ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΥΦΥΗ ΠΡΟΓΡΑΜΜΑΤΙΖΟΜΕΝΑ ΔΙΚΤΥΑ (SOFTWARE DEFINED NETWORKS) ΣΤΟ ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ (INTERNET OF THINGS)

Διπλωματική Εργασία

του

Γεωργίου Φατσή

Θεσσαλονίκη, Ιούνιος 2020

ΕΥΦΥΗ ΠΡΟΓΡΑΜΜΑΤΙΖΟΜΕΝΑ ΔΙΚΤΥΑ (SOFTWARE DEFINED NETWORKS) ΣΤΟ ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ (INTERNET OF THINGS)

Φατσής Γεώργιος

Εκπαιδευτικός Ηλεκτρολόγος Μηχανικός και Ηλεκτρολόγος Μηχανικός Τ.Ε., Α.Σ.ΠΑΙ.Τ.Ε.,2014

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Μαμάτας Ελευθέριος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30/06/2020

Μαμάτας Ελευθέριος	Πετρίδου Σοφία	Ψάννης Κωνσταντίνος
.....	.....	.....

Φατσής Γεώργιος

.....

Περίληψη

Καθώς το Διαδίκτυο των Αντικειμένων (IoT) αναπτύσσεται συνδέονται όλο και περισσότερες συσκευές. Το όλο και μεγαλύτερο πλήθος των συσκευών και εφαρμογών που διασυνδέονται στο διαδίκτυο δημιουργεί νέα προβλήματα τα οποία είναι: το πλήθος των συσκευών, η διαφορετικότητα των συσκευών και των εφαρμογών για τις οποίες χρησιμοποιούνται, η δυνατότητα των συσκευών να μετακινούνται, η ασφάλεια και η εξασφάλιση της απρόσκοπτης λειτουργίας. Μια από τις νέες τεχνολογίες η οποία έχει εισαχθεί στα δίκτυα είναι τα Ευφυή Προγραμματιζόμενα Δίκτυα (Software Defined Networks) ή όπως είναι πιο γνωστά ως SDN, ως Software Defined Networks ή SDN. Τα οποία αποτελούν μια δυναμική προσέγγιση της διαχείρισης του δικτύου η οποία επιτρέπει την προγραμματιστικά αποδοτική διαμόρφωση του δικτύου προκειμένου να αυξηθεί η απόδοση και η προσαρμοστικότητα του δικτύου. Επειδή τα ευφυή προγραμματιζόμενα δίκτυα (SDN) μας επιτρέπουν να διαχειριζόμαστε δυναμικά το πλήθος των συσκευών, την διαδικτυακή κίνηση την οποία δημιουργούν και την διαφορετικότητα των συσκευών υπάρχει όλο και μεγαλύτερο ενδιαφέρον για την εισαγωγή των SDN στο Διαδίκτυο των Αντικειμένων. Υπάρχουν πολλές διαφορετικές προσεγγίσεις για να καταφέρουν να εισαχθούν τα SDN στο Διαδίκτυο των Αντικειμένων, δύο από τις οποίες είναι το CORAL και το SDN-WISE.

Σκοπός της παρούσας εργασίας είναι να πραγματοποιηθεί μια μελέτη όπου θα παρουσιαστεί η εισαγωγή των Ευφυών προγραμματιζόμενων δικτύων στο Διαδίκτυο των Αντικειμένων. Επίσης, θα παρουσιάσουμε τον τρόπο με τον οποίο προσεγγίζουν την εισαγωγή των SDN στο Διαδίκτυο των Αντικειμένων διαφορετικές υλοποιήσεις, όπως το CORAL και το SDN WISE. Στόχος της συγκεκριμένης εργασίας είναι πραγματοποιηθεί μια σύγκριση μεταξύ των CORAL και SDN-WISE προκειμένου να εξαχθούν χρήσιμα συμπεράσματα σχετικά με τις διαφορετικές προσεγγίσεις τις οποίες ακολουθούν οι δύο αυτές υλοποιήσεις.

Λέξεις Κλειδιά:

SDN, IoT, SDN controller, Contiki OS, Cooja, SDN-WISE, CORAL

Abstract

As the Internet of Things (IoT) develops, more and more devices are connected. The increasing number of devices and applications that are interconnected creates new issues some of these issues are: the growing number of devices, the diversity of devices and applications, the mobility of the sensor Nodes, Security and ensuring uninterrupted operation. One of the new technologies that have been introduced into the networks is Software-Defined Networks, or as they are more commonly known as SDN. Software-Defined Networks establishes a dynamic approach to network management that allows programmatically efficient network configuration in order to increase network performance and adaptability. Because Software Defined Networks (SDN) will enable us to dynamically manage the number of devices, the internet traffic they create, and the diversity of devices, there is an increasing interest in the introduction of SDN on the Internet of Things. There are many different approaches to adapt SDN into the Internet of Things, two of which are CORAL and SDN-WISE.

The purpose of this paper is to carry out a study where we will present the introduction of Software Defined Networks at the Internet of Things. We will also show how different implementations such as CORAL and SDN WISE approach the adaption of SDN on the Internet of Things. This work aims to present a comparison between CORAL and SDN-WISE to extract useful conclusions about the different approaches followed by these two implementations.

Keywords:

SDN, IoT, SDN controller, Contiki OS, Cooja, SDN-WISE, CORAL

Πρόλογος – Ευχαριστίες

Η παρούσα διπλωματική εργασία είναι προϊόν έρευνας η οποία πραγματοποιήθηκε στα πλαίσια του Προγράμματος Μεταπτυχιακών σπουδών και δεν θα μπορούσαν να υλοποιηθούν χωρίς την καθοριστική συμβολή του επιβλέποντα καθηγητή κύριου Μαμάτα Ελευθέριου, καθώς επίσης και του Υποψήφιου Διδάκτορα Τρύφωνα Θεοδώρου.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

1 Εισαγωγή

1.1 Πρόβλημα – Σημαντικότητα του θέματος

Στο Διαδίκτυο των Αντικειμένων (IoT) το όλο και μεγαλύτερο πλήθος των συσκευών που διασυνδέονται στο διαδίκτυο εισάγουν νέα προβλήματα τα οποία χρήζουν αντιμετώπισης. Τα προβλήματα αυτά είναι: το πλήθος των συσκευών, η διαφορετικότητα των συσκευών και των εφαρμογών για τις οποίες χρησιμοποιούνται, η δυνατότητα των συσκευών να μετακινούνται, η ασφάλεια και η εξασφάλιση της απρόσκοπτης λειτουργίας.

Αναλύοντας τα παραπάνω προβλήματα, παρατηρούμε ότι οι ίδιες οι συσκευές παρουσιάζουν διαφορετικά τεχνικά χαρακτηριστικά και χρησιμοποιούν διαφορετικά πρωτόκολλα προκειμένου να μπορούν να επικοινωνήσουν [1]. Εμφανίζουν όμως και κοινά χαρακτηριστικά τα οποία έχουν να κάνουν με φυσικούς περιορισμούς στους οποίους υπόκεινται, όπως το μικρό μέγεθος της μνήμης, η μικρή επεξεργαστική ισχύς, ο πιο βασικός τους όμως περιορισμός είναι στην κατανάλωση ενέργειας [2]. Από τα παραπάνω συμπεραίνουμε ότι οι συσκευές αυτές έχουν διαφορετικές ιδιότητες τις οποίες εκμεταλλευόμαστε προκειμένου να καταφέρουμε να πετύχουμε το προσδοκώμενο αποτέλεσμα στην εκάστοτε εφαρμογή για την οποία τις χρησιμοποιούμε πχ. Έξυπνη Πόλη ή Έξυπνο Σπίτι.

Το πλήθος των συσκευών επηρεάζεται άμεσα από την εφαρμογή την οποία υλοποιούμε. Για παράδειγμα για τον έλεγχο της κυκλοφορίας σε μία έξυπνη πόλη χρειαζόμαστε πάρα πολλούς ασύρματους αισθητήρες, πολλοί από τους οποίους θα είναι κινητοί. Εν αντιθέσει με τον έλεγχο του σακχάρου στο αίμα ενός ανθρώπου που μας αρκεί ένας αισθητήρας. Επίσης, το πλήθος των συσκευών έχει σημαίνοντα ρόλο στην εξέλιξη των δικτύων καθώς οι όλο και περισσότερες διασυνδεμένες συσκευές εξωθούν στα άκρα την υπάρχουσα υποδομή του διαδικτύου και επιφέρουν τεράστιες αλλαγές στα πρωτόκολλα και τον τρόπο λειτουργίας τους (εισαγωγή του IPv6) . Οι αλλαγές αυτές προκύπτουν καθώς οι συσκευές αυτές εισάγουν νέα χαρακτηριστικά και νέες δυνατότητες τις οποίες πρέπει το υπάρχον δίκτυο να μπορεί να υποστηρίζει. Έτσι παρατηρούμε ότι τα δίκτυα επικοινωνιών εξελίσσονται ραγδαία προκειμένου να υποστηρίξουν τις νέες συσκευές και τα νέα χαρακτηριστικά τα οποία εισάγουν π.χ. κινητοί κόμβοι για τα οποία τα υπάρχοντα δίκτυα πρέπει να εξελιχθούν προκειμένου να τα υποστηρίξουν [3]. Μέρος της εξέλιξης αυτής είναι διάφορες νέες τεχνολογίες οι οποίες καλούνται να επιλύσουν τα προβλήματα που δημιουργούνται, καθώς και να εξελίξουν την ήδη υπάρχουσα υποδομή.

Μια από τις νέες τεχνολογίες η οποία έχει εισαχθεί και χρησιμοποιείται ευρέως είναι τα Ευφυή Προγραμματιζόμενα Δίκτυα (Software Defined Networks) ή όπως είναι πιο γνωστά ως SDN. Ως Software Defined Networks ή SDN ορίζεται μία δυναμική προσέγγιση της διαχείρισης του δικτύου η οποία επιτρέπει την προγραμματιστικά αποδοτική διαμόρφωση του δικτύου προκειμένου να αυξηθεί η απόδοση και η προσαρμοστικότητα του δικτύου [3]. Τα ευφυή προγραμματιζόμενα δίκτυα (SDN) μας επιτρέπουν να διαχειριστούμε πιο εύκολα το πλήθος των συσκευών και της διαδικτυακής κίνησης την οποία δημιουργούν. Επίσης, μας επιτρέπουν να διαχειριστούμε την διαφορετικότητα των συσκευών. Ένα άλλο χαρακτηριστικό των SDN και της χρησιμότητας τους είναι ότι έχουν την δυνατότητα να προχωρούν σε αλλαγές στη δρομολόγηση της διαδικτυακής κίνησης στα ενσύρματα δίκτυα χωρίς να απαιτείται κάποια αλλαγή ή τροποποίηση στο λειτουργικό σύστημα των συσκευών που δρομολογούν την διαδικτυακή κίνηση. Στο παρελθόν μια τέτοια αλλαγή θα απαιτούσε ανθρώπινη παρέμβαση σε όλες τις εμπλεκόμενες συσκευές, όμως με την χρήση των SDN κατέστη δυνατόν να γίνονται τέτοιες αλλαγές αυτόματα από τους ελεγκτές [2]. Για όλους τους παραπάνω λόγους υπάρχει όλο και μεγαλύτερο ενδιαφέρον για την εισαγωγή των SDN στο Διαδίκτυο των Αντικειμένων.

Προκειμένου να πραγματοποιηθεί η ενσωμάτωση των SDN στο Διαδίκτυο των Αντικειμένων έχουν παρουσιαστεί πολλές διαφορετικές προσεγγίσεις για το πώς θα επιτευχθεί η ζεύξη των αυτών δύο τεχνολογιών αιχμής. Η κάθε προσέγγιση προσπαθεί να καλύψει όσο το δυνατόν καλύτερα τις απαιτήσεις του Διαδικτύου των Αντικειμένων, καθώς και να εισάγει μέσω του SDN νέα χαρακτηριστικά στο δίκτυο έτσι ώστε να πετύχει καλύτερα αποτελέσματα. Κάποια από τα χαρακτηριστικά είναι: QoS, QoE, ενισχυμένες δυνατότητες στη διαχείριση των δικτύων, εισαγωγή ευφυών τεχνολογιών και αλγορίθμων για την αποδοτικότερη διαχείριση των πόρων του συστήματος.

Η εισαγωγή και η χρήση των ευφυών προγραμματιζόμενων δικτύων (SDN) στο Διαδίκτυο των Αντικειμένων (IoT) είναι ένα ζήτημα το οποίο απασχολεί έντονα όλη την ακαδημαϊκή και τεχνολογική κοινότητα καθώς πρόκειται για το πάντρεμα δύο τεχνολογιών αιχμής οι οποίες είναι πολλά υποσχόμενες. Έτσι, η εισαγωγή των SDN στο IoT θα μας επιτρέψει να αναπτύξουμε εφαρμογές οι οποίες θα βελτιώσουν δραματικά την ανθρωπότητα [4]. Σχετικά με το ζήτημα αυτό υπάρχουν πολλές προσεγγίσεις και πολλές ερευνητικές προτάσεις κάποιες από τις οποίες στοχεύουν να εξελίξουν το Διαδίκτυο των Αντικειμένων έτσι ώστε να το φέρουν πιο κοντά στα SDN και άλλες στο να εισάγουν την χρήση των

SDN στο Διαδίκτυο των Αντικειμένων προκειμένου να καταφέρουν να αλλάξουν ριζικά τον τρόπο με τον οποίο λειτουργεί το δίκτυο [5].

Κάποιες από τις υλοποιήσεις στοχεύουν στην εξέλιξη του IoT έτσι ώστε να έρθει πιο κοντά στα SDN, μία από αυτές είναι το μSDN, όπου οι Baddeley, Nejabati [6] παρουσιάζουν ένα πολύ ελαφρύ πλαίσιο εμπνευσμένο από το SDN το οποίο βασίζεται στην κεντρική διαχείριση των δικτύων IEEE 802.15.4 . Η πρόταση αυτή είναι αρθρωτή και έχει σχεδιαστεί έτσι ώστε να υποστηρίζει SDN με χρήση σε IPv6 δίκτυα με υποστήριξη του δικτυακού πρωτοκόλλου RPL. Πραγματοποιεί κεντρικό έλεγχο στο δίκτυο και προκειμένου να παρέχει Quality Of Service για τις σημαντικές λειτουργίες του δικτύου. Μία παρόμοια λύση προτείνουν και οι Esteban Municio, Johann Marquez-Barja η οποία είναι το Whisper [7]. Το Whisper είναι μια υλοποίηση η οποία μπορεί να λειτουργήσει με το δικτυακό πρωτόκολλο RPL καθώς και με το 6TiSCH. Το Whisper χρησιμοποιεί τον ελεγκτή Whisper ο οποίος απομακρυσμένα ελέγχει τους κόμβους χρησιμοποιώντας μηνύματα δρομολόγησης και προγραμματισμού, τα μηνύματα αυτά είναι προσαρμοσμένα έτσι ώστε να λειτουργούν είτε σε RPL είτε σε 6TiSCH.

Όμως, όπως έχουμε ήδη αναφέρει άλλες υλοποιήσεις προσπαθούν να εισάγουν την χρήση των SDN στο Διαδίκτυο των Αντικειμένων με σκοπό να αλλάξουν ριζικά τον τρόπο με τον οποίο λειτουργεί το δίκτυο. Μία από της υλοποιήσεις αυτές είναι: το CORAL, το οποίο προσπαθεί να προσαρμόσει την τεχνολογία των SDN για εφαρμογή στα δίκτυα IPv6 τα οποία χρησιμοποιούν πρωτόκολλο επικοινωνίας RPL. Το CORAL είναι μια πρόταση η οποία διαχωρίζει το επίπεδο ελέγχου από το επίπεδο των δεδομένων. Προκειμένου να το πετύχει αυτό χρησιμοποιεί έναν ελεγκτή SDN ο οποίος προσπαθεί να ρυθμίσει τις παραμέτρους του πρωτοκόλλου δυναμικά, έτσι ώστε να πετύχει τη βέλτιστη λειτουργία στο πρωτόκολλο και να μειώσει την επικοινωνία μεταξύ κόμβων και ελεγκτή προκειμένου να πετύχει στόχους όπως η εξοικονόμηση ενέργειας στους κόμβους, αλλά και η υποστήριξη ετερογενών κόμβων [4]. Μια διαφορετική πρόταση είναι το SDN-WISE το οποίο βασίζεται στο SDWN [8] και το επεκτείνει έτσι ώστε να καταφέρει να χρησιμοποιεί stateful πίνακες δρομολόγησης, καθώς και Δυναμική δρομολόγηση προκειμένου να μειώσει τη συνεχή επικοινωνία μεταξύ ελεγκτή και κόμβων [9]. Όπως και η επέκταση του SDN-WISE η οποία προσπαθεί να πετύχει την διαχείρηση των SDN-IoT μαζί με κλασικά δίκτυα. Προκειμένου να το πετύχει αυτό ενσωματώνεται η χρήση του SDN-WISE με το Open Network Operating System (ONOS) [2].

Όπως βλέπουμε τα τελευταία χρόνια έχουν πραγματοποιηθεί αρκετές έρευνες και έχουν δημιουργηθεί διάφορες υλοποιήσεις για την εισαγωγή των SDN στο Διαδίκτυο των Αντικειμένων. Για αυτό το λόγο θεωρούμε σκόπιμο να πραγματοποιηθεί μία συγκριτική μελέτη μεταξύ των διαφορετικών υλοποιήσεων. Έτσι, στα πλαίσια της παρούσας διατριβής θα επικεντρωθούμε στην σύγκριση δύο διαφορετικών υλοποιήσεων, του CORAL, καθώς και του SDN-WISE. Οι δύο συγκεκριμένες υλοποιήσεις επιλέχθηκαν μεταξύ των υπολοίπων υλοποιήσεων καθώς και οι δύο έχουν σαν στόχο να αλλάξουν ριζικά τον τρόπο με τον οποίο λειτουργούν τα δικτυακά πρωτόκολλα. Επίσης, ένας άλλος λόγος είναι ότι το SDN-WISE είναι μια stateful λύση η οποία εισάγει τη χρήση διαφορετικών ελεγκτών, καθώς επίσης και των NFV (ONOS) στο Διαδίκτυο των Αντικειμένων. Επίσης, το CORAL από την πλευρά του παρέχει δυνατότητες για εφαρμογές οι QoE (Quality of Experience) για τους χρήστες και QoS (Quality Of Service) για τις εφαρμογές που αναπτύσσονται στο Διαδίκτυο των Αντικειμένων [10]. Τέλος, η επιλογή των δύο αυτών υλοποιήσεων πραγματοποιήθηκε επειδή η υλοποίηση του SDN-WISE αποτέλεσε πηγή έμπνευσης για την υλοποίηση του CORAL [4].

1.2 Σκοπός – Στόχοι

Σκοπός της παρούσας εργασίας είναι να πραγματοποιηθεί μια μελέτη όπου θα παρουσιαστεί η εισαγωγή των Ευφυών προγραμματιζόμενων δικτύων στο Διαδίκτυο των Αντικειμένων. Επίσης, σκοπός είναι να παρουσιαστεί ο τρόπος με τον οποίο προσεγγίζουν την εισαγωγή των SDN στο Διαδίκτυο των Αντικειμένων διαφορετικές υλοποιήσεις όπως το CORAL και το SDN WISE. Τέλος, στόχος της συγκεκριμένης εργασίας είναι πραγματοποιηθεί μια σύγκριση μεταξύ των CORAL και SDN-WISE προκειμένου να εξαχθούν χρήσιμα συμπεράσματα σχετικά με τις διαφορετικές προσεγγίσεις και τα χαρακτηριστικά που παρουσιάζει η κάθε υλοποίηση ξεχωριστά.

1.3 Συνεισφορά

Με βάση την υπάρχουσα βιβλιογραφία και τις διαφορετικές υλοποιήσεις οι οποίες έχουν πραγματοποιηθεί προχωρήσαμε σε μελέτη των διαφορετικών υλοποιήσεων και εντοπίσαμε τις διαφορές, καθώς και τις ομοιότητες τους. Επίσης, διαπιστώσαμε τον τρόπο με τον οποίο οι ελεγκτές και τα πρωτόκολλα διαμορφώνουν τα χαρακτηριστικά του δικτύου στο οποίο χρησιμοποιούνται. Οι διαφορές και οι ομοιότητες οι οποίες εμφανίζουν είναι το αντικείμενο μελέτης της παρούσας εργασίας. Τέλος, μέσα από την χρήση των δύο υλοποιήσεων προέκυψαν επιπλέον συμπεράσματα, καθώς επίσης και κάποια script τα

οποία αυτοματοποιούν την διαδικασία της εγκατάστασης και εκτέλεσης των δύο υλοποιήσεων.

1.4 Βασική Ορολογία

Υπάρχουν διάφοροι όροι οι οποίοι θα χρησιμοποιηθούν στην παρούσα εργασία και τους οποίους οφείλουμε να αναφέρουμε:

- **Internet of Things (IoT):** Διαδίκτυο των Αντικειμένων, είναι η διασύνδεση μέσω του Διαδικτύου με υπολογιστικές συσκευές με μικρή επεξεργαστική δυνατότητα σε καθημερινά αντικείμενα που τους επιτρέπει να στέλνουν και να λαμβάνουν δεδομένα. Αν επεκτείνουμε την έννοια του IoT μπορούμε να θεωρήσουμε ότι είναι η ενοποίηση του πεδίου των πληροφοριών με το φυσικό χώρο έτσι ώστε οι χρήστες να μπορούν να έχουν πρόσβαση στις πληροφορίες πιο αποτελεσματικά [11].
- **Software Defined Networks (SDN):** Ευφυή προγραμματιζόμενα δίκτυα, είναι τα δίκτυα στα οποία χρησιμοποιούμε έναν προγραμματιζόμενο ελεγκτή προκειμένου να ελέγξουμε και διαχειριστούμε την διαδικτυακή κίνηση. Ουσιαστικά διαχωρίζουμε το δίκτυο σε δύο επίπεδα το επίπεδο ελέγχου (Control Plane) και το επίπεδο των δεδομένων (Data Plane). Επίσης υπάρχει η κεντρική 'οντότητα' που ελέγχει το δίκτυο η οποία όμως δεν ανήκει στο δίκτυο [12].
- **Controller:** Ελεγκτής ουσιαστικά είναι ο «εγκέφαλος» του δικτύου είναι η εφαρμογή η οποία βρίσκεται στο επίπεδο ελέγχου και καθορίζει τον τρόπο με τον οποίο θα διαχειριστεί τα δεδομένα το επίπεδο των δεδομένων [13]. Ο ελεγκτής ουσιαστικά διαχειρίζεται τη ροή των δεδομένων μεταξύ των κόμβων του IoT δικτύου και της υπόλοιπης διαδικτυακής υποδομής [14].
- **Wireless Sensor Network (WSN)**: Είναι το Ασύρματο Δίκτυο Αισθητήρων αποτελείται από διασκορπισμένους αυτόνομους αισθητήρες για την παρακολούθηση φυσικών ή περιβαλλοντολογικών συνθηκών όπως η θερμοκρασία, ο ήχος, η ατμοσφαιρική πίεση κτλ. και μέσω συνεργασίας να μεταφέρει τα δεδομένα μέσω του δικτύου σε μια συγκεκριμένη τοποθεσία.

1.5 Διάρθρωση της μελέτης

Η διάρθρωση της παρούσας μελέτης παρουσιάζει στο Κεφάλαιο 2 εργασίες σχετικές με το αντικείμενο της έρευνας. Το Κεφάλαιο 3 παρουσιάζει τον τρόπο με τον οποίο τα SDN εισάγονται στο Διαδίκτυο των Αντικειμένων, καθώς επίσης και παραδείγματα εφαρμογών στα οποία χρησιμοποιούνται οι εφαρμογές τις οποίες μπορούν να υλοποιήσουν. Επίσης, στο Κεφάλαιο 3 παρουσιάζονται οι υλοποιήσεις του CORAL και του SDN-WISE, καθώς και κάποιες οδηγίες σχετικά με εγκατάσταση και εκτέλεση των δύο υλοποιήσεων. Στο Κεφάλαιο 4 παρουσιάζεται η σύγκριση μεταξύ των 2 υλοποιήσεων. Τέλος, το Κεφάλαιο 5 αποτελεί το κλείσιμο της παρούσας εργασίας όπου παρουσιάζονται τα αποτελέσματα της εργασίας.

2 Βιβλιογραφική Επισκόπηση – Θεωρητικό Υπόβαθρο

Το ζήτημα της εισαγωγής των SDN στο Διαδίκτυο των Αντικειμένων αποτελεί ένα από τα πιο σημαντικά θέματα που απασχολεί την ερευνητική κοινότητα. Κατά αυτό τον τρόπο υπάρχουν διαφορετικές υλοποιήσεις η κάθε μια από τις οποίες δημιουργήθηκε με βάση την οπτική των ανθρώπων που την υλοποίησαν. Αναλύοντας της βιβλιογραφία παρατηρούμε ότι υπάρχουν αρκετές υλοποιήσεις σχετικά με το ζήτημα αυτό οι οποίες αξιοποιούν με διαφορετικό τρόπο τους ελεγκτές που χρησιμοποιούν. Σε κάποιες από τις υλοποιήσεις οι ελεγκτές χρησιμοποιούνται προκειμένου να διαχωρίσουν το επίπεδο δεδομένων από το επίπεδο ελέγχου, ενώ άλλες υλοποιήσεις προσπαθούν να βελτιστοποιήσουν τις παραμέτρους του πρωτοκόλλου το οποίο χρησιμοποιούν. Οι διαφορετικές υλοποιήσεις δεν εστιάζουν μόνο στον τρόπο με τον οποίο θα γίνει πραγματικότητα η χρήση των SDN στο Διαδίκτυο των Αντικειμένων, αλλά στοχεύουν και στο να επιλύσουν ζητήματα τα οποία προκύπτουν κατά την εισαγωγή των SDN στο Διαδίκτυο των Αντικειμένων. Ένα τέτοιο ζήτημα είναι το overhead που εισάγουν τα SDN στο δίκτυο από τη συνεχή επικοινωνία μεταξύ ελεγκτή και κόμβων. Έτσι, μπορούμε να ξεχωρίσουμε μεταξύ των υλοποιήσεων μοναδικές προτάσεις όπως η P2P επικοινωνία των κόμβων, η χρήση κόμβων ως μεσολαβητή μεταξύ ελεγκτή και των υπολοίπων κόμβων, η δυναμική ρύθμιση των παραμέτρων του πρωτόκολλου που χρησιμοποιούν. Κάποιες από τις προτάσεις αυτές είναι οι παρακάτω:

- **μSDN:** Είναι μια υλοποίηση η οποία προσπαθεί να μειώσει το overhead που δημιουργεί το SDN, εφαρμόζοντας κεντρική διαχείριση στο δίκτυο IEEE 802.15.4 . Η πρόταση αυτή είναι αρθρωτή και έχει σχεδιαστεί έτσι ώστε να υποστηρίζει SDN με χρήση σε IPv6 δίκτυα με υποστήριξη του δικτυακού πρωτοκόλλου RPL. Πραγματοποιεί κεντρικό έλεγχο στο δίκτυο και προκειμένου να παρέχει Quality Of Service για τις σημαντικές λειτουργίες του δικτύου [6].
- **Whisper:** Το Whisper είναι μια υλοποίηση η οποία μπορεί να λειτουργήσει με το δικτυακό πρωτόκολλο RPL καθώς και με το 6TiSCH . Το Whisper χρησιμοποιεί τον ελεγκτή Whisper ο οποίος απομακρυσμένα ελέγχει τους κόμβους χρησιμοποιώντας μηνύματα δρομολόγησης και προγραμματισμού, τα μηνύματα αυτά είναι προσαρμοσμένα έτσι ώστε να λειτουργούν είτε σε RPL είτε σε 6TiSCH [7].

- Οι Hai Huang1, Jiping Zhu1, Lei Zhang στο [11] προτείνουν μια νέα οπτική πάνω στο ζήτημα αυτό, να αλλάξουν τον τρόπο με τον οποίο λειτουργεί το δίκτυο έτσι ώστε να εφαρμόσουν Machine-to-Machine (M2M) τεχνικές προκειμένου να δημιουργήσουν P2P επικοινωνία μεταξύ των κόμβων, έτσι ώστε η μετάδοση των δεδομένων να μην βασίζεται σε πύλες (Gateways) .
- **Software Defined Wireless Network (SDWN)** : είναι μια πρόταση η οποία θέτει μια σειρά από τεχνικές προδιαγραφές και απαιτήσεις οι οποίες πρέπει να καλύπτονται προκειμένου να μπορεί να λειτουργήσει σωστά ένα SDN πρωτόκολλο [8].
- **SDN-WISE:** Μια διαφορετική πρόταση είναι το SDN-WISE το οποίο βασίζεται στο SDWN και το επεκτείνει έτσι ώστε να καταφέρει να χρησιμοποιεί stateful πίνακες δρομολόγησης και Δυναμική δρομολόγηση. Στόχος της χρήσης των δύο αυτών δυναμικών προσεγγίσεων είναι να μειώσει τη συνεχή επικοινωνία μεταξύ ελεγκτή και κόμβων [9]. Βασικός στόχος της επέκταση αυτή του SDWN είναι να πετύχει την ενσωμάτωση των SDN-IoT σε περιβάλλοντα SDN. Προκειμένου να πετύχει ακόμη καλύτερα αποτελέσματα, καθώς και να προσφέρει νέες δυνατότητες ενσωματώνεται η χρήση του SDN-WISE με το Open Network Operating System (ONOS) [2].
- **CORAL:** Το CORAL προσπαθεί να προσαρμόσει την τεχνολογία των SDN για εφαρμογή στα δίκτυα IPv6 τα οποία χρησιμοποιούν πρωτόκολλο επικοινωνίας RPL. Το CORAL είναι μια πρόταση η οποία διαχωρίζει το επίπεδο ελέγχου από το επίπεδο των δεδομένων. Προκειμένου να το πετύχει αυτό χρησιμοποιεί έναν ελεγκτή SDN ο οποίος προσπαθεί να ρυθμίσει τις παραμέτρους του πρωτοκόλλου δυναμικά έτσι ώστε να πετύχει καλύτερη λειτουργία στο πρωτόκολλο και να πετύχει μείωση της επικοινωνίας μεταξύ κόμβων και ελεγκτή, καθώς και να πετύχει στόχους όπως η εξοικονόμηση ενέργειας στους κόμβους, αλλά και η υποστήριξη ετερογενών κόμβων [4].
- **SDWSN:** Είναι μια πρόταση η οποία προτείνει μία αρχιτεκτονική SDN σε όλα τα επιμέρους κομμάτια τα οποία αποτελούν ένα δίκτυο IoT έτσι ώστε να μπορεί να υποστηρίξει την εισαγωγή του SDN. Η πρόταση αυτή προτείνει νέους κόμβους (VSensor), ελεγκτές (SDIoT controller), πρωτόκολλο επικοινωνίας με τον ελεγκτή (SFlow) τα οποία αποτελούν την βάση της πρότασης αυτής [12].

- **ESD-WSN:** είναι μία αρχιτεκτονική η οποία έχει σαν στόχο να εκμεταλλευτεί στο έπακρο τα θετικά στοιχεία των SDN και να αποφύγει τα αρνητικά στοιχεία που έχουν τα SDN, όπως μεγάλο Overhead στο δίκτυο. Για να το πετύχει αυτό ο ελεγκτής επιλέγει δυναμικά κάποιους κόμβους και τους χρησιμοποιεί σαν μεσολαβητή. Σκοπός της λειτουργίας αυτής είναι ο κόμβος μεσολαβητής να ελέγχει την κυκλοφορία των δεδομένων μέσα στο δίκτυο [10].
- Οι Zhijing Qin, Grit Denker στο [13] έχοντας ως βάση το: Multinetwork Information Architecture (MINA) το επεκτείνουν με την εισαγωγή ενός IoT SDN ελεγκτή ο οποίος υποστηρίζει εντολές οι οποίες του επιτρέπουν να αλλάζει τον προγραμματισμό των ροών και τον τρόπο με τον οποίο επικοινωνούν οι κόμβοι. Επίσης, ο συγκεκριμένος ελεγκτής βελτιστοποιεί την διαθεσιμότητα των συσκευών που υποστηρίζει.
- Οι Hai Anh TRAN, Duc TRAN στο [3] προτείνουν μια SDN αρχιτεκτονική η οποία βασίζεται σε έναν κεντρικό ελεγκτή ο οποίος σκοπό έχει να παρατηρεί και να προσαρμόζεται δυναμικά στην αλλαγές που εμφανίζονται στα ετερογενή δίκτυα τα οποία διαχειρίζεται. Επίσης, ο Ελεγκτής δίνει την δυνατότητα στον χρήστη να διαχειρίζεται τον προγραμματισμό των ροών.

Όπως βλέπουμε υπάρχουν αρκετά διαφορετικές υλοποιήσεις οι οποίες ακολουθούν διαφορετικές μεθοδολογίες και θέτουν διαφορετικούς στόχους για να πετύχουν την όσο το δυνατόν πιο αρμονική εισαγωγή των SDN στο Διαδίκτυο των Αντικειμένων. Για το λόγο αυτό στην παρούσα εργασία θα επικεντρωθούμε στην σύγκριση δύο διαφορετικών προσεγγίσεων του CORAL, καθώς και του SDN-WISE. Η επιλογή αυτή έγινε καθώς οι δύο λύσεις αυτές παρουσιάζουν αρκετά κοινά χαρακτηριστικά, αλλά και μεγάλες διαφορές. Επίσης, καθώς και οι δύο υλοποιήσεις αλλάζουν τον τρόπο με τον οποίο λειτουργούν τα δικτυακά πρωτόκολλά και θεωρήσαμε χρήσιμο να συγκρίνουμε τον τρόπο με τον οποίο κάθε μία υλοποίηση πραγματοποιεί τις αλλαγές αυτές.

3 Μεθοδολογία

Στο Διαδίκτυο των Αντικειμένων οι διαφορετικές συσκευές οι οποίες ενσωματώνονται σταδιακά στα δίκτυα ωθούν στα άκρα την υπάρχουσα υποδομή. Έτσι, η εισαγωγή των συσκευών αυτών δημιουργεί νέες προκλήσεις, τέτοιες προκλήσεις είναι: η ετερογένεια των συσκευών, οι κινητοί κόμβοι, η ασφάλεια και η αξιοπιστία της παρεχόμενης υπηρεσίας (QoS) . Η υιοθέτηση και χρήση των SDN μπορεί να μας παρέχει χρήσιμες λύσεις στις παραπάνω προκλήσεις [15].

Προκειμένου να πραγματοποιηθεί το πάντρεμα των δύο τεχνολογιών έχουν εμφανιστεί διαφορετικές υλοποιήσεις οι οποίες παρόλο που προσπαθούν να πετύχουν τον ίδιο στόχο εμφανίζουν πολύ διαφορετικά χαρακτηριστικά και διαμορφώνουν με διαφορετικό τρόπο το περιβάλλον τους. Έτσι, μπορούμε να παρατηρήσουμε τον τρόπο με τον οποίο διαφορετικοί ελεγκτές και διαφορετικά πρωτόκολλα τα οποία χρησιμοποιούνται διαμορφώνουν τα χαρακτηριστικά με τα οποία λειτουργεί το δίκτυο. Επίσης, παρατηρούμε ότι οι υλοποιήσεις προκειμένου να πετύχουν τους στόχους που θέτει το Διαδίκτυο των Αντικειμένων μεταφέρουν όλο το φόρτο της δρομολόγησης και διαχείρισης του δικτύου στους ελεγκτές. Προκειμένου να συμβεί αυτό η κάθε υλοποίηση εκμεταλλεύεται συγκεκριμένα χαρακτηριστικά των ελεγκτών, των πρωτοκόλλων και των κόμβων που χρησιμοποιεί. Επίσης, άλλες υλοποιήσεις αλλάζουν τον τρόπο με τον οποίο πραγματοποιείται η ανίχνευση των γειτονικών κόμβων, η δρομολόγηση των πακέτων προς άλλους κόμβους, η επικοινωνία των κόμβων με τους ελεγκτές, ακόμα και ρυθμίσεις εντός του πρωτοκόλλου που χρησιμοποιούν. Όλες αυτές οι επιλογές δίνουν την δυνατότητα να λάβουμε διαφορετικές υπηρεσίες, όπως ασφάλεια πχ firewall, αλλά και προτεραιότητα σε διαφορετικούς κόμβους σε περιπτώσεις ανάγκης.

Όμως, όλες οι υλοποιήσεις εμφανίζουν και κάποια κοινά χαρακτηριστικά τα οποία είναι οι απαιτήσεις σε ενέργεια, η υποστήριξη ετερογενών συσκευών, οι κινητοί κόμβοι. Ακόμα, όλες οι υλοποιήσεις που έχουν παρουσιαστεί χρησιμοποιούν διαφορετικές τεχνολογίες προκειμένου να καταφέρουν να παρουσιάσουν τα αποτελέσματα τους, όπως η εφαρμογή σε φυσικούς ή προσομοιωμένους κόμβους ή η χρήση διαφορετικών ελεγκτών ή πρωτοκόλλων. Οι διαφορετικές υλοποιήσεις τις οποίες θα συγκρίνουμε είναι το CORAL SDN, καθώς και το SDN-WISE. Αυτές οι δυο υλοποιήσεις θα μας απασχολήσουν στη συνέχεια της παρούσας εργασίας.

3.1 Εφαρμογές που χρησιμοποιούνται για τις προσομοιώσεις των IoT

Οι παρακάτω εφαρμογές χρησιμοποιούνται από τις δύο υλοποιήσεις τις οποίες πρόκειται να συγκρίνουμε. Πιο συγκεκριμένα οι δυο υλοποιήσεις χρησιμοποιούν λογισμικό το οποίο επιτρέπει την προσομοίωση ενός δικτύου IoT. Οι προσομοιώσεις αυτές έχουν σκοπό να μας δώσουν δεδομένα σχετικά με τα χαρακτηριστικά του δικτύου, καθώς και να συλλέξουν δεδομένα προκειμένου να μας δώσουν αποτελέσματα από τα οποία μπορούν να εξαχθούν χρήσιμα συμπεράσματα. Το λογισμικό που χρησιμοποιείται για τις προσομοιώσεις είναι το παρακάτω:

3.1.1 Contiki OS

Το Contiki OS είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα το οποίο έχει αναπτυχθεί προκειμένου να μας επιτρέπει να δημιουργούμε δίκτυα από IoT κόμβους, οι οποίοι έχουν την δυνατότητα να επικοινωνούν με το διαδίκτυο. Το Contiki OS χρησιμοποιεί και αξιοποιεί διαφορετικά διαδικτυακά πρωτόκολλα όπως IPv4, IPv6, καθώς και διαδικτυακά πρωτόκολλα τα οποία χρησιμοποιούνται κυρίως στο Διαδίκτυο των Αντικειμένων όπως 6LoWPAN, RPL, CoAP. Το Contiki έχει σχεδιαστεί προκειμένου να λειτουργεί σε συσκευές IoT που έχουν περιορισμένη μνήμη, ισχύ, ισχύ επεξεργασίας και εύρος ζώνης επικοινωνίας. Ένα τυπικό σύστημα IoT το οποίο χρησιμοποιεί Contiki OS έχει μνήμη της τάξης των kilobytes, κατανάλωση ισχύος της τάξης των milliwatts, ταχύτητα επεξεργασίας που μετράται σε mega Hertz και εύρος ζώνης επικοινωνίας της τάξης των εκατοντάδων kilobits / sec. Τέτοια συστήματα περιλαμβάνουν πολλούς τύπους ενσωματωμένων συστημάτων (embedded systems) και παλιούς υπολογιστές 8-bit.

3.1.2 Προσομοιωμένοι κόμβοι

Οι κόμβοι οι οποίοι χρησιμοποιούνται από το Contiki OS είναι προσομοιωμένοι κόμβοι των οποίων ο κώδικας της προσομοίωσης μεταγλωττίζεται για τον συγκεκριμένο κόμβο. Το Contiki OS υποστηρίζει διάφορους κόμβους από τους οποίους οι πιο σημαντικοί είναι:

- MicaZ
- Eth1120
- Trxeb1120
- Trxeb2520
- Exp2420
- Exp1101

- Exp1120
- CC430
- EXP430F5438
- WisMote
- Z1 Mote
- Sky Mote
- ESB Mote
- Cooja Mote

Επίσης, υποστηρίζει και κόμβους οι οποίοι είναι υλοποιημένοι σε Java προκειμένου να επιτρέψει στον χρήστη που δημιουργεί την προσομοίωση να δημιουργήσει τον δικό του κόμβο με τα μοναδικά χαρακτηριστικά τα οποία χρειάζεται.

3.1.3 Cooja simulator

Το Contiki OS ενσωματώνει τον προσομοιωτή δικτύων Cooja ο οποίος χρησιμοποιείται προκειμένου να μπορέσει ο χρήστης να δημιουργήσει διαφορετικά σενάρια με τη χρήση διαφορετικών κόμβων. Οι κόμβοι αυτοί χωρίζονται σε τρείς κατηγορίες:

1. **Προσομοιωμένοι κόμβοι (Cooja Motes):** Είναι κόμβοι οι οποίοι δεν είναι πραγματικοί και χρησιμοποιούνται μόνο εντός του Cooja Simulator.
2. **Προσομοίωση φυσικών κόμβων:** Όπου ο κώδικας της προσομοίωσης μεταγλωττίζεται για τον συγκεκριμένο κόμβο και εκτελείται η προσομοίωση στον φυσικό κόμβο.
3. **Κόμβοι Java (Java Motes):** Όπου η συμπεριφορά του κόμβου πρέπει να δημιουργηθεί ως κλάση Java.

Αφού ο χρήστης δημιουργήσει την προσομοίωση η οποία καλύπτει τις ανάγκες του (φυσικών ή προσομοιωμένων κόμβων) μπορεί να δει τα αποτελέσματα της εκτέλεσης. Ένα από τα βασικά χαρακτηριστικά του Cooja είναι η δυνατότητα να δημιουργεί ο χρήστης μεγάλες προσομοιώσεις από πραγματικές διαδικτυακές διατάξεις εφαρμογών ή με προσομοιώσεις με πάρα πολλή λεπτομέρεια σε πλήρως προσομοιωμένους κόμβους.

3.2 SDN-WISE

3.2.1 Τι είναι το SDN-WISE

SDN-WISE ή Software Defined Networking solution for Wireless Sensor Networks είναι μια stateful υλοποίηση η οποία ως βασικό στόχο έχει την ενσωμάτωση της τεχνολογίας των SDN στο Διαδίκτυο των Αντικειμένων. Αναπτύχθηκε στο Πανεπιστήμιο της Catania. Βασικοί στόχοι τους οποίους θέλει να πετύχει το SDN-WISE είναι:

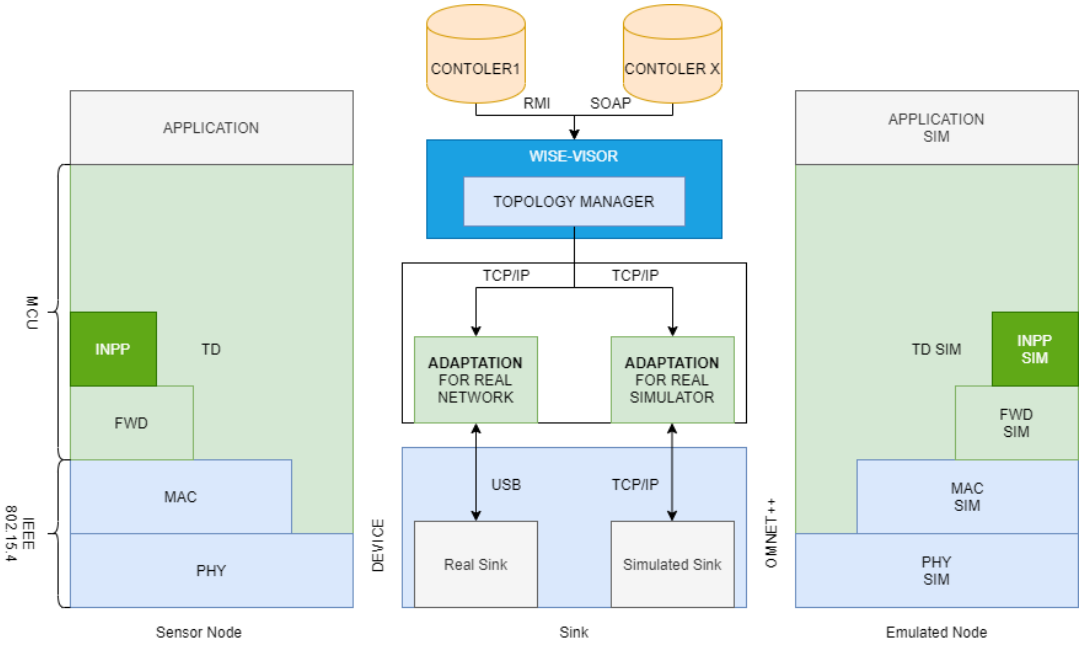
1) **Μείωση των διακινούμενων πακέτων με τους ελεγκτές:** Να μειώσει την ποσότητα της πληροφορίας που ανταλλάσσεται μεταξύ του SDN ελεγκτή και των κόμβων του δικτύου
2) **Εκμετάλλευση της επεξεργαστικής δύναμης των κόμβων:** Να κάνει τους κόμβους προγραμματιζόμενους, έτσι ώστε να εκτελούν ενέργειες οι οποίες δεν παρέχονται από stateless εφαρμογές.
3) **Stateful:** Να διατηρεί την πληροφορία σχετικά με την κατάσταση του κάθε κόμβου μέσα στους κόμβους και η κατάσταση των κόμβων να μπορεί να μεταβληθεί από τους ελεγκτές [9].
4) **Υποστήριξη ευέλικτων κανόνων:** Να υποστηρίζει ευέλικτους κανόνες οι οποίοι μπορούν να περιλαμβάνουν κομμάτια του πακέτου και όχι μόνο πεδία στην κεφαλίδα, έτσι ώστε να μπορεί να εκμεταλλευτεί στο έπακρο τις σχεσιακές συνθήκες που χρησιμοποιεί για να αντιληφθεί εάν κάποιος κανόνας μπορεί να εφαρμοστεί ή όχι.
5) **Εξοικονόμηση Πόρων:** Να είναι ενεργειακά αποδοτικό, καθώς το SDN-WISE απευθύνεται σε συσκευές οι οποίες λειτουργούν στο IoT και εμφανίζουν πολλούς περιορισμούς στην κατανάλωση ενέργειας προκειμένου να μπορούν λειτουργούν απρόσκοπτα για μεγάλα διαστήματα. Οι συσκευές αυτές για να πετύχουν εξοικονόμηση ενέργειας έχουν πολύ μικρές δυνατότητες σε επεξεργαστική ισχύ καθώς και μνήμη. Επίσης οι εφαρμογές που υποστηρίζουν δεν είναι απαιτητικές στον ρυθμό μετάδοσης των δεδομένων. Για αυτό το λόγο το SDN-WISE σχεδιάστηκε έτσι ώστε να είναι αποδοτικό σε όλους τους παραπάνω περιορισμούς. Για να το πετύχει αυτό το SDN-WISE εκμεταλλεύεται το βαθμό χρησιμοποίησης (duty cycle), δηλαδή ρυθμίζει για πόσο χρόνο θα είναι ανοιχτός ή κλειστός ο δέκτης του αισθητήρα και σε τι ποσοστό της μέγιστης ισχύος του εκπέμπει ο

πομπός [2]. Όμως, χρησιμοποιεί και (Data Aggregation) συσσώρευση των δεδομένων.

Επίσης, μετά την υλοποίηση του SDN-WISE παρουσιάστηκαν επιπλέον υλοποιήσεις με βάση το SDN-WISE οι οποίες εισάγουν την χρήση του NFV (Network Function Virtualization) στο IoT. Το NFV επιτρέπει να διαχωριστούν κάποιες υπηρεσίες δικτύου από τις συσκευές που τις εκτελούν. Για αυτό το λόγο εισάγεται η χρήση του Open Networking Operating System (ONOS) στο IoT, έτσι ώστε να υπάρχει δυνατότητα να εκτελούνται μικρά προγράμματα στους κόμβους τα οποία θα τα στέλνει ο Ελεγκτής. Παρέχει την δυνατότητα όμως να εκμεταλλευτεί η παραπάνω δυνατότητα προκειμένου να μπορούν να εκτελεστούν στους κόμβους κάποιες από τις λειτουργίες του δικτύου. Για παράδειγμα ο διαμοιρασμός της κεντρικής λειτουργίας της προώθησης πακέτων να μεταφερθεί στους κόμβους (geographic routing) [2]. Μια επιπλέον ιδέα η οποία εμφανίστηκε με την χρήση του SDN-WISE, καθώς και των NFV είναι η διαχείριση των Big Data τα οποία παράγουν τα IoT απευθείας στους κόμβους με χρήση MapReduce δομών [16]. Τέλος, με τη χρήση του ONOS δίνεται η δυνατότητα διαχείρισης ετερογενών δικτύων [17].

3.2.2 Αρχιτεκτονική SDN-WISE

Βασικό στοιχείο της αρχιτεκτονικής του SDN-WISE είναι ο διαχωρισμός που υπάρχει μεταξύ των κόμβων. Οι κόμβοι χωρίζονται στους απλούς κόμβους (Motes) και τους κόμβους προορισμού (Sink Nodes) . Οι κόμβοι προορισμού είναι σε κάθε τοπολογία του SDN-WISE ένας ή περισσότεροι κόμβοι και ενώνουν τους απλούς κόμβους (Motes) οι οποίοι υπάρχουν στο Επίπεδο Δεδομένων με τους Ελεγκτές οι οποίοι υπάρχουν στο Επίπεδο Ελέγχου. Όπως και τα SDN, το SDN-WISE διαχωρίζει το Επίπεδο των Δεδομένων από το Επίπεδο Ελέγχου [18].



Εικόνα 1 SDN-WISE Δομή Πρωτοκόλλου

Το πρωτόκολλο του SDN-WISE αποτελείται από διάφορα επίπεδα, τα οποία χρησιμοποιούνται προκειμένου να λειτουργεί αποδοτικά το πρωτόκολλο. Αναλυτικά τα επίπεδα είναι τα παρακάτω:

3.2.2.1 Επίπεδο Δεδομένων (Data Plane)

Είναι το χαμηλότερο επίπεδο του SDN-WISE. Είναι το επίπεδο στο οποίο υπάρχουν οι κόμβοι, ο κάθε κόμβος περιέχει έναν δέκτη για IEEE 802.15.4 , καθώς και ένα μικροελεγκτή (MCU) .

3.2.2.2 Επίπεδο Προώθησης πακέτων (Forwarding Layer)

Το Επίπεδο Προώθησης πακέτων τρέχει επάνω από το IEEE 802.15.4. Είναι το επίπεδο το οποίο διαχειρίζεται τα πακέτα τα οποία λαμβάνει ο κόμβος. Προκειμένου ο κόμβος να γνωρίζει πως θα διαχειριστεί τα πακέτα που λαμβάνει ελέγχει το WISE Flow table και προχωράει στην διαχείριση με βάση τους κανόνες που υπάρχουν σε αυτό τον πίνακα. Το επίπεδο προώθησης πακέτων ανανεώνει συνεχώς τον πίνακα αυτό με νέους κανόνες με βάση τις οδηγίες που έρχονται από τους ελεγκτές.

3.2.2.3 Επίπεδο επεξεργασίας Πακέτων (In-Network Packet Processing (INPP) layer)

Το επίπεδο επεξεργασίας πακέτων λειτουργεί πάνω από το Επίπεδο Προώθησης πακέτων. Είναι το επίπεδο το οποίο πραγματοποιεί την συσσώρευση των δεδομένων (Data Aggregation) ή οποιαδήποτε άλλη δικτυακή επεξεργασία χρειάζεται. Στο SDN-WISE το επίπεδο INPP συγκεντρώνει και ενώνει τα πακέτα που ταξιδεύουν προς το ίδιο σημείο της τοπολογίας (data aggregation) . Με τον τρόπο αυτό το SDN-WISE επιτυγχάνει να μειώνει την διακίνηση των πακέτων και να πετυχαίνει εξοικονόμηση ενέργειας.

3.2.2.4 Επίπεδο Ανακάλυψης της Τοπολογίας (Topology Discovery (TD))

Είναι το επίπεδο το οποίο συλλέγει πληροφορίες από όλους τους κόμβους και ρυθμίζει την συμπεριφορά των κόμβων με βάση τους κανόνες του Ελεγκτή. Προκειμένου να το πετύχει αυτό έχει πρόσβαση σε όλα τα επίπεδα του πρωτοκόλλου χρησιμοποιώντας τα κατάλληλα API. Επίσης, μέσω ενός API τροφοδοτεί με δεδομένα την Εφαρμογή η οποία λειτουργεί στον κόμβο κατά αυτό τον τρόπο επεκτείνεται το IEEE 802 API [9].

3.2.2.5 Επίπεδο ελέγχου (Control plane)

Είναι το επίπεδο στο οποίο πραγματοποιείται η διαχείριση του δικτύου. Η διαχείριση πραγματοποιείται από έναν ή περισσότερους ελεγκτές, ένας από τους οποίους είναι ο WISE-Visor. Βασική λειτουργία του WISE-Visor είναι η Διαχείριση της Τοπολογίας (TM) .

Η Διαχείριση της Τοπολογίας αποτελεί ένα ξεχωριστό επίπεδο το οποίο διαχειρίζεται τους πόρους του δικτύου προκειμένου στους ίδιους κόμβους να μπορούν να εφαρμοστούν οι πολιτικές διαχείρισης του κάθε ελεγκτή για το κάθε διαφορετικό δίκτυο. Προκειμένου να το πετύχει αυτό χρησιμοποιεί διαφορετικά API τα οποία του δίνουν πρόσβαση σε όλα τα επίπεδα του πρωτοκόλλου.

Κύριος σκοπός του επίπεδου Διαχείρισης της Τοπολογίας είναι να συλλέγει πληροφορίες από τους κόμβους και να τις στέλνει στους Ελεγκτές σε μορφή γράφου του δικτύου, άρα διατηρεί έναν γράφο ο οποίος περιέχει την τοπολογία του δικτύου και ο γράφος αυτός ανανεώνεται συνεχώς. Επίσης, ελέγχει όλα τα επίπεδα του πρωτοκόλλου με βάση τις οδηγίες των ελεγκτών. Ανάμεσα στους κόμβους προορισμού (Sink Nodes) και τον WISE-Visor ελεγκτή υπάρχει και το Επίπεδο προσαρμογής.

Οι ελεγκτές οι οποίοι διαχειρίζονται το SDN-WISE μπορεί είτε να λειτουργούν στον ίδιο κόμβο που φιλοξενεί το Επίπεδο διαχείρισης τοπολογίας, είτε να λειτουργούν σε κάποια απομακρυσμένη συσκευή πχ Server. Προκειμένου να μπορούν οι ελεγκτές να

επικοινωνήσουν με το Επίπεδο Διαχείρισης της Τοπολογίας μπορούν να χρησιμοποιήσουν διαφορετικό τρόπο επικοινωνίας ανάλογα με το αν τρέχουν στο ίδιο επίπεδο ή αν είναι απομακρυσμένοι. Εάν βρίσκονται στο ίδιο επίπεδο τότε επικοινωνούν μέσω Java μεθόδων που παρέχει το Επίπεδο Διαχείριση της Τοπολογίας. Σε αντίθετη περίπτωση μπορούν να χρησιμοποιήσουν Java Remote Method Invocation (RMI) ή Simple Object Access Protocol (SOAP) . Κατά αυτό τον τρόπο στο SDN-WISE μπορούν να χρησιμοποιηθούν διάφοροι ελεγκτές. Στα παραδείγματα χρήσης του SDN-WISE υπάρχουν 2 διαφορετικοί ελεγκτές οι οποίοι είναι υλοποιημένοι σε Java, αλλά και ο ONOS. Με τη χρήση του ONOS το SDN-WISE μετεξελίσσεται έτσι ώστε να εκτελεί ολιστική διαχείριση του δικτύου, καθώς ολοκληρωμένη προώθηση πακέτων μεταξύ δικτύων μεταφοράς και ασύρματων αισθητήρων [17] .

3.2.2.6 Επίπεδο Προσαρμογής (Adaptation layer)

Το πρωτόκολλο SDN-WISE χρησιμοποιεί το Επίπεδο Προσαρμογής έτσι ώστε να μπορεί να υποστηρίξει προσομοιωμένους κόμβους. Επομένως, το επίπεδο αυτό επιτρέπει στο Επίπεδο ελέγχου να επικοινωνεί και να διαχειρίζεται τους προσομοιωμένους κόμβους.

3.2.3 Διαχείριση Πακέτων

Τα πακέτα που μεταδίδονται στο SDN-WISE διαχειρίζονται από τους κόμβους με βάση κάποια πεδία που υπάρχουν στην κεφαλίδα, καθώς και από το περιεχόμενο του πακέτου με βάση κάποιες συγκρίσεις π.χ. μεγαλύτερο από, μικρότερο από κτλ. Επίσης, ο κάθε κόμβος έχει 3 πίνακες με βάση τους οποίους πραγματοποιεί την διαχείριση των πακέτων. Οι πίνακες αυτοί είναι: WISE States Array, Accepted IDs Array και WISE Flow Table, αυτοί οι πίνακες περιέχουν πληροφορίες οι οποίες έρχονται από τον/τους ελεγκτές. Αναλυτικά ο κάθε πίνακας περιέχει τις παρακάτω πληροφορίες:

- **WISE States Array:** Περιέχει τον χαρακτηρισμό της κατάστασης που έχει ο κόμβος εκείνη την στιγμή από τον κάθε ελεγκτή ο οποίος υπάρχει στο δίκτυο.
- **Accepted IDs Array:** Περιέχει πληροφορίες σχετικά με το ποια πακέτα θα πρέπει να επεξεργαστεί ο κόμβος έτσι ώστε ο κάθε κόμβος να διαχειρίζεται μόνο όποια πακέτα πρέπει να διαχειριστεί. Σε κάθε πακέτο που λαμβάνει ο κόμβος ελέγχει το ID. Αν το ID του πακέτου περιέχεται στον πίνακα με τα ID's τότε προχωράει στην διαχείριση, αλλιώς το πετάει.
- **WISE Flow Table:** Περιέχει πληροφορίες σχετικά με τους κανόνες με βάση τους οποίους διαχειρίζεται τα πακέτα. Εάν κανένας από τους κανόνες που υπάρχουν

μέσα στον πίνακα δεν ικανοποιείται, τότε ο κόμβος επικοινωνεί με τον ελεγκτή προκειμένου να λάβει νέο κανόνα για το πως θα πρέπει διαχειριστεί το πακέτο. Εκτός όμως από τους κανόνες, περιέχει και την πληροφορία σχετικά με τους γειτονικούς κόμβους και την καλύτερη γειτονική διαδρομή σε απόσταση 1 κόμβου (1 hop) για την μετάδοση των δεδομένων προς τον μοναδικό προορισμό (Sink Node) . Η ανακάλυψη των καλύτερων γειτόνων γίνεται από τους ίδιους τους κόμβους.

3.2.4 Πακέτα SDN-WISE και προώθηση πακέτων

Τα πακέτα του SDN-WISE είναι περιέχουν μια κεφαλίδα 10 Byte η οποία έχει την παρακάτω μορφή:

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th></tr>
<tr><td>0</td><td colspan="8">Packet Length</td><td colspan="8">Scope</td></tr>
<tr><td>2</td><td colspan="16">Source Address</td></tr>
<tr><td>4</td><td colspan="16">Destination Address</td></tr>
<tr><td>6</td><td>U</td><td colspan="7">Type</td><td colspan="8">TTL</td></tr>
<tr><td>8</td><td colspan="16">Next Hop ID</td></tr>
</table>

Πίνακας 1 WISE Κεφαλίδα Πακέτου

Αναλυτικά τα πεδία της κεφαλίδας είναι τα εξής:

- Το Μέγεθος του Πακέτου (Packet Lenght) σε bytes δείχνει εάν το πακέτο είναι κενό ή όχι.
- Σκοπός (Scope) : δείχνει το ποιοι ελεγκτές ενδιαφέρονται για το περιεχόμενο του πακέτου. Η προεπιλεγμένη τιμή είναι 0 και μεταβάλλεται ανάλογα με το WISE Flow Table του κόμβου που δημιούργησε το πακέτο.
- Διεύθυνση Αποστολέα και παραλήπτη: 2bytes διευθύνσεις δηλώνουν το ποιος δημιούργησε το πακέτο και τον τελικό παραλήπτη.
- U: Είναι ένα πεδίο που ενημερώνει εάν το πακέτο πρέπει να φτάσει στον πιο κοντινό Sink.
- Τύπος (Type) : Είναι το πεδίο που δείχνει τον τύπο του πακέτου πχ δεδομένα, TD κτλπ
- TTL: Είναι ο χρόνος ζωής του πακέτου και σε κάθε κόμβο πέφτει η τιμή του TTL κατά 1.

- Ταυτότητα του επόμενου κόμβου (Next Hop ID) : Το πεδίο αυτό δείχνει στον κόμβο εάν πρέπει να διαχειριστεί το πακέτο. Για να προχωρήσει στην διαχείριση, πρέπει το Next Hop ID να υπάρχει στα ID Που δέχεται ο κόμβος, δηλαδή στο Accepted IDs Array.

Προκειμένου ο κόμβος να διαχειριστεί το πακέτο πρέπει το Next Hop ID να είναι μέσα σε αυτά που δέχεται ο κόμβος (στο Accepted IDs Array), τότε ο κόμβος προχωράει στην επεξεργασία του πακέτου. Για να το κάνει αυτό πρέπει να συμβουλευτεί το WISE Flow Table το οποίο είναι όπως ο παρακάτω πίνακας.

Matching Rule					Matching Rule					Matching Rule					Action					Statistics	
Op	Size	S	Addr.	Value	Op	Size	S	Addr.	Value	Op	Size	S	Addr.	Value	Type	M	S	Addr	Value	TTL	Counter
=	2	0	2	B	>	2	0	10	X_{Thr}	=	1	1	0	0	Modify	1	1	0	1	122	23
=	2	0	2	B	≤	2	0	10	X_{Thr}	=	1	1	0	1	Modify	1	1	0	0	122	120
=	2	0	2	B	-	-	-	-	-	-	0	-	-	-	Forward	0	0	0	D	122	143
=	2	0	2	A	=	1	1	0	0	-	0	-	-	-	Drop	0	0	-	-	100	42
=	2	0	2	A	=	1	1	0	1	-	0	-	-	-	Forward	0	0	0	D	100	32

Πίνακας 2 WISE Flow Table

Η δομή του WISE Flow Table μοιάζει με την δομή του OpenFlow, δηλαδή υπάρχουν 3 πεδία: Κανόνες Αντιστοίχισης (Matching Rules), Ενέργειες (Actions), Στατιστικά (Statistics) .

Τα πεδία του πίνακα είναι τα εξής:

- **S:** Ορίζει αν ο κανόνας είναι για το πακέτο S=0 ή για την κατάσταση S=1
- **Offset:** Είναι το πρώτο Byte του πακέτου
- **Size:** Είναι το μέγεθος του πακέτου
- **Operator:** Κάνει την σύγκριση με την τιμή που υπάρχει στο πεδίο Value

Εάν οι συνθήκες που υπάρχουν στον κανόνα ικανοποιούνται τότε γίνεται η αντίστοιχη ενέργεια και το πεδίο των στατιστικών ανανεώνεται [19].

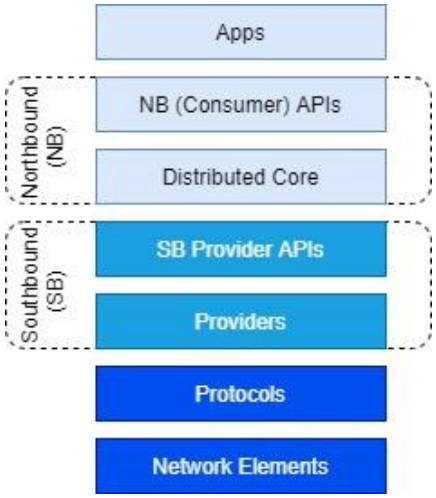
Οι ενέργειες έχουν τα εξής πεδία:

- **Type:** Ποια είναι η ενέργεια που θα πραγματοποιηθεί. Μπορεί να είναι μια από τις παρακάτω: "Forward to" , "Drop" , "Modify" , "Send to INPP" , "Turn off radio".
- **M:** Παίρνει δύο τιμές 0 ή 1, εάν η τιμή είναι 0 τότε εκτελείται η ενέργεια και μετά σταματάει να ψάχνει για άλλους κανόνες που μπορεί να ικανοποιούνται, όμως εάν είναι 1 τότε συνεχίζει.
- **Offset και Value:** αυτά τα πεδία εξαρτώνται από τον τύπο του κανόνα π.χ αν είναι "Forward to" πρέπει να ξέρουμε σε ποιον κόμβο θα πρέπει να πάει το πακέτο.

Τα Στατιστικά (Statistics) έχουν τα ίδια πεδία με το OpenFlow.

3.2.5 Ελεγκτής SDN-WISE

Όπως είδαμε και στο επίπεδο ελέγχου το SDN-WISE έχει τη δυνατότητα να χρησιμοποιεί διαφορετικούς ελεγκτές προκειμένου να ελέγχει το δίκτυο. Βασικός όμως ελεγκτής είναι ο WISE-Visor. Βασική λειτουργία του WISE-Visor είναι η Διαχείριση της Τοπολογίας (TM). Έτσι, ο WISE-Visor διαχειρίζεται την τοπολογία του δικτύου και διατηρεί ένα γράφο τον οποίο διαμοιράζεται με τους υπόλοιπους ελεγκτές, φροντίζει όμως να επιβάλει στο δίκτυο και τις πολιτικές διαχείρισης των εξωτερικών ελεγκτών. Ένας από τους εξωτερικούς ελεγκτές τους οποίους έχουν ενσωματώσει στο SDN-WISE είναι ο Open Network Operating System (ONOS), ο οποίος είναι ένα κατανεμημένο ανοιχτού κώδικα λειτουργικό σύστημα το οποίο δημιουργήθηκε για να διαχειρίζεται τις λειτουργίες του δικτύου με βάση την αρχιτεκτονική των SDN και είναι υλοποιημένος σε Java. Προκειμένου να ενσωματώσει την αρχιτεκτονική των SDN διαχωρίζει τα στοιχεία του δικτύου τα οποία διαχειρίζεται, καθώς επίσης και τη διαχείριση του δικτύου από την αρχιτεκτονική που χρησιμοποιείται στο δίκτυο. Έτσι πετυχαίνει να διαχειρίζεται ετερογενή δίκτυα.



Εικόνα 2 Αρχιτεκτονική επιπέδων του ONOS

Ο ONOS όπως βλέπουμε στην Εικόνα 2, χρησιμοποιεί μία αρχιτεκτονική επιπέδων η οποία επιτρέπει την χρήση και διαχείριση διαφορετικών πρωτοκόλλων. Όπως βλέπουμε και στην Εικόνα 2 κάποια από τα στοιχεία του ONOS ανήκουν σε πολλαπλά επίπεδα τα οποία αποτελούν τα υποσυστήματα του ONOS και υλοποιούν βασικές λειτουργίες του ONOS. Οι βασικές λειτουργίες είναι:

- Διαχείριση συσκευών

- Διαχείριση συνδέσεων
- Διαχείριση των κανόνων δρομολόγησης
- Διαχείριση της τοπολογίας

Η επικοινωνία μεταξύ των επιπέδων του ONOS πραγματοποιείται με 2 τρόπους, ανάλογα από το σημείο που προέρχεται. Έτσι τα υψηλότερα επίπεδα επικοινωνούν με τα χαμηλότερα μέσω APIs (Northbound/Southbound), ενώ τα χαμηλότερα με τα υψηλότερα με την χρήση συμβάντων (events) . Το επίπεδο του πρωτοκόλλου περιλαμβάνει τα προγράμματα οδήγησης που χρησιμοποιεί το κάθε πρωτόκολλο επικοινωνίας των συσκευών. Βασική λειτουργία όμως του επιπέδου του πρωτοκόλλου είναι να διαχειρίζεται την επικοινωνία με τους κόμβους στο χαμηλότερο επίπεδο το οποίο είναι οι συσκευές οι οποίες συνδέονται με τον ONOS. Πιο συγκεκριμένα στην υλοποίηση του SDN-WISE η συσκευή αυτή είναι ο Sink κόμβος [17].

Στην νότια πλευρά (southbound (SB)) του ONOS υπάρχει το επίπεδο Παραγωγού (Provider) όπου περιλαμβάνει στοιχεία που είναι υπεύθυνα για την μεταγλώττιση των abstractions που έρχονται από τα υψηλότερα επίπεδα σε εντολές διαχείρισης δικτύου για το πρωτόκολλο που χρησιμοποιείται, αλλά και να μεταγλωττίζει τις πληροφορίες σχετικά με το δίκτυο που στέλνει το επίπεδο πρωτοκόλλου σε abstractions που χρησιμοποιούνται στα υψηλότερα επίπεδα.

Στην βόρεια πλευρά (Northbound (NB)) του ONOS υπάρχουν στοιχεία τα οποία διαχειρίζονται τα διαθέσιμα abstractions, όπως κανόνες προώθησης, εισερχόμενα και εξερχόμενα πακέτα, συσκευές. Έτσι το NB με τη χρήση των κατάλληλων API παρουσιάζει τα abstractions στο επίπεδο Καταναλωτή (Consumer). Έτσι, αυτά τα abstractions προωθούνται στο Core επίπεδο το οποίο διατηρεί πληροφορίες σχετικά με τις συσκευές, την τοπολογία, τις συνδέσεις μεταξύ των κόμβων.

Στο επίπεδο Εφαρμογής (Application) τα στοιχεία του εκμεταλλεύονται τις πληροφορίες σχετικά με την κατάσταση του δικτύου τις οποίες λαμβάνουν από το NB και εκτελούν περίπλοκες ενέργειες οι οποίες χρησιμοποιούν αρκετά υποσυστήματα. Οι βασικές εφαρμογές του ONOS ενεργοποιούνται μετά από συγκεκριμένα γεγονότα τα οποία συμβαίνουν στο δίκτυο και δημιουργούν με βάση συγκεκριμένα κριτήρια τους απαραίτητους κανόνες δρομολόγησης τους οποίους στέλνουν στο επίπεδο του δικτύου.

3.2.6 Πως λειτουργεί η Ανακάλυψη της τοπολογίας

Προκειμένου το Επίπεδο ελέγχου να έχει όλα τα απαιτούμενα δεδομένα για να μπορεί να διαχειριστεί το δίκτυο χρειάζεται να έχει έγκυρες πληροφορίες σχετικά με την τοπολογία του δικτύου. Όμως και οι κόμβοι θα πρέπει να έχουν έγκυρες πληροφορίες στο WISE Flow Table σχετικά με τους γειτονικούς τους κόμβους προκειμένου να γνωρίζουν την πιο αποδοτική διαδρομή προς τον Sink Node, καθώς και τους γειτονικούς κόμβους. Έτσι, το πρωτόκολλο ανακάλυψης της τοπολογίας εκτελείται σε όλους τους κόμβους του δικτύου και υπεύθυνος για την δημιουργία και τη συγκέντρωση των δεδομένων αυτών είναι ο ελεγκτής WISE-Visor. Περιοδικά οι Sink του δικτύου στέλνουν ένα broadcast πακέτο Ανακάλυψης Τοπολογίας (TD packet) . Το πακέτο αυτό περιέχει τα παρακάτω δεδομένα:

- Το αναγνωριστικό του Sink που το δημιούργησε
- Την απόσταση από τον Sink (Αρχικά είναι 0 μόλις το μεταδώσει ο Sink)
- Την στάθμη της μπαταρίας

Όταν ένας κόμβος Α λάβει το TD Πακέτο από ένα γειτονικό κόμβο Β πραγματοποιεί τις παρακάτω ενέργειες:

1. Βάζει τον Β στην Λίστα Γειτόνων μαζί με τις πληροφορίες RSSI και Στάθμη μπαταρίας ή ανανεώνει την εγγραφή με τις πληροφορίες RSSI και Στάθμη μπαταρίας εάν ήταν ήδη στην Λίστα Γειτόνων.
2. Ελέγχει εάν έχει λάβει TD Πακέτο με μικρότερη τιμή απόστασης από τον Sink. Εάν δεν είχε λάβει τέτοιο πακέτο, τότε ο Α αυξάνει κατά ένα την τιμή της απόστασης από τον Sink και αποθηκεύει τον Β ως τον καλύτερο γειτονικό κόμβο προς τον Sink.
3. Ενημερώνει το πεδίο της Στάθμης της μπαταρίας στο πακέτο με την στάθμη της μπαταρίας του.
4. Μεταδίδει το ενημερωμένο πακέτο μέσω Broadcast.

Περιοδικά ο κάθε κόμβος δημιουργεί και στέλνει στον WISE-Visor ένα πακέτο που περιέχει την λίστα τον γειτόνων. Προκειμένου να φτάσει το πακέτο στον WISE-Visor το στέλνει στον γειτονικό κόμβο που έχει την μικρότερη τιμή απόστασης από τον Sink και ο γειτονικός κόμβος το στέλνει προς τον Sink.

Όπως είναι λογικό, όσο πιο συχνά διακινούν τέτοια πακέτα η απόδοση του πρωτοκόλλου πέφτει καθώς δημιουργείται μεγάλο overhead και κατά συνέπεια και μεγαλύτερη κατανάλωση ενέργειας στους κόμβους καθώς δημιουργούν, διαχειρίζονται και προωθούν τέτοια πακέτα. Όμως όταν τα σενάρια που τρέχουν είναι δυναμικά, δηλαδή οι κόμβοι

κινούνται μέσα στο δίκτυο πρέπει να εκτελείται συχνά η διαδικασία αυτή δημιουργώντας τελικά ζητήματα overhead στο δίκτυο με αποτέλεσμα ακόμα και να χάνεται η επικοινωνία μεταξύ των κόμβων και του Sink κόμβου.

3.2.7 Ανακάλυψη γειτονικών κόμβων

Η ανακάλυψη των γειτονικών κόμβων πραγματοποιείται από τους ίδιους τους κόμβους. Το πρωτόκολλο ανακάλυψης εκτελείται από το Επίπεδο Ανακάλυψης Τοπολογίας (Topology Discovery layer) . Μέσα στο δίκτυο διακινούνται τα πακέτα ανακάλυψης (TD) τα οποία περιέχουν πληροφορίες σχετικά με την στάθμη της μπαταρίας και τον κοντινότερο Sink Node σε αριθμό κόμβων (hop) . Όταν ένας κόμβος λάβει ένα τέτοιο πακέτο συγκρίνει τις πληροφορίες που περιέχει το πακέτο με τις πληροφορίες που έχει στο WISE Flow Table και επιλέγει τον καλύτερο με βάση:

- Τον αριθμό τον ενδιάμεσων κόμβων (hop)
- Την τιμή της ένταση των λαμβανόμενων σημάτων (Received Signal Strength Indicator) (RSSI)
- Τη στάθμη της μπαταρίας

Αφού πραγματοποιήσει την επιλογή, ανανεώνει το WISE Flow Table. Η πληροφορία αυτή αποθηκεύεται και σε μία λίστα με γείτονες, η οποία περιέχει πληροφορίες όπως διευθύνσεις, RSSI, στάθμη μπαταρίας. Η λίστα των γειτόνων περιοδικά στέλνεται στο Επίπεδο διαχείρισης της Τοπολογίας (Topology Management TM) και χρησιμοποιείται για να δημιουργηθεί ένας γράφος με την τοπολογία του δικτύου. Η λίστα αυτή καθαρίζεται και ξαναδημιουργείται κάθε φορά που στέλνεται πακέτο TD, έτσι ώστε να είναι πάντα ενημερωμένος ο γράφος με την τρέχουσα τοπολογία.

3.2.8 Εξοικονόμηση Ενέργεια

Πιο συγκεκριμένα προκειμένου να είναι ενεργειακά αποδοτικό το SDN-WISE χρησιμοποιεί βαθμό χρησιμοποίησης (duty cycle), δηλαδή ρυθμίζει για πόσο χρόνο θα είναι ανοιχτός ή κλειστός ο δέκτης του αισθητήρα. Όμως χρησιμοποιεί και (Data Aggregation) συσσώρευση των δεδομένων. Ο τρόπος με τον οποίο γίνεται η συσσώρευση των δεδομένων είναι ο εξής: συγκεντρώνονται σε ένα κόμβο του δικτύου αρκετά μηνύματα, τα μηνύματα που πρόκειται να κινηθούν σε ένα συγκεκριμένο κομμάτι της τοπολογίας μετατρέπονται σε ένα μήνυμα αντί να μεταδοθούν το κάθε ένα ξεχωριστά [20]. Ένας ακόμα τρόπος με τον οποίο το SDN-WISE εξοικονομεί ενέργεια είναι να μειώνει την δύναμη της εκπομπής του πομπού [2] του κόμβου. Τέλος, προκειμένου να πετύχει

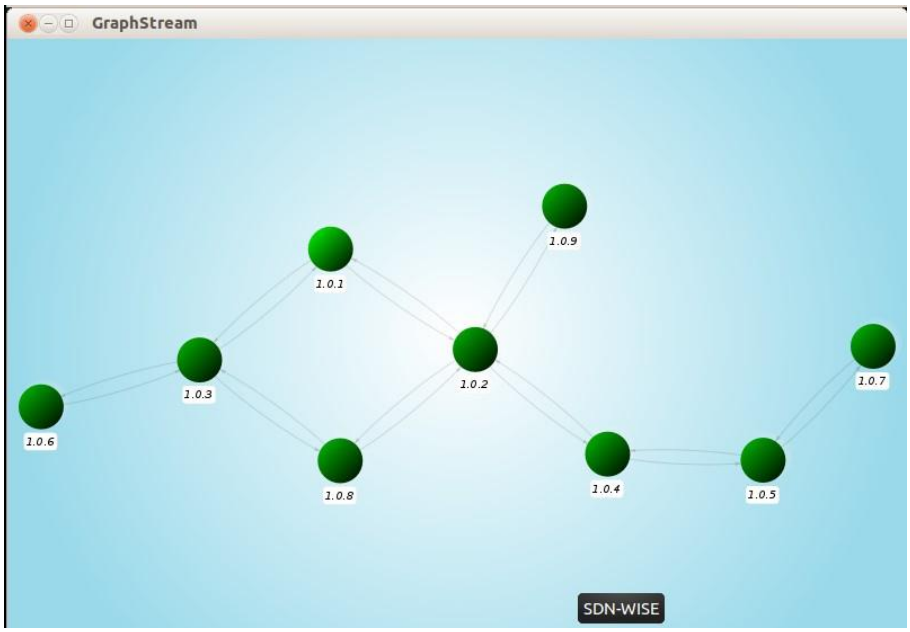
εξοικονόμηση σε όλους τους πόρους του συστήματος οι κόμβοι θα πρέπει να ελαχιστοποιήσουν την επικοινωνία με τον ελεγκτή όσο το δυνατόν περισσότερο, άρα κάποια από την προγραμματιστική λογική του συστήματος πρέπει να μεταφερθεί στους κόμβους.

3.2.9 Χρήση SDN-WISE

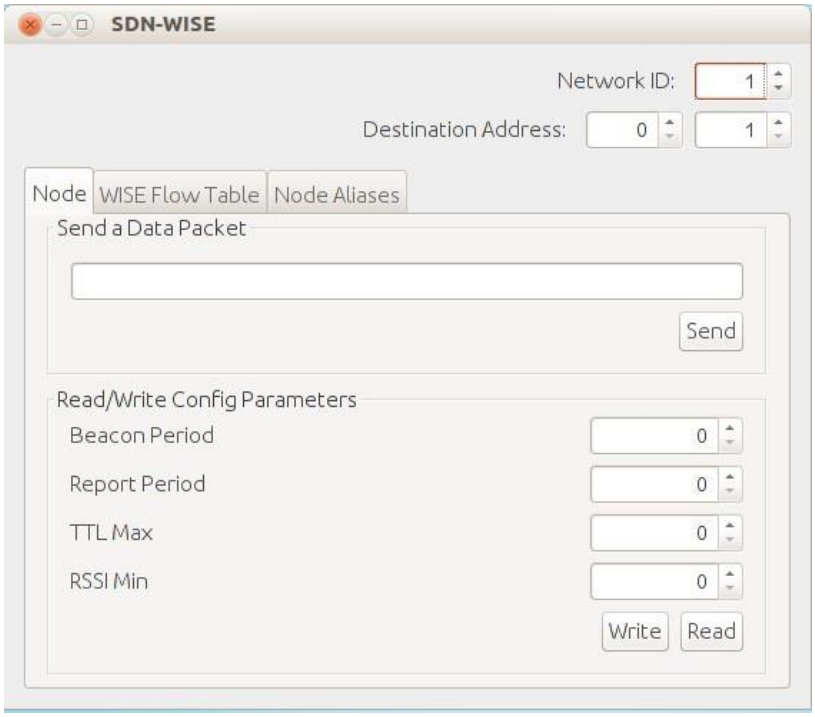
Το SDN-WISE παρέχει τρεις διαφορετικές υλοποιήσεις τις οποίες μπορεί ο χρήστης να χρησιμοποιήσει προκειμένου να μπορέσει να εκτελέσει προσομοιώσεις για να δει τον τρόπο με τον οποίο λειτουργεί το SDN-WISE. Οι διαφορετικές υλοποιήσεις είναι:

3.2.9.1 SDN-WISE JAVA

Είναι μια υλοποίηση σε Java του SDN-WISE η οποία περιέχει προσομοιωμένους κόμβους υλοποιημένους, ένα μικρό Επίπεδο Ελέγχου το οποίο χρησιμοποιείται προκειμένου να διαχειρίζεται ο χρήστης το προσομοιωμένο δίκτυο και τέλος τον βασικό πυρήνα του SDN-WISE, το οποίο παρέχει όλη την βασική αρχιτεκτονική του SDN-WISE. Δημιουργεί μια προσομοίωση η οποία περιέχει 9 κόμβους, την τοπολογία της οποίας την αποτυπώνει σε ξεχωριστό παράθυρο, καθώς και ένα GUI μέσω του οποίου ο χρήστης μπορεί να πειράξει τις παραμέτρους της προσομοίωσης και να δει τους κανόνες οι οποίοι χρησιμοποιούνται. Επίσης, ο χρήστης έχει τη δυνατότητα να στείλει κάποιο πακέτο με το μήνυμα το οποίο θέλει να περιέχει.

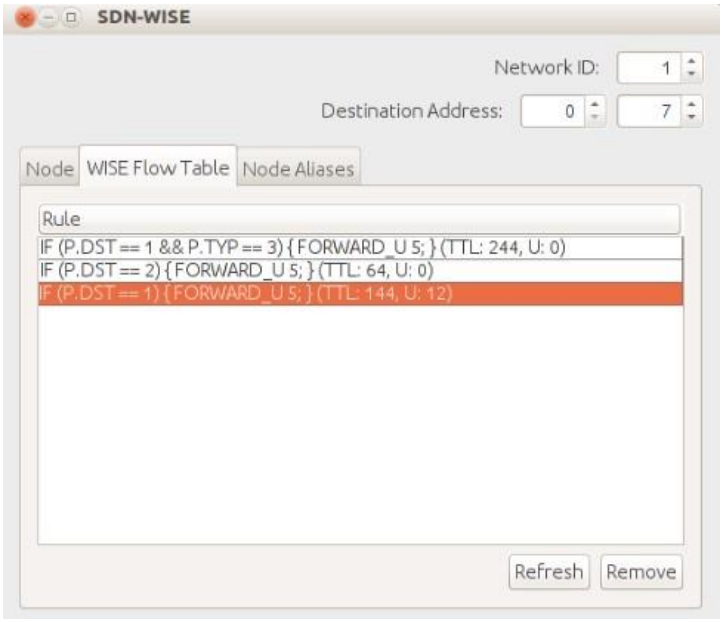


Εικόνα 3 Τοπολογία του δικτύου

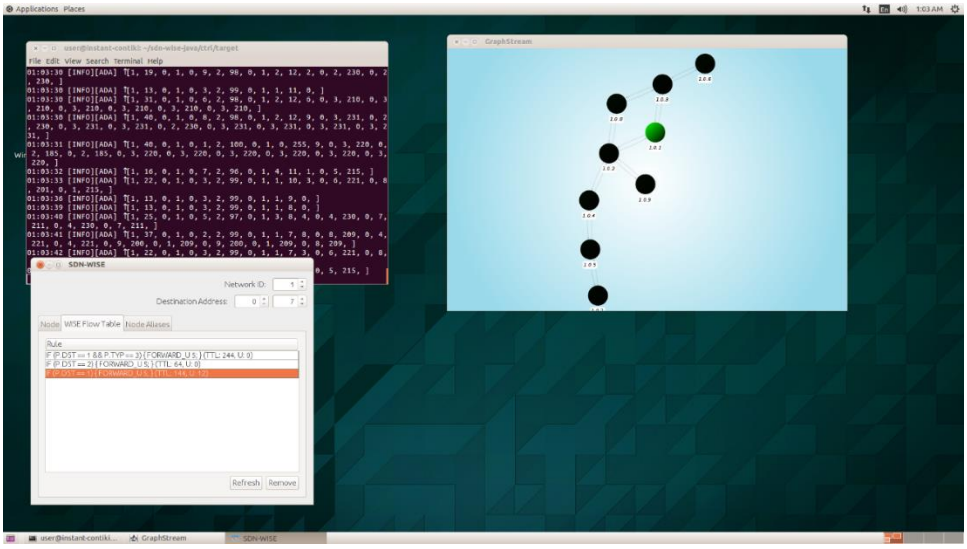


Εικόνα 4 GUI με τις επιλογές του χρήστη

Στο συγκεκριμένο παράθυρο Εικόνα 4 ο χρήστης έχει τη δυνατότητα να επιλέξει από επάνω δεξιά την διεύθυνση του κόμβου (Destination Address) στον οποίο θέλει να στείλει το πακέτο. Στο πεδίο Send a Data Packet ο χρήστης πληκτρολογεί το μήνυμα το οποίο θέλει να στείλει. Τέλος, στο πεδίο Read/Write Config Parameters μπορεί να στείλει αίτημα στους κόμβους για να στείλουν τις παραμέτρους που χρησιμοποιούν (Read) και αφού τις επεξεργαστεί μπορεί να στείλει πίσω στους κόμβους τις νέες παραμέτρους. Όπως βλέπουμε στην Εικόνα 5 ο χρήστης μπορεί να δει τους κανόνες δρομολόγησης οι οποίοι χρησιμοποιούνται, καθώς και να διαγράψει κάποιον από τους κανόνες. Οι κανόνες δημιουργούνται αφού ο χρήστης στείλει κάποιο πακέτο προς κάποιο κόμβο και εγκαθιδρύονται από τον ελεγκτή. Στην Εικόνα 6 βλέπουμε συνολικά τα επιμέρους παράθυρα που βλέπει ο χρήστης κατά την εκτέλεση της προσομοίωσης.



Εικόνα 5 WISE Flow Table



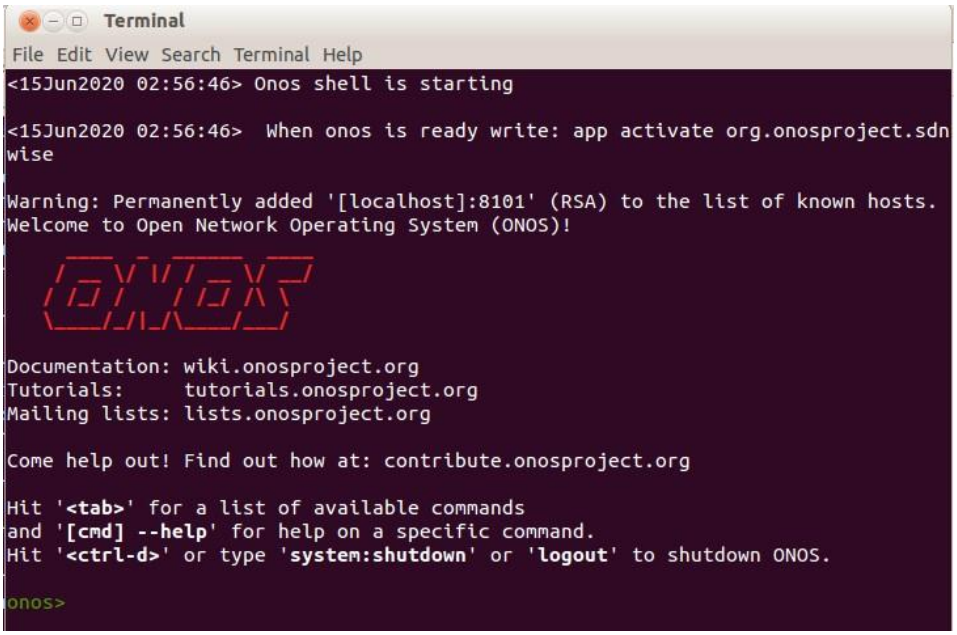
Εικόνα 6 Συνολική εικόνα

Προκειμένου να εκτελεστεί η προσομοίωση ο χρήστης μέσω των εκτελέσιμων script που δημιουργήθηκαν στα πλαίσια της παρούσας εργασίας και υπάρχουν διαθέσιμα στο Παράρτημα Α μπορεί να εγκαταστήσει τα απαραίτητα αρχεία για να ξεκινήσει η προσομοίωση. Για να το κάνει αυτό προτείνεται η χρήση λειτουργικού συστήματος Ubuntu 16.04 . Για να εγκατασταθούν τα απαραίτητα αρχεία ο χρήστης πρέπει να εκτελέσει το script: **install_sdn_wise_java.sh**

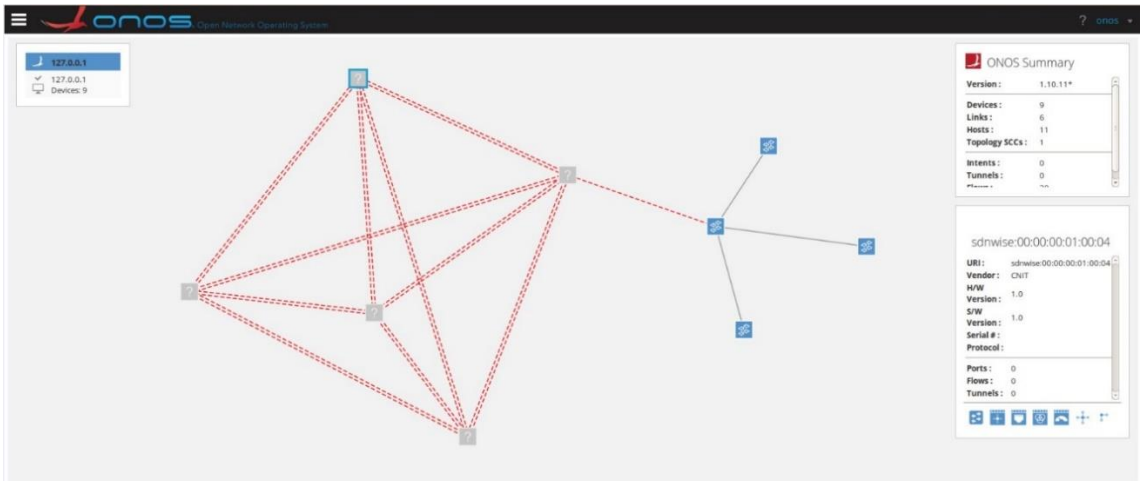
και για να εκτελέσει την προσομοίωση: **run_sdn_wise_java.sh**

3.2.9.2 SDN-WISE υλοποίηση για τον έλεγχο ετερογενών δικτύων

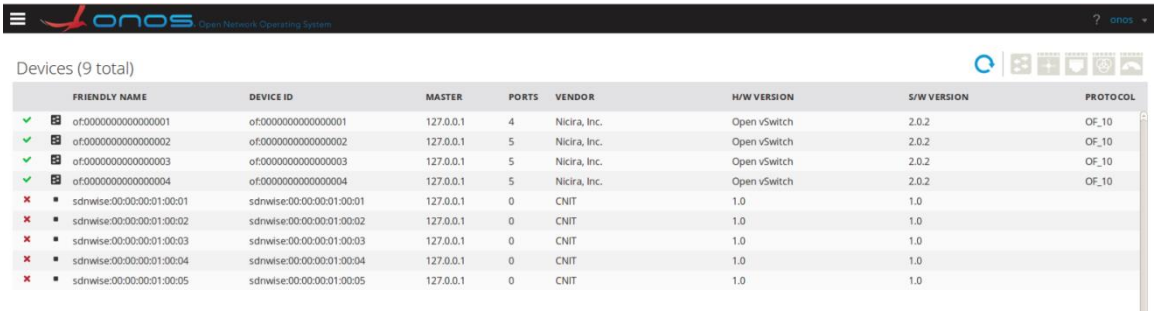
Αποτελεί την βασική υλοποίηση του SDN-WISE, η υλοποίηση αυτή επιτρέπει στον χρήστη να δημιουργήσει μία προσομοίωση η οποία δημιουργεί ένα ολοκληρωμένο δίκτυο που ελέγχεται από τον ελεγκτή ONOS, όπου ένα εικονικό δίκτυο από μεταγωγείς (switch) μπορεί να επικοινωνήσει με τους κόμβους ενός εξομοιωμένου δικτύου ασύρματων αισθητήρων SDN-WISE. Η υλοποίηση αυτή χρησιμοποιεί τον ελεγκτή ONOS ο οποίος πραγματοποιεί τον έλεγχο στα δύο δίκτυα των switch και των SDN-WISE κόμβων. Ο χρήστης μέσω του CLI Εικόνα 7 του ONOS έχει την δυνατότητα να τροποποιήσει τον τρόπο με τον οποίο λειτουργεί ο ελεγκτής, όπως για παράδειγμα να μην υπάρχουν εγκατεστημένα στους κόμβους και τα switch οι κανόνες δρομολόγησης εξαρχής (Proactive Forward). Επίσης και μέσω του web interface έχει την δυνατότητα να παρατηρήσει τις αλλαγές στην συνδεσμολογία Εικόνα 8, και να δει τις συσκευές οι οποίες είναι συνδεδεμένες Εικόνα 9, καθώς επίσης να δει και τους κανόνες δρομολόγησης οι οποίοι είναι σε χρήση Εικόνα 10.



Εικόνα 7 ONOS CLI

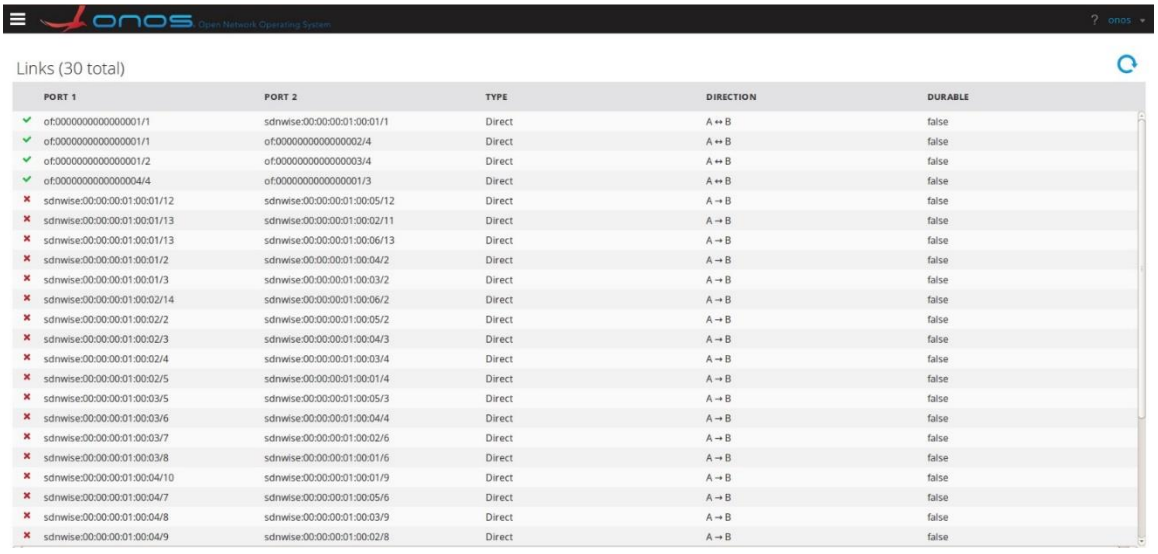


Εικόνα 8 Τοπολογία του δικτύου



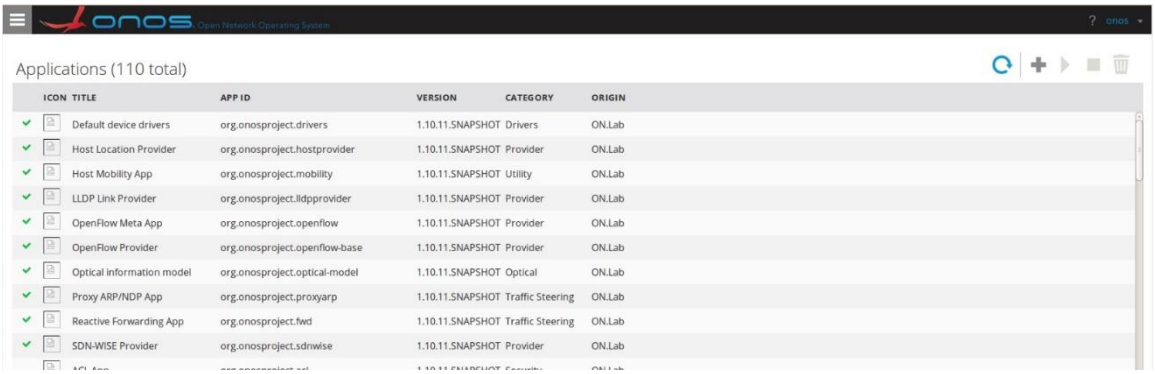
	FRIENDLY NAME	DEVICE ID	MASTER	PORTS	VENDOR	H/W VERSION	S/W VERSION	PROTOCOL
✓	of:0000000000000001	of:0000000000000001	127.0.0.1	4	Nicira, Inc.	Open vSwitch	2.0.2	OF_10
✓	of:0000000000000002	of:0000000000000002	127.0.0.1	5	Nicira, Inc.	Open vSwitch	2.0.2	OF_10
✓	of:0000000000000003	of:0000000000000003	127.0.0.1	5	Nicira, Inc.	Open vSwitch	2.0.2	OF_10
✓	of:0000000000000004	of:0000000000000004	127.0.0.1	5	Nicira, Inc.	Open vSwitch	2.0.2	OF_10
✗	sdnwise:00:00:00:01:00:01	sdnwise:00:00:00:01:00:01	127.0.0.1	0	CNIT	1.0	1.0	
✗	sdnwise:00:00:00:01:00:02	sdnwise:00:00:00:01:00:02	127.0.0.1	0	CNIT	1.0	1.0	
✗	sdnwise:00:00:00:01:00:03	sdnwise:00:00:00:01:00:03	127.0.0.1	0	CNIT	1.0	1.0	
✗	sdnwise:00:00:00:01:00:04	sdnwise:00:00:00:01:00:04	127.0.0.1	0	CNIT	1.0	1.0	
✗	sdnwise:00:00:00:01:00:05	sdnwise:00:00:00:01:00:05	127.0.0.1	0	CNIT	1.0	1.0	

Εικόνα 9 Συσκευές του δικτύου



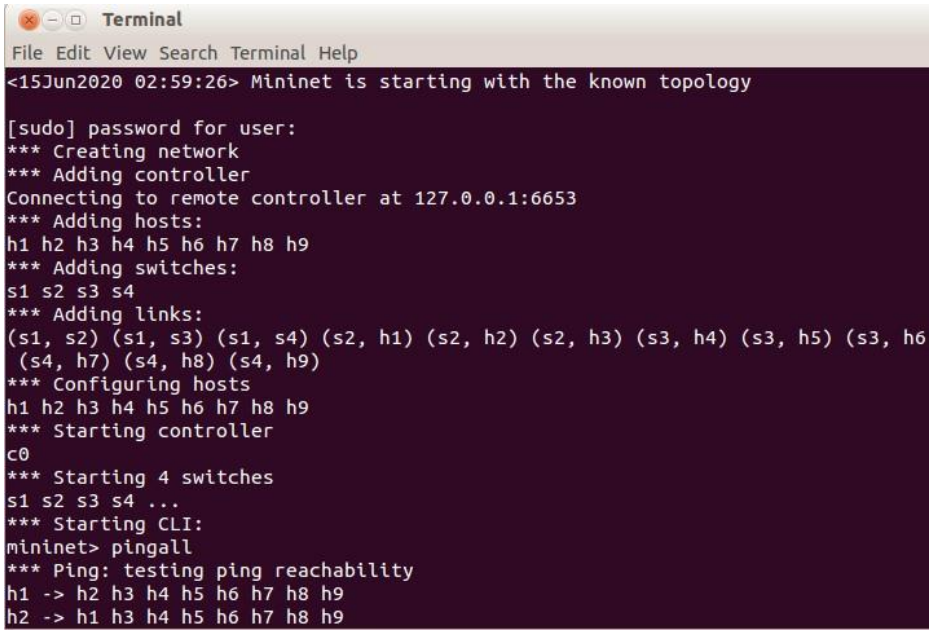
PORT 1	PORT 2	TYPE	DIRECTION	DURABLE
of:0000000000000001/1	sdnwise:00:00:00:01:00:01/1	Direct	A ↔ B	false
of:0000000000000001/1	of:0000000000000002/4	Direct	A ↔ B	false
of:0000000000000001/2	of:0000000000000003/4	Direct	A ↔ B	false
of:0000000000000004/4	of:0000000000000001/3	Direct	A ↔ B	false
sdnwise:00:00:00:01:00:01/12	sdnwise:00:00:00:01:00:05/12	Direct	A → B	false
sdnwise:00:00:00:01:00:01/13	sdnwise:00:00:00:01:00:02/11	Direct	A → B	false
sdnwise:00:00:00:01:00:01/13	sdnwise:00:00:00:01:00:06/13	Direct	A → B	false
sdnwise:00:00:00:01:00:01/2	sdnwise:00:00:00:01:00:04/2	Direct	A → B	false
sdnwise:00:00:00:01:00:01/3	sdnwise:00:00:00:01:00:03/2	Direct	A → B	false
sdnwise:00:00:00:01:00:02/14	sdnwise:00:00:00:01:00:06/2	Direct	A → B	false
sdnwise:00:00:00:01:00:02/2	sdnwise:00:00:00:01:00:05/2	Direct	A → B	false
sdnwise:00:00:00:01:00:02/3	sdnwise:00:00:00:01:00:04/3	Direct	A → B	false
sdnwise:00:00:00:01:00:02/4	sdnwise:00:00:00:01:00:03/4	Direct	A → B	false
sdnwise:00:00:00:01:00:02/5	sdnwise:00:00:00:01:00:01/4	Direct	A → B	false
sdnwise:00:00:00:01:00:03/5	sdnwise:00:00:00:01:00:05/3	Direct	A → B	false
sdnwise:00:00:00:01:00:03/6	sdnwise:00:00:00:01:00:04/4	Direct	A → B	false
sdnwise:00:00:00:01:00:03/7	sdnwise:00:00:00:01:00:02/6	Direct	A → B	false
sdnwise:00:00:00:01:00:03/8	sdnwise:00:00:00:01:00:01/6	Direct	A → B	false
sdnwise:00:00:00:01:00:04/10	sdnwise:00:00:00:01:00:01/9	Direct	A → B	false
sdnwise:00:00:00:01:00:04/7	sdnwise:00:00:00:01:00:05/6	Direct	A → B	false
sdnwise:00:00:00:01:00:04/8	sdnwise:00:00:00:01:00:03/9	Direct	A → B	false
sdnwise:00:00:00:01:00:04/9	sdnwise:00:00:00:01:00:02/8	Direct	A → B	false

Εικόνα 10 Κανόνες που είναι σε χρήση

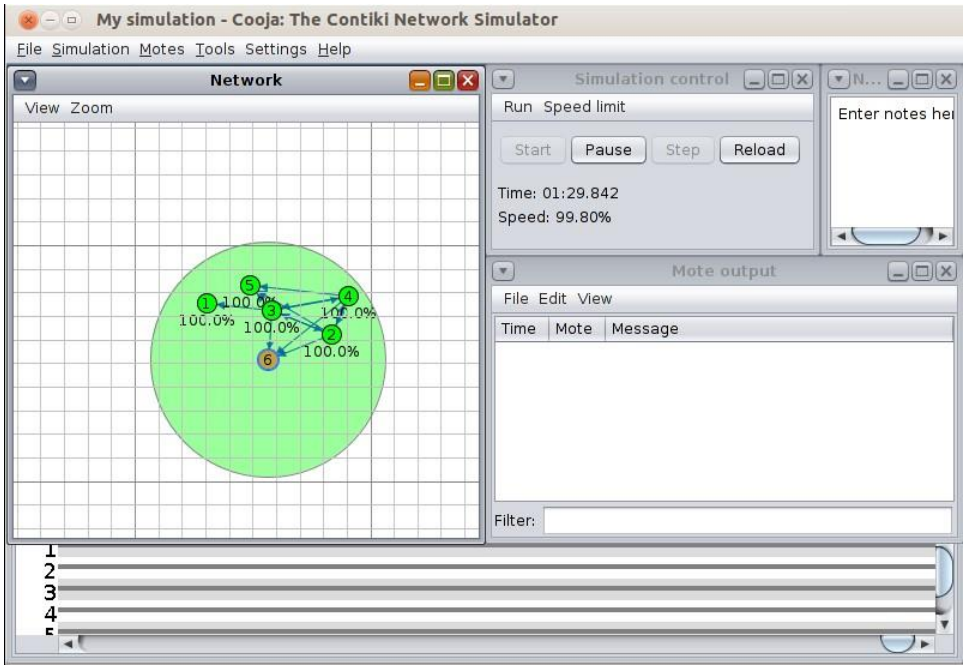


Εικόνα 11 Εφαρμογές του ONOS που είναι ενεργές

Η προσομοίωση εκτελείται μέσω του Cooja Εικόνα 13 από όπου ο χρήστης εισάγει τους κόμβους και έχει δυνατότητα να προκαλεί αλλαγές στο δίκτυο. Επίσης, για έλεγχο ετερογενών δικτύων χρησιμοποιείται το mininet Εικόνα 12 το οποίο επιτρέπει την δημιουργία εικονικών δικτύων με χρήση εικονικών switch.

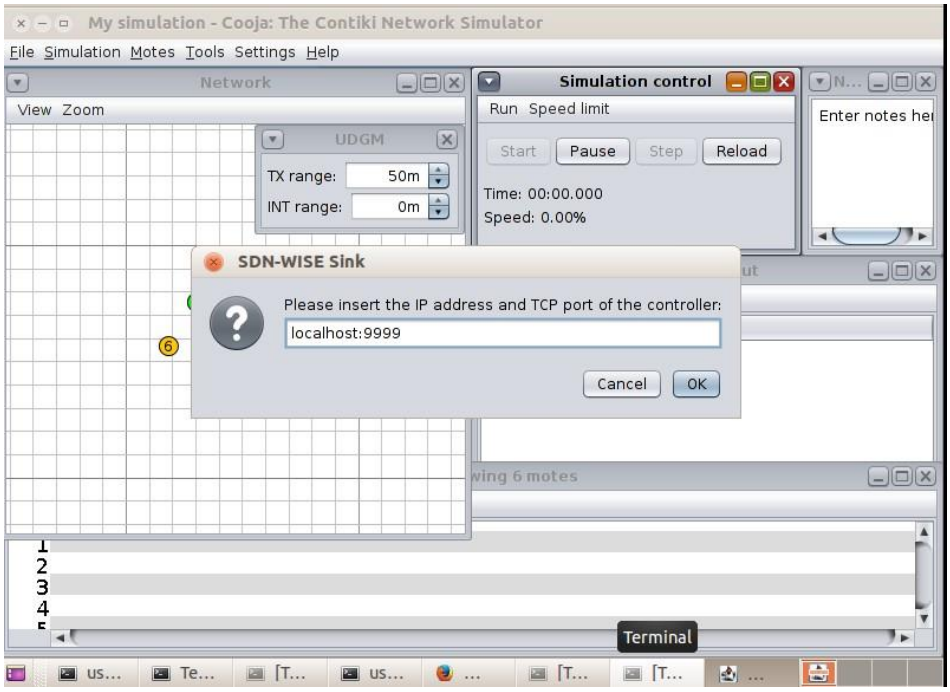


Εικόνα 12 Mininet



Εικόνα 13 Cooja Simulator

Η υλοποίηση του SDN-WISE στο Cooja χρησιμοποιεί δυο διαφορετικούς προσομοιωμένους κόμβους οι οποίοι είναι υλοποιημένοι σε Java. Οι διαφορετικοί αυτοί κόμβοι είναι οι 1) SDN-WISE Sink, ο οποίος σαν σκοπό έχει να διαχειρίζεται όλη την επικοινωνία με τον ελεγκτή, όπως και να στέλνει δεδομένα στον ελεγκτή σχετικά με το δίκτυο (αναγνώριση τοπολογίας. Επίσης, ο Sink είναι ο μοναδικός κόμβος ο οποίος έχει επικοινωνία με τον controller. Προκειμένου να μπορεί να επικοινωνεί ο Sink με τον controller στην αρχή της προσομοίωσης ο χρήστης θα πρέπει να εισάγει την IP και την πόρτα στην οποία λειτουργεί ο ελεγκτής, στο παράδειγμα αυτό η TCP πόρτα είναι η 9990 Εικόνα 14.



Εικόνα 14 Αρχικοποίηση Sink Node

2) Οι κόμβοι SDN-WISE (Motes) συλλέγουν δεδομένα από τους αισθητήρες τους και ανταλλάσσουν πακέτα μεταξύ τους προκειμένου να γνωρίζουν πως θα διαχειριστούν κάποιο εισερχόμενο πακέτο χρειάζεται να λάβουν νέους κανόνες από τον ελεγκτή. Η επικοινωνία αυτή πραγματοποιείται μέσω του Sink. Οι κόμβοι, όπως και ο Sink, είναι υλοποιημένοι σε Java. Ο ελεγκτής στην υλοποίηση αυτή είναι ο ONOS, ο οποίος διαχειρίζεται και τα δύο ετερογενή δίκτυα.

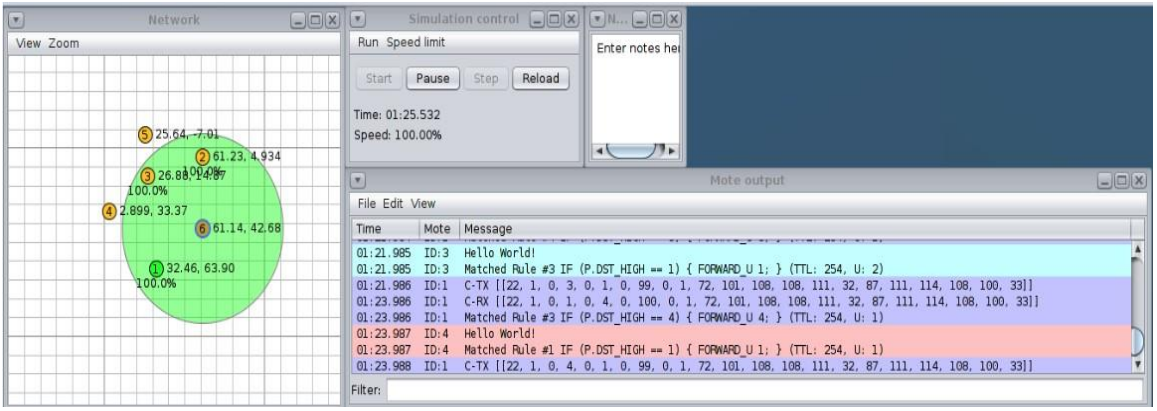
Προκειμένου να εκτελεστεί η προσομοίωση ο χρήστης μέσω των εκτελέσιμων script που δημιουργήθηκαν στα πλαίσια της παρούσας εργασίας και υπάρχουν διαθέσιμα στο Παράρτημα Α μπορεί να εγκαταστήσει και να εκτελέσει τα απαραίτητα αρχεία για να ξεκινήσει η προσομοίωση. Για να το κάνει αυτό προτείνεται η χρήση λειτουργικού συστήματος Ubuntu 16.04 . Για να εγκατασταθούν τα απαραίτητα αρχεία ο χρήστης πρέπει να εκτελέσει το script: **install_SDN_WISE_ONOS.sh**

και για να εκτελέσει την προσομοίωση: **start_ONOS.sh**

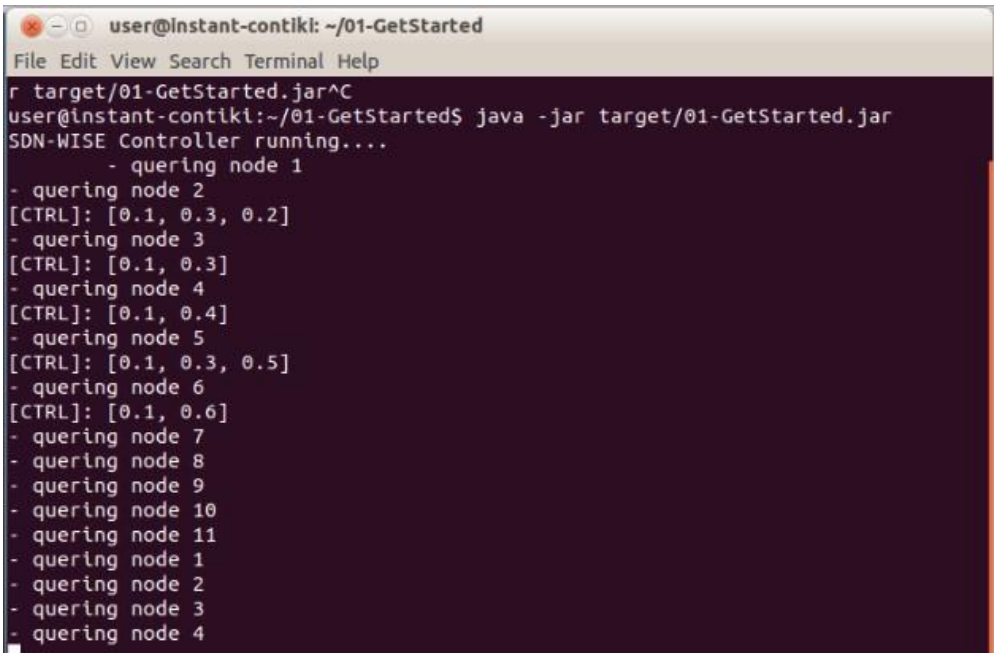
3.2.9.3 SDN-WISE υλοποίηση GetStarted

Η υλοποίηση αυτή δημιουργήθηκε προκειμένου να μπορεί ο χρήστης να πραγματοποιήσει προσομοίωση του SDN-WISE με την χρήση του Cooja Εικόνα 15. Η υλοποίηση αυτή χρησιμοποιεί προσομοιωμένους κόμβους υλοποιημένους σε Java οι οποίοι είναι: Ο SDN-WISE Sink ο οποίος σαν σκοπό να διαχειρίζεται όλη την επικοινωνία με τον ελεγκτή, όπως και στέλνει δεδομένα στον ελεγκτή σχετικά με το δίκτυο (αναγνώριση τοπολογίας) και οι

SDN-WISE κόμβοι (Motes) οι οποίοι συλλέγουν δεδομένα από τους αισθητήρες τους και ανταλλάσσουν πακέτα μεταξύ τους. Στη συγκεκριμένη υλοποίηση χρησιμοποιείται ένας ελεγκτής υλοποιημένος σε Java ο οποίος εποπτεύει το δίκτυο Εικόνα 16.



Εικόνα 15 Προσομοίωση των κόμβων στο Cooja



Εικόνα 16 Ελεγκτής

Προκειμένου να εκτελεστεί η προσομοίωση, ο χρήστης μέσω των εκτελέσιμων script που δημιουργήθηκαν στα πλαίσια της παρούσας εργασίας και υπάρχουν διαθέσιμα στο Παράρτημα Α, μπορεί να εγκαταστήσει τα απαραίτητα αρχεία για να ξεκινήσει η προσομοίωση. Για να το κάνει αυτό προτείνεται η χρήση λειτουργικού συστήματος Ubuntu 16.04 . Για να εγκατασταθούν τα απαραίτητα αρχεία ο χρήστης πρέπει να εκτελέσει το script: **install_SDN_WISE_simple_controler.sh**

και για να εκτελέσει την προσομοίωση: **startSDN_Java.sh**

3.3 CORAL

3.3.1 Τι είναι το CORAL

Το CORAL (Cross-Layer Control of Data Flows) είναι μια πλατφόρμα η οποία αναπτύχθηκε στο Πανεπιστήμιο Μακεδονίας και παρέχει στον χρήστη δυνατότητες SDN προκειμένου να υλοποιήσει βελτιωμένες εφαρμογές QoE (Quality of Experience) για τους χρήστες και QoS (Quality Of Service) για τις εφαρμογές που αναπτύσσονται στο Διαδίκτυο των Αντικειμένων [4]. Παραδείγματα εφαρμογών στα οποία το QoS είναι βασικό χαρακτηριστικό είναι σε εφαρμογές οι οποίες έχουν να κάνουν με συνεχή επικοινωνία, δηλαδή με εφαρμογές όπως υγείας ή ασφάλειας που θέλουμε να λαμβάνουμε συνεχώς δεδομένα ακόμα και στα πιο δύσκολα περιβάλλοντα.

Προκειμένου να το πετύχει αυτό το CORAL παρέχει στον χρήστη διάφορα χαρακτηριστικά. Συνοπτικά, τα χαρακτηριστικά είναι τα παρακάτω:

- Αρχιτεκτονική αντίστοιχη με SDN που το επίπεδο των δεδομένων είναι ανεξάρτητο από το επίπεδο Ελέγχου.
- **CORAL Dashboard:** Γραφικό περιβάλλον για τον χρήστη, μέσω του οποίου ο υπάρχει δυνατότητα ελέγχου και παρακολούθησης της υλοποίησης.
- Δυο διαφορετικά δικτυακά πρωτόκολλα:
    - **CORAL-SDN:** Ένα πρωτόκολλο παρόμοιο με το OpenFlow το οποίο το οποίο επιτυγχάνει έλεγχο στην τοπολογία και στρατηγικές δρομολόγησης προσαρμοσμένες στο IoT.
    - **Adaptable RPL**: Μια προσαρμοσμένη έκδοση του IPv6 πρωτόκολλου δρομολόγησης RPL για δίκτυα με χαμηλής ισχύος με απώλειες
- **CORAL Controller:** Είναι ο SDN ελεγκτής στο CORAL ο οποίος παρέχει κεντρικό έλεγχο σε όλα τα επίπεδα του δικτύου για τις ροές των δεδομένων, αλλάζει δυναμικά τους κανόνες δρομολόγησης προκειμένου να πετυχαίνει την βέλτιστη ροή δεδομένων εντός του δικτύου.

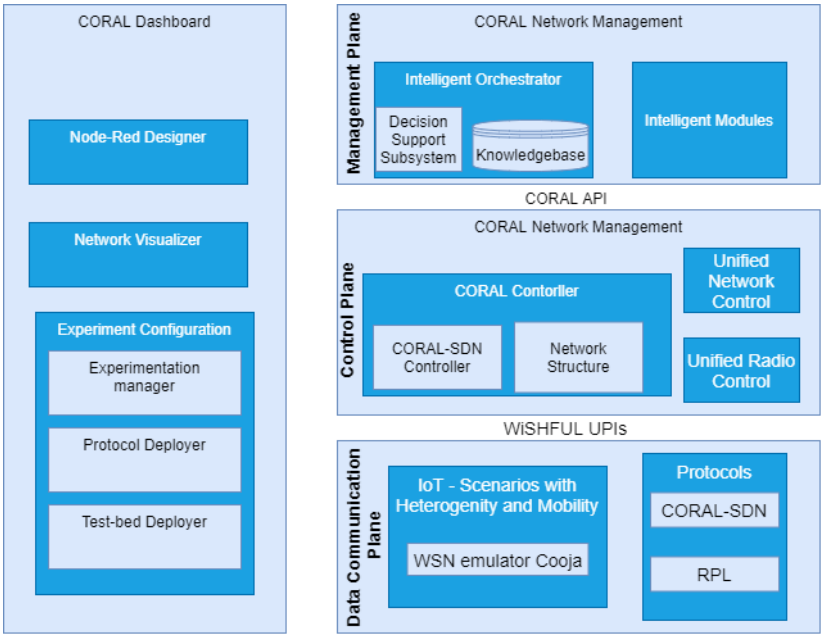
Το CORAL αναπτύχθηκε προκειμένου να καλύψει τους 5 παρακάτω τομείς και να πετύχει την αρμονική λειτουργία μεταξύ των διαφορετικών πρωτοκόλλων και εφαρμογών:

- **Ευέλικτη Διαχείριση του δικτύου (Elasticity):** Δυναμική προσαρμοστικότητα στις αλλαγές του δικτύου.
- **Λειτουργία όλων των επιπέδων (Cross-layer operation):** Εναρμόνιση της διαμόρφωσης των διαφορετικών πρωτοκόλλων και των διαφορετικών επιπέδων.

- **Ευφυΐα (Intelligence):** Η παροχή μηχανισμών υψηλού επιπέδου οι οποίοι μπορούν να λαμβάνουν σαν είσοδο δεδομένα από το δίκτυο και να παρέχουν σαν έξοδο του κανόνες δρομολόγησης ή στρατηγικές για την αποδοτικότερη λειτουργία του πρωτοκόλλου.
- **Ετερογένεια (Heterogenity):** Να παρέχει την δυνατότητα επικοινωνίας μεταξύ διαφορετικού hardware ή λογισμικού
- **Μετακινούμενοι κόμβοι (Mobility):** Να διαχειρίζεται τους κόμβους οι οποίοι μετακινούνται εντός του δικτύου, πράγμα που απαιτεί συνεχή έλεγχο στην τοπολογία του δικτύου για τις αλλαγές που προκύπτουν.

3.3.2 Αρχιτεκτονική του CORAL

Το CORAL προκειμένου να λειτουργήσει αποτελείται από κάποια επιμέρους στοιχεία τα οποία του δίνουν τα διαφορετικά χαρακτηριστικά τα οποία εκμεταλλεύεται προκειμένου να καταφέρει να καλύψει τους 5 βασικούς τομείς οι οποίοι είναι Elasticity, Cross-layer Operation, Intelligence, Heterogenity, Mobility. Προκειμένου να το πετύχει αυτό το CORAL χρησιμοποιεί διαφορετικά δικτυακά πρωτόκολλα και ενσωματώνει μέσα στα διαφορετικά επίπεδα που χρησιμοποιεί διάφορες επιμέρους μονάδες οι οποίες του επιτρέπουν να καλύψει τις απαιτήσεις διάφορων εφαρμογών. Τα επίπεδα τα οποία συνθέτουν το CORAL είναι διακριτά όπως και στο SDN όπου το επίπεδο ελέγχου (Control Plane) και το επίπεδο των δεδομένων (Data Plane) διαχωρίζονται [21] . Έτσι, αντίστοιχα το CORAL ακολουθεί την αρχιτεκτονική των SDN, χωρίζεται σε 3 διαφορετικά επίπεδα. Τα επίπεδα: Επίπεδο Διαχείρισης Δικτύου, Επίπεδο Ελέγχου, Επίπεδο Επικοινωνίας Δεδομένων. Στο κάθε επίπεδο υπάρχουν και επιμέρους στοιχεία τα πιο βασικά από τα οποία είναι: Ελεγκτής CORAL, CORAL Dashboard, δικτυακά πρωτόκολλα RPL και CORAL-SDN. Μια απεικόνιση της αρχιτεκτονικής είναι η παρακάτω:



Εικόνα 17. Αρχιτεκτονική CORAL

3.3.2.1 Επίπεδο διαχείρισης δικτύου

Είναι το επίπεδο στο οποίο γίνεται όλη η υψηλού επιπέδου διαχείριση του δικτύου. Ουσιαστικά το επίπεδο αυτό είναι υπεύθυνο για την έξυπνη διαχείριση του δικτύου η οποία επιτρέπει στο CORAL να υλοποιήσει κάποιους από τους στόχους του, όπως την Ευέλικτη διαχείριση του δικτύου (Elasticity), την Ετερογένεια (Heterogenity) του δικτύου, καθώς και να διαχειρίζεται αποτελεσματικά τους Μετακινούμενους κόμβους (Mobility) [4]. Προκειμένου να πετυχαίνει τους στόχους αυτούς το CORAL μεταφέρει την πολυπλοκότητα από το Επίπεδο Επικοινωνίας Δεδομένων στο επίπεδο Ελέγχου δηλαδή στον CORAL Controler [22]. Με τον τρόπο αυτό το CORAL επιτυγχάνει την ισορροπία μεταξύ ευέλικτης διαχείρισης και της πολυπλοκότητας του δικτύου, καθώς λαμβάνει υπόψη του τα βασικά χαρακτηριστικά του ασύρματου δικτύου με κινητούς κόμβους (θέματα σήματος, αποσυνδέσεις από το δίκτυο). Έτσι επιτυγχάνει να διαχειρίζεται την ετερογένεια, όμως υποστηρίζει την ευέλικτη και δυναμική διαμόρφωση των συσκευών IoT προκειμένου να βελτιώσει την απόδοσης και κατανομή των πόρων και ως εκ τούτου διαχειρίζεται τους Κινητούς Κόμβους (Mobility) .

Όπως αναφέραμε ήδη, μέσα στο Επίπεδο διαχείρισης δικτύου υπάρχει η υψηλού επιπέδου διαχείριση του δικτύου. Προκειμένου να λειτουργήσει η διαχείριση αυτή υπάρχουν κάποια βασικά στοιχεία. Τα στοιχεία αυτά είναι: Ο Ευφυής Ενορχηστρωτής (Intelligent Orchestrator) ο οποίος είναι το βασικό στοιχείο του συγκεκριμένου επιπέδου, καθώς και

στοιχεία τα οποία προσφέρουν επιπλέον ευφυή εργαλεία και αλγορίθμους. Αναλυτικά, κατά τη λειτουργία του δικτύου ο Ευφυής Ενορχηστρωτής αναλαμβάνει να ορίσει σε υψηλό επίπεδο τις προϋποθέσεις που πρέπει να καλυφθούν. Μόλις οριστικοποιηθούν οι προϋποθέσεις, τότε το Υποσύστημα Υποστήριξης Αποφάσεων (Decision Support Subsystem) ορίζει τις ενέργειες οι οποίες πρέπει να πραγματοποιηθούν προκειμένου να λάβει την απόφαση. Το Υποσύστημα Υποστήριξης αποφάσεων λαμβάνει υπόψιν του την κατάσταση του δικτύου και τους κανόνες οι οποίοι είναι αποθηκευμένοι στη Βάση Δεδομένων. Προκειμένου οι αποφάσεις να είναι πιο εξειδικευμένες και πιο ευφυείς χρησιμοποιούνται τα επιπλέον ευφυή εργαλεία:

3.3.2.2 Επίπεδο Ελέγχου

Είναι το επίπεδο στο οποίο πραγματοποιείται ο έλεγχος του δικτύου. Βασικό στοιχείο του επιπέδου ελέγχου είναι ο ελεγκτής (CORAL Controler) . Σε αυτό το επίπεδο είναι που υπάρχει η πληροφορία σχετικά με την δομή του δικτύου. Ουσιαστικά το Επίπεδο Ελέγχου δρα ως συνδετικός κρίκος μεταξύ του Επιπέδου Διαχείρισης και του Επιπέδου Επικοινωνίας Δεδομένων. Ο Ελεγκτής δημιουργεί και κρατάει ενημερωμένο έναν γράφο του δικτύου όπου οι κόμβοι και συνδέσεις μεταξύ των κόμβων αποτελούν τις δικτυακές συσκευές και την ασύρματη σύνδεση μεταξύ τους.

Ο γράφος είναι αποθηκευμένος στη μονάδα Δομής Δικτύου (Network Structure) και ενημερώνεται με πληροφορίες όπως πχ η χρήση της μπαταρίας. Στο επίπεδο ελέγχου ενσωματώνονται και Modules που επιτρέπουν στο CORAL να διαχειρίζεται την ετερογένεια του δικτύου με χρήση του Ενοποιημένου Ελέγχου Δικτύου και Μετάδοσης (Unified Network and Radio Control) τα οποία λαμβάνουν δεδομένα από το API του CORAL. Άξιο αναφοράς επίσης είναι ότι ο ελεγκτής βασίζεται στην WiSHFUL πλατφόρμα που του παρέχει Ενοποιημένο Έλεγχο Δικτύου και Μετάδοσης (Unified Radio and Network Control) . Το WiSHFUL είναι μία πλατφόρμα η οποία παρέχει ένα σύνολο από Ενοποιημένες εντολές [15].

Ο Ελεγκτής του CORAL είναι ουσιαστικά ο SDN ελεγκτής στο δίκτυο, όπως και στα κλασσικά SDN ο ελεγκτής είναι υπεύθυνος για τον κεντρικό έλεγχο της ροής των δεδομένων. Επίσης, λαμβάνει αποφάσεις και ορίζει δυναμικά τους κανόνες προώθησης πακέτων. Προκειμένου να δημιουργήσει τους κανόνες αυτούς χρησιμοποιεί τα δεδομένα τα οποία έχει στην διάθεση του, τα οποία είναι η δομή του δικτύου, καθώς και οι

αποφάσεις από το Υποσύστημα Υποστήριξης Αποφάσεων (Decision Support Subsystem) και τα δεδομένα από τα επιπλέον ευφυή εργαλεία.

3.3.2.3 Επίπεδο Επικοινωνίας Δεδομένων

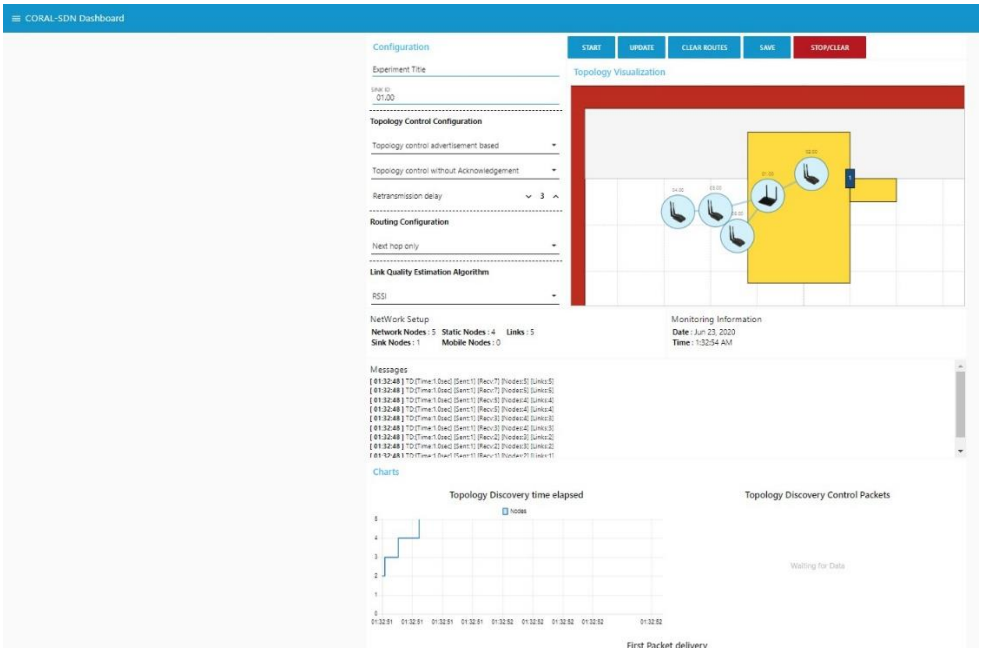
Είναι το επίπεδο το χαμηλότερο επίπεδο στην αρχιτεκτονική του CORAL το οποίο περιλαμβάνει το πεδίο στο οποίο πραγματοποιείται το πείραμα (πολλοί φυσικοί κόμβοι IoT ή εξομοιωμένοι μέσω του COOJA), καθώς και το πρωτόκολλο (CORAL-SDN ή RPL) το οποίο χρησιμοποιούμε για το πείραμα. Γενικά το επίπεδο αυτό περιλαμβάνει δεδομένα σχετικά με την συμπεριφορά των κόμβων προκειμένου να αποκαλυφθούν οι δυνατότητες λειτουργίας όλων των επιπέδων του, καθώς και η προτεινόμενη αρχιτεκτονική μέσω της διαμόρφωσης διαφορετικών πρωτοκόλλων.

3.3.2.4 CORAL Dashboard

Είναι το γραφικό περιβάλλον το οποίο έχει σχεδιαστεί προκειμένου ο Χρήστης να μπορεί να αλληλοεπιδρά με το CORAL, καθώς και να παρακολουθεί τα αποτελέσματα. Έχει σχεδιαστεί σε Node-RED και παρέχει στον χρήστη διάφορες δυνατότητες [23].

Ο χρήστης μέσω του Dashboard έχει τη δυνατότητα να διαλέξει ένα από τα δυο διαθέσιμα πρωτόκολλα και να παρακολουθεί τα αποτελέσματα ανάλογα με το πρωτόκολλο που επέλεξε μέσω της οπτικοποίησης των δεδομένων. Προκειμένου να παρέχει τις δυνατότητες αυτές το Dashboard χρησιμοποιεί τρία υποσυστήματα [22]:

1. **Experiment Configuration Dashboard:** Παρέχει τις δυνατότητες ρύθμισης ανάλογα με τα διαθέσιμα πρωτόκολλα
2. **Network Visualizer:** όπου αποτυπώνεται η τοπολογία του δικτύου και τα αποτελέσματα, καθώς και η πρόοδος του πειράματος
3. **Node-RED Designer:** Όπου παρέχει μια βιβλιοθήκη με τα βασικά χαρακτηριστικά του CORAL, τα οποία εμφανίζονται ως κόμβοι του Node-Red, και τις ροές. Έτσι, δίνεται η δυνατότητα στον χρήστη να αλλάξει εύκολα τα χαρακτηριστικά του CORAL.



Εικόνα 18. CORAL Dashboard

3.3.2.5 Πρωτόκολλα που υποστηρίζει και χρησιμοποιεί το CORAL

Όπως έχουμε αναφέρει ήδη το CORAL χρησιμοποιεί δυο διαφορετικά δικτυακά πρωτόκολλα το CORAL-SDN και το Adaptable-RPL.

3.3.3 Πρωτόκολλο CORAL-SDN

Το CORAL-SDN είναι ένα πρωτόκολλο το οποίο δημιουργήθηκε για να προσφέρει προσαρμοστικότητα με ένα ευρύ φάσμα εφαρμογών IoT, καθώς και σε περιορισμούς του δικτύου όπως θέματα με Κινητούς κόμβους και θέματα με τη συνδεσιμότητα των κόμβων. Προκειμένου να λειτουργήσει το CORAL-SDN χρησιμοποιεί τέσσερις εναλλακτικούς μηχανισμούς

1. Δυο διαφορετικούς αλγόριθμους ανακάλυψης και ελέγχου τοπολογίας, ο κάθε ένας από τους οποίους έχει διαφορετικό σκοπό. Ο πρώτος βασίζεται στην διαφήμιση των κόμβων (TC-NA) και ο δεύτερος στα αιτήματα των γειτόνων (TC-NR) [23]
2. Δυο διαφορετικούς αλγορίθμους για την εγκαθίδρυση ροής. Όπου διαμορφώνεται η πλήρης διαδρομή ή ο επόμενος κόμβος μόνο. Για παράδειγμα, σε μια σταθερή τοπολογία με έναν κινητό κόμβο, είναι δαπανηρή η διαμόρφωση ολόκληρης της διαδρομής κάθε φορά που ένας μόνο κόμβος αλλάζει το σημείο προσάρτησης του. Επομένως, χρησιμοποιείται, η δεύτερη επιλογή η οποία είναι η διαμόρφωση του επόμενου κόμβου, που είναι πιο κατάλληλη σε αυτή την περίπτωση [23].

Τέλος, το CORAL-SDN υποστηρίζει επίσης άλλες επιλογές διαμόρφωσης. Όπως για παράδειγμα, την μέθοδο εκτίμησης ποιότητας συνδέσμου, τη χρήση αναγνωρίσεων και το χρόνο διαστήματος των μηνυμάτων ελέγχου για τον έλεγχο τοπολογίας.

3.3.4 Πρωτόκολλο Adaptable-RPL

Το RPL θεωρείται ότι είναι το πιο κατάλληλο πρωτόκολλο για χρήση στο IoT. Το RPL οργανώνει τους κόμβους σε έναν γράφο ο οποίος ονομάζεται Destination-Oriented Directed Acyclic Graph (DODAG), στον οποίο όλοι οι κόμβοι καταλήγουν σε ένα μοναδικό προορισμό ο οποίος ονομάζεται root ή Sink. Το RPL συντηρεί τον γράφο DODAG με ένα αλγόριθμο τον trickle timer ο οποίος συγχρονίζει την μετάδοση των μηνυμάτων ανακάλυψης της τοπολογίας (DIO) [6]. Επειδή το RPL δεν υποστηρίζει πλήρως κάποιες εφαρμογές όπως π.χ. τους Κινητούς κόμβους, εάν ένας κόμβος μετακινηθεί εκτός της εμβέλειας του προηγούμενου κόμβου, τότε αποσυνδέεται από τον γράφο και αυτό δημιουργεί πρόβλημα στην μετάδοση των δεδομένων αλλά και συνολικά στην λειτουργία του δικτύου. Το CORAL για να επιλύσει το πρόβλημα αυτό αλλάζει τις παραμέτρους του αλγορίθμου trickle timer δυναμικά και ξεχωριστά για κάθε κόμβο [4].

3.3.5 Ελεγκτής CORAL

Ο Ελεγκτής του CORAL λειτουργεί όπως ακριβώς ένας ελεγκτής του SDN. Βασική λειτουργία του ελεγκτή είναι να παρέχει κεντρικό διαχείριση στις ροές του δικτύου. Ο ελεγκτής του CORAL παρέχει τις παρακάτω λειτουργίες:

1. διατηρεί την τοπολογία του δικτύου στο Network Modeler Module αξιοποιώντας τους διαφορετικούς αλγόριθμους ελέγχου τοπολογίας.
2. ελέγχει την ροή των δεδομένων στον δίκτυο.
3. Αλλάζει τις παραμέτρους του Πρωτοκόλλου που χρησιμοποιείται δυναμικά ανάλογα με τις απαιτήσεις.
4. Συλλέγει δεδομένα σχετικά με την συνδεσιμότητα των κόμβων, όπως με την ένταση των λαμβανόμενων σημάτων (Received Signal Strength Indicator) (RSSI) και τον δείκτη ποιότητας της σύνδεσης (Link Quality Indicator) (LQI).

Ο CORAL Ελεγκτής αναπτύχθηκε σε Java και σχεδιάστηκε έτσι ώστε να μπορεί να υποστηρίξει νέους αλγόριθμους και νέες ευφυείς δυνατότητες. Άξιο αναφοράς επίσης είναι ότι ο ελεγκτής βασίζεται στην WiSHFUL πλατφόρμα που του παρέχει Ενοποιημένο Έλεγχο Δικτύου και Μετάδοσης (Unified Radio and Network Control) [15]. Το WiSHFUL είναι μία πλατφόρμα η οποία παρέχει ένα σύνολο από ενοποιημένες εντολές

διεπαφής προγραμματισμού (Unified Programming Interface commands) (UPIs) οι οποίες παρέχουν δυνατότητες για τη συλλογή δεδομένων μέσα από δικτυακό πρωτόκολλο.

3.3.6 Πως λειτουργεί η Ανακάλυψη της τοπολογίας

Το πρωτόκολλο CORAL-SDN χρησιμοποιεί δύο διαφορετικούς αλγορίθμους ανακάλυψης τοπολογίας. Και στους δύο διαφορετικούς αλγορίθμους ο ελεγκτής ξεκινάει την διαδικασία ανακάλυψης της τοπολογίας στέλνοντας το πρώτο μήνυμα ανακάλυψης της τοπολογίας [22]. Ο ελεγκτής διατηρεί την τοπολογία σε μορφή γράφου στη μονάδα Δομής Δικτύου. Είναι πολύ σημαντικό ο ελεγκτής να γνωρίζει την τοπολογία του δικτύου προκειμένου να μπορεί να διαχειρίζεται αποδοτικά την ανταλλαγή των πακέτων εντός του δικτύου. Οι δύο διαφορετικού αλγόριθμοι τους οποίους αξιοποιεί το πρωτόκολλο CORAL-SDN είναι :

3.3.6.1 Ο αλγόριθμος ανακάλυψης της τοπολογίας ο οποίος βασίζεται στην διαφήμιση των κόμβων (TC-NA)

Ο ελεγκτής ξεκινάει την διαδικασία ανακάλυψης των γειτονικών κόμβων στέλνοντας το μήνυμα ανακάλυψης τοπολογίας στον πρώτο κόμβο. Ο κάθε κόμβος ο οποίος λαμβάνει το μήνυμα αυτό απαντάει μέσω broadcast. Έτσι, όλο το δίκτυο γεμίζει από μηνύματα ανακάλυψης τοπολογίας τα οποία ανταλλάσσονται μέσω broadcast και διακινούνται μεταξύ των κόμβων. Έτσι, μόλις ένας κόμβος λάβει το πακέτο ανακάλυψης τοπολογίας αμέσως στέλνει μέσω broadcast ένα μήνυμα το οποίο διαφημίζει την θέση του στους γειτονικούς κόμβους. Το μήνυμα αυτό περιέχει πληροφορίες σχετικές με τον κόμβο που το έστειλε. Ο κόμβος που λαμβάνει το πακέτο αυτό υπολογίζει:

- Την ένταση του σήματος.
- Την ποιότητα της ζεύξης μεταξύ των δυο.
- Την ταυτότητα του κόμβου που το έστειλε.

Και σχηματίζει ένα μήνυμα το οποίο είναι η απάντηση του κόμβου στο μήνυμα ανακάλυψης της τοπολογίας το οποίο στέλνει στον ελεγκτή και περιέχει τις εξής πληροφορίες:

- Την ένταση του σήματος.
- Την ποιότητα της ζεύξης μεταξύ των δυο.
- Τη στάθμη της μπαταρίας του.
- Την ταυτότητα του κόμβου.

Μόλις ο ελεγκτής λάβει την απάντηση τότε ανανεώνει τον γράφο του δικτύου που είναι αποθηκευμένος στη μονάδα Δομής Δικτύου (Network Structure).

3.3.6.2 Ο αλγόριθμος ανακάλυψης της τοπολογίας ο οποίος βασίζετε στα αιτήματα των γειτόνων (TC-NR)

Η διαδικασία του ελέγχου τοπολογίας με βάση τα αιτήματα των γειτονικών κόμβων ξεκινάει με ένα μήνυμα ανακάλυψης τοπολογίας το οποίο στέλνει ο ελεγκτής στον πρώτο κόμβο. Μόλις ο κόμβος λάβει το μήνυμα ανακάλυψης τοπολογίας, στέλνει νέο μήνυμα μέσω broadcast σε όλους τους γειτονικούς κόμβους οι οποίοι είναι εντός της εμβέλειας του πομπού του. Ο κάθε κόμβος ο οποίος λαμβάνει το broadcast μήνυμα στέλνει ένα νέο μήνυμα μέσω Unicast μόνο στον αποστολέα του broadcast μηνύματος το οποίο περιέχει πληροφορίες όπως:

- Την ένταση του σήματος.
- Την ποιότητα της ζεύξης μεταξύ των δυο.
- Την ταυτότητα του κόμβου που το έστειλε.

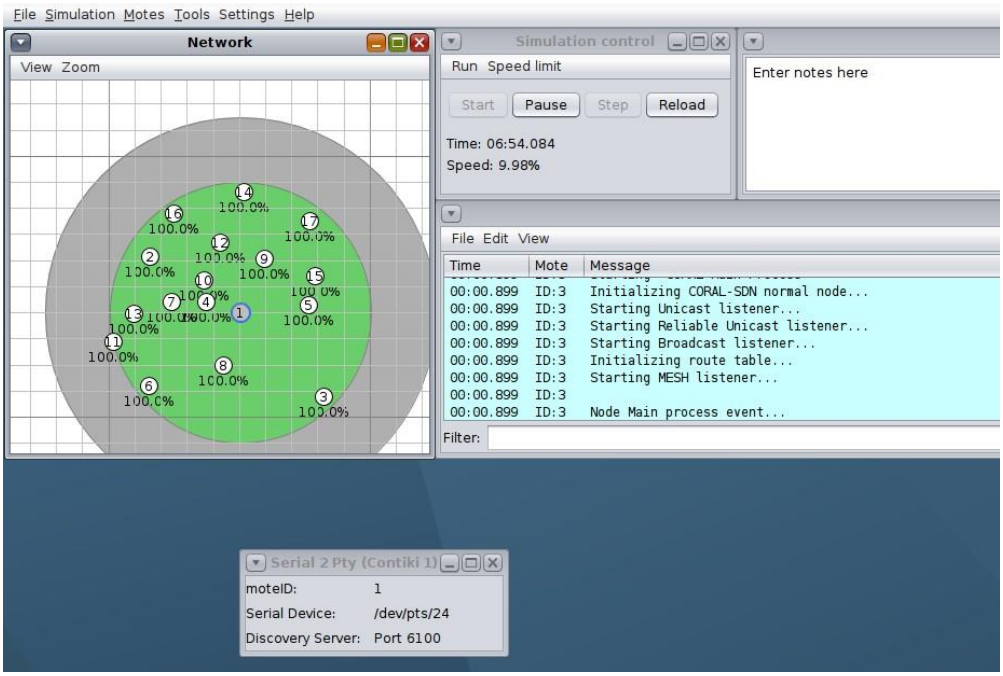
Ο παραλήπτης του μηνύματος αυτού απαντάει στα μηνύματα των γειτονικών κόμβων και ενημερώνει τον ελεγκτή ο οποίος ανανεώνει τον γράφο του δικτύου που είναι αποθηκευμένος στη μονάδα Δομής Δικτύου (Network Structure).

3.3.7 Χρήση CORAL

Το CORAL-SDN στην υλοποίησή του παρέχει τη δυνατότητα στον χρήστη να εκτελέσει τα σενάρια τα οποία επιθυμεί. Προκειμένου να το πετύχει αυτό χρησιμοποιεί βασικά στοιχεία:

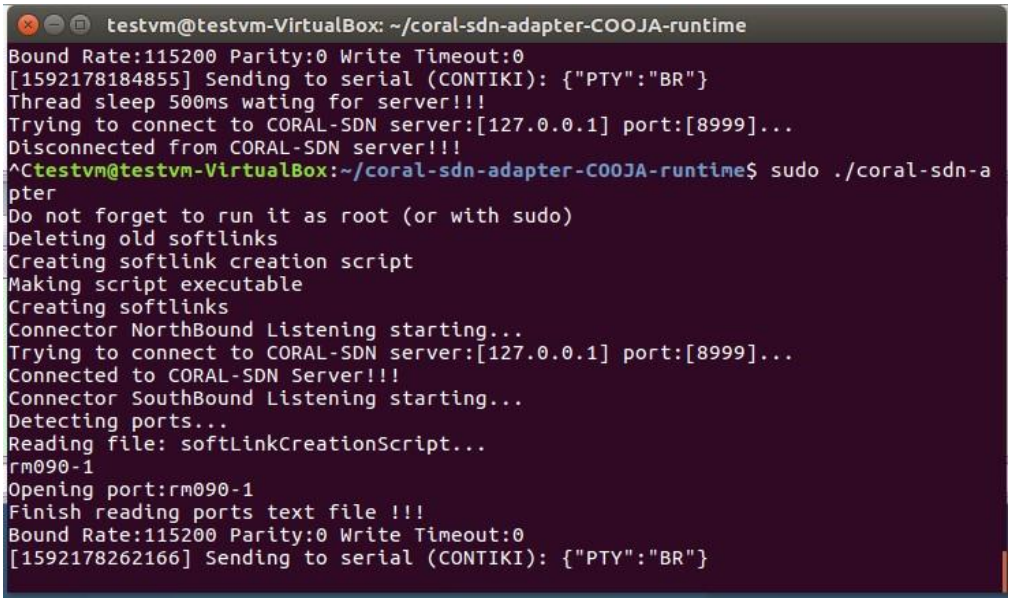
1) **Cooja simulator:** Όπου εκτελείται η προσομοίωση των κόμβων
2) **CORAL Adapter:** Είναι ένας προσαρμογέας ο οποίος είναι υλοποιημένος σε python και Java. Βασικός στόχος του είναι να επικοινωνεί με το border line router της προσομοίωσης μέσω σειριακής επικοινωνίας και με τον ελεγκτή CORAL.
3) **CORAL Controler:** Είναι ο ελεγκτής του δικτύου ο οποίος ελέγχει την τοπολογία και ρυθμίζει την εγκαθίδρυση των ροών στους κόμβους προκειμένου οι κόμβοι να μπορούν να διαχειρίζονται τα εισερχόμενα πακέτα.
4) **Dashboard:** Επικοινωνεί με τον ελεγκτή και αποτελεί το σημείο στο οποίο ο χρήστης παρατηρεί και ρυθμίζει τις παραμέτρους του δικτύου.

Αναλυτικά στο Cooja πραγματοποιείται η εκτέλεση της προσομοίωσης στην οποία συμμετέχουν δύο διαφορετικοί κόμβοι: τα CORAL Nodes, οι οποίοι είναι οι κόμβοι που συλλέγουν δεδομένα από τους αισθητήρες τους και ανταλλάσσουν πακέτα μεταξύ τους και ο κόμβος Border Line, ο οποίος είναι κάτι αντίστοιχο με τους Sink Nodes του SDN-WISE. Είναι ο κόμβος ο οποίος διαχειρίζεται όλη την επικοινωνία με τον ελεγκτή, όπως και στέλνει δεδομένα στον ελεγκτή σχετικά με το δίκτυο (αναγνώριση τοπολογίας). Επίσης, είναι ο μοναδικός κόμβος ο οποίος έχει επικοινωνία με τον ελεγκτή. Στην Εικόνα 19 βλέπουμε ένα στιγμιότυπο από την εκτέλεση της προσομοίωσης, όπου παρατηρούμε τον κόμβο 1 ο οποίος είναι ο Border line κόμβος, καθώς και τους υπολοίπους CORAL κόμβους. Επίσης, βλέπουμε στο κάτω μέρος την σειριακή επικοινωνία μεταξύ του κόμβου 1 και του CORAL adapter.



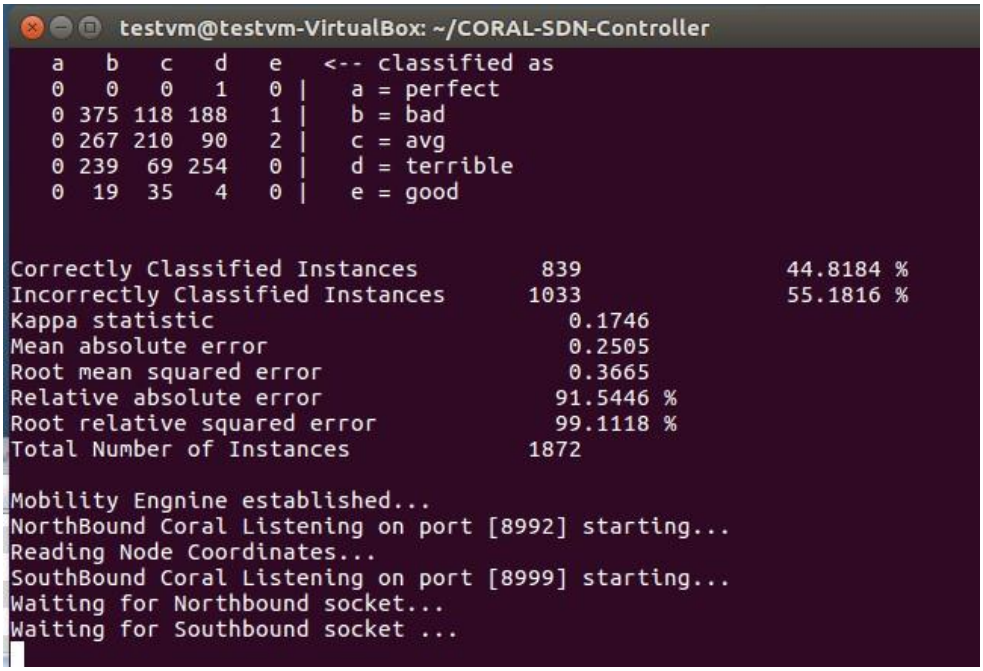
Εικόνα 19 Στιγμιότυπο από το Cooja

Ο CORAL adapter σαν σκοπό έχει να εξασφαλίσει την επικοινωνία μεταξύ του border line κόμβου και του ελεγκτή. Προκειμένου να γίνει αυτό χρησιμοποιεί σειριακή επικοινωνία με τον κόμβο και στέλνει τα δεδομένα τα οποία λαμβάνει από τον κόμβο στον ελεγκτή. Προκειμένου να επικοινωνήσουν σειριακά χρησιμοποιείται το serial2pty το οποίο καθιστά δυνατή την επικοινωνία αυτή.



Εικόνα 20 CORAL Adapter

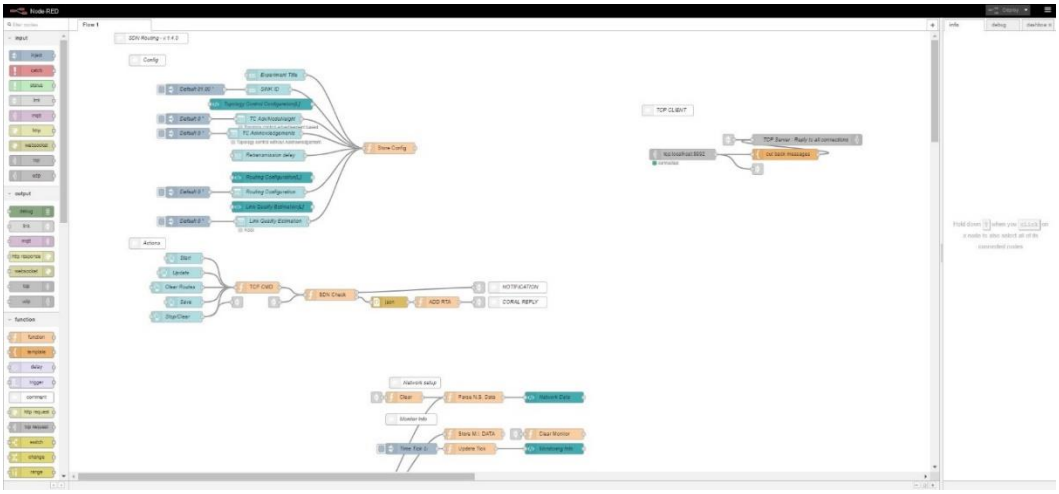
Ένα από τα πιο βασικά στοιχεία της υλοποίησης αυτής είναι ο ελεγκτής CORAL. Ο ελεγκτής είναι υλοποιημένος σε Java και σκοπό έχει να εγκαθιδρύσει τους κανόνες προώθησης πακέτων στους κόμβους και την προσαρμογή του πρωτοκόλλου που χρησιμοποιείται ανάλογα με τις αλλαγές που εφαρμόζει ο χρήστης μέσω του dashboard.



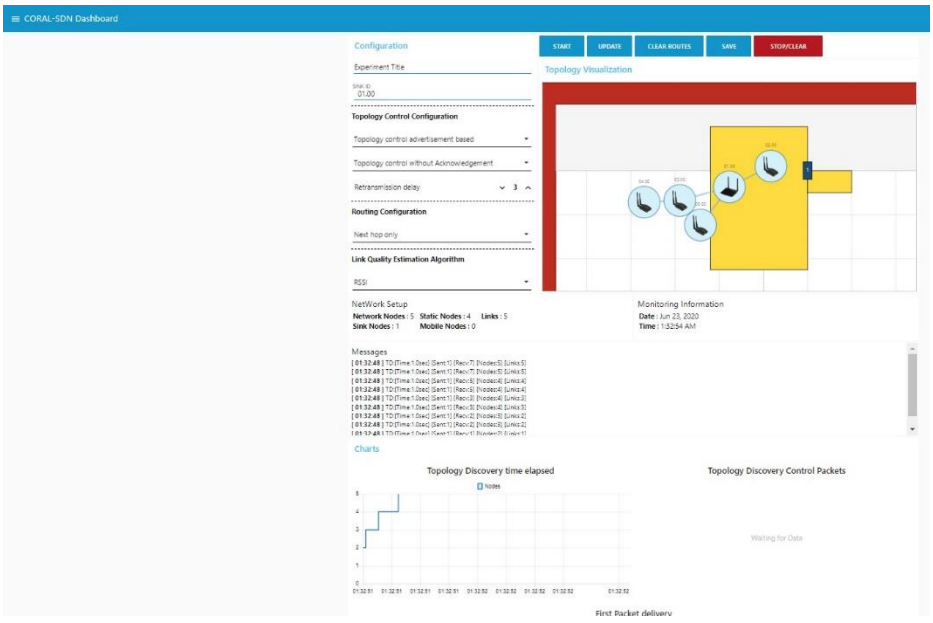
Εικόνα 21 Ελεγκτής CORAL

Dashboard είναι το σημείο διεπαφής με τον χρήστη. Το dashboard το οποίο είναι υλοποιημένο σε Node-Red συλλέγει και παρουσιάζει δεδομένα τα οποία χρειάζεται ο

χρήστης. Επίσης, δίνει την δυνατότητα στον χρήστη να προχωρήσει σε ρυθμίσεις και αλλαγές στη λειτουργία του πρωτοκόλλου.



Εικόνα 22 Η υλοποίηση του dashboard σε NodeRed



Εικόνα 23 Το Dashboard

Προκειμένου να εκτελεστεί η προσομοίωση ο χρήστης μέσω των εκτελέσιμων script που δημιουργήθηκαν στα πλαίσια της παρούσας εργασίας, και υπάρχουν διαθέσιμα στο Παράρτημα Α, μπορεί να εγκαταστήσει και να εκτελέσει τα απαραίτητα αρχεία για να ξεκινήσει η προσομοίωση. Για να το κάνει αυτό ο χρήστης θα πρέπει να χρησιμοποιήσει κάποιο virtual machine με χρήση λειτουργικού συστήματος Ubuntu 16.04 . Για να εγκατασταθούν τα απαραίτητα αρχεία, ο χρήστης πρέπει να εκτελέσει το script: **install_CORAL_SDN.sh**

και για να εκτελέσει την προσομοίωση: **run_CORAL.sh**

Σημείωση: τα συγκεκριμένα script δεν εγκαθιστούν και δεν εκτελούν το Node-red το οποίο θα πρέπει ο χρήστης να εγκαταστήσει ξεχωριστά στο Host σύστημα των windows.

3.4 Παρατηρήσεις από την χρήση των δύο υλοποιήσεων

Η παρατήρηση των δύο υλοποιήσεων κατά την εκτέλεση των προσομοιώσεων μας έδωσε την δυνατότητα να ξεχωρίσουμε αμέσως τα διαφορετικά χαρακτηριστικά τα οποία έχουν. Έτσι, εντοπίσαμε ότι το SDN-WISE δεν υποστηρίζει κινητούς κόμβους, καθώς η διαδικασία ανακάλυψης γειτονικών κόμβων δημιουργεί μεγάλη διακίνηση πακέτων. Αποτέλεσμα της διακίνησης αυτής ήταν να χάνεται η σύνδεση μεταξύ ελεγκτή και Sink κόμβου. Επίσης, παρατηρήσαμε στο CORAL αυτό δεν συμβαίνει. Ένα άλλο χαρακτηριστικό του SDN-WISE είναι ότι στις διαφορετικές υλοποιήσεις τις οποίες έχει δεν δίνει ουσιαστικά στον χρήση αρκετό έλεγχο, καθώς είτε δεν παρέχει κάποια εικόνα από την τοπολογία του δικτύου ή δεν παρέχει δυνατότητες για ρυθμίσεις στον χρήστη. Επίσης, από τη χρήση του SDN-WISE είδαμε ότι ο ONOS παρέχει αρκετές επιλογές για ρυθμίσεις, αλλά απαιτεί από τον χρήστη να χρησιμοποιήσει ένα CLI και όχι το GUI, πράγμα που απαιτεί από τον χρήστη να πρέπει να προχορήσει σε αναλυτική μελέτη του εγχειριδίου του ONOS. Αντίθετα το CORAL χρησιμοποιώντας το dashboard σαν κεντρικό στοιχείο επιτρέπει στο χρήση να ελέγχει, να ρυθμίζει και να παρατηρεί τις αλλαγές τις οποίες πραγματοποίησε. Επίσης, παρατηρήσαμε ότι ο ONOS δίνει την δυνατότητα στον χρήστη να υλοποιήσει ακόμα και ένα cluster από ελεγκτές ONOS επιτρέποντας να υπάρχει υψηλή διαθεσιμότητα στους ελεγκτές. Ο ONOS προσφέρει με την σειρά του στο GUI δυνατότητες στον χρήστη να παρακολουθεί την τοπολογία καθώς και κάποια άλλα χαρακτηριστικά του.

Επίσης, καθώς και οι δύο υλοποιήσεις απαιτούν αρκετή γνώση στη χρήση του λειτουργικού Linux προκειμένου να εγκατασταθούν και να εκτελεστούν προχωρήσαμε στην υλοποίηση των script, τα οποία είναι διαθέσιμα στο Παράρτημα Α, τα οποία αυτοματοποιούν τις διαδικασίες αυτές. Τέλος, υπάρχουν πολλαπλά στοιχεία των δύο υλοποιήσεων τα οποία μπορούμε να συγκρίνουμε προκειμένου να εξάγουμε χρήσιμα συμπεράσματα.

4 Θεωρητική σύγκριση CORAL – SDN-WISE

Όπως έχουμε ήδη αναφέρει θέμα της παρούσας εργασίας είναι η σύγκριση μεταξύ του CORAL και του SDN-WISE. Όπως έχουμε παρατηρήσει και από την περιγραφή των δύο υλοποιήσεων και οι δύο εμφανίζουν αρκετές ομοιότητες κάτι που είναι λογικό, καθώς η υλοποίηση του CORAL ξεκίνησε σαν ιδέα από την υλοποίηση του SDN-WISE [4]. Αρχικά κάποιες από τις ομοιότητες τις οποίες έχουν οι δύο υλοποιήσεις είναι ότι και οι δύο λαμβάνουν υπόψη τους φυσικούς περιορισμούς που έχουν οι συσκευές του IoT. Επίσης, και οι δύο υποστηρίζουν την εκτέλεση των πειραμάτων σε φυσικούς ή προσομοιωμένους κόμβους μέσω του Cooja. Τέλος, και οι δύο υλοποιήσεις χρησιμοποιούν διαφορετικά επίπεδα και διαχωρίζουν τα πρωτόκολλα λειτουργίας τους σε διακριτά επίπεδα όπως συμβαίνει και στο OpenFlow.

Επομένως, έχουμε να συγκρίνουμε δυο υλοποιήσεις οι οποίες προσπαθούν να επιτύχουν εισαγωγή των SDN στο Διαδίκτυο των Αντικειμένων διαχωρίζοντας το Επίπεδο Ελέγχου από το Επίπεδο των Δεδομένων. Αναλύοντας περαιτέρω τις δύο υλοποιήσεις βλέπουμε δυο διαφορετικές προσεγγίσεις οι οποίες έχοντας έναν κοινό στόχο καταφέρνουν να διαφοροποιηθούν η μια από την άλλη, θέτοντας επιπλέον επιμέρους στόχους οι οποίοι είναι βασισμένοι στην όσο το δυνατόν καλύτερη προσαρμογή των SDN σε ένα δίκτυο με πάρα πολύ ιδιαίτερες απαιτήσεις.

Προχωρώντας σε μία ενδελεχή σύγκριση των δύο υλοποιήσεων θα σταθούμε σε κάποια βασικά χαρακτηριστικά των δύο υλοποιήσεων προκειμένου να καταφέρουμε να εξάγουμε ένα ασφαλές συμπέρασμα σχετικά με τους τομείς τους οποίους η κάθε υλοποίηση θεραπεύει καλύτερα. Οι βασικοί τομείς στους οποίους θα πραγματοποιηθεί η σύγκριση είναι:

4.1 Εφαρμογές που υποστηρίζει η κάθε υλοποίηση

Στην υλοποίηση του CORAL ένας από τους κύριους στόχους του είναι να υποστηρίξει βελτιωμένες εφαρμογές QoE (Quality of Experience) για τους χρήστες και QoS (Quality Of Service) για τις εφαρμογές που αναπτύσσονται στο Διαδίκτυο των Αντικειμένων [4]. Επίσης, παρατηρούμε ότι οι εφαρμογές τις οποίες οι δημιουργοί του CORAL θέλουν να μπορούν να υλοποιηθούν στην συγκεκριμένη πλατφόρμα είναι εφαρμογές με περίπλοκες απαιτήσεις χωρίς να είναι απαραίτητο να πραγματοποιηθεί κάποια τροποποίηση στον τρόπο με τον οποίο λειτουργεί το CORAL από την πλευρά του χρήστη. Χαρακτηριστικό

παράδειγμα τέτοιας εφαρμογής είναι η προτεραιότητα σε κάποιους κόμβους σε καταστάσεις εκτάκτου ανάγκης [4]. Επιπλέον, το CORAL δίνει την δυνατότητα στον χρήστη να προχωρήσει σε ρυθμίσεις ή αλλαγές μέσω του Dashboard.

Από την άλλη στο SDN-WISE παρατηρούμε ότι ο στόχος ο οποίος θέτει είναι να υποστηρίξει καινοτόμες εφαρμογές [9] χωρίς να φαίνεται ξεκάθαρα το πόσο εύκολο είναι να πραγματοποιηθεί κάτι τέτοιο. Επίσης, στην περίπτωση του SDN-WISE παρατηρούμε ότι με την χρήση του ONOS υπάρχει περιθώριο για την ανάπτυξη διαφορετικών εφαρμογών οι οποίες μπορούν να υλοποιηθούν χωρίς να χρειάζεται ο χρήστης να χρησιμοποιήσει κάποιον ειδικό σε κάθε περίπτωση ελεγκτή. Με τη χρήση του ONOS το SDN-WISE έχει την δυνατότητα να υποστηρίζει και ετερογενή δίκτυα δίνοντας δυνατότητες για καθολικό έλεγχο σε ετερογενή δίκτυα, παρέχοντας έτσι την δυνατότητα στον χρήστη να υλοποιήσει εφαρμογές όχι μόνο στο Ασύρματο Δίκτυο Αισθητήρων [17]. Άρα βλέπουμε ότι το CORAL είναι υλοποιημένο έτσι ώστε να δίνει την δυνατότητα να υλοποιηθούν εξειδικευμένες και απαιτητικές εφαρμογές όμως και το SDN-WISE με τη χρήση του ONOS παρέχει αυξημένες δυνατότητες για υλοποίηση εφαρμογών.

4.2 Αρχιτεκτονική

Η αρχιτεκτονική των δύο υλοποιήσεων διαφέρει αρκετά καθώς στην υλοποίηση του SDN-WISE αρκετά από τα Επίπεδα εκτελούνται απευθείας στους κόμβους. Κατά τον τρόπο αυτό διαπιστώνουμε ότι υπάρχει πολυπλοκότητα στους κόμβους και πιθανώς αυξημένη ζήτηση για επεξεργαστική ισχύ σε αυτούς, το οποίο συνεπάγεται αυξημένη κατανάλωση ενέργειας. Επίσης, με την χρήση του ONOS υπάρχει δυνατότητα για εκτέλεση επιπλέον μικρών προγραμμάτων στους κόμβους, πράγμα το οποίο θέτει επιπλέον ερωτηματικά σχετικά με την κατανάλωση πόρων στους κόμβους.

Παρατηρώντας από την άλλη την αρχιτεκτονική του CORAL παρατηρούμε ότι όλη η πολυπλοκότητα μεταφέρεται από το Επίπεδο Επικοινωνίας Δεδομένων στο Επίπεδο Ελέγχου, άρα όλη η πολυπλοκότητα μεταφέρεται στον ελεγκτή. Με αυτό τον τρόπο το CORAL καταφέρνει να υποστηρίζει την Ευέλικτη διαχείριση του δικτύου (Elasticity), την Ετερογένεια (Heterogenity) του δικτύου, καθώς και να διαχειρίζεται αποτελεσματικά τους Μετακινούμενους κόμβους (Mobility) .

4.3 Απαιτήσεις δικτύου IoT

Όπως είδαμε το δίκτυο του IoT είναι δίκτυο με ιδιαίτερα χαρακτηριστικά λειτουργίας και ιδιαίτερες απαιτήσεις. Κάποιες από τις απαιτήσεις αυτές είναι:

- Εξοικονόμηση ενέργειας
- Ετερογενείς συσκευές
- Ευέλικτη Διαχείριση του Δικτύου
- Λειτουργία ανεξάρτητη επιπέδων
- Διαχείριση των κινητών κόμβων

Και οι δύο πλατφόρμες λαμβάνουν υπόψη τους την Εξοικονόμηση ενέργειας. Από τη μια στο SDN-WISE μειώνουν τη δύναμη εκπομπής του πομπού, εφαρμόζουν την ενοποίηση των πακέτων που μεταφέρονται προς τον ίδιο προορισμό, και ενεργοποιούν ή απενεργοποιούν τον πομπό του κόμβου έτσι ώστε να εξασφαλίζουν την περαιτέρω εξοικονόμηση ενέργειας. Από την άλλη το CORAL χρησιμοποιεί την ευφυΐα του πρωτοκόλλου προκειμένου να πετύχει καλύτερη ενεργειακή διαχείριση.

Στην ετερογένεια το CORAL αξιοποιεί διάφορα Modules που του επιτρέπουν να διαχειρίζεται την ετερογένεια του δικτύου. Τα Modules αυτά είναι ο Ενοποιημένος Έλεγχος Δικτύου και Μετάδοσης (Unified Network and Radio Control), τα οποία λαμβάνουν δεδομένα από το API του CORAL. Επίσης, το SDN-WISE με τη χρήση του ONOS παρέχει αυξημένες δυνατότητες για διαχείριση της ετερογένειας.

Για την Ευέλικτη διαχείριση του δικτύου το CORAL υποστηρίζει δυναμική διαμόρφωση των συσκευών IoT, καθώς και των παραμέτρων του πρωτοκόλλου προκειμένου να βελτιώσει την απόδοση και κατανομή των πόρων. Με αυτόν τον τρόπο πετυχαίνει να διαχειρίζεται και τους Κινητούς Κόμβους (Mobility), ενώ η υλοποίηση του SDN-WISE δείχνει ότι σε σχέση με του Κινητούς Κόμβους δεν μπορεί να πετύχει αποτελεσματική διαχείριση, καθώς με τον τρόπο που λειτουργεί ο αλγόριθμος εύρεσης γειτονικών κόμβων στην περίπτωση των κινητών κόμβων δημιουργεί πολύ μεγάλη διακίνηση πακέτων δημιουργώντας μεγάλο overhead στο δίκτυο πράγμα που υποβαθμίζει την συνολική απόδοση του δικτύου. Επίσης, κατά τη διάρκεια εκτέλεσης προσομοιώσεων με κινητούς κόμβους στο SDN-WISE παρατηρήσαμε ότι η τοπολογία του δικτύου δεν ανανεωνόταν, καθώς επίσης και ο ελεγκτής έχανε την σύνδεση του με τον Sink κόμβο μετά από κάποιο χρόνο, πράγμα το οποίο δείχνει ότι το SDN-WISE υστερεί σημαντικά στον τομέα αυτό.

4.4 Πρωτόκολλο λειτουργίας

Το CORAL υποστηρίζει 2 διαφορετικά πρωτόκολλα το CORAL-SDN, καθώς και το Adaptable-RPL. Στο πρωτόκολλο του CORAL υπάρχουν Intelligent Modules, καθώς και

δυνατότητα για επιπλέον εισαγωγή νέων αλγορίθμων ή νέων module, πράγμα το οποίο μας δείχνει ότι πρόκειται για ένα αρθρωτό και επεκτάσιμο πρωτόκολλο. Επίσης, το CORAL δίνει την δυνατότητα στον χρήστη να επιλέξει το πρωτόκολλο που θα χρησιμοποιήσει. Επίσης, και στα δυο πρωτόκολλα που χρησιμοποιεί το CORAL πραγματοποιείται δυναμικά παραμετροποίηση του πρωτοκόλλου από τον Ελεγκτή με αποτέλεσμα να μπορεί να λειτουργήσει σωστά το πρωτόκολλο ανάλογα με τις συνθήκες που υπάρχουν στο δίκτυο εκείνη τη στιγμή. Τέλος, ανάλογα με την περίπτωση προσαρμόζεται και η εγκαθίδρυση των ροών, καθώς εάν έχουμε μια σταθερή τοπολογία με έναν κινητό κόμβο είναι δαπανηρή η διαμόρφωση ολόκληρης της διαδρομής κάθε φορά που ένας μόνο κόμβος αλλάζει το σημείο προσάρτησης του. Με τις δυναμικές αυτές αλλαγές το CORAL υποστηρίζει κινητούς κόμβους, την ετερογένεια των διαφορετικών κόμβων. Σε αυτές όλες τις δυνατότητες το SDN-WISE υστερεί σημαντικά καθώς οι παράμετροι του πρωτοκόλλου δεν προσαρμόζονται δυναμικά ανάλογα με τις ανάγκες εκείνης της στιγμής.

4.5 Ελεγκτής

Το SDN-WISE χρησιμοποιεί έναν ελεγκτή τον WISE-Visor ο οποίος είναι ένας ελεγκτής υλοποιημένος σε Java και οποίος είναι ο βασικός ελεγκτής του δικτύου καθώς εκεί λειτουργεί ο Έλεγχος της Τοπολογίας. Όμως μαζί με αυτόν τον ελεγκτή ο χρήστης μπορεί να χρησιμοποιήσει οποιοδήποτε άλλο ελεγκτή επιθυμεί μέσω του API που προσφέρει. Επίσης, το SDN-WISE εισάγει και την χρήση του ελεγκτή ONOS ο οποίος επιτρέπει στον χρήστη να πραγματοποιήσει αλλαγές στον τρόπο με τον οποίο λειτουργεί και επιπλέον έχει την δυνατότητα να διαχειρίζεται ετερογενή δίκτυα πράγμα το οποίο αυξάνει κατακόρυφα τις δυνατότητες του SDN-WISE.

Στην περίπτωση του CORAL ο ελεγκτής που χρησιμοποιείται είναι ο CORAL-SDN και είναι υλοποιημένος σε Java. Επίσης, το CORAL δεν έχει δυνατότητα για χρήση πολλαπλών ελεγκτών, αλλά μέσω API μπορεί να τροφοδοτήσει δεδομένα σε κάποιο άλλο σύστημα όπως πχ Artificial Intelligence. Επίσης, έχει την δυνατότητα υποστηρίξει νέους αλγόριθμους και νέες ευφυείς δυνατότητες.

4.6 Χρήση Εφαρμογής

Μετά τη χρήση του CORAL και του SDN-WISE στις υλοποιήσεις που προσφέρει το κάθε ένα παρατηρήσαμε αμέσως τη χρησιμότητα του πάνελ που παρέχει το CORAL μέσω του οποίου ήταν δυνατόν να πραγματοποιηθούν ρυθμίσεις στο πρωτόκολλο, καθώς επίσης και

να παρατηρήσουμε τυχόν αλλαγές στην τοπολογία. Από την άλλη το SDN-WISE με την χρήση του ONOS προσφέρει δυνατότητες στον χρήστη για αλλαγές στον τρόπο με τον οποίο λειτουργεί ο ίδιος ο ελεγκτής, όχι όμως μέσω κάποιου GUI. Επίσης, παρέχει εικόνα στον χρήστη για την τοπολογία του δικτύου, καθώς και για τους κανόνες που υπάρχουν και χρησιμοποιούνται εκείνη τη στιγμή.

Μια άλλη παράμετρος την οποία διαπιστώσαμε κατά τη χρήση του SDN-WISE είναι ότι επιτρέπει στον χρήστη να χρησιμοποιήσει μια συστοιχία από ελεγκτές ONOS πράγμα το οποίο προσφέρει στην συγκεκριμένη υλοποίηση επεκτασιμότητα και υψηλή διαθεσιμότητα καθώς αν ένας ελεγκτής τεθεί εκτός λειτουργίας οι υπόλοιποι συνεχίζουν να λειτουργούν κανονικά. Στις υπόλοιπες υλοποιήσεις του SDN-WISE παρατηρήσαμε ότι δεν υπάρχει κάποιο πάνελ από το οποίο ο χρήστης να ελέγχει την τοπολογία ή παρόλο που υπήρχε η δυνατότητα για παρατήρηση της τοπολογίας ή για ρυθμίσεις στο πρωτόκολλο οι δυνατότητες που έδινε η προσομοίωση ήταν πολύ περιορισμένες, με αποτέλεσμα να είναι πρακτικά αδύνατο να υλοποιηθεί κάποιο διαφορετικό σενάριο εκτέλεσης.

Με τον τρόπο αυτό παρατηρούμε ότι το CORAL στηρίζεται στο dashboard, το οποίο παρέχει αυξημένες δυνατότητες στον χρήστη και από την άλλη παρατηρούμε ότι το SDN-WISE στηρίζεται κυρίως στην υλοποίηση με τον ONOS που προσφέρει κάποιες δυνατότητες στον χρήστη.

Στο κομμάτι της προσομοίωσης παρατηρήσαμε ότι και οι δύο υλοποιήσεις χρησιμοποιούν το Cooja προκειμένου να εκτελεστούν οι προσομοιώσεις τους, αλλά οι ελεγκτές είναι εξωτερικοί πράγμα που σημαίνει ότι ο χρήστης έχει την δυνατότητα να εγκαταστήσει τον ελεγκτή σε οποιαδήποτε πλατφόρμα επιθυμεί υπό την προϋπόθεση ότι θα υπάρχει επικοινωνία μεταξύ του ελεγκτή και της προσομοίωσης. Από τα παραπάνω συμπεραίνουμε ότι πρόκειται για υλοποιήσεις οι οποίες έχουν την δυνατότητα να εγκατασταθούν σε διαφορετικά περιβάλλοντα και ο ελεγκτής τους να είναι εντελώς αυτόνομος.

Από την πρακτική χρήση του SDN-WISE παρατηρήσαμε ότι χρησιμοποιεί κόμβους υλοποιημένους σε Java, πράγμα το οποίο δεν επιτρέπει στον χρήστη να υλοποιήσει εύκολα σενάρια εκτέλεσης π.χ. να ρυθμίσει τους κόμβους να στέλνουν συγκεκριμένα μηνύματα ή να προχωρήσει σε παραμετροποίηση των κόμβων που χρησιμοποιεί. Αντίθετα, το CORAL χρησιμοποιεί τα Cooja Motes, καθώς και Z1 Motes και δίνει στον χρήστη μεγαλύτερη ελευθερία στα σενάρια τα οποία θέλει να εκτελέσει.

Κατά τη διάρκεια της πρακτικής χρήσης του SDN-WISE παρατηρήσαμε ότι ο ελεγκτής ONOS δεν ανανέωνε την τοπολογία του δικτύου ειδικά σε περιπτώσεις στις οποίες χρησιμοποιήσαμε την επέκταση του Cooja Mobility που επιτρέπει στους κόμβους να μετακινούνται κατά τη διάρκεια της προσομοίωσης. Αυτή η παρατήρηση επιβεβαιώνει ότι το SDN-WISE δεν υποστηρίζει σωστά τους κινητούς κόμβους. Επίσης, παρατηρήσαμε ότι συχνά ο ελεγκτής ONOS έχανε τη σύνδεση με την προσομοίωση μετά από κάποιο χρόνο πράγμα το οποίο δημιουργεί προβλήματα με την εκτέλεση σεναρίων ειδικά περίπλοκων. Αντίθετα, το CORAL δεν αντιμετωπίζει τέτοια ζητήματα ούτε με τους κινητούς κόμβους, αλλά ούτε και με τον ελεγκτή ο οποίος δεν έχασε την σύνδεση του ακόμα και σε προσομοιώσεις οι οποίες εκτελούνταν για αρκετές ώρες.

4.7 Αποτελέσματα που προκύπτουν από την σύγκριση

Με βάση τη σύγκριση που προηγήθηκε παρατηρούμε ότι η κάθε υλοποίηση έχει κάποια στοιχεία τα οποία την κάνουν να ξεχωρίζει από την άλλη. Εάν παρατηρήσουμε τη γενική εικόνα των δύο υλοποιήσεων διαπιστώνουμε ότι και οι δύο είναι υλοποιημένες έτσι ώστε να μπορούν να λειτουργήσουν για την εκτέλεση πειραμάτων είτε σε φυσικούς κόμβους είτε σε προσομοιωμένους. Ο τρόπος με τον οποίο λειτουργούν είναι αρκετά όμοιος, αν όμως παρατηρήσουμε τα επιμέρους στοιχεία τα οποία αποτελούν την κάθε υλοποίηση μπορούμε να διαπιστώσουμε άμεσα ότι η υλοποίηση του SDN-WISE δεν είναι φιλική προς τον χρήστη και δεν παρέχει την απαιτούμενη ευελιξία που απαιτεί το Διαδίκτυο των Αντικειμένων. Τέλος, δεν παρέχει εναλλακτικές επιλογές στον χρήστη (διαφορετικό πρωτόκολλο ή ευχέρεια ρυθμίσεων) , όμως παρέχει δυνατότητες όπως πολλαπλούς ελεγκτές. Αντίθετα, παρατηρούμε ότι το CORAL κάνει χρήση διάφορων τεχνολογιών και εισάγει την ευφυΐα στο SDN προκειμένου να καταφέρει να πετύχει καλύτερα αποτελέσματα στους στόχους που έχει θέσει.

Αντικείμενα σύγκρισης	**CORAL**	**SDN-WISE**
Υποστηρίζει πολλαπλά πρωτόκολλα	ΝΑΙ	ΟΧΙ
Εφαρμογές προσανατολισμένες σε QoS QoE	ΝΑΙ	ΟΧΙ
Dashboard	ΝΑΙ	ΟΧΙ
Ευφυής ρύθμιση πρωτοκόλλου	ΝΑΙ	ΟΧΙ
Ο χρήστης επιλέγει το πρωτόκολλο	ΝΑΙ	ΟΧΙ
Υποστηρίζει ετερογενή δίκτυα	ΝΑΙ	ΝΑΙ

Ο ελεγκτής είναι επεκτάσιμος	ΝΑΙ	ΝΑΙ
Το πρωτόκολλο είναι επεκτάσιμο	ΝΑΙ	ΟΧΙ
Υποστηρίζει κινητούς κόμβους	ΝΑΙ	ΟΧΙ
Μπορεί να χρησιμοποιεί πολλαπλούς ελεγκτές	ΟΧΙ	ΝΑΙ
Χρησιμοποιεί ευφυΐα	ΝΑΙ	ΟΧΙ
Δίνει την δυνατότητα στον χρήστη να πραγματοποιήσει αλλαγές	ΝΑΙ	ΟΧΙ
Υποστηρίζει εξωτερικούς ελεγκτές	ΟΧΙ	ΝΑΙ

Πίνακας 3 Πίνακας Σύγκρισης CORAL και SDN-WISE

5 Επίλογος

Όπως παρατηρήσαμε στην παρούσα εργασία η χρήση του SDN στο Ασύρματο Δίκτυο Αισθητήρων, καθώς και στο Διαδίκτυο των Αντικειμένων παίζει σημαίνοντα ρόλο καθώς παρέχει όλα τα απαραίτητα εργαλεία προκειμένου να επιτρέψει την περαιτέρω εξέλιξη του Διαδικτύου των Αντικειμένων. Επίσης, δίνει την δυνατότητα για παροχή πιο εξειδικευμένων υπηρεσιών ή εφαρμογών. Τέλος, παρατηρήσαμε τον τρόπο με τον οποίο διαφορετικές υλοποιήσεις στην εισαγωγή του SDN στο Διαδίκτυο των Αντικειμένων μπορούν να χρησιμοποιηθούν προκειμένου να προσφέρουν διαφορετικές υπηρεσίες και παροχές προς τους χρήστες, αλλά και να απλοποιήσουν τον τρόπο με τον οποίο πραγματοποιείται, ρυθμίζεται και λειτουργεί το Διαδίκτυο των Αντικειμένων μετά την εισαγωγή του SDN.

5.1 Σύνοψη και συμπεράσματα

Από το θεωρητικό υπόβαθρο της εργασίας συμπεραίνουμε ότι τα ευφυή προγραμματιζόμενα δίκτυα διαχωρίζοντας το επίπεδο του ελέγχου από αυτό των δεδομένων, παρέχουν μία νέα δυναμική αρχιτεκτονική μέσω της οποίας δημιουργούνται δυνατότητες για την εισαγωγή νέων εφαρμογών στο Διαδίκτυο των Αντικειμένων. Επίσης, παρατηρήσαμε ότι οι διαφορετικές υλοποιήσεις των SDN στο Διαδίκτυο των Αντικειμένων δεν καλούνται να επιλύσουν μόνο ζητήματα τα οποία υπάρχουν στο Διαδίκτυο των Αντικειμένων όπως ετερογένεια κόμβων, εξοικονόμηση ενέργειας, λειτουργία κάτω από δύσκολες συνθήκες αλλά και ζητήματα τα οποία εμφανίζονται με την εισαγωγή των SDN όπως overhead στο δίκτυο, επικοινωνία μεταξύ ελεγκτή-κόμβων. Με βάση το θεωρητικό υπόβαθρο, αλλά και με βάση τα αποτελέσματα της σύγκρισης παρατηρούμε ότι η κάθε υλοποίηση έχει κάποια στοιχεία τα οποία την κάνουν να ξεχωρίζει από την άλλη, αλλά και πολλά κοινά στοιχεία.

Επίσης, από τη θεωρητική σύγκριση μεταξύ CORAL και SDN-WISE παρατηρούμε ότι οι δύο υλοποιήσεις εμφανίζουν αρκετά κοινά χαρακτηριστικά, όπως ότι και οι δυο είναι υλοποιημένες έτσι ώστε να μπορούν να λειτουργήσουν για την εκτέλεση πειραμάτων είτε σε φυσικούς είτε σε προσομοιωμένους κόμβους, αλλά και ότι δημιουργήθηκαν έτσι ώστε να είναι επεκτάσιμες και να υποστηρίζουν καινοτόμες εφαρμογές. Όμως, εμφανίζουν και αρκετές διαφορές έτσι μπορούμε να δούμε άμεσα ότι η υλοποίηση του SDN-WISE δεν είναι αρκετά φιλική προς τον χρήστη, καθώς δεν παρέχει αρκετές δυνατότητες για ρύθμιση

του πρωτοκόλλου το οποίο χρησιμοποιεί. Επίσης, δεν εμφανίζει και χαρακτηριστικά ευελιξίας που απαιτεί το Διαδίκτυο των Αντικειμένων, όπως υποστήριξη κινητών κόμβων, όμως παρέχει διαφορετικές δυνατότητες με τη χρήση πολλαπλών ελεγκτών ή εξωτερικών ελεγκτών όπως ο ONOS. Αντίθετα, διαπιστώνουμε ότι το CORAL κάνει χρήση διάφορων τεχνολογιών και εισάγει την ευφυΐα στο SDN προκειμένου να καταφέρει να παρέχει δυνατότητες οι οποίες είναι πάρα πολύ σημαντικές στο Διαδίκτυο των Αντικειμένων, όπως υποστήριξη κινητών κόμβων, υποστήριξη ετερογενών κόμβων. Επίσης, παρέχει στον χρήστη αυξημένες δυνατότητες για έλεγχο με την χρήση του Dashboard.

Από την πρακτική ενασχόληση με τις δύο υλοποιήσεις παρατηρούμε ότι το CORAL είναι μια πλατφόρμα η οποία επιτρέπει στον χρήστη να διαμορφώσει επακριβώς τα σενάρια εκτέλεσης που επιθυμεί και του δίνει περισσότερες επιλογές λόγω των κόμβων που χρησιμοποιεί εν αντιθέσει με το SDN-WISE που δεν προσφέρει τέτοιες δυνατότητες λόγω των custom υλοποιημένων σε Java κόμβων. Επίσης, μετά από διάφορες εκτελέσεις παρατηρήσαμε ότι το SDN-WISE δεν παρέχει την απαραίτητη σταθερότητα σε προσομοιώσεις οι οποίες εκτελούνταν για αρκετό χρόνο, καθώς ο ελεγκτής έχανε την σύνδεση του με τον Sink κόμβο. Επιπλέον, κατά την εκτέλεση προσομοιώσεων με κινητούς κόμβους επιβεβαιώσαμε ότι η τοπολογία δεν ανανεωνόταν σωστά πράγμα το οποίο δείχνει ότι το SDN-WISE δεν υποστηρίζει σωστά κινητούς κόμβους εντός της προσομοίωσης. Αντίθετα, στο CORAL δεν αντιμετωπίσαμε τέτοια προβλήματα κατά την προσομοίωση. Τέλος, και οι δύο υλοποιήσεις χρησιμοποιούν ελεγκτές οι οποίοι λειτουργούν ανεξάρτητα από την προσομοίωση πράγμα που δίνει δυνατότητα στον χρήστη να στήσει τον ελεγκτή σε οποιαδήποτε πλατφόρμα επιθυμεί.

5.2 Όρια και περιορισμοί της έρευνας

Η συγκεκριμένη εργασία περιορίστηκε σε μια βιβλιογραφική επισκόπηση των δύο διαφορετικών υλοποιήσεων, τη σύγκριση μεταξύ των δύο υλοποιήσεων, καθώς και την χρήση των δύο υλοποιήσεων. Δεν πραγματοποιήθηκε εκτέλεση των προσομοιώσεων των διαφορετικών υλοποιήσεων και των διαφορετικών ελεγκτών, καθώς δεν υπήρχε πλήρης δυνατότητα για πραγματική υλοποίηση με χρήση κινητών κόμβων, καθώς κάποιες από τις υλοποιήσεις δεν υποστηρίζονται πλήρως από τους κόμβους οι οποίοι υπάρχουν διαθέσιμοι αυτή τη στιγμή. Επίσης, κάποια από τα αρχικά σενάρια που επιλέχθηκαν προκειμένου να αναδειχτούν οι δυνατότητες και των δύο υλοποιήσεων δεν ήταν δυνατόν να υλοποιηθούν στο SDN-WISE λόγω περιορισμού στις δυνατότητες του λογισμικού και των

προσομοιωμένων κόμβων. Τέλος, κατά τη διάρκεια συγγραφής της παρούσας εργασίας δημοσιεύθηκε η νέα έκδοση του CORAL SDN η οποία ονομάζεται VERO-SDN [5] και λόγω των χρονικών περιορισμών της παρούσας διατριβής περιοριστήκαμε στην ανάλυση μόνο του CORAL.

5.3 Μελλοντικές Επεκτάσεις

Στο μέλλον θα μπορούσε να γίνει περεταίρω σύγκριση των δύο υλοποιήσεων με τη χρήση πραγματικών κόμβων ή εξομοιωμένων κόμβων πάνω σε κοινά σενάρια προκειμένου να συλλεχθούν δεδομένα και να πραγματοποιηθεί περαιτέρω ανάλυση του τρόπου με τον οποίο λειτουργούν οι δύο υλοποιήσεις, αλλά και των αποτελεσμάτων που πετυχαίνουν οι υλοποιήσεις αυτές σε τομείς όπως η εξοικονόμηση ενέργειας. Επίσης, θα μπορούσαν να γίνουν περεταίρω πειραματικές υλοποιήσεις με χρήση πραγματικών κόμβων και διαφορετικών πρωτοκόλλων προκειμένου να δούμε εάν τα ευφυή προγραμματιζόμενα δίκτυα μας επιτρέπουν να πραγματοποιήσουμε διασύνδεση ακόμα και διαφορετικών πρωτοκόλλων με την χρήση του SDN. Επίσης, μια εναλλακτική πρόταση θα ήταν να πραγματοποιηθεί εισαγωγή στο CORAL διαφορετικών ελεγκτών όπως ο ONOS, προκειμένου να πραγματοποιηθεί μία σύγκριση των δύο ελεγκτών (CORAL και ONOS) με βάση την ίδια υλοποίηση. Τέλος, μια εναλλακτική πρόταση για την υλοποίηση του SDN-WISE είναι η επέκταση του έτσι ώστε να υποστηρίζει πλήρως τους κινητούς κόμβους.

Βιβλιογραφία

[1] Y. Li, X. Su, J. Riekki, T. Kanter, and R. Rahmani, "A SDN-based architecture for horizontal Internet of Things services," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–7, doi: 10.1109/ICC.2016.7511053.

[2] A.-C. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SD-WISE: A Software-Defined WIreless SEnsor network," *Computer Networks*, vol. 159, pp. 84–95, Aug. 2019, doi: 10.1016/j.comnet.2019.04.029.

[3] H. A. Tran, D. Tran, L. G. Nguyen, A. Mellouk, H. Mac, and V. Tong, "A Novel SDN Controller Based on Ontology and Global Optimization for Heterogeneous IoT Architecture," in *Proceedings of the Eighth International Symposium on Information and Communication Technology*, New York, NY, USA, 2017, pp. 293–300, doi: 10.1145/3155133.3155143.

[4] T. Tryfon, V. George, V. Polychronis, P. Sophia, and M. Lefteris, "Cross-Layer Control of Data Flows for the Internet of Things," Feb. 2018.

[5] T. Theodorou and L. Mamatas, "A Versatile Out-of-Band Software-Defined Networking Solution for the Internet of Things," *IEEE Access*, pp. 1–1, 2020, doi: 10.1109/ACCESS.2020.2999087.

[6] M. Baddeley, R. Nejabati, G. Oikonomou, M. Sooriyabandara, and D. Simeonidou, "Evolving SDN for Low-Power IoT Networks," *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pp. 71–79, Jun. 2018, doi: 10.1109/NETSOFT.2018.8460125.

[7] E. Municio, J. Marquez-Barja, S. Latré, and S. Vissicchio, "Whisper: Programmable and Flexible Control on Industrial IoT Networks," *Sensors*, vol. 18, no. 11, Art. no. 11, Nov. 2018, doi: 10.3390/s18114048.

[8] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software Defined Wireless Networks: Unbridling SDNs," in *2012 European Workshop on Software Defined Networking*, Oct. 2012, pp. 1–6, doi: 10.1109/EWSDN.2012.12.

[9] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, Apr. 2015, pp. 513–521, doi: 10.1109/INFOCOM.2015.7218418.

[10] Z. Zhang, Z. Zhang, R. Wang, Z. Jia, H. Lei, and X. Cai, "ESD-WSN: An Efficient SDN-Based Wireless Sensor Network Architecture for IoT Applications," in *Algorithms and Architectures for Parallel Processing*, 2017, pp. 735–745.

[11] H. Huang, J. Zhu, and L. Zhang, "An SDN_based management framework for IoT devices," in *25th IET Irish Signals Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*, Jun. 2014, pp. 175–179, doi: 10.1049/cp.2014.0680.

[12] T. M. C. Nguyen, D. B. Hoang, and Z. Chaczko, "Can SDN Technology Be Transported to Software-Defined WSN/IoT?," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Dec. 2016, pp. 234–239, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.63.

[13] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A Software Defined Networking architecture for the Internet-of-Things," in *2014 IEEE*

Network Operations and Management Symposium (NOMS), May 2014, pp. 1–9, doi: 10.1109/NOMS.2014.6838365.

[14] S. Chakrabarty and D. W. Engels, "A secure IoT architecture for Smart Cities," in *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan. 2016, pp. 812–813, doi: 10.1109/CCNC.2016.7444889.

[15] T. Theodorou and L. Mamatas, "CORAL-SDN: A software-defined networking solution for the Internet of Things," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov. 2017, pp. 1–2, doi: 10.1109/NFV-SDN.2017.8169870.

[16] A. G. Anadiotis, G. Morabito, and S. Palazzo, "An SDN-Assisted Framework for Optimal Deployment of MapReduce Functions in WSNs," *IEEE Transactions on Mobile Computing*, vol. 15, no. 9, pp. 2165–2178, Sep. 2016, doi: 10.1109/TMC.2015.2496582.

[17] A. G. Anadiotis, S. Milardo, G. Morabito, and S. Palazzo, "Toward Unified Control of Networks of Switches and Sensors Through a Network Operating System," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 895–904, Apr. 2018, doi: 10.1109/JIOT.2018.2805191.

[18] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "Reprogramming Wireless Sensor Networks by using SDN-WISE: A hands-on demo," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2015, pp. 19–20, doi: 10.1109/INFCOMW.2015.7179322.

[19] A. G. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "Towards a software-defined Network Operating System for the IoT," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Dec. 2015, pp. 579–584, doi: 10.1109/WF-IoT.2015.7389118.

[20] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *Proceedings 22nd International Conference on Distributed Computing Systems*, Jul. 2002, pp. 457–458, doi: 10.1109/ICDCS.2002.1022289.

[21] M. M. Mazhar, M. A. Jamil, A. Mazhar, A. Ellahi, M. S. Jamil, and T. Mahmood, "Conceptualization of Software Defined Network layers over internet of things for future smart cities applications," in *2015 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, Dec. 2015, pp. 1–4, doi: 10.1109/WiSEE.2015.7393104.

[22] T. Theodorou and L. Mamatas, "Software defined topology control strategies for the Internet of Things," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov. 2017, pp. 236–241, doi: 10.1109/NFV-SDN.2017.8169884.

[23] G. Violettas, T. Theodorou, S. Petridou, A. Tsioukas, and L. Mamatas, "Demo abstract: An experimentation facility enabling flexible network control for the Internet of Things," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, May 2017, pp. 992–993, doi: 10.1109/INFCOMW.2017.8116526.

Παράρτημα Α - Κώδικας

Τα παρακάτω script δημιουργήθηκαν στα πλαίσια της παρούσας διατριβής και χρησιμοποιούνται προκειμένου να εγκατασταθεί ή να εκτελεστεί το απαραίτητο λογισμικό έτσι ώστε να είναι πιο εύκολο για τον χρήστη να χρησιμοποιήσει τις πλατφόρμες του CORAL και του SDN-WISE. Επίσης, ο κώδικας είναι διαθέσιμος στο GitHub μέσω του παρακάτω συνδέσμου.

```
#!/bin/bash
## install_CORAL_SDN.sh ##############################################################################
#
###################################### Description ##################################################
#
# This script is used to install the CORAL-SDN along with the needed libraries in a clean
#   installation of Ubuntu using a VM.
#
###################################### Arguments ####################################################
#
# No argument is required
#
#########################################################################################################

# Update to Java 8 and install Ant and Maven
sudo apt-get update
sudo apt-get install default-jdk ant maven git python-pip curl gcc-msp430 build-essential vim \
   software-properties-common -y

# Update the .profile with the Java Paths
latestJdk=$(ls -lrt /usr/lib/jvm | grep "java-8-openjdk" | tail -1 | awk -F' ' '{print $9}')
echo "JAVA_HOME=/usr/lib/jvm/$latestJdk" >> ~/.profile
echo "JRE_HOME=/usr/lib/jvm/$latestJdk" >> ~/.profile
echo "PATH=$PATH:$JRE_HOME/bin:$JAVA_HOME/binexport" >> ~/.profile
echo "export JAVA_HOME" >> ~/.profile
echo "export JRE_HOME" >> ~/.profile
echo "export PATH" >> ~/.profile

# source the updated profile
source ~/.profile

# Checkout contiki
cd ~
git clone https://github.com/SWNRG/coral-sdn.git

# Copy Contiki to home
cp -r ~/coral-sdn/infrastructure-plane/contiki ~
```

```
# Install serial2pty in cooja
cd ~/contiki/tools/cooja/apps
rm -rf serial2pty
git clone https://github.com/cmorty/cooja-serial2pty.git serial2pty
cd serial2pty
ant jar
echo "org.contikios.cooja.Cooja.PLUGINS = + de.fau.cooja.plugins.Serial2Pty" > cooja.config
echo "org.contikios.cooja.Cooja.JARFILES = + serial2pty.jar" >> cooja.config
echo "DESCRIPTION = serial2pty" >> cooja.config

# Compile contiki
cd ~/contiki/tools/cooja
ant clean
ant jar

# Copy the coral adapter to home
cd ~
cp -r ~/coral-sdn/control-plane/CORAL-SDN_Adapters/coral-sdn-adapter-COOJA-runtime ~
cd ~/coral-sdn-adapter-COOJA-runtime
chmod +x coral-sdn-adapter createSoftLinkCreationScript.py softLinkCreationScript

# Copy the controller to home
cp -r ~/coral-sdn/control-plane/CORAL-SDN-Controller/ ~
```

```
#!/bin/bash
## install_sdn_wise_java.sh ###############################################################################
#
######################################## Description ###################################################
#
# This script is used to install the SDN-WISE along with the needed libraries and the Java SDN
#    controller in a clean installation of Ubuntu.
#
######################################## Arguments #####################################################
#
# No argument is required
#
################################################################################################################

# Update to Java 8 and install Ant and Maven
sudo apt-get update
sudo apt-get install openjdk-8-jdk maven librxtx-java git -y

# Setup environment variables
latestJdk=$(ls -lrt /usr/lib/jvm | grep "java-8-openjdk" | tail -1 | awk -F' ' '{print $9}')
echo "JAVA_HOME=/usr/lib/jvm/$latestJdk" >> ~/.bashrc
source ~/.bashrc
```

```
cd ~
git clone https://github.com/sdnwiselab/sdn-wise-java.git
cd ~/sdn-wise-java
mvn clean install
cd ctrl/build

# Finish message
echo "The installations finished, source again the bashrc or exit and login again."
```

```
#!/bin/bash
## install_SDN_WISE_simple_controler.sh ########################################################
#
######################################## Description ###########################################
#
# This script is used to install the SDN-WISE along with the needed libraries and the Java SDN
#    controller in a clean installation of Ubuntu.
#
######################################## Arguments #############################################
#
# No argument is required
#
################################################################################################

# Update to Java 8 and install Ant and Maven
sudo apt-get update
sudo apt-get install openjdk-8-jdk ant maven git -y

# Setup environment variables
latestJdk=$(ls -lrt /usr/lib/jvm | grep "java-8-openjdk" | tail -1 | awk -F' ' '{print $9}')
echo "JAVA_HOME=/usr/lib/jvm/$latestJdk" >> ~/.bashrc
source ~/.bashrc

# Checkout contiki
cd ~
git clone https://github.com/contiki-os/contiki

# Download sdn-wise-contiki
cd ~/contiki/tools
wget https://sdnwiselab.github.io/tools/sdn-wise_java.tar.gz
tar xvf sdn-wise_java.tar.gz
rm sdn-wise_java.tar.gz
cd cooja
git submodule update --init
ant jar_cooja
cd examples/sdn-wise_java
ant compile

# Download the controller
```

```
cd ~
wget http://sdn-wise.dieei.unict.it/tools/01-GetStarted.tar.gz
tar xvf 01-GetStarted.tar.gz
rm 01-GetStarted.tar.gz
cd ~/01-GetStarted
mvn package
cd ~

# Finish message
echo "The installations finished, source again the bashrc or exit and login again."
```

```
#!/bin/bash
## 1compile_onos.sh ##############################################################################
#
###################################### Description ##############################################
#
# This script is used to Start Onos controller
#
###################################### Arguments ################################################
#
# No argument is required
#
#################################################################################################

. ./functions.sh

Log "Onos is starting"
cd ~/onos &&
tools/build/onos-buck build onos --show-output && tools/build/onos-buck run onos-local -
- clean debug
```

```
#!/bin/bash
## 2start_onos.sh ################################################################################
#
###################################### Description ##############################################
#
# This script is used to start ONOS Shell
#
###################################### Arguments ################################################
#
# No argument is required
#
#################################################################################################

. ./functions.sh
Log "Onos shell is starting "
Log " When onos is ready write: app activate org.onosproject.sdnwise "
cd ~/onos && tools/test/bin/onos localhost
```

```
#!/bin/bash
## 3gui_onos.sh ######################################################################################
#
######################################## Description ################################################
#
# This script is used to start Firefox with ONOS GUI
#
######################################## Arguments ##################################################
#
# No argument is required
#
#############################################################################################################

. ./functions.sh

Log "Onos interface is starting..."
xdg-open http://localhost:8181/onos/ui
```

```
#!/bin/bash
## 4start_mininet.sh #################################################################################
#
######################################## Description ################################################
#
# This script is used to start Mininet using the standard topology
#
######################################## Arguments ##################################################
#
# No argument is required
#
#############################################################################################################

. ./functions.sh

Log "Mininet is starting with the known topology"
cd ~/
sudo mn --topo tree,depth=2,fanout=3 --controller=remote
```

```
#!/bin/bash
## 5startcooja.sh ####################################################################################
#
######################################## Description ################################################
#
# This script is used to start Cooja
#
######################################## Arguments ##################################################
#
# No argument is required
```

```
#
##############################################################################################################


. ./functions.sh

Log " Cooja is starting"
cd ~/sdn-wise-contiki/contiki/tools/cooja/ && ant run
```

```
#!/bin/bash
## functions.sh ##############################################################################################
#
###################################### Description ###########################################
#
# This script is holding the functions needed by the scripts.
#    Source this script in order to use the Functions.
#
###################################### Arguments ###########################################
#
# No argument is required
#
##############################################################################################################

function Log {
# Writes the argument as log for the notification
# $1: the message that we want to log
#
   typeset msg=$1
   typeset logline="<$(date +%d%b%Y' '%H:%M:%S)> $msg \n"
   printf "%b\n" "$logline" >&1
}
```

```
#!/bin/bash
## run_contiki_sdn-controller.sh ##############################################################################
#
###################################### Description ###########################################
#
# This script starting the SDN-WISE Get started Java controller
#
###################################### Arguments ###########################################
#
# No arguments required
#
##############################################################################################################
. ./functions.sh
Log "The controller 01-GetStarted is starting now"
cd ~/01-GetStarted
java -jar target/01-GetStarted.jar
```

```
#!/bin/bash
## run_CORAL.sh ##############################################################################
#
###################################### Description ##############################################
#
# This script starting the CORAL-SDN Implementation
#
###################################### Arguments ################################################
#
# No arguments required
#
#########################################################################################################
. ./functions.sh
Log "CORAL simple implementation is starting now"
Log "cooja is starting"
gnome-terminal -e "bash -c \" ~/SDN_scripts/startcooja.sh; exec bash\" "
sleep 60;

Log "Starting CORAL Controller"
gnome-terminal -e "bash -c \" cd ~/CORAL-SDN-Controller ; java -
jar CoralSDNController.jar; exec bash\" "
sleep 60;

Log "Coral Adapter is starting"
gnome-terminal -e "bash -c \" cd ~/coral-sdn-adapter-COOJA-runtime ; sudo ./coral-sdn-
adapter; exec bash\" "
```

```
#!/bin/bash
## run_sdn_wise_java.sh ######################################################################
#
###################################### Description ##############################################
#
# This script starting the SDN-WISE Java basic implementation
#
###################################### Arguments ################################################
#
# No arguments required
#
#########################################################################################################

. ./functions.sh
Log "SDN-WISE simple implementation is starting now"
cd ~/sdn-wise-java/ctrl/target
findBuild=$(ls -lrt | grep jar-with-dependencies | tail -1 | awk -F' ' '{print $9}')
java -jar $findBuild
```

```
#!/bin/bash
## start_Onos.sh ##############################################################################
#
##################################### Description ############################################
#
# This script is wrapper script and its usage is to start in a separate terminal all the scripts
#    needed for the SDN-WISE with ONOS.
#
##################################### Arguments ##############################################
#
# No argument is required
#
##############################################################################################

. ./functions.sh
Log "Each script is starting at new terminal"
Log "1compile_onos is starting at a new terminal"
gnome-terminal -e "bash -c \" ~/SDN_scripts/1compile_onos.sh; exec bash\" "
sleep 100;

for x in 2start_onos.sh 3gui_onos.sh 4start_mininet.sh 5startcooja.sh; do
   Log "$x is starting at a new terminal"
   gnome-terminal -e "bash -c \" ~/SDN_scripts/$x; exec bash\" "
   sleep 80;
done
```

```
#!/bin/bash
## startcooja.sh ##############################################################################
#
##################################### Description ############################################
#
# This script is used to start cooja in contiki
#
##################################### Arguments ##############################################
#
# No argument is required
#
##############################################################################################

. ./functions.sh

Log " Cooja is starting"
cd ~/contiki/tools/cooja && ant run
```

```
#!/bin/bash
## startSDN_Java.sh ###########################################################################
#
##################################### Description ############################################
#
# This wrapper script is used to executed all the other scripts needed in order to start Cooja an
d
#   the Java 01 controller for SDN-WISE.
#
##################################### Arguments ##############################################
#
# No argument is required
#
##############################################################################################

. ./functions.sh

Log "Cooja is starting at a new terminal"
gnome-terminal -e "bash -c \" ~/SDN_scripts/startcooja.sh; exec bash\" "
sleep 100;

gnome-terminal -e "bash -c \" ~/SDN_scripts/run_contiki_sdn-controller.sh; exec bash\" "
```