ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

NETMODE – NETWORK MANAGEMENT & OPTIMAL DESIGN LABORATORY

Σχεδιασμός και ανάπτυξη μηχανισμών παρακολούθησης Ευφυών Προγραμματιζόμενων Δικτύων σε πλατφόρμες πολλαπλών ενοίκων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Κ. Παπαδήμας

Επιβλέπων :Βασίλειος Μάγκλαρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ
NETMODE – NETWORK MANAGEMENT & OPTIMAL DESIGN LABORATORY

Σχεδιασμός και ανάπτυξη μηχανισμών παρακολούθησης Ευφυών Προγραμματιζόμενων Δικτύων σε πλατφόρμες πολλαπλών ενοίκων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Κ. Παπαδήμας

Επιβλέπων :Βασίλειος Μάγκλαρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την.

.....	.....	.....
Βασίλειος Μάγκλαρης Καθηγητής Ε.Μ.Π.	Νεκτάριος Κοζύρης Καθηγητής Ε.Μ.Π.	Ευστάθιος Συκάς Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2017

.....
Δημήτριος Παπαδήμας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Η παρούσα διπλωματική εργασία αποτελεί το τελευταίο στάδιο των προπτυχιακών σπουδών μου στο Εθνικό Μετσόβιο Πολυτεχνείο.

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Βασίλειο Μάγκλαρη για την εμπιστοσύνη που μου έδειξε, την ευκαιρία που μου έδωσε καθώς και για τις πολύτιμες συμβουλές του. Ακόμη, ευχαριστώ όλα τα μέλη του εργαστηρίου NETMODE για την βοήθεια τους, και ιδιαιτέρως τον υπεύθυνο της διπλωματικής μου εργασίας Αδάμ Παυλίδη, για την άριστη συνεργασία μας καθώς και για την πολύτιμη καθοδήγηση του σε όλα τα στάδια εκπόνησης της.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για την ανεκτίμητη υποστήριξη και συμπαράσταση τους, στη διάρκεια των σπουδών μου.

Περίληψη

Σκοπός της διπλωματικής εργασίας είναι η ανάπτυξη μηχανισμών συλλογής δεδομένων δικτυακής κίνησης από ευφυώς προγραμματιζόμενα δίκτυα σε πλατφόρμες πολλαπλών ενοίκων. Δίνεται ιδιαίτερη έμφαση στην προσαρμογή των μηχανισμών σε τέτοιες υποδομές, ξεπερνώντας πιθανόν προβλήματα που δημιουργούν τα επιμέρους χαρακτηριστικά τους. Τα δεδομένα που συλλέγονται είναι χρήσιμα στους διαχειριστές της κάθε υποδομής, καθώς επίσης και στους διαφόρους χρήστες-ενοίκους της. Ως αποτέλεσμα, ιδιαίτερα σημαντική είναι η κατανομή των δεδομένων ανάλογα με τους πόρους που αντιστοιχούν στον κάθε ένοικο.

Πιο συγκεκριμένα, χρησιμοποιήθηκε ένας μηχανισμός παρακολούθησης για αρχιτεκτονικές Δικτύων Οριζόμενων από Λογισμικό (SDN), βασισμένων στο πρωτόκολλο OpenFlow. Ο μηχανισμός αυτός αξιοποιεί την κίνηση σηματοδοσίας που ανταλλάσουν τα επιμέρους δικτυακά στοιχεία μίας υποδομής SDN καθώς και δειγματοληψία κίνησης μέσω του προγράμματος sFlow. Στόχος είναι η προσαρμογή του μηχανισμού σε εικονικά περιβάλλοντα όπου συνυπάρχουν πολλαπλοί ένοικοι, λαμβάνοντας υπόψιν τις ιδιαιτερότητες μίας τέτοιας υποδομής. Απαραίτητο χαρακτηριστικό αποτελεί η ικανότητα του μηχανισμού να ομαδοποιεί τα συλλεγμένα δικτυακά δεδομένα με βάση τον ένοικο που του αναλογούν. Με αυτό τον τρόπο δίνεται η δυνατότητα στον εκάστοτε ενδιαφερόμενο να έχει πρόσβαση στις πληροφορίες που τον αφορούν. Τέλος, μελετάται η συμπεριφορά του μηχανισμού σε πιο “abstract” δομές δικτυακών τοπολογιών, διαμορφωμένες μέσω τεχνικών εικονικοποίησης.

Ο παραπάνω μηχανισμός μπορεί να αποτελέσει βάση για τη δημιουργία ενός ενοποιημένου πλαισίου συλλογής δικτυακών δεδομένων από πειραματικές πλατφόρμες πολλαπλών ενοίκων. Στόχος είναι η αυτόματη παρουσίαση των δεδομένων στους ενδιαφερόμενους χρήστες, καθώς και η αξιοποίηση τους για τη βελτίωση των υπηρεσιών που παρέχει μία τέτοια υποδομή.

Λέξεις Κλειδιά: << Δίκτυα Οριζόμενα από Λογισμικό, Παρακολούθηση Δικτύων, Πλατφόρμες Πολλαπλών Ενοίκων, Εικονική Δικτύωση, OpenFlow, OpenVirteX >>

Abstract

The scope of this thesis is the development of mechanisms for collecting data of network traffic from Software Defined Networks through multitenant platforms. Special emphasis is placed in the adaptiveness of the mechanisms to such infrastructures, overcoming potential problems caused by their special characteristics. The collected data are useful to the administrators of such infrastructures, as well as the users-tenants. As a result, it is particularly important to allocate the data according to the resources assigned to each tenant.

More specifically, the mechanism used is designed to monitor architectures of Software Defined Networks based on the OpenFlow protocol. This mechanism uses traffic signals that are exchanged from the network elements of an SDN infrastructure, as well as traffic sampling through the sFlow program. The aim is to adapt the mechanism to virtual environments shared by multiple tenants, taking into account the specificities of such an infrastructure. An essential feature is the ability of the mechanism to group the collected network data on the basis of their assigned tenant. This enables the users to have access to the information that concern them. Finally, the behavior of the mechanism in more abstract network topologies, modeled by virtualization techniques, is studied.

The above mechanism can serve as a basis for creating a unified network data collection framework from multitenant experiment platforms. The aim is to automatically present the data to the tenants concerned, as well as to use them in order to improve the services provided by such an infrastructure.

Keywords: << Software Defined Networks, Network Monitoring, Multitenant Platforms, Network Virtualization, OpenFlow, OpenVirteX >>

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Αντικείμενο διπλωματικής

Η εξέλιξη των τεχνικών δικτύωσης και η ευρεία καθιέρωση των τεχνολογιών υπολογιστικού νέφους δημιούργησαν την ανάγκη ανάπτυξης πλατφόρμων διαμοιρασμού δικτυακών πόρων σε πολλαπλούς χρήστες-ενοίκους. Τέτοιες πλατφόρμες αποτελούν χρήσιμα εργαλεία σε “cloud” υποδομές, ενώ επιπλέον χρησιμοποιούνται σαν πειραματικές υποδομές (testbeds) για την ανάπτυξη και την αξιολόγηση νέων τεχνικών και τεχνολογιών δικτύωσης.

Στα πλαίσια της παρούσας διπλωματικής εργασίας μελετώνται εργαλεία και μηχανισμοί για την συλλογή δικτυακών δεδομένων σε εικονικές πλατφόρμες πολλαπλών ενοίκων. Τέτοιες πλατφόρμες επιτρέπουν την ταυτόχρονη χρήση μίας δικτυακής υποδομής από πολλαπλούς χρήστες, διαιρώντας τη σε διαφορετικά εικονικά δίκτυα. Ακόμη, προσφέρουν τη δυνατότητα στους ενοίκους να διαμορφώνουν οι ίδιοι το εικονικό τους δίκτυο, και να του αποδίδουν εξατομικευμένα χαρακτηριστικά. Οι παραπάνω ιδιότητες λαμβάνονται υπ’όψιν στο σχεδιασμό ενός μηχανισμού παρακολούθησης, με σκοπό την άντληση δικτυακών δεδομένων σε εικονικά περιβάλλοντα. Απαραίτητη είναι η ικανότητα του μηχανισμού να ομαδοποιεί τα δεδομένα με βάση τον ένοικο από τον οποίο προέρχονται. Με αυτό τον τρόπο δίνεται η δυνατότητα στους χρήστες να αντλούν και να επεξεργάζονται αποκλειστικά τα δεδομένα που τους αφορούν. Τα δεδομένα που συλλέγονται είναι εξαιρετικά χρήσιμα

στους χρήστες τέτοιων υποδομών διότι βρίσκουν εφαρμογές στην αξιολόγηση νέων μεθόδων διαχείρισης της δικτυακής κίνησης και στην ανάπτυξη μηχανισμών ασφάλειας δικτύων.

Πιο συγκεκριμένα, ο τρόπος λειτουργίας του μηχανισμού στηρίζεται στην αρχιτεκτονική Δικτύων Οριζόμενων από Λογισμικό (SDN) βασισμένων στο πρωτόκολλο OpenFlow. Χρησιμοποιεί κίνηση σηματοδοσίας που ανταλλάσσουν τα στοιχεία μιας υποδομής SDN και δειγματοληψία πακέτων με χρήση του προγράμματος sFlow. Για το διαμοιρασμό των δικτυακών πόρων σε πολλαπλούς ενοίκους χρησιμοποιείται η πλατφόρμα εικονικοποίησης (virtualization platform) OpenVirteX (OVX). Η συγκεκριμένη πλατφόρμα επιλέχθηκε καθώς προσφέρει ιδιαίτερη ευελιξία στους χρήστες κατά τον ορισμό και τη διαχείριση ενός εικονικού δικτύου. Ο μηχανισμός παρακολούθησης προσαρμόστηκε ώστε να είναι δυνατή η συλλογή δικτυακών δεδομένων από εικονικά δίκτυα βασισμένα στο OVX και η αντιστοίχιση τους στους διάφορους ενοίκους.Το τελευταίο αποτελεί βασικό και απαραίτητο χαρακτηριστικό του μηχανισμού καθώς κάθε ένοικος πρέπει να έχει πρόσβαση αποκλειστικά στα δεδομένα που αντιστοιχούν σε πόρους που του έχουν εκμισθωθεί.

Τέλος, μελετάται η συμπεριφορά του μηχανισμού σε εικονικές τοπολογίες με πιο σύνθετες δομές. Η πλατφόρμα OVX χρησιμοποιεί εικονικές αναπαραστάσεις δικτυακών στοιχείων τις οποίες αντιστοιχίζει με φυσικά μηχανήματα. Οι αντιστοιχίσεις αυτές επιτρέπεται να είναι πιο σύνθετες από μία 1:1 αντιστοιχία, δημιουργώντας εικονικές τοπολογίες που διαφέρουν από την φυσική. Μελετάται η συμπεριφορά του μηχανισμού σε τέτοια περιβάλλοντα και προτείνονται εναλλακτικές προσεγγίσεις με στόχο την εξαγωγή ακριβέστερων αποτελεσμάτων.

1.2 Οργάνωση κειμένου

Η παρούσα διπλωματική εργασία οργανώνεται με τον εξής τρόπο. Το δεύτερο κεφάλαιο αποτελεί το θεωρητικό υπόβαθρο της εργασίας καθώς σε αυτό παρουσιάζονται οι απαραίτητες πληροφορίες για την κατανόηση της διπλωματικής. Στο τρίτο κεφάλαιο παρουσιάζεται η βασική αρχιτεκτονική του μηχανισμού καθώς και οι απαραίτητες προσαρμογές που απαιτούνται για τη λειτουργία του σε εικονικό περιβάλλον. Το τέταρτο κεφάλαιο αποτελεί την ανάλυση της υλοποίησης. Παρουσιάζονται δηλαδή τα αρχεία που δημιουργήθηκαν ή τροποποιήθηκαν μαζί με τις ακριβείς ενέργειες που εξυπηρετούν. Το πέμπτο κεφάλαιο αναλύει την πειραματική διαδικασία που ακολουθήθηκε με στόχο την αξιολόγηση της υλοποίησης. Τέλος, το έκτο και τελευταίο κεφάλαιο αποτελεί μία σύνοψη

όλων όσων συζητήθηκαν στην παρούσα διπλωματική, εξάγωντας συμπεράσματα και προτείνοντας μελλοντικές βελτιώσεις και επεκτάσεις.

2

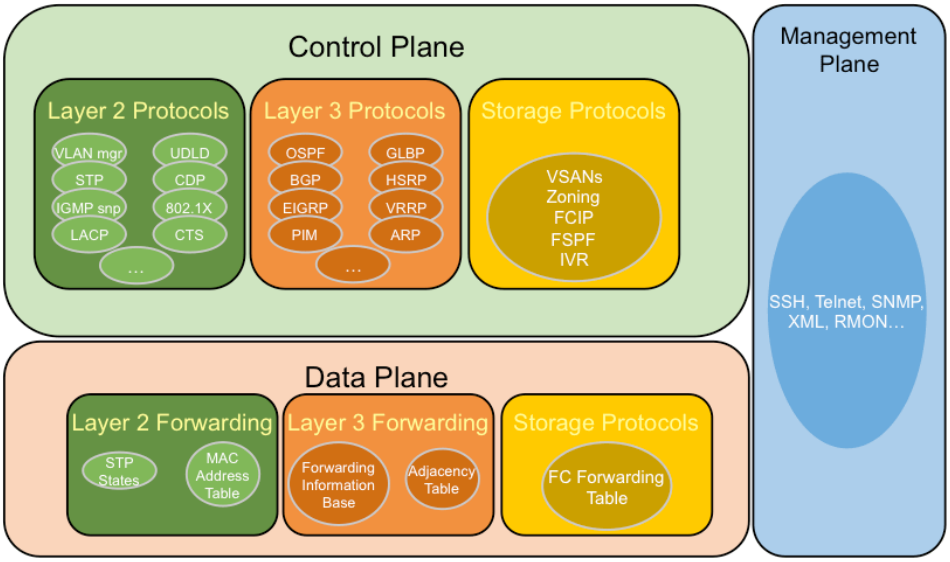
Θεωρητικό υπόβαθρο

Στο συγκεκριμένο κεφάλαιο αναλύεται το θεωρητικό υπόβαθρο, πάνω στο οποίο στηρίζεται η παρούσα διπλωματική εργασία.

2.1 Βασικά χαρακτηριστικά δικτύων υπολογιστών

Η δικτύωση υπολογιστών μπορεί να διαιρεθεί σε τρία επίπεδα λειτουργίας. Το επίπεδο δεδομένων (data plane), το επίπεδο ελέγχου (control plane) και το επίπεδο διαχείρισης (management plane).

- **Data plane**
  Το επίπεδο δεδομένων αντιστοιχεί στα στοιχεία των συσκευών δικτύωσης που είναι υπεύθυνα για την αποτελεσματική προώθηση των δεδομένων.
- **Control plane**
  Το επίπεδο ελέγχου αντιπροσωπεύει τα πρωτόκολλα, τη σηματοδοσία και γενικότερα τα στοιχεία του δικτύου που είναι υπεύθυνα για τη διαμόρφωση και διαχείριση του data plane.
- **Management plane**
  Το επίπεδο διαχείρισης περιλαμβάνει υπηρεσίες λογισμικού (π.χ.SNMP) που χρησιμοποιούνται για την παρακολούθηση του δικτύου και τη διαμόρφωση των λειτουργιών ελέγχου.



Σχήμα 2.1 Network Planes

Η πολιτική δρομολόγησης του δικτύου καθορίζεται στο management plane, το control plane επιβάλλει την πολιτική αυτή στο δίκτυο και το data plane εκτελεί την προώθηση των δεδομένων.

Τα κατανεμημένα πρωτόκολλα ελέγχου και μεταφοράς στους μεταγωγούς και δρομολογητές αποτελούν την κύρια τεχνολογία διάδοσης πληροφοριών στο Διαδίκτυο με τη μορφή ψηφιακών πακέτων. Παρά την ευρεία υιοθέτηση τους, τα παραδοσιακά δίκτυα χαρακτηρίζονται από μειωμένη ευελιξία και περίπλοκη διαχείριση και έλεγχο. Για την εφαρμογή πολιτικών δικτύωσης υψηλού επιπέδου, οι χειριστές των δικτύων πρέπει να ρυθμίσουν κάθε συσκευή δικτύου ξεχωριστά, χρησιμοποιώντας χαμηλού επιπέδου και ,αρκετά συχνά, καθορισμένες από τον προμηθευτή εντολές.

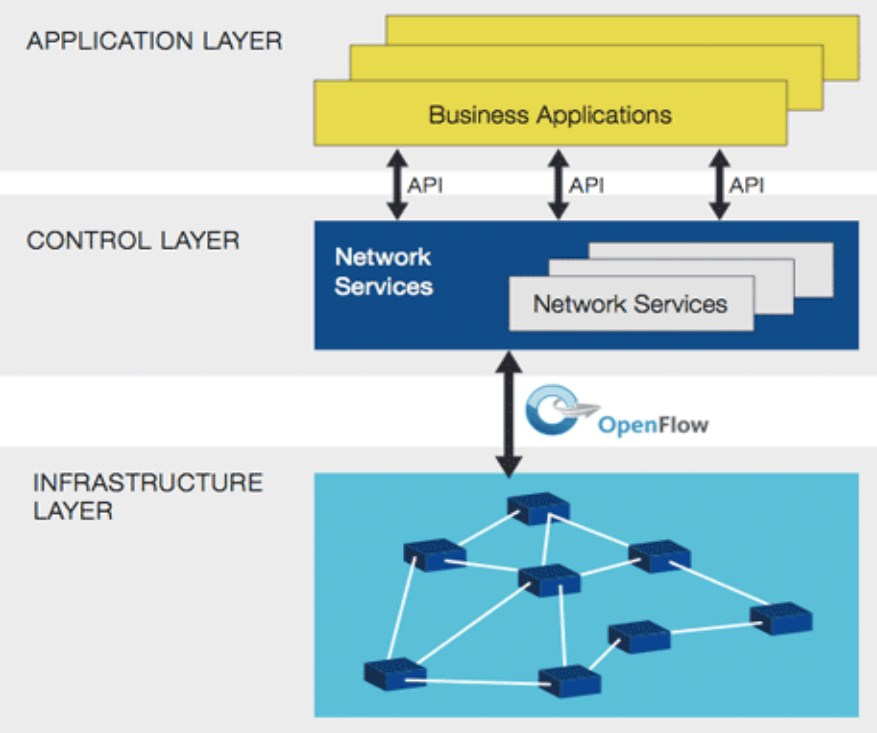
Επιπλέον, ένα δικτυακό περιβάλλον χρειάζεται να έχει αντοχή σε λάθη και να προσαρμόζεται σε πιθανές μεταβολές φορτίου. Μηχανισμοί αυτόματης αναρρύθμισης και απόκρισης δεν υφίστανται και συνεπώς, η εφαρμογή των απαραίτητων πολιτικών δικτύωσης σε ένα τόσο δυναμικό περιβάλλον αποτελεί πρόκληση.

Ακόμη, στα παραδοσιακά δίκτυα IP, τα επίπεδα ελέγχου και δεδομένων (control και data plane) είναι ισχυρά συνδεδεμένα μεταξύ τους και ενσωματωμένα στις συσκευές δικτύωσης, σχηματίζοντας μία αρκετά αποκεντρωμένη δομή. Αυτό το

χαρακτηριστικό θεωρήθηκε αρκετά σημαντικό στη σχεδίαση του Διαδικτύου τον πρώτο καιρό, καθώς προσέδιδε ανθεκτικότητα στο δίκτυο. Επιπροσθέτως, αυτή η προσέγγιση ήταν αρκετά αποτελεσματική και όσον αφορά την απόδοση του δικτύου. Παρόλα αυτά, το αποτέλεσμα ήταν η διαμόρφωση μίας πολύπλοκης και αρκετά στατικής αρχιτεκτονικής.

Τα παραπάνω χαρακτηριστικά δημιουργούν μία βιομηχανία στην οποία η καινοτομία και η εξέλιξη της δικτυακής υποδομής καθίσταται δύσκολη.

2.2 Software Defined Networking



Σχήμα 2.2 Αρχιτεκτονική SDN δικτύων

Η Δικτύωση Οριζόμενη από Λογισμικό (Software Defined Networking) αποτελεί μία αρχιτεκτονική δικτύωσης που στοχεύει να ξεπεράσει τους περιορισμούς της τρέχουσας δικτυακής υποδομής. Χαρακτηρίζεται από την αποσύνδεση του επιπέδου ελέγχου (control plane) και της διαδικασία προώθησης (data plane). Οι μεταγωγείς δικτύου (network switches) μετατρέπονται σε απλές συσκευές προώθησης των πακέτων και ο έλεγχος μεταφέρεται σε μία κεντρικοποιημένη εξωτερική οντότητα , τον SDN controller, ο οποίος αποτελεί το λειτουργικό σύστημα του δικτύου (Network Operating System / NOS) . Ως αποτέλεσμα, το δίκτυο είναι

πλέον προγραμματίσιμο μέσω διαφόρων εφαρμογών που τρέχουν πάνω από το NOS. Ο SDN controller επιτρέπει την επικοινωνία των εφαρμογών με τις συσκευές του data plane, παρέχοντας τους μία κεντρικοποιημένη και αφαιρετική εικόνα της δικτυακής υποδομής, αντιμετωπίζοντας τη σαν μία λογική οντότητα.

Η λογική ενός κεντρικοποιημένου ελέγχου προσφέρει σημαντικά πλεονεκτήματα:

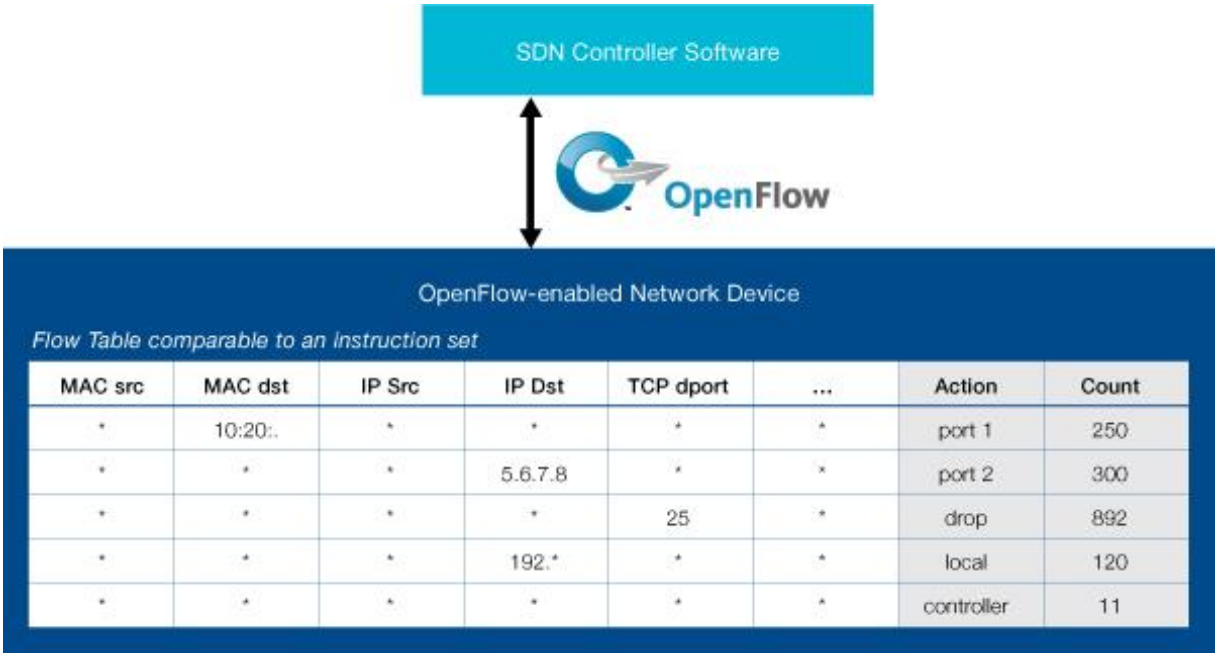
- Απλοποιεί την επεξεργασία των πολιτικών δρομολόγησης του δικτύου μέσω γλωσσών προγραμματισμού υψηλού επιπέδου και ειδικού λογισμικού σε σύγκριση με ρυθμίσεις χαμηλού επιπέδου, εξειδικευμένες για κάθε συσκευή.
- Ένα πρόγραμμα ελέγχου έχει τη δυνατότητα να αντιδρά αυτόματα σε ξαφνικές αλλαγές της κατάστασης του δικτύου, διατηρώντας σε αρκετά μεγάλο βαθμό ανέπαφες τις πολιτικές και τη λειτουργικότητα του.

Η κεντρικοποιημένη και καθολική είκονα της δομής και της κατάστασης του δικτύου στον SDN controller διευκολύνει την ανάπτυξη πιο εξελιγμένων δικτυακών υπηρεσιών και εφαρμογών.

2.2.1 Openflow

Η επικοινωνία του SDN controller με τα στοιχεία του data plane πραγματοποιείται μέσω μίας καλά ορισμένης διεπαφής προγραμματισμού εφαρμογών (API). Υπάρχουν αρκετές προσεγγίσεις για την υλοποίηση μίας τέτοιας διεπαφής. Η επικρατέστερη υλοποίηση χρησιμοποιεί το πρωτόκολλο Openflow.

Το πρωτόκολλο Openflow βασίζεται στην έννοια της ροής πακέτων (packet flow). Μία ροή απαρτίζεται από δύο μέρη. Ένα σύνολο τιμών πεδίου πακέτου που λειτουργεί σαν κριτήριο αντιστοίχισης και ένα σύνολο εντολών που καθορίζει την επεξεργασία και δρομολόγηση των πακέτων.



Σχήμα 2.3 Openflow Instruction Set

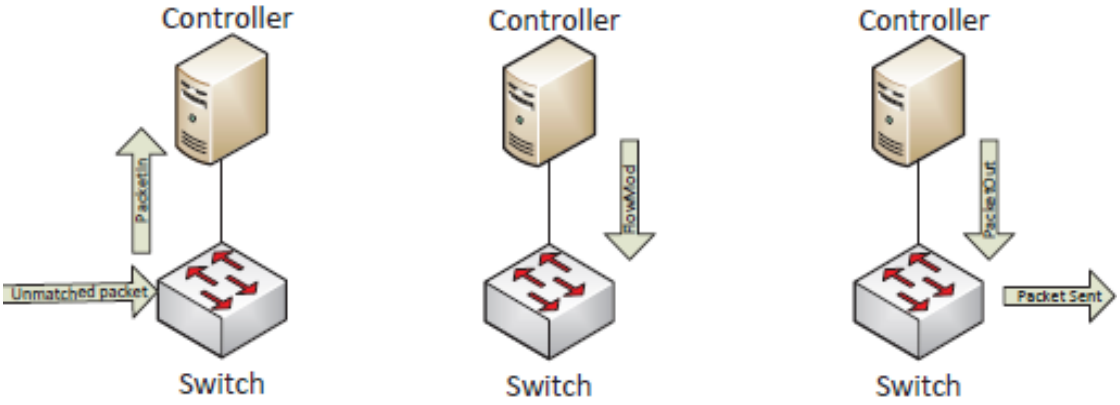
Κάθε δρομολογητής σε ένα δίκτυο SDN διατηρεί πίνακες ροών για την εγκατάσταση των οποίων είναι υπεύθυνος ο controller. Η κίνηση που καταφτάνει σε ένα δρομολογητή αντιστοιχίζεται σε μία ροή και με βάση το σύνολο των εντολών που την αποτελούν επεξεργάζεται και δρομολογείται κατάλληλα.

2.2.2 SDN Controller

Ο έλεγχος της κίνησης στα SDN δίκτυα απομακρύνεται από τα switches του δικτύου και περνάει σε μία κεντρικοποιημένη λογισμική οντότητα, τον SDN controller. Τα switches επικοινωνούν με τον controller μέσω ορισμένων διεπαφών, με επικρατέστερη αυτή του OpenFlow. Μέσω της διεπαφής αυτής, ο controller έχει τη δυνατότητα να εγκαταστήσει στο switch κανόνες προώθησης κίνησης, αλλά και να εξυπηρετήσει αιτήσεις του switch για δρομολόγηση πακέτων που δεν αντιστοιχίζονται σε κάποιο κανόνα. Εκτός από τη διαχείριση του δικτύου, ο controller είναι ικανός να ζητήσει στατιστικά για τη δικτυακή κίνηση, ταξινομημένα ανά ροή.

Πιο συγκεκριμένα, τα SDN switches στέλνουν ένα μήνυμα PacketIn στον controller, κάθε φορά που δέχονται ένα νέο πακέτο το οποίο δεν αντιστοιχίζεται σε κάποιο κανόνα ροής. Ο controller χρησιμοποιώντας μηνύματα τροποποίησης των flow

tables (FlowMod messages) εισάγει νέο κανόνα στο switch προκειμένου να δρομολογηθεί το πακέτο και ρυθμίζει την αποστολή του πακέτου μέσω μηνυμάτων PacketOut. Οι κανόνες που εισάγονται από τα FlowMod έχουν συγκεκριμένο χρόνο ζωής στο switch (idle duration, hard timeout duration). Ο controller ενημερώνεται για κάθε διαγραφή κανόνων ροής μέσω συγκεκριμένων μηνυμάτων FlowRemoved από το switch.



a) First packet of a new connection arrives

b) Installation of forwarding rules

c) Retransmitting the captured packet

Σχήμα 2.4 Διαδικασία εγκατάστασης μίας νέας ροής

Εν κατακλείδι, ο controller χρησιμοποιώντας PacketIn και FlowRemoved μηνύματα ρυθμίζει τους κανόνες ροής στα SDN switches. Ακόμη, τα FlowRemoved μηνύματα ενσωματώνουν τη διάρκεια παραμονής ενός κανόνα στο switch, καθώς και μετρητές πακέτων και byte για τον κανόνα που διαγράφτηκε. Με αυτό τον τρόπο δίνεται η δυνατότητα στον controller να διατηρεί στατιστικά για την κίνηση στο εσωτερικό του δικτύου του.

2.3 Network Virtualization in a SDN environment

Σε αυτό το σημείο θα αναλύσουμε την εικονική δικτύωση (Network Virtualization), η οποία μπορεί να αποτελέσει ένα επιφανές παράδειγμα χρήσης των δικτύων SDN. Κατά τη διαδικασία "εικονικοποίησης" ενός δικτύου αρχικά "διασπάται" η υποκείμενη φυσική τοπολογία και στη συνέχεια, μέσω ειδικών κανόνων δημιουργούνται ξεχωριστά εικονικά δίκτυα. Με αυτό τον τρόπο επιτρέπεται σε πολλαπλά εικονικά δίκτυα να λειτουργούν πάνω σε μία κοινή υποδομή, με τη

δυνατότητα καθένα από αυτά να έχει απλούστερη τοπολογία και μια πιο αφαιρετική εικόνα για το φυσικό δίκτυο. Παρόλο που η SDN τεχνολογία εννοιολογικά είναι ανεξάρτητη από αυτή του virtualization, τα τελευταία χρόνια έχει αναπτυχθεί μία έντονη συσχέτιση μεταξύ τους.

2.3.1 Relationship of Network Virtualization to SDN

Η έννοια του virtualization δικτύων δεν απαιτεί την ύπαρξη ενός περιβάλλοντος SDN. Παρομοίως, ένα δίκτυο SDN δεν συνεπάγεται και virtualization. Παρόλα αυτά, έχει αναπτυχθεί μία συμβίωση αυτών των δύο τεχνολογιών, η οποία καταλύει πολλές ερευνητικές περιοχές. Οι δύο αυτές τεχνολογίες συσχετίζονται με τρεις βασικούς τρόπους:

- Διευκόλυνση του virtualization δικτύων μέσω SDN. Ένας κόμβος ενός εικονονικού δικτύου αποτελεί πλέον ένα switch λογισμικού (όπως το Open vSwitch του πρωτοκόλλου Openflow) που απορροφά κίνηση που προορίζεται για εικονικές μηχανές. Ένας κεντρικοποιημένος λογικός SDN controller εγκαθιστά κανόνες σε αυτά τα switch για να επεξεργάζονται τα πακέτα που απορροφούν και τροποποιεί τους κανονες αυτούς όταν χρειαστεί.
- Αξιολόγηση και δοκιμές των SDN δικτύων μέσω εικονικών δικτύων. Η δυνατότητα διαχωρισμού μίας εφαρμογής ελέγχου από το data plane δίνει τη δυνατότητα να δοκιμαστεί η εφαρμογή αυτή σε ένα εικονικό περιβάλλον δικτύωσης πριν την εφαρμογή της σε ένα λειτουργικό δίκτυο. Για παράδειγμα, το Mininet αντιστοιχίζει OpenFlow switches, end hosts και SDN controllers σε νήματα διεργασιών σε ένα μηχάνημα (είτε φυσικό είτε εικονικό), με αποτέλεσμα να προσομοιώνει ένα ολόκληρο δίκτυο στο μηχάνημα αυτό. Σε ένα τέτοιο περιβάλλον είναι εύκολη η αξιολόγηση μίας εφαρμογής ελέγχου σε μία ολικής κλίμακας προσομοίωση.
- Virtualization (τεμαχισμός) ενός SDN δικτύου. Στα συμβατικά δίκτυα, η δημιουργία ενός εικονικού switch αποτελεί περίπλοκη διαδικασία καθώς κάθε εικονική μηχανή απαιτείται να τρέχει το δικό της λογισμικό σε επίπεδο control plane. Αντιθέτως, το virtualization ενός switch αποσυνδεδεμένο από το επίπεδο ελέγχου είναι πολύ πιο απλοποιημένη διαδικασία. Η κεντρική ιδέα είναι ο διαχωρισμός του flowspace (χώρος ροών) σε slices (τεμάχια), όπου κάθε slice χρησιμοποιεί ένα κομμάτι των πόρων του δικτύου και το διαχειρίζεται ένας ξεχωριστός controller. Χρησιμοποιείται μία λογισμική οντότητα (hypervisor), σα μεσολαβητής για τη σωστή επικοινωνία των controllers και τις υποκείμενης φυσικής τοπολογίας.

2.3.2 Infrastructure as a Service

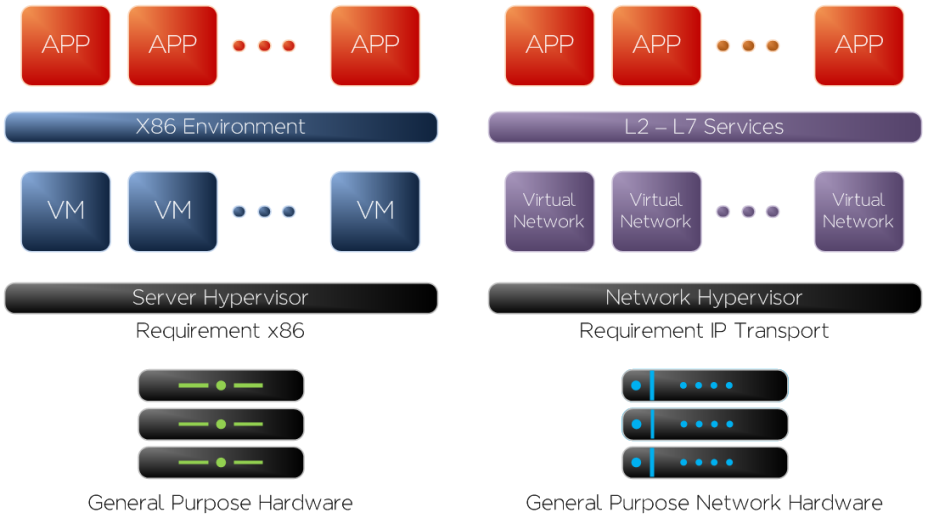
Αξίζει να σημειωθεί ότι η ιδέα του τεμαχισμού των δικτυακών πόρων μιας φυσικής δικτυακής υποδομής για τη δημιουργία εικονικών slices βασίζεται στο μοντέλο της υποδομής σαν υπηρεσία (Infrastructure as a Service, IaaS). Σε αυτό το μοντέλο, οι φυσικοί πόροι ενός δικτύου μοιράζονται μεταξύ διαφόρων χρηστών. Τα λειτουργικά θέματα του δικτύου (συμφόρηση δικτύου, βλάβες κόμβων/γραμμών, κτλπ) απασχολούν αποκλειστικά τους διαχειριστές του, χωρίς να επιρεάζουν τους ίδιους τους χρήστες. Αυτή η επιλογή είναι ελκυστική για τους χρήστες – ενοίκους του δικτύου που επιθυμούν να έχουν μία αφαιρετική εικόνα του δικτύου και να μην ασχολούνται με τις λεπτομέρειες της υποκείμενης τοπολογίας. Ακόμη, η ιδέα αυτή βρίσκει πολλές εφαρμογές σαν πειραματική πλατφόρμα (testbed) για τη δοκιμή τεχνολογιών και τεχνικών δικτύωσης νέας γενιάς.

2.4 Network Hypervisor

Όπως είδαμε προηγουμένως, το virtualization ενός SDN δικτύου επιτρέπει σε πολλαπλούς ενοίκους να μοιράζονται τους φυσικούς δικτυακούς πόρους μίας υποδομής. Κάθε ένοικος εκμεταλλεύεται το δικό του τεμάχιο (slice) του φυσικού δικτύου στο οποίο τρέχει τις δικές του εφαρμογές. Βασική συνιστώσα για τη διαδικασία αυτή αποτελεί η ύπαρξη ενός SDN hypervisor. Ο hypervisor είναι υπεύθυνος για την διάσπαση του δικτύου SDN σε πολλαπλά απομονωμένα μεταξύ τους εικονικά δίκτυα SDN, καθένα με το δικό του SDN controller.

2.4.1 Χαρακτηριστικά των Hypervisors

2.4.1.1 Hypervisors: From Virtual Machines to Virtual Networks



Σχήμα 2.5 Αντιστοιχία Virtual Machine - Virtual Network

Οι hypervisors αρχικά αναπτύχθηκαν για την παρακολούθηση και λειτουργία εικονικών μηχανών (virtual machine, VM). Πολλαπλά VMs είχαν τη δυνατότητα να λειτουργήσουν πάνω σε μία υπολογιστική πλατφόρμα, καθένα τρέχοντας το δικό του λειτουργικό σύστημα. Ο hypervisor ήταν υπεύθυνος για την παρακολούθηση των VMs αυτών, τη διανομή των υπολογιστικών πόρων καθώς και για την απομόνωση και ασφαλή συνύπαρξη των Vms στην ίδια πλατφόρμα. Η ύπαρξη των VMs επέτρεπε στις διάφορες εφαρμογές να χρησιμοποιήσουν το λειτουργικό τους σύστημα χωρίς να τους επηρεάζουν οι ξεχωριστές λεπτομέρειες και τα χαρακτηριστικά της υποκείμενης υπολογιστικής πλατφόρμας.

Παρόμοια με τα VMs, τα εικονικά δίκτυα αναπτύχθηκαν για να επιτρέπουν σε νέες δικτυακές υπηρεσίες και εφαρμογές να χρησιμοποιηθούν χωρίς να επηρεάζονται από τα ειδικά χαρακτηριστικά της κάθε φυσικής δικτυακής τοπολογίας. Επίσης, μέσω virtualization, πολλαπλά εικονικά δίκτυα μπορούν να λειτουργήσουν πάνω στην ίδια φυσική υποδομή. Ο hypervisor παρακολουθεί τα εικονικά δίκτυα, και είναι υπεύθυνος για τη διανομή των δικτυακών πόρων και την απομόνωση της δικτυακής κίνησης των διαφορετικών ενοίκων.

2.4.1.2 Managing a Physical SDN Network with a Hypervisor

Η χρήση ενός κεντρικοποιημένου SDN controller σε σύζευξη με την υποτυπώδη διεπαφή του (API) έχουν ως αποτέλεσμα τη δημιουργία «προγραμματιζόμενων» (ευφυών) δικτύων. Τα δίκτυα δεν αποτελούνται πλέον από ένα σύνολο συσκευών που απαιτεί ατομική ρύθμιση. Αντιθέτως φαίνονται και λειτουργούν σαν μία ενιαία και προγραμματίσιμη οντότητα. Η προγραμματιστική αυτή ιδιότητα μπορεί να χρησιμοποιηθεί για την δημιουργία εικονικών SDN δικτύων.

Πιο συγκεκριμένα, ένα SDN δίκτυο μπορεί να εικονικοποιηθεί με την εισαγωγή ενός hypervisor ανάμεσα στο φυσικό επίπεδο και στο SDN control plane. Ο hypervisor αλληλεπιδρά με το σύνολο της φυσικής υποδομής μέσω ενός interface (data – control plane interface, D-CPI). Με πολλαπλά D-CPI αλληλεπιδρά επίσης με τους ξεχωριστούς SDN controllers των διαφόρων ενοίκων. Η μεσολάβηση αυτή του hypervisor θεωρείται καθοριστική ιδιότητα για την επίτευξη εικονικής δικτύωσης σε δίκτυα SDN.

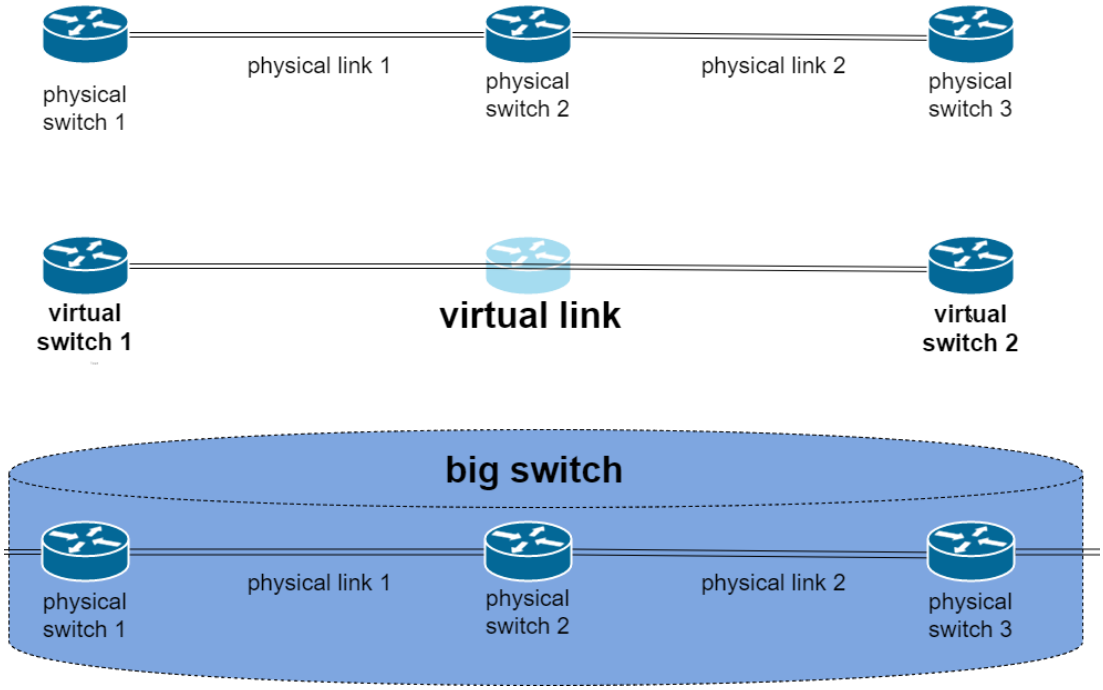
Ο hypervisor ουσιαστικά αποσπά τη φυσική τοπολογία και δημιουργεί απομονωμένα εικονικά δίκτυα τα οποία είναι διαχειρίσιμα από τους αντίστοιχους εικονικούς SDN controllers τους.

2.4.1.3 Network Attribute Virtualization

Όπως αναφέραμε προηγουμένως, ο hypervisor αποσπά τα επιμέρους στοιχεία της δικτυακής τοπολογίας και δημιουργεί αντιστοιχίσεις φυσικών κι εικονικών συσκευών, παρουσιάζοντας μία απλοποιημένη αναπαράσταση της δικτυακής υποδομής. Η απλοποιημένη αυτή αναπαράσταση μεταφέρεται από τον hypervisor στους controllers των διαφόρων ενοίκων των εικονικών δικτύων. Τα τρία πιο κοινά χαρακτηριστικά του δικτύου, τα οποία παρουσιάζονται απλοποιημένα, είναι η δικτυακή τοπολογία, οι δικτυακοί πόροι των φυσικών κόμβων και οι δικτυακοί πόροι των φυσικών γραμμών.

- **Φυσική τοπολογία**: Ο hypervisor έχει τη δυνατότητα να αντιστοιχίσει ένα μονοπάτι που διασχίζει ολόκληρο το δίκτυο, περιλαμβάνοντας πολλαπλά φυσικά switch και links σαν μία απλή εικονική γραμμή.
  Ο μικρότερος βαθμός virtualization μίας τοπολογίας αντιστοιχεί σε μία 1 προς 1 αντιστοίχιση της φυσικής με την εικονική τοπολογία. Αντίστοιχα, στο

μεγαλύτερο βαθμό ολόκληρη η φυσική τοπολογία αντιστοιχίζεται είτε σε ένα εικονικό link ή σε ένα εικονικό switch. Σε γενικές γραμμές υπάρχει σημαντική διακύμανση ανάμεσα στο υψηλότερο και στο χαμηλότερο βαθμό virtualization ενός δικτύου.



Σχήμα 2.6 Διαφορετικές εικονικές αναπαραστάσεις μίας φυσικής τοπολογίας

- **Δικτυακοί πόροι φυσικών κόμβων**: Όσον αφορά το συγκεκριμένο χαρακτηριστικό, κατά κύριο λόγο οι πόροι του CPU και οι πόροι αποθήκευσης των πινάκων ροής (flow tables) ενός SDN switch καθορίζουν το επίπεδο virtualization. Η διακύμανση στο συγκεκριμένο σημείο καθορίζεται από το ποσοστό των δικτυακών πόρων (υπολογιστικοί πόροι και μνήμη) της φυσικής συσκευής που είναι διαθέσιμα στον κάθε εικονικό κόμβο.
- **Δικτυακοί πόροι φυσικών γραμμών**: Το επίπεδο virtualization για το συγκεκριμένο χαρακτηριστικό καθορίζεται από το εύρος ζώνης των γραμμών, τις διαθέσιμες ουρές προτεραιότητας στις γραμμές και τους διαθέσιμους buffers.

2.4.1.4 Isolation

Μία από τις ευθύνες του hypervisor είναι να παρέχει απομόνωση μεταξύ των διαφορετικών εικονικών SDN δικτύων που μοιράζονται το φυσικό δίκτυο. Η

απομόνωση αυτή οφείλει να περιλαμβάνει το data plane και το control plane, καθώς και τη διευθυνσιοδότηση του κάθε δικτύου.

- **Control Plane Isolation**: Στα SDN δίκτυα ο έλεγχος βρίσκεται στους κεντρικοποιημένους λογικούς controllers. Ο SDN controller ενός εικονικού δικτύου επικοινωνεί με το hypervisor, ο οποίος είναι υπεύθυνος για τη μεταφορά των εντολών στο φυσικό δίκτυο. Ο κάθε εικονικός controller πρέπει να αντιλαμβάνεται ότι ελέγχει το δίκτυο για το οποίο είναι υπεύθυνος, χωρίς παρεμβάσεις από άλλα εικονικά δίκτυα (ή κατ' επέκταση από άλλους controllers).
  Ακόμη, η απόδοση του control plane επηρεάζεται από τους διαθέσιμους πόρους των συσκευών που παρέχει ο hypervisor για κάθε εικονικό δίκτυο. Οι διαθέσιμοι υπολογιστικοί πόροι επηρεάζουν την επεξεργασία και μετάφραση των πακέτων ενώ οι διαθέσιμοι αποθηκευτικοί πόροι περιορίζουν το control plane buffering.
- **Data Plane Isolation**: Η απομόνωση σε επίπεδο φυσικών links αφορά το ρυθμό μετάδοσης (bit rate), ενώ η απόδοση των φυσικών κόμβων εξαρτάται από την υπολογιστική τους ικανότητα. Κάτω από διαφορετικούς φόρτους εργασιών, η απαίτηση σε πόρους αλλάζει, οδηγώντας σε αυξομειώσεις σε επίπεδο απόδοσης. Οι πόροι των φυσικών switch πρέπει να ανατίθενται σε κάθε εικονικό δίκτυο. Μηχανισμοί απομόνωσης χρειάζονται για να αποτραπούν αλληλεπιδράσεις μεταξύ των πολλαπλών δικτύων, π.χ. ένα εικονικό δίκτυο να χρησιμοποιεί μεγαλύτερο ποσοστό των πόρων από αυτό που του αναλογεί, οδηγώντας τα υπόλοιπα δίκτυα σε υπολειτουργία.
  Ένα ακόμη χαρακτηριστικό των SDN switches είναι η ικανότητα να αποθηκεύουν και να αντιστοιχίζουν κανόνες ροής (flow rules). Για σωστή απομόνωση και λειτουργία των εικονικών δικτύων, πρέπει συγκεκριμένη ποσότητα αποθηκευτικού χώρου να ανατίθεται σε κάθε ένοικο.
  Τέλος, ο hypervisor πρέπει να παρέχει απομόνωση σε επίπεδο ρυθμού μετάδοσης των φυσικών γραμμών στοχεύοντας στην αύξηση της ποιότητας των υπηρεσιών του (Quality-of-Service, QoS). Ο hypervisor πρέπει να είναι ικανός να αναθέτει συγκεκριμένο ρυθμό μετάδοσης δεδομένων σε κάθε ένοικο.
- **vSDN Addressing Isolation**: Η SDN αρχιτεκτονική αντιστοιχίζει της ροές πακέτων σε κανόνες οριζόμενους από τους controllers. Για το λόγο αυτό, οι ροές δεδομένων των διαφορετικών ενοίκων πρέπει να είναι μοναδικά διευθυνσιοδοτημένες. Για να επιτευχθεί αυτό, μία προσέγγιση είναι να μοιραστεί ο χώρος διευθύνσεων ώστε οι ένοικοι να χρησιμοποιούν μη επικαλυπτόμενες διευθύνσεις. Μία ακόμη λύση είναι η χρήση μηχανισμών, π.χ. αξιοποίηση πεδίων εκτός των πεδίων αντιστοίχισης των ροών, για το

διαχωρισμό των πακέτων μεταξύ των ενοίκων διαθέτοντας τους με αυτό τον τρόπο ολόκληρο το χώρο διευθύνσεων.

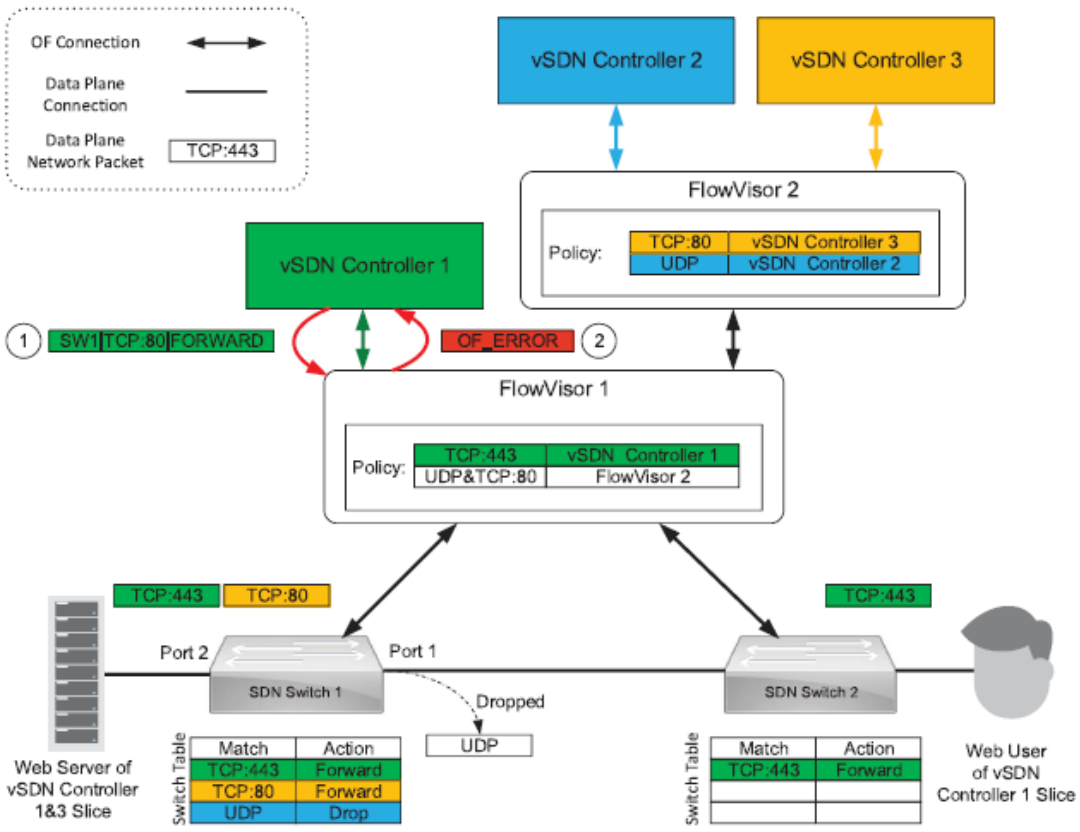
2.4.2 FlowVisor

Ο FlowVisor είναι ο πρώτος hypervisor που αναπτύχθηκε με σκοπό τον τεμαχισμό SDN δικτύων και τη δημιουργία εικονικών δικτύων με βάση το πρωτόκολλο OpenFlow (OF protocol). Κύριος στόχος του FlowVisor είναι η παράλληλη λειτουργία κανονικών και πειραματικών δικτύων στην ίδια φυσική SDN υποδομή. Συνεπώς, επικεντρώνεται στην απομόνωση της πειραματικής δικτυακής κίνησης από την υπόλοιπη κίνηση του δικτύου. Επιπλέον, στις σχεδιαστικές αρχές του FlowVisor ανήκει και η λειτουργία με διαφανή τρόπο, χωρίς δηλαδή αντιλαμβάνεται την παρουσία του ένα εικονικό δίκτυο, καθώς και η δημιουργία ενός επεκτάσιμου και ευέλικτου τρόπου καθορισμού των διαφόρων slices του δικτύου.

2.4.2.1 Αρχιτεκτονική

Ο FlowVisor αποτελεί καθαρά μία υλοποίηση λογισμικού. Μπορεί να αναπτυχθεί πάνω σε οποιοδήποτε γενικού σκοπού υπολογιστικό server που τρέχει κάποιο εμπορικό λειτουργικό σύστημα. Για να έχει τη δυνατότητα να διαχειρίζεται και να ελέγχει τα διάφορα slices, ο FlowVisor λειτουργεί σε ένα επίπεδο ανάμεσα στους controllers των ενοίκων και τα διάφορα φυσικά switches. Με αυτό τον τρόπο ελέγχει την εικόνα που έχει ο κάθε controller για τη φυσική τοπολογία, καθώς και την πρόσβαση του σε αυτή.

Ο FlowVisor εισάγει τον όρο flowspace (χώρος ροών) για να προσδιορίσει ένα υποσύνολο του χώρου των πεδίων της OpenFlow κεφαλίδας σε ένα δίκτυο βασισμένο στο πρωτόκολλο OF. Κατανέμει σε κάθε ένοικο το δικό του flowspace, δηλαδή το δικό του υποσύνολο των OF πεδίων της κεφαλίδας και εξασφαλίζει ότι τα διάφορα διαφορετικά flowspaces δεν επικαλύπτονται.



Σχήμα 2.7 Παράδειγμα λειτουργίας του FlowVisor

2.4.2.2 Abstraction and Isolation Features

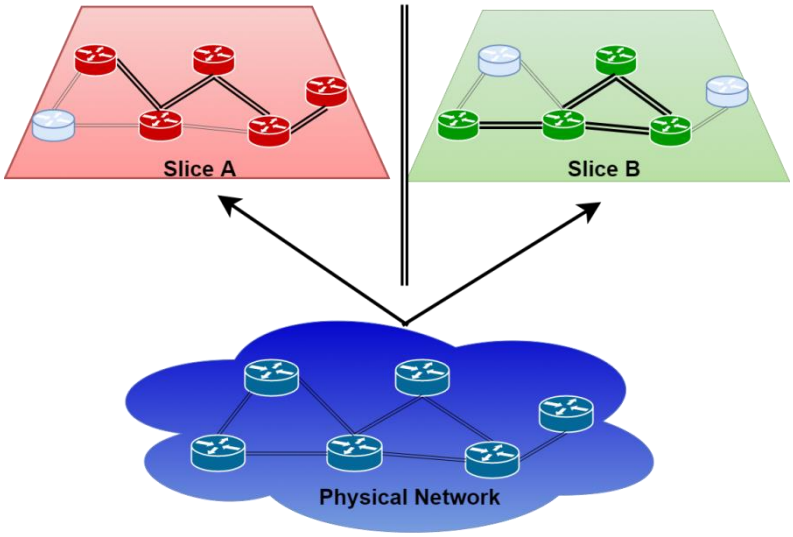
Ο FlowVisor παρέχει στα εικονικά SDN δίκτυα απομόνωση σε επίπεδο εύρους ζώνης γραμμών, τοπολογίας, υπολογιστικών πόρων των switches, flowspace, καταχωρήσεων ροών καθώς και απομόνωση του OpenFlow καναλιού ελέγχου (D-CPI, Data- Control Plane Interface).

α) Εύρος ζώνης γραμμών

Για το διαμοιρασμό του εύρους ζώνης γραμμών, ο FlowVisor αντιστοιχεί τα πακέτα ενός slice μέσω ενός Virtual Local Area Network Priority Code Point (VLAN PCP) πεδίου σε ουρές προτεραιότητας. Το VLAN PCP αποτελεί ένα 3-bit πεδίο συνεπώς έχει τη δυνατότητα να ορίζει 8 διαφορετικές ουρές προτεραιότητας. Περισσότερες, πιο εξελιγμένες μέθοδοι για απομόνωση και χρονοδρομολόγηση εξετάζονται, οι οποίες επεκτείνουν τον FlowVisor σε άλλες υλοποιήσεις όπως π.χ. ο Enhanced FlowVisor.

β) Τοπολογία

Όσον αφορά την απομόνωση τοπολογίας, ο FlowVisor παρουσιάζει στους διάφορους controllers των ενοίκων μόνο τους φυσικούς πόρους , switches κλπ, που ανήκουν στο δικό τους slice. Για να το επιτύχει αυτό, λειτουργεί σαν ενδιάμεσος (proxy) controller μεταξύ των controllers και του φυσικού δικτύου και επεξεργάζεται τα OF μηνύματα ώστε να αναφέρουν τους φυσικούς πόρους ενός slice μόνο στον controller που το διαχειρίζεται. Με αυτό τον τρόπο έχει τη δυνατότητα να δημιουργεί υποσύνολα ενός δικτύου και να αναθέτει τη διαχείριση καθενός από αυτά σε διαφορετικό controller.



Σχήμα 2.8 Παράδειγμα τεμαχισμού ενός δικτύου σε slices

γ) Υπολογιστικοί πόροι

Η επεξεργασία των OF μηνυμάτων μπορεί να υπερφορτώσει την κεντρική υπολογιστική μονάδα (CPU) ενός SDN switch, καθιστώντας το ανίκανο να εξυπηρετήσει την υπόλοιπη δικτυακή κίνηση. Για να εξασφαλιστεί ότι κάθε slice θα μπορεί να χρησιμοποιεί τους υπολογιστικούς πόρους του switch, ο FlowVisor ασκεί περιορισμούς στο ρυθμό ανταλλαγής των διαφορετικών OF μηνυμάτων μεταξύ των controllers και των φυσικών switches.

δ) Flowspace

Όπως αναφέρθηκε προηγουμένως, τα flowspaces των διαφορετικών εικονικών SDN δικτύων δεν επιτρέπεται να επικαλύπτονται. Όταν ένας ένοικος προσπαθεί να ορίσει έναν κανόνα που επηρεάζει την κίνηση έξω από το δικό του slice, ο FlowVisor αρχικά τροποποιεί τον κανόνα αυτό ώστε να μη ξεφεύγει από το flowspace του slice του και σε περίπτωση που η τροποποίηση αυτή είναι αδύνατη, στέλνει μήνυμα σφάλματος και απορρίπτει τον κανόνα.

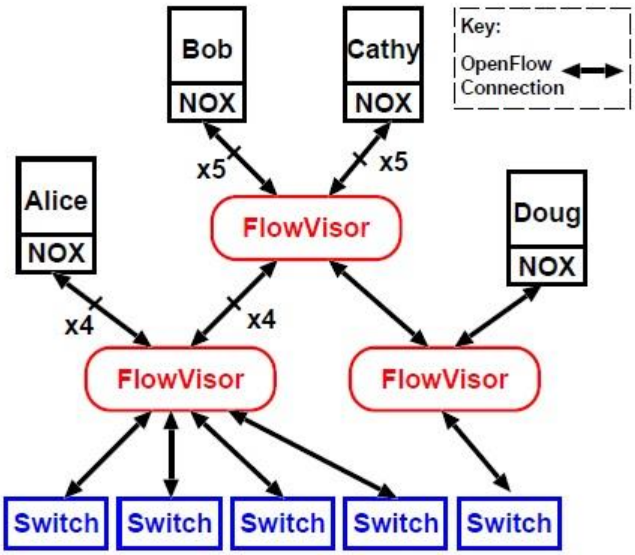
ε) Καταχωρήσεις Ροών

Τα SDN switches αποθηκεύουν της εγγραφές ροών (flow entries/flow rules) σε μία εσωτερική TCAM μνήμη. Ο FlowVisor αναθέτει σε κάθε ένοικο ένα μέρος αυτής της μνήμης για την αποθήκευση των κανόνων που εισάγει ο controller του. Για να επιτευχθεί απομόνωση, κάθε switch παρακολουθεί τον αριθμό των εγγραφών ροών που εισάγει κάθε ένοικος και σε περίπτωση που αυτός ξεπερνά το όριο που του έχει ανατεθεί, ο FlowVisor ενημερώνει τον controller ότι η μνήμη του switch είναι γεμάτη.

στ) OpenFlow κανάλι ελέγχου

Εκτός από απομόνωση σε επίπεδο φυσικών πόρων σε ένα SDN δίκτυο, απαραίτητη είναι και η απομόνωση σε επίπεδο επικοινωνίας data με control plane. Για να το επιτύχει αυτό, ο FlowVisor επεξεργάζεται ένα πεδίο του OF πρωτοκόλλου, το OF transaction identifier (αναγνωριστικό συναλλαγής). Με αυτό τον τρόπο εξασφαλίζει ότι τα διαφορετικά εικονικά SDN δίκτυα χρησιμοποιούν διαφορετικούς identifiers.

2.4.2.3 Nesting



Σχήμα 2.9 FlowVisor's Nesting Attribute

Με στόχο την παροχή απλών μέσων για τον ορισμό διαφορετικών πολιτικών, ένα πρόγραμμα του FlowVisor έχει τη δυνατότητα να τρέχει πάνω από ένα άλλο. Αυτό σημαίνει ότι δύο ή παραπάνω προγράμματα FlowVisor μπορούν να είναι ένθετα, προσφέροντας διαφορετικά επίπεδα αφαιρετικότητας και επεξεργασίας του δικτύου.

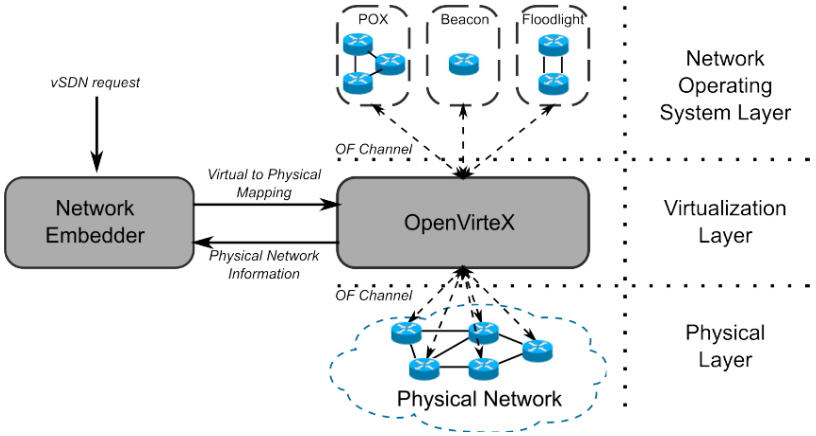
2.4.3 OpenVirteX

Ο OpenVirteX (OVX) είναι ένας δικτυακός hypervisor, ο οποίος δημιουργήθηκε για να παρέχει:

i) virtualization σε επίπεδο διευθύνσεων
ii) virtualization σε επίπεδο τοπολογίας
iii) τη δυνατότητα σε κάθε ένοικο του δικτύου να χρησιμοποιεί το δικό του Δικτυακό Λειτουργικό Σύστημα (Network Operating System , NOS).

Η ανάπτυξη του OVX βασίστηκε πάνω στον FlowVisor. Όπως και ο FlowVisor, ο OVX χρησιμοποιεί το πρωτόκολλο OpenFlow σαν διεπαφή για επικοινωνία με τα switches της τοπολογίας και με τους controllers των διαφόρων ενοίκων. Λειτουργεί σαν ενδιάμεσος (proxy) controller παρουσιάζοντας εικονικά OpenFlow δίκτυα στους ενοίκους, ενώ παράλληλα ελέγχει ο ίδιος την φυσική δικτυακή τοπολογία.

Δημιουργεί δηλαδή πολλαπλά εικονικά δίκτυα σε μία SDN υποδομή και μεταφέρει τον έλεγχό τους στους ενοίκους μέσω των δικών τους NOS.



Σχήμα 2.10 Αρχιτεκτονική Συστήματος OVX

Σε αντίθεση με τον FlowVisor που απλά τεμαχίζει το δίκτυο, δημιουργώντας υποσύνολα αυτού, ο OVX έχει τη δυνατότητα να παρέχει ένα πλήρως εικονικοποιημένο δίκτυο, με τοπολογία προσαρμοσμένη στις ανάγκες του εκάστοτε tenant. Ακόμη, τα slices του FlowVisor μοιράζονται τον ίδιο χώρο διευθύνσεων και ροών και συνεπώς δεν μπορούν να είναι τελείως ανεξάρτητα μεταξύ τους. Ο OVX επιτρέπει στους ενοίκους του να ορίσουν οι ίδιοι τον χώρο διευθύνσεων που επιθυμούν, προσφέροντας έτσι πλήρη ανεξαρτησία, ενώ χρησιμοποιεί διάφορους άλλους μηχανισμούς για να εξασφαλίσει απομόνωση σε κάθε εικονικό δίκτυο.

2.4.3.1 Network Embedder

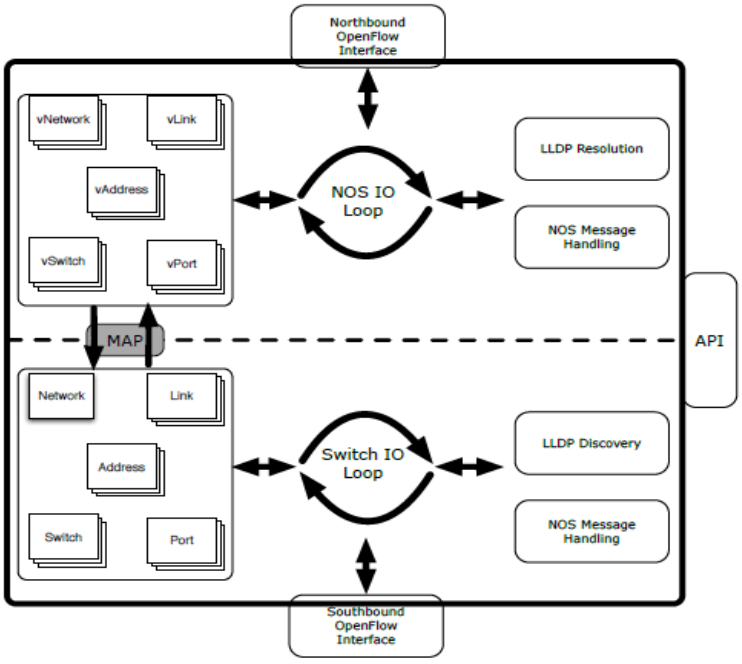
Ο OVX είναι μία πλατφόρμα εικονικής δικτύωσης ικανή να παράγει εικονικά δίκτυα SDN που χρησιμοποιούν το πρωτόκολλο OpenFlow. Σε αυτά τα δίκτυα παρέχεται μεγάλη ελευθερία κατά τον ορισμό της τοπολογίας και του συνόλου διευθύνσεων. Ο ένοικος ορίζει την επιθυμητή μορφή του εικονικού του δικτύου χρησιμοποιώντας μία διεπαφή (API) που επικοινωνεί με τον OVX με χρήση ενός εργαλείο, το network embedder (σχήμα 2.10).

Αρχικά, ο χρήστης ορίζει το χώρο διευθύνσεων του δικτύου του, τη μορφή της επιθυμητής τοπολογίας και ένα σύνδεσμο με τον controller του (NOS) στον embedder. Ο embedder δημιουργεί μία αντιστοίχιση εικονικών και φυσικών πόρων,

βασιζόμενος σε πληροφορίες του δικτύου που αντλεί από τον OVX. Στη συνέχει, η αντιστοίχιση αυτή μεταφέρεται στον OVX ο οποίος εγκαθιστά το εικονικό δίκτυο πάνω στη φυσική τοπολογία.

2.4.3.2 Εσωτερική Αρχιτεκτονική

Εσωτερικά, ο OVX βασίζεται στην αποσύνδεση των εικονικών δικτυακών στοιχείων από τα φυσικά αντίστοιχα τους. Για το σκοπό αυτό προσομοιώνονται όλα τα φυσικά και εικονικά εξαρτήματα του δικτύου (π.χ. PhysicalSwitch – VirtualSwitch) και αντιστοιχίζονται μεταξύ τους από τον embedder. Η αντιστοίχιση αυτή διατηρείται στον OVX.



Σχήμα 2.11 Εσωτερική αρχιτεκτονική του OVX

Όλα τα εικονικά στοιχεία του δικτύου αντιστοιχίζονται τουλάχιστον σε ένα φυσικό. Η συγκεκριμένη αντιστοίχιση δεν καθορίζει τον τρόπο υλοποίησης της δημιουργίας εικονικών δικτύων. Κάθε εικονικό στοιχείο ουσιαστικά αποτελεί ένα δείκτη προς κάποιο πραγματικό δικτυακό εξάρτημα. Υπάρχει η δυνατότητα ενεργοποίησης/απενεργοποίησης, τροποποίησης και αναδιοργάνωσης των εικονικών δικτυακών στοιχείων πριν αλλά και κατά τη διάρκεια της εκτέλεσης του vSDN δικτύου.

Ο OVX είναι υλοποιημένος σε γλώσσα java. Όλα τα δικτυακά στοιχεία αναπαριστώνται σε κλάσεις και χρησιμοποιούνται υποκλάσεις που επεκτείνουν τις κλάσεις αυτές για την αναπαράσταση των φυσικών και εικονικών τους εκδοχών, όπως φαίνεται στους παρακάτω πίνακες.

Base Class	Representation	Definition
Network	Ολόκληρη η δικτυακή τοπολογία	public abstract class Network<T1 extends Switch, T2 extends Port, T3 extends Link>
Switch	Ένα switch	public abstract class Switch<T extends Port>
Port	Μία πόρτα σε ένα switch	public class Port<T1 extends Switch, T2 extends Link>
Link	Μία σύνδεση μεταξύ δύο πορτών	public abstract class Link<T1 extends Port, T2 extends Switch>
Host	Ένας δικτυακός host	public class Host
IPAddress	Μία IP Address	public abstract class IPAddress

Πίνακας 2.1 Κλάσεις αναπαράστασης δικτυακών στοιχείων στον OpenVirteX

Base Class	Physical Component Class	Virtual Component Class(es)
Network	PhysicalNetwork	OVXNetwork
Switch	PhysicalSwitch	OVXSwitch (child classes OVXSingleSwitch, OVXBigSwitch)
Port	PhysicalPort	OVXPort
Link	PhysicalLink	OVXLink, SwitchRoute
Host	–	Host
IPAddress	PhysicalIPAddress	OVXIPAddress

Πίνακας 2.2 Αντιστοίχιση φυσικών - OVX κλάσεων

Η κλάση Switch

Το Switch αποτελεί την αναπαράσταση ενός datapath στον OVX. Περιγράφεται από ένα σύνολο πορτών, ένα switch (ή αλλιώς datapath ID ,DPID) και συγκεκριμένες περιγραφές για τις δυνατότητες και τα χαρακτηριστικά του.

Χρησιμοποιείται σα σημείο αναφοράς για το OF κανάλι επικοινωνίας μεταξύ του OVX και των εικονικών switches των ενοίκων ή μεταξύ του OVX και των φυσικών switches του δικτύου . Τα μηνύματα που καταφθάνουν στον OpenVirteX είτε θα

δρομολογηθούν απευθείας είτε θα περάσουν μέσα από συγκεκριμένες μεθόδους του Switch για ιδιαίτερη διαχείριση.

PhysicalSwitch (extends Switch)

Η κλάση PhysicalSwitch αναπαριστά ένα φυσικό switch του δικτύου που έχει συνδεθεί στον OVX. Τα χαρακτηριστικά του PhysicalSwitch καθορίζονται από τα περιεχόμενα των OF μηνυμάτων κατά τη διάρκεια της σύνδεσης του switch με τον OVX και μέσω μηνυμάτων κατάστασης που ανταλλάσσονται κατά τη διάρκεια της λειτουργίας του. Η κλάση PhysicalSwitch περιέχει μία αναπαράσταση του flow table που βρίσκεται στο switch που της αντιστοιχεί και μία εσωτερική κλάση, SwitchDeregAction, για το συγχρονισμό με το switch αυτό.

OVXSwitch (extends Switch)

Το OVXSwitch αποτελεί την κλάση που αναπαριστά τα εικονικά switches του OVX (τα switches όπως είναι ορατά από τον κάθε controller). Διατηρεί στο εσωτερικό του ένα εικονικό flow table (flow table με τις καταχωρήσεις που αφορούν τον ένοικο που του αντιστοιχεί) και είναι ικανό να συνδέεται με πολλαπλούς controllers και να διαχειρίζεται το ρόλο του καθενός (role management).

Οι δύο κλάσεις που επεκτείνουν το OVXSwitch αντιπροσωπεύουν τους δύο τρόπους virtualization των switches.

- OVXSingleSwitch: Ένα εικονικό switch που αντιστοιχίζεται σε ένα φυσικό switch στο δίκτυο.

- OVXBigSwitch: Ένα εικονικό switch που αντιστοιχίζεται σε πολλαπλά φυσικά switches του δικτύου, ενώ στον controller του ενοίκου εμφανίζεται ως ένα απλό switch (Big Virtual Switch, BVS). Εκτός από τα στοιχεία που αναπαριστά, ένα Big Switch περιέχει αλγορίθμους δρομολόγησης της κίνησης στο εσωτερικό του, καθώς και ένα χάρτη των διαδρομών που το αποτελούν.

Η κλάση Port

Οι πόρτες του κάθε switch αναπαριστούνται στην Port κλάση και αποθηκεύονται σε μία δομή (portMap) στο εσωτερικό της κλάσης Switch. Οι πόρτες των switches

χαρακτηρίζονται ως edge ports (πόρτες στην άκρη ενός δικτύου) αν δεν συνδέεται πάνω τους κάποιο link.

PhysicalPort (extends Port)

Η PhysicalPort είναι η αναπαράσταση μίας πόρτας πάνω σε ένα φυσικό switch. Διατηρεί την αντιστοίχιση μεταξύ της φυσικής με την εικονική πόρτα. Μία φυσική πόρτα μπορεί να αναπαρασταθεί με μία το πολύ πόρτα ανά εικονικό δίκτυο.

OVXPort (extends Port)

Αποτελεί την αναπαράσταση μιας εικονικής πόρτας πάνω σε ένα OVXSwitch

Links and Routes

Ο OVX αναπαριστά τη σύνδεση των Switches μέσω Links. Τα Links καθορίζονται από δύο τελικά σημεία, ένα source Switch/Port ζεύγος και ένα destination Switch/Port ζεύγος.

PhysicalLink (extends Link)

Ένα PhysicalLink συνδέει δύο PhysicalSwitches μέσω PhysicalPorts και ακολουθεί μία 1:1 αντιστοίχιση με τα links του φυσικού δικτύου. Ο OVX βρίσκει τα PhysicalLinks μέσω της διαδικασίας αναγνώρισης της τοπολογίας (topology discovery) κατά τη διάρκεια λειτουργίας του.

OVXLink (extends Link)

Αντίστοιχα με το PhysicalLink, ένα OVXLink συνδέει δύο OVXPorts πάνω σε δύο OVXSwitches που ανήκουν στον ίδιο ένοικο. Ένα OVXLink μπορεί να αντιστοιχίζεται είτε με ένα PhysicalLink, είτε με πολλαπλά PhysicalLinks και τα αντίστοιχα ενδιάμεσα PhysicalSwitches. Σε κάθε περίπτωση, στον controller του ενοίκου εμφανίζεται σαν ένα απλό link.

SwitchRoute (extends Link)

Το SwitchRoute συνδέει δύο OVXPorts που ανήκουν στο ίδιο OVXBigSwitch, καθορίζοντας μία διαδρομή στο εσωτερικό του. Χρησιμοποιούν είτε ένα , είτε περισσότερα PhysicalLinks και χαρακτηρίζονται από δύο ειδών σημεία:

- BVS ingress/egress port: OVXPorts οι οποίες είναι ορατές στον ένοικο σαν πόρτες του Big Switch
- SwitchRoute endpoints: PhysicalPorts στο εσωτερικό του BVS, πάνω στις οποίες συνδέονται τα εσωτερικά PhysicalLinks

Hosts

Οι hosts αντιπροσωπεύουν τελικά σημεία στα δίκτυα των ενοίκων. Καθορίζονται από διευθύνσεις (MAC, IP) και από μία μοναδική ID. Για τον OpenVirteX, κάθε host αποτελεί ένα εικονικό κατασκεύασμα και συνεπώς δεν διαχωρίζεται από φυσική και εικονική αναπαράσταση. Η φυσική υπόσταση ενός host καθορίζεται από το σημείο σύνδεσης του στο δίκτυο και τις δικτυακές του διευθύνσεις. Κάθε host μπορεί να ενσωματώνεται σε ένα μόνο εικονικό δίκτυο.

IP Addresses

Οι IP addresses αποτελούν μεταβλητές που συσχετίζονται με τους hosts του δικτύου. Ο OVX χρησιμοποιεί δύο ειδών IP διευθύνσεις:

- OVXIPAddress: Αποτελεί τη διεύθυνση IP ενός host στο εικονικό δίκτυο, καθορίζεται από τον ένοικο του δικτύου, αποτελεί την IP διεύθυνση που αναγνωρίζει ο controller του δικτύου και είναι μοναδική σε όλο το εικονικό δίκτυο.
- PhysicalIPAddress: Καθορίζεται από τον OVX, αποτελεί την διεύθυνση IP ενός host στο φυσικό δίκτυο και είναι μοναδική σε όλη τη φυσική τοπολογία. Χρησιμοποιείται για να διαφοροποιηθεί η κίνηση μεταξύ των ενοίκων στο εσωτερικό της φυσικής δικτυακής υποδομής.

Network

Η κλάση Network αποθηκεύει χαρτογραφήσεις που περιγράφουν τις σχέσεις μεταξύ των παραπάνω κλάσεων σε ένα δίκτυο. Υπάρχουν δύο ειδών αναπαραστάσεις που προκύπτουν από την κλάση αυτή:

PhysicalNetwork

Η κλάση αυτή αποτελεί την εικόνα που διατηρεί ο OVX για τη φυσική τοπολογία. Ιδανικά αντιστοιχεί σε μία 1 προς 1 αναπαράσταση την δικτυακής υποδομής και της κατάστασης της.

OVXNetwork

Η κλάση αυτή αποτελεί την αναπαράσταση του δικτύου, όπως παρουσιάζεται στον κάθε ένοικο. Ο OVX διατηρεί διαφορετικό OVXNetwork instance για κάθε εικονικό δίκτυο, το οποίο αποθηκεύει τα χαρακτηριστικά του και χαρακτηρίζεται από ένα μοναδικό tenant ID.

2.4.3.3 Topology Virtualization

Ο OVX επιτρέπει στους ενοίκους του δικτύου του να καθορίζουν οι ίδιοι την επιθυμητή τοπολογία του εικονικού τους δικτύου. Η τοπολογία αυτή μπορεί να είναι απλή, π.χ. ένα εικονικό Big Switch που περικλείει όλο το δίκτυο, ή πιο σύνθετη, π.χ. ένα δίκτυο με πολλαπλές back up διαδρομές για αντιμετώπιση σφαλμάτων. Οι τοπολογίες αυτές δεν υποχρεούνται να ταυτίζονται με την υποκείμενη φυσική τοπολογία. Ο μόνος περιορισμός που εισάγει ο OVX είναι ότι ένα φυσικό switch δεν αντιστοιχίζεται σε περισσότερα από ένα εικονικά switches. Όπως αναφέρθηκε προηγουμένως, η αντίστροφη αντιστοίχιση είναι υλοποιήσιμη (Big Switch) και σε συνδυασμό με τη δυνατότητα δημιουργίας πολύπλοκων OVXLinks προσφέρει ιδιαίτερη ευελιξία στο σχεδιασμό της επιθυμητής τοπολογίας κάθε ενοίκου.

2.4.3.4 Control Function Virtualization

Κάθε εικονικό δίκτυο χρησιμοποιεί το δικό του δικτυακό λειτουργικό σύστημα (NOS) για να ρυθμίζει, να παρακολουθεί και γενικότερα να λειτουργεί τα εικονικά του switches. Ο OVX είναι υπεύθυνος για την ρύθμιση των διαφόρων εφαρμογών ελέγχου του εικονικού δικτύου ώστε να έχουν τις επιθυμητές λειτουργίες πάνω στο φυσικό δίκτυο. Σε πολλές περιπτώσεις, ο OVX αναγκάζεται να μεταφράσει μία απλή διαδικασία ελέγχου που απευθύνεται σε ένα εικονικό switch σε πολλαπλές εντολές πάνω στο φυσικό control plane. Για να επιτύχει τη ρύθμιση αυτή, ο OVX εκμεταλλεύεται την ενδιάμεση θέση του ανάμεσα στο εικονικό control plane και στην φυσική υποδομή. Λειτουργεί δηλαδή σαν proxy controller, απορροφά πακέτα ελέγχου των εικονικών δικτύων από τους διάφορους controllers και τα μετατρέπει έτσι ώστε να είναι συμβατά με το control plane του φυσικού δικτύου. Αυτό δίνει την

«ψευδαίσθηση» στον κάθε controller ότι είναι το μοναδικό λειτουργικό σύστημα πάνω στην δικτυακή τοπολογία

2.4.3.5 Address Virtualization

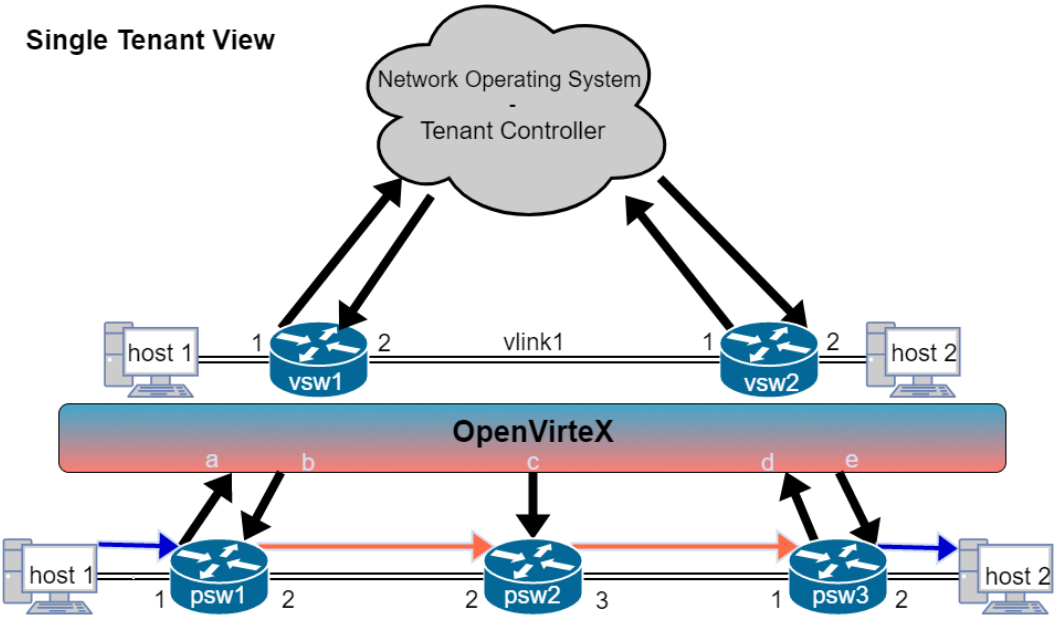
Ο OpenVirteX επιτρέπει στους ενοίκους του δικτύου του να καθορίσουν οι ίδιοι τις δικτυακές διευθύνσεις των τελικών τους hosts, επιτρέποντας πιθανές επικαλύψεις στα σύνολα των διευθύνσεων ανάμεσα σε διαφορετικά εικονικά δίκτυα. Με σκοπό τη διαφοροποίηση των hosts, ο OVX δημιουργεί για κάθε ένοικο μία μοναδική ID και για τους διαφόρους hosts μοναδικές φυσικές IP διευθύνσεις (PhysicalIPAddress). Οι φυσικές αυτές IPs περιέχουν στο εσωτερικό τους τη μοναδική ID του ενοίκου στον οποίο ανήκουν, συνεπώς μέσω αυτών μπορεί να ταυτοποιηθεί για κάθε πακέτο κίνησης το εικονικό δίκτυο στο οποίο ανήκει.

Κατά την προώθηση δικτυακών κανόνων, πραγματοποιείται από τον OVX επανεγγραφή IP διευθύνσεων για κανόνες επιπέδου 3 και MAC διευθύνσεων για κανόνες επιπέδου 2. Συνεπώς υποστηρίζεται virtualization σε επίπεδο δικτύου (layer 3) και σε επίπεδο ζεύξης (layer 2). Με αυτό τον τρόπο, συναντούνται δύο διαφορετικά είδη διευθύνσεων IP και δύο διαφορετικά είδη διευθύνσεων MAC για κάθε πακέτο κίνησης που παράγει κάποιος host, ανάλογα με το σημείο στο οποίο βρίσκεται το πακέτο στο δίκτυο.

- **OVXIPAddress, OVXMACAddress:** διευθύνσεις που αντιλαμβάνεται ο controller και ο ένοικος
- **PhysicalIPAddress, PhysicalMACAddress:** διευθύνσεις που χρησιμοποιεί ο OVX στο φυσικό επίπεδο για να διαφοροποιήσει την κίνηση μεταξύ των ενοίκων

Για να επιτευχθεί απομόνωση δικτυακής κίνησης, ο OVX είναι υπεύθυνος να τροποποιήσει τις διευθύνσεις στα πακέτα των host, μόλις αυτά εισέλθουν στο εσωτερικό του φυσικού δικτύου. Για να το πετύχει αυτό εισάγει κανόνες ροής (flow rules) στα edge switches του δικτύου, τα switches δηλαδή που είναι συνδεδεμένα με τους host που αποστέλλουν ή παραλαμβάνουν κίνηση, οι οποίοι μετατρέπουν τις OVX διευθύνσεις (μοναδικές στο κάθε εικονικό δίκτυο) σε Physical (μοναδικές σε όλο το φυσικό δίκτυο). Αντίστοιχα, όταν τα πακέτα αυτά φτάσουν στο switch πάνω

στο οποίο είναι συνδεδεμένος ο host αποδέκτης, εισάγονται flow rules για την αντίστροφη μετατροπή (Physical -> OVX).



a) Το PacketIn μεταφέρεται στον controller χωρίς τροποποιήσεις
b) Ένα FlowMod μετατρέπει τα OVX δεδομένα του πακέτου στα αντίστοιχα Physical
c) Το vlink1 μεταφράζεται στην αντίστοιχη διαδρομή που το αποτελεί
d) Οι τιμές του PacketIn μεταφράζονται στις αντίστοιχες OVX τιμές που αναγνωρίζει ο controller
e) Ένα FlowMod μεταφράζει τα Physical δεδομένα του πακέτου στα OVX που έχουν οριστεί από τον ένοικο

Σχήμα 2.12 Διαδικασία εικονικοποίησης διευθύνσεων τριών switches

Αξίζει να σημειωθεί ότι η παραπάνω διαδικασία δεν είναι αντιληπτή από τον controller του εικονικού δικτύου. Στο εσωτερικό του φυσικού δικτύου, η επικοινωνία των switches με τον controller γίνεται μέσω του OVX, ο οποίος τροποποιεί τη διευθυνσιοδότηση από και προς τα switches, προκειμένου να μην επηρεάζεται το control plane των ενοίκων από τις εικονικές διευθύνσεων.

2.4.3.6 API

Ο OpenVirteX καθορίζει δύο ειδών API, το Monitoring API και το Tenant API. Το πρώτο χρησιμοποιείται για την άντληση πληροφοριών σχετικά με τη διαμόρφωση και την κατάσταση του δικτύου ενώ το δεύτερο χρησιμοποιείται για τη δημιουργία και την τροποποίηση εικονικών δικτύων.

2.4.3.7 Monitoring API

Method Name	Method Description
getPhysicalTopology	Επέστρεφεί την φυσική τοπολογία
listVirtualNetworks	Επιστρέφει λίστα με όλα τα εικονικά δίκτυα
getVirtualTopology	Επιστρέφει μία συγκεκριμένη εικονική τοπολογία
getVirtualSwitchMapping	Επιστρέφει αντιστοιχία physical-virtual switches για ένα εικονικό δίκτυο
getVirtualLinkMapping	Επιστρέφει αντιστοιχία physical links – virtual link
getVirtualHosts	Επιστρέφει το σύνολο των hosts ενός εικονικού δικτύου (περιγράφονται από τις εικονικές διευθύνσεις τους)
getPhysicalSwitchPorts	Επιστρέφει της φυσικές πόρτες ενός switch
getPhysicalHosts	Επιστρέφει το σύνολο των hosts του δικτύου(περιγράφονται από τις φυσικές διευθύνσεις τους)
getSubnet	Επιστρέφει την Address/Mask ενός εικονικού δικτύου
getVirtualFlowtable	Επιστρέφει το flow table ενός OVXSwitch
getPhysicalFlowtable	Επιστρέφει το flow table ενός PhysicalSwitch
getVirtualAddressMapping	Επιστρέφει την αντιστοιχία OVX-Physical διευθύνσεων για ένα εικονικό δίκτυο
getVirtualSwitchPorts	Επιστρέφει όλες τις πόρτες ενός εικονικου switch

Πίνακας 2.3 Σύνολο μεθόδων του Monitoring API

2.4.3.8 Tenant API

Name	Description
addControllers	Προσθέτει ένα controller για σύνδεση σε ένα OVXSwitch
removeControllers	Απομακρύνει ένα controller από τη λίστα των διαθέσιμών controllers που μπορεί να συνδεθεί ένα switch
createNetwork	Δημιουργεί ένα νέο εικονικό δίκτυο

createSwitch	Δημιουργεί ένα νέο OVXSwitch
createPort	Δημιουργεί μία εικονική πόρτα σε ένα OVXSwitch
SetOVXBigSwitchRouting	Διαμορφώνει τον αλγόριθμο δρομολόγησης στο εσωτερικό ενός BVS
connectHost	Συνδέει ένα host σε ένα εικονικό δίκτυο
connectLink	Δημιουργεί μία νέα εικονική γραμμή
setLinkPath	Προσθέτει μία φυσική διαδρομή σε ένα εικονικό link
connectRoute	Δημιουργεί μία νέα διαδρομή στο εσωτερικό ενός BVS
removeNetwork	Διαγράφει ένα εικονικό δίκτυο
removeSwitch	Διαγράφει ένα OVXSwitch από ένα εικονικό δίκτυο
removePort	Διαγράφει μία εικονική πόρτα από ένα OVXSwitch
disconnectHost	Αποσυνδέει ένα host από ένα εικονικό δίκτυο
disconnectLink	Διαγράφει μία εικονική γραμμή
disconnectRoute	Διαγράφει μία εσωτερική διαδρομή σε ένα Big Virtual Switch
startNetwork	Θέτει σε λειτουργία ένα εικονικό δίκτυο
startSwitch	Θέτει σε λειτουργία ένα εικονικό switch
startPort	Ενεργοποιεί μία εικονική πόρτα
stopNetwork	Σταματάει προσωρινά τη λειτουργία ενός εικονικού δικτύου
stopSwitch	Σταματάει προσωρινά τη λειτουργία ενός OVXSwitch
stopPort	Απενεργοποιεί προσωρινά μία εικονική πόρτα

Πίνακας 2.4 Σύνολο μεθόδων για τη δημιουργία εικονικών δικτύων

2.4.3.9 Features

Η δομή του OpenVirteX, με την ευέλικτη αντιστοίχιση φυσικών και εικονικών στοιχείων (Big Switches, σύνθετα Virtual Links) προσδίδει τα παρακάτω χαρακτηριστικά στα εικονικά SDN δίκτυα:

- Ευέλικτη προσαρμογή τοπολογίας: Οι εικονικές τοπολογίες που δημιουργούνται δεν αποτελούν αυστηρό αντίγραφο της φυσικής υποδομής, ούτε περιορίζονται σε υποσύνολα αυτής. Τα σύνθετα Virtual Links και τα Big Switches προσφέρουν ένα επιπλέον επίπεδο αφαιρετικότητας στη δημιουργία εικονικών δικτύων επιτρέποντας την διαμόρφωση της τοπολογίας στα επιθυμητά πρότυπα του ενοίκου.
- Ανθεκτικότητα: Ένα εικονικό switch ή link έχει τη δυνατότητα να αντιστοιχιστεί σε πολλαπλά φυσικά στοιχεία του δικτύου. Ένα τέτοιο εικονικό link μπορεί να περιέχει πολλαπλές διαδρομές μεταξύ των σημείων που ενώνει. Ακόμη, ένα Big Switch περιέχει πολλαπλά φυσικά switches και

links στο εσωτερικό του δημιουργώντας ποικίλες συνδέσεις μεταξύ των πορτών του. Δημιουργούνται συνεπώς back up paths για την γρήγορη αντιμετώπιση σφαλμάτων στην φυσική τοπολογία.

- Δυναμική τροποποίηση vSDN: Η χαρτογράφηση των διαφόρων εικονικών – φυσικών οντοτήτων του δικτύου είναι κεντρικοποιημένη και καθορίζεται από λογισμικό. Τα εικονικά στοιχεία ενός δικτύου αποτελούν απλώς δείκτες πάνω στη φυσική τοπολογία. Για το λόγο αυτό απλοποιείται σε μεγάλο βαθμό η διαμόρφωση του κάθε δικτύου, ενώ είναι δυνατή και η δυναμική (κατά τη διάρκεια της λειτουργίας του) επαναδιαμόρφωση του.
- Διατήρηση κατάστασης: Κάθε εικονικό δίκτυο αποτελείται από χαρακτηριστικά και ιδιότητες που, καθώς καθορίζονται από λογισμικό, μπορούν να αποθηκευτούν και να διατηρηθούν. Είναι δηλαδή εύκολη η απενεργοποίηση ενός εικονικού δικτύου και στη συνέχεια η επαναφορά του στην ίδια κατάσταση. Το χαρακτηριστικό αυτό φέρνει πιο κοντά τη δυνατότητα δημιουργίας δικτυακών snapshots.

2.5 Network Monitoring

Η «παρακολούθηση δικτύων» αναφέρεται σε διαδικασίες που ανήκουν στο επίπεδο διαχείρισης (management plane) ενός δικτύου. Οι διαχειριστές απαιτείται να συλλέγουν διαρκώς δεδομένα δικτυακής κίνησης και να τα τροφοδοτούν σε διάφορες εφαρμογές διαχείρισης του δικτύου τους. Παραδείγματα χρήσης τέτοιων εφαρμογών είναι τα εξής:

- Ανίχνευση, διάγνωση και επισκευή διαδικτυακών προβλημάτων
- Διαχείριση συμφόρησης γραμμών σε πραγματικό χρόνο
- Ανάλυση της δικτυακής κίνηση για την ανίχνευση μη εξουσιοδοτημένων ενεργειών και τον εντοπισμό πηγών DOS (Denial-Of-Service) επιθέσεων
- Ομαδοποίηση και βελτιστοποίηση διαδρομών
- Traffic Engineering

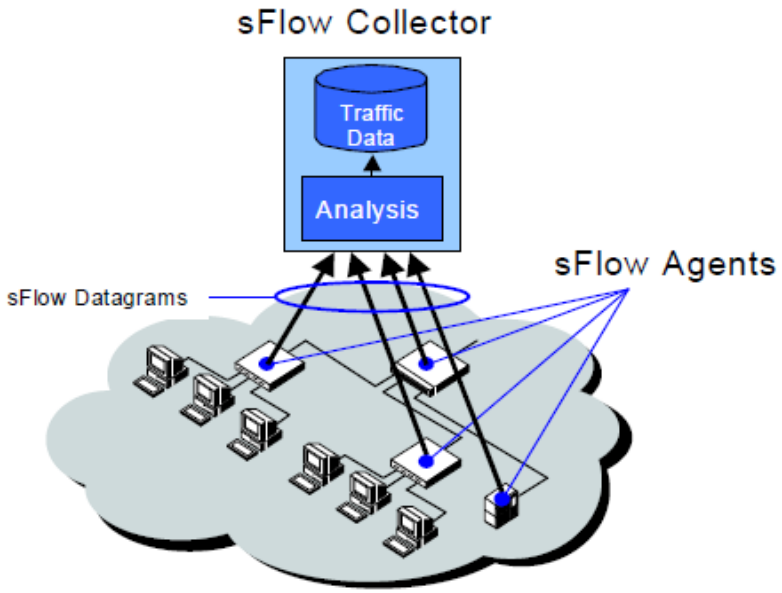
Δημιουργούνται συνεπώς αυξημένες ανάγκες για συνεχόμενη άντληση λεπτομερών μετρήσεων της δικτυακής κίνησης.

Παρόλα αυτά, στα μοντέρνα δίκτυα όπου το εύρος της δικτυακής κίνησης είναι ιδιαίτερα αυξημένο, τεχνολογικοί περιορισμοί και περιορισμοί πόρων στα δικτυακά μηχανήματα (routers, switches) δεν επιτρέπουν την συνολική καταγραφή των πακέτων ή των ροών που αυτά διαχειρίζονται. Για το λόγο αυτό, εισάγεται η έννοια της δειγματοληψίας (sampling). Τα switches βασίζονται σε τεχνικές δειγματοληψίας

για να καταγράψουν ένα αντιπροσωπευτικό τμήμα της υποκείμενης δικτυακής τους κίνησης, σε αντιστοιχία πάντα και με τις υπολογιστικές και χωρητικές τους ικανότητες. Για παράδειγμα, τα περισσότερα σύγχρονα routers εκτελούν ομοιόμορφη δειγματοληψία πακέτων με χρήση κάποιας δειγματοληπτικής πιθανότητας (από 0.001 έως 0.01). Με τη χρήση του sampling διατηρείτε η εγκυρότητα της εκτίμησης του όγκου δικτυακής κίνησης και η ακρίβεια των εφαρμογών του management plane με ελάχιστη χρήση δικτυακών πόρων.

2.5.1 sFlow

Το sFlow αποτελεί μία τεχνολογία δειγματοληψίας η οποία είναι ενσωματωμένη στα switches ενός δικτύου. Παρέχει συνεχή παρακολούθηση των ροών (flows) κίνησης του δικτύου μέσω δειγματοληψίας πακέτων. Υποστηρίζεται από τις δικτυακές συσκευές πολλών κατασκευαστών και εξάγει δεδομένα που είναι αναγνωρίσιμα από ποικίλες εφαρμογές διαχείρισης και παρακολούθησης.



Σχήμα 2.13 Αρχιτεκτονική Συστήματος sFlow

Για την άντληση των δειγμάτων (flow samples) ένα sFlow σύστημα αποτελείται από δύο συνιστώσες:

- Ένα σύνολο από «sFlow agents» που βρίσκονται στο εσωτερικό των δικτυακών μηχανών (switches, routers), οι οποίοι συλλέγουν και αποστέλλουν τα samples
- Έναν (ή περισσότερους) κεντρικοποιημένο συλλέκτη (sFlow collector) ο οποίος δέχεται και επεξεργάζεται τα δείγματα

2.5.1.1 sFlow agent

Ο sFlow agent είναι μία λογισμική οντότητα που αποτελεί κομμάτι του δικτυακού λογισμικού διαχείρισης στο εσωτερικό μίας συσκευής δικτύωσης. Σκοπός της είναι να συλλέγει τα flow samples καθώς και μετρητές από τα interfaces της συσκευής και συνδυάζοντας τα να δημιουργεί sFlow δεδομενογράμματα (datagrams). Στη συνέχεια, αποστέλλει τα datagrams αυτά σε κάποιο συγκεκριμένο sFlow Collector.

Ο sFlow agent χρησιμοποιεί ελάχιστους δικτυακούς πόρους. Η διαδικασία της δειγματοληψίας πραγματοποιείται από την ίδια τη συσκευή δικτύωσης. Ο agent απλώς «πακετάρει» τα samples σε datagrams και τα αποστέλλει άμεσα στο δίκτυο. Η άμεση αυτή προώθηση των δειγμάτων ελαχιστοποιεί τις απαιτήσεις του sFlow συστήματος σε υπολογιστικούς και αποθηκευτικούς πόρους.

2.5.1.2 sFlow properties

Η χρήση του sFlow συστήματος για την παρακολούθηση SDN δικτύων έχει τα εξής χαρακτηριστικά:

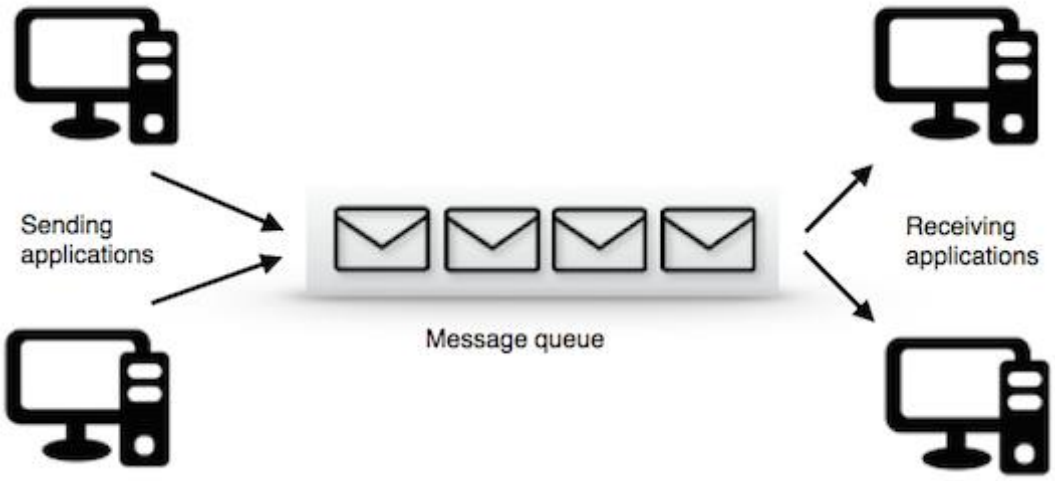
- Ακρίβεια: Η δειγματοληψία είναι αρκετά απλοποιημένη ώστε να εκτελείται σε υψηλές ταχύτητες. Ακόμη, το σύστημα είναι σχεδιασμένο ώστε να μπορεί να καθοριστεί η επιθυμητή ακρίβεια των μετρήσεων.
- Λεπτομερής Ανάλυση: Η ολοκληρωτική δειγματοληψία της κεφαλίδας των πακέτων και των πληροφοριών δρομολόγησης επιτρέπει την λεπτομερή ανάλυση των ροών κίνησης
- Κλιμακωτή Αρχιτεκτονική: Λόγω της δομής του (ύπαρξη πολλαπλών sflow agent που επικοινωνούν είτε με ένα, είτε με περισσότερους sflow collectors) είναι εφικτή η κλιμακωτή ανάπτυξη του συστήματος.
- Χαμηλό Κόστος: Οι sflow agent έχουν εύκολη υλοποίηση και προσθέτουν ελάχιστο επιπλέον κόστος σε μία συσκευή δικτύωσης.
- Έγκαιρη παρακολούθηση: Ο sFlow collector έχει μία συνεχώς ανανεωμένη εικόνα για το σύνολο του δικτύου. Η χρονική ακρίβεια είναι

σημαντική για εφαρμογές που παρέχουν υπηρεσίες σε πραγματικό χρόνο.

2.6 Message Queue

Η χρήση μιας ουράς μηνυμάτων επιτρέπει την επικοινωνία και την ανταλλαγή μηνυμάτων μεταξύ εφαρμογών. Στην ουσία αποτελεί ένα προσωρινό αποθηκευτικό χώρο για μηνύματα των οποίων ο παραλήπτης δεν είναι άμεσα διαθέσιμος.

Οι ουρές μηνυμάτων παρέχουν ένα πρωτόκολλο ασύγχρονης επικοινωνίας καθώς ένα σύστημα το οποίο δημοσιεύει μηνύματα σε μία ουρά μηνυμάτων δεν απαιτεί άμεση απόκριση προκειμένου να συνεχίσει της διεργασίες του. Ως αποτέλεσμα, αυτός ο τρόπος επικοινωνίας αποσυνδέει τα συστήματα που επικοινωνούν, διότι δεν απαιτείται να αλληλεπιδρούν με την ουρά μηνυμάτων ταυτόχρονα.



Σχήμα 2.14 Εικονική αναπαράσταση ουράς μηνυμάτων

Η βασική αρχιτεκτονική μιας ουράς μηνυμάτων είναι αρκετά απλή. Χρησιμοποιούνται εφαρμογές “client”, οι οποίες ονομάζονται “producers”, για τη δημιουργία μηνυμάτων και την αποστολή τους στην ουρά. Μία άλλη εφαρμογή, που ονομάζεται “consumer” συνδέεται στην ουρά προκειμένου να αντλήσει τα μηνύματα για να επεξεργαστούν. Τα μηνύματα που εισέρχονται στην ουρά αποθηκεύονται έως ότου να αντληθούν από κάποιο consumer.

2.6.1 *Kafka*

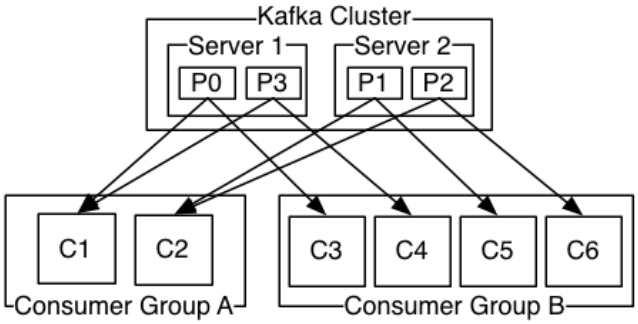
Το Kafka αποτελεί μία πλατφόρμα ροής δεδομένων λειτουργώντας σαν σύμπλεγμα ενός ή περισσότερων servers. Αποθηκεύει ροές καταχωρήσεων δεδομένων σε κατηγορίες, τις οποίες ονομάζει topics. Κάθε κατηγορία αποτελείται από ένα μοναδικό κλειδί, μία τιμή και μία μεταβλητή χρόνου δημιουργίας (timestamp).

Η πλατφόρμα αυτή έχει τέσσερις διεπαφές εφαρμογής (API):

- Producer API: Επιτρέπει σε μία εφαρμογή να εκδίδει μία ροή δεδομένων σε ένα ή περισσότερα topics.
- Consumer API: Επιτρέπει σε μία εφαρμογή να κάνει εγγραφή σε ένα ή περισσότερα topics προκειμένου να λαμβάνει και να επεξεργάζεται ροές δεδομένων από αυτά.
- Streams API: Επιτρέπει σε μία εφαρμογή να απορροφά μία ροή από ένα ή περισσότερα topics, να την επεξεργάζεται και να την δημοσιεύει σαν ένα νέο topic.
- Connector API: Επιτρέπει την κατασκευή producers και consumers οι οποίοι συνδέουν topics με εφαρμογές ή συστήματα δεδομένων.

2.6.1.1 *Kafka as a Messaging System*

Με τη χρήση των παραπάνω διεπαφών είναι δυνατή η χρήση του Kafka σαν ένα σύστημα διανομής μηνυμάτων και δεδομένων. Οι εφαρμογές που θέλουν να αποστείλουν μηνύματα ή δεδομένα τα εκδίδουν σε κάποιο topic και αντίστοιχα οι εφαρμογές-αποδέκτες των μηνυμάτων αυτών εγγράφονται στο topic αυτό για να τα παραλάβουν.



Σχήμα 2.15 Παράδειγμα ενός Kafka cluster

Το πλεονέκτημα του Kafka σε σύγκριση με παρόμοια συστήματα είναι η ανοχή σε σφάλματα (fault-tolerance) και η επεκτασιμότητα του (scalability). Τα μηνύματα που εκδίδονται σε ένα topic διατηρούνται στο σύμπλεγμα των server του Kafka προκειμένου να επαναμεταδοθούν με ασφάλεια σε περίπτωση κάποιου σφάλματος. Ακόμα, καθώς η υλοποίηση αποτελείται από ένα σύμπλεγμα από servers αλλά και από πολλαπλούς producers και consumers σε κάθε topic, είναι εύκολη η κατανομή του φόρτου εργασίας σε ένα σύστημα που το χρησιμοποιεί αλλά και η επεκτασιμότητα του συστήματος σε μεγαλύτερες κλίμακες.

2.6.2 RabbitMQ

Το RabbitMQ είναι ένα λογισμικό ουρών μηνυμάτων, το οποίο αποτελεί ένα message broker ή αλλιώς message manager. Στην ουσία μέσω του λογισμικού αυτού δημιουργούνται ουρές μηνυμάτων, ενώ μέσω των διεπαφών που προσφέρει επιτρέπει σε εφαρμογές να συνδέονται στις ουρές αυτές και να μεταδίδουν μηνύματα.

2.6.2.1 Exchange

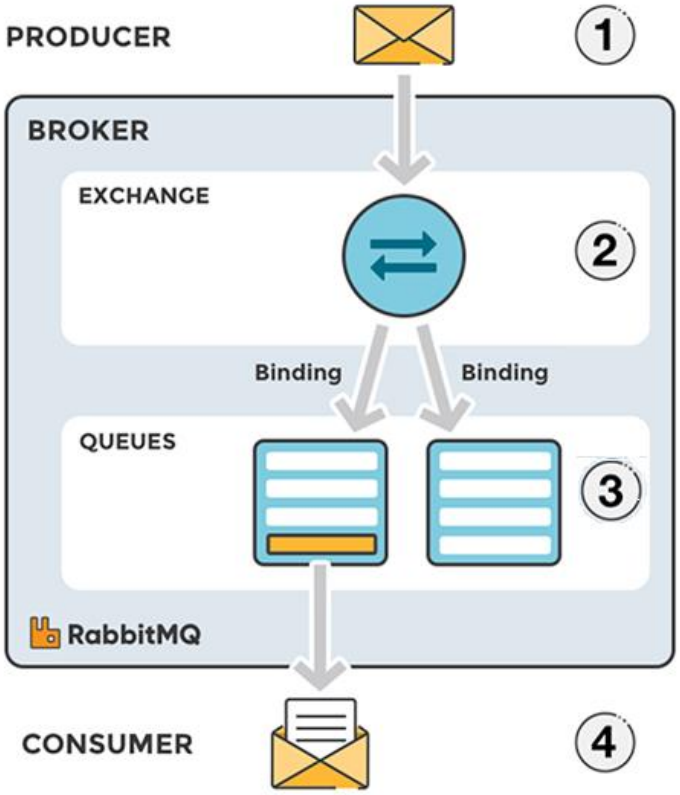
Στις ουρές μηνυμάτων που δημιουργούνται οι διάφοροι consumers δεν δημοσιεύουν μηνύματα απευθείας στην ουρά, αντιθέτως τα αποστέλλουν αρχικά σε κάποιο exchange. Τα exchanges είναι agents, οριζόμενοι από το RabbitMQ, υπεύθυνοι για τη δρομολόγηση των μηνυμάτων στις διάφορες ουρές. Υπάρχουν τέσσερα είδη exchange, τα οποία χρησιμοποιούν διαφορετικούς τρόπους δρομολόγησης, μέσω συγκεκριμένων παραμέτρων και χαρακτηριστικών των μηνυμάτων. Τέτοιοι παράμετροι είναι τα bindings (link που ορίζεται για να συνδέσει μία ουρά με κάποιο exchange) και τα routing keys (χαρακτηριστικό των μηνυμάτων το οποίο αναζητείται από συγκεκριμένα exchanges για να καθοριστεί η δρομολόγηση του μηνύματος)

Τα είδη exchange είναι τα παρακάτω:

- Direct: Δρομολογεί τα μηνύματα με βάση το routing key

- Fanout: Δρομολογεί τα μηνύματα σε όλες τις ουρές που έχουν συνδεθεί με το συγκεκριμένο exchange
- Topic: Χρησιμοποιεί αντιστοίχιση μεταξύ routing key και routing pattern που καθορίζεται από το binding του συγκεκριμένου exchange
- Headers: Χρησιμοποιεί τα header πεδία του μηνύματος για τη δρομολόγηση

2.6.2.2 Message flow in RabbitMQ



Σχήμα 2.16 Ροή μηνυμάτων στο RabbitMQ

1. Ο producer δημοσιεύει ένα μήνυμα σε συγκεκριμένο exchange
2. Το exchange δρομολογεί το μήνυμα σε κάποια ουρά, με βάση το είδος του και τα χαρακτηριστικά του μηνύματος
3. Το μήνυμα παραμένει στην ουρά έως ότου αντληθεί από κάποιο consumer
4. Ο consumer απορροφά και επεξεργάζεται το μήνυμα

3

Σχεδιαστικές Αρχές

Η συλλογή δεδομένων δικτυακής κίνησης στις σύγχρονες υποδομές δικτύωσης βρίσκει πολλές εφαρμογές. Διαδικασίες συλλογής χρησιμοποιούνται από τους διαχειριστές τέτοιων δικτύων προκειμένου να αυξηθεί η ασφάλεια των δεδομένων τους αλλά και η ποιότητα των υπηρεσιών τους (Quality Of Service, QoS). Επιπλέον, όπως αναφέρθηκε στο Κεφάλαιο 2, σε περιβάλλοντα δικτύων SDN, η δημιουργία ενός εικονικού δικτύου και η διάθεση του σε πολλαπλούς ενοίκους παρέχει πολλές και σημαντικές πρακτικές, είτε στα πλαίσια ανάπτυξης πειραματικών υποδομών είτε με τη μορφή παροχής υπηρεσιών (IaaS).

Στόχος της συγκεκριμένης διπλωματικής εργασίας είναι η ανάπτυξη μηχανισμών συλλογής δικτυακής κίνησης και παρακολούθησης SDN δικτύων σε εικονικά περιβάλλοντα πολλαπλών ενοίκων. Πιο συγκεκριμένα, στόχος είναι η προσαρμογή ενός μηχανισμού παρακολούθησης SDN δικτύων προκειμένου να είναι ικανός να παρακολουθήσει διαφορετικά εικονικά δίκτυα πάνω στην ίδια δικτυακή υποδομή. Απαραίτητος είναι ο διαχωρισμός της δικτυακής κίνησης του κάθε ενοίκου, η δυνατότητα καθολικής παρακολούθησης του δικτύου αλλά και η απομόνωση και παροχή πληροφοριών δικτυακής κίνησης σε κάθε ένοικο ξεχωριστά για το εικονικό δίκτυο που του αντιστοιχεί.

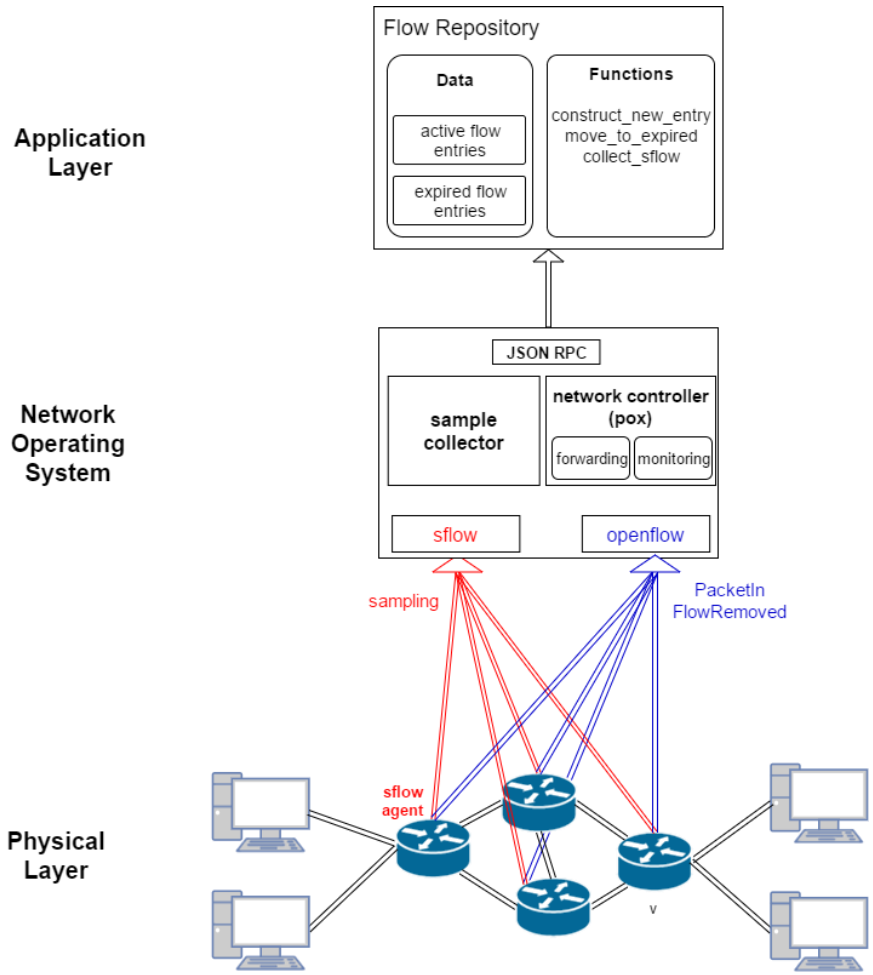
3.1 Λειτουργία Μηχανισμού Παρακολούθησης SDN δικτύων

Η ιδέα πίσω από την υλοποίηση του μηχανισμού παρακολούθησης βασίστηκε στην ιδιαιτερότητα της επικοινωνία των SDN switches με τον controller, και πιο συγκεκριμένα στην ανταλλαγή OpenFlow μηνυμάτων. Αξιοποιώντας τα μηνύματα αυτά είναι δυνατή η δημιουργία ενός αποθηκευτικού μηχανισμού για τις ροές που εισάγονται στα switches (Flow Repository), λαμβάνοντας υπόψη και τους χρόνους εισαγωγής και διαγραφής κάθε ροής. Με τον τρόπο αυτό, οι ροές αποθηκεύονται σε δύο δομές:

- Active flows: Στη δομή αυτή καταχωρούνται τα flow entries των switches που είναι ακόμα σε ισχύ και διατηρούνται εκεί από τη στιγμή της εισαγωγής τους έως ότου διαγραφούν.
- Expired flows: Αμέσως μετά τη διαγραφή ενός flow από το switch, η αντίστοιχη καταχώρηση του στο Flow Repository μεταφέρεται από την active δομή στην expired.

Αξίζει να σημειωθεί ότι οι καταχωρήσεις χαρακτηρίζονται από τους χρόνους εισαγωγής και διαγραφής, συνεπώς δύο ίδιοι κανόνες σε διαφορετικές χρονικές στιγμές αποτελούν διαφορετικές εγγραφές.

Μέσω ενός μηχανισμού δειγματοληψίας (sFlow), γίνεται συλλογή δειγμάτων κατά πακέτα στη φυσική τοπολογία. Τα δείγματα αυτά αρχικά συλλέγονται σε μία εξωτερική εφαρμογή και στη συνέχεια αποστέλλονται στο Flow Repository όπου αντιστοιχίζονται μέσω μίας αλγοριθμικής διαδικασίας στην εγγραφή της ροής που ανήκουν και αυξάνονται καταλλήλως οι αντίστοιχοι μετρητές.



Σχήμα 3.1 Αρχιτεκτονική μηχανισμού παρακολούθησης δικτύων SDN

Συγκεντρωτικά, η λειτουργία του μηχανισμού είναι η εξής:

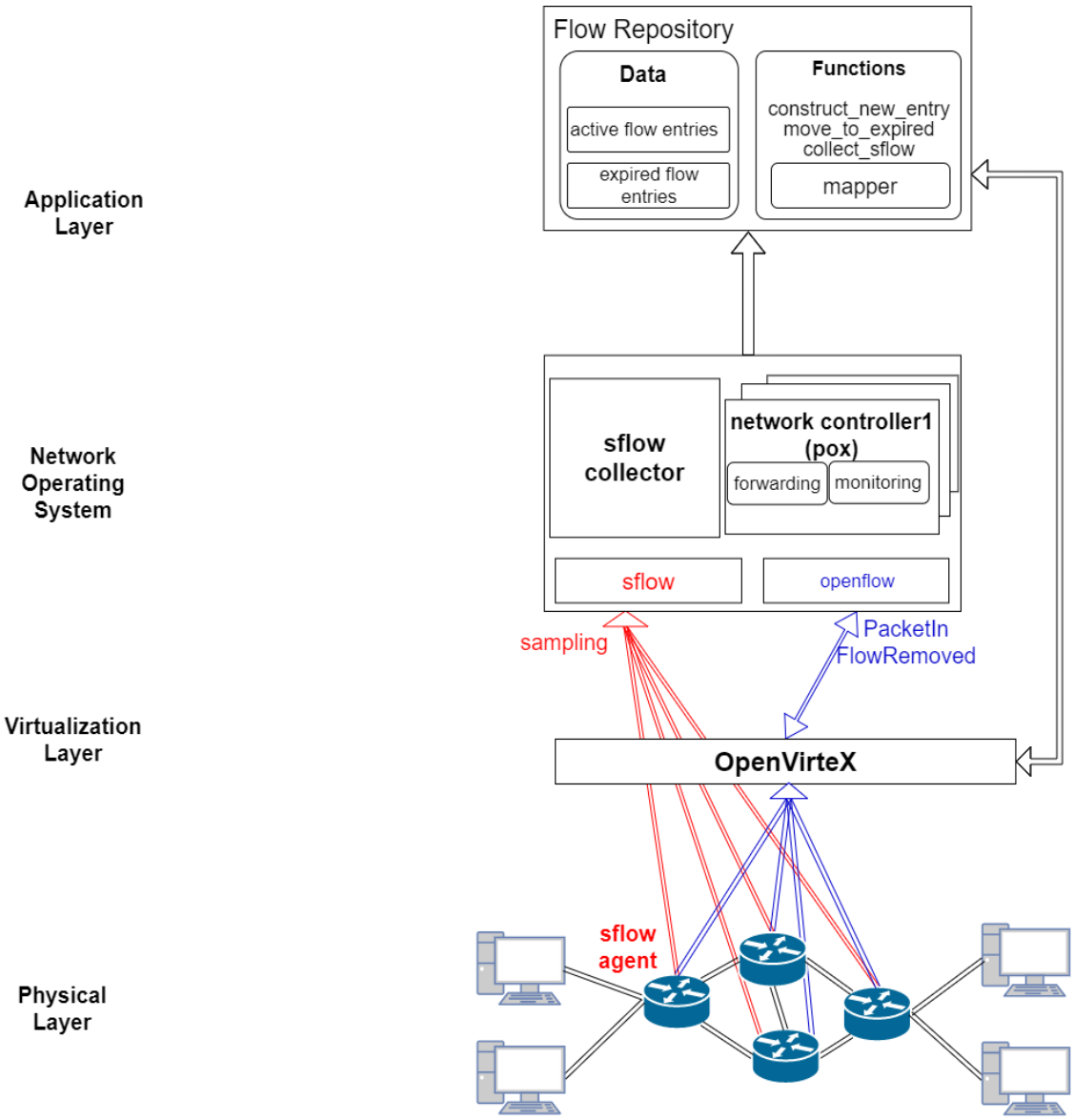
1. Κατά την άφιξη ενός νέου πακέτου, το switch στέλνει ένα PacketIn Event στον controller
2. Ο controller παίρνει αποφάσεις για την δρομολόγηση του πακέτου και τη δημιουργία νέου flow rule. Σε περίπτωση δημιουργίας ενός νέου κανόνα, επικοινωνεί με το Flow Repository για τη δημιουργία μίας νέας active εγγραφής.
3. Πραγματοποιείται συνεχώς δειγματοληψία στα switches του φυσικού επιπέδου. Τα δείγματα αποστέλλονται στο Flow Repository προκειμένου να γίνει αντιστοίχιση τους στις active εγγραφές.
4. Όταν ένας κανόνας εκπνέυσει χρονικά, το switch στο οποίο βρισκόταν στέλνει ένα FlowRemoved Event στον controller. Ο controller με τη σειρά του επικοινωνεί με το Flow Repository για τη μεταφορά της αντίστοιχης καταχώρησης, μαζί με τις μετρήσεις των samples, στην Expired δομή.

3.2 Εισαγωγή Virtualization Layer

Για τους σκοπούς της διπλωματικής εργασίας, απαιτήθηκε η αναπροσαρμογή της αρχιτεκτονικής του παραπάνω μηχανισμού ώστε να είναι συμβατή με εικονικές δικτυακές υποδομές πολλαπλών ενοίκων. Σαν πλατφόρμα διαμοιρασμού δικτυακων πόρων χρησιμοποιήθηκε το OpenVirteX (OVX).

Το OVX είναι ένας hypervisor που επιτρέπει τον ορισμό πολλαπλών εικονικών δικτύων πάνω στην ίδια τοπολογία, καθένα ανεξάρτητο από τα υπόλοιπα και διαχειρίσιμο από τον δικό του ξεχωριστό controller. Μέσω του OVX, κάθε ένοικος έχει τη δυνατότητα απόδοση εξατομικευμένων χαρακτηριστικών στο εικονικό του δίκτυο. Σε αυτό το πλαίσιο, ο OpenVirteX παρέχει virtualization σε επίπεδο τοπολογίας και διευθύνσεων. Επιτρέπει ουσιαστικά σε κάθε ένοικο να προσαρμόσει τη δικτυακή του τοπολογία ανάλογα με τις ανάγκες του και να χρησιμοποιήσει ολόκληρο το address space για τα εικονικά του μηχανήματα, "αδιαφορώντας" για πιθανές επικάλυψης διευθύνσεων με άλλους ενοίκους. Επίσης, επιπλέον χαρακτηριστικά που προσφέρει το OVX στα δίκτυα που ορίζει είναι ανθεκτικότητα σε βλάβες στις δικτυακές γραμμές (**resiliency**), καθώς επίσης και δυνατότητα αποθήκευσης της κατάστασης και δεδομένων ενός εικονικού δικτύου μια συγκεκριμένη στιγμή (**network snapshotting**).

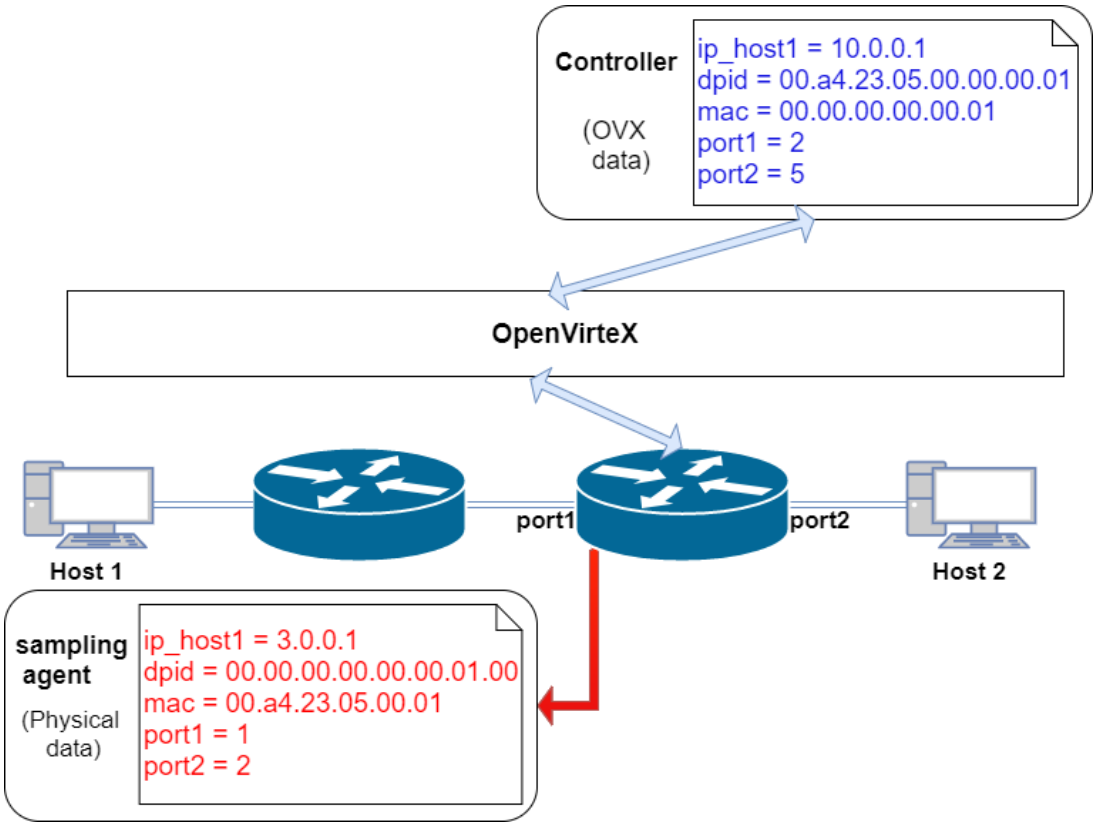
Όπως αναφέρθηκε στο 2^{ο} Κεφάλαιο, το OVX δημιουργεί εικονικές απεικονίσεις των επιμέρους δικτυακών στοιχείων, τις οποίες αντιστοιχίζει με πραγματικά δικτυακά μηχανήματα. Με αυτό τον τρόπο ορίζει εικονικά δίκτυα για τον κάθε ένοικο. Τα διαφορετικά εικονικά δίκτυα διαχωρίζονται μεταξύ τους με τη χρήση ενός μοναδικού κλειδιού, του tenant id (tid). Το συγκεκριμένο κλειδί αποτελεί την ταυτότητα του εικονικού δικτύου και αποδίδεται σε αυτό από το OVX κατά τον αρχικό ορισμό του. Επιπλέον, για την απομόνωση της δικτυακής κίνησης των διαφόρων ενοίκων, το OVX αναλαμβάνει να μεταφράσει τις δικτυακές διευθύνσεις που έχει ορίσει ο κάθε ένοικος (OVX addresses) σε διευθύνσεις μοναδικές σε όλα τα εικονικά δίκτυα (Physical addresses).



Σχήμα 3.2 Αρχιτεκτονική μηχανισμού παρακολούθησης για πλατφόρμες πολλαπλών ενοίκων

Με την ενσωμάτωση του OVX, εισάγεται ένα νέο επίπεδο (virtualization layer) στην προκείμενη υλοποίηση ανάμεσα στη φυσική τοπολογία και τους controllers των ενοίκων. Πρακτικά λειτουργεί ως proxy controller ρυθμίζοντας την επικοινωνία των πολλαπλών control planes με το δίκτυο. Αναλαμβάνει να μεταφράσει τις διευθύνσεις που περιέχουν τα πακέτα των διάφορων hosts, έτσι ώστε να περιέχουν Physical τιμές στο φυσικό επίπεδο. Ταυτόχρονα τροποποιεί τη σηματοδοσία μεταξύ των switches και των controllers, προκειμένου να φτάσει στους δεύτερους το address space που έχουν ορίσει και αντιλαμβάνονται (OVX addresses).

Η παραπάνω διαδικασία του address mapping δημιουργεί ένα συγκεκριμένο πρόβλημα στο μηχανισμό παρακολούθησης. Στο Flow Repository δημιουργούνται εγγραφές ροών με OVX διευθύνσεις, καθώς αυτό το address space χρησιμοποιούν οι διάφοροι controllers. Παράλληλα, η διαδικασία της δειγματοληψίας πραγματοποιείται στο φυσικό επίπεδο, επομένως συλλέγονται δείγματα που μπορεί να περιέχουν φυσικές διευθύνσεις. Είναι συνεπώς αδύνατη η αντιστοίχιση των καταχωρήσεων με τα δείγματα.



Σχήμα 3.3 Παράδειγμα διαφορετικών διευθύνσεων στο φυσικό επίπεδο και στον controller για το ίδιο πακέτο

Ουσιαστικά, ανάλογα με το σημείο του δικτύου στο οποίο πραγματοποιείται δειγματοληψία, ένα πακέτο μπορεί να περιέχει είτε OVX διευθύνσεις, είτε Physical διευθύνσεις, είτε ένα συνδυασμό των δύο. Στον πίνακα 3.1 φαίνεται η διαφορετική διευθυνσιοδότηση που χρησιμοποιεί το OVX για τα δικτυακά πακέτα, ανάλογα με το σημείο του δικτύου που βρίσκονται.

	Source IP	Destination IP	Source MAC	Destination MAC
Σηματοδοσία του OVX	-		OpenVirteX signals	
Edge Switch του δικτύου / Switch εισόδου σε edge BVS	OVX Addresses		OVX Addresses	
Core Switch του δικτύου / Switch εισόδου σε core BVS	Physical Addresses		Physical Addresses	
Υπόλοιπα Switches σε BVS	Physical Addresses		OVX Addresses	

- **Edge switch : switch πάνω στο οποίο συνδέεται ο host που παράγει την κίνηση**
- **Core switch : τα υπόλοιπα switch του δικτύου (όχι edge)**
- **BVS: Big Virtual Switch (Κεφάλαιο 2.4.3.2)**
- **Σηματοδοσία του OVX: αφορά πακέτα που χρησιμοποιεί το OVX για την αναγνώριση της δικτυακής τοπολογίας τα οποία δεν περιέχουν IP field στα headers**

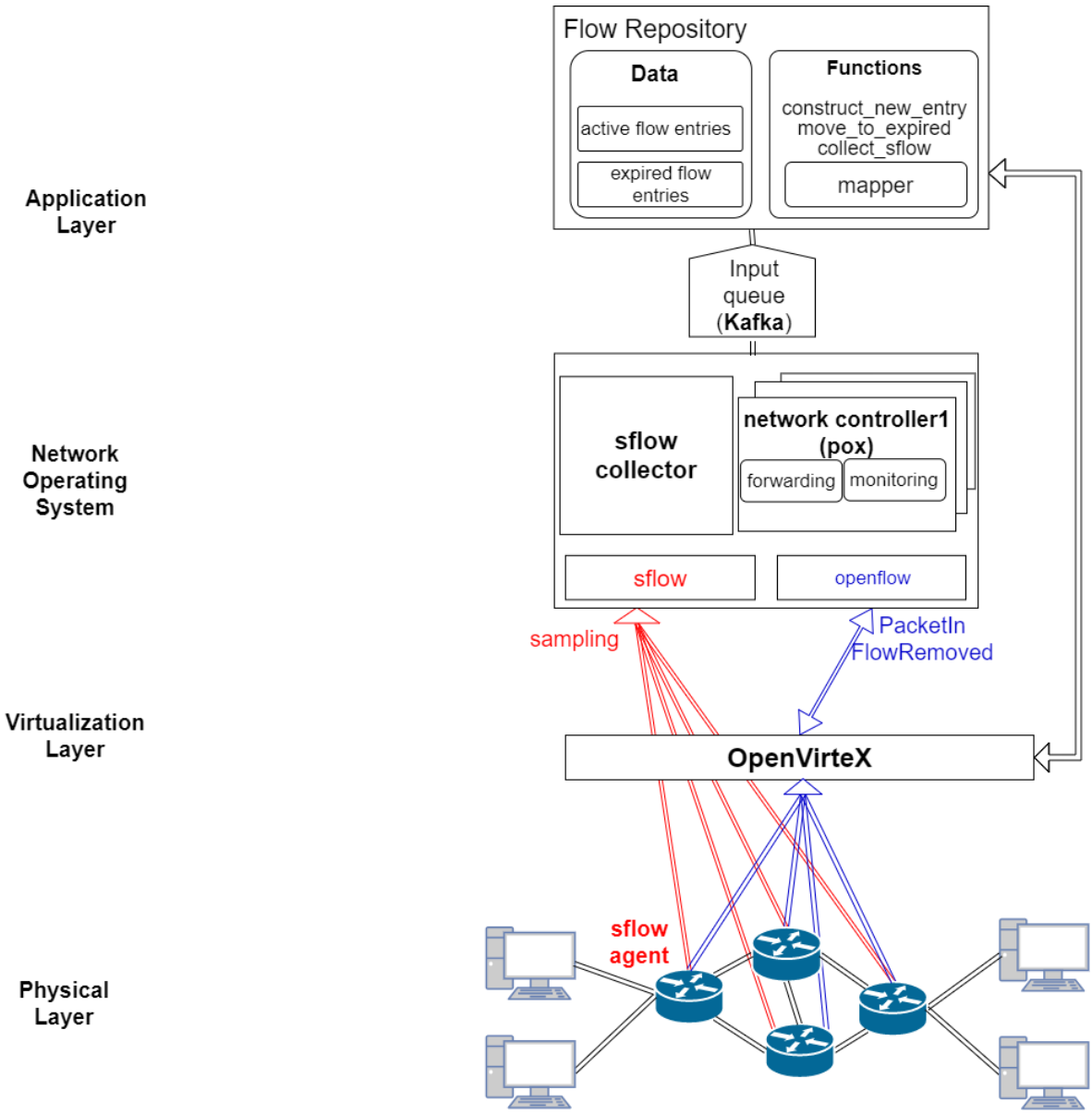
Πίνακας 3.1 Διευθυνσιοδότηση του OpenVirteX στα διαφορετικά σημεία του δικτύου

Για να είναι εφικτή η αντιστοίχιση των δειγμάτων με τις εγγραφές στο Flow Repository, κατασκευάζεται μία ειδική βιβλιοθήκη λειτουργιών (mapper). Η συγκεκριμένη βιβλιοθήκη επιτρέπει την επικοινωνία του Flow Repository με το OVX, την άντληση των αντιστοιχίσεων που περιέχει για τις διευθύνσεις (OVX <-> Physical) και την πραγματοποίηση των απαραίτητων μεταφράσεων σε αυτές. Επιπλέον, για την ομαδοποίηση των εγγραφών που προστίθενται στο Flow Repository ανά tenant, χρησιμοποιείται σαν κλειδί το tenant id. Το συγκεκριμένο κλειδί μπορεί να αντληθεί είτε μέσω του ειδικού API του OVX ή μέσω της ειδικής διευθυνσιοδότησης που αυτό χρησιμοποιεί.

3.3 Kafka σα διεπαφή εισόδου

Ένα σύστημα παρακολούθησης δικτύων απαιτείται να είναι ικανό να διαχειριστεί μεγάλο όγκο δεδομένων σε σύντομο χρονικό διάστημα. Σε ένα virtual περιβάλλον όπου το δίκτυο μοιράζεται από πολλαπλούς ενοίκους, ο όγκος αυτός των δεδομένων αυξάνεται ακόμα περισσότερο. Στη συγκεκριμένη περίπτωση, ο server του συστήματος μας καλείται να διαχειριστεί αιτήσεις από πολλά διαφορετικά

εργαλεία (sample agents, virtual controllers etc). Το πλήθος των ενοίκων και ο ρυθμός δειγματοληψίας στο δίκτυο καθορίζουν τον όγκο των δεδομένων εισόδου που επεξεργάζεται συνεχώς ο server. Προκειμένου να διασφαλιστεί η σωστή λειτουργία του μηχανισμού κάτω από περιπτώσεις αυξημένου αριθμού αιτήσεων, χρησιμοποιείται σα διεπαφή εισόδου δεδομένων ένα κατανεμημένο σύστημα διαχείρισης μηνυμάτων, το Kafka. Το σύστημα αυτό ουσιαστικά εξασφαλίζει τη συλλογή και διανομή μεγάλου όγκου δεδομένων με μικρό χρόνο καθυστέρησης και με αυξημένη ανοχή στα σφάλματα.



Σχήμα 3.4 Αρχιτεκτονική μηχανισμού παρακολούθησης δικτύων SDN σε πλατφόρμες πολλαπλών ενοίκων

4

Ανάλυση Υλοποίησης

Όπως αναφέρθηκε στο Κεφάλαιο 3, στόχος της διπλωματικής εργασίας είναι η ρύθμιση ενός μηχανισμού παρακολούθησης SDN δικτύων προκειμένου να λειτουργεί σε πλατφόρμες πολλαπλών ενοίκων. Η ρύθμιση αυτή περιλαμβάνει

- προσθήκη μίας ουράς μηνυμάτων σα διεπαφή εισόδου με στόχο τη διαχείριση μεγάλου όγκου δεδομένων
- μετατροπές στο Flow Repository για κατάλληλη επεξεργασία και αποθήκευση των καταχωρήσεων
- μετατροπές στα εξαρτήματα των SDN controllers και του συλλέκτη των δειγμάτων για σωστή επικοινωνία με το Flow Repository
- την ανάπτυξη μίας βιβλιοθήκης που περιέχει λειτουργίες για την ανταλλαγή μηνυμάτων μεταξύ του server και του hypervisor και την αντιστοίχιση OVX και Physical διευθύνσεων στα πακέτα.

Στη συνέχεια θα αναλυθεί ο τρόπος υλοποίησης των παραπάνω μετατροπών.

4.1 kafka-python

Το εξάρτημα αυτό αποτελεί την υλοποίηση της ουράς μηνυμάτων Apache Kafka στη γλώσσα python. Είναι σχεδιασμένο να λειτουργεί με τον ίδιο τρόπο λειτουργίας του επίσημου java client, χρησιμοποιώντας διεπαφές της γλώσσας python. Για τη

δημοσίευση των μηνυμάτων χρησιμοποιούνται δύο εξαρτήματα, ο KafkaProducer και ο KafkaConsumer.

- KafkaProducer: Δέχεται σαν όρισμα το url address του kafka client και επιστρέφει έναν ασύγχρονο producer μηνυμάτων ο οποίος δημοσιεύει μηνύματα σε συγκεκριμένα topics
- KafkaConsumer: Δέχεται σαν όρισμα το url address του kafka client, το topic από το οποίο θα δέχεται μηνύματα και συγκεκριμένες ρυθμίσεις σχετικά με το ρυθμό άντλησης δεδομένων και επιστρέφει έναν consumer που συνεχώς δέχεται μηνύματα.

Αξίζει να σημειωθεί ότι, το Kafka δίνει τη δυνατότητα ομαδοποίησης των consumers ενός topic σε partitions, έτσι ώστε καθένα από αυτά να απορροφά διαφορετικά μηνύματα για χάρη scaling. Ο consumer κάθε φορά επιστρέφει μία τούπλα (tuple), η οποία περιέχει τα βασικά χαρακτηριστικά του μηνύματος. (topic, partition, offset, msg) .

topic	το topic από το όποιο προέρχεται το μήνυμα
partition	το partition των consumers για το οποίο προορίζεται το μήνυμα
offset	ο αύξων αριθμός του μηνύματος στο συγκεκριμένο partition
msg	dictionary που περιέχει τα δεδομένα του μηνύματος

Πίνακας 4.1 Δομή μηνύματος όπως αποστέλλεται μέσω του Kafka

4.2 Ανάλυση του αρχείου Flow_Repository.py

Το συγκεκριμένο αρχείο είναι γραμμένο σε γλώσσα python και αποτελεί την υλοποίηση του Flow Repository. Ορίζονται συναρτήσεις για την επικοινωνία του με τα διάφορα εξωτερικά εργαλεία (sflow collector, sflow agents, tenant controllers, hypervisor), για τη δημιουργία active και expired flow καταχωρήσεων και για την προσμέτρηση των δειγμάτων στις καταχωρήσεις αυτές. Ακόμη, προκειμένου ο μηχανισμός να παρακολουθεί εικονικά δίκτυα, δέχεται σαν όρισμα ένα αρχείο το οποίο περιέχει στοιχεία για την επικοινωνία του με τα διάφορα components και

ενεργοποιεί μέσω μίας Boolean μεταβλητής τις απαραίτητες τροποποιήσεις για να είναι συμβατός με τον OVX.

Hypervisor	Το όνομα του hypervisor που θα χρησιμοποιηθεί
Hypervisor url	Το url μέσω του οποίου επικοινωνεί με τον hypervisor
Messaging queue url	Το url μέσω του οποίου επικοινωνεί με το messaging queue
queue gid	Το group id του messaging queue στο οποίο ανήκουν οι consumers του Flow Repository

Πίνακας 4.2 Δομή Input αρχείου που δέχεται το Flow_Repository.py

Δομές δεδομένων που χρησιμοποιούνται:

- Boolean:
  - OVX_enable: Χρησιμοποιείται για να ενεργοποιήσει τις απαραίτητες τροποποιήσεις για να είναι η υλοποίηση συμβατή με τον OpenVirteX.
- Dictionaries:
  - active: Χρησιμοποιείται για την αποθήκευση των Active flow καταχωρήσεων
  - expired: Χρησιμοποιείται για την αποθήκευση των Expired flow καταχωρήσεων
  - mac_table: Χρησιμοποιείται για την αντιστοίχιση Source MAC address με Ingress port
  - mapper: Χρησιμοποιείται για την αντιστοίχιση των sflow πεδίων με τα αντίστοιχα πεδία του Openflow
  - hypervisor_var: Χρησιμοποιείται για την καταχώρηση των στοιχείων του hypervisor, καθώς και για να αποθηκεύει τις διάφορες αντιστοιχίσεις εικονικών-φυσικών διευθύνσεων

Συναρτήσεις που ορίζονται:

α) **construct_new_entry(args)** :

Η συνάρτηση έχει σκοπό τη δημιουργία μίας νέας active εγγραφής στο Flow Repository. Δέχεται ως όρισμα μία τούπλα (args), το μέγεθος της οποίας εξαρτάται από την ύπαρξη ή όχι του OpenVirteX. Σε περίπτωση που το δίκτυο που παρακολουθείται ρυθμίζεται από τον OVX, η τούπλα αποτελείται από πέντε στοιχεία και έχει την παρακάτω μορφή:

args[0]	OpenFlow Match	τα στοιχεία της ροής που δημιουργήθηκε στο switch λόγω του PacketIn event
args[1]	DPID	Datapath ID του OpenFlow switch
args[2]	Timestamp	η χρονική στιγμή κατά την οποία δημιουργήθηκε το νέο flow στο switch
args[3]	TID	το Tenant ID του ενοίκου από τον οποίο προέρχεται η συγκεκριμένη καταχώρηση
args[4]	Passwd	ο κωδικός του ενοίκου για ταυτοποίηση

Πίνακας 4.3 Τούπλα δεδομένων του controller για τη δημιουργία active entry

Σε διαφορετική περίπτωση, η τούπλα αποτελείται από τα τρία αρχικά στοιχεία, παραλείπεται δηλαδή το TID και το Password, τα οποία αφορούν κάποιο πιθανό ένοικο σε ένα εικονικό περιβάλλον.

Η αρχικοποίηση των παραμέτρων εξαρτάται από την ύπαρξη ή όχι του OpenVirteX, η οποία εξακριβώνεται μέσω της μεταβλητής OVX_enable.

- OVX_enable = False : σαν dpid χρησιμοποιείται η τιμή του args[1] ενώ η τιμή του TID ρυθμίζεται ίση με ‘0’ για να προσδιορίσει την απουσία του OVX.
- OVX_enable = True: χρησιμοποιείται μία συνάρτηση από τη βιβλιοθήκη mapper (mod_dpid) προκειμένου να τροποποιηθεί το Datapath ID σε μορφή συμβατή με αυτή των δειγμάτων που θα συλλεχτούν, ενώ για το TID χρησιμοποιείται η τιμή του args[3].

Στη συνέχεια, μέσω της συνάρτησης construct_hashed_key δημιουργείται μία hashed τιμή του OpenFlow Match. Ελέγχεται αν υπάρχει καταχώρηση active με

συγκεκριμένο tenant id (tid=0 σε περίπτωση απουσίας OVX) και αν δεν υπάρχει δημιουργείται με τιμή ένα κενό dictionary. Παρόμοια, ελέγχεται αν υπάρχει καταχώρηση στο active[tid] με όρισμα το Datapath ID (dpid) και αν δεν υπάρχει δημιουργείται με τιμή ένα κενό dictionary.

Στο active[tid][dpid] δημιουργείται, αν δεν υπάρχει ήδη, μία νέα εγγραφή με κλειδί την τιμή κατακερματισμού του OF match και τιμή ένα dictionary που έχει την παρακάτω δομή :

Keys	Values
"Counters"	Dictionary με τους μετρητές που χρησιμοποιούνται
"Match"	Dictionary που περιλαμβάνει ως keys τα πεδία του OpenFlow Match και ως values τις τιμές των πεδίων αυτών
"Timestamps"	Dictionary με keys "start", "end" και τις αντίστοιχες χρονικές στιγμές
"Tenant ID"	Το ID του ενοίκου από τον οποίο προέρχεται η εγγραφή

Πίνακας 4.4 Dictionary που αποθηκεύεται στα active entries

Τέλος, στο mac_table dictionary, καταχωρείται μλία αντιστοίχιση του ingress port του OF switch και της source MAC address με συγκεκριμένο tenant id και datapath id.

Αξίζει να σημειωθεί πως στη συγκεκριμένη συνάρτηση, απαραίτητη τροποποίηση για τη λειτουργία του μηχανισμού σε εικονικό περιβάλλον αποτέλεσε η ταξινόμηση των active καταχωρήσεων με βάση το tenant id. Με αυτό τον τρόπο διαχωρίζονται οι εγγραφές των διαφορετικών tenant και απλοποιείται η αναζήτηση της δικτυακής κίνησης που αφορά ένα συγκεκριμένο ένοικο.

β) **move_to_expired(args)** :

Η συνάρτηση δέχεται παρόμοιο όρισμα με αυτό της **construct_new_entry(args)** με τη διαφορά ότι το timestamp περιέχει τη χρονική στιγμή διαγραφής του συγκεκριμένου flow**.** Χρησιμοποιούνται οι ίδιες συναρτήσεις για την τροποποίηση του dpid, εάν είναι απαραίτητο (με βάση το OVX_enable), τη ρύθμιση του tid (με βάση το OVX_enable), και τη δημιουργία της hashed τιμής του match (το οποίο

αφορά πλέον μία ροή που εξέπνευσε). Στη συνέχεια ακολουθείται μία διαφορετική διαδικασία.

Ελέγχεται αν υπάρχει καταχώρηση active για το συγκεκριμένο tenant id και dpid με κλειδί τη hash τιμή του match. Αναμένεται να υπάρχει, παρόλα αυτά σε περίπτωση που δεν υπάρχει, η συνάρτηση επιστρέφει. Αν υπάρχει, αφαιρείται από το dictionary. Καλείται η συνάρτηση κατακερματισμού για υπολογισμό της hashed τιμής του match εκ νέου, αυτή τη φορά συμπεριλαμβάνοντας και τις χρονικές στιγμές δημιουργίας και διαγραφής της ροής. Τέλος, δημιουργείται μία νέα expired καταχώρηση expired[tid][dpid] με κλειδί την συγκεκριμένη hash τιμή και τιμή το dictionary που περιείχε η active καταχώρηση που αφαιρέθηκε .

Παρόμοια με τις active καταχωρήσεις, οι expired οργανώνονται με βάση το tenant id και στη συνέχεια με το dpid και το κατάλληλο hash.

γ) **collect_sflow(flow)** :

Η συνάρτηση δέχεται ως όρισμα ένα dictionary που περιέχει ένα sflow sample. Το δείγμα αυτό επεξεργάζεται και αποθηκεύεται κατάλληλα για να αντιστοιχιστεί με την κατάλληλη active εγγραφή.

Όπως αναφέρθηκε στο Κεφάλαιο 3, ο OVX μετατρέπει τις διευθύνσεις που περιέχουν τα πακέτα στο φυσικό επίπεδο για να διαχωρίσει την κίνηση μεταξύ των ενοίκων. Συνεπώς, προτού πραγματοποιηθεί αντιστοίχιση και προκειμένου αυτή να είναι εφικτή, πρέπει να μεταφραστούν ξανά οι διευθύνσεις του πακέτου στις OVX τιμές που περιέχονται στις εγγραφές.

Αρχικά, μέσω της μεταβλητής OVX_enable ελέγχεται η ύπαρξη του OpenVirteX. Σε περίπτωση που τα δείγματα δεν προέρχονται από κάποιο OVX περιβάλλον, δεν απαιτείται περαιτέρω επεξεργασία στις διευθύνσεις που περιέχουν. Η διαδικασία συνεχίζεται κανονικά, ενώ η μεταβλητή tid που χρησιμοποιείται για την ομαδοποίηση των εγγραφών με βάση τους ενοίκους του δικτύου αποκτά την τιμή ‘0’.

Σε ένα OVX περιβάλλον απαιτείται η μετάφραση των Physical διευθύνσεων που μπορεί να περιέχουν τα πακέτα σε OVX. Για το σκοπό αυτό ακολουθείται η παρακάτω διαδικασία:

Σύμφωνα με τον πίνακα 3.1 του Κεφαλαίου 3, τα δείγματα που περιέχουν PhysicalMACAddress, σίγουρα περιέχουν και PhysicalIPAddress, ενώ τα δείγματα που περιέχουν OVXMACAddress μπορεί να περιέχουν είτε OVXIPAddress (προέρχονται από δειγματοληψία σε edge switch του δικτύου) ή PhysicalIPAddress (προέρχονται από δειγματοληψία σε switch στο εσωτερικό ενός Big Switch). Ακόμη, υπάρχει και η περίπτωση το δείγμα να αποτελεί εσωτερικό μήνυματα του OpenVirteX, το οποίο δεν περιέχουν στην κεφαλίδα διευθύνσεις IP αποστολέα και παραλήπτη. Στην περίπτωση αυτή το δείγμα αγνοείται και η συνάρτηση επιτρέφει. Διαφορετικά διακρίνονται οι εξής περιπτώσεις, με βάση τις MAC διευθύνσεις του δείγματος:

A) PhysicalMACAddress:

Τα δείγματα αυτά εντοπίζονται με βάση τη διευθυνσιοδότηση που ακολουθεί ο OVX για την δημιουργία physical MAC διευθύνσεων. Σύμφωνα με αυτή, τα πρώτα 4 bytes μίας τέτοιας διεύθυνσης MAC αποτελούν την ακολουθία 00:a4:23:05, ενώ τα επόμενα 2 bytes καθορίζουν το tenant id του ενοίκου. Μέσω της MAC του πακέτου συνεπώς καθορίζεται αν το πακέτο χρίζει μετάφρασης διευθύνσεων και σε κάθε τέτοια περίπτωση εντοπίζεται ο ένοικος στον οποίο ανήκει.

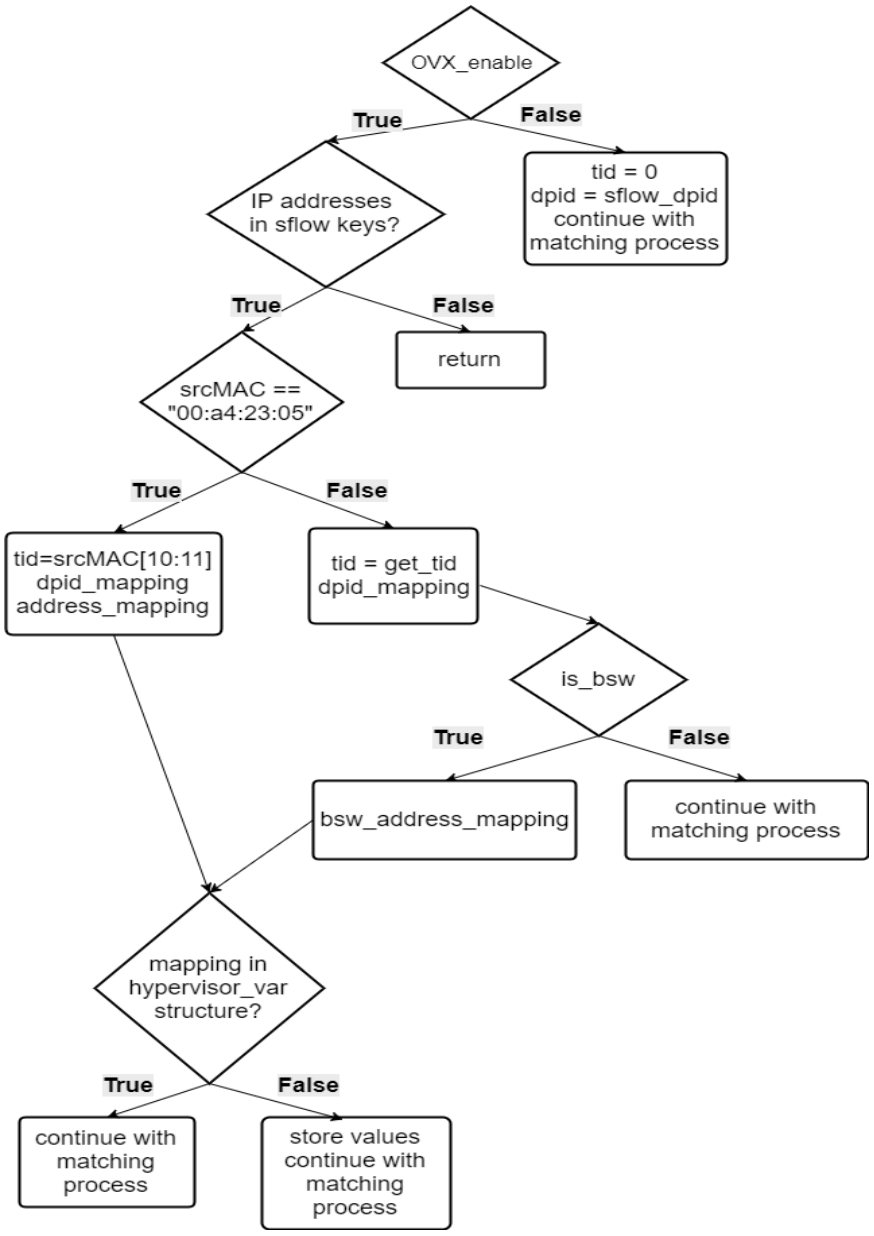
Στη συνέχεια, μέσω μίας συνάρτησης της βιβλιοθήκης mapper (dpid_mapping) γίνεται αντιστοίχιση της Physical datapath id με την αντίστοιχη OVX. Επιπλέον, μέσω μίας άλλης συνάρτησης από την ίδια βιβλιοθήκη (address_mapping) αντιστοιχίζονται οι φυσικές IP και MAC διευθύνσεις πηγής (srcIP, srcMAC) με τις αντίστοιχες OVX. Η διαδικασία πραγματοποιείται μία ακόμη φορά προκειμένου να αντιστοιχιστούν και οι IP και MAC διευθύνσεις προορισμού (dstIP, dstMAC).

B)OVXMACAddress:

Στα δείγματα αυτά η MAC διεύθυνση δεν περιέχει την ακολουθία 00:a4:23:05 στα αρχικά bytes. Σε αυτή την περίπτωση, μέσω μίας συνάρτησης της βιβλιοθήκης mapper (get_tid) ανακαλύπτεται το tenant id του πακέτου, ενώ μέσω μίας άλλης συνάρτησης της ίδιας βιβλιοθήκης(dpid_mapping) γίνεται αντιστοίχιση της Physical datapath id με την OVX. Η ίδια συνάρτηση επιστρέφει μία Boolean μεταβλητή (is_bsw) που προσδιορίζει αν το δείγμα ανήκει σε κάποιο big switch. Αν το δείγμα δεν προέρχεται από κάποιο big switch, δεν χρειάζεται περαιτέρω επεξεργασία. Στην

αντίθετη περίπτωση χρησιμοποιείται μία συνάρτηση από την mapper βιβλιοθήκη (bsw_address_mapping) για τη μετάφραση των διευθύνσεων IP του πακέτου.

Οι αντιστοιχίσεις που πραγματοποιούνται στις διευθύνσεις IP και MAC αποθηκεύονται στη δομή hypervisor_var σε περίπτωση που χρειαστεί να χρησιμοποιηθούν μελλοντικά, μειώνοντας με αυτό τον τρόπο τον αριθμό αιτήσεων του FlowRepository στον hypervisor.



Σχήμα 4.1 Διάγραμμα ροής για το address mapping των samples

Με αυτό τον τρόπο, το πακέτο είναι πλέον έτοιμο να αντιστοιχιστεί με την κατάλληλη active εγγραφή ροής. Χρησιμοποιώντας τη συνάρτηση construct_hashed_sflow υπολογίζεται η hashed τιμή του δείγματος και μέσω του

tenant id και στη συνέχεια της τιμής αυτής εντοπίζεται, στο active dictionary, η καταχώρηση που του αντιστοιχεί προκειμένου να αυξηθούν οι κατάλληλοι δείκτες.

δ) **register_queue() :**

Η συγκεκριμένη συνάρτηση χρησιμοποιείται για την επικοινωνία του server με το Kafka (σύστημα διαχείρισης μηνυμάτων που λειτουργεί σαν input interface). Χρησιμοποιεί μία βιβλιοθήκη υλοποιημένη σε python για τη δημιουργία δύο consumers, οι οποίοι συνδέονται με δύο διαφορετικά topics :

- main_consumer: Συνδέεται με το topic “main”. Στο συγκεκριμένο topic δημοσιεύουν μηνύματα οι διάφοροι sflow agents καθώς και οι virtual controllers
- client_consumer: Συνδέεται με το topic “client”. Στο topic αυτό δημοσιεύουν αιτήσεις τα προγράμματα των διαφόρων ενοίκων ή του διαχειριστή του δικτύου προκειμένου να αντληθούν οι ήδη υπάρχουσες καταχωρήσεις που αφορούν το δίκτυο.

Αξίζει να σημειωθεί πως ανάμεσα στους δύο consumers, ο πρώτος αποτελεί την κύρια μέθοδο εισαγωγής δεδομένων στον server καθώς δημοσιεύονται συνεχώς μηνύματα από τα διάφορα components του μηχανισμού προκειμένου να καταγράφεται η δικτυακή κίνηση. Ο client_consumer ουσιαστικά επιτρέπει επικοινωνία και άντληση πληροφοριών από το repository. Παρόλα αυτά, καθώς κρίνεται αναγκαία η γρήγορη απόκριση του server στις αιτήσεις των clients, για να μην καθυστερούν οι αιτήσεις αυτές λόγω της συνεχής ροής μηνυμάτων στον main consumer, η ανάγνωση και επεξεργασία των μηνυμάτων του client consumer αποκτά προτεραιότητα.

ε) **get_input_from_queue(serialized_request)** :

Η συνάρτηση αυτή δέχεται ως όρισμα μία σειριοποιημένο αίτηση, η οποία προέρχεται από τα μηνύματα που αντλεί ο server από τα παραπάνω topic. Η αποσειριοποίηση της πραγματοποιείται με χρήση της βιβλιοθήκης pickle. Η αίτηση περιέχει μία τούπλα, το πρώτο στοιχείο της οποίας καθορίζει τη συνάρτηση του server που θα χρησιμοποιηθεί και τα επόμενα περιέχουν τα στοιχεία που η συνάρτηση αυτή θα δεχτεί σαν ορίσματα. Με βάση την τούπλα αυτή καλείται η κατάλληλη συνάρτηση για να εξυπηρετήσει την αίτηση.

4.3 Ανάλυση του αρχείου mapper.py

Το συγκεκριμένο αρχείο αποτελεί την υλοποίηση μίας βιβλιοθήκης σε γλώσσα python. Η βιβλιοθήκη εξυπηρετεί στην επικοινωνία του Flow Repository με τον OpenVirteX προκειμένου να γίνει αντιστοίχιση μεταξύ των OVX και Physical διευθύνσεων. Η υλοποίηση της βιβλιοθήκης αυτής βασίστηκε στο API του OpenVirteX και πιο συγκεκριμένα στο αρχείο ovxctl.py. Από το συγκεκριμένο αρχείο χρησιμοποιήθηκαν οι συναρτήσεις buildRequest και connect (περιγράφονται παρακάτω). Οι συναρτήσεις αυτές καλλούνται από τις υπόλοιπες συναρτήσεις της βιβλιοθήκης προκειμένου να επικοινωνήσουν με τον OpenVirteX και να αντλήσουν τα επιθυμητά δεδομένα.

Η βιβλιοθήκη περιέχει της εξής συναρτήσεις:

α) **buildRequest (data, url, cmd)** :

Η συνάρτηση χρησιμοποιείται για να "χτίσει" το αίτημα που θα σταλθεί στον OpenVirteX. Δέχεται τρία ορίσματα.

1. data : Περιέχει τα δεδομένα που θα σταλθούν στον OVX προς επεξεργασία
2. url : Περιέχει τη διεύθυνση και την πόρτα στην οποία δέχεται αιτήσεις ο OVX
3. cmd : Περιέχει την εντολή που περιγράφει το αίτημα (πίνακας).

cmd	input	output (Dictionary)
getVirtualSwitchMapping	tenant ID	Αντιστοίχιση OVX/φυσικών dpids των switches ενός ενοίκου
getPhysicalHosts	-	Αντιστοίχιση PhysicalIP/MAC όλων των host του δικτύου
getVirtualHosts	tenant ID	Αντιστοίχιση OVXIP/MAC όλων των host ενός ενοίκου

Πίνακας 4.5 Δομή του request που αποστέλλεται στον OVX

β) **connect (url, cmd , data=None, tid=None, passwd=None)** :

Η συνάρτηση δέχεται πέντε ορίσαμτα. Τα δύο τελευταία (tid=None, passwd=None) αφορούν την ταυτοποίηση του ενοίκου για τον οποίο θα ζητηθούν πληροφορίες. Τα αρχικά τρία ορίσματα είναι ίδια με αυτά που περιγράφτηκαν στην προηγούμενη συνάρτηση. Αφού ταυτοποιηθεί ο ένοικος, χρησιμοποιείται η παραπάνω συνάρτηση (buildRequest (data, url, cmd)) προκειμένου να δημιουργηθεί το αίτημα στην κατάλληλη μορφή για τον OVX και στη συνέχεια αποστέλλεται στο καθορισμένο URL για να εξυπηρετηθεί.

γ) **mod_dpid (dpid)** :

Η συνάρτηση χρησιμοποιείται για την τροποποίηση της μορφής του Datapath ID προκειμένου να ταυτίζεται με τη μορφή του dpid όπως αποθηκεύεται στον OpenVirteX. Δέχεται σαν όρισμα το dpid με τη μορφή που το στέλνει ο εκάστοτε controller. Παρακάτω φαίνεται η αρχική μορφή ενός dpid και η τροποποιημένη:

Μορφή αποστολής από controller	**Μορφή συμβατή με OpenVirteX**
a4230500000001	00:a4:23:05:00:00:00:01

Πίνακας 4.6 Παράδειγμα input/output της συνάρτησης mod_dpid

δ) **get_tid(url, mac, passwd="")** :

Η συνάρτηση δέχεται σαν πρώτο όρισμα το url στο οποίο επικοινωνεί ο OpenVirteX και σαν δεύτερο μία από τις δύο MAC addresses που περιέχει το δείγμα που συλλέχτηκε (μπορεί να χρησιμοποιηθεί και η source και η destination address).

Σκοπός της συνάρτησης είναι ο καθορισμός του ID του ενοίκου στον οποίο ανήκει το δείγμα στην περίπτωση που το δείγμα που συλλέχτηκε περιέχει OVX διευθύνσεις. Αυτό απαιτείται διότι σε αυτή την περίπτωση οι IPs που χρησιμοποιούνται είναι μοναδικές μόνο στο εικονικό δίκτυο του κάθε ενοίκου και ενδέχεται να υπάρχουν επικαλύψεις.

Καλείται η συνάρτηση connect χρησιμοποιώντας σαν cmd την εντολή "getPhysicalHosts". Ο OpenVirteX θα επιστρέψει στην προκειμένη περίπτωση ένα dictionary με όλους τους Hosts του δικτύου για τους οποίους έχει δημιουργήσει

φυσικές IPs (αυτούς δηλαδή που έχουν στείλει πακέτα στο δίκτυο), παραθέτοντας την Physical IP και τη διεύθυνση MAC του καθενός.

Με βάση τη MAC του πακέτου εντοπίζεται ο host που είναι είτε δημιουργός είτε αποδέκτης του (ανάλογα τη MAC που χρησιμοποιήσαμε). Στο συγκεκριμένο σημείο, εκμεταλλευόμαστε το γεγονός ότι κάθε host ανήκει σε ένα μόνο εικονικό δίκτυο, καθώς και τον αλγόριθμο διευθυνσιοδότησης που ακολουθεί ο OVX. Σύμφωνα με τον αλγόριθμο αυτό, η Physical IP κάθε host περιέχει στο πρώτο της byte την ID του ενοίκου (π.χ. ένοικος με ID = 1 -> IPs των hosts 1.0.0.x). Με αυτόν τον τρόπο εντοπίζεται και επιστρέφεται η ID του ενοίκου στον οποίο ανήκει το πακέτο.

ε**) dpid_mapping (url, dpid, ten_id, passwd="")** :

Η συγκεκριμένη συνάρτηση χρησιμοποιείται για την αντιστοίχιση εικονικών και φυσικών Datapath IDs των switches στα πακέτα δειγματοληψίας. Δέχεται σαν όρισμα το url στο οποίο "ακούει" ο OpenVirteX, την Physical dpid του switch, το ID του ενοίκου στον οποίο ανήκει το πακέτο και τον κωδικό του ενοίκου για να γίνει ταυτοποίηση. Ακόμη, κάθε φορά που καλείται η συνάρτηση, αρχικοποιείται μία Boolean μεταβλητή is_bsw στην τιμή False. Η συγκεκριμένη μετραβλητή χρησιμοποιείται για τον προσδιορισμό των δειγμάτων που αντλήθηκαν από big switches.

Μέσω της συνάρτησης connect γίνεται επικοινωνία με τον OpenVirteX χρησιμοποιώντας ως εντολή την "getVirtualSwitchMapping". Ο OVX επιστρέφει ένα dictionary το οποίο περιέχει τις εικονικές dpids όλων των virtual switches του συγκεκριμένου ενοίκου σε αντιστοιχία με τις φυσικές dpids. Η αντιστοιχία αυτή είναι 1 προς 1 σε περίπτωση απλού εικονικού switch, ενώ για τα Big Switches μία OVX διεύθυνση αντιστοιχεί σε πολλαπλές physical.

Χρησιμοποιώντας τέλος, την υπάρχουσα physical dpid , εντοπίζεται η ζητούμενη OVX Datapath ID για το συγκεκριμένο switch. Ακόμη, σε περίπτωση που η τιμή αυτή αντιστοιχίζεται με πολλαπλές physical dpids, η μεταβλητή is_bsw παίρνει την τιμή True. Η συνάρτηση επιστρέφει μία τούπλα που αποτελείται από τη μεταβλητή is_bsw και τη ζητούμενη ovx_dpid.

στ) **address_mapping (url, ten_ip, ten_id, passwd="")** :

Η συνάρτηση αποσκοπεί στη μεταγλώττιση των φυσικών διευθύνσεων ενός πακέτου δειγματοληψίας σε OVX, προκειμένου να είναι ικανή η αντιστοίχιση τους στις active εγγραφές του Flow Repository. Δέχεται σαν όρισμα τέσσερις μεταβλητές. Τη μεταβλητή url που περιέχει την διεύθυνση του OpenVirteX (όπως είδαμε και παραπάνω), τη μεταβλητή ten_ip η οποία περιέχει τη Physical ip προς μετάφραση και τις μεταβλητές ten_id και passwd που περιέχουν την ID του ενοίκου και τον κωδικό ταυτοποίησης του.

Αρχικά καθορίζεται η OVX MAC διεύθυνση που αντιστοιχεί στον host με τη συγκεκριμένη IP. Για το σκοπό αυτό καλείται η συνάρτηση connect με όρισμα την εντολή "getPhysicalHosts" η οποία επιστρέφει ένα dictionary με όλους τους host στο δίκτυο που έχουν λάβει Physical IP. Στο dictionary περιέχεται η Physical IP των hosts και η MAC διεύθυνση τους. Με χρήση της μεταβλητής ten_ip εντοπίζεται η ζητούμενη MAC.

Στη συνέχεια, καλείται άλλη μία φορά η συνάρτηση connect, έχοντας ως όρισμα την εντολή "getVirtualHosts" , καθώς και το tenant ID του ενοίκου. Ο OVX απαντά στέλνοντας ένα dictionary που περιέχει όλους τους hosts που ανήκουν στο δίκτυο του συγκεκριμένου ενοίκου. Στο dictionary περιλαμβάνεται η OVX IP των hosts καθώς και η MAC τους. Έχοντας εντοπίσει ήδη τη ζητούμενη MAC είναι πλέον δυνατός ο καθορισμός της OVX IP που αναζητήθηκε.

ζ) **bsw_address_mapping (url, ten_ip, ten_id, mac, passwd=""):**

Η συνάρτηση αποσκοπεί στη μεταγλώττιση της Physical IP διεύθυνσης ενός πακέτου δειγματοληψίας σε OVX, σε περίπτωση που το πακέτο έχει αντληθεί από κάποιο big switch. Δέχεται σα μεταβλητές πέντε ορίσματα. Τη μεταβλητή url που περιέχει την διεύθυνση του OpenVirteX (όπως είδαμε και παραπάνω), τη μεταβλητή ten_ip η οποία περιέχει τη Physical ip προς μετάφραση, τη μεταβλητή mac η οποία αποτελεί την OVX MAC διεύθυνση του πακέτου και τις μεταβλητές ten_id και passwd που περιέχουν την ID του ενοίκου και τον κωδικό ταυτοποίησης του.

Καλείται η συνάρτηση connect, έχοντας ως όρισμα την εντολή "getVirtualHosts" , καθώς και το tenant ID του ενοίκου. Ο OVX απαντά στέλνοντας ένα dictionary που περιέχει όλους τους hosts που ανήκουν στο δίκτυο του συγκεκριμένου ενοίκου. Στο

dictionary περιλαμβάνεται η OVX IP των hosts καθώς και η MAC τους. Με χρήση της MAC διεύθυνσης του πακέτου, την οποία έχει δεχτεί σαν όρισμα η συνάρτηση, εντοπίζεται η ζητούμενη OVX IP του πακέτου.

4.4 Ανάλυση των αρχείων flowrem.py και sflow.py

Τα συγκεκριμένα αρχεία αποτελούν εργαλεία που συλλέγουν δεδομένα δικτυακής κίνησης και τα αποστέλλουν στο Flow Repository.

4.4.1 flowrem.py

Το **flowrem.py** είναι ένα εξάρτημα του controller POX το οποίο παρέχει δύο συναρτήσεις για το χειρισμό δύο διαφορετικών Events(CustomEvent, FlowRemoved). Τα συγκεκριμένα event γίνονται raised κάθε φορά που ο controller επιθυμεί να εγκαταστήσει ένα νέο κανόνα σε κάποιο switch (CustomEvent) και κάθε φορά που ένας κανόνας εκπνέει και απομακρύνεται (FlowRemoved). Το flowrem.py, στις περιπτώσεις αυτές επικοινωνεί με το Flow Repository με σκοπό να δημιουργήσει ένα νέο active entry ή να μεταφερθεί ένα active entry στα expired.

Ακόμη, σε ένα περιβάλλον OVX το αρχείο flowrem.py δημοσιεύει δεδομένα που προέρχονται από τον controller ενός συγκεκριμένου ενοίκου. Επομένως, δέχεται, κατά την εκκίνηση του, σαν όρισμα το tenant ID και το passwd του ενοίκου που του αντιστοιχεί. Αυτό συμβαίνει προκειμένου να είναι δυνατή η ταξινόμηση των μηνυμάτων που στέλνει στο Flow Repository ανά tenant αλλά και να εξασφαλιστεί η αξιοπιστία των μηνυμάτων αυτών.

4.4.2 sflow.py

Το **sflow.py** χρησιμοποιείται για τη συλλογή των sFlow samples και την αποστολή τους στο Flow Repository. Ο collector που χρησιμοποιείται είναι ο sflowtool. Είναι πρόγραμμα ανοιχτού κώδικα υλοποιημένο σε γλώσσα C. Προκειμένου να είναι συμβατός με το python περιβάλλον χρησιμοποιήθηκε ένας wrapper μέσω της βιβλιοθήκης subprocess. Το sflow.py συλλέγει τα διάφορα δείγματα από τους sflow

agents στο φυσικό δίκτυο, τα επεξεργάζεται προκειμένου να μπορεί να τα διαχειριστεί το FlowRepository και στη συνέχεια τα αποστέλλει σε αυτό

4.4.3 Τροποποιήσεις

Στα πλαίσια της παρούσας διπλωματικής δε χρειάστηκε να εφαρμοστούν τροποποιήσεις στον τρόπο συλλογής ή επεξεργασίας των δεδομένων των δύο αυτών εφαρμογών. Η μοναδική μετατροπή που πραγματοποιήθηκε είναι η χρήση του Kafka για την επικοινωνία των συγκεκριμένων εφαρμογών με το Flow Repository.

Για τα σκοπό αυτό δημιουργήθηκε η παρακάτω συνάρτηση :

- **register_queue (url)** : Η συνάρτηση δέχεται σαν όρισμα τη διεύθυνση στην οποία βρίσκεται ο server του Kafka και δημιουργεί μέσω της βιβλιοθήκης kafka ,υλοποιημένης σε python, ένα producer, ο οποίος είναι υπεύθυνος να δημοσιεύει τα μηνύματα στο topic που είναι εγγεγραμμένο το Flow Repository. Το topic αυτό ονομάζεται “main”.

Κάθε φορά που τα παραπάνω στοιχεία είναι έτοιμα να στείλουν δεδομένα στο repository, δημιουργούν μία τούπλα. Το πρώτο όρισμα της τούπλας αφορά τη συνάρτηση που θα εξυπηρετήσει το συγκεκριμένο αίτημα (construct_new_entry, move_to_expired, collect_sflow) στο Flow Repository ενώ τα επόμενα στοιχεία ταυτίζονται με τη δομή του input που περιμένει η συνάρτηση αυτή. Η συγκεκριμένη τούπλα σειριοποιείται με τη βοήθεια της βιβλιοθήκης pickle και δημοσιεύεται από τον producer στο “main” topic.

function	tuple
construct_new_entry	(“construct_new_entry”,ofp.match,dpid,timestamp,tid,passwd)
move_to_expired	(“move_to_expired”, ofp.match, dpid, timestamp, tid, passwd)
collect_sflow	(“collect_sflow”,flow)

Πίνακας 4.7 Μορφή δεδομένων κατά την αποστολή τους στο Flow Repository, ανάλογα με τη λειτουργία την οποία καλούν

5

Αξιολόγηση

Στο κεφάλαιο αυτό παρουσιάζεται η αξιολόγηση της υλοποίησης. Καθορίζεται το σύστημα αξιολόγησης, αναλύεται η πειραματική διαδικασία που ακολουθήθηκε, ενώ εκθέτονται και αξιολογούνται τα δεδομένα που αντλήθηκαν από τη διαδικασία αυτή.

5.1 Σύστημα αξιολόγησης

Για την αξιολόγηση του συστήματος εξετάστηκαν τρία διαφορετικά χαρακτηριστικά του:

- Ελέγθηκε η σωστή λειτουργία της διαδικασίας αντιστοίχισης διευθύνσεων και η κατανάλωση υπολογιστικών πόρων και πόρων μνήμης που προσθέτει η διαδικασία αυτή στο μηχανισμό.
- Μελετήθηκε η ικανότητα του μηχανισμού να παρακολουθεί διαφορετικά εικονικά δίκτυα ταυτόχρονα, κυρίως σε περιπτώσεις όπου κάποιος ένοικος παράγει μεγάλα ποσά δικτυακής κίνησης σε σύγκριση με τους υπόλοιπους.
- Εξετάστηκε η συμπεριφορά του μηχανισμού κατά την παρακολούθηση εικονικών δικτύων με πιο σύνθετες τοπολογίες, καθώς αποτελούν δομές που χρήζουν ιδιαίτερης μεταχείρισης.

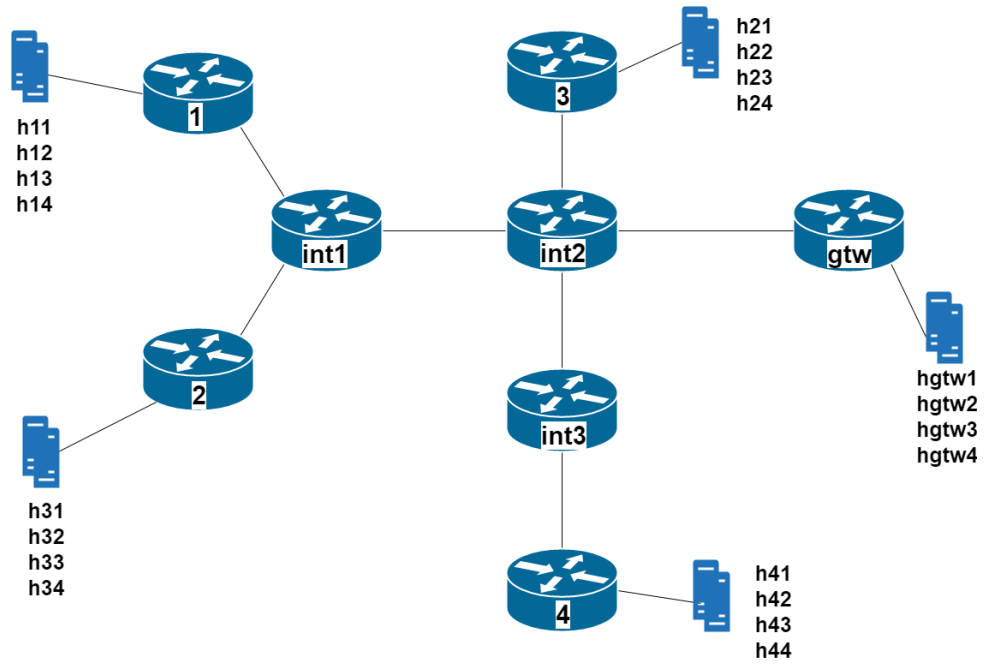
Για την πραγματοποίηση των πειραμάτων χρησιμοποιήθηκε μία εικονική μηχανή (λειτουργικό σύστημα Ubuntu 14.04), ενώ οι διάφορες δικτυακές τοπολογίες προσομοιώθηκαν από το πρόγραμμα Mininet [25]. Ακόμη, εκτός από τα αρχεία κώδικα της υλοποίησης, που περιγράφονται εκτενώς στο κεφάλαιο 4, κατά την πειραματική διαδικασία χρησιμοποιήθηκαν τα ακόλουθα προγράμματα:

- Top[26]
- Iperf[24]
- OpenVirteX
- Kafka

5.2 Λειτουργία σε εικονικό περιβάλλον

Αρχικά μελετάται η λειτουργία του μηχανισμού στο εικονικό περιβάλλον που δημιουργεί ο OVX, εστιάζοντας στην ικανότητα σωστής αντιστοίχισης OVX-physical διευθύνσεων καθώς και στην χρήση των υπολογιστικών πόρων και των πόρων σε μνήμη που αυτή απαιτεί.

5.2.1 Πειραματική διάταξη

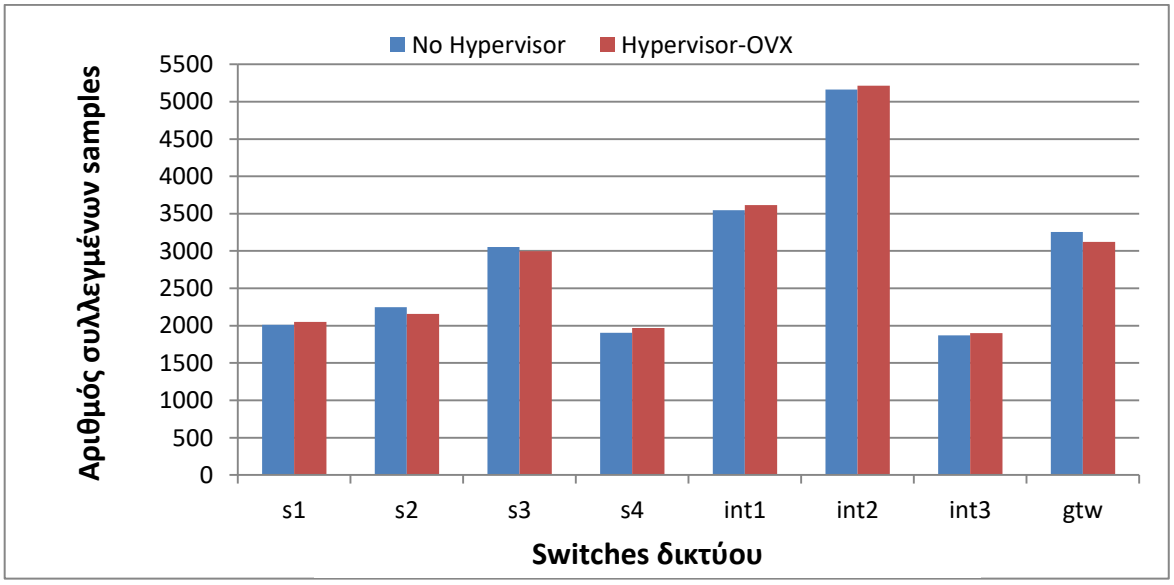


Σχήμα 5.1 Διάταξη πειρματικής τοπολογίας 1

Προσομοιώθηκε, για τους σκοπούς του πειράματος, ένα δίκτυο που αποτελείται από οχτώ switches με τοπολογία στα πρότυπα ενός δικτύου campus (Σχήμα 5.1). Για την προσομοίωση του δικτύου χρησιμοποιήθηκε το πρόγραμμα Mininet. Όπως φαίνεται και στο παρακάτω σχήμα, τα ακραία switches της τοπολογίας είναι συνδεδεμένα με τέσσερις hosts το καθένα. Μέσω των hosts αυτών παράγεται δικτυακή κίνηση (πακέτα UDP και TCP) με χρήση του προγράμματος Iperf, η οποία παρακολουθείται από το μηχανισμό μας.

5.2.2 Αποτελέσματα

Η παρακολούθηση του παραπάνω δικτύου πραγματοποιήθηκε σε δύο περιβάλλοντα, στη φυσική μορφή του δικτύου και σε εικονικό περιβάλλον με χρήση του OVX. Και στις δύο περιπτώσεις δημιουργήθηκαν συνολικά πενήντα ροές πακέτων UDP/TCP με διαφορετική διάρκεια και ένταση. Παρακάτω παρουσιάζονται τα αποτελέσματα (αριθμός πακέτων) που αντλήθηκαν από τα οκτώ switches του δικτύου.

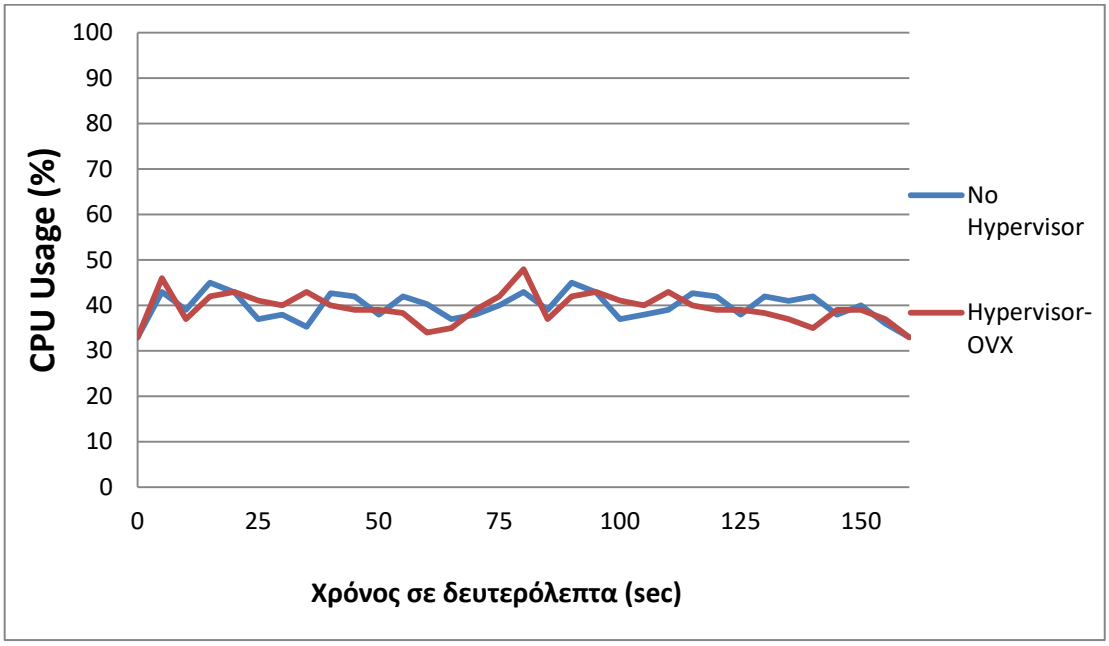


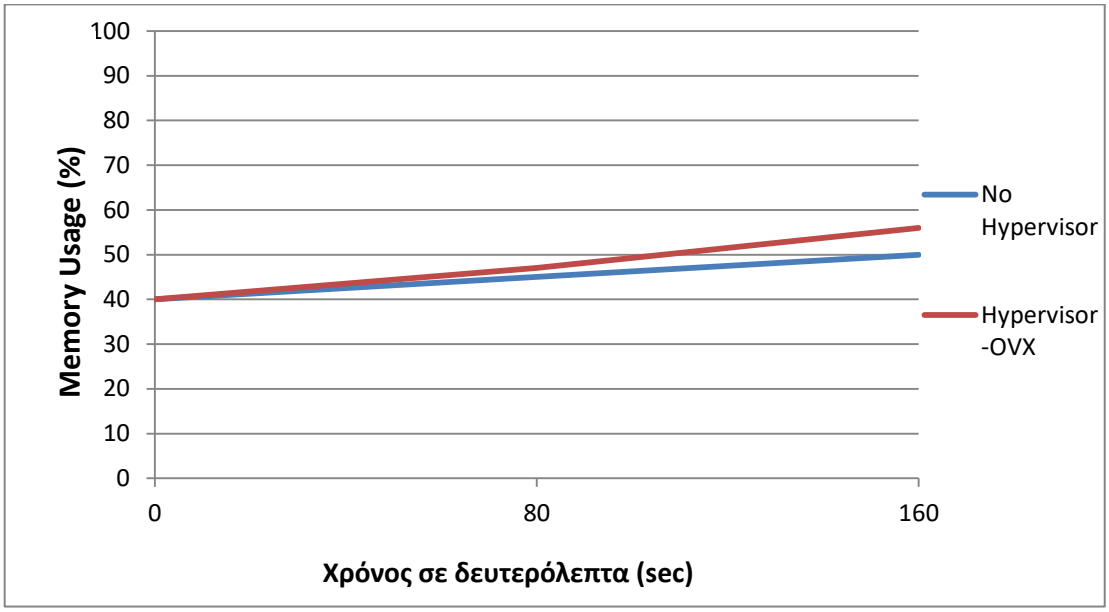
Σχήμα 5.2 Αποτελέσματα συλλογής δειγμάτων σε δίκτυα χωρίς και με παρουσία OVX

Και στις δύο περιπτώσεις, ο μηχανισμός μας ήταν ικανός να εντοπίσει το σύνολο των ροών που δημιουργήθηκαν. Παρατηρούμε πως και στο εικονικό περιβάλλον του OVX και στην απλή φυσική τοπολογία, ο αριθμός των πακέτων που συλλέχθηκαν είναι περίπου ίσος. Ως αποτέλεσμα, συμπεραίνουμε ότι η αντιστοίχιση OVX-physical διευθύνσεων του μηχανισμού μας είναι ακριβής και τα δεδομένα που συλλέγει για την κίνηση σε εικονικά δίκτυα είναι αξιόπιστα.

Οι μικρές διαφορές στον αριθμό των πακέτων ανά switch οφείλονται κυρίως σε αναμεταδόσεις πακέτων (κυρίως πακέτα UDP) από το Iperf και σε μικρότερο βαθμό στην ύπαρξη πακέτων που παράγει ο OVX για την εσωτερική αναγνώριση και σηματοδοσία των διαφόρων switches που ελέγχει. Ποσοστιαία, η ύπαρξη των πακέτων του OVX είναι <0.1%.

Στη συνέχεια ακολουθούν μετρήσεις σχετικά με το πρόσθετο υπολογιστικό κόστος καθώς και το κόστος σε μνήμη που επιφέρει η εκτέλεση του μηχανισμού σε εικονικό περιβάλλον. Οι μετρήσεις λήφθηκαν μέσω του προγράμματος TOP και παρουσιάζονται στα παρακάτω διαγράμματα. Στο πρώτο διάγραμμα φαίνεται η μέση χρήση του επεξεργαστή για τα δύο παραπάνω πειράματα ενώ στο επόμενο παρουσιάζεται η κατανάλωση σε μνήμη.





Σχήμα 5.3 Αποτελέσματα κατανάλωσης υπολογιστικών πόρων και πόρων μνήμης για τις δύο περιπτώσεις

Παρατηρούμε πως το υπολογιστικό κόστος είναι παρόμοιο και στις δύο περιπτώσεις, παρόλο που στην περίπτωση της εικονικής τοπολογίας ο μηχανισμός χρειάζεται περισσότερους υπολογισμούς προκειμένου να αντιστοιχίσει OVX-physical διευθύνσεις. Αυτό συμβαίνει διότι, οι αντιστοιχίσεις αυτές αποθηκεύονται για μελλοντική χρήση. Η αποθήκευση των αντιστοιχίσεων περιορίζει το επιπλέον κόστος καθώς, σε κάθε περίπτωση, το σύνολο των πακέτων προέρχεται από συγκεκριμένους hosts των ενοίκων, συνεπώς οι ίδιες αντιστοιχίσεις διευθύνσεων επαναλαμβάνονται.

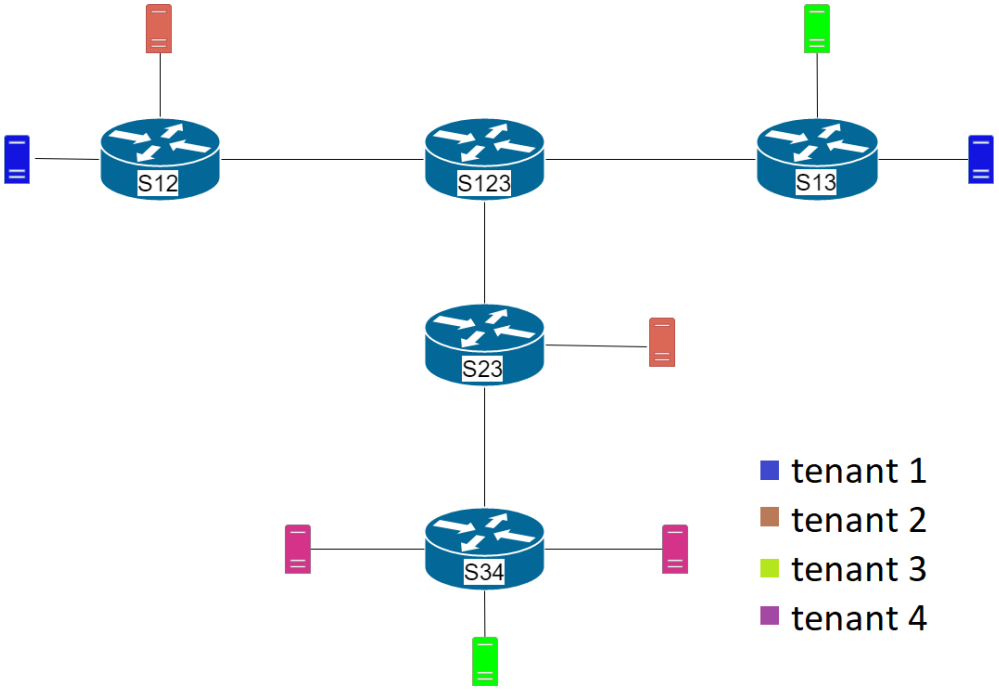
Όσον αφορά το κόστος σε μνήμη, όπως ήταν αναμενόμενο, παρατηρούμε αύξηση στην κατανάλωση μνήμης κατά τη χρήση της υλοποίησης σε εικονικό περιβάλλον. Η αύξηση αυτή προέρχεται κυρίως από τις επιπλέον πληροφορίες που διατηρούνται στις εγγραφές (OVX-physical dpids, tenant ids), καθώς και από τις επιπλέον πληροφορίες που διατηρούνται σχετικά με τα χαρακτηριστικά του OVX hypervisor και τις αντιστοιχίσεις OVX-physical διευθύνσεων, όπως αναφέρθηκε παραπάνω.

5.3 Multitenancy

Όπως αναφέρθηκε προηγουμένως, η δειγματοληψία πακέτων του μηχανισμού πραγματοποιείται στο φυσικό επίπεδο, στο οποίο συνυπάρχει δικτυακή κίνηση όλων των εικονικών δικτύων. Σκοπός του πειράματος είναι ο έλεγχος της ικανότητας του μηχανισμού να εντοπίζει δικτυακή κίνηση από όλους τους ενοίκους, κυρίως σε περιπτώσεις όπου κάποιος από αυτούς παράγει μεγάλα ποσά κίνησης, «γεμίζοντας» το δίκτυο με δικά του πακέτα.

5.3.1 Πειραματική διάταξη

Για τους σκοπούς του πειράματος προσομοιώθηκε μέσω του Mininet ένα δίκτυο που αποτελείται από 5 switches, το όποιο μέσω του OVX μοιράστηκε σε 4 διαφορετικούς tenants(tid 1,2,3,4). Οι tenants μοιράζονται ανά δύο τα switches του δικτύου εκτός από το κεντρικό switch το οποίο «ανήκει» σε τρεις ενοίκους (Σχήμα 5.4).



Σχήμα 5.4 Διάταξη πειραματικής τοπολογίας 2

Τα switches μοιράζονται στους tenants με τον παρακάτω τρόπο:

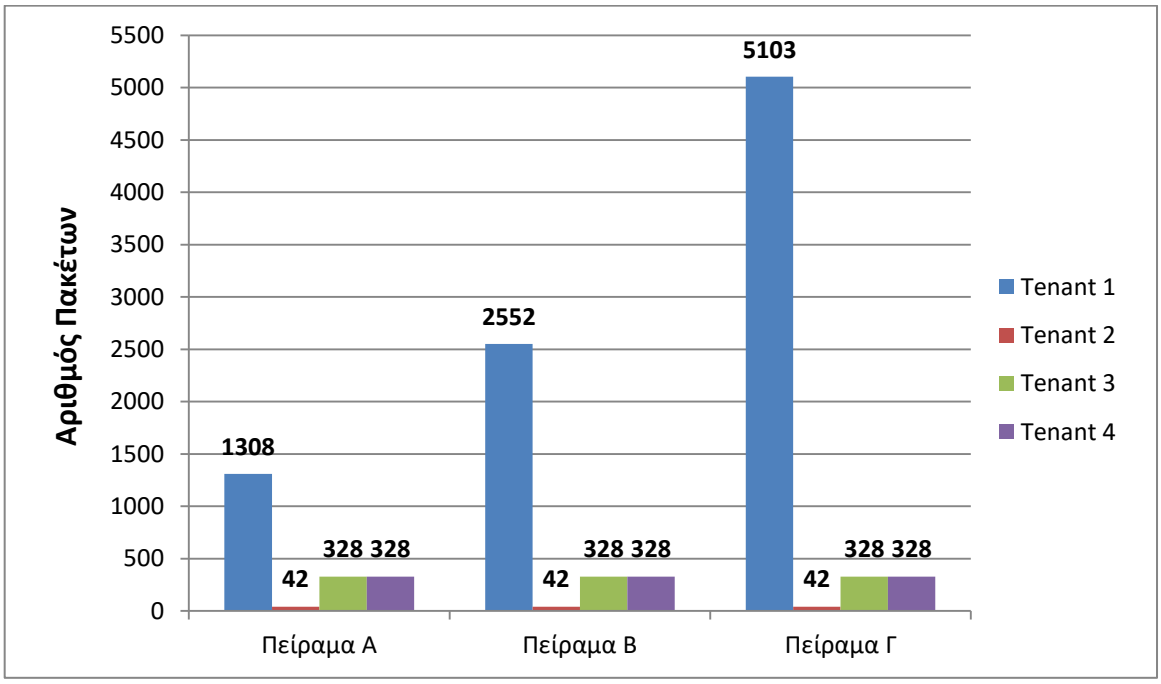
- Switch12 : tenant 1, tenant 2
- Switch123: tenant 1, tenant 2, tenant 3
- Switch13 : tenant1, tenant 3
- Switch23 : tenant 2, tenant 3
- Switch34 : tenant 3, tenant 4

Οι ένοικοι 3,4 παράγουν «κανονική» κίνηση και αποτελούν το σημείο αναφοράς του πειράματος. Ο ένοικος 1 παράγει αυξημένη κίνηση ενώ ο ένοικος 2 μειωμένη. Η κίνηση αποτελείται από πακέτα UDP, τα οποίαπαράγονται μέσω του προγράμματος iperf.

5.3.2 Αποτελέσματα

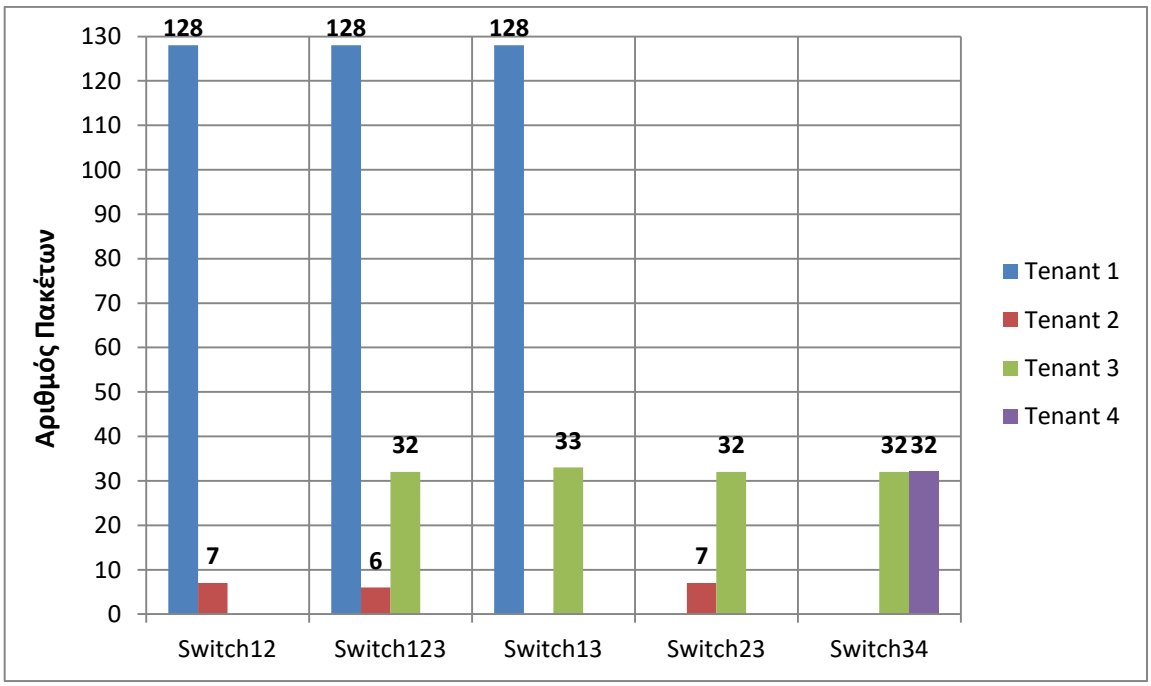
Πραγματοποιήθηκαν 3 πειράματα στα οποία αυξήθηκε σταδιακά ο αριθμός των πακέτων που παρήγαγε ο tenant 1. Στα παρακάτω διάγραμμα (Σχήμα 5.5) παρουσιάζεται ο αριθμός των πακέτων που παράχθηκαν ανά ένοικο.

Καθώς το sampling έχει τυχαίο χαρακτήρα, τα πειράματα επαναλήφθηκαν αρκετές φορές και τα αποτελέσματα που παρατίθενται αποτελούν το μέσο όρο όλων των μετρήσεων.

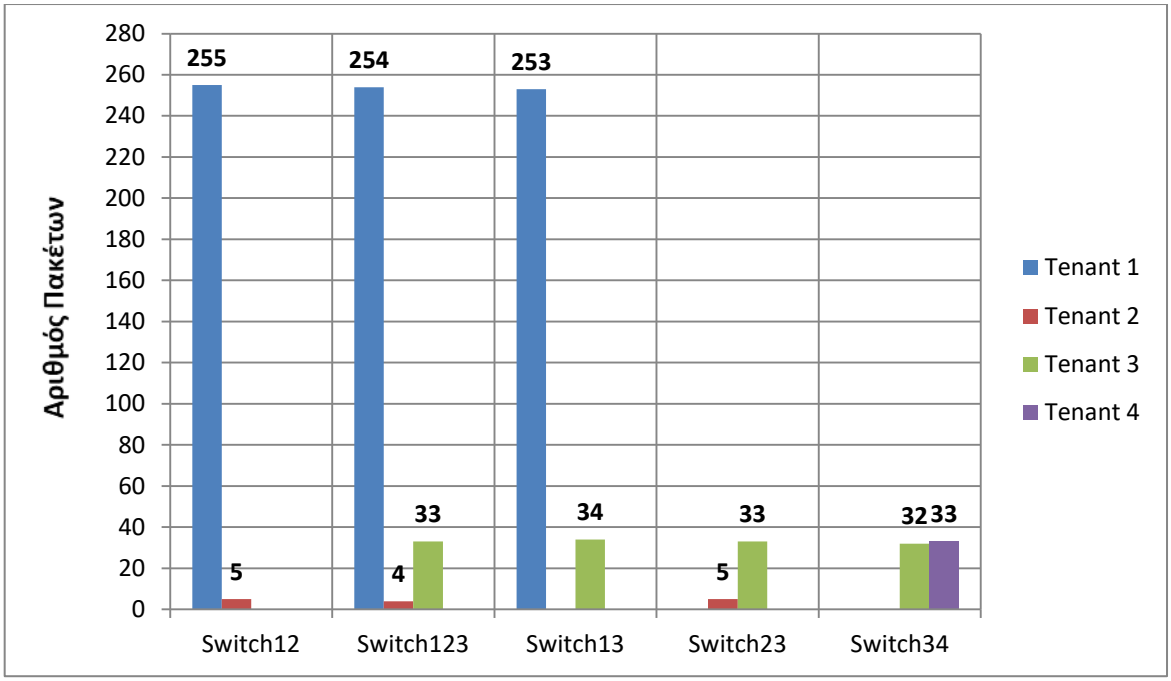


Σχήμα 5.5 Αριθμός πακέτων που παρήγαγαν οι τέσσερις ενοίκους ανά πείραμα

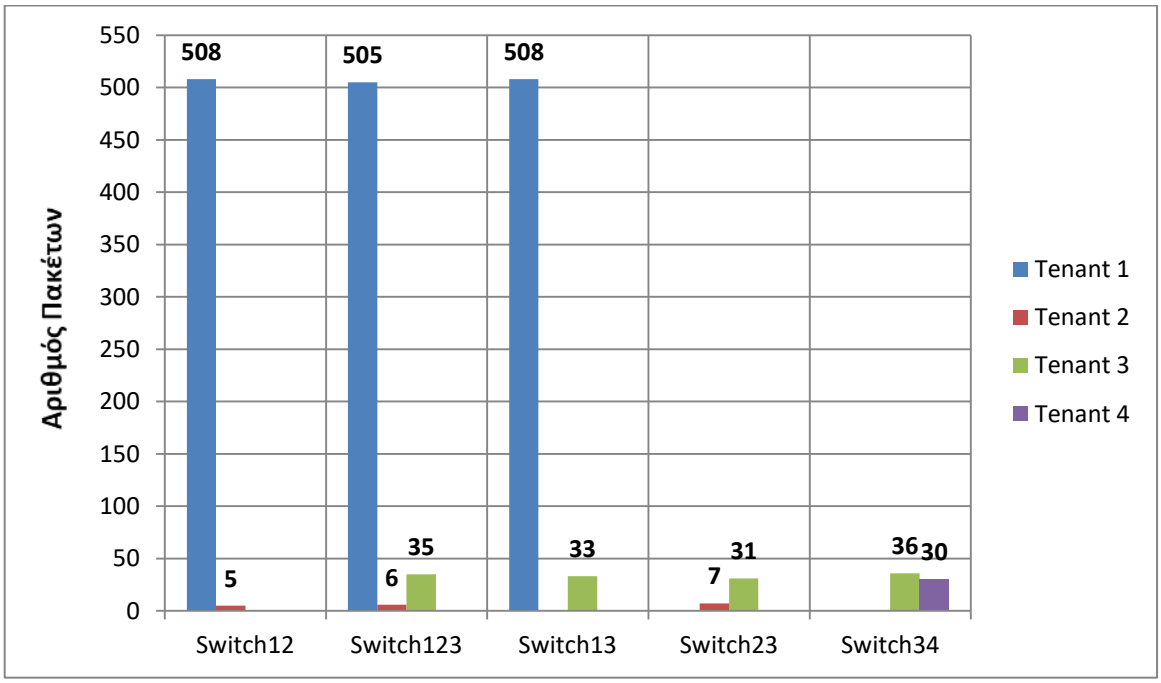
Παρακάτω παρουσιάζονται τα πακέτα που συλλέχθηκαν ανά switch για τα τρία διαφορετικά πειράματα που πραγματοποιήθηκαν.



α)Αριθμός πακέτων που συλλέχθηκαν σε κάθε Switch για Πείραμα Α



β)Αριθμός πακέτων που συλλέχθηκαν σε κάθε Switch για Πείραμα Β



γ)Αριθμός πακέτων που συλλέχθηκαν σε κάθε Switch για Πείραμα Γ

Αρχικά, διαπιστώνεται ότι ο μηχανισμός εντοπίζει τις ροές πακέτων όλων των ενοίκων, ανεξαρτήτως της ποσότητας των δικών τους πακέτων στο δίκτυο. Αυτό συμβαίνει διότι εκμεταλλεύεται τα Openflow μηνύματα μεταξύ των switches και του controller. Ακόμη, όσον αφορά τη δειγματοληψία των πακέτων, παρατηρείτα ότι, παρόλη την αυξημένη ποσότητα πακέτων δικτυακής κίνησης από έναν ένοικο

(tenant 1), ο μηχανισμός είναι ικανός να συλλέξει πακέτα από όλα τα εικονικά δίκτυα. Ο αριθμός των δειγμάτων ανά δίκτυο ικανοποιεί σε μεγάλο βαθμό το ρυθμό δειγματοληψίας που χρησιμοποιήθηκε (1/10).

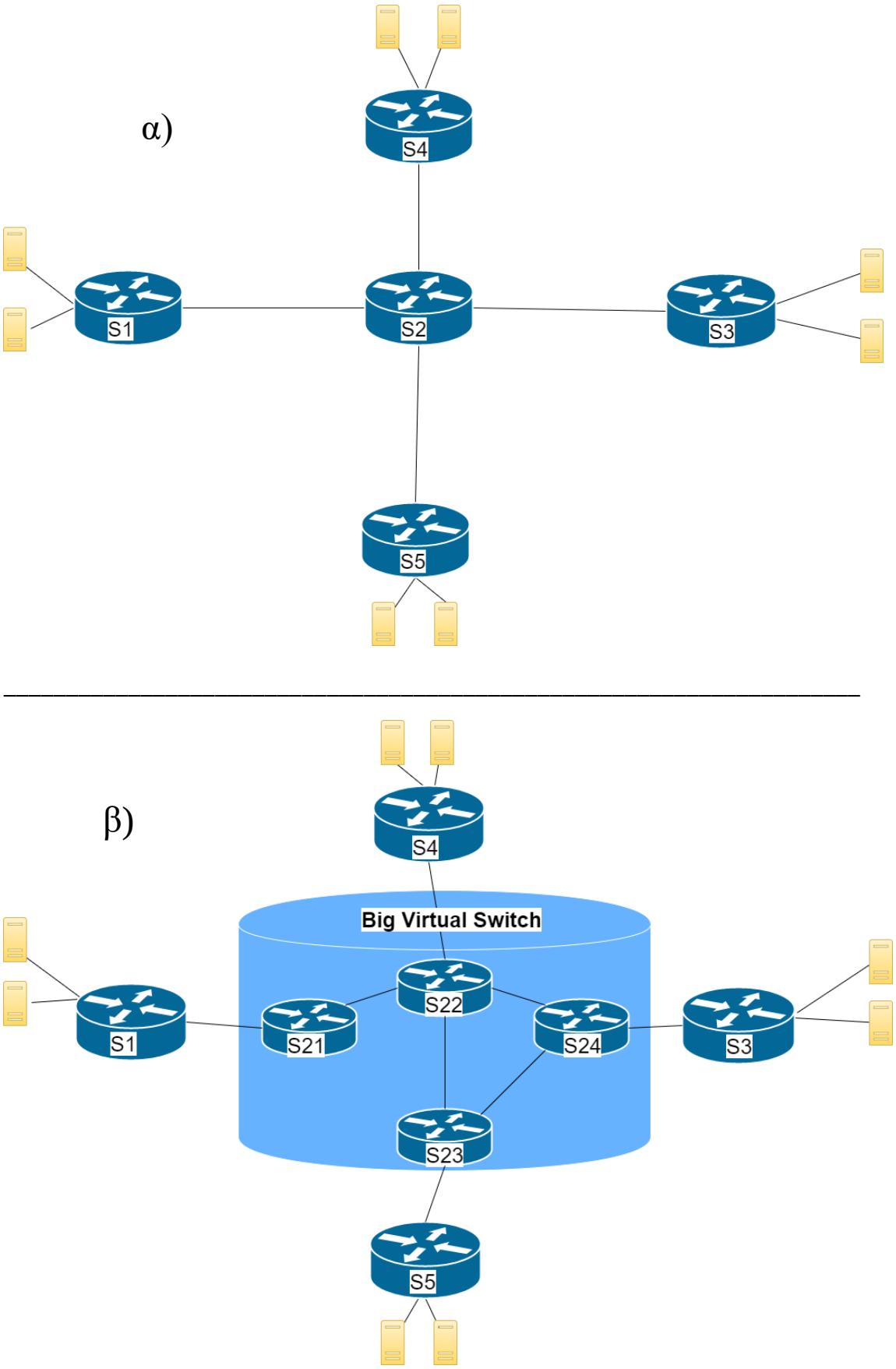
5.4 Abstract Virtual Topologies

Στο συγκεκριμένο σημείο μελετάται η συμπεριφορά του μηχανισμού σε εικονικές τοπολογίες τροποποιημένες ώστε να διαφέρουν από τη φυσική υποδομή. Πιο συγκεκριμένα, μελετάται ένας μηχανισμός που παρέχει ο OVX για την τροποποίηση εικονικών τοπολογιών, το OVXBigSwitch (Κεφάλαιο 2.4.3.2)

Το OVXBigSwitch (BVS) αποτελεί μία δόμη η οποία, στα πλαίσια της εικονικής δικτύωσης, ομαδοποιεί ένα σύνολο από φυσικά switches με σκοπό να εμφανίζονται στο control plane του εκάστοτε ένοικου ως ένα switch. Η συγκεκριμένη δομή χρειάζεται ιδιαίτερη αντιμετώπιση από τον μηχανισμό μας. Οι sampling agents που υπάρχουν στα συγκεκριμένα φυσικά switches συλλέγουν δείγματα τα οποία συλλογικά ταυτίζονται σε ένα εικονικό BVS με αποτέλεσμα να αυξάνεται ο ρυθμός δειγματοληψίας του μηχανισμού για το switch αυτό συγκριτικά με τα υπόλοιπα. Η αύξηση αυτή μπορεί να οδηγήσει σε λανθασμένα συμπεράσματα σχετικά με την ένταση της κίνησης στο συγκεκριμένο κόμβο. Μέσω του πειράματος αυτού μελετάται η συμπεριφορά του μηχανισμού μας σε τέτοιες τροποποιημένες δομές.

5.4.1 Πειραματική διάταξη

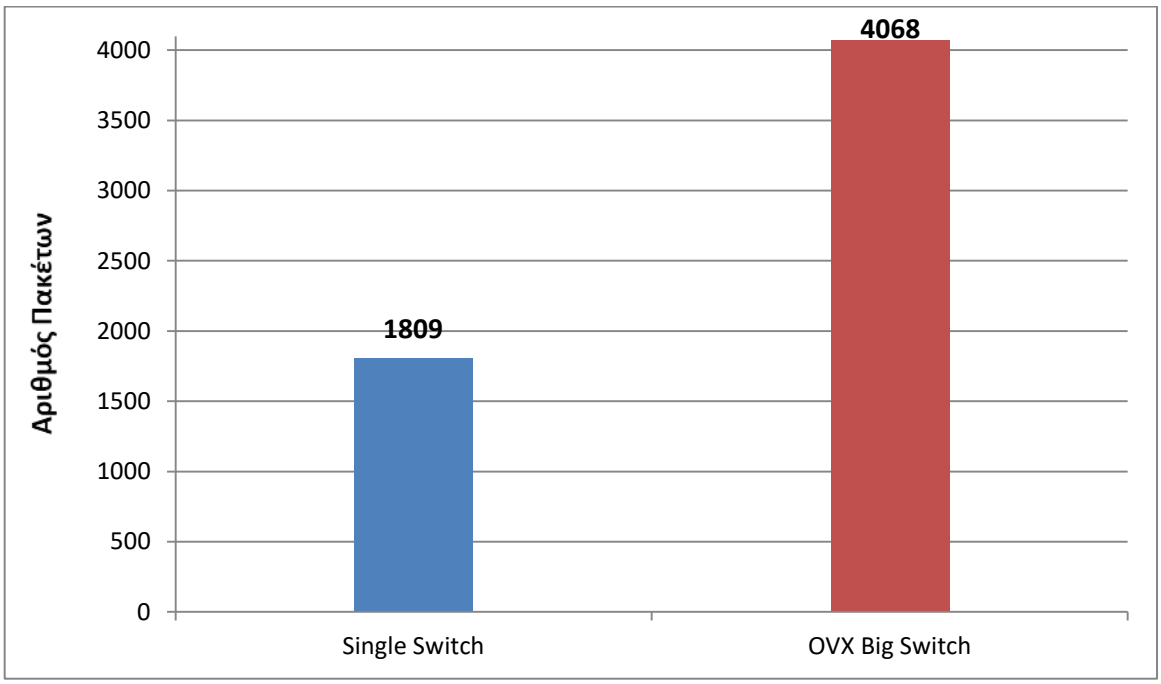
Για τους σκοπούς του πειράματος δημιουργήθηκαν, μέσω του Mininet, οι δύο παρακάτω τοπολογίες (Σχήμα 5.6). Οι τοπολογίες αυτές αποτελούνται από τέσσερα edge switches, τα οποία συνδέονται με δύο host machines το καθένα, και από ένα κεντρικό switch , το οποίο στην τοπολογία Α είναι απλό switch σε αντίθεση με την τοπολογία Β που είναι OVXBigSwitch (BVS) απαρτιζόμενο από τέσσερα φυσικά switches. Αξίζει να σημειωθεί πως ο εικονικός controller, που διαχειρίζεται τις τοπολογίες αυτές, τις αντιλαμβάνεται ως πανομοιότυπες.



Σχήμα 5.6 Διάταξη πειραματική τοπολογίας α) χωρίς BVS β) με BVS

5.4.2 Αποτελέσματα

Μέσω των hosts παράγουμε δικτυακή κίνηση (συγκεκριμένος και σταθερός αριθμός πακέτων UDP) με χρήση του προγράμματος iperf , η οποία διασχίζει τα κεντρικό switches των δύο τοπολογιών. Η κίνηση αυτή παρακολουθείται από το μηχανισμό μας. Στο παρακάτω διάγραμμα παρουσιάζεται ο αριθμός των πακέτων που ανίχνευσε ο μηχανισμός για τα switches αυτά. Καθώς το sampling έχει τυχαίο χαρακτήρα, τα πειράματα επαναλήφθηκαν αρκετές φορές και τα αποτελέσματα που παρατίθενται αποτελούν το μέσο όρο όλων των μετρήσεων.



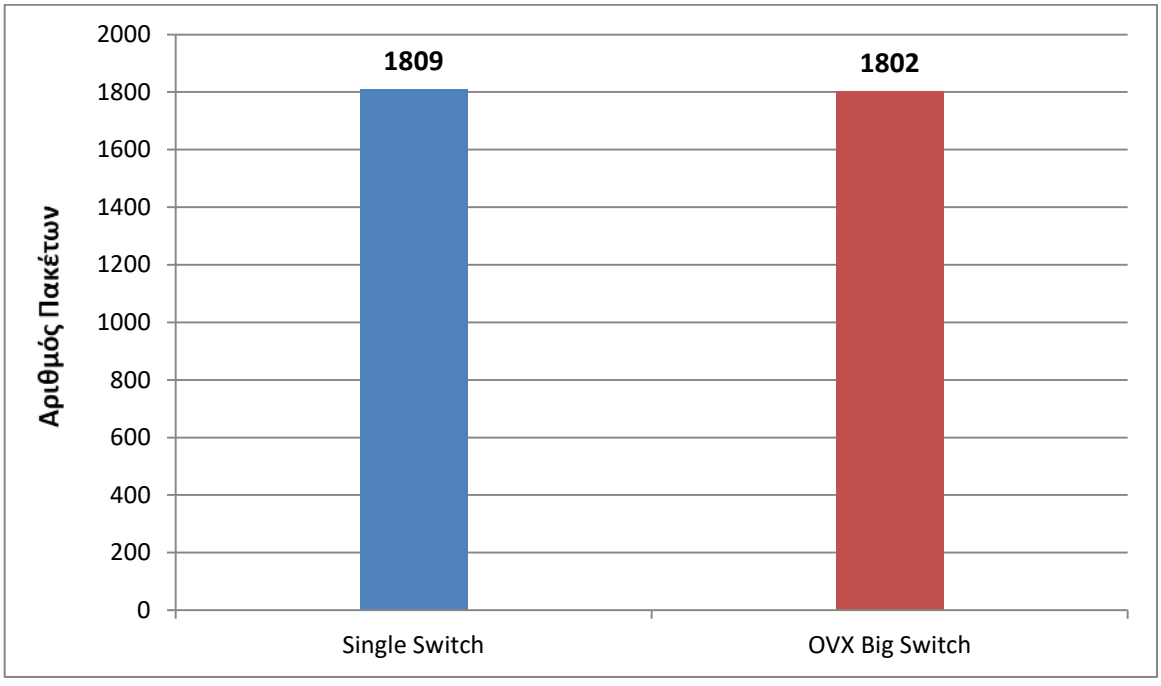
Σχήμα 5.7 Αποτελέσματα συλλογής δειγμάτων σε Single Switch / OVXBigSwitch

Παρατηρούμε πως, όπως ήταν αναμενόμενο, ο αριθμός των πακέτα που συλλέχθηκαν από το BVS είναι σημαντικά μεγαλύτερος από τον αντίστοιχο αριθμό πακέτων στο single switch. Ακόμη συμπεραίνουμε πως, παρόλο που το BVS αποτελείται από τέσσερα switches και κατ' επέκταση από τέσσερις sflow agents, το πλήθος των δειγμάτων στην τοπολογία B δεν είναι τετραπλάσιο από αυτό της τοπολογίας A (4068/1809 ~= 2,25). Αυτό συμβαίνει διότι οι ροές πακέτων στο εσωτερικό του BVS δεν διασχίζουν απαραίτητα το σύνολο των φυσικών switches συνεπώς δεν δειγματοληπτούνται από το σύνολο των sflow agents. Ο αριθμός των sflow agents που δειγματοληπτούν την ίδια ροή σε ένα BVS δεν είναι σταθερός και

εξαρτάται από τη δομή του και τον αλγόριθμο δρομολόγησης των πακέτων στο εσωτερικό του. Ως αποτέλεσμα, ο συλλογισμός της διαίρεσης του πλήθους των δειγμάτων στο OVXBigSwitch με τον αριθμό των switches που το αποτελούν δεν είναι λειτουργικός.

Προκειμένου να αντλούνται αξιόπιστα αποτελέσματα κατα τη συλλογή δικτυακής κίνησης σε τοπολογίες που περιέχουν OVXBigSwitches, ακολουθείται μία διαφορετική προσέγγιση. Τα δείγματα που προέρχονται από τέτοια switches προσμετρούνται στις εγγραφές του Flow Repository μόνο εφόσον συλλεχθούν κατά την είσοδο τους στο BVS, στην περίπτωση δηλαδή που προέρχονται από κάποιο switch εισόδου. Τα δείγματα αυτά διαχωρίζονται από τα υπόλοιπα δείγματα με βάση τη διευθυνσιοδότησης που ακολουθεί η πλατφόρμα OVX, όπως αναφέρθηκε στο 3^{ο} κεφάλαιο (Πίνακας 3.1). Με αυτό τον τρόπο εξασφαλίζεται η δειγματοληψία όλων των ροών από ένα μόνο switch.

Στη συνέχεια επαναλαμβάνεται το πείραμα με χρήση της παραπάνω **επιλεκτικής δειγματοληψίας**. Τα αποτελέσματα από τα δείγματα που συλλέχθηκαν φαίνονται παρακάτω.



Σχήμα 5.8 Αποτελέσματα συλλογής δειγμάτων σε Single Switch / OVXBigSwitch με χρήση επιλεκτικής δειγματολήψίας

Παρατηρούμε ότι ο αριθμός των δειγμάτων στα δύο switches (single και BVS) συγκλίνει, επιτρέποντας την εξαγωγή αξιόπιστων συμπερασμάτων για τη δικτυακή κίνηση σε προσαρμοσμένες εικονικές τοπολογίες που περιέχουν τέτοιες δομές.

6

Επίλογος

Σε αυτό το σημείο παρουσιάζεται μία σύνοψη της διπλωματικής εργασίας και εξάγονται συμπεράσματα βασισμένα στην πειραματική διαδικασία. Ακόμη, προτάσσονται συγκεκριμένες ιδέες με στόχο τη βελτίωση της υλοποίησης αλλά και μελλοντικές επεκτάσεις της.

6.1 Σύνοψη και συμπεράσματα

Στην παρούσα διπλωματική εργασία παρουσιάστηκε ένας μηχανισμός συλλογής δικτυακών δεδομένων σε πλατφόρμες πολλαπλών ενοίκων. Ο συγκεκριμένος μηχανισμός δημιουργει καταχωρήσεις ροών σε ένα Flow Repository εκμεταλλευόμενος την κίνηση σηματοδοσίας των επιμέρους στοιχείων μιας υποδομής SDN και τη χρήση δειγματοληψίας πακέτων στο φυσικό επίπεδο. Οι καταχωρήσεις αποθηκεύονται με τέτοιο τρόπο ώστε να είναι δυνατή η εξαγωγή τους ομαδοποιημένες **ανά ένοικο**, ανά συσκευή και τέλος ανά flow. Επιπλέον, στοχεύοντας στην αποδοτική επικοινωνία του Flow Repository με πολλαπλά components, χρησιμοποιείται σαν διεπαφή εισόδου σε αυτό μία κατανεμημένη ουρά μηνυμάτων (Kafka).

Ως πλατφόρμα πολλαπλών ενοίκων χρησιμοποιήθηκε το OpenVirteX (OVX) καθώς παρέχει στους χρήστες της ιδιαίτερη ευελιξία κατά τον ορισμό ενός εικονικού δικτύου. Η συγκεκριμένη πλατφόρμα χρησιμοποιεί εικονικές απεικονίσεις δικτυακών στοιχείων και

των διευθύνσεων που τους αντιστοιχούν. Οι απεικονίσεις αυτές αντιστοιχίζονται με τα πραγματικά δικτυακά στοιχεία με σκοπό τη δημιουργία εικονικών υποδομών με εξατομικευμένα για κάθε ένοικο χαρακτηριστικά. Ωστόσο, δημιουργείται πρόβλημα στην ικανότητα αντιστοίχισης καταχωρήσεων ροών και δειγμάτων στο μηχανισμό παρακολούθησης. Αυτό συμβαίνει διότι οι τιμές διευθύνσεων κατά την εισαγωγή των δικτυακών ροών στο Flow Repository ήταν διαφορετικές σε σχέση με τις αντίστοιχες τιμές στα πακέτα δειγματοληψίας (OVX-Physical addresses).

Όσον αφορά το παραπάνω πρόβλημα, κατασκευάστηκε μία βιβλιοθήκη λειτουργιών προκειμένου να αποκασταθεί η λειτουργικότητα του μηχανισμού σε τέτοια περιβάλλοντα. Η συγκεκριμένη βιβλιοθήκη (mapper) είναι γραμμένη σε python και βασίστηκε στο API του OVX. Περιέχει τις απαραίτητες λειτουργίες για την επικοινωνία με την πλατφόρμα, την άντληση των απεικονίσεων που διατηρεί για τις διευθύνσεις και την μετάφραση των απαραίτητων διευθύνσεων με σκοπό την αντιστοίχισή τους στις καταχωρήσεις ροών. Με τη χρήση της παραπάνω βιβλιοθήκης, ο μηχανισμός είναι ικανός να συσχετίσει τα διάφορα πακέτα που συλλέγονται στο φυσικό επίπεδο με τις καταχωρήσεις που τους αντιστοιχούν.

Επιπροσθέτως, πραγματοποιήθηκαν μετρήσεις σε περιβάλλον πολλαπλών ενοίκων, στο οποίο οι tenants παράγουν ταυτόχρονα δικτυακή κίνηση σε διαφορετικούς ρυθμούς. Αρχικά, παρατηρήθηκε ότι ο μηχανισμός είναι ικανός να εντοπίσει το σύνολο των ροών που δημιουργούνται και να τις αντιστοιχίσει τους διάφορους ενοίκους. Επιπλέον, παρόλο που συγκεκριμένοι ένοικοι παρήγαγαν σημαντικά μεγαλύτερα επίπεδα κίνησης, αντιστοιχίστηκαν στο Flow Repository πακέτα προερχόμενα από όλα τα εικονικά δίκτυα. Οι ποσότητες των πακέτων που συλλέχθησαν ικανοποιούν σε μεγάλο βαθμό το ρυθμό δειγματοληψίας που χρησιμοποιήθηκε.

Κλείνοντας, μελετήθηκε η συμπεριφορά του μηχανισμού σε τεχνικές δικτύωσης που στοχεύουν στην προσαρμογή εικονικών τοπολογιών. Πιο συγκεκριμένα, αναλύθηκε στο Κεφάλαιο 2 η δομή του Big Virtual Switch (BVS), η οποίο αποτελείται από ένα εικονικό switch πάνω στο οποίο αντιστοιχίζονται πολλαπλοί φυσικοί μεταγωγείς. Όπως φαίνεται στο Κεφάλαιο 5, η δομή αυτή οδηγεί σε αύξηση του ρυθμού δειγματοληψίας πακέτων στο εσωτερικό της. Ακόμη, παρατηρήθηκε ότι η αύξηση αυτή δεν είναι ανάλογη με τον αριθμό των physical switches που περιέχονται, αλλά εξαρτάται από την εσωτερική αρχιτεκτονική του BVS και από τις ροές που διέρχονται κάθε στιγμή. Στη συνέχεια, προτάθηκε μία διαφορετική προσέγγιση για τη δειγματοληψία τέτοιων δομών, κατά την οποία προσμετρούνται δείγματα από συγκεκριμένα switches κάθε φορά. Όπως φανερώνουν τα

αποτελέσματα στο Κεφάλαιο 5, η χρήση επιλεκτικής δειγματοληψίας διορθώνει το παραπάνω πρόβλημα και οδηγεί στην εξαγωγή ασφαλών αποτελεσμάτων.

6.2 Μελλοντικές επεκτάσεις

Στ συγκεκριμένο σημείο αναφέρονται διάφορες προτάσεις για τη βελτίωση του μηχανισμού παρακολούθησης, καθώς και μελλοντικές επεκτάσεις που μπορούν να υλοποιηθούν.

Αρχικά, σημαντική βελτίωση θα μπορούσε να αποτελέσει η χρήση ενός συστήματος αποθήκευσης, διαχείρισης και ανάλυσης μεγάλου όγκου δεδομένων, με στόχο την καλύτερη αποθήκευση των δεδομένων και την ευκολότερη πρόσβαση σε αυτά. Μία τέτοια προσθήκη είναι εξαιρετικά σημαντική κατά την παρακολούθηση υποδομών μεγάλης κλίμακας, καθώς ο όγκος των δεδομένων αποτελεί κομβικό παράγοντα στη λειτουργικότητα του μηχανισμού. Ένα επιπλέον χαρακτηριστικό που προσδίδει ένα τέτοιο σύστημα είναι η κλιμακωσιμότητα. Διατηρώντας τις καταχωρήσεις σε μία κεντρική δομή εκτός server, δίνεται η δυνατότητα σε πολλαπλούς servers να λειτουργούν ταυτόχρονα κατά μήκος ενός δικτύου μοιράζοντας το φόρτο εργασίας. Τονίζεται πως η ομαδοποίηση των δεδομένων ανά ένοικο και ροή είναι σημαντικό να διατηρηθεί καθώς αποτελεί βασικό χαρακτηριστικό για τη λειτουργικότητα του μηχανισμού.

Επιπλέον, ο μηχανισμός μπορεί να συνεισφέρει στη βελτίωση των υπηρεσιών που παρέχει η πλατφόρμα OpenVirteX. Όπως αναφέρθηκε σε προηγούμενα κεφάλαια, το OVX παρέχει στους ενοίκους τη δυνατότητα δημιουργίας σύνθετων μεταγωγέων (Big Virtual Switch) και γραμμών (OVX Link), οι οποίοι περικλύουν στο εσωτερικό τους πολλαπλά δικτυακά στοιχεία. Οι ένοικοι αντιλαμβάνονται τέτοια στοιχεία σαν απλά switches ή links, “αδιαφορώντας” για τη διαχείριση του εσωτερικού δικτύου που περιέχουν. Τη συγκεκριμένη διαχείριση αναλαμβάνει το OVX. Οι πληροφορίες που συλλέγει ο μηχανισμός και συσχετίζονται με τις παραπάνω δομές θα μπορούσαν να ανατροφοδοτούνται στο OpenVirteX, στοχεύοντας στην καλύτερη δρομολόγηση πακέτων και γενικότερα διαχείρισης του εσωτερικού των δομών αυτών.

Κλείνοντας, σημαντική επέκταση είναι η λειτουργία του μηχανισμού παρακολούθησης σε συνδυασμό με εφαρμογές ανάλυσης δεδομένων και παρουσίασης αποτελεσμάτων. Στόχος θα είναι η τροφοδότηση των δεδομένων που συλλέγονται στις εφαρμογές αυτές για την εξαγωγή πληροφοριών δικτυακής κίνησης. Στο πλαίσιο αυτό είναι δυνατή η δημιουργία ενός ενοποιημένου εργαλείου παρακολούθησης, διαθέσιμο σε κάθε ένοικο για τη διαχείριση του δικτύου του.

7

Βιβλιογραφία

[1] Diego Kreut, Fernando M. V. Ramos, Member, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky and Steve Uhlig, “Software-Defined-Networking: A Comprehensive Survey”

[2] Open Networking Foundation, “Software-Defined Networking: The New Norm Of Networks”

[3] Open Networking Foundation, “ OpenFlow-enabled SDN and Network Functions Virtualization”

[4] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner, “ OpenFlow: Enabling Innovation in Campus Networks “

[5] Masayoshi Kobayashi, Srini Seetharaman, Guru Parulkar, Guido Appenzellerq, Joseph Little, Johan van Reijendam, Paul Weissmann, Nick McKeown, ”Maturing of OpenFlow and Software Defined Networking through Deployments”

[6] Nick Feamster, Jennifer Rexford, Ellen Zegura, “ The Road to SDN: An Intellectual History of Programmable Networks“

[7] Open Networking Lab, POX Wiki - https://openflow.stanford.edu/display/ONL/POX+Wiki

[8] Andreas Blenk, Arsany Basta, Martin Reisslein and Wolfgang Kellerer, “Survey on Network Virtualization Hypervisors for Software Defined Networking”

[9] Rob Sherwood, Glen Gibb, Kok-Kiong Yap,Guido Appenzellery, Martin Casado, Nick McKeown, Guru Parulkar, “FlowVisor: A Network Virtualization Layer”

[10] Ali Al-Shabibi, Marc De Leenheer, Matteo Gerolay, Ayaka Koshibe, William Snow, Guru Parulkar, “OpenVirteX: A Network Hypervisor”

[11] Ali Al-Shabibi, Marc De Leenheer, Ayaka Koshibe, Guru Parulkar, Bill Snow, Matteo Gerola and Elio Salvadori, “OpenVirteX: Make Your Virtual SDNs Programmable”

[12] OpenVirteX documentation - http://ovx.onlab.us/documentation

[13] OpenVirteX tutorial - http://ovx.onlab.us/getting-started/tutorial

[14] sFlow, “Traffic Monitoring using sFlow” – www.sflow.org

[15] K. Giotis, C. Argyropoulos, G. Androulidakis , D. Kalogeras, V. Maglaris, “Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments”

[16] Georgios Androulidakis, Vassilis Chatzigiannakis, and Symeon Papavassiliou, “Network Anomaly Detection and Classification via Opportunistic Sampling”

[17] Niels L. M. van Adrichem, Christian Doerr and Fernando A. Kuipers, “OpenNetMon: Network Monitoring in OpenFlow Software-Defined Networks”

[18] Kafka documentation - https://kafka.apache.org/documentation

[19] kafka-python - https://github.com/dpkp/kafka-python

[20] Kafka tutorial - https://kafka.apache.org/quickstart

[21] RabbitMQ - http://www.rabbitmq.com/documentation.html

[22] CloudAMQP, “What is message queueing” – https://www.cloudamqp.com/blog/2014-12-03-what-is-message-queuing.html

[23] Adam Pavlidis, “Σχεδιασμός και ανάπτυξη μηχανισμών συλλογής δεδομένων σε πειραματικές πλατφόρμες για το Internet του μέλλοντος”

[24] iPerf - The ultimate speed test tool for TCP, UDP and SCTP - https://iperf.fr

[25] Mininet: An Instant Virtual Network on your Laptop (or other PC) - http://mininet.org

[26] TOP manual page - http://www.manpages.info/linux/top.1.html

[27] ubuntu - http://www.ubuntu.com

[28] Vyas Sekar, Michael K. Reiter, Walter Willinger, Hui Zhang, Ramana Rao Kompella, David G. Andersen, cSAMP: A System for Network-Wide Flow Monitoring

[29] Andrew R. Curtis Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, “DevoFlow: Scaling Flow Management for High-Performance Networks”

[30] Lavanya Jose, Minlan Yu and Jennifer Rexford, “Online Measurement of Large Traffic Aggregates on Commodity Switches”