Πανεπιστήμιο Πατρών
Πολυτεχνική Σχολή
Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής

Κατανεμημένοι Αλγόριθμοι Επικοινωνίας Ετερογενών Αδόμητων Ασύρματων Δικτύων

Ιωάννης Παπαβασιλείου
ΑΜ: 3718

Επιβλέπων: Καθ. Ζαρολιάγκης Χρήστος
Συνεπιβλέπων: Δρ. Χατζηγιαννάκης Ιωάννης

Πάτρα, Ιούνιος 2011

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντά μου, Καθηγητή κ. Ζαρολιάγκη Χρήστο, ο οποίος με τίμησε με τη συνεργασία του, τις συμβουλές του και μου έδωσε την δυνατότητα να εντρυφήσω σε ένα πολύ ενδιαφέρον γνωστικό πεδίο. Επίσης, θα ήθελα να ευχαριστήσω το συνεπιβλέποντα Δρ. Χατζηγιαννάκη Ιωάννη για την πολύτιμη βοήθεια, καθοδήγηση και τις γνώσεις που μου μετέδωσε.

Ακόμη, ευχαριστώ τους Ορέστη Ακριβόπουλο, Χρήστο Κονίνη και Βασίλειο Γεωργιτζίκη. Ο Ορέστης με βοήθησε σε όλα τα μήκη και πλάτη αυτής της διπλωματικής, ο Χρήστος και ο Βασίλης μου παρείχαν τον κώδικα για το iSense και Arduino αντίστοιχα και γι΄ αυτό τους ευχαριστώ θερμά.

Επίσης θέλω να ευχαριστήσω τον Απόστολο Μάστορη και τους άλλους φίλους μου για την κατανόηση και την συμπαράσταση που μου έδειξαν όλο αυτό τον καιρό. Τέλος ευχαριστώ την οικογένειά μου για την παροχή όλων των μέσων για την επίτευξη των ονείρων μου.

Παπαβασιλείου Ιωάννης,
Πάτρα, Ιούνιος 2011

Περίληψη

Στην παρούσα διπλωματική εργασία γίνεται μελέτη για την δημιουργία ενός ασύρματου ετερογενούς δικτύου αισθητήρων. Οι συσκευές που χρησιμοποιούμε υλοποιούν το πρωτόκολλο MAC IEEE802.15.4, παρόλα αυτά η μεταξύ τους επικοινωνία δεν είναι εφικτή. Αρχικά εστιάζουμε στα προβλήματα που εμποδίζουν την επικοινωνία, προτείνοντας λύσεις. Στη συνέχεια αναπτύσσουμε αλγόριθμο γειτνίασης ώστε να επιτραπεί η συνδεσιμότητα των συσκευών. Έπειτα, παρουσιάζουμε μια σειρά από πειράματα ώστε να αξιολογήσουμε την ρυθμοαπόδοση των συσκευών, και την ποιότητα της επικοινωνίας, έπειτα συγκρίνουμε τις αποδόσεις των συσκευών στις μετρήσεις και παραθέτουμε μελλοντικές επεκτάσεις της παρούσας εργασίας.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο και σημασία

Βρισκόμαστε στην εποχή της περιρρέουσας υπολογιστικής νοημοσύνης και της επικοινωνίας, η οποία ριζικά έχει μετατρέψει την συλλογική, κοινωνική και προσωπική μας σφαίρα. Ο Mark Weiser (πατέρας του όρου περιρρέουσας υπολογιστικής νοημοσύνης – ubiquitous computing) παρατήρησε ότι 'οι πιο βαθιές τεχνολογίες είναι αυτές που εξαφανίζονται... υφαίνουν τους εαυτούς τους μαζί με το ύφασμα της καθημερινής ζωής μέχρις ότου να είναι δυσδιάκριτες από αυτήν'. Αυτό αποδεικνύεται εύκολα στην κινητή τηλεφωνία. Ο αριθμός των κινητών τηλεφώνων το 2010 ξεπέρασε τα 5 δισεκατομμύρια[1] παγκοσμίως. Αυτές οι μικροσυσκευές έχουν γίνει ένα αναπόσπαστο και οικείο μέρος της καθημερινής ζωής πολλών εκατομμυρίων ανθρώπων, ακόμη περισσότερο και από το διαδίκτυο.

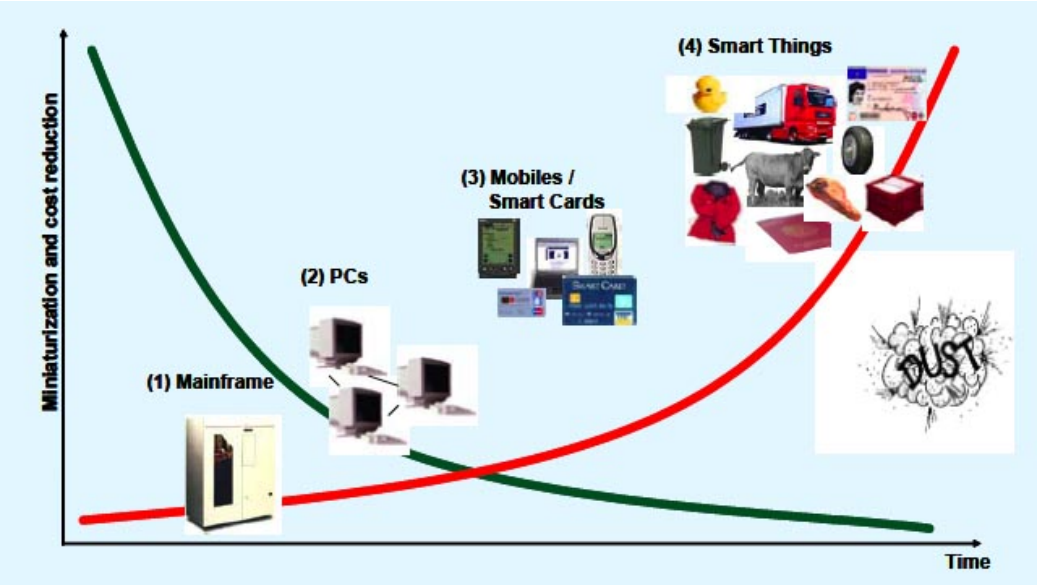
Σήμερα, η ανάπτυξη είναι στο δρόμο να πάει αυτό το φαινόμενο ένα βήμα παραπέρα, ενσωματώνοντας μικρού εύρους ασύρματους πομποδέκτες μέσα σε ένα ευρύ φάσμα επιπρόσθετων καθημερινών αντικειμένων, και ενεργοποιώντας νέο είδος επικοινωνίας μεταξύ ανθρώπων και αντικειμένων, και μεταξύ των ίδιων των αντικειμένων. Μια νέα διάσταση έχει προστεθεί στον κόσμο της πληροφορίας και των τεχνολογιών επικοινωνίας: το *οποτεδήποτε, οπουδήποτε* συνδεσιμότητα για *οποιονδήποτε*, θα γίνει συνδεσιμότητα για *οποιονδήποτε* και

[1] πηγή: http://www.bbc.co.uk/news/10569081

οτιδήποτε.

Οι συνδέσεις θα πολλαπλασιάσουν και θα δημιουργήσουν ένα καινούργιο δίκτυο – το Διαδίκτυο των Πραγμάτων [6] [15]. Το Διαδίκτυο των Πραγμάτων δεν είναι ούτε επιστημονική φαντασία ούτε βιομηχανική διαφήμιση, αλλά βασίζεται σε σταθερές τεχνολογικές προόδους που διαφαίνονται ενθουσιωδώς. Είναι μια τεχνολογική επανάσταση που αντιπροσωπεύει το μέλλον των υπολογιστών και των επικοινωνιών, η ανάπτυξή του εξαρτάται από την δυναμική τεχνολογική καινοτομία σε έναν αριθμό σημαντικών πεδίων, από ασύρματους αισθητήρες μέχρι την νανοτεχνολογία.

Εισάγωντας νοημοσύνη στα πράγματα–αντικείμενα μπορεί να ενισχύσει την δύναμη του δικτύου μεταβιβάζοντας δυνατότητες επεξεργασίας πληροφορίας στα άκρα του δικτύου, ενώ οι πρόοδοι της μικρογράφησης και της νανοτεχνολογίας σημαίνουν ότι μικρότερα και εξυπνότερα πράγματα θα έχουν την δυνατότητα να αλληλεπιδράσουν και να συνδεθούν (Σχήμα 1.1). Ένας συνδυασμός αυτής της ανάπτυξης θα δημιουργήσει ένα Διαδίκτυο των Πραγμάτων το οποίο συνδέει τα αντικείμενα με ένα αισθητήριο και έξυπνο τρόπο.



Σχήμα 1.1: Ανάπτυξη έξυπνων αντικειμένων

Είναι κατανοητό ότι τέτοια δίκτυα με τόσες διαφορετικές εφαρμογές και δυνατότητες έχουν σαφώς μεγάλη ετερογένεια. Η ετερογένεια αυτή μπορεί να

αναφέρεται είτε σε θέματα υλικού (hardware) είτε σε θέματα λογισμικού (software). Δεν είναι εφικτό, παρά την μεγάλη τεχνολογική πρόοδο που αναφέραμε προηγουμένως, να υπάρχει ένας κόμβος που να ικανοποιεί όλες τις απαιτήσεις ενός τέτοιου δικτύου. Αυτό θα οδηγούσε σε απαγορευτικό κόστος, κάτι το οποίο δεν είναι επιθυμητό. Αν γινόταν αυτό σίγουρα κάποιες απαιτήσεις δεν θα ικανοποιούνταν.

Το ζήτημα που τίθεται εδώ δεν είναι το πως θα αποφύγουμε την ετερογένεια τέτοιων δικτύων. Αντίθετα, η ετερογένεια είναι άκρως επιθυμητή και μπορεί να αυξήσει την διάρκεια ζωής του δικτύου [18], αλλά και να πολλαπλασιάσει τις δυνατότητές του. Χρειαζόμαστε κόμβους στο δίκτυο με ειδικές–απλές δυνατότητες, όπως οι αισθητήρες, αλλά χρειαζόμαστε και κόμβους με περισσότερες δυνατότητες, όπως κόμβοι που είναι διασυνδεδεμένοι με άλλα επίπεδα του δικτύου (πύλες). Αυτό που χρειάζεται είναι μια ολοκληρωμένη λύση που θα λαμβάνει υπόψιν και θα εκμεταλλεύεται τις ιδιαιτερότητες κάθε κόμβου.

1.2 Στόχος διπλωματικής εργασίας

Στόχος τις διπλωματικής αυτής είναι να αξιοποιήσουμε τις υπάρχουσες τεχνολογίες ασύρματων δικτύων αισθητήρων για να δημιουργήσουμε ένα ετερογενές αδόμητο δίκτυο. Επίσης ένας επιπλέον στόχος είναι να αξιολογήσουμε τα αποτελέσματα αυτής της επικοινωνίας μεταξύ των κόμβων του δικτύου που θα δημιουργήσουμε.

Το δίκτυο αυτό θα αποτελείται από συσκευές που έχουν ασύρματους πομποδέκτες οι οποίοι ακολουθούν το πρότυπο IEEE 802.15.4. Κάθε πλατφόρμα έχει δικές τις ιδιαιτερότητες όσον αφορά το υλικό και το λογισμικό. Οι πλατφόρμες που θα χρησιμοποιηθούν γι΄ αυτό το σκοπό είναι οι: SunSOT, TelosB, Arduino και iSense. Παρόλο που όλες οι πλατφόρμες ακολουθούν το ίδιο πρωτόκολλο MAC, η ανταλλαγή μηνυμάτων δεν είναι δυνατή. Στόχος της εργασίας αυτής είναι λοιπόν να εντοπιστούν οι λόγοι που συμβαίνει αυτό και να προταθεί κάποια λύση στο πρόβλημα. Μέλημα επίσης είναι η δημιουργία κάποιου αλγόριθμου που θα επιτρέπει την συνδεσιμότητα μεταξύ των κόμβων του δικτύου. Επιπλέον στόχος είναι η μέτρηση της απόδοσης επικοινωνίας αλλά και της ποιότητας αυτής χρησιμοποιώντας αρκετές μετρικές γι΄ αυτό το σκοπό.

1.3 Συνεισφορά διπλωματικής εργασίας

Στο πλαίσιο της παρούσας διπλωματικής εργασίας προτάθηκαν και υλοποιήθηκαν λύσεις για την παράκαμψη των προβλημάτων επικοινωνίας που μόλις αναφέραμε. Βρέθηκαν οι λόγοι που είναι μη επιτρεπτή η επικοινωνία μεταξύ των συσκευών και επιτεύχθηκε η επιτυχής ανταλλαγή μηνυμάτων από και προς όλες τις τέσσερις αυτές πλατφόρμες.

Ακόμη αναπτύχθηκε αλγόριθμος που επιτρέπει την ανακάλυψη και σε επόμενο στάδιο σύνδεση δύο γειτονικών συσκευών, ο αλγόριθμος αυτός πήρε το όνομα πρωτόκολλο Echo. Το πρωτόκολλο Echo υλοποιεί, ακόμη, και μια υποτυπώδη δρομολόγηση πακέτων απλού άλματος, επιτρέποντας ταχύτατη προσαρμογή στις αλλαγές της τοπολογίας του πακέτου.

Επιτεύχθηκε παράλληλα η προσθήκη θυρών σε κάθε αποστελλόμενο μήνυμα (port messaging) ώστε οι εφαρμογές χρήστη που θα χρησιμοποιήσουν το πρωτόκολλο να μπορούν να διακρίνουν τα μηνύματά τους σε κατηγορίες. Τέλος παρουσιάζουμε μια σειρά πειραμάτων ώστε να μετρηθεί η ρυθμοαπόδοση του δικτύου, καθώς και η ποιότητα των συνδέσμων.

Είναι πρώτη φορά που γίνεται προσπάθεια δημιουργίας ενός τέτοιου δικτύου από αυτές τις τέσσερις διαφορετικές πλατφόρμες. Γι αυτό το λόγο κρίνεται πολύ σημαντική η πραγματοποίηση αυτών των μετρήσεων ώστε να δούμε τι δυνατότητες μπορεί να έχει αυτό το νέο ετερογενές κατανεμημένο δίκτυο. Παρόμοια εργασία έχει γίνει στο παρελθόν [1] μόνο για δύο από τις συσκευές που αναφέραμε (SunSOT και TelosB).

1.4 Δομή διπλωματικής εργασίας

Κατά την συγγραφή αυτής της διπλωματικής εργασίας έγινε προσπάθεια να ακολουθηθεί μια λογική ροή, ώστε ο αναγνώστης να κατανοήσει τα βήματα που ακολουθήθηκαν με χρονολογική σειρά και να φτάσουμε τελικά στο επιθυμητό αποτέλεσμα.

Στο κεφάλαιο 2 θα εξετάσουμε μερικές από τις υπάρχουσες εφαρμογές ασύρματων δικτύων αισθητήρων και θα διαπιστώσουμε την ετερογένεια που υφίσταται σε αυτές τις εφαρμογές. Στη συνέχεια θα γίνει μια μικρή παρουσίαση

των πλατφορμών SunSOT, TelosB, Arduino και iSense από πλευράς χρησιμοποιημένης τεχνολογίας λογισμικού και υλικού.

Στο κεφάλαιο 3 θα παρουσιάσουμε το πρωτόκολλο MAC IEEE 802.15.4, δείχνοντας τις βασικές του προδιαγραφές. Έπειτα θα αναλύσουμε την υλοποίηση που έχει γίνει σε κάθε πλατφόρμα που παρουσιάσαμε στο κεφάλαιο 2 εστιάζοντας στις μεταξύ τους διαφορές. Θα τελειώσουμε το κεφάλαιο με προτάσεις για την λύση των διαφορετικών υλοποιήσεων σε κάθε μία από τις τέσσερις πλατφόρμες.

Στο κεφάλαιο 4 θα παρουσιαστεί το πρωτόκολλο Echo για την επίτευξη της συνδεσιμότητας μεταξύ των συσκευών και την δρομολόγηση των ασύρματων μηνυμάτων–πακέτων. Το πρωτόκολλο Echo προσφέρει ταχύτατη αναζήτηση γειτονικών συσκευών και δρομολόγηση απλού άλματος (single–hop routing).

Στο κεφάλαιο 5 θα γίνει αξιολόγηση της δυνατότητας επικοινωνίας μεταξύ αυτών των τεσσάρων συσκευών. Συγκεκριμένα θα παρουσιαστούν αποτελέσματα από τα πειράματα που έγιναν σε θέματα ικανότητας διαβίβασης δεδομένων, απώλειας σταλθέντων μηνυμάτων, θέματα ισχύος των μηνυμάτων που στάλθηκαν και, τέλος, της ποιότητας ζεύξης μεταξύ των συσκευών.

Θα κλείσουμε με το κεφάλαιο 6 στο οποίο γίνεται αναφορά για τα συμπεράσματα αυτής της εργασίας καθώς και μελλοντική επέκταση της εργασίας αυτής.

Κεφάλαιο 2

Συσκευές συνδεδεμένες μέσω του διαδικτύου

2.1 Εισαγωγή

Στόχος του Διαδικτύου των Πραγμάτων [6] (Internet of Things) είναι να παρέχει την αρχιτεκτονική και τα τεχνολογικά θεμέλια για να αναπτύξει κατανεμημένα συστήματα αξιόπιστα, ενήμερα του πλαισίου (context-aware) και αποδοτικά σε θέματα ενέργειας τα οποία περιέχουν συνεργαζόμενους αισθητήρες και άλλες έξυπνες συσκευές και αντικείμενα. Αυτό θα μπορεί να ενεργοποιήσει την ανθρώπου/αντικειμένου και αντικειμένου/αντικειμένου επικοινωνία βασισμένη στο διαδίκτυο και να ανοίξει ένα νέο εύρος υπηρεσιών διαδικτύου. Οι βασικές προκλήσεις της αρχιτεκτονικής είναι να κινηθεί πέρα από τα σύνορα των τομέων της αρχικής αντίληψης του Διαδικτύου των Πραγμάτων, με σκοπό να αντιμετωπίσει την ετερογένια των υποκειμένων τεχνολογιών, και να ενσωματώσει νέες υποστηριζόμενες υπηρεσίες.

2.2 Εφαρμογές δικτύων αισθητήρων

Υπάρχει πληθώρα εφαρμογών δικτύων αισθητήρων, τόσο ερευνητικών αλλά και εμπορικών. Τα δίκτυα αισθητήρων έκαναν την εμφανισή τους κυρίως για στρατιωτικές εφαρμογές. Η ιδέα ήταν να τοποθετηθούν διάφοροι κρυφοί

αισθητήρες στην περιοχή του εχθρού ώστε οι στρατιωτικές δυνάμεις να έχουν την άμεση ενημέρωση για τις κινήσεις του εχθρού.

Τα δίκτυα αυτά όμως αναπτύχθηκαν και επεκτάθηκαν και σε άλλες περιοχές ενδιαφέροντος. Βασική εφαρμογή των δικτύων αισθητήρων είναι η παρακολούθηση περιοχής–ζώνης. Οι εφαρμογές εστιάζουν κυρίως στην καταγραφή των συνθηκών του περιβάλλοντος όπως θερμοκρασία, υργρασία, ατμοσφαιρική πίεση, ηλιοφάνεια, μόλυνση του αέρα. Επίσης υπάρχουν εφαρμογές για την ανίχνευση φωτιάς, κατολισθήσεων αλλά και παρακολούθηση καταστασης δικτύων μεταφοράς καυσίμων, όπως πετρέλαιο και υγραέριο, και δικτύων ύδρευσης–άρδευσης–αποχέτευσης. Αλλες εφαρμογές των δικτύων αισθητήρων είναι σε δίκτυα μεταφοράς όπως αυτοκίνητα, τραίνα και πλοία.

Νέες ιδέες για την επέτκαση των εφαρμογών αυτών των δικτύων με στόχο το Διαδίκτυο των Πραγμάτων είναι εφαρμογές που είναι πιο κοντά στην καθημερινότητά μας. Αυτές περιλαμβάνουν ενσωμάτωση αισθητήρων στους φωτεινούς σηματοδότες για την ρύθμιση της κυκλοφορίας στους δρόμους, ενσωμάτωση αισθητήρων στα ψυγεία για την καλύτερη διατήρηση των τροφίμων και την ενημέρωση μας για ξεχασμένα τρόφιμα πριν την ημερομηνία λήξης. Αυτές οι εφαρμογές έχουν να κάνουν με τα έξυπνα σπίτια, Σχήμα 2.1.

2.3 Ανάγκη για ετερογένεια

Βλέπουμε λοιπόν ότι η ετερογένεια είναι εγγενές στοιχείο αυτών των εφαρμογών. Κάθε κόμβος του δικτύου έχει διαφορετικές απαιτήσεις. Οι διαφορετικές απαιτήσεις μπορεί να ανταποκρίνονται είτε στο λογισμικό είτε στο υλικό του κόμβου. Υπάρχουν κόμβοι που έχουν βασική προϋπόθεση την ελάχιστη κατανάλωση ενέργειας, με σκοπό την μακροζωία, ενώ υπάρχουν κόμβοι που έχουν βασική απαίτηση τον υψηλή ασύρματη κάλυψη του δικτύου και πιθανώς έχουν κάποια σταθερή τροφοδοσία.

Από μεριάς λογισμικού υπάρχει επίσης ετερογένεια. Υπάρχουν κόμβοι που χρειάζονται να παρέχουν ελλάχιστες υπηρεσίες στο δίκτυο, όπως οι αισθητήριοι κόμβοι. Σε αυτή την περίπτωση θέλουμε λογισμικό που απλά να έχει πρόσβαση στην διεπαφή του αισθητήρα που βρίσκεται συνδεδεμένος στην συσκευή και απλά να καταγράφει την μέτρηση του αισθητήρα, προοθώντας την σε κάποιο

Σχήμα 2.1: Έξυπνο σπίτι

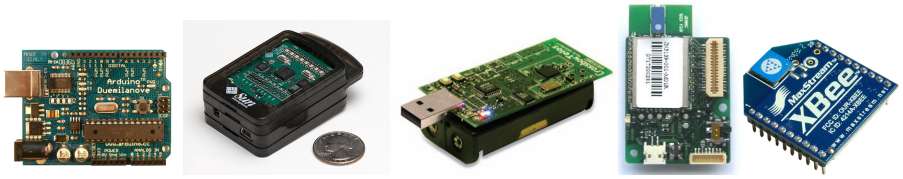
υψηλότερο επίπεδο της αρχιτεκτονικής. Αντίθετα μπορεί να υπάρχουν κόμβοι με πολύ περισσότερες απαιτήσεις υπολογισμών και προσφερόμενων υπηρεσιών. Τέτοιοι κόμβοι μπορεί να είναι οι πύλες οι οποίες παρέχουν την διασύνδεση του δικτύου με το διαδίκτυο.

2.4 Πλατφόρμες που χρησιμοποιήσαμε

Σε αυτή την ενότητα θα παρουσιάσουμε τις πλατφόρμες–τεχνολογίες που χρησιμοποιήσαμε για την ανάπτυξη του ετερογενούς ασύρματου δικτύου. Οι συσκευές φαίνονται στο Σχήμα 2.2

2.4.1 SunSPOT

Το Sun SPOT [11] (SUN Small Programable Object Technology) (Σχήμα 2.2β′) είναι μια μικρή ασύρματη συσκευή με ενσωματωμένους αισθητήρες που λειτουργεί με μπαταρία και εκτελεί την εικονική μηχανή της Java: Squawk



(α′) Arduino (β′) SunSPOT (γ′) TelosB (δ′) iSense (ε′) xBee

Σχήμα 2.2: Οι συσκευές που χρησιμοποιήσαμε

JVM. Αναπτύχθηκε στα εργαστήρια της Sun Microsystems (θυγατρική της Oracle πλέον) και ο κώδικας που εκτελεί είναι γραμμένος σε Java 2 Micro Edition (J2ME).

Ένα free-range SunSPOT διαθέτει ένα processor board, ένα sensor board και την μπαταρία. Στο processor board είναι συνδεδεμένος ο επεξεργαστής και ο πομποδέκτης. Ο επεξεργαστής του είναι ο 32-bit ARM920T που είναι χρονισμένος στα 180MHz. Διαθέτει μνήμη RAM 512KB και εξωτερική μνήμη Flash 4MB. Ο πομποδέκτης είναι ο CC2420 Chipcon που υλοποιεί το πρωτόκολλο mac IEEE 802.15.4 [19]. Το sensor board διαθέτει αισθητήρες θερμοκρασίας, φωτός και επιταχυνσιόμετρο 3 αξόνων. Επιπλέον έχει 8 LED τριών χρωμάτων, 6 αναλογικές εισόδους που διαβάζονται μέσω Analog to Digital Converter, 2 κουμπιά, 5 pins εισόδου/εξόδου γενικού σκοπού και 4 pins εξόδου υψηλού ρεύματος. Διαθέτει επίσης θύρα mini-USB για την φόρτιση και τον προγραμματισμό του.

2.4.2 TelosB

Το TelosB [12] (Σχήμα 2.2γ′) είναι μια ασύρματη συσκευή συμβατή με το πρωτόκολλο mac IEEE 802.15.4 [19]. Διαθέτει μικροεπεξεργαστή TI-MSP430 16-bit που είναι χρονισμένος στα 4MHz. Έχει μνήμη RAM 10KB και εξωτερική μνήμη Flash 48KB. Η τροφοδοσία του γίνεται είτε μέσω της θύρας USB που διαθέτει για συλλογή δεδομένων και τον προγραμματισμό του, είτε μέσω της υποδοχής για 2 μπαταρίες AA που διαθέτει. Το TelosB έχει λειτουργικό σύστημα TinyOS [13] και το λογισμικό που εκτελεί είναι γραμμένο στην γλώσσα προγραμματισμού nesC. Ο πομποδέκτης που διαθέτει είναι ίδιος με αυτόν του SunSPOT, ο CC2420.

Το TinyOS είναι λειτουργικό σύστημα ανοιχτού κώδικα, σχεδιασμένο για ασύρματες συσκευές μικρής ισχύος. Αυτές μπορεί να είναι είτε συσκευές που χρησιμοποιούνται σε δίκτυα αισθητήρων, είτε ενσωματωμένες (embedded) συσκευές, κτλ. Ο προγραμματισμός στο TinyOS βασίζεται σε συνιστώσες (component-based programming). Κάθε διεργασία του συστήματος είναι τοποθετημένη σε διαφορετικές συνιστώσες έτσι ώστε όλα τα δεδομένα και οι συναρτήσεις μέσα σε μια συνιστώσα να είναι σημασιολογικά συσχετισμένες. Οι συνιστώσες είναι συνδεδεμένες μεταξύ τους μέσω διεπαφών (interfaces). Κάθε συνιστώσα παρέχει και χρησιμοποιεί άλλες συνιστώσες για να αποπερατώσει το έργο της. Το TinyOS παρέχει συνιστώσες για την διαχείριση μερών του συστήματος όπως: επικοινωνία, δρομολόγηση πακέτων, αισθητήρων και αποθήκευσης.

Το TinyOS έχει μία στοίβα (stack) με αποτέλεσμα να είναι απολύτως non-blocking. Έτσι διεργασίες που καθυστερούν αρκετά, όπως αποστολή μηνυμάτων, εκτελούνται στο παρασκήνιο και όταν ολοκληρωθούν πυροδοτείται η εκτέλεση γεγονότων (events) υπεύθυνων για την διαχείρησή τους. Συνεπώς, κάθε εφαρμογή που είναι γραμμένη για το TinyOS υπακούει στο μοντέλο προγραμματισμού εφαρμογών οδηγούμενων από γεγονότα (event-driven applications).

2.4.3 iSense

Το iSense [7] (Σχήμα 2.2δ΄) αναπτύχτηκε από την Coalesenses GmbH που έχει βάση στο Λούμπεκ της Γερμανίας. Το iSense Core module περιλαμβάνει τον πομποδέκτη JN5139 που χρησιμοποιεί το πρωτόκολλο mac IEEE 802.15.4 Zigbee. Ο επεξεργαστής του είναι ο 32 bit RISC Controller που είναι χρονισμένος στα 16MHz. Διαθέτει μνήμη RAM 96KB και εξωτερική σειριακή μνήμη Flash 128KB. Διαθέτει υποδοχές επέκτασης για όλα τα είδη από άλλες μονάδες και ενεργειακές πηγές. Διαθέτει επίσης ρυθμιστή τάσης ελεγχόμενη από το λογισμικό, ο οποίος συνδυάζει υψηλή ενεργειακή απόδοση με ένα ευρύ φάσμα τάσης εφοδιασμού. Επίσης διαθέτει και υποδοχή κεραίας SMA εκτός από την ενσωματωμένη κεραμική κεραία.

Το λειτουργικό σύστημα που διαθέτει και το λογισμικό που μπορεί να ε-

κτελέσει είναι γραμμένο σε C++. Το iSense λογισμικό σύστημα παρέχει ένα μεγάλο αριθμό από έτοιμες προς εκτέλεση υπηρεσίες και πρωτόκολλα, όπως δρομολόγηση, χρονικό συγχρονισμό, προγραμματισμό *πάνω από τον αέρα* (ασύρματα), αξιόπιστη μεταφορά μεμονωμένων πακέτων αλλά και ροής πακέτων (streams), υπηρεσίες λειτουργικού συστήματος και οδηγούς (drivers) ανεξαρτήτως πλατφόρμας για μια ευρεία ποικιλία αισθητήρων και άλλων επεκτάσεων.

2.4.4 Arduino Duemilanove – XBee

Το Arduino [3] (Σχήμα 2.2α′) είναι μια πλατφόρμα ανοιχτού κώδικα. Το Arduino Duemilanove είναι μια μικροελεγκτική συσκευή βασισμένη στον μικροεπεξεργαστή ATmega328. Η μνήμη RAM του έχει μέγεθος 16KB. Μπορεί να *αισθανθεί* το περιβάλλον και να το επηρεάσει ελέγχοντας φώτα, κινητήρες, κτλ. Διαθέτει 14 ακροδέκτες εισόδου/εξόδου, 6 αναλογικές εισόδους, ένα ψηφιακό ταλαντωτή 16MHz, μία θύρα USB, ακροδέκτη τροφοδοσίας, ένα διασυνδετικό αγωγό ICSP και ένα κουμπί επαναφοράς (reset button). Οι βασικές βιβλιοθήκες του Arduino είναι γραμμένες στην γλώσσα C και C++ και έχουν μεταγλωττιστεί χρησιμοποιώντας τους μεταγλωττιστές avr-gcc και Avr Libc. Προγραμματίζεται χρησιμοποιώντας την γλώσσα προγραμματισμού Wiring [16], η οποία μοιάζει με την C++, με κάποιες απλοποιήσεις και αλλαγές. Δεν διαθέτει κάποιο λειτουργικό σύστημα όπως οι υπόλοιπες συσκευές, διαθέτει όμως πληθώρα βιβλιοθηκών και οδηγών (drivers) που κάνουν την ανάπτυξη του κώδικα πολύ εύκολη. Το Arduino δεν έχει ενσωματωμένο πομποδέκτη, γι′ αυτό το λόγο το συνδέουμε με ένα XBee Series 1 [17] κύκλωμα κεραίας. Το XBee (Σχήμα 2.2ε′) παρέχει δικτυακή σύνδεση στο Arduino μέσω του πρωτοκόλλου mac IEEE 802.15.4 [19].

Στον παρακάτω πίνακα φαίνονται συγκεντρωμένες όλες οι πλατφόρμες, Πίνακας 2.1.

	Arduino	**SunSPOT**	**TelosB**	**iSense**
Επεξεργαστής	ATmega328	ARM920T	MSP430	JN5139
Συχνότητα λειτουργίας(MHz)	16	180	16	16
Μνήμη RAM	16 KB	512 KB	10 KB	96 KB
Μνήμη Flash	32 KB	4 MB	48 KB	128 KB
Πομποδέκτης	XBee Series 1	CC2420	CC2420	JN5139
Γλώσσα προγραμματισμού	Wiring (C++)	J2ME	nesC	C++
Λειτουργικό Περιβάλλον	—	Squawk	TinyOS	iSense

Πίνακας 2.1: Σύγκριση των πλατφορμών.

Κεφάλαιο 3

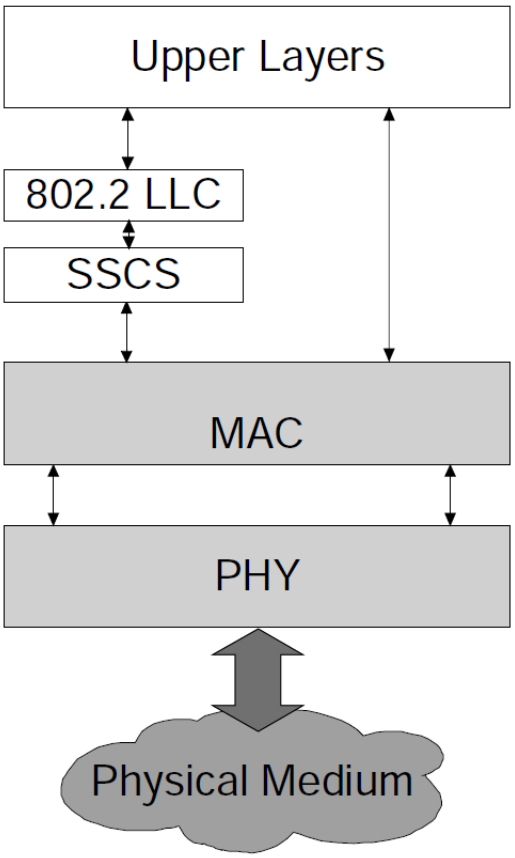
Mac IEEE 802.15.4 και διαφορετικές υλοποιήσεις

3.1 Εισαγωγικά

Όπως έχουμε δει, οι συσκευές που θα χρησιμοποιήσουμε για την ανάπτυξη του δικτύου μας χρησιμοποιούν το πρωτόκολλο Mac IEEE 802.15.4 [19]. Το πρωτόκολλο αναπτύχθηκε από την ομάδα εργασίας 4 του IEEE 802.15 [20] και έχει σκοπό να ορίσει το φυσικό επίπεδο (Physical Layer – PHY) και το επίπεδο πρόσβασης μέσου (Medium Access Control – MAC), Σχήμα 3.1, για ασύρματη συνδεσιμότητα χαμηλού ρυθμού. Οι συσκευές που μπορεί να συμμετέχουν σε μία τέτοια διασύνδεση μπορεί να είναι σταθερές ή κινητές, με μηδενική ή πολύ περιορισμένη απαίτηση για κατανάλωση ενέργειας από μπαταρία, λειτουργώντας σε *προσωπική* περιοχή των 10 μέτρων (personal operating space – POS).

Τα ασύρματα δίκτυα προσωπικής περιοχής (wireless personal area networks – WPANs) χρησιμοποιούνται για την μετάδοση πληροφορίας σε σχετικά μικρές αποστάσεις. Αντίθετα από τα ασύρματα τοπικά δίκτυα (wireless local area networks – WLANs), οι συνδέσεις που πραγματοποιούνται στα WPANs εμπλέκουν μικρή ή καθόλου υποδομή. Αυτό το χαρακτηριστικό επιτρέπει σε μικρές, αποδοτικής κατανάλωσης, οικονομικές λύσεις να υλοποιηθούν σε ένα ευρύ φάσμα συσκευών.

Ο σκοπός του 802.15.4 είναι να παρέχει ένα πρότυπο για εξαιρετικά χαμηλή



Σχήμα 3.1: Η αρχιτεκτονική της στοίβας δικτύου.

πολυπλοκότητα, εξαιρετικά χαμηλό κόστος, εξαιρετικά χαμηλή κατανάλωση ενέργειας και χαμηλούς ρυθμούς δεδομένων ασύρματης συνδεσιμότητας μεταξύ φθηνών συσκευών. Ο ρυθμός μετάδοσης θα πρέπει να είναι υψηλός αρκετά (με μέγιστη τιμή τα $250\,kb/sec$), ώστε να ικανοποιήσει ένα σύνολο απλών αναγκών, όπως διαδραστικά παιχνίδια, αλλά κλιμακώσιμο στις απαιτήσεις των αισθητήρων και τις ανάγκες της αυτοματοποίησης ($20\,kb/sec$ ή λιγότερο), για ασύρματες επικοινωνίες.

Λίγο διαφορετικό πρότυπο είναι το Bluetooth [4], το οποίο είναι μια ασύρματη τεχνολογία για συστήματα επικοινωνίας μικρού εύρους, που προορίζεται για να αντικαταστήσει τα καλώδια που συνδέουν φορητές ή/και σταθερές ηλεκτρονικές συσκευές. Στην έκδοση 4 του Bluetooth υπάρχουν δύο τύποι

ασύρματης τεχνολογίας: η βασικού ρυθμού (Basic Rate – BR) και η χαμηλής ενέργειας (Low Energy – LE). Ο τύπος βασικού ρυθμού περιέχει προαιρετικούς ενισχυμένους ρυθμούς δεδομένων (Enhanced Data Rate – EDR), εναλλακτικό επίπεδο πρόσβασης μέσου (Alternate Medium Access Control – MAC) και φυσικό επίπεδο (Physical Layer – PHY). Επίσης οι ενισχυμένοι ρυθμοί δεδομένων που παρέχει μπορεί να φτάσουν τα $24\,Mbps$ (EDR). Αντίθετα ο τύπος χαμηλής ενέργειας (LE) περιέχει χαρακτηριστικά τα οποία είναι σχεδιασμένα να ενεργοποιούν προϊόντα τα οποία απαιτούν χαμηλότερα επίπεδα κατανάλωσης ενέργειας, χαμηλότερη πολυπλοκότητα και χαμηλότερο κόστος σε σχέση με τον τύπο (BR/EDR).

Παρατηρούμε λοιπόν ότι ο τύπος χαμηλής ενέργειας (LE) του Bluetooth έχει χαρακτηριστικά που μπορεί να είναι κατάλληλα για τις πλατφόρμες που παρουσιάσαμε στο προηγούμενο κεφάλαιο. Ο προσανατολισμός του όμως είναι περισσότερο για συσκευές με συγκεκριμένες δυνατότητες, όπως ρολόγια, αισθητήρες σώματος, αυτοματισμούς γραφείου–σπιτιού, αισθητήρες μέσα στο αυτοκίνητο κτλ. Έτσι λοιπόν το 802.15.4 κρίνεται ιδανικό για τις πλατφόρμες που παρουσιάσαμε. Στη συνέχεια θα δείξουμε τα βασικά χαρακτηριστικά του, θα εξετάσουμε τα πεδία του προτύπου που είναι σημαντικά για την επικοινωνία αυτών των τεσσάρων διαφορετικών συσκευών, καθώς και τις απαραίτητες αλλαγές που χρειάστηκε να γίνουν, ώστε να επιτευχθεί η μεταξύ τους επικοινωνία.

3.2 Γενικά χαρακτηριστικά

Το IEEE 802.15.4 είναι σχεδιασμένο για να μπορεί να υποστηρίζει δίκτυα με λίγους κόμβους μέσα σε ένα κτήριο αλλά και χιλιάδες κόμβους σε κάποια γεωγραφική περιοχή. Οι κόμβοι αυτοί μπορεί να δημιουργήσουν κυκλοφορία είτε με κάποιο σταθερό ρυθμό είτε με πιθανές εκρήξεις. Έτσι λοιπόν ο φόρτος του δικτύου είναι απρόβλεπτος αλλά εξαρτάται σε μεγάλο βαθμό από την εφαρμογή. Η λίστα που ακολουθεί παρουσιάζει κάποια από τα βασικά χαρακτηριστικά του πρωτοκόλλου.

- Ρυθμός μετάδοσης δεδομένων $250\,kb/s$, $40\,kb/s$ και $20\,kb/s$
- Λειτουργία αστέρα ή peer-to-peer

- Εύρος διευθύνσεων 16 bit ή 64 bit
- Κατανομή εγγυημένων χρονοθυρίδων (guaranteed time slots – GTSs)
- Πολλαπλή πρόσβαση καναλιού με ανίχνευση φέροντος και αποφυγή συγκρούσεων (Carrier sense multiple access with collision avoidance – CSMA-CA)
- Αξιόπιστη μεταφορά με Acknowledgement μηνύματα
- Χαμηλή κατανάλωση ενέργειας
- Ανίχνευση Ενέργειας (Energy detection – ED)
- Ένδειξη ποιότητας σύνδεσης (Link quality indication – LQI)
- 16 κανάλια μέσα στη 2450 MHz ζώνη, 10 κανάλια στη 915 MHz ζώνη και 1 κανάλι στη 868 MHz ζώνη

3.3 Μορφή πλαισίων MAC

Τα πλαίσια MAC (MAC frames) είναι συλλογές δεδομένων που στέλνονται μέσω του φυσικού επιπέδου (PHY). Για κάθε πακέτο δεδομένων που έρχεται από τα ανώτερα επίπεδα, το επίπεδο MAC προσθέτει δικά του δεδομένα ώστε να επιτευχθεί η ασύρματη επικοινωνία. Έτσι δημιουργείται το πλαίσιο MAC. Η δομή των πλαισίων έχει σχεδιαστεί για να διατηρεί την πολυπλοκότητα απλή και ταυτόχρονα να το καθιστά αρκετά ισχυρό για την μετάδοση σε ένα θορυβώδες κανάλι. Κάθε επίπεδο πρωτοκόλλου προσθέτει στη δομή δεδομένα στην αρχή και στο τέλος του πλαισίου (headers and footers). Κάθε πλαίσιο έχει μέγεθος μέχρι 127 bytes και αποτελείται από τις ακόλουθες βασικές συνιστώσες.

1. **MAC header – MHR**: Η κεφαλίδα του πλαισίου
2. **MAC payload**: Το ωφέλιμο φορτίο του πλαισίου, το οποίο περιέχει πληροφορίες σχετικές με τον τύπο του πλαισίου. Εδώ υπάρχουν και τα δεδομένα από τα υψηλότερα επίπεδα.

3. **MAC footer – MFR**: Η κατάληξη του πλαισίου, η οποία περιέχει μια σειρά ελέγχου (frame check sequence)

Η μορφή του γενικού πλαισίου φαίνεται στο Σχήμα 3.2.

<table>
<tr><th>Octets: 2</th><th>1</th><th>0/2</th><th>0/2/8</th><th>0/2</th><th>0/2/8</th><th>variable</th><th>2</th></tr>
<tr><td rowspan="2">Frame control</td><td rowspan="2">Sequence number</td><td>Destination PAN identifier</td><td>Destination address</td><td>Source PAN identifier</td><td>Source address</td><td rowspan="2">Frame payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="6">MHR</td><td>MAC payload</td><td>MFR</td></tr>
</table>

Σχήμα 3.2: Γενικό πλαίσιο MAC.

MAC header

Όπως είδαμε και στα γενικά χαρακτηριστικά, το 802.15.4 παρέχει δυνατότητα διευθυνσιοδότησης 16 *bit* ή 64 *bit*. Αυτό καθορίζεται μέσα στο πεδίο Frame control, του οποίου τα περιεχόμενα φαίνονται στο Σχήμα 3.3. Στα bits

Bits: 0–2	3	4	5	6	7–9	10–11	12–13	14–15
Frame type	Security enabled	Frame pending	Ack. request	Intra-PAN	Reserved	Dest. addressing mode	Reserved	Source addressing mode

Σχήμα 3.3: MAC Frame contol.

0-2 του Frame control καθορίζεται ο τύπος του πλαισίου. Τα bits αυτά μπορούν να πάρουν διάφορες τιμές, καθορίζοντας αν το πλαίσιο θα είναι τύπου Beacon, δεδομένων, αναγνώρισης (Acknowledgment) ή διαχείρισης MAC. Οι συσκευές που εξετάζουμε δεν υποστηρίζουν beacon-based τρόπο λειτουργίας και υποστηρίζουν μόνο πλαίσια δεδομένων και αναγνώρισης. Τα πλαίσια δεδομένων χρησιμοποιούνται για την αποστολή των δεδομένων από τα υψηλότερα επίπεδα (π.χ. επίπεδο εφαρμογής). Ο αποστολέας των δεδομένων μπορεί να ζητήσει, για το πλαίσιο που έστειλε, την αναγνώριση της επιτυχούς λήψης από τον παραλήπτη. Έτσι ο παραλήπτης είναι υποχρεωμένος να απαντήσει με

ένα πλαίσιο αναγνώρισης (Acknowledgment) μέσα σε 12 συμβολικές περιόδους (symbol periods), δηλαδή 192 μsec. Το bit 6 από το υποπεδίο Frame control μας δείχνει μια άλλη δυνατότητα του 802.15.4. Σε αυτό το bit καθορίζεται αν το πλαίσιο έχει προορισμό κάποιο προσωπικό δίκτυο με διαφορετικό αναγνωριστικό από αυτό που ανήκει η συσκευή. Τα bits 10-11 (Destination addressing mode) και τα 14-15 (Source addressing mode) δίνουν την δυνατότητα καθορισμού του είδους διευθυνσιοδότησης. Οι τιμές που μπορούν να πάρουν φαίνονται στο Σχήμα 3.4. Με αυτό τον τρόπο δίνεται η δυνατότητα καθορισμού διαφορε-

Addressing mode value $b_1\, b_0$	**Description**
00	PAN identifier and address field are not present.
01	Reserved.
10	Address field contains a 16 bit short address.
11	Address field contains a 64 bit extended address.

Σχήμα 3.4: Τρόποι διευθυνσιοδότησης.

τικού εύρους διευθυνσιοδότησης στον αποστολέα απ΄ ότι στον παραλήπτη. Αν επιθυμούμε διευθυνσιοδότηση 16 *bit* σε αποστολέα και παραλήπτη θα πρέπει τα bits 10-11 και τα 14-15 του πεδίου Frame control να πάρουν την τιμή 10.

Η οκτάδα 1 από το γενικό πλαίσιο (Σχήμα 3.2) περιέχει τον αριθμό ακολουθίας (sequence number), ο οποίος χρησιμοποιήται για την ταυτοποίηση των πλαισίων αναγνώρισης (Acknowledgment MAC Frames). Το επίπεδο MAC διαιρεί όλους τους κόμβους που λειτουργούν στο ίδιο κανάλι σε πολλαπλά προσωπικά δίκτυα (PAN). Κάθε ένα από τα δίκτυα έχει το μοναδικό του αναγνωριστικό. Το πεδίο αναγνωριστικού προσωπικού δικτύου προορισμού (Destination PAN identifier), Σχήμα 3.2, έχει εύρος 16 bits και καθορίζει το μοναδικό αναγνωριστικό του δικτύου που έχει ο παραλήπτης. Αν αυτό το πεδίο πάρει τιμή 0xffff τότε πρόκειται για broadcast αναγνωριστικό και έτσι θα γίνει αποδεκτό από κάθε συσκευή που *ακούει* στο κανάλι. Το πεδίο διεύθυνσης παραλήπτη (Destination address field) έχει, όπως είδαμε, μεταβλητό εύρος (16 *bit* ή 64 *bit*), και καθορίζει την διεύθυνση του παραλήπτη. Τα πεδία αναγνωριστικού προσωπικού δικτύου αποστολέα (Source PAN identifier) και διεύθυνσης αποστολέα (Source address field) καθορίζουν τις αντίστοιχες τιμές για το αναγνωριστικό

και την διεύθυνση στον αποστολέα του πλαισίου.

MAC payload

Το ωφέλιμο φορτίο του πλαισίου έχει μεταβλητό μέγεθος και περιέχει πληροφορίες σχετικά με μεμονωμένα είδη πλαισίων. Αν το υποπεδίο ενεργοποίησης ασφάλειας (security enabled subfield) έχει την τιμή 1 (Σχήμα 3.3) τότε το ωφέλιμο φορτίο προστατεύεται όπως έχει οριστεί η ασφάλεια γι΄ αυτή τη σχέση.

MAC footer

Η κατάληξη του πλαισίου περιέχει την σειρά ελέγχου, η οποία έχει μέγεθος 16 *bit* και έχει ITU-T κυκλικό έλεγχο πλεονασμού (CRC). Η σειρά ελέγχου υπολογίζεται μέσω του ακόλουθου πολυωνύμου γεννήτορα, βαθμού 16:
$G_{16}(x) = x^{16} + x^{12} + x^5 + 1$

3.4 Υλοποιήσεις του πρωτοκόλλου

Οι πλατφόρμες που παρουσιάσαμε υλοποιούν το 802.15.4 με διαφορετικό τρόπο, χωρίς να είναι συμβατοί αυτοί οι τρόποι μεταξύ τους. Γι αυτό το λόγο η επικοινωνία των τεσσάρων αυτών συσκευών δεν είναι δυνατή χωρίς να υπάρξει κάποια τροποποίηση σε κάποια από αυτές. Από την προηγούμενη ενότητα βλέπουμε ότι υπάρχει η δυνατότητα σε αυτές τις συσκευές να έχουν διευθυνσιοδότηση 16 *bit* ή 64 *bit*. Παρόλα αυτά η στοίβα δικτύου του SunSPOT υποστηρίζει μόνο διευθύνσεις των 64 *bit*, ενώ το TelosB υποστηρίζει μόνο διευθύνσεις των 16 *bit*. Οι στοίβες των Arduino και iSense υποστηρίζουν και τα δύο είδη διευθυνσιοδότησης.

Βασισμένο στις προδιαγραφές του LowPan [8], η βιβλιοθήκη του SunSPOT παρέχει δρομολόγηση (routing), προώθηση πακέτων σε άλλους κόμβους (meshing) και κατακερματισμό πακέτων (fragmentation), δημιουργώντας πακέτα μεγέθους μεχρι και 1260 bytes. Το LowPan προσθέτοντας επιπλέον κεφαλίδες στα 802.15.4 πλαίσια επιτυγχάνει να δημιουργήσει αυτές τις δυνατότητες. Γνωρίζουμε ότι όταν γίνει λήψη ενός πλαισίου που έχει το bit “ACK” στην τιμή 1

ότι πρέπει να απαντήσει με ένα πλαίσιο τύπου acknowledgement (σε 12 συμβολικές περιόδους). Αυτό στο SunSPOT υλοποιείται με την αυτόματη απάντηση που παρέχει το κύκλωμα ασύρματης επικοινωνίας. Αντίθετα το TinyOS αφήνει την διαχείριση αυτής της απαίτησης στο λογισμικό του χρήστη, καθώς με την λήψη ενός πακέτου δεν εγγυάται η παράδοσή του από το radio chip CC2420 στον μικροεπεξεργαστή. Γι΄ αυτό το λόγο η χρήση αυτόματης αποστολής πλαισίων αναγνώρισης από το υλικό (hardware-ACKs) μπορεί να έχει αποτέλεσμα εσφαλμένες αναγνωρίσεις.

Οι διαφορές αυτές που μόλις αναφέραμε μπορεί να φανούν συνολικά στον Πίνακα 3.1. Την θέση του Arduino στον πίνακα έχει πάρει το XBee καθώς αυτό είναι που του προσφέρει την ασύρματη δικτύωση.

	XBee	**SunSPOT**	**TelosB**	**iSense**
μέγιστο payload	100 bytes	113 bytes	114 bytes	116 bytes
διευθύνεις 16 bit	✓	✗	✓	✓
διευθύνεις 64 bit	✓	✓	✗	✓
port messaging	✗	✓	✗	✗
PAN ID	0x1–0xffff	0x1–0xffff	0x1–0xffff	0x1
κανάλι	0x0B-0x1A	0x0B-0x1A	0x0B-0x1A	0x0B-0x1A
ασυμβατότητες	κεφαλίδες MaxStream	κεφαλίδες LowPan	το auto ack είναι ανενεργό	

Πίνακας 3.1: Διαφορές στην υλοποίηση για κάθε πλατφόρμα.

3.5 Οι αλλαγές που χρειάστηκε γίνουν

Για την επίτευξη της επικοινωνίας των τεσσάρων συσκευών, έπρεπε να ξεπεράσουμε αυτές τις διαφορετικές υλοποιήσεις που παρουσιάσαμε προηγουμένως. Έτσι λοιπόν έγιναν οι ακόλουθες επιλογές:

- εύρος διευθύνσεων: 16 bit
- αναγνωριστικό προσωπικού δικτύου: 0x1
- προσθήκη 2 επιπρόσθετων byte στα πακέτα επιπέδου εφαρμογής (application layer) με σκοπό την παράκαμψη του LowPan στο SunSPOT

- προσθήκη αριθμού θύρας (port number) σε κάθε πακέτο
- ενεργοποίηση των αυτόματων μηνυμάτων αναγνώρισης στο TelosB (hardware-ACKs)

Η προσθήκη αυτών των 2 επιπρόσθετων byte είναι κομβικής σημασίας, καθώς βοηθάει στην παράκαμψη του LowPan στο SunSPOT. Παρόλα αυτά πακέτα που έχουν κεφαλίδες για το LowPan θα γίνουν δεκτά από το SunSPOT χωρίς κανένα πρόβλημα, αφού η τροποποίηση που έγινε ήταν η προσθήκη αυτού του κώδικα για την παράκαμψη και όχι η απενεργοποίησή του. Έτσι επιτρέπουμε σε υπάρχουσες εφαρμογές που χρησιμοποιούν το προηγούμενο SDK (v5) να μπορούν να εκτελεστούν κανονικά.

Η προσθήκη του αριθμού θύρας (port number) σε κάθε πακέτο έγινε κι αυτή για λόγους συμβατότητας του SunSPOT με τις άλλες συσκευές. Στο SunSPOT κάθε πακέτο που στέλνεται έχει κι 1 byte πριν από το ωφέλιμο φορτίο ώστε η επικοινωνία των εφαρμογών να γίνεται μέσω θυρών. Όπως φαίνεται και στον πίνακα 3.1 η δυνατότητα αυτή δεν υπάρχει στις άλλες πλατφόρμες. Η δομή των πακέτων σε επίπεδο εφαρμογής (application layer) φαίνεται στον Πίνακα 3.2. Οι τιμές των σταθερών είναι:**LP1** = 0x7f και **LP2** = 105 = 0x69 και

1 byte {	LP1
1 byte {	LP2
1 byte {	port No.
113 bytes {	payload

Πίνακας 3.2: Δομή πακέτου στο επίπεδο εφαρμογής.

φαίνονται στο ακόλουθο τμήμα κώδικα java:

```
1 /**
2   * Denotes the following header field is not in the scope
      of the LowPan spec
3   */
4 public static final byte DISPATCH_SPOT = DISPATCH_ESC = (
    byte)0x7f;
5 public static final byte PROTOCOL_NUMBER = 105;
6 public static final String PROTOCOL_NAME = "radiogram";
```

Κεφάλαιο 4

Αλγόριθμος για εύρεση γειτόνων σε δίκτυα αισθητήρων

4.1 Το πρωτόκολλο Echo

Το πρωτόκολλο Echo είναι μια υπηρεσία που υλοποιεί την λειτουργία της εύρεσης γειτόνων και καθιστά ικανή την συνδεσιμότητα στα δίκτυα αισθητήρων. Με αυτό το πρωτόκολλο, μια συσκευή ανακαλύπτει άλλες γειτονικές συσκευές και παρατηρεί την συνδεσιμότητά τους. Το πρωτόκολλο Echo είναι κατάλληλο για ένα δίκτυο χαρακτηριζόμενο από υψηλή κινητικότητα και μεταβλητό εύρος μετάδοσης. Η τοπολογία τέτοιων δικτύων αλλάζει ταχύτατα και με έναν απρόβλεπτο τρόπο, καθώς οι συσκευές κινούνται χωρίς να ακολουθούν κάποιο προκαθορισμένο πρότυπο. Η ονομασία του προέρχεται από την λέξη 'ηχώ', όπως ακριβώς και ο αντίλαλος που δημιουργείται όταν φωνάξουμε και υπάρχει μπροστά μας σε κάποια απόσταση κάποιο ογκώδες αντικείμενο. Ουσιαστικά πρόκειται για ένα πρωτόκολλο δρομολόγησης απλού-άλματος (single-hop routing protocol).

Το πρωτόκολλο Echo έχει σχεδιαστεί για να εφαρμοστεί σε συσκευές με περιορισμένους πόρους. Είναι ανθεκτικό, ικανό να προσαρμοστεί ταχύτατα σε συχνές και σημαντικές αλλαγές στην τοπολογία και ικανό να ξεχωρίσει τους

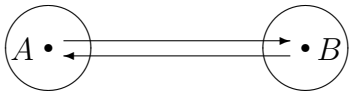
διαφορετικούς ρόλους των ανακαλυφθέντων γειτονικών κόμβων, όπως απλών κόμβων και πυλών (gateways). Επιπρόσθετα επιτρέπει παραμετροποίηση των μεταδιδόμενων μηνυμάτων και κυρίως είναι ικανό να ξεχωρίζει εάν η επικοινωνία με τις περιβάλλουσες συσκευές είναι αμφίδρομη ή μονόδρομη.

Κάθε συσκευή περιέχει δύο ταξινομημένες λίστες από αντικείμενα *κόμβων*. Ένα αντικείμενο κόμβου αναπαριστά μια συσκευή και αποτελείται από πληροφορίες που έχουν σχέση με κάθε συσκευή–κόμβο. Αυτές οι πληροφορίες είναι οι διευθύνσεις mac των συσκευών και οι χρονικές στιγμές που λήφθηκε το τελευταίο πακέτο από αυτές της συσκευές. Η μία λίστα περιέχει κόμβους με τους οποίους είναι δυνατή η αμφίδρομη επικοινωνία, και ονομάζεται *bi-neighbor* λίστα, και η άλλη λίστα περιέχει τους κόμβους με τους οποίους είναι δυνατή μονόδρομη επικοινωνία, και ονομάζεται *uni-neighbor* λίστα.

Με την μονόδρομη επικοινωνία εννοούμε ότι μια συσκευή B μπορεί να λάβει πακέτα από μια άλλη συσκευή A, ενώ η συσκευή A δεν μπορεί να λάβει πακέτα από την B (Σχήμα 4.1). Αυτό γίνεται συνήθως λόγω χαμηλής ισχύος εκπομπής της συσκευής B ή μικρής ευαισθησίας της κεραίας της συσκευής A. Αντίθετα στην αμφίδρομη επικοινωνία κάθε συσκευή A και B μπορούν να ανταλλάξουν πακέτα (Σχήμα 4.2). Με αυτή την έννοια, η λίστα *bi-neighbor* περιέχει τις συσκευές με τις οποίες, η συσκευή που αναφερόμαστε μπορεί να στείλει και να λάβει πίσω πακέτα. Αντίθετα η λίστα *uni-neighbor* περιέχει συσκευές από τις οποίες μια συσκευή μπορεί να λάβει πακέτα αλλά δεν είναι σίγουρο ότι τα πακέτα που θα στείλει θα παραδοθούν επιτυχώς. Η λίστα *bi-neighbor* είναι υποσύνολο της *uni-neighbor* λίστας, καθώς η συσκευή στην οποία αναφερόμαστε μπορεί να λάβει πακέτα από όλες τις συσκευές που είναι στην *bi-neighbor* λίστα.

Το πρωτόκολλο έχει δύο τρόπους λειτουργίας: *bi-directional* (αμφίδρομος) και *uni-directional* (μονόδρομος). Η λογική που επικρατεί και στους δύο τρόπους λειτουργίας είναι η ίδια. Κάθε συσκευή στέλνει περιοδικά σε όλες τις υπόλοιπες συσκευές του δικτύου, δηλαδή κάνει broadcast, ένα μήνυμα Echo Protocol. Αυτό το μήνυμα καλείται beacon (ραδιοφάρος) πολλές φορές. Το beacon περιέχει πληροφορίες και δεδομένα της συσκευής που το στέλνει. Η λογική του είναι ότι κάθε συσκευή, μέσω αυτού του περιοδικού μηνύματος, δηλώνει την παρουσία της στο δίκτυο, κι έτσι οι υπόλοιπες συσκευές την προσθέτουν στις γειτονικές συσκευές τους. Η αποστολή γίνεται στο port 110

Σχήμα 4.1: Μονόδρομη επικοινωνία.



Σχήμα 4.2: Αμφίδρομη επικοινωνία.

ώστε να μπορούν να ξεχωριστούν τα μηνύματα beacon από αυτά που στέλνουν οι εφαρμογές χρήστη.

Η περίοδος αποστολής των πακέτων είναι μια σταθερά που την ονομάζουμε $BEACON\_INTERVAL$. Το $BEACON\_INTERVAL$ στις δικές μας μετρήσεις το θέσαμε $500\,msec$. Ουσιαστικά το $BEACON\_INTERVAL$ καθορίζει την ταχύτητα απόκρισης του πρωτοκόλλου στις αλλαγές της τοπολογίας και τα μεγέθη του δικτύου. Όταν περάσει κάποιο χρονικό περιθώριο και η συσκευή δεν λάβει μήνυμα Echo από κάποια συσκευή που βρίσκεται στις λίστες γειτνίασής της, τότε θεωρεί ότι δεν βρίσκεται πλέον στην γειτονιά της και την αφαιρεί από την λίστα. Ο χρόνος που περιμένει μέχρι την αφαίρεση της συσκευής από την λίστα καλείται $DEADTIME$ και είναι: $DEADTIME = 3 \times BEACON\_INTERVAL + \delta$, όπου το δ είναι μια μικρή σταθερά με τιμή $50\,msec$.

Έχει παρατηρηθεί ότι σε περιπτώσεις υψηλού φόρτου, όταν το μέγεθος της γειτονιάς είναι μεγάλο και συνεπώς έχουμε πολλά beacons που λαμβάνονται, υπάρχει καθυστέρηση στην επεξεργασία αυτών των μηνυμάτων. Το αποτέλεσμα πολλές φορές είναι να λήγει το χρονικό περιθώριο $DEADTIME$ λόγω της καθυστέρησης στην επεξεργασία του μηνύματος. Αυτός είναι και ο λόγος ύπαρξης της σταθεράς δ.

4.1.1 Αμφίδρομος τρόπος λειτουργίας

Το beacon στην περίπτωση λειτουργίας με τον αμφίδρομο τρόπο περιέχει τις λίστες γειτνίασης της συσκευής. Η δομή του μηνύματος φαίνεται στον Πίνακα 4.1.

1 byte {	τύπος κόμβου	$\in \{0, 1\}$
1 byte {	μέγεθος bi-neighbor λίστας	$\in \{0 \dots 255\}$
1 byte {	αχρησιμοποιήτο	$\in \{0\}$
2 bytes {	mac διεύθυνση γείτονα 1	$\in \{0x0000 \dots 0xffff\}$
2 bytes {	mac διεύθυνση γείτονα 2	$\in \{0x0000 \dots 0xffff\}$
	$\vdots$	
2 bytes {	mac διεύθυνση γείτονα n	$\in \{0x0000 \dots 0xffff\}$

Πίνακας 4.1: Το αμφίδρομο beacon.

Η πρώτη θέση του μηνύματος περιέχει 1 byte που παριστάνει τον ρόλο της συσκευής που στέλνει το μήνυμα. Αυτό το byte μπορεί να πάρει 2 τιμές. Αν η συσκευή που στέλνει το beacon είναι πύλη για κάποιο άλλο επίπεδο (gateway), τότε παίρνει την τιμή 0. Αλλιώς αν είναι ένας απλός κόμβος παίρνει τιμή 1. Το επόμενο byte είναι το μέγεθος της *bi-neighbor* λίστας της συσκευής που στέλνει το μήνυμα. Μπορεί να πάρει τιμές από 0 έως 255. Το επόμενο byte είναι δεσμευμένο και έχει πάντα την τιμή 0. Αυτό έγινε για ευκολία καθώς στο iSense σε αυτή τη θέση του πακέτου υπάρχει το μέγεθος του Piggybacking. Στην δική μας περίπτωση αυτό το byte δεν χρησιμοποιείται ποτέ. Ύστερα ακολουθούν οι διευθύνσεις mac των γειτονικών κόμβων που υπάρχουν στην λίστα bi-neighbor της συσκευής. Καθώς χρησιμοποιούμε διευθυνσιοδότηση με 16 bit, το μέγεθος των διευθύνσεων είναι 2 bytes.

Όταν μια συσκευή A λάβει ένα beacon, από κάποια άλλη συσκευή B, την κατηγοριοποιεί ανάλογα με τον ρόλο της, το πρώτο byte του ληφθέντος μηνύματος. Στην συνέχεια ελέγχει αν η δική της διεύθυνση mac είναι στην λίστα με τις διευθύνσεις που έχει η συσκευή B. Αν αυτό ισχύει τότε η διεύθυνση mac της συσκευής B προστίθεται στην bi-neighbor λίστα της συσκευής A. Αν αυτό δεν ισχύει τότε η προστίθεται στην λίστα uni-neighbor. Σε κάθε περίπτωση μαζί με την διεύθυνση mac της συσκευής προστίθεται και η χρονική στιγμή που λήφθηκε το πακέτο.

Αν η διεύθυνση υπάρχει ήδη στην λίστα που έγινε προσπάθεια εισαγωγής τότε απλά ενημερώνεται η χρονική στιγμή της τελευταίας λήψης από την συσκευή αυτή. Αν το μήνυμα προέρχεται από gateway και είναι δυνατή η αμφίδρομη επικοινωνία με αυτό, δηλαδή η mac διεύθυνση της συσκευής A βρίσκεται στην λίστα της συσκευής B, τότε η συσκευή A θέτει σαν πύλη την συσκευή B και συνδέεται με αυτήν. Προϋπόθεση γι αυτό θα πρέπει να είναι ότι η συσκευή A δεν έχει συνδεθεί με καμία άλλη πύλη. Όταν περάσει το χρονικό περιθώριο $DEADTIME$ και η συσκευή A δεν λάβει κανένα πακέτο από την συσκευή B, τότε η εγγραφή από τη λίστα διαγράφεται και η συσκευή B δεν θεωρείται γείτονας πλέον.

Οι λίστες είναι ταξινομημένες σύμφωνα με τον χρόνο λήψης του τελευταίου πακέτου από κάθε συσκευή. Έτσι οι συσκευές οι οποίες έστειλαν beacon πρόσφατα εμφανίζονται ψηλά στην λίστα. Αντίθετα οι συσκευές που έχει περάσει κάποιος χρόνος και δεν έχουν στείλει κάποιο μήνυμα πρόσφατα εμφανίζονται τελευταίες στην λίστα. Αυτό γίνεται για να μπορεί η συσκευή να ελέγχει μόνο τις τελευταίες εγγραφές στην λίστα για το αν έχει περάσει το χρονικό περιθώριο, και να μην χρειάζεται να διατρέξει ολόκληρη την λίστα.

4.1.2 Μονόδρομος τρόπος λειτουργίας

Κατά την λειτουργία του πρωτοκόλλου στον μονόδρομο τρόπο κάνουμε την υπόθεση ότι μπορούμε να έχουμε με κάθε γειτονική συσκευή αμφίδρομη επικοινωνία, παρόλο που πολλές φορές αυτό δεν ισχύει. Το μήνυμα Echo Protocol σε αυτήν την περίπτωση περιέχει μόνο τον τύπο–ρόλο της συσκευής και τίποτε άλλο. Αυτό φαίνεται στον Πίνακα 4.2.

1 byte {	τύπος κόμβου	$\in \{0, 1\}$

Πίνακας 4.2: Το μονόδρομο beacon.

Όταν μια συσκευή A λάβει ένα beacon από μια άλλη συσκευή B, την προσθέτει απευθείας στην λίστα bi-neighbor. Αν η συσκευή B είναι gateway τότε η συσκευή A συνδέεται με την B σε περίπτωση που δεν έχει συνδεθεί σε κάποια άλλη πύλη.

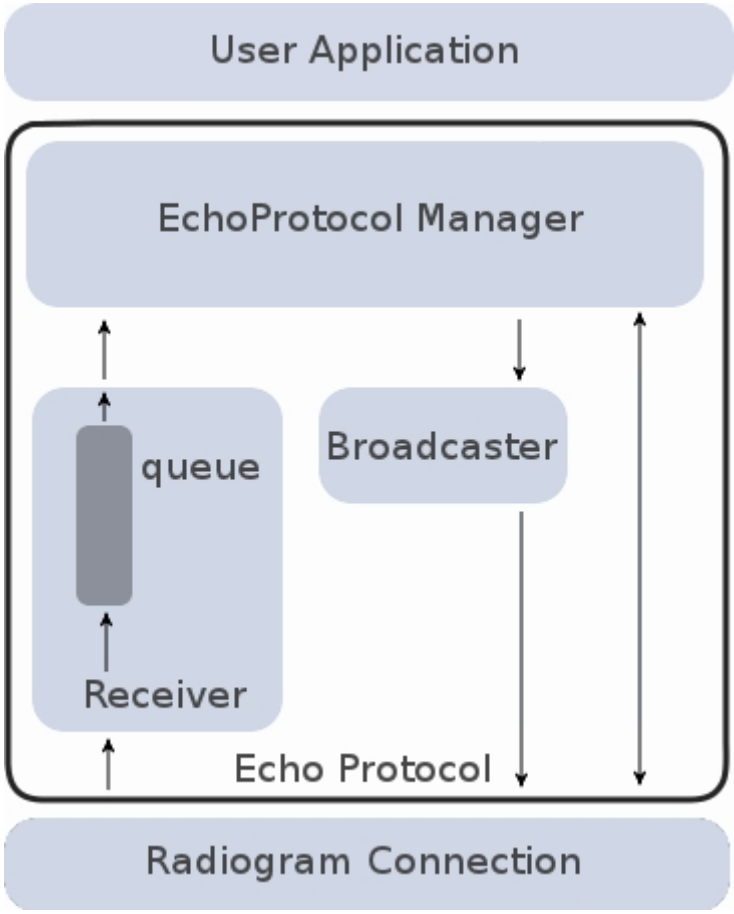
Στη συνέχεια θα δούμε τις λεπτομέρειες της υλοποίησης σε κάθε πλατφόρμα ξεχωριστά.

4.2 Υλοποίηση στο SunSPOT

Η υλοποίηση του πρωτοκόλλου στο SunSPOT αποτελείται από τρείς βασικές κλάσεις: Broadcaster, Receiver και EchoProtocolManager. Κάθε κλάση είναι αφοσιωμένη για κάποιο συγκεκριμένο σκοπό. Αυτό γίνεται χρησιμοποιώντας νήματα (threads), τα οποία μας παρέχει η J2ME, και κάθε κλάση τρέχει το δικό της νήμα. Η αποστολή και λήψη των μηνυμάτων γίνεται χρησιμοποιώντας την κλάση *radiogram*, που μας παρέχει το SunSPOT SDK. Η κλάση αυτή παρέχει ασύρματη αποστολή και λήψη πακέτων μέσω του radio chip. Η αποστολή και λήψη των πακέτων γίνεται στο port 110. Αυτό εμφανίζεται μέσα στον κώδικα με την σταθερά *ECHO_PORT* Η αρχιτεκτονική του Echo πρωτοκόλλου φαίνεται στο Σχήμα 4.3.

- **Broadcaster**: πρόκειται για την κλάση που έχει σκοπό να δημιουργεί και να στέλνει σε όλες τις γειτονικές συσκευές περιοδικά τα beacon μηνύματα στο ECHO_PORT.

- **Receiver**: η Receiver έχει σκοπό να 'ακούει' για beacons. Η υλοποίησή της έγινε με δύο νήματα. Το πρώτο νήμα είναι υπεύθυνο για κάθε beacon που λαμβάνει (αντικείμενο radiogram) να το τοποθετεί σε μια ουρά προτεραιότητας. Ένα άλλο νήμα δημιουργεί ένα αντικείμενο κόμβου (Node), συλλέγει τα δεδομένα από το beacon, ενημερώνοντας το αντικείμενο κόμβου και το στέλνει στην EchoProtocolManager η οποία είναι υπεύθυνη για την ενημέρωση των λιστών γειτνίασης. Ο λόγος ύπαρξης της ουράς είναι για να μπορεί η Receiver να λαμβάνει γρήγορα πολλά μηνύματα και σε δεύτερο χρόνο να τα επεξεργάζεται. Με αυτό τον τρόπο αποφεύγουμε την απώλεια πακέτων που μπορεί να υπήρχε αν γινόταν επεξεργασία των πακέτων αμέσως μετά την λήψη τους. Το νήμα που εισάγει τα πακέτα στην ουρά καλείται QueueConsumer και το νήμα που αφαιρεί τα πακέτα της ουράς και τα επεξεργάζεται QueueProducer.



Σχήμα 4.3: Η αρχιτεκτονική των κλάσεων του πρωτοκόλλου Echo στο SunSPOT

- **EchoProtocolManager**: η κλάση αυτή χρησιμοποιείται για την διαχείριση των λιστών γειτνίασης και την λειτουργία της δρομολόγησης. Είναι υπεύθυνη για την ενημέρωση των λιστών σύμφωνα με τα αντικείμενα κόμβου που έλαβε από την Receiver. Επίσης είναι υπεύθυνη για την διαγραφή 'πεθαμένων' κόμβων. Δηλαδή κόμβων των οποίο το χρονικό περιθώριο έληξε και δεν έγινε λήψη κάποιου beacon από αυτούς. Η δρομολόγηση γίνεται υλοποιώντας την διεπαφή του SunSPOT *IroutingManager*, με την οποία καθένας μπορεί να υλοποιήσει το δικό του πρωτόκολλο δρομολόγησης.

Στη συνέχεια θα παραθέσουμε τμήματα του κώδικα που κρίνονται σημαντικά. Η τεχνική που χρησιμοποιούμε είναι να δημιουργούμε για κάθε κλάση ένα μοναδικό αντικείμενο (singleton), έτσι ώστε να αποφεύγεται η δημιουργία πολλαπλών αντικειμένων και να μην προκληθεί κάποιο σφάλμα. Παραδείγματος χάριν, δεν θέλουμε η κλάση EchoProtocolManager, που υλοποιεί το πρωτόκολλο δρομολόγησης να έχει δύο στιγμιότυπα instances.

Πρέπει να υπενθυμίσουμε ότι παρόλο που το πρωτόκολλο Echo χρησιμοποιεί διευθύνσεις των 16 *bit*, μέσα στον κώδικα του SunSPOT οι διευθύνσεις είναι τύπου long (64 *bit*). Αυτό γίνεται για να μπορεί κάθε εφαρμογή που είναι γραμμένη για το SunSPOT SDK να είναι συμβατή και με το πρωτόκολλο Echo. Αρχικά θα αναφερθούμε στον κώδικα της κλάσης Broadcaster και θα παραθέσουμε το τμήμα της μεθόδου run() που στέλνει περιοδικά τα beacon μηνύματα.

```
1  public void run() {
2    ...
3    while (isEnabled) {
4      tmp = System.currentTimeMillis();
5      try {
6       rg.reset();
7       rg.writeByte(EchoProtocolManager.getInstance().
            getType());
8       if (EchoProtocolManager.getInstance().getEcho_mode()
            == EchoProtocolManager.BI_DIRECTIONAL) {
9        //if echo mode is Bidirectional send my neighbours
10       final long[] neighs = EchoProtocolManager.
             getInstance().getSemiNeighAddr();
11       rg.writeByte(neighs.length); //neighbours size
12       rg.writeByte(0);
13       for (int i = 0; i < neighs.length; i++) {
14         rg.writeShort((short)neighs[i]);  //write
               addresses
15       }
16      }
17      rgConnection.send(rg);
```

```
18        } catch (IOException ex) {
19          ex.printStackTrace();
20        }
21        //every 3 * BEACON_INTERVAL clean the dead neighbours
22        if (t % 3 == 0) {
23          EchoProtocolManager.getInstance().cleanDeadNeighs()
              ;
24          t = 0;
25        }
26        t++;
27        tmp = System.currentTimeMillis() - tmp;
28        if(EchoProtocolManager.BEACON_INTERVAL - tmp > 0){
29         Utils.sleep(EchoProtocolManager.BEACON_INTERVAL -
             tmp);
30        }
31      }
```

Στη συνέχεια παραθέτουμε τμήμα από το QueueConsumer νήμα της Receiver που επεξεργάζεται τα ληφθέντα πακέτα:

```
1  while (isEnabled) {
2      final Radiogram rg = (Radiogram) queue.get();
3      try {
4        final Node node = new Node();
5        node.setType((short) rg.readByte());
6        node.setMyaddress(rg.getAddress());
7        node.setExpiryTime(System.currentTimeMillis() +
           EchoProtocolManager.EXPIRATION_TIME);
8        node.setMode(EchoProtocolManager.UNI_DIRECTIONAL);
9        if (EchoProtocolManager.getInstance().getEcho_mode()
           == EchoProtocolManager.BI_DIRECTIONAL) {
10         //if echo mode is bidirectional read neighbours
             list
11         final byte neighs_num = rg.readByte();
12         rg.readByte();
13         for (byte i = 0; i < neighs_num; i++) {
```

```
14              if (my_address == rg.readShort()) {
15                node.setMode(EchoProtocolManager.BI_DIRECTIONAL
                      );
16                break;
17              }
18            }
19          }
20          //update the neighbour lists
21          EchoProtocolManager.getInstance().updateNeighs(node);
22        } catch (IOException ex) {
23          ex.printStackTrace();
24          System.out.println(IEEEAddress.toDottedHex(
                IEEEAddress.toLong(rg.getAddress())));
25        }
26    }
```

Ακολουθεί το αντίστοιχο τμήμα της QueueProducer:

```
1   while (isEnabled) {
2     try {
3         // We receive messages
4         final Radiogram rg = (Radiogram)rgConnection.
              newDatagram(rgConnection.getMaximumLength());
5         rg.reset();
6         rgConnection.receive(rg);
7         // produce messages
8         queue.put(rg);
9     } catch (IOException e) {
10        System.out.println("Nothing received");
11    }
12  }
```

Η κλάση Receiver είναι αυτή που εκκινεί τα δύο αυτά νήματα.

Στη συνέχεια θα ασχοληθούμε με την κλάση EchoProtocolManager, η οποία είναι αυτή που υλοποιεί το πρωτόκολλο δρομολόγησης. Ενδιαφέρον παρουσιάζει η μέθοδος updateNeighs(Node node) η οποία διαχειρίζεται τις λίστες γειτνίασης. Σε αυτό το τμήμα φαίνεται η διαχείριση των αντικειμένων κόμβου που

λαμβάνονται από την QueueConsumer, νήμα της Receiver:

```
1  if (node.getType() == SIMPLE_NODE) {
2    if (echo_mode == BI_DIRECTIONAL) {
3      if (!uni_neighs.insertElement(node)) {
4        System.out.println("Inserted uni-Node with address: "
             + IEEEAddress.toDottedHex(node.getMyaddress().
           longValue()));
5      }
6      //bi_neighs is subset of uni_neighs and stores neighs
         that can listen to us
7      if (node.getMode() == BI_DIRECTIONAL) {
8        if (!bi_neighs.insertElement(node)) {
9          System.out.println("Inserted Node with address: " +
               IEEEAddress.toDottedHex(node.getMyaddress().
             longValue()));
10       }
11     }
12   } else if (echo_mode == UNI_DIRECTIONAL) {
13     //we suppose that every neighbour is bidirectional so
         we update only bi_neighs list
14     if (!bi_neighs.insertElement(node)) {
15       System.out.println("Inserted Node with address: " +
           IEEEAddress.toDottedHex(node.getMyaddress().
           longValue()));
16     }
17   }
18 }
```

Παρόμοια διαχείριση υπάρχει και για την περίπτωση που ο ληφθέν κόμβος είναι πύλη (gateway). Στη συνέχεια παραθέτουμε τον κώδικα που αφαιρεί από τις λίστες τους κόμβους που έχει ξεπεραστεί το $DEADTIME$ περιθώριο.

```
1  if (uni_neighs.getFirstElement() != null) {
2    while (uni_neighs.getFirstElement().getExpiryTime() <
3        (System.currentTimeMillis() + CLEAN_DELTA)) {
4      final Node tmp = uni_neighs.getFirstElement();
```

```
5       uni_neighs.removeFirstElement();
6       if (uni_neighs.getFirstElement() == null)
7         break;
8     }
9   }
```

Παρόμοιος είναι και ο κώδικας του 'καθαρισμού' της bi-directional λίστας και του gateway. Έπειτα θα παραθέσουμε τον κώδικα που υλοποιεί την συνάρτηση getRouteInfo(long address) της διεπαφής IroutingManager, η οποία αναζητεί στους πίνακες δρομολόγησης αν υπάρχει διαδρομή για μία διεύθυνση μέσα στο δίκτυο. Στην δική μας περίπτωση οι πίνακες δρομολόγησης είναι οι λίστες γειτνίασης.

```
1  public RouteInfo getRouteInfo(long address) {
2    final RouteInfo info;
3    if (bi_neighs.containsKey(new Long(address)))
4        info = new RouteInfo(address, address, 1);
5    else
6        info = new RouteInfo(address, -1, 0);
7    return info;
8  }
```

4.3 Υλοποίηση στο TelosB

Για την υλοποίηση του πρωτοκόλλου Echo στο TelosB δημιουργήσαμε μια νέα συνιστώσα (component). Η συνιστώσα αυτή παρέχει μια διεπαφή (interface) με μεθόδους αποστολής/λήψης πακέτων και χρησιμοποιεί γι' αυτό το σκοπό άλλες διεπαφές όπως *AMPacket, AMSend, Receive, CC2420Config.* Η υλοποίηση είναι απλή, όσο απλό είναι και το TinyOS στις υπηρεσίες και τις δυνατότητες που παρέχει. Γι' αυτό το λόγο η υλοποίηση έγινε με την χρήση μόνο μιας λίστας γειτνίασης, καθώς υπάρχει περιορισμένη μνήμη. Οι μέθοδοι που παρέχει η διεπαφή στις εφαρμογές χρήστη είναι:

- EchoProtocol.init(uint8_t mode, uint8_t type): καλείται για την αρχικοποίηση της λειτουργίας του πρωτοκόλλου

- EchoProtocol.start(): καλείται για την εκκίνηση της λειτουργίας του πρωτοκόλλου
- EchoProtocol.stop(): καλείται για την διακοπή της λειτουργίας του πρωτοκόλλου
- EchoProtocol.send(am_addr_t addr, nx_uint8_t port, message_t * msg, uint8_t len): καλείται για την ασύρματη αποστολή πακέτων σε άλλες συσκευές. Η αποστολή γίνεται χρησιμοποιώντας τις λίστες γειτνίασης.
- event message_t * receive(message_t * msg, nx_uint8_t port, void * payload, uint8_t len): πρόκειται για το γεγονός (event) που σηματοδοτεί την λήψη κάποιου πακέτου από μια γειτονική συσκευή.

Στην συνέχεια θα παραθέσουμε τον κώδικα που εκτελείται περιοδικά (με χρήση timer) ώστε να γίνει broadcast το beacon σε όλες τις γειτονικές συσκευές:

```
1  if(echoMode == UNIDIRECTIONAL) {
2    btr_pkt = (TinySPOTPacket * )(call Packet.getPayload(&pkt
       ,sizeof(TinySPOTPacket)));
3    btr_pkt->port = ECHO_PORT;
4    btr_pkt->type = nodeType;
5    if(call AMSend.send(destination, &pkt,sizeof(
       TinySPOTPacket) - sizeof(nx_uint16_t)) == SUCCESS) {
6      br_busy = TRUE;
7    }
8  }
9  else {
10   //BIDIRECTIOANL
11   atomic brk_pkt = (bidirectPacket * )(call Packet.
       getPayload(&pkt, sizeof(bidirectPacket) + (listSize -
       1) * sizeof(am_addr_t)));
12   brk_pkt->port = ECHO_PORT;
13   brk_pkt->type = nodeType;
14   brk_pkt->iSense_reserved = 0;
15   neighs_list = &(brk_pkt->neighs);
```

```
16    atomic {
17        brk_pkt->size = (nx_uint8_t) listSize;
18        currentNode = head;
19        for(i = 0; i < listSize; i++) {
20            neighs_list[i] = currentNode->address;
21            currentNode = currentNode->next;
22        }
23        if(call AMSend.send(destination,&pkt,sizeof(
              bidirectPacket) + (listSize - 1) * sizeof(
              am_addr_t)) == SUCCESS) {
24            br_busy = TRUE;
25        }
26    }
27  }
28  post removeDeadNeighs();
29  atomic if(running) {
30      call Timer0.startPeriodic(TIMER_PERIOD_MILLI);
31  }
```

Στη συνέχεια παραθέτουμε τον κώδικα της μεθόδου receive η οποία πυροδοτείται όταν γίνει λήψη κάποιου πακέτου από το δίκτυο:

```
1   if(port != ECHO_PORT) {
2     //application message
3     signal EchoProtocol.receive(msg, *((nx_uint8_t * )
          payload), (void * ) &((nx_uint8_t * ) payload)[1], len
          );
4   }
5   ...
6   atomic {
7     currentNode = head;
8     while(currentNode != NULL) {
9       if(currentNode->address == call AMPacket.source(msg)) {
10        found = TRUE;
11        break;
12      }
```

```
13      currentNode = currentNode->next;
14    }
15    if(found) {
16      ... //remove node from list with adress AMPacket.source
            (msg)
17    }
18    currentNode = tail;
19    tail = (struct node * ) malloc(sizeof(nodestr));
20    tail->next = NULL;
21    tail->previous = currentNode;
22    tail->expiryTime = (call LocalTime.get()) +
        EXPIRATION_TIME;
23    tail->mode = UNIDIRECTIONAL;
24    tail->address = call AMPacket.source(msg);
25    listSize++;
26    if(currentNode != NULL) {
27      currentNode->next = tail;
28    }
29    else {
30      head = tail;
31    }
32  }
33  if(echoMode == BIDIRECTIONAL) {
34    neigh_size = ((bidirectPacket * ) payload)->size;
35    neighs = &(((bidirectPacket * ) payload)->neighs);
36    i = 0;
37    while(i < neigh_size) {
38      if(neighs[i] == TOS_NODE_ID) {
39        tail->mode = BIDIRECTIONAL;
40        break;
41      }
42      i++;
43    }
44  }
```

Κεφάλαιο 5

Πειραματική Αξιολόγηση Πρωτοκόλλου

5.1 Εισαγωγικά

Σε αυτό το κεφάλαιο θα αξιολογήσουμε την απόδοση του ετερογενούς δικτύου που μόλις παρουσιάσαμε. Παρόμοια αξιολόγηση δεν έχει γίνει ξανά στο παρελθόν, οπότε είναι πολύ σημαντικό να δούμε την ποιότητα και την απόδοση της επικοινωνίας μεταξύ αυτών των συσκευών. Θα παρουσιάσουμε μετρικές όπως ληφθέντα πακέτα ανά δευτερόλεπτο (pps), απώλεια πακέτων (packet loss), ισχύς ληφθέντος σήματος (rssi) και ποιότητα συνδέσμου (lqi). Τα πειράματα έγιναν ανάμεσα σε κάθε ζεύγος συσκευών και επαναλήφθηκαν για διαφορετικές αποστάσεις. Στο τέλος του κεφαλαίου θα γίνει μια συζήτηση και θα παρουσιάσουμε τα συμπεράσματα αυτής της αξιολόγησης.

Όπως είδαμε παρόμοια πειραματική αξιολόγηση ανάμεσα σε αυτές τις 4 πλατφόρμες δεν έχει γίνει στο παρελθόν. Υπάρχουν κάποιες εργασίες, όπως το TinySPOTComm [1], όπου δοκιμάζεται η ασύρματη επικοινωνία μεταξύ Sun SPOT [11] και TelosB [12] που χρησιμοποιεί το λειτουργικό μικρών συσκευών TinyOS [13]. Είναι πολύ σημαντικό συνεπώς, εφόσον καταστήσαμε δυνατό να θέσουμε σε λειτουργία αυτό το ετερογενές δίκτυο των 4 διαφορετικών πλατφορμών, να παρουσιάσουμε και να συγκρίνουμε την απόδοσή του.

5.2 Τοπολογία πειραμάτων

Οι μετρήσεις έγιναν ανάμεσα σε δύο ομάδες συσκευών, τις συσκευές λήψης και τις συσκευές αποστολής. Κάθε ομάδα αποτελούνταν από τις 4 διαφορετικές συσκευές: Sun SPOT [11], TelosB [12], Arduino [3] και iSense [7]. Οι μετρήσεις έγιναν σε εξωτερικό χώρο ώστε να ελαχιστοποιηθούν οι ηλεκτρονικές παρεμβολές από άλλες συσκευές, οι ανακλάσεις από τους τοίχους κτλ. Κάθε ομάδα συσκευών ήταν τοποθετημένη σε απόσταση 60 εκατοστόμετρων (cm) από το έδαφος ώστε να περιοριστούν οι ανακλάσεις του εδάφους. Η απόσταση μεταξύ των 2 ομάδων συσκευών ήταν μεταβλητή για κάθε πείραμα που πραγματοποιήσαμε και πήρε τιμές: 1 μέτρο, 3 μέτρα και 8.5 μέτρα. Η τοποθέτηση των 4 συσκευών κάθε ομάδας ήταν παρόμοια, με τρόπο ώστε κάθε συσκευή μιας ομάδας να έχει προσανατολισμό στις κεραίες της άλλης ομάδας. Η τοπολογία των πειραμάτων φαίνεται και στο Σχήμα 5.1. Η τοποθέτηση τους στον εξωτερικό χώρο φαίνεται στο Σχήμα 5.2.

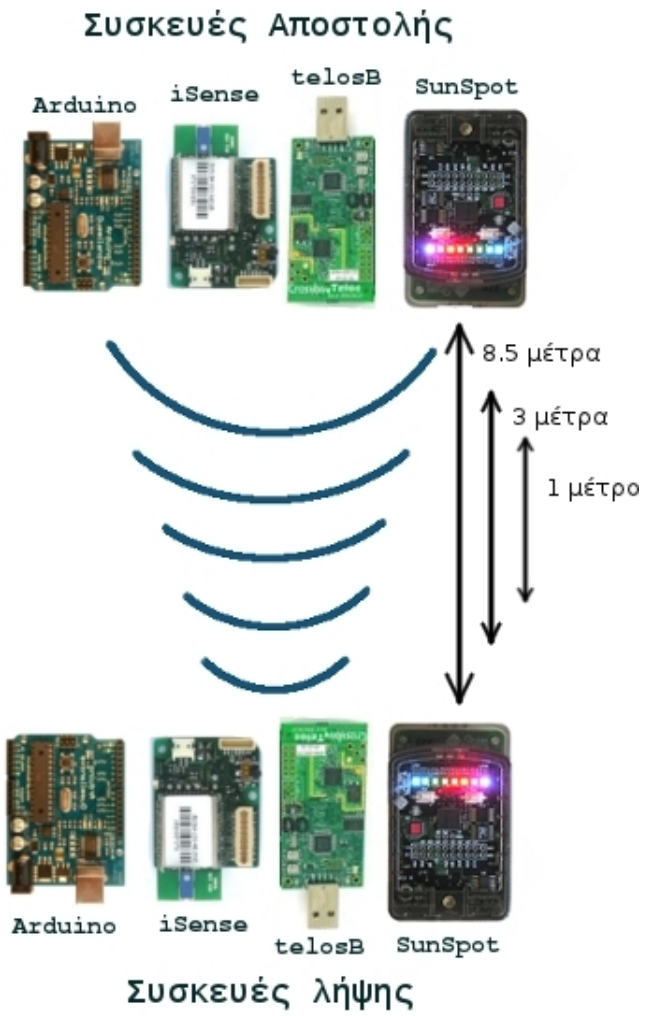
5.3 Μετρικές

Οι μετρικές που θα χρησιμοποιήσουμε για να αξιολογήσουμε την επικοινωνία των συσκευών είναι οι ακόλουθες:

- **Ληφθέντα πακέτα ανά δευτερόλεπτο** (packets per second - pps): Πρόκειται για τον αριθμό των πακέτων που έχουν δεχθεί σε ένα δευτερόλεπτο οι συσκευές λήψης. Η μετρική αυτή μπορεί να χρησιμοποιηθεί σε συνδυασμό με το μέγεθος των πακέτων και να υπολογιστεί η ικανότητα διαβίβασης δεδομένων (effective throughput) κάθε συσκευής που χρησιμοποιήσαμε.

- **Απώλεια πακέτων** (packets loss): Είναι η μετρική που καθορίζει το ποσοστό των πακέτων που αποτυγχάνουν να φθάσουν-ληφθούν σε κάποια συσκευή λήψης.

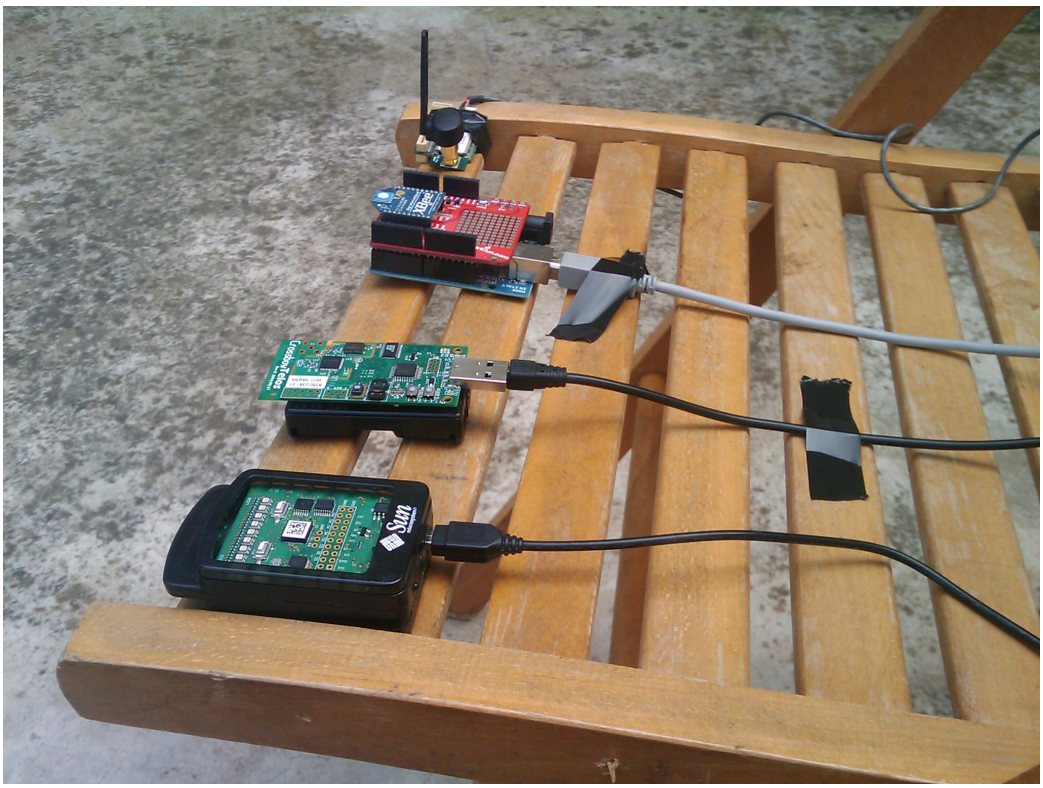
- **Δείκτης ισχύος ληφθέντος σήματος** (received signal strength indicator - rssi): Ο δείκτης ισχύος του ληφθέντος σήματος συνιστά μέτρο της ισχύος του σήματος που λαμβάνεται από στην κεραία RF του δέκτη



Σχήμα 5.1: Η τοπολογία του πειράματος. Οι συσκευές λήψης *ακούνε* για πακέτα από μια συσκευή αποστολής.

και μετράται σε dBm ("db"-milliwatts). Ο υπολογισμός του δείκτη αυτού γίνεται απ΄ ευθείας από το υλικό (hardware) της συσκευής που υπάρχει στο κύκλωμα για την ασύρματη επικοινωνία και παρέχεται στον προγραμματιστή μέσω ενός καταχωρητή. Όσο υψηλότερος είναι ο δείκτης τόσο πιο ισχυρό είναι το σήμα.

- **Δείκτης ποιότητας ζεύξης** (link quality indicator - lqi): Ο δείκτης ποιότητας ζεύξης καθορίζει το πόσο καλή είναι η ποιότητα της επικοινωνίας, εισάγοντας σε αυτόν το ποσοστό του σήματος προς τον θόρυβο. Το



Σχήμα 5.2: Τοποθέτηση των συσκευών.

πρωτόκολλο MAC IEEE 802.15.4 [19] δεν δίνει τις λεπτομέρειες για τον υπολογισμό του κι έτσι κάθε πλατφόρμα τον υπολογίζει με διαφορετικό τρόπο.

Σε κάθε επανάληψη των πειραμάτων μια συσκευή από τις συσκευές αποστολής έκανε μετάδοση πακέτων broadcast. Τα πακέτα broadcast είναι πακέτα τα οποία όταν αποσταλούν από κάποιο κόμβο γίνονται δεκτά από κάθε άλλο κόμβο του δικτύου. Έτσι λοιπόν όλες οι συσκευές της ομάδας λήψης λαμβάνουν τα πακέτα που στέλνει μια συσκευή από την ομάδα αποστολής. Κάθε συσκευή από την ομάδα συσκευών αποστολής έκανε μετάδοση 500 πακέτων broadcast, για 20 διαφορετικά μεγέθη πακέτων. Το payload (ωφέλιμοι χαρακτήρες-bytes του πακέτου) πήρε τιμές από 6 έως 96 bytes. Κάθε συσκευή λήψης για κάθε ληφθέν πακέτο κατέγραφε την τιμή rssi, την τιμή lqi και την χρονική στιγμή της λήψης (timestamp). Ακόμη κατέγραφε και τον συνολικό αριθμό των πακέτων που έλαβε. Κάθε μία από τις συσκευές αποστολής βρισκόταν σε 3 διαφορετικές

Σχήμα 5.3: Οι συσκευές λήψης.

αποστάσεις από τις συσκευές λήψης: 1, 3 και 8.5 μέτρα. Με το πέρας των πειραμάτων πήραμε τις μέσες τιμές για κάθε μέγεθος πακέτων και για κάθε απόσταση αποστολής.

5.4 Αποτελέσματα αξιολόγησης

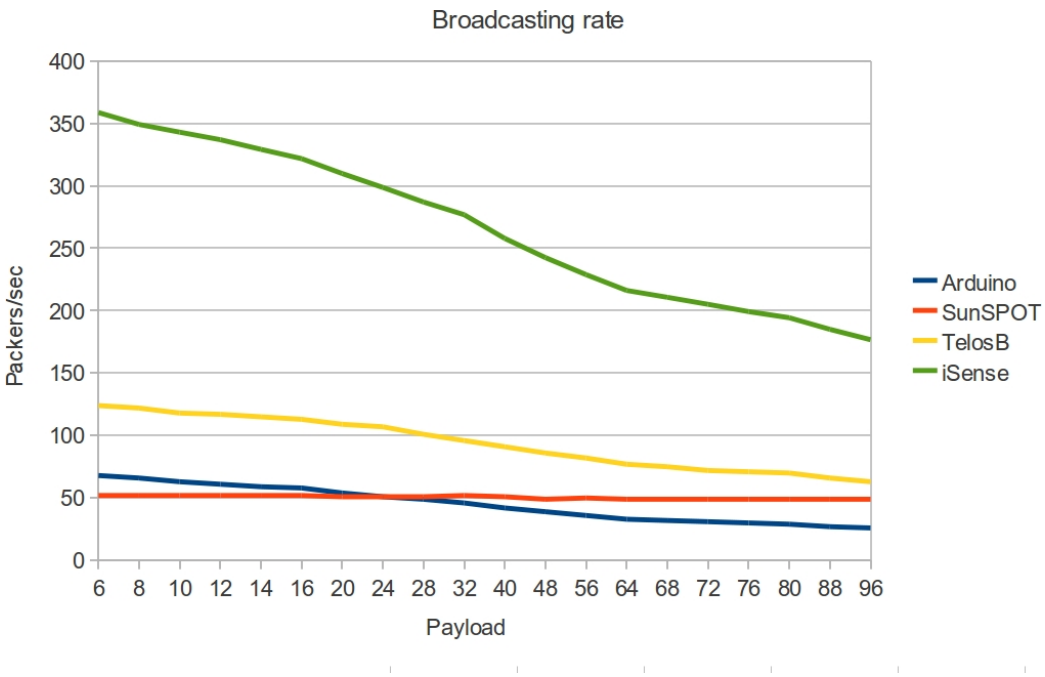
Σε αυτή την ενότητα θα παρουσιάσουμε τα αποτελέσματα της αξιολόγησης σε γραφικές παραστάσεις.

5.4.1 Αποστολή πακέτων (broadcast)

Αρχίζουμε εκθέτοντας τις μετρήσεις που πήραμε όσον αφορά την αποστολή πακέτων. Συγκεκριμένα μετρήσαμε την ταχύτητα αποστολής πακέτων (pps) κάθε συσκευής αποστολής ώστε να έχουμε ένα μέτρο σύγκρισης για την τα-

χύτητα λήψης πακέτων στην άλλη ομάδα συσκευών. Κάθε συσκευή αποστολής έστειλε συνεχόμενα 500 πακέτα για 20 διαφορετικά μεγέθη πακέτων, με το ωφέλιμο μέγεθος των πακέτων να κυμαίνεται από 6 έως 96 bytes, χωρίς να εκτελεί κάποια άλλη διεργασία ώστε να μετρήσουμε την μέγιστη ικανότητα αποστολής. Για κάθε συσκευή μετρήσαμε την συνολική χρονική περίοδο που χρειάστηκε για την αποστολή των 500 πακέτων. Επαναλάβαμε κάθε μέτρηση 9 φορές ώστε να πάρουμε καλές μέσες τιμές. Οι μέσες τιμές φαίνονται στο Σχήμα 5.4.

Θα πρέπει να αναφέρουμε ότι τα XBee modules [17] συνδέονται με το Arduino [3] χρησιμοποιώντας την σειριακή διεπαφή UART και υποστηρίζουν ρυθμούς μετάδοσης (Baud Rate) μέχρι 115200 bps (δυαδικά ψηφία ανά δευτερόλεπτο). Ξεπερνώντας όμως τον ρυθμό των 38400 bps παρατηρούμε ασταθή συμπεριφορά του Arduino σε υψηλούς ρυθμούς μετάδοσης πακέτων. Συγκεκριμένα όταν τα ληφθέντα πακέτα ανά δευτερόλεπτο ξεπεράσουν τα 150, το Arduino συνεχώς επανεκκινεί την λειτουργία του. Στις δικές μας μετρήσεις ο ρυθμός μετάδοσης ήταν 38400 bps.



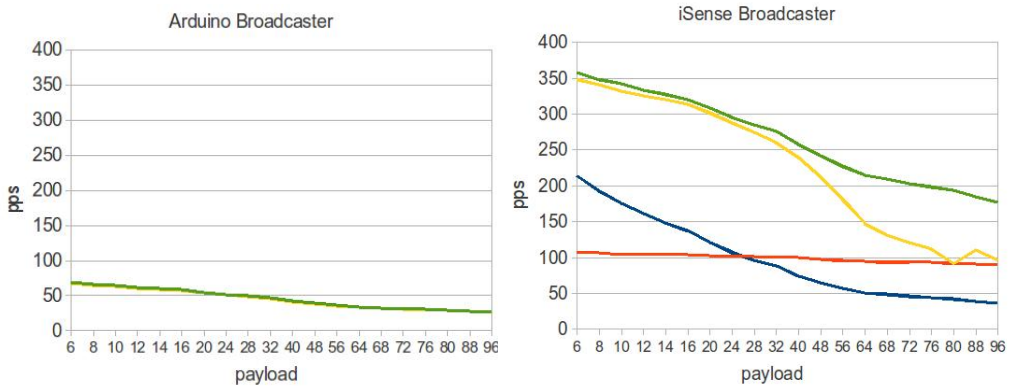
Σχήμα 5.4: Ταχύτητα αποστολής πακέτων σε πακέτα/δευτερόλεπτο(pps) συναρτήσει του ωφέλιμου μεγέθους των πακέτων σε byes

Παρατηρούμε ότι η συσκευή με τον υψηλότερο ρυθμό αποστολής είναι το iSense, ξεπερνώντας τις άλλες συσκευές κατα πολύ. Το Arduino και το SunSPOT έχουν τον χαμηλότερο ρυθμό ενώ το TelosB βρίσκετε στην μέση με σχεδόν διπλάσιο ρυθμό από το SunSPOT. Το SunSPOT παρά το ισχυρότερο hardware που διαθέτει βρίσκεται αρκετά χαμηλότερα λόγω της εικονικής μηχανής που διαθέτει, την Squawk Java Virtual Machine. Παρόλα αυτά, διαθέτει σταθερό ρυθμό ανεξαρτήτως μεγέθους των πακέτων.

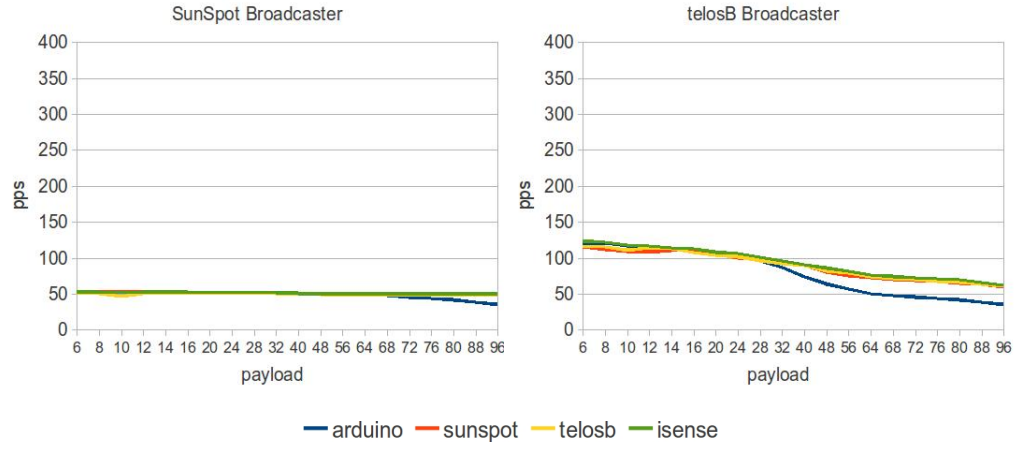
5.4.2 Λήψη πακέτων

Ο ρυθμός λήψης πακέτων σε κάποια συσκευή εξαρτάται από τον ρυθμό μετάδοσης πακέτων του αποστολέα και δεν μπορεί να τον ξεπεράσει. Βλέπουμε ότι για συσκευές με μικρό ρυθμό μετάδοσης, όπως Arduino και SunSPOT, ο ρυθμός λήψης πακέτων από αυτές τις συσκευές είναι σχεδόν παρόμοιος με τον ρυθμό αποστολής. Για συσκευές με υψηλούς ρυθμούς μετάδοσης, όπως το iSense, οι ρυθμοί λήψης πακέτων από αυτές τις συσκευές μειώνονται όσο αυξάνεται η μεταξύ τους απόσταση. Σε γενικές εκτιμήσεις μέσα από τα ακόλουθα αποτελέσματα έχουμε ότι το iSense έχει τον καλύτερο ρυθμό λήψης, ανεξάρτητα απόστασης και συσκευής αποστολής. Το Arduino έχει τους χαμηλότερους ρυθμούς λήψης, με μερικές περιπτώσεις να φθάνει ακόμη και μηδενικούς ρυθμούς. Το SunSPOT με το telosB βρίσκονται ενδιάμεσα με το SunSPOT να κατατάσσεται ελαφρώς χαμηλότερα καθώς σε κάποιες περιπτώσεις αγγίζει τους μηδενικούς ρυθμούς λήψης, όταν έχουμε μετάδοση πακέτων από το iSense.

1 μέτρο



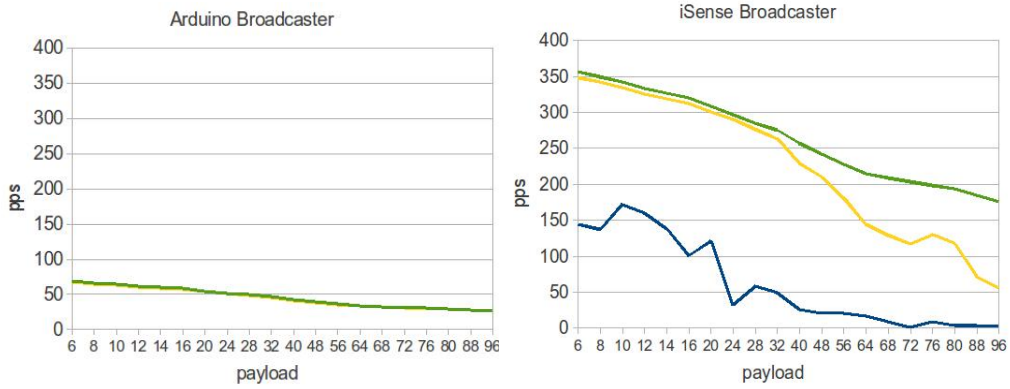
Arduino: όλες οι συσκευές έχουν τον ίδιο ρυθμό λήψης πακέτων. Όσος είναι δηλαδή και ο ρυθμός αποστολής πακέτων του Arduino.
iSense: Παρατηρούμε ότι το iSense έχει ίδιο ρυθμό λήψης με τον ρυθμό αποστολής του. Το telosB για μικρό μέγεθος πακέτων έχει παρόμοιο ρυθμό αλλά όσο το μέγεθος αυξάνεται ο ρυθμός λήψης του μειώνεται. Το SunSPOT έχει χαμηλότερο ρυθμό αλλά παραμένει σταθερός, περίπου 100 πακέτα/δευτερόλεπτο, και δεν αλλάζει σε σχέση με το μέγεθος των πακέτων. Το Arduino ξεκινάει καλύτερα από το SunSPOT αλλά όσο το μέγεθος των πακέτων αυξάνεται ο ρυθμός λήψης μειώνεται αρκετά.



SunSPOT: Βλέπουμε ότι όλες οι συσκευές έχουν τον ίδιο ρυθμό λήψης, όσος είναι και ο ρυθμός αποστολής του SunSPOT
telosB: Όλες οι συσκευές έχουν τον ίδιο ρυθμό λήψης, όσος είναι και ο ρυθμός

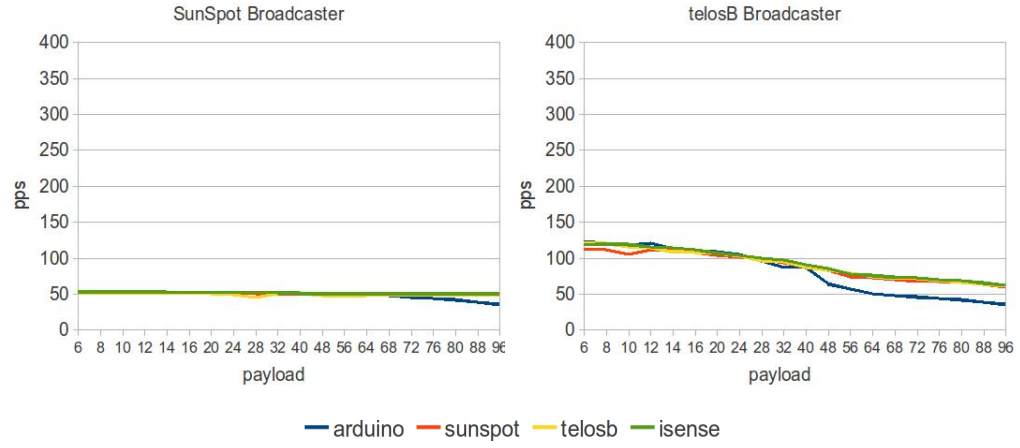
αποστολής του telosB, εκτός από το Arduino που σε μεγαλύτερο μέγεθος πακέτων μειώνει ελάχιστα τον ρυθμό του.

3 μέτρα



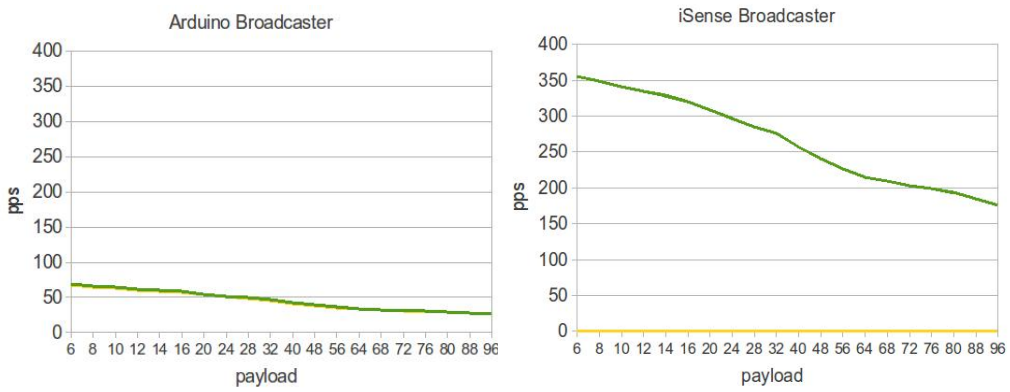
Arduino: όλες οι συσκευές έχουν τον ίδιο ρυθμό λήψης πακέτων, όσος και ο ρυθμός αποστολής πακέτων του Arduino. Δεν άλλαξε κάτι παρόλο που αυξήσαμε την απόσταση.

iSense: Παρατηρούμε ότι το iSense και το telosB έχουν ίδιους ρυθμούς λήψης με αυτούς που είχαν και στο 1 μέτρο λήψης. Το SunSPOT, παρόλο που δεν φαίνεται στο σχήμα, έχει μηδενικό ρυθμό λήψης. Το Arduino έχει μειώσει τον ρυθμό λήψης σε σχέση με το 1 μέτρο και για πακέτα με ωφέλιμο μέγεθος μεγαλύτερα των 68 bytes έχει ρυθμό λήψης σχεδόν μηδενικό.

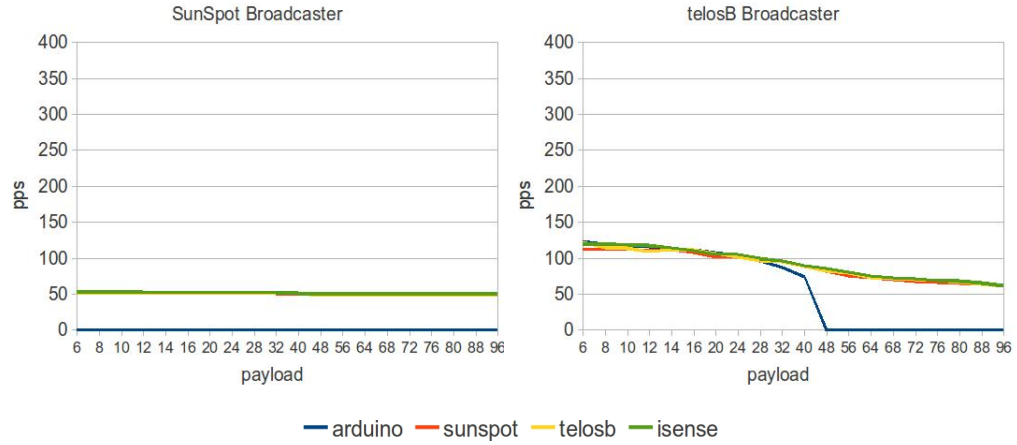


SunSPOT: Βλέπουμε ότι όλες οι συσκευές έχουν τον ίδιο ρυθμό λήψης, όσος είναι και ο ρυθμός αποστολής του SunSPOT. Καμία αλλαγή σε σχέση με το 1 μέτρο.
telosB: Παρατηρούμε ότι ο ρυθμός λήψης όλων των συσκευών είναι σχεδόν ίδιος με τον ρυθμό αποστολής του telosB, ελάχιστες διαφορές σε σχέση με το 1 μέτρο λήψης.

8.5 μέτρα



Arduino: Καμία αλλαγή σε σχέση με τις μικρότερες αποστάσεις. Όλες οι συσκευές λαμβάνουν πακέτα με ρυθμό ίσο του ρυθμού αποστολής του Arduino.
iSense: Παρατηρούμε ότι το iSense κρατάει σταθερό τον υψηλό ρυθμό λήψης του και στα 8.5 μέτρα. Παρόλα αυτά, όλες οι υπόλοιπες συσκευές έχουν μηδενικό ρυθμό λήψης. Η επικοινωνία δεν είναι εφικτή στα 8.5 μέτρα για τις υπόλοιπες συσκευές.



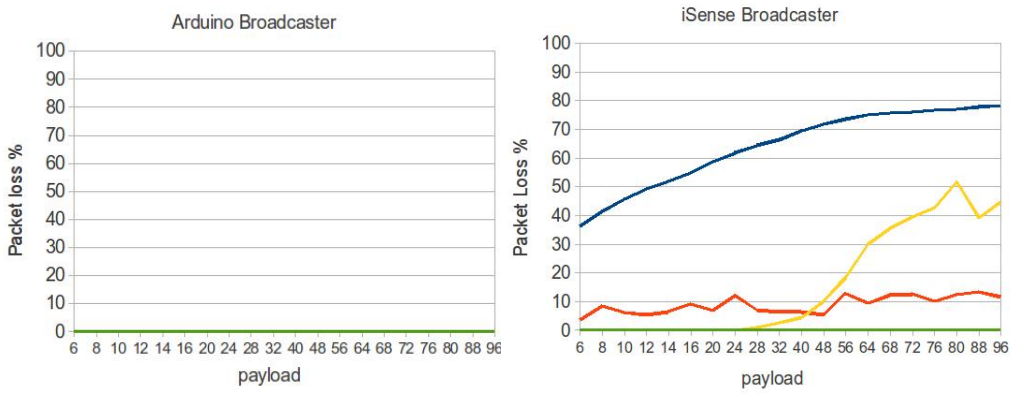
SunSPOT: Βλέπουμε ότι όλες οι συσκευές εκτός του έχουν τον ίδιο ρυθμό λήψης, όσος είναι και ο ρυθμός αποστολής του SunSPOT. Το Arduino έχει μηδενικό ρυθμό λήψης.
telosB: Παρατηρούμε ότι ο ρυθμός λήψης όλων των συσκευών είναι σχεδόν ίδιος με τον ρυθμό αποστολής του telosB για ωφέλιμο μέγεθος πακέτων μέχρι 32 bytes, για μέγεθος μεγαλύτερο αυτού το Arduino έχει μηδενικό ρυθμό λήψης ενώ οι άλλες συσκευές δεν επηρεάζονται καθόλου.

Συνοψίζοντας βλέπουμε ότι όταν είχαμε αποστολή πακέτων από το Arduino όλες οι συσκευές έκαναν λήψη πακέτων με ρυθμό ίσο του ρυθμού αποστολής του Arduino. Όταν είχαμε αποστολή πακέτων από το SunSPOT, η ο ρυθμός λήψης ήταν ίδιος για όλες τις αποστάσεις και συσκευές, εκτός του Arduino που στα 8.5 μέτρα λήψης είχε μηδενικό ρυθμό. Παρόμοια αποτελέσματα έχουμε και όταν αποστολέας ήταν το telosB, με το Arduino να μηδενίζει τον ρυθμό λήψης για πακέτα μεγαλύτερα των 32 bytes και απόσταση λήψης στα 8.5 μέτρα. Όταν είχαμε αποστολέα το iSense ο ρυθμός λήψης ήταν μειωμένος για όλες τις συσκευές και ειδικά στα 8.5 μέτρα ήταν μηδενικός. Αυτό οφείλετε εν μέρη και στο γεγονός ότι το iSense έχει πολύ υψηλό ρυθμό αποστολής, με αποτέλεσμα οι υπόλοιπες συσκευές να μην μπορούν να ακολουθήσουν αυτόν τον ρυθμό. Ακολουθούν τα διαγράμματα απώλειας πακέτων.

5.4.3 Απώλεια πακέτων

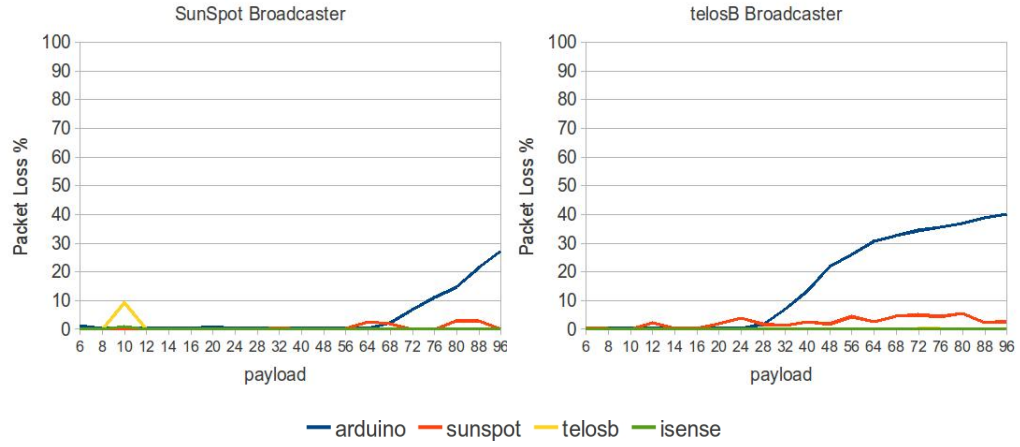
Η απώλεια πακέτων συνδέεται άμεσα με τον ρυθμό λήψης. Όταν σε μια σύνδεση έχουμε υψηλή απώλεια πακέτων αναμένεται να έχουμε και μειωμένο ρυθμό λήψης πακέτων. Έτσι περιμένουμε συσκευές οι οποίες έχουν χαμηλό ρυθμό λήψης πακέτων, όπως το Arduino, να παρουσιάζουν αυξημένη απώλεια πακέτων. Αυτό οφείλεται στην εξασθένηση του σήματος όταν αυξάνεται η απόσταση.

1 μέτρο



Arduino: Καμία συσκευή δεν έχει απώλεια πακέτων.

iSense: Παρατηρούμε αυξημένη απώλεια πακέτων από το Arduino, το οποίο για πακέτα με ωφέλιμο μέγεθος 6 bytes έχει απώλεια πακέτων περίπου 40% και για πακέτα ωφέλιμου μεγέθους 96 bytes αγγίζει απώλεια 80%. Το telosB για πακέτα ωφέλιμου μεγέθους μεγαλύτερου των 32 bytes αρχίζει να παρουσιάζει απώλεια, η οποία φθάνει μέχρι και 50%. Το SunSPOT παρουσιάζει μια απώλεια μεγέθους 10%, ενώ το iSense δεν παρουσιάζει απώλεια πακέτων.

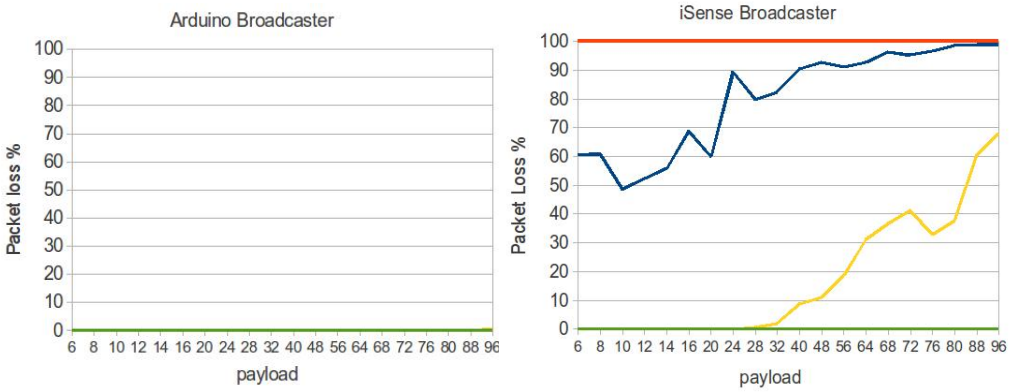


SunSPOT: Το Arduino παρουσιάζει μια απώλεια της τάξης των 30% για πακέτα με ωφέλιμο μέγεθος μεγαλύτερο των 64 bytes. Οι άλλες συσκευές παρουσιάζουν ελάχιστες απώλειες.

telosB: Παρόμοια αποτελέσματα έχουμε και σε αυτήν την περίπτωση με το Arduino να παρουσιάζει μια απώλεια της τάξης των 40% για πακέτα μεγαλύτερα των 28 bytes. Το SunSPOT παρουσιάζει μια μικρή απώλεια μεγέθους περίπου

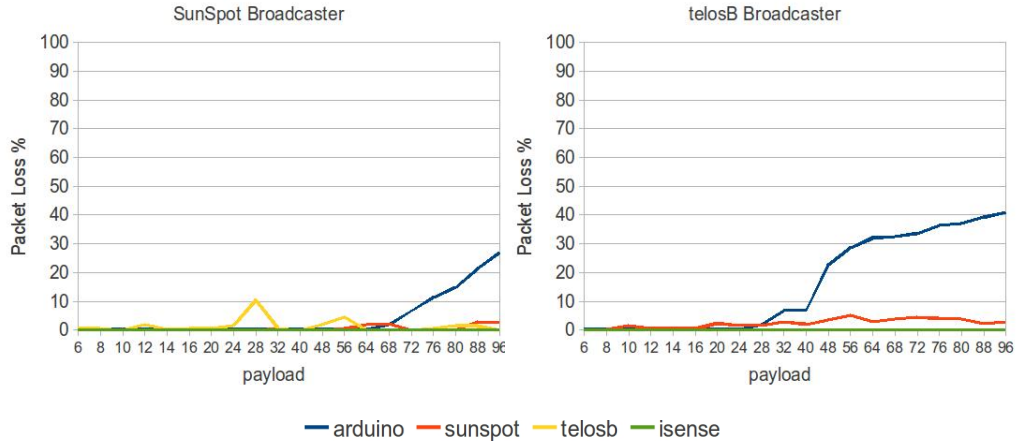
5% για πακέτα μεγαλύτερου μεγέθους.

3 μέτρα



Arduino: Όμοια και με το 1 μέτρο λήψης δεν έχουμε απώλειες σε καμία συσκευή.

iSense: Παρατηρούμε ότι το SunSPOT έχει ολική απώλεια πακέτων 100%. Το Arduino έχει απώλειες που ξεκινούν από 60% και φθάνουν μέχρι και 100%. Το telosB αρχικά παρουσιάζει μηδενικές απώλειες αλλά για πακέτα ωφέλιμου μεγέθους 32 bytes και πάνω παρουσιάζει απώλειες μέχρι και 70%. Τέλος το iSense παρουσιάζει μηδενικές απώλειες.

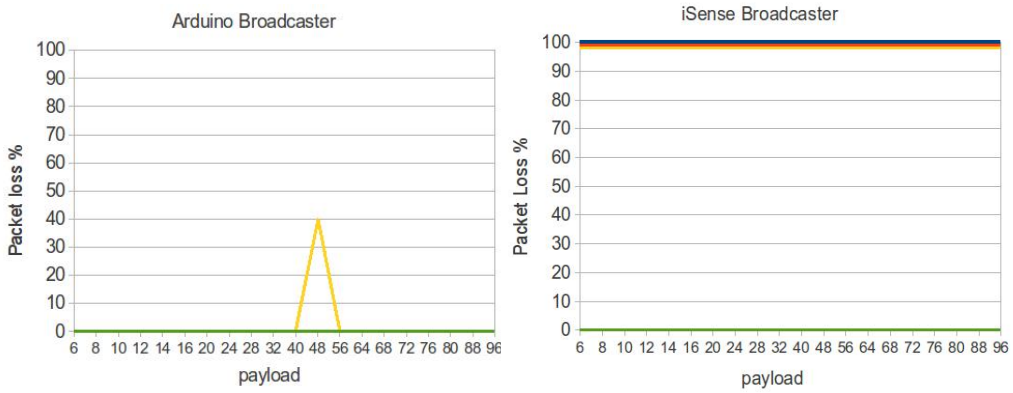


SunSPOT: Το Arduino, όμοια με το 1 μέτρο λήψης, παρουσιάζει μια απώλεια της τάξης των 30% για πακέτα με ωφέλιμο μέγεθος μεγαλύτερο των 64 bytes. Οι άλλες συσκευές παρουσιάζουν ελάχιστες απώλειες.

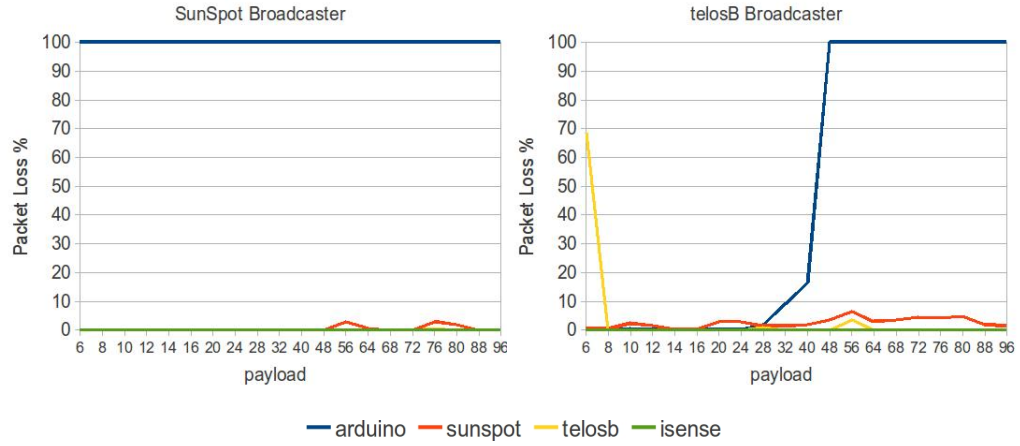
telosB: Παρόμοια αποτελέσματα έχουμε και σε αυτήν την περίπτωση με το

Arduino να παρουσιάζει μια απώλεια της τάξης των 40% για πακέτα μεγαλύτερα των 28 bytes. Το SunSPOT παρουσιάζει μια μικρή απώλεια μεγέθους περίπου 5% για πακέτα μεγαλύτερου μεγέθους.

8.5 μέτρα



Arduino: Βλέπουμε ότι όλες οι συσκευές έχουν μηδενικές απώλειες, εκτός από το telosB που παρουσιάζει μια ανωμαλία με απώλειες έως 40% για πακέτα ωφέλιμου μεγέθους 48 bytes.
iSense: Παρατηρούμε ότι όλες οι συσκευές έχουν απώλειες πακέτων 100%. Εξαιρείτε το iSense που έχει μηδενικές απώλειες.



SunSPOT: Βλέπουμε ότι όλες οι συσκευές έχουν μηδενικές απώλειες εκτός από το Arduino που έχει 100% απώλειες.
telosB: Σε αυτήν την περίπτωση βλέπουμε ένα μη σύνηθες φαινόμενο. Το telosB έχει απώλειες 70% μόνο για πακέτα μικρού μεγέθους (6 bytes). Το

Arduino για πακέτα μεγαλύτερα των 28 bytes παρουσιάζει απώλειες έως και 100%. Το SunSPOT παρουσιάζει απώλειες μικρότερες του 10%.

Συνολικά παρατηρούμε ότι το Arduino έχει τις υψηλότερες απώλειες σε σχέση με κάθε άλλη συσκευή. Αυτό κατά μεγάλο ποσοστό οφείλετε στην αργή σειριακή σύνδεσή του με το XBee module. Επίσης παρατηρούμε ολική απώλεια πακέτων και από το SunSPOT όταν γίνεται λήψη από το iSense στα 3 και 8.5 μέτρα. Το telosB παρουσιάζει μερική απώλεια πακέτων με αποστολέα το iSense στα 3 μέτρα και ολική απώλεια στα 8.5 μέτρα. Τέλος βλέπουμε ότι το iSense δεν παρουσιάζει καμία απώλεια, ανεξάρτητα απόστασης λήψης και συσκευής αποστολής.

5.4.4 Δείκτης ισχύος ληφθέντος σήματος (RSSI)

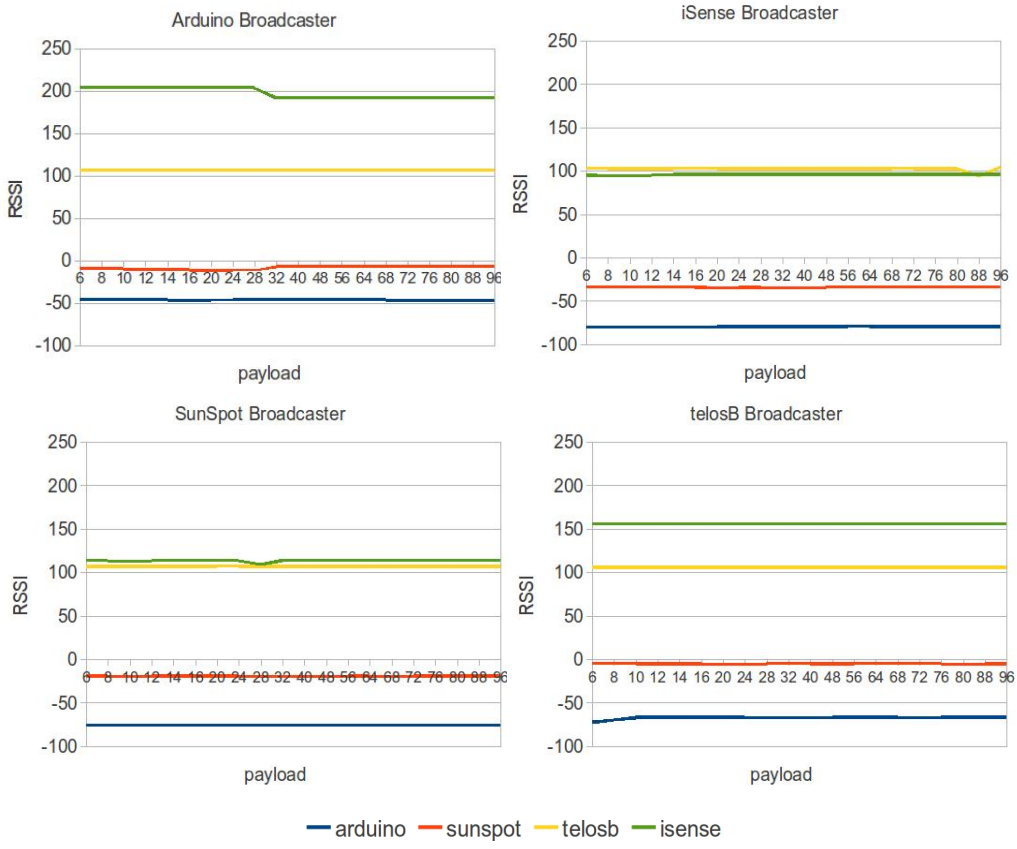
Η εκτίμηση της ποιότητας μιας ζεύξης είναι υψηλής σημασίας σχεδόν για κάθε πρωτόκολλο δρομολόγησης. Μια καλή ένδειξη για την ποιότητα της ζεύξης είναι η μετρική RSSI. Ο δείκτης ισχύος ληφθέντος σήματος – RSSI υποδεικνύει την ισχύ που είχε το τελευταίο ληφθέν πακέτο. Η ισχύς του ληφθέντος σήματος συνδέεται στενά με την ποιότητα της ζεύξης. Αυτό συμβαίνει διότι ένα ισχυρό σήμα που έφθασε στον δέκτη είναι πολύ πιθανό να έχει επηρεαστεί ελάχιστα από θόρυβο. Ο δείκτης υπολογίζεται σε κάθε συσκευή από το υλικό και παρέχεται στην εφαρμογή χρήστη μέσω ενός καταχωρητή, όπως καθορίζεται και στο IEEE 802.15.4 [19]. Οι τιμές του RSSI εξαρτώνται από την τεχνική της διαμόρφωσης (modulation format). Οι συσκευές χρησιμοποιούν όλες την ίδια τεχνική διαμόρφωσης που είναι η μη ευθυγραμμισμένη ορθογώνια μεταλλαγή μετατόπισης φάσης (offset quadrature phase-shift keying – OQPSK).

Για το chip CC2420 ο καταχωρητής που αποθηκεύει την τιμή RSSI είναι ο RSSI.RSSI_VAL. Η τιμή είναι αποθηκευμένη σε συμπλήρωμα ως προς 2 και έχει μέγεθος 8 bits. Υπολογίζεται παίρνοντας τον μέσο όρο από 8 περιόδους συμβόλου (128 μsec). Ο τύπος συσχέτισης της τιμής του καταχωρητή με την ισχύ είναι ο ακόλουθος: $P = RSSI\_VAL + RSSI\_OFFSET[dBm]$, όπου το RSSI_OFFSET υπολογίζεται εμπειρικά κατά την ανάπτυξη του συστήματος και έχει βρεθεί ότι είναι περίπου -45. Το οποίο σημαίνει ότι αν η αποθηκευμένη τιμή του καταχωρητή είναι -50 τότε η πραγματική τιμή της ισχύος θα είναι -5.

Το πρόβλημα υπάρχει στο TelosB διότι το tinyOS επιστρέφει την τιμή του καταχωρητή χωρίς να λαμβάνει υποψιών το RSSI_OFFSET και να εφαρμόζει την συνάρτηση συμπληρώματος ως προς 2. Αντίθετα η υλοποίηση του SunSPOT εφαρμόζει της παραπάνω απαιτήσεις.

Στη συνέχεια παραθέτουμε τις γραφικές παραστάσεις των τιμών RSSI για κάθε συσκευή και για κάθε απόσταση λήψης. Η ισχύς είναι άμεσα συνδεδεμένη με την απόσταση εκπομπής-λήψης.

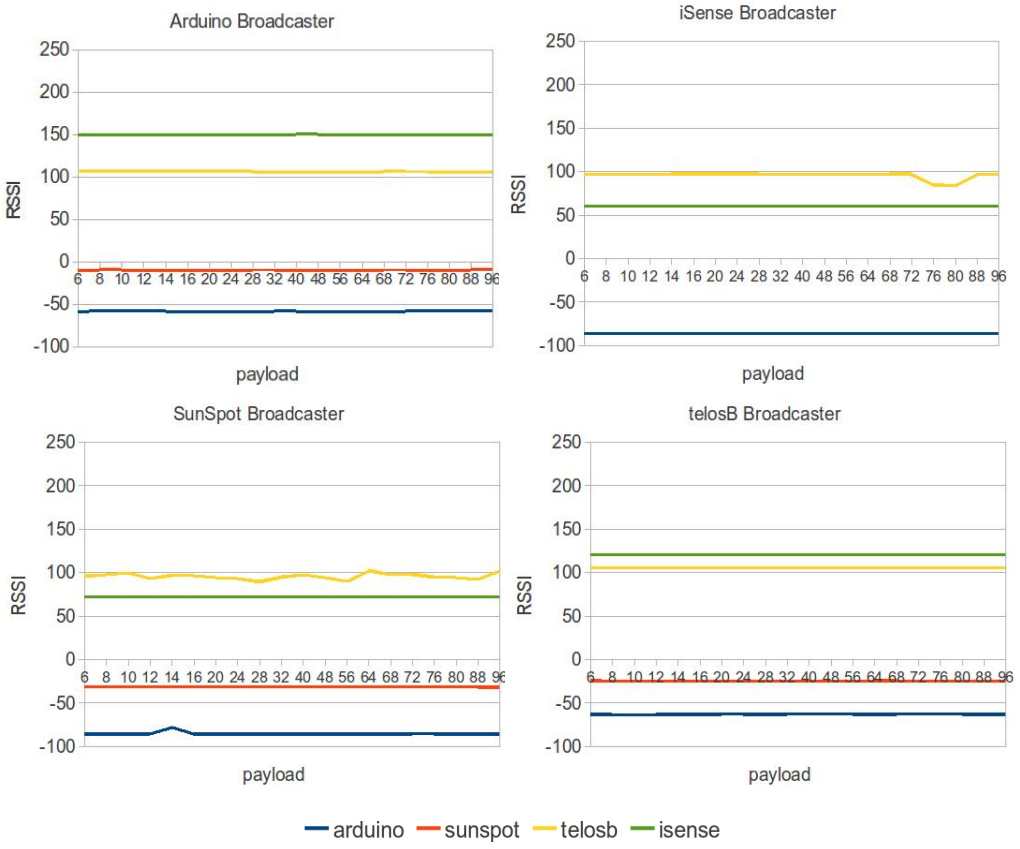
1 μέτρο



Βλέπουμε ότι το iSense μεταβάλει την τιμή του RSSI μέσα στο εύρος (100 – 200). Οι υπόλοιπες συσκευές, εκτός από κάποιες μικρές αλλαγές διατηρούν σταθερή την τιμή του RSSI, ανεξάρτητα από την συσκευή αποστολής. Βλέπουμε επίσης ότι το Arduino έχει την χαμηλότερη τιμή, ενώ το iSense την υψηλότερη. Στην μέση κυμαίνεται το SunSPOT και το TelosB. Επίσης παρατηρούμε

κάτι αναμενόμενο, το SunSPOT και το TelosB παρόλο που έχουν το ίδιο κύκλωμα αποστολής/λήψης πακέτων (chip CC2420) υπάρχει σημαντική διαφορά στις τιμές. Όπως εξηγήσαμε προηγουμένως αυτό οφείλεται στην ελλιπή υλοποίηση του tinyOS της συνάρτησης ανάκτησης της τιμής του RSSI. Μπορούμε επίσης να εξάγουμε το ίδιο συμπέρασμα όπως και στην απώλεια πακέτων. Βλέπουμε ότι το Arduino, που είχε την υψηλότερη απώλεια, να μειώνει την τιμή του RSSI κάτω από το -50 για κάθε άλλη συσκευή λήψης εκτός από την λήψη από τον εαυτό του.

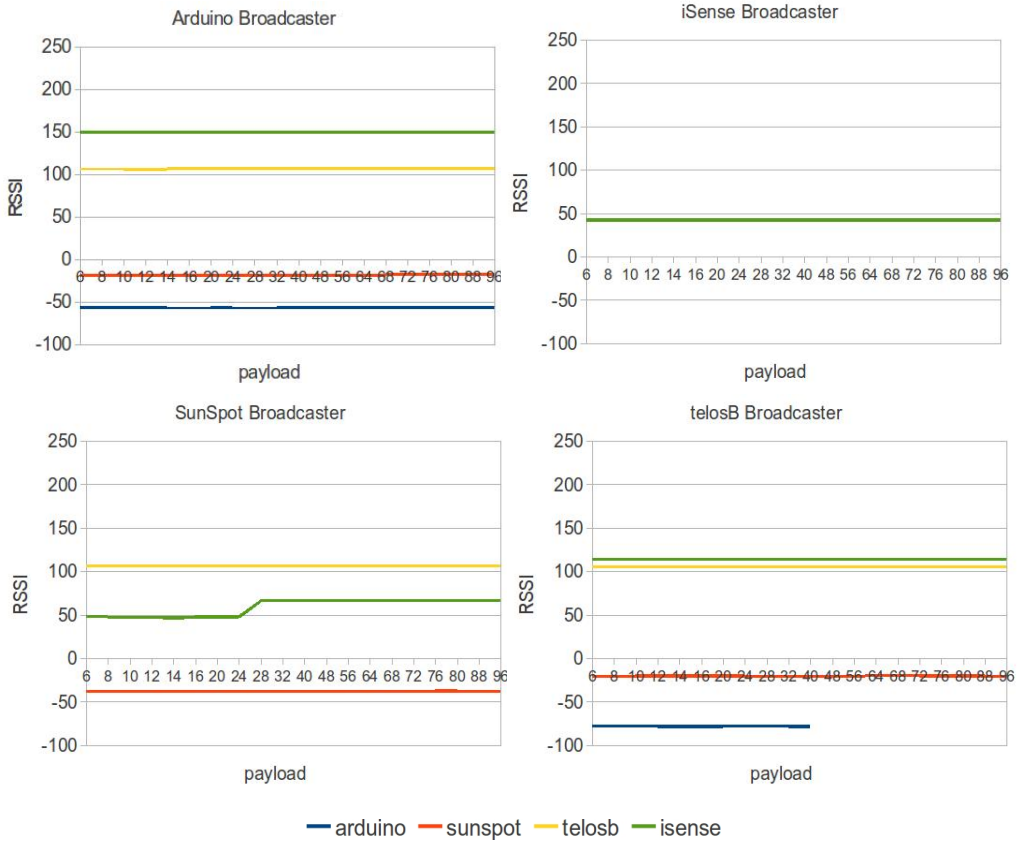
3 μέτρα



Παρατηρούμε παρόμοια αποτελέσματα όπως και στο 1 μέτρο λήψης. Εξαίρεση αποτελεί η περίπτωση που έχουμε συσκευή αποστολής το iSense και συσκευή λήψης το SunSPOT. Όπως είδαμε και στην προηγούμενη υποενότητα η απώλεια πακέτων σε αυτή την περίπτωση είναι 100% συνεπώς το SunSPOT δεν εμφα-

νίζεται στις γραφικές παραστάσεις. Επίσης βλέπουμε μια μικρή πτώση τις τιμής του RSSI για το TelosB όταν συσκευή αποστολής είναι το τεξτλατινιΣενσε σε μεγαλύτερα πακέτα, το οποίο συμβαδίζει με την απώλεια που παρουσιάζει το TelosB σε αυτή την περίπτωση. Παρόλα αυτά η απώλεια που παρουσιάζει το Arduino όταν συσκευές αποστολής είναι το iSense και το TelosB δεν αποτυπώνεται στις γραφικές παραστάσεις.

8.5 μέτρα



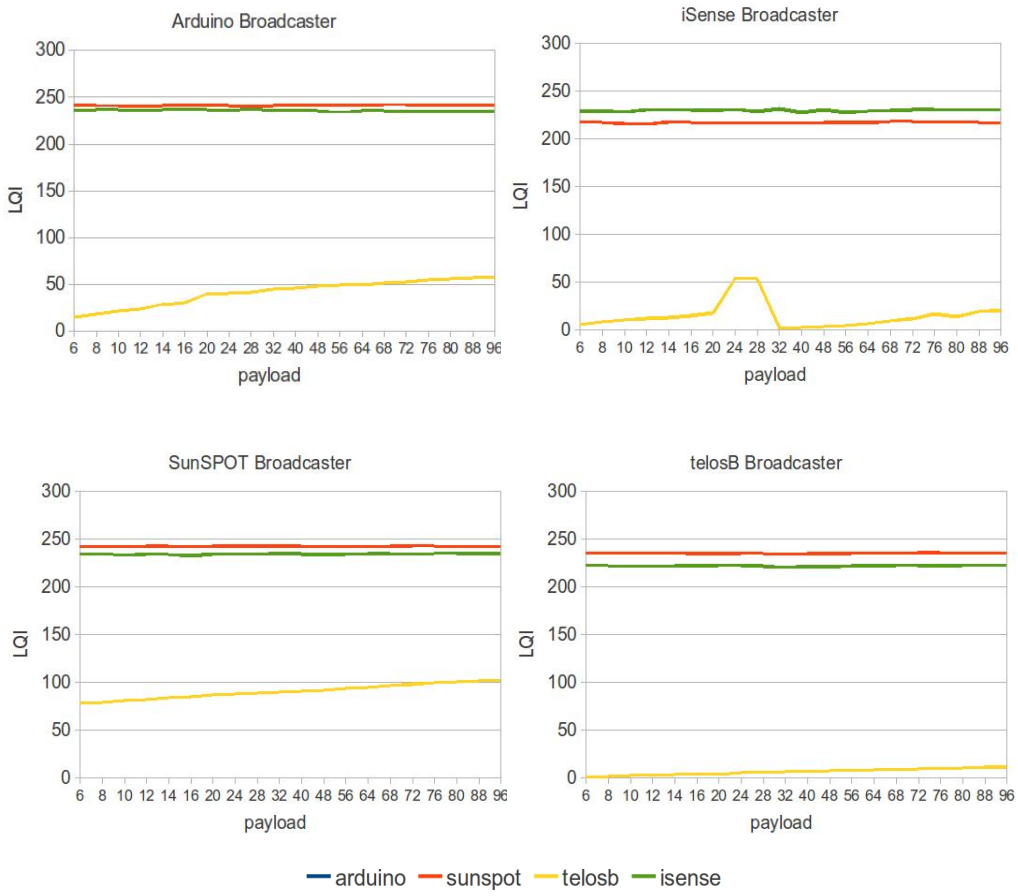
Παρατηρούμε σχεδόν παρόμοια αποτελέσματα. Αρχικά βλέπουμε να επιβεβαιώνεται η 100% απώλεια για όλες τις συσκευές, εκτός από το iSense, όταν συσκευή αποστολής είναι το iSense. Επίσης φαίνεται η 100% απώλεια που παρουσιάζει το Arduino όταν αποστολέας είναι το SunSPOT και το TelosB (για πακέτα μεγαλύτερα από 40 bytes), καθώς οι τιμές του RSSI δεν εμφανίζονται στις γραφικές.

5.4.5 Δείκτης ποιότητας ζεύξης (LQI)

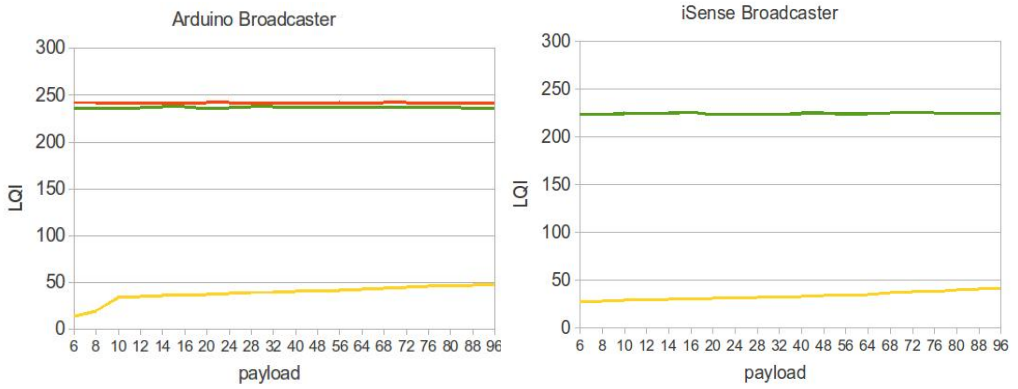
Οι πλατφόρμες που χρησιμοποιήσαμε εκτός από τον δείκτη RSSI παρέχουν και δείκτη ποιότητας ζεύξης – LQI, ο οποίος είναι ένα μέτρο του ρυθμού λαθών του κυκλώματος λήψης (chip error rate). Αυτό έγινε καθώς παρατηρήθηκε περιορισμός στις δυνατότητες του RSSI σε παλιότερες πλατφόρμες, όπως πλατφόρμες βασισμένες στο radio chip TR1000 [5] και CC1000 [9]. Έτσι οι σχεδιαστές οδηγήθηκαν στην υιοθέτηση του LQI ως μετρική της ποιότητας του ληφθέντος σήματος. Παρόλη την ευρεία υιοθέτηση του LQI, δεν έχει γίνει κάποια αξιολόγηση για το αν το LQI αποτελεί μια καλή μετρική. Η εργασία [10] ασχολείται με αυτό το θέμα και αποδεικνύει ότι το RSSI είναι μια καλή μετρική για τις σύγχρονες 802.15.4 πλατφόρμες, όπως αυτές που είναι βασισμένες στο CC2420 chip. Συγκεκριμένα αποδεικνύεται ότι όταν το RSSI είναι μεγαλύτερο από το όριο ευαισθησίας του RF δέκτη, που για το CC2420 είναι $-87\,dBm$, τότε η απώλεια πακέτων είναι μικρότερη από 15%. Το LQI από την άλλη μεταβάλλεται ευρύτερα μέσα στο χρόνο, αν όμως υπολογιστεί ο μέσος του LQI από πολλά πακέτα, τότε θα έχει καλύτερη συσχέτιση με την απώλεια πακέτων.

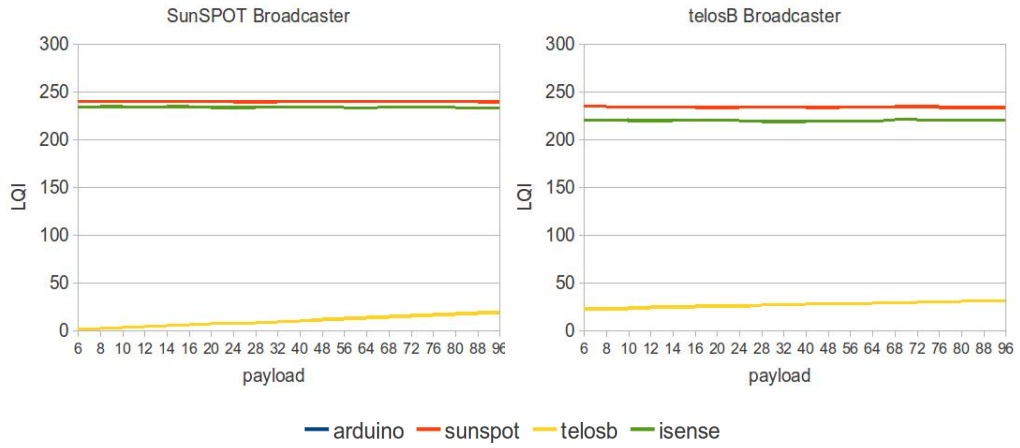
Εμείς χρησιμοποιήσαμε και τις δύο μετρικές στα πειράματά μας για να ερευνήσουμε τυχόν διαφορές που μπορεί να υπάρχουν. Το IEEE 802.15.4 [19] παρόλα αυτά δεν δίνει σαφές οδηγίες για τον υπολογισμό του LQI κι έτσι κάθε πλατφόρμα είναι ελεύθερη να ακολουθήσει τον δικό της τρόπο. Ορίζεται όμως ότι θα πρέπει να είναι ένας αριθμός μεταξύ 0 και 255. Το Arduino δεν παρέχει αυτή τη μετρική οπότε δεν φαίνεται στις γραφικές παραστάσεις που ακολουθούν.

1 μέτρο

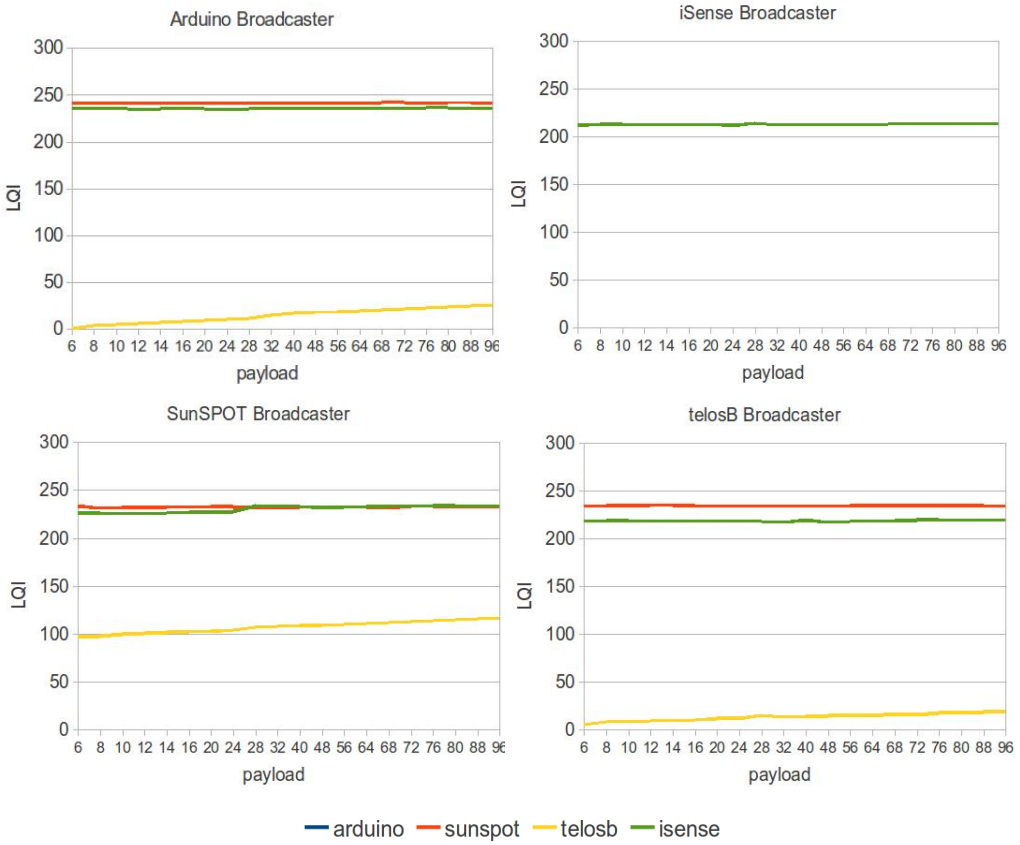


3 μέτρα





8.5 μέτρα



Παρατηρούμε ότι το iSense και το SunSPOT διατηρούν την τιμή του LQI σταθερή και σχεδόν ίδια για κάθε συσκευή αποστολής και για κάθε απόσταση

λήψης. Εξαιρείται η περίπτωση που έχουμε αποστολέα το iSense στα 3 και τα 8.5 μέτρα. Σε αυτή την περίπτωση όπως ξέρουμε το SunSPOT έχει 100% απώλειες οπότε δεν εμφανίζεται στις γραφικές παραστάσεις. Το TelosB παρουσιάζει διαφορετική συμπεριφορά. Εύκολα βλέπουμε ότι το LQI μειώνεται όταν συσκευή αποστολής είναι το iSense και απόσταση λήψης είναι το 1 μέτρο. Στην συγκεκριμένη περίπτωση γνωρίζουμε ότι το TelosB παρουσιάζει απώλεια πακέτων για μέγεθος πακέτων μεγαλύτερο από 32 bytes, το οποίο συμβαδίζει και με τον δείκτη LQI. Παρόλα αυτά γνωρίζοντας ότι το ίδιο φαινόμενο συμβαίνει και στην περίπτωση λήψης 3 μέτρων, κάτι τέτοιο δεν αποτυπώνεται στην γραφική παράσταση.

Κεφάλαιο 6

Συμπεράσματα και Προοπτικές

6.1 Συμπεράσματα

Στην παρούσα διπλωματική εργασία ασχοληθήκαμε με την υλοποίηση ενός ετερογενούς αδόμητου δικτύου. Βασικά συστατικά του δικτύου αυτού ήταν συσκευές οι οποίες ήταν εφοδιασμένες με πομποδέκτη που ακολουθεί το πρότυπο 802.15.4. Αρχικά καταφέραμε να δημιουργήσουμε τις προϋποθέσεις για την επικοινωνία αυτών των συσκευών. Έπειτα αναπτύξαμε πρωτόκολλο εύρεσης γειτόνων και δρομολόγησης απλού άλματος. Τέλος παρουσιάσαμε μια σειρά μετρήσεων για την αξιολόγηση της ικανότητας διαβίβασης δεδομένων μεταξύ όλων των συσκευών αλλά και της ποιότητας αυτής της επικοινωνίας.

Μελετώντας τα αποτελέσματα που παρουσιάσαμε παραπάνω συμπεραίνουμε ότι η συσκευή με τον υψηλότερο ρυθμό αποστολής και λήψης δεδομένων είναι το iSense. Είναι η μόνη συσκευή με 0% απώλεια πακέτων, ανεξάρτητα από την συσκευή αποστολής, την απόσταση λήψης και το μέγεθος του πακέτου.

Σε αντίθεση με το iSense, το Arduino είναι η μόνη συσκευή με απώλεια πακέτων μεγαλύτερη του 50% για μεγάλες αποστάσεις λήψης και μέγεθος πακέτου μεγαλύτερο των 50 bytes. Επιπλέον, ο ρυθμός αποστολής και λήψης του Arduino, είναι πολύ περιορισμένος. Αυτό μπορεί να εξηγηθεί από το γεγονός ότι το Arduino και το XBee module είναι συνδεδεμένα μέσω της διεπαφής UART,

σε αντίθεση με τις υπόλοιπες συσκευές που ο πομποδέκτης είναι συνδεδεμένος μέσω διεπαφής SP1.

Ο ρυθμός αποστολής πακέτων του SunSPOT είναι σταθερός. το SunSPOT εκτελεί την εικονική μηχανή της Java, η οποία αποδεικνύεται ότι βάζει όρια στην απόδοση της συσκευής. Η καθυστέρηση υπεισέρχεται λόγω της δημιουργίας και καταστροφής αντικειμένων "Java Datagram" σε 802.15.4 πλαίσια και το αντίστροφο. Αυτό αποδεικνύεται εύκολα από το γεγονός ότι το TelosB επιτυγχάνει πολύ υψηλότερους ρυθμούς, παρόλο που διαθέτει τον ίδιο πομποδέκτη (CC2420).

Παρατηρώντας τις τιμές των LQI και RSSI συμπεραίνουμε ότι το LQI δεν μπορεί να χρησιμοποιηθεί για την εξαγωγή ασφαλούς αποτελέσματος σχετικά με την απώλεια πακέτων. Αντίθετα το RSSI είναι καλύτερο σε αυτόν τον τομέα. Παρόλα αυτά οι τιμές παρουσιάζουν μειωμένη διακύμανση, το οποίο δεν μας επιτρέπει να εξάγουμε ασφαλή συμπεράσματα. Βλέπουμε επίσης για τις τιμές του RSSI στις συσκευές λήψης ότι δεν μπορούμε να βγάλουμε κάποιο σαφές συμπέρασμα για την συσχέτιση μεταξύ των τιμών. Κάθε συσκευή λήψης δίνει διαφορετική μέτρηση για πακέτα που λήφθηκαν από την ίδια συσκευή αποστολής.

6.2 Προοπτικές

Οι προοπτικές της συγκεκριμένης εργασίας περιλαμβάνουν την προσπάθεια προσθήκης επιπλέον συσκευών που βασίζονται στο πρότυπο επικοινωνίας 802.15.4. Εφόσον καταστήσαμε δυνατή την επικοινωνία μεταξύ των τεσσάρων αυτών πλατφορμών φαίνεται ότι η προσθήκη επιπλέον συσκευών στο δίκτυο, είναι υλοποιήσιμη.

Μια ακόμη προοπτική εξέλιξης του δικτύου είναι η τροποποίηση των ήδη υπαρχόντων αλγορίθμων δρομολόγησης σε κάθε συσκευή. Αυτό που χρειάζεται να γίνει είναι η προσαρμογή στις απαιτήσεις του παρόντος δικτύου (κυρίως ως προς τον τρόπο αποστολής και λήψης μηνυμάτων).

Μια ακόμη προοπτική είναι η προσθήκη του προτύπου LowPan σε καθεμία από τις συσκευές του δικτύου. Το LowPan, όπως είδαμε προσφέρει δυνατότητες δρομολόγησης, προώθησης, αλλά και κατακερματισμού των πακέτων μέσα

στο δίκτυο χρησιμοποιώντας τους ενδιάμεσους κόμβους.

Τέλος προοπτική αποτελεί η χρησιμοποίηση αυτού του δικτύου από μια πραγματική εφαρμογή ώστε να διαφανεί η πραγματική δυνατότητα που εισάγει η ετερογένεια.

Βιβλιογραφία

[1] Daniel van den Akker, Kurt Smolderen, Peter De Cleyn, Bart Braem, and Chris Blondia: "TinySPOTComm: Facilitating communication over IEEE 802.15.4 between Sun SPOTs and TinyOS-based motes"

[2] Orestis Akribopoulos, Vasileios Georgitzikis, Christos Koninis, Ioannis Papavasileiou and Ioannis Chatzigiannakis "Deployment and Evaluation of a 802.15.4 Heterogeneous Network" Διαθέσημο στον σύνδεσμο: http://ru1.cti.gr/aigaion/?page=publication&kind=single&ID=863

[3] Arduino: http://www.arduino.cc/

[4] Προδιαγραφές bluetooth από το Bluetooth Special Interest Group: https://www.bluetooth.org/Technical/Specifications/adopted.html

[5] ChipCon Inc: http://www.chipcon.com.

[6] Internet of Things: http://www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf

[7] iSense: http://www.coalesenses.com/

[8] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler: Transmission of IPv6 Packets over IEEE 802.15.4 Networks

[9] RF Monolithics: http://www.rfm.com

[10] Srinivasan, K., & Levis, P. (2006). "RSSI is Under Appreciated". Power, 2006. Citeseer. Διαθέσημο στον σύνδεσμο:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.130.3630&rep=rep1&type=pdf

[11] SunSPOT: http://www.sunspotworld.com/

[12] TelosB: http://www.xbow.com

[13] TinyOS: http://www.tinyos.net

[14] TinyOS στην wikipedia: http://en.wikipedia.org/wiki/TinyOS

[15] "Το Διαδίκτυο των Πραγμάτων": http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//TEXT+IM-PRESS+20100312STO70527+0+DOC+XML+V0//EL

[16] Γλώσσα προγραμματισμού Wiring: http://wiring.org.co/

[17] Xbee module: http://www.digi.com/

[18] Yarvis, M.; Kushalnagar, N.; Singh, H.; Rangarajan, A.; Liu, Y.; Singh, S.; Intel Res. & Dev., Hillsboro, OR, USA: "Exploiting heterogeneity in sensor networks"

[19] IEEE std. 802.15.4 - 2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs): http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf

[20] IEEE 802.15 WPAN^{TM} Task Group 4 (TG4): http://www.ieee802.org/15/pub/TG4.html