Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Υλοποίηση Ασύρματου Δικτύου Αισθητήρων ελεγχόμενης κινητικότητας σταθμού βάσης με τη τεχνολογία των αισθητήρων JAVA Sun SPOTs**
Ονοματεπώνυμο Φοιτητή	**Δήμας Ζαχαρίας**
Πατρώνυμο	**Ελευθέριος**
Αριθμός Μητρώου	**ΜΠΣΠ/ 09015**
Κατεύθυνση	**Δικτυοκεντρικά Πληροφοριακά Συστήματα**
Επιβλέπων	**Χαράλαμπος Κωνσταντόπουλος, Λέκτορας**

Ημερομηνία Παράδοσης **Οκτώβριος 2012**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Χρήστος Δουληγέρης Καθηγητής	Δημήτριος Βέργαδος Επίκουρος Καθηγητής	Χαράλαμπος Κωνσταντόπουλος Λέκτορας

ΠΕΡΙΕΧΟΜΕΝΑ

Περίληψη 4
Abstract 4
1. Εισαγωγή 5
1.1 Διαφορές μεταξύ Ad-hoc δικτύων και δικτύων σταθερής υποδομής 5
1.2 Ασύρματα Δίκτυα Αισθητήρων 6
1.3 Διαφορές μεταξύ Ad-hoc δικτύων και δικτύων αισθητήρων 6
2. Η κινητικότητα στα Ασύρματα Δίκτυα Αισθητήρων 8
2.1 Προσεγγίσεις στη κινητικότητα 8
2.2 Δίκτυα αισθητήρων πραγματικού χρόνου 10
2.2.1 Μετατόπιση κόμβου συλλογής δεδομένων 11
2.2.2 Διάχυση δεδομένων 13
2.3 Δίκτυα αισθητήρων με ανοχή στη καθυστέρηση 14
2.3.1 Συλλογή δεδομένων με απευθείας επαφή (Direct-contact data collection) 15
2.3.2 Συλλογή δεδομένων βασιζόμενη σε σημεία συνάντησης (Rendez-vous) 17
3. Η τεχνολογία των αισθητήρων JAVA Sun SPOTs 19
3.1 Η Κεντρική μονάδα του eSpot 21
3.2 Η δευτερεύουσα μονάδα eDEMO 25
4. Ο προγραμματισμός των αισθητήρων JAVA Sun SPOTs 26
4.1. Ο κύκλος ζωής ενός MIDlet 26
4.1.1. Manifest και resources 27
4.2. Η αρχιτεκτονική του συστήματος των Sun SPOTS 28
4.3. Οι βιβλιοθήκες των Sun SPOTs 30
4.3.1. Βιβλιοθήκη συσκευών 30
4.3.2. Βιβλιοθήκη αισθητήρων 31
4.3.3. Βιβλιοθήκη συνδέσεων και πρωτοκόλλων επικοινωνίας 31
4.4. Πολιτικές Δρομολόγησης 38
4.5. Τα πρωτοκόλλα δρομολόγησης των Sun SPOTs 39
4.5.1. Συνοπτική ανάλυση κλάσεων και μεθόδων των πρωτόκολλων δρομολόγησης και επικοινωνίας των Sun SPOTS 40
5. Υλοποίηση ενός δικτύου αισθητήρων ελεγχόμενης κινητικότητας σταθμού βάσης 44
5.1 Θεωρητική περιγραφή του αλγορίθμου 44
5.2 Εισαγωγή στην υλοποίηση 46
5.3 1η Φάση – Δημιουργία των συστάδων 47
5.4 2η Φάση – Επικοινωνία μεταξύ των κόμβων των συστάδων 51
5.4.1 1η Προσέγγιση 51
5.4.2 2η Προσέγγιση 52
5.5 3η Φάση - Επικοινωνία μεταξύ επικεφαλής συστάδων και κινητού κόμβου 61
5.6 Σύνοψη του αλγορίθμου 68
6. Συμπεράσματα – Περίληψη 70
Πηγές - Βιβλιογραφία 71

Περίληψη

Τα ασύρματα δίκτυα αισθητήρων είναι μια τεχνολογία που τα τελευταία χρόνια αποτελεί αντικείμενο εκτεταμένης μελέτης από την βιομηχανία και από τον ακαδημαϊκό χώρο. Οι προκλήσεις κατά το σχεδιασμό και την ανάπτυξη των δικτύων αυτών είναι πολλές, ωστόσο η ενέργεια αποτελεί το πρωταρχικό και κρίσιμο παράγοντα. Πολλοί αλγόριθμοι δρομολόγησης και τεχνικές βελτιστοποίησης της ενέργειας έχουν αναπτυχτεί για το σκοπό αυτό στα ασύρματα Ad-hoc δίκτυα, ωστόσο η μεταφορά τους στα δίκτυα αισθητήρων δεν έχει πάντοτε τα προσδοκώμενα ενεργειακά οφέλη κυρίως λόγο της ιδιαίτερης φύσης των δικτύων αυτών. Για το σκοπό αυτό, τελευταία κερδίζουν έδαφος ολοένα και περισσότερο δίκτυα αισθητήρων στα οποία όλοι οι κόμβοι τους δεν είναι στατικοί αλλά κάποιοι μπορούν να κινούνται στο δίκτυο.

Στην εργασία αυτή, αρχικά γίνεται μια ανασκόπηση των χαρακτηριστικότερων λύσεων και προσεγγίσεων που αφορούν τη κινητικότητα στα ασύρματα δίκτυα αισθητήρων. Στη συνέχεια γίνεται μια ολοκληρωμένη μελέτη της τεχνολογίας ασυρμάτων δικτύων αισθητήρων JAVA Sun SPOTs όσον αφορά το υλικό (hardware) και το λογισμικό (software) τους. Ιδιαίτερη έμφαση δίνεται στο προγραμματισμό τους και στη μελέτη των πρωτοκόλλων επικοινωνίας που χρησιμοποιούν.

Εν κατακλείδι, η μελέτη μας ολοκληρώνεται με την πρακτική υλοποίηση ενός δικτύου αισθητήρων ελεγχόμενης κινητικότητας με βάση την εργασία των [1] με χρήση αποκλειστικά της τεχνολογίας των JAVA Sun SPOTs. Μέσα από την υλοποίηση μας, δίνονται λύσεις σε μια πλειάδα πρακτικών ζητημάτων οι οποίες ως επί το πλείστον καλύπτονταν περισσότερο μόνο σε θεωρητικό επίπεδο.

Abstract

Wireless Sensor Networks are a technology which has been the subject of extensive study by industry and academia in the last years. The challenges in the design and development of these networks are many, yet energy saving remains the primary and crucial factor. Many routing algorithms and power saving techniques have been developed for this purpose in wireless Ad-hoc networks, but when it comes to wireless sensor networks they do not always have the desirable energy saving effect mainly because of the special nature of sensor networks. For this purpose, mobile solutions have been proposed for wireless sensor networks where all nodes are not static but some of them can move in the network.

In this thesis, firstly, we make a study on the most significant approaches and solutions related to mobility in wireless sensor networks. Secondly, we make a comprehensive study on the JAVA Sun SPOTs wireless sensor networks technology as far as their hardware and software is concerned. Particular emphasis is given on programming of Sun SPOTs as well as on the communication protocols they use.

Our study ends up with the design and development of a mobile controllable wireless sensor network based on the paper of [1] using exclusively the JAVA Sun SPOTs technology. Through our implementation, we provide solutions to a variety of practical issues which were mostly covered only in theory.

1. Εισαγωγή

Η ασύρματη δικτύωση είναι μια ταχύτατα αναπτυσσόμενη τεχνολογία που επιτρέπει στους χρήστες να έχουν πρόσβαση σε πληροφορίες και υπηρεσίες ηλεκτρονικά, ανεξάρτητα από τη γεωγραφική τους θέση. Τα ασύρματα δίκτυα μπορούν να είναι δίκτυα υποδομής ή δίκτυα άνευ υποδομής (Ad-hoc δίκτυα).

Ένα Ad-hoc δίκτυο είναι μια συλλογή από κινητούς κόμβους που διαμορφώνουν ένα προσωρινό δίκτυο χωρίς την βοήθεια κάποιας κεντρικής αρχής διαχείρισης ή προτυποποιημένων συσκευών υποστήριξης που είναι συχνά διαθέσιμα στα συμβατικά δίκτυα. Τα δίκτυα αυτής της μορφής είναι επίσης γνωστά και ως κινητά Ad-hoc δίκτυα (MANET - Mobile Ad-hoc NETworks).

Οι κόμβοι σε ένα κινητό Ad-hoc δίκτυο επικοινωνούν μεταξύ τους χρησιμοποιώντας ασύρματη επικοινωνία και λειτουργούν ακολουθώντας ένα μοντέλο ομότιμων (peer-to-peer) οντοτήτων. Αυτοί οι κόμβοι έχουν γενικά μια περιορισμένη ακτίνα μετάδοσης και, έτσι, κάθε κόμβος επιδιώκει τη βοήθεια των γειτονικών του κόμβων κατά την αποστολή πακέτων. Ως εκ τούτου οι κόμβοι σε ένα Ad-hoc δίκτυο μπορούν να ενεργήσουν και ως δρομολογητές και ως οικοδεσπότες, κατά συνέπεια ένας κόμβος μπορεί να προωθεί πακέτα μεταξύ άλλων κόμβων καθώς επίσης και να εκτελεί εφαρμογές χρηστών.

1.1 Διαφορές μεταξύ Ad-hoc δικτύων και δικτύων σταθερής υποδομής

Τα Ad-hoc δίκτυα παρουσιάζουν πολλές και σημαντικές διαφορές σε σχέση με τα παραδοσιακά δίκτυα σταθερής υποδομής. Οι σημαντικότερες διαφορές είναι:

- *Έλλειψη κεντρικής αρχής*:

  Στα Ad-hoc δίκτυα υπάρχει έλλειψη κεντρικής αρχής και τα δίκτυα αυτά μπορούν να δημιουργούνται οπουδήποτε χωρίς να υπάρχει κάποια κεντρική οντότητα που να καθορίζει τη ροη της πληροφορίας μεταξύ των κόμβων.

- *Μεταβλητή τοπολογία*:

  Η τοπολογία σε ένα Ad-hoc δίκτυο δεν είναι σταθερή αλλά μεταβάλλεται συνέχεια καθώς οι κόμβοι μπορούν να κινούνται συνεχώς και ελεύθερα στο χώρο, εντός και εκτός της εμβέλειας του δικτύου. Αντίθετα, σε ένα δίκτυο σταθερής υποδομής οι κόμβοι είναι σταθεροί και οι αλλαγές στη τοπολογία είναι σπάνιες.

- *Εφήμερη σύνδεση και περιορισμένη διαθεσιμότητα*:

  Σε ένα Ad-hoc δίκτυο ορισμένοι κόμβοι μπορεί να μην είναι διαθέσιμοι για κάποια περίοδο και να βρίσκονται σε κατάσταση εξοικονόμησης ενέργειας (sleep mode) για να μεγιστοποιούν τη διάρκεια «ζωής» τους στο δίκτυο.

- *Περιορισμένη ενέργεια*:

  Η βασική πηγή ενέργειας των κόμβων ενός Ad-hoc δικτύου είναι συνήθως η μπαταρία η οποία έχει πεπερασμένη ενέργεια και συγκεκριμένη διάρκεια ζωής σε αντίθεση με τους κόμβους ενός δικτύου σταθερής υποδομής οι οποίοι τροφοδοτούνται από σταθερή πηγή ρεύματος. Συνεπώς, ένα δίκτυο σταθερής υποδομής δεν υπόκειται σε ενεργειακούς περιορισμούς ενώ για ένα Ad-hoc δίκτυο η ενέργεια είναι ζωτικής σημασίας για τη λειτουργία του.

- *Κόμβοι - δρομολογητές*:

  Κάθε κόμβος σε ένα Ad-hoc δίκτυο μπορεί να λειτουργεί ως δρομολογητής προωθώντας πληροφορίες και δεδομένα. Οι κόμβοι που βρίσκονται εκτός της εμβέλειας ενός κόμβου δεν μπορούν να ανιχνευθούν και να λάβουν απ' ευθείας δεδομένα απο αυτόν παρά μονό μέσω άλλων ενδιάμεσων κόμβων που αναλαμβάνουν τη δρομολόγηση.

- *Διαμοιρασμένο φυσικό μέσο μετάδοσης*:

  Αντίθετα με τα ενσύρματα δίκτυα, κάθε κόμβος μπορεί να χρησιμοποιήσει το μέσο μετάδοσης μόνο μέσα στα όρια εμβέλειάς του.

- *Περιορισμένη επεξεργαστική ισχύ, αποθηκευτικά μέσα και λοιπές πηγές*:

  Πολλές κινητές συσκευές έχουν φθηνούς και αργούς επεξεργαστές για να έχουν χαμηλότερο κόστος. Είναι προφανές λοιπόν ότι οι κόμβοι αυτοί χρειάζονται περισσότερο χρόνο για να εκτελούν πολύπλοκους υπολογισμούς. Λόγω του μεγέθους και των περιορισμών κόστους οι περισσότερες κινητές συσκευές έχουν περιορισμένη ικανότητα αποθήκευσης.

1.2 Ασύρματα Δίκτυα Αισθητήρων

Τα ασύρματα δίκτυα αισθητήρων (Wireless Sensor Networks) [40] θεωρούνται ευρέως ως μία από τις σημαντικότερες τεχνολογίες των τελευταίων ετών και παρουσιάζουν μεγάλο ενδιαφέρον τόσο από ερευνητικής όσο και από πρακτικής άποψης. Ένα δίκτυο αισθητήρων αποτελείται συνήθως από ένα μεγάλο αριθμό χαμηλού κόστους, χαμηλής ισχύος κόμβων με αισθητήρες και δυνατότητες επεξεργασίας. Οι κόμβοι αισθητήρων επικοινωνούν μεταξύ τους σε μικρές αποστάσεις μέσω ενός ασύρματου μέσου και συνεργάζονται για να επιτευχθεί ένα κοινό αποτέλεσμα, όπως για παράδειγμα η παρακολούθηση της θερμοκρασίας του περιβάλλοντος χώρου, η επιτήρηση μιας στρατιωτικής περιοχής, ο έλεγχος μιας βιομηχανικής διαδικασίας κτλ [40].

Η βασική φιλοσοφία πίσω από τα ασύρματα δίκτυα αισθητήρων είναι ότι, ενώ η ικανότητα του κάθε κόμβου αισθητήρα είναι περιορισμένη, η αθροιστική ισχύς σε ολόκληρο το δίκτυο είναι επαρκής για την απαιτούμενη αποστολή και λήψη των δεδομένων. Σε πολλές εφαρμογές δικτύων αισθητήρων, η ανάπτυξη των κόμβων πραγματοποιείται με Ad-hoc διαδικασία χωρίς προσεκτικό σχεδιασμό. Μόλις αναπτυχθούν στο χώρο, οι κόμβοι πρέπει να είναι σε θέση να οργανωθούν αυτόνομα σε ένα ασύρματο δίκτυο επικοινωνίας.

Οι κόμβοι αισθητήρων συνήθως λειτουργούν με μπαταρίες ενώ αναμένεται να λειτουργούν χωρίς παρακολούθηση για σχετικά μεγάλα χρονικά διαστήματα κάθε φορά. Στις περισσότερες περιπτώσεις είναι πρακτικά πολύ δύσκολο έως και αδύνατο να αλλαχτούν ή να επαναφορτιστούν οι μπαταρίες τους. Τα ασύρματα δίκτυα αισθητήρων χαρακτηρίζονται από πυκνή ανάπτυξη κατά την εγκατάσταση τους στο χώρο, μεγαλύτερη αναξιοπιστία των κόμβων σε σχέση με τα παραδοσιακά Ad-hoc δίκτυα, περιορισμένη επεξεργαστική ισχύ και αρκετούς περιορισμούς μνήμης. Έτσι, τα μοναδικά χαρακτηριστικά και οι περιορισμοί των δικτύων αυτών δημιουργούν πολλές νέες προκλήσεις στο σχεδιασμό, την ανάπτυξη και την εφαρμογή των ασύρματων δικτύων αισθητήρων.

Τα περισσότερα παραδοσιακά πρωτόκολλα δρομολόγησης των Ad-hoc δικτύων [41] παρουσιάζουν πολλές ελλείψεις όταν εφαρμόζονται στα ασύρματα δίκτυα αισθητήρων, οι οποίες οφείλονται κυρίως στην ιδιαίτερη φύση των εν λόγω δικτύων να διαθέτουν περιορισμένη ενέργεια.

1.3 Διαφορές μεταξύ Ad-hoc δικτύων και δικτύων αισθητήρων

Αν και τα δίκτυα αισθητήρων αποτελούν μια ειδική κατηγορία των Ad-hoc δικτύων παρουσιάζουν ωστόσο όπως είδαμε ορισμένες σημαντικές διαφορές μεταξύ τους. Οι σημαντικότερες διαφορές συγκεντρωτικά είναι:

- Ο αριθμός των κόμβων σε ένα δίκτυο αισθητήρων είναι συνήθως αρκετά μεγαλύτερος από ότι σε ένα Ad-hoc δίκτυο.
- Οι κόμβοι στα δίκτυα αισθητήρων αναπτύσσονται πυκνά σε αντίθεση με τα Ad-hoc δίκτυα όπου οι κόμβοι είναι πιο αραιοί μεταξύ τους.
- Οι κόμβοι στα δίκτυα αισθητήρων έχουν μεγάλη πιθανότητα αποτυχίας και έτσι η εκπομπή των μηνυμάτων γίνεται συνήθως χωρίς την ύπαρξη κάποιου μηχανισμού επιβεβαίωσης.

- Η τοπολογία ενός δικτύου αισθητήρων μπορεί να μεταβάλλεται αρκετά συχνά. Παράγοντες όπως η προαναφερόμενη πιθανότητα αποτυχίας και η εξάντληση της μπαταρίας ενός ή περισσοτέρων κόμβων οδηγούν σε συνεχή αλλαγή της τοπολογίας.
- Στα Ad-hoc δίκτυα οι κόμβοι έχουν την ικανότητα να κινούνται στο χώρο, ενώ σε ένα δίκτυο αισθητήρων μόνο η δεξαμενή (sink) μπορεί να κινείται.
- Οι κόμβοι στα δίκτυα αισθητήρων έχουν μικρή επεξεργαστική ισχύ, περιορισμένες δυνατότητες για την εκτέλεση πολύπλοκων υπολογισμών και μικρή μνήμη. Αντίθετα, στα Ad-hoc δίκτυα οι κόμβοι δεν έχουν τόσο μεγάλο περιορισμό ως προς την επεξεργαστική τους ισχύ ή την μνήμη και έτσι μπορούμε να έχουμε συστήματα ισχύς αρκετών GHz και αρκετών GB μνήμης.
- Οι κόμβοι στα δίκτυα αισθητήρων πιθανόν να μην έχουν μια κοινή ταυτότητα για όλο το δίκτυο λόγω του μεγάλου αριθμού των κόμβων. Αντίθετα, στα Ad-hoc δίκτυα οι κόμβοι έχουν ένα μοναδικό αναγνωριστικό, π.χ. διεύθυνση MAC ή διεύθυνση IP.

2. Η κινητικότητα στα Ασύρματα Δίκτυα Αισθητήρων

Από τις πολυάριθμες προκλήσεις που όπως είδαμε μπορεί να αντιμετωπίσει κάποιος κατά τον σχεδιασμό και την υλοποίηση των Ασυρμάτων Δικτύων Αισθητήρων και πρωτοκόλλων επικοινωνίας, η διατήρηση της συνδεσιμότητας και η μεγιστοποίηση της διάρκειας ζωής του δικτύου είναι οι πλέον κρίσιμοι παράγοντες που πρέπει πάντοτε να λαμβάνονται υπόψη [4]. Η συνδεσιμότητα συνήθως επιτυγχάνεται με τους εξής τρόπους:

- ανάπτυξη επαρκούς αριθμού αισθητήρων στο χώρο έτσι ώστε και οι απομονωμένοι κόμβοι να έχουν συνδεσιμότητα με το υπόλοιπο δίκτυο
- χρήση εξειδικευμένων κόμβων με μεγάλη εμβέλεια επικοινωνίας που διατηρούν τη συνδεσιμότητα και με τους πιο απομακρυσμένους κόμβους του δικτύου.

Ο δεύτερος παράγοντας, η διάρκεια ζωής του δικτύου, συνδέεται άμεσα με το πόσο καιρό θα διαρκέσουν οι πηγές ενέργειας των κόμβων. Η διάρκεια ζωής του δικτύου συνήθως μπορεί να αυξηθεί με μεθόδους διατήρησης της ενέργειας όπως:

- χρήση ενεργειακά αποδοτικών πρωτοκόλλων και αλγορίθμων επικοινωνίας
- τεχνικές αναπλήρωσης της μπαταρίας.

Μια άλλη εναλλακτική προσέγγιση που κερδίζει όλο και περισσότερο έδαφος τα τελευταία χρόνια που εστιάζει τόσο στη συνδεσιμότητα αλλά και στη διάρκεια ζωής του δικτύου είναι η χρήση της κινητικότητας στα δίκτυα αισθητήρων. Κινητές συσκευές μπορούν να χρησιμοποιηθούν ως μια ορθογώνια μέθοδος για την αντιμετώπιση της συνδεσιμότητας και τα προβλήματα ζωής στα ασύρματα δίκτυα αισθητήρων [34] ως εξής:

- Συνδεσιμότητα

  Οι κινητές συσκευές μπορούν να χρησιμοποιηθούν για τη μεταφορά πληροφοριών μεταξύ των μεμονωμένων και απομακρυσμένων κόμβων του δικτύου.

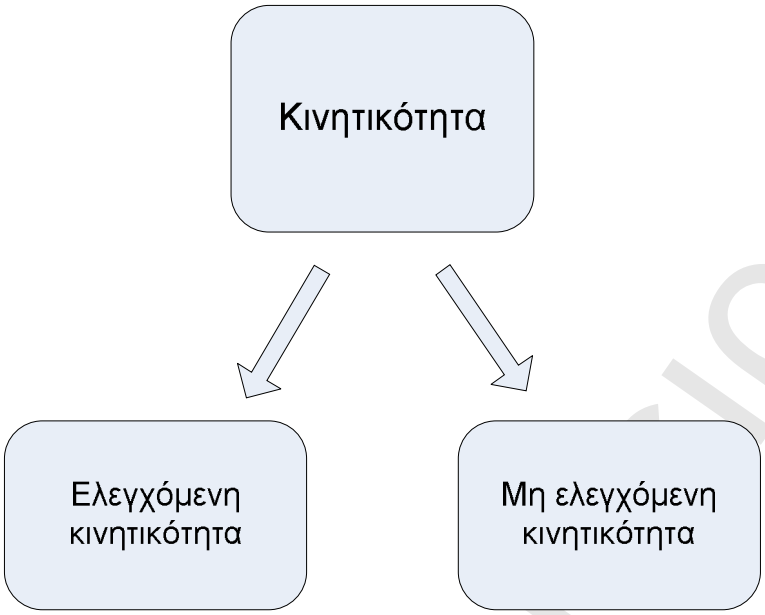
- Ενεργειακή απόδοση

  Οι πληροφορίες που μεταφέρονται μέσω των κινητών συσκευών μπορούν να μειώσουν την κατανάλωση ενέργειας των κόμβων αισθητήρων με τη μείωση του αριθμού των hops κατά την αποστολή και λήψη δεδομένων μεταξύ των κόμβων.

2.1 Προσεγγίσεις στη κινητικότητα

Σε σχέση με το είδος της κίνησης των συσκευών, μπορούμε να διακρίνουμε δυο γενικές κατηγορίες κινητικότητας [5]:

A. Ελεγχόμενη κινητικότητα
B. Μη ελεγχόμενη κινητικότητα



Εικόνα 2.1. Τα είδη της κινητικότητας

Στη πρώτη περίπτωση η κινητικότητα επιτυγχάνεται με την προσάρτηση ενός κόμβου συλλογής δεδομένων σε κάποια κινούμενη ζωντανή οντότητα, όπως π.χ. ένα ζώο ή σε μια τεχνητή οντότητα όπως κάποιο όχημα (π.χ. λεωφορείο, συρμός τρένου κτλ) η οποία υπάρχει ήδη στο περιβάλλον εγκατάστασης των αισθητήρων και είναι εκτός του ελέγχου του δικτύου. Στη δεύτερη περίπτωση η κινητικότητα επιτυγχάνεται με την σκόπιμη προσθήκη μιας κινητής οντότητας όπως π.χ. ένα κινούμενο ρομπότ ή κάποιο άλλο κινούμενο όχημα στο δίκτυο στο όποιο έχει ενσωματωθεί ένας κόμβος συλλογής δεδομένων. Στην περίπτωση αυτή, η κινητή οντότητα αποτελεί αναπόσπαστο μέρος του ίδιου του δικτύου αισθητήρων και ως εκ τούτου μπορεί να είναι πλήρως ελεγχόμενη.

Τα τελευταία χρόνια έχουν γίνει σημαντικές προτάσεις και προσεγγίσεις για την αξιοποίηση της κινητικότητας στα ασύρματα δίκτυα αισθητήρων με σκοπό τη συλλογή δεδομένων. Αυτές οι προσεγγίσεις μπορούν να κατηγοριοποιηθούν με βάση τις ιδιότητες των κόμβων συλλογής δεδομένων (sink mobility) καθώς επίσης και ως προς τις μεθόδους ασύρματης επικοινωνίας που χρησιμοποιούνται για τη μεταφορά των δεδομένων. Έτσι, μπορούμε να διακρίνουμε τις παρακάτω λύσεις στο ζήτημα της κινητικότητας [4]:

- Λύσεις που βασίζονται σε κινητούς σταθμούς βάσης (Mobile Base Station)

  Ο κινητός σταθμός βάσης είναι ένα κινητός κόμβος συλλογής δεδομένων (sink) που αλλάζει θέση στο δίκτυο κατά τη διάρκεια της λειτουργίας του. Τα δεδομένα που παράγονται από τους αισθητήρες αναμεταδίδονται στο σταθμό βάσης χωρίς να γίνεται μακροπρόθεσμη αποθήκευση στη μνήμη τους.

- Λύσεις που βασίζονται σε κινητούς συλλέκτες δεδομένων (Mobile data collectors)

  Ο κινητός συλλέκτης δεδομένων είναι επίσης ένα κινητός κόμβος συλλογής δεδομένων (sink) που επισκέπτεται τους αισθητήρες του δικτύου. Τα δεδομένα αποθηκεύονται στους αισθητήρες-πηγή μέχρι ο κινητός συλλέκτης δεδομένων επισκεφτεί τους αισθητήρες και αρχίσει να λαμβάνει τα δεδομένα τους απευθείας σε ένα μόνο hop.

- Λύσεις που βασίζονται σε σημεία συνάντησης (Rendez-vous)

  Οι λύσεις που βασίζονται σε σημεία συνάντησης είναι υβριδικές λύσεις όπου τα δεδομένα από τους αισθητήρες αποστέλλονται σε συγκεκριμένους κόμβους - σημεία συνάντησης που βρίσκονται κοντά στη διαδρομή των κινητών συσκευών. Τα δεδομένα των αισθητήρων αποθηκεύονται προσωρινά στα σημεία συνάντησης μέχρις ότου να παραληφτούν από τις κινητές συσκευές.

Οι παραπάνω προσεγγίσεις έρχονται για να δώσουν λύση στα ενεργειακά ζητήματα και στα προβλήματα συνδεσιμότητας ανάλογα με τη φύση και τη σκοπιμότητα του εκάστοτε δικτύου αισθητήρων. Συνεπώς, είναι σκόπιμο πριν γίνει οποιαδήποτε περαιτέρω ανάλυση των παραπάνω προσεγγίσεων να διακρίνουμε πρώτα τις δυο γενικές κατηγορίες των ασυρμάτων δικτύων και στη συνέχεια να αναφέρουμε ποιες προσεγγίσεις ενδείκνυνται για τη κάθε κατηγορία:

A. Δίκτυα αισθητήρων πραγματικού χρόνου (Real-time Sensor Networks).

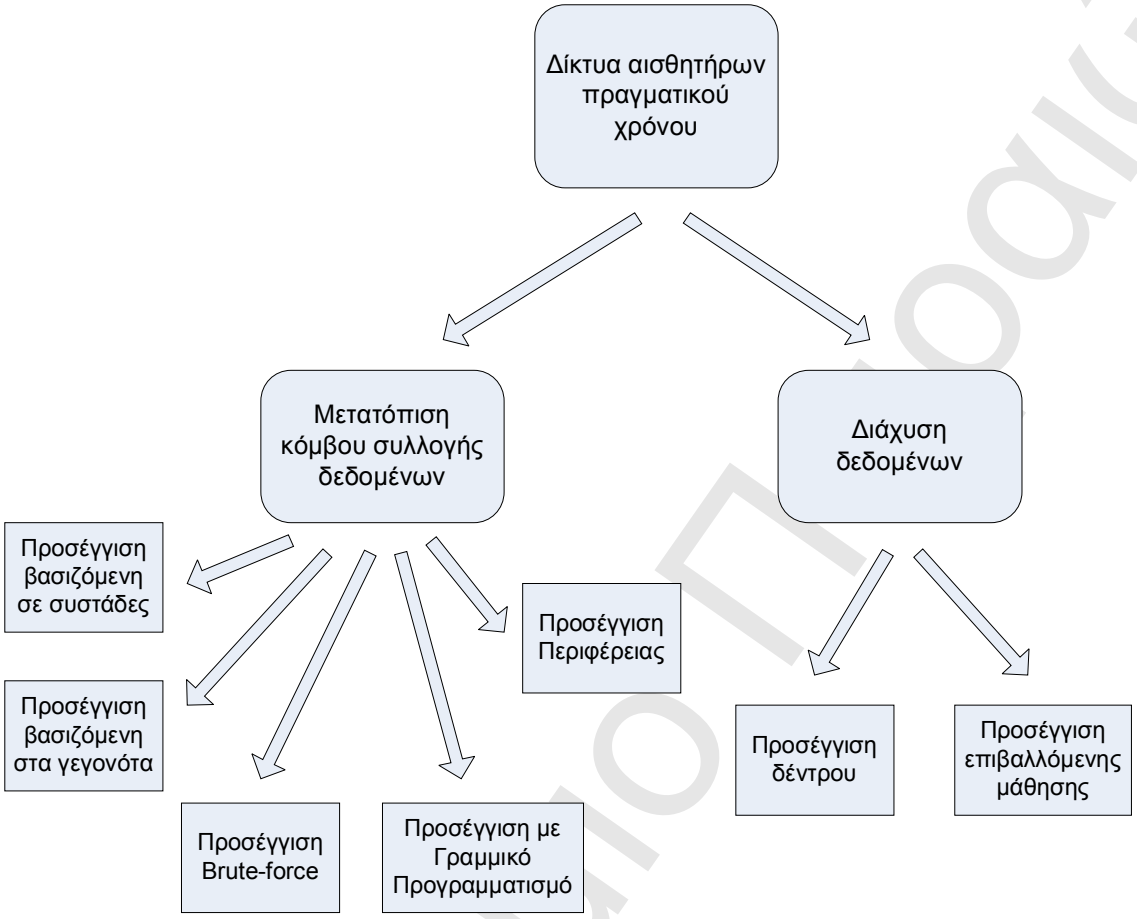
   Στα δίκτυα αυτά πρέπει τα δεδομένα των αισθητήρων να συλλέγονται άμεσα από τους κόμβους συλλογής δεδομένων ενώ οποιεσδήποτε καθυστερήσεις πρέπει να είναι χρονικά ελαχιστοποιημένες. Με διάφορες τεχνικές διάχυσης δεδομένων μεταξύ του κινητού σταθμού βάσης και των αισθητήρων η διάρκεια ζωής του δικτύου μπορεί να αυξηθεί με τη κατάλληλη μετατόπιση (όποτε θεωρείται σκόπιμο) του κινητού σταθμού βάσης σε σημεία στο δίκτυο όπου θα επιτυγχάνεται το μεγαλύτερο ενεργειακό όφελος.

B. Δίκτυα αισθητήρων με ανοχή στη καθυστέρηση (Delay-tolerant Sensor Networks).

   Στα δίκτυα αυτά, σε αντίθεση με τα δίκτυα πραγματικού χρόνου, τα δεδομένα των αισθητήρων δεν είναι απαραίτητο να συλλέγονται άμεσα από τους κόμβους συλλογής. Δυνητικά, μια μεγαλύτερη ανοχή στη καθυστέρηση οδηγεί και σε περεταίρω εξοικονόμηση της ενέργειας.

2.2 Δίκτυα αισθητήρων πραγματικού χρόνου

Στη κατηγορία αυτή των δικτύων αισθητήρων βρίσκουμε τις λύσεις που βασίζονται σε κινητούς σταθμούς βάσης. Παρακάτω παρουσιάζουμε τις χαρακτηριστικότερες προσεγγίσεις που ενδείκνυνται για τη κατηγορία των δικτύων αισθητήρων πραγματικού χρόνου.



Εικόνα 2.2. Οι προσεγγίσεις στα Δίκτυα Αισθητήρων πραγματικού χρόνου

2.2.1 Μετατόπιση κόμβου συλλογής δεδομένων

Ένας κόμβος συλλογής δεδομένων κινείται προς τους αισθητήρες έτσι ώστε να μειωθεί το μήκος της διαδρομής για τη λήψη των δεδομένων και να εξισορροπηθεί η κατανάλωση ενέργειας μεταξύ των κόμβων του δικτύου. Κατά τη διάρκεια της κίνησης ο κόμβος συλλογής δεδομένων μπορεί να συνεχίζει να λαμβάνει δεδομένα. Οι κυριότερες προσεγγίσεις στη κατηγορία αυτή είναι:

Προσέγγιση βασιζόμενη σε συστάδες

Η μελέτη των [6] εξετάζει την κινητικότητα μέσα σε ένα δίκτυο αισθητήρων χωρισμένο σε συστάδες με πολλαπλούς κινητούς κόμβους. Υπέθεσαν ότι κάθε αισθητήρας προσχωρεί σε μια και μόνο μια πλησιέστερη συστάδα και στέλνει τα δεδομένα του στον αντίστοιχο κόμβο συλλογής δεδομένων που είναι και επικεφαλής της συστάδας (cluster head).

Οι κόμβοι συλλογής δεδομένων σχηματίζουν μεταξύ τους ένα συνδεδεμένο δίκτυο επικάλυψης για τη περαιτέρω συγχώνευση των δεδομένων σε ένα σταθερό σταθμό βάσης. Οι κόμβοι συλλογής δεδομένων υποτίθεται ότι πρέπει αρχικά να έχουν τοποθετηθεί με τέτοιο τρόπο ώστε κάθε συστάδα να καλύπτει περίπου την ίδια επιφάνεια δικτύου, χωρίς μεγάλες επικαλύψεις μεταξύ των συστάδων, ενώ το σύνολο του δικτύου να καλύπτεται πλήρως. Οι συστάδες, αφού σχηματιστούν στην αρχική φάση, παραμένουν αμετάβλητες καθόλη τη διάρκεια ζωής του δικτύου.

Οι κόμβοι συλλογής δεδομένων κινούνται μόνο μέσα στη δική τους συστάδα. Καθώς κινούνται, εκτελούν λειτουργίες εύρεσης διαδρομής (route discovery) με το υπόλοιπο δίκτυο των συστάδων έτσι ώστε να διατηρούν πάντα μια ενεργή διαδρομή με το σταθερό σταθμό βάσης. Αν δε βρίσκουν μια τέτοια ενεργή διαδρομή τότε επιστρέφουν πίσω στην προηγούμενη τους θέση. Στη μελέτη αυτή μπορούμε να διακρίνουμε τρεις στρατηγικές ελεγχόμενης κινητικότητας.

α. Στρατηγική βασιζόμενη στην υπολειπόμενη ενέργεια

Στη στρατηγική αυτή ένας κόμβος κινείται πάντα προς το υπολειπόμενο ενεργειακό κέντρο της συστάδας του για να εξισορροπήσει την κατανάλωση ενέργειας. Το υπολειπόμενο ενεργειακό κέντρο είναι ο μέσος όρος των θέσεων των αισθητήρων της συστάδας σταθμίζομενος με την υπολειπόμενη ενέργεια τους. Κάθε αισθητήρας στέλνει το υπολειπόμενο ενεργειακό του φορτίο και την τοπική εκτίμηση του (με βάση το ιστορικό) για τη ροή της ενέργειας στο κόμβο συλλογής δεδομένων, ο οποίος στη συνέχεια προβλέπει την ενέργεια του αισθητήρα αναλόγως εάν είναι απαραίτητο. Ωστόσο, το υπολειπόμενο ενεργειακό κέντρο θα μπορούσε να βρίσκεται μακριά από την τρέχουσα θέση που πραγματοποιείται η μεγαλύτερη ροή των δεδομένων, κάτι που θα προκαλούσε αύξηση της απόσταση για τη μετάδοση δεδομένων (πολλαπλά hops) και ως εκ τούτου αύξηση της συνολικής κατανάλωσης ενέργειας.

β. Στρατηγική βασιζόμενη στα γεγονότα

Στη στρατηγική αυτή, ένας κόμβος κινείται πάντα προς την περιοχή που εκδηλώνονται τα περισσότερα γεγονότα, δηλαδή στη περιοχή με τη μέγιστη ροή δεδομένων, μέσα στη συστάδα του. Ο στόχος της στρατηγικής αυτής είναι να μειωθούν οι διαδρομές που απαιτούνται για τη μετάδοση των δεδομένων, να μειωθεί το ενεργειακό φορτίο των αισθητήρων από την προώθηση των μηνυμάτων και τελικά να αυξηθεί η διάρκεια ζωής του δικτύου. Ωστόσο, αφού ο κόμβος φτάσει στη περιοχή με τη μέγιστη ροη δεδομένων θα παραμείνει στάσιμος εκεί. Στην περίπτωση αυτή, το σύνολο των κόμβων συλλογής δεδομένων θα παραμείνει αμετάβλητο με αποτέλεσμα περαιτέρω εξισορρόπησης της ενέργειας να μην είναι εφικτή.

γ. Υβριδική στρατηγική

Οι περιορισμοί των δύο παραπάνω στρατηγικών είχαν ως αποτέλεσμα οι συντάκτες της μελέτης να προτείνουν μια περαιτέρω στρατηγική που να συνδυάζει τα πλεονεκτήματα των δυο. Ένας κόμβος κινείται πρώτα προς την περιοχή που εκδηλώνονται τα γεγονότα και στη συνέχεια προς το υπολειπόμενο ενεργειακό κέντρο. Όταν ένας κόμβος απομακρύνεται από την περιοχή που εκδηλώνονται τα γεγονότα ενώ ένας άλλος κινείται κοντά σε αυτήν, οι αισθητήρες ανιχνεύουν το γεγονός και στέλνουν τα δεδομένα τους στο κόμβο που θα παράγει το μεγαλύτερο ενεργειακό κέρδος.

Προσέγγιση βασιζόμενη στα γεγονότα

Οι [7] αντιμετώπισαν το πρόβλημα της κινητικότητας ως πρόβλημα κίνησης ενός μοναδικού κόμβου συλλογής δεδομένων σε ένα δίκτυο αισθητήρων το οποίο βασίζεται στην εκδήλωση γεγονότων. Οι αισθητήρες στο δίκτυο αυτό παράγουν και στέλνουν τα δεδομένα των μετρήσεων τους μόνο όταν ανιχνεύουν κάποιο γεγονός. Οι κόμβοι-πηγές στέλνουν τα δεδομένα τους στο κινητό κόμβο χρησιμοποιώντας τις μικρότερες διαδρομές. Είναι αυτονόητο ότι άλλοι κόμβοι που δεν έχουν ανιχνεύσει κάποιο γεγονός δε μεταδίδουν δικά τους δεδομένα.

Η έννοια του γεγονότος αναφέρεται σε έναν «εισβολέα» που κινείται με σταθερή ταχύτητα και σε τυχαία κατεύθυνση. Δηλαδή, μόλις ένας κόμβος ανιχνεύσει την παρουσία κάποιου κινητού τέτοιου κόμβου, θα έχει ανιχνεύσει ουσιαστικά ένα γεγονός με αποτέλεσμα να αρχίσει να παράγει και να εκπέμπει τα δεδομένα του σε αυτόν.

Οι συγγραφείς τελικά καταλήγουν στο συμπέρασμα ότι, για να επιτευχθεί η χαμηλότερη συνολικά κατανάλωση ενέργειας στο δίκτυο, το άθροισμα των αποστάσεων των γεγονότων πρέπει να ελαχιστοποιηθεί με τη κατάλληλη μετακίνηση του κινητού κόμβου.

Προσέγγιση Brute-force

Οι [8] παρουσίασαν έναν brute-force αλγόριθμο για τη κινητικότητα πολλαπλών κινητών κόμβων σε ένα δίκτυο αισθητήρων. Ο αλγόριθμος αυτός εκτελείται περιοδικά για να ελέγξει αν οι κινητοί κόμβοι πρέπει να μετακινηθούν σε άλλη θέση. Η μετακίνηση των κόμβων πραγματοποιείται αν και μόνο αν η νέα θέση οδηγεί σε μείωση του συνολικού κόστους της ενέργειας στο δίκτυο. Οι κινητοί κόμβοι υποτίθεται ότι έχουν συνολική εικόνα του δικτύου και έτσι είναι σε θέση να εκτελέσουν ένα συγκεντρωτικό αλγόριθμο πάνω στο δίκτυο.

Σε κάθε ακμή στο γράφημα του δικτύου αποδίδεται και ένα βάρος για κάθε μια από τις δύο κατευθύνσεις μετάδοσης. Το βάρος κάθε ακμής εξαρτάται από το άθροισμα δύο παραγόντων: α. το αντίστροφο της εναπομένουσας ενέργειας του κόμβου που λαμβάνει τα δεδομένα και β. της ενέργειας που καταναλώνεται από τη μετάδοση των μηνυμάτων κατά μήκος της ακμής.

Όσο μειώνεται η εναπομένουσα ενέργεια του κόμβου, το βάρος της ακμής αλλάζει με την πάροδο του χρόνου. Μια συγκεντρωτική προσέγγιση δρομολόγησης των μηνυμάτων (π.χ. αλγόριθμος Dijkstra) χρησιμοποιείται για να βρεθεί η συντομότερη διαδρομή από το κάθε αισθητήρα στο πλησιέστερο κόμβο συλλογής δεδομένων του.

Προσέγγιση με Γραμμικό Προγραμματισμό

Οι [9] αντιμετώπισαν το πρόβλημα της κινητικότητας ενός μοναδικού κόμβου συλλογής δεδομένων μοντελοποιώντας το ως ένα πρόβλημα μεικτού ακέραιου γραμμικού προγραμματισμού.

Το πεδίο των αισθητήρων χωρίζεται σε ένα δισδιάστατο πλέγμα. Το σύνολο S των σημείων του πλέγματος, που ονομάζονται τοποθεσίες κόμβων συλλογής (sink sites), είναι όλες οι υποψήφιες τοποθεσίες που μπορεί να επισκεφθεί ο κόμβος συλλογής δεδομένων. Ο κόμβος κινείται βήμα-βήμα. Κάθε βήμα αποτελεί μια τοποθεσία στο δίκτυο με συγκεκριμένη ακτίνα λήψης (dMAX) στην οποία θα πρέπει να παραμένει για τουλάχιστον ένα προκαθορισμένο χρονικό διάστημα.

Όταν ο κόμβος μετακινείται οι αισθητήρες δεν μπορούν να μεταδίδουν τα διαθέσιμα δεδομένα τους. Ο κόμβος χρησιμοποιεί τη τεχνική της πλημμύρας στο δίκτυο για να ενημερώσει τους αισθητήρες για την τοποθεσία του και την ενδεχόμενη προσβασιμότητα του.

Το σύνολο S των σημείων του πλέγματος και η απόσταση dMAX ορίζουν ένα γράφο. Το πρόβλημα του μεικτού ακέραιου γραμμικού προγραμματισμού αναφέρεται στο προσδιορισμό της αρχική τοποθεσίας του κινητού κόμβου, της διαδρομής που θα κινηθεί και το χρόνο παραμονής του σε κάθε σημείο στο γράφο έτσι ώστε να μεγιστοποιηθεί ο ωφέλιμος χρόνος αποστολής των δεδομένων από τους αισθητήρες.

Στη μελέτη των [27] αντιμετωπίζεται επιπλέον το πρόβλημα της κινητικότητας για πολλαπλούς κόμβους συλλογής δεδομένων με τη χρήση ευριστικών αλγορίθμων με σκοπό τη μεγιστοποίηση της διάρκειας ζωής του δικτύου. Οι [37] παρουσιάζουν επίσης ένα ενοποιημένο πλαίσιο που καλύπτει τα ζητήματα της κινητικότητας του κόμβου συλλογής, τη δρομολόγηση των πακέτων μεταξύ των κόμβων του δικτύου και τη καθυστέρηση κατά τη μετάδοση της πληροφορίας.

Προσέγγιση Περιφέρειας

Σύμφωνα με την εργασία των [10], όταν ένα δίκτυο αισθητήρων έχει κυκλικό σχήμα και η δρομολόγηση των δεδομένων γίνεται με βάση τη συντομότερη διαδρομή η βέλτιστη στρατηγική κίνησης για τον κινητό κόμβο είναι να κινείται κατά μήκος της περιφέρειας του δικτύου.

2.2.2 Διάχυση δεδομένων

Η διάχυση δεδομένων ασχολείται με την επικοινωνία του κόμβου συλλογής δεδομένων και των αισθητήρων. Όταν η πληροφορία κινείται από ένα κόμβο – πηγή στον κόμβο συλλογής

δεδομένων η ροη δεδομένων θεωρείται καθοδική (downstream) ενώ στην ακριβώς αντίθετη περίπτωση θεωρείται ανοδική (upstream). Το κεντρικό πρόβλημα της διάχυσης δεόμενων είναι η δρομολόγηση των δεδομένων στο κόμβο συλλογής. Στη περίπτωση τώρα που έχουμε και κινητούς κόμβους συλλογής τότε το πρόβλημα της διάχυσης δεδομένων μετατρέπεται σε ένα μεικτό πρόβλημα εύρεσης τοποθεσίας και δρομολόγησης, όπου οι κόμβοι του δικτύου μοιράζονται κάποιους κοινούς προορισμούς, όπως π.χ. τους κόμβους συλλογής. Στην εργασία των [31] παρουσιάζεται ένα αποδοτικό πρωτόκολλο δρομολόγησης βασιζόμενο στη ροή των δεδομένων (Data-Driven Routing Protocol) με σκοπό την αποτελεσματική μείωση του φορτίου (overhead) του πρωτοκόλλου επικοινωνίας που προκαλείται από την ανάγκη ύπαρξης επιπλέον πληροφορίας στο πρωτόκολλο για τη συλλογή των δεδομένων από τους κινητούς κόμβους. Οι κυριότερες προσεγγίσεις στη κατηγορία αυτή είναι:

Προσέγγιση δέντρου

Οι [11] παρουσίασαν ένα πρωτόκολλο κλιμακωμένης ενεργειακής απόδοσης για ασύγχρονη διάχυση δεδομένων (Scalable Energy-efficient Asynchronous Dissemination Protocol). Για κάθε πηγή δεδομένων ο αλγόριθμος κατασκευάζει ένα Steiner Ελάχιστο Δέντρο (Steiner Minimum Tree) [42] για τη διάχυση των δεδομένων που ονομάζεται δ-δέντρο και έχει ως ρίζα τη πηγή. Αρχικά, ένα δ-δέντρο περιέχει μόνο τον κόμβο-ρίζα του. Κάθε κόμβος συλλογής επιλέγει ένα πλησιέστερο γειτονικό αισθητήρα ως κόμβο πρόσβασης. Όταν θέλει να λάβει δεδομένα από ένα κόμβο-πηγή, προσχωρεί στο δ-δέντρο του συγκεκριμένου κόμβου-πηγή μέσω του κόμβου πρόσβασης. Σε ένα δ-δέντρο, τα φύλλα είναι οι κόμβοι πρόσβασης για το κινητό κόμβο ενώ οι εσωτερικοί κόμβοι είναι προστιθέμενοι Steiner κόμβοι που ονομάζονται κόμβοι ρέπλικα (replica nodes). Η επιλογή του κόμβου πρόσβασης κάθε φορά γίνεται με διάφορα ενεργειακά κριτήρια έτσι ώστε να επιλεγεί τελικά η διαδρομή με το μικρότερο δυνατό ενεργειακό κόστος.

Προσέγγιση επιβαλλόμενης μάθησης

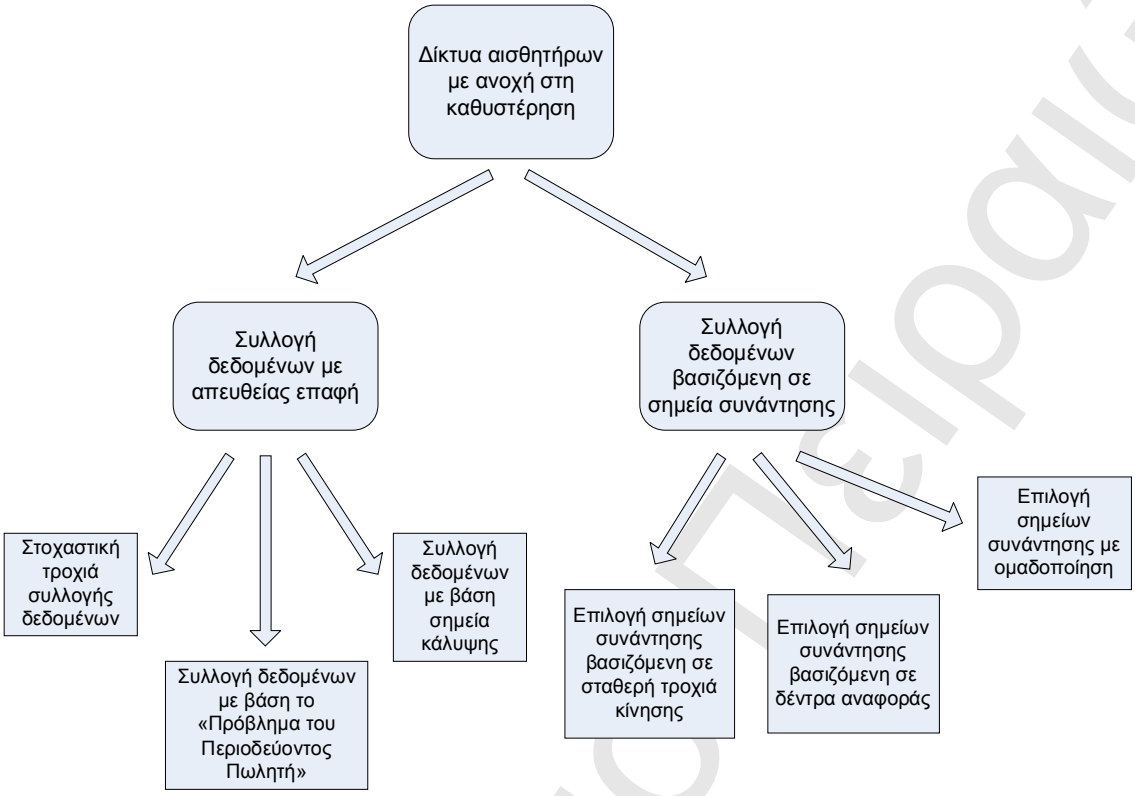
Οι [12] παρουσίασαν ένα υβριδικό αλγόριθμο επιβαλλόμενης μάθησης για τη διάδοση δεδομένων σε ένα κινητό κόμβο. Ο κινητός κόμβος κινείται ακολουθώντας ένα συγκεκριμένο μοτίβο. Η διαδρομή του κινητού κόμβου ορίζεται από τη μικρότερη χρονική διάρκεια μετά την οποία το μοτίβο μετακίνησης του επαναλαμβάνεται.

Κάθε κόμβος χωρίζει ομοιόμορφα την διαδρομή του κινητού κόμβου σε ένα προκαθορισμένο αριθμό πεδίων χρόνου και αποδίδει ένα συγκεκριμένο βάρος σε όλους τους γειτονικούς του αισθητήρες για κάθε τέτοιο πεδίο χρόνου. Το βάρος δηλώνει ποιος είναι ο καλύτερος τρόπος για να μεταβιβαστεί ένα πακέτο δεδομένων στο κινητό κόμβο στο συγκεκριμένο χρόνο.

Αρχικά, όλοι οι γείτονές έχουν την ίδια βαρύτητα, πράγμα που σημαίνει ότι είναι όλοι εξίσου «καλοί» για τη διάχυση των δεδομένων. Ο στόχος τελικά είναι να βρεθεί η καλύτερη διαδρομή για τη δρομολόγηση των πακέτων από τον κάθε αισθητήρα στο κινητό κόμβο στα διαφορετικά πεδία του χρόνου. Ο στόχος αυτός επιτυγχάνεται με την κατάλληλη προσαρμογή των βαρών των γειτονικών κόμβων μέσω μιας διαδικασία μάθησης.

2.3 Δίκτυα αισθητήρων με ανοχή στη καθυστέρηση

Στη κατηγορία αυτή των δικτύων αισθητήρων βρίσκουμε τις λύσεις που βασίζονται σε κινητούς συλλέκτες δεδομένων και σε σημεία συνάντησης. Παρακάτω παρουσιάζουμε τις χαρακτηριστικότερες προσεγγίσεις που ενδείκνυνται για τη κατηγορία αυτή των δικτύων.



Εικόνα 2.3. Οι προσεγγίσεις στα Δίκτυα Αισθητήρων με ανοχή στη καθυστέρηση

2.3.1 Συλλογή δεδομένων με απευθείας επαφή (Direct-contact data collection)

Στη συλλογή δεδομένων με απευθείας επαφή, ένα κινητός κόμβος συλλέγει τα δεδομένα απευθείας από τους αισθητήρες σε ένα ακριβώς άλμα (hop). Ο κινητός κόμβος μπορεί να αναμεταδίδει τα δεδομένα σε άλλους κινητούς κόμβους ή να τα μεταφέρει σε ένα σταθερό σταθμό βάσης. Η προσέγγιση ελαχιστοποιεί την ενέργεια που καταναλώνεται μεταξύ των αισθητήρων για την επικοινωνία αφού δε χρειάζεται να μεσολαβούν ενδιάμεσοι κόμβοι για να μεταφέρουν τα δεδομένα. Στις περιπτώσεις αυτές, ο πρωταρχικός σκοπός είναι ο υπολογισμός της καλύτερης δυνατής διαδρομής που θα κινούνται οι κινητοί κόμβοι η οποία θα καλύπτει αφενός μεν όλες τις πηγές δεδομένων και παράλληλα θα ελαχιστοποιεί τη καθυστέρηση στη συλλογή των δεδομένων [32].

Στοχαστική τροχιά συλλογής δεδομένων

Οι [13] πρότειναν έναν απλό αλγόριθμο συλλογής δεδομένων βασιζόμενοι στη στοχαστική κινητικότητα του κόμβου συλλογής. Στην πρότασή τους, οι αισθητήρες αποθηκεύουν προσωρινά τοπικά τις μετρήσεις τους και περιμένουν την άφιξη του κινητού κόμβου. Το σενάριο τους περιλαμβάνει επίσης και την ύπαρξη περισσοτέρων του ενός κινητών κόμβων. Κάθε κόμβος κινείται τυχαία και συλλέγει δεδομένα από αισθητήρες που βρίσκονται στην εμβέλεια του. Τα δεδομένα που συλλέγονται μεταφέρονται στη συνέχεια σε ένα ασύρματο σημείο πρόσβασης (π.χ. σταθερός σταθμό βάσης).

Είναι εύκολο να παρατηρήσουμε πως στο σενάριο αυτό η κατανάλωση της ενέργειας συμβαίνει κατά την ανακάλυψη του κινητού κόμβου από τους αισθητήρες και τη μετέπειτα μεταφορά των δεδομένων. Για τους λόγους αυτούς έγιναν κάποιες προτάσεις όπως το να κινείται ο κόμβος συλλογής σε ένα σταθερό μονοπάτι έτσι ώστε οι αισθητήρες να μπορούν να

προβλέπουν την ύπαρξη του κόμβου συλλογής μετά από μια διαδικασία μάθησης της διαδρομής που ακολουθεί.

Τέλος, μετά την ανακάλυψη του κόμβου συλλογής η μεταφορά των δεδομένων θα πρέπει να γίνεται με ένα έξυπνο τρόπο έτσι ώστε να μειώνεται η πιθανότητα απώλειας των μηνυμάτων και κατά συνέπεια η συχνότητα επανεκπομπής τους που αυξάνουν εντέλει το ενεργειακό κόστος.

Συλλογή δεδομένων με βάση το «Πρόβλημα του Περιοδεύοντος Πωλητή»

Όταν η κινητικότητα του κινητού κόμβου είναι ελεγχόμενη τότε η καθυστέρηση στη συλλογή δεδομένων από τους αισθητήρες μπορεί να μειωθεί με τη κατάλληλη επιλογή της τροχιάς στην οποία θα κινείται ο κινητός κόμβος. Είναι εύκολο να κατανοήσουμε ότι η επιλογή μιας τέτοιας τροχιάς κίνησης ουσιαστικά συνιστά το κλασσικό πρόβλημα του περιοδεύοντος πωλητή [42], στο οποίο ο κινητός κόμβος κατά αντιστοιχία θα πρέπει να επισκεφτεί όλους τους αισθητήρες ακριβώς μια φορά επιλέγοντας τη συντομότερη διαδρομή και στη συνέχεια να επιστρέψει στο αρχικό σημείο εκκίνησης.

Βασιζόμενη στη παραπάνω σκέψη, οι [14] [15] μοντελοποίησαν το πρόβλημα αυτό ως μια παραλλαγή του προβλήματος του περιοδεύοντος πωλητή κατά το οποίο ο κινητός κόμβος θα πρέπει να επισκέπτεται τη «γειτονιά» του κάθε αισθητήρα ακριβώς μια φορά. Η ιδέα είναι ότι ο κινητός κόμβος χρειάζεται απλά να βρίσκεται στην ακτίνα επικοινωνίας του κάθε αισθητήρα που επισκέπτεται ώστε να λαμβάνει τα δεδομένα από αυτόν.

Οι [14] παρουσίασαν επίσης έναν αλγόριθμο για την εύρεση της καλύτερης τροχιάς. Ο αλγόριθμος αυτός χρειάζεται να γνωρίζει τις θέσεις όλων των αισθητήρων στο δίκτυο. Αρχικά, επιλέγει τη σειρά που θα επισκεφτεί τις «γειτονιές» των αισθητήρων θέτοντας κάποια ενεργειακά κριτήρια όπως π.χ. το επίπεδο της ενέργειας των αισθητήρων της κάθε γειτονιάς. Στη συνεχεία, ο αλγόριθμος υπολογίζει το βέλτιστο αριθμό των σημείων αντίστοιχα που θα πρέπει να επισκεφτεί.

Πρόσφατα, οι [36] παρουσίασαν έναν κατανεμημένο αλγόριθμο για δίκτυα αισθητήρων με ανοχή στη καθυστέρηση που μπορεί να εκτελείται τοπικά σε κάθε κόμβο με σκοπό τη μεγιστοποίηση της διάρκειας ζωής του δικτύου μέσα από τη μεγιστοποίηση του αριθμού των περιοδειών που μπορεί να εκτελέσει ο κινητός κόμβος.

Συλλογή δεδομένων με βάση σημεία κάλυψης

Οι [16] [17] αντιμετώπισαν το πρόβλημα της επιλογής της διαδρομής του κινητού κόμβου με σκοπό την ελαχιστοποίηση της καθυστέρησης στη συλλογή των δεδομένων από τους αισθητήρες. Η κύρια διαφοροποίηση στη προσέγγιση τους είναι ότι εξάλειψαν την απαίτηση ο κινητός κόμβος να επισκέπτεται τη «γειτονιά» του αισθητήρα ακριβώς μια φορά. Η ιδέα είναι ότι ο χρόνος περιφοράς του κινητού κόμβου θα μπορούσε να είναι αρκετά μεγάλος αν η ακτίνα μετάδοσης των αισθητήρων που βρίσκονται στο μήκος της διαδρομής του είναι μικρή, αφού τότε ο κινητός κόμβος θα είναι αναγκασμένος να επιβραδύνει στα σημεία αυτά για να συλλέξει όλα τα δεδομένα από τους αισθητήρες.

Συνεπώς, η στρατηγική της μοναδικής επίσκεψης του κάθε κόμβου μπορεί να μην είναι πάντα η ενδεδειγμένη λύση. Οι πολλαπλές επισκέψεις μαζί με τον κατάλληλο έλεγχο της ταχύτητας του κινητού κόμβου μπορούν να αποτελέσουν μια ιδανική λύση [26] [29].

Ο στόχος τελικά της προσέγγισης αυτής είναι να βρεθεί μια τροχιά ελαχίστου κόστους κατά την οποία ο κινητός κόμβος μπορεί να συλλέγει δεδομένα από όλους τους αισθητήρες. Με άλλα λόγια, η συντομότερη τροχιά κατά την οποία όλα τα συνδεδεμένα σημεία καλύπτουν όλους τους αισθητήρες.

2.3.2 Συλλογή δεδομένων βασιζόμενη σε σημεία συνάντησης (Rendez-vous)

Η συλλογή δεδομένων με απευθείας επαφή έχει μεγάλο πλεονέκτημα για την εξοικονόμηση ενέργειας, ωστόσο αυξάνει σημαντικά τη καθυστέρηση στη συλλογή των δεδομένων λόγω της χαμηλής ταχύτητας που κινείται ο κινητός κόμβος. Η συλλογή δεδομένων βασιζόμενη σε σημεία συνάντησης αποσκοπεί στο να πετύχει τη χρυσή τομή ανάμεσα στην καταναλωμένη ενέργεια και τη χρονική καθυστέρηση [23] [24] [28].

Οι αισθητήρες στέλνουν τα δεδομένα τους με πολλαπλά hops σε ένα υποσύνολο αισθητήρων που ονομάζονται σημεία συνάντησης (Rendez-vous points). Ένας κινητός κόμβος συλλογής κινείται στο δίκτυο και συλλέγει τα δεδομένα από τα σημεία συνάντησης. Η χρήση των σημείων συνάντησης επιτρέπει στον κινητό κόμβο να συλλέγει ένα μεγάλο όγκο δεδομένων έγκαιρα χωρίς να χρειάζεται να διανύει μεγάλες αποστάσεις μειώνοντας έτσι αισθητά τη καθυστέρηση στη συλλογή των δεδομένων. Οι περισσότερες προσεγγίσεις στη κατηγορία αυτή επικεντρώνονται κυρίως στους τρόπους επιλογής των σημείων συνάντησης [1] [19] [20] [23]. Θα πρέπει να παρατηρήσουμε ότι αφού τα σημεία συνάντησης είναι στατικοί κόμβοι η διάχυση των δεδομένων στα σημεία συνάντησης είναι παρόμοια με αυτή στους στατικούς κόμβους συλλογής δεδομένων η οποία και έχει μελετηθεί εξονυχιστικά στα παραδοσιακά στατικά δίκτυα αισθητήρων.

Επιλογή σημείων συνάντησης βασιζόμενη σε σταθερή τροχιά κίνησης

Οι [18] πρότειναν τη χρήση μια ευθείας διαδρομής που θα ακολουθεί ο κινητός κόμβος κατά τη συλλογή των δεδομένων.

Σε μια φάση αρχικοποίησης, ο κινητός κόμβος μεταδίδει ένα μήνυμα Beacon, ενώ κινείται κατά μήκος μιας ευθείας γραμμής. Το μήνυμα έχει ένα πεδίο που υποδηλώνει τον αριθμό των hops που έχει διανύσει. Κάθε στατικός κόμβος που λαμβάνει το μήνυμα το αναμεταδίδει αν και μόνο αν το μήνυμα έχει μικρότερο αριθμό hops από τον αριθμό που έχει αποθηκευμένο στη μνήμη του. Πριν αναμεταδώσει το μήνυμα, ο κόμβος αφενός μεν φροντίζει να αυξήσει κατά ένα τον αριθμό των hops του πεδίου και αφετέρου να αποθηκεύσει στη μνήμη του το αναγνωριστικό του κόμβου από τον οποίο έλαβε το μήνυμα. Έτσι, μετά το τέλος της φάσης αρχικοποίησης θα έχει σχηματιστεί ένας αριθμός από δέντρα, καθένα από τα οποία θα έχει ως ρίζα του ένα κόμβο που θα βρίσκεται στην άμεση εμβέλεια του κινητού κόμβου ενώ ο κάθε κόμβος θα ανήκει σε ακριβώς ένα μόνο τέτοιο δέντρο.

Η ρίζα του κάθε δέντρου λειτουργεί ως σημείο συνάντησης με το κινητό κόμβο. Οι αισθητήρες κάθε τέτοιου δέντρου στέλνουν τα δεδομένα τους στη ρίζα του δέντρου. Καθώς κινείται ο κινητός κόμβος οι κόμβοι που βρίσκονται στα σημεία συνάντησης στέλνουν τα δικά τους δεδομένα μαζί με αυτά που έχουν λάβει από τους κόμβους του δέντρου τους στο κινητό κόμβο.

Στην εργασία τους παρουσιάζουν επίσης δύο αλγορίθμους ελέγχου της κίνησης του κινητού κόμβου με σκοπό την αύξηση του αριθμού των συλλεγόμενων δεδομένων από τους στατικούς κόμβους. Στον πρώτο αλγόριθμο ο κινητός κόμβος σταματά για κάποιο χρονικό διάστημα σε σημεία όπου βρίσκονται αισθητήρες που έχουν ακόμα δεδομένα να στείλουν. Στον άλλο αλγόριθμο, ο κινητός κόμβος κινείται βραδύτερα σε περιοχές όπου το ποσοστό επιτυχίας των δεδομένων που παραλαμβάνονται είναι μικρό ενώ σταματά τελείως προσωρινά σε περιοχές όπου η απώλεια των δεδομένων είναι αρκετά σοβαρή.

Η εργασία των [1] παρουσιάζει μια αναλυτική πρόταση για την υλοποίηση ενός τέτοιου συστήματος δίνοντας συγκεκριμένες λύσεις που αφορούν κάθε φάση του αλγορίθμου με σκοπό αφενός μεν τη μείωση της χρονικής καθυστέρησης για την αποστολή των δεδομένων στο κινητό κόμβο και αφετέρου τη μείωση της καταναλωμένης ενέργειας.

Επιλογή σημείων συνάντησης βασιζόμενη σε δέντρα αναφοράς

Οι [19] [20] μελέτησαν την επιλογή των σημείων συνάντησης με δέντρα αναφοράς που υπόκειται σε ένα συγκεκριμένο χρονικό όριο για τη συλλογή των δεδομένων. Με βάση τη

προσέγγιση αυτή θα πρέπει τα σημεία συνάντησης να επιλεγούν από ένα δέντρο αναφοράς κατά τέτοιο τρόπο έτσι ώστε η περιοδεία του κινητού κόμβου από τα σημεία συνάντησης δεν θα είναι μεγαλύτερη από τη μέγιστη απόσταση L που μπορεί να διανύσει ο κινητός κόμβος μέσα στο συγκεκριμένο χρονικό όριο. Στην εργασία τους περιγράφουν την κατασκευή ενός τέτοιου δέντρου αναφοράς και προτείνουν λύσεις όπως τη χρήση ευριστικών αλγορίθμων για τη κίνηση του κινητού κόμβου και τη χρήση ενός Steiner Ελαχίστου Δέντρου [42] (ένα δέντρο ελαχίστου μήκους που συνδέει έναν αριθμό δεδομένων σημείων) ως δέντρο αναφοράς για την επιλογή των σημείων συνάντησης.

Επιλογή σημείων συνάντησης με ομαδοποίηση

Οι [21] [25] παρουσίασαν ένα γενικό πλαίσιο επικοινωνίας για την συλλογή των δεδομένων που βασίζεται σε κινητό κόμβο με την ενσωμάτωση στο πλαίσιο αυτό αρκετών υφιστάμενων αλγορίθμων. Σε αυτό το πλαίσιο αρχικά κατασκευάζεται ένα σύνολο k-hop κόμβων. Το σύνολο των κόμβων αυτών ονομάζεται κυρίαρχο σύνολο (dominating set). Όλοι οι κόμβοι που βρίσκονται στο κυρίαρχο σύνολο ονομάζονται παράγοντες πλοήγησης (navigation agents). Δύο παρακείμενοι παράγοντες πλοήγησης είναι τουλάχιστον k+1 και το πολύ 2k + 1 hops μακριά ο ένας από τον άλλον.

Κάθε παράγοντας πλοήγησης κατασκευάζει ένα δέντρο ελαχίστου μήκους με ρίζα τον εαυτό του και επεκτείνεται σε ένα μέγιστο βάθος 2k + 1 hops κόμβων. Κατά τη κατασκευή του δέντρου ο παράγοντας πλοήγησης αναγνωρίζει τους παρακείμενους παράγοντες πλοήγησης και χτίζει τις συντομότερες διαδρομές με αυτούς. Οι κόμβοι που βρίσκονται ανάμεσα σε μια τέτοια διαδρομή ονομάζονται ενδιάμεσοι κόμβοι και εξυπηρετούν στο να καθοδηγούν το κινητό κόμβο καθώς κινείται ανάμεσα από τα σημεία συνάντησης.

Με τη κατάλληλη ρύθμιση της παραμέτρου k, το πλαίσιο αυτό περνά από την απευθείας επαφή για τη συλλογή των δεδομένων, μέσω της συλλογής δεδομένων με βάση τα σημεία συνάντησης, στην παραδοσιακή συλλογή των δεδομένων με βάση στατικούς κόμβους συλλογής.

Οι [33] πρότειναν ένα νέο αλγόριθμο κατά τον οποίο η επιλογή της τροχιάς του κινητού κόμβου και η κατασκευή των δέντρων για την αποστολή των δεδομένων στα σημεία συνάντησης αντιμετωπίζονται ως ξεχωριστές φάσεις ενώ ο αλγόριθμος εναλλάσσεται μεταξύ των φάσεων αυτών με σκοπό τη βελτίωση των αποτελεσμάτων της μιας φάσης με βάση τα αποτελέσματα της άλλης.

3. Η τεχνολογία των αισθητήρων JAVA Sun SPOTs

Τα Sun SPOTs (Small Programmable Object Technology) είναι μικρές ασύρματες συσκευές με ενσωματωμένους αισθητήρες που λειτουργούν με μπαταρίες και είναι προγραμματιζόμενες σχεδόν εξολοκλήρου σε γλώσσα προγραμματισμού Java [2] [38].

Η τρέχουσα διαμόρφωση της πλατφόρμας του Sun SPOT που μελετάμε για τις ανάγκες της παρούσας διπλωματικής εργασίας είναι το eSPOT. Το eSPOT έχει έναν κεντρικό επεξεργαστή ο οποίος «τρέχει» την εικονική μηχανή της Java (JAVA VM) με την ονομασία «Squawk» και χρησιμεύει ως ένας ασύρματος κόμβος δικτύου τύπου IEEE 802.15.4. Το eSPOT έχει ευέλικτη διαχείριση ενέργειας και μπορεί να τροφοδοτείται μέσω επαναφορτιζόμενης μπαταρίας, μέσω σύνδεσης USB από κάποιο τερματικό μηχάνημα ή να τροφοδοτείται εξωτερικά.

Το Sun SPOTs έχουν σχεδιαστεί με τέτοιο τρόπο ώστε να αποτελούν μια ευέλικτη πλατφόρμα ανάπτυξης, ικανή να φιλοξένει μια ευρεία γκάμα από διαφορετικές μονάδες εφαρμογών. Το πακέτο ανάπτυξης των Sun SPOTs είναι το Sun SPOT Java Development Kit και περιέχει δύο διαφορετικές διαμορφώσεις των Sun SPOTs.

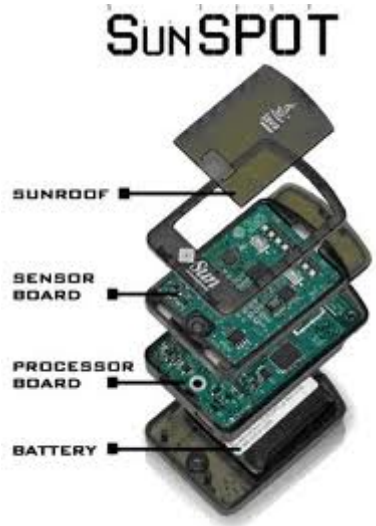
Εικόνα 3.1. Το Sun SPOT Java Development Kit [Sun Microsystems]

Συγκεκριμένα, οι διαμορφώσεις των Sun SPOTs που περιλαμβάνει το πακέτο ανάπτυξης είναι:

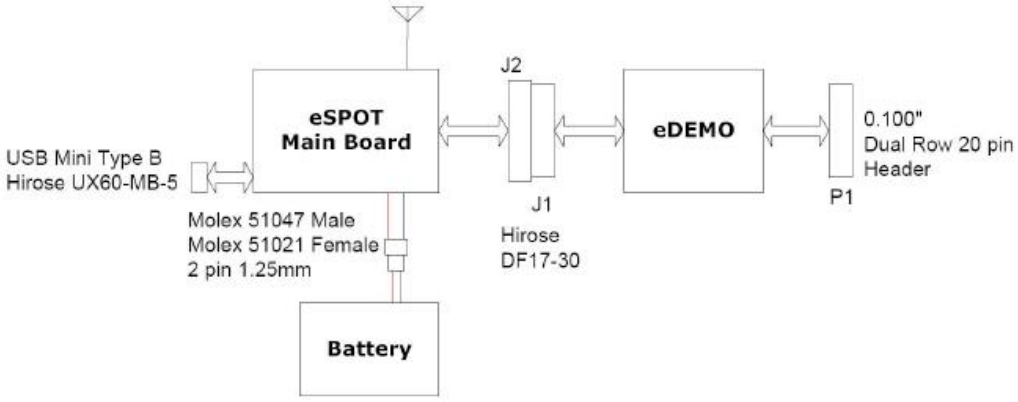
- *Σταθμός Βάσης (Basestation)*
  Ο Σταθμός Βάσης αποτελείται από μια κεντρική μονάδα eSPOT (main board) χωρίς όμως μπαταρία και μονάδα εφαρμογών (πίνακα αισθητήρων). Η τροφοδοσία του σταθμού βάσης γίνεται μέσω σύνδεσης USB από ένα τερματικό Η/Υ (workstation). Ο Σταθμός Βάσης λειτουργεί ως ένας δρομολογητής ασύρματης επικοινωνίας μεταξύ

άλλων Sun SPOTs (καθώς και θεωρητικά οποιασδήποτε άλλης συσκευής τύπου 802.15.4) και του τερματικού Η/Υ στο οποίο είναι συνδεδεμένος.

- *eSPOT*
  Το eSPOT είναι η συνηθισμένη διαμόρφωση ενός Sun SPOT και περιλαμβάνει πέρα από τη κεντρική μονάδα eSPOT μια πρισματική επαναφορτιζόμενη μπαταρία λιθίου (LI-ION) και μια δευτερεύουσα μονάδα eSPOT (daughterboard) γνωστή ως eDEMO.



Εικόνα 3.2. Τα διάφορα μέρη ενός Sun SPOT [Sun Microsystems]



Εικόνα 3.3. Η αρχιτεκτονική ενός eSPOT και οι διασυνδέσεις του [2]

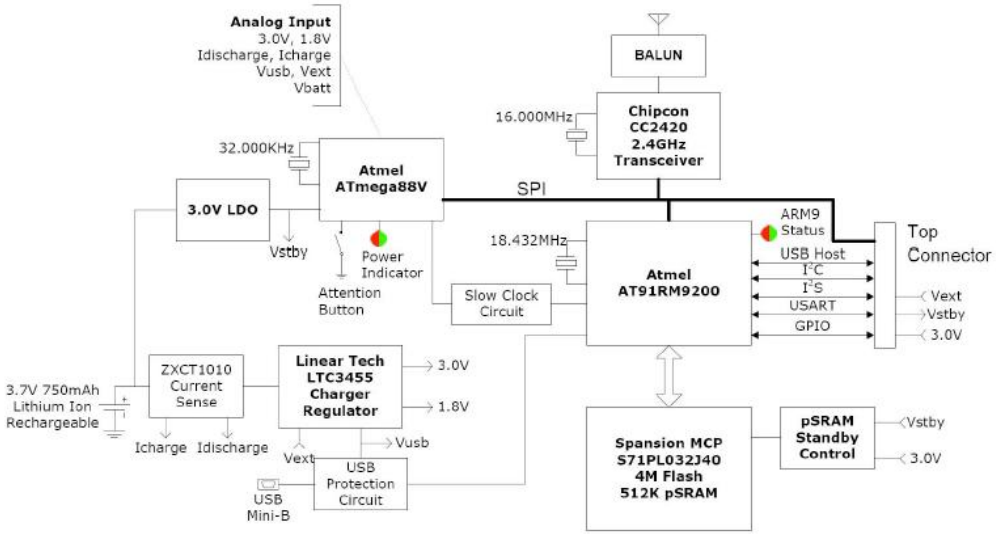
Το eDEMO ανήκει στη κατηγορία των daughterboards που είναι συμβατά με τη κεντρική μονάδα του eSPOT. Η πλατφόρμα του eDEMO περιέχει [2]:

- τρισδιάστατο επιταχυνσιόμετρο (3-axis accelerometer)
- αισθητήρα περιβάλλοντος φωτός
- αισθητήρα θερμοκρασίας
- οκτώ τρίχρωμα LEDs (tricolor LEDs)
- δυο πιεζόμενα κουμπιά (push buttons)
- έξι αναλογικές εισόδους
- τέσσερις αναλογικές εξόδους υψηλής τάσης
- πέντε ψηφιακές θύρες εισόδου – εξόδου γενικού σκοπού

3.1 Η Κεντρική μονάδα του eSpot

Η κεντρική μονάδα του eSpot περιλαμβάνει τα εξής ηλεκτρονικά μέρη [2]:

- κεντρικό επεξεργαστή
- μονάδα μνήμης
- ηλεκτρονικό κύκλωμα διαχείρισης της ενέργειας
- ράδιο-δέκτη τύπου 802.15.4 και κεραία
- διασύνδεση με τη μπαταρία
- διασύνδεση με τη δευτερεύουσα μονάδα



Εικόνα 3.4. Τα κύρια μέρη της κεντρικής μονάδας του eSPOT [2]

Κεντρικός επεξεργαστής

Ο κεντρικός επεξεργαστής είναι ένα Atmel AT91RM9200 system on a chip (SOC) ενσωματωμένο κύκλωμα. Η μονάδα αυτή περιέχει τον επεξεργαστή ARM920T ARM Thumb, αρχιτεκτονικής τύπου v4T ARM με την ονομασία ARM9TDMI. Το ρεύμα που δέχεται ο επεξεργαστής για το I/O είναι τάσης 3V ενώ η τάση του πυρήνα είναι 1,8V. Σε κανονικές συνθήκες λειτουργίας ο επεξεργαστής καταναλώνει περίπου 44mW ρεύμα ενώ η μέγιστη ταχύτητα εκτέλεσης εντολών φτάνει τα 180 MHz. Το ενσωματωμένο κύκλωμα του επεξεργαστή περιλαμβάνει επίσης μια 64-πλευρη σχεσιακή cache μνήμη δεδομένων (data cache) των 16Kbytes και μια cache μνήμη εντολών (instruction cache) των 16Kbytes επίσης.

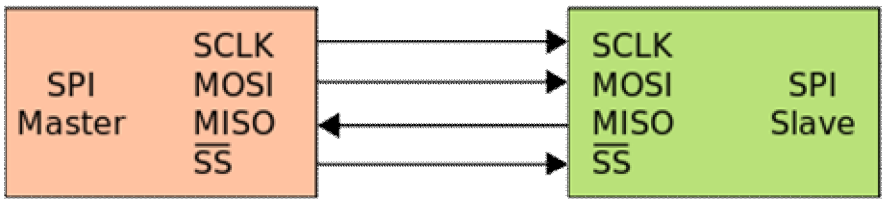
Επικοινωνία με τη κεντρική μονάδα

Η επικοινωνία με τη κεντρική μονάδα του eSpot γίνεται με τους εξής τρόπους:

- USB (έκδοση USB 1.1 και 2.0)
- SPI

Το SPI (Serial Peripheral Interface Bus) είναι η πρωταρχική διεπαφή για τις περισσότερες ενσωματωμένες συσκευές της κεντρικής μονάδας και για την επικοινωνία των συσκευών μεταξύ τους . Το SPI είναι ένα σύγχρονο σειριακό πρότυπο επικοινωνίας δεδομένων και λειτουργεί σε κατάσταση full duplex. Οι συσκευές επικοινωνούν με το πρότυπο master – slave, όπου η συσκευή που είναι στο master είναι υπεύθυνη για το συγχρονισμό και την εκκίνηση αποστολής των δεδομένων.

Η επικοινωνία στο SPI βασίζεται σε 4 σήματα όπως φαίνεται στο παρακάτω σχήμα:



Εικόνα 3.5. Η επικοινωνία στο SPI [Wikipedia]

SCLK: Serial Clock (output from master)

MOSI; SIMO: Master Output, Slave Input (output from master)

MISO; SOMI: Master Input, Slave Output (output from slave)

SS: Slave Select (active low, output from master)

Μνήμη

Η μνήμη στη Κεντρική Μονάδα είναι η Spansion S71PL032J40, και αποτελείται από 4Mbyte NOR flash και 512Kbyte pSRAM (pseudo-SRAM) που βρίσκονται στο ίδιο chip. Ο χρόνος πρόσβασης(access time) για την pSRAM είναι 70nsec και για την Flash 65nsec και έχουν 16-bit δίαυλο δεδομένων. Και οι δυο χρησιμοποιούν τροφοδοσία 3Volt και σε κανονικές συνθήκες λειτουργίας η κατανάλωση είναι 25ma για την pSRAM και 22ma για την Flash. Τα δεδομένα στην pSRAM διατηρούνται όσο το Sun SPOT είναι συνδεδεμένο σε κάποια τροφοδοσία ή μπαταρία. Όταν το Sun SPOT βρίσκεται σε κατάσταση deep-sleep, όπου τα περισσότερα υποσυστήματα δεν τροφοδοτούνται για εξοικονόμηση ενέργειας η pSRAM καταναλώνει περίπου 8μA για την διατήρηση των δεδομένων της ενώ η flash απενεργοποιείται. Η flash είναι προγραμματισμένη ήδη από το εργοστάσιο και περιέχει τον bootloader, την Squawk VM, τις βασικές βιβλιοθήκες και μια προ εγκατεστημένη εφαρμογή (bounce demo).

Κύκλωμα τροφοδοσίας

Το Sun SPOT μπορεί να λειτουργήσει χρησιμοποιώντας οποιοδήποτε συνδυασμό από της εξής πηγές: την επαναφορτιζόμενη μπαταρία, το USB Host ή την εξωτερική τροφοδοσία. Το κύκλωμα τροφοδοσίας είναι υπεύθυνο για να φορτίζει την ενσωματωμένη μπαταρία, να ρυθμίζει το ρεύμα που παρέχεται στα υποσυστήματα του Main Board και του Sensor Board (eDEMO Board) είτε το Sun SPOT βρίσκεται σε κανονική λειτουργία είτε σε deep-sleep. Το κύκλωμα αποτελείται από δύο τμήματα-κυκλώματα, κάθε ένα με διαφορετική λειτουργία LTC3455 και το TPS79730. Το LTC3455 έχει ενσωματωμένο, ένα κύκλωμα για την φόρτιση της μπαταρίας Li-ION ένα διαχειριστή ρεύματος για την USB και ένα διπλό σταθεροποιητή τάσης. Το LTC3455 διαχειρίζεται το ρεύμα που λαμβάνεται από την USB. Ανάλογα με τις απαιτήσεις της συσκευής, ο επεξεργαστής επιτρέπει την κατανάλωση περισσότερου ρεύματος από την USB. Το TPS79730 είναι ένας σταθεροποιητής τάσης και παρέχει μικρή ποσότητα ρεύματος στα 3Volt στην περίπτωση που το Sun SPOT εισέλθει σε κατάσταση stand-by, επίσης παρέχει σταθερό ρεύμα στον Atmega88 και στην pSRAM και σε περίπτωση που η τάση πέσει κάτω από τα ασφαλή όρια λειτουργίας του επεξεργαστή τον απενεργοποιεί.

Καταστάσεις λειτουργίας

Τα SPOT έχουν ειδικό firmware για εξοικονόμηση ενέργειας που μπορεί να θέσει την συσκευή σε τρεις καταστάσεις λειτουργίας:

- Run

Είναι η βασική κατάσταση στην οποία όλοι οι επεξεργαστές και το radio τροφοδοτούνται και λειτουργούν κανονικά. Η κατανάλωση σε αυτήν την κατάσταση φτάνει κυμαίνεται από 70mA ως 120mA, ενώ η κάρτα επέκτασης μπορεί να καταναλώνει μέχρι 400mA.

- Idle

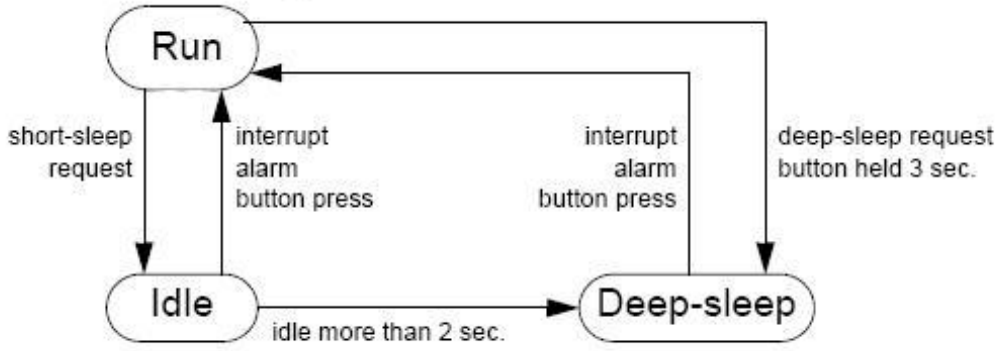
  Σε αυτή την κατάσταση το ρολόι του επεξεργαστή σταματάει και το radio απενεργοποιείται ενώ η κατανάλωση πέφτει στα 24mA.

- Deep-Sleep

  Σχεδόν όλα τα κυκλώματα τροφοδοσίας απενεργοποιούνται εκτός από το κύκλωμα που δίνει ελάχιστο ρεύμα για την διατήρηση των δεδομένων της pSRAM. Η επαναφορά της συσκευής από την κατάσταση Deep-Sleep διαρκεί περίπου 2msec με 10msec.

Για να εισέλθει η συσκευή σε κατάσταση χαμηλής κατανάλωσης(Deep-Sleep) πρέπει το radio να είναι απενεργοποιημένο, να μην παρέχεται ρεύμα από εξωτερική συσκευή και να μην είναι ενεργοποιημένη η USB. Το Sun SPOT εισέρχεται στις καταστάσεις Deep-Sleep και Idle καλώντας κατάλληλες συναρτήσεις της βιβλιοθήκης. Επιπλέον, θα μπορούσαμε να θέσουμε την συσκευή σε Deep-Sleep πατώντας το attention κουμπί για περισσότερα από 3 δευτερόλεπτα. Για να εξέλθει η συσκευή από Deep-Sleep πρέπει να χρησιμοποιήσουμε κάποιο εξωτερικό interrupt ή να πιέσουμε το attention κουμπί.

Η παρακάτω εικόνα δείχνει τις μεταβάσεις που μπορεί να γίνουν μεταξύ των διαφορετικών καταστάσεων λειτουργίας.



Εικόνα 3.6. Οι καταστάσεις λειτουργίας των Sun SPOTs [2]

Μπαταρία

Η μπαταρία που χρησιμοποιείται στα Sun SPOTs είναι μια επαναφορτιζόμενη μπαταρία ιόντων λιθίου Li-ION στα 3.7V με χωρητικότητα 720mAH. Η μπαταρία ενσωματώνει κυκλώματα για την προστασία της από πλήρη αποφόρτιση, από υπερφόρτιση και από υψηλή τάση. Η φόρτιση μπορεί να γίνει είτε χρησιμοποιώντας ένα USB καλώδιο με βύσμα τύπου B είτε από οποιαδήποτε πηγή 5Volt (+/- 10%). Όταν δεν χρησιμοποιείται χάνει περίπου 2% της χωρητικότητας κάθε μήνα και σε περιπτώσεις υψηλής θερμοκρασίας ο ρυθμός αυτός αυξάνει. Τα κυκλώματα φόρτισης και διαχείρισης ρεύματος είναι ρυθμισμένα με ακρίβεια για να λειτουργούν με τον συγκεκριμένο τύπο μπαταρίας και για αυτό δεν πρέπει να αντικατασταθεί από άλλου τύπου.

Ελεγκτής τροφοδοσίας

Πρόκειται για τον 8-bit μικροελεγκτή Atmega88 της Atmel. Έχει ενσωματωμένο firmware που είναι υπεύθυνο για την λειτουργία του 64-bit ρολογιού, την επαναφορά της συσκευής σε περίπτωση που δεχτεί εξωτερικό interrupt και την επαναφορά ή είσοδό σε Deep-Sleep όταν πιεστεί το attention κουμπί. Η επικοινωνία με τον επεξεργαστή γίνεται μέσω του SPI διαύλου, από τον οποίο μεταφέρονται εντολές και δεδομένα κατάστασης από και προς τον Atmega88.

Επίσης ο ελεγκτής μετράει και παρακολουθεί το φορτίο της μπαταρίας και τις τάσεις της USB, της μπαταρίας και των εσωτερικών υποσυστημάτων χρησιμοποιώντας ένα 10-bit ACD. Ακόμα ο Atmega88 ελέγχει το power LED και δηλώνει διαφορετικές καταστάσεις (προβληματικές ή όχι) του Sun SPOT μέσω ενδείξεων αυτού του LED. Για παράδειγμα όταν ανιχνεύσει ότι η μπαταρία έχει σχεδόν αποφορτιστεί ο ελεγκτής θέτει το power LED μόνιμα κόκκινο.

Ασύρματος πομποδέκτης (Wireless Radio)

Τα Sun SPOTs για την ασύρματη μετάδοση δεδομένων ενσωματώνουν τον ασύρματο πομποδέκτη CC2420. Το CC2420 συμμορφώνεται με το πρότυπο IEEE 802.15.4 και λειτουργεί σε συχνότητες από 2.4GHz ως 2.4835GHz οι συχνότητες αυτές ανήκουν στην ISM ζώνη και εξαιρούνται αδειοδότησης. Το κύκλωμα CC2420 εκτός από τον πομποδέκτη περιέχει δυο FIFOs 128bytes για τα TX και RX δεδομένα, δυνατότητα για μέτρηση του RSSI με ευαισθησία 100db και ρύθμιση ισχύς του πομπού από -24dBm ως 0dBm. Ο πρακτικός ρυθμός μετάδοσης δεδομένων φτάνει τα 250Kbit/s ενώ η ευαισθησία του δέκτη είναι -90dBm. Για τα σήματα ελέγχου και δεδομένων από και προς το CC2420 στη Κεντρική Μονάδα χρησιμοποιούνται PIO θύρες και ο δίαυλος SPI. Στις PIO θύρες συνδέονται τα σήματα ελέγχου όπως reset, power down, start of frame (SFD) και σήματα κατάστασης όπως FIFO και FIFOP που ενημερώνουν αν η ουρά δεδομένων είναι άδεια ή αν έχουν ληφθεί δεδομένα. Ο δίαυλος SPI χρησιμοποιείται για την μεταβίβαση δεδομένων προς το CC2420. Το κύκλωμα καταναλώνει 20mA όταν ο δέκτης λαμβάνει δεδομένα και 18mA κατά την διάρκεια μετάδοσης με ισχύ 0dBm.

Channel	Center Frequency
11	2405MHz
12	2410MHz
13	2415MHz
14	2420MHz
15	2425MHz
16	2430MHz
17	2435MHz
18	2440MHz

Channel	Center Frequency
19	2445MHz
20	2450MHz
21	2455MHz
22	2460MHz
23	2465MHz
24	2470MHz
25	2475MHz
26	2480MHz

Εικόνα 3.7. Τα κανάλια και οι συχνότητες επικοινωνίας των Sun SPOTs [2]

PA_LEVEL	Output Power
31	0dBm
27	-1dBm
23	-3dBm
19	-5dBm

PA_LEVEL	Output Power
15	-7dBm
11	-10dBm
7	-15dBm
3	-25dBm

Εικόνα 3.8. Τα επίπεδα ισχύος του πομπού [2]

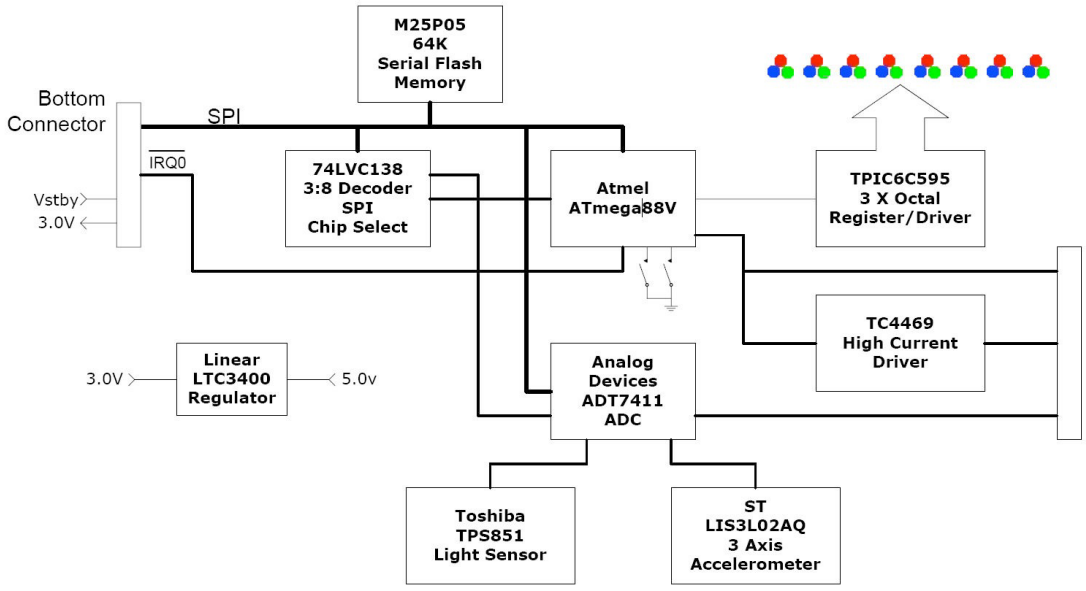
Κεραία

Η κεραία του Sun SPOT είναι τύπου inverted-F, τυπωμένη στην άνω επιφάνεια του PCB (Printed Circuit Board) του Main Board. Είναι σχεδιασμένη για να συντονίζεται στη συχνότητα 2450MHz με ωμική αντίσταση 115Ω. Λόγο της θέσης την κεραίας θα πρέπει να αποφεύγουμε την τοποθέτηση μεταλλικών αντικειμένων ή γραμμών τροφοδοσίας κοντά σε αυτήν. Σε εξωτερικό χώρο, κάτω από καλές καιρικές συνθήκες η εμβέλεια φτάνει τα 100m ενώ σε εσωτερικούς χώρους περιορίζεται στα 30m.

3.2 Η δευτερεύουσα μονάδα eDEMO

Το eDEMO Board είναι η κάρτα επέκτασης (daughterboard) του eSPOT. Αυτή η κάρτα είναι ενσωματωμένη στα Sun SPOTs που υπάρχουν στο αναπτυξιακό πακέτο της SUN και προσφέρει μια ποικιλία από αισθητήρες και I/O θύρες. Στο eSPOT Main Board μπορούν να συνδεθούν και διαφορετικές κάρτες επέκτασης και αυτή την στιγμή είναι υπό σχεδίαση αρκετές κάρτες με πιο προηγμένες δυνατότητες και πιο ευαίσθητα όργανα. Προϋπόθεση για την προθήκη μιας κάρτας επέκτασης στο eSPOT είναι να συνδέεται με το Main Board μέσω ενός βύσματος Hirose DF17-30, να υποστηρίζει το SPI interface αφού μέσω αυτού του διαύλου γίνεται η επικοινωνία και να περιέχει μια SPI flash για την αποθήκευση πληροφορίας σχετικά με τις παραμέτρους λειτουργίας της.

Στο παρακάτω σχήμα μπορούμε να δούμε το υλικό (hardware) του eDEMO Board:



Εικόνα 3.9. Η αρχιτεκτονική του eDEMO [2]

4. Ο προγραμματισμός των αισθητήρων JAVA Sun SPOTs

Ο προγραμματισμός των SUN SPOTs γίνεται με την γλώσσα προγραμματισμού Java [3]. Συγκεκριμένα, οι προγραμματιζόμενες εφαρμογές ακολουθούν τις προδιαγραφές του προτύπου MIDP (Mobile Information Device Profile) που είναι χτισμένο πάνω στο framework CLDC (Connected Limited Device Configuration) και προσθέτει μια επιπλέον συλλογή από APIs για εφαρμογές σε ενσωματωμένες συσκευές. Το MIDP χρησιμοποιείται σε πολλά ενσωματωμένα συστήματα και συσκευές όπως για παράδειγμα τα κινητά τηλέφωνα.

Τα συγκεκριμένα προγράμματα που ακολουθούν τις παραπάνω προδιαγραφές καλούνται MIDlets και έχουν συγκεκριμένη δομή και περιορισμούς. Τα MIDlets τρέχουν σε μια μικρή εικονική μηχανή Java ME (J2ME) με την κωδική ονομασία «Squawk VM». Είναι σημαντικό να τονίσουμε ότι τα SPOTs δεν έχουν λειτουργικό σύστημα αλλά τον ρόλο του λειτουργικού αναλαμβάνει απευθείας η Squawk VM. Μαζί με τα MIDlet του χρήστη αλλά σε «χαμηλότερο» επίπεδο τρέχουν και μια σειρά άλλων εφαρμογών του συστήματος που δεν είναι άμεσα «ορατές» στον χρήστη:

- Η εφαρμογή του Bootloader
  Ο Bootloader είναι υπεύθυνος για την διαχείριση της USB σύνδεσης μεταξύ του SPOT και του Η/Υ, εκκινεί τις εφαρμογές και επικοινωνεί με τα ant scripts του Η/Υ που είναι συνδεδεμένο το SPOT.
- Η σουίτα εφαρμογών bootstrap suite
  Περιλαμβάνει τις πρότυπες κλάσεις της Java ME.
- Η σουίτα εφαρμογών library suite
  Περιλαμβάνει την βιβλιοθήκη με τις κλάσεις που είναι σχετικές με τα Sun SPOTs.

Η Squawk VM χρησιμοποιεί ανεξάρτητες περιοχές για εκτέλεση εφαρμογών, τα isolates. Κάθε isolate συνιστά ένα διαφορετικό σύνολο από νήματα και αντικείμενα που σχετίζονται με αυτά. Το Sun SPOT έχει πάντα ένα master isolate, στο οποίο τρέχουν διάφορα νήματα του συστήματος (deamon threads) που διαχειρίζονται τις βασικές λειτουργίες του. Αυτά τα νήματα είναι τμήμα της βασικής βιβλιοθήκης των Sun SPOTs και φροντίζουν για την διαχείριση της ενέργειας (απενεργοποιώντας υποσυστήματα που δεν χρησιμοποιούνται), παρακολουθούν την κατάσταση της σύνδεσης USB (USB listener) και αποτελούν μέρος της υλοποίησης ράδιο στοίβας (radiostack).

Τα υπόλοιπα isolates που μπορεί να δημιουργηθούν καλούνται child isolates. Η προκαθορισμένη επιλογή στα Sun SPOTs είναι οι εφαρμογές του χρήστη να τρέχουν στο master isolate χωρίς όμως αυτό να είναι υποχρεωτικό. Τα νήματα μπορούν να ορισθούν προγραμματιστικά στα MIDlet μέσω της κλάσης Thread και μπορούμε να ορίσουμε την προτεραιότητα των νημάτων από την ελάχιστη τιμή 1 (Thread.MIN_PRIORITY) έως και μέγιστη τιμή 10 (Thread.MAX_PRIORITY).

Επίσης, υπάρχει η δυνατότητα να δοθούν ακόμα υψηλότερες προτεραιότητες όπως είναι οι «προτεραιότητες του συστήματος» (system priorities), αυτές όμως αφορούν ορισμένα νήματα του συστήματος και δεν προορίζονται για χρήση από τα MIDlets.

Θα πρέπει να αναφέρουμε ότι για την σωστή λειτουργία των νημάτων στα Sun SPOTs, πρέπει να αναθέτουμε προτεραιότητα χαμηλότερη από 5 (Thread.NORMAL) όταν οι εφαρμογές μας είναι απαιτητικές σε επεξεργαστική ισχύ καθώς υψηλές προτεραιότητες μπορεί να προκαλέσουν προβλήματα σε νήματα στης βιβλιοθήκης των Sun SPOTs.

4.1. Ο κύκλος ζωής ενός MIDlet

Όταν θέλουμε να υλοποιήσουμε μια εφαρμογή για τα Sun SPOTs αυτή πρέπει οπωσδήποτε να υπακούει στο πρότυπο MIDlet που αναφέραμε προηγουμένως. Συγκεκριμένα, όλες οι εφαρμογές που υλοποιούμε για τα Sun SPOTs πρέπει να κληρονομούν (extends) τα στοιχεία της κλάσης MIDlet και να υλοποιούν τις εξής μεθόδους [3]:

- *startApp()*
  Η μέθοδος αυτή καλείται όταν πρόκειται να εκτελεστεί το MIDlet.

- *pauseApp()*
  Η μέθοδος αυτή καλείται όταν πρόκειται να ανασταλεί η εκτέλεση του MIDlet.
- *destroyApp()*
  Η μέθοδος αυτή καλείται όταν το MIDlet τερματίζεται από το σύστημα.

Για τον τερματισμό ενός MIDlet θα πρέπει πάντα να χρησιμοποιείται η μέθοδος *notifyDestroyed()*, ενώ οι εφαρμογές δεν πρέπει ποτέ να καλούν την μέθοδο System.exit().

4.1.1. Manifest και resources

Η δομή των φακέλων όταν αναπτύσσουμε μια εφαρμογή για τα SPOT πρέπει να έχουν συγκεκριμένη δομή. Στο φάκελο κάθε εφαρμογής πρέπει να υπάρχουν 2 αρχεία, ένα αρχείο build.xml και ένα build.properties που χρησιμοποιούνται από τα ant scripts για την μετάφραση και την εκτέλεση των MIDlet. Επιπλέον πρέπει να έχουμε και 3 υποφακέλους, έναν με όνομα src που τοποθετούμε τους καταλόγους με τα αρχεία πηγαίου κώδικα και έναν με τον όνομα resources/META-INF.

Στο φάκελο resources/META-INF υπάρχει το αρχείο MANIFEST.MF που περιέχει πληροφορίες που χρησιμοποιούνται από την Squawk VM για την εκκίνηση των εφαρμογών. Συγκεκριμένα περιέχει τα όνομα των αρχικών κλάσεων των MIDlets και ορισμένες ιδιότητες αυτών. Επιπλέον ο φάκελος μπορεί να περιλαμβάνει αρχεία που ορίζει ο χρήστης και είναι διαθέσιμα στην εφαρμογή κατά την εκτέλεση της.

Η δομή ενός τυπικού MANIFEST.MF είναι [3]:

```
MIDlet-Name: Air Text demo
MIDlet-Version: 1.0.0
MIDlet-Vendor: Sun Microsystems Inc
MIDlet-1: AirText, , org.sunspotworld.demo.AirTextDemo
MicroEdition-Profile: IMP-1.0
MicroEdition-Configuration: CLDC-1.1
SomeProperty: some value
```

Εικόνα 4.1. Η δομή ενός τυπικού MANIFEST.MF [3]

Η σύνταξη κάθε γραμμής ακολουθεί το πρότυπο: <property-name>:<space><property-value>.

Η πιο σημαντική γραμμή για κάθε πρόγραμμα είναι η: MIDlet-1:<όρισμα 1>,<όρισμα 2>,<όρισμα 3>.

Το πρώτο όρισμα είναι μια περιγραφή της εφαρμογής, το δεύτερο όρισμα είναι μια εικόνα που σχετίζεται με το MIDlet, αλλά στην παρούσα έκδοση δεν υποστηρίζεται αυτή η επιλογή και το τρίτο όρισμα είναι η κύρια κλάση του MIDlet. Μέσα από την εφαρμογή μπορούμε να διαβάσουμε τις τιμές για τις παραπάνω ιδιότητες χρησιμοποιώντας την μέθοδο:

<όνομα MIDlet>.getAppProperty("<property-name>").

Το manifest μπορεί να καθορίσει πολλά MIDlets (MIDlet-1, MIDlet-2, κ.λπ.), με το καθένα να εχει τη δική του κύρια κλάση. Το manifest στο σύνολο του καθορίζει μια σουίτα MIDlet το όνομα του οποίου καθορίζεται από την μεταβλητή MIDlet-Name.

Μια παρενέργεια της εντολής deploy είναι να αλλάξει τη διαμόρφωση του Sun SPOT ώστε η φορτωμένη εφαρμογή να εκτελεστεί κατά την επόμενη επανεκκίνηση του Sun SPOT. Εξορισμού, το προεπιλεγμένο MIDlet που θα εκτελεστεί είναι το MIDlet-1. Θα πορούσαμε να καθορίσουμε ένα διαφορετικό MIDlet κατά το φόρτωμα με την εντολή:

ant deploy –Dmidlet=2

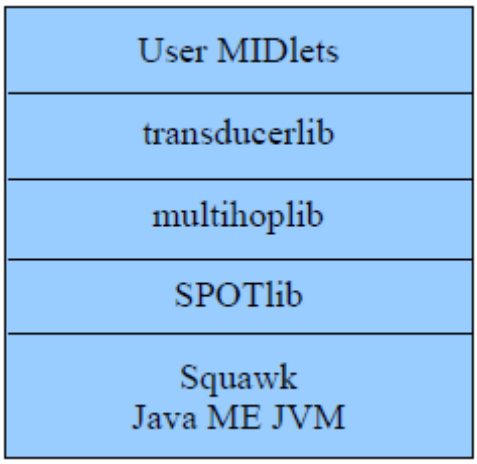
Όλα τα αρχεία μέσα στον φάκελο resources είναι διαθέσιμα στην εφαρμογή κατά την διάρκεια που εκτελείται. Μπορούμε να ανοίξουμε ένα stream εισόδου για να διαβάσουμε τα περιεχόμενά τους μέσω της μεθόδου:

InputStream is = getClass().getResourceAsStream("/res1.txt ").

4.2. Η αρχιτεκτονική του συστήματος των Sun SPOTS

Sun SPOT

Στο παρακάτω διάγραμμα μπορούμε να δούμε μια απεικόνιση της στοίβας του λογισμικού που εκτελείται σε ένα Spot που δεν είναι σταθμός βάσης.



Εικόνα 4.2. Η στοίβα λογισμικού ενός Sun SPOT [3]

Στην κορυφή βρίσκεται η εφαρμογή του χρήστη, η οποία επεκτείνει την Java ME MIDlet κλάση. Στο κάτω μέρος βρίσκεται η Sqwawk VM. Δεν υπάρχει κάποιο λειτουργικό σύστημα. Η Sqwawk VM «τρέχει» απευθείας πάνω στο hardware. Στα ενδιάμεσα στρώματα βρίσκονται οι διάφορες βιβλιοθήκες του Sun SPOT.

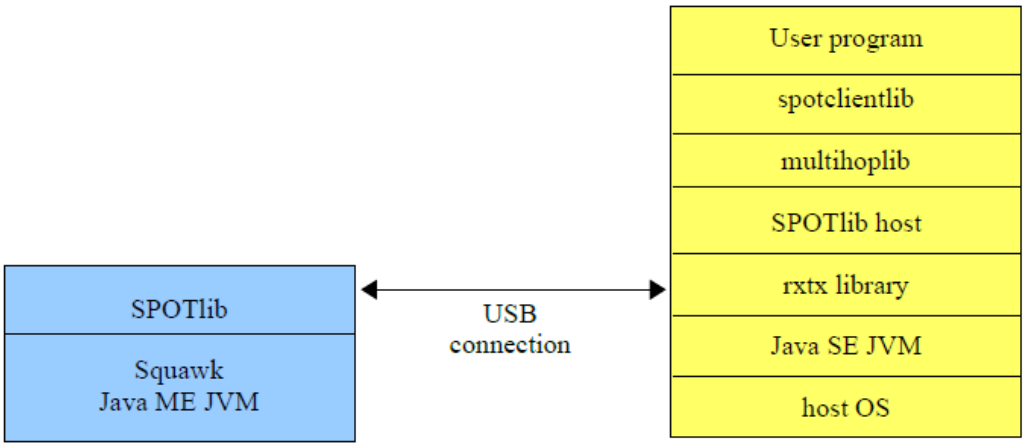
Η πρόσβαση στη συσκευή του Sun SPOT και οι βασικές βιβλιοθήκες εισόδου / εξόδου παρέχονται από τη βιβλιοθήκη SPOTlib. Η βιβλιοθήκη αυτή περιλαμβάνει πρόσβαση σε χαμηλού επιπέδου πρωτόκολλα επικοινωνίας όπως το MAC.

Η βιβλιοθήκη multihoplib παρέχει υψηλότερου επιπέδου πρωτόκολλα επικοινωνίας, όπως είναι για παράδειγμα το Radiostream, καθώς επίσης φροντίζει για την δρομολόγηση πακέτων σε άλλα SPOT που δε βρίσκονται στην άμεση εμβέλεια.

Η βιβλιοθήκη transducerlib παρέχει όλες τις κλάσεις που χρειαζόμαστε για να αποκτήσουμε πρόσβαση στο πίνακα αισθητήρων του eDemo, όπως π.χ. το επιταχυνσιόμετρο, τον αισθητήρα θερμοκρασίας, τα διάφορα LEDs, τους διακόπτες κ.λπ.

Host εφαρμογή Sun SPOT

Στο παρακάτω διάγραμμα μπορούμε να δούμε την απεικόνιση της στοίβας λογισμικού μιας εφαρμογής που εκτελείται σε έναν υπολογιστή και πως αυτός συνδέεται με ένα SPOT σταθμό βάσης.



Εικόνα 4.3. Η αρχιτεκτονική μιας host εφαρμογής Sun SPOT [3]

Στη κορυφή της στοίβας βρίσκεται και πάλι η εφαρμογή του χρήστη η οποία ακολουθεί τα πρότυπα ενός κοινού Java SE προγράμματος. Η εφαρμογή αυτή μπορεί να εκτελεί ότι και ένα κοινό πρόγραμμα Java, όπως να διαβάζει και να γράφει αρχεία, να υλοποιεί γραφικό περιβάλλον GUI κτλ. Η κυριότερη λειτουργία του είναι το γεγονός ότι μπορεί να στέλνει και να λαμβάνει δεδομένα από τα Spots με τη προϋπόθεση ότι στον υπολογιστή υπάρχει συνδεδεμένος σταθμός βάσης.

Στο κάτω μέρος της στοίβας βρίσκεται το λειτουργικό σύστημα στο οποίο εκτελείται η εφαρμογή του χρήστη, π.χ. Linux, Windows, Mac OS X ή Solaris.

Ακριβώς πάνω από το λειτουργικό βρίσκεται η εικονική μηχανή της Java SE (JVM) μαζί με όλες τις κοινές βιβλιοθήκες της Java. Στα ενδιάμεσα στρώματα βρίσκονται οι διάφορες βιβλιοθήκες των Sun SPOTs.

Η πρόσβαση στα SPOT και οι βασικές λειτουργίες εισόδου / εξόδου παρέχονται από τη host έκδοση της βιβλιοθήκης SPOTlib. Αυτή περιλαμβάνει πρόσβαση στο χαμηλού επιπέδου πρωτόκολλο επικοινωνίας MAC και είτε χρησιμοποιεί τη σύνδεση μέσω USB για τη πρόσβαση στο ράδιο του σταθμού βάσης είτε χρησιμοποιεί socket συνδέσεις για την επικοινωνία με άλλες host εφαρμογές του χρήστη.

Η βιβλιοθήκη multihoplib παρέχει και πάλι υψηλότερου επιπέδου πρωτόκολλα επικοινωνίας, όπως για παράδειγμα το Radiostream και το Radiogram, ενώ φροντίζει επίσης για τη δρομολόγηση των πακέτων.

Η βιβλιοθήκη spotclientlib δίνει πρόσβαση σε μια σειρά από εντολές OTA (Over-the-air) που μπορούν να σταλούν στα Sun SPOTs. Όπως για παράδειγμα η "Hello" εντολή που χρησιμοποιείται για να ανακαλύψουμε τα Sun SPOTs που βρίσκονται εντός της εμβέλειας του δικτύου.

Η βιβλιοθήκη rxtx χρησιμοποιείται για την σειριακή επικοινωνία με τον σταθμό βάσης μέσω της σύνδεσης USB.

Sun SPOT σταθμός βάσης

Ο σταθμός βάσης επιτρέπει στις εφαρμογές που εκτελούνται σε έναν υπολογιστή να επικοινωνούν με τα Sun SPOTs χρησιμοποιώντας το ραδιοδίκτυο του σταθμού βάσης.

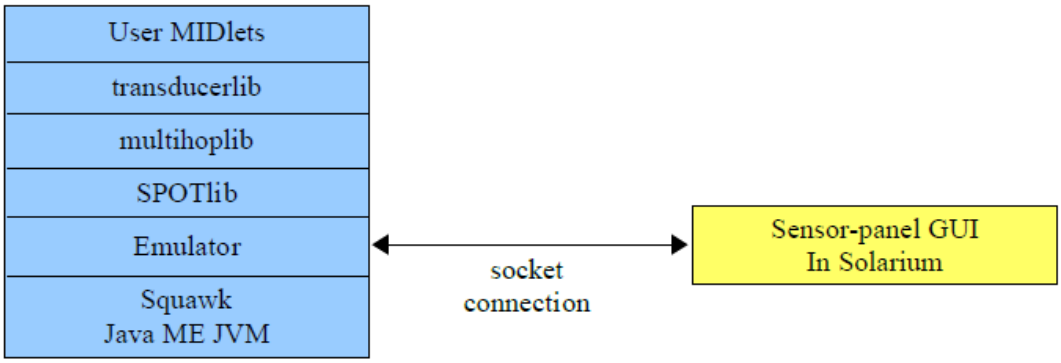
Θα πρέπει να τονίσουμε ότι η host εφαρμογή εκτελείται αποκλειστικά στον υπολογιστή και κανένας κώδικας της εφαρμογής δεν τρέχει στο σταθμό βάσης. Τα πακέτα που θέλουμε να στείλουμε στα Sun SPOTs στέλνονται πρώτα μέσω USB στο σταθμό βάσης, ο οποίος τα στέλνει στη συνέχεια στο δίκτυο μέσω του ραδιοδικτύου του. Ακριβώς το ίδιο συμβαίνει και στην αντίθετη περίπτωση, όταν ο σταθμού βάσης λάβει ένα πακέτο από το δίκτυο το προωθεί στην εφαρμογή του χρήστη.

Εικονικό Sun SPOT

Μέσα από την εφαρμογή Spot Manager από όπου μπορούμε να διαχειριστούμε εύκολα τα Spot μας και τα εγκατεστημένα SDK βρίσκουμε μια πολύ χρήσιμη εφαρμογή, το Solarium. Η εφαρμογή αυτή μας επιτρέπει να δοκιμάζουμε τοις διάφορες εφαρμογές μας για τα Spot σε εικονικά Spot τα οποία εξομοιώνουν τις πραγματικές συσκευές.

Όταν δημιουργούμε ένα νέο εικονικό SPOT στο Solarium, ξεκινά μια νέα διεργασία οποία τρέχει τον κώδικα στη «Squawk VM». Ο κώδικας επικοινωνεί μέσω μιας socket σύνδεσης με το κώδικα GUI του εικονικού Sun SPOT. Έτσι, με αυτό το τρόπο μπορεί η εφαρμογή να επικοινωνεί με το εικονικό Spot και αυτό να ανταποκρίνεται στις εντολές της εφαρμογής.

Στο παρακάτω διάγραμμα μπορούμε να δούμε μια απεικόνιση της στοίβας ενός εικονικού Sun SPOT.



Εικόνα 4.4. Η στοίβα λογισμικού ενός εικονικού Sun SPOT [3]

Κάθε εικονικό Sun SPOT έχει τη δική του «Squawk VM» η οποία εκτελείται σε ξεχωριστή διεργασία στον κεντρικό υπολογιστή.

4.3. Οι βιβλιοθήκες των Sun SPOTs

4.3.1. Βιβλιοθήκη συσκευών

Η βιβλιοθήκη των συσκευών βρίσκεται στα αρχεία spotlib_device.jar και spotlib_common.jar. Ο πηγαίος κώδικας της βιβλιοθήκης (των δυο παραπάνω jar) βρίσκεται στο spolib_source.jar στο φάκελο “Sun\SunSPOT\sdk\src“ του SDK και περιέχει οδηγούς (drivers) για τις παρακάτω συσκευές:

- On-board LEDs
- Τους διαύλους PIO, USART και τα ρολόγια/μετρητές του επεξεργαστή
- Τον πομποδέκτη CC2420
- Το SPI interface
- Την ενσωματωμένη μνήμη flash

Επίσης, περιλαμβάνει κλάσεις για τον χειρισμό του σταθμού βάσης (com.sun.spot.peripheral.basestation), κλάσεις για έλεγχο των Sun SPOTs μέσω του σταθμού βάσης, OTA (Over The Air), ένα framework για την επικοινωνία των εφαρμογών που εκτελούνται σε διαφορετικά isolates, και ένα handler για το «attention» κουμπί.

Κάθε στοιχείο και υποσύστημα ελέγχεται από τους παραπάνω drivers, ενώ στον προγραμματιστή δίνεται πρόσβαση στα υποσυστήματα μέσω των παρακάτω διεπαφών:

Device	Interface
LED	ILed
PIO	IAT91_PIO
AIC	IAT91_AIC
Timer-Counter	IAT91_TC
CC2420	I802_15_4_PHY
MAC layer	I802_15_4_MAC
RadioPolicyManager	IRadioPolicyManager
SPI	ISpiMaster
Flash memory	IFlashMemoryDevice
Power controller	IPowerController
OTA Command Server	OTACommandServer (class)
Attention button handler	FiqInterruptDaemon (class)

Εικόνα 4.5. Τα υποσυστήματα ενός Sun SPOT και οι διεπαφές τους [3]

Οι κλάσεις που υλοποιούν τις παραπάνω διεπαφές δημιουργούνται από ένα αντικείμενο της κλάσης Spot που είναι υπεύθυνο για την δημιουργία των drivers και τη διαχείριση της πρόσβασης σε αυτούς. Για παράδειγμα, για να επιλέξουμε το πράσινο LED της κύριας μονάδας του Spot χρησιμοποιούμε τον ακόλουθο κώδικα:

```
Iled theLed = Spot.getInstance().getGreenLed();
```

Με τις παρακάτω εντολές ενεργοποιούμε και απενεργοποιούμε το LED αντίστοιχα:

```
theLed.setOn();
theLed.setOff();
```

4.3.2. Βιβλιοθήκη αισθητήρων

Πρόκειται για την βιβλιοθήκη που περιέχει όλες τις κλάσεις και τις διεπαφές που χρειαζόμαστε για την διαχείριση των συστημάτων και των αισθητήρων του eDEMO Board και βρίσκεται στο αρχείο transducerlib_rt.jar. Με τις κλάσεις αυτές μπορούμε να χειριστούμε πολύ εύκολα το επιταχυνσιόμετρο, τα LED, τον αισθητήρα φωτός, τον αισθητήρα θερμοκρασίας, τις ψηφιακές θύρες εισόδου/εξόδου καθώς και τους διακόπτες.

4.3.3. Βιβλιοθήκη συνδέσεων και πρωτοκόλλων επικοινωνίας

Πρόκειται για την βιβλιοθήκη που είναι υπεύθυνη για την δημιουργία και τον έλεγχο όλων των συνδέσεων και πρωτοκόλλων των SUN SPOT. Οι κλάσεις που υλοποιούν τμήματα της radio στοίβας πάνω από το στρώμα MAC βρίσκονται στη βιβλιοθήκη multihoplib_rt.jar ο αντίστοιχος πηγαίος κώδικας στη βιβλιοθήκη multihoplib_source.jar. Η τρέχουσα έκδοση του SUN SPOT SDK χρησιμοποιεί το GCF (Generic Connection Framework) για την δημιουργία συνδέσεων και την ασύρματη επικοινωνία μεταξύ των SPOTs χρησιμοποιώντας δυο πρωτόκολλα επικοινωνίας [3]:

- Radiostream
  Το radiostream παρέχει αξιόπιστη μετάδοση δεδομένων μεταξύ δυο κόμβων, ενώ επίσης χρησιμοποιεί buffers για καλύτερη απόδοση. Το πρωτόκολλο αυτό βασίζεται στην επικοινωνία μέσω streams.
- Radiogram

Στο radiogram πρωτόκολλο η μεταφορά δεδομένων γίνεται με datagrams και το πρωτόκολλο αυτό δεν παρέχει καμία εγγύηση ότι τα πακέτα θα παραλειφθούν σωστά ή ότι θα φτάσουν στον προορισμό με την σειρά που στάλθηκαν. Όταν ένα πακέτο στέλνεται μέσω άλλων κόμβων (δηλ. σε περισσότερα του ενός hop) υπάρχει περίπτωση:

- να χαθεί χωρίς καμία ειδοποίηση
- να παραλειφθεί από τον προορισμό περισσότερες από μια φορές
- να μην φτάσει με την σειρά που στάλθηκε.

Αντίθετα, στην περίπτωση που τα datagrams στέλνονται σε γειτονικό κόμβο (ένα hop) η μόνη δυσλειτουργία είναι κάποια να παραλειφθούν περισσότερες από μια φορές.

Το πρωτόκολλο Radiostream

Το πρωτόκολλο radiostream είναι ένα peer-to-peer πρωτόκολλο επικοινωνίας που παρέχει αξιόπιστη επικοινωνία μεταξύ δύο κόμβων με τη μορφή streams για την αποστολή και τη λήψη της πληροφορίας. Για να ανοίξουμε μια τέτοια σύνδεση γράφουμε στον κώδικα μας:

RadiostreamConnection conn =

(RadiostreamConnection)Connector.open("radiostream:<destAddr>:<portNo>");

όπου:

- destAddr: είναι η 64bit διεύθυνση IEEE του Spot που βρίσκεται στην άλλη άκρη,
- portNo: ο αριθμός της θύρας (με εύρος τιμών από 0 έως 255)

Η σύνδεση ανοίγει χρησιμοποιώντας τις προεπιλεγμένες ιδιότητες δικτύου εκτός και αν έχουμε φροντίσει στον κώδικα μας να τις αλλάξουμε. Για να δημιουργήσουμε μια σύνδεση πρέπει και τα δύο άκρα να ανοίξουν συνδέσεις προσδιορίζοντας τον ίδιο αριθμό θύρας και τις αντίστοιχες IEEE διευθύνσεις των Sun SPOTs. Μόλις η σύνδεση ανοίξει, κάθε τέλος μπορεί να δημιουργήσει streams για την αποστολή και λήψη δεδομένων ορίζοντας δυο τέτοια αντικείμενα:

DataInputStream dis = conn.openDataInputStream();

DataOutputStream dos = conn.openDataOutputStream();

Παρακάτω παραθέτουμε ένα ενδεικτικό ολοκληρωμένο παράδειγμα που γίνεται χρήση αυτού του πρωτοκόλλου:

Πρόγραμμα 1

```
RadiostreamConnection conn =
(RadiostreamConnection)Connector.open("radiostream://0014.4F01.0000.0006:100");

DataInputStream dis = conn.openDataInputStream();
DataOutputStream dos = conn.openDataOutputStream();

try {
        dos.writeUTF("Hello up there");
        dos.flush();
        System.out.println ("Answer was: " + dis.readUTF());
} catch (NoRouteException e) {
        System.out.println ("No route to 0014.4F01.0000.0006");
} finally {
        dis.close();
```

```
            dos.close();
            conn.close();
    }
```

Πρόγραμμα 2

```
    RadiostreamConnection conn =
    (RadiostreamConnection)Connector.open("radiostream://0014.4F01.0000.0007:100");

    DataInputStream dis = conn.openDataInputStream();
    DataOutputStream dos = conn.openDataOutputStream();

    try {
            String question = dis.readUTF();
            if (question.equals("Hello up there")) {
                    dos.writeUTF("Hello down there");
            } else {
                    dos.writeUTF("What???");
            }
            dos.flush();
    } catch (NoRouteException e) {
            System.out.println ("No route to 0014.4F01.0000.0007");
    } finally {
            dis.close();
            dos.close();
            conn.close();
    }
```

Σε αυτό το παράδειγμα τα δυο προγράμματα ανοίγουν μια σύνδεση τύπου radiostream και ορίζουν τα αντίστοιχα stream εισόδου/εξόδου. Μετά, το πρόγραμμα 2 αναμένει να λάβει δεδομένα από το stream εισόδου ενώ το πρόγραμμα 1 στέλνει το μήνυμα "Hello up there". Αν το μήνυμα παραλειφθεί σωστά, το πρόγραμμα 2 απαντά “Hello down there” και η απάντηση αποτυπώνεται στην οθόνη από το πρόγραμμα 1.

Τα δεδομένα στέλνονται ασύρματα όταν γεμίσει το buffer του radiostream ενώ αν θέλουμε να σταλούν τα δεδομένα άμεσα πρέπει να χρησιμοποιήσουμε την εντολή flush(). Επίσης, αν το πρόγραμμα 1 ξεκινήσει πριν το πρόγραμμα 2 τότε υπάρχει περίπτωση να χαθεί το μήνυμα "Hello up there", αφού η αποστολή θα γίνει πριν το πρόγραμμα 2 ζητήσει δεδομένα. Για να είμαστε σίγουροι ότι θα εκτελεστούν με την σωστή σειρά θα μπορούσαμε να εφαρμόσουμε κάποιο είδος handshake ή να εισάγουμε μια καθυστέρηση πριν το πρόγραμμα 2 στείλει το μήνυμα. Στη περίπτωση που δεν μπορεί να βρεθεί μια διαδρομή προς τον προορισμό τότε παράγεται μια εξαίρεση τύπου NoRouteException.

Σε κάθε μετάδοση ενός πακέτου το στρώμα MAC περιμένει την αποστολή ενός πακέτου ACK (πακέτο επιβεβαίωσης), που δηλώνει την επιτυχή λήψη του πακέτου. Σε περίπτωση που ο τελικός προορισμός βρίσκεται σε απόσταση ενός hop, η επιβεβαίωση σε MAC επίπεδο είναι αρκετή για να διασφαλίσουμε ότι έγινε σωστά η λήψη. Σε διαφορετική περίπτωση το

radiostream θα ζητήσει από τον προορισμό να στείλει ένα πακέτο επιβεβαίωσης πίσω στον αρχικό αποστολέα. Για την βελτίωση της απόδοσης του πρωτοκόλλου όταν στέλνουμε δεδομένα, το stream δεν περιμένει την επιβεβαίωση από τον προορισμό αλλά επιστρέφει άμεσα. Σε περίπτωση που δεν παραλάβουμε το πακέτο επιβεβαίωσης σε κάποιο προκαθορισμένο χρονικό διάστημα τότε ξαναστέλνεται αυτόματα. Μετά από έναν αριθμό αποτυχημένων προσπαθειών προκαλείται μια εξαίρεση τύπου NoMeshLayerAckException στην επόμενη προσπάθεια για αποστολή δεδομένων. Τέλος μια άλλη εξαίρεση που μπορεί να προκληθεί και στα δυο πρωτόκολλα επικοινωνίας είναι η εξαίρεση τύπου ChannelBusyException. Αυτή η εξαίρεση είναι ένδειξη ότι το κανάλι επικοινωνίας είναι απασχολημένο, δηλαδή ότι μεταδίδουν ταυτόχρονα και άλλες συσκευές.

Το πρωτόκολλο Radiogram

Το πρωτόκολλο radiogram ακολουθεί το μοντέλο επικοινωνίας client-server και παρέχει επικοινωνία μεταξύ δυο Sun SPOTs μέσω datagrams. Για να «ανοίξουμε» μια σύνδεση από την πλευρά του server χρησιμοποιούμε τον ακόλουθο κώδικα:

```
RadiogramConnection conn =
        (RadiogramConnection) Connector.open("radiogram://:<portNo>");
```

Η παράμετρος portNo είναι ο αριθμός της θύρας με τιμές από 0 ως 255 που χαρακτηρίζει μοναδικά την συγκεκριμένη σύνδεση. Μπορούμε να επιλέξουμε οποιοδήποτε αριθμό θύρας θέλουμε από την παραπάνω περιοχή εκτός των θυρών 1 ως 31 που είναι δεσμευμένα για χρήση από το σύστημα.

Για να «ανοίξουμε» μια σύνδεση από την πλευρά του client χρησιμοποιούμε τον ακόλουθο κώδικα:

```
RadiogramConnection conn =
    RadiogramConnection)Connector.open("radiogram://<serveraddr>:<portNo>");
```

Όπου:

destinationAddr: είναι η 64-bit διεύθυνση του server και

portNo: είναι ο αριθμός της θύρας της radiogram σύνδεσης που έχει ανοίξει ο server και που θέλουμε να συνδεθούμε.

Τα δεδομένα μεταξύ του client και του server στέλνονται μέσω datagrams. Προκειμένου να στείλουμε δεδομένα πρέπει να κατασκευάσουμε ένα αντικείμενο τύπου datagram από τη σύνδεση που έχουμε ανοίξει, να γράψουμε σε αυτό τα δεδομένα που θέλουμε και μετά να το αποστείλουμε στον προορισμό μέσω της σύνδεσης.

Παρακάτω παραθέτουμε ένα παράδειγμα που έχει την ίδια λειτουργία με τα προγράμματα 1 και 2 του πρωτόκολλου radiostream:

Κώδικας από την πλευρά του Client

```
RadiogramConnection conn =

(RadiogramConnection)Connector.open("radiogram://0014.4F01.0000.0006:100");

Datagram dg = conn.newDatagram(conn.getMaximumLength());

try {
        dg.writeUTF("Hello up there");
        conn.send(dg);
        conn.receive(dg);
```

```
            System.out.println ("Received: " + dg.readUTF());
        } catch (NoRouteException e) {
            System.out.println ("No route to 0014.4F01.0000.0006");
        } finally {
            conn.close();
        }
```

Κώδικας από την πλευρά του Server

```
        RadiogramConnection conn =
        (RadiogramConnection)Connector.open("radiogram://:100");

        Datagram dg = conn.newDatagram(conn.getMaximumLength());
        Datagram dgreply = conn.newDatagram(conn.getMaximumLength());

        try {
            conn.receive(dg);
            String question = dg.readUTF();
            dgreply.reset(); // reset stream pointer
            dgreply.setAddress(dg); // copy reply address from input
            if (question.equals("Hello up there")) {
            dgreply.writeUTF("Hello down there");
            } else {
                dgreply.writeUTF("What???");
            }
            conn.send(dgreply);
        } catch (NoRouteException e) {
            System.out.println ("No route to " + dgreply.getAddress());
        } finally {
            conn.close();
        }
```

Ορισμένα σημαντικά χαρακτηριστικά των datagrams που θα πρέπει να αναφέρουμε είναι:

- Τα datagrams που στέλνονται μέσω μιας datagram σύνδεσης πρέπει να έχουν κατασκευαστεί από αυτό, με τη μέθοδο newDatagram() της ανοιχτής σύνδεσης. Δεν μπορούμε να αποστείλουμε datagrams σε μια σύνδεση, όταν αυτά έχουν προέλθει από μια άλλη.
- Μια σύνδεση datagram που έχει ανοιχτεί για μια συγκεκριμένη διεύθυνση δεν μπορεί να χρησιμοποιηθεί για να σταλούν πακέτα και σε άλλο προορισμό. Η αποστολή του datagram σε διαφορετικό προορισμό θα προκαλούσε εξαίρεση στην εφαρμογή κατά την εκτέλεση του κώδικα.
- Είναι επίσης δυνατόν και αρκετά χρήσιμο όταν υλοποιούμε εφαρμογές να έχουμε στο ίδιο Sun SPOT ανοιχτές δυο συνδέσεις: μια server και μια client στην ίδια θύρα. Τα datagrams που λαμβάνονται και προορίζονται για την συγκεκριμένη θύρα προωθούνται αυτομάτως στη σύνδεση server.

- Επίσης, τέλος θα πρέπει να αναφέρουμε ότι στην τρέχουσα υλοποίηση του πρωτοκόλλου στα Sun SPOTs όταν ο server κλείσει τη σύνδεση, τότε κλείνει αυτόματα και η σύνδεση του client.

Ρύθμιση ιδιοτήτων σύνδεσης

Οι κλάσεις RadiostreamConnection και RadiogramConnection παρέχουν επίσης μεθόδους για τον καθορισμό ενός χρονικού ορίου (σε ms) για την αποδοχή και αποστολή των μηνυμάτων. Για παράδειγμα:

```
RadiostreamConnection conn =
(RadiostreamConnection) Connector.open ("radiostream://0014.4F01.0000.0006: 100");
conn.setTimeout (1000);  // 1000ms χρόνος αναμονής για λήψη μηνύματος
```

Υπάρχουν ορισμένα σημαντικά σημεία που πρέπει να σημειώσουμε σχετικά με τη χρήση αυτής της παραμέτρου:

- Μια εξαίρεση τύπου TimeoutException παράγεται εάν παρέλθει το χρονικό διάστημα που έχουμε ορίσει και δεν έχει παραληφθεί κάποιο μήνυμα.
- Μια τιμή 0 ως χρονικό όριο θα είχε ως αποτέλεσμα η εξαίρεση να παράγεται άμεσα αν δεν υπάρχουν διαθέσιμα δεδομένα.
- Αν δώσουμε τιμή -1 τότε απενεργοποιείται ο χρονικός περιορισμός με αποτέλεσμα η σύνδεση να περιμένει για πάντα μέχρι να λάβει δεδομένα.

Ευρεία Εκπομπή μηνυμάτων (Broadcast)

Το πρωτόκολλο radiogram μπορεί να χρησιμοποιηθεί και για την αποστολή broadcast πακέτων. Η προκαθορισμένη συμπεριφορά είναι να αποστέλλονται τα broadcast πακέτα σε ακτίνα 2 hops για να μπορούν να λαμβάνονται από κοντινές συσκευές που όμως βρίσκονται εκτός ακτίνας μετάδοσης. Ο κώδικας για την αποστολή broadcast πακέτων είναι ο εξής:

```
DatagramConnection conn =(DatagramConnection)Connector.open("radiogram://broadcast:");
```

Για την λήψη των broadcasts πακέτων δεν μπορεί να χρησιμοποιηθεί η παραπάνω σύνδεση αλλά θα πρέπει να ορίσουμε μια νέα client σύνδεση ως εξής:

```
RadiogramConnection conn = (RadiogramConnection)Connector.open("radiogram://:");
```

Αν θέλουμε να τροποποιήσουμε τον αριθμό των hops που θα μεταβεί το broadcast μήνυμα τότε μπορούμε να χρησιμοποιήσουμε την παρακάτω μέθοδο στο “ανοιχτό” connection θέτοντας το n στον αριθμό των hops που θέλουμε :

```
((RadiogramConnection)conn).setMaxBroadcastHops(n);
```

Θα πρέπει να προσθέσουμε ότι ο μηχανισμός μετάδοσης broadcast δεν παρέχει καμία εγγύηση για σωστή παραλαβή, αφού δεν χρησιμοποιεί πακέτα επιβεβαίωσης, ούτε πραγματοποιούνται αναμεταδόσεις broadcast πακέτων. Στην περίπτωση που το μέγεθος των broadcast πακέτων είναι μεγάλο, τότε αυτά στέλνονται σε τμήματα (fragments) και αυξάνεται η πιθανότητα να συμβεί κάποιο σφάλμα σε τουλάχιστον ένα από αυτά. Το μέγιστο μέγεθος δεδομένων των broadcast πακέτων που μπορούμε να στείλουμε είναι 1260 bytes, ενώ το μέγιστο μέγεθος κάθε πακέτου του 802.15.4 radio ξεχωριστά είναι περίπου 100 bytes. Στην περίπτωση που στέλνονται πολλά fragments τα SPOTs αντιμετωπίζουν και ένα επιπλέον πρόβλημα που οδηγεί σχεδόν σίγουρα στην απώλεια δεδομένων. Το πρόβλημα είναι ότι η συσκευή που δέχεται τα πακέτα έχει ένα περιορισμό στο πόσο γρήγορα μπορεί να αδειάζει τον packet buffer, κάτι που μπορεί να επιδεινώνεται σε περιπτώσεις που δέκτης έχει μεγάλο αριθμό από ενεργά νήματα ή τυχαίνει να καλεί συχνά τον gc (garbage collector). Για αυτό τον λόγο προτείνεται να μην στέλνουμε broadcast radiograms με περισσότερα από 200 bytes δεδομένων, δηλαδή το πολύ δύο radio packets για κάθε broadcast και να εισάγουμε μια μικρή καθυστέρηση μεταξύ διαδοχικών broadcast ώστε να προλαβαίνει ο δέκτης να τα παραλαμβάνει.

Χρήση αριθμού θυρών

Το έγκυρο εύρος τιμών των θυρών που μπορούμε να χρησιμοποιήσουμε στα πρωτόκολλα επικοινωνίας των Sun SPOTs είναι από 0 έως 255. Ωστόσο, για κάθε πρωτόκολλο, οι τιμές που βρίσκονται στην περιοχή από 0 έως 31 προορίζονται αποκλειστικά για χρήση του συστήματος. Οι εφαρμογές που υλοποιούμε δεν πρέπει να χρησιμοποιούν αριθμούς θυρών που βρίσκονται σε αυτήν την περιοχή.

Στον παρακάτω πίνακα μπορούμε να δούμε τους αριθμούς των θυρών που χρησιμοποιούν διάφορα πρωτόκολλα του συστήματος.

Port number	Protocol	Usage
8	radiogram://	OTA Command Server
9	radiostream://	Debugging
10	radiogram://	http proxy
11	radiogram://	Manufacturing tests
12	radiostream://	Remote printing (Master isolate)
13	radiostream://	Remote printing (Child isolate)
14	radiogram://	Shared basestation discovery
20	radiogram://	Trace route server

Εικόνα 4.6. Αριθμοί θυρών και πρωτόκολλα του συστήματος [3]

Ιδιότητες του Δικτύου

Η κλάση RadioPolicyManager παρέχει λειτουργίες που μας δίνουν τη δυνατότητα να ορίζουμε και να τροποποιούμε χαρακτηριστικά του δικτύου όπως το κανάλι επικοινωνίας (channel number), το αναγνωριστικό PAN (Pan ID) και την ισχύ εκπομπής (output power). Για παράδειγμα:

```
IRadioPolicyManager rpm = Spot.getInstance (). getRadioPolicyManager ();
int currentChannel rpm.getChannelNumber = ();
σύντομο currentPan = rpm.getPanId ();
int currentPower rpm.getOutputPower = ();
rpm.setChannelNumber (11); / / έγκυρο εύρος τιμών: 11 - 26
rpm.setPanId ((short) 6);
rpm.setOutputPower (-31); / / έγκυρο εύρος τιμών: -32 έως 31
```

Η ισχύς του σήματος

Όταν χρησιμοποιούμε το radiogram πρωτόκολλο επικοινωνίας μπορούμε να λαμβάνουμε διάφορες μετρήσεις που αφορούν τη ποιότητα και την ισχύ του λαμβανομένου σήματος, όπως:

- RSSI (Received Signal Strength Indicator – Ένδειξη ισχύος λαμβανόμενου σήματος)

  Η ένδειξη αυτή μετρά τη δύναμη του σήματος για το πακέτο. Το RSSI κυμαίνεται από 60 (ισχυρό) έως -60 (αδύναμο).
- CORR

  Μετρά τη μέση τιμή συσχέτισης για τα πρώτα 4 Bytes της επικεφαλίδας του πακέτου. Μια τιμή συσχέτισης που κυμαίνεται κοντά στα 110 υποδηλώνει μέγιστη ποιότητα πακέτου, ενώ μια τιμή κοντά στα 50 είναι συνήθως η χαμηλότερη ποιότητα πακέτου που ανιχνεύεται από το δέκτη του Sun SPOT.
- LQI (Link Quality Indication - Ένδειξη ποιότητας γραμμής)

Είναι μια ένδειξη για το χαρακτηρισμό της ποιότητας του λαμβανόμενου πακέτου. Η τιμή του υπολογίζεται από την CORR, τιμή συσχέτισης. Το LQI κυμαίνεται από 0 (κακή ποιότητα) έως 255 (καλή ποιότητα).

Τις τιμές αυτές μπορούμε να τις λάβουμε προγραμματιστικά με τη χρήση των παρακάτω μεθόδων:

myRadiogram.getRssi ();

myRadiogram.getCorr ();

myRadiogram.getLinkQuality ();

4.4. Πολιτικές Δρομολόγησης

Όταν τα πακέτα δεδομένων μεταξύ των Sun SPOTs δρομολογούνται σε περισσότερα του ενός hop, τότε ένα πρωτόκολλο δρομολόγησης χρησιμοποιείται για την εύρεση της καλύτερης διαδρομής και την δρομολόγηση των πακέτων στους παραλήπτες.

Κάθε Sun SPOT μπορεί να ασκήσει συγκεκριμένο έλεγχο στο ρόλο που διαδραματίζει στη δρομολόγηση των πακέτων στο δίκτυο μέσω της κλάσης RoutingPolicyManager, η οποία εφαρμόζει τη διεπαφή IRoutingPolicyManager. Η πολιτική δρομολόγησης μπορεί να οριστεί είτε μέσω των ιδιοτήτων του συστήματος είτε να την ορίσουμε εμείς προγραμματιστικά μέσα στον κώδικα μας.

Οι τέσσερις διαθέσιμες πολιτικές δρομολόγησης είναι οι εξής [3]:

- ALWAYS

  Το Sun SPOT θα λειτουργεί πάντα ως πλήρης δρομολογητής. Θα ανταποκρίνεται και θα δρομολογεί τα RREQ πακέτα και θα προωθεί τα πακέτα των άλλων Sun SPOTs του δικτύου.

- IFAWAKE

  Το Sun SPOT θα λειτουργεί πάντα ως πλήρης δρομολογητής εφόσον δε βρίσκεται σε κατάσταση “Deep sleep”. Θα ανταποκρίνεται και θα δρομολογεί τα RREQ πακέτα και θα προωθεί τα πακέτα των άλλων Sun SPOTs του δικτύου.

- ENDNODE

  Το Sun SPOT θα δημιουργήσει RREQs πακέτα κατά το άνοιγμα μιας σύνδεσης. Θα ανταποκρίνεται σε αυτά αν και μόνο αν είναι ο τελικός προορισμός. Δεν θα επαναλαμβάνει τα RREQs πακέτα σε άλλα Sun SPOTs και δε θα προωθεί τα πακέτα άλλων Sun SPOTs εκτός και μόνο αν είναι είτε ο αποστολέας είτε ο τελικός παραλήπτης.

- SHAREDBASESTATION

  Αυτή η πολιτική δρομολόγησης προσωρινά δεν τίθεται σε εφαρμογή και συμπεριφέρεται όπως η πολιτική ALWAYS.

Η ιδιότητα *spot.mesh.route.enable* μπορεί να παίρνει τις εξής τιμές:

- ALWAYS
- ENDNODE
- IFAWAKE.

Μπορεί επίσης να ρυθμιστεί ώστε να έχουν εναλλακτικές τιμές που ελέγχει την προεπιλεγμένη πολιτική ως εξής:

Property value	Default policy
True	ALWAYS
false	ENDNODE
<not set>	IFAWAKE

Εικόνα 4.7. Ο πίνακας τιμών για την ιδιότητα spot.mesh.route.enable [3]

Μέσα στον κώδικα ορίζουμε προγραμματιστικά τη πολιτική δρομολόγησης με τις παρακάτω εντολές:

```
RoutingPolicy rp = new RoutingPolicy(RoutingPolicy.IFAWAKE);
RoutingPolicyManager.getInstance().policyHasChanged(rp);
```

4.5. Τα πρωτοκόλλα δρομολόγησης των Sun SPOTs

Το πρωτόκολλο AODV (Ad Hoc On Demand Distance Vector) είναι ένα distance vector, αντιδραστικό (κατ' απαίτηση) πρωτόκολλο δρομολόγησης κατάλληλο για ασύρματα ad-hoc δίκτυα γενικότερα. Το πρωτόκολλο αυτό επιτρέπει τη δυναμική, αυτοεκκινούμενη, multihop δρομολόγηση μεταξύ κινητών κόμβων που επιθυμούν να δημιουργήσουν ένα ad-hoc δίκτυο [22].

Οι διαδρομές προς τους προορισμούς δημιουργούνται μόνο όταν χρειάζεται όπως άλλωστε και σε κάθε αντιδραστικό (reactive) πρωτόκολλο. Η αντίδραση των κόμβων σε διακοπές ζεύξεων και αλλαγές στη τοπολογία του δικτύου είναι έγκαιρη, αφού στην περίπτωση που μια ζεύξη τεθεί εκτός λειτουργίας, το πρωτόκολλο ενημερώνει τους κόμβους που επηρεάζονται προκειμένου να ακυρώσουν τις διαδρομές που χρησιμοποιούν τη χαμένη ζεύξη. Με τον τρόπο αυτό επιτυγχάνεται η γρήγορη σύγκλιση σε περιπτώσεις αλλαγής της τοπολογίας του δικτύου.

Οι βρόχοι δρομολόγησης αποφεύγονται στον AODV με τη χρήση του αριθμού ακολουθίας προορισμού (destination sequence number) για κάθε αποθηκευμένη διαδρομή. Ο αριθμός ακολουθίας προορισμού δημιουργείται από τον κόμβο-προορισμό για να συμπεριληφθεί μαζί με τις υπόλοιπες πληροφορίες της διαδρομής που στέλνει στους κόμβους. Όταν δίνεται η δυνατότητα επιλογής μεταξύ δύο διαδρομών για έναν προορισμό, ο κόμβος που ζήτησε τη διαδρομή, επιλέγει εκείνη με τον μεγαλύτερο αριθμό ακολουθίας προορισμού.

Όταν ένας κόμβος επιχειρεί να βρει μια διαδρομή, εκπέμπει ένα μήνυμα αίτησης διαδρομής RREQ (Route Request) με συγκεκριμένο αύξοντα αριθμό πηγής προς όλους τους γειτονικούς κόμβους. Το RREQ διαδίδεται μέσα στο δίκτυο μέχρι να φτάσει στον προορισμό ή κάποιον κόμβο με μια πρόσφατη διαδρομή προς τον προορισμό. Οι ενδιάμεσοι κόμβοι που λαμβάνουν το μήνυμα αυτό εφόσον δεν έχουν διαδρομή προς τον προορισμό επανεκπέμπουν το μήνυμα, κρατώντας παράλληλα στον δικό τους πίνακα δρομολόγησης τη διαδρομή προς την πηγή. Η διαθέσιμη διαδρομή γίνεται γνωστή στην πηγή μέσω ενός μηνύματος RREP (Route Reply). Το μήνυμα αυτό εκπέμπεται τελικά προς την πηγή, σύμφωνα με τις διαδρομές που έχουν δημιουργηθεί κατά την μετάδοση του AODVrequest.

Το πρωτόκολλο AODV προϋπήρχε στις προηγούμενες εκδόσεις του Sun SPOT SDK και ήταν το προεπιλεγμένο πρωτόκολλο δρομολόγησης μέχρι και την $4^{η}$ έκδοση (BLUE) του Sun SPOT SDK. Η $5^{η}$ έκδοση (RED) του Sun SPOT SDK έχει ως προεπιλεγμένο πρωτόκολλο το LQRP (Link Quality Routing Protocol) το οποίο είναι «χτισμένο» πάνω στο προϋπάρχον AODV πρωτόκολλο.

Στο νέο πρωτόκολλο, τα αιτήματα για μια διαδρομή σε ένα συγκεκριμένο σημείο (RREQs) εκπέμπονται από τον αιτούντα, και στη συνέχεια μεταδίδονται εκ νέου από κάθε Sun SPOT που τις λαμβάνει. Κάθε Sun SPOT που γνωρίζει τη διαδρομή για το ζητούμενο σημείο στέλνει στη συνέχεια μια απάντηση στον αιτούντα. Η διαδρομή που θα χρησιμοποιηθεί τελικά θα είναι εκείνη με την καλύτερη ποιότητα σύνδεσης [3].

4.5.1. Συνοπτική ανάλυση κλάσεων και μεθόδων των πρωτόκολλων δρομολόγησης και επικοινωνίας των Sun SPOTS

Η κύρια κλάση AODVManager

Η δρομολόγηση των δεδομένων ρυθμίζεται από την κλάση AODVManager η οποία καθορίζει πως θα χρησιμοποιηθούν όλες οι κλάσεις που αφορούν τη δρομολόγηση. Οι σημαντικότερες μέθοδοι που μπορούμε να διακρίνουμε σε αυτή τη κλάση είναι:

- findRoute(long address, RouteEventClient eventClient, Object uniqueKey)
- int getCurrentSequenceNumber()
- RouteTable getRoutingTable()
- synchronized void initialize(long ourAddress, ILowPan lowPan)
- boolean initiateRouteDiscovery(long address)
- boolean invalidateRoute(long originator, long destination)
- int getStatus()

Ο AODV Manager βρίσκει τις ζητούμενες διαδρομές, αιτείται τον τρέχοντα αύξοντα αριθμό των πακέτων, αρχικοποιεί, ξεκινά και σταματά την υπηρεσία, ανακαλύπτει τις διαδρομές και ακυρώνει μια διαδρομή εάν δεν είναι έγκυρη.

Στο ίδιο πακέτο (*com.sun.spot.peripheral.radio.mhrp.aodv*) υπάρχουν επίσης οι κλάσεις του Αποστολέα (Sender) και του Παραλήπτη (Receiver). Η κυριότερη λειτουργία της κλάσης του παραλήπτη είναι να διαχειρίζεται τα μηνύματα του AODV πρωτοκόλλου (RREQ, RREP, RERR) και να επεξεργάζεται τα εισερχόμενα δεδομένα.

Οι σημαντικότερες μέθοδοι που μπορούμε να διακρίνουμε σε αυτή τη κλάση είναι:

- void handleRERRMessage(RERR message, long lastHop)
- void handleRREPMessage(RREP message, long lastHop)
- void handleRREQMessage(RREQ message, long lastHop)
- void processIncomingData(byte[] payload, LowPanHeaderInfo headerInfo)

Οι κυριότερες λειτουργίες του Αποστολέα είναι να προωθεί και να στέλνει τα AODV μηνύματα:

- boolean forwardAODVMessage(AODVMessage message)
- boolean sendNewRREP(RREP message)
- boolean sendNewRREQ(long address, RouteEventClient eventClient, Object uniqueKey)
- private void sendRERR(RERR message)
- private void sendRREP(RREP message)
- private void sendRREQ(RREQ message, RouteEventClient eventClient, Object uniqueKey)

Μηνύματα δρομολόγησης του AODVManager

Όλα τα μηνύματα δρομολόγησης της στοίβας του δικτύου βρίσκονται στο πακέτο *com.sun.spot.peripheral.radio.mhrp.aodv.messages*. Τα μηνύματα αυτά είναι τα ακόλουθα:

- RREQ(long origAddress, long destinationMACaddress):
- RREP(RREQ message):
- RERR(long originator, long destination):

Πρόσθετα, υπάρχει επίσης άλλη μια κλάση, η AODVMessage.java, την οποία και κληρονομούν όλες οι κλάσεις μηνυμάτων: *RREQ.java, RREP.java* και *RERR.java*. Από τη

κλάση αυτή μπορούμε να πάρουμε πληροφορίες για το τύπο του μηνύματος, τον αριθμό ακολουθίας του, την διεύθυνση προέλευσης και την διεύθυνση προορισμού του.

AODV.REQUEST

Στο επόμενο πακέτο, που είναι το *com.sun.spot.peripheral.radio.mhrp.aodv.request* περιλαμβάνονται οι παρακάτω κλάσεις:

• RequestEntry

Συγκρίνει δυο καταχωρήσεις αιτήσεων.

• RequestTable

Κατασκευάζει ένα νέο πίνακα αίτησης.

- RequestTable()
- boolean addRREQ(RREQ message, RouteEventClient eventClient, Object uniqueKey)

• RequestTableCleaner

Καλεί τη μέθοδο που είναι υπεύθυνη για το καθαρισμό του πίνακα αίτησης.

Ο Πίνακας δρομολόγησης του AODVManager

Ο πίνακας δρομολόγησης βρίσκεται στο πακέτο *com.sun.spot.peripheral.radio.mhrp.aodv.routing* και περιλαμβάνει τις παρακάτω κλάσεις:

• RoutingEntry

Περιέχει τις καταχωρήσεις του πίνακα δρομολόγησης.

• RoutingNeighbor

Περιοδική αποστολή RREP πακέτων στους γειτονικούς κόμβους για να παραμένει ενεργή η διαδρομή.

• RoutingTable

Κατασκευάζει τον πίνακα δρομολόγησης και διαχειρίζεται όλες τις διαθέσιμες διαδρομές.

- void addRoute(long address, long nextHopMACAddress, int hopCount, int routeCost, int destSeqNumber)
- void addRoute(long senderMACAddress, RREQ message)
- void addRoute(long senderMACAddress, RREP message)
- int getDestinationSequenceNumber(long address)
- long cleanTable()
- void doTableAddition(Long key, RoutingEntry newEntry)
- boolean freshenRoute(long address)
- void deactivateRoute(long originator, long destination)
- Vector getAllEntries()
- void dumpTable()

• RoutingTableCleaner

Καλεί τη μέθοδο που είναι υπεύθυνη για το καθαρισμό του πίνακα δρομολόγησης.

Η κύρια κλάση LQRPManager

Όπως και η κύρια κλάση AODVManager έτσι και η LQRPManager βασίζεται στο AODV πρωτόκολλο δρομολόγησης. Επιπρόσθετα όμως, λαμβάνει υπόψη και άλλους δυο σημαντικούς παράγοντες για τη δρομολόγηση των πακέτων:

- Διάρκεια ζωής του κόμβου (node lifetime)
- Ποιότητα σύνδεσης (link quality)

Όλη η δομή των πακέτων και των κλάσεων είναι παρόμοια με αυτήν του AODVManager που έχουμε περιγράψει με την ύπαρξη κάποιων επιπρόσθετων κλάσεων και μεθόδων που παρουσιάζουμε παρακάτω:

• synchronized void initialize(long ourAddress, ILowPan lowPan)

• RouteInfo getRouteInfo(long address)

• findRoute(long address, RouteEventClient eventClient, Object uniqueKey)

• boolean invalidateRoute(long originator, long destination)

• void addEventListener(IMHEventListener listener)

• void removeEventListener(IMHEventListener listener)

• int getNextSequenceNumber(int givenNumber)

• int getCurrentSequenceNumber()

• RouteTable getRoutingTable()

• boolean initiateRouteDiscovery(long address)

• int getStatus()

• void addLQRPListener(ILQRPEventListener listener)

• void removeLQRPListener(ILQRPEventListener listener)

Η κλάση του Παραλήπτη (Receiver) είναι παρόμοια με αυτή του AODVManager εκτός από κάποιες προσθήκες που αφορούν το πρωτόκολλο LQRP.

• void handleLQREPMessage(LQREP message, long lastHop)

Διαχειρίζεται τα LQREP μηνύματα (Link Quality Reply Messages).

• void handleLQREQMessage(LQREQ message, long requestor)

Διαχειρίζεται τα LQREQ μηνύματα (Link Quality Request Messages).

Η κλάση του Αποστολέα (Sender) είναι παρόμοια με αυτή του AODVManager εκτός από κάποιες σημαντικές προσθήκες:

• boolean forwardLQRPMessage(LQRPMessage message)

Τοποθετεί ένα μήνυμα στην ουρά προκειμένου να το προωθήσει.

• void sendLQREP(LQREP message)

Αποστέλλει ένα LQREP μήνυμα.

• void sendLQREQ(LQREQ message)

Αποστέλλει ένα LQREQ μήνυμα.

Οι προσθήκες βρίσκονται στο πακέτο *com.sun.spot.peripheral.radio.mhrp.lqrp.linkParams*. Στο πακέτο αυτό βρίσκουμε τέσσερις χαρακτηριστικές κλάσεις:

• ConfigLinkParams

Περιέχει διάφορες παραμέτρους για τη σύνδεση όπως το χρονικό παράθυρο για κάθε μετάδοση, τον αριθμό των χρονικών θυρίδων (timeslots), μία παράμετρο βάρους για τον υπολογισμό του κόστους της διαδρομής, το μέγιστο ρυθμό μεταφοράς δεδομένων και τη μέγιστη χωρητικότητα της μπαταρίας.

• NbrLinkInfo

Περιέχει διάφορες πληροφορίες σχετικά με το πόσο καλά οι γειτονικοί κόμβοι «ακούν» τα πακέτα μας, πόσο καλά «ακούμε» εμείς τα δικά τους και διάφορες άλλες πληροφορίες.

• NodeLifeAndLinkMonitor

Οι πιο σημαντικές μέθοδοι που περιέχονται σε αυτή τη κλάση είναι οι ακόλουθες:

- void notifyPacket(long srcAddress, long dest, int rssi, int corr, int lq, int

packetSize)

Ενημερώνει τη παράμετρο LQI που αφορά τη ποιότητα της γραμμής για κάθε πακέτο που λαμβάνεται.

- NbrLinkInfo addLinkWithAddress(long address, double lq)

Προσθέτει μια νέα σύνδεση προς ένα γειτονικό κόμβο.

- void removeLinkWithAddress(long address)

Αφαιρει μια σύνδεση προς ένα γειτονικό κόμβο.

- double getCurrNormLQForAddress(long address)

Επιστρέφει τη τρέχουσα ποιότητα γραμμής για τη συγκεκριμένη διεύθυνση.

- void setNbrLQ(long address, double cost)

- void updateLinkInfo()

• NodeLifetime

Η κλάση αυτή περιέχει ένα πολύ σημαντικό μέρος, τον υπολογισμό του μέγιστου χρόνου ζωής. Ο τύπος που χρησιμοποιείται για τον υπολογισμό του μέγιστου χρόνου ζωής του κόμβου είναι ο ακόλουθος:

currNormExpLifetime = remainingBattFrac * (1.0 – 1000 * burnRate /

(ConfigLinkParams.MAX_DATA_RATE * ENERGY_PER_BIT_RX))

Παρατηρούμε ότι ο τύπος αυτός λαμβάνει υπόψη τόσο το επίπεδο της μπαταρίας του κόμβου (remainingBattFrac) όσο και άλλους παράγοντες που αφορούν τη κατανάλωση ενέργειας από την αποστολή και λήψη δεδομένων (burnRate).

Το πακέτο *com.sun.spot.peripheral.radio.mhrp.lqrp.messages* περιέχει τα ιδία μηνύματα με αυτά του AODVManager με τη προθήκη δυο επιπλέον μηνυμάτων. Η κλάση LQRPMessage είναι παρόμοια με την κλάση AODVMessage.

• LQREP

- LQREP(LQREQ request, double cost)

Κατασκευάζει ένα νέο LQREP μήνυμα με βάση το LQREQ μήνυμα που έλαβε.

- LQREP(byte[] newMessage)

Κατασκευάζει ένα νέο μήνυμα αίτησης διαδρομής.

- byte[] writeMessage()

Γράφει τις ιδιότητες του μηνύματος στις κατάλληλες θέσεις του buffer.

• LQREQ

- LQREQ(long origSourceAddress, long destinationMACaddress, double cost)

Κατασκευάζει ένα νέο LQREQ μήνυμα.

- LQREQ(byte[] newMessage, int lqi)

Κατασκευάζει ένα νέο μήνυμα αίτησης διαδρομής.

- byte[] writeMessage()

Γράφει τις ιδιότητες του μηνύματος στις κατάλληλες θέσεις του buffer.

Τα άλλα δυο πακέτα είναι παρόμοια με τα πακέτα του AODVManager:

- com.sun.spot.peripheral.radio.mhrp.lqrp.request

- com.sun.spot.peripheral.radio.mhrp.lqrp.routing

5. Υλοποίηση ενός δικτύου αισθητήρων ελεγχόμενης κινητικότητας σταθμού βάσης

Μετά την ανασκόπηση της βιβλιογραφίας και τη μελέτη του συστήματος των Sun SPOTs προχωράμε στην υλοποίηση της προσέγγισης των [1], που όπως είδαμε σε προηγούμενο κεφάλαιο αναφέρεται στην ανάπτυξη ενός δικτύου αισθητήρων ελεγχόμενης κινητικότητας με σημεία συνάντησης, προσαρμοσμένη στις ιδιότητες αλλά και στους περιορισμούς των Sun SPOTs.

5.1 Θεωρητική περιγραφή του αλγορίθμου

Συνοπτικά, οι φάσεις του αλγορίθμου που παρουσιάζεται στην εργασία των [1] είναι:

I. Φάση Αρχικοποίησης

Ο κινητός κόμβος κινείται κατά μήκος μιας προκαθορισμένης τροχιάς και αρχίζει να μεταδίδει μηνύματα τύπου Beacon με πεδία {*SenderIdentity*, L_g (*sender*), *Clusterhead*}, όπου:

SenderIdentity: είναι η ταυτότητα του κόμβου που μεταδίδει το Beacon μήνυμα,

L_g (*sender*): είναι η τιμή του επιπέδου L_g που έχει ορίσει στον εαυτό του και

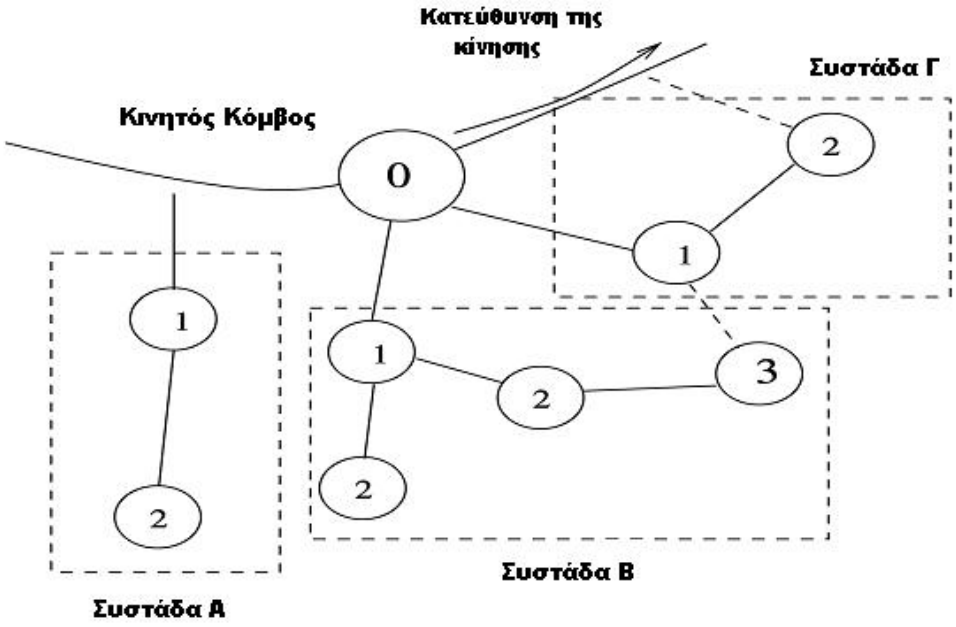
Clusterhead: είναι η ταυτότητα του επικεφαλής κόμβου της συστάδας στην οποία ανήκει ο κόμβος.

Η μεταβλητή L_g ουσιαστικά υποδηλώνει τον αριθμό των hops που έχει διανύσει το μήνυμα. Κάθε στατικός κόμβος που λαμβάνει το μήνυμα το αναμεταδίδει αν και μόνο αν το μήνυμα έχει μικρότερο αριθμό hops από τον αριθμό που έχει αποθηκευμένο στη μνήμη του. Πριν αναμεταδώσει το μήνυμα, ο κόμβος αφενός μεν φροντίζει να αυξήσει κατά ένα τον αριθμό των hops του πεδίου και αφετέρου να αποθηκεύσει στη μνήμη του το αναγνωριστικό του κόμβου από τον οποίο έλαβε το μήνυμα.

Στις παρακάτω εικόνες παρουσιάζεται ο αλγόριθμος που εκτελείται από τους στατικούς κόμβους στη φάση αρχικοποίησης και η διαδικασία σχηματισμού των συστάδων:

1. Initialize: L_g *(ID)* = ∞
2. **while** TRUE
3. Listen for packets
4. **if** Receive BEACON packet
5. **if** BEACON {L_g (*sender*)} +1) < L_g (*ID*)
6. L_g (*ID*):= BEACON {L_g (*sender*)} + 1
7. **if** L_g (*ID*) == 1
8. *Clusterhead*(*ID*):= *ID*
9. **else**
10. *Clusterhead*(ID):= BEACON {*Clusterhead*}
11. Set BEACON = {*ID*, L_g (*ID*), *Clusterhead*(*ID*)}
12. Broadcast BEACON

Εικόνα 5.1. Ο αλγόριθμος που εκτελείται στη φάση αρχικοποίησης από τους στατικούς κόμβους [1]



Εικόνα 5.2. Η διαδικασία σχηματισμού των συστάδων [1]

Έτσι, μετά το τέλος της φάσης αρχικοποίησης θα έχει σχηματιστεί ένας αριθμός από δέντρα, καθένα από τα οποία θα έχει ως ρίζα του ένα κόμβο που θα βρίσκεται στην άμεση εμβέλεια του κινητού κόμβου ενώ ο κάθε κόμβος θα ανήκει σε ακριβώς ένα μόνο τέτοιο δέντρο.

II. Επικοινωνία μεταξύ των κόμβων των συστάδων

Μετά τη φάση αρχικοποίησης θα έχουν πλέον σχηματιστεί οι αντίστοιχες συστάδες οι οποίες θα έχουν ακριβώς έναν επικεφαλής κόμβο. Στη φάση αυτοί οι στατικοί κόμβοι θέλουν να στείλουν τα δεδομένα τους στον επικεφαλής της συστάδας ανεξάρτητα από το αν υπάρχει μια ενεργή διαδρομή μέχρι τον τελικό προορισμό που είναι ο κινητός κόμβος. Ο σκοπός είναι να κερδίσουμε πολύτιμο χρόνο μέχρις ότου ο κινητός κόμβος βρεθεί στην εμβέλεια. Για το σκοπό αυτό εισάγεται ένας μηχανισμός επικοινωνίας μεταξύ του επικεφαλής κόμβου και των κόμβων της συστάδας με τη χρήση ενός πρωτοκόλλου κατευθυνόμενης διάχυσης δεδομένων [30], όπου ο επικεφαλής της συστάδας εκδηλώνει το ενδιαφέρον του στους κόμβους της συστάδας και οι κόμβοι ανταποκρίνονται και στέλνουν τα δεδομένα τους ακολουθώντας την αντίθετη διαδρομή.

Τελικά, ο επικεφαλής της συστάδας λαμβάνει τα δεδομένα από τους κόμβους, τα αποθηκεύει προσωρινά στη μνήμη του και αναλαμβάνει να τα παραδώσει στο τελικό προορισμό που είναι ο κινητός κόμβος.

III. Επικοινωνία μεταξύ επικεφαλής συστάδων και κινητού κόμβου

Στη φάση αυτή οι επικεφαλής των συστάδων θέλουν να στείλουν τα δεδομένα που έχουν συλλέξει από τους κόμβους της συστάδας τους στο κινητό κόμβο. Απαραίτητη προϋπόθεση είναι οι επικεφαλής των συστάδων να γνωρίζουν πότε ο κινητός κόμβος βρίσκεται στην εμβέλεια τους και για πόσο χρονικό διάστημα. Για τους λόγους αυτούς, αρχικά εισάγεται ένας μηχανισμός αποστολής POLL πακέτων ανά συγκεκριμένο χρονικό διάστημα (Tpoll) από το κινητό κόμβο που γνωστοποιεί στους στατικούς κόμβους τη παρουσία του στην άμεση εμβέλεια τους. Μόλις ένας επικεφαλής κόμβος λάβει ένα τέτοιο POLL πακέτο γνωρίζει ότι μπορεί να αρχίσει να

στέλνει τα δεδομένα που έχει προσωρινά αποθηκεύσει στη τοπική του μνήμη. Πριν αρχίσει την αποστολή κάποιου πακέτου ο επικεφαλής της συστάδας ξεκινά δυο χρονομετρητές:

Α. Χρονομετρητής αναμετάδοσης (Retransmit Timer): Ο επικεφαλής ξαναστέλνει το πακέτο για το οποίο δεν έλαβε επιβεβαίωση παραλαβής μόλις λήξει αυτός ο χρονομετρητής. Η ιδιότητα αυτή είναι πολύ σημαντική γιατί έτσι αποφεύγεται η απώλεια πακέτων που οφείλεται σε διάφορα σφάλματα στο κανάλι επικοινωνίας και στις συγκρούσεις που συμβαίνουν σε επίπεδο MAC μεταξύ των διαφόρων επικεφαλής των συστάδων.

Β. Χρονομετρητής απώλειας της σύνδεσης (Connection Lost Timer): Ο επικεφαλής της συστάδας σταματά να στέλνει δεδομένα στον κινητό κόμβο αν δε λάβει κάποιο POLL πακέτο μέσα σε αυτό το διάστημα. Η ιδιότητα αυτή είναι πολύ σημαντική γιατί με αυτό το τρόπο ο επικεφαλής κόμβος μπορεί να γνωρίζει πότε ο κινητός κόμβος έχει απομακρυνθεί από την εμβέλεια του και συνεπώς η σύνδεση μεταξύ τους δεν είναι πλέον ενεργή. Η τιμή του χρονομετρητή ορίζεται ιση με 3 x Tpoll έτσι ώστε να υπάρχει μια χρονική ανοχή σε περίπτωση που χαθούν μέχρι και δυο POLL πακέτα λόγω σφαλμάτων στο κανάλι επικοινωνίας ή των συγκρούσεων που συμβαίνουν σε επίπεδο MAC.

5.2 Εισαγωγή στην υλοποίηση

Με την ανάπτυξη του αλγορίθμου που παρουσιάστηκε στην εργασία των [1] στα Sun SPOTs αντιμετωπίζεται μια πλειάδα από πρακτικά ζητήματα και δίνονται λύσεις για μελλοντική ανάπτυξη διαφόρων εφαρμογών με τη τεχνολογία των Sun SPOTs.

Η διενέργεια των πειραμάτων και των ελέγχων γίνεται σε Sun SPOTs στα οποία προηγουμένως εγκαταστήσαμε την 5^{η} έκδοση (RED) του Sun SPOT SDK που περιγράψαμε στο προηγούμενο κεφάλαιο. Η βασική διαφορά της έκδοσης αυτής από την προηγούμενη, 4^{η} έκδοση (BLUE), που είχαν προ-εγκατεστημένη τα Sun SPOTs είναι ότι η συγκεκριμένη έκδοση περιέχει το νεο πρωτόκολλο δρομολόγησης, το LQRP, που όπως είδαμε προσφέρει αρκετές βελτιώσεις στο σύστημα επικοινωνίας των Sun SPOTs και μεγαλύτερη αξιοπιστία.

Είναι επίσης σημαντικό να αναφέρουμε ότι βασική προϋπόθεση για να εγκαταστήσουμε επιτυχώς ένα δίκτυο από Sun SPOT είναι αυτά να ρυθμιστούν με κοινές ιδιότητες δικτύου έτσι ώστε να είναι δυνατή η επικοινωνία μεταξύ τους στο δίκτυο. Για πρακτικούς λόγους αλλά και για να είναι ευκολότερη η «παρακολούθηση» του δικτύου ορίζουμε χαμηλά την ισχύ εκπομπής των Sun SPOTs. Με την παρακάτω συνάρτηση αρχικοποιούμε τις ιδιότητες δικτύου των Sun SPOTs που συμμετέχουν στο δίκτυο μας:

```
protected void initialize() {

    IRadioPolicyManager rpm = Spot.getInstance().getRadioPolicyManager();

    rpm.setChannelNumber(26); // valid range is 11 to 26
    rpm.setPanId((short) 3);
    rpm.setOutputPower(-30); // valid range is -32 to +31

    System.out.println("Channel Number: "+ rpm.getChannelNumber());
    System.out.println("Pan ID: " + rpm.getPanId());
    System.out.println("Output Power: " + rpm.getOutputPower());

    RoutingPolicy rp = new RoutingPolicy(RoutingPolicy.ALWAYS);
    RoutingPolicyManager.getInstance().policyHasChanged(rp);

}
```

Οι ίδιες ιδιότητες δικτύου ορίζονται και για τον κινητό κόμβο με την μόνη διαφορά ότι δε θα πρέπει να συμμετέχει στα μηνύματα δρομολόγησης του LQRP για να μην επηρεάζει τα μηνύματα δρομολόγησης που ανταλλάσουν οι στατικοί κομβόι. Έτσι, η πολιτική δρομολόγησης για το κινητό κόμβο θα είναι η ENDNODE:

```
RoutingPolicy rp = new RoutingPolicy(RoutingPolicy.ENDNODE);
RoutingPolicyManager.getInstance().policyHasChanged(rp);
```

Στο παράδειγμα μας παρουσιάζουμε πως λειτουργεί ο αλγόριθμος που υλοποιήσαμε σε κάθε φάση χρησιμοποιώντας μια μικρή ενδεικτική ομάδα από τρία Sun SPOTs τα οποία τα έχουμε τοποθετήσει σε σειρά έτσι ώστε να είναι όσο το δυνατόν πιο κατανοητή και ενδεικτική η εν γένει λειτουργία του αλγορίθμου ενώ το τέταρτο Sun SPOT λειτουργεί ως κινητός κόμβος. Κάθε Sun SPOT ορίσαμε να μετρά και να αποστέλλει την θερμοκρασία του περιβάλλοντος (σε βαθμούς Κελσίου).

Εικόνα 5.3. Η διάταξη των Sun SPOTs

5.3 1η Φάση – Δημιουργία των συστάδων

Στη φάση αυτή γίνεται η δημιουργία των συστάδων στο δίκτυο μας και είναι από τις πλέον σημαντικές καθώς οι πληροφορίες που συλλέγονται θα χρησιμοποιούνται συνεχώς στις επόμενες φάσεις. Η φάση αυτή μπορεί να διαρκεί όσο χρειάζεται μέχρι να διανύσει ο κινητός κόμβος το μήκος μιας διαδρομής. Η διαδρομή που ακολουθεί ο κινητός κόμβος είναι προκαθορισμένη και δε μπορεί να μεταβάλλεται.

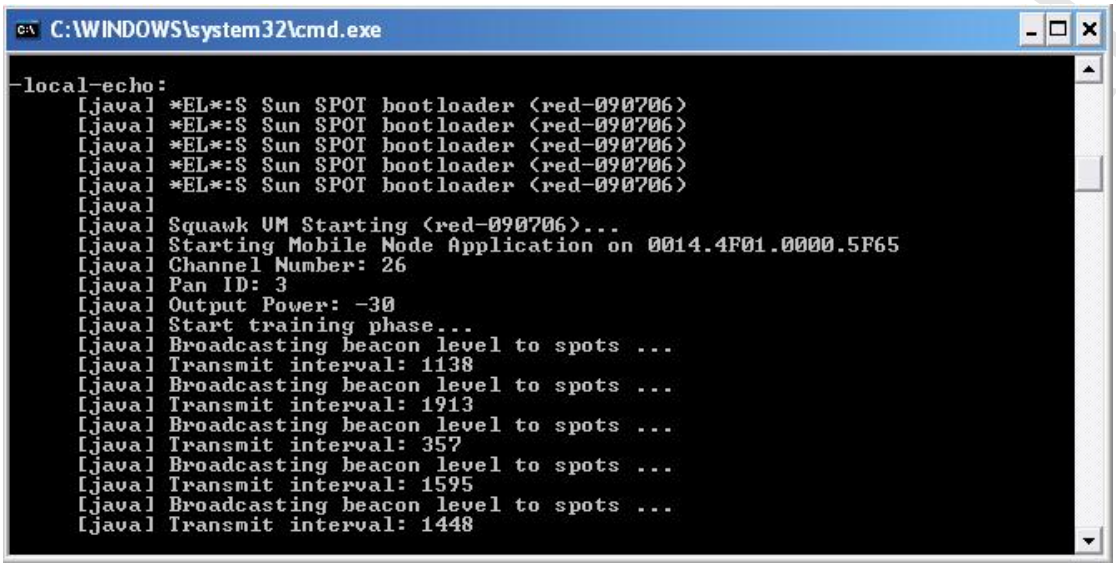
Ο σκοπός της αρχικής αυτής φάσης είναι να δημιουργηθούν οι διάφορες συστάδες με το κάθε κόμβο να ανήκει σε μια ακριβώς συστάδα. Εν τέλει, μετά το πέρας του αλγορίθμου κάποιοι κόμβοι που θα βρίσκονται στην άμεση εμβέλεια του κινητού κόμβου θα γίνουν οι επικεφαλής των συστάδων που θα δημιουργηθούν ενώ οι υπόλοιποι κόμβοι θα λειτουργήσουν ως κόμβοι-πηγές δημιουργώντας έτσι δέντρα από SPOT με ρίζες τους επικεφαλής των συστάδων.

Ο **κινητός κόμβος** στη φάση αυτή αρχίζει να κινείται στη διαδρομή και να μεταδίδει περιοδικά πακέτα τύπου BEACON που γνωστοποιούν τη παρουσία του στο δίκτυο στα στατικά Sun SPOTs (Εικόνα 5.4). Είναι σαφές πως για το κινητό κόμβο θα πρέπει να ανοίξουμε αρχικά μια σύνδεση Radiogram τύπου broadcast για να γίνεται ευρεία εκπομπή των datagrams που θα περιέχουν τα πακέτα τύπου BEACON με τις απαραίτητες πληροφορίες.

```
// Initialize broadcast connection

try {
```

```
            broadcastConn = (RadiogramConnection) Connector.open("radiogram://broadcast:" +
            HOST_PORT);

            broadcast_dg = broadcastConn.newDatagram(broadcastConn.getMaximumLength());

            broadcastConn.setMaxBroadcastHops(1); // 1 hop

} catch (Exception e) {
              System.err.println("Caught " + e + " in connection initialization.");
}

// Start training phase to begin creation of clusters

protected void startTraining() {

        long now = 0L ;

        System.out.println("Start training phase...");

        try {

           now = System.currentTimeMillis();

           while ( System.currentTimeMillis() - now < TRAINING_PERIOD ) {

              // Transmit the Bacon Packet in free range spots
              //System.out.println(pt.BeaconPacket);
              broadcast_dg.reset();
              broadcast_dg.writeUTF(pt.BeaconPacket);
              broadcast_dg.writeInt(Level);
              broadcastConn.send(broadcast_dg);

              System.out.println("Broadcasting bacon level to spots ...");
              int transmit_interval = generator.nextInt(2000-100) + 100;
              System.out.println("Transmit interval: " + transmit_interval);

              Utils.sleep(transmit_interval);
           }

        } catch (Exception e) {
              System.err.println("Caught " + e + " in connection initialization.");
        }

      System.out.println("Training phase completed after " + (System.currentTimeMillis() - now) +
" seconds" );
```

}

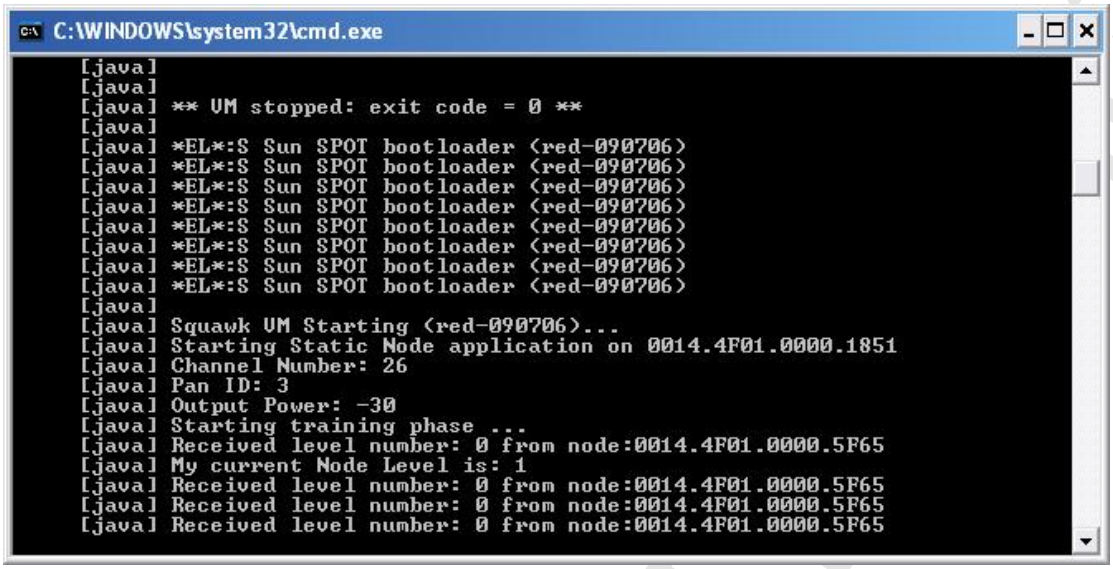


Εικόνα 5.4. Η μετάδοση των Beacon πακέτων από το κινητό κόμβο

Οι **στατικοί κόμβοι** που βρίσκονται στην εμβέλεια του κινητού κόμβου αρχίζουν να λαμβάνουν τα πακέτα τύπου BEACON μέσω της Radiogram σύνδεσης που έχουν ανοίξει για να λαμβάνουν Datagrams πακέτα και εκτελούν τον αλγόριθμο που περιγράφεται στην εργασία των [1] (Εικόνα 5.5). Ο στατικοί κόμβοι προωθούν τα BEACON πακέτα στους υπόλοιπους κόμβους μέσω μιας αντίστοιχης Radiogram σύνδεσης τύπου broadcast.

```
protected void training () {

    System.out.println("Starting training phase ...");

    long now = System.currentTimeMillis();

    while (System.currentTimeMillis() - now < TRAINING_PERIOD) {

      try {
        server_dg.reset();
        serverConn.receive(server_dg);

        String type = server_dg.readUTF();

        if (type.equals(pt.BeaconPacket)) {

          int tempNodeLevel = server_dg.readInt();

              System.out.println("Received level number: " + tempNodeLevel + " from node:"
              + server_dg.getAddress());
```

```
            if (tempNodeLevel + 1 < NodeLevel) {

                NodeLevel = tempNodeLevel + 1;
                System.out.println(NodeLevel);

                if (NodeLevel == 1) {
                    isClusterHead = true;
                    clusterHeadAddress = System.getProperty("IEEE_ADDRESS");
                    mobileNodeAddress = server_dg.getAddress();
                }
                else {
                    clusterHeadAddress = server_dg.readUTF();
                    parentAddress=server_dg.getAddress();
                }

                //System.out.println(nextSpotAddress);
                for (int i=0; i<numberOfRetries; i++) {
                    int transmit_interval = generator.nextInt(300-100) + 100;
                    broadcastBeacon();
                    Utils.sleep(transmit_interval);
                }
            }
        }
        else {
            System.out.println("Wrong Packet");
        }

    } catch (Exception e) {

    }
  }
}
```



Εικόνα 5.5. Λήψη των Beacon πακέτων από τα στατικά Sun SPOTs

5.4 2η Φάση – Επικοινωνία μεταξύ των κόμβων των συστάδων

Μετά το τέλος της πρώτης φάσης θα έχουν δημιουργηθεί οι αντίστοιχες συστάδες που αποτελούν πλέον το δίκτυο μας. Τα Sun SPOTs που βρίσκονται στην άμεση εμβέλεια του κινητού κόμβου έχουν τιμή επιπέδου 1 και είναι υπεύθυνα για την μεταβίβαση όλων των δεδομένων των αισθητήρων της συστάδας τους στο κινητό κόμβο. Οι επικεφαλής των συστάδων λειτουργούν ως ενδιάμεσοι κόμβοι που συνδέουν τα στατικά Sun SPOTs με το κινητό κόμβο για να μεταφέρουν τα δεδομένα τους.

5.4.1 1η Προσέγγιση

Η πρώτη προσέγγιση για την επικοινωνία των Sun SPOTs με τους επικεφαλής των συστάδων είναι η πλέον απλή. Κάθε κόμβος στέλνει τα δεδομένα του σε ακριβώς ένα hop στον κόμβο-πατέρα του, ο οποίος με τη σειρά του τα προωθεί στο δικό του κόμβο-πατέρα μέχρι τα δεδομένα να φτάσουν τελικά στον επικεφαλής της συστάδας.

Οι αποστολές των δεδομένων γίνονται με τη χρήση μιας σύνδεσης Radiogram με παραλήπτη τον αντίστοιχο κόμβο-πατέρα σε ακριβώς ένα hop. Η μέθοδος αυτή αν και είναι η πλέον προφανής γρήγορα συνειδητοποιήσαμε ότι δεν αποτελεί ιδανική λύση στο πρόβλημα της επικοινωνίας καθώς υστερεί σημαντικά στη διαχείριση των αλλαγών στο δίκτυο. Δηλαδή, θα ήταν δύσκολο με τον τρόπο αυτό να επιτύχουμε μια δυναμική δρομολόγηση των πακέτων μεταξύ των κόμβων και των επικεφαλής τους. Οι διαδρομές που θα ακολουθούσαν τα πακέτα θα ήταν στατικές ενώ δε θα λαμβάνονταν υπόψη σημαντικά ενεργειακά και πρακτικά κριτήρια, όπως η ποιότητα της σύνδεσης και το επίπεδο ισχύος της μπαταριάς των Sun SPOTs.

Σε συνέχεια της προσέγγισης μας, θα πρέπει να αναφέρουμε ότι θα ήταν επίσης λανθασμένο να χρησιμοποιήσουμε την Radiogram σύνδεση για την αποστολή των δεδομένων απευθείας στον επικεφαλής κόμβο. Όπως έχουμε δει ήδη σε προηγούμενο κεφάλαιο το πρωτόκολλο αυτό δεν παρέχει καμία εγγύηση ότι τα πακέτα θα παραλειφθούν σωστά ή ότι θα φτάσουν στον προορισμό με την σειρά που στάλθηκαν όταν μεσολαβούν περισσότερα του ενός hop [3].

Έτσι, η λύση είναι να στραφούμε στο άλλο πρωτόκολλο επικοινωνίας των Sun SPOTs, το radiostream, το οποίο παρέχει αξιόπιστη μετάδοση δεδομένων μεταξύ δυο Sun SPOTs σε πολλαπλά hops με χρήση του πρωτοκόλλου δρομολόγησης LQRP [3].

5.4.2 2η Προσέγγιση

Στη 2η μας προσέγγιση, αντιλαμβανόμενοι πλέον το «κενό» επικοινωνίας με τη Radiogram σύνδεση υλοποιούμε μια μέθοδο επικοινωνίας μεταξύ των Sun SPOTs κάθε συστάδας και του επικεφαλής της συστάδας με χρήση radiostream συνδέσεων και τη δρομολόγηση των πακέτων να αναλαμβάνει αποκλειστικά το πρωτόκολλο δρομολόγησης LQRP σε αντίθεση με τη θεωρητική προσέγγιση που προτείνεται στην εργασία των [1] όπου όπως περιγράψαμε γίνεται χρήση ενός πρωτοκόλλου κατευθυνόμενης διάχυσης δεδομένων. Η επιλογή του πρωτοκόλλου LQRP είναι η ενδεδειγμένη αφενός μεν γιατί είναι ένα δυναμικό πρωτόκολλο δρομολόγησης που έχει κατασκευαστεί ειδικά για τα Sun SPOTs και αποτελεί μέρος του Sun SPOT SDK και αφετέρου γιατί λαμβάνει υπόψη, όπως έχουμε δει, σημαντικούς παράγοντες για το δίκτυο, όπως η ποιότητα της γραμμής και η διάρκεια ζωής των Sun SPOTs.

Μετά το πέρας της 1ης φάσης κάθε Sun SPOT θα γνωρίζει αν είναι κόμβος-πηγή ή αν είναι επικεφαλής συστάδας, οπότε διακρίνουμε δυο περιπτώσεις με τις ενέργειες που πρέπει να γίνουν από κάθε Sun SPOT. Η σημαντική επισήμανση που πρέπει να κάνουμε στο σημείο αυτό είναι ότι για να χρησιμοποιήσουμε τις radiostream συνδέσεις για την επικοινωνία μεταξύ των Sun SPOTs και τον επικεφαλής της συστάδας θα πρέπει και τα δύο άκρα κάθε φορά να ανοίξουν συνδέσεις προσδιορίζοντας τον ίδιο αριθμό θύρας και τις αντίστοιχες IEEE διευθύνσεις των Sun SPOTs. Δηλαδή, από τη μεριά του ο επικεφαλής της συστάδας θα πρέπει να γνωρίζει όλες τις IEEE διευθύνσεις των Sun SPOTs της συστάδας του προκειμένου να ανοίξει τις αντίστοιχες συνδέσεις από τη μεριά του και από την άλλη τα Sun SPOTs θα πρέπει να γνωρίζουν τον μοναδικό αριθμό θύρας που τους έχει δοθεί έτσι ώστε να ανοίξουν και αυτά τη σύνδεση στην άλλη άκρη.

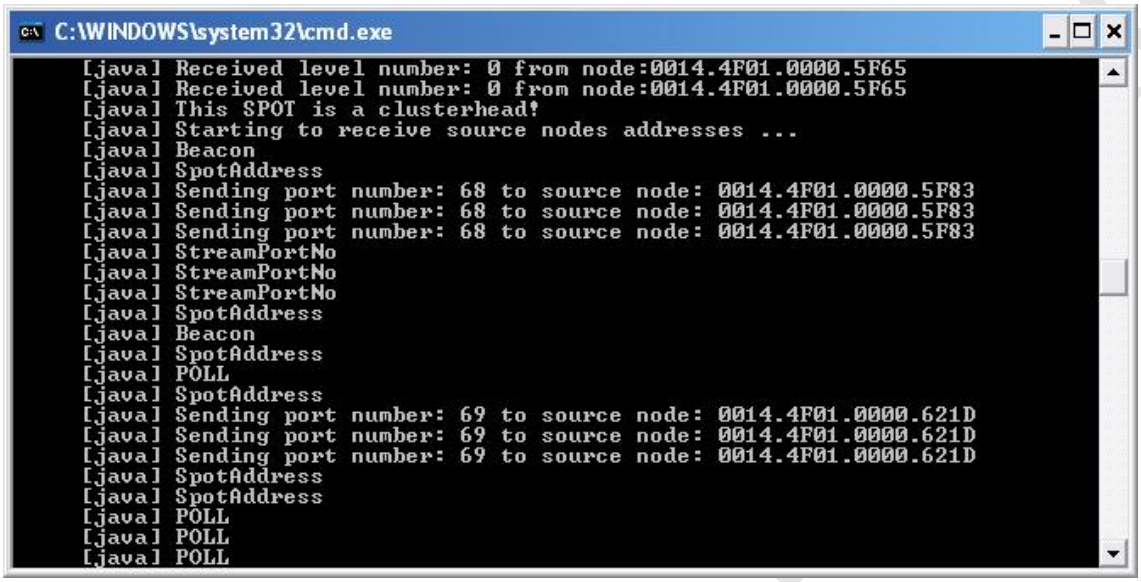
Η προϋπόθεση αυτή για να χρησιμοποιήσουμε τις radiostream συνδέσεις μας οδήγησε στην υλοποίηση ενός μηχανισμού ανταλλαγής διευθύνσεων και αριθμού θυρών μεταξύ των Sun SPOTs της κάθε συστάδας. Οι ενέργειες που θα πρέπει να γίνουν από τα Sun SPOTs και από τον επικεφαλής της συστάδας είναι οι εξής:

- Ο **επικεφαλής** της συστάδας αρχίζει να λαμβάνει τις IEEE διευθύνσεις των Sun SPOTs της συστάδας του μέσω της Radiogram σύνδεσης που έχει ανοίξει από την 1η φάση και αποστέλλει πίσω στο καθένα τον μοναδικό αριθμό θύρας που του ανέθεσε για να ανοίξουν μετέπειτα την radiostream σύνδεση (Εικόνα 5.6). Η λήψη των διευθύνσεων και η αποστολή του αριθμού των θυρών γίνεται μέσω των Radiogram συνδέσεων που έχουμε ανοίξει ήδη από τη 1η φάση. Για να αποφύγουμε τον κίνδυνο απώλειας κάποιου αριθμού θύρας λόγω ενδεχόμενων συγκρούσεων σε επίπεδο MAC η λόγω σφαλμάτων στη γραμμή ο επικεφαλής της συστάδας εκπέμπει κάθε μοναδικό αριθμό θύρας σε ένα αριθμό τριών επαναλήψεων.

```
// start collecting cluster's source nodes addresses and send back the given ports

protected void receiveSourceNodesAddresses() {

    System.out.println("Starting to receive source nodes addresses ...");

    long now = System.currentTimeMillis();
    while (System.currentTimeMillis() - now < TRAINING_PERIOD) {

      try {
        server_dg.reset();
        serverConn.receive(server_dg);
        String type = server_dg.readUTF();
```

```
            System.out.println(type);

            if (type.equals(pt.SpotAddressPacket)) {

               String receivedClusterHeadAddress = server_dg.readUTF();

               // check if it is a packet from our cluster
               if (receivedClusterHeadAddress.equals(clusterHeadAddress)) {

                  String SourceNodeAddress = server_dg.readUTF();

                  if (SpotsAddresses.contains(SourceNodeAddress) == false) {

                     SpotsAddresses.addElement(SourceNodeAddress);

                           SpotsAddressesStreamPorts.addElement(Integer.toString(STARTING_
                           STREAM_PORT));

                     for (int i=0; i<numberOfRetries; i++) {

                        int transmit_interval = generator.nextInt(200-100) + 50;

                       // send given stream port number to spot
                        System.out.println("Sending port number: " +
                        STARTING_STREAM_PORT + " to source node: " +
                        SourceNodeAddress);

                        broadcastStreamPortNo(SourceNodeAddress,
                        STARTING_STREAM_PORT);

                        Utils.sleep(transmit_interval);
                     }

                     STARTING_STREAM_PORT++; // increment stream port number
                  }

               }
            }

         } catch (Exception e) {

         }

      }
```

}



Εικόνα 5.6. Η αποστολή των θυρών στους κόμβους-πηγές της συστάδας

- Ο κάθε **κόμβος – πηγή** αφενός μεν στέλνει τη IEEE διεύθυνση του στον επικεφαλής της συστάδας, λειτουργεί όμως και ως ενδιάμεσος κόμβος για να προωθεί τις διευθύνσεις και των άλλων κόμβων της συστάδας καθώς επίσης και να προωθεί τους αριθμούς των θυρών που έχει στείλει ο επικεφαλής της συστάδας.

  Η αποστολή των διευθύνσεων και των αριθμών των θυρών και στις δυο περιπτώσεις γίνεται με χρήση της σύνδεσης Radiogram τύπου broadcast που έχουμε ανοίξει ήδη από την 1^{η} φάση. Για να αποφύγουμε τον κίνδυνο απώλειας κάποιων διευθύνσεων λόγω ενδεχόμενων συγκρούσεων σε επίπεδο MAC η λόγω σφαλμάτων στη γραμμή κάθε κόμβος εκπέμπει τη διεύθυνση του σε ένα αριθμό τριών επαναλήψεων.

```
// First, notify cluster head of each Spot's address

// Forward other spots address to cluster head

new Thread() {
   public void run() { forwardSpotsAddressAndPortsToClusterHead(); }
}.start();

Utils.sleep(5*1000);

for (int i=0; i<numberOfRetries; i++) {

   // send my address to cluster head
   System.out.println("Sending my address to my cluster head...");

    sendAddressToClusterHead(System.getProperty("IEEE_ADDRESS"));
```

```
            int transmit_interval = generator.nextInt(500-100) + 100;
            Utils.sleep(transmit_interval);

        }
```

Όπου η συνάρτηση *forwardSpotsAddressAndPortsToClusterHead*() ορίζεται ως εξής:

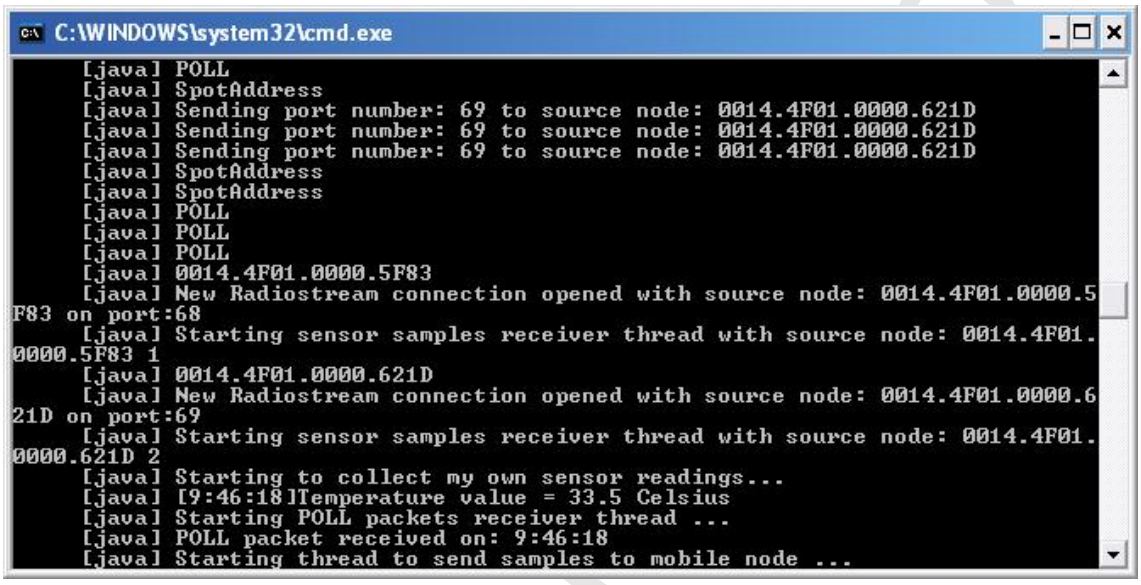
```
   // forward other Spots address to sink node
   protected void forwardSpotsAddressAndPortsToClusterHead() {

      long now=System.currentTimeMillis();
      while (System.currentTimeMillis() - now < TRAINING_PERIOD) {

         try {
            server_dg.reset();
            serverConn.receive(server_dg);

            String type = server_dg.readUTF();

            if (type.equals(pt.SpotAddressPacket)) {

               String receivedClusterHeadAddress = server_dg.readUTF();
               int receivedNodeLevel = server_dg.readInt();

               if (receivedClusterHeadAddress.equals(clusterHeadAddress) &&
(receivedNodeLevel>=NodeLevel) ) {

                  String SourceNodeAddress = server_dg.readUTF();

                   sendAddressToClusterHead(SourceNodeAddress);

               }
            }

            if (type.equals(pt.StreamPortNoPacket)) {

               String receivedClusterHeadAddress = server_dg.readUTF();
               int receivedNodeLevel = server_dg.readInt();

               if (receivedClusterHeadAddress.equals(clusterHeadAddress) &&
(receivedNodeLevel <= NodeLevel)) {

                  String SourceNodeAddress = server_dg.readUTF();

                  // if the packet refers to our spot
```

```
                if (SourceNodeAddress.equals(System.getProperty("IEEE_ADDRESS"))) {
                   //set the streamport number
                   STREAM_PORT = server_dg.readInt();
                   System.out.println("I was given the port number: " + STREAM_PORT + " from
my clusterhead!");
                }

                // broadcast port number to other spots
                else {
                      broadcastStreamPortNo(SourceNodeAddress,  server_dg.readInt());
                }

             }
          }

       } catch (Exception e) {

       }

    }
 }
```

- Ο **επικεφαλής** της συστάδας έχοντας λάβει πλέον τις IEEE διευθύνσεις των κόμβων της συστάδας του και έχοντας αναθέσει σε κάθε έναν από αυτούς από ένα μοναδικό αριθμό θύρας αρχίζει να ανοίγει από τη μεριά του τόσες συνδέσεις radiostream όσες και οι κόμβοι (Εικόνα 5.7).

```
       // for each source node open a datastream connection to start receiving samples.

       SpotsAddressesEnumeration = SpotsAddresses.elements();
       SpotsStreamPortsEnumeration = SpotsAddressesStreamPorts.elements();

       SpotCount = 0;

       while (SpotsAddressesEnumeration.hasMoreElements()) {

          SpotCount++;
          SpotAddress = SpotsAddressesEnumeration.nextElement().toString();
          STREAM_PORT =
Integer.parseInt(SpotsStreamPortsEnumeration.nextElement().toString());
          System.out.println(SpotAddress);

          System.out.println("New Radiostream connection opened with source node: " +
SpotAddress + " on port:" + STREAM_PORT + " " + SpotCount);

          // spawn a thread to receive samples from free range spots using datastreams
```

```
            new Thread() {
            public void run () { receiveSourceNodesSamples(SpotAddress, STREAM_PORT,
SpotCount); }
            }.start();

            Utils.sleep(1000);
        }
```



Εικόνα 5.7. Ο επικεφαλής της συστάδας ανοίγει από την μεριά του τις radiostream συνδέσεις με τους κόμβους-πηγές της συστάδας του

Μετά την εκτέλεση του παραπάνω κώδικα θα τρέχουν πλέον τα αντίστοιχα παράλληλα νήματα, το καθένα από τα οποία θα είναι υπεύθυνο για τη λήψη των δεδομένων από κάθε κόμβο της συστάδας (Εικόνα 5.8). Τα δεδομένα που λαμβάνονται από κάθε κόμβο αποθηκεύονται σε μια ξεχωριστή για το καθένα μεταβλητή τύπου Vector έτσι ώστε σε επόμενη φάση να αποσταλούν στον κινητό κόμβο.

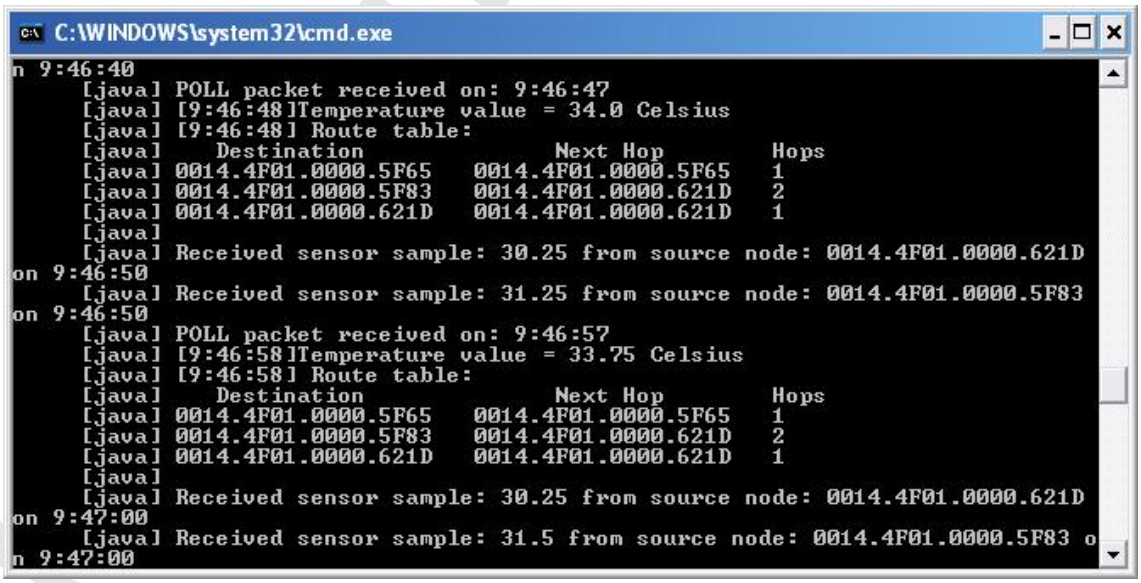
```
protected void receiveSourceNodesSamples(String LocalSpotAddress, int StreamPort, int
LocalSpotCount) {

    RadiostreamConnection streamConn = null;
    DataInputStream dis = null;
    DataOutputStream dos;

    SpotsQueue[LocalSpotCount] = new Vector();

    System.out.println("Starting sensor samples receiver thread with source node: "  +
LocalSpotAddress + " " + LocalSpotCount);

    try {
        streamConn =(RadiostreamConnection)Connector.open("radiostream://" +
LocalSpotAddress + ":" + StreamPort);
```

```
            streamConn.setTimeout(5*1000);
            dis = streamConn.openDataInputStream();
        } catch (Exception ex) {

        }

        while (true) {

           try {

               //dos = streamConn.openDataOutputStream();

               String type = dis.readUTF();
               System.out.println ("dis.readUTF() " + type);

               if (type.equals(pt.SamplePacket)) {

                  long now = dis.readLong();
                  double temperature = dis.readDouble();
                  System.out.println("Received sensor sample: " + temperature + " from source node:
" + LocalSpotAddress + " on " + displayCurrentTime());
                  SpotsQueue[LocalSpotCount].addElement(Double.toString(temperature));

               }

           } catch (Exception e) {

           }

        }

     }
```

Αμέσως μετά ο επικεφαλής της συστάδας ξεκινά ένα άλλο νήμα για να συλλέγει και τα δικά του δεδομένα.

```
protected void collectOwnSample () {

        long now = 0L;
        int reading = 0;
        double temperature = 0;

        SpotsQueue[0] = new Vector();

        System.out.println("Starting to collect my own sensor readings...");
```

```
        while (true) {

            try {
                // Get the current time and sensor reading
                now = System.currentTimeMillis();
                reading = tempSensor.getValue();
                temperature = tempSensor.getCelsius();

                // Flash an LED to indicate a sampling event
                leds[7].setRGB(255, 255, 255);
                leds[7].setOn();

                System.out.println("["+ displayCurrentTime()+ "]" + "Temperature value = " +
temperature + " Celsius");
                SpotsQueue[0].addElement(Double.toString(temperature));

                leds[7].setOff();

                // Go to sleep to conserve battery
                Utils.sleep(SAMPLE_PERIOD - (System.currentTimeMillis() - now));

            } catch (Exception e) {
                System.err.println("Caught " + e + " while collecting my own sample!");
            }
        }
    }
```



Εικόνα 5.8. Ο επικεφαλής της συστάδας λαμβάνει τις θερμοκρασίες από τους κόμβους-πηγές της συστάδας του

- Ο **κόμβος-πηγή** στέλνει τα δεδομένα του στον επικεφαλής κόμβο μέσω της radiostream σύνδεσης που έχουν ανοίξει στα δυο άκρα. Για την αποστολή των δεδομένων ξεκινά ένα καινούριο νήμα με τον κόμβο πηγή να στέλνει τα δεδομένα που συλλέγει περιοδικά:

```
// Establish a datastream connection with clusterhead to start sending sensor samples

new Thread() {
   public void run() { sendMySampleToClusterHead(); }
}.start();
```

Η συνάρτηση *sendMySampleToClusterHead()* ορίζεται ως εξής:

```
// Send spot's sample reading

protected void sendMySampleToClusterHead () {

    RadiostreamConnection streamConn = null;
    DataInputStream dis = null;
    DataOutputStream dos = null;

    long now = 0L;
    int reading = 0;
    double temperature = 0;

    System.out.println("Starting to send my sensor readings to clusterhead...");

    try {
       streamConn =(RadiostreamConnection)Connector.open("radiostream://" +
clusterHeadAddress + ":" + STREAM_PORT);
       dos = streamConn.openDataOutputStream();
    } catch (Exception ex) {

    }

    while (true ) {

       try {

         // Get the current time and sensor reading
         now = System.currentTimeMillis();
         reading = tempSensor.getValue();
          temperature = tempSensor.getCelsius();

         // Flash an LED to indicate a sampling event
```

```
            leds[7].setRGB(255, 255, 255);
            leds[7].setOn();

            // Package the time and sensor reading into a datastream and send it.
            dos.writeUTF(pt.SamplePacket);
            dos.writeLong(now);
            dos.writeDouble(temperature);

            dos.flush();
            //dis.close();
            //dos.close();

            System.out.println("Temperature value = " + temperature + " Celsius");

            leds[7].setOff();

            // Go to sleep to conserve battery
            Utils.sleep(SAMPLE_PERIOD - (System.currentTimeMillis() - now));

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

    }

}
```

Μετά το τέλος της 2^{ης} φάσης οι κόμβοι κάθε συστάδας έχουν εγκαταστήσει πλέον τις συνδέσεις με τον επικεφαλής της συστάδας στον οποίο στέλνουν τα δεδομένα τους ανεξάρτητα αν ο κινητός κόμβος βρίσκεται στην εμβέλεια του επικεφαλής της συστάδας. Το γεγονός αυτό αν και επιβαρύνει προσωρινά τον επικεφαλής της συστάδας με την αποθήκευση των δεδομένων όλων των Sun SPOTs της συστάδας του εντούτοις κάνει το δίκτυο γρηγορότερο και πιο αξιόπιστο στην αποστολή των δεδομένων στο τελικό προορισμό που είναι ο κινητός κόμβος.

5.5 3η Φάση - Επικοινωνία μεταξύ επικεφαλής συστάδων και κινητού κόμβου

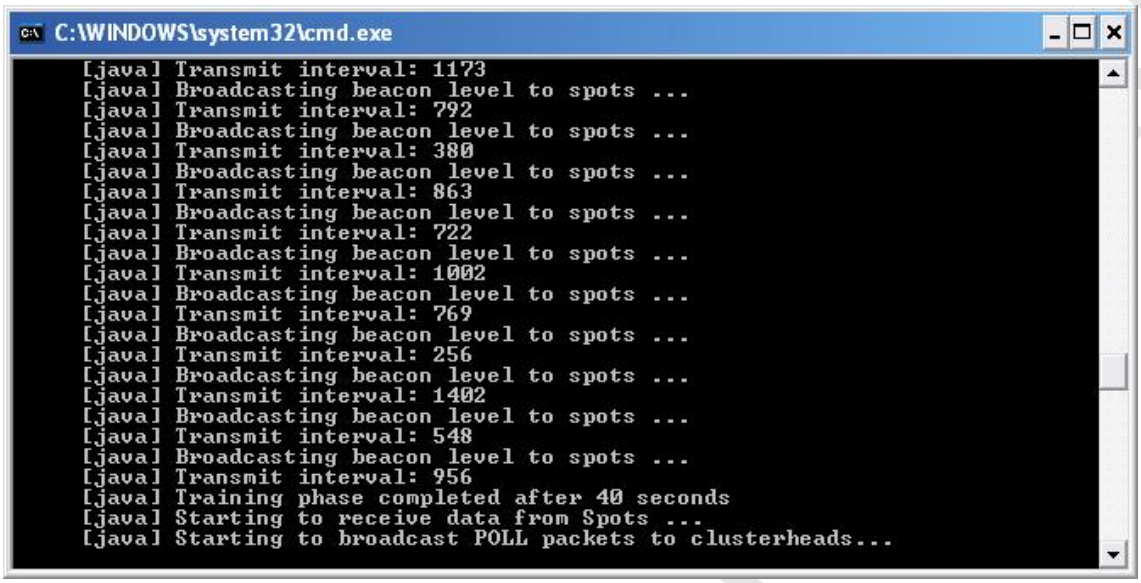
Στη φάση αυτή οι επικεφαλής των συστάδων θέλουν να στείλουν τα δεδομένα τους και τα δεδομένα που έχουν συλλέξει από τους κόμβους της συστάδας τους στο κινητό κόμβο. Για να συμβεί αυτό βασική προϋπόθεση είναι να γνωρίζουν πότε ο κινητός κόμβος βρίσκεται εντός της εμβέλειας τους και για πόσο διάστημα έτσι ώστε τη χρονική περίοδο αυτή να μπορούν να στέλνουν τα δεδομένα που έχουν προσωρινά αποθηκεύσει στη μνήμη τους (μεταβλητή τύπου Vector).

- Ο **κινητός** κόμβος αρχίζει να αποστέλλει πακέτα τύπου POLL τα οποία γνωστοποιούν τη παρουσία του στο χώρο. Έτσι, μετά τη φάση της δημιουργίας των συστάδων ξεκινάμε ένα νήμα στο κινητό κόμβο που είναι υπεύθυνο για την περιοδική αποστολή των POLL πακέτων (Εικόνα 5.9).

```
// Spawn a thread to start transmitting POLL packets to Spots
new Thread() {
    public void run() { broadcastPoLL(); }
}.start();
```

Η συνάρτηση *broadcastPoLL ()* ορίζεται ως εξής:

```
// Periodically send POLL messages to clusterHeads nodes

protected void broadcastPoLL () {

    ITriColorLED[] leds = EDemoBoard.getInstance().getLEDs();
    long now = 0L;

    System.out.println("Starting to broadcast POLL packets to clusterheads...");

    while (true) {

        try {
            // Get the current time and sensor reading
            now = System.currentTimeMillis();

            // Flash an LED to indicate a sampling event
            leds[7].setRGB(255, 255, 255);
            leds[7].setOn();

            // Package the time and Poll Packet into a radio datagram and send it.
            broadcast_dg.reset();
            broadcast_dg.writeUTF(pt.PollPacket);
            broadcast_dg.writeLong(now);

            broadcastConn.send(broadcast_dg);

            leds[7].setOff();

            // Tpoll transmition interval
            Utils.sleep(SAMPLE_PERIOD - (System.currentTimeMillis() - now));

        } catch (Exception e) {
            System.err.println("Caught " + e + " while collecting/sending sensor sample.");
        }
    }

}
```



Εικόνα 5.9 Ο κινητός κόμβος ξεκινά το νήμα που είναι υπεύθυνο για την περιοδική αποστολή των POLL πακέτων

- Ο **επικεφαλής** της συστάδας ξεκινά ένα νήμα που είναι υπεύθυνο για την λήψη των POLL πακέτων από το κινητό κόμβο. Μόλις λάβει το POLL πακέτο αποθηκεύει την ώρα που το έλαβε (Tpoll_Reception_Time) καθώς επίσης και μια ένδειξη που υποδηλώνει το διάστημα που μεσολαβεί μέχρι την αποστολή του επομένου POLL πακέτου από το κινητό κόμβο (Tpoll_INTERVAL). Οι δυο αυτές παράμετροι είναι πολύ σημαντικές γιατί θα χρησιμοποιηθούν αμέσως μετά από τον επικεφαλής της συστάδας για την αποστολή των δεδομένων, που είναι αποθηκευμένα στη μνήμη του, στο κινητό κόμβο.

```
// spawn a thread to receive POLL packets from mobile node

new Thread() {
   public void run() { receivePollPacketsFromMobileNode(); }
}.start();
```

Η συνάρτηση *receivePollPacketsFromMobileNode ()* ορίζεται ως εξής:

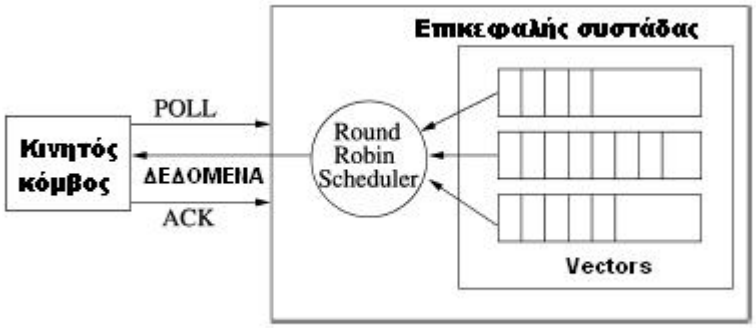
```
protected void receivePollPacketsFromMobileNode() {

      System.out.println("Starting POLL packets receiver thread ...");

      while (true) {

        try {
           server_dg.reset();
           serverConn.receive(server_dg);
           String type = server_dg.readUTF();
```

```
                System.out.println(type);

                if (type.equals((pt.PollPacket))) {

                    Tpoll_INTERVAL = server_dg.readLong();
                    Tpoll_Reception_Time = System.currentTimeMillis();
                }

            } catch (Exception e) {

            }

        }
    }
```

- Στη συνέχεια, ο **επικεφαλής** της συστάδας ξεκινά το νήμα που είναι υπεύθυνο για την τελική αποστολή των δεδομένων στο κινητό κόμβο.

```
// spawn a thread to send the samples to mobile node

new Thread() {
   public void run() { sendSamplesToMobileNode(); }
}.start();
```

Η αποστολή των δεδομένων γίνεται ακριβώς όπως περιγράφεται και στην εργασία των [1] στο μεσοδιάστημα που μεσολαβεί από τη λήψη δυο POLL πακέτων. Τα δεδομένα που έχουν συλλεχτεί από κάθε κόμβο όπως έχουμε ήδη αναφέρει αποθηκεύονται σε μια ξεχωριστή μεταβλητή τύπου Vector και αποστέλλονται στο κινητό κόμβο σε ένα ακριβώς hop με κυκλική σειρά εξυπηρέτησης όλων των Vectors (Round Robin) όπως φαίνεται στη παρακάτω εικόνα:



Εικόνα 5.10. Η αποστολή των δεδομένων στο κινητό κόμβο [1]

Η συνάρτηση *sendSamplesToMobileNode ()* τελικά υλοποιείται ως εξής:

```
protected void sendSamplesToMobileNode() {

	System.out.println("Starting thread to send samples to mobile node ...");

	// Initialize radiogram connection

	// Connection objects
	RadiogramConnection radiogramConn = null;
	Datagram radiogram_dg = null;

	try {
		radiogramConn = (RadiogramConnection) Connector.open("radiogram://" +
mobileNodeAddress + ":" + HOST_PORT);
		radiogram_dg =
radiogramConn.newDatagram(radiogramConn.getMaximumLength());  //
		radiogramConn.setMaxBroadcastHops(1); // 1 hop

	} catch (Exception e) {
		System.err.println("Caught " + e + " in connection initialization.");

	}

while (true) {

// only send samples to mobile node as long as the mobile node is on range
while( (System.currentTimeMillis() - Tpoll_Reception_Time) < 3 * Tpoll_INTERVAL ) {

		//System.out.println("Sending samples to mobile node ....");

		for (int i=0; i <= SpotCount; i++) {

		   if (SpotsQueue[i].isEmpty()== false){

			try {
			    radiogram_dg.reset();
			    radiogram_dg.writeUTF(pt.SamplePacket);
			    radiogram_dg.writeUTF(SpotsQueue[i].firstElement().toString());
			    radiogramConn.send(radiogram_dg);

			    SpotsQueue[i].removeElementAt(0);
			    int transmit_interval = generator.nextInt(500-100) + 100;
			    Utils.sleep(transmit_interval);
```

```
                } catch (Exception ex) {

                }

            }

        }

    }
}
```

Μετά την επιτυχή αποστολή των δεδομένων φροντίζουμε να τα σβήνουμε από τη μνήμη έτσι ώστε να ελευθερώνεται ο χώρος για την αποθήκευση των νέων δεδομένων που λαμβάνει ο επικεφαλής της συστάδας.

- Ο **κινητός** κόμβος καθώς κινείται λαμβάνει τα δεδομένα από τους επικεφαλής των συστάδων που βρίσκονται στην άμεση εμβέλεια του και τα αποθηκεύει στη μνήμη του.

```
// Spawn a thread to start receiving samples form spots

new Thread() {
   public void run() { receiveSamplesFromSpots(); }
}.start();
```

Η συνάρτηση *receiveSamplesFromSpots ()* ορίζεται ως εξής:

```
// Receive sample readings from Spots
protected void receiveSamplesFromSpots() {

    System.out.println("Starting to receive data from Spots ...");

    while (true) {

        try {
            server_dg.reset();
            serverConn.receive(server_dg);
            String type = server_dg.readUTF();

            if (type.equals(pt.SamplePacket)) {

                String temperature = server_dg.readUTF();
                System.out.println("Received temperature " + temperature + " from spot: " +
server_dg.getAddress());
```

```
                System.out.println( " on " + displayCurrentTime());
                spotSamplesReceived.addElement(temperature); // save temperatures in
spotSamplesReceived vector

            }

        } catch (Exception e) {

        }

    }
}
```

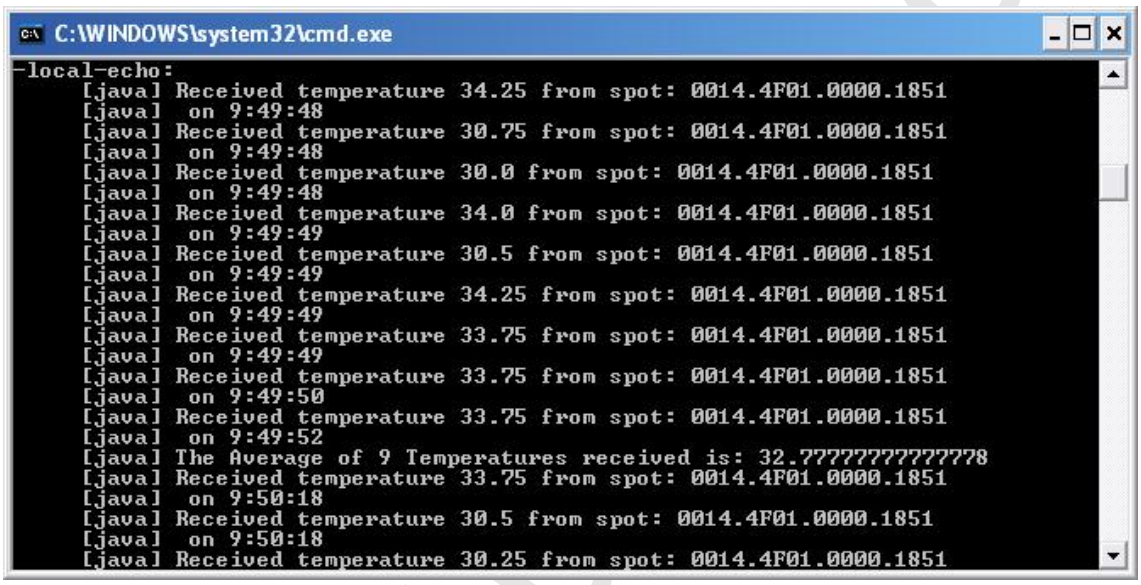
Τα δεδομένα που συλλέγει ο κινητός κόμβος μπορεί να τα αξιοποιήσει ποικιλοτρόπως, όπως π.χ. να εξάγει κάποιο αθροιστικό αποτέλεσμα των δεδομένων, να τα στείλει σε άλλο κινητό κόμβο ή σε κάποιο σταθερό σταθμό βάσης κ.ο.κ. (Εικόνα 5.11).

```
        // Spawn a thread to start calculating AVG of Samples received

         new Thread() {
            public void run() { calculateAVGofSamples(); }
         }.start();
```

Η συνάρτηση *calculateAVGofSamples()* υπολογίζει το μέσο όρο των θερμοκρασιών ανά συγκεκριμένο χρονικό διάστημα:

```
    protected void calculateAVGofSamples() {

        while (true) {

            Utils.sleep(CALCULATION_PERIOD);

            double AvgOfSamples = 0;
            double SumOfSamples = 0;

            int numberOfReceivedSamples = spotSamplesReceived.size();

            for (int i=0; i< numberOfReceivedSamples; i++) {
                SumOfSamples = SumOfSamples +
Double.parseDouble(spotSamplesReceived.elementAt(i).toString());
            }

            // Calculate AVG of Samples
            AvgOfSamples = (SumOfSamples / numberOfReceivedSamples);

            System.out.println("The Average of Temperatures received is: " + AvgOfSamples);
```

```
            // free memory

            for (int i=0; i< numberOfReceivedSamples; i++) {
               spotSamplesReceived.removeElementAt(0);
            }
         }
      }
```



Εικόνα 5.11. Ο κινητός κόμβος λαμβάνει τις θερμοκρασίες από τον επικεφαλής της συστάδας που βρίσκεται στην εμβέλεια του και υπολογίζει το μέσο όρο των θερμοκρασιών ανά συγκεκριμένο χρονικό διάστημα

5.6 Σύνοψη του αλγορίθμου

Συνοπτικά, τα βήματα του αλγορίθμου που εκτελείται στα Sun SPOTs είναι:

Κινητός κόμβος

1. Initiate Training Phase
   While training
      BroadcastBeaconPackets {Level:0}
   End while
2. Initiate Data collection Phase from Spots
   Start Thread { receiveSamplesFromSpots }
   Start Thread { broadcastPoll }
3. Initiate Processing of collected Data
   Start Thread { calulateAVGof Samples}

Στατικοί κόμβοι

```
1. Enter Training and Cluster Formation Phase
        While training
                ReceiveBeaconPackets {Level}
                If Level + 1 < CurrentLevel
                        CurrentLevel = Level +1
                        BroadcastBeaconPacket { CurrentLevel }
                End If
        End while
        If CurrentLevel = 1
                isClusterHead = true
        Else
                isClusterHead = false
        End If
2. Initiate Intra - Cluster Communication Phase
        If isClusterHead
                receiveSourceNodesAdresses
                broadcastStreamPortNumbers
                For each SourceNode
                        Start Thread { receiveSourceNodesSamples }
                End for
                Start Thread { collectOwnSample }
        Else
                forwardSpotsAddressAndPortsToClusterHead
                sendAddressToClusterHead { MyAddress }
                Start Thread { sendMySampleToClusterHead }
        End if
3. Initiate ClusterHead - MobileNode Communication Phase
        Start Thread { receivePollPacketsFromMobileNode }
        Start Thread { sendSamplesToMobileNode }
```

6. Συμπεράσματα – Περίληψη

Στην εργασία αυτή αρχικά έγινε μια μελέτη των χαρακτηριστικότερων προσεγγίσεων στο τομέα της κινητικότητας στα ασύρματα δίκτυα αισθητήρων. Από τη βιβλιογραφία είναι κατανοητό πως η μεγαλύτερη πρόκληση κατά τον σχεδιασμό και την ανάπτυξη ενός δικτύου αισθητήρων παραμένει η μείωση της καταναλωμένης ενέργειας γεγονός που οφείλεται στην ιδιαίτερη φύση των δικτύων αυτών. Όλες οι προσεγγίσεις στην κινητικότητα στοχεύουν αφενός μεν στη μείωση του ενεργειακού φορτίου και αφετέρου στη συνδεσιμότητα όλων των κόμβων του δικτύου με σκοπό η επικοινωνία να γίνεται όσο το δυνατόν ταχύτερα και αξιόπιστα με βάση τις ανάγκες και τους περιορισμούς του δικτύου.

Στη συνέχεια, έγινε μια σχολαστική μελέτη της τεχνολογίας ασύρματων δικτύων JAVA Sun SPOTs που τα τελευταία χρόνια αρχίζουν να γίνονται αντικείμενο μελέτης τόσο για πειραματικούς όσο και για πρακτικούς σκοπούς. Ιδιαίτερη έμφαση δόθηκε στο προγραμματισμό των συσκευών αυτών με σκοπό αφενός μεν τη πραγματική κατανόηση των λειτουργιών και των δυνατοτήτων τους και αφετέρου η μελέτη αυτή να αποτελέσει προπομπό για τη μετέπειτα υλοποίηση ενός δικτύου αισθητήρων. Το μεγάλο πλεονέκτημα των Sun SPOTs είναι ότι προσφέρουν μεγάλη ευελιξία στον προγραμματιστή καθώς είναι πλήρως προγραμματιζόμενα μέσω της Java. Η κατανόηση των πρωτοκόλλων επικοινωνίας (Radiogram, Radiostream) και των ιδιοτήτων των Sun SPOTs υπήρξε κομβικής σημασίας για την εξέλιξη της εργασίας.

Η εξοικείωση εν τέλει με τα Sun SPOTs μας οδήγησε στην κατασκευή ενός δικτύου αισθητήρων ελεγχόμενης κινητικότητας βασιζόμενη σε σημεία συνάντησης όπως αυτή προτάθηκε στην εργασία των [1]. Στην εργασία μας υλοποιήσαμε βήμα προς βήμα κάθε φάση του αλγορίθμου που περιγραφόταν σε θεωρητικό επίπεδο, δένοντας έτσι τα κενά που υπήρχαν μεταξύ θεωρίας και πράξης. Η βασική συμβολή υπήρξε στη δεύτερη φάση του αλγορίθμου, που αφορά την επικοινωνία μεταξύ των επικεφαλής της συστάδας και των κόμβων, όπου αφού εγκαταστήσαμε ένα μηχανισμό ανταλλαγής διευθύνσεων και αριθμού θυρών η επικοινωνία γίνεται πλέον αξιόπιστα μέσω ενός δυναμικού πρωτοκόλλου δρομολόγησης που είχαν εγκατεστημένο τα Sun SPOTs, το LQRP, κάτι το οποίο έχει σημαντικά ενεργειακά και πρακτικά οφέλη για το δίκτυο, αφού λαμβάνονται υπόψη κρίσιμοι παράγοντες όπως το επίπεδο της μπαταρίας των κόμβων και η ποιότητα της γραμμής [3].

Η εργασία μας έχει σημαντικές προοπτικές και μελλοντικές επεκτάσεις όπως:

Α. Υλοποίηση διαφόρων αλγορίθμων ελέγχου της κίνησης κατά τη συλλογή των δεδομένων με την ενσωμάτωση κάποιου Sun SPOT σε ένα προγραμματιζόμενο κινούμενο ηλεκτρονικό όχημα (π.χ. TrackBot [39]).

Β. Υλοποίηση περαιτέρω προσεγγίσεων και αλγορίθμων από τη βιβλιογραφία, που αφορούν τη κινητικότητα στα ασύρματα δίκτυα αισθητήρων, με τη τεχνολογία των Sun SPOTs βασιζόμενοι στις λύσεις που δόθηκαν κατά την υλοποίηση του υπάρχοντος δικτύου.

Γ. Πρακτική μελέτη των δικτύων αισθητήρων σε πραγματικές συσκευές και σε πραγματικές συνθήκες και όχι μόνο σε θεωρητικό επίπεδο ή σε επίπεδο προσομοίωσης.

Πηγές - Βιβλιογραφία

[1] A. A. Somasundara, A. Kansal, D. Jea, D. Estrin, and M. B. Srivastava, “Controllably Mobile Infrastructure for Low Energy Embedded Networks”, IEEE Transactions on Mobile Computing (TMC), 2006 (doi: 10.1109/TMC.2006.109).

[2] Sun Labs, “Sun SPOT Theory of Operation, Red Release 5.0”, Sun Microsystems, 2009.

[3] Sun Labs, ”Sun Small Programmable Object Technology (Sun SPOT) Developer’s Guide, Red Release 5.0”, Sun Microsystems, 2009.

[4] E. Ekici, Y. Gu, and D. Bozdag, “Mobility-Based Communication in Wireless Sensor Networks”, IEEE Communications Magazine, 2006 (http://awin.cs.ccu.edu.tw/magazine/IEEE_com/2006/July/01668382.pdf).

[5] X. Li, A. Nayak, and I. Stojmenovic, “Sink Mobility in Wireless Sensor Networks, in Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication (eds A. Nayak and I. Stojmenovic)”, John Wiley & Sons, 2010 (http://www.site.uottawa.ca/~xuli/publications/ch06.pdf).

[6] T. Banerjee, B. Xie, J.H. Jun, and D.P. Agrawal, “Increasing Lifetime of Wireless Sensor Networks Using Controllable Mobile Cluster Heads”, Wireless Communications and Mobile Computing, 2008 (doi: 10.1002/wcm.763).

[7] Z. Vincze, D. Vass, R. Vida, A. Vidacs, and A. Telcs, “Adaptive sink mobility in event-driven densely deployed wireless sensor networks”, Ad Hoc & Sensor Wireless Networks, 2007(http://www.hsnlab.hu/~vidacs/publications/aswin51.pdf).

[8] L. Friedmann and L. Boukhatem, “Efficient Multi-sink Relocation in Wireless Sensor Network”, Ad Hoc & Sensor Wireless networks, 2009 (doi: 10.1109/ICNS.2007.57).

[9] S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli, and Z.M. Wang, “Controlled sink mobility for prolonging wireless sensor networks lifetime”, ACM Wireless Networks, 2008 (doi: 10.1007/s11276-007-0017-x).

[10] J. Luo and J.P. Hubaux, “Joint mobility and routing for lifetime elongation in wireless sensor networks”, In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 2005 (doi: 10.1109/INFCOM.2005.1498454).

[11] H.S. Kim, T.F. Abdelzaher, and W.H. Kwon, “Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks”, In Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems, 2003 (doi: 10.1145/958491.958515).

[12] P. Baruah, R. Urgaonkar, and B. Krishnamachari, “Learning enforced time domain routing to mobile sinks in wireless sensor fields”, In Proceedings of the 29th IEEE International Conference on Local Computer Networks, 2004 (doi: 10.1109/LCN.2004.71).

[13] R.C. Shah, S. Roy, S. Jain, and W. Brunette, “Data MULEs: modelling and analysis of a three-tier architecture for sparse sensor networks” Ad Hoc Networks, 2003 (http://www.ee.washington.edu/research/funlab/Publications/2003/data_mules.pdf).

[14] S. Nesamony, M.K. Vairamuthu, and M.E. Orlowska, “On Optimal Route of a Calibrating Mobile Sink in a Wireless Sensor Network”, In Proceedings of the 4th International Conference on Networked Sensing Systems, 2007 (doi: 10.1109/INSS.2007.4297389).

[15] S. Nesamony, M.K. Vairamuthu, M.E. Orlowska, and S.W. Sadiq, “On optimal route computation of mobile sink in a wireless sensor network”, Technical Report, University of Queensland, 2006 (http://espace.library.uq.edu.au/eserv/UQ:7693/TR-465.pdf).

[16] R. Sugihara and R.K. Gupta, “Scheduling under Location and Time Constraints for Data Collection in Sensor Networks”, In Proceedings of the 28th IEEE Real-Time Systems Symposium Work in Progress Session, 2007 (http://dar.ucsd.edu/site/pubs/Sugihara_RTSS07.pdf).

[17] R. Sugihara and R.K. Gupta, “Improving the Data Delivery Latency in Sensor Networks with Controlled Mobility”, In Proceedings of the 4th IEEE International Conference on Distributed

Computing in Sensor Systems, 2008 (http://dar.ucsd.edu/site/pubs/Sugihara_DCOSS08.pdf).

[18] A. Kansal, A.A. Somasundara, D.D. Jea, M.B. Srivastava, and D. Estrin, “Intelligent Fluid Infrastructure for Embedded Networks”, In Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services, 2004 (doi: 10.1145/990064.990080).

[19] G. Xing, T. Wang, W. Jia, and M. Li, “Rendezvous Design Algorithms for Wireless Sensor Networks with a Mobile Base Station”, In Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2008 (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.163.619&rep=rep1&type=pdf).

[20] G. Xing, T. Wang, Z. Xie, and W. Jia, “Rendez vous Planning in Mobility-assisted Wireless Sensor Networks”, In Proceedings of the 28th IEEE International Real-Time Systems Symposium, 2007 (doi: 10.1109/RTSS.2007.44).

[21] J. Rao and S. Biswas, “Joint Routing and Navigation Protocols for Data Harvesting in Sensor Networks”, In Proceedings of the 5th IEEE International Conference on Mobile Ad-hoc and Sensor Systems, 2008 (doi: 10.1109/MAHSS.2008.4660041).

[22] G. Sklyarenko, “AODV Routing Protocol“, Seminar Technische Informatik, Institut fur Informatik, Freie Universitat Berlin, 2006.

[23] C. Konstantopoulos, G. Pantziou, D. Gavalas, A. Mpitziopoulos, B. Mamalis, “A Rendezvous-Based Approach for Energy-Efficient Sensory Data Collection from Mobile Sinks”, IEEE Transactions on Parallel and Distributed Systems, 2012 (doi: 10.1109/TPDS.2011.237).

[24] C. Konstantopoulos, B. Mamalis, G. Pantziou, and V. Thanasias, “Watershed-Based Clustering for Energy Efficient Data Gathering in Wireless Sensor Networks with Mobile Collector”, Euro-Par 2012 Parallel Processing, 2012 (doi: 10.1007/978-3-642-32820-6_75).

[25] J. Rao, S. Biswas, “Network-assisted Sink Navigation for Distributed Data Gathering: Stability and Delay-energy Trade-offs”, Computer Communications, 2010 (doi: 10.1016/j.comcom.2009.08.009).

[26] R. Sugihara, R. Gupta, “Optimal Speed Control of Mobile Node for Data Collection in Sensor Networks”, IEEE Transactions on Mobile Computing, 2010 (doi: 10.1109/TMC.2009.113).

[27] S. Basagni, A. Carosi, C. Petrioli, C. Phillips, “Coordinated and controlled mobility of multiple sinks for maximizing the lifetime of Wireless Sensor Networks”, Wireless Networks, 2011 (doi: 10.1007/s11276-010-0313-8).

[28] B. Galilée, F. Mamalet, M. Renaudin, P. Coulon, “Parallel Asynchronous Watershed Algorithm”, IEEE Transactions on Parallel and Distributed Systems, 2007 (doi: 10.1109/TPDS.2007.253280).

[29] R. Sugihara and R. Gupta, “Improving the Data Delivery Latency in Sensor Networks with Controlled Mobility,” Proc. Fourth IEEE International Conference of Distributed Computing in Sensor Systems (DCOSS), 2008 (doi: 10.1007/978-3-540-69170-9_26).

[30] C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks,” Proc. ACM MobiCom, 2000 (doi: 10.1145/345910.345920).

[31] L. Shi, B. Zhang, H. T. Mouftah, and J. Ma, “DDRP: An efficient data-driven routing protocol for wireless sensor networks with mobile sinks”, International Journal of Communication Systems, 2012 (doi: 10.1002/dac.2315).

[32] L. He, J. Pan, and J. Xu, “A Progressive Approach to Reducing Data Collection Latency in Wireless Sensor Networks with Mobile Elements”, IEEE Transactions on Mobile Computing, 2012 (doi: 10.1109/TMC.2012.105).

[33] K. Almi'ani, A. Viglas, L. Libman, "Energy-Efficient Data Gathering with Tour Length-Constrained Mobile Elements in Wireless Sensor Networks", 35th Annual IEEE Conference on Local Computer Networks, 2010 (doi: 10.1109/LCN.2010.5735777).

[34] W. Liu, K. Lu, J. Wang, G. Xing, and L. Huang, "Performance Analysis of Wireless Sensor Networks With Mobile Sinks", IEEE Transactions on Vehicular Technology, 2012 (doi: 10.1109/TVT.2012.2194747).

[35] W. Liang, P. Schweitzer, and Z. Xu, "Approximation Algorithms for Capacitated Minimum Forest Problems in Wireless Sensor Networks with a Mobile Sink", IEEE Transactions on Computers, 2012 (doi: 10.1109/TC.2012.124).

[36] Y. Yun, Y. Xia, B. Behdani and J. C. Smith, "Distributed Algorithm for Lifetime Maximization in Delay-Tolerant Wireless Sensor Network with Mobile Sink", IEEE Transactions on Mobile Computing, 2010 (doi: 10.1109/CDC.2010.5717520).

[37] Y. Gu, Y. Ji, J. Li, and B. Zhao, "ESWC: Efficient Scheduling for the Mobile Sink in Wireless Sensor Networks with Delay Constraint", IEEE Transactions on Parallel and Distributed Systems, 2012 (doi: 10.1109/TPDS.2012.210).

[38] Sun Labs, “Sun SPOT Owner’s Manual, Red Release 5.0”, Sun Microsystems, 2009.

[39] Systronix TrackBot, “ TrackBot with Robot Area Network and Digital Reflexes”, Systronix, http://www.systronix.com/trackbot

[40] Wikipedia, “Wireless sensor network“, http://en.wikipedia.org/wiki/Wsn.

[41] Wikipedia, “List of ad hoc routing protocols”, http://en.wikipedia.org/wiki/Ad_hoc_protocol_list

[42] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, “Introduction to Algorithms, Third Edition”, The MIT Press, 2009.

Εργαλεία Ανάπτυξης - Εφαρμογές

- Sun SPOT Manager Application, “Managing Sun SPOTs properties and software”, http://www.sunspotworld.com/SPOTManager/index.html
- NetBeans IDE with Sun SPOTs module, “Developing Sun SPOTs applications and coding”, http://netbeans.org/
- Apache Ant, “Building and deploying Sun SPOTs applications”, http://ant.apache.org/
- Solarium Application, “Emulator and Remote Management of Sun SPOTs”, http://www.sunspotworld.com/SPOTManager/index.html

Χρήσιμες Διευθύνσεις

- Sun SPOT World, “Official Sun SPOT Website”, http://www.sunspotworld.com/index.html
- Sun SPOT World, “Community / Support Forum for Sun SPOTs”, http://www.sunspotworld.com/forums