ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΑ ΔΙΚΤΥΑ ΚΑΙ ΥΠΗΡΕΣΙΕΣ ΤΗΛΕΜΑΤΙΚΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

SDN ΤΕΧΝΟΛΟΓΙΑ, ΤΑ ΔΙΚΤΥΑ ΣΤΗΝ ΝΕΑ ΕΠΟΧΗ

ΗΛΙΑΣ Κ. ΣΧΙΣΜΕΝΟΣ

Επιβλέπων : **Γεώργιος Δημητρακόπουλος,** Επίκουρος Καθηγητής

ΑΘΗΝΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2016

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

SDN τεχνολογία, τα δίκτυα στην νέα εποχή

Ηλίας Κ. Σχισμένος
Α.Μ.: 13210

Επιβλέπων : **Γεώργιος Δημητρακόπουλος,** Επίκουρος Καθηγητής

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ :
Χρήστος Μιχαλακέλης, Επίκουρος Καθηγητής
Θωμάς Καμαλάκης, Επίκουρος Καθηγητής

Σεπτέμβριος 2016

ΠΕΡΙΛΗΨΗ

Η συγκεκριμένη εργασία πραγματοποιήθηκε στα πλαίσια των υποχρεώσεων μου για το Μεταπτυχιακό Πρόγραμμα «Τηλεπικοινωνιακά δίκτυα και υπηρεσίες τηλεματικής» του τμήματος Πληροφορικής και Τηλεματικής του Χαροκόποιειου Πανεπιστημίου Αθηνών.

Σκοπός της εργασίας είναι να παρουσιαστεί της η Software Defined Networks τεχνολογία και η επίδρασή της στο μέλλον των δικτύων. Η εργασία αποτελείται από πέντε κεφάλαια:

- Στο πρώτο κεφάλαιο, παρουσιάζονται οι βασικές αρχές στις οποίες στηρίζεται η SDN τεχνολογία, οι έννοιες καθώς και τα δομικά στοιχεία που την απαρτίζουν.
- Στο δεύτερο κεφάλαιο, περιγράφεται αναλυτικά το Openflow πρωτόκολλο που αποτελεί το βασικότερο πρωτόκολλο επικοινωνίας στα SDN δίκτυα.
- Στο τρίτο κεφάλαιο, παρουσιάζονται εφαρμογές τις SDN τεχνολογίας σε διάφορα πεδία και είδη δικτυακών τοπολογιών.
- Στο τέταρτο κεφάλαιο, αναφερόμαστε στις προκλήσεις που έχουν να αντιμετωπίσουν τα SDN δίκτυα στο μέλλον.
- Το πέμπτο κεφάλαιο , αποτελεί το πρακτικό μέρος της εργασίας μας , όπου περιγράφεται αναλυτικά η ανάπτυξη μίας εφαρμογής για τον POX controller , με σκοπό να ελέγχει τα arp πακέτα που λαμβάνει για την αντιμετώπιση πιθανών επιθέσεων. Χρησιμοποιώντας τα κατάλληλα εργαλεία , οι επιθέσεις προσομοιώνονται και επιβεβαιώνεται η λειτουργία της εφαρμογής μας.

Στο τέλος της εργασίας μας , καταλήγουμε στο συμπέρασμα ότι τα SDN δίκτυα μπορούν να αντιμετωπίσουν τις συνεχώς αυξανόμενες απαιτήσεις των υπολογιστικών συστημάτων , ωστόσο βρίσκονται σε πρώιμο στάδιο και είναι πολλά τα βήματα που απαιτούνται για να αποτελέσουν την βασική τεχνολογία δικτύωσης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Υπολογιστικά δίκτυα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: δίκτυα ,προγραμματισμός, ασφάλεια, εικονοποίηση, εκλεκτής

ABSTRACT

This master's thesis has been carried out as a part of my master's program «Telecommunication Networks and Telematics Services» at Harokopeio University of Athens.

Goal of this thesis is to present Software Defined Networks technology and its influence at network's future. It consists of five chapters:

- At the first chapter, SDN 's basic principles, main parts and components are being presented.
- At the second one, it is being described thoroughly openflow protocol which is the main communication protocol of SDN networks.
- At the third chapter, applications of SDN technology at different fields and types of network topologies are being presented.
- At the fourth chapter, challenges that SDN networks will have to face at the future are being described.
- The fifth chapter is the practical part of our thesis. It consists of an analytical description of an application that has been developed for POX controller. Its aim is to add a security feature at the controller in order check arp packets and detect possible attacks. Proper tools have been used in order to simulate these attacks and to confirm its operation.

At the end of our thesis, we come at a conclusion that SDN networks are being able to fulfill the rapidly increasing requirements of computer systems, however as they are at an early stage, they have a lot of challenges to face before to be considered as the main network technology.

SUBJECT AREA: Computer Networks,

KEYWORDS: networks, programmability, security, virtualization, controller

Η παρακάτω εργασία αφιερώνεται στην Λαμπρινή ,τον Ντίνο, τον Στάθη,τον Γιώργο&την Έρση..

ΕΥΧΑΡΙΣΤΙΕΣ

Με την συγκεκριμένη ευκαιρία θα ήθελα να ευχαριστήσω τον καθηγητή μου Κ. Γεώργιο Δημητρακόπουλο για την ευκαιρία να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα , τους συναδέλφους μου για την συνεργασία κατά την πορεία του μεταπτυχιακού προγράμματος ,τους υπόλοιπους καθηγητές του τμήματος , και τέλος την κοινότητα των developers που ασχολούνται με την ανάπτυξη των SDN δικτύων καθώς χωρίς την βοήθειά τους δεν θα ήταν δυνατή η ολοκλήρωση της παρούσας εργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΠΡΟΛΟΓΟΣ

Η επιστήμη των υπολογιστών έχει αναπτυχθεί ραγδαία τα τελευταία χρόνια αποτελώντας απαραίτητο κομμάτι σε όλες τις πτυχές της ζωής μας. Βασικό δομικό στοιχείο αποτελεί η δικτύωση , η οποία ωστόσο παραμένει απαράλλαχτη στις βασικές αρχές της αποτελώντας ένα εμπόδιο στην ανάπτυξη καινοτόμων υπηρεσιών καθώς και στην ταχεία ανάπτυξη του Internet. Το βασικό πρόβλημα έγκειται στην χρήση και διασύνδεση του δικτυακού εξοπλισμού, με την αυξανόμενη πολυπλοκότητά του, και την υποστήριξη πολλών και διαφορετικών πρωτοκόλλων χωρίς πολλές φορές την δυνατότητα συνεργασίας μεταξύ τους. Στο συγκεκριμένο πλαίσιο , είναι δύσκολη η καινοτομία και η ανάπτυξη προσαρμοσμένων λύσεων στις ανάγκες των πελατών λόγω του αυξανόμενου κόστους , των προβλημάτων διαχείρισης του καθώς των περιορισμών της δομής του υλικού το εξοπλισμού.

Την λύση στο παραπάνω πρόβλημα έρχεται να προσφέρει η ανάπτυξη των SDN δικτύων. Με την χρήση των SDN δικτύων τα δίκτυα μετατρέπονται σε προγραμματιζόμενα μέσω λογισμικού που δεν υπόκειται σε περιορισμούς κατασκευαστή. Σκοπός του είναι ο διαχωρισμός του control plane που είναι υπεύθυνο για την λήψη αποφάσεων από τον εξοπλισμό , από το data plane που είναι υπεύθυνο για την διαχείριση των πακέτων. Την λήψη αποφάσεων(control plane) αναλαμβάνει για ολόκληρη την τοπολογία , ένας μοναδικός (ή πολλαπλοί) δικτυακός ελεγκτής(controllers) ο οποίος κατασκευάζει και ελέγχει την πλήρη τοπολογία του δικτύου .Ο controller είναι υπεύθυνος τόσο για την ενημέρωση του δικτυακού εξοπλισμού(είτε υλικού είτε εικονοποιημένου) για τον τρόπο διαχείριση των πακέτων , όσο και των εφαρμογών για την τοπολογία του δικτύου. Με αυτόν τον τρόπο είναι δυνατή η διαχείριση του δικτύου μέσω ενός μοναδικού σημείου χωρίς να απαιτείται πρόσβαση σε όλο τον εξοπλισμό. Επιπρόσθετα , είναι δυνατή η εύκολη προσθήκη και δοκιμή νέων υπηρεσιών σε μία τοπολογία , χωρίς να απαιτείται ο επαναπρογραμματισμός των συσκευών και να τίθενται σε κίνδυνο οι ισχύουσες υπηρεσίες.

Πολλές εταιρίες έχουν αναπτύξει και ενσωματώσει SDN τεχνολογίες τόσο στις λύσεις που προσφέρουν στους πελάτες τους , όσο και την διαχείριση των δικών τους δικτύων. Καθώς ακόμη βρίσκεται σε ένα πρώιμο στάδιο , πέρα από τις βασικές του αρχές πάνω στις οποίες δομείται , αποτελεί σημείο έρευνας ,ενώ αρκετές είναι και οι διαφοροποιήσεις ανάλογα την λύση που προσφέρεται. Μεγάλες είναι και οι προκλήσεις που πρέπει να αντιμετωπιστούν για την περαιτέρω ενσωμάτωση και προτίμησή τους απέναντι σε κλασσικές δικτυακές τοπολογίες.

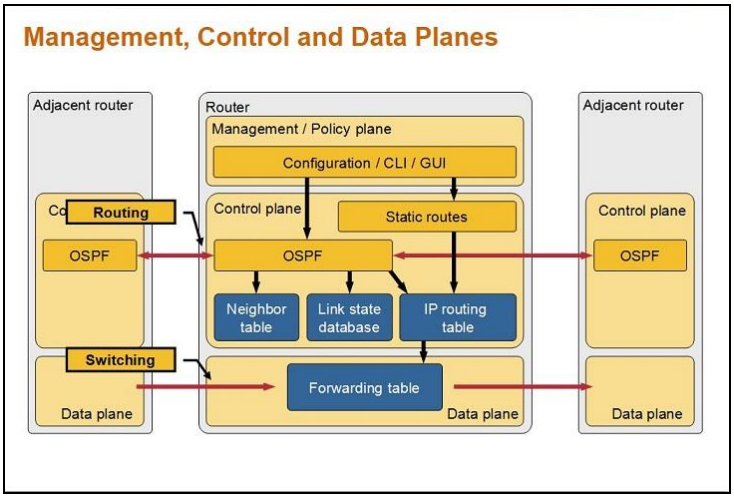
1 SOFTWARE DEFINED NETWORKS

1.1 Ορισμός και βασικές αρχές του SDN

Υπάρχει ένας μεγάλος αριθμός ορισμών που σχετίζονται με τα SDN δίκτυα καθώς βρίσκονται ακόμη σε ένα πρώιμο στάδιο ανάπτυξης. Η βασική ιδέα είναι ότι το SDN αποτελεί μία δικτυακή αρχιτεκτονική κατά την οποία η διαχείριση των αποφάσεων για την δρομολόγηση λαμβάνεται από ένα κεντρικό σημείο διαχείρισης , τον controller.

Σε όλα τα υφιστάμενα δίκτυα που βασίζονται στις βασικές αρχές των κλασσικών δικτύων υπάρχει ένας διαχωρισμός μεταξύ των λειτουργιών τους σε τρεις βασικές κατηγορίες:

1. **Management Plane:** αναφέρεται στις υπηρεσίες που προσφέρουν διαχείριση και επίβλεψη στην λειτουργία του εξοπλισμού από την απλή πρόσβαση με ένα Command Line Interface(CLI) μέχρι την χρήση εξειδικευμένων εφαρμογών όπως το Netflow.
2. **Data Plane:** αναφέρεται στις διεπαφές ή τις πόρτες που χρησιμοποιούνται για την μεταφορά και λήψη πακέτων όπου για την επεξεργασία τους χρησιμοποιούνται οι ανάλογοι πίνακες (routing πίνακες, TCAM πίνακες, Forwarding Information Base(FIB))
3. **Control Plane:** αναφέρεται στην λήψη αποφάσεων για την επεξεργασία των πακέτων καθώς και την δημιουργία των απαραίτητων πινάκων στους οποίους θα βασιστεί το Data Plane για την δρομολόγησή τους.



Εικόνα 1:Βασικές λειτουργίες δικτύου

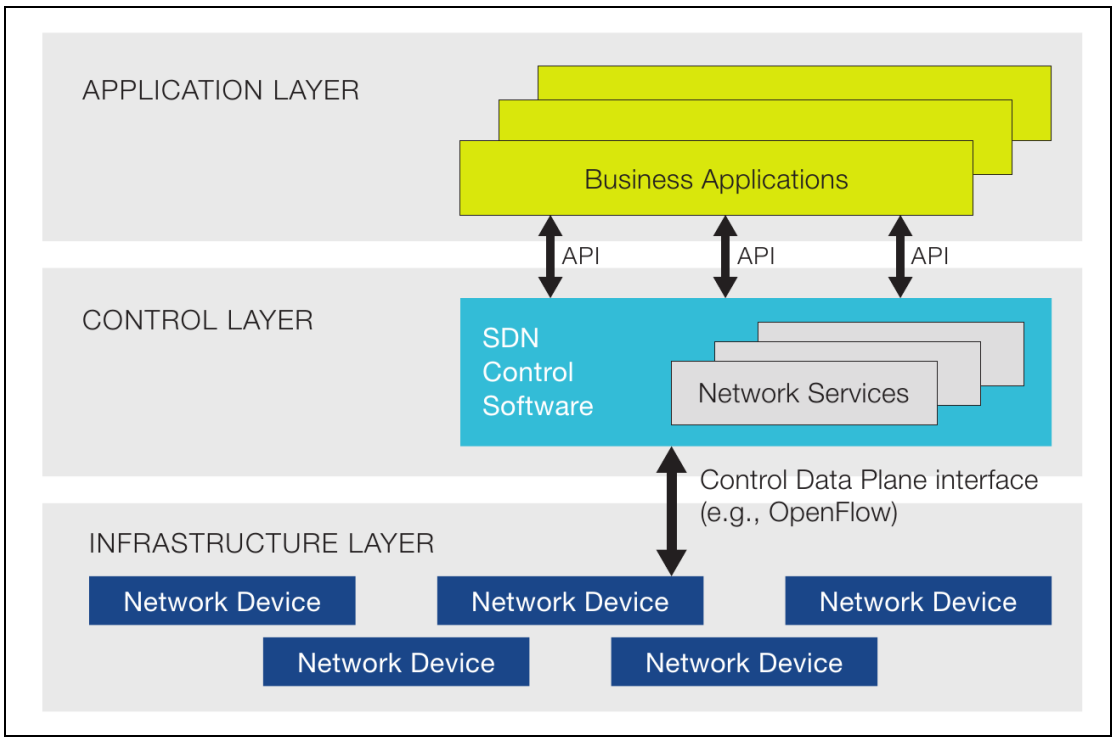
Το control plane θεωρείται το «μυαλό» του εξοπλισμού καθώς σε αυτό λαμβάνονται οι αποφάσεις για την βέλτιστη δρομολόγηση, την αποφυγή βρόχων, το failover και την ανάκτηση λειτουργίας σε περίπτωση σφάλματος. Ωστόσο σε ένα SDN δίκτυο υπάρχει διαχωρισμός μεταξύ των planes όπου το control plane αφαιρείται από τις αρμοδιότητες του εξοπλισμού και μεταφέρεται στον κεντρικό controller. Ο εξοπλισμός δεν χρειάζεται πια να λαμβάνει αποφάσεις πάρα να προωθεί τα πακέτα βασιζόμενος στις πληροφορίες και τις πίνακες που δημιουργεί και ενημερώνει ο controller.

1.2 Αρχιτεκτονική δικτύου SDN

Η αρχιτεκτονική ενός SDN δικτύου αποτελείται από 3 δομικά στοιχεία:

- **SDN εφαρμογές**: Οι SDN εφαρμογές που επικοινωνούν άμεσα με τους SDN Controllers μέσω ενός application programming interface (APIs) ώστε να λάβουν τις απαραίτητες πληροφορίες για το δομή και την συμπεριφορά ενός δικτύου. Βάση αυτών των πληροφορίων μπορούν να έχουν επακριβή εικόνα της τοπολογίας του δικτύου και πιθανών αλλαγών που διαδραματίζονται και να δράσουν ανάλογα. Παραδείγματα τέτοιων εφαρμογών είναι εφαρμογές ασφαλείας, διαχείρισης και ανάλυσης δικτύων και διάφορων ειδών εμπορικές εφαρμογές που χρησιμοποιούνται με μεγάλα data center.
- **SDN Controller**: Ο SDN Controller είναι μία λογική οντότητα ανάμεσα στις εφαρμογές και τις δικτυακές συσκευές. Λαμβάνει τις εντολές από τις εφαρμογές και τις προωθεί στις συσκευές ως κομμάτι του control plane ρόλο του. Παράλληλα, λαμβάνει τις απαραίτητες πληροφορίες για την κατάσταση του δικτύου και διάφορα συμβάντα από τις συσκευές και τις προωθεί στις εφαρμογές.
- **SDN Δικτυακές Συσκευές**: Οι δικτυακές συσκευές εκτελούν τον ρόλο της προώθησης των πακέτων στο δίκτυο ανάλογα τις πληροφορίες και τις εντολές που λαμβάνουν από τον SDN controller.

Τα APIS της SDN αρχιτεκτονικής αναφέρονται συχνά ως οι διεπαφές προς βορρά και προς νότο(northbound και southbound interfaces) ορίζοντας έτσι την επικοινωνία μεταξύ εφαρμογών , controller και συσκευών. Σαν η «προς βορά» διεπαφή ορίζεται η επικοινωνία εφαρμογών με controller και «ως προς νότο» η διεπαφή μεταξύ controller και συσκευών .Καθώς αναφερόμαστε σε ένα πλήρως εικονοποιημένο δίκτυο δεν απαιτείται τα δομικά αυτά στοιχεία να βρίσκονται στον ίδιο φυσικό χώρο.



Εικόνα 2: SDN αρχιτεκτονική

Για την επικοινωνία μεταξύ controller και εξοπλισμού απαιτείται η χρήση ενός πρωτοκόλλου , το οποίο θα μεταφέρει μηνύματα χωρίς να ενδιαφέρεται για το λογισμικό που διαθέτει και τον κατασκευαστή του , παρά μόνο για την υποστήριξή του στο συγκεκριμένο πρωτόκολλο. Για την συγκεκριμένη διαδικασία , επικρατέστερο πρωτόκολλο είναι το Openflow , στο οποίο θα αναφερθούμε σε παρακάτω κεφάλαιο.

1.3 Πλεονεκτήματα SDN

Τα επιχειρήματα για την χρήση SDN δικτύων σε σχέση με τα κοινά είναι ισχυρά:

- **Κεντρικοποιημένη διαχείριση**:Η διαχείριση του δικτύου βασίζεται σε ένα κεντρικό σημείο , τον controller που διατηρεί μία σφαιρική άποψη της τοπολογίας και εμφανίζεται στις εφαρμογές που επικοινωνεί ως ένας μοναδικός εξοπλισμός

- **Απευθείας προγραμματίσιμα δίκτυα**: Νέες εφαρμογές και λειτουργίες μπορούν να προστεθούν στον εξοπλισμό ,χρησιμοποιώντας open source εργαλεία και ανεξαρτήτως τον κατασκευαστή του

- **Μείωση του CAPEX**: Το μεγαλύτερο κόστος μίας δικτυακής συσκευής υψηλής απόδοσης σχετίζεται με το control plane και το λογισμικό που διαθέτει. Σε αυτές τις λειτουργίες έχουν επενδύσει οι κατασκευαστές , βάση αυτών επενδύει μία εταιρία στον εξοπλισμό. Με τον διαχωρισμό τους , το κόστος αυτό μεταφέρεται στον controller με αποτέλεσμα να μειώνεται σημαντικά το κόστος του δικτυακού εξοπλισμού
- **Μείωση του OPEX**: Αντίστοιχα μειώνεται και το κόστος του ανθρώπινου δυναμικού καθώς μειώνεται το κόστος των συσκευών που θα πρέπει να ρυθμιστεί. Η ρύθμιση και η προσθήκη λειτουργιών μεταφέρεται σε ένα μοναδικό σημείο από την κάθε συσκευή χωριστά.
- **Αύξηση της αξιοπιστίας**: Η μείωση της ανθρώπινης παρέμβασης σε εξοπλισμό και η δυνατότητα αυτοματοποιημένων διαδικασιών προσφέρει αύξηση της αξιοπιστίας των SDN δικτύων
- **Υψηλή διαθεσιμότητα:** Η δυνατότητα επίβλεψης και ρύθμισης της δικτυακής τοπολογίας από ένα κεντρικό σημείο αυξάνει την δυνατότητα πρόβλεψης σφαλμάτων και της έγκαιρης διόρθωσής τους, μειώνοντας την πιθανότητα απώλειας της δικτυακής τοπολογίας.
- **Καινοτομία:** Προσφέρει την δυνατότητα ανάπτυξης νέων υπηρεσιών και την δοκιμή τους χωρίς την παρεμβολή και παρεμπόδιση των ήδη ενεργών.
- **Εικονοποίηση δικτύων:** Με την μείωση της πολυπλοκότητας του εξοπλισμού γίνεται ευκολότερη η μετάβαση σε εικονοποιημένο εξοπλισμό χωρίς μεγάλες δυνατότητες και απαιτήσεις.

1.4 SDN Controllers

Υπάρχει ένα σύνολο από SDN controller που έχουν αναπτυχθεί με διαφορετικές σκοπούς και στο συγκεκριμένο υποκέφαλαιο θα αναφερθούμε σε χαρακτηριστικές περιπτώσεις περιγράφοντας την δομή τους και τις λειτουργίες τους

1.4.1 Openflow Reference Controller

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

1.4.2 Nox Controller

Ο Nox[1] αποτελεί έναν από τους κύριους Openflow Controller. Βασίζεται στην ιδέα του δικτυακού λειτουργικού συστήματος , όπου αποτελεί την βάση πάνω στην οποία αναπτύσσονται οι διάφορες λειτουργίες του δικτύου. Με αυτόν τον τρόπο , λειτουργίες ελέγχου όπως δρομολόγηση , έλεγχος κίνησης , spanning tree κλπ προσφέρονται ως εφαρμογές οι οποίες είναι εγκατεστημένες στο δικτυακό λειτουργικό σύστημα. Το λειτουργικό σύστημα προσφέρει αυτόματα μία όψη του δικτύου που περιέχει και την δικτυακή τοπολογία Openflow.Βασιζόμενη σε αυτή την τοπολογία οι αλγόριθμοι διαχείρισης και ελέγχου εκτελούνται. Παράλληλα , προσφέρει μία διεπαφή όπου συλλέγει στοιχεία για την κίνηση των δεδομένων, προσφέροντας στον προγραμματιστή την ευχέρεια να επιλέξει ποια δεδομένα χρειάζεται για την εφαρμογή του. Ακόμη, προσφέρεται το σύνολο των διαθέσιμων ενεργειών των Openflow μηνυμάτων.

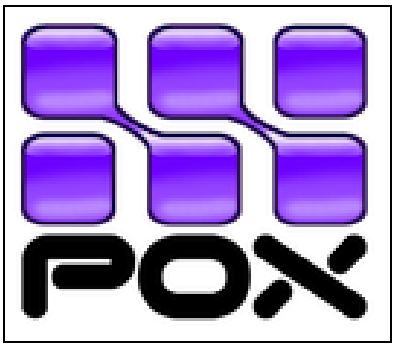
Ο Nox controller μπορεί να δημιουργεί ροές προοδευτικά η κατ’ απαίτηση. Στην κατ’ απαίτηση περίπτωση , οι ροές δημιουργούνται αποκλειστικά σύμφωνα με τα πακέτα που λαμβάνει ο controller καθώς και με τα γεγονότα για τα οποία ενημερώνεται. Αντίθετα στην περίπτωση των προοδευτικών ροών , με την σύνδεση ενός μεταγωγέα με τον controller, δημιουργούνται κάποιες δεδομένες ροές με σκοπό την αποφυγή πιθανής καθυστέρησης από την μελλοντική προσθήκη τους. Η τελική μορφή του Nox controller έχει αναπτυχθεί αποκλειστικά στην C++.

1.4.3 Pox controller

Ο Pox[2] controller αποτελεί την εξέλιξη μίας διεπαφής Openflow που είχε αναπτυχθεί σε γλώσσα python για τον Nox controller και στην συνέχεια αποσύρθηκε, και αποτέλεσε ένα βολικό μονοπάτι για την εισαγωγή στην ανάπτυξη SDN εφαρμογών. Εκτός από την υλοποίηση του openflow σε Python , προσφέρει απλές εφαρμογές όπως ο Nox, μπορεί να

εκτελεστεί σε όλα τα λειτουργικά συστήματα και ενσωματώνει μία σειρά από γραφικά εργαλεία. Έχει κερδίσει , μεγάλο μερίδιο στην εκπαίδευση και στην έρευνα , για την ανάπτυξη εφαρμογών και τεχνικών πάνω στην σχεδίαση SDN τεχνολογιών.

Αποτελεί τον controller που θα χρησιμοποιήσουμε για να αναπτύξουμε και την δική μας εφαρμογή στο πλαίσιο της συγκεκριμένης εργασίας μας .



Εικόνα 3:Pox Controller

1.4.4 Onix Controller

Ο Onix[3] είναι ένας controller που είναι εγκατεστημένος και λειτουργεί σε ένα σύνολο από φυσικούς server όπου σε κάθε server μπορεί να τρέχει ταυτόχρονα ο controller. Ο controller προσφέρει μία προγραμματιστική πρόσβαση στην τοπολογία του δικτύου, όπου το σύνολο το server ενημερώνεται ταυτόχρονα για την κατάστασή του.

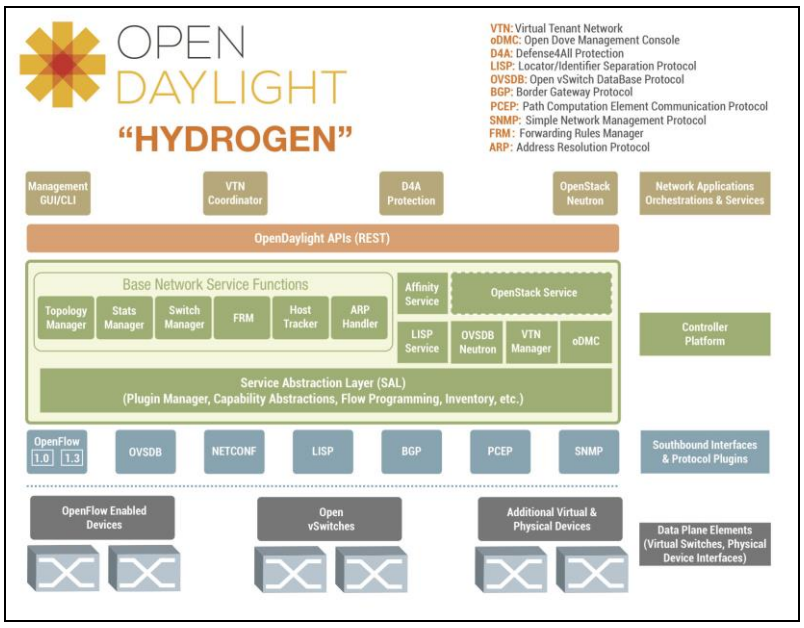
Το βασικό του πλεονέκτημα είναι ότι το control plane που διαχειρίζεται το δίκτυο αποτελεί ένα κατανεμημένο σύστημα. Σε σχέση με τον Nox , είναι πιο αξιόπιστος και πιο επεκτάσιμος καθώς πολλαπλές του διεργασίες εκτελούνται ταυτόχρονα σε διαφορετικά φυσικά μηχανήματα σε αντίθεση με τον NOX που προσφέρει μονάχα ένα κεντρικό σημείο διαχείρισης. Διαφορετικές διεργασίες του εκτελούνται για την διαχείριση του δικτύου και τον έλεγχό του η οποία τα συλλέγει και τα διαμοιράζει , και διαφορετικές για το ενημέρωση του εξοπλισμού και την δημιουργία των κατάλληλων πινάκων. Παράλληλα προσφέρει μία πιο φιλική προγραμματιστική διεπαφή ενώ δεν υποστηρίζει μονάχα το openflow ως πρωτόκολλο επικοινωνίας μεταξύ controller και εξοπλισμού.

1.4.5 Beacon και Opendaylight Controller

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

1.4.6 Opendaylight project

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



Εικόνα 4:OpenDaylight framework

1.5 Δρομολόγηση ή μεταγωγή;

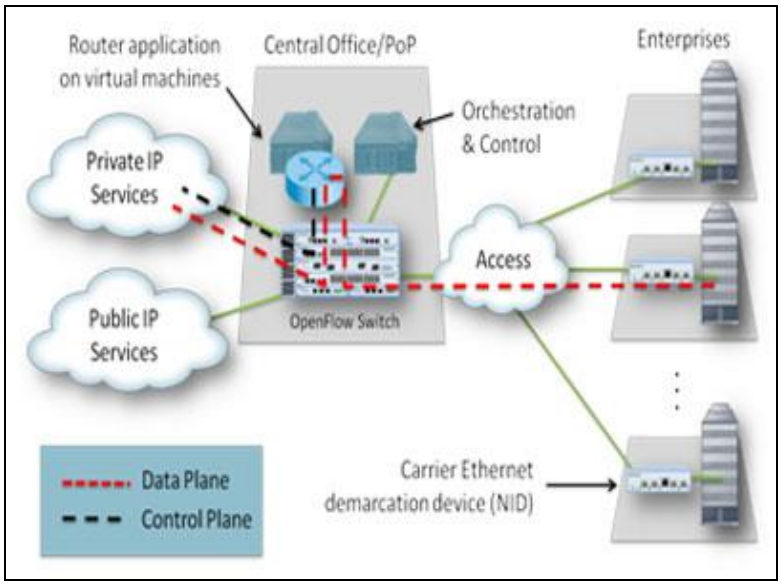
Συνηθίζουμε να διαχωρίζουμε τον δικτυακό εξοπλισμό , σε δρομολογητές και μεταγωγείς ανάλογα το επίπεδο του OSI βάση του οποίου αποφασίζουν για την προώθηση των πακέτων[4]. Οι δρομολογητές χρησιμοποιούν διευθύνσεις επιπέδου τρία (IP πρωτόκολλο) ώστε βάση των αντίστοιχων πρωτοκόλλων που χρησιμοποιούνται να δημιουργηθούν και οι αντίστοιχοι πίνακες δρομολόγησης(routing tables) , και ενώ οι μεταγωγείς σε επίπεδο δύο(Ethernet πρωτόκολλο). Αντίστοιχα υπάρχουν και μεταγωγείς επιπέδου 3 ακόμη και load balancers που οι αποφάσεις τους λαμβάνονται βάση υψηλότερου επιπέδου του OSI πρωτόκολλου(από τέσσερα έως επτά). Στην SDN τεχνολογία , χρησιμοποιείται ο όρος του προώθησης και των ροών(flows) . Σκοπός του SDN είναι η δημιουργία πινάκων ροών(flow tables) βάση των οποίων το data plane του εξοπλισμού θα δρομολογήσει τα πακέτα. Αυτές οι ροές περιέχουν πεδία όπως οι MAC διευθύνσεις , οι IP διευθύνσεις , VLAN , MPLS tags κλπ. Συνεπώς η SDN τεχνολογία δεν χρησιμοποιεί τους όρους της δρομολόγησης ή της μεταγωγής αλλά τον όρο της προώθησης. Σε επίπεδο εξοπλισμού χρησιμοποιείται πάντως η έννοια του μεταγωγέα.

1.6 SDN και Network Function Virtualization

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

Η NFV ξεκίνησε όταν οι service providers προσπάθησαν να επιταχύνουν την διαδικασία ανάπτυξη νέων δικτυακών υπηρεσιών και να αποφύγουν τους περιορισμούς του εξοπλισμού. Βασιζόμενη στις βασικές αρχές των τεχνολογιών εικονοποίησης στην επιστήμη των υπολογιστών , ξεκίνησε η ανάπτυξη του NFV που επιτάχυνε την ανάπτυξη και την καινοτομία.

Παρόλο που και οι δύο τεχνολογίες έχουν ως βάση την διαχείριση των δικτύων , βασίζονται σε διαφορετικές αρχές[5]. Η SDN τεχνολογία στηρίζεται στον διαχωρισμό του control plane και του data plane προσφέροντας μία κεντρικοποιημένη άποψη του δικτύου ενώ η NFV στην βελτιστοποίηση των παρεχόμενων υπηρεσιών μέσω της εικονοποίηση τους. Συμπερασματικά , οι δύο τεχνολογίες δεν δρουν ανταγωνιστικά αλλά συμπληρωματικά καθώς μπορούν να συνεργαστούν ενώ και στην SDN τεχνολογία ο ρόλος του εξοπλισμού είναι μειωμένος και μπορεί να αντικατασταθεί από εικονοποιημένες τοπολογίες.



Εικόνα 5:Αρχιτεκτονική NFV

1.7 Επίλογος

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

2 OPENFLOW ΠΡΩΤΟΚΟΛΛΟ

2.1 Εισαγωγή

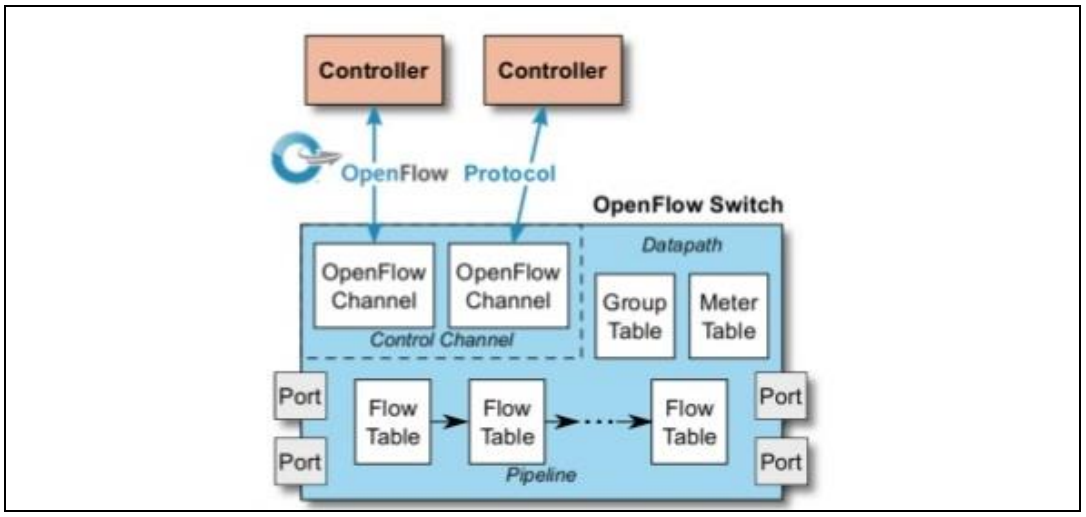
Το Openflow πρωτόκολλο θεωρείται από πολλούς λανθασμένα αναπόσπαστο κομμάτι των SDN δικτύων. Στην πραγματικότητα , μέσα στην τεράστια δομή ενός SDN δικτύου αποτελεί την προς νότο διεπαφής ενός SDN controller που επιτρέπει την επικοινωνία μεταξύ του controller και των μεταγωγών. Ωστόσο δεν αποτελεί το μοναδικό πρωτόκολλο επικοινωνίας

Τα πρότυπά του καθορίζονται από το Open Networking Foundation[6] ενός μη κερδοσκοπικού οργανισμού που ιδρύθηκε από έναν μεγάλο αριθμό εταιριών με σκοπό την προώθηση και την ανάπτυξη SDN δικτύων.

2.2 Δομικά μέρη Openflow

Τα βασικά δομικά μέρη του Openflow πρωτοκόλλου αποτελούν[7]:

- Ο Openflow controller
- Οι Openflow μεταγωγείς που αποτελούνται από
  - I. πόρτες
  - II. Το ασφαλές κανάλι Openflow
  - III. Πίνακες ροών



Εικόνα 6:Δομή Openflow

2.2.1 Openflow controller

Το Openflow αποτελεί το μέσο επικοινωνίας μεταξύ ενός controller και του δικτυακού εξοπλισμού μέσω ενός καναλιού. Ένας controller που χρησιμοποιεί το openflow έχει την δυνατότητα να αποστέλλει μηνύματα στο δίκτυο μέσω ενός καναλιού επικοινωνίας ,χωρίς το ίδιο το πρωτόκολλο να μπορεί να καθορίσει την δομή του , ακόμη και παράλληλα με κάποιο άλλο πρωτόκολλο. Στο προηγούμενο κεφάλαιο παρουσιάσαμε χαρακτηριστικά παραδείγματα.

2.2.2 Openflow μεταγωγέας

Ένας openflow μεταγωγέας αποτελεί εξοπλισμό ο οποίος έχει την δυνατότητα να προσπελάσει τα μηνύματα που του αποστέλλονται από τον controller και αφορούν την διαχείριση των πακέτων. Οι διαχείριση βασίζεται σε ένα σύνολο πινάκων, τους flow πίνακες ,τους group πίνακες και τους meter πίνακες. Οι πίνακες αυτοί συμπληρώνονται βάση των μηνυμάτων που λαμβάνονται από τον controller.

Οι μεταγωγείς αποτελούνται από δύο κατηγορίες:

- **Openflow-only μεταγωγείς**: οι οποίοι έχουν την δυνατότητα να διαχειριστούν πακέτα μόνο βάση του openflow πρωτοκόλλου
- **Υβριδικοί μεταγωγείς**: οι οποίοι αποτελούν υβριδικό εξοπλισμό με δυνατότητα να υποστηρίξουν τόσο openflow μεταγωγή όσο και παραδοσιακή μεταγωγή με την χρήση Ethernet πρωτοκόλλου.

Οι πόρτες σε έναν Openflow μεταγωγέα λειτουργούν με παρόμοιο τρόπο όπως σε κάθε μεταγωγέα. Τα πακέτα εισέρχονται σε μία πόρτα εισόδου , επεξεργάζονται από τον μεταγωγέα και μεταφέρονται σε μία πόρτα εξόδου. Παρόμοια , μία πόρτα μπορεί να προστεθεί, να αλλάξει ή να αφαιρεθεί. Καθώς μπορεί να υπάρχουν εγγραφές σε openflow πίνακες που κατευθύνουν πακέτα σε πόρτες που έχουν αφαιρεθεί , είναι σημαντικό να ενημερώνεται ο controller για κάθε αλλαγή στην κατάσταση των πορτών ώστε αντίστοιχα να ενημερώνεται και οι πίνακες.

Το Openflow ορίζει τρία είδη πορτών τα οποία θα πρέπει όλα να υποστηρίζονται από ένα openflow μεταγωγέα:

I. **Φυσική πόρτα**: Μία φυσική πόρτα αναφέρεται σε μία πόρτα του δικτυακού εξοπλισμού. Όταν στο υλικό υποστηρίζονται περισσότερα του ενός εικονοποιημένου μεταγωγέα , μία openflow φυσική πόρτα αποτελεί ένα μέρος μίας πόρτας του εξοπλισμού αντίστοιχα με την λειτουργία των VLANs.

II. **Λογική πόρτα:** Οι λογικές πόρτες δεν αντιστοιχούν στις φυσικές του εξοπλισμού. Αντίστοιχα με οποιοδήποτε λογική διεπαφή οποιουδήποτε δικτυακού εξοπλισμού μπορούν να αναφέρονται σε tunnel , loopback κλπ διεπαφές.Το πως αντιστοιχίζονται σε φυσικές πόρτες είναι ανεξάρτητο του Openflow. Το openflow τις αντιμετωπίζει παρόμοια με τις φυσικές.

III. **Κλειστές(reserved) πόρτες:** Αποτελεί ειδική κατηγορία πορτών όπου χρησιμοποιούνται για εσωτερική διαχείριση πακέτων. Χωρίζονται σε 5 υποχρεωτικές:

- **ALL:** αναφέρονται σε όλες τις πόρτες που μπορούν να χρησιμοποιηθούν σαν εξωτερικές πόρτες. Δηλαδή όλες οι πόρτες εκτός της εισερχόμενης πόρτας και των πορτών που έχουν αποκλειστεί από την προώθηση πακέτων
- **CONTROLLER:** είναι η πόρτα όπου ο switch επικοινωνεί μέσω του ασφαλούς καναλιού με τον controller.
- **TABLE:** είναι η εισερχόμενη πόρτα σε ένα pipeline
- **IN_PORT:** είναι μία εισερχόμενη και εξερχόμενη πόρτα ταυτόχρονα. Χρησιμοποιείται όταν κάποιο πακέτο πρέπει να επιστρέψει από την πόρτα που προήλθε
- **ANY:** μπορεί να αποτελέσει εισερχόμενη , εξερχόμενη ή οποιαδήποτε πόρτα. Χρησιμοποιείται όταν το Openflow απαιτεί περιγραφή πόρτας και δεν υπάρχει αντίστοιχη κατηγορία.

Παράλληλα υπάρχουν και οι προαιρετικές πόρτες:

- **LOCAL:** οι εσωτερικές και εξωτερικές πόρτες του εξοπλισμού για την διαχείριση του μεταγωγέα
- **NORMAL:** Αφορούν τους υβριδικούς μεταγωγείς και αναφέρονται στις εξερχόμενες πόρτες κατά την μετάβαση από Openflow στην κανονική λειτουργία του εξοπλισμού
- **FLOOD:** παρόμοιες με τις NORMAL αφορούν τους υβριδικούς μεταγωγείς. Χρησιμοποιούνται για όταν πακέτα πρέπει να αποσταλούν σε όλες τις πόρτες(flooded) μέσω της κανονικής λειτουργίας του εξοπλισμού.

2.2.3 Ασφαλές κανάλι

Για την επικοινωνία μεταξύ controller και εξοπλισμού απαιτείται ένα κανάλι όπου θα μεταφέρονται όλα τα μηνύματα και τα πακέτα που αφορούν την λειτουργία του δικτύου. Το κανάλι δημιουργείται μέσω TCP σύνδεσης. Παρόλο που δεν απαιτείται να είναι κρυπτογραφημένη η επικοινωνία συνήθως κρυπτογραφείται μέσω του TLS πρωτοκόλλου.

Με την δημιουργία της TCP επικοινωνίας , ο controller και ο μεταγωγέας διαπραγματεύονται την έκδοση του openflow που θα χρησιμοποιήσουν. Κάθε πλευρά αναφέρει την νεότερη έκδοση που υποστηρίζει και χρησιμοποιείται η νεότερη που υποστηρίζουν και οι δύο πλευρές. Χρησιμοποιείται η διαδικασία ECHO REQUEST /REPLY για τον έλεγχο της ποιότητας της σύνδεσης.

Σε περίπτωση που δεν υποστηρίζεται καμία κοινή έκδοση , αποστέλλεται μήνυμα σφάλματος και η σύνδεση τερματίζει. Αντίστοιχα σε περίπτωση αποσύνδεσης , ο μεταγωγέας εισέρχεται σε μία από τις παρακάτω καταστάσεις:

- **Fail Secure Mode:** όπου δεν πραγματοποιούνται προσπάθειες επικοινωνίας με τον controller από την πλευρά του μεταγωγέα. Χρησιμοποιεί τις εγγραφές στους υπάρχοντες πίνακες για την διαχείριση των πακέτων μέχρι να λήξει η ισχύ τους.
- **Fail Standalone Mode:** Ο μεταγωγέας επανέρχεται στην κανονική του λειτουργία μία επιλογή που αφορά αποκλειστικά τους υβριδικούς μεταγωγείς/

Εφόσον επανασυνδεθεί ο εξοπλισμός με τον controller , είτε ο εξοπλισμός μπορεί να ζητήσει την ανανέωση των υπάρχων εγγραφών ,είτε ο ίδιος ο controller να διαγράψει και να ανανεώσει τις εγγραφές ,καθώς η διατήρηση τους εγκυμονεί κινδύνους για την λειτουργία του δικτύου.

Η αξιοπιστία της αποστολής και λήψης των Openflow μηνυμάτων βασίζεται αποκλειστικά στον μηχανισμό TLS και στην TCP σύνδεση καθώς δεν υφίσταται κάποια λειτουργία επιβεβαίωσης λήψης καθώς και ούτε διατήρησης της σωστής σειράς.

2.3 Openflow πίνακες

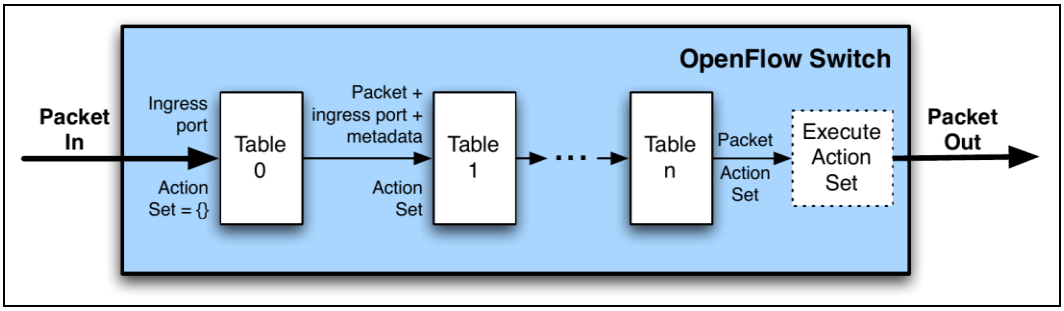
Στόχος της ανταλλαγής μηνυμάτων μεταξύ Controller και μεταγωγέα είναι η δημιουργία openflow πινάκων όπου βάση των εγγραφών τους θα διαχειρίζονται τα πακέτα οι μεταγωγείς.

Κάθε εγγραφή σε έναν πίνακα αποτελείται από τα εξής πεδία:

- **Match:** καθορίζει τις συνθήκες βάση των οποίων θεωρείται ότι ένα πακέτα ταιριάζει στην εγγραφή Για παράδειγμα η πόρτα εισόδου, ή η επικεφαλίδα είτε του Ethernet είτε του IP πακέτου ακόμη και συνδυασμός των παραπάνω
- **Priority:** σε συνδυασμό με το match πεδίο καθορίζει την προτεραιότητα της εγγραφής
- **Counter:** αναφέρεται σε στατιστικά στοιχεία των πακέτων που ταιριάζουν στην εγγραφή.
- **Instruction:** καθορίζει τις ενέργειες που θα πρέπει να εκτελέσει ο μεταγωγέας όταν κάποιο πακέτο ταιριάζει στην εγγραφή.
- **Timeouts:** αναφέρεται στον μέγιστο χρόνο ισχύς της εγγραφής
- **Flags:** χρησιμοποιείται για την αλλαγή του τρόπου διαχείρισης των εγγραφών

Σε περίπτωση που κάποιο πακέτο ταιριάζει σε περισσότερες από μία εγγραφές, εκτελούνται τα instructions της εγγραφής που ταιριάζει σε περισσότερο στο πεδίο match.

Ακόμη , αντί για την διατήρηση ενός μόνο πίνακα συνηθίζεται η δημιουργία πολλών που συνδέονται μεταξύ τους . Οι πίνακες έχουν αύξοντα αρίθμηση ξεκινώντας από το 0. Το πακέτο με την άφιξη του ελέγχεται βάση των εγγραφών του πρώτου πίνακα και συνεχίζει στους επόμενους. Σε κάποιον από τους πρώτους πίνακες μπορεί να υπάρχει κάποια εγγραφή που θα ταιριάζει και θα εκτελεστεί μία εντολή Go-to η οποία ουσιαστικά θα προσπελάσει τους ενδιάμεσους πίνακες ώστε να ελεγχθούν οι εγγραφές κάποιου μεταγενέστερου πίνακα που θα περιέχει τις σωστές εγγραφές για το συγκεκριμένο πακέτο. Συνήθως τελευταίο βρίσκεται ένας *table-miss* πίνακας όπου αναφέρεται στις διαχείριση πακέτων που δεν ταιριάζουν σε καμία εγγραφή κανενός πίνακα. Σε αντίθετη περίπτωση το πακέτο απορρίπτεται από τον μεταγωγέα.

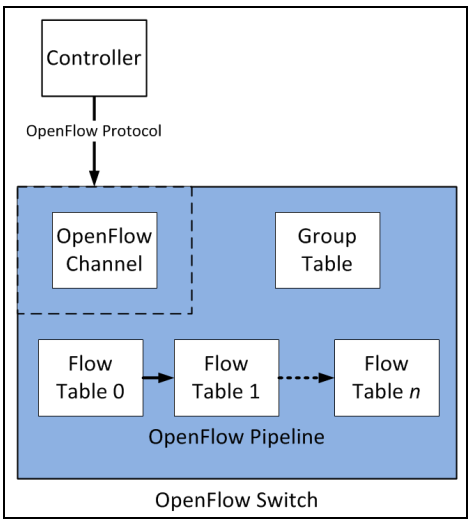


Εικόνα 7:Openflow πίνακες

Αντίστοιχα με τους Flow πίνακες υπάρχουν οι Group πίνακες όπου οι εγγραφές τους αναφέρονται σε ένα σύνολο πακέτων που θα πρέπει να αντιμετωπιστούν παρόμοια. Αναγνωρίζοντας ένα πακέτο ως μέλος ενός group μπορεί ένα σύνολο κανόνων να εκτελεστεί αποτελεσματικά για μία ροή πακέτων.

Οι group πίνακες περιέχουν τα εξής πεδία:

- **Group identifier:** ένας 32-bit αριθμός αναγνωριστικός του group
- **Group type:** που αποτελείται από τα εξής είδη:
  - **I. All:** εκτελείται το σύνολο των ενεργειών του group, χρησιμοποιείται για multicast ή broadcast πακέτα.
  - **II. Select:** χρησιμοποιώντας κάποιο αλγόριθμό εκτελείται μόνο ένα σύνολο ενεργειών για το πακέτο
  - **III. Indirect:** εκτελείται μόνο μία ενέργεια για όλα τα πακέτα που ταιριάζουν στο group
  - **IV. Fast Failover:** αναφέρεται σε ενέργειες που αφορούν συγκεκριμένη πόρτα. Όπως προδίδει το όνομά της , χρησιμοποιείται κυρίως σε περίπτωση αλλαγής της δρομολόγησης όταν αποτυγχάνει η λειτουργία κάποιας πόρτας
- **Counter:** αναφέρεται σε στατιστικά στοιχεία των πακέτων που ταιριάζουν στην εγγραφή.
- **Action buckets:** όπου περιέχεται ένα σύνολο ενεργειών προς εκτέλεση.



Εικόνα 8:Group πίνακες

Η τελευταία έκδοση του Openflow εισήγαγε και τον Meter πίνακα όπου προσφέρει την δυνατότητα στον controller να δημιουργήσει έναν μηχανισμό Quality Of Service όπου μπορεί να περιοριστεί ο ρυθμός μίας ροής πακέτων.

Τα πεδία που περιλαμβάνουν οι εγγραφές τους είναι τα εξής:

- **Meter identifier:** ένας 32-bit αριθμός αναγνωριστικός του πίνακα
- **Meter bands:** μία λίστα από κανόνες που καθορίζει τον ρυθμό και την διαχείριση των πακέτων. Αποτελείται από:
  I. **Band Type:** καθορίζει την διαχείριση των πακέτων
  II. **Rate:** καθορίζει το όριο πάνω από το οποίο ενεργοποιούνται οι κανόνες
  III. **Burst:** καθορίζει τον βαθμό ανάλυσης των κανόνων
  IV. **Counters:** αυξάνεται με τον αριθμό των πακέτων που διαχειρίζεται
  V. **Type specific arguments:** χρησιμοποιείται όταν απαιτούνται περισσότερες ενέργειες

- **Counters:** αυξάνεται με τον αριθμό των πακέτων που διαχειρίζεται

2.4 Openflow μηνύματα

Για την συμπλήρωση των πινάκων και την δημιουργία κανόνων για την διαχείριση των πακέτων , ένα σύνολο openflow μηνυμάτων ανταλλάσσονται μεταξύ controller και switch

2.4.1 Δομή Openflow μηνυμάτων

Τα μηνύματα openflow ενθυλακώνονται σε headers ,οι οποίοι περιλαμβάνουν τα εξής πεδία:

- **Version:** καθορίζει την έκδοση του openflow που χρησιμοποιείται
- **Type:** αναφέρεται στο είδος του μηνύματος.
- **Length:** το μήκος του μηνύματος
- **Transaction ID(XID):** χρησιμοποιείται για την αναγνώριση ενός μηνύματος για τον διαχωρισμό του από παρόμοια μηνύματα.

Τα είδη των μηνυμάτων διαχωρίζονται σε τρείς κατηγορίες :

- **Controller-to-Switch messages:** τα συγκεκριμένο μηνύματα χρησιμοποιούνται από τον controller την διαχείριση του εξοπλισμού.
- **Asynchronous messages:** αποστέλλονται από τον μεταγωγέα στον controller. Μπορεί να περιλαμβάνει κάποιο πακέτο που δεν γνωρίζει πως να το διαχειριστεί ή κάποια ενημέρωση για πιθανή αλλαγή σε κάποιων από τους πίνακες.
- **Symmetric messages:** αποστέλλονται και από τις δύο πλευρές και περιλαμβάνουν μηνύματα hellos ,echo requests και replies.

2.4.2 Διαχείριση ροών

Όπως αναφέραμε στην προηγούμενη ενότητα, το πεδίο μίας εγγραφής βάση της οποίας θα αντιμετωπιστεί ένα πακέτο είναι το πεδίο *match.* Οι διαθέσιμες επιλογές του συγκεκριμένου πεδίου διαχωρίζονται σε τρεις κατηγορίες:

- **Flow match:** αναφέρεται σε έναν συνδυασμό παραμέτρων όπως οι διευθύνσεις layer 2 και layer 3, τα QOS bits και οι πόρτες του TCP.
- **Header match:** αναφέρεται σε πεδία που αντιστοιχούν με τους headers των Layer 2 και Layer 3 πακέτων
- **Pipeline match:** αναφέρεται σε πεδία που προστίθενται στα πακέτα για την προσπέλασή τους σε περίπτωση πολλαπλών πινάκων
- **Experimenter Flow match:** ένα προαιρετικό πεδίο που μπορεί να χρησιμοποιηθεί σε περιπτώσεις ερευνών και δοκιμών και γι' αυτό τον λόγο δεν διαθέτει καθορισμένες επιλογές.

Όταν βρεθεί μία εγγραφή που αντιστοιχεί , στην συνέχεια θα πρέπει να καθοριστεί το σύνολο των ενεργειών για την προσπέλασή του. Για το σύνολο των ενεργειών χρησιμοποιούνται έξι όροι στην ορολογία του Openflow οι οποίοι είναι άρρηκτα συνδεδεμένοι μεταξύ τους.

1. **Instructions:** Αποτελούνται από έξη κατηγορίες. Κάποιες συνδέονται άμεσα με **actions ,** κάποιες άλλες όχι , όλες ωστόσο προκαλούν αλλαγές είτε σε πακέτα είτε σε εγγραφές είτε σε προσπέλαση των πινάκων.

2. **Instruction Sets:** αποτελείται από ένα σύνολο instructions, ωστόσο μπορεί να περιέχει μόνο ένα από κάθε κατηγορία επομένως μέχρι έξη στο σύνολο.

3. **Action lists:** Μία action list μπορεί να περιλαμβάνει μία ή περισσότερες ενέργειες και σχετίζεται είτε με την Apply-Actions instruction είτε με την Write-Actions instructions είτε με τα πακέτα PACKET-OUT.

- Σε περίπτωση apply-actions instruction οι ενέργειες εκτελούνται με την σειρά που βρίσκονται στην λίστα.
- Σε περίπτωση Write-actions instruction ενέργειες που περιέχονται στην action list μεταφέρονται σε action set και δεν εκτελούνται.
- Τα PACKET OYT πακέτα αποστέλλονται από τον controller και περιλαμβάνουν μία action list για την τροποποίηση η προώθηση κάποιου πακέτου.

4. **Actions:** αποτελούν τις ενέργειες προς εκτέλεση

5. **Action Sets:** χρησιμοποιείται κατά την προσπέλαση πολλαπλών πινάκων. Καθώς ένα πακέτο συγκρίνει τις εγγραφές όλων των πινάκων σε περίπτωση που βρει εγγραφή που ταιριάζει προσθέτει τις action list των εγγραφών με write-instructions στο action set του. Με το τέλος των πινάκων , εκτελούνται οι ενέργειες που περιέχονται στο action set του. Κάθε action set μπορεί να περιέχει μόνο ένα είδος ενέργειας.

6. **Action Buckets:** αποτελούνται από ένα σύνολο από action sets και χρησιμοποιούνται σε περιπτώσεις group πινάκων.

2.5 Επίλογος

Στο συγκεκριμένο κεφάλαιο προχωρήσαμε σε μία εκτενή παρουσίαση του Openflow πρωτοκόλλου που αποτελεί το κύριο πρωτόκολλο επικοινωνίας μεταξύ Controller και εξοπλισμού και έχει διαδραματίσει έναν σημαντικό ρόλο στην συνολική ανάπτυξη των SDN δικτύων και την προώθησή τους. Αναφερθήκαμε στην δομή του, τον τρόπο επικοινωνίας με τον εξοπλισμό , την δημιουργία των πινάκων ροών και τον εμπλουτισμό τους με εγγραφές μέσω των Openflow μηνυμάτων.

3 ΕΦΑΡΜΟΓΕΣ SDN ΔΙΚΤΥΩΝ

Στο συγκεκριμένο κεφάλαιο θα παρουσιάσουμε την εφαρμογή των SDN δικτύων , σε μία σειρά από τεχνολογίες δικτύωσης με διαφορετικά χαρακτηριστικά και απαιτήσεις

3.1 Δίκτυα ευρείας περιοχής(Wide Area Networks)

Τα δίκτυα ευρείας περιοχής χρησιμοποιούνται ιστορικά για την σύνδεση ενός συνόλου δικτύων που διαχωρίζονται γεωγραφικά αλλά ανήκουν στον ίδιο οργανισμό.
Ένα σύνολο από τεχνολογίες χρησιμοποιούνται για προσφέρουν WAN συνδεσιμότητα όπως:

- **Μισθωμένες γραμμές**: Η συγκεκριμένη τεχνολογία σύνδεσης σημείο προς σημείο χρησιμοποιήθηκε κατά κόρον στις πρώτες μέρες της δικτύωσης με κύριο χαρακτηριστικό τους το υψηλό κόστος.

- **Μεταγωγή κυκλώματος**: οι Dialup συνδέσεις αποτελούν ένα χαρακτηριστικό παράδειγμα αυτού του είδους της σύνδεσης που χαρακτηρίζονταν από περιορισμένο εύρος ζώνης αλλά και από οικονομική υλοποίηση

- **Μεταγωγή πακέτων**: Δίκτυα με ιεραρχία ,όπου η θέση του δικτύου του πάροχου είναι κεντρική και διανέμει την κίνηση βασιζόμενο σε τεχνολογίες όπως X.25 και Frame Relay

- **Κυψελωτά δίκτυα**: Τεχνολογία όπως η ATM που χαρακτηρίζεται από εικονικά κυκλώματα και σταθερό μέγεθος πακέτων , χρησιμοποιούνται και σήμερα.

- **Ψηφιακή Συνδρομητική Γραμμή(DSL):** Η χαρακτηριστικότερη τεχνολογία συνδεσιμότητας των σπιτιών μας στο διαδίκτυο την σημερινή εποχή.

Οι μεγαλύτερες προκλήσεις που αντιμετωπίσουν οι μηχανικοί δικτύων στα δίκτυα ευρείας περιοχής είναι η αξιοπιστία και η αποτελεσματικότερη χρήση του διαθέσιμου εύρους ζώνης. Σε σύγκριση με τα τοπικά δίκτυα όπου μπορούν να δημιουργηθούν εύκολα εφεδρικές τοπολογίες , απλά προσθέτοντας περισσότερες πόρτες και συνδέσεις, στα WAN το κόστος των συνδέσεων είναι μεγάλο με αποτέλεσμα να απαιτείται η όσο το δυνατόν αποτελεσματικότερη χρήση τους όπως και του εύρους ζώνης. Η δυνατότητα που προσφέρει η SDN τεχνολογία είναι η κεντρική διαχείριση και ελέγχου του δικτύου. Οι εφεδρικές συνδέσεις χρησιμοποιούνται κατά κόρον σε όλα τα δίκτυα , ωστόσο κατά την διάρκεια της

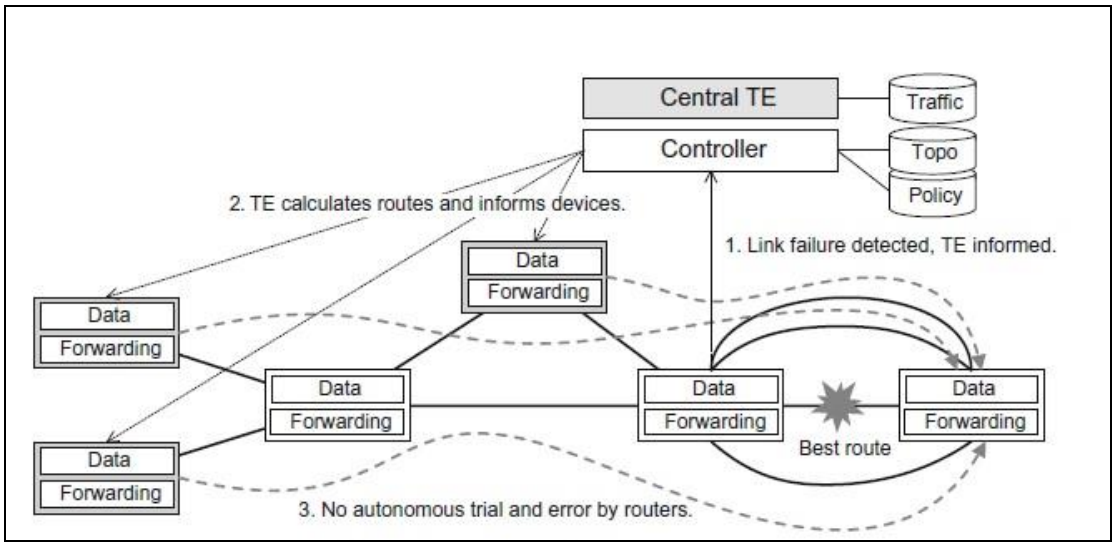
μετάβασης στα παραδοσιακά δίκτυα, η δρομολόγηση των πακέτων είναι αμφίβολη όπως και επιλογή της βέλτιστης λύσης , ενώ περιορίζονται αρκετά από τα control plane των συσκευών.

Αντίθετα, στα SDN δίκτυα χρησιμοποιείται για την διαχείριση ένας server υψηλής απόδοσης που έχει τις δυνατότητες να προβαίνει στις βέλτιστες αποφάσεις λαμβάνοντας υπόψιν όλες τις παραμέτρους.

3.1.1 SDN-WAN σε MPLS δίκτυα

Στα MPLS (Multiprotocol Label Switching) δίκτυα χρησιμοποιείται ή έννοια του LSP (label switched paths) μονοπατιού. Το LSP είναι ένα μονοπάτι μέσα στο MPLS δίκτυο όπου χρησιμοποιούνται οι ενδιάμεσοι δρομολογητές για την προώθηση πακέτων μεταξύ δύο σημείων. Σε περίπτωση σφάλματος , θα πρέπει όλα τα LSPs να αναδρομολογηθούν , μία σειριακή διαδικασία , όπου το πρώτο LSP επιλέγει το μονοπάτι και το εύρος ζώνης που απαιτείται και συνεχίζεται μέχρι και το τελευταίο με αποτέλεσμα να είναι και αρκετά χρονοβόρα.

Η Google[8] είναι μία από τις εταιρίες που έχει προχωρήσει στην υιοθέτηση SDN για την βελτιστοποίηση της άνωθεν διαδικασίας. Συνδέοντας το MPLS δίκτυό της με έναν openflow controller και χρησιμοποιώντας έναν μηχανισμό ονόματι MPSL-TE , είναι δυνατόν να προαποφασίζεται η βέλτιστη διαδρομή και το εύρος ζώνης που θα πρέπει να ανατεθεί ανά LSP δυναμικά και πριν την παραμετροποίησή τους.



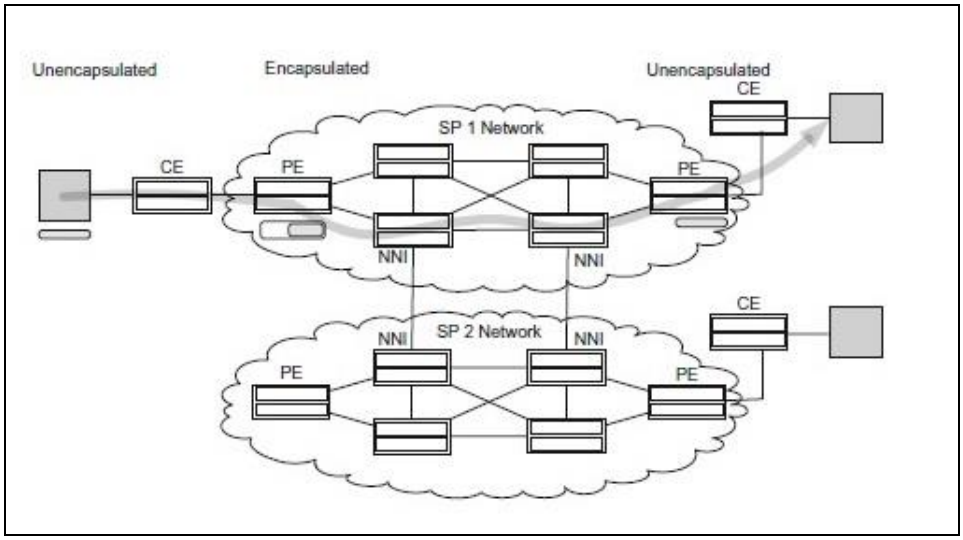
Εικόνα 9:SDN σε MPLS δίκτυο

3.2 Δίκτυα πάροχων

Τα δίκτυα πάροχων (SP) αποτελούν δίκτυα ευρείας κλίμακας που μεταφέρουν μεγάλο όγκο δεδομένων και φωνής συνήθως εκ μέρους των πάροχων υπηρεσιών διαδικτύου(ISPs). Στο παρελθόν , η κύρια υπηρεσία τους ήταν η παροχή φωνής , ωστόσο αυτή την στιγμή καλύπτουν ένα μεγάλο εύρος υπηρεσιών , ενώ έχει εκτιναχθεί και ο όγκος των δεδομένων που μεταφέρουν. Χωρίς ένα κεντρικό σύστημα διαχείρισης του δικτύου, το κόστος για την προσφορά των υπηρεσιών είναι μεγάλο. Συνήθως, την διαχείριση του εύρους ζώνης , αναλαμβάνουν πλατφόρμες διαχείρισης , οι οποίες βασίζονται σε πρότυπα κίνησης τα οποία όμως πολλές φορές για να εφαρμοστούν απαιτούν ανθρώπινη παρέμβαση. Η δυνατότητα της δυναμικής προσφοράς εύρους ζώνης , και επιλογής της βέλτιστης διαδρομής βάση εύρους ζώνης και καθυστέρησης , αποτελεί βασικό μέλημα στην σύγχρονη εποχή.

Ακόμη , σημαντική απαίτηση αποτελεί και η μείωση του κόστους του εξοπλισμού. Πολλές φορές , το OPEX κόστος για την διαχείριση μίας πολύπλοκης μονάδας , ξεπερνάει το αντίστοιχο CAPEX για την αγορά της. Οπότε μία απλή συσκευή , μειώνει διπλά το απαιτούμενο κόστος.

Ο πάροχος υπηρεσιών, είναι υπεύθυνος για την μεταφορά των πακέτων μεταξύ δύο σημείων που περιέχει το δίκτυο του πρώτου σημείου (edge network), το δίκτυο του πάροχου και το δίκτυο του δεύτερου σημείου. Τα συγκεκριμένα δίκτυα διαχωρίζονται με όρια που μπορούν να αυξηθούν με την προσθήκη περισσότερων πάροχων. Τα δίκτυα ενώνονται στην άκρη πελάτη(CE) και στην άκρη πάροχου(PE). Η διεπαφή δίκτυο προς δίκτυο(NNI) είναι το όριο μεταξύ δύο πάροχων. Η δρομολόγηση σε κάθε δίκτυο είναι ξεχωριστή ενώ το πακέτο κάθε φορά ενθυλακώνεται και με την απαραίτητη πληροφορία για την δρομολόγησή του.



Εικόνα 10:Δίκτυο πάροχων υπηρεσιών

Η προσφορά της SDN τεχνολογίας στα δίκτυα πάροχων[9], βασίζεται στην αποτελεσματική διαχείριση του εύρους ζώνης, την μείωση του CAPEX και του OPEX και την εύκολη εφαρμογή πολιτικών στα όρια του δικτύου. Ένα SDN δίκτυο προσφέρει την δυνατότητα για αυτοματοποιημένες αλλαγές με την ελάχιστη διακοπή των υπηρεσιών. Ακόμη , το κόστος του εξοπλισμού είναι μικρό ενώ το openflow πρωτόκολλο υποστηρίζει την ενθυλάκωση των πακέτων για την μεταφορά μεταξύ των δικτύων.

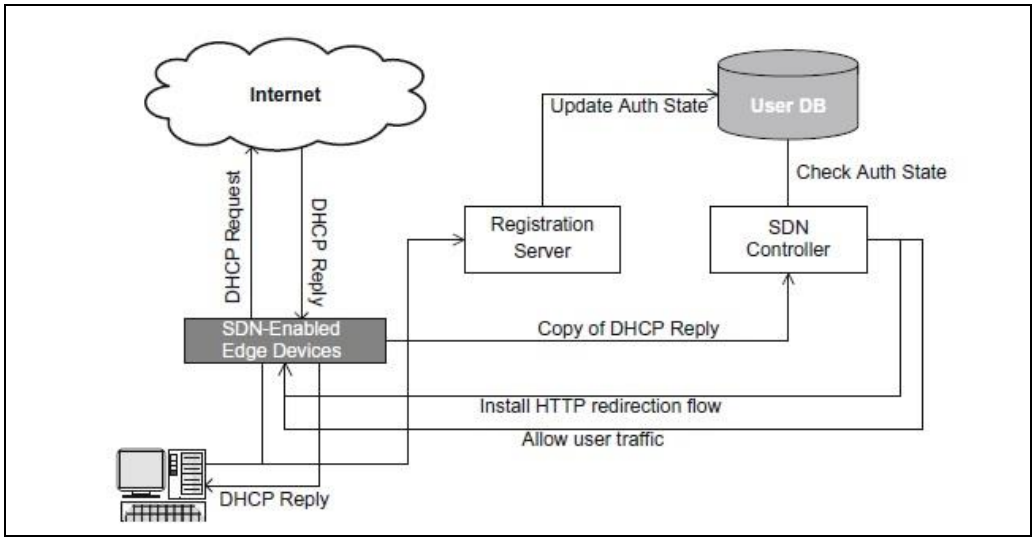
3.3 Campus Δίκτυα

Τα campus δίκτυα αποτελούν ένα σύνολο από τοπικά δίκτυα τα οποία βρίσκονται συγκεντρωμένα και ενοποιημένα σε μία γεωγραφική περιοχή ,με χαρακτηριστικό παράδειγμα να αποτελεί το δίκτυο μίας πανεπιστημιούπολης. Συνήθως οι χρήστες συνδέονται είτε ενσύρματα είτε ασύρματα , ενώ μπορούν να χρησιμοποιήσουν και δικό τους εξοπλισμό. Στηρίζετε στη φιλοσοφία «Φέρε την δική σου συσκευή»(BYOD) , ενώ θα πρέπει να παρέχει διαφορετικά δικαιώματα και έλεγχο πρόσβασης καθώς και τοίχος προστασίας για τους τελικούς χρήστες. Για παράδειγμα ανάλογα τις αρμοδιότητες , οι εργαζόμενοι θα έχουν πρόσβαση σε διαφορετικές υπηρεσίες με την ανάλογη ποιότητα και εύρος ζώνης.

Το φαινόμενο BYOD[10] έχει αναπτυχθεί ταυτόχρονα με την εξάπλωση των φορητών συσκευών. Οι χρήστες μπορούν εύκολα πια να συνδεθούν σε ένα δίκτυο , συνήθως μέσω εξουσιοδότησης που στηρίζεται είτε στον έλεγχο των MAC διευθύνσεων των συσκευών ,είτε στο 802.1X πρωτόκολλο. Παράλληλα, μπορούν χρησιμοποιήσουν την ανακάλυψη υπηρεσιών όπως για παράδειγμα η χρήση ασύρματων εκτυπωτών , η οποία όμως δημιουργεί επιπρόσθετο φόρτο στο δίκτυο. Ο μεγαλύτερος κίνδυνος είναι οι συσκευές να είναι μολυσμένες και να εξαπλώσουν κακόβουλο λογισμικό που θα προσπαθήσει να ανακαλύψει αδυναμίες μέσα στο δίκτυο.

Η SDN τεχνολογία μπορεί εύκολα να βρει εφαρμογή στα συγκεκριμένα δίκτυα , τόσο στην αυθεντικοποίηση των χρηστών όσο και στην έξυπνη εφαρμογή τοίχους προστασίας. Για παράδειγμα μπορεί να χρησιμοποιηθεί η τεχνική «captive portal» όπου οι χρήστες θα παίρνουν IP από έναν DHCP server , θα ενημερώνεται ο controller και θα θεωρεί έγκυρο τον χρήστη και θα προσθέτει ροή που θα επιτρέψει την κίνηση του προς έναν συγκεκριμένο HTTP server. Εφόσον συνδεθεί ο χρήστης με τα στοιχεία του πετυχημένα στον HTTP server , θα προστεθούν οι κατάλληλες ροές που θα επιτρέπουν την πρόσβασή του ανάλογα το επίπεδό του. Παράλληλα , ενημερώνεται ο μεταγωγέας πρόσβασής του , να στέλνει στον controller

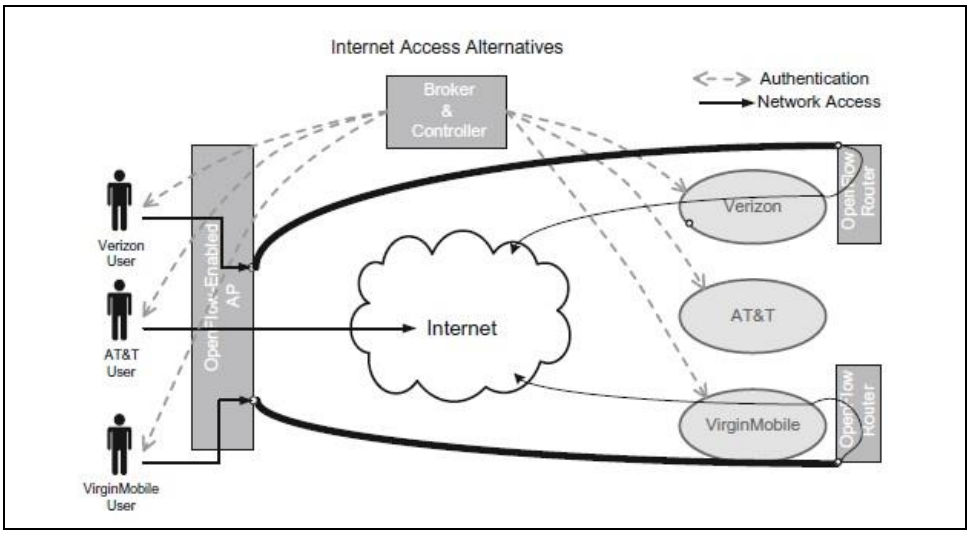
πληροφορίες για την κίνησή του , ώστε σε περίπτωση που εμφανιστεί κακόβουλη κίνηση να διακοπεί.



Εικόνα 11:SDN σε Campus δίκτυα

3.4 Δίκτυα κινητής πρόσβασης

Οι πάροχοι δικτύων κινητής πρόσβασης χρησιμοποιούν τεχνολογίες όπως η LTE , η HSPA και λοιπές κυψελυδωτές για να προσφέρουν φωνή και δεδομένα σε ασύρματες συσκευές. Ο ανταγωνισμός την σημερινή εποχή βασίζεται στην ποιότητα των και τον όγκο των προσφερόμενων δεδομένων φωνής. Οι χρήστες πολλές φορές προτιμούν εφόσον είναι διαθέσιμη την πρόσβαση σε WIFI hotspot , με αποτέλεσμα οι πάροχοι να χάνουν τον έλεγχο των πελατών τους , να υφίσταται ασφάλεια για τους πελάτες αλλά και να μην έχουν όφελος. Ωστόσο η συγκεκριμένη μετάβαση , βοηθά στην αποσυμφόρηση του δικτύου τους. Η SDN τεχνολογία θα μπορούσε να βοηθήσει σε μία υβριδική τεχνολογία[11] όπου WIFI εγκατεστημένα από τους παρόχους θα μπορούσαν να συνδυαστούν με το κυψελωτό δίκτυό τους. Τα WIFI θα μπορούσαν να είναι κοινά για το σύνολο των εταιριών. Με την σύνδεση ενός χρήστη , θα χρησιμοποιηθεί η τεχνολογία captive portal όπως την περιγράψαμε στο προηγούμενο υπό-κεφάλαιο , για να συνδεθεί με τα στοιχεία του ανάλογα την εταιρία που ανήκει. Στην συνέχεια , ο controller θα προσφέρει την απαραίτητη ασφάλεια και ανάλογα την επιλογή του η κίνηση θα προωθηθεί απευθείας στο ίντερνετ ή μέσω του κυψελωτού δικτύου. Αντίστοιχα θα εφαρμόζονται οι απαραίτητες πολιτικές πρόσβασης και οι κανόνες ποιότητας υπηρεσίας



Εικόνα 12:πρόσβαση μέσω WIFI σε δίκτυα κινητής

3.5 Data Center

Τα σημερινά Data Center αποτελούνται από χιλιάδες φυσικούς server και χωρίζονται σε τρεις συγκεκριμένες κατηγορίες:

- **Ιδιωτικά με μοναδικό κάτοχο**: Κατηγορία όπου ανήκουν τα data center που διατηρούν οργανισμοί για προσωπική χρήση .
- **Ιδιωτικά με πολλαπλούς κατόχους**: Στην συγκεκριμένη κατηγορία ανήκουν ειδικά data center που παρέχονται από εταιρίες που ειδικεύονται στην προσφορά υπηρεσιών Τα συγκεκριμένα data center χρησιμοποιούνται από πολλούς πελάτες που έχουν όμως συνάψει συμβόλαιο με την εταιρία παροχής.
- **Δημόσια με πολλαπλούς κατόχους**: αναφέρονται σε γενικής κατηγορίας data center που είναι διαθέσιμα σε όλους. Την σημερινή εποχή είναι προσβάσιμοι μέσω του ίντερνετ , και περιγράφονται με τον όρο υπολογιστικό νέφος(cloud computing) .

Συνδυάζοντας την αυξανόμενη πυκνότητα των server και τις δυνατότητες σε ταχύτητα και εύρος ζώνης των σημερινών δικτύων , ο σκοπός είναι η αποθήκευση όλο και μεγαλύτερο όγκο πληροφοριών σε όλο και μεγαλύτερα data center. Ένας από τους λόγους είναι και η μείωση του κόστους, μέσω της συγχώνευσης μικρότερων data center , σε ένα μεγαλύτερο. Ακόμη , προς αυτή την κατεύθυνση οδηγεί και η εικονοποίηση των φυσικών server , όπου ένας μοναδικός server μπορεί να φιλοξενεί έναν μεγάλο αριθμό από εικονικές μηχανές , Η εικονοποίηση προσφέρει επίσης ένα σύνολο από πλεονεκτήματα όπως μείωση την απαιτήσεων ισχύς , αποτελεσματικότερη χρήση του υλικού και την ικανότητα για άμεση προσθήκη και αφαίρεση υπηρεσιών.

Ωστόσο η εικονοποίηση των data center , δημιουργεί απαιτήσεις και από άποψη δικτύωσης, καθώς θα πρέπει να αντιμετωπιστούν μία σειρά από περιορισμοί των σημερινών δικτύων[12] όπως:

- **Διαθεσιμότητα MAC διευθύνσεων**: Οι μεταγωγείς και οι δρομολογητές βασίζονται για την προώθηση των πακέτων στους πίνακες MAC διευθύνσεων , οι οποίοι όμως υπόκειται σε φυσικούς περιορισμούς . Ωστόσο με την επέκταση του Ethernet πρωτοκόλλου και στα WAN δίκτυα , οι απαιτήσεις έχουν εκτοξευτεί.

- **Αριθμός εικονικών τοπικών δικτύων(VLANs):** ο μέγιστος αριθμός των διαθέσιμων εικονικών δικτύων , βάση του πρωτοκόλλου 802.1q είναι 4096 ό οποίος είναι αρκετά μεγάλος στην περίπτωση data center που χρησιμοποιούνται από έναν οργανισμό ,ωστόσο στις περιπτώσεις πολλαπλών κατόχων , αυτός ο αριθμός μπορεί εύκολα να ξεπεραστεί.

- **Επικαλύπτοντα δέντρα (Spanning Trees):** Το πρωτόκολλο επικαλυπτόντων δέντρων χρησιμοποιείται για να αποφεύγονται πιθανοί βρόχοι , όπου οι συνδέσεις και οι διαδρομές μεταξύ των μεταγωγών επαναυπολογίζονται ύστερα από κάθε αλλαγή. Στο παρελθόν ο συγκεκριμένος χρόνος ήταν ελάχιστος , ωστόσο με την αύξηση του αριθμού των εικονικών συσκευών, η συχνότητα των αλλαγών έχει αυξηθεί με το αποτέλεσμα να μην καλύπτει πια το πρωτόκολλο επικαλυπτόντων δέντρων.

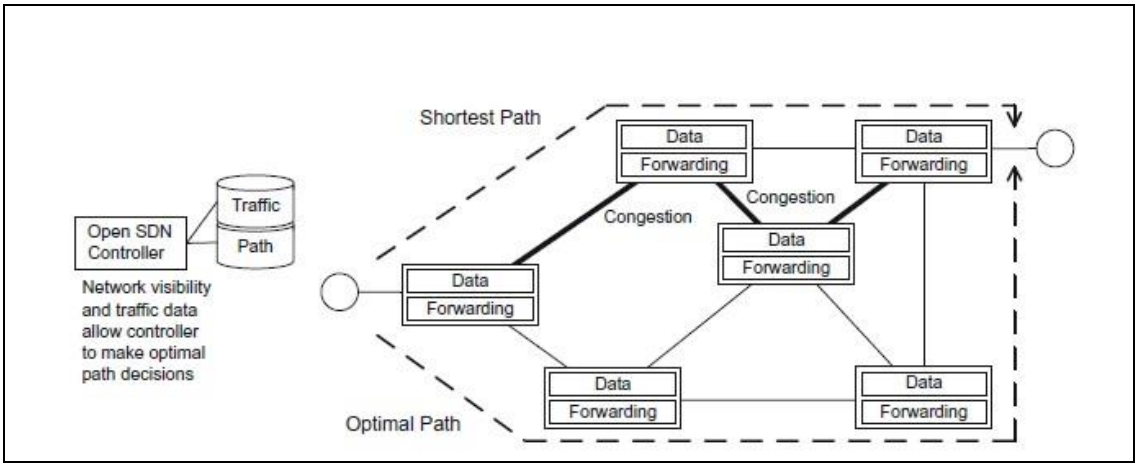
- **Αλλαγή διαθέσιμων πόρων** : Η εικονοποίηση των data center προσφέρουν την δυνατότητα για εύκολη διαχείριση των πόρων των εικονικών συσκευών και αντίστοιχα θα πρέπει και το δίκτυο με το οποίο επικοινωνούν, να προσαρμόζεται εύκολα στις απαιτούμενες αλλαγές.

- **Ανάκαμψη από σφάλμα**: Τα data center , απαιτούν άμεση ανάκαμψη σε περίπτωση σφάλματος , ωστόσο με την λήψη αποφάσεων για την δικτύωση , να λαμβάνεται από τις συσκευές , όπου πρέπει να συνεργαστούν μεταξύ τους , η πιθανή ανάκαμψη σε περίπτωση σφάλματος παρουσιάζει απρόβλεπτη συμπεριφορά.

- **Ταυτόχρονη χρήση πόρων:** Τα data center πολλαπλών κατόχων , χαρακτηρίζονται από την φιλοξενία server και εικονικών συσκευών διαφορετικών ιδιοκτητών στον ίδιο

χώρο και πολλές φορές και στο ίδιο μηχάνημα. Αντίστοιχα θα πρέπει η δικτύωσή του να είναι ικανή να ξεχωρίσει την πρόσβαση και την κίνηση μεταξύ των διαφορετικών συσκευών.

Η SDN τεχνολογία μπορεί να αντιμετωπίσει τις παραπάνω προκλήσεις , σε συνεργασία με την τεχνολογία της εικονοποίησης δικτύων. Πιο συγκεκριμένα , την τεχνική της δημιουργίας MAC-to-IP tunneling[13] όπου τα MAC πακέτα , ενθυλακώνονται σε IP πακέτα, με τις εικονικές συσκευές να θεωρούν ότι χρησιμοποιούν ένα φυσικό δίκτυο. Με αυτόν τον τρόπο αντιμετωπίζονται τα προβλήματα με τους MAC πίνακες ,καθώς οι συσκευές χρειάζεται να γνωρίζουν μόνο τις MAC διευθύνσεις των δύο άκρων που αποτελούν το τούνελ. Επιπλέον με την χρήση των τούνελ δεν απαιτείται πια η χρησιμοποίηση των VLANs. Παρόμοια αντιμετωπίζεται και το πρόβλημα διαχωρισμού της κίνησης και της ταυτόχρονης χρήσης του διαθέσιμου εύρους ζώνης , μεταξύ εικονικών συσκευών που ανήκουν στον server. Για τις υπόλοιπες προκλήσεις , τα SDN δίκτυα υπερέχουν λόγω της κεντρικοποιημένης διαχείρισης των συσκευών και της δικτυακής τοπολογίας , όπου μπορούν εύκολα να ανακάμψουν από σφάλματα και διαμοιράσουν το διαθέσιμο εύρος ζώνης μεταξύ των χρηστών.



Εικόνα 13:επιλογή διαδρομής μέσω SDN σε Data Center

3.6 Επίλογος

Στο συγκεκριμένο κεφάλαιο παρουσιάσαμε τις εφαρμογές και τις λύσεις που μπορούν να προσφέρουν τα SDN δίκτυα , σε μία σειρά από τοπολογίες και δίκτυα , με διαφορετικά χαρακτηριστικά και απαιτήσεις , στις οποίες ωστόσο μπορούν να ανταπεξέλθουν αυξάνοντας τις δυνατότητές τους.

4 ΜΕΛΛΟΝΤΙΚΕΣ ΠΡΟΚΛΗΣΕΙΣ ΓΙΑ ΤΑ SDN ΔΙΚΤΥΑ

Η αγορά των SDN δικτύων συνεχίζει να αναπτύσσεται και να εξελίσσεται με ένα μεγάλο εύρος από προϊόντα να είναι διαθέσιμα στην αγορά όπου συνδυάζουν διαφορετικές δυνατότητες και βασίζονται σε διαφορετική φιλοσοφία υλοποίησης. Το μικρό χρονικό διάστημα ανάπτυξης της τεχνολογίας καθώς και οι διαφορετικές αντιλήψεις πάνω στην έννοια και στον τρόπο λειτουργίας της , θέτει σε αμφιβολία την συμβατότητα και την επεκτασιμότητα πολλών από τις αναπτυσσόμενες πλατφόρμες. Οι κατασκευαστές και οι ερευνητές έχουν να αντιμετωπίσουν μία σειρά από προκλήσεις και να λάβουν υπόψιν μία σειρά από παραμέτρους στην εξέλιξη των SDN δικτύων.

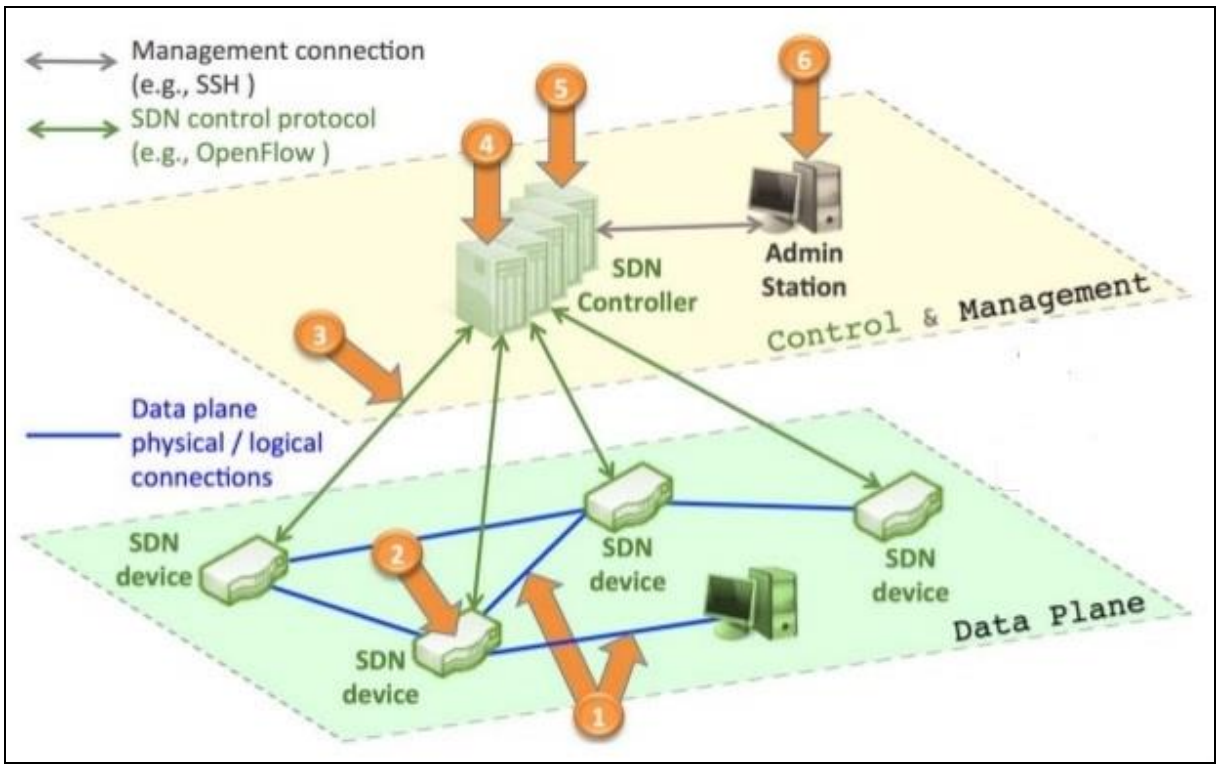
4.1 Ζητήματα Ασφαλείας

Η αντιμετώπιση κινδύνων ασφαλείας παραμένει ένα κομβικό σημείο το οποίο θα πρέπει να διασφαλιστεί για την ορθή και αδιάληπτη λειτουργία ενός δικτύου. Καθώς τα SDN δίκτυα βασίζονται σε μία εντελώς καινούρια φιλοσοφία ως προς την λειτουργιά των παραδοσιακών δικτύων , αναπτύσσονται αντίστοιχα και καινούριοι κίνδυνοι που δεν υφίσταντο πριν[14]. Έχουν χαρακτηριστικά διαγνωστεί οι εξής πιθανοί κίνδυνοι:

- **Λανθασμένες εγγραφές στον εξοπλισμό**: Πιθανές λάθος εγγραφές στους πίνακες του εξοπλισμού που δημιουργήθηκαν από το Controller λόγω κακού σχεδιασμού και προγραμματισμού , τις οποίες μπορεί να εκμεταλλευτεί έναν επιτιθέμενος.
- **Αδυναμίες του εξοπλισμού**: Αδυναμίες του εξοπλισμού , τις οποίες μπορεί να εκμεταλλευτεί ο επιτιθέμενος
- **Επίθεση απόρριψης υπηρεσίας στον Controller**: Με τον controller να αποτελεί ουσιαστικά τον διαχειριστή του δικτύου, δημιουργείται ένα μοναδικό σημείο σφάλματος και στο οποίο θα προσπαθήσει κάποιος να επιτεθεί. Σε περίπτωση επιτυχίας του, θα θέσει το δίκτυο εκτός λειτουργίας.
- **Δυσλειτουργία του Controller**: Καθώς ο controller είναι υπεύθυνος για το data plane του εξοπλισμού , πιθανή παραπλάνησή του μπορεί να οδηγήσει σε λανθασμένη δρομολόγηση των πακέτων ώστε να την εκμεταλλευτεί ο επιτιθέμενος. Ο controller θα πρέπει να ενσωματώνει τα απαραίτητα χαρακτηριστικά ασφαλείας για την αποσόβησή τους.

- **Δυσλειτουργία των εφαρμογών:** Μία δυσλειτουργική εφαρμογή που επικοινωνεί άμεσα με τον Controller μπορεί να οδηγήσει στον επαναπρογραμματισμό ολόκληρου του δικτύου.
- **Επίθεση σε σταθμούς διαχείρισης:** Ο έλεγχος του Controller και στην συνέχεια ολόκληρου του δικτύου είναι δυνατός και από τους διάφορους σταθμούς διαχείρισης του Controller στους οποίους εφόσον καταφέρει κάποιος να αποκτήσει πρόσβαση , διαχειρίζεται τον Controller.

Από τους παραπάνω κινδύνους, οι δύο πρώτοι προκύπτουν και στα παραδοσιακά δίκτυα ενώ οι τέσσερις τελευταίοι μόνο στα SDN καθώς δημιουργούνται από τον διαχωρισμό Control και Data plane.



Εικόνα 14:Κίνδυνοι ασφαλείας σε SDN δίκτυα

4.2 Διαθεσιμότητα και αξιοπιστία

Η διαθεσιμότητα και η αξιοπιστία αποτελούν βασικές αρχές της δικτύωσης. Από τα SDN δίκτυα απαιτείται να παρουσιάσουν τουλάχιστον τον ίδιο βαθμό με τις διαθέσιμες εναλλακτικές τους. Ο διαχωρισμός Control και Data plane στον οποίο βασίζεται η SDN τεχνολογία , θέτει υπό αμφισβήτηση την δυνατότητα του για ελαστικότητα στην περίπτωση σφάλματος μεταξύ controller και εξοπλισμού που μπορεί να οδηγήσει σε ένα δίκτυο χωρίς

“εγκέφαλο”. Η μεταφορά του control plane από τον εξοπλισμό στον Controller, δημιουργεί προκλήσεις όσο αφορά κομβικές λειτουργίες όπως η ανίχνευση σφάλματος συνδέσεων και η γρήγορη αντίδραση και λήψη αποφάσεων. Η αξιοπιστία του δικτύου βασίζεται τόσο στην ανοχή σε σφάλματα του data plane του εξοπλισμού, όσο και στην υψηλή διαθεσιμότητα του Controller.

4.3 Επεκτασιμότητα

Η επεκτασιμότητα των SDN δικτύων αποτελεί μία από τις μεγαλύτερες ανησυχίες. Αποτελεί ένα υπαρκτό πρόβλημα στα παραδοσιακά δίκτυα και θα συνεχίσει να υφίσταται και στα SDN. Η επιπλέον κίνηση που δημιουργείται μεταξύ controller και μεταγωγών , αυξάνει τον φόρτο του δικτύου ενώ παρουσιάζεται και μία επιπρόσθετη καθυστέρηση από την δημιουργία των ροών στους μεταγωγείς κατά την δρομολόγηση των πακέτων. Σε μεγάλης κλίμακας δικτυακές τοπολογίες , οι controllers θα πρέπει να διαχειρίζονται εκατομμύρια ροών ανά δευτερόλεπτο χωρίς να επηρεάζουν την ποιότητα των υπηρεσιών των δικτύων. Οι προσπάθειες επίλυσης των προβλημάτων επεκτασιμότητας επικεντρώνονται σε τρία σημεία:

- **data plane**: θα μπορούσε να διατηρηθεί η δυνατότητα λήψης αποφάσεων στον εξοπλισμό ο οποίος θα συμβουλεύεται τον Controller μόνο για αποφάσεις υψηλότερου επιπέδου.
- **control plane**: ανάπτυξη controller υψηλής απόδοσης βασισμένοι σε βασικές αρχές αρχιτεκτονικής υπολογιστών πχ buffering, παράλληλα συστήματα
- **Υβριδικά συστήματα**: Χρησιμοποιούνται ειδικοί ενδιάμεσοι μεταγωγείς που αναλαμβάνουν την δημιουργία των κανόνων στους υπόλοιπους μεταγωγείς της τοπολογίας , την στιγμή που ο Controller συνεχίζει να δημιουργεί τις ροές που βασίζονται σε εφαρμογές καθώς και την συνολική επίβλεψη του δικτύου.

4.4 Αξιολόγηση απόδοσης

Όπως έχουμε αναφέρει, έχουν αναπτυχθεί αρκετές εφαρμογές τόσο από κατασκευαστές υλικού όσο και λογισμικού. Η περαιτέρω ανάπτυξή τους θα οδηγήσει σε έναν αυξανόμενο αριθμό πειραμάτων και δοκιμών στο εγγύς μέλλον. Η απόδοση και η επεκτασιμότητα των SDN δικτύων βρίσκονται ακόμη σε αμφιβολία , και θα πρέπει να ακολουθήσουν μία σειρά από πειράματα και αναλυτικά μοντέλα που θα δώσουν την δυνατότητα στους σχεδιαστές

δικτύων να προσομοιώσουν την ποιότητα και την απόδοση της εφαρμογής τους πριν την υιοθέτησή της. Παράμετροι που θα πρέπει να ληφθούν υπόψιν είναι η ποιότητα και οι δυνατότητες του εξοπλισμού και η παραμετροποίηση τους, ο αριθμός των Controller που θα πρέπει να χρησιμοποιηθεί ανάλογα το εύρος του μοντέλου, ο τρόπος δημιουργίας των εγγραφών στους πίνακες, καθώς και το μέγεθος των ροών και των πακέτων μεταξύ Controller και μεταγωγών . Όλες οι παραπάνω παράμετροι , ανάλογα και το πεδίο εφαρμογής τους , αλλάζουν δραματικά. Σημαντικότερο ρόλο θα διαδραματίσουν τα αναλυτικά μοντέλα , που απλοϊκά προσφέρουν συμπεράσματα λαμβάνοντας υπόψιν τους διαφορετικούς παράγοντες.

4.5 Ανάπτυξη εφαρμογών – προγραμματισμός δικτύων

Ο προγραμματισμός των SDN δικτύων είναι από τα βασικότερα δομικά στοιχεία του[15]. Ο controller επικοινωνεί μέσω της προς βορρά διεπαφής με εφαρμογές και προγραμματίζει το δίκτυο δυναμικά. Παραδείγματος χάρη , εφαρμογές ελέγχουν την διακύμανση του διαθέσιμου εύρους ζώνης και ενημερώνουν τον Controller να δημιουργήσει τις αντίστοιχες ροές για την αποτελεσματικότερη χρήση του δικτύου. Σημαντικό ρόλο στην υιοθέτηση ενός SDN controller θα διαδραματίσει το πλήθος των προγραμματιστών που απασχολούνται με την ανάπτυξη εφαρμογών για να συνεργαστούν με τον συγκεκριμένο controller καθώς και η υποστήριξή τους. Για να επιλέξει κάποιος την μετάβαση προς την SDN τεχνολογία θα πρέπει να είναι εγγυημένη η μακροχρόνια υποστήριξη και ανάπτυξή τους. Μία από τις μεγαλύτερες προκλήσεις λοιπόν ενός controller, είναι να προκαλέσει το ενδιαφέρον προς μία μεγάλη κοινότητα προγραμματιστών για την περαιτέρω ανάπτυξή του.

4.6 Διαλειτουργικότητα

Είναι αμφίβολο κατά πόσο όσοι επιθυμούν να προχωρήσουν στην υιοθέτηση της SDN τεχνολογίας , είναι πρόθυμοι να προχωρήσουν και στην αλλαγή του δικτυακού εξοπλισμού τους ειδικά εφόσον έχουν προχωρήσει σε μία μεγάλη επένδυση. Είναι σημαντικό για έναν Controller να είναι σε θέση να συνεργάζεται με τον υπάρχον διαθέσιμο εξοπλισμό ανεξαρτήτως κατασκευαστή. Η υιοθέτησή ενός κοινού open-source πρωτοκόλλου όπως είναι το Openflow βοηθάει προς αυτή την κατεύθυνση , ωστόσο οι κατασκευαστές προχωρούν σε δικές τους υλοποιήσεις με διαφορετικές φιλοσοφίες όπου θα πρέπει να λάβουν σοβαρά υπόψιν τους της δυνατότητα συνεργασίας του controller τους και με εξοπλισμό ανταγωνιστών τους.

4.7 Υιοθέτηση της τεχνολογίας

Τα SDN δίκτυα υπόσχονται να προσφέρουν εύκολη σχεδίαση , λειτουργία και διαχείριση υπολογιστικών δικτύων. Μία από τις προκλήσεις τους που θα πρέπει να αντιμετωπίσουν είναι η διστακτικότητα ως προς την υιοθέτησή τους ακριβώς λόγο των αυτοματοποιημένων διαδικασιών που προσφέρουν. Η υιοθέτηση τους δημιουργεί αλλαγές στους ρόλους των εργαζομένων καθώς μετατοπίζεται η ευθύνη και οι υποχρεώσεις ως προς την δικτυακή τοπολογία καθώς οι server διαδραματίζουν μεγάλο ρόλο ,με αποτέλεσμα οι μηχανικοί δικτύων να πρέπει να προσαρμοστούν ή να συνεργαστούν με τους μηχανικούς συστημάτων που διαχειρίζονται τους server. Αντίστοιχα αναβαθμίζεται ο ρόλος και των προγραμματιστών που θα αναπτύσσουν τις δικτυακές εφαρμογές με τις οποίες θα επικοινωνεί ο Controller. Παρόμοια κοινωνικά ζητήματα τίθενται και στα τμήματα πωλήσεων των κατασκευαστών με την ανάπτυξη του ανταγωνισμού μεταξύ τους.

4.8 Επίλογος

Στο συγκεκριμένο κεφάλαιο αναφερθήκαμε στις προκλήσεις που έχει να αντιμετωπίσει η SDN τεχνολογία στο μέλλον. Παρότι οι βασικές αρχές που πρέπει να καλύψουν επιτυχώς παραμένουν παρόμοιες με των κλασσικών δικτύων , η αλλαγή φιλοσοφίας έχει δημιουργήσει νέους κινδύνους ενώ και στο μέλλον είναι προφανές ότι θα προκύψουν και καινούριοι που θα πρέπει να αντιμετωπιστούν. Η διάσπαση των βασικών δομικών στοιχείων μίας δικτυακής τεχνολογίας και η απαραίτητη αγαστή συνεργασία μεταξύ τους , είναι η σημαντικότερη λεπτομέρεια στην οποία θα πρέπει να επικεντρωθούν στο μέλλον οι ερευνητές.

5 ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΣΤΟΝ POX CONTROLLER

5.1 Εισαγωγή

Στο συγκεκριμένο κεφάλαιο θα δείξουμε πως μπορούμε εύκολα να προσθέσουμε μία δυνατότητα σε ένα controller. Στην συγκεκριμένη περίπτωση θα χρησιμοποιήσουμε τον POX controller όπου θα του προσθέσουμε την δυνατότητα του Dynamic Arp Inspection(DAI)[16] για την πρόληψη και αντιμετώπιση Address Resolution Protocol(Arp) επιθέσεων .

5.1.1 Dynamic Arp Inspection

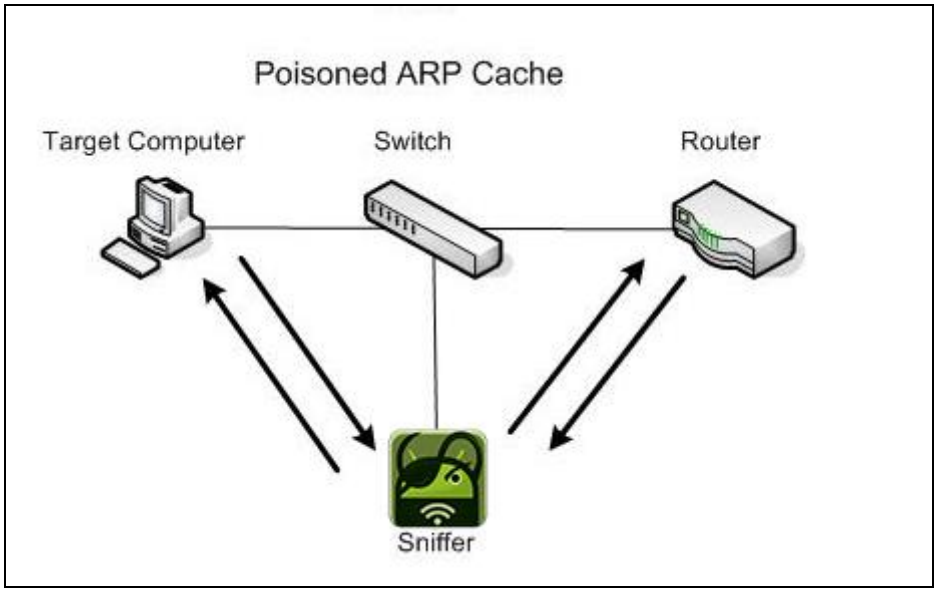
Το DAI είναι μία λειτουργία ασφαλείας που διαθέτουν προηγμένοι εξοπλισμοί επιπέδου 2 και η οποία επικυρώνει την ορθότητα των ARP πακέτων, ενώ επεμβαίνει και απορρίπτει τα άκυρα πακέτα. Για την επικύρωσή τους χρησιμοποιείται μία βάση που περιέχει αντιστοιχίσεις MAC διευθύνσεων σε IP διευθύνσεις. Η συγκεκριμένη βάση δημιουργείται από την δυνατότητα dhcp snooping[17] εφόσον είναι ενεργοποιημένη στον εξοπλισμό, όπου αποθηκεύονται οι διευθύνσεις που έχουν αποδοθεί από έναν dhcp server. Παράλληλα μπορεί να χρησιμοποιηθεί και μία ARP static access list που θα αντιστοιχίζει στατικές IP διευθύνσεις με τις MAC διευθύνσεις των host.

Στόχος του DAI είναι να αντιμετωπίσει επιθέσεις «ενδιάμεσου ανθρώπου» (man-in-the middle). Οι μεταγωγείς προωθούν τα πακέτα βάση των MAC διευθύνσεων που έχουν αποθηκεύσει στον πίνακα Arp. Ο συγκεκριμένος πίνακας δημιουργείται βάση των εισερχόμενων arp πακέτων όπου ελέγχεται η IP και η MAC διεύθυνση του εισερχόμενου πακέτου και αποθηκεύεται η αντιστοιχία τους. Με αυτόν τον τρόπο σε περίπτωση που χρειάζεται να προωθηθεί ένα πακέτο σε μία συγκεκριμένη IP , συμβουλεύεται τον συγκεκριμένο πίνακα ώστε να το προωθήσει στην σωστή συσκευή.

Στις περιπτώσεις επιθέσεων «ενδιάμεσου ανθρώπου» , μία συσκευή αποστέλλει arp πακέτα όπου αναφέρεται ψευδώς η πραγματική MAC διεύθυνσή της με την IP διεύθυνση μία άλλης συσκευής στην οποίας την κίνηση επιθυμεί να παρέμβει. Αποτέλεσμα η κίνηση προς την IP διεύθυνση να αποστέλλεται στον επιτιθέμενο και όχι στον πραγματικό του παραλήπτη.

Αντίστοιχα , η συγκεκριμένη επίθεση μπορεί να έχει ως σκοπό όχι την υποκλοπή της επικοινωνίας από τον επιτιθέμενο ,αλλά την απόρριψη της υπηρεσίας (Denial of Service) με

μεγάλο αριθμό πακέτων να αποστέλλεται από τον επιτιθέμενο ώστε να μην μπορεί να τα διαχειριστεί ο δικτυακός εξοπλισμός και να σταματήσει την λειτουργία του. Για την αντιμετώπισή του το DAI προβλέπει συγκεκριμένο ρυθμό εισερχόμενων πακέτων



Εικόνα 15:Επίθεση παρεμβολής

5.1.2 Περιβάλλον ανάπτυξης προγράμματος

Το συγκεκριμένο πρόγραμμα αναπτύχθηκε στον προσωπικό μου υπολογιστή με Windows 10 και χρησιμοποιήθηκαν τα εξής προγράμματα:

- **Virtual box**: Λογισμικό της Oracle για την ανάπτυξη και διαχείριση εικονικών συσκευών
- **Notepad++:** Editor που υποστηρίζει την ανάπτυξη κώδικα πολλών γλωσσών προγραμματισμού
- **Putty:** client που επιτρέπει την διασύνδεση μέσω πρωτοκόλλων επικοινωνίας όπως το ssh και το telnet
- **Xming:** server προβολής X11
- **Mininet:** Ένας δικτυακός προσομοιωτής[18] για την ανάπτυξη δικτυακών τοπολογιών ο οποίος έχει προ εγκατεστημένο τον POX controller. Μία από τις διαθέσιμες εφαρμογές του είναι και το πρόγραμμα miniedit που προσφέρει την δυνατότητα διαχείρισης της τοπολογίας σε γραφικό περιβάλλον

- **Winscp:** ftp client για την μεταφορά αρχείων μεταξύ windows και linux συστημάτων και στην προκειμένη περίπτωση του προγράμματος που αναπτύχθηκε στο Notepad++ στο Mininet.
- **Ostinato:** πρόγραμμα προσομοίωσης δικτυακής κίνησης που προστέθηκε στο Mininet για την προσομοίωση arp επιθέσεων

5.2 Εφαρμογή Dynamic Arp Inspection

Η συγκεκριμένη εφαρμογή αναπτύχθηκε στην γλώσσα προγραμματισμού python στην οποία είναι βασισμένος και ο POX controller. Αναπτύχθηκε βάση των αρχών του DAI όπως παρουσιάστηκαν παραπάνω για την αντιμετώπιση arp επιθέσεων.

5.2.1 Αλγόριθμος εφαρμογής

Για την ορθή λειτουργία της εφαρμογής απαιτείται ο Controller να λειτουργεί ως ο DHCP server του δικτύου και να διαχειρίζεται τις αιτήσεις για IP διευθύνσεις από τις συσκευές. Με την εκκίνηση του controller και την σύνδεση των μεταγωγών που υποστηρίζουν το openflow πρωτόκολλο , αποστέλλονται οι πρώτες εγγραφές στους Openflow πίνακές τους βάση των οποίων θα πρέπει όλα τα dhcp και arp πακέτα να προωθούνται στο controller προς διαχείριση.

Η εφαρμογή μας ελέγχει και διαχειρίζεται όλα τα λαμβανόμενα πακέτα στον controller. Σε περίπτωση που είναι dhcp πακέτα σταματάει αυτόματα την διαχείρισή τους ώστε να τα διαχειριστεί η dhcp εφαρμογή του controller. Σε περίπτωση που η dhcp εφαρμογή διαθέσει IP διεύθυνση σε κάποια συσκευή η εφαρμογή μας ενημερώνεται, και αντίστοιχα ενημερώνει δύο πίνακες που αντιστοιχούν διευθύνσεις IP σε MAC και σε MAC σε IP. Στους αντίστοιχους πίνακες έχουν ήδη προστεθεί στατικές εγγραφές που είναι διαθέσιμες σε ένα αρχείο που προσομοιώνει μία στατική λίστα πρόσβασης.

Στην περίπτωση εισερχόμενου arp πακέτου ελέγχεται η MAC διεύθυνση του αποστολέα και σε περίπτωση που δεν βρίσκεται στην λίστα με τις επιτρεπόμενες ,απορρίπτεται η αίτησή του και παράλληλα ενημερώνεται ο μεταγωγέας μέσω Openflow να απορρίπτει την κίνηση από την συγκεκριμένη πόρτα που είναι συνδεδεμένος ο επιτιθέμενος για συγκεκριμένο χρονικό διάστημα. Σε αντίθετη περίπτωση ,εφόσον το πακέτο είναι ARP request , απαντάει δημιουργώντας ένα πακέτο arp reply το οποίο και αποστέλλει στον αρχικό αποστολέα.

Ωστόσο, εφόσον το arp request αφορά IP παραλήπτη(destination ip) που δεν βρίσκεται στον πίνακα IptoMac το απορρίπτει. Εδώ να αναφέρουμε ότι η προσομοίωση αναφέρεται σε δίκτυα που επιτρέπουν μόνο εσωτερική επικοινωνία και όχι εξωτερική δρομολόγηση. Παράλληλα, για κάθε πόρτα καθενός μεταγωγέα διατηρείται ένας πίνακας με τον αριθμό των εισερχόμενων arp πακέτων. Σε περίπτωση που ξεπεράσει τον ρυθμό που έχουμε ορίσει , τα συγκεκριμένα πακέτα απορρίπτονται και ο μεταγωγέας ενημερώνεται να απορρίπτει την κίνηση από την συγκεκριμένη πόρτα για συγκεκριμένο χρονικό διάστημα.Ο συγκεκριμένος πίνακας μηδενίζει κάθε πέντε δευτερόλεπτα.

Τέλος, για κάθε εισερχόμενο πακέτο διατηρείται ένας πίνακας που αναφέρει την πόρτα όπου είναι συνδεδεμένος ο συγκεκριμένος αποστολέας. Με αυτόν τον τρόπο, οποιοδήποτε άλλο πακέτο(εκτός dhcp και arp) προωθείται τον παραλήπτη εφόσον γνωρίζουμε την πόρτα που είναι συνδεδεμένος και ενημερώνεται ο μεταγωγέας ώστε από εδώ και στο εξής να προωθεί στον συγκεκριμένο παραλήπτη την κίνηση με την διεύθυνση του ως διεύθυνση προορισμού. Σε αντίθετη περίπτωση ενημερώνεται ο μεταγωγέας να αποστείλει το πακέτο σε όλες τις πόρτες εκτός από την πόρτα που έλαβε το πακέτο (flooding).

5.2.2 Ανάλυση κώδικα

Στο συγκεκριμένο μέρος θα παρουσιάσουμε αναλυτικά τον κώδικα της εφαρμογής μας.

Αρχικά προσθέτουμε τις βιβλιοθήκες που είναι διαθέσιμες στον pox controller και απαραίτητες για το πρόγραμμά μας.

```
# Import necessary pox features
from pox.core import core
import pox.openflow.libopenflow_01 as of
import pox.lib.packet as pkt
from pox.lib.addresses import IPAddr,EthAddr,parse_cidr
from pox.lib.revent import *
from pox.lib.packet.ethernet import ethernet
from pox.lib.packet.ipv4 import ipv4
from pox.lib.packet.arp import arp
```

Καλώντας το πρόγραμμά μας προς εκτέλεση , εκτελείται η συναρτηση lauch() κατά την οποία διαβάζεται το αρχείο access_list.txt που περιέχει τις στατικές μας IP, και προστίθενται οι εγγραφές του στους κατάλληλους πίνακες , ενώ παράλληλα εκτελούνται τρείς listeners , ένας που ακούσει σε event από την εφαρμογή proto.dhcp όταν αναθέτει ip διευθύνσεις , ένας που ακούει σε εισερχόμενα πακέτα και ένας που ακούει σε συνδέσεις μεταγωγών. Εκτελούνται οι αντίστοιχες συναρτήσεις και για τους τρείς. Τέλος καλείται η συνάρτηση Timer η οποία ανά πέντε δευτερόλεπτα μηδενίζει τον πίνακα με τον αριθμό των arp πακέτων για όλες τις πόρτες , όλων των συνδεδεμένων μεταγωγών.

```
def launch ():
 from pox.lib.recoco import Timer

 # open file access_list.txt and add its entries to the
 # appropriate tables

 with open('/home/mininet/pox/ext/access_list.txt') as f:
  my_lines = [line.rstrip('\n') for line in f]
  log.info("lines %s" % (my_lines))
 for x in my_lines:
  a,b= x.split("-")
  macToIp[EthAddr(a)]=IPAddr(b)
  IpTomac[IPAddr(b)]=EthAddr(a)
  log.info("MAC ADDRESS %s has been leased to IP ADDRESS %s" % (a,macToIp[EthAddr(a)]))
 close('/home/mininet/pox/ext/access_list.txt')

  #add listener to listen events from proto.dhcp , PacketIn and ConnectionUp

 core.openflow.addListenerByName("PacketIn", _handle_PacketIn)
core.openflow.addListenerByName("ConnectionUp", _handle_ConnectionUp)
 core.DHCPD.addListenerByName('DHCPLease', dhcp_blinding)
 log.info("Dynamic arp inspection controller is  running.")

 # call function Timer to delete entries at port table.

 Timer(5, _timer_func, recurring=True)
```

Η συνάρτηση timer_func , μηδενίζει τις εγγραφές του πίνακα port που αναφέρεται στις πόρτες κάθε μεταγωγέα και περιέχει των αριθμό των πακέτων arp που έλαβε ο controller τα τελευταία πέντε δευτερόλεπτα . Αντίστοιχα η συνάρτηση _handle_ConnectionUp ,με την σύνδεση ενός μεταγωγέα στον Controller , προσθέτει μία εγγραφή στον Openflow πίνακά του , κατά την οποία προωθεί τα arp πακέτα στον Controller προς έλεγχο.

```
# A timer that resets port counters every 5 seconds

def _timer_func ():
 for n in port.keys():
  port[n]=0

# when a new switch is connected , inform it to send arp packets to the controller

def _handle_ConnectionUp (self, event):
  if self._install_flow:
   fm = of.ofp_flow_mod()
   fm.priority = 0x7000 # Pretty high
   fm.match.dl_type = ethernet.ARP_TYPE
   fm.actions.append(of.ofp_action_output(port=of.OFPP_CONTROLLER))
   event.connection.send(fm)
```

Σε περίπτωση ανάθεσης μίας IP από τον dhcp server σε έναν host, καλείται η συνάρτηση dhcp_blinding όπου η ανατιθέμενη IP και η MAC διεύθυνση του host στον οποίο ανατέθηκε , προστίθενται στους πίνακες για να προσομοιώσουμε την λειτουργία dhcp blinding. Η συνάρτηση drop καλείται όταν επιθυμούμε να απορρίψουμε την κίνηση από μία συγκεκριμένη πόρτα, και προσθέτουμε μία εγγραφή στον Openflow πίνακα του μεταγωγέα για τον σκοπό αυτόν.

```
# A dhcp bling tables that keeps records of dhcp leases

def dhcp_blinding (event):
    mac=event.host_mac
    ip=event.ip
    macToIp[mac]=ip
    IpTomac[ip]=mac
    log.info("MAC ADDRESS %s has been leased to IP ADDRESS %s" % (mac, ip))

def drop (event , packet):
    """
    Drops this packet and installs a flow to continue
    dropping similar ones for a while
    """
    msg = of.ofp_flow_mod()
    msg.match = of.ofp_match.from_packet(packet)
    msg.idle_timeout = 10
    msg.hard_timeout = 10
    msg.actions = []
    event.connection.send(msg)
    log.debug("DROP IT")
```

Η συνάρτηση _handle_PacketIn είναι και η σημαντικότερη του προγράμματός μας καθώς διαχειρίζεται τα εισερχόμενα πακέτα στον Controller. Αρχικά ,ελέγχει αν πρόκειται για dhcp πακέτα. Αν αυτό είναι αληθές σταματάει την εκτέλεσή της καθώς τα dhcp πακέτα θα πρέπει να διαχειριστούν από την εφαρμογή proto.dhcpd. Σε περίπτωση που δεν πρόκειται για dhcp πακέτο , συνεχίζει την εκτέλεσή της όπου ενημερώνει τον πίνακα που αντιστοιχίζει MAC διευθύνσεις με πόρτες ώστε να γνωρίζει που βρίσκεται ο host με την συγκεκριμένη MAC διεύθυνση. Ακόμη , σε περίπτωση που δεν είναι πρώτη φόρα που λαμβάνουμε πακέτο από την συγκεκριμένη πόρτα, προσθέτουμε ένα πεδίο στον πίνακα port με εγγραφή μηδέν.

```
def _handle_PacketIn (event):

 # check if it is a dhcp packet

 p = event.parsed.find('dhcp')
 if p:
    log.debug("DHCP NOT FOR ME")

    # if it is a dhcp packet , stop handling it

    event.halt=True
 else:
  packet = event.parsed
  dpid = event.dpid
  inport = event.port

  # Learn the source

  table[(event.connection,packet.src)] = event.port

  # check if we have know destination port

  dst_port = table.get((event.connection,packet.dst))

  # keep counting of arp packets

  num = port.get((dpid,inport))

  # if we have not yet an entry for this pair of switch and port

  if num is None:

  # add a new entry with zero as value

    port[(dpid,inport)]=0
```

Στην συνέχεια , σε περίπτωση που πρόκειται για arp πακέτο, αυξάνει τον μετρητή στο αντίστοιχο πεδίο του πίνακα port. Ελέγχει τον ρυθμό των εισερχόμενων πακέτων από την συγκεκριμένη πόρτα. Εφόσον είναι μεγαλύτερος από συγκεκριμένο όριο , καλείται η συνάρτηση drop(). Ακολούθως ελέγχει η MAC διεύθυνση του εισερχόμενου πακέτου. Αν ανήκει στις επιτρεπόμενες , συνεχίζει η εκτέλεση του προγράμματος , διαφορετικά καλείται πάλι η συνάρτηση drop(). Εφόσον περάσει ένα πακέτο πετυχημένα τον συγκεκριμένο έλεγχο, ελέγχεται η IP διεύθυνση προορισμού, και σε περίπτωση που δεν ανήκει στο δίκτυό μας το πακέτο απορρίπτεται.

```
# check if it an arp packet

  if packet.type == packet.ARP_TYPE:
       port[(dpid,inport)]=port[(dpid,inport)] + 1
       log.info("arp packets from %s switch %s port are %s" % (dpid,
inport,port[(dpid,inport)]))

       # if the rate incoming of arp packets is largest than limit ,install a flow to drop them

       if port[(dpid,inport)]>100:
         drop(event , packet)
         return
       log.debug("YES IS ARP_TYPE")

       #drop packets if packet source mac address not in macToIp table

       if packet.src not in macToIp:
        log.info("packet source is %s " % packet.src)
        log.debug("NOT IN HERE LETS DROP IT")
        drop(event , packet)
        return

       #drop only the specific packet if ip address not in IpTomac table

       elif IPAddr(packet.payload.protodst) not in IpTomac:
        event.halt=True
```

Σε περίπτωση που πρόκειται για arp request δημιουργούμε ένα πακέτο arp reply και το αποστέλλουμε στον αρχικό αποστολέα.

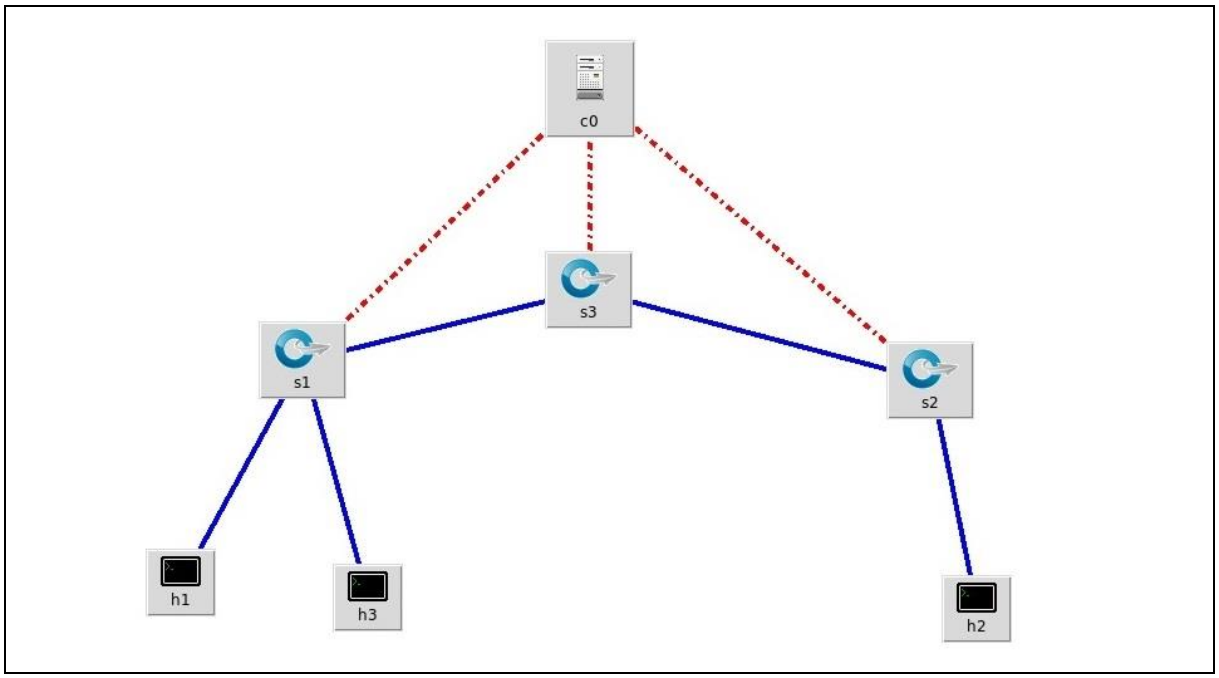
```
# if the packet is a arp request i can answer!!

          if packet.payload.opcode == arp.REQUEST:
            log.debug("I will answer!!")
            arp_reply = arp()
            arp_reply.hwsrc = IpTomac[IPAddr(packet.payload.protodst)]
            arp_reply.hwdst = packet.src
            arp_reply.opcode = arp.REPLY
            arp_reply.protosrc = packet.payload.protodst
            arp_reply.protodst = packet.payload.protosrc
            ether = ethernet()
            ether.type = ethernet.ARP_TYPE
            ether.dst = packet.src
            ether.src = IpTomac[IPAddr(packet.payload.protodst)]
            ether.payload = arp_reply
            msg = of.ofp_packet_out()
            msg.data = ether.pack()
            msg.actions.append(of.ofp_action_output(port =of.OFPP_IN_PORT))
            log.debug("Here is your reply!!")
            msg.in_port = inport
            event.connection.send(msg)
```

Σε περίπτωση που δεν πρόκειται για arp packet, εφόσον γνωρίζουμε που βρίσκεται ο παραλήπτης , δημιουργούμε την κατάλληλη εγγραφή στον Openflow πίνακα του μεταγωγέα ώστε να γνωρίζει από εδώ και πέρα να προωθεί τα πακέτα στον συγκεκριμένο παραλήπτη ενώ προωθούμε και το πακέτο στον παραλήπτη. Αλλιώς ενημερώνουμε τον μεταγωγέα να προωθήσει το πακέτο σε όλες τις πόρτες του εκτός της πόρτας που το έλαβε , ώστε να βρεθεί ο παραλήπτης.

```
else:
   if dst_port is None:

   #we do not know the destination so flood the packet.

    msg = of.ofp_packet_out(data = event.ofp)
    msg.actions.append(of.ofp_action_output(port = of.OFPP_FLOOD))
    log.debug("I do not know this address,flood it")
    event.connection.send(msg)
   else:

   # As we know both source and destination install a flow based to
them

    msg = of.ofp_flow_mod()
    msg.match.dl_dst = packet.src
    msg.match.dl_src = packet.dst
    msg.actions.append(of.ofp_action_output(port = event.port))
    event.connection.send(msg)

   # resend the packet that has came in.

    msg = of.ofp_flow_mod()
    msg.data = event.ofp # Forward the incoming packet
    msg.match.dl_src = packet.src
    msg.match.dl_dst = packet.dst
    msg.actions.append(of.ofp_action_output(port = dst_port))
    event.connection.send(msg)
    log.debug("Installing %s <-> %s" % (packet.src, packet.dst))
```

5.3 Προσομοίωση εφαρμογής

Για να ελέγξουμε την ορθότητα της εφαρμογής θα εκτελέσουμε μία προσομοίωση κατά την οποία δύο hosts θα επικοινωνούν μεταξύ τους ενώ ένας επιτιθέμενος θα προσπαθήσει να υποκλέψει την μεταξύ τους επικοινωνία, καθώς και να θέσει τον controller εκτός λειτουργίας.



Εικόνα 16:τοπολογία εφαρμογής

Στην παραπάνω τοπολογία , οι host h1 και h2 αποτελούν έμπιστους υπολογιστές του δικτύου μας , ενώ ο h3 προσομοιώνει τον επιτιθέμενο.

Για να εκτελέσουμε το πρόγραμμά μας χρησιμοποιούμε την εξής εντολή στο virtual machine μας.

./pox.py info.packet_dump proto.dhcpd --network=10.0.0.0/24 --ip=10.0.0.100 --first=101 dai samples.pretty_log log.level --DEBUG

Η οποία αποτελείται από τα εξής μέρη:

- **pox.py**: καλούμε προς εκτέλεση τον pox controller
- **info.packet_dump**: εφαρμογή του controller η οποία μας εμφανίζει τα logs από τα πακέτα που λαμβάνει
- **proto.dhcpd --network=10.0.0.0/24 --ip=10.0.0.100 --first=101**: καλούμε τον ενσωματωμένο dhcp server[19] του pox controller , ο οποίος θα έχει για ip

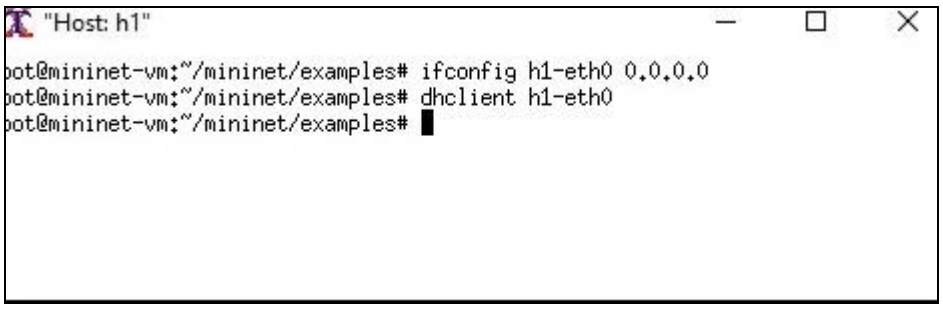
διεύθυνση την 10.0.0.100 και θα διαθέτει διευθύνσεις από το εύρος 10.0.0.0/24 με πρώτη διεύθυνση την 10.0.0.101

- **dai**:η δική μας εφαρμογή
- **samples.pretty_log**: αλλάζει την εμφάνιση των logs ώστε να εμφανίζονται πιο διακριτά.
- **log.level –DEBUG**: θέλουμε να εμφανίζονται τα log επιπέδου DEBUG και άνω.

Με την εκτέλεσή της , ο POX controller ενεργοποιείται , οι διαθέσιμοι μεταγωγείς συνδέονται , η εφαρμογή μας διαβάζει το αρχείο “access_list.txt” , ενώ στέλνονται και οι πρώτες εγγραφές στους πίνακες των μεταγωγών για να ενημερωθούν να προωθούν τα πακέτα arp και dhcp στον controller.

```
INFO:info.packet_dump:Packet dumper running
INFO:dai:lines ['00:00:00:00:00:00-10.0.0.1', '00:00:00:00:00:01-10.0.0.2']
INFO:dai:MAC ADDRESS 00:00:00:00:00:00 has been leased to IP ADDRESS 10.0.0.1
INFO:dai:MAC ADDRESS 00:00:00:00:00:01 has been leased to IP ADDRESS 10.0.0.2
INFO:dai:Dynamic arp inspection controller is  running.
[core                  ] POX 0.2.0 (carp) going up...
[core                  ] Running on CPython (2.7.6/Mar 22 2014 22:59:56)
[core                  ] Platform is Linux-3.13.0-24-generic-x86_64-with-Ubuntu-14.04-trusty
[core                  ] POX 0.2.0 (carp) is up.
[openflow.of_01        ] Listening on 0.0.0.0:6633
[openflow.of_01        ] [00-00-00-00-00-03 1] connected
[openflow.of_01        ] [00-00-00-00-00-01 3] connected
[openflow.of_01        ] [00-00-00-00-00-02 2] connected
```

Στην συνέχεια , στους host h1 και h2 , αναθέτουμε ip μέσω του dhcp server του controller.



```
[dump:00-00-00-00-00-01 ] [ethernet][ipv4][udp][dhcp]
[dai                    ] DHCP NOT FOR ME
[dump:00-00-00-00-00-01 ] [ethernet][ipv4][udp][dhcp]
[dai                    ] MAC ADDRESS 1e:ed:d4:cf:8c:7c has been leased to IP AD
DRESS 10.0.0.102
[proto.dhcpd            ] Leased 10.0.0.102 to 1e:ed:d4:cf:8c:7c
[dai                    ] DHCP NOT FOR ME
[dump:00-00-00-00-00-02 ] [ethernet][ipv4][udp][dhcp]
[dai                    ] DHCP NOT FOR ME
[dump:00-00-00-00-00-02 ] [ethernet][ipv4][udp][dhcp]
[dai                    ] MAC ADDRESS 76:f4:a9:80:f9:ef has been leased to IP AD
DRESS 10.0.0.103
[proto.dhcpd            ] Leased 10.0.0.103 to 76:f4:a9:80:f9:ef
[dai                    ] DHCP NOT FOR ME
```

Η εφαρμογή μας αντιλαμβάνεται ότι πρόκειται για dhcp πακέτα τα οποία πρόκειται να διαχειριστεί ο dhcp server , εμφανίζει το κατάλληλο μήνυμα ενώ ο dhcp server αναθέτει τις διευθύνσεις. Στην συνέχεια ο h2 host προσπαθεί να επικοινωνήσει με τον h1 χρησιμοποιώντας το πρωτόκολλο icmp.Η σύνδεση είναι επιτυχής καθώς η εφαρμογή μας επιτρέπει την επικοινωνία μεταξύ τους , απαντάει στο arp request του h2 και δημιουργεί τις απαραίτητες εγγραφές στους πίνακες των μεταγωγών.

```
oot@mininet-vm:~/mininet/examples# ping 10.0.0.102
ING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
4 bytes from 10.0.0.102: icmp_seq=1 ttl=64 time=108 ms
4 bytes from 10.0.0.102: icmp_seq=2 ttl=64 time=0.526 ms
4 bytes from 10.0.0.102: icmp_seq=3 ttl=64 time=0.109 ms
4 bytes from 10.0.0.102: icmp_seq=4 ttl=64 time=0.109 ms
4 bytes from 10.0.0.102: icmp_seq=5 ttl=64 time=0.104 ms
4 bytes from 10.0.0.102: icmp_seq=6 ttl=64 time=0.103 ms
4 bytes from 10.0.0.102: icmp_seq=7 ttl=64 time=0.114 ms
C
-- 10.0.0.102 ping statistics ---
 packets transmitted, 7 received, 0% packet loss, time 6008ms
tt min/avg/max/mdev = 0.103/15.612/108.224/37.808 ms
```

```
[dump:00-00-00-00-00-02 ] [ethernet][arp]
[dai                    ] arp packets from 2 switch 1 port are 1
[dai                    ] YES IS ARP_TYPE
[dai                    ] I will answer!!
[dai                    ] Here is your reply!!
[dump:00-00-00-00-00-02 ] [ethernet][ipv4][icmp][echo][56 bytes]
[dai                    ] I do not know this address,flood it
[dump:00-00-00-00-00-03 ] [ethernet][ipv4][icmp][echo][56 bytes]
[dai                    ] I do not know this address,flood it
[dump:00-00-00-00-00-01 ] [ethernet][ipv4][icmp][echo][56 bytes]
[dai                    ] I do not know this address,flood it
[dump:00-00-00-00-00-01 ] [ethernet][arp]
[dai                    ] arp packets from 1 switch 2 port are 1
[dai                    ] YES IS ARP_TYPE
[dai                    ] I will answer!!
[dai                    ] Here is your reply!!
[dump:00-00-00-00-00-01 ] [ethernet][ipv4][icmp][echo][56 bytes]
[dai                    ] Installing 1e:ed:d4:cf:8c:7c <-> 76:f4:a9:80:f9:ef
[dump:00-00-00-00-00-03 ] [ethernet][ipv4][icmp][echo][56 bytes]
[dai                    ] Installing 1e:ed:d4:cf:8c:7c <-> 76:f4:a9:80:f9:ef
[dump:00-00-00-00-00-02 ] [ethernet][ipv4][icmp][echo][56 bytes]
[dai                    ] Installing 1e:ed:d4:cf:8c:7c <-> 76:f4:a9:80:f9:ef
```

Αναλυτικά στην παραπάνω εικόνα , βλέπουμε ότι η εφαρμογή μας λαμβάνει ένα arp request από τον host h2, απαντάει στον h2 με την MAC διεύθυνση του h1. Στην συνέχεια , ο h2 αποστέλλει icmp πακέτα, ο πρώτος μεταγωγέας δεν γνωρίζει την διεύθυνση του h1 , οπότε στέλνει τα πακέτα στον controller. Ο controller δεν έχει λάβει πακέτα ποτέ από τον h1 άρα δεν γνωρίζει την θέση στο δίκτυο. Ενημερώνει τον μεταγωγέα να στείλει το πακέτο σε όλες τις πόρτες εκτός από την πόρτα που το έλαβε. Αντίστοιχα οι δύο άλλοι μεταγωγείς ακολουθούν την ίδια διαδικασία. Τελικά εγκαθιδρύεται η επικοινωνία μεταξύ h1 και h2 όπως και οι κατάλληλες εγγραφές στους Openflow πίνακες των μεταγωγών.

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=44.99s, table=0, n_packets=4, n_bytes=392, idle_age=42, dl_src=1e:ed:d4:cf:8c:7c,dl_dst=76:f4:a9:80:f9:ef actions=output:3
 cookie=0x0, duration=45.029s, table=0, n_packets=3, n_bytes=294, idle_age=42, dl_src=76:f4:a9:80:f9:ef,dl_dst=1e:ed:d4:cf:8c:7c actions=output:2
 cookie=0x0, duration=429.808s, table=0, n_packets=2, n_bytes=684, idle_age=120, udp,tp_src=68,tp_dst=67 actions=CONTROLLER:65535
 cookie=0x0, duration=429.77s, table=0, n_packets=1, n_bytes=42, idle_age=45, priority=28672,arp actions=CONTROLLER:65535
*** s3 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=44.964s, table=0, n_packets=4, n_bytes=392, idle_age=42, dl_src=1e:ed:d4:cf:8c:7c,dl_dst=76:f4:a9:80:f9:ef actions=output:2
 cookie=0x0, duration=44.965s, table=0, n_packets=3, n_bytes=294, idle_age=42, dl_src=76:f4:a9:80:f9:ef,dl_dst=1e:ed:d4:cf:8c:7c actions=output:1
 cookie=0x0, duration=429.819s, table=0, n_packets=0, n_bytes=0, idle_age=429, udp,tp_src=68,tp_dst=67 actions=CONTROLLER:65535
 cookie=0x0, duration=429.781s, table=0, n_packets=0, n_bytes=0, idle_age=429, priority=28672,arp actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=44.963s, table=0, n_packets=4, n_bytes=392, idle_age=42, dl_src=1e:ed:d4:cf:8c:7c,dl_dst=76:f4:a9:80:f9:ef actions=output:1
 cookie=0x0, duration=44.964s, table=0, n_packets=3, n_bytes=294, idle_age=42, dl_src=76:f4:a9:80:f9:ef,dl_dst=1e:ed:d4:cf:8c:7c actions=output:2
 cookie=0x0, duration=429.819s, table=0, n_packets=2, n_bytes=684, idle_age=105, udp,tp_src=68,tp_dst=67 actions=CONTROLLER:65535
 cookie=0x0, duration=429.781s, table=0, n_packets=1, n_bytes=42, idle_age=45, priority=28672,arp actions=CONTROLLER:65535
mininet>
```

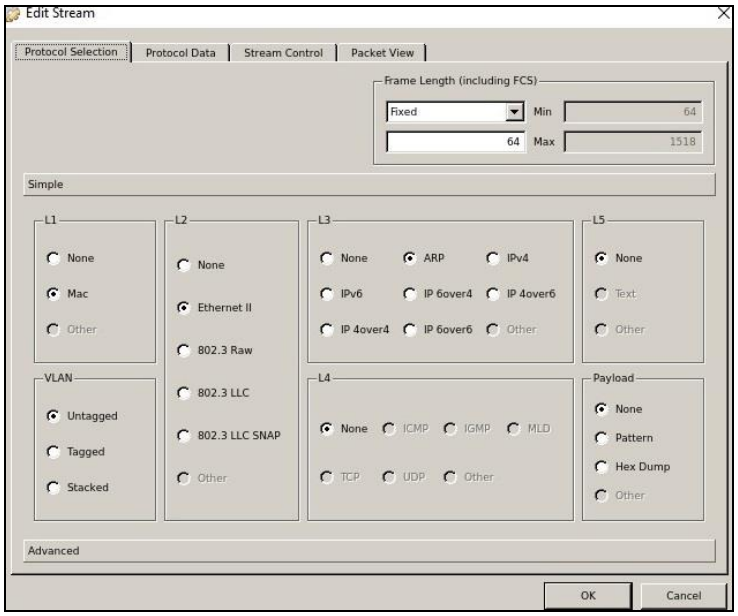
Στην συνέχεια , θα προσομοιώσουμε δύο επιθέσεις από τον host h3. Αρχικά , θα δούμε ότι δεν μπορεί να συνδεθεί με κανέναν host εφόσον χρησιμοποιεί την πραγματική του MAC διεύθυνση. Η εφαρμογή μας αντιλαμβάνεται ότι δεν πρόκειται για μία “έμπιστη διεύθυνση” και απορρίπτει τα πακέτα του. Στην προσπάθειά του να αποστείλει icmp πακέτα , τα arp request του απορρίπτονται καθώς δεν ανήκει στον πίνακα με τις “έμπιστες διευθύνσεις”

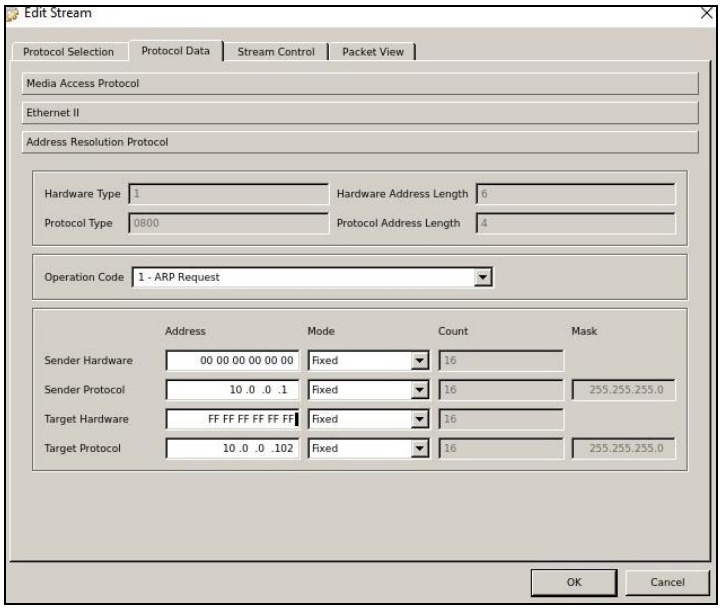
```
[dump:00-00-00-00-00-01 ] [ethernet][arp]
[dai                    ] arp packets from 1 switch 1 port are 1
[dai                    ] YES IS ARP_TYPE
[dai                    ] packet source is 16:cb:3e:3b:e7:94
[dai                    ] NOT IN HERE LETS DROP IT
[dai                    ] DROP IT
```

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=418.114s, table=0, n_packets=4, n_bytes=392, idle_age=415, dl_src=1e:ed:d4:cf:8c:7c,dl_dst=76:f4:a9:80:f9:ef actions=output:3
 cookie=0x0, duration=418.153s, table=0, n_packets=3, n_bytes=294, idle_age=415, dl_src=76:f4:a9:80:f9:ef,dl_dst=1e:ed:d4:cf:8c:7c actions=output:2
 cookie=0x0, duration=802.932s, table=0, n_packets=2, n_bytes=684, idle_age=493, udp,tp_src=68,tp_dst=67 actions=CONTROLLER:65535
 cookie=0x0, duration=802.894s, table=0, n_packets=571, n_bytes=34044, idle_age=2, priority=28672,arp actions=CONTROLLER:65535
 cookie=0x0, duration=2.532s, table=0, n_packets=1, n_bytes=42, idle_timeout=10, hard_timeout=10, idle_age=1, arp,vlan_tci=0x0000,dl_src=16:cb:3e:3b:e7:94,dl_dst=ff:ff:ff:ff:ff:ff,arp_spa=10.0.0.3,arp_tpa=10.0.0.103,arp_op=1 actions=drop
*** s3 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=418.088s, table=0, n_packets=4, n_bytes=392, idle_age=415, dl_src=1e:ed:d4:cf:8c:7c,dl_dst=76:f4:a9:80:f9:ef actions=output:2
 cookie=0x0, duration=418.089s, table=0, n_packets=3, n_bytes=294, idle_age=415, dl_src=76:f4:a9:80:f9:ef,dl_dst=1e:ed:d4:cf:8c:7c actions=output:1
 cookie=0x0, duration=802.943s, table=0, n_packets=0, n_bytes=0, idle_age=802, udp,tp_src=68,tp_dst=67 actions=CONTROLLER:65535
 cookie=0x0, duration=802.905s, table=0, n_packets=0, n_bytes=0, idle_age=802, priority=28672,arp actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=418.087s, table=0, n_packets=4, n_bytes=392, idle_age=415, dl_src=1e:ed:d4:cf:8c:7c,dl_dst=76:f4:a9:80:f9:ef actions=output:1
 cookie=0x0, duration=418.088s, table=0, n_packets=3, n_bytes=294, idle_age=415, dl_src=76:f4:a9:80:f9:ef,dl_dst=1e:ed:d4:cf:8c:7c actions=output:2
 cookie=0x0, duration=802.943s, table=0, n_packets=2, n_bytes=684, idle_age=478, udp,tp_src=68,tp_dst=67 actions=CONTROLLER:65535
 cookie=0x0, duration=802.905s, table=0, n_packets=1, n_bytes=42, idle_age=418, priority=28672,arp actions=CONTROLLER:65535
mininet>
```

Για να προσομοιώσουμε μία επίθεση απόρριψης της υπηρεσίας θα χρησιμοποιήσουμε την εφαρμογή ostinato στον host h3. Θα την ρυθμίσουμε ώστε ο host να στέλνει συνεχώς arp πακέτα όπου τα arp request θα έχουν ως source MAC και IP το πρώτο ζεύγος που περιέχεται στο αρχείο access_list.txt. Κοινώς , η MAC διεύθυνση θα είναι 00:00:00:00:00:00 και η IP διεύθυνση 10.0.0.1 άρα θα θεωρείται έμπιστη. Παράλληλα το arp πακέτο θα αναζητά την διεύθυνση του h1 δηλαδή την 10.0.0.102.

Οι ρυθμίσεις του ostinato είναι οι εξής:





Διατηρώντας την ροή των πακέτων σε ρυθμό 10 packets/sec , η εφαρμογή μας αναγνωρίζει τα πακέτα ως έμπιστα και απαντάει κατάλληλα στον επιτιθέμενο με την MAC διεύθυνση του host h1.

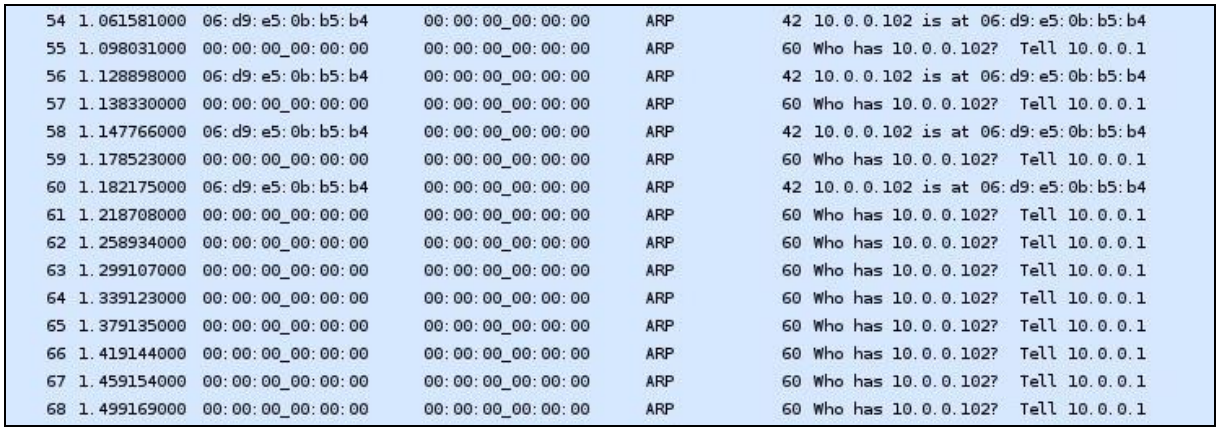
Ωστόσο, εφόσον αυξήσουμε τον ρυθμό των πακέτων πάνω από 20 πακέτα/sec , σε 25 η εφαρμογή μας θα απορρίψει τα πακέτα όταν αυτά περάσουν το όριο των 100 πακέτων ανά πέντε δευτερόλεπτα.
Τα logs του pox controller επιβεβαιώνουν την απόρριψη.

```
[dump:00-00-00-00-00-01 ] [ethernet][arp][18 bytes]
[dai                    ] arp packets from 1 switch 1 port are 97
[dai                    ] YES IS ARP_TYPE
[dai                    ] I will answer!!
[dai                    ] Here is your reply!!
[dump:00-00-00-00-00-01 ] [ethernet][arp][18 bytes]
[dai                    ] arp packets from 1 switch 1 port are 98
[dai                    ] YES IS ARP_TYPE
[dai                    ] I will answer!!
[dai                    ] Here is your reply!!
[dump:00-00-00-00-00-01 ] [ethernet][arp][18 bytes]
[dai                    ] arp packets from 1 switch 1 port are 99
[dai                    ] YES IS ARP_TYPE
[dai                    ] I will answer!!
[dai                    ] Here is your reply!!
[dump:00-00-00-00-00-01 ] [ethernet][arp][18 bytes]
[dai                    ] arp packets from 1 switch 1 port are 100
[dai                    ] YES IS ARP_TYPE
[dai                    ] I will answer!!
[dai                    ] Here is your reply!!
[dump:00-00-00-00-00-01 ] [ethernet][arp][18 bytes]
[dai                    ] arp packets from 1 switch 1 port are 101
[dai                    ] DROP IT
```

Παράλληλα δημιουργείται και η αντίστοιχη εγγραφή στον openflow πίνακα του μεταγωγέα 1.

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=284.37s, table=0, n_packets=4, n_bytes=392, idle_age=281, dl_src=1e:ed:d4:cf:8c:7c,dl_dst=76:f4:a9:80:f9:ef actions=output:3
 cookie=0x0, duration=284.409s, table=0, n_packets=3, n_bytes=294, idle_age=281, dl_src=76:f4:a9:80:f9:ef,dl_dst=1e:ed:d4:cf:8c:7c actions=output:2
 cookie=0x0, duration=669.188s, table=0, n_packets=2, n_bytes=684, idle_age=359, udp,tp_src=68,tp_dst=67 actions=CONTROLLER:65535
 cookie=0x0, duration=669.15s, table=0, n_packets=487, n_bytes=29202, idle_age=2, priority=28672,arp actions=CONTROLLER:65535
 cookie=0x0, duration=2.007s, table=0, n_packets=145, n_bytes=8700, idle_timeout=10, hard_timeout=10, idle_age=0, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00
:00:00:00:00:00,arp_spa=10.0.0.1,arp_tpa=10.0.0.102,arp_op=1 actions=drop
*** s3 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=284.347s, table=0, n_packets=4, n_bytes=392, idle_age=281, dl_src=1e:ed:d4:cf:8c:7c,dl_dst=76:f4:a9:80:f9:ef actions=output:2
 cookie=0x0, duration=284.348s, table=0, n_packets=3, n_bytes=294, idle_age=281, dl_src=76:f4:a9:80:f9:ef,dl_dst=1e:ed:d4:cf:8c:7c actions=output:1
 cookie=0x0, duration=669.202s, table=0, n_packets=0, n_bytes=0, idle_age=669, udp,tp_src=68,tp_dst=67 actions=CONTROLLER:65535
 cookie=0x0, duration=669.164s, table=0, n_packets=0, n_bytes=0, idle_age=669, priority=28672,arp actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=284.345s, table=0, n_packets=4, n_bytes=392, idle_age=281, dl_src=1e:ed:d4:cf:8c:7c,dl_dst=76:f4:a9:80:f9:ef actions=output:1
 cookie=0x0, duration=284.346s, table=0, n_packets=3, n_bytes=294, idle_age=281, dl_src=76:f4:a9:80:f9:ef,dl_dst=1e:ed:d4:cf:8c:7c actions=output:2
 cookie=0x0, duration=669.201s, table=0, n_packets=2, n_bytes=684, idle_age=344, udp,tp_src=68,tp_dst=67 actions=CONTROLLER:65535
 cookie=0x0, duration=669.163s, table=0, n_packets=1, n_bytes=42, idle_age=284, priority=28672,arp actions=CONTROLLER:65535
mininet>
```

Τέλος , επιβεβαιώνεται και από την καταγραφή πακέτων στον host h3 χρησιμοποιώντας το wireshark,όπου παρατηρούμε μέχρι ενός σημείου τα request να λαμβάνουν απαντήσεις , οι οποίες σταματούν όταν ξεπεράσουν το επιτρεπτό όριο, και τα απορρίψει η εφαρμογή μας.



54	1.061581000	06:d9:e5:0b:b5:b4	00:00:00_00:00:00	ARP	42 10.0.0.102 is at 06:d9:e5:0b:b5:b4
55	1.098031000	00:00:00_00:00:00	00:00:00_00:00:00	ARP	60 Who has 10.0.0.102? Tell 10.0.0.1
56	1.128898000	06:d9:e5:0b:b5:b4	00:00:00_00:00:00	ARP	42 10.0.0.102 is at 06:d9:e5:0b:b5:b4
57	1.138330000	00:00:00_00:00:00	00:00:00_00:00:00	ARP	60 Who has 10.0.0.102? Tell 10.0.0.1
58	1.147766000	06:d9:e5:0b:b5:b4	00:00:00_00:00:00	ARP	42 10.0.0.102 is at 06:d9:e5:0b:b5:b4
59	1.178523000	00:00:00_00:00:00	00:00:00_00:00:00	ARP	60 Who has 10.0.0.102? Tell 10.0.0.1
60	1.182175000	06:d9:e5:0b:b5:b4	00:00:00_00:00:00	ARP	42 10.0.0.102 is at 06:d9:e5:0b:b5:b4
61	1.218708000	00:00:00_00:00:00	00:00:00_00:00:00	ARP	60 Who has 10.0.0.102? Tell 10.0.0.1
62	1.258934000	00:00:00_00:00:00	00:00:00_00:00:00	ARP	60 Who has 10.0.0.102? Tell 10.0.0.1
63	1.299107000	00:00:00_00:00:00	00:00:00_00:00:00	ARP	60 Who has 10.0.0.102? Tell 10.0.0.1
64	1.339123000	00:00:00_00:00:00	00:00:00_00:00:00	ARP	60 Who has 10.0.0.102? Tell 10.0.0.1
65	1.379135000	00:00:00_00:00:00	00:00:00_00:00:00	ARP	60 Who has 10.0.0.102? Tell 10.0.0.1
66	1.419144000	00:00:00_00:00:00	00:00:00_00:00:00	ARP	60 Who has 10.0.0.102? Tell 10.0.0.1
67	1.459154000	00:00:00_00:00:00	00:00:00_00:00:00	ARP	60 Who has 10.0.0.102? Tell 10.0.0.1
68	1.499169000	00:00:00_00:00:00	00:00:00_00:00:00	ARP	60 Who has 10.0.0.102? Tell 10.0.0.1

5.4 Επίλογος

Στο συγκεκριμένο κεφάλαιο δημιουργήσαμε μια εφαρμογή την οποία προσθέσαμε ως δυνατότητα του POX controller. Με την συγκεκριμένη εφαρμογή , ο POX controller μπορεί να ελέγχει τα arp πακέτα και να αντιμετωπίζει πιθανές επιθέσεις που σχετίζονται με αυτά. Η εφαρμογή βασίζεται στις αρχές της Dynamic Arp Inspection λειτουργίας. Προσομοιώσαμε πιθανές επιθέσεις και αποδείξαμε ότι ο Controller έχει την δυνατότητα να τις αναγνωρίζει και να τις αντιμετωπίζει. Αρχικός μας σκοπός ήταν να αποδείξουμε πως από πολλούς controller λείπουν βασικές λειτουργίες ασφαλείας αλλά και πως μπορούμε να τις προσθέσουμε εύκολα αναπτύσσοντας τα κατάλληλα προγράμματα.

6 ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην παρούσα εργασία , παρουσιάσαμε συνολικά το πλαίσιο στο οποίο κινούνται τα SDN δίκτυα και αναφερθήκαμε στην φιλοσοφία τους, την δομή τους, τις δυνατότητές τους τις εφαρμογές τους , και την προκλήσεις που έχουν να αντιμετωπίσουν στο μέλλον. Παράλληλα , αναπτύξαμε μία εφαρμογή την οποία προσθέσαμε σε έναν controller επεκτείνοντας τις δυνατότητές του. Αποδείξαμε πως μπορούμε εύκολα , χωρίς παρέμβαση στην δομή του δικτύου και χωρίς πολύπλοκες παραμετροποιήσεις να αυξήσουμε την ασφάλειά του.

Συμπερασματικά , τα SDN δίκτυα αποτελούν τον μέλλον της δικτύωσης. Τα δίκτυα είναι υποχρεωμένα να ακολουθήσουν την εξέλιξη στους υπόλοιπους τομείς της υπολογιστικής επιστήμης και να βρουν τον τρόπο να καλύψουν τις απαιτήσεις. Σε μία εποχή που ανεξαρτητοποιείτε συνεχώς από τον υλικό και δίνει έμφαση στην ανάπτυξη λογισμικού, τα παραδοσιακά δίκτυα φαίνεται να μην μπορούν να ανταπεξέλθουν. Η βασική αιτία είναι οι φυσικοί περιορισμοί αλλά και η φιλοσοφία λειτουργίας τους. Τα SDN δίκτυα δείχνουν ικανά να καλύψουν το κενό.

Τα SDN δίκτυα αποτελούν το νέο «μαύρο» στο τομέα των δικτύων με όλους τους μεγάλους κατασκευαστές και παρόχους να επενδύουν πάνω τους σε προϊόντα που ενσωματώνουν την τεχνολογία.

Ωστόσο, πρόκειται για μία πρώιμη τεχνολογία , η οποία απαιτεί ακόμη εκτεταμένη έρευνα πριν την καθιέρωσή της. Δημιουργεί νέους κίνδυνους , με πολλούς να μην έχουν αναγνωριστεί ακόμη , ενώ αλλάζει τους ρόλους και τις δεξιότητες που απαιτούνται από τους επαγγελματίες του χώρου. Οι μηχανικοί δικτύων θα πρέπει να εξελιχθούν σε προγραμματιστές δικτύων[20] για να μπορούν να διατηρήσουν τον έλεγχο της τοπολογίας τους και να ανταπεξέλθουν στις νέες καταστάσεις.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Software	Λογισμικό
Hardware	Υλικό
Controller	Ελεγκτής
Management	Διαχείριση
Data	Δεδομένα
Control	Έλεγχος
Virtualization	Εικονοποίηση
Routing tables	Πίνακες δρομολόγησης
Tags	Ετικέτες
Spanning tree	Επικαλύπτον δέντρο
Packets	Πακέτα
seconds	Δευτερόλεπτα
Service Provider	Πάροχος υπηρεσιών
Router	Δρομολογητής
Switch	Μεταγωγέας
Firewall	Τοίχος προστασίας
Northbound interfaces	Προς βορρά διεπαφές
Southbound interfaces	Προς Νότο διεπαφές
Headers	Επικεφαλίδες
Load balancers	Εξισορροπητές φορτίου

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

SDN	Software Defined Networks
CLI	Command Line Interface
TCAM	Ternary Content Addressable Memory
FIB	Forwarding information base
API	Application programming interface
CAPEX	Capital Expenditure
OPEX	Operational Expenditure
OSI	Open Systems Interconnection
IP	Internet Protocol
MAC	Media Access Control
VLAN	Virtual Local Area Networks
MPLS	Multiprotocol Label Switching
NFV	Network function virtualization
NAT	Network Address Translation
DNS	Domain Name System
ONF	Open Networking Foundation
TCP	Transmission Control Protocol
TLS	Transport Layer Security
WAN	Wide Area Network
ATM	Asynchronous Transfer Mode
DSL	Digital subscriber line
LSP	Label-Switched Paths
SP	Service Provider
CE	Customer Edge
PE	Provider Edge
BYOD	Bring Your Own Device
ARP	Address Resolution Protocol
DAI	Dynamic Arp Inspection
DHCP	Dynamic Host Configuration Protocol
SSH	Secure Shell
FTP	File Transfer Protocol

ΠΑΡΑΡΤΗΜΑ Ι-ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

```
"""
An SDN simulation of DAI security feature
"""
# Import necessary pox features
from pox.core import core
import pox.openflow.libopenflow_01 as of
import pox.lib.packet as pkt
from pox.lib.addresses import IPAddr,EthAddr,parse_cidr
from pox.lib.revent import *
from pox.lib.packet.ethernet import ethernet
from pox.lib.packet.ipv4 import ipv4
from pox.lib.packet.arp import arp

# Even a simple usage of the logger is much nicer than print!
log = core.getLogger()

# table that maps mac addresses to IPs
macToIp={}

#table that maps mac addresses to IPs
IpTomac={}

# a table that counts arp packets for each switch's port
port={}

# A table maps (switch,MAC-addr) pairs to the port on 'switch'
table = {}

# A timer that resets port counters every 5 seconds
def _timer_func ():
  for n in port.keys():
    port[n]=0
```

```
# when a new switch is connected , inform it to send arp packets to the controller
def _handle_ConnectionUp (self, event):
   if _install_flow:
     fm = of.ofp_flow_mod()
     fm.priority = 0x7000 # Pretty high
     fm.match.dl_type = ethernet.ARP_TYPE
     fm.actions.append(of.ofp_action_output(port=of.OFPP_CONTROLLER))
     event.connection.send(fm)

# A dhcp bling tables that keeps records of dhcp leases
def dhcp_blinding (event):
     mac=event.host_mac
     ip=event.ip
     macToIp[mac]=ip
     IpTomac[ip]=mac
     log.info("MAC ADDRESS %s has been leased to IP ADDRESS %s" % (mac, ip))

def drop (event , packet):
      """
      Drops this packet and installs a flow to continue
      dropping similar ones for a while
      """
      msg = of.ofp_flow_mod()
      msg.match = of.ofp_match.from_packet(packet)
      msg.idle_timeout = 10
      msg.hard_timeout = 10
      msg.actions = []
      event.connection.send(msg)
      log.debug("DROP IT")

def _handle_PacketIn (event):
  # check if it is a dhcp packet
```

```
 p = event.parsed.find('dhcp')


 if p:
    log.debug("DHCP NOT FOR ME")

    # if it is a dhcp packet , stop handling it
    event.halt=True
 else:
  packet = event.parsed
  dpid = event.dpid
  inport = event.port

  # Learn the source
  table[(event.connection,packet.src)] = event.port

  # check if we have know destination port
  dst_port = table.get((event.connection,packet.dst))

  # keep counting of arp packets
  num = port.get((dpid,inport))

  # if we have not yet an entry for this pair of switch and port
  if num is None:

  # add a new entry with zero as value
    port[(dpid,inport)]=0

  # check if it an arp packet
  if packet.type == packet.ARP_TYPE:
       port[(dpid,inport)]=port[(dpid,inport)] + 1
       log.info("arp packets from %s switch %s port are %s" % (dpid,
inport,port[(dpid,inport)]))
```

```
# if the rate incoming of arp packets is largest than limit ,install a flow to drop them
if port[(dpid,inport)]>100:
   drop(event , packet)
   return
log.debug("YES IS ARP_TYPE")

#drop packets if packet source mac address not in macToIp table
if packet.src not in macToIp:
  log.info("packet source is %s " % packet.src)
  log.debug("NOT IN HERE LETS DROP IT")
  drop(event , packet)
  return

#drop only the specific packet if ip address not in IpTomac table
elif IPAddr(packet.payload.protodst) not in IpTomac:
  event.halt=True
else:

# if the packet is a arp request i can answer!!
   if packet.payload.opcode == arp.REQUEST:
     log.debug("I will answer!!")
     arp_reply = arp()
     arp_reply.hwsrc = IpTomac[IPAddr(packet.payload.protodst)]
     arp_reply.hwdst = packet.src
     arp_reply.opcode = arp.REPLY
     arp_reply.protosrc = packet.payload.protodst
     arp_reply.protodst = packet.payload.protosrc
     ether = ethernet()
     ether.type = ethernet.ARP_TYPE
     ether.dst = packet.src
     ether.src = IpTomac[IPAddr(packet.payload.protodst)]
     ether.payload = arp_reply
     msg = of.ofp_packet_out()
     msg.data = ether.pack()
```

```
            msg.actions.append(of.ofp_action_output(port =of.OFPP_IN_PORT))
            log.debug("Here is your reply!!")
            msg.in_port = inport
            event.connection.send(msg)

  else:
   if dst_port is None:

   #we do not know the destination so flood the packet.
    msg = of.ofp_packet_out(data = event.ofp)
    msg.actions.append(of.ofp_action_output(port = of.OFPP_FLOOD))
    log.debug("I do not know this address,flood it")
    event.connection.send(msg)
   else:

   # As we know both source and destination install a flow based to them
    msg = of.ofp_flow_mod()
    msg.match.dl_dst = packet.src
    msg.match.dl_src = packet.dst
    msg.actions.append(of.ofp_action_output(port = event.port))
    event.connection.send(msg)

   # resend the packet that has came in.
    msg = of.ofp_flow_mod()
    msg.data = event.ofp # Forward the incoming packet
    msg.match.dl_src = packet.src
    msg.match.dl_dst = packet.dst
    msg.actions.append(of.ofp_action_output(port = dst_port))
    event.connection.send(msg)
    log.debug("Installing %s <-> %s" % (packet.src, packet.dst))

def launch ():
  from pox.lib.recoco import Timer
  # open file access_list.txt and add its entries to the
```

```
  # appropriate tables
  with open('/home/mininet/pox/ext/access_list.txt') as f:
    my_lines = [line.rstrip('\n') for line in f]
    log.info("lines %s" % (my_lines))
  for x in my_lines:
   a,b= x.split("-")
   macToIp[EthAddr(a)]=IPAddr(b)
   IpTomac[IPAddr(b)]=EthAddr(a)
   log.info("MAC ADDRESS %s has been leased to IP ADDRESS %s" %
(a,macToIp[EthAddr(a)]))

   # add listener to listen events from proto.dhcp and PacketIn
  core.openflow.addListenerByName("PacketIn", _handle_PacketIn)
  core.DHCPD.addListenerByName('DHCPLease', dhcp_blinding)
  log.info("Dynamic arp inspection controller is  running.")

  # call function Timer to delete entries at port table.
  Timer(5, _timer_func, recurring=True)
```

ΑΝΑΦΟΡΕΣ

[1] http://www.noxrepo.org/nox/about-nox/

[2] https://openflow.stanford.edu/display/ONL/POX+Wiki

[3] Koponen, T., M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R.Ramanathan et al. 2010. "Onix: A distributed control platform for largescale production networks". In *Proceedings of the 9th USENIX Conferenceon Operating Systems Design and Implementation (OSDI '10)*. Berkeley: USENIX Association, pp. 1–6

[4] Doug Marschke , Jeff Doyle and Pete Moyer "SDN: Anatomy of Openflow" page 3

[5] Pate P."NFVand SDN: what's the difference?" SDxCentral;March 30, 2013

[6] https://www.opennetworking.org/

[7] OpenFlow switch specification, Version 1.3.0 (wire protocol 0x04). Open Networking Foundation; June 25,2012. Retrieved from <www.opennetworking.org/sdn-resources/onf-specifications>.

[8] Hoelzle U. OpenFlow@Google. Google, Open Networking Summit, Santa Clara, CA, USA; April 2012.

[9] "SDN and NFV: Transforming the Service Provider Organization" Juniper Networks, white paper 2013

[10] "Unleash the full potential of BYOD with confidence" HP white paper , 2013

[11] Jorge Hortlelano , Juan-Carlos Cano,Carlos T .Calafate "RuralNet:A Captive Portal Based System Supporting Wireless Internet in Rural Areas"

[12] Paul Göransson Chuck Black, "Software Defined Networks A Comprehensive Approach" page 185

[13] MahalingamM, Dutt D, Duda K, Agarwal P, Kreeger L, Sridhar T, et al. VXLAN: a framework for overlayingvirtualized layer 2 networks over layer 3 networks. Internet Engineering Task Force; August 26, 2011 [internet draft].

[14] Nagaraj Hedge and Fei Hu " Security Issues in SDN/Openflow"

[15] "The Future of Network Virtualization and SDN Controllers" SDxCentral:2016 Market Report

[16] "Catalyst 6500 Release 12.2SX Software Configuration Guide" Retrieve from cisco web page http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/dynarp.html

[17] "Catalyst 6500 Release 12.2SX Software Configuration Guide" Retrieve from cisco web page http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/snoodhcp.html

[18] http://mininet.org/

[19] "https://openflow.stanford.edu/display/ONL/POX+Wiki#POXWiki-proto.dhcp_client"

[20] "How network admins can survive SDN" article at networkworld retrieved from http://www.networkworld.com/article/2881176/sdn/how-network-admins-can-survive-sdn.html