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

# Εφαρμογή Parental Control με τεχνικές SDN/NFV και Service Function Chaining

Μπουρδούβαλης Βασίλειος

**Επιβλέπων καθηγητής:** Συκάς Ευστάθιος

Αθήνα,
Ιανουάριος 2019

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

# Εφαρμογή Parental Control με τεχνικές SDN/NFV και Service Function Chaining

Διπλωματική εργασία

Μπουρδούβαλη Βασίλειου

**Επιβλέπων καθηγητής:** Συκάς Ευστάθιος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ... Ιανουαρίου 2019.

| | | |
|---|---|---|
| ……. | ……. | ……. |
| Συκάς Ευστάθιος | Στασινόπουλος Γεώργιος | Ρουσσάκη Ιωάννα |
| Καθηγητής Ε.Μ.Π. | Καθηγητής Ε.Μ.Π. | Επίκουρος Καθηγήτρια Ε.Μ.Π. |

Αθήνα,
Ιανουάριος 2019

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

# Περίληψη

Με τον όρο NFV (Network Function Virtualization) ονομάζεται η νέα δικτυακή αρχιτεκτονική όπου οι δικτυακοί κόμβοι για την παροχή υπηρεσιών «εικονοποιούνται» και εκτελούνται σε εικονικές μηχανές αντί για παραδοσιακούς φυσικούς κόμβους. Για την καινούργια αρχιτεκτονική είναι απαραίτητος ο διαχωρισμός του λογισμικού από το υλικό και η εκτέλεση του λογισμικού σε εξυπηρετητές γενικής χρήσης (COTS - Common of the self). Η χρήση VNF από την πλευρά τον κόμβων είναι συνδεδεμένη με τη χρήση SDN στο επίπεδο δικτύου.

Το SDN (Software Defined Networking) είναι επίσης μία νέα αρχιτεκτονική που διαχωρίζει το στρώμα προώθησης δεδομένων από το στρώμα διαχείρισης. Το στρώμα διαχείρισης μπορεί απευθείας να προγραμματιστεί, ενώ το φυσικό δίκτυο που είναι κοινό, να μπορεί να αντιληφθεί τις διάφορες υπηρεσίες. Βασικό συστατικό της αρχιτεκτονικής είναι ο ελεγκτής (controller) που συντηρεί τον πίνακα δρομολόγησης των διαφορετικών ροών όπως ενημερώνονται από το επίπεδο ελέγχου. Ο ελεγκτής είναι ένα κεντρικοποιημένο σύστημα διαχείρισης της πληροφορίας δρομολόγησης.

Η παροχή υπηρεσιών περιεχομένου όπως η προστασία περιεχομένου μπορεί να επιτευχθεί με τη χρήση τεχνολογιών NFV/SDN με δυναμικό τρόπο. Η δημιουργία και διαχείριση τέτοιων υπηρεσιών υλοποιείται με τη χρήση «αλυσίδων» πιο απλών υπηρεσιών υλοποιημένων σε VNFs (Service Chaining)

Σκοπός της εργασίας είναι η δημιουργία αλυσίδων υπηρεσιών για την ανάπτυξη εφαρμογής σε επίπεδο δικτύου για την προστασία από ακατάλληλο περιεχόμενο κατά την πλοήγηση σε ιστοσελίδες με δύο τρόπους. Η πρώτη υλοποίηση πραγματοποιείται μια αλυσίδα υπηρεσιών όπου ο χρήστης συνδέεται μέσω ενός vCPE στον εξοπλισμό που την υλοποιεί και χρησιμοποιούνται τεχνικές δικτύων επικάλυψης(overlay networks) και εικονικών μεταγωγέων. Η δεύτερη με την χρήση τεχνικών Αλυσιδωτής Λειτουργίας Υπηρεσίας (Service Function Chaining – SFC) που παρουσιάζεται ο τρόπος που μπορεί να γίνει με την χρήση διαφόρων προεκτάσεων, όπου δημιουργείται μια αλυσίδα υπηρεσίας ελεγχόμενη από έναν ελεγκτή SDN.

Λέξεις κλειδιά: SDN, NFV, Opendaylight Controller, VXLAN, overlay networks, Openflow, Openvswitch, NSH, flows

# Abstract

NFV (Network Functions Virtualization) is a new network architecture where network nodes for service delivery are "rendered" and run on virtual machines instead of traditional physical nodes. For this new architecture, it is necessary to separate the software from the hardware and run the software on COTS (Common of the Self) servers. The use of VNF on the side of the nodes is linked to the use of SDN at the network level.

SDN (Software Defined Networking) is also a new architecture that separates the data layer from the management layer. The management layer can be directly programmed, while the physical network that is common, can understand the various services. The basic component of the architecture is the controller that maintains the routing table of the different flows as updated by the control level. The controller is a centralized routing information management system.

The provision of content services such as content protection can be achieved through the use of NFV / SDN technologies in a dynamic way. The creation and management of such services is realized through the use of "chains" of more simple services implemented in VNFs (Service Chaining)

The purpose of this thesis, is to create service chains to deploy a network-level application to protect against inappropriate content when navigating to web pages in two ways. The first implementation is a service chain where the user is connected via a vCPE to the equipment that implements the chain with the use of overlay networks and virtual switches. In the second, using Service Function Chaining (SFC) techniques, it is presented how to use various extensions to create a service chain controlled by an SDN controller.

Keywords: SDN, NFV, Opendaylight Controller, VXLAN, overlay networks, Openflow, Openvswitch, NSH, flows

# Ευχαριστίες

Η εκπόνηση της παρούσας διπλωματικής εργασίας σηματοδοτεί και το τέλος των σπουδών μου στην Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών. Θα ήθελα σε αυτό το σημείο να ευχαριστήσω ειλικρινά τους ανθρώπους που βοήθησαν στο τελευταίο στάδιο αυτής της πορείας στα «θρανία» της σχολής και όχι μόνο.

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή κ. Ευστάθιο Συκά για την ανάθεση της διπλωματικής εργασίας και τον υποψήφιο διδάκτορα κ. Πάρι Χαραλάμπου για την καλή συνεργασία, την βοήθεια του, την υπομονή του και την ευελιξία που μου με άφησε να έχω κατά την εκπόνηση της εργασίας.

Έπειτα θα ήθελα να ευχαριστήσω τους συμφοιτητές μου και σε όλους τους ανθρώπους που γνώρισα στα φοιτητικά μου χρόνια, και τα κάναν πιο όμορφα και ενδιαφέροντα.

Στην συνέχεια θα ήθελα να ευχαριστήσω τους φίλους μου, που σ’ αυτά τα όμορφα αλλά και κατά καιρούς δύσκολα φοιτητικά μου χρόνια στάθηκαν στο πλάι μου.

Ένα μεγάλο ευχαριστώ οφείλω και στα ανήσυχα παιδιά μπροστά από τους κόκκινους τοίχους της σχολής, που μου έδωσαν το μεγαλύτερο μάθημα.

Τέλος, το μεγαλύτερο ευχαριστώ το οφείλω στην οικογένεια μου και ιδιαίτερα στο πατέρα μου, για την υπομονή του και την στήριξη του όλα αυτά τα χρόνια που ήταν πάντα στο πλευρό μου, και στην μητέρα μου που μου έδωσε όλα τα εφόδια για να μπορέσω να φτάσω μέχρι εδώ. Η εργασία αφιερώνεται αποκλειστικά στην μνήμη της.

Μπουρδούβαλης Βασίλης
Δεκέμβρης 2018

# Περιεχόμενα

# Κατάλογος Εικόνων

# Κατάλογος Πινάκων

## Εισαγωγή

Στην παρούσα διπλωματική εργασία εξετάζεται η δημιουργία αλυσίδων υπηρεσιών για την ανάπτυξη εφαρμογής σε επίπεδο δικτύου, για την προστασία από ακατάλληλο περιεχόμενο κατά την πλοήγηση σε ιστοσελίδες με δύο τρόπους. Στην πρώτη υλοποίηση, πραγματοποιείται μια αλυσίδα υπηρεσιών όπου ο χρήστης συνδέεται μέσω ενός vCPE στον εξοπλισμό που την υλοποιεί και χρησιμοποιούνται τεχνικές δικτύων επικάλυψης(overlay networks) και εικονικών μεταγωγέων. Στη δεύτερη, με τη χρήση τεχνικών Αλυσιδωτής Λειτουργίας Υπηρεσίας (Service Function Chaining – SFC) παρουσιάζεται ο τρόπος που μπορεί να γίνει με τη χρήση διαφόρων προεκτάσεων, όπου δημιουργείται μια αλυσίδα υπηρεσίας ελεγχόμενη απομακρυσμένα από έναν ελεγκτή SDN. Η ανάπτυξη τους έγινε σε ένα πλαίσιο εικονικοποίησης δικτυακών λειτουργιών (Network Function Virtualization) και με την αξιοποίηση της τεχνολογίας των Δικτύων ορισμένων από Λογισμικό (Software-Defined Networks – SDN).

Στο 1ο κεφάλαιο αναλύονται οι έννοιες της Εικονικοποίησης και πιο συγκεκριμένα της Εικονικοποίσης Δικτυακών Λειτουργιών (NFV), γίνεται μια αναγκαία εισαγωγή στα Δίκτυα ορισμένα από λογισμικό (SDN) και μια αναφορά στην σχέση μεταξύ SDN και NFV

Στο 2ο κεφάλαιο αναλύονται τα βασικά πρωτόκολλα για δίκτυα overlay για να γίνει η κατάλληλη επιλογή για την εργασία μας.

Στο 3ο κεφάλαιο γίνεται μια αναγκαία περιγραφή των μεταγωγέων Openflow, και ιδιαίτερα στον εικονικό μεταγωγέα OVS και την υποστήριξη του σε δίκτυα overlay.

Στο 4ο κεφάλαιο αναλύεται πλήρως η έννοια, η αρχιτεκτονική και τα δομικά στοιχεία της Αλυσιδωτής Λειτουργίας Υπηρεσίας (SFC)

Στο 5ο κεφάλαιο παραθέτονται οι αρχιτεκτονικές των συστημάτων που θα αναπτυχθούν και στο 6ο κεφάλαιο τα δομικά στοιχεία του συστήματος όσον αφορά τον εξοπλισμό που χρησιμοποιήθηκε.

Στο 7ο κεφάλαιο ενσωματώνονται όλες οι τεχνολογίες και παραθέτονται όλες οι ρυθμίσεις και τα αποτελέσματα των υλοποιήσεων

Τέλος στο 8ο κεφάλαιο προτείνονται κάποιες μελλοντικές επεκτάσεις-προτάσεις για την παρούσα εργασία αλλά και τις τεχνικές που χρησιμοποιήθηκαν γενικότερα, ενώ στο 9ο κεφάλαιο καταλήγουμε σε κάποια γενικά συμπεράσματα.

# Κεφάλαιο 1ο

# Εικονικοποίηση Δικτυακών Λειτουργιών (NFV) και Δίκτυα οριζόμενα από λογισμικό (SDN)

## 1.1) Εικονικοποίηση (Virtualization)

Η εικονικοποίηση είναι ένας ευρύς όρος των υπολογιστικών συστημάτων που αναφέρεται σε έναν μηχανισμό αφαίρεσης, που έχει σκοπό την απόκρυψη λεπτομερειών της υλοποίησης και της κατάστασης ορισμένων υπολογιστικών πόρων από πελάτες των πόρων αυτών (π.χ. εφαρμογές, άλλα συστήματα, χρήστες κλπ.). Η εν λόγω αφαίρεση μπορεί είτε να αναγκάζει έναν πόρο να συμπεριφέρεται ως σύνολο πόρων (π.χ. μία συσκευή αποθήκευσης σε διακομιστή τοπικού δικτύου), είτε πολλαπλούς πόρους να συμπεριφέρονται ως ένας (π.χ. συσκευές αποθήκευσης σε κατανεμημένα συστήματα). Η εικονικοποίηση δημιουργεί μία εξωτερική διασύνδεση η οποία αποκρύπτει την υποκείμενη υλοποίηση (π.χ. πολυπλέκοντας την πρόσβαση από διαφορετικούς χρήστες). Αυτή η προσέγγιση στην εικονικοποίηση αναφέρεται ως εικονικοποίηση πόρων. Μία άλλη προσέγγιση, ίδιας όμως νοοτροπίας, είναι η **εικονικοποίηση πλατφόρμας**, όπου η αφαίρεση που επιτελείται προσομοιώνει ολόκληρους υπολογιστές. Το αντίθετο της εικονικοποίησης είναι η διαφάνεια: ένας εικονικός πόρος είναι ορατός, αντιληπτός, αλλά στην πραγματικότητα ανύπαρκτος, ενώ ένας διαφανής πόρος είναι υπαρκτός αλλά αόρατος.[1]

### 1.1.1) Εικονικοποίηση πλατφόρμας

Στην παρούσα εργασία θα ασχοληθούμε κυρίως με την εικονικοποίηση πλατφόρμας. Στην εικονικοποίηση πλατφόρμας, ένα λογισμικό ελέγχου(επόπτης ή hypervisor) εκτελούμενο σε πραγματικό υλικό προσομοιώνει ένα υπολογιστικό περιβάλλον, μία εικονική μηχανή (Virtual Machine - VM), επάνω από το οποίο μπορεί να τρέξει κάποιο φιλοξενούμενο λογισμικό(συνήθως ένας πλήρης πυρήνας), απομονωμένο από το υπόλοιπο σύστημα. Ένα VM είναι ένα πλήρες εικονικό σύστημα, αποτελούμενο από εικονικό υλικό (hardware), και λειτουργεί κάτω από το λειτουργικό σύστημα ενός φυσικού υπολογιστή. Ένα VM συμπεριφέρεται ως φυσικό υπολογιστικό σύστημα: έχει τη δυνατότητα να φέρει σκληρούς δίσκους, κάρτα ήχου, επεξεργαστή με έναν ή περισσοτέρους πυρήνες, μνήμη RAM, κάρτες δικτύου, θύρες USB, κύκλωμα γραφικών, οπτικές μονάδες αποθήκευσης και BIOS. Μπορούν να υποστηρίξουν οποιοδήποτε λειτουργικό σύστημα και εφαρμογή επιθυμεί ο χρήστης.

Η θεμελιώδης λογική πίσω από την εικονικοποίηση πλατφόρμας είναι η αρχή πως οποιαδήποτε λειτουργία μπορεί να εκτελεστεί είτε από λογισμικό είτε από εξειδικευμένο υλικό. Οι μόνες διαφορές αφορούν την ευελιξία και την απόδοση. Είναι δυνατόν να προσομοιώνονται ταυτόχρονα πολλαπλές εικονικές μηχανές, εντελώς απομονωμένες μεταξύ τους, από το ίδιο λογισμικό ελέγχου. Υπάρχουν πολλά είδη εικονικοποίησης πλατφόρμας με σημαντικότερα τα:

**α) Εξομοίωση:** η εικονική μηχανή προσομοιώνει εξ' ολοκλήρου μία αρχιτεκτονική υλικού, πιθανώς διαφορετική από το πραγματικό υποκείμενο υλικό, επιτρέποντας έτσι να εκτελεστεί επάνω της ένα μη τροποποιημένο, φιλοξενούμενο Λειτουργικό Σύστημα σχεδιασμένο για τον εξομοιωμένο επεξεργαστή (π.χ. QEMU, που θα χρησιμοποιήσουμε στην υλοποίηση μας). Η εξομοίωση είναι διερμηνεία (interpration) σε χρόνο εκτέλεσης του κώδικα του φιλοξενούμενου Λειτουργικού Συστήματος, με έναν κύκλο ανάγνωσης-αποκωδικοποίησης-εκτέλεσης όπου κάθε εντολή που ανήκει στο σύνολο εντολών του επεξεργαστή-πηγή μεταφράζεται σε μία εντολή του συνόλου εντολών του επεξεργαστή-στόχου. Παράλληλα η εικονική μηχανή παρέχει μία αφαίρεση της μνήμης, των συσκευών Εισόδου/Εξόδου κλπ., φροντίζοντας ώστε κάθε μεταφρασμένη εντολή που απευθύνεται σε αυτά τα υποσυστήματα να τροποποιεί μόνο τις αφαιρέσεις/λογικές αναπαραστάσεις τους, οι οποίες κατευθύνονται και υλοποιούνται από το λογισμικό ελέγχου, και όχι το πραγματικό υλικό.

**β) Πλήρης**: η εικονική μηχανή προσομοιώνει επαρκές τμήμα του πραγματικού υλικού ώστε να επιτρέπει την εκτέλεση ενός μη τροποποιημένου, φιλοξενούμενου Λειτουργικού Συστήματος σχεδιασμένου για τον ίδιο τύπο επεξεργαστή με την πραγματική CPU Ενδεικτικά λογισμικά που χρησιμοποιούν τη συγκεκριμένη αρχιτεκτονική είναι το VirtualPC, VMware, Win4Lin κλπ. Στην πλήρη εικονικοποίηση δεν χρειάζεται εξομοίωση του συνόλου εντολών του επεξεργαστή και μάλιστα ένα τμήμα του κώδικα του φιλοξενούμενου Λειτουργικού Συστήματος μπορεί να εκτελείται απευθείας από το υλικό, χωρίς μεσολάβηση του επόπτη (hypervisor), αρκεί να μην επηρεάζει υποσυστήματα εκτός του άμεσου ελέγχου του τελευταίου. Τα κρίσιμα σημεία του φιλοξενούμενου κώδικα ωστόσο, όπως αυτά που προσπαθούν να αποκτήσουν πρόσβαση στο υλικό, συλλαμβάνονται από το λογισμικό ελέγχου και προσομοιώνονται, αφού τα αποτελέσματα κάθε λειτουργίας που επιτελείται σε μία εικονική μηχανή δεν επιτρέπεται να τροποποιούν την κατάσταση άλλων εικονικών μηχανών, του επόπτη ή του υλικού. Αν το πραγματικό υλικό βοηθά και επιταχύνει τη λειτουργία του λογισμικού ελέγχου, τότε η πλήρης εικονικοποίηση ονομάζεται εγγενής (native). Η βοήθεια αυτή αφορά κυρίως εύκολη διάκριση μεταξύ εντολών που μπορούν να εκτελεστούν απευθείας και εντολών που πρέπει να προσομοιωθούν από το λογισμικό.

**γ) Παραεικονικοποίηση**: η εικονική μηχανή δεν προσομοιώνει επακριβώς το υλικό αλλά παρέχει στις εικονικές μηχανές, μία προγραμματιστική διεπαφή ώστε να επιτρέπει την εκτέλεση επάνω της ενός τροποποιημένου, φιλοξενούμενου Λειτουργικού Συστήματος σχεδιασμένου για εκτέλεση από τον συγκεκριμένο hypervisor (π.χ. XEN). Η διεπαφή αυτή ονομάζεται διασύνδεση υπερκλήσεων και ένα λειτουργικό σύστημα πρέπει να μεταφερθεί ρητά σε έκδοση κατάλληλη για εκτέλεση από ένα σύστημα παραεικονικοποίησης, ώστε ο φιλοξενούμενος πυρήνας αντί να προσπελαύνει το υλικό άμεσα να εκτελεί υπερκλήσεις και να αναμένει απαντήσεις ή ασύγχρονες ειδοποιήσεις από τον επόπτη.

### 1.1.2) Τεχνική περιγραφή

Αναλόγως των τεχνικών χαρακτηριστικών του φυσικού εξυπηρετητή που φιλοξενεί τις εικονικές μηχανές, ενδέχεται να συνυπάρχουν περισσότερα του ενός VMs στο ίδιο φυσικό μηχάνημα. Οι εικονικές μηχανές αυτές λειτουργούν ταυτόχρονα κι επικοινωνούν μεταξύ τους, εν δυνάμει

μπορεί να συμμετέχουν σε ένα εικονικό, τοπικό δίκτυο. Βασικός περιοριστικός παράγοντας για την ταυτόχρονη λειτουργία δύο ή περισσοτέρων VMs στον ίδιο φυσικό υπολογιστή, είναι η επάρκεια συνολικής μνήμης RAM και πυρήνων του κεντρικού επεξεργαστή του φυσικού εξυπηρετητή.

Εναλλακτικά, τα VMs μπορούν να συμμετέχουν στο ίδιο το τοπικό δίκτυο που συμμετέχει κι ο αληθινός υπολογιστής. Σε αυτήν την περίπτωση, βρίσκονται στην ίδια τοπολογία με τον υπολογιστή που τα φιλοξενεί, πίσω από κατάλληλο API(προγραμματιστική διεπαφή), με αποτέλεσμα να εμφανίζονται ως υπολογιστές του οικιακού, τοπικού δικτύου. Σε ένα τέτοιο σενάριο μπορούν να μοιράζονται την ίδια (ενσύρματη ή ασύρματη) κάρτα δικτύου του αληθινού υπολογιστή ή μερικές να μοιράζονται μία κάρτα δικτύου και άλλες διαφορετικές κάρτες.

Τα φυσικά μηχανήματα στα οποίο γίνεται η εικονικοποίηση αποκαλούνται μηχανήματα οικοδεσπότες (host machines), ενώ τα εικονικά μηχανήματα αποκαλούνται μηχανήματα επισκέπτες (guest machines). Οι έννοιες host και guest χρησιμοποιούνται για να διαχωρίσουν το λογισμικό που εκτελείται στο φυσικό μηχάνημα από το λογισμικό που εκτελείται στο εικονικό μηχάνημα. Στο εξειδικευμένο λογισμικό (software/firmware) που δημιουργεί εικονικές μηχανές στους host καλείται hypervisor ή Διαχειριστής Εικονικών Μηχανών (Virtual Machine Manager – VMM).[2]

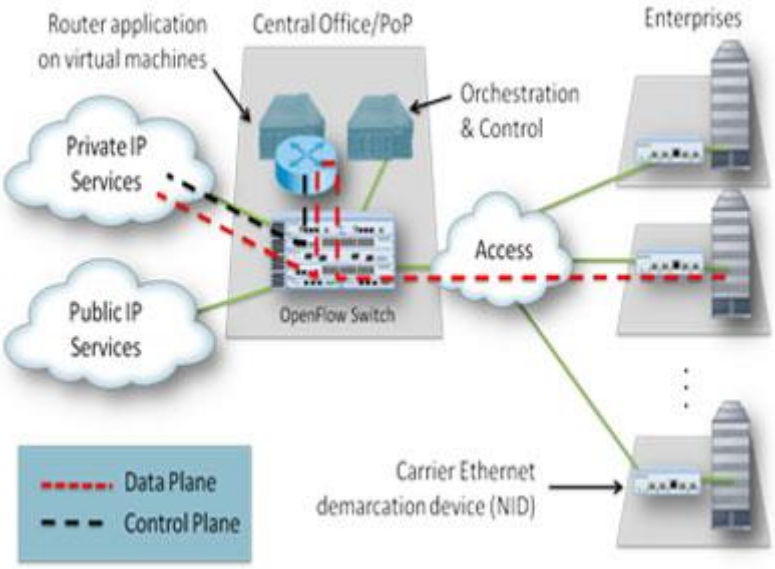
**2) Εικονικοποίηση Δικτυακών Λειτουργιών(NFV)**

**1.2.1) Γενικά στοιχεία**

Η παροχή υπηρεσιών δικτύωσης, στον τομέα των τηλεπικοινωνιών, παραδοσιακά βασίστηκε σε φορείς εκμετάλλευσης δικτύων που χρησιμοποιούν φυσικές ιδιόκτητες συσκευές και εξοπλισμό για κάθε λειτουργία που αποτελεί μέρος μιας συγκεκριμένης υπηρεσίας. Επιπλέον, τα συστατικά στοιχεία της υπηρεσίας έχουν αυστηρή αλυσίδα και/ή παραγγελία που πρέπει να αντικατοπτρίζονται στην τοπολογία του δικτύου και στον εντοπισμό των στοιχείων της υπηρεσίας. Αυτά, σε συνδυασμό με απαιτήσεις για υψηλή ποιότητα, σταθερότητα και αυστηρή τήρηση πρωτοκόλλου, έχουν οδηγήσει σε μακρούς κύκλους προϊόντων, πολύ χαμηλή ευελιξία υπηρεσιών και μεγάλη εξάρτηση από εξειδικευμένο υλικό. Αυτά σε συνδυασμό με τις ολοένα και αυξανόμενες απαιτήσεις των χρηστών έχουν αναγκάσει τους τηλεπικοινωνιακούς παρόχους να βρουν τρόπους οικοδόμησης πιο δυναμικών και εξυπηρετικών δικτύων με στόχο τη μείωση των κύκλων των προϊόντων, των λειτουργικών και κεφαλαιουχικών δαπανών και τη βελτίωση της ευελιξίας των υπηρεσιών.[3]

Η εικονικοποίηση δικτυακών λειτουργιών(NFV) έχει προταθεί ως ένας τρόπος αντιμετώπισης αυτών των προκλήσεων αξιοποιώντας την τεχνολογία εικονικοποίησης, για να προσφέρει έναν νέο τρόπο σχεδιασμού, ανάπτυξης και διαχείρισης υπηρεσιών δικτύωσης. Η βασική ιδέα του NFV είναι η αποσύνδεση του φυσικού εξοπλισμού δικτύου από τις λειτουργίες που λειτουργούν πάνω σε αυτές. Αυτό σημαίνει ότι μια λειτουργία δικτύου - όπως ένα τείχος προστασίας - μπορεί να αποσταλεί σε έναν τηλεπικοινωνιακό πάροχο ως παράδειγμα απλού λογισμικού. Αυτό επιτρέπει

την ενοποίηση πολλών τύπων εξοπλισμού δικτύου σε διακομιστές, μεταγωγείς και συσκευές αποθήκευσης, οι οποίοι θα μπορούσαν να βρίσκονται σε κέντρα δεδομένων, κατανεμημένους κόμβους δικτύου και σε χώρους τελικού χρήστη. Με αυτόν τον τρόπο, μια υπηρεσία μπορεί να αποσυντεθεί σε ένα σύνολο λειτουργιών εικονικού δικτύου (VNFs), οι οποίες θα μπορούσαν στη συνέχεια να υλοποιηθούν σε λογισμικό που εκτελείται σε έναν ή περισσότερους φυσικούς διακομιστές. Εν συνεχεία, τα VNFs μπορούν να μεταφερθούν και να εκτελεστούν σε διαφορετικές τοποθεσίες δικτύου (π.χ. με σκοπό την εισαγωγή μιας υπηρεσίας που στοχεύει τους πελάτες σε μια δεδομένη γεωγραφική θέση) χωρίς απαραίτητα να απαιτεί την αγορά και την εγκατάσταση νέου υλικού.[3]




**Εικόνα 1.1 Υπηρεσία δρομολογητή με NFV[4]**

Για να επιτευχθούν αυτά τα οφέλη, η τεχνολογία NFV εισάγει ορισμένες διαφορές στον τρόπο με τον οποίο πραγματοποιείται η παροχή υπηρεσιών δικτύου σε σύγκριση με την συνηθισμένη λογική. Οι διαφορές αυτές είναι κυρίως οι ακόλουθες[3]:

- **Αποσύνδεση λογισμικού από υλικό.** Δεδομένου ότι το στοιχείο δικτύου δεν είναι πλέον μια σύνθεση ολοκληρωμένων στοιχείων υλικού και λογισμικού, η εξέλιξη και των δύο είναι ανεξάρτητη η μία από την άλλη. Αυτό επιτρέπει ξεχωριστά χρονοδιαγράμματα ανάπτυξης και ξεχωριστή συντήρηση λογισμικού και υλικού.
- **Ευέλικτη ανάπτυξη λειτουργίας δικτύου.** Η αποσύνδεση του λογισμικού από το υλικό βοηθά στην εκ νέου εκχώρηση και την κοινή χρήση των πόρων της υποδομής, συνεπώς, μαζί, το υλικό και το λογισμικό, μπορούν να εκτελούν διαφορετικές λειτουργίες σε διαφορετικές χρονικές στιγμές. Αυτό βοηθά τους φορείς εκμετάλλευσης δικτύων να αναπτύσσουν νέες υπηρεσίες δικτύου ταχύτερα μέσω της ίδιας φυσικής πλατφόρμας. Επομένως, τα συστατικά μπορούν να δημιουργηθούν σε κατάσταση λειτουργίας σε οποιαδήποτε συσκευή με δυνατότητα NFV στο δίκτυο και οι συνδέσεις τους μπορούν να ρυθμιστούν με ευέλικτο τρόπο.
- **Δυναμική κλιμάκωση.** Η αποσύνδεση της λειτουργικότητας της λειτουργίας του δικτύου σε κομμάτια παρέχει μεγαλύτερη ευελιξία για την κλιμάκωση της πραγματικής απόδοσης της NFV με πιο δυναμικό τρόπο

### 1.2.2) Εικονικοποιημένες Δικτυακές Λειτουργίες(VNFs)

Εικονικοποιημένες δικτυακές λειτουργία ή VNFs είναι οι υλοποιήσεις λογισμικού των διαφόρων λειτουργιών δικτύου που μπορούν να αναπτυχθούν σε μια υποδομή NFV και χρειάζονται για να ενεργοποιηθεί το δίκτυο. Μία εικονικοποιημένη δικτυακή λειτουργία χειρίζεται μια συγκεκριμένη λειτουργία δικτύου, που εκτελείται σε μία ή περισσότερες εικονικές μηχανές πάνω από την υποδομή υλικού δικτύωσης. Οι λειτουργίες αυτές μπορούν να θεωρηθούν δομικές μονάδες και μπορούν να συνδεθούν ή να συνδυαστούν, παρέχοντας όλες τις δυνατότητες που απαιτούνται για την παροχή μιας πλήρους υπηρεσίας επικοινωνίας δικτύου.[5]

Παραδείγματα διαφόρων λειτουργιών εικονικού δικτύου μπορούν να βρεθούν σε όλες τις περιοχές ενός τηλεπικοινωνιακού δικτύου και μπορούν να περιλαμβάνουν:

- Λειτουργίες μεταγωγής: BNG, CG-NAT, δρομολογητές.
- Στοιχεία πύλης σήραγγας: Πύλες IPSec / SSL VPN.
- Ανάλυση κίνησης: Μέτρηση DPI, QoEI.
- Σηματοδότηση: SBCs, IMS.
- Βελτιστοποίηση επιπέδου εφαρμογής: CDNs, εξισορροπητές δικτύου.
- Δρομολογητές οικιακής χρήσης.
- Κόμβοι κινητού δικτύου: HLR / HSS, MME, SGSN, GGSN / PDN-GW, RNC.
- Λειτουργίες σε όλο το δίκτυο: Έλεγχος πολιτικής διακομιστών AAA, πλατφόρμες χρέωσης.
- Λειτουργίες ασφαλείας: τείχη προστασίας, συστήματα ανίχνευσης εισβολών, σαρωτές ιών, προστασία από ανεπιθύμητα μηνύματα.
- Μετάφραση διευθύνσεων Δικτύου(NAT)
- Σύστημα ονομάτων περιοχών(Domain Name System – DNS)

Με αυτό τον τρόπο, μπορεί να φανεί ότι ένας τεράστιος αριθμός VNFs μπορεί να εκτελεστεί σε ένα δίκτυο χρησιμοποιώντας την τεχνική NFV.

### 1.2.3) Δομικά στοιχεία της εικονικοποίησης δικτυακών λειτουργιών(NFV)

Η λειτουργία του NFV αποτελείτε από 3 βασικά μέρη[6]:

- Τις εικονικοποιημένες δικτυακές λειτουργίες (VNFs) οι οποίες είναι υλοποιήσεις σε λογισμικό διάφορων δικτυακών λειτουργιών που μπορούν να αναπτυχθούν πάνω σε υποδομή εικονικοποίησης δικτυακών λειτουργιών (NFVI).
- Την υποδομή εικονικοποίησης δικτυακών λειτουργιών που είναι το σύνολο του υλικού και λογισμικού που συνθέτει το περιβάλλον όπου αναπτύσσονται οι VNFs. Η υποδομή NFV μπορεί να εκτείνεται σε περισσότερες από μια τοποθεσίες ενώ το δίκτυο που παρέχει συνδεσιμότητα ανάμεσα σε αυτές τις τοποθεσίες θεωρείται μέρος της υποδομής.
- Το πλαίσιο διαχείρισης και ενορχήστρωσης της εικονικοποίησης δικτυακών λειτουργιών (NFV-MANO), που είναι το σύνολο όλων των λειτουργικών μονάδων, των δεδομένων που αυτά χρησιμοποιούν, των σημείων αναφοράς και των διεπαφών μέσω των οποίων αυτές ανταλλάσσουν πληροφορίες, με σκοπό τη διαχείριση και την ενορχήστρωση των VNFs και της υποδομής NFV.

Βασικό στοιχείο τόσο για την υποδομή NFV όσο και για το NFV-MANO είναι η πλατφόρμα NFV. Από την πλευρά της υποδομής NFV, αυτή αποτελείται από τους εικονικούς και φυσικούς πόρους επεξεργασίας και αποθήκευσης και από λογισμικό εικονικοποίησης. Από την πλευρά του NFV-MANO, η πλατφόρμα NFV αποτελείται από το λογισμικό διαχείρισης και εικονικοποίησης. Η πλατφόρμα NFV υλοποιεί λειτουργίες που την καθιστούν κατάλληλη για χρήση σε τηλεπικοινωνιακό περιβάλλον, όπως διαχείριση και επίβλεψη των διάφορων συνθετικών της πλατφόρμας, ανάνηψη από σφάλματα και αποτελεσματική ασφάλεια, στοιχεία απαραίτητα για ένα δίκτυο δημόσιας χρήσης.

### 1.2.4) Κατανεμημένη NFV[6]

Η αρχική αντίληψη για την NFV, ήταν ότι η ικανότητα εικονικοποίησης θα πρέπει να εφαρμόζεται σε Data Centers(Υπολογιστικά Κέντρα). Αυτή η προσέγγιση λειτουργεί σε πολλές, αλλά όχι σε όλες τις περιπτώσεις. Η τεχνολογία NFV προϋποθέτει και τονίζει την ευρύτερη δυνατή ευελιξία ως προς τη φυσική θέση του εικονικοποιημένων λειτουργιών.

Ιδανικά, οι εικονικοποιημένες λειτουργίες θα πρέπει να βρίσκονται στο σημείο που είναι πιο αποτελεσματικές και λιγότερο δαπανηρές. Αυτό σημαίνει ότι ένας πάροχος υπηρεσιών θα πρέπει να είναι ελεύθερος να αναπτύξει NFV σε όλες τις πιθανές τοποθεσίες, από το κέντρο δεδομένων, στους κόμβους του δικτύου έως και στις εγκαταστάσεις του καταναλωτή. Η προσέγγιση αυτή, που είναι γνωστή ως κατανεμημένη NFV, έχει τονιστεί από την αρχή καθώς η NFV αναπτυσσόταν, και κατέχει εξέχουσα θέση στα τελευταία έγγραφα της ομάδας NFV.

Για ορισμένες περιπτώσεις, υπάρχουν σαφή πλεονεκτήματα για ένα πάροχο υπηρεσιών αν αναπτύξει εικονικοποιημένες λειτουργίες στις εγκαταστάσεις του πελάτη. Τα πλεονεκτήματα αυτά κυμαίνονται από οικονομικά έως και την απόδοση.

Κατά το σχεδιασμό και την ανάπτυξη του λογισμικού που παρέχουν οι VNFs, οι κατασκευαστές μπορούν να δομήσουν το λογισμικό σε υποσυστήματα και τα υποσυστήματα αυτά να τα παρέχουν εικονικές μηχανές. Αυτά τα υποσυστήματα ονομάζονται VNF Υποσυστήματα(VNFCs). Μια VNF υλοποιείται με ένα ή περισσότερα VNFCs.

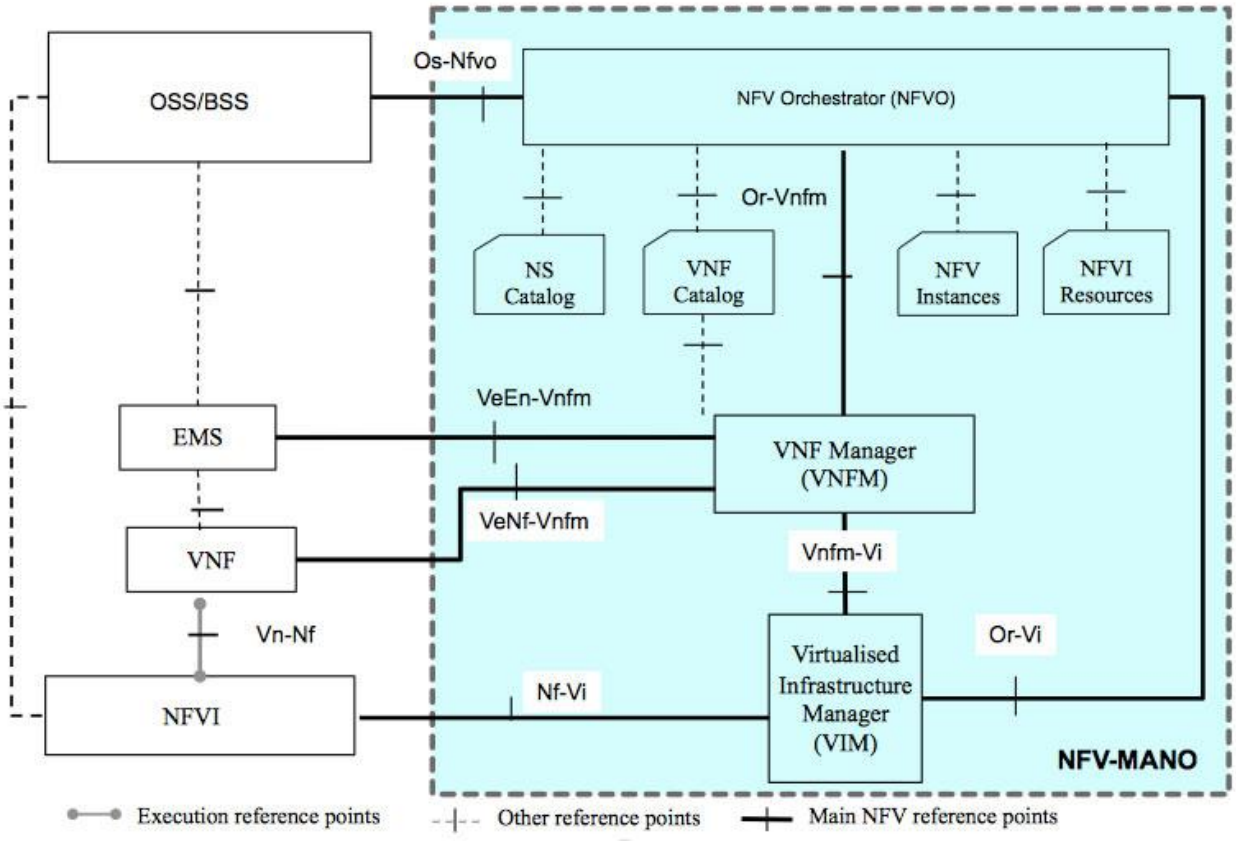
### 1.2.5) Διαχείριση και ενορχήστρωση της NFV(NFV – Management and Orchestration/MANO)

Ένα σημαντικό ζήτημα που προκύπτει στην NFV είναι η διαχείριση , πράγμα που επιλύει η NFV-MANO). Το NFV MANO προέκυψε από τον οργανισμό του Ευρωπαϊκού Ινστιτούτου Τηλεπικοινωνιακών Προτύπων(ETSI NFV). Πρόκειται για το καθορισμένο από το ETSI πλαίσιο για τη διαχείριση και ενορχήστρωση όλων των πόρων του ενός νεφελώδους(cloud) κέντρου δεδομένων. Αυτό περιλαμβάνει πόρους υπολογιστών, δικτύωσης, αποθήκευσης και εικονικής μηχανής(VM). Η κύρια εστίαση του NFV MANO είναι να επιτρέψει την ευέλικτη επιβίβαση και να παρακάμψει το χάος που μπορεί να συσχετιστεί με την γρήγορη περιστροφή των εξαρτημάτων του δικτύου.

Το NFV MANO χωρίζεται σε τρία λειτουργικά τμήματα[7]:

- NFV Orchestrator: Υπεύθυνος για την εγκατάσταση των νέων υπηρεσιών δικτύου. Βασικά καθήκοντα του είναι η διαχείριση του κύκλου ζωής του δικτυακού συστήματος, η διαχείριση των πόρων. η επικύρωση και η εξουσιοδότηση των αιτήσεων πόρων υποδομής virtualization infrastructure (NFVI)
- VNF Manager: Παρακολουθεί τη διαχείριση του κύκλου ζωής των VNF, συντονίζει και προσαρμόζει τη διαμόρφωση και την αναφορά συμβάντων μεταξύ NFVI και συστημάτων διαχείρισης δικτύου.
- VIM (Εικονικοποιημένος διαχειριστής υποδομής): Ελέγχει και διαχειρίζεται τους υπολογιστές NFVI, τους αποθηκευτικούς πόρους και τους πόρους δικτύου

Προκειμένου η αρχιτεκτονική NFV MANO να λειτουργήσει σωστά και αποτελεσματικά, πρέπει να ενσωματωθεί με ανοικτές διεπαφές προγράμματος εφαρμογής(API) στα υπάρχοντα συστήματα. Το στρώμα MANO λειτουργεί με πρότυπα για τις βασικές VNF και δίνει στους χρήστες τη δυνατότητα επιλογής και επιλογής από υπάρχοντες πόρους NFVI για την ανάπτυξη της πλατφόρμας ή του στοιχείου τους.




**Εικόνα 1.2 Αρχιτεκτονική NFV-MANO[7]**

Οι υπηρεσίες NFV αναπτύσσονται σε πλατφόρμα υλικού COTS (Comercial off-the-shelf) και τυπικά λειτουργούν σε υλικό βασισμένο σε Intel X86 και στο βασικό εξοπλισμό μεταγωγής. Το πρώιμο μοντέλο του NFV, ETSI MANO, είναι μια κοινή αρχιτεκτονική αναφοράς. Η αρχιτεκτονική NFV που αναπτύχθηκε από το ETSI MANO περιλαμβάνει επίσης τα συστήματα διαχείρισης στοιχείων (EMS),

τα οποία περιγράφουν τον τρόπο διαχείρισης των επιμέρους VNF σε μια πλατφόρμα υλικού. Συνήθως, το EMS διαχειρίζεται τις λειτουργίες και τις δυνατότητες μέσα σε κάθε NE(Network Element-Στοιχείο Δικτύου), αλλά δεν διαχειρίζεται την κυκλοφορία μεταξύ διαφορετικών NE στο δίκτυο. Το EMS παρέχει τα θεμέλια για την εφαρμογή αρχιτεκτονικών Συστημάτων Υποστήριξης Λειτουργιών(OSS) με στρώματα TMN, οι οποίες επιτρέπουν στους παρόχους υπηρεσιών να ικανοποιούν τις ανάγκες των πελατών για την ταχεία ανάπτυξη νέων υπηρεσιών, καθώς και να πληρούν τις αυστηρές απαιτήσεις ποιότητας υπηρεσιών (QoS).[8]

**3)Δίκτυα ορισμένα από λογισμικό(SDN)**

Τα δίκτυα υπολογιστών μπορούν να χωριστούν σε τρία επίπεδα λειτουργιών[9]:

- Το επίπεδο δεδομένων αντιστοιχεί στις συσκευές δικτύωσης, οι οποίες είναι υπεύθυνες για την (αποτελεσματική)διαβίβαση δεδομένων.
- Το επίπεδο ελέγχου αντιπροσωπεύει τα πρωτόκολλα που χρησιμοποιούνται για τη συμπλήρωση των πινάκων προώθησης των στοιχείων του επιπέδου δεδομένων.
- Το επίπεδο διαχείρισης περιλαμβάνει τις υπηρεσίες λογισμικού, όπως τα εργαλεία που βασίζονται σε SNMP και χρησιμοποιούνται για την απομακρυσμένη παρακολούθηση και διαμόρφωση των λειτουργιών ελέγχου.

Με λίγα λόγια δηλαδή, οι πολιτικές δικτύου ορίζονται στο επίπεδο διαχείρισης, το επίπεδο ελέγχου επιβάλλει την πολιτική και το επίπεδο δεδομένων εκτελεί τη διαβίβαση δεδομένων.

Στα παραδοσιακά δίκτυα IP, τα επίπεδα ελέγχου και δεδομένων είναι στενά συζευγμένα, ενσωματωμένα στις ίδιες συσκευές δικτύωσης και η όλη δομή είναι αποκεντρωμένη. Αυτό θεωρήθηκε σημαντικό για το σχεδιασμό του Διαδικτύου στις αρχές της ανάπτυξής του γιατί φαινόταν ο καλύτερος τρόπος για να διασφαλιστεί η ανθεκτικότητα του δικτύου, ο οποίος ήταν ένας κρίσιμος στόχος του σχεδιασμού. Η προσέγγιση αυτή ήταν αρκετά αποτελεσματική όσον αφορά την απόδοση του δικτύου, με ταχεία αύξηση του ρυθμού γραμμής και της πυκνότητας των θυρών. Ωστόσο, το αποτέλεσμα είναι μια πολύ σύνθετη και σχετικά στατική αρχιτεκτονική, και είναι ο βασικός λόγος για τον οποίο τα παραδοσιακά δίκτυα είναι δύσκολα και πολύπλοκα για τη διαχείριση και τον έλεγχο.[9]

Αυτό το πρόβλημα επιλύουν τα Δίκτυα οριζόμενα από λογισμικό (Software Defined Networks – SDN), τα οποία αλλάζουν τον τρόπο με τον οποίο σχεδιάζουμε και διαχειριζόμαστε τα δίκτυα. Το SDN έχει δύο καθοριστικά χαρακτηριστικά. Ο όρος SDN σχεδιάστηκε αρχικά για να αντιπροσωπεύει τις ιδέες και να εργάζεται γύρω από την ανάπτυξη του πρωτοκόλλου OpenFlow στο Πανεπιστήμιο του Στάνφορντ . Όπως ορίστηκε, το SDN αναφέρεται σε μια αρχιτεκτονική δικτύου όπου η κατάσταση προώθησης στο επίπεδο δεδομένων ελέγχεται από ένα επίπεδο απομακρυσμένου ελέγχου αποσυνδεδεμένο από το πρώτο.[9]

### 1.3.1) Γενικά στοιχεία

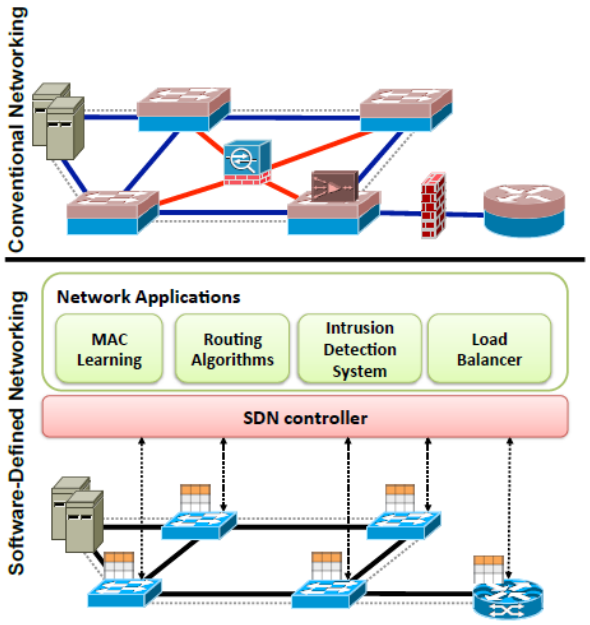
Για να μην ορίζεται ως SDN οτιδήποτε αφορά λογισμικό, ορίζουμε ένα Δίκτυο ορισμένο από Λογισμικό ως μια αρχιτεκτονική δικτύου με τέσσερις πυλώνες[9]:

1) Τα επίπεδα ελέγχου και δεδομένων αποσυνδέονται. Η λειτουργία ελέγχου αφαιρείται από συσκευές δικτύου που θα γίνουν απλά στοιχεία προώθησης(πακέτων).

2) Οι αποφάσεις προώθησης στις πιο συνήθεις περιπτώσεις βασίζονται στις ροές(flows), αντί βάσει προορισμού. Μια ροή καθορίζεται ευρέως από ένα σύνολο τιμών πεδίων πακέτων που λειτουργούν ως κριτήριο αντιστοίχισης (φίλτρου) και ένα σύνολο ενεργειών (οδηγίες). Στο περιβάλλον SDN / OpenFlow, μια ροή είναι μια ακολουθία πακέτων μεταξύ μιας πηγής και ενός προορισμού. Όλα τα πακέτα μιας ροής λαμβάνουν ίδιες πολιτικές υπηρεσιών στις συσκευές προώθησης. Η έννοια της ροής επιτρέπει την ενοποίηση της συμπεριφοράς διάφορων τύπων συσκευών δικτύου, συμπεριλαμβανομένων δρομολογητών, μεταγωγέων, τείχους προστασίας και middleboxes. Ο προγραμματισμός ροής επιτρέπει την άνευ προηγουμένου ευελιξία, που περιορίζεται μόνο στις δυνατότητες των πινάκων ροής που εφαρμόζονται.

3) Η λογική ελέγχου μετακινείται σε μια εξωτερική οντότητα, τον λεγόμενο ελεγκτή που είναι μια στοίβα λογισμικού που λειτουργεί με τεχνολογία διακομιστών βασικών προϊόντων και παρέχει τους βασικούς πόρους και τις αφαιρέσεις για να διευκολύνει τον προγραμματισμό των συσκευών προώθησης, βάσει μιας λογικά συγκεντρωτικής, αφηρημένης προβολής δικτύου

4) Το δίκτυο είναι προγραμματιζόμενο μέσω εφαρμογών λογισμικού που εκτελούνται στο επίπεδο εφαρμογής, που αλληλοεπιδρά με τις υποκείμενες συσκευές επιπέδου δεδομένων μέσω του επιπέδου ελέγχου. Αυτό αποτελεί θεμελιώδες χαρακτηριστικό της SDN αρχιτεκτονικής, που θεωρείται ως η κύρια αξία της




**Εικόνα 1.3 Σύγκριση παραδοσιακής και SDN αρχιτεκτονικής[9]**

Με βάση τα προηγούμενα, ως βασικά πλεονεκτήματα της, είναι ότι η αρχιτεκτονική SDN είναι[9]:

- **Άμεσα προγραμματιζόμενη:** Ο έλεγχος δικτύου είναι άμεσα προγραμματιζόμενος επειδή αποσυνδέεται από τις λειτουργίες προώθησης.

- **Ευέλικτη:** Η μετακίνηση του ελέγχου από την προώθηση σε άλλο αφαιρετικό επίπεδο επιτρέπει στους διαχειριστές να προσαρμόζουν δυναμικά τη ροή της κίνησης σε όλο το δίκτυο για να ανταποκρίνονται στις μεταβαλλόμενες ανάγκες.

- **Κεντρικοποιημένη διαχείριση:** Η νοημοσύνη του δικτύου είναι λογικά συγκεντρωμένη σε ελεγκτές SDN που βασίζονται σε λογισμικό και διατηρούν μια συνολική εικόνα του δικτύου, η οποία εμφανίζεται σε εφαρμογές και μηχανισμούς πολιτικών ως ένας και μόνος λογικός διακόπτης.

- **Διαμορφώνεται Προγραμματιστικά:** Το SDN επιτρέπει στους διαχειριστές δικτύου να ρυθμίζουν, να διαχειρίζονται, να ασφαλίζουν και να βελτιστοποιούν πολύ γρήγορα τους πόρους του δικτύου μέσω δυναμικών αυτοματοποιημένων προγραμμάτων SDN, τα οποία μπορούν να γράψουν, επειδή τα προγράμματα δεν εξαρτώνται από ιδιόκτητο λογισμικό.

### 1.3.2) Το πρωτόκολλο Openflow

Το πρωτόκολλο Openflow αποτελεί το βασικότερο πρωτόκολλο που χρησιμοποιείτε στα Δίκτυα που ορίζονται από Λογισμικό και είναι ένα πρωτόκολλο επικοινωνίας που παρέχει πρόσβαση στο επίπεδο προώθησης ενός μεταγωγέα ή ενός δρομολογητή. Στην ουσία είναι το πρωτόκολλο που μεταφέρει τις απαραίτητες πληροφορίες από το επίπεδο δεδομένων στο επίπεδο ελέγχου αλλά και τις αποφάσεις για την διαχείριση δικτύου στην αντίστροφη κατεύθυνση. Υπάρχουν κι άλλα πρωτόκολλα που μπορούν να χρησιμοποιηθούν για αυτή την διεργασία, όπως το NETCONF, αλλά στην παρούσα εργασία θα χρησιμοποιήσουμε το πρωτόκολλο OpenFlow.

Το OpenFlow επιτρέπει στους ελεγκτές δικτύου να καθορίζουν τη διαδρομή των πακέτων δικτύου σε ένα δίκτυο μεταγωγέων. Οι ελεγκτές διακρίνονται από τους μεταγωγείς. Αυτός ο διαχωρισμός του ελέγχου από την προώθηση επιτρέπει την πιο εξελιγμένη διαχείριση της κυκλοφορίας από ότι είναι εφικτό χρησιμοποιώντας λίστες ελέγχου πρόσβασης(ACL) και πρωτόκολλα δρομολόγησης. Επίσης, το OpenFlow επιτρέπει τη διαχείριση απομακρυσμένων διασυνδέσεων από διαφορετικούς παρόχους χρησιμοποιώντας ένα απλό, ανοιχτό πρωτόκολλο. Όλα αυτά καθιστούν το OpenFlow έναν παράγοντα που επιτρέπει τη δημιουργία δικτύων καθορισμένων από λογισμικό.[10]

Το OpenFlow επιτρέπει την απομακρυσμένη διαχείριση των πινάκων προώθησης πακέτων ενός μεταγωγέα, προσθέτοντας, τροποποιώντας και καταργώντας κανόνες και ενέργειες αντιστοίχισης πακέτων. Με αυτόν τον τρόπο, οι αποφάσεις δρομολόγησης μπορούν να γίνονται περιοδικά ή ad hoc από τον ελεγκτή και να μεταφράζονται σε κανόνες και ενέργειες με ρυθμιζόμενη διάρκεια ζωής, οι οποίες στη συνέχεια αναπτύσσονται στον πίνακα ροής ενός μεταγωγέα, αφήνοντας την

πραγματική προώθηση των αντιστοιχισμένων πακέτων στον μεταγωγέα σε ταχύτητα καλωδίου τη διάρκεια των κανόνων αυτών. Τα πακέτα που είναι ασύγκριτα από τον μεταγωγέα μπορούν να προωθηθούν στον ελεγκτή. Ο ελεγκτής μπορεί στη συνέχεια να αποφασίσει να τροποποιήσει τους υπάρχοντες κανόνες πίνακα ροής σε έναν ή περισσότερους διακόπτες ή να εφαρμόσει νέους κανόνες, ώστε να αποφευχθεί μια δομική ροή κίνησης μεταξύ διακόπτη και ελεγκτή. Μπορεί ακόμη και να αποφασίσει να προωθήσει την ίδια την κυκλοφορία, υπό την προϋπόθεση ότι έχει πει στην συσκευή Openflow να μεταφέρει ολόκληρα πακέτα αντί μόνο στην επικεφαλίδα τους.

Το πρωτόκολλο OpenFlow τοποθετείται πάνω από το Πρωτόκολλο Ελέγχου Μεταφοράς (TCP) και προδιαγράφει τη χρήση του TLS (Transport Layer Security). Οι ελεγκτές θα πρέπει να ακούν την θύρα TCP 6653 για τους διακόπτες που θέλουν να δημιουργήσουν μια σύνδεση. Παλαιότερες εκδόσεις του πρωτοκόλλου OpenFlow χρησιμοποιούσαν ανεπίσημα τη θύρα 6633

### 1.3.3) Αρχιτεκτονική των Δικτύων ορισμένων από λογισμικό[11]

#### 1.3.3.1) Εφαρμογή SDN (εφαρμογή SDN)

Οι εφαρμογές SDN είναι προγράμματα τα οποία, μέσω των διεπαφών μεταξύ επιπέδου ελέγχου και εφαρμογής(Northbound Interfaces - NBI), επικοινωνούν ρητά, άμεσα και προγραμματικά με τις απαιτήσεις δικτύου και την επιθυμητή συμπεριφορά δικτύου στον ελεγκτή SDN. Επιπλέον, μπορούν να καταγράφουν μια αφηρημένη άποψη του δικτύου για τους εσωτερικούς τους σκοπούς λήψης αποφάσεων.

Μια εφαρμογή SDN αποτελείται από μία λογική εφαρμογής SDN και έναν ή περισσότερους οδηγούς NBI. Οι εφαρμογές SDN μπορούν να εκθέσουν οι ίδιες ένα άλλο στρώμα εξαντλημένου ελέγχου δικτύου, προσφέροντας έτσι μία ή περισσότερες NBI υψηλότερου επιπέδου μέσω αντίστοιχων παραγόντων NBI.

#### 1.3.3.2) Ελεγκτής SDN

Ο ελεγκτής SDN είναι μια λογικά συγκεντρωμένη οντότητα που είναι υπεύθυνη για (i) τη μετάφραση των απαιτήσεων από το επίπεδο εφαρμογής SDN μέχρι τα SDN Datapaths και (ii) την παροχή των αιτήσεων SDN με μια αφηρημένη άποψη του δικτύου (που μπορεί να περιλαμβάνει στατιστικά στοιχεία και γεγονότα).

Ένας ελεγκτής SDN αποτελείται από έναν ή περισσότερους αντιπροσώπους των NBI, τη λογική ελέγχου SDN και το πρόγραμμα οδήγησης CDPI. Ο ορισμός ως μια λογικά συγκεντρωμένη οντότητα ούτε προδιαγράφει ούτε αποκλείει λεπτομέρειες εφαρμογής όπως η ομοσπονδία πολλών ελεγκτών, η ιεραρχική σύνδεση των ελεγκτών, οι διεπαφές επικοινωνίας μεταξύ των ελεγκτών, ούτε η εικονικοποίηση ή ο τεμαχισμός των πόρων του δικτύου.

#### 1.3.3.3) SDN Datapath

Το Datapath SDN είναι μια λογική συσκευή δικτύου, η οποία αποκαλύπτει την ορατότητα και τον ανεπιθύμητο έλεγχο των δυνατοτήτων προώθησης και επεξεργασίας δεδομένων. Η λογική αναπαράσταση μπορεί να περιλαμβάνει όλα ή ένα υποσύνολο των φυσικών πόρων του υποστρώματος.

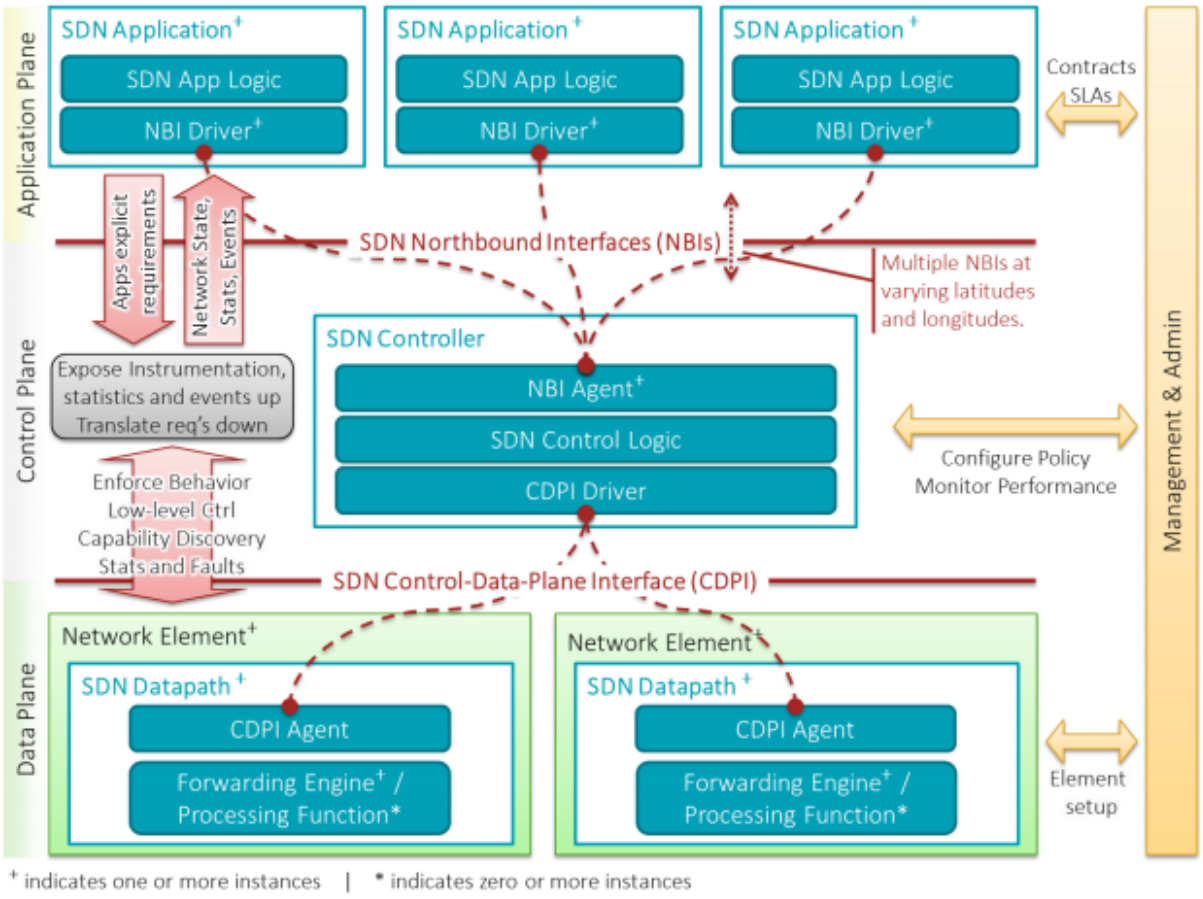
Ένα SDN Datapath περιλαμβάνει μια διεπαφή μεταξύ του επιπέδου δεδομένων και του επιπέδου ελέγχου και ένα σύνολο από μία ή περισσότερες μηχανές προώθησης της κυκλοφορίας και μηδέν ή περισσότερες λειτουργίες επεξεργασίας κυκλοφορίας. Αυτοί οι κινητήρες και λειτουργίες μπορεί να περιλαμβάνουν απλή προώθηση μεταξύ των εξωτερικών διεπαφών του datapath ή των εσωτερικών λειτουργιών επεξεργασίας ή τερματισμού της κυκλοφορίας. Ένα ή περισσότερα SDN Datapaths μπορεί να περιέχονται σε ένα στοιχείο (φυσικού) δικτύου - ένας ολοκληρωμένος φυσικός συνδυασμός πόρων επικοινωνίας, που διαχειρίζεται ως μονάδα. Ένα SDN Datapath μπορεί επίσης να οριστεί σε πολλαπλά φυσικά στοιχεία δικτύου. Αυτός ο λογικός ορισμός ούτε προδιαγράφει ούτε αποκλείει λεπτομέρειες υλοποίησης, όπως η λογική προς τη φυσική χαρτογράφηση, η διαχείριση των κοινών φυσικών πόρων, η εικονικοποίηση ή ο τεμαχισμός του SDN Datapath, η διαλειτουργικότητα με τη μη SDN δικτύωση ή η λειτουργία επεξεργασίας δεδομένων.

#### 1.3.3.4) Έλεγχος SDN σε διεπαφή δεδομένων(CDPI)

Το SDP-CDPI ή SBI(Southbound) είναι η διεπαφή που ορίζεται μεταξύ ενός ελεγκτή SDN και ενός Datapath SDN, το οποίο παρέχει τουλάχιστον (i) προγραμματικό έλεγχο όλων των λειτουργιών προώθησης, (ii) διαφήμιση δυνατοτήτων, (iii) αναφορά στατιστικών στοιχείων και (iv) ειδοποίηση συμβάντος. Μια αξία του SDN έγκειται στην προσδοκία ότι το CDPI εφαρμόζεται με ανοικτό, ουδέτερο και διαλειτουργικό τρόπο.

#### 1.3.3.5) Διασυνδέσεις SDN Northbound (NBI)

Τα SDN NBIs είναι διασυνδέσεις μεταξύ εφαρμογών SDN και ελεγκτών SDN και συνήθως παρέχουν αφηρημένες προβολές δικτύου και επιτρέπουν την άμεση έκφραση της συμπεριφοράς και των απαιτήσεων του δικτύου. Αυτό μπορεί να συμβεί σε οποιοδήποτε επίπεδο αφαίρεσης (γεωγραφικό πλάτος) και σε διαφορετικά σύνολα λειτουργικότητας (γεωγραφικό μήκος). Μια τιμή του SDN έγκειται στην προσδοκία ότι οι διεπαφές αυτές θα υλοποιηθούν με ανοικτό, ουδέτερο και με διαλειτουργικό τρόπο προμηθευτή.




**Εικόνα 1.4 Αρχιτεκτονική SDN**

**4) Σχέση μεταξύ SDN και NFV[12]**

Η βασική ομοιότητα μεταξύ των δικτύων ορισμένων από το λογισμικό (SDN) και της εικονικοποίησης δικτυακών λειτουργιών (NFV) είναι ότι και οι δύο χρησιμοποιούν την έννοια της αφαίρεσης του δικτύου. Το SDN επιδιώκει να διαχωρίσει τις λειτουργίες ελέγχου δικτύου από τις λειτουργίες προώθησης δικτύου, ενώ η NFV επιδιώκει να μεταφερθούν σε αφηρημένο επίπεδο οι υπηρεσίες προώθησης δικτύου και άλλες λειτουργίες δικτύωσης από το υλικό στο οποίο εκτελείται. Έτσι, και οι δύο εξαρτώνται σε μεγάλο βαθμό από την εικονικοποίηση, ώστε να επιτρέπουν τη σχεδίαση του δικτύου και την υποδομή να αφαιρούνται από το λογισμικό και στη συνέχεια να υλοποιούνται από υποκείμενο λογισμικό σε πλατφόρμες και συσκευές υλικού.

Όταν εκτελείται το SDN σε υποδομή NFV, η SDN προωθεί πακέτα δεδομένων από μια συσκευή δικτύου σε άλλη. Ταυτόχρονα, οι λειτουργίες ελέγχου δικτύωσης της SDN για τη δρομολόγηση, τον ορισμό πολιτικής και τις εφαρμογές εκτελούνται σε μια εικονική μηχανή κάπου στο δίκτυο. Έτσι, το NFV παρέχει βασικές λειτουργίες δικτύωσης, ενώ το SDN ελέγχει και τις ενορχηστρώνει για συγκεκριμένες χρήσεις. Το SDN επιτρέπει περαιτέρω τη διαμόρφωση και τη συμπεριφορά για τον προγραμματισμό και την τροποποίηση του προγράμματος.

Το SDN και το NFV διαφέρουν ως προς τον τρόπο με τον οποίο χωρίζουν τις λειτουργίες και τους αφηρημένους πόρους. Το SDN περιγράφει τους πόρους της φυσικής δικτύωσης - τους διακόπτες, τους δρομολογητές κ.ο.κ. - και μετακινεί τη λήψη αποφάσεων σε ένα εικονικό επίπεδο ελέγχου δικτύου. Σε αυτήν την προσέγγιση, το επίπεδο ελέγχου αποφασίζει πού να στείλει την κυκλοφορία, ενώ το υλικό συνεχίζει να κατευθύνει και να χειρίζεται την κυκλοφορία. Το NFV στοχεύει στο virtualization όλων των φυσικών πόρων του δικτύου κάτω από έναν hypervisor, το οποίο επιτρέπει στο δίκτυο να αναπτυχθεί χωρίς την προσθήκη περισσότερων συσκευών.

Ενώ τόσο το SDN όσο και το NFV καθιστούν τις αρχιτεκτονικές δικτύωσης πιο ευέλικτες και δυναμικές, εκτελούν διαφορετικούς ρόλους στον καθορισμό αυτών των αρχιτεκτονικών και της υποδομής που υποστηρίζουν.

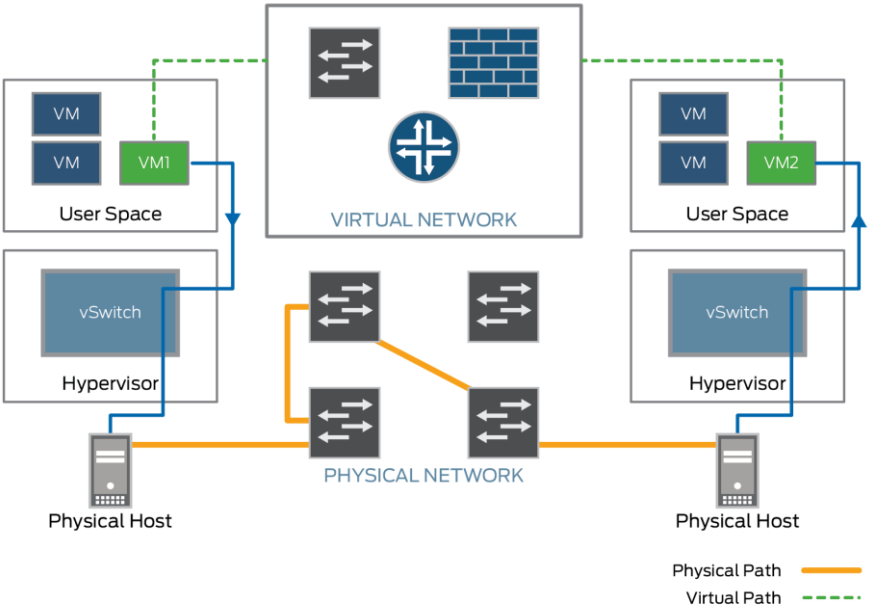
## Κεφάλαιο 2ο

# Τεχνικές και πρωτόκολλα για υλοποίηση δικτύων επικάλυψης(overlay networks) και πρωτόκολλα σήραγγας(tunneling protocols)

Μια κοινή μέθοδος που χρησιμοποιείται σήμερα για την υποστήριξη της εικονικοποίησης δικτύων είναι η χρήση δικτύων επικάλυψης(overlay networks) όπου ένα overlay δίκτυο χρησιμοποιεί υπάρχουσες υποδομές από το φυσικό δίκτυο (underlay network) για τη δημιουργία μίας τοπολογίας. Αυτή η αποσύνδεση του εικονικού από το φυσικό επιτρέπει την ταχεία προγραμματική παροχή του δικτύου για οποιαδήποτε εφαρμογή. Δεν χρειάζεται πλέον να ενορχηστρώνονται οι αλλαγές σε ένα σύνολο φυσικών συσκευών.

Η δημιουργία ενός εικονικού overlay δικτύου απλοποιεί τις λειτουργίες που πρέπει να εκτελεί το φυσικό δίκτυο επιτρέποντας την χρήση του πρωτοκόλλου IP από άκρο σε άκρο αφαιρώντας την πολυπλοκότητα της συντήρησης μίας Layer-2 τοπολογίας. Το overlay δίκτυο προσθέτει εκτός από την απλότητα, την ευκαμψία και την επεκτασιμότητα στο φυσικό δίκτυο, ενώ άλλα δίκτυα επικάλυψης λόγος κερδίζουν δημοτικότητα.[13]

Η Εικόνα 2.1 απεικονίζει ένα δίκτυο επικάλυψης. Από τις προοπτικές της εικονικής μηχανής 1 και της εικονικής μηχανής 2 (VM1 και VM2), η κίνηση μεταξύ τους λαμβάνει τη διαδρομή που εμφανίζεται από τη διακεκομμένη γραμμή, περνώντας από τις παραδοσιακές συσκευές δικτύωσης, όπως οι μεταγωγείς, οι δρομολογητές και τα τείχη προστασίας, . Ωστόσο, η κίνηση ακολουθεί την διαδρομή που φαίνεται στο κάτω μέρος της εικόνας.[13]




**Εικόνα 2.1 Παράδειγμα Overlay Δικτύου[13]**

Τα overlay δίκτυα προσφέρουν μια σειρά από πλεονεκτήματα που μπορούν να βοηθήσουν στην αντιμετώπιση ορισμένων από τις προκλήσεις των σύγχρονων δικτύων και κυρίως των κέντρων δεδομένων[14]:

- **Βελτιστοποιημένες λειτουργίες συσκευής.** Τα overlay δίκτυα επιτρέπουν τον διαχωρισμό των λειτουργιών της συσκευής με βάση το πού χρησιμοποιείται μια συσκευή στο δίκτυο. Μια συσκευή άκρης(edge device) μπορεί να βελτιστοποιήσει τις λειτουργίες της και όλα τα σχετικά πρωτόκολλά της με βάση τις πληροφορίες και την κλίμακα των τελικών δεδομένων και μια συσκευή πυρήνα(core device), μπορεί να βελτιστοποιήσει τις λειτουργίες και τα πρωτόκολλά της με βάση την κατάσταση σύνδεσης, βελτιστοποιώντας τη γρήγορη σύγκλιση. Αυτή η προσέγγιση μειώνει επίσης την πολυπλοκότητα των συσκευών δικτύου. Στην περίπτωση των overlays που βασίζονται σε διακομιστές, αυτή η λειτουργία υλοποιείται στον εξυπηρετητή. Στην περίπτωση επικαλύψεων με βάση το δίκτυο, αυτή η λειτουργία εφαρμόζεται στον πρώτο διακόπτη (στην κορυφή του rack).

- **Δυνατότητα κλιμάκωσης και ευελιξία.** Οι τεχνολογίες overlay επιτρέπουν στο δίκτυο να κλιμακώνεται με εστίαση της κλιμάκωσης στις συσκευές ακμής επικάλυψης. Με overlays που χρησιμοποιούνται στην άκρη του δικτύου, οι συσκευές του πυρήνα απελευθερώνονται από την ανάγκη να προστεθούν πληροφορίες τελικού χρήστη στους πίνακες προώθησης τους. Επιπλέον, η τοποθέτηση των τελικών κεντρικών υπολογιστών είναι πιο ευέλικτη επειδή το εικονικό overlay δίκτυο δεν χρειάζεται πλέον να περιορίζεται σε μια ενιαία φυσική τοποθεσία.

- **Δυνατότητα σύνδεσης Layer 2 πάνω από υπάρχουσες Layer-3 τοπολογίες.** Ένα άλλο πλεονέκτημα των τεχνολογιών επικάλυψης δικτύου είναι ότι μπορούν να αποσυνδέσουν την υπηρεσία δικτύου που παρέχεται στους τελικούς κεντρικούς υπολογιστές από την τεχνολογία που χρησιμοποιείται στο φυσικό δίκτυο. Για παράδειγμα, τα δρομολογημένα δίκτυα Layer 3 μπορούν να λειτουργούν διαδοχικά σε όλο το data-center. Ωστόσο, με τη χρήση ορισμένων τεχνολογιών overlay δικτύου, οι υπηρεσίες Layer 2 μπορούν επίσης να επεκταθούν σε μια δρομολογημένη τοπολογία.

- **Επικαλυπτόμενη διεύθυνση.** Οι περισσότερες τεχνολογίες επικαλύψεων που χρησιμοποιούνται σε ένα data-center επιτρέπουν την ταυτότητα εικονικού δικτύου σε μοναδικό πεδίο εφαρμογής και τον εντοπισμό μεμονωμένων ιδιωτικών δικτύων. Αυτό το πεδίο οριοθέτησης επιτρέπει την πιθανή επικάλυψη διευθύνσεων MAC και IP μεταξύ ενοικιαστών. Το overlay επιτρέπει επίσης τη διαχείριση του υποκείμενου χώρου διεύθυνσης υποδομής ξεχωριστά από τον χώρο διεύθυνσης του ενοικιαστή.

- **Διαχωρισμός των ρόλων και των ευθυνών.** Με την ενθυλάκωση που χρησιμοποιείται στις τεχνολογίες overlay, μπορεί επίσης να επιτευχθεί ο διαχωρισμός των τομέων χορήγησης. Ο διαχειριστής της υποδομής μπορεί να είναι υπεύθυνος για την αντιμετώπιση, τη διαθεσιμότητα και την εξισορρόπηση φορτίου και οι επιμέρους ενοικιαστές μπορούν να είναι υπεύθυνοι για την πολιτική και τις υπηρεσίες τους χωρίς να επηρεάζουν τις πολιτικές υποδομών.

Στις επόμενες ενότητες παραθέτονται μια σειρά από πρωτόκολλα σήραγγας(tunneling) που μπορούν να χρησιμοποιηθούν για τη δημιουργία overlay δικτύων.

### 2.1) Generic Routing Encapsulation (GRE)

Η ενθυλάκωση γενικής δρομολόγησης(Generic Routing Encapsulation – GRE) είναι ένα πρωτόκολλο επικοινωνίας που χρησιμοποιείται για την καθιέρωση μιας άμεσης, από σημείο σε σημείο, σύνδεσης μεταξύ κόμβων δικτύου. Αναπτύχθηκε από την Cisco Systems και χρησιμοποιείται για tunneling ανάμεσα σε δρομολογητές πηγής και προορισμού. Δεδομένου ότι είναι μια απλή και αποτελεσματική μέθοδος μεταφοράς δεδομένων μέσω δημόσιου δικτύου, όπως το Διαδίκτυο, το GRE επιτρέπει σε δύο κόμβους να μοιράζονται δεδομένα που δεν θα μπορούν να μοιράζονται μέσω του ίδιου του δημόσιου δικτύου.

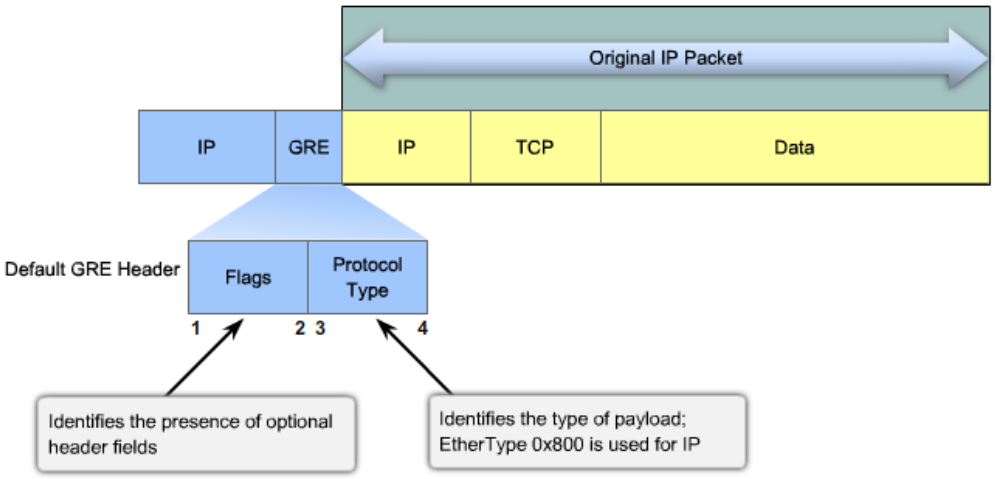
Το GRE υποστηρίζει σήραγγες πολλαπλών πρωτοκόλλων. Μπορεί να ενσωματώνει πολλαπλούς τύπους πακέτων πρωτοκόλλου μέσα σε μια σήραγγα IP. Με την προσθήκη μιας πρόσθετης κεφαλίδας GRE μεταξύ του ωφέλιμου φορτίου και της κεφαλίδας της IP σήραγγας παρέχει τη λειτουργικότητα πολλαπλών πρωτοκόλλων. Το IP tunneling με GRE, επιτρέπει την επέκταση του δικτύου συνδέοντας δίκτυα πολλαπλών πρωτοκόλλων σε ένα περιβάλλον κορμού ενός πρωτοκόλλου. Το GRE υποστηρίζει επίσης σήραγγες IP πολλαπλής διανομής. Τα πρωτόκολλα δρομολόγησης που χρησιμοποιούνται σε ολόκληρη τη σήραγγα επιτρέπουν τη δυναμική ανταλλαγή πληροφοριών δρομολόγησης στο εικονικό δίκτυο.[15]


**Εικόνα 2.2 GRE Tunnel[15]**

Το GRE ενσωματώνει ολόκληρο το αρχικό πακέτο IP με μια τυπική κεφαλίδα IP και κεφαλίδα GRE. Μια κεφαλίδα σήραγγας GRE περιέχει τουλάχιστον δύο υποχρεωτικά πεδία δύο byte:

- GRE σημαία
- Τύπος πρωτοκόλλου

Το GRE χρησιμοποιεί ένα πεδίο τύπου πρωτοκόλλου στην κεφαλίδα GRE για να υποστηρίξει την ενσωμάτωση οποιουδήποτε πρωτοκόλλου Layer 3. Η κεφαλίδα GRE, μαζί με την κεφαλίδα IP της σήραγγας, δημιουργεί τουλάχιστον 24 byte πρόσθετων επιβαρύνσεων για τα πακέτα με σήραγγα.




**Εικόνα 2.3 Ενθυλάκωση GRE[15]**

Σε γενικές γραμμές, το πρωτόκολλο GRE προσφέρει πολλά πλεονεκτήματα, όπως:

- Χρήση πολλαπλών πρωτοκόλλων σε μια ραχοκοκαλιά ενός πρωτοκόλλου.
- Παροχή λύσεων για δίκτυα με περιορισμένα hops
- Σύνδεση μη συνεχόμενων υποδικτύων.
- Είναι λιγότερο απαιτητικός μηχανισμός από τις εναλλακτικές λύσεις (π.χ. IPsec VPN).

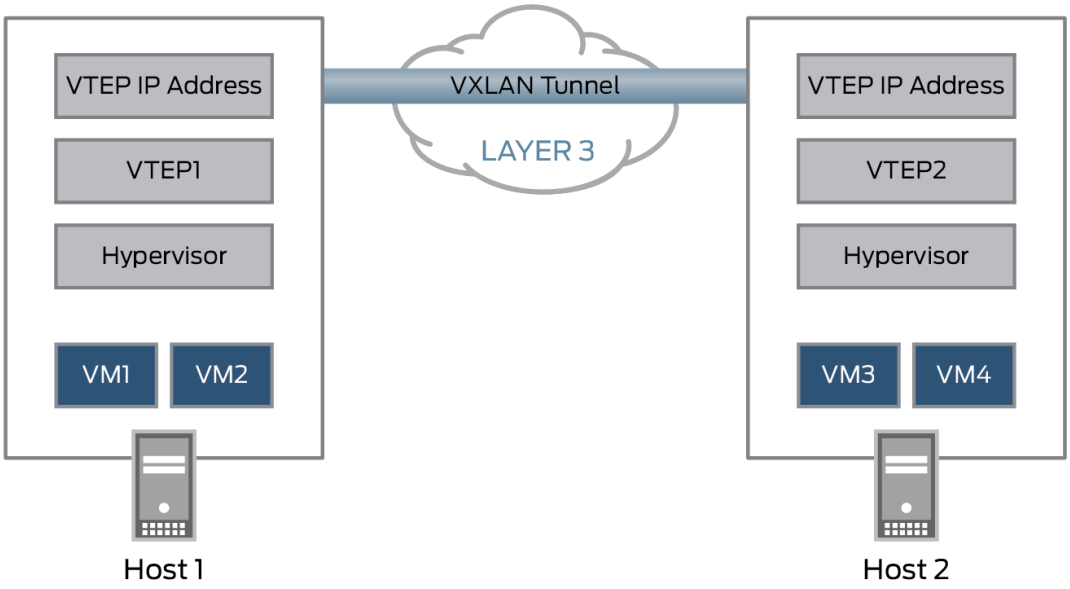
**2) Virtual Extensible LAN (VXLAN)**

Το VXLAN είναι ένα πρωτόκολλο σήραγγας που ενσωματώνει τα πλαίσια Ethernet Layer 2 στα πακέτα UDP Layer 3, επιτρέποντάς σας να δημιουργείτε υποδίκτυα ή τμήματα εικονικού Layer 2 που καλύπτουν φυσικά δίκτυα Layer 3. Τρία βασικά σημεία του VXLAN είναι:[13]

**α)Αναγνωριστικό δικτύου VXLAN (VNI)[13]**Κάθε υποδίκτυο επιπεδου-2(OSI) ή τμήμα του αναγνωρίζεται με μοναδικό τρόπο από ένα αναγνωριστικό δικτύου VXLAN (VNI) το οποίο καταλαμβάνει τμήματα με τον ίδιο τρόπο που το κάνει η κίνηση των τμημάτων VLAN ID του IEEE 802.1Q. Όπως συμβαίνει με ένα VLAN, τα οι εικονικές μηχανές στο ίδιο VNI μπορούν να επικοινωνούν απευθείας μεταξύ τους, ενώ σε διαφορετικά VNI χρειάζονται ένα δρομολογητή για να επικοινωνούν μεταξύ τους.

**β)Το τελικό σημείο σήραγγας VXLAN (VTEP)[13]**

Η οντότητα που εκτελεί την ενθυλάκωση και την από-ενθυλάκωση των πακέτων ονομάζεται τελικό σημείο σήραγγας VXLAN(Virtual Tunnel End-Point).Τα VTEPs συνήθως βρίσκονται σε κεντρικούς υπολογιστές hypervisor, όπως ο οι κεντρικοί υπολογιστές KVM. Κάθε VTEP έχει δύο διεπαφές. Η μία είναι μια διεπαφή μεταγωγής που επικοινωνεί τις εικονικές μηχανές στο host και παρέχει επικοινωνία μεταξύ των εικονικών μηχανών στο τοπικό τμήμα LAN. Η δεύτερη μια διεπαφή IP που επικοινωνεί με το δίκτυο Layer 3. Κάθε VTEP έχει μια μοναδική διεύθυνση IP που χρησιμοποιείται για τη δρομολόγηση των πακέτων UDP μεταξύ VTEPs.




**Εικόνα 2.4 Παράδειγμα VXLAN σε data-center[13]**

Σε ένα απλό παράδειγμα στην εικόνα 2.4 όταν το VTEP1 δέχεται ένα πλαίσιο Ethernet από το VM1 που απευθύνεται στο VM3, χρησιμοποιεί το VNI και το MAC προορισμού για να αναζητήσει στον πίνακα προώθησής του το VTEP για να στείλει το πακέτο. Το VTEP1 προσθέτει στη συνέχεια μια κεφαλίδα VXLAN που περιέχει το VNI στο πλαίσιο Ethernet, ενσωματώνει το πλαίσιο σε ένα πακέτο UDP επιπέδου 3 και δρομολογεί το πακέτο στο VTEP2 μέσω του δικτύου επιπέδου 3.Το VTEP2 αποκρυπτογραφεί το αρχικό πλαίσιο Ethernet και το προωθεί στο VM3. Τα VM1 και VM3 αγνοούν εντελώς τη σήραγγα VXLAN και το δίκτυο Layer 3 μεταξύ τους.

**γ)Μορφή πακέτων VXLAN[13]**

Τα βασικά πεδία για το πακέτο VXLAN σε κάθε μία από τις κεφαλίδες πρωτοκόλλου είναι:

- Εξωτερική κεφαλίδα MAC - Περιέχει τη διεύθυνση MAC της πηγής VTEP και τη διεύθυνση MAC του επόμενου δρομολογητή. Κάθε δρομολογητής κατά μήκος της διαδρομής του πακέτου επανεγγράφει αυτή την κεφαλίδα έτσι ώστε η διεύθυνση πηγής να είναι η διεύθυνση MAC του δρομολογητή και η διεύθυνση προορισμού είναι η διεύθυνση MAC του next-hop δρομολογητή.

- Εξωτερική κεφαλίδα IP - Περιέχει τις διευθύνσεις IP των VTEP πηγής και προορισμού.

- Εξωτερική κεφαλίδα UDP - Περιέχει θύρες UDP προέλευσης και προορισμού: i) Θύρα UDP προέλευσης - Το πρωτόκολλο VXLAN επαναπροσδιορίζει αυτό το τυποποιημένο πεδίο σε UDP κεφαλίδα πακέτων. Αντί να χρησιμοποιήσετε αυτό το πεδίο για τη θύρα UDP πηγής, χρησιμοποιεί το πρωτόκολλο ως αριθμητικό αναγνωριστικό για τη συγκεκριμένη ροή μεταξύ VTEPs. Το VXLAN δεν καθορίζει τον τρόπο με τον οποίο παράγεται αυτός ο αριθμός, αλλά συνήθως η πηγή VTEP το υπολογίζει από ένα hash κάποιου συνδυασμού πεδίων από

το εσωτερικό Layer 2 πακέτου και τις κεφαλίδες του Layer 3 ή του Layer 4. ii) Θύρα UDP προορισμού, δηλαδή η θύρα UDP VXLAN. Οι αριθμοί που σύμφωνα με την IANA έχει διαθέτει η θύρ 4789 στο VXLAN.

- Κεφαλίδα VXLAN - Περιέχει τον VNI 24 bit.

- Πρωτότυπο πλαίσιο Ethernet - Περιέχει το αρχικό πλαίσιο Layer 2 Ethernet.


**Εικόνα 2.5 Κεφαλίδα VXLAN[13]**

Συνολικά, η ενθυλάκωση VXLAN προσθέτει μεταξύ 50 και 54 byte επιπλέον κεφαλίδας πληροφορίες στο αρχικό πλαίσιο Ethernet. Επειδή αυτό μπορεί να οδηγήσει σε πλαίσια Ethernet που υπερβαίνουν την προεπιλεγμένη MTU 1514 byte, η βέλτιστη πρακτική είναι η εφαρμογή πλαισίων jumbo σε όλο το δίκτυο.

### 2.3) Overlay Transport Virtualization (OTV)[16]

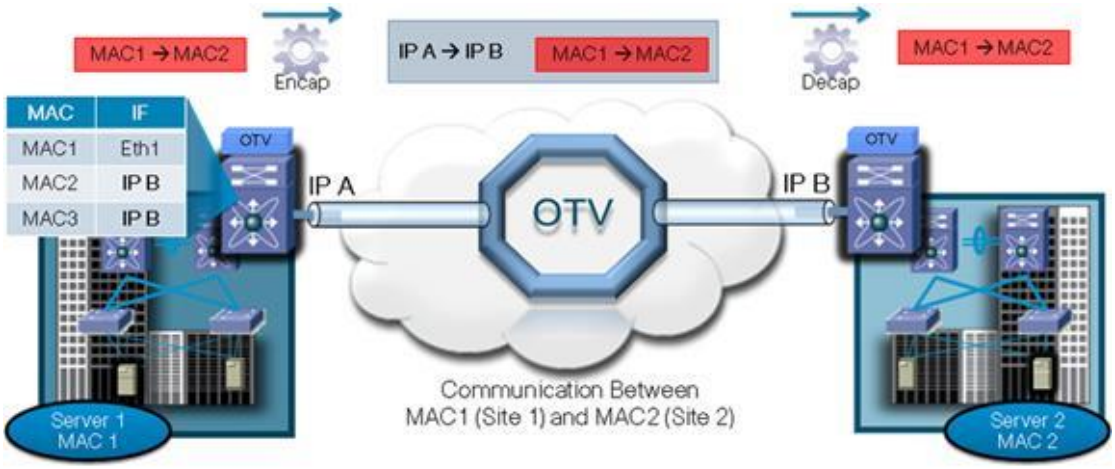
Η εικονικοποιημένη επικάλυψη μεταφοράς (OTV) είναι ένα πρωτόκολλο της Cisco για τη μετάδοση κίνησης επιπέδου 2 μεταξύ των δικτύων υπολογιστών του επιπέδου 3. Το OTV παρέχει μια λειτουργικά βελτιστοποιημένη λύση για την επέκταση της συνδεσιμότητας επιπέδου 2 σε κάθε μεταφορά. Επομένως, το OTV είναι κρίσιμο για την αποτελεσματική ανάπτυξη των κατανεμημένων data-center για τη στήριξη της διαθεσιμότητας των εφαρμογών και της ευελιξίας της κινητικότητας του φόρτου εργασίας

Η OTV ακολουθεί μια τεχνική "διεύθυνση MAC σε IP" για την υποστήριξη των VPN Layer-2 και την επέκταση των LAN(τοπικών δικτύων) σε κάθε μεταφορά. Η μεταφορά μπορεί να βασίζεται στο Layer-2, να βασίζεται στο Layer-3, να αλλάζει IP, να αλλάζει ετικέτες ή οτιδήποτε άλλο, αρκεί να μπορεί να μεταφέρει πακέτα IP. Χρησιμοποιώντας τις αρχές της δρομολόγησης MAC, η OTV παρέχει overlay που επιτρέπει τη σύνδεση επιπέδου 2 μεταξύ ξεχωριστών domains του Layer-2, διατηρώντας ταυτόχρονα αυτά τα domains ανεξάρτητα και διατηρώντας τα πλεονεκτήματα απομόνωσης, ανθεκτικότητας και εξισορρόπησης φορτίου μιας διασύνδεσης που βασίζεται στην IP.

Οι βασικές αρχές στις οποίες λειτουργεί η OTV είναι η χρήση πρωτοκόλλου ελέγχου για τη διαφήμιση πληροφοριών προσέγγισης διευθύνσεων MAC (αντί της χρήσης εκμάθησης του επιπέδου δεδομένων) και η μεταγωγή πακέτων της ενθυλακωμένης στην IP κίνηση Layer-2 (αντί της χρήσης μεταγωγής κυκλώματος) για διαβίβαση δεδομένων. Αυτά τα χαρακτηριστικά είναι μια σημαντική απόκλιση από τον κεντρικό μηχανισμό των παραδοσιακών VPN στο Layer-2. Στα

παραδοσιακά δίκτυα Layer-2 VPN διατηρείται ένα στατικό δίκτυο κυκλωμάτων μεταξύ όλων των συσκευών του VPN για να καταστεί δυνατή η πλημμύρα(flooding) της κίνησης και της εκμάθησης των διευθύνσεων MAC με βάση την πηγή. Αυτό το πλήρες πλέγμα των κυκλωμάτων είναι ένας απεριόριστος τομέας πλημμύρας στον οποίο μεταδίδεται ολόκληρη η κίνηση. Η διατήρηση αυτού του πλήρους πλέγματος κυκλωμάτων περιορίζει σοβαρά την επεκτασιμότητα των υφιστάμενων προσεγγίσεων Layer-2 VPN. Ταυτόχρονα, η έλλειψη ενός επιπέδου ελέγχου περιορίζει την επεκτασιμότητα των σημερινών λύσεων Layer-2 VPN για να αντιμετωπίσει σωστά τις απαιτήσεις για την επέκταση LAN σε όλα τα data-centers.

Η OTV χρησιμοποιεί ένα πρωτόκολλο ελέγχου για τη αντιστοίχιση-χαρτογράφηση των προορισμών διευθύνσεων MAC στο επόμενο IP hop και είναι προσβάσιμο μέσω του πυρήνα του δικτύου. Η OTV μπορεί να θεωρηθεί ως δρομολόγηση MAC, στην οποία ο προορισμός είναι μια διεύθυνση MAC, το επόμενο hop είναι μια διεύθυνση IP και η κυκλοφορία είναι ενσωματωμένη στην IP, ώστε να μπορεί απλά να μεταφερθεί στη MAC δρομολόγηση του επόμενου hop στο βασικό δίκτυο IP. Έτσι, μια ροή μεταξύ διευθύνσεων MAC προέλευσης και προορισμού μεταφράζεται στην επικάλυψη σε μια ροή IP μεταξύ της διεύθυνσης IP πηγής και προορισμού των σχετικών ακραίων(edge) συσκευών. Αυτή η διαδικασία ονομάζεται ενθυλάκωση και όχι σήραγγα, καθώς η ενθυλάκωση επιβάλλεται δυναμικά και οι σήραγγες δεν διατηρούνται. Δεδομένου ότι η κίνηση είναι IP-forwarded, η OTV είναι εξίσου αποτελεσματική με το βασικό δίκτυο IP και θα προσφέρει βέλτιστη εξισορρόπηση φορτίου κυκλοφορίας, αναπαραγωγή κυκλοφορίας πολλαπλών μεταδόσεων και γρήγορο failover όπως και ο πυρήνας. Η Εικόνα 2.6 απεικονίζει αυτόν τον δυναμικό μηχανισμό εγκλεισμού.




**Εικόνα 2.6 Λειτουργία OTV[16]**

Οι αντιστοιχίσεις της διεύθυνσης MAC με το επόμενο IP hop στον πίνακα προώθησης που απεικονίζεται στην Εικόνα 2.6 διαφημίζονται από ένα πρωτόκολλο ελέγχου, εξαλείφοντας έτσι την ανάγκη πλημμύρας άγνωστης κίνησης unicast σε όλη την επικάλυψη. Το πρωτόκολλο ελέγχου είναι επεκτάσιμο και περιλαμβάνει χρήσιμες πληροφορίες για τη διεύθυνση MAC όπως VLAN, αναγνωριστικό ιστότοπου και συσχετισμένη διεύθυνση IP (για κεντρικούς υπολογιστές IP). Αυτές οι πλούσιες πληροφορίες, οι περισσότερες από τις οποίες δεν είναι διαθέσιμες όταν βασίζεστε στην εκμάθηση πλημμυρών δεδομένων, συμβάλλουν στην ανάπτυξη στην OTV της απαραίτητης

ευφυΐας για την εφαρμογή πολλαπλών λειτουργιών, την ισορροπία φορτίου, την αποτροπή βρόχων, τον εντοπισμό του πρωτοκόλλου First Hop Resiliency Protocol (FHRP) και Address Resolution Protocol (ARP) χωρίς να δημιουργηθούν πρόσθετα λειτουργικά έξοδα για κάθε λειτουργία.

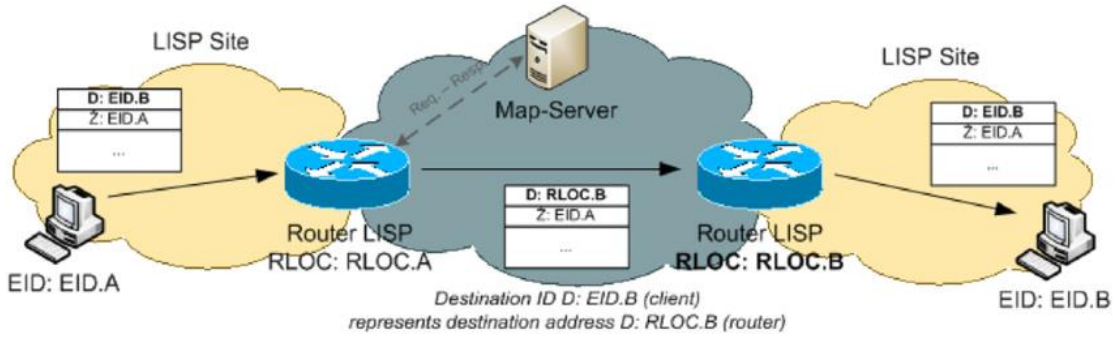
**2.4) Locator ID Separation Protocol (LISP)**

Το πρωτόκολλο διαχωρισμού θέσης/ταυτότητας είναι ένα πρωτόκολλο "map-and-encapsulate" το οποίο αναπτύσσεται από την ομάδα εργασίας IETF LISP. Η βασική ιδέα πίσω από αυτό το διαχωρισμό είναι ότι η αρχιτεκτονική του Διαδικτύου συνδυάζει δύο λειτουργίες, εντοπιστής δρομολόγησης (όπου ένας πελάτης είναι συνδεδεμένος στο δίκτυο) και αναγνωριστικό(ποιος είναι ο πελάτης) σε ένα χώρο: τη διεύθυνση IP. Το LISP υποστηρίζει το διαχωρισμό του χώρου διεύθυνσης IPv4 και IPv6 μετά από ένα σχέδιο map-and-encapsulation που βασίζεται στο δίκτυο. Στο LISP, και τα δύο αναγνωριστικά και εντοπιστές μπορούν να είναι διευθύνσεις IP ή αυθαίρετα στοιχεία όπως ένα σύνολο συντεταγμένων GPS ή μια διεύθυνση MAC.[17]

Μια βασική ιδέα του LISP είναι ότι τα τελικά συστήματα (hosts) λειτουργούν με τον ίδιο τρόπο που λειτουργούν και σήμερα. Οι διευθύνσεις IP που χρησιμοποιούν οι κεντρικοί υπολογιστές για την αποστολή και λήψη πακέτων, δεν αλλάζουν. Στην ορολογία LISP, αυτές οι διευθύνσεις IP ονομάζονται προσδιοριστές τελικών σημείων (EID).

Οι δρομολογητές συνεχίζουν να προωθούν τα πακέτα με βάση τις διευθύνσεις προορισμού IP. Όταν ένα πακέτο είναι LISP ενθυλακωμένο, αυτές οι διευθύνσεις αναφέρονται ως εντοπιστές δρομολόγησης (RLOC). Οι περισσότεροι δρομολογητές κατά μήκος μιας διαδρομής ανάμεσα σε δύο κεντρικούς υπολογιστές δεν θα αλλάξουν και συνεχίζουν να πραγματοποιούν αναζητήσεις δρομολόγησης / προώθησης στις διευθύνσεις προορισμού. Για τους δρομολογητές μεταξύ του host και του ITR καθώς και των δρομολογητών από το ETR στον κεντρικό υπολογιστή προορισμού, η διεύθυνση προορισμού είναι ένα EID. Για τους δρομολογητές μεταξύ του ITR και του ETR, η διεύθυνση προορισμού είναι RLOC.[18]

Μια άλλη βασική ιδέα LISP είναι ο "Router Tunnel". Ένας δρομολογητής σήραγγας προεξοφλεί τις κεφαλίδες LISP στα πακέτα που προέρχονται από τον κεντρικό υπολογιστή πριν από την τελική παράδοση στον προορισμό τους. Οι διευθύνσεις IP σε αυτήν την "εξωτερική κεφαλίδα" είναι RLOC. Κατά την ανταλλαγή πακέτων από άκρο σε άκρο μεταξύ δύο κεντρικών υπολογιστών του Διαδικτύου, ένα ITR προικοδοτεί μια νέα κεφαλίδα LISP σε κάθε πακέτο και ένα ETR λωρίδες στη νέα κεφαλίδα. Το ITR εκτελεί αναζητήσεις EID-to-RLOC για να καθορίσει τη διαδρομή δρομολόγησης προς το ETR, το οποίο έχει το RLOC ως μία από τις διευθύνσεις IP του.[18]

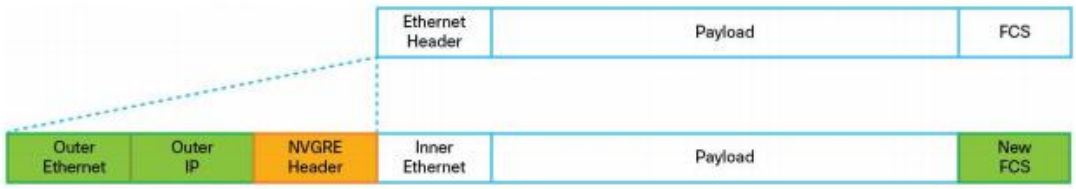



**Εικόνα 2.7 Λειτουργία του LISP[19]**

## 2.5) Network Virtualization Generic Routing Protocol (NVGRE) [14]

Η Εικονικοποίηση Δικτύων με χρήση της Ενσωμάτωσης Γενικής Δρομολόγησης (NVGRE) είναι μια τεχνολογία εικονικοποίησης δικτύου που προσπαθεί να ανακουφίσει τα προβλήματα κλιμάκωσης που σχετίζονται με τις εκτεταμένες εφαρμογές του cloud computing. Χρησιμοποιεί το GRE για να πραγματοποιήσει σήραγγα(tunnel) Layer-2 πακέτα πάνω από τα δίκτυα του Layer 3.

Όπως το VXLAN, το NVGRE υποστηρίζει αναγνωριστικό τμήματος 24 bit ή εικονικό αναγνωριστικό υποδικτύου (VSID), παρέχοντας έως και 16 εκατομμύρια εικονικά τμήματα που μπορούν να αναγνωρίσουν με μοναδικό τρόπο ένα συγκεκριμένο τμήμα ή ένα χώρο διεύθυνσης ενός συγκεκριμένου ενοικιαστή.




**Εικόνα 2.8 Ενθυλάκωση Κεφαλίδας NVGRE[14]**

Τα τελικά σημεία(end-points) του NVGRE είναι υπεύθυνα για την προσθήκη ή την αφαίρεση της εγκατάστασης NVGRE και μπορούν να υπάρχουν σε μια συσκευή δικτύου ή σε ένα φυσικό διακομιστή. Τα τελικά σημεία της NVGRE εκτελούν λειτουργίες παρόμοιες με αυτές που εκτελούνται από το VTEP σε περιβάλλον VXLAN και είναι επίσης υπεύθυνες για την εφαρμογή οποιασδήποτε σημασιολογίας Layer 2 και για την εφαρμογή πολιτικών απομόνωσης βασισμένων στο VSID.

Μια κύρια διαφορά μεταξύ του VXLAN και του NVGRE είναι ότι η κεφαλίδα NVGRE περιλαμβάνει ένα προαιρετικό πεδίο ID της ροής. Σε τοποθετήσεις πολλαπλών διαδρομών, δρομολογητές δικτύου και μεταγωγείς δικτύου που μπορούν να αναλύσουν αυτήν την κεφαλίδα, μπορούν να χρησιμοποιήσουν αυτό το πεδίο μαζί με το VSID για να προσθέσουν εντροπία με βάση τη ροή, αν και αυτή η δυνατότητα απαιτεί πρόσθετες δυνατότητες υλικού.

Όπως συμβαίνει με το VXLAN, το πρότυπο NVGRE δεν καθορίζει μια μέθοδο για την ανακάλυψη της προσβασιμότητας του τελικού σημείου. Αντίθετα, προτείνει ότι αυτές οι πληροφορίες μπορούν να παρέχονται μέσω ενός επιπέδου διαχείρισης ή να λαμβάνονται μέσω ενός συνδυασμού προσεγγίσεων μάθησης της κατανομής του επιπέδου ελέγχου ή της μάθησης των επιπέδων δεδομένων.

**2.6)Stateless Transport Tunneling(STT)[14]**

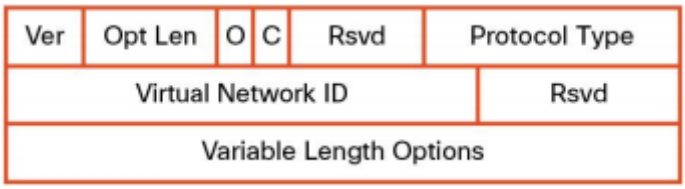
Η χωριστή σήραγγα μεταφοράς (STT) είναι ένα σχήμα ενθυλάκωσης-επικάλυψης πάνω από δίκτυα Layer-3 που χρησιμοποιούν μια κεφαλίδα τύπου TCP μέσα στην κεφαλίδα IP. Η χρήση πεδίων TCP έχει προταθεί για την παροχή συμβατότητας με τις υπάρχουσες υλοποιήσεις των NIC ώστε να καταστεί δυνατή η απομάκρυνση της λογικής και επομένως το STT είναι ιδιαίτερα χρήσιμο για εφαρμογές που είναι τελικά συστήματα στόχων(όπως οι εικονικοί μεταγωγείς σε φυσικούς διακομιστές). Όπως υποδηλώνει το όνομα, τα πεδία TCP δεν χρησιμοποιούν καμία κατάσταση σύνδεσης TCP.

Μία περιοχή που αντιμετωπίζει ειδικά το STT είναι η αναντιστοιχία μεγέθους μεταξύ πλαισίων Ethernet και της μέγιστη μονάδα μεταφοράς (MTU) που υποστηρίζεται από το υποκείμενο φυσικό δίκτυο. Τα περισσότερα λειτουργικά συστήματα end-host σήμερα ορίζουν το MTU σε μικρό μέγεθος έτσι ώστε ολόκληρο το πλαίσιο, καθώς και κάθε πρόσθετη (overlay) ενθυλάκωση να μπορούν να μεταφερθούν μέσω του φυσικού δικτύου. Αυτή η ρύθμιση μπορεί να οδηγήσει σε δυνητική υποβάθμιση της απόδοσης και πρόσθετη επιβάρυνση σε σύγκριση με πλαίσια που μπορούν να μεταδοθούν με το επιθυμητό μέγιστο μέγεθος τμήματος (MSS). Το STT επιδιώκει να εκμεταλλευτεί σήμερα τις ικανότητες απομακρυσμένης τοποθέτησης TCP (TSO) που έχουν ενσωματωθεί σε πολλά NICs σήμερα, ώστε να επιτρέψει την κατακερματισμό πλαισίων με τις κατάλληλες κεφαλίδες διευθύνσεων TCP, IP και MAC καθώς και την επανασυναρμολόγηση αυτών των τμημάτων στην πλευρά λήψης.

Παρόμοια με άλλες ενθυλακώσεις, το STT περιέχει ένα εικονικό αναγνωριστικό δικτύου που χρησιμοποιείται για την προώθηση του πλαισίου στο σωστό περιβάλλον του εικονικού δικτύου. Αυτό το αναγνωριστικό περιέχεται σε ένα πεδίο αναγνωριστικού πλαισίου 64-bit και έχει μεγαλύτερο χώρο για την αντιμετώπιση ποικίλων μοντέλων υπηρεσιών και για μελλοντική επέκταση.

**2.7) Generic Encapsulation Virtualization Network (Geneve) [20]**

Το πρωτόκολλο Generic Encapsulation Virtualization Network (Geneve) προσφέρει μια άλλη προσέγγιση στην ενθυλάκωση που έχει σχεδιαστεί για να προσφέρει ανεξαρτησία μεταξύ των επιπέδων ελέγχου μεταξύ των τελικών σημείων της σήραγγας. Το πρωτόκολλο καθορίζει μόνο ένα σχήμα επιπέδου δεδομένων που χρησιμοποιεί έναν αριθμό επιλογών μεταβλητού μήκους. Κάθε πακέτο περιέχει μια παράμετρο μήκους επιλογής και η ερμηνεία των διαφορετικών επιλογών είναι εκτός των προδιαγραφών. Το σχήμα 1 δείχνει την επικεφαλίδα της Geneve.




**Εικόνα 2.9 Κεφαλίδα Πρωτοκόλλου GENEVE**

Το Geneve σχεδιάστηκε για να παρέχει μέγιστη ευελιξία σε διαφορετικές εφαρμογές του επιπέδου ελέγχου και περιπτώσεις χρήσης. Επιτρέπει ουσιαστικά την κωδικοποίηση οποιασδήποτε πληροφορίας σε ένα πακέτο και τη μετάβαση μεταξύ τελικών σημείων της σήραγγας. Το Geneve είναι ένα σχέδιο IETF που προτείνεται από τις εταιρείες VMware, Microsoft, Red Hat και Intel.

### 2.8) VXLAN- GPE (VXLAN Generic Protocol Extension) και Network Service Header (NSH)

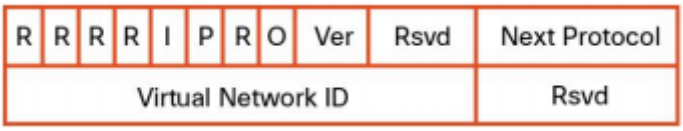
Η ομάδα εργασίας της IETF ασχολείται τα τελευταία χρόνια με δύο άλλες προσεγγίσεις ενσωμάτωσης που θα χρειαστούν στην υλοποίηση μας με την τεχνική SFC: την Επέκταση γενικού πρωτοκόλλου VXLAN (VXLAN-GPE) και την Κεφαλίδα Υπηρεσίας Δικτύου (NSH). Αυτά τα πρωτόκολλα αφορούν μια από τις συνήθεις περιπτώσεις χρήσης στις οποίες μπορεί ενδεχομένως να εφαρμοστεί η Geneve αλλά το κάνουν με μια λιγότερο γενική προσέγγιση, για την οποία είναι ευκολότερο να σχεδιαστούν πλατφόρμες υλικού και να συμβάλει στη διασφάλιση της διαλειτουργικότητας πολλών παραγόντων.[20]

Η επικεφαλίδα VXLAN δεν προσδιορίζει το πρωτόκολλο που είναι ενθυλακωμένο και ως εκ τούτου περιορίζεται επί του παρόντος στην ενσωμάτωση μόνο του ωφέλιμου φορτίου πλαισίου Ethernet, ούτε παρέχει τη δυνατότητα ορισμού πρωτοκόλλων Operations, Administration και Maintenance(OAM). Επιπλέον, οι νέες μεταφορές δεν χρειάζεται να χρησιμοποιούν αριθμούς θυρών στρώματος μεταφοράς για να προσδιορίσουν το ωφέλιμο φορτίο της σήραγγας, αλλά ενθαρρύνουν τις ενθυλακώσεις να χρησιμοποιούν τα δικά τους αναγνωριστικά στοιχεία για το σκοπό αυτό. Το VxLAN-GPE προορίζεται για την επέκταση του υπάρχοντος πρωτοκόλλου VXLAN ώστε να παρέχει OAM δυνατότητες.[20]

Τα ακόλουθα είναι τα κύρια χαρακτηριστικά της σήραγγας VxLAN GPE[21]:

• Ενσωματώνει τα πακέτα στρώματος-3 απευθείας σε μια σήραγγα VxLAN χωρίς απαιτήσεις ή εξαρτήσεις για το πεδίο-γέφυρα-2.
• Παρέχει οφέλη για την εντροπία πολλαπλών διαδρομών (ECMP) με ίσο κόστος στο βασικό δίκτυο, υπολογίζοντας τη θύρα UDP εξωτερικής πηγής με βάση το εσωτερικό πρωτόκολλο IP, τις διευθύνσεις IP προέλευσης ή προορισμού και τους αριθμούς θύρας L4 (5 tuple) .
• Χρησιμοποιεί το πρότυπο IETF VxLAN-GPE, το οποίο παρέχει άμεσες επιλογές πρωτόκολλων ανώτερου στρώματος, όπως IPv4, IPv6, Ethernet (MAC), Network Service Header (NSH) χωρίς κεφαλίδα Layer-2.

• Η ενσωμάτωση επικάλυψης υποστηρίζεται τόσο για IPv4 όσο και για IPv6, ενώ η ενσωμάτωση υποστρώματος υποστηρίζεται μόνο για IPv4.
• Υποστηρίζονται διεπαφές 8K Tunnel με λειτουργία VxLAN GPE ή VxLAN dummy-L2.
• Το 3-tuple hash είναι ενεργοποιημένο για τη δημιουργία θύρας προέλευσης UDP και για τις δύο σήραγγες VxLAN GPE και VxLAN Dummy-L2 για όλα τα πακέτα.
• Το hash θύρας προέλευσης UDP χρησιμοποιεί 3 πλειάδες για θραύσματα και 5 πλειάδες για μη θραύσματα.




**Εικόνα 2.10 Κεφαλίδα VXLAN-GPE**

Η Κεφαλίδα Υπηρεσίας Δικτύου(Network Service Header) καθορίστηκε για να προσφέρει έναν μηχανισμό για την αναγνώριση της διαδρομής υπηρεσίας δικτύου και τη μετάδοση μεταδεδομένων (Εικόνα 2.11). Ο στόχος της NSH είναι να δημιουργηθεί ένας ανεξάρτητος από την τοπολογία τρόπος καθορισμού μιας διαδρομής εξυπηρέτησης. Το NSH περιλαμβάνει επίσης έναν αριθμό υποχρεωτικών κεφαλίδων πλαισίου σταθερού μεγέθους που έχουν σχεδιαστεί για τη συλλογή πληροφοριών πλατφόρμας δικτύου. Το NSH περιέχει ακόμα ένα προαιρετικό πεδίο μεταδεδομένων μεταβλητού μήκους για επιπλέον επεκτασιμότητα. Ωστόσο, σε αντίθεση με το Geneve, το NSH σχεδιάστηκε έτσι ώστε να περιλαμβάνει όλες τις απαιτούμενες πληροφορίες μέσα στα πεδία σταθερού μεγέθους. Αν και το NSH λειτουργεί καλά με VXLAN-GPE, έχει σχεδιαστεί για πολλαπλές ενθυλακώσεις (VXLAN, LISP, MPLS κτλ.).[20]


**Εικόνα 2.11 Κεφαλίδα NSH[8]**

Στην δεύτερη υλοποίηση της εργασίας η τεχνολογία αυτή θα χρησιμοποιηθεί για την επικοινωνία μεταξύ κάποιων στοιχείων της αρχιτεκτονικής SFC.

# Κεφάλαιο 3ο

# Μεταγωγείς Openflow

Οι μεταγωγείς Openflow και συγκεκριμένα ο OpenvSwitch(OVS) αποτελούν βασικό κομμάτι της εργασίας τόσο για την υλοποίηση του VXLAN overlay όσο και για την υποστήριξη διάφορων λειτουργιών στην αρχιτεκτονική του SFC. Γι' αυτό είναι αναγκαία μια περιγραφή της λειτουργίας και των δυνατοτήτων που μας παρέχουν.

## 3.1)Λειτουργία και βασικά στοιχεία του Μεταγωγέα Openflow

### 3.1.1) Γενικά στοιχεία

Σε γενικές γραμμές μια συσκευή προώθησης(μεταγωγέας) που υποστηρίζει το πρωτόκολλο OpenFlow αποτελείται από τουλάχιστον τρία μέρη:

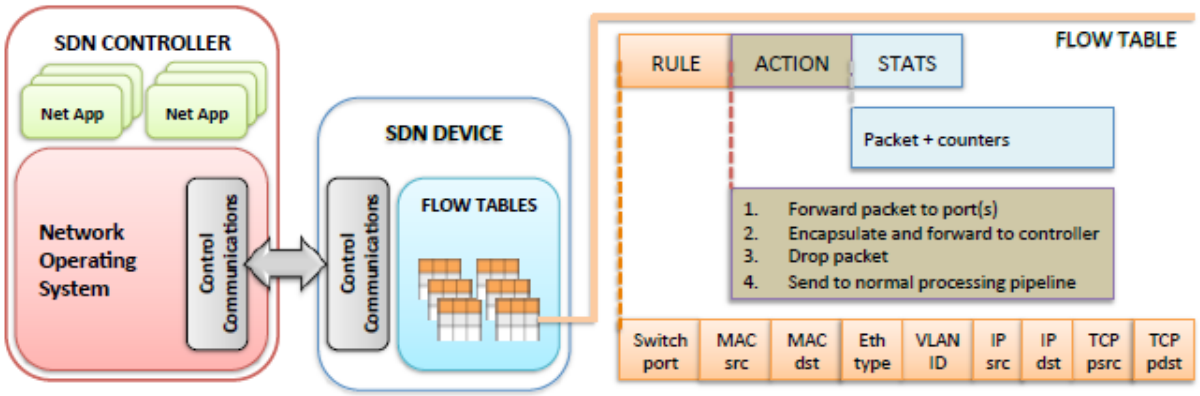
- Ένα πίνακα ροών(flows), με μια ενέργεια(action) που σχετίζεται με κάθε είσοδο ροής, για να κατευθύνει τον μεταγωγέα πώς να επεξεργάζεται τη ροή
- Ένα ασφαλές κανάλι(secure channel) που συνδέει τον μεταγωγέα με μια διαδικασία τηλεχειρισμού (τον ελεγκτή SDN), επιτρέποντας την αποστολή εντολών και πακέτων μεταξύ ενός ελεγκτή και του μεταγωγέα.
- Το πρωτόκολλο OpenFlow, το οποίο παρέχει έναν ανοικτό και τυποποιημένο τρόπο επικοινωνίας ενός ελεγκτή με ένα διακόπτη.

Ορίζοντας μια τυποποιημένη διεπαφή (το πρωτόκολλο OpenFlow) μέσω του οποίου μπορούν να οριστούν εξωτερικά οι καταχωρήσεις στον πίνακα ροών, ο μεταγωγέας OpenFlow δέχεται απομακρυσμένα τις αναγκαίες εντολές για την ρύθμιση της κίνησης.

### 3.1.2) Λειτουργία συσκευών Openflow[9][22]

Στη αρχιτεκτονική SDN/OpenFlow, όπως προαναφέρθηκε υπάρχουν δύο κύρια στοιχεία, οι ελεγκτές και οι συσκευές προώθησης, όπως φαίνεται στην Εικόνα 3.1. Μια συσκευή επιπέδου δεδομένων είναι ένα στοιχείο υλικού ή λογισμικού που ειδικεύεται στην προώθηση πακέτων, ενώ ένας ελεγκτής είναι μια στοίβα λογισμικού που τρέχει σε έναν εξυπηρετητή ή μια εικονική μηχανή. Μια συσκευή προώθησης με δυνατότητα OpenFlow βασίζεται σε ένα pipeline πινάκων ροής όπου κάθε είσοδος ενός πίνακα ροής έχει τρία μέρη[9]:

- έναν κανόνα αντιστοίχισης,
- τις ενέργειες που πρέπει να εκτελεστούν σε αντιστοιχούν πακέτα και
- τους μετρητές που διατηρούν στατιστικά στοιχεία αντιστοίχισης πακέτων.




**Εικόνα 3.1 Λειτουργία συσκευών που υποστηρίζουν Openflow[9]**

Αυτό το υψηλού επιπέδου και απλοποιημένο μοντέλο που προέρχεται από το OpenFlow είναι σήμερα ο πιο διαδεδομένος σχεδιασμός των συσκευών επιπέδου δεδομένων στα SDN. Παρόλα αυτά, ακολουθούνται και άλλες προδιαγραφές των συσκευών προώθησης με δυνατότητα SDN, τα οποία δεν θα μελετηθούν στην παρούσα εργασία, όπως τα Negotiable Datapath Models (NDMs) από την ομάδα εργασίας ONF Forwarding Abstractions.

Μέσα σε μια συσκευή OpenFlow, μια διαδρομή(path) μέσω μιας σειράς πινάκων ροής(flow-tables) καθορίζει τον τρόπο χειρισμού των πακέτων. Όταν φτάσει ένα νέο πακέτο, η διαδικασία lookup ξεκινά στον πρώτο πίνακα και τελειώνει είτε με μια αντιστοίχιση σε έναν από τους πίνακες του pipeline είτε με ένα κενό(miss) όταν δεν υπάρχει κανένας κανόνας για το συγκεκριμένο πακέτο. Ένας κανόνας ροής(flow-rule) μπορεί να οριστεί συνδυάζοντας διαφορετικά πεδία αντιστοίχισης, όπως απεικονίζεται στην Εικόνα 3.1. Εάν δεν υπάρχει κανένας προεπιλεγμένος κανόνας, το πακέτο θα απορριφθεί. Ωστόσο, η συνηθισμένη περίπτωση είναι η εγκατάσταση ενός προεπιλεγμένου κανόνα ο οποίος να «λέει» στον μεταγωγέα να στείλει το πακέτο στον ελεγκτή. Η προτεραιότητα των κανόνων ακολουθεί τον φυσικό αριθμό ακολουθίας των πινάκων και τη σειρά των γραμμών σε έναν πίνακα ροής. Οι πιθανές ενέργειες περιλαμβάνουν

- την προώθηση του πακέτου στις εξερχόμενες θύρες
- την ενθυλάκωση του και την προώθηση του στον ελεγκτή
- την απόρριψή του
- την αποστολή του στον κανονικό αγωγό επεξεργασίας
- την αποστολή του στον επόμενο πίνακα ροής ή σε ειδικούς πίνακες, όπως πίνακες ομαδοποίησης ή μέτρησης που έχουν εισαχθεί στο πιο πρόσφατο πρωτόκολλο OpenFlow.

Σε κάθε έκδοση του OpenFlow παρουσιάστηκαν νέα πεδία αντιστοίχισης όπως τα Ethernet, IPv4 / v6, MPLS, TCP/UDP κλπ. Ωστόσο, μόνο ένα υποσύνολο αυτών των αντιστοίχων πεδίων είναι υποχρεωτικό να είναι συμβατό με μια συγκεκριμένη έκδοση πρωτοκόλλου. Ομοίως, πολλές ενέργειες και τύποι θυρών είναι προαιρετικά χαρακτηριστικά. Οι κανόνες αντιστοίχισης ροής μπορούν να βασίζονται σε σχεδόν αυθαίρετους συνδυασμούς δυαδικών ψηφίων των διαφορετικών κεφαλίδων των πακέτων χρησιμοποιώντας μάσκες bit(bit masks) για κάθε πεδίο. Η προσθήκη νέων πεδίων αντιστοίχισης διευκολύνθηκε με τις δυνατότητες επεκτασιμότητας που εισήχθησαν στο OpenFlow στην έκδοση 1.2 μέσω ενός OpenFlow Extensible Match(OXM) με βάση

τις δομές type-length-value(TLV). Για να βελτιωθεί η συνολική επεκτασιμότητα πρωτοκόλλου, με τις δομές TFV 1.4 OpenFlow έχουν επίσης προστεθεί σε θύρες, πίνακες και ουρές σε αντικατάσταση των αντιστοίχων σκληρών κωδικοποιημένων παλαιότερων εκδόσεων πρωτοκόλλου.[9]

| OpenFlow Version | Match fields | Statistics | # Matches | | # Instructions | | # Actions | | # Ports | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Req | Opt | Req | Opt | Req | Opt | Req | Opt |
| v 1.0 | Ingress Port | Per table statistics | 18 | 2 | 1 | 0 | 2 | 11 | 6 | 2 |
| | Ethernet: src, dst, type, VLAN | Per flow statistics | | | | | | | | |
| | IPv4: src, dst, proto, ToS | Per port statistics | | | | | | | | |
| | TCP/UDP: src port, dst port | Per queue statistics | | | | | | | | |
| v 1.1 | Metadata, SCTP, VLAN tagging | Group statistics | 23 | 2 | 0 | 0 | 3 | 28 | 5 | 3 |
| | MPLS: label, traffic class | Action bucket statistics | | | | | | | | |
| v 1.2 | OpenFlow Extensible Match (OXM) | | 14 | 18 | 2 | 3 | 2 | 49 | 5 | 3 |
| | IPv6: src, dst, flow label, ICMPv6 | | | | | | | | | |
| v 1.3 | PBB, IPv6 Extension Headers | Per-flow meter | 14 | 26 | 2 | 4 | 2 | 56 | 5 | 3 |
| | | Per-flow meter band | | | | | | | | |
| v 1.4 | — | — | **14** | **27** | **2** | **4** | **2** | **57** | **5** | **3** |
| | | Optical port properties | | | | | | | | |

**Εικόνα 3.2 Προσθήκες σε κάθε έκδοση του Openflow[9]**

Διάφορες συσκευές προώθησης με δυνατότητα OpenFlow είναι διαθέσιμες στην αγορά, τόσο ως εμπορικά όσο και ως προϊόντα ανοικτής πηγής. Οι κατασκευαστές υλικού δικτύωσης, έχουν παραγάγει διάφορα είδη συσκευών με δυνατότητα OpenFlow. Οι συσκευές αυτές ποικίλλουν από εξοπλισμό για μικρές επιχειρήσεις (π.χ. μεταγωγείς GbE) έως εξοπλισμό κέντρων δεδομένων υψηλής ποιότητας (π.χ. πλαίσιο υψηλής συχνότητας με δυνατότητα σύνδεσης έως και 100GbE για εφαρμογές από άκρη σε πυρήνα(edge-to-core), με δεκάδες terabits ανά δευτερόλεπτο δυνατότητας μεταγωγής).

### 3.2) Εικονικός μεταγωγέας OVS(OpenvSwitch)[23]

Το OVS(OpenvSwitch) είναι μια εφαρμογή ανοικτού κώδικα ενός κατανεμημένου εικονικού μεταγωγέα πολλαπλών στρώσεων. Ο κύριος σκοπός του OVS είναι να παρέχει μια στοίβα μεταγωγής για περιβάλλοντα εικονικοποίησης υλικού, υποστηρίζοντας ταυτόχρονα πολλαπλά πρωτόκολλα και πρότυπα που χρησιμοποιούνται στα δίκτυα υπολογιστών.

Το OVS επιτρέπει την αποτελεσματική αυτοματοποίηση δικτύων μέσω προγραμματιστικών επεκτάσεων, υποστηρίζοντας ταυτόχρονα τις βασικές διεπαφές διαχείρισης και πρωτόκολλα όπως τα OpenFlow, NetFlow, sFlow, SPAN, RSPAN, CLI, LACP και 802.1ag. Επιπλέον, έχει σχεδιαστεί για να υποστηρίζει τη διαφανή διανομή σε πολλούς φυσικούς διακομιστές, επιτρέποντας τη δημιουργία cross-server μεταγωγέων μεταξύ τους με τρόπο που να μεταφέρει σε αφαιρετικό επίπεδο(abstracted level) την υποκείμενη αρχιτεκτονική εξυπηρετητών.

Ο OVS μπορεί να λειτουργήσει τόσο ως μεταγωγέας δικτύου που βασίζεται σε λογισμικό που τρέχει μέσα σε έναν hypervisor εικονικής μηχανής (VM), όσο και ως στοίβα ελέγχου για αποκλειστικό εξοπλισμό μεταγωγής. Ως αποτέλεσμα, έχει μεταφερθεί σε πολλαπλές πλατφόρμες virtualization, switching chipsets και επιταχυντές υλικού δικτύωσης. Το OpenvSwitch έχει επίσης

ενσωματωθεί σε διάφορες πλατφόρμες λογισμικού cloud computing και συστήματα διαχείρισης virtualization(όπως το OpenStack).

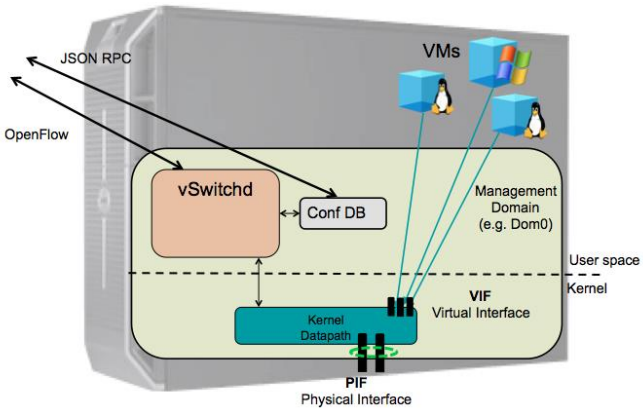
### 3.3) Το πρωτόκολλο OVSDB[23]

Το OVSDB, η βάση δεδομένων OpenvSwitch, είναι ένα πρωτόκολλο ρύθμισης του Openflow και είναι σχεδιασμένο για την διαχείριση των διάφορων υλοποιήσεων του OVS. Πιο απλά, είναι ένα σύστημα βάσης δεδομένων προσβάσιμο σε δίκτυο. Τα σχήματα στο OVSDB καθορίζουν τους πίνακες σε μια βάση δεδομένων και τους τύπους των στηλών τους και μπορούν να περιλαμβάνουν περιορισμούς δεδομένων, μοναδικότητας και αναφοράς ακεραιότητας. Το OVSDB προσφέρει ατομικές, συνεπείς, απομονωμένες, ανθεκτικές συναλλαγές. Το RFC 7047 καθορίζει το πρωτόκολλο JSON-RPC που χρησιμοποιούν οι πελάτες και οι διακομιστές OVSDB για να επικοινωνούν.

Το πρωτόκολλο OVSDB είναι κατάλληλο για συγχρονισμό κατάστασης, επειδή επιτρέπει σε κάθε πελάτη να παρακολουθεί το περιεχόμενο μιας ολόκληρης βάσης δεδομένων ή ενός υποσυνόλου της. Κάθε φορά που ένα παρακολουθούμενο τμήμα της βάσης δεδομένων αλλάζει, ο διακομιστής λέει στον πελάτη ποιες σειρές προστέθηκαν ή τροποποιήθηκαν (συμπεριλαμβανομένων των νέων περιεχομένων) ή διαγράφηκαν. Έτσι, οι πελάτες OVSDB μπορούν να παρακολουθούν εύκολα τα νεότερα περιεχόμενα οποιουδήποτε τμήματος της βάσης δεδομένων.

Η κύρια χρήση του OVSDB σε συστήματα με OVS, είναι για τη διαμόρφωση και την παρακολούθηση του ovs-vswitchd, του daemon switch open vSwitch, χρησιμοποιώντας το σχήμα που τεκμηριώνεται στο ovs-vswitchd.conf.db. Επίσης, το OVS περιλαμβάνει το σχήμα "VTEP", γιατί πολλά εξαρτήματα υλικού τρίτων υποστηρίζουν τη διαμόρφωση του VXLAN, αν και η ίδια η OVS δεν χρησιμοποιεί αυτόματα αυτό το σχήμα.

Η προδιαγραφή πρωτοκόλλου OVSDB επιτρέπει την ανάπτυξη ανεξάρτητων υλοποιήσεων του OVSDB με την δυνατότητα πολλών παράλληλων λειτουργιών. ΤΟ OVS περιλαμβάνει μια εφαρμογή διακομιστή OVSDB που ονομάζεται ovsdb-server , η οποία υποστηρίζει αρκετές επεκτάσεις πρωτοκόλλων. Στην χρήση του OVS αναφερόμαστε πολλές φορές σε αυτές τις υλοποιήσεις OVSDB ως απλά "OVSDB", αν και αυτό είναι διαφορετικό από το πρωτόκολλο OVSDB.




**Εικόνα 3.3 Το OVS εσωτερικά[23]**

Εκτός από αυτά τα γενικά εργαλεία διακομιστή και πελάτη OVSDB, το OVS περιλαμβάνει εργαλεία για εργασίες με βάσεις δεδομένων που έχουν συγκεκριμένα σχήματα όπως:

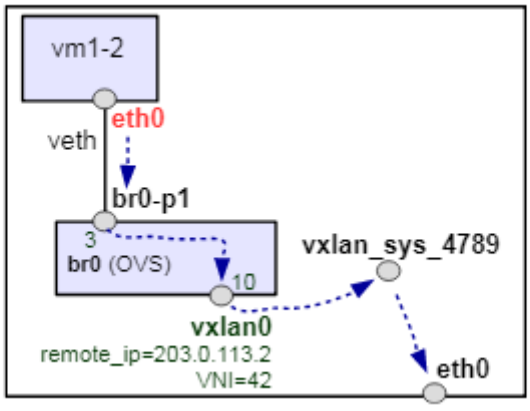
- το ovs-vsctl που λειτουργεί με τη βάση δεδομένων ρύθμισης ovs-vswitchd
- το vtep-ctl που συνεργάζεται με τη βάση δεδομένων VTEP,
- το ovn-nbctl που συνεργάζεται με τη βάση δεδομένων OVN Northbound

Τέλος, το OVS περιλαμβάνει το εργαλείο ovsdb-tool για να δουλεύει με τις δικές του μορφές βάσεων δεδομένων. Το πιο αξιοσημείωτο χαρακτηριστικό αυτής της μορφής είναι ότι το ovsdb-tool διευκολύνει τους χρήστες να βλέπουν τις ρυθμίσεις και καταχωρήσεις που έχουν αλλάξει σε μια βάση δεδομένων από την τελευταία φορά που συμπιέστηκε. Αυτή η λειτουργία είναι συχνά χρήσιμη για την αντιμετώπιση προβλημάτων.

### 3.4) Υλοποίηση VXLAN overlay σε OVS

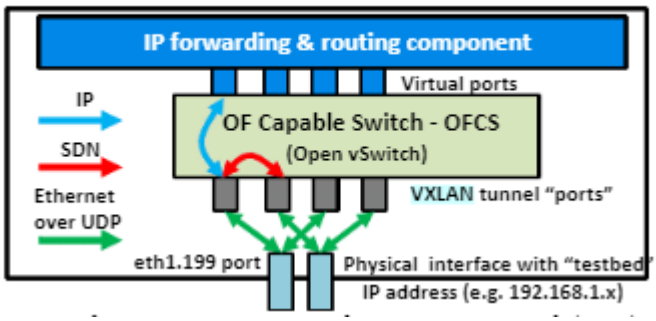
Μια πιθανή προσέγγιση για την ενίσχυση της απόδοσης ενός VXLAN overlay είναι να βασιστούμε σε συγκεκριμένο υλικό ή σε ενότητες λογισμικού σε βελτιστοποιημένη βιβλιοθήκη εισόδου/εξόδου(I/O), όπως η DPDK[24]. Σε αυτή την βάση και βλέποντας στις προηγούμενες ενότητες τις δυνατότητες που παρέχουν τα OVS/OVSDB, το OVS είναι ένας άλλος τρόπος δημιουργίας και διαχείρισης VXLAN(και άλλων) overlays μέσα σε ένα σύστημα Linux. Μια πρώτη διαφορά μεταξύ των εφαρμογών του OVS και του Kernel είναι ότι το OVS δεν υποστηρίζει multicast. Μια άλλη διαφορά είναι ότι σε συσκευές OVS με VXLAN δημιουργούνται εσωτερικοί μεταγωγείς OVS και αυτές οι συσκευές δεν είναι ορατές από το σύστημα. [24]

Το OVS υλοποιεί VXLAN tunnels στον χώρο του πυρήνα, βελτιώνοντας δραματικά τις επιδόσεις. Με την δυνατότητα υποστήριξης του Openflow, το OVS χρησιμοποιείται επίσης για την πραγματοποίηση της ενθυλάκωσης και της αποκόλλησης των σηράγγων VXLAN. Κάθε σήραγγα αντιστοιχεί σε μια θύρα στον μεταγωγεά. Για το VXLAN(όπως και για το GRE) χρησιμοποιώντας το OpenFlow, είναι δυνατό να ελέγξουμε τα πακέτα που έχουν ήδη ρυθμιστεί (και συγκεκριμένες αντιστοιχίες έχουν εισαχθεί στην έκδοση OF 1.4 του πρωτοκόλλου Openflow για VXLAN).




**Εικόνα 3.4 Παράδειγμα υλοποίησης VXLAN με OVS[25]**

Στο OVS λόγω του Openflow, μπορούμε να χρησιμοποιήσουμε την προώθηση πακέτων βάσει των ροών(flow-based forwarding) και σε συσκευές που χρησιμοποιούν VXLAN. Είναι αξιοσημείωτο το γεγονός ότι το OVS με αυτή την δυνατότητα(flow-based forwarding) που είναι πολύ πιο ευέλικτη, μας επιτρέπει πολύ πιο χαμηλό επίπεδο ελέγχου για το ποια κίνηση αποστέλλεται μέσω της VXLAN συσκευής σε σχέση με την υλοποίηση του VXLAN overlay σε Linux Kernel.[25]




**Εικόνα 3.5 Παράδειγμα OVS με θύρες VXLAN[24]**

# Κεφάλαιο 4ο

# SFC: Service function Chaining (Αλυσιδωτή Λειτουργία Υπηρεσιών)

## 4.1.Το πρόβλημα που προσπαθεί να λύσει η SFC

Οι ροές(flows) κίνησης των εφαρμογών από άκρο σε άκρο(edge-to-edge), απαιτείται να διασχίσουν διάφορες υπηρεσίες δικτύου όπως IPS/IDS, τείχη προστασίας, βελτιστοποιητές WAN και αντισταθμιστές φορτίου κατά μήκος μιας προκαθορισμένης διαδρομής. Για παράδειγμα, ένα flow ενός χρήστη που προσεγγίζει έναν διακομιστή-εξυπηρετητή εφαρμογών σε μια απομακρυσμένη τοποθεσία, ίσως χρειαστεί να περάσει από τους βελτιστοποιητές WAN για καλύτερη απόδοση, από τείχη προστασίας ή ένα IPS για ασφάλεια και τέλος από ένα διακομιστή VPN(Εικονικό Ιδιωτικό Δίκτυο) πριν φτάσει στο διακομιστή εφαρμογών.

Τα παραδοσιακά δίκτυα αντιμετωπίζουν ορισμένες ανεπάρκειες[26]:

**Περιορισμοί τοπολογίας:** Οι παραδοσιακές αρχιτεκτονικές για την ανάπτυξη υπηρεσιών δικτύου εξαρτώνται σε μεγάλο βαθμό από την τοπολογία του δικτύου. Τα στοιχεία του δικτύου συνδέονται μεταξύ τους με ένα περίπλοκο και πολύπλοκο σύνολο διαμορφώσεων για τη δημιουργία μιας διαδρομής εξυπηρέτησης. Η πραγματοποίηση αλλαγών σε αυτό το εξαιρετικά σύνθετο περιβάλλον, όπως η δυναμική εισαγωγή μιας νέας υπηρεσίας, τείνει να είναι μια πολύπλοκη προσπάθεια με υπερχρησιμοποίηση πόρων αλλά και χρονοβόρα.

**Σταθερές αλυσίδες εξυπηρέτησης:** Οι περιορισμοί τοπολογίας επιβάλλουν όλη την κυκλοφορία μέσω του πολύπλοκου συνόλου λειτουργιών εξυπηρέτησης, ακόμη και αν κομμάτι της κίνησης ενδέχεται να απαιτεί μόνο ορισμένες από τις λειτουργίες εξυπηρέτησης. Πολλές φορές μόνο τα πρώτα πακέτα μπορεί να χρειαστούν να περάσουν από ορισμένες λειτουργίες υπηρεσίας και τα επόμενα πακέτα μπορεί να τις παρακάμψουν. Αυτή η συμπεριφορά επιβάλλει περιττό κόστος και κόστος καθυστέρησης.

**Σύνθετη διαμόρφωση:** Οι τοπολογικές εξαρτήσεις καθιστούν το σύνολο της διαμόρφωσης πολύ πιο πολύπλοκο, καθιστώντας πιο δύσκολη την αντιμετώπιση προβλημάτων.
α) Συνεπής παραγγελία των λειτουργιών εξυπηρέτησης: Διασφάλιση της ροής κυκλοφορίας μέσω ενός συνόλου λειτουργιών υπηρεσιών είναι συχνά μια διαδικασία που είναι αργή και επιρρεπής σε σφάλματα.
β) Συστηματικές ροές κίνησης: Πολλές υπηρεσίες δικτύου, όπως τείχη προστασίας, απαιτούν αμφίδρομες ροές κίνησης. Οι υπάρχουσες μέθοδοι για την εξασφάλιση συμμετρικών ροών συχνά απαιτούν πολύπλοκη διαμόρφωση κάθε λειτουργίας υπηρεσίας δικτύου κατά μήκος της διαδρομής.

**Ταξινόμηση κίνησης και επιβολή πολιτικής:** Οι δυνατότητες ταξινόμησης στις περισσότερες λειτουργίες της υπηρεσίας είναι γενικές: ένας ή οποιοσδήποτε συνδυασμός παραμέτρων L1-L4

όπως την διεπαφή, το VLAN(εικονικό δίκτυο που ανήκει), την διεύθυνση IP, την θύρα κλπ. Η γενικευμένη ταξινόμηση συχνά οδηγεί σε γενικευμένη επιβολή της πολιτικής. Δηλαδή, η κίνηση που δεν απαιτεί επιβολή υπηρεσιών εξακολουθεί να πρέπει να διασχίσει τις λειτουργίες της υπηρεσίας και μπορεί να υπόκειται χωρίς επιφυλάξεις στην επιβολή πολιτικών.

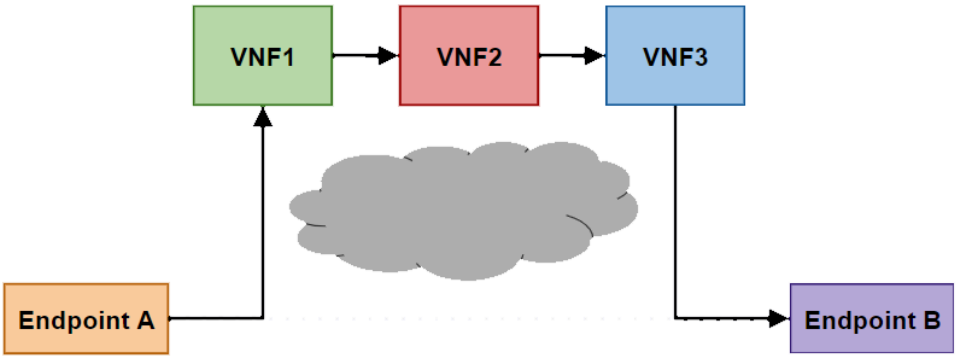
**Περιορισμένη διαθεσιμότητα:** Καθώς οι υπηρεσίες δικτύου συχνά μοιράζονται πολλές εφαρμογές, οποιαδήποτε αλλαγή στη συμπεριφορά ή στις απαιτήσεις ικανότητας μιας ροής εφαρμογών μπορεί να επηρεάσει αρνητικά τις ροές άλλων εφαρμογών που διέρχονται από την ίδια υπηρεσία.

**Πολύπλοκη κλιμακωτή αρχιτεκτονική:** Ορισμένες λειτουργίες υπηρεσιών που απαιτούν συμμετρικές ροές κίνησης ή/και σταθερές ροές, όπως τα τείχη προστασίας, είναι δύσκολο να εξαλειφθούν. Απαιτείται προσεκτικός προγραμματισμός για να προστεθούν αυτές οι υπηρεσίες σε ιδιαίτερα πολύπλοκα τοπολογικά περιβάλλοντα.

**Περιορισμένη ορατότητα για την αντιμετώπιση προβλημάτων:** Η αντιμετώπιση προβλημάτων που σχετίζονται με την υπηρεσία L4-L7 απαιτεί διαχειριστές που είναι έμπειροι στις λειτουργίες υπηρεσιών L2-L3 και L4-L7. Η αντιμετώπιση προβλημάτων σε εικονικά περιβάλλοντα που λειτουργούν σε πολλαπλά στρώματα δικτύων overlays/underlays, απαιτεί εργαλεία με καλύτερη ορατότητα(visibility) στα underlay δίκτυα, καθώς και εκείνα που μπορούν να παρέχουν προβολές(views) από άκρο σε άκρο.

Ο βαθμός πολυπλοκότητας αυξάνεται εκθετικά σε περιβάλλοντα παρόχων υπηρεσιών, όπου συνυπάρχουν αρκετοί πελάτες(tenants/clients) με τις δικές τους, διαφορετικές ανάγκες. Αυτοί μπορούν να εγγραφούν στη δική τους σειρά υπηρεσιών, οι οποίοι συχνά παρέχονται από συσκευές λειτουργικής εξυπηρέτησης που μοιράζονται μεταξύ πολλών πελατών. Ο βαθμός πολυπλοκότητας αυξάνεται ακόμα περισσότερο όταν αυτοί οι πελάτες επιθυμούν να εισάγουν, να αφαιρούν ή να ανακατευθύνουν δυναμικά την κίνηση, ανάλογα με τις πολιτικές κυκλοφορίας και ασφάλειας, καθώς και τις συμφωνίες επιπέδου υπηρεσιών.

Η αλυσιδωτή λειτουργία υπηρεσιών(SFC) με βάση τα Δίκτυα οριζόμενα από λογισμικό(SDN) χρειάζεται για να επιτρέπει λύσεις για το παραπάνω σύνολο προβλημάτων.




**Εικόνα 4.1 Συμβατική Αλυσίδα Υπηρεσιών Δικτύου[27]**

### 4.2) Γενικά στοιχεία

Η αλυσιδωτή λειτουργία υπηρεσιών (SFC), είναι μια δυνατότητα που χρησιμοποιεί τις δυνατότητες των Δικτύων ορισμένων από λογισμικό(SDN), για τη δημιουργία μιας αλυσίδας υπηρεσιών από συνδεδεμένες υπηρεσίες δικτύου (όπως firewalls L4-L7, προστασία από εισβολές) και την σύνδεσή τους σε μια εικονική αλυσίδα. Αυτή η δυνατότητα μπορεί να χρησιμοποιηθεί από τους φορείς εκμετάλλευσης δικτύων για τη δημιουργία καταλόγων συνδεδεμένων υπηρεσιών που επιτρέπουν τη χρήση μιας ενιαίας σύνδεσης δικτύου για πολλές υπηρεσίες με διαφορετικά χαρακτηριστικά.[28]

Το κύριο πλεονέκτημα της αλυσιδωτής λειτουργίας υπηρεσιών είναι η αυτοματοποίηση του τρόπου με τον οποίο μπορούν να ρυθμιστούν οι εικονικές συνδέσεις δικτύου για να διαχειριστούν τις ροές κυκλοφορίας για τις συνδεδεμένες υπηρεσίες. Για παράδειγμα, ένας ελεγκτής SDN θα μπορούσε να λάβει τις ενδεδειγμένες αλυσίδες υπηρεσιών και να τις εφαρμόσει σε διαφορετικές ροές κυκλοφορίας ανάλογα με την πηγή, τον προορισμό ή τον τύπο της κίνησης. Η SFC αυτοματοποιεί αυτό που κάνουν οι παραδοσιακοί διαχειριστές δικτύου όταν συνδέουν μια σειρά φυσικών συσκευών L4-L7, για να επεξεργάζονται την εισερχόμενη και εξερχόμενη κίνηση δικτύου, η οποία μπορεί να απαιτεί μια σειρά χειροκίνητων βημάτων.

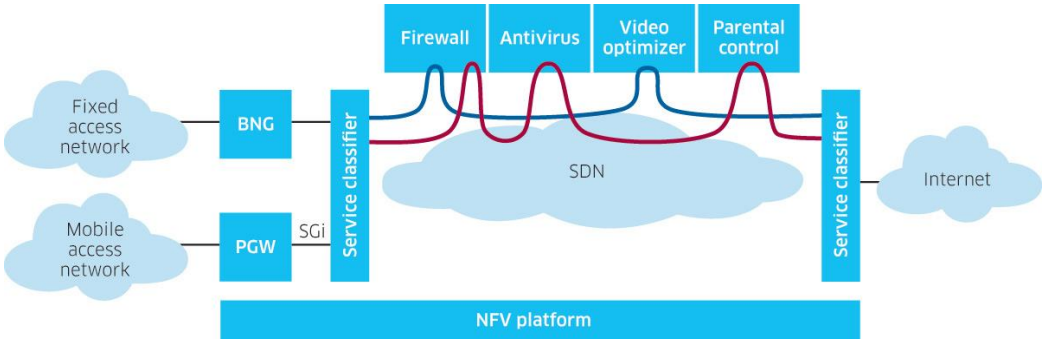
Η αλυσίδα υπηρεσιών περιλαμβάνεται σε πολλές εφαρμογές των SDN και NFV όπως η ανάπτυξη των υπολογιστικών κέντρων (συνδέοντας μεταξύ τους εικονικές ή φυσικές λειτουργίες δικτύου), δίκτυα φορέων (υπηρεσίες για S/Gi-LAN) συμπεριλαμβανομένης και της ανάπτυξης εικονικών εγκαταστάσεων πελατών [vCPE].

Η αλυσιδωτή λειτουργία υπηρεσιών μπορεί να είναι λειτουργικά ευεργετική, επιτρέποντας την αυτοματοποιημένη παροχή των εφαρμογών δικτύου που μπορεί να έχουν διαφορετικά χαρακτηριστικά. Για παράδειγμα, μια συνεδρία βίντεο ή VoIP, έχει περισσότερες απαιτήσεις από την απλή πρόσβαση στο Web. Η αυτοματοποιημένη αλυσίδα υπηρεσιών δικτύου μπορεί να επιτρέψει τη δημιουργία και τη διάσπαση αυτών των περιόδων λειτουργίας χωρίς να απαιτείται ανθρώπινη παρέμβαση. Αυτό βοηθά επίσης να εξασφαλιστεί ότι συγκεκριμένες εφαρμογές λαμβάνουν τους κατάλληλους πόρους ή χαρακτηριστικά δικτύου (εύρος ζώνης, κρυπτογράφηση, ποιότητα QoS).

Ένα άλλο πλεονέκτημα της SFC όταν χρησιμοποιείται σε συνδυασμό με το SDN, είναι η βελτιστοποίηση της χρήσης των πόρων του δικτύου και η βελτίωση της απόδοσης των εφαρμογών. Τα εργαλεία ανάλυσης SDN και επιδόσεων μπορούν να χρησιμοποιήσουν τους καλύτερους διαθέσιμους πόρους δικτύου και να βοηθήσουν να διαπραγματευτούν τα ζητήματα συμφόρησης δικτύου, με αυτοματοποιημένο τρόπο.

Η "αλυσίδα" στην αλυσίδα υπηρεσιών αντιπροσωπεύει τις υπηρεσίες που μπορούν να συνδεθούν στο δίκτυο χρησιμοποιώντας την παροχή λογισμικού. Αυτό είναι ιδιαίτερα σημαντικό στον "κόσμο" του NFV, όπου οι νέες υπηρεσίες μπορούν να παρουσιαστούν ως λογισμικό μόνο, τρέχοντας σε υλικό βασικών προϊόντων.

Οι δυνατότητες της SFC σημαίνουν ότι ένας μεγάλος αριθμός λειτουργιών εικονικού δικτύου μπορούν να συνδεθούν μαζί σε περιβάλλον NFV. Επειδή γίνεται στο λογισμικό που χρησιμοποιεί εικονικούς μεταγωγείς αυτές οι συνδέσεις μπορούν να ρυθμιστούν όπως απαιτείται, με την παροχή αλυσίδας υπηρεσιών μέσω του στρώματος ενορχηστρώσης NFV.




**Εικόνα 4.2 Παράδειγμα SFC σε δίκτυα κινητών τηλεφώνων[28]**

### 4.3)Στοιχεία αρχιτεκτονικής SFC[26][29]

Είναι χρήσιμο πριν προχωρήσουμε στην εξήγηση της αρχιτεκτονικής SFC να ορίσουμε κάποια βασικά στοιχεία:

- **Υπηρεσία δικτύου(Network Service):** Μια “προσφορά” που παρέχεται από έναν πάροχο(operator) που παραδίδεται χρησιμοποιώντας μία ή περισσότερες λειτουργίες υπηρεσιών. Αυτό μπορεί επίσης να αναφέρεται ως "σύνθετη υπηρεσία". Ο όρος "υπηρεσία" χρησιμοποιείται για να δηλώσει μια "υπηρεσία δικτύου" στο πλαίσιο του SFC. Γενικώς, ο όρος "υπηρεσία" είναι υπερφορτωμένος με διαφορετικούς ορισμούς. Για παράδειγμα, σε κάποια υπηρεσία υπάρχει μια προσφορά που αποτελείται από πολλά στοιχεία εντός του δικτύου του φορέα εκμετάλλευσης, ενώ σε άλλες περιπτώσεις μια υπηρεσία ή πιο συγκεκριμένα μια υπηρεσία δικτύου είναι ένα διακριτό στοιχείο όπως ένα "τείχος προστασίας".

- **Κατηγοριοποιητής/ταξινομητής SFC(SFC classifier):** Μια οντότητα που ταξινομεί τις ροές για την αλυσιδωτή εξυπηρέτηση σύμφωνα με τους κανόνες ταξινόμησης που ορίζονται στον Πίνακα Πολιτικής SFC στον ελεγκτή SDN. Τα πλαίσια/πακέτα των ροών, σημειώνονται με τον αντίστοιχο αναγνωριστικό αλυσίδας SF. Ο ταξινομητής SFC μπορεί να λειτουργεί σε μια ανεξάρτητη(φυσική ή εικονική) πλατφόρμα, να βρίσκεται σε μια διαδρομή δεδομένων και επίσης να εκτελείται ως εφαρμογή πάνω από έναν ελεγκτή δικτύου.

- **Αλυσίδα λειτουργίας υπηρεσιών(SFC):** Είναι μια λίστα με παραγγελίες των οντοτήτων λειτουργιών εξυπηρέτησης. Μια αλυσίδα λειτουργιών υπηρεσίας ορίζει μια σειρά από λειτουργίες υπηρεσίας και περιορισμούς παραγγελίας που πρέπει να εφαρμόζονται σε πακέτα/πλαίσια/ροές που έχουν επιλεγεί ως αποτέλεσμα ταξινόμησης. Ένα παράδειγμα μιας λειτουργίας υπηρεσίας είναι ένα τείχος προστασίας. Η σιωπηρή εντολή ενδέχεται να μην είναι γραμμική, καθώς η αρχιτεκτονική επιτρέπει σε SFC που αντιγράφουν σε

περισσότερους από έναν κλάδους και επιτρέπει επίσης περιπτώσεις όπου υπάρχει ευελιξία με τη σειρά με την οποία πρέπει να εφαρμόζονται λειτουργίες υπηρεσιών.

- **Λειτουργία υπηρεσίας (SF)**: Λειτουργία δικτύου που παρέχει υπηρεσία προστιθέμενης αξίας στις ροές κίνησης. Μια λειτουργία υπηρεσίας μπορεί να εκτελεί τις λειτουργίες της σε ένα ή περισσότερα Layers. Οι λειτουργίες της υπηρεσίας μπορεί να είναι ένα τείχος προστασίας(firewall), η ανάλυση πακέτων(Deep Packet Inspection), η μετάφραση διεύθυνσης δικτύου(NAT), η λειτουργία εμπλουτισμού κεφαλίδων HTTP, η βελτιστοποίηση TCP, ένας εξισορροπητής φορτίου(load balancer), όπως εξηγείτε στο κεφάλαιο 1 στις VNFs.

- **Προωθητής Λειτουργιών Υπηρεσίας (Service Function Forwarder - SFF):** είναι υπεύθυνος για την προώθηση της κυκλοφορίας σε μία ή περισσότερες συνδεδεμένες λειτουργίες υπηρεσίας σύμφωνα με τις πληροφορίες που μεταφέρονται κατά την ενθυλάκωση SFC, καθώς και τον χειρισμό της κυκλοφορίας που επιστρέφει από τις λειτουργίες υπηρεσίας. Επιπλέον, ένας SFF είναι υπεύθυνος για την παράδοση κίνησης σε έναν Classifier όταν αυτό απαιτείται και υποστηρίζεται, μεταφέροντας την κυκλοφορία σε ένα άλλο SFF (στον ίδιο ή διαφορετικό τύπο επικάλυψης) και τερματίζοντας το Service Function Path (SFP).

- **Διαδρομή λειτουργίας υπηρεσίας (Service Function Path):** Η διαδρομή λειτουργίας υπηρεσίας είναι μια περιορισμένη εξειδίκευση του πού πρέπει να πάει τα πακέτα που έχουν εκχωρηθεί σε μια συγκεκριμένη διαδρομή λειτουργίας. Ενώ μπορεί να είναι τόσο περιορισμένη ώστε να εντοπίζει τις ακριβείς τοποθεσίες, μπορεί επίσης να είναι λιγότερο συγκεκριμένη. Η SFP παρέχει ένα επίπεδο διεύθυνσης μεταξύ της αλυσίδας εξυπηρέτησης ως ακολουθία λειτουργιών εξυπηρέτησης, που πρέπει να παραδοθούν και ποια SFFs/SFs το πακέτο θα επισκέπτεται όταν πραγματικά διασχίζει το δίκτυο. Επιτρέποντας στον ελεγκτή SDN να καθορίσει αυτό το επίπεδο έμμεσης εστίασης, ο χειριστής μπορεί να ελέγχει τον βαθμό της αρχής επιλογής SFF/SF που μεταβιβάζεται στο δίκτυο.

- **Ενθυλάκωση SFC(SFC encapsulation)**: Η ενθυλάκωση SFC παρέχει, ταυτοποίηση των SFP και χρησιμοποιείται από τις λειτουργίες SFC-aware(που υποστηρίζουν την διαδικασία SFC), όπως το SFF και το SFC-aware SFs. Η ενθυλάκωση του SFC δεν χρησιμοποιείται για την προώθηση πακέτων δικτύου. Εκτός από την αναγνώριση SFP, η ενθυλάκωση SFC μεταφέρει μεταδεδομένα, συμπεριλαμβανομένων των πληροφοριών πλαισίου δεδομένων.

- **Rendered Service Path (RSP)**: Σε ένα SFP, τα ίδια τα πακέτα μεταδίδονται φυσικά από και προς συγκεκριμένα σημεία του δικτύου, διασχίζοντας μια συγκεκριμένη σειρά SFFs και SFs. Αυτή η ακολουθία πραγματικών επισκέψεων από ένα πακέτο σε συγκεκριμένα SFFs και SFs στο δίκτυο είναι γνωστή ως το Rendered Service Path (RSP).

- **Κεφαλίδα αλυσίδας λειτουργίας υπηρεσιών(SFC Header)**: Μια κεφαλίδα που είναι ενσωματωμένη στο πακέτο ροής από τον Classifier για να διευκολύνει την προώθηση των πακέτων ροής κατά μήκος της διαδρομής της αλυσίδας λειτουργίας. Αυτή η κεφαλίδα

επιτρέπει επίσης τη μεταφορά μεταδεδομένων για την υποστήριξη διαφόρων λειτουργιών που σχετίζονται με την αλυσίδα υπηρεσιών.

- **Κόμβος λειτουργίας υπηρεσιών(SF Node):** Δηλώνει οποιονδήποτε κόμβο εντός ενός τομέα με δυνατότητα SFC που ενσωματώνει ένα ή πολλαπλά SF.

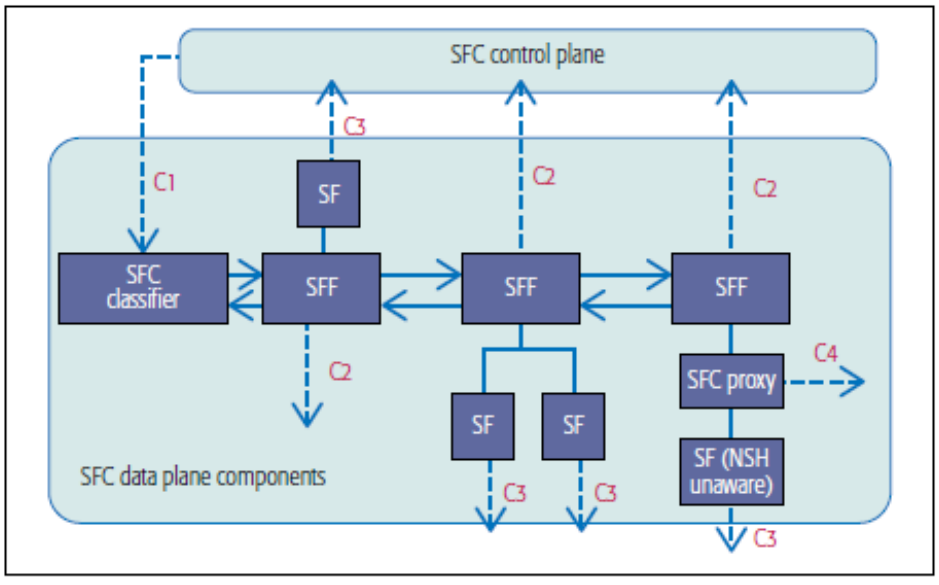
- **Proxy Λειτουργίας Υπηρεσιών(SF Proxy):** Καταργεί και εισάγει ενθυλάκωση SFC για λογαριασμό μιας υπηρεσίας που είναι SFC-unaware(δεν υποστηρίζει ενθυλάκωση). Τα Proxy Λειτουργίας Υπηρεσιών είναι λογικά στοιχεία.

- **Μεταδεδομένα(metadata):** Παρέχουν πληροφορίες με βάση τα contexts για τα πακέτα δεδομένων που διασχίζουν μια αλυσίδα υπηρεσιών. Τα μεταδεδομένα μπορούν να χρησιμοποιηθούν για τη μετάδοση πληροφοριών σχετικών με τα συμφραζόμενα(context) που δεν είναι διαθέσιμα σε μια θέση στο δίκτυο σε άλλη τοποθεσία στο δίκτυο όπου αυτές οι πληροφορίες δεν είναι άμεσα διαθέσιμες. Ενώ προορίζονται κυρίως για χρησιμοποίηση από SF, τα μεταδεδομένα μπορούν επίσης να ερμηνεύονται από άλλες οντότητες SFC.

- **Περιοχή με δυνατότητα υλοποίησης αλυσίδας λειτουργίας υπηρεσιών(SFC-enabled domain):** Δηλώνει ένα δίκτυο (ή μια περιοχή αυτού) που υλοποιεί το SFC.

### 4.4) Αρχιτεκτονική και λειτουργία SFC[30]

Με βάση τις προδιαγραφές της ομάδας εργασίας της IETF για την αρχιτεκτονική της SFC (draftietf-sfc-control plane-06), μια τυπική αρχιτεκτονική SFC βασισμένη σε SDN αποτελείται από στοιχεία που ομαδοποιούνται σε δύο επίπεδα, το επίπεδο ελέγχου και το επίπεδο δεδομένων, όπως φαίνεται στην Εικόνα 4.3.




**Εικόνα 4.3 IETF SFC Αρχιτεκτονική[30]**

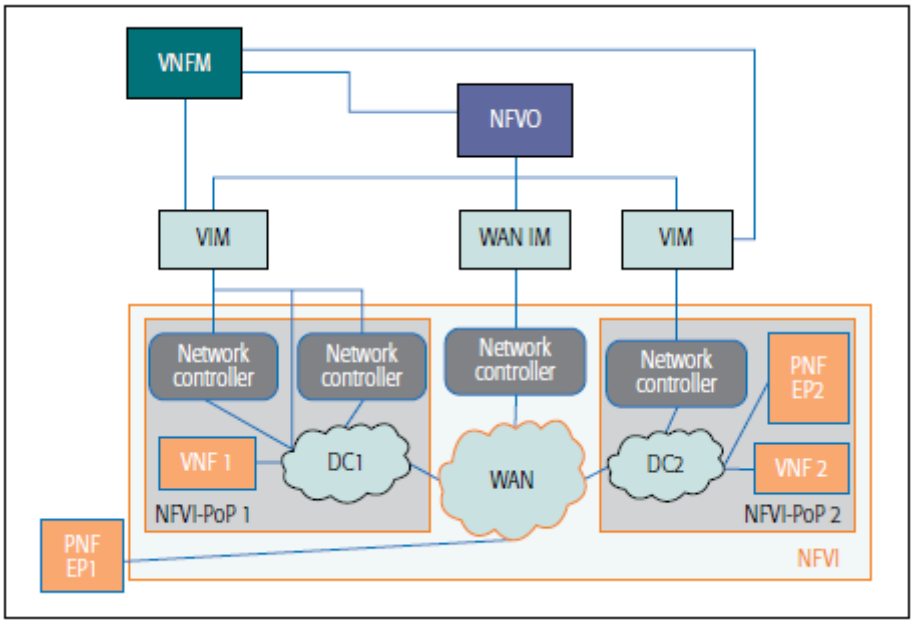
Το επίπεδο ελέγχου έχει ως αρμοδιότητα την διαχείριση της SFC(Αλυσίδας Λειτουργίας Υπηρεσιών), την διαχείριση των SF, την αντιστοίχιση μιας SFC σε ένα συγκεκριμένο SFP, την εγκατάσταση και τη διαχείριση των κανόνων προώθησης του προωθητή λειτουργίας υπηρεσιών. Τα στοιχεία του επιπέδου ελέγχου SFC αλληλλεπιδρούν με τα στοιχεία του επιπέδου δεδομένων SFC μέσω τεσσάρων διεπαφών. Η πρώτη διεπαφή C1 είναι υπεύθυνη για την προώθηση των κανόνων ταξινόμησης SFC που ορίζονται από το επίπεδο ελέγχου SFC στους καταταγμένους SFC. Τα SFFs αναφέρουν την κατάσταση συνδεσιμότητας των προσαρτημένων SF τους στο επίπεδο ελέγχου SFC. Η διεπαφή C3 είναι μεταξύ των SFs που είναι συμβατές με την ενθυλάκωση NSH και του επιπέδου ελέγχου SFC. Χρησιμοποιείται για τη συλλογή ορισμένων στατιστικών επεξεργασίας πακέτων (π.χ., ενημέρωση φόρτωσης SFs) από τα SFs. Για SFS που δεν είναι συμβατές με την ενθυλάκωση NSH, παρέχεται ένας διακομιστής μεσολάβησης SFC(SFC-proxy) για τη συλλογή στατιστικών στοιχείων (π.χ., λανθάνοντα χρόνο επεξεργασίας SF και φόρτο εργασίας) και τη μετάδοση αυτών των πληροφοριών μέσω της διασύνδεσης C4 στο επίπεδο ελέγχου SFC. Το επίπεδο ελέγχου SFC χρησιμοποιεί αυτά τα στατιστικά στοιχεία (που λαμβάνονται μέσω διεπαφών C2, C3 και C4) για τη δυναμική ρύθμιση των SFP.

Τα κύρια στοιχεία του επιπέδου δεδομένων SFC, όπως φαίνεται στο σχήμα 3, είναι ο SFC Classifier, ο SFF, η SF και το SFC proxy. Ο SFC Classifier διαφοροποιεί την εισερχόμενη κίνηση σε ροές, με βάση την εφαρμογή-στόχο και άλλες προκαθορισμένες απαιτήσεις. Ο SFC Classifier επεξεργάζεται κάθε ροή, προσθέτοντας μια κεφαλίδα SFC που περιέχει ένα αναγνωριστικό διαδρομής λειτουργίας υπηρεσίας(SFP) σε κάθε κεφαλίδα πακέτων ροής. Το αναγνωριστικό διαδρομής σχετίζεται με ένα SFC και προσδιορίζει το διατεταγμένο σύνολο των αφηρημένων SF που πρέπει να εκτελούνται στη συγκεκριμένη ροή. Το SFP είναι η πραγματική διαδρομή (τα ακριβή SFFs / SFs) που περνούν τα πακέτα.

Μία SF εκτελεί ένα συγκεκριμένο σύνολο ενεργειών στα εισερχόμενα πακέτα (π.χ. ανάλυση πακέτων ή λειτουργίες τείχους προστασίας) και μπορεί να επεξεργάζεται πακέτα που ανήκουν σε διάφορα SFP. Μία SF μπορεί να υπάρχει με πολλαπλές, κατανεμημένες παρουσίες στο δίκτυο (π.χ., για λόγους κλιμάκωσης). Ένας SFF είναι υπεύθυνος για την αποστολή της εισερχόμενης κίνησης σε SFs ή άλλους SFFs, σύμφωνα με τα καθορισμένα SFPs. Για το σκοπό αυτό, ο SFF χρησιμοποιεί και εισάγει συγκεκριμένες πληροφορίες SFP σε μια πρόσθετη κεφαλίδα πακέτων (encapsulation πακέτων SFP). Η ομάδα εργασίας IETF SFC δεν τυποποιεί ένα συγκεκριμένο SFF, αλλά την ειδική κεφαλίδα SFC, που ονομάζεται κεφαλίδα υπηρεσίας δικτύου (NSH) (draft-ietf-sfc nsh-02). Ένας διακομιστής μεσολάβησης SFC μπορεί να απαιτείται μεταξύ του SFF και των SFs, καθώς η πλειονότητα των SF δεν αναγνωρίζει τις κεφαλίδες πακέτων SFC (NSH). Ο διακομιστής μεσολάβησης SFC εκτελεί δέσμη πακέτων SFC για τα πακέτα που προωθούνται στα SFS που δεν γνωρίζουν την NSH και ενσωματώνει αυτά τα πακέτα προτού τα στείλει στο SFF (IETF SFC RFC 7665).

Στα πλαίσια του SDN, το Open Networking Foundation(ONF) πρότεινε επίσης ένα άλλο μοντέλο για την αρχιτεκτονική SFC L4-L7, βασισμένο στον ελεγκτή SDN/OpenFlow(ONF TS-027). Το σύστημα SFC ONF βασίζεται στην προδιαγραφή IETF SFC, δεδομένου ότι καθορίζει το SFF από εκτεταμένη έκδοση μεταγωγέα OpenFlow που υποστηρίζει NSH.

Μια λειτουργική αρχιτεκτονική επιπέδου ελέγχου SFC αντιμετωπίζεται από την αρχιτεκτονική ETSI NFV (ETSI GS NFV-MAN 001 V1.1.1) (βλέπε σχήμα 4). Τα κύρια συστατικά στοιχεία της αρχιτεκτονικής NFV του ETSI είναι: ο NFV orchestrator (NFVO), ο διαχειριστής λειτουργίας εικονικού δικτύου (VNFM) και ο εικονικός διαχειριστής υποδομής (VIM) όπως αναλύθηκαν στο πρώτο κεφάλαιο.




**Εικόνα 4.4 Αρχιτεκτονική SFC του ONF[30]**

Η NFVO είναι υπεύθυνη για τη διαχείριση και ενορχήστρωση των υπηρεσιών δικτύου(NS). Κάθε NS προσδιορίζεται από έναν περιγραφέα(descriptor) υπηρεσίας δικτύου (Network Service Descriptor). Ένα NS μπορεί να εκτείνεται σε πολλαπλούς τομείς δικτύου που ανήκουν στον ίδιο ή διαφορετικούς διακομιστές. Κάθε τομέας δικτύου περιέχει έναν διαχειριστή επιπέδου δικτύου που ονομάζεται ελεγκτής δικτύου ο οποίος είναι υπεύθυνος για τη διαχείριση της σύνδεσης δικτύου. Στην περίπτωση ενός NS που εκτείνεται σε πολλαπλούς διοικητικούς(administrative) τομείς, η συνολική διαχείριση από άκρο σε άκρο των NS πραγματοποιείται με τη συνεργασία των συμμετεχόντων NFVO, είτε με ιεραρχική είτε με ομότιμη μέθοδο. Στην περίπτωση της ιεραρχικής ρύθμισης, ένα πρόσθετο NFVO εισάγεται στην αρχιτεκτονική. Κάθε τομέας εικονικής υποδομής διοικείται από το λεγόμενο VIM (π.χ. στην περίπτωση του OpenStack, ο διαχειριστής υποδομής εικονικού δικτύου είναι το Neutron). Επίσης, η NFVO ασχολείται με την εκπόνηση/ανανέωση/τερματισμό των SFC (δηλ. Τη διαχείριση κύκλου ζωής του SFC) και των συστατικών VNF τους (εκδοχή, ενημέρωση, κλιμάκωση, μετανάστευση και τερματισμός) σε συντονισμό με τους VNFMs. Ο VNFM είναι υπεύθυνος για τη διαχείριση του κύκλου ζωής των VNF, όπως είναι η παράσταση VNFs, η ενημέρωση/αναβάθμιση, η κλιμάκωση και ο τερματισμός. Το VIM ασχολείται με τον έλεγχο και τη διαχείριση των υπολογιστικών πόρων, των πόρων μέσων αποθήκευσης και των δικτυακών πόρων της NFV υποδομής (NFVI), όπως η παροχή διασύνδεσης "Network as a Service" στην διεπαφή προς τα ανώτερα στρώματα (NFVO και VNFM), και την κλήση των διασυνδέσεων NFVI southbound (ελεγκτή και VNFs/PNFs) για την κατασκευή της υπηρεσίας εντός του τομέα. Κάθε NS περιέχει τουλάχιστον ένα γράφημα προώθησης VNF (VNFFG)

το οποίο περιγράφει την τοπολογία δικτύου του NS ή ένα τμήμα του NS με αναφορά στις VNFs, PNFs, network forwarding path (NFP) που παρέχει τη σειρά των εμπλεκόμενων VNFs ή PNFs VNFFG και οι εικονικοί σύνδεσμοι που τις συνδέουν. Στην ορολογία SFC, το VNFFG θεωρείται ως η αλυσίδα SFC, τα VNFs ή τα PNF είναι τα SFs, τα NFPs είναι τα SFPs και οι εικονικές συνδέσεις υλοποιούνται από ένα ή διαφορετικά SFFs. Το σχήμα 4 δείχνει ένα παράδειγμα δύο VNFFGs (SFCs) ενσωματωμένων στην ίδια υποδομή εικονικού δικτύου.

**4.5) Ενθυλάκωση NSH στον OVS[27]**

Στην συγκεκριμένη εργασία θα προσεγγίσουμε την υλοποίηση της SFC με την IETF SFC αρχιτεκτονική, όπου τους Classifiers και τους SFFs θα τους υλοποιήσουμε με την βοήθεια μεταγωγέων Openflow τύπου OVS. Επειδή την περίοδο που αναπτύχθηκε η παρούσα εργασία δεν είχε ανακοινωθεί η έκδοση του OVS που υποστηρίζει ενθυλάκωση NSH χρησιμοποιήθηκε ένα Patch του OVS βασισμένο στην βιβλιοθήκη DPDK(Data Plane Development Kit) και υλοποιεί κανονικά την διαδικασία που απαιτείται και είναι συμβατό με τον ελεγκτή SDN που χρησιμοποιήθηκε, τον Opendaylight(ODL) για την αναγκαία παραμετροποίηση και υλοποίηση της SFC διάταξης μας. Επομένως είναι χρήσιμο πριν προχωρήσουμε στην υλοποίηση του πειράματος να εξηγηθεί ο τρόπος που το OVS χρησιμοποιεί το NSH.

Πιο συγκεκριμένα στην εργασία μας χρησιμοποιούμε την τεχνική SFC OpenFlow Renderer (SFC OF Renderer) που υλοποιεί την αλυσίδα εξυπηρέτησης σε μεταγωγείς OpenFlow. Παίρνοντας εντολή από το επίπεδο ελέγχου του ελεγκτή ODL για τη δημιουργία ενός Rendered Service Path (RSP) προς τον αποθηκευτικό χώρο του ελεγκτή, προγραμματίζει τους Service Function Forwarders (SFF) που φιλοξενούνται σε μεταγωγείς OpenFlow για την προώθηση πακέτων μέσω της αλυσίδας υπηρεσιών.

Το SFC OpenFlow Renderer χρησιμοποιεί τους παρακάτω πίνακες για τον αγωγό του Flow[6]:

- Πίνακας Table 0, Classifier(Ταξινομητής)
- Πίνακας Table 1, Transport Ingress(Είσοδος)
- Πίνακας Table 2, Path Mapper (Αντιστοίχιση διαδρομών)
- Πίνακας Table 3, ACL Mapper Path(αντιστοίχιση διαδρομών σε ACL)
- Πίνακας Table 4, Next Hop(Επόμενος βήμα)
- Πίνακας Table 10, Transport Egress (Έξοδος)

Το pipeline των πινάκων OpenFlow προορίζεται να είναι γενικό για να λειτουργεί για όλες τις διαφορετικές ενθυλακώσεις που υποστηρίζει το SFC.

Όσον αφορά την λειτουργία των πινάκων:

**α)Πίνακας Classifier**

Είναι πιθανό το SFF να λειτουργεί και ως Classifier. Αυτός ο πίνακας αντιστοιχεί την κίνηση που έρχεται από τους χρήστες στα RSP και υλοποιεί όλη την διαδικασία ενός Classifier. Αν ο SFF δεν είναι Classifier, τότε αυτός ο πίνακας θα έχει απλά μια ροή Goto Table 1.

**β)Πίνακας Transport Ingress**

Ο πίνακας 1 έχει μια είσοδο ανά αναμενόμενο τύπο μεταφοράς σήραγγας που πρόκειται να ληφθεί σε ένα συγκεκριμένο SFF, όπως καθορίζεται στη διαμόρφωση της SFC.

**γ)Πίνακας Path Mapper**

Ο πίνακας 2 έχει μια καταχώρηση ανά αναμενόμενη πληροφορία μεταφοράς σήραγγας που πρόκειται να ληφθεί σε έναν συγκεκριμένο SFF, όπως καθορίζεται στην διαμόρφωση SFC. Οι πληροφορίες μεταφοράς σήραγγας χρησιμοποιούνται για τον προσδιορισμό του αναγνωριστικού διαδρομής RSP και αποθηκεύονται στα μεταδεδομένα του OpenFlow. Αυτός ο πίνακας δεν χρησιμοποιείται για NSH, δεδομένου ότι το αναγνωριστικό διαδρομής RSP αποθηκεύεται στην κεφαλίδα NSH.

**δ)Πίνακας ACL Mapper Path**

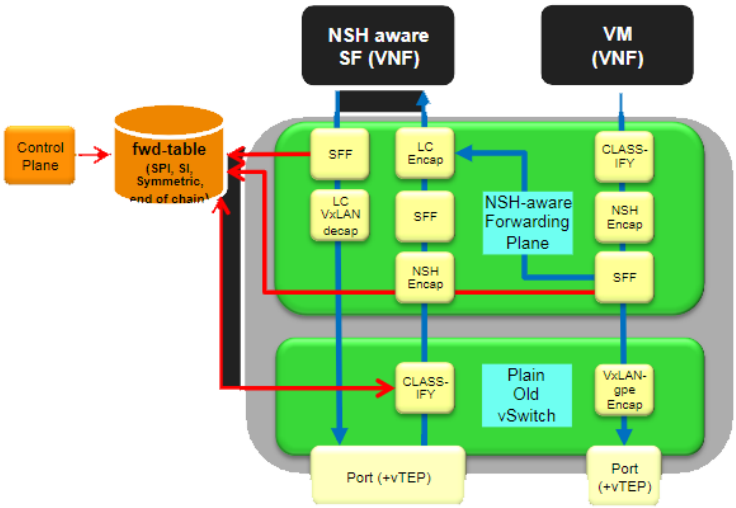
Αυτός ο πίνακας συμπληρώνεται μόνο όταν τα πακέτα PacketIn λαμβάνονται από τον μεταγωγέα για SFs τύπου TcpProxy. Αυτές οι ροές δημιουργούνται με χρονόμετρο αδράνειας 60 δευτερολέπτων και θα διαγραφούν αυτόματα μετά τη λήξη τους.

**ε)Πίνακας Next Hop**

Ο πίνακας Next Hop χρησιμοποιεί το RSP Path ID και τα κατάλληλα πεδία πακέτων για να καθορίσει πού να αποσταλεί το επόμενο πακέτο. Για το NSH, απαιτούνται μόνο τα πεδία NSP (Network Service Path, RSP ID) και NSI (Network Service Index, next hop) από την κεφαλίδα NSH για τον προσδιορισμό του IP προορισμού της σήραγγας VXLAN.

**στ)Πίνακας Transport Egress**

Ο πίνακας "Έλεγχος μεταφορών" προετοιμάζει πληροφορίες σήραγγας εξόδου και αποστέλλει τα πακέτα




**Εικόνα 4.5 OVS και NSH[27]**

# Κεφάλαιο 5ο

## Αρχιτεκτονική συστήματος

Προτού γίνει ανάλυση των δομικών στοιχείων των δύο τοπολογιών δικτύου που θα αναπτυχθούν για την εργασία, παρουσιάζεται η συνολική αρχιτεκτονική των συστημάτων για να υπάρχει μια συνολική εποπτεία. Αναλυτική περιγραφή της υλοποίησης κάθε μιας τεχνολογίας, των ρυθμίσεων στο δίκτυο και μια πιο λεπτομερής προσέγγιση του δικτύου θα παρουσιαστούν στα κεφάλαια 6,7.

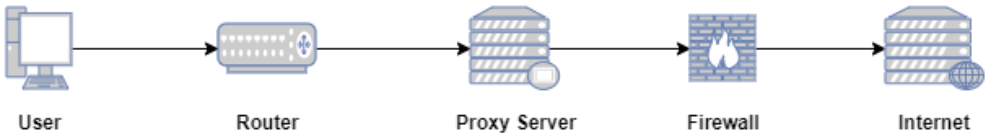
Σκοπός της εργασίας είναι η δημιουργία εφαρμογής Parental Control(Γονικού ελέγχου) χρησιμοποιώντας δύο διαφορετικές τεχνολογίες για την δημιουργία του overlay δικτύου:

- Την υλοποίηση του σε απλή αλυσίδα υπηρεσιών που θα συνδέονται μεταξύ τους με VXLAN overlay και ο χρήστης θα συνδέεται μέσω ενός vCPE στην εικονικοποιημένη υποδομή που θα αναπτύσσονται οι υπηρεσίες.
- Την υλοποίηση με την χρήση της τεχνολογίας SFC, όπου η κίνηση θα ανταλλάσσεται μεταξύ ενός πελάτη και ενός εξυπηρετητή, και ο έλεγχος θα γίνεται από ελεγκτή SDN.

### 5.1) Αρχιτεκτονική συστήματος με χρήση VXLAN overlay

Στην πρώτη υλοποίηση για την δημιουργία μιας αλυσίδας υπηρεσιών που θα παρέχει Parental Control σκοπός είναι η προερχόμενη κίνηση από τον χρήστη προς το διαδίκτυο να περνάει μέσα από δύο λειτουργίες:

- Έναν διακομιστή μεσολάβησης(http proxy server), για να βελτιώσει την ταχύτητα πλοήγησης και να εφαρμόσει φίλτρα επιπέδου 7 με τεχνολογίες URL filtering.
- Ένα τείχος προστασίας(firewall) για να μπλοκάρει την κίνηση σε διευθύνσεις IP που παραπέμπουν σε ακατάλληλες ιστοσελίδες ή έχουν επιλεγεί από τον γονέα



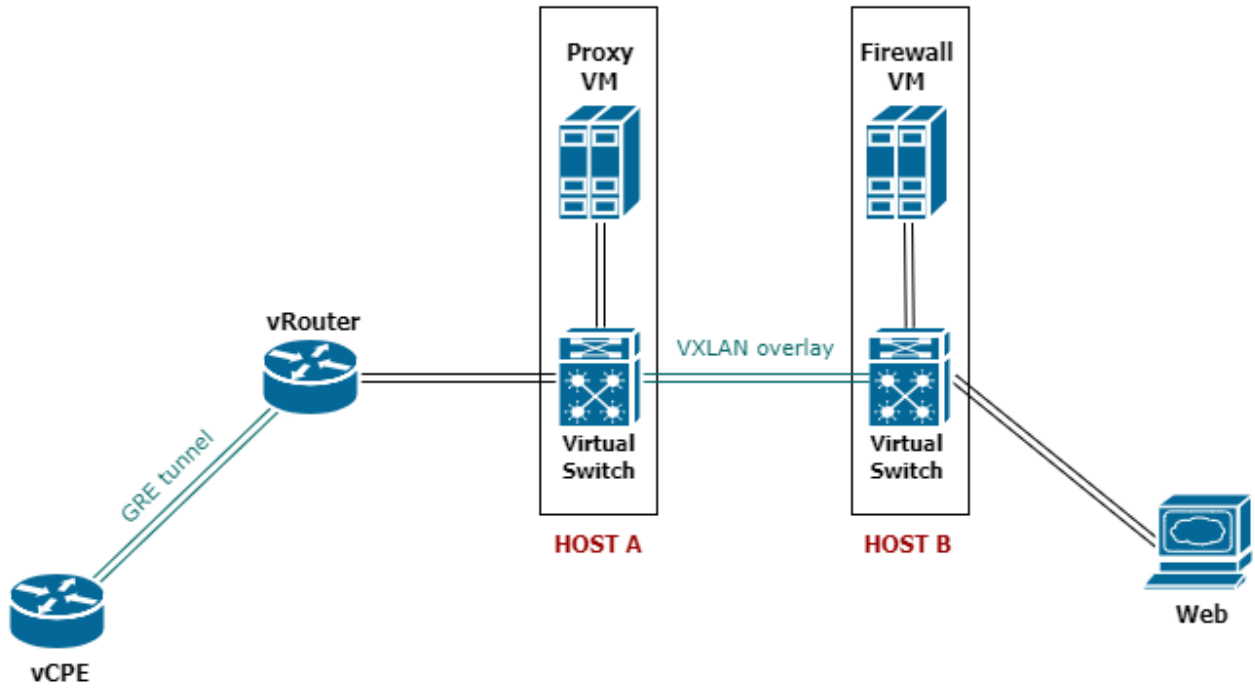

**Εικόνα 5.1 Αλυσίδα υπηρεσιών προς υλοποίηση**

Για να εισέλθει η κίνηση από τον χρήστη που θέλουμε να εφαρμόσουμε γονικό έλεγχο, ως πύλη δρομολόγησης(gateway) προς την εικονικοποιημένη υποδομή των δύο λειτουργιών χρησιμοποιείται ένα Raspberry Pi, το οποίο λειτουργεί ως εικονικό CPE(virtual Customer Premise Equipment – vCPE). Μέσω ενός GRE tunnel η εισερχόμενη κίνηση στο Raspberry Pi προωθείται σε έναν εικονικό δρομολογητή (virtual router) τεχνολογίας VyOS, ο οποίος με την σειρά του προωθεί την κίνηση στην αλυσίδα με τις δυο λειτουργίες. Η χρήση ενός εικονικού δρομολογητή σε συνδυασμό με έναν φθηνό οικιακό δρομολογητή χρησιμοποιώντας κάποια έκδοση tunneling πρωτοκόλλου (όπως το GRE) είναι ευρέως διαδεδομένη για τη παροχή αντίστοιχων υπηρεσιών σε cloud περιβάλλον.

Για την υλοποίηση του http-proxy και του firewall χρειαζόμαστε δύο Hosts: τους HostA και HostB. Σε κάθε Host περιλαμβάνονται:

- Μία εικονική μηχανή(VM) που υλοποιείται η κάθε λειτουργία υπηρεσίας
- Ένας μεταγωγέα OVS που προωθεί την κίνηση από και προς τα δύο VMs, και αποτελεί το VTEP του VXLAN tunnel.

Η επικοινωνία μεταξύ των HostA και HostB θα γίνεται μέσω ενός VXLAN overlay ανάμεσα στους δύο μεταγωγείς. Η πλήρης διάταξη φαίνεται στην Εικόνα 5.2.

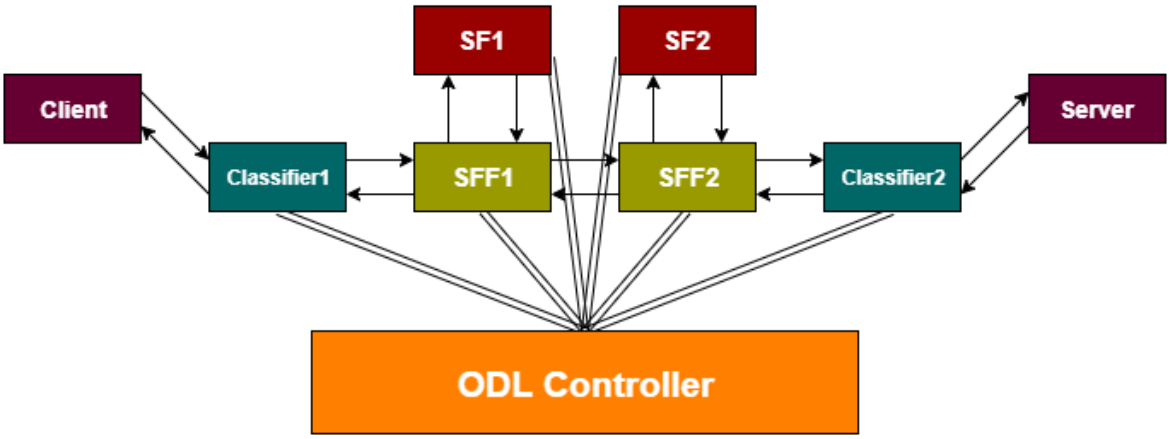



**Εικόνα 5.2 Αλυσίδα υπηρεσιών με χρήση VXLAN**

### 5.2) Αρχιτεκτονική συστήματος με SFC

Στην δεύτερη υλοποίηση μας χρησιμοποιούμε την αρχιτεκτονική SFC που προκρίνεται από την ομάδα εργασίας της IETF και αναλύθηκε στο προηγούμενο κεφάλαιο. Η αλυσίδα που θα αναπτυχθεί αφορά δύο λειτουργίες υπηρεσιών:

- Την λειτουργία DPI(Deep Packet Inspection)
- Την λειτουργία τείχους προστασίας.

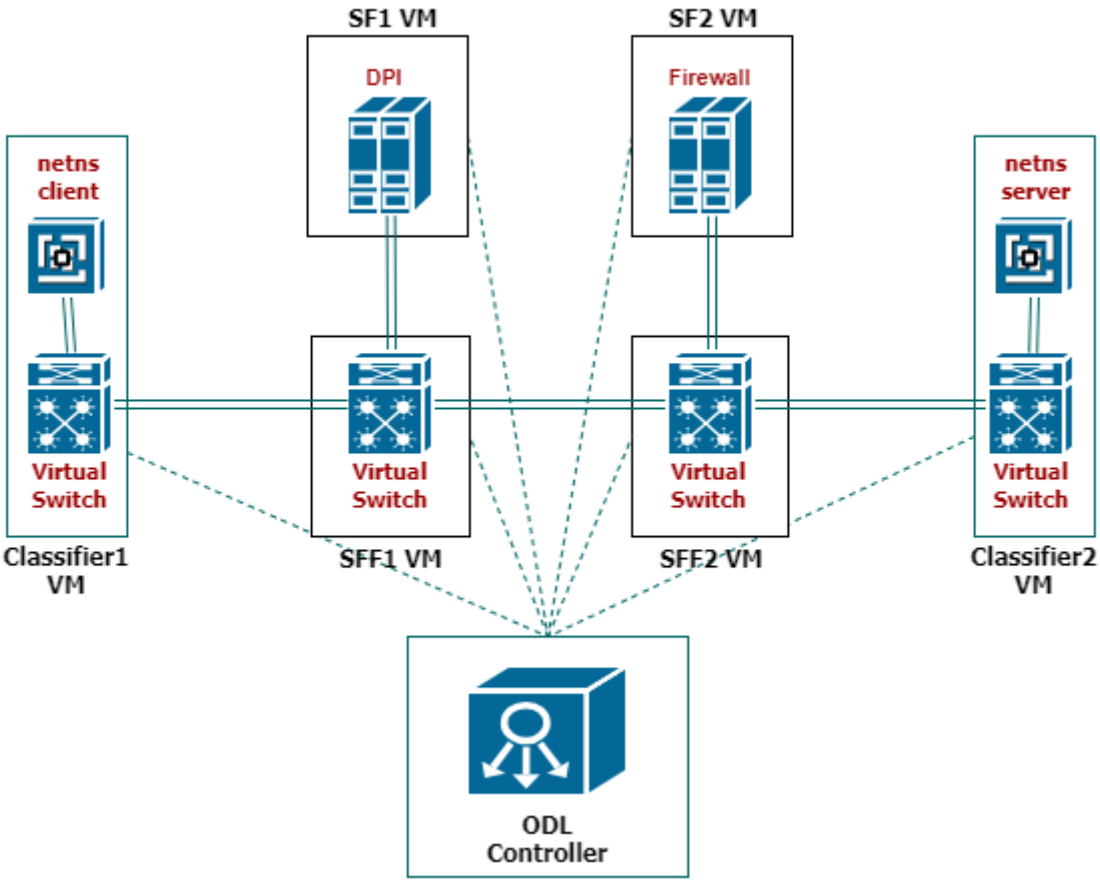



**Εικόνα 5.3 Γενική αρχιτεκτονική υλοποίησης με SFC**

Στην Εικόνα 5.3 φαίνεται μια γενική οπτική της αρχιτεκτονικής. Για να την υλοποιήσουμε χρειαζόμαστε:

- 2 VMs που θα υλοποιούν τις δυο λειτουργίες.
- 2 VMs με έναν μεταγωγέα OVS εσωτερικά στο καθένα όπου θα λειτουργούν ως SFFs.
- 1 VM με τον Classifier της εισερχόμενης κίνησης στην αλυσίδα ο οποίος θα είναι πάνω σε έναν μεταγωγέα OVS. Η εισερχόμενη θα δίνεται από ένα client που αναπτύσσεται με το εργαλείο netns.
- 1 VM με τον Classifier της εξερχόμενης κίνησης από την αλυσίδα ο οποίος θα είναι πάνω σε έναν μεταγωγέα OVS. Η κίνηση προορίζεται προς έναν http server που αναπτύσσεται με το εργαλείο netns.

Για την παραμετροποίηση των SFF και των Classifier προκειμένου να λειτουργήσει η SFC όπως εξηγήθηκε στο κεφάλαιο 4, βασικό ρόλο επιτελεί ο SDN Ελεγκτής(SDN Controller), ο οποίος με την χρήση του πρωτοκόλλου Openflow, ελέγχει την διακινούμενη πληροφορία επικοινωνώντας με τους μεταγωγείς OVS. Ο έλεγχος αυτός γίνεται με την μορφή ροών(flows) που αποστέλλονται στον SDN ελεγκτή μέσω του REST API που αυτός διαθέτει, χρησιμοποιώντας το πρωτόκολλο RESTCONF. Ο ελεγκτής που χρησιμοποιούμε, ο Opendaylight επιτελεί αυτό τον ρόλο και στέλνει τα κατάλληλα flows για να δημιουργηθούν και να λειτουργούν οι SFFs και οι Classifiers, και επικοινωνεί με τους agents που είναι στα VMs που αναπτύσσονται οι δύο δικτυακές λειτουργίες.




**Εικόνα 5.4 Αρχιτεκτονική συστήματος υλοποιημένο με SFC**

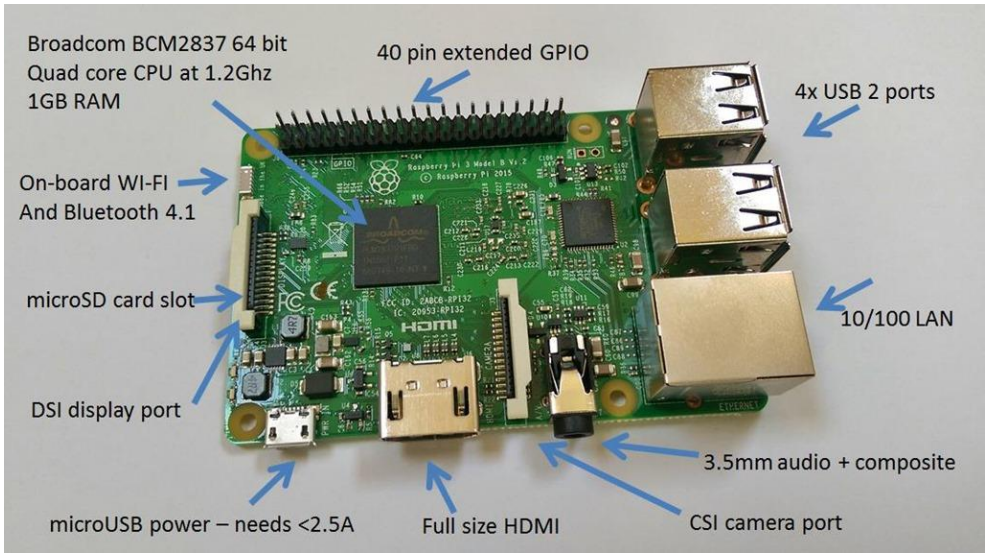
# Κεφάλαιο 6ο

# Τα δομικά στοιχεία του συστήματος

## 6.1) Raspberry Pi(RPi)

Το Raspberry Pi είναι ένας υπολογιστής μεγέθους πιστωτικής κάρτας που σχεδιάστηκε αρχικά για εκπαιδευτικούς λόγους, ως μια συσκευή χαμηλού κόστους που θα χρησιμοποιούνταν για την εκμάθηση της επιστήμης των υπολογιστών (βελτίωση δεξιοτήτων προγραμματισμού και κατανόηση του υλικού). Λόγω του μικρού μεγέθους και της προσιτής τιμής, υιοθετήθηκε γρήγορα από κατασκευαστές ηλεκτρονικών για τα έργα που απαιτούν περισσότερο από έναν βασικό μικροελεγκτή(όπως συσκευές Arduino) και χρησιμοποιήθηκε όμως σε πληθώρα άλλων εφαρμογών, όπως η ρομποτική, οι υπηρεσίες νέφους τα κέντρα δεδομένων κλπ.[31]

Η συσκευή Raspberry Pi μοιάζει με μια μητρική πλακέτα, με τα τσιπάκια και τις θύρες να είναι εκτεθειμένες αλλά έχει όλα τα στοιχεία που χρειάζεστε για να συνδεθεί η είσοδος, η έξοδος και οι συσκευές αποθήκευσης για να ξεκινήσει οποιαδήποτε διαδικασία.




**Εικόνα 6.1 Παράδειγμα στοιχείων Raspberry Pi[32]**

Τα κύρια μέρη ενός Raspberry Pi είναι:

- **ARM CPU/GPU:** Πρόκειται για ένα σύστημα Broadcom BCM σε ένα Chip (SoC) που αποτελείται από μια κεντρική μονάδα επεξεργασίας ARM (CPU) και μια μονάδα επεξεργασίας γραφικών (GPU).
- **GPIO:** εκτεθειμένα σημεία σύνδεσης εισόδου/εξόδου γενικής χρήσης.
- **RCA:** υποδοχή που επιτρέπει τη σύνδεση αναλογικών τηλεοράσεων(ή παρόμοιων συσκευών).
- **Έξοδος ήχου:** πρόκειται για μια τυπική υποδοχή για τη σύνδεση συσκευών εξόδου ήχου.
- **LEDs:** δίοδοι εκπομπής φωτός για όλες τις ενδεικτικές λυχνίες.

- **USB:** κοινή θύρα σύνδεσης για περιφερειακές συσκευές όλων των τύπων.
- **HDMI:** υποδοχή που επιτρέπει τη σύνδεση με οποιαδήποτε συμβατή συσκευή με καλώδιο HDMI.
- **Ισχύς:** υποδοχή τροφοδοσίας Micro USB 5V, στην οποία συνδέεται συμβατό τροφοδοτικό.
- **Κάρτα SD:** υποδοχή κάρτας SD πλήρους μεγέθους. Για την εκκίνηση της συσκευής απαιτείται κάρτα SD με εγκατεστημένο λειτουργικό σύστημα (OS).
- **Ethernet:** επιτρέπει την ενσύρματη πρόσβαση στο δίκτυο.

Στα πλαίσια της εργασίας θα χρησιμοποιήσουμε ένα Rpi που θα λειτουργήσει ως το μέσο σύνδεσης του πελάτη με την εικονικοποιημένη υποδομή για την πρώτη υλοποίηση, μέσω της θύρας Ethernet. Το μοντέλο που χρησιμοποιούμε είναι το μοντέλο Raspberry Pi Model B, το οποίο τρέχει το λειτουργικό σύστημα Raspbian.

Το Raspbian πρόκειται για διανομή Linux βασισμένη στο λειτουργικό σύστημα Debian και αναπτύχθηκε ειδικά για το RPi. Είναι το επίσημο λειτουργικό σύστημα που χρησιμοποιείται στα RPis, και διανέμεται επισήμως και από το ίδρυμα Raspberry Pi Foundation που το σχεδίασε. Το Raspbian είναι εξαιρετικά βελτιστοποιημένο για τους επεξεργαστές ARM χαμηλής απόδοσης της σειράς Raspberry Pi και είναι ο βασικός λόγος της ευρείας χρήσης του.[33]

Το Rpi κατά την υλοποίηση μας θα συνδέεται με έναν εικονικό δρομολογητή μέσω μιας διόδου GRE και για τον σκοπό αυτό θα κατασκευάσουμε μια διεπαφή tunnel έτσι ώστε να συνδέεται ασφαλώς με την εικονικοποιημένη υποδομή.

### 6.2) Εικονικός δρομολογητής VyOS[34]

Βασικό κομμάτι για την δρομολόγηση της κίνησης από έναν πελάτη στην πρώτη υλοποίηση, δηλαδή που θα δρομολογεί την κίνηση που έρχεται από το Raspberry Pi στην εικονικοποιημένη υποδομή, είναι ο εξοπλισμός ενός εικονικού μηχανήματος(VM) με το VyOS. Το VyOS είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα δικτύου το οποίο μπορεί να εγκατασταθεί σε φυσικό υλικό ή σε εικονική μηχανή στον σε οικιακό διακομιστή ή σε πλατφόρμα σύννεφου, λόγω της δυνατότητάς του να τρέχει σε αρχιτεκτονική συστημάτων x86. Βασίζεται στο GNU / Linux και συνδέει πολλαπλές εφαρμογές μέσα από μια ενιαία διεπαφή διαχείρισης.

Σε αντίθεση με άλλες λύσεις όπως το OpenWRT ή το pfSense, το VyOS είναι πιο κοντά στους παραδοσιακούς δρομολογητές υλικού, με έμφαση στην ολοκληρωμένη υποστήριξη για προηγμένες δυνατότητες δρομολόγησης όπως τα πρωτόκολλα δυναμικής δρομολόγησης και η διεπαφή γραμμής εντολών. Βασικές λειτουργίες του VyOS vRouter είναι:

- **Εικονικά δίκτυα(VLANs):** 802.1q και QinQ
- **Στατική και δυναμική δρομολόγηση:** BGP για IPv4 και IPv6, OSPFv2, RIP, RIPng, δρομολόγηση με βάση την πολιτική, ίση δαπάνη πολλαπλών διαδρομών
- **Τείχος προστασίας:** Σύνολα κανόνων τείχους προστασίας για την επισκεψιμότητα IPv4 και IPv6 που μπορούν να αντιστοιχούνται σε διασυνδέσεις, τείχος προστασίας βάσει ζώνης, ομάδες διεύθυνσης/δικτύου/θύρας για τείχη προστασίας IPv4

- **Διεπαφές σήραγγας:** PPPoE, GRE, IPIP, SIT, στατική L2TPv3, VXLAN
- **VPN:** IPsec site-to-site για IPv4 και IPv6, διακομιστή L2TP / IPsec, διακομιστή PPTP, OpenVPN για site-to-site και απομακρυσμένη πρόσβαση.
- **Μετάφραση διεύθυνσης δικτύου(NAT):** Πηγή NAT, λιμάνι προς τα εμπρός, μεταφράσεις ένας προς ένα, ένας προς πολλούς και πολλοί προς πολλούς.
- **DHCP:** διακομιστής και ρελέ DHCP και DHCPv6
- **Επαναφορά:** VRRP, συγχρονισμός πίνακα σύνδεσης
- **Συλλογή μετρικών ροής:** NetFlow και sFlow
- **Διακομιστής μεσολάβησης:** Διακομιστής μεσολάβησης Web και φιλτράρισμα διευθύνσεων URL
- **Διαμόρφωση:** Πολιτικές QoS (πτώση ουράς, δίκαιη ουρά και άλλες), ανακατεύθυνση επισκεψιμότητας.

Στην παρούσα εργασία χρησιμοποιήσαμε το **Quagga,** το λογισμικό που χρησιμοποιεί το VyOS για τη διαχείριση της δρομολόγησης στο δίκτυο, γνωστό σε πλατφόρμες που τρέχουν σε Unix-like λειτουργικά συστήματα. Χρησιμοποιήθηκε για την εγκατάσταση των απαραίτητων διεπαφών δικτύου, την δρομολόγηση της κίνησης και την κατασκευή της διόδου GRE με το Raspberry Pi.

### 6.3) Opendaylight Controller

Σημαντικό κομμάτι της υλοποίησης μας αποτελεί ο έλεγχος της δικτυακής κίνησης εντός της εικονικοποιημένης υποδομής αλλά και ο έλεγχος/παραμετροποίηση/δημιουργία των αλυσίδων λειτουργίας υπηρεσιών(SFC) στην δεύτερη υλοποίηση. Γι' αυτό τον λόγο χρησιμοποιήσαμε το OpenDayLight Project, ένα πρότζεκτ ανοιχτού κώδικα που στην ουσία θα αποτελεί το επίπεδο ελέγχου αλλά μπορεί να αποτελέσει και κομμάτι του επιπέδου διαχείρισης/εφαρμογών μέσω της διεπαφής χρήστη που διαθέτει και κάποιων εφαρμογών που διαθέτει(και δεν χρησιμοποιήσαμε στην παρούσα εργασία).

Το ODL είναι ένα λογισμικό αναπτυγμένο σε Java, το οποίο διαχειρίζεται η κοινοπραξία του Linux Foundation. Η αρχιτεκτονική ODL αναπτύσσεται με βάση την Open Services Gateway Initiative (OSGi), η οποία είναι ένα αρθρωτό(modular) πλαίσιο ανάπτυξης όπου οι χαλαρά συζευγμένες μονάδες συνθέτουν την πλατφόρμα. Οι ενότητες μπορούν να κατασκευαστούν ανεξάρτητα με την δυνατότητα εισαγωγής και εξαγωγής δεδομένων μεταξύ τους. Η αρχιτεκτονική ODL σχηματίζεται σε μια πολυεπίπεδη δομή: το επίπεδο εφαρμογών δικτύου στην κορυφή, το επίπεδο ελεγκτή πλατφόρμας στη μέση και τα στοιχεία δικτύου αντιπροσωπεύουν το κάτω επίπεδο.[35]

Το βασικό επίπεδο του ODL είναι το μεσαίο στρώμα που περιέχει [35]:

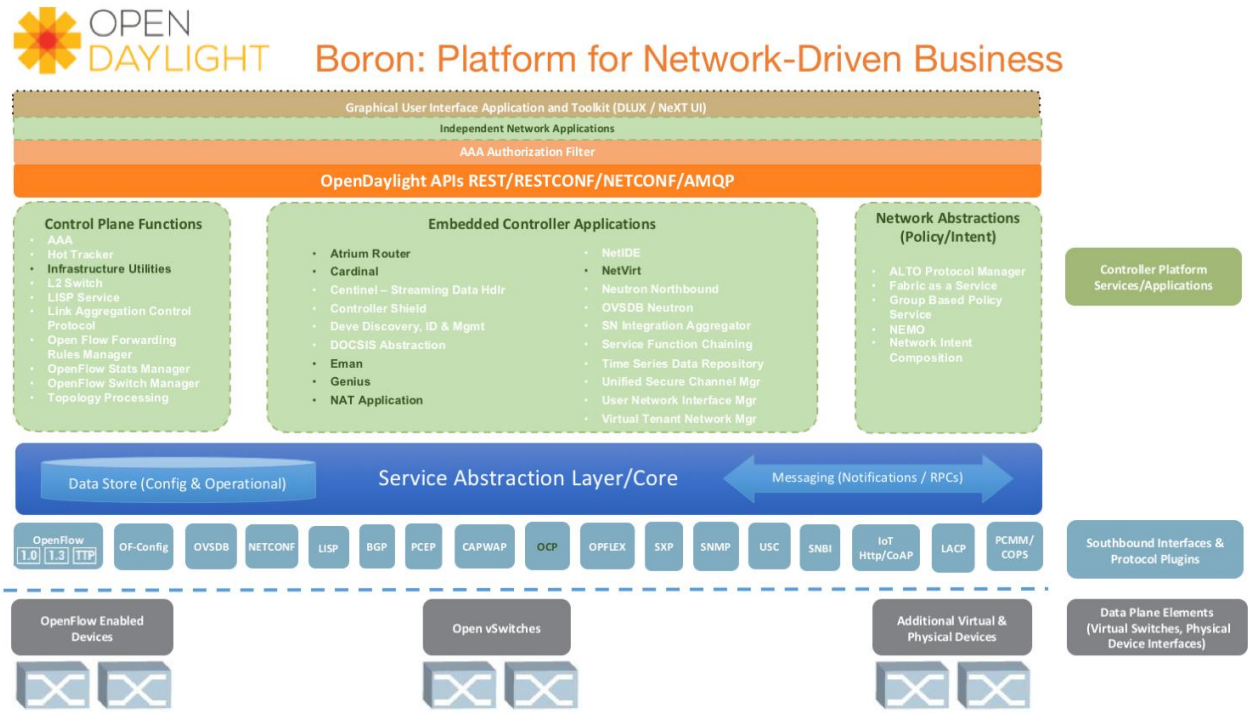
- τις βασικές λειτουργίες του δικτύου όπως η τοπολογία, οι στατιστικές και οι υπηρεσίες προώθησης
- τις λειτουργίες δικτύων πλατφόρμας που περιλαμβάνουν ενότητες για συγκεκριμένες εργασίες δικτύωσης,
- το Service Abstraction Layer(Επίπεδο Αφαίρεσης Υπηρεσιών) που αντιπροσωπεύει ένα αφηρημένο επίπεδο υπηρεσίας μεταξύ του κατώτερου στρώματος και του ανώτερου

στρώματος και επίσης δρομολογεί την υπηρεσία μεταξύ των ενοτήτων του στρώματος αιτήσεων.

Στο 1ο κεφάλαιο αναλύθηκε η λειτουργία του ελεγκτή SDN, οπότε εδώ θα επικεντρωθούμε στην περιγραφή των χαρακτηριστικών της υλοποίησης του OpenDaylight. Ο ελεγκτής ODL αναπτύσσεται σε μεγάλη ποικιλία δικτύων, προσφέροντας ένα μοντελοποιημένο πλαίσιο λειτουργίας και υποστήριξη των περισσότερων SDN πρωτοκόλλων για την επικοινωνία με το επίπεδο δεδομένων(μέσω των Southbound Interfaces) σε σχέση με αντίστοιχες άλλες πλατφόρμες(πχ Floodlight Project, ONOS, NOX, Ryu Controller).

Ο ελεγκτής ODL μέσω των Northbound APIs, δίνει την δυνατότητα σύνδεσης του επιπέδου ελέγχου με διάφορες εφαρμογές. Οι εφαρμογές αυτές χρησιμοποιούν τον ελεγκτή για να πάρουν τις απαραίτητες πληροφορίες για το δίκτυο(τοπολογία, κίνηση κτλ.), και μετά την επεξεργασία αυτών των δεδομένων, να στείλουν προς τις συσκευές που ελέγχονται τους νέους κανόνες για να ρυθμιστεί το δίκτυο.

Για την ανάπτυξη του ODL χρησιμοποιείται αποκλειστικά η γλώσσα Java και ο ελεγκτής διατηρείται εντός της ξεχωριστής εικονικής μηχανής JVM(Java Virtual Machine). Στην υλοποίηση της εργασίας χρησιμοποιήθηκε η έκδοση Boron-SR3, με χρήση JVM 1.8 και εγκατάσταση σε ένα Linux VM, καθώς για τις διεργασίες που χρειαζόμασταν ήταν επαρκής.




**Εικόνα 6.2 Αρχιτεκτονική ODL Boron Controller[36]**

Στις επόμενες ενότητες ακολουθεί περιγραφή των βασικών πυλώνων του ODL ελεγκτή και εξειδίκευση όσον αφορά την τεχνική SFC που χρησιμοποιήθηκε.

### 6.3.1) Πλατφόρμα ελεγκτή

Ο ελεγκτής λειτουργεί ως ενδιάμεσο λογισμικό στην δομή του OpenDaylight. Είναι το πλαίσιο που συνδέει τις εφαρμογές που απαιτούν και αλλάζουν δεδομένα των συσκευών δικτύου και των πρωτοκόλλων που μιλούν στις συσκευές δικτύου για την εξαγωγή δεδομένων και υπηρεσιών. Η Northbound διεπαφή μεταφέρει χαμηλού επιπέδου πληροφορία προς τα ανώτερα στρώματα, διασφαλίζοντας την σωστή λειτουργία και ενδοεπικοινωνία των ξεχωριστών λειτουργιών του ελεγκτή ODL.

Με απλά λόγια, αυτή η στρώση εκθέτει τα ανοιχτά NB APIs στις εφαρμογές δικτύου για τον έλεγχο και τη διαχείριση των φυσικών και εικονικών στοιχείων στο δίκτυο. Επίσης, αποτελείται από τις Βασικές Λειτουργίες Δικτυακής Εξυπηρέτησης (Base Network Serrvice Functions), τις λειτουργίες δικτύου της πλατφόρμας(Platform Network Services Platform) και το στρώμα αφαίρεσης υπηρεσιών (Service Abstraction Layer, SAL).[36]

### 6.3.2)Διεπαφή Southbound

Οι διεπαφές Southbound είναι εκείνα τα πρωτόκολλα και οι επεκτάσεις τους(plug-ins) που διευκολύνουν τον αποτελεσματικό έλεγχο του δικτύου και επιτρέπουν στον ελεγκτή SDN να πραγματοποιεί δυναμικές αλλαγές σύμφωνα με τις απαιτήσεις και τις ανάγκες σε πραγματικό χρόνο και να παίρνει τα δεδομένα που χρειάζεται ώστε να παραμετροποιήσει το δίκτυο. Καθορίζουν τον τρόπο με τον οποίο ο ελεγκτής SDN θα πρέπει να αλληλεπιδρά με το επίπεδο προώθησης για να πραγματοποιήσει προσαρμογές στο δίκτυο, ώστε να μπορεί καλύτερα να προσαρμοστεί στις μεταβαλλόμενες απαιτήσεις. Το ODL υποστηρίζει πληθώρα τέτοιων πρωτοκόλλων επικοινωνίας με το επίπεδο δεδομένων, από τα οποία εμείς θα χρησιμοποιήσουμε το Openflow και το OVSDB, των οποίων η λειτουργία εξηγήθηκε σε προηγούμενα κεφάλαια.[36]

### 6.3.3) Βασικές λειτουργίες δικτυακής Εξυπηρέτησης(Base Network Service Functions)

Ο ελεγκτής ODL παρέχει μια σειρά από βασικές λειτουργίες, η οποίες εγκαθίστανται όπως τα υπόλοιπα features του ελεγκτή όπως θα δείξουμε στο επόμενο κεφάλαιο. Κάποιες από τις λειτουργίες αυτές που θα χρειαστούμε και στην υλοποίηση μας είναι οι εξής:[35]

Ο **Διαχειριστής τοπολογίας(Topology Manager**) αποθηκεύει πληροφορίες σχετικά με τους διαχειριζόμενες συσκευές στο λειτουργικό υπό-δέντρο της τοπολογίας. Σχηματίζει αυτό το δευτερεύον δέντρο ακούγοντας τις ειδοποιήσεις όταν προστίθεται ή αφαιρείται ένας μεταγωγέας. Οι εφαρμογές δικτύου που απαιτούν προβολή δικτύου μπορούν να έχουν πρόσβαση στον Διαχειριστή Τοπολογίας μέσω των APIs της NB .

Ο **Διαχειριστής Στατιστικών(Statistics Manager)** συλλέγει στατιστικές πληροφορίες από τους διαχειριζόμενους μεταγωγείς. Στέλνει αιτήσεις στατιστικών στοιχείων σε όλους τους διακόπτες και διατηρεί τις απαντήσεις στο λειτουργικό υπό-δέντρο στατιστικών στοιχείων. Στατιστικές

πληροφορίες σχετικά με τις θύρες, τους πίνακες και τις ροές μεταγωγής παρέχονται από τον Διαχειριστή Στατιστικών.

Ο **Διαχειριστής Μεταγωγέων(Switch Manager)** αποθηκεύει λεπτομέρειες σχετικά με τους μεταγωγείς και τις θύρες τους για να εντοπίσει τους διακόπτες. Για κάθε εντοπισμένο διακόπτη, αποθηκεύει τις παραμέτρους του στο δέντρο δεδομένων Switch Manager.

Ο **Διαχειριστής κανόνων προωθήσεων(FRM)** ελέγχει για ενημερώσεις ροής, επιλύει τις διενέξεις τους και τις επικυρώνει. Παρέχει βασικούς κανόνες προώθησης όπως οι κανόνες OF, καθώς εγκαθιστά τους κανόνες προώθησης στους διαχειριζόμενους μεταγωγείς μέσω της διεπαφής Southbound.

### 6.3.4) Διεπαφή Northbound

Η διεπαφή Northbound συνδέει το επίπεδο ελέγχου με το επίπεδο εφαρμογών όπως αναλύθηκε και στο πρώτο κεφάλαιο. Οι περισσότεροι ελεγκτές SDN υλοποιούν τα δικά τους Northbound API χωρίς να ακολουθούν οποιοδήποτε τυποποιημένο πρότυπο πληροφοριών. Οι περισσότερες πλατφόρμες ελεγκτών, συμπεριλαμβανομένου και του OpenDaylight, εφαρμόζουν δύο παραδείγματα API για Norhtbound για κάθε υπηρεσία:

• Εσωτερικές διασυνδέσεις (π.χ. διεπαφές Java) που θα χρησιμοποιούνται από συστατικά λογισμικού που αναπτύσσονται ως εσωτερικές μονάδες της ίδιας της πλατφόρμας και ενεργούν ως καταναλωτές της υπηρεσίας.

• API REST, με βάση το πρωτόκολλο HTTP, για να επιτρέπουν σε εξωτερικές εφαρμογές να εκτελούν λειτουργίες CRUD πάνω από τους πόρους που εκτίθενται από τον ελεγκτή.

Ειδικότερα, το OpenDaylight υποστηρίζει ένα εξειδικευμένο είδος REST API, βασισμένο στο πρωτόκολλο RESTCONF και καθορίζεται μέσω των μοντέλων πληροφοριών YANG.

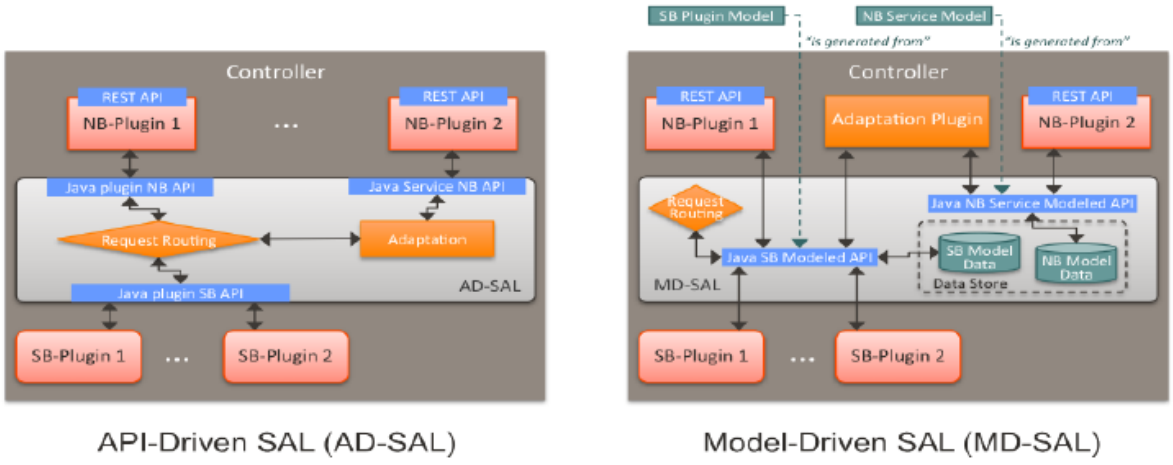
### 6.3.5) Επίπεδο Αφαίρεσης Μοντελοποιημένων Υπηρεσιών[37]

Το SAL(Secure Abstraction Layer), επιτρέπει στην ODL να υποστηρίζει πολλαπλά πρωτόκολλα Southbound(μέσω plug-ins) και να παρέχει ένα ενιαίο σύνολο υπηρεσιών σε άλλες ενότητες και εφαρμογές δικτύου. Για παράδειγμα, η συσκευή Discovery είναι μια υπηρεσία που παρέχεται από το SAL και καταναλώνεται από το Topology Manager για να διαμορφώσει την τοπολογία του δικτύου και να δημιουργήσει δυνατότητες ενός στοιχείου. Οι περισσότερες από τις υπηρεσίες SAL είναι κατασκευασμένες με βάση τις δυνατότητες των Southbound plug-ins, όπου η αιτούμενη υπηρεσία για έναν συγκεκριμένο μεταγωγέα εκπληρώνεται από το SAL, ανεξάρτητα από το Southbound πρωτόκολλο.

Το SAL λειτουργεί ως μητρώο μεγάλων υπηρεσιών όπου οι παραγωγοί “διαφημίζουν” τις υπηρεσίες τους μέσω των API τους. Όταν ένας καταναλωτής ζητά μια διαφημιζόμενη υπηρεσία από ένα γενικό API, το SAL συνδέει και δεσμεύει τόσο τον παραγωγό όσο και τον καταναλωτή. Οι προγραμματιστές του ODL άρχισαν να κωδικοποιούν το αρχικό SAL με μια αρχιτεκτονική SAL με

οδηγό API. Το αριστερό μέρος του παρακάτω σχήματος υπογραμμίζει το API-Driven SAL ή AD-SAL, όπου οι προγραμματιστές έπρεπε να προγραμματίσουν τα SAL API (για να κατευθύνουν τις αιτήσεις παροχής υπηρεσιών μεταξύ των καταναλωτών και των παρόχων) και η λειτουργικότητα προσαρμογής. Παρά το γεγονός ότι η AD-SAL αποκρύπτει την πολυπλοκότητα του στοιχείου, η δυνατότητα κλιμάκωσης(scalability) του ODL μπορεί να περιοριστεί με την κωδικοποίηση των SAL API καθώς και της λειτουργικότητας προσαρμογής κάθε νέου plugin κάθε φορά.

Ως αποτέλεσμα της ανάγκης για μια κλιμακωτή αρχιτεκτονική, εφαρμόζεται μια νέα αρχιτεκτονική βασισμένη στο μοντέλο της SAL (που αναφέρεται ως Model-Driven SAL ή MD-SAL). Όπως απεικονίζεται στα δεξιά της Εικόνας 6.3, η αρχιτεκτονική MD-SAL «κρύβει» την πολυπλοκότητα των SAL API και δρομολογεί τα δεδομένα μεταξύ καταναλωτών και παραγωγών χρησιμοποιώντας APIs της Java που παράγονται από μοντέλα YANG. Όταν συντάσσεται ένα Southbound plugin (το οποίο παρέχει κατά κύριο λόγο υπηρεσίες), ο μεταγλωττιστής YANG δημιουργεί αυτά τα API (όπως RPCs και ειδοποιήσεις) για τους καταναλωτές που είναι μέρος του plugin. Επιπλέον, τα data stores του MD-SAL μπορούν να χρησιμοποιηθούν για την αποθήκευση δεδομένων που παράγονται από μοντέλα. Η αποθήκευση MD-SAL χρησιμοποιείται για την ανταλλαγή δεδομένων μεταξύ παρόχων και καταναλωτών. Η λειτουργικότητα προσαρμογής δεν αποτελεί μέρος του MD-SAL. Παρέχεται από plugins για την αντιστοίχιση μοντέλου-μοντέλου μεταξύ δύο API. Ως εκ τούτου, το MD-SAL δεν έχει API για συγκεκριμένα πρόσθετα, αυτό συνεπάγεται την ικανότητά του να προσαρμόζει τυχόν πρόσθετα Soutbound ή Nourthbound που έχουν φορτωθεί στον ODL.




**Εικόνα 6.3 AD-SAL και MD-SAL[37]**

Στην επόμενη ενότητα θα εξηγηθεί το μοντέλο YANG που είναι απαραίτητο για την όλη λειτουργία που περιγράψαμε.

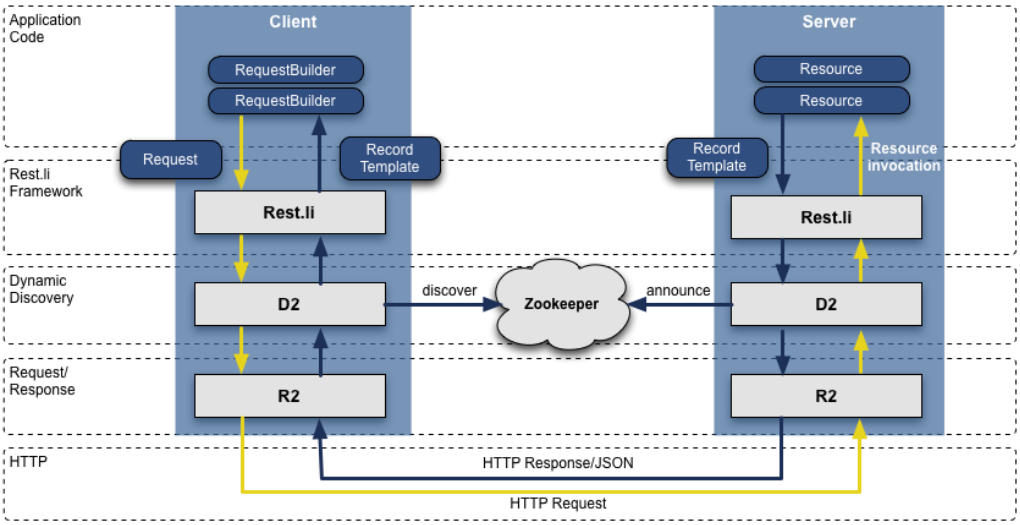
### 6.3.6) Αρχιτεκτονική REST και πρωτόκολλο RESTCONF

#### 6.3.6.1) Αρχιτεκτονική REST

Η αναπαραστατική κατάσταση μεταφοράς (Representational State Transfer – REST) είναι ένα αρχιτεκτονικό στυλ λογισμικού που ορίζει ένα σύνολο περιορισμών που πρέπει να χρησιμοποιηθούν για τη δημιουργία υπηρεσιών ιστού. Οι υπηρεσίες Web που συμμορφώνονται με το αρχιτεκτονικό στυλ REST, ονομάζονται RESTful web services και παρέχουν διαλειτουργικότητα μεταξύ υπολογιστικών συστημάτων στο Διαδίκτυο. Οι πιο απλές υπηρεσίες ιστού επιτρέπουν στα αιτούμενα συστήματα να έχουν πρόσβαση και να χειρίζονται κειμενικές(textual) αναπαραστάσεις των πόρων του διαδικτύου χρησιμοποιώντας ένα ομοιόμορφο και προκαθορισμένο σύνολο stateless λειτουργιών. Άλλα είδη υπηρεσιών ιστού, όπως οι υπηρεσίες ιστού SOAP, εκθέτουν τα δικά τους αυθαίρετα σύνολα λειτουργιών.[38]

Οι "Διαδικτυακοί πόροι" καθορίστηκαν για πρώτη φορά στον Παγκόσμιο Ιστό ως έγγραφα ή αρχεία που προσδιορίζονται από τις διευθύνσεις URL τους. Ωστόσο, σήμερα έχουν έναν πολύ πιο γενικό και αφηρημένο ορισμό που περιλαμβάνει κάθε πράγμα ή οντότητα που μπορεί να προσδιοριστεί, να ονομαστεί, να αντιμετωπιστεί ή να αντιμετωπιστεί με οποιονδήποτε τρόπο στο διαδίκτυο. Σε μια υπηρεσία RESTful web, τα αιτήματα που υποβάλλονται στο URI ενός πόρου θα προκαλέσουν μια απάντηση με ένα φορτίο φορμαρισμένο σε HTML, XML, JSON ή σε κάποια άλλη μορφή. Η απόκριση μπορεί να επιβεβαιώσει ότι έχουν γίνει ορισμένες αλλαγές στον αποθηκευμένο πόρο και η απάντηση μπορεί να παρέχει συνδέσεις υπερκειμένου με άλλους σχετικούς πόρους ή συλλογές πόρων. Όταν χρησιμοποιείται το HTTP, όπως είναι πιο συνηθισμένο, οι διαθέσιμες λειτουργίες είναι GET, POST, PUT, DELETE και άλλες προκαθορισμένες μεθόδους CRUD HTTP.[38]

Χρησιμοποιώντας ένα stateless πρωτόκολλο και τυποποιημένες λειτουργίες, τα συστήματα RESTful αποσκοπούν στη γρήγορη απόδοση, αξιοπιστία και δυνατότητα ανάπτυξης, επαναχρησιμοποιώντας συστατικά που μπορούν να διαχειρίζονται και να ενημερώνονται χωρίς να επηρεάζουν το σύστημα στο σύνολό του, ακόμα και όταν εκτελείται.




**Εικόνα 6.4 Αρχιτεκτονική REST[39]**

#### 6.3.6.2) Πρωτόκολλο RESTCONF

Όπως αναφέρθηκε στην προηγούμενη ενότητα, το OpenDaylight υποστηρίζει ένα εξειδικευμένο είδος REST API, βασισμένο στο πρωτόκολλο RESTCONF και καθορίζεται μέσω των μοντέλων πληροφοριών YANG. Παρέχεται δηλαδή μια προγραμματιστική διεπαφή για πρόσβαση στα δεδομένα χρησιμοποιώντας τα data stores του ελεγκτή. Πιο αναλυτικά:[37]

**α)**Το **YANG** είναι ένα πρότυπο IETF, το οποίο καθορίζεται από την ομάδα εργασίας (Netmod WG), για τη μοντελοποίηση της διαμόρφωσης στοιχείων δικτύου στο πρωτόκολλο Netconf. Στο OpenDaylight, το YANG είναι η γλώσσα μοντελοποίησης που χρησιμοποιείται στο MD-SAL για τον προσδιορισμό των διεπαφών των υπηρεσιών και των προσθηκών. Συγκεκριμένα, χρησιμοποιείται για να μοντελοποιήσει:

• Configuration και operational δέντρα δεδομένων(data stores). Οι δομές δεδομένων που αντιπροσωπεύουν τις παραμετροποιήσιμες παραμέτρους και την κατάσταση των στοιχείων και συστημάτων. Κάθε τμήμα ενός δευτερεύοντος δέντρου αναγνωρίζεται με μοναδικό τρόπο στη διαμόρφωση ή στον λειτουργικό χώρο μέσω ενός αναγνωριστικού στιγμιότυπου.

• Απομακρυσμένες κλήσεις διαδικασίας(RPC), που χρησιμοποιούνται για κλήσεις και invocations που διασχίζουν τα όρια μεταξύ διαφορετικών ενοτήτων. Εφαρμόζονται από μονάδες που λειτουργούν ως Παροχείς Υπηρεσιών και επικαλούνται οι Καταναλωτές Υπηρεσιών.

• Ειδοποιήσεις, για ασύγχρονα συμβάντα που δημοσιεύονται από τους παρόχους υπηρεσιών για τους εγγεγραμμένους ακροατές.

Το YANG αποτελεί μια πλήρη επίσημη γλώσσα συμβόλων με πλούσια σύνταξη και σημασιολογία για την ανάπτυξη εφαρμογών και χαρακτηρίζεται από ιεραρχικά και εξαιρετικά αρθρωτά μοντέλα τα οποία μπορούν εύκολα να ξαναχρησιμοποιηθούν και να επεκταθούν μέσω της "αύξησης" για τον ορισμό νέων υπηρεσιών.

**β)**Το **RESTCONF** είναι ένα πρωτόκολλο τύπου REST που εκτελείται μέσω HTTP για την πρόσβαση σε δεδομένα που σχεδιάστηκαν στο YANG. Απορρέει από το NETCONF, χωρίς να το αντικαθιστά, αλλά απλώς παρέχει πρόσθετα και απλουστευμένα REST APIs, ιδιαίτερα κατάλληλα για την άντληση πόρων με προσανατολισμό πόρων. Το RESTCONF είναι το πρωτόκολλο που χρησιμοποιείται στα Northbound API στο OpenDaylight και επιτρέπει στις εξωτερικές εφαρμογές να έχουν πρόσβαση στα configuration και operational data stores του ελεγκτή ή να επικαλούνται τις RPC για συγκεκριμένες λειτουργίες που ορίζονται στα μοντέλα δεδομένων YANG των μονάδων του OpenDaylight. Το ίδιο το πρωτόκολλο υποστηρίζει επίσης ασύγχρονες ειδοποιήσεις μέσω υποδοχών ιστού.

Ακολουθώντας τις παραδοσιακές αρχές REST, το RESTCONF παρέχει λειτουργίες CRUD σε ένα χώρο αποθήκευσης δεδομένων, με δεδομένα που διαμορφώνονται ως πόροι, τα οποία αναγνωρίζονται από ένα URI, τα οποία μπορούν να υποστούν επεξεργασία ανάλογα με τα δικαιώματα πρόσβασης τους. Οι λειτουργίες RESTCONF μεταφράζονται μέσω μηνυμάτων HTTP, με λειτουργίες επεξεργασίας που έχουν αντιστοιχιστεί μέσω μεθόδων POST, PUT, PATCH ή DELETE HTTP και αιτήσεις ανάκτησης βάσει μεθόδων GET ή HEAD. Κάθε λειτουργία αντιπροσωπεύεται

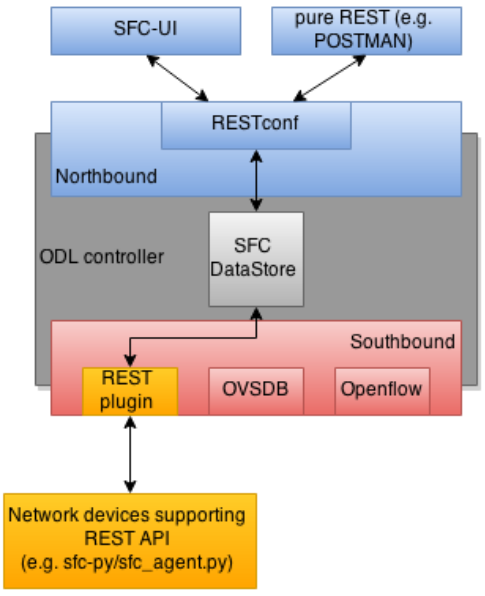
από ένα ζευγάρι <method, URI> όπου τα URI έχουν μια τυπική μορφή: "/ restconf / <path>? <Query>". Το περιεχόμενο των μηνυμάτων μπορεί να είναι σε μορφή XML ή JSON. Παραδείγματα τέτοιων αρχείων αλλά και RPC θα παρουσιαστούν στο επόμενο κεφάλαιο.[37]

### 6.3.7) SFC στον Opendaylight Controller[40]

Η αλυσιδωτή λειτουργία υπηρεσίας (SFC) στον OpenDaylight παρέχει τη δυνατότητα καθορισμού μιας διαταγμένης λίστας υπηρεσιών δικτύου (π.χ. τείχη προστασίας, load balancers). Αυτές οι υπηρεσίες στη συνέχεια συνδέονται στο δίκτυο για να δημιουργήσουν μια αλυσίδα υπηρεσιών. Το project αυτό παρέχει την υποδομή (αλυσίδα λογικής, API) που απαιτούνται για τον ODL για να παρέχει μια αλυσίδα υπηρεσιών στο δίκτυο και μια εφαρμογή τελικού χρήστη για τον ορισμό τέτοιων αλυσίδων, που παραμετροποιεί μέσω του ODL τα παρακάτω στοιχεία που εξηγήθηκαν σε προηγούμενο κεφάλαιο:

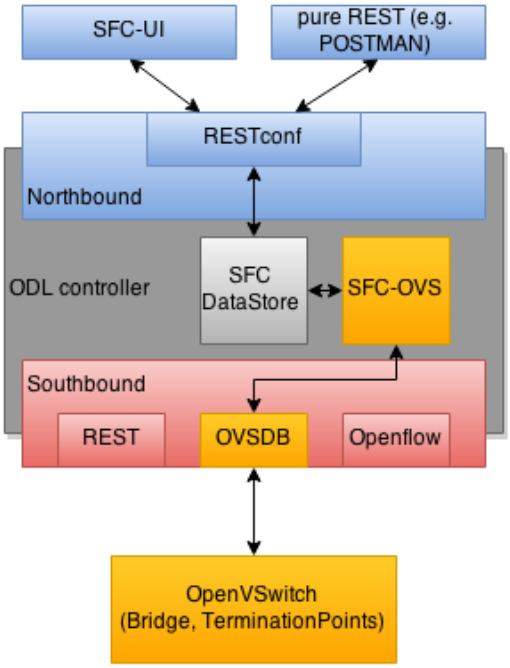
- ACE - Access Control Entry
- ACL - Access Control List
- SCF - Service Classifier Function
- SF - Service Function
- SFC - Service Function Chain
- SFF - Service Function Forwarder
- SFP - Service Function Path
- RSP - Rendered Service Path
- NSH - Network Service Header

Το ODL παρέχει ένα REST Southbound plugin για την λειτουργία SFC, που χρησιμοποιείται για την αποστολή παραμέτρων από το Data Store σε συσκευές δικτύου που υποστηρίζουν ένα REST API. Υποστηρίζει τις λειτουργίες POST / PUT / DELETE, οι οποίες ενεργοποιούνται ανάλογα με τις αλλαγές στα Data Stores του SFC.




**Εικόνα 6.5 Υλοποίηση SFC Southbound Plugin[40]**

Το στοιχείο SFC-OVS παρέχει σύνδεση του SFC με συσκευές Open vSwitch (OVS). Η ενσωμάτωση γίνεται με αντιστοίχιση αντικειμένων SFC (όπως SF, SFF, Classifier κλπ.), σε αντικείμενα του OVS όπως Bridge, TerminationPoint = Port/Interface). Η αντιστοίχιση φροντίζει για την αυτόματη ρύθμισης του αντίστοιχου αντικειμένου. Για παράδειγμα, όταν δημιουργείται ένα νέο SFF, το πρόσθετο SFC-OVS θα δημιουργήσει μια νέα γέφυρα OVS και όταν δημιουργηθεί μια νέα γέφυρα OVS, το πρόσθετο SFC-OVS θα δημιουργήσει ένα νέο SFF. Το χαρακτηριστικό αυτό προορίζεται για χρήστες SFC που επιθυμούν να χρησιμοποιήσουν το Open vSwitch ως υποκείμενη υποδομή δικτύου για την ανάπτυξη RSPs (Rendered Service Paths) όπως και στην 2η υλοποίηση μας.




**Εικόνα 6.6 Σύνδεση SFC με OVS[40]**

Στην διάταξή μας θα χρησιμοποιήσουμε δύο ταξινομητές τύπου Classifier OpenFlow, ο οποίος εφαρμόζει τα κριτήρια ταξινόμησης με βάση τους κανόνες OpenFlow που αναπτύσσονται σε έναν μεταγωγέα OpenFlow. Ένα OVS θα αναλάβει το ρόλο ενός ταξινομητή και θα εκτελεί διάφορες ενθυλακώσεις όπως NSH, VLAN, MPLS, κλπ. Στην υπάρχουσα υλοποίηση, ο ταξινομητής μπορεί να υποστηρίξει την ενθυλάκωση NSH. Οι πληροφορίες αντιστοίχισης βασίζονται σε ACL για διευθύνσεις MAC, θύρες, πρωτόκολλο, IPv4 και IPv6. Τα υποστηριζόμενα πρωτόκολλα είναι TCP, UDP και SCTP. Οι πληροφορίες των ενεργειών στους κανόνες OF, είναι η διαβίβαση των εγκλωβισμένων πακέτων με συγκεκριμένες πληροφορίες σχετικά με το RSP. Οι πίνακες ροών που θα χρησιμοποιηθούν στους Classifier αναδείχθηκαν στον 4ο κεφάλαιο.

Τέλος, με την λειτουργία SFC OpenFlow Renderer (SFC OF Renderer), ο ODL υλοποιεί την αλυσίδα εξυπηρέτησης σε μεταγωγείς OpenFlow. «Ακούει» τη δημιουργία ενός Rendered Path of Service (RSP), και αφού «λάβει» τους Service Forwarders Service (SFF) που φιλοξενούνται σε OVS για να κατευθύνουν τα πακέτα μέσω της αλυσίδας υπηρεσιών. Περισσότερα για την υλοποίηση αυτής της λειτουργίας θα αναδειχθούν στο επόμενο κεφάλαιο.

### 6.4) OVS[41]

Το OVS όπως αναφερθήκαμε και σε προηγούμενο κεφάλαιο είναι ένας μεταγωγέας λογισμικού πολλαπλών στρώσεων με άδεια χρήσης υπό την άδεια ανοικτού κώδικα Apache 2. Είναι μια πλατφόρμα μεταγωγής που υποστηρίζει τις τυποποιημένες διεπαφές διαχείρισης και ανοίγει τις λειτουργίες προώθησης σε προγραμματική επέκταση και έλεγχο.

Ο OVS είναι κατάλληλος για λειτουργία ως εικονικός μεταγωγέας σε περιβάλλοντα VM και για αυτό τον λόγο επιλέχθηκε και στα δύο συστήματα. Εκτός από την παρουσίαση τυποποιημένων διεπαφών ελέγχου και ορατότητας στο εικονικό στρώμα δικτύωσης, σχεδιάστηκε για να υποστηρίζει τη διανομή σε πολλαπλούς φυσικούς διακομιστές. Το OVS υποστηρίζει πολλαπλές τεχνολογίες virtualization που βασίζονται στο Linux, συμπεριλαμβανομένης και της KVM που χρησιμοποιήθηκε στην δημιουργία των VMs που χρειάστηκαν στην πρώτη διάταξη.

Το μεγαλύτερο μέρος του κώδικα γράφεται σε ανεξάρτητη από πλατφόρμα C και μεταφέρεται εύκολα σε άλλα περιβάλλοντα. Η τρέχουσα έκδοση του OpenvSwitch( 2.9.2) υποστηρίζει τις ακόλουθες δυνατότητες:

- Πρότυπο 802.1Q μοντέλο VLAN με θύρες πρόσβασης
- Σύνδεση NIC με ή χωρίς LACP στον μεταγωγέα
- NetFlow, sFlow (R) και καθρέφτες για αυξημένη ορατότητα
- Διαμόρφωση ποιότητας υπηρεσίας (QoS), καθώς και αστυνόμευση
- Geneve, GRE, VXLAN, STT και σήραγγες LISP
- Διαχείριση σφαλμάτων συνδεσιμότητας 802.1ag
- OpenFlow 1.0 συν πολλές επεκτάσεις
- Διαδραστική βάση δεδομένων διαμόρφωσης με δεσμεύσεις C και Python
- Μεταφορά υψηλής απόδοσης χρησιμοποιώντας μια ενότητα πυρήνα του Linux
- Η ενσωματωμένη μονάδα πυρήνα του Linux υποστηρίζει Linux 3.10 και άνω.

Τα κύρια συστατικά της διανομής που χρησιμοποιήσαμε είναι:

- ovs-vswitchd, ένας δαίμονας που υλοποιεί τον μεταγωγέα, μαζί με μια συνοδευτική μονάδα πυρήνα Linux για μεταγωγή βάσει ροής.
- ovsdb-server, ένας ελαφρύς διακομιστής βάσης δεδομένων στον οποίο η ovs-vswitchd στέλνει queries για να λάβει τη διαμόρφωσή της.
- ovs-dpctl, ένα εργαλείο για τη διαμόρφωση της μονάδας πυρήνα μεταγωγέα.
- ovs-vsctl, ένα βοηθητικό πρόγραμμα για την ερώτηση(queries) και την ενημέρωση της διαμόρφωσης του ovs-vswitchd.
- ovs-appctl, ένα βοηθητικό πρόγραμμα που στέλνει εντολές για την εκτέλεση OVS δαιμόνων.

Το Open vSwitch παρέχει επίσης ορισμένα εργαλεία, που κάποια χρησιμοποιήθηκαν:

- ovs-ofctl, ένα βοηθητικό πρόγραμμα για την αναζήτηση και τον έλεγχο των μεταγωγέων και των ελεγκτών OpenFlow.

- ovs-pki, ένα βοηθητικό πρόγραμμα για τη δημιουργία και τη διαχείριση της υποδομής δημόσιου κλειδιού

### 6.5) Πρόσθετα εργαλεία

#### 6.5.1) QEMU/KVM/Libvirt

Για την δημιουργία και εγκατάσταση των εικονικών μηχανών μέσα στους HOST A και B, χρησιμοποιήθηκαν οι κάτωθι τεχνολογίες:

Το **QEMU** είναι ένας γενικός και ανοικτού κώδικα μηχανισμός εξομοίωσης και εικονικοποιητής(virtualizer). Όταν χρησιμοποιείται ως εξομοιωτής μηχανής, το QEMU μπορεί να εκτελεί λειτουργίες λειτουργικού συστήματος και προγράμματα που γίνονται για ένα μηχάνημα (π.χ. μια πλακέτα ARM) σε διαφορετικό μηχάνημα (π.χ. ένα PC). Χρησιμοποιώντας δυναμική μετάφραση, επιτυγχάνει πολύ καλή απόδοση. Όταν χρησιμοποιείται ως virtualizer, το QEMU επιτυγχάνει σχεδόν εγγενή απόδοση εκτελώντας τον κωδικό επισκέπτη απευθείας στο κεντρικό CPU. Το QEMU υποστηρίζει την εικονικοποίηση όταν εκτελείται κάτω από τον hypervisor Xen ή χρησιμοποιώντας τη μονάδα πυρήνα KVM στο Linux όπως γίνεται στην περίπτωση μας.

Το **KVM** (για εικονική μηχανή βασισμένο στο Kernel) είναι μια λύση πλήρους εικονικοποίησης για το Linux σε x86 υλικό που περιέχει επεκτάσεις εικονικοποίησης (Intel VT ή AMD-V). Αποτελείται από μια λειτουργική μονάδα kernel με δυνατότητα φόρτωσης, την kvm.ko, η οποία παρέχει την βασική υποδομή εικονικοποίησης και μια συγκεκριμένη μονάδα επεξεργαστή, kvm-intel.ko ή kvm-amd.ko. Χρησιμοποιώντας KVM, μπορεί κανείς να τρέξει πολλαπλές εικονικές μηχανές που εκτελούν μη τροποποιημένες εικόνες Linux ή Windows. Κάθε εικονική μηχανή διαθέτει ιδιωτικό εικονικό υλικό: κάρτα δικτύου, δίσκο, προσαρμογέα γραφικών, κ.λπ. Το KVM είναι λογισμικό ανοιχτού κώδικα. Η συνιστώσα του πυρήνα της KVM περιλαμβάνεται στο βασικό Linux, από την έκδοση του Linux Kernel 2.6.20 κι έπειτα. Η συνιστώσα χρήστη του KVM περιλαμβάνεται στην κύρια γραμμή QEMU, από την 1.3.

Το **Libvirt** είναι ένα API ανοιχτού κώδικα, που λειτουργεί ως δαίμονας και εργαλείο διαχείρισης για τη διαχείριση της εικονικής πλατφόρμας. Μπορεί να χρησιμοποιηθεί για τη διαχείριση των KVM, Xen, VMware ESX, QEMU και άλλων τεχνολογιών εικονικοποίησης. Αυτά τα API χρησιμοποιούνται ευρέως στη στρώση ορχηστρών των hypervisors για την ανάπτυξη μιας λύσης που βασίζεται σε σύννεφο.

#### 6.5.2) Vagrant-Virtualbox[42]

Το Vagrant είναι ένα εργαλείο για τη δημιουργία και τη διαχείριση περιβαλλόντων εικονικών μηχανών σε μια ενιαία ροή εργασιών. Με μια εύκολη στη χρήση ροή εργασίας και με έμφαση στην αυτοματοποίηση, το Vagrant μειώνει το χρόνο εγκατάστασης του περιβάλλοντος ανάπτυξης.

Το Vagrant παρέχει εύκολη ρύθμιση παραμέτρων, αναπαραγώγιμα και φορητά περιβάλλοντα εργασίας που είναι χτισμένα πάνω στην τεχνολογία που βασίζεται στη βιομηχανία και ελέγχονται από μια ενιαία συνεπή ροή εργασιών που βοηθά στη μεγιστοποίηση της παραγωγικότητας και της ευελιξίας σας και της ομάδας σας.

Τα μηχανήματα παρέχονται πάνω από το VirtualBox, το VMware, το AWS ή οποιοδήποτε άλλο πάροχο. Στην περίπτωση της υλοποίησης του SFC θα το χρησιμοποιήσουμε με την βοήθεια του VirtualBox.

### 6.5.3) ip-netns[43]

Σε γενικές γραμμές, μια εγκατάσταση του Linux μοιράζεται ένα ενιαίο σύνολο διεπαφών δικτύου και καταχωρήσεις πίνακα δρομολόγησης. Οι καταχωρήσεις του πίνακα δρομολόγησης μπορούν να τροποποιηθούν χρησιμοποιώντας τη δρομολόγηση πολιτικής, αλλά αυτό δεν αλλάζει θεμελιωδώς το γεγονός ότι το σύνολο των διεπαφών δικτύου και των πινάκων δρομολόγησης μοιράζονται σε ολόκληρο το λειτουργικό σύστημα. Τα network namespaces αλλάζουν αυτή τη θεμελιώδη παραδοχή, γιατί μπορούμε να έχουμε διαφορετικές και ξεχωριστές παρουσίες διεπαφών δικτύου και πινάκων δρομολόγησης που λειτουργούν ανεξάρτητα το ένα από το άλλο

Πάνω σ’ αυτή την λογική, το εργαλείο ip-netns μας επιτρέπει να δημιουργούμε απομονωμένα, ιδιωτικά εικονικά υποδίκτυα σε έναν κεντρικό υπολογιστή και να τα συνδέουμε μεταξύ τους με εικονικές συσκευές Ethernet, για να προσομοιώνουμε ένα μεγαλύτερο LAN σε έναν κεντρικό υπολογιστή.

Αυτές οι δυνατότητες του συγκεκριμένου εργαλείου βοήθησαν ώστε **να προσομοιώσουμε τον client και τον http-sever στο σύστημα με SFC**.

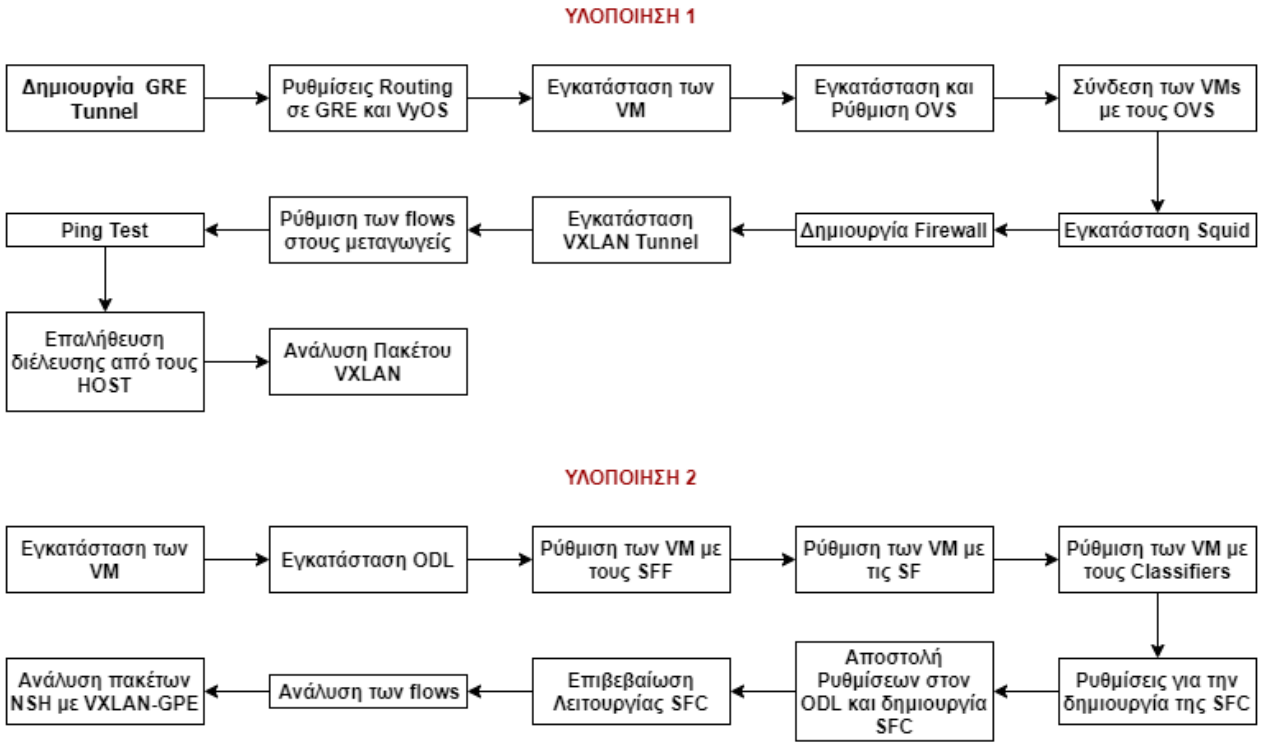
### 6.5.4) SFC Agent[44]

Ο SFC Agent είναι ένα εργαλείο για την παροχή SF που υποστηρίζουν NSH. Τον διαχειρίζεται ο Opendaylight μέσω του REST API και μόλις ρυθμίζεται η αλυσίδα στον ODL, ο agent θα ακούσει (από προεπιλογή στη θύρα 5000) για τη συγκεκριμένη διαμόρφωση και παροχή SF χρησιμοποιώντας τις πληροφορίες που παρέχονται μέσω ODL (τύπος, θύρα UDP, κτλ. ). Στην ουσία υποκαθιστά τον VNF Manager που συναντάμε σε διάφορες προσεγγίσεις της αρχιτεκτονικής SFC και δημιουργεί dummy λειτουργίες υπηρεσίας.

# Κεφάλαιο 7ο

## Ενσωμάτωση Τεχνολογιών/Ρυθμίσεων και πειραματική επιβεβαίωση

Στο κεφάλαιο αυτό προχωράμε στην υλοποίηση των δύο διατάξεων που προκρίναμε στο κεφάλαιο 5. Στο παρακάτω διάγραμμα απεικονίζονται τα βήματα που θα κάνουμε σε κάθε υλοποίηση για την ενσωμάτωση των τεχνολογιών και ρυθμίσεων και για την επιβεβαίωση τους.




**Εικόνα 7.1 Βηματισμός για τις δύο υλοποιήσεις**

### 7.1) Υλοποίηση 1

#### Βήμα 1: Δημιουργία GRE Tunnel για την σύνδεση του Raspberry Pi με το VyOS Router

Για την επικοινωνία του Rpi με τον εικονικό δρομολογητή χρειαζόμαστε μια δίοδο που θα υλοποιηθεί με GRE Tunnel.

Τα δύο στοιχεία έχουν τις εξής διευθύνσεις IP:

- Rpi: 147.102.40.69(διεπαφή eth0)
- VyOS:147.102.7.79(διεπαφή eth0) και 147.102.40.77(διεπαφή eth1)

Πρώτα δουλεύοντας στο configure terminal του VyOS, δημιουργούμε τον απαιτούμενο χώρο στην επικεφαλίδα TCP για να περιλαμβάνει και την επικεφαλίδα του πρωτοκόλλου ενθυλάκωσης. Γι' αυτό τον λόγο δημιουργούμε μια νέα πολιτική δρομολόγησης(routing policy) που την ονομάζουμε change-mss όπου:

- Θέτουμε ως Maximum Segment Size(MSS) που μεταδίδονται σε ένα πακέτο τα 1360 bytes, δίνοντας τα υπόλοιπα 40 bytes για την κεφαλίδα GRE.
- Ορίζουμε ως πρωτόκολλο πολιτικής το TCP
- Επιλέγουμε η ρύθμιση του MSS να γίνει στο αρχικό πακέτο εγκατάστασης της σύνδεσης(SYN).

Μετά την παραμετροποίηση των πακέτων, προχωράμε στην δημιουργία του GRE tunnel στον εικονικό δρομολογητή. Η δίοδος GRE λογίζεται ως ανεξάρτητο δίκτυο με μάσκα υποδικτύου 30 καθώς χρειαζόμαστε 4 διευθύνσεις:

- Διεύθυνση δικτύου 10.0.0.0
- Διεύθυνση αποστολέα και παραλήπτη, της 10.0.0.1 και 10.0.0.2.
- Διεύθυνση broadcast την 10.0.0.3

Για την δημιουργία της διόδου ορίζουμε στον εικονικό δρομολογητή την διεπαφή tun0 με τις παρακάτω ρυθμίσεις:

- Τύπος ενθυλάκωσης GRE
- Διεύθυνση IP της διεπαφής ως 10.0.0.1/30
- MTU ως 1400 bytes
- Επιλογή της πολιτικής δρομολόγησης change-mss που ορίσαμε πριν
- Τοπική διεύθυνση δημόσιου δικτύου την IP του VyOS(147.102.7.79)
- Απομακρυσμένη διεύθυνση δικτύου την διεύθυνση του Rpi(147.102.40.69)

Στο παράρτημα εμφανίζεται ο τρόπος που υλοποιείται το GRE Tunnel.

**Βήμα 2 Ρυθμίσεις προώθησης πακέτων σε Rpi και VyOS**

Για να επαληθεύσουμε την λειτουργία του γονικού ελέγχου θα πρέπει να γίνεται κατάλληλη δρομολόγηση από το vCPE προς το VyOS, και από το VyOS προς τον HOST A για να τεστάρουμε την αλυσίδα.

Με την εμφάνιση των πινάκων δρομολόγησης επιβεβαιώνεται και η προώθηση πακέτων και η δίοδος GRE:

```
bill@pi-pchara:~ $ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         147.102.40.200  0.0.0.0         UG        0 0          0 eth0
8.8.4.4         10.0.0.1        255.255.255.255 UGH       0 0          0 tun0
8.8.8.8         10.0.0.1        255.255.255.255 UGH       0 0          0 tun0
10.0.0.0        0.0.0.0         255.255.255.252 U         0 0          0 tun0
147.102.40.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

**Εικόνα 7.2 Πίνακες δρομολόγησης του Rpi**

Αντίστοιχα στο αποτέλεσμα της εντολής show configuration το παρακάτω απόσπασμα του αποτελέσματος επιβεβαιώνει τις ρυθμίσεις που επιδιώκουμε:

```
protocols {
   static {
      route 0.0.0.0/0 {
         next-hop 147.102.7.200 {
            distance 1
         }
      }
      route 8.8.4.4/32 {
         next-hop 147.102.7.84 {
         }
         next-hop 147.102.7.96 {
            disable
         }
      }
      route 8.8.8.8/32 {
         next-hop 147.102.7.84 {
         }
      }
   }
}
```

Επομένως τα πακέτα που θέλουμε θα εισέρχονται στην αλυσίδα των δύο HOST.

**Βήμα 3 Εγκατάσταση των εικονικών μηχανών VMs στους HOST**

Οι HOST που θα υλοποιήσουμε τις δικτυακές λειτουργίες και θα εγκαταστήσουμε τους εικονικούς μεταγωγείς OVS έχουν τις διεπαφές του παρακάτω πίνακα:

| **HOST** | **Διεπαφή** | **Διεύθυνση IP** |
|---|---|---|
| HOST A | ens160 | 147.102.7.84 |
| | ens192 | 147.102.39.28 |
| HOST B | Ens160 | 147.102.40.76 |
| | ens192 | 147.102.39.29 |

**Πίνακας 1 Διεπαφές των HOST A και B**

Με την χρήση του δαίμονα libvirtd και των QEMU/KVM δημιουργούμε τα εικονικά μηχανήματα VM1 στον HOST A και VM2 στον HOST B με τα εξής χαρακτηριστικά:

- 512 MB μνήμη RAM
- 8 GB σκληρό δίσκο
- Λειτουργικό σύστημα Linux

**Βήμα 4 Εγκατάσταση και ρύθμιση μεταγωγέων OVS**

Για την υλοποίηση της αλυσίδας θα χρειαστούμε από ένα OVS σε κάθε μηχάνημα, τους br1 και br2 αντίστοιχα. Για την δημιουργία των OVS χρησιμοποιούμε το εργαλείο ovs-vsctl. Κάθε μεταγωγέας θα συνδέεται με το ens192 σε κάθε HOST, και θα δίνεται η διεύθυνση IP της κάθε διεπαφής στον αντίστοιχο μεταγωγέα.(βλ. παράρτημα)

**Βήμα 5 Σύνδεση των VM με τα OVS**

Τα VMs είναι συνδεδεμένα στο default δίκτυο του libvirt, το οποίο συνδέεται στην εικονικό δίκτυο 192.168.122.0 μέσω της διεπαφής virbr0. Για την δημιουργία και την παραμετροποίηση δικτύων για τα μηχανήματα μέσω libvirt, το εργαλείο virsh μας δίνει κάποιες δυνατότητες που θα χρησιμοποιήσουμε για να συνδέσουμε το μηχάνημα μέσω των OVS. Κάθε δίκτυο και κάθε VM έχουν ένα αρχείο xml που χρησιμοποιείται για την αναγκαία ρύθμισή τους. Με την κατάλληλη παραμετροποίηση αυτών των αρχείων και την σύνδεση της κάθε εικονικής διεπαφής vnet0 με τον αντίστοιχο OVS σε κάθε HOST (βλ. παράρτημα) επιτυγχάνουμε την σύνδεση τους με τα OVS.

Μετά την σύνδεση με τους μεταγωγείς τα VMs παίρνουν από το δίκτυο 147.102.39.0/24 της διευθύνσεις IP:

- VM1:147.102.39.40
- VM2:147.102.39.41

**Βήμα 6 Εγκατάσταση και έναρξη λειτουργίας proxy στο VM1**

Για την εγκατάσταση της λειτουργίας θα χρησιμοποιήσουμε τον squid server που παρέχεται από το apt στο Linux. Το Squid είναι μια πλήρης εφαρμογή web proxy cache server, η οποία παρέχει υπηρεσίες διακομιστή μεσολάβησης και cache μνήμης HTTP, FTP και άλλα πρωτόκολλα δικτύου. Το Squid μπορεί να υλοποιήσει την προσωρινή αποθήκευση και το proxying των αιτημάτων SSL (Secure Sockets Layer) και την προσωρινή αποθήκευση των αναζητήσεων του DNS (Domain Name Server) και να εκτελέσει διαφανή προσωρινή αποθήκευση. Με την εγκατάσταση από το apt-package γίνεται αυτόματα η εγκατάσταση και η έναρξη του squid proxy με τις default ρυθμίσεις του που στην περίπτωσή μας δεν χρειάζεται να τις αλλάξουμε

**Βήμα 7 Εγκατάσταση και έναρξη λειτουργίας firewall στο VM2**

Το firewall που χρειαζόμαστε στην περίπτωση μας, πρέπει να εκτελεί μια απλή λειτουργία, όπου θα μπλοκάρει την προωθούμενη κίνηση προς την διεύθυνση 8.8.4.4. Για την δημιουργία του θα χρησιμοποιήσουμε τα iptables. Για αυτό τον λόγο δημιουργήσαμε το αρχείο /etc/iptables.rules με τους κανόνες για την εισερχόμενη, εξερχόμενη και προωθούμενη κίνηση στο VM2.

**Αρχείο /etc/iptables.rules:**

```
*filter
:INPUT ACCEPT [0:0] //αποδέχεται όλη την εισερχόμενη κίνηση
```

:FORWARD ACCEPT [0:0] //αποδέχεται όλη την εξερχόμενη κίνηση
:OUTPUT ACCEPT [55:7624] //αποδέχεται όλη την προωθούμενη κίνηση στις θύρες 55 έως 7624
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT //αποδέχεται την είσοδο με SSH στο VM2
-A FORWARD -d 8.8.4.4/32 -j LOG --log-prefix IP_Dropped //αποθηκεύει με την μορφή logs την //προωθούμενη κίνηση προς την 8.8.4.4
-A FORWARD -d 8.8.4.4/32 -j DROP //απορρίπτει την προωθούμενη κίνηση προς την 8.8.4.4
COMMIT

Μετά την δημιουργία του αρχείου με την εντολή **sudo iptables-restore < /etc/iptables.rules** εφαρμόζονται οι νέες ρυθμίσεις στα iptables, όπως φαίνεται στο παρακάτω σχήμα:

```
root@vm2:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
LOG        all  --  anywhere             8.8.4.4              LOG level warning prefix "IP_Dropped"
DROP       all  --  anywhere             8.8.4.4

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

**Εικόνα 7.3 Ρυθμίσεις iptables στο VM2**

Για να δούμε τα πακέτα που απορρίφθηκαν χρησιμοποιούμε την εντολή **cd /var/log && sudo dmesg | grep IP_Dropped** όπως θα δούμε και παρακάτω.

**Βήμα 8 Εγκατάσταση VXLAN Tunnel**

Για την υλοποίηση μας χρειάζεται να δημιουργήσουμε ένα VXLAN tunnel ανάμεσα στα br1 και br2 με τα εξής χαρακτηριστικά:

- VTEP1:147.102.39.28(br1)
- VTEP2:147.102.39.29(br2)
- VN1:42
- Θύρα Openflow σε κάνε μεταγωγέα την θύρα 10.
- Θύρα σε κάθε HOST την θύρα 4789(θύρα VXLAN)
- Όνομα tunnel και τον αντίστοιχων θυρών σε κάθε OVS, vxlan1

Για κάθε OVS χρησιμοποιούμε τις παρακάτω εντολές για την δημιουργία του tunnel:

- sudo ovs-vsctl add-port br1 vxlan1 -- set Interface vxlan1 type=vxlan option:remote_ip=147.102.39.29 option:key=42 option:dst_port=4789 ofport_request=10
- sudo ovs-vsctl add-port br2 vxlan1 -- set Interface vxlan1 type=vxlan option:remote_ip=147.102.39.29 option:key=42 option:dst_port=4789 ofport_request=10

Όπως περιγράφηκε και στο 2ο κεφάλαιο σε κάθε HOST δημιουργείτε μια διεπαφή με το όνομα vxlan_sys_4789:

**root@hostB:~# ifconfig vxlan_sys_4789**

vxlan_sys_4789 Link encap:Ethernet HWaddr 3e:cc:50:06:06:58

inet6 addr: fe80::3ccc:50ff:fe06:658/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:65485 Metric:1

RX packets:3498 errors:0 dropped:0 overruns:0 frame:0

TX packets:3356 errors:0 dropped:8 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:293832 (293.8 KB) TX bytes:147500 (147.5 KB)

**Βήμα 9 Ρύθμιση των ροών στα br1 και br2**

Για την προώθηση των πακέτων που θέλουμε κατά μήκος της αλυσίδας, χρειάζεται να ρυθμίσουμε τους μεταγωγείς OVS κατάλληλα, με την χρήση του πρωτοκόλλου Openflow(έκδοση 1.3), έτσι ώστε κάθε πακέτο που έρχεται από το rpi να εισέρχεται και να εξέρχεται και από τα δύο VMs που έχουν τις δικτυακές μας λειτουργίες. Επομένως θα χρειαστούμε κάποια ρυθμισμένα flows στους br1 και br2.

**Στον μεταγωγέα br1:**

```
root@hostA:~# ovs-ofctl dump-ports br1
OFPST_PORT reply (xid=0x2): 4 ports
  port 10: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=6523, bytes=639254, drop=0, errs=0, coll=0
  port LOCAL: rx pkts=27630894, bytes=2571135732, drop=7091823, errs=0, frame=0, over=0, crc=0
           tx pkts=22237686, bytes=3078485427, drop=0, errs=0, coll=0
  port  1: rx pkts=105248054, bytes=9828409519, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=33387154, bytes=3972863593, drop=0, errs=0, coll=0
  port  2: rx pkts=151544, bytes=48767854, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=14280438, bytes=1222253965, drop=3693872, errs=0, coll=0
```

**Εικόνα 7.4 Θύρες μεταγωγέα br1**

Με την χρήση του εργαλείου ovs-ofctl και την εντολή **sudo ovs-ofctl -O OpenFlow13 add-flow br1 <flow>**, εκχωρούμε τις παρακάτω ροές:

Στον Πίνακα 0:

Προώθηση των πακέτων με προορισμό την 147.102.39.29 στην θύρα 1(που συνδέεται με το ens192):
**table=0,priority=1000,dl_type=0x0800,ip,ip_dst=147.102.39.29,actions=output:1**

Προώθηση των πακέτων που προέρχονται από την 147.102.40.69 με προορισμό την 8.8.4.4 στην θύρα 2(που συνδέεται με το VM1):
**table=0,priority=201,dl_type=0x0800,ip_src=147.102.40.69,ip_dst=8.8.4.4,in_port=LOCAL,actions=mod_dl_dst=52:54:00:1b:28:53,output=2**

Προώθηση των πακέτων που προέρχονται από την 147.102.40.69 με προορισμό την 8.8.8.8 στην θύρα 2(που συνδέεται με το VM1):
**table=0,priority=200,dl_type=0x0800,ip_src=147.102.40.69,ip_dst=8.8.8.8,in_port=LOCAL,actions=mod_dl_dst=52:54:00:1b:28:53,output=2**

Προώθηση των πακέτων που προέρχονται από το VM1 στον πίνακα 1:
**table=0,dl_type=0x0800,in_port=2,actions:output=goto_table:1**

Στον Πίνακα 1:

Προώθηση των πακέτων που προέρχονται από το VM1 στο LOCAL:
**table=1,priority=500,dl_type=0x0800,in_port=2,actions:output=LOCAL**

Προώθηση των πακέτων που προέρχονται από το VM1 με προορισμό την 8.8.4.4 στην θύρα VXLAN:
**table=1,priority=1000,dl_type=0x0800,in_port=2,nw_dst=8.8.4.4,actions:output=10**

Προώθηση των πακέτων που προέρχονται από το VM1 με προορισμό την 8.8.8.8 στην θύρα VXLAN:
**table=1,priority=1001,dl_type=0x0800,in_port=2,nw_dst=8.8.8.8,actions:output=10**

**Στον μεταγωγέα br2:**

```
root@hostB:~# ovs-ofctl dump-ports br2
OFPST_PORT reply (xid=0x2): 4 ports
  port 10: rx pkts=3504, bytes=343392, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=3356, bytes=194484, drop=0, errs=0, coll=0
  port LOCAL: rx pkts=73663, bytes=4435531, drop=81135, errs=0, frame=0, over=0, crc=0
           tx pkts=2810, bytes=157012, drop=0, errs=0, coll=0
  port  1: rx pkts=71923178, bytes=6397668746, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=95211, bytes=24098434, drop=0, errs=0, coll=0
  port  3: rx pkts=4969, bytes=1684942, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=94, bytes=8484, drop=0, errs=0, coll=0
```

**Εικόνα 7.5 Θύρες μεταγωγέα br2**

Και σε αυτή την περίπτωση, με την χρήση του εργαλείου ovs-ofctl και την εντολή **sudo ovs-ofctl -O OpenFlow13 add-flow br2 <flow>**, εκχωρούμε τις παρακάτω ροές:

Στον Πίνακα 0:

Προώθηση των εισερχόμενων πακέτων στην θύρα 1(ens192) με προορισμό την 147.102.39.29 στο LOCAL:
**table=0,dl_type=0x0800,in_port=1,ip_dst=147.102.39.29,actions:output=LOCAL**

Προώθηση των πακέτων που με προορισμό την 8.8.8.8 στην θύρα 3(που συνδέεται με το VM2):

**table=0,dl_type=0x0800,in_port=10,ip,nw_dst=8.8.4.4,actions=mod_dl_dst=52:54:00:36:db:dc,output:3**

Προώθηση των πακέτων που με προορισμό την 8.8.8.8 στην θύρα 3(που συνδέεται με το VM2):
**table=0,dl_type=0x0800,in_port=10,ip_dst=8.8.8.8,actions=mod_dl_dst=52:54:00:36:db:dc,output:3**

Προώθηση των πακέτων με προορισμό την 147.102.39.28 στην θύρα 10(VXLAN):
**table=0,priority=1000,dl_type=0x0800,ip,ip_dst=147.102.39.28,actions=output:10**

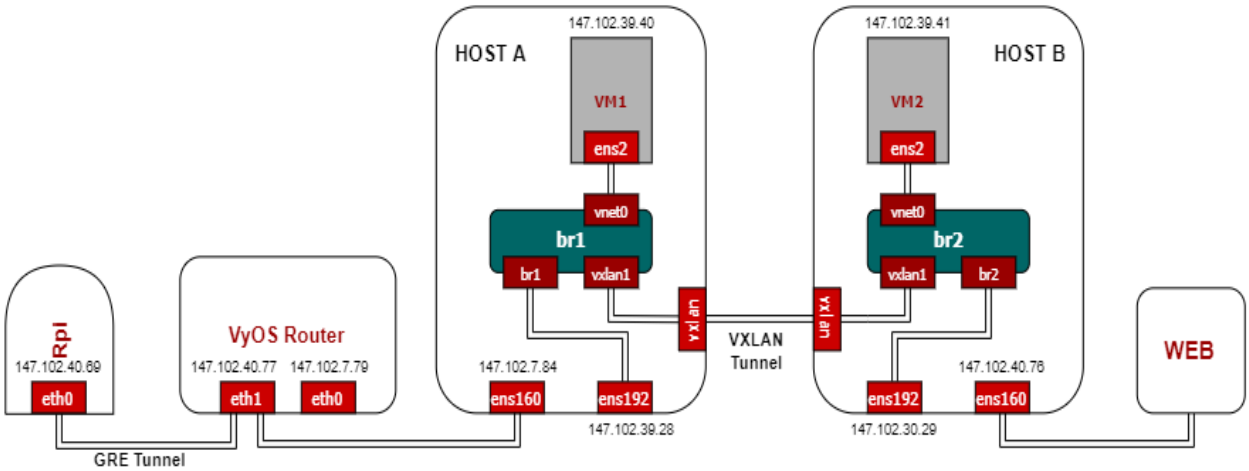
Στον Πίνακα 1:

Προώθηση των πακέτων που προέρχονται από το VM1 στον πίνακα 1:
**table=0,dl_type=0x0800,in_port=2,actions:output=goto_table:1**

Προώθηση των πακέτων με προορισμό την 8.8.4.4 στην θύρα 1(συνδεδεμένη με ens192):
**table=1,priority=105,dl_type=0x0800,in_port=3,ip_dst=8.8.4.4,actions:output=1**

Προώθηση των πακέτων με προορισμό την 8.8.4.4 στην θύρα 1(συνδεδεμένη με ens192):
**table=1,priority=5,dl_type=0x0800,in_port=3,ip_dst=8.8.8.8,actions:output=1**

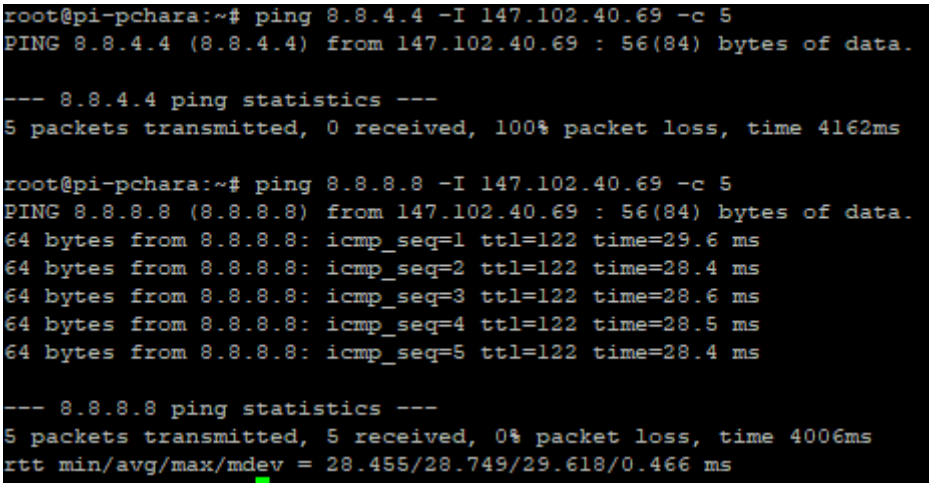
Μετά τις τελευταίες ρυθμίσεις η διάταξη είναι έτοιμη για δοκιμές.




**Εικόνα 7.6 Αναλυτική διάταξη υλοποίησης 1**

**Βήμα 10 Βασική Δοκιμή με Ping στους DNS Servers**

Για να επαληθεύσουμε ότι η υλοποίηση μας λειτουργεί θα εκτελεστεί η εντολή **ping από το Raspberry Pi προς τις IP 8.8.8.8 και 8.8.4.4**. Στην δεύτερη περίπτωση πρέπει να μην λαμβάνει απάντηση Echo Reply, όπως τελικώς επαληθεύεται:

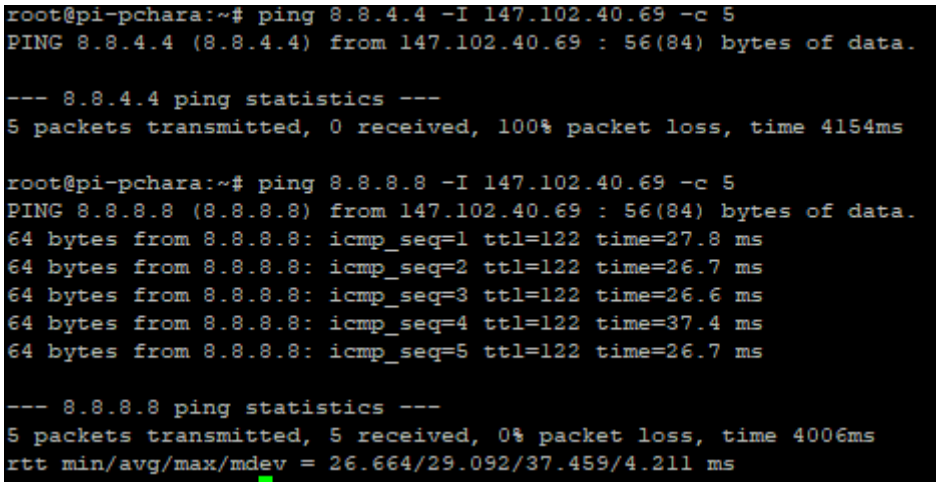



**Εικόνα 7.7 Ping Test στην διάταξη**

**Βήμα 11 Ταυτόχρονη δοκιμή διέλευσης από κάθε VM και HOST**

Για να δείξουμε την κίνηση μέσα σε κάθε μηχάνημα με την χρήση του εργαλείου tcpdump καταγράφουμε των πακέτων στις διεπαφές των μεταγωγέων br1 και br2 με φιλτράρισμα των πακέτων στην θύρα 4789 της αλυσίδας και στις διεπαφές vnet0 για την εισερχόμενη/εξερχόμενη κίνηση στα VM1/VM2 επαναλαμβάνοντας το ίδιο Ping Test:

**Ping Test στο Raspberry:**




**Εικόνα 7.8 2ο Ping Test στο Raspberry**

Επαναλαμβάνεται το ίδιο αποτέλεσμα με πριν προφανώς.

**Καταγραφή πακέτων στον br1 για την θύρα 4789:**

```
root@hostA:~# sudo tcpdump -i br1 port 4789
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on br1, link-type EN10MB (Ethernet), capture size 262144 bytes
01:29:57.619815 IP 147.102.39.28.59451 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 1, length 64
01:29:58.652674 IP 147.102.39.28.59451 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 2, length 64
01:29:59.692340 IP 147.102.39.28.59451 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 3, length 64
01:30:00.732159 IP 147.102.39.28.59451 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 4, length 64
01:30:01.772194 IP 147.102.39.28.59451 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 5, length 64
01:30:15.375364 IP 147.102.39.28.53312 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 1, length 64
01:30:16.377165 IP 147.102.39.28.53312 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 2, length 64
01:30:17.378582 IP 147.102.39.28.53312 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 3, length 64
01:30:18.385334 IP 147.102.39.28.53312 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 4, length 64
01:30:19.381122 IP 147.102.39.28.53312 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 5, length 64
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

**Εικόνα 7.9 Tcpdump στον br1**

**Καταγραφή πακέτων στον br2 για την θύρα 4789:**

```
root@hostB:~# sudo tcpdump -i br2 port 4789
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on br2, link-type EN10MB (Ethernet), capture size 262144 bytes
01:30:43.590427 IP 147.102.39.28.59451 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 1, length 64
01:30:44.623076 IP 147.102.39.28.59451 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 2, length 64
01:30:45.662705 IP 147.102.39.28.59451 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 3, length 64
01:30:46.702483 IP 147.102.39.28.59451 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 4, length 64
01:30:47.742534 IP 147.102.39.28.59451 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 5, length 64
01:31:01.346868 IP 147.102.39.28.53312 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 1, length 64
01:31:02.348250 IP 147.102.39.28.53312 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 2, length 64
01:31:03.349667 IP 147.102.39.28.53312 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 3, length 64
01:31:04.356462 IP 147.102.39.28.53312 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 4, length 64
01:31:05.352309 IP 147.102.39.28.53312 > 147.102.39.29.4789: VXLAN, flags [I] (0x08), vni 42
IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 5, length 64
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

**Εικόνα 7.10 Tcpdump στον br2**

Από τις δύο παραπάνω καταγραφές φαίνεται ότι λειτουργεί το VXLAN Tunnel.

Καταγραφή πακέτων στο vnet0 του HOST A:

```
root@hostA:~# tcpdump -i vnet0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on vnet0, link-type EN10MB (Ethernet), capture size 262144 bytes
01:29:57.618106 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 1, length 64
01:29:57.619651 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 1, length 64
01:29:58.651845 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 2, length 64
01:29:58.652639 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 2, length 64
01:29:59.691830 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 3, length 64
01:29:59.692297 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 3, length 64
01:30:00.527663 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from 52:54:00:1b:28:53 (oui Unknown),
th 300
01:30:00.731677 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 4, length 64
01:30:00.732126 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 4, length 64
01:30:01.771557 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 5, length 64
01:30:01.772149 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 5, length 64
01:30:13.906233 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from 52:54:00:1b:28:53 (oui Unknown),
th 300
01:30:15.374570 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 1, length 64
01:30:15.375125 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 1, length 64
01:30:16.376352 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 2, length 64
01:30:16.377117 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 2, length 64
01:30:17.378041 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 3, length 64
01:30:17.378535 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 3, length 64
01:30:18.384815 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 4, length 64
01:30:18.385289 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 4, length 64
01:30:19.380723 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 5, length 64
01:30:19.381078 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 5, length 64
^C
```

**Εικόνα 7.11 Tcpdump στο vnet0 του HOST A**

**Καταγραφή πακέτων στο vnet0 του HOST B:**

```
root@hostB:~# tcpdump -i vnet0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on vnet0, link-type EN10MB (Ethernet), capture size 262144 bytes
01:30:43.590756 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 1, length 64
01:30:44.623158 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 2, length 64
01:30:44.650707 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from 52:54:00:36:db:dc (oui Unknown),
length 300
01:30:45.662789 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 3, length 64
01:30:46.702620 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 4, length 64
01:30:47.742630 IP pi2.cn.ece.ntua.gr > google-public-dns-b.google.com: ICMP echo request, id 15652, seq 5, length 64
01:30:55.886977 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from 52:54:00:36:db:dc (oui Unknown),
length 300
01:31:01.347135 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 1, length 64
01:31:01.347861 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 1, length 64
01:31:02.348360 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 2, length 64
01:31:02.348704 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 2, length 64
01:31:03.349740 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 3, length 64
01:31:03.350319 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 3, length 64
01:31:04.356541 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 4, length 64
01:31:04.356908 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 4, length 64
01:31:05.232717 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from 52:54:00:36:db:dc (oui Unknown),
length 300
01:31:05.352439 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 5, length 64
01:31:05.353133 IP pi2.cn.ece.ntua.gr > google-public-dns-a.google.com: ICMP echo request, id 15653, seq 5, length 64
^C
```

**Εικόνα 7.12 Tcpdump στο vnet0 του HOST B**

Από τις παραπάνω καταγραφές επιβεβαιώνεται ότι η κίνηση προς τους δύο DNS Servers εισέρχεται και εξέρχεται κανονικά από το VM1, ενώ στο VM2 η κίνηση προς τον 8.8.4.4 μπλοκάρεται και δεν εξέρχεται από αυτό, ενώ προς το 8.8.8.8 εξέρχεται κανονικά επιβεβαιώνοντας την λειτουργία του firewall.

Για την διαδικασία logging των πακέτων που απορρίφθηκαν με τον τρόπο που αναφέραμε πριν επιβεβαιώνεται:

```
root@vm2:~# cd /var/log
root@vm2:/var/log# dmesg | grep IP_Dropped
[ 1528.408699] IP_DroppedIN=ens2 OUT=ens2 MAC=52:54:00:36:db:dc:52:54:00:1b:28:53:08:00 SRC=147.102.40
.69 DST=8.8.4.4 LEN=84 TOS=0x00 PREC=0x00 TTL=60 ID=53611 DF PROTO=ICMP TYPE=8 CODE=0 ID=6515 SEQ=1
[ 1529.438046] IP_DroppedIN=ens2 OUT=ens2 MAC=52:54:00:36:db:dc:52:54:00:1b:28:53:08:00 SRC=147.102.40
.69 DST=8.8.4.4 LEN=84 TOS=0x00 PREC=0x00 TTL=60 ID=53635 DF PROTO=ICMP TYPE=8 CODE=0 ID=6515 SEQ=2
[ 1530.477979] IP_DroppedIN=ens2 OUT=ens2 MAC=52:54:00:36:db:dc:52:54:00:1b:28:53:08:00 SRC=147.102.40
.69 DST=8.8.4.4 LEN=84 TOS=0x00 PREC=0x00 TTL=60 ID=53719 DF PROTO=ICMP TYPE=8 CODE=0 ID=6515 SEQ=3
[ 1531.518137] IP_DroppedIN=ens2 OUT=ens2 MAC=52:54:00:36:db:dc:52:54:00:1b:28:53:08:00 SRC=147.102.40
.69 DST=8.8.4.4 LEN=84 TOS=0x00 PREC=0x00 TTL=60 ID=53760 DF PROTO=ICMP TYPE=8 CODE=0 ID=6515 SEQ=4
```

**Εικόνα 7.13 Logs των πακέτων που απορρίφθηκαν στο Firewall**

**Βήμα 12 Ανάλυση πακέτου VXLAN**

Με την χρήση του tcpdump καταγράφουμε πακέτα από τις διεπαφές των HOST A και HOST B με προορισμό τον 8.8.8.8(προφανώς το ίδιο θα ισχύει και για τον 8.8.4.4) για να αναλύσουμε την ενθυλάκωση του VXLAN.

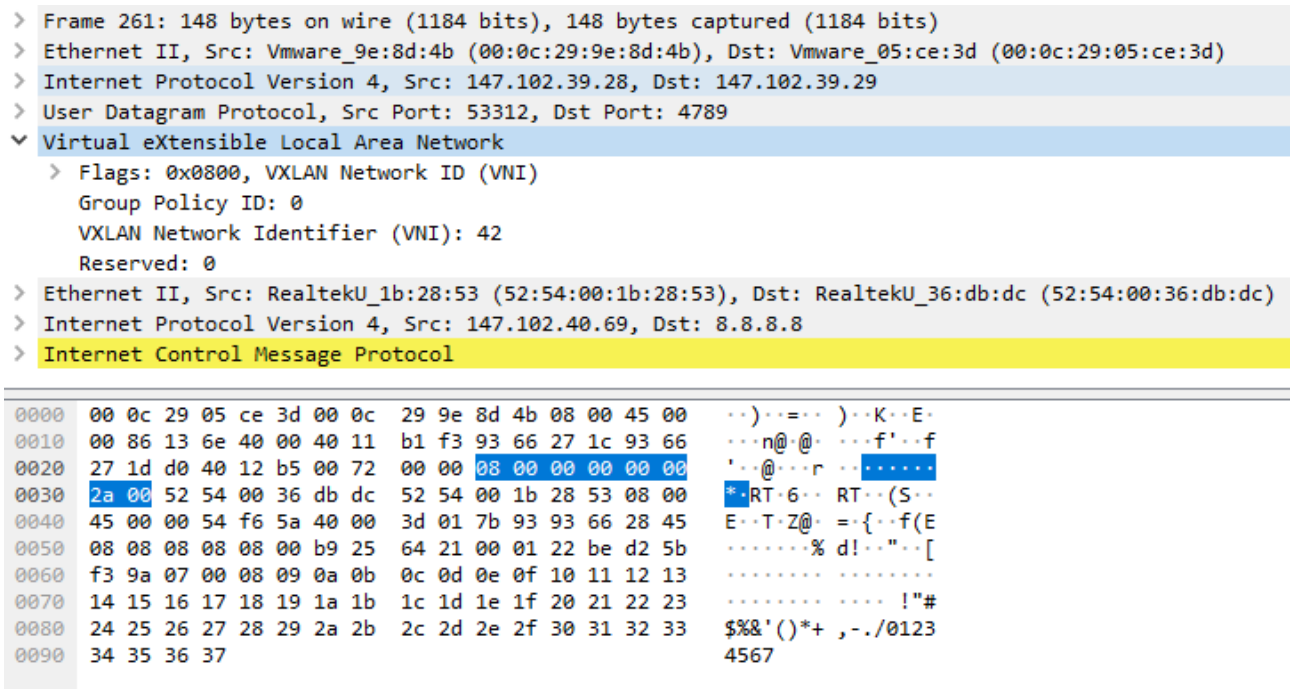
**Πακέτο από τον HOST A στον HOST B:**

```
> Frame 1: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits)
> Ethernet II, Src: Vmware_9e:8d:4b (00:0c:29:9e:8d:4b), Dst: Vmware_05:ce:3d (00:0c:29:05:ce:3d)
> Internet Protocol Version 4, Src: 147.102.39.28, Dst: 147.102.39.29
> User Datagram Protocol, Src Port: 53312, Dst Port: 4789
v Virtual eXtensible Local Area Network
  > Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 42
    Reserved: 0
> Ethernet II, Src: RealtekU_1b:28:53 (52:54:00:1b:28:53), Dst: RealtekU_36:db:dc (52:54:00:36:db:dc)
> Internet Protocol Version 4, Src: 147.102.40.69, Dst: 8.8.8.8
> Internet Control Message Protocol
```

```
0000  00 0c 29 05 ce 3d 00 0c  29 9e 8d 4b 08 00 45 00   ··)··=·· )··K··E·
0010  00 86 13 6e 40 00 40 11  b1 f3 93 66 27 1c 93 66   ···n@·@· ···f'··f
0020  27 1d d0 40 12 b5 00 72  00 00 08 00 00 00 00 00   '··@···r ········
0030  2a 00 52 54 00 36 db dc  52 54 00 1b 28 53 08 00   *·RT·6·· RT··(S··
0040  45 00 00 54 f6 5a 40 00  3d 01 7b 93 93 66 28 45   E··T·Z@· =·{··f(E
0050  08 08 08 08 08 00 b9 25  64 21 00 01 22 be d2 5b   ·······% d!··"··[
0060  f3 9a 07 00 08 09 0a 0b  0c 0d 0e 0f 10 11 12 13   ···· ···· ····
0070  14 15 16 17 18 19 1a 1b  1c 1d 1e 1f 20 21 22 23   ···· ···· ···· !"#
0080  24 25 26 27 28 29 2a 2b  2c 2d 2e 2f 30 31 32 33   $%&'()*+ ,-./0123
0090  34 35 36 37                                       4567
```

**Εικόνα 7.14 Απεικόνιση στο Wireshark VXLAN πακέτου(1)**

**Πακέτο που προωθείται στον HOST B:**




**Εικόνα 7.15 Απεικόνιση στο Wireshark VXLAN πακέτου(2)**

Όπως διακρίνουμε στην ανάλυση των πακέτων και τα δυο πακέτα έχουν τις τιμές εξωτερικά των IP και των MAC διευθύνσεων των δύο VTEP, ενώ είναι ενθυλακωμένες και οι διευθύνσεις πηγής(147.102.40.69) και προορισμού(8.8.8.8) του ping πριν το πακέτο icmp.

Στην ενθυλάκωση VXLAN χρησιμοποιούνται 8 Bytes:

- 2 bytes για τον τύπο του πρωτοκόλλου σήραγγας, για να δείξουν δηλαδή ότι χρησιμοποιούμε VXLAN με τιμή 0800.
- 2 bytes για την Group Policy αναγνωριστικό(ID) που στην δική μας περίπτωση είναι η τιμή του 0000.
- 3 bytes για το VNI, το αναγνωριστικό του VXLAN overlay, που έχει την τιμή 00002a που παραπέμπει στον αριθμό 42 που τον ορίσαμε παραπάνω.
- 1 byte για την τιμή Reserved που εδώ είναι 00.

### 7.2) Υλοποίηση 2

**Βήμα 1 Εγκατάσταση και παραμετροποίηση των εικονικών μηχανών**

Η αρχιτεκτονική του συστήματος με SFC περιγράφηκε στο 5ο κεφάλαιο, όπως και οι εικονικές μηχανές που θα χρειαστούν. Στον παρακάτω πίνακα φαίνεται τα βασικά στοιχεία όσων αφορά το hardware και τις διεπαφές των VM που θα εγκαταστήσουμε.

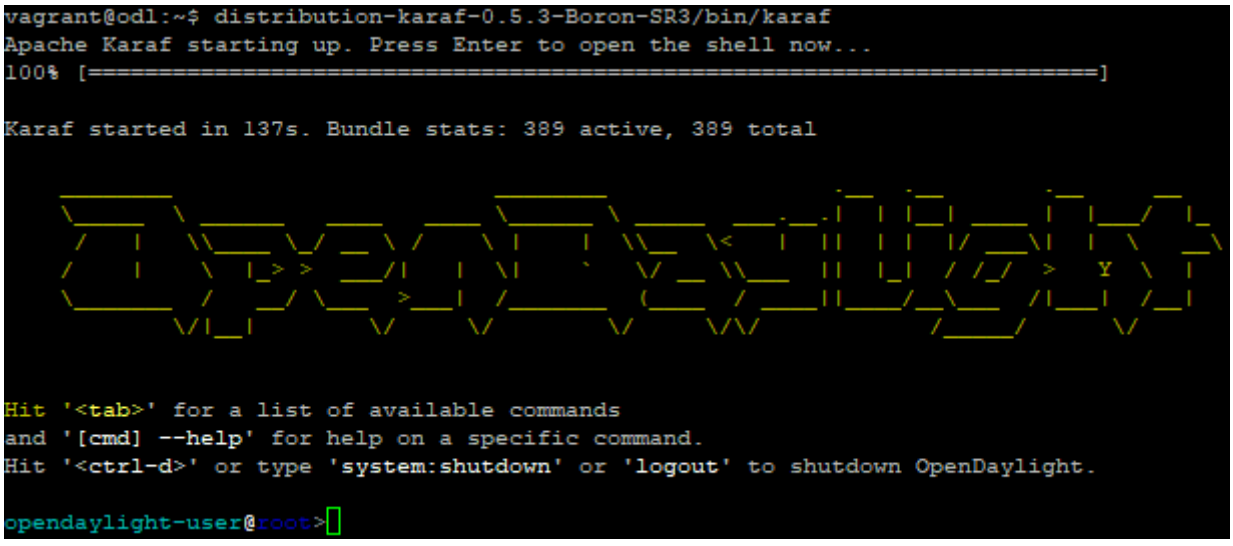
| VM | Διεπαφή eth1 | Μνήμη RAM(MB) | CPUs | Σκληρός δίσκος |
|---|---|---|---|---|
| odl | 192.168.1.5 | 4096 | 2 | Δυναμικό μέγεθος |
| sf1 | 192.168.1.30 | 1048 | 1 | Δυναμικό μέγεθος |
| sf2 | 192.168.1.40 | 1048 | 1 | Δυναμικό μέγεθος |
| sff1 | 192.168.1.20 | 1048 | 1 | Δυναμικό μέγεθος |
| sff2 | 192.168.1.50 | 1048 | 1 | Δυναμικό μέγεθος |
| classifier1 | 192.168.1.10 | 1048 | 1 | Δυναμικό μέγεθος |
| classifier2 | 192.168.1.60 | 1048 | 1 | Δυναμικό μέγεθος |

**Πίνακας 2 Χαρακτηριστικά VM για την διάταξη SFC**

Για την εγκατάστασή τους χρησιμοποιούμε το εργαλείο vagrant με την βοήθεια του virtual box, και χρησιμοποιώντας την εντολή **vagrant up** εγκαθιστά τα VMs με βάση το αρχείο Vagrantfile σε Ruby που έχουμε τροποποιήσει καταλλήλως και βρίσκεται στο Παράρτημα.

**Βήμα 2 Εγκατάσταση ODL**

Για να μπορούμε να τρέξουμε την πλατφόρμα OpenDaylight πρέπει να εγκαταστήσουμε πρώτα την κατάλληλη έκδοση της JAVA και του εργαλείου Apache Maven (βλ. Παράρτημα)




**Εικόνα 7.16 Έναρξη και Είσοδος στην Πλατφόρμα ODL**

Στο προηγούμενο σχήμα φαίνεται η είσοδος στον ODL και το terminal του που χρησιμοποιεί το Karaf Shell. Το ODL δεν έχει εγκατεστημένα όλα τα features παρά μόνο τα απολύτως απαραίτητα. Για την υλοποίηση μας θα εγκαταστήσουμε όλα τα σχετικά features που είναι σχετικά με SFC, OVSDB και Openflow καθώς και της βασικές λειτουργίες του ODL.


**Εικόνα 7.17 Παράδειγμα εγκατάστασης feature στον ODL**

Μετά την εγκατάσταση των απαραίτητων feature ο ελεγκτής είναι έτοιμος για χρήση.

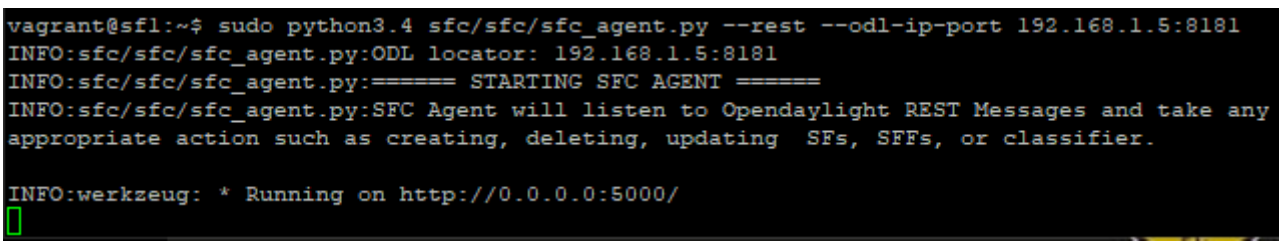
**Βήμα 3 Ρυθμίσεις στα VM με τους SFF**

Τον ρόλο των SFF στην αρχιτεκτονική μας τον αναλαμβάνουν οι μεταγωγείς OVS, με την χρήση ενός patch για να είναι συμβατοί με την ενθυλάκωση NSH. Σε κάθε VM(sff1 και sff2) θα εγκαταστήσουμε το OVS με το patch και θα δημιουργήσουμε έναν μεταγωγέα OVS με όνομα br-sfc σε κάθε VM, που θα λειτουργεί με το πρωτόκολλο Openflow(έκδοση 1.3) και θα ελέγχεται από τον ODL.

**Βήμα 4 Ρυθμίσεις στα VM με τις SF**

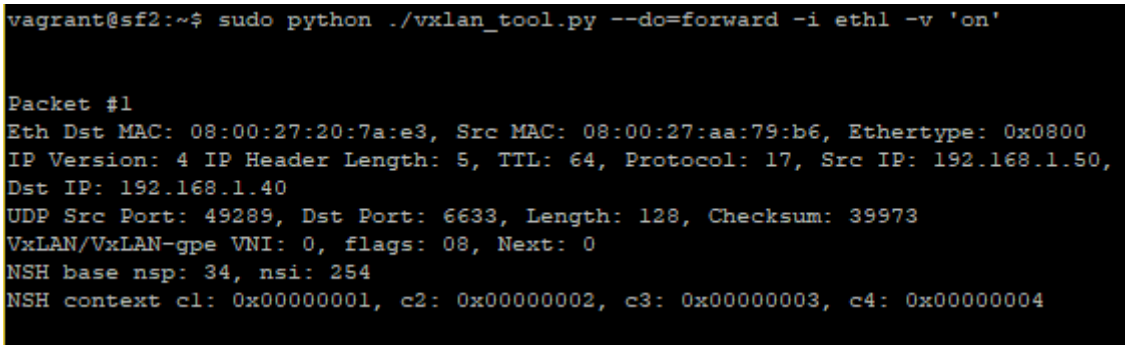
Στα VMs με τις SF, όπως αναλύθηκε και σε προηγούμενο κεφάλαιο θα χρησιμοποιήσουμε έναν sfc agent που έχει αναπτυχθεί σε γλώσσα Python και διανέμεται από τον ODL. Η χρήση του είναι για να εξομοιώνει dummy Service Functions και μαζί με το εργαλείο vxlan_tool.py επιτυγχάνουν την προώθηση των NSH πακέτων πίσω στους SFF και στην ουσία αντικαθιστούν την λειτουργία του VNF Manager που θα μπορούσε να υλοποιηθεί αλλιώς με Openstack (βλ. Κεφάλαιο 8) το οποίο απαιτεί πολύ μεγάλους υπολογιστικούς πόρους.

Ο SFC Agent επικοινωνεί με τον ODL δεχόμενος μηνύματα στην θύρα 5000, με τα οποία ενημερώνεται για τους SFF που υπάρχουν στην διάταξη, δημιουργεί και διαγράφει dummy SFs.




**Εικόνα 7.18 Έναρξη SFC Agent**

Το vxlan_tool.py συμπληρωματικά με τον SFC Agent προωθεί τα πακέτα NSH στην διεπαφή που ενώνεται με τον αντίστοιχο SFF της SF που δημιουργούμε και στην ουσία αντικαθιστά το vxlan interface που χρειάζεται για να γίνει η μεταφορά πακέτων από και προς τον SFF. Ενεργοποιείται με την εντολή **sudo python ./vxlan_tool.py --do=forward -i eth1**, και δίνει και την δυνατότητα απεικόνισης των πακέτων NSH.




**Εικόνα 7.19 Απεικόνιση πακέτων NSH με vxlan_tool.py**

**Βήμα 5 Ρυθμίσεις στα VM με τους Classifiers**

Οι Classifiers στην αρχιτεκτονική του συστήματος, θα υλοποιηθούν πάνω σε μεταγωγείς OVS, οπότε σε κάθε VM(classifier1 και classifier2) χρειάζεται να δημιουργήσουμε από έναν μεταγωγέα br-sfc με τις ρυθμίσεις ακριβώς όπως και στους SFF.

Για την δημιουργία των client και server θα χρησιμοποιήσουμε το εργαλείο netns για την δημιουργία namespaces και εφαρμογών πάνω σ' αυτά. Ο Client(192.168.2.1) θα είναι στο classifier1 VM και ο Server(192.168.1.2) θα είναι στο classifier2 VM, όπου με την εγκατάσταση ενός http-deamon θα παρέχει υπηρεσία downloading στον client. (βλ. παράρτημα)

**Βήμα 6 Ρυθμίσεις για την δημιουργία της SFC**

Για την δημιουργία και ρύθμιση της SFC θα χρησιμοποιήσουμε τον ελεγκτή ODL με τις δυνατότητες που έχει για τον έλεγχο και την παραμετροποίηση των στοιχείων της αλυσίδας. Τα δεδομένα που χρειάζεται ο ελεγκτής θα δοθούν μέσω του REST API του, αφού τρέξουμε ένα python script το setup.py(βλ. παράρτημα) για να εκτελέσει αυτόματα τις εντολές REST μιας και δεν χρησιμοποιούμε την διεπαφή χρήστη του ODL, αφού είναι σε τοπική διεύθυνση IP. Οι εντολές REST που χρησιμοποιούμε είναι είτε PUT είτε POST και τα δεδομένα τα αποστέλλουμε σε μορφή JSON. Τα στοιχεία που χρειάζεται να ρυθμιστούν είναι:

**A) Service Nodes.**

Τα Service Nodes που χρειάζεται να ορίσουμε είναι στον παρακάτω πίνακα:

| Service Node | Διεύθυνση IP | Service Functions |
|---|---|---|
| Node0 | 192.168.1.10 | |
| Node1 | 192.168.1.20 | |
| Node2 | 192.168.1.30 | dpi-1 |
| Node3 | 192.168.1.40 | firewall-1 |
| Node4 | 192.168.1.50 | |
| Node5 | 192.168.1.60 | |

**Πίνακας 3 Χαρακτηριστικά των Service Nodes**

Στην ουσία σαν Service Node ορίζουμε κάθε VM που περιέχει ένα στοιχείο του SFC(SFs, SFFs και Classifiers) και ορίζονται με την εντολή PUT στο URI:
http://192.168.1.5:8181/restconf/config/service-node:service-nodes/

**B)Service Functions**

Τα σημεία που πρέπει να ρυθμίσουμε για την SF1 είναι:

- Όνομα: dpi-1
- Διεύθυνση IP: 192.168.1.30
- URI για επικοινωνία με ODL: http://192.168.1.30:5000
- Τύπος της SF: dpi

Αντίστοιχα, τα σημεία που πρέπει να ρυθμίσουμε για την SF1 είναι:

- Όνομα: firewall-1
- Διεύθυνση IP: 192.168.1.40
- URI για επικοινωνία με ODL: http://192.168.1.40:5000
- Τύπος της SF: firewall

Για την επικοινωνία των SF με τους SFF χρειάζεται η δημιουργία των data-plane-locators που στους SFF αποτελούν μια θύρα στο OVS ενώ στις SF αναλαμβάνει ο SFC Agent την εξομοίωση τους στην περίπτωσή μας.

Στην SF1 ο data-plane locator έχει τα εξής χαρακτηριστικά:

- Όνομα: sf1-dpl
- Θύρα: 6633
- Διεύθυνση IP: 192.168.1.30
- Τρόπος μεταφοράς: vxlan-gpe
- Συνδεδεμένος SFF: SFF1

Αντίστοιχα στην SF2:

- Όνομα: sf1-dpl
- Θύρα: 6633
- Διεύθυνση IP: 192.168.1.30
- Τρόπος μεταφοράς: vxlan-gape
- Συνδεδεμένος SFF: SFF1

Τα Service Functions ορίζονται με την εντολή PUT στο URI:
http://192.168.1.5:8181/restconf/config/service-function:service-functions/

**Γ)Service Function Forwarders**

Οι Classifiers Openflow για να δημιουργηθούν χρειάζονται την δημιουργία ενός SFF σε κάθε br-sfc στα αντίστοιχα VMs. Συνολικά οι SFFs που χρειαζόμαστε είναι στον πίνακα:

| | **SFF0** | **SFF1** | **SFF2** | **SFF3** |
|---|---|---|---|---|
| Service Node | node0 | node1 | node4 | node5 |
| OVS | br-sfc | br-sfc | br-sfc | br-sfc |
| **Πληροφορίες για τους αντίστοιχους data-plane-locators** | | | | |
| Όνομα | sff0-dpl | sff1-dpl | sff2-dpl | sff3-dpl |
| Τρόπος μεταφοράς | vxlan-gpe | vxlan-gpe | vxlan-gpe | vxlan-gpe |
| Θύρα | 6633 | 6633 | 6633 | 6633 |
| Διεύθυνση IP | 192.168.1.10 | 192.168.1.20 | 192.168.1.50 | 192.168.1.60 |
| Locator αντίστοιχης SF | | sf1-dpl | sf2-dpl | |

**Πίνακας 4 Χαρακτηριστικά των Service Function Forwarders**

Για την δημιουργία και ρύθμιση των Service Function Forwarders με εντολή PUT στο URI:
http://192.168.1.5:8181/restconf/config/service-function-forwarder:service-function-forwarders

**Δ)Service Function Chains**

Η αλυσίδα που θέλουμε να δημιουργήσουμε, η SFC1 προωθεί την κίνηση, πρώτα από την SF1 και μετά από την SF2. Η δημιουργία της γίνεται με ένα PUT στο URI:
http://192.168.1.5:8181/restconf/config/service-function-chain:service-function-chains/

**Ε)Service Function Paths**

Το Service Function Path που θα δημιουργήσουμε έχει τα εξής χαρακτηριστικά:

- Όνομα: SFP1
- Αλυσίδα: SFC1
- Η τιμή του starting index είναι 255
- Είναι συμμετρικό
- Ταξινομητής: Classifier1

- Συμμετρικός ταξινομητής: Classifier2
- Τα context-metadata του είναι τα NSH1 που θα οριστούν πιο κάτω.

Και το δημιουργούμε με PUT στο URI:
http://192.168.1.5:8181/restconf/config/service-function-path:service-function-paths/

**ΣΤ)Metadata**

Τα στοιχεία που πρέπει να ρυθμίσουμε για τα metadata είναι:

- Όνομα: NSH1
- context-header1: 1
- context-header2: 2
- context-header3: 3
- context-header4: 4

Εισαγωγή των στοιχείων των Service Function Metadata με PUT στο URI:
http://192.168.1.5:8181/restconf/config/service-function-path-metadata:service-function-metadata/

**Ζ)Rendered Service Paths**

Για το αντίστοιχο SFP πρέπει να δημιουργήσουμε και το αντίστοιχο RSP το οποίο όμως θέλουμε να είναι και συμμετρικό και να έχει το όνομα RSP1(RSP1-Reverse το συμμετρικό). Αυτό γίνεται με POST στο URI για την δημιουργία των Rendered Service Paths:
http://192.168.1.5:8181/restconf/operations/rendered-service-path:create-rendered-path/

**Η)Access List**

Στους Classifiers κατά την είσοδο στην αλυσίδα θα πρέπει να δίνουν μία Access List(Λίστα πρόσβασης) που θα δίνει τους όρους για το ποια κίνηση μπορεί να περάσει την αλυσίδα και μέσω ποιου RSP. Στην περίπτωσή μας χρειαζόμαστε μία ACL για κάθε Classifier τις ACL1 και ACL2 με μία καταχώρηση ACE1 και ACE2 το καθένα αντίστοιχα:

| Access List | ACL1 | ACL2 |
|---|---|---|
| Όνομα καταχώρησης | ACE1 | ACE2 |
| Δίκτυο προορισμού | 192.168.2.0/24 | 192.168.2.0/24 |
| Δίκτυο πηγής | 192.168.2.0/24 | 192.168.2.0/24 |
| Τύπος πρωτόκολλου | 6 | 6 |
| Θύρες προορισμού | >80 | >0 |
| Θύρες πηγής | >0 | >80 |
| RSP | RSP1 | RSP1-Reverse |

**Πίνακας 5 Access Lists**

Για την δημιουργία και ρύθμιση των Access List PUT στο URI:
http://192.168.1.5:8181/restconf/config/ietf-access-control-list:access-lists/

**Θ)Classifiers**

Οι Openflow Classifiers θα δημιουργηθούν πάνω στις SFF0 και SFF3 που δημιουργήθηκαν για αυτόν τον λόγο. Συγκεκριμένα:

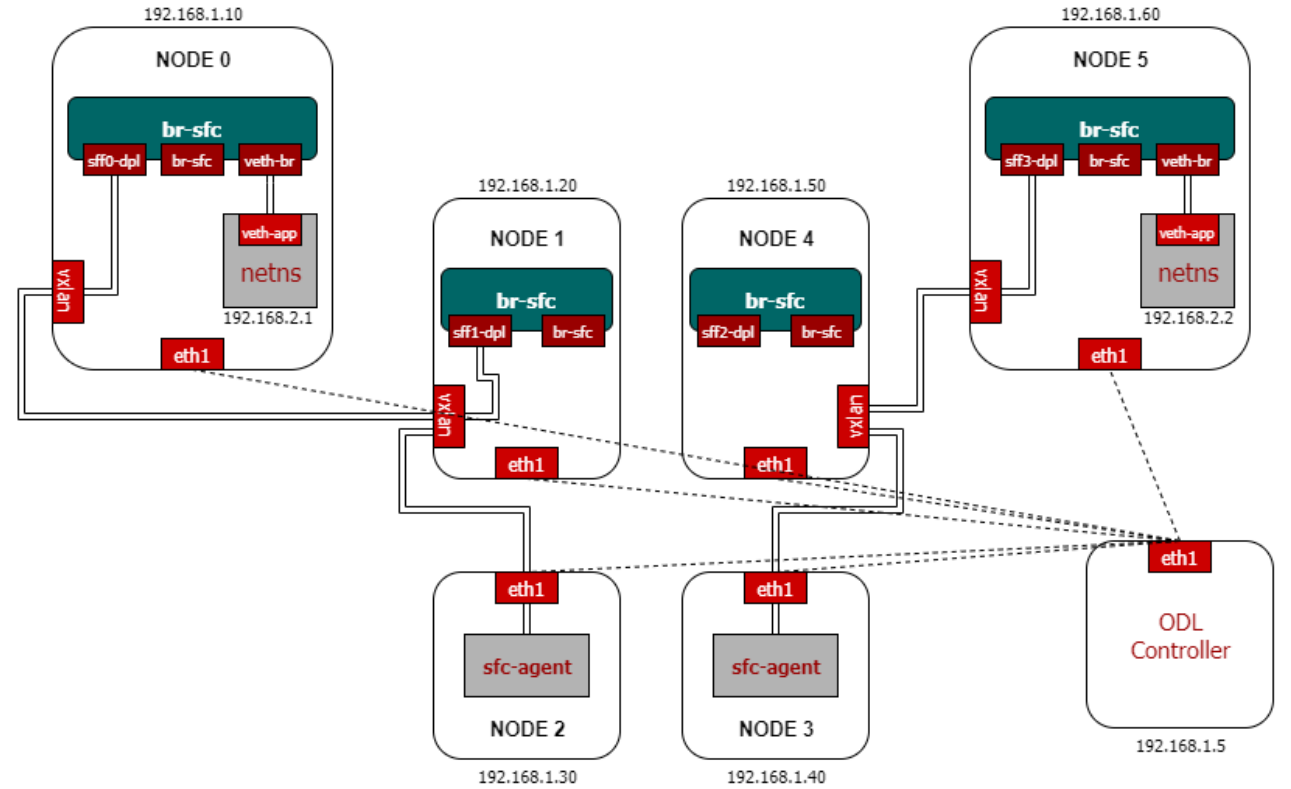
| Όνομα | **Classifier1** | **Classifier2** |
|---|---|---|
| SFF | SFF0 | SFF3 |
| Διεπαφή | veth-br | veth-br |
| ACL | ACL1 | ACL2 |

**Πίνακας 6 Χαρακτηριστικά των Classifiers**

URI για την δημιουργία και ρύθμιση των Openflow Classifiers:
http://192.168.1.5:8181/restconf/config/service-function-classifier:service-function-classifiers/

**Βήμα 7 Αποστολή ρυθμίσεων στον ελεγκτή και δημιουργία συστήματος SFC**

Με την χρήση του setup.py(βλ. παράρτημα) στέλνουμε τις απαραίτητες ρυθμίσεις στον ελεγκτή ODL μέσω του REST API και δημιουργείτε έτσι το σύστημα SFC.




**Εικόνα 7.20 Αναλυτική διάταξη υλοποίησης 2**

**Βήμα 8 Επιβεβαίωση λειτουργίας της αλυσίδας**

Αφού στείλουμε τις εντολές REST παρατηρούμε την δημιουργία των SF1 και SF2 από τους SFC Agents:

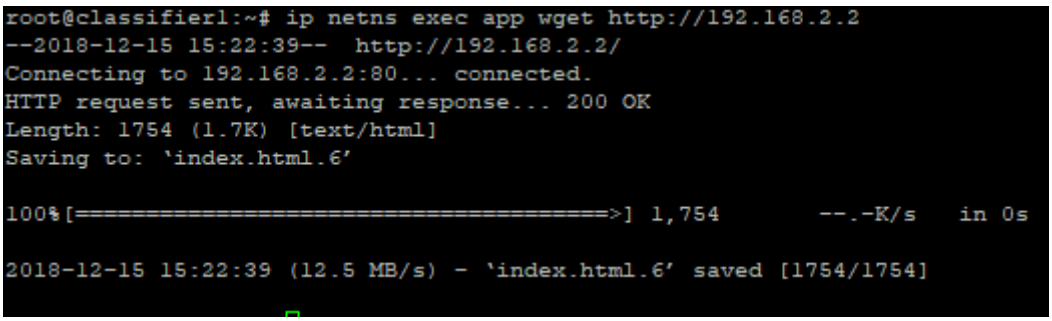
```
vagrant@sf1:~$ sudo python3.4 sfc/sfc/sfc_agent.py --rest --odl-ip-port 192.168.
1.5:8181
INFO:sfc/sfc/sfc_agent.py:ODL locator: 192.168.1.5:8181
INFO:sfc/sfc/sfc_agent.py:====== STARTING SFC AGENT ======
INFO:sfc/sfc/sfc_agent.py:SFC Agent will listen to Opendaylight REST Messages an
d take any appropriate action such as creating, deleting, updating  SFs, SFFs, o
r classifier.

INFO:werkzeug: * Running on http://0.0.0.0:5000/
INFO:sfc/sfc/sfc_agent.py:Received request for SF creation: dpi-1
INFO:sfc/sfc/sfc_agent.py:Received request from ODL to create SF ...
INFO:sfc.common.launcher:Starting Service Function: dpi-1
INFO:sfc.common.launcher:Starting DPI serving as dpi-1 at 192.168.1.30:6633, ser
vice type:dpi
INFO:sfc.common.launcher:Starting control UDP server for dpi-1 at 192.168.1.30:6
000
INFO:werkzeug:192.168.1.5 - - [15/Dec/2018 15:21:49] "PUT /config/service-functi
on:service-functions/service-function/dpi-1 HTTP/1.1" 201 -
```

**Εικόνα 7.21 Επιτυχής δημιουργία SF1**

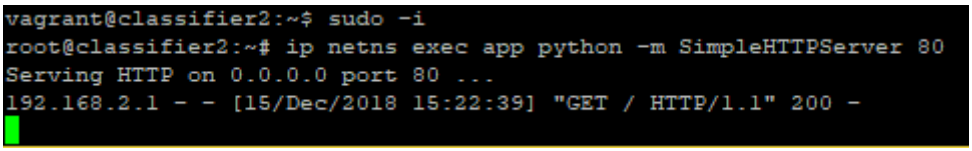
```
vagrant@sf2:~$ sudo python3.4 sfc/sfc/sfc_agent.py --rest --odl-ip-port 192.168.
1.5:8181
INFO:sfc/sfc/sfc_agent.py:ODL locator: 192.168.1.5:8181
INFO:sfc/sfc/sfc_agent.py:====== STARTING SFC AGENT ======
INFO:sfc/sfc/sfc_agent.py:SFC Agent will listen to Opendaylight REST Messages an
d take any appropriate action such as creating, deleting, updating  SFs, SFFs, o
r classifier.

INFO:werkzeug: * Running on http://0.0.0.0:5000/
INFO:sfc/sfc/sfc_agent.py:Received request for SF creation: firewall-1
INFO:sfc/sfc/sfc_agent.py:Received request from ODL to create SF ...
INFO:sfc.common.launcher:Starting Service Function: firewall-1
INFO:sfc.common.launcher:Starting FIREWALL serving as firewall-1 at 192.168.1.40
:6633, service type:firewall
INFO:sfc.common.launcher:Starting control UDP server for firewall-1 at 192.168.1
.40:6000
INFO:werkzeug:192.168.1.5 - - [15/Dec/2018 15:21:49] "PUT /config/service-functi
on:service-functions/service-function/firewall-1 HTTP/1.1" 201 -
```

**Εικόνα 7.22 Επιτυχής δημιουργία SF2**

Στην συνέχεια επιβεβαιώνεται η λειτουργία της αλυσίδας συμμετρικά με την αποστολή ενός αιτήματος http για το downloading ενός αρχείου από τον client(192.168.2.1) στον http server(192.168.2.2) και η διαδικασία αυτή γίνεται επιτυχώς κατεβάζοντας το αρχείο.




**Εικόνα 7.23 Αποστολή αιτήματος από τον client και επιτυχές κατέβασμα αρχείου**




**Εικόνα 7.24 Αποδοχή αιτήματος από τον server**

**Βήμα 9 Ανάλυση των ροών σε SFFs και Classifiers**

Ο ελεγκτής ODL με αυτόματο τρόπο στέλνει στα OVS της διάταξης flows, ώστε να μπορούν να λειτουργήσουν ως SFF και ως Classifiers με βάση τα δεδομένα που εισάγαμε στο βήμα 7. Στους SFF διενεργείτε η διαδικασία που επεξηγήθηκε στο κεφάλαιο 4 για τα flows και τους πίνακες των αντίστοιχων br-sfc για κάθε SFF.

Οι τιμές που δόθηκαν από τον ελεγκτή για τα RSPs είναι:

- RSP1:34
- RSP1-Reverse: 8388642
- Αρχική τιμή NSI(Service Chain Hop):255

Τέλος πριν προχωρήσουμε την ανάλυση των flows, να επισημάνουμε ότι η χρήση του NSH γίνεται με την χρήση VXLAN-GPE.

**Flows του Classifier1:**

Ενθυλάκωση NSH των πακέτων που πληρούν τις προδιαγραφές της ACL1 για το RSP1 και προώθηση τους στην θύρα 6(στον SFF1):
cookie=0x0, table=0,
priority=1000,tcp,in_port=1,nw_src=192.168.2.0/24,nw_dst=192.168.2.0/24,tp_dst=80
actions=push_nsh,load:0x1->NXM_NX_NSH_MDTYPE[],load:0x3->NXM_NX_NSH_NP[],load:0x22->NXM_NX_NSP[0..23],load:0xff->NXM_NX_NSI[],load:0x1->NXM_NX_NSH_C1[],load:0x2->NXM_NX_NSH_C2[],load:0x3->NXM_NX_NSH_C3[],load:0x4->NXM_NX_NSH_C4[],load:0x4->NXM_NX_TUN_GPE_NP[],load:0xc0a80114->NXM_NX_TUN_IPV4_DST[],output:6

Τερματισμός της ενθυλάκωσης NSH για πακέτα που έρχονται από τον SFF1 και χρησιμοποίησαν τον RSP1-Reverse και προώθηση τους στην θύρα 1(client):
cookie=0x0, table=0, priority=1000,nsi=253,nsp=8388642 actions=pop_nsh,output:1

**Flows του SFF1:**

Πίνακας 0(Classifier)

Προώθηση πακέτων που RSP1-Reverse που προέρχονται από την SF1 και προωθούνται στον Classifier1:
cookie=0x0, table=0, priority=1000,nsi=253,nsp=8388642 actions=load:0x4->NXM_NX_TUN_GPE_NP[],load:0xc0a8010a->NXM_NX_TUN_IPV4_DST[],move:NXM_NX_NSP[0..23]->NXM_NX_NSP[0..23],move:NXM_NX_NSI[]->NXM_NX_NSI[],move:NXM_NX_NSH_C1[]->NXM_NX_NSH_C1[],move:NXM_NX_NSH_C2[]->NXM_NX_NSH_C2[],IN_PORT

Προώθηση όλων των πακέτων στον Πίνακα 1, μιας και η διαδικασία ταξινόμησης για το RSP1 γίνεται στον Classifier1:
cookie=0x14, table=0, priority=5 actions=goto_table:1

Πίνακας 1(Transport Ingress)

Προώθηση των πακέτων του RSP1 στον Πίνακα 4:
cookie=0x14, table=1, priority=250,nsp=34 actions=goto_table:4

Προώθηση των πακέτων του RSP1 στον Πίνακα 4:
cookie=0x14 , table=1, priority=250,nsp=8388642 actions=goto_table:4

Διακοπή της κίνησης σε οποιαδήποτε άλλη περίπτωση:
cookie=0x14, table=1, priority=5 actions=drop

Πίνακας 2(Path Mapper)

Ο Πίνακας 2 δεν χρησιμοποιείται στην περίπτωση χρήσης NSH με VXLAN-GPE στην SFC:
cookie=0x14, table=2, priority=5 actions=goto_table:3

Πίνακας 3(Path Mapper ACL)

Ο Πίνακας 3 δεν χρησιμοποιείται στην περίπτωση χρήσης NSH με VXLAN-GPE στην SFC:
cookie=0x14, table=3, priority=5 actions=goto_table:4

Πίνακας 4(Next Hop)

Ο πίνακας 4 καθορίζει πού πρέπει να αποσταλούν τα πακέτα SFC στη συνέχεια, είτε στην συνδεδεμένη SF είτε σε άλλο SFF. Υπάρχει αντιστοίχιση τόσο με το NSP (αναγνωριστικό αλυσίδας υπηρεσίας) όσο και με το NSI (hop chain service) για να προσδιοριστεί το επόμενο hop. Μετά προωθούνται τα πακέτα στον Πίνακα 10:

Για πακέτα της RSP1, που πρέπει να προωθηθούν στην SF1:
cookie=0x14, table=4, priority=550,nsi=255,nsp=34 actions=load:0xc0a8011e->NXM_NX_TUN_IPV4_DST[],goto_table:10

Για πακέτα της RSP1, που πρέπει να προωθηθούν στον SFF2:
cookie=0x14, table=4, priority=550,nsi=254,nsp=34 actions=load:0xc0a80132->NXM_NX_TUN_IPV4_DST[],goto_table:10

Για πακέτα της RSP1-Reverse, που πρέπει να προωθηθούν στην SF1:
cookie=0x14, table=4, priority=550,nsi=254,nsp=8388642 actions=load:0xc0a8011e->NXM_NX_TUN_IPV4_DST[],goto_table:10

Σε οποιαδήποτε άλλη περίπτωση:
cookie=0x14, table=4, priority=5 actions=goto_table:10

Πίνακας 10(Transport Egress)

Ο πίνακας 10 προετοιμάζει πακέτα για έξοδο ρυθμίζοντας πληροφορίες της σήραγγας VXLAN-GPE. Αυτές οι ροές καθορίζουν επίσης τη θύρα εξόδου όπου πρέπει να αποστέλλονται τα πακέτα. Οι ροές VXLAN GPE NSH TransportEgress είναι πιο περίπλοκες από τις υπόλοιπες και αναγνωρίζονται από τις τιμές cookie τους:

- 0xba5eba1100000101 - TRANSPORT_EGRESS_NSH_VXGPE_COOKIE
- 0xba5eba1100000102 - TRANSPORT_EGRESS_NSH_VXGPE_NSC_COOKIE
- 0xba5eba1100000103 - TRANSPORT_EGRESS_NSH_VXGPE_LASTHOP_COOKIE

Όπως προκύπτει από τα flows του VXLAN-GPE NSH παρακάτω, όλα τα flows NSH Transport Egress ταιριάζουν τουλάχιστον στο NSP(αναγνωριστικό αλυσίδας υπηρεσιών) και στην NSI (hop στην αλυσίδα). Ορισμένα flows ταιριάζουν με το in_port και στη συνέχεια εξάγουν τα πακέτα σε IN_PORT, ενώ άλλα φαινομενικά ίδια εξάγουν τα πακέτα σε μια συγκεκριμένη θύρα χωρίς να ταιριάζουν στο in_port. Αυτές οι ροές είναι πράγματι ακριβώς ίδιες, εκτός από τις προαναφερθείσες διαφορές και τις προτεραιότητες των flows. Αυτό συμβαίνει επειδή σύμφωνα με την προδιαγραφή του OpenFlow, ο μόνος τρόπος που ένα πακέτο μπορεί να αποσταλεί στην ίδια θύρα που λαμβάνεται, είναι να το αποστέλλει εκουσίως χρησιμοποιώντας τη συμβολοσειρά θύρας IN_PORT ή να μπλοκαριστεί, προσπαθώντας να αποφύγει βρόχους πακέτων.

Απαιτείται κάποια επιπλέον λογική για το τελευταίο hop, το οποίο είναι όταν τα πακέτα έχουν διασχίσει ολόκληρη την αλυσίδα υπηρεσιών και πρέπει να βγουν από το SFC. Οι πληροφορίες για

το σημείο αποστολής του πακέτου μετά το SFC καθορίζονται στις κεφαλίδες μεταδεδομένων C1 και C2 NSH από τον ταξινομητή SFC. Η επικεφαλίδα C1 είναι η διεύθυνση IPv4 της σήραγγας VXLAN-GPE και το C2 είναι το πεδίο VXLAN-GPE VNI.

cookie=0xba5eba1100000102, table=10, priority=660,nsi=253,nsp=8388642,nshc1=0
actions=IN_PORT

cookie=0xba5eba1100000101, table=10, priority=650,nsi=255,nsp=34
actions=move:NXM_NX_NSH_MDTYPE[]->NXM_NX_NSH_MDTYPE[],
move:NXM_NX_NSH_NP[]->NXM_NX_NSH_NP[],
move:NXM_NX_NSH_C1[]->NXM_NX_NSH_C1[],
move:NXM_NX_NSH_C2[]->NXM_NX_NSH_C2[],
move:NXM_NX_TUN_ID[0..31]->NXM_NX_TUN_ID[0..31],
load:0x4->NXM_NX_TUN_GPE_NP[],IN_PORT

cookie=0xba5eba1100000101, table=10, priority=650,nsi=254,nsp=34
actions=move:NXM_NX_NSH_MDTYPE[]->NXM_NX_NSH_MDTYPE[],
move:NXM_NX_NSH_NP[]->NXM_NX_NSH_NP[],
move:NXM_NX_NSH_C1[]->NXM_NX_NSH_C1[],
move:NXM_NX_NSH_C2[]->NXM_NX_NSH_C2[],
move:NXM_NX_TUN_ID[0..31]->NXM_NX_TUN_ID[0..31],
load:0x4->NXM_NX_TUN_GPE_NP[],IN_PORT

cookie=0xba5eba1100000103, table=10, priority=650,nsi=253,nsp=8388642
actions=move:NXM_NX_NSH_MDTYPE[]->NXM_NX_NSH_MDTYPE[],
move:NXM_NX_NSH_NP[]->NXM_NX_NSH_NP[],
move:NXM_NX_NSI[]->NXM_NX_NSI[],
move:NXM_NX_NSP[0..23]->NXM_NX_NSP[0..23],
move:NXM_NX_NSH_C1[]->NXM_NX_TUN_IPV4_DST[],
move:NXM_NX_NSH_C2[]->NXM_NX_TUN_ID[0..31],
load:0x4->NXM_NX_TUN_GPE_NP[],IN_PORT

cookie=0xba5eba1100000101, table=10, priority=650,nsi=254,nsp=8388642
actions=move:NXM_NX_NSH_MDTYPE[]->NXM_NX_NSH_MDTYPE[],
move:NXM_NX_NSH_NP[]->NXM_NX_NSH_NP[],
move:NXM_NX_NSH_C1[]->NXM_NX_NSH_C1[],
move:NXM_NX_NSH_C2[]->NXM_NX_NSH_C2[],
move:NXM_NX_TUN_ID[0..31]->NXM_NX_TUN_ID[0..31],
load:0x4->NXM_NX_TUN_GPE_NP[],IN_PORT

cookie=0x14, table=10, priority=5 actions=drop

**Flows του SFF2:**

Πίνακας 0(Classifier)

Προώθηση πακέτων που RSP1 που προέρχονται από την SF2 και προωθούνται στον Classifier2:
cookie=0x0, table=0, priority=1000,nsi=253,nsp=34 actions=load:0x4->NXM_NX_TUN_GPE_NP[],load:0xc0a8013c->NXM_NX_TUN_IPV4_DST[],move:NXM_NX_NSP[0..23]->NXM_NX_NSP[0..23],move:NXM_NX_NSI[]->NXM_NX_NSI[],move:NXM_NX_NSH_C1[]->NXM_NX_NSH_C1[],move:NXM_NX_NSH_C2[]->NXM_NX_NSH_C2[],IN_PORT

Προώθηση όλων των πακέτων στον Πίνακα 1, μιας και η διαδικασία ταξινόμησης για το RSP1-Reverse γίνεται στον Classifier2:
cookie=0x14, table=0, priority=5 actions=goto_table:1

Πίνακας 1(Transport Ingress)

Προώθηση των πακέτων του RSP1 στον Πίνακα 4:
cookie=0x14, table=1, priority=250,nsp=34 actions=goto_table:4

Προώθηση των πακέτων του RSP1-Reverse στον Πίνακα 4:
cookie=0x14, table=1, priority=250,nsp=8388642 actions=goto_table:4

Διακοπή της κίνησης σε οποιαδήποτε άλλη περίπτωση:
cookie=0x14, table=1, priority=5 actions=drop

Πίνακας 2(Path Mapper)

cookie=0x14, table=2, priority=5 actions=goto_table:3

Πίνακας 3(Path Mapper ACL)

cookie=0x14, table=3, priority=5 actions=goto_table:4

Πίνακας 4(Next Hop)

Για πακέτα της RSP1, που πρέπει να προωθηθούν στην SF2:
cookie=0x14, table=4, priority=550,nsi=254,nsp=34 actions=load:0xc0a80128->NXM_NX_TUN_IPV4_DST[],goto_table:10

Για πακέτα της RSP1-Reverse, που πρέπει να προωθηθούν στην SF2:

cookie=0x14, table=4, priority=550,nsi=255,nsp=8388642 actions=load:0xc0a80128->NXM_NX_TUN_IPV4_DST[],goto_table:10

Για πακέτα της RSP1-Reverse, που πρέπει να προωθηθούν στον SFF2:
cookie=0x14, table=4, priority=550,nsi=254,nsp=8388642 actions=load:0xc0a80114->NXM_NX_TUN_IPV4_DST[],goto_table:10

Σε οποιαδήποτε άλλη περίπτωση:
cookie=0x14, table=4, priority=5 actions=goto_table:10

Πίνακας 10(Transport Egress)

cookie=0xba5eba1100000102, table=10, priority=660,nsi=253,nsp=34,nshc1=0 actions=IN_PORT

cookie=0xba5eba1100000103, table=10, priority=650,nsi=253,nsp=34
actions=move:NXM_NX_NSH_MDTYPE[]->NXM_NX_NSH_MDTYPE[],
move:NXM_NX_NSH_NP[]->NXM_NX_NSH_NP[],
move:NXM_NX_NSI[]->NXM_NX_NSI[],
move:NXM_NX_NSP[0..23]->NXM_NX_NSP[0..23],
move:NXM_NX_NSH_C1[]->NXM_NX_TUN_IPV4_DST[],
move:NXM_NX_NSH_C2[]->NXM_NX_TUN_ID[0..31],
load:0x4->NXM_NX_TUN_GPE_NP[],IN_PORT

cookie=0xba5eba1100000101, table=10, priority=650,nsi=254,nsp=34
actions=move:NXM_NX_NSH_MDTYPE[]->NXM_NX_NSH_MDTYPE[],
move:NXM_NX_NSH_NP[]->NXM_NX_NSH_NP[],
move:NXM_NX_NSH_C1[]->NXM_NX_NSH_C1[],
move:NXM_NX_NSH_C2[]->NXM_NX_NSH_C2[],
move:NXM_NX_TUN_ID[0..31]->NXM_NX_TUN_ID[0..31],
load:0x4->NXM_NX_TUN_GPE_NP[],IN_PORT

cookie=0xba5eba1100000101, table=10, priority=650,nsi=255,nsp=8388642
actions=move:NXM_NX_NSH_MDTYPE[]->NXM_NX_NSH_MDTYPE[],
move:NXM_NX_NSH_NP[]->NXM_NX_NSH_NP[],
move:NXM_NX_NSH_C1[]->NXM_NX_NSH_C1[],
move:NXM_NX_NSH_C2[]->NXM_NX_NSH_C2[],
move:NXM_NX_TUN_ID[0..31]->NXM_NX_TUN_ID[0..31],
load:0x4->NXM_NX_TUN_GPE_NP[],IN_PORT

cookie=0xba5eba1100000101, table=10, priority=650,nsi=254,nsp=8388642
actions=move:NXM_NX_NSH_MDTYPE[]->NXM_NX_NSH_MDTYPE[],
move:NXM_NX_NSH_NP[]->NXM_NX_NSH_NP[],
move:NXM_NX_NSH_C1[]->NXM_NX_NSH_C1[],

move:NXM_NX_NSH_C2[]->NXM_NX_NSH_C2[],
move:NXM_NX_TUN_ID[0..31]->NXM_NX_TUN_ID[0..31],
load:0x4->NXM_NX_TUN_GPE_NP[],IN_PORT

cookie=0x14, table=10, priority=5 actions=drop

**Flows του Classifier2:**

Ενθυλάκωση NSH των πακέτων που πληρούν τις προδιαγραφές της ACL2 για το RSP1-Reverse και προώθηση τους στην θύρα 7(στον SFF2):
cookie=0x0, table=0,
priority=1000,tcp,in_port=1,nw_src=192.168.2.0/24,nw_dst=192.168.2.0/24,tp_src=80
actions=push_nsh,load:0x1->NXM_NX_NSH_MDTYPE[],load:0x3-
>NXM_NX_NSH_NP[],load:0x800022->NXM_NX_NSP[0..23],load:0xff->NXM_NX_NSI[],load:0x1-
>NXM_NX_NSH_C1[],load:0x2->NXM_NX_NSH_C2[],load:0x3->NXM_NX_NSH_C3[],load:0x4-
>NXM_NX_NSH_C4[],load:0x4->NXM_NX_TUN_GPE_NP[],load:0xc0a80132-
>NXM_NX_TUN_IPV4_DST[],output:7

Τερματισμός της ενθυλάκωσης NSH για πακέτα που έρχονται από τον SFF2 και χρησιμοποίησαν τον RSP1 και προώθηση τους στην θύρα 1(client):

cookie=0x0, table=0, priority=1000,nsi=253,nsp=34 actions=pop_nsh,output:1

**Βήμα 10 Ανάλυση Πακέτων NSH με VXLAN GPE**

Για να δούμε την λειτουργία των πακέτων με ενθυλάκωση VXLAN, με το εργαλείο tcpdump αναλύουμε τα πακέτα που διέλευσαν από τον SFF2. Η ενθυλάκωση NSH με VXLAN GPE έχει την εξής μορφή:


**Εικόνα 7.25 Κεφαλίδα VXLAN-GPE με NSH**

Με βάση αυτό και την ανάλυση πακέτου που προωθείται από τον SFF2 στην SF2 προκύπτουν:

Ethernet κεφαλίδα: 00 00 08 00 27 20 7a e3 08 00 27 aa 79 b6 08 00

IP κεφαλίδα: 45 00 00 94 ca 58 40 00 40 11 ec 55 c0 a8 01 32 c0 a8 01 28

UDP VXLAN κεφαλίδα: c0 89 19 e9 00 80 9c 25

VXLAN κεφαλίδα: 08 00 00 00 00 00 00 00

(00 00 00): VNI=0x000000

Ενθυλακωμένη κεφαλίδα Ethernet: 00 00 22 22 22 22 00 00 11 11 11 11 89 4f
0x894f: Τύπος Ethernet για NSH

Κεφαλίδα NSH Base: 00 06 06 03

Κεφαλίδα Service Path: 00 00 22 fe

NSP: 34 (0x000022)

NSI: 254 (0xfe)

Mandatory Context-Header 1: 00 00 00 01

Mandatory Context-Header 2: 00 00 00 02

Mandatory Context-Header 3: 00 00 00 03

Mandatory Context-Header 4: 00 00 00 04

```
> Frame 41: 162 bytes on wire (1296 bits), 162 bytes captured (1296 bits)
> Ethernet II, Src: PcsCompu_aa:79:b6 (08:00:27:aa:79:b6), Dst: PcsCompu_20:7a:e3 (08:00:27:20:7a:e3)
> Internet Protocol Version 4, Src: 192.168.1.50, Dst: 192.168.1.40
> User Datagram Protocol, Src Port: 49289, Dst Port: 6633
> Data (120 bytes)
```

```
0000  08 00 27 20 7a e3 08 00  27 aa 79 b6 08 00 45 00   ··' z··· '·y···E·
0010  00 94 ca 58 40 00 40 11  ec 55 c0 a8 01 32 c0 a8   ···X@·@· ·U···2··
0020  01 28 c0 89 19 e9 00 80  9c 25 08 00 00 00 00 00   ·(······ ·%······
0030  00 00 00 00 22 22 22 22  00 00 11 11 11 11 89 4f   ····"""" ·······O
0040  00 06 01 03 00 00 22 fe  00 00 00 01 00 00 00 02   ·····"· ········
0050  00 00 00 03 00 00 00 04  00 00 22 22 22 22 00 00   ········ ··""""··
0060  11 11 11 11 08 00 45 00  00 3c 24 17 40 00 40 06   ·····E· ·<$·@·@·
0070  91 51 c0 a8 02 01 c0 a8  02 02 d9 ef 00 50 d6 2e   ·Q····· ·····P··
0080  6f 1f 00 00 00 00 a0 02  6a 40 df 15 00 00 02 04   o····· j@·····
0090  05 50 04 02 08 0a 0f 24  4b 08 00 00 00 00 01 03   ·P·····$ K·····
00a0  03 07                                             ··
```

**Εικόνα 7.26 Πακέτο από SFF2 σε SF2**

# Κεφάλαιο 8ο

# Μελλοντικές επεκτάσεις

Στην παρούσα εργασία ασχοληθήκαμε με την υλοποίηση αλυσίδων εξυπηρέτησης, είτε χρησιμοποιώντας VXLAN overlays είτε με τη χρήση ODL ελεγκτή για την παραμετροποίηση και δημιουργία αλυσίδων SFC. Σε αυτό το κεφάλαιο θα παρουσιάσουμε ορισμένες από τις μελλοντικές χρήσεις που μπορεί να έχει η τεχνολογία SFC αλλά και κάποιες τεχνικές προτάσεις για βελτίωση της υλοποίησης μας.

## 8.1) Επιτάχυνση δικτυακών λειτουργιών σε data-centers με χρήση SFC [45]

Τα κέντρα δεδομένων αναπτύσσουν κόμβους υπηρεσιών (service nodes) σε διάφορα σημεία της τοπολογίας του δικτύου. Αυτοί οι κόμβοι παρέχουν μια σειρά λειτουργιών εξυπηρέτησης και το σύνολο λειτουργιών εξυπηρέτησης που φιλοξενούνται σε έναν δεδομένο κόμβο υπηρεσίας μπορεί να επικαλύπτεται με λειτουργίες υπηρεσιών που φιλοξενούνται σε άλλους κόμβους υπηρεσιών.

Συχνά, οι τοπολογίες των data-centers ακολουθούν έναν ιεραρχικό σχεδιασμό με πυρήνα, συγκέντρωση (aggregation), πρόσβαση και εικονική πρόσβαση σε συσκευές δικτύου. Σε τέτοιες τοπολογίες, οι κόμβοι υπηρεσίας αναπτύσσονται είτε στα στρώματα aggregation ή πρόσβασης.

Σε δίκτυα μεγάλης κλίμακας, όπως τα δίκτυα μεταφοράς (carrier networks), υπάρχουν πολλά data-centers που διανέμονται σε μεγάλες γεωγραφικές περιοχές. Κάθε data-center αναπτύσσει κόμβους υπηρεσιών και λειτουργίες υπηρεσιών ποικίλων τύπων σε μια σειρά υλικών.

Ο πρωταρχικός σκοπός της ανάπτυξης λειτουργιών εξυπηρέτησης σε διάφορα σημεία του δικτύου είναι η εφαρμογή λειτουργιών εξυπηρέτησης σε διαφορετικούς τύπους κίνησης:

i) Κυκλοφορία που προέρχεται από φυσικό ή εικονικό φόρτο εργασίας (workload) στο data-center και προορίζεται για φυσικό ή εικονικό φόρτο εργασίας στο data-center.

ii) Κυκλοφορία που προέρχεται από τοποθεσία που είναι απομακρυσμένη από το data-center και προορίζεται για φυσικό ή εικονικό φόρτο εργασίας στο data-center. Για παράδειγμα, κυκλοφορία που προέρχεται από υποκατάστημα ή περιφερειακό γραφείο και προορίζεται για ένα από τα πρωτεύοντα data-centers μιας επιχείρησης ή κυκλοφορία που προέρχεται από έναν από τους μισθωτές ενός παρόχου υπηρεσιών που προορίζεται για τις εν λόγω αιτήσεις ενοικιαστών στο data-center του παρόχου υπηρεσιών. Ακόμη μια άλλη παραλλαγή αυτού του τύπου κυκλοφορίας περιλαμβάνει τρίτους προμηθευτές και συνεργάτες του χειριστή του data-center που έχουν πρόσβαση από απόσταση στις εφαρμογές τους στο data-center μέσω ασφαλών συνδέσεων.

iii) Κυκλοφορία που προέρχεται από μια τοποθεσία που είναι απομακρυσμένη από το data-center και προορίζεται για μια τοποθεσία απομακρυσμένη από το data-center, αλλά που διέρχεται μέσω

του data-center. για παράδειγμα, κίνηση που προέρχεται από κινητή συσκευή προοριζόμενη για διακομιστές στο Διαδίκτυο που δρομολογούνται μέσω του data-center προκειμένου να εξυπηρετηθούν.

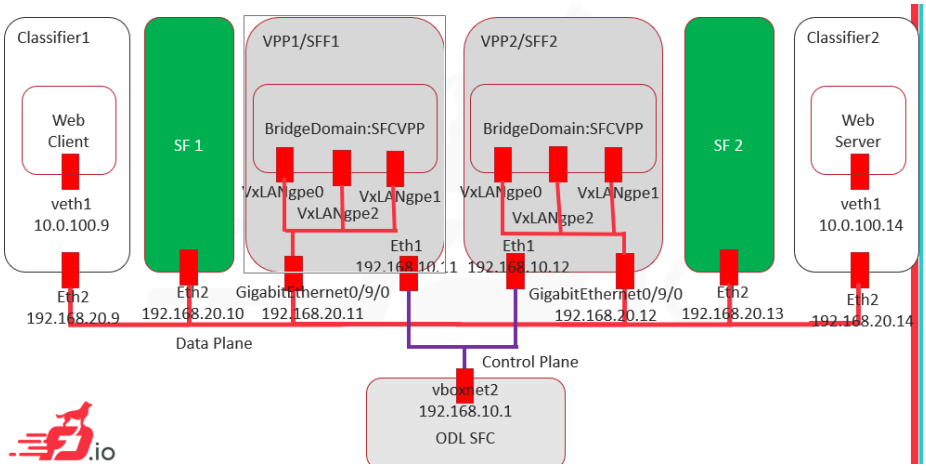
Η συντήρηση της κυκλοφορίας περιλαμβάνει την κατεύθυνση της κίνησης μέσω μιας σειράς λειτουργιών εξυπηρέτησης που μπορεί να βρίσκονται σε διαφορετικά σημεία του δικτύου ή μέσα σε μία μόνο συσκευή συνδεδεμένη στο δίκτυο ή σε οποιοδήποτε συνδυασμό μεταξύ τους. Η παράδοση πολλών λειτουργιών εξυπηρέτησης σε μια ακολουθία, σε ένα data-center ή μεταξύ πολλών data-center, δημιουργεί έτσι πολλές απαιτήσεις σχετικά με τη συνολική αρχιτεκτονική παροχής υπηρεσιών. Αυτές οι αρχιτεκτονικές μπορούν να ονομάζονται αρχιτεκτονικές αλυσίδων λειτουργίας υπηρεσίας ενώ ο κατάλογος των λειτουργιών εξυπηρέτησης που εφαρμόζονται στην κίνηση είναι μια αλυσίδα λειτουργίας υπηρεσίας(SFC), και με τη χρήση του SDN και παρομοίων αρχιτεκτονικών με αυτές που αναλύσαμε πιο πάνω μπορούν να αυξήσουν την απόδοση.

### 8.2) Χρησιμοποίηση της τεχνολογίας Vector Packets Processing (VPP) ως SFF

Το fd.io VPP εκτελείται στο χώρο του χρήστη σε πολλαπλές αρχιτεκτονικές, συμπεριλαμβανομένων των αρχιτεκτονικών x86, ARM και Power σε διακομιστές x86 και σε ενσωματωμένες συσκευές. Ο σχεδιασμός του VPP είναι agnostic ως προς το hadware και τον πυρήνα.

Το VPP πιέζει τα ακραία όρια απόδοσης και κλίμακας. Ανεξάρτητοι έλεγχοι δείχνουν ότι, σε κλίμακα, το FD.io που τροφοδοτείται με VPP είναι δύο τάξεις μεγέθους ταχύτερα από τις διαθέσιμες σήμερα τεχνολογίες. Το VPP διαβάζει το μεγαλύτερο διαθέσιμο διάνυσμα πακέτων από το στρώμα δικτύου I/O.[46]

Στην περίπτωση του SFC μπορούμε να χρησιμοποιήσουμε ως SFF ένα VPP, γεγονός που μπορεί να αυξήσει την απόδοση της αλυσίδας. Το OpenDaylight σε συνδυασμό με το εργαλείο Honeycomb παρέχει ένα API που μέσω της NETCONF/YANG δίνει την δυνατότητα επικοινωνίας του VPP με τον ελεγκτή.

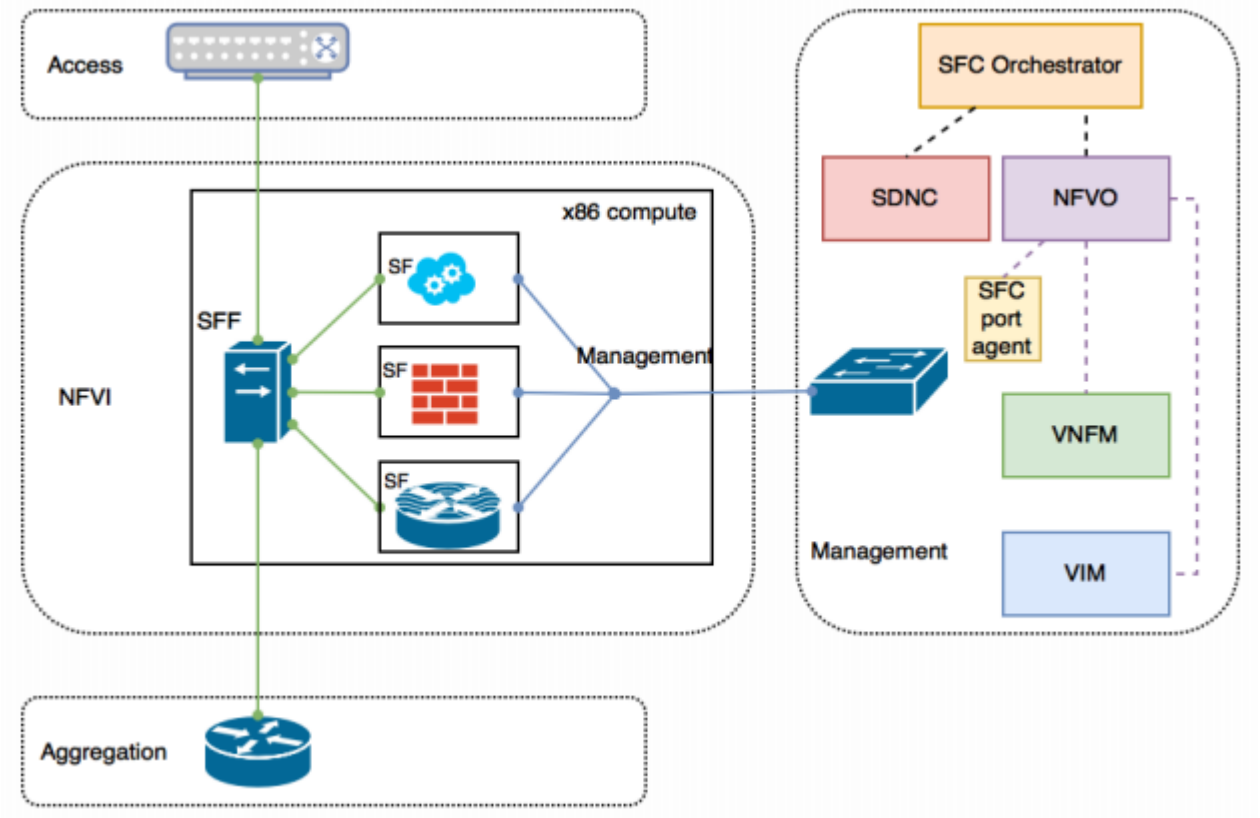



**Εικόνα 8.1 Υλοποίηση SFC με χρήση fd.io VPP [47]**

### 8.3) Χρήση της τεχνολογίας των containers για τις SF

Για την αύξηση της απόδοσης αλλά και για την δημιουργία SF και SFF με πιο lightweight τρόπο μπορούμε να χρησιμοποιήσουμε containers για την υλοποίηση τους, και συγκεκριμένα το Docker. Ένα container είναι μια τυποποιημένη μονάδα λογισμικού που πακετάρει τον κώδικα και όλες τις εξαρτήσεις του, έτσι ώστε η εφαρμογή να εκτελείται γρήγορα και αξιόπιστα από ένα περιβάλλον υπολογιστών σε άλλο. Μια εικόνα Docker container είναι ένα ελαφρύ, αυτόνομο, εκτελέσιμο πακέτο λογισμικού που περιλαμβάνει όλα όσα χρειάζονται για την εκτέλεση μιας εφαρμογής: κώδικας, χρόνου εκτέλεσης, εργαλεία συστήματος, βιβλιοθήκες συστήματος και ρυθμίσεις.

Η πλατφόρμα Opendaylight μας δίνει την δυνατότητα χρησιμοποίησης του Docker για τα στοιχεία της SFC που αναφέρθηκαν πριν.




**Εικόνα 8.2 Παράδειγμα αρχιτεκτονικής SFC με χρήση VPP ως SFF, και εκτέλεση των SF σε Containers[48]**

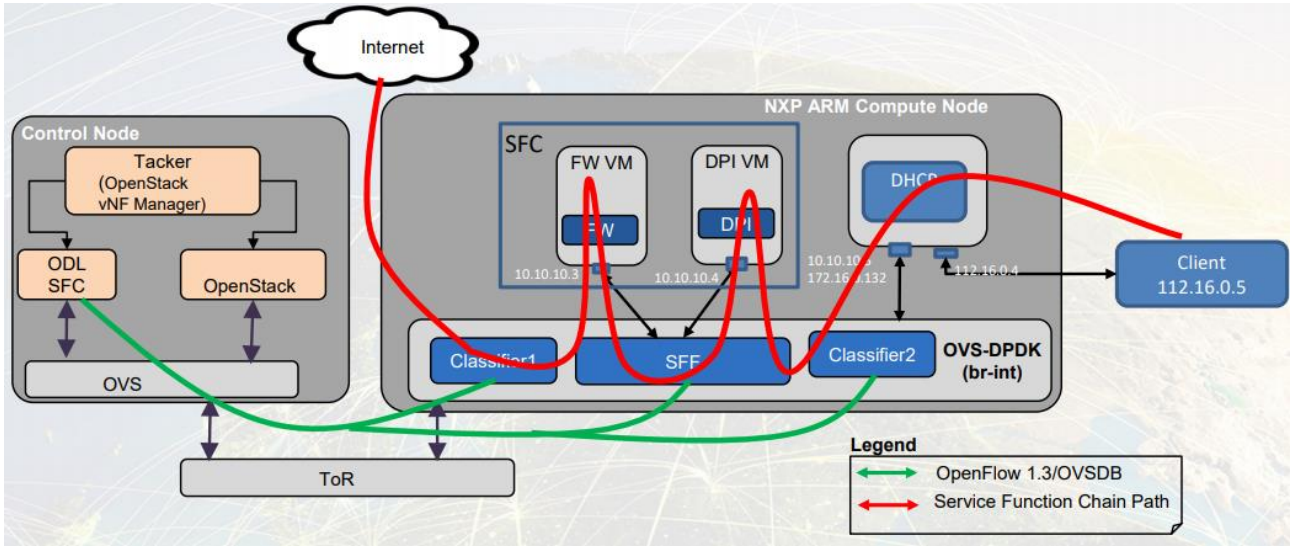
### 8.4) Χρήση πλατφόρμας Openstack και υπηρεσίας Tacker στην υλοποίηση του SFC

Το Openstack είναι μια πλατφόρμα για την δημιουργία και υποστήριξη υπηρεσιών υπολογιστικού νέφους. Μεταξύ άλλων, παρέχει την υπηρεσία Neutron για να παρέχει ολοκληρωμένες υπηρεσίες δικτύου.

Το project OpenStack SFC (networking-sfc), δημιουργεί APIs για την αλυσίδα λειτουργιών εξυπηρέτησης (SFC) που μπορούν να υλοποιηθούν από οποιονδήποτε πάροχο υπηρεσιών

δικτύων υποστήριξης για να υποστηρίξουν το SFC. Αυτά τα API παρέχουν έναν τρόπο ενσωμάτωσης του OpenStack SFC σε οποιονδήποτε από τους υποστηρικτές του SFC. Το OpenDaylight SFC παρέχει υλοποίηση του SFC, αλλά προς το παρόν δεν υπάρχει επίσημος μηχανισμός ολοκλήρωσης για την χρήση του OpenDaylight ως φορέα SFC για το networking-sfc.[49]

Με το πρόγραμμα Tacker του Openstack, ανοίγονται πολλές δυνατότητες για την πραγματοποίηση των περιπτώσεων χρήσης NFV (π.χ. SFC) χρησιμοποιώντας το OpenStack ως πλατφόρμα. Η παροχή μιας επίσημης σύνδεσης μεταξύ του OpenStack και του OpenDaylight για τη χρήση του SFC θα βοηθήσει τους χρήστες NFV να εκμεταλλευτούν το συνδυασμό OpenStack, Tacker και OpenDaylight ως λύση. Μέχρι στιγμής, η Tacker επικοινωνεί απευθείας με τους παρόχους OpenDaylight SFC και ταξινομητές και όχι μέσω API OpenStack SFC (networking-sfc), παρόλα αυτά θα βοηθούσε στην επέκταση της παρούσας εργασίας ως προς την διαχείριση των SF.




**Εικόνα 8.3 Παράδειγμα SFC με Tacker[50]**

# Κεφάλαιο 9ο

## Συμπεράσματα

Στην παρούσα διπλωματική εργασία εξετάζεται η δημιουργία αλυσίδων υπηρεσιών. Στην πρώτη υλοποίηση πραγματοποιήθηκε μια αλυσίδα υπηρεσιών με τεχνικές δικτύων επικάλυψης (overlay networks) για την σύνδεση των VM που εκτελούνται οι υπηρεσίες και εικονικούς μεταγωγείς. Στην δεύτερη υλοποίηση ακολουθήθηκε η αρχιτεκτονική SFC, δημιουργήθηκε μια αλυσίδα υπηρεσίας ελεγχόμενη από έναν ελεγκτή SDN, όπου παραμετροποίησε τις ρυθμίσεις της. Η ανάπτυξη των υπηρεσιών έγινε σε ένα πλαίσιο εικονικοποίησης δικτυακών λειτουργιών (Network Function Virtualization) και με την αξιοποίηση της τεχνολογίας SDN.

Με τη χρήση του OpenFlow δόθηκε η δυνατότητα να δρομολογηθεί η κίνηση στις αλυσίδες αντιμετωπίζοντας τις προκλήσεις που παρουσιάζονται σε τέτοιου είδους περιβάλλοντα, και χωρίς πολύπλοκες παραμετροποιήσεις να αυξήσουμε την ασφάλειά του δικτύου. Αποδείχθηκε παράλληλα πόσο σημαντικό εργαλείο είναι οι μεταγωγείς Openflow και πως μπορούν σαν προέκταση να χρησιμοποιηθούν και σε δίκτυα μεγάλης κλίμακας.

Με την χρήση του VXLAN overlay, παρουσιάστηκε η δυνατότητα κλιμάκωσης και σταθερότητας του δικτύου. Η μεγαλύτερη δυνατότητα κλιμάκωσης σε σχέση με τα δίκτυα VLAN που χρησιμοποιούν ένα αναγνωριστικό VLAN 12 bit δίνοντας τη δυνατότητα επεκτασιμότητας μόνο έως 4094 VLAN, στο VXLAN με την χρήση αναγνωριστικού 24 bit, επιτρέπεται η ταυτόχρονη συνύπαρξη έως 16 εκατομμυρίων τμημάτων VXLAN στην ίδια κοινή υποδομή δικτύου.

Με την χρήση της αρχιτεκτονικής SFC, σε συνδυασμό με την παραμετροποίηση από τον ελεγκτή SDN, παρατηρήθηκε μια τεράστια δυνατότητα για τη δημιουργία αλυσιδωτών λειτουργιών με εύκολη παραμετροποίηση μέσω του ελεγκτή, αλλά και οι δυνατότητες που δίνει η ενθυλάκωση NSH και η χρήση του VXLAN-GPE για μια ριζοσπαστική αλλαγή στην τεχνολογία των data-centers.

Τα SDN δίκτυα σε συνδυασμό με την NFV αποτελούν το μέλλον της δικτύωσης, ανοίγοντας τεράστιες δυνατότητες για τους παρόχους αλλά και τα data-centers να αυξήσουν την απόδοση των υπηρεσιών αλλά και την ταχύτητα περάτωσής τους.

## Βιβλιογραφία


[1] https://en.wikipedia.org/wiki/Virtualization

[2] https://searchservervirtualization.techtarget.com/definition/virtualization

[3] https://www.sdxcentral.com/nfv/definitions/nfv-mano/

[4] Network Function Virtualization: State-of-the-art and Research Challenges Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, Raouf Boutaba (https://arxiv.org/pdf/1509.07675.pdf)

[5] https://www.electronics-notes.com/articles/connectivity/nfv-network-functions-virtualisation/virtualized-network-functions-vnf.php

[6] https://el.wikipedia.org/wiki/Εικονικοποίηση_δικτυακών_λειτουργιών

[7] https://www.sdxcentral.com/nfv/definitions/nfv-mano/

[8] https://www.sdxcentral.com/nfv/definitions/nfv-elements-overview/

[9] Software-Defined Networking: A Comprehensive Survey Diego Kreutz, Fernando M. V. Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig (https://arxiv.org/pdf/1406.0440.pdf)

[10] https://en.wikipedia.org/wiki/OpenFlow

[11] http://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf

[12] https://www.cisco.com/c/en/us/solutions/software-defined-networking/sdn-vs-nfv.html

[13] Juniper Networks, Learn About VXLAN in Virtualized Data Center Networks

[14] Cisco White Paper: Data Center Overlay Technologies https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-730116.pdf

[15] Configuring-site-to-site-gre-tunnel(http://tcpflag.blogspot.com)

[16] Cisco Overlay Transport Virtualization for Geographically Dispersed Virtual Data Centers - Improve Application Availability and Portability Solution Overview

[17] https://en.wikipedia.org/wiki/Locator/Identifier_Separation_Protocol

[18] https://tools.ietf.org/html/rfc6830

[19] https://www.researchgate.net/figure/Diagram-of-the-LISP-protocol_fig8_273698755

[20] https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-733127.pdf

[21] https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cether/configuration/xe-16-8/ce-xe-16-8-book/vxlan-gpe-tunnel.pdf

[22] ONF OpenFlow Switch Specification Version 1.4.1 ( Protocol version 0x05)

[23] http://www.openvswitch.org/

[24] Hybrid IP/SDN Networking: Open Implementation and Experiment Management Tools May 2015, IEEE Transactions on Network and Service Management 13(1):1-1,

[25] CREATION OF A VIRTUAL OVERLAY NETWORK WITH SDN AND VXLAN, Fernado Lama (https://upcommons.upc.edu/bitstream/handle/2117/109765/TFG_Fernando_Lama.pdf)

[26] L4-L7 Service Function Chaining Solution Architecture Version 1.0 ONF TS-027

[27] http://www.openvswitch.org/support/ovscon2015/16/1040-elzur.pdf

[28] https://www.sdxcentral.com/sdn/network-virtualization/definitions/what-is-network-service-chaining/

[29] IETF, Service Function Chaining (SFC) Architecture, RFC 7665

[30] Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges, Ahmed M. Medhat, Tarik Taleb, Asma Elmangoush, Giuseppe A. Carella, Stefan Covaci, Thomas Magedanz

[31] https://opensource.com/resources/raspberry-pi

[32] https://computer.howstuffworks.com/raspberry-pi2.htm

[33] https://www.raspberrypi.org/documentation/faqs/

[34] https://vyos.io/

[35] Facilitation of The OpenDaylight Architecture, Conference Paper · May 2017, Ahmad Hemid

[36] www.opendaylight.org

[37] https://cordis.europa.eu/docs/projects/cnect/2/619572/080/deliverables/001-619572CONSIGND32renditionDownload.pdf

[38] https://en.wikipedia.org/wiki/Representational_state_transfer

[39] https://engineering.linkedin.com/architecture/restli-restful-service-architecture-scale

[40] https://docs.opendaylight.org/en/stable-boron/user-guide/service-function-chaining.html#

[41] https://docs.openvswitch.com

[42] https://www.vagrantup.com/intro/index.html

[43] https://coderwall.com/p/uf_44a/quick-ip-netns

[44] https://pypi.org/project/sfc/

[45] https://tools.ietf.org/html/draft-ietf-sfc-dc-use-cases-06

[46] https://fd.io/technology/

[47] ODL SFC and VPP Integration Yi Yang,Intel, Keith Burns, CISCO, Hongjun Ni, Intel

[48] http://events17.linuxfoundation.org/sites/events/files/slides/ContainerServiceChaining_MartinSunal.pdf

[49] https://docs.openstack.org/networking-odl/ocata/specs/sfc-driver.html

[50] https://dpdksummit.com/Archive/pdf/2017India/DPDK-India2017-Gorja-SFC.pdf

# Παράρτημα

**Παραμετροποίηση για την δημιουργία GRE Tunnel μεταξύ Raspberry Pi και VyOS Router**

**Στο terminal του VyOS Router:**

```
//είσοδος στο configure terminal
configure

//δημιουργία χώρου για το GRE Tunnels και δημιουργία της πολιτικής change-mss
set policy route CHANGE-MSS rule 1 set tcp-mss 1360
set policy route change-mss rule 1 protocol tcp
set policy route change-mss rule 1 tcp flags SYN

//δημιουργία GRE Tunnel
set interfaces tunnel tun0 encapsulation gre
set interfaces tunnel tun0 address 10.0.0.1/30
set interfaces tunnel tun0 mtu 1400
set interfaces tunnel tun0 policy route CHANGE-MSS
set interfaces tunnel tun0 local-ip 147.102.7.79
set interfaces tunnel tun0 remote-ip 147.102.40.69

//επιβεβαίωση και αποθήκευση της παραμετροποίησης
commit
save

//έλεγχος δημιουργίας tunnel
show interfaces tunnel
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface       IP Address                S/L  Description
---------       ----------                ---  -----------
tun0            10.0.0.1/30               u/u
```

Στο Rpi ορίζουμε αντίστοιχα την διεπαφή tun0 με τις παρακάτω ρυθμίσεις:

- Τύπος ενθυλάκωσης GRE
- Τοπική διεύθυνση δημόσιου δικτύου την IP του Rpi(147.102.40.69)
- Απομακρυσμένη διεύθυνση δημόσιου δικτύου την IP του VyOS(147.102.7.79)
- Διεύθυνση IP της διεπαφής ως 10.0.0.2/30
- Σύζευξη της διεπαφής με την 10.0.0.1/30

**Στο terminal του Raspberry:**

```
//είσοδος σε κατάσταση superuser
sudo -i

//δημιουργία διόδου GRE
ip tunnel add tun0 mode gre local 147.102.40.69 remote 147.102.7.79
ip link set dev tun0 up
ip addr add 10.0.0.2 dev tun0 peer 10.0.0.1/30
```

**Παραμετροποίηση των VM1 και VM2 για δυνατότηρα εισόδου στο terminal τους μέσω του εργαλείου virsh**

Για να μπορέσουμε να χρησιμοποιήσουμε την είσοδο στο terminal των VM μέσω του εργαλείου virsh χρειάζεται να αλλάξουμε τις ρυθμίσεις στο αρχείο /etc/default/grub στα VMs στα οποία συνδεόμαστε με SSH αφού βρήκαμε μέσω του εργαλείου nmap την εικονική διεύθυνση τους που τους δόθηκε από την σύνδεση τους στο default linux bridge του Libvirt. Οι αλλαγές που κάνουμε στο αρχείο αυτό είναι η προσθήκη των παρακάτω γραμμών:

```
GRUB_CMDLINE_LINUX='console=tty0 console=ttyS0,19200n8'
GRUB_TERMINAL=serial
GRUB_SERIAL_COMMAND="serial --speed=19200 --unit=0 --word=8 --parity=no --stop=1"
```

Μετά εκτελώντας την εντολή update-grub σε κατάσταση superuser και επανεκκινώντας τα μηχανήματα μπορούμε να χρησιμοποιήσουμε το εργαλείο virsh και με την εντολή virsh console να εισερχόμαστε στο terminal κάθε VM.

**Εγκατάσταση και ρύθμιση μεταγωγέων OVS στους HOST**

Στον HOST A:

```
//δημιουργία μεταγωγέα br1
sudo ovs-vsctl add-br1

//σύνδεση μεταγωγέα με την διεπαφή ens192
sudo ovs-vsctl add-port br1 ens192

//αφαίρεση της IP από το ens192
sudo ifconfig ens192 0

//απόδοση της IP του ens192 στην br1 δίνοντας και τις αντίστοιχες παραμέτρους του υποδικτύου
sudo ifconfig br1 147.102.39.28 netmask 255.255.255.0 broadcast 147.102.39.63
```

Αντίστοιχα στον HOST B

```
sudo ovs-vsctl add-br br2
sudo ovs-vsctl add-port br2 ens192
sudo ifconfig ens192 0
sudo ifconfig br1 147.102.39.29 netmask 255.255.255.0 broadcast 147.102.39.63
```

**Σύνδεση των VM με τα OVS**

Το default δίκτυο και στους δύο HOST είναι της μορφής:

**bill@hostA:~$ sudo virsh net-edit default**

```
<network>
  <name>default</name>
  <uuid>dd36a258-3e63-40f8-9daa-78a5cc2aeb29</uuid>
  <forward mode='nat'/>
  <bridge name='virbr0' stp='on' delay='0'/>
  <mac address='52:54:00:83:be:a7'/>
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254'/>
    </dhcp>
  </ip>
</network>
```

Για να γίνει η σύνδεση στα OVS χρειαζόμαστε την δημιουργία του δικτύου ovs-net:

//δημιουργία αρχείου xml για το δίκτυο
**bill@hostA:~$ sudo nano ovs-net.xml**

```
<network>
  <name>ovsnet</name>
  <forward mode='bridge'/>
  <bridge name='br1'/> // όπου br1 βάζουμε br2 για τον HOSTB
  <virtualport type='openvswitch'/>
</network>
```

//ορισμός δικτύου ovsnet
**bill@hostA:~$ sudo virsh net-define ovsnet**

//εκκίνηση δικτύου ovsnet
**bill@hostA:~$ sudo virsh net-start ovsnet**

//εμφάνιση διαθέσιμων δικτύων
**bill@hostA:~$ sudo virsh net-list**

```
 Name             State     Autostart     Persistent
----------------------------------------------------------
 default          active    yes           yes
 ovsnet           active    yes           yes
```

Με την δημιουργία των VM δημιουργείται μια εικονική διεπαφή η vnet0 σε κάθε HOST η οποία είναι συνδεδεμένη στο virbr0. Για να συνδέσουμε το VM στο δίκτυο ovsnet πρέπει πρώτα να συνδέσουμε κάθε διεπαφή στα OVS με τις παρακάτω εντολές

- sudo ovs-vsctl add-port br1 vnet0(στον HOST A)
- sudo ovs-vsctl add-port br2 vnet0(στον HOST B)

Στην συνέχεια, τελευταία κίνηση είναι να ρυθμίσουμε τα xml αρχεία των VM1, VM2 για να τα συνδέσουμε με το ovsnet, και μετά να επανεκκινήσουμε τα VMs.

Το αρχείο xml του VM1 για σύνδεση στο default δίκτυο είναι:

**bill@hostA:~$ sudo virsh edit VM1**

```
[...]
   <interface type='bridge'>
     <mac address='52:54:00:1b:28:53'/> //η αντίστοιχη διεύθυνση MAC για το VM1
     <source bridge='virbr0'/>
     <model type='virtio'/>
     <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
   </interface>
[…]
```

Για την σύνδεση στο ovsnet κάνουμε τις παρακάτω αλλαγές:

**bill@hostA:~$ sudo virsh edit VM1**

```
[...]
   <interface type='bridge'>
     <mac address='52:54:00:1b:28:53'/> //Για το VM2 η αντίστοιχη MAC
     <source bridge='br1'/> //Για το VM2 το br2
     <virtualport type='openvswitch'/>
     <model type='virtio'/>
     <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
   </interface>
[…]
```

Μετά την επανεκκίνηση των δύο VM, αυτά είναι συνδεδεμένα με τα αντίστοιχα OVS.

**Vagrantfile**

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

  config.vm.box = "trusty-server-cloudimg-amd64-vagrant-disk1.box"
  config.vm.box_url = "https://cloud-images.ubuntu.com/vagrant/trusty/current/trusty-server-
cloudimg-amd64-vagrant-disk1.box"
  config.vm.provider :virtualbox do |v|
    v.customize ["modifyvm", :id, "--memory", 1024]
    v.customize ["modifyvm", :id, "--cpus", 1]
  end

  config.vm.synced_folder "../..", "/sfc"

  config.vm.define "odl" do | h |
    h.vm.host_name = "odl"
    h.vm.network :private_network, ip: "192.168.1.5"
    h.vm.provider :virtualbox do |v|
      v.customize ["modifyvm", :id, "--memory", 4096]
      v.customize ["modifyvm", :id, "--cpus", 2]
    end
  end

  config.vm.define "classifier1" do | h |
    h.vm.host_name = "classifier1"
    h.vm.network :private_network, ip: "192.168.1.10"
  end

  config.vm.define "sff1" do | h |
    h.vm.host_name = "sff1"
    h.vm.network :private_network, ip: "192.168.1.20"
  end

  config.vm.define "sf1" do | h |
    h.vm.host_name = "sf1"
    h.vm.network :private_network, ip: "192.168.1.30"
  end

  config.vm.define "sf2" do | h |
    h.vm.host_name = "sf2"
    h.vm.network :private_network, ip: "192.168.1.40"
  end
```

```
  config.vm.define "sff2" do | h |
    h.vm.host_name = "sff2"
    h.vm.network :private_network, ip: "192.168.1.50"
  end

  config.vm.define "classifier2" do | h |
    h.vm.host_name = "classifier2"
    h.vm.network :private_network, ip: "192.168.1.60"
  end
end
```

**Εγκατάσταση Java για το ODL**

```
//εντολές στο terminal
sudo apt-get install default-jre-headless
sudo apt-get install maven
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java9-installer
sudo /usr/sbin/update-alternatives --config java
sudo apt-get install openjdk-8-jdk
sudo apt-get install openjdk-8-jre
sudo apt-get install oracle-java8-set-default
sudo apt-get install oracle-java8-installer
sudo nano ~/.bashrc

//προσθήκη στο ~/.bashrc
export JAVA_HOME=/usr/lib/jvm/java-8-oracle

//εντολές στο terminal
source ~/.bashrc
unzip distribution-karaf-0.5.3-Boron-SR3.zip

//έναρξη ODL με την εντολή
distribution-karaf-0.5.3-Boron-SR3/bin/karaf
```

**Ρυθμίσεις στα VM με τους SFF**

**//εγκατάσταση OVS με το patch για το NSH**
sudo -i
curl https://raw.githubusercontent.com/yyang13/ovs_nsh_patches/master/start-ovs-deb.sh | bash
chmod +x ovs_nsh_patches/start-ovs-deb.sh
./ovs_nsh_patches/start-ovs-deb.sh

**//ρύθμιση του διαχειριστή όλων των OVS**
ovs-vsctl set-manager tcp:192.168.1.5:6640

**//δημιουργία μεταγωγέα br-sfc**
ovs-vsctl add-br br-sfc

**//ρύθμιση του ελεγκτή του br-sfc**
ovs-vsctl set-controller br-sfc tcp:192.168.1.5:6653

**//ρύθμιση της έκδοσης του Openflow που θα χρησιμοποιηθεί**
ovs-vsctl set bridge br-sfc protocols=OpenFlow13

**Δημιουργία client και server με netns**

Και στα δύο VM θα κάνουμε από το terminal τους της βασικές κοινές ρυθμίσεις για την δημιουργία των namespaces:

```
//είσοδος σε κατάσταση superuser
sudo -i

//δημιουργία namespace app
ip netns add app

//δημιουργία εικονικής διεπαφής veth-br που συνδέεται με την διεπαφή veth-app του namespace
ip link add veth-app type veth peer name veth-br

//σύνδεση διεπαφής veth-br με το br-sfc
ovs-vsctl add-port br-sfc veth-br

//ενεργοποίηση veth-br
ip link set dev veth-br up

//ορισμός του veth-app ως διεπαφής του app
ip link set veth-app netns app
```

Στο classifier1 VM θα κάνουμε τις παρακάτω ρυθμίσεις για την δημιουργία του Client:

```
//απόδοση διεύθυνσης IP
ip netns exec app ifconfig veth-app 192.168.2.1/24 up

//απόδοση διεύθυνσης MAC
ip netns exec app ip link set dev veth-app  addr 00:00:11:11:11:11

//καταχώρηση στο πίνακα arp για την διεπαφή veth-app
ip netns exec app arp -s 192.168.2.2 00:00:22:22:22:22 -i veth-app

//ενεργοποίηση της veth-app
ip netns exec app ip link set dev veth-app up

//ενεργοποίηση του loopback της app
ip netns exec app ip link set dev lo up

//ρύθμιση της mtu
ip netns exec app ifconfig veth-app mtu 1400

//αποστολή αιτήματος στον server(αφού δημιουργηθεί η αλυσίδα)
ip netns exec app wget http://192.168.2.2
```

Στο classifier2 VM θα κάνουμε τις παρακάτω ρυθμίσεις για την δημιουργία και εκκίνηση του Http Server:

```
//απόδοση διεύθυνσης IP
ip netns exec app ifconfig veth-app 192.168.2.2/24 up

//απόδοση διεύθυνσης MAC
ip netns exec app ip link set dev veth-app  addr 00:00:22:22:22:22

//καταχώρηση στο πίνακα arp για την διεπαφή veth-app
ip netns exec app arp -s 192.168.2.1 00:00:11:11:11:11 -i veth-app

//ενεργοποίηση της veth-app
ip netns exec app ip link set dev veth-app up

//ενεργοποίηση του loopback της app
ip netns exec app ip link set dev lo up

//ρύθμιση της mtu
ip netns exec app ifconfig veth-app mtu 1400

//έναρξη του Http Server στην θύρα 80
ip netns exec app python -m SimpleHTTPServer 80
```

## Αρχείο setup.py για την δημιουργία αλυσίδας SFC

```
#!/usr/bin/python
import argparse
import requests,json
from requests.auth import HTTPBasicAuth
from subprocess import call
import time
import sys
import os

//IP και θύρα ODL
controller='192.168.1.5'
DEFAULT_PORT='8181'

//στοιχεία ταυτοποιήσης χρήστη ODL
USERNAME='admin'
PASSWORD='admin'

//δημιουργία GET
def get(host, port, uri):
   url='http://'+host+":"+port+uri
   r = requests.get(url, auth=HTTPBasicAuth(USERNAME, PASSWORD))
   jsondata=json.loads(r.text)
   return jsondata

//δημιουργία PUT
def put(host, port, uri, data, debug=False):
   '''Perform a PUT rest operation, using the URL and data provided'''

   url='http://'+host+":"+port+uri

   headers = {'Content-type': 'application/yang.data+json',
           'Accept': 'application/yang.data+json'}
   if debug == True:
      print "PUT %s" % url
      print json.dumps(data, indent=4, sort_keys=True)
   r = requests.put(url, data=json.dumps(data), headers=headers, auth=HTTPBasicAuth(USERNAME, PASSWORD))
   if debug == True:
      print r.text
   r.raise_for_status()
   time.sleep(5)

//δημιουργία POST
def post(host, port, uri, data, debug=False):
   '''Perform a POST rest operation, using the URL and data provided'''

   url='http://'+host+":"+port+uri
   headers = {'Content-type': 'application/yang.data+json',
           'Accept': 'application/yang.data+json'}
   if debug == True:
      print "POST %s" % url
      print json.dumps(data, indent=4, sort_keys=True)
   r = requests.post(url, data=json.dumps(data), headers=headers, auth=HTTPBasicAuth(USERNAME, PASSWORD))
   if debug == True:
      print r.text
   r.raise_for_status()
   time.sleep(5)
```

```
//URI για Service Nodes
def get_service_nodes_uri():
   return "/restconf/config/service-node:service-nodes"

//JSON για Service Nodes
def get_service_nodes_data():
   return {
   "service-nodes": {
      "service-node": [
         {
            "name": "node0",
            "service-function": [
            ],
            "ip-mgmt-address": "192.168.1.10"
         },
         {
            "name": "node1",
            "service-function": [
            ],
            "ip-mgmt-address": "192.168.1.20"
         },
         {
            "name": "node2",
            "service-function": [
               "dpi-1"
            ],
            "ip-mgmt-address": "192.168.1.30"
         },
         {
            "name": "node3",
            "service-function": [
               "firewall-1"
            ],
            "ip-mgmt-address": "192.168.1.40"
         },
         {
            "name": "node4",
            "service-function": [
            ],
            "ip-mgmt-address": "192.168.1.50"
         },
         {
            "name": "node5",
            "service-function": [
            ],
            "ip-mgmt-address": "192.168.1.60"
         }
      ]
   }
}

//URI για Service Functions
def get_service_functions_uri():
   return "/restconf/config/service-function:service-functions"

//JSON για Service Functions
def get_service_functions_data():
   return {
   "service-functions": {
```

```
    "service-function": [
      {
        "name": "dpi-1",
        "ip-mgmt-address": "192.168.1.30",
        "rest-uri": "http://192.168.1.30:5000",
        "type": "dpi",
        "nsh-aware": "true",
        "sf-data-plane-locator": [
          {
            "name": "sf1-dpl",
            "port": 6633,
            "ip": "192.168.1.30",
            "transport": "service-locator:vxlan-gpe",
            "service-function-forwarder": "SFF1"
          }
        ]
      },
      {
        "name": "firewall-1",
        "ip-mgmt-address": "192.168.1.40",
        "rest-uri": "http://192.168.1.40:5000",
        "type": "firewall",
        "nsh-aware": "true",
        "sf-data-plane-locator": [
          {
            "name": "sf2-dpl",
            "port": 6633,
            "ip": "192.168.1.40",
            "transport": "service-locator:vxlan-gpe",
            "service-function-forwarder": "SFF2"
          }
        ]
      }
    ]
  }
}
```

**//URI για Service Function Forwarders**

```
def get_service_function_forwarders_uri():
  return "/restconf/config/service-function-forwarder:service-function-forwarders"
```

**//JSON για Service Function Forwarders**

```
def get_service_function_forwarders_data():
  return {
  "service-function-forwarders": {
    "service-function-forwarder": [
      {
        "name": "SFF0",
        "service-node": "node0",
        "service-function-forwarder-ovs:ovs-bridge": {
          "bridge-name": "br-sfc",
        },
        "sff-data-plane-locator": [
          {
            "name": "sff0-dpl",
            "data-plane-locator": {
              "transport": "service-locator:vxlan-gpe",
              "port": 6633,
              "ip": "192.168.1.10"
```

```
        },
        "service-function-forwarder-ovs:ovs-options": {
          "remote-ip": "flow",
          "dst-port": "6633",
          "key": "flow",
          "nsp": "flow",
          "nsi": "flow",
          "nshc1": "flow",
          "nshc2": "flow",
          "nshc3": "flow",
          "nshc4": "flow"
        }
      }
    ],
},
{
    "name": "SFF1",
    "service-node": "node1",
    "service-function-forwarder-ovs:ovs-bridge": {
      "bridge-name": "br-sfc",
    },
    "sff-data-plane-locator": [
      {
        "name": "sff1-dpl",
        "data-plane-locator": {
          "transport": "service-locator:vxlan-gpe",
          "port": 6633,
          "ip": "192.168.1.20"
        },
        "service-function-forwarder-ovs:ovs-options": {
          "remote-ip": "flow",
          "dst-port": "6633",
          "key": "flow",
          "nsp": "flow",
          "nsi": "flow",
          "nshc1": "flow",
          "nshc2": "flow",
          "nshc3": "flow",
          "nshc4": "flow"
        }
      }
    ],
    "service-function-dictionary": [
      {
        "name": "dpi-1",
        "sff-sf-data-plane-locator": {
           "sf-dpl-name": "sf1-dpl",
           "sff-dpl-name": "sff1-dpl"
        }
      }
    ],
},
{
    "name": "SFF2",
    "service-node": "node4",
    "service-function-forwarder-ovs:ovs-bridge": {
      "bridge-name": "br-sfc",
    },
    "sff-data-plane-locator": [
```

```
        {
          "name": "sff2-dpl",
          "data-plane-locator": {
            "transport": "service-locator:vxlan-gpe",
            "port": 6633,
            "ip": "192.168.1.50"
          },
          "service-function-forwarder-ovs:ovs-options": {
            "remote-ip": "flow",
            "dst-port": "6633",
            "key": "flow",
            "nsp": "flow",
            "nsi": "flow",
            "nshc1": "flow",
            "nshc2": "flow",
            "nshc3": "flow",
            "nshc4": "flow"
          }
        }
      ],
      "service-function-dictionary": [
        {
          "name": "firewall-1",
          "sff-sf-data-plane-locator": {
            "sf-dpl-name": "sf2-dpl",
            "sff-dpl-name": "sff2-dpl"
          }
        }
      ]
    },
    {
      "name": "SFF3",
      "service-node": "node5",
      "service-function-forwarder-ovs:ovs-bridge": {
        "bridge-name": "br-sfc",
      },
      "sff-data-plane-locator": [
        {
          "name": "sff3-dpl",
          "data-plane-locator": {
            "transport": "service-locator:vxlan-gpe",
            "port": 6633,
            "ip": "192.168.1.60"
          },
          "service-function-forwarder-ovs:ovs-options": {
            "remote-ip": "flow",
            "dst-port": "6633",
            "key": "flow",
            "nsp": "flow",
            "nsi": "flow",
            "nshc1": "flow",
            "nshc2": "flow",
            "nshc3": "flow",
            "nshc4": "flow"
          }
        }
      ],
    }
]
```

```
    }
}

//URI για Service Function Chains
def get_service_function_chains_uri():
    return "/restconf/config/service-function-chain:service-function-chains/"

//JSON για Service Function Chains
def get_service_function_chains_data():
    return {
    "service-function-chains": {
       "service-function-chain": [
          {
             "name": "SFC1",
             "symmetric": "true",
             "sfc-service-function": [
                {
                   "name": "dpi-abstract1",
                   "type": "dpi"
                },
                {
                   "name": "firewall-abstract1",
                   "type": "firewall"
                }
             ]
          }
       ]
    }
}

//URI για Service Function Paths
def get_service_function_paths_uri():
    return "/restconf/config/service-function-path:service-function-paths/"

//JSON για Service Function Paths
def get_service_function_paths_data():
    return {
    "service-function-paths": {
       "service-function-path": [
          {
             "name": "SFP1",
             "service-chain-name": "SFC1",
             "starting-index": 255,
             "symmetric": "true",
              "context-metadata": "NSH1"
          }
       ]
    }
}

//URI για Service Function Metadata
def get_service_function_metadata_uri():
    return "/restconf/config/service-function-path-metadata:service-function-metadata/"

//JSON για Service Function Metadata
def get_service_function_metadata_data():
    return {
  "service-function-metadata": {
    "context-metadata": [
```

```
    {
      "name": "NSH1",
      "context-header1": "1",
      "context-header2": "2",
      "context-header3": "3",
      "context-header4": "4"
    }
   ]
  }
}
```

**//URI για Service Function Paths**

```
def get_service_function_paths_uri():
   return "/restconf/config/service-function-path:service-function-paths/"
```

**//JSON για Service Function Paths Data**

```
def get_service_function_paths_data():
   return {
  "service-function-paths": {
   "service-function-path": [
    {
      "name": "SFP1",
      "service-chain-name": "SFC1",
      "classifier": "Classifier1",
      "symmetric-classifier": "Classifier2",
      "context-metadata": "NSH1",
      "symmetric": "true"
    }
   ]
  }
}
```

**//URI για Rendered Service Path**

```
def get_rendered_service_path_uri():
   return "/restconf/operations/rendered-service-path:create-rendered-path/"
```

**//JSON για Rendered Service Path**

```
def get_rendered_service_path_data():
   return {
   "input": {
      "name": "RSP1",
      "parent-service-function-path": "SFP1",
      "symmetric": "true"
   }
}
```

**//URI για ACL**

```
def get_service_function_acl_uri():
   return "/restconf/config/ietf-access-control-list:access-lists/"
```

**//JSON για ACL**

```
def get_service_function_acl_data():
   return  {
  "access-lists": {
   "acl": [
    {
      "acl-name": "ACL1",
      "access-list-entries": {
       "ace": [
```

```
        {
          "rule-name": "ACE1",
          "actions": {
            "service-function-acl:rendered-service-path": "RSP1"
          },
          "matches": {
            "destination-ipv4-network": "192.168.2.0/24",
            "source-ipv4-network": "192.168.2.0/24",
            "protocol": "6",
            "source-port-range": {
               "lower-port": 0
            },
            "destination-port-range": {
               "lower-port": 80
            }
          }
        }
       ]
      }
     },
     {
      "acl-name": "ACL2",
      "access-list-entries": {
       "ace": [
        {
          "rule-name": "ACE2",
          "actions": {
            "service-function-acl:rendered-service-path": "RSP1-Reverse"
          },
          "matches": {
            "destination-ipv4-network": "192.168.2.0/24",
            "source-ipv4-network": "192.168.2.0/24",
            "protocol": "6",
            "source-port-range": {
               "lower-port": 80
            },
            "destination-port-range": {
               "lower-port": 0
            }
          }
        }
       ]
      }
     }
   ]
  }
}
```

**//URI για Service Function Classifiers**

```
def get_service_function_classifiers_uri():
   return "/restconf/config/service-function-classifier:service-function-classifiers/"
```

**//JSON για Service Function Classifiers**

```
def get_service_function_classifiers_data():
   return  {
  "service-function-classifiers": {
   "service-function-classifier": [
    {
      "name": "Classifier1",
```

```
      "scl-service-function-forwarder": [
       {
         "name": "SFF0",
         "interface": "veth-br"
       }
      ],
      "access-list": "ACL1"
     },
     {
      "name": "Classifier2",
      "scl-service-function-forwarder": [
       {
         "name": "SFF3",
         "interface": "veth-br"
       }
      ],
      "access-list": "ACL2"
     }
   ]
  }
}
```

**//εκτέλεση**

```
if __name__ == "__main__":
print "sending service nodes"
put(controller, DEFAULT_PORT, get_service_nodes_uri(), get_service_nodes_data(), True)
print "sending service functions"
put(controller, DEFAULT_PORT, get_service_functions_uri(), get_service_functions_data(), True)
print "sending service function forwarders"
put(controller, DEFAULT_PORT, get_service_function_forwarders_uri(), get_service_function_forwarders_data(), True)
print "sending service function chains"
put(controller, DEFAULT_PORT, get_service_function_chains_uri(), get_service_function_chains_data(), True)
print "sending service function metadata"
put(controller, DEFAULT_PORT, get_service_function_metadata_uri(), get_service_function_metadata_data(), True)
print "sending service function paths"
put(controller, DEFAULT_PORT, get_service_function_paths_uri(), get_service_function_paths_data(), True)
print "sending service function acl"
put(controller, DEFAULT_PORT, get_service_function_acl_uri(), get_service_function_acl_data(), True)
print "sending rendered service path"
post(controller, DEFAULT_PORT, get_rendered_service_path_uri(), get_rendered_service_path_data(), True)
print "sending service function classifiers"
put(controller, DEFAULT_PORT, get_service_function_classifiers_uri(), get_service_function_classifiers_data(), True)
```