Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων Πληροφορικης

Παρακολούθηση σηματοδοσίας SIP με χρήση SDN & SmartNIC

Διπλωματικη Εργασια

του

ΕΜΜΑΝΟΥΗΛ Ι. ΣΤΡΑΤΟΓΙΑΝΝΗ

Επιβλέπων: Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2021

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής

Παρακολούθηση σηματοδοσίας SIP με χρήση SDN & SmartNIC

Διπλωματικη Εργασια

του

ΕΜΜΑΝΟΥΗΛ Ι. ΣΤΡΑΤΟΓΙΑΝΝΗ

Επιβλέπων: Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 8η Οκτωβρίου 2021.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Ευστάθιος Συκάς	Παπαβασιλείου Συμεών	Ιωάννα Ρουσσάκη
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκ. Καθηγήτρια Ε.Μ.Π.

Αθήνα, Οκτώβριος 2021

(Υπογραφή)

.....

Εμμανουηλ Στρατογιαννης

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

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής

Ευχαριστίες

Το παρόν έργο διεξήχθη στις υποδομές του Εργαστηρίου Δικτύων Υπολογιστών του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ του Εθνικού Μετσόβιου Πολυτεχνείου, υπό την επίβλεψη και καθοδήγηση του καθηγητή Ευστάθιου Συκά. Η διπλωματική αυτή θεμελιώνεται στις κυριότερες αρχές που διέπουν τη βελτιστοποίηση δικτύων, αλλά ταυτοχρόνως διερευνά την ομορφιά και το μεγαλείο των πιο σύγχρονων διαδικτυακών τεχνολογιών του σήμερα.

Με την περάτωση της διπλωματικής μου εργασίας θα ήθελα να ευχαριστήσω ιδιαιτέρως τον καθηγητή και επιβλέποντα μου Ευστάθιο Συκά για την επιστημονική καθοδήγηση και τη συνεχή υποστήριξη του καθ΄όλη τη διάρκεια της εκπόνησης της εργασίας, καθώς και τον κ. Πάρι Χαραλάμπου για τις εποικοδομητικές υποδείξεις του και την πολύτιμη βοήθεια του. Ένα μεγάλο ευχαριστώ αξίζει και σε όλους τους υπόλοιπους καθηγητές μου, για τις πολύτιμες γνώσεις που μου μετέδωσαν κατά τη διάρκεια των σπουδών μου.

Ευχαριστώ τον αδερφό και την αδερφή μου που συνεισέφεραν, ο καθένας με τον δικό του τρόπο, στην προσπάθειά μου αυτή, καθώς επίσης και τους γονείς μου, οι οποίοι με ενθάρρυναν καθημερινά για να κυνηγώ τα όνειρά μου και να αγαπώ καθετί με το οποίο καταπιάνομαι. Τέλος, αφιερώνω το παρόν έργο σε όλους τους φίλους και συγγενείς, για τη συνεχή συμπαράσταση τους. Σας είμαι ευγνώμων για τον άνθρωπο στον οποίο έχω εξελιχθεί σήμερα.

Περίληψη

Με την εφαρμογή τεχνολογιών NFV (Network Function Virtualization) η εκτέλεση των λειτουργιών των στοιχείων της αρχιτεκτονικής IMS γίνεται πλέον σε εικονικές μηχανές (VM) στις υποδομές cloud του παρόχου. Η νέα αυτή αρχιτεκτονική επιφέρει μεγάλες δυσκολίες στην αποτελεσματική παρακολούθηση των μηνυμάτων σηματοδοσίας καθώς δεν είναι πλέον δυνατή η χρήση παραδοσιακών τεχνικών για την αντιγραφή των IP πακέτων στο φυσικό μέσο.

Με τη χρήση αρχιτεκτονικών SDN σε Data Centers και την ανάπτυξη SDN εφαρμογών καθίσταται δυνατή η «έξυπνη» αλλαγή των πινάκων προώθησης στους μεταγωγείς για την προώθηση πακέτων. Παράλληλα με την εισαγωγή των SmartNIC είναι δυνατή η εκτέλεση των λειτουργιών αντιγραφής των πακέτων σύμφωνα με προγραμματιστικά κριτήρια με βέλτιστο τρόπο.

Σκοπός της εργασίας είναι η ανάπτυξη εφαρμογής για την αποτελεσματική παρακολούθηση SIP μηνυμάτων σε αρχιτεκτονική IMS με τη χρήση τεχνολογιών SDN και SmartNIC. Ο ελεγκτής SDN θα είναι υπεύθυνος για την προώθηση των SIP μηνυμάτων στον κατάλληλο εξυπηρετητή ενώ το Smartnic θα είναι υπεύθυνο για την προώθηση των μηνυμάτων σε εξωτερικό σύστημα καταγραφής.

Λέξεις Κλειδιά

IMS, NFV, SDN, ODL, REST API, OpenFlow, Agilio SmartNIC, OVS, Virtual Machine, KVM, Kamailio.

Abstract

With the application of NFV (Network Function Virtualization) technologies, the module functions of the IMS architecture are now executed in virtual machines (VM) in the cloud infrastructures of the service provider. This new architecture brings new challenges in effective monitoring of signaling messages as it is no longer possible to copy IP packets, using traditional techniques, in the forwarding plane.

With the SDN architecture for packet forwarding within the Data Center, it is possible to "smartly" change the forwarding tables of every switch by developing SDN applications. Along with the help of SmartNICs, it is possible to perform packet mirroring operations in an optimal way.

The purpose of this thesis is to develop an application for effective monitoring of SIP messages in IMS architecture using SDN and SmartNIC technologies. The SDN controller will be responsible for forwarding the SIP messages to the appropriate server while Smartnic for forwarding the messages to an external monitoring system.

Keywords

IMS, NFV, SDN, ODL, REST API, OpenFlow, Agilio SmartNIC, OVS, Virtual Machine, KVM, Kamailio.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Παραθέσεων

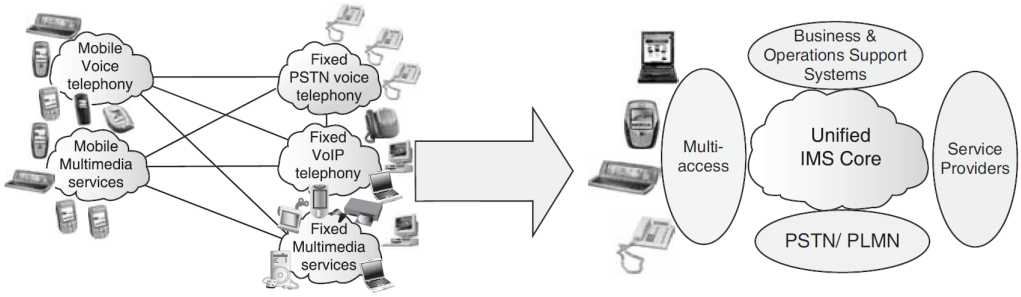
Κεφάλαιο 1

Εισαγωγή

1.1 Η εξέλιξη του δικτύου τηλεπικοινωνιών

Τα σταθερά και κινητά δίκτυα έχουν περάσει από σημαντικές μεταβάσεις τα τελευταία 20 χρόνια. Στον κόσμο των κινητών, τα συστήματα πρώτης γενιάς (1G) πρωτοεμφανίστηκαν στα μέσα της δεκαετίας του 1980. Αυτά τα δίκτυα προσέφεραν βασικές υπηρεσίες στους χρήστες, κυρίως ομιλία, μέσω μετάδοσης αναλογικών σημάτων, έως ότου αντικαταστάθηκαν από τα συστήματα δεύτερης γενιάς (2G), τη δεκαετία του 1990, προσθέτοντας ορισμένες υπηρεσίες δεδομένων και πιο εξελιγμένες συμπληρωματικές υπηρεσίες στους χρήστες, όπως το SMS και το MMS, μέσω μετάδοσης ψηφιακών σημάτων, αξιοποιώντας πιο αποδοτικά το φάσμα ραδιοσυχνοτήτων. Σταθμό στην ιστορία αποτέλεσε η τρίτη γενιά (3G και 3.5G) καθώς και η εξέλιξή του (4G LTE) που επιτρέπει ταχύτερους ρυθμούς μετάδοσης δεδομένων και ακόμα περισσότερες υπηρεσίες πολυμέσων, όπως τη σύνδεση στο διαδίκτυο και την κρυπτογραφημένη επικοινωνία μεταξύ χρηστών. Μέχρι σήμερα, το παραδοσιακό δημόσιο τηλεφωνικό δίκτυο μεταγωγής (PSTN) και τα δίκτυα ολοκληρωμένων υπηρεσιών ψηφιακού δικτύου (ISDN) έχουν κυριαρχήσει στην παραδοσιακή επικοινωνία φωνής και βίντεο.

Τα τελευταία όμως χρόνια η χρήση του Διαδικτύου έχει σημειώσει ραγδαία άνοδο με αποτέλεσμα όλο και περισσότεροι χρήστες να εκμεταλλεύονται την ταχύτερη και φθηνότερη σύνδεση στο Διαδίκτυο, με ασύμμετρη ψηφιακή γραμμή συνδρομητών (ADSL) και ταχεία VDSL γραμμή. Αυτοί οι τύποι συνδέσεων επιτρέπουν συνεχή συνδεσιμότητα στο Διαδίκτυο, κάτι που είναι απαραίτητο για τους χρήστες που χρησιμοποιούν μέσα επικοινωνίας πραγματικού χρόνου - π.χ. εφαρμογές συνομιλίας, διαδικτυακά παιχνίδια, Voice over IP (VoIP) κα. Προς το παρόν βιώνουμε τη γρήγορη σύγκλιση σταθερών και κινητών κόσμων καθώς η διείσδυση των κινητών συσκευών αυξάνεται σε ετήσια βάση. Πλέον χρησιμοποιούνται έξυπνες φορητές συσκευές SmartPhones εξοπλισμένες με μεγάλες οθόνες υψηλής ακρίβειας, ενσωματωμένες κάμερες και πολλούς διαθέσιμους πόρους επεξεργαστή για εφαρμογές που είναι μονίμως συνδεδεμένες στο διαδίκτυο. Οι εφαρμογές έπαψαν πλέον να είναι απομονωμένες οντότητες που ανταλλάσσουν πληροφορίες μόνο με το περιβάλλον εργασίας χρήστη, αντιθέτως είναι peer-to-peer οντότητες, οι οποίες διευκολύνουν την κοινή



Σχήμα 1.1: Η μετάβαση του παραδοσιακού δικτύου PSTN σε IMS (Πηγή: [1])

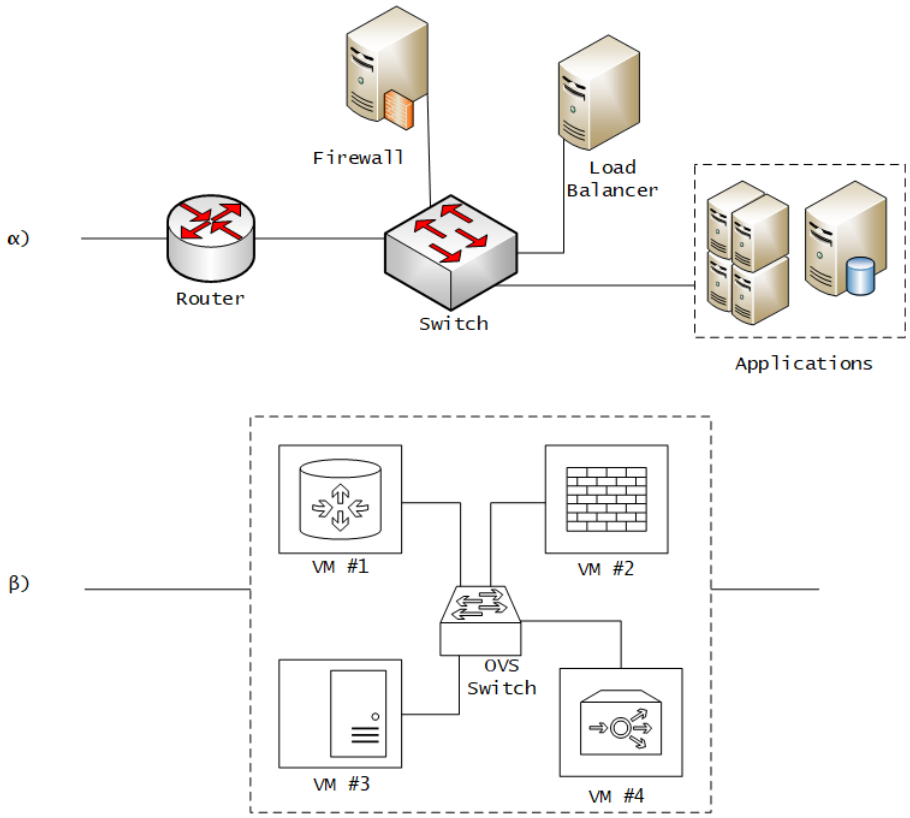
χρήση: κοινή περιήγηση, κοινόχρηστους πίνακες, κοινή εμπειρία παιχνιδιού, κοινόχρηστη αμφίδρομη ραδιοφωνική συνεδρία (δηλ. Push to Talk Over Cellular) κα.

Η κλήση ενός αριθμού και η ομιλία έχουν θεωρηθεί ήδη ως στενό υποσύνολο της δικτύωσης. Η ικανότητα να δημιουργηθεί μια ομότιμη σύνδεση μεταξύ συσκευών με δυνατότητα πρωτοκόλλου Διαδικτύου (IP) είναι το βασικό συστατικό που απαιτείται. Αυτό το νέο πρότυπο των επικοινωνιών ξεπερνά όλες τις δυνατότητες του παραδοσιακού τρόπου μεταγωγής κυκλωμάτων. Για αυτόν τον λόγο απαιτείται μια νέα αρχιτεκτονική δικτύου ικανή να καλύψει όλες τις απαιτήσεις του χρήστη και να ενοποιήσει το δίκτυο σταθερής και κινητής. Απαιτείται δηλαδή ένα μεγάλο άλμα μετάβασης από το απλό παραδοσιακό τηλεφωνικό δίκτυο σε ένα καθολικό ευρυζωνικό δίκτυο. Μια τέτοια μετάβαση απεικονίζεται στο Σχ. 1.1), όπου όλες οι, έως τώρα, ξεχωριστές μονάδες (πχ. σταθερό τηλέφωνο, κινητή συσκευή, FAX, κα.) συνδέονται μεταξύ τους σε ένα ενιαίο νέφος που ονομάζεται IMS. [1]

1.2 Το πρόβλημα των κέντρων δεδομένων

Με την εξέλιξη της τεχνολογίας και τον καταιγισμό τόσης πληροφορίας, εγείρονται πολλά ερωτήματα σχετικά με την ορθή διαχείριση των δεδομένων στα κέντρα δεδομένων. Οι απαιτήσεις των χρηστών για υπηρεσίες είναι πλέον τεράστιες όπως και η ανάγκη για εφαρμογές πραγματικού χρόνου. Μια παραδοσιακή προσέγγιση επίλυσης θα ήταν η εξ ολοκλήρου αναβάθμιση του υλικού σε ένα κέντρο δεδομένων, ωστόσο αυτό δεν αποτελεί λύση, εφόσον είναι τρομακτικά κοστοβόρα και καθόλου πρακτική. Έτσι οι πάροχοι τηλεπικοινωνιακών υπηρεσιών αναγκάστηκαν να αναζητήσουν νέους τρόπους παροχής αυτών των υπηρεσιών πιο ευέλικτους λαμβάνοντας υπόψιν την εξοικονόμηση OPEX και CAPEX. Το NFV είναι μια λύση εικονικοποίησης του εξοπλισμού του δικτύου, όπου εκτελείται ανοιχτό λογισμικό και ως εκ τούτου επιτρέπει στους παρόχους τηλεπικοινωνιακών υπηρεσιών να γίνουν πιο ευέλικτοι, γρηγορότεροι σε υπηρεσίες καινοτομίας και να μειώσουν το κόστος λειτουργίας και συντήρησης. Είναι σαφές ότι το NFV, μαζί με τα στενά συνδεδεμένα και συμπληρωματικά πεδία του SDN και του Cloud Computing ίσως αποτελέσουν μεγάλο μέρος της μελλοντικής παροχής τηλεπικοινωνιακών υπηρεσιών.

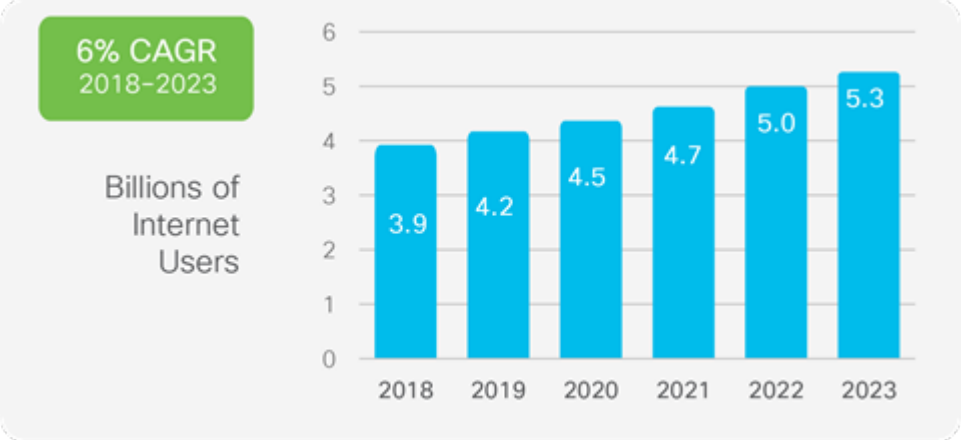
Σε ένα παραδοσιακό δίκτυο (βλ. Σχήμα 1.2α) ο φυσικός μεταγωγέας είναι υπεύθυνος για την δικτυακή κίνηση μεταξύ των φυσικών μέσων. Οποιαδήποτε τροποποίηση της κίνησης καθορίζεται αποκλειστικά από αυτόν και όλα τα μηχανήματα θα πρέπει να είναι συνδεδεμένα στο φυσικό επίπεδο. Αντιθέτως, σε ένα εικονικό δίκτυο NFV (βλ. Σχήμα 1.2β), ο server είναι αυτός που φιλοξενεί εικονικές μηχανές VMs και η διασύνδεση τους γίνεται με έναν εικονικό μεταγωγέα σε επίπεδο λογισμικού. Η νέα αυτή αρχιτεκτονική εικονικοποίησης παρότι μας εξασφαλίζει ένα ανοιχτού κώδικα σύστημα πιο συμπαγές και επεκτάσιμο, δημιουργεί προκλήσεις σχετικά με τη διαχείριση και αποτελεσματική παρακολούθηση των πακέτων, καθώς δεν είναι πλέον δυνατή η χρήση παραδοσιακών τεχνικών για την αντιγραφή των IP πακέτων στο φυσικό μέσο. [2]



Σχήμα 1.2: α) Παραδοσιακό δίκτυο β) Εικονικό δίκτυο αρχιτεκτονικής NFV

Τα κέντρα δεδομένων ανέκαθεν αντιμετώπιζαν προκλήσεις μαζικής αποθήκευσης και μεγάλου φόρτου επεξεργασίας δεδομένων. Το 2022 η παγκόσμια κίνηση δεδομένων σε κέντρα δεδομένων υπολογίζεται να ξεπεράσει τα 20.6 ZettaBytes με συνολικά 5 δισεκατομμύρια ενεργούς χρήστες (βλ. Σχήμα 1.3) [3] [4].

Συνεπώς η σωστή διαχείριση ενός τόσου μεγάλου όγκου δεδομένων είναι πλέον απαραίτητη. Επιπλέον με τα κατά παραγγελία cloud μοντέλα, οι πάροχοι cloud και τα παραδοσιακά κέντρα δεδομένων αντιμετωπίζουν σοβαρά προβλήματα στην αγορά. Λύσεις που βασίζονται σε λογισμικό (software-based) μπορούν να προσφέρουν ευελιξία και ταχύτητα όσον αφορά τη δικτυακή κίνηση. Συγκεκριμένα η Netronome SmartNIC, που θα



Σχήμα 1.3: 5 Δισεκατομμύρια χρήστες στο διαδίκτυο το 2022 (Πηγή: [4])

χρησιμοποιηθεί και στην παρούσα διπλωματική, είναι ιδανική για δίκτυα που βασίζονται σε εξυπηρετητές, προσφέροντας πιο ασφαλείς εικονικές μηχανές VMs ανά εξυπηρετητή και εξοικονομώντας κύκλους επεξεργαστικής ισχύος.

Τα κέντρα δεδομένων cloud εξυπηρετούν εκατοντάδες χιλιάδες ¨ενοικιαστές' και διαχειρίζονται εφαρμογές από εκατομμύρια VMs που τρέχουν σε χιλιάδες εξυπηρετητές. Για να επιτρέψουν την ομαλή μετάβαση των εφαρμογών στο cloud, οι πάροχοι υπηρεσιών προσφέρουν στους πελάτες τη δυνατότητα να φιλοξενούνται σε δικό τους δίκτυο με τις δικές του πολιτικές ασφαλείας. Ένα ολοένα αυξανόμενο σύνολο από VMs και εφαρμογές, υπεύθυνο για το 75% της κίνησης εντός του κέντρου δεδομένων, προκαλεί προβλήματα ασφαλείας καθώς κάθε κανόνας ασφαλείας στο firewall (τείχος προστασίας) πρέπει να σχετίζεται άμεσα με το αντίστοιχο VM. Τελικά αυτό προκαλεί μία έκρηξη από μια πληθώρα δικτύων, πολιτικών ασφαλείας και κλειδιών για τα οποία απαιτείται ενδελεχή επιτήρηση και αποτελεσματική διαχείριση. Η ανάγκη για τέτοιου είδους 'granular' πολιτικές προστασίας, πιο κοντά στα VMs και τις εφαρμογές, οδήγησε στην υλοποίηση των δικτυακών εργασιών που χρησιμοποιούν τον εικονικό μεταγωγέα OVS τον οποίο θα αναλύσουμε στα επόμενα κεφάλαια. Κάποιες εσωτερικές τυπικές δικτυακές εργασίες είναι η ενθυλάκωση σε τούνελ (tunneling), η εξισορρόπηση φορτίου και η επεξεργασία ροών (αντιστοίχιση, μέτρηση, ενέργεια κα.).

Ωστόσο με συνδέσεις 10,25,40 και 100GbE στους εξυπηρετητές, κάτι τέτοιο θα ήταν απαγορευτικό αν αναλογιστούμε τους πόρους από πυρήνες CPU και την κατανάλωση σε ισχύ. Εκτός των άλλων, οι αστοχίες στις κρυφές μνήμες μπορούν να προκαλέσουν ανεπιθύμητες μεταβολές και διακυμάνσεις στην απόδοση. Γι΄ αυτό το σκοπό, η SmartNIC είναι σχεδιασμένη να επιλύσει τέτοιες προκλήσεις στην απόδοση και την κλιμακωσιμότητα, χρησιμοποιώντας μία μέθοδο μείωσης του φορτίου (offloading) στο υλικό. Με αυτόν τον τρόπο βελτιώνεται σημαντικά η απόδοση καθώς απελευθερώνει χρήσιμους για τον εξυπηρετητή CPU πόρους και πετυχαίνει ταυτόχρονα πιο προστατευμένες εικονικές μηχανές στον εξυπηρετητή.

1.3 Οργάνωση του τόμου

Η δομή της παρούσας εργασίας είναι οργανωμένη σε έξι επί μέρους κεφάλαια:

1. Στο **2ο Κεφάλαιο** παρουσιάζεται η απαραίτητη θεωρία για την πλήρη κατανόηση της υλοποίησης, δηλαδή του εργαστηριακού μέρους. Σε αυτό το κεφάλαιο αναφέρονται σημαντικές έννοιες σχετικά με την εικονικοποίηση και την επιτάχυνση του δικτύου.

2. Στο **3ο Κεφάλαιο** περιγράφεται αναλυτικά η αρχιτεκτονική του δικτύου που θα εξετάσουμε, καθώς επίσης και όλα τα απαραίτητα εργαλεία λογισμικού και ο εξοπλισμός που χρησιμοποιήθηκαν για την ολοκλήρωση του πειραματικού μέρους.

3. Στο **4ο Κεφάλαιο** γίνεται διεξοδική ανάλυση της διαδικασίας που ακολουθήθηκε για την εγκατάσταση του εξοπλισμού και παρουσιάζεται βήμα-βήμα η διαδικασία υλοποίησης του συστήματος μαζί με τις σχετικές επεξηγήσεις και τον αντίστοιχο κώδικα.

4. Στο **5ο Κεφάλαιο** διεξάγεται το πειραματικό μέρος και παρουσιάζονται τα αποτελέσματα του σε μορφή πίνακα και γραφημάτων.

5. Στο **6ο Κεφάλαιο** ολοκληρώνεται η εργασία με τον επίλογο, συνοψίζοντας όλα τα συμπεράσματα που αναφέρθηκαν και παρουσιάζοντας νέες μεθόδους και προκλήσεις για μελλοντικές προεκτάσεις.

Στο τέλος της παρούσας διπλωματικής εργασίας παρατίθεται η βιβλιογραφία που χρησιμοποιήθηκε κατά τη συγγραφή της, ο κατάλογος ακρωνύμων με όλα τα ακρώνυμα που αναφέρθηκαν και το παράρτημα με τον σχετικό κώδικα.

Κεφάλαιο 2

Θεωρητικό & Τεχνολογικό υπόβαθρο

2.1 Εικονικοποίηση κέντρων δεδομένων

2.1.1 Network Functions Virtualization

Το NFV είναι η εικονικοποίηση δικτυακών εργασιών-υπηρεσιών όπως, δρομολογητές, τείχη προστασίας και εξισορροπιστές φορτίου, που παραδοσιακά εκτελούνταν σε ιδιόκτητο υλικό. Αυτές οι υπηρεσίες εκτελούνται είτε σαν εικονικές μηχανές σε γενικής χρήσης εξυπηρετητές (πχ. COTS) είτε σαν containers σε cloud native περιβάλλοντα. Το COTS, στην επιστήμη των υπολογιστών, είναι το εμπορικώς διαθέσιμο "από το ράφι" προϊόν (Commercial off-the-self) το οποίο προσαρμόζεται στις ανάγκες της εκάστοτε επιχείρησης και μπορεί να εξυπηρετήσει διάφορες λύσεις σε αντίθεση με έτοιμα συστήματα ενσωματωμένου λογισμικού. Αυτό επιτρέπει στους παρόχους υπηρεσιών να εκτελούν υπηρεσίες δικτύου σε δικούς τους εξυπηρετητές, χωρίς την ανάγκη για ακριβά ιδιόκτητα υλικά συγκεκριμένων λειτουργιών. [5]

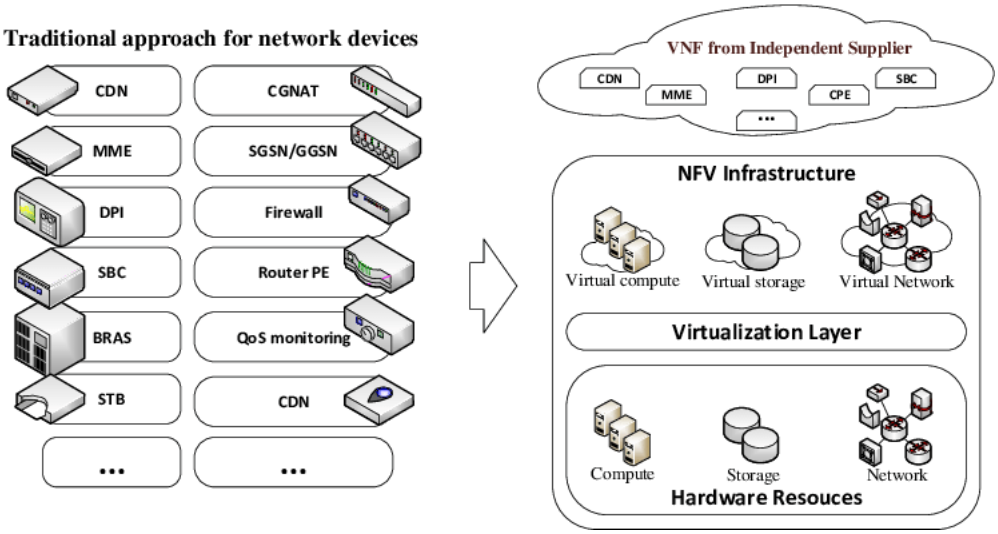
Με το NFV δεν χρειάζεται να έχουμε ξεχωριστό υλικό για κάθε δικτυακή εργασία (βλ. Σχήμα 2.1). Το NFV βελτιώνει την επεκτασιμότητα και την ευελιξία του δικτύου με το να επιτρέπει στους παρόχους υπηρεσιών να χρησιμοποιούν καινούργιες δικτυακές υπηρεσίες και εφαρμογές κατά παραγγελία, χωρίς να απαιτούνται επιπλέον πόροι υλικού.

2.1.1.1 NFV Αρχιτεκτονική

Η NFV αρχιτεκτονική προτάθηκε από το Ευρωπαϊκό Ινστιτούτο Τηλεπικοινωνιακών Προτύπων (ETSI) για να βοηθήσει τον ορισμό προτύπων υλοποίησης του NFV. Κάθε στοιχείο της αρχιτεκτονικής βασίζεται σε αυτά τα πρότυπα για να επιτευχθεί καλύτερη σταθερότητα και διαλειτουργικότητα.

Η NFV αρχιτεκτονική αποτελείται από:

- Τις εικονικοποιημένες δικτυακές εργασίες (VNFs) – όπως μετάδοση αρχείων, υπηρεσίες καταλόγου και ρυθμίσεις IP.



Σχήμα 2.1: Προσέγγιση NFV αρχιτεκτονικής έναντι της παραδοσιακής (Πηγή: [6])

- Την υποδομή εικονικοποίησης δικτυακών εργασιών (NFVi) – αποτελείται από δομικά στοιχεία επεξεργασίας, αποθήκευσης, δικτύωσης σε μία πλατφόρμα εικονικοποίησης όπως αυτή του KVM ή πλατφόρμα διαχείρισης περιεκτών (containers για την εκτέλεση δικτυακών εφαρμογών.
- Τη διαχείριση, αυτοματοποίηση και ενορχήστρωση δικτύου (MANO) – παρέχει το πλαίσιο για τη διαχείριση της NFV υποδομής και την προμήθεια νέων εργασιών VNFs.

2.1.1.2 Πλεονεκτήματα του NFV

Με το NFV, οι πάροχοι υπηρεσιών μπορούν να εκτελούν δικτυακές εργασίες σε τυπικό υλικό αντί για εξειδικευμένο. Επίσης, επειδή οι δικτυακές εργασίες είναι εικονικοποιημένες, πολλές εργασίες μπορούν να εκτελούνται σε έναν μόνο εξυπηρετητή. Αυτό σημαίνει ότι αξιοποιείται μικρότερος φυσικός χώρος με λιγότερο υλικό, επιτυγχάνοντας έτσι μια σημαντική μείωση σε κατανάλωση πόρων και ισχύος.

Το NFV παρέχει στους παρόχους την ευελιξία να εκτελούν VNFs σε διαφορετικούς εξυπηρετητές ή να τους τροποποιούν κατά απαίτηση. Αυτό οδηγεί σε ταχύτερες εφαρμογές και υπηρεσίες. Για παράδειγμα, αν ένας πελάτης κάνει αίτηση για μια νέα δικτυακή εργασία, μπορούν να εγκαταστήσουν ένα VM για να τον εξυπηρετήσουν και αν κάποια στιγμή αργότερα πάψει να τη χρειάζεται τότε αυτή καταργείται. Αυτό μπορεί επίσης να χρησιμοποιηθεί και για ασφαλείς δοκιμές καινούργιων υπηρεσιών [5].

2.1.2 Software-Defined Networking

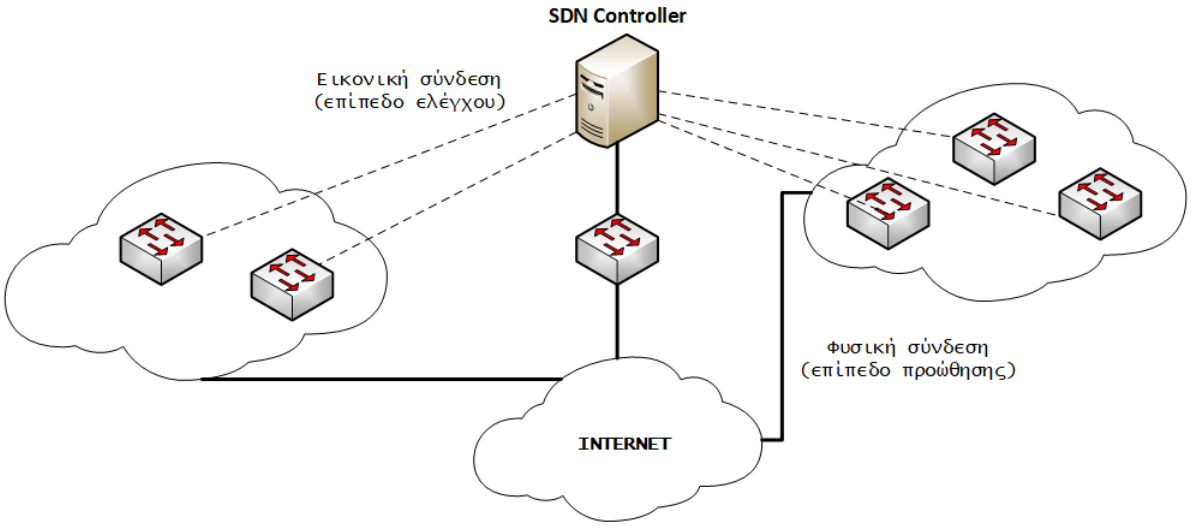
Το SDN -Δικτύωση βασισμένη σε λογισμικό- είναι μια αρχιτεκτονική η οποία σχεδιάστηκε με σκοπό την εύκολη και αποδοτικότερη διαχείριση δικτύου [7]. Τα τελευταία χρόνια το SDN αποτελεί μια πολύ καλή λύση σε αναδυόμενες τεχνολογίες δικτύου. Σε

αντίθεση με τα παραδοσιακά δίκτυα που χρησιμοποιούν ενσωματωμένο hardware και software για να κατευθύνουν την κίνηση στο διαδίκτυο μέσω routers και switches, το SDN δίνει τη δυνατότητα μιας κεντρικής διαχείρισης και βελτίωσης της υπάρχουσας υποδομής δικτύου, μέσω κώδικα προγραμματισμού.

Η αρχική ιδέα του SDN ήταν η εικονικοποίηση του δικτύου διαχωρίζοντας το επίπεδο ελέγχου (control plane) από το επίπεδο προώθησης δεδομένων (forwarding-data plane), προσφέροντας μια ευελιξία στη διαχείριση του δικτύου (βλ. Σχήμα 2.2). Αυτό επιτυγχάνεται μέσω ενός ελεγκτή Controller που τρέχει κάποιο εξειδικευμένο λογισμικό, μέσω του οποίου πραγματοποιείται η διαχείριση όλης της κυκλοφορίας των δεδομένων, όπως για παράδειγμα η προώθηση πακέτων IP κατά βούληση σε κάποιο data center.

2.1.2.1 Πλεονεκτήματα του SDN

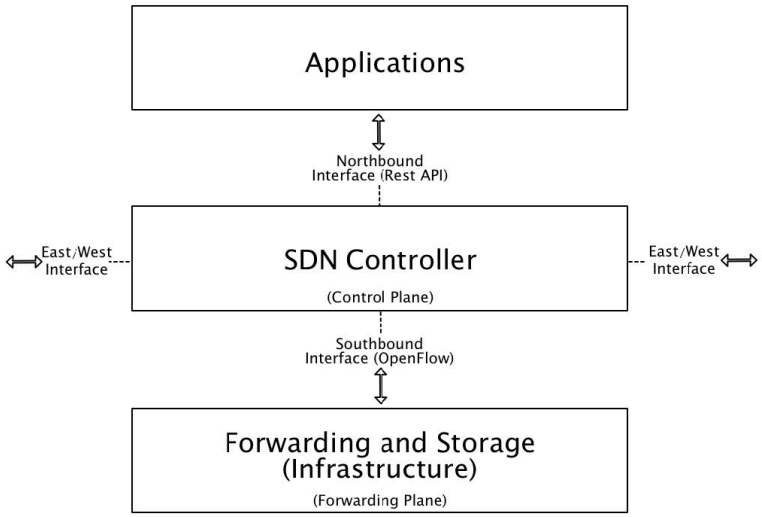
Η εικονικοποίηση του δικτύου φέρει πολλά πλεονεκτήματα. Τα δίκτυα μπορούν να αλλάζουν δυναμικά-αυτόματα και όχι στατικά-χειροκίνητα όπως γινόταν μέχρι σήμερα, ανάλογα με τις απαιτήσεις σε bandwidth και latency και την σειρά προτεραιότητας κάθε εφαρμογής. Επίσης δίνεται η δυνατότητα για εγκατάσταση πολιτικών ασφαλείας σε κάθε δίκτυο μεμονωμένα, πράγμα το οποίο προσφέρει ένα ακόμη σημαντικό μέτρο ασφαλείας. Το μεγαλύτερο, όμως, πλεονέκτημα είναι η δυνατότητα αποδέσμευσης του εξαντλητικού υπολογιστικού πόρου από την υπάρχουσα υποδομή. Έτσι ο υπάρχων δικτυακός εξοπλισμός χρησιμοποιείται κυρίως για την προώθηση των πακέτων δεχόμενος εντολές από μια και μόνο κεντρική διαχείριση, τον SDN Controller. Σήμερα το SDN έχει εξελιχθεί, επιτρέποντας τη χρήση και εκτός των Data Centers, όπως σε επιχειρήσεις ή ακόμα και σε εφαρμογές IoT.



Σχήμα 2.2: Τοπολογία SDN αρχιτεκτονικής

2.1.2.2 Διεπαφές του SDN

Στο Σχήμα 2.3 απεικονίζεται ένα μοντέλο SDN. Σχετικά με τον SDN Controller υπάρχουν 3 διεπαφές: Northbound, Southbound και East/West.



Σχήμα 2.3: Μοντέλο αναφοράς SDN (Πηγή: [8])

Η **Northbound** διαχειρίζεται την επικοινωνία μεταξύ των εφαρμογών και του Controller μέσω εφαρμογών υψηλού επιπέδου APIs (Application Programming Interface). Ένα παράδειγμα είναι το REST API που χρησιμοποιείται στον OpenDayLight (ODL) Controller και μας επιτρέπει να εφαρμόζουμε ή να διαγράφουμε καταχωρήσεις ροών (flow entries).

Η **Southbound** αποτελεί τον δίαυλο επικοινωνίας μεταξύ του Controller και της υποδομής δικτύου όπου μεταφέρονται πληροφορίες από τον Controller στο επίπεδο δεδομένων Forwarding Plane και το αντίστροφο. Η κίνηση των πακέτων IP καθορίζεται από τους πίνακες ροών (Flow Tables) και συνήθως χρησιμοποιείται το πρωτόκολλο OpenFlow, μέσω του οποίου επεξεργαζόμαστε τις καταχωρήσεις ροών.

Τέλος η **East/West** επιτρέπει τη διασύνδεση περαιτέρω SDN Controllers με σκοπό την ανταλλαγή πληροφοριών για QoS, πολιτικών κα. [8]

2.2 Εικονικοποίηση Δικτύου

2.2.1 Ο εικονικός μεταγωγέας OVS

Το Open vSwitch, ή αλλιώς OVS, είναι ένα ευρέως χρησιμοποιημένο παράδειγμα ενός εικονικού μεταγωγέα σε δίκτυο βασιζόμενο σε εξυπηρετητές και ελεγχόμενο με SDN. Τα **πλεονεκτήματα** του OVS σε τέτοιου είδους δίκτυα είναι πλέον αποδεδειγμένα:

- Ευελιξία χάριν του λογισμικού

- Έλεγχος των διαδικασιών στις διαδρομές δεδομένων (datapaths)
- Πρωτοφανή γρήγορα χαρακτηριστικά
- Όλα τα οφέλη του οικοσυστήματος ανοιχτού κώδικα.

Το OVS είναι μια ανοιχτού κώδικα υλοποίηση ενός εικονικού πολύ-επίπεδου μεταγωγέα (switch). Ο κύριος σκοπός του OVS είναι να παρέχει ένα σωρό μεταγωγής (switching stack) σε περιβάλλον εικονικοποίησης υλικού, υποστηρίζοντας πολλαπλά πρωτόκολλα και πρότυπα δικτύων. Το OVS είναι ιδανικό για χρήση ως εικονικό switch σε εικονικές μηχανές (VMs), προσφέροντας τον τυπικό έλεγχο και ορατές διεπαφές στο επίπεδο εικονικού δικτύου, υποστηρίζοντας επίσης πολλαπλές τεχνολογίες εικονικοποίησης που βασίζονται σε Linux, όπως το Xen/XenServer, KVM, VirtualBox.[9, 10]

Παρακάτω παρουσιάζονται μερικά χρήσιμα τεχνικά χαρακτηριστικά:

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ:

- Standard 802.1Q VLAN μοντέλο με trunk και access ports
- NIC bonding με ή χωρίς LACP σε upstream switch
- NetFlow, sFlow(R), και mirroring για αυξημένο έλεγχο κίνησης
- QoS (Quality of Service) και policing
- Geneve, GRE, VXLAN, STT, και LISP tunneling
- 802.1ag διαχείριση σφάλματος συνδεσιμότητας
- OpenFlow 1.0 με πολλά extensions
- Προώθηση υψηλής απόδοσης χρησιμοποιώντας τον πυρήνα του Linux

ΤΑ ΚΥΡΙΑ ΣΥΣΤΑΤΙΚΑ:

- **ovs-vswitchd**, ο daemon που υλοποιεί τη λειτουργία του switch μαζί με τον πυρήνα του Linux για switching βασιζόμενο σε ροές (flows).
- **ovsdb-server,** ένας ελαφρύς εξυπηρετητής βάσης δεδομένων τον οποίο ρωτάει το ovs-vswitchd για να λάβει τις ρυθμίσεις του.
- **ovs-dpctl,** ένα εργαλείο για τη ρύθμιση της δομής του πυρήνα του switch.
- **ovs-vsctl,** ένα χρήσιμο εργαλείο για να παίρνουμε πληροφορίες από το ovs-vswitchd και να το ενημερώνουμε.
- **ovs-appctl,** ένα εργαλείο το οποίο στέλνει εντολές σε ενεργά Open vSwitch daemons.

- **ovs-ofctl**, ένα εργαλείο για να παίρνουμε πληροφορίες και να ελέγχουμε τα OpenFlow switches και controllers.

- **ovs-pki**, ένα εργαλείο για τη δημιουργία και διαχείριση της υποδομής δημοσίων κλειδιών για OpenFlow switches.

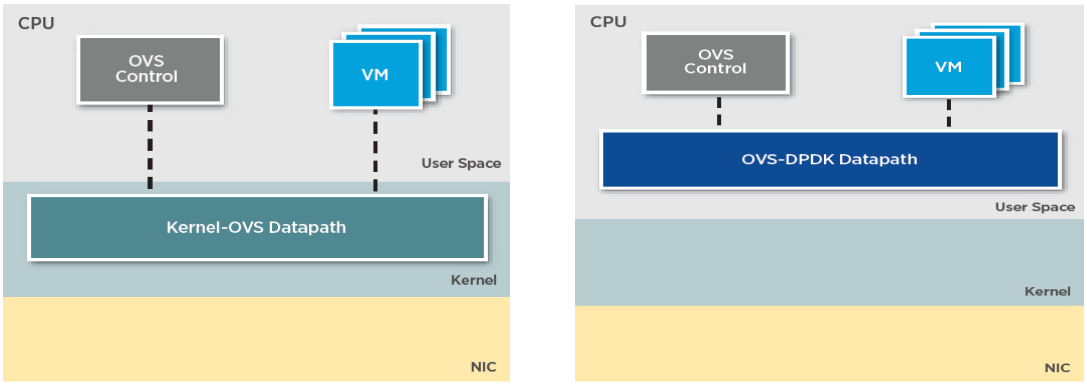
2.2.2 Οι διαδρομές δεδομένων (datapaths) του OVS

Ο πυρήνας OVS (Kernel-OVS) είναι η περισσότερο χρησιμοποιημένη διαδρομή OVS. Ο πυρήνας OVS υλοποιήθηκε ως μια μηχανή προώθησης με εντολές αντιστοίχισης/ενέργειας βασισμένη σε εισερχόμενες, τροποποιημένες ή διαγραμμένες ροές στον χώρο χρήστη (User Space). Το 2012, το OVS ενισχύθηκε περαιτέρω με μία ακόμη διαδρομή δεδομένων στον χώρο χρήστη βασισμένο σε ένα σετ εργαλείων ανάπτυξης στο επίπεδο των δεδομένων, το DPDK (Data Plane Development Kit). Η προσθήκη του OVS-DPDK βελτίωσε την απόδοση δημιουργώντας όμως και κάποιες προκλήσεις. Πιο συγκεκριμένα το OVS-DPDK παρακάμπτει τη δικτυακή στοίβα (Networking Stack) του πυρήνα του Λειτουργικού Συστήματος Linux, χρειάζεται επιπλέον λογισμικό τρίτων και τέλος καθορίζει το μοντέλο ασφαλείας για την πρόσβαση στον χώρο του χρήστη. Με αυτόν τον τρόπο οι εφαρμογές DPDK είναι δύσκολο να ρυθμιστούν βέλτιστα και ενώ υπάρχουν λύσεις διαχείρισης OVS-DPDK, η αποσφαλμάτωση αποτελεί πρόκληση λόγω έλλειψης πρόσβασης σε εργαλεία αντίστοιχα με αυτά που θα χρησιμοποιούσαμε στον πυρήνα. Έχει γίνει σαφής η ανάγκη για μια καλύτερη λύση. Έτσι, ύστερα από 6 χρόνια παρουσιάζεται το OVS-TC (Traffic Control). Το OVS-TC είναι OVS που χρησιμοποιεί ελεγκτή κίνησης βασισμένο στον πυρήνα, και βελτιώνει τόσο το Kernel-OVS όσο και το OVS-DPDK, παρέχοντας μια, αντίθετα με το ρεύμα, λύση για επιτάχυνση στο υλικό. Παρακάτω θα αναλύσουμε τα τρία μοντέλα και τους λόγους για επιτάχυνση στο υλικό.

2.2.2.1 Kernel-OVS Διαδρομή Δεδομένων

Το 2009, το OVS έγινε ο πιο ευρέως χρησιμοποιούμενος εικονικός μεταγωγέας (vSwitch) σε περιβάλλον Linux. Η τυπική OVS αρχιτεκτονική περιέχει στοιχεία στον χώρο του χρήστη καθώς και του πυρήνα. Ο δαίμονας μεταγωγέα εκτελείται στον χώρο του χρήστη και ελέγχει τη διαδρομή των πακέτων που γίνεται μέσω του πυρήνα (βλ. Σχήμα 2.4α′).

Σε μια λογισμικής προσέγγισης λύση, ο πυρήνας δεν είναι ιδανικός για εικονική μεταγωγή, επειδή παραχωρεί μικρά χρονικά κβάντα σε διεργασίες και τα διαχειρίζεται όπως οποιαδήποτε άλλη διεργασία του συστήματος. Με αποτέλεσμα, η αδράνεια να είναι μεγαλύτερη από την πραγματική εκτέλεση της υπηρεσίας, απασχολώντας άσκοπα την κεντρική μονάδα επεξεργασίας (CPU). Επιπλέον η εικονική μεταγωγή απαιτεί συχνές, ανά πακέτο, κλήσεις συστήματος το οποίο αναγκάζει το vSwitch να παραχωρήσει τη CPU στο λειτουργικό σύστημα, έτσι ώστε αυτό να εκτελέσει τις απαραίτητες I/O εργασίες. Η διαδικασία αυτή λοιπόν οδηγεί σε υποβαθμισμένη απόδοση του δικτύου.



(α′) Kernel OVS Διαδρομή Δεδομένων (β′) OVS-DPDK Διαδρομή Δεδομένων

Σχήμα 2.4: Οι 2 OVS Διαδρομές Δεδομένων (Πηγή: [11])

2.2.2.2 OVS-DPDK Διαδρομή Δεδομένων

Το OVS-DPDK είναι μια αξιόλογη προσπάθεια επίλυσης των θεμελιωδών περιορισμών του Kernel-OVS. Υλοποιώντας την OVS διαδρομή δεδομένων στον χώρο του χρήστη, ο DPDK οδηγός παραδίδει τα πακέτα απευθείας στην ειδική εφαρμογή στον χώρο του χρήστη, παρακάμπτοντας τη στοίβα του πυρήνα Linux. Αυτό αποκλείει οποιαδήποτε υπερφόρτωση της στοίβας και επιτρέπει περαιτέρω βελτιστοποιήσεις για το vSwitch, όπως φόρτωση πακέτων απευθείας στις προσωρινές μνήμες και ομαδοποιημένη επεξεργασία. Η DPDK κοινότητα τακτικά παρέχει βελτιστοποιήσεις για την εγκατάσταση του OVS-DPDK σε κάρτες δικτύων (NICs) (βλ. Σχήμα 2.4β′).

Το OVS-DPDK επιτρέπει την επιτάχυνση μέσω λογισμικού, και σε μερικές περιπτώσεις, ο χρήστης μπορεί να το παραμετροποιήσει και να λύσει το τεράστιο πρόβλημα απόδοσης που εμφανίζεται στο Kernel-OVS. Το DPDK γενικά χρησιμοποιεί συγκεκριμένους λογικούς πυρήνες για να ενισχύσει επαρκώς την απόδοση του δικτύου, το οποίο όμως δημιουργεί άλλες προκλήσεις σχετικά με την κλιμακωσιμότητα, αν είναι να χρησιμοποιηθεί ως επιταχυντής με χρήση λογισμικού. Επιπλέον το OVS-DPDK δεν αποτελεί μέρος του πυρήνα του Linux οπότε και αυτή η λύση είναι απαγορευτική σε συστήματα υψηλών απαιτήσεων.

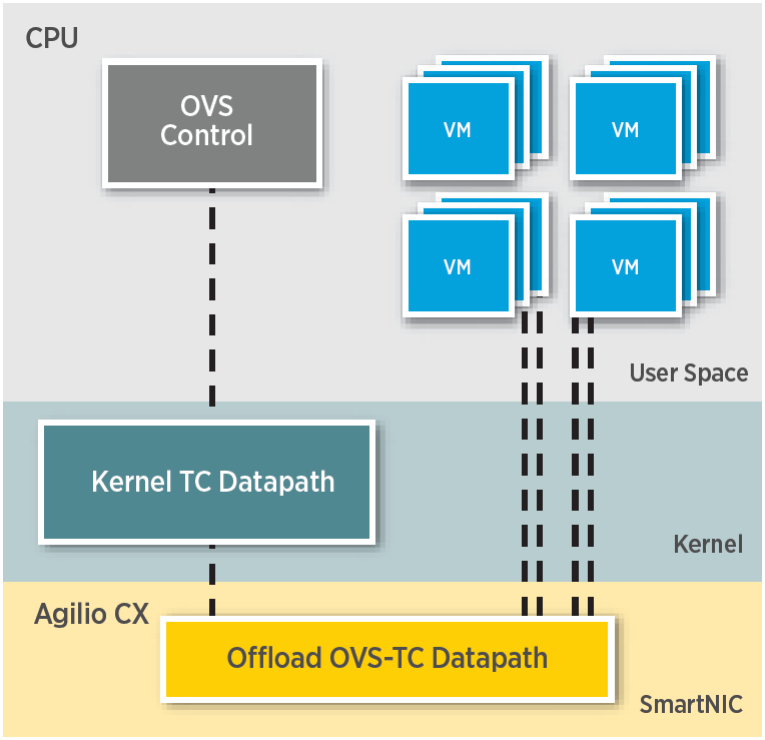
2.2.2.3 OVS-TC Διαδρομή Δεδομένων

Το Linux TC (Traffic Control) είναι ένας μηχανισμός στον πυρήνα υπεύθυνος για την ταξινόμηση πακέτων. Το OVS-TC επιτρέπει την αντιστοίχιση σε μια πληθώρα από προκαθορισμένα στοιχεία κλειδιά. Για παράδειγμα ο χρήστης μπορεί να αντιστοιχίσει IP διευθύνσεις, UDP/TCP πόρτες, μετα-δεδομένα κα. Επίσης, όπως το OVS, έτσι και το OVS-TC διαθέτει τη δυνατότητα τροποποίησης, προώθησης ή απόρριψης πακέτων. Τέλος είναι σημαντικό να αναφέρουμε την επέκταση του TC, το TC Flower το οποίο μας επιτρέπει τη μείωση φορτίου ροών και ενεργειών κατευθείαν από το υλικό.

Η γραμμή εντολών TC παρέχει ένα σύνολο εργαλείων για ρύθμιση πειθαρχιών αναμονής, ταξινομητών και ενεργειών. Ο ταξινομητής TC Flower σε συνδυασμό με τις ενέργειες,

μπορεί να χρησιμοποιηθεί για να παρέχει συμπεριφορές αντιστοίχισης/ενέργειας , όπως έκανε και το Kernel-OVS και το OVS-DPDK. Το OVS αξιοποιεί την TC Flower διαδρομή δεδομένων για να πετύχει επιτάχυνση υλικού (βλ. Σχήμα 2.5).

Οι πάροχοι υπηρεσιών έχουν ανάγκη από έναν κλιμακώσιμο εικονικό μεταγωγέα (vSwitch), και πλέον υπάρχει μια ανοιχτού κώδικα, ριζοσπαστική και συμβατή με τον πυρήνα λύση, το OVS-TC το οποίο διατηρεί όλα τα προνόμια του Kernel-OVS και του OVS-DPDK. Τέλος, οι επιταχυντές υλικού OVS-TC παρέχουν καλύτερη επίδοση του επεξεργαστή, λιγότερη πολυπλοκότητα, ενισχυμένη κλιμακωσιμότητα και αυξημένη απόδοση δικτύου. [11]



Σχήμα 2.5: Το επιταχυνόμενο OVS-TC Μονοπάτι (Πηγή: [11])

2.3 Η έξυπνη κάρτα δικτύου SmartNIC

Οι Netronome Agilio CX SmartNICs επιτρέπουν διαφανή μείωση φορτίου μέσω της TC διαδρομής δεδομένων. Ενώ το OVS λογισμικό δεν παύει να εκτελείται στον εξυπηρετητή, τα τμήματα αντιστοίχισης/ενέργειας στην OVS-TC διαδρομή δεδομένων είναι συγχρονισμένα στο Agilio SmartNIC μέσω "άγκιστρων" (hooks) που παρέχονται από τον πυρήνα του Linux. Χάρη στους 60 πυρήνες (480 νήματα) που διαθέτει η Agilio SmartNIC πετυχαίνουμε βιομηχανικών προδιαγραφών επιτάχυνση υλικού και υψηλό ROI-Return Of Investment καταναλώνοντας λιγότερο από 25W. Τέλος υπάρχει πλήρης υποστήριξη από την Netronome όσον αφορά την επιτάχυνση υλικού, καθώς ενεργοποιεί την επιτάχυνση νέων χαρακτηριστικών, με αναβαθμίσεις firmware, όσο η OVS κοινότητα προσθέτει ολοένα και περισσότερα χαρακτηριστικά στο OVS-TC. Στις επόμενες ενότητες θα μελετήσουμε το

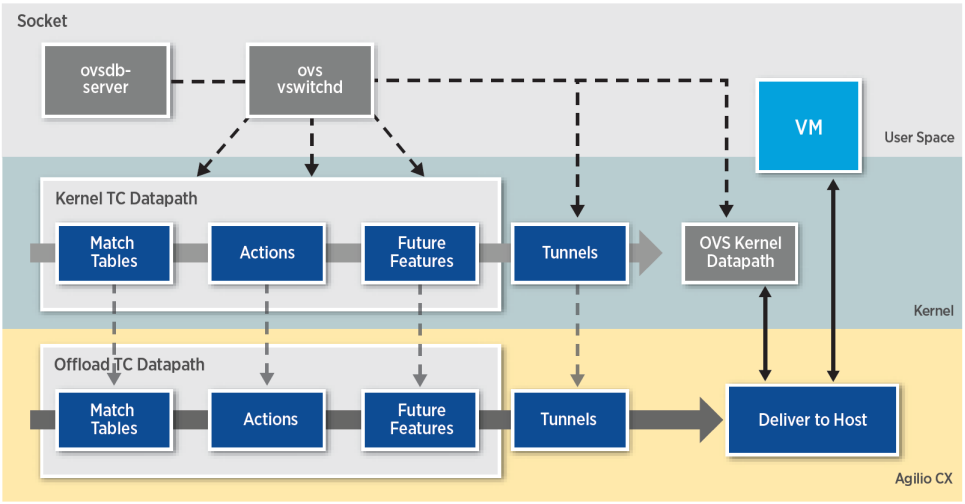
μηχανισμό λειτουργίας της μείωσης φορτίου.

2.3.1 Διαφανής OVS Αρχιτεκτονική μείωσης φορτίου

Εκτελώντας τις OVS-TC λειτουργίες δεδομένων στο Agilio SmartNIC, η διαδρομή δεδομένων TC επιταχύνεται δραματικά, ενώ ταυτόχρονα αφήνονται υψηλότερου επιπέδου εργασίες και χαρακτηριστικά στον έλεγχο του λογισμικού. Η επιτάχυνση υλικού πετυχαίνει σημαντικές βελτιώσεις στην απόδοση, διατηρώντας τα οφέλη που προέρχονται από μια αρκετά μεγάλη κοινότητα ανάπτυξης ανοιχτού κώδικα.

Όταν χρησιμοποιείται το OVS-TC, έχουμε 3 Διαδρομές δεδομένων:

1. Τη Διαδρομή δεδομένων μέσω SmartNIC
2. Τη Διαδρομή δεδομένων μέσω TC kernel
3. Τη Διαδρομή δεδομένων μέσω OVS kernel



Σχήμα 2.6: Η διαφανής OVS-TC αρχιτεκτονική μείωσης φορτίου της Agilio (Πηγή: [11])

Το OVS περιέχει έναν πράκτορα (agent) στον χώρο του χρήστη και μια διαδρομή δεδομένων στον χώρο του πυρήνα. Ο πράκτορας στον χώρο του χρήστη είναι υπεύθυνος για τη ρύθμιση του μεταγωγέα και τον πληθυσμό του πίνακα ροών. Όπως φαίνεται στο Σχήμα 2.6, είναι μοιρασμένο σε 2 θεμελιώδη τμήματα: **ovs-vswitchd** και **ovsdb-server**. Ο πράκτορας στον χώρο του χρήστη δέχεται ρυθμίσεις είτε από την τοπική γραμμή εντολών (πχ. ovs-vsctl) είτε από έναν απομακρυσμένο ελεγκτή. Στην περίπτωση του απομακρυσμένου ελεγκτή συνηθίζεται η χρήση OVSDB που αλληλεπιδρά με τον ovsdb-server. Στη διαδρομή δεδομένων μέσω TC kernel εισέρχονται τα "άγκιστρα" για τη μείωση φορτίου. Με αυτήν τη λύση, το λογισμικό OVS συνεχίζει να εκτελείται στον εξυπηρετητή, όμως τα τμήματα αντιστοίχισης/ενέργειας στην OVS-TC διαδρομή δεδομένων είναι συγχρονισμένα στο Agilio SmartNIC μέσω των "άγκιστρων" που παρέχονται από τον πυρήνα του Linux.[11]

2.3.2 Περιγραφή λογισμικού Agilio

Η δικτύωση βασισμένη σε εξυπηρετητές έχει κερδίσει την προσοχή μας τα τελευταία χρόνια, χάρη στη σημασία της λειτουργικότητας και απόδοσης στα κέντρα δεδομένων. Η βασική πρόκληση είναι η σύνδεση των εικονικών μηχανών και εφαρμογών με το δίκτυο του κέντρου δεδομένων. Λόγω της ραγδαίας εξέλιξης της εικονικοποίησης στα κέντρα δεδομένων και της αυξημένης κίνησης του δικτύου, δημιουργείται ολοένα η ανάγκη για βελτίωση της απόδοσης.

Το λογισμικό της Netronome Agilio ειδικεύεται στη μείωση φορτίου (offloading) και επιτάχυνση της δικτύωσης που βασίζεται σε εξυπηρετητές. Οι έξυπνες κάρτες δικτύων SmartNICs της Agilio στοχεύουν στην επιτάχυνση του Open vSwitch (OVS), καλύπτοντας περιπτώσεις κόμβων σε εικονικά περιβάλλοντα IaaS ή SaaS, NFV, και μη εικονικών κόμβων υπηρεσιών. Σε αυτές τις περιπτώσεις είναι συνήθης η ύπαρξη πληθώρας επιπέδων δικτύου και πολιτικών ασφαλείας που εφαρμόζονται στον εξυπηρετητή. Στην παρούσα διπλωματική εργασία θα εστιάσουμε στη μείωση φορτίου μέσω OVS (OVS offload), καθώς αποτελεί ένα ουσιαστικό πρότυπο για υλοποίηση επιπέδων και πολιτικών ασφαλείας σε επίπεδο δικτύου ανάμεσα σε πολλούς διαχειριστές κέντρων δεδομένων.

Το λογισμικό της Agilio έχει σαν βάση τον κώδικα OVS, διατηρεί τη συμβατότητα των διεπαφών και ταυτόχρονα προσφέρει παρόμοια εμπειρία χρήσης όσον αφορά την παραμετροποίηση και διαχείριση του. Αυτό σημαίνει ότι δεν υπάρχει διακριτή διαφορά στην χρήση μεταξύ ενός επιταχυνόμενου OVS και μη, παρά μόνο στην απόδοση του συστήματος.[12]

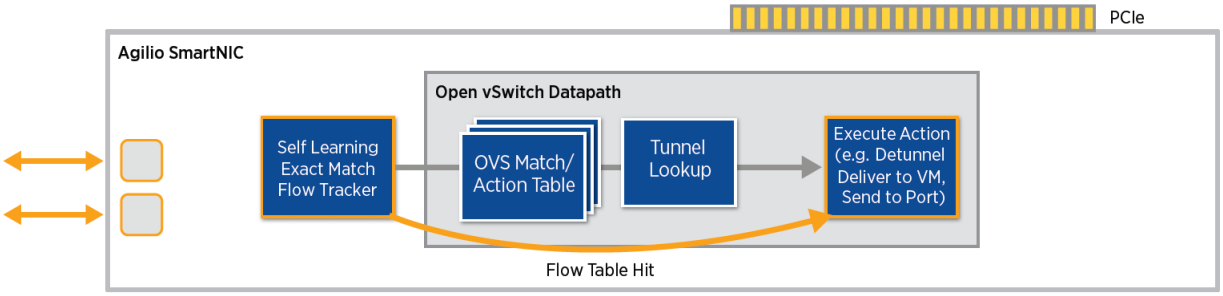
2.3.2.1 Αρχιτεκτονική του λογισμικού Agilio

Ο πυρήνας του λογισμικού της Agilio, που παρέχει όλα τα πλεονεκτήματα σε επίπεδο συστήματος, είναι η επιταχυνόμενη διαδρομή δεδομένων που εκτελείται στον προσαρμογέα Agilio. Αν μειωθεί το φορτίο μιας ροής (flow) στον Agilio SmartNIC, τότε περιέχεται πλήρως και η αναμενόμενη OVS αντιστοίχιση και η ενέργεια επεξεργασίας. Αυτή η μείωση φορτίου επιταχύνει την πολύπλοκη επεξεργασία πακέτων, εξοικονομώντας κύκλους μηχανής οι οποίοι μπορούν να αξιοποιηθούν για την ανάθεση περισσότερων εικονικών μηχανών στον εξυπηρετητή. Η επιταχυνόμενη διαδρομή δεδομένων αποτελείται από διάφορα στοιχεία (βλ. Σχήμα 2.7), εκ των οποίων τα πιο σημαντικά είναι ο ιχνηλάτης ακριβούς αντιστοίχισης ροής, οι επαναλαμβανόμενοι πίνακες κατακερματισμού, και οι οδηγοί PCIe. Αναλυτικότερα:

- **Ο ιχνηλάτης ακριβούς αντιστοίχισης ροής** αποτελεί το μεγαλύτερο μέρος της επιτάχυνσης στο σύστημα. Αυτός ο πίνακας είναι μια κρυφή μνήμη από καταχωρήσεις ροών την οποία επεξεργάζεται το σύστημα. Θα αναλυθεί περαιτέρω αργότερα.
- **Ο επαναλαμβανόμενος πίνακας κατακερματισμού** στο SmartNIC δεν είναι τίποτε άλλο παρά ένα συγχρονισμένο αντίγραφο του πίνακα στην OVS διαδρομή δεδομένων του πυρήνα. Η OVS διαδρομή δεδομένων του πυρήνα αλληλεπιδρά με το

λογισμικό του Agilio για να αφαιρεί για παράδειγμα καταχωρήσεις ροών, να συγκεντρώνει στατιστικά ή ακόμα και να αποκτά δεδομένα χρονοσφραγίδας (timestamps).

- **Οι οδηγοί PCIe** είναι υπεύθυνοι για τη μετάδοση και λήψη των πακέτων από και προς τον οικοδεσπότη (host) και τις εικονικές μηχανές. Επίσης επιτρέπουν τη χρήση SR-IOV εικονικών διαδικασιών (VFs) σε συνδυασμό με τη διαδρομή δεδομένων OVS, κάτι το οποίο δεν θα συνέβαινε χωρίς το SmartNIC. Κάθε VF στο SmartNIC αναπαριστά μία εικονική πόρτα στο vSwitch. Για παράδειγμα, όταν ένα πακέτο εισέρχεται στην φυσική πόρτα δικτύου και επεξεργάζεται από το SmartNIC, μια ενέργεια του πακέτου μπορεί να είναι να μεταδοθεί σε μια εικονική πόρτα η οποία αντιστοιχίζεται σε ένα VF. Αυτό επιτρέπει το VM να δέχεται κίνηση μέσω VF. Το ίδιο ισχύει και για την αντίστροφη κίνηση, όταν δηλαδή το VM θέλει να στείλει ένα πακέτο, αυτό εξέρχεται από το VF. Όταν το πακέτο εισέρχεται στο VF, δηλαδή στην εικονική πόρτα του επιταχυνόμενου vSwitch, τότε εκτελείται στο πακέτο η αναζήτηση της ροής και οι ενέργειες (πχ. Entunnel σε VXLAN).



Σχήμα 2.7: Η επιταχυνόμενη διαδρομή δεδομένων του Agilio SmartNIC (Πηγή: [12])

Η επιταχυνόμενη διαδρομή δεδομένων λαμβάνει τις ρυθμίσεις από την OVS διαδρομή δεδομένων του πυρήνα. Μετά την OVS διαδρομή δεδομένων του πυρήνα λαμβάνει με τη σειρά του τις ρυθμίσεις από τον OVS Πράκτορα στον χώρο του χρήστη. Αυτά τα στοιχεία μένουν αναλλοίωτα στο λογισμικό της Agilio, αποτελώντας το κλειδί για τη συμβατότητα προς τα πίσω, την πλήρη υποστήριξη των χαρακτηριστικών OVS και την ευκολία χρήσης. [12]

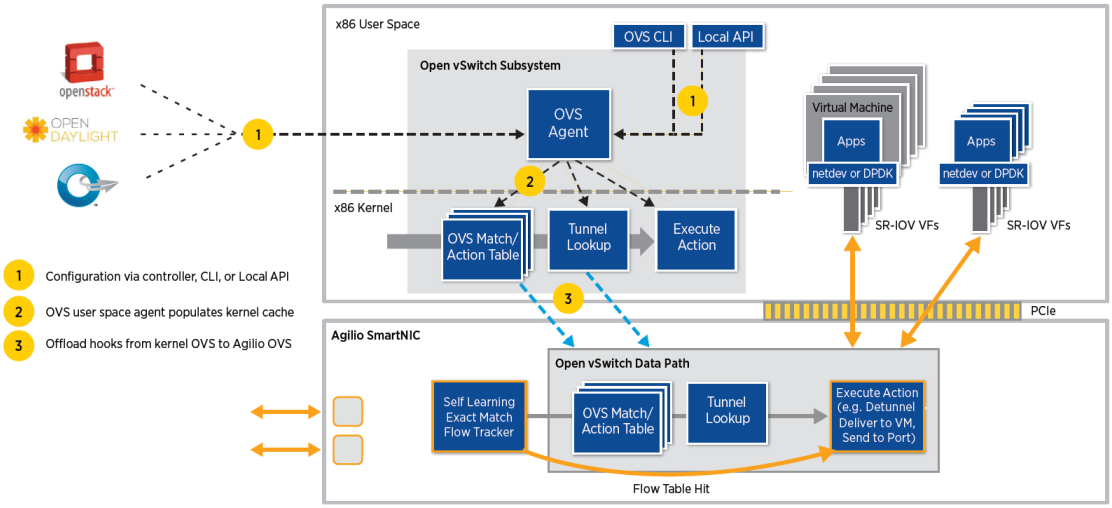
2.3.2.2 Διεπαφές ρύθμισης και "άγκιστρα" για διαφανή μείωση φορτίου

Προτού προχωρήσουμε στη διαδικασία της μείωσης φορτίου, είναι σημαντικό να μελετήσουμε πρώτα την εσωτερική δομή του OVS σε υψηλό επίπεδο.

Το OVS περιέχει έναν πράκτορα (agent) στον χώρο του χρήστη και μια διαδρομή δεδομένων στον χώρο του πυρήνα. Ο πράκτορας στον χώρο του χρήστη είναι υπεύθυνος για τη ρύθμιση του μεταγωγέα και τον πληθυσμό του πίνακα ροών. Η πραγματική προώθηση και επεξεργασία των ροών στο επίπεδο προώθησης δεδομένων πραγματοποιείται από

επαναλαμβανόμενους πίνακες κατακερματισμού στη διαδρομή δεδομένων μέσω του πυρήνα. Η διαδρομή δεδομένων μέσω του πυρήνα λειτουργεί ως μια τοπική κρυφή μνήμη για το vSwitch. Όσο ο πράκτορας στον χώρο του χρήστη επεξεργάζεται τις ροές, αυτές αποθηκεύονται προσωρινά στη διαδρομή δεδομένων του πυρήνα έτσι ώστε τα επακόλουθα πακέτα της ροής να μπορούν να πάρουν το "γρήγορο μονοπάτι".

Η διαδρομή δεδομένων μέσω OVS του πυρήνα είναι εκείνη στην οποία εισέρχονται τα "άγκιστρα" μείωσης φορτίου της Agilio. Οι διαδρομές δεδομένων μέσω του πυρήνα ή του προσαρμογέα δεν θα επεξεργαστούν καινούργιες εισερχόμενες ροές επειδή οι καταχωρήσεις ροών δεν έχουν γίνει ακόμα γνωστές. Ως αποτέλεσμα, ο πράκτορας στον χώρο του χρήστη επεξεργάζεται νέες ροές και ενημερώνει διαρκώς τον επαναλαμβανόμενο πίνακα κατακερματισμού στην OVS διαδρομή δεδομένων του πυρήνα. Τα άγκιστρα μείωσης φορτίου στον πυρήνα μαθαίνουν για την ενημέρωση και διασφαλίζουν ότι η ίδια καταχώρηση ροής εφαρμόζεται στους επαναλαμβανόμενους πίνακες κατακερματισμού στον προσαρμογέα του εξυπηρετητή, στέλνοντας ένα μήνυμα ελέγχου στην επιταχυνόμενη διαδρομή δεδομένων μέσω PCIe. Σε αυτό το σημείο, το επόμενο πακέτο της ροής θα δεχθεί επεξεργασία στο SmartNIC.



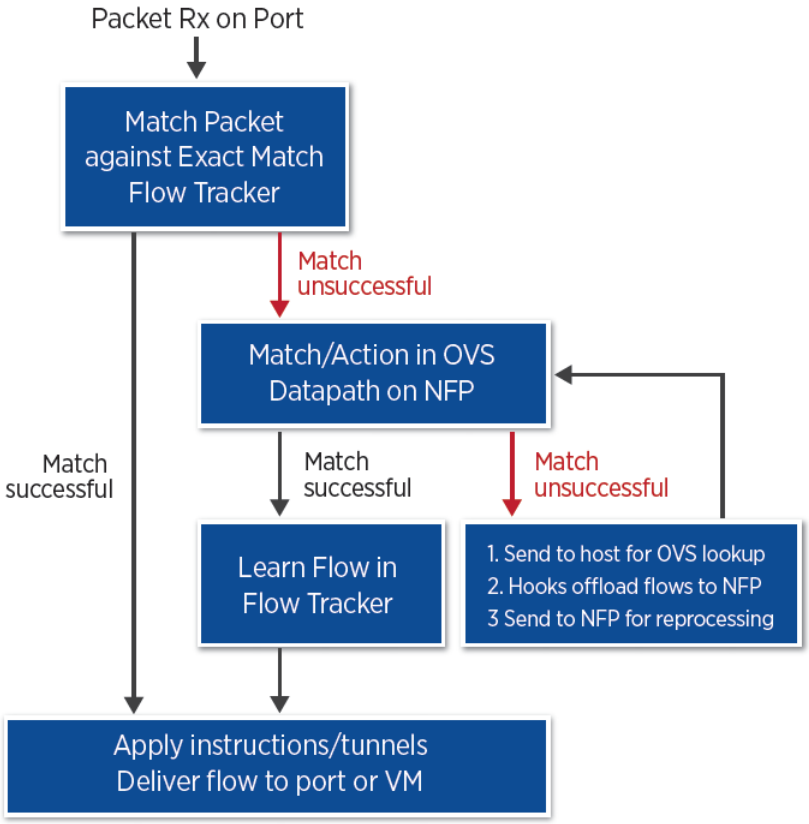
Σχήμα 2.8: Η διαδικασία ρύθμισης του OVS και των διαδρομών δεδομένων της Agilio (Πηγή: [12])

2.3.2.3 Επιτάχυνση μέσω ιχνηλάτη ακριβούς αντιστοίχισης ροής

Το λογισμικό Agilio στο SmartNIC περιλαμβάνει έναν ιχνηλάτη ακριβούς αντιστοίχισης ροής ο οποίος ανιχνεύει κάθε ροή (ή μικροροή) που διαπερνά το σύστημα. Η εισερχόμενη πόρτα καθώς και όλα τα πεδία επικεφαλίδων του πακέτου που υποστηρίζονται από το σύστημα αναγνωρίζουν τη ροή. Ο ιχνηλάτης ακριβούς αντιστοίχισης ροής βελτιώνει την απόδοση σε σύγκριση με τον επαναλαμβανόμενο πίνακα κατακερματισμού επειδή χρειάζεται μόνο μια αναζήτηση ροής, σε αντίθεση με τις πολλαπλές αναζητήσεις, καθεμία με διαφορετική μάσκα, που εκτελεί ο πίνακας κατακερματισμού. Συνοπτικά, θα μπορούσε να πει κανείς ότι ο ιχνηλάτης

ακριβούς αντιστοίχισης ροής μπορεί να θεωρηθεί ως μία L1 κρυφή μνήμη, ο επαναλαμβανόμενος πίνακας κατακερματισμού στο Agilio SmartNIC ως μια L2 κρυφή μνήμη και η OVS διαδρομή δεδομένων του πυρήνα ως μια L3 κρυφή μνήμη.

Το παρακάτω διάγραμμα (βλ. Σχήμα 2.9) απεικονίζει την ιεραρχία εκμάθησης ροών στο Agilio όταν εισέρχονται πακέτα στο SmartNIC:



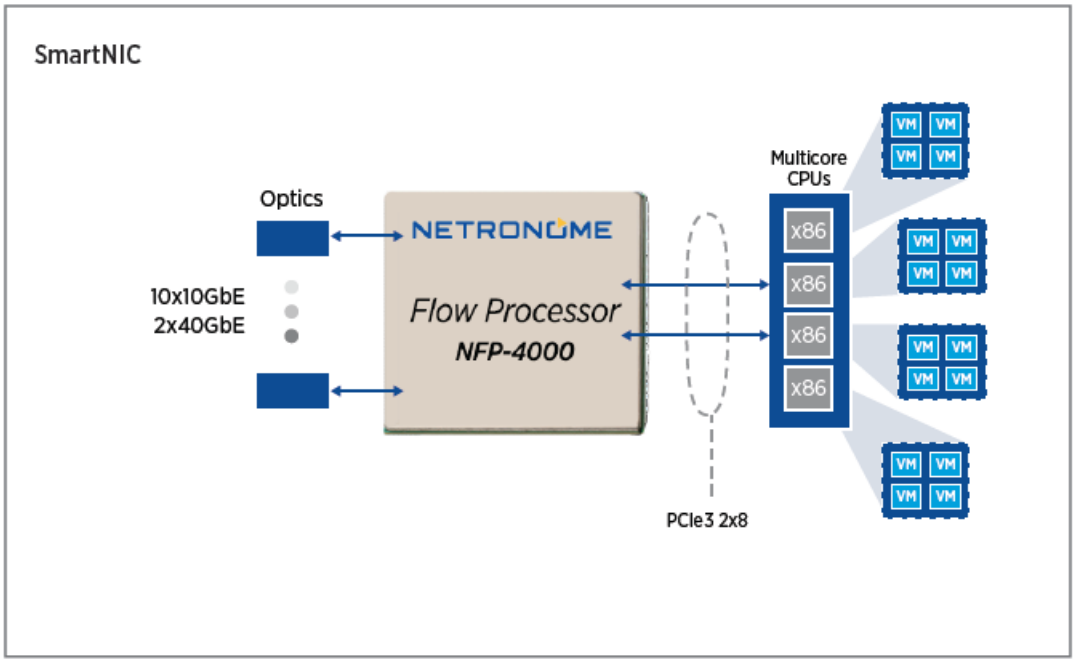
Σχήμα 2.9: Διάγραμμα Ροής απεικονίζει πώς προγραμματίζονται οι ροές στη διαδρομή δεδομένων (Πηγή: [12])

Όταν ένα πακέτο εισέρχεται στο σύστημα εκτελείται μια αναζήτηση στον ιχνηλάτη ακριβούς αντιστοίχισης ροής. Αν μια καταχώρηση δεν βρεθεί, τότε προστίθεται. Για νέες ροές, ή μέχρι να γνωστοποιηθεί η πολιτική που σχετίζεται με τη ροή, χρησιμοποιείται ο επαναλαμβανόμενος πίνακας κατακερματισμού στον Agilio SmartNIC. Αν μια καταχώρηση βρεθεί από την επαναληψιμότητα του κατακερματισμού, τότε ενημερώνεται η πολιτική που σχετίζεται με την εκάστοτε καταχώριση ροής και όλα τα εναπομείναντα πακέτα της ροής θα δεχθούν επεξεργασία χρησιμοποιώντας την ακριβούς αντιστοίχισης καταχώρηση ροής. Αν η επαναληψιμότητα του κατακερματισμού δεν φέρει επιτυχή αντιστοίχιση, τότε το πακέτο στέλνεται στο x86 για τον OVS πυρήνα και πιθανώς για επεξεργασία στον χώρο του χρήστη.

2.3.3 Ο Επεξεργαστής Ροών της Netronome (Netronome Flow Processor) NFP-4000

Ο πολυνηματικός και πολυπύρηνος επεξεργαστής ροών NFP-4000 6ης γενιάς στοχεύει στην έξυπνη επεξεργασία στο επίπεδο των δεδομένων σε υψηλές αποδόσεις με χαμηλή κατανάλωση ισχύος και κόστους, ενώ ταυτόχρονα διαθέτει όλα τα μέσα για να προσαρμοστεί στα καινούργια χαρακτηριστικά δικτύων. Η βέλτιστη αρχιτεκτονική επεξεργασίας ροών με τη μέθοδο αντιστοίχισης/ενέργειας επιτρέπει σε κάθε χρήστη, κάθε εικονικής μηχανής και κάθε εφαρμογής την υποστήριξη για μικρό-καταμερισμό (βλ. Σχήμα 2.10).

Ο μικρό-καταμερισμός (Micro-segmentation) είναι μια τεχνική ασφαλείας δικτύου, η οποία επιτρέπει στους αρχιτέκτονες του δικτύου να μοιράζουν λογικά το κέντρο δεδομένων σε ξεχωριστά μέρη-τμήματα ασφαλείας στο ατομικό επίπεδο εργασίας και ύστερα να καθορίζουν ελέγχους ασφαλείας και να παραδίδουν υπηρεσίες για κάθε τμήμα ξεχωριστά. Με αυτόν τον τρόπο εξασφαλίζουν μεγαλύτερη ευελιξία αφού δεν χρειάζεται να δημιουργούν πολλαπλά φυσικά τείχη προστασίας και αυξημένη προστασία σε κάθε VM ξεχωριστά.[14]



Σχήμα 2.10: Απεικόνιση NFP στην SmartNIC (Πηγή: [13])

Τέλος ο NFP-4000 παρέχει ασφαλή πρόσβαση σε δεδομένα, σε επίπεδο cloud, μέσω ενός ενσωματωμένου, μαζικής κρυπτογράφησης, επιταχυντή υψηλής απόδοσης. Παρακάτω παρατίθενται τα χαρακτηριστικά του επεξεργαστή ροής:

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ:

- 60 Προγραμματιζόμενοι πυρήνες για επεξεργασία ροών.
- 48 Πυρήνες για επεξεργασία πακέτων.
- PCIe Gen3 2x8 Διεπαφή (2PF/64VF η κάθε μια).

- SR-IOV με 256 ουρές υλικού.
- Προσαρμοστική διαχείριση προσωρινής αποθήκευσης.
- Ρυθμιζόμενο 2x32-bit ή 1x64-bit ECC protected DDR3-1866.
- Περισσότερη από 19MB μνήμη on-chip.
- 10GBase-KR Serializer/Deserializer που υποστηρίζει
  - Ethernet 10Gb/40Gb/100Gb.
  - Interlaken -έως 100Gb.
- Μαζική κρυπτογραφία στο υλικό για τις περισσότερες κρυπτογραφημένες σουίτες.
- 60+ ειδικοί επιταχυντές για Εκτεταμένη Ανάλυση Πακέτου (Deep Packet Inspection-DPI).
- Arm11 πυρήνας (L1 (64K), L2 (256K)).
- Διαχείριση κίνησης.

Στο Σχ 2.11 απεικονίζεται το Μπλοκ Διάγραμμα του NFP-4000 Επεξεργαστή Ροής.

2.4 Η εξέλιξη του δικτύου τηλεπικοινωνιών

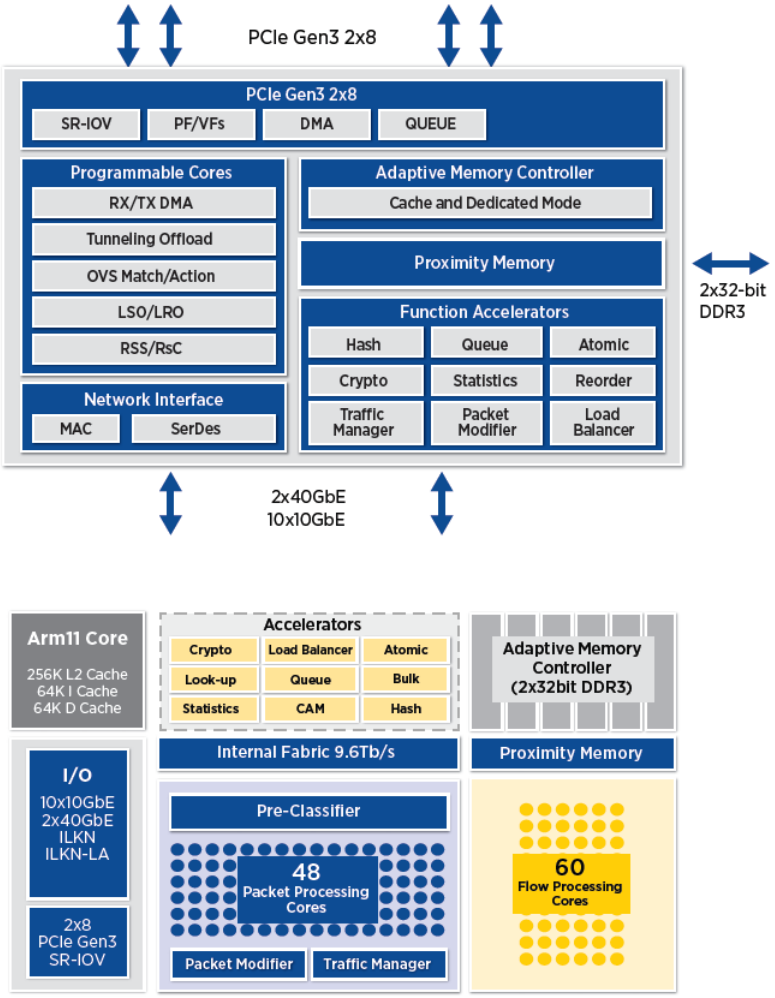
2.4.1 Εισαγωγή στο IMS

Το IMS (IP multimedia subsystem) ή στα ελληνικά (IP Υποσύστημα Πολυμέσων) σχεδιάστηκε από τον οργανισμό ασύρματων προτύπων 3ης γενιάς (3GPP) ως μέρος του σχεδίου για την εξέλιξη των δικτύων κινητής πέρα από το GSM, με σκοπό να ορίσει μια νέα αρχιτεκτονική στα πλαίσια των δικτύων κινητής για την παροχή τηλεπικοινωνιακών υπηρεσιών βασισμένες στο πρωτόκολλο IP. Κεντρικός στόχος είναι η σύγκλιση των δικτύων σταθερής και κινητής με την υποστήριξη εφαρμογών και υπηρεσιών, υψηλών απαιτήσεων, εξ ολοκλήρου στο επίπεδο IP.

Οι υπηρεσίες του IMS σχεδιάστηκαν να καλύπτουν τις ανάγκες για υπηρεσίες πολυμέσων πραγματικού χρόνου, όπως οι κλήσεις βίντεο και το live streaming. Αυτές οι υπηρεσίες εκμεταλλεύονται τη χρήση ενός οριζόντιου επίπεδου ελέγχου το οποίο διαχωρίζει το επίπεδο μεταφοράς από το επίπεδο υπηρεσιών, με αποτέλεσμα να μην έχουν ανάγκη για δικές τους λειτουργίες ελέγχου.

Στη σχεδίαση του IMS, αποτέλεσε σημαντικό κομμάτι το θέμα της διασφάλισης της επικοινωνίας μεταξύ των παρόχων και των τελικών χρηστών, προσφέροντας στους παρόχους τη δυνατότητα εφαρμογής λειτουργιών ελέγχου για τη σωστή λειτουργία και την αποφυγή παράδοσης υπηρεσιών προς κακόβουλους τελικούς χρήστες.

Έτσι χάρη στο IMS και την ενοποίηση διαφορετικών μέσων αναπτύχθηκαν νέες υπηρεσίες, συγκριτικά με τις ήδη υπάρχουσες στην αγορά, βελτιώνοντας σημαντικά την



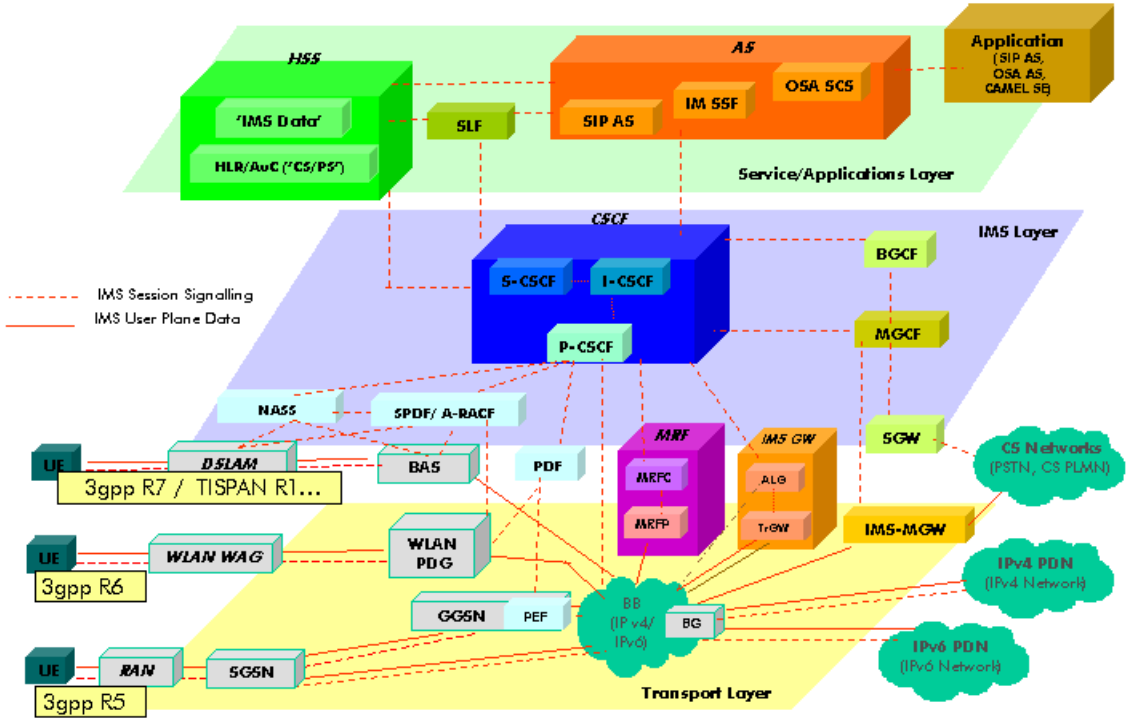
Σχήμα 2.11: Μπλοκ Διάγραμμα του NFP-4000 Επεξεργαστή Ροής (Πηγή: [13])

τελική εμπειρία των χρηστών. Παραδείγματα τέτοιων υπηρεσιών είναι το VoLTE, VoWiFi (Wifi Calling), το RCs κα. Συγκεκριμένα το VoLTE είναι μια πιο απλοποιημένη εκδοχή του IMS, η οποία ορίστηκε το 2010 εξαιτίας της πολύπλοκης και ακριβής δομής του IMS γενικότερα. [15]

2.4.2 Αρχιτεκτονική του IMS

Το IMS δεν θα πρέπει να αναγνωρίζεται ως μια ακόμα τεχνολογία, αλλά ως μια ισχυρή αρχιτεκτονική συνόδων και υπηρεσιών που δημιουργεί την πλατφόρμα για νέας γενιάς υπηρεσίες. Ουσιαστικά πρόκειται για μια συλλογή διαφορετικών διεπαφών που επιτελούν διάφορες λειτουργίες, όπως την ενεργοποίηση συνόδων πολυμέσων μεταξύ δύο ή περισσότερων τελικών συσκευών (βλ. Σχήμα 2.12).

Η Αρχιτεκτονική IMS βασίζεται στο πρωτόκολλο σηματοδότησης SIP το οποίο έχει σχεδιαστεί για τη δημιουργία, διαχείριση και τερματισμό συνόδων (sessions) σε ένα δίκτυο IP.



Σχήμα 2.12: Αρχιτεκτονική στοιχείων IMS (Πηγή: [15])

2.4.2.1 Δίκτυο πρόσβασης IMS

Ο χρήστης μπορεί να συνδεθεί στο IMS με διάφορους τρόπους, οι περισσότεροι εκ των οποίων χρησιμοποιούν το πρότυπο IP. Τα τερματικά που χρησιμοποιούν IP και είναι SIP User Agents μπορούν να εγγραφούν απευθείας στο IMS, ακόμα και σε περιπτώσεις περιαγωγής από άλλο δίκτυο ή χώρα. Τέτοιες συσκευές μπορεί να είναι είτε σταθερές (DSL, Cable Modems, Ethernet κα.) είτε κινητές όπως τηλέφωνα, PDA μέσω W-CDMA, GSM, GPRS, WLAN κα. Στην περίπτωση παλιών αναλογικών συσκευών (POTS, H.323 και μη συμβατών, με το IMS, συστημάτων) υποστηρίζεται η χρήση gateway για την αξιοποίηση τους.

2.4.2.2 Δίκτυο πυρήνα IMS

Το κεντρικό δίκτυο IMS αποτελείται από μια ομάδα (μοναδικών ή επαναλαμβανόμενων) στοιχείων που εκτελούν συγκεκριμένες λειτουργίες. Ορισμένες από τις κύριες οντότητες του κεντρικού δικτύου (βλ. Σχήμα 2.13) περιγράφονται συνοπτικά παρακάτω:

- **CSCF (Call Session Control Function)** – Είναι το πιο σημαντικό στοιχείο της διάταξης, καθώς αναλαμβάνει τη δρομολόγηση της σηματοδοσίας SIP, όπως των πακέτων εγγραφής χρήστη, έναρξης μιας συνόδου και ταυτοποίησης χρήστη. Το CSCF μπορεί να χωριστεί σε τρείς κατηγορίες:

  - **Proxy CSCF (P-CSCF)**: Είναι ο διαμεσολαβητής μεταξύ άλλων στοιχείων και αποτελεί την αφετηρία για τους τελικούς χρήστες. Είναι υπεύθυνος για τη διασφάλιση της επικοινωνίας μεταξύ του δικτύου και του τελικού χρήστη, επιθεωρώντας τα μηνύματα ανταλλαγής. Ο κύριος ρόλος του είναι η ταυτοποίηση των χρηστών πραγματοποιώντας μια IPsec σύνδεση ασφαλείας με το IMS τερματικό εμποδίζοντας την εισβολή κακόβουλων χρηστών.

  - **Interrogating CSCF (I-CSCF)**: «Ανακρίνει» τον HSS για το επόμενο βήμα στην επικοινωνία, το οποίο είναι προς κάποιον εξυπηρετητή εφαρμογών AS ή τον S-CSCF και στη συνέχεια δρομολογεί την κίνηση των πακέτων σηματοδοσίας αντίστοιχα.

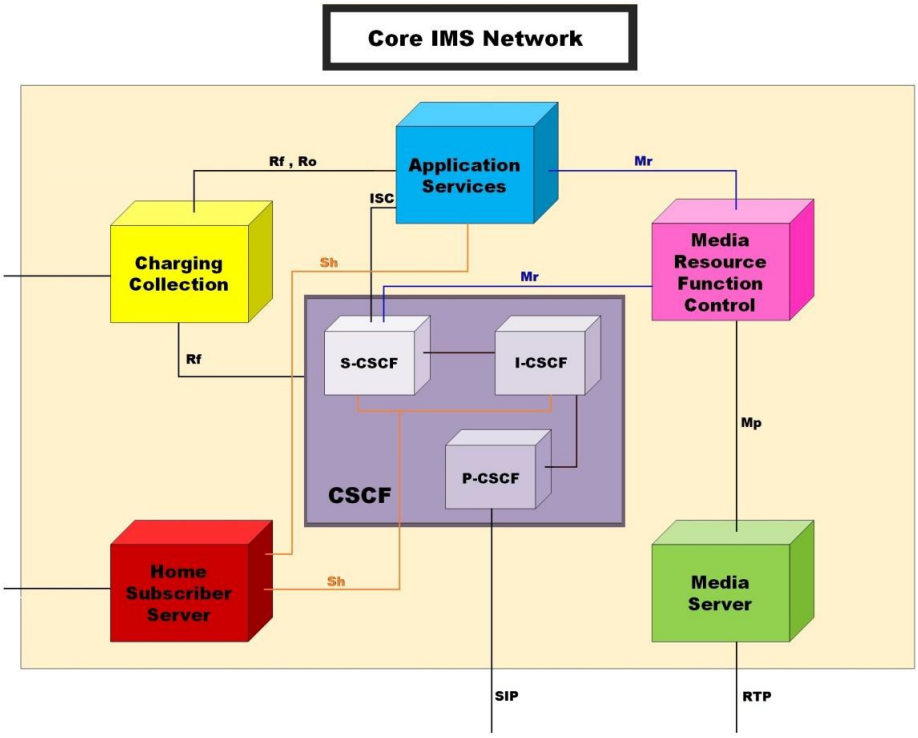
  - **Serving CSCF (S-CSCF)**: Είναι στο κέντρο της διάταξης και εκτελεί υπηρεσίες ελέγχου των συνόδων, ως SIP Registrar. Έχει τη δυνατότητα να φορτώσει διάφορα προφίλ χρήστη από τον HSS και να τα εφαρμόσει ανάλογα με τις απαιτήσεις του κάθε χρήστη, για να επιλέξει τον κατάλληλο εξυπηρετητή εφαρμογών.

- **Home Subscriber Server (HSS)**: Κρατάει αποθηκευμένα όλα τα προφίλ των χρηστών και το ιστορικό υπηρεσιών, καθώς επίσης και την τοποθεσία τους, όπως γίνεται και στο GSM HLR.

- **Application Server (AS)**: Εξυπηρετητής SIP, υπεύθυνος για την φιλοξενία και εκτέλεση των υπηρεσιών (πχ. Αναγνώριση κλήσης, αναμονή κλήσης, προώθηση κλήσεων, υπηρεσία Voicemail κα.). Μπορεί να βρίσκεται στο "Home" δίκτυο ή ακόμα και εκτός του IMS δικτύου όπου μπορεί να λειτουργεί και σαν μεμονωμένος εξυπηρετητής.

- **Breakout Gateway Control Function (BGCF)**: Επιλέγει το κατάλληλο δίκτυο για τη δρομολόγηση από PSTN αιτήματα, με βάση τον αριθμό τηλεφώνου.

- **Media Resource Function (MRF)**: Το στοιχείο που είναι υπεύθυνο για τον χειρισμό και την ανάλυση μέσων (media) όπως για παράδειγμα τη μετατροπή μεταξύ διαφορετικών codecs, την αναπαραγωγή φωνής κα.



Σχήμα 2.13: Δίκτυο πυρήνα IMS (Πηγή: [15])

2.4.3 Πρωτόκολλο σηματοδοσίας SIP

Σε μια διάταξη IMS χρησιμοποιούνται διάφορα πρωτόκολλα επικοινωνίας όπως το IPv6, SIP, SDP, Diameter, RTP, MEGACO και RSVP. Στην παρούσα διπλωματική θα μελετήσουμε τη σηματοδοσία SIP, την οποία και θα εξηγήσουμε συνοπτικά παρακάτω.

Το πρωτόκολλο SIP δημιουργήθηκε με σκοπό την επικοινωνία μεταξύ κάποιου χρήστη UA και του εξυπηρετητή. Αποτελεί το μέσο για τη δημιουργία, τροποποίηση ή τερματισμό μιας αμφίδρομης επικοινωνίας που περιλαμβάνει εφαρμογές πολυμέσων όπως φωνή, εικόνα, instant messaging κα.

Κεφάλαιο 3

Αρχιτεκτονική Δικτύου

3.1 Γενική Περιγραφή

Σε αυτήν την παράγραφο θα αναλυθούν οι απαιτήσεις υλικού και λογισμικού για την επιτυχή διεξαγωγή του πειράματος. Σχετικά με τα υλικά χρειαστήκαμε δύο Η/Υ όπου χρησιμοποιήθηκαν για την εγκατάσταση του Open Day Light ελεγκτή και την επιτάχυνση και αντιγραφή πακέτων με τη χρήση της Agilio SmartNIC, αντίστοιχα. Επίσης χρησιμοποιήθηκε και ενα Raspberry Pi 3 Model B ως πελάτης SIP και γεννήτρια κίνησης δικτύου. Όσον αφορά το απαραίτητο λογισμικό, εγκαταστήσαμε διάφορα αναγκαία προγράμματα όπως το (ODL, KVM, Kamailio, OVS, Scapy κά.).

3.2 Απαιτήσεις Υλικού

Παρακάτω θα αναφερθούν τα απαραίτητα εργαλεία υλικού που χρειάστηκαν για την πραγματοποίηση του πειραματικού μέρους. Όπως έχει ήδη αναφερθεί, όλος ο εξοπλισμός ανήκει στο Δίκτυο Υπολογιστών του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών και το πείραμα διεξήχθη στο εργαστήριο.

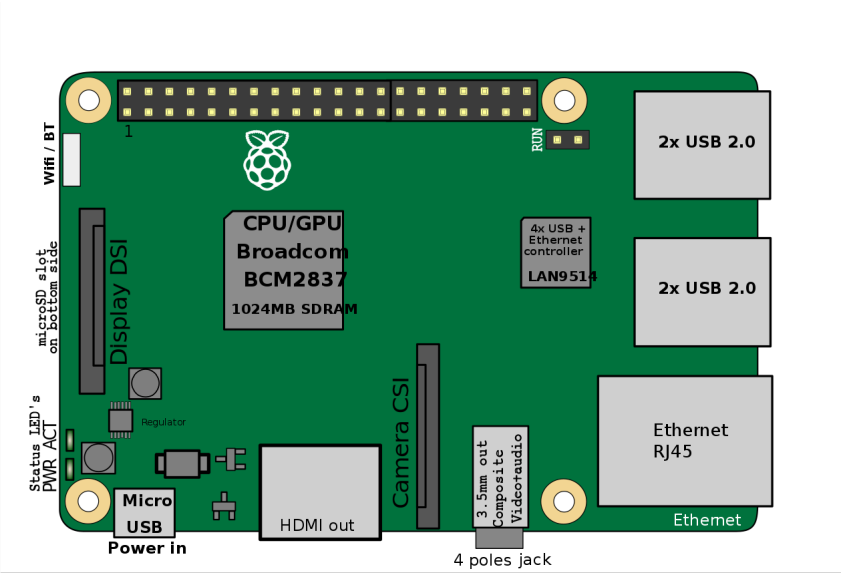
3.2.1 Raspberry Pi 3 Model B

Σχήμα 3.1: Raspberry Pi 3 Model B (Πηγή: [16])

Το Raspberry Pi είναι ένας μικρός υπολογιστής ο οποίος χρησιμοποιείται ευρέως σε πληθώρα εφαρμογών και επιστημονικών κλάδων, όπως στη ρομποτική, μετεωρολογία, εκπαίδευση κά. χάρη στο χαμηλό του κόστος και την υψηλή του φορητότητα. Στην παρούσα διπλωματική εργασία θα το χρησιμοποιήσουμε ως πελάτη SIP για να δημιουργήσουμε μια σύνδεση SIP Client-Server και να ελέγξουμε την κίνηση. Πρώτα όμως αξίζει να αναφερθούμε στα χαρακτηριστικά ενός Rpi.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ:

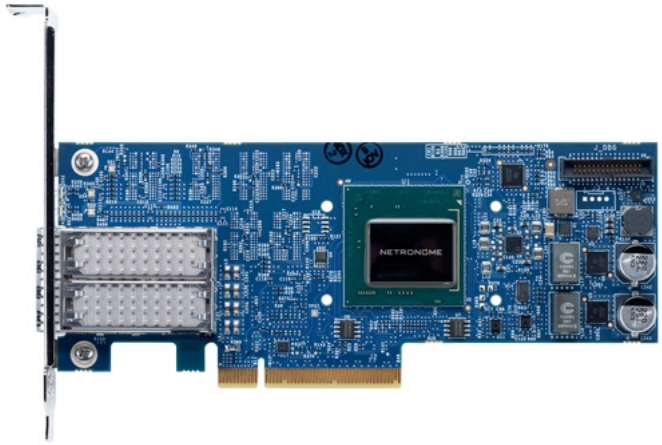
- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A



Σχήμα 3.2: Raspberry Pi 3 Model B: Διεπαφές (Πηγή: [17])

3.2.2 Agilio® CX 2x10GbE SmartNIC

Η έξυπνη κάρτα δικτύου SmartNIC είναι σχεδιασμένη για δικτύωση Cloud, SDN και NFV υψηλών αποδόσεων.



Σχήμα 3.3: Agilio® CX 2x10GbE SmartNIC (Πηγή: [18])

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

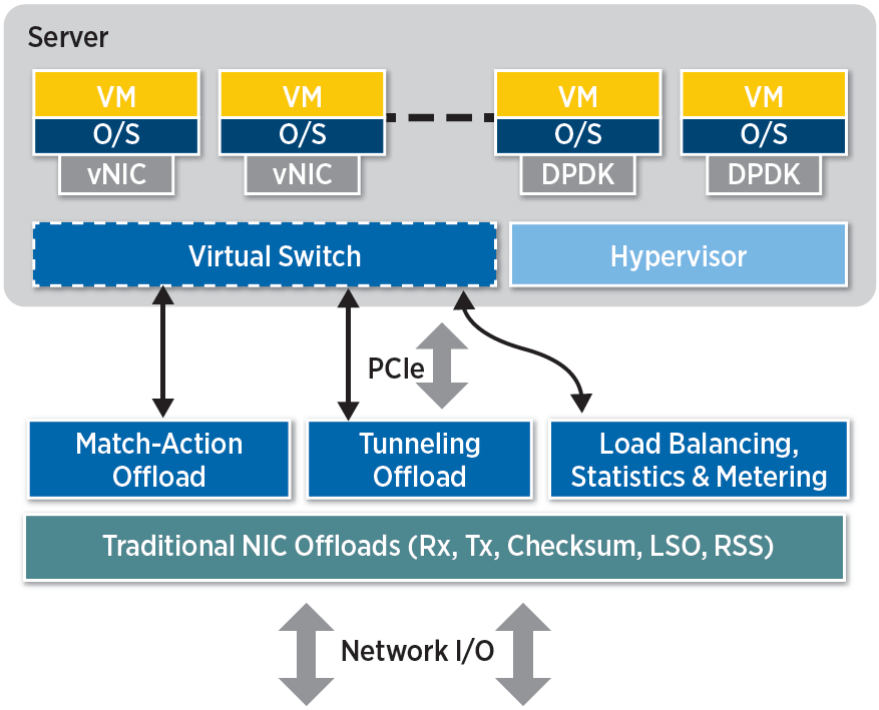
- Εκφόρτωση στη vSwitch διαδρομή δεδομένων σε δημοφιλή λειτουργικά συστήματα
  - VXLAN, NVGRE, MPLS τούνελ ενθυλάκωση και απενθυλάκωση
  - Προγραμματίσιμο για ειδικούς τύπους τούνελ
  - Ευέλικτες και κλιμακώσιμες πολιτικές αντιστοίχισης-ενέργειας
  - Κρυφή μνήμη στο υλικό για τρομερή επιτάχυνση
  - Εξισορρόπηση φορτίου
  - Στατιστική ανάλυση ροής και QoS
- Επιτάχυνση δικτύου και εκφόρτωση
  - TCP/UDP/IP εκφόρτωση
  - RSS (Receive-side scaling)
  - SR-IOV
  - Πολλαπλές ουρές ανά εικονική μηχανή
  - DPDK, zero-copy, παράκαμψη πυρήνα
- Επιτάχυνση επίπονων υπολογιστικά εργασιών
  - Έλεγχος πακέτου εις βάθος (DPI)
  - Ατομικές εκτελέσεις

  - Στατιστικά ανά ροή σε πραγματικό χρόνο

- Ικανότητα επεξεργασίας έως 2M ροών
- Υποστήριξη έως 500K τούνελ
- Προγραμματίσιμη σε P4 και C

ΠΡΟΔΙΑΓΡΑΦΕΣ	
Διεπαφές	2-port 10GbE, SFP+
Μνήμη	2GB DDR3 onboard memory
Λειτουργικά Συστήματα	Red Hat Enterprise Linux (RHEL), CentOS, Ubuntu
Hypervisors	Linux KVM
Ethernet	PCIe Base 3.0-compliant1.1- and 2.0-compatible 2.5, 5.0 or 8.0GT/s link rate x8 MSI-X vector per RX/TX queue pair Interrupt coalescing PCIe mapped LAN, UART
Συνδεσιμότητα	IEEE Std 802.3ae 10 Gigabit Ethernet IEEE Std 802.3ba 40 Gigabit Ethernet IEEE Std 802.3ad Link aggregation and failover IEEE Std 802.1Q.1p VLAN tags and priority IEEE P802.1Qaz D0.2 ETS Jumbo frame support (9.6KB) Configuration and diagnostic tools
ΠΕΡΙΒΑΛΛΟΝΤΙΚΕΣ ΑΠΑΙΤΗΣΕΙΣ	
Θερμοκρασία λειτουργίας	0-55°C
Θερμοκρασία αποθήκευσης	-40-70°C
Σχετική υγρασία	5% to 95% non-condensing
Ροή αέρα	250 LFM (min)

Πίνακας 3.1: Agilio® CX 2x10GbE SmartNIC Προδιαγραφές & Απαιτήσεις



Σχήμα 3.4: Agilio® CX 2x10GbE SmartNIC Μπλοκ Διάγραμμα (Πηγή: [18])

3.3 Εργαλεία Λογισμικού

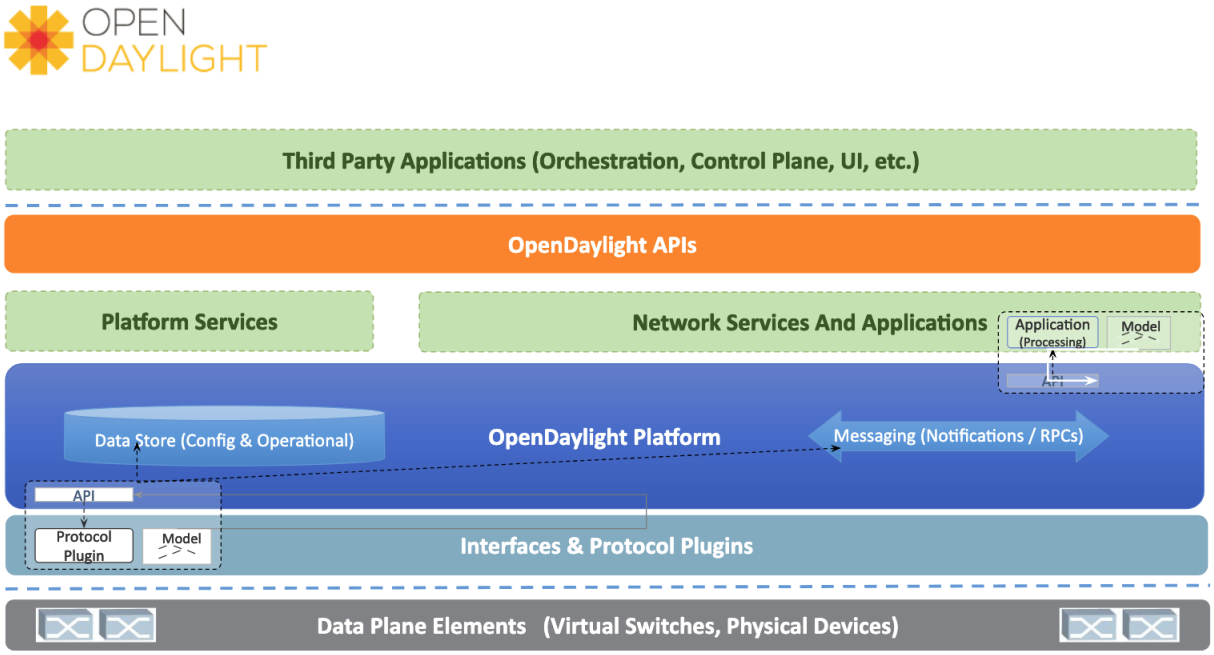
Παρακάτω παρουσιάζεται το απαραίτητο λογισμικό Software που χρησιμοποιήθηκε κατά την εκπόνηση της εργασίας.

3.3.1 OpenDayLight Controller - ODL

Το OpenDayLight (ODL) είναι ένας SDN Controller, δηλαδή μια ανοιχτού κώδικα πλατφόρμα για επεξεργασία και αυτοματοποίηση δικτύων οποιασδήποτε κλίμακας. Το OpenDayLight Project ξεκίνησε με αφορμή την ανάπτυξη του SDN με σκοπό τον προγραμματισμό δικτύου. Είναι γραμμένο σε γλώσσα Java.

Το ODL αποτελείται από **3 επίπεδα** (βλ. Σχήμα 3.5):

- **Η Βόρεια διεπαφή (Northbound)** όπως πχ. REST/NETCONF η οποία επιτρέπει την εφαρμογή υψηλού επιπέδου πολιτικών στις δικτυακές συσκευές ή την ενσωμάτωση του ODL με άλλες πλατφόρμες.
- **Ο πυρήνας** ο οποίος χρησιμοποιείται από το Service Abstraction Layer (SAL) το οποίο βασίζεται στο OSGi που κατευθύνει την κίνηση που διαπερνά τον Controller όσο είναι ενεργός.
- **Η Νότια διεπαφή** η οποία επικοινωνεί με τις δικτυακές συσκευές (πχ. OpenFlow).



Σχήμα 3.5: OpenDayLight Αρχιτεκτονική (Πηγή: [19])

Πλεονεκτήματα:

- **Ευελιξία/Επεκτασιμότητα**: Χάριν στους περιέκτες OSGi μπορούν να φορτωθούν πολλά κομμάτια κώδικα ζωντανά κατά την εκτέλεση, κάνοντας το δίκτυο πιο εύχρηστο, ευέλικτο και εύκολα επεκτάσιμο.
- **Κλιμακωσιμότητα** : Το ODL χρησιμοποιεί μια βασιζόμενη σε μοντέλα προσέγγιση, που σημαίνει ότι χρειάζεται μια συνολική εικόνα του δικτύου για να εκτελέσει λογικές πράξεις. Υποστηρίζει μεθόδους προσέγγισης γεωγραφικής θέσης με πολλαπλές τοποθεσίες και ανοχή στα σφάλματα. Ακόμα χρησιμοποιεί BGP Routing για να κατευθύνει τις ροές κίνησης μεταξύ των "νησιών" SDN.[19]

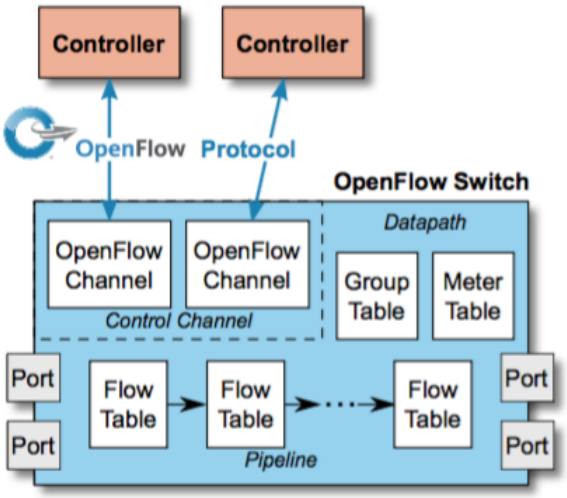
3.3.2 OpenFlow

Το Openflow (OF) αποτελεί ένα από τα πρώτα πρότυπα δικτύων ορισμένων από λογισμικό (SDN). Αρχικά καθόρισε το πρωτόκολλο επικοινωνίας σε περιβάλλοντα SDN που επέτρεπε στον ελεγκτή SDN να αλληλεπιδρά απευθείας με το επίπεδο δεδομένων των δικτυακών συσκευών, όπως δρομολογητών και μεταγωγέων, είτε φυσικών είτε εικονικών.

Ο ελεγκτής SDN είναι ο «εγκέφαλος» του SDN δικτύου που μεταφέρει πληροφορίες στις δικτυακές συσκευές μέσω APIs του Νότιου επιπέδου και στις εφαρμογές μέσω APIs του Βόρειου επιπέδου. Με την ολοένα αυξανόμενη χρήση των SDN δικτύων στις επιχειρήσεις, οι ελεγκτές SDN έχουν επιφορτιστεί με τη χρήση κοινών διεπαφών εφαρμογών, όπως το OpenFlow και τη βάση δεδομένων του εικονικού μεταγωγέα (OVSDB).

Για τη διαχείριση ενός OF περιβάλλοντος, είναι απαραίτητο κάθε συσκευή που θέλει να επικοινωνήσει με τον ελεγκτή SDN να υποστηρίζει το πρωτόκολλο OpenFlow. Μέσω αυτής της διεπαφής, ο ελεγκτής SDN προωθεί προς τα κάτω αλλαγές στους πίνακες ροών των δικτυακών συσκευών επιτρέποντας έτσι στους διαχειριστές δικτύων να διαχειρίζονται τη δικτυακή κίνηση, να ελέγχουν ροές για βέλτιστη απόδοση, και να εκτελούν δοκιμές σε νέες ρυθμίσεις και εφαρμογές. [20]

Ένας μεταγωγέας OpenFlow επικοινωνεί με έναν εξωτερικό ελεγκτή μέσω ενός καναλιού OpenFlow. Εκτελεί αναζήτηση και προώθηση πακέτων βάσει του πίνακα (ή των πινάκων) ροών. Στο σχήμα 3.6 απεικονίζεται το εσωτερικό ενός OpenFlow μεταγωγέα και η επικοινωνία του με τον ελεγκτή.



Σχήμα 3.6: Η ανατομή ενός Openflow μεταγωγέα (Πηγή: [21])

3.3.3 ClearWater IMS

Το Clearwater είναι ένα IMS στο Cloud. Το IMS (IP Multimedia Subsystem) είναι μια αρχιτεκτονική βασισμένη σε πρότυπα την οποία έχουν υιοθετήσει πολλές μεγάλες εταιρίες τηλεπικοινωνιών ως τη βάση για φωνή, βίντεο και μηνύματα μέσω IP, αντικαθιστώντας το παλιό σύστημα κυκλώματος με μεταγωγή και τα προηγούμενης γενιάς συστήματα VoIP που βασίζονταν σε μεταγωγή μέσω λογισμικού (soft switching). Το Clearwater ακολουθεί τις αρχιτεκτονικές αρχές του IMS και υποστηρίζει όλες τις διεπαφές κλειδιά που εμφανίζονται στον πυρήνα του IMS δικτύου. Σε αντίθεση με τις παραδοσιακές υλοποιήσεις IMS, το Clearwater σχεδιάστηκε εξαρχής για το Cloud, προσφέροντας κλιμακωσιμότητα και εξαιρετική αποδοτικότητα κόστους. Τέλος το Project Clearwater είναι χορηγία της Metaswitch Networks.

Το Clearwater προσφέρει έλεγχο σε κλήσεις που βασίζονται στο πρωτόκολλο SIP για φωνή και εικόνα καθώς επίσης και για υπηρεσίες μηνυμάτων. Μπορεί να χρησιμοποιηθεί είτε αυτόνομα σαν λύση σε υπηρεσίες VoIP στην αγορά, βασιζόμενο στο ενσωματωμένο σετ χαρακτηριστικών για κλήσεις και τη βάση δεδομένων για τους συνδρομητές, είτε ως ο πυρήνας του IMS σε συνδυασμό με άλλα στοιχεία, όπως Εξυπηρετητές Εφαρμογής Τηλεφωνίας και Εξυπηρετητές Οικιακού Συνδρομητή. Στη δεύτερη περίπτωση, το Clearwater εκτελεί οποιαδήποτε εργασία θα έκανε ένας πυρήνας IMS, συμπεριλαμβάνοντας τον διαμεσολαβητή CSCF (Call Session Control Function), τον I-CSCF (Interrogating CSCF) και τον S-CSCF (Serving CSCF) μαζί με το BGCF (Breakout Gateway Control Function). Επίσης το Clearwater περιλαμβάνει το WebRTC Gateway, υποστηρίζοντας τη διεργασία μεταξύ WebRTC πελατών και SIP πελατών, χρησιμοποιώντας σηματοδοσία SIP πάνω σε WebSocket.

Με τον καιρό, πολλές εταιρείες τηλεπικοινωνιών αναμένεται να τρέχουν όλες τις δικτυακές διαδικασίες στο επίπεδο ζεύξης δεδομένων (Επίπεδο 2) και πάνω σε περιβάλλον NFV, συμπεριλαμβάνοντας το IMS. Συνεπώς το Clearwater, αφού σχεδιάστηκε εξ 'αρχής για να λειτουργεί σε εικονικό περιβάλλον, είναι ιδανικό για NFV χρησιμοποιώντας στο μέγιστο την ευελιξία του Cloud. Το NFV (Network Functions Virtualization) είναι αναμφίβολα ένα φλέγον θέμα στον χώρο των τηλεπικοινωνιών αυτήν τη στιγμή. Είναι μια προσέγγιση στην κατασκευή τηλεπικοινωνιακών δικτύων τα οποία ξεφεύγουν από ιδιόκτητα κουτιά, όπου είναι δυνατό, χρησιμοποιώντας λογισμικό που εκτελείται σε βιομηχανικών προτύπων εικονικές υποδομές IT.

3.3.3.1 Αρχιτεκτονική του ClearWater IMS

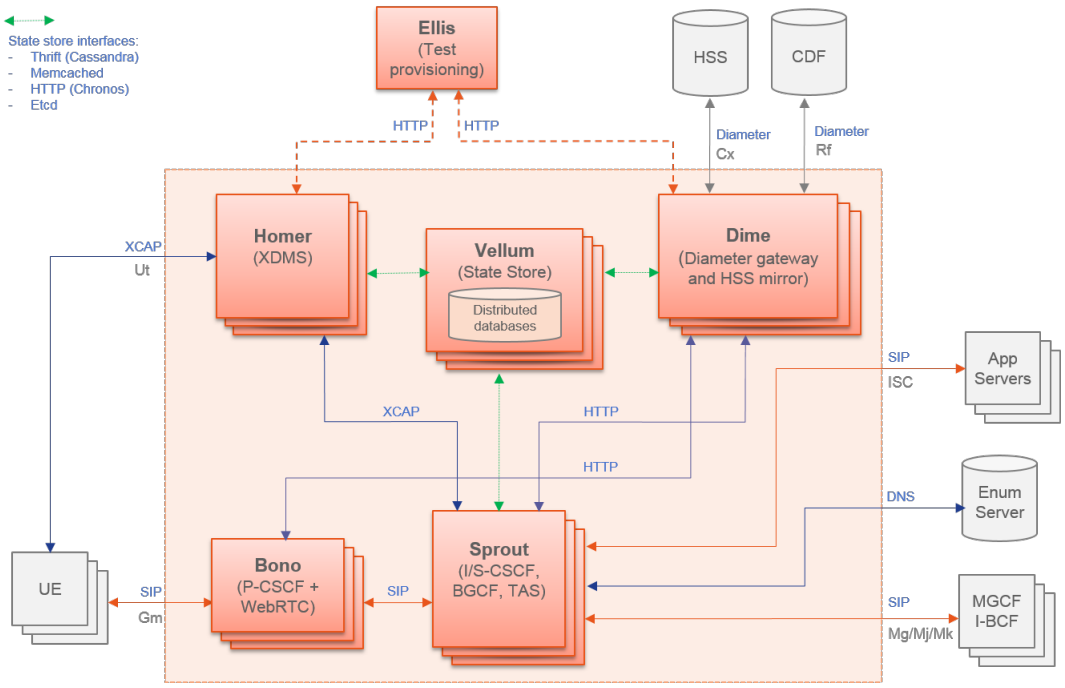
Όπως προαναφέραμε, το Clearwater φέρει αρκετές ομοιότητες με την παραδοσιακή αρχιτεκτονική του IMS, ωστόσο δεν είναι ίδια. Πιο συγκεκριμένα:

- Όλα τα στοιχεία είναι οριζοντίως κλιμακώσιμα χρησιμοποιώντας απλή εξισορρόπηση φορτίου χωρίς κατάσταση (stateless).
- Όλες οι μακροχρόνιες καταστάσεις αποθηκεύονται σε έναν εξειδικευμένο κόμβο τον

Vellum, ο οποίος χρησιμοποιεί τεχνολογίες βελτιστοποίησης στο Cloud όπως την Cassandra. Καμία μακροχρόνια κατάσταση δεν αποθηκεύεται σε κανέναν από τους κόμβους παραγωγής, καθιστώντας έτσι εύκολη και άμεση την δυναμική κλιμάκωση της συστάδας, ελαχιστοποιώντας τη ζημιά που θα επέφερε ένας χαμένος κόμβος.

- Οι διεπαφές μεταξύ των front-end SIP στοιχείων και των back-end υπηρεσιών χρησιμοποιούν RESTful υπηρεσίες ιστού.
- Οι διεπαφές μεταξύ διαφόρων στοιχείων χρησιμοποιούν ομαδοποίηση σύνδεσης με στατιστική ανακύκλωση των συνδέσεων για να διασφαλίσουν ότι το φορτίο μεταδόθηκε εξίσου, όσο κόμβοι προστίθενται και αφαιρούνται σε κάθε επίπεδο.

Το Σχήμα 3.7 παρακάτω απεικονίζει την αρχιτεκτονική του Clearwater με τα στοιχεία του:



Σχήμα 3.7: ClearWater IMS Αρχιτεκτονική (Πηγή: [22])

Το Clearwater αποτελείται από τα παρακάτω IMS στοιχεία:

- I-CSCF
- S-CSCF
- BGCF
- P-CSCF
- Έναν βασικός HSS (Home Subscriber Server)
- Έναν εξυπηρετητή εφαρμογών IR.92

- Έναν XDMS για να υποστηρίζει τον εξυπηρετητή εφαρμογών
- Μια αυτό-εξυπηρετούμενη πύλη για provisioning
- WebRTC - IMS SIP gateway

Ένα από τα στοιχεία που θα εξετάσουμε στην παρούσα διπλωματική εργασία είναι το Bono, καθώς θα χρησιμοποιηθεί, όπως θα δούμε στο πειραματικό στάδιο.

3.3.3.2 Bono (Διαμεσολαβητής ακμής)

Οι κόμβοι Bono σχηματίζουν έναν οριζόντια κλιμακώσιμο SIP διαμεσολαβητή ακμής παρέχοντας ταυτόχρονα και μια SIP IMS Gm συμβατή διεπαφή και μια διεπαφή WebRTC για τους πελάτες. Οι συνδέσεις μεταξύ των πελατών είναι εξισορροπημένες κατά μήκος των κόμβων. Ο κόμβος Bono παρέχει ένα σημείο αγκύρωσης για τη σύνδεση του πελάτη με το σύστημα Clearwater, συμπεριλαμβανομένης της υποστήριξης των μηχανισμών για διάσχιση NAT. Ο πελάτης είναι, για αυτόν τον λόγο, αγκυρωμένος σε έναν συγκεκριμένο κόμβο Bono, όσο διαρκεί η εγγραφή, ωστόσο μπορεί να μετακινήσει σε άλλον κόμβο Bono σε περίπτωση αποτυχίας της εγγραφής. Οι πελάτες συνδέονται στον Bono μέσω SIP/UDP ή SIP/TCP. Ο Bono υποστηρίζει οποιοδήποτε πελάτη WebRTC ο οποίος εκτελεί εγκαθίδρυση κλήσης με σηματοδοσία SIP πάνω από WebSocket.

Τα στοιχεία επεξεργασίας του Clearwater, που ονομάζονται Bono και Sprout, είναι διαμεσολαβητές που διατηρούν τις καταστάσεις των συναλλαγών (transaction-stateful) όπως αυτοί ορίζονται στο RFC3261. Ως διαμεσολαβητές, περνάνε όλες τις επικεφαλίδες SIP με πλήρη διαφάνεια. Αυτό σημαίνει ότι εγγενώς υποστηρίζουν μια πληθώρα προτύπων που σχετίζονται με το πρωτόκολλο SIP που καθορίζουν επιπλέον συμπεριφορές SIP, καθώς αυτές οι συμπεριφορές είναι αντιληπτές στους UA (User Agent) πελάτες και UA εξυπηρετητές.

Ο Bono υλοποιεί τον IMS διαμεσολαβητή CSCF (Proxy CSCF). Οι Session Border Controllers (SBC) συνήθως υλοποιούν και αυτοί τον διαμεσολαβητή CSCF, χωρίς να σημαίνει απαραίτητα ότι ο Bono είναι SBC. Οι SBC εκτελούν ένα ευρύ φάσμα λειτουργιών πέραν των λειτουργιών του διαμεσολαβητή CSCF, όπως να προστατεύει το IMS δίκτυο από διάφορες κακόβουλες επιθέσεις, διαχειρίζοντας το περιεχόμενο των SIP μηνυμάτων για λόγους διαλειτουργικότητας και αναμετάδοσης μέσων που υποστηρίζουν τη διάσχιση μέσω NAT. Ο Bono δεν εκτελεί καμία από αυτές τις λειτουργίες. Γι' αυτόν τον σκοπό θα χρησιμοποιήσουμε έναν διαμεσολαβητή SIP τον Kamailio, όπως θα δούμε και παρακάτω.[22]

3.3.4 Kamailio

Το Kamailio, πρώην OpenSER, είναι ένας SIP εξυπηρετητής υπό την άδεια της GNU General Public License, ικανός να χειριστεί χιλιάδες κλήσεις το δευτερόλεπτο και ιδανικός για real-time υπηρεσίες τηλεπικοινωνίας. Μπορεί να χρησιμοποιηθεί είτε σε μικρό γραφείο είτε σαν αναπληρωματικό PBX τηλεφωνικό κέντρο σε επιχειρήσεις, ακόμα και σε παρόχους υπηρεσιών διαδικτύου. Η εγκατάσταση του είναι δυνατή στις περισσότερες εκδόσεις λειτουργικού συστήματος βασισμένου σε Linux όπως πχ. Ubuntu, RaspberryPi, FreeBSD κά. Μπορεί να παραμετροποιηθεί ως SIP Registrar (Διαχειριστής Εγγραφών), SIP Proxy (διαμεσολαβητής ή εξυπηρετητής ανακατεύθυνσης) και περιέχει χαρακτηριστικά όπως αναγνώριση απουσίας χρήστη, υπηρεσία απομακρυσμένης ταυτοποίησης χρήστη RADIUS (Remote Authentication Dial-in User Service), syslog για καταγραφή δραστηριότητας, απομακρυσμένο έλεγχο βασισμένο σε JSON-RPC ή XML-RPC, εργαλεία για SQL και NoSQL backends, δυνατότητα επέκτασης σε IMS/VoLTE κά.

Συγκεκριμένα μπορεί να χρησιμοποιηθεί ως:

- SIP τηλεφωνικό σύστημα
- SIP εξισορροπιστής φορτίου
- SIP τείχος προστασίας
- Οικονομική μηχανή δρομολόγησης
- IMS/VoLTE πλατφόρμα
- Υπηρεσία μηνυμάτων και αναγνώρισης απουσίας χρήστη
- SIP IPv4-IPv6 πύλη
- SIP-WebRTC πύλη

“ Το Kamailio είναι Χαβανέζικη λέξη και σημαίνει ομιλία.”

Το Kamailio χαρακτηρίζεται κυρίως από την κλιμακωσιμότητα του, αφού μπορούν να συνυπάρχουν περισσότεροι από ένας εξυπηρετητής Kamailio σε ένα δίκτυο. Μπορεί να τρέχει σε ενσωματωμένα συστήματα με περιορισμένους πόρους και η απόδοση μπορεί να φτάσει και εκατοντάδες κλήσεις το δευτερόλεπτο. Αν πάλι χρησιμοποιηθεί ως εξισορροπιστής φορτίου, τότε μπορεί να διαχειρίζεται πάνω από 5000 κλήσεις το δευτερόλεπτο. Επίσης σε ένα σύστημα με 4GB RAM, το Kamailio μπορεί να εξυπηρετήσει έως και 300.000 εν ενεργεία συνδρομητές. Τέλος από θέμα δρομολόγησης είναι δυνατή η διαχείριση εκατομμυρίων κανόνων δρομολόγησης με ξεκάθαρους κανόνες ανακατεύθυνσης και εφεδρείας. [23][24]

Πιο αναλυτικά, τα **χαρακτηριστικά** του Kamailio είναι τα εξής:

- **Στιβαρός και αποδοτικός SIP εξυπηρετητής**

  - Εξυπηρετητής εγγραφής (Registrar Server).
  - Εξυπηρετητής τοποθεσίας (Location Server).
  - Εξυπηρετητής διαμεσολάβησης (Proxy Server).
  - Εξυπηρετητής εφαρμογών σηματοδοσίας SIP (SIP Application Server).
  - Εξυπηρετητής ανακατεύθυνσης (Redirect Server).

- **Ευελιξία**
  - Μικρό αποτύπωμα, ιδανικό για ενσωματωμένες συσκευές. Το δυαδικό αρχείο που αποτελεί τον πυρήνα είναι αρκετά μικρού μεγέθους και η λειτουργικότητα εξαρτάται εξ ολοκλήρου από τα τμήματα εφαρμογών (modules).
  - Plug & Play τμήματα εφαρμογών. Η ικανότητα προσθαφαίρεσης επεκτάσεων, αφήνοντας τον πυρήνα άθικτο, κάτι το οποίο προσφέρει επίσης μεγάλη σταθερότητα.
  - Modular Architecture (Αρθρωτή αρχιτεκτονική). Ο πυρήνας, οι εσωτερικές βιβλιοθήκες και οι modular-αρθρωτές διεπαφές επεκτείνουν την λειτουργικότητα του εξυπηρετητή.
  - Εντυπωσιακό πλήθος επεκτάσεων. Διατίθενται περισσότερα από 150 έτοιμα τμήματα εφαρμογών στο διαδίκτυο.

- **Ικανότητα δρομολόγησης SIP**
  - Stateless ή Statefull Επεξεργασία μηνυμάτων SIP.
  - Σειριακή ή παράλληλη διεργασία.
  - Υποστήριξη διάσχισης NAT για τη σηματοδοσία SIP και την RTP κίνηση.
  - Εξισορρόπηση φορτίου με αλγορίθμους διανομής και υποστήριξη ανακατεύθυνσης.
  - Ανακατεύθυνση δρομολόγησης.

- **Επίπεδα μεταφοράς**
  - Υποστήριξη επικοινωνίας μέσω UDP, TCP, TLS και SCTP.
  - IPv4 και IPv6.
  - Δημιουργία πύλης στο επίπεδο μεταφοράς (IPv4 σε IPv4, UDP σε TLS κοκ.).
  - WebSocket για WebRTC.
  - SCTP πολλαπλής ροής.

- **Ασύγχρονη επεξεργασία**
  - Ασύγχρονη διαχείριση TCP.
  - Ασύγχρονη επεξεργασία SIP μηνυμάτων.
  - Κατανεμημένη ουρά μηνυμάτων.

- **Ασφαλής Επικοινωνία**
  - Έλεγχος ταυτότητας χρήστη SIP.
  - Εξουσιοδότηση μέσω ACL ή συμμετοχής σε ομάδα.
  - Υποστήριξη TLS για τη σηματοδοσία SIP.
  - Διαφανής διαχείριση των SRTP πακέτων για κρυπτογραφημένη μετάδοση ήχου.
- **IMS**
  - I-CSCF, P-CSCF, S-CSCF.

3.3.5 QEMU (Quick EMUlator)

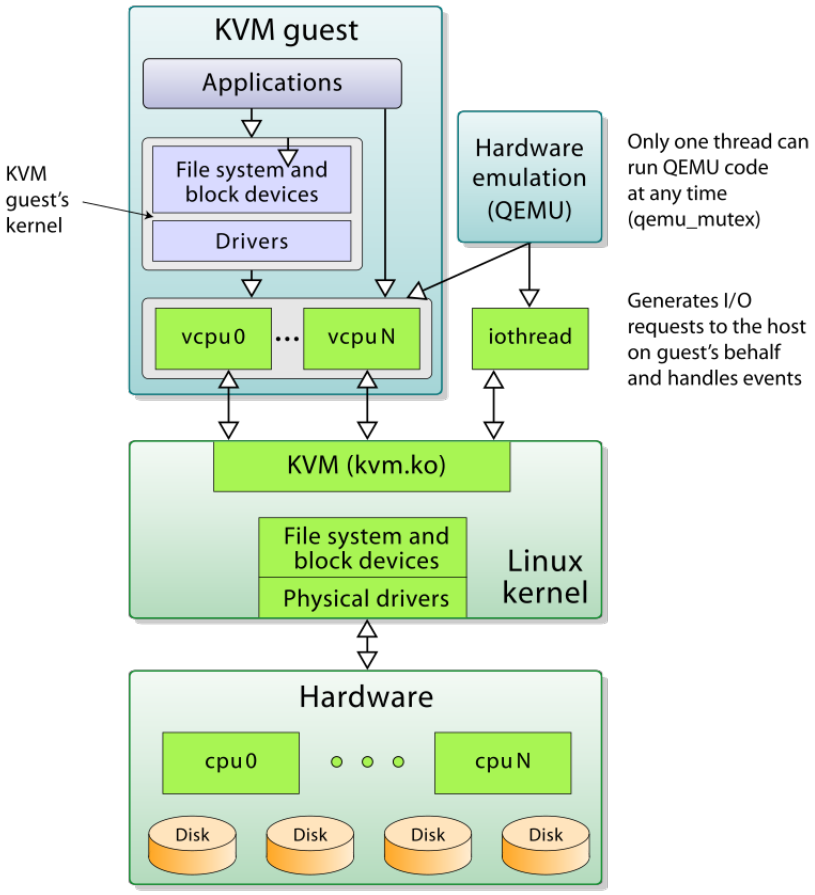
Ο QEMU είναι ένας εξομοιωτής ανοιχτού κώδικα ο οποίος εκτελεί εικονικοποίηση στο υλικό. Εξομοιώνει τον επεξεργαστή του μηχανήματος μέσω δυναμικής δυαδικής μετάφρασης και παρέχει ένα σύνολο από διαφορετικά μοντέλα υλικού και συσκευών για το μηχάνημα, δίνοντας τη δυνατότητα εκτέλεσης διάφορων φιλοξενούμενων λειτουργικών συστημάτων. Επίσης μπορεί να εξομοιώσει και διεργασίες στο επίπεδο του χρήστη, επιτρέποντας έτσι κάποιες εφαρμογές που είχαν μεταγλωττιστεί για μια αρχιτεκτονική να εκτελούνται και σε άλλες. Στην περίπτωση μας θα το χρησιμοποιήσουμε με το KVM και θα αναλάβει το στήσιμο και το migration των εικόνων KVM. Θα εξακολουθεί να συμμετέχει στην εξομοίωση του υλικού, αλλά η εκτέλεση του επισκέπτη θα γίνεται από το KVM όπως ζητήθηκε από τον QEMU. [26]

3.3.6 KVM (Kernel-based Virtual Machine)

Το KVM (Kernel-based Virtual Machine), εικονική μηχανή βασισμένη στον πυρήνα, αποτελεί μια λύση πλήρους εικονικοποίησης για Linux με x86 υλικό που περιέχει επεκτάσεις εικονικοποίησης (Intel VT ή AMD-V). Αποτελείται από μια λειτουργική μονάδα πυρήνα, την kvm.ko, η οποία παρέχει την υποδομή για εικονικοποίηση του πυρήνα και μια ειδική μονάδα επεξεργασίας, την kvm-intel.ko ή kvm-amd.ko. Με το KVM δίνεται η δυνατότητα εκτέλεσης πολλαπλών εικονικών μηχανών Linux ή και Windows. Κάθε εικονική μηχανή έχει το προσωπικό της εικονικό υλικό, όπως κάρτα δικτύου, δίσκο, προσαρμογέα γραφικών κά. [27] Στο Σχήμα (3.8) απεικονίζεται μια συνολική εικόνα υψηλού επιπέδου της εικονικοποίησης KVM/QEMU.

3.3.7 Libvirt

Το Libvirt είναι μια συλλογή από λογισμικό που χρησιμοποιείται στη διαχείριση των εικονικών μηχανών και άλλων λειτουργιών εικονικοποίησης, όπως διαχείριση δικτυακών διεπαφών. Αυτά τα τμήματα κώδικα αποτελούν μια βιβλιοθήκη API, έναν δαίμονα (libvirtd) και ένα εργαλείο στη γραμμή εντολών (virsh). Ο κύριος σκοπός του libvirt είναι να παρέχει έναν μοναδικό τρόπο για τη διαχείριση πολλαπλών εικονικών hypervisors (πχ. KVM, Xen, VMWare ESX κά.). [28]



Σχήμα 3.8: Εικονική μηχανή στον πυρήνα με χρήση KVM & QEMU (Πηγή: [29])

3.3.8 PJSUA API-Softphone API υψηλού επιπέδου

Το PJSUA API είναι ένα API υψηλού επιπέδου για την κατασκευή εφαρμογών πολυμέσων SIP. Συνδυάζει όλες τις λειτουργικότητες σηματοδοσίας και μέσων σε ένα εύκολο στη χρήση API, παρέχοντας διαχείριση λογαριασμών, διαχείριση φίλων, παρουσία χρήστη, άμεσα μηνύματα, καθώς επίσης και χαρακτηριστικά πολυμέσων όπως τηλεδιάσκεψη, μετάδοση αρχείων σε πραγματικό χρόνο, ηχογράφηση φωνής, τοπική αναπαραγωγή κά. Οι εφαρμογές πρέπει να συνδέονται με τη βιβλιοθήκη pjsua-lib για να χρησιμοποιήσουν το API. Επιπλέον, αυτή η βιβλιοθήκη εξαρτάται από τις ακόλουθες:

- pjsip-ua
- pjsip-simple
- pjsip-core
- pjmedia
- pjmedia-codec
- pjlib-util

- pjlib

συνεπώς η εφαρμογή θα πρέπει να συνδέει και τις ανωτέρω. Η παραπάνω βιβλιοθήκη χρησιμοποιήθηκε στην παρούσα εφαρμογή και η εφαρμογή της οδήγησε στην κατασκευή του Python προγράμματος, που αφορά την εγγραφή χρήστη SIP και την κλήση σε άλλον χρήστη. [30]

3.3.9 Scapy

Το Scapy είναι ένα πρόγραμμα σε Python με το οποίο ο χρήστης μπορεί να στέλνει, να παρακολουθεί, να αναλύει και να κατασκευάζει πακέτα δικτύου. Με λίγα λόγια είναι ένα πανίσχυρο πρόγραμμα χειραγώγησης πακέτων. Έχει τη δυνατότητα να κατασκευάσει ή να αποκρυπτογραφήσει πακέτα από ένα μεγάλο εύρος πρωτοκόλλων, να τα στείλει στο ''σύρμα'', να τα ''συλλάβει", να αντιστοιχήσει αιτήματα και απαντήσεις, κά. Ενώ το scapy μπορεί να χρησιμοποιηθεί για πάρα πολλές εφαρμογές, στην παρούσα διπλωματική το αξιοποιήσαμε για να στείλουμε συγκεκριμένα πακέτα στην πόρτα 5060 SIP με συγκεκριμένο ρυθμό και να παράγουμε αυξημένη κίνηση δεδομένων για να παρατηρήσουμε πώς ανταπεξέρχεται το σύστημα μας. [31]

Κεφάλαιο 4

Υλοποίηση Συστήματος

4.1 Γενική Περιγραφή

Σε αυτό το κεφάλαιο, θα περιγράψουμε αναλυτικά τη διαδικασία που ακολουθήσαμε για την διεξαγωγή του πειραματικού μέρους. Σκοπός μας είναι να κατευθύνουμε την κίνηση SIP μέσω ενός SIP Proxy (Kamailio), όπου θα ελέγχεται, θα αντιγράφεται και τελικά θα καταλήγει στον επιθυμητό προορισμό (Bono). Η αλλαγή της κατεύθυνσης της κίνησης θα επιτευχθεί με χρήση του SDN Controller (ODL) δημιουργώντας και στέλνοντας ροές μέσω του REST API και του Openflow. Στη συνέχεια η κίνηση θα φροντίσουμε να περνάει μέσα από το μονοπάτι του SmartNIC, για να πετύχουμε την επιτάχυνση, και τέλος θα δοκιμάσουμε το σύστημα σε πραγματικές συνθήκες με τη χρήση του Raspberry Pi.

Για την υλοποίηση του πειράματος θα χρησιμοποιήσουμε τρία μηχανήματα:

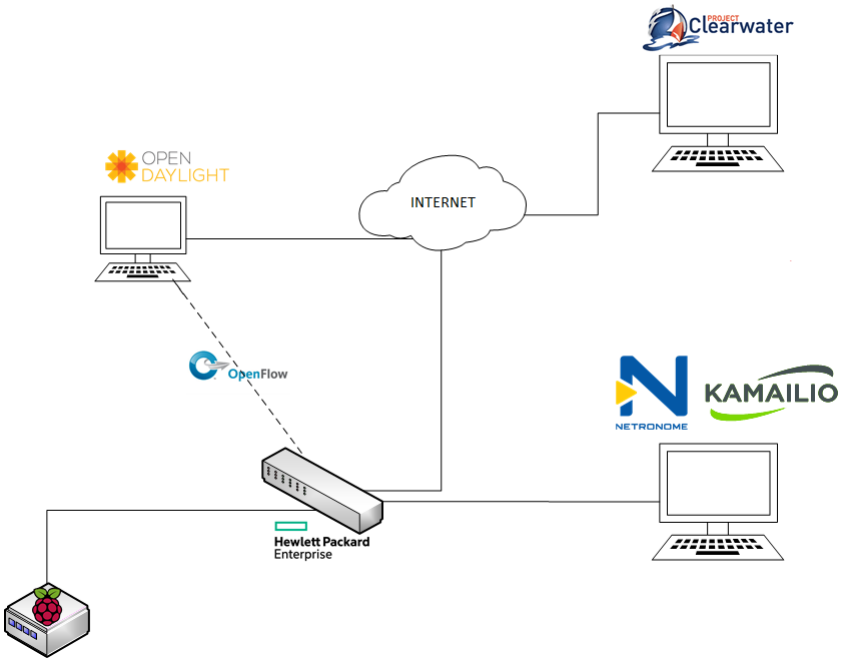
1. Η/Υ για το ODL
2. Το RaspberryPi
3. Η/Υ που διαθέτει δύο κάρτες δικτύου
   (α′) Intel 82566DM-2 Gigabit Network Connection (χρήση για ssh)
   (β′) Agilio SmartNIC 2x10GbE

Η διαχείριση των μηχανημάτων γίνεται μέσω ssh. Για την καλύτερη κατανόηση της διαδικασίας που θα ακολουθηθεί, παραθέτουμε την τοπολογία του εργαστηριακού περιβάλλοντος (βλ. Σχήμα 4.1), καθώς και τον πίνακα με τις απαραίτητες διευθύνσεις κάθε διεπαφής (βλ. Πίνακα 4.1).

4.2 Εγκατάσταση SDN δικτύου με χρήση ODL

4.2.1 ΒΗΜΑ 1: Προετοιμασία συστήματος

Συνδεόμαστε μέσω SSH στο μηχάνημα μας με IP 147.102.7.75. Αρχικά εκτελούμε την ενημέρωση apt-get για να εξασφαλίσουμε ότι ο εξυπηρετητής λαμβάνει τα πιο πρόσφατα πακέτα



Σχήμα 4.1: Τοπολογία δικτύου

	Ethernet	IP
Router-39 (gateway)	00:1c:f6:b1:56:59	147.102.39.1/24
RaspberryPi	b8:27:eb:5d:1f:e1	147.102.39.10/24
ODL	00:0c:29:ca:7b:df	147.102.7.75/24
Agilio SmartNIC	00:15:4d:13:5d:54	147.102.39.20/24
Kamailio (VM1)	52:54:00:db:a0:b1	147.102.39.21/24
Monitor (VM2)	52:54:00:24:3e:bf	147.102.39.22/24
bono	00:0c:29:e7:97:18	147.102.7.18/24

Πίνακας 4.1: Πίνακας διευθύνσεων MAC και IP

ασφαλείας και εφαρμογών:

```
$ sudo apt-get update
```

Ύστερα εγκαθιστούμε κάποια χρήσιμα πακέτα:

```
$ sudo apt-get -y install unzip vim wget
```

Τέλος θα δημιουργήσουμε έναν απομονωμένο (detached) χώρο με χρήση screen έτσι ώστε να μην παύει να εκτελείται το ODL όταν αποσυνδεόμαστε από το SSH.

```
$ sudo apt-get screen
$ screen -S odl
```

Και μετά την εγκατάσταση και παραμετροποίηση του ODL, θα πατήσουμε Ctrl + a και μετά d για να κάνουμε detach.

4.2.2 ΒΗΜΑ 2: Εγκατάσταση JAVA JRE

Στη συνέχεια θα χρειαστεί να εγκαταστήσουμε το Java JRE:

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

Πρέπει να βεβαιωθούμε ότι το Ubuntu δείχνει στο JAVA 8, εκτελώντας την παρακάτω εντολή:

```
$ sudo update-alternatives --config java
```

Με τις παρακάτω εντολές ενημερώνουμε το αρχείο BASHRC και κάνουμε source:

```
$ echo 'export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre' >> ~/.bashrc
$ source ~/.bashrc
```

Τέλος πρέπει να ελέγξουμε ότι το &JAVA_HOME ζεί στο περιβάλλον και τελειώνει με την κατάληξη /jre:

```
$ echo $JAVA_HOME
   /usr/lib/jvm/java-8-openjdk-amd64/jre
```

4.2.3 ΒΗΜΑ 3: Εγκατάσταση ODL

Αρχικά θα κατεβάσουμε την επιθυμητή έκδοση (εν προκειμένω την 0.7.3 Nitrogen) από την επίσημη ιστοσελίδα:

```
$ wget https://nexus.opendaylight.org/content/repositories/opendaylight.release
    /org/opendaylight/integration/karaf/0.7.3/karaf-0.7.3.zip
```

Για την εγκατάσταση θα δημιουργήσουμε έναν φάκελο και θα μεταφέρουμε εκεί το αρχείο για να το αποσυμπιέσουμε:

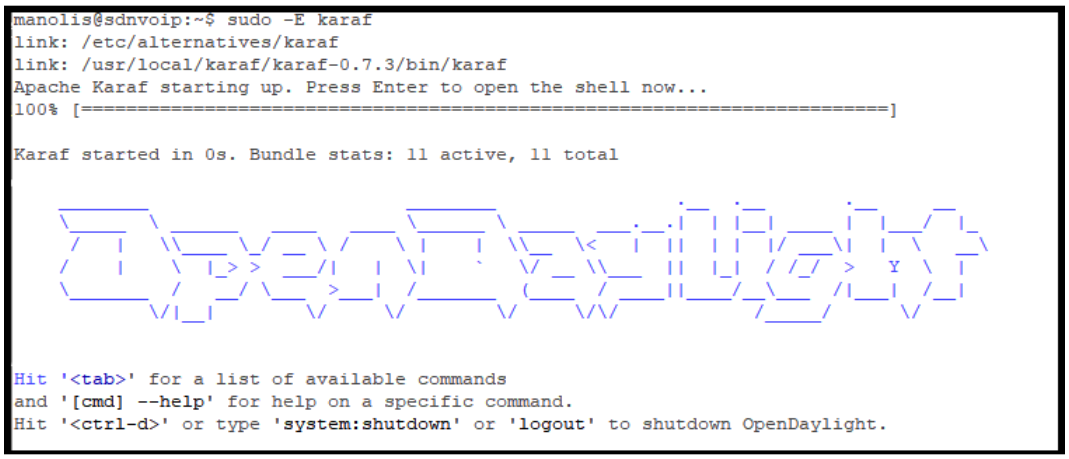
```
$ sudo mkdir /usr/local/karaf
$ sudo mv karaf-0.7.3.zip /usr/local/karaf
$ sudo unzip /usr/local/karaf/karaf-0.7.3.zip -d /usr/local/karaf/
```

Ακολουθεί η εγκατάσταση του λογισμικού:

```
$ sudo update-alternatives --install /usr/bin/karaf karaf /usr/local/karaf/
    karaf-0.7.3/bin/karaf 1
$ sudo update-alternatives --config karaf
$ which karaf
```

Για την εκτέλεση του ODL εκτελούμε την εντολή (βλ. Σχήμα 4.2):

```
$ sudo -E karaf
```



Σχήμα 4.2: Έναρξη Karaf

Στη συνέχεια θα χρειαστεί να εγκαταστήσουμε κάποια απαραίτητα χαρακτηριστικά. Τα χαρακτηριστικά του ODL μπορεί να τα εξερευνήσει κανείς πληκτρολογώντας:

```
$ opendaylight-user@root: feature:list
```

Από αυτά τα εγκατεστημένα μπορεί να τα δει κανείς πληκτρολογώντας:

```
$ opendaylight-user@root: feature:list -i
```

Για την παρούσα εργασία θα χρειαστούμε τα εξής:

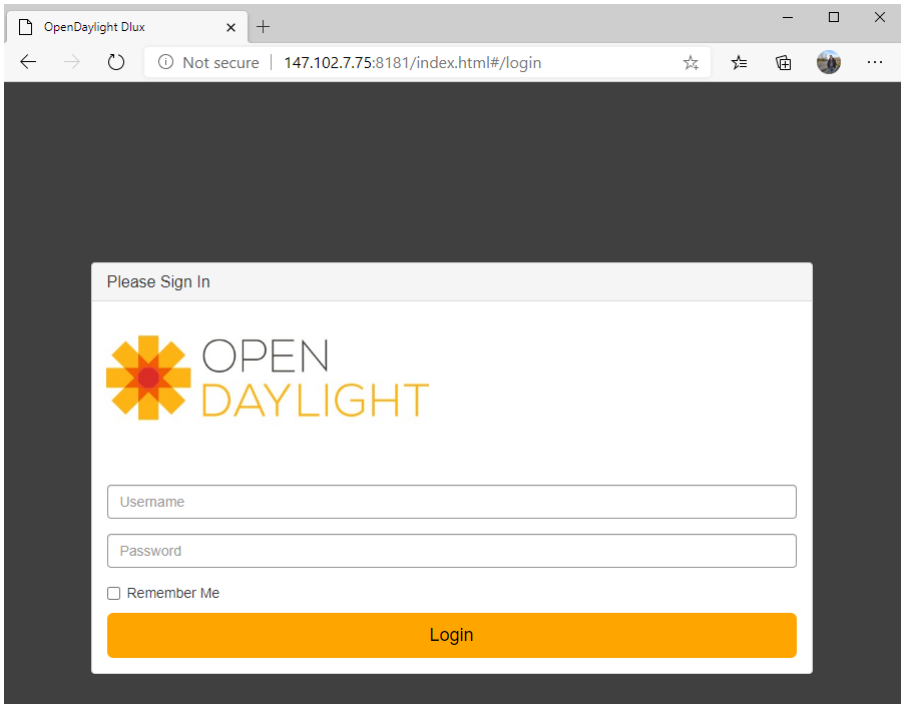
```
$ opendaylight-user@root: feature:install odl-l2switch-switch-ui
$ opendaylight-user@root: feature:install odl-mdsal-clustering
$ opendaylight-user@root: feature:install odl-dlux-core
$ opendaylight-user@root: feature:install odl-dluxapps-nodes
$ opendaylight-user@root: feature:install odl-dluxapps-yangui
```

Επιβεβαιώνουμε ότι όλα εκτελούνται καλά ελέγχοντας τη λειτουργία του γραφικού περιβάλλοντος του ODL από οποιοδήποτε φυλλομετρητή πληκτρολογώντας τη διεύθυνση `http://147.102.7.75:8181/index.html#/login` με τα στοιχεία username: admin, password: admin (βλ. Σχήμα 4.3).

Μέσω του γραφικού περιβάλλοντος μπορούμε να δούμε την τοπολογία του δικτύου μας πληκτρολογώντας τη διεύθυνση:
`http://147.102.7.75:8181/index.html#/topology`
από οποιοδήποτε μηχάνημα (βλ. Σχήμα 4.4), όπου απεικονίζεται το εικονικό switch openflow:348167938829696 το οποίο είναι το Hpe Switch που έχουμε συνδέσει μέσω Openflow.

4.2.4 PUT flows με POSTMAN

Εφόσον ολοκληρώθηκε η εγκατάσταση του ODL, θα πρέπει να ορίσουμε κάποιες ροές (flows) για να κατευθύνουμε τα πακέτα κατά βούληση. Αρχικά το ODL πλημμυρίζει το δίκτυο με πακέτα, δηλαδή τα πακέτα μοιράζονται στο δίκτυο και τα βλέπουν όλοι. Για να αποφύγουμε λοιπόν οποιαδήποτε σύγχυση θα πρέπει να προσθέσουμε κάποιες ροές για να περιορίσουμε



Σχήμα 4.3: YANG GUI ODL σελίδα log-in

την κίνηση και να είναι πιο ξεκάθαρη η ανάλυση των πακέτων. Συγκεκριμένα θα προσθέσουμε 3 ροές και θα εξηγήσουμε τη διαδικασία αναλυτικά στη συνέχεια.

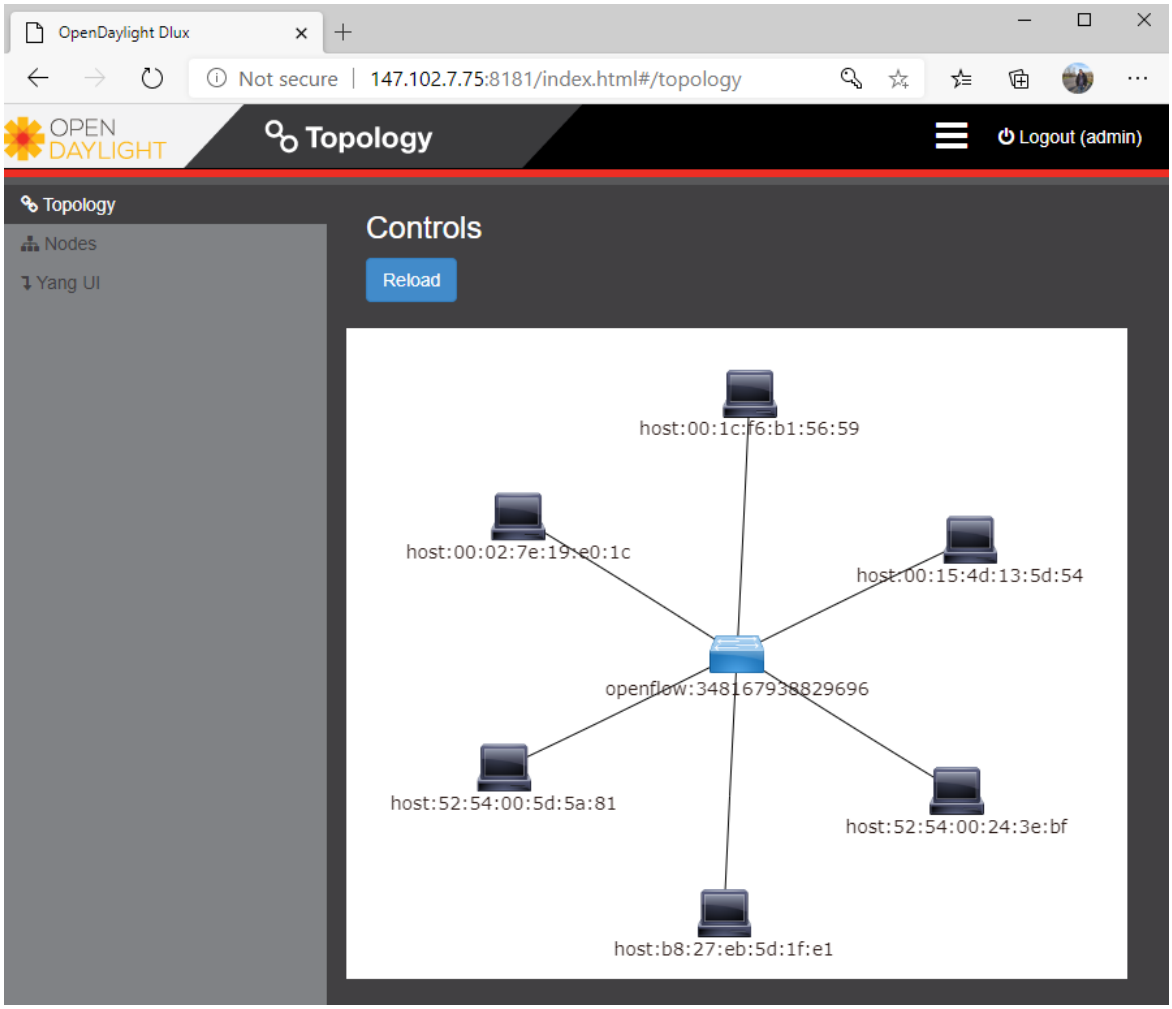
Το εργαλείο που θα χρησιμοποιήσουμε λέγεται POSTMAN και επιτρέπει την επικοινωνία μας με τον ODL Controller μέσω της διεπαφής REST. Παρόμοια με έναν HTTP εξυπηρετητή, θα στέλνουμε μηνύματα GET, PUT, DELETE κά. για να πάρουμε κάποια πληροφορία (πχ. πίνακας ροών), να τοποθετήσουμε κάποια ροή ή να διαγράψουμε κάποια ροή ή πίνακα ροών.

Αρχικά εκκινούμε το POSTMAN από οποιοδήποτε μηχάνημα. Σκοπός μας είναι να στείλουμε στον κατάλογο του restconf του ODL μία ροή που θα αντιστοιχεί στον πίνακα 0 και θα αναφέρεται στο switch που ελέγχουμε, δηλαδή το hpe, με αναγνωριστικό openflow:348167938829696.
Συνεπώς πληκτρολογούμε την παρακάτω διεύθυνση στην μπάρα εισαγωγής: URL: `http://147.102.7.75:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:348167938829696/table/0/flow/1`
όπου πληκτρολογούμε το Όνομα χρήστη και τον κωδικό, admin και admin αντίστοιχα (βλ. Σχήμα 4.5).

Στη συνέχεια επιλέγουμε να προσθέσουμε τις επικεφαλίδες. Στο μήνυμα PUT που θα στείλουμε πρέπει να αναφέρεται στην επικεφαλίδα ότι είναι αρχείο εφαρμογής xml. Για αυτόν τον λόγο το συμπληρώνουμε στη ροή μας (βλ. Σχήμα 4.6).

Τέλος προσθέτουμε τον κορμό του μηνύματος στην κατηγορία Body (βλ. Σχήμα 4.7).

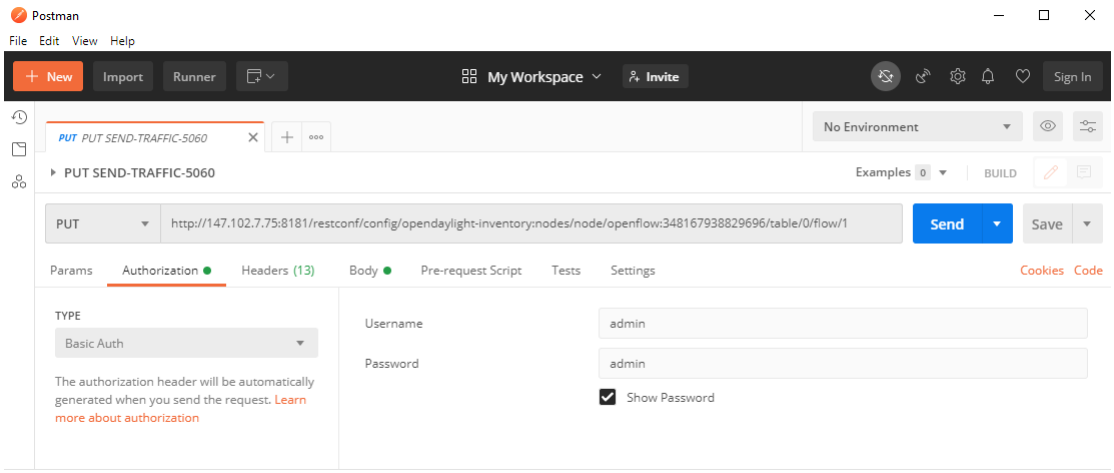
Το σώμα της ροής μας θα πρέπει να συμβαδίζει με τα πρότυπα. Περιλαμβάνει ένα σύνολο από κανόνες αντιστοίχισης και ενεργειών καθώς επίσης προτεραιότητα και όνομα. Εμείς θα προσθέσουμε, αρχικά, μια ροή που θα κατευθύνει όλη την SIP κίνηση από το Raspberry Pi



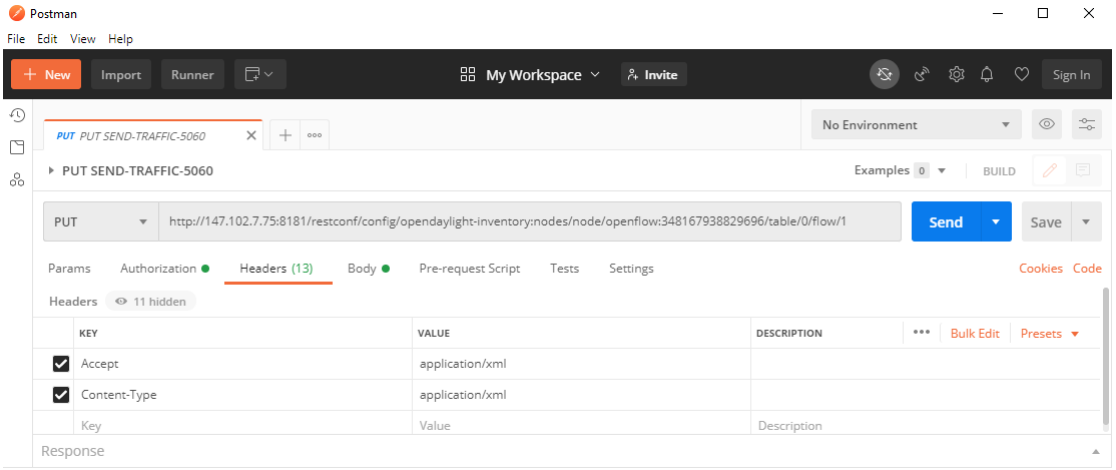
Σχήμα 4.4: Σελίδα Τοπολογίας ODL

στον διαμεσολαβητή SIP Kamailio (Βλ. Παράθεση 4.1).

```
{<flow xmlns="urn:opendaylight:flow:inventory">
    <priority>65535</priority>
    <flow-name> SEND TRAFFIC 5060 TO KAMAILIO </flow-name>
    <match>
        <ethernet-match>
            <ethernet-type>
                <type>2048</type>
            </ethernet-type>
            <ethernet-destination>
                <address>00:1c:f6:b1:56:59</address>
            </ethernet-destination>
            <ethernet-source>
                <address>b8:27:eb:5d:1f:e1</address>
            </ethernet-source>
        </ethernet-match>
        <ipv4-source>147.102.39.10/32</ipv4-source>
```

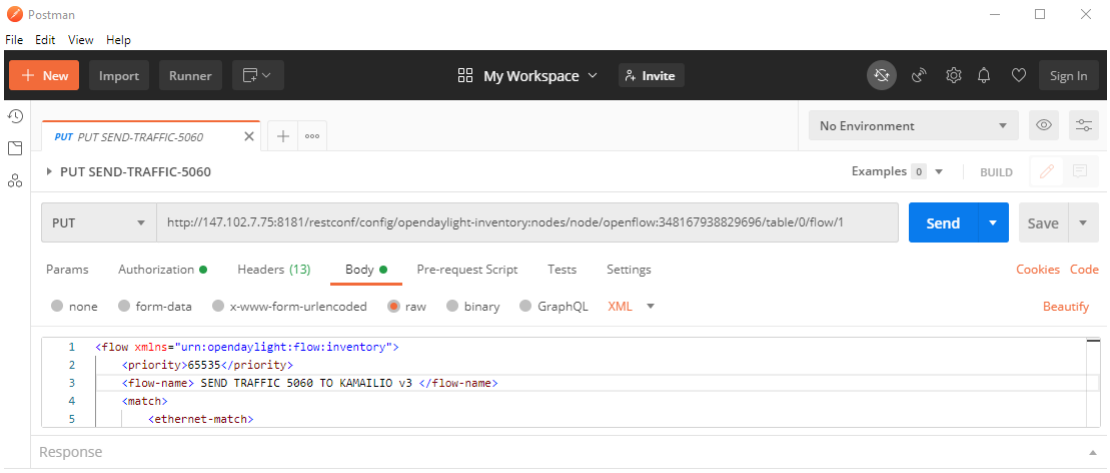


Σχήμα 4.5: Δημιουργία ροής στο ODL (α΄)



Σχήμα 4.6: Δημιουργία ροής στο ODL (β΄)

```
17          <ipv4-destination>147.102.7.18/32</ipv4-destination>
18          <ip-match>
19              <ip-protocol>17</ip-protocol>
20          </ip-match>
21          <udp-destination-port>5060</udp-destination-port>
22      </match>
23
24      <id>1</id>
25      <table_id>0</table_id>
26      <instructions>
27          <instruction>
28              <order>0</order>
29              <apply-actions>
30                  <action>
31                      <order>2</order>
32                      <output-action>
33                          <output-node-connector>28</output-node-connector>
```



Σχήμα 4.7: Δημιουργία ροής στο ODL (γ΄)

```
                        <max-length>60</max-length>
                    </output-action>
                </action>
                <action>
                    <order>1</order>
                    <set-dl-dst-action>
                        <address>52:54:00:db:a0:b1</address>
                    </set-dl-dst-action>
                </action>
                <action>
                    <order>0</order>
                    <set-nw-dst-action>
                        <ipv4-address>147.102.39.21/32</ipv4-address>
                    </set-nw-dst-action>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
</flow>
```

Παράθεση 4.1: Το σώμα της ροής 1 (κατεύθυνση κίνησης SIP προς τον Kamailio)

Στη συνέχεια επιλέγουμε στο POSTMAN να στείλει PUT και πατάμε το SEND. Ένα διακριτικό πράσινο μήνυμα θα εμφανιστεί στο πρόγραμμα με την ένδειξη 200 Created. Με την ίδια λογική θα προσθέσουμε άλλες δυο ροές που θα κατευθύνουν την κίνηση μεταξύ του RPi (147.102.39.10) και του Gateway (147.102.39.1) για να αποφύγουμε την πλημμύρα μηνυμάτων. Οι ροές παρουσιάζονται παρακάτω:

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <priority>100</priority>
    <flow-name> SEND ALL RPi TRAFFIC TO GATEWAY </flow-name>
    <match>
        <ethernet-match>
            <ethernet-type>
```

```
                <type>2048</type>
            </ethernet-type>
            <ethernet-destination>
                <address>00:1c:f6:b1:56:59</address>
            </ethernet-destination>
            <ethernet-source>
                <address>b8:27:eb:5d:1f:e1</address>
            </ethernet-source>
        </ethernet-match>
    </match>

    <id>2</id>
    <table_id>0</table_id>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                    <order>0</order>
                    <output-action>
                        <output-node-connector>1</output-node-connector>
                        <max-length>60</max-length>
                    </output-action>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
</flow>
```

Παράθεση 4.2: POH 2: Αποστολή όλων των πακέτων που προορίζονται για τον Gateway

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <priority>100</priority>
    <flow-name> SEND ALL RPi TRAFFIC TO GATEWAY </flow-name>
    <match>
        <ethernet-match>
            <ethernet-type>
                <type>2048</type>
            </ethernet-type>
            <ethernet-destination>
                <address>b8:27:eb:5d:1f:e1</address>
            </ethernet-destination>
            <ethernet-source>
                <address>00:1c:f6:b1:56:59 </address>
            </ethernet-source>
        </ethernet-match>
    </match>

    <id>3</id>
    <table_id>0</table_id>
    <instructions>
```

```
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                    <order>0</order>
                    <output-action>
                        <output-node-connector>5</output-node-connector>
                        <max-length>60</max-length>
                    </output-action>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
</flow>
```

Παράθεση 4.3: POH 3: Αποστολή όλων των πακέτων που προορίζονται για το Rpi

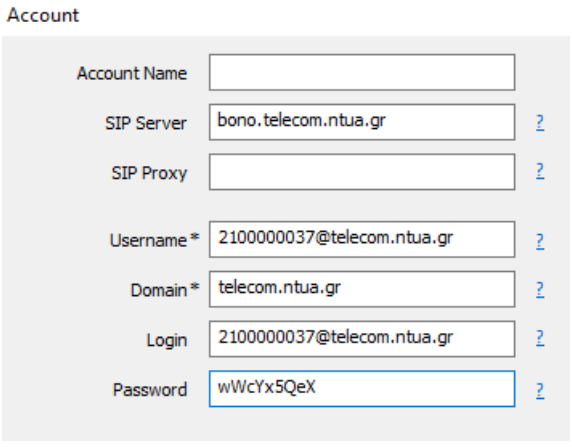
Ομοίως με προηγουμένως, θα πρέπει να εμφανιστεί αντίστοιχα ένα μήνυμα **200 Created**. Για να επιβεβαιώσουμε την ύπαρξη των ροών που μόλις προσθέσαμε, αρκεί να στείλουμε ένα μήνυμα GET στον εξυπηρετητή, με διεύθυνση:

```
http://147.102.7.75:8181/restconf/config/opendaylight-inventory:nodes/node/
    openflow:348167938829696/table/0}
```

όπου ζητάμε να δούμε τις ροές που είναι τοποθετημένες στον Πίνακα 0 του μεταγωγέα με διακριτικό openflow:348167938829696.

4.3 Δημιουργία λογαριασμού SIP στον Bono

Για τις ανάγκες του πειράματος χρειάστηκε να δημιουργήσουμε έναν χρήστη SIP που θα επικοινωνεί με το IMS και θα μπορεί να καλεί χρήστες συνδεδεμένους στο ClearWater IMS. Τα στοιχεία του λογαριασμού απεικονίζονται στο Σχημα 4.8.



Σχήμα 4.8: Λογαριασμός SIP στον Bono

Επίσης έχουμε δημιουργήσει και έναν δοκιμαστικό χρήστη για τις ανάγκες του πειράματος:

sip:2100000095@telecom.ntua.gr.

4.4 Δημιουργία πελάτη SIP με Python στο RPi

Με χρήση της βιβλιοθήκης pjsua που μελετήσαμε στη θεωρία, δημιουργήσαμε έναν πελάτη SIP με τα στοιχεία του λογαριασμού στο IMS, σε γλώσσα προγραμματισμού Python. Το πρόγραμμα sip_client.py εκτελεί την Εγγραφή στο IMS στέλνοντας μηνύματα SIP Register και ύστερα καλεί οποιοδήποτε χρήστη επιθυμούμε. Ο κώδικας παρουσιάζεται στην Παράθεση 4.4.

- sip_client.py

```
import sys
import pjsua as pj
import threading
import time

# Method to print Log of callback class
def log_cb(level, str, len):
    print(str),

# Account Callback class to get notifications of Account registration
class MyAccountCallback(pj.AccountCallback):
    def __init__(self, acc):
        pj.AccountCallback.__init__(self, acc)

# Call Callback to receive events from Call
class SRCallCallback(pj.CallCallback):
    def __init__(self, call=None):
        pj.CallCallback.__init__(self, call)


    def on_state(self):
        print("Call is :", self.call.info().state_text),
        print("last code :", self.call.info().last_code),
        print("(" + self.call.info().last_reason + ")")

    # This is the notification when call's media state is changed
    def on_media_state(self):
        global lib
        if self.call.info().media_state == pj.MediaState.ACTIVE:
            # Connect the call to sound device
            call_slot = self.call.info().conf_slot
            lib.conf_connect(call_slot, 0)
            lib.conf_connect(0, call_slot)
            print("Hey !!!!! Hope you are doing GOOD !!!!!!!!!!")
            print (lib)

```

```
# Lets start our main loop here

try:
    # Start of the Main Class
    # Create library instance of Lib class
    lib = pj.Lib()

    # Instantiate library with default config
    lib.init(log_cfg = pj.LogConfig(level=3, callback=log_cb))

    # Configuring one Transport Object and setting it to listen at 5060 port and UDP protocol
    trans_conf = pj.TransportConfig()

    print "------LETS START THE REGISTRATION PROCESS--------"
    print "\n\n"

    trans_conf.port = 5060       # 5060 is default port for SIP
    a="147.102.39.10"
    print "The IP address of the Client: 147.102.39.10 "
    print "Using the default port number for SIP: 5060"
    trans_conf.bound_addr = a
    transport = lib.create_transport(pj.TransportType.UDP,trans_conf)

    # Starting the instance of Lib class
    lib.start()
    lib.set_null_snd_dev()

    # Configuring Account class to register with Registrar server
    # Giving information to create header of REGISTER SIP message
    ab4="147.102.7.18"
    print "The IP address of the Server: 147.102.7.18"
    ab="2100000037@telecom.ntua.gr"
    print "Username: 2100000037@telecom.ntua.gr"
    ab1="wWcYx5QeX"
    print "Password: wWcYx5QeX "
    ab2=raw_input("Do you want to use the display name same as the username  Y/N ??")
    if ab2=="y" or ab2=="Y":
        ab3=ab
    else:
        ab3=raw_input("Enter Display Name: ")

    acc_conf = pj.AccountConfig(domain = ab4, username = ab, password =ab1, display = ab3)

        # registrar = 'sip:'+ab4+':5060', proxy = 'sip:'+ab4+':5060')

    acc_conf.id ="sip:"+ab
    acc_conf.reg_uri ='sip:'+ab4+':5060'
```

```
84      acc_callback = MyAccountCallback(acc_conf)
85      acc = lib.create_account(acc_conf,cb=acc_callback)
86
87      print('\n')
88      print "Registration Complete-----------"
89      print('Status= ',acc.info().reg_status, \
90
91            '(' + acc.info().reg_reason + ')')
92
93      ab5=raw_input("Do you want to make a call right now ??   Y/N\n")
94      print "\n"
95
96      if ab5=="y" or ab5=="Y":
97
98      # Starting Calling process.
99          b=raw_input("Enter the destination URI: ")
100         call = acc.make_call(b, SRCallCallback())
101
102         # Waiting Client side for ENTER command to exit
103         print('Press <ENTER> to exit and destroy library')
104         input = sys.stdin.readline().rstrip('\r\n')
105
106         # We're done, shutdown the library
107         lib.destroy()
108         lib = None
109
110     else:
111         print" Unregistering ----------------------------"
112         time.sleep(2)
113         print "Destroying Libraries --------------"
114         time.sleep(2)
115         lib.destroy()
116         lib = None
117         sys.exit(1)
118
119 except pj.Error, e:
120     print("Exception: " + str(e))
121     lib.destroy()
122     lib = None
123     sys.exit(1)
```

Παράθεση 4.4: Υλοποίηση Πελάτη SIP με Python: sip_client.py

4.5 Εγκατάσταση και ρύθμιση Agilio SmartNIC

Στην παρούσα υποενότητα θα ασχοληθούμε με τη ρύθμιση του μηχανήματος που διαθέτει την Agilio SmartNIC, παραμετροποιώντας κατάλληλα το δίκτυο, δημιουργώντας γέφυρα OVS και εγκαθιστώντας τελικά το KVM όπου θα δημιουργήσουμε δύο VMs, ένα με τον SIP Proxy Kamailio και ένα Ubuntu για παρακολούθηση της κίνησης SIP.

4.5.1 Εγκατάσταση Δικτύου

Βεβαιωνόμαστε ότι μόνο η enp0s35 έχει στατική IP:

```
$ sudo nano /etc/netplan/50-cloud-init.yaml
```

```
#This file is generated from information provided by the datasource.  Changes
# to it will not persist across an instance reboot.  To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    version: 2
    ethernets:
        enp0s25:
           dhcp4: no
           addresses: [147.102.40.78/24]
           gateway4: 147.102.40.200
           nameservers:
               addresses: [147.102.40.1]
               search: [cn.ntua.gr]
```

Μετά τις τροποποιήσεις στο /etc/netplan/50-cloud-init.yaml πρέπει να εφαρμόσουμε τις αλλαγές:

```
$ sudo netplan generate
$ sudo netplan apply
```

Προσθέτουμε στατική δρομολόγηση για ανακατεύθυνση της κίνησης προς τον bono μέσω της διεπαφής enp1s0np1np1 του SmartNIC:

```
$ sudo route add -host 147.102.7.18 gw 147.102.39.1
```

4.5.2 Έλεγχος και Παραμετροποίηση κάρτας δικτύου

Για να δούμε το Serial Number της διεπαφής του SmartNIC

```
$ sudo nano identification.sh
```

```
#!/bin/bash
DEVICE=$1
ethtool -W ${DEVICE} 0
DEBUG=$(ethtool -w ${DEVICE} data /dev/stdout | strings)
SERIAL=$(echo "${DEBUG}" | grep "^SN:")
ASSY=$(echo ${SERIAL} | grep -oE AMDA[0-9]{4})
echo ${SERIAL}
echo Assembly: ${ASSY}
```

Παράθεση 4.5: identification.sh

Για να χρησιμοποιήσουμε την εντολή strings θα χρειαστούμε το binutils:

```
$ sudo apt-get install binutils
```

```
$ chmod +x identification.sh
$ sudo ./identification.sh enp1s0
```

```
    SN: SMAAMDA0096-000117290690
    Assembly: AMDA0096
```

Στη συνέχεια θα εγκαταστήσουμε το TC Firmware:

```
$ sudo nano agilio-tc-fw-select.sh
```

```
#!/bin/bash
DEVICE=${1}
DEFAULT_ASSY=scan
ASSY=${2:-${DEFAULT_ASSY}}
APP=${3:-flower}

if [ "x${DEVICE}" = "x" -o ! -e /sys/class/net/${DEVICE} ]; then
    echo Syntax: ${0} device [ASSY] [APP]
    echo
    echo This script associates the TC Offload firmware
    echo with a Netronome SmartNIC.
    echo
    echo device: is the network device associated with the SmartNIC
    echo ASSY: defaults to ${DEFAULT_ASSY}
    echo APP: defaults to flower. flower-next is supported if updated
    echo      firmware has been installed.
    exit 1
fi

# It is recommended that the assembly be determined by inspection
# The following code determines the value via the debug interface
if [ "${ASSY}x" = "scanx" ]; then
    ethtool -W ${DEVICE} 0
    DEBUG=$(ethtool -w ${DEVICE} data /dev/stdout | strings)
    SERIAL=$(echo "${DEBUG}" | grep "^SN:")
    ASSY=$(echo ${SERIAL} | grep -oE AMDA[0-9]{4})
fi

PCIADDR=$(basename $(readlink -e /sys/class/net/${DEVICE}/device))
FWDIR="/lib/firmware/netronome"

# AMDA0081 and AMDA0097 uses the same firmware
if [ "${ASSY}" = "AMDA0081" ]; then
    if [ ! -e ${FWDIR}/${APP}/nic_AMDA0081.nffw ]; then
       ln -sf nic_AMDA0097.nffw ${FWDIR}/${APP}/nic_AMDA0081.nffw
   fi
fi

FW="${FWDIR}/pci-${PCIADDR}.nffw"
ln -sf "${APP}/nic_${ASSY}.nffw" "${FW}"

# insert distro-specific initramfs section here...
```

```
43
44 # Ubuntu 18.04 distro-specific initramfs section
45 HOOK=/etc/initramfs-tools/hooks/agilio_firmware
46 cat >${HOOK} << EOF
47 #!/bin/sh
48 PREREQ=""
49 prereqs()
50 {
51     echo "\$PREREQ"
52 }
53 case "\$1" in
54 prereqs)
55     prereqs
56     exit 0
57     ;;
58 esac
59 . /usr/share/initramfs-tools/hook-functions
60 cp "${FW}" "\${DESTDIR}${FW}"
61 if have_module nfp ; then
62     manual_add_modules nfp
63 fi
64 exit 0
65 EOF
66 chmod a+x "${HOOK}"
67 update-initramfs -u
```

Παράθεση 4.6: agilio-tc-fw-select.sh

```
$ sudo chmod +x agilio-tc-fw-select.sh
$ sudo ./agilio-tc-fw-select.sh enp1s0np1np1 scan
$ sudo rmmod nfp
$ sudo modprobe nfp
```

```
$ sudo apt-get install agilio-flower-app-firmware
```

Εγκατάσταση OVS:

```
$ systemctl status openvswitch-switch
$ systemctl status ovsdb-server
$ systemctl status ovs-vswitchd
```

Ενεργοποιούμε το OVS, έτσι ώστε να εκτελείται σε κάθε επανεκκίνηση του συστήματος:

```
$ systemctl enable openvswitch-switch
```

Ελέγχουμε την έκδοση που εγκαταστήσαμε (εδώ OVS 2.9.5):

```
$ ovs-vswitchd --version
```

Ελέγχουμε ότι είναι ενεργοποιημένο το IP Forwarding (ip forward=1)

```
$ cat /proc/sys/net/ipv4/ip_forward
```

Με την εντολή ethtool μπορούμε να δούμε όλα τα χαρακτηριστικά της φυσικής διεπαφής netdev:

```
$ ethtool -k enp1s0
```

Στη συνέχεια θα ενεργοποιήσουμε τα εξής:

- Receive Checksum Offload

  ```
  $ sudo ethtool -K enp1s0 rx on
  ```

- Transmit Checksum Offload

  ```
  $ sudo ethtool -K enp1s0 tx on
  ```

- Scatter/Gather

  ```
  $ sudo ethtool -K enp1s0 sg on
  ```

- TCP Segmentation Offload

  ```
  $ sudo ethtool -K enp1s0 tso on
  ```

- Generic Segmentation Offload

  ```
  $ sudo ethtool -K enp1s0 gso on
  ```

- Generic Receive Offload

  ```
  $ sudo ethtool -K enp1s0 gro on
  ```

Τέλος θα ρυθμίσουμε το TC HW offload στους εκπροσώπους κάθε πόρτας:

```
$ ethtool -k enp1s0np1np1 | grep hw-tc-offload
$ sudo ethtool -K enp1s0np1np1 hw-tc-offload on
$ ethtool -i enp1s0np1np1
```

Εγκατάσταση OVS Bridge

```
$ sudo ovs-vsctl add-br br1
$ sudo ovs-vsctl add-port br1 enp1s0np1np1
$ sudo ifconfig enp1s0np1np1 0
$ sudo ip addr add 147.102.39.20/26 dev br1
$ sudo ifconfig br1 up
$ sudo ip link set enp1s0np1np1 up
```

Ενεργοποίηση HW OFFLOAD:

```
$ sudo ovs-vsctl set Open_vSwitch . other_config:hw-offload=true other_config:
    tc-policy=none
$ sudo ovs-vsctl set Open_vSwitch . other_config:max-idle=60000
$ sudo systemctl restart openvswitch-switch
```

4.5.3 Δημιουργία VM1 και VM2 με χρήση KVM

Αρχικά εγκαθιστούμε τα απαραίτητα εργαλεία για την επιτυχή εκτέλεση του KVM:

```
$ sudo apt-get install qemu qemu-kvm libvirt-bin virtinst bridge-utils cpu-
    checker
$ sudo apt install virt-manager
```

Ελέγχουμε αν το σύστημα υποστηρίζει επιτάχυνση υλικού:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

Ελέγχουμε επίσης ότι υπάρχει το KVM:

```
$ sudo kvm-ok
```

Εκκινούμε το libvirtd για να ξεκινήσουμε την εγκατάσταση των VMs:

```
$ sudo service libvirtd start
$ sudo update-rc.d libvirtd enable
$ service libvirtd status
```

Στις εικονικές μηχανές που θα δημιουργήσουμε θα εγκαταστήσουμε τα Ubuntu 18.04. Συνεπώς θα κατεβάσουμε το αρχείο .img από την επίσημη ιστοσελίδα:

```
$ cd /home/manolis/ubuntu
$ sudo wget https://cloud-images.ubuntu.com/releases/bionic/release/ubuntu
    -18.04-server-cloudimg-amd64.img
```

Στη συνέχεια θα κάνουμε ένα ακριβές αντίγραφο του αρχείου για να το χρησιμοποιήσουμε για τη δεύτερη εικονική μηχανή που θέλουμε επίσης να είναι Ubuntu 18.04:

```
$ cp ubuntu-18.04-server-cloudimg-amd64.img ubuntu-18.04-server-cloudimg-
    amd64_1.img
```

Για το VM1 (Kamailio): Αρχικά θα δημιουργήσουμε ένα απαραίτητο αρχείο με τα στοιχεία εισόδου στην εικονική μηχανή και θα το ενσωματώσουμε στο αρχείο που κατεβάσαμε:

```
$ sudo nano /home/manolis/ubuntu/cloud_kamailio.txt
```

```
1  #cloud-config
2  password: w3lcome
3  chpasswd: { expire: False }
4  ssh_pwauth: True
5  hostname: kamailio
```

```
$ sudo cloud-localds /var/lib/libvirt/images/cloud_kamailio.img cloud_kamailio.
    txt
$ sudo qemu-img convert -f qcow2 ubuntu-18.04-server-cloudimg-amd64.img /var/
    lib/libvirt/images/kamailio.img
```

```
$ sudo virt-install  -n kamailio  \
--description "Kamailio on Ubuntu"  \
--os-type=Linux  \
--os-variant=rhel7  \
--ram=2048  --vcpus=2  \
```

```
--disk path=/home/manolis/ubuntu/ubuntu-18.04-server-cloudimg-amd64.img,bus=
    virtio,format=qcow2  \
--disk /var/lib/libvirt/images/cloud_kamailio.img,device=cdrom \
--network=bridge:br1,model=virtio,virtualport_type=openvswitch \
--graphics none
--noautoconsole
--import
```

Για το VM2 (MONITOR): Αντίστοιχα με προηγουμένως θα δημιουργήσουμε το απαραίτητο αρχείο με τα στοιχεία εισόδου και θα το ενσωματώσουμε στο αντίστοιχο αρχείο που τελικά θα χρησιμοποιήσουμε για την εγκατάσταση:

```
$ sudo nano /home/manolis/ubuntu/cloud_monitor.txt
```

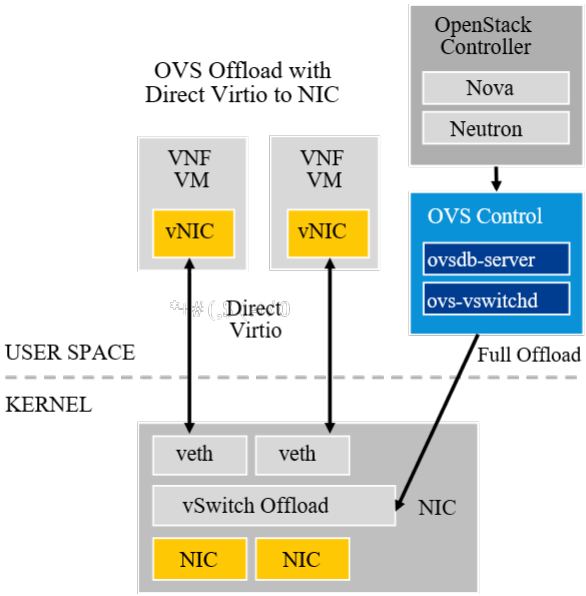
```
1  #cloud-config
2  password: w3lcome
3  chpasswd: { expire: False }
4  ssh_pwauth: True
5  hostname: monitor
```

```
$ sudo cloud-localds /var/lib/libvirt/images/cloud_monitor.img cloud_monitor.
    txt
$ sudo qemu-img convert -f qcow2 ubuntu-18.04-server-cloudimg-amd64_1.img /var/
    lib/libvirt/images/monitor.img
```

```
$ sudo virt-install  -n monitor  \
--description "VOIP Monitor"   \
--os-type=Linux  \
--os-variant=rhel7  \
--ram=2048  \
--vcpus=2  \
--disk path=/home/manolis/ubuntu/ubuntu-18.04-server-cloudimg-amd64_1.img,bus=
    virtio,format=qcow2  \
--disk /var/lib/libvirt/images/cloud_monitor.img,device=cdrom \
--network=bridge:br1,model=virtio,virtualport_type=openvswitch  \
--graphics none  \
--noautoconsole  \
--import \
```

Τέλος αφού ολοκληρώθηκε η εγκατάσταση των VMs θα συνδέσουμε τις διεπαφές τους με την γέφυρα br1 που δημιουργήσαμε νωρίτερα (βλ. Σχήμα 4.9):

```
$ sudo ovs-vsctl add-port br1 vnet0
$ sudo ovs-vsctl add-port br1 vnet1
$ sudo ip link set vnet0 up
$ sudo ip link set vnet1 up
```



Σχήμα 4.9: Μοντέλο Υλοποίησης γεφύρωσης OVS των VMs

Και φυσικά μπορούμε να συνδεθούμε στην αντίστοιχη κονσόλα για να τη ρυθμίσουμε:

```
$ virsh start kamailio
$ virsh start monitor
```

4.6 Εγκατάσταση Kamailio στο VM1

Αφού συνδεθώ στην κονσόλα του Kamailio ορίζω στατική IP για το VM1

```
ubuntu@kamailio:~$ sudo nano /etc/netplan/50-cloud-init.yaml
```

```
  GNU nano 2.9.3            /etc/netplan/50-cloud-init.yaml

# This file is generated from information provided by the datasource.  Changes
# to it will not persist across an instance reboot.  To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    ethernets:
        ens2:
            dhcp4: no
            addresses: [147.102.39.21/24]
            gateway4: 147.102.39.1
           # match:
           #     macaddress: 52:54:00:de:c1:65
           # set-name: ens2
    version: 2
```

Παράθεση 4.7: /etc/netplan/50-cloud-init.yaml (Kamailio)

Στη συνέχεια επιβάλλουμε τις αλλαγές:

```
$ netplan generate
$ netplan apply
```

Τέλος ρυθμίζουμε τον DNS Server προσθέτοντας την παρακάτω γραμμή στο αρχείο resolv.conf:

```
$ sudo nano /etc/resolv.conf
```

```
    nameserver 8.8.8.8
```

Για την εγκατάσταση του Kamailio απαιτείται πρώτα η προσθήκη των παρακάτω γραμμών στο αρχείο sources.list:

```
$ sudo nano /etc/apt/sources.list
```

```
deb http://cz.archive.ubuntu.com/ubuntu bionic main
deb     http://deb.kamailio.org/kamailio53 bionic main
deb-src http://deb.kamailio.org/kamailio53 bionic main
```

Τώρα μπορούμε να ενημερώσουμε το apt του Ubuntu και να εγκαταστήσουμε τα απαραίτητα modules:

```
$ sudo apt update
$ sudo apt install mariadb-server
$ sudo apt install kamailio kamailio-mysql-modules
$ sudo apt install kamailio-websocket-modules kamailio-tls-modules
```

Ελέγχουμε την έκδοση που εγκαταστήσαμε (εδώ kamailio 5.1.2):

```
$ kamailio -V
```

Ενεργοποιούμε τη μηχανή βάσης δεδομένων στο kamctlrc αφαιρώντας το σχόλιο (δηλαδή τη δίεση) μπροστά από το DBENGINE και DBHOST:

```
$ sudo nano /etc/kamailio/kamctlrc
```

Έπειτα δημιουργούμε τη βάση δεδομένων:

```
$ sudo kamdbctl create
```

```
MySQL password for root:
INFO: test server charset
WARNING: Your current default mysql characters set cannot be used to create DB.
     Please choice another one from the following list:
big5
dec8
cp850
hp8
koi8r
latin1
etc.
    Enter character set name:
    latin5
INFO: creating database kamailio ...
```

```
INFO: granting privileges to database kamailio ...
INFO: creating standard tables into kamailio ...
INFO: Core Kamailio tables succesfully created.
Install presence related tables? (y/n): y
INFO: creating presence tables into kamailio ...
INFO: Presence tables succesfully created.
Install tables for imc cpl siptrace domainpolicy carrierroute
     drouting userblacklist htable purple uac pipelimit mtree sca mohqueue
     rtpproxy rtpengine? (y/n): y
INFO: creating extra tables into kamailio ...
INFO: Extra tables succesfully created.
Install tables for uid_auth_db uid_avp_db uid_domain uid_gflags
     uid_uri_db? (y/n): y
INFO: creating uid tables into kamailio ...
INFO: UID tables succesfully created.
```

Παράθεση 4.8: Δημιουργία Βάσης Δεδομένων

Για να ολοκληρώσουμε την εγκατάσταση, προσθέτουμε στο αρχείο kamctlrc τη διεύθυνση IP 147.102.7.18 του SIP DOMAIN

```
$ sudo nano /etc/kamailio/kamctlrc
```

και στο αρχείο ρυθμίσεων kamailio.cfg προσθέτουμε τις εξής γραμμές:

```
$ sudo nano /etc/kamailio/kamailio.cfg
```

```
#!define WITH_MYSQL
#!define WITH_AUTH
#!define WITH_USRLOCDB
#!define WITH_ACCDB
```

Τέλος θα παραμετροποιήσουμε το αρχείο ρυθμίσεων του Kamailio, έτσι ώστε να προωθεί-ανακατευθύνει όλα τα πακέτα SIP, δηλαδή της πόρτας 5060, από το Raspberry Pi, στο Bono που είναι στην διεύθυνση 147.102.7.18. Αυτό θα γίνει αντικαθιστώντας την συνάρτηση request-route με την δική μας:

```
$ sudo nano /etc/kamailio/kamailio.cfg
```

```
/* Main SIP request routing logic
 * - processing of any incoming SIP request starts with this route
 * - note: this is the same as route { ... } */
request_route {
        if(src_ip==147.102.39.10) {
        $du = "sip:147.102.7.18:5060;transport=udp";
        xlog("L_INFO", "FORWARDING $rm FROM Rpi to $du \n");
        forward();
        };
}
```

Παράθεση 4.9: kamailio.cfg

Το Kamailio έχει εγκατασταθεί επιτυχώς, οπότε τώρα μπορούμε να το εκκινήσουμε:

```
$ sudo kamctl start
$ sudo service kamailio start
```

Η κατάσταση του καθώς και τα ενημερωτικά μηνύματα φαίνονται με την εντολή:

```
$ sudo systemctl status kamailio
```

4.7 Χρήση Monitor στο VM2

Αφού συνδεθώ στην κονσόλα του monitor ορίζω στατική IP για το VM2:

```
ubuntu@monitor:~$ sudo nano /etc/netplan/50-cloud-init.yaml
```

```
  GNU nano 2.9.3           /etc/netplan/50-cloud-init.yaml

# This file is generated from information provided by the datasource.  Changes
# to it will not persist across an instance reboot.  To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    ethernets:
        ens2:
            dhcp4: no
            addresses: [147.102.39.22/24]
            gateway4: 147.102.39.1
           # match:
           #     macaddress: 52:54:00:de:c1:65
           # set-name: ens2
    version: 2
```

Παράθεση 4.10: /etc/netplan/50-cloud-init.yaml (Monitor)

Στη συνέχεια επιβάλλουμε τις αλλαγές:

```
$ netplan generate
$ netplan apply
```

Τέλος ρυθμίζουμε τον DNS Server προσθέτοντας την παρακάτω γραμμή στο αρχείο resolv.conf:

```
$ sudo nano /etc/resolv.conf
```

```
    nameserver 8.8.8.8
```

Επόμενο βήμα είναι η προσθήκη ροών στο OVS για ¨καθρεπτισμό' της κίνησης στο VM2. Συνδεόμαστε στο host μηχάνημα και δημιουργούμε αρχικά πειθαρχίες ουρών (queue discipline) για την εξερχόμενη, από την διεπαφή enp1s0np1np1 και vnet0, κίνηση αντίστοιχα:

```
$ sudo tc qdisc add dev enp1s0np1np1 handle 1: root prio
$ sudo tc qdisc add dev vnet0 handle 1: root prio
```

Στη συνέχεια προσθέτουμε ροή στην διεπαφή enp1s0np1np1 όπου όλα τα εισερχόμενα και εξερχόμενα πακέτα ανεξαρτήτως πρωτοκόλλου καθρεπτίζονται στην διεπαφή vnet1:

```
$ sudo tc filter add dev enp1s0np1np1 parent ffff: protocol all u32 match u32 0
    0 action mirred egress mirror dev vnet1
$ sudo tc filter add dev enp1s0np1np1 parent 1: protocol all u32 match u32 0 0
   action mirred egress mirror dev vnet1
```

Ομοίως για την διεπαφή vnet0:

```
$ sudo tc filter add dev vnet0 parent ffff: protocol all u32 match u32 0 0
   action mirred egress mirror dev vnet1
$ sudo tc filter add dev vnet0 parent 1: protocol all u32 match u32 0 0 action
   mirred egress mirror dev vnet1
```

Για την παρακολούθηση της κίνησης SIP από το RaspberryPi εκτελούμε την εντολή:

```
$ tcpdump -i eth0 -n -e ether host b8:27:eb:5d:1f:e1
```

Κεφάλαιο 5

Διεξαγωγή & Αποτελέσματα

5.1 Λειτουργικότητα συστήματος

5.1.1 Επιβεβαίωση Offload

Αρχικά ελέγχουμε ότι γίνεται κανονικά η εκφόρτωση των πακέτων, δημιουργώντας μια κίνηση ICMP πακέτων χρησιμοποιώντας ping στον Kamailio:

```
manolis@RPi-CN-2:~ $ ping 147.102.39.21
PING 147.102.39.21 (147.102.39.21) 56(84) bytes of data.
64 bytes from 147.102.39.21: icmp_seq=1 ttl=64 time=3.23 ms
64 bytes from 147.102.39.21: icmp_seq=2 ttl=64 time=2.65 ms
...
```

Παράθεση 5.1: ping 147.102.39.21

Κατά τη διάρκεια του ping παρατηρούμε πώς αυξάνονται τα πακέτα που εκφορτώνονται :

```
manolis@snic:~$ sudo ovs-dpctl dump-flows | grep b8:27:eb:5d:1f:e1
in_port(4),eth(src=52:54:00:db:a0:b1,dst=b8:27:eb:5d:1f:e1),eth_type(0x0800),
    packets:4, bytes:336, used:0.070s, actions:3
in_port(3),eth(src=b8:27:eb:5d:1f:e1,dst=52:54:00:db:a0:b1),eth_type(0x0800),
    packets:4, bytes:336, used:0.070s, actions:4
in_port(3),eth(src=b8:27:eb:5d:1f:e1,dst=52:54:00:db:a0:b1),eth_type(0x0806),
    packets:0, bytes:0, used:4.060s, actions:4
manolis@snic:~$
manolis@snic:~$ sudo ovs-dpctl dump-flows | grep b8:27:eb:5d:1f:e1
in_port(4),eth(src=52:54:00:db:a0:b1,dst=b8:27:eb:5d:1f:e1),eth_type(0x0800),
    packets:5, bytes:420, used:0.740s, actions:3
in_port(3),eth(src=b8:27:eb:5d:1f:e1,dst=52:54:00:db:a0:b1),eth_type(0x0800),
    packets:5, bytes:420, used:0.740s, actions:4
in_port(3),eth(src=b8:27:eb:5d:1f:e1,dst=52:54:00:db:a0:b1),eth_type(0x0806),
    packets:0, bytes:0, used:5.740s, actions:4
manolis@snic:~$
manolis@snic:~$ sudo ovs-dpctl dump-flows | grep b8:27:eb:5d:1f:e1
in_port(4),eth(src=52:54:00:db:a0:b1,dst=b8:27:eb:5d:1f:e1),eth_type(0x0800),
    packets:10, bytes:840, used:0.950s, actions:3
```

```
in_port(3),eth(src=b8:27:eb:5d:1f:e1,dst=52:54:00:db:a0:b1),eth_type(0x0800),
    packets:10, bytes:840, used:0.950s, actions:4
in_port(3),eth(src=b8:27:eb:5d:1f:e1,dst=52:54:00:db:a0:b1),eth_type(0x0806),
    packets:0, bytes:0, used:10.960s, actions:4
manolis@snic:~$
manolis@snic:~$ sudo ovs-dpctl dump-flows | grep b8:27:eb:5d:1f:e1
in_port(4),eth(src=52:54:00:db:a0:b1,dst=b8:27:eb:5d:1f:e1),eth_type(0x0800),
    packets:13, bytes:1092, used:0.500s, actions:3
in_port(3),eth(src=b8:27:eb:5d:1f:e1,dst=52:54:00:db:a0:b1),eth_type(0x0800),
    packets:13, bytes:1092, used:0.500s, actions:4
in_port(3),eth(src=b8:27:eb:5d:1f:e1,dst=52:54:00:db:a0:b1),eth_type(0x0806),
    packets:0, bytes:0, used:13.510s, actions:4
manolis@snic:~$
manolis@snic:~$ sudo ovs-dpctl dump-flows | grep b8:27:eb:5d:1f:e1
in_port(4),eth(src=52:54:00:db:a0:b1,dst=b8:27:eb:5d:1f:e1),eth_type(0x0800),
    packets:16, bytes:1344, used:0.360s, actions:3
in_port(3),eth(src=b8:27:eb:5d:1f:e1,dst=52:54:00:db:a0:b1),eth_type(0x0800),
    packets:16, bytes:1344, used:0.370s, actions:4
in_port(3),eth(src=b8:27:eb:5d:1f:e1,dst=52:54:00:db:a0:b1),eth_type(0x0806),
    packets:0, bytes:0, used:16.380s, actions:4
```

Παράθεση 5.2: Πακέτα ICMP που εκφορτώθηκαν

Ύστερα από το ping από το Raspberry, ελέγχουμε όλα τα πακέτα που εκφορτώνονται από την πλευρά της SmartNIC:

```
manolis@snic:~$
sudo ovs-dpctl dump-flows type=offloaded
in_port(2),eth(src=52:54:00:24:3e:bf,dst=00:1c:f6:b1:56:59),eth_type(0x0800),
    packets:2441, bytes:382214, used:0.700s, actions:3
in_port(4),eth(src=52:54:00:db:a0:b1,dst=00:1c:f6:b1:56:59),eth_type(0x0800),
    packets:1445, bytes:231571, used:4.700s, actions:3
in_port(4),eth(src=52:54:00:db:a0:b1,dst=b8:27:eb:5d:1f:e1),eth_type(0x0800),
    packets:26, bytes:2184, used:0.650s, actions:3
in_port(4),eth(src=52:54:00:db:a0:b1,dst=b8:27:eb:5d:1f:e1),eth_type(0x0806),
    packets:0, bytes:0, used:21.500s, actions:3
in_port(3),eth(src=00:1c:f6:b1:56:59,dst=52:54:00:db:a0:b1),eth_type(0x0800),
    packets:1284, bytes:134056, used:4.380s, actions:4
in_port(3),eth(src=00:1c:f6:b1:56:59,dst=52:54:00:24:3e:bf),eth_type(0x0800),
    packets:2101, bytes:226827, used:0.350s, actions:2
in_port(3),eth(src=00:1c:f6:b1:56:59,dst=00:02:7e:19:e0:1c),eth_type(0x0800),
    packets:159, bytes:11164, used:1.730s, actions:drop
in_port(3),eth(src=b8:27:eb:5d:1f:e1,dst=52:54:00:db:a0:b1),eth_type(0x0800),
    packets:26, bytes:2184, used:0.650s, actions:4
in_port(3),eth(src=00:02:7e:19:e0:1c,dst=00:02:7e:19:e0:1c),eth_type(0x9000),
    packets:101, bytes:6060, used:9.440s, actions:drop
in_port(3),eth(src=3c:a8:2a:50:ed:a4,dst=01:80:c2:00:00:0e),eth_type(0x88cc),
    packets:33, bytes:8547, used:22.970s, actions:drop
in_port(3),eth(src=b8:27:eb:5d:1f:e1,dst=52:54:00:db:a0:b1),eth_type(0x0806),
    packets:0, bytes:0, used:21.500s, actions:4
```

Παράθεση 5.3: Όλα τα πακέτα που εκφορτώθηκαν

Όπως παρατηρούμε στην Παράθεση 5.3, εκφορτώνονται διάφορα πακέτα, εκ των οποίων συμπεριλαμβάνονται και πακέτα από και προς το Raspberry (Βλ. διεύθυνση b8:27:eb:5d:1f:e1).

5.1.2 Βασικές SIP κλήσεις

Στη συνέχεια θα επιχειρήσουμε να εκτελέσουμε τον κώδικα Python για τον Πελάτη SIP που αναφέραμε στο προηγούμενο κεφάλαιο, πραγματοποιώντας έτσι την εγγραφή του λογαριασμού **sip:2100000037@telecom.ntua.gr** στο IMS. Η διαδικασία εγγραφής πραγματοποιήθηκε επιτυχώς.

```
manolis@RPi-CN-2:~ $ sudo python sip_client.py
14:50:42.096 os_core_unix.c !pjlib 2.10-dev for POSIX initialized
14:50:42.098 sip_endpoint.c  .Creating endpoint instance...
14:50:42.098          pjlib  .select() I/O Queue created (0x1f8a448)
14:50:42.098 sip_endpoint.c  .Module "mod-msg-print" registered
14:50:42.098 sip_transport.  .Transport manager created.
14:50:42.098   pjsua_core.c  .PJSUA state changed: NULL --> CREATED
14:50:42.174   pjsua_core.c  .pjsua version 2.10-dev for Linux-4.9.35/armv7l/
    glibc-2.19 initialized
----------LETS START THE REGISTRATION PROCESS-----------

The IP address of the Client: 147.102.39.10
Using the default port number for SIP: 5060
The IP address of the Server: 147.102.7.18
Username: 2100000037@telecom.ntua.gr
Password: wWcYx5QeX
Do you want to use the display name same as the username  Y/N ??y

Registration Complete-----------
('Status= ', 100, '(In Progress)')
Do you want to make a call right now ??   Y/N
15:01:33.885    pjsua_acc.c !....sip:2100000037@telecom.ntua.gr: registration
    success, status=200 (OK), will re-register in 300 seconds
```

Παράθεση 5.4: sip_client.py Εγγραφή χρήστη

Όπως αποδείχθηκε ήδη τα πακέτα μεταδίδονται μέσω του Kamailio, όπως αναμενόταν. Από την πλευρά του Kamailio μπορούμε να το επιβεβαιώσουμε ελέγχοντας την κατάσταση του Kamailio:

```
ubuntu@kamailio:~$ sudo systemctl status kamailio > kamailio_status.txt
ubuntu@kamailio:~$ cat kamailio_status.txt
kamailio.service - Kamailio (OpenSER) - the Open Source SIP Server
   Loaded: loaded (/lib/systemd/system/kamailio.service; enabled; vendor preset
    : enabled)
   Active: active (running) since Wed 2020-10-14 14:48:53 UTC; 4min 35s ago
  Process: 31341 ExecStart=/usr/sbin/kamailio -P /var/run/kamailio/kamailio.pid
     -f $CFGFILE -m $SHM_MEMORY -M $PKG_MEMORY -u $USER -g $GROUP (code=exited,
     status=0/SUCCESS)
 Main PID: 31355 (kamailio)
```

```
8     Tasks: 32 (limit: 2362)
9    CGroup: /system.slice/kamailio.service
10           31355 /usr/sbin/kamailio -P /var/run/kamailio/kamailio.pid -f /etc/
     kamailio/kamailio.cfg -m 64 -M 8 -u kamailio -g kamailio
11           ....
12 Oct 14 14:48:52 kamailio /usr/sbin/kamailio[31400]: INFO: jsonrpcs [
     jsonrpcs_sock.c:443]: jsonrpc_dgram_process(): a new child 0/31400
13 Oct 14 14:48:52 kamailio /usr/sbin/kamailio[31403]: INFO: ctl [io_listener.c
     :214]: io_listen_loop(): io_listen_loop:  using epoll_lt io watch method (
     config)
14 Oct 14 14:48:53 kamailio systemd[1]: Started Kamailio (OpenSER) - the Open
     Source SIP Server.
15 Oct 14 15:01:33 kamailio /usr/sbin/kamailio[31375]: {1 6182 REGISTER
     JvZkPEpCdirT6C0aC5hGE-pJ.QaYrby1} INFO: <script>: FORWARDING REGISTER FROM
     Rpi to sip:147.102.7.18:5060;transport=udp
16 Oct 14 15:01:33 kamailio /usr/sbin/kamailio[31377]: {1 6182 REGISTER
     JvZkPEpCdirT6C0aC5hGE-pJ.QaYrby1} INFO: <script>: FORWARDING REGISTER FROM
     Rpi to sip:147.102.7.18:5060;transport=udp
```

Παράθεση 5.5: Κατάσταση Kamailio

Οι δυο τελευταίες καταχωρήσεις στο log αρχείο (Βλ. Παράθεση 5.5) δείχνουν την επιτυχή προώθηση των SIP Register μηνυμάτων προς τον Bono. Τέλος επαναλαμβάνουμε την εκτέλεση του κώδικα, πραγματοποιώντας ξανά την εγγραφή του χρήστη, αυτήν τη φορά όμως θα δοκιμάσουμε να καλέσουμε τον δοκιμαστικό αριθμό sip:2100000095@telecom.ntua.gr:

```
1  manolis@RPi-CN-2:~ $ sudo python sip_client.py
2  [sudo] password for manolis:
3  16:58:34.117 os_core_unix.c !pjlib 2.10-dev for POSIX initialized
4  16:58:34.118 sip_endpoint.c  .Creating endpoint instance...
5  16:58:34.118          pjlib  .select() I/O Queue created (0x1fb6448)
6  16:58:34.118 sip_endpoint.c  .Module "mod-msg-print" registered
7  16:58:34.118 sip_transport.  .Transport manager created.
8  16:58:34.118   pjsua_core.c  .PJSUA state changed: NULL --> CREATED
9  16:58:34.195   pjsua_core.c  .pjsua version 2.10-dev for Linux-4.9.35/armv7l/
     glibc-2.19 initialized
10 --------LETS START THE REGISTRATION PROCESS---------
11
12 The IP address of the Client: 147.102.39.10
13 Using the default port number for SIP: 5060
14 The IP address of the Server: 147.102.7.18
15 Username: 2100000037@telecom.ntua.gr
16 Password: wWcYx5QeX
17 Do you want to use the display name same as the username  Y/N ??y
18
19 Registration Complete-----------
20 ('Status= ', 100, '(In Progress)')
21 Do you want to make a call right now ??   Y/N
22 16:58:35.887    pjsua_acc.c !....sip:2100000037@telecom.ntua.gr: registration
     success, status=200 (OK), will re-register in 300 seconds
```

```
y

Enter the destination URI: sip:2100000095@telecom.ntua.gr
('Call is :', 'CALLING') ('last code :', 0) ()
Press <ENTER> to exit and destroy library
('Call is :', 'EARLY') ('last code :', 180) (Ringing)
('Call is :', 'CONNECTING') ('last code :', 200) (OK)
Hey !!!!! Hope you are doing GOOD !!!!!!!!!!
Lib
('Call is :', 'CONFIRMED') ('last code :', 200) (OK)
^C
16:59:03.831    pjsua_acc.c  .....sip:2100000037@telecom.ntua.gr:
    unregistration success
```

Παράθεση 5.6: sip_client.py: Κλήση προς sip:2100000095@telecom.ntua.gr

Ομοίως με προηγουμένως, παρακολουθούμε την κίνηση στο Monitor:

```
ubuntu@monitor:/$ sudo tcpdump -n ether host b8:27:eb:5d:1f:e1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens2, link-type EN10MB (Ethernet), capture size 262144 bytes

16:58:35.621248 IP 147.102.39.10.5060 > 147.102.39.21.5060: SIP: REGISTER sip
    :147.102.7.18:5060 SIP/2.0
16:58:35.737647 IP 147.102.39.21.5060 > 147.102.39.10.5060: SIP: SIP/2.0 401
    Unauthorized
16:58:35.788826 IP 147.102.39.10.5060 > 147.102.39.21.5060: SIP: REGISTER sip
    :147.102.7.18:5060 SIP/2.0
16:58:35.867942 IP 147.102.39.21.5060 > 147.102.39.10.5060: SIP: SIP/2.0 200 OK
16:58:40.952072 ARP, Request who-has 147.102.39.10 tell 147.102.39.21, length
    28
16:58:40.960145 ARP, Reply 147.102.39.10 is-at b8:27:eb:5d:1f:e1, length 46
16:58:47.454142 IP 147.102.39.10.5060 > 147.102.39.21.5060: SIP: INVITE sip
    :2100000037@telecom.ntua.gr SIP/2.0
16:58:47.538457 IP 147.102.39.21.5060 > 147.102.39.10.5060: SIP: SIP/2.0 100
    Trying
16:58:47.886917 IP 147.102.39.21.5060 > 147.102.39.10.5060: SIP: SIP/2.0 180
    Ringing
16:58:50.401079 IP 147.102.39.21.5060 > 147.102.39.10.5060: SIP: SIP/2.0 200 OK
16:58:50.455416 IP 147.102.39.10.5060 > 147.102.39.21.5060: SIP: ACK sip
    :2100000095@telecom.ntua.gr:5060;ob SIP/2.0
16:58:50.916846 IP 147.102.39.10.5060 > 147.102.39.21.5060: SIP
16:59:03.604191 IP 147.102.39.10.5060 > 147.102.39.21.5060: SIP: BYE sip
    :2100000095@telecom.ntua.gr:5060;ob SIP/2.0
16:59:03.606990 IP 147.102.39.10.5060 > 147.102.39.21.5060: SIP: REGISTER sip
    :147.102.7.18:5060 SIP/2.0
16:59:03.686572 IP 147.102.39.21.5060 > 147.102.39.10.5060: SIP: SIP/2.0 401
    Unauthorized
16:59:03.689117 IP 147.102.39.10.5060 > 147.102.39.21.5060: SIP: REGISTER sip
    :147.102.7.18:5060 SIP/2.0
16:59:03.832752 IP 147.102.39.21.5060 > 147.102.39.10.5060: SIP: SIP/2.0 200 OK
```

Παράθεση 5.7: Παρακολούθηση πακέτων κλήσης SIP

5.1.3 Monitor

Από την πλευρά του VM2 (Monitor) έχουμε πλήρη εικόνα της κίνησης SIP και παρατηρούμε τα πακέτα από και προς το RPi και Bono αντίστοιχα:

```
ubuntu@monitor:/$ sudo tcpdump -n ether host b8:27:eb:5d:1f:e1 -vv
tcpdump: listening on ens2, link-type EN10MB (Ethernet), capture size 262144
    bytes
```

```
15:01:33.586870 IP (tos 0x0, ttl 64, id 41572, offset 0, flags [DF], proto UDP
    (17), length 597)
    147.102.39.10.5060 > 147.102.39.21.5060: [udp sum ok] SIP, length: 569
        REGISTER sip:147.102.7.18:5060 SIP/2.0
        Via: SIP/2.0/UDP 147.102.39.10:5060;rport;branch=z9hG4bKPjN.
    OaZB04uoWw8ZK.j3YV5Aq-ocU0u9rr
        Route: <sip:147.102.7.18;lr>
        Max-Forwards: 70
        From: <sip:2100000037@telecom.ntua.gr>;tag=1LrwYv0nb-pIYvMezaXqB9Noxc.1
    tzC0
        To: <sip:2100000037@telecom.ntua.gr>
        Call-ID: M30.cP0GiutkRng1BYM.oW6-3Z9yUe1j
        CSeq: 35509 REGISTER
        User-Agent: pjsip python
        Contact: <sip:2100000037@147.102.39.10:5060;ob>
        Expires: 300
        Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, INFO, SUBSCRIBE, NOTIFY
    , REFER, MESSAGE, OPTIONS
        Content-Length:  0
```

Παράθεση 5.8: REGISTER: Εγγραφή χρήστη (1η απόπειρα)

```
15:01:33.726046 IP (tos 0x10, ttl 64, id 21322, offset 0, flags [none], proto
    UDP (17), length 549)
    147.102.39.21.5060 > 147.102.39.10.5060: [bad udp cksum 0x770e -> 0x56e6!]
    SIP, length: 521
        SIP/2.0 401 Unauthorized
        Via: SIP/2.0/UDP 147.102.39.10:5060;rport=5060;received=147.102.39.10;
    branch=z9hG4bKPjN.OaZB04uoWw8ZK.j3YV5Aq-ocU0u9rr
        Call-ID: M30.cP0GiutkRng1BYM.oW6-3Z9yUe1j
        From: <sip:2100000037@telecom.ntua.gr>;tag=1LrwYv0nb-pIYvMezaXqB9Noxc.1
    tzC0
        To: <sip:2100000037@telecom.ntua.gr>;tag=
    z9hG4bKPjB9GGHJ7vKgWitsJAZqnYKPLdOYr1P4uV
        CSeq: 35509 REGISTER
        WWW-Authenticate: Digest  realm="telecom.ntua.gr",nonce="5433
    a3a155c498f5",opaque="50095853594d45b9",algorithm=MD5,qop="auth"
        Content-Length:  0
```

Παράθεση 5.9: 401 Unauthorized: Ανεπιτυχής Εγγραφή χρήστη (1η απόπειρα)

```
15:01:33.775494 IP (tos 0x0, ttl 64, id 41583, offset 0, flags [DF], proto UDP
    (17), length 890)
```

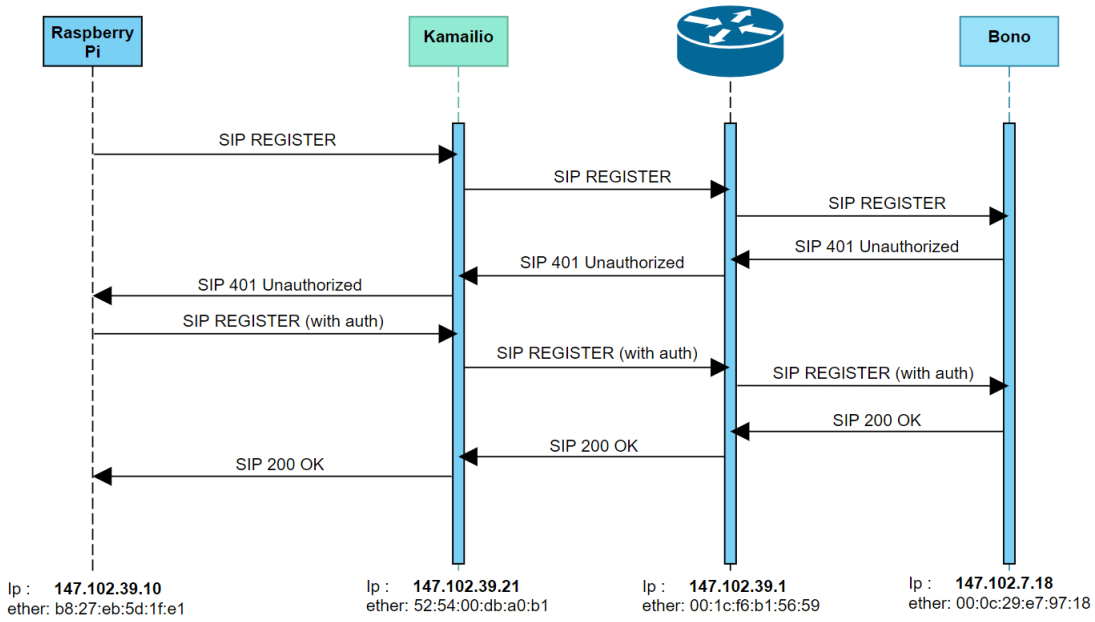
```
    147.102.39.10.5060 > 147.102.39.21.5060: [udp sum ok] SIP, length: 862
        REGISTER sip:147.102.7.18:5060 SIP/2.0
        Via: SIP/2.0/UDP 147.102.39.10:5060;rport;branch=
    z9hG4bKPjyx8KQDbABLHrlSCkScg50vI8GWBLtt2z
        Route: <sip:147.102.7.18;lr>
        Max-Forwards: 70
        From: <sip:2100000037@telecom.ntua.gr>;tag=1LrwYv0nb-pIYvMezaXqB9Noxc.1
    tzC0
        To: <sip:2100000037@telecom.ntua.gr>
        Call-ID: M30.cP0GiutkRng1BYM.oW6-3Z9yUe1j
        CSeq: 35510 REGISTER
        User-Agent: pjsip python
        Contact: <sip:2100000037@147.102.39.10:5060;ob>
        Expires: 300
        Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, INFO, SUBSCRIBE, NOTIFY
    , REFER, MESSAGE, OPTIONS
        Authorization: Digest username="2100000037@telecom.ntua.gr", realm="
    telecom.ntua.gr", nonce="5433a3a155c498f5", uri="sip:147.102.7.18:5060",
    response="83644f6a698f413481a95493c3903a20", algorithm=MD5, cnonce=".
    fmtEfYYthF7IPVaTzVFl6h1rw9.NVr", opaque="50095853594d45b9", qop=auth, nc
    =00000001
        Content-Length:  0
```

Παράθεση 5.10: REGISTER: Εγγραφή χρήστη (2η απόπειρα)

```
15:01:33.885196 IP (tos 0x10, ttl 64, id 21342, offset 0, flags [none], proto
    UDP (17), length 739)
    147.102.39.21.5060 > 147.102.39.10.5060: [bad udp cksum 0x77cc -> 0xd573!]
    SIP, length: 711
        SIP/2.0 200 OK
        Service-Route: <sip:scscf.sprout.telecom.ntua.gr;transport=TCP;lr;orig;
    username=2100000037%40telecom.ntua.gr;nonce=5433a3a155c498f5>
        Via: SIP/2.0/UDP 147.102.39.10:5060;rport=5060;received=147.102.39.10;
    branch=z9hG4bKPjyx8KQDbABLHrlSCkScg50vI8GWBLtt2z
        Call-ID: M30.cP0GiutkRng1BYM.oW6-3Z9yUe1j
        From: <sip:2100000037@telecom.ntua.gr>;tag=1LrwYv0nb-pIYvMezaXqB9Noxc.1
    tzC0
        To: <sip:2100000037@telecom.ntua.gr>;tag=z9hG4bKPjD-
    aa6EymTbzhj1F53S5bdX.MfZ9B8pVE
        CSeq: 35510 REGISTER
        Supported: outbound
        Contact: <sip:2100000037@147.102.39.10:5060;ob>;expires=300
        Path: <sip:NCQolzJ7Cw@147.102.7.18:5058;transport=TCP;lr>
        P-Associated-URI: sip:2100000037@telecom.ntua.gr
```

Παράθεση 5.11: 200 OK: Επιτυχής Εγγραφή χρήστη (2η απόπειρα)

Στο Σχήμα 5.1 απεικονίζεται αναλυτικά η ροή των πακέτων για την εγγραφή του χρήστη SIP.



Σχήμα 5.1: Ροή πακέτων για την εγγραφή χρήστη SIP

5.2 Ανάλυση απόδοσης συστήματος

Στην παρούσα ενότητα θα μελετήσουμε την επίδραση που έχει η προσθήκη του Monitor στο σύστημα μας και κατά πόσο βελτιώνεται η απόδοση της συνολικής υπηρεσίας με την προσθήκη του χαρακτηριστικού HW-Offload που αναλύθηκε εξονυχιστικά στα προηγούμενα κεφάλαια, δηλαδή τη διαδικασία εκφόρτωσης κοστοβόρων, για τον επεξεργαστή, διαδικασιών. Θυμίζουμε ότι λειτουργεί το ODL και προωθεί κανονικά όλα τα SIP πακέτα με προορισμό το 147.102.7.18 μέσω του Kamailio. Θα ερευνήσουμε κατά πόσο επηρεάζεται η επίδοση του συστήματος με την προσθήκη του Monitor με χρήση εικονικού περιβάλλοντος KVM στο μηχάνημα που είναι εγκατεστημένη η SmartNIC. Επίσης αναμένουμε να δούμε μια βελτίωση στην κατανομή της επεξεργαστικής ισχύος μετά την ενεργοποίηση της λειτουργίας εκφόρτωσης πακέτων (hw-offload). Τέλος θα μελετήσουμε κατά πόσο επηρεάζεται η καθυστέρηση μετάδοσης (latency), δηλαδή ο χρόνος που απαιτείται για να μεταδοθεί το μήνυμα SIP REGISTER συν τον αντίστοιχο χρόνο για να επιστρέψει η απάντηση SIP 401 από τον Bono (147.102.7.18) (βλ. Σχήμα 5.1).

Οι εντολές που θα τρέξουμε στο RPi θα είναι οι εξής:

```
1 $ sudo sipp 147.102.7.18:5060 -sf /sipp/docs/register5.xml -r 1 -rp 1000 -
    trace_stat -timeout 10
2 $ sudo sipp 147.102.7.18:5060 -sf /sipp/docs/register5.xml -r 5 -rp 1000 -
    trace_stat -timeout 10
3 $ sudo sipp 147.102.7.18:5060 -sf /sipp/docs/register5.xml -r 10 -rp 1000 -
    trace_stat -timeout 10
4 $ sudo sipp 147.102.7.18:5060 -sf /sipp/docs/register5.xml -r 100 -rp 1000 -
    trace_stat -timeout 10
```

```
$ sudo sipp 147.102.7.18:5060 -sf /sipp/docs/register5.xml -r 1000 -rp 1000 -
    trace_stat -timeout 10
$ sudo sipp 147.102.7.18:5060 -sf /sipp/docs/register5.xml -r 10000 -rp 1000 -
    trace_stat -timeout 10
```

Παράθεση 5.12: Εντολές δοκιμών

Για την παρακολούθηση των πόρων της CPU θα χρησιμοποιήσουμε την εξής εντολή, όπου μετράμε το CPU Utilization κάθε 1 sec για διάστημα 20 sec:

```
$ mpstat 1 20
```

Δοκιμάζουμε δυο σενάρια. Ένα με Monitor και ένα χωρίς. Σε κάθε περίπτωση το hw-offload είναι απενεργοποιημένο! Η χρήση Monitor υλοποιήθηκε προσθέτοντας φίλτρα αντικατοπτρισμού/ανακατεύθυνσης στη διεπαφή enp1s0np1np1 της SmartNIC, όπως περιγράφηκε στην υποενότητα 4.7. Στη συνέχεια κάνουμε την τελευταία δοκιμή μας, με ενεργοποιημένο το HW-Offload και το Monitor, για να δούμε πώς επηρεάζεται τελικά η επίδοση του συστήματος και της ποιότητας της υπηρεσίας (QoS).

Παρακάτω αναφέρονται οι εντολές που χρειάστηκαν για την ενεργοποίηση και απενεργοποίηση του HW-Offload:

ΓΙΑ ΤΗΝ ΑΠΕΝΕΡΓΟΠΟΙΗΣΗ ΤΟΥ hw-offload:

```
$ sudo ovs-vsctl set Open_vSwitch . other_config:hw-offload=false other_config:
    tc-policy=none
$ sudo systemctl restart openvswitch-switch
$ sudo ethtool -K enp1s0np1np1 hw-tc-offload off
```

ΕΛΕΓΧΟΣ:

```
$ ethtool -k enp1s0np1np1 | grep hw-tc-offload  //check hw-offload is off
$ sudo ovs-dpctl dump-flows | grep b8:27:eb:5d:1f:e1  //check RPi packets
```

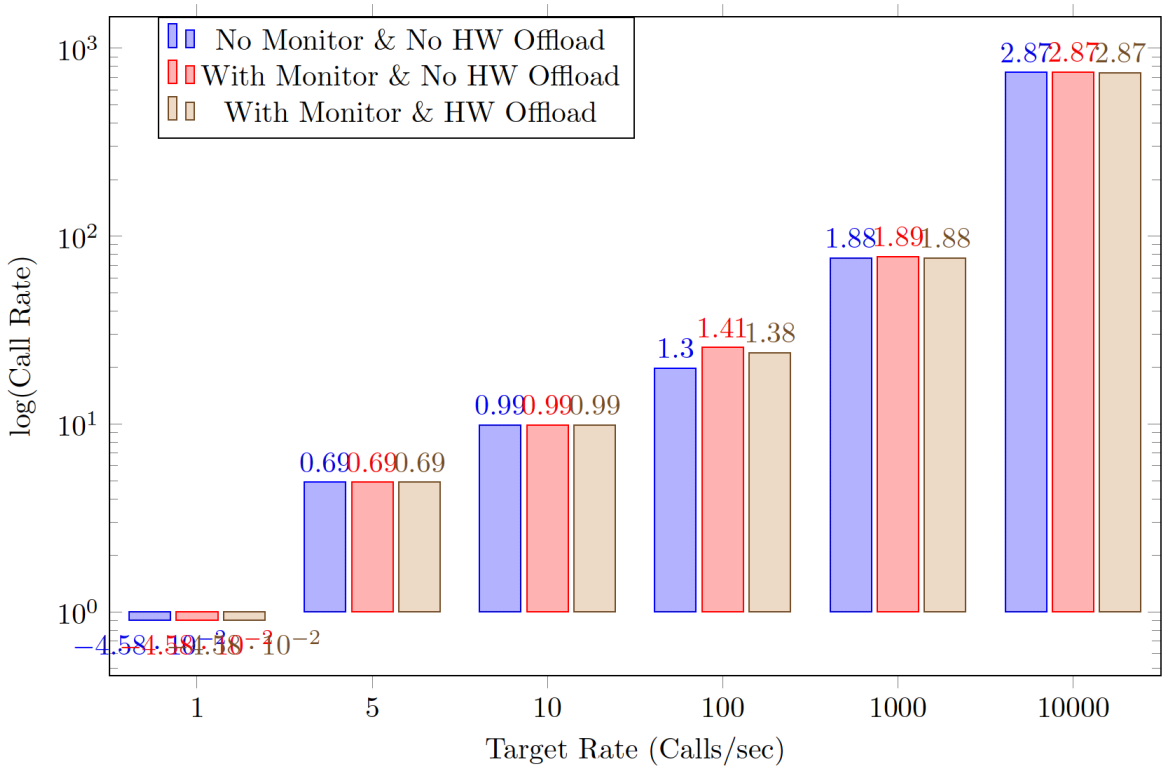
ΓΙΑ ΤΗΝ ΕΝΕΡΓΟΠΟΙΗΣΗ ΤΟΥ hw-offload:

```
$ sudo ethtool -K enp1s0np1np1 hw-tc-offload on
$ sudo ovs-vsctl set Open_vSwitch . other_config:hw-offload=true other_config:
    tc-policy=none
$ sudo systemctl restart openvswitch-switch
```

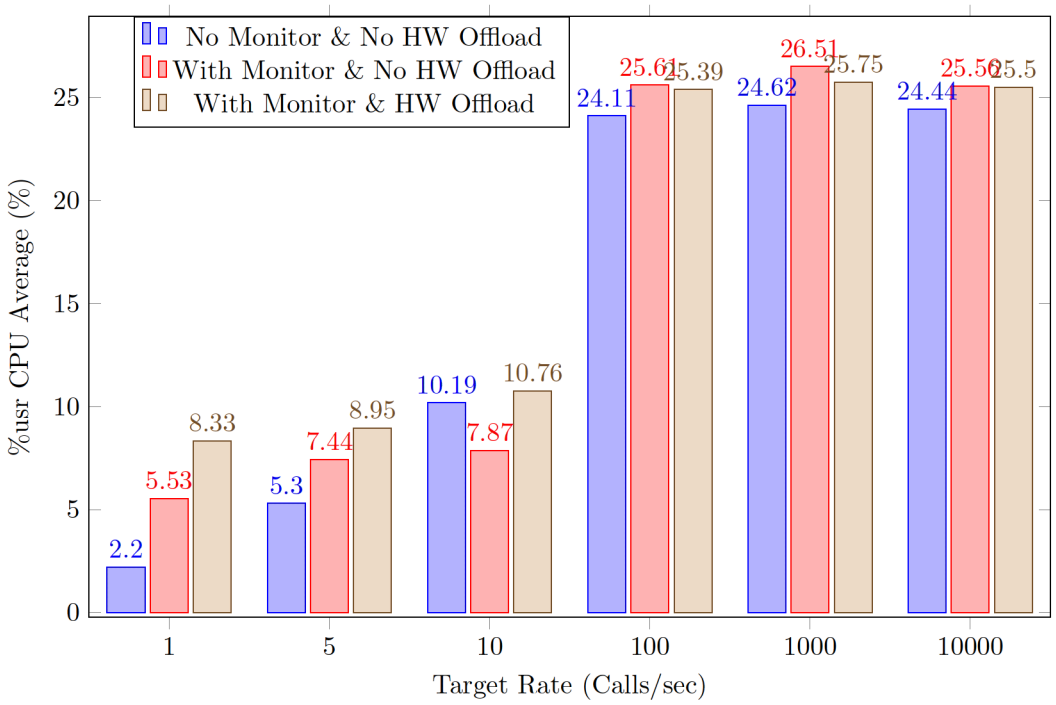
Με τη δημιουργία του σεναρίου /sipp/docs/register5.xml (βλ. Παράρτημα Κώδικα) καταφέραμε να υλοποιήσουμε μια αυτοματοποιημένη ανταλλαγή μηνυμάτων εγγραφής μεταξύ πελάτη-διακομιστή (βλ. Σχήμα 5.1). Αρχικά ορίσαμε ένα μεταβλητό ρυθμό κλήσης με λογαριθμική σχέση, δοκιμάζοντας έτσι με λογικές και ακραίες τιμές τη συμπεριφορά του συστήματος. Παρακάτω παρουσιάζονται τα αποτελέσματα σε μορφή πίνακα καθώς επίσης και σε μορφή γραφημάτων.

HW-Offload	Monitor	Target Rate (calls/sec)	CallRate (calls/sec)	%usr CPU Average (%)	%sys CPU Peak (%)	%sys CPU Average (%)	Latency Average (msec)
No	No	1	0.9	2.20	0.50	0.32	75
No	Yes	1	0.9	5.53	0.50	0.27	60
Yes	Yes	1	0.9	8.33	0.75	0.33	84
No	No	5	4.9	5.30	0.99	0.39	59
No	Yes	5	4.9	7.44	0.76	0.45	80
Yes	Yes	5	4.9	8.95	0.76	0.38	58
No	No	10	9.86547	10.19	2.70	0.94	65
No	Yes	10	9.88024	7.87	1.02	0.44	55
Yes	Yes	10	9.88715	10.76	1.01	0.42	57
No	No	100	19.8991	24.11	**5.49**	**3.92**	5951
No	Yes	100	25.5582	25.61	1.26	0.80	5162
Yes	Yes	100	23.9597	25.39	1.01	0.66	7117
No	No	1000	76.5813	24.62	**4.85**	**3.73**	15804
No	Yes	1000	77.7761	26.51	1.02	0.69	15464
Yes	Yes	1000	76.7238	25.75	1.52	0.89	14354
No	No	10000	745.342	24.44	**5.57**	**3.78**	7605
No	Yes	10000	749.139	25.56	1.77	0.98	6957
Yes	Yes	10000	742651	25.50	1.50	0.81	13988

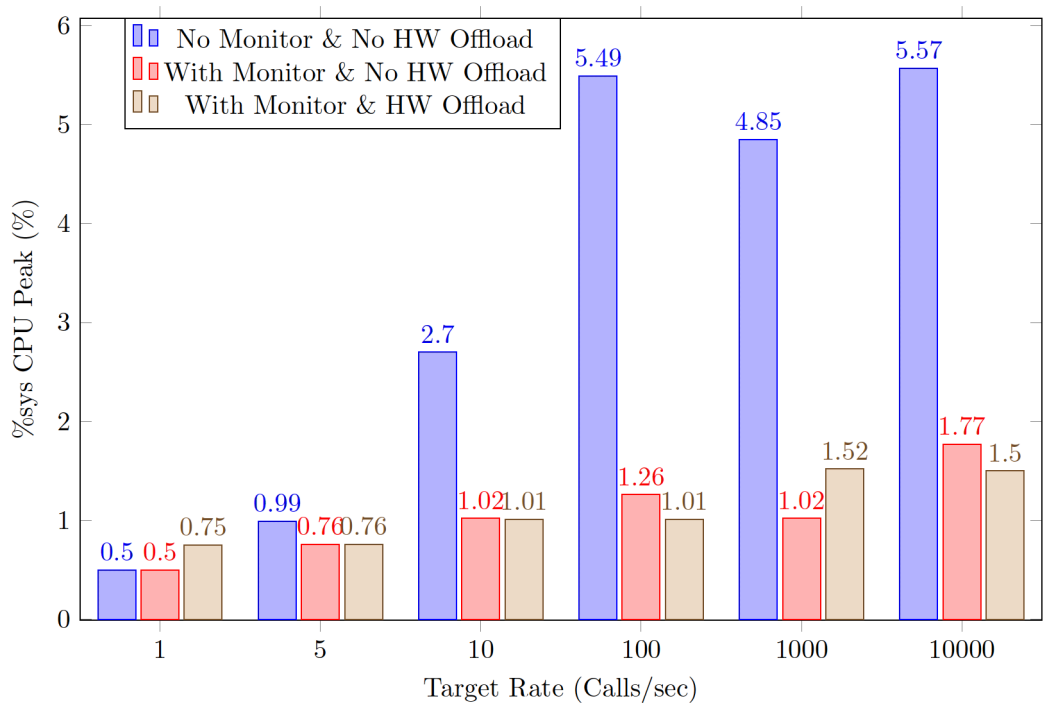
Πίνακας 5.1: Πίνακας αποτελεσμάτων επίδοσης συστήματος



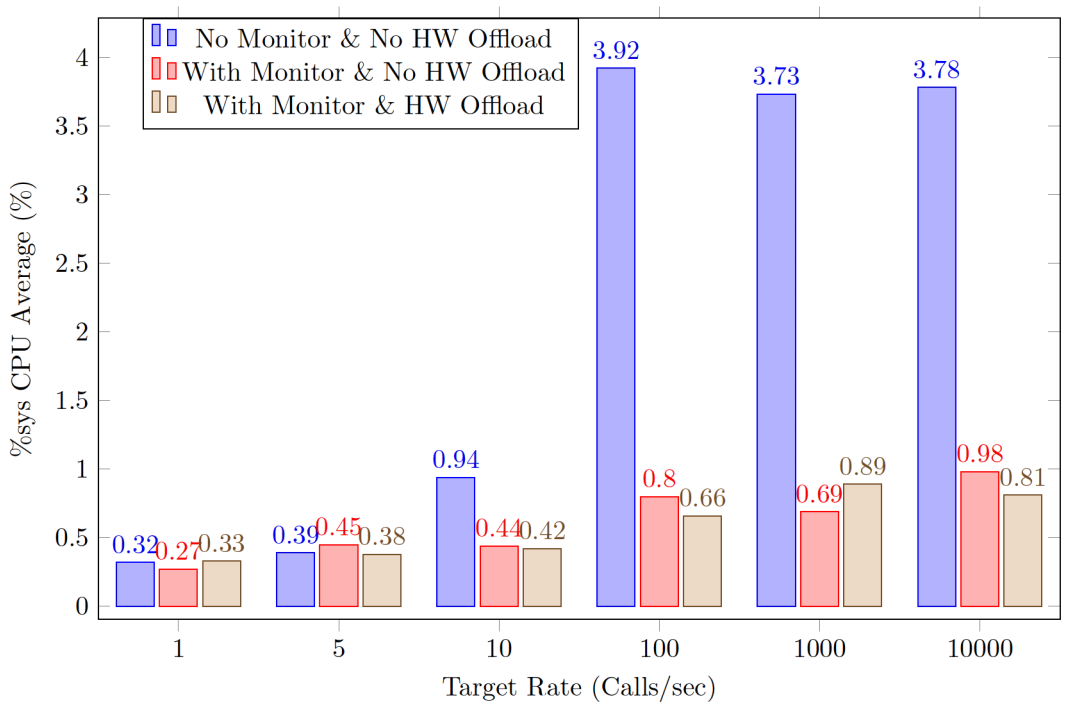
Σχήμα 5.2: Ρυθμός κλήσης (calls/sec) σε λογαριθμική κλίμακα



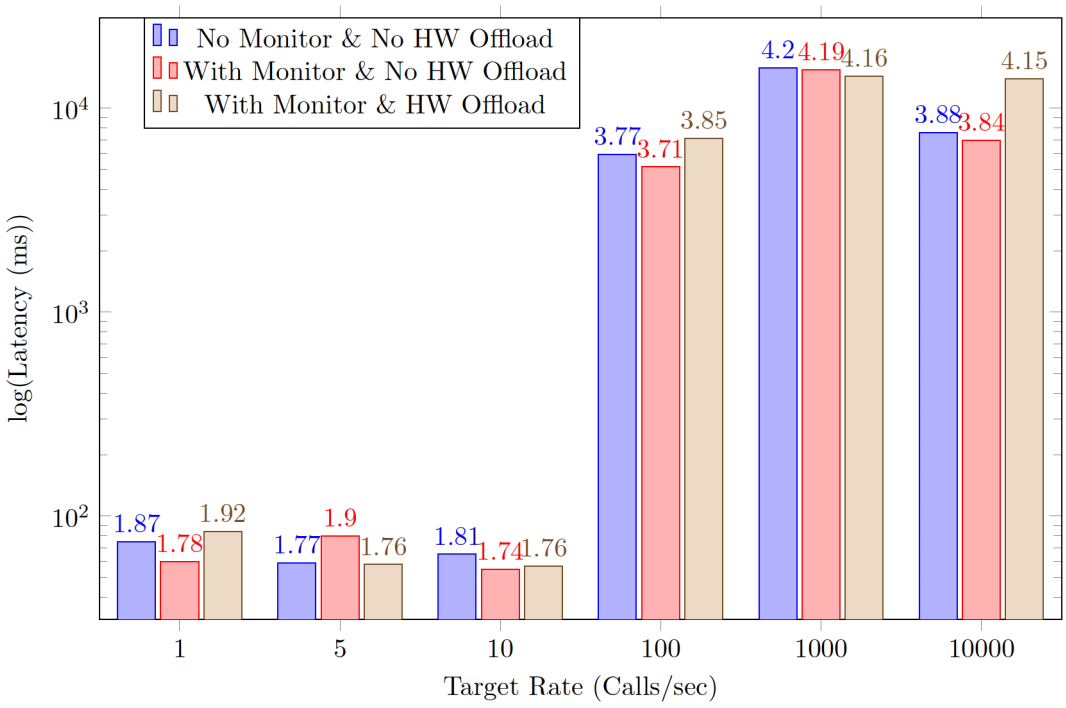
Σχήμα 5.3: Μέση τιμή κατανάλωσης ισχύος επεξεργαστή στο περιβάλλον χρήστη (%)



Σχήμα 5.4: Ακραία τιμή κατανάλωσης ισχύος επεξεργαστή στο σύστημα (%)



Σχήμα 5.5: Μέση τιμή κατανάλωσης ισχύος επεξεργαστή στο σύστημα (%)



Σχήμα 5.6: Καθυστέρηση μετάδοσης πακέτων εγγραφής (msec)

Αξίζει να αναφερθούμε κυρίως στα Σχήματα 5.4 και 5.5, όπου φαίνεται από ένα σημείο και μετά η βελτίωση της συνολικής απόδοσης σχετικά με την κατανάλωση πόρων επεξεργαστή

από το σύστημα. Όπως αναμέναμε και από τη θεωρία, για μεγάλους ρυθμούς κλήσεων, στο σύστημα μειώνεται δραματικά το ποσοστό απασχόλησης του επεξεργαστή τόσο κατ΄απόλυτη τιμή όσο και αθροιστικά.

Συνολικά, συμπεραίνουμε ότι τόσο η προσθήκη του Monitor όσο και η ενεργοποίηση του HW-Offload προσφέρει στο σύστημα μια σημαντική βελτίωση στην επίδοση διατηρώντας ταυτόχρονα μια αναλλοίωτη υπηρεσία όσον αφορά τον ρυθμό κλήσεων (βλ. Σχήμα 5.2) και την καθυστέρηση μετάδοσης (βλ. Σχήμα 5.6).

Κεφάλαιο 6

Επίλογος

6.1 Σύνοψη

Στόχος της παρούσας διπλωματικής εργασίας ήταν η υλοποίηση ενός συστήματος παρακολούθησης IP πακέτων και συγκεκριμένα πακέτων σηματοδοσίας SIP, με χρήση τεχνολογιών προγραμματισμού του επιπέδου δεδομένων και επιτάχυνσης της δικτυακής κίνησης. Η υλοποίηση βασίζεται σε εικονικές μηχανές για τη διάδοση της δικτυακής κίνησης από και προς τον εξυπηρετητή SIP. Υπάρχει επίσης και η φυσική συσκευή του Raspberry Pi που αναπαριστά τον πελάτη SIP και εκτελεί τις απαιτούμενες ενέργειες εγγραφής και κλήσης χρήστη. Για την επιτυχή υλοποίηση χρησιμοποιήθηκαν τεχνολογίες SDN, Εικονικοποίησης δικτύων και Επιτάχυνσης δικτυακής κίνησης με χρήση της έξυπνης κάρτας δικτύου Agilio SmartNIC. Με τη χρήση αυτής της τεχνολογίας επιτυγχάνεται αφενός η καλύτερη διαχείριση της κίνησης σε οποιοδήποτε δίκτυο (είτε ιδιώτη είτε επιχείρησης) μέσω προσθήκης ή αφαίρεσης ροών, με τη χρήση του ελεγκτή SDN, και αφετέρου η βέλτιστη απόδοση δικτύου σε επίπεδο κέντρων δεδομένων, τόσο ενεργειακή, επιφέροντας μειωμένο κόστος κατανάλωσης ισχύος, όσο και χωροταξική χάρη στο NFV.

Αφού παρουσιάστηκε αναλυτικά η αρχιτεκτονική και ο σχεδιασμός της τοπολογίας του δικτύου καθώς και τα απαραίτητα εργαλεία ελεύθερου λογισμικού, στη συνέχεια πραγματοποιήθηκαν διάφορες δοκιμές μετάδοσης SIP μηνυμάτων, συμπεραίνοντας την ορθή λειτουργία της υλοποίησης.

6.2 Συμπεράσματα & Μελλοντικές Επεκτάσεις

Όπως αποδείχθηκε η ανακατεύθυνση της κίνησης μέσω SDN τεχνολογίας μπορεί να υλοποιηθεί με ευκολία, ανοίγοντας νέα μονοπάτια βελτίωσης στο δίκτυο, όσον αφορά το κόστος δικτυακού εξοπλισμού αλλά και την ευελιξία που παρέχει η προγραμματίσιμη και κεντροποιημένη διαχείριση του δικτύου. Συγκεκριμένα το SDN παρέχει μεγαλύτερο έλεγχο στην παρακολούθηση της κίνησης, καθώς επίσης και τη δυνατότητα άμεσης αλλαγής ροής των πακέτων. Αυτό είναι πολύ σημαντικό στα δίκτυα νέας γενιάς 5G, όπου απαιτείται η δυναμική ανάθεση εύρους ζώνης, ανάλογα με την εφαρμογή, με τελικό στόχο το QoS

(Quality of Service), δηλαδή την τέλεια εμπειρία χρήστη. Σε συνδυασμό με την προσθήκη ενός διαμεσολαβητή SIP, όπως το Kamailio, μπορεί κανείς να αποκτήσει έλεγχο της κίνησης SIP σε ένα δίκτυο, εκτελώντας λειτουργίες προώθησης, απόρριψης κλήσεων, ακόμα και εγγραφή χρηστών σε μία βάση δεδομένων. Η επανάσταση, εν προκειμένω, θα είναι η δυνατότητα έξυπνης διαχείρισης κλήσεων σε περιπτώσεις έκτακτης ανάγκης, όπως η αυτόματη κλήση ασθενοφόρου μέσω εφαρμογών IoT, η παραχώρηση προτεραιότητας κίνησης, η εναλλακτική δρομολόγηση κλήσεων σε περίπτωση κατάρρευσης γραμμής κά.

Στην παρούσα διπλωματική η δυνατότητα εκφόρτωσης της SIP κίνησης μας παρέχει άλλη μια πληροφορία σχετικά με την κίνηση πακέτων, ενώ ταυτόχρονα επιταχύνει τη ροή πακέτων επιτρέποντας στο host μηχάνημα να εκτελεί ανενόχλητο οποιαδήποτε άλλη εργασία. Η εξέλιξη των καρτών δικτύων, την τελευταία δεκαετία, έχει συμβάλει, τόσο στη βελτιστοποίηση και επιτάχυνση της δικτυακής κίνησης, όσο και στην ασφάλεια δικτύου, επιτρέποντας την διαφανή εκφόρτωση-απόρριψη ύποπτων ή/και αδιάφορων πακέτων. Πλέον υπάρχει η δυνατότητα προσθήκης νέων χαρακτηριστικών κρυπτογράφησης απευθείας στο υλικό ή ακόμα αυτοματοποίησης ελέγχου από πιθανές επιθέσεις DDoS. Η λογική που επιτελεί η έξυπνη κάρτα δικτύου μοιάζει με τη λογική της κάρτας γραφικών που κάλυψε την ανάγκη επεξεργασίας όλων των σχετικών διεργασιών, απελευθερώνοντας χρήσιμους, για τον επεξεργαστή, πόρους.

Τέλος αξίζει να αναφέρουμε τον προγραμματισμό P4 ενός δικτυακού chip καθώς μπορεί να επιφέρει την απόλυτη εξουσία και διαχείριση οποιουδήποτε δικτυακού εξοπλισμού το επιτρέπει. Συγκεκριμένα το προγραμματίσιμο επίπεδο δεδομένων προσφέρει τη δυνατότητα εύκολης προσθήκης νέων χαρακτηριστικών ή την υποστήριξη οποιουδήποτε υπάρχοντος πρωτοκόλλου (πχ. BGP, OSPF, Spanning Tree κά.). Επίσης δίνεται η δυνατότητα αφαίρεσης οποιουδήποτε πρωτοκόλλου δεν χρησιμοποιείται και τέλος παρέχει απόλυτη παρακολούθηση πακέτων χρησιμοποιώντας μετα-δεδομένα, δηλαδή την προσθήκη ετικετών σε πακέτα που μεταδίδονται στο δίκτυο, επιτρέποντας με αυτόν τον τρόπο τη βέλτιστη ανάθεση δικτυακού latency από τον διαχειριστή δικτύου.

Είναι πλέον γεγονός ότι ζούμε στον κόσμο της πληροφορίας. Η χρήση διαδικτύου και τηλεφωνικών συσκευών αποτελεί ένα αναπόσπαστο κομμάτι της καθημερινότητας. Συνεπώς η ανάγκη για έλεγχο των δεδομένων γίνεται ολοένα και μεγαλύτερη, εάν μάλιστα λάβουμε υπόψιν μας και τους κινδύνους που ελλοχεύουν στον κυβερνοχώρο, είναι χρέος μας να επενδύσουμε σε αυτές τις καινοτόμες τεχνολογίες.

Βιβλιογραφία

[1] Miikka Poikselkä, Georg Mayer. The IMS: IP Multimedia Concepts and Services, 3rd Edition

[2] Rashid Mijumbi,Joan Serrat,Juanluis Gorricho, Niels Bouten. Network Function Virtualization: State-of-the-Art and Research Challenges
https://www.researchgate.net/publication/281524200_Network_Function_Virtualization_State-of-the-Art_and_Research_Challenges",
2015. Last accessed on September 2020 .

[3] Global data center IP traffic from 2013 to 2021
https://www.statista.com/statistics/227246/global-data-center-ip-traffic-development-forecast/
Last accessed on September 2020.

[4] Cisco Annual Internet Report (2018–2023) White Paper.
https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html
Last accessed on September 2020.

[5] RedHat, What is NFV
https://www.redhat.com/en/topics/virtualization/what-is-nfv
Last accessed on September 2020.

[6] Researchgate.net, Traditional-vs-NFV-approach-comporation
www.researchgate.net/figure/Traditional-vs-NFV-approach-comporation_fig1_325132982
Last accessed on July 2021.

[7] What is software-defined networking?
https://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html
Last accessed on September 2020.

[8] VoIP support with Software-Defined Networking (SDN) using OpenDaylight Controller
https://www.academia.edu/34322672/VoIP_support_with_Software_Defined_Networking_SDN_using_OpenDaylight_Controller
Last accessed on September 2020.

[9] What Is Open vSwitch?
https://docs.openvswitch.org/en/latest/intro/what-is-ovs/
Last accessed on September 2020

[10] Open vSwitch From Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/Open_vSwitch
Last accessed on September 2020

[11] Netronome 25GbE SmartNICs with Open vSwitch Hardware Offload Drive Unmatched Cloud and Data Center Infrastructure Performance
https://www.netronome.com/m/documents/WP_Netronome_25GbE_SmartNICs_with_Open_vSwitch_Hardware_Offload.pdf
Last accessed on September 2020.

[12] Agilio® OVS Software Architecture FOR SERVER-BASED NETWORKING
https://www.netronome.com/m/documents/WP_Agilio_SW.pdf
Last accessed on September 2020.

[13] Netronome NFP-4000 Flow Processor
https://www.netronome.com/m/documents/PB_NFP-4000-7-20.pdf
Last accessed on September 2020.

[14] VMWare What is Micro-Segmentation?
https://www.vmware.com/topics/glossary/content/micro-segmentation
Last accessed on September 2020.

[15] IP Multimedia Subsystem (IMS) Wikipedia
https://en.wikipedia.org/wiki/IP_Multimedia_Subsystem
Last accessed on October 2020.

[16] Raspberry Pi 3 Model B
https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
Last accessed on September 2020.

[17] Raspberry Pi 3 Model B Interfaces
https://en.wikipedia.org/wiki/Raspberry_Pi#/media/File:RaspberryPi_3B.svg
Last accessed on September 2020.

[18] Agilio® OVS Software OFFLOAD AND ACCELERATE SERVER-BASED NETWORKING
https://www.netronome.com/m/documents/PB_Agilio_OVS_SW-7-20.pdf
Last accessed on September 2020.

[19] COMPARISON OF SOFTWARE DEFINED NETWORKING (SDN) CONTROLLERS. PART 3: OPENDAYLIGHT (ODL)
aptira.com/comparison-of-software-defined-networking-\sdn-controllers-part-3-opendaylight-odl/
Last accessed on September 2020.

[20] What Is OpenFlow? Definition and How it Relates to SDN
https://www.sdxcentral.com/networking/sdn/definitions/what-is-openflow/
Last accessed on September 2020

[21] OpenFlow Switch: What Is It and How Does it Work?
http://www.cables-solutions.com/whats-openflow-switch-how-it-works.html
Last accessed on September 2020

[22] Github Project ClearWater IMS
https://github.com/Metaswitch/clearwater-website-archive
Last accessed on September 2020.

[23] Kamailio Features
https://www.kamailio.org/w/features/
Last accessed on September 2020

[24] About Kamailio (OpenSER) SIP Server project
https://www.voip-info.org/kamailio/
Last accessed on September 2020

[25] Kamailio From Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/Kamailio
Last accessed on September 2020

[26] QEMU From Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/QEMU
Last accessed on September 2020

[27] Kernel Virtual Machine
https://www.linux-kvm.org/page/Main_Page
Last accessed on September 2020

[28] Libvirt FAQ What is libvirt?
https://wiki.libvirt.org/page/FAQ#What_is_libvirt.3F
Last accessed on September 2020

[29] KVM Figure
https://upload.wikimedia.org/wikipedia/commons/thumb/4/40/Kernel-based_Virtual_Machine.svg/400px-Kernel-based_Virtual_Machine.svg.png
Last accessed on September 2020

[30] PJSUA API - High Level Softphone API
https://www.pjsip.org/pjsip/docs/html/group__PJSUA__LIB.htm
Last accessed on September 2020

[31] About Scapy
https://scapy.readthedocs.io/en/latest/introduction.html#about-scapy
Last accessed on September 2020

Κατάλογος Ακρωνύμων

ADSL Asymmetric Digital Subscriber Line

API Application Programming Interface

BGCF Breakout Gateway Control Function

BGP Border Gateway Protocol

BLE Bluetooth Low Energy

BSD Berkeley Software Distribution

CAM Computer-aided manufacturing

CAPEX Capital expenditure

COTS Commercial off-the-shelf

CPU Central processing unit

CSCF Call Session Control Function

DDoS Distributed Denial-of-Service

DMA Direct memory access

DPDK Data Plane Development Kit

DPI Deep Packet Inspection

ECC Error-correcting code

ETSI European Telecommunications Standards Institute

GB Gigabyte

GbE Gigabit Ethernet

GNU General Public License

GPIO General-purpose input/output

GRE Generic Routing Encapsulation

HSS Home Subscriber Server

IaaS Infrastructure as a service

I-CSCF (Interrogating Call Session Control Function

IEEE Institute of Electrical and Electronics Engineers

IMS IP Multimedia Subsystem

IoT Internet of Things

IP Internet Protocol

ISDN Integrated Services Digital Network

IT Information Technology

KVM Kernel-based Virtual Machine

LACP Link Aggregation Control Protocol

LTE Long-Term Evolution

MAC Media Access Control

MANO Management and orchestration

MB MegaByte

MPLS Multiprotocol Label Switching

NAT Network address translation

NETCONF Network Configuration Protocol

NFV Network functions virtualization

NFVi Network functions virtualization infrastructure

NIC Network interface controller

NVGRE Network Virtualization using Generic Routing Encapsulation

ODL OpenDayLight

OPEX Operating expense

OSGi Open Services Gateway Initiative

OVS Open Virtual Switch

OVSDB Open vSwitch Database Management Protocol

PBX Private Branch Exchange

PCIe Peripheral component interconnect express

P-CSCF Proxy Call Session Control Function

PF Physical Function

PKI Public Key Infrastructure

PSTN Public switched telephone network

QoS Quality of service

RADIUS Remote Authentication Dial-in User Service

REST Representational state transfer

RFC Request for Comments

RHEL Red Hat Enterprise Linux

ROI Return on investment

Rpi Raspberry Pi

RTP Real-time Transport Protocol

SaaS Software as a service

SAL Service Abstraction Layer

SBC Session Border Controllers

S-CSCF Serving Call Session Control Function

SCTP Stream Control Transmission Protocol

SDN Software-defined networking

SerDes Serializer/Deserializer

SFP small form-factor pluggable

SIP Session Initiation Protocol

SQL Structured Query Language

SR-IOV Single-root input/output virtualization

SRTP Secure Real-time Transport Protocol

TC Traffic control

TCP Transmission Control Protocol

TLS Transport Layer Security

UA User Agent

UART Universal asynchronous receiver-transmitter

UDP User Datagram Protocol

USB Universal Serial Bus

VF Virtual Function

VLAN Virtual LAN

VM Virtual Machine

VNF Virtual Network Function

VoIP Voice over Internet Protocol

VoLTE Voice over Long-Term Evolution

VXLAN Virtual Extensible LAN

WebRTC Web Real-Time Communication

XDMS Extensive Document Management System

Παράρτημα Κώδικα

- flow_3.xml

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <priority>100</priority>
    <flow-name> SEND ALL RPi TRAFFIC TO GATEWAY </flow-name>
    <match>
        <ethernet-match>
            <ethernet-type>
                <type>2048</type>
            </ethernet-type>
            <ethernet-destination>
                <address>b8:27:eb:5d:1f:e1</address>
            </ethernet-destination>
            <ethernet-source>
                <address>00:1c:f6:b1:56:59 </address>
            </ethernet-source>
        </ethernet-match>
    </match>

    <id>3</id>
    <table_id>0</table_id>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                    <order>0</order>
                    <output-action>
                        <output-node-connector>5</output-node-connector>
                        <max-length>60</max-length>
                    </output-action>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
</flow>
```

- register5.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="register_client">
<send retrans="1000" start_rtd="register_latency">
<![CDATA[ REGISTER sip:147.102.7.18:5060 SIP/2.0
Via: SIP/2.0/UDP 147.102.39.10:5060;rport;branch=[branch]
Route: <sip:147.102.7.18;lr>
From: <sip:2100000037@telecom.ntua.gr>;tag=[call_number]
To: <sip:2100000037@telecom.ntua.gr>
Call-ID: [call_id]
CSeq: [cseq] REGISTER
Contact: <sip:2100000037@147.102.39.10:5060;ob>;transport=UDP
Max-Forwards: 70
Expires: 300
User-Agent: SIPp/Win32
Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, INFO, SUBSCRIBE, NOTIFY, REFER,
     MESSAGE, OPTIONS
Content-Length: 0 ]]>
</send>
<!--  asterisk  -->
<recv response="100" optional="true"> </recv>
<recv response="401" auth="true" rtd="register_latency"> </recv>
<send retrans="1000">
<![CDATA[ REGISTER sip:147.102.7.18:5060 SIP/2.0
Via: SIP/2.0/UDP 147.102.39.10:5060;rport;branch=[branch]
Route: <sip:147.102.7.18;lr>
From: <sip:2100000037@telecom.ntua.gr>;tag=[call_number]
To: <sip:2100000037@telecom.ntua.gr>
Call-ID: [call_id]
CSeq: [cseq] REGISTER
Contact: <sip:2100000037@147.102.39.10:5060;ob>;transport=UDP
Max-Forwards: 70
Expires: 300
Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, INFO, SUBSCRIBE, NOTIFY, REFER,
     MESSAGE, OPTIONS
[authentication username=2100000037@telecom.ntua.gr password=wWcYx5QeX]
User-Agent: SIPp/Win32
Content-Length: 0 ]]>
</send>
<!--  asterisk  -->
<recv response="100" optional="true"> </recv>
<recv response="200"> </recv>
<!--  response time repartition table (ms)     -->
<ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>
<!--  call length repartition table (ms)       -->
<CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>
</scenario>
```