Πανεπιστήμιο Πειραιά
Τμήμα Ψηφιακών Συστημάτων
ΠΜΣ: Ψηφιακές Επικοινωνίες και Δίκτυα

ΔΙΑΧΕΙΡΙΣΗ ΤΟΥ SOFTWARE DEFINED NETWORKING (SDN) ΜΕΣΩ ΤΟΥ OPENDAYLIGHT ΕΛΕΓΚΤΗ

ΨΗΜΙΤΗΣ – ΧΡΙΣΤΟΔΟΥΛΟΠΟΛΟΣ ΑΝΤΩΝΗΣ
ΜΕ 14042
Φεβρουάριος 2017

Περίληψη

Η εργασία επικεντρώνεται στη διαχείριση του Software Defined Networking (SDN) μέσα από δύο εφαρμογές, πιο συγκεκριμένα δύο *Application Programming Interfaces(APIs),* που η μία αφορά την τοπολογία ενός δικτύου και η άλλη την εμφάνιση στατιστικών δεδομένων από τους κόμβους που αποτελούν αυτή την δικτυακή τοπολογία. Πρωταγωνιστικό ρόλο παίζει ο OpenDaylight ελεγκτής, ο οποίος αποτελεί την απαραίτητη πλατφόρμα λογισμικού, στην οποία έχουν αναπτυχθεί οι εφαρμογές αυτές, και πραγματοποιούνται ο έλεγχος και η παρακολούθηση του δικτύου. Το δίκτυο αυτό έχει στηθεί εικονικά μέσω του εικονικού προσομοιωτή δικτύων, το Mininet.

Στόχο της εργασίας αποτελεί η παρουσίαση και η κατανόηση της νέας τάσης για αλλαγή στο σχεδιασμό και την υλοποίηση των δικτύων μέσω του SDN. Το SDN υπόσχεται να φέρει μεγάλες αλλαγές στην διαχείριση, την ευελιξία, τον έλεγχο αλλά κυρίως στην απόδοση των δικτύων που πρέπει να ανταπεξέλθουν στις νέες τάσεις και τεχνολογίες. Η βασική διαφοροποίηση του νέου τρόπου δικτύωσης από τον παραδοσιακό είναι ο διαχωρισμός του *επιπέδου ελέγχου* από το *επίπεδο προώθησης δεδομένων* με αποτέλεσμα το λογισμικό να αποσπάτε από το υλικό μίας δικτυακή συσκευής, έτσι ώστε η συσκευή να μην είναι πλέον αυτόνομη αλλά να ελέγχεται από ένα κεντρικό λογισμικό το οποίο έχει στην υπό βλέψη του όλες τις συσκευές του δικτυού. Έτσι ξεκινάει η μεγάλη αλλαγή της δικτύωσης μετατρέποντας πλέον τα νέα δίκτυα από κατανεμημένα σε κεντροποιημένα.

Η εργασία προσεγγίζει το SDN, μέσω του OpenDaylight, ως προς την δομή του , την αρχιτεκτονική του, την κατασκευή του, τη λειτουργικότητά και την χρησιμότητα του από τους διαχειριστές δικτύων. Αναφέρεται στο πρωτόκολλο OpenFlow και το σημαντικό ρόλο που παίζει στη συνεχόμενη ανάπτυξη του SDN και του OpenDaylight. Περιγράφονται σε αυτήν αναλυτικά όλα τα στοιχεία που δομούν τον ελεγκτή και τα βασικά στοιχεία που χρειάζονται για την εγκατάστασή του. Τέλος, αφότου έχουν γίνει κατανοητά όλα τα παραπάνω, εφαρμόζονται τα API (network-topology και opendaylight-inventory) που δείχνουν έμπρακτα πλέον τις δυνατότητες του SDN μέσω του OpenDaylight ελεγκτή.

Περιεχόμενα

Εισαγωγή

Μερικά από τα σημαντικότερα θέματα στον τομέα της σύγχρονης δικτύωσης, αποτελεί η ανάπτυξη του **Software Defined Networking (SDN)** και του **Network Virtualization (NV)**. Οι αυξημένες απαιτήσεις, όσο και οι ανάγκες των εταιρειών παροχής πληροφοριακών συστημάτων και κινητής τηλεφωνίας, αποτελούν αναμφισβήτητα, έναν από τους βασικούς παράγοντες, ανάπτυξης καινοτόμων ιδεών, από την πλευρά των κατασκευαστών δικτύων και δικτυακών συσκευών.

Το **Software Defined Networking (SDN)** αποτελεί σήμερα ένα από σημαντικότερα παραδείγματα της μετατόπισης προς την βιομηχανία των δικτύων. Το SDN δημιουργήθηκε εξαιτίας των νέων ισχυρών απαιτήσεων και αναγκών, που πρέπει να ανταπεξέλθουν τα σύγχρονα δίκτυα ως προς τις τάσεις του εικονικού υπολογιστικού νέφους (***virtualized cloud computing***) και των εφαρμογών μεγάλου όγκου δεδομένων (***Big Data***). Στόχος τους είναι να επιδιώξουν όσο το δυνατόν γρηγορότερα και αποτελεσματικά μια αυτοματοποιημένη διαχείριση μεγάλων δικτύων.

Υπάρχει, όμως, μεγάλη σύγχυση μεταξύ των επιχειρήσεων πληροφορικής σε σχέση με θέματα οργάνωσης και ανάπτυξης του SDN. Ο αριθμός των προμηθευτών είναι μεγάλος και ο εκάστοτε προμηθευτής ισχυρίζεται ότι προσφέρει SDN ή / και NV λύσεις καθώς έχει την δυνατότητα να παρέχει λύσεις που λύνουν διαφορετικά προβλήματα χρησιμοποιώντας διαφορετικές αρχιτεκτονικές και τεχνολογίες.

Παρόλα αυτά το SDN μπορεί δυνητικά να παρέχει μία πολύ σημαντική αξία σε οργανισμούς και επιχειρήσεις πληροφορικής. Βέβαια το περιβάλλον του SDN και οι εφαρμογές που έχουν αναπτυχθεί είναι σε στάδιο το οποίο δεν είναι εντελώς κατάλληλο για να εδραιωθεί στον χώρο των εταιριών. Με δεδομένη την ανωριμότητα των σημερινών προϊόντων, τα πρότυπα και τα Application Programming Interfaces(APIs), οποιασδήποτε εταιρία πληροφορικής που θέλει να εφαρμόσει το SDN πρότυπο ,στο άμεσο μέλλον. θα πρέπει να το κάνει με περιορισμένο τρόπο, δηλαδή μια εφαρμογή SDN για μια πολύ συγκεκριμένη περίπτωση χρήσης.

Οι οργανισμοί πληροφορικής μπορούν να αναμένουν μια συνεχή σειρά από ανακοινώσεις τόσο από πλευράς νέων προϊόντων, καθώς και νέων ή βελτιωμένων πρωτοκόλλων, τα οποία δείχνουν πώς τα βασικά δομικά στοιχεία ενός SDN τείνουν να σταθεροποιηθούν. Υπάρχουν, όμως, δύο γεγονότα που πρέπει να συμβούν προκειμένου το SDN να είναι έτοιμο για την επικρατούσα τάση της αγοράς. Το ένα είναι η διαθεσιμότητα των λειτουργιών που δίνουν τη δυνατότητα στους οργανισμούς να διαχειρίζονται αποτελεσματικά αυτή τη νέα μορφή δικτύωσης και το δεύτερο είναι η διαθεσιμότητα ενός ευρύ φάσματος εφαρμογών που αξιοποιούν το κεντρικό έλεγχο και συνδέονται με τις περισσότερες μορφές ενός SDN.

Η πλατφόρμα OpenDaylight έχε την ικανότητα να συγκεντρώνει μία μεγάλη ποικιλία διαφόρων τύπου δικτύου, κάτω από ένα κοινό πλαίσιο ελέγχου και διαχείρισης. Χρησιμοποιεί ως βάση το πρωτόκολλο OpenFlow, μέσω του οποίου ελέγχονται οι «εικονικοί διακόπτες», ή αλλιώς virtual switches. Επίσης σε συνεργασία με το πρωτόκολλο OpenFlow, πρωτόκολλα όπως το NETCONF, τα BGP / LS / PCEP, εξυπηρετούν στον ίδιο βαθμό, την επέκταση των υπηρεσιών μέσα από μια σειρά διαφορετικών τύπων συσκευών δικτύου.

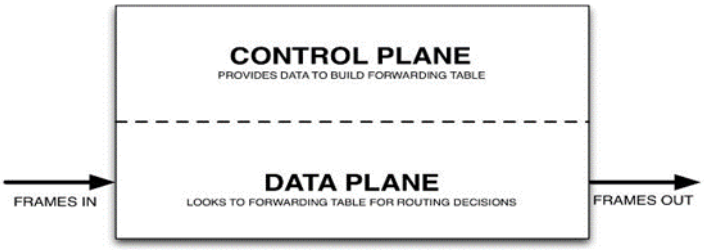
Η δυνατότητα σύνδεσης ενός ευρύ φάσματος δικτύων σε μια κοινή ένωση υπηρεσιών αποτελεί βασική ικανότητα της αρχιτεκτονικής OpenDaylight.Τα δίκτυα αυξάνονται με εκθετικό ρυθμό στον κόσμο της κινητής τηλεφωνίας και αντιπροσωπεύουν πλέον το μεγαλύτερο μέρος της κίνησης στο διαδίκτυο καθώς το Internet of Things (IoT) έχει αυξηθεί κατά πολύ. Ο πυρήνας του κάθε δικτύου συνδέει τους χρήστες και τις συσκευές, ενώ η ευελιξία της πλατφόρμας, εξ αποστάσεως, παρέχει μια κοινή διεπαφή για να ώστε να συνδέσει τους χρήστες με τις εφαρμογές, ανεξάρτητα από το πώς θα έχει συνδεθεί ο καθένας στο δίκτυο.

Κεφάλαιο 1
Software Defined Networking και το OpenFlow

1.1 Software Defined Networking (SDN)

Όπως αναφέρθηκε στην εισαγωγή, το SDN με την βοήθεια των προγραμματιστικών εφαρμογών που επικοινωνούν με το κεντροποιημένο έλεγχο όλων των δικτυακών συσκευών , η διαχείριση μεγάλων δικτύων φαίνεται να γίνεται πιο εύκολη και ευέλικτη. Το SDN χρησιμοποιεί το **OpenFlow** πρωτόκολλο , το οποίο δίνει την δυνατότητα στις δικτυακές συσκευές, ανεξαρτήτως κατασκευαστή, να μπορούν να επικοινωνούν μεταξύ τους και να ελέγχονται από τον κεντρικό ελεγκτή του δικτύου. Το OpenFlow πρωτόκολλο το συναντάμε στο επίπεδο 2 και 3 ενός δικτύου.

Για να ορίσουμε την έννοια ενός **Software Defined Networking(SDN)** θα μπορούσαμε να το προσδιορίσουμε ως τον φυσικό διαχωρισμό του **επιπέδου ελέγχου (control plane)** από το **επίπεδο δεδομένων (data plane)**, καθώς και τον τρόπο ελέγχου των διαφόρων συσκευών που αποτελούν το δίκτυο.



Σχήμα 1 Ο διαχωρισμός του επιπέδου Ελέγχου με το επίπεδο ης προώθησης των Δεδομένων

Η ιδέα του διαχωρισμού αυτού δεν αποτελεί μια καινούργια ιδέα στον τομέα της δικτύωσης. Έχει εμφανιστεί στο παρελθόν ως ένα βασικό χαρακτηριστικό των MPLS (Multi-Protocol Label Switching) δικτύων, όπως επίσης και πολλών σύγχρονων δικτύων Wi-Fi.

Σιγά σιγά, ο σχεδιασμός και ο τρόπος λειτουργίας των δικτύων θα αναπτυχθεί διαφορετικά και ενώ έως τώρα γνωρίζουμε μέσα από τα παραδοσιακά δίκτυα έναν κατανεμημένο σχεδιασμό , πλέον θα αρκεστούμε σε έναν κεντροποιημένο και περισσότερο λειτουργικό. Μέσα από την ανάλυση της αρχιτεκτονικής του SDN , τη δομή και την λειτουργία του OpenFlow αλλά και τον τρόπο ανάπτυξης της SDN εφαρμογής , μπορεί να γίνουν περισσότερο κατανοητές οι νέες αλλαγές.

1.2 Η έξοδος από τα παραδοσιακά δίκτυα

Η συνεχόμενη ανάπτυξη των κινητών συσκευών, η σύνδεσή τους με το διαδίκτυο και σιγά σιγά με το IoT, η εικονοποιήση των server και η έλευση των υπηρεσιών Cloud είναι από τις τάσεις που οδηγούν τη βιομηχανία δικτύωσης να επανεξετάσει τις παραδοσιακές αρχιτεκτονικές των δικτύων. Όλα τα συμβατικά δίκτυα είναι ιεραρχικά χτισμένα με βαθμίδες και χρησιμοποιούν

διακόπτες Ethernet τοποθετημένους σε μια δομή δέντρου. Αυτό το σχέδιο έχει νόημα όταν ο τύπος επικοινωνίας ‘client-server’ κυριαρχεί, αλλά μια τέτοια στατική αρχιτεκτονική είναι ακατάλληλη προς τις δυναμικές εξελίξεις της πληροφορική και τις σημερινές ανάγκες αποθήκευσης δεδομένων των επιχειρήσεων, πανεπιστημιουπόλεων, και φορέων που έχουν να κάνουν με τη διαχείριση και αποθήκευση μεγάλου όγκου δεδομένων. Έτσι λοιπόν τα παραδοσιακά δίκτυα δεν μπορούν να ανταπεξέλθουν σε κάποιες από τις βασικές υπολογιστικές τάσεις όπως :

- **Changing traffic patterns** *(Διακυμάνσεις κίνησης):* Εφαρμογές που συνήθως έχουν πρόσβαση σε γεωγραφικά κατανεμημένες βάσεις δεδομένων και servers μέσω δημόσιων και ιδιωτικών Cloud, απαιτούν εξαιρετικά ευέλικτη διαχείριση της κυκλοφορίας και της πρόσβασης, όσο αφορά το εύρος ζώνης της ζήτησης.

- **"The “consumerization of IT”(***Η «καταναλωτικοποίηση της πληροφορικής):* Η "Φέρτε" τη δική σας συσκευή (BYOD) τάση απαιτεί δίκτυα που να είναι τόσο ευέλικτα όσο και ασφαλές.

- **The rise of cloud services** *(Η άνοδος των υπηρεσιών Cloud):* Οι χρήστες αναμένουν on-demand πρόσβαση σε εφαρμογές, υποδομές, και άλλους πόρους πληροφορικής.

- **“Big data” means more bandwidth** *("Big Data" σημαίνει μεγαλύτερο εύρος ζώνης):* Ο χειρισμός σημερινών μεγάλου όγκου δεδομένων απαιτεί μαζική παράλληλη επεξεργασία ώστε να τροφοδοτεί μια σταθερή ζήτηση για επιπλέον δυναμικότητα και any-to-any συνδεσιμότητα.

Οι σχεδιαστές δικτύων περιορίζονται από τους περιορισμούς των παραδοσιακών δικτύων στα εξής :

- **Complexity that leads to stasis** *(Πολυπλοκότητα που οδηγεί σε στασιμότητα):* Η προσθήκη-αφαίρεση συσκευών και η εφαρμογή πολιτικών σε όλο το δίκτυο είναι πολύπλοκες, χρονοβόρες, και κυρίως ριψοκίνδυνες, αποθαρρύνοντας έτσι οποιαδήποτε αλλαγή στο δικτύου.

- **Inability to scale** *(Αδυναμία να αναβαθμιστούν):* Η προσέγγιση της σύνδεσης προεγγραφής για την παροχή κλιμάκωσης δεν είναι αποτελεσματική με τα δυναμικά μοτίβα κυκλοφορίας σε εικονικά δίκτυα. Ένα πρόβλημα που είναι ακόμη πιο έντονο σε δίκτυα παροχής υπηρεσιών με μεγάλης κλίμακας αλγόριθμων παράλληλης επεξεργασίας καθώς και των συνόλων δεδομένων σε ένα ευρύ σύνολο υπολογιστών.

- **Vendor dependence** *(Εξάρτηση από τον προμηθευτή):* Οι συσκευές εξοπλισμού επωνύμων προμηθευτών και η έλλειψη προτύπου σε ανοικτές διεπαφές, περιορίζουν τη δυνατότητα των φορέων εκμετάλλευσης του δικτύου να προσαρμόσει το δίκτυο σε μεμονωμένα χώρους.

[1]

Στα παραδοσιακά δίκτυα, οι μεταγωγοί πακέτων, δρομολογητές και άλλες συσκευές του δικτύου διοικούνται ξεχωριστά, και επικεντρώνονται στις ανάγκες της συσκευής αντί για τις ανάγκες των εφαρμογών. Το SDN ελέγχει μεμονωμένες συσκευές, δίνοντας στους διαχειριστές μία end-to-end προβολή των ροών του δικτύου, καθώς και την ικανότητα να βελτιστοποιούν της διαδρομές της ροής των πακέτων.

Το SDN επικεντρώνεται κυρίως στην παροχή προγραμματιστικής διασύνδεσης με τον εξοπλισμό του εκάστοτε δικτύου, γι’ αυτό και η αξία του αποκτά μεγαλύτερο εύρος καθώς η

μείωση της λανθάνουσας κατάστασης ενός δικτύου είναι από τα βασικά τους επιτεύγματα. Ο προγραμματιστικός έλεγχος με τον οποίο τα δεδομένα ρέουν μέσα από ένα δίκτυο διευκολύνει τη διαχειρισιμότητα, υποστηρίζει την αυτοματοποίηση, τον εντοπισμό σφαλμάτων και την επιδιόρθωσή τους, τον έλεγχο μεγάλων αριθμών δικτυακών συσκευών, υπηρεσιών και πολιτικών δικτύου χρησιμοποιώντας υψηλού επιπέδου γλώσσες προγραμματισμού και APIs(Application Programming Interfaces). Έτσι βελτιώνουν κατά πολύ την επίδοση του δικτύου, την παρακολούθηση του, την λειτουργικότητα του και βοηθά τους διαχειριστές να παραδώσουν πιο γρήγορα τις εξατομικευμένες υπηρεσίες που είναι υψίστης σημασίας για το σημερινό δυναμικό επιχειρηματικό περιβάλλον.

1.3 SDN αρχιτεκτονική

Το **Software Defined Networking (SDN)** αποτελείται από μία αναδυόμενη αρχιτεκτονική που είναι δυναμική, εύχρηστη, αποδοτική, και προσαρμόσιμη, καθιστώντας τη ιδανική για τις, υψηλού εύρους ζώνης, σημερινές εφαρμογές. Αυτή η αρχιτεκτονική αποσυνδέει τον έλεγχο του δικτύου και με την προώθηση των λειτουργιών επιτρέπει ο έλεγχος του δικτύου να γίνει άμεσα και προγραμματιζόμενα με τη σχετική υποδομή που πρέπει να αντληθεί για τις εφαρμογές και τις υπηρεσίες του δικτύου.

Η αρχιτεκτονική των SDN δικτύων καθορίζεται από τα παρακάτω χαρακτηριστικά:

- **Directly programmable (**Άμεσα προγραμματιζόμενη**)**: Ο έλεγχος του δικτύου προγραμματίζεται άμεσα επειδή είναι αποσυνδεδεμένος από τις λειτουργίες προώθησης.

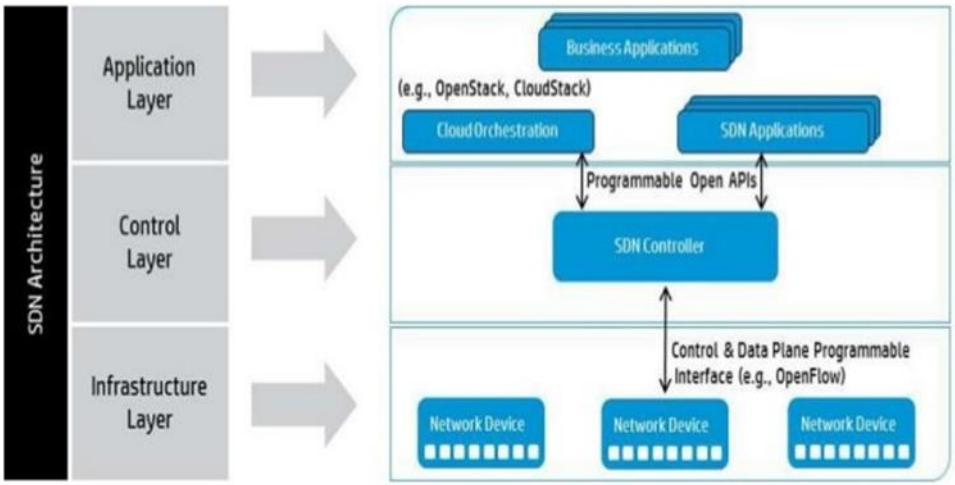
- **Agile (**Ευέλικτη**):** Διαχωρίζοντας το επίπεδο ελέγχου από το επίπεδο προώθησης, επιτρέπεται στους διαχειριστές να προσαρμόσουν δυναμικά τη ροή της κυκλοφορίας σε όλο το δίκτυο καθώς και να ανταποκρίνονται στις μεταβαλλόμενες ανάγκες.

- **Central Managed (**Κεντροποιημένη**):** Η νοημοσύνη του δικτύου είναι “λογικά” συγκεντρωμένη από ελεγκτές SDN, οι οποίοι αποτελούνται από λογισμικό το οποίο βοηθάει στην διατήρηση μίας συνολικής εικόνας του δικτύου. Παίζει τον ρόλο του ενιαίου και λογικού διακόπτη σε εφαρμογές και μηχανές πολιτικής (policy engines).

- **Programmatically configured (**Προγραμματιστικές παραμετροποιήσεις**):** Τα SDN επιτρέπουν στους διαχειριστές του δικτύου να ρυθμίσουν, διαχειριστούν, ασφαλίσουν, και να βελτιστοποιήσουν τους πόρους του δικτύου πολύ γρήγορα μέσω δυναμικών, αυτοματοποιημένων προγραμμάτων SDN, τα οποία μπορούν οι ίδιοι να γράφουν, επειδή τα προγράμματα δεν εξαρτώνται από το ιδιόκτητο λογισμικό.

- **Open standards-based and vendor-neutral:** Με την υλοποίηση ανοικτών προτύπων, το SDN απλοποιεί το σχεδιασμό του δικτύου και τη λειτουργία επειδή οι οδηγίες παρέχονται από τους ελεγκτές SDN αντί από πολλές, συγκεκριμένες συσκευές καθώς και πρωτόκολλα.

[1]



Σχήμα 2 : Διάγραμμα της αρχιτεκτονικής ενός SDN

Τα επίπεδα της αρχιτεκτονική ενός SDN φαίνονται στο **Σχήμα 2**. Σε αυτή την αρχιτεκτονική, η λειτουργικότητα του επιπέδου ελέγχου (**control layer**) είναι κεντρική στο λογισμικό του ελεγκτή SDN. Τα επίπεδα είναι τρία:

- **Application Layer:** Στο επίπεδο αυτό αναπτύσσονται εφαρμογές που χρησιμοποιούνται άμεσα από τους τελικούς χρήστες. Οι εφαρμογές αυτές μπορούν να είναι είτε SDN εφαρμογές και να δίνουν δυνατότητες ελέγχου, διαχείρισης και παρακολούθησης του δικτύου με τη βοήθεια, παραδείγματος χάρη, στατιστικών πινάκων ή εμφάνισης της τοπολογίας του δικτύου. Επίσης μπορούνε να κτιστούν εφαρμογές οι οποίες συμβαδίζουν με εφαρμογές που σχετίζονται με υπολογιστικά νέφη (Cloud Computing),όπως το OpenStack, το CloudStack και πολλές άλλες, καθώς και εφαρμογές που σχετίζονται με την επιχειρηματική λογική και τις πελατειακές σχέσεις.

- **Control Layer:** Στο επίπεδο αυτό του δικτύου βρίσκεται ο SDN ελεγκτής που προσφέρει μια κεντρική άποψη του συνολικού δικτύου και επιτρέπει στους διαχειριστές του δικτύου να ενημερώνουν τους διακόπτες και τους δρομολογητές, πως το επίπεδο προώθησης θα πρέπει να χειριστεί την κυκλοφορία του δικτύου.
  Οι εφαρμογές του επιπέδου εφαρμογών έχουν γραφτεί σε ένα σύνολο από APIs που παρέχονται από τον ελεγκτή SDN τα οποία όμως δεν είναι τυποποιημένα. Άρα μια εφαρμογή που τρέχει σε έναν ελεγκτή SDN, θα πρέπει να τροποποιηθεί για να μπορέσει να τρέξει σε έναν άλλο ελεγκτή SDN, καθώς και να προσαρμοστεί σε αυτό.

- **Infrastructure Layer:** Στο επίπεδο της υποδομής βρίσκεται όλες οι συσκευές που αποτελούν το εκάστοτε δίκτυο και επικοινωνούνε με το επίπεδο ελέγχου μέσω πρωτοκόλλων διεπαφής τα οποία χρησιμοποιούνται για τον προγραμματισμό της συμπεριφοράς της προώθηση πακέτων από τον διακόπτη.

Οι πάροχοι των SDN προσφέρουν μια ευρεία επιλογή ανταγωνιστικών αρχιτεκτονικών, αλλά στη πιο απλή του μορφή, η SDN μέθοδος δικτύωσης συγκεντρώνει τον έλεγχο του δικτύου διαχωρίζοντας τον από τις συσκευές του δικτύου.

Όλα τα μοντέλα SDN έχουν κάποια έκδοση ενός **SDN ελεγκτή**, καθώς και **South-Bound** και **North-Bound APIs** :

- **Northbound APIs:** Χρησιμοποιούνται από το SDN δίκτυο για να επικοινωνεί με τις εφαρμογές και την επιχειρηματική λογική. Οι διαχειριστές του δικτύου πρέπει να είναι σε θέση να βοηθούν στη διαμόρφωση προγραμματισμού της κυκλοφορίας και της ανάπτυξης των υπηρεσιών.

- **Southbound APIs:** Χρησιμοποιούνται από το SDN δίκτυο για την αναμετάδοση πληροφοριών στους διακόπτες και δρομολογητές. Το OpenFlow, θεωρείται το πρώτο πρότυπο στο SDN και παραμένει ως ένα από τα πιο κοινά πρωτόκολλα. Παρά την αποτυχημένη προσπάθεια προσαρμογής του OpenFlow και του SDN να είναι το ίδιο, το OpenFlow παραμένει να αποτελεί ένα μεγάλο κομμάτι του SDN.

Ο ελεγκτής SDN υποστηρίζει μια σειρά από οδηγούς που ελέγχουν τη συμπεριφορά των στοιχείων του δικτύου, έτσι ώστε το δίκτυο να παρέχει τις επιθυμητές υπηρεσίες δικτύου. Ο ελεγκτής παρέχει τη λειτουργικότητα του επιπέδου διαχείρισης, όπως η διαχείριση των επιδόσεων και των σφαλμάτων μέσω του SNMP καθώς και άλλα τυποποιημένα πρωτόκολλα. Συνήθως χειρίζεται τη διαχείριση της διάρθρωσης των συμβατών συσκευών OpenFlow προκειμένου να παράσχει την τοπολογία του δικτύου, τη προώθηση, το QoS (Quality Of Service), και τη διαχείριση σύνδεσμο.

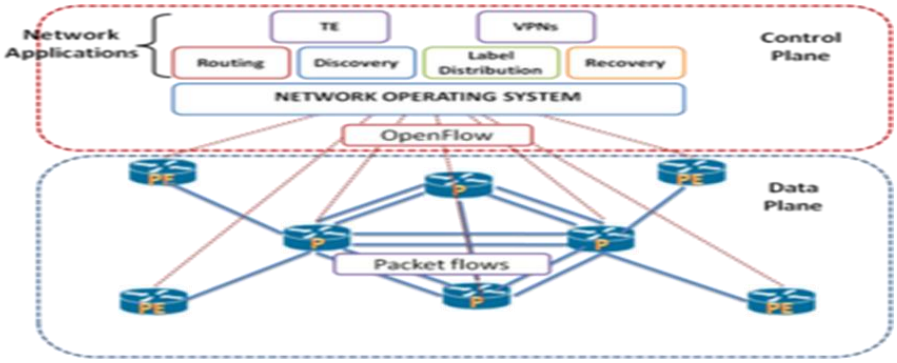
1.4 OpenFlow Πρωτόκολλο

Το **πρωτόκολλο OpenFlow** αναπτύχθηκε στο Πανεπιστήμιο του Στάνφορντ, με την πρώτη του έκδοση 1.0 να δημοσιεύεται στο τέλος του 2009 και την αμέσως επόμενη έκδοσή του 1.1 στις αρχές του 2011. Τον Μάρτιο του 2011, η **Open Networking Foundation(ONF)** δημιουργήθηκε και τα δικαιώματα πνευματικής ιδιοκτησίας του OpenFlow μεταβιβάστηκαν σε αυτή. Η ONF ήθελε να μπορεί να ελέγχει και να εμπορευματοποιήσει το OpenFlow. Με αυτόν τον στόχο κυκλοφόρησε το *OpenFlow v1.3* και, το Μάρτιο του 2012, χορήγησε μια εκδήλωση λειτουργικότητας που ήταν ανοικτή σε όλα τα μέλη της ONF.

Το **OpenFlow** ορίζεται ως ένα ανοικτό πρωτόκολλο επικοινωνίας το οποίο δίνει πρόσβαση στο επίπεδο προώθησης πακέτων δεδομένων σε όλο το δίκτυο. Είναι το μέσο επικοινωνίας μεταξύ ενός διακόπτη ή δρομολογητή με τον κεντρικό ελεγκτή (controller) του δικτύου.

Επειδή τα σύγχρονα δίκτυα κρίνονται πλέον ως απαραίτητα για την υποδομή πολλών επιχειρήσεων, πανεπιστημίων ,ομίλων, σταθμών διαχείρισης και αποθήκευσης μεγάλου όγκου δεδομένων, η τάση να είναι πιο δυναμικά μέσα από προγραμματιστικές εφαρμογές, αυξάνεται συνεχώς. Με το διαχωρισμό που κάνει το OpenFlow πρωτόκολλο μεταξύ του επιπέδου προώθησης πακέτων δεδομένων και του επιπέδου ελέγχου, καθιστά πλέον το δίκτυο πιο δυναμικό, ευέλικτο και ευκολότερο στη διαχείρισή του. Έτσι ο εκάστοτε διακόπτης δεν είναι πλέον αυτόνομος γιατί δεν χρησιμοποιεί το δικό του Λειτουργικό Σύστημα (Λ.Σ.) αλλά ελέγχεται από ένα κεντρικό Λ.Σ. του δικτύου με αποτέλεσμα να ελευθερώνεται από τις συγκεκριμένες δομές που του παρέχει ο κατασκευαστής του και να μπορεί να ενσωματωθεί σε διάφορα δίκτυα που αποτελούνται από διακόπτες διαφορετικών κατασκευαστών.

Στο Σχήμα 3 φαίνεται η αρχιτεκτονική ενός δικτύου το οποίο χρησιμοποιεί το OpenFlow πρωτόκολλο. Η αρχιτεκτονική του δικτύου που απεικονίζεται στο παρακάτω σχήμα αποτελεί την βασική δομή της αρχιτεκτονικής ενός SDN (Software Defined Networking) .

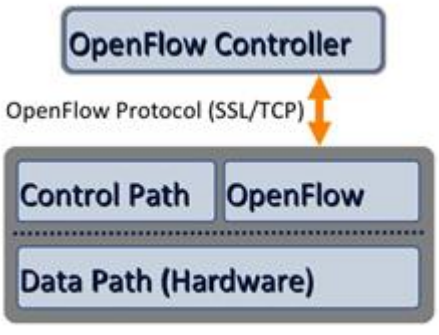


Σχήμα 3: Η βασική δομή της αρχιτεκτονικής ενός SDN χρησιμοποιώντας το OpenFlow Πρωτόκολλο

Όπως βλέπουμε στο παραπάνω σχήμα το OpenFlow πρωτόκολλο είναι ο συνδετικός κρίκος της επικοινωνίας του *επιπέδου προώθησης δεδομένων* (**Data Plane**) με το *επίπεδο ελέγχου* (**Control Plane**). Στο επίπεδο προώθησης χρησιμοποιούμε πλέον για συσκευές μετάδοσης και δρομολόγησης δεδομένων τους *OpenFlow διακόπτες* (**OpenFlow Switches**) και τους *OpenFlow δρομολογητές* (**OpenFlow Routers**), καθώς και στο επίπεδο ελέγχου τον OpenFlow ελεγκτή (**OpenFlow Controller)**.

Το πρωτόκολλο OpenFlow επιτρέπει την απομακρυσμένη διαχείριση των πινάκων προώθηση των πακέτων ενός διακόπτη στο επίπεδο 3, με την προσθήκη, τροποποίηση και την άρση των κανόνων και των δράσεων που ταιριάζουν στα πακέτα. Με αυτό τον τρόπο, οι αποφάσεις δρομολόγησης μπορούν να γίνονται περιοδικά από τον ελεγκτή και να μετατρέπονται σε κανόνες και δράσεις με ρυθμιζόμενη διάρκεια ζωής, τα οποία στη συνέχεια αναπτύσσονται στο πίνακα ροής του διακόπτη. Τα πακέτα που είναι ταιριαστά μεταξύ τους προωθούνται από το διακόπτη με την ταχύτητα του φυσικού τους μέσου ενώ τα πακέτα που είναι αταίριαστα προωθούνται αποκλειστικά από τον ελεγκτή. Ο ελεγκτής μπορεί να αποφασίσει και να τροποποιήσει τους υφιστάμενους κανόνες ενός πίνακα ροής σε έναν ή περισσότερους διακόπτες ή να αναπτύξει νέους, για να αποφευχθεί η διαρθρωτική ροή της κυκλοφορίας μεταξύ του διακόπτη και του ελεγκτή. Θα μπορούσε ακόμη και να αποφασίσει να προωθήσει την ίδια την κυκλοφορία, υπό την προϋπόθεση ότι έχει δώσει εντολή στον διακόπτη να προωθεί ολόκληρα τα πακέτα αντί απλά την επικεφαλίδα τους.

Το πρωτόκολλο OpenFlow επιστρώνεται πάνω από το Transmission Control Protocol (**TCP**), και προβλέπει τη χρησιμοποίηση του Transport Layer Security (**TLS**) καθώς και τον προκάτοχο του το Secure Socket Layer **(SSL**). Ο Ελεγκτής ακούει στη θύρα TCP 6653 για όσους διακόπτες θέλουν να συνδεθούν με αυτόν. [2]

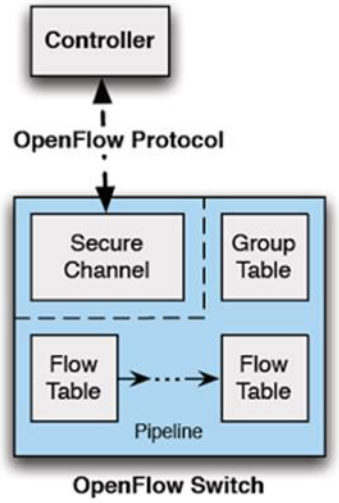


Σχήμα 4 : OpenFlow Protocol over SSL/TCP

1.5 OpenFlow Διακόπτης

Οι περισσότεροι σύγχρονοι διακόπτες και δρομολογητές περιέχουν πίνακες ροής **(flow-tables)**, που εκτελούν τη προώθηση των πακέτων σε έναν εξωτερικό OpenFlow Ελεγκτή. Οι πίνακες-ροής του κάθε προμηθευτή μπορεί να είναι διαφορετικοί, αλλά έχουν προσδιορίσει ένα κοινό σύνολο λειτουργιών που τρέχει σε πολλούς διακόπτες και δρομολογητές, το οποίο εκμεταλλεύεται το OpenFlow για να μπορεί να ενοποιήσει τις συσκευές αυτές ώστε να ελέγχονται απομακρυσμένα από τον ελεγκτή του δικτύου. Ο διακόπτης που επικοινωνεί μέσω του OpenFlow πρωτοκόλλου ονομάζεται *OpenFlow Διακόπτης* **(OpenFlow Switch)**.Οπότε ισχύει ότι , τον OpenFlow Διακόπτη τον διαχειρίζεται ο OpenFlow Ελεγκτής διαμέσου του OpenFlow Πρωτοκόλλου.

Χρησιμοποιώντας αυτό το πρωτόκολλο , ο ελεγκτής μπορεί να προσθέσει , αναβαθμίσει και διαγράψει εγγεγραμμένες ροές (**flow-entries**) του εκάστοτε δικτύου. Γι' αυτό το λόγο το πρωτόκολλο είναι ανοικτό για τον προγραμματισμό των ροών ώστε να είναι δυναμικές και να εντάσσονται ανάλογα με τις ανάγκες του δικτύου.



Σχήμα 5 : Η δομή ενός OpenFlow Switch

Όπως φαίνεται στο Σχήμα5 ένας OpenFlow διακόπτης αποτελείται από το ***Flow-Table***, τον αγωγό σύνδεσης με άλλο Flow-Table δηλαδή το ***Pipeline***, το ***Group-Table*** και το ***Secure Channel***.

- **Flow-Table:** Κάθε *πίνακες ροής* του διακόπτη, περιέχει ένα σύνολο από *εγγεγραμμένων ροών,* όπου η κάθε εγγεγραμμένη ροή αποτελείται από ταιριαστά πεδία (*match-fields)*, μετρητές, και οδηγίες που αποσκοπούν στα ταιριαστά πακέτα (*matching-packets)*. Το ταίριασμα ξεκινάει στο πρώτο *πίνακα ροής* και συνεχίζει στα υπόλοιπα. Οι *εγγεγραμμένες ροές* ταιριάζουν τα πακέτα με σειρά προτεραιότητας με το πρώτο *εγγεγραμμένο ταίριασμα* να χρησιμοποιείται ήδη. Εάν βρεθεί ένα *εγγεγραμμένο ταίριασμα* , τότε οι οδηγίες που έχουν τα *ταιριαστά πακέτα* εκτελούνται αμέσως, αλλιώς , ανάλογα με τις παραμετροποιήσεις του κάθε διακόπτη , τα πακέτα θα προωθηθούν στον ελεγκτή ή θα "πέσουν" ή θα συνεχίσουν στο επόμενο *πίνακα ροής*.

- **Pipeline processing*:*** Οι οδηγίες που σχετίζονται με κάθε *εγγεγραμμένη ροή* περιγράφουν την προώθηση των πακέτων, την επεξεργασία των *ομαδοποιημένων πινάκων (group tables)* και την επεξεργασία του **pipeline**. Ηεπεξεργασία του *pipeline* επιτρέπει στα πακέτα να αποστέλλονται στους επόμενους *πίνακες ροής* για περισσότερη επεξεργασία κα να επιτρέπουν την μετάδοση της πληροφορίας μεταξύ των *πινάκων ροής* με την μορφή των *metadata*.

- **Group-Table:** Οι *ομαδοποιημένοι πίνακες* περιέχουν ομαδοποιημένες εγγραφές, όπου η κάθε εγγραφή δίνει στη ροή τη ικανότητα να ενεργοποιεί το OpenFlow πρωτόκολλο με τις κατάλληλες μεθόδους προώθησης πακέτων.

- **Secure Channel:** Το *secure channel* είναι το τελευταίο στάδιο όπου θα περάσουν τα πακέτα ώστε να τα παραλάβει ο OpenFlow ελεγκτής. Αποτελεί την διεπαφή του OpenFlow ελεγκτή με τον OpenFlow διακόπτη και είναι κρυπτογραφημένο χρησιμοποιώντας το TLS πρωτόκολλο.

[2]

1.6 OpenFlow Ελεγκτής

Ένας *OpenFlow Ελεγκτής* (**OpenFlow Controller**) είναι ένα είδος SDN ελεγκτή που χρησιμοποιεί το πρωτόκολλο OpenFlow για να συνδέεται και να ρυθμίζει τις συσκευές δικτύου (δρομολογητές, διακόπτες, κλπ.) ώστε να καθορίσει την καλύτερη διαδρομή για την κυκλοφορία δεδομένων ενός δικτύου. Ο *OpenFlow Ελεγκτής* προσπαθεί να συγκεντρώσει πλήρως τις αποφάσεις του πακέτου προώθησης, να διαχωρίζει τη λήψη αποφάσεων και με παραδοσιακά πρωτόκολλα δρομολόγησης να εκτελέσει και να επιτρέψει στις εφαρμογές να τροποποιήσουν τις αποφάσεις δρομολόγησης.

Οι SDN Ελεγκτές μπορούν να απλοποιήσουν τη διαχείριση του δικτύου, το χειρισμό όλων των επικοινωνιών μεταξύ των εφαρμογών και των συσκευών για μία αποτελεσματική διαχείριση και μπορούν να τροποποιήσουν τις ροές του δικτύου ώστε να ανταποκρίνεται στις μεταβαλλόμενες ανάγκες. Ειδικότερα, ο *OpenFlow Ελεγκτής* μπορεί να δημιουργήσει ένα κεντρικό σημείο ελέγχου για να επιβλέψει μια ποικιλία από ενεργά στοιχεία του δικτύου.

Περισσότερες λεπτομέρειες όσο αφορά τον *OpenFlow Ελεγκτής* αναφέρονται σε επόμενα κεφάλαια, μέσω του *OpenDaylight Ελεγκτή*. Περιγράφονται, αναλυτικά, αρκετές από τις λειτουργίες του μέσω βασικών του εφαρμογών καθώς και αναλυτικά όλα τα στοιχεία που τον συνθέτουν ώστε να μπορεί να προσφέρει σε ένα SDN τη πλήρη λειτουργία και αποδοτικότητα του.

Κεφάλαιο 2
SDN και OpenDaylight Ελεγκτής

2.1 SDN/OpenFlow Ελεγκτές

Όπως αναφέρθηκε κατά την πρώτη περιγραφή του **OpenFlow Ελεγκτή**, οι SDN Ελεγκτές σε ένα SDN είναι οι "εγκέφαλοι" του δικτύου. Μπορούν να απλοποιήσουν τη διαχείριση του δικτύου, το χειρισμό όλων των επικοινωνιών μεταξύ των εφαρμογών και συσκευών για μία αποτελεσματική διαχείριση και να τροποποιούν τις ροές του δικτύου ώστε να ανταποκρίνεται στις μεταβαλλόμενες ανάγκες. Όταν το επίπεδο ελέγχου του δικτύου υλοποιείται σε λογισμικό οι διαχειριστές μπορούν να διαχειριστούν την κυκλοφορία του δικτύου πιο δυναμικά και σε πιο λεπτομερές επίπεδο.

Οι SDN Ελεγκτές περιέχουν μια συλλογή από «pluggable» ενότητες που μπορούν να εκτελέσουν διάφορες εργασίες του δικτύου. Μερικά από τα βασικά καθήκοντα αυτών είναι η καταγραφή δεδομένων στις συσκευές, καθώς και οι δυνατότητες που μπορούν να ενισχύουν την λειτουργικότητα και την υποστήριξη του δικτύου με περισσότερο προηγμένες δυνατότητες, όπως οι αλγόριθμοι εύρεσης στατιστικών ανάλυσης και ενορχήστρωσης στους νέους κανόνες του δικτύου.

Το πιο σημαντικό όμως είναι πως αποτελούν την εφαρμογή που λειτουργεί ως στρατηγικό σημείο ελέγχου στο SDN, και τη διαχείριση του ελέγχου της ροής δεδομένων στους διακόπτες / δρομολογητές την εφαρμόζουν μέσω των ***South-Bound APIs*** και τις εφαρμογές που τρέχουν στο πάνω μέρος της αρχιτεκτονικής του τις εκτελούν μέσω των ***North-Bound APIs*** και έτσι επιτυγχάνουν την ανάπτυξη ευφυών δικτύων.

2.2 North-Bounds APIs

Τα North-Bound APIs χρησιμοποιούνται για την επικοινωνία μεταξύ του SDN ελεγκτή και τις υπηρεσίες-εφαρμογές που εκτελούνται μέσω του δικτύου. Τα North-Bound APIs μπορούν να χρησιμοποιηθούν για μία πιο αποτελεσματική ενορχήστρωση και αυτοματοποίηση του δικτύου, ώστε να ευθυγραμμιστεί με τις ανάγκες των διαφορετικών εφαρμογών μέσω του SDN προγραμματισμού του δικτύου.

Τα North-Bound APIs είναι αναμφισβήτητα τα πιο κρίσιμα APIs στο περιβάλλον SDN, δεδομένου ότι η αξία των SDN συνδέεται με τις καινοτόμες εφαρμογές που μπορεί ενδεχομένως να υποστηριχθούν και να ενεργοποιηθούν. Τα North-Bound APIs μπορούν να υποστηρίξουν μεγάλο εύρος εφαρμογών, με αποτέλεσμα να μην ταιριάζουν με όλες τις διασυνδέσεις του δικτύου. Αυτός είναι πιθανώς ο λόγος που τα North-Bound APIs υπάρχουν σε διαφορετικές θέσεις πάνω σε μία στοίβα διασυνδέσεων ώστε να μπορούν να ελέγξουν τους διαφορετικούς τύπους εφαρμογών μέσω ενός SDN ελεγκτή.

Τα North-Bound APIs χρησιμοποιούνται επίσης για την ενσωμάτωση του ελεγκτή SDN με στοίβες αυτοματισμού, SaltStack, Ansible και CFEngine, καθώς και πλατφόρμες ενορχήστρωση, όπως OpenStack, vCloud Director VMware ή την ανοικτή CloudStack πηγή. Ο στόχος είναι να μπορούν οι διαχειριστές, όσο αφορά την εσωτερική λειτουργία του δικτύου, να χρησιμοποιούν τις προγραμματιστικές εφαρμογές, οι οποίες ενσωματώνονται στο δίκτυο,

και να κάνουν όποιες αλλαγές θέλουν ανάλογα με τις ανάγκες της εφαρμογής, χωρίς να χρειάζεται να επηρεάζουν το δίκτυο.

Στην Open Networking Foundation **(ONF**) έχουν δώσει μεγάλη προσοχή στην ανάπτυξη των North-Bound APIs. Ασχολούνται συνεχώς με την ανάπτυξη κώδικα και πρωτοτύπων και εξετάζουν κατά πόσον πρέπει ή όχι να δημιουργηθούν πρότυπα διεπαφής και να διευκρινίζονται με σαφήνεια γύρω από το τι είναι και τι μπορούν να κάνουν.

2.3 South-bounds APIs

Τα South-Bound APIs χρησιμοποιούνται για την επικοινωνία μεταξύ του SDN ελεγκτή και των δικτυακών συσκευών, δηλαδή τους διακόπτες και τους δρομολογητές του δικτύου. Μπορούν να είναι είτε ανοικτά προς όλους είτε αποκλειστικά. Διευκολύνουν τον αποτελεσματικό έλεγχο του δικτύου με την ενεργοποίηση του SDN Ελεγκτή να κάνει δυναμικά τις αλλαγές ανάλογα με το πραγματικό χρόνο των απαιτήσεων και των αναγκών του δικτύου. Το OpenFlow είναι το πρώτο και ίσως το πιο γνωστό South-Bound API. Καθορίζει τον τρόπο αλληλεπίδρασης του ελεγκτή SDN με το επίπεδο προώθησης δεδομένων για να γίνουν προσαρμογές στο δίκτυο, έτσι ώστε να μπορούν να προσαρμοστούν καλύτερα στις μεταβαλλόμενες απαιτήσεις των επιχειρήσεων. Με το OpenFlow, εγγραφές μπορούν να προστεθούν και να αφαιρεθούν στην εσωτερική ροή του πίνακα των διακοπτών και, ενδεχομένως, οι δρομολογητές μπορούν να κάνουν το δίκτυο να ανταποκρίνεται περισσότερο στις απαιτήσεις της κυκλοφορίας σε πραγματικό χρόνο. Εκτός του OpenFlow, το Cisco OpFlex είναι επίσης ένα πολύ γνωστό South-Bound API.

Παρόλο που το OpenFlow είναι το πιο γνωστό από τα πρωτόκολλα SDN για τα South-Bound APIs, δεν είναι το μόνο διαθέσιμο στην ανάπτυξη. Υπάρχουν επίσης και άλλα πρωτόκολλα SDN που ένας ελεγκτής μπορεί να χρησιμοποιήσει, όπως το OpFlex της Cisco, το OVSDB, το Yang και το NetConf. Καθιερωμένα πρωτόκολλα δικτύωσης αναπτύσσονται με τέτοιους τρόπους ώστε να μπορούν να προσαρμοστούν και να τρέξουν σε ένα περιβάλλον SDN. Η Task Force Internet Engineering (**IETF**), για να μπορεί να γίνει εφικτή η διασύνδεση με το σύστημα δρομολόγησης (i2rs), αναπτύσσει ένα πρότυπο SDN που επιτρέπει σε έναν ελεγκτή SDN να αποδειχθεί παραδοσιακά πρωτόκολλα, όπως το OSPF, MPLS, BGP, και IS-IS

Υπάρχουν πολλοί προμηθευτές δικτυακού εξοπλισμού που έχουν ανακοινώσει την υποστήριξή τους για το OpenFlow, συμπεριλαμβανομένης της Cisco, Juniper, Extreme Networks, IBM, και άλλων μεγάλων εταιριών.

2.4 OpenDaylight Ελεγκτής (ODL)

Ο **OpenDaylight (ODL) Ελεγκτής** είναι μια υποδομή ελέγχου πολλαπλών πρωτοκόλλων υψηλής διαθεσιμότητας, η οποία χτίστηκε για την ανάπτυξη σύγχρονων ετερογενών δικτύων που χρησιμοποιούν το *Software Defined Networking* (SDN). Χρησιμοποιεί ανοικτά πρωτόκολλα για την παροχή κεντρικού, προγραμματιστικού ελέγχου καθώς και τη παρακολούθηση του δικτύου.

Φιλοξενείται από το **Ίδρυμα Linux**, και είναι ένα ανοιχτού κώδικα (open-source) SDN έργο που αποσκοπεί στην ενίσχυση των SDN, προσφέροντας μια κοινότητα υπό την ηγεσία και τη βιομηχανία που υποστηρίζει τον *ODL ελεγκτή*, η οποία έχει μετονομαστεί σε *OpenDaylight Πλατφόρμα* . Είναι ανοιχτή σε όλους, συμπεριλαμβανομένων των τελικών χρηστών και των πελατών, και παρέχει μια κοινή πλατφόρμα για εκείνους που έχουν στόχο να εργαστούν μαζί για να βρούμε νέες λύσεις. Σύμφωνα με το Ίδρυμα Linux, ο ODL περιλαμβάνει υποστήριξη για το πρωτόκολλο OpenFlow, αλλά μπορεί επίσης να υποστηρίξει και άλλα SDN πρότυπα.

Όπως κάθε SDN ελεγκτής έτσι κ αυτός χρησιμοποιεί ανοιχτά *North-Bound APIs*, τα οποία χρησιμοποιούνται από τις εφαρμογές αυτές οι οποίες χρησιμοποιούν τον *ODL* για τη συλλογή πληροφοριών σχετικά με το δίκτυο, τη διεξαγωγή *Στατιστικών Αναλύσεων*, καθώς και για τη δημιουργία νέων κανόνων σε όλο το δίκτυο.

Ο ODL υλοποιείται αποκλειστικά ως λογισμικό, και διατηρείται στο δικό του Java Virtual Machine **(JVM)** περιβάλλον. Αυτό σημαίνει ότι μπορεί να αναπτυχθεί σε πλατφόρμες όπου το λειτουργικό σύστημα τους, επί των πλείστων Linux, να μπορεί να υποστηρίζει το JVM Για καλύτερα αποτελέσματα, προτείνεται ότι ο ελεγκτής OpenDaylight χρησιμοποιεί μια πρόσφατη διανομή Linux και τουλάχιστον Java Virtual Machine 1.7

Ο ODL έχει στηριχθεί, ως βασική του ανάπτυξη, στο περιβάλλον του Open Service Gateway initiative **(OSGi)**, που είναι μία πλατφόρμα γραμμένη σε γλώσσα προγραμματισμού Java και διατηρείται σε JVM. Παρέχει τον **Apache Karaf** και τον χρησιμοποιεί ως ένα *OSGi bundle*. Μέσα στο περιβάλλον του Apache Karaf μπορούν πλέον να αναπτυχθούν όλες οι εφαρμογές και τα εξαρτήματα που χρειάζεται ο ODL για να ολοκληρώσει το έργο του ως ένας SND ελεγκτής.

2.5 Open Service Gateway initiative (OSGi)

Η *Open Services Gateway initiative* είναι μια ανοιχτή οργάνωση προτύπων που ιδρύθηκε το 1999 και από τότε συνεχίζει να διατηρεί το πρότυπο OSGi. Είναι ένα Java πλαίσιο (framework) το οποίο χρειάζεται για την ανάπτυξη προγραμμάτων λογισμικού και βιβλιοθηκών.

Το πλαίσιο αυτό χωρίζεται στις ακόλουθες περιοχές :

- *Bundles*
  Είναι μια ομάδα *Java κλάσεων και jars* που αποτελούν το αρχείο δήλωσης Manifest.mf σε όλα τα περιεχόμενα του, καθώς και πρόσθετες υπηρεσίες που προσφέρουν εξελιγμένες συμπεριφορές.

- *Services*
  Καθορίζονται από μία *διεπαφή Java* και είναι πακέτα τα οποία μπορούν να εφαρμοστούν.

- *Services Registry*
  Είναι η διεπαφή προγραμματισμού εφαρμογών για την παροχή υπηρεσιών διαχείρισης (Service Registration, Service Tracker και Service Reference).

- *Life-cycle*
  Η διεπαφή προγραμματισμού εφαρμογών για τη διαχείριση του κύκλου ζωής (εγκατάσταση, εκκίνηση, τερματισμός, ενημέρωση, και απεγκατάσταση) όσο αφορά τα bundles.

- Modules
  Αποτελούν τον τρόπο που τα bundles εισάγουν και εξάγουν έναν κώδικα.

- Security
  Είναι το επίπεδο που χειρίζεται τα θέματα ασφαλείας με τον περιορισμό της λειτουργικότητας των bundles σε προκαθορισμένες δυνατότητες.[4]

2.6 Apache Karaf

Ο **Apache Karaf** είναι ένα OSGi Bundle το οποίο αποτελείται από πολλές λειτουργίες (features) που βοηθούν στην ανάπτυξη των περισσοτέρων εφαρμογών και εργαλείων της δομής του ODL ελεγκτή.

Εδώ είναι μια σύντομη λίστα των λειτουργιών που υποστηρίζονται από το Karaf:

- **Ανάπτυξη:** Karaf υποστηρίζει ανάπτυξη των OSGi bundles παρακολουθώντας τα jar αρχεία μέσα από συγκεκριμένο κατάλογο. Κάθε φορά που ένα jar αντιγράφεται σε αυτόν το φάκελο, θα εγκατασταθεί μέσα στο χρόνο εκτέλεσης που του φακέλου. Στη συνέχεια ενημερώνεται ή διαγράφεται καθώς οι αλλαγές αυτές πρέπει να γίνονται αυτόματα.

- **Δυναμική διαμόρφωση:** Οι υπηρεσίες συνήθως ρυθμίζονται μέσω της υπηρεσίας Configuration Admin OSGi. Τέτοια διαμόρφωση μπορεί να οριστεί στον Karaf χρησιμοποιώντας αρχεία ιδιοκτησίας στο εσωτερικό του κατάλογο. Αυτές οι διαμορφώσεις παρακολουθούνται και οι αλλαγές στα αρχεία ιδιοτήτων ενημερώσουν τις υπηρεσίες.

- **Σύστημα καταγραφής:** Χρησιμοποιώντας μια συγκεντρωτική καταγραφή που υποστηρίζεται από log4j, ο Karaf υποστηρίζει μια σειρά από διαφορετικά APIs, όπως JDK και Tomcat.

- **Τροφοδότηση:** Η τροφοδότηση των βιβλιοθηκών ή των αιτήσεων μπορεί να γίνει μέσα από μια σειρά από διαφορετικούς τρόπους, με τους οποίους ο διαχειριστής, μέσω του Karaf, θα κατεβάσει τοπικά, θα εγκαταστήσει και θα θέσει σε λειτουργία.

- **Ενσωμάτωση τοπικού λειτουργικού συστήματος:** Μέσω του Karaf ο διαχειριστής μπορεί να ενσωματώσει το δικό του λειτουργικό σύστημα ως υπηρεσία, έτσι ώστε η διάρκεια ζωής να δεσμεύεται στο λειτουργικό του σύστημα.

- **Επεκτάσιμη κονσόλα:** Ο Karaf διαθέτει ένα περιβάλλον που τρέχει στη κονσόλα οποιουδήποτέ λειτουργικού συστήματος Linux , όπου ο διαχειριστής διαχειρίζεται τις υπηρεσίες, την εγκατάσταση νέων εφαρμογών ή βιβλιοθηκών και την κατάστασή τους. Η ανάπτυξη νέων εντολών γίνεται δυναμικά και, μαζί με τις νέες λειτουργίες ή εφαρμογές στο περιβάλλον αυτό, είναι εύκολα επεκτάσιμες

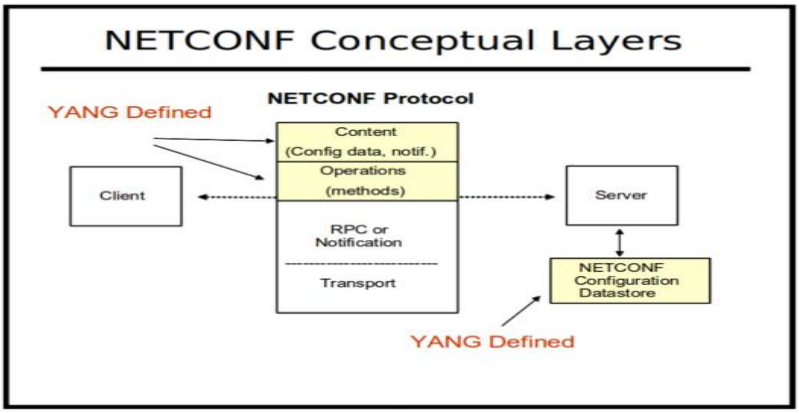
- **Απομακρυσμένη πρόσβαση**: Με οποιοδήποτε SSH client γίνεται η σύνδεση με τον Karaf και εκδίδει εντολές στη κονσόλα.

- **Διαχείριση πολλαπλών περιπτώσεων:** Ο Karaf παρέχει απλές εντολές για τη διαχείριση πολλαπλών περιπτώσεων. Ο διαχειριστής μπορεί εύκολα να δημιουργήσει, να διαγράψει, να ξεκινήσει και να σταματήσει περιπτώσεις Karaf μέσω της κονσόλας. [5]

2.7 Network Configuration Protocol (NetConf)

Το **Network Configuration Protocol (NetConf)** είναι ένα από τα βασικότερα πρωτόκολλα για την διαχείριση του ODL ελεγκτή, το οποίο ενσωματώθηκε από την πρώτη κιόλας έκδοσή του , το *Hydrogen-Base*. Το NetConf είναι πρωτόκολλο διαχείρισης δικτύου και αναπτύχθηκε για πρώτη φορά από την *IETF*. Παρέχει μηχανισμούς εγκατάστασης, χειρισμούς προσθήκης και αφαίρεσης των διαμορφώσεων των δικτυακών συσκευών και χωρίζεται εννοιολογικά σε τέσσερα επίπεδα:

- Το **επίπεδο περιεχομένου**, που αποτελείται από δεδομένα διαμόρφωσης και δεδομένα κοινοποίηση.
- Το **επίπεδο επιχειρήσεων**, που ορίζει ένα σύνολο λειτουργιών πρωτοκόλλου βάσης για να μπορούν οι διαχειριστές να ανακτούν και να επεξεργάζονται τα δεδομένα διαμόρφωσης.
- Το **επίπεδο μηνυμάτων**, που παρέχει ένα μηχανισμό για την κωδικοποίηση κλήσεων απομακρυσμένης διαδικασίας (*Remote Procedure Calls **(RPC))*** και τις κοινοποιήσεις.
- Το **επίπεδο ασφαλής μεταφοράς**, που παρέχει μια ασφαλή και αξιόπιστη μεταφορά μηνυμάτων μεταξύ ενός πελάτη και ενός εξυπηρετητή.



Σχήμα 1:Τα επίπεδα του NetConf πρωτόκολλου σε μια επικοινωνία ενός Client-Server

Το NetConf παρέχει εφαρμογές βασισμένες σε ένα πρότυπο *framework* και σε ένα σύνολο τυποποιημένων *RPCs* για να χειραγωγήσουν τη διαμόρφωση μιας συσκευής δικτύου. Συνήθως μεταφέρεται μέσω του πρωτοκόλλου *SSH*, με τη χρήση του υπό-συστήματος NetConf που είναι παρόμοια με το υπό-σύστημα SFTP, και σχεδιαστικέ για να αντικαταστήσει το *Command Line Interface(CLI)*. Ωστόσο, χρησιμοποιεί δομημένα δεδομένα, και παρέχει λεπτομερείς δομημένες πληροφορίες επιστροφής σφαλμάτων, τα οποία δεν μπορεί να προσφέρει το CLI από μόνο του.

Τα δεδομένα διαμόρφωσης της συσκευής, και το ίδιο το πρωτόκολλο, είναι κωδικοποιημένα με τη γλώσσα σήμανσης, *Extensible Markup Language* **(XML)**. Τα τυποποιημένα εργαλεία της XML όπως *το XML Path Language* (*XPath*) χρησιμοποιούνται για να παρέχουν την ανάκτηση ενός υποσυνόλου δεδομένων διαμόρφωσης. Όλα τα μηνύματα NetConf είναι κωδικοποιημένα σε Namespaces XML.

Το NetConf πρέπει να επιτρέπει στα δεδομένα διαμόρφωσης των συσκευών, να είναι κλειδωμένα ή ξεκλείδωτα, επεξεργάσιμα, και να μπορούν αποθηκεύονται. Επιπλέον, όλες οι τροποποιήσεις στα δεδομένα διαμόρφωσης πρέπει να αποθηκευτούν σε μια μνήμη.

Το πρωτόκολλο κατανέμεται, με βάση μιας δυνατότητας η οποία δίνεται μοναδικά και διαφημίζεται από το διακομιστή όταν ο πελάτης ξεκινά μια συνεδρία NetConf. Μια δυνατότητα μπορεί να αποτελείται από ένα API μεταξύ του διακομιστή και του πελάτη. Αντιπροσωπεύει ένα σύνολο λειτουργιών που δεν μπορούν να μειωθούν με άλλες δυνατότητες.

Τέλος, σημαντικό είναι να σημειωθεί πως υπάρχει ένα βασικό σύνολο ενεργειών που πρέπει πάντα να υποστηρίζεται από το διακομιστή. Για να χρησιμοποιηθεί οποιεσδήποτε πρόσθετη προαιρετική δραστηριότητα, ο πελάτης θα πρέπει να βεβαιωθεί ότι ο διακομιστής υποστηρίζει τη δυνατότητα που συνδέεται με την εν λόγω πράξη.

2.8 Service Abstraction Layer (SAL)

Το **Service Abstraction Layer (SAL)** είναι στην ουσία μια «μπάρα» που συνδέει τα plugins πρωτόκολλα με τις λειτουργίες του SDN, όπως για παράδειγμα την λειτουργία διαχείρισης της τοπολογίας του δικτύου ή την λειτουργία διαχείρισης των διακοπτών Switch κλπ.. Το SAL είναι ανεπτυγμένο σε δύο εκδόσεις. Στην αρχή ξεκίνησε να λειτουργεί με βασικό του οδηγό το RestConf API γι' αυτό και ονομάστηκε **API-Driven Service Abstraction Layer (AD-SAL)** και στη συνέχεια αναπτύχθηκε με βάση ένα μοντέλο οδήγησης των RestConf APIs και πήρε την ονομασία **Model-Driven Service Abstraction Layer (MD-SAL)**.

2.9 API-Driven Service Abstraction Layer (AD-SAL)

Το **API-Driven SAL (AD-SAL)** είναι ένα Service Abstraction Layer που δημιουργήθηκε ειδικά για τον ODL ελεγκτή, για να υπάρχει ένα επίπεδο στο οποίο μπορούν να αναπτυχθούν οι εφαρμογές χωρίς να αναλαμβάνει κάποιο SDN πρωτόκολλο. Η ανάπτυξη του έγινε όταν υπήρξε η ανάγκη να ελέγχει διαφορετικού τύπου στοιχεία, όπως, για παράδειγμα, ποιοι είναι οι γειτονικοί κόμβοι που αποτελούν το γράφημα/τοπολογία του δικτύου ή ποιες διεπαφές είναι συνδεδεμένες σε αυτόν καθώς και ποιες ιδιότητες μεταφέρουν.

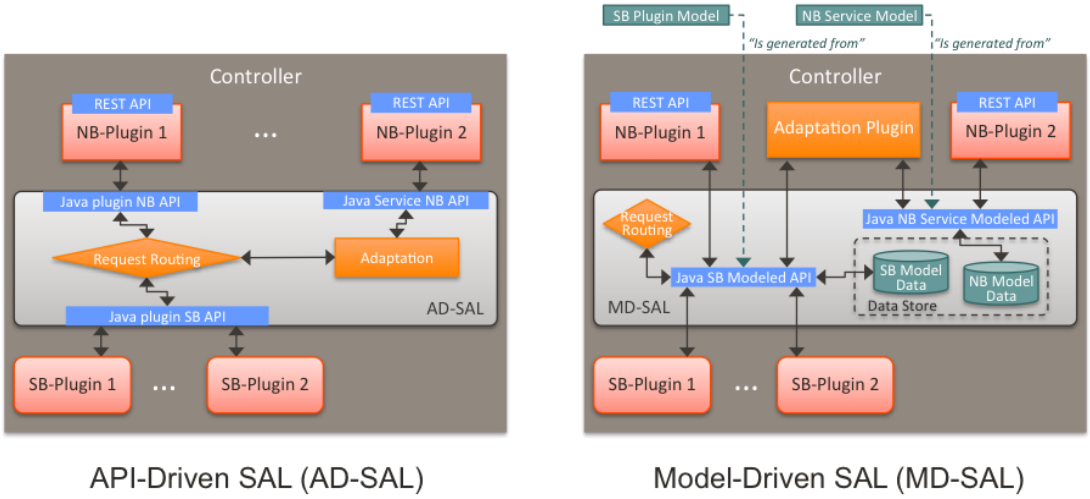
Ανεξάρτητα από την πραγματική λειτουργία ενός δικτυακού πρωτοκόλλου, όταν αυτό συμπεριλαμβάνεται σε ένα SDN, λειτουργεί πλέον ως *SDN plugin-πρωτόκολλο* και περιγράφεται από ένα σύνολο υπηρεσιών που αποτελούνται από ένα σύνολο Java διασυνδέσεων με υποστηριζόμενα αντικείμενα δεδομένων. Τέτοιου είδους Java συμβάσεις μπορεί να είναι ένας *Servlet 3.0* οποίος υλοποιείται μέσω ενός *Tomcat Server ή κάποιον άλλον Servlet.* Όταν μια σύμβαση υλοποιείται απ' όλα τα SDN πρωτόκολλα τότε η ροή του δικτύου είναι φυσιολογική. Όταν, όμως, μία υπηρεσία δεν υλοποιείται από ένα plugin-πρωτόκολλο του συνόλου και δεν μπορεί να επικοινωνήσει με τα υπόλοιπα, τότε χρειάζεται μια εφαρμογή να τρέχει από πάνω, ώστε να μπορέσει να ρυθμίσει τις κατάλληλες ενέργειες για να γίνει η γεφύρωση και να επικοινωνούν όλα τα plugin-πρωτόκολλα μέσω της υπηρεσίας αυτής.

Γενικά η χρησιμότητα του AD-SAL είναι για μπορεί να παρέχει στα plugin-πρωτόκολλα την κατάλληλη υπηρεσία, ώστε αυτά να προσαρμόζονται με τη λειτουργία του SDN έχοντας το καθένα την εφαρμογή που του ταιριάζει. Έτσι τα plugin-πρωτόκολλα μπορούν να είναι είτε πάροχοι δεδομένων είτε καταναλωτές και να συνδέονται μεταξύ τους.

2.10 Από το AD-SAL στο Model-Driven SAL (MD-SAL)

Το πέρασμα στο **Model-Driven SAL (MD-SAL)** δεν έχει στην ουσία μεγάλες αλλαγές από το AD-SAL. Όπως και στο AD-SAL έτσι και στο MD-SAL τα plugin-πρωτόκολλα των παρόχων δεδομένων συνδέονται με τα plugins των καταναλωτών και διευκολύνει την προσαρμογή των δεδομένων μεταξύ τους. Η βασική τους διαφορά είναι πως, ενώ στο AD-SAL τα APIs που χρησιμοποιεί για την δρομολόγηση των δεδομένων μεταξύ παρόχων και καταναλωτών είναι στατικά και ορίζονται κατά την σύνταξή τους , το MD-SAL ορίζει τα API δρομολόγησης του από μοντέλα και οι προσαρμογές δεδομένων γίνονται από «εσωτερικά» plugins προσαρμογής. Ο κώδικας API που παράγεται από τα μοντέλα, φορτώνεται στον ODL ελεγκτή μαζί με το υπόλοιπο plugin που περιέχει το μοντέλο, όταν το plugin bundle OSGi φορτώνεται στον ελεγκτή.

Το AD-SAL και το MD-SAL φαίνονται δίπλα δίπλα στο παρακάτω Σχήμα2:



Σχήμα 2 : Οι διαφορές της αρχιτεκτονικής μεταξύ του AD-SAL και του MD-SAL

Το AD-SAL παρέχει ένα αίτημα δρομολόγησης, δηλαδή επιλέγει ένα South-Bound (SB) plugin-πρωτόκολλο ανάλογα με τον τύπο υπηρεσίας, και προαιρετικά, παρέχει την προσαρμογή των υπηρεσιών, εάν μια North-Bound (NB) υπηρεσία API είναι διαφορετική από το αντίστοιχο SB API του. Για παράδειγμα, στο παραπάνω σχήμα οι διαδρομές AD-SAL από NB-Plugin 1 έως SB Plugins 1 και 2. Σημειώνεται ότι τα plugin SB και NB APIs σε αυτό το παράδειγμα είναι ουσιαστικά το ίδιο και το αίτημα δρομολόγησης βασίζεται στο ότι το SAL ξέρει σε πια περίπτωση ο κόμβος εξυπηρετείται από τα plugins. Όταν ένα Plugin ζητά μια λειτουργία σε ένα δεδομένο κόμβο, το αίτημα δρομολογείται στο κατάλληλο plugin το οποίο στη συνέχεια δρομολογεί το αίτημα στον κατάλληλο κόμβο. Το AD-SAL μπορεί επίσης να παρέχει αφαιρέσεις υπηρεσιών και προσαρμογές. Για παράδειγμα, στο παραπάνω σχήμα, το NB Plugin 2 χρησιμοποιεί ένα αφηρημένο API για την πρόσβαση στις υπηρεσίες που παρέχονται από τα SB Plugins 1 και 2.

Το MD-SAL παρέχει ένα αίτημα δρομολόγησης και την υποδομή για να στηρίξει την προσαρμογή των υπηρεσιών. Ωστόσο, δεν παρέχει την ίδια προσαρμογή των υπηρεσιών γιατί παρέχεται από τα plugins. Το Plugin προσαρμογής παρέχει στοιχεία για το SAL, και καταναλώνει δεδομένα που αποτελούν το SAL μέσω των APIs που παράγονται από τα μοντέλα. Εκτελεί ουσιαστικά το μοντέλο-σε-μοντέλο τρόπο μετάφρασης ανάμεσα σε δύο

APIs. Η αίτηση δρομολόγησης στο MD-SAL γίνεται σε περιπτώσεις πρωτοκόλλου και σε περιπτώσεις κόμβου, εφόσον ο κόμβος εξάγει τα δεδομένα από το plugin στο SAL και το μοντέλο περιέχει πληροφορίες δρομολόγησης.

Τα απλούστερα MD-SAL APIs, που παράγονται από τα μοντέλα, είναι λειτουργικά και ισοδύναμα με τα AD-SAL APIs. Μοντέλα όπως τα RPCs και Notifications υποστηρίζονται στη γλώσσα μοντελοποίησης Yang , το γνωστό Yang Model. Επιπλέον, το MD-SAL μπορεί να αποθηκεύσει δεδομένα για τα μοντέλα που ορίζονται από τα plugins. Παρόχοι και καταναλωτές μπορούν να ανταλλάξουν δεδομένα μέσω της αποθήκευσης του MD-SAL.

Σημαντικό είναι να σημειωθεί, όπως φαίνεται και στο παραπάνω σχήμα, ότι τόσο τα NB AD-SAL όσο και τα NB MD-SAL plugins παρέχουν **Rest APIs** για τις εφαρμογές του ODL ελεγκτή. Η δομή του AD-SAL έχει ισάξιο αριθμό NB APIs με τα SB APIs χρησιμοποιώντας την λειτουργία αντιστοίχισης 1:1 των NB και SB plugins. Η δομή του MD-SAL όμως διαφέρει γιατί επιτρέπει τόσο στα NB όσο και στα SB plugins να χρησιμοποιούν το ίδιο API που παράγεται από το μοντέλο. Το ένα API μπορεί να παίξει το ρόλο του plugin παρόχου αλλά και του καταναλωτή. Έτσι εξαλείφει την ανάγκη να οριστούν δύο διαφορετικά API και παρέχει τρεις διαφορετικές εφαρμογές ακόμα και σε περιπτώσεις που αντιστοιχίζονται το NB plugin με το SB, 1:1.

[6]

Στις νεότερες εκδόσεις του ODL ελεγκτή, από το Helium και κυρίως από το Lithium, το MD-SAL έχει αντικαταστήσει το AD-SAL σε πολύ μεγάλο βαθμό, τέτοιο ώστε το 2^{ο} να χρειάζεται σε πιο ειδικές περιπτώσεις.

Κεφάλαιο 3

Οι εκδόσεις του OpenDaylight Ελεγκτή

Μια ιστορική αναδρομή για τον OpenDaylight Ελεγκτή είναι σημαντικό να σημειωθεί διότι θα βοηθήσει στη κατανόηση της δημιουργίας , της εξέλιξης και όλη την πορεία της ανάπτυξής του. Έτσι θα γίνει περισσότερο κατανοητή η μελέτη και η επεξήγηση της λειτουργίας της 4η έκδοση του, το **Beryllium** (ODL-BE). Μελετώντας, λοιπόν την σταδιακή πορεία και εξέλιξη του ελεγκτή η εγκατάσταση και ο τρόπος λειτουργίας του μπορεί να φανεί πιο εύκολος και κατανοητός.

Οι εκδόσεις του ODL Ελεγκτή διακρίνονται στις εξής :

- **Hydrogen (2013-2014)**
- **Helium (2014-2015)**
- **Lithium (2015-2016)**
- **Beryllium (2016 – Σήμερα)**

3.1 Hydrogen

Στις 8 Απριλίου του 2013, η ODF, η οποία αποτελεί μέρος του Ιδρύματος Linux, ανακοίνωσε την πρώτη έκδοση του ODL ελεγκτή , το **Hydrogen** ,το οποίο προσφέρει τρεις διαφορετικές εκδόσεις για τους χρήστες του, τη Base , τη Virtualization και την έκδοση με τον Service Provider. Με το **Hydrogen** δόθηκε η αρχή στους διαχειριστές δικτύων να έχουν μια πρώτη επαφή με το SDN.

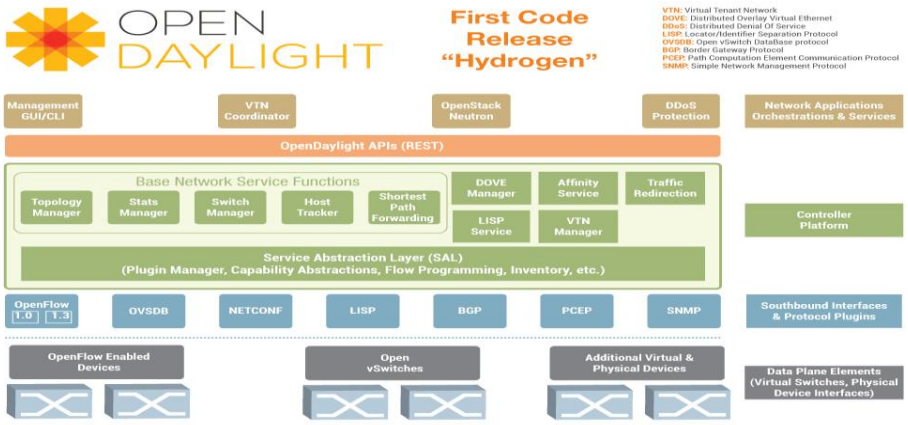
1. Η έκδοση Base, είναι το δομικό στοιχείο για τις άλλες δύο εκδόσεις και έχει σχεδιαστεί για δοκιμές και πειραματικούς σκοπούς. Περιλαμβάνει το OpenFlow 1.3 και την υλοποίηση Open vSwitch για τις βάσης δεδομένων, η οποία επιτρέπει στη διαχείριση των OpenFlow Switches και τις περιπτώσεις του OVSDB δηλαδή ενός Southbound plugin και μίας OpenStack Neutron ενσωμάτωσης.

2. Η έκδοση Virtualization βασίζεται στην έκδοση Base με πρόσθετες τεχνολογίες εικονικοποίησης. Περιλαμβάνει τα εικονικά ενοικιαζόμενα Δίκτυα (Virtual Tenant Networks), τα οποία παρέχουν τη δυνατότητα διαχείρισης των φυσικών διακοπτών μέσω του OpenFlow και την απομόνωση του ενοικιαστή καθώς και το Open DOVE. Διαθέτει επίσης τη Defense4All, και τη εφαρμογή Affinity Metadata Service, καθώς και APIs τα οποία αποτελούν βασικές εφαρμογές του δικτύου.

3. Η έκδοση Service Provider έχει σκοπό να βοηθήσει τους παρόχους υπηρεσιών και φορέων ώστε να αναπτύξουν ένα σχέδιο για να μεταναστεύσουν σε SDN. Περιλαμβάνει, και αυτό, τις Defense4All και Affinity Metadata Service εφαρμογές όπως και την υποστήριξη για μία μηχανική κίνηση χρησιμοποιώντας τα BGP-LS και PCEP πρωτόκολλα. Έχει επίσης την υποστήριξη του πρωτοκόλλου SNMP και APIs για τη διαχείριση του δικτυακού εξοπλισμού.

Για να μπορέσουμε να ξεκινήσουμε το Web User Interface του ODL ελεγκτή (το οποίο αντικαταστάθηκε στις επόμενες εκδόσεις με το DLUX UI), χρησιμοποιούμε την τοπική IP διεύθυνση του οικοδεσπότη (HostIP) και την πόρτα 8080 : *ή* *http://localhost:8080/* [10]

Στο σχήμα 4 φαίνεται χαρακτηριστικά η αρχιτεκτονική που χρησιμοποίησε για πρώτη φορά ο OpenDaylight ελεγκτής με την έκδοση το **Hydrogen**.



Σχήμα 1: Η αρχιτεκτονική του OpenDaylight Controller Hydrogen

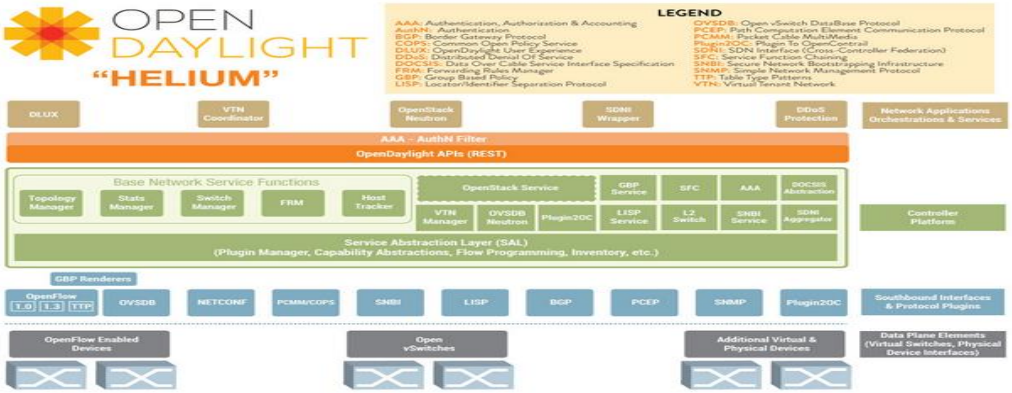
3.2 Helium

Στις 29 Σεπτεμβρίου του 2014, κυκλοφόρησε η δεύτερη έκδοση του OpenDaylight ελεγκτή, το **Helium**, και στη συνέχεια κυκλοφόρησαν οι τέσσερις διαφορετικές υπό-εκδόσεις για τους χρήστες του, οι οποίες είναι η SR1 / SR2 / SR3 / SR4. Το **Helium** ,όπως και **Hydrogen**, είναι της μορφής ανοιχτού κώδικα και βοήθησε στην ανάπτυξη των SDN οποιουδήποτε μεγέθους. Εταιρείες όπως η Cisco και η Brocade έχουν συμμετάσχει στο χώρο ανάπτυξης του OpenDaylight ελεγκτή με βάση το hydrogen Extensible η πρώτη και με βάση το Helium η δεύτερη. Η Cisco μετά την κυκλοφορία του Helium, εντάχθηκε στην OpenDaylight πλατφόρμα οριστικά και συνεχίζει να αναπτύσσεται μέσα από την κοινότητα της έρευνας και ανάπτυξής της, **DevNet.**

Αξίζει να σημειωθεί πώς η δεύτερη έκδοση του ODL περιλαμβάνει 11 νέα πρωτόκολλα, εφαρμογές και τεχνολογίες καθιστώντας τον πιο ευέλικτο και πιο λειτουργικό με αποτέλεσμα οι χρήστες να μπορούν να στραφούν σε έναν ODL ελεγκτή για μία σίγουρη δοκιμή και να τους ανοίξει ο δρόμος για ένα πλέον καινοτόμο και 100% λειτουργικό SDN.

Η βασικότερή βελτίωση που προστέθηκε στο **Helium** είναι το πρωτόκολλο DLUX (OpenDaylight UI) το οποίο έδωσε και το οριστικό πέρασμα από την δοκιμαστική εκδοχή του ελεγκτή στην επίσημη. Στην πρωταρχική έκδοση του DLUX για να μπορέσουμε να το εμφανίσουμε , αφότου πρώτα έχουν γίνει τα απαραίτητα βήματα εγκατάστασης του στον ODL,επισκεπτόμαστε το *http://localhost:8181/dlux/index.html* [11]. Άλλο ένα σημαντικό πρωτόκολλο που έχει προστεθεί είναι το L2Switch καθώς ενσωματώνεται για πρώτη φορά ο Apache Karaf, ο οποίος βελτιώνει κατά πολύ τη λειτουργία και τη διαχείριση του ελεγκτή μέσα από το τερματικό περιβάλλον του λειτουργικού συστήματος Linux από το οποίο τρέχει.

Στο σχήμα 2 φαίνεται χαρακτηριστικά η αρχιτεκτονική του OpenDaylight ελεγκτή με την έκδοση του **Helium.**



Σχήμα 2: Η αρχιτεκτονική του OpenDaylight ελεγκτή Helium

3.4 Lithium

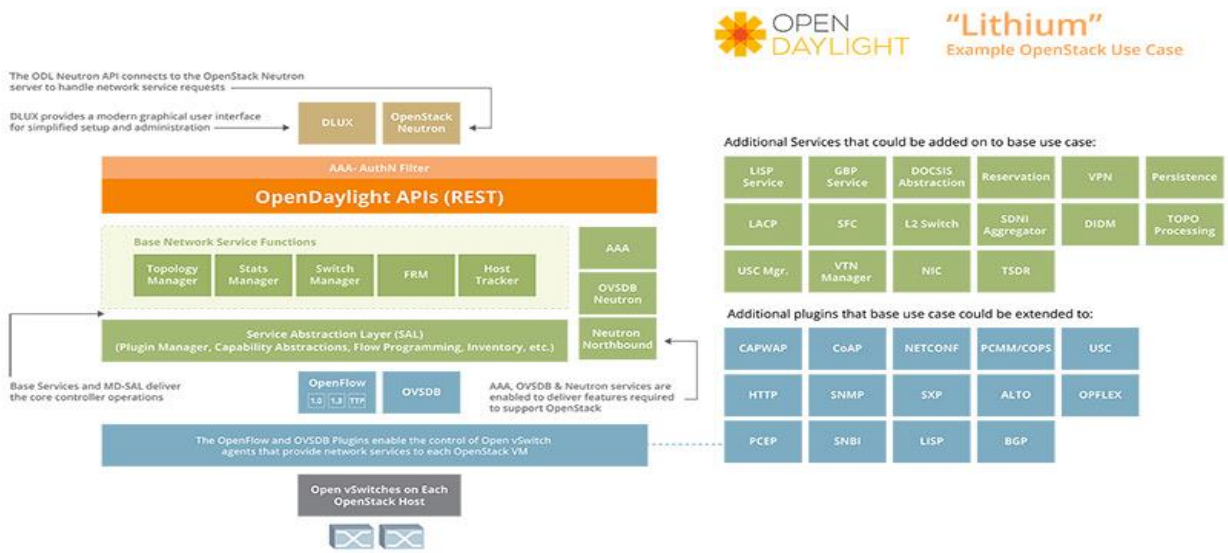
Στις 29 Ιουνίου του 2015 , κυκλοφόρησε η Τρίτη έκδοση του ODL , το **Lithium** ,και στη συνέχεια κυκλοφόρησαν οι τρεις διαφορετικές υπό-εκδόσεις για τους χρήστες του, οι οποίες είναι η SR1 / SR2 / SR3. Με την έκδοση αυτή ο ODL ελεγκτή κατάφερε πλέον να γίνει ένα από τα ταχύτερα αναπτυσσόμενα έργα ανοικτού κώδικα και για την επίτευξή του αυτή συμμετείχαν 466 άτομα. Με το Lithium, οι πάροχοι υπηρεσιών και οι επιχειρήσεις μπορούν πλέον να συνθέσουν τις δικές τους αρχιτεκτονικές σε μία πλατφόρμα η οποία μπορεί να ενσωματώσει τις υπηρεσίες ενός δικτύου, δυναμικά, σε ένα περιβάλλον *Cloud Computing* και *Network Function Virtualization (NFV).*

Με το Lithium εμφανίζεται για πρώτη φορά μία «συνεργασία» μεταξύ του OpenStack και του ODL ελεγκτή για την βελτίωση της επεξεργασία του δικτύου με την παροχή βελτιωμένων υπηρεσιών. Η βασική περίπτωση χρήσης του μπορεί να υλοποιηθεί με την έναρξη του πυρήνα του ελεγκτή ODL, επιτρέποντας την εφαρμογή υπηρεσιών όπως AAA, νετρονίων και OVSDB. Οι υπηρεσίες αυτές μπορούν να υποστηριχθούν από το OpenStack, από ένα εικονικό αυτοματοποιημένο δίκτυο, την παροχή κεντρικού ελέγχου και τη διαχείριση των κατανεμημένων εικονικών διακοπτών και δρομολογητών σε ένα μεγάλου μεγέθους cloud-OpenStack.

Οι βασικές υπηρεσίες και εφαρμογές έχουν πλέον ως βάση ένα *Java Virtual Machine* περιβάλλον με τη βοήθεια της χρήσης του πλαισίου *OSGi* και του *Karaf*. Αυτό επιτρέπει στον ODL να τρέξει ένα σύνολο βασικών υπηρεσιών ώστε να απαιτείται μόνο πρόσθετος κώδικας για την ενεργοποίηση λειτουργιών και μόνο όταν χρησιμοποιούνται. Έτσι καταφέρνει να μπορεί να περιστρέφεται πάνω και κάτω, χωρίς επανεκκινήσεις και χωρίς να διακινδυνεύσει τη σταθερότητα του συστήματος. Οι πρόσθετες λειτουργίες μπορούν να θεωρηθούν ως *add-ons* όπου ο διαχειριστής μπορεί απλά να ενεργοποιεί και να ρυθμίζει τις παραμέτρους για την παροχή πρόσθετων δυνατοτήτων προστιθέμενης αξίας. Για παράδειγμα, ένας διαχειριστής μπορεί να επιθυμεί να εφαρμόσει πρόσθετη ασφάλεια και μπορεί να προσθέσει τα *Unified Secure* χαρακτηριστικά καναλιού για την κρυπτογράφηση των επικοινωνιών, ή να ενεργοποιήσει την πολιτική ομάδα *tagging*, που βασίζεται σε ασφαλείς υπηρεσίες, για να αυτοματοποιήσει την επιβολή πολιτικής σε ολόκληρη την υποδομή. Εάν μια περίπτωση χρήσης απαιτεί εικονικό *firewall*, εξισορρόπηση φορτίου ή άλλων επιπέδου εφαρμογών-υπηρεσιών, οι διαχειριστές μπορούν να ενεργοποιήσουν τη λειτουργία αλυσιδωτής υπηρεσίας για να ενωθούν οι αλυσίδες με πολύ μεγαλύτερη ευκολία, ευελιξία και δυνατότητες από ό,τι με μηχανική κίνηση.

Η συνεχής δημιουργία νέων σχεδίων που επιτρέπουν την ανάπτυξη πάρα πολλών χαρακτηριστικών, η συστοιχία των υπηρεσιών με τα πρωτόκολλα που είναι εκτεταμένη, τα νέα έργα και οι νέες δυνατότητες προστίθενται συνεχώς με κάθε νέα έκδοση του ODL ελεγκτή.

Στο σχήμα 6 φαίνεται χαρακτηριστικά η αρχιτεκτονική του OpenDaylight ελεγκτή της 3ης έκδοση του, **Lithium**.



Σχήμα 3: Η αρχιτεκτονική του OpenDaylight Controller Lithium

3.5 Beryllium

Στις 22 Φλεβάρη του 2016 , κυκλοφόρησε η τέταρτη έκδοση του ODL, το **Beryllium (Be)**, και στην συνέχεια κυκλοφορήσαν οι τέσσερις διαφορετικές υπό-εκδόσεις για τους χρήστες του, οι οποίες είναι η SR1 / SR2 / SR3 / SR4. Μαζί με την νέα έκδοση του ODL κυκλοφόρησε το **NeXt Toolkit.** Το NeXt παρέχει μια κεντρική User Interface (UI) τοπολογία δικτύου με υψηλή απόδοση και πλούσια λειτουργικότητα. Χαρακτηριστικά αυτής τη εργασίας είναι να εμφανίζει ένα μεγάλο συγκρότημα τοπολογιών δικτύου, συγκεντρωτικούς κόμβους του δικτύου, απεικονίσεις, σήραγγες, ομάδα κυκλοφορίας, διαδρομές και να περιλαμβάνει διαφορετικούς αλγόριθμους διάταξης χάρτη επικάλυψης, και κυρίως να είναι φιλικό προς το χρήστη προκαθορισμένες αλληλεπιδράσεις. Με το toolkit δίνεται πλέον η δυνατότητα στο NeXt να μπορεί να συνεργαστεί με το DLUX πρωτόκολλο ώστε να είναι εφικτό να χτίσουν εφαρμογές για τον ODL.

Ο ***ODL-Be*** υποστηρίζει μια ευρεία σειρά περιπτώσεων χρήσης και θεμελιώνει πλέον τη βάση για τα δίκτυα του μέλλοντος. Οι πάροχοι υπηρεσιών και επιχειρήσεων χρησιμοποιούν τον ODL για την επίλυση των βασικών προκλήσεων των δικτύων τους που σχετίζονται με την αυτοματοποιημένη υπηρεσία βελτιστοποίησης πόρων ενός δικτύου, με υπολογιστικά νέφη (Cloud Computing) και NFV δίκτυα, με την έρευνα καθώς και με την ορατότητα και τον έλεγχο. Ο *ODL-Be* συνεχίζει να ενισχύει την αρχιτεκτονική του βασιζόμενος στο MD-SAL για να επιδώσει σε υψηλότερες κλίμακες και να μπορεί να ενσωματώνει εύκολα νέες εφαρμογές και πρωτόκολλα.

Με τον *ODL-Be* νέες οι υπηρεσίες δικτύου έχουν την δυνατότητα να προσφέρουν clustering και υψηλή διαθεσιμότητα, βελτιωμένο χειρισμό των δεδομένων και μηνυμάτων για τις μεταφορές, ευρεία διαχείριση των στοιχείων του δικτύου, καθώς και ένα νέο GUI. Στον *ODL-Be* υπάρχουν αρκετές νέες εφαρμογές που κάνουν τη μετάβαση σε SDN ακόμα πιο εύκολη.

Επίσης, νέες δυνατότητες προστίθενται όπως, ισχυρότερη ανάλυση και δοκιμές της ομαδοποίησης, μεγαλύτερη απόδοση και επεκτασιμότητα. Περιλαμβάνει όλα τα απαραίτητα στοιχεία για την πλήρη υποστήριξη του OpenStack σε υψηλή διαθεσιμότητα και ομαδοποίηση, με βελτιωμένη υποστήριξη για APIs και χαρακτηριστικά συστατικά νετρονίων. Ο *ODL-Be* επιτρέπει την τοποθέτηση του φόρτου εργασίας σε υπολογιστές με DPDK-επιταχυνόμενους

εικονικούς διακόπτες.

Επίσης περιλαμβάνει ένα ευρύτερο φάσμα συνθέσεων για την πολιτική και την πρόθεση του κάθε ελεγκτή ή πλατφόρμα. Οι τέσσερις μέθοδοι που υποστηρίζονται - NEMO, Application-Layer Traffic Optimization Protocol(ALTO), της Group Based Policy (GBP) και του Network Interface Controller (NIC) , παρέχουν μία μεγάλη ευελιξία για τη διαχείριση και την κατεύθυνση των δικτυακών υπηρεσιών, καθώς και των πρόσθετων πόρων που βασίζονται.

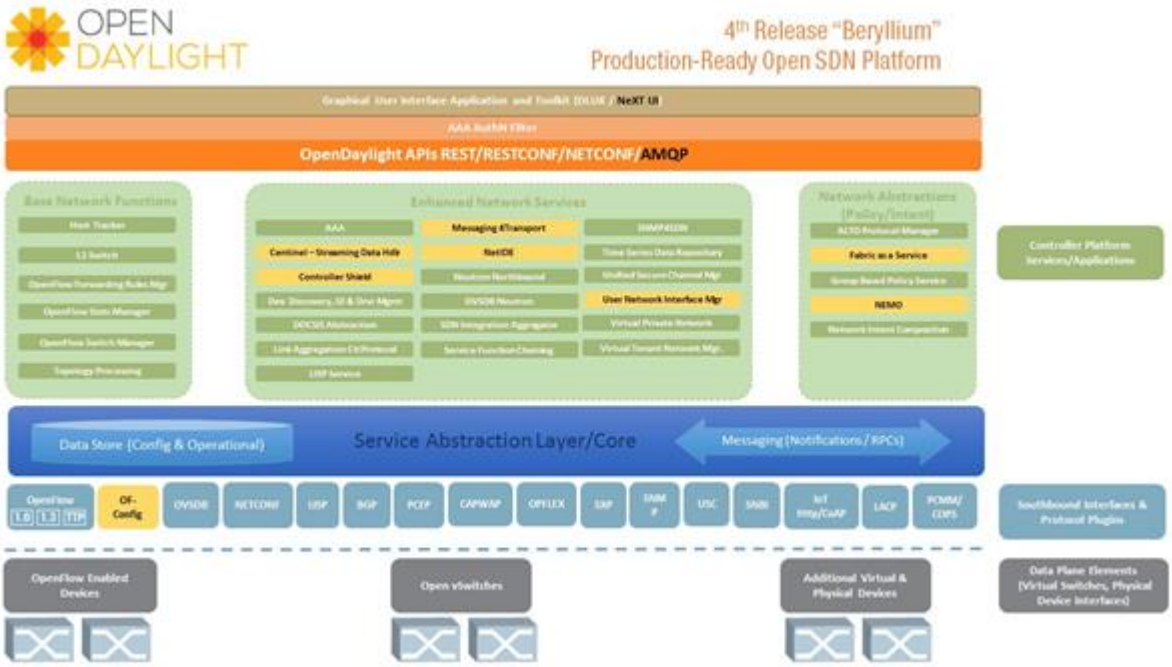
Τέλος παρέχει την ευρύτερη σειρά των περιπτώσεων χρήσης SDN , για το φορέα παροχής υπηρεσιών και επιχειρήσεων. Νέες υπηρεσίες και αρχιτεκτονικές βελτιώσεις στον *ODL-Be* επιτρέπουν νέες περιπτώσεις χρήσης στις περιοχές του Cloud Computing κα των NFV καθώς και την προσθήκη μίας κλίμακας και ευελιξίας, για τις παραδοσιακές περιπτώσεις χρήσης στους τομείς της βελτιστοποίησης των πόρων του δικτύου και την αυτοματοποιημένη παροχή υπηρεσιών.

3.6 Η Αρχιτεκτονική του Beryllium

Η αρχιτεκτονική του *ODL-Be* επιτρέπει στους χρήστες να ελέγχουν τις εφαρμογές, τα πρωτόκολλα και τα plugins, καθώς και παρέχει συνδέσεις μεταξύ των καταναλωτών και των παρόχων. Η ανάπτυξη του *ODL-Be* οδηγείται από μια μεγάλη σε παγκόσμια, πλέον, κοινότητα και συνεχώς προσαρμόζεται στην υποστήριξη ενός ευρύτερου συνόλου του κλάδου των περιπτώσεων χρήσης SDN και NFV.

Ο *ODL-Be* χρησιμοποιεί ένα μοντέλο αρχιτεκτονικής με το οποίο προσπαθεί να περιγράψει το δίκτυο μέσα από τις λειτουργίες που πρέπει να εκτελεστούν σε αυτό και την κατάσταση που έχει επιτύχει. Με την ανταλλαγή των δομών δεδομένων Yang σε ένα κοινό χώρο αποθήκευσης δεδομένων και μηνυμάτων, μπορεί να επιτρέψει σε υπηρεσίες να επιλύουν πολύπλοκα προβλήματα. Με το MD-SAL πρωτόκολλο, οποιαδήποτε εφαρμογή ή λειτουργία μπορεί να συνδυαστεί σε μια υπηρεσία που φορτώνεται στον ελεγκτή. Υπηρεσίες μπορούν να ρυθμιστούν και να προσαρμοστούν ανάλογα με τις ανάγκες του δικτύου.

Στο σχήμα 4 φαίνεται χαρακτηριστικά η αρχιτεκτονική του OpenDaylight ελεγκτή της 4ης έκδοση του, Beryllium.



Σχήμα 3: Η αρχιτεκτονική του OpenDaylight Controller " Beryllium "

Κεφάλαιο 4
RestConf APIs, Yang Model και Yang Tools

4.1 RestConf APIs

Το **RestConf** είναι το πρωτόκολλο που περιγράφεται σε RFC περιβάλλοντα (Restful Core) και παρέχει πρόσβαση διαδικτυακά, μέσω του HTTP δηλαδή, στους δύο τρόπους αποθήκευσης δεδομένων που διαχειρίζεται ο ODL ελεγκτής :

- *config (παραμετροποίηση) και*
- *operational (επιχειρηματικότητα)*

Με τη επιλογή *config* στην ουσία δίνει την δυνατότητα στον διαχειριστή να εισάγει ,διαγράψει και να καλέσει δεδομένα μέσω του ODL ελεγκτή . και να τα παραματρεποιήσει σύμφωνα με τις προδιαγραφές τις οποίες έχει θέσει. Δεν έχουν όλα τα Rest API την δυνατότητα του config παρά μόνο αυτά που είναι εφικτό να πραγματοποιηθεί.

Από την άλλη, με την επιλογή *operational* ο διαχειριστής καλεί δεδομένα που υπάρχουν ήδη στο δίκτυο και έτσι μπορεί να παρακολουθήσει την λειτουργικότητα και την τήρηση των προδιαγραφών του δικτύου και των δικτυακών συσκευών που του έχουν δοθεί.

Το πρωτόκολλο παρέχει ουσιαστικά μια προγραμματική διεπαφή μέσω του HTTP για την πρόσβαση των δεδομένων που έχουν οριστεί στο Yang Model , χρησιμοποιώντας τις αποθηκεύσεις δεδομένων που ορίζονται στο NetConf πρωτόκολλο. Για να μπορέσει να λειτουργήσει σωστά, προτιμήθηκε η ανάπτυξη **Restful APIs** τα οποία αποτελούν τον πιο διαδεδομένο τρόπο ανάπτυξης διαδικτυακών εφαρμογών που χειρίζονται την επικοινωνία μεταξύ ενός *server* και ενός *client.*

Με την ανάπτυξη των Restful APIs δίνεται η δυνατότητα να εφαρμοστούν οι επιλογές που παρέχουν όπως : PUT, GET, POST και DELETE, οι οποίες κρίνονται απαραίτητες για να μπορεί να γίνει σωστά η διαχείριση ενός δικτύου μέσου του ODL ελεγκτή. Έτσι περιγράφονται οι διάφορες λειτουργίες οι οποίες μπορούν να εκτελεστούν μέσω του πρωτοκόλλου HTTP και να φέρουν ως αποτέλεσμα την υλοποίηση εντολών λήψης συγκεκριμένων πληροφοριών ή λειτουργιών αποθήκευσης δεδομένων.

Τα δεδομένα αιτήματος και απόκρισης του πρωτοκόλλου RestConf, μπορούν να εμφανίζονται σε μορφή JSON ή XML σύμφωνα πάντα με τη δομή που θα έχει υποστηριχτεί από τη γλώσσα μοντελοποίησης Yang, δηλαδή Yang-JSON ή Yang-XML.

Το RestConf χρησιμοποιεί ένα URI για να λειτουργήσει μέσω του HTTP και ακούει στην πόρτα :8181.Το URI έχει ως βασικά στοιχεία τη IP διεύθυνση όπου φιλοξενείται ο ελεγκτής και στη συνέχεια /restconf/config ή /restconf/operational αναλόγως τον τρόπο αποθήκευσης δεδομένων που θα χρησιμοποιείται *π.χ.: http://<hostIP>:8181/restconf/config/<identifier>* , όπου ως <identifier> χαρακτηρίζονται τα στοιχεία των δεδομένων των οποίων καλεί ο διαχειριστής μέσω των Rest API. Το URI αυτό εμφανίζεται είτε μία νέα καρτέλα ενός browser είτε μέσα στο DLUX UI του ODL-ελεγκτή. Υπάρχει βέβαια και η δυνατότητα της εμφάνισης των δεδομένων μέσα από τον API browser που καλείται αφότου έχει εκτελεστεί μια από τις λειτουργίες του ODL-ελεγκτή, το apidoc : *http://<hostIP>:8181/apidoc/explorer/index.html .[11]*

4.2 Yang Model

Η γρήγορη ανάπτυξη του πρωτοκόλλου **NetConf** έθεσε ως προτεραιότητα να οριστεί μια γλώσσα μοντελοποίησης δεδομένων για τη συμπλήρωση του. Μοντελοποιμένες γλώσσες, όπως τα SMI (SNMP), UML, XML Schema, και άλλες υπάρχουν ήδη. Ωστόσο, καμία από αυτές τις γλώσσες δεν μπορεί να διαχειριστεί τις ανάγκες του *NetConf*. Δεν είναι εύκολο να διαβαστούν και να κατανοηθούν από τους ανθρώπου που τις υλοποιούν, και έτσι υπολείπονταν στην παροχή μηχανισμών για την επικύρωση μοντέλων δεδομένων διαμόρφωσης όσο αφορά τη σημασιολογία και τη σύνταξη τους.

Τον Οκτώβριο του 2010, η γλώσσα μοντελοποίησης **Yang (Yang Model)** έδωσε τη λύση στο πρόβλημα ανάγνωσης και κατανόησης των γλωσσών μοντελοποίησης. Αναπτύχθηκε από τη ομάδα netMod της IETF, δημοσιεύτηκε ως RFC6020 και θεωρείται πλέον ως η καταλληλότερη γλώσσα μοντελοποίησης δεδομένων για τον ορισμό και την παραμετροποίηση των δεδομένων που αποστέλλονται από το πρωτόκολλο **NetConf.**

Ως γλώσσα μοντελοποίησης δεδομένων μπορεί να χρησιμοποιηθεί για να διαμορφώσει δύο δεδομένα διαμόρφωσης καθώς και την κατάσταση δεδομένων των στοιχείων του δικτύου. Επιπλέον μπορεί να χρησιμοποιηθεί για να προσδιορίσει τη μορφή των κοινοποιήσεων που εκπέμπονται από τα στοιχεία του δικτύου και επιτρέπει στα *μοντέλα-δεδομένα* να καθορίσουν την υπογραφή της απομακρυσμένης κλήσης που επικαλείται τα στοιχεία του δικτύου μέσω του πρωτοκόλλου NetConf. Στη συνέχεια μπορεί να μετατραπεί σε οποιαδήποτε μορφή κωδικοποίησης, π.χ. XML ή JSON, εφόσον το πρωτόκολλο διαμόρφωσης δικτύου υποστηρίζει.

Η Yang αντιπροσωπεύει δομές δεδομένων σε μορφή δέντρου XML. Η γλώσσα μοντελοποίησης δεδομένων έρχεται με μια σειρά από ενσωματωμένους τύπους δεδομένων όπου πρόσθετες εφαρμογές συγκεκριμένων τύπων δεδομένων μπορούν να προέλθουν σε αυτή. Χρησιμοποιεί εκφράσεις XPath για να καθορίσει τους περιορισμούς σχετικά με τα στοιχεία ενός μοντέλου δεδομένων Yang.

4.3 Yang tools

Για να μπορέσει όμως η γλώσσα Yang να εφαρμοστεί και να υποστηριχτεί από εφαρμογές που είναι βασισμένες σε γλώσσα προγραμματισμού Java (JVM-Based), αναπτύχθηκε το **Yang Tools,** που είναι ένα έργο υποδομής το οποίο στοχεύει στην ανάπτυξη των απαραίτητων εργαλείων και βιβλιοθηκών που υποστηρίζουν το πρωτόκολλο NetConf και την γλώσσα Yang. Με το Yang tools έχει επιτευχθεί το *η αντιστοίχιση* του *Yang* κώδικα σε κώδικα *Java*. Τα περιεχόμενά του σχετίζονται με αντικείμενα όπως, δεσμευτικές προδιαγραφές για την Java, αναλυτή για τη γλώσσα Yang, σημασιολογικά μοντέλα και Maven plugins για την επεξεργασία αρχείων της Yang.

Το Yang Tools παίζει σημαντικό ρόλο στον ανάπτυξη εφαρμογών για τον ODL ελεγκτή καθώς οι προγραμματιστές αυτών των εφαρμογών το χρησιμοποιούν για να ομαδοποιήσουν τα Yang μοντέλα σε συνεργασία πάντα με το πρωτόκολλο MD-SAL και με τα Rest APIs που εκτελούνται στην κορυφή ως Northbound μέσω της βοήθειας του RestConf πρωτοκόλλου.

Με τον *ODL-Be* πολλά προβλήματα που υπήρχαν, όσο αφορά το Yang Tools , έχουν επιδιορθωθεί. Στο κομμάτι της παραγωγής κώδικα του MD-SAL, έχουν σημειωθεί αρκετές επιτυχίες και εξελίξεις οι οποίες θα συνεχίσουν μειώνοντας τη σύζευξη μεταξύ του Yang καθορισμένου χρόνου εκτέλεσης και τις πραγματικές μορφές των μηνυμάτων που χρησιμοποιούνται στο ελεγκτή. Έχει ξαναγραφεί εντελώς το Yang parser και ο Yang compiler

με μεγαλύτερη ορθότητα και επεκτασιμότητα, αλλά επίσης και η καλύτερη αναφορά σφαλμάτων μαζί με την ικανότητα να αναλύει ταYIN αρχεία.

Το Yang Data Tree επιβάλλει πλέον υποχρεωτικά στοιχεία, την κατάλληλη περίπτωση αποκλεισμού και τη καταμέτρηση στοιχείων στις λίστες. Επιπλέον, τώρα δημιουργεί και απομακρύνει τα *bundles* αυτόματα, οδηγώντας σε καλύτερη ευθυγράμμιση με τη σημασιολογία RFC6020. Η εφαρμογή της λειτουργίας της συγχώνευσης έχει αναδιατυπωθεί ώστε να είναι πολύ πιο αποδοτική σε CPU και μνήμη. Οι κωδικοποιητές έχουν λάβει σημαντική προσοχή, υποστηρίζοντας το Yang πρότυπο δεδομένων στους κόμβους και τη βελτίωση της αποδοτικότητας της μνήμης, όταν συναντούν τους κόμβους.

4.4 Αντιστοίχιση του κώδικα Yang με τον κώδικα Java

Ένα τρόπος για να κατανοήσουμε την αντιστοίχιση του κώδικα *Yang* με τον κώδικα *Java* είναι η εφαρμογή ***network-topology***, η οποία είναι μία από τις σημαντικότερες εφαρμογές για την διαχείριση ενός δικτύου σε SDN περιβάλλον. Η εφαρμογή αυτή περιγράφεται, σαν τρόπο λειτουργίας της, σε παρακάτω κεφάλαιο όπου και αναφέρεται στον τρόπο διαχείρισης μίας δικτυακής τοπολογία δια μέσου το ODL-Be ελεγκτή.

Το μοντέλο Yang επιτρέπει στην εφαρμογή να έχει μια ολιστική άποψη της τοπολογίας ενός ολόκληρου δικτύου και βασίζεται σε ένα μοντέλο πληροφοριών για κάθε τοπολογία δικτύου. Για να συλλάβει τις πληροφορίες που είναι ειδικά για ένα συγκεκριμένο είδος τοπολογίας, το βασικό μοντέλο είναι έτσι σχεδιασμένο ώστε να μπορεί να αυξηθεί και να προσαρμοστεί σε πολλές τοπολογίες δικτύου.

Συγκεκριμένα, το μοντέλο αυτό ορίζει μια τοπολογία του δικτύου η οποία αποτελείται από μοντέλα-στοιχεία, όπως οι κόμβοι και οι ακμές , καθώς και τα σημεία τερματισμού που περιέχονται στους κόμβους και τερματίζουν πραγματικά τις άκρες τους στην γραφική παράσταση. Ένα δίκτυο μπορεί να περιέχει πολλές τοπολογίες, για παράδειγμα τοπολογίες σε διαφορετικά στρώματα και τοπολογίες επικάλυψης. Το μοντέλο ως εκ τούτου, μπορεί να συλλάβει τη σχέση μεταξύ των τοπολογιών, καθώς και τις εξαρτήσεις μεταξύ των κόμβων και των τερματικών σε όλη τοπολογία.

1. Τα βασικά στοιχεία που αποτελούν την Yang φαίνονται παρακάτω:

Module: Η ενότητα Yang μετατρέπεται σε Java ως δύο Java classes. Η κάθε κλάση είναι σε ξεχωριστό αρχείο Java.

Name of the Package: Το όνομα του πακέτου είναι της μορφής:

org.opendaylight.yang.gen.v1.urn:2:case#module.rev201379 .Με την αντικατάσταση των ψηφίων και των λέξεων κλειδιών το πακέτο μετατρέπεται στην παρακάτω μορφή: *org.opendaylight.yang.gen.v1.urn._2._case.module.rev201379*

Το κάθε πακέτο περιέχει τα παρακάτω στοιχεία:

Opendaylight Prefix :

Κάθε όνομα του πακέτου ξεκινά με το πρόθεμα org.opendaylight.yang.gen.v που είναι ενσωματωμένοο στον BindingGeneratorUtil.moduleNamespaceToPackageName ().

Yang version:

Καθορίζει την έκδοση Yang. YANG έκδοση ενημερώνεται μέσω της μονάδας yang-έκδοση.

Namespace:

Καθορίζει την αξία της ενότητας καθώς και την ονομασία της. Οι χαρακτήρες ονομάτων είναι: /: - @ $ # "* +,? =. Ομάδα χαρακτήρα: / αντικαθίστανται με τελείες (.).

Revision:

Καθορίζει την ημερομηνία αναθεώρησης και συντάσσεται χωρίς τα αρχικά μηδενικά πριν από το μήνα και την ημέρα. Για παράδειγμα: rev201379

παράδειγμα από :

Yang	Java

module network-topology {
yang-version 1;
namespace
"urn:TBD:params:xml:ns:yang:network-topology";
// replace with IANA namespace when assigned
prefix "nt";
import ietf-inet-types { prefix "inet"; }
organization "TBD";
contact "WILL-BE-DEFINED-LATER";
description
"This module defines a model for the topology of a network….
";
revision 2013-10-21 {
description
"Initial revision.";
}
	package org.opendaylight.yang.gen.v1. urn:TBD:params:xml:ns:yang:network-topology.rev201379;
	
	public interface NetworkTopology {
	
	}

Typedef: είναι η δήλωση που αντιστοιχίζει την Yang σε JAVA. Το Typedef περιέχει τα ακόλουθα υπό στοιχεία:

Υπό στοιχεία	**Αντιστοίχιση σε JAVA**
type	class attribute
descripton	δεν αντιστοιχίζεται
units	δεν αντιστοιχίζεται
default	δεν αντιστοιχίζεται

παράδειγμα :

Yang	Java

typedef topology-id {
type inet:uri;
description
	public class TopologyId {
	
	private Typedefinet:uri{
	}

"An identifier for a topology.";	}

Container: Αντιστοιχεί σε περιβάλλον Java το οποίο εκτείνεται σε μια διεπαφή αντικειμένου δεδομένων (DataObject), ενισχυμένη από το περιεχομένου (Augmentable <container_interface) και το όνομα που αντιστοιχίζεται με την διεπαφή αυτή.

List: Το Yang στοιχείο της λίστας έχει αντιστοιχιστεί με μια διασύνδεση Java όπου το ανώτερο στοιχείο δημιουργείται ανάλογα με το είδος της Λίστας-επιστροφής των παραγόμενων διεπαφών.

Leaf: Κάθε leaf πρέπει να περιέχει τουλάχιστον έναν τύπο υπο-στοχείου και να αντιστοιχιστεί με τη μέθοδο του ανώτερου στοιχείου όπου ο τύπος απόδοσης να είναι ίσος με την αξία του υπό-στοιχείου.

Leaf-list: Κάθε leaf-list πρέπει να περιέχει τουλάχιστον έναν τύπο υπο-στοχείου και να αντιστοιχιστεί με τη μέθοδο του ανώτερου στοιχείου όπου ο τύπος απόδοσης να είναι ίσος με την αξία του υπό-στοιχείου.

παράδειγμα :

Yang	Java

container network-topology {
list topology {
description "This is the model of an abstract topology…
";
key "topology-id";
leaf topology-id {
type topology-id;
description "
It is presumed that a datastore will contain many topologies…
";
}
	TopologyId.java
	package org.opendaylight.yang.gen.v1.urn:TBD:params:xml:ns:yang:network-topology.rev201379.NetworkTopology.Topology;
	
	import org.opendaylight.yangtools.yang.binding.DataObject;
	import org.opendaylight.yangtools.yang.binding.Augmentable;
	
	public interfaceTopologId extends DataObject, Augmentable<TopologyId>
	{
	
	TopolgyId getITopologyId();
	}
	TopologIdKey.java
	package org.opendaylight.yang.gen.v1.urn:TBD:params:xml:ns:yang:network-topology.rev201379.NetworkTopology;
	
	import org.opendaylight.yang.gen.v1.urn.module.rev201379.cont.TopologyIdKey;
	import java.math.BigInteger;
	import java.util.List;
	
	public class TopologyIdKey {
	
	private BigInteger TopologyIdKey;
	
	public TopologyIdKey(BigInteger TopologIdKey) {
	super();
	this.TopologyIdKey = TopologyIdkey;
	}
	
	public BigInteger getTopologyId() {
	return TopologyId;
	}
	}
	Topology.java
	package org.opendaylight.yang.gen.v1.urn:TBD:params:xml:ns:yang:network-topology.rev201379.NetworkTopology;
	
	import org.opendaylight.yangtools.yang.binding.DataObject;
	import org.opendaylight.yangtools.yang.binding.Augmentable;
	import java.util.List;

	import org.opendaylight.yang.gen.v1.urn:TBD:params:xml:ns:yang:network-topology.rev201379.NetworkTopology;
	outter.list.TopologyId;
	
	public interface Topology extends DataObject, Augmentable<Topology> {
	
	List<TopologyId> getTopologyIf();
	TopologyKey getTopologyKey();
	}
	NetworlTopology.java
	
	package org.opendaylight.yang.gen.v1.urn:TBD:params:xml:ns:yang:network-topology.rev201379;
	
	import org.opendaylight.yangtools.yang.binding.DataObject;
	import org.opendaylight.yangtools.yang.binding.Augmentable;
	import java.util.List;
	import org.opendaylight.yang.gen.v1.urn:TBD:params:xml:ns:yang:network-topology.rev201379.NetworkTopology.Topology;
	
	public interface NetworkTopology extends DataObject, Augmentable<NetworkTopology> {
	
	List<Topology> getTopology(); }

[7] & [9]

2. Το βασικό διάγραμμα του network-topology Yang

Η δομή του μοντέλου δεδομένων network-topology απεικονίζεται στον ακόλουθο διάγραμμα. Οι αγκύλες περικλείουν μία λίστα από κλειδιά, «rw» που έχουν δηλαδή τη δυνατότητα ανάγνωσης και έγγραφής. Το «Ro»: η κατάσταση των δεδομένων επιχειρησιακής και ";" ορίζει προαιρετική κόμβους. Το ποσοστό αυτό δεν απεικονίζουν όλους τους ορισμούς πρόκειται ν απεικονίζουν τη συνολική δομή.

```
module: network-topology
    +--rw network-topology
       +--rw topology [topology-id]
          +--rw topology-id          topology-id
          +--ro server-provided?     boolean
          +--rw topology-types
          +--rw underlay-topology [topology-ref]
          |  +--rw topology-ref    topology-ref
          +--rw node [node-id]
          |  +--rw node-id             node-id
          |  +--rw supporting-node [node-ref]
          |  |  +--rw node-ref    node-ref
          |  +--rw termination-point [tp-id]
          |     +--rw tp-id     tp-id
          |     +--ro tp-ref*   tp-ref
          +--rw link [link-id]
             +--rw link-id           link-id
             +--rw source
             |  +--rw source-node    node-ref
             |  +--rw source-tp?     tp-ref
             +--rw destination
             |  +--rw dest-node    node-ref
             |  +--rw dest-tp?     tp-ref
             +--rw supporting-link [link-ref]
                +--rw link-ref    link-ref
```

[8]

3. Ο κώδικας του network-topology Yang,

```
<CODE BEGINS>
file network-topology@2013-10-21.yang

module network-topology  {
   yang-version 1;
   namespace "urn:TBD:params:xml:ns:yang:network-topology";
   // replace with IANA namespace when assigned
   prefix "nt";

   import ietf-inet-types { prefix "inet"; }

   organization "TBD";

   contact "WILL-BE-DEFINED-LATER";

   description
      "This module defines a model for the topology of a network.
      Key design decisions are as follows:
      A topology consists of a set of nodes and links.
      Links are point-to-point and unidirectional.
      Bidirectional connections need to be represented through
      two separate links.
      Multipoint connections, broadcast domains etc can be represented
      through a hierarchy of nodes, then connecting nodes at
      upper layers of the hierarchy.";

   revision2016-12-24 {
      description
         "Initial revision.";
   }

   typedef topology-id {
      type inet:uri;
      description
         "An identifier for a topology.";
   }

   typedef node-id {
      type inet:uri;
      description
         "An identifier for a node in a topology.
         The identifier may be opaque.
         The identifier SHOULD be chosen such that the same node in a
         real network topology will always be identified through the
         same identifier, even if the model is instantiated in separate
         datastores. An implementation MAY choose to capture semantics
         in the identifier, for example to indicate the type of node
         and/or the type of topology that the node is a part of.";
   }
   typedef link-id {
      type inet:uri;
      description
         "An identifier for a link in a topology.
         The identifier may be opaque.
         The identifier SHOULD be chosen such that the same link in a
         real network topology will always be identified through the
         same identifier, even if the model is instantiated in separate
         datastores. An implementation MAY choose to capture semantics
         in the identifier, for example to indicate the type of link
         and/or the type of topology that the link is a part of.";

   }
   typedef tp-id {
      type inet:uri;
      description
         "An identifier for termination points on a node.
         The identifier may be opaque.
         The identifier SHOULD be chosen such that the same TP in a
         real network topology will always be identified through the
```

```
        same identifier, even if the model is instantiated in separate
        datastores. An implementation MAY choose to capture semantics
        in the identifier, for example to indicate the type of TP
        and/or the type of node and topology that the TP is a part of.";
  }
  typedef tp-ref {
     type leafref {
        path "/network-topology/topology/node/termination-point/tp-id";
     }
     description
        "A type for an absolute reference to a termination point.
        (This type should not be used for relative references.
        In such a case, a relative path should be used instead.)";
  }
  typedef topology-ref {
     type leafref {
        path "/network-topology/topology/topology-id";
     }
     description
        "A type for an absolute reference a topology instance.";
  }

  typedef node-ref {
     type leafref {
        path "/network-topology/topology/node/node-id";
     }
     description
        "A type for an absolute reference to a node instance.
        (This type should not be used for relative references.
        In such a case, a relative path should be used instead.)";
  }
  typedef link-ref {
     type leafref {
        path "/network-topology/topology/link/link-id";
     }
     description
        "A type for an absolute reference a link instance.

        (This type should not be used for relative references.
        In such a case, a relative path should be used instead.)";
  }
  grouping tp-attributes {
     description
        "The data objects needed to define a termination point.
        (This only includes a single leaf at this point, used
        to identify the termination point.)
        Provided in a grouping so that in addition to the datastore,
        the data can also be included in notifications.";
     leaf tp-id {
        type tp-id;
     }
     leaf-list tp-ref {
        type tp-ref;
        config false;
        description
           "The leaf list identifies any termination points that the
           termination point is dependent on, or maps onto.
           Those termination points will themselves be contained
           in a supporting node.
           This dependency information can be inferred from
           the dependencies between links.  For this reason,
           this item is not separately configurable.  Hence no
           corresponding constraint needs to be articulated.
           The corresponding information is simply provided by the
           implementing system.";
      }
  }
  grouping node-attributes {
     description
        "The data objects needed to define a node.
        The objects are provided in a grouping so that in addition to
        the datastore, the data can also be included in notifications
        as needed.";
     leaf node-id {
        type node-id;
```

```
        description
          "The identifier of a node in the topology.
          A node is specific to a topology to which it belongs.";
      }
      list supporting-node {
        description
          "This list defines vertical layering information for nodes.
          It allows to capture for any given node, which node (or nodes)
          in the corresponding underlay topology it maps onto.
          A node can map to zero, one, or more nodes below it;
          accordingly there can be zero, one, or more elements in the list.
          If there are specific layering requirements, for example
          specific to a particular type of topology that only allows
          for certain layering relationships, the choice
          below can be augmented with additional cases.
          A list has been chosen rather than a leaf-list in order
          to provide room for augmentations, e.g. for
          statistics or priorization information associated with
          supporting nodes.";
        key "node-ref";
        leaf node-ref {
          type node-ref;
        }
      }
    }
    grouping link-attributes {
      // This is a grouping, not defined inline with the link definition itself,
      // so it can be included in a notification, if needed
      leaf link-id {
        type link-id;
        description
          "The identifier of a link in the topology.
          A link is specific to a topology to which it belongs.";
      }
      container source {
        leaf source-node {
          mandatory true;
          type node-ref;
          description
            "Source node identifier, must be in same topology.";
        }
        leaf source-tp {
          type tp-ref;
          description
            "Termination point within source node that terminates the link.";
        }
      }
      container destination {
        leaf dest-node {
          mandatory true;
          type node-ref;
          description
            "Destination node identifier, must be in same topology.";
        }
        leaf dest-tp {
          type tp-ref;
          description
            "Termination point within destination node that terminates the link.";
        }
      }
      list supporting-link {
        key "link-ref";
        leaf link-ref {
          type link-ref;
        }

      }
    }
    container network-topology {
      list topology {
        description "This is the model of an abstract topology.
                     A topology contains nodes and links.
                     Each topology MUST be identified by
                     unique topology-id for reason that a network could contain many
                     topologies";
```

```
        key "topology-id";
        leaf topology-id {
          type topology-id;
          description "
            It is presumed that a datastore will contain many topologies. To
            distinguish between topologies it is vital to have UNIQUE
            topology identifiers.
          ";
        }
        leaf server-provided {
          type boolean;
          config false;
          description "
            Indicates whether the topology is configurable by clients,
            or whether it is provided by the server.  This leaf is
            populated by the server implementing the model.
            It is set to false for topologies that are created by a client;
            it is set to true otherwise.  If it is set to true, any
            attempt to edit the topology MUST be rejected.
          ";
        }
        container topology-types {
          description
            "This container is used to identify the type, or types
            (as a topology can support several types simultaneously),
            of the topology.
            Topology types are the subject of several integrity constraints
            that an implementing server can validate in order to
            maintain integrity of the datastore.
            Topology types are indicated through separate data nodes;
            the set of topology types is expected to increase over time.
            To add support for a new topology, an augmenting module
            needs to augment this container with a new empty optional
            container to indicate the new topology type.
            The use of a container allows to indicate a subcategorization
            of topology types.
            The container SHALL NOT be augmented with any data nodes
            that serve a purpose other than identifying a particular
            topology type.
          ";
        }
        list underlay-topology {
          key "topology-ref";
          leaf topology-ref {
            type topology-ref;
          }
          // a list, not a leaf-list, to allow for potential augmentation
          // with properties specific to the underlay topology,
          // such as statistics, preferences, or cost.
          description
            "Identifies the topology, or topologies, that this topology
            is dependent on.";
        }
        list node {
          description "The list of network nodes defined for the topology.";
          key "node-id";
          uses node-attributes;
          must "boolean(../underlay-topology[*]/node[./supporting-nodes/node-ref])";
            // This constraint is meant to ensure that a referenced node is in fact
            // a node in an underlay topology.
          list termination-point {
            description "A termination point can terminate a link.
              Depending on the type of topology, a termination point could,
              for example, refer to a port or an interface.";

            key "tp-id";
            uses tp-attributes;
          }
        }
        list link {
          description "
            A Network Link connects a by Local (Source) node and
            a Remote (Destination) Network Nodes via a set of the
            nodes' termination points.
            As it is possible to have several links between the same
```

```
                source and destination nodes, and as a link could potentially
                be re-homed between termination points, to ensure that we
                would always know to distinguish between links, every link
                is identified by a dedicated link identifier.
                Note that a link models a point-to-point link, not a multipoint
                link.
                Layering dependencies on links in underlay topologies are
                not represented as the layering information of nodes and of
                termination points is sufficient.
              ";
              key "link-id";
              uses link-attributes;
              must "boolean(../underlay-topology/link[./supporting-link]";
                 // Constraint: any supporting link must be part of an underlay topology
              must "boolean(../node[./source/source-node])";
                 // Constraint: A link must have as source a node of the same topology
              must "boolean(../node[./destination/dest-node])";
                 // Constraint: A link must have as source a destination of the same topology
              must "boolean(../node/termination-point[./source/source-tp])";
                 // Constraint: The source termination point must be contained in the source node
              must "boolean(../node/termination-point[./destination/dest-tp])";
                 // Constraint: The destination termination point must be contained
                 // in the destination node
            }
        }
    }
}

<CODE ENDS>
```

[8]

Σημείωση: *Ο Yang κώδικας της εφαρμογής network-topology είναι από την* ***IETF Tools,*** *και το αρχείο draft-clemm-i2rs-yang-network-topo-00.txt.[8] Στα παρακάτω παραδείγματα έχουν τοποθετηθεί μέρη από τον yang model της εφαρμογής και η αντιστοιχία σε Java έγινε από την πλευρά μου, με βοήθεια από το Yang Tools: Yang to Java Mapping[9] στην ιστοσελίδα της εγκυκλοπαίδειας της opendaylight κοινότητας και Inocybe Technologies: Developer's Guide [7].*

Κεφάλαιο 5
Εγκατάσταση του ODL-Be Ελεγκτή

5.1 Προεπισκόπηση

Ο ***ODL Beryllium*** (***ODL-Be***) ελεγκτής είναι λογισμικό βασισμένο σε *JVM* περιβάλλον και μπορεί να εκτελεστεί σε οποιοδήποτε λειτουργικό σύστημα το οποίο υποστηρίζει *Java, κυρίως Linux*. Ο *ODL-Be* είναι, όπως έχει προαναφερθεί, η υλοποίηση ενός SDN και χρησιμοποιεί τα ακόλουθα εργαλεία :

- Maven: χρησιμοποιείται για ευκολότερη κατασκευή αυτοματισμού. Ο Maven χρησιμοποιεί το αρχείο pom.xml (Object Model) για να μπορεί να εκτελέσει τις εξαρτήσεις μεταξύ των bundles καθώς και για να μπορεί να περιγράψει πια από αυτά μπορεί να τα φορτώσει και να τα ξεκινήσει.

- OSGi: Αυτό το πλαίσιο είναι το back-end του *ODL-Be* και επιτρέπει δυναμικά στη φόρτωση των bundles, των αρχείων JAR, και δεσμευτικών bundles, για την ανταλλαγή πληροφορίας.

- Java Interfaces**:** Οι διεπαφές Java χρησιμοποιούνται για την εκδήλωση ακροάσεων, προδιαγραφών, και σχηματισμό μοτίβων. Αυτός είναι ο κύριος τρόπος με τον οποίο συγκεκριμένα bundles εφαρμόζουν λειτουργίες επανάκλησης και επίσης για να δείξει την ευαισθητοποίηση μίας συγκεκριμένης κατάστασης.

- Rest APIs: Είναι τα Northbound APIs όπως, *network-topology* και το *opendaylight-inventory* καθώς τα οποία εφαρμόζουν λειτουργίες όπως, network-management, host-tracking, flow-programming, static-routing και ούτω καθεξής.

Ο *ODL-Be* εκθέτει τα Northbound APIs τα οποία χρησιμοποιούνται από τις εφαρμογές καθώς υποστηρίζουν το OSGi και τα Rest APIs. Το OSGi χρησιμοποιείται κυρίως για εφαρμογές οι οποίες εκτελούνται στο ίδιο διευθυνσιοδοτημένο χώρο με τον ελεγκτή ενώ τα Rest APIs χρησιμοποιούνται από εφαρμογές από εφαρμογές που δεν είναι απαραίτητο να εκτελούνται από το ίδιο διευθυνσιοδοτημένο χώρο (ακόμα και στο ίδιο σύστημα) με τον ελεγκτή.

Όπως θα δούμε και παρακάτω, οι εφαρμογές χρησιμοποιούν αλγόριθμους για να μπορέσουν να συλλέξουν ευφυή δικτυακή πληροφορία ώστε να μπορέσουν να εκμεταλλευτούν τα δεδομένα αυτά για ανάλυση και ομαδοποίηση καθώς και για την ενορχήστρωση νέων κανόνων μέσα στο δίκτυο. Σημαντικό είναι να σημειωθεί πως χαρακτηριστικό αυτής της έκδοσης αυτής είναι η μετανάστευση των code-base και ο επανασχεδιασμός των APIs. Όσο αφορά τα Southbound πρωτόκολλα είναι αρκετά και πολλαπλά αυτά που υποστηρίζονται από τον ελεγκτή, καθώς και αυτό που αποτελεί την βασικότερη υποδομή του ελεγκτή είναι το *OpenFlow 1.3* .

Λόγο της επιτυχής μετανάστευσης συστατικών παραγωγής κώδικα από το έργο *Yang Tools*, τα πρότυπα μηνυμάτων από το έργο του *ODL-Be*, προκειμένου να μειωθεί η σύζευξη μεταξύ της γλώσσας Yang με τα πραγματικά σχέδια του μηνύματος, χρησιμοποιούνται στον ελεγκτή για την εφαρμογή της ομαδοποίηση τους. Σημαντικότερο, τα MD-SAL τηρούνται πλήρως φορητά και απαιτούν μόνο JVM συμβατό με Java 7 SE ή Java SE 8 για να τρέξουν. Ο κώδικας

για τα Yang Maven plugin απαιτεί Maven έκδοση 3.1.1 ή νεότερη έκδοση καιYang Maven Plugin έκδοση 0.8.

[12][13]

Για να μπορέσουν να εφαρμοστούν οι δυνατότητες που μπορεί να προσφέρει ο ελεγκτής, εκτός από τον ίδιο θα πρέπει να εγκατασταθεί το **Mininet** το οποίο είναι ένα εικονικό δίκτυο και μπορεί να προσφέρει τα απαραίτητα δικτυακά στοιχεία και τις δικτυακές συσκευές ώστε να έχουμε μια πλήρη εικόνα των δυνατοτήτων του ελεγκτή.

5.2 L2 Switch

Πέρα από τα τέσσερα βασικά εργαλεία που χρειάζεται ο ODL-Be για να εκτελεστεί, για να μπορέσει να διαχειριστεί μία δικτυακή τοπολογία πρέπει να εγκατασταθεί και η εφαρμογή του **L2 Switch**. Συγκεκριμένα, η L2 Switch εφαρμογή αποτελεί έναν διακόπτη πακέτων στο επίπεδο δικτύου που αποτελεί τη ζεύξη των δεδομένων. Η Karaf ονομασία της είναι ***l2switch-switch*** και είναι μία εφαρμογή βελτιστοποίησης βασισμένη στο MD-SAL, η οποία μαθαίνει τον διακόπτη πως προωθείται το κάθε πακέτο. Ο κύριος ρόλος της είναι να διαβιβάσει τα πλαίσια του επιπέδου της ζεύξης των δεδομένων από το τον κόμβο προώθησης στον τελικό κόμβο προορισμού μέσα σε ένα δίκτυο LAN, κοιτάζοντας τη MAC διεύθυνση του κόμβου προορισμού στην κεφαλίδα του πλαισίου. Διαφορετικά, αναμεταδίδει το μήνυμα σε όλες τις εξωτερικές θύρες του δικτύου. Οι εσωτερικές θύρες αποτελούνται από τις συνδέσεις των διακοπτών μεταξύ τους (switch-to-switch) και τι συνδέσεις μεταξύ ελεγκτή και διακόπτη (controller –to-switch). Οι εξωτερικές θύρες είναι όλες οι υπόλοιπες θύρες οι οποίες συνδέονται με τους χρήστες (hosts). Έτσι διασυνδέει πολλαπλούς τελικούς κόμβους ενός δικτύου LAN και έξυπνα προωθεί την κυκλοφορία μεταξύ τους.

Η L2 Switch κυμαίνεται σε τιμές ταχυτήτων όπως 10Mbps, 100 Mbps, 1 Gbps, 10 Gbps κ.λπ. και υποστηρίζει full-duplex επικοινωνία σε κάθε μία από της θύρες του. Τέλος, διευκολύνει την επέκταση του δικτύου και τη σύνδεση του με το υπόλοιπο δίκτυο μέσω των θυρών υψηλής ταχύτητας, uplink θύρες, και μπορεί να συνδεθεί είτε με άλλους διακόπτες L2 ή με L3 δρομολογητές (Routers).

5.3 Εγκατάσταση του ODL ελεγκτή

Η εγκατάσταση αφορά τον ODL-Be SR3 και παρακάτω αναφέρονται τα απαραίτητα εργαλεία τα οποία έχω εγκαταστήσει ώστε να μπορέσει να τρέξει ο ελεγκτής σωστά :

i) Λειτουργικό Σύστημα: Ubuntu 14.04 LTS

ii) Java Runtime Environment (JRE) v 7 - OpenJDK

iii) Apache Maven v 3.3.3

iv) Hardware: Laptop: i5-4core
8 GB Ram
256 GB SSD

Για να ξεκινήσουμε την εγκατάσταση πρέπει να ακολουθήσουμε τα παρακάτω βήματα:

Βήμα 1ο: Εγκατάσταση JRE 7

Ανοίγουμε ένα τερματικό και εκτελούμε τις παρακάτω εντολές

tonypx@tonypx-Lenovo-G50-70:~$ sudo apt-get install update

tonypx@tonypx-Lenovo-G50-70:~$: sudo apt-get install openjdk-7-jdk

Βήμα 2ο: Εγκατάσταση Apache Maven v 3.3.3

Πρώτα δημιουργούμε έναν φάκελο μέσα στο usr/local του Linux :

tonypx@tonypx-Lenovo-G50-70:~$ sudo mkdir -p /usr/local/apache-maven

Μετά κατεβάζουμε τον Maven

tonypx@tonypx-Lenovo-G50-70:~/Λήψεις$ wget http://ftp.wayne.edu/apache/maven/maven3/3.3.3/binaries/apache-maven-3.3.3-bin.tar.gz

Και τώρα είμαστε έτοιμοι για την εγκατάστασή του εκτελώντας τις παρακάτω εντολές :

tonypx@tonypx-Lenovo-G50-70:~$ sudo mkdir /usr/local/apache-maven

tonypx@tonypx-Lenovo-G50-70:~/Λήψεις$ sudo mv apache-maven-3.3.3-bin.tar.gz /usr/local/apache- maven

tonypx@tonypx-Lenovo-G50-70:~$ sudo tar -xzvf /usr/local/apache-maven/apache-maven-3.3.3- bin.tar.gz -C /usr/local/apache-maven/

tonypx@tonypx-Lenovo-G50-70:~$ sudo update-alternatives --install /usr/bin/mvn mvn /usr/local/apache-maven/apache-maven-3.3.3/bin/mvn 1

tonypx@tonypx-Lenovo-G50-70:~$ sudo update-alternatives --config mvn

tonypx@tonypx-Lenovo-G50-70:~$ sudo apt-get install vim

tonypx@tonypx-Lenovo-G50-70:~$ vim ~/.bashrc

Εξάγουμε τα παρακάτω στοιχεία :

tonypx@tonypx-Lenovo-G50-70:~$ export M2_HOME=/usr/local/apache-maven/apache-maven-3.3.3

tonypx@tonypx-Lenovo-G50-70:~$ export MAVEN_OPTS="-Xms256m -Xmx512m" # Very important to put the "m" on the end

tonypx@tonypx-Lenovo-G50-70:~$ export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64 # this matches sudo update-alternatives --config java

Βήμα 3ο: Εγκατάσταση Beryllium-SR3 *(ODL-Be)*

i) Επισκεπτόμαστε την σελίδα από την οποία μπορούμε να κατεβάσουμε την τελευταία έκδοση του OpenDaylight, και πιο συγκεκριμένα την έκδοση του Beryllium-SR3 http://www.opendaylight.org/software/downloads

ii) Εξάγουμε το συμπιεσμένο αρχείο

tonypx@tonypx-Lenovo-G50-70:~/Λήψεις$ tar -xzvf distribution-karaf-0.4.3-Beryllium-SR3.tar.gz

iii) Τρέχουμε τον Karaf

Προαιρετικά!
Τοποθετούμε τον φάκελο του Beryllium-SR3 στην τοποθεσία που μας βολεύει πχ /home/tonypx/ODLs/Beryllium/distribution-karaf-0.4.3-Beryllium-SR3 και εισερχόμαστε στο παρακάτω φάκελο

tonypx@tonypx-Lenovo-G50-70:~$ cd ODLs/Beryllium/distribution-karaf-0.4.3-Beryllium-SR3/bin

Εγώ το έκανα γιατί ο φάκελος ODLs περιέχει όλες τις εκδόσεις του OpenDaylight, από το Hydrogen μέχρι τον Beryllium SR3.

Εκτελούμε : . /karaf –of13 (Το –of13 για να σιγουρέψουμε πως

tonypx@tonypx-Lenovo-G50-70:~/ODLs/Beryllium/distribution-karaf-0.4.3-Beryllium-SR3/bin$ sudo . /karaf -of13

εικόνα 3.2.1. Εκκίνηση ODL-Be

Βήμα 4ο: Εγκατάσταση βασικών λειτουργιών (features) του ODL-Be

α) αφότου εισέλθουμε στον ODL-Be για να μπορεί να λειτουργήσει και να δομηθεί σωστά θα πρέπει να εγκαταστήσουμε τις παρακάτω βασικές εφαρμογές :

i) *odl-restconf: επιτρέπει τη πρόσβαση στα RestConf APIs.*

ii) odl-mdsal-apidocs: επιτρέπει την πρόσβαση στα Yang APIs

iii) odl-dlux-all: για την ενεργοποίηση του δικτυακού User Interface του ελεγκτή

iv) odl-l2switch-switch: Παρέχει τη λειτουργικότητα του δικτύου παρόμοια με ένα διακόπτη Ethernet

v) odl-adsal-northbound: Περιέχει τα northbound plugins

β) Για να εγκαταστήσουμε οποιαδήποτε εφαρμογή χρησιμοποιούμε την παρακάτω εντολή : *opendaylight-user@root>feature:install*

Άρα για τις 5 βασικές εφαρμογές έχουμε :

opendaylight-user@root>feature:install odl-restconf odl-mdsal-apidocs odl-dlux-all odl-l2switch-switch odl-adsal-northbound

γ)Για να μπορέσουμε να δούμε όλε τις διαθέσιμες εφαρμογές εκτελούμε την παρακάτω εντολή:

opendaylight-user@root>feature:list

Για να μπορέσουμε να δούμε τις εφαρμογές που είναι εγκατεστημένες εκτελούμε την παρακάτω εντολή:

opendaylight-user@root>feature:list--installed

και μας εμφανίζει όλες τις εγκαταστημένες λειτουργίες του ODL

Άμα θέλουμε να δούμε συγκεκριμένες εφαρμογές πχ του ODL τότε εκτελούμε :

opendaylight-user@root>feature:list |grep odl

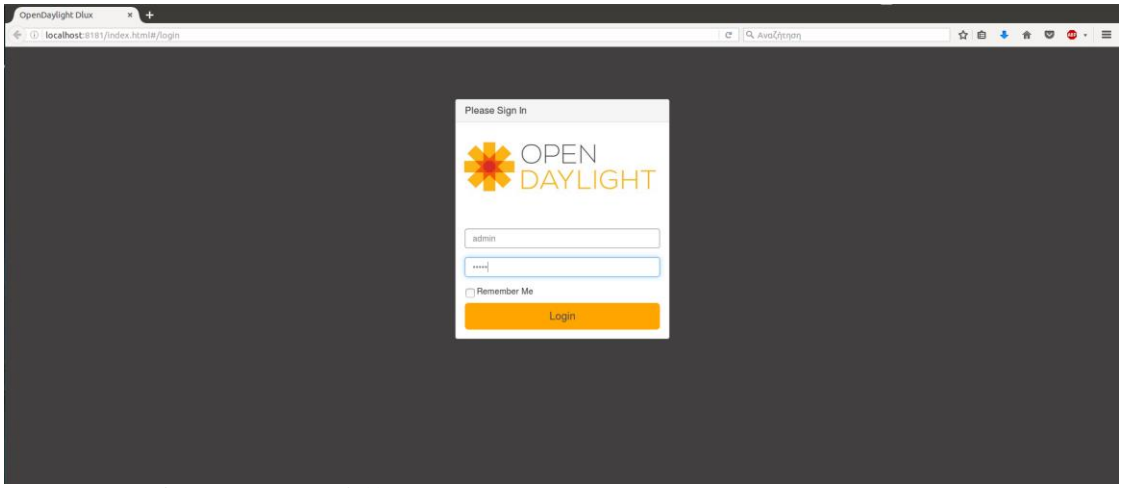
εικόνα 3.2.2. Λίστα εγκατεστημένων λειτουργιών του ODL-Be

Βήμα 5ο: Σύνδεση με το Web Interface του ODL-Be

Σε έναν Web Browser της επιλογής μας, μπορούμε να εισέλθουμε στο Web Interface του ODL-Be, που μας προσφέρει η λειτουργία DLUX , στην διεύθυνση :

http://localhost:8181/index.html#/login
Username: admin
Password: admin
[12][13]

Το localhost είναι η τοπική IP διεύθυνση του υπολογιστή σας η οποία θα μπορούσε να γραφτεί κ ως : *127.0.0.1* ή τη IP διεύθυνση που έχει από το τοπικός σας δίκτυο, πχ σε έμενα είναι η *192.168.1.4*



εικόνα 3.2.3. Είσοδος στο γραφικό περιβάλλον του ODL-Be

5.4 Εγκατάσταση του Mininet

Το ***mininet*** είναι ένα εικονικό δίκτυο, ή καλυτέρα ένας εξομοιωτής δικτύου που τρέχει μια συλλογή των τελικών χρηστών (hosts),διακοπτών (switches), δρομολογητών (routers), και συνδέσμους (links) πάνω από έναν ενιαίο πυρήνα του Linux. Χρησιμοποιεί ελαφρύ τύπου εικονοποιήση για να μπορεί να κάνει ένα ενιαίο σύστημα να μοιάζει με ένα πλήρες δίκτυο, που τρέχει τον ίδιο πυρήνα, το ίδιο σύστημα και τον χρήστη. Για την λειτουργία του ***mininet*** μπορούν να χρησιμοποιηθούν δύο περιπτώσεις:

1η Περίπτωση:

Ένα το ***Mininet*** τρέχει πάνω σε ένα εικονικό μηχάνημα (virtual machine) συμπεριφέρεται ακριβώς όπως ένα πραγματικό δίκτυο, δηλαδή, μπορείς σαν διαχειριστής να συνδεθείς με ssh σε αυτό και να το γεφυρώσεις με το δίκτυο του πραγματικού μηχανήματος όπως μπορεί να συμβεί σε οποιοδήποτε VM και να τρέχει αυθαίρετα προγράμματα με την προϋπόθεση να είναι εγκατεστημένα στο σύστημα Linux. Τα προγράμματα που μπορούν να τρέχουν στέλνουν πακέτα μέσα από ένα πραγματικό Ethernet interface, με δεδομένη την ταχύτητα σύνδεσης και την καθυστέρηση. Τα πακέτα επεξεργάζονται από έναν διακόπτη, δρομολογητή ή ακόμη και από ένα ενδιάμεσο «κουτί» με δεδομένη ποσότητα αναμονής. Όταν δύο προγράμματα, όπως ένας πελάτης και ένας server επικοινωνούν μέσω Mininet, η μετρούμενη απόδοση θα πρέπει να ταιριάζει με αυτή των δύο αλλά θα είναι πιο αργή του πραγματικού μηχανήματος.

2η Περίπτωση:

Στην 2η περίπτωση, είναι η εγκατάσταση του **Mininet** σε ένα πραγματικό μηχάνημα το οποίο τρέχει λειτουργικό σύστημα Linux και να χρησιμοποιεί του φυσικούς δικτυακούς πόρους του μηχανήματος. Στη συγκεκριμένη μελέτη η εγκατάσταση έγινε στο ίδιο φυσικό μηχάνημα όπου έχει εγκατασταθεί και ο ελεγκτής για καλύτερη αξιοποίηση των δικτυακών πόρων.

Παρακάτω παρουσιάζονται οι κατάλληλες εντολές για την εγκατάσταση και διαχείριση του Mininet μέσα από το τερματικό (terminal) του Linux:

i) Για την εγκατάσταση ανοίγουμε ένα 2ο τερματικό (διότι στο 1ο τρέχει ο OpenDaylight) και εκτελούμε τις παρακάτω εντολές:

α) Αρχικά της εγκατάστασης-εκκίνησης του Mininet πρέπει να είναι ενεργό το Southbound API το οποίο έχει ήδη εγκατασταθεί στον ODL-Be ελεγκτή, το OpenVSwitch, από την εντολή

feature:install odl-l2switch-switch ώστε να ενεργοποιηθεί το DLUX και ο έλεγχος των RestConf APIs

tonypx@tonypx-Lenovo-G50-70:~/ODLs/Beryllium/distribution-karaf-0.4.3-Beryllium-SR3/bin$ sudo service openvswitch-switch start

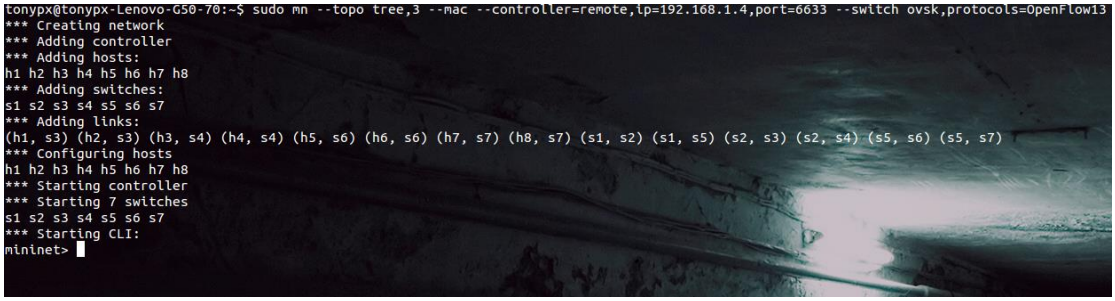
Σημείωση: Σε περίπτωση που δεν υπάρχει ή δεν είναι εγκατεστημένο το openvswitch-switch service, τότε εκτελούμε την παρακάτω εντολή :
[12][13]
tonypx@tonypx-Lenovo-G50-70:~/ODLs/Beryllium/distribution-karaf-0.4.3-Beryllium-SR3/bin$ sudo apt-get install openvswitch-switch

β) Εκτελούμε την παρακάτω εντολή εγκατάστασής του Mininet :

t*onypx@tonypx-Lenovo-G50-70:~ sudo apt-get install mininet*

γ) Μετά την εγκατάστασή ξεκινάμε το *mininet* με οποιαδήποτε τοπολογία θέλουμε

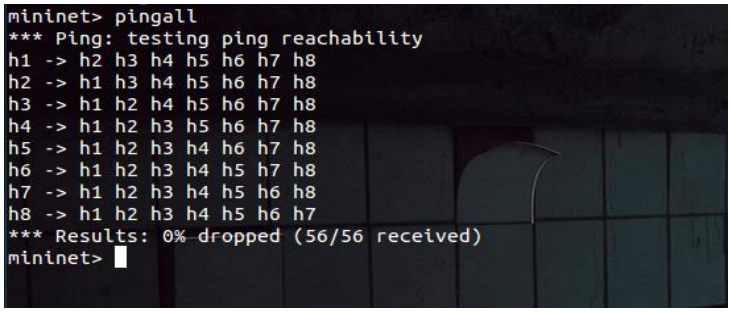
tonypx@tonypx-Lenovo-G50-70:~$ sudo mn --topo tree, 3 --mac --controller=remote,ip=192.168.1.4,port=6633 --switch ovsk,protocols=OpenFlow13



εικόνα 3.3.1. Εκκίνηση του Mininet

ii) Για να ελέγξουμε ότι όλοι οι κόμβοι επικοινωνούν μεταξύ του εκτελούμε την εντολή:

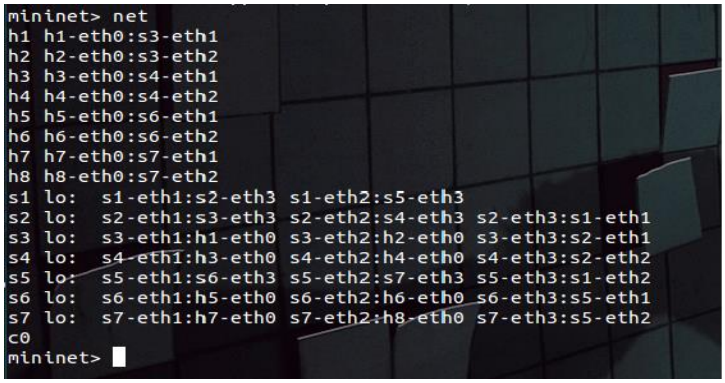
mininet> pingall



εικόνα 3.3.2. Εντολή pingall

iii) Για να μπορούμε να δούμε όλους τους συνδέσμους του δικτύου εκτελούμε την εντολή:

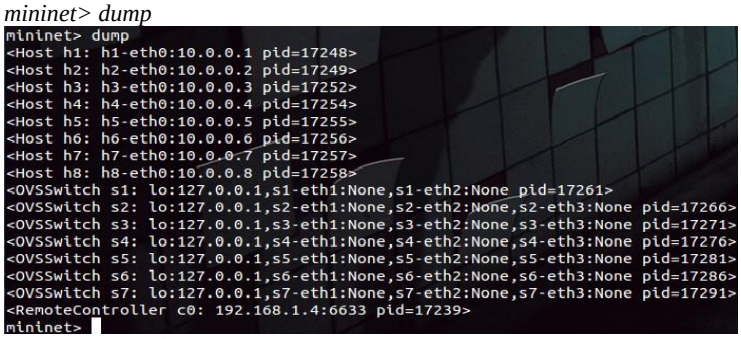
mininet> net



εικόνα 3.3.3. Εντολή net

iii) Για να μπορούμε να δούμε τις IP διευθύνσεις και το pid των χρηστών (hosts), των διακοπτών (switches), των συνδέσμων (links) και του ελεγκτή του δικτύου εκτελούμε την εντολή:

mininet> dump



εικόνα 3.3.4. Εντολή dump

Κεφάλαιο 6

Λειτουργίες και εφαρμογές του ODL-Be Ελεγκτή

6.1 DLUX

Για μία εύχρηστη διαχείριση ενός SDN καθώς και ενός ODL ελεγκτή, χρησιμοποιούμε την εφαρμογή OpenDaylight User Experience (DLUX). Την εφαρμογή αυτή την έχουμε ήδη εγκαταστήσει στον ελεγκτή ODL-Be, και ανοίγοντας έναν Web browser εισάγουμε το παρακάτω σύνδεσμο : http://localhost:8181/index.html#/login.[12][13] Αφότου δώσουμε τα σωστά πιστοποιητικά, εισερχόμαστε στο Web Interface του ελεγκτή , δηλαδή στην εφαρμογή DLUX.

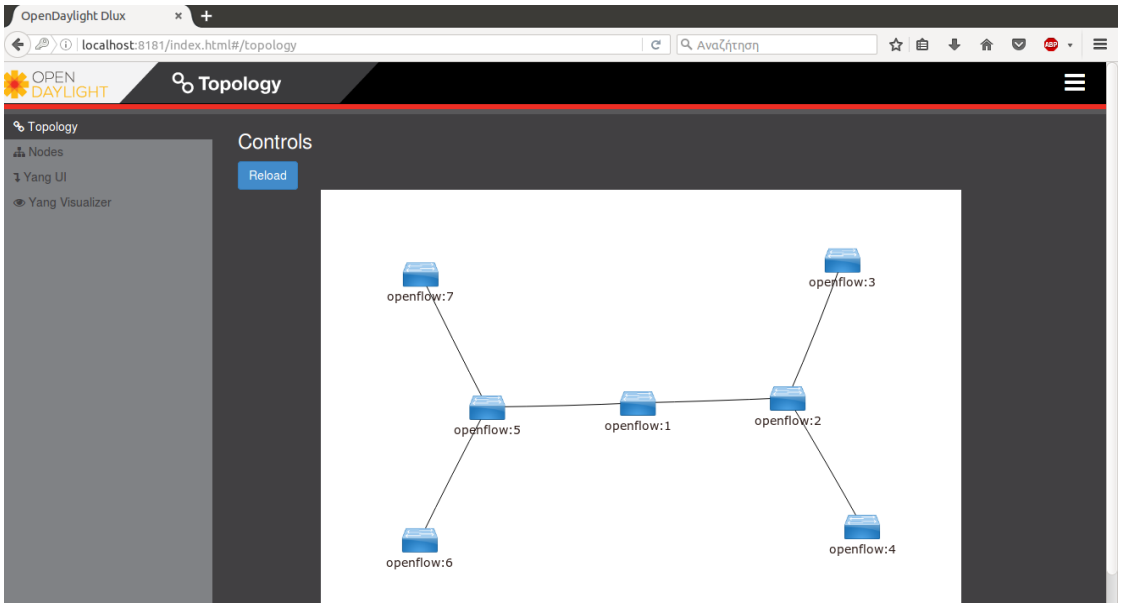
Το DLUX παρέχει ένα αριθμό από διαφορετικές λειτουργίες του Karaf, οι οποίες μπορούν να ενεργοποιηθούν ξεχωριστά Αυτές είναι οι : *odl-dlux-core, odl-dlux-node, odl-dlux-yangui, odl-dlux-yangvisualizer.*

Η βασική λειτουργία διαχείρισης του ODL ελεγκτή είναι η **DLUX (**OpenDaylight User Experience) πλατφόρμα η οποία απεικονίζει το γραφικό περιβάλλον του ελεγκτή και διευκολύνει του διαχειριστές των SDN.

Χαρακτηριστική και σημαντική εφαρμογή της γλώσσας μοντελοποίησης Yang, είναι το **Yang UI (**User Interface) το οποίο είναι μια βασική εφαρμογή για το γραφικό περιβάλλον ,GUI (Graphic User Interface), του ελεγκτή και δίνει πρόσβαση στη διαμόρφωση αλλά και την κατάσταση του δικτύου. Το Yang UI είναι βασισμένο στο ***ODL DLUX UI*** και χρησιμεύει για μια πιο εύχρηστη διαχείριση ενός ODL ελεγκτή.

6.2 Topology

Η βασική προϋπόθεση της καλής λειτουργίας του DLUX είναι η εγκατάσταση των εφαρμογών MD-SAL και η δημιουργία μίας οποιαδήποτε δικτυακής τοπολογίας. Το MD-SAL το έχουμε ήδη εγκαταστήσει καθώς και μέσω του Mininet έχουμε μία ήδη ενεργή δικτυακή τοπολογία. Την τοπολογία μπορούμε να την δούμε επιλέγοντας την ενότητα «*Topology*» ,όπως φαίνεται και στην εικόνα 1.



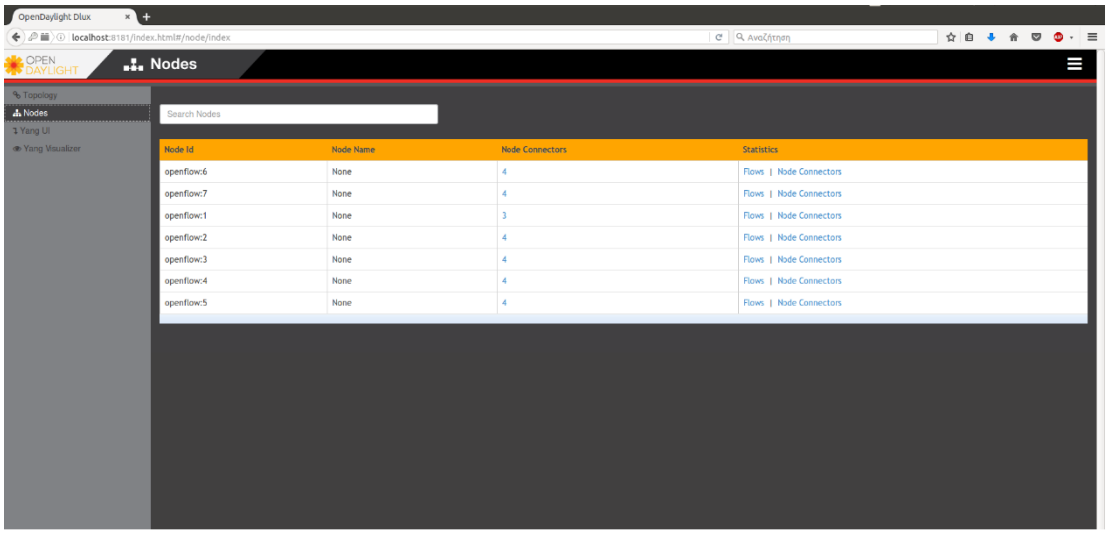
εικόνα 1. Topology display μέσω του DLUX UI

Στην εικόνα 1 φαίνεται σχηματικά η τοπολογία δέντρου την οποία έχουμε καλέσει από το Mininet και βλέπουμε του κόμβους-διακόπτες (switches) οι οποίοι την αποτελούν. Το DLUX δίνει την δυνατότητα να τοποθετήσουμε το ποντίκι στου Η/Υ στους διακόπτες για να δούμε της πηγές και τους προορισμούς. Επίσης μπορούμε να μεγεθύνομε και να σμικρύνουμε την τοπολογία χρησιμοποιώντας την κύλισης του ποντικιού για να την παρατηρήσομε ολόκληρη ή τις μικρότερες τοπολογίες που την αποτελούν.

Σημείωση: Το DLUX δεν παρέχει οποιαδήποτε τροποποίηση της τοπολογίας. Αυτό είναι εφικτό να γίνει σε άλλες ενότητες, όπως είναι το YangUI όπου και υπάρχουν εγκατεστημένα τα απαραίτητα MD-SAL plugins, με τα οποία μπορούμε να προσθέσουμε-αφαιρέσουμε και γενικά να αποθηκεύσουμε δεδομένα όπου το DLUX μπορεί να εμφανίσει. [13]

6.3 Nodes

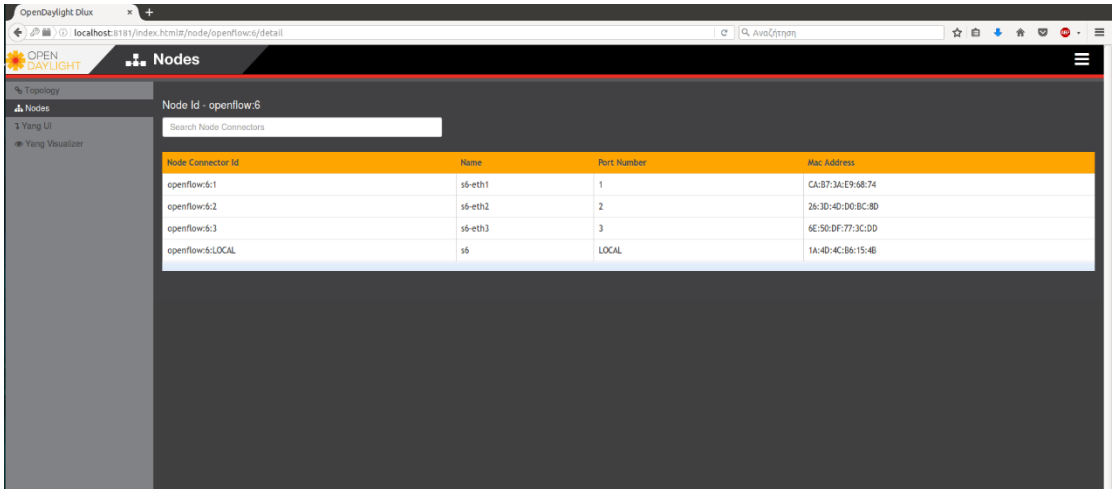
Στην ενότητα Nodes, όπως φαίνεται στην εικόνα 2 , εμφανίζονται η λίστα των κόμβων-διακόπτες του δικτύου , οι συνδέσεις μεταξύ τους καθώς και τα στατιστικά.



εικόνα 2. Nodes display μέσω του DLUX UI

Εάν διαλέξουμε ένα node id πχ το *Openflow:6* μπορούμε να επιλέξουμε το νούμερο των συνδέσεων του (Node Connectors) και να εισέλθουμε σε ένα πίνακα όπου θα εμφανιστούν όλες οι συνδέσεις του κόμβου αυτού καθώς και τα ονόματα των συνδέσεων, ο αριθμός των πορτών επικοινωνίας και η MAC διεύθυνση της κάθε μίας.

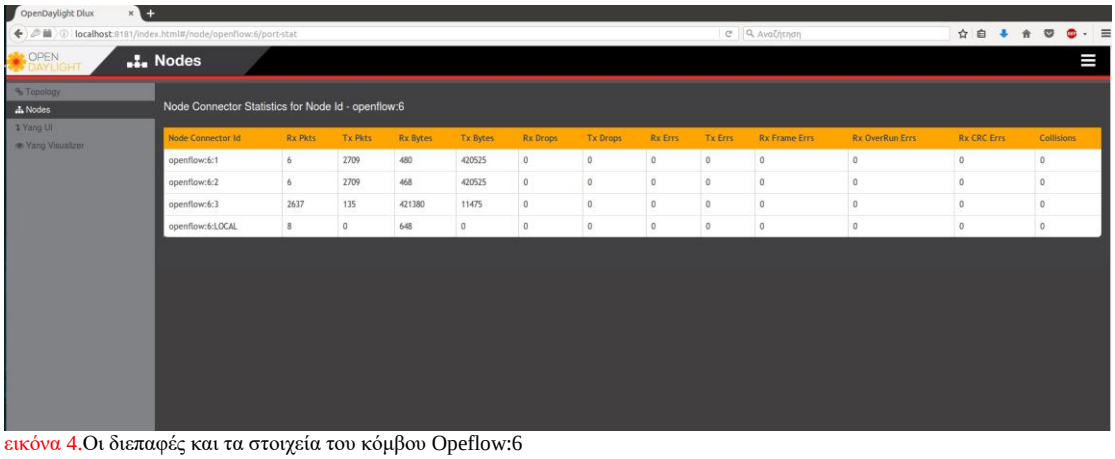
Στην εικόνα 3 φαίνεται ο πίνακας των Node Connectors του κόμβου *Openflow:6*



εικόνα 3. Node-Connectors display μέσω του DLUX UI

Στο ίδιο Node id επιλέγοντας στην στήλη Statistics την επιλογή Flows μας δίνεται η δυνατότητα να δούμε έναν Πίνακα Στατιστικών Ροής δεδομένων (Flow Table Statistics) και να δούμε ποιες ροές είναι ενεργές, πόσα πακέτα ταιριάζουν κλπ.

Τέλος στην στήλη Statistics υπάρχει και η επιλογή των συνδέσεων του κόμβου (Node Connectors) όπου μας εμφανίζει ένα πίνακα με τα βασικά στατιστικά της κάθε σύνδεσης όπως των αριθμών των πακέτων του δέκτη και παραλήπτη ,τον αριθμό των bytes , τον αριθμό των χαμένων πακέτων και κάποια ακόμα όπως φαίνονται στην εικόνα 4 .



εικόνα 4.Οι διεπαφές και τα στοιχεία του κόμβου Opeflow:6

6.4 Yang UI και οι εφαρμογές διαχείρισης τοπολογίας καθώς και εμφάνισης στατιστικών δεδομένων

Χαρακτηριστική και σημαντική εφαρμογή της γλώσσας μοντελοποίησης Yang, είναι το **Yang UI (**User Interface) το οποίο είναι μια βασική εφαρμογή για το γραφικό περιβάλλον,GUI (Graphic User Interface), του ελεγκτή και δίνει πρόσβαση στη διαμόρφωση αλλά και την κατάσταση του δικτύου. Το Yang UI είναι βασισμένο στο ***ODL DLUX UI*** και χρησιμεύει για μια πιο εύχρηστη διαχείριση ενός ODLελεγκτή.

6.4.1 Εφαρμογή διαχείρισης δικτυακής τοπολογίας : *network-topology API*

Με το network-topology API , ο ODL-Be ελεγκτής προσφέρει στον διαχειριστή έναν άμεσο και ευκολότερο τρόπο διαχείρισης της τοπολογίας του δικτύου του. Το API αυτό προσφέρει αρκετές δυνατότητες όσο αφορά τη διαχείριση της τοπολογίας του δικτύου. Μερικές από αυτές είναι:

- Topology (topology-id): Αναφέρεται στο τύπο της τοπολογίας του δικτύου καθώς και στο επίπεδο 3 του δικτύου, για συγκεκριμένα πρωτόκολλά όπως το ISIS και το OSPF.

- Node (node-id): Αναφέρεται στου κόμβους που αποτελούν την δικτυακή τοπολογία, στα δικτυακά τους γνωρίσματα (igp-attributes) και στα τερματικά *(termination-point (tp-id))* τα οποία τους αποτελούν.

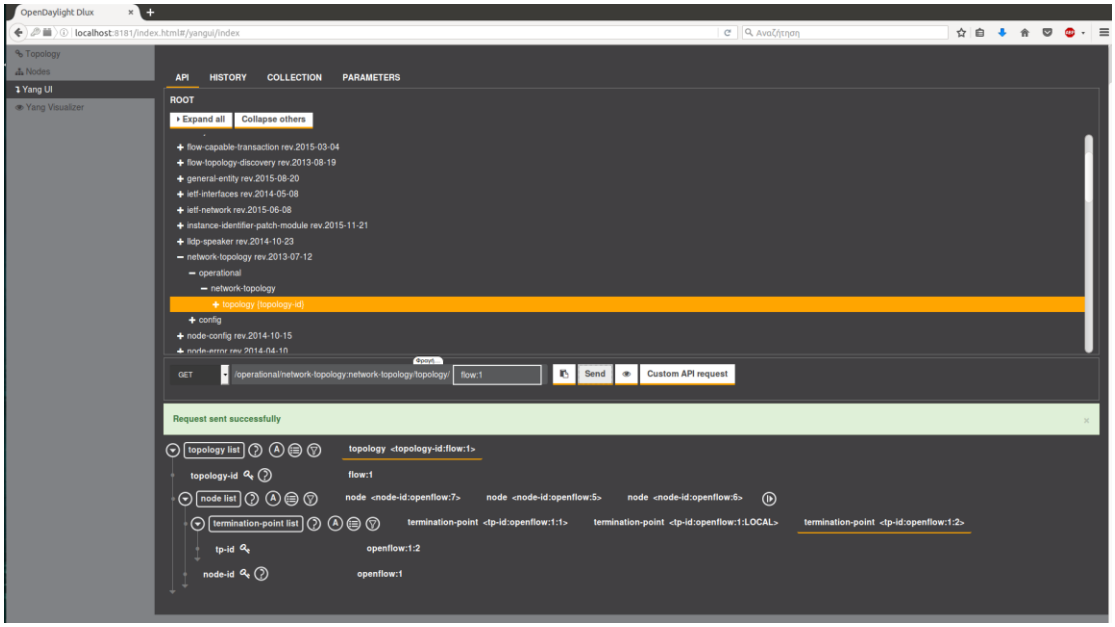
- Link (link-id): Αναφέρεται στους συνδέσμους μεταξύ των κόμβων αλλά και των τερματικών όσο αφορά την πηγή σύνδεσης, την κατεύθυνση σύνδεσης, τον σύνδεσμο υποστήριξης καθώς και για τα δικτυακά τους γνωρίσματα (*igp-link-attributes*).

operational

Για να μπορέσει ο διαχειριστής να εμφανίσει όλα τα δικτυακά δεδομένα καθώς και την τοπολογία του δικτύου, μέσω του Yang UI, επιλέγει αρχικά την επιλογή operational και αμέσως μετά πατάει το κουμπί ‘send’ έχοντας επιλεγμένη τη μέθοδο ‘GET’ στο network-topology. Τότε εκτελείται το API και φέρνει τα επιθυμητά αποτελέσματα. Με το κουμπί ‘Display Topology’εμφανίζεται σχηματικά η τοπολογία του δικτύου.

Εικόνα 5. Display Network Topology μέσω του Yang UI

Όπως φαίνεται στην εικόνα 5 το κάλεσμα με το 'GET' έγινε με επιτυχία και εμφάνισε τα δεδομένα της τοπολογίας που έχουμε ορίσει αρχικά από το Mininet σε ένα δέντρο το οποίο αποτελείται από τμήματα όπου διαχωρίζουν τα στοιχεία που αποτελούν την τοπολογία του δικτύου. Καλώντας λοιπόν το API *Network-Topology* εμφανίζεται στην κορυφή το *topology-list* το οποίο αποτελείται από όλες τις ροές του δικτύου.

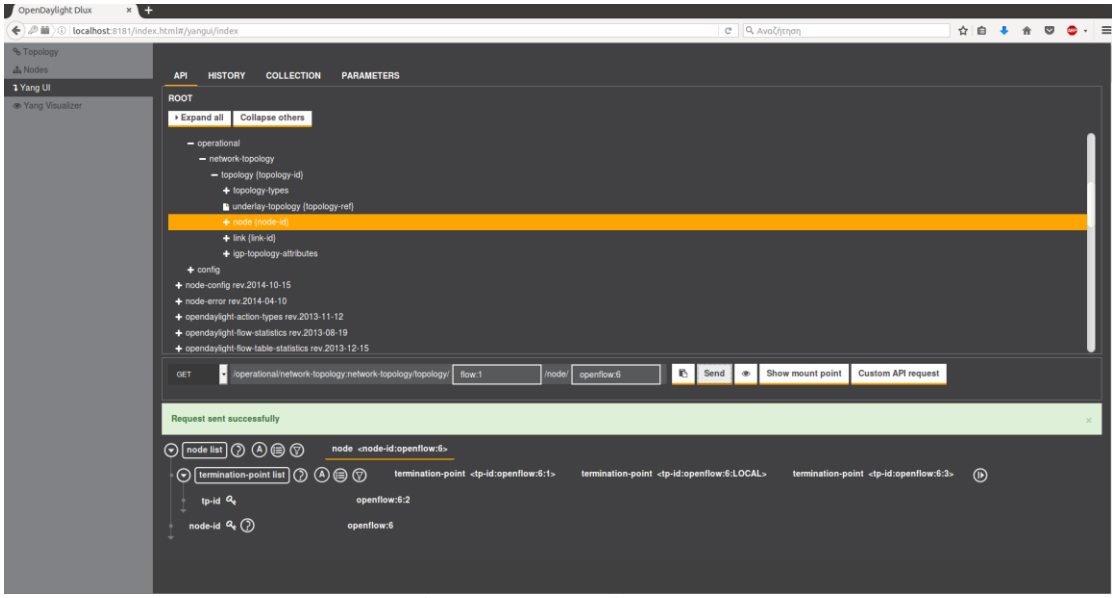


εικόνα 6 API: Network Topology: flow: 1

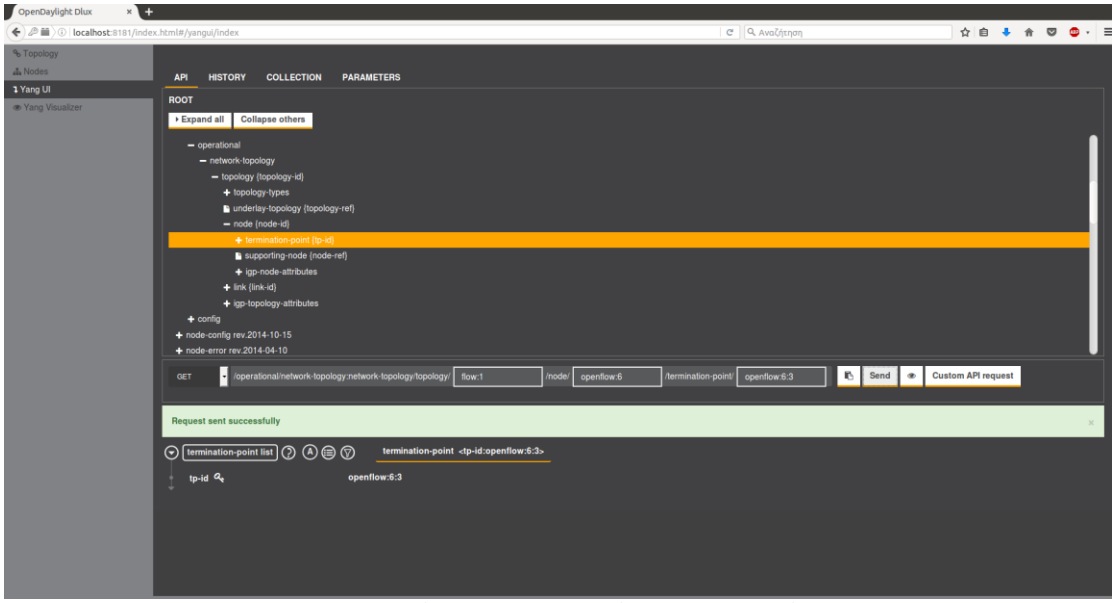
Στο συγκεκριμένο παράδειγμα έχουμε μία ροή την *flow:1* με το αντίστοιχο topology-id : *flow:1* το οποίο καθιστά την τοπολογία ως μοναδική (εικόνα 6). Στη συνέχεια του δέντρου εμφανίζεται το n*ode-list* το οποίο περιέχει όλους τους κόμβους της τοπολογίας καθώς και τα μοναδικά *node-id* του εκάστοτε κόμβου. Κάτω ακριβώς από το *node-list* βρίσκεται το *termination-point list* το οποίο περιέχει με την σειρά του όλα τερματικά στοιχεία τα οποία περιέχει ο κάθε κόμβος ξεχωριστά και τέλος το κάθε τερματικό εμπεριέχει το *termination-point-id (tp-id)* που αποδεικνύει κι αυτό με τη σειρά του την μοναδικότητα του κάθε

τερματικού.

Στις παρακάτω εικόνες (εικόνα 7, εικόνα 8) απεικονίζεται η πορεία μίας πιο εξεζητημένη αναζήτηση μέσω του *API Network Topology* ενός τερματικού με *tp-id: Openflow:6:3* το οποίο ανήκει στον κόμβο *Openflow:6* . Παρατηρείται ότι όσο περισσότερα στοιχεία αναζητάς από την τοπολογία του δικτύου, τόσο λιγότερα αλλά συγκεκριμένα αποτελέσματα φέρνει.



εικόνα 7. API: Network Topology: topology-id: flow:1 / node-id: openflow:6



εικόνα 8. API: Network Topology: topology-id: flow: 1 / node-id: openflow: 6 / tp-Id: openflow:6:3

Επιλέγοντας την display επιλογή (εικόνα 3.5.4) θα εμφανίζει το path URI του Restconf API το οποίο όταν το καλείς κατευθείαν από τον browser εμφανίζει το δέντρο σε XML μορφοποίηση .

Το URI είναι :

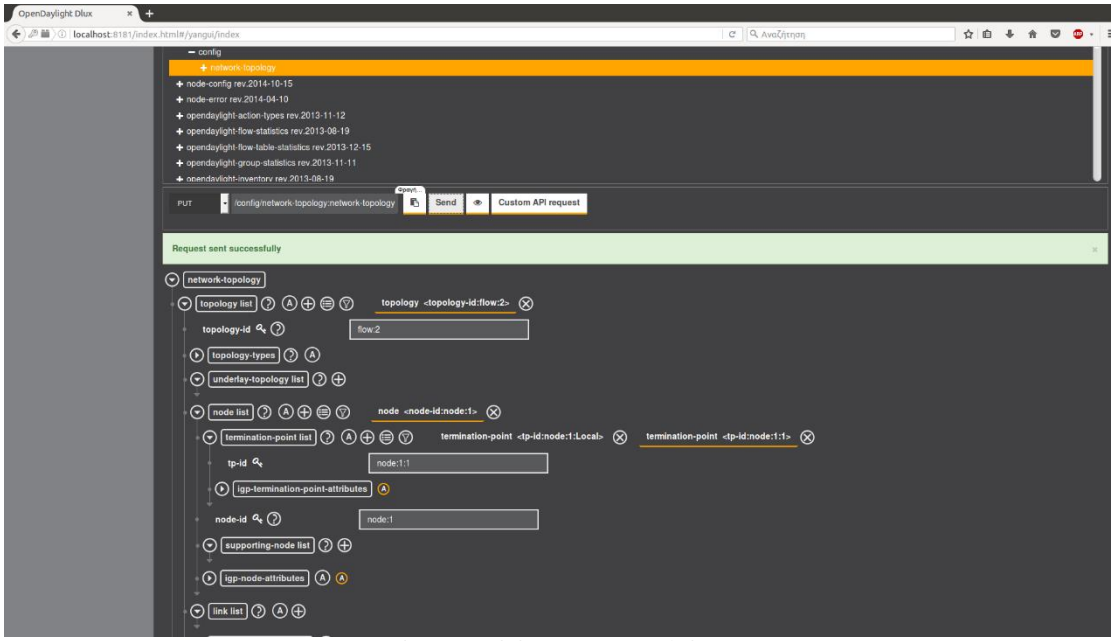
Το XML δέντρο είναι:

```
<Termination-point>
        <tp-id>openflow: 6:3</tp-id>
        <Inventory-node-connector-ref>
                /a: nodes/a: node [a:id='openflow:6']/a: node-connector[a: id='openflow:6:3']
        </inventory-node-connector-ref>
</termination-point>
```

Όπως φαίνεται και στις παραπάνω εικόνες ανάλογα με τα στοιχεία αναζήτησης εμφανίζονται και τα αντίστοιχα αποτελέσματα . Επειδή το τερματικό *Openflow:6:3* δεν αποτελείται από πολλά στοιχεία το δέντρο φέρνει λιγότερα αποτελέσματα . Στην αρχική περίπτωση που καλείται ολόκληρη η τοπολογία (*flow:1*), το API εμφανίζει στο δέντρο όλα τα στοιχεία της τοπολογίας αυτής. Επίσης το ίδιο ισχύει και όταν καλείται εκτός του ελεγκτή όπου εμφανίζεται με την XML μορφοποίηση.

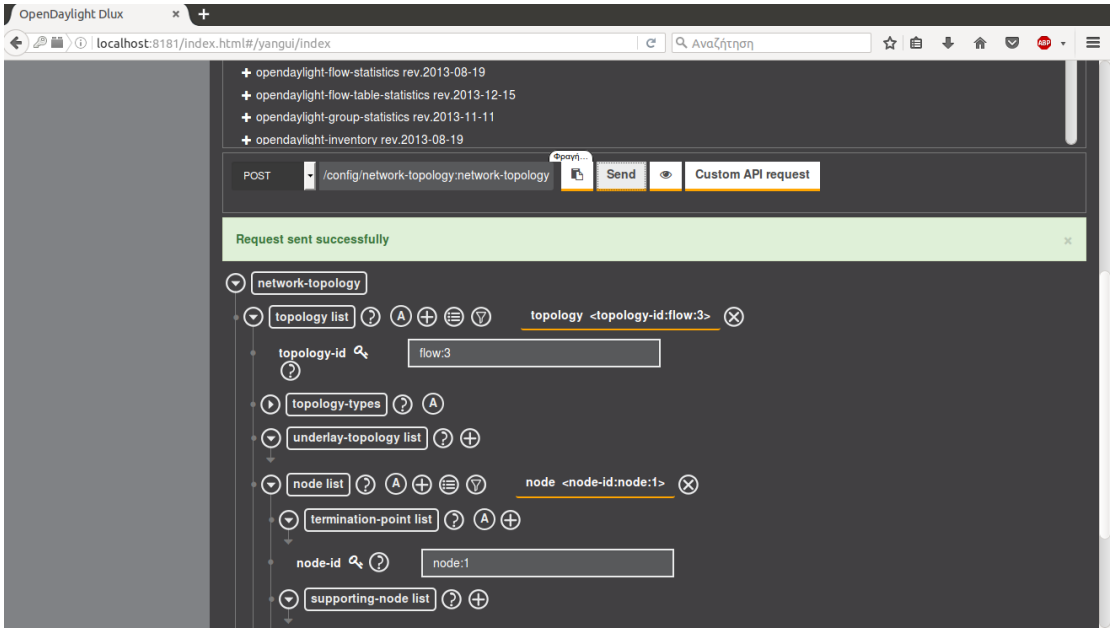
config

Με το Config επιτρέπει ο ελεγκτής στον διαχειριστή να επεξεργαστεί και να διαχειριστεί τα δικά του δικτυακά στοιχεία και δεδομένα. Στο παράδειγμα. του *Network Topology API* εισάγονται τα βασικά δεδομένα που μπορούν να αποτελέσουν μια μικρή τοπολογία. Δηλαδή στη επιλογή του *Network Topology* εισάγεται το *topology-id : flow:2* και ένας κόμβος με *node-id : node:1* , ο οποίος αποτελείται από δύο τερματικά με *tp-id : node:1:1* και *node:1:Local*. Εισαγωγή των δεδομένων αυτών γίνεται με την επιλογή 'PUT' και φαίνεται χαρακτηριστικά στην εικόνα εικόνα 9



εικόνα 9. API: Network Topology: Εισαγωγή δικτυακών δεδομένων με τη μέθοδο 'PUT'

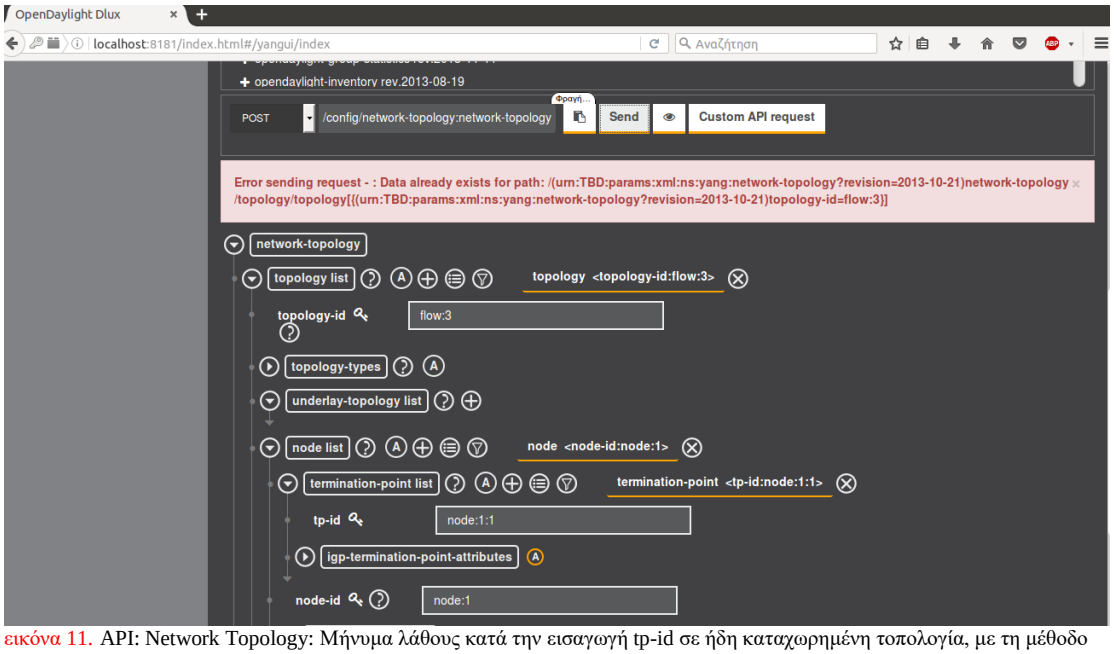
Ένας άλλος πιο πρακτικός τρόπος εισαγωγής αλλά και εμφάνισης εισαχθέντων δεδομένων γίνεται με τη μέθοδο 'POST'. Με αυτή τη μέθοδο δίνεται η δυνατότητα στο διαχειριστή να εισάγει σε μία καινούργια τοπολογία δικτύου όσα στοιχεία και δεδομένα θέλει.



εικόνα 10 API: Network Topology: Εισαγωγή δικτυακών δεδομένων με την μέθοδο 'POST

Μετά την εκτέλεση της μεθόδου αυτής ο διαχειριστής δεν έχει την δυνατότητα να προσθέσει επιπλέον στοιχεία ή δεδομένα χρησιμοποιώντας τη ίδια μέθοδο διότι μετά το πέρας της εκτέλεσης της μεθόδους τα νέα δεδομένα έχουν καταχωρηθεί και έχουν εμφανιστεί (καλεστεί) σχεδόν ταυτόχρονα. Οπότε ανοίγοντας την λίστα της νέας καταχώρισης, πχ ενός τερματικού (termination-point list), με την μέθοδο 'POST' θα πρέπει να εισαχθούν και να εμφανιστούν, μαζί με την νέα καταχώριση, τα ήδη υπάρχοντα δεδομένα. Αυτό είναι ανορθόδοξο γι' αυτό και ο ODL-Be ελεγκτής εμφανίζει μήνυμα λάθους.

Σε περίπτωση προσθήκης του τερματικού *node:1:1* στον κόμβο *node:1* μετά το πέρας της επιτυχής εισαγωγής , το API θα εμφανίσει μήνυμα λάθους ενημερώνοντας τον χρήστη πώς δεν είναι εφικτό να προσθέτει όμοιο καταχωρημένο στοιχείο (εικόνα 11).

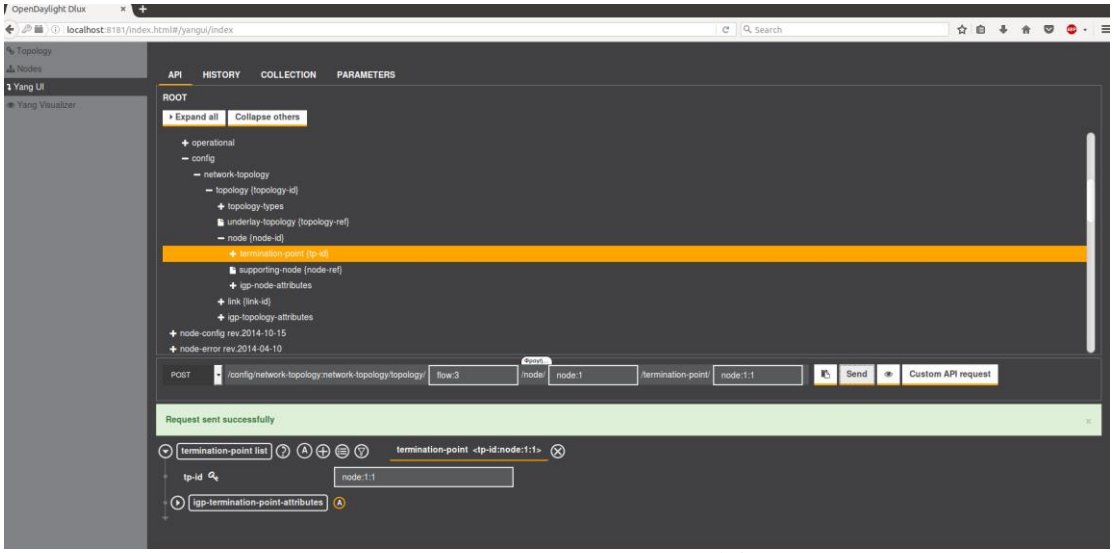


εικόνα 11. API: Network Topology: Μήνυμα λάθους κατά την εισαγωγή tp-id σε ήδη καταχωρημένη τοπολογία, με τη μέθοδο 'POST'

Για να γίνει εφικτή η νέα καταχώρηση χρησιμοποιώντας τη μέθοδο 'POST' θα πρέπει να είναι

λίστα των τερματικών μόνη της χωρίς την εμφάνιση των ήδη καταχωρημένων δεδομένων. Στη περίπτωση προσθήκης επιπλέον δεδομένων σε ήδη καταχωρημένα, συνιστάται η μέθοδος ‘PUT’ η οποία εισάγει τα δεδομένα χωρίς να τα καλεί.

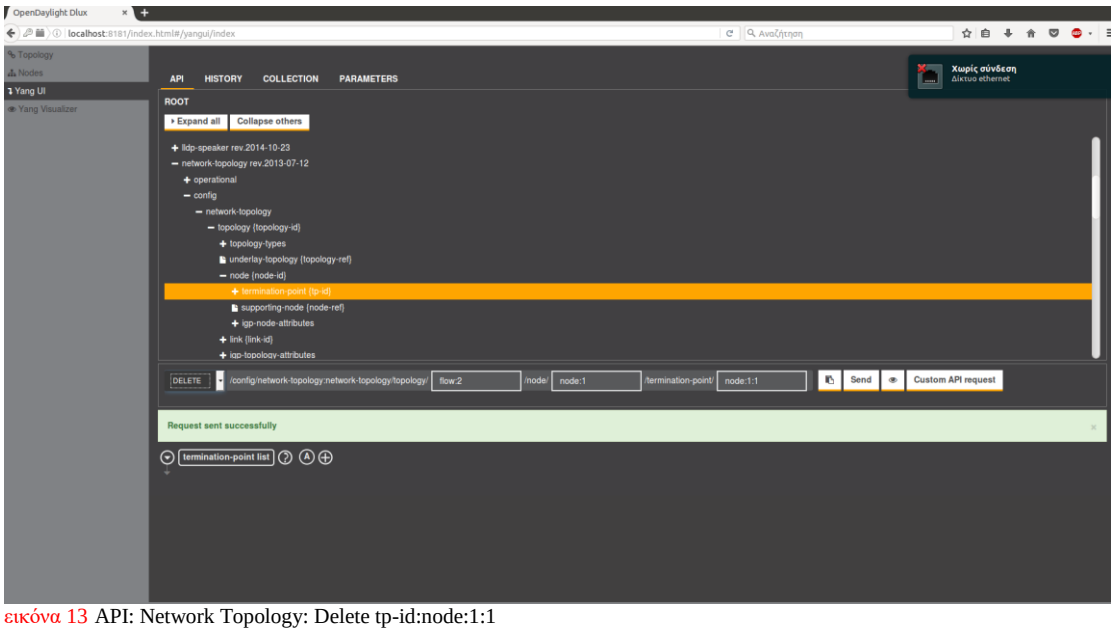
Για εισαχθεί με επιτυχία το τερματικό με tp-id:*node:1:1* θα πρέπει να εισέλθουμε στη λίστα με τα τερματικά (termination-point-list) και τότε να εισάγουμε το τερματικό με τη μέθοδο ‘POST’ ώστε να καταχωρηθεί και να εμφανιστεί ταυτόχρονα, όπως φαίνεται στην εικόνα 12.



εικόνα 12 API: Network Topology: Εισαγωγή τερματικού tp-id: node: 1:1 με τη μέθοδο ‘POST’

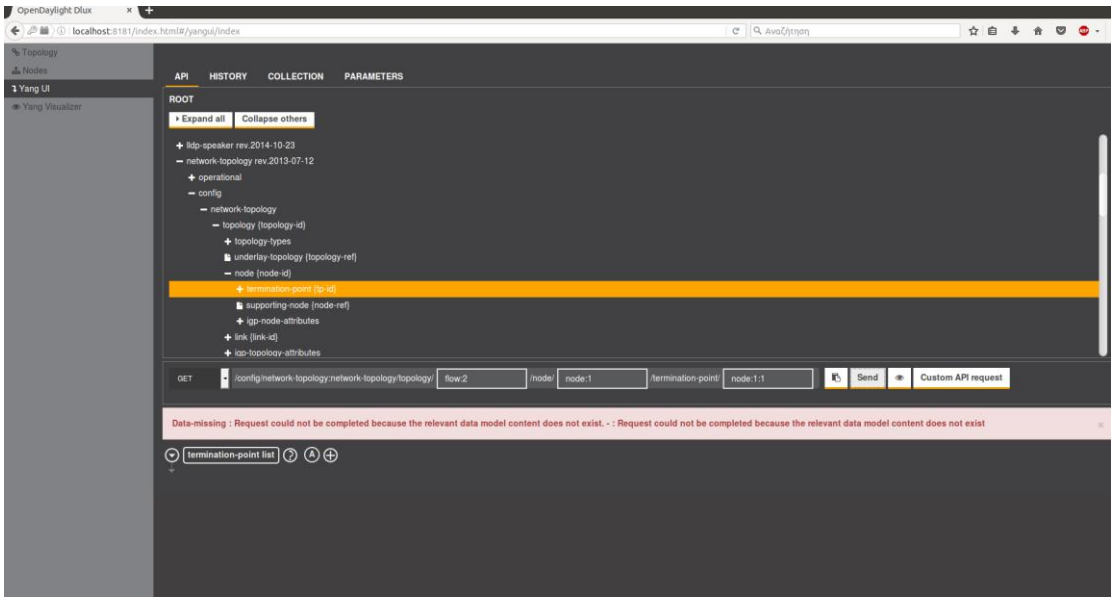
Με τη μέθοδο ‘GET’ μπορεί ο διαχειριστής να καλέσει τα δεδομένα που έχει εισάγει. Όπως και στην και στο operational έτσι και στο config δίνεται η δυνατότητα της πιο εξεζητημένης αναζήτησης καθώς και του δέντρου εμφάνισης των δεδομένων αυτών.

Τέλος , με την μέθοδο ‘DELETE’ ,μπορεί ο διαχειριστής να διαγράψει οποιοδήποτε δεδομένο που έχει αρχικά εισάγει όπως ένα τερματικό, ένα σύνδεσμο, έναν κόμβο ακόμα και μια τοπολογία. Στη εικόνα 13 φαίνεται η επιτυχής διαγραφή του τερματικού *node:1:1* που άνηκε στο κόμβο *node:1* της τοπολογία *flow:2* .



εικόνα 13 API: Network Topology: Delete tp-id:node:1:1

Σε περίπτωση που εκτελεστεί μία επιλογή ενώ δεν υπάρχουν τα δεδομένα τα οποία προσπαθεί να ανατρέξει το API, τότε εμφανίζει μήνυμα λάθους. Ένα παράδειγμα μηνύματος λάθους φαίνεται στην εικόνα 14 όπου ο διαχειριστής προσπαθεί να καλέσει με 'GET' το τερματικό *node:1:1*.



εικόνα 14 API: Network Topology: Data Missing error

6.4.2 Εφαρμογή στατιστικών δεδομένων : *opendaylight-inventory API*

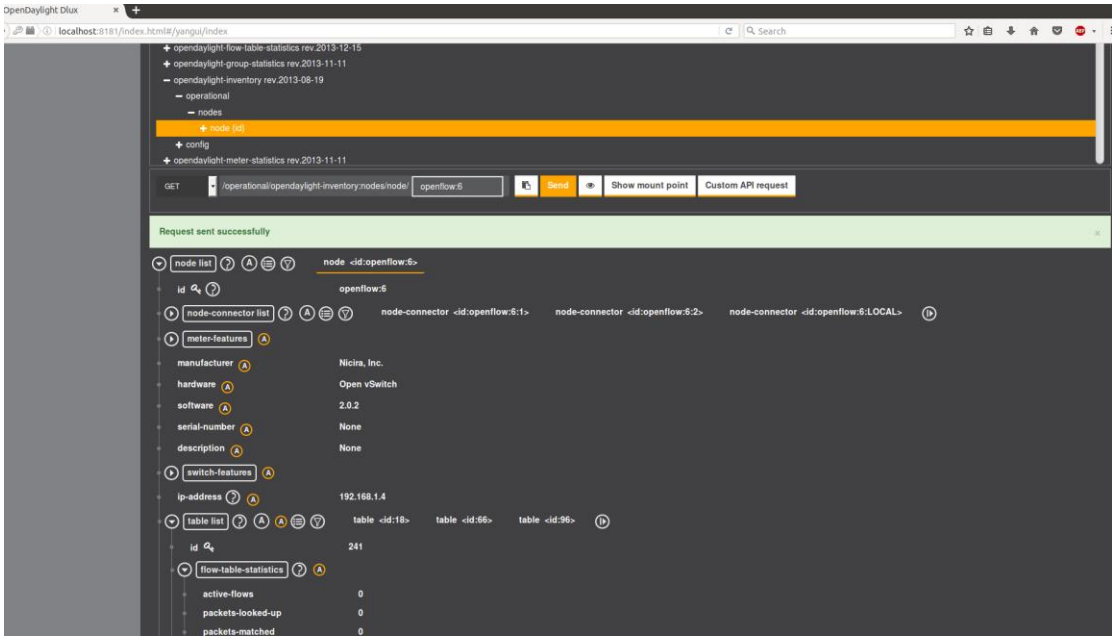
Με το opendaylight-inventory API, ο ODL-Be ελεγκτής προσφέρει στον διαχειριστή έναν άμεσο και ευκολότερο τρόπο διαχείρισης ενός ή και όλων των κόμβων του δικτύου. Ο διαχειριστή μπορεί πλέον να παρακολουθεί συνεχώς τη ροή των κόμβων του δίκτυού καταγράφοντας τα στατιστικά δεδομένα τα οποία επιδεικνύουν τη κατάσταση των κόμβων του δικτύου και των στοιχείων τους. Το API αυτό προσφέρει αρκετές δυνατότητες καταγραφής στατιστικών δεδομένων των κόμβων του δικτύου. Κάποιες σημαντικές δυνατότητες αναφέρονται παρακάτω:

- node-list & node-connector-list
- flow-capable-node-connector-statistics
- status
- state
- port-number & hardware-address
- switch-features
- table-list

operational

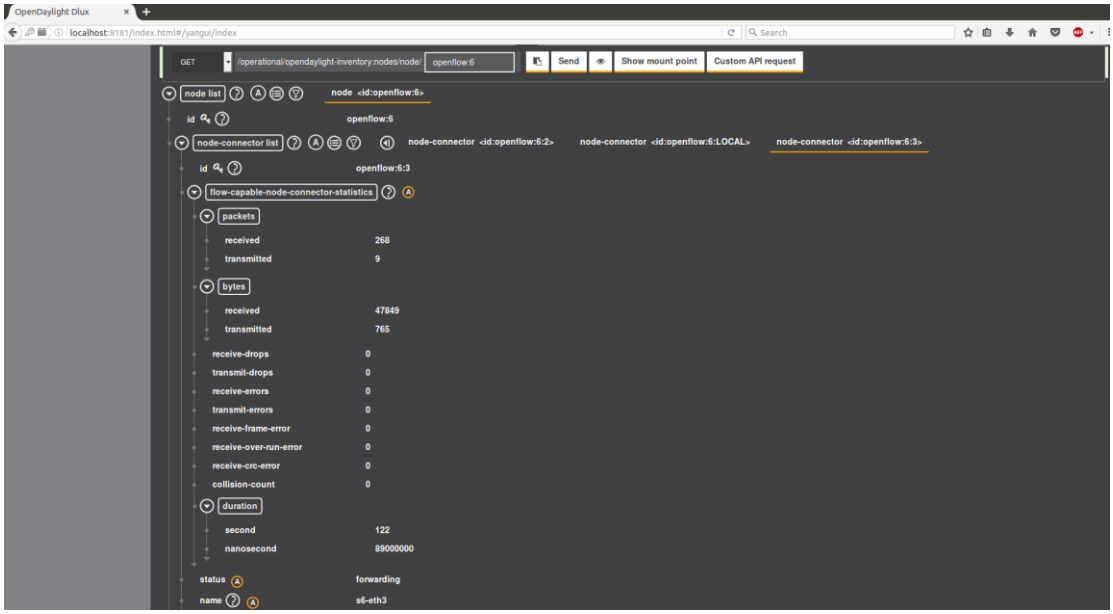
Για να μπορέσει ο διαχειριστής να καταγράψει όλους τους κόμβους του δικτύου καθώς και τα στατιστικά τους δεδομένα, επιλέγει αρχικά την επιλογή operational και αμέσως μετά πατάει το κουμπί 'send' έχοντας επιλεγμένη τη μέθοδο 'GET' στο node. Τότε εκτελείται το API και φέρνει τα επιθυμητά αποτελέσματα. Για να μπορέσει να εξετάσει αναλυτικά έναν κόμβο μπορεί ή να τον επιλέξει από το ήδη εμφανισμένο node-list ή να ανοίξει το *node* ώστε να εμφανιστεί

το *node{id}* και να απομονώσει τον κόμβο τον οποίο θέλει να μελετήσει. Στο παρακάτω παράδειγμα ο διαχειριστής καλεί τον ελεγκτή να εμφανίσει τα στατιστικά δεδομένα του κόμβου *"openflow:6"*.



εικόνα 15. API: opendaylight-inventory: node <id:openflow:6 >

Όπως φαίνεται στην εικόνα 15, ο διαχειριστής καλεί τον *openflow:6* κόμβο με την μέθοδο της 'GET' ώστε να εμφανιστούν τα στατιστικά δεδομένα που αποτελούν τον κόμβο αυτό. Στην επιλογή node-list εμφανίζεται ο κόμβος οποίος έχει επιλεχθεί να εμφανιστεί. Βασικά στατιστικά δεδομένα που χαρακτηρίζουν τον κόμβο αυτό όπως, ο κατασκευαστής του (manufacture : Nicira,Inc) , το υλικό από το οποίο αποτελείται (hardware : OpenVSwitch) και η IP address : 192.186.1.4 (localhost) τα οποία τα έχουμε δηλώσει κατά την εγκατάσταση του ελεγκτή και του δικτύου στο προηγούμενο κεφάλαιο.

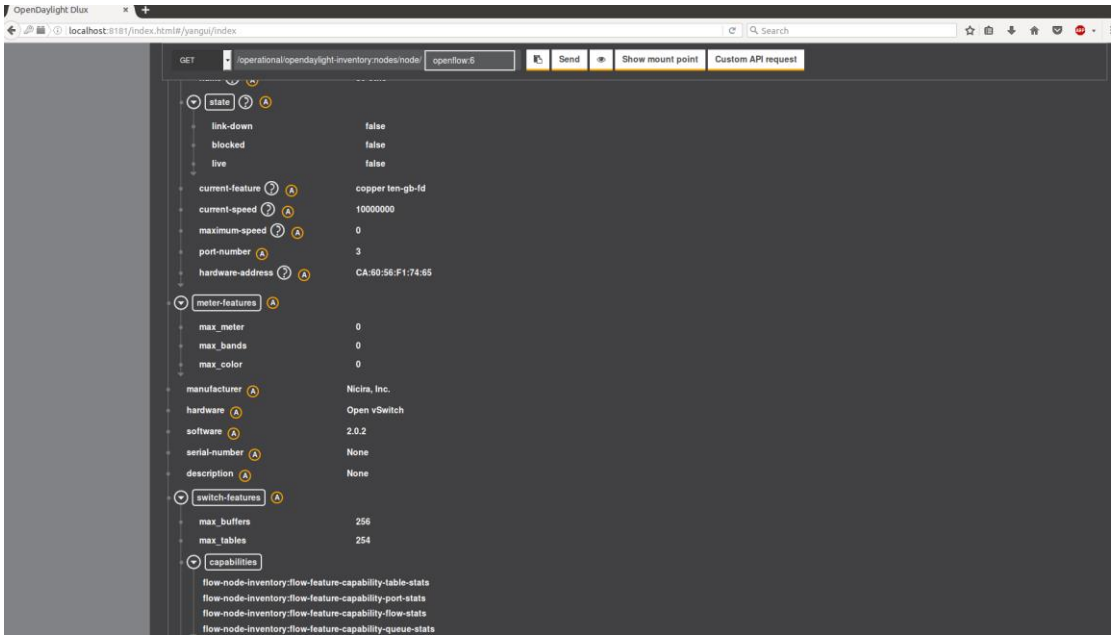


εικόνα 16. API: opendaylight-inventory: node-connector <id:openflow:6:3>

Το node-connector-list αναφέρεται στους συνδέσμους του εκάστοτε κόμβου. Ο *node-connector*, ή αλλιώς *termination point* για το *network-topology* API, είναι το κάθε τερματικό που συνδέεται με τον κόμβο και αποτελείται από πολλά στατιστικά δεδομένα και στοιχεία τα οποία το χαρακτηρίζουν. Όπως φαίνεται στην εικόνα 16, ο διαχειριστής επιλέγει τον σύνδεσμο openflow:6:3

Στο *flow-capable-node-connector-statistics* περιέχονται τα τελευταία στατιστικά δεδομένα της χρονικής στιγμή που εκτελέστηκε η μέθοδος 'GET', όπως η ποσότητα των πακέτων που έχουν μεταδοθεί και παραληφθεί από τον node-connector , πόσα bytes έχουν μεταδοθεί και παραληφθεί καθώς και πόσα έχουν χαθεί, έχουν φρακάρει και έχουν προκαλέσει συμφόρηση στο δίκτυο ή κάποιο άλλο σφάλμα που αφορά την μεταφορά τους. Τέλος, τα αποτελέσματα της διάρκειας της σύνδεσης του σε seconds και nanoseconds, την κατάσταση του και το όνομα του.

Στον *openflow:6:3* συγκεκριμένα φαίνονται χαρακτηριστικά τα πως έχει λάβει 268 πακέτα και 47849 bytes καθώς και έχει μεταδώσει 9 πακέτα και 765 bytes. Η διάρκεια σύνδεσης του είναι 122 seconds ή 89000000 nanoseconds, η κατάσταση του είναι forwarding δηλαδή σε κατάσταση προώθησης και το όνομα του είναι το όνομα που του έχει δοθεί κατά την εκτέλεση του δικτύου από το Mininet,*s6-eth3*. Ο σύνδεσμος αυτός θεωρείται υγιής διότι δεν παρουσίασε κάποιο σφάλμα κατά την διαδικασίας της μεταφοράς δεδομένων.



εικόνα 17. API: opendaylight-inventory: node-connector <id:openflow:6:3> (συνέχεια)

Στη συνέχεια παρατηρείται ,στην εικόνα 17 , η κατάσταση του συνδέσμου εάν είναι κατεβασμένος, μπλοκαρισμένος και σε πραγματικό χρόνο (live) καθώς και σημαντικά δεδομένα όπως τον αριθμό της πόρτας (port-number) στην οποία ακούει και η φυσική του διεύθυνση (hardware-address) . Ο σύνδεσμος *openflow:6:3* έχει το '3' στο τέλος το οποίο χαρακτηρίζεται από το port-number που ισούται με 3.

Παρακάτω παρουσιάζονται βασικά γνωρίσματα μετρητών που στο συγκεκριμένο παράδειγμα είναι 0 και βασικά γνωρίσματα του διακόπτη όπως το *max_buffers* που είναι 256 ρυθμιστές και το *max-tables* που είναι 254 πίνακες. Στην περίπτωση των ικανοτήτων (capabilities) του κόμβου αναφέρονται όλες οι δυνατότητες των στατιστικών των πινάκων, των δικτυακών πορτών και ροών καθώς και ουρών αναμονής που σχετίζονται με τον κόμβο. Τέλος , όπως

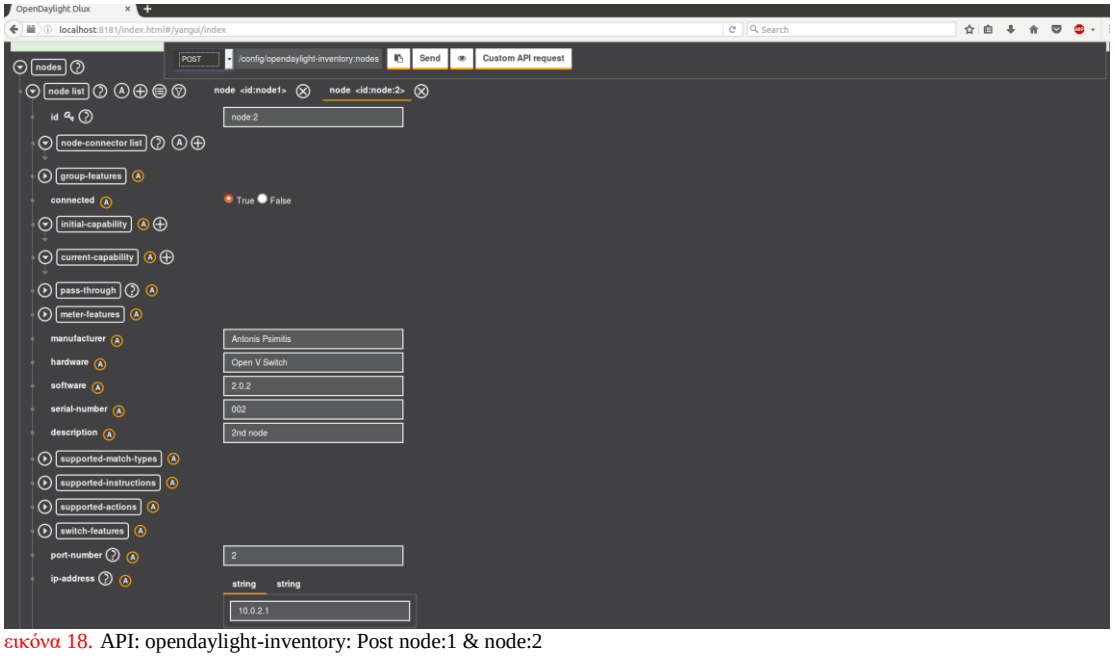
φαίνονται στην εικόνα 1, καταγράφονται ξεχωριστά όλοι οι πίνακες και στον καθένα ξεχωριστά τα στατιστικά που τον αποτελούν όπως πόσες ροές είναι ενεργές και πόσα πακέτα στέλνονται προς τις πάνω διαδρομές και πόσα είναι ταιριαστά.

config

Όπως και με το συμβαίνει με το *network-topology API* έτσι και εδώ με το *config* ο ODL-Be ελεγκτής δίνει την δυνατότητα στον διαχειριστή να προσθέσει και να διαχειριστεί τους δικούς του κόμβους καθώς και όλα τα στατιστικά δεδομένα που τους αποτελούνε.

Όλα τα στοιχεία που υπάρχουν στην επιλογή *operational* αντιστοιχούν και στην επιλογή *config* με την διαφορά πως θα πρέπει ο διαχειριστή να τα εισάγει μόνος του. Έτσι, θα μπορέσει να δημιουργήσει το δικό του node-list καθώς και όλα περιεχόμενα του. Ανοίγοντας το node-list ο διαχειριστής μπορεί να προσθέσει όσους κόμβους θέλει και σε κάθε έναν από αυτούς να του δημιουργήσει τους node-connectors που χρειάζεται, τις ομάδες λειτουργιών τους , την ποσότητα των μετρητών-ρυθμιστών τους, τα βασικά χαρακτηριστικά που τακτοποιούν τον κόμβο όπως η IP διεύθυνσή τους καθώς και άλλες λειτουργίες τις οποίες συναντήσαμε στη επιλογή *operational* .

Παρακάτω θα περιγράψω ένα παράδειγμα εισαγωγής στατιστικών δεδομένων για τους κόμβους node:1 και node:2, όπου βοηθάει στην κατανόηση της λειτουργίας του *config* για το *opendaylight-inventory API.*

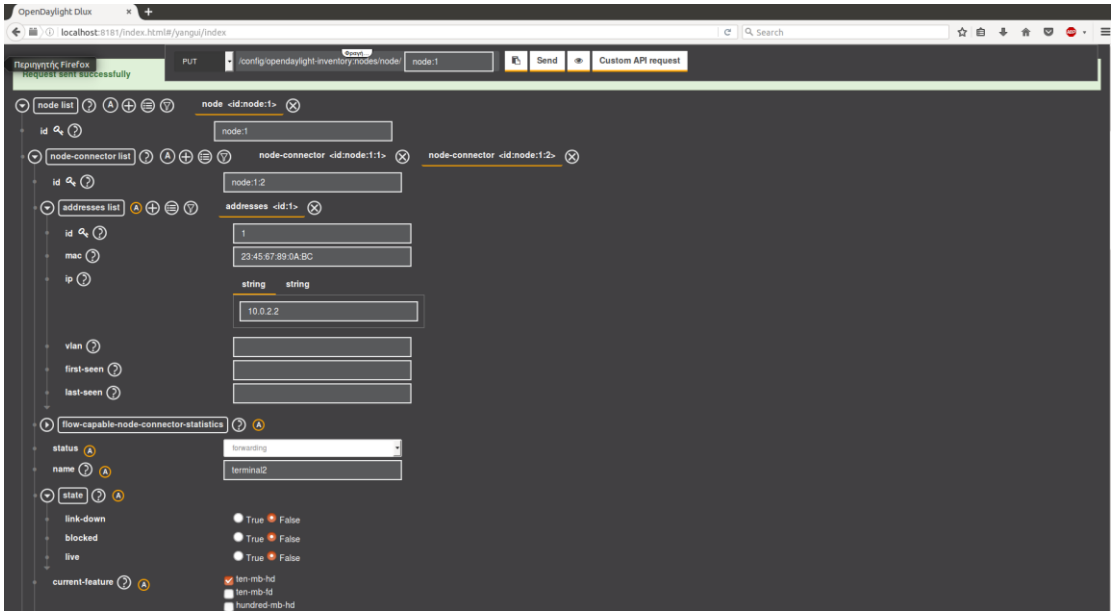


εικόνα 18. API: opendaylight-inventory: Post node:1 & node:2

Με την μέθοδο ‘POST’, επειδή είναι και αυτό ένα Restful API, εισήγαγα 2 κόμβους τους node:1 και node:2. Στην εικόνα 18 φαίνεται χαρακτηριστικά πως τα πρώτα και βασικά στοιχεία των κόμβων είναι συμπληρωμένα και πλέον αποτελούν την ταυτότητα τους. Συγκεκριμένα μας δείχνει πως για τον node:2 το όνομα του κατασκευαστή του είναι ‘Antonis Psimitis’ το υλικό του είναι ‘Open V Swtich’ , η έκδοση του λογισμικού είναι η ‘2.0.2’ , σειριακός του αριθμός είναι ‘002’ , μια σύντομη περιγραφή και είναι συνδεδεμένος καθώς ακούει στην πόρτα ‘:2’ με IP διεύθυνση ‘10.0.2.1’. Τα ίδια περίπου στοιχεία ισχύουν και για τον κόμβο node:1 με την διαφορά σε χαρακτηριστικά που τον ταυτοποιούν όπως το nodeid (node:1) , IP διεύθυνση

(10.0.1.1) και πόρτα(:1) καθώς και σειριακό αριθμό (001).

Στη συνέχεια, για να προσθέσω στοιχεία στου 2 κόμβους , χρησιμοποιώ την μέθοδο ‘PUT’ και έχω τη δυνατότητα να κάνω όσες αλλαγές θέλω. Επιλέγω τον 1ο κόμβο με id : node:1 και ανοίγω στο δέντρο με τα στοιχεία του το node-connector list . Εισάγω 2 τερματικά με id : node:1:1 και node:1:2 και έπειτα συμπληρώνω τα λοιπά στοιχεία που τα αποτελούν.



εικόνα 19. API: opendaylight-inventory: Put node:1:1 & node:1:2

Όπως φαίνεται και στην εικόνα 5, κάποια από τα βασικά στοιχεία που πρέπει να εισαχθούν είναι το node-connector id , addresses list , status ,name , state κτλ. Συγκεκριμένα στο terminal2 με node-connector id : node:1:2, έχουν καταχωρηθεί στοιχεία όπως :

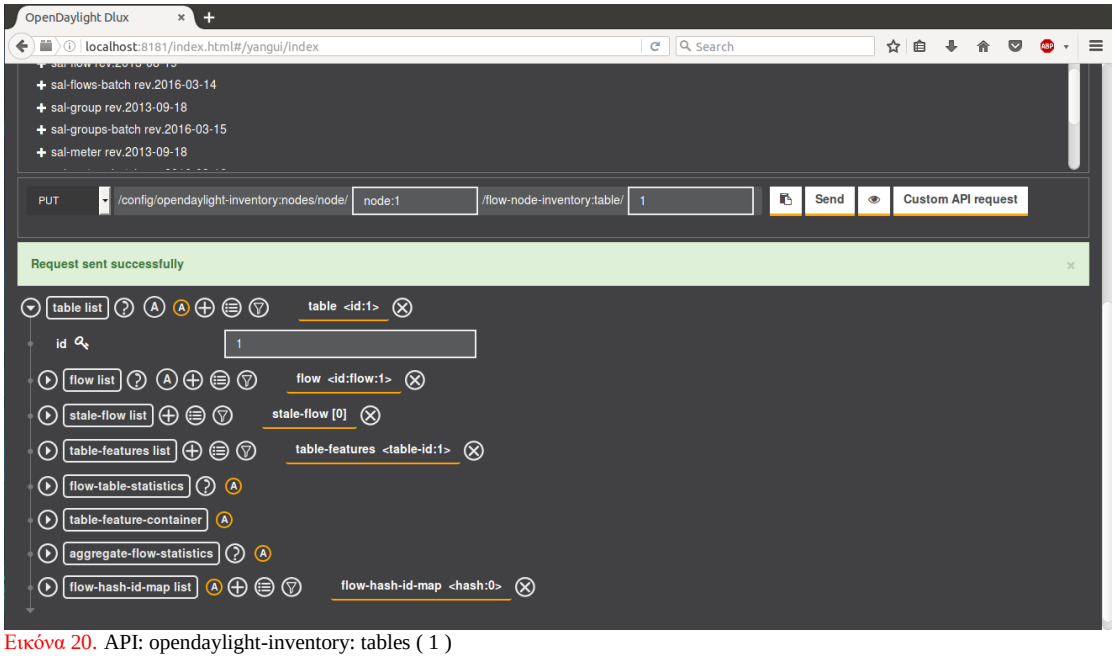
- address id : 1
- MAC address: 23:45:67:89:0A:BC (η οποία είναι ίδια με την Hardware-address που ζητείται παρακάτω)
- IP address: 10.0.2.2
- state: forwarding
- name: terminal2
- link up
- non blocked
- no live

Επίσης, άλλα στοιχεία που μπορούν να καταχωρηθούν είναι λειτουργίες όπως η σε ποιες ταχύτητες υποστηρίζεται αυτός και ομότιμοι του για παράδειγμα στο terminal2 τα current, supported και peer features είναι επιλεγμένα να υποστηρίζονται σα 10 MB.

Μερικές από τις υπόλοιπες επιλογές καταχώρησης στοιχείων ενός κόμβου παρουσιάζονται επιγραμματικά παρακάτω με σημαντικότερη την λίστα πινάκων καθώς οι πίνακες παίζουν σημαντικό ρόλο στο προώθηση πακέτων μέσα σε ένα δίκτυο διότι περιλαμβάνουν τα στοιχεία του αποστολέα αλλά και του παραλήπτη και του πακέτου το οποίο μεταδίδεται και βοηθάνε τους ενδιάμεσους κόμβους να μπορούν να βρίσκουν τη κατάλληλη διαδρομή για να φτάσει το στο προορισμό του.

- group features
- initial & current capability
- meter features

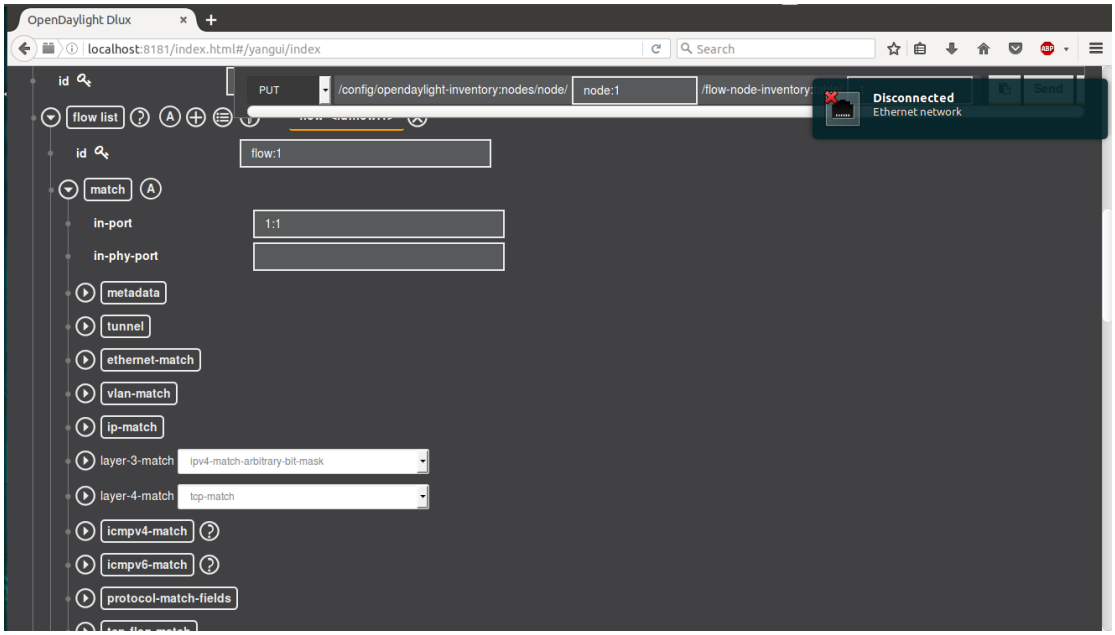
- switch features
- supported match-types, instructions & actions
- group & meter list
- table list



Εικόνα 20. API: opendaylight-inventory: tables (1)

Όσο αφορά το table list είναι πολύ σημαντικό να σημειωθούν αρκετές από τις λειτουργίες τους διότι όπως προανέφερα οι πίνακες σε ένα δίκτυο αποθηκεύουν και μαζί τους μεταφέρουν όλες τις πληροφορίες χρήσιμες και μη , για τα πακέτα που είναι σε θέσει να μεταδοθούν.

Όπως ανοίγει ο διαχειριστής το table list το πρώτο πράγμα που πρέπει να συμπληρώσει είναι το id του πίνακα και ακριβώς από κάτω του δίνεται μία λίστα για να δηλώσει τις δικτυακές ροές που θα ακολουθήσει ο πίνακας (flow-list). Για να του δοθούν τα στοιχεία που αποτελούν μία ροή, θα πρέπει πρώτα να προσθέσει στην λίστα τουλάχιστον μία. Τότε θα του εμφανιστούνε στοιχεία όπως το id που είναι πάντα βασικό, προαιρετικά οφείλει να συμπληρώσει μία κατηγορία η οποία κάνει το ταίριασμα των πακέτων κατά την διάρκεια της διαδρομής τους , όπως τα *metadata* που θα χρησιμοποιηθούν , το *Ethernet* , εάν είναι σε *vlan*, ποιες *ip διευθύνσεις* θα ταιριάξουν . Για το επίπεδο 3 του δικτύου μπορούν να δηλωθούν οι IP διευθύνσεις της πηγής και του παραλήπτη όπως και για το επίπεδο 4 πιο πρωτόκολλο θα ακολουθήσουν πχ TCP.

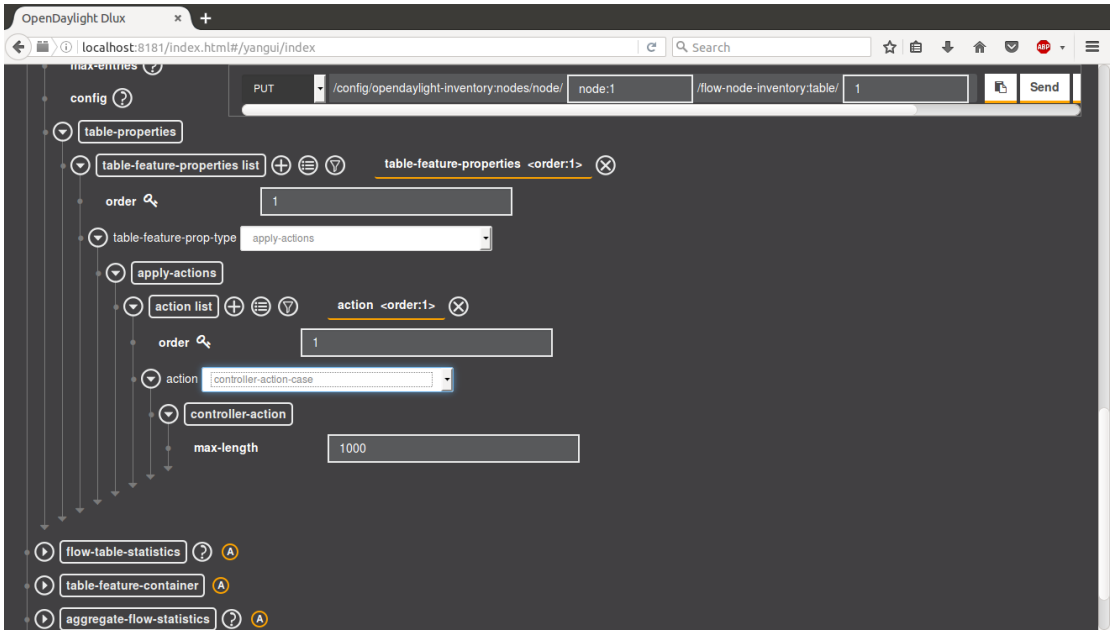


εικόνα 21. API: opendaylight-inventory: table (2)

Μετά το match ακολουθούν κάποιες οδηγίες (instructions) οι οποίες μπορούν να δοθούν μπαίνοντας και αυτές με τη σειρά του σε μία λίστα. Οι οδηγίες αυτές κατευθύνουν το πακέτο στο πώς πρέπει αν λειτουργήσει αφότου φτάσει στο ν προορισμό του πχ υπάρχει η επιλογή *apply-actions-case* όπου δίνει τη δυνατότητα διαγράφης του πακέτου με το που ολοκληρώσει το σκοπό του ή αντιγραφής με το που φτάσει στον προορισμό του, και αρκετές άλλες. Επίσης, χαρακτηριστικά όπως το όνομα του περιεχομένου καθώς και το όνομα της ροής ή την επιλογή τι είδους μετρητές πρέπει να έχει δεν πρέπει να μένουν ασυμπλήρωτα.

Στη συνέχεια από τη λίστα των ροών ακολουθεί η λίστα των stale ροών (stale-flow-list) που εφόσον επιλέξει ο διαχειριστής να έχει έστω μία ροή , τα περιεχόμενα της είναι ακριβώς ίδια με αυτά του *flow-list* . Στο table-feature list περιέχονται στοιχεία όπως , το id και το όνομα του πίνακα, πληροφορίες για metadata, καθώς και οι ιδιότητες του πίνακα οι οποίες κατατάσσονται σε μία λίστα με σειρά προτεραιότητας. Εκεί επιλέγει ο διαχειριστής τον τύπο της ιδιότητας το πίνακα ένα θα είναι apply-actions ,instructions , match , next-table και πολλούς άλλους και μετά αναλόγως τι θα επιλέξει θα κάνει το ίδιο με την επιλεγμένη ιδιότητα.

Για παράδειγμα στον *table-feature-prop-type:apply-actions* , μπορεί ο διαχειριστής να επιλέξει η δράση να γίνεται από τον ελεγκτή με το controller-action-case και να δηλώσει το μέγεθος των δράσεων που θα προβάλει ο ελεγκτής όπως φαίνεται στην εικόνα.



εικόνα 22. API: opendaylight-inventory: table (3)

Στο flow-table-statistics καταχωρούνται πόσες ενεργές ροές υπάρχουν καθώς και πόσα πακέτα είναι αναρτημένα και πόσα ταιριαστά. Στο table-feature-container εμπεριέχεται το table-feature-list το οποίο έχει τα ίδια στοιχεία που ανέφερα παραπάνω στη αντίστοιχη λίστα. Τέλος το δέντρο του πίνακα κλείνει με το *aggregate-flow-statistics* στο οποίο καταχωρούνται οι μετρητές πακέτων , bytes και ροών ,καθώς και το *flow-hash-id-map list* .

Κεφάλαιο 7
Συμπεράσματα και άλλα έργα για τον ODL-ελεγκτή

Ο OpenDaylight ελεγκτής διευκολύνει την συνεχόμενη ανάπτυξη των SDN διότι αποτελεί έναν SDN ελεγκτή ανοιχτού κώδικα. Έτσι, μπορούν να αναλάβουν αρκετοί, ανεξάρτητοι αλλά και μη προγραμματιστές, μία επιταχυνόμενη αλλά και μεγάλου εύρους ανάπτυξη εφαρμογών. Τα τελευταία τρία χρόνια, ο ODL, έχει αναπτυχθεί δυναμικά και έχει προσεγγίσει έναν μεγάλο αριθμό δικτυακών πρωτοκόλλων μέσα από μία λίστα έργων η οποία αυξάνεται συνεχώς. Με τις δυο εφαρμογές που αναφέρθηκαν, στην εργασία, είναι προφανές πως η διαχείριση δικτύων μπορεί να γίνει περισσότερο ευέλικτη ανεξαρτήτως το μέγεθος του δικτύου, αρκεί πάντα να υπάρχουν οι κατάλληλοι πόροι.

Η ανάπτυξη των εφαρμογών, όμως, απαιτεί ένα αρκετά προχωρημένο επίπεδο κυρίως στον τομέα του προγραμματισμού διότι χρειάζεται έναν λεπτομερή και απαιτητικό προσεγγιστικό σχεδιασμό των παλαιότερων δικτυακών εφαρμογών με τους νέο τρόπο λειτουργίας των δικτύων. Τα έργα είναι μεγάλα, δύσκολα αλλά ως τώρα όχι ακατόρθωτά και μπορούν να προσδιοριστούν ως επαναστατικά και ελπιδοφόρα για τον κόσμο των εταιρών κατασκευής δικτύων.

Η **CISCO** για παράδειγμα, μία από της κολοσσούς εταιρίες στο χώρο της ανάπτυξής δικτύων αλλά και πόσο μάλιστα στην κατασκευή δικτυακών συσκευών, έχει ενταχθεί στον κόσμο του SDN χρησιμοποιώντας τον OpenDaylight ελεγκτή από την έκδοσης Helium και συμμετέχει στην ανάπτυξή του μέσω του ***DevNet***, το οποίο αποτελεί την βασική κοινότητα έρευνας και ανάπτυξης της εταιρίας.

Παρακάτω παρουσιάζονται κάποια από έργα τα οποία έχουν υλοποιηθεί, αναπτύσσονται και βρίσκονται σε πειραματικό στάδιο. Τα έργα αυτά παίζουν σημαντικό ρόλο στην αύξηση και στην απόδοση των δυνατοτήτων και των λειτουργιών του OpenDaylight ελεγκτή καθώς και για SDN.

1. Λίστα με τα έργα που έχουν ήδη υλοποιηθεί

Όνομα	Περιγραφή	Όνομα Karaf
Authentication	Αυθεντικοποιεί την υποστήριξη της Apache Shiro ομοσπονδίας	odl-aaa-shiro
BGP	Παρέχει υποστήριξη για Border Gateway Protocol σαν πηγή στο επίπεδο 3 μίας δικτυακής τοπολογίας	odl-bgpcep-bgp
Fabric as a Service (FaaS)	Δημιουργεί ένα κοινό *abstraction layer* στην κορυφή του φυσικού δικτύου έτσι ώστε τα *Northbound APIs* να μπορούν να αντιστοιχιστούν εύκολα με το δίκτυο σαν σταθερές συσκευές που παραμετροποιούνται εύκολα	odl-faas-all
OVSDB	Υλοποιεί το πρωτόκολλο OVSDB (RFC 7047), καθώς και τα Open vSwitch	odl-ovsdb-southbound-impl-ui

	schema (OVSDB Southbound και hw_vtep schema (OVSDB HWVTEP Southbound)	odl-ovsdb-sfc-ui
Group Based Policy (GBP)	Αφορά το Group Policy και χωρίζεται σε τρεις κατηγορίες όπως το GBP UI διαδικτυακή διεπαφή), GBP FaaS renderer (ενεργοποιεί το FaaS renderer) και GBP Neutron support (παρέχει OpenStack Neutron υποστήριξη	odl-groupbasedpolicy-ofoverlay odl-groupbasedpolicyi-ui odl-groupbasedpolicy-faas odl-groupbasedpolicy-neutronmapper
Nemo Openflow	Αντιστοιχίζεται και υλοποιείται για openflow συσκευές	odl-nemo-openflow-renderer
NETCONF cover SSH	Η διαχείριση τωνNETCONF συσκευών προστατεύεται από το πρωτόκολλο SSH	odl-netconf-connector-ssh
SNMP4SDN	Ενεργοποιεί *openflow-έλεγχο* δικτυακών στοιχείων δια μέσου του SNMP	odl-snmp4sdn-all
VTN Manager	Υποστηρίζει *Virtual Tenant Networks*	odl-vtn-manager-rest

[14]

2. Λίστα με τα έργα που αναπτύσσονται

Όνομα	Περιγραφή	Όνομα Karaf
OpFlex	Παρέχει τον OpFlex agent για το Open vSwitch ο οποίος ενισχύει τη πολιτική ενός δικτύου, όπως το GBP, για τα ήδη υπάρχοντα τοπικά εικονικά μηχανήματα	n/a
NeXt	Παρέχει ένα εργαλείο για προγραμματιστές, οι οποίοι σχεδιάζουν τα APIs	n/a

[14]

3. Λίστα με τα έργα που είναι σε πειραματικό στάδιο

Όνομα	Περιγραφή	Όνομα Karaf
Authorization	Παραμετροποιεί του ρόλους εξουσιοδότησης	odl-aaa-authz
ALTO	Υποστηρίζει την βελτιστοποίηση της κίνησης των πακέτων στο επίπεδο εφαρμογών	odl-alto-coreself
CAPWAP	Ελέγχει τις ασύρματες εφαρμογές	odl-capwap-ac-restall
Clustered Authentication	Αυθεντικοποιεί το σύμπλεγμά των αποθηκευμένων δεδομένων του MD-SAL	odl-aaa-authn-mdsal-clusterall
Controller Shield	Παρέχει ασφαλείς πληροφορίες στις Northbound εφαρμογές	odl-usecplugin
IoT Data Management	Υποστηρίζει τις προδιαγραφές για M2M εφαρμογές	odl-iotdm-onem2m

YANG PUBSUB	Τοποθετεί τις νέα συνδρομές σε αποθηκευτικούς χώρους που βρίσκονται σε απομακρυσμένες συσκευές, σε υπό-δέντρα της Yang ώστε να μη χρειάζεται ο ελεγκτής να κάνει συνεχόμενες αιτήσεις	odl-yangpush-rest

[14]

Αναφορές

[1] Open Networking Foundation: Software Defined Networking (SDN) Definition. © 2011

[2] https://en.wikipedia.org/wiki/OpenFlow : OpenFlow from Wikipedia, the free encyclopedia

[3] OpenFlow Switch Speciation : Version 1.1.0 Implemented (Wire Protocol 0x02).February 28,2011

[4] https://en.wikipedia.org/wiki/OSGi : OSGi from Wikipedia, the free encyclopedia

[5] https://karaf.apache.org/manual/latest/overview.html : Overview. Last updated 2016-04-27

[6] Website of Opendaylight wiki: "OpenDaylight_Controller:MD-SAL:FAQ". Powered by MediaWiki
https://wiki.opendaylight.org/view/OpenDaylight_Controller:MD-SAL:FAQ

[7] Inocybe Technologies: Developer's Guide. Opendaylight Community, master (2014-11-13), pages 340,343,347,348. Copyright © 2014 The Linux Foundation®. All rights reserved.

[8] A. Clemm, J. Medved, & T. Tkacik (Cisco), N. Bahadur (Bracket Computing), H. Ananthakrishnan (Juniper Networks), R. Varga (Pantheon Technologies SRO): A Data Model for Network Topologies (draft-clemm-i2rs-yang-network-topo-00.txt). IEFT Tools**,** pages 6,17-24. February 14, 2014
https://tools.ietf.org/html/draft-clemm-i2rs-yang-network-topo-00

[9] Website of Opendaylight wiki: "Yang tools: Yang to Java Mapping". Powered by MediaWiki
https://wiki.opendaylight.org/view/YANG_Tools:YANG_to_Java_Mapping

[10] Release Hydrogen: Service Provider, Virtualization & Base Installation Guide

[11] Opendaylight User Guide," Helium", pages 4.

[12] Opendaylight User Guide Beryllium, pages 2,3. Feb 21,2016

[13] Opendaylight Installation Guide, pages 6,8. Feb 21,2016

[14] Opendaylight: Documentation Release Beryllium, Opendaylight Project, pages 19,20,21. Jan 20, 2017

Βιβλιογραφία

N. McKeown & Guru Parulkar (Stanford University), T. Anderson (University of Washington), H. Balakrishnan (MIT), L. Peterson & Jennifer Rexford (Princeton University) Scott Shenker (University of California), Berkeley Jonathan Turner (Washington University in St. Louis): OpenFlow: Enabling Innovation in Campus Networks March 14 ,2008

Jean Tourrilhes, Puneet Sharma, Sujata Banerjee, HP Labs. A Standards Perspective: The Evolution of SDN and OpenFlow

J. Medved, E. Warnicke, A. Tkacik. R. Varga: Developing OpenDaylight Apps with MD-SAL, Cisco, ODL Summit, February 04, 2014

C. Metz, D. Malachovsky, J Sebin: Yang User Interface (YANGUI) in OpenDaylight, ODL Summit, July 29 ,2015

M Bjorklund, Tail-f Systems: YANG – A Data Modeling Language for the Network Configuration Protocol (NETCONF). IETF, Standards Track, ISSN:2070-1721, October 2010

Srini Seetharaman: App Development Tutorial, SDN Hub. August, 2015

Opendaylight Installation Guide / User Guide / Developer Guide Beryllium, Feb 21,2016

Opendaylight: Documentation Release Beryllium, Opendaylight Project, Jan 20, 2017

Linux Foundation: Opendaylight Community: Developer's Guide. Master (2014-11-13)

Website of Opendaylight encyclopedia: "Karaf Step by Step", "OpenDaylight Controller: MD-SAL", "Yang Tools", *https://wiki.opendaylight.org*. Powered by MediaWiki

Website of Opendaylight community:"Lithium - The third release", "ODL Beryllium (Be) – The forth Release of Opendaylight", *http://www.opendaylight.org.*
© Copyright 2016, Opendaylight Project.

Website of Openflow archives. *http://archive.openflow.org/wp/documents* Copyright © 2011. All rights reserved. Powered by MediaWiki and WordPress.

Sumit Arora: Opendaylight SDN Controller, Slideshare, November 11, 2014 *http://www.slideshare.net/esumit/opendaylight-sdn-controller?related=1*

http://sdnhub.org/tutorials/opendaylight/ Website of SDN Hub. Copyright © 2014, Terrifico Theme powered by WordPress

Website of Mininet: "Mininet Walkthrough", *http://mininet.org/walkthrough/*, Copyright © 2017 - Mininet Team - Powered by Octopress

Brian Linkleter: Set up the Mininet network simulator, September 17,2013 *http://www.brianlinkletter.com/set-up-mininet*