ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μαζική διαχείριση ασύρματων σημείων πρόσβασης μέσω διεπαφής ιστού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Μ. Πολυσίου

Επιβλέπων : Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μαζική διαχείριση ασύρματων σημείων πρόσβασης μέσω διεπαφής ιστού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Μ. Πολυσίου

Επιβλέπων : Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11η Ιουλίου 2018.

Αθήνα, Ιούλιος 2018

.....	.....	.....
Συκάς Ευστάθιος	Στασινόπουλος Γεώργιος	Ρουσσάκη Ιωάννα
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκουρη Καθηγήτρια Ε.Μ.Π.

.....
Δημήτριος Μ. Πολυσίου

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

Περίληψη

Στα πλαίσια της παρούσας διπλωματικής εργασίας αναπτύχθηκε μια εφαρμογή Django η οποία επιτρέπει την απομακρυσμένη διαχείριση των σημείων πρόσβασης ενός ασύρματου δικτύου μέσω μιας διεπαφής ιστού. Η εφαρμογή στηρίζεται στην πλατφόρμα λογισμικού OpenWISP 2 για τη διαχείριση της διάρθρωσης (configuration) ενσωματωμένων δικτυακών συσκευών με λειτουργικό σύστημα OpenWrt. Για τη διευκόλυνση της διαχείρισης μεγάλου αριθμού συσκευών υλοποιήθηκαν καινούργιες λειτουργίες όπως η ικανότητα αναζήτησης συσκευών με βάση την τοποθεσία ή το υλισμικό τους και η μαζική εφαρμογή προτύπων διάρθρωσης σε μια ομάδα συσκευών.

Η εφαρμογή υποστηρίζει την ύπαρξη πολλαπλών διαχειριστικών ομάδων, καθεμία από τις οποίες έχει την ιδιοκτησία ενός υποσυνόλου των συσκευών του δικτύου. Η είσοδος στην εφαρμογή γίνεται μέσω πρωτοκόλλου CAS 3. Κατά την είσοδό τους στην εφαρμογή, οι χρήστες ανατίθενται στη διαχειριστική τους ομάδα σύμφωνα με ένα χαρακτηριστικό που παρέχεται από τον εξυπηρετητή CAS. Επίσης, υποστηρίζεται η ύπαρξη υπέρ-διαχειριστών ικανών να αλλάζουν τα δικαιώματα των διαχειριστικών ομάδων και να μεταβιβάζουν την ιδιοκτησία των συσκευών σε διαφορετική ομάδα.

Με χρήση της τεχνολογίας OpenStreetMap επιτυγχάνεται η γεωγραφική απεικόνιση του δικτύου στο χάρτη. Επιπλέον, η εφαρμογή επιτρέπει τη συλλογή μετρικών για τη δικτυακή παρακολούθηση των συσκευών με χρήση του δαίμονα Collectd και την οπτικοποίηση των μετρικών αυτών μέσω γραφημάτων του εργαλείου RRDtool. Η εγκατάσταση της εφαρμογής είναι αυτοματοποιημένη μέσω των εργαλείων ansible και vagrant.

Λέξεις Κλειδιά:
Σημείο πρόσβασης, OpenWISP, NetJSON, OpenWrt, διαχείριση διάρθρωσης, Collectd, RRDtool, Django, ansible, vagrant

Abstract

Within the scope of this diploma thesis, a Django application was developed which allows allows for the remote management of a wireless network's access points through a web interface. The application is based on the OpenWISP 2 software platform for managing the configuration of embedded network devices running the OpenWrt operating system. In order to facilitate the management of a large number of devices certain new features were implemented, such as the ability to search for devices based on their location or hardware and the mass application of configuration templates to a group of devices.

The application supports the existence of multiple administration groups, each having ownership over a subset of the network's devices. Users log in to the application using the CAS 3 protocol. During authentication, they are assigned to their administration group according to an attribute provided by the CAS server. Furthermore, the application supports the existence of super-administrators capable of altering the permissions of each administration group and transferring ownership of a device to a different group.

The application displays the network on a map according to each device's geographic location using OpenStreetMap. Additionally, the application enables the collection of metrics from the devices for the purposes of network monitoring using the Collectd daemon and the visualization of said metrics with graphs generated by RRDtool. Deploying the application is automated using the tools ansible and vagrant.

Keywords:
Access point, OpenWISP, NetJSON, OpenWrt, Configuration management, Collectd, RRDtool, Django, ansible, vagrant

Ευχαριστίες

Καταρχάς θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κ. Ευστάθιο Συκά για την ανάθεση του θέματος της εργασίας καθώς και για την εμπιστοσύνη που μου έδειξε κατά την εκπόνησή της. Επίσης, του είμαι ευγνώμων για την άμεση επικοινωνία και τις πολύτιμες συμβουλές του.

Ιδιαίτερες ευχαριστίες οφείλω και στο δρ. Δημήτρη Καλογερά για το χρόνο που μου διέθεσε και την εξαιρετική καθοδήγησή του. Χωρίς την απαραίτητη και πολύ σημαντική βοήθειά του θα ήταν αδύνατη η ολοκλήρωση της εργασίας.

Τελειώνοντας, θέλω να ευχαριστήσω τους γονείς μου για την υποστήριξη και την υπομονή τους κατά τη διεξαγωγή της εργασίας αλλά και καθ' όλη τη διάρκεια των σπουδών μου.

Πίνακας περιεχομένων

Κεφάλαιο 1

Εισαγωγή

Με τον όρο **ασύρματο σημείο πρόσβασης** ή **σταθμός βάσης** αναφερόμαστε στις συσκευές που συνδέει μεταξύ τους ασύρματες συσκευές επικοινωνίας για το σχηματισμό ενός ασύρματου δικτύου. Καθώς η εμβέλεια και οι πόροι ενός ασύρματου σημείου πρόσβασης είναι περιορισμένοι, για την κάλυψη ενός μεγάλου χώρου ή την εξυπηρέτηση μεγάλου αριθμού χρηστών απαιτείται ένα δίκτυο από σημεία πρόσβασης. Ένα ασύρματο δίκτυο που καλύπτει μια πόλη μπορεί να περιέχει δεκάδες χιλιάδες σημεία πρόσβασης [1]. Ακόμη, τα δημοτικά δίκτυα δεκάδων πόλεων μπορεί να ενώνονται στο πλαίσιο ενός εθνικού δικτύου. Η διαχείριση των ασύρματων σημείων πρόσβασης ενός τόσο μεγάλου δικτύου απαιτεί αυτοματοποίηση.

Σκοπός αυτής της εργασίας είναι η δημιουργία μιας διεπαφής ιστού που επιτρέπει την κεντρική διαχείριση ενός μεγάλου αριθμού ασύρματων σημείων πρόσβασης. Μέσω της διεπαφής αυτής μπορούμε εύκολα να διαπιστώσουμε την καλή λειτουργία των συσκευών μας, να περιγράψουμε αλλαγές στην επιθυμητή δικτυακή τους διάρθρωση και να εφαρμόσουμε τις αλλαγές αυτές μαζικά σε ένα μεγάλο πλήθος συσκευών. Επιπλέον, η εφαρμογή μας επιτρέπει την κατανεμημένη διαχείριση του δικτύου από διαφορετικές διαχειριστικές ομάδες, καθεμία από τις οποίες είναι υπεύθυνη για ένα υποσύνολο των συσκευών του δικτύου. Η εφαρμογή στηρίζεται στην πλατφόρμα λογισμικού OpenWISP και υποστηρίζει μηχανήματα ανεξαρτήτως κατασκευαστή στα οποία έχει εγκατασταθεί το λειτουργικό σύστημα ανοικτού κώδικα OpenWrt.

Η εργασία χωρίζεται σε 6 κεφάλαια. Ύστερα από μια σύντομη εισαγωγή, μελετούμε στο **κεφάλαιο 2** το πρόβλημα της διαχείρισης ενσωματωμένων δικτυακών συσκευών και εξηγούμε τους λόγους για τους οποίους επιλέξαμε να χρησιμοποιήσουμε την πλατφόρμα OpenWISP. Στο **κεφάλαιο 3** παρουσιάζουμε το λειτουργικό σύστημα OpenWrt και τα αρχεία διάρθρωσης UCI. Στο **κεφάλαιο 4** παρουσιάζουμε την πλατφόρμα λογισμικού OpenWISP και μελετούμε την αρχιτεκτονική της δεύτερης έκδοσής της. Στο **κεφάλαιο 5** εξηγούμε τις δικές μας προσθήκες στο λογισμικό OpenWISP οι οποίες επιτρέπουν την κατανεμημένη διαχείριση του δικτύου, την παρακολούθηση των συσκευών και τη μαζική εφαρμογή εφαρμογή αλλαγών στη διάρθρωσή τους. Τέλος, στο **κεφάλαιο 6** δίνουμε οδηγίες για την εγκατάσταση του λογισμικού OpenWISP μαζί με τις προσθήκες μας, είτε χειροκίνητα είτε με χρήση των εργαλείων ansible ή vagrant.

Κεφάλαιο 2

Διαχείριση ενσωματωμένων δικτυακών συσκευών

2.1. Το διαχειριστικό μοντέλο αναφοράς FCAPS

Η διαχείριση δικτύων είναι ένας εκτενής τομέας που περιλαμβάνει πολλές λειτουργίες. Με στόχο την καλύτερη εποπτεία της διαχείρισης δικτύων, ο οργανισμός OSI εισήγαγε στις αρχές του 1980 το διαχειριστικό μοντέλο αναφοράς FCAPS [1], το οποίο ομαδοποιεί τους στόχους της διαχείρισης δικτύων σε πέντε κατηγορίες:

1. Διαχείριση Σφαλμάτων (Fault Management) :
Στόχος της είναι να ανακαλύψουμε, να απομονώσουμε, να διορθώσουμε και να καταγράψουμε τα σφάλματα σε ένα δίκτυο.

2. Διαχείριση Διάρθρωσης (Configuration Management):
Αφορά την προσθήκη, διαγραφή και τροποποίηση των δεδομένων διάρθρωσης του δικτύου.

3. Λογιστική Διαχείριση (Accounting Management):
Περιλαμβάνει τη λήψη στατιστικών χρήσης αλλά και τον έλεγχο της πρόσβασης στο δίκτυο από τους χρήστες.

4. Διαχείριση Επιδόσεων (Performance Management):
Έχει να κάνει με την ανάλυση της υπάρχουσας απόδοσης του δικτύου και τη συλλογή πληροφοριών για τη μελλοντική του εξέλιξη. Επίσης, έχει να κάνει με το εάν οι επί μέρους δικτυακές υπηρεσίες είναι διαθέσιμες , γρήγορες και ανταποκρίνονται στις επιθυμίες του οργανισμού και των χρηστών.

5. Διαχείριση Ασφάλειας (Security Management):
Ασχολείται με τον έλεγχο πρόσβασης των χρηστών και με έλεγχο στα συστήματα και τις διαδικασίες ασφαλείας.

Από της πέντε κατηγορίες της διαχείρισης δικτύων, η σημαντικότερη για τις ενσωματωμένες δικτυακές συσκευές είναι η διαχείριση της διάρθρωσης. Οι υπόλοιπες κατηγορίες στηρίζονται σε αυτή για να επιτύχουν το επιθυμητό τους αποτέλεσμα. Για παράδειγμα, η διαχείριση σφαλμάτων
απαιτεί την αναδιάρθρωση (reconfiguration) της συσκευής για την επίλυση των προβλημάτων, η διαχείριση επιδόσεων απαιτεί αναδιάρθρωση για τη βελτιστοποίηση της λειτουργίας του δικτύου και η διαχείριση ασφάλειας απαιτεί αναδιάρθρωση για την επίλυση κάποιας παραβίασης της ασφάλειας. Επιπλέον, οι ενσωματωμένες συσκευές δεν μπορούν να λειτουργήσουν καθόλου χωρίς σωστή διάρθρωση. Χωρίς τη σωστή ρύθμιση των δικτυακών συσκευών, δεν υπάρχει δίκτυο.

Η κακή διαχείριση της διάρθρωσης του δικτύου είναι πολύ σημαντικό πρόβλημα και ευθύνεται για περισσότερο από το 50% των σφαλμάτων στα δίκτυα υπολογιστών [2][3][4]. Ορισμένα σφάλματα μπορούν να έχουν

δραματικά αποτελέσματα όπως να εισάγουν αδυναμίες στην ασφάλεια του δικτύου ή να οδηγήσουν σε παγκόσμιες διακοπές σύνδεσης. Για παράδειγμα, σφάλματα στην παραμετροποίηση του AS 9121 είχαν ως αποτέλεσμα να χαθεί η κίνηση δεκάδων χιλιάδων δικτύων [5]. Οι Hoelzle και Barrose αναφέρουν ότι η κακή παραμετροποίηση είναι η δεύτερη μεγαλύτερη αιτία αποτυχίας σε επίπεδο υπηρεσίας σε μια υπηρεσία της Google [6]. Μια άλλη έρευνα [7] αναφέρει πως οι υπηρεσίες νέφους της Amazon, το Microsoft Azure και το Facebook αντιμετώπισαν διαταραχές στις υπηρεσίες τους λόγω κακής παραμετροποίησης οι οποίες επηρέασαν εκατομμύρια χρήστες.

Δεν είναι δύσκολο να καταλάβουμε για ποιο λόγο η διαχείριση της διάρθρωσης ενός δικτύου παρουσιάζει τόσο μεγάλη πολυπλοκότητα. Η γλώσσες των αρχείων παραμετροποίησης είναι χαμηλού επιπέδου και διαφέρουν από συσκευή σε συσκευή. Επιπλέον, τα αρχεία παραμετροποίησης είναι μεγάλα και μπορούν εύκολα να περιέχουν χιλιάδες γραμμές εντολών σε κάθε δρομολογητή. Επίσης, η παραμετροποίηση ενός δρομολογητή μπορεί να επηρεάσει τη συμπεριφορά πολλών άλλων δρομολογητών του δικτύου. Επομένως, η πολυπλοκότητα της διαχείριση της διάρθρωσης ενός μεγάλου δικτύου είναι ανάλογη με τον προγραμματισμό κατανεμημένων συστημάτων σε γλώσσα χαμηλού επιπέδου.

Από τους παραπάνω λόγους είναι προφανή τα πλεονεκτήματα της χρήσης ενός εργαλείου το οποίο επιτρέπει την κεντρική διαχείριση της διάρθρωσης του δικτύου μέσω μιας διεπαφής φιλικής προς το χρήστη, όπως ένα γραφικό περιβάλλον.

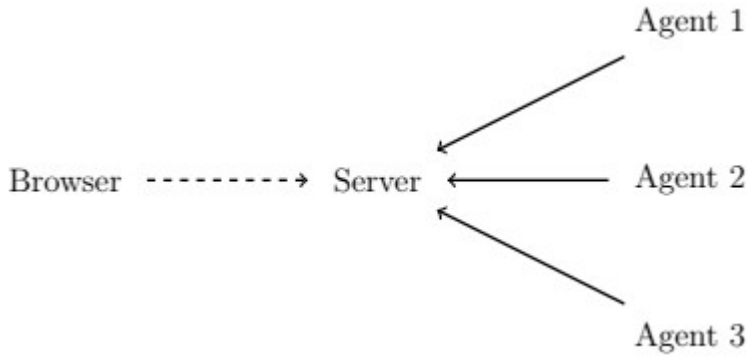
2.2. Εργαλεία διαχείρισης διάρθρωσης

Η διαχείριση σφαλμάτων και επιδόσεων των ενσωματωμένων δικτυακών συσκευών μπορεί να επιτευχθεί με τη χρήση εργαλείων παρακολούθησης του δικτύου όπως το Zabbix, ή το Nagios. Αντίστοιχα, η λογιστική διαχείριση στο επίπεδο των δρομολογητών ή ασύρματων σημείων πρόσβασης μπορεί να επιτευχθεί με χρήση ενός εξυπηρετητή RADIUS, όπως το FreeRadius. Τι εργαλεία όμως υπάρχουν για τη διαχείριση διάρθρωσης των ενσωματωμένων δικτυακών συσκευών;

Τα τελευταία χρόνια έχουν αναπτυχθεί αρκετά εργαλεία κεντρικής διαχείριση διάρθρωσης δικτύου ανοικτού κώδικα, όπως το Puppet, CFEngine, Chef, Salt και Ansible. Τα εργαλεία αυτά μπορούν να χωριστούν σε δύο κατηγορίες ανάλογα με την αρχιτεκτονική που ακολουθούν: αρχιτεκτονική πράκτορα-εξυπηρετητή (agent-server) ή αρχιτεκτονική χωρίς πράκτορα (agentless).

Agent-Server

Τα εργαλεία που ακολουθούν αυτήν την αρχιτεκτονική χρησιμοποιούν προγράμματα που αποκαλούνται πράκτορες (agents) τα οποία τρέχουν στα απομακρυσμένα μηχανήματα που θέλουμε να διαχειριστούμε. Τα προγράμματα αυτά επικοινωνούν με έναν κεντρικό εξυπηρετητή (server), πληροφορούνται από αυτόν εάν υπάρχει αλλαγή στην επιθυμητή διάρθρωση και την εφαρμόζουν στο μηχάνημα που τρέχουν.



Σχήμα 2.1: *Αρχιτεκτονική Agent-Server*

Βασικό μειονέκτημα τις χρήσης αυτής της αρχιτεκτονικής για τη διαχείριση ενσωματωμένων δικτυακών συσκευών είναι η κατανάλωση πόρων από τον πράκτορα. Τα εργαλεία αυτά, όταν είναι σχεδιασμένα για τη διαχείριση εξυπηρετητών και όχι δρομολογητών, χρησιμοποιούν πράκτορες των οποίων οι απαιτήσεις σε μνήμη ή ακόμα και αποθηκευτικό χώρο είναι πολύ μεγάλες για μηχανήματα που μπορεί να έχουν μόνο μερικά megabyte μνήμης.

Agentless

Τα εργαλεία που ακολουθούν την αρχιτεκτονική αυτή δε χρησιμοποιούν πράκτορες στις συσκευές που θέλουμε να διαχειριστούμε. Όταν τροποποιήσουμε τη διάρθρωση μερικών συσκευών στον κεντρικό εξυπηρετητή, η εφαρμογή που τρέχει στον εξυπηρετητή συνδέεται με τη σειρά στις συσκευές αυτές (συνήθως μέσω ssh) και εκτελεί τις απαραίτητες ενέργειες ώστε η συσκευή να έρθει στην επιθυμητή κατάσταση. Δυστυχώς, η έλλειψη πρακτόρων δεν συνεπάγεται πάντα μείωση στην κατανάλωση πόρων των συσκευών ή τη χρήση τους χωρίς εγκατάσταση λογισμικού. Για παράδειγμα, το εργαλείο Ansible, που ακολουθεί την αρχιτεκτονική αυτή, απαιτεί την ύπαρξη μεταφραστή της γλώσσας Python στα μηχανήματα που θέλουμε να διαχειριστούμε. Επιπλέον, η αρχιτεκτονική αυτή μειονεκτεί όσον αφορά την κλιμάκωση, αφού με κάθε συσκευή που προσθέτουμε στο δίκτυο αυξάνεται γραμμικά ο χρόνος για την εφαρμογή νέας διάρθρωσης στις συσκευές.

Ανεξάρτητα από την αρχιτεκτονική τους, ένα πρόβλημα που συναντάται στα εργαλεία διαχείρισης διάρθρωσης σχετικά με τις ενσωματωμένες δικτυακές συσκευές είναι η έλλειψη γενικότητας. Τα εργαλεία πολύ συχνά έχουν περιορισμούς στο είδος των μηχανημάτων που υποστηρίζουν. Για παράδειγμα, ένα εργαλείο μπορεί να υποστηρίζει μόνο μηχανήματα συγκεκριμένης εταιρείας. Επίσης, δεν παρέχουν ένα κοινό τρόπο διαχείρισης όλων των μηχανημάτων που υποστηρίζουν, πράγμα που προκαλεί προβλήματα σε ετερογενή δίκτυα με διαφορετικούς τύπους μηχανημάτων.

2.3. OpenWISP

Η πλατφόρμα λογισμικού OpenWISP, την οποία θα χρησιμοποιήσουμε, είναι σχεδιασμένη για τη διαχείριση ασύρματων σημείων πρόσβασης και άλλων ενσωματωμένων δικτυακών συσκευών. Αποτελείται από ένα πλήθος εφαρμογών σχετικές με τη διαχείριση δικτύων, οι οποίες μπορούν να χρησιμοποιηθούν ανεξάρτητα η μία από την άλλη. Εξέχουσα θέση έχει η εφαρμογή της κεντρικής διαχείρισης της διάρθρωσης των ελεγχόμενων συσκευών.

Η εφαρμογή διαχείρισης διάρθρωσης του OpenWISP ακολουθεί την αρχιτεκτονική agent-server, πράγμα που τις δίνει μεγάλες δυνατότητες κλιμάκωσης. Επιπλέον, οι agents που χρησιμοποιεί είναι σχεδιασμένοι να τρέχουν σε ενσωματωμένες συσκευές, με αποτέλεσμα να έχουν πολύ μικρή κατανάλωση πόρων του συστήματος και να μην έχουν προαπαιτούμενα για την εγκατάστασή τους.

Η πλατφόρμα OpenWISP είναι συμβατή με το λειτουργικό σύστημα OpenWrt, μια διανομή Linux για ενσωματωμένες δικτυακές συσκευές. Το OpenWrt υποστηρίζει ένα μεγάλο πλήθος συσκευών πολλών εταιριών, λύνοντας έτσι σε μεγάλο βαθμό το πρόβλημα της γενίκευσης. Επιπλέον, επιτρέπει την εγκατάσταση πακέτων λογισμικού μέσω του διαχειριστή πακέτων opkg. Με τον τρόπο αυτό μπορούμε να προσθέσουμε όποιες δυνατότητες επιθυμούμε στη συσκευή μας, όπως πχ. Δυνατότητες traffic shaping, QoS (Quality of Service), κλπ.

Μελλοντικά, η πλατφόρμα OpenWISP στοχεύει να εξυπηρετεί κι άλλα λειτουργικά συστήματα πέρα από το OpenWrt και τα παράγωγά του. Ήδη υποστηρίζει σε πειραματικό επίπεδο τα συστήματα AirOS και Raspbian. Για τους λόγους αυτούς, η πλατφόρμα OpenWISP αποτελεί καινοτόμα λύση στο πρόβλημα της διαχείρισης ασύρματων σημείων πρόσβασης και άλλων ενσωματωμένων δικτυακών συσκευών.

Κεφάλαιο 3

OpenWrt

Το OpenWrt [3.1] είναι μια ανοιχτή διανομή GNU / Linux για ενσωματωμένες δικτυακές συσκευές. Τα βασικά κομμάτια είναι ο πυρήνας του Linux, η βιβλιοθήκη UClibc ή musl και το λογισμικό BusyBox. Όλα τα κομμάτια έχουν βελτιστοποιηθεί για μέγεθος, ώστε να είναι αρκετά μικρά για να χωρέσουν στον περιορισμένο χώρο αποθήκευσης και τη μνήμη που διαθέτουν οι οικιακοί δρομολογητές.

Το OpenWrt παραμετροποιείται χρησιμοποιώντας τη γραμμή εντολών ή μια διεπαφή ιστού (πχ. LuCI). Υπάρχουν διαθέσιμα περίπου 3500 προαιρετικά πακέτα λογισμικού για εγκατάσταση μέσω του συστήματος διαχείρισης πακέτων opkg.

3.1 Ιστορικό

Το OpenWrt δημιουργήθηκε επειδή η Linksys έγραψε το firmware των δρομολογητών της σειράς WRT54G χρησιμοποιώντας δημόσια διαθέσιμο κώδικα υπό την άδεια GPL. Σύμφωνα με τους όρους της άδειας, η Linksys υποχρεώθηκε να καταστήσει διαθέσιμο τον κώδικα της τροποποιημένης έκδοσής της υπό την ίδια άδεια. Χρησιμοποιώντας αυτόν τον κώδικα ως βάση, ανεξάρτητοι προγραμματιστές δημιούργησαν μια διανομή Linux με πολλά χαρακτηριστικά που παλαιότερα δεν ήταν διαθέσιμα για οικιακούς δρομολογητές. Αρχικά το OpenWrt υποστήριζε μόνο μηχανήματα της σειράς WRT54G, αλλά πλέον υποστηρίζει ένα μεγάλο εύρος αρχιτεκτονικών, όπως mips, arm, powerpc και x86.

3.2 Δημιουργία εικόνων συστήματος OpenWrt [3.2]

Τα μηχανήματα για τα οποία είναι σχεδιασμένο το OpenWrt διαθέτουν περιορισμένη υπολογιστική δύναμη. Επομένως, η δημιουργία πακέτων και εικόνων συστήματος OpenWrt για τα μηχανήματα αυτά πρέπει να γίνει σε πιο ισχυρούς υπολογιστές (cross compilation). Τη διαδικασία αυτή διευκολύνει η αλυσίδα εργαλείων OpenWrt build system, η οποία τρέχει σε λειτουργικά Linux, BSD και OS X. Μπορούμε να κατεβάσουμε την τελευταία έκδοση του OpenWrt build system μέσω git με την εντολή:

```
git clone https://github.com/openwrt/openwrt.git
```

Εάν θέλουμε να χρησιμοποιήσουμε άλλη έκδοση, πχ. την chaos calmer, κατεβάζουμε το αντίστοιχο branch:

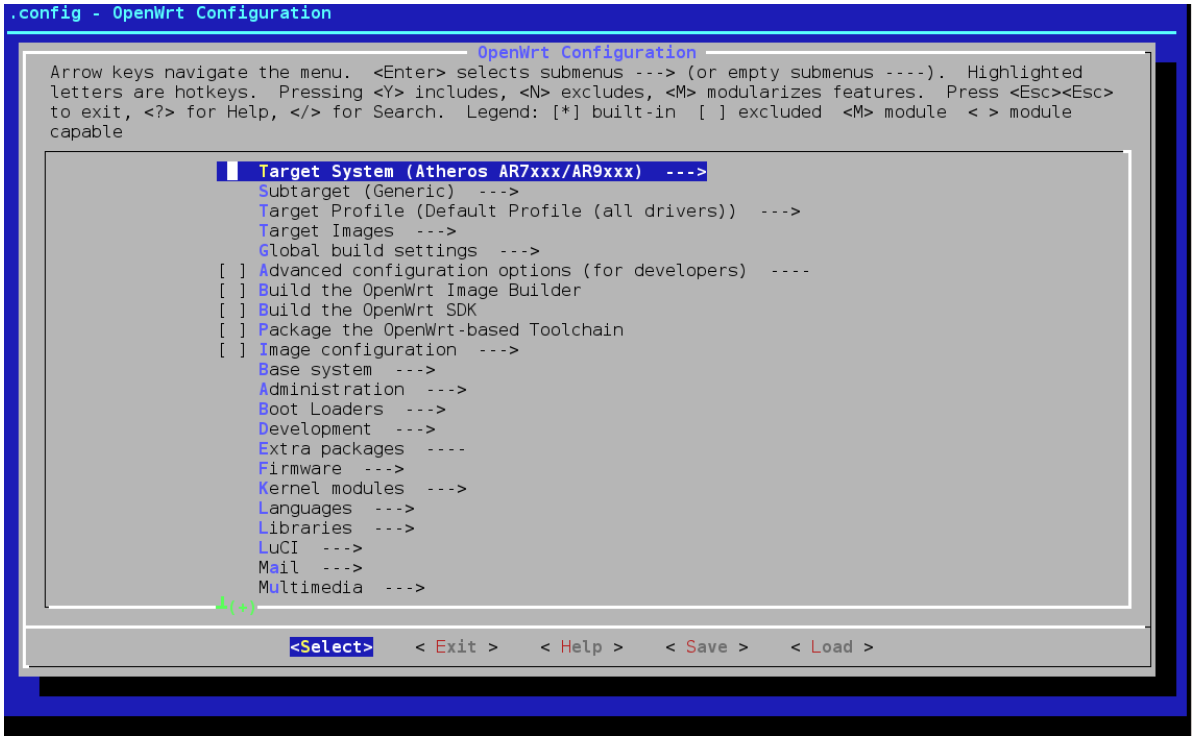
```
git clone https://www.github.com/openwrt/openwrt -b
chaos_calmer
```

Έπειτα, κατεβάζουμε τον κώδικα για όλα τα διαθέσιμα πακέτα. Μέσα στο φάκελο του build system εκτελούμε:

```
./scripts/feeds update -a
./scripts/feeds install -a
```

Τώρα μπορούμε να επιλέξουμε από το μενού τις αρχιτεκτονικές για τις οποίες θέλουμε να δημιουργήσουμε εικόνες και τα πακέτα που θέλουμε να συμπεριλάβουμε στις εικόνες. Επίσης, μπορούμε να δημιουργήσουμε πακέτα χωρίς να τα συμπεριλάβουμε σε εικόνες, ώστε να μπορούμε να τα εγκαταστήσουμε μέσω opkg. Ανοίγουμε το μενού με την εντολή

```
make menuconfig
```



Μπορούμε ακόμα να προσθέσουμε αρχεία στο φάκελο ./files/ τα οποία θα συμπεριληφθούν στις εικόνες που δημιουργούμε. Για παράδειγμα, άμα θέλουμε να αλλάξουμε τις ρυθμίσεις για το firewall, μπορούμε να προσθέσουμε ένα δικό μας αρχείο στη θέση `./files/etc/firewall` .

Τέλος, δημιουργούμε τις εικόνες εκτελώντας μια εντολή όπως `make` ή `ionice -c 3 nice -n19 make -j 2` .

3.3 Εγκατάσταση του OpenWrt [3.3]

Ο τρόπος εγκατάστασης του OpenWrt εξαρτάται από τη συσκευή στην οποία θέλουμε να το εγκαταστήσουμε. Στη γενική περίπτωση, υπάρχουν 4 τρόποι εγκατάστασης του OpenWrt firmware.

1. Μέσω του OEM firmware της συσκευής: Ανεβάζουμε το firmware στο μηχάνημα μέσω της διεπαφής ιστού του OEM firmware του μηχανήματος.
2. Μέσω Bootloader και θύρας Ethernet: Ανεβάζουμε το firmware μέσω FTP ή TFTP.
3. Μέσω Bootloader και σειριακής θύρας: Ανεβάζουμε το firmware μέσω της σειριακής θύρας της συσκευής.
4. Μέσω JTAG: Ανεβάζουμε το firmware μέσω της διεπαφής JTAG.

3.4 Αρχεία παραμετροποίησης του OpenWrt [3.4]

Τα αρχεία παραμετροποίησης των συστημάτων Linux συνήθως βρίσκονται στο φάκελο /etc/ και το καθένα έχει το δικό του συντακτικό. Στο OpenWrt έχει γίνει μια προσπάθεια για ενοποίηση όλων των αρχείων παραμετροποίησης υπό μια κεντρική διεπαφή. Το UCI (Unified Configuration Interface) είναι ένα εργαλείο γραμμένο σε C που μας επιτρέπει να επεξεργαζόμαστε τα αρχεία παραμετροποίησης με ένα κεντρικό τρόπο. Τα αρχεία παραμετροποίησης που είναι βασισμένα στο UCI βρίσκονται στο φάκελο /etc/config/ και χρησιμοποιούν ένα κοινό συντακτικό. Μερικά βασικά αρχεία παραμετροποίησης του OpenWrt βασισμένα στο UCI είναι τα παρακάτω:

- `/etc/config/system`

Περιλαμβάνει ρυθμίσεις του συστήματος, όπως hostname και timezone.

- `/etc/config/network`

Περιλαμβάνει ρυθμίσεις των δικτυακών διεπαφών.

- `/etc/config/dhcp`

Περιλαμβάνει ρυθμίσεις του dns και dhcp.

- `/etc/config/wireless`

Περιλαμβάνει ρυθμίσεις των ασύρματων διεπαφών.

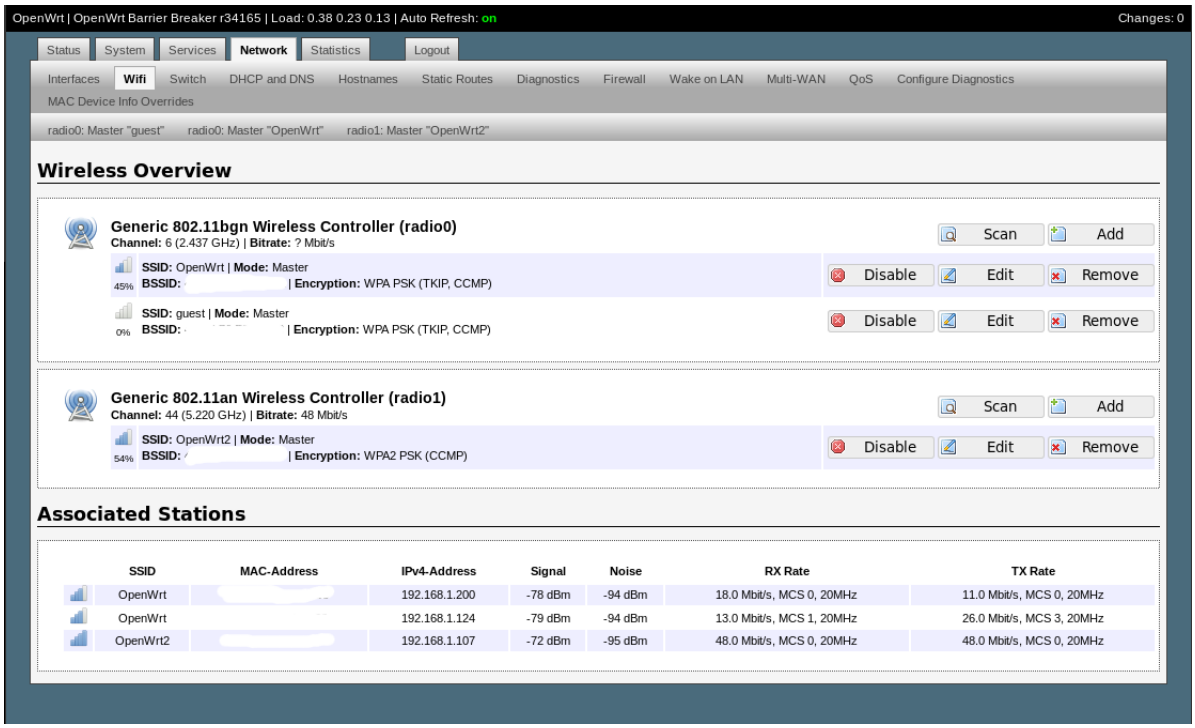
Παρακάτω παραθέτουμε ένα παράδειγμα ενός απλού αρχείου παραμετροποίησης που χρησιμοποιεί το συντακτικό του UCI.

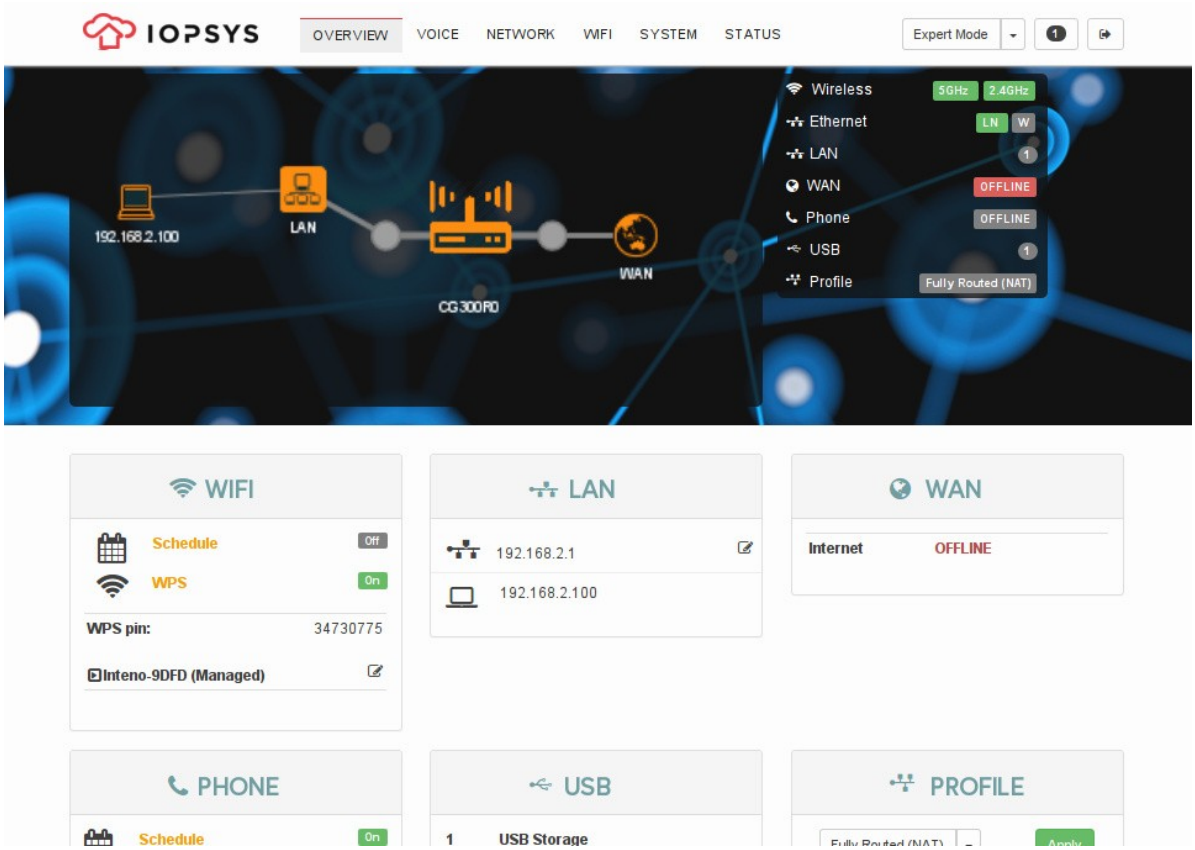
```
package 'example'

config 'example' 'test'
        option   'string'      'some value'
        option   'boolean'     '1'
        list     'collection'  'first item'
        list     'collection'  'second item'
```

Τα αρχεία χωρίζονται σε τομείς (sections). Η δήλωση `config 'example' 'test'` ορίζει την αρχή ενός τομέα με τύπο example και όνομα test. Μπορούμε επίσης να έχουμε ανώνυμους τομείς, μόνο με τύπο, χωρίς όνομα. Ο τύπος είναι σημαντικός για να γνωρίζουν τα προγράμματα που επεξεργάζονται τα αρχεία πως να χειριστούν τις επιλογές που εσωκλείονται στον τομέα. Οι γραμμές `option 'string' 'some value'` και `option 'boolean' '1'` ορίζουν απλές τιμές μέσα στον τομέα. Στις γραμμές με τη λέξη-κλειδί list, ορίζουμε μια επιλογή με πολλαπλές τιμές, στην περίπτωσή μας την επιλογή collection.

Υπάρχουν δύο τρόποι να επεξεργαστούμε τα αρχεία παραμετροποίησης που είναι βασισμένα στο UCI. Ο πρώτος είναι να χρησιμοποιήσουμε ένα πρόγραμμα επεξεργασίας κειμένου ώστε να κάνουμε αλλαγές απευθείας στο αρχείο. Ο δεύτερος είναι να χρησιμοποιήσουμε ένα πρόγραμμα βασισμένο στη βιβλιοθήκη libuci, όπως το εργαλείο γραμμής εντολών uci ή τη διεπαφή ιστού Luci.




IOPSYS
OVERVIEW
VOICE
NETWORK
WIFI
SYSTEM
STATUS
Expert Mode
192.168.2.100
LAN
CG300P0
WAN
Wireless
5GHz
2.4GHz
Ethernet
LAN
WAN
OFFLINE
Phone
OFFLINE
USB
Profile
Fully Routed (NAT)
WIFI
Schedule
Off
WPS
On
WPS pin:
34730775
Inteno-9DFD (Managed)
LAN
192.168.2.1
192.168.2.100
WAN
Internet
OFFLINE
PHONE
Schedule
On
USB
1
USB Storage
PROFILE
Apply

Κεφάλαιο 4

OpenWISP

Η πλατφόρμα λογισμικού OpenWISP δημιουργήθηκε το 2008 με σκοπό να χρησιμοποιηθεί για τη μαζική διαχείριση ασύρματων σημείων πρόσβασης ως μέρος του έργου ProvinciaWiFi. Το έργο αυτό (αργότερα μετονομασμένο σε WiFi Metropolitano) είχε ως στόχο να δημιουργήσει ένα δωρεάν δημόσιο δίκτυο Wi-Fi στη Ρώμη και σε 120 άλλες πόλεις με σύνολο περίπου 4.5 εκατομμύρια κατοίκους. Από την αρχή είχε αποφασιστεί ότι το έργο θα χρησιμοποιούσε λογισμικό και λειτουργικά συστήματα ανοικτού κώδικα και πως οποιαδήποτε αλλαγή ή βελτίωση θα καθίσταται διαθέσιμη στην κοινότητα του ανοικτού λογισμικού. Σε αυτό το πλαίσιο δημιουργήθηκε το OpenWISP ως μια πλατφόρμα για την κεντρική διαχείριση γεωγραφικά απομακρυσμένων σημείων πρόσβασης.

4.1. OpenWISP 1 [4.1]

Η πρώτη έκδοση του OpenWISP αναπτύχθηκε το 2008 με 2011 και ήταν σε χρήση μέχρι το 2016 Χρησιμοποιούσε δύο βασικά κομμάτια λογισμικού: το πακέτο OpenWISP Firmware (OWF) και τη εφαρμογή OpenWISP Manager (OWM). Το OpenWISP Firmware είναι ένα πακέτο για το λειτουργικό σύστημα OpenWrt το οποίο περιέχει ένα σύνολο από shell και web cgi scripts και παρέχει:

1. ένα δαίμονα που περιοδικά ανακτά τις δικτυακές ρυθμίσεις της συσκευής από έναν εξυπηρετητή που τρέχει την εφαρμογή OWM

2. μια διεπαφή ιστού που επιτρέπει τη βασική παραμετροποίηση και την πραγματοποίηση δοκιμών ώστε να εξασφαλιστεί η καλή λειτουργία του μηχανήματος και η επικοινωνία με τον εξυπηρετητή OWM

Ο OpenWISP Manager είναι μια εφαρμογή Ruby on Rails η οποία τρέχει σε έναν κεντρικό εξυπηρετητή, επικοινωνεί με μηχανήματα OpenWrt εξοπλισμένα με το πακέτο OWF και επιτρέπει την παραμετροποίησή τους από μια κεντρική διεπαφή.

Πέρα από αυτούς τους βασικούς πυλώνες, η πρώτη έκδοση προσφέρει 3 επιπλέον εφαρμογές:

- OpenWISP User System Management (OpenWUMS)

Μια εφαρμογή για τον αυτό-εφοδιασμό των διαπιστευτηρίων και την αυτό-διαχείρηση των δεδομένων των χρηστών. Επιτρέπει τη δημιουργία λογαριασμού χρήστη μέσω τηλεφώνου και την εποπτεία του ιστορικού ενός χρήστη μέσω γραφημάτων.

- OpenWISP Geographic Monitoring (OpenWGM)

Μια εφαρμογή που παρακολουθεί τα σημεία πρόσβασης και τα εμφανίζει στο χάρτη μαζί με πληροφορίες σε μορφή γραφημάτων σχετικές με τη διαθεσιμότητά τους.

- OpenWISP Captive Portal Manager (OWCPM)

Βασισμένη στο λογισμικό netfilter για λειτουργικό Linux, η εφαρμογή αυτή επιτρέπει να ρυθμίσουμε τα σημεία πρόσβασης ώστε να χρησιμοποιούν ένα captive portal.

4.2. OpenWISP 2

Η πρώτη έκδοση του OpenWISP, αν και είχε υιοθετήσει τις αρχές του ανοικτού κώδικα, δεν κατάφερε να προσελκύσει πολλούς προγραμματιστές της κοινότητας του ανοικτού κώδικα ή να βρει χρήση πέρα από τα δημοτικά δίκτυα wifi. Ως βασική αίτια αναγνωρίστηκαν οι σημαντικοί περιορισμοί που παρουσίαζε η εφαρμογή OpenWISP Manager, οι οποίοι καθιστούσαν δύσκολη την επέκταση ή την παραμετροποίησή της [4.2].

Οι πιο σημαντικοί περιορισμοί του OWM είναι οι παρακάτω [4.3]:

1. Ο κώδικας που παράγει τα αρχεία τη αρχεία παραμετροποίησης UCI των συσκευών OpenWrt είναι στενά συσχετισμένος με το πλαίσιο ανάπτυξης λογισμικού Ruby on Rails. Επομένως, οποιαδήποτε αλλαγή στον τρόπο παραγωγής των αρχείων παραμετροποίησης απαιτεί εκμάθηση του συγκεκριμένου πλαισίου από τον τελικό χρήστη.

2. Τα παραγόμενα αρχεία παραμετροποίησης μπορούν να χρησιμοποιηθούν μόνο από συσκευές στις οποίες είναι εγκατεστημένο το OpenWISP Firmware. Επομένως, ο OWM δε μπορεί να επεκταθεί ώστε να ελέγχει συσκευές που χρησιμοποιούν άλλο λειτουργικό, ακόμα και απλά μηχανήματα OpenWrt.

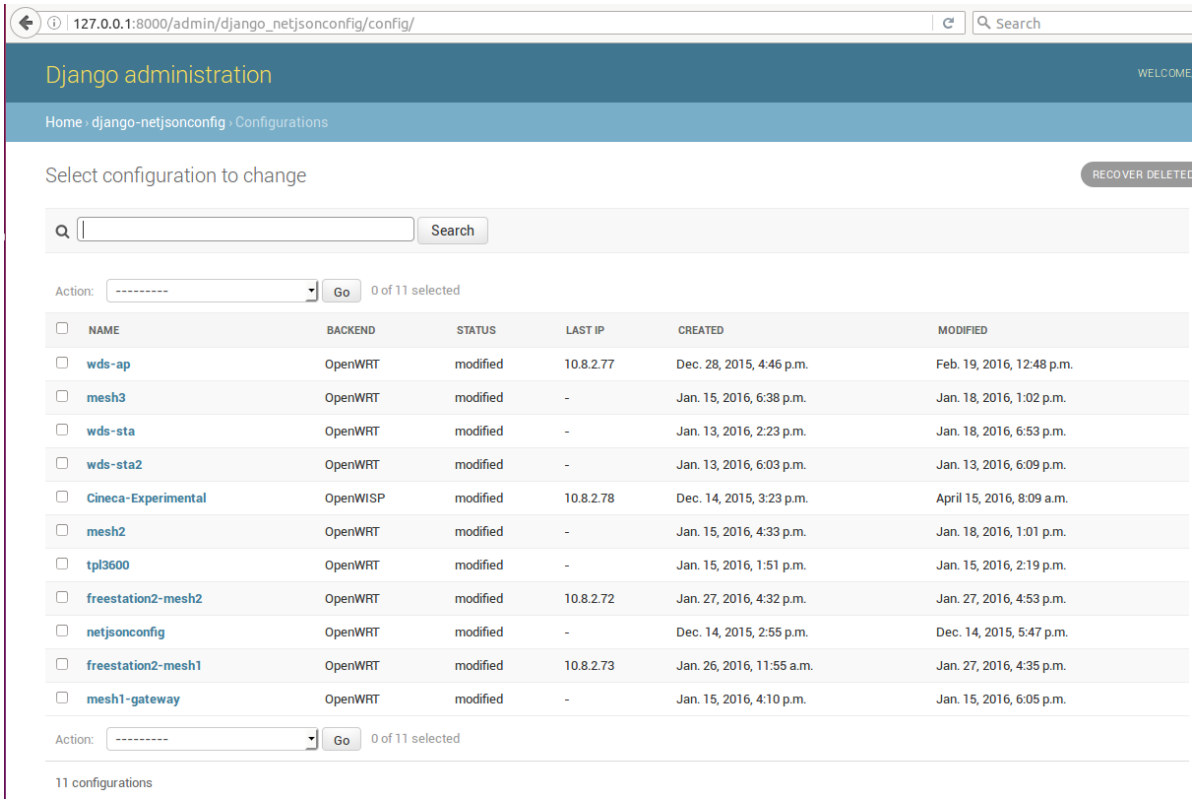
3. Η προσθήκη νέων επιλογών παραμετροποίησης για τις συσκευές απαιτεί αλλαγές όχι μόνο στον κώδικα αλλά και στη βάση δεδομένων του OWM.

4. Ο OpenWISP Manager είναι μια μονολιθική εφαρμογή. Δε χρησιμοποιεί διαφορετικά αρχεία κώδικα για τις δυνατότητές της, με αποτέλεσμα ο κώδικας να έχει μεγάλη πολυπλοκότητα και να μην είναι εύκολο να διατηρηθεί από τους προγραμματιστές.

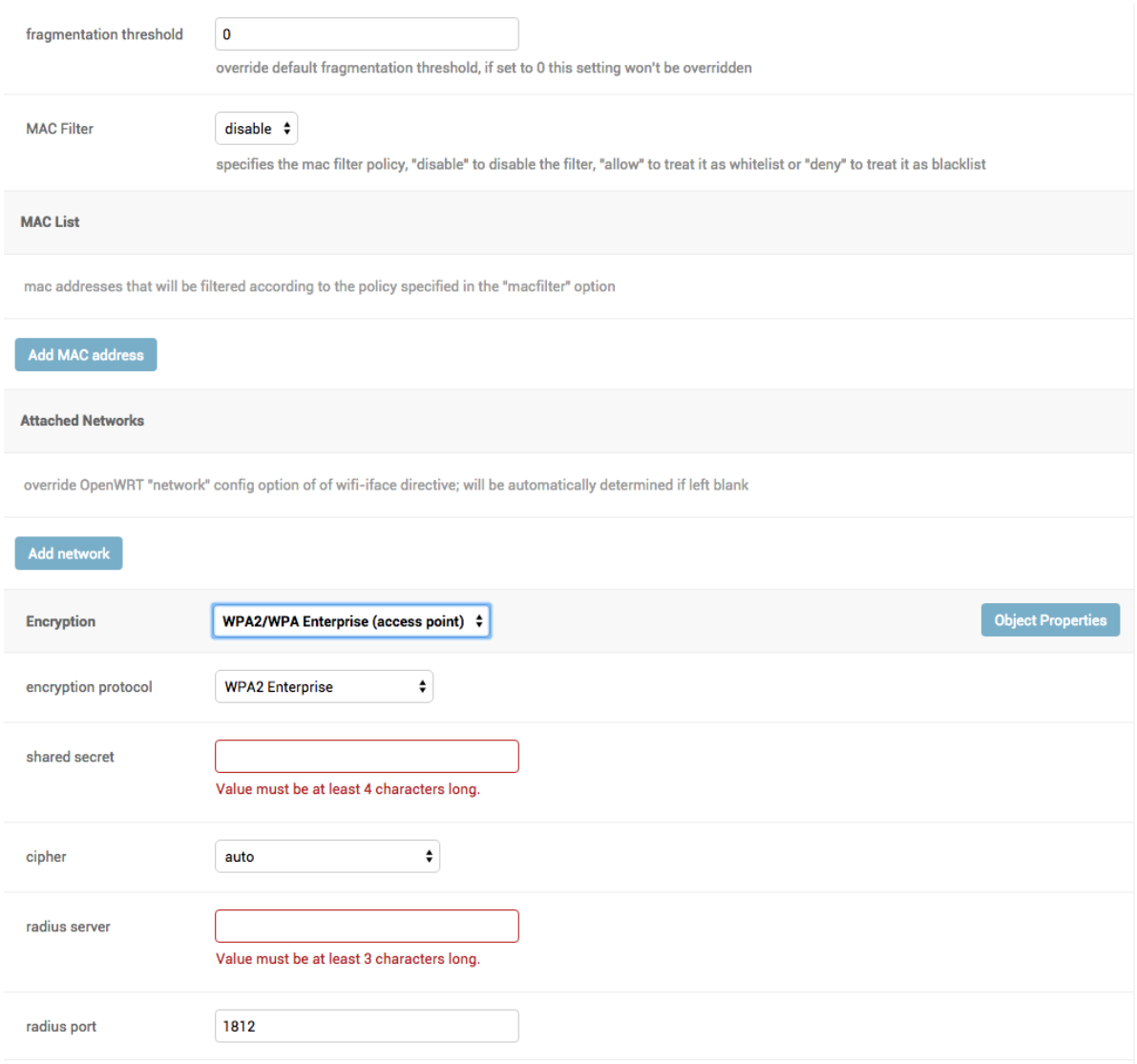
Για τους λόγους αυτούς, ξεκίνησε το 2016 η ανάπτυξη μιας νέας έκδοσης του OpenWISP με στόχο την άρση αυτών των περιορισμών ώστε το η πλατφόρμα του OpenWISP να μπορεί να χρησιμοποιηθεί σε διαφορετικά πλαίσια (πέρα από τα δημοτικά δίκτυα) και σε συνδυασμό με διαφορετικές τεχνολογίες, όπως για παράδειγμα συσκευές με λειτουργικό σύστημα AirOS [4.4] ή Raspbian [4.5] αντί για OpenWrt.

Η νέα έκδοση του OpenWISP αποσκοπεί στην αντικατάσταση του OpenWISP Manager και OpenWISP Firmware από την εφαρμογή openwisp-controller και το πακέτο openwisp-config αντίστοιχα. Το νέο λογισμικό έχει σχεδιαστεί έτσι ώστε να είναι ευέλικτο, επαναχρησιμοποιήσιμο, δομοστοιχειωτό και εύκολο στην εγκατάσταση και χρήση [4.6].

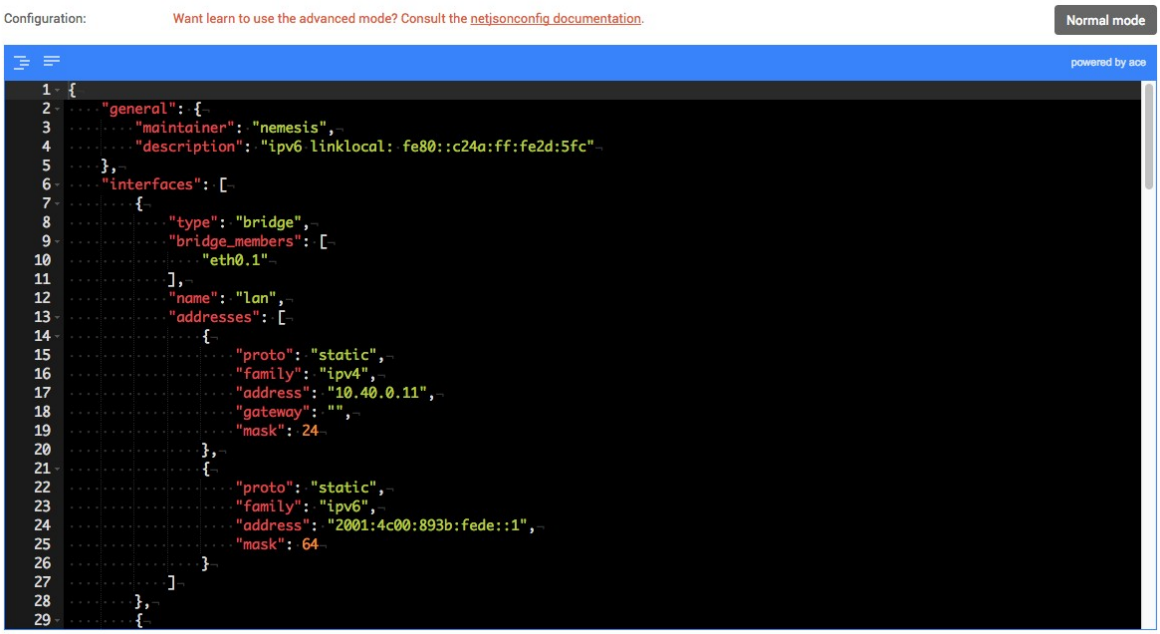
4.2.1 OpenWISP2 Controller



Το openwisp-controller είναι μια εφαρμογή ιστού αποτελούμενη από βιβλιοθήκες και υπό-εφαρμογές γραμμένες στη γλώσσα Python χρησιμοποιώντας το πλαίσιο ανάπτυξης λογισμικού Django. Επιτρέπει την κεντρική διαχείριση διάρθρωσης ενσωματωμένων δικτυακών συσκευών μέσω μιας διεπαφής ιστού. Η διεπαφή προσφέρει ένα GUI με μορφή φόρμας στην οποία ο χρήστης επιλέγει τις ρυθμίσεις που επιθυμεί. Η φόρμα, αφού συμπληρωθεί, μετατρέπεται σε σήμανση JSON συμβατή με το μορφότυπο NetJSON. Έπειτα, με χρήση της εφαρμογής netjsonconfig, οι ρυθμίσεις σε μορφή JSON μεταφράζονται σε αρχεία παραμετροποίησης του λειτουργικού συστήματος της συσκευής. Οι συσκευές με τη σειρά τους, επικοινωνούν περιοδικά με τον εξυπηρετητή που εκτελεί το openwisp-controller και στην περίπτωση που ο χρήστης έχει αλλάξει τις ρυθμίσεις τους, κατεβάζουν τις νέες ρυθμίσεις από τον εξυπηρετητή και τις εφαρμόζουν.



Στην περίπτωση που κάποιος έμπειρος χρήστης το επιθυμεί, μπορεί αντί να συμπληρώσει τη φόρμα των ρυθμίσεων, να γράψει απευθείας τις ρυθμίσεις της συσκευής σε μορφή JSON. Η διεπαφή προσφέρει εργαλεία για την επικύρωση του κειμένου JSON και την επιβεβαίωση της συμβατότητάς του με το μορφότυπο NetJSON.



Για μεγαλύτερη ευκολία στη διαχείριση των συσκευών, πέρα από την αποθήκευση των ρυθμίσεων συγκεκριμένων συσκευών, η εφαρμογή επιτρέπει και την αποθήκευση ρυθμίσεων για πρότυπα (templates) συσκευών τα οποία ορίζει ο χρήστης και να τα εφαρμόσει σε όποιες συσκευές επιθυμεί. Ο χρήστης μπορεί να εφαρμόσει πολλαπλά πρότυπα σε μια συσκευή ή ακόμα και να ορίσει πως κάποια πρότυπα πρέπει να εφαρμόζονται αυτόματα σε όλες τις νέες συσκευές. Για παράδειγμα, ένας χρήστης μπορεί να δημιουργήσει ένα πρότυπο που περιέχει τις απαραίτητες ρυθμίσεις για τη σύνδεση μιας συσκευής σε κάποιο VPN. Ύστερα, εάν θέλει να αλλάξει τις ρυθμίσεις για το VPN, θα χρειαστεί να επεξεργαστεί μόνο αυτό το πρότυπο και οι αλλαγές θα εφαρμοστούν σε όλες τις συσκευές που κληρονομούν από αυτό το πρότυπο.

Το openwisp-controller χρησιμοποιεί τη βιβλιοθήκη netjsonconfig, η οποία μετατρέπει αντικείμενα NetJSON σε αρχεία παραμετροποίησης.

4.2.2. NetJSON

Το NetJSON [4.7][4.8] είναι ένα μορφότυπο ανταλλαγής δεδομένων (Data Interchange Format) βασισμένο στη σήμανση JavaScript Object Notation (JSON) που επιτρέπει την περιγραφή δεδομένων για τα επίπεδα 2 (επίπεδο ζεύξης δεδομένων) και 3 (επίπεδο δικτύου) ενός δικτύου. Ορίζει μερικούς τύπους αντικειμένων JSON και τον τρόπο με τον οποίο συνδυάζονται για να αναπαραστήσουν τα βασικά δομικά στοιχεία που συνθέτουν ένα δίκτυο υπολογιστών:

- τη δικτυακή παραμετροποίηση των συσκευών (device network configuration)
- τα δεδομένα παρακολούθησης (monitoring data)
- τις πληροφορίες δρομολόγησης (routing information)
- την τοπολογία του δικτύου (network topology)

Οι έννοιες του NetJSON προήλθαν από υπάρχοντα έργα ανοικτού κώδικα του τομέα της δικτύωσης, όπως:

- τη γλώσσα Community Networking Markup Language (CNML)
- εργαλεία παρακολούθησης δικτύων όπως το Nagios
- πρωτόκολλα δυναμικής δρομολόγησης όπως τα OLSRd, Babel, Batman-adv και BMX
- διανομές Linux για δικτυακές συσκευές όπως το OpenWrt

Το NetJSON ορίζει 4 τύπους αντικειμένων:

1. Αντικείμενα τύπου `DeviceConfiguration`

Ένα αντικείμενο `DeviceConfiguration` περιγράφει τη δικτυακή διάρθρωση μιας συσκευής όπως ένας δρομολογητής, ένας εξυπηρετητής ή οποιαδήποτε άλλη συσκευή που έχει τη δυνατότητα να στέλνει πακέτα μέσω ενός δικτύου υπολογιστών.

2. Αντικείμενα τύπου `DeviceMonitoring`

Ένα αντικείμενο `DeviceMonitoring` περιγράφει την κατάσταση μιας δικτυακής συσκευής με στατιστικά όπως το χρόνο λειτουργίας και το ποσοστό χρήσης του επεξεργαστή.

3. Αντικείμενα τύπου `NetworkRoutes`

Ένα αντικείμενο `NetworkRoutes` περιγράφει μια λίστα διαδρομών που είναι εγκατεστημένες σε ένα πίνακα δρομολόγησης.

4. Αντικείμενα τύπου `NetworkGraph`

Ένα αντικείμενο `NetworkGraph` περιγράφει την τοπολογία ενός δικτύου και αποτελείται από μια λίστα κόμβων και μια λίστα ζεύξεων μεταξύ των κόμβων.

5. Αντικείμενα τύπου `NetworkCollection`

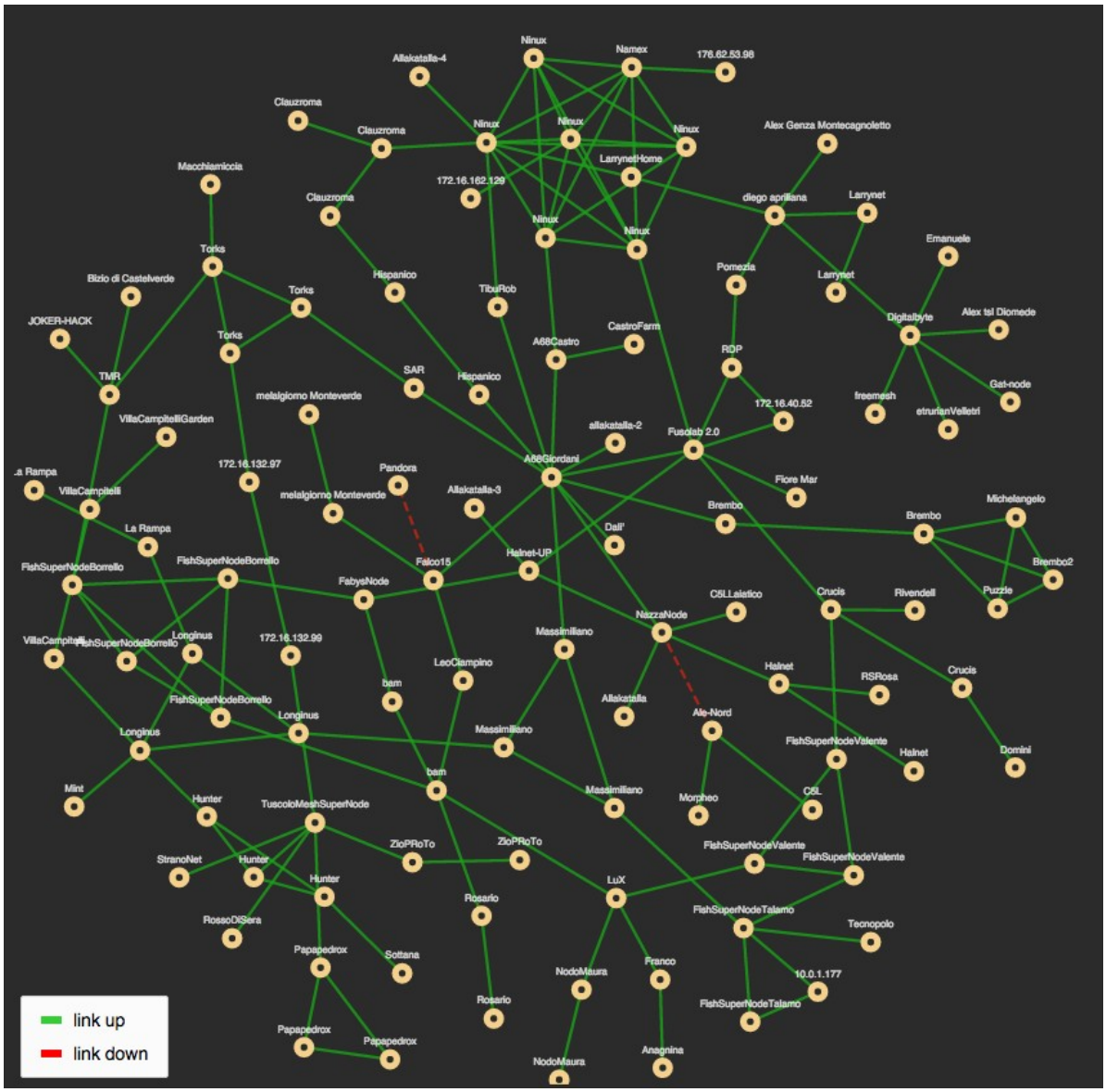
Ένα αντικείμενο `NetworkCollection` χρησιμοποιείται για την ομαδοποίηση πολλών αντικειμένων NetJSON σε ένα.

Περιγραφή της διάρθρωσης μιας συσκευής σε NetJSON:

```
{
    "type": "DeviceConfiguration",
    "general": {
        "hostname": "RouterA"
    },
    "interfaces": [
        {
            "name": "lo0",
            "type": "ethernet",
            "addresses": [
                {
                    "address": "127.0.0.1",
                    "mask": 8,
                    "proto": "static",
                    "family": "ipv4"
                }
            ]
        },
        {
            "name": "eth0",
            "type": "ethernet",
            "addresses": [
                {
                    "address": "192.168.1.1",
                    "mask": 24,
                    "proto": "static",
                    "family": "ipv4"
                }
            ]
        }
    ]
}
```

Το προηγούμενο παράδειγμα περιγράφει μια συσκευή με όνομα RouterA η οποία έχει δύο δικτυακές διεπαφές: την τυπική διεπαφή loopback και μια διεπαφή ethernet με όνομα eth0 η οποία έχει μια στατικά εκχωρημένη διεύθυνση IPv4 192.168.1.1 με μάσκα 24 bits (σε σήμανση CIDR).

Το NetJSON έχει βρει χρήση σε εφαρμογές που ασχολούνται με τη διαχείριση διάρθρωσης (στο OpenWISP2), τη δρομολόγηση [4.9], την οπτικοποίηση τοπολογίας δικτύων [4.10] και την παρακολούθηση δικτύων [4.11].



4.2.3. netjsonconfig

Το netjsonconfig [4.12] είναι μια βιβλιοθήκη Python η οποία υλοποιεί το μορφότυπο NetJSON και μετατρέπει αντικείμενα `DeviceConfiguration` σε πραγματικές διαρθρώσεις δρομολογητή οι οποίες μπορούν να εφαρμοστούν σε δικτυακές συσκευές με συγκεκριμένο λειτουργικό σύστημα (αρχικά OpenWrt, αλλά στην πορεία προστέθηκε υποστήριξη για AirOS, Raspbian κ.α.).

Καθώς τα αντικείμενα που επεξεργάζεται το netjsonconfig είναι όλα του ίδιου τύπου `DeviceConfiguration`, μπορούμε να παραβλέψουμε το πεδίο type από τα αντικείμενά μας, τα οποία ονομάζονται επίσης και λεξικά διάρθρωσης (configuration dictionaries). Ακολουθεί ένα παράδειγμα λεξικού διάρθρωσης που μπορεί να επεξεργαστεί το netjsonconfig:

```
{
    "interfaces": [
        {
            "name": "eth0.1",
            "type": "ethernet",
            "addresses": [
                {
                    "address": "192.168.1.2",
                    "gateway": "192.168.1.1",
                    "mask": 24,
                    "proto": "static",
                    "family": "ipv4"
                },
                {
                    "address": "192.168.2.1",
                    "mask": 24,
                    "proto": "static",
                    "family": "ipv4"
                },
                {
                    "address": "fd87::2",
                    "gateway": "fd87::1",
                    "mask": 64,
                    "proto": "static",
                    "family": "ipv6"
                }
            ]
        }
    ]
}
```

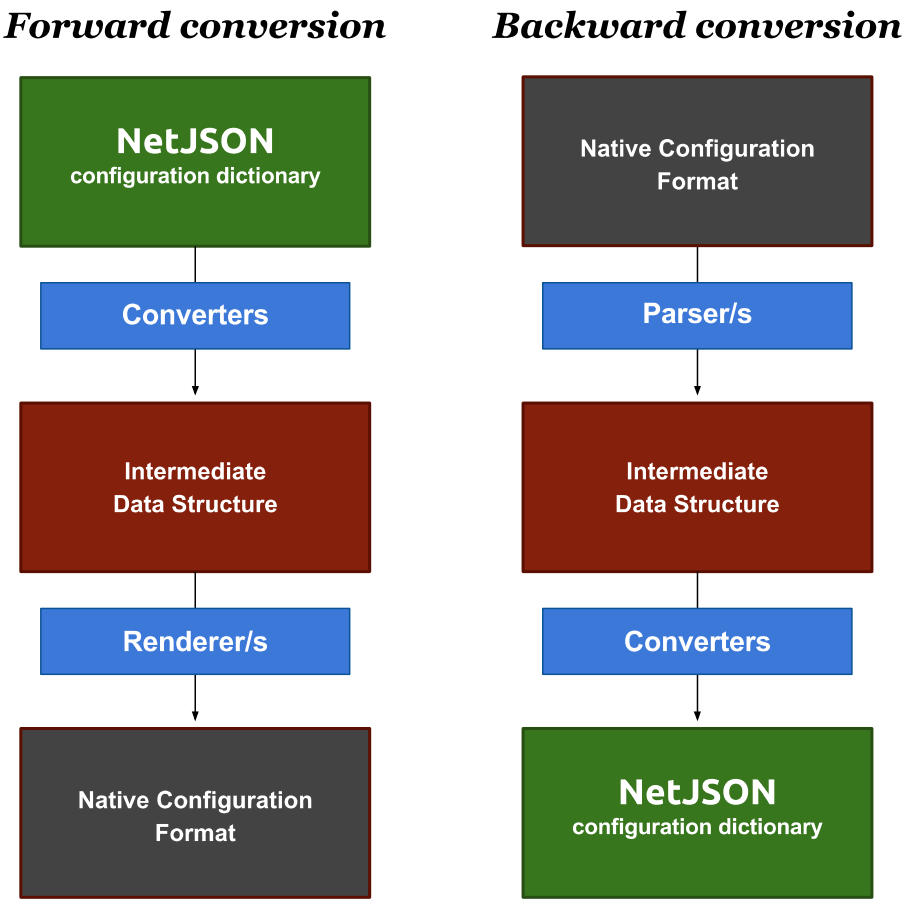
Αυτό το παράδειγμα μπορεί να μετατραπεί στην παρακάτω διάρθρωση UCI για συστήματα OpenWrt:

```
package network

config interface 'eth0_1'
        option gateway '192.168.1.1'
        option ifname 'eth0.1'
        option ip6addr 'fd87::2/64'
        option ip6gw 'fd87::1'
        list ipaddr '192.168.1.2/24'
        list ipaddr '192.168.2.1/24'
        option proto 'static'
```

Για τη μετατροπή των λεξικών διάρθρωσης στο μορφότυπο που χρησιμοποιεί εγγενώς ο δρομολογητής και αντίστροφα, το netjsonconfig χρησιμοποιεί μια κλάση python που ονομάζεται backend. Κάθε υποστηριζόμενο λειτουργικό σύστημα ή firmware έχει το δικό του backend και νέα backends τα οποία υποστηρίζουν άλλα λειτουργικά συστήματα μπορούν να γραφούν από τρίτους. Κάθε backend πρέπει να υλοποιεί τους δικούς του parsers, renderers και converters.

Οι converters μετατρέπουν αντικείμενα NetJSON στην ενδιάμεση δομή δεδομένων και αντίστροφα. Οι renderers μετατρέπουν την ενδιάμεση δομή δεδομένων στο εγγενές μορφότυπο. Οι parsers εκτελούν την αντίστροφη πράξη από τους renderers: διαβάζουν το εγγενές μορφότυπο του δρομολογητή και χτίζουν την ενδιάμεση δομή δεδομένων.



Κάθε backend χρησιμοποιεί το δικό του σχήμα JSON, το οποίο προέρχεται από ένα κοινό γονέα. Καθώς διαφορετικά backend μπορεί να υποστηρίζουν διαφορετικές λειτουργίες, το καθένα μπορεί να επεκτείνει το σχήμα που χρησιμοποιεί προσθέτοντας τους δικούς του ορισμούς.

4.2.4. OpenWISP2 Firmware

Ο νέος πράκτορας διάρθρωσης (configuration agent) του OpenWISP για μηχανήματα OpenWrt αποτελείται
από δύο πακέτα:

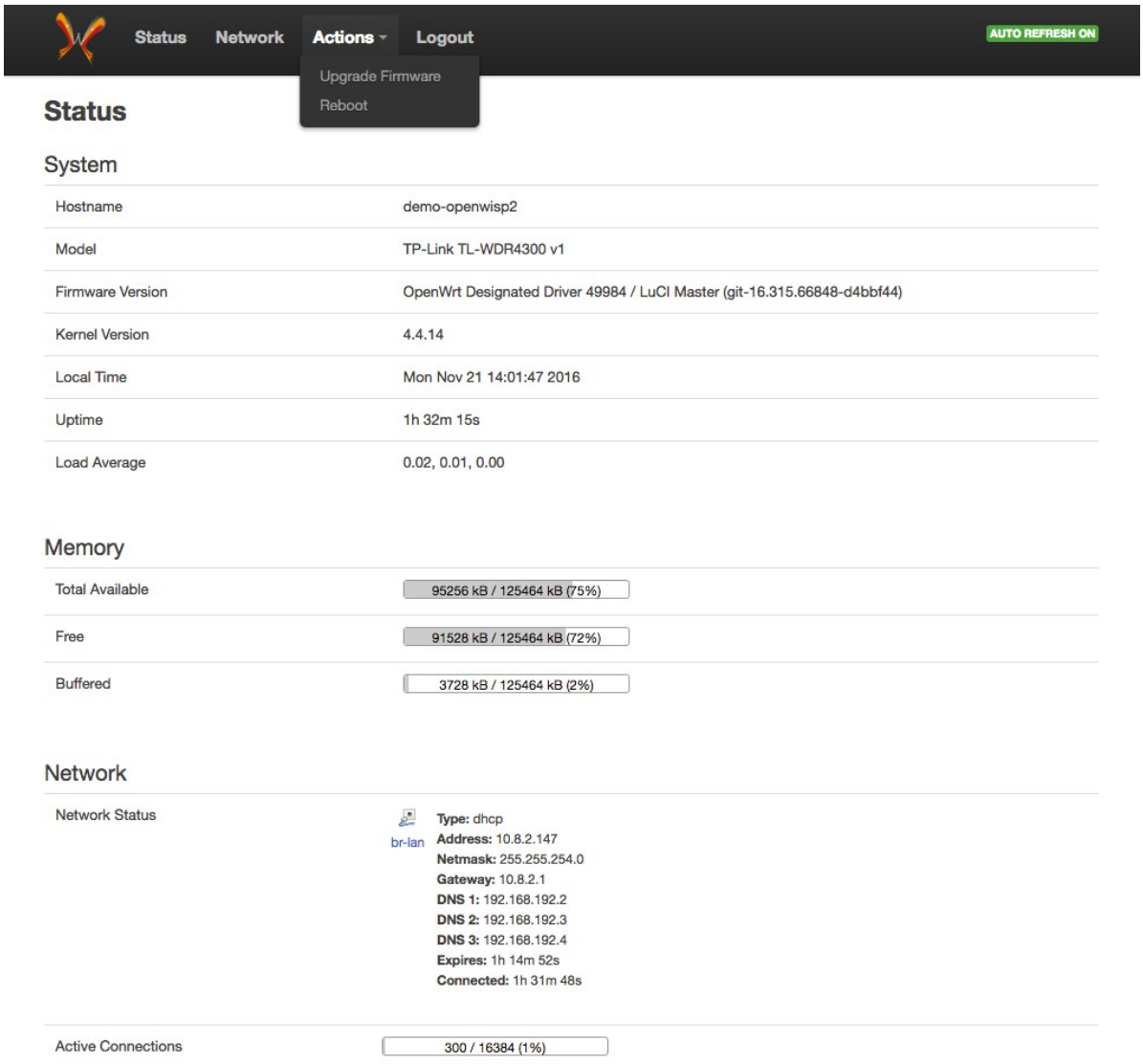
1. Το πακέτο openwisp-config

Το πακέτο αυτό υλοποιεί τις βασικές λειτουργίες του agent. Κατά την εκκίνηση, εάν απουσιάζουν οι ρυθμίσεις uuid και key, δοκιμάζει να πραγματοποιήσει εγγραφή στον κεντρικό εξυπηρετητή. Όταν ολοκληρωθεί η εγγραφή, ο agent θα θέσει τις ρυθμίσεις uuid και key στο αρχείο /etc/config/openwisp. Για να πραγματοποιηθεί η αυτόματη εγγραφή πρέπει να έχουμε θέσει σωστή τιμή στη ρύθμιση shared_secret.

Ο agent περιοδικά (κάθε 2 λεπτά από προεπιλογή) επικοινωνεί με τον εξυπηρετητή και εξετάζει εάν έχουν αλλάξει τα αρχεία διάρθρωσης. Εάν πράγματι έχουν αλλάξει, τότε κατεβάζει τα πιο πρόσφατα αρχεία από τον εξυπηρετητή, δημιουργεί αντίγραφο ασφαλείας της τρέχουσας διάρθρωσης και προσπαθεί να συγχωνεύσει τα νέα αρχεία διάρθρωσης με τα υπάρχοντα (εκτός αν έχουμε θέσει τιμή 0 στην επιλογή merge_config στο αρχείο /etc/config/openwisp). Ύστερα, εκτελεί ορισμένες δοκιμές ώστε να επιβεβαιώσει την καλή λειτουργία του μηχανήματος. Εάν οι δοκιμές επιτύχουν, ο agent ενημερώνει τον εξυπηρετητή πως χρησιμοποιεί τη νέα διάρθρωση και διαγράφει το αντίγραφο ασφαλείας της προηγούμενης. Διαφορετικά επαναφέρει την παλιά διάρθρωση από το αντίγραφο ασφαλείας και ενημερώνει τον κεντρικό εξυπηρετητή για την αποτυχία.

2. Το πακέτο luci-openwisp

Το πακέτο αυτό προσφέρει μια περιορισμένη διεπαφή ιστού μέσω της οποίας οι χρήστες μπορούν να εισάγουν μόνο τις απολύτως απαραίτητες ρυθμίσεις για να επικοινωνήσει η συσκευή με τον κεντρικό εξυπηρετητή. Ο λόγος που το πακέτο αυτό περιορίζει τις δυνατότητες της προεπιλεγμένης διεπαφής ιστού (luci) είναι ότι αποσκοπεί να προτρέψει τους χρήστες να χρησιμοποιήσουν τον κεντρικό εξυπηρετητή για τη διάρθρωση των συσκευών τους και να μην εισάγουν ρυθμίσεις τοπικά.



Τα πακέτα αυτά μπορούν να εγκατασταθούν μέσω του διαχειριστή πακέτων opkg σε μηχανήματα OpenWrt/LEDE και OWF (firmware της πρώτης έκδοσης του OpenWISP). Για να επικοινωνήσουν με τον κεντρικό εξυπηρετητή, συσκευές που χρησιμοποιούν άλλα λειτουργικά συστήματα ή firmware θα πρέπει να χρησιμοποιήσουν το δικό τους agent.

Κεφάλαιο 5

Επεκτείνοντας το OpenWISP

Δυστυχώς, η δεύτερη έκδοση του OpenWISP είναι ακόμα σε στάδιο ανάπτυξης και δεν υποστηρίζει όλες τις λειτουργίες της πρώτης. Επιπλέον, ορισμένες δυνατότητες τις οποίες χρειαζόμαστε λείπουν ακόμα και από την πρώτη έκδοση του OpenWISP. Επομένως, για να καταφέρουμε να χρησιμοποιήσουμε την πλατφόρμα αυτή για τη δική μας περίπτωση χρήσης, θα πρέπει να επεκτείνουμε τον openwisp-controller ώστε να προσθέσουμε τις δυνατότητες που επιθυμούμε.

5.1 Διαχειριστικές ομάδες

Στο δίκτυό μας έχουμε πολλούς διαχειριστές σημείων πρόσβασης, οι οποίοι είναι χωρισμένοι σε ομάδες. Κάθε διαχειριστική ομάδα είναι υπεύθυνη για την υποστήριξη ενός υποσυνόλου των σημείων πρόσβασης του δικτύου. Το σύστημά μας δεν πρέπει να επιτρέπει στους χρήστες ούτε να παρακολουθούν ούτε να αλλάζουν τη διάρθρωση των συσκευών που δεν ανήκουν στη δική τους διαχειριστική ομάδα.

Για να υλοποιήσουμε τη δυνατότητα εξυπηρέτησης πολλαπλών ομάδων από έναν εξυπηρετητή (multi-tenancy) θα πρέπει να αποφασίσουμε πρώτα πώς θα επιτύχουμε τις λειτουργίες της αυθεντικοποίησης (authentication) και εξουσιοδότησης (authorization) στο σύστημά μας. Στην περίπτωσή μας, όλες οι πληροφορίες για τους διαχειριστές, συμπεριλαμβανομένης και της διαχειριστικής ομάδας στην οποία ανήκουν, βρίσκονται συγκεντρωμένες σε έναν εξυπηρετητή LDAP. Ο εξυπηρετητής LDAP είναι συνδεδεμένος με έναν εξυπηρετητή CAS τον οποίο μπορούμε να χρησιμοποιήσουμε για την αυθεντικοποίηση των χρηστών.

5.1.1 Μοντελοποίηση

Δικαιώματα

Για την προστασία της ιδιοκτησίας κάθε διαχειριστικής ομάδας θα χρησιμοποιήσουμε ένα απλό σύστημα δικαιωμάτων. Στο σύστημά μας υπάρχουν 3 τύποι δικαιωμάτων: δικαίωμα δημιουργίας, επεξεργασίας και διαγραφής. Τα δικαιώματα αυτά ανατίθενται σε διαχειριστικές ομάδες αυτόματα σε διαχειριστικές ομάδες, αλλά μπορούν να ανατεθούν χειροκίνητα και σε μεμονωμένους χρήστες. Για τη δημιουργία ενός αντικειμένου απαιτείται είτε ο χρήστης είτε η ομάδα στην οποία ανήκει να έχει το δικαίωμα δημιουργίας για αντικείμενα αυτού του τύπου (table-level permission). Αντίθετα, για την επεξεργασία ή τη διαγραφή ενός αντικειμένου απαιτείται το αντίστοιχο δικαίωμα για το συγκεκριμένο μόνο αντικείμενο (row-level permission). Η έννοια της ιδιοκτησίας ενός αντικειμένου συνεπάγεται την ύπαρξη δικαιώματος επεξεργασίας και διαγραφής του αντικειμένου αυτού.

Διαρθρώσεις και πρότυπα

Ο openwisp-controller ορίζει δύο τύπους αντικειμένων τα οποία θα χρησιμοποιήσουμε: διαρθρώσεις (configurations) και πρότυπα (templates). Ένα αντικείμενο τύπου διάρθρωσης περιέχει πληροφορίες σχετικές με τη διάρθρωση μίας συσκευής, όπως το λεξικό που περιγράφει την επιθυμητή διάρθρωση της συσκευής (όπως περιγράφεται στο κεφάλαιο 4), το backend που αντιστοιχεί στο λειτουργικό σύστημα της συσκευής, την τελευταία διεύθυνση IP της συσκευής που κατέβασε και δοκίμασε να εφαρμόσει τη διάρθρωση και την κατάσταση της επιθυμητής διάρθρωσης ("running" εάν είναι ήδη σε εφαρμογή στη συσκευή, "modified" εάν δεν έχει ακόμα εφαρμοστεί ή "error" εάν η εφαρμογή απέτυχε και η συσκευή επανέφερε την τελευταία διάρθρωση που πέρασε επιτυχώς τις δοκιμές). Ένα αντικείμενο τύπου προτύπου περιέχει μόνο ένα λεξικό διάρθρωσης το οποίο έχουμε σκοπό να εφαρμόσουμε σε πάνω από μια συσκευή.

Οι διαχειριστικές ομάδες δεν έχουν δικαίωμα δημιουργίας νέων αντικειμένων διάρθρωσης στο σύστημα. Τα αντικείμενα αυτά δημιουργούνται αυτόματα κατά την εγγραφή ενός agent στον εξυπηρετητή μας. Οι διαχειριστές μπορούν έπειτα να δημιουργήσουν μια νέα συσκευή στο σύστημα και να τη συσχετίσουν με τη διάρθρωση που δημιούργησε ο agent. Τότε, η διαχειριστική τους ομάδα αποκτά δικαιώματα επεξεργασίας και διαγραφής αυτής της διάρθρωσης. Όλες οι ομάδες έχουν δικαίωμα δημιουργίας νέων προτύπων και δικαιώματα επεξεργασίας και διαγραφής των προτύπων που έχουν δημιουργήσει.

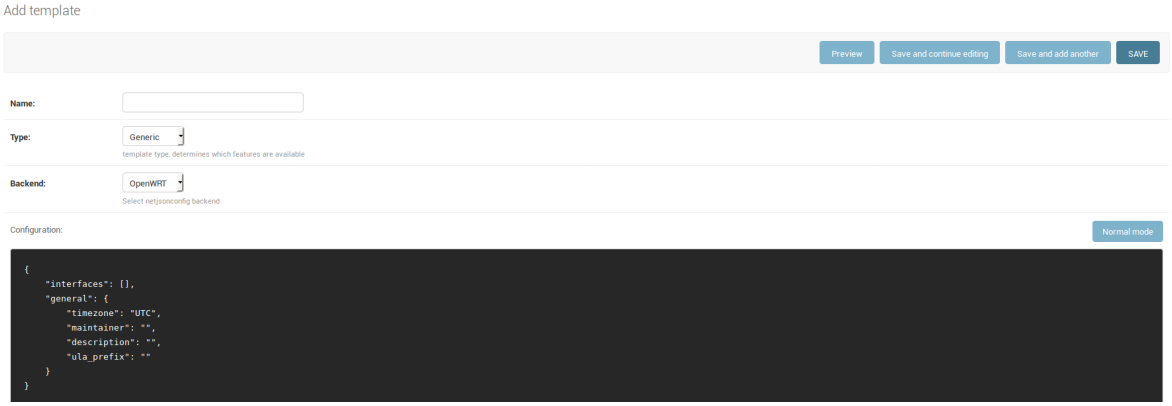
Συσκευές

Ένα αντικείμενο τύπου συσκευής περιέχει πληροφορίες για τη συσκευή που δεν είναι σχετικές με τη διάρθρωσή της, όπως τον κατασκευαστή της, το μοντέλο της και την τοποθεσία της. Κάθε αντικείμενο τύπου συσκευής είναι συνδεδεμένο μέσω μιας σχέσης ένα-προς-ένα με ένα αντικείμενο διάρθρωσης. Οι διαχειριστικές ομάδες έχουν δικαίωμα δημιουργίας νέων συσκευών και δικαιώματα επεξεργασίας και διαγραφής των συσκευών που έχουν δημιουργήσει.

Χρήστες και διαχειριστικές ομάδες

Στο σύστημά μας κάθε διαχειριστής έχει ένα όνομα χρήστη και ανήκει σε μία και μοναδική διαχειριστική ομάδα. Κατά την είσοδο ενός χρήστη στο σύστημά μας, λαμβάνουμε από τον εξυπηρετητή CAS ένα χαρακτηριστικό (attribute) το οποίο αντιστοιχεί στο αναγνωριστικό της ομάδας του. Η εφαρμογή μας ενημερώνεται για τη διαχειριστική ομάδα στην οποία ανήκει ο χρήστης σε κάθε είσοδό του. Κάθε διαχειριστική ομάδα έχει την ιδιοκτησία κάποιων συσκευών του δικτύου, των αντίστοιχων διαρθρώσεων τους που είναι αποθηκευμένες στο σύστημα και κάποιων προτύπων διάρθρωσης τα οποία έχουν δημιουργηθεί από μέλη της ομάδας.

Εντός μιας διαχειριστικής ομάδας, όλα τα μέλη έχουν τον ίδιο ρόλο και τα ίδια δικαιώματα πάνω στις συσκευές. Όλοι ο διαχειριστές μπορούν να επεξεργαστούν και να διαγράψουν τις συσκευές, τις διαρθρώσεις και τα πρότυπα της ομάδας. Επίσης, μπορούν να εισαγάγουν νέες συσκευές στο δίκτυο (οι οποίες τίθενται αυτόματα υπό την ιδιοκτησία της ομάδας του διαχειριστή) και να δημιουργήσουν νέα πρότυπα διάρθρωσης.



Στο σύστημα επίσης μπορεί να υπάρχουν και υπέρ-διαχειριστές. Ένας υπέρ-διαχειριστής έχει όλα τα δικαιώματα επάνω σε όλες τις συσκευές, τις διαρθρώσεις και τα πρότυπα του δικτύου, ακόμα και εκείνες που δεν ανήκουν στη διαχειριστική του ομάδα. Επίσης, οι υπέρ-διαχειριστές μπορούν να αλλάξουν τα δικαιώματα ενός διαχειριστή ή μιας διαχειριστικής ομάδας ακόμα και να μετατρέψουν έναν άλλο χρήστη σε υπέρ-διαχειριστή. Για παράδειγμα, ένας διαχειριστής μπορεί να αφαιρέσει το δικαίωμα δημιουργίας συσκευών από μία ομάδα, να αλλάξει την ομάδα στην οποία ανήκει μια συσκευή, να επιτρέψει σε δύο διαχειριστικές ομάδες να επεξεργάζονται το ίδιο πρότυπο ή και να δώσει την αποκλειστική ιδιοκτησία μιας συσκευής σε ένα μεμονωμένο διαχειριστή και όχι στην ομάδα του. Ένας υπέρ-διαχειριστής δεν μπορεί να αλλάξει την ομάδα στην οποία ανήκει ένας άλλος διαχειριστής. Αυτό απαιτεί αλλαγή στο αντίστοιχο πεδίο της εγγραφής LDAP του διαχειριστή.

5.1.2 Υλοποίηση

Δικαιώματα ανά αντικείμενο

Το πλαίσιο ανάπτυξης εφαρμογών Django χρησιμοποιεί ένα απλό σύστημα δικαιωμάτων για την ανάθεση ρόλων στους χρήστες. Για κάθε μοντέλο της εφαρμογής μας, το Django δημιουργεί 3 τύπους δικαιωμάτων, τα οποία μπορούν να ανατεθούν σε χρήστες ή ομάδες χρηστών.

1. `add_<όνομα μοντέλου>` : Επιτρέπει τη δημιουργία νέων αντικειμένων αυτού του τύπου
2. `change_<όνομα μοντέλου>` : Επιτρέπει την επεξεργασία των αντικειμένων αυτού του τύπου
3. `delete_<όνομα μοντέλου>` : Επιτρέπει τη διαγραφή των αντικειμένων αυτού του τύπου

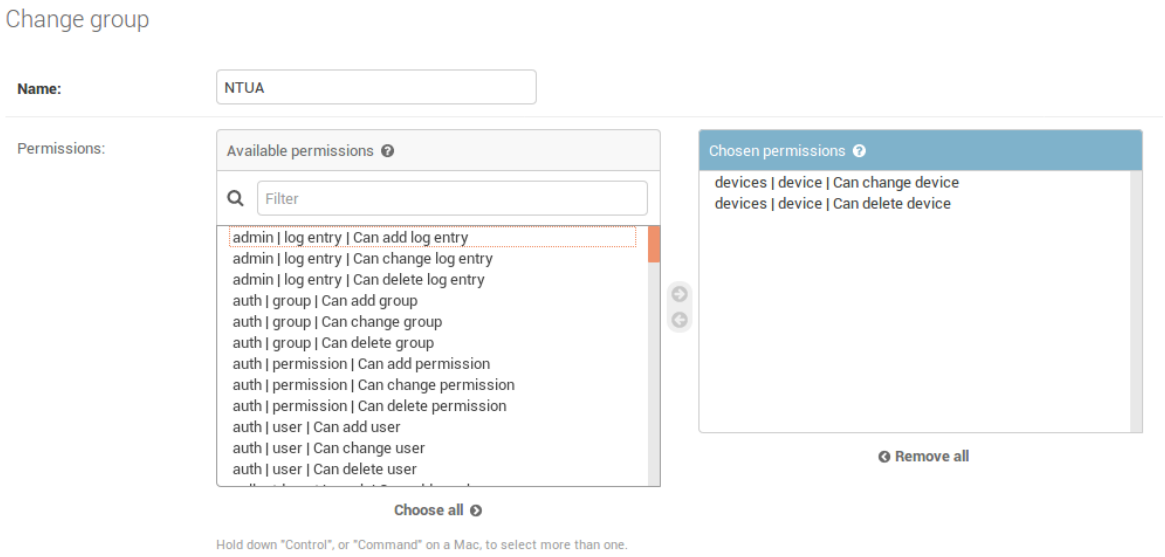
Όπως βλέπουμε, ενώ το Django μας προσφέρει ένα σύστημα δικαιωμάτων, δεν καλύπτει όλες τις ανάγκες μας επειδή υποστηρίζει μόνο δικαιώματα σε επίπεδο τύπου αντικειμένου και όχι δικαιώματα ανά αντικείμενο. Για την υλοποίηση δικαιωμάτων σε επίπεδο αντικειμένου θα χρησιμοποιήσουμε το πακέτο django-guardian [1]. Το πακέτο αυτό επιτρέπει την ανάθεση δικαιωμάτων ανά αντικείμενο στους χρήστες μας και επίσης παρέχει ένα φιλικότερο τρόπο να διαχειριζόμαστε τα δικαιώματα, τόσο προγραμματιστικά όσο και μέσω του γραφικού περιβάλλοντος της εφαρμογής μας. Επίσης, θα χρησιμοποιήσουμε το πακέτο `rules` για τις κλάσης ελέγχου πρόσβασης που παρέχει. Για να χρησιμοποιήσουμε τα πακέτα αυτά, πρέπει να προσθέσουμε τα αντίστοιχα backends (`ObjectPermissionBackend`) στα authentication backends της εφαρμογής μας.

Η ανάθεση των δικαιωμάτων γίνεται μέσω της συνάρτησης `guardian.shortcuts.assign_perm()`, η οποία δέχεται τρία ορίσματα: τον τύπο του δικαιώματος (έναν από τους 3 που μας παρέχει το Django ή κάποιο που έχουμε δημιουργήσει οι ίδιοι), ένα αντικείμενο χρήστη ή ομάδας στο οποίο αναθέτουμε το δικαίωμα και το αντικείμενο στο οποίο αναφέρεται το δικαίωμα. Το τρίτο όρισμα παραλείπεται στην περίπτωση που θέλουμε να αναθέσουμε δικαιώματα για όλα τα αντικείμενα του ίδιου τύπου. Όπως είναι προφανές, το δικαίωμα δημιουργίας αντικειμένων δεν αναφέρεται ποτέ σε ένα μεμονωμένο αντικείμενο, αλλά πάντα σε τύπο αντικειμένων.

Στην εφαρμογή μας θέλουμε όλοι οι χρήστες να μπορούν να εισάγουν νέες συσκευές και να δημιουργήσουν πρότυπα διάρθρωσης για την ομάδα τους. Επομένως, κατά την εισαγωγή ενός χρήστη στο σύστημα, του αναθέτουμε τα δικαιώματα της δημιουργίας συσκευών και προτύπων (`devices.add_device` και `django_netjsonconfig.add_template` αντίστοιχα). Κατά τη δημιουργία μιας νέας συσκευής ή προτύπου, αναθέτουμε στην ομάδα του δημιουργού τα δικαιώματα επεξεργασίας και διαγραφής του συγκεκριμένου αντικειμένου. Για τα αντικείμενα τύπου διάρθρωσης, τα οποία δημιουργούνται αυτόματα κατά την εγγραφή ενός πράκτορα στο σύστημα, θα πρέπει να χρησιμοποιήσουμε μια διαφορετική μέθοδο, καθώς δε γνωρίζουμε τον ιδιοκτήτη τους κατά τη δημιουργία τους.

Η πρώτη στιγμή που ανακαλύπτουμε των ιδιοκτήτη αυτών των αντικειμένων είναι κατά τη συσχέτισή τους με μία συσκευή. Επομένως, κατά τη συσχέτιση μιας συσκευής με ένα αντικείμενο διάρθρωσης, αναθέτουμε τα δικαιώματα επεξεργασίας και διαγραφής του αντικειμένου διάρθρωσης στον ιδιοκτήτη της συσκευής.

Πέρα από την αυτόματη ανάθεσή τους, η διαχείριση των δικαιωμάτων μπορεί να γίνει και χειροκίνητα από τους υπέρ-διαχειριστές. Ένας υπέρ-διαχειριστής μπορεί μέσω της εφαρμογής να προσθέσει ή να αφαιρέσει τα δικαιώματα που επιθυμεί από οποιονδήποτε χρήστη ή ομάδα. Για να προσθέσουμε στην εφαρμογή μας τη δυνατότητα διαχείρισης των δικαιωμάτων σε επίπεδο αντικειμένων, χρησιμοποιούμε τις κλάσεις `GuardedModelAdminMixin` και `ObjectPermissionsModelAdminMixin`.



Έχοντας ορίσει τον τρόπο ανάθεσης των δικαιωμάτων της εφαρμογής, μπορούμε τώρα να τα χρησιμοποιήσουμε για να περιορίσουμε την πρόσβαση των χρηστών μόνο στις πληροφορίες και τις ενέργειες για τις οποίες έχουν εξουσιοδότηση. Καταρχάς, σε όλες τις οθόνες της εφαρμογής μας απαιτούμε την αυθεντικοποίηση των χρηστών. Γι' αυτό το λόγο, όλες οι οθόνες κληρονομούν από την κλάση `LoginRequiredMixin`, η οποία απαγορεύει την είσοδο σε ανώνυμους χρήστες και τους ανακατευθύνει στον εξυπηρετητή CAS για αυθεντικοποίηση. Στη συνέχεια, οι οθόνες που εμφανίζουν τις συσκευές ή τα πρότυπα διάρθρωσης στα οποία έχει πρόσβαση ένας χρήστης κληρονομούν από την κλάση `PermissionListMixin`, η οποία λειτουργεί σαν φίλτρο το οποίο κρύβει τα αντικείμενα στα οποία δεν έχει πρόσβαση ο χρήστης. Τέλος, οι οθόνες επεξεργασίας ή διαγραφής αντικειμένων κληρονομούν από την κλάση `PermissionRequiredMixin`, η οποία επιτρέπει την πρόσβαση μόνο στους χρήστες που διαθέτουν το αντίστοιχο δικαίωμα.

Σύνδεση με τον εξυπηρετητή CAS

Για τη σύνδεση με τον εξυπηρετητή CAS θα χρησιμοποιήσουμε το πακέτο `django-cas-ng`. Στο αρχείο των ρυθμίσεων (settings.py) προσθέτουμε το `CASBackend` στα authentication backends και ορίζουμε τα εξής πεδία:

- `CAS_SERVER_URL`

Ορίζει τη διεύθυνση του εξυπηρετητή CAS μέσω του οποίου θα γίνεται η αυθεντικοποίηση των χρηστών

- `CAS_VERSION = '3'`

Ορίζει την έκδοση του πρωτοκόλλου CAS που θα χρησιμοποιήσουμε. Καθώς χρησιμοποιούμε attributes για την ομαδοποίηση των χρηστών,
θα πρέπει να χρησιμοποιήσουμε την έκδοση 3 του πρωτοκόλλου.

- `CAS_REDIRECT_URL = 'devices_home'`

Ορίζει την οθόνη στην οποία θα ανακατευθύνουμε τους χρήστες ύστερα από την επιτυχή τους είσοδο.

- `AFFILIATION_FIELD = "userPrincipalName"`

Ορίζει το όνομα του χαρακτηριστικού (CAS attribute) με βάση το οποίο θα ομαδοποιούμε τους χρήστες.
Θεωρούμε ότι δύο χρήστες ανήκουν στην ίδια ομάδα όταν έχουν την ίδια τιμή σε αυτό το χαρακτηριστικό.

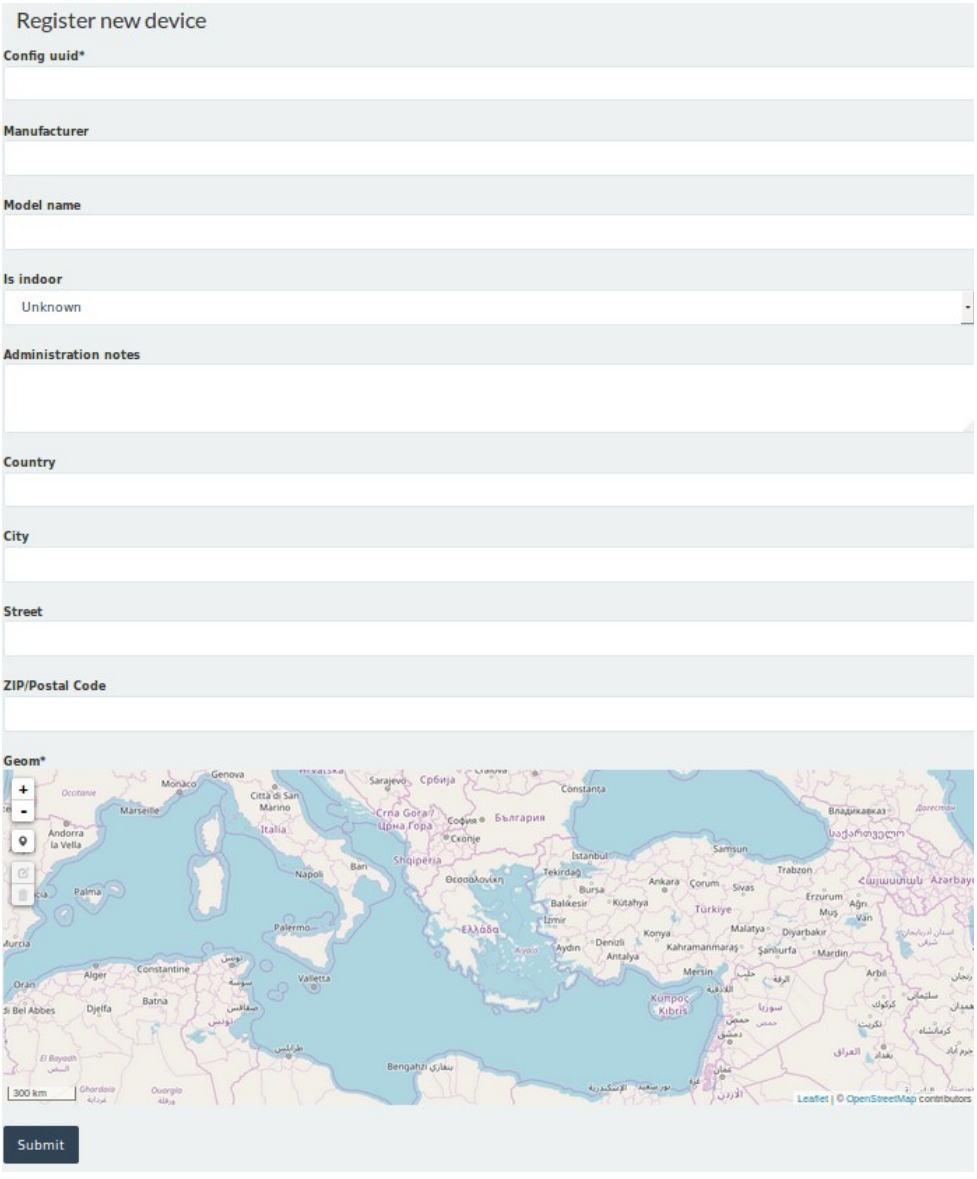
- `LOGIN_URL` και `LOGOUT_URL`

Τα πεδία αυτά ορίζουν τις οθόνες που θα χρησιμοποιεί η εφαρμογή μας για είσοδο και έξοδο των χρηστών.

Έχοντας ορίσει τις σχετικές με το CAS ρυθμίσεις, μένει μόνο η ομαδοποίηση των χρηστών. Μετά την επιτυχή είσοδο ενός χρήστη, λαμβάνουμε το σήμα `cas_user_authenticated`. Στο αρχείο signals.py, η συνάρτηση `cas_authentication_handler` καλείτε με ορίσματα τα χαρακτηριστικά που μας επιστρέφει ο εξυπηρετητής CAS. Αυτά χρησιμοποιούμε για τη δημιουργία του χρήστη και της ομάδας του (εφόσον δεν υπάρχει ήδη). Επίσης, μέσα στη συνάρτηση αυτή αναθέτουμε στο χρήστη τα δικαιώματά του.

Προσθήκη νέας συσκευής

Μετά την παραμετροποίηση και εκκίνηση του δαίμονα openwisp-config σε μία συσκευή OpenWrt, πραγματοποιείται αυτόματη εγγραφή της συσκευής στον εξυπηρετητή. Ο εξυπηρετητής δημιουργεί ένα νέο αντικείμενο τύπου `Config` και του αναθέτει ένα μοναδικό αναγνωριστικό και ένα κλειδί με το οποίο μπορούμε να κατεβάσουμε την επιθυμητή διάρθρωση. Ο εξυπηρετητής στέλνει αυτά τα δύο στοιχεία στον πράκτορα, ο οποίος τα αποθηκεύει στο αρχείο /etc/config/openwisp, στις μεταβλητές `uuid` και `key`. Αφού σημειώσουμε το `UUID`, συνδεόμαστε στον εξυπηρετητή και δημιουργούμε ένα νέο αντικείμενο τύπου `Device`. Στη φόρμα που εμφανίζεται συμπληρώνουμε τα στοιχεία της συσκευής. Για να συσχετίσουμε αυτό το αντικείμενο `Device` με το σωστό αντικείμενο `Config`, πρέπει στο πεδίο `Config` της φόρμας να εισάγουμε το `UUID` που σημειώσαμε προηγουμένως. Με την ολοκλήρωση της διαδικασίας, ο εξυπηρετητής αναθέτει στη διαχειριστική μας ομάδα την ιδιοκτησία τόσο του αντικειμένου `Device` που δημιουργήσαμε όσο και του αντικειμένου `Config` με το οποίο το συσχετίσαμε.



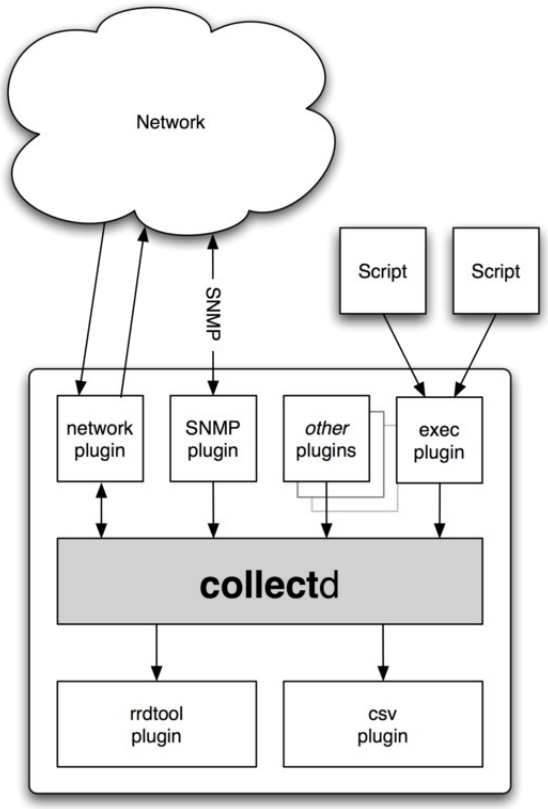
5.2 Παρακολούθηση δικτύου

Μία από τις δυνατότητες της πρώτης έκδοσης του OpenWISP η οποία δεν έχει ακόμα υλοποιηθεί για το νέο controller είναι η δυνατότητα της παρακολούθησης του δικτύου. Η λειτουργία αυτή είναι ιδιαίτερα χρήσιμη, καθώς μας επιτρέπει να εντοπίσουμε προβλήματα στο δίκτυο, να τα διορθώσουμε αλλάζοντας τη διάρθρωση των συσκευών μας και να επιβεβαιώσουμε την επίλυσή τους σε δευτερόλεπτα μέσω της εφαρμογής μας. Επομένως, είναι σημαντικό να προσθέσουμε τη δυνατότητα αυτή στον controller της δεύτερης έκδοσης.

Οι σχετικές με την παρακολούθηση του δικτύου λειτουργίες που θα προσθέσουμε θα είναι πολύ απλές και σε καμία περίπτωση δεν μπορούν να ανταγωνιστούν ένα πλήρες και ώριμο εργαλείο όπως το Zabbix. Για το λόγο αυτό, αυτή η λειτουργία της εφαρμογής μας είναι προαιρετική και μπορεί να αντικατασταθεί με οποιοδήποτε άλλο εργαλείο επιθυμούν οι διαχειριστές των συσκευών.

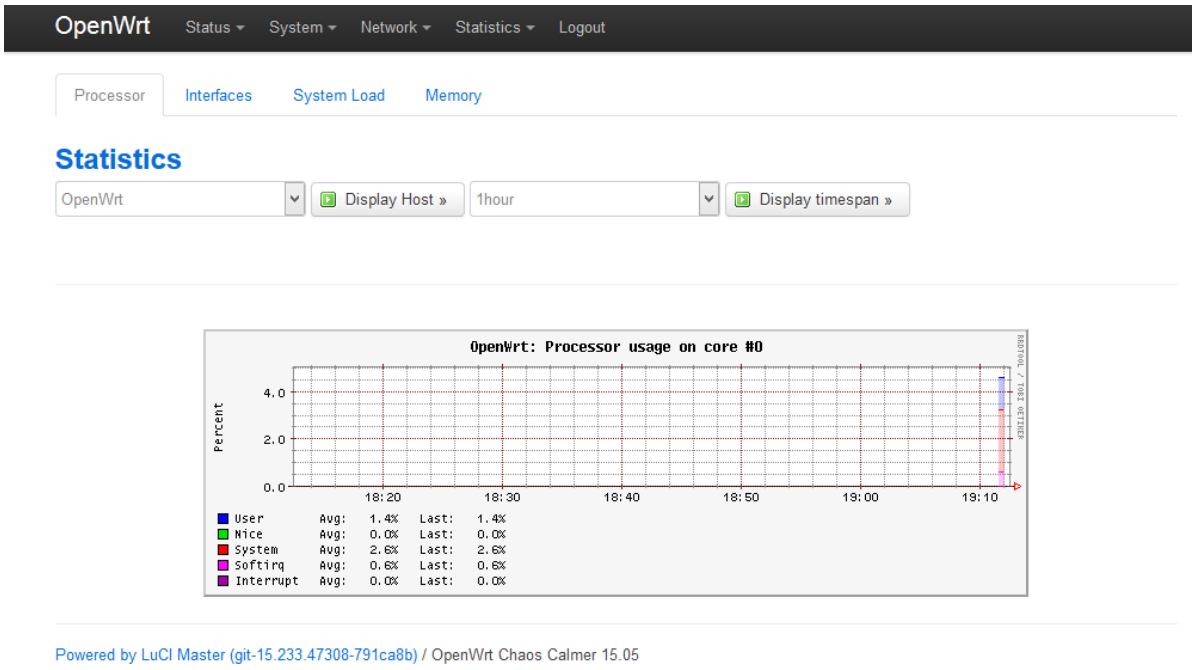
5.2.1 Συλλογή μετρικών με collectd

Η λειτουργία παρακολούθησης δικτύου που υλοποιούμε αποτελείται από δύο μέρη, τη συλλογή των μετρικών επίδοσης από τις συσκευές και την παρουσίασή τους με τη μορφή γραφημάτων. Για τη συλλογή των μετρικών και τη μεταφορά τους στον κεντρικό εξυπηρετητή θα χρησιμοποιήσουμε το δαίμονα collectd, ο οποίος χρησιμοποιεί λίγους πόρους και επιτρέπει τη συλλογή μετρικών με μεγάλη ανάλυση ακόμα και σε ενσωματωμένες συσκευές που τρέχουν OpenWrt [2] .



Η σχεδίαση του collectd είναι αρθρωτή. Ο ίδιος ο δαίμονας υλοποιεί μόνο μια υποδομή για το φιλτράρισμα και την προώθηση των δεδομένων, ενώ η συλλογή και η αποθήκευσή τους γίνεται μέσω ειδικών αρθρωμάτων (plugins). Αυτό μας επιτρέπει να εγκαταστήσουμε μόνο τα αρθρώματα που θέλουμε να χρησιμοποιήσουμε. Στην περίπτωσή μας, χρησιμοποιούμε μερικά αρθρώματα για τη συλλογή μετρικών, όπως το `CPU plugin` για τις μετρικές του επεξεργαστή, το `Network plugin` για τη μεταφορά των δεδομένων από τις συσκευές στον κεντρικό εξυπηρετητή και τέλος, το `RRDtool plugin` για την αποθήκευση των δεδομένων στον κεντρικό εξυπηρετητή σε εύχρηστη μορφή. Εάν το επιθυμούμε, μπορούμε να χρησιμοποιήσουμε και το `Write HTTP plugin` για να επικοινωνία με τον εξυπηρετητή μέσω HTTP αντί για UDP.

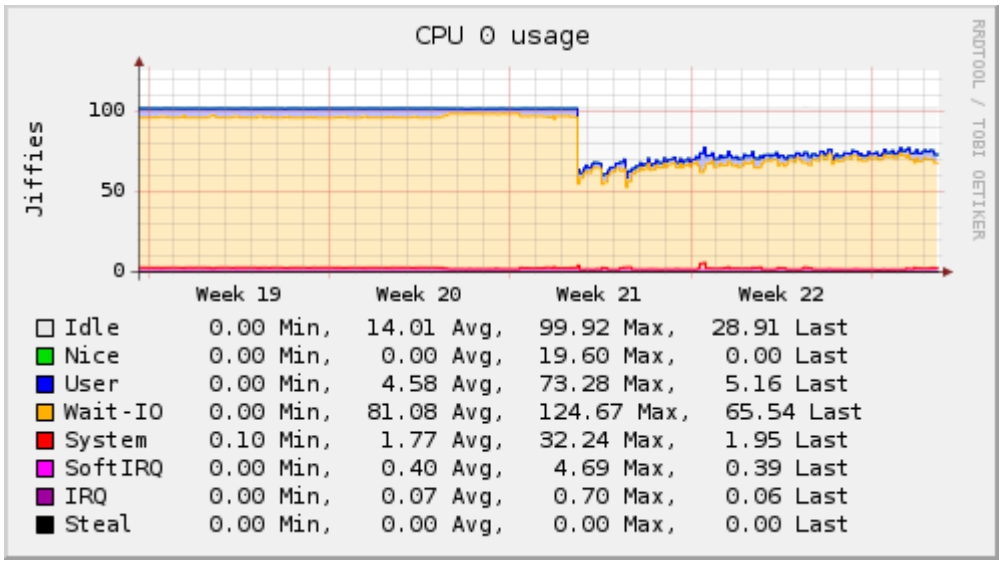
Ο δαίμονας collectd, καθώς και τα αρθρώματα που χρησιμοποιούμε, μπορούν να εγκατασταθούν σε συσκευές που τρέχουν λειτουργικό OpenWrt μέσω του διαχειριστή πακέτων opkg. Για τη χρήση των αρθρωμάτων και τη σύνδεση με τον εξυπηρετητή, θα πρέπει να εισάγουμε τις σωστές ρυθμίσεις στο αρχείο `/etc/collectd.conf` . Για μεγαλύτερη ευκολία στην περίπτωση της χειροκίνητης ρύθμισης, μπορούμε να εγκαταστήσουμε το πακέτο `luci-app-statistics`, το οποίο έχει ως προαπαιτούμενα όλα τα πακέτα που χρειαζόμαστε και προσφέρει ένα γραφικό περιβάλλοντος για τη ρύθμιση του collectd μέσω της διεπαφής ιστού luci.



Στον εξυπηρετητή, τα εισερχόμενα δεδομένα αποθηκεύονται σε κυκλικά ενημερώσιμες βάσεις δεδομένων (RRD - Round Robin Database), οι οποίες καταλαμβάνουν σταθερό χώρο. Έτσι μπορούμε να συλλέγουμε δεδομένα συνεχόμενα χωρίς να χρειάζεται ολοένα και αυξανόμενος χώρος για την αποθήκευσή τους. Για κάθε συσκευή που στέλνει δεδομένα, το collectd δημιουργεί ένα φάκελο (με όνομα το hostname της συσκευής) στον οποίο αποθηκεύει τα δεδομένα κάθε αρθρώματος σε διαφορετικό αρχείο rrd. Στην περίπτωση που προσθέσουμε επιπλέον αρθρώματα, θα δημιουργηθούν αυτόματα νέα αρχεία rrd, χωρίς να επηρεάσουν τα ήδη υπάρχοντα.

Ειδική μέριμνα πρέπει να δοθεί στις ρυθμίσεις του αρθρώματος `RRDtool plugin` για να αποφευχθούν προβλήματα εισόδου-εξόδου από τη μαζική και επαναλαμβανόμενη εγγραφή και ανάγνωση δεδομένων από το δίσκο. Στην περίπτωση που τα αρχεία rrd στον εξυπηρετητή δε χωράνε στη μνήμη του συστήματος (πράγμα πολύ πιθανό όταν συλλέγουμε δεδομένα από μεγάλο αριθμό συσκευών), το σύστημα αναγκάζεται να ζητά διαρκώς τυχαία πρόσβαση στο δίσκο. Οι σκληροί δίσκοι δεν μπορούν εύκολα να αντεπεξέλθουν σε πολλαπλές μικρές εγγραφές σε μη συνεχόμενα κομμάτια μνήμης και η ταχύτητα εγγραφής τους μπορεί να πέσει κάτω από τα 2 megabyte το δευτερόλεπτο όταν οι εγγραφές ακολουθούν τέτοιο μοτίβο. Αυτή η ταχύτητα δεν επαρκεί για τον όγκο των δεδομένων που θέλουμε να γράψουμε (αφού η ενημέρωση μιας και μόνο τιμής σε ένα αρχείο rrd απαιτεί τη μεταφορά ενός block 512 bytes από το δίσκο στη μνήμη και πίσω).

Για την αντιμετώπιση αυτής της συμφόρησης ενδείκνυται ο περιορισμός των εγγραφών στο δίσκο θέτοντας την επιλογή `WritesPerSecond` σε κάποιο χαμηλό νούμερο, όπως πχ. 50 ανά δευτερόλεπτο, και η χρήση της εντολής `FLUSH` όταν θέλουμε να δημιουργήσουμε ένα γράφημα, για να εξαναγκάσουμε το σύστημα να γράψει άμεσα τα πιο πρόσφατα δεδομένα στο δίσκο [3].



5.2.2 Δημιουργία γραφημάτων με RRDtool

Για τη δημιουργία γραφημάτων από τις μετρικές που έχουμε συλλέξει, θα χρησιμοποιήσουμε το εργαλείο RRDtool, το οποίο είναι γραμμένο από τον Tobi Oetiker, δημιουργό του MRTG (Multi Router Traffic Grapher), και αποτελεί ουσιαστικά την επόμενη γενιά του MRTG. Το RRDtool είναι σχεδιασμένο για την αποθήκευση χρονοσειρών και την παρουσίασή τους με μορφή γραφημάτων. Η αποθήκευση των δεδομένων γίνεται, όπως προαναφέραμε, σε κυκλικά ενημερώσιμες βάσεις δεδομένων. Τα νέα δεδομένα που εισάγονται στη βάση αντικαθιστούν τα παλαιότερα. Επιπλέον, όσο παλαιότερα είναι τα δεδομένα, τόσο μικραίνει ο βαθμός λεπτομέρειας με τον οποίο είναι αποθηκευμένα. Έτσι επιτυγχάνουμε η βάση να έχει σταθερό μέγεθος, υψηλή ανάλυση στα πρόσφατα δεδομένα και μεγάλο χρόνο διατήρησης των δεδομένων, με αποτέλεσμα να μπορούμε να δημιουργήσουμε γραφήματα που παρουσιάζουν τις μεταβολές των μετρικών που συλλέγουμε τόσο σε πραγματικό χρόνο όσο και ιστορικά.

Το RRDtool μπορεί να δημιουργεί γραφήματα από τα αρχεία rrd μέσω της εντολής rrdtool graph. Μέσω παραμέτρων μπορούμε να ρυθμίσουμε το χρονικό διάστημα που θα παραστήσουμε στο γράφημα, το μέγεθος, τον τίτλο και τις μονάδες των αξόνων του γραφήματος τα δεδομένα που θέλουμε να απεικονίσουμε και τη μορφή της απεικόνισης. Ακολουθεί ένα παράδειγμα:

```
rrdtool graph cpu-user.png                                  \
         --start now-1000s --end now                        \
         DEF:cpu-user-avg=cpu-user.rrd:value:AVERAGE        \
         LINE2:cpu-user-avg#FF0000
```

Η παραπάνω εντολή δημιουργεί ένα γράφημα το οποίο απεικονίζει με μια κόκκινη γραμμή το ποσοστό χρήσης της cpu σε επίπεδο χρήστη τα τελευταία 1000 δευτερόλεπτα.

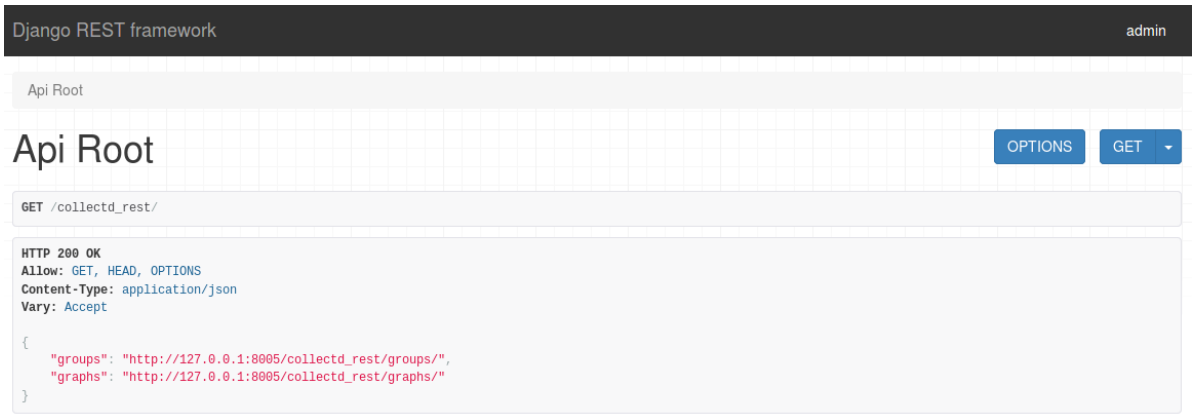
Τα αρχεία rrd δημιουργούνται αυτόματα από το collectd σε ένα φάκελο που ορίζεται στις ρυθμίσεις του RRDtool plugin (π.χ. `/var/lib/collectd/rrd/`). Για κάθε συσκευή που στέλνει δεδομένα, το collectd δημιουργεί ένα φάκελο με όνομα το hostname της συσκευής. Εκεί αποθηκεύει τα εισερχόμενα δεδομένα σε αρχεία rrd, τα ονόματα των οποίον ξεκινούν με το όνομα του αντίστοιχου αρθρώματος που σύλλεξε τα δεδομένα (πχ. cpu0.rrd για τις μετρικές του πρώτου πυρήνα της CPU της συσκευής). Για να δημιουργήσουμε λοιπόν τα γραφήματά μας, διατρέχουμε το φάκελο που ορίσαμε στις ρυθμίσεις του collectd και ψάχνουμε για κάθε συσκευή έναν υπό-φάκελο με όνομα το hostname της συσκευής. Για κάθε αρχείο rrd (των αρθρωμάτων που χρησιμοποιούμε) στον υποφάκελο αυτό, δημιουργούμε στην εφαρμογή ένα αντικείμενο τύπου γραφήματος με την αντίστοιχη εντολή rrdtool graph.

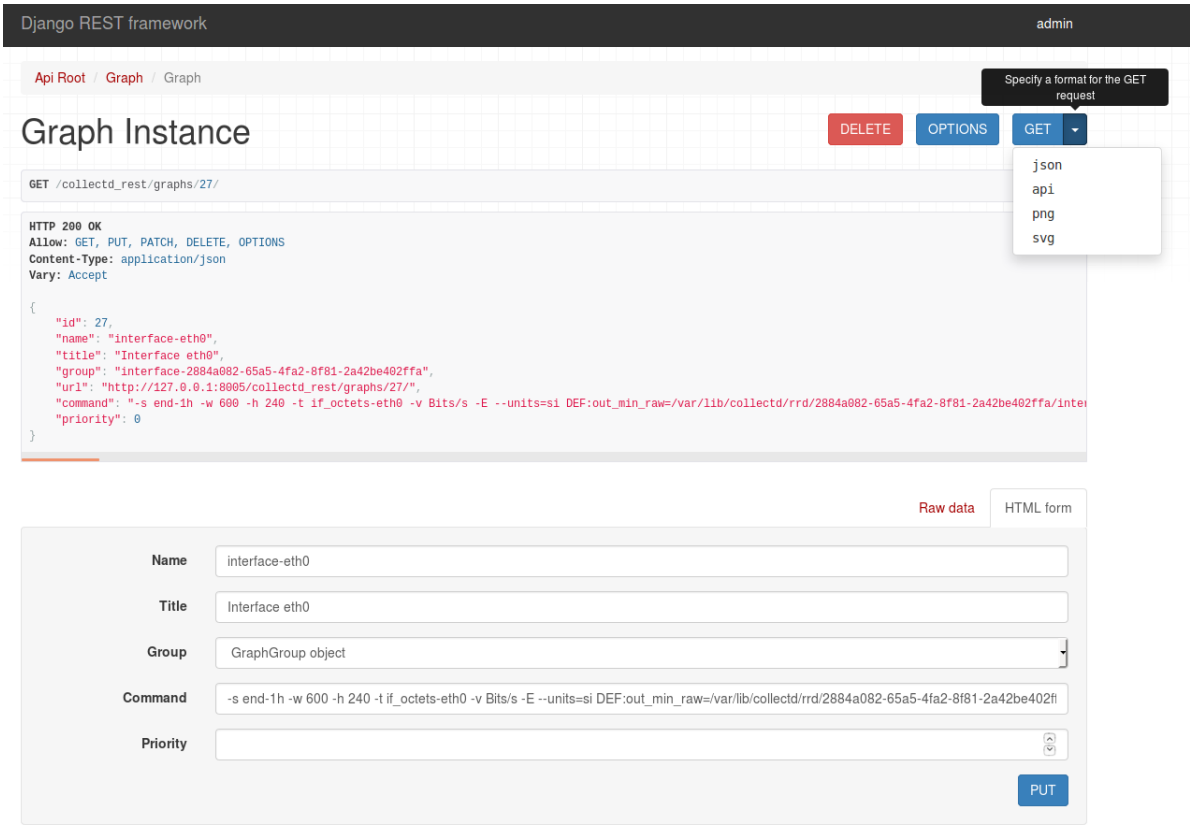
```
def make_graphs(device):
    hostname = str(device.config.id)
    path = RRD_DIR + hostname + "/"
    cpugroup,created =
GraphGroup.objects.get_or_create(name="cpu-"+hostname, title="cpu")
    if not created:
        cpugroup.graphs.all().delete()
    memorygroup,created =
GraphGroup.objects.get_or_create(name="memory-"+hostname,
title="memory")
    if not created:
        memorygroup.graphs.all().delete()
    interfacegroup,created =
GraphGroup.objects.get_or_create(name="interface-"+hostname,
title="interface")
    if not created:
        interfacegroup.graphs.all().delete()
    loadgroup,created =
GraphGroup.objects.get_or_create(name="load-"+hostname, title="load")
    if not created:
        loadgroup.graphs.all().delete()
    for file in os.listdir(path):
        if file.startswith("cpu"):
            make_cpu_graph(path+file, cpugroup, file[4:])
        if file.startswith("memory"):
            make_memory_graph(path+file, memorygroup)
        if file.startswith("interface"):
            make_interface_graph(path+file, interfacegroup, file[10:])
        if file.startswith("load"):
            make_load_graph(path+file, loadgroup)
    graphmanager,created =
GraphManager.objects.get_or_create(device=device,
                        cpugraphs = cpugroup,
                        memorygraphs = memorygroup,
                        interfacegraphs = interfacegroup,
                        loadgraphs = loadgroup,
                        )
```

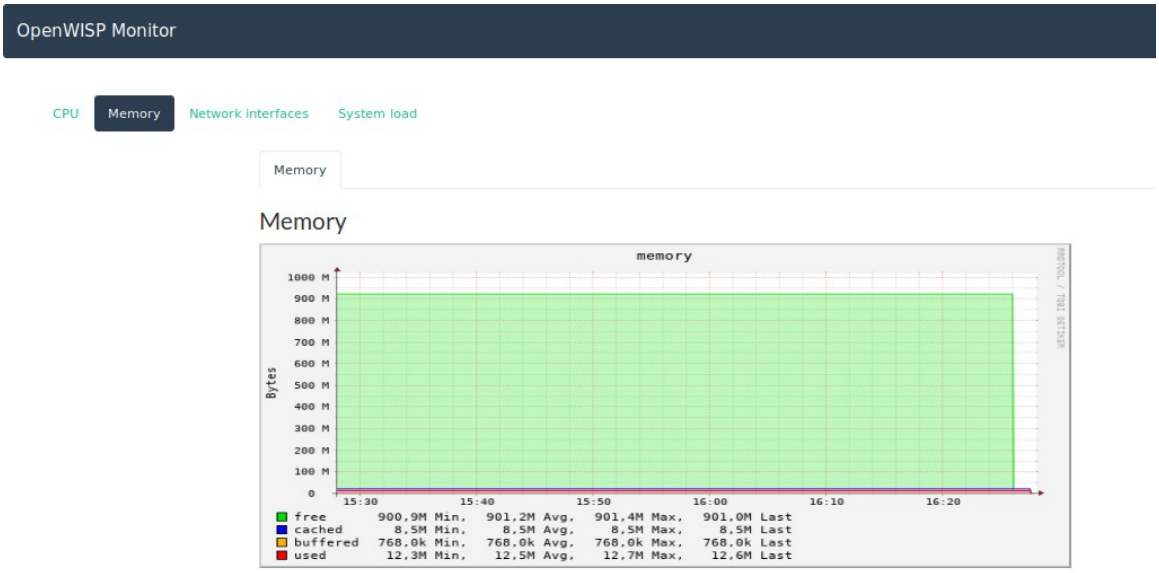
Παράδειγμα δημιουργίας γραφημάτων για τις μετρικές των δικτυακών διεπαφών τις συσκευής:

```
def make_interface_graph(interfacerrd, group,
interface_name):
    interface_args = [
        '-s end-1h',
        '-w 600',
        '-h 240',
        '-t if_octets-eth0' ,
        '-v Bits/s' ,
        '-E' ,
        '--units=si' ,
```

```
        'DEF:out_min_raw='+interfacerrd+'/if_octets.rrd:tx:MIN' ,
        'DEF:out_avg_raw='+interfacerrd+'/if_octets.rrd:tx:AVERAGE' ,
        'DEF:out_max_raw='+interfacerrd+'/if_octets.rrd:tx:MAX' ,
        'DEF:inc_min_raw='+interfacerrd+'/if_octets.rrd:rx:MIN' ,
        'DEF:inc_avg_raw='+interfacerrd+'/if_octets.rrd:rx:AVERAGE' ,
        'DEF:inc_max_raw='+interfacerrd+'/if_octets.rrd:rx:MAX' ,
        'CDEF:out_min=out_min_raw,8,*' ,
        'CDEF:out_avg=out_avg_raw,8,*' ,
        'CDEF:out_max=out_max_raw,8,*' ,
        'CDEF:inc_min=inc_min_raw,8,*' ,
        'CDEF:inc_avg=inc_avg_raw,8,*' ,
        'CDEF:inc_max=inc_max_raw,8,*' ,
        'CDEF:overlap=out_avg,inc_avg,GT,inc_avg,out_avg,IF' ,
        'CDEF:mytime=out_avg_raw,TIME,TIME,IF' ,
        'CDEF:sample_len_raw=mytime,PREV(mytime),-' ,
        'CDEF:sample_len=sample_len_raw,UN,0,sample_len_raw,IF' ,
        'CDEF:out_avg_sample=out_avg_raw,UN,0,out_avg_raw,IF,sample_le
n,*' ,
        'CDEF:out_avg_sum=PREV,UN,0,PREV,IF,out_avg_sample,+' ,
        'CDEF:inc_avg_sample=inc_avg_raw,UN,0,inc_avg_raw,IF,sample_le
n,*' ,
        'CDEF:inc_avg_sum=PREV,UN,0,PREV,IF,inc_avg_sample,+' ,
        'AREA:out_avg#B7EFB7' ,
        'AREA:inc_avg#B7B7F7' ,
        'AREA:overlap#89B3C9' ,
        '"LINE1:out_avg#00E000:Outgoing"' ,
        '"GPRINT:out_avg:AVERAGE:%5.1lf%s Avg,"' ,
        '"GPRINT:out_max:MAX:%5.1lf%s Max,"' ,
        '"GPRINT:out_avg:LAST:%5.1lf%s Last"' ,
        '"GPRINT:out_avg_sum:LAST:(ca. %5.1lf%sB Total)\\l"' ,
        '"LINE1:inc_avg#0000FF:Incoming"' ,
        '"GPRINT:inc_avg:AVERAGE:%5.1lf%s Avg,"' ,
        '"GPRINT:inc_max:MAX:%5.1lf%s Max,"' ,
        '"GPRINT:inc_avg:LAST:%5.1lf%s Last"' ,
        '"GPRINT:inc_avg_sum:LAST:(ca. %5.1lf%sB Total)\\l"'
    ]
    interface_line_args = " ".join(interface_args)
    graph = Graph(name="interface-"+interface_name, group=group)
    graph.title="Interface "+interface_name
    graph.command=interface_line_args
    graph.priority=0
    graph.save()
```


Django REST framework
admin
Api Root
Api Root
OPTIONS
GET
GET /collectd_rest/
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"groups": "http://127.0.0.1:8005/collectd_rest/groups/",
"graphs": "http://127.0.0.1:8005/collectd_rest/graphs/"
}

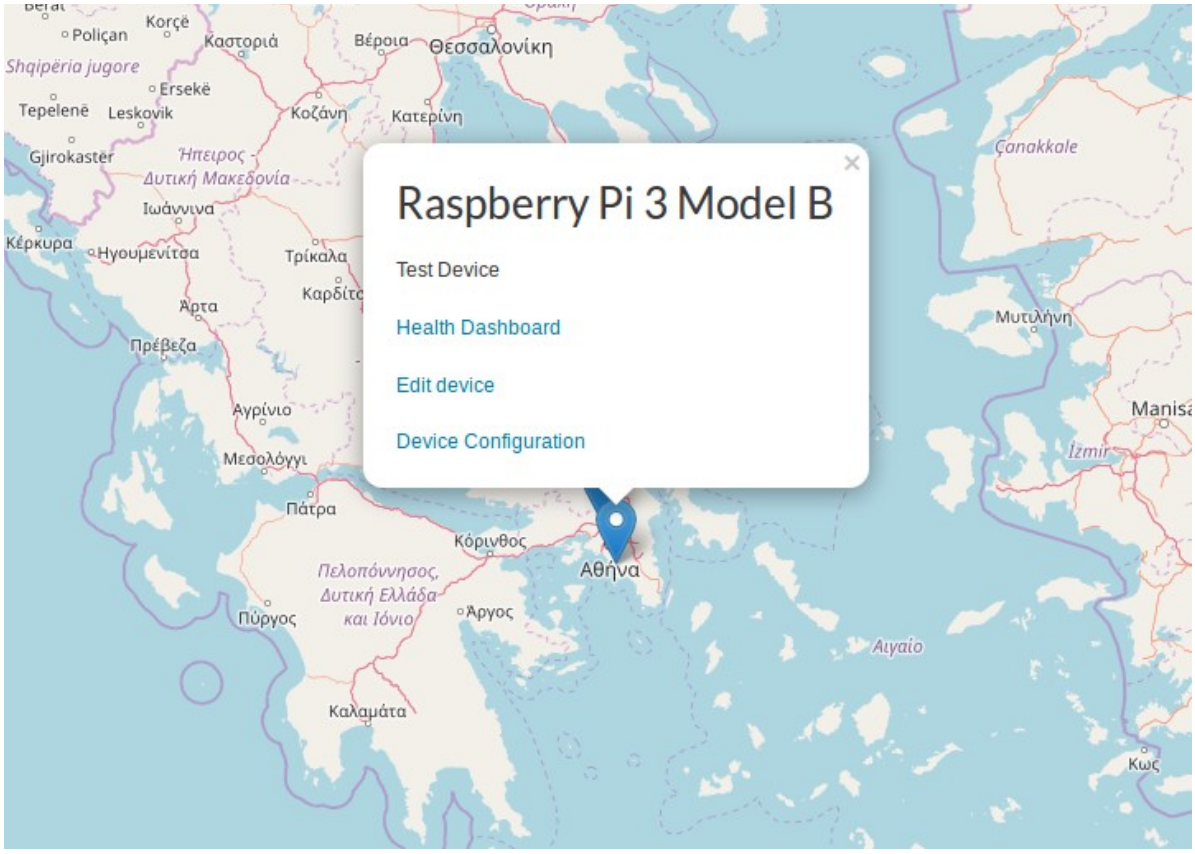

Django REST framework
admin
Api Root / Graph / Graph
Specify a format for the GET request
Graph Instance
DELETE
OPTIONS
GET
json
api
png
svg
GET /collectd_rest/graphs/27/
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, OPTIONS
Content-Type: application/json
Vary: Accept
{
"id": 27,
"name": "interface-eth0",
"title": "Interface eth0",
"group": "interface-2884a082-65a5-4fa2-8f81-2a42be402ffa",
"url": "http://127.0.0.1:8005/collectd_rest/graphs/27/",
"command": "-s end-1h -w 600 -h 240 -t if_octets-eth0 -v Bits/s -E --units=si DEF:out_min_raw=/var/lib/collectd/rrd/2884a082-65a5-4fa2-8f81-2a42be402ffa/inte
"priority": 0
}
Raw data
HTML form
Name
interface-eth0
Title
Interface eth0
Group
GraphGroup object
Command
-s end-1h -w 600 -h 240 -t if_octets-eth0 -v Bits/s -E --units=si DEF:out_min_raw=/var/lib/collectd/rrd/2884a082-65a5-4fa2-8f81-2a42be402ffa
Priority
PUT


OpenWISP Monitor
CPU
Memory
Network interfaces
System load
Memory
Memory
memory
1000 M
900 M
800 M
700 M
600 M
500 M
400 M
300 M
200 M
100 M
0
Bytes
15:30
15:40
15:50
16:00
16:10
16:20
free 900,9M Min, 901,2M Avg, 901,4M Max, 901,0M Last
cached 8,5M Min, 8,5M Avg, 8,5M Max, 8,5M Last
buffered 768,0k Min, 768,0k Avg, 768,0k Max, 768,0k Last
used 12,3M Min, 12,5M Avg, 12,7M Max, 12,6M Last

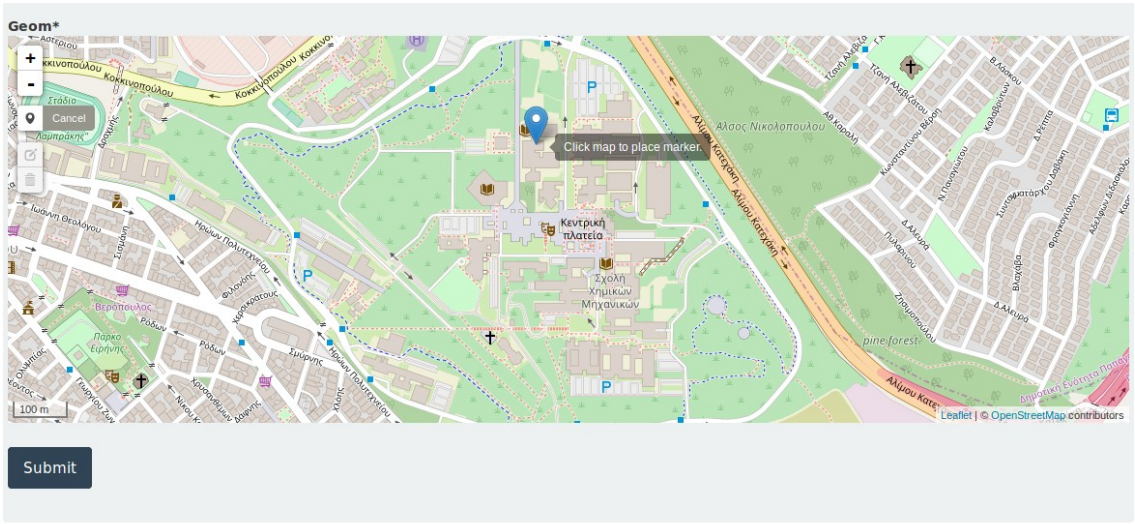

OpenWISP Monitor
CPU
Memory
Network interfaces
System load
Interface eth0
Interface wlan0
Interface eth0
if_octets-eth0
2,0 k
1,8 k
1,6 k
1,4 k
1,2 k
1,0 k
0,8 k
0,6 k
0,4 k
0,2 k
Bits/s
15:30
15:40
15:50
16:00
16:10
16:20
Outgoing 1,5k Avg, 1,8k Max, 1,2k Last (ca. 647,5kB Total)
Incoming 193,2 Avg, 549,2 Max, 28,2 Last (ca. 85,4kB Total)

5.2.3 Γεωγραφική απεικόνιση

Μία ακόμα λειτουργία της πρώτης έκδοσης του OpenWISP, την οποία θέλουμε να υλοποιήσουμε για τη δεύτερη, είναι η δυνατότητα απεικόνισης των συσκευών μας πάνω στο χάρτη, σύμφωνα με την τοποθεσία τους. Συγκεκριμένα, η κεντρική οθόνη της εφαρμογής μας θέλουμε να απεικονίζει όλες τις συσκευές υπό την κατοχή μας ως κόμβους σε ένα χάρτη. Επιλέγοντας ένα κόμβο, ο διαχειριστής θα λαμβάνει βασικές πληροφορίες για τη συσκευή και θα έχει πρόσβαση στις βασικές ενέργειες της συσκευής, όπως η παρακολούθηση των μετρικών της και η επεξεργασία της διάρθρωσης της.



Για να υλοποιήσουμε αυτή τη λειτουργία πρέπει καταρχάς να αποθηκεύσουμε την τοποθεσία κάθε συσκευής στη βάση δεδομένων της εφαρμογής. Στο μοντέλο των συσκευών έχουμε προσθέσει ένα πεδίο τύπου `PointField` του πακέτου `django-geojson`. Αυτό το πεδίο αποθηκεύει τις συντεταγμένες της συσκευής στη βάση με τη μορφή ενός αντικειμένου τύπου `Point` του μορφότυπου GeoJSON [4] . Στις οθόνες δημιουργίας και επεξεργασίας των συσκευών, παρέχουμε στους χρήστες έναν εύκολο τρόπο εισαγωγής των συντεταγμένων. Παρουσιάζουμε στους χρήστες ένα χάρτη, μέσα από τον οποίο μπορούν να εντοπίσουν την τοποθεσία της συσκευής και να τη σημειώσουν πάνω στο χάρτη.



5.3 Μαζική εφαρμογή προτύπων

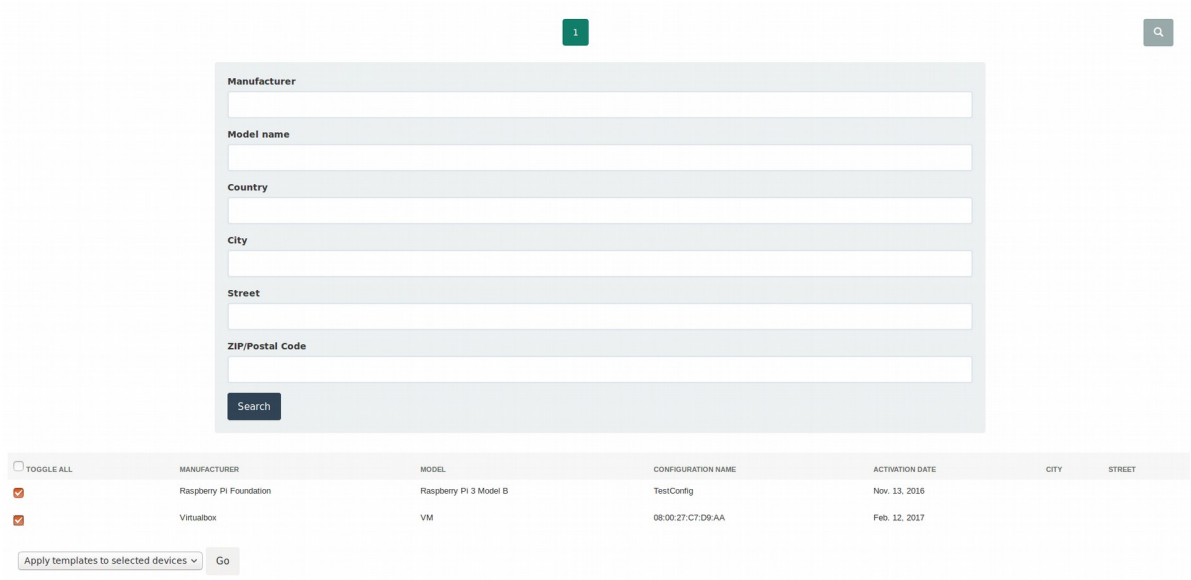
Για την διαχείριση μιας ομάδας συσκευών με κοινή διάρθρωση, το OpenWISP χρησιμοποιεί τα πρότυπα διάρθρωσης. Οι διαχειριστές μπορούν να δημιουργήσουν ένα πρότυπο με όλες τις ρυθμίσεις που αφορούν κάποια κοινή λειτουργία ή κοινό χαρακτηριστικό των συσκευών και έπειτα να εφαρμόσουν το πρότυπο αυτό σε όλες τις συσκευές που επιθυμούν. Εάν στην πορεία αλλάξουν οι απαιτήσεις, οι διαχειριστές θα χρειαστεί να επέμβουν μόνο σε αυτό το πρότυπο, χωρίς να επαναλάβουν τις ίδιες αλλαγές για όλες τις συσκευές και αποφεύγοντας λάθη.

Η ιδέα των προτύπων είναι πολύ χρήσιμη για τη διαχείριση σημείων πρόσβασης, επειδή λόγω της λειτουργίας τους είναι συνήθως οργανωμένα σε ομάδες με κοινές ρυθμίσεις. Όμως, σε ένα μεγάλο δίκτυο με εκατοντάδες ή χιλιάδες σημεία πρόσβασης, η εφαρμογή ενός ή περισσότερων προτύπων διάρθρωσης απαιτεί πολλές ενέργειες από το χρήστη (μία για κάθε συσκευή και πρότυπο). Για να μειώσουμε τις απαραίτητες ενέργειες του χρήστη για την εφαρμογή προτύπων, θέλουμε να υλοποιήσουμε μια ακόμα λειτουργία της εφαρμογής όπου ο χρήστης με μία εντολή θα μπορεί να εφαρμόζει ένα σύνολο προτύπων σε μια ομάδα συσκευών που έχουν κοινά χαρακτηριστικά.

5.3.1 Αναζήτηση συσκευών

Πρώτο βήμα για τη μαζική εφαρμογή προτύπων είναι η γρήγορη επιλογή των συσκευών-στόχων. Θεωρούμε ότι οι συσκευές έχουν κάποιο κοινό χαρακτηριστικό πχ. ίδιο μοντέλο/κατασκευαστή ή ίδια διεύθυνση. Θέλουμε να επιτρέψουμε στο χρήστη να αναζητά τις συσκευές της διαχειριστικής του ομάδας με βάση τα δεδομένα που έχουμε αποθηκευμένα για αυτές. Για να πραγματοποιήσει ο χρήστης αυτή την αναζήτηση του παρέχουμε μια φόρμα αναζήτησης μέσα από την οποία μπορεί να βρει όλες τις συσκευές με κοινές τιμές σε κάποια πεδία. Στη συνέχεια, μπορεί να επιλέξει τις συσκευές που τον ενδιαφέρουν μία προς μία ή ακόμα και να επιλέξει όλες τις συσκευές που επέστρεψε η αναζήτηση με μία κίνηση. Όταν ολοκληρώσει την επιλογή του, την αποθηκεύουμε προσωρινά στον εξυπηρετητή μέσα στο αντικείμενο της συνεδρίας (session) του χρήστη καθώς θα το χρειαστούμε αργότερα.

```
def select_devices_action(view, queryset):
        view.request.session['devices_queryset'] =
pickle.dumps(queryset.query)
         return
HttpResponseRedirect(reverse("devices_apply_template"))
```



5.3.2 Εφαρμογή πολλαπλών προτύπων σε ομάδα συσκευών

Έχοντας επιλέξει μία ή περισσότερες συσκευές, ο χρήστης οδηγείται στην επόμενη οθόνη όπου μπορεί να επιλέξει ποια από τα διαθέσιμα πρότυπα θέλει να εφαρμόσει στις συσκευές. Έπειτα, ανακτούμε από το αντικείμενο της συνεδρίας του χρήστη το σύνολο των επιλεγμένες συσκευών και σε καθεμία από αυτές εφαρμόζουμε με τη σειρά όλα τα επιλεγμένα πρότυπα διάρθρωσης.

```
def apply_template_action(view, queryset):
    if 'devices_queryset' not in view.request.session:
        return HttpResponseBadRequest("Session is missing the
devices_queryset key.")
    devices_queryset = Device.objects.all()[:1]
    devices_queryset.query =
pickle.loads(view.request.session.get('devices_queryset'))
    templates_queryset = queryset
    for device in devices_queryset.iterator():
        config = Config.objects.get(id=device.config.id)
        for template in templates_queryset.iterator():
            config.templates.add(template)
            config.save()
    return HttpResponseRedirect(reverse("devices_home"))
```

Κεφάλαιο 6

Εγκατάσταση

Στο κεφάλαιο αυτό θα παρουσιάσουμε οδηγίες για την εγκατάσταση της πλατφόρμας OpenWISP, μαζί με τις δικές μας προσθήκες, για χρήση στο δίκτυό μας. Η εγκατάσταση του εξυπηρετητή OpenWISP2 μέσω αυτών των οδηγιών έχει δοκιμαστεί σε μηχανήματα Ubuntu, Debian και CentOS. Ως ασύρματα σημεία πρόσβασης για τις δοκιμές, χρησιμοποιήθηκαν μηχανήματα Raspberry Pi Model 2B και Model 3 με λειτουργικό σύστημα OpenWrt.

6.1 Εγκατάσταση του εξυπηρετητή

Η εγκατάσταση του OpenWISP2 controller μπορεί να γίνει αυτόματα, είτε μέσω του εργαλείου ansible είτε μέσω του εργαλείου vagrant.

Το εργαλείο ansible συνδέεται μέσω ssh στο μηχάνημα όπου θέλουμε να εγκαταστήσουμε τον εξυπηρετητή μας και εκτελεί μια σειρά ταυτοδύναμων (idempotent) ενεργειών για την εγκατάσταση και την παραμετροποίηση ολόκληρης της στοίβας λογισμικού που χρησιμοποιεί ο εξυπηρετητής. Το εργαλείο αυτό μας επιτρέπει να κάνουμε τις δικές μας επιλογές για τα επιμέρους τμήματα της στοίβας λογισμικού, όπως για παράδειγμα τη βάση δεδομένων που θέλουμε να χρησιμοποιήσουμε.

Αντίθετα, το εργαλείο vagrant μας προμηθεύει με ένα εικονικό μηχάνημα Ubuntu στο οποίο είναι ήδη εγκατεστημένος ο εξυπηρετητής OpenWISP2 controller. Στη συνέχεια, συνδεόμαστε στο εικονικό μηχάνημα μέσω ssh και παραμετροποιούμε τον εξυπηρετητή σύμφωνα με τις απαιτήσεις μας. Στην περίπτωση που είμαστε ικανοποιημένοι από την τεχνολογική στοίβα του εικονικού μηχανήματος, η εγκατάσταση μέσω vagrant είναι ευκολότερη από την εγκατάσταση μέσω ansible. Ωστόσο, η χρήση ενός εικονικού μηχανήματος συνεπάγεται με έμμεσα κόστη στη μνήμη και την απόδοση του συστήματος.

Τέλος, ο εξυπηρετητής μπορεί να εγκατασταθεί και χειροκίνητα. Αυτή η μέθοδος μας δίνει τη μεγαλύτερη ευελιξία στην παραμετροποίηση και την επιλογή λογισμικού για τον εξυπηρετητή, αλλά απαιτεί πολύ περισσότερο κόπο και χρόνο από την αυτοματοποιημένη εγκατάσταση και είναι περισσότερο επιρρεπής σε λάθη. Θα ξεκινήσουμε περιγράφοντας τη διαδικασία χειροκίνητης εγκατάστασης, ώστε να εξηγήσουμε ολόκληρη τη στοίβα λογισμικού που απαιτεί ο εξυπηρετητής για τη λειτουργία του.

6.1.1 Χειροκίνητη εγκατάσταση

Python, pip και virtualenv

Η εφαρμογή OpenWISP2 controller είναι γραμμένη στη γλώσσα Python. Επομένως, για να την εκτελέσουμε θα χρειαστούμε έναν διερμηνέα Python. Εάν το σύστημά μας δεν έχει ήδη εγκατεστημένο ένα διερμηνέα έκδοσης 2.7 ή 3.4 και άνω, τότε θα πρέπει να τον εγκαταστήσουμε μέσω του διαχειριστή πακέτων του λειτουργικού μας συστήματος. Επίσης, θα χρειαστούμε και το διαχειριστή πακέτων pip μέσω του οποίου θα εγκαταστήσουμε την εφαρμογή και τις βιβλιοθήκες που χρησιμοποιεί. Εάν το πρόγραμμα pip δεν είναι ήδη εγκατεστημένο, μπορούμε να το εγκαταστήσουμε κατεβάζοντας το script get-pip.py [1] και εκτελώντας το μέσω του διερμηνέα python. Προαιρετικά, μπορούμε να εγκαταστήσουμε μέσω του pip και το εργαλείο virtualenv για τη δημιουργία ενός απομονωμένου (εικονικού) περιβάλλοντος python στο οποίο μπορούμε να εγκαταστήσουμε όλα τα προαπαιτούμενα της εφαρμογής, αποφεύγοντας τις συγκρούσεις με το υπόλοιπο σύστημα και διευκολύνοντας έτσι τη διαδικασία εγκατάστασης και αναβάθμισης της εφαρμογής.

Στη συνέχεια, κατεβάζουμε την εφαρμογή μέσα σε ένα φάκελο της επιλογής μας (για τις οδηγίες που ακολουθούν θεωρούμε ότι επιλέξαμε το φάκελο /opt/openwisp2/). Μπορούμε να την κατεβάσουμε είτε σε μορφή zip μέσω HTTPS από την τοποθεσία

https://github.com/DimPolissiou/OpenWISP-Monitor/archive/master.zip

είτε με χρήση κάποιου συστήματος ελέγχου πηγαίου κώδικα, όπως το Git ή το Subversion, από την τοποθεσία

https://github.com/DimPolissiou/OpenWISP-Monitor.git .

Εάν επιλέξαμε να χρησιμοποιήσουμε το εργαλείο virtualenv, δημιουργούμε ένα εικονικό περιβάλλον με την εντολή `virtualenv venv` και το ενεργοποιούμε με την εντολή `source venv/bin/activate`. Τέλος, εγκαθιστούμε όλα τα προαπαιτούμενα της εφαρμογής με την εντολή `pip install -r requirements.txt`.

Παραμετροποίηση της εφαρμογής

Αφού εγκαταστήσουμε την εφαρμογή, μπορούμε να την παραμετροποιήσουμε τροποποιώντας το αρχείο settings.py στο φάκελο devices. Στο παράρτημα 2 παραθέτουμε ένα ενδεικτικό αρχείο settings.py . Μερικές από τις επιλογές που είναι πιθανό να χρειάζονται αλλαγή είναι οι παρακάτω:

- SECRET_KEY : Μια μυστική συμβολοσειρά. Μπορούμε να επιλέξουμε αυθαίρετα το περιεχόμενό της ή να το δημιουργήσουμε με τον παρακάτω κώδικα python

```
    import random

    chars = 'abcdefghijklmnopqrstuvwxyz' \
                'ABCDEFGHIJKLMNOPQRSTUVXYZ' \
                '0123456789' \
          '#()^[]-_*%&=+/'
   SECRET_KEY = ''.join([random.SystemRandom().choice(chars)
for i  in range(50)])
```

- `ALLOWED_HOSTS` : Η λίστα αυτή πρέπει να περιέχει όλες τις διευθύνσεις στις οποίες θέλουμε να ακούει ο εξυπηρετητής.

- `DATABASES` : Εδώ τοποθετούμε τις ρυθμίσεις σύνδεσης της βάσης δεδομένων που θα χρησιμοποιεί ο εξυπηρετητής. Μπορούμε να συμβουλευτούμε το εγχειρίδιο του Django [2] .

- `TIME_ZONE` : Η ζώνη ώρας του εξυπηρετητή. Εάν χρησιμοποιούμε το collectd, πρέπει να είναι ίδια με τη ζώνη ώρας των δρομολογητών μας.

- `NETJSONCONFIG_SHARED_SECRET` : Το μυστικό κλειδί με το οποίο πραγματοποιούν οι δρομολογητές αυτόματη εγγραφή στο σύστημα.

- `CAS_SERVER_URL` : Η διεύθυνση του εξυπηρετητή CAS.

- `AFFILIATION_FIELD` : Το όνομα του attribute που μας επιστρέπει ο εξυπηρετητής CAS το οποίο αντιστοιχεί στη διαχειριστική ομάδα που ανήκει ο χρήστης.

- `COLLECTD_RRD_DIR` : Ο φάκελος στον οποίο το collectd τοποθετεί τα αρχεία RRD.

Μετά την παραμετροποίηση, πρέπει να αρχικοποιήσουμε τη βάση δεδομένων, δηλαδή να δημιουργήσουμε τους πίνακες που χρησιμοποιεί η εφαρμογή. Αυτό το επιτυγχάνουμε με την εντολή `python manage.py migrate`. Επίσης, με την εντολή `python manage.py collectstatic` συγκεντρώνουμε όλα τα στατικά αρχεία της εφαρμογής στο φάκελο που ορίζει το αρχείο `setttings.py`. Τώρα μπορούμε να δοκιμάσουμε την εφαρμογή εκτελώντας τον εξυπηρετητή ανάπτυξης (development server) του πλαισίου Django με την εντολή `python manage.py runserver 0.0.0.0:8080`. Μέσω ενός φυλλομετρητή ιστού, επισκεπτόμαστε τη διεύθυνση του εξυπηρετητή (θύρα 8080) και επιβεβαιώνουμε την καλή λειτουργία της εφαρμογής.

Εξυπηρετητής WSGI

Για την επικοινωνία με εξυπηρετητές ιστού, η εφαρμογή OpenWISP2 controller υλοποιεί την προδιαγραφή WSGI. Η προδιαγραφή WSGI (Web Server Gateway Interface) [3] ορίζει έναν τρόπο επικοινωνίας μεταξύ εφαρμογών python και εξυπηρετητών ιστού όπως ο Apache ή ο Nginx. Οι εξυπηρετητές ιστού δέχονται αιτήματα τύπου HTTP request και απαντούν με

μηνύματα HTTP response. Για να μετατρέψουμε τα μηνύματα HTTP σε αντικείμενα python και αντίστροφα πρέπει να χρησιμοποιήσουμε έναν εξυπηρετητή WSGI. Παραπάνω χρησιμοποιήσαμε το development server του πλαισίου Django. Ο εξυπηρετητής αυτός δεν πρέπει να χρησιμοποιείται σε παραγωγικό περιβάλλον, επειδή τρέχει σε ένα νήμα και δεν μπορεί να ανταπεξέλθει σε μεγάλο όγκο μηνυμάτων. Επομένως, θα πρέπει να εγκαταστήσουμε έναν άλλο εξυπηρετητή WSGI, όπως το uWSGI ή το Green Unicorn.

Μπορούμε να εγκαταστήσουμε τον εξυπηρετητή uWSGI μέσω του διαχειριστή πακέτων pip, με την εντολή `pip install uwsgi`. Παρακάτω παραθέτουμε ένα ενδεικτικό αρχείο παραμετροποίησης του uWSGI. Αφού τοποθετήσουμε το αρχείο αυτό στο φάκελο του openwisp με όνομα uwsgi.ini, μπορούμε να δοκιμάσουμε τον εξυπηρετητή με την εντολή `uwsgi --ini uwsgi.ini`. Αν όλα δουλεύουν σωστά, ο εξυπηρετητής μας πρέπει να ακούει στη θύρα 3031.

Περιεχόμενα του αρχείου uwsgi.ini:

```
[uwsgi]
chdir=/opt/openwisp2
module=openwisp2.wsgi:application
master=True
pidfile=/opt/openwisp2/uwsgi.pid
socket=127.0.0.1:3031
processes=2
threads=2
harakiri=20
max-requests=5000
vacuum=True
enable-threads=True
env=HTTPS=on
buffer-size=8192
```

Αφού εγκαταστήσουμε, παραμετροποιήσουμε και δοκιμάσουμε τον εξυπηρετητή WSGI, μπορούμε να χρησιμοποιήσουμε το δαίμονα Supervisor για να παρακολουθούμε την εφαρμογή και να την επανεκκινούμε όταν τερματίζει λόγο κάποιου σφάλματος ή εξωτερικού παράγοντα. Μπορούμε να εγκαταστήσουμε το δαίμονα Supervisor μέσω του διαχειριστή πακέτων pip με την εντολή `pip install supervisor`. Φροντίζουμε να μην εγκαταστήσουμε το Supervisor μέσα σε κάποιο εικονικό περιβάλλον (μπορούμε να σταματήσουμε να χρησιμοποιούμε το τρέχον εικονικό περιβάλλον με την εντολή `deactivate`). Έπειτα, δημιουργούμε ένα αρχείο παραμετροποίησης με όνομα openwisp2.conf στο φάκελο /etc/supervisor/conf.d/ (για Debian ή Ubuntu) ή στο φάκελο /etc/supervisord.d/ (για CentOS ή RHEL), με τα εξής περιεχόμενα:

```
[program:openwisp2]
user=www-data
directory=/opt/openwisp2
command=/opt/openwisp2/venv/bin/uwsgi --ini uwsgi.ini
```

```
autostart=true
autorestart=true
stopsignal=INT
redirect_stderr=true
stdout_logfile=/opt/openwisp2/log/uwsgi.log
stdout_logfile_maxbytes=30MB
stdout_logfile_backups=5
```

Επανεκκινούμε το δαίμονα supervisor με την εντολή `supervisorctl reload`. Τώρα μπορούμε να εκκινήσουμε, να σταματήσουμε ή να επανεκκινήσουμε τον εξυπηρετητή μας με τις παρακάτω εντολές:

```
supervisorctl start openwisp2
supervisorctl stop openwisp2
supervisorctl restart openwisp2
```

Collectd και RRDtool

Εάν θέλουμε να χρησιμοποιήσουμε την προαιρετική λειτουργία παρακολούθησης δικτύου, είναι απαραίτητο να εγκαταστήσουμε στον εξυπηρετητή το δαίμονα collectd, για τη συλλογή στατιστικών από τους δρομολογητές μας, και το εργαλείο RRDtool για τη δημιουργία γραφημάτων. Η εγκατάσταση αυτών των εργαλείων μπορεί να γίνει μέσω του διαχειριστή πακέτων του λειτουργικού μας συστήματος. Από όλα τα αρθρώματα του collectd, ο εξυπηρετητής έχει ανάγκη μόνο το άρθρωμα δικτύωσης (Network plugin), το άρθρωμα εγγραφής αρχείων RRD (RRDtool plugin) και ένα άρθρωμα για τη δημιουργία αρχείων καταγραφής (Logfile plugin). Αυτό σημαίνει πως, εάν ο διαχειριστής πακέτων μας δίνει την επιλογή, μπορούμε να προτιμήσουμε ένα πακέτο με τα βασικά μόνο αρθρώματα του collectd και να εξοικονομήσουμε χώρο στον εξυπηρετητή. Ένα παράδειγμα αρχείου παραμετροποίησης του collectd για τον εξυπηρετητή, είναι το παρακάτω:

```
Hostname "Server"
FQDNLookup true
AutoLoadPlugin true

LoadPlugin logfile
LoadPlugin syslog

<Plugin logfile>
    LogLevel "info"
    File STDOUT
    Timestamp true
    PrintSeverity false
</Plugin>

<Plugin syslog>
    LogLevel info
</Plugin>

LoadPlugin network
```

```
LoadPlugin rrdtool

<Plugin network>
   Listen "0.0.0.0" "25826"
</Plugin>

<Plugin rrdtool>
   DataDir "/var/lib/collectd/rrd"
   CacheTimeout 120
   CacheFlush 900
   WritesPerSecond 50
</Plugin>
```

Nginx

Ως εξυπηρετητή ιστού για τα πρωτόκολλα HTTP και HTTPS μπορούμε να χρησιμοποιήσουμε τον nginx. Ο nginx μας επιτρέπει να εξυπηρετούμε αιτήματα για στατικά αρχεία πολύ πιο αποδοτικά από όσο θα μπορούσε μια εφαρμογή python. Επίσης, μας επιτρέπει να δρομολογούμε αιτήματα HTTP και HTTPS προς διαφορετικές εφαρμογές που ακούν στην ίδια διεύθυνση (σε διαφερετική όμως θύρα). Με αυτό τον τρόπο μπορούμε στον ίδιο εξυπηρετητή να έχουμε, πέρα από τον OpenWISP2 controller, εφαρμογές όπως ο εξυπηρετητής SSO ή ο εξυπηρετητής collectd εάν θέλουμε να χρησιμοποιήσουμε το άρθρωμα WriteHTTP.

Η εγκατάσταση του nginx μπορεί να γίνει μέσω ενός διαχειριστή πακέτων όπως το aptitude ή το yum. Τα αρχεία ρύθμισης του nginx βρίσκονται στο φάκελο /etc/nginx . Οι βασικές ρυθμίσεις του nginx βρίσκονται στο αρχείο /etc/nging/nginx.conf . Επιβεβαιώνουμε ότι στο αρχείο αυτό υπάρχει η γραμμή `include /etc/nginx/sites-enabled/*;` . Η γραμμή αυτή ενημερώνει τον nginx να αναζητήσει επιπλέον αρχεία ρύθμισης στο φάκελο /etc/nginx/sites-enabled/. Για κάθε εφαρμογή που θέλουμε να εξυπηρετεί ο nginx, δημιουργούμε στο φάκελο αυτό το αντίστοιχο αρχείο ρύθμισης. Για παράδειγμα, το παρακάτω αρχείο openwisp.conf ορίζει τον nginx να εξυπηρετεί ο ίδιος όλα τα αιτήματα για στατικά αρχεία του OpenWISP2 controller και να δρομολογεί όλα τα υπόλοιπα αιτήματα στη θύρα 3031 όπου ακούει ο εξυπηρετητής uWSGI.

```
server {
    listen 443 ssl spdy;

    # Στη θέση του example.com τοποθετούμε το domain name
του εξυπηρετητή μας
    server_name example.com;

    root /opt/openwisp2/public_html;
    index index.html index.htm;

    # error log
    access_log /opt/openwisp2/log/nginx.access.log;
    error_log /opt/openwisp2/log/nginx.error.log error;
```

```
    # set client body size #
    client_max_body_size 5M;

    ssl on;
   # Στη θέση του ssl_cert τοποθετούμε το μονοπάτι του
πιστοποιητικού ssl που θέλουμε να χρησιμοποιήσουμε
    ssl_certificate ssl_cert;
   # Στη θέση του ssl_cert τοποθετούμε το μονοπάτι του
ιδιωτικού κλειδιού του παραπάνω πιστοποιητικού
    ssl_certificate_key ssl_key;
    # optimizations
    ssl_session_cache shared:SSL:20m;
    ssl_session_timeout 10m;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_ciphers
ECDH+AESGCM:ECDH+AES256:ECDH+AES128:DH+3DES:!ADH:!AECDH:!
MD5;
    add_header Strict-Transport-Security "max-age=31536000";
    add_header X-Content-Type-Options nosniff;

    location @uwsgi {
        uwsgi_pass 127.0.0.1:3031;
        include uwsgi_params;
        uwsgi_param HTTP_X_FORWARDED_PROTO https;
    }

    location / {
        try_files
/opt/openwisp2/public_html/maintenance.html $uri @uwsgi;
    }

    location /static {
        alias /opt/openwisp2/static/;
    }

    location /media {
        alias /opt/openwisp2/media/;
    }
}

server {
    listen 80;

    # Στη θέση του example.com τοποθετούμε το domain name
του εξυπηρετητή μας
    server_name example.com;

    root /opt/openwisp2/public_html;
```

```
    # Necessary for Let's Encrypt Domain Name ownership
validation
    location /.well-known/ {
        try_files   $uri /dev/null =404;
    }

   # redirect all requests to https
    location / {
        return 301 https://$host$request_uri;
    }

}
```

Με την εντολή `nginx -t` δοκιμάζουμε εάν τα αρχεία ρύθμισης που φορτώνει ο nginx περιέχουν συντακτικά λάθη. Εάν δεν περιέχουν λάθη, επανεκκινούμε τον nginx και επισκεπτόμαστε μέσω ενός φυλλομετρητή τη δημόσια διεύθυνση του εξυπηρετητή μας για να δοκιμάσουμε εάν δρομολογεί σωστά τα αιτήματά μας προς τον controller.

6.1.2 Εγκατάσταση μέσω ansible

Η εγκατάσταση του εξυπηρετητή OpenWISP, μαζί με τις τροποποιήσεις μας, μπορεί να γίνει αυτόματα με χρήση του εργαλείου ansible. Όπως αναφέραμε στο κεφάλαιο 2, το εργαλείο ansible ακολουθεί την αρχιτεκτονική agentless. Το εργαλείο τρέχει στο τοπικό μας μηχάνημα, συνδέεται στο μηχάνημα-στόχο μέσω ssh και εκτελεί εκεί τις απαραίτητες ενέργειες για την εγκατάσταση του εξυπηρετητή μας. Για να χρησιμοποιήσουμε το εργαλείο, θα πρέπει να το εγκαταστήσουμε στον τοπικό μας υπολογιστή (είτε μέσω του διαχειριστή πακέτων της διανομής μας είτε μέσω του pip) και να ελέγξουμε ότι το μηχάνημα-στόχος διαθέτει ένα διερμηνέα python 2 στο φάκελο /usr/bin/python.

Αφού εγκαταστήσουμε το εργαλείο στο τοπικό μας μηχάνημα, μπορούμε να εγκαταστήσουμε το ρόλο Dimpolissiou.openwisp2 από την ιστοσελίδα Galaxy με την εντολή:

```
ansible-galaxy install Dimpolissiou.openwisp2
```

Ο ρόλος αυτός περιέχει τις ενέργειες και τους κανόνες σύμφωνα με τους οποίους θα δράσει το ansible για την εγκατάσταση του εξυπηρετητή. Εάν θέλουμε να χρησιμοποιήσουμε πιστοποιητικό SSL της αρχής πιστοποίησης letsencrypt, μπορούμε να εγκαταστήσουμε επίσης το ρόλο thefinn93.letsencrypt για την αυτόματη διαχείρησή του (εγκατάσταση και ανανέωση).

Αυτό που δεν περιέχει ο ρόλος είναι οι δικές μας επιθυμητές ρυθμίσεις και επιλογές, όπως η διεύθυνση του εξυπηρετητή CAS και η χρήση ή όχι πρωτοκόλλου HTTPS. Για τις ρυθμίσεις αυτές θα πρέπει να δημιουργήσουμε το αντίστοιχο αρχείο playbook.yml σε ένα φάκελο της επιλογής μας (στο τοπικό μηχάνημα). Στον ίδιο φάκελο θα πρέπει να δημιουργήσουμε και ένα αρχείο καταλόγου απογραφής με όνομα hosts το οποίο θα περιέχει το

hostname του μηχανήματος στόχου. Ένα παράδειγμα αρχείων playbook και inventory είναι το παρακάτω:

```
hosts:
[openwisp2]
openwisp2.example.com

playbook.yml:
- hosts: openwisp2
  become: "{{ become | default('yes') }}"
  roles:
    - openwisp.openwisp2
  vars:
    openwisp2_shared_secret: changemeplease
    openwisp2_path: /opt/openwisp2
    openwisp2_database:
        engine: django.db.backends.postgresql
        name: openwisp2
        user: postgres
        password: ""
        host: ""
        port: ""
        options: {}
    openwisp2_time_zone: Europe/Athens
    openwisp2_cas_server: "http://127.0.0.1/cas/"
    openwisp2_cas_affiliation: "userPrincipalName"
    openwisp2_collectd_rrd_dir: "/var/lib/collectd/rrd/"
```

Το παράρτημα 3 περιγράφει όλες τις μεταβλητές που μπορούμε να συμπεριλάβουμε στο αρχείο playbook.yml.

Αφού ολοκληρώσουμε τις επιλογές μας, τρέχουμε το playbook με την εντολή

```
ansible-playbook -i hosts playbook.yml -u  <user> -k
--become -K
```

όπου στη θέση του <user> βάζουμε το όνομα ενός χρήστη του μηχανήματος στόχου. Το ansible θα συνδεθεί στο μηχάνημα με αυτό το χρήστη για να εκτελέσει τις εντολές. Ο χρήστης αυτός θα πρέπει να έχει δικαιώμα να εκτελεί το πρόγραμμα sudo στο μηχάνημα. Η επιλογή -k μας επιτρέπει να συνδεθούμε στο μηχάνημα με κωδικό πρόσβασης. Εάν εγκαταστήσουμε το δημόσιο SSH κλειδί μας στο μηχάνημα-στόχο, δε χρειαζόμαστε τις επιλογές -k, --become και -K.

6.1.3 Εγκατάσταση μέσω vagrant

Ένας ακόμα εύκολος τρόπος εγκατάστασης του συστήματος είναι μέσω του προγράμματος vagrant [4]. Το vagrant είναι ένα εργαλείο διαχείρισης εικονικών μηχανών. Μέσω του εργαλείου αυτού μπορούμε να κατεβάσουμε στον υπολογιστή μας μια εικονική μηχανή Ubuntu, η οποία έχει εγκατεστημένο τον OpenWISP2 controller, το collectd και τον εξυπηρετητή nginx, και να προωθήσουμε δικτυακά πακέτα με προορισμό τον υπολογιστή μας στην εικονική μηχανή. Με τον τρόπο αυτό δε χρειάζεται να ασχοληθούμε με τον τρόπο εγκατάστασης του OpenWISP2 controller, παρά μόνο με τις ρυθμίσεις που μας ενδιαφέρουν πραγματικά.

Αρχικά, για να χρησιμοποιήσουμε το εργαλείο Vagrant, θα πρέπει να το εγκαταστήσουμε στον υπολογιστή μας μαζί με το πρόγραμμα Virtualbox [5]. Στη συνέχεια, κατεβάζουμε την εικονική μας μηχανή με την εντολή `vagrant box add dimpolissiou/openwisp2`. Σε ένα φάκελο της επιλογής μας αρχικοποιούμε το περιβάλλον της εικονικής μηχανής με την εντολή `vagrant init openwisp2`. Θα χρειαστεί να επεξεργαστούμε το αρχείο Vagrantfile ώστε να προωθήσουμε τις θύρες 8080 tcp (openwisp2 controller) και 25826 (collectd server) της εικονικής μηχανής σε κάποιες θύρες του υπολογιστή μας. Για παράδειγμα, εάν το αρχείο Vagrantfile περιέχει τα παρακάτω, τότε η θύρα 8080 tcp του εικονικού μηχανήματος θα προωθηθεί στη θύρα 80 tcp του υπολογιστή μας και η θύρα 25826 udp του εικονικού μηχανήματος θα προωθηθεί στη θύρα 25826 udp του υπολογιστή μας.

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

Vagrant.configure("2") do |config|
  config.vm.box = "dimpolissiou/openwisp2"
  config.vm.network "forwarded_port", guest: 80, host: 8080
  config.vm.network "forwarded_port", guest: 25826, host: 25826,
protocol: "udp"
end
```

Αφού εκκινήσουμε την εικονική μηχανή με την εντολή `vagrant up`, μπορούμε να συνδεθούμε μέσω ssh (`vagrant ssh`) και αλλάξουμε της ρυθμίσεις του controller που βρίσκονται στο αρχείο `/opt/openwisp2/openwisp_monitor/settings.py`. Συγκεκριμένα, θα πρέπει σίγουρα να αλλάξουμε τις τιμές των πεδίων

1. `CAS_SERVER_URL`,
2. `AFFILIATION_FIELD`,
3. `SECRET` και
4. `NETJSONCONFIG_SHARED_SECRET` (παρ. 6.1.1).

Επίσης, εάν θέλουμε να χρησιμοποιήσουμε το collectd θα πρέπει να βεβαιωθούμε ότι το πεδίο `TIME_ZONE` είναι σύμφωνο με τις ρυθμίσεις των ενσωματωμένων δικτυακών συσκευών μας. Τέλος, εάν θέλουμε να χρησιμοποιήσουμε το σύστημα στην παραγωγή, θα πρέπει να δώσουμε τιμή `False` στο πεδίο `DEBUG`. Αφού κάνουμε τις αλλαγές αυτές,

επανεκκινούμε τον controller με την εντολή `sudo supervisorctl restart openwisp2`.

Εάν θέλουμε να αλλάξουμε τις ρυθμίσεις του nginx, π.χ. για να χρησιμοποιήσουμε πρωτόκολλο HTTPS, μπορούμε να τροποποιήσουμε το αρχείο /etc/nginx/sites-enabled/openwisp2.conf. Ύστερα, επιβεβαιώνουμε ότι δεν υπάρχουν σφάλματα στο αρχείο αυτό με την εντολή `nginx -t` και επανεκκινούμε τον nginx με την εντολή
`sudo systemctl restart nginx`.

6.2 Εγκατάσταση του πράκτορα σε συσκευές OpenWrt

Αφού εγκαταστήσουμε τον OpenWISP2 controller, μπορούμε να συνδέσουμε τις OpenWrt συσκευές μας με τον εξυπηρετητή ώστε να τις διαχειριζόμαστε μέσα από το OpenWISP. Σε κάθε συσκευή που θέλουμε να διαχειριστούμε, θα πρέπει να εγκαταστήσουμε και να ρυθμίσουμε τον πράκτορα openwisp-config. Ο πράκτορας αυτός επικοινωνεί περιοδικά με τον εξυπηρετητή μας, εντοπίζει αλλαγές στην επιθυμητή διάρθρωση και εφαρμόζει τις αλλαγές μέσω του συστήματος UCI. Επίσης, μπορούμε να εγκαταστήσουμε τα πακέτα collectd, στην περίπτωση που θέλουμε να χρησιμοποιήσουμε τη λειτουργία παρακολούθησης, και luci-openwisp, το οποίο αφαιρεί επιλογές από τη διεπαφή του luci ώστε να αποφεύγονται οι αλλαγές στη διάρθρωση του μηχανήματος έξω από το σύστημα OpenWISP. Η εγκατάσταση αυτών τον πακέτων μπορεί να γίνει μέσω του διαχειριστή opkg. Εναλλακτικά, μπορούμε να δημιουργήσουμε μια εικόνα OpenWrt που να περιέχει αυτά τα πακέτα και τα αντίστοιχα αρχεία ρύθμισης.

6.2.1 Εγκατάσταση μέσω opkg

Πακέτο openwisp-config

Για να εγκαταστήσουμε το πακέτο openwisp-config μέσω του opkg, μπορούμε είτε να το μεταφράσουμε από τον πηγαίο κώδικα που βρίσκεται στη σελίδα https://github.com/openwisp/openwisp-config, είτε να χρησιμοποιήσουμε ένα μεταφρασμένο πακέτο από τη σελίδα http://downloads.openwisp.org/openwisp-config/. Αφού σημειώσουμε το URL του πακέτου που θέλουμε να χρησιμοποιήσουμε, εκτελούμε στο OpenWrt μηχάνημα τις εντολές:

```
opkg update
opkg install <URL>
```

Στη συνέχεια, ανοίγουμε το αρχείο /etc/config/openwisp, και αλλάζουμε τις εξής ρυθμίσεις:

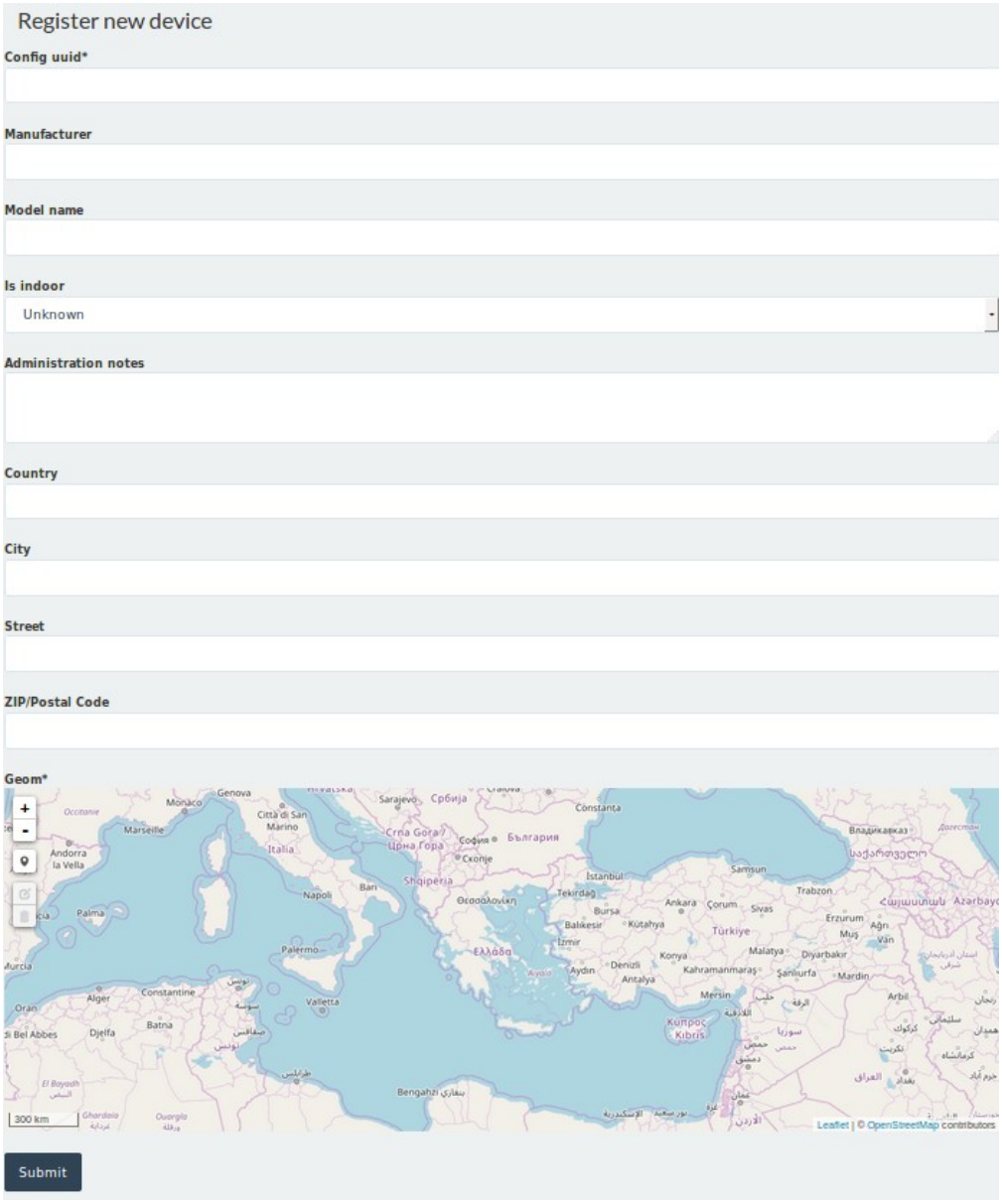
- url : Εδώ γράφουμε το URL του εξυπηρετητή OpenWISP2 controller με τον οποίο θα επικοινωνεί ο πράκτορας
- interval: Η τιμή αυτή είναι ο χρόνος σε δευτερόλεπτα μεταξύ των ελέγχων για αλλαγή στην επιθυμητή διάρθρωση
- verify_ssl: Βάζουμε τιμή 0 εάν δε θέλουμε να γίνεται έλεγχος SSL κατά την επικοινωνία με τον εξυπηρετητή

- shared_secret: Για την αυτόματη εγγραφή του μηχανήματος στον εξυπηρετητή, πρέπει να σημειώσουμε εδώ το μυστικό κωδικό που δηλώσαμε στο αρχείο settings.py του εξυπηρετητή, ή στην επιλογή openwisp2_shared_secret του αρχείου playbook.yml

Το παράρτημα 4 περιγράφει όλες τις επιλογές του αρχείου /etc/config/openwisp .

Τέλος, εκκινούμε τον πράκτορα με την εντολή
`/etc/init.d/openwisp_config start`

Εάν ανοίξουμε πάλι το αρχείο /etc/config/openwisp, θα πρέπει να δούμε τις τιμές uuid και key να έχουν αλλάξει. Σημειώνουμε την τιμή uuid όπως εμφανίζεται στο αρχείο. Συνδεόμαστε μέσω CAS στον εξυπηρετητή και δημιουργούμε μια νέα συσκευή από το μενού ενεργειών. Στη φόρμα δημιουργίας συσκευής εισάγουμε την τιμή uuid που σημειώσαμε παραπάνω και όλες τις πληροφορίες της συσκευής, όπως την τοποθεσία της.

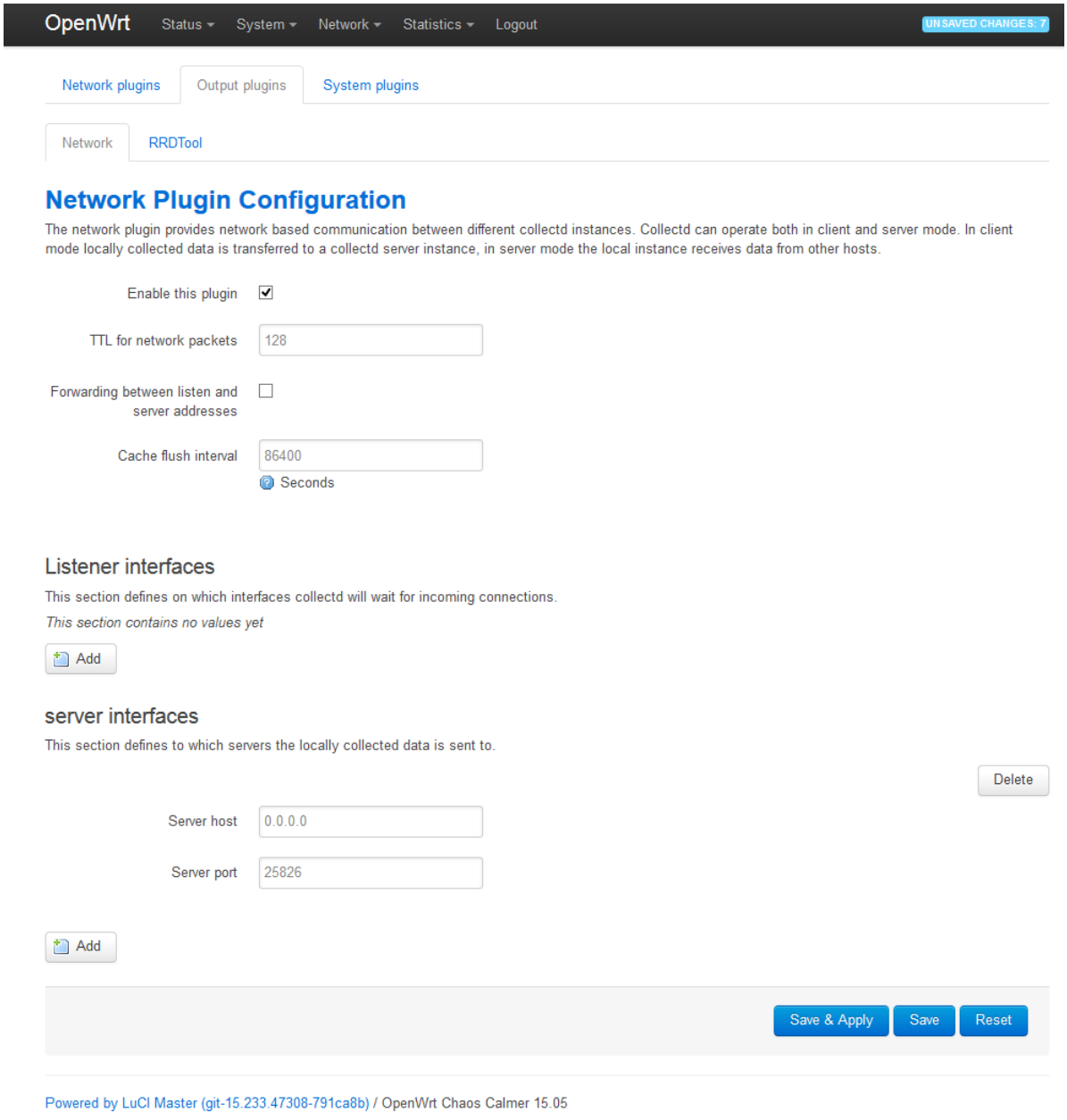


Πακέτο collectd

Ο ευκολότερος τρόπος να εγκαταστήσουμε και να ρυθμίσουμε το δαίμονα collectd είναι μέσω του πακέτου luci-app-statistics. Έχοντας εγκαταστήσει το πακέτο luci, εκτελούμε τις εντολές:

```
opkg update
opkg install luci-app-statistics
```

Στη συνέχεια, ρυθμίζουμε το δαίμονα collectd μέσω της διεπαφής ιστού luci. Φροντίζουμε να ενεργοποιήσουμε το άρθρωμα δικτύωσης και να εισάγουμε την διεύθυνση του εξυπηρετητή μας.



Εναλλακτικά, εγκαθιστούμε το πακέτο collectd, μαζί με τα πακέτα των αρθρωμάτων που θέλουμε και εισάγουμε τις ρυθμίσεις μας στο αρχείο /etc/collectd.conf :

```
opkg update
opkg install collectd collectd-mod-cpu collectd-mod-
interface collectd-mod-memory collectd-mod-load collectd-
mod-network collectd-mod-rrdtool collectd-mod-wireless
```

Ένα παράδειγμα αρχείου ρυθμίσεων collectd.conf είναι το παρακάτω:

```
BaseDir "/var/run/collectd"
Include "/etc/collectd/conf.d"
PIDFile "/var/run/collectd.pid"
PluginDir "/usr/lib/collectd"
TypesDB "/usr/share/collectd/types.db"
Interval 30
ReadThreads 2
Hostname "2884a082-65a5-4fa2-8f81-2a42be402ffa" # εδώ
βάζουμε την τιμή uuid του αρχείου /etc/config/openwisp

LoadPlugin splash_leases

LoadPlugin conntrack

LoadPlugin rrdtool
<Plugin rrdtool>
   DataDir "/tmp/rrd"
   RRARows 100
   RRASingle true
   RRATimespan 3600
   RRATimespan 86400
   RRATimespan 604800
   RRATimespan 2678400
   RRATimespan 31622400
</Plugin>

LoadPlugin iwinfo
<Plugin iwinfo>
   IgnoreSelected false
</Plugin>

LoadPlugin processes
<Plugin processes>
   Process uhttpd
   Process dnsmasq
   Process dropbear
</Plugin>

LoadPlugin interface
<Plugin interface>
   IgnoreSelected true
```

```
    Interface lo
</Plugin>

LoadPlugin network
<Plugin network>
    Server "192.168.52.2" "25826"
    Forward false
</Plugin>

LoadPlugin memory

LoadPlugin cpu

LoadPlugin load
```

Τέλος, δοκιμάζουμε τις ρυθμίσεις μας με την εντολή collectd -t και επανεκκινούμε το δαίμονα με την εντολή:

```
/etc/init.d/collectd restart
```

Οποιαδήποτε από τις δύο μεθόδους και να χρησιμοποιήσουμε, είναι σημαντικό τα μηχανήματα να χρησιμοποιούν την ίδια ζώνη ώρας (ή και τον ίδιους εξυπηρετητές NTP) με τον OpenWISP2 controller.

Πακέτο luci-openwisp

Η εγκατάσταση του πακέτου luci-openwisp είναι ευκολότερη. Εκτελούμε απλά τις εντολές

```
opkg update
opkg install http://downloads.openwisp.org/luci-
openwisp/latest/luci-mod-openwisp_0.1.2-1_all.ipk
opkg install http://downloads.openwisp.org/luci-
openwisp/latest/luci-theme-openwisp_0.1.2-1_all.ipk
```

Συνδεόμαστε στη διεπαφή luci με όνομα χρήστη `operator` και κωδικό `password`. Μπορούμε να αλλάξουμε τον κωδικό μέσω της εντολής openwisp-passwd:

```
openwisp-passwd
Changing password for luci-mod-openwisp, username: operator
New password:
secret
Retype password:
secret
luci-mod-openwisp password for user operator changed
successfully
```

6.2.2 Δημιουργία εικόνας OpenWrt με προεγκατεστημένο πράκτορα

Δημιουργία εικόνας μέσω του OpenWrt build system

Για να δημιουργήσουμε εικόνες OpenWrt με προεγκατεστημένο πράκτορα OpenWISP, μπορούμε να χρησιμοποιήσουμε το OpenWrt build system, μέσω της διαδικασίας που περιγράψαμε στην παράγραφο 3.2.2. Αφού κατεβάσουμε το build system, προσθέτουμε στα feeds την πηγή του πακέτου openwisp-config με την παρακάτω εντολή:

```
echo "src-git openwisp https://github.com/openwisp/openwisp-
config.git" >> feeds.conf
```

Στη συνέχεια, κατεβάζουμε των κώδικα των πακέτων με τις εντολές:

```
./scripts/feeds update -a
./scripts/feeds install -a
```

Από το γραφικό μενού (εντολή menuconfig) επιλέγουμε την αρχιτεκτονική των συσκευών μας καθώς και ποια πακέτα θα τους εγκαταστήσουμε. Φροντίζουμε να επιλέξουμε το πακέτο openwisp-config και, εάν το επιθυμούμε, το πακέτο collectd με τα αρθρώματα network, rrdtool, cpu, memory, load και interface. Στο φάκελο files τοποθετούμε το αρχείο collectd.conf και στο φάκελο ./files/etc/config/ δημιουργούμε ένα αρχείο με όνομα openwisp και περιεχόμενο παρόμοιο με το παρακάτω:

```
config controller 'http'
    option url 'https://controller.example.com'
    option shared_secret 'mysharedsecret'
    list unmanaged 'system.@led'
    list unmanaged 'network.loopback'
    list unmanaged 'network.@switch'
    list unmanaged 'network.@switch_vlan'
```

Τέλος, δημιουργούμε τις εικόνες εκτελώντας μια εντολή όπως `make` ή `ionice -c 3 nice -n19 make -j 2`. Οι εικόνες μας θα περιέχουν τα πακέτα που ορίσαμε και μόλις μεταφορτωθούν σε κάποια συσκευή, η συσκευή αυτή
θα επιχειρήσει αυτόματη εγγραφή στον εξυπηρετητή μας.

Δημιουργία εικόνας μέσω του OpenWISP2 image generator

Ο δεύτερος τρόπος δημιουργίας εικόνων OpenWrt με προεγκατεστημένο πράκτορα είναι μέσω του OpenWISP2 image generator. Ο image generator μας επιτρέπει να χειριστούμε το περιβάλλον δημιουργίας εικόνων OpenWrt μέσω του εργαλείου ansible. Για να χρησιμοποιήσουμε τον image generator, αρχικά εγκαθιστούμε το εργαλείο ansible όπως περιγράψαμε στην παράγραφο 6.1.2 . Στη συνέχεια, εγκαθιστούμε μέσω του ansible-galaxy το ρόλο openwisp.openwisp2-imagegenerator. Έπειτα, σε ένα φάκελο της επιλογής μας δημιουργούμε τα αρχεία hosts και playbook.yml . Υποθέτοντας ότι η κατασκευή των εικόνων (μια διαδικασία ιδιαίτερα χρονοβόρα) θέλουμε να γίνει στο μηχάνημα μεταγλώττισης με διεύθυνση compiler.mydomain.com, το περιεχόμενο του αρχείου hosts μπορεί να είναι το παρακάτω:

```
[server]
compiler.mydomain.com ansible_user=<user>
ansible_become_pass=<sudo-password>
```

Ο χρήστης <user> πρέπει να μην είναι ο root και να έχει δικαίωμα να εκτελεί το πρόγραμμα sudo. Στο αρχείο playbook.yml τοποθετούμε όλες τις ρυθμίσεις μας. Ένα παράδειγμα αρχείου playbook είναι το παρακάτω:

```
- hosts: server
 roles:
 - openwisp.openwisp2-imagegenerator
 vars:
 openwisp2fw_source_dir: /home/user/openwisp2-firmware-source
 openwisp2fw_generator_dir: /home/user/openwisp2-firmware-generator
 openwisp2fw_bin_dir: /home/user/openwisp2-firmware-builds
 openwisp2fw_source_repo: git://git.openwrt.org/15.05/openwrt.git
 openwisp2fw_source_version: "HEAD"
 openwisp2fw_source_default_packages:
 - openwisp-config-{{ openwisp2fw_ssl_lib }}
 - luci-openwisp-{{ openwisp2fw_ssl_lib }}
 - openvpn-{{ openwisp2fw_ssl_lib }}
 - collectd
 - collectd-mod-cpu
 - collectd-mod-df
 - collectd-mod-disk
 - collectd-mod-interface
 - collectd-mod-load
 - collectd-mod-memory
 - collectd-mod-network
 - collectd-mod-processes
 - collectd-mod-rrdtool
 openwisp2fw_source_targets:
 - system: x86
          subtarget: generic
          profile: Generic
    openwisp2fw_organizations:
        - name: sch # name of the org
```

```
    flavours: # supported flavours
      - standard
    luci_openwisp: # /etc/config/luci_openwisp
      # other config keys can be added freely
      username: "root"
      # clear text password that will be encrypted in
      # /etc/config/luci_openwisp
      password: "rootpassword"
    openwisp: # /etc/config/openwisp
      # other config keys can be added freely
      url: "https://controller.example.com"
      shared_secret: "mysharedsecret"
      unmanaged: "{{ openwisp2fw_default_unmanaged }}"
    # clear text password that will be encrypted in /etc/shadow
      root_password: "rootpassword"
```

Όπως παρατηρούμε, στο αρχείο playbook μπορούμε να ορίσουμε πολλαπλά προφίλ, όπως για παράδειγμα ένα κανονικό προφίλ για τις περισσότερες συσκευές και ένα προφίλ που περιέχει μόνο τα απολύτως απαραίτητα πακέτα για συσκευές με πολύ περιορισμένο χώρο. Επίσης, μπορούμε να ορίσουμε πολλούς οργανισμούς, με διαφορετικές ρυθμίσεις για τον καθένα. Ο image generator θα δημιουργήσει μια εικόνα OpenWrt για κάθε συνδυασμό οργανισμού, προφίλ συσκευής και αρχιτεκτονικής που έχουμε ορίσει.

Τα αρχεία ρύθμισης των πακέτων που έχουμε προετοιμάσει, μπορούμε να τα τοποθετήσουμε μέσα σε ένα φάκελο με όνομα files, ο οποίος πρέπει να βρίσκεται στον ίδιο φάκελο με τα αρχεία hosts και playbook.yml. Για παράδειγμα, μπορούμε να τοποθετήσουμε ένα αρχείο στο φάκελο files/etc/collectd.conf . Το αρχείο αυτό θα βρίσκεται σε κάθει εικόνα που θα κατασκευάσει ο image generator, στη θέση /etc/collectd.conf .

Εκτελούμε το playbook με την εντολή:

```
ansible-playbook -i hosts playbook.yml -e "recompile=1
cores=4"
```

Στη θέση του cores=4 μπορούμε να βάλουμε τον αριθμό πυρήνων που έχουμε στη διάθεσή μας. Όταν τελειώσει η εκτέλεση, μπορούμε να βρούμε τις εικόνες που δημιουργήσαμε στο μηχάνημα μεταγλώττισης, στο φάκελο που ορίζει η μεταβλητή openwisp2fw_bin_dir του playbook.

Κεφάλαιο 7

Συμπεράσματα

7.1 Σύνοψη

Στην παρούσα διπλωματική εργασία αναπτύχθηκε μια διεπαφή ιστού μέσω της οποίας μπορούμε διαχειριστούμε ένα μεγάλο αριθμό ασύρματων σημείων πρόσβασης εξοπλισμένα με το λειτουργικό σύστημα OpenWrt. Η εφαρμογή μας υποστηρίζει την ύπαρξη πολλαπλών διαχειριστικών ομάδων, καθεμία από τις οποίες έχει τον έλεγχο ενός υποσυνόλου των σημείων πρόσβασης του δικτύου. Δόθηκε προσοχή στον έλεγχο των δικαιωμάτων των διαχειριστικών ομάδων, ώστε κανένας χρήστης να μην μπορεί να επηρεάσει τη διάρθρωση των σημείων πρόσβασης που δεν ανήκουν στην ομάδα του.

Με σκοπό τη διευκόλυνση των διαχειριστών στο έργο τους, η εφαρμογή μας διαθέτει έναν αριθμό λειτουργιών που βοηθούν στην οργάνωση και την αυτοματοποίηση της διαχείρισης, αποτρέποντας έτσι πολλά από τα σφάλματα που προκύπτουν από τη χειροκίνητη διαχείριση μεγάλου πλήθους συσκευών. Μία τέτοια λειτουργία είναι η αναζήτηση συσκευών με βάση κάποια χαρακτηριστικά τους όπως το μοντέλο τους ή η τοποθεσία τους. Η λειτουργία αυτή ενισχύεται από τη δυνατότητα δημιουργίας προτύπων διάρθρωσης που περιλαμβάνουν κάποιες κοινές ρυθμίσεις που θέλουμε να εφαρμόσουμε σε πολλά μηχανήματα. Με τον τρόπο αυτό, η ενεργοποίηση πχ. ενός προγράμματος VPN σε μια ομάδα μηχανημάτων μπορεί να γίνει εύκολα με λίγες κινήσεις.

Επιπλέον, η εφαρμογή διαθέτει και ορισμένες λειτουργίες που αφορούν την επιτήρηση του δικτύου των σημείων πρόσβασης και της λειτουργίας τους με οπτικά μέσα. Μέσω του δαίμονα collectd, ο κεντρικός εξυπηρετητής συλλέγει στατιστικά από τα σημεία πρόσβασης, που αφορούν τη χρήση του επεξεργαστή, της μνήμης και της δικτυακής κίνησης, και τα παρουσιάζει σε μορφή γραφημάτων με χρήση του εργαλείου RRDtool. Τέλος, χρησιμοποιώντας τους χάρτες του OpenStreetMap, η εφαρμογή παρουσιάζει μια γεωγραφική απεικόνιση του δικτύου τοποθετώντας ένα διαδραστικό σημάδι στη θέση κάθε συσκευής.

Η εφαρμογή μπορεί να εγκατασταθεί εύκολα μέσω του εργαλείου ansible με χρήση του αντίστοιχου ρόλου που έχουμε ανεβάσει στην ιστοσελίδα Ansible Galaxy [7.1]. Επίσης, μέσω του προγράμματος vagrant μπορεί κάποιος να κατεβάσει μια εικονική μηχανή η οποία έχει ήδη εγκατεστημένο τον εξυπηρετητή της εφαρμογής μας.

7.2 Προϋποθέσεις χρήσης

Βασική προϋπόθεση για τη χρήση της εφαρμογής είναι η χρήση μηχανημάτων που υποστηρίζουν το λειτουργικό σύστημα OpenWrt. Το λειτουργικό αυτό σύστημα είναι σχεδιασμένο για ενσωματωμένες δικτυακές συσκευές και έχει δοκιμαστεί σε μεγάλο και διαρκώς αυξανόμενο πλήθος μηχανημάτων. Παρόλα αυτά, υπάρχουν μερικές συσκευές οι οποίες δεν το υποστηρίζουν, πιθανότατα επειδή οι πόροι τους είναι εξαιρετικά περιορισμένοι (λιγότερο από 4MB μνήμης flash ή λιγότερο από 32MB μνήμης RAM). Για να επιβεβαιώσουμε ότι οι συσκευές μας υποστηρίζουν το λειτουργικό σύστημα OpenWrt μπορούμε να ανατρέξουμε στην αντίστοιχη ιστοσελίδα [7.2]

Μία ακόμα προϋπόθεση για τη χρήση του συστήματος είναι η χρήση του πρωτοκόλλου CAS 3 για την αυθεντικοποίηση των διαχειριστών. Επομένως, θα πρέπει να διαθέτουμε έναν εξυπηρετητή αυθεντικοποίησης που να υποστηρίζει το πρωτόκολλο αυτό. Στην αντίθετη περίπτωση, θα πρέπει να εγκαταστήσουμε έναν εξυπηρετητή CAS 3 πιθανώς στο ίδιο μηχάνημα με τον εξυπηρετητή της εφαρμογής μας. Θα χρειαστούμε όμως μια βάση δεδομένων ή ένα κατάλογο τύπου LDAP που να περιέχει τα στοιχεία των διαχειριστών.

7.3 Πιθανές εφαρμογές

Καθώς η υλοποίησή μας επέκτεινε την πλατφόρμα OpenWISP με την προσθήκη διαχειριστικών ομάδων, θα μπορούσε να βρει εφαρμογή σε οποιοδήποτε δίκτυο είναι αρκετά μεγάλο ώστε να απαιτεί καταμερισμό στη διαχείρισή του. Ένα παράδειγμα θα ήταν το μητροπολιτικό δίκτυο μιας πόλης στο οποίο κάθε δήμος είναι υπεύθυνος για τη διαχείριση των δικών του σημείων πρόσβασης ή ενός μεγάλου πανεπιστημιακού campus όπου η διαχείριση των σημείων πρόσβασης θα ήταν καταμερισμένη στις σχολές.

Αξίζει να σημειωθεί ότι το μόνο που απαιτεί η εφαρμογή για τη διαχείριση των σημείων πρόσβασης είναι η δυνατότητα επικοινωνίας με τον εξυπηρετητή μέσω HTTP. Επομένως, τα σημεία πρόσβασης δεν είναι απαραίτητο να βρίσκονται στο ίδιο υποδίκτυο ή στον ίδιο χώρο. Για το λόγο αυτό, η εφαρμογή θα μπορούσε να χρησιμοποιηθεί για τη διαχείριση σημείων πρόσβασης που βρίσκονται σε μεγάλη απόσταση μεταξύ τους, όπως σχολεία, υποκαταστήματα ενός οργανισμού, κέντρα εξυπηρέτησης πολιτών ή ακόμα και ένα στόλο λεωφορείων. Τα σημεία πρόσβασης θα μπορούσαν να βρίσκονται ακόμα και σε γεωγραφικά απομονωμένες περιοχές της χώρας, επιτρέποντας έτσι την απομακρυσμένη κεντρική διαχείρισή τους.

Τέλος, η εφαρμογή θα μπορούσε να βρει χρήση στα κοινοτικά ασύρματα δίκτυα στα οποία δεν υπάρχει κεντρική διαχείριση των κόμβων αλλά ιδιώτες ή οργανισμοί μπορούν να συμμετάσχουν ως κομβούχοι, αναλαμβάνοντας τη διαχείριση του εξοπλισμού τους. Στα δίκτυα αυτά η εφαρμογή θα επέτρεπε την ομοιόμορφη διαχείριση των σημείων πρόσβασης (ή και άλλων δικτυακών συσκευών), επιτρέποντας παράλληλα σε κάθε κομβούχο να διαχειρίζεται ο ίδιος τα μηχανήματά του.

7.4 Μελλοντικές επεκτάσεις

Η πιο σημαντική μελλοντική επέκταση της εφαρμογής θα ήταν η υποστήριξη περισσοτέρων πρωτοκόλλων αυθεντικοποίησησς, όπως τα πρωτόκολλα SAML και Oauth. Η προϋπόθεση χρήσης του πρωτοκόολου CAS δυσχεραίνει την εγκατάσταση της εφαρμογής σε περιβάλλοντα όπου ήδη χρησιμοποιείται κάποιος εξυπηρετητής κεντρικής αυθεντικοποποίησης ο οποίος δεν υποστηρίζει αυτό το πρωτόκολλο. Εάν μπορούσαμε να άρουμε τον περιορισμό αυτό, τότε θα μπορούσαμε να ενσωματώσουμε την εφαρμογή μας σε οποιοδήποτε περιβάλλον με ελάχιστη παραμετροποίηση.

Μελλοντικά θεωρούμε ότι θα πρέπει να γίνει επικαιροποίηση του κώδικα της εφαρμογής ώστε να είναι συμβατός με τις τελευταίες εκδώσεις των βιβλιοθηκών Django και netjsonconfig. Το πλαίσιο ανάπτυξης λογισμικού Django κατά καιρούς δημοσιεύει ενημερώσεις ασφαλείας οι οποίες είναι καίριας σημασίας για την προστασία των χρηστών της εφαρμογής. Επίσης, η βιβλιοθήκη netjsonconfig αναπτύσσεται με εντατικούς ρυθμούς, προσθέτοντας διαρκώς νέες λειτουργίες που διευκολύνουν τη διαχείριση των δικτυακών συσκευών.

Ενδιαφέρον θα είχε επίσης η επέκταση της εφαρμογής ώστε να ενσωματώνει κάποια ώριμη λύση οπτικοποίησης των μετρικών που συλλέγουμε μέσω του collectd, όπως πχ. ένα ταμπλό Grafana [7.3]. Τέλος, υπάρχει ακόμα χώρος για βελτίωση των λειτουργιών που επιτρέπουν την εύκολη εφαρμογή κοινών ρυθμίσεων σε μεγάλο αριθμό συσκευών όπως για παράδειγμα η χρήση ετικετών για την ομαδοποίηση τόσο των συσκευών όσο και των προτύπων διάρθρωσης.

Βιβλιογραφία

[1.1]Shin, Seungjae and Tucci, Jack E., "Lesson from WiFi Municipal Wireless Network" (2009). *AMCIS 2009 Proceedings*. 145.
https://aisel.aisnet.org/amcis2009/145

[2.1] ISO/IEC 10040, 1998, "Information technology – Open Systems Interconnection – Systems management overview"

[2.2] R. Mahajan, D. Wetherall, and T. Anderson, “Understanding BGP misconfigurations”, in Proc. ACM SIGCOMM, Aug. 2002.

[2.3] D. Oppenheimer, A. Ganapathi, and D. Patterson, “Why do Internet services fail, and what can be done about it?” in Proc. USITS, 2003.

[2.4] A. Wool, “A quantitative study of firewall configuration errors”,
IEEE Computer, June 2004.

[2.5] A. C. Popescu, B. J. Premore, and T. Underwood,
“Anatomy of a leak: AS9121”, NANOG 34, May 2005.

[2.6] Urs Hoelzle and Luiz Andre Barroso.
“The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines.” Morgan and Claypool Publishers, 1st edition, 2009.

[2.7] Tianyin Xu, Jiaqi Zhang, Peng Huang, Jing Zheng, Tianwei Sheng, Ding Yuan, Yuanyuan Zhou, and Shankar Pasupathy.
“Do not blame users for misconfigurations.” In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, pages 244–259. ACM, 2013.

[3.1] https://en.wikipedia.org/wiki/OpenWrt

[3.2] https://wiki.openwrt.org/doc/howto/buildroot.exigence

[3.3] https://wiki.openwrt.org/doc/howto/generic.flashing

[3.4] https://wiki.openwrt.org/doc/uci

[4.1] https://media.readthedocs.org/pdf/openwisp-firmware/latest/openwisp-firmware.pdf

[4.2] http://nemesisdesign.net/blog/coding/netjsonconfig-convert-netjson-to-openwrt-uci/

[4.3] https://www.pycon.it/conference/talks/applying-the-unix-philosophy-to-django-projects-a-report-from-the-real-world

[4.4] https://edoput.github.io/openwispgsoc/about/

[4.5] http://netjsonconfig.openwisp.org/en/raspbian/backends/raspbian.html

[4.6] https://archive.fosdem.org/2017/schedule/event/openwisp2/

[4.7] http://netjson.org/docs/

[4.8] http://netjson.org/rfc.html

[4.9] https://github.com/AdvancedNetworkingSystems/poprouting

[4.10] https://github.com/netjson/netjsongraph.js

[4.11] https://github.com/netjson/django-netjsongraph

[4.12] http://netjsonconfig.openwisp.org/en/stable/general/basics.html

[5.1] https://django-guardian.readthedocs.io/en/stable/

[5.2] https://collectd.org/features.shtml

[5.3] https://collectd.org/wiki/index.php/Inside_the_RRDtool_plugin

[5.4] https://tools.ietf.org/html/rfc7946

[6.1] https://bootstrap.pypa.io/get-pip.py

[6.2] https://docs.djangoproject.com/en/2.0/ref/settings/#databases

[6.3] http://legacy.python.org/dev/peps/pep-3333/

[6.4] https://www.vagrantup.com/

[6.5] https://www.virtualbox.org/

[7.1] https://galaxy.ansible.com/

[7.2] https://openwrt.org/toh/start

[7.3] https://grafana.com/

Παράρτημα 1

Πηγαίος κώδικας της εφαρμογής

Ο κώδικας της εφαρμογής βρίσκεται στη σελίδα
https://github.com/DimPolissiou/django-devices
Ο φάκελος που περιέχει τον κώδικα έχει την εξής μορφή:

```
src
|   requirements.txt
|   setup.py
+---devices
|   |   actions.py
|   |   admin.py
|   |   apps.py
|   |   forms.py
|   |   graphs.py
|   |   models.py
|   |   signals.py
|   |   tests.py
|   |   urls.py
|   |   views.py
|   |   __init__.py
|   +---migrations
|   +---static
|   |   \---devices
|   |       +---css
|   |       \---js
|   |                graphs.js
|   \---templates
\---django_actions
```

- **src/devices/actions.py:**

```
from django.utils.translation import ugettext as _
from django.http import HttpResponseRedirect, HttpResponseBadRequest
from django.urls import reverse
from django_netjsonconfig.models import Config
from .models import Device
from .graphs import make_graphs
import pickle

def select_devices_action(view, queryset):
        view.request.session['devices_queryset'] =
pickle.dumps(queryset.query)
        return HttpResponseRedirect(reverse("devices_apply_template"))

def rebuild_graphs_action(view, queryset):
    for device in queryset.iterator():
        make_graphs(device)
    return HttpResponseRedirect(reverse("devices_home"))

def apply_template_action(view, queryset):
    if 'devices_queryset' not in view.request.session:
        return HttpResponseBadRequest("Session is missing the
devices_queryset key.")
    devices_queryset = Device.objects.all()[:1]
    devices_queryset.query =
pickle.loads(view.request.session.get('devices_queryset'))
    templates_queryset = queryset
    for device in devices_queryset.iterator():
        config = Config.objects.get(id=device.config.id)
        for template in templates_queryset.iterator():
            config.templates.add(template)
            config.save()
    return HttpResponseRedirect(reverse("devices_home"))

select_devices_action.short_description = _('Apply templates to
selected devices')
apply_template_action.short_description = _('Apply selected
templates')
rebuild_graphs_action.short_description = _('Rebuild graphs (fix
missing graphs)')
```

- **src/devices/admin.py:**

```
from leaflet.admin import LeafletGeoAdmin
from django.contrib import admin
from . import models as device_models
from django_netjsonconfig.admin import ConfigAdmin, TemplateAdmin
from django_netjsonconfig.models import Config, Template
from guardian.admin import GuardedModelAdminMixin
from guardian.shortcuts import get_objects_for_user
from rules.contrib.admin import ObjectPermissionsModelAdminMixin
from django.core.urlresolvers import reverse
from django.http import HttpResponseRedirect

class DeviceAdmin(LeafletGeoAdmin):
    list_display = [field.name for field in
device_models.Device._meta.fields if field.name != "id" and field.name
!= "geom"]

class GuardedConfigAdmin(GuardedModelAdminMixin,
ObjectPermissionsModelAdminMixin, ConfigAdmin):

    def response_add(self, request, obj, post_url_continue="../%s/"):
        if not '_continue' in request.POST:
            return HttpResponseRedirect(reverse('devices_home'))
        else:
            return super(GuardedConfigAdmin,
self).response_add(request, obj, post_url_continue)

    def response_change(self, request, obj):
        if not '_continue' in request.POST:
            return HttpResponseRedirect(reverse('devices_home'))
        else:
            return super(GuardedConfigAdmin,
self).response_change(request, obj)

class GuardedTemplateAdmin(GuardedModelAdminMixin,
ObjectPermissionsModelAdminMixin, TemplateAdmin):
    list_display = ('name', 'type', 'backend', 'created', 'modified')
    list_filter = ('backend', 'type', 'created',)
    fields = ['name',
                'type',
                'backend',
                'vpn',
                'auto_cert',
                'config',
                'created',
        'modified']
```

```
    def save_model(self, request, obj, form, change):
        user_group = request.user.groups.all().first()
        assign_perm('django_netjsonconfig.change_template',
user_group, obj)
        assign_perm('django_netjsonconfig.delete_template',
user_group, obj)
        return super(GuardedTemplateAdmin, self).save_model(request,
obj, form, change)

admin.site.unregister(Config)
admin.site.register(Config, GuardedConfigAdmin)

admin.site.unregister(Template)
admin.site.register(Template, GuardedTemplateAdmin)

admin.site.register(device_models.Device, DeviceAdmin)
```

- **src/devices/apps.py:**

```
from django.apps import AppConfig


class DevicesConfig(AppConfig):
    name = 'devices'
```

- **src/devices/froms.py:**

```
from django.forms import Form, ModelForm, CharField,
UUIDField, Textarea
from django.db import models
#from leaflet.admin import LeafletGeoAdminMixin
from leaflet.forms.fields import PointField
from django.utils.translation import ugettext as _
from django.core.exceptions import ValidationError
from django.forms.widgets import NullBooleanSelect
from crispy_forms.helper import FormHelper
from crispy_forms.layout import Submit, Button
from leaflet.forms.widgets import LeafletWidget

from django_netjsonconfig.models import Config

from .models import Device

class CustomizedLeafletWidget(LeafletWidget):
    geom_type = 'POINT'
    map_template = 'leaflet/admin/widget.html'
    map_width = '100%'
    map_height = '400px'
    display_raw = False
    include_media = True
    settings_overrides = {}
```

```
class DeviceUpdateForm(ModelForm):

    def __init__(self, *args, **kwargs):
        super(DeviceUpdateForm, self).__init__(*args, **kwargs)
        self.helper = FormHelper()
        self.helper.form_method = 'post'
        self.helper.form_class = 'form-horizontal'
        self.helper.add_input(Submit('submit', 'Submit',
css_class='btn-primary'))
        self.helper.add_input(Button('delete', 'Delete',
css_class='btn btn-danger',
onclick='window.location.href="{}"'.format('delete')))

    class Meta:
        model = Device
        exclude = ['config', 'owner']
        widgets = {'geom' : CustomizedLeafletWidget(), 'notes' :
Textarea(attrs={'rows': 2, 'cols': 40})}

class DeviceRegisterForm(DeviceUpdateForm):
    config_uuid = UUIDField()
    field_order = ['config_uuid']

    def clean_config_uuid(self):
        config_uuid = self.cleaned_data['config_uuid']
        if Config.objects.filter(id=config_uuid).exists() == False:
            raise ValidationError('UUID does not correspond to a valid
device configuration')
        return config_uuid

    def __init__(self, *args, **kwargs):
        super(DeviceRegisterForm, self).__init__(*args, **kwargs)
        self.helper = FormHelper()
        self.helper.form_method = 'post'
        self.helper.form_class = 'form-horizontal'
        self.helper.add_input(Submit('submit', 'Submit',
css_class='btn-primary'))

class DeviceSearchForm(ModelForm):

    class Meta:
        model = Device
        exclude = ['config', 'notes', 'is_indoor', 'geom', 'owner']
```

- **src/devices/graphs.py:**

```
import os
from django.conf import settings
from collectd_rest.models import Graph, GraphGroup
from .models import Device, GraphManager

if hasattr(settings, 'COLLECTD_RRD_DIR'):
```

```
    RRD_DIR = settings.COLLECTD_RRD_DIR
else:
    RRD_DIR = "/var/lib/collectd/rrd/"

def make_cpu_graph(cpurrd, group, core):
    cpu_args = [
        '-s end-1h',
    #   '-e 1303382682',
        '-w 600' ,
        '-h 240' ,
        '-t cpu' ,
        '-v Percent' ,
        '-E' ,
        '-r' ,
        '-u 100' ,
        'DEF:idle_min='+cpurrd+'/cpu-idle.rrd:value:MIN' ,
        'DEF:idle_avg='+cpurrd+'/cpu-idle.rrd:value:AVERAGE' ,
        'DEF:idle_max='+cpurrd+'/cpu-idle.rrd:value:MAX' ,
        'CDEF:idle_nnl=idle_avg,UN,0,idle_avg,IF' ,
        'DEF:wait_min='+cpurrd+'/cpu-wait.rrd:value:MIN' ,
        'DEF:wait_avg='+cpurrd+'/cpu-wait.rrd:value:AVERAGE' ,
        'DEF:wait_max='+cpurrd+'/cpu-wait.rrd:value:MAX' ,
        'CDEF:wait_nnl=wait_avg,UN,0,wait_avg,IF' ,
        'DEF:nice_min='+cpurrd+'/cpu-nice.rrd:value:MIN' ,
        'DEF:nice_avg='+cpurrd+'/cpu-nice.rrd:value:AVERAGE' ,
        'DEF:nice_max='+cpurrd+'/cpu-nice.rrd:value:MAX' ,
        'CDEF:nice_nnl=nice_avg,UN,0,nice_avg,IF' ,
        'DEF:user_min='+cpurrd+'/cpu-user.rrd:value:MIN' ,
        'DEF:user_avg='+cpurrd+'/cpu-user.rrd:value:AVERAGE' ,
        'DEF:user_max='+cpurrd+'/cpu-user.rrd:value:MAX' ,
        'CDEF:user_nnl=user_avg,UN,0,user_avg,IF' ,
        'DEF:system_min='+cpurrd+'/cpu-system.rrd:value:MIN' ,
        'DEF:system_avg='+cpurrd+'/cpu-system.rrd:value:AVERAGE' ,
        'DEF:system_max='+cpurrd+'/cpu-system.rrd:value:MAX' ,
        'CDEF:system_nnl=system_avg,UN,0,system_avg,IF' ,
        'DEF:softirq_min='+cpurrd+'/cpu-softirq.rrd:value:MIN' ,
        'DEF:softirq_avg='+cpurrd+'/cpu-softirq.rrd:value:AVERAGE' ,
        'DEF:softirq_max='+cpurrd+'/cpu-softirq.rrd:value:MAX' ,
        'CDEF:softirq_nnl=softirq_avg,UN,0,softirq_avg,IF' ,
        'DEF:interrupt_min='+cpurrd+'/cpu-interrupt.rrd:value:MIN' ,
        'DEF:interrupt_avg='+cpurrd+'/cpu-interrupt.rrd:value:AVERAGE'
,
        'DEF:interrupt_max='+cpurrd+'/cpu-interrupt.rrd:value:MAX' ,
        'CDEF:interrupt_nnl=interrupt_avg,UN,0,interrupt_avg,IF' ,
        'DEF:steal_min='+cpurrd+'/cpu-steal.rrd:value:MIN' ,
        'DEF:steal_avg='+cpurrd+'/cpu-steal.rrd:value:AVERAGE' ,
        'DEF:steal_max='+cpurrd+'/cpu-steal.rrd:value:MAX' ,
        'CDEF:steal_nnl=steal_avg,UN,0,steal_avg,IF' ,
        'CDEF:steal_stk=steal_nnl' ,
        'CDEF:interrupt_stk=interrupt_nnl,steal_stk,+' ,
        'CDEF:softirq_stk=softirq_nnl,interrupt_stk,+' ,
        'CDEF:system_stk=system_nnl,softirq_stk,+' ,
```

```
'CDEF:user_stk=user_nnl,system_stk,+' ,
'CDEF:nice_stk=nice_nnl,user_stk,+' ,
'CDEF:wait_stk=wait_nnl,nice_stk,+' ,
'CDEF:idle_stk=idle_nnl,wait_stk,+' ,
'AREA:idle_stk#CEFFEA' ,
'"LINE1:idle_stk#CEFFEA:idle     "' ,
'"GPRINT:idle_min:MIN:%6.1lf Min,"' ,
'"GPRINT:idle_avg:AVERAGE:%6.1lf Avg,"' ,
'"GPRINT:idle_max:MAX:%6.1lf Max,"' ,
'"GPRINT:idle_avg:LAST:%6.1lf Last\\l"' ,
'AREA:wait_stk#ffebbf' ,
'"LINE1:wait_stk#ffb000:wait     "' ,
'"GPRINT:wait_min:MIN:%6.1lf Min,"' ,
'"GPRINT:wait_avg:AVERAGE:%6.1lf Avg,"' ,
'"GPRINT:wait_max:MAX:%6.1lf Max,"' ,
'"GPRINT:wait_avg:LAST:%6.1lf Last\\l"' ,
'AREA:nice_stk#bff7bf' ,
'"LINE1:nice_stk#00e000:nice     "' ,
'"GPRINT:nice_min:MIN:%6.1lf Min,"' ,
'"GPRINT:nice_avg:AVERAGE:%6.1lf Avg,"' ,
'"GPRINT:nice_max:MAX:%6.1lf Max,"' ,
'"GPRINT:nice_avg:LAST:%6.1lf Last\\l"' ,
'AREA:user_stk#bfbfff' ,
'"LINE1:user_stk#0000ff:user     "' ,
'"GPRINT:user_min:MIN:%6.1lf Min,"' ,
'"GPRINT:user_avg:AVERAGE:%6.1lf Avg,"' ,
'"GPRINT:user_max:MAX:%6.1lf Max,"' ,
'"GPRINT:user_avg:LAST:%6.1lf Last\\l"' ,
'AREA:system_stk#ffbfbf' ,
'"LINE1:system_stk#ff0000:system   "' ,
'"GPRINT:system_min:MIN:%6.1lf Min,"' ,
'"GPRINT:system_avg:AVERAGE:%6.1lf Avg,"' ,
'"GPRINT:system_max:MAX:%6.1lf Max,"' ,
'"GPRINT:system_avg:LAST:%6.1lf Last\\l"' ,
'AREA:softirq_stk#ffbfff' ,
'"LINE1:softirq_stk#ff00ff:softirq  "' ,
'"GPRINT:softirq_min:MIN:%6.1lf Min,"' ,
'"GPRINT:softirq_avg:AVERAGE:%6.1lf Avg,"' ,
'"GPRINT:softirq_max:MAX:%6.1lf Max,"' ,
'"GPRINT:softirq_avg:LAST:%6.1lf Last\\l"' ,
'AREA:interrupt_stk#e7bfe7' ,
'"LINE1:interrupt_stk#a000a0:interrupt"' ,
'"GPRINT:interrupt_min:MIN:%6.1lf Min,"' ,
'"GPRINT:interrupt_avg:AVERAGE:%6.1lf Avg,"' ,
'"GPRINT:interrupt_max:MAX:%6.1lf Max,"' ,
'"GPRINT:interrupt_avg:LAST:%6.1lf Last\\l"' ,
'AREA:steal_stk#bfbfbf' ,
'"LINE1:steal_stk#000000:steal    "' ,
'"GPRINT:steal_min:MIN:%6.1lf Min,"' ,
'"GPRINT:steal_avg:AVERAGE:%6.1lf Avg,"' ,
'"GPRINT:steal_max:MAX:%6.1lf Max,"' ,
'"GPRINT:steal_avg:LAST:%6.1lf Last\\l"'
```

```
    ]
    cpu_line_args = " ".join(cpu_args)
    graph = Graph(name="cpu-"+core, group=group)
    graph.title="Cpu core "+core
    graph.command=cpu_line_args
    graph.priority=0
    graph.save()

def make_memory_graph(memoryrrd, group):
    memory_args = [
        '-s end-1h',
    #   '-e 1303382682',
        '-w 600',
        '-h 240',
        '-t memory' ,
        '-E' ,
        '-b 1024' ,
        '-v Bytes' ,
        '-l 0',
        '-M',
        'DEF:free_min='+memoryrrd+'/memory-free.rrd:value:MIN' ,
        'DEF:free_avg='+memoryrrd+'/memory-free.rrd:value:AVERAGE' ,
        'DEF:free_max='+memoryrrd+'/memory-free.rrd:value:MAX' ,
        'CDEF:free_nnl=free_avg,UN,0,free_avg,IF' ,
        'DEF:cached_min='+memoryrrd+'/memory-cached.rrd:value:MIN' ,
        'DEF:cached_avg='+memoryrrd+'/memory-cached.rrd:value:AVERAGE'
,
        'DEF:cached_max='+memoryrrd+'/memory-cached.rrd:value:MAX' ,
        'CDEF:cached_nnl=cached_avg,UN,0,cached_avg,IF' ,
        'DEF:buffered_min='+memoryrrd+'/memory-buffered.rrd:value:MIN'
,
        'DEF:buffered_avg='+memoryrrd+'/memory-
buffered.rrd:value:AVERAGE' ,
        'DEF:buffered_max='+memoryrrd+'/memory-buffered.rrd:value:MAX'
,
        'CDEF:buffered_nnl=buffered_avg,UN,0,buffered_avg,IF' ,
        'DEF:used_min='+memoryrrd+'/memory-used.rrd:value:MIN' ,
        'DEF:used_avg='+memoryrrd+'/memory-used.rrd:value:AVERAGE' ,
        'DEF:used_max='+memoryrrd+'/memory-used.rrd:value:MAX' ,
        'CDEF:used_nnl=used_avg,UN,0,used_avg,IF' ,
        'CDEF:used_stk=used_nnl' ,
        'CDEF:buffered_stk=buffered_nnl,used_stk,+' ,
        'CDEF:cached_stk=cached_nnl,buffered_stk,+' ,
        'CDEF:free_stk=free_nnl,cached_stk,+' ,
        'AREA:free_stk#bff7bf' ,
        '"LINE1:free_stk#00e000:free    "' ,
        '"GPRINT:free_min:MIN:%5.1lf%s Min,"' ,
        '"GPRINT:free_avg:AVERAGE:%5.1lf%s Avg,"' ,
        '"GPRINT:free_max:MAX:%5.1lf%s Max,"' ,
        '"GPRINT:free_avg:LAST:%5.1lf%s Last\\l"' ,
        'AREA:cached_stk#bfbfff' ,
        '"LINE1:cached_stk#0000ff:cached  "' ,
```

```
        '"GPRINT:cached_min:MIN:%5.1lf%s Min,"' ,
        '"GPRINT:cached_avg:AVERAGE:%5.1lf%s Avg,"' ,
        '"GPRINT:cached_max:MAX:%5.1lf%s Max,"' ,
        '"GPRINT:cached_avg:LAST:%5.1lf%s Last\\l"' ,
        'AREA:buffered_stk#ffebbf' ,
        '"LINE1:buffered_stk#ffb000:buffered"' ,
        '"GPRINT:buffered_min:MIN:%5.1lf%s Min,"' ,
        '"GPRINT:buffered_avg:AVERAGE:%5.1lf%s Avg,"' ,
        '"GPRINT:buffered_max:MAX:%5.1lf%s Max,"' ,
        '"GPRINT:buffered_avg:LAST:%5.1lf%s Last\\l"' ,
        'AREA:used_stk#ffbfbf' ,
        '"LINE1:used_stk#ff0000:used    "' ,
        '"GPRINT:used_min:MIN:%5.1lf%s Min,"' ,
        '"GPRINT:used_avg:AVERAGE:%5.1lf%s Avg,"' ,
        '"GPRINT:used_max:MAX:%5.1lf%s Max,"' ,
        '"GPRINT:used_avg:LAST:%5.1lf%s Last\\l"'
    ]
    memory_line_args = " ".join(memory_args)
    graph = Graph(name="memory", group=group)
    graph.title="Memory"
    graph.command=memory_line_args
    graph.priority=0
    graph.save()

def make_interface_graph(interfacerrd, group, interface_name):
    interface_args = [
        '-s end-1h',
    #   '-e 1303382682',
        '-w 600',
        '-h 240',
        '-t if_octets-eth0' ,
        '-v Bits/s' ,
        '-E' ,
        '--units=si' ,
        'DEF:out_min_raw='+interfacerrd+'/if_octets.rrd:tx:MIN' ,
        'DEF:out_avg_raw='+interfacerrd+'/if_octets.rrd:tx:AVERAGE' ,
        'DEF:out_max_raw='+interfacerrd+'/if_octets.rrd:tx:MAX' ,
        'DEF:inc_min_raw='+interfacerrd+'/if_octets.rrd:rx:MIN' ,
        'DEF:inc_avg_raw='+interfacerrd+'/if_octets.rrd:rx:AVERAGE' ,
        'DEF:inc_max_raw='+interfacerrd+'/if_octets.rrd:rx:MAX' ,
        'CDEF:out_min=out_min_raw,8,*' ,
        'CDEF:out_avg=out_avg_raw,8,*' ,
        'CDEF:out_max=out_max_raw,8,*' ,
        'CDEF:inc_min=inc_min_raw,8,*' ,
        'CDEF:inc_avg=inc_avg_raw,8,*' ,
        'CDEF:inc_max=inc_max_raw,8,*' ,
        'CDEF:overlap=out_avg,inc_avg,GT,inc_avg,out_avg,IF' ,
        'CDEF:mytime=out_avg_raw,TIME,TIME,IF' ,
        'CDEF:sample_len_raw=mytime,PREV(mytime),-' ,
        'CDEF:sample_len=sample_len_raw,UN,0,sample_len_raw,IF' ,
        'CDEF:out_avg_sample=out_avg_raw,UN,0,out_avg_raw,IF,sample_le
n,*' ,
```

```
        'CDEF:out_avg_sum=PREV,UN,0,PREV,IF,out_avg_sample,+' ,
        'CDEF:inc_avg_sample=inc_avg_raw,UN,0,inc_avg_raw,IF,sample_le
n,*' ,
        'CDEF:inc_avg_sum=PREV,UN,0,PREV,IF,inc_avg_sample,+' ,
        'AREA:out_avg#B7EFB7' ,
        'AREA:inc_avg#B7B7F7' ,
        'AREA:overlap#89B3C9' ,
        '"LINE1:out_avg#00E000:Outgoing"' ,
        '"GPRINT:out_avg:AVERAGE:%5.1lf%s Avg,"' ,
        '"GPRINT:out_max:MAX:%5.1lf%s Max,"' ,
        '"GPRINT:out_avg:LAST:%5.1lf%s Last"' ,
        '"GPRINT:out_avg_sum:LAST:(ca. %5.1lf%sB Total)\\l"' ,
        '"LINE1:inc_avg#0000FF:Incoming"' ,
        '"GPRINT:inc_avg:AVERAGE:%5.1lf%s Avg,"' ,
        '"GPRINT:inc_max:MAX:%5.1lf%s Max,"' ,
        '"GPRINT:inc_avg:LAST:%5.1lf%s Last"' ,
        '"GPRINT:inc_avg_sum:LAST:(ca. %5.1lf%sB Total)\\l"'
    ]
    interface_line_args = " ".join(interface_args)
    graph = Graph(name="interface-"+interface_name, group=group)
    graph.title="Interface "+interface_name
    graph.command=interface_line_args
    graph.priority=0
    graph.save()

def make_load_graph(loadrrd, group):
    load_args = [
        '-s end-1h',
    #   '-e 1305798665',
        '-w 600',
        '-h 240',
        '-v "System load"',
        '-t Load',
    #   '-E',
        '-l 0',
        '-X 0',
        '-Y',
        'DEF:s_avg='+loadrrd+'/load.rrd:shortterm:AVERAGE',
        'DEF:s_min='+loadrrd+'/load.rrd:shortterm:MIN',
        'DEF:s_max='+loadrrd+'/load.rrd:shortterm:MAX',
        'DEF:m_avg='+loadrrd+'/load.rrd:midterm:AVERAGE',
        'DEF:m_min='+loadrrd+'/load.rrd:midterm:MIN',
        'DEF:m_max='+loadrrd+'/load.rrd:midterm:MAX',
        'DEF:l_avg='+loadrrd+'/load.rrd:longterm:AVERAGE',
        'DEF:l_min='+loadrrd+'/load.rrd:longterm:MIN',
        'DEF:l_max='+loadrrd+'/load.rrd:longterm:MAX',
        'AREA:s_max#B7EFB7',
        'AREA:s_min#FFFFFF',
        '"LINE1:s_avg#FF0000: 1m average"',
        '"GPRINT:s_min:MIN:%4.2lf Min,"',
        '"GPRINT:s_avg:AVERAGE:%4.2lf Avg,"',
        '"GPRINT:s_max:MAX:%4.2lf Max,"',
```

```
            '"GPRINT:s_avg:LAST:%4.2lf Last\\n"',
            '"LINE1:m_avg#FF6600: 5m average"',
            '"GPRINT:m_min:MIN:%4.2lf Min,"',
            '"GPRINT:m_avg:AVERAGE:%4.2lf Avg,"',
            '"GPRINT:m_max:MAX:%4.2lf Max,"',
            '"GPRINT:m_avg:LAST:%4.2lf Last\\n"',
            '"LINE1:l_avg#FFAA00:15m average"',
            '"GPRINT:l_min:MIN:%4.2lf Min,"',
            '"GPRINT:l_avg:AVERAGE:%4.2lf Avg,"',
            '"GPRINT:l_max:MAX:%4.2lf Max,"',
            '"GPRINT:l_avg:LAST:%4.2lf Last\\n"'
        ]
        load_line_args = " ".join(load_args)
        graph = Graph(name="load", group=group)
        graph.title="Load"
        graph.command=load_line_args
        graph.priority=0
        graph.save()

    def make_graphs(device):
        hostname = str(device.config.id)
        path = RRD_DIR + hostname + "/"
        cpugroup,created =
    GraphGroup.objects.get_or_create(name="cpu-"+hostname, title="cpu")
        if not created:
            cpugroup.graphs.all().delete()
        memorygroup,created =
    GraphGroup.objects.get_or_create(name="memory-"+hostname,
    title="memory")
        if not created:
            memorygroup.graphs.all().delete()
        interfacegroup,created =
    GraphGroup.objects.get_or_create(name="interface-"+hostname,
    title="interface")
        if not created:
            interfacegroup.graphs.all().delete()
        loadgroup,created =
    GraphGroup.objects.get_or_create(name="load-"+hostname, title="load")
        if not created:
            loadgroup.graphs.all().delete()
        for file in os.listdir(path):
            if file.startswith("cpu"):
                make_cpu_graph(path+file, cpugroup, file[4:])
            if file.startswith("memory"):
                make_memory_graph(path+file, memorygroup)
            if file.startswith("interface"):
                make_interface_graph(path+file, interfacegroup, file[10:])
            if file.startswith("load"):
                make_load_graph(path+file, loadgroup)
        graphmanager,created =
    GraphManager.objects.get_or_create(device=device,
                            cpugraphs = cpugroup,
```

```
                        memorygraphs = memorygroup,
                        interfacegraphs = interfacegroup,
                        loadgraphs = loadgroup,
                               )
```

- **src/devices/models.py:**

```
from django.db import models
from django.contrib.auth.models import Group
from django_netjsonconfig.models import Config
from djgeojson.fields import PointField, PolygonField
from django.utils.translation import ugettext as _
from collectd_rest.models import  GraphGroup

class Device(models.Model):
    config = models.ForeignKey(Config,
                on_delete=models.PROTECT)
    owner = models.ForeignKey(Group, editable=False, null=True)
    manufacturer = models.CharField(_("Manufacturer"), max_length=256,
blank=True)
    model_name = models.CharField(_("Model name"), max_length=256,
blank=True)
    activation_date = models.DateField(auto_now_add=True)
    is_indoor = models.NullBooleanField()

    notes =  models.TextField(_("Administration notes"), blank=True)

    country = models.CharField(_("Country"), max_length=50,
blank=True)
    city = models.CharField(_("City"), max_length=50, blank=True)
    street = models.CharField(_("Street"), max_length=100, blank=True)
    zip_code = models.CharField(_("ZIP/Postal Code"), max_length=12,
blank=True)

    geom = PointField(help_text=" ")

    def __unicode__(self):
        return self.model_name

class GraphManager(models.Model):
    device = models.OneToOneField(Device,
                on_delete=models.CASCADE,
                primary_key=True)
    cpugraphs = models.OneToOneField(GraphGroup,
on_delete=models.CASCADE, related_name="cpu_graphmanager")
    memorygraphs = models.OneToOneField(GraphGroup,
on_delete=models.CASCADE, related_name="memory_graphmanager")
    interfacegraphs = models.OneToOneField(GraphGroup,
on_delete=models.CASCADE, related_name="interface_graphmanager")
    loadgraphs = models.OneToOneField(GraphGroup,
on_delete=models.CASCADE, related_name="load_graphmanager")
```

```
from .signals import *
```

- **src/devices/signals.py:**

```
from django.db.models.signals import post_save
from django_cas_ng.signals import cas_user_authenticated
from django.conf import settings
from django.dispatch import receiver
from django.contrib.auth import get_user_model
from django.contrib.auth.models import Group
from guardian.shortcuts import assign_perm
from .models import Device
from django_netjsonconfig.models import Config

@receiver(cas_user_authenticated)
def cas_authentication_handler(sender, **kwargs):
    if hasattr(settings, 'AFFILIATION_FIELD'):
        affiliation_field = settings.AFFILIATION_FIELD
    else:
        affiliation_field = "affiliation"
    if (affiliation_field in kwargs["attributes"]) and
kwargs["attributes"][affiliation_field]:
        affiliation = kwargs["attributes"][affiliation_field]
    else:
        print("********************Error in authentication signal,
user has no affiliation********************")
        return
    authentication_username = kwargs["user"]
    User = get_user_model()
    try:
        user = User.objects.get(username = authentication_username)
    except User.DoesNotExist:
        print("********************Error in authentication signal,
user not found in database********************")
        return
    user.is_staff=True
    assign_perm('django_netjsonconfig.add_template', user)
    user.save()
    group, created = Group.objects.get_or_create(name = affiliation)
    user.groups.clear()
    user.groups.add(group)

@receiver(post_save, sender=Device)
def device_post_save(sender, **kwargs):
    device = kwargs["instance"]
    assign_perm('devices.change_device', device.owner, device)
    assign_perm('devices.delete_device', device.owner, device)

@receiver(post_save, sender=Config)
def config_creation(sender, **kwargs):
    config, created = kwargs["instance"], kwargs["created"]
    if created:
```

```
        config.name = str(config.id)
        config.save()
```

- **src/devices/urls.py:**

```
from django.conf import settings
from django.conf.urls import url
from django.conf.urls.static import static
from django.views.generic import TemplateView, UpdateView
from djgeojson.views import GeoJSONLayerView

from .models import Device
from . import views

urlpatterns = [
    url(r'^$', views.IndexView.as_view(), name='devices_home'),
    url(r'^data.geojson/(?P<pk>\d+)$',
views.DeviceGeomonitorView.as_view(model=Device,
properties=('model_name, notes, id, config')), name='data'),
    url(r'^device/(?P<pk>\d+)/update$', views.DeviceUpdate.as_view(),
name='device_update'),
    url(r'^device/(?P<pk>\d+)/delete$', views.DeviceDelete.as_view(),
name='device_delete'),
    url(r'^device/register', views.DeviceRegister.as_view(),
name='device_register'),
    url(r'^device/search', views.DeviceSearch.as_view(),
name='device_search'),
    url(r'^device/select_templates', views.TemplateList.as_view(),
name='devices_apply_template'),
    url(r'^device/(?P<device_id>\d+)/(?P<graph_type>[a-z]+)$',
views.graph_view, name='device_graphs'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

- **src/devices/views.py:**

```
from django.shortcuts import render, get_object_or_404
from django.http import HttpResponse, HttpResponseRedirect, Http404
from django.core.urlresolvers import reverse
from .forms import DeviceUpdateForm, DeviceRegisterForm,
DeviceSearchForm
from .models import Device, GraphManager
from django.views.generic import TemplateView, ListView
from django.views.generic.edit import FormView, UpdateView,
CreateView, DeleteView
from leaflet.forms.widgets import LeafletWidget
from django_netjsonconfig.models import Config, Template
from rules.contrib.views import LoginRequiredMixin,
PermissionRequiredMixin
from guardian.shortcuts import assign_perm
from guardian.mixins import PermissionListMixin
from djgeojson.views import GeoJSONLayerView
from django.contrib.auth.models import User
```

```
from search_listview.list import SearchableListView
from django_actions.views import ActionViewMixin
from .actions import select_devices_action, apply_template_action,
rebuild_graphs_action
from collectd_rest.models import Graph, GraphGroup
from .graphs import make_graphs

class IndexView(LoginRequiredMixin, TemplateView):
    template_name = 'devices/index.html'

class DeviceGeomonitorView(PermissionListMixin, GeoJSONLayerView):
    permission_required = 'devices.change_device'

    def dispatch(self, request, *args, **kwargs):
        request.user = User.objects.get(pk=kwargs['pk'])
        return super(DeviceGeomonitorView, self).dispatch(request,
args, kwargs)

class DeviceRegister(LoginRequiredMixin, CreateView):
    model = Device
    form_class = DeviceRegisterForm
    template_name_suffix = '_register'

    def form_valid(self, form):
        form.instance.config =
Config.objects.get(id=form.cleaned_data['config_uuid'])
        user_group = self.request.user.groups.first()
        form.instance.owner = user_group
        assign_perm('django_netjsonconfig.change_config', user_group,
form.instance.config)
        assign_perm('django_netjsonconfig.delete_config', user_group,
form.instance.config)
        return super(DeviceRegister, self).form_valid(form)

    def get_success_url(self, *args, **kwargs):
                return reverse("devices_home")

class DeviceUpdate(PermissionRequiredMixin, UpdateView):
    model = Device
    form_class = DeviceUpdateForm
    template_name_suffix = '_update_form'
    permission_required = 'devices.change_device'

    def form_valid(self, form):
        self.object = form.save(commit=False)
        self.object.user = self.request.user
        self.object.save()
        return HttpResponseRedirect(self.get_success_url())

    def get_success_url(self, *args, **kwargs):
        return reverse("devices_home")
```

```
class DeviceDelete(PermissionRequiredMixin, UpdateView):
    model = Device
    fields = []
    template_name_suffix = '_delete_form'
    permission_required = 'devices.delete_device'

    def get_success_url(self, *args, **kwargs):
        return reverse("devices_home")

class DeviceSearch(ActionViewMixin, PermissionListMixin,
SearchableListView):
    permission_required = 'devices.change_device'
    actions = [select_devices_action, rebuild_graphs_action]
    model = Device
    paginate_by = 10
    template_name = "devices/device_search.html"
    searchable_fields = ["manufacturer", "model_name", "country",
"city", "street", "zip_code"]
    specifications = {
        "manufacturer": "__icontains",
        "model_name": "__icontains",
    }

    def get_search_form(self):
        forms = []
        form = DeviceSearchForm
        form.prefix = Device.__name__

        initial = list(self.request.GET.lists())
        initial_tmp = {}
        for k, vals in initial:
            tmp_list = k.split(Device.__name__ + "-")
            if len(tmp_list) == 2:
                list_val_tmp = vals[0] if len(vals) == 1 else [val for
val in vals if val != '']
                initial_tmp[tmp_list[-1]] = list_val_tmp
        form.initial = initial_tmp
        forms.append(form)
        return forms

class TemplateList(ActionViewMixin, PermissionListMixin, ListView):
    permission_required = 'django_netjsonconfig.change_template'
    actions = [apply_template_action]
    model = Template
    template_name = "devices/template_list.html"

def graph_view(request, device_id, graph_type):
    device = get_object_or_404(Device, pk=device_id)
    try:
        graph_manager = GraphManager.objects.get(pk=device_id)
    except GraphManager.DoesNotExist:
        make_graphs(device)
```

```
        graph_manager = get_object_or_404(GraphManager, pk=device_id)
    if (graph_type == "cpu"):
        graph_group = graph_manager.cpugraphs
    elif (graph_type == "memory"):
        graph_group = graph_manager.memorygraphs
    elif (graph_type == "interface"):
        graph_group = graph_manager.interfacegraphs
    elif (graph_type == "load"):
        graph_group = graph_manager.loadgraphs
    else:
        raise Http404
    return render(request, 'devices/device_graphs.html',
        {'device' : device, 'group' : graph_group})
```

- **src/devices/static/js/graphs.js:**

```
function makeTabs(group) {
        var name  = group['name'];
        var title = group['title'];
        var graphs= group['graphs'];

        var code = "<h2>"+"</h2>\n";
        code += "<ul class=\"nav nav-tabs\">\n";
        graphs.map(function(g,i){
        if (i === 0) {
                    code += "<li class=\"active\"><a data-
toggle=\"tab\"  href=\"#"+g['name']+"\"\>"+g['title']+"</a></li>\n";
        } else {
                    code += "<li><a data-toggle=\"tab\"
href=\"#"+g['name']+"\"\>"+g['title']+"</a></li>\n";
        }
        });
        code += "</ul>\n\n";
        return code;
}

function renderGraph(graph) {
        var name  = graph['name'];
        var title = graph['title'];
        var url   = graph['url'];

    var url = url.substring(0, url.length-1);
    var url2 = url.concat("?format=png");

        var code = "<div id=\"" + name + "\" class=\"django-collectd-
rest-graph\">\n";
        code += "<h3>" + title + "</h3>\n";
        code += "<img src=" + url2 + " height=\"360\"
width=\"900\">\n";
        code += "</div>\n";
        return code;
}
```

```
function renderGroup(group) {
        var name  = group['name'];
        var title = group['title'];
        var graphs= group['graphs'];

        var code = "<div class=\"tab-content\">\n";
        graphs.map(function(g,i){
        if (i === 0) {
            code += "<div id=\""+g['name']+"\" class=\"tab-pane fade
in active\">\n";
        } else {
            code += "<div id=\""+g['name']+"\" class=\"tab-pane
fade\">\n";
        }
                code +=  renderGraph(g) + "\n</div>\n";
        });
        code += "</div>\n";
        return code;
}

$(document).ready(function(){

        $.get( "/collectd_rest/groups/", function( data ) {
                data.forEach(function(entry) {
                        var name = entry['name'];
                        var code= "<div class=\"container\">"
            code += makeTabs(entry);
            code += renderGroup(entry);
            code += "</div>";
                        $(".django-collectd-rest-group#" +
name).html(code);
                });
        });
});
```

Παράρτημα 2

Αρχείο settings.py

```
import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
SECRET_KEY = 'Unsafe key - please replace'
ALLOWED_HOSTS = ['*']
INSTALLED_APPS = [
    'django_netjsonconfig',
    'django_netjsonconfig.admin_theme',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'sortedm2m',
    'reversion',
    'django_x509',

    'leaflet',
    'djgeojson',

    'django_cas_ng',
    'guardian',
    'rules.apps.AutodiscoverRulesConfig',
    'django_actions',
    'search_listview',

    'bootstrap_themes',
    'crispy_forms',

    'rest_framework',
    'collectd_rest',

    'devices',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
ROOT_URLCONF = 'openwisp_monitor.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'openwisp_monitor.wsgi.application'
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValida
tor',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'Europe/Athens'
USE_I18N = True
USE_L10N = True
USE_TZ = True
STATIC_ROOT = os.path.join(BASE_DIR, "static/")
STATIC_URL = '/static/'
```

```
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR
NETJSONCONFIG_SHARED_SECRET = "secret"

LEAFLET_CONFIG = {
    'DEFAULT_CENTER': (51.47879, 0),
    'DEFAULT_ZOOM': 5,
    'MIN_ZOOM': 3,
    'MAX_ZOOM': 18,
    'PLUGINS': {
        'basics': {
            'js': ['http://fgnass.github.io/spin.js/dist/spin.min.js',

'http://makinacorpus.github.io/Leaflet.Spin/leaflet.spin.js'],
            'auto-include': True
        }
    }
}

SERIALIZATION_MODULES = {
    'geojson': 'djgeojson.serializers'
}

AUTHENTICATION_BACKENDS = (
    'rules.permissions.ObjectPermissionBackend',
    'django.contrib.auth.backends.ModelBackend', # default
    'guardian.backends.ObjectPermissionBackend',
    'django_cas_ng.backends.CASBackend',
)
GUARDIAN_RENDER_403 = True
LOGIN_URL = 'cas_ng_login'
LOGIN_REDIRECT_URL = 'devices_home'
LOGOUT_URL = 'cas_ng_login'
LOGOUT_REDIRECT_URL = 'cas_ng_login'
CRISPY_TEMPLATE_PACK = 'bootstrap3'
SESSION_SERIALIZER =
'django.contrib.sessions.serializers.PickleSerializer'
CAS_SERVER_URL = 'http://127.0.0.1/cas/'
CAS_VERSION = '3'
CAS_REDIRECT_URL = 'devices_home'
COLLECTD_RRD_DIR = "/var/lib/collectd/rrd/"
AFFILIATION_FIELD = "userPrincipalName"
```

Παράρτημα 3

Μεταβλητές του ρόλου ansible Dimpolissiou.openwisp2

Όνομα μεταβλητής	Παράδειγμα
openwisp2_cas_server	"http://127.0.0.1/cas/"
openwisp2_cas_affiliation	"userPrincipalName"
openwisp2_collectd_rrd_dir	"/var/lib/collectd/rrd/"
openwisp2_shared_secret	mysharedsecret
openwisp2_python	python2.7
openwisp2_path	/opt/openwisp2
openwisp2_database	engine: django.db.backends.postgresql name: openwisp2 user: postgres password: "" host: "" port: "" options: {}
openwisp2_language_code	en-gb
openwisp2_time_zone	UTC
openwisp2_allowed_hosts	- myadditionalhost.openwisp.org
openwisp2_ssl_cert	"/etc/nginx/ssl/server.crt"
openwisp2_ssl_key	"/etc/nginx/ssl/server.key"
openwisp2_ssl_country	"US"
openwisp2_ssl_state	"California"
openwisp2_ssl_locality	"San Francisco"
openwisp2_ssl_organization	"IT dep."
openwisp2_default_cert_validity	1825
openwisp2_default_ca_validity	3650
openwisp2_http_allowed_ip	"10.8.0.0/16"
openwisp2_extra_django_apps	- django_extensions
openwisp2_nginx_spdy	true
openwisp2_nginx_ipv6	false

Παράρτημα 4

Επιλογές του πακέτου openwisp-config

- url: διεύθυνση του εξυπηρετητή OpenWISP2 controller
- interval: Ο χρόνος (σε δευτερόλεπτα) μεταξύ δύο διαδοχικών ελέγχων για αλλαγή της διάρθρωσης. Από προεπιλογή είναι 120.
- verify_ssl: Εάν θέλουμε να πραγματοποιείται έλεγχος SSL. Από προεπιλογή έχει τιμή 1.
- shared_secret: Το μυστικό κλειδί με το οποίο πραγματοποιούν οι δρομολογητές αυτόματη εγγραφή στο σύστημα.
- consistent_key: Εάν θέλουμε το κλειδί να παράγεται από την πράξη `md5sum(mac_address + shared_secret)` αντί να είναι τυχαίο. Από προεπιλογή έχει τιμή 1.
- merge_config: Εάν θέλουμε η επιθυμητή διάρθρωση να συγχωνεύεται με την υπάρχουσα. Από προεπιλογή έχει τιμή 1.
- test_config: Εάν θέλουμε η νέα διάρθρωση να ελέγχεται μετά την εφαρμογή της. Από προεπιλογή έχει τιμή 1.
- test_script: Μονοπάτι για σενάριο ελέγχου που έχει δημιουργήσει ο χρήστης. Για περισσότερες πληροφορίες δείτε εδώ: https://github.com/openwisp/openwisp-config#configuration-test
- uuid: Μοναδικό αναγνωριστικό της διάρθρωσης της συσκευής. Παράγεται αυτόματα κατά την εγγραφή της συσκευής.
- key: Το κλειδί που χρειάζεται για να κατεβάσουμε τη διάρθρωση από τον εξυπηρετητή. Παράγεται αυτόματα κατά την εγγραφή της συσκευής.

- unmanaged: Λίστα από επιλογές διάρθρωσης που δε θα αλλάξουν από το openwisp. Για περισσότερες πληροφορίες δείτε εδώ: https://github.com/openwisp/openwisp-config#unmanaged-configurations

- capath: Τιμή που περνάμε στην παράμετρο --capath του curl.

- cacert: Τιμή που περνάμε στην παράμετρο --cacet του curl.

- connect_timeout: Τιμή που περνάμε στην παράμετρο --connect-timeout του curl. Από προεπιλογή είναι 15.

- max_time: Τιμή που περνάμε στην παράμετρο --max-time του curl. Από προεπιλογή είναι 30.

- mac_interface: Η διεπαφή που θα χρησιμοποιηθεί για την αυτόματη εγγραφή. Από προεπιλογή είναι η eth0.

- pre_reload_hook: Μονοπάτι ενός εκτελέσιμου αρχείου που θέλουμε να εκτελείται μετά από κάθε εφαρμογή νέας διάρθρωσης. Για περισσότερες πληροφορίες δείτε εδώ: https://github.com/openwisp/openwisp-config#pre-reload-hook