flowspace dpid4port3video-dst 4 10 in_port=3,nw_dst=10.0.0.1 video=7
fvctl -f /dev/null add-flowspace dpid4port3non-video 4 1 in_port=3 non-video=7
```

Σειρά έχουν οι πόρτες 1 και 2. Όσον αφορά την πόρτα 1, η οποία συνδέεται με το switch s2, ανήκει στο non-video. Η πόρτα 2 η οποία συνδέεται με το switch s3 ανήκει στο video. Επομένως:

```
fvctl -f /dev/null add-flowspace dpid4port1non-video 4 1 in_port=1 non-video=7
fvctl -f /dev/null add-flowspace dpid4port2video 4 1 in_port=2 video=7
```

Ελέγχουμε τα flowspaces που δημιουργήσαμε με την εντολή:

```
fvctl -f /dev/null list-flowspace
```

Εάν όλα τα παραπάνω έχουν γίνει σωστά, κάνουμε ένα reboot την τοπολογία μας και στην συνέχεια, εφόσον την ξεκινήσουμε ξανά, και δίνοντας την εντολή **pingall** στο CLI, θα πρέπει να δούμε πως όλοι οι hosts επικοινωνούν μεταξύ τους.


```

*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h4 h3
h2 -> h1 h4 h3
h4 -> h1 h2 h3
h3 -> h1 h2 h4
*** Results: 0% dropped (12/12 received)
mininet>

```

Εικόνα 5.20: Αποτέλεσμα εντολής *pingall*

Για να αποδείξουμε την ορθή λειτουργία των slice που δημιουργήσαμε, θα πρέπει να δείξουμε πως μία κίνηση που αφορά το video, θα πηγαίνει από το slice video και θα έχει bandwidth γύρω στα 10 Mb/s. Αντίστοιχα μια κίνηση που δεν αφορά την κίνηση ενός video, θα πηγαίνει από το non-video slice και θα έχει bandwidth γύρω στο 1Mb/s. Αυτό θα το ελέγξουμε, με την βοήθεια της εντολής *iperf*, η οποία στέλνει πακέτα από έναν host στον άλλον, και ελέγχει το bandwidth. Επομένως στέλνοντας ένα πακέτο για παράδειγμα από τον h4 στον h1 (ο οποίος αφορά μόνο τα video), θα πρέπει να δούμε πως η μεταφορά γίνεται με bandwidth γύρω στα 10Mb/s.

Να υπενθυμίσουμε στο σημείο αυτό, πως οι hosts έχουν επικοινωνία όλοι μεταξύ τους.

Τώρα, θα δείξουμε πως η κίνηση video και non-video, οδηγούνται στα σωστά μονοπάτια.

Από το σημείο αυτό και έπειτα, εννοείτε πως η τοπολογία μας είναι σε λειτουργία (RUN) .

Ανοίγουμε το terminal για τους host h1 και h4 από το CLI:

```
xterm h1 h4
```

Στο τερματικό του h1 δίνουμε την εντολή:

```
iperf -s -p
```

Στο τερματικό του h4, δίνουμε την εντολή:

```
iperf -c 10.0.0.1 -p 5001 -t 10 -i 1
```

μέσω της οποίας στέλνουμε πακέτα στον h1.

Αποτελέσματα σε κάθε τερματικό θα πρέπει να είναι τα παρακάτω:

Για τον h1:

```

"Node: h1"@mininet-vm
root@mininet-vm:~# iperf -s -p
iperf: option requires an argument -- p
-----
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 28] local 10.0.0.1 port 5001 connected with 10.0.0.3 port 48628
[ ID] Interval      Transfer    Bandwidth
[ 28] 0.0-12.7 sec   14.5 MBytes  9.55 Mbits/sec

```

Εικόνα 5.21: Εκκίνηση server από το Τερματικό του h1

Για τον h3:

```

root@mininet-vm:~# iperf -c 10.0.0.1 -p 5001 -t 10 -i 1
-----
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 6] local 10.0.0.4 port 58532 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 6] 0.0- 1.0 sec   1.38 MBytes  11.5 Mbits/sec
[ 6] 1.0- 2.0 sec   1.25 MBytes  10.5 Mbits/sec
[ 6] 2.0- 3.0 sec   1.12 MBytes  9.44 Mbits/sec
[ 6] 3.0- 4.0 sec   1.12 MBytes  9.44 Mbits/sec
[ 6] 4.0- 5.0 sec   1.25 MBytes  10.5 Mbits/sec
[ 6] 5.0- 6.0 sec   1.38 MBytes  11.5 Mbits/sec
[ 6] 6.0- 7.0 sec   1.50 MBytes  12.6 Mbits/sec
[ 6] 7.0- 8.0 sec   1.62 MBytes  13.6 Mbits/sec
[ 6] 8.0- 9.0 sec   1.38 MBytes  11.5 Mbits/sec
[ 6] 9.0-10.0 sec   1.88 MBytes  15.7 Mbits/sec
[ 6] 0.0-10.4 sec   14.0 MBytes  11.3 Mbits/sec
root@mininet-vm:~#

```

Εικόνα 5.22: Αποτέλεσμα iperf από τον h3 στον h1

Βλέπουμε πως το bandwidth που έχουμε σε αυτήν την περίπτωση δεν αποκλίνει πολύ από τα 10Mb/s. Επομένως μία κίνηση που αφορά την αποστολή video, ορθά πηγαίνει από το slice που αφορά τα video.

Ομοίως μπορούμε να ελέγξουμε το bandwidth για μία κίνηση που δεν αφορά κάποιο video. Όπως παραπάνω, έτσι και τώρα θα ανοίξουμε τα terminal των h2 και h4.

Στο τερματικό του h2, δίνουμε την εντολή:

```
iperf -s -p
```

Ενώ από τον h4 θα στείλουμε πακέτα στον h2, επομένως θα δώσουμε την εντολή:

```
iperf -c 10.0.0.2 -p 5001 -t 10 -i 1
```

Τα αποτελέσματα από κάθε τερματικό είναι τα παρακάτω:

Για τον h2:

```

root@mininet-vm:~# iperf -s -p
iperf: option requires an argument -- p
-----
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 28] local 10.0.0.2 port 5001 connected with 10.0.0.4 port 34678
[ ID] Interval      Transfer    Bandwidth
[ 28] 0.0-25.2 sec  2.88 MBytes  957 Kbits/sec

```

Εικόνα 5.14: Εκκίνηση server από το Τερματικό του h2

Για τον h4:

```

^Croot@mininet-vm:~# iperf -c 10.0.0.2 -p 5001 -t 10 -i 1
-----
Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 6] local 10.0.0.4 port 34678 connected with 10.0.0.2 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 6] 0.0- 1.0 sec   384 KBytes  3.15 Mbits/sec
[ 6] 1.0- 2.0 sec   256 KBytes  2.10 Mbits/sec
[ 6] 2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[ 6] 3.0- 4.0 sec   256 KBytes  2.10 Mbits/sec
[ 6] 4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[ 6] 5.0- 6.0 sec   256 KBytes  2.10 Mbits/sec
[ 6] 6.0- 7.0 sec   256 KBytes  2.10 Mbits/sec
[ 6] 7.0- 8.0 sec   512 KBytes  4.19 Mbits/sec
[ 6] 8.0- 9.0 sec    0.00 Bytes  0.00 bits/sec
[ 6] 9.0-10.0 sec   640 KBytes  5.24 Mbits/sec
[ 6] 10.0-11.0 sec  0.00 Bytes  0.00 bits/sec
[ 6] 0.0-11.2 sec  2.88 MBytes  2.16 Mbits/sec
root@mininet-vm:~#

```

Εικόνα 5.15: Αποτέλεσμα iperf από τον h4 στον h2

Βλέπουμε λοιπόν, πως στην περίπτωση κίνησης που δεν αφορά video, έχουμε bandwidth το οποίο δεν αποκλίνει πολύ από το 1Mb/s, και άρα ορθά πάει από το slice non-video.

6. Συμπεράσματα

Με την ανάπτυξη της τεχνολογίας κινητών επικοινωνιών, το παραδοσιακό ενιαίο μοντέλο δικτύου δεν μπόρεσε να ανταποκριθεί στον βαθμό που έπρεπε για τις ανάγκες των χρηστών, και η ζήτηση για διαφοροποιημένες υπηρεσίες όλο ένα και αυξανόταν. Προκειμένου να λυθεί αυτό το πρόβλημα, δημιουργήθηκε η πέμπτη γενιά τεχνολογίας κινητών επικοινωνιών 5G, και η τεχνολογία του network slicing.

Το Network Slicing είναι μία από τις τεχνολογίες με την μεγαλύτερη επιρροή στον τομέα του 5G και έχει αλλάξει την βιομηχανία των τηλεπικοινωνιών. Το 5G απαιτεί την υποδοχή ταχέως αυξανόμενων συσκευών και χρηστών στο δίκτυο, και ο τρόπος προκειμένου να επιτευχθεί αυτό στο έπακρον, είναι ο μηχανισμός του τεμαχισμού δικτύου. Για να καταστεί δυνατός ο τεμαχισμός, απαιτείται η χρήση τεχνικών SDN και NFV. Τα SDN και NFV έχουν γίνει το κύριο τμήμα του δικτύου τεχνικής υποστήριξης, και αυτό οφείλεται στην ζήτηση κυρίως που υπάρχει στην επιχειρηματική σκηνή, για προσαρμοσμένη κοπή δικτύου ανάλογα με τις ανάγκες που υπάρχουν, καθώς επίσης και στην βελτιστοποίηση που παρέχουν στις επιχειρηματικές διαδικασίες και στην δρομολόγηση δεδομένων. Στην εργασία αυτή, έγινε μία επίδειξη τεμαχισμού δικτύου σε εικονικό δίκτυο, με την χρήση του SDN controller FlowVisor.

Τα slices τα οποία δημιουργούνται με τον μηχανισμό του τεμαχισμού δικτύου, μπορούν να καλύπτουν πολλούς τομείς δικτύου, συμπεριλαμβανομένης της πρόσβασης, του πυρήνα και των μεταφορών, και να αναπτύσσονται σε πολλούς φορείς εκμετάλλευσης. Το network slicing, μπορεί να υποστηρίξει υπηρεσίες με τελείως διαφορετικές απαιτήσεις, από μία απλή φωνητική κλήση έως και ένα συνδεδεμένο όχημα, οι οποίες απαιτούν και διαφορετική απόδοση, αξιοπιστία και latency. Αποτέλεσμα του τεμαχισμού δικτύου, είναι η δημιουργία slices, τα οποία είναι «ξένα» μεταξύ τους, και αφορούν διαφορετικές υπηρεσίες ενώ βρισκόμαστε στο ίδιο δίκτυο. Με τον τρόπο αυτό, μία φωνητική κλήση, δεν θα μπορεί να επηρεάσει ένα συνδεδεμένο όχημα, ή μία αποστολή βίντεο, ή μία βιντεοκλήση, τα οποία όμως βρίσκονται πάνω στο ίδιο δίκτυο. Κάθε slice έχει την δική του αρχιτεκτονική, διαχείριση και ασφάλεια για την υποστήριξη μιας συγκεκριμένης περίπτωσης χρήσης. Επομένως, δυνατότητες όπως η ταχύτητα δεδομένων, η χωρητικότητα, η συνδεσιμότητα, η ποιότητα, ο λανθάνων χρόνος, η αξιοπιστία και οι υπηρεσίες μπορούν να προσαρμοστούν σε κάθε slice ώστε να συμμορφώνονται σε μία συγκεκριμένη συμφωνία επιπέδου υπηρεσιών.

Εν κατακλείδι, το 5G είναι η επαναστατική καινοτομία στην τεχνολογία κινητών επικοινωνιών η οποία ανταποκρίνεται στις ανάγκες των ανθρώπων για ποιότητα ζωής, και παρέχει πάντα δίκτυα χαμηλής λανθάνουσας κατάστασης, χαμηλής ισχύος και υψηλής ασφάλειας. Ο τεμαχισμός του δικτύου είναι ο μηχανισμός ο οποίος είναι απαραίτητος για να ξεχωρίσει το 5G από τις προηγούμενες γενιές κινητών επικοινωνιών.

Βιβλιογραφία

1. 5G - Architecture. Online Tutorials Library. https://www.tutorialspoint.com/5g/5g_architecture.htm
2. Caroline Chan: Benefits of 5G Technology. Unleashing Innovation: <https://www.intel.com/content/www/us/en/wireless-network/5g-benefits-features.html>
3. Singh, S. 1G, 2G,...& 5G: The evolution of the G's | MS&E 238 Blog. MS&E 238 Blog | Leading Trends In Information Technology. <https://mse238blog.stanford.edu/2017/07/ssound/1g-2g-5g-the-evolution-of-the-gs/>
4. Ciena. Blue Planet - transformational software solutions from Ciena - Blue Planet. <https://www.blueplanet.com/resources/what-is-network-slicing.html#:~:text=Network%20slicing%20is%20a%20method,a%20common%20multi-domain%20infrastructure>
5. 5G network slicing | Benefits of network slicing | Rantcell. Mobile Network Drive Testing and Monitoring Tools | Mobile Network Monitoring Tools | Mobile Network Drive Test Tools | Mobile Network Testing Tools. <https://www.rantcell.com/5g-network-slicing.html#:~:text=Network%20slicing%20considerably%20transforms%20the,network%20architecture%20principles%20and%20capabilities>
6. Xin Li, Mohammed Samaka, H. Antony Chan, Fellow, IEEE, Deval Bhamare, Lav Gupta, Senior member, Chengcheng Guo, and Raj Jain, Fellow
School of Electronic Information, Wuhan University, Wuhan, Hubei 430072, China
Department of Computer Science and Engineering, Qatar University, Doha, Qatar
Huawei Technologies, Plano, TX, USA
Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO 63130, USA Je: https://www.cse.wustl.edu/~jain/papers/ftp/slic_ic.pdf
7. What is Software-Defined Networking (SDN)? . Ciena - A networking systems, services, and software company - Ciena. <https://www.ciena.com/insights/what-is/What-Is-SDN.html>
8. Ciena. Blue Planet - transformational software solutions from Ciena - Blue Planet. <https://www.blueplanet.com/resources/what-is-network-slicing.html#:~:text=Network%20slicing%20is%20a%20method,a%20common%20multi-domain%20infrastructure>
9. Rawal, A. Introduction to OpenFlow Protocol. Engineering Education (EngEd) Program | Section. <https://www.section.io/engineering-education/openflow-sdn/>
10. Platform Overview - OpenDaylight.
OpenDaylight. <https://www.opendaylight.org/about/platform-overview>
11. Molenaar, R. Introduction to SDN OpenDayLight.
NetworkLessons.com. <https://networklessons.com/cisco/ccna-routing-switching-icnd2-200-105/introduction-to-sdn-opensdncore>

12. English, J. (2017, March 23). What is ONOS (Open Network Operating System)? - Definition from WhatIs.com.
SearchNetworking. <https://www.techtarget.com/searchnetworking/definition/ONOS-Open-Network-Operating-System>
13. Pakzad, F. Comparison of Software Defined Networking (SDN) Controllers. Part 2: Open Network Operating System (ONOS) - Aptira. Aptira: Cloud Solutions to Transform your Business. <https://aptira.com/comparison-of-software-defined-networking-sdn-controllers-part-2-open-network-operating-system-onos/>
14. How To Fix the "Warning: Remote Host Identification Has Changed" Error. Kinsta®. <https://kinsta.com/knowledgebase/warning-remote-host-identification-has-changed/>
15. sdn_onos/INSTALL.md at master · jatj/sdn_onos.
GitHub. https://github.com/jatj/sdn_onos/blob/master/INSTALL.md
16. FlowVisor, a network virtualization layer. Wiki
MiNET. <https://wiki.minet.net/sdn/nethypervisor/flowvisor>
17. Kubernetes, C. Install Apache Ant on Ubuntu using the Snap Store | Snapcraft. Snapcraft. <https://snapcraft.io/install/ant/ubuntu>
18. Telenet: Network slicing: realizing the benefits of 5G by tailored use of network capabilities: <https://www.telenor.com/media/public-policy/capturing-the-societal-benefits-of-5g/network-slicing-realising-the-benefits-of-5g-by-tailored-use-of-network-capabilities/#:~:text=Network%20slicing%20is%20a%20mechanism,to%20fulfil%20various%20user%20needs.>
19. Homepage | University of
Southampton. https://www.southampton.ac.uk/~drn1e09/ofertie/openflow_qos_mininet.pdf
20. Hailam. MiniNet -- use iperf tool to test bandwidth. (n.d.). Programming VIP - Very Interesting Programming. <https://programming.vip/docs/mininet-use-iperf-tool-to-test-bandwidth.html>
21. James F. Kurose, Keith W. Ross: Δικτύωση Υπολογιστών Προσέγγιση από Πάνω προς τα Κάτω, ΕΒΔΟΜΗ ΕΚΔΟΣΗ
22. Andrew S. Tanenbaum, David J. Wetherall: Δίκτυα Υπολογιστών, Πέμπτη Αμερικάνικη Έκδοση
23. Abuibaid, M. 5G Network Slicing Using Mininet. Share and Discover Knowledge on SlideShare. https://www.slideshare.net/M_A_Abuibaid/5g-network-slicing-using-mininet
24. nikhilh github: <https://github-wiki-see.page/m/onstutorial/onstutorial/wiki/Flowvisor-Exercise>
25. Zhu, A. What Is OpenFlow Switch and How Does it Work? Fiber Optic Cabling Solutions. <https://www.cables-solutions.com/whats-openflow-switch-how-it-works.html>

26. Cole, B. Network firms form software-defined networking OpenDaylight Project - Embedded.com. Embedded.com. <https://www.embedded.com/network-firms-form-software-defined-networking-opensdaylight-project/>
27. Moriarty-Siler, E. 2016. *What Is an OpenDaylight Controller? AKA: OpenDaylight Platform:* <https://www.sdxcentral.com/networking/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/what-is-sdn-controller/openflow-controller/opensdaylight-controller/>
28. Celona, T. *Network Slicing: What It Is & Why It Matters*. Private 5G LANs for the Enterprise. <https://www.celona.io/5g-lan/network-slicing>
29. Mehrota, S. 5G. Micron Technology, Inc. <https://www.micron.com/solutions/5g>
30. Dabkiewicz, S. OpenFlow network virtualization with FlowVisor - PDF Free Download. Enjoy free comfortable tools to publish, exchange, and share any kind of documents online! <https://docplayer.net/4847775-Openflow-network-virtualization-with-flowvisor.html>
31. Stiernerling, M. Re: [vnrg] FlowVisor : A Network Virtualization Layer. Welcome to CS - CS. <http://eprints.cs.univie.ac.at/2893/1/msg00249.html>
32. KiS, S. (2021, June 11). What is 3G? | Wireless Logic. Wireless Logic. <https://www.wirelesslogic.com/what-is-3g/>
33. Docker Hub. Docker Hub. <https://hub.docker.com/r/onosproject/onos/>
34. ONOS -- Install, and Running as a service - HackMD. HackMD. <https://hackmd.io/@nosignal/rkXUpWMrz?type=view>
35. 5G Network Slicing, What is it? | 5G Slicing Architecture and Solutions. VIAVI Solutions | Network Test, Monitoring, and Assurance. <https://www.viavisolutions.com/en-us/5g-network-slicing>
36. Network slicing: Where are we today and how far is there to go?. STL Partners. <https://stlpartners.com/articles/telco-cloud/network-slicing-where-are-we-today-and-how-far-is-there-to-go/>
37. Network Slicing | Networks Solutions | Samsung Business Global Networks. Samsung global_nw. <https://www.samsung.com/global/business/networks/solutions/network-slicing/>
38. Network Slicing Overview | Paragon Automation 21.2 | Juniper Networks. (n.d.). Juniper Networks – Leader in AI Networking, Cloud, & Connected Security Solutions. <https://www.juniper.net/documentation/us/en/software/paragon-automation21.2/paragon-automation-user-guide/topics/concept/pf-network-slicing-overview.html>

