Υλοποίηση Πρωτόκολλου CoAP σε ετερογενή ασύρματα δίκτυα αισθητήρων και ελεκτών

Δημήτρης Γιαννακόπουλος

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής

Πανεπιστήμιο Πατρών

Επιβλέπων Καθηγητής : Παύλος Σπυράκης

Συνεπιβλέπων : Ιωάννης Χατζηγιαννάκης

Σεπτέμβριος 2013

Ευχαριστίες

Με τη διπλωματική εργασία αυτή ολοκληρώνεται ένας κύκλος σπουδών. Στο σημείο αυτό νιώθω την ανάγκη να ευχαριστήσω όλους όσους συνέβαλαν με το δικό τους τρόπο ο καθένας σε αυτή την ολοκλήρωση.

Καταρχάς, θα ήθελα να ευχαριστήσω θερμά τους επιβλέποντες μου Καθηγητή Παύλο Σπυράκη και Δρ. Ιωάννη Χατζηγιαννάκη για το ενδιαφέρον θέμα που μου έδωσαν την ευκαιρία να ασχοληθώ αλλά κυρίως για την εμπιστοσύνη και την καθοδήγησή τους.

Επίσης θα ήθελα να ευχαριστήσω τον Δημήτρη Αμαξιλάτη και τον Βασίλη Γεωργιτζίκη για την υπομονή και την πολύτιμη βοήθειά τους.

Περίληψη

Στα πλαίσια της διπλωματικής εργασίας του συγγραφέα, παρουσιάστηκε και υλοποιήθηκε το πρωτόκολλο εφαρμογών CoAP στην βιβλιοθήκη αλγορίθμων Wiselib και στην πλατφόρμα ανοιχτού υλικού Arduino. Υλοποιήθηκαν εφαρμογές βασισμένες στο CoAP σε συσκευές ασύρματων δικτύων αισθητήρων. Επεκτάθηκε υπάρχον σύστημα συλλογής και αποθήκευσης πληροφοριών ώστε να υποστηρίζει το CoAP. Παρουσιάζονται όλα τα χαρακτηριστικά, οι υπηρεσίες και οι ευκολίες του συστήματος που παρέχει στους τελικούς χρήστες. Γίνεται αξιολόγησή του και παρουσιάζονται πιθανές μελλοντικές κατευθύνσεις.

Περιεχόμενα

1 Εισαγωγή

1.1 Κίνητρο και σημασία

Τα Ασύρματα Δίκτυα Αισθητήρων (Wireless Sensor Network, WSN) βρίσκονται σε μια συνεχή άνοδο τα τελευταία χρόνια έχοντας κερδίσει το ενδιαφέρον της επιστήμης της πληροφορικής τόσο στην βιομηχανία, όσο και στον ακαδημαϊκό χώρο. Αποτελούνται συνήθως από τυχαία κατανεμημένες αυτόνομες συσκευές με περιορισμένους πόρους και υπολογιστική ισχύ, οι οποίες επικοινωνούν μεταξύ τους χρησιμοποιώντας ασύρματα πρωτόκολλα επικοινωνίας, όπως το IEEE 802.15.4 [1]. Φέρουν αισθητήρες μέσω των οποίον είναι ικανές να αναγνωρίσουν διάφορα χαρακτηριστικά του περιβάλλοντος χώρου, όπως θερμοκρασία, ένταση φωτός, βαρομετρική πίεση, ποιότητα αέρα, κίνηση. Παράλληλα μπορούν να εκτελούν εντολές που τους μεταβιβάζονται ή να τις εκτελούν αυτόματα αν τηρηθούν κάποιες συνθήκες, χωρίς εξωτερική παρέμβαση.

Μερικές εφαρμογές των ασύρματων δικτύων αισθητήρων είναι οι εξής:

- Παρακολούθηση κτηρίων (Building Monitoring System – BMS) μέσω των αισθητήρων. Μπορούν να ελέγχουν τον φωτισμό ή τον κλιματισμό με στόχο τον περιορισμό την ενεργειακής κατανάλωσης του κτηρίου. Μπορούν να ελέγχουν την δραστηριότητα, όπως κίνηση, πόρτες και παράθυρα, για λόγους ασφαλείας.

- Μέτρηση ποιότητα αέρα, σε μία πόλη ή σε μία βιομηχανική περιοχή.

- Εντοπισμός πυρκαγιών σε δασικές εκτάσεις για πρόβλεψη κινδύνου και ενημέρωση σε περίπτωση συμβάντος.

Τα δίκτυα αισθητήρων έχουν αποτελέσει επίκεντρο για πολλές ερευνητικές ομάδες, τόσο σε θεωρητικό, όσο και σε πρακτικό επίπεδο. Έχει δοθεί ιδιαίτερη βάση στην δομή των δικτύων και στην υλοποίηση αλγορίθμων στο κατανεμημένο περιβάλλον τους. Προβλήματα ετερογενών ασύρματων δικτύων, δηλαδή ενιαία δίκτυα που αποτελούνται από συσκευές διαφορετικών χαρακτηριστικών και υλοποιήσεων στο επίπεδο του δικτύου, έχουν αντιμετωπιστεί σε σημαντικό βαθμό. Παράλληλα έχουν παρουσιαστεί τρόποι που διευκολύνουν την εγκατάσταση, την διαχείριση και τον προγραμματισμό τέτοιων δικτύων καθώς και οργανωμένα συστήματα συλλογής πληροφοριών και αυτοματοποίησης.

Ωστόσο οι προγραμματιστές που καλούνται να υλοποιήσουν εφαρμογές σε αυτά τα δίκτυα, έρχονται αντιμέτωποι με την απουσία κάποιου πρωτοκόλλου στο επίπεδο των εφαρμογών (application layer), κάτι που τους αναγκάζει να δημιουργούν τα δικά τους ιδιόκτητα (proprietary) πρότυπα και πρωτόκολλα επικοινωνίας ανάλογα με τις εκάστοτε ανάγκες. Αυτό καθιστά συχνά αδύνατη την

διαλειτουργικότητα (interoperability) των εφαρμογών και της κοινής χρήσης διαφορετικών υλοποιήσεων σε ένα δίκτυο.

Βάσει τέτοιων εφαρμογών, ολοκληρωμένα συστήματα έχουν σχεδιαστεί και υλοποιηθεί με σκοπό την συλλογή και αποθήκευση πληροφοριών σε πραγματικό χρόνο. Περιορίζονται όμως στην αποκλειστική χρήση των συγκεκριμένων υλοποιήσεων. Η προσθήκη νέων συσκευών προϋποθέτει την προσαρμογή του συστήματος στο πρωτόκολλο επικοινωνίας που χρησιμοποιεί η διαφορετική εφαρμογή. Συστήματα που επιχειρούν κάτι τέτοιο συχνά βασίζονται σε reverse engineering των διαφορετικών υλοποιήσεων, με την συντήρηση και την διαχείρισή τους να απαιτεί καθημερινή παρέμβαση από τους διαχειριστές.

Η πρόκληση που εμφανίζεται είναι να εκμεταλλευτούμε στο έπακρο τις δυνατότητες που μας παρέχονται ήδη σε ένα πιο μαζικό και αυτοματοποιημένο επίπεδο, μέσω της υλοποίησης ενός αυστηρά ορισμένου πρωτοκόλλου εφαρμογών για τα ασύρματα δίκτυα αισθητήρων.

Ένα πρωτόκολλο που ταυτίζεται πλήρως με τις ανάγκες των δικτύων αισθητήρων είναι το Constrained Application Protocol (CoAP) [2], βασισμένο στην θεμελιώδη Representation State Transfer (REST) [3] αρχιτεκτονική του παγκόσμιου ιστού. Γενικά είναι παρόμοιο με το HTTP τόσο στον τρόπο λειτουργίας, όσο και στις υπηρεσίες που παρέχει. Σε αντίθεση με το HTTP, έχει σχεδιαστεί αποκλειστικά με γνώμονα συσκευές περιορισμένης ισχύος και δυνατοτήτων. Λειτουργεί στο επίπεδο των εφαρμογών και δημιουργεί την κατάλληλη αφαίρεση από το επίπεδο δικτύου. Οι εφαρμογές που βασίζονται σε αυτό μπορούν να αγνοούν τις ιδιαιτερότητες του δικτύου και της συσκευής στις οποίες είναι υλοποιημένες. Παρόλο που το πρωτόκολλο είναι ακόμα υπό ανάπτυξη, έχει αποτελέσει επίκεντρο ερευνητικών και βιομηχανικών προγραμμάτων. Οι υπάρχουσες υλοποιήσεις όμως στα ασύρματα δίκτυα αισθητήρων υποστηρίζουν περιορισμένο αριθμό οικογενειών συσκευών.

Με την ύπαρξη ενός ενιαίου πρωτοκόλλου εφαρμογών όπως το CoAP, ολοκληρωμένα συστήματα διαχείρισης μπορούν να βασιστούν σε αυτό και να ξεπεραστούν οι περιορισμοί και οι δυσκολίες που υπήρχαν με τα ιδιόκτητα πρωτόκολλα επικοινωνίας. Η διαχείριση και η υποστήριξη νέων συσκευών απλοποιείται σημαντικά και η ανθρώπινη παρέμβαση περιορίζεται στο ελάχιστο.

1.2 Στόχος

Στόχος της διπλωματικής εργασίας είναι βρεθούν τα κατάλληλα εργαλεία, υπάρχουσες πλατφόρμες και υλοποιήσεις, ώστε αρχικά να υλοποιηθεί ένα ενιαίο επίπεδο εφαρμογών σε διάφορες συσκευές που απαρτίζουν ένα ασύρματο δίκτυο αισθητήρων και να παρουσιαστούν εφαρμογές που βασίστηκαν σε αυτό. Με βάση αυτές να παρουσιαστεί ένα ενιαίο σύστημα αυτοματοποίησης και διαχείρισης του δικτύου, που θα αντιμετωπίζει τα υπάρχοντα προβλήματα. Επιχειρείται όχι απλά

η υλοποίηση, αλλά και η εκμετάλλευση των πρόσθετων δυνατοτήτων που παρέχονται από ένα ενιαίο πρωτόκολλο.

Στόχος είναι επίσης η δυνατότητα χειρισμού του συστήματος και των αισθητήρων από τον τελικό χρήστη μέσω ενός απλού περιβάλλοντος, αποκρύπτοντας τις τεχνικές ιδιαιτερότητες του δικτύου.

1.3 Συνεισφορά

Η συνεισφορά αυτής της εργασίας στο πεδίο των ασύρματων δικτύων αισθητήρων είναι ότι παρέχεται πλέον η υλοποίηση του CoAP, ενός κοινού πρωτοκόλλου εφαρμογών σε ένα μεγάλο εύρος συσκευών μέσω της βιβλιοθήκης αλγορίθμων Wiselib. Η υλοποίηση είναι ανοιχτού κώδικα και διατίθεται ελεύθερα υπό την άδεια GNU GPLv3.

Η επέκταση ενός υπάρχοντος συστήματος παρακολούθησης κτηρίων με την προσθήκη CoAP δυνατοτήτων, τόσο στο επίπεδο επικοινωνίας με τα δίκτυα αισθητήρων, όσο και στο επίπεδο συστήματος – πελάτη. Η επέκταση δεν θα περιοριστεί στην απλή εισαγωγή του CoAP, αλλά και στην χρήση των ιδιαιτεροτήτων και των κανόνων που ορίζει, ώστε δημιουργηθεί ένα αυτόνομο σύστημα παρακολούθησης, απλοποιώντας διαδικασίες που διαφορετικά θα απαιτούσαν ανθρώπινη παρέμβαση. Το σύστημα είναι ανοιχτού κώδικα και διατίθεται ελεύθερο.

Στα πλαίσια της εργασίας έγινε μία σχετική δημοσίευση στο συνέδριο Emerging Technologies & Factory Automation (ETFA) [26] το 2012 στην Πολωνία.

1.4 Δομή

Στο 2ο κεφάλαιο παρουσιάζονται οι συσκευές του ασύρματου δικτύου αισθητήρων που χρησιμοποιήθηκαν στα πλαίσια της εργασίας. Γίνεται αναφορά στα χαρακτηριστικά τους αλλά και στις ιδιαιτερότητες που προκύπτουν από αυτά.

Στο 3ο κεφάλαιο παρουσιάζονται πλατφόρμες πάνω στις οποίες βασίστηκε η εργασία, για την ανάπτυξη του ενιαίου πρωτοκόλλου, καθώς και υπάρχοντα συστήματα που χρησιμοποιήθηκαν.

Στο 4ο κεφάλαιο γίνεται εκτενής παρουσίαση του CoAP και των χαρακτηριστικών του. Στην συνέχεια παρουσιάζονται επιπρόσθετα χαρακτηριστικά που χρησιμοποιήθηκαν αλλά δεν είναι στο κύριο πρωτόκολλο.

Στο 5ο κεφάλαιο ακολουθεί η ανάλυση της υλοποίησης του πρωτοκόλλου που έγινε στην εργασία, η υλοποίηση σε Java που χρησιμοποιήθηκε καθώς και το Firefox plug-in Copper που δίνει την δυνατότητα στον browser να στέλνει και να λαμβάνει CoAP μηνύματα.

Στο 6ο κεφάλαιο παρουσιάζεται το σύστημα και η αρχιτεκτονική που σχεδιάστηκε καθώς και τα χαρακτηριστικά και οι υπηρεσίες που προσφέρει στους διαχειριστές και τους χρήστες.

Στο 7ο κεφάλαιο αξιολογείται η εφαρμογή και παρουσιάζονται πειραματικά αποτελέσματα.

Τέλος στο 8ο κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας και κάποιες μελλοντικές κατευθύνσεις και στόχοι.

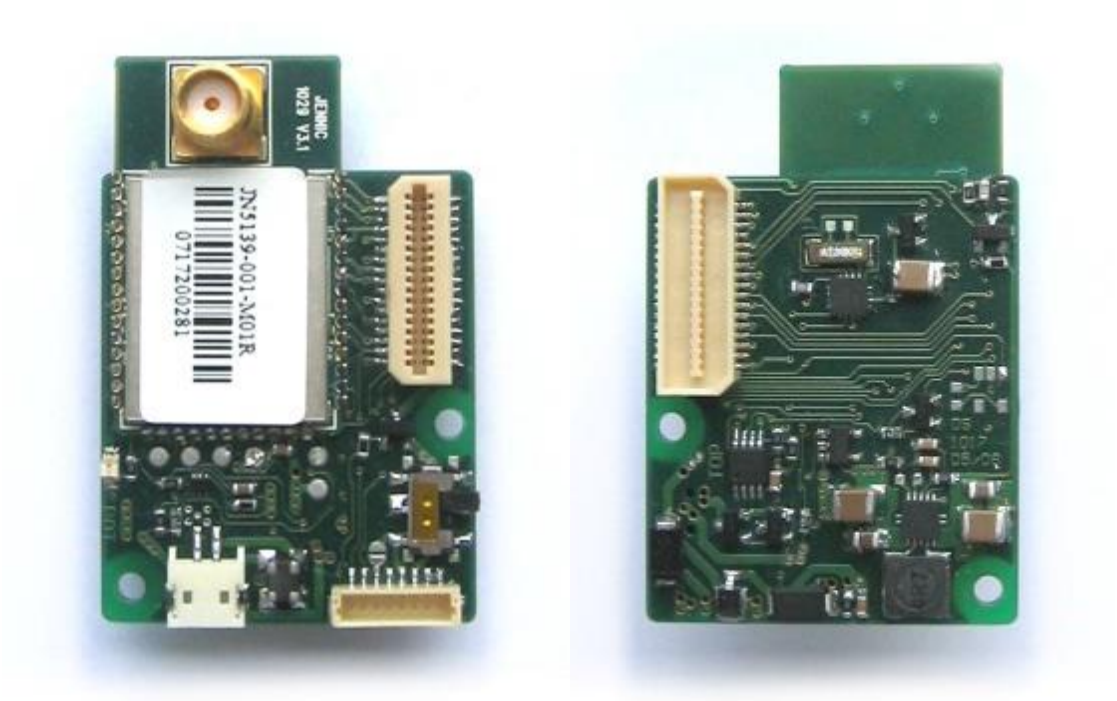
2 Συσκευές

2.1 iSense

2.1.1 iSense Core Module

Η κύρια οικογένεια συσκευών που χρησιμοποιήθηκε για την εργασία είναι το iSense Core Module 1 [8] (εν συντομία θα αναφέρονται απλά ως iSense) από την Coalesneses. Βασίζεται στον Jennic JN5148 wireless microcontroller [6], ένα ολοκληρωμένο κύκλωμα που συνδυάζει τον επεξεργαστή και τον ελεγκτή ασύρματου δικτύου σε ένα ολοκληρωμένο κύκλωμα. Ο επεξεργαστής υποστηρίζει 32 bit RISC τεχνολογία με ελεγχόμενη συχνότητα λειτουργίας μεταξύ 4 και 32 MHz. Η μνήμη είναι τύπου Flash και ανέρχεται στα 128 KBytes η οποία χρησιμοποιείται για τον κώδικα του προγράμματος και τα δεδομένα αυτού, κάτι που δίνει ελευθερία επιλογών στο πόσο θα καταλαμβάνει το καθένα.

Ο ελεγκτής δικτύου είναι συμβατός με το IEEE 802.15.4 πρότυπο, υποστηρίζει 16 κανάλια επικοινωνίας και ρυθμό μετάδοσης 250 Kbps. Παρέχει κρυπτογράφηση AES από το υλικό και είναι συμβατός με το ZigBee [7].



Εικόνα 2.1 iSense Core Module

Η συσκευή είναι εξοπλισμένη με εσωτερικό ρολόι υψηλής ακρίβειας (σφάλμα < 20ppm) για να υποστηρίζει μεγάλους κύκλους αδρανοποίησης. Υπάρχει επίσης ένα προγραμματιζόμενο LED για την διαδικασία αποσφαλμάτωσης. Τέλος, ένα βύσμα 34 ακροδεκτών βρίσκεται και στις δύο πλευρές της πλακέτας που επιτρέπει την σύνδεση άλλων συμπληρωματικών εξαρτημάτων (modules)

συμβατών με την συσκευή, κάτι που επιτρέπει την προσθήκη αισθητήρων και δυνατοτήτων.

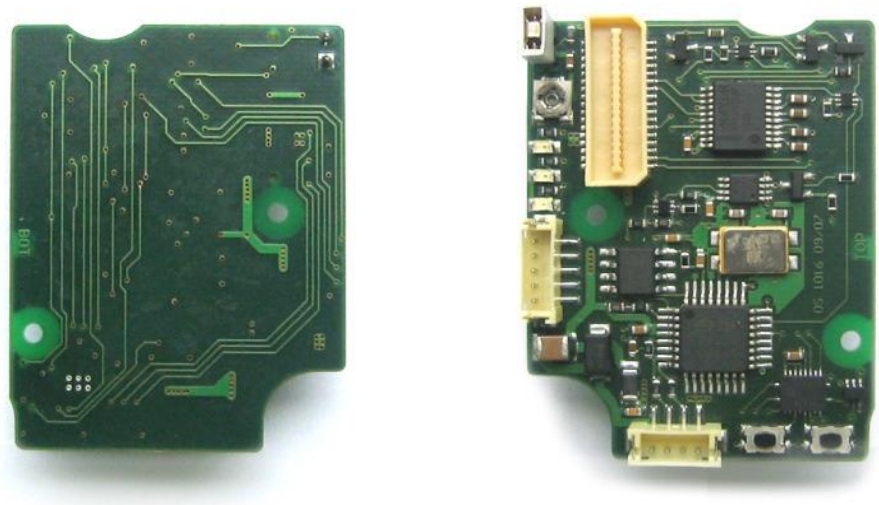
Οι συσκευές μπορούν να τροφοδοτηθούν από μπαταρίες, αντάπτορα ρεύματος ή μέσω θύρας USB του gateway module.

Υποστηρίζεται προγραμματισμός μέσω του ασύρματου δικτύου (over-the-air programming – OTAP) κάτι που επιτρέπει τον εκ νέου προγραμματισμό των συσκευών, αφού αυτές έχουν εγκατασταθεί χωρίς να απαιτείται κάποια φυσική πρόσβαση πλέον σε αυτές. Διαφορετικά προγραμματίζονται κανονικά μέσω του gateway module. Η γλώσσα προγραμματισμού είναι η C++ μέσω του ba-elf compiler.

2.1.2 iSense Modules

Προσθέτουν επιπλέον δυνατότητες στην συσκευή, ενώ είναι έτσι σχεδιασμένα ώστε να μπορεί να τοποθετηθεί το ένα πάνω στο άλλο σε μορφή στοίβας (stack). Η βιβλιοθήκη λογισμικού, ώστε να χρησιμοποιηθούν σε εφαρμογές, παρέχεται και η λειτουργία τους περιγράφεται στα έγγραφα του εκάστοτε συμπληρωματικού εξαρτήματος. Κάποια πολύ βασικά συμπληρωματικά εξαρτήματα που χρησιμοποιήθηκαν στα πλαίσια της εργασίας είναι τα Gateway Module, Environmental Sensor Module, Weather Sensor Module και όλα διατίθενται από την Coalesenses.

- **Gateway Module:** Προσφέρει δυνατότητα σύνδεσης με υπολογιστή μέσω USB και RS-232. Μπορεί να χρησιμοποιηθεί για μεταφορά δεδομένων προς τον υπολογιστή αλλά και για προγραμματισμό του συνδεδεμένου iSense.

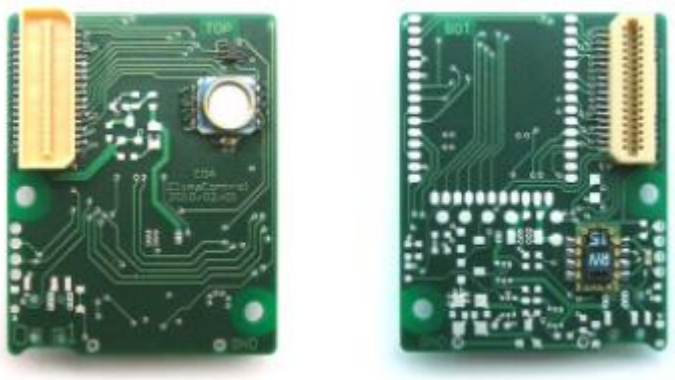


Εικόνα 2.2 Gateway Module

- **Environmental Sensor Module:** Προσφέρει δύο βασικούς αισθητήρες. Τον αισθητήρα θερμοκρασίας με συχνότητα δειγματοληψίας τα 10 Hz, ευαισθησία 1°C με εύρος από −55 έως 125°C. Τον αισθητήρα φωτός, με συχνότητα δειγματοληψίας το 1 Hz που δίνει ως έξοδο την ένταση του ορατού φωτός σε lux.

Εικόνα 2.3 Environmental Sensor Module

- **Weather Sensor Module:** Παρέχει αισθητήρες υψηλής ακρίβειας για θερμοκρασία, υγρασία και βαρομετρική πίεση. Η μικρή κατανάλωση της τάξης του 1 μΑ ενδείκνυται για χρήση σε συσκευές με μπαταρία που έχουν σκοπό να λειτουργούν για χρόνια.



Εικόνα 2.4 Weather Sensor Module

- **Security Module:** Παρέχει αισθητήρα κίνησης μέσω υπερύθρων. Εντοπίζει κίνηση αντικειμένων που έχουν θερμοκρασία διαφορετική του περιβάλλοντος χώρου (πχ άνθρωποι) σε απόσταση έως και 10 μέτρα με γωνία 110°. Μπορεί να ρυθμιστεί με τέτοιο τρόπο ώστε κάθε φορά που εντοπίζεται κίνηση να προκαλείται διακοπή (interrupt) στο σύστημα και να εκτελείται μια συνάρτηση που έχει ορίσει ο χρήστης.

Εικόνα 2.5 Security Sensor

2.1.3 Ιδιαιτερότητες Συσκευής

Η κύρια ιδιαιτερότητα βρίσκεται στην μνήμη της συσκευής. Έχοντας μια κοινή μνήμη για τον κώδικα του προγράμματος και τα δεδομένα του, δίνει την ελευθερία επιλογών αλλά στην περίπτωση πολύπλοκων υλοποιήσεων και προγραμμάτων εισάγει ένα σημαντικό πρόβλημα. Όσο μεγαλύτερος είναι ο κώδικας τόση λιγότερη μνήμη μένει για την εκτέλεση αυτού. Προβλήματα που προκύπτουν λόγω περιορισμένης μνήμης κατά την εκτέλεση είναι δύσκολο να εντοπιστούν και να λυθούν.

Μπορεί η δυνατότητα για ασύρματο προγραμματισμό να είναι ιδιαίτερα εύχρηστη, καταλαμβάνει όμως ένα σημαντικό μέρος στον εκτελέσιμο κώδικα, περιορίζοντας την διαθέσιμη μνήμη. Για τον λόγο αυτό χρησιμοποιείται κάποιες φορές μια ειδική έκδοση του firmware του iSense που αφαιρεί αυτήν την δυνατότητα.

2.2 Arduino

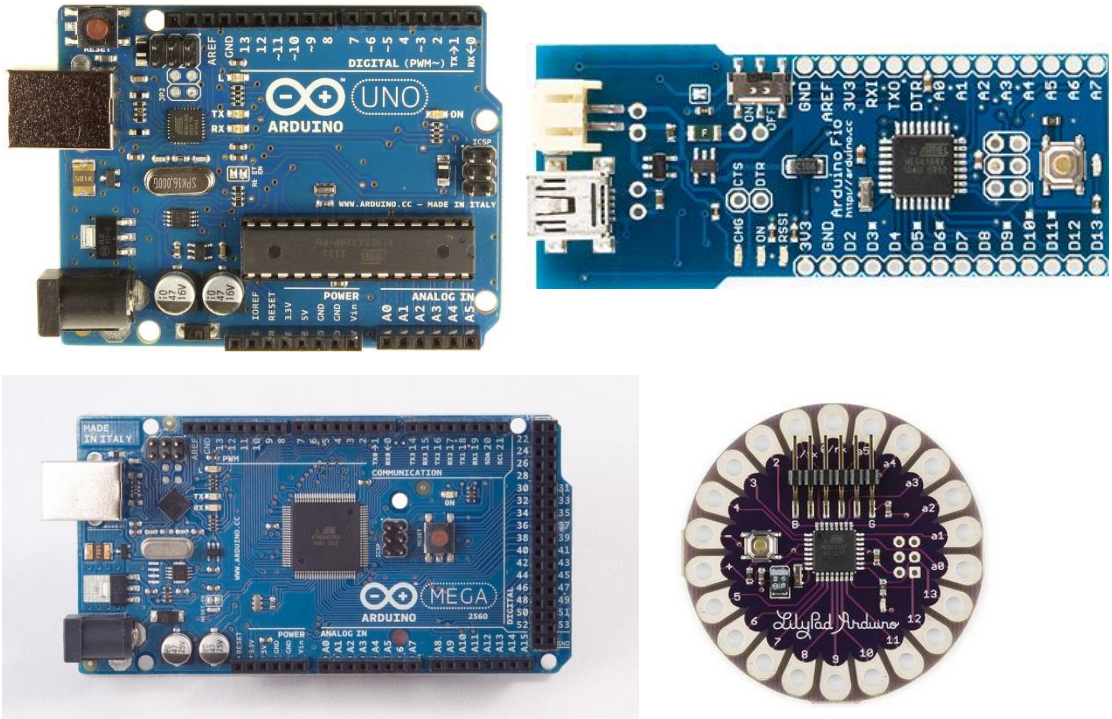
2.2.1 Arduino

Με την άνοδο των embedded συστημάτων εμφανίστηκε η ανάγκη για την δημιουργία μιας ανοιχτής πλατφόρμας ανάπτυξης, χωρίς τους περιορισμούς που συχνά υπάρχουν στα υπάρχοντα “κλειστά” συστήματα. Το Arduino επιχειρεί να φέρει την λογική του ανοιχτού λογισμικού στο υλικό, ώστε ένας ενδιαφερόμενος να μπορεί να φτιάξει το δικό του embedded σύστημα φτηνά, εύκολα, με δημόσια διαθέσιμα έγγραφα γύρω από τον σχεδιασμό, ανοιχτούς κώδικες υλοποίησης και την κοινότητα να τον υποστηρίζει.

Το Arduino είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή (Atmel AVR – Atmega328, ATmega168 στις νεότερες εκδόσεις, ATmega8 παλιότερα) και pin εισόδου/εξόδου. Ο μικροελεγκτής είναι από κατασκευής προγραμματισμένος με έναν bootloader που απλοποιεί την μεταφόρτωση προγραμμάτων στην εσωτερική μνήμη τύπου flash, έτσι ώστε να μην χρειάζεται εξωτερικός προγραμματιστής. Έχει τρεις ξεχωριστά διαθέσιμες μνήμες στο chip, την flash που είναι ο bootloader και η εφαρμογή που τρέχει, η SRAM που είναι η μνήμη δεδομένων κατά την εκτέλεση της εφαρμογής και την EEPROM όπου αποθηκεύονται πληροφορίες από την εφαρμογή ακόμα και μετά το κλείσιμο της συσκευής. Υπάρχουν διάφορες έτοιμες πλακέτες διαθέσιμες, με ποικίλα μεγέθη και δυνατότητες, όπως φαίνεται στον παρακάτω πίνακα όπου παρουσιάζονται μερικές από τις πιο γνωστές και χρησιμοποιημένες πλακέτες:

Όνομα	Μικροελεγκτής	Συχνότητα	Flash	SRAM	EEPROM	Pins	Διαστάσεις
Uno	ATmega328P	16 MHz	32 KB	2 KB	1 KB	20	68.6 mm × 53.3 mm
Mega 2560	ATmega2560	16 MHz	256 KB	8 KB	4 KB	70	101.6 mm × 53.3 mm
Duemilanove	ATmega168	16 MHz	16 KB	1 KB	512 B	20	68.6 mm × 53.3 mm
Fio	ATmega328P	8 MHz	32 KB	2 KB	1 KB	22	66.0 mm × 27.9 mm
LilyPad	Atmega 168V	8 MHz	16 KB	1 KB	512 B	20	51 mm (διάμετρος)
Nano v3.0	ATmega328	16 MHz	32 KB	2 KB	1 KB	22	43.18 mm × 18.54 mm
Mini	ATmega168	16 MHz	32 KB	2 KB	1 KB	22	17.8 mm × 33.0 mm

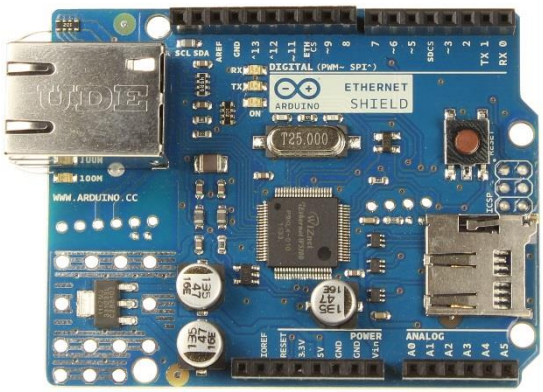
Πίνακας 2-1 Arduino Boards



Εικόνα 2.6 Arduino Boards, UNO (πάνω αριστερά), Fio (πάνω δεξιά), Mega256 (κάτω αριστερά), LilyPad (κάτω αριστερά)

Σε εννοιολογικό επίπεδο, στην χρήση του Arduino software stack, όλες οι πλακέτες προγραμματίζονται μέσω μίας RS-232 σειριακής σύνδεσης, αλλά ο τρόπος που επιτυγχάνεται αυτό, διαφέρει ανάλογα με την επιλογή του υλικού. Τα τωρινά Arduino προγραμματίζονται μέσω USB, αυτό καθιστάται δυνατό μέσω της εφαρμογής προσαρμοστικών chip USB-to-Serial. Κάποιες παραλλαγές, όπως το Arduino mini και το ανεπίσημο Boarduino, χρησιμοποιούν ένα αφαιρούμενο USB-to-Serial καλώδιο ή board, Bluetooth ή άλλες μεθόδους. Οι εφαρμογές προγραμματίζονται με τη γλώσσα Wiring, που ουσιαστικά πρόκειται για τη C++ με κάποιες απλοποιήσεις και μετατροπές, η οποία μετατρέπεται σε καθαρή C++ κατά την μεταγλώττιση. Οι βασικές βιβλιοθήκες είναι υλοποιημένες σε C και C++ και χρησιμοποιείται ο μεταγλωττιστής avr-gcc και AVR Libc.

Το Arduino έχει δύο μοναδικά χαρακτηριστικά. Η γλώσσα προγραμματισμού του είναι πολύ απλή στην χρήση και μπορεί κανείς πολύ γρήγορα να υλοποιήσει και να δοκιμάσει νέα χαρακτηριστικά. Το δεύτερο μεγάλο χαρακτηριστικό του είναι τα προγραμματιζόμενα pin εισόδου/εξόδου που μπορούν να χρησιμοποιηθούν για οποιαδήποτε σκοπό. Χωρίζονται σε δύο κατηγορίες, τα ψηφιακά pin εισόδου/εξόδου και τα αναλογικά pin εισόδου. Τα ψηφιακά μπορούν να χρησιμοποιηθούν για γενική χρήση και μπορούν να τεθούν σε λειτουργία εισόδου ή εξόδου και οι τιμές που παίρνουν είναι HIGH (1) ή LOW (0). Αντίθετα τα αναλογικά μπορούν μόνο να διαβάσουν ένα εύρος τιμών που μετατρέπεται σε ψηφιακή τιμή μέσω ενός 10-bit analog-to-digital (ADC) μετατροπέα. Η πιο σημαντική χρήση των pins όμως είναι η δυνατότητα να συνδεθούν σε αυτά, έτοιμες συμπληρωματικές πλακέτες, τα λεγόμενα Arduino Shields.



Εικόνα 2.7 Ethernet Shield

Τα Shields προσθέτουν πρόσθετες δυνατότητες στο Arduino. Υπάρχουν Shields που αυξάνουν την συνδεσιμότητά του (πχ WiFi, GSM), άλλα που παρέχουν διάφορους αισθητήρες, οθόνη LCD, GPS δέκτη, κλπ. Όλα συνδέονται με τα pin εισόδου/εξόδου ενώ παρέχεται η βιβλιοθήκη μέσω της οποίας μπορούν να χρησιμοποιηθούν σε μία εφαρμογή. Η λογική του open-hardware όμως επιτρέπει

στον καθένα να φτιάξει ένα δικό του Shield ανάλογα με τις ανάγκες του χωρίς να περιορίζεται στα εμπορικά διαθέσιμα.

Στην περίπτωση της εργασίας χρησιμοποιήθηκε μία πλακέτα που έχει φτιαχτεί για τις ανάγκες του Ινστιτούτο Έρευνας και Τεχνολογίας. Παρέχει αισθητήρες θερμότητας, φωτεινότητας και κίνησης. Επίσης, υποστηρίζει μέχρι και 5 διακόπτες ρεύματος, οι οποίοι ελέγχονται από το Arduino.

2.2.2Xbee

Το Arduino όμως δεν έχει άμεσα διαθέσιμη συνδεσιμότητα με 802.15.4 δίκτυα. Για τον λόγο αυτό χρησιμοποιείται το Xbee Shield [9], ένα συμπληρωματικό εξάρτημα που είναι σχεδιασμένο να συνδέεται σε υπάρχοντα συστήματα και να προσφέρει δυνατότητες δικτύωσης βάσει του πρωτοκόλλου 802.15.4. Στο Arduino συνδέεται μέσω της σειριακής θύρας, στέλνοντας τα δεδομένα που λαμβάνει από αυτήν και προωθώντας τα μηνύματα που λαμβάνει από το ασύρματο δίκτυο σε αυτήν. Η προεπιλογή όμως του Xbee είναι να κάνει broadcast ότι δεδομένα λαμβάνει από την σειριακή, το οποίο κρίνεται προβληματικό για τα ασύρματα δίκτυα αισθητήρων. Για τον λόγο αυτό, χρησιμοποιείται το Xbee σε λειτουργία API που ενεργοποιείται μέσω ειδικού προγράμματος, το οποίο επιτρέπει την αποστολή unicast μηνυμάτων και την χρήση του με πιο καθορισμένο από τον προγραμματιστή τρόπο.

Εικόνα 2.8 Xbee

2.3 Ομοιογενές 802.15.4

Ένα γενικότερο πρόβλημα της βιομηχανίας είναι ότι η κάθε εταιρία που φτιάχνει μια συσκευή ή έναν ελεγκτή δικτύου, πολύ συχνά εισάγει διαφοροποιήσεις στο πρότυπο που υλοποιεί, συνήθως για τις συγκεκριμένες ανάγκες του συστήματος. Το πρόβλημα αυτό εμφανίζεται και στην συγκεκριμένη περίπτωση, που παρόλο που και οι δύο διαφορετικές οικογένειες συσκευών (iSense, Xbee) παρουσιάζονται ως 802.15.4 συμβατές, κάτι τέτοιο δεν είναι εφικτό χωρίς παρέμβαση του προγραμματιστή.

Στην περίπτωση του Xbee έχει υλοποιηθεί το πρωτόκολλο 802.15.4, με την διαφορά ότι υπάρχουν δύο επιπλέον byte στην κεφαλίδα του πακέτου για την ανάγκη αποστολής ειδικών εντολών προς το Xbee, την αναφορά κατάστασης προς το Arduino μέσω των συνδεδεμένων pin, αλλά και για την χρήση επιπλέον πακέτων επιβεβαίωσης που χρησιμοποιούνται επιπλέον στα ήδη υπάρχοντα του πρωτοκόλλου. Αυτό δημιουργεί πολλά προβλήματα, κάνοντας άλλες 802.15.4 συσκευές να λαμβάνουν δύο επιπλέον byte σκουπιδιών απορρίπτοντας τα μηνύματα ως μη-συμβατά και το Xbee να αδυνατεί να λάβει πακέτα από άλλη συσκευή, αφού θα λείπουν αυτά τα δύο byte.

Για να παρακαμφθεί αυτή η ιδιαιτερότητα και να γίνουν απόλυτα συμβατές οι δύο οικογένειες συσκευών, γίνεται χρήση της βιβλιοθήκης MkSense [10]. Με το MkSense, έχει γίνει reverse engineering η κλειστή υλοποίηση του Xbee ώστε να βρεθεί ο τρόπος με τον οποίο λειτουργούν αυτά τα επιπλέον byte και να μπορέσουν να χρησιμοποιηθούν on demand, όποτε αυτό κρίνεται απαραίτητο. Αυτό επέτρεψε τις συσκευές να επικοινωνούν κανονικά και να δημιουργούν ένα κοινό ομοιογενές δίκτυο από ετερογενείς συσκευές.

3 Πλατφόρμες Ανάπτυξης

3.1 Wiselib

Η Wiselib [11] είναι μια βιβλιοθήκη αλγορίθμων για ασύρματα δίκτυα αισθητήρων. Υποστηρίζει μια πληθώρα συσκευών συμπεριλαμβανομένου των iSense και Arduino (κατά την διάρκεια εκπόνησης της εργασίας δεν ήταν ακόμα διαθέσιμη η υποστήριξη για Arduino) καθώς και διαφόρων άλλων γνωστών συσκευών (Contiki, Sunspot, κλπ.). Υποστηρίζει επίσης την έξοδο στον εξομοιωτή δικτύων αισθητήρων Shawn [12] για αποσφαλμάτωση και δοκιμές.

Η Wiselib παρέχει μια αρκετά μεγάλη συλλογή αλγορίθμων ήδη υλοποιημένων ή υπό ανάπτυξη, που χρησιμοποιούνται σε διαφόρους τομείς της έρευνας πάνω στα ασύρματα δίκτυα αισθητήρων, όπως δικτύωση, δρομολόγηση, τοπικοποίηση, συνάθροιση.

Είναι υλοποιημένη σε C++ και χρησιμοποιεί εκτενώς τα templates για την δημιουργία γενικών και παραμετροποιήσιμων συναρτήσεων. Ο κώδικας των αλγορίθμων είναι ανεξάρτητος από την συσκευή που τελικά θα τρέξει, ενώ η επιλογή του λειτουργικού γίνεται κατά την μεταγλώττιση, ανάλογα την πλατφόρμα που θέλουμε να τρέχει ο αλγόριθμος. Παρέχει διεπαφές για πρόσβαση στο λειτουργικό της εκάστοτε συσκευής και τυχόν συμπληρωματικά εξαρτήματα ή επιπλέον λειτουργίες που είναι ήδη υλοποιημένες.

Αυτά έχουν ως αποτέλεσμα να διευκολύνεται η ανάπτυξη και να μειώνεται σημαντικά η ανάγκη συγγραφής χαμηλότερου επιπέδου κώδικα. Υλοποιώντας έναν αλγόριθμο στην Wiselib υπάρχει πλέον η δυνατότητα υποστήριξης όλων των συμβατών πλατφορμών με ελάχιστο επιπλέον κόστος. Μόνο στις εφαρμογές που υλοποιούνται βάσει κάποιου αλγορίθμου απαιτείται να γίνει προσαρμογή, ανάλογα με τις συσκευές που στοχεύει, αλλά και πάλι είναι σημαντικά λιγότερη η προσπάθεια απ' ότι να γινόταν υλοποίηση απευθείας στην συσκευή.

Στον κώδικα 3.1 που ακολουθεί, παρουσιάζεται ένα παράδειγμα μιας απλής εφαρμογής αποστολής και λήψης μηνυμάτων σε Wiselib. Η επιλογή της συσκευής για την οποία θα γίνει η μεταγλώττιση επιλέγεται από τον χρήστη, ο κώδικας όμως παραμένει κοινός για όλες τις συμβατές με την Wiselib συσκευές. Το όρισμα "Os::AppMainParameter& value" είναι μέσω του οποίου περνάει η επιλογή της συσκευής στο πρόγραμμα. Η init() συνάρτηση εκτελείται αυτόματα μία φορά κατά την εκκίνηση του συστήματος και μετά από 5 δευτερόλεπτα εκτελείται η start(). Η receive_radio_message() καλείται όποτε φτάνει κάποιο μήνυμα στην συσκευή.

```
#include "external_interface/external_interface.h"
#include "algorithms/routing/tree/tree_routing.h"
typedef wiselib::OSMODEL Os;
class ExampleApplication
{
   public:
      void init( Os::AppMainParameter& value )
      {
         radio_ = &wiselib::FacetProvider<Os, Os::Radio>::get_facet( value );
         timer_ = &wiselib::FacetProvider<Os, Os::Timer>::get_facet( value );
         debug_ = &wiselib::FacetProvider<Os, Os::Debug>::get_facet( value );

         radio_->enable_radio();
         debug_->debug( "Hello World from Example Application!\n" );
         radio_->reg_recv_callback<ExampleApplication,
                                   &ExampleApplication::receive_radio_message>( this );
         timer_->set_timer<ExampleApplication,
                           &ExampleApplication::start>( 5000, this, 0 );
      }
      // -----------------------------------------------------------------------
      void start( void* )
      {
         debug_->debug( "broadcast message at %d \n", radio_->id() );
         Os::Radio::block_data_t message[] = "hello!\0";
         radio_->send( Os::Radio::BROADCAST_ADDRESS, sizeof(message), message );
         // following can be used for periodic messages to sink
         timer_->set_timer<ExampleApplication,
                           &ExampleApplication::start>( 5000, this, 0 );
      }
      // -----------------------------------------------------------------------
        void receive_radio_message( Os::Radio::node_id_t from, Os::Radio::size_t len,
Os::Radio::block_data_t *buf )
      {
         debug_->debug( "received msg at %x from %x", radio_->id(), from );
         debug_->debug( "  message is %s\n", buf );
      }
   private:
      Os::Radio::self_pointer_t radio_;
      Os::Timer::self_pointer_t timer_;
      Os::Debug::self_pointer_t debug_;
};
// -----------------------------------------------------------------------
wiselib::WiselibApplication<Os, ExampleApplication> example_app;
// -----------------------------------------------------------------------
void application_main( Os::AppMainParameter& value )
{
  example_app.init( value );
}
```

Κώδικας 3.1 Wiselib Example App

3.2 Wisebed

Το πρόγραμμα Wisebed [13] είναι μια προσπάθεια 9 ακαδημαϊκών και ερευνητικών ινστιτούτων σε όλη την Ευρώπη. Στόχος του προγράμματος είναι να προσφέρει μια υποδομή διασυνδεδεμένων Testbed (ειδικές πλατφόρμες διεξαγωγής πειραμάτων για εφαρμογές υπό ανάπτυξη) δικτύων ασύρματων αισθητήρων για ερευνητικούς σκοπούς. Αυτό δείχνει το πως πολλά μικρά ετερογενή δίκτυα αισθητήρων μπορούν να συνδεθούν και να σχηματίσουν μεγάλες οργανωμένες δομές. Δίνει την δυνατότητα στους ερευνητές να δουλέψουν σε διαφορετικές πραγματικές καταστάσεις, δομές και συσκευές που δεν έχουν φυσική πρόσβαση.

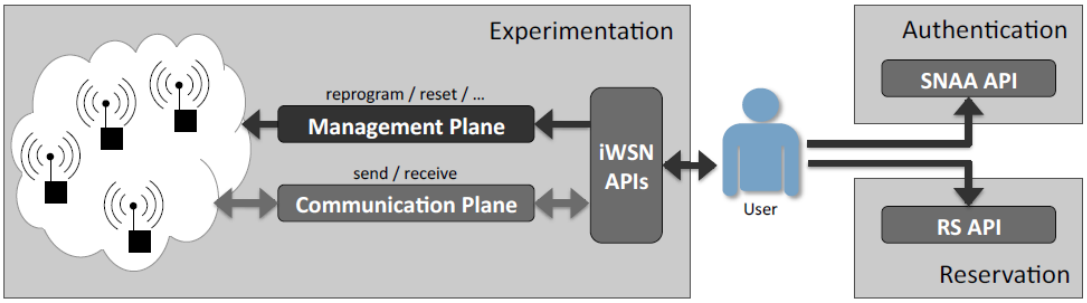
Ορίζεται ένα API για την αλληλεπίδραση με τις συσκευές, το οποίο έχει τρία διακριτά μέρη:

- **SNAA (Sensor Network Authentication and Authorization)**: Είναι υπεύθυνο για την πιστοποίηση των χρηστών και των δικαιωμάτων τους στο σύστημα.

- **RS (Reservation System):** Είναι υπεύθυνο για την δέσμευση ενός Testbed ή μέρη αυτού. Μέσω αυτού δεσμεύονται κόμβοι για να χρησιμοποιηθούν για ένα πείραμα για όσο διάστημα κρίνεται αναγκαίο από τον ερευνητή, αν αυτοί είναι διαθέσιμοι. Το σύστημα εξασφαλίζει ότι μόνο ένας χρήστης έχει δικαίωμα σε κάθε κόμβο του συστήματος, σε κάθε χρονική στιγμή.

- **iWSN (Wireless Sensor Network API):** Είναι υπεύθυνο για την πραγματική διαχείριση των αισθητήρων, τον προγραμματισμό, την συλλογή δεδομένων και ότι έχει να κάνει με την εκτέλεση του πειράματος του χρήστη.

Ο συνδυασμός τους και ο τρόπος λειτουργίας τους φαίνεται στην εικόνα 3.1. Ένας χρήστης πρέπει πρώτα να αποκτήσει πρόσβαση μέσω του SNAA, να πραγματοποιήσει μία κράτηση για κάποιους ή όλους τους κόμβους ενός Testbed μέσω του RS και στην συνέχεια να συνδεθεί με το iWSN για να πραγματοποιήσει το πείραμά του.



Εικόνα 3.1 TestbedRuntime

3.3 TestbedRuntime

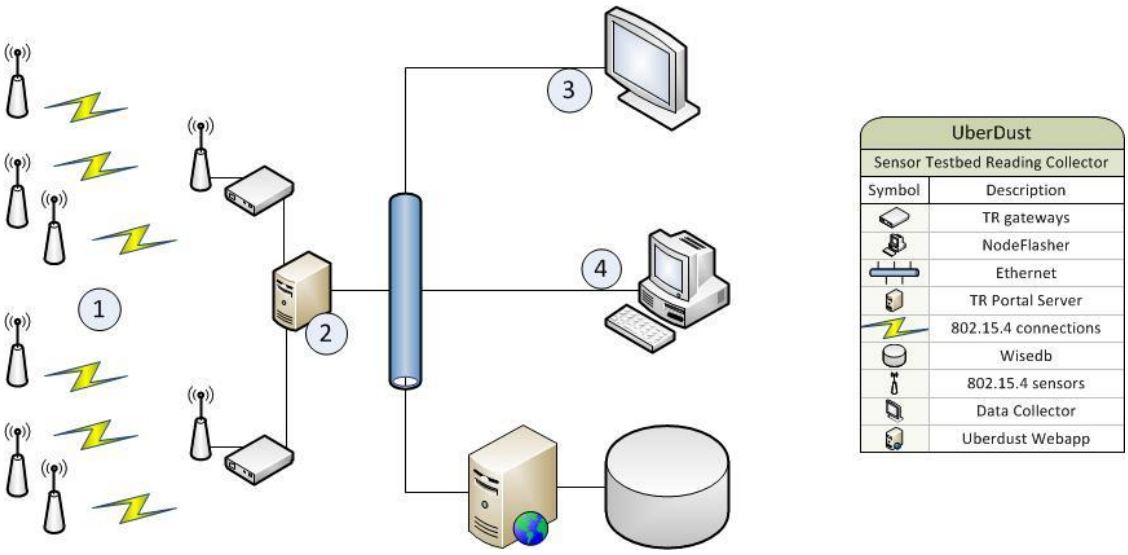
Για την πραγματοποίηση των πειραμάτων σε πραγματικές συσκευές απαιτείται η φόρτωση της εφαρμογής, ο συγχρονισμός όλων των συσκευών (τουλάχιστον στην εκκίνηση του αλγορίθμου), η συλλογή αποτελεσμάτων και η λήψη πληροφοριών κατά την διάρκεια εκτέλεσής του, αλλά και η αποστολή μηνυμάτων προς τις συσκευές. Για τον σκοπό αυτό χρησιμοποιήθηκε το λογισμικό TestbedRuntime [14] (TR) που προσφέρει την δυνατότητα ελέγχου πολλών αισθητήρων. Το TR υποστηρίζει αισθητήρες τύπου iSense, Pacemate, TelosB και Sunspot. Υλοποιεί το API όπως ορίζεται από το Wisebed. Προσφέρει πλήρη έλεγχο στις συσκευές και διευκολύνει σε μεγάλο βαθμό την εκτέλεση των πειραμάτων. Με την χρήση του λαμβάνεται έξοδος από όλες τις συσκευές σε πραγματικό χρόνο, στέλνονται εντολές (όπως μηνύματα εκκίνησης, αλλαγής κατάστασης και λήψης αναφοράς) και προσομοιώνονται ευκολότερα βλάβες και άλλες ειδικές καταστάσεις. Για την λειτουργία του TR οι συσκευές είναι συνδεδεμένες σε υπολογιστές που ελέγχονται από έναν κεντρικό Portal Server προσβάσιμο από τους χρήστες του συστήματος για την πραγματοποίηση των πειραμάτων. Προσφέρει ένα σύστημα ταυτοποίησης χρηστών ανάλογα με τις ανάγκες του διαχειριστή, καθώς και ένα σύστημα πραγματοποίησης κρατήσεων για οποιοδήποτε αριθμό κόμβων του συστήματος για όσο χρονικό διάστημα επιθυμεί. Ο χρήστης στην συνέχεια έχοντας το δικαίωμα να διαχειριστεί τους κόμβους του συστήματος μπορεί να περάσει στους αισθητήρες τον αλγόριθμο που θέλει να δοκιμάσει και να πραγματοποιήσει το πείραμά του.

3.4 Überdust

Το Überdust [15] είναι ένα σύστημα αποθήκευσης και διαμοιρασμού πληροφοριών μέσω web services. Η αλληλεπίδραση με το σύστημα γίνεται ανεξαρτήτου πλατφόρμας, με τους χρήστες να μπορούν να επικοινωνούν με αυτό μέσω REST [3] και WebSockets [16]. Τα δεδομένα αποθηκεύονται σε μία βάση δεδομένων χρησιμοποιώντας το WiseDB. Η έξοδος του συστήματος μπορεί να είναι σε διάφορους τύπους δεδομένων, όπως απλό κείμενο, HTML, JSON, RDF κα. Σχεδιάστηκε με σκοπό να συλλέγει πληροφορίες από ασύρματα δίκτυα

αισθητήρων, αλλά μπορεί να δεχτεί πληροφορίες από οποιαδήποτε πηγή. Είναι υλοποιημένο σε Java κάνοντας χρήση τεχνολογιών όπως Spring, Hibernate, JSP & JSTL κα.

Το Überdust λειτουργεί ως πελάτης για το TestbedRuntime και η δομή του συστήματος φαίνεται στην εικόνα 3.2.



Εικόνα 3.2 Überdust

1. Το ασύρματο δίκτυο αισθητήρων, όπου εκτελούνται συγκεκριμένες εφαρμογές στις συσκευές.
2. Ο διαχειριστής του TestbedRuntime.
3. TestbedListener, η εφαρμογή που συλλέγει τις πληροφορίες των συσκευών μέσω του TestbedRuntime.
4. NodeFlasher, η εφαρμογή που αναλαμβάνει να φορτώσει την εφαρμογή των αισθητήρων σε αυτούς.

Οι συσκευές του δικτύου αισθητήρων έχουν προγραμματιστεί με ειδικές εφαρμογές, οι οποίες παρέχονται μαζί με το Übserdust, μέσω του NodeFlasher που αναλαμβάνει αυτή την διαδικασία. Οι εφαρμογές αυτές κάνουν broadcast περιοδικά τις τιμές όλων των αισθητήρων τους στο δίκτυο. Κάποιοι κεντρικοί κόμβοι που έχουν οριστεί ως πύλες (Gateways) συλλέγουν αυτές τις πληροφορίες και τις τυπώνουν στην σειριακή θύρα. Οι πύλες είναι συνδεδεμένες με το TestbedRuntime, το οποίο διαβάζει τις πληροφορίες που τυπώνουν οι συσκευές. Από εκεί και έπειτα το Übserdust μέσω του TestbedListener συλλέγει τις πληροφορίες, τις μορφοποιεί και τις αποθηκεύει κατάλληλα στην βάση δεδομένων του. Η πρόσβαση σε αυτές γίνεται μέσω της Web εφαρμογής του.

Ο σχεδιασμός του επιτρέπει την δημιουργία πρόσθετων εφαρμογών που λειτουργούν ως πελάτες της Web εφαρμογής. Χρησιμοποιώντας τις πληροφορίες που συλλέγονται και είναι αποθηκευμένες στην βάση δεδομένων, μπορούν να υλοποιήσουν διάφορες αυτόματες διαδικασίες που ενεργούν ανάλογα με τα

δεδομένα. Παράδειγμα τέτοιας εφαρμογής είναι η διαχείριση του φωτισμού των γραφείων, όπου αν εντοπιστεί κίνηση ανάβουν τα φώτα, ενώ αν για ένα μεγάλο διάστημα δεν υπάρχει κίνηση αυτά σβήνουν.

4 CoAP

4.1 Εισαγωγή

Η χρήση των web services σε εφαρμογές διαδικτύου είναι πλέον ένα σύνηθες φαινόμενο και βασίζονται στην θεμελιώδη Representation State Transfer (REST) αρχιτεκτονική του παγκόσμιου ιστού. Η ομάδα του Constrained RESTful Environments (CoRE) [4] στοχεύει στον σχεδιασμό μιας REST αρχιτεκτονικής ιδανικής για συσκευές περιορισμένων δυνατοτήτων, όπως 8-bit microcontrollers με περιορισμένη μνήμη RAM, ROM και δυνατότητες δικτύωσης (πχ 6LoWPAN). Δίκτυα όπως το 6LoWPAN [5] υποστηρίζουν τον κατακερματισμό IPv6 πακέτων, κάτι που κρίνεται δαπανηρό σε πόρους. Ένας σχεδιαστικός στόχος ήταν να δημιουργηθεί ένα πρωτόκολλο που θα έχει μικρό overhead ώστε να περιορίζεται όσο είναι δυνατόν η ανάγκη για κατακερματισμό των πακέτων.

Το αποτέλεσμα της ομάδας του CoRE ήταν το Constrained Application Protocol (CoAP). Ο κύριος σχεδιαστικός στόχος ήταν ο σχεδιασμός ενός πρωτοκόλλου για τον παγκόσμιο ιστό για τις συγκεκριμένες ανάγκες του περιορισμένου περιβάλλοντος και συσκευών, ιδιαίτερα τις ανάγκες για περιορισμένη κατανάλωση, αυτοματοποίησης κτηρίων και άλλες machine-to-machine (M2M) εφαρμογές. Φέρει υπηρεσίες παρόμοιες του HTTP σε συσκευές με περιορισμένες υπολογιστικές ικανότητες, το οποίο κρίθηκε ιδανικό στην περίπτωση των ασύρματων δικτύων αισθητήρων. Με την υλοποίηση και χρήση του CoAP επιτυγχάνεται ο επιθυμητός διαχωρισμός, της τελικής εφαρμογής, από το πρωτόκολλο επικοινωνίας και το πρωτόκολλο δικτύου. Πλέον οι εφαρμογές τρέχουν πάνω από το επίπεδο του CoAP αγνοώντας τις ιδιαιτερότητες του επιπέδου του δικτύου.

Το CoAP είναι υπό ανάπτυξη (working draft) αλλά βρίσκεται πλέον κοντά στο τελικό πρότυπο. Μέχρι σήμερα έχουν εκδοθεί 18 συνολικά εκδόσεις του προτύπου. Κατά την διάρκεια υλοποίησης της εργασίας η έκδοση 08 ήταν διαθέσιμη και πάνω σε αυτήν βασίστηκε, για αυτό και η παρουσίασή του θα γίνει βάσει αυτής της έκδοσης. Στο τέλος γίνεται αναφορά στην σημερινή έκδοση και τι έχει αλλάξει.

4.2 Γενικά

Το μοντέλο επικοινωνίας βασίζεται στο γενικότερο μοντέλο του πελάτη/διακομιστή (client/server) του HTTP. Μία βασική διαφορά είναι ότι έχει σχεδιαστεί και για μηχανή προς μηχανή επικοινωνία (machine to machine – M2M) όπου μία συσκευή μπορεί να δρα ως πελάτης και ως διακομιστής. Παρόμοια με το HTTP, μία αίτηση (request) στέλνεται προς έναν διακομιστή, αναφερόμενη σε ένα αντικείμενο, με μία συγκεκριμένη μέθοδο αίτησης (request

method). Η απάντηση (response) περιλαμβάνει συχνά την αναπαράσταση του αντικειμένου που ζητήθηκε και τον κωδικό απάντησης (response code).

Σε αντίθεση με το HTTP η επικοινωνία είναι ασύγχρονη, χωρίς την δημιουργία κάποιας σύνδεσης και βασίζεται στο UDP που δεν παρέχει τρόπους αξιόπιστης μετάδοσης μηνυμάτων όπως στο TCP. Παρέχονται όμως ειδικά μηνύματα και τρόποι ελέγχου από το CoAP, ώστε να γίνει η επικοινωνία αξιόπιστη εφόσον αυτό είναι επιθυμητό από την εκάστοτε εφαρμογή. Ορίζονται 4 τύποι μηνυμάτων: Confirmable (CON), Non-Confirmable (NON), Acknowledgement (ACK), Reset (RST) και σε συνδυασμό με την μέθοδο αίτησης ή τον κωδικό απάντησης που περιέχεται σε κάποιους από αυτούς του τύπους μηνυμάτων, δηλώνουν αν πρόκειται για αίτηση ή για απάντηση.

Υποστηρίζει αιτήσεις προς multicast IP διευθύνσεις καθώς και διάφορους τρόπους προστασίας και ασφάλισης της επικοινωνίας.

Παράλληλα με το CoAP, αναπτύσσονται διάφορες επεκτάσεις του πρωτοκόλλου που προσθέτουν επιπλέον δυνατότητες και χαρακτηριστικά σε αυτό.

4.3 CoAP Μήνυμα

Τα μηνύματα του CoAP είναι απλά δυαδικά και αποτελούνται από μια σταθερού μεγέθους κεφαλίδα (header) ακολουθούμενη από κανένα ή περισσότερα options, ο αριθμός των οποίων δηλώνεται στην κεφαλίδα, και τέλος το payload αν υπάρχει.

	Octet	0							1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Ver			T		OC			Code								Message ID															
4	32	Options (εάν υπάρχουν)																															
...	...	Payload (εάν υπάρχει)																															

Εικόνα 4.1 CoAP Header

Τα πεδία της κεφαλίδας είναι τα εξής:

- **Version (Ver):** 2-bit unsigned integer. Δηλώνει την έκδοση του CoAP που χρησιμοποιείται, στην συγκεκριμένη έκδοση του πρωτοκόλλου πρέπει να είναι 1.

- **Type (T):** 2-bit unsigned integer. Δηλώνει αν το μήνυμα είναι τύπου Confirmable (0), NON-Confirmable (1), Acknowledgement (2) ή Reset (3).

- **Option Count (OC):** 4-bit unsigned integer. Δηλώνει πόσα options περιέχονται στο μήνυμα αμέσως μετά το header. Αν είναι 0, τότε αμέσως μετά το header αρχίζει το payload (αν υπάρχει).

- **Code:** 8-bit unsigned integer. Δηλώνει αν το μήνυμα είναι αίτηση (1-31), αν είναι απάντηση (64-191) ή αν είναι άδειο (0). Στην περίπτωση της αίτησης, το πεδίο αυτό ορίζει τον κωδικό αίτησης, ενώ στην απάντηση τον κωδικό απάντησης.

- **Message ID:** 16-bit unsigned integer. Δηλώνει τον κωδικό του μηνύματος. Χρησιμοποιείται για τον εντοπισμό διπλότυπων μηνυμάτων, για αντιστοίχηση αίτησης – απάντησης, αλλά και για τις επιβεβαιώσεις του αξιόπιστου μοντέλου επικοινωνίας.

Κάθε option έχει έναν συγκεκριμένο μοναδικό αριθμό (Option Number) που δηλώνεται στο πρότυπο του CoAP. Τα option που ακολουθούν μετά την κεφαλίδα πρέπει να είναι με την σειρά βάσει του option number. Κωδικοποίηση delta χρησιμοποιείται μεταξύ των options με τον αριθμό του καθενός να υπολογίζεται από το άθροισμα όλων των προηγούμενων options, με το option delta πεδίο του συγκεκριμένου. (παράδειγμα εικόνα)

	Octet	0							
Octet	Bit	0	1	2	3	4	5	6	7
0	0	Option Delta				Length 0-14			
1	8	Option Value							
…	…	…							

	Octet	0								1							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	Option Delta				1	1	1	1	Length 15-270							
2	16	Option Value															
…	…	…															

Εικόνα 4.2 Options format

Τα πεδία κάθε option είναι:

- **Option delta:** 4-bit unsigned integer. Δηλώνει την διαφορά του option number του συγκεκριμένου option από το προηγούμενο. Κάποια option numbers (14, 28, 42, ..) δεν δηλώνουν option αλλά χρησιμοποιούνται σαν “fenceposts” ώστε να υπολογιστούν μεγαλύτερα option number.

- **Length:** Δηλώνει το μέγεθος της τιμής του option σε bytes. Κανονικά είναι 4-bit unsigned integer και υποστηρίζει μεγέθη 0-14 bytes αλλά αν έχει την τιμή 15, προστίθεται άλλο ένα byte (8-bit unsigned integer) του οποίου η τιμή προστίθεται με το 15 για να δώσει το τελικό μέγεθος της τιμής, επιτρέποντας έτσι μεγέθη έως 270 bytes.

Το μέγεθος και ο τύπος του κάθε option value εξαρτάται από το κάθε option και το μέγεθος μπορεί να είναι μεταβλητό. Τα format που υποστηρίζονται είναι τα εξής:

- **uint:** Unsigned Integer, δηλαδή μη-αρνητικός ακέραιος αριθμός.

- **string:** Unicode String με κωδικοποίηση UTF-8 Net-Unicode [17]. Σημειώνεται επίσης, ότι ASCII Strings είναι συμβατά με αυτή την κωδικοποίηση.

- **opaque:** Μία ακολουθία από Byte χωρίς κάποια κωδικοποίηση.

4.4 Μοντέλο μηνυμάτων

Το CoAP όντας βασισμένο σε UDP μηνύματα, δεν μπορεί να επιβεβαιώσει από το επίπεδο του δικτύου ότι ένα μήνυμα παραδόθηκε ή χάθηκε, πότε πρέπει να το ξαναστείλει, αν τα μηνύματα έφτασαν με λάθος σειρά ή αν παραλήφθηκε περισσότερες από μία φορές το ίδιο μήνυμα. Το κανάλι επικοινωνίας κρίνεται μη-αξιόπιστο και για το λόγο αυτό ορίζονται διάφοροι κανόνες για να λυθούν τέτοιου είδους προβλήματα, παρόλο που πολλές εφαρμογές δεν επηρεάζονται από αυτά. Έχει τα ακόλουθα βασικά χαρακτηριστικά:

- Απλό μοντέλο παύσης και αναμονής για την επαναμετάδοση των Confirmable μηνυμάτων, η οποία συμβαίνει σε εκθετικά αυξανόμενο χρόνο.
- Μηχανισμό εντοπισμού διπλότυπων μηνυμάτων για τα CON, NON.
- Υποστήριξη multicast.

4.4.1 Τύποι μηνύματος

Ο τύπος του μηνύματος προσδιορίζεται στο πεδίο T της κεφαλίδας.

- **Confirmable (CON):** Τα μηνύματα που χρειάζονται επιβεβαίωση μαρκάρονται ως CON. Όταν δεν χάνεται κανένα πακέτο στο δίκτυο, σε κάθε CON αντιστοιχεί ακριβώς ένα ACK ή RST.
- **NON-Confirmable (NON):** Τα μηνύματα αυτά δεν πρέπει να επιβεβαιωθούν, πάντα περιέχουν μια αίτηση ή μια απάντηση και δεν μπορούν να είναι ποτέ άδεια.
- **Acknowledgement (ACK):** Δηλώνει ότι ένα συγκεκριμένο CON μήνυμα παραλήφθηκε. Δεν φέρει πληροφορία για το αν η αίτηση ήταν επιτυχής ή απέτυχε.
- **Reset (RST):** Δηλώνει ότι το μήνυμα που παραλήφθηκε είχε κάποιο πρόβλημα και δεν μπόρεσε να το επεξεργαστεί. (πχ μη συμβατή έκδοση CoAP)

4.4.2 Αξιόπιστα μηνύματα

Σχεδιάστηκε ένας αξιόπιστος μηχανισμός διαχείρισης των μηνυμάτων χωρίς όμως να επιχειρεί να αντιγράψει το TCP, καθώς κάτι τέτοιο δεν θα είχε ιδιαίτερο νόημα. Η αξιόπιστη μετάδοση ενός μηνύματος γίνεται αν αυτό είναι τύπου CON. Ο παραλήπτης πρέπει να επιβεβαιώσει την λήψη του απαντώντας με ένα ACK μήνυμα ή αν δεν μπορεί να καταλάβει και να επεξεργαστεί το μήνυμα να στείλει RST. Αν ο αποστολέας δεν λάβει κάποια απάντηση μέσα σε ένα συγκεκριμένο χρονικό διάστημα, ξαναστέλνει το μήνυμα και αυτή την φορά θα περιμένει περισσότερη ώρα για απάντηση από τον παραλήπτη. Όταν εξαντλήσει κάποιο προσδιορισμένο αριθμό προσπαθειών, εγκαταλείπει. Ο αποστολέας πρέπει να κρατάει μία λίστα με τα μηνύματα που περιμένει για επιβεβαίωση, πόσες φορές έχει επιχειρήσει να στείλει το καθένα. Ο αρχικός χρόνος αναμονής (timeout) διπλασιάζεται μετά από κάθε αποτυχημένη προσπάθεια.

Η σχέση των ACK, RST με το CON μήνυμα γίνεται μέσω του Message ID και της διεύθυνσης προέλευσης. Η απάντηση πρέπει να φέρει πάντα το ίδιο Message ID ώστε να ταυτιστεί με ένα μήνυμα που στάλθηκε και να διαγραφεί από την λίστα των μηνυμάτων που αναμένεται η επιβεβαίωσή τους. Το Message ID παράγεται από τον αποστολέα και περιέχεται στο μήνυμα. Η πιο συνηθισμένη πρακτική είναι να αρχικοποιείται κάποιο αρχικό Message ID με κάποια συνάρτηση τυχαίου αριθμού και στην συνέχεια για κάθε νέα αποστολή να αυξάνεται κατά ένα. Σε καμία περίπτωση το ίδιο Message ID δεν πρέπει να ξαναχρησιμοποιηθεί στο παράθυρο αποστολής, δηλαδή τον χρόνο της χειρότερης περίπτωσης που θα κάνει ένα CON μήνυμα να επιβεβαιωθεί, για να μην υπάρχει σύγκρουση δύο διαφορετικών μηνυμάτων με το ίδιο Message ID.

Ο παραλήπτης πρέπει να είναι σε θέση να λάβει ένα CON μήνυμα πολλές φορές (σε περίπτωση πχ που το ACK δεν μπόρεσε να φτάσει). Για κάθε διπλότυπο μήνυμα μπορεί να απαντήσει με το ίδιο ακριβώς ACK χωρίς να επεξεργαστεί ξανά το μήνυμα. Αυτός ο κανόνας βέβαια μπορεί να παραβλεφθεί ανάλογα με τις ανάγκες και τις δυνατότητες της συσκευής. Για παράδειγμα, μια συσκευή είναι πιο εύκολο να επεξεργαστεί ξανά ένα απλό POST μήνυμα αλλαγής μιας απλής μεταβλητής, παρά να κρατάει δομές για διπλότυπα μηνύματα.

4.4.3 Μη-αξιόπιστα μηνύματα

Τα αναξιόπιστα μηνύματα (NON) στέλνονται δίχως ο αποστολέας να μπορεί στο επίπεδο του CoAP να εγγυηθεί την παράδοσή τους. Είναι μια πιο απλή μέθοδος αποστολής, καθώς δεν χρειάζεται καμία δομή για την παρακολούθηση των μηνυμάτων. Ο παραλήπτης αν δεν μπορεί να επεξεργαστεί το μήνυμα μπορεί να το απορρίψει με ένα RST αλλιώς το αγνοεί. Μπορεί να γίνει πολλαπλή αποστολή του ίδιου μηνύματος, αυξάνοντας τις πιθανότητες παράδοσης και ο παραλήπτης

να αγνοήσει τα διπλότυπα, επεξεργαζόμενος μόνο μια φορά το μήνυμα. Όπως και πριν, ο κανόνας αυτός μπορεί να παραβλεφθεί.

4.5Κανόνες Αίτησης/Απάντησης

Το CoAP λειτουργεί με ένα παρόμοιο τρόπο με το HTTP. Ένας κόμβος στον ρόλο του πελάτη στέλνει αιτήσεις προς έναν διακομιστή για ένα συγκεκριμένο αντικείμενο, ο οποίος αναλαμβάνει να απαντήσει. Σε αντίθεση με το HTTP δεν εγκαθιδρύεται κάποια σύνδεση μέσω της οποίας γίνεται η ανταλλαγή μηνυμάτων, αλλά γίνεται ασύγχρονα. Στην ενότητα αυτή παρουσιάζονται όλοι οι κανόνες αίτησης/απάντησης του CoAP και πως πρέπει να αντιμετωπίζονται κάποιες καταστάσεις από τους πελάτες και τους διακομιστές.

4.5.1 Αιτήσεις

Μία αίτηση αποτελείται από την μέθοδο της αίτησης που περιέχεται στην κεφαλίδα, το μοναδικό αναγνωριστικό του αντικειμένου στις επιλογές, το payload με τον ορισμό του τύπου του (αν υπάρχει) και άλλες επιλογές ανάλογα με την αίτηση.

Το CoAP υποστηρίζει 4 βασικές μεθόδους, GET, POST, PUT, DELETE, συγκρίσιμα με αυτά του HTTP. Ανάλογα με τα χαρακτηριστικά τους, κρίνονται αν είναι ασφαλείς (safe) και αυτοδύναμες (idempotent). Ασφαλείς κρίνονται μόνο όσες κάνουν αποκλειστικά ανάκτηση δεδομένων, ενώ αυτοδύναμες κρίνονται αυτές που όσες φορές και αν κληθούν με τα ίδια χαρακτηριστικά θα έχουν το ίδιο ακριβώς αποτέλεσμα.

Μία αίτηση με μη-αναγνωρίσιμη ή μη-υλοποιημένη μέθοδο αίτησης επιστρέφει πάντα τον κωδικό απάντησης 4.05 – Method-Not-Allowed.

4.5.2 Απαντήσεις

Αφού λάβει μία αίτηση ένας διακομιστής, διερμηνεύει το μήνυμα και απαντάει σε αυτήν. Ένα μήνυμα απάντηση αναγνωρίζεται από τον αν στο πεδίο Code της κεφαλίδας, περιέχεται κωδικός απάντησης. Σε αντιστοιχία με το HTTP, ο κωδικός αυτός δηλώνει κατά πόσο ο διακομιστής μπόρεσε να καταλάβει το μήνυμα και να το ικανοποιήσει. Οι κωδικοί χωρίζονται σε τρεις υποκατηγορίες ανάλογα με το πρώτο ψηφίο:

- **2.xx:** Δηλώνει ότι η αίτηση εξυπηρετήθηκε σωστά και αναφέρει πιο ήταν το αποτέλεσμά της.

- **4.xx:** Δηλώνει ότι η αίτηση είχε κάποιο πρόβλημα και δεν μπόρεσε να εξυπηρετηθεί δηλώνοντας την αιτία.

- **5.xx:** Δηλώνει ότι παρόλο που η αίτηση ήταν σωστή, ο διακομιστής αδυνατεί να την εξυπηρετήσει.

Η πλήρης περιγραφή τους βρίσκεται στην ενότητα 4.5.7.

Τα μηνύματα απάντησης μπορούν να σταλούν με διάφορους τρόπους:

- Στις περισσότερες περιπτώσεις, η απάντηση είναι εμφωλευμένη στο ACK (εφόσον η αίτηση ήταν τύπου CON). Η μέθοδος αυτή ονομάζεται Piggy-backed. Η απάντηση περιέχεται ανεξάρτητα από το αν η εξυπηρέτηση της αίτησης ήταν επιτυχημένη ή όχι. Αυτό έχει ως αποτέλεσμα να στέλνεται ένα λιγότερο μήνυμα, με το μήνυμα απάντησης να είναι εμφωλευμένο στο μήνυμα επιβεβαίωσης παράδοσης.

- Υπάρχουν φορές που αυτός ο τρόπος δεν είναι διαθέσιμος, για παράδειγμα, ο διακομιστής πρέπει να εκτελέσει μια ενέργεια που θα διαρκέσει ώρα και για να μην καθυστερεί το μήνυμα επιβεβαίωσης, το στέλνει και στέλνει αργότερα την απάντηση, όταν αυτή γίνει διαθέσιμη. Στην περίπτωση αυτή, αφού ο πελάτης λάβει την επιβεβαίωση, αναμένει την απάντηση. Για να βεβαιωθεί όμως ο διακομιστής ότι η απάντηση έφτασε, στέλνεται ως CON μήνυμα για να είναι αξιόπιστο.

- Αν η αίτηση ήταν τύπου NON τότε ο διακομιστής επιστρέφει την απάντηση επίσης σε τύπου NON μήνυμα. Παρόλα αυτά όμως, ένας κόμβος θα πρέπει να είναι έτοιμος να λάβει NON απάντηση ακόμα και αν έστειλε CON αίτηση και το αντίθετο.

4.5.3Options

Τόσο τα μηνύματα αιτήσεων, όσο και τα μηνύματα απάντησης μπορεί να περιέχουν κάποια option με επιπλέον πληροφορίες για το μήνυμα. Ορίζονται τα option όπως φαίνονται στον πίνακα 4.1. Να σημειωθεί ότι δεν έχουν όλα τα option νόημα με όλες τις μεθόδους αίτησης και απάντησης.

Κάθε option ανήκει σε μία από τις δύο κατηγορίες, σημαντικό (critical) ή επιλεκτικό (elective). Η διαφορά μεταξύ των δύο έγκειται στο πως αντιδρά ένας κόμβος όταν λαμβάνει ένα μη-αναγνωρίσιμο option:

- Αν αυτό είναι επιλεκτικό, τότε μπορεί να το αγνοήσει και να συνεχίσει στην διερμηνεία του μηνύματος.

- Αν είναι σημαντικό και περιέχεται σε αξιόπιστη αίτηση, τότε η απάντηση πρέπει να έχει κωδικό 4.02 – Bad Option. Αν είναι δυνατόν

εμπεριέχεται στο payload και μία περιγραφή του σφάλματος σε ανθρώπινα αναγνώσιμη μορφή.

- Αν είναι σημαντικό και περιέχεται σε αξιόπιστη απάντηση, τότε αυτό θα έπρεπε να απορριφθεί με ένα RST μήνυμα.

- Αν είναι σημαντικό και περιέχεται σε μη-αξιόπιστο μήνυμα, τότε αυτό πρέπει να αγνοηθεί.

Σε όποια και από τις δύο κατηγορίες και αν ανήκει, ένα option δεν είναι υποχρεωτικό. Ο παραπάνω διαχωρισμός υπάρχει ώστε οι εκάστοτε υλοποιήσεις να μπορούν να απορρίπτουν options που δεν έχουν υλοποιηθεί ή δεν μπορούν να ερμηνευθούν.

No.	**C/E**	**Όνομα**	**Τύπος**	**Μέγεθος**	**Προεπιλογή**
1	Critical	Content-Type	uint	0-2 B	--
2	Elective	Max-Age	uint	0-4 B	60
3	Critical	Proxy-Uri	string	1-270 B	--
4	Elective	ETag	opaque	1-8 B	--
5	Critical	Uri-Host	string	1-270 B	
6	Elective	Location-Path	string	1-270 B	--
7	Critical	Uri-Port	uint	0-2 B	5683
8	Elective	Location-Query	string	1-270 B	--
9	Critical	Uri-Path	string	1-270 B	--
11	Critical	Token	opaque	1-8 B	άδειο
12	Elective	Accept	uint	0-2 B	--
13	Critical	If-Match	opaque	0-8 B	--
15	Critical	Uri-Query	string	1-270 B	--
21	Critical	If-None-Match	--	0 B	--

Πίνακας 4-1 CoAP Options

Υπάρχουν συγκεκριμένα άνω και κάτω όρια για το μέγεθος της τιμής του option και αν αυτή είναι εκτός των ορίων αυτών, θα πρέπει να μαρκάρεται ως μη αναγνωρίσιμο και να ακολουθεί τον προηγούμενο κανόνα. Επίσης υπάρχει μια προκαθορισμένη τιμή για μερικά option, η οποία δεν είναι υποχρεωτικό να στέλνεται και θεωρείται δεδομένη αν δεν περιέχεται στο μήνυμα.

Τέλος, μερικά option μπορούν να περιέχονται παραπάνω από μία φορές στο μήνυμα, ενώ άλλα αποκλειστικά μία φορά το πολύ.

Αναλυτικά:

- **Token:** Χρησιμοποιείται για την αντιστοίχηση αιτήσεων – απαντήσεων. Κάθε αίτηση μπορεί να έχει ένα token που παράγεται από τον πελάτη και ο διακομιστής πρέπει να το συμπεριλάβει αυτούσιο στην απάντησή του. Η προεπιλογή είναι η κενή τιμή στην περίπτωση απουσίας του option. Με το token ο πελάτης μπορεί να ξεχωρίζει τις απαντήσεις διαφόρων αιτήσεων που συνέβησαν ταυτόχρονα. Για τον λόγο αυτό κάθε ταυτόχρονη αίτηση από τον ίδιο πελάτη προς τον ίδιο διακομιστή πρέπει να έχει διαφορετικό token, ενώ αν δεν υπάρχει ταυτόχρονη αποστολή αιτήσεων το option μπορεί να παραλειφθεί και να θεωρηθεί η κενή τιμή.

  Είναι σημαντικό option και δεν πρέπει να υπάρχει πάνω από μια φορά.

- **Uri-Host, Uri-Port, Uri-Path, Uri-Query:** Τα option αυτά χρησιμοποιούνται για να δηλώσουν την διεύθυνση του αντικειμένου που ζητείται. Το Uri-Host δηλώνει την διεύθυνση του διακομιστή στο internet. Το Uri-Port, την συγκεκριμένη θύρα του διακομιστή. Το Uri-Path δηλώνει την ακριβή διεύθυνση και το όνομα του αντικειμένου εντός του διακομιστή. Για την δημιουργία της συνολικής διεύθυνσης, ενώνονται τα Uri-Path μαζί με "/". Τέλος το Uri-Query, για να δηλωθούν τυχόν ορίσματα που θα παραμετροποιήσουν το αντικείμενο. Οι προεπιλεγμένες τιμές για τα Uri-Host και Uri-Port είναι αυτές που χρησιμοποιούνται από το UDP (destination address, port) και δεν χρειάζονται στο μήνυμα τις περισσότερες φορές, εκτός και αν ο κόμβος έχει πολλούς εικονικούς διακομιστές, όπου η χρήση τους κρίνεται απαραίτητη.

  Ο συνδυασμός τους δημιουργεί την διεύθυνση παρόμοια με το HTTP:

  ```
  "coap:" "//" host [ ":" port ] path [ "?" query ]
  ```

  Παράδειγμα:

  ```
  coap://uberdust.cti.gr:5683/.well-known/core
  ```

  Όπου "`uberdust.cti.gr`" είναι το Uri-Host, 5683 το Uri-Port, "`.well-known`" και "`core`" τα δύο Uri-Path και δεν υπάρχει Uri-Query.

  Τα option αυτά είναι σημαντικά. Τα Uri-Host και Uri-Port πρέπει να υπάρχουν το πολύ μία φορά, ενώ τα Uri-Path και Uri-Query μπορούν να υπάρχουν περισσότερες και από μία φορές.

- **Proxy-Uri:** Χρησιμοποιείται σε αιτήσεις προς αντιπροσώπους, οι οποίοι θα απαντήσουν απευθείας από κάποιο έγκυρο αντίγραφο στην κρυφή μνήμη, ή θα προωθήσουν την αίτηση σε άλλο αντιπρόσωπο ή στον αρχικό διακομιστή. Ένας κόμβος που λαμβάνει τέτοιες αιτήσεις αλλά δεν λειτουργεί ως αντιπρόσωπος πρέπει να επιστρέψει το σφάλμα 5.05 – Proxying Not Supported.

  Το option αυτό είναι σημαντικό και μπορεί να υπάρχει παραπάνω από μια φορές σε ένα μήνυμα και έχει υψηλότερη σημασία από τα Uri-Host, Uri-Port, Uri-Path και Uri-Query, τα οποία δεν θα πρέπει να υπάρχουν στο ίδιο μήνυμα.

- **Content-Type:** Δηλώνει τον τύπο του αντικειμένου που έχει συμπεριληφθεί στο payload, είναι σημαντικό και δεν πρέπει να υπάρχει πάνω από μία φορά.

- **Accept:** Συμπεριλαμβάνεται σε αιτήσεις ώστε να δηλώσει ποιοι τύποι αντικειμένων γίνονται δεκτοί από τον πελάτη. Αν ο διακομιστής δεν μπορεί να ικανοποιήσει τον πελάτη επιστρέφει 4.06 – Not Acceptable. Αν δεν υπάρχει στην αίτηση το option τότε δεν χρησιμοποιείται κάποια προεπιλεγμένη τιμή, αλλά ο διακομιστής αποφασίζει τι τύπο θα στείλει.

  Είναι επιλεκτικό και μπορεί να υπάρχει πολλές φορές.

- **Max-Age:** Δηλώνει τον μέγιστο χρόνο (σε δευτερόλεπτα) για τον οποίο το αντικείμενο μπορεί να θεωρηθεί πρόσφατο. Η προεπιλεγμένη τιμή είναι 60 δευτερόλεπτα και θεωρείται δεδομένη σε περίπτωση απουσίας του option.

  Είναι επιλεκτικό και μπορεί να υπάρχει μόνο μια φορά.

- **ETag:** Το option αυτό σε μία απάντηση δηλώνει την τρέχουσα τιμή του entity-tag του αντικειμένου που περιέχεται σε αυτήν. Η χρήση του entity-tag είναι να μπορεί να διακρίνεται από το σύστημα αν το αντικείμενο έχει μεταβληθεί στον χρόνο. Δημιουργείται με διάφορους τρόπους, όπως hash, checksum, έκδοση. Κάθε κόμβος που λαμβάνει κάποιο entity-tag πρέπει να το χειρίζεται αυτούσιο και να μην θεωρεί κάποιο τύπο δεδομένων. Αν έχει κρυφή μνήμη για τις απαντήσεις, αποθηκεύει και το ETag μέσω του οποίου θα μπορέσει αργότερα να επικυρώσει αν το αντικείμενο είναι πρόσφατο ή όχι.

  Είναι επιλεκτικό, μπορεί να υπάρχει μόνο μια φορά σε απάντηση, αλλά περισσότερες από μία σε αίτηση.

- **Location-Path, Location-Query:** Δηλώνουν την διεύθυνση στην οποία δημιουργήθηκε ένα νέο αντικείμενο μετά από μία POST αίτηση.

Αν η απάντηση που τα περιέχει περάσει από κρυφή μνήμη η οποία έχει αποθηκευμένο κάποιο αντικείμενο με αυτό το URI, τότε αυτό μαρκάρεται ως μη-πρόσφατο.

Είναι επιλεκτικά και μπορούν να υπάρχουν πάνω από μία φορές.

- **If-Match:** Χρησιμοποιείται σαν συνθήκη σε αιτήσεις. Η τιμή μπορεί να είναι ένα ETag ή κενό. Στην 1η περίπτωση, δηλώνει στον διακομιστή ότι αν για το ETag του συγκεκριμένου αντικειμένου της αίτησης ταυτίζεται με τον διακομιστή, τότε η αίτηση μπορεί να προχωρήσει και να εξυπηρετηθεί. Στην περίπτωση της κενής τιμής, απαιτείται απλά η ύπαρξη του αντικειμένου. Αν η συνθήκη δεν ισχύει τότε ο διακομιστής επιστρέφει 4.12 – Precondition Failed και δεν θα εκτελέσει την αίτηση. Σε κάθε περίπτωση όμως, αν η αίτηση κανονικά (χωρίς το If-Match) επέστρεφε κάτι άλλο πέρα από 2.xx ή 4.12, τότε το option πρέπει να αγνοηθεί.

  Είναι σημαντικό και μπορεί να υπάρχει πάνω από μία φορά.

- **If-None-Match:** Χρησιμοποιείται σε αιτήσεις που θα δημιουργήσουν νέο αντικείμενο στον διακομιστή και βάζει την συνθήκη ότι αυτό θα δημιουργηθεί μόνο αν δεν υπάρχει ήδη αντικείμενο με το ίδιο URI, αποφεύγοντας έτσι τυχαία overwrite. Αν η συνθήκη δεν ισχύει, η αίτηση δεν εξυπηρετείται και επιστρέφεται 4.12 – Precondition Failed.

  Είναι σημαντικό και μπορεί να υπάρχει μόνο μία φορά.

4.5.4 Payload

Τα μηνύματα αίτησης και απάντησης μπορεί να περιέχουν κάποιο payload, ανάλογα την μέθοδο αίτησης και απάντησης αντίστοιχα. Μέθοδοι αίτησης με payload είναι οι PUT και POST, ενώ μέθοδοι απάντησης με payload είναι 2.05 – Content και οι μέθοδοι σφάλματος. Κάποιες φορές εισάγεται payload και στις μεθόδους απάντησης 2.01 – Created, 2.02 – Deleted και 2.04 – Changed.

Σε κάθε περίπτωση το είδος του περιεχομένου δηλώνεται από το option Content-Type χωρίς να υπάρχει κάποια προεπιλεγμένη τιμή στην περίπτωση απουσίας του. Εξαίρεση αποτελούν τα payload που περιέχουν τα σφάλματα, όπου ο τύπος είναι πάντα UTF-8 Net-Unicode και δεν είναι αναγκαίο να δηλώνεται ως option.

Αν μία μέθοδος αίτησης ή απάντησης δεν έχει οριστεί να δέχεται payload, τότε ο αποστολέας δεν θα έπρεπε να βάζει και ο παραλήπτης πρέπει να το αγνοεί.

4.5.5Κρυφές μνήμες και αντιπρόσωποι

Οι CoAP κόμβοι μπορούν να αποθηκεύουν σε κρυφές μνήμες τις απαντήσεις ώστε να μειώσουν στο μέλλον τον χρόνο απάντησης και την χρήση του δικτύου σε ίδιες αιτήσεις. Ο στόχος είναι να αποθηκεύεται μία απάντηση ώστε να χρησιμοποιηθεί σε μια μελλοντική αίτηση προς το ίδιο αντικείμενο. Ένας μηχανισμός που επιβεβαιώνει κατά πόσο η αποθηκευμένη απάντηση είναι πρόσφατη ("freshness" mechanism) χρησιμοποιείται για αυτό το σκοπό. Ακόμα και όταν μία νέα απάντηση απαιτείται, μπορεί να χρησιμοποιηθεί το payload της αποθηκευμένης, με έναν μηχανισμό επικύρωσης (validation).

Σε αντίθεση με το HTTP, κατά πόσο μία απάντηση είναι δυνατόν να αποθηκευτεί στην κρυφή μνήμη δεν κρίνεται από την μέθοδο αίτησης, αλλά από τον κωδικό απάντησης. Στην ανάλυση των κωδικών απάντησης στην ενότητα 4.5.7 δηλώνεται αν μπορεί να χρησιμοποιηθεί η κάθε μια σε κρυφή μνήμη. Ένας κόμβος δεν μπορεί να απαντήσει χρησιμοποιώντας την απάντηση της κρυφής μνήμης εκτός αν:

- Η αρχική αίτηση που η απάντησή της μπήκε στην κρυφή μνήμη, είναι ίδια με την νέα αίτηση.
- Όλα τα option μεταξύ των δύο αιτήσεων πρέπει να είναι ίδια με την εξαίρεση των Token, Max-Age και ETag.
- Η αποθηκευμένη απάντηση είναι πρόσφατη ή επικυρωμένη όπως ορίζεται στην συνέχεια.

Όταν μια απάντηση στην κρυφή μνήμη είναι πρόσφατη μπορεί να χρησιμοποιηθεί αντί να προωθηθεί η αίτηση στον αρχικό διακομιστή. Αυτό επιτυγχάνεται με τον ορισμό του Max-Age option στην απάντηση από τον διακομιστή. Η τιμή δηλώνει για πόσα δευτερόλεπτα μπορεί να θεωρείται πρόσφατη η απάντηση. Αν ο διακομιστής δεν επιθυμεί οι απαντήσεις που στέλνει να αποθηκεύονται σε κρυφές μνήμες, πρέπει να ορίσει τον Max-Age με τιμή μηδέν.

Όταν η αποθηκευμένη απάντηση δεν είναι πρόσφατη για μια αίτηση τύπου GET, μπορεί να γίνει χρήση του ETag option, δίνοντας την δυνατότητα στον διακομιστή να επιλέξει μια αποθηκευμένη απάντηση και να την μαρκάρει ως πρόσφατη ξανά. Αυτό ονομάζεται μηχανισμός επικύρωσης. Αν ο διακομιστής επιστρέψει μέθοδο απάντησης 2.03 – Valid δηλώνει ότι η αποθηκευμένη απάντηση με το συγκεκριμένο ETag μπορεί να ξαναχρησιμοποιηθεί για όσα ακόμα δευτερόλεπτα δηλώνει το Max-Age. Σε οποιαδήποτε άλλη περίπτωση καμία αποθηκευμένη απάντηση δεν επικυρώθηκε και ο διακομιστής θα έχει απαντήσει με το αντικείμενο που ζητήθηκε.

Παρόμοια με το HTTP ορίζονται κόμβοι που μπορούν να λειτουργούν ως αντιπρόσωποι (proxies), ο οποίοι αναλαμβάνουν να εξυπηρετήσουν μία αίτηση εκ μέρους του αρχικού διακομιστή και συχνά έχουν κρυφή μνήμη ώστε να απαντάνε στις αιτήσεις κατευθείαν. Η διαφορά εδώ είναι ότι στο CoAP οι αιτήσεις προς

αντιπροσώπους διαφέρουν από τις κανονικές. Εδώ γίνεται χρήση του Proxy-Uri option για να δηλωθεί το αντικείμενο που ζητείται. Αν μία τέτοια αίτηση γίνει προς έναν κόμβο που δεν μπορεί να δράσει ως αντιπρόσωπος τότε απορρίπτει την αίτηση με 5.05 – Proxying Not Supported. Οι αντιπρόσωποι πρέπει να είναι σε θέση να αναλύσουν το Proxy-Uri option στα αντίστοιχα Uri-Host, Uri-Port, Uri-Path, Uri-Query. Όλα τα option της αίτησης, ακολουθούν τους ίδιους κανόνες, μόνο που αν δεν αναγνωρίζονται, δεν προωθούνται στον αρχικό διακομιστή και αν είναι σημαντικά απορρίπτονται από τον αντιπρόσωπο με 4.02 – Bad Option.

4.5.6 Μέθοδοι αίτησης

Αναλυτικά οι μέθοδοι αίτησης και τι δηλώνει η κάθε μία:

- **GET:** Επιστρέφει το αντικείμενο που ζητήθηκε μέσω του URI της αίτησης. Αν η αίτηση περιέχει ένα ή περισσότερα Accept option, δηλώνει ποια Content-Type προτιμώνται στην απάντηση. Αν υπάρχει ETag option τότε αυτό επικυρώνεται και μόνο σε περίπτωση αποτυχίας αποστέλλεται το αντικείμενο. Σε περίπτωση επιτυχία επιστρέφει `2.05` – Content ή `2.03` Valid.

  Η μέθοδος είναι ασφαλής και αυτοδύναμη.

- **POST:** Στέλνεται ένα αντικείμενο στον διακομιστή και πρέπει να επεξεργαστεί από αυτόν. Η ακριβής λειτουργεία εξαρτάται από τον διακομιστή αλλά συνήθως ένα αντικείμενό του ανανεώνεται ή δημιουργείται. Στην 1η περίπτωση η απάντηση είναι 2.04 – Changed, ενώ στην 2η 2.01 – Created και στο URI του αντικειμένου που δημιουργήθηκε στο Location-Path option της απάντησης.

  Η μέθοδος δεν είναι ούτε ασφαλής ούτε αυτοδύναμη.

- **PUT:** Στέλνεται ένα αντικείμενο προς ένα URI και ο τύπος του δηλώνεται με το Content-Type option. Επιστρέφει `2.01` – Created σε περίπτωση δημιουργίας νέου αντικειμένου και `2.04` – Changed σε περίπτωση ανανέωσης, ενώ στην αποτυχία επιστρέφει το ανάλογο σφάλμα.

  Η μέθοδος δεν είναι ασφαλής αλλά είναι αυτοδύναμη.

- **DELETE:** Το αντικείμενο που ορίζεται από το URI θα σβηστεί και η μέθοδος θα επιστρέψει `2.02` – Deleted σε περίπτωση επιτυχίας ή αν το αντικείμενο δεν υπήρχε εξ αρχής.

  Η μέθοδος δεν είναι ασφαλής αλλά είναι αυτοδύναμη.

4.5.7 Κωδικοί απάντησης

Κωδικοί που δηλώνουν επιτυχία, 2.xx:

- **2.01 Created:** Χρησιμοποιείται μόνο σε απαντήσεις POST και PUT αιτήσεων. Στο payload, αν υπάρχει, περιέχεται το αντικείμενο που δημιουργήθηκε και ο τύπος του δηλώνεται στο Content-Type. Αν στην απάντηση περιέχεται ένα ή περισσότερα Location-Path options, τότε οι τιμές αυτών δηλώνουν την τοποθεσία που δημιουργήθηκε το αντικείμενο, διαφορετικά, αυτό δημιουργήθηκε στο URI της αίτησης.

  Δεν μπορεί να αποθηκευτεί σε κρυφή μνήμη.

- **2.02 Deleted:** Χρησιμοποιείται μόνο ως απάντηση σε επιτυχείς DELETE αιτήσεις, ενώ δεν περιέχεται κάποιο payload.

  Η απάντηση δεν μπορεί να αποθηκευτεί σε κρυφή μνήμη, παρόλα αυτά μπορούν να μαρκαριστούν όλα τα αποθηκευμένα αντίτυπα του αντικειμένου που διαγράφτηκε ως μη-πρόσφατα.

- **2.03 Valid:** Δηλώνει ότι το αντικείμενο με το συγκεκριμένο ETag που περιέχεται στην απάντηση δεν έχει μεταβληθεί και είναι πρόσφατο.

  Η κρυφή μνήμη ανανεώνει την τιμή του Max-Age της αποθηκευμένης απάντησης με την νέα τιμή που περιεχόταν στην απάντηση.

- **2.04 Changed:** Χρησιμοποιείται ως απάντηση σε επιτυχείς PUT και POST αιτήσεις. Δεν είναι απαραίτητο να περιέχεται κάποιο payload, αλλά αν υπάρχει, είναι η παρουσίαση του αποτελέσματος της ενέργειας, με το Content-Type να δηλώνει τον τύπο του.

  Η απάντηση δεν μπορεί να αποθηκευτεί σε κρυφή μνήμη, παρόλα αυτά μπορούν να μαρκαριστούν όλα τα αποθηκευμένα αντίτυπα του αντικειμένου που διαγράφτηκε ως μη-πρόσφατα.

- **2.05 Content:** Χρησιμοποιείται σε απαντήσεις GET αιτήσεων. Στο payload, αν υπάρχει, περιέχεται το αντικείμενο που ζητήθηκε και ο τύπος του στο Content-Type.

  Μπορεί να αποθηκευτεί σε κρυφή μνήμη, με το Max-Age να καθορίζει για πόσα δευτερόλεπτα θα είναι πρόσφατο και το ETag, αν υπάρχει, για επικύρωση.

Κωδικοί που δηλώνουν εσφαλμένη αίτηση, 4.xx:

- **Bad Request:** Η αίτηση δεν μπορούσε να επεξεργασθεί από τον διακομιστή λόγο λανθασμένης σύνταξης του μηνύματος.
- **4.01 Unauthorized:** Ο πελάτης δεν έχει τα απαραίτητα δικαιώματα για την ενέργεια που ζήτησε.
- **4.02 Bad Option:** Ο διακομιστής δεν μπόρεσε να επεξεργαστεί ένα ή περισσότερα σημαντικά option στο μήνυμα της αίτησης.
- **4.03 Forbidden:** Ο διακομιστής αρνείται να πραγματοποιήσει την αίτηση και οποιαδήποτε προσπάθεια για εξουσιοδότηση δεν θα αλλάξει την κατάσταση.
- **4.04 Not Found:** Δεν βρέθηκε το ζητούμενο αντικείμενο στον διακομιστή.
- **4.05 Method Not Allowed:** Δεν είναι επιτρεπόμενη η συγκεκριμένη μέθοδος της αίτησης στο αντικείμενο.
- **4.06 Not Acceptable:** Δηλώνει ότι οι τύποι που ζητήθηκαν από την αίτηση μέσω του Accept option δεν είναι διαθέσιμοι από τον διακομιστή για το συγκεκριμένο αντικείμενο.
- **4.12 Precondition Failed:** Οι προϋποθέσεις που είχε ορίσει ο πελάτης δεν τηρούνταν και η αίτηση δεν πραγματοποιήθηκε.
- **4.13 Request Entity Too Large:** Ο διακομιστής δεν μπορεί να πραγματοποιήσει την αίτηση γιατί το αντικείμενο που στάλθηκε ήταν μεγαλύτερο από όσο μπορεί να δεχτεί.
- **4.15 Unsupported Media Type:** Το αντικείμενο που στάλθηκε είναι κάποιου τύπου που ο διακομιστής δεν υποστηρίζει.

Κωδικοί που δηλώνουν σφάλμα στον διακομιστή, 5.xx:

- **5.00 Internal Server Error:** Κάποιο απροσδιόριστο σφάλμα συνέβει στον διακομιστή στην προσπάθειά του να πραγματοποιήσει την αίτηση.
- **5.01 Not Implemented:** Δηλώνει ότι ο διακομιστής δεν υποστηρίζει τις απαραίτητες λειτουργίες που απαιτούνται για την πραγματοποίηση της αίτησης.
- **5.02 Bad Gateway:** Σε περίπτωση που ο αντιπρόσωπος έλαβε από τον ανώτερο διακομιστή κάποια μη-έγκυρη απάντηση στην προσπάθειά του να πραγματοποιήσει την αίτηση.
- **5.03 Service Unavailable:** Κάποιο προσωρινό πρόβλημα εμποδίζει τον διακομιστή να πραγματοποιήσει την αίτηση. Μέσω του Max-Age ορίζεται σε πόση ώρα ενδέχεται να είναι διαθέσιμος να την πραγματοποιήσει.
- **5.04 Gateway Timeout:** Ο αντιπρόσωπος δεν έλαβε απάντηση εντός του παραθύρου αναμονής από τον αρχικό διακομιστή.

- **5.05 Proxying Not Supported:** Ο διακομιστής δεν μπορεί να λειτουργήσει ως αντιπρόσωπος για το αντικείμενο που δηλώνεται στο Proxy-Uri option.

4.6 Ασφάλεια

Για την ασφαλή μετάδοση των μηνυμάτων είναι απαραίτητη η ύπαρξη κάποιου πρωτοκόλλου ασφαλείας. Ορίζονται τρία διαφορετικά προφίλ ασφαλείας, βασισμένα στο DTLS [18] και ένα στο IPsec [19]:

- **NoSec (No Security):** Δεν υπάρχει κάποιο πρωτόκολλο ασφαλείας, το DTLS είναι απενεργοποιημένο. Τα μηνύματα στέλνονται αυτούσια μέσω UDP στην προεπιλεγμένη θύρα του CoAP. Παρόλα αυτά, μπορεί να υπάρχει ασφαλής μετάδοση μέσω του IPsec που υλοποιείται στο επίπεδο του δικτύου. Στο επίπεδο του CoAP δεν υπάρχει η ανάγκη να υλοποιηθεί κάτι.
- **PreSharedKey:** Το DTSL είναι ενεργοποιημένο και υπάρχει μία έτοιμη λίστα με κλειδιά που έχει ήδη μοιραστεί στις συσκευές. Κάθε κλειδί δηλώνει με ποια συσκευή μπορεί να χρησιμοποιηθεί. Όταν μία σύνδεση δημιουργείται προς μία νέα συσκευή, επιλέγεται ένα κατάλληλο κλειδί για τις συσκευές που θέλει να επικοινωνήσει και στην συνέχεια, δημιουργεί την σύνδεση με DTLS χρησιμοποιώντας το PSK (Pre-Shared Key). Στην πιο ακραία περίπτωση αντιστοιχεί ένα κλειδί για κάθε συσκευή.
- **RawPublicKey:** Το DTSL είναι ενεργοποιημένο και η συσκευή έχει ένα ζευγάρι ασύμμετρων κλειδιών. Η συσκευή έχει επίσης μία ταυτότητα δημιουργημένη από το δημόσιο κλειδί και την λίστα των ταυτοτήτων των συσκευών που επιθυμεί να επικοινωνήσει.
- **Certificate:** Το DTSL είναι ενεργοποιημένο και η συσκευή έχει ένα ζευγάρι ασύμμετρων κλειδιών μαζί με ένα πιστοποιητικό X.509 [20] που το αντιστοιχεί με την ταυτότητα της συσκευής και είναι υπογεγραμμένο από κάποιο κόμβο κοινής εμπιστοσύνης, μία λίστα των οποίων οφείλει να έχει αποθηκευμένη για να επικυρώνει τα πιστοποιητικά.

4.7Επεκτάσεις

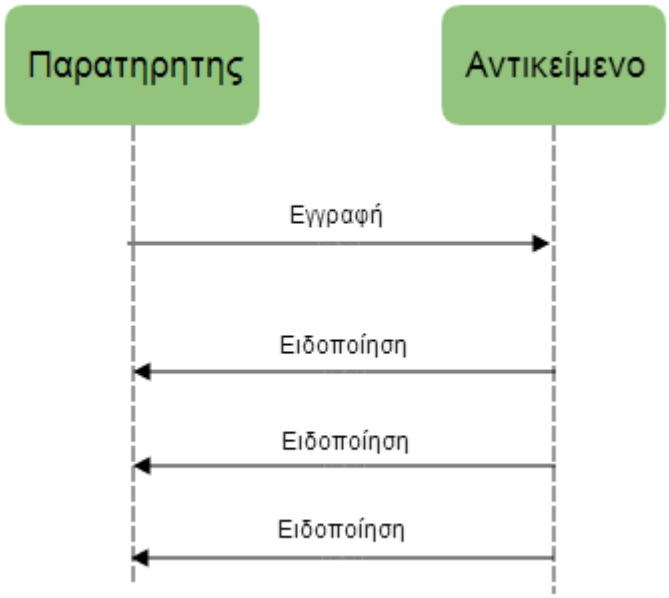
Το CoAP έχει διάφορες επεκτάσεις οι οποίες αναπτύσσονται παράλληλα με το κυρίως πρότυπο και εισάγουν κάποια διαφορετικά χαρακτηριστικά σε αυτό.

4.7.1 Observe

Μία πολύ βασική ανάγκη στα συστήματα παρακολούθησης αισθητήρων είναι η ανάγκη να έχουμε μια πρόσφατη ένδειξη του αισθητήρα. Η αποστολή αιτήσεων είναι όμως μη αποδοτική γιατί πολλές φορές η τιμή δεν έχει αλλάξει ακόμα ή δεν έχει συμβεί κάποιο γεγονός (πχ εντοπίστηκε κίνηση). Γενικά ο πελάτης δεν έχει γνώση για το κάθε πότε πρέπει να ζητάει νέα τιμή. Το CoAP Observe [21] πρότυπο δίνει την δυνατότητα στους ενδιαφερόμενους πελάτες να γραφτούν στον διακομιστή σαν παρατηρητές (observers) ενός αντικειμένου (πχ η έξοδος του αισθητήρα). Ο διακομιστής φροντίζει οι παρατηρητές να λάβουν την νέα τιμή όταν αυτή γίνει διαθέσιμη, έτσι αποφεύγονται περιττές αιτήσεις προς το σύστημα.

Ορίζεται ένα νέο CoAP Option, το Observe, για την ανάγκη αυτή, η τιμή του οποίου δηλώνει τον αριθμό της ειδοποίησης, ώστε να μπορεί ο πελάτης να καταλάβει αν κάποια ειδοποίηση έφτασε ετεροχρονισμένα σε αυτόν και έχει λάβει και πιο πρόσφατη ειδοποίηση άρα πρέπει να την απορρίψει. Η τιμή του στην αίτηση πρέπει να είναι μηδέν και αγνοείται από τον διακομιστή. Είναι επιλεκτικό option οπότε αν ο διακομιστής δεν μπορεί να εισάγει τον πελάτη στην λίστα με τους παρατηρητές, τότε απλά το αγνοεί και απαντάει στην αίτηση. Μπορεί να υπάρχει μόνο μία φορά σε ένα μήνυμα.

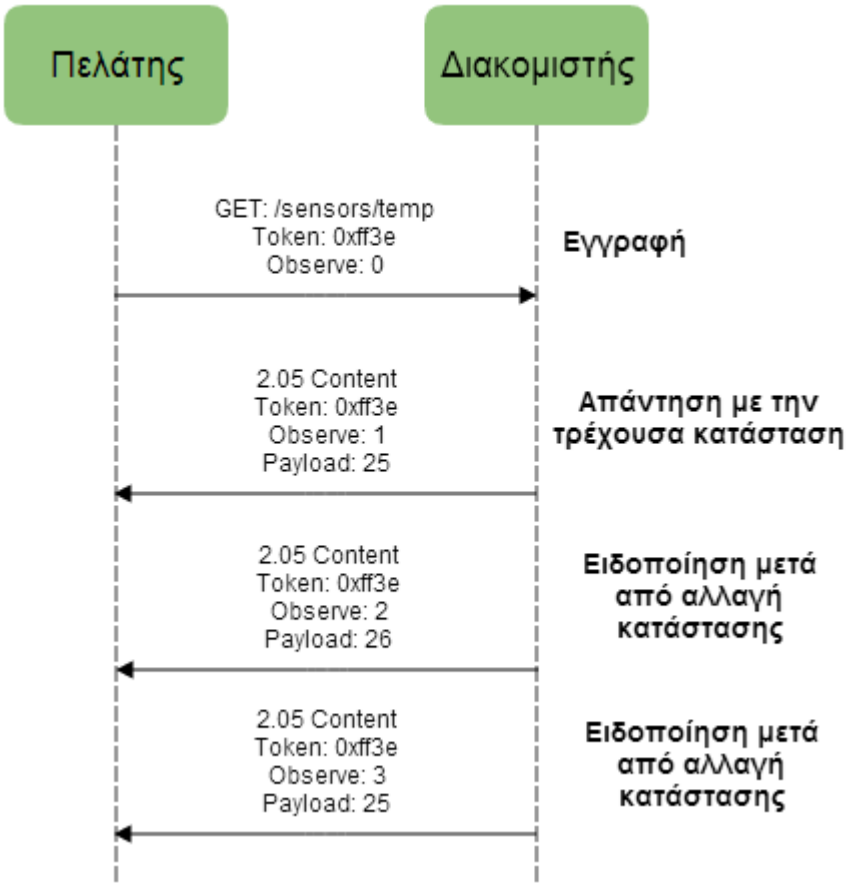
Στην εικόνα που ακολουθεί φαίνεται η λογική του Observe:



Εικόνα 4.3 CoAP Observe

- **Αντικείμενο:** Ένα CoAP αντικείμενο που βρίσκεται σε έναν διακομιστή και συχνά είναι η έξοδος κάποιου αισθητήρα. Η κατάστασή του μπορεί να αλλάξει, με την συχνότητα της αλλαγής να είναι κυμαινόμενη.
- **Παρατηρητής:** Ένας CoAP κόμβος που ενδιαφέρεται να λαμβάνει νέα κατάσταση του αντικειμένου.
- **Εγγραφή:** Η διαδικασία κατά την οποία ένας παρατηρητής δηλώνει το ενδιαφέρον στον διακομιστή στέλνοντας ένα διευρυμένο GET αίτημα.
- **Ειδοποίηση:** Όποτε μία καινούρια κατάσταση του αντικειμένου γίνεται διαθέσιμη, στέλνεται ειδοποίηση προς τους εγγεγραμμένους παρατηρητές.

Ακολουθεί παράδειγμα όπου φαίνεται το πρότυπο, αρχικά ο παρατηρητής στέλνει μία GET αίτηση προς ένα αντικείμενο, μαζί με το Observe option για να δηλώσει το ενδιαφέρον του να λαμβάνει ειδοποιήσεις, όπως επίσης με ένα Token που θα πρέπει να περιέχεται σε κάθε ειδοποίηση ώστε ο παρατηρητής να μπορεί να την συσχετίσει με την αρχική αίτηση.



Εικόνα 4.4 CoAP Observe παράδειγμα

Ένας κόμβος που στέλνει Observe GET αίτηση και λάβει απάντηση με μέθοδο 2.xx και το Observe option, τότε ξέρει ότι εγγράφτηκε στους παρατηρητές του διακομιστή και μπορεί να περιμένει ειδοποιήσεις.

Οι ειδοποιήσεις μπορούν να είναι είτε αξιόπιστα μηνύματα είτε μη-αξιόπιστα. Τα πρώτα ο πελάτης πρέπει να τα επιβεβαιώνει ώστε να δηλώσει στον διακομιστή ότι επιθυμεί να παραμείνει παρατηρητής. Αν η ειδοποίηση δεν μπορεί να αναγνωριστεί τότε απορρίπτεται με ένα RST μήνυμα. Αν μία ειδοποίηση δεν έχει επιβεβαιωθεί ακόμα και στο ενδιάμεσο προκύψει νέα ειδοποίηση, τότε ο διακομιστής σταματάει να προσπαθεί να στείλει την παλιά, αλλά κρατάει τον αριθμό των προσπαθειών που έγιναν και τον μεταφέρει στην καινούρια, ώστε μετά από τον προκαθορισμένο αριθμό μέγιστων προσπαθειών να διαγράψει τον παρατηρητή.

Όταν ο παρατηρητής δεν ενδιαφέρεται πλέον για ειδοποιήσεις μπορεί είτε να απορρίψει μία ειδοποίηση είτε να στείλει νέα GET αίτηση προς το ίδιο αντικείμενο χωρίς Observe option.

4.7.2 Resource Discovery

Το CoAP έχοντας σχεδιαστεί για M2M εφαρμογές, είναι απαραίτητο να υπάρχει ένας μηχανισμός που να επιτρέπει ένας κόμβος να μαθαίνει τα διαθέσιμα αντικείμενα ενός άλλου κόμβου. Το Resource Discovery πρότυπο ορίζει τον τρόπο με τον οποίο γίνεται αυτό, ώστε τα αντικείμενα με τις διευθύνσεις τους να γίνονται γνωστά.

Ορίζεται ένα αντικείμενο που βρίσκεται σε συγκεκριμένη θέση σε όποιον κόμβο υποστηρίζει το πρότυπο. Το URI είναι το "`/.well-known/core`" με το format του συγκεκριμένου να ορίζεται από το "CoRE Link Format" [22]. Το συγκεκριμένο αντικείμενο επιστρέφει τα διαθέσιμα αντικείμενα του διακομιστή, τα URI τους, καθώς και τυχόν γνωρίσματα (attributes):

- **Resource Type – rt:** Χρησιμοποιείται για να δηλώσει τον τύπο του αντικειμένου, μπορεί να είναι μία απλή λέξη (πχ OutdoorTemperature), ένα Universal Resource Name (URN) (πχ urn:temperature:outdoor) ή ακόμα και ένα URI προς ένα αντικείμενο που ορίζει αναλυτικά τον τύπο (πχ http://sweet.jpl.nasa.gov/2.0/phys.owl#Temperature). Μπορούν να υπάρχουν πολλά τέτοια γνωρίσματα σε ένα αντικείμενο.
- **Interface Description – if:** Παρέχει ένα όνομα, URI ή URN για την περιγραφή του αντικειμένου και πως θα αλληλεπιδράσει με αυτό. Είναι γενικότερο από το rt, για παράδειγμα τα rt "TemperatureC", "LightLux" μπορούν να ανήκουν στην γενικότερη κλάση του if "Sensor".
- **Maximum Size – sz:** Δηλώνει το εκτιμώμενο μέγιστο μέγεθος του αντικειμένου και περιέχεται συνήθως σε αντικείμενα που είναι μεγαλύτερα από το μέγεθος του πακέτου.

Τα URI των αντικειμένων βρίσκονται εντός “<” “>”, ακολουθούμενα από τα γνωρίσματα χωρισμένα με “;”, ενώ τα URI χωρίζονται με “,”. Παράδειγμα μιας απάντησης προς το “`/.well-known/core`”:

```
</sensors/temp>;rt="TemperatureC";if="sensor",

</sensors/light>;rt="LightLux";if="sensor"
```

4.7.3Block

Το μέγεθος του μηνύματος που μπορεί να στείλει ένας κόμβος περιορίζεται από το μέγεθος του πακέτου που υποστηρίζεται. Υπάρχουν περιπτώσεις που το μήνυμα που πρέπει να σταλεί να είναι μεγαλύτερο από το διαθέσιμο μέγεθος του πακέτου. Ορίζεται λοιπόν το Block extension [23] ώστε να μπορέσει ένα κόμβος να κατακερματίσει ένα μήνυμα στο επίπεδο του CoAP, στην περίπτωση που αυτό είναι απαραίτητο, και να το στείλει κατά μπλοκ. Η λειτουργία αυτή ορίζεται τόσο για τις αιτήσεις όσο και για τις απαντήσεις, αν και είναι πιο σύνηθες στις απαντήσεις. Γενικά τα χαρακτηριστικά που παρέχει είναι τα εξής:

- Μεταδόσεις μηνυμάτων μεγαλύτερων από το επιτρεπτό όριο του δικτύου ή της συσκευής γίνονται εφικτές.
- Η μετάδοση του κάθε μπλοκ επιβεβαιώνεται, ενώ γίνεται επαναμετάδοση του μπλοκ σε διαφορετική περίπτωση.
- Και οι δύο κόμβοι που εμπλέκονται σε μία μετάδοση κατά μπλοκ έχουν δικαίωμα να δηλώσουν την προτίμησή τους για το μέγεθος του μπλοκ.
- Δεν απαιτείται επιπλέον κόστος και προσπάθεια από τους κόμβους σε μια μπλοκ μετάδοση στο επίπεδο του CoAP, από ότι αν συνέβαινε στο IP επίπεδο.

Ορίζονται δύο επιπλέον option για το πρότυπο:

No.	C/E	Όνομα	Τύπος	Μέγεθος	Προεπιλογή
19	Critical	Block1	uint	1-3 B	0
17	Critical	Block2	uint	1-3 B	0

Πίνακας 4-2 CoAP Block Options

Το Block1 χρησιμοποιείται στις αιτήσεις που πρέπει να σταλούν κατά μπλοκ, ενώ το Block2 στις απαντήσεις. Για τα δύο αυτά option ισχύουν τα ίδια χαρακτηριστικά. Αν και η υλοποίηση του προτύπου είναι επιλεκτική, τα option είναι σημαντικά και δεν μπορούν να αγνοηθούν.

Η τιμή και των δύο έχει το εξής format ανάλογα το μέγεθος του NUM:

	Octet	0							
Octet	Bit	0	1	2	3	4	5	6	7
0	0	NUM				M	SZX		

	Octet	0								1							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	NUM												M	SZX		

	Octet	0								1								2							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	NUM																				M	SZX		

Εικόνα 4.5 Block1 και Block2 format με το NUM να είναι κυμαινόμενο

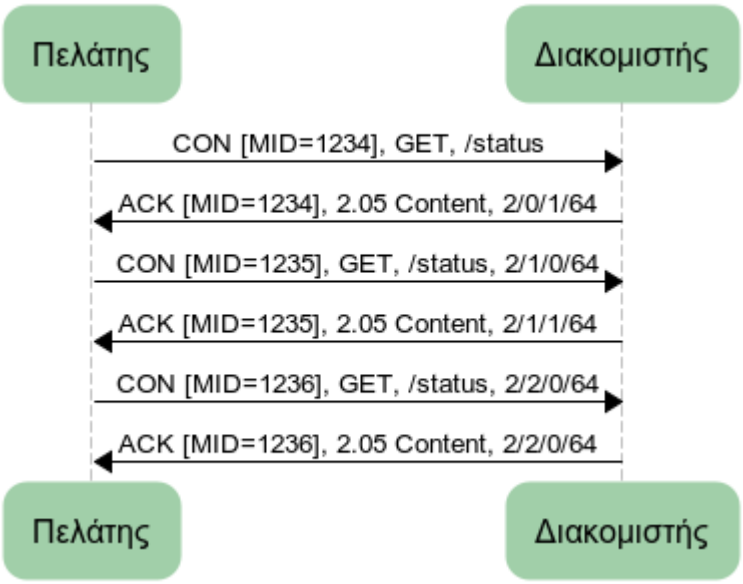
- **NUM – Block Number:** Δηλώνει τον αριθμό του μπλοκ που περιέχεται ή ζητείται στο payload. Το μέγεθός του είναι μεταβλητό (4, 12, ή 20 bit) και με μηδέν συμβολίζεται το πρώτο μπλοκ του αντικειμένου.
- **M – More Flag:** Είναι ένα bit και δηλώνει αν αυτό είναι ενδιάμεσο μπλοκ και έπονται και άλλα. Αν είναι μηδέν σημαίνει ότι αυτό είναι το τελευταίο μπλοκ.
- **SZX – Block Size:** Δηλώνει το μέγεθος του μπλοκ το οποίο υπολογίζεται με τον εξής τύπο: 2^(SZX+4), με το SZX να παίρνει τιμές μεταξύ 0 και 6, άρα το ελάχιστο μέγεθος του μπλοκ είναι 16 και το μέγιστο 1024.

Το Block2 option σε μία αίτηση δηλώνει πιο μπλοκ επιθυμεί από τον διακομιστή και προτείνει ένα μέγεθος μπλοκ σε περίπτωση που είναι η 1η αίτηση (δηλαδή ζητείται το μπλοκ 0) ή αντιγράφει το μέγεθος μπλοκ από την προηγούμενη απάντηση. Σε κάθε περίπτωση, στην αίτηση, το M flag πρέπει να είναι 0.

Το Block2 σε μία απάντηση ή το Block1 σε μία αίτηση, περιγράφουν το μπλοκ που φέρει το payload του μηνύματος και κατά πόσο θα ακολουθήσουν και άλλα ή είναι το τελευταίο. Αν το M flag είναι 1 τότε το μέγεθος του payload πρέπει να είναι δύναμη του 2, όπως ακριβώς δηλώνει το SZX, διαφορετικά, αν πρόκειται για το τελευταίο μπλοκ, αυτό μπορεί να είναι και μικρότερου μεγέθους.

Το Block1 σε μία απάντηση δηλώνει πιο μπλοκ επιβεβαιώνεται (σε PUT, POST αιτήσεις) και αν το M flag είναι 0, τότε σημαίνει ότι επιβεβαιώθηκε και το τελευταίο μπλοκ και η μετάδοση ολοκληρώθηκε.

Ακολουθεί παράδειγμα με το Block2, με το option να είναι με την μορφή 2/x/y/z, όπου x είναι το NUM, y το M flag και z το υπολογισμένο μέγεθος του payload βάσει του SZX:



Εικόνα 4.6 CoAP Block2 παράδειγμα

4.8 Το CoAP Σήμερα

Η τελευταία έκδοση του CoAP είναι η 18 με τις σημαντικότερες αλλαγές να βρίσκονται στις εκδόσεις 09, 10, 12 και 13. Συνολικά το πρωτόκολλο δεν έχει αλλάξει και οι στόχοι του παραμένουν ίδιοι. Έχουν υπάρξει όμως σημαντικές αλλαγές στην σύνταξη του μηνύματος όπως φαίνεται και στην εικόνα 4.7.

	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Ver			T		TKL			Code								Message ID															
4	32	Token (εάν υπάρχει, TLK Bytes) …																															
…	…	Options (εάν υπάρχουν) …																															
…	…	1	1	1	1	1	1	1	1	Payload (εάν υπάρχει) …																							

Εικόνα 4.7 CoAP 18 Header

Όπως φαίνεται το Option Count αντικαταστάθηκε από το TKL που δηλώνει το μήκος του Token, το οποίο πλέον ακολουθεί αμέσως μετά την κεφαλίδα και δεν είναι Option. Τα Options που ακολουθούν το Token έχουν την ίδια γενικά μορφή, μόνο που υπάρχει πλέον υποστήριξη για πολύ μεγαλύτερα μεγέθη, έως 1034 Bytes. Με την απουσία του Option Count, ένας κόμβος αντιλαμβάνεται ότι διάβασε όλα τα Option, όταν στην θέση του επόμενου Option Delta συναντήσει το Payload Marker, που είναι 8 διαδοχικά bit με την τιμή 1. Ακολουθεί το payload.

Η χρήση των αντιπροσώπων και των κρυφών μνημών βελτιώθηκε σημαντικά εισάγοντας ένα ακόμα χαρακτηριστικό στα option, αν είναι ασφαλές να προωθηθεί το καθένα στον αρχικό διακομιστή (safe/unsafe), σε περίπτωση που ο αντιπρόσωπος δεν το αναγνωρίζει.

Οι αλλαγές αυτές κάνουν οποιαδήποτε προσπάθεια επικοινωνίας με παλιότερες εκδόσεις μη εφικτή και πέρα από την μορφή του μηνύματος που άλλαξε, έχουν αλλάξει και τα περισσότερα Option Number. Οι βασικές αλλαγές όμως μπορούν να γίνουν αρκετά εύκολα σε μία υπάρχουσα υλοποίηση, ώστε να επιτραπεί η επικοινωνία και η υποστήριξη του νέου προτύπου.

Επίσης, αυτό που έχει αλλάξει με τις εκδόσεις είναι και η σαφήνεια του πρωτοκόλλου που παρουσιάζεται. Έχουν αντιμετωπιστεί πολλά σημεία ασάφειας τόσο στο CoAP όσο και στις επεκτάσεις του, ενώ πλέον διάφορες ειδικές περιπτώσεις έχουν οριστεί αυστηρά. Από την έκδοση 14 και έπειτα οι αλλαγές που έχουν γίνει είναι μόνο τέτοιου χαρακτήρα και δεν επηρεάζουν την διαλειτουργικότητα των υλοποιήσεων, κάτι που δείχνει ότι ο σχεδιασμός του τείνει προς το τέλος.

5 Υλοποιήσεις του CoAP

5.1 Υλοποίηση στην Wiselib

5.1.1 Δομή

Η βασική υλοποίηση του CoAP ξεκίνησε στην Wiselib, γιατί όπως αναφέρθηκε, μέσω της Wiselib υπάρχει πρόσβαση σε μεγάλη γκάμα συσκευών, στις οποίες συμπεριλαμβάνονται και τα iSense που ήταν διαθέσιμα. Ο κώδικας διατίθεται ελεύθερα: https://github.com/ibr-alg/wiselib/tree/master/wiselib.testing/algorithms/protocols/coap

Συμβαδίζοντας με τον τρόπο συγγραφής της Wiselib, το CoAP υλοποιήθηκε σε μορφή template. Η δομή του CoAP χωρίστηκε σε τέσσερις κλάσεις:

- **Coap:** Αποτελεί το βασικό template που υλοποιεί τον πυρήνα του πρωτοκόλλου και αναλαμβάνει την αποστολή και λήψη των μηνυμάτων, την εξυπηρέτηση των αιτήσεων, την διατήρηση της λίστας των αναμεταδόσεων καθώς και των παρατηρητών, ενώ αποτελεί και την διεπαφή του CoAP προς τις εφαρμογές.
- **CoapPacket:** Εδώ υλοποιούνται όλες οι λειτουργίες που είναι απαραίτητες στο επίπεδο του πακέτου. Γίνεται η ανάλυση του πακέτου ώστε να ληφθούν τα πεδία της κεφαλίδας, τα option με τις τιμές τους, καθώς και το payload. Είναι υπεύθυνο επίσης και για την αντίστροφη λειτουργία, να λάβει δηλαδή ένα μήνυμα CoAP και να το μετατρέψει σε πακέτο έτοιμο για αποστολή.
- **ResourceController:** Το template που είναι υπεύθυνο για την διαχείριση των αντικειμένων ενός κόμβου. Μέσω αυτού δημιουργούνται νέα αντικείμενα στον κόμβο και ορίζεται η συνάρτηση που θα κληθεί (callback function) αν ζητηθεί αυτό το αντικείμενο. Η ύπαρξή του είναι σημαντική για να παρέχει την δυνατότητα στο CoAP να καλεί συναρτήσεις της εφαρμογής, οι οποίες είναι ανεξάρτητες από το πρωτόκολλο.
- **Queries:** Αν μία αίτηση έχει κάποιο Uri-Query, αυτό αποθηκεύεται και η εφαρμογή αναλαμβάνει πως και αν θα το χειριστεί, μέσω αυτής της κλάσης.

Στις συγκεκριμένες συσκευές της εργασίας δεν υπάρχει τρόπος για εύκολη διαδικασία αποσφαλμάτωσης και ο μόνος τρόπος είναι αποστολή μηνυμάτων μέσω της σειριακής θύρας. Αυτό εισάγει μεγάλο overhead στον κώδικα και για τον λόγο αυτό έχει δοθεί η δυνατότητα να ενεργοποιείται τμηματικά ανάλογα το επίπεδο της πληροφόρησης που θέλει να έχει ο προγραμματιστής.

5.1.2 Περιορισμοί

Το CoAP έχει σχεδιαστεί για συσκευές με περιορισμένες δυνατότητες, στην περίπτωση βέβαια των iSense, οι περιορισμοί είναι ακόμα μεγαλύτεροι και έπρεπε να γίνουν κάποιες σχεδιαστικές επιλογές ώστε να μπορέσει ο πυρήνας και τα βασικά χαρακτηριστικά να τρέξουν στις συσκευές, θυσιάζοντας κάποια λιγότερο απαραίτητα. Τα iSense χρησιμοποιώντας κοινή μνήμη για τον κώδικα και τα δεδομένα του, ανάγκασε περικοπές στον κώδικα ώστε να υπάρχει αρκετός χώρος για την εκτέλεση του CoAP και των εφαρμογών του, για τις οποίες δεν είναι γνωστό εκ των προτέρων πόση μνήμη απαιτούν, οπότε έπρεπε να δοθεί ένας σημαντικός χώρος ώστε να υπάρχει περιθώριο και για περίπλοκες εφαρμογές.

Σε επίπεδο υλοποίησης το εξαιρούμενα χαρακτηριστικά είναι τα options που δεν κρίθηκαν σημαντικά για τις ανάγκες της υλοποίησης, τα οποία αναγνωρίζονται αν περιέχονται στο μήνυμα αλλά δεν μπορούν να επεξεργαστούν. Αυτά τα option είναι τα: Proxy-Uri, ETag, Location-Path, Location-Query, Accept, If-Match, If-None-Match, Block1. Η επιλογή των συγκεκριμένων έγινε γιατί δεν επηρεάζουν την βασική λειτουργία του CoAP. Αυτό έχει σαν αποτέλεσμα οι κόμβοι να μην μπορούν να δράσουν ως αντιπρόσωποι, να μην είναι διαθέσιμος ο μηχανισμός επικύρωσης, να μην μπορεί η συσκευή να δημιουργήσει αντικείμενο σε διαφορετική διεύθυνση από αυτή που ζητήθηκε, να αγνοεί τις προτιμήσεις του πελάτη για τον τύπο της απάντησης και να μην μπορεί να κάνει υπό συνθήκη αιτήσεις. Επίσης, δεν έχει γίνει χρήση κάποιου προφίλ ασφαλείας.

Για τις εφαρμογές δίνεται η δυνατότητα να οριστούν κάποια όρια για το CoAP, που ανάλογα τις ανάγκες, θα αφήσουν περισσότερη μνήμη ελεύθερη ή θα δώσουν την δυνατότητα για περισσότερους πόρους στο CoAP. Τα όρια αυτά είναι:

- Ο μέγιστος αριθμός αντικειμένων που μπορούν να υπάρχουν σε έναν κόμβο.
- Ο μέγιστος αριθμός Queries που μπορεί να υποστηρίζει κάθε αντικείμενο.
- Ο μέγιστος αριθμός παρατηρητών που μπορούν να εγγραφούν στο σύστημα.
- Το μέγιστο μέγεθος του εσωτερικού καταχωρητή που αποθηκεύεται ένα μήνυμα προς αποστολή, πριν τον κατακερματισμό.

5.1.3 API

Η κλάση **wiselib::CoapPacket** ορίζει ένα αντικείμενο που λειτουργεί ως CoAP μήνυμα και ορίζεται ως εξής:

```
namespace wiselib {
#ifdef DEBUG_OPTION

   template<typename Debug_P>
#endif
   class CoapPacket
```

Αν ενεργοποιηθεί η δυνατότητα για debug κατά την δήλωση του template απαιτείται σαν όρισμα το debug interface που έχει δηλωθεί στην εφαρμογή. Υπάρχει πρόσβαση σε μεθόδους μέσω των οποίων μπορεί να το χειριστεί μία εφαρμογή:

- init: αρχικοποιεί τις εσωτερικές μεταβλητές του μηνύματος.
- getters και setters: για όλα τα υποστηριζόμενα option της υλοποίησης καθώς και για τα πεδία της κεφαλίδας, υπάρχουν οι ανάλογες συναρτήσεις για να επιστρέφουν την τιμή τους ή για να θέτουν νέα.
- buffer_to_packet: Παίρνει ως είσοδο το πακέτο, όπως το επέστρεψε το επίπεδο του δικτύου, μαζί με το μέγεθός του και το αποκωδικοποιεί, θέτοντας τις εσωτερικές μεταβλητές στις ανάλογες τιμές που υπάρχουν στο πακέτο. Τυχόν σφάλμα στην ανάγνωση του μηνύματος (πχ εσφαλμένο option) θα εμφανιστεί εδώ.
- packet_to_buffer: Κάνει την ακριβώς αντίθετη διαδικασία και δημιουργεί από το αντικείμενο το ανάλογο πακέτο που θα αποσταλεί.

Οι υπόλοιπες συναρτήσεις είναι εσωτερικές (protected) και μπορούν να κληθούν μόνο από το ίδιο το αντικείμενο. Βοηθούν στην ανάλυση των option καθώς και στην δημιουργία τους.

Η κλάση **wiselib::ResourceController** είναι ο ελεγκτής των αντικειμένων που θα δημιουργούν οι εφαρμογές του CoAP. Κάθε ResourceController είναι και ένα CoAP αντικείμενο. Η βασική λειτουργία του βασίζεται στα delegates που δίνουν την δυνατότητα να αποθηκεύεται εσωτερικά στην δομή του ελεγκτή, δείκτης προς την συνάρτηση που θα κληθεί αν ζητηθεί αυτό το αντικείμενο (callback function), κάτι που δημιουργείται κατά την εκτέλεση του προγράμματος. Ο ελεγκτής δηλώνει μία δομή με μεταβλητές που θα περνάει ως όρισμα στις συναρτήσεις των εφαρμογών όπως φαίνεται στον κώδικα 5.1.

```
struct callback_arg {
   uint8_t method;
   uint8_t* input_data;
   size_t input_data_len;
   uint8_t* output_data;
   uint16_t* output_data_len;
#ifdef ENABLE_URI_QUERIES
   queries_t* uri_queries;
#endif
};
typedef callback_arg callback_arg_t;
```

Κώδικας 5.1

Κάθε συνάρτηση που επιθυμεί να εγγραφεί ως callback function στον ελεγκτή πρέπει να δέχεται ως μοναδικό όρισμα έναν δείκτη αυτής της δομής. Με method δηλώνεται η μέθοδος αίτησης, input_data είναι το payload της αίτησης και output_data το payload της απάντησης. uri_queries είναι δείκτης προς τα Queries της αίτησης. Επίσης, η συνάρτηση πρέπει να επιστρέφει coap_status_t τύπο δεδομένων, όπως ορίζεται στο CoapPacket, που είναι ουσιαστικά η μέθοδος απάντησης.

Οι μέθοδοι που υποστηρίζει το αντικείμενο:

- Constructor: Υπάρχει ένας κενός constructor αλλά και ένας με ορίσματα, ώστε κατά την δημιουργία του αντικειμένου να αρχικοποιηθούν κάποιες τιμές, όπως το όνομα (δηλαδή το URI), οι μέθοδοι αίτησης που υποστηρίζει, αν υποστηρίζει piggybacked απαντήσεις, τον χρόνο σε δευτερόλεπτα, στον οποίο το αντικείμενο ενδέχεται να έχει νέα τιμή καθώς και τον τύπο δεδομένων που επιστρέφει.
- reg_callback: Μέσω αυτής της συνάρτησης δηλώνεται η callback function για το αντικείμενο.
- execute: Καλεί την callback function.
- getters και setters: Για τις εσωτερικές μεταβλητές του αντικειμένου.

Η κλάση **wiselib::Queries** αναλαμβάνει να αποθηκεύσει τα Uri-Queries που περιέχονται σε μία αίτηση. Η εφαρμογή στην συνέχεια αναλαμβάνει να τα χρησιμοποιήσει και μέσω αυτού του αντικειμένου μπορεί να λαμβάνει την τιμή ενός συγκεκριμένου Uri-Query. Η μέθοδος που απασχολεί τις εφαρμογές είναι η `char* value_of( String name )`. Καλείται στην συνάρτηση που έχει οριστεί ως callback function για ένα αντικείμενο με όρισμα το όνομα του Query που επιθυμεί να λάβει την τιμή του. Για παράδειγμα, αν η αίτηση ήταν "`coap://host/resource?act=true`", τότε η κλήση `value_of( "act");` θα επέστρεφε "true".

Η κύρια κλάση του CoAP είναι η **wiselib::Coap**, μέσω της οποίας γίνονται όλες οι ενέργειες. Μία εφαρμογή καλείται να δημιουργήσει ένα Coap αντικείμενο και να αλληλεπιδρά αποκλειστικά μέσω αυτού και των συναρτήσεών του (με εξαίρεση τα Queries). Τα υπόλοιπα αντικείμενα είναι διαθέσιμα αλλά η εφαρμογή δεν είναι απαραίτητο να τα χρησιμοποιεί, αφού η χρήση τους γίνεται καλύτερα μέσα από το ίδιο το Coap. Απαιτεί μια σειρά ορισμάτων για την δημιουργία του, βάσει των προτύπων της Wiselib:

```
template<typename OsModel_P, typename Radio_P, typename Timer_P, typename Debug_P,
typename Clock_P, typename Rand_P, typename String_P>
```

Το σημαντικότερο είναι ότι δέχεται ως όρισμα το Radio interface της εφαρμογής, δηλαδή το template που ορίζει πως γίνεται η επικοινωνία με τις υπόλοιπες συσκευές, που μπορεί να είναι είτε κάποιο απλό πρωτόκολλο χωρίς δρομολόγηση (1 hop), είτε κάποιο πρωτόκολλο με δρομολόγηση (multi hop). Η επιλογή είναι στην ευχέρεια της εφαρμογής και η υλοποίηση του CoAP είναι ανεξάρτητη αυτής.

Οι μέθοδοι που παρέχει και είναι κρίσιμες για τις εφαρμογές είναι οι εξής:

- init: Για την ενεργοποίηση του διακομιστή και των εσωτερικών ρολογιών του, απαραίτητα για την παροχή ειδοποιήσεων και για το σύστημα επαναμεταδόσεων.
- add_resource, update_resource, delete_resource: Συναρτήσεις για την δημιουργία, ανανέωση ή διαγραφή αντικειμένων αντίστοιχα. Είναι ουσιαστικά η διεπαφή προς τον ResourceController.
- coap_send: Για την αποστολή ενός CoapPacket.
- receiver: Η συνάρτηση που αναλαμβάνει την επεξεργασία ενός εισερχόμενου CoAP μηνύματος. Πρόκειται για την κύρια συνάρτηση του αντικειμένου που πρέπει να καλείται από την εφαρμογή όταν καταφτάνει ένα καινούριο CoAP μήνυμα. Αναλαμβάνει την αποστολή επιβεβαιώσεων και την ανάλυση και εξυπηρέτηση των αιτήσεων.
- coap_notify_from_interrupt: Καλείται από την εφαρμογή για να δηλώσει ότι ένα συγκεκριμένο URI ανανεώθηκε με νέα τιμή, οπότε πρέπει να γίνει ενημέρωση των παρατηρητών.

5.1.4 Παράδειγμα εφαρμογής

Παρουσιάζεται ένα παράδειγμα εφαρμογής σε Wiselib που κάνει χρήση της συγκεκριμένης υλοποίησης και τρέχει σε iSense Core Module. Έχει ένα αντικείμενο που αλληλεπιδρά με το led της συσκευής.

```
#include "external_interface/external_interface.h"
#include "util/delegates/delegate.hpp"
#include "util/pstl/map_static_vector.h"
#include "util/pstl/static_string.h"
#include <isense/modules/core_module/core_module.h>
#include "algorithms/protocols/coap/coap.h"
typedef wiselib::Coap<Os, Os::Radio, Os::Timer, Os::Debug, Os::Clock, Os::Rand,
wiselib::StaticString> coap_t;
```

Κώδικας 5.3 Απαραίτητες δηλώσεις

```
class iSenseCoapCollectorApp
{
public:

    void init(Os::AppMainParameter& value) {
        ospointer = &value;
        radio_ = &wiselib::FacetProvider<Os, Os::TxRadio>::get_facet(value);
        timer_ = &wiselib::FacetProvider<Os, Os::Timer>::get_facet(value);
        debug_ = &wiselib::FacetProvider<Os, Os::Debug>::get_facet(value);
        rand_ = &wiselib::FacetProvider<Os, Os::Rand>::get_facet(value);
        clock_ = &wiselib::FacetProvider<Os, Os::Clock>::get_facet(value);

        radio_->set_channel(12);
#ifdef CORE_COLLECTOR
        cm_ = new isense::CoreModule(value);
        led_status_ = 0;
        cm_->led_off();
#endif
        radio_->enable_radio();
        // coap init
        rand_->srand(radio_->id());
        mid_ = (uint16_t) rand_->operator()(65536 / 2);
        debug_->debug("iSense CoAP Collector App %x", radio_->id());
        add_resources();

        coap_.init(*radio_, *timer_, *debug_, *clock_, mid_);
                                radio_->reg_recv_callback<iSenseCoapCollectorApp,
&iSenseCoapCollectorApp::receive_radio_message > (this);
    }
```

Κώδικας 5.2 Η init function που καλείται κατά την εκίνηση

```
void receive_radio_message(Os::Radio::node_id_t from, Os::Radio::size_t len,
Os::Radio::block_data_t *buf) {

        if (buf[0] == WISELIB_MID_COAP) {
            debug_->debug("received from %x [%d]", from, buf[0]);
            coap_.receiver(len, buf, from);
        }
    }
```

Κώδικας 5.4 Εδώ λαμβάνονται τα μηνύματα του δικτύου

```
resource_t core_resource("led", GET | POST, true, 600, TEXT_PLAIN);
core_resource.reg_callback<iSenseCoapCollectorApp, &iSenseCoapCollectorApp::led >
(this);
coap_.add_resource(&core_resource);
```

Κώδικας 5.5 Η δημιουργία ενός CoAP αντικειμένου με το όνομα "led", που δέχεται GET και POST αιτήσεις, υποστηρίζει piggy-backed, κάθε 600 δευτερόλεπτα θα στέλνει ειδοποίηση και ο τύπος του Content-Type είναι TEXT_PLAIN. Ορίζει την callback function και το δηλώνει στο CoAP

```
coap_status_t led(callback_arg_t* args) {
        if (args->method == COAP_GET) {
            *(args->output_data_len) = sprintf((char*) args->output_data, "%d", led_status_);
            return CONTENT;
        } else if (args->method == COAP_POST) {
            if (*(args->input_data) == 0x30) {
                led_status_ = 0;
                cm_->led_off();
              *(args->output_data_len) = sprintf((char*) args->output_data, "%d", led_status_);
                return CHANGED;
            } else if (*(args->input_data) == 0x31) {
                led_status_ = 1;
              *(args->output_data_len) = sprintf((char*) args->output_data, "%d", led_status_);
                cm_->led_on();
                return CHANGED;
            }
            return NOT_IMPLEMENTED;
        }
        return INTERNAL_SERVER_ERROR;
    }
```

Κώδικας 5.6 Η callback function

```
private:
    Os::TxRadio::self_pointer_t radio_;
    Os::Timer::self_pointer_t timer_;
    Os::Debug::self_pointer_t debug_;
    Os::Clock::self_pointer_t clock_;
    Os::Rand::self_pointer_t rand_;

    coap_t coap_;
    uint16_t mid_;
    bool pir_sensor_;
    Os::AppMainParameter* ospointer;

    isense::CoreModule* cm_;
    uint8_t led_status_;
};
// ------------------------------------------------------------------------
wiselib::WiselibApplication<Os, iSenseCoapCollectorApp> coap_app;
// ------------------------------------------------------------------------

void application_main(Os::AppMainParameter& value) {
    coap_app.init(value);
}
```

Κώδικας 5.7 Οι απαραίτητες private μεταβλητές και η main

5.2 Υλοποίηση σε Arduino

Κατά την εκπόνηση της εργασίας, η υποστήριξη για Arduino από την Wiselib ήταν ακόμα σε πειραματικό στάδιο, για αυτό η υλοποίηση σε Arduino έγινε απ' ευθείας στην πλατφόρμα, παίρνοντας τα περισσότερα χαρακτηριστικά από αυτήν της Wiselib. Βασικός στόχος ήταν οι δύο υλοποιήσεις να είναι όμοιες στην λειτουργία τους και να υπάρχει ομαλή αλληλεπίδραση μεταξύ των διαφορετικών οικογενειών συσκευών. Η υλοποίηση διατίθεται ελεύθερα: https://github.com/dgiannakop/Arduino-CoAP

Η δομή και τα χαρακτηριστικά της υλοποίησης είναι παρόμοια με αυτή της Wiselib, με τα βασικά αντικείμενα να είναι ίδια, με κάποιες μικρές διαφορές στην υλοποίηση, αλλά με την ίδια λογική. Εξαίρεση αποτελεί η απουσία υποστήριξης για Uri-Queries στο Arduino.

Βασική διαφορά για τις εφαρμογές εδώ είναι ότι τα CoAP αντικείμενα δεν υλοποιούνται με την μορφή συναρτήσεων, αλλά με κλάσεις. Αυτό ήταν απαραίτητο διότι για να μπορέσει να δημιουργηθεί δείκτης προς συνάρτηση και να λειτουργήσει σαν callback function έπρεπε να ανήκει σε μία κλάση, διαφορετικά αν ήταν εντός του βασικού sketch (όπως αποκαλούνται τα Arduino

projects) δεν γινόταν δεκτό κατά την μεταγλώττιση. Ακολουθεί ένα παράδειγμα απλής CoAP εφαρμογής:

```
//Include XBEE Libraries
#include <XBee.h>
#include <XbeeRadio.h>

//Include CoAP Libraries
#include <coap.h>
#include "myGETSensor.h"
#include "myPOSTSensor.h"

//Create the XbeeRadio object we'll be using
XBeeRadio xbee = XBeeRadio();
//Create a reusable response object for responses we expect to handle
XBeeRadioResponse response = XBeeRadioResponse();
//Create a reusable rx16 response object to get the address
Rx16Response rx = Rx16Response();

//CoAP object
Coap coap;

myGETSensor aSensor = myGETSensor("resGET1" , A0);
myPOSTSensor bSensor = myPOSTSensor("resGET-POST" , 3);

//Runs only once
void setup()
{

  pinMode(3, OUTPUT);
  // comment out for debuging
  xbee.initialize_xbee_module();
  //start our XbeeRadio object and set our baudrate to 38400.
  xbee.begin( 38400 );
  //Initialize our XBee module with the correct values (using the default channel,
channel 12)
  xbee.init(12);
  // init coap service
  coap.init( &xbee, &response, &rx );

  //add the resourse resGET
  coap.add_resource(&aSensor);
  coap.add_resource(&bSensor);
  //add the resources resGET-POST
}
void loop()
{
  //run the handler on each loop to respond to incoming requests
  coap.handler();
}
```

Κώδικας 5.8 Παράδειγμα εφαρμογής σε Arduino, το Sketch

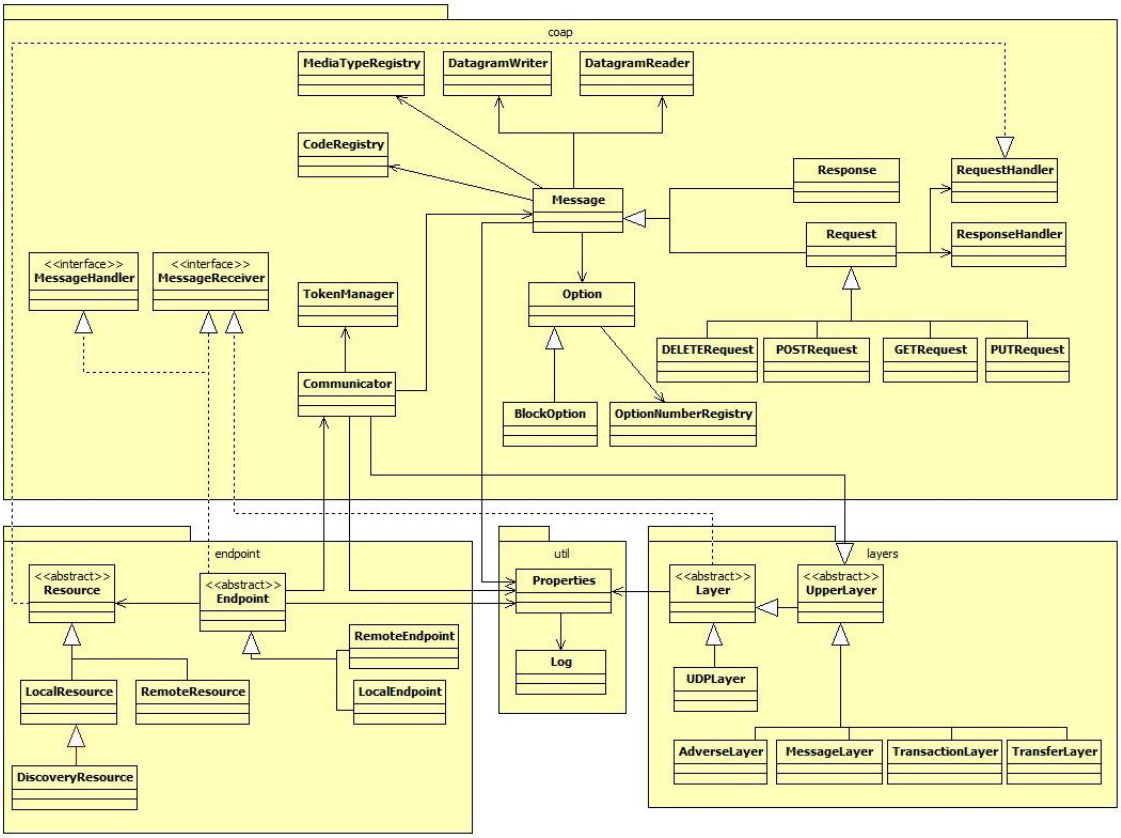
```
#include <coapSensor.h>

class myPOSTSensor :
public CoapSensor
{
public:
  int pin, status;
  myPOSTSensor(String name, int pin):
  CoapSensor(name)
  {
    this->pin = pin;
    this->status = 0;
    pinMode(pin, OUTPUT);
    digitalWrite(pin, LOW);
  }
  void get_value( uint8_t* output_data, size_t* output_data_len)
  {
    this->status = digitalRead(this->pin);
    *output_data_len = sprintf( (char*)output_data, "%d", this->status );
  }
  void set_value(uint8_t* input_data, size_t input_data_len, uint8_t* output_data,
size_t* output_data_len)
  {
    this->set(*input_data-0x30);
    output_data[0] = 0x30 + status;
    *output_data_len = 1;
  }
  void set(uint8_t value)
  {
    this->status = value;
    digitalWrite(pin, status);
  }
};
```

Κώδικας 5.9 Παράδειγμα εφαρμογής σε Arduino, η κλάση με τις callback functions

5.3 Californium

Το Californium [24] είναι σχεδιασμένο από τον Matthias Kovatsch και είναι μια υλοποίηση του CoAP σε Java. Παρέχει όλα τα χαρακτηριστικά του CoAP και των επεκτάσεών του. Είναι σχεδιασμένο για κανονικά συστήματα γενικής χρήσης και όχι για συστήματα περιορισμένων δυνατοτήτων. Η δομή του χωρίζεται σε επίπεδα (εικόνα 5,1) απομονώνοντας τις υλοποιήσεις των διαφορετικών μερών μεταξύ τους, που επιτρέπει στην εφαρμογή που θα το χρησιμοποιήσει να κάνει χρήση των ενδιάμεσων κλάσεων και αντικειμένων. Μπορεί να χρησιμοποιηθεί σε εφαρμογές πελάτη και διακομιστή. Επίσης, παρέχει υποστήριξη για παλαιότερες εκδόσεις του CoAP αν αυτό κρίνεται απαραίτητο. Έχει αποδειχθεί ότι υλοποιεί σωστά το

πρωτόκολλο καθώς έχει περάσει επιτυχώς όλες τις δοκιμές που έγιναν στα πλαίσια του "ETSI IoT CoAP Plugtest", μία από τις εκδηλώσεις που συμβαίνουν στα πλαίσια της ανάπτυξης του CoAP για να επιβεβαιωθεί η διαλειτουργικότητα των διαφόρων υλοποιήσεων και η συμμόρφωσή τους με το πρωτόκολλο. Υποστηρίζεται ενεργά από τον δημιουργό και εξελίσσεται συνεχώς ακολουθώντας τις νέες εκδόσεις του πρωτοκόλλου.



Εικόνα 5.1 Californium Classes

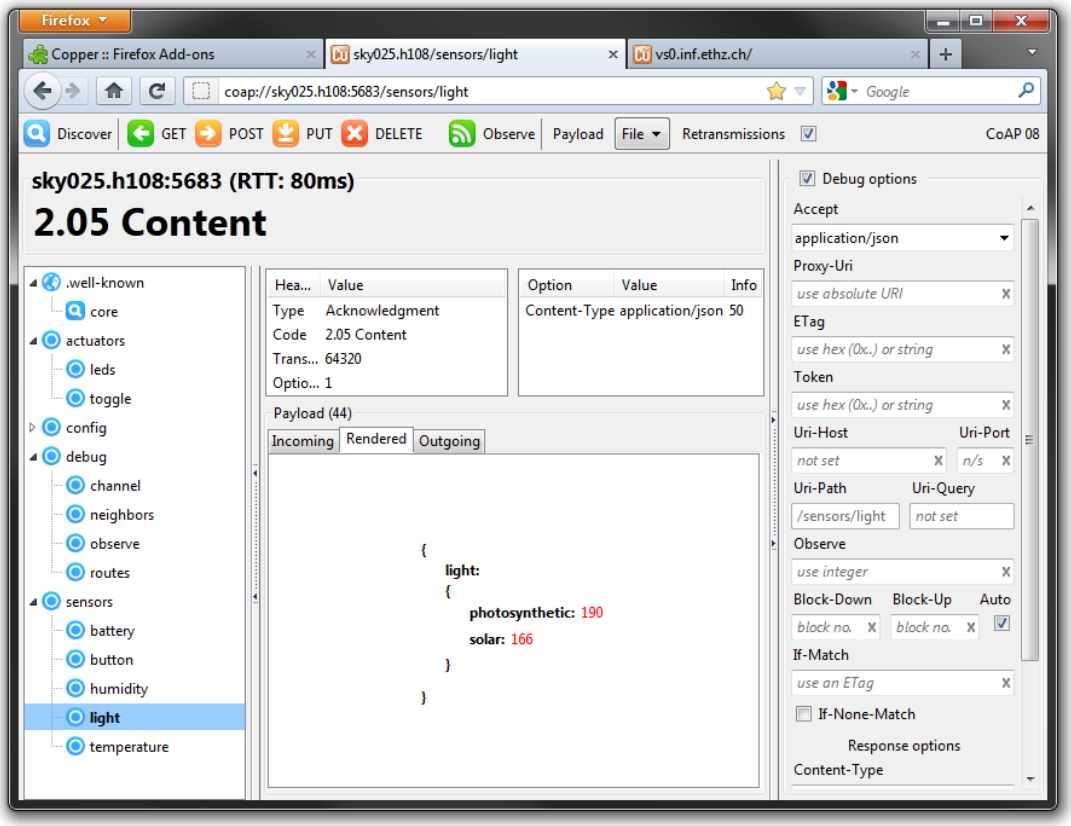
5.4Copper

Το Copper [25] είναι ένα plug-in για τον Firefox από τον Matthias Kovatsch. Δίνει την δυνατότητα στον browser να επικοινωνήσει με CoAP κόμβους. Παρέχει ουσιαστικά μια υλοποίηση του CoAP σε Javascript καθώς και ένα interface ώστε να μπορεί ο χρήστης να αλληλεπιδρά με τους κόμβους απευθείας μέσω CoAP μηνυμάτων και όχι μέσω κάποιου HTTP mapping. Μπορεί κανείς να δει αναλυτικά όλα τα περιεχόμενα του μηνύματος καθώς και να δημιουργήσει μηνύματα με τα ακριβή χαρακτηριστικά που επιθυμεί.

Υποστηρίζει εξ ολοκλήρου το CoAP, το resource discovery πρότυπο, το Observe πρότυπο, καθώς και μεταδόσεις κατά μπλοκ. Παρέχει πλούσιες δυνατότητες για debugging με την χρήση της κονσόλας του browser. Όσο το CoAP εξελίσσεται με νέες εκδόσεις του πρωτοκόλλου, το Copper δίνει την δυνατότητα να επιλεγεί η

έκδοση που θα χρησιμοποιηθεί κάθε στιγμή και υποστηρίζει τις εκδόσεις 03, 06, 08, 12, 13 και 18. Τέλος, παρουσιάζει και το round trip time (RTT) μιας αίτησης-απάντησης.

Η εμφάνιση του Copper φαίνεται στην εικόνα 5.2 με τις διαθέσιμες ενέργειες και τον τρόπο παρουσίασης των πληροφοριών.

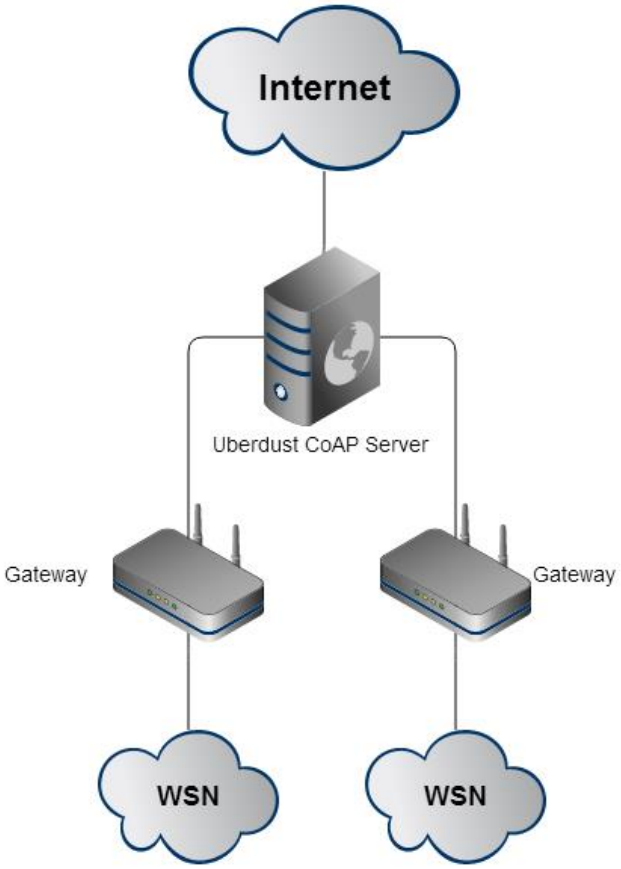


Εικόνα 5.2 Copper plug-in on Firefox

6 Σύστημα αυτοματοποίησης

6.1 Αρχιτεκτονική

Η ύπαρξη των ασύρματων δικτύων αισθητήρων εξοπλισμένων με CoAP ανοίγει τον δρόμο στην δημιουργία ενός μεγαλύτερου συστήματος, που θα εκμεταλλεύεται στο μέγιστο τις δυνατότητες που παρέχονται από τις συσκευές και θα αυτοματοποιεί πολλές διαδικασίες που διαφορετικά θα ήθελαν συνεχή παρέμβαση και παρακολούθηση από τον τελικό πελάτη, είτε αυτός ήταν ο διαχειριστής είτε ένας απλός χρήστης. Σχεδιάστηκε ένα σύστημα που θα απλοποιήσει την διαδικασία διαχείρισης και χρήσης του δικτύου αισθητήρων και θα παρέχει μια σειρά υπηρεσιών. Η αρχιτεκτονική αυτού του συστήματος χωρίζεται σε τρία επίπεδα και παρουσιάζεται στην εικόνα 6.1.



Εικόνα 6.1 Η αρχιτεκτονική του συστήματος

Τα επίπεδα του συστήματος:

- Στο **1ο επίπεδο** είναι τα ετερογενή ασύρματα δίκτυα αισθητήρων (WSN), όπου όλες οι συσκευές τρέχουν μια εφαρμογή βασισμένη στο CoAP. Αυτά σχηματίζονται από συσκευές iSense και Arduino με τοπολογία αστέρα, όπου κεντρικός κόμβος είναι ειδικές συσκευές που ανήκουν στο 2ο επίπεδο. Οι συσκευές είναι εξοπλισμένες με διάφορους αισθητήρες και διακόπτες και τρέχουν μία εφαρμογή βασισμένη στο CoAP. Τα iSense είναι εξοπλισμένα με το Environmental Sensor Module, άρα έχουν αισθητήρες θερμότητας και φωτεινότητας. Μερικά έχουν επίσης το Security Module, άρα και αισθητήρα κίνησης. Τα Arduino είναι συνδεδεμένα με την ειδική πλακέτα που παρουσιάστηκε, άρα έχουν αισθητήρα κίνησης και θερμοκρασίας, ενώ ελέγχουν έως και πέντε διακόπτες ρεύματος.

  Κάθε αισθητήρας και διακόπτης είναι και ένα αντικείμενο προσβάσιμο μέσω του CoAP. Όλα τα αντικείμενα υποστηρίζουν την μέθοδο GET, ενώ οι διακόπτες και την POST ώστε να στέλνονται εντολές προς αυτούς για το άνοιγμα ή το κλείσιμό τους. Παρέχεται επίσης η δυνατότητα δημιουργίας νέων αντικειμένων που θα περιέχουν μία απλή λέξη ή πρόταση μέσω CoAP αιτήσεων κατά την λειτουργία της εφαρμογής. Επίσης υποστηρίζεται το Observe πρότυπο, όπου η δημιουργία ειδοποιήσεων προκύπτει είτε απευθείας από τον αισθητήρα προκαλώντας διακοπή (interrupt) στην εφαρμογή (πχ ο αισθητήρας εντόπισε κίνηση), είτε μετά το πέρας ενός χρονικού διαστήματος που η εφαρμογή κρίνει ότι πρέπει να ενημερωθεί ο παρατηρητής. Παρέχεται το αντικείμενο "`/.well-known/core`" που είναι υπεύθυνο για την δήλωση των διαθέσιμων αντικειμένων της συσκευής. Τέλος, υποστηρίζεται η μετάδοση κατά μπλοκ για τις απαντήσεις.

- Στο **2ο επίπεδο** είναι οι πύλες (gateways), όπου είναι τα κέντρα της τοπολογίας αστέρα. Το δίκτυο αισθητήρων λειτουργεί στο 802.15.4 όπως έχει ήδη αναφερθεί και οι πύλες επιτρέπουν την σύνδεση αυτού του δικτύου με τα κοινά 802.3 (Ethernet) και 802.11 (WiFi). Γεφυρώνουν ουσιαστικά τα διαφορετικά δίκτυα, επιτρέποντας την μετάδοση μηνυμάτων από τον υπολογιστή προς τις συσκευές του 1ου επιπέδου και αντίστροφα. Πρόκειται για iSense συσκευές με το Gateway Module, συνδεδεμένες με έναν υπολογιστή που τρέχει το TestbedRuntime. Μέσω του TestbedRuntime προωθούνται μηνύματα προς τις συσκευές και διαβάζονται οι απαντήσεις τους. Τα μηνύματα που φτάνουν στο iSense προς αποστολή, περιέχουν την διεύθυνση της συσκευής που πρέπει να σταλούν, ενώ όταν λαμβάνουν μηνύματα από το δίκτυο, δηλώνουν στο TestbedRuntime την προέλευσή τους. Αυτό

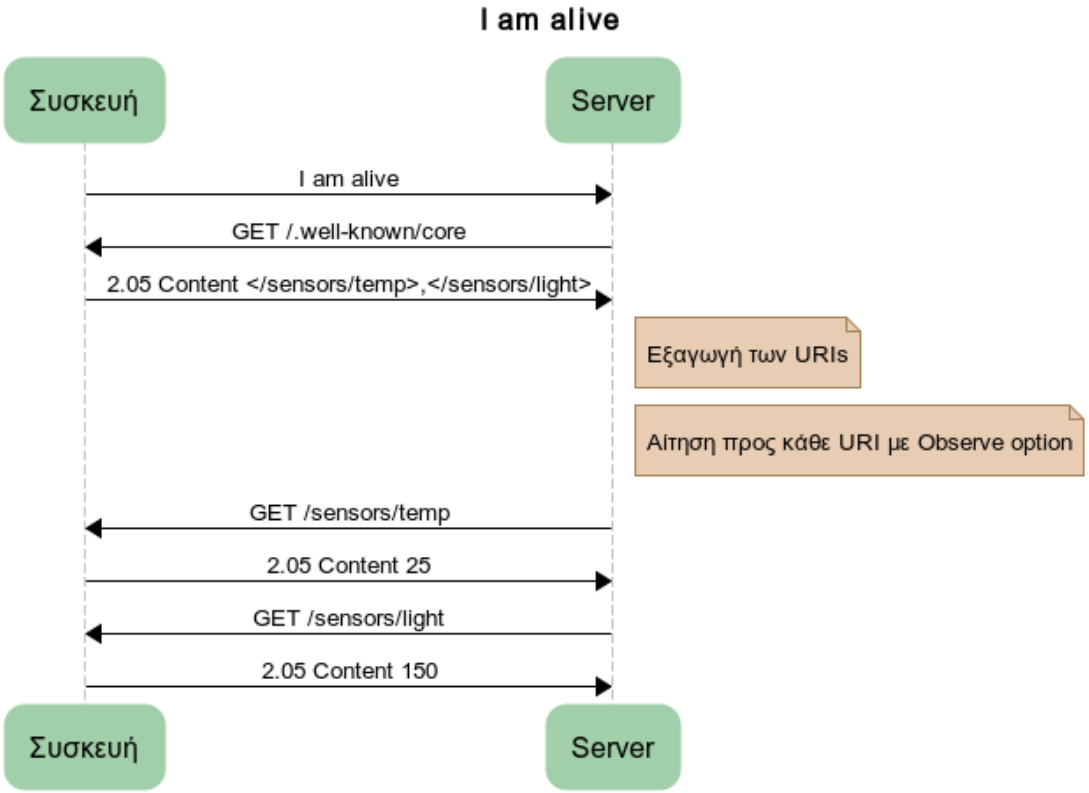
γίνεται προσθέτοντας 2 Byte πριν το CoAP μήνυμα που δηλώνει το αναγνωριστικό της συσκευής. Η πύλη αναλαμβάνει να τα αφαιρέσει και να στείλει το μήνυμα στην αντίστοιχη συσκευή, ενώ τα προσθέτει όταν λαμβάνει ένα μήνυμα από μία συσκευή.

- Στο **3ο επίπεδο** είναι το λογισμικό διαχείρισης, ελέγχου και αυτοματοποίησης και λειτουργεί ως διακομιστής για την εξυπηρέτηση πελατών. Είναι βασισμένο στο Überdust και έχει προστεθεί η υποστήριξη για CoAP με το Californium, ώστε να μπορεί να επεξεργάζεται CoAP μηνύματα. Είναι υπεύθυνο να αντιλαμβάνεται αλλαγές στο δίκτυο αισθητήρων, να συλλέγει τις πληροφορίες τους, να κρατάει ιστορικό των τιμών τους και να τις αποθηκεύει σε κρυφή μνήμη. Παρέχει διάφορα χαρακτηριστικά και υπηρεσίες στους χρήστες του συστήματος όπως παρουσιάζονται στην επόμενη ενότητα.

6.2Χαρακτηριστικά και υπηρεσίες

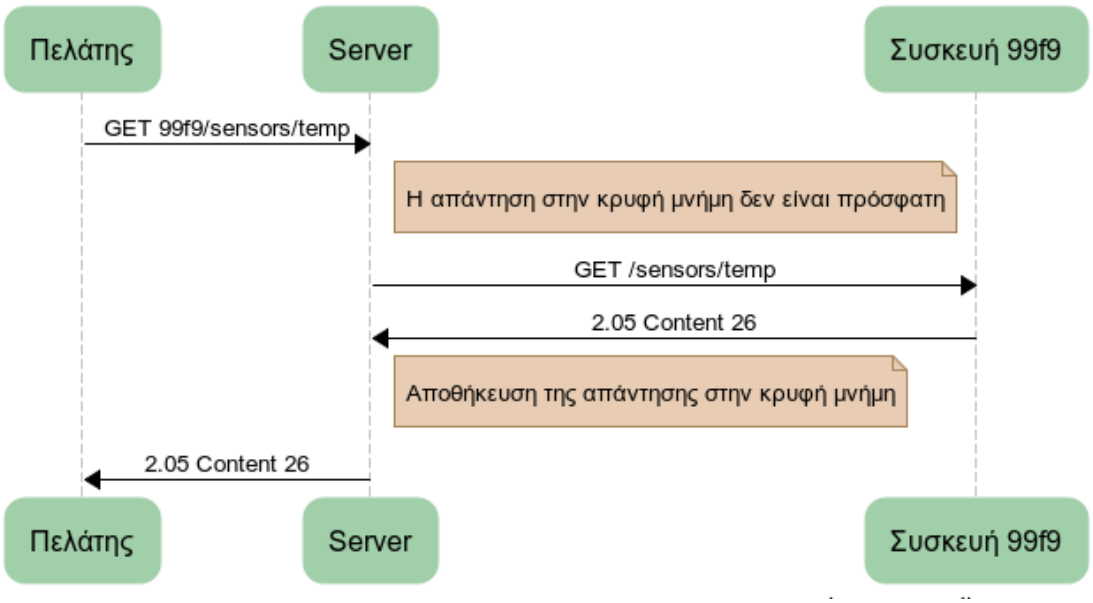
Η βάση του συστήματος είναι το Überdust και όλα τα χαρακτηριστικά και οι υπηρεσίες που παρέχονται από αυτό έχουν διατηρηθεί. Βασική διαφορά αποτελεί ο τρόπος επικοινωνίας με το ασύρματο δίκτυο. Οι συσκευές δεν βασίζονται στο περιοδικό broadcast των τιμών τους στο δίκτυο. Αντίθετα η πρόσβαση στις τιμές γίνεται μέσω CoAP αιτήσεων που στέλνονται μέσω των πυλών, ενώ η επικοινωνία γίνεται με unicast μηνύματα. Επίσης, δίνεται η δυνατότητα χρήσης οποιασδήποτε συσκευής που δρα ως CoAP διακομιστής να συνδεθεί στο σύστημα και να παρέχει τις τιμές του σε αυτό, χωρίς κάποια ειδική εφαρμογή.

Ένα ασύρματο δίκτυο μπορεί να μεταβάλλεται συνεχώς. Συσκευές να μετακινούνται με αποτέλεσμα να αλλάζουν πύλη, νέες συσκευές να εισέρχονται στο δίκτυο και υπάρχουσες να αφαιρούνται. Αυτά είναι όλα γεγονότα που πρέπει να έχουν προβλεφθεί από ένα σύστημα αυτοματοποίησης, ώστε να μειώνεται η ανάγκη ανθρώπινης παρεμβολής για κάθε μία από αυτές τις αλλαγές. Το πρόβλημα αυτό λύνεται με την σχεδίαση ενός τρόπου αυτόματης ρύθμισης (auto configuration) των συσκευών. Για τον σκοπό αυτό οι συσκευές έχουν την δυνατότητα να στέλνουν περιοδικά ένα ειδικό μήνυμα ώστε να δηλώνουν την παρουσία τους στο δίκτυο, το οποίο ονομάζεται “I am alive”. Το μήνυμα αυτό προωθείται μέσω μιας πύλης στο κεντρικό σύστημα. Εκεί αν βρεθεί κάποια αλλαγή ή η συσκευή είναι νέα, στέλνεται μία αίτηση προς το αντικείμενο “/.well-known/core” της συσκευής. Λαμβάνοντας στην συνέχεια τα διαθέσιμα αντικείμενα της συσκευής, στέλνονται αιτήσεις προς το κάθε αντικείμενο, οι οποίες δηλώνουν παράλληλα ενδιαφέρον για ενημερώσεις. Με τον τρόπο αυτό, το κεντρικό σύστημα θα έχει πάντα τις πρόσφατες τιμές από τις συσκευές.

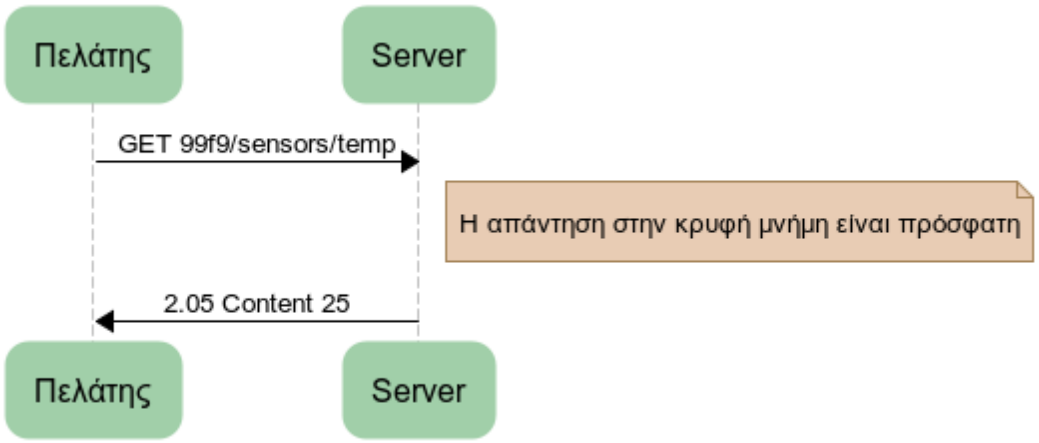


Εικόνα 6.2 Η λογική του "I am alive"

Με την λογική αυτή, ότι στο κεντρικό σύστημα είναι πάντα πρόσφατες τιμές, τοποθετήθηκε μία κρυφή μνήμη. Σε αυτήν αποθηκεύονται όλες οι πρόσφατες απαντήσεις των συσκευών. Όταν ένας χρήστης στέλνει μία αίτηση προς ένα αντικείμενο μιας συσκευής, τότε ακολουθώντας τους κανόνες του CoAP, αν η απάντηση που είναι αποθηκευμένη στην κρυφή μνήμη είναι πρόσφατη, επιλέγεται αυτή για απάντηση, διαφορετικά το αίτημα προωθείται στην συσκευή. Με τον τρόπο αυτό αποφεύγονται άσκοπες αιτήσεις προς τις συσκευές, που έχει ως αποτέλεσμα λιγότερη κατανάλωση ενέργειας, λιγότερο φόρτο στο δίκτυο και στις συσκευές, οι οποίες αν κατακλιστούν από αιτήματα είναι πιθανόν να παρουσιάσουν σφάλμα. Ουσιαστικά το σύστημα δρα ως αντιπρόσωπος με κρυφή μνήμη και ο τρόπος λειτουργίας του φαίνεται στις εικόνες 6.3 και 6.4.



Εικόνα 6.4 Παράδειγμα κρυφής μνήμης



Εικόνα 6.3 Παράδειγμα κρυφής μνήμης

Ο συνδυασμός των παραπάνω χαρακτηριστικών δίνει την δυνατότητα στο σύστημα να επανέρχεται και να επιδιορθώνει σφάλματα που μπορεί να προκύψουν κατά την λειτουργία των αισθητήρων, κάτι που είναι αρκετά συχνό φαινόμενο λόγω της φύσης των εφαρμογών. Οι συσκευές εξάλλου που απαρτίζουν το ασύρματο δίκτυο αισθητήρων είναι περιορισμένων δυνατοτήτων και αρκετές φορές έχουν ως αποτέλεσμα κάποια απρόοπτη επανεκκίνηση. Άλλες φορές χάνονται μηνύματα ή οι συσκευές βγαίνουν εκτός εμβέλειας για κάποιο χρονικό διάστημα. Ο συνδυασμός του "I am alive", με τα Observe και τα αξιόπιστα μηνύματα επιτρέπουν στο σύστημα να ανακάμπτει από τέτοια προβλήματα χωρίς κάποια ενέργεια, ενώ ακόμα και αν δεν τα καταφέρει, στην κρυφή μνήμη έχει μείνει η τελευταία απάντηση της συσκευής και η χρονική στιγμή που αυτό έγινε, ώστε να βοηθήσει τον διαχειριστή να εντοπίσει το πρόβλημα.

Το σύστημα λειτουργεί ως CoAP διακομιστής, επιτρέποντας σε έναν χρήστη του διαδικτύου να επικοινωνήσει με αυτό μέσω CoAP μηνυμάτων από συμβατές συσκευές. Με την χρήση του Copper plug-in η πρόσβαση είναι εφικτή και από τον browser. Πέρα από την πρόσβαση στις συσκευές του 1ου επιπέδου, έχουν υλοποιηθεί διάφορα αντικείμενα που είναι διαθέσιμα:

1 /.well-known/core

2 activeRequests

3 pendingRequests

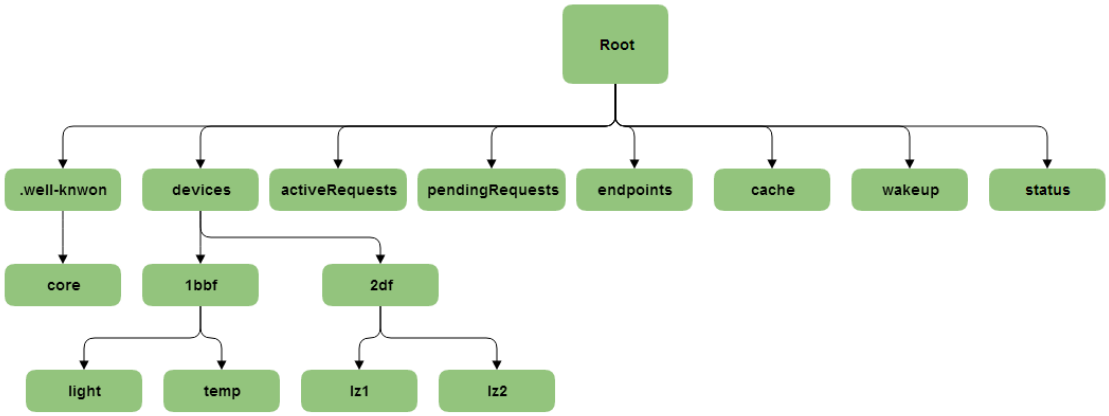
4 endpoints

5 cache

6 wakeup

7 status

Ως διακομιστής οφείλει να έχει το αντικείμενο "`/.well-known/core`" ώστε να ενημερώνονται οι πελάτες για τα διαθέσιμα αντικείμενα. Κάθε συσκευή του 1ου επιπέδου που είναι συνδεδεμένη στο σύστημα εμφανίζεται κάτω από το όνομα "`/device/`" ακολουθούμενη από το αναγνωριστικό της συσκευής (πχ `1bbf` άρα συνολικά είναι: "`/device/1bbf`"). Ακολουθούν τα διαθέσιμα αντικείμενα της κάθε συσκευής (πχ "`/device/1bbf/light`"). Μέσω αυτών των Uri-Path ο πελάτης έχει πρόσβαση στα ξεχωριστά αντικείμενα. Η αίτησης που φτάνει στον διακομιστή επεξεργάζεται και αν πρέπει να σταλεί στην συσκευή, δημιουργείται νέο μήνυμα, βασισμένο στην αρχική αίτηση αλλά με διαφορετικά Uri-Path ώστε να γίνουν κατανοητά από την συσκευή (πχ το "`device/1bbf/light`", γίνεται "`/light`" και στέλνεται στην συσκευή `1bbf`).



Εικόνα 6.5 Τα αντικείμενα που παρέχει το σύστημα στον χρήστη

Ο διακομιστής δέχεται αιτήσεις μέσω του διαδικτύου, οπότε πρέπει να κρατάει δομές για τους πελάτες και τα χαρακτηριστικά τους. Για κάθε πελάτη δημιουργείται μία εγγραφή με την IP διεύθυνσή του, την θύρα, το αναγνωριστικό της συσκευής στο οποίο στάλθηκε η αίτηση και το Message ID. Επίσης, αν η αίτηση περιείχε Token option τότε αποθηκεύεται και αυτό. Το χαρακτηριστικό αυτό είναι απαραίτητο για το Observe πρωτόκολλο, διότι ο διακομιστής όταν διαβάζει μια ειδοποίηση από το δίκτυο με το συγκεκριμένο Token, αναζητά τον πελάτη στον οποίο αντιστοιχεί. Η συγκεκριμένη δομή μπορεί να ανακτηθεί μέσω του "`/pendingRequests`" με το αποτέλεσμα να είναι όπως στην εικόνα 6.6.

Device	IP:PORT	Message ID	Token
191	/87.203.79.99:63754	20088	BA3C
42f	/150.140.5.102:57725	6963	-

Εικόνα 6.6 Παράδειγμα του "/pendingRequests"

Παράλληλα, πρέπει να διατηρεί δομές για κάθε αίτηση που στέλνει ο διακομιστής προς τις συσκευές. Αποθηκεύει το αναγνωριστικό της συσκευής, το αντικείμενο που ζητήθηκε, το Message ID του μηνύματος, το Token και την χρονική στιγμή που συνέβη. Αν πρόκειται για αιτήσεις παρατήρησης αντικειμένου, αποθηκεύεται ο αριθμός των ειδοποιήσεων που έχει λάβει από την στιγμή της αίτησης. Μέσω αυτού ο διακομιστής μπορεί να αντιστοιχεί τις απαντήσεις που λαμβάνει από τις συσκευές, με τις αιτήσεις που έχει στείλει. Η συγκεκριμένη δομή μπορεί να ανακτηθεί μέσω του "`/activeRequests`" με το αποτέλεσμα να είναι όπως στην εικόνα 6.7.

Device	Token	Message ID	Resource	Timestamp	Notifications
46e	--	22867	/.well-known/core	1,38023E+12	0
296	--	20129	/.well-known/core	1,38022E+12	0
1ccd	--	20138	/.well-known/core	1,38022E+12	0
897a	--	19592	/.well-known/core	1,38022E+12	0
8979	AA 38 39 37 39 F7 9C 00	56434	/pir	1,38023E+12	6621
f15	AA 66 31 35 7F 9D 11 BB	78	/pir	1,3801E+12	68
4ec	AA 34 65 63 FC 55 6D DF	35445	/temp	1,38023E+12	3158
931	AA 39 33 31 0D A5 73 42	21508	/light	1,38023E+12	5373
42f	AA 34 32 66 84 E9 26 45	23452	/light	1,38023E+12	676
2df	AA 32 64 66 BC B9 9B 21	11946	/lz4	1,38023E+12	1236
997a	AA 39 39 37 61 A2 D5 01	46351	/light	1,38023E+12	684
42f	AA 34 32 66 42 9B 88 D4	10951	/pir	1,38023E+12	1379
995d	AA 39 39 35 64 21 38 31	65440	/light	1,38021E+12	11
1cde	AA 31 63 64 65 CD B4 1B	25092	/light	1,38023E+12	510
931	AA 39 33 31 CE 3C F0 CB	21510	/pir	1,38023E+12	16853
494	AA 34 39 34 F4 F3 AB 84	6932	/lz1	1,38023E+12	3522
2df	AA 32 64 66 E2 D5 1A 76	11945	/lz2	1,38023E+12	1249

Εικόνα 6.7 Παράδειγμα του "/activeRequests"

Το αντικείμενο "`/endpoints`" περιέχει την τελευταία χρονική στιγμή που υπήρχε επικοινωνία με το κάθε αντικείμενο κάθε συσκευής. Το χαρακτηριστικό αυτό επιτρέπει στον διαχειριστή να εντοπίσει πιθανόν ελαττωματικές συσκευές που για κάποιο λόγο έπαψαν να επικοινωνούν με το σύστημα. Όπως φαίνεται στην εικόνα 6.8 η συσκευή "`f15`" έπαψε να επικοινωνεί.

Time	URI
Wed Sep 25 12:46:42 EEST 2013	f15/.well-known/core
Wed Sep 25 12:46:25 EEST 2013	f15/lz1
Wed Sep 25 12:45:59 EEST 2013	f15/temp
Wed Sep 25 12:47:35 EEST 2013	f15/pir
Wed Sep 25 12:46:04 EEST 2013	f15/light
Thu Sep 26 22:06:04 EEST 2013	897a/.well-known/core
Thu Sep 26 21:52:31 EEST 2013	897a/temp
Thu Sep 26 21:48:33 EEST 2013	897a/parent
Thu Sep 26 21:49:06 EEST 2013	897a/light
Thu Sep 26 23:22:08 EEST 2013	191/.well-known/core
Thu Sep 26 23:20:20 EEST 2013	191/parent
Thu Sep 26 23:18:14 EEST 2013	191/temp
Thu Sep 26 23:18:15 EEST 2013	191/light
Thu Sep 26 23:22:29 EEST 2013	190/.well-known/core
Thu Sep 26 23:18:53 EEST 2013	190/pir
Thu Sep 26 23:22:17 EEST 2013	190/parent

Εικόνα 6.8 Παράδειγμα του "/endpoints"

Παρόλο που το σύστημα υποστηρίζει αυτόματη ρύθμιση των συσκευών, μπορεί να υπάρξουν συσκευές που δεν στέλνουν το "I am alive" μήνυμα. Μέσω του αντικειμένου "`/wakeup`" μπορεί ο χρήστης να αναγκάσει το σύστημα να κάνει αυτόματη ρύθμιση μιας συγκεκριμένης συσκευής, σαν να είχε λάβει το "I am alive" από αυτήν. Με τον τρόπο αυτό, οι συγκεκριμένες συσκευές προστίθενται εύκολα στο σύστημα, αρκεί να είναι γνωστό το αναγνωριστικό της συσκευής.

Το αντικείμενο "`/cache`" παρουσιάζει τα περιεχόμενα όλης της κρυφής μνήμης. Κάθε εγγραφή περιέχει το αναγνωριστικό της συσκευής, το αντικείμενο στο οποίο αναφέρεται, η τιμή του, η χρονική στιγμή της αποθήκευσης και τα δευτερόλεπτα που έχουν περάσει από τότε, καθώς και πόσες φορές χάθηκε κάποια ειδοποίηση με αποτέλεσμα η εγγραφή να θεωρηθεί μη-πρόσφατη. Στην εικόνα 6.9 φαίνεται παράδειγμα του αντικειμένου και όσες εγγραφές δεν είναι πρόσφατες έχουν μαρκαριστεί με αστεράκι στο πεδίο Age.

Device	Resource	Value	Time	Age	Observes lost
120	/light	0	Thu Sep 26 23:20:44 EEST 2013	269sec	6
120	/temp	30	Thu Sep 26 23:20:35 EEST 2013	278sec	5
130	/light	0	Thu Sep 26 23:21:14 EEST 2013	239sec	9
130	/temp	30	Thu Sep 26 23:22:40 EEST 2013	153sec	18
152f	/light	0	Thu Sep 26 23:23:44 EEST 2013	88sec	0
152f	/temp	32	Thu Sep 26 23:20:52 EEST 2013	260sec	4
1538	/light	0	Thu Sep 26 23:23:28 EEST 2013	104sec	1
1538	/temp	30	Thu Sep 26 23:23:27 EEST 2013	105sec	0
190	/pir	0	Thu Sep 26 23:23:53 EEST 2013	79sec	10
191	/light	0	Thu Sep 26 23:23:15 EEST 2013	118sec	8
191	/temp	31	Thu Sep 26 23:23:16 EEST 2013	117sec	5
1b77	/light	0	Thu Sep 26 23:20:29 EEST 2013	284sec	7
1b77	/temp	28	Thu Sep 26 23:22:57 EEST 2013	136sec	7
1bbf	/temp	4294967169	Thu Sep 26 16:27:21 EEST 2013	25072sec *	0
1bed	/light	0	Thu Sep 26 23:20:50 EEST 2013	263sec	2
1bed	/temp	32	Thu Sep 26 23:20:47 EEST 2013	266sec	1
1ccd	/light	5	Thu Sep 26 19:52:43 EEST 2013	12750sec *	47
1ccd	/pir	0	Thu Sep 26 19:59:18 EEST 2013	12354sec *	57
1ccd	/temp	29	Thu Sep 26 19:53:08 EEST 2013	12725sec *	40
1cde	/light	0	Thu Sep 26 23:24:51 EEST 2013	21sec	3
1cde	/temp	27	Thu Sep 26 23:25:07 EEST 2013	5sec	2
296	/light	4	Thu Sep 26 22:12:05 EEST 2013	4388sec *	32
296	/parent	0x80c	Thu Sep 26 22:12:02 EEST 2013	4391sec *	29
296	/temp	30	Thu Sep 26 22:14:03 EEST 2013	4270sec *	34

Εικόνα 6.9 Παράδειγμα του "/cache"

Τέλος, το αντικείμενο "`/status`" παρουσιάζει πληροφορίες για το κεντρικό σύστημα, όπως φαίνεται στην εικόνα 6.10.

```
Online
Uptime:0 days, 22 hours, 28 min, 34 sec
Running Threads:593
Cache Size:29 nodes
Pending Connections:1
Used Memory:160 MB
Free Memory:167 MB
Total Memory:327 MB
Max Memory:860 MB
Version:1.0
Build:

****
.well-known/core requests: 87665
Observe Requests:8543
Observe Responses:139496
Observe Lost:8412
```

Εικόνα 6.10 Παράδειγμα του "/status"

7 Αξιολόγηση Λειτουργιών

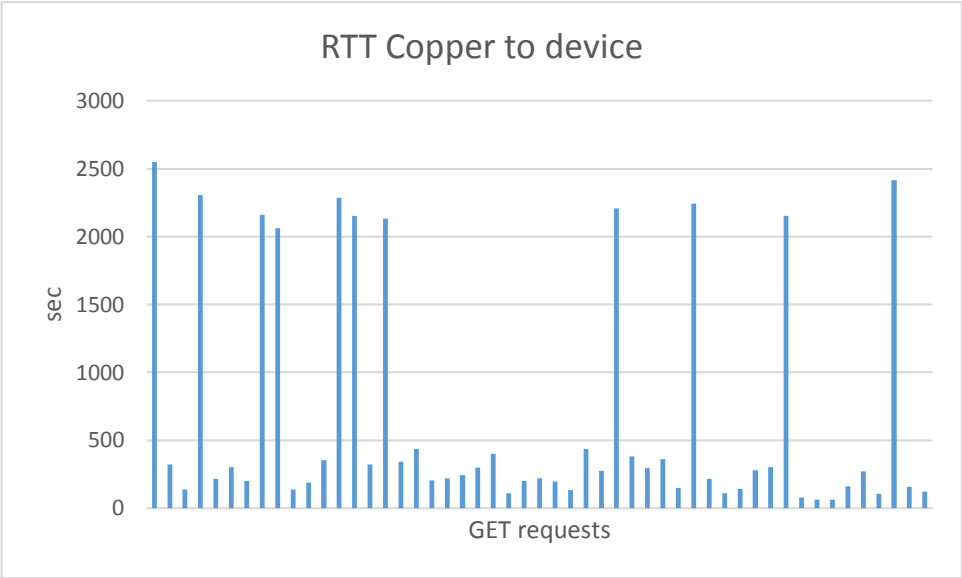
7.1 Απόκριση συστήματος

Εξετάζεται η ταχύτητα απόκρισης του συστήματος. Για τις μετρήσεις χρησιμοποιείται το Round Trip Time (RTT), δηλαδή ο χρόνος που απαιτείται να ολοκληρωθεί μια αίτηση από την στιγμή που στέλνεται η αίτηση μέχρι την λήψη της απάντησης. Στην εικόνα 7.1 φαίνονται τα σημεία που εισάγουν καθυστερήσεις.

Εικόνα 7.1

Η καθυστέρηση που εισάγει η σύνδεση του πελάτη με το Überdust για τις συγκεκριμένες μετρήσεις είναι περίπου 11 ms (μετρήθηκε κάνοντας ping τον διακομιστή), ενώ 14 ms για CoAP αιτήσεις. Αυτό μπορεί να δικαιολογηθεί από το γεγονός ότι για να απαντηθεί και η πιο απλή αίτηση, το σύστημα πρέπει να την επεξεργαστεί, ενώ η απάντηση στο ping είναι σχεδόν άμεση.

Αρχικά έγιναν μετρήσεις απόκρισης του συστήματος από άκρη σε άκρη (end to end), δηλαδή από τον πελάτη (μέσω Copper) μέχρι μία συσκευή του ασύρματου δικτύου και πίσω. Τα αποτελέσματα παρουσιάζονται στην εικόνα 7.2. Αυτό που παρατηρήθηκε είναι ότι με πιθανότητα 22% υπάρχει κάποια απώλεια πακέτου, είτε αίτησης, είτε απάντησης, με αποτέλεσμα το Copper μετά τα 2 δευτερόλεπτα αναμονής να μεταδίδει ξανά το πακέτο. Συνολικά, το μέσο RTT είναι 663,2 ms. Αν όμως αφαιρεθούν οι περιπτώσεις των χαμένων πακέτων, τότε το μέσο RTT υπολογίζεται στα 229 ms. Αντίθετα, το μέσο RTT των χαμένων πακέτων είναι 2211.5 ms, πολύ κοντά στο μέσο RTT χωρίς σφάλματα προσαυξημένο κατά 2000 ms που είναι το timeout.



Εικόνα 7.2 RTT

Στο κομμάτι του ασύρματου δικτύου έγιναν πειράματα για τον υπολογισμό του RTT στην επικοινωνία συσκευής με συσκευή. Η τιμή αυτή υπολογίστηκε στα 35 ms. Επειδή η τοπολογία του δικτύου είναι ενός hop αυτή είναι και η μέση καθυστέρηση, από την στιγμή που μία αίτηση θα σταλεί από την πύλη μέχρι να φτάσει η απάντηση πίσω σε αυτήν.

Το συμπέρασμα είναι ότι περίπου 180 ms απαιτούνται για την επικοινωνία του κεντρικού συστήματος με την πύλη, παρόλο που βρίσκονται σε κοινό ενσύρματο δίκτυο. Ενδιάμεσα όμως βρίσκεται η λειτουργία του TestbedRuntime καθώς και του TestbedListener, των οποίων η απόκριση δεν μπορεί να μετρηθεί. Σε ιδανικές συνθήκες, όπου όλες οι συσκευές είναι συνδεδεμένες με το σύστημα και χρησιμοποιούνται αποκλειστικά από αυτό, δεν θα ήταν απαραίτητο να υπάρχει το TestbedRuntime, οπότε η καθυστέρηση θα περιοριζόταν στην απόκριση της σειριακής θύρας επικοινωνίας.

Με την χρήση της κρυφής μνήμης φαίνεται πως μπορεί να μειωθεί σημαντικά η απόκριση του συστήματος σε εξωτερικές αιτήσεις από χρήστες. Με την κρυφή μνήμη οι αιτήσεις απαντώνται επ' ευθείας από το σύστημα σε 14 ms κατά μέσο όρο, ενώ αν η αίτηση έπρεπε να φτάσει μέχρι την συσκευή, ο χρόνος ανέρχεται στα 663.2 ms. Ακόμα και αν το δίκτυο ήταν τέλειο και δεν υπήρχαν χαμένα πακέτα, ο χρόνος θα ήταν 229 ms. Βέβαια εάν εξαλειφθεί η ανάγκη του TestbedRuntime τότε μπορεί τα αποτελέσματα να μην είναι τόσο θεαματικά.

7.2 Observe

Μετρήσεις έγιναν και στις εσωτερικές αιτήσεις του συστήματος, καταγράφοντας πόσες αιτήσεις για παρατήρηση αντικειμένου έγιναν, πόσες ειδοποιήσεις στάλθηκαν και πόσες χάθηκαν. Το αποτέλεσμα είναι ότι χάνεται κατά μέσο όρο 17.3% των ειδοποιήσεων. Παρατηρώντας την προέλευση των χαμένων ειδοποιήσεων, το 89% έχουν προέλευση κάποια Arduino συσκευή. Επί του

συνόλου των ειδοποιήσεων, το 72% έχουν προέλευση κάποιο Arduino. Οι ενεργές Arduino συσκευές κατά την διάρκεια των πειραμάτων ήταν 8, ενώ τα iSense 19. Συνολικά, από αυτά προκύπτει ότι, στα iSense υπάρχει πιθανότητα 8.6% να χαθεί μία ειδοποίηση, ενώ στα Arduino η πιθανότητα αυτή γίνεται 25.9%.

8 Συμπεράσματα και μελλοντικές κατευθύνσεις

8.1 Συμπεράσματα

Με το σύστημα που παρουσιάστηκε μία σειρά λειτουργιών αυτοματοποιούνται και δεν χρειάζονται επιπλέον πόροι για την πραγματοποίησή τους. Οι συσκευές πλέον απαιτούν την δημιουργία μιας εφαρμογής στα πλαίσια του CoAP, με αυτό να αναλαμβάνει όλες τις επιμέρους λειτουργίες στα κατώτερα επίπεδα. Δημιουργείται πλέον το κατάλληλο επίπεδο αφαίρεσης, ενώ αυτοματοποιούνται πολλές λειτουργίες, όπως η αξιόπιστη μετάδοση μηνυμάτων και η παροχή ειδοποιήσεων. Μια τέτοια εφαρμογή, σαφώς ανεβάζει το επίπεδο πολυπλοκότητας συνολικά και φέρνει κάποιες συσκευές στα όριά τους, μπορεί δηλαδή να μην ενδείκνυται η χρήση του CoAP για εξαιρετικά απλές εφαρμογές διότι αυτά που παρέχει το πρωτόκολλο θα είναι αχρείαστα. Βοηθάει όμως στην δημιουργία μιας ενιαίας πλατφόρμας, ενός προτύπου που μελλοντικά ενδέχεται να μονοπωλήσει στα embedded συστήματα και τα ασύρματα δίκτυα αισθητήρων, όπως το HTTP στα γενικής χρήσης συστήματα.

Παρουσιάστηκε ένα ολοκληρωμένο σύστημα παρακολούθησης ασύρματων δικτύων αισθητήρων βασισμένο στο CoAP. Παρέχει εύκολο έλεγχο και διαχείριση των συσκευών κάνοντας την προσθαφαίρεσή τους απλή, με τον διαχειριστή να μην χρειάζεται να κάνει τίποτα άλλο πέρα από το να εγκαταστήσει την συσκευή. Το σύστημα φροντίζει να παρακολουθεί τους αισθητήρες και να παρέχει στους τελικούς χρήστες τις πιο πρόσφατες τιμές του, ενώ παράλληλα βάσει αυτών, αυτόματες ενέργειες μπορούν εύκολα να προγραμματιστούν. Στην πραγματική χρήση το σύστημα απέδειξε πως μπορεί να βελτιώσει θεαματικά την απόκριση του συστήματος με την εισαγωγή της κρυφής μνήμης. Προβλήματα που προέκυψαν στην πορεία δεν επηρέασαν την λειτουργία του συστήματος, αντίθετα λύθηκαν αυτόματα όταν αυτό κατέστει δυνατό.

8.2Μελλοντικές κατευθύνσεις

Το CoAP συνεχώς εξελίσσεται, με την μορφή του να έχει διαφοροποιηθεί σε μερικά σημεία αρκετά από την συγκεκριμένη υλοποίηση. Με την παρουσίαση του τελικού προτύπου, οι υλοποιήσεις θα πρέπει να ανανεωθούν για να υποστηρίξουν πλήρως και σωστά το πρότυπο. Μέχρι αυτό να συμβεί, βελτιώσεις μπορούν να γίνουν σε διάφορα τμήματα της υλοποίησης για την περεταίρω εξοικονόμηση πόρων από τις συσκευές, οι οποίες συχνά αδυνατούν να εξυπηρετήσουν πολλαπλά μηνύματα και οι εφαρμογές περιορίζονται από την διαθέσιμη μνήμη. Τα αποτελέσματα των μετρήσεων έδειξαν ότι τα Arduino υστερούν λίγο σε σχέση με τα iSense σε επίπεδο απόδοσης, οπότε υπάρχει σίγουρα περιθώριο για βελτίωση.

Το κεντρικό σύστημα αντίθετα που δεν έχει τέτοιο περιορισμό σε πόρους μπορεί να αναπτυχθεί ακόμα περισσότερο, με προσθήκη γνωστών από το HTTP δυνατότητων, όπως κάποια DNS υπηρεσία για την πιο εύκολη πρόσβαση στις συσκευές. Παράλληλα μπορεί να γίνει ένα κατάλληλο περιβάλλον αλληλεπίδρασης με τις συσκευές, ώστε το CoAP να μεταφράζεται και σε HTTP και να παρέχεται πρόσβαση σε αντικείμενα χωρίς την χρήση κάποιου CoAP πελάτη ή του Copper plug-in.

Παράρτημα : Εικόνες

Βιβλιογραφία

[1] IEEE Computer Society, IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs).

[2] Z. Shelby, Sensinode, K. Hartke, C. Bormann, Universitaet Bremen TZI, B. Frank, SkyFoundry. Constrained Application Protocol (CoAP). Internet-Draft, IETF, November 2011. Work in Progress.

[3] Roy Thomas Fielding. Abstract of the Dissertation, Chapter 5, Representational State Transfer (REST). University of California, Irvine, 2000

[4] Constrained RESTful Environments (CoRE) Working Group. https://datatracker.ietf.org/wg/core/charter/, September 2013

[5] G. Montenegro, Microsoft Corporation, N. Kushalnagar, Intel Corp, J. Hui, D. Culler, Arch Rock Corp, Transmission of IPv6 Packets over IEEE 802.15.11 Networks, RFC 4944, September 2007

[6] NXP Laboratories UK Ltd (Formerly Jennic Ltd). JN5148-001 Data Sheet. 2012.

[7] ZigBee Home automation public application profile. Technical report, ZigBee Alliance, 2010

[8] Coalesenses GmbH. iSense Core Module Data Sheet. 2008.

[9] Digi International, Inc. XBee/XBee Pro ZB RF Modules. 2010.

[10] O. Akribopoulos, V. Georgitzikis, A. Protopapa, and I. Chatzigiannakis. Building a platform-agnostic wireless network of interconnected smart objects. In 15th Panhellenic Conf. on Informatics with international participation, PCI'11.

[11] T. Baumgartnerm I. Chatzigiannakis, S. P. Fekete, C. Koninis, A. Kroller and A. Pyrgelis. Wiselib: A generic algorithm library for heterogeneous sensor networks. In 7th European Conf. on Wireless Sensor Networks, EWSN 2010, pages 162-177.

[12] A. Kroeller, D. Pfisterer, C. Buschmann, S.P. Fekete, S. Fischer. Shawn: A new approach to simulating wireless sensor networks, Proceedings of the Design, Analysis, and Simulation of Distributed Systems (DASD 05), San Diego, 2005.

[13] Coulson Geoff, Porter Barry, Chatzigiannakis Ioannis, Koninis Christos, Fischer Stefan, Pfisterer Dennis, Bimschas Daniel, Braun Torsten, Hurni Philipp, Anwander Markus, Wagenknecht Gerald, Fekete Sandor P., Kröller Alexander, Baumgartner Tobias. Flexible experimentation in wireless sensor networks. In Commun. ACM, ACM, volume 55, 2012.

[14] Daniel Bimschas, Dennis Pfisterer. Testbed Runtime – The Wisebed WSN Testbed Infrastructure Software. PIK – Praxis der Informationsverarbeitung und Kommunikation, Volume 36, Issue 1, page 54. February 2013

[15] Überdust - a data storage and brokerage web service for IoT data providers, data consumers and application developers. https://github.com/Uberdust/webapp/wiki

[16] I. Fette, Google Inc, A. Melnikov, Isode Ltd. The WebSocket Protocol. RFC 6455 IETF, December 2011.

[17] J. Klensin, M. Padlipsky. Unicode Format for Network Interchange. Network Working Group. RFC 5198. March 2008.

[18] E, Rescorla, RTFM Inc, N. Modadugu. Datagram Transport Layer Security. Network Working Group. RFC 4347. Stanford University, April 2006.

[19] S. Kent, BBN Technologies. IP Encapsulating Security Payload (ESP). Network Working Group. RFC 4303. December 2005.

[20] D. Cooper, NIST, S. Santesson, Microsoft, S. Farrell, Trinity College Dublin, S. Boeyen, Entrust, R. Housley, Vigil Security. W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Network Working Group. RFC 5280. NIST, May 2008.

[21] K. Hartke. Observing Resources in CoAP. CoRE Working Group. Internet-Draft. Universitaet Bremen TZI, February 2012.

[22] Z. Shelby. CoRE Link Format. IETF. RFC 6690. August 2012.

[23] C. Bormann, Z. Shelby. Blockwise transfers in CoAP. CoRE Working Group. Internet-Draft. 2012.

[24] Matthias Kovatsch, Daniel Pauli, Dominique Im Obersteg. Californium (Cf) CoAP Framework in Java. Spring 2010.

[25] Matthias Kovatsch. Copper (Cu) CoAP user-agent for Firefox. 2012

[26] Amaxilatis Dimitrios, Georgitzikis Vasileios, Giannakopoulos Dimitrios, Chatzigiannakis Ioannis. Employng Internet of Things Technologies for Building Automation. ETFA 2012.