ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ – ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ & ΔΙΚΤΥΩΝ

ΟΛΟΚΛΗΡΩΜΕΝΟ ΣΥΣΤΗΜΑ ΠΡΟΣΟΜΟΙΩΣΗΣ ΕΝΕΡΓΕΙΑΚΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΑΣΥΡΜΑΤΩΝ ΠΡΩΤΟΚΟΛΛΩΝ

ΓΕΡΑΣΙΜΟΣ ΦΡΑΓΚΟΣ

2007

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κο Γεώργιο Σταμούλη και τον Δρ. Παναγιώτη Κίκιρα για την καθοδήγησή τους στην εκπόνηση αυτής της εργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1 - ΕΣΑΓΩΓΗ

1.1 Ασύρματα Δίκτυα Αισθητήρων

Οι τεχνολογικές εξελίξεις τόσο στη σχεδίαση ψηφιακών κυκλωμάτων όσο και στις ασύρματες επικοινωνίες, επέτρεψε την κατασκευή μικρών, φθηνών, χαμηλής ενεργειακής κατανάλωσης αισθητήρες. Η δυνατότηταν αυτών να επικοινωνούν μεταξύ τους και να αλληλεπιδρούν με το περιβάλλον, άνοιξε το δρόμο για μια πληθώρα εφαρμογών σε ποικίλους τομείς, όπως σε στρατιωτικές εφαρμογές, στην παρακολούθηση περιβαλλοντικών παραγόντων, στην ασφάλεια, στην αγροτική παραγωγή κλπ.

Δεδομένων των περιορισμών που επιφέρει το μικρό μέγεθος των αισθητήρων στους διαθέσιμους πόρους (ενέργεια, ασύρματη εμβέλεια κλπ) μαζί με το γεγονός πως μπορεί ένας σημαντικός αριθμός τέτοιων συσκευών (εκατοντάδες ή ακόμη και χιλιάδες) να αναπτυχθεί σε κάποια περιοχή, κάνει σημαντικό το θέμα της δρομολόγησης των αισθητηριακών δεδομένων σε κάποιο σταθμό βάσης για συλλογή και επεξεργασία.

1.2 Σκοποί και στόχοι

Σκοπός της παρούσας διπλωματικής εργασίας είναι η μελέτη θεμάτων δρομολόγησης σε ασύρματα δίκτυα αισθητήρων, τα ιδιαίτερα χαρακτηριστικά τους, τα πλεονεκτήματα και τα μειονεκτήματα. Θα παρουσιαστεί ένα περιβάλλον προσομοίωσης για ασύρματα δίκτυα αισθητήρων, το J-SIM, με τη βοήθεια του οποίου θα γίνει παρουσίαση και σύγκριση κάποιων γνωστών πρωτοκόλλων δρομολόγησης καθώς και ενός νέου προτεινόμενου, με κύρια χαρακτηριστικά την ιεραρχικότητα και τη δυνατότητα προσδιορισμού της θέσης του κάθε αισθητήρα. Ακόμη θα παρουσιαστούν τρόποι εξασφάλισης της βέλτιστης ενεργειακής απόδοσης αυτών των δικτύων.

1.3 Δομή της παρούσας εργασίας

Στην παρούσα εργασία παρουσιάζονται τα εξής:

1. Μια εισαγωγή στα ασύρματα δίκτυα αισθητήρων με έμφαση στα ιδιαίτερα γνωρίσματά τους, στις εφαρμογές για τις οποίες είναι κατάλληλα και στην αρχιτεκτονική και τα χαρακτηριστικά τους.

2. Παρουσίαση και σύγκριση αλγορίθμων και πρωτοκόλλων δρομολόγησης σε ασύρματα δίκτυα αισθητήρων. Αναφορά στα αντιπροσωπευτικότερα από αυτά, στα πλεονεκτήματα και στα μειονεκτήμτά τους.

3. Παρουσίαση των πρωτοκόλλων που θα προσομοιώσουμε: One-Hop, Multi-Hop, LEACH καθώς και το προτεινόμενο ιεραρχικό πρωτόκολλο σε δύο παραλλαγές.

4. Η σημασία των εργαλείων πρωσομοίωσης είναι πολύ σπουδαία για την έρευνα και ανάπτυξη των συστημάτων και ειδικότερα των δικτύων δεδομένων (ενσύρματων και ασύρματων) όπου έχουν παίξει σημαντικό ρόλο. Στη συνέχεια λοιπόν γίνεται παρουσίαση του εργαλείου προσομοίωσης J-SIM, τα χαρακτηριστικά του, τα πλεονεκτήματα αλλά και τα μειονεκτήματα αυτού. Κατόπιν θα παρουσιαστεί ο τρόπος υλοποίησης στο J-SIM των πρωτοκόλλων του προηγούμενου κεφαλαίου προκειμένου να προσομοιωθεί η συμπεριφορά τους.

5. Στο τελευταίο τμήμα της εργασίας θα δούμε τα αποτελέσματα των προσομοιώσεων, όπου θα φανούν τα χαρακτηριστικά του κάθε πρωτοκόλλου, όπως η κατανάλωση ενέργειας, η αξιοπιστία στη μεταφορά των αισθητηριακών δεδομένων, η καθυστέρηση στην άφιξη των δεδομένων στο Σταθμό Βάσης.

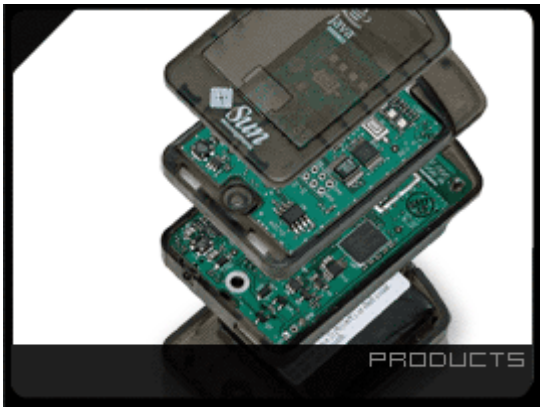
6. Η ολοκλήρωση της εργασίας γίνεται με την παρουσίαση των συμπερασμάτων όπως αυτά προκύπτουν από τα προηγούμενα τμήματα, αλλά και κάποιες πιθανές προτάσεις και σκέψεις για τη βελτίωση της ενεργειακής κυρίως, συμπεριφοράς των παρουσιαζόμενων πρωτοκόλλων, όπου θα μπορούσαν να αποτελέσουν μελλοντική εργασία. Τέλος παρατίθενται οι βιβλιογραφικές αναφορές σε εργασίες οι οποίες χρησιμοποιήθηκαν.

Κεφάλαιο 2 – ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ

2.1 Εισαγωγή

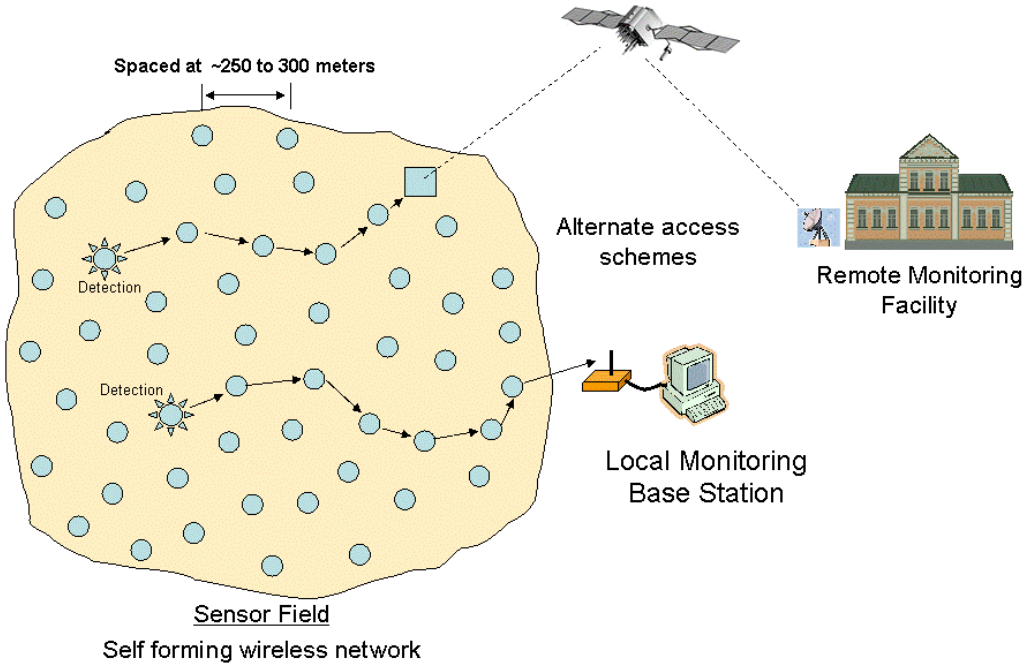
Όπως αναφέρθηκε και προηγούμενα, η αλματώδης εξέλιξη στους τομείς της ηλεκτρονικής και των τηλεπικοινωνιών επέτρεψαν την ανάπτυξη χαμηλού κόστους και χαμηλής κατανάλωσης, κόμβων αισθητήρων, με κύρια χαρακτηριστικά το μικρό μέγεθος και τη δυνατότητα να επικοινωνούν σε μικρές αποστάσεις (Εικόνες 1, 2). Οι κόμβοι αυτοί αποτελούνται από τμήματα καταγραφής του εξωτερικού αισθητηριακού ερεθίσματος ή συμβάντος, επεξεργασίας των καταγραφόμενων δεδομένων και αποστολής αυτών μέσω ενός κατάλληλα διαμορφωμένου επικοινωνιακού φορέα. Η ανάπτυξη πλήθους, πυκνά τοποθετημένων τέτοιων κόμβων, πολύ κοντά στο φαινόμενο που καλούνται να παρατηρήσουν, ακόμα και σε δύσβατες ή ακατάλληλες για τον άνθρωπο περιοχές, αλλά και η τυχαία τοπολογία δημιουργεί ένα ασύρματο δίκτυο με πολύ ιδιαίτερα χαρακτηριστικά.

Εικόνα 1 - Mica Mote



Εικόνα 2 - Sun SPOT

Τα ιδιαίτερα αυτά χαρακτηριστικά των δικτύων αισθητήρων τα καθιστούν κατάλληλα για μια σειρά από εφαρμογές σε ένα πλήθος περιοχών της ανθρώπινης δραστηριότητας. Χαρακτηρηστικοί τομείς ανάπτυξης εφαρμογών αισθητήρων είναι η υγεία, με προσανατολισμό στην ανάπτυξη δικτύων αισθητήρων τα οποία θα καταγράφουν την κατάσταση του ασθενή και θα ενημερώνουν συνεχώς το ιατρικό και νοσηλευτικό προσωπικό ή εξειδικευμένες συσκευές καταγραφής και παρακολούθησης, το φυσικό περιβάλλον με δίκτυα έγκαιρης προειδοποίησης δασικών πυρκαγιών ή ακραίων καιρικών φαινομένων, το εργασιακό περιβάλλον π.χ στον βιομηχανικό έλεγχο, στο ατομικό περιβάλλον με μια σειρά από εφαρμογές όπως οι αυτοματισμοί στο σπίτι (Έξυπνα Σπίτια) αλλά και στις στρατιωτικές εφαρμογές όπου τα ιδιαίτερα χαρακτηριστικά των δικτύων αισθητήρων, δηλαδή, η δυνατότητα ταχείας ανάπτυξής τους, η ικανότητα αυτοοργάνωσης, και η αντοχή στην αστοχία μέρους του δικτύου τους, τα καθιστά μια πολύ ελκυστική τεχνολογία για τα στρατιωτικά συστήματα Διοίκησης, Ελέγχου, Επικοινωνιών, Υπολογιστών, κλπ.



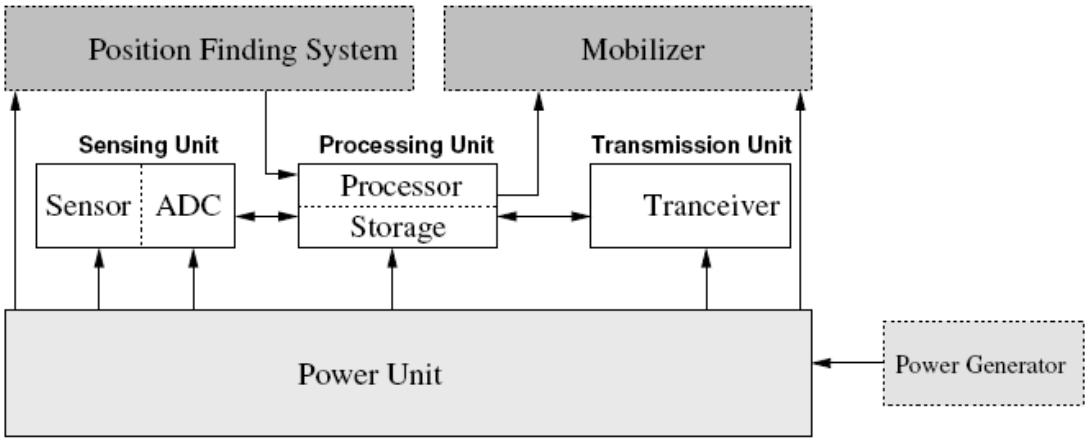
Εικόνα 3 - Ασύρματο Δίκτυο Αισθητήρων

2.2 Κόμβος Ασύρματου Ασθητήρα

Η βασική δομική μονάδα των ασυρμάτων δικτύων αισθητήρων, είναι ο κόμβος του ασύρματου αισθητήρα.

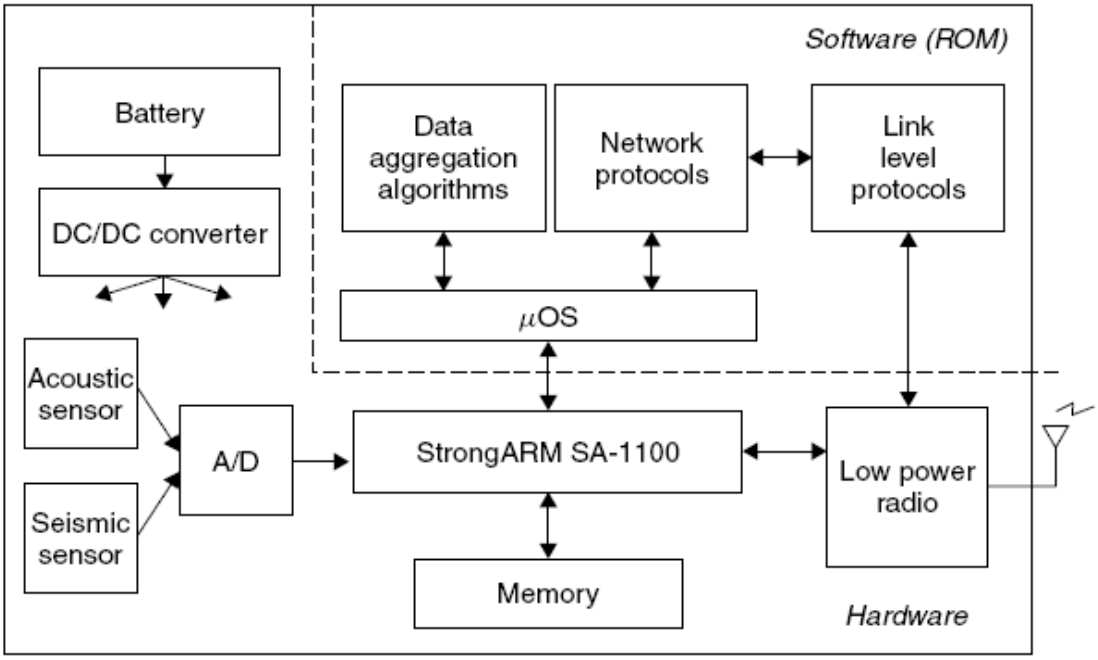
Η βασική αρχιτεκτονική ενός τέτοιου κόμβου περιλαμβάνει τα παρακάτω βασικά υποσυστήματα (Εικόνα 4):

- Υποσύστημα αισθητήρων
- Υποσύστημα Επεξεργασίας
- Υποσύστημα Επικοινωνιών
- Υποσύστημα Τροφοδοσίας
- Υποσύστημα Εύρεσης Θέσης (προαιρετικά)



Εικόνα 4 - Αρχιτεκτονική κόμβου αισθητήρα

Εκτός βέβαια από το υλικό που απαρτίζει έναν κόμβο αισθητήρα, σημαντικό ρόλο παίζει και το λογισμικό που είναι υπεύθυνο για τις λειτουργίες του, όπως το δικτυακό πρωτόκολλο, την επεξεργασία των αισθητηριακών δεδομένων, τη διαχείριση της ενέργειας κλπ. Συνήθως αποθηκεύεται σε κάποια μνήμη τύπου ROM όπως στον αισθητήρα μAmps (Εικόνα 5).



Εικόνα 5 - Αρχιτεκτονική Υλικού και Λογισμικού κόμβου αισθητήρα τύπου μAmps

Ο κόμβος αισθητήρα διαθέτει χαρακτηριστικά που αποτελούν ουσιαστικά δείκτες ποιότητας και λειτουργικότητας:

- Ενεργειακή κατανάλωση: ο χρόνος ζωής των κόμβων πρέπει να είναι ιδιαίτερα μεγάλος οπότε πρέπει να καταναλώνουν πάρα πολύ μικρή ποσότητα ενέργειας για κάθε λειτουργία (λειτουργία αισθητήρα, συλλογή, επεξεργασία, αποστολή και λήψη δεδομένων).

- Ευελιξία: λόγω της ποικιλίας των εφαρμογών που μπορούν να χρησιμοποιηθούν, απαιτείται η αρχιτεκτονική σχεδίασης τους να είναι ευέλικτη και ευπροσάρμοστη. Θα πρέπει, δηλαδή, να είναι εύκολη η μεταβολή παραμέτρων όπως ο ρυθμός δειγματοληψίας, ο χρόνος απόκρισης, το είδος και η μέθοδος επεξεργασίας.

- Ασφάλεια: εξαιτίας της ευκολίας υποκλοπής, αλλά και παρεμβολής του ασύρματου καναλιού, για να διατηρηθούν τα δεδομένα αναλλοίωτα ο κόμβος πρέπει να είναι ικανός να εκτελεί πολύπλοκους αλγορίθμους κρυπτογράφησης και αυθεντικοποίησης.

- Δυνατότητα επικοινωνίας: βασικό χαρακτηριστικό των κόμβων είναι η ικανότητα τους να επικοινωνούν. Βασικά χαρακτηριστικά της επικοινωνίας τους είναι ο ρυθμός μετάδοσης, η διαμόρφωση, η κατανάλωση ενέργειας και η ακτίνα μετάδοσης.

- Υπολογιστική Ισχύς: σημαντικός παράγοντας απόδοσης ενός κόμβου είναι η υπολογιστική ισχύς του. Θα πρέπει η μονάδα επεξεργασίας να μπορεί να ανταπεξέλθει σε εργασίες όπως η επεξεργασία δεδομένων (καταγραφή, μετατροπή από αναλογικό σε ψηφιακό, κρυπτογράφηση – αποκρυπτογράφηση κλπ), και η εκτέλεση των τηλεπικοινωνιακών πρωτοκόλλων.

- Συγχρονισμός: για την υποστήριξη χρονικά συσχετισμένων δεδομένων από διαφορετικούς κόμβους του δικτύου, απαιτείται αυτά να διατηρούν μια μέθοδο συντονισμού πάρα πολύ μεγάλης ακρίβειας π.χ. με τη χρήση συστήματος γεωγραφικού προσδιορισμού θέσης (GPS), ή με ειδικής σηματοδοσίας υλοποιημένη σε κατάλληλο πρωτόκολλο [2] κ.λ.π.

- Μέγεθος και κόστος: το μέγεθος και το κόστος του κάθε κόμβου παίζει σημαντικό ρόλο στην συνολική ευκολία ανάπτυξης ενός δικτύου. Δεδομένου ότι ο αριθμός ενός δικτύου αισθητήρων μπορεί να ανέλθει σε εκατοντάδες ή ακόμη και χιλιάδες, το κόστος μονάδας είναι ουσιαστικός παράγοντας που ισοδυναμεί με την ανάπτυξη περισσότερων κόμβων σε κάποιο πεδίο παρατήρησης για τον καλύτερο εντοπισμό του επιθυμητού γεγονότος. Ακόμη, μικροί σε μέγεθος κόμβοι μπορούν να τοποθετηθούν σε

περισσότερα σημεία και να χρησιμοποιηθούν σε περισσότερες διαφορετικές εφαρμογές.

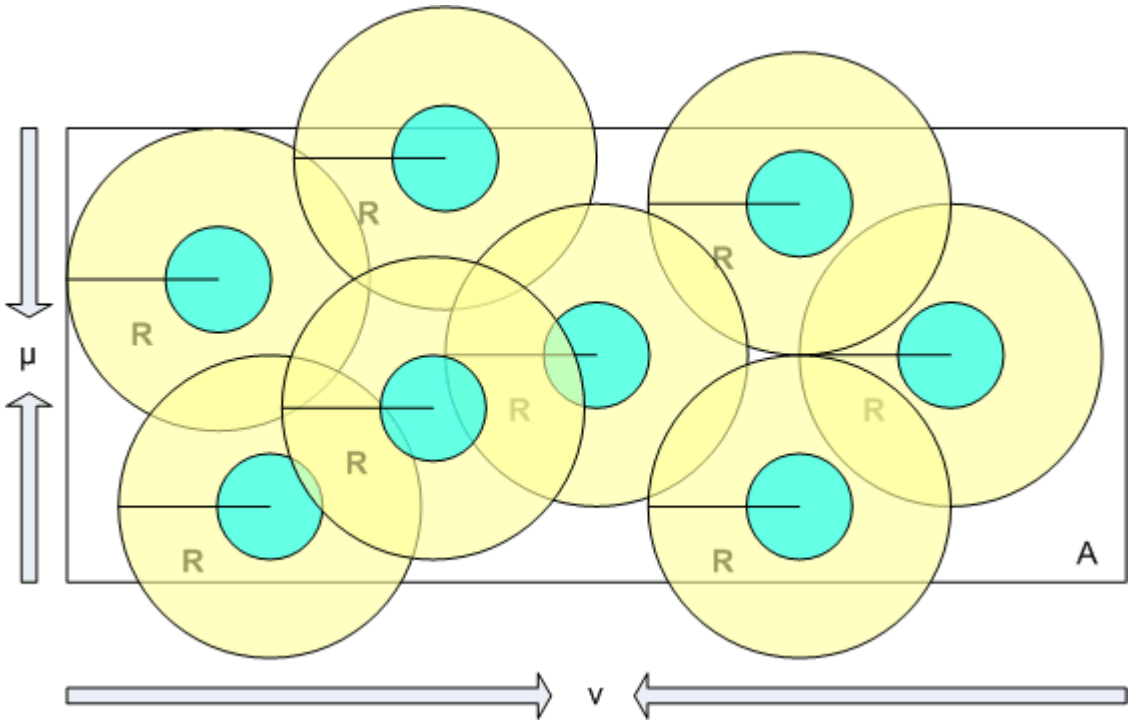
2.3 Στοιχεία Αρχιτεκτονικής Ασύρματων Δικτύων Ασθητήρων

Όπως ήδη έχει αναφερθεί, η ανάπτυξη ενός δικτύου αισθητήρων μπορεί να περιλαμβάνει από λίγους (ακόμη κι έναν) κόμβους έως και χιλιάδες στην περιοχή ενδιαφέροντος. Η πυκνότητα μ(R) αισθητήρων σε μια περιοχή δίνεται από την σχέση [1]:

$$\mu(R) = \frac{(NR^2\pi)}{A}$$

όπου N ο αριθμός των αισθητήρων στην περιοχή A και R είναι η μέγιστη ακτίνα του πομπού του κόμβου.

Στην Εικόνα 6, βλέπουμε ένα δίκτυο αισθητήρων το οποίο έχει αναπτυχθεί σε μια περιοχή *A* διαστάσεων $\mu \times \nu$.



Εικόνα 6 - Ανάπτυξη δικτύου αισθητήρων

Το υποσύστημα ασύρματης μετάδοσης δεδομένων είναι το πλέον σημαντικό υποσύστημα ενός κόμβου, αφού αποτελεί τον μεγαλύτερο καταναλωτή ενέργειας, φθάνοντας έως και το 80% της συνολικής ενέργειας του κόμβου. Αποτελεί, λοιπόν, το σημαντικότερο παράγοντα καθορισμού της διάρκειας ζωής του κόμβου και κατ' επέκταση του δικτύου.

Η φύση της ασύρματης μετάδοσης, δηλαδή η χρήση του ηλεκτρομαγνητικού φάσματος, παίζει σημαντικό ρόλο στον σχεδιασμό των επικοινωνιών του συστήματος. Το κυριότερο μειονέκτημα της ασύρματης μετάδοσης σε σχέση με την ενσύρματη, είναι ο αυξημένος ρυθμός εμφάνισης λαθών ανά εκπεμπόμενο bit (Bit Error Rate – BER). Ακόμη σημαντικές είναι οι παρεμβολές του σήματος από γειτονικούς κόμβους, αλλά και η εξασθένηση του εξαιτίας των φαινόμενων της διάθλασης, αντανάκλασης, και σκέδασης τα οποία οφείλονται στον τρόπο με τον οποίο διαδίδεται το ηλεκτρομαγνητικό κύμα στο χώρο.

Οι διατιθέμενες για χρήση συχνότητες στα ασύρματα δίκτυα αισθητήρων τοποθετούνται στις συχνότητες που αναγνωρίζονται ως Βιομηχανικές, Επιστημονικές και Ιατρικές συχνότητες (Industrial, Scientific and Medical Bands-ISM) οι τιμές των οποίων για την Ευρώπη φαίνονται στον Πίνακα 1.

Συχνότητα (MHz)	**Εύρος Ζώνης (MHz)**	**Διαχωρισμός Καναλιών**	**Τεχνική Διαμόρφωσης**	**Ρυθμός Μετάδοσης (Bit rate)**
433.5-437.9	**1.740**	**Δεν Καθορίζεται**	**free**	**free**
868.0-868.6	**0.600**	**25 kHz**	**free**	**free**
		100 kHz	**SS**	**free**
2400-2483.5	**83.5**	**100 kHz**	**FHSS/DSSS**	**>250 kbps**

Πίνακας 1 - ETSI Frequency Spectrum

Για την ελαχιστοποίηση της κατανάλωσης ενέργειας του πομποδέκτη έχουν αναπτυχθεί μια σειρά τεχνικών τόσο στο επίπεδο του υλικού, όσο και στο επίπεδο του λογισμικού.

Σε επίπεδο υλικού, η βασικότερη προσπάθεια εντοπίζεται στην ανάπτυξη πομποδεκτών οι οποίοι να είναι ικανοί να λειτουργούν σε διαφορετικές

ενεργειακές στάθμες, ανάλογα με την εργασία που εκτελούν (αποστολή – λήψη, αναμονή, και «ύπνωση» (sleep state)).

Η υλοποίηση της διαχείρισης εναλλαγής φάσεων λειτουργίας γίνεται, σε επίπεδο λογισμικού, με την ανάπτυξη κατάλληλων πρωτοκόλλων δρομολόγησης και MAC. Τα χαρακτηριστικά της ασύρματης μετάδοσης που διαμορφώνουν την απόδοση και την ενεργειακή συμπεριφορά ενός κόμβου είναι:

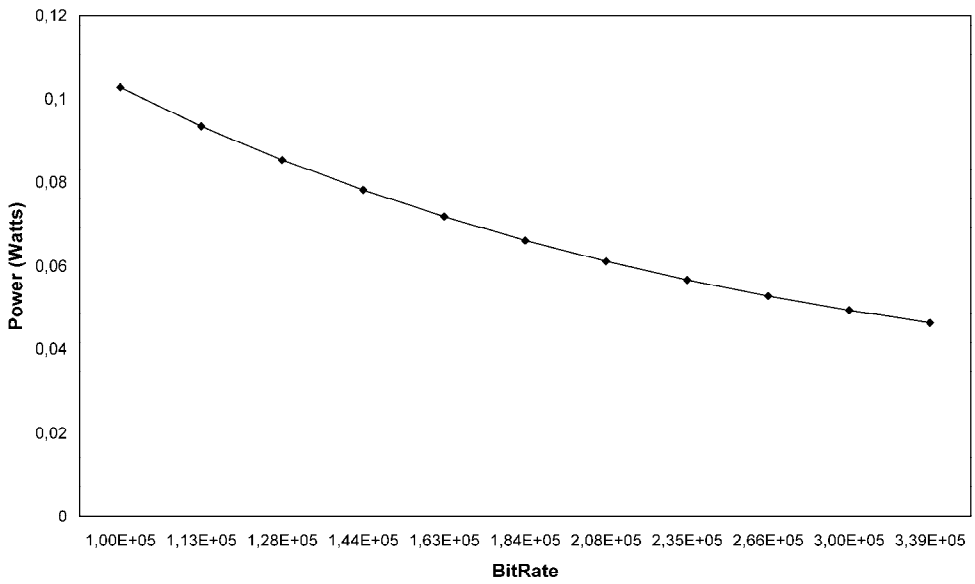
- Ακτίνα Εκπομπής: η ακτίνα εκπομπής ενός πομπού συσχετίζεται με την ενέργεια του σήματος P που λαμβάνεται από έναν δέκτη, ο οποίος βρίσκεται σε απόσταση r από τον πομπό δίνεται από με μια σχέση της μορφής $P(r) = \lambda \cdot r^{-\alpha}$ όπου λ είναι μια σταθερά και ο δείκτης α χαρακτηρίζει το περιβάλλον μετάδοσης. Ο δείκτης α παίρνει συνήθως τιμές από 2 έως και 5, με το 2 να αντιστοιχεί σε επίπεδο χωρίς εμπόδια περιβάλλον και το 5 να αντιστοιχεί σε αστικό περιβάλλον. Επιπλέον παράγοντες που επηρεάζουν την ακτίνα εκπομπής είναι η ευαισθησία του δέκτη, το κέρδος και το είδος της κεραίας και ο μηχανισμός κωδικοποίησης του καναλιού.

- Τύπος διαμόρφωσης: βασικό χαρακτηριστικό οποιασδήποτε RF συσκευής είναι ο μηχανισμός διαμόρφωσης. Στα ασύρματα δίκτυα χρησιμοποιούνται τεχνικές ψηφιακής διαμόρφωσης με τα παρακάτω πλεονεκτήματα:
    - Αντοχή στον θόρυβο.
    - Ευκολότερη πολυπλεξία της πληροφορίας.
    - Υποστήριξη τεχνικών ελέγχου και διόρθωσης λαθών, αλλά και κωδικοποίησης και κρυπτογράφησης του σήματος.

- Ρυθμός μετάδοσης: οι περισσότερες εφαρμογές στα ασύρματα δίκτυα αισθητήρων χαρακτηρίζονται από ρυθμούς μετάδοσης οι οποίοι κυμαίνονται από 10 - 250 Kbps. Η σχέση μεταξύ ρυθμού μετάδοσης και κατανάλωσης ενέργειας φαίνεται στην πιο κάτω εξίσωση:

$$RF\_Power\_Tx = RF\_Power\_Elec + (Number\_of\_bits\_Tx / BitRate) \cdot RF\_Power\_ActiveState$$

  όπου RF_Power_Tx η καταναλισκόμενη ισχύς, $Number\_of\_bits\_Tx$ ο αριθμός των προς αποστολή bits, $BitRate$ ο ρυθμός μετάδοσης των δεδομένων και $Rf\_Power\_ActiveState$ η κατανάλωση ισχύος του πομπού για την εκπομπή όπως αυτή δίνεται από τον κατασκευαστή του. Όπως φαίνεται στο πιο κάτω διάγραμμα (Εικόνα 7), όσο αυξάνει ο ρυθμός μετάδοσης τόσο μειώνεται η καταναλισκόμενη ενέργεια. Αυτό συμβαίνει γιατί ο μεγαλύτερος ρυθμός μετάδοσης σημαίνει ταχύτερη αποστολή των δεδομένων άρα και μικρότερο χρόνο λειτουργίας του πομπού στην ενεργή κατάσταση λειτουργίας, εκεί δηλαδή που υπάρχει η μεγαλύτερη κατανάλωση ενέργειας.



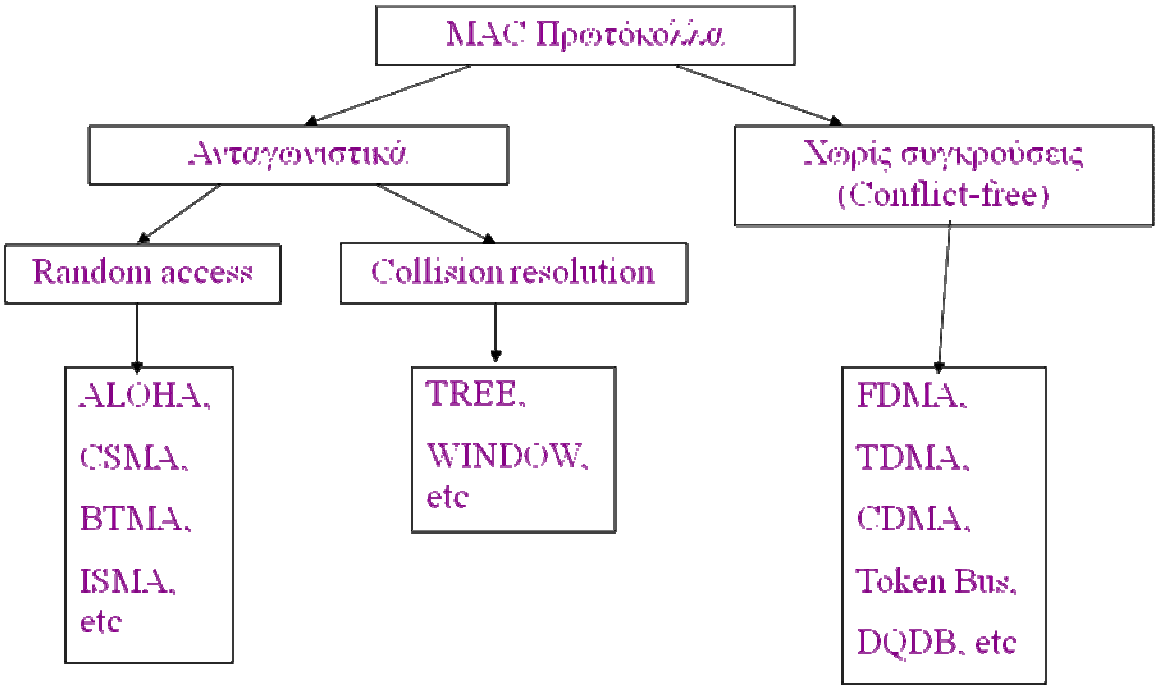
Εικόνα 7 - Κατανάλωση ισχύος σε σχέση με τον ρυθμό μετάδοσης δεδομένων

- MAC πρωτόκολλα: Οι ιδιαιτερότητες των δικτύων ασυρμάτων αισθητήρων, δεν κάνουν κατάλληλα τα MAC πρωτόκολλα που χρησιμοποιούνται ήδη σε άλλα είδη ασυρμάτων δικτύων [36], [33]. Έτσι πρέπει οι σχεδιαστές πρωτοκόλλων να λαμβάνουν υπόψη τις πιο κάτω απαιτήσεις:
    - Αποφυγή Συγκρούσεων
    - Ελαχιστοποίηση της Κατανάλωσης Ισχύος
    - Επεκτασιμότητα και κλιμακωσιμότητα
    - Προσαρμοστικότητα και αντοχή σε μεταβολές
        - Συνθήκες καναλιού
        - Αστοχία κόμβων
        - Περιβαλλοντικές αλλαγές
    - Αξιοπιστία
    - Βέλτιστη Χρησιμοποίηση του Καναλιού Εκπομπής
    - Καθυστέρηση μετάδοσης
    - Δικαιοσύνη (fairness)
    - Throughput

Όπως έχουμε ήδη τονίσει η ελαχιστοποίηση της κατανάλωσης ενέργειας είναι σημαντικότατη και αποτελεί έναν από τους βασικότερους στόχους. Από την πλευρά του MAC πρωτοκόλλου η κατανάλωση ενέργειας εντοπίζεται:

1. στην παραμονή στην κατάσταση αναμονής λήψης δεδομένων (idle), απαραίτητο όμως για τα CSMA πρωτόκολλα
2. στην ακούσια ακρόαση-λήψη δεδομένων (overhearing) που συμβαίνει όταν ένας κόμβος λαμβάνει δεδομένα, τα οποία προορίζονται για άλλο αποδέκτη
3. στις συγκρούσεις πακέτων (πρόβλημα του κρυμμένου κόμβου)
4. στην εσφαλμένη παρεμπόδηση εκπομπής (πρόβλημα του εκτεθιμένου κόμβου)
5. στο overhead από τα χρησιμοποιούμενα από το πρωτόκολλο πακέτα ελέγχου (π.χ. RTS/CTS ή DATA/ACK)

Τα MAC πρωτόκολλα διακρίνονται σε δύο γενικές κατηγορίες, όπως φαίνονται στην Εικόνα 8, τα οποία έχουν στόχο τον έλεγχο του πομποδέκτη, ώστε να αποφεύγεται ή να μειώνεται η σπατάλη ενέργειας από τις παραπάνω αναφερθείσες αιτίες.



Εικόνα 8 - Κατηγορίες MAC Πρωτοκόλλων

Στην κατηγορία Χωρίς-Συγκρούσεις (conflict-free) ανήκουν τα πρωτόκολλα προγραμματισμένης πρόσβασης. Η βασική ιδέα πίσω από αυτά είναι η αποφυγή συγκρούσεων μέσω της προγραμματισμένης πρόσβασης των κόμβων στο μέσο μετάδοσης είτε με καταμερισμό χρόνου (Time Division Multiple Access – TDMA), είτε συχνότητας (Frequency Division Multiple Access – FDMA), είτε με βάση έναν μοναδικό κωδικό (Code Division Multiple Access – CDMA). Στην άλλη κατηγορία, των ανταγωνιστικών πρωτοκόλλων περιλαμβάνονται πρωτόκολλα όπως τα ALOHA, 802.11, B-MAC όπου η αποφυγή συγκρούσεων επιτυγχάνεται συνήθως με ακρόαση του μέσου και μετάδοση αν αυτό δεν βρίσκεται σε χρήση.

- Κεραίες: Επειδή οι κόμβοι των ασύρματων δικτύων αισθητήρων είναι πάρα πολύ μικροί σε μέγεθος και με βάση δύο από τα κριτήρια αξιολόγησης των κόμβων που είναι το κόστος και η κατανάλωση ισχύος, οι κεραίες μπορούν να παίξουν σημαντικό ρόλο σε έναν επιτυχημένο σχεδιαστικά κόμβο. Επειδή η τοποθέτηση των κόμβων στην περιοχή παρατήρησης, σε αρκετές περιπτώσεις, γίνεται τυχαία και με μη ελεγχόμενο τρόπο ως προς την θέση της κεραίας (ενδέχεται να βρίσκεται προς το έδαφος ή να «βλέπει» σε κάποιο εμπόδιο) είναι πιθανόν να μην υπάρχει ισοτροπική κατανομή του εκπεμπομένου σήματος προς όλες τις διευθύνσεις. Για το λόγο αυτό προτιμώνται οι πολύκατευθυντικές κεραίες ως καταλληλότερες για χρήση στα δίκτυα αισθητήρων, αν και τελευταία εμφανίζονται προτάσεις στην βιβλιογραφία για την χρήση «έξυπνων κεραιών», οι οποίες φέρονται να εξασφαλίζουν μεγαλύτερο κέρδος και καλύτερη απόδοση σε σχέση με τις πολυκατευθυντικές [4].

Κεφάλαιο 3 – ΠΡΩΤΟΚΟΛΛΑ ΔΡΟΜΟΛΟΓΗΣΗΣ ΣΕ ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ

3.1 Εισαγωγή

Η υιοθέτηση των ασύρματων δικτύων αισθητήρων σε ολοένα και αυξανόμενο αριθμό εφαρμογών και η ταχύτατη εξέλιξή τους, οδήγησαν στην ανάπτυξη πολλών νέων πρωτοκόλλων ειδικά σχεδιασμένων για δίκτυα αισθητήρων όπου η ενέργεια αποτελεί σημαντικό παράγοντα.

Συνήθως, η ανάπτυξη ενός ασύρματου δικτύου αισθητήρων σε μια περιοχή παρατήρησης, γίνεται με σκοπό τη συλλογή δεδομένων που αφορούν στο φαινόμενο και την προώθηση των αποτελεσμάτων που προκύπτουν από την επεξεργασία των παραπάνω δεδομένων σε έναν σταθμό βάσης του δικτύου. Όμως η διαδικασία της δρομολόγησης των δεδομένων από ένα κόμβο του δικτύου στο σταθμό βάσης είναι μια σύνθετη διαδικασία, που οφείλει την πολυπλοκότητα της στα ιδιαίτερα χαρακτηριστικά των ασύρματων δικτύων αισθητήρων. Κάποια από αυτά τα χαρακτηριστικά είναι:

- Η μεγάλη πυκνότητα ανάπτυξης των κόμβων σε μια περιοχή παρατήρησης: Ο ιδιαίτερα μεγάλος αριθμός κόμβων σε μια περιοχή παρατήρησης καθιστά απαγορευτική την χρήση ενός μοντέλου διευθυνσιοδότησης, όπως π.χ στα δίκτυα δεδομένων, που βασίζονται στο πρωτόκολλο IP (Internet Protocol), αφού το διαχειριστικό κόστος της λειτουργίας ενός τέτοιου μοντέλου είναι απαγορευτικό. Έτσι, δεν είναι δυνατή η χρήση στα δίκτυα αισθητήρων των κλασσικών, προερχομένων από τα IP δίκτυα, πρωτοκόλλων (π.χ. OSPF, BGP, IGRP).

- Το μοντέλο διακίνησης της πληροφορίας: Στα δίκτυα αισθητήρων σε αντίθεση με τα υπόλοιπα δίκτυα μεταφοράς δεδομένων η ροή της πληροφορίας είναι του τύπου «από πολλούς» (κόμβους, επικεφαλής ομάδων) «προς έναν» (επικεφαλής ομάδας, σταθμός βάσης) και το αντίστροφο, γεγονός το οποίο περιορίζει, όσο φτάνουμε πιο κοντά στον «έναν» τις δυνατότητες επιλογής δρομολογίων προώθησης των δεδομένων.

- Το είδος των δεδομένων που παράγονται: Τα παραγόμενα δεδομένα σε μια συγκεκριμένη περιοχή από μια ομάδα κόμβων και τα οποία αφορούν στο ίδιο γεγονός, παρουσιάζουν πολύ μεγάλη ομοιότητα, η οποία καλό είναι να αξιοποιείται από το πρωτόκολλο δρομολόγησης ώστε να ελαχιστοποιείται η κατανάλωση ενέργειας και να μεγιστοποιείται ο χρόνος αντίδρασης δικτύου,

κάτι το οποίο δεν απαιτείται, τόσο επιτακτικά, σε κάποιο από τα άλλα είδη δικτύων.

Έτσι ένα πρωτόκολλο δρομολόγησης το οποίο προορίζεται για χρήση σε δίκτυα ασυρμάτων αισθητήρων θα πρέπει να ικανοποιεί όσο το δυνατόν περισσότερα από τα παρακάτω χαρακτηριστικά:

- Χαμηλή υπολογιστική πολυπλοκότητα: Το πρωτόκολλο για την εκτέλεση μιας λειτουργίας του θα πρέπει να εκτελεί όσο το δυνατό λιγότερα υπολογιστικά βήματα με τη μικρότερη κατανάλωση ενέργειας στον μικρότερο δυνατό χρόνο.

- Μικρές απαιτήσεις σε χώρο αποθήκευσης: Εξαιτίας του γεγονότος ότι στους κόμβους των δικτύων αισθητήρων, οι διαθέσιμοι χώροι μνήμης για την αποθήκευση δεδομένων συνήθως είναι μικροί, το πρωτόκολλο θα πρέπει να απαιτεί όσο το δυνατόν μικρότερο χώρο για την αποθήκευση των πληροφοριών που σχετίζονται με την δρομολόγηση.

- Μικρές απαιτήσεις σε ανταλλαγή μηνυμάτων: Επειδή το κόστος για την ανταλλαγή πακέτων δεδομένων ανάμεσα στους κόμβους είναι μεγάλο, το πρωτόκολλο θα πρέπει να απαιτεί για την λειτουργία του (υπολογισμό και εύρεση διαδρομών, διαχείριση, κ.τ.λ) την ανταλλαγή, κατά το δυνατόν, του μικρότερου αριθμού μηνυμάτων.

- Δικαιοσύνη στην χρήση των διαδρομών: Εξαιτίας του μοντέλου διακίνησης της πληροφορίας «many-to-one», παρουσιάζεται το φαινόμενο της ταχύτερης εκφόρτισης των κόμβων, οι οποίοι βρίσκονται πιο κοντά στα σημεία συγκέντρωσης της πληροφορίας (αρχηγούς ομάδων, σταθμό βάσης). Αυτό συμβαίνει εξαιτίας της μεγάλης χρήσης τους σαν αναμεταδότες των δεδομένων τρίτων κόμβων προς τους κόμβους που συγκεντρώνουν τα δεδομένα. Έτσι μια σημαντική ιδιότητα του πρωτοκόλλου είναι η χρησιμοποίηση διαφορετικών διαδρομών, δηλαδή η πιο δίκαιη χρησιμοποίηση των πόρων των κόμβων που βρίσκονται κοντά τους, ώστε να αποφεύγεται η γρήγορη εκφόρτισή τους.

- Αποφυγή κυκλικών διαδρομών: Μια διαδρομή θεωρείται κυκλική όταν περιλαμβάνει την διέλευση από το σημείο εκκίνησης παραπάνω από μια φορές.

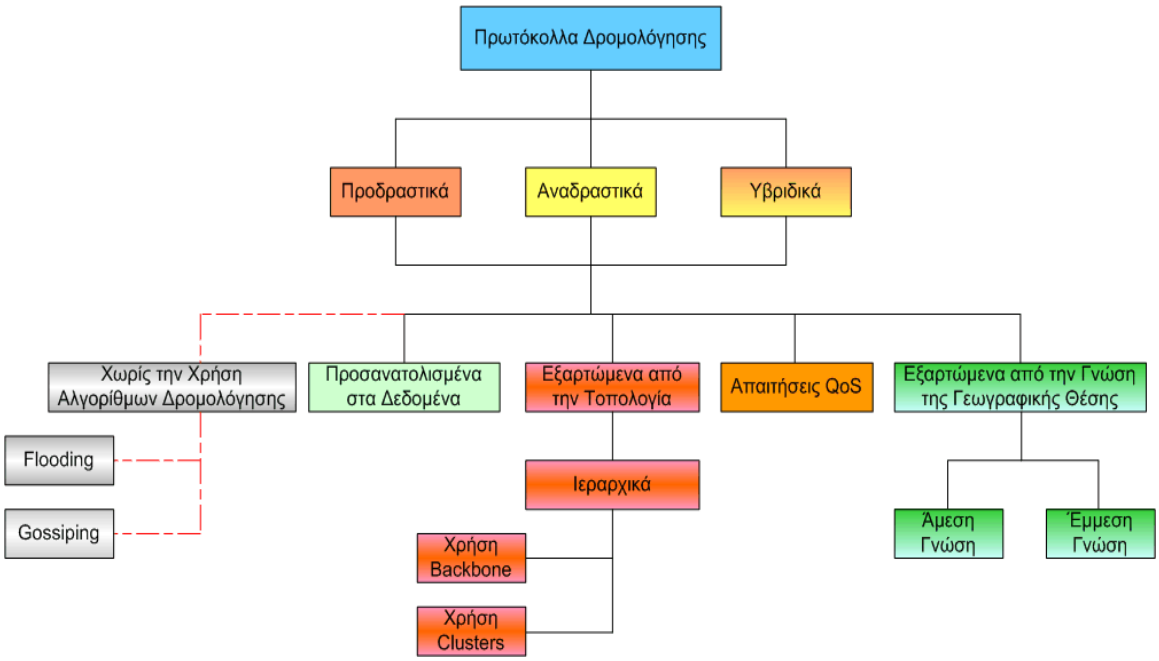
- Ταχεία απόκριση σε αλλαγές στην τοπολογία ή τη συνδεσιμότητα: Το πρωτόκολλο θα πρέπει να είναι ικανό να ανταποκρίνεται στις αλλαγές στην κατάσταση της τοπολογίας του δικτύου (προσθήκη – διαγραφή κόμβων, αλλαγή στην θέση λειτουργίας, αποφόρτιση και παύση λειτουργίας κάποιου κόμβου, κτλ)

- Υπολογισμός πολλαπλών διαδρομών: Ο υπολογισμός μιας κυρίως διαδρομής και αριθμού εναλλακτικών είναι επιθυμητός σε αρκετές από τις εφαρμογές των δικτύων αισθητήρων.

- Υποστήριξη απαιτήσεων παροχής υπηρεσιών με χαρακτηριστικά QoS (Quality of Service support): Σε ορισμένες απαιτητικές εφαρμογές των δικτύων αισθητήρων, π.χ ιατρικές εφαρμογές, στρατιωτικές, η δρομολόγηση δεδομένων πρέπει να χαρακτηρίζεται από την ικανοποίηση κάποιων ποιοτικών ή/και ποσοτικών χαρακτηριστικών, όπως ο ελάχιστος αριθμός των δεδομένων που πρέπει να παραδίδεται στη μονάδα του χρόνου (throughput), ο μέγιστος χρόνος από την στιγμή που ένα πακέτο είναι έτοιμο προς αποστολή στον κόμβο μέχρι την άφιξη του στον σταθμό βάσης (End-to-end data packet delay), η μέγιστη επιβάρυνση σε bit από το πρωτόκολλο δρομολόγησης (Routing overhead), ο μέγιστος χρόνος υπολογισμού μιας διαδρομής, η μέγιστη κατανάλωση ενέργειας κ.τ.λ.

3.2 Κατηγοριοποίηση Πρωτοκόλλων Δρομολόγησης

Τα πρωτόκολλα δρομολόγησης διαχωρίζονται σε κατηγορίες ανάλογα με κάποια χαρακτηριστικά τους, όπως ο τρόπος του υπολογισμού των διαδρομών δρομολόγησης, η τοπολογία του δικτύου τους, η γεωγραφική γνώση της θέσης των κόμβων, ο τρόπος ανάκτησης των δεδομένων τους, η παροχή ή όχι QoS υπηρεσιών, και τέλος από το εάν χρησιμοποιούν η όχι κάποιο αλγόριθμο δρομολόγησης (Εικόνα 9).



Εικόνα 9 - Κατηγοριοποίηση Πρωτοκόλλων Δρομολόγησης

3.3 Ενέργεια και Δρομολόγηση

Ένα από τα πιο σημαντικά μέτρα που χρησιμοποιούνται για τη σύγκριση των πρωτοκόλλων, αποτελεί αυτό της ενέργειας. Λόγω της φύσης των δικτύων αισθητήρων η εξοικονόμηση ενέργειας και κατά συνέπεια η επιμήκυνση του χρόνου ζωής του δικτύου αποτελεί πρωταρχικό στόχο.

Η επικοινωνία ανάμεσα στους κόμβους επιτυγχάνεται μέσω των πομποδεκτών τους με την εκπομπή και λήψη σημάτων προς και από αυτούς αντίστοιχα. Η κατανάλωση ενέργειας που εκτιμάται ότι καταναλώνεται για την μετάδοση ενός σήματος από έναν κόμβο i σε έναν κόμβο j δίνεται από τον τύπο

$$E_{tx} = E_{radio} + E(pkt\_size, d)$$

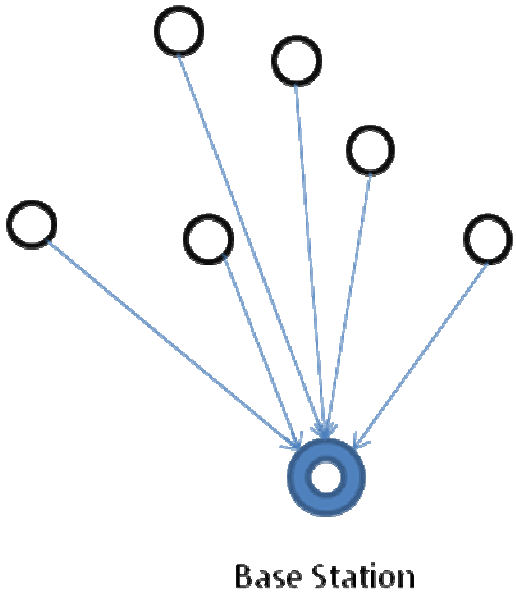
Όπου E_{radio} η ενέργεια που καταναλώνεται από το ασύρματο υποσύστημα και $E(pkt\_size, d)$ η ενέργεια που απαιτείται για τη μετάδοση δεδομένων μεγέθους pkt_size σε απόσταση d που χωρίζει τον εκπομπό από τον αποδέκτη, ενώ η ενέργεια που απαιτείται για τη λήψη τους είναι E_{rx}. Σε κάθε περίπτωση η καταναλισκόμενη ενέργεια είναι συνάρτηση του χρόνου για τον οποίο λειτουργεί ο πομπός ή ο δέκτης, που με τη σειρά του εξαρτάται από το μέγεθος του προς μετάδοση πακέτου και το εύρος ζώνης της μετάδοσης. Για τη μετάδοση του ραδιοσήματος σε απόσταση d και ανάλογα με το περιβάλλον μετάδοσης (φυσικά εμπόδια, υλικό εμποδίων κλπ) απαιτείται ισχύς ανάλογη του d^n, όπου n συντελεστής που συνήθως κυμαίνεται μεταξύ 2 και 5.

Λαμβάνοντας υπόψη τα παραπάνω για την επίτευξη εξοικονόμησης ενέργειας στο δίκτυο θα πρέπει να επιτευχθεί τόσο μείωση των αποστάσεων μετάδοσης όσο και μείωση του πλήθους των λειτουργιών αποστολής και λήψης για κάθε μήνυμα. Για όλες τις μετρήσεις που ακολουθούν έχει γίνει η υπόθεση ότι το κανάλι είναι συμμετρικό. Με άλλα λόγια η ενέργεια που απαιτείται για τη μετάδοση ενός μηνύματος από τον κόμβο Α στον κόμβο Β είναι ίδια με αυτήν που απαιτείται για τη μετάδοση αυτού του μηνύματος από τον κόμβο Β στον Α. .

3.4. Παραδοσιακά Πρωτόκολλα

3.4.1 One-Hop (Απευθείας μετάδοση)

Ίσως το πιο απλό πρωτόκολλο είναι αυτό της άμεσης επικοινωνίας (direct communication). Σύμφωνα με αυτό το πρωτόκολλο κάθε αισθητήρας στέλνει τα δεδομένα του απευθείας στον σταθμό βάσης (Εικόνα 10). Όσο ο τελευταίος τοποθετείται σε όλο και πιο μακρινή απόσταση από τους κόμβους του δικτύου, τόσο μεγαλύτερες απαιτήσεις σε ενέργεια υπάρχουν για τη μετάδοση δεδομένων (σύμφωνα με τα παραπάνω). Με άλλα λόγια, όσο αυξάνεται η απόσταση μεταξύ πομπού και δέκτη τόσο μεγαλύτερο ποσό ενέργειας καταναλώνεται (στον πομπό) για τη μετάδοση ενός μηνύματος. Συνέπεια αυτού αποτελεί η γρήγορη εξάντληση της μπαταρίας των κόμβων και συνεπώς η μείωση του κύκλου ζωής του συστήματος. Η παραλαβή μηνυμάτων σύμφωνα με το πρωτόκολλο άμεσης επικοινωνίας, η οποία απαιτεί κάποιο ποσό ενέργειας, περιορίζεται στον σταθμό βάσης. Κατά συνέπεια, στην περίπτωση που ο σταθμός βάσης βρίσκεται σε κοντινή απόσταση σε σχέση με του κόμβους το πρωτόκολλο της άμεσης επικοινωνίας κρίνεται αποδεκτή (αν όχι βέλτιστη) μέθοδος για την επικοινωνία αυτή. Χαρακτηριστικό του πρωτοκόλλου είναι ότι οι κόμβοι του δικτύου «πεθαίνουν» σε σχέση με την απόστασή τους από το σταθμό βάσης, από τους πιο απομακρυσμένους προς τους εγγύτερους.



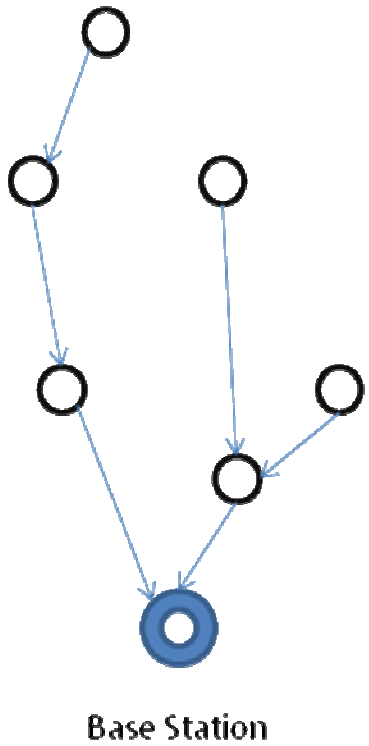
Εικόνα 10 - One-Hop

3.4.2 Multi-Hop

Στη multihop μετάδοση ένας κόμβος αντί να μεταδώσει απευθείας στο σταθμό βάσης, μεταδίδει σε γείτονά του που βρίσκεται σε κοντινότερη απόσταση από ότι η απόστασή του από το σταθμό βάσης και προς την κατεύθυνση αυτού (Εικόνα 11). Για να συμβεί βέβαια αυτό θα πρέπει κάθε κόμβος με κάποιο τρόπο να γνωρίζει τη

θέση των γειτόνων του. Όπως θα δούμε αργότερα και στις προσομοιώσεις που κάναμε το κριτήριο της απόστασης και μόνο δεν είναι αρκετό για να κάνει το multihoping ενεργειακά συμφέρουσα πρακτική αλλά πρέπει να λάβουμε υπόψη την ενεργειακή συμπεριφορά κάθε κόμβου που συμμετέχει (τόσο στην εκπομπή όσο και στη λήψη πακέτων).

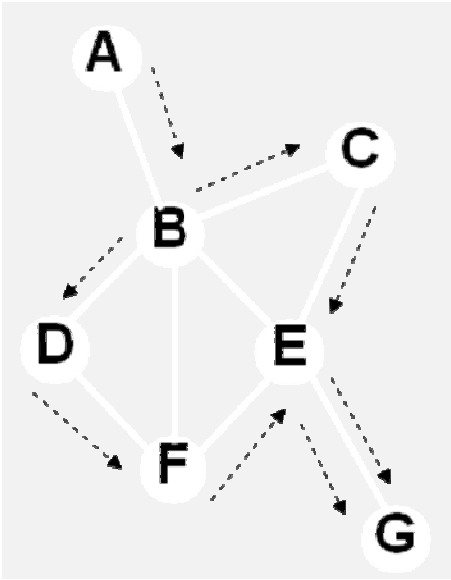
Χαρακτηριστικό αυτού του πρωτοκόλλου είναι ότι οι κόμβοι που βρίσκονται κοντύτερα στο σταθμό βάσης, εξαντλούνται ενεργειακά πολύ νωρίτερα από τους άλλους καθώς παίζουν το ρόλο του αναμεταδότη πολύ συχνά. Σε σχέση με το One-Hop μοντέλο οι κόμβοι του δικτύου «πεθαίνουν» με την αντίθετη κατεύθυνση, δηλαδή από το σταθμό βάσης προς την περιφέρεια.



Εικόνα 11 - Multi-Hop

3.4.3 Classic Flooding

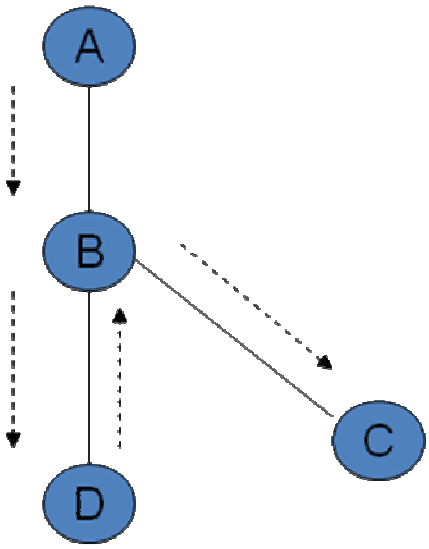
Με τον αλγόριθμο classic flooding [10], ένας κόμβος που επιθυμεί να διαδώσει κάποια πληροφορία στο δίκτυο αρχίζει στέλνοντας ένα αντίγραφο αυτής της πληροφορίας σε όλους τους γείτονες του. Όταν ένας κόμβος λάβει νέα δεδομένα φτιάχνει αντίγραφα και τα στέλνει σε όλους τους γειτονικούς του, εκτός από αυτόν που του έστειλε τα δεδομένα. Ο αλγόριθμος τελειώνει ή συγκλίνει όταν όλοι οι κόμβοι του δικτύου έχουν λάβει αντίγραφο των δεδομένων.



Εικόνα 12 - Flooding

3.4.4 Gossiping

Μια εναλλακτική προσέγγιση του κλασικού αλγορίθμου flooding αποτελεί ο αλγόριθμος gossiping [19] που χρησιμοποιεί την τυχαιότητα για να εξοικονομήσει ενέργεια. Αντί να προωθεί σε όλους του γείτονες τα δεδομένα, ένας gossiping κόμβος στέλνει την πληροφορία σε έναν τυχαίο γείτονα . Με αυτόν τον τρόπο όμως μπορεί ένας κόμβος να στείλει δεδομένα σε κάποιον κόμβο και αυτός να τα επιστρέψει αν τυχαία επιλέξει σαν αποδέκτη τον κόμβο που του τα έστειλε. Επίσης υπάρχει πιθανότητα κάποιοι κόμβοι να μην λάβουν ποτέ κάποια δεδομένα.

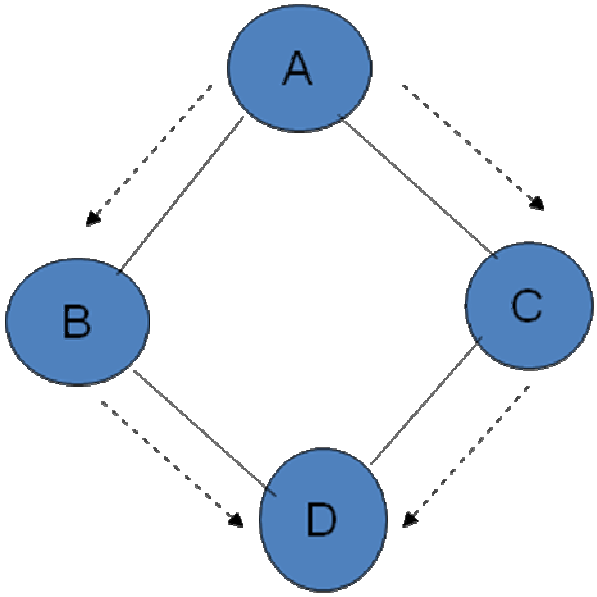


Εικόνα 13 - Gossiping

3.4.5 Προβλήματα & Αδυναμίες

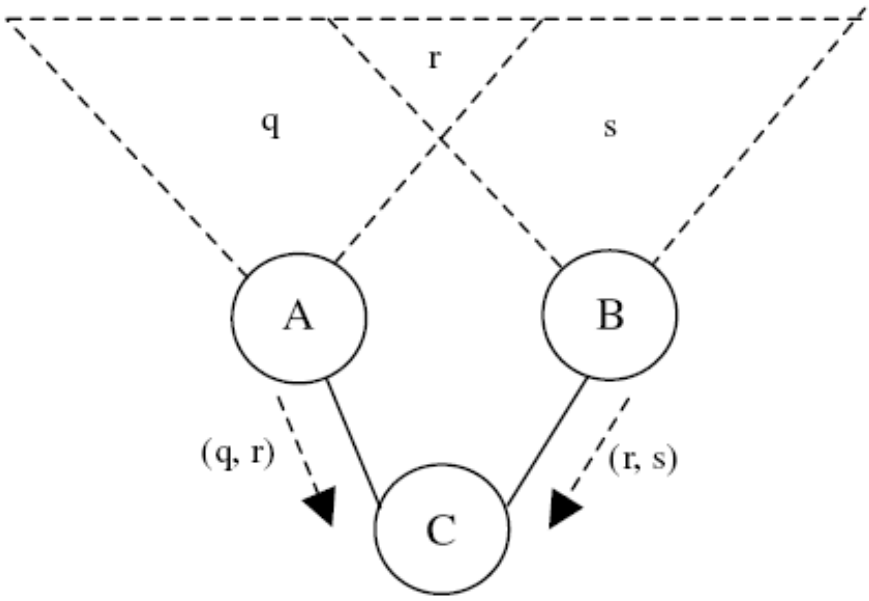
Παρόλο που τα παραπάνω πρωτόκολλα είναι ιδιαίτερα απλά και μεταδίδουν τα δεδομένα γρήγορα παρουσιάζουν σημαντικά προβλήματα και αδυναμίες με αποτέλεσμα να οδηγηθούμε στην ανάπτυξη νέων πρωτοκόλλων. Υπάρχουν τρία προβλήματα που κάνουν τις πιο πάνω προσεγγίσεις ανεπαρκείς.

- Κατάρρευση (implosion). Με το classic flooding, ένας κόμβος πάντα στέλνει δεδομένα στους γείτονες του ανεξάρτητα αν ο γειτονικός αυτός κόμβος έχει ήδη λάβει τα ίδια δεδομένα από κάποιον άλλο κόμβο. Αυτό οδηγεί στο πρόβλημα της κατάρρευσης που φαίνεται και στο παρακάτω σχήμα (Εικόνα 14). Στο συγκεκριμένο γράφημα ο κόμβος A στέλνει δεδομένα στους γειτονικούς του κόμβους B και C. Αυτοί οι κόμβοι αποθηκεύουν τα δεδομένα και στέλνουν ένα αντίγραφο στο δικό του γείτονα D. Το πρωτόκολλο λοιπόν, σπαταλά ενέργεια στέλνοντας δυο αντίγραφα στον κόμβο D.



Εικόνα 14 - Implosion

- Επικάλυψη (overlap). Συχνά οι κόμβοι αισθητήρων καλύπτουν επικαλυπτόμενες γεωγραφικές περιοχές με αποτέλεσμα οι κόμβοι να συγκεντρώνουν ίδια πληροφορία πολλές φορές. Το παρακάτω σχήμα διευκρινίζει τι γίνεται όταν δυο κόμβοι (A και B) αποκτούν επικαλυπτόμενα δεδομένα και στη συνέχεια τα προωθούν στον κοινό τους γειτονικό κόμβο (C). Ξανά ο αλγόριθμος σπαταλά ενέργεια και εύρος ζώνης στέλνοντας δυο αντίγραφα της πληροφορίας στον ίδιο κόμβο. Το πρόβλημα της επικάλυψης είναι δυσκολότερο να λυθεί από αυτό της κατάρρευσης. Αυτό οφείλεται στο γεγονός ότι η κατάρρευση είναι συνάρτηση μόνο της τοπολογίας του δικτύου, ενώ η επικάλυψη είναι συνάρτηση και της τοπολογίας και της αντιστοίχησης των παρατηρούμενων δεδομένων στους κόμβους αισθητήρων.



Εικόνα 15 - The Overlap problem

- Άγνοια διαθέσιμης ενέργειας (resource blindness). Στο πρωτόκολλο classic flooding οι κόμβοι δεν τροποποιούν τις δραστηριότητες τους ανάλογα με τη διαθέσιμη ενέργεια τη δεδομένη χρονική στιγμή. Ένα δίκτυο αισθητήρων θα πρέπει να μπορεί να προσαρμόζει την επικοινωνία και την επεξεργασία των δεδομένων ανάλογα με το επίπεδο της διαθέσιμης ενέργειας προκειμένου να γίνεται εξοικονόμηση.

3.5. FLAT Πρωτόκολλα

Τα πιο αντιπροσωπευτικά πρωτόκολλα της κατηγορίας αυτής αποτελούν η οικογένεια πρωτοκόλλων SPIN και το Directed Diffusion.

3.5.1 SPIN (Sensor Protocols for Information via Negotiation)

Μια οικογένεια πρωτοκόλλων που ονομάζονται SPIN (Sensor Protocols for Information via Negotiation) [18] ενσωματώνει δυο σημαντικές καινοτομίες που ξεπερνούν τις ατέλειες που αναφέρθηκαν πιο πάνω: τη διαπραγμάτευση και την προσαρμογή με βάση τη διαθέσιμη ενέργεια.

Για να λειτουργούν αποδοτικά και να εξοικονομούν ενέργεια οι εφαρμογές των αισθητήρων θα πρέπει να επικοινωνούν μεταξύ τους και να γνωστοποιούν τα δεδομένα που έχουν λάβει και τα δεδομένα που επιθυμούν να αποκτήσουν. Η ανταλλαγή δεδομένων μπορεί να είναι ενεργοβόρα διαδικασία, αλλά η ανταλλαγή δεδομένων σχετικά με τα δεδομένα (metadata) μπορεί να μην είναι. Έτσι οι SPIN κόμβοι διαπραγματεύονται μεταξύ τους πριν την μετάδοση των δεδομένων. Η διαπραγμάτευση εξασφαλίζει ότι μόνο χρήσιμη πληροφορία θα μεταδοθεί. Για να διαπραγματευθούν επιτυχώς οι κόμβοι θα πρέπει να είναι ικανοί να περιγράψουν ή να ονοματίσουν τα δεδομένα που κατέχουν. Τα δεδομένα-περιγραφείς που χρησιμοποιούνται στις διαπραγματεύσεις SPIN λέγονται μετα-δεδομένα (metadata).

Στο SPIN οι κόμβοι εξετάζουν την διαθέσιμη ενέργεια τους πριν μεταδώσουν δεδομένα. Κάθε κόμβος διατηρεί έναν διαχειριστή ενέργειας (resource manager) που καταγράφει την κατανάλωση ενέργειας. Οι εφαρμογές συμβουλεύονται τον διαχειριστή πριν μεταδώσουν ή επεξεργαστούν δεδομένα. Αυτό επιτρέπει στους κόμβους να αναστείλουν κάποιες δραστηριότητες αν η ενέργεια είναι χαμηλή π.χ. δεν προωθούν δεδομένα τρίτων. Επίσης επιτρέπουν στους κόμβους να υπολογίζουν το ενεργειακό κόστος μιας λειτουργίας και να λειτουργούν αναλόγως. Για παράδειγμα ένας SPIN κόμβος μπορεί να αποφασίσει να στείλει τα δεδομένα χωρίς να έχει προηγηθεί διαπραγμάτευση αν υπάρχει ενεργειακό συμφέρον.

Τα δυο πιο πάνω χαρακτηριστικά βοηθούν στην αντιμετώπιση των προβλημάτων της κατάρρευσης, της επικάλυψης και της άγνοιας της διαθέσιμης ενέργειας. Η διαπραγμάτευση περιορίζει την κατάρρευση με το να περιορίζει τα πλεονάζοντα δεδομένα. Η χρήση των μετα-δεδομένων περιορίζει την πιθανότητα εμφάνισης του προβλήματος της επικάλυψης, αφού επιτρέπει στους κόμβους να ονοματίσουν την πληροφορία που επιθυμούν να αποκτήσουν. Τέλος, ο έλεγχος τη ενέργειας κάνει τους κόμβους ιδιαίτερα προσεκτικούς στην επιλογή των δραστηριοτήτων που θα εκτελέσουν.

Μετα-δεδομένα (meta-data)

Οι αισθητήρες χρησιμοποιούν μετα-δεδομένα για να περιγράψουν συνοπτικά, αλλά ολοκληρωμένα τα δεδομένα που συγκεντρώνουν. Αν $\underline{x}$ είναι τα μετα-δεδομένα και $\underline{X}$ τα πραγματικά δεδομένα, τότε το μέγεθος των $\underline{x}$ σε bytes πρέπει να είναι μικρότερο από το μέγεθος των $\underline{X}$ ώστε το SPIN να είναι αποδοτικό. Αν δυο πληροφορίες είναι διαχωρίσιμες, τότε και τα μετα-δεδομένα που τις περιγράφουν πρέπει να είναι διαχωρίσιμα.

Τα πρωτόκολλα SPIN δεν προσδιορίζουν την μορφή (format) που πρέπει να έχουν τα μετα-δεδομένα κι αυτό γιατί κάθε εφαρμογή χρησιμοποιεί τη δική της μορφή.

SPIN μηνύματα

Οι κόμβοι χρησιμοποιούν τρία είδη μηνυμάτων για την επικοινωνία τους.

- **ADV** – γνωστοποίηση νέων δεδομένων (advertisement). Όταν ένας κόμβος διαθέτει δεδομένα προς διάθεση τα διαφημίζει στέλνοντας metadata.
- **REQ** – αίτηση για δεδομένα (request). Ένας κόμβος SPIN που θέλει να αποκτήσει κάποια συγκεκριμένα δεδομένα στέλνει μια αίτηση.
- **DATA** – δεδομένα. Πρόκειται για τα πραγματικά δεδομένα που έχουν για επικεφαλίδα metadata.

Το πρωτόκολλο αρχίζει όταν κάποιος κόμβος διαφημίζει νέα δεδομένα που επιθυμεί να μεταδώσει. Αυτό το κάνει στέλνοντας μήνυμα ADV στους κόμβους με τους οποίους γειτνιάζει. Όταν λάβει ένα μήνυμα ADV ένας κόμβος εξετάζει αν έχει ήδη λάβει ή αν έχει ζητήσει τα δεδομένα που διαφημίζονται. Αν όχι, ανταποκρίνεται στέλνοντας μήνυμα REQ για τα δεδομένα που δεν έχει και επιθυμεί να αποκτήσει. Το πρωτόκολλο ολοκληρώνεται με τον αρχικό κόμβο να στέλνει τα δεδομένα DATA απαντώντας στο μήνυμα REQ.

Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα εφαρμογής του πρωτοκόλλου.

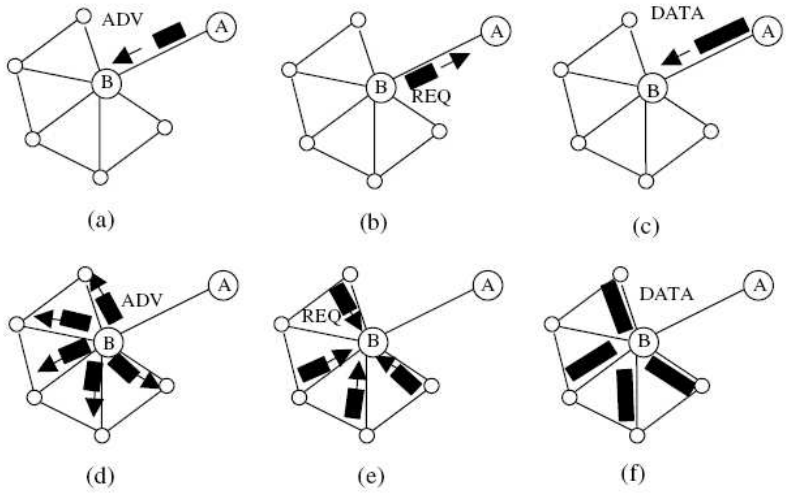


Fig. 3. SPIN protocol. Node A starts by advertising its data to node B (a). Node B responds by sending a request to node A (b). After receiving the requested data (c), node B then sends out advertisements to its neighbors (d), who in turn send requests back to B (e–f).

Εικόνα 16 - SPIN-PP protocol (three-stage handshake protocol)

SPIN Πρωτόκολλα

Υπάρχουν τέσσερα πρωτόκολλα που ακολουθούν την φιλοσοφία SPIN, οπότε αναφερόμαστε σε μια οικογένεια πρωτοκόλλων. Δυο από αυτά τα πρωτόκολλα το SPIN-PP και το SPIN-BC αντιμετωπίζουν το πρόβλημα της μετάδοσης δεδομένων κάτω από ιδανικές συνθήκες, όπου η ενέργεια είναι άφθονη και τα πακέτα δεν χάνονται. Το SPIN-PP χρησιμοποιεί τη μετάδοση σημείο-προς-σημείο (point-to-point), ενώ το SPIN-BC χρησιμοποιεί broadcasting. Σε μια broadcast μετάδοση οι κόμβοι στο δίκτυο επικοινωνούν χρησιμοποιώντας ένα κοινό κανάλι. Αν ένας κόμβος επιθυμεί να στείλει ένα μήνυμα και αφουγκράζεται ότι το κανάλι χρησιμοποιείται θα πρέπει να περιμένει μέχρι αυτό να ελευθερωθεί. Το μειονέκτημα τέτοιων δικτύων είναι ότι όταν ένας κόμβος στέλνει ένα μήνυμα, όλοι οι κόμβοι που βρίσκονται στην εμβέλεια του θα πληρώσουν το τίμημα της μετάδοσης και σε χρόνο και σε ενέργεια. Παρόλ' αυτά η επικοινωνία one-to-many είναι κατά 1/n οικονομικότερη από την point-to-point επικοινωνία, όπου n ο αριθμός των γειτόνων κάθε κόμβου. Τα άλλα δυο πρωτόκολλα το SPIN-EC και το SPIN-RL είναι τροποποιημένες εκδόσεις των ανωτέρω πρωτοκόλλων. Το SPIN-EC είναι μια έκδοση του SPIN-PP με κύριο χαρακτηριστικό τη μείωση των μηνυμάτων που ανταλλάσσονται μεταξύ των κόμβων όταν η ενέργεια είναι χαμηλή. Όταν ένας κόμβος παρατηρήσει ότι η ενέργεια του πλησιάζει ένα κατώφλι χαμηλής ενέργειας μειώνει τη συμμετοχή του στο πρωτόκολλο. Γενικά ένας κόμβος συμμετέχει σε μια λειτουργία του δικτύου αν πιστεύει ότι θα ολοκληρώσει όλες τις διαδικασίες του πρωτοκόλλου χωρίς να περάσει το κατώφλι χαμηλής ενέργειας. Αυτή η προσέγγιση δεν εμποδίζει έναν κόμβο να δέχεται μηνύματα ADV και REQ όταν πέσει κάτω από το κατώφλι η ενέργεια του, αλλά εμποδίζει να χειρίζεται δεδομένα. Το SPIN-RL είναι μια αξιόπιστη έκδοση του SPIN-BC που ανακτά την ισορροπία του δικτύου όταν χάνονται πακέτα με το να επανεκπέμπει επιλεκτικά κάποια μηνύματα.

Προσομοιώσεις

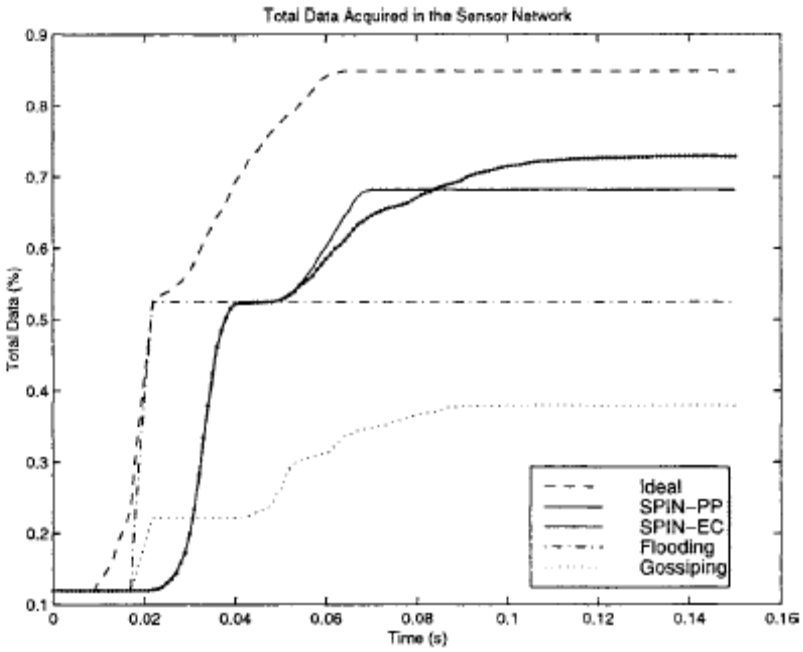
Προσομοιώσεις έγιναν σε δίκτυο με τα πιο κάτω χαρακτηριστικά (εικόνα 17):

Characteristics of the 25-node wireless test network.

Parameter	Value
Nodes	25
Edges	59
Average degree	4.7 neighbors
Diameter	8 hops
Average shortest path	3.2 hops
Antenna reach	10 m
Radio propagation delay	3×10^8 m/s
Processing delay	5–10 ms
Radio speed	1 Mbps
Transmit cost	600 mW
Receive cost	200 mW
Data size	500 bytes
Meta-data size	16 bytes
Network losses	None
Queuing delays	None

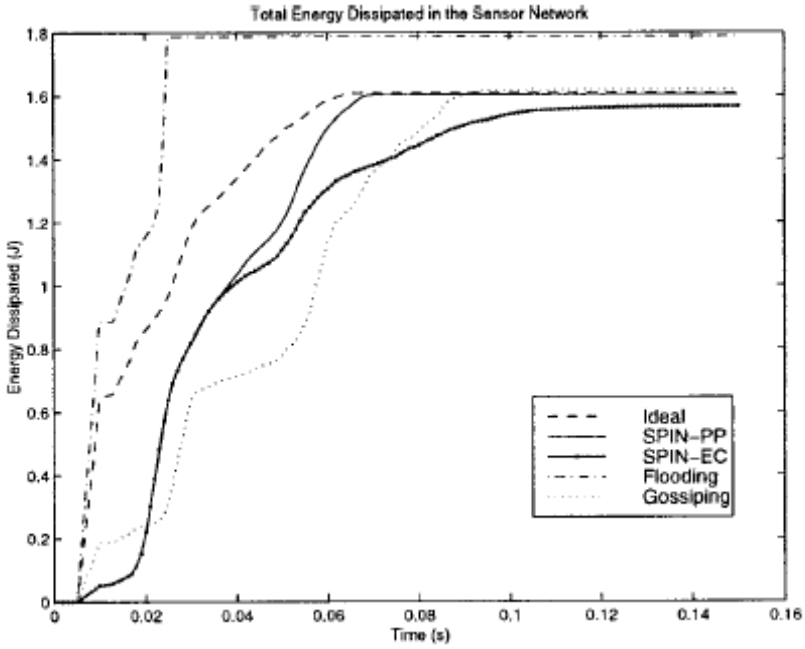
Εικόνα 17 - Χαρακτηριστικά Δικτύου 25 κόμβων

Όταν η ενέργεια του συστήματος περιορίζεται στα 1,6 J η γραφική παράσταση των συνολικών δεδομένων που μεταδίδονται σε συνάρτηση με το χρόνο για τα πρωτόκολλα Ideal, SPIN-PP, SPIN-EC, Flooding και Gossiping είναι η παρακάτω.



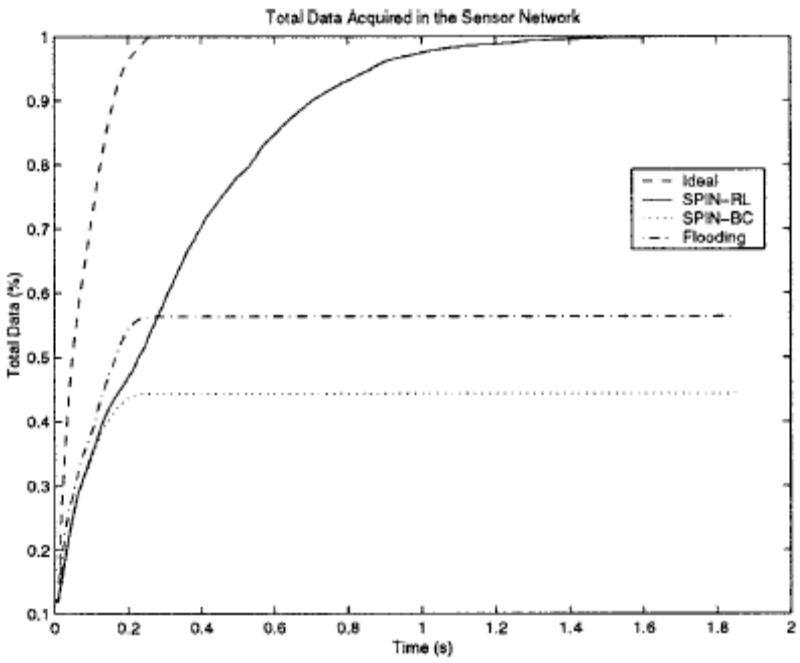
Εικόνα 18 - Ποσοστό συνολικών δεδομένων που αποκτήθηκαν από το σύστημα

Η ενέργεια που καταναλώνουν τα πρωτόκολλα στο χρόνο φαίνεται πιο κάτω:

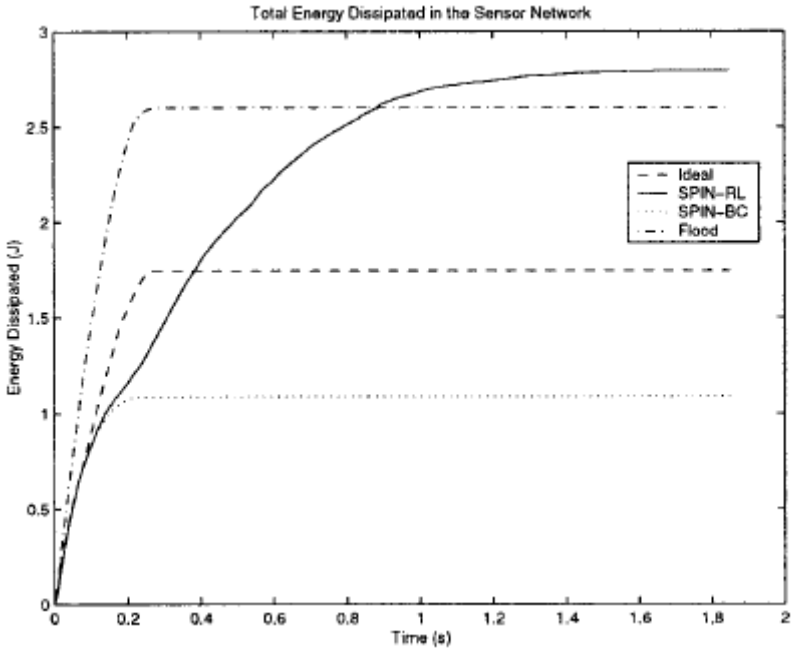


Εικόνα 19 - Κατανάλωση ενέργειας στο χρόνο

Όταν γίνεται χρήση SPIN-RL και SPIN-BC για broadcast σε δίκτυο με απώλειες πακέτων τα αντίστοιχα αποτελέσματα φαίνονται παρακάτω:



Εικόνα 20 - Ποσοστό συνολικών δεδομένων που αποκτήθηκαν από το σύστημα σε lossy broadcast δίκτυο



Εικόνα 21 - Κατανάλωση ενέργειας στο χρόνο σε lossy broadcast δίκτυο

Από τις προσομοιώσεις φαίνεται ότι το SPIN-PP μπορεί να μεταδώσει το 68% των συνολικών δεδομένων, το flooding 53% και το gossiping το 38% των συνολικών

δεδομένων. Το SPIN-EC μπορεί να διανείμει το 73% των συνολικών δεδομένων συγκρινόμενο με το ideal πρωτόκολλο που μετακινεί το 85% των δεδομένων.

Το SPIN-BC δεν έχει καλύτερα αποτελέσματα από το flooding σε δίκτυα όπου έχουμε απώλειες πακέτων. Αυτό οφείλεται στο γεγονός ότι οι SPIN-BC κόμβοι πρέπει να στείλουν και να λάβουν τρία μηνύματα για να μεταδώσουν την πληροφορία στον επόμενο κόμβο, ενώ το flooding μεταδίδει μόνο ένα. Επομένως το πρωτόκολλο SPIN-BC είναι πιο ευάλωτο στις απώλειες από το flooding.

Συμπεράσματα

Η περιγραφή των δεδομένων με την χρήση μετα-δεδομένων και η διαπραγμάτευση των μεταδόσεων των δεδομένων αντιμετωπίζει αποτελεσματικά τα προβλήματα της κατάρρευσης και της επικάλυψης. Επίσης τα πρωτόκολλα SPIN είναι απλά και διαδίδουν τα δεδομένα αποτελεσματικά, ενώ διατηρούν μόνο τοπική πληροφορία για τους κοντινότερους γειτονικούς κόμβους. Αυτά τα πρωτόκολλα είναι τα πλέον κατάλληλα για περιβάλλοντα όπου οι αισθητήρες κινούνται, επειδή βασίζουν τις αποφάσεις προώθησης των δεδομένων στην τοπική πληροφορία της γειτονιάς τους. Το SPIN-PP χρησιμοποιεί μόνο το 25% της ενέργειας που χρησιμοποιεί ένα κλασικό flooding πρωτόκολλο. Επίσης το SPIN-EC μπορεί να μεταδώσει 60% περισσότερα δεδομένα ανά μονάδα ενέργειας. Σε όλες τις προσομοιώσεις τα πρωτόκολλα SPIN-PP και SPIN-EC παρουσίασαν καλύτερα αποτελέσματα από το gossiping, ενώ σε ορισμένες περιπτώσεις αυτά είναι πολύ κοντά στα αποτελέσματα που εμφανίζει ένα ιδανικό πρωτόκολλο. Τα πρωτόκολλα SPIN-BC και SPIN-RL υποστηρίζουν την επικοινωνία one-to-many, μεταδίδοντας μάλιστα τα δεδομένα ταχύτερα από το flooding πρωτόκολλο χρησιμοποιώντας λιγότερη ενέργεια. Το πρωτόκολλο SPIN-RL όχι μόνο λειτουργεί και κατά την παρουσία απωλειών στο δίκτυο, αλλά μπορεί να μεταδώσει τα διπλάσια δεδομένα από το flooding. Ένα σημαντικό μειονέκτημα των πρωτοκόλλων SPIN είναι η αδυναμία παροχής εγγυήσεων για την αποστολή των δεδομένων σε κάθε κόμβο του δικτύου. Έτσι αν ένας ενδιάμεσος κόμβος σε ένα μονοπάτι δεν επιθυμεί να λάβει κάποια δεδομένα τότε αυτά δεν φτάνουν στο τέλος του μονοπατιού.

3.5.2 Directed Diffusion

Στο Directed Diffusion [14], [17] ένας κόμβος ζητάει δεδομένα στέλνοντας Interests (ερωτήσεις). Η διάδοση των ερωτήσεων αφήνει ίχνη ώστε τα δεδομένα που ταιριάζουν με το ενδιαφέρον/ερώτηση να μπορούν να φθάσουν στον κόμβο που τα επιθυμεί. Οι ενδιάμεσοι κόμβοι μπορούν να επεξεργαστούν (π.χ. αποθηκεύουν, συναθροίζουν, μεταδίδουν) ή να προωθούν δεδομένα και ερωτήσεις απευθείας για

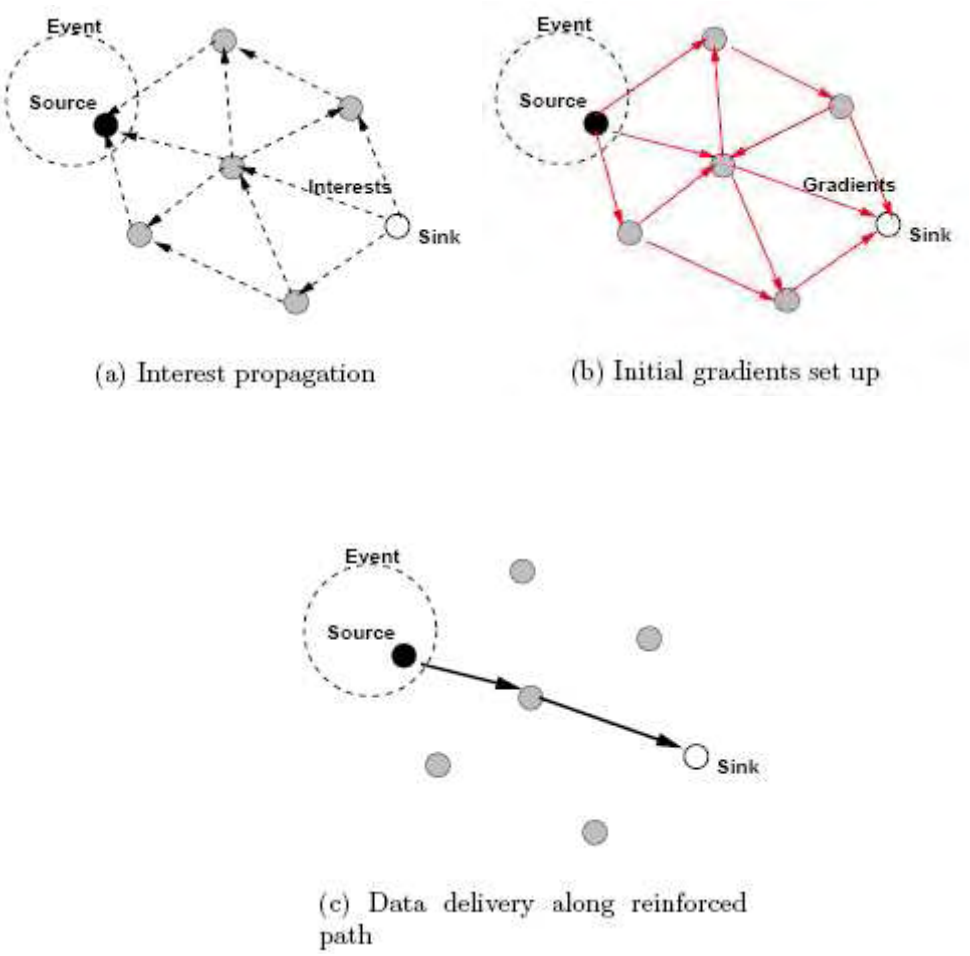
να εξοικονομούν ενέργεια και εύρος ζώνης. Το Directed Diffusion είναι τελείως διαφορετικό από την IP based επικοινωνία. Η IP επικοινωνία βασίζεται στην end-to-end μετάδοση πακέτων τα οποία μετακινούνται στο δίκτυο με βάση το ID του κόμβου προορισμού. Αντίθετα το Directed Diffusion είναι data-centric, δηλαδή η επικοινωνία βασίζεται στα δεδομένα και όχι στις διευθύνσεις των κόμβων.

Το πρωτόκολλο Directed Diffusion αποτελείται από διάφορα συστατικά: interests (ερωτήσεις), data messages (δεδομένα), gradients (διανύσματα) και reinforcement (επιλογή μονοπατιού). Interest καλούμε μια ερώτηση που προσδιορίζει τι θέλει ο χρήστης. Ουσιαστικά είναι ένα μήνυμα που περιέχει την περιγραφή των δεδομένων που ενδιαφέρουν τον χρήστη. Δεδομένα σε ένα δίκτυο αισθητήρων είναι η παρατηρήσιμη ή η επεξεργασμένη πληροφορία ενός φαινομένου που «απαντά» στην ερώτηση του χρήστη. Τα δεδομένα αυτά μπορεί να είναι ένα γεγονός δηλαδή μια περιγραφή του παρατηρούμενου φαινομένου. Στο Directed Diffusion τα δεδομένα περιγράφονται χρησιμοποιώντας κάποια χαρακτηριστικά. Οι ερωτήσεις διοχετεύονται στο δίκτυο με σκοπό την ανάσυρση των δεδομένων από τον χρήστη. Η διάδοση των ερωτημάτων εγκαθιστά διανύσματα (gradients) μέσα στο δίκτυο με κατεύθυνση αντίθετη με αυτή των ερωτημάτων. Σε κάθε κόμβο δηλαδή δημιουργείται ένα διάνυσμα προς τον κόμβο που του έστειλε την ερώτηση με αποτέλεσμα τα γεγονότα να διαδίδονται προς την κατεύθυνση που δείχνει το διάνυσμα. Δημιουργούνται έτσι πολλαπλά μονοπάτια διανυσμάτων από τα οποία επιλέγεται ένα ή περισσότερα από αυτά.

Στο Directed Diffusion οι εργασίες (tasks) περιγράφονται χρησιμοποιώντας κάποια χαρακτηριστικά. Τα χαρακτηριστικά που επιλέγονται για την περιγραφή μιας εργασίας εξαρτώνται από την εφαρμογή και η σωστή επιλογή τους συμβάλει στην αποδοτικότητα του πρωτοκόλλου.

Για κάθε εργασία, ένας κόμβος sink στέλνει περιοδικά μια ερώτηση σε όλους τους γειτονικούς κόμβους. Η αρχική αυτή ερώτηση περιέχει τα επιθυμητά χαρακτηριστικά. Η περιοδική αποστολή της ερώτησης είναι απαραίτητη, επειδή οι ερωτήσεις δεν μεταδίδονται αξιόπιστα. Έτσι ο κόμβος sink ξαναστέλνει την ερώτηση με αυξανόμενη τιμή στο πεδίο time stamp. Κάθε κόμβος αποθηκεύει προσωρινά τις ερωτήσεις που λαμβάνει, ενώ αυτές (οι ερωτήσεις) δεν περιέχουν πληροφορία για τον κόμβο sink, αλλά μόνο για τον αμέσως προηγούμενο κόμβο. Όταν ένας κόμβος λάβει μια ερώτηση ελέγχει αν υπάρχει ίδια ερώτηση στην cache. Αν όχι δημιουργεί μια καταχώρηση στην cache με πεδία αυτά που περιγράφονται στα χαρακτηριστικά του ερωτήματος. Αφού λάβει την ερώτηση ένας κόμβος μπορεί να αποφασίσει να στείλει την ερώτηση σε κάποιον ή σε όλους τους γειτονικούς κόμβους. Αν δεν υπάρχει πληροφορία για το ποιοι κόμβοι μπορούν να ικανοποιήσουν/απαντήσουν την ερώτηση , στέλνεται σε όλους τους γείτονες.

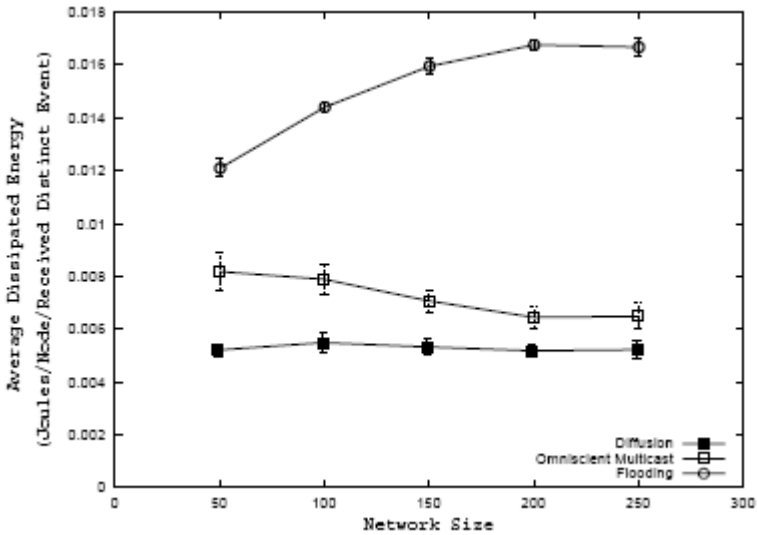
Γενικά ένα διάνυσμα αποτελείται από μια τιμή και μια κατεύθυνση στην οποία θα σταλούν τα δεδομένα. Συνήθως η τιμή αυτή είναι ο ρυθμός μετάδοσης, αλλά το Directed Diffusion δίνει την ελευθερία στον σχεδιαστή να επιλέξει την επιθυμητή σημασιολογία για την συγκεκριμένη τιμή. Για να επιλέξει κάποιον γείτονα ο κόμβος που έθεσε το ερώτημα (sink node) ξαναστέλνει το αρχικό αυτό ερώτημα με μικρότερη τιμή της παραμέτρου interval. Ο κόμβος αντιλαμβάνεται ότι έχει ένα διάνυσμα προς τον κόμβο sink με ρυθμό μετάδοσης χαμηλότερο αυτού που προσδιορίζεται στην ερώτηση, οπότε καταλαβαίνει ότι ανήκει στο μονοπάτι που επιλέγεται/ενισχύεται (reinforcement) για την αποστολή δεδομένων προς τον κόμβο sink. Με τη σειρά του λοιπόν πρέπει να επιλέξει/ενισχύσει ένα δικό του γειτονικό κόμβο με τον ίδιο τρόπο. Σχηματίζεται λοιπόν με αυτόν τον τρόπο ένα ενισχυμένο μονοπάτι από την πηγή των δεδομένων που ενδιαφέρουν τον χρήστη προς τον χρήστη όπου επιτυγχάνεται μεγαλύτερος ρυθμός μετάδοσης. Στα παρακάτω σχήματα φαίνεται η παραπάνω διαδικασία.



Εικόνα 22 - Directed Diffusion

Προσομοιώσεις

Συγκρίνοντας την ενέργεια που καταναλώνεται από το πρωτόκολλο Directed Diffusion με αυτή των πρωτοκόλλων Flooding και Omniscient Multicast παρατηρούμε ότι η κατανάλωση του Directed Diffusion είναι κατά δυο ή τρεις φορές μικρότερη ανάλογα με το μέγεθος του δικτύου.



Εικόνα 23 - Καταλισκούμενη ενέργεια

Στα θετικά του πρωτοκόλλου συγκαταλέγονται η κατασκευή μονοπατιών κατά απαίτηση καθώς και η δυνατότητα κάθε κόμβου να συναθροίζει και να αποθηκεύει δεδομένα με αποτέλεσμα εξοικονόμηση ενέργειας και μικρή καθυστέρηση (low latancy). Το κυριότερο μειονέκτημα του πρωτοκόλλου είναι το επιπλέον overhead που επιφέρει για την αντιστοίχηση ερώτησης και δεδομένων ενώ δεν είναι καλή επιλογή για συνεχόμενη αποστολή δεδομένων, αφού είναι query driven.

3.6. Ιεραρχικά Πρωτόκολλα

Ένα άλλο πρωτόκολλο δρομολόγησης ασύρματων δικτύων αποτελεί η συσταδοποίηση. Σύμφωνα με αυτό οι κόμβοι του δικτύου ομαδοποιούνται σε συστάδες. Ένας κόμβος σε κάθε συστάδα ονομάζεται κόμβος επικεφαλής και έχει αναλάβει την μετάδοση δεδομένων στον σταθμό βάσης. Με τον τρόπο αυτό επιτυγχάνεται μείωση της απόστασης που επικοινωνούν οι κόμβοι (αποστολή

δεδομένων σε επικεφαλή συστάδας). Για το λόγο αυτό, η συσταδοποίηση αποτελεί ενεργειακά αποδοτικό πρωτόκολλο επικοινωνίας [22]. Ωστόσο, ο επικεφαλής κάθε συστάδας πρέπει να χαρακτηρίζεται από υψηλό ποσοστό ενέργειας, διαφορετικά υπάρχει κίνδυνος γρήγορης ενεργειακής εξάντλησης αυτού.

Τα πιο διαδεδομένα ιεραρχικά πρωτόκολλα είναι τα: LEACH, PEGASIS, TEEN και APTEEN, που εξετάζονται πιο αναλυτικά στη συνέχεια.

3.6.1 LEACH

Το LEACH (Low-Energy Adaptive Clustering Hierarchy) [5] αποτελεί ένα αυτό-οργανώσιμο πρωτόκολλο συσταδοποίησης το οποίο επιτυγχάνει άρτια κατανομή της ενέργειας στους κόμβους του δικτύου. Οι κόμβοι οργανώνονται από μόνοι τους (αυτό-οργανώσιμοι) σε τοπικές συστάδες, ορίζοντας έναν κόμβο επικεφαλή (cluster-head) της συστάδας. Ο τελευταίος επιφορτίζεται με την μεγαλύτερη ενεργειακά εργασία, καθώς δέχεται τα δεδομένα των κόμβων-μελών της συστάδας του, εφαρμόζει συνάθροιση (aggregation) σε αυτά και σε τελικό στάδιο τα αποστέλλει στον απομακρυσμένο σταθμό βάσης. Με αυτόν τον τρόπο ο χρήστης λαμβάνει τα επιθυμητά δεδομένα από τον σταθμό βάσης. Σε συμβατικούς αλγόριθμους δρομολόγησης ορίζονται εκ των προτέρων οι επικεφαλείς κόμβοι του δικτύου. Έτσι, καθ΄ όλη τη διάρκεια ζωής του δικτύου οι κόμβοι επικεφαλής λόγο της φύσης του ρόλου τους αποτελούν κόμβους οι οποίοι χάνουν ενέργεια με ρυθμό μεγαλύτερο από ότι οι κοινοί κόμβοι. Το γεγονός αυτό έχει σαν αποτέλεσμα οι κόμβοι επικεφαλής να εξαντλούνται ενεργειακά γρήγορα, με απώτερη επίπτωση τη μείωση του κύκλου ζωής του δικτύου. Για το λόγο αυτό, το πρωτόκολλο LEACH εναλλάσσει με τυχαίο τρόπο τους κόμβους επικεφαλείς – κόμβοι υψηλών ενεργειακών θέσεων - επιτυγχάνοντας έτσι τη μείωση του ρυθμού απώλειας ενέργειας των κόμβων και συνεπώς την αύξηση του χρόνου ζωής του συνόλου του δικτύου. Όπως προαναφέρθηκε, ο επικεφαλής κόμβος κάθε συστάδας αφού συγκεντρώσει τα δεδομένα από τους κόμβους της ομάδας του, εκτελεί σε αυτά συγκερασμό και στη συνέχεια τα αποστέλλει στον σταθμό βάσης. Με τον συγκερασμό των δεδομένων επιτυγχάνεται μείωση του όγκου των δεδομένων προς μετάδοση, με αποτέλεσμα λιγότερη απώλεια ενέργειας για τους κόμβους επικεφαλής. Το γεγονός αυτό συνεισφέρει ακόμα πιο πολύ στη βελτίωση του χρόνου ζωής ενός ασύρματου δικτύου αισθητήρων.

Όπως προαναφέρθηκε, στους κόμβους επικεφαλής καταναλώνεται μεγάλο ποσοστό ενέργειας. Για την ομοιόμορφη κατανομή της κατανάλωσης ενέργειας σε όλο το δίκτυο, το LEACH δεν επιτρέπει οι κόμβοι επικεφαλείς να είναι μόνιμοι. Με λίγα λόγια, ανά τακτά διαφορετικά χρονικά διαστήματα αυτό-ορίζεται διαφορετικό σύνολο κόμβων του δικτύου ως επικεφαλής. Για παράδειγμα, αν το σύνολο των

κόμβων επικεφαλής του δικτύου τη χρονική στιγμή t1 είναι C, τη χρονική στιγμή t1+d το σύνολο των κόμβων επικεφαλείς θα είναι C'. Η απόφαση για την επιλογή των κόμβων επικεφαλείς των συστάδων εξαρτάται από το ποσό της ενέργειας που έχει απομείνει σε κάθε κόμβο. Με αυτόν τον τρόπο, κόμβοι με μεγαλύτερη εναπομείναντα ενέργεια θα αντεπεξέλθουν καλύτερα στις ενεργειακά αυξημένες απαιτήσεις του ρόλου τους ως επικεφαλείς για το δίκτυο. Κάθε κόμβος αποφασίζει για τον αν θα γίνει επικεφαλής ανεξάρτητα από τους άλλους κόμβους του δικτύου. Έτσι δεν απαιτείται επιπλέον διαπραγμάτευση για τον καθορισμό των επικεφαλείς αυτού.

Το σύστημα μπορεί να καθορίσει εκ των προτέρων, το βέλτιστο αριθμό συστάδων που απαιτούνται. Το τελευταίο εξαρτάται από πολλαπλούς παράγοντες, όπως η τοπολογία του δικτύου και τα σχετικά υπολογιστικά και επικοινωνιακά κόστη. Στην περίπτωση που είτε δεν υπάρχουν επικεφαλείς ή όλοι οι κόμβοι είναι επικεφαλείς, το δίκτυο έχει την ίδια συμπεριφορά με αυτό της άμεσης επικοινωνίας. Από πειράματα έχει βρεθεί ότι υπάρχει ένα βέλτιστο ποσοστό κόμβων N που μπορούν να οριστούν ως επικεφαλείς σε ένα δίκτυο. Αν το πλήθος των επικεφαλείς κόμβων είναι μικρότερο του N, τότε μερικοί κόμβοι βρίσκονται μακριά από τους επικεφαλείς με αποτέλεσμα να απαιτείται μεγαλύτερη κατανάλωση ενέργειας από αυτούς τους κόμβους για την αποστολή των δεδομένων τους στους επικεφαλείς της συστάδας τους. Με αυτόν τον τρόπο προκαλείται συνολική αύξηση κατανάλωσης ενέργειας στο δίκτυο. Ενώ στην περίπτωση που οι κόμβοι που έχουν επιλεγεί ως επικεφαλείς είναι περισσότεροι του N, τότε η απόσταση μεταξύ των κόμβων και των επικεφαλείς σε κάθε συστάδα δεν μειώνεται σημαντικά ενώ υπάρχουν περισσότεροι κόμβοι επικεφαλείς που πρέπει να μεταδώσουν τα δεδομένα στο σταθμό βάσης που βρίσκεται σε μεγάλη απόσταση από το δίκτυο και η συμπίεση που γίνεται στα δεδομένα τοπικά είναι προφανώς μικρότερη. Για τις παραμέτρους του πειράματος και την τοπολογία υπολογίσθηκε ότι το πλήθος του ιδανικού πλήθους κόμβων επικεφαλείς είναι N=5% επί του συνολικού αριθμού κόμβων του δικτύου.

Αλγόριθμος LEACH

Η λειτουργία του LEACH χωρίζεται σε γύρους (rounds), όπου κάθε ένας από αυτούς χωρίζεται σε τέσσερις φάσεις: φάση ενημέρωσης (Advertisement Phase), φάση διαμόρφωσης συστάδας (Cluster Set-Up Phase), φάση δημιουργίας προγράμματος (Schedule Creation) και φάση μετάδοσης δεδομένων (Data Transmission). Στη συνέχεια γίνεται μια σύντομη αναφορά σε αυτές τις φάσεις.

A. Φάση Ενημέρωσης

Αρχικά, κατά τη δημιουργία συστάδων, κάθε κόμβος αποφασίζει για το αν θα γίνει επικεφαλής για τον τρέχον γύρο. Αυτή η απόφαση βασίζεται στο προτεινόμενο

ποσοστό από επικεφαλείς για το δίκτυο (προκαθορισμένο εκ των προτέρων) και του αριθμού των φορών που κάποιος κόμβος έχει ήδη βρεθεί στη θέση του επικεφαλή. Η απόφαση λαμβάνεται από τον κόμβο n επιλέγοντας έναν αριθμό τυχαία έναν αριθμό από το 0 εως το 1. Αν ο αριθμός είναι κάτω από ένα κατώφλι T(n) τότε ο κόμβος γίνεται επικεφαλής. Το κατώφλι ορίζεται ως εξής:

$$\mathrm{T}(n)=\begin{cases}\dfrac{P}{1-P*\left(r \bmod \dfrac{1}{P}\right)}, \alpha\nu \quad n\varepsilon G \\ 0, \quad \delta\iota\alpha\varphi o\rho\varepsilon\tau\iota\kappa\acute{\alpha}\end{cases}$$

όπου P είναι το επιθυμητό ποσοστό κόμβων επικεφαλείς(P=0.05), r είναι ο τρέχων γύρος και G το σύνολο των κόμβων που δεν έχουν γίνει επικεφαλείς τους τελευταίους $\frac{1}{P}$ γύρους. Με αυτόν τον τρόπο κάθε κόμβος γίνεται επικεφαλείς κάποια στιγμή στους $\frac{1}{P}$ γύρους. Κατά τη διάρκεια του πρώτου γύρου(r=0) κάθε κόμβος έχει πιθανότητα P για να γίνει επικεφαλής. Οι κόμβοι που ορίζονται επικεφαλείς τον πρώτο γύρο, δεν μπορούν να επιλεγούν ξανά ως επικεφαλείς για τους επόμενους $\frac{1}{P}$ γύρους. Με αυτόν τον τρόπο, αυξάνεται η πιθανότητα των υπόλοιπων κόμβων να γίνουν επικεφαλείς καθώς λιγότεροι κόμβοι είναι υποψήφιοι για να γίνουν επικεφαλείς. Μετά από $\frac{1}{P}-1$ γύρους, το κατώφλι T=1 για κάθε κόμβο που δεν έχει γίνει επικεφαλής. Ενώ μετά από $\frac{1}{P}$ γύρους όλοι οι κόμβοι είναι ξανά το ίδιο πιθανοί να γίνουν επικεφαλείς.

Κάθε κόμβος που έχει αυτό-ορισθεί επικεφαλής για τον τρέχον γύρο μεταδίδει ένα μήνυμα ενημέρωσης στους υπόλοιπους κόμβους για να τους ειδοποιήσει σχετικά με την απόφασή του να είναι επικεφαλής. Κατά τη διάρκεια της φάσης αυτής, οι επικεφαλείς χρησιμοποιώντας το πρωτόκολλο CSMA MAC μεταδίδουν το παραπάνω μήνυμα ενημέρωσης καταναλώνοντας όλοι την ίδια ενέργεια. Ενώ, οι πομποί των απλών κόμβων είναι ανοικτοί για τη λήψη των μηνυμάτων ενημέρωσης.

Β. Φάση Διαμόρφωσης Συστάδας

Μετά το πέρας της φάσης ενημέρωσης, λαμβάνεται απόφαση από τους απλούς κόμβους του δικτύου σχετικά με την συστάδα στην οποία θα ανήκουν για τον τρέχον γύρο. Η απόφαση αυτή βασίζεται στην ισχύ του σήματος που λήφθηκε το μήνυμα ενημέρωσης. Υποθέτοντας συμμετρικά κανάλια μετάδοσης, το μήνυμα ενημέρωσης του επικεφαλή με τη μεγαλύτερη ισχύ υποδεικνύει τον επικεφαλή του οποίου η μετάδοση δεδομένων προς αυτόν γίνεται με την ελάχιστη ενέργεια. Επομένως κάθε κόμβος τοποθετείται στη συστάδα του κόμβου επικεφαλή με τον οποίο έχει το ελάχιστο ενεργειακά κόστος επικοινωνίας. Το κόστος επικοινωνίας δεν είναι πάντοτε ανάλογο της απόστασης. Για παράδειγμα, αν κάποιος απλός κόμβος Α βρίσκεται γεωγραφικά πιο κοντά με τον κόμβο επικεφαλή Β παρά από τον κόμβο επικεφαλή Γ, τότε ο Α επιλέγει να τοποθετηθεί στη συστάδα του Β, αφού μικρότερη απόσταση συνεπάγεται χαμηλότερο ενεργειακό κόστος επικοινωνίας. Όμως στην περίπτωση που ανάμεσα στον Α και στον Β παρεμβάλλεται κάποιο εμπόδιο(π.χ. κτίριο, δέντρο), ο Α επιλέγει να τοποθετηθεί στη συστάδα του Γ, που βρίσκεται πιο μακριά του γεωγραφικά αλλά πιο «κοντά» του ενεργειακά.

Μετά την επιλογή της συστάδας που κάθε απλός κόμβος του δικτύου επιλέγει να ανήκει, λαμβάνει χώρα η ειδοποίηση των κόμβων επικεφαλείς για τους απλούς κόμβους αυτούς που έχουν τοποθετηθεί στη συστάδα του. Έτσι, κάθε απλός κόμβος είναι υπεύθυνος για τη μετάδοση της παραπάνω πληροφορία στον κόμβο επικεφαλή της συστάδας του, κάνοντας χρήση του πρωτοκόλλου CSMA MAC. Άμεση απόρροια αυτού αποτελεί η διατήρηση των πομπών των κόμβων επικεφαλείς σε λειτουργία.

Γ. Δημιουργία Προγράμματος

Κατά τη διάρκεια της δημιουργίας προγράμματος, κάθε επικεφαλής κόμβος λαμβάνει τα μηνύματα από τους απλούς κόμβους που συμμετέχουν στη συστάδα του. Βασιζόμενος στο πλήθος των κόμβων αυτών ο επικεφαλής κόμβος δημιουργεί TDMA πρόγραμμα, το οποίο καθορίζει τον χρόνο που μπορεί να μεταδώσει κάθε ένας από τους απλούς κόμβους της συστάδας του (δεσμεύει χρονοθυρίδες). Στη συνέχεια, πραγματοποιείται η μετάδοση αυτού από τον επικεφαλής κόμβο στους απλούς κόμβους της συστάδας του.

Δ. Μετάδοση Δεδομένων

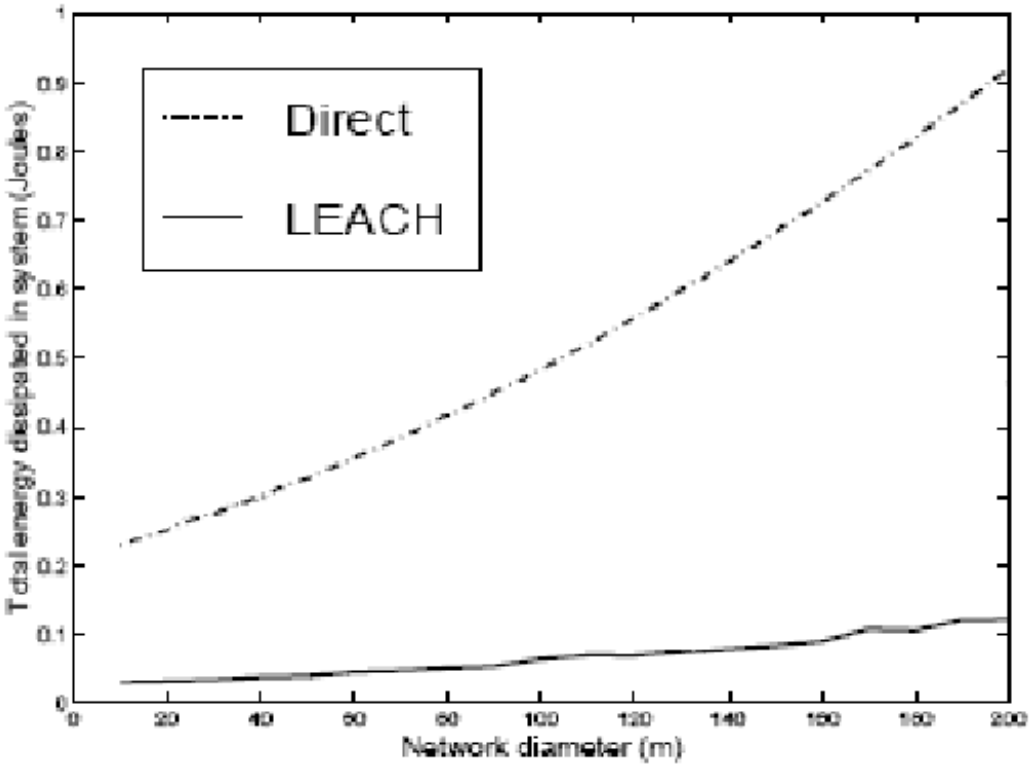
Αφού έχουν δημιουργηθεί οι συστάδες και το πρόγραμμα TDMA έχει καθοριστεί, μπορεί να ξεκινήσει η μετάδοση δεδομένων. Υποθέτοντας ότι όλοι οι κόμβοι έχουν δεδομένα να μεταδώσουν, τα αποστέλλουν κατά τη διάρκεια του δικού τους χρόνου μετάδοσης στον επικεφαλής. Ο πομπός κάθε μη επικεφαλή κόμβου μπορεί να

απενεργοποιηθεί μέχρι να έρθει ο δικός του χρόνος εκπομπής, μειώνοντας με αυτόν τον τρόπο την απώλεια ενέργειας στους κόμβους αυτούς. Ενώ, ο κόμβος επικεφαλής πρέπει να έχει ανοιχτό τον δέκτη για να μπορεί να λάβει τα μηνύματα των κόμβων της συστάδας του. Όταν όλα τα δεδομένα έχουν ληφθεί, ο κόμβος επικεφαλής εκτελεί λειτουργίες επεξεργασίας σήματος για τη συμπίεση των δεδομένων σε ένα μόνο σήμα. Το σήμα αυτό αποστέλλεται τελικά στον σταθμό βάσης. Η μετάδοση αυτή χαρακτηρίζεται ενεργειακά υψηλή μετάδοση καθώς ο σταθμός βάσης είναι απομακρυσμένος.

Μετά από κάποιο προκαθορισμένο χρονικό διάστημα, ξεκινάει ο επόμενος γύρος με κάθε κόμβο να αποφασίζει για τον αν θα γίνει επικεφαλής, κοκ.

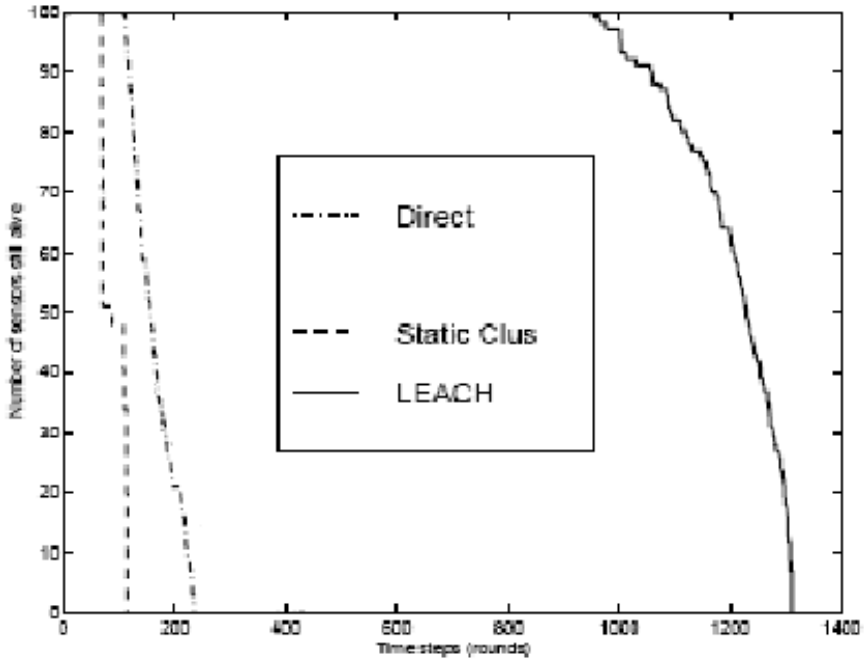
Σύγκριση LEACH και direct (One-Hop) communication

Σε αυτό το σημείο γίνεται σύγκριση μεταξύ των προτοκόλλων LEACH και direct communication σύμφωνα με μετρήσεις από προσομειώσεις αυτών σε δίκτυα. Στην εικόνα που ακολουθεί απεικονίζεται η λειτουργία κάθε αλγορίθμου καθώς το μέγεθος του δικτύου μεγαλώνει. Από αυτήν προκύπτει ότο το LEACH επιτυγχάνει 7 με 8 φορές μείωση της ενέργειας σε σχέση με την άμεση επικοινωνία.



Εικόνα 24 - Συνολική ενέργεια που καταναλώνεται στο δίκτυο

Πιο κάτω απεικονίζεται το ποσό της ενέργειας που καταναλώνεται στο δίκτυο με το LEACH, με πρωτόκολλο static clustering και με το πρωτόκολλο άμεσης επικοινωνίας.



Εικόνα 25 - Χρόνος ζωής δικτύου

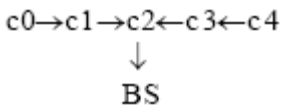
3.6.2 PEGASIS

Βασική ιδέα του PEGASIS(Power-Efficient Gathering in Sensor Information System) [6], αποτελεί κάθε κόμβος του δικτύου να λαμβάνει από και να μεταδίδει προς τον πιο κοντινό του γείτονα. Επιπρόσθετα, ο κόμβος αρχηγός που επιλέγεται για την αποστολή των δεδομένων στο σταθμό βάσης δεν είναι σταθερός, εναλλάσσεται σε κάθε γύρο του αλγόριθμου. Το παραπάνω έχει σαν αποτέλεσμα την ομοιόμορφη κατανομή της ενέργειας στους κόμβους του δικτύου. Σε αρχικό στάδιο οι κόμβοι τοποθετούνται τυχαία στο δίκτυο. Ενώ στη συνέχεια οργανώνονται έτσι ώστε να σχηματίζουν αλυσίδα. Αυτό επιτυγχάνεται είτε από τους ίδιους τους κόμβους κάνοντας χρήση ενός άπληστου (greedy) αλγόριθμου είτε από τον σταθμό βάσης ο οποίος αφού δημιουργήσει την αλυσίδα τη μεταδίδει στους κόμβους του δικτύου.

Στη συνέχεια γίνεται η υπόθεση ότι όλοι οι κόμβοι έχουν σφαιρική γνώση του δικτύου και κάνουν χρήση ενός άπληστου αλγόριθμου για την κατασκευή της

αλυσίδας, η οποία λαμβάνει χώρα πριν τον πρώτο γύρο επικοινωνίας. Για την κατασκευή της αλυσίδας, αρχικά επιλέγεται ο πιο απομακρυσμένος κόμβος από τον σταθμό βάσης. Στη συνέχεια επιλέγεται ο πιο κοντινός γείτονας σε αυτόν, κοκ έως ότου όλοι οι κόμβοι του δικτύου να έχουν συνδεθούν μεταξύ τους σε μια αλυσίδα. Όπως είναι φυσικό η απόσταση μεταξύ των γειτόνων θα αυξάνεται βαθμιαία καθώς κόμβοι που έχουν ήδη επισκεφθεί δεν μπορούν να επιλεγούν ξανά. Όταν ένας κόμβος εξαντλήσει την ενέργειά του ("πεθάνει") η αλυσίδα ξανακατασκευάζεται με τον ίδιο τρόπο προσπερνώντας αυτήν την φορά τον ανενεργό κόμβο.

Κάθε κόμβος του δικτύου λαμβάνει δεδομένα από τον γείτονά του, τα συμπιέζει μαζί με τα δικά του και τα αποστέλλει στον άλλο γείτονα της αλυσίδας. Με τον τρόπο αυτό πραγματοποιείται η συλλογή δεδομένων σε κάθε γύρο. Ο κόμβος που επιλέγεται ως αρχηγός και αναλαμβάνει την αποστολή δεδομένων στον σταθμό βάσης αλλάζει κυκλικά σε κάθε γύρω. Έτσι, στον γύρω i επιλέγεται για αρχηγός ο κόμβος i mod N, όπου N το πλήθος των κόμβων του δικτύου. Με τον τρόπο αυτό ο αρχηγός σε κάθε γύρο βρίσκεται σε τυχαία θέση της αλυσίδας. Το γεγονός αυτό είναι ιδιαίτερα σημαντικό καθώς οι κόμβοι απενεργοποιούνται σε τυχαίες τοποθεσίες. Η ιδέα της απενεργοποίησης(ενεργειακής εξάντλησης) των κόμβων σε τυχαίες θέσεις είναι για να κάνουν το δίκτυο πιο εύρωστο στις βλάβες. Σε κάθε γύρο γίνεται χρήση μιας απλής προσέγγισης μετάδοσης ενός κουπονιού ελέγχου από τον αρχηγό για την εκκίνηση της αποστολής δεδομένων των κόμβων. Το κόστος της μετάδοσης αυτής είναι πολύ μικρό αφού το μέγεθος του κουπονιού είναι πολύ μικρό. Στο σχήμα που ακολουθεί ο κόμβος c2 είναι ο αρχηγός, ο οποίος μεταδίδει τα το κουπόνι κατά μήκος της αλυσίδας στον κόμβο c0. Ο κόμβος c0 αποστέλλει τα δεδομένα του προς τον κόμβο c2. Μετά τη λήψη των δεδομένων από τον κόμβο c2, ο τελευταίος προωθεί το κουπόνι στον κόμβο c4. Ο κόμβος αυτός με τη σειρά του μεταδίδει τα δεδομένα του στον κόμβο c2.



Εικόνα 26 - Τεχνική περάσματος κουπονιού

Το PEGASIS εφαρμόζει συνάθροιση δεδομένων σε κάθε κόμβο εκτός από τους κόμβους από όπου ξεκινάει η μετάδοση δεδομένων (ακριανοί κόμβοι αλυσίδας). Κάθε κόμβος πραγματοποιεί συνάθροιση στα δεδομένα του γείτονά του με τα δικά του δημιουργώντας ένα πακέτο ίδιου μεγέθους και το μεταδίδει στον άλλο γείτονά του (αν έχει δύο γείτονες). Στο παραπάνω παράδειγμα, ο κόμβος c0 μεταδίδει τα

δεδομένα του στον κόμβο c1. Ο κόμβος αυτός συμπιέζει τα δεδομένα που έλαβε από τον c0 μαζί με τα δικά του και τα μεταδίδει στον αρχηγό κόμβο. Αφού ο κόμβος c4 έχει λάβει το κουπόνι από τον αρχηγό κόμβο, αρχίζει τη μετάδοση των δεδομένων του στον κόμβο c3. Ο κόμβος αυτός συμπιέζει τα δεδομένα που έλαβε από τον c4 μαζί με δικά του και τα αποστέλλει στον c2. Ο κόμβος αυτός περιμένει μέχρι να λάβει δεδομένα και από τους δύο γείτονές του και στη συνέχεια συμπιέζει τα δικά του δεδομένα με αυτά των γειτόνων του και τα μεταδίδει στον σταθμό βάσης.

Ένας παράγοντας που λαμβάνεται υπόψη κατά τη συλλογή δεδομένων αποτελεί η μέση καθυστέρηση κίνησης ανά γύρο. Σύμφωνα με το πρωτόκολλο άμεσης επικοινωνίας, οι κόμβοι μεταδίδουν μία φορά στο σταθμό βάσης δημιουργώντας καθυστέρηση ίση με το πλήθος κόμβων του δικτύου (1 μονάδα καθυστέρησης ανά μετάδοση). Με την δημιουργία γραμμικής αλυσίδας που προτείνει το PEGASIS ενώ είναι ενεργειακά αποδοτικό απαιτεί την ίδια καθυστέρηση που έχει το πρωτόκολλο άμεσης επικοινωνίας. Το γεγονός αυτό οφείλεται στο ότι οι μεταδόσεις είναι σειριακές, οπότε για μετάδοση δεδομένων σε δίκτυο N κόμβων απαιτείται καθυστέρηση ίση με N μονάδες (όσο πλήθος μεταδόσεων). Για τη μείωση της καθυστέρησης και συνεπώς τη βελτίωση του αλγορίθμου απαιτείται η ταυτόχρονη μετάδοση από τους κόμβους. Στην προσπάθεια αυτή προτάθηκαν τα ακόλουθα σχήματα: (1) δυαδικό σχήμα που βασίζεται στην έννοια της αλυσίδας και οι κόμβοι του δικτύου υποστηρίζουν CDMA και (2) σχήμα τριών επιπέδων το οποίο βασίζεται στην δημιουργία αλυσίδας χωρίς την υποστήριξη CDMA από τους κόμβους του δικτύου [7].

Σύμφωνα με τον πρώτο αλγόριθμο οι κόμβοι επικοινωνούν ανά δύο μεταξύ τους, οπότε επιτυγχάνεται N/2(N πλήθος κόμβων) παράλληλες μεταδόσεις κατά τη αρχική φάση ενός γύρου(πρώτο επίπεδο) χωρίς ιδιαίτερες παρεμβολές(λόγω της ύπαρξης CDMA). Ενώ για την ολοκλήρωση ενός γύρου απαιτούνται logN επίπεδα. Για την κατανόηση των παραπάνω ακολουθεί η περιγραφή ενός δικτύου 100 κόμβων που έχει υιοθετήσει τον αλγόριθμο αυτό.

BS

↑

c3

c3 ←c7

c1→ c3 c5→ c7

c0→c1 c2→c3 c4→c5 c6→c7

Εικόνα 27 - Data gathering in a chain-based binary scheme

Για δίκτυα με κόμβους που δεν υποστηρίζουν CDMA υιοθετείται ο δεύτερος αλγόριθμος τριών επιπέδων βασιζόμενος στη δημιουργία αλυσίδας. Σύμφωνα με αυτόν τον αλγόριθμο ένας γύρος ολοκληρώνεται σε τρία επίπεδα. Παράλληλα μπορούν να μεταδώσουν κόμβοι που είναι γεωγραφικά απομακρυσμένοι διαφορετικά υπάρχει κίνδυνος παρεμβολών. Η λειτουργία του αλγορίθμου αυτού γίνεται κατανοητή με το παράδειγμα που ακολουθεί.

BS

↑

c18

c18←c68

c8 →c18←c28←c38←c48 c58→ c68← c78 ←c88← c98

c0→c1→c2→c3→c4→c5→c6→c7→c8←c9 c10→c11…c18←c19 …c90→c91…c98← c99

Εικόνα 28 - Chain-based 3 level scheme for a sensor network with non-CDMA nodes

Με τα δύο παραπάνω πρωτόκολλα επιτυγχάνεται τόσο εξοικονόμηση ενέργειας αφού ακολουθούν την αλυσιδωτή δομή του PEGASIS όσο και μείωση της καθυστέρησης μετάδοσης. Αυτό που επιτυγχάνεται είναι η μείωση του γινομένου ενέργειας και καθυστέρησης (energy x delay). Στη συνέχεια ακολουθεί πίνακας που απεικονίζει συγκρίσεις των πρωτοκόλλων άμεσης επικοινωνίας, LEACH, PEGASIS, chain-based binary και chain-based 3 level με βάση το γινόμενο αυτό για δίκτυο 100m x 100m. Η ενέργεια μετριέται σε Joules, ενώ η καθυστέρηση σε μεταδόσεις πακέτων (units). Για παράδειγμα για τη μετάδοση ενός πακέτου από έναν κόμβο A σε ένα γείτονά του κόμβο B θεωρείται καθυστέρηση μιας μονάδας.

Protocol	Energy	Delay	*Energy *Delay*
Direct	1.280459	100	128.0459
PEGASIS	0.036107	100	3.6107
LEACH (CDMA nodes)	0.204786	27	5.5292
Binary (CDMA nodes)	0.055898	8	0.4516
3 Level (non-CDMA nodes)	0.058287	15	0.8743

Όπως φαίνεται από τον πίνακα το PEGASIS είναι καλύτερο από το LEACH. Πρώτα από όλα το PEGASIS δεν έχει overhead που δημιουργείται στο LEACH κατά τη διαδικασία μορφοποίησης συστάδων. Επιπρόσθετα, στο PEGASIS η μετάδοση δεδομένων γίνεται από γειτονικούς κόμβους, γεγονός που οδηγεί σε μικρότερη απώλεια ενέργειας. Παράλληλα, στο PEGASIS ένας κόμβος είναι υπεύθυνος για τη μετάδοση δεδομένων στο σταθμό βάσης σε αντίθεση με το LEACH όπου οι κόμβοι επικεφαλείς έχουν αναλάβει την εργασία αυτή. Για όλους τους παραπάνω λόγους το PEGASIS επιτυγχάνει μεγαλύτερη εξοικονόμηση ενέργειας από το LEACH. Και στους δύο αλγόριθμους πραγματοποιείται αλλαγή των κόμβων που αναλαμβάνουν την εργασία μετάδοσης στον σταθμό βάσης. Με τον τρόπο αυτό επιτυγχάνεται περαιτέρω εξισορρόπηση της ενέργειας που καταναλώνεται στο δίκτυο, ενώ παράλληλα διατηρείται η ευρωστία του δικτύου καθώς οι κόμβοι εξαντλούνται σε τυχαίες τοποθεσίες. Με την κατανομή της ενέργειας στους κόμβους αυξάνεται ο χρόνος ζωής και η ποιότητα του δικτύου. Τέλος, από μετρήσεις έχει δειχθεί ότι ο αλγόριθμος PEGASIS έχει καλύτερη επίδοση όσο το μέγεθος του δικτύου μεγαλώνει.

Όπως είναι φυσικό το μέγεθος των καθυστερήσεων μειώνεται σημαντικά στα δύο σχήματα βελτίωσης του PEGASIS (δυαδικό, τριών επιπέδων) καθώς πραγματοποιούνται σε αυτά παράλληλες μεταδόσεις. Η απώλεια ενέργειας των δύο αυτών σχημάτων είναι λίγο μεγαλύτερη από αυτή του PEGASIS. Το γεγονός αυτό οφείλεται στην αύξηση της απόστασης κόμβων (μεγαλύτερη απώλεια ενέργειας) στα επίπεδα (εκτός του πρώτου) κάθε σχήματος(δυαδικό, τριών επιπέδων).

Συνοψίζοντας από τον πίνακα το δυαδικό σχήμα είναι περίπου 12 φορές καλύτερο από τον αλγόριθμο LEACH και 280 φορές καλύτερο από την άμεση επικοινωνία. Ενώ το σχήμα τριών επιπέδων είναι περίπου 4 φορές καλύτερο από το PEGASIS και 140 φορές καλύτερο από τον αλγόριθμο άμεσης επικοινωνίας. Παρ' όλα αυτά δεν είναι σαφές ποιο είναι το καλύτερο σχήμα το οποίο να βελτιστοποιεί την τιμή του μέτρου Ενέργειας x Καθυστέρηση. Καθώς η απώλεια ενέργειας για μετάδοση εξαρτάται από την χωρική κατανομή των κόμβων, ίσως να μην υπάρχει μόνο ένα σχήμα βέλτιστο για όλα τα μεγέθη των δικτύων.

3.6.3 TEEN

Το TEEN (Threshold sensitive Energy Efficient sensor Network protocol) [8] αποτελεί το πρώτο πρωτόκολλο που αναπτύχθηκε για αναδραστικά (reactive) δίκτυα. Το TEEN, όπως και το LEACH, χρησιμοποιεί την ιδέα των συστάδων. Κάθε συστάδα έχει έναν επικεφαλή ο οποίος είναι υπεύθυνος για τη συλλογή των δεδομένων, τη συνάθροιση αυτών και σε τελικό στάδιο την προώθησή τους είτε στον σταθμό βάσης είτε στον ιεραρχικά επόμενο επικεφαλή κόμβο.

Αφού διαμορφωθούν οι συστάδες του δικτύου, κάθε κόμβος επικεφαλής μεταδίδει στους κόμβους-μέλη του:

Attributes: σύνολο από φυσικές παραμέτρους - γνωρίσματα που ενδιαφέρουν το χρήστη

Hard Threshold: τιμή κατωφλίου που ορίζεται για καθεμία από τις αντίστοιχες τιμές των γνωρισμάτων. Σε περίπτωση που η απόλυτη τιμή του γνωρίσματος που έχει αντιληφθεί ο αισθητήρας από το περιβάλλον ξεπερνάει την τιμή του ισχυρού κατωφλίου, αυτή αποθηκεύεται σε εσωτερική μεταβλητή του κόμβου. Η τιμή αυτή χαρακτηρίζεται ως sensed value.

Soft Threshold: μικρή αλλαγή στην τιμή του γνωρίσματος που έχει γίνει αντιληπτή από τον πράκτορα. Αν η νέα τιμή του γνωρίσματος είναι ίση ή μεγαλύτερη από την αποθηκευμένη (sensed value) με τιμή ίση με αυτήν του ασθενούς κατωφλίου, τότε ο πομπός του κόμβου ενεργοποιείται και μεταδίδει τα δεδομένα που έχει αντιληφθεί στον κόμβο επικεφαλή.

Στο TEEN η παρακολούθηση του δικτύου είναι συνεχής. Την πρώτη φορά που μια παράμετρος από το σύνολο των γνωρισμάτων ξεπεράσει την τιμή του ισχυρού κατωφλίου, ο κόμβος αποθηκεύει σε μια εσωτερική μεταβλητή αυτού την παραπάνω τιμή (sensed value). Ο κόμβος μεταδίδει τα δεδομένα που έχει αντιληφθεί από το περιβάλλον όταν εκπληρώνονται και οι δύο ακόλουθες συνθήκες:

1. Η τρέχουσα τιμή του γνωρίσματος που έχει γίνει αντιληπτό είναι μεγαλύτερη του ισχυρού περιορισμού, και
2. Η τιμή του γνωρίσματος που συνεχώς αντιλαμβάνεται ο κόμβος διαφέρει από την sensed value κατά πόσο ίσο ή μεγαλύτερο από την τιμή του ασθενούς κατωφλίου.

Οποτεδήποτε ένας κόμβος μεταδίδει δεδομένα, η sensed value αυτού ορίζεται ίση με την τρέχουσα τιμή του γνωρίσματος που έγινε αντιληπτό.

Με αυτόν τον τρόπο, το ισχυρό κατώφλι προσπαθεί να μειώσει το πλήθος των μεταδόσεων επιτρέποντας στους κόμβους να μεταδώσουν μόνο όταν τα γνωρίσματα που εντοπίστηκαν είναι μέσα στο πεδίο ενδιαφέροντος του χρήστη. Το ασθενές κατώφλι μειώνει περισσότερο τον αριθμό των μεταδόσεων εξαφανίζοντας τις μεταδόσεις που αναπόφευκτα θα συνέβαιναν όταν υπάρχει μικρή ή καθόλου αλλαγή στο γνώρισμα που εντοπίζεται.

Στη συνέχεια γίνεται αναφορά στα κυριότερα χαρακτηριστικά του αλγορίθμου αυτού. Πρώτα απ' όλα, όταν τα «κρίσιμα» δεδομένα γίνονται αντιληπτά από τους αισθητήρες του δικτύου μεταδίδονται στιγμιαία στον σταθμό βάσης. Σε αυτό οφείλεται η καταλληλότητα του αλγορίθμου για εφαρμογές που απαιτούν άμεση απόκριση σε μη-συνηθισμένα γεγονότα στο δίκτυο. Δεύτερον, η μετάδοση μηνύματος χαρακτηρίζεται υψηλότερη ενεργειακά ενέργεια από αυτήν της συνεχούς αντίληψης του περιβάλλοντος. Επιπρόσθετα, οι τιμές του ασθενούς κατωφλίου ποικίλλουν ανάλογα με την κρισιμότητα του γνωρίσματος και της εφαρμογής. Μικρή τιμή ασθενούς κατωφλίου επιτυγχάνει την απόκτηση πιο ακριβής εικόνας του δικτύου, με αντάλλαγμα την αυξημένη απώλεια ενέργειας λόγω συχνότερων μεταδόσεων. Αφήνεται στον χρήστη για την επιλογή στο trade-off ανάμεσα στην εξοικονόμηση ενέργειας και στην ακρίβεια της εικόνας του δικτύου. Τέλος, σε κάθε αλλαγή κόμβου επικεφαλή, μεταδίδονται ξανά οι τιμές των γνωρισμάτων, δίνοντας στον χρήστη τη δυνατότητα αλλαγής αυτών.

Κύριο μειονέκτημα του αλγορίθμου αυτού αποτελεί το γεγονός ότι αν ποτέ οι κόμβοι δεν αντιληφθούν από το περιβάλλον τιμή για κάποιο γνώρισμα μεγαλύτερη από αυτό που έχει προσδιοριστεί, οι κόμβοι δεν θα μεταδώσουν. Άμεση συνέπεια αυτού είναι ότι ο χρήστης δεν θα πάρει ποτέ δεδομένα από το δίκτυο και δεν θα γνωρίζει αν οι κόμβοι έχουν εξαντληθεί. Για το λόγο αυτό, ο TEEN δεν είναι κατάλληλος για εφαρμογές που ζητούν ανά τακτά χρονικά διαστήματα να γνωρίζουν την κατάσταση του δικτύου. Παραδείγματα εφαρμογών κατάλληλων για το πρωτόκολλο αυτό αποτελούν η ανίχνευση εισβολών, η ανίχνευση πυρκαγιάς, κ.ο.κ.

3.6.4 APTEEN

Το πρωτόκολλο APTEEN (Adaptive Periodic Threshold-sensitive Energy Efficient Sensor Network Protocol) [9] αναπτύχθηκε για υβριδικά δίκτυα. Το APTEEN ακριβώς όπως το LEACH και το TEEN έχει υιοθετήσει τη λογική των συστάδων και των επικεφαλείς αυτών. Έτσι, μετά την ολοκλήρωση της φάσης δημιουργίας συστάδας, οι επικεφαλής κόμβοι μεταδίδουν τα ακόλουθα στους κόμβους μέλη τους:

Γνωρίσματα (Attributes): σύνολο από φυσικές παραμέτρους που ο χρήστης ενδιαφέρεται να μάθει

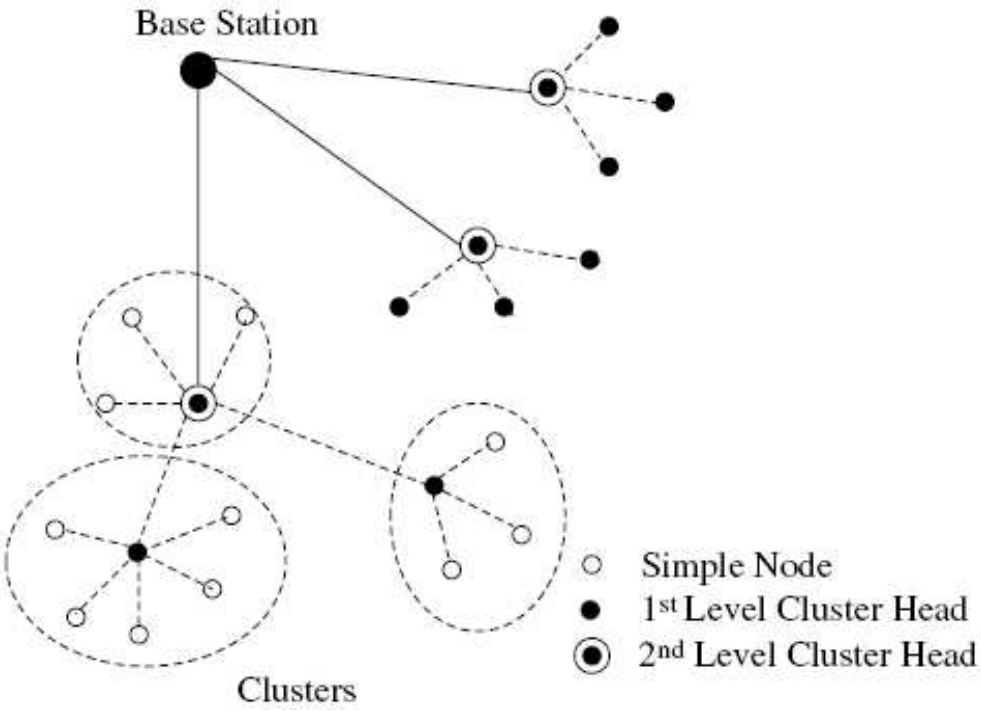
Κατώφλια (Thresholds): αυτή η παράμετρος περιλαμβάνει το ισχυρό κατώφλι(H_T) και το ασθενές κατώφλι(S_T). Το H_T είναι μια συγκεκριμένη τιμή γνωρίσματος η οποία αν ξεπεραστεί ο κόμβος που έχει αντιληφθεί αυτήν την τιμή αποθηκεύει αυτήν σε εσωτερική μεταβλητή του κόμβου. Το S_T αποτελεί μια μικρή αλλαγή στην τιμή του γνωρίσματος η οποία προκαλεί την ενεργοποίηση του πομπού του κόμβου και την αποστολή δεδομένων στον επικεφαλής.

Πρόγραμμα (Schedule): αποτελεί ένα TDMA πρόγραμμα το οποίο θέτει χρονοθυρίδες στους κόμβους, δηλαδή καθορίζει τη χρονική στιγμή στη διάρκεια της οποίας οι κόμβοι μέλη κάθε συστάδας μπορεί να εκπέμψει.

Χρόνος (Count Time): αποτελεί τη μέγιστη περίοδο ανάμεσα σε δύο επιτυχημένες αναφορές που έχουν αποσταλεί από έναν κόμβο.

Σε ένα δίκτυο αισθητήρων, οι κόμβοι που βρίσκονται κοντά ανήκουν στην ίδια συστάδα, αντιλαμβάνονται τα ίδια δεδομένα και προσπαθούν να στείλουν τα δεδομένα τους ταυτόχρονα, προκαλώντας με τον τρόπο αυτό συγκρούσεις. Για το λόγο αυτό, γίνεται χρήση του TDMA όπου εκχωρούνται διαφορετικές χρονικές στιγμές μετάδοσης σε κάθε κόμβο της συστάδας.

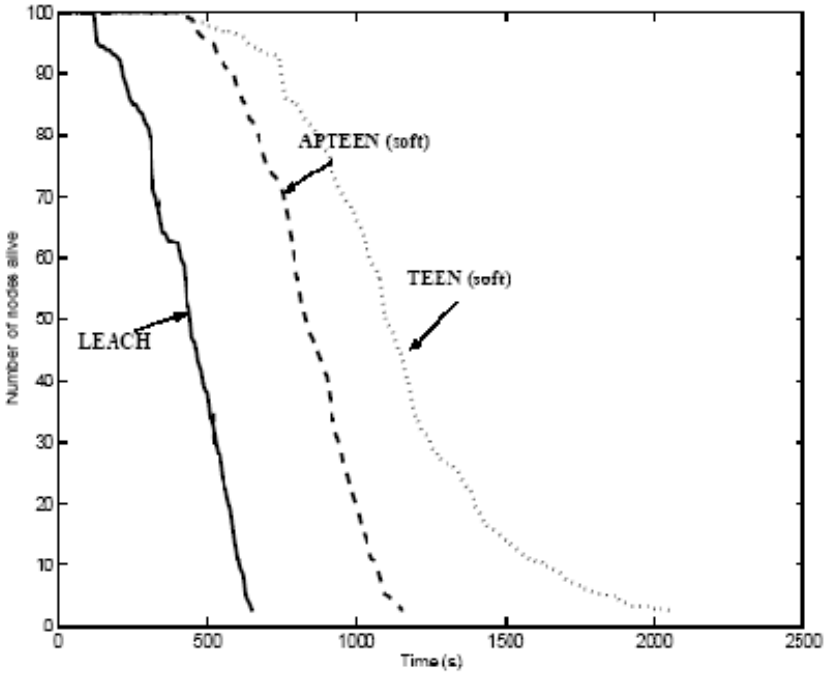
Στη συνέχεια γίνεται αναφορά στα κύρια χαρακτηριστικά του πρωτοκόλλου αυτού. Πρώτα από όλα, ο χρήστης έχει πλήρη της κατάστασης του δικτύου με την περιοδική αποστολή δεδομένων των κόμβων του δικτύου σε αυτόν. Επιπρόσθετα, αντιδρά άμεσα σε ξαφνικές αλλαγές στο δίκτυο, γεγονός που του δίνει τη δυνατότητα να ανταποκρίνεται σε κρίσιμες καταστάσεις. Για το λόγο αυτό, το APTEEN συνδυάζει τόσο προδραστικές όσο και αναδραστικές στρατηγικές. Δεύτερον, προσφέρει προσαρμοστικότητα επιτρέποντας στο χρήστη την αλλαγή τιμών τόσο του count time όσο και των κατωφλίων. Ανάλογα με τις τιμές που αποδίδονται στις παραπάνω μεταβλητές προσαρμόζεται και το ποσό απώλειας/εξοικονόμησης ενέργειας. Ένα υβριδικό δίκτυο μπορεί να μετατραπεί σε προδραστικό ή αναδραστικό ανάλογα με τις τιμές των κατωφλίων και του count time. Το κύριο μειονέκτημα αυτού του πρωτοκόλλου αποτελεί η επιπρόσθετη πολυπλοκότητα για την δημιουργία των συναρτήσεων κατωφλίων και του count time. Ωστόσο, αυτό αποτελεί άλλο ένα trade-off ανάμεσα στην πολυπλοκότητα και στην προσαρμοστικότητα.



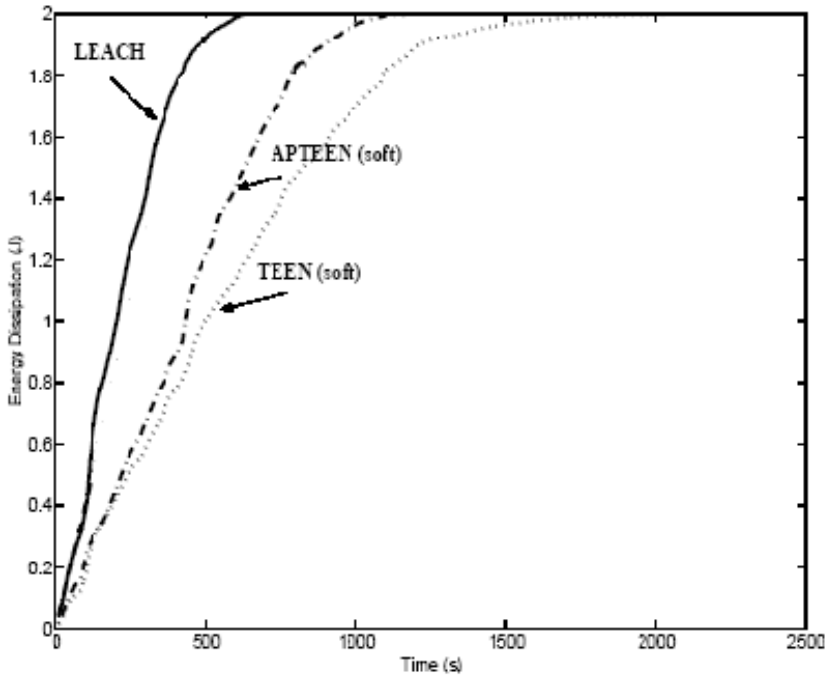
Εικόνα 29 - Hierarchcal Clustering in TEEN and APTEEN

Σύγκριση πρωτοκόλλων LEACH, TEEN, APTEEN

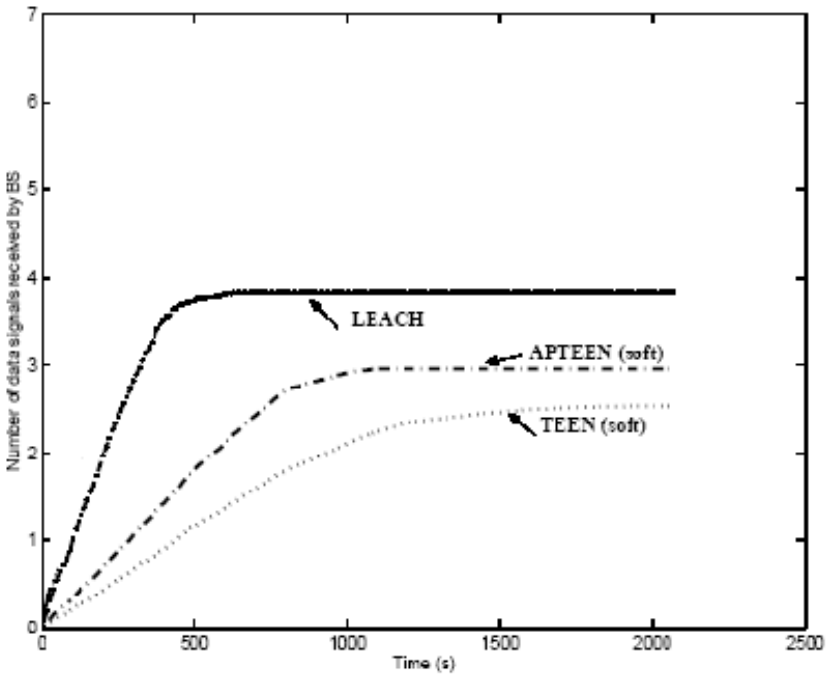
Στις εικόνες που ακολουθούν γίνεται σύγκριση των αλγορίθμων LEACH, TEEN και APTEEN σε σχέση με την κατανάλωση ενέργειας (εικόνα 30), το πλήθος των ενεργών κόμβων (εικόνα 31) και του συνολικού πλήθους δεδομένων που λαμβάνονται από τον σταθμό βάσης στη διάρκεια του χρόνου (εικόνα 32). Η απόδοση του APTEEN βρίσκεται ανάμεσα σε αυτή του TEEN και του LEACH. Αυτό ήταν αναμενόμενο καθώς το TEEN μεταδίδει μόνο όταν αντιληφθεί κρίσιμα δεδομένα στο περιβάλλον του. Για τη εξάληψη των μειονεκτημάτων του πρωτοκόλλου αυτού, προτάθηκε το APTEEN. Επίσης το APTEEN είναι πιο αποδοτικό από το LEACH, αφού το πρώτο μεταδίδει δεδομένα ανάλογα με τις τιμές των κατωφλίων και του count time ενώ το δεύτερο μεταδίδει δεδομένα συνεχώς.



Εικόνα 30 - Πλήθος ενεργών κόμβων στο χρόνο



Εικόνα 31 - Κατανάλωση ενέργειας δικτύου στο χρόνο



Εικόνα 32 - Δεδομένα που λαμβάνονται από τον σταθμό βάσης στο χρόνο

3.7 Location Based Πρωτόκολλα

Σε αυτό το είδος δρομολόγησης οι κόμβοι αισθητήρων σηματοδοτούνται από την τοποθεσία τους στο δίκτυο. Η απόσταση ανάμεσα σε γειτονικούς κόμβους μπορεί να προσδιοριστεί ανάλογα με την ισχύ του εισερχόμενου σήματος. Εναλλακτικά, η τοποθεσία των κόμβων μπορεί να προσδιοριστεί άμεσα μέσω της επικοινωνίας με δορυφόρο, χρησιμοποιώντας GPS, στην περίπτωση που οι κόμβοι διαθέτουν δέκτη GPS. Για την εξοικονόμηση ενέργειας μερικοί αλγόριθμοι τοποθετούν τους κόμβους του δικτύου που είναι ανενεργοί σε κατάσταση ύπνου. Με τον τρόπο αυτό όσο περισσότεροι κόμβοι βρίσκονται σε κατάσταση ύπνου στο δίκτυο τόσο μεγαλύτερο ποσό ενέργειας εξοικονομείται. Στη συνέχεια ακολουθεί μια παρουσίαση των πιο διαδεδομένων αλγορίθμων δρομολόγησης που βασίζονται στη γεωγραφία των κόμβων του δικτύου [11].

3.7.1 GEAR

Κίνητρο για τη δημιουργία του παρακάτω αλγορίθμου αποτέλεσε το γεγονός ότι το είδος των ερωτήσεων που απευθύνεται σε δίκτυο αισθητήρων είναι πολλές φορές γεωγραφικού περιεχομένου. Λαμβάνοντας υπόψη και την εξοικονόμηση ενέργειας που είναι επιθυμητή στα δίκτυα αισθητήρων δημιουργήθηκε ο αλγόριθμος GEAR (Geographic and Energy Aware Routing) [12]. Όπως υποδηλώνεται από το όνομά του ο αλγόριθμος αυτός λαμβάνει υπόψη του τόσο την εξοικονόμηση ενέργειας του δικτύου όσο και τη γεωγραφική θέση των κόμβων αυτού έτσι ώστε να επιτυγχάνει την καλύτερη ‘ενεργειακά’ και γεωγραφικά δρομολόγηση των πακέτων. Για παράδειγμα, όταν το ενδιαφέρον μιας εφαρμογής δικτύου αισθητήρων επικεντρώνεται στην ενημέρωση για τη θερμοκρασία μιας συγκεκριμένης περιοχής. Ένας αποτελεσματικός τρόπος για την διάδοση της ερώτησης αυτής στην συγκεκριμένη περιοχή είναι η προσαρμογή της γνώσης της επιθυμητής τοποθεσίας στην ερώτηση. Με αυτόν τον τρόπο επιτυγχάνεται η απευθείας δρομολόγηση του πακέτου στην επιθυμητή περιοχή, αποφεύγοντας την πλημμύρα αυτού σε όλο το δίκτυο. Για το λόγο αυτό ο αλγόριθμος GEAR μπορεί να θεωρηθεί συμπλήρωμα του αλγορίθμου Directed Diffussion.

Περιγραφή αλγορίθμου

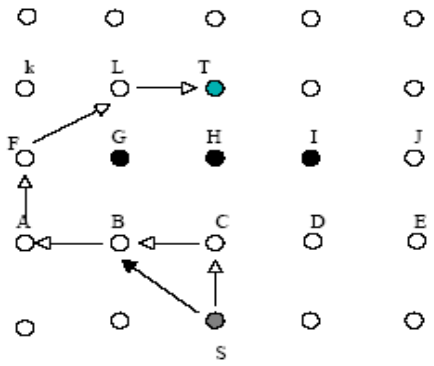
Όπως προαναφέρθηκε στόχος του αλγορίθμου είναι η δρομολόγηση πακέτων στην επιθυμητή περιοχή επιλέγοντας την αποδοτικότερη ενεργειακά διαδρομή. Η διαδικασία προώθησης πακέτου σε όλους τους κόμβους της επιθυμητής περιοχής

αποτελείται από δύο φάσεις. Στη διάρκεια της πρώτης φάσης επιτυγχάνεται η προώθηση του πακέτου στην περιοχή στόχος. Ο αλγόριθμος χρησιμοποιεί ευριστική μέθοδο για τη δρομολόγηση του πακέτου λαμβάνοντας υπόψη τη γεωγραφική θέση και ενέργεια γειτονικών κόμβων. Υπάρχουν δύο περιπτώσεις: όταν υπάρχει γείτονας πλησιέστερα στην επιθυμητή περιοχή και όταν δεν υπάρχει. Στην πρώτη περίπτωση, ο αλγόριθμος επιλέγει το γείτονα που απέχει τη μικρότερη απόσταση από την περιοχή στόχο. Στη δεύτερη περίπτωση ο αλγόριθμος επιλέγει τον γείτονα κόμβο ο οποίος ελαχιστοποιεί κάποια τιμή κόστους. Ενώ κατά τη διάρκεια της δεύτερης φάσης ο αλγόριθμος διαδίδει το πακέτο σε όλους τους κόμβους της περιοχής. Συνήθως χρησιμοποιείται αναδρομικός γεωγραφικός αλγόριθμος για τη διάδοση του πακέτου στην περιοχή. Στη συνέχεια ακολουθεί μια πιο αναλυτική επισκόπηση των δύο φάσεων του αλγορίθμου.

Κατά τη διάρκεια της πρώτης φάσης ο αλγόριθμος GEAR δρομολογεί το πακέτο από τον προορισμό προς την επιθυμητή περιοχή. Κάθε κόμβος αποθηκεύει δύο τιμές οι οποίες αναφέρονται στο κόστος που απαιτείται για την δρομολόγηση ενός πακέτου στον προορισμό του μέσω των γειτονικών κόμβων του. Οι δύο αυτές τιμές καλούνται εκτιμώμενο (estimated) και learning κόστος. Το εκτιμώμενο κόστος αποτελεί ένα συνδυασμό της εναπομείναντος ενέργειας του κόμβου και της απόστασης αυτού από την επιθυμητή περιοχή. Συμβολικά, c(N, R) = a* d(N, R) + (a-1) * e(N), όπου a μια μεταβλητή τιμή, d η απόσταση του κόμβου N από την περιοχή στόχος R και e η ενέργεια του κόμβου N. Το learning κόστος αποτελεί μια βελτίωση του εκτιμώμενου κόστους στην περίπτωση που υπάρχουν τρύπες (holes) στο δίκτυο. Τρύπα στο δίκτυο δημιουργείται όταν για έναν κόμβο δεν υπάρχει γειτονικός κόμβος που να βρίσκεται σε πιο κοντινή απόσταση προς την περιοχή στόχος από αυτόν. Όταν δεν υπάρχουν τρύπες στο δίκτυο, το learned κόστος είναι ίσο με το εκτιμώμενο. Συμβολικά, το learned κόστος είναι ίσο με h(N,R) = h(Nmin, R) + C(N, Nmin), όπου Nmin ο πλησιέστερος γείτονας του κόμβου N, R η περιοχή στόχος και C(N, Nmin) η απόσταση ανάμεσα στους δύο γειτονικούς κόμβους.

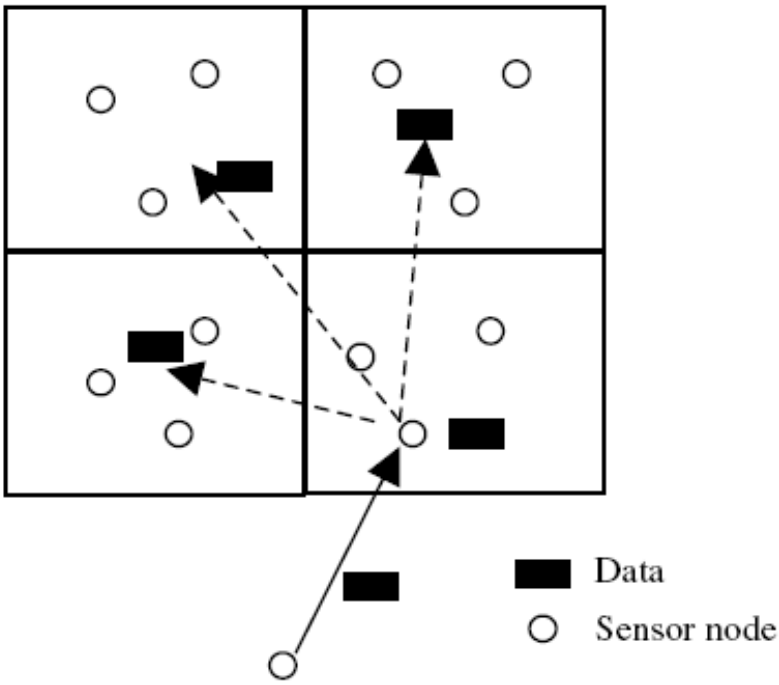
Για την καλύτερη κατανόηση της λειτουργίας του αλγορίθμου δίδεται ένα παράδειγμα. Στην τοπολογία του δικτύου της εικόνας 36, ο κόμβος S είναι υπεύθυνος για η δρομολόγηση ενός πακέτου προς την περιοχή με centroid τον κόμβο T. Οι κόμβοι G, H, I είναι κόμβοι που έχουν εξαντλήσει την ενέργειά τους. Επίσης γίνεται η υπόθεση ότι a=1, για τον υπολογισμό του εκτιμώμενου κόστους. Αρχικά, για τον κόμβο S, τα κόστη είναι h(B,T) = c(B,T) = $\sqrt{5}$, h(C,T) = c(C,T) = 2 και h(D,T) = c(D,T) = $\sqrt{5}$. Με την επιλογή του γειτονικού κόμβου με το μικρότερο κόστος, το πακέτο προωθείται στον κόμβο C. Τώρα ο κόμβος βρίσκεται σε τρύπα αφού δεν υπάρχει γειτονικός του κόμβος που να βρίσκεται πιο κοντά στον κόμβο T από ότι ο ίδιος. Στην περίπτωση αυτή ο κόμβος C προωθεί το πακέτο τυχαία σε κάποιον γείτονά του π.χ. B και ενημερώνει το learned κόστος του σε h(C,T) = h(B,T) +

C(C,B) = h(B,T) + 1. Με αυτόν τον τρόπο την επόμενη φορά που πακέτο θα πρέπει να δρομολογηθεί από τον κόμβο S στον T, θα επιλεγεί ο γειτονικός κόμβος του S, B και όχι ο C.



Εικόνα 33 - Δρομολόγηση όταν στο δίκτυο υπάρχουν τρύπες

Αφού το πακέτο φτάσει στην περιοχή στόχος απομένει η διάδοσή αυτού στους κόμβους της περιοχής αυτής. Το γεγονός αυτό αποτελεί τη δεύτερη φάση του αλγορίθμου GEAR. Υπάρχουν δύο τρόποι διάδοσης του πακέτου στους κόμβους της περιοχής, είτε με αναδρομική γεωγραφική διάδοση είτε με πλημμύρα. Ο δεύτερος τρόπος είναι κατάλληλος όταν το πλήθος των κόμβων της περιοχής δεν είναι μεγάλο, γιατί η πλημμύρα χαρακτηρίζεται ως υψηλής κατανάλωσης ενέργειας διαδικασία. Ενώ σε δίκτυα με μεγάλο πλήθος κόμβων προτιμάται η αναδρομική γεωγραφική διάδοση. Σύμφωνα με αυτή τη διαδικασία, αφού η περιοχή χωριστεί σε τέσσερις υποπεριοχές, ο κόμβος που έχει λάβει το πακέτο δημιουργεί τέσσερα αντίγραφα αυτού και τα αποστέλλει σε καθεμία από τις υποπεριοχές. Αυτή η διαδικασία χωρισμού της περιοχής και διάδοσης των πακέτων επαναλαμβάνεται μέχρι κάθε υποπεριοχή να περιλαμβάνει ένα κόμβο.



Εικόνα 34 - Recursive Geographic Forwarding in GEAR

3.8 Σύγκριση Πρωτοκόλλων Δρομολόγησης

Στον παρακάτω πίνακα απεικονίζονται συνοπτικά κύρια χαρακτηριστικά κάθε κατηγορίας πρωτοκόλλων δρομολόγησης, σύμφωνα με τον αρχικό διαχωρισμό ως προς την τοπολογία [15].

Hierarchical Routing	Flat Routing	Geographical Routing
Εκπομπή σε timeslots	Εκπομπή με βάση το περιεχόμενο	Εκπομπή με βάση το περιεχόμενο
Δεν υπάρχουν συγκρούσεις	Υπάρχουν συγκρούσεις	Υπάρχουν συγκρούσεις
Μειώνεται ο χρόνος που κάθε κόμβος παραμένει ενεργός λόγω της περιοδικής απενεργοποίησής τους	Ο χρόνος ζωής που κάθε κόμβος παραμένει ενεργός ποικίλλει, ελέγχοντας τον χρόνο που κάθε κόμβος παραμένει απενεργοποιημένος	Ο χρόνος που κάθε κόμβος παραμένει ανενεργός μειώνεται λόγω ύπαρξης ισοδύναμων κόμβων
Η συνάθροιση δεδομένων πραγματοποιείται από τους επικεφαλείς	Ο κάθε κόμβος πραγματοποιεί συνάθροιση εισερχόμενων δεδομένων από τους γείτονές του	Δεν πραγματοποιείται συγκερασμός δεδομένων
Απλή αλλά όχι βέλτιστη δρομολόγηση	Η δρομολόγηση μπορεί να γίνει βέλτιστη με επιπρόσθετη πολυπλοκότητα	Η δρομολόγηση μπορεί να γίνει βέλτιστη με επιπρόσθετη πολυπλοκότητα
Απαιτείται γενικός και τοπικός συγχρονισμός	Δεν υπάρχει συγχρονισμός	Δεν υπάρχει συγχρονισμός
Η διαμόρφωση των συστάδων δημιουργεί overhead στο δίκτυο	Οι διαδρομές διαμορφώνονται μόνο σε περιοχές που έχουν δεδομένα να μεταδώσουν	Η διαμόρφωση των εικονικών πλεγμάτων δημιουργεί overhead
Ομοιόμορφη απώλεια	Η απώλεια ενέργειας	Η απώλεια ενέργειας

ενέργειας	εξαρτάται από την κίνηση του δικτύου	εξαρτάται από το πόσο συχνά αλλάζει η τοπολογία του δικτύου
Δίκαιη κατανομή καναλιού	Δεν εγγυάται δικαιοσύνη στην κατανομή του καναλιού	Δεν εγγυάται δικαιοσύνη στην κατανομή του καναλιού

3.9 Βέλτιστη Ενεργειακά Δρομολόγηση στα Δίκτυα Αισθητήρων

Στις επόμενες παραγράφους θα παρουσιαστεί ένα πρωτόκολλο δρομολόγησης, το οποίο βασίζεται στην γνώση της γεωγραφικής θέσης των κόμβων με άμεσο τρόπο, με την χρήση GPS στους κόμβους και του οποίου η βέλτιστη διαδρομή δρομολόγησης υπολογίζεται με κριτήριο τη συμφερότερη ενεργειακά διαδρομή.

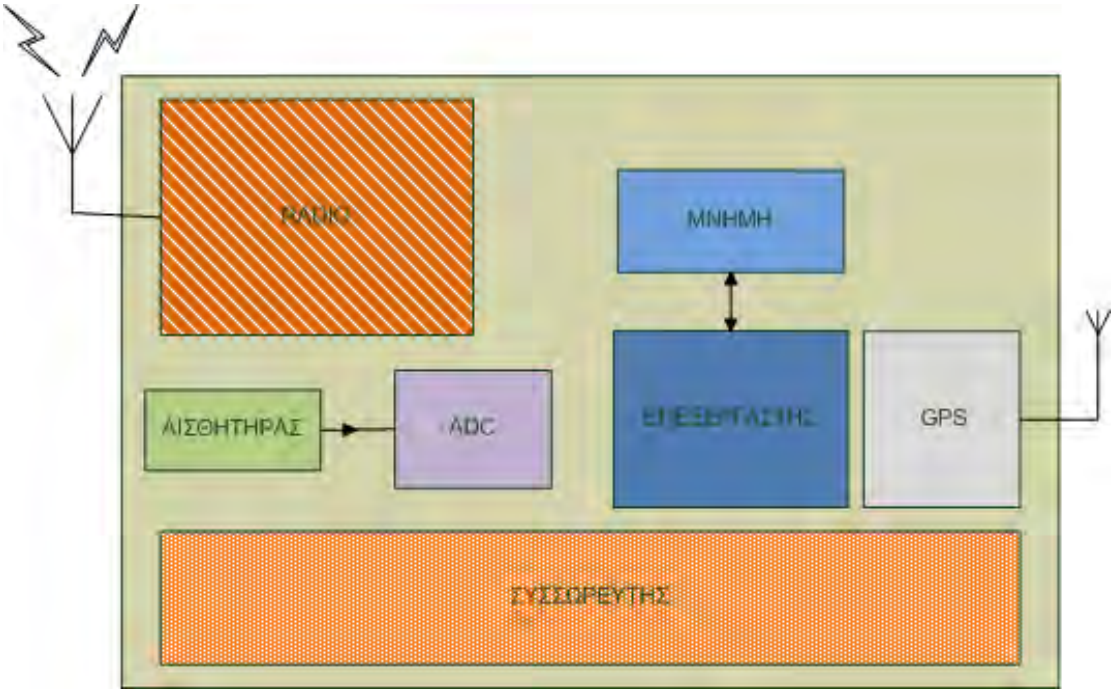
3.9.1 Επισκόπηση του Πρωτοκόλλου

3.9.1.1 Τοπολογία Δικτύου

Το καταλληλότερο σχήμα μεταφοράς δεδομένων στα ασύρματα δίκτυα αισθητήρων είναι το ιεραρχικό με ομάδες πολλαπλών επιπέδων, εξαιτίας του γεγονότος ότι επιτρέπει την εκμετάλλευση όλων των ιδιαίτερων χαρακτηριστικών των δικτύων αισθητήρων [23].

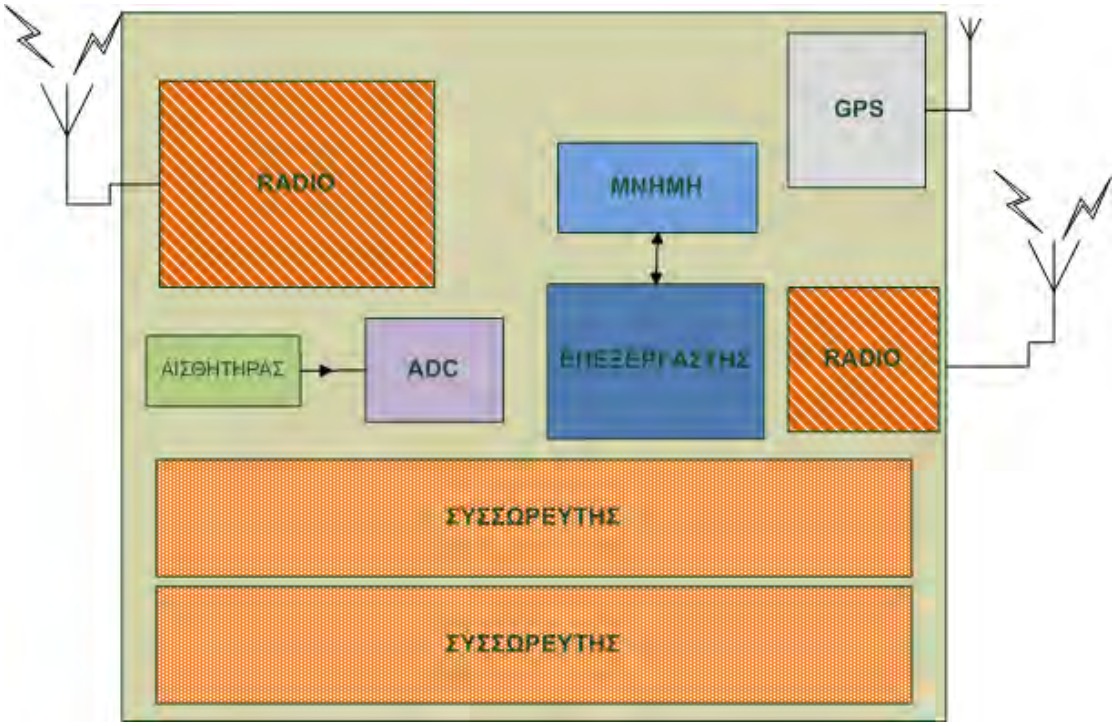
Το δίκτυο αποτελείται από ένα σταθμό βάσης, μακριά από τους κόμβους, στον οποίο συλλέγονται τα δεδομένα από το δίκτυο. Ο σταθμός βάσης δεν έχει κανένα περιορισμό ως προς την ενέργεια που μπορεί να καταναλώσει. Έτσι μπορεί να εκτελεί πολύπλοκους υπολογισμούς (π.χ σύντηξη δεδομένων, αποθήκευση σε κατάλληλη βάση δεδομένων των μετρήσεων των κόμβων, υπολογισμό αλγορίθμων δρομολόγησης κλπ), και να εκπέμπει με τέτοια ισχύ, ώστε να μπορεί να φτάσει μέχρι και τον τελευταίο κόμβο του δικτύου, κάτι που δεν ισχύει αντίστροφα εξαιτίας των ενεργειακών περιορισμών των κόμβων.

Όλοι οι κόμβοι του δικτύου είναι ομογενείς και έχουν την ίδια αρχική ενέργεια, ο κάθε κόμβος (εικόνα 35) περιλαμβάνει στη σύνθεσή του ένα χαμηλής κατανάλωσης ενέργειας δέκτη GPS και χαρακτηρίζεται μοναδικά με μια τύπου MAC διεύθυνση, ενώ υποστηρίζει την μετάβαση σε διάφορες καταστάσεις λειτουργίας – κατανάλωσης ενέργειας (sleep, active, idle, off) σε όλα τα υποσυστήματα του.



Εικόνα 35 - Κόμβος Δικτύου προτεινόμενης Αρχιτεκτονικής

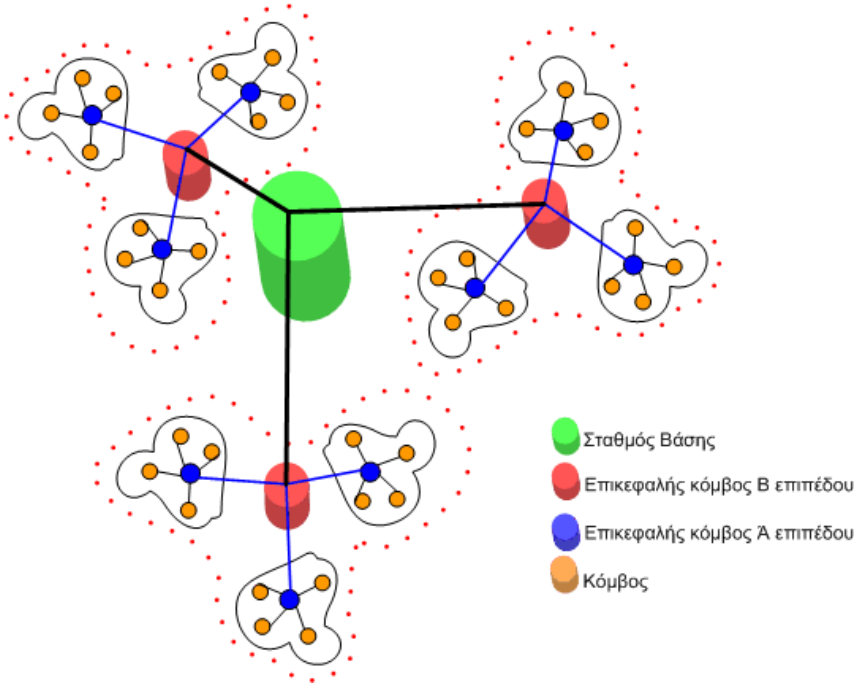
Το προτεινόμενο σχήμα χρησιμοποιεί ιεραρχική δομή με ομάδες πολλαπλών επιπέδων. Θεωρούμε ότι η κόμβοι επικεφαλής των ομάδων, είναι κόμβοι ειδικού τύπου, έτσι ώστε δεν έχουν κάποιο περιορισμό ως προς την υπολογιστική ισχύ, την κατανάλωση ενέργειας ή την ακτίνα εκπομπής (εικόνα 36).



Εικόνα 36 - Σύνθεση Επικεφαλής Κόμβου Δικτύου προτεινόμενης Αρχιτεκτονικής

Έστω το τμήμα ενός ασύρματου δικτύου αισθητήρων που απεικονίζεται στην εικόνα 37. Σε αυτήν απεικονίζονται απλοί κόμβοι που σημειώνονται με πορτοκαλί κύκλο. Κάθε τέτοιος κόμβος ανήκει σε μία ομάδα Α επιπέδου, η οποία έχει επικεφαλής έναν κόμβο, ο οποίος στην εικόνα σημειώνεται με μπλε κύκλο. Ο επικεφαλής κόμβος συγκεντρώνει τα δεδομένα από τους κόμβους που ανήκουν στη ομάδα του και τα προωθεί στον επικεφαλής ομάδας Β επιπέδου, ο οποίος στην εικόνα σημειώνεται με κόκκινο κύκλο. Καθήκον του επικεφαλής κόμβου μιας ομάδας Β επιπέδου είναι να συγκεντρώνει δεδομένα από τους επικεφαλής κόμβους των ομάδων Α επιπέδου να τα επεξεργάζεται και να τα προωθεί ή σε ομάδα μεγαλύτερου επιπέδου, όταν αυτή υπάρχει ή κατευθείαν στον σταθμό βάσης.

Στο παρακάτω σχήμα ο σταθμός βάσης αποτελεί την κορυφή της ιεραρχικής πυραμίδας μας και επιβλέπει την λειτουργία του συνόλου του δικτύου.



Εικόνα 37 - Τμήμα δικτύου προτεινόμενης αρχιτεκτονικής

3.9.1.2 Βασικά Χαρακτηριστικά

Τα βασικά χαρακτηριστικά της προτεινόμενης αρχιτεκτονικής είναι:

- Κάθε κόμβος έχει σαν τελικό αποδέκτη τον επικεφαλής της ομάδας στην οποία ανήκει.
- Απαιτητικές λειτουργίες σε ενέργεια και υπολογιστική ισχύ εκτελεί μόνο ο επικεφαλής κάθε ομάδας, ή ο Σταθμός Βάσης, μειώνοντας με αυτό τον τρόπο τη συνολική κατανάλωση ενέργειας της ομάδας και αυξάνοντας έτσι των συνολικό χρόνο ζωής του δικτύου.
- Ο συγχρονισμός του δικτύου μπορεί να επιτευχθεί σε ομάδες υψηλού επίπεδου και στη συνέχεια να διαδοθεί μέχρι τον τελευταίο κόμβο, επιπλέον η ύπαρξη δέκτη GPS σε όλους τους κόμβους παρέχει επιπλέον δυνατότητες συγχρονισμού χρησιμοποιώντας το εσωτερικό ρολόι του.
- Καθώς υπεύθυνοι για τη δρομολόγηση είναι οι επικεφαλής των ομάδων έχουμε μεγάλη μείωση της πολυπλοκότητας δρομολόγησης αλλά και αύξηση της ταχύτητας απόκρισης του δικτύου.
- Για μεγαλύτερη αντοχή σε σφάλματα είναι δυνατό κάποιος επικεφαλής ομάδας να μπορεί να επικοινωνεί με περισσότερους από έναν επικεφαλής ομάδων μεγαλύτερου επιπέδου από το δικό του, ή ακόμα γειτονικές ομάδες να επικαλύπτουν η μία την άλλη κατά κάποιο ποσοστό.

3.9.2 Αναλυτική περιγραφή του Πρωτοκόλλου

3.9.2.1 Επισκόπηση Λειτουργίας

Το πρωτόκολλο, λειτουργεί σε δύο φάσεις, τη φάση της αρχικοποίησης και τη φάση της κανονικής λειτουργίας. Στη φάση της αρχικοποίησης λαμβάνουν χώρα μια σειρά από ενέργειες, οι οποίες εκτελούνται από τους κόμβους, τους επικεφαλής ομάδων κόμβων, καθώς και από τον σταθμό βάσης. Στις ενέργειες αυτές, συμπεριλαμβάνονται, η δημιουργία του δικτύου ανάμεσα στον σταθμό βάσης και τους επικεφαλής ομάδων κόμβους, στον εντοπισμό με την χρήση του GPS δέκτη της θέσης κάθε στοιχείου του δικτύου, την ομαδοποίηση των κόμβων και τέλος τον υπολογισμό των μονοπατιών δρομολόγησης των δεδομένων από τον κάθε κόμβο προς τον επικεφαλής της ομάδας του.

Στην φάση της κανονικής λειτουργίας πέραν των διαδικασιών δρομολόγησης δεδομένων από τους κόμβους, το πρωτόκολλο περιλαμβάνει μια σειρά από ενέργειες διαχείρισης του δικτύου όπως ο περιοδικά ή μετά από αίτηση κάποιου κόμβου επανυπολογισμός των μονοπατιών δρομολόγησης, η προσθήκη ή διαγραφή απλών ή επικεφαλής ομάδων κόμβων κ.τ.λ.

Έτσι λοιπόν, το πρωτόκολλο βασίζεται στην επικοινωνία, ανάμεσα στα στοιχεία που το αποτελούν, με ανταλλαγή συγκεκριμένων τύπων μηνυμάτων, τα οποία διαφέρουν ανάλογα με τη φάση λειτουργίας στην οποία βρίσκεται το δίκτυο.

Οι τύποι των μηνυμάτων που ανταλλάσσονται είναι:

1) Αρχικοποίησης (Initialization – I): Στο μήνυμα αυτό τοποθετούνται όλες οι πληροφορίες αρχικοποίησης των στοιχείων του δικτύου. Περιλαμβάνει πληροφορίες όπως: γεωγραφικές συντεταγμένες, ισχύς που απομένει στον κόμβο, κατάσταση λειτουργίας του κόμβου. Μήνυμα αρχικοποίησης μπορεί να σταλεί και κατά τη διάρκεια της φάσης κανονικής λειτουργίας, όταν προστίθεται νέος κόμβος στο δίκτυο.

2) Δεδομένων (Data – D): Ένα μήνυμα δεδομένων αποτελείται από δύο τμήματα: από την επικεφαλίδα και το σώμα, στο οποίο τοποθετούνται τα δεδομένα. Η επικεφαλίδα του μηνύματος δεδομένων είναι διαφορετική από αυτή του μηνύματος αρχικοποίησης και περιλαμβάνει τις πληροφορίες όπως: είδος μηνύματος, διεύθυνση αποστολέα, διεύθυνση παραλείπτη για δρομολόγηση, τελικό αποδέκτη.

Τα μηνύματα αυτού του είδους περιλαμβάνουν και ένα δεύτερο τμήμα, το σώμα, στο οποίο περιέχονται τα προς αποστολή δεδομένα. Το μέγεθος τού σώματος του μηνύματος τύπου D είναι μεταβλητό και εξαρτάται από το μέγεθος των προς

αποστολή δεδομένων. Φυσικά για λόγους οικονομίας ενέργειας θα πρέπει τα προς αποστολή δεδομένα να έχουν όσο το δυνατόν μικρότερο μέγεθος.

3) Διαχείρισης Δικτύου (Network Maintenance- N): Τα μηνύματα αυτής της μορφής έχουν ως σκοπό τη διατήρηση της λειτουργίας του δικτύου, των εντοπισμό βλαβών στους κόμβους, τον επανυπολογισμό των μονοπατιών δρομολόγησης και γενικά την εξασφάλιση της αποδοτικής λειτουργίας του. Η αποστολή αυτού του είδους μηνύματος μποπεί να γίνεται είτε περιοδικά (προγραμματισμένα), όταν ο αποστολέας είναι ο σταθμός βάσης ή κάποιος από τους επικεφαλής ομάδων και παραλήπτες οι επικεφαλής ή οι απλοί κόμβοι αντίστοιχα, είτε σε οποιαδήποτε φάση κριθεί απαραίτητο, π.χ. βλάβη στην λειτουργία κάποιου υποσυστήματος ενός κόμβου.

3.9.2.2 Φάση Αρχικοποίησης

Μετά το πέρας της ανάπτυξης τους, όλα τα στοιχεία του δικτύου ξεκινούν τη λειτουργία τους, εκτελούν αυτοέλεγχο των συστημάτων τους για να διαπιστώσουν τυχόν βλάβες σε αυτά, και στη συνέχεια:

(1) με την χρήση του GPS δέκτη τους εντοπίζουν τις ακριβείς γεωγραφικές συντεταγμένες της θέσης τους (Γεωγραφικό Μήκος - x, Γεωγραφικό Πλάτος - y, Ύψος από την επιφάνεια της θάλασσας – z),

(2) υπολογίζουν την ισχύ που απομένει προς κατανάλωση στον συσσωρευτή τους,

(3) δημιουργούν ένα μήνυμα αρχικοποίησης, που περιλαμβάνει όλες τις πληροφορίες που αφορούν στις παραμέτρους αρχικοποίησης τους,

(4) μπαίνουν σε κατάσταση αναμονής περιμένοντας μήνυμα από τον σταθμό βάσης ή από επικεφαλής ομάδων κόμβων,

(5) ο Σταθμός Βάσης (BS), για να υλοποιήσει το δίκτυο κορμού, αποστέλλει μήνυμα προς όλους τους Επικεφαλής Ομάδων (CH), από τους οποίους ζητά να του διαβιβάσουν τις ταυτότητες, την απομένουσα ενέργεια, την κατάσταση λειτουργίας και τις συντεταγμένες της θέσης των κόμβων, που βρίσκονται εντός της ακτίνας εκπομπής τους.

(6) ο κάθε CH στέλνει μήνυμα προς όλους τους κόμβους, που βρίσκονται στην ακτίνα επηροής του, ζητώντας να του στείλουν το μήνυμα αρχικοποίησης το οποίο έχουν δημιουργήσει νωρίτερα.

(7) Με την λήψη του μηνύματος αρχικοποίησης από έναν CH ο κάθε κόμβος ορίζει σαν τελικό αποδέκτη των δεδομένων αυτόν τον CH.

(8) Όταν ο κάθε CH λάβει τα μηνύματα τύπου I από τους κόμβους που βρίσκονται εντός της ακτίνας εκπομπής τους, προετοιμάζει ένα μήνυμα τύπου D στο σώμα του οποίου περιλαμβάνει για κάθε κόμβο τον κωδικό του, τις συντεταγμένες του, την απομένουσα σε αυτό ενέργεια και την κατάσταση λειτουργίας του.

(9) Όταν ο BS λάβει το σύνολο των μηνυμάτων τύπου D από τους CH, αποθηκεύει τα δεδομένα σε κατάλληλη βάση γεωγραφικών δεδομένων, και είναι έτοιμος να ξεκινήσει, αρχικά τη διαδικασία υπολογισμού των ομάδων και της ανάθεσης συγκεκριμένων κόμβων σε κάθε CH και στη συνέχεια τον υπολογισμό των βέλτιστων διαδρομών εντός της κάθε ομάδας.

- Δημιουργία Ομάδων: Η δημιουργία ομάδων είναι μια υπηρεσία η οποία παρέχεται από το λογισμικό ενδιάμεσου επιπέδου, που εκτελείται από τον *BS*. Το βασικό κριτήριο με το οποίο δημιουργείται μία ομάδα κόμβων είναι η απόσταση κάθε κόμβου από τους CH, με τελική επιλογή τον CH με τη μμικρότερη απόσταση.
- Υπολογισμός Ελάχιστων Διαδρομών: Ο υπολογισμός της βέλτιστης διαδρομής ανάμεσα σε δύο σημεία στο χώρο είναι ένα πρόβλημα που έχει μελετηθεί επισταμένα [21]. Οι περισσότερες προσεγγίσεις βασίζονται στην μεταφορά του προβλήματος από την εύρεση της βέλτιστης διαδρομής στον χώρο στην εύρεση αυτής σε έναν γράφο που απεικονίζει τη γεωγραφία του δικτύου στο επίπεδο.

3.9.2.3 Φάση Κανονικής Λειτουργίας

Στην φάση της κανονικής λειτουργίας πέραν της δρομολόγησης δεδομένων από τους κόμβους, με αποστολή μηνυμάτων τύπου D, το πρωτόκολλο περιλαμβάνει μια σειρά από διαδικασίες διαχείρισης του δικτύου, όπως ο περιοδικά ή μετά από αίτηση κάποιου κόμβου επανυπολογισμός των μονοπατιών δρομολόγησης, η αναφορά από κάποιον κόμβο κάποιας δυσλειτουργίας σε κάποιο από τα υποσυστήματα του, ή η μεταβολή στην διαθέσιμη σε κάποιον κόμβο ενέργεια. Αυτού του είδους η κίνηση, διεκπεραιώνεται με ανταλλαγή μηνυμάτων τύπου N, τα οποία μπορούν να έχουν σαν πηγή τόσο τον BS όσο και κάποιον από τους CH ή Sn. Τέλος μια τελευταία κατηγορία διαδικασιών, οι οποίες μπορούν να λάβουν χώρα είναι αυτές οι οποίες προκαλούν μεταβολή στη σύνθεση του δικτύου, πχ προσθήκη ή διαγραφή απλών ή επικεφαλής ομάδων κόμβων. Στην περίπτωση αυτή για καθένα από τα νέα στοιχεία του δικτύου που προστίθεται, ξεκινά η διαδικασία αρχικοποίησης, όπως αυτή περιγράφηκε προηγουμένως.

3.9.3 Εκτίμηση του Πρωτοκόλλου

3.9.3.1 Ικανοποίηση των Επιθυμητών Χαρακτηριστικών των Πρωτοκόλλων Δρομολόγησης

Το πρωτόκολλο που παρουσιάστηκε πιο πάνω υλοποιεί τα περισσότερα από τα επιθυμητά χαρακτηριστικά των πρωτοκόλλων δρομολόγησης.

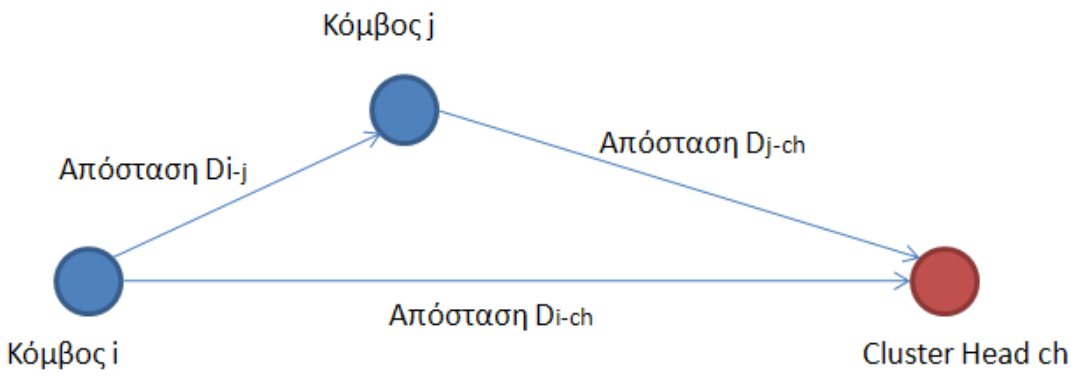
Ειδικότερα το πρωτόκολλο παρουσιάζει:

1. Χαμηλή υπολογιστική πολυπλοκότητα, τουλάχιστον στο τμήμα του το οποίο εκτελείται στους κόμβους.
2. Μικρές απαιτήσεις σε χώρο αποθήκευσης.
3. Μικρές απαιτήσεις ανταλλαγής μηνυμάτων: Για την λειτουργία του, το πρωτόκολλο δεν απαιτεί, πέραν της αρχικής φάσεως, την ανταλλαγή άλλων μηνυμάτων.
4. Αποφυγή Κυκλικών Διαδρομών: Το πρωτόκολλο, δεν επιτρέπει τις κυκλικές διαδρομές.

3.9.3.2 Ως προς την Κατανάλωση Ενέργειας

Όπως έχουμε δει σε προηγούμενα κεφάλαια, η ενέργεια, η οποία καταναλώνεται από έναν κόμβο για την εκτέλεση της αποστολής του είναι από τους σημαντικότερους σχεδιαστικούς παράγοντες, που επηρεάζει σημαντικά τον χρόνο ζωής και λειτουργίας τόσο του κόμβου όσο και του δικτύου συνολικά. Το συστατικό που απομυζά την περισσότερη ενέργεια σε έναν κόμβο είναι, όπως έχουμε πει, το ασύρματο σύστημα. Για την ελαχιστοποίηση αυτής της ενέργειας πρέπει κάθε κόμβος να επιλέγει σωστά το γείτονα που θα λειτουργήσει σαν αναμεταδότης των δεδομένων του προς τον επικεφαλής (cluster head) της ομάδας στην οποία ανήκει.

Πολύ εύκολα θα μπορούσε να επιλεγεί σαν γείτονας ο πλησιέστερος κόμβος ο οποίος βρίσκεται προς την κατεύθυνση του Cluster Head, ελαχιστοποιώντας έτσι και την ισχύ εκπομπής του ασύρματου συστήματος.



Εικόνα 38 - Επιλογή κοντινότερου κόμβου

Τα βήματα για την επιλογή του κόμβου j σαν αναμεταδότη είναι:

1. Εύρεση των κόμβων που ισχύει $D_{i-j} < D_{i-ch}$ και $D_{j-ch} < D_{i-ch}$
2. Επιλογή του κόμβου j με την ελάχιστη απόσταση D_{i-j}

Ο πιο πάνω αλγόριθμος μας δίνει τον κοντινότερο δυνατό κόμβο προς την κατεύθυνση του Cluster Head, αλλά όπως θα δούμε και στις προσομοιώσεις που κάναμε δεν αποτελεί και τη βέλτιστη ενεργειακά λύση. Υπάρχει η πιθανότητα το σύνολο της ενέργειας που απαιτείται για την αποστολή ενός πακέτου δεδομένων από τον κόμβο i στον ch μέσω του j, να είναι μεγαλύτερη από αυτή που απαιτείται για την απευθείας μετάδοση στον ch. Για το λόγο αυτό λάβαμε υπόψη μας άλλα κριτήρια για την επιλογή του κόμβου αναμεταδότη που έχουν σχέση με την καταναλισκόμενη ενέργεια. Θεωρούμε τα εξής:

1. $E_{tx(A-B)} = E_{radio} + E_{txD(A-B)} + E_{cpu}$

Όπου $E_{tx(A-B)}$ η συνολική απαιτούμενη ενέργεια για τη μεταφορά πακέτου δεδομένων από το A στο B, E_{radio} η ενέργεια για τη λειτουργία των ηλεκτρονικών του ασύρματου συστήματος, $E_{txD(A-B)}$ η ενέργεια για τη μετάδοση σε απόσταση D(A-B) και E_{cpu} η ενέργεια που καταναλώνει η επεξεργαστική μονάδα.

2. E_{rx} η ενέργεια που απαιτείται για τη λήψη του πακέτου από το δέκτη ενός κόμβου και ισχύει: $E_{rx} = E_{radio} + E_{cpu}$

Στις πιο πάνω περιπτώσεις η ενέργεια ισούται με ***Ισχύς x Χρόνος (P x t)***. Σε κάθε περίπτωση η απαιτούμενη ισχύς είναι γνωστή, ενώ ο χρόνος εξαρτάται από το μέγεθος του πακέτου και το ρυθμό μετάδοσης δεδομένων.

3. t_{tx} = pkt_size/bandwidth

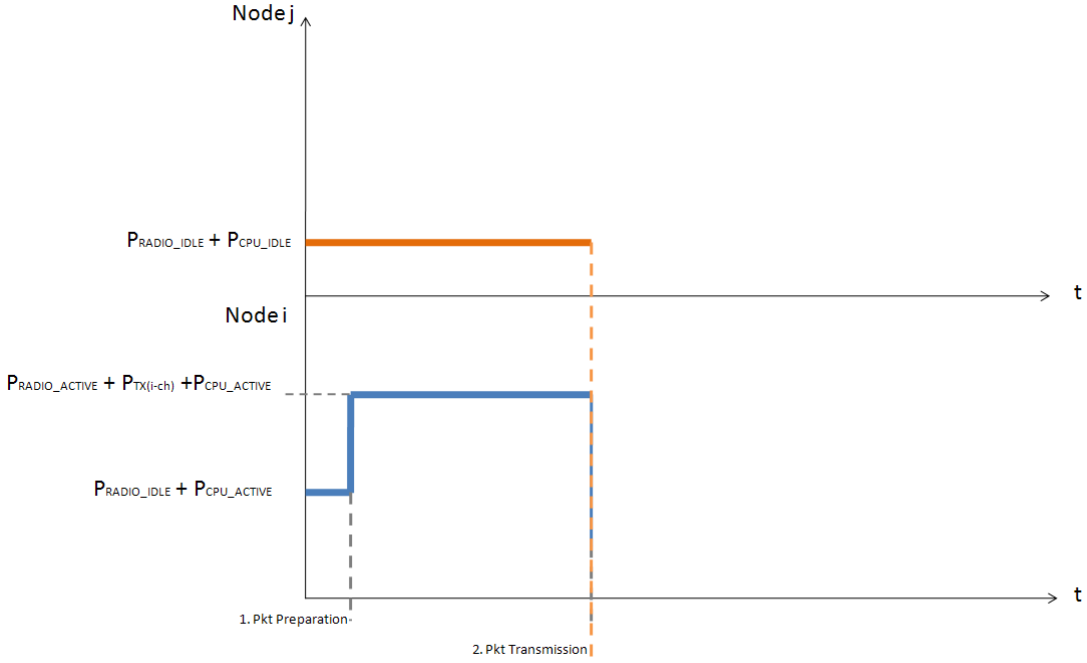
οπότε στο πιο πάνω παράδειγμα εξετάζουμε δύο περιπτώσεις μετάδοσης:

- Απευθείας μετάδοση πακέτου από τον κόμβο i στον Cluster Head. Η ενέργεια που πρέπει να καταναλωθεί σε αυτή την περίπτωση είναι: $E_{tx(i\text{-}ch)} = E_{radio} + E_{txD(i\text{-}ch)} + E_{cpu}$
- Μετάδοση του πακέτου μέσω του κόμβου j. Σε αυτή την περίπτωση καταναλώνεται ενέργεια και τον κόμβο i και στον κόμβο j και ισχύουν τα εξής:
    - $E_i = E_{tx(i\text{-}j)} = E_{radio} + E_{txD(i\text{-}j)} + E_{cpu}$
    - $E_j = E_{rx} + E_{tx(j\text{-}ch)} = (E_{radio} + E_{cpu}) + (E_{radio} + E_{txD(j\text{-}ch)} + E_{cpu})$
    - $E_{tx(i\text{-}j\text{-}ch)} = E_i + E_j$

Για να επιλεγεί, λοιπόν, ο κόμβος j σαν αναμεταδότης των δεδομένων του i θα πρέπει να ισχύει: $E_{tx(i\text{-}j\text{-}ch)} < E_{tx(i\text{-}ch)}$

Πιο αναλυτικά αναπαριστώντας την απαιτούμενη ισχύ του κάθε κόμβου στο χρόνο, ισχύουν τα εξής:

Απευθείας μετάδοση των δεδομένων του κόμβου i στον Cluster Head



$t_{preparation}$ = ο χρόνος που χρειάζεται για την προετοιμασία του πακέτου προς αποστολή

t_{tx} = ο χρόνος που απαιτείται για την αποστολή του πακέτου και ισούται με pkt_size/bandwidth

Κατά την προετοιμασία του πακέτου απαιτείται ισχύς P_{CPU_ACTIVE}

Για την αποστολή του πακέτου απαιτείται ισχύς $P_{RADIO\_ACTIVE} + P_{TX(i\text{-}ch)} + P_{CPU\_ACTIVE}$

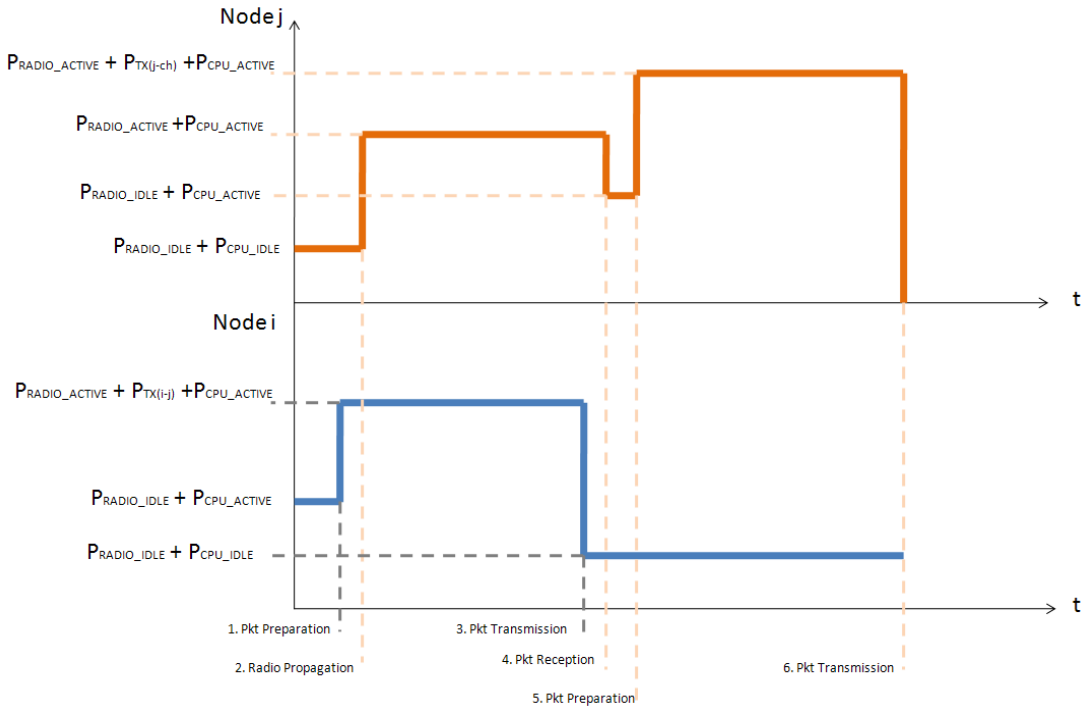
Το σύνολο της καταναλισκόμενης ενέργειας για το σύστημα των κόμβων i και j είναι

$E_i = (P_{CPU\_ACTIVE} \times t_{preparation}) + (P_{RADIO\_ACTIVE} + P_{TX(i\text{-}ch)} + P_{CPU\_ACTIVE}) \times t_{tx}$

$E_j = (P_{RADIO\_IDLE} + P_{CPU\_IDLE}) \times (t_{preparation} + t_{tx})$

$E_{i\text{-}ch} = E_i + E_j$

Μετάδοση των δεδομένων του κόμβου i μέσω του κόμβου j



Σε αυτή την περίπτωση ισχύουν, εκτός από τα πιο πάνω και τα εξής:

$t_{propagation}$ = ο χρόνος διάδοσης του ραδιοκύματος σε απόσταση, από τον κόμβο i στον κόμβο j

P_{CPU_IDLE} = η ισχύς που απαιτείται από τον επεξεργαστή σε κατάσταση αδράνειας

P_{RADIO_IDLE} = η ισχύς που απαιτείται από το ασύρματο υποσύστημα σε κατάσταση αδράνειας

Για την αποστολή του πακέτου από τον i στον j απαιτείται ισχύς $P_{RADIO\_ACTIVE} + P_{TX(i-j)} + P_{CPU\_ACTIVE}$

Για την αποστολή του πακέτου από τον j στον ch απαιτείται ισχύς $P_{RADIO\_ACTIVE} + P_{TX(j-ch)} + P_{CPU\_ACTIVE}$

Για τη λήψη του πακέτου απαιτείται ισχύς $P_{RADIO\_ACTIVE} + P_{CPU\_ACTIVE}$

Το σύνολο της καταναλισκόμενης ενέργειας για το σύστημα των κόμβων i και j είναι

$E_i = ((P_{RADIO\_IDLE} + P_{CPU\_ACTIVE}) \times t_{preparation}) + ((P_{RADIO\_ACTIVE} + P_{TX(i-j)} + P_{CPU\_ACTIVE}) \times t_{tx}) + ((P_{RADIO\_IDLE} + P_{CPU\_IDLE}) \times (t_{propagation} + t_{preparation} + t_{tx}))$

$E_j = ((P_{RADIO\_IDLE} + P_{CPU\_IDLE}) \times (t_{propagation} + t_{preparation})) + ((P_{RADIO\_ACTIVE} + P_{CPU\_ACTIVE}) \times t_{tx}) + ((P_{RADIO\_IDLE} + P_{CPU\_ACTIVE}) \times t_{preparation}) + ((P_{RADIO\_ACTIVE} + P_{TX(j-ch)} + P_{CPU\_ACTIVE}) \times t_{tx})$

Οπότε και $E_{i\text{-}j\text{-}ch} = E_i + E_j$

Όπως διατυπώσαμε και πιο πάνω, για υπάρχει συμφέρον αναμετάδοσης μέσω του κόμβου j θα πρέπει να ισχύει: $E_{i\text{-}j\text{-}ch} < E_{i\text{-}ch}$

Όπως θα δούμε και στις προσομοιώσεις, για να υπολογίσουμε την ισχύ που απαιτείται για τη μετάδοση σήματος σε απόσταση *d* θα πρέπει καθορίσουμε ένα μοντέλο διάδοσης και σύμφωνα με αυτό να κάνουμε τους υπολογισμούς μας. Στις δικές μας προσομοιώσεις χρησιμοποιήσαμε το free space model.

$$P_r = P_t \cdot G_r \cdot G_t \cdot \frac{\lambda^2}{(4\pi d)^2}$$

Όπου *Pr* η λαμβανόμενη ισχύς, *Pt* η εκπεμπόμενη ισχύς, *Gr* το κέρδος της κεραίας του δέκτη, *Gt* το κέρδος της κεραίας του πομπού, *λ* το μήκος κύματος του εκπεμπόμενου σήματος και *d* η απόσταση των κόμβων.

Στην ανισότητα $E_{i\text{-}j\text{-}ch} < E_{i\text{-}ch}$ όλα τα μεγέθη είναι γνωστά και έχουν σχέση με το hardware του κάθε κόμβου (ηλεκτρονικά, ασύρματο σύστημα, κεραία, CPU) και τελικά παίρνει τη μορφή: $\mathrm{E} + d_{i-j}^2 + d_{j-ch}^2 < d_{i-ch}^2$ με το *E* σταθερά που εξαρτάται από τα πιο πάνω συστήματα.

Κεφάλαιο 4 – ΤΟ ΕΡΓΑΛΕΙΟ ΠΡΟΣΟΜΟΙΩΣΗΣ J-SIM

4.1 Εισαγωγή

Η ανάπτυξη σωστών εργαλείων προσομοίωσης ήταν ένα βήμα κλειδί για την περαιτέρω έρευνα και ανάπτυξη των συστημάτων. Γενικά, η προσομοίωση μπορεί να παρέχει ένα τρόπο μελέτης των εναλλακτικών τρόπων σχεδίασης ενός συστήματος, σε ένα ελεγχόμενο από όλες τις απόψεις περιβάλλον. Έτσι ελέγχεται τι επιφέρουν οι διάφορες ρυθμίσεις σε ένα σύστημα που είναι δύσκολο να κατασκευαστεί και να ρυθμιστεί επανειλημμένα. Επίσης παρατηρούνται οι διάφορες αλληλεπιδράσεις του συστήματος για κάθε σύνολο ρυθμίσεων που εφαρμόζεται, κάτι που μπορεί να είναι δύσκολο να ανιχνευθεί σε ένα πραγματικό σύστημα.

Τέτοια εργαλεία προσομοίωσης είναι πολύ χρήσιμα και στα δίκτυα ασύρματων αισθητήρων, τα οποία λόγω της φύσης κατασκευής και εγκατάστασης τους είναι δύσκολο και οικονομικά ασύμφορο να μελετηθούν σε πραγματική κλίμακα, ώστε να βρεθούν οι κατάλληλες ρυθμίσεις που θα τα κάνουν ικανά να φέρουν σε πέρας την αποστολή τους. Τέτοια εργαλεία θα πρέπει να μπορούν να μελετούν συμπεριφορές της εφαρμογής μέχρι το χαμηλότερο επίπεδο των συστατικών ενός κόμβου, καθώς επίσης θα πρέπει να μπορούν να χειριστούν ένα μεγάλο πλήθος κόμβων και τις αλληλεπιδράσεις μεταξύ τους αλλά και με το περιβάλλον.

Ένας εξομοιωτής θα πρέπει να διαθέτει τα εξής χαρακτηριστικά:

α) Κλιμακωσιμότητα: Θα πρέπει να μπορεί να προσομοιώσει και να χειριστεί ένα μεγάλο πλήθος αισθητήρων (εκατοντάδες ή και χιλιάδες).

β) Πληρότητα: Θα πρέπει να καλύπτει όσο το δυνατόν πιο πολλά συστήματα, και να αντιλαμβάνεται και να αναπαριστά τις συμπεριφορές τους σε όλα τα επίπεδα λεπτομέρειας (δηλ. από το επίπεδο εφαρμογής και το επίπεδο δρομολόγησης ως το τελευταίο συστατικό του αισθητήρα, και αυτό να μπορεί να εφαρμοστεί για κάθε εφαρμογή).

γ) Αξιοπιστία: Η συμπεριφορά του δικτύου θα πρέπει να γίνεται αντιληπτή και να αναπαρίσταται σε μεγάλη λεπτομέρεια και με ακρίβεια. Επίσης θα πρέπει να αποκαλύπτει και μη επιθυμητές συμπεριφορές του δικτύου και όχι μόνο αυτές που υποπτεύονται οι κατασκευαστές του.

δ) Γεφύρωση: Θα πρέπει να γεφυρώνει το χάσμα μεταξύ της σχεδίασης των αλγορίθμων και της υλοποίησης τους, ώστε οι κατασκευαστές κώδικα να μπορούν να τον δοκιμάσουν πριν τον εφαρμόσουν σε πραγματικά δίκτυα. Πολύ συχνά ένας

αλγόριθμος φαίνεται πολύ καλός, αλλά η υλοποίηση του μπορεί να απέχει πολύ από το θεωρητικό σχέδιο.

Το εργαλείο που χρησιμοποιήθηκε στην παρούσα εργασία για την προσομοίωση πρωτοκόλλων δρομολόγησης είναι το J-SIM το οποίο και παρουσιάζεται πιο κάτω.

4.2 J-SIM

Το J-sim [27], είναι ένα ανοιχτού κώδικα περιβάλλον προσομοίωσης βασισμένο σε συστατικά το οποίο έχει αναπτυχθεί ολοκληρωτικά σε Java. Έχει χτιστεί επάνω σε μια αυτόνομη αρχιτεκτονική συστατικών (Autonomous Component Architecture ACA) και το εκτεταμένο πλαίσιο εργασίας του διαδικτύου (extensible internetworking framework INET). Οι βασικές οντότητες του ACA είναι τα συστατικά (components), τα οποία επικοινωνούν το ένα με το άλλο μέσω αποστολής/λήψης δεδομένων στις θύρες τους (ports). Ο τρόπος συμπεριφοράς των συστατικών καθορίζεται στον σχεδιασμό του συστήματος με συμβόλαια (contracts), αλλά η σύνδεση μεταξύ τους λαμβάνει χώρα κατά τη σύνθεση του συστήματος. Με τον διαχωρισμό αυτό (του δεσίματος των συμβολαίων και του δεσίματος των συστατικών), το J-sim παρέχει μια αρχιτεκτονική χαλαρά συνδεδεμένων συστατικών (loosely coupled component architecture) δηλαδή ένα συστατικό μπορεί να σχεδιαστεί, να υλοποιηθεί, και να δοκιμαστεί αυτόνομα. Με αυτόν τον τρόπο το ACA επιτρέπει σε νέα συστατικά να εισάγονται στο J-sim με τρόπο άμεσης τοποθέτησης και λειτουργίας (plug-n-play fashion). Στην κορυφή του ACA βρίσκεται το INET.

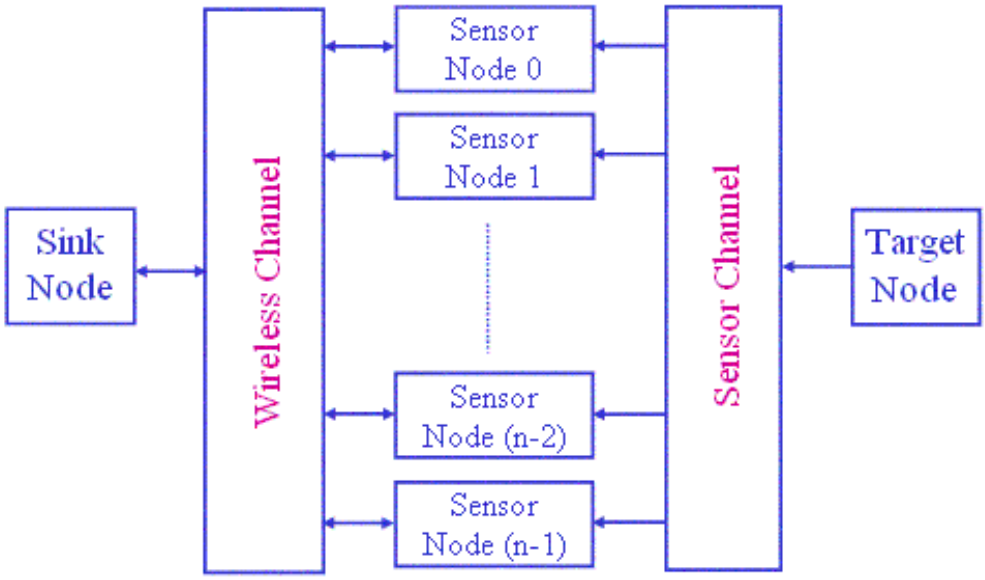
Η γλώσσα των scripts προσομοίωσης είναι η Tcl ή πιο συγκεκριμένα η Jacl (Java/Tcl) μέσω της οποίας μπορούμε να προσπελάσουμε και να χειριστούμε τα συστατικά του J-SIM προκειμένου να δημιουργήσουμε σενάρια προσομοιώσεων.

Το J-SIM προσφέρει ένα αντικειμενοστραφή ορισμό για :

α) Κόμβους στόχους (target nodes), αισθητήριους κόμβους (sensor nodes) και συγκεντρωτές κόμβους (sink nodes). Οι κόμβοι στόχοι παράγουν τα διάφορα φυσικά συμβάντα του περιβάλλοντος, τα οποία αντιλαμβάνονται οι αισθητήριοι κόμβοι και κατόπιν επεξεργασίας τα στέλνουν ως δεδομένα προς τους συγκεντρωτές κόμβους οι οποίοι τα συγκεντρώνουν για να τα παραδώσουν στον τελικό χρήστη,

β) αισθητήρια κανάλια και κανάλια ασύρματης επικοινωνίας

γ) φυσικά μέσα όπως είναι τα κανάλια απόδοσης των σεισμικών κυμάτων, μοντέλο κινητικότητας και μοντέλο ενέργειας (και όσον αφορά στην παραγωγή της αλλά και στην κατανάλωση της).



Εικόνα 39 - Μοντέλο ενός τυπικού συστήματος προσομοίωσης δικτύου αισθητήρων

Στο J-sim έχουν ενσωματωθεί επίσης κάποιες κλάσεις και μηχανισμοί που έχουν τη δυνατότητα να ενσωματώσουν το περιβάλλον της προσομοίωσης με πραγματικές συσκευές, προκειμένου να διευκολύνουν την αξιολόγηση της απόδοσής τους σε πραγματικές αλλά ελεγχόμενες συνθήκες. Το βασικό πρόβλημα σε μια τέτοια προσομοίωση είναι ο συγχρονισμός του χρόνου μεταξύ του πραγματικού συστήματος και του εξομοιωτή καθώς αυτά τα δύο αλληλεπιδρούν ανταλλάσσοντας πακέτα δεδομένων. Στο [26] υπάρχει ένας τέτοιος συνδυασμός ενός μελλοντικού συστήματος μάχης.

4.2.1. Αρχιτεκτονική

Όπως αναφέρθηκε και παραπάνω υπάρχουν 3 ειδών κόμβοι στον εξομοιωτή J-sim. Οι κόμβοι στόχοι που παράγουν τα ερεθίσματα, οι αισθητήριοι κόμβοι που τα ανιχνεύουν και παράγουν κάποια δεδομένα. Τα δεδομένα αυτά τα στέλνουν μέσω ενός ασύρματου καναλιού προς τους κόμβους συγκεντρωτές. Στην Εικόνα 39 απεικονίζεται μια υψηλού επιπέδου όψη της εν λόγω αρχιτεκτονικής. Πρέπει να σημειωθεί ότι μεταξύ των κόμβων στόχων και των αισθητήριων κόμβων παρεμβάλλεται ένα εντελώς διαφορετικό κανάλι από το προαναφερθέν ασύρματο, που ονομάζεται αισθητήριο κανάλι και χρησιμοποιείται από τον εξομοιωτή για την μεταφορά των ερεθισμάτων προς τους αισθητήριους κόμβους. Συνεπώς υπάρχουν δύο μοντέλα διάδοσης (propagation models) α) το αισθητήριο και β) το ασύρματο.

Ένας αισθητήριος κόμβος αποτελείται από :

α) *Στοίβα πρωτοκόλλου αισθήσεως*, που του επιτρέπει να ανιχνεύει τα ερεθίσματα που παράγονται από τους κόμβους στόχους στο αισθητήριο κανάλι

β) *Στοίβα πρωτοκόλλου επικοινωνίας*, που χρειάζεται για την αποστολή αναφορών προς άλλους ασύρματους κόμβους και τελικά προς τον κόμβο συγκεντρωτή μέσω του ασύρματου καναλιού.

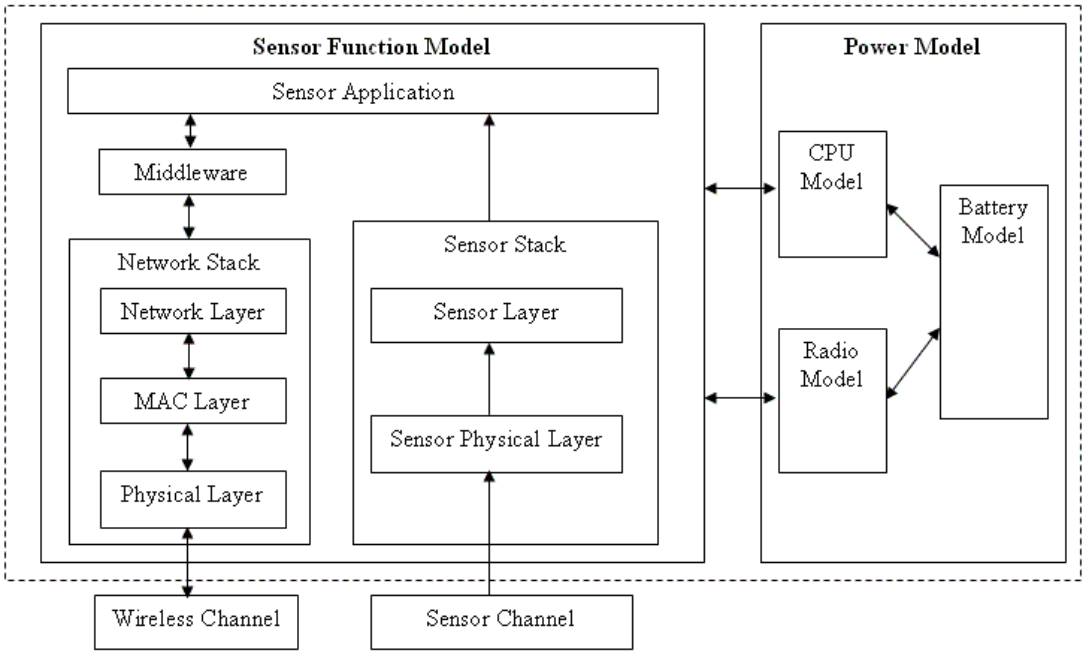
γ) *Μοντέλο ενέργειας* που περιλαμβάνει τα συστατικά που παράγουν την ενέργεια (π.χ. μπαταρία) και αυτά τα οποία την καταναλώνουν (π.χ. ασύρματος ή επεξεργαστής).

δ) *Μοντέλο κινητικότητας* προκειμένου να μπορεί να προσομοιώσει κινούμενους κόμβους.

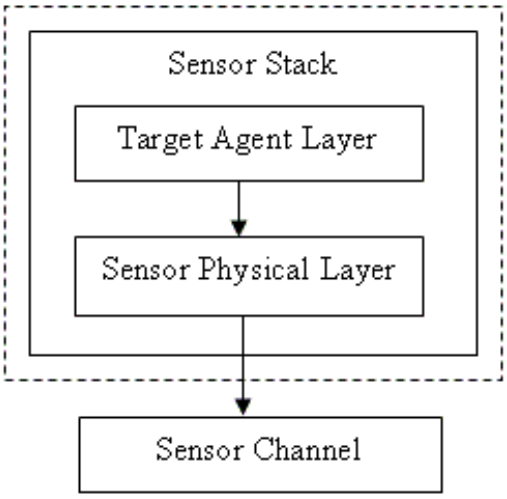
Ένας κόμβος στόχος αποτελείται μόνο από το (α), ενώ ένας κόμβος συγκετρωτής μόνο από το (β). Στις εικόνες 40, 41, 42 απεικονίζονται όσα αναφέρθηκαν παραπάνω.

Ένας κόμβος στόχος εκπέμπει ερεθίσματα ανά τυχαία διαστήματα προσομοιώνοντας κάποιο φυσικό φαινόμενο. Αυτού του είδους οι κόμβοι μπορούν μόνο να στείλουν αλλά όχι να λάβουν δεδομένα πάνω από το αισθητήριο κανάλι. Τα ερεθίσματα αυτά μπορεί να υποστούν εξασθένιση κατά τη διάδοσή τους μέσα από το αισθητήριο κανάλι. Ο αισθητήριος κόμβος λαμβάνει τα ερεθίσματα αυτά των οποίων η ένταση είναι πάνω από ένα προκαθορισμένο όριο. Ο υπολογισμός της έντασης του λαμβανόμενου σήματος γίνεται από το μοντέλο διάδοσης το οποίο χρησιμοποιείται στον εξομοιωτή.

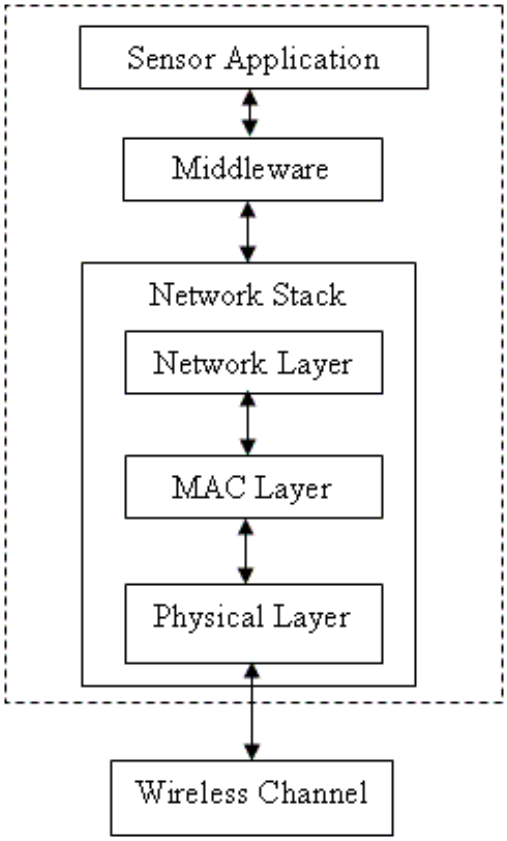
Ο αισθητήριος κόμβος, αφού λάβει τα ερεθίσματα, θα στείλει τα δεδομένα προς έναν ή περισσότερους συγκεντρωτές κόμβους. Μέσα στον κόμβο ο συντονισμός μεταξύ του αισθητήριου και του ασύρματου καναλιού γίνεται από τον επίπεδο της εφαρμογής και το επίπεδο μεταφοράς. Για παράδειγμα, ανάλογα με την εκτελούμενη εφαρμογή, ο κόμβος μπορεί να στείλει τα δεδομένα άμεσα μόλις τα αντιληφθεί ή να τα επεξεργαστεί και μετά να τα εκπέμψει. Στο επίπεδο της εφαρμογής μπορεί να υλοποιηθεί οποιοσδήποτε τέτοιος μηχανισμός.



Εικόνα 40 - Sensor node architecture



Εικόνα 41 - Target node architecture



Εικόνα 42 - Sink node architecture

Ο συγκεντρωτής κόμβος μπορεί να μην είναι άμεσος γείτονας του αισθητήριου κόμβου. Συνεπώς η επικοινωνία και μετάδοση των δεδομένων θα γίνει μέσω πολλαπλών αλμάτων (multihop) και θα παρεμβληθούν άλλοι αισθητήριοι κόμβοι οι οποίοι λειτουργούν ως αναμεταδότες. Αυτό εξηγεί το γεγονός γιατί ένας αισθητήριος κόμβος πρέπει να μπορεί να λαμβάνει αλλά και να μεταδίδει πακέτα στο ασύρματο κανάλι. Είναι πιθανόν κάποιοι κόμβοι να αποτύχουν ή να τεθούν εκτός λειτουργίας λόγω εξάντλησης της ενέργειας τους. Τότε η τοπολογία του δικτύου αλλάζει δυναμικά και θα πρέπει το πρωτόκολλο δρομολόγησης να μπορεί να αντεπεξέλθει σε αυτήν την αλλαγή.

Ο κόμβος συγκεντρωτής ή οποιοσδήποτε άλλος κόμβος στο δίκτυο θα λάβει τα δεδομένα που στάλθηκαν από κάποιον άλλο κόμβο, αν η ισχύς λήψης τους είναι πάνω από ένα προκαθορισμένο όριο. Ο υπολογισμός της ισχύος του λαμβανόμενου σήματος γίνεται από το χρησιμοποιούμενο μοντέλο διάδοσης του ασύρματου καναλιού. Η τελευταία έκδοση του J-sim χρησιμοποιεί 3 διαφορετικά μοντέλα διάδοσης: free space, 2-ray ground και irregular terrain. Ανάλογα με το περιεχόμενο των δεδομένων, ο συγκεντρωτής κόμβος μπορεί να στείλει τα δεδομένα προς το τελικό χρήστη ή να δώσει κάποιες νέες εντολές ή ερωτήματα προς τους

αισθητήριους κόμβους. Αυτός είναι και ο λόγος που πρέπει να μπορεί να λαμβάνει αλλά και να αποστέλλει δεδομένα.

Το μοντέλο ενέργειας περιλαμβάνει συστατικά παραγωγής (π.χ. μπαταρία) και κατανάλωσης ενέργειας (π.χ. επεξεργαστής και ασύρματο σύστημα). Το μοντέλο λειτουργιών του αισθητήρα (δηλ. ο συνδυασμός της στοίβας πρωτοκόλλου αισθήσεως, της στοίβας πρωτοκόλλου δικτύου, του επιπέδου εφαρμογής και του επιπέδου μεταφοράς) υπόκειται στο μοντέλο ενέργειας. Για παράδειγμα, η ενέργεια που καταναλώνεται για το χειρισμό ενός ληφθέντος πακέτου δεδομένων υπαγορεύεται από το μοντέλο της CPU, ενώ η ενέργεια που καταναλώνεται κατά την αποστολή ή λήψη πακέτων δεδομένων υπαγορεύεται από το μοντέλο του ασυρμάτου. Στο J-sim κάθε ένα από τα προηγούμενα 2 μοντέλα μπορεί να βρίσκεται σε διαφορετική κατάσταση λειτουργίας κάτι που επηρεάζει άμεσα την καταναλισκόμενη ενέργεια. Για παράδειγμα το μοντέλο του ασυρμάτου μπορεί να είναι σε μια από τις επόμενες καταστάσεις λειτουργίας : αδρανής, κατάσταση ύπνου (sleep), εκτός λειτουργίας (off), εκπομπή ή λήψη. Η κατάσταση λειτουργίας του μοντέλου (ασυρμάτου ή επεξεργαστή) αναφέρεται στο μοντέλο λειτουργιών του αισθητήρα (sensor function model) και το τελευταίο έχει την ικανότητα να αλλάζει την κατάσταση λειτουργίας του επεξεργαστή ή του ασυρμάτου.

4.2.2 Πλεονεκτήματα - Μειονεκτήματα

Το J-SIM διαθέτει σημαντικά πλεονεκτήματα σαν εργαλείο προσομοίωσης και ειδικότερα σε ό,τι αφορά τα ασύρματα δίκτυα αισθητήρων. Συγκεκριμένα:

1. Το γεγονός ότι είναι γραμμένο εξ’ ολοκλήρου σε JAVA το κάνει συμβατό με όλα ουσιαστικά τα λειτουργικά συστήματα.

2. Έχει μια ιδιαίτερα ανοιχτή αρχιτεκτονική που κάνει δυνατή τη συγγραφή συστατικών σε όλα τα επίπεδα (φυσικό επίπεδο, MAC, application, wireless, antenna κλπ)

3. Ειδικά για δίκτυα αισθητήρων, μπορεί να ενσωματώσει πολύ καλά την ενεργειακή συμπεριφορά όλων των συστατικών (CPU, wireless)

4. Κάνει καλή χρήση της μνήμης οπότε και μπορούν να προσομοιωθούν δίκτυα με εκατοντάδες κόμβους [26].

Στα μειονεκτήματα του πακέτου θα περιλαμβάναμε τα εξής:

1. Λόγω του ότι είναι γραμμένο σε JAVA, η απόδοσή του από πλευράς ταχύτητας δεν είναι ιδιαίτερα καλή. Μάλιστα παρατηρήσαμε και σημαντική διαφορά στην ταχύτητα εκτέλεσης σε λειτουργικό σύστημα Windows σε

σχέση με το Linux. Τρέχοντας το ίδιο script στα δύο λειτουργικά συστήματα, η εκτέλεση στα Windows ήταν τουλάχιστο 3 έως 4 φορές πιο αργή.

2. Η τεκμηρίωση δεν είναι ιδιαίτερα λεπτομερής με αποτέλεσμα να απαιτείται ιδιαίτερη προσπάθεια προκειμένου να γραφτεί κάποιο καινούργιο συστατικό (απαιτείται ανάγνωση του κώδικα των υπαρχόντων συστατικών για την κατανόηση των λειτουργιών τους και του τρόπου συνεργασίας τους).

4.3 Ανάπτυξη J-SIM components

Για τις ανάγκες της παρούσας εργασίας αναπτύχθηκαν components που υλοποιούν κάποια πρωτόκολλα δρομολόγησης. Βασιζόμενοι στην αρχιτεκτονική του J-SIM και σε κάποια από τα ήδη διαθέσιμα components [43] αναπτύχθηκαν τα εξής:

1. Πρωτόκολλο Multi-Hop

    a. Sink application: το επίπεδο εφαρμογής για το σταθμό βάσης. Υλοποιούνται ουσιαστικά όσα αναφέρθηκαν και πιο πάνω για την εφαρμογή One-Hop.

    b. Multi-Hop packet: περιγραφή του πακέτου δεδομένων όμοιο με το One-Hop packet.

    c. Neighbor Query Contract: το συμβόλαιο με το οποίο η εφαρμογή ενός κόμβου αισθητήρα, ρωτά για τον γειτονικό κόμβο στον οποίο θα μπορούσε να στείλει τα δεδομένα, αντί του σταθμού βάσης (όλοι οι κόμβοι ξεκινούν με next-hop neighbor το σταθμό βάσης του δικτύου μας).

    d. Multi-Hop application: το πρωτόκολλο που καθορίζει τη συμπεριφορά του δικτύου. Μέσω της ίδιας κλάσης υλοποιούμε και το ιεραρχικό πρωτόκολλο που περιγράφηκε στο προηγούμενο κεφάλαιο, με τις κατάλληλες μεθόδους που καθορίζουν τη συμπεριφορά των κόμβων αισθητήρων και των κόμβων επικεφαλής συστάδων (cluster heads) και τη δρομολόγηση των δεδομένων μεταξύ τους.

Σε κάθε περίπτωση η διαχείριση της ενέργειας των κόμβων μεταφέρθηκε στο επίπεδο της εφαρμογής, ώστε να υπάρχει έλεγχος της ενέργειας και της κατανάλωσής της, ανάλογα με τη λειτουργία του πρωτοκόλλου (πότε λειτουργεί ο επεξεργαστής, πότε λαμβάνει και πότε στέλνει δεδομένα το συστατικό ασύρματης μετάδοσης κλπ).

Το πρωτόκολλο απευθείας μετάδοσης μπορεί να προσομοιωθεί με το Multi-Hop πρωτόκολλο αρκεί να μην κληθεί η μέθοδος *setNeighbor()*. Όλοι οι κόμβοι του δικτύου έχουν σαν αρχική ρύθμιση το Σταθμό Βάσης σαν Next Hop Neighbor.

Το ιεραρχικό πρωτόκολλο που περιγράφηκε στο προηγούμενο κεφάλαιο προσομοιώνεται με το Multi-Hop πρωτόκολλο θέτοντας σαν τελικό προορισμό για κάθε κόμβο αισθητήρα, τον επικεφαλής της ομάδας στην οποία ανήκει.

Για την προσομοίωση του LEACH, χρησιμοποιήθηκε έτοιμη εφαρμογή με συστατικά παρόμοιας δομής με τις εφαρμογές που περιγράφηκαν πιο πάνω και η οποία έχει μεταφερθεί από υλοποίηση που έγινε στο ns-2. Επιπλέον, χρησιμοποιήθηκε και το κατάλληλο MAC συστατικό με τα απαραίτητα χαρακτηριστικά (non-persistent CSMA με scheduled access).

Τέλος, έγιναν αρκετές διορθώσεις και τροποποιήσεις σε συστατικά τα οποία δεν είχαν σωστή ή κατάλληλη για τις προσομοιώσεις μας συμπεριφορά. Κυρίως οι αλλαγές αυτές αφορούσαν στα συστατικά SensorNodePositionTracker.java (εδώ υλοποιήθηκε ο αλγόριθμος εύρεσης του next-hop-neighbor) και WirelessPhy.java (ό,τι έχει σχέση με την ασύρματη μετάδοση).

Κεφάλαιο 5 – ΠΡΟΣΟΜΟΙΩΣΕΙΣ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

5.1 Ρυθμίσεις και δεδομένα προσομοιωτή

Σε όλες τις προσομοιώσεις που έγιναν, χρησιμοποιήθηκαν κοινές ρυθμίσεις ώστε να μπορεί να γίνει άμεση σύγκριση των αποτελεσμάτων. Πιο συγκεκριμένα:

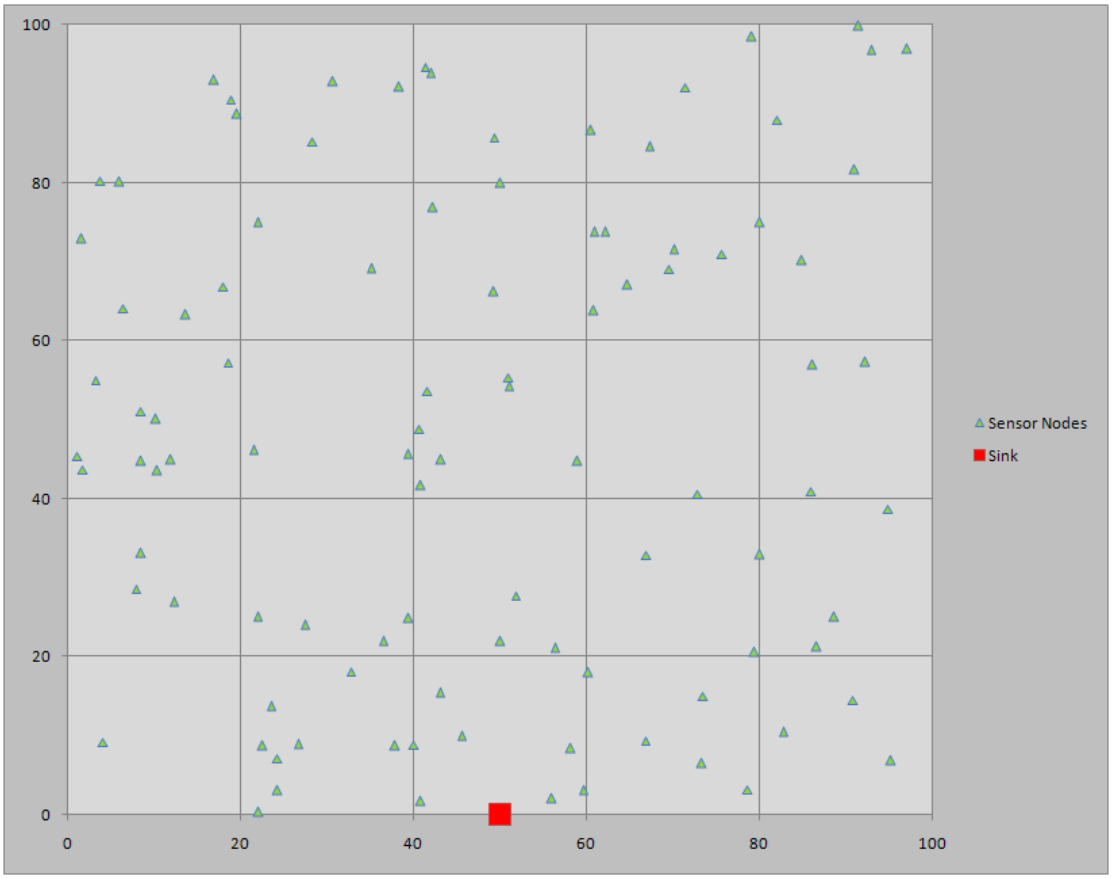
- Το μοντέλο διάδοσης που χρησιμοποιήθηκε είναι το free space propagation model
- Κέρδος κεραίας εκπομπής Gt = 1dbi
- Κέρδος κεραίας λήψης Gr = 1dbi
- Το hardware που προσομοιώθηκε είναι η πλατφόρμα μAmps με τα εξής χαρακτηριστικά:
    - Συχνότητα λειτουργίας του ασύρματου συστήματος είναι τα 914MHz
    - Bandwidth μετάδοσης 1Mbps
    - Κατώφλι ευαισθησίας λήψης $6x10^{-9}$ Watts
    - Κατώφλι ευαισθησίας Carrier Sense (CSThreshold) $1x10^{-9}$ Watts
    - Απαιτούμενη ισχύς εκπομπής
      $Pt = (2,408x10^{-10}/(Gr \times Gt)) \times bandwidth \times distance^2$
    - Ισχύς λειτουργίας ηλεκτρονικών ασύρματου συστήματος $50x10^{-9}$ Watts
    - Ισχύς λειτουργίας επεξεργαστή:
        - CPU_IDDLE = $2,9x10^{-3}$ Watts
        - CPU_SLEEP = $1,9x10^{-6}$ Watts
        - CPU_ACTIVE = $2,9x10^{-3}$ Watts
        - CPU_OFF = $1x10^{-9}$ Watts
- Η ενέργεια κάθε αισθητήριου κόμβου είναι 0,25 Joules
- Μέγεθος πακέτου (μαζί τους όποιους headers) = 175 bytes (1400 bits)

- Το πλήθος των κόμβων αισθητήρων του δικτύου είναι 100
- Η περιοχή ανάπτυξης του δικτύου είναι ή 100m x 100m για πυκνό δίκτυο ή 200m x200m για αραιό δίκτυο.
- Σε όλες τις προσομοιώσεις εκτός του LEACH το MAC πρωτόκολλο που χρησιμοποιήθηκε είναι το 802.11
- Οι κόμβοι στόχοι παράγουν αισθητηριακά δεδομένα με ρυθμό 1 event/sec
- Το μοντέλο διάδοσης των αισθητηριακών δεδομένων είναι το σεισμικό μοντέλο

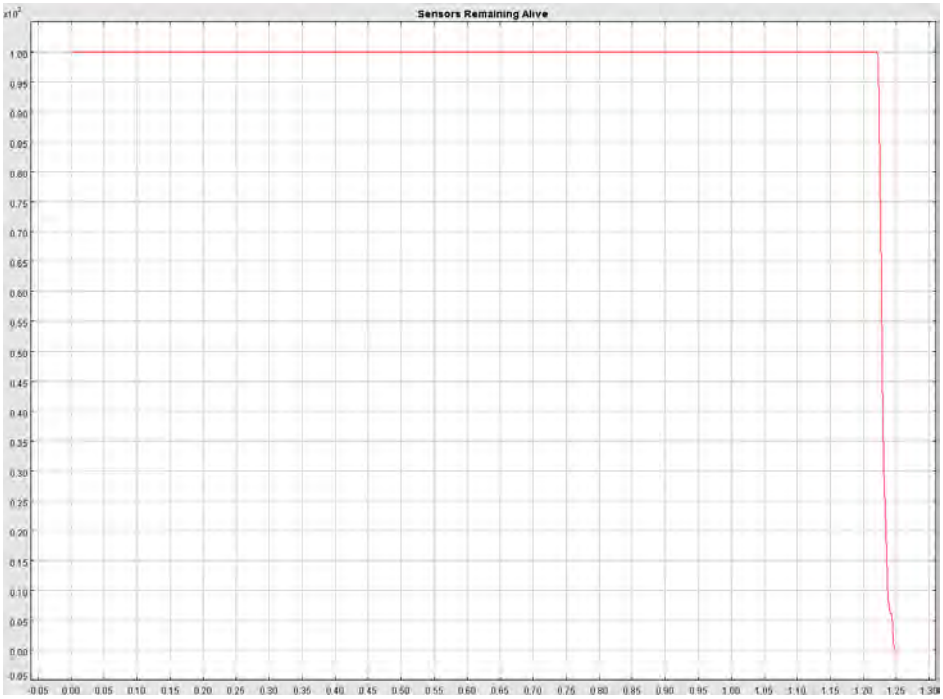
5.2 Πρωτόκολλα

5.2.1 One-Hop (απευθείας μετάδοση)

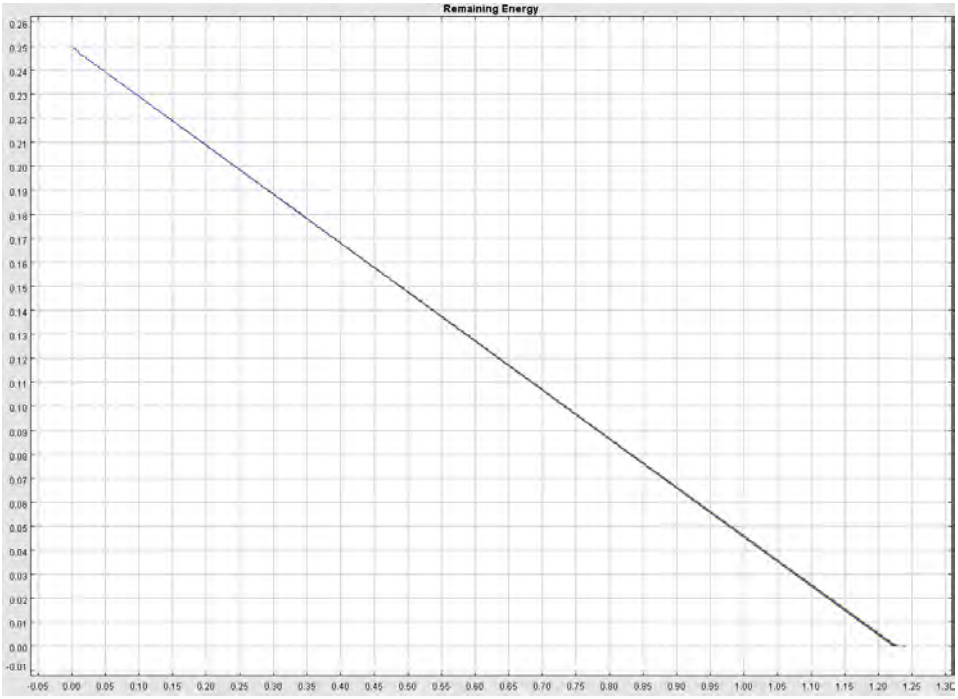
Όπως έχουμε περιγράψει σε προηγούμενο κεφάλαιο η πιο απλή μορφή επικοινωνίας είναι η απευθείας αποστολή δεδομένων στο σταθμό βάσης. Τοποθετήσαμε 100 κόμβους αισθητήρες με τυχαίο τρόπο σε πλέγμα 100x100 όπως φαίνεται στο παρακάτω σχήμα:



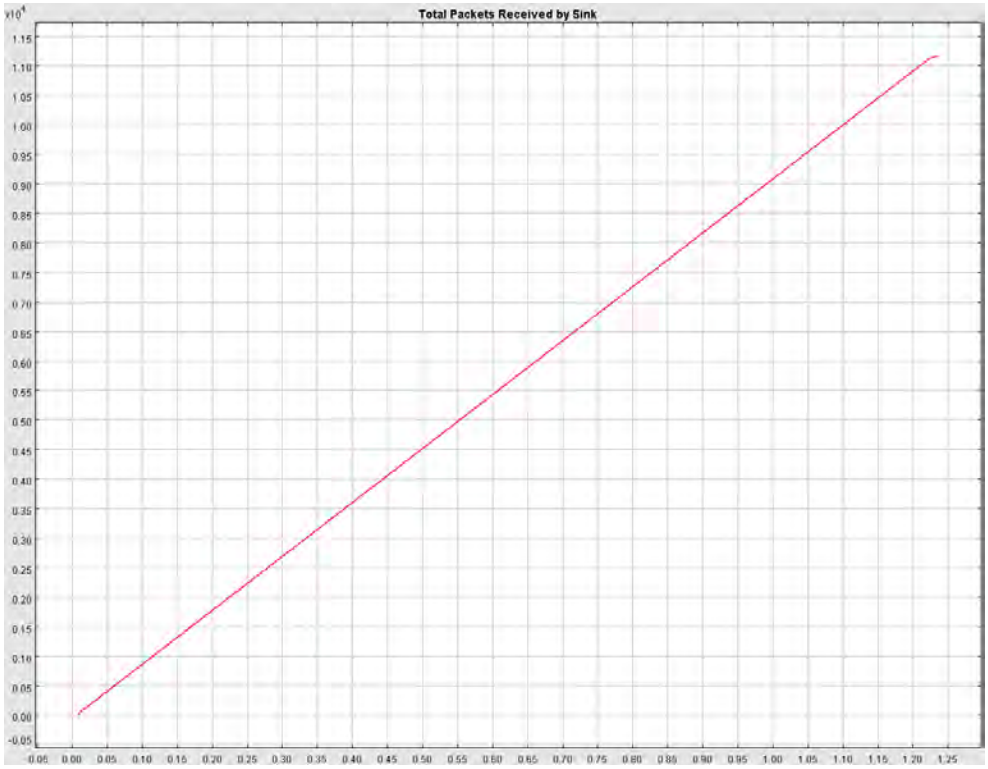
Εικόνα 43 - Τοπολογία δικτύου αισθητήρων σε πλέγμα 100x100



Εικόνα 44 - Ζωντανοί κόμβοι στο χρόνο



Εικόνα 45 - Ενέργεια κόμβων στο χρόνο



Εικόνα 46 - Αριθμός ληφθέντων πακέτων στο χρόνο

Στατιστικά Στοιχεία One-Hop δικτύου σε περιοχή 100x100

100 nodes dead at 1247.0

Total packets dropped at Application layer: 0

Total packets dropped at physical layer: 77

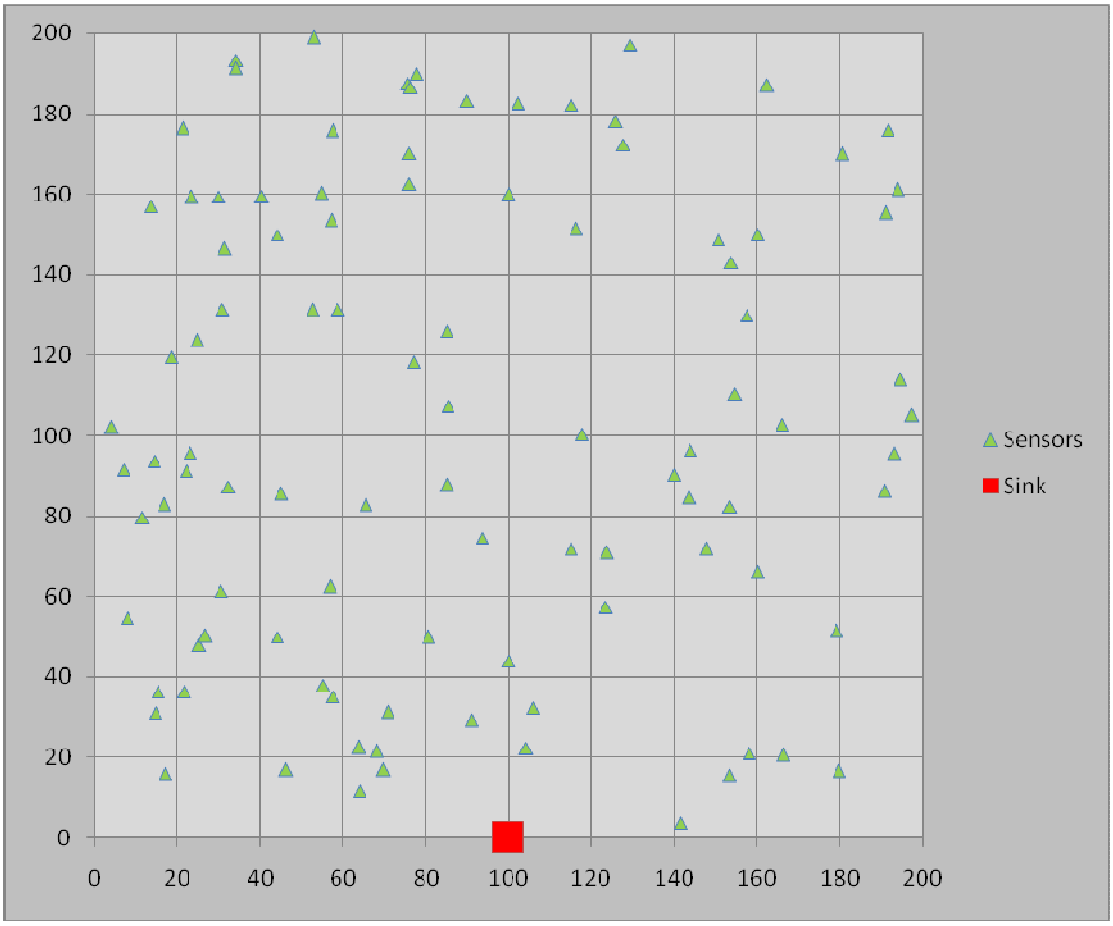
Average Latency on BS is: 0.00219696869601

Total Packets sent from all nodes: 11500

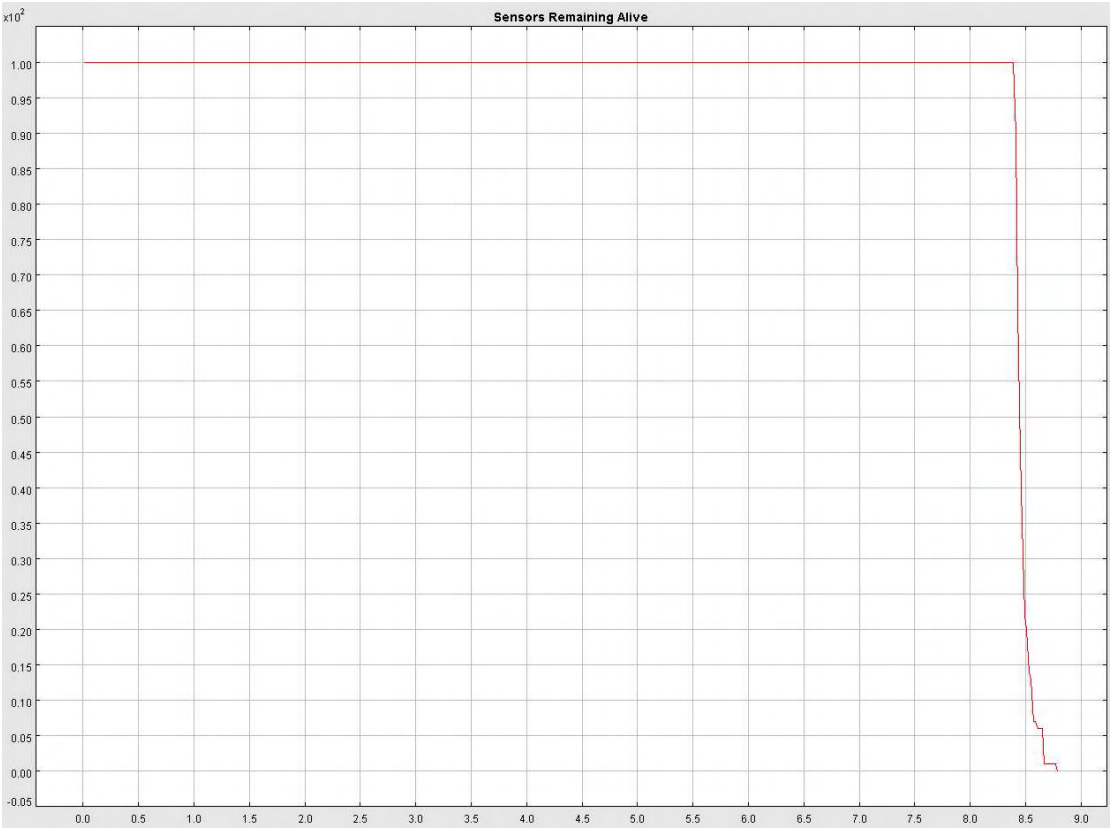
Number of Dropped Packets: 77

Success Rate: 97.1043478261

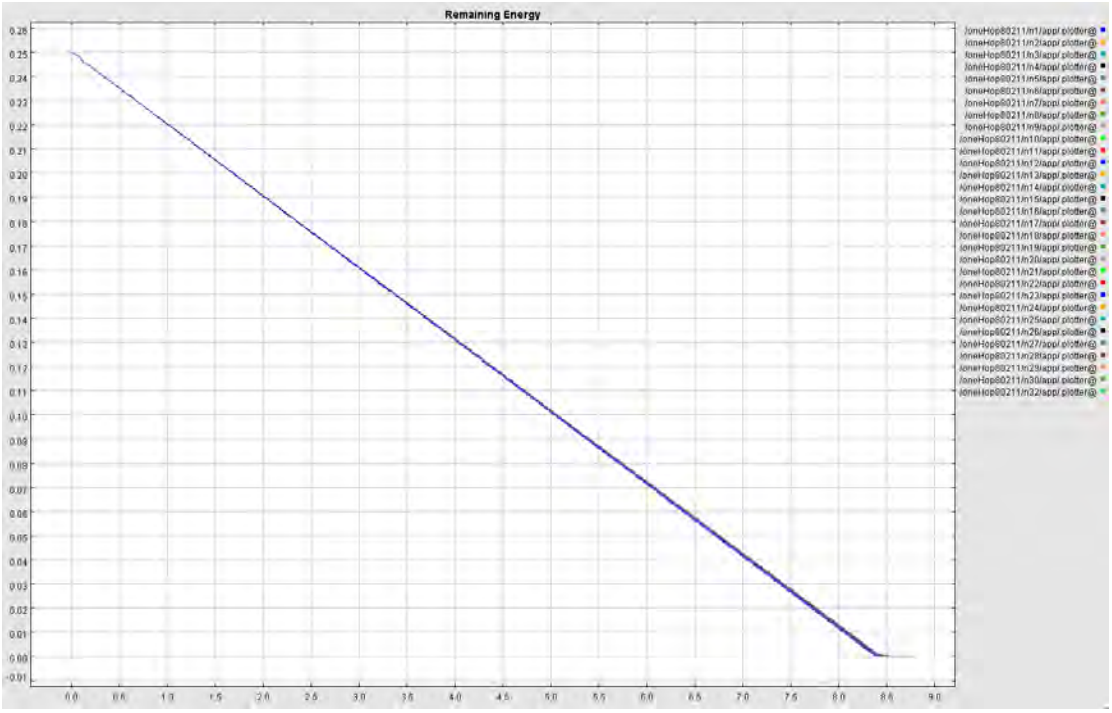
Τυχαία τοποθέτηση 100 αισθητήρων σε πλέγμα 200x200



Εικόνα 47 - Τοπολογία δικτύου αισθητήρων σε πλέγμα 200x200



Εικόνα 48 - Ζωντανοί κόμβοι στο χρόνο



Εικόνα 49 - Ενέργεια κόμβων στο χρόνο

Εικόνα 50 Αριθμός ληφθέντων πακέτων στο χρόνο

Στατιστικά Στοιχεία One-Hop δικτύου σε περιοχή 200x200

100 nodes dead at 879.0

Total packets dropped at Application layer: 0

Total packets dropped at physical layer: 57

Average Latency on BS is: 0.00233019275189

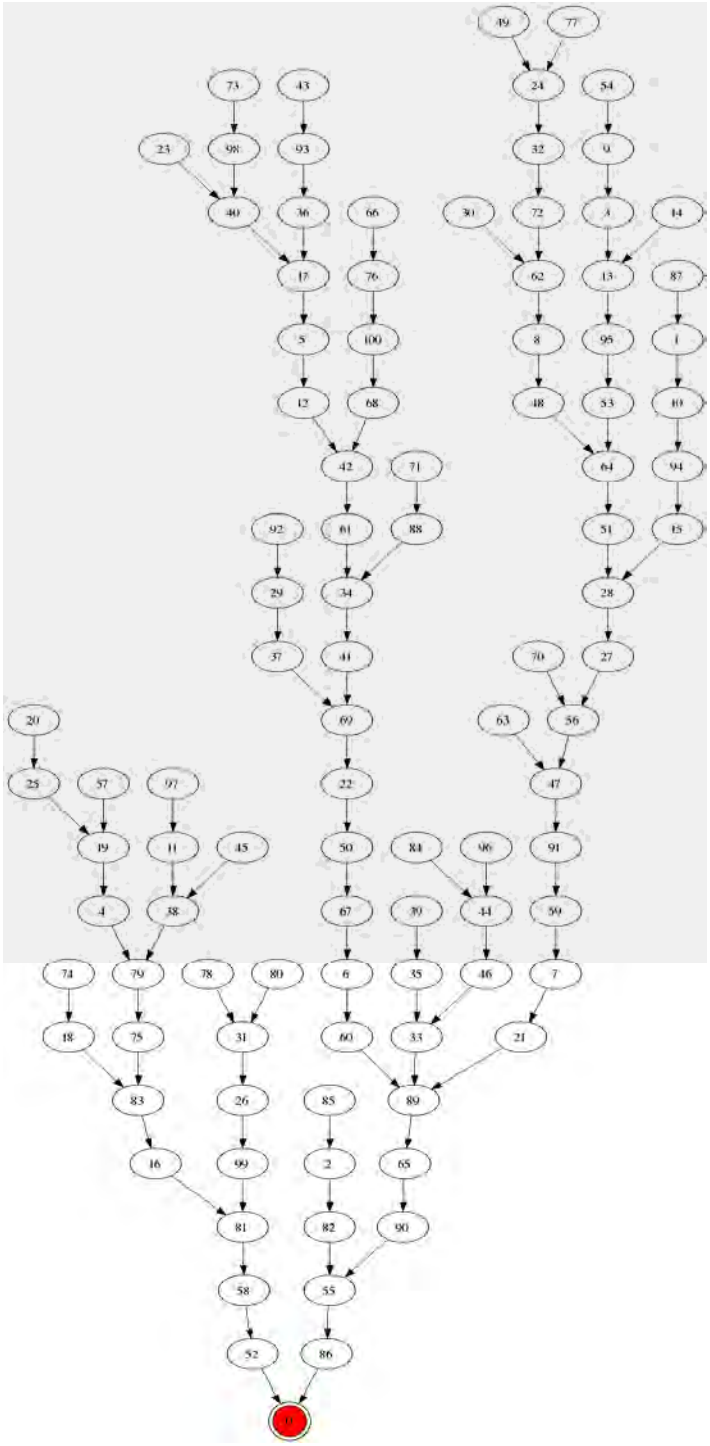
Total Packets sent from all nodes: 7812

Number of Dropped Packets: 57

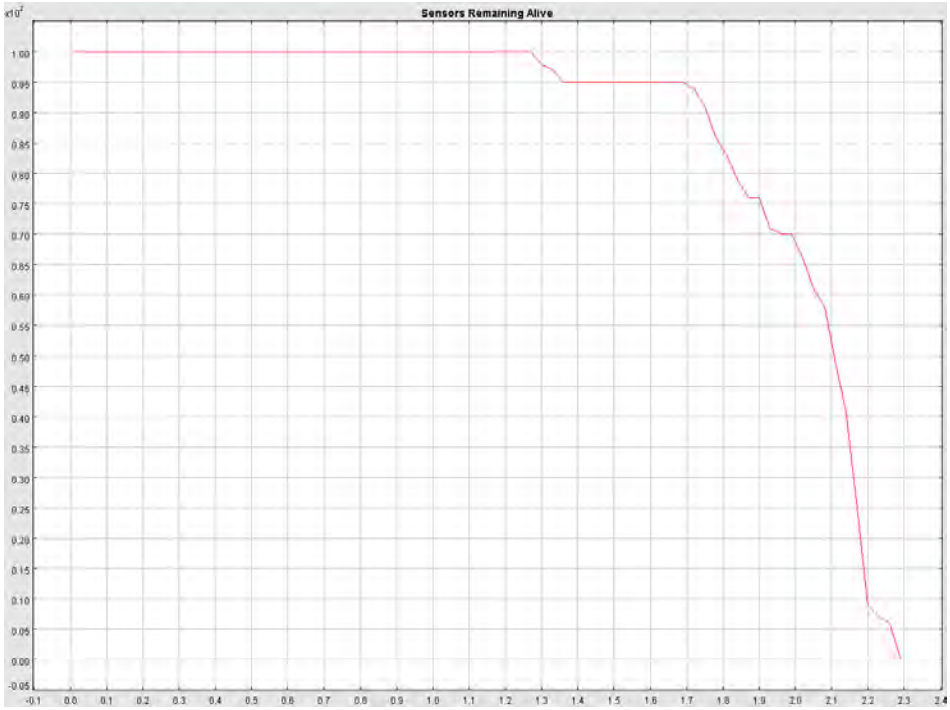
Success Rate: 40.2713773682

5.2.2 Multi-Hop

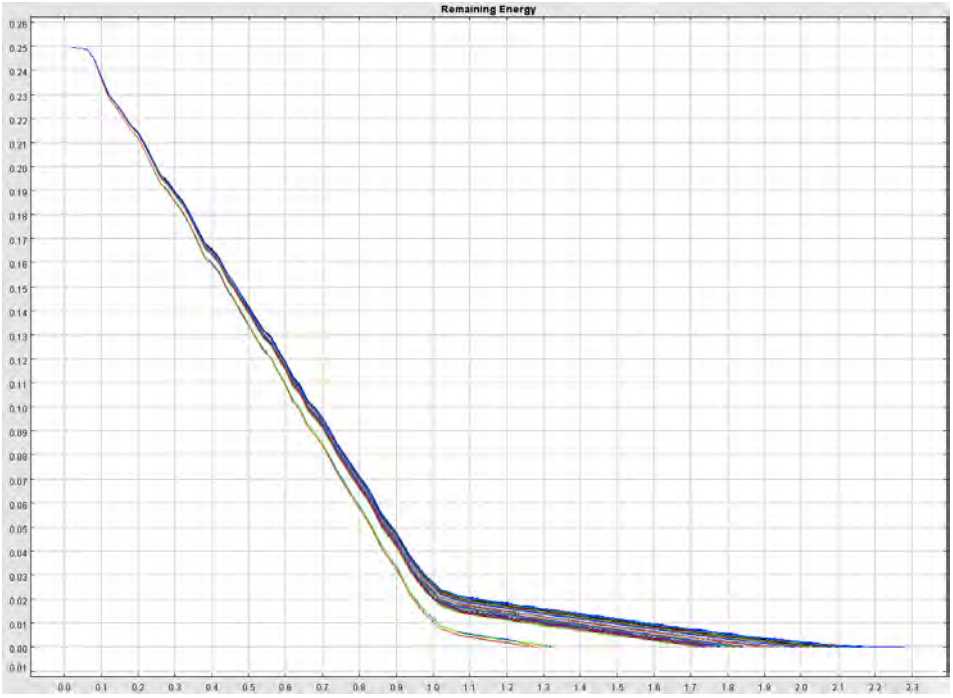
1. Δίκτυο 100 κόμβων σε περιοχή 100m x 100m



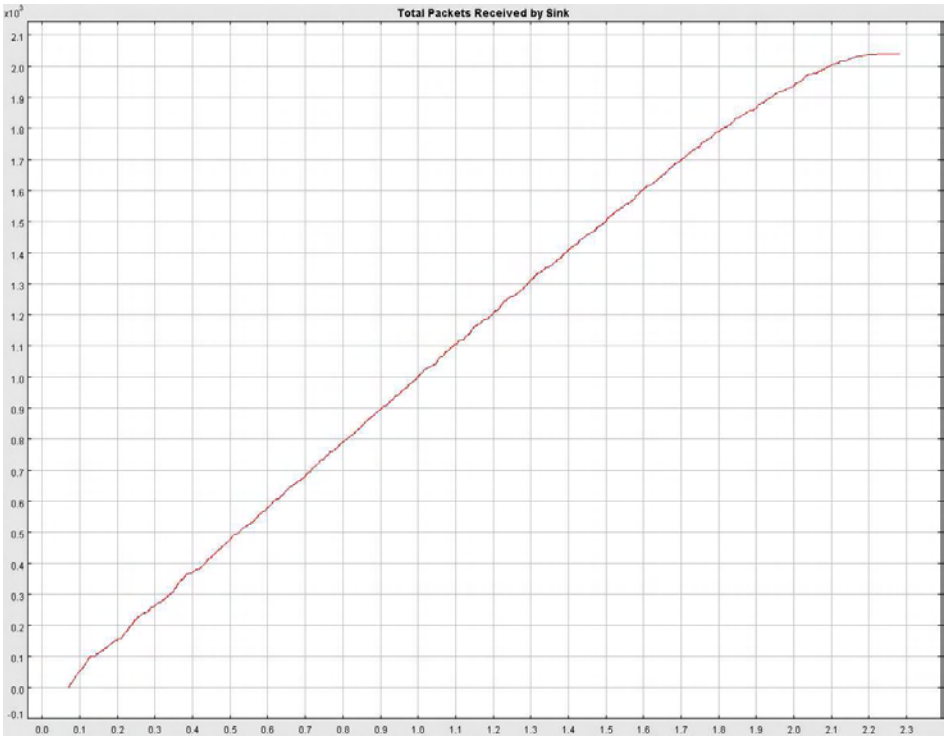
Εικόνα 51 - Γράφος MultiHop δικτύου με metric την απόσταση



Εικόνα 52- Ζωντανοί κόμβοι στο χρόνο



Εικόνα 53 - Ενέργεια κόμβων στο χρόνο



Εικόνα 54 Αριθμός ληφθέντων πακέτων στο χρόνο

Στατιστικά Στοιχεία Multi-Hop δικτύου σε περιοχή 100x100 με metric την απόσταση

100 out of 100 nodes dead at 229.0

Total packets dropped at Application layer: 0

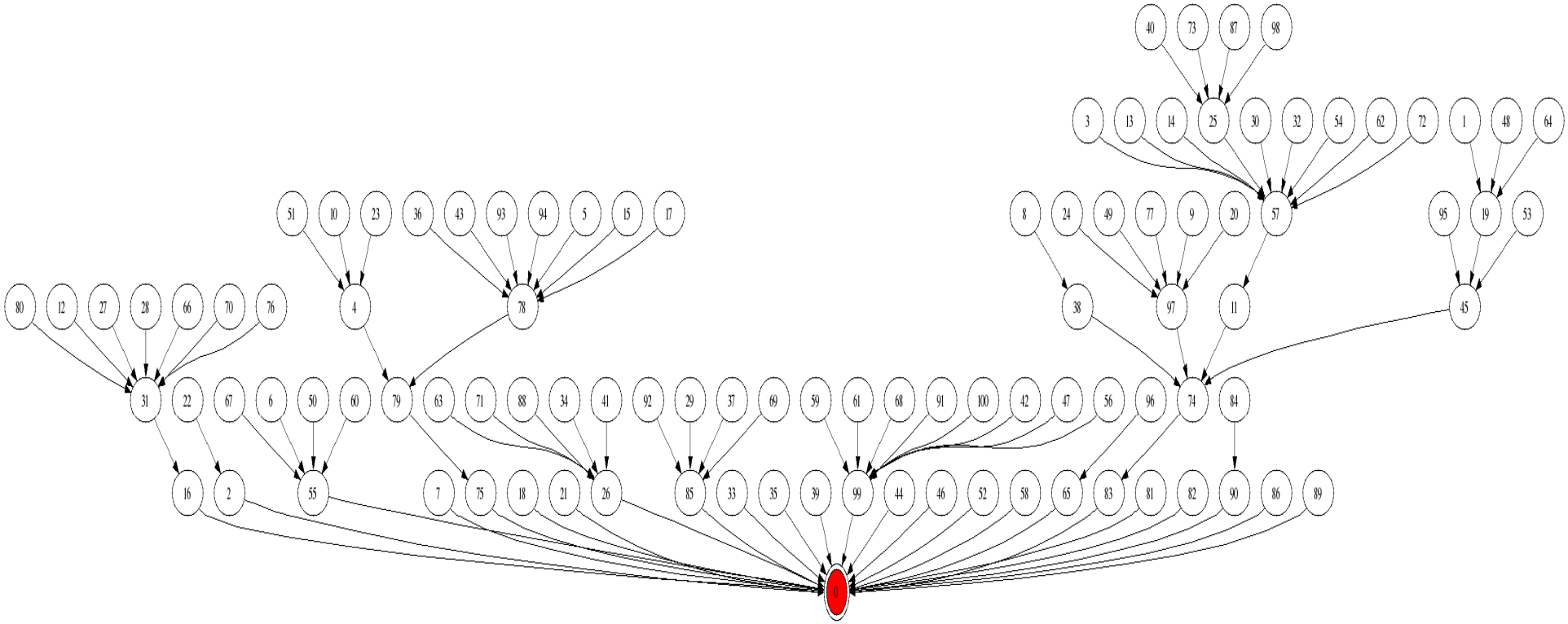
Total packets dropped at physical layer: 53

Average Latency on BS is: 0.0216562640552

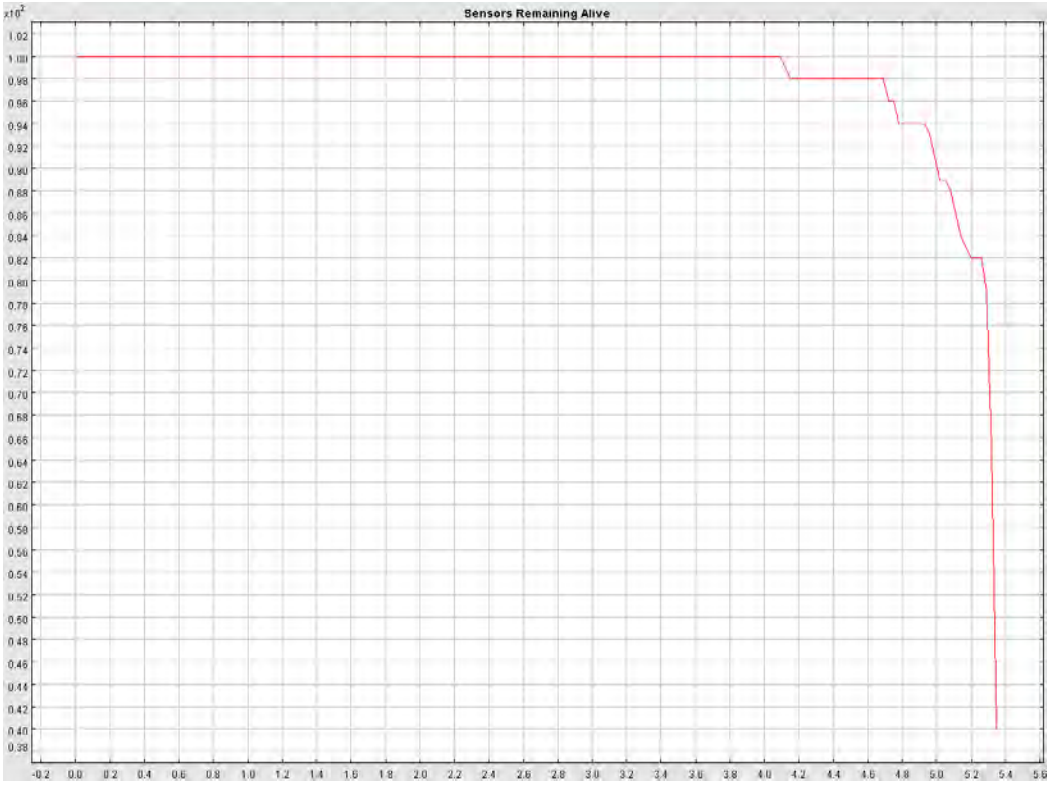
Total Packets sent from all nodes: 2072

Number of Dropped Packets: 53

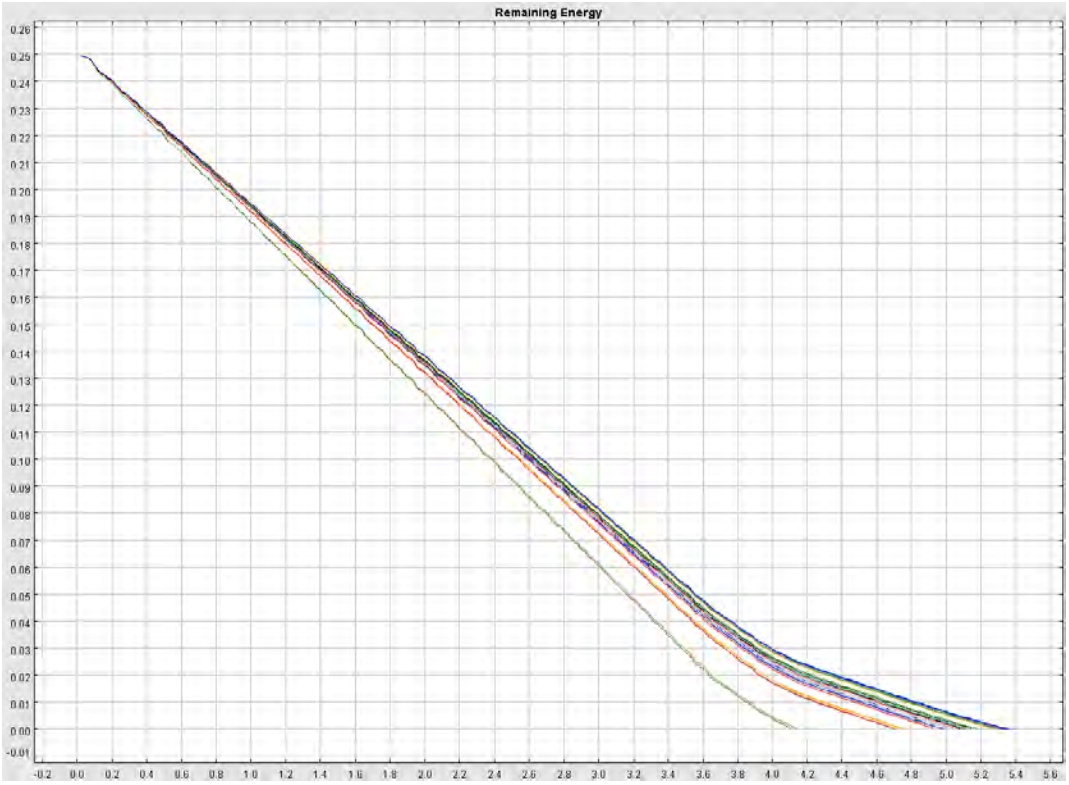
Success Rate: 98.5038610039



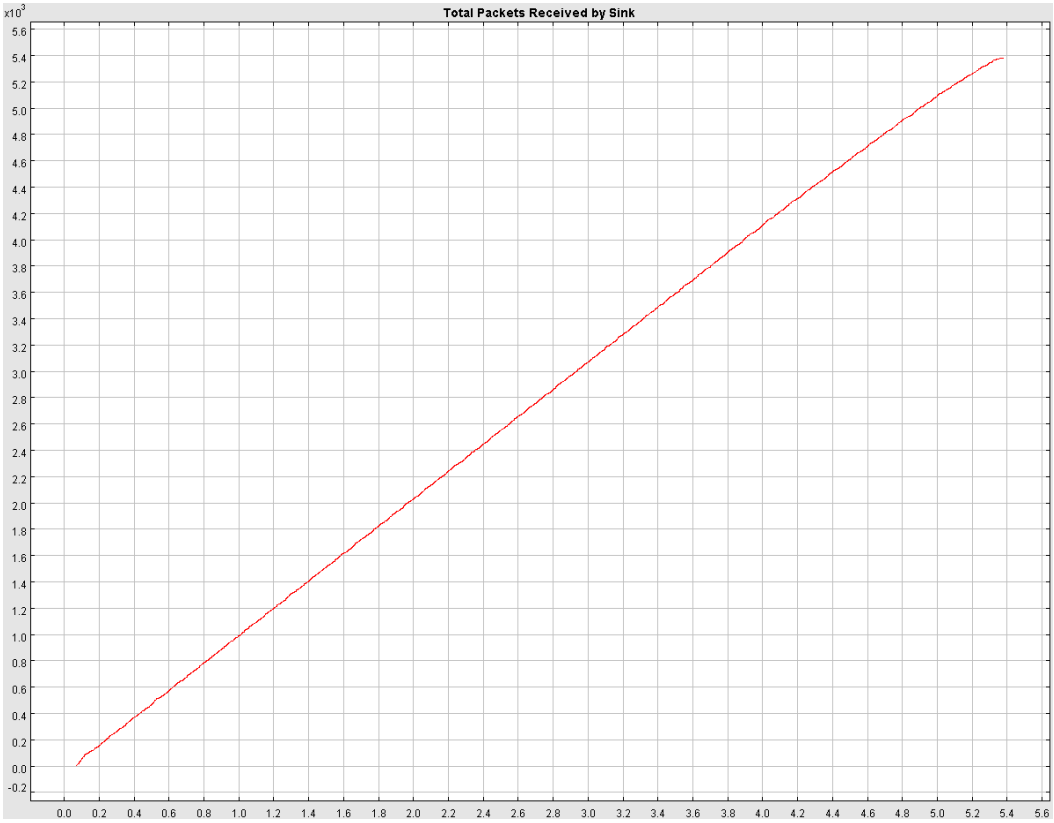
Εικόνα 55 - Γράφος MultiHop δικτύου με metric την ενέργεια



Εικόνα 56 - Ζωντανοί κόμβοι στο χρόνο



Εικόνα 57 - Ενέργεια κόμβων στο χρόνο



Εικόνα 58 - Αριθμός ληφθέντων πακέτων στο χρόνο

Στατιστικά Στοιχεία Multi-Hop δικτύου σε περιοχή 100x100 με metric την ενέργεια

100 out of 100 nodes dead at 544.0

Total packets dropped at Application layer: 0

Total packets dropped at physical layer: 48

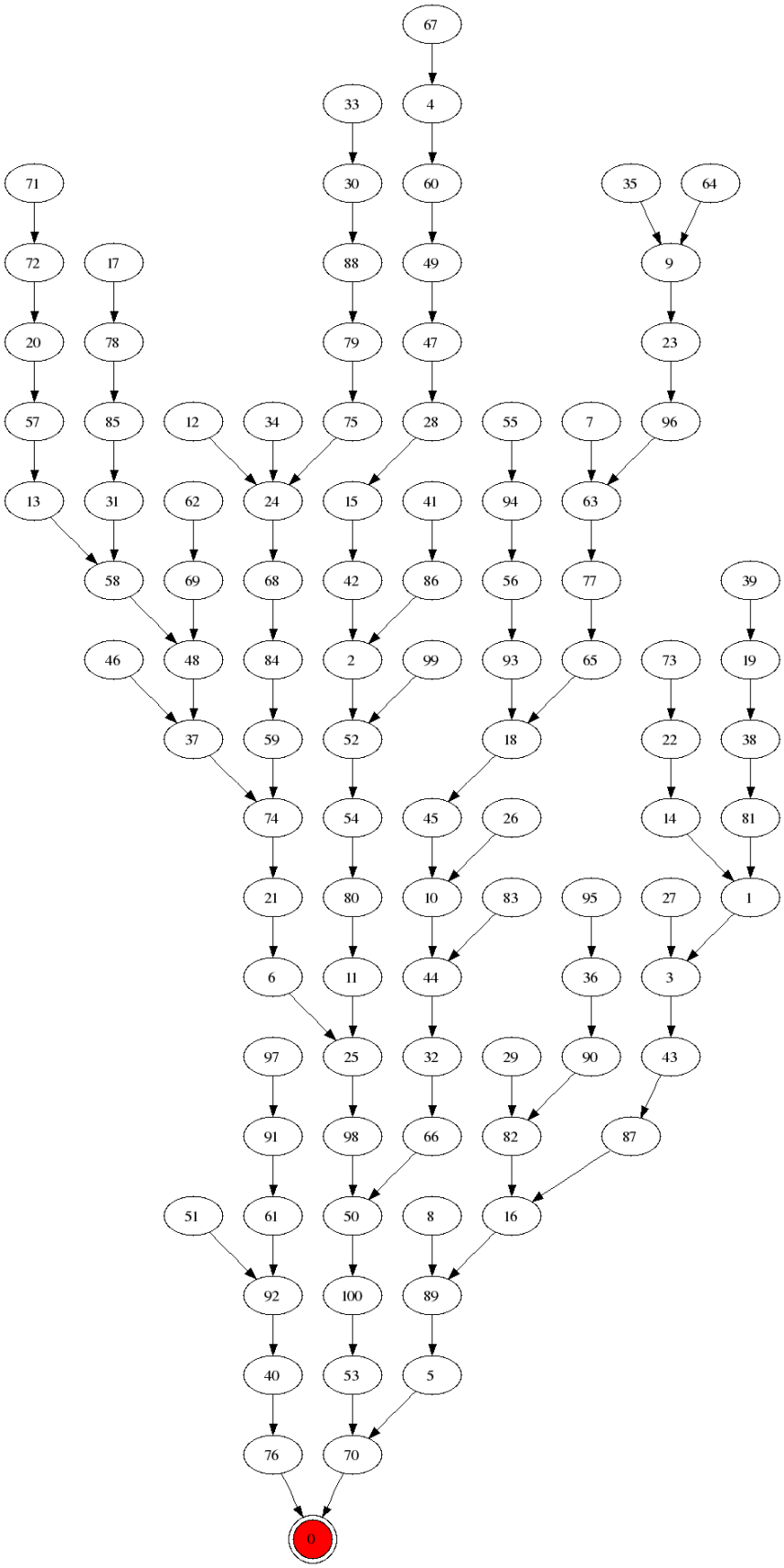
Average Latency on BS is: 0.00569471703546

Total Packets sent from all nodes: 5428

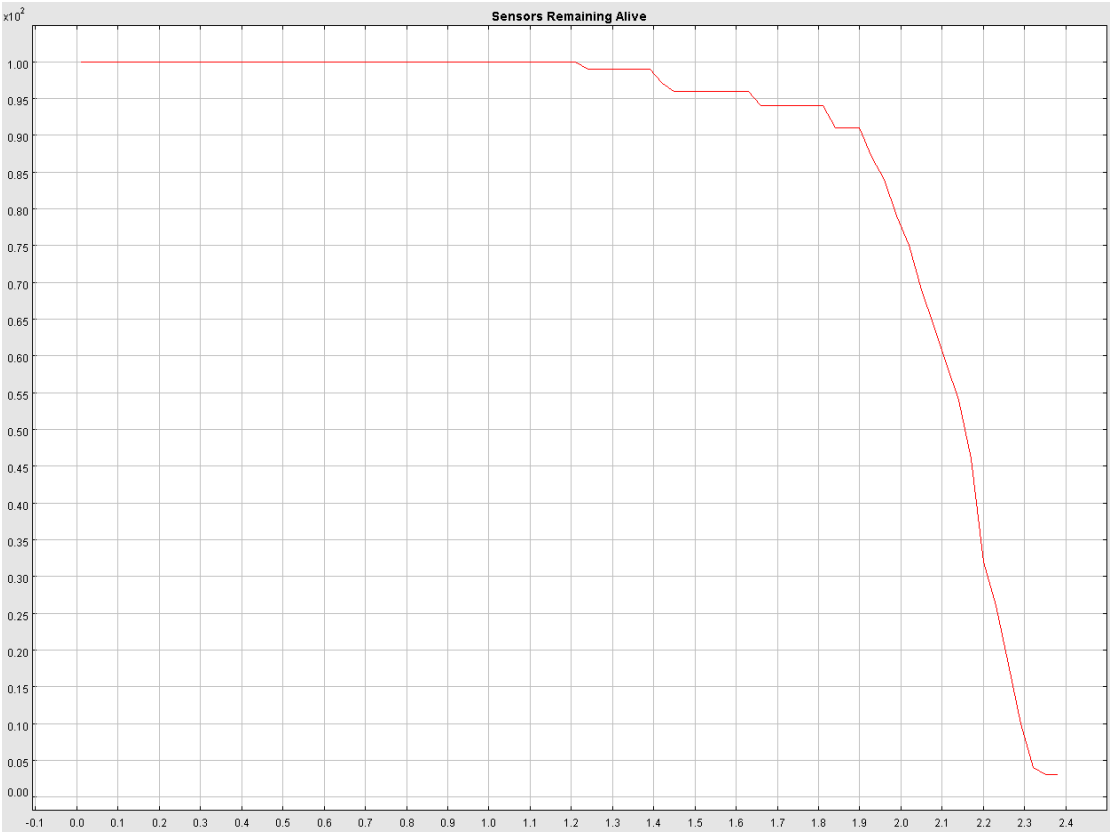
Number of Dropped Packets: 48

Success Rate: 99.1893883567

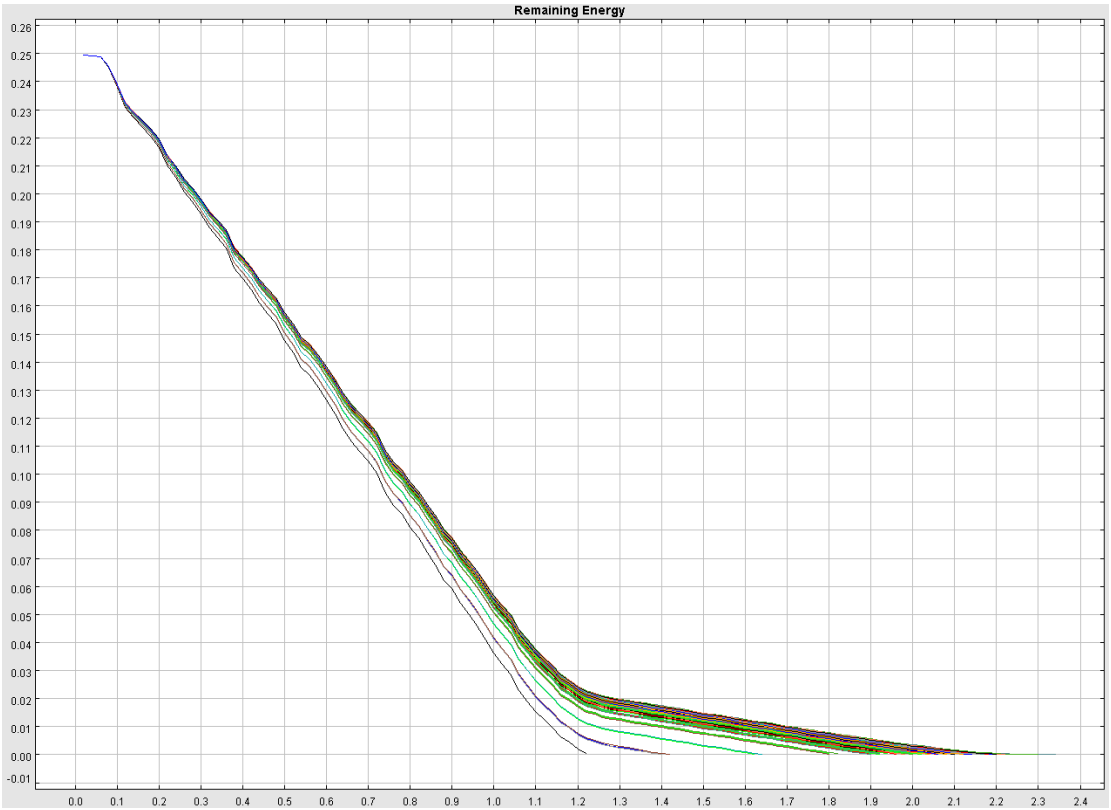
2. Δίκτυο 100 κόμβων σε περιοχή 200m x 200m



Εικόνα 59 - Γράφος MultiHop δικτύου με metric την απόσταση



Εικόνα 60 - Ζωντανοί κόμβοι στο χρόνο



Εικόνα 61 - Ενέργεια κόμβων στο χρόνο

Εικόνα 62 - Αριθμός ληφθέντων πακέτων στο χρόνο

Στατιστικά Στοιχεία Multi-Hop δικτύου σε περιοχή 200x200 με metric την απόσταση

100 out of 100 nodes dead at 241.0

Total packets dropped at Application layer: 0

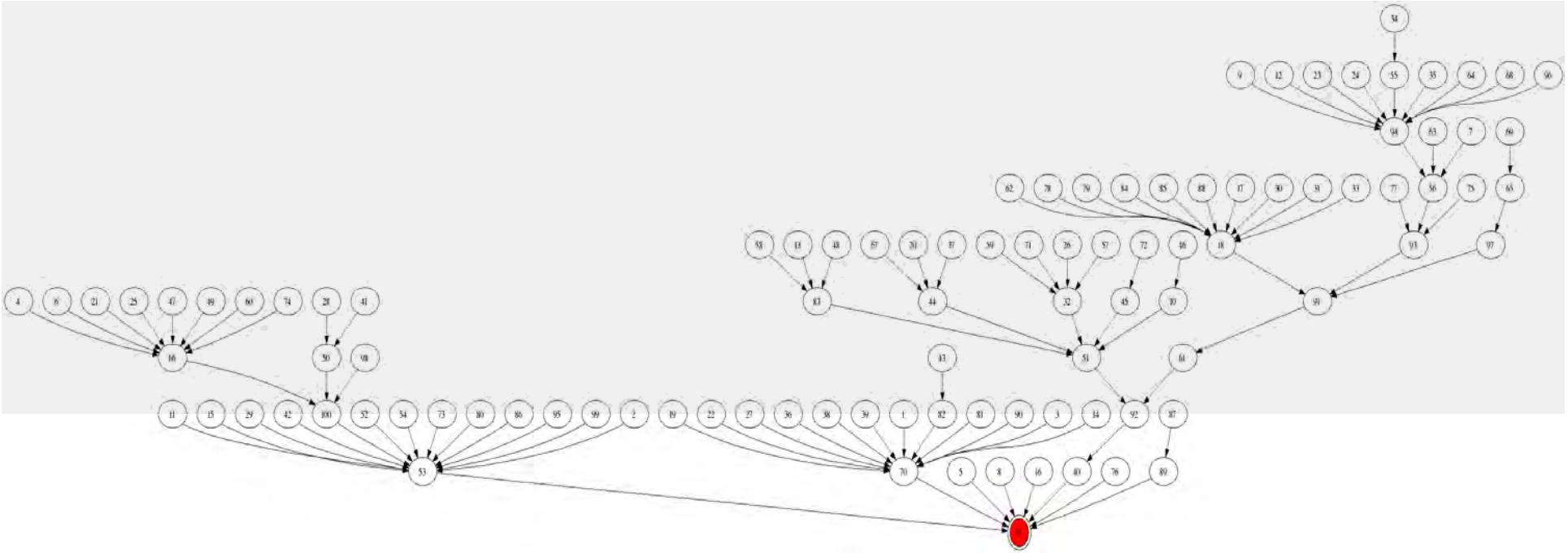
Total packets dropped at physical layer: 39

Average Latency on BS is: 0.0239991048468

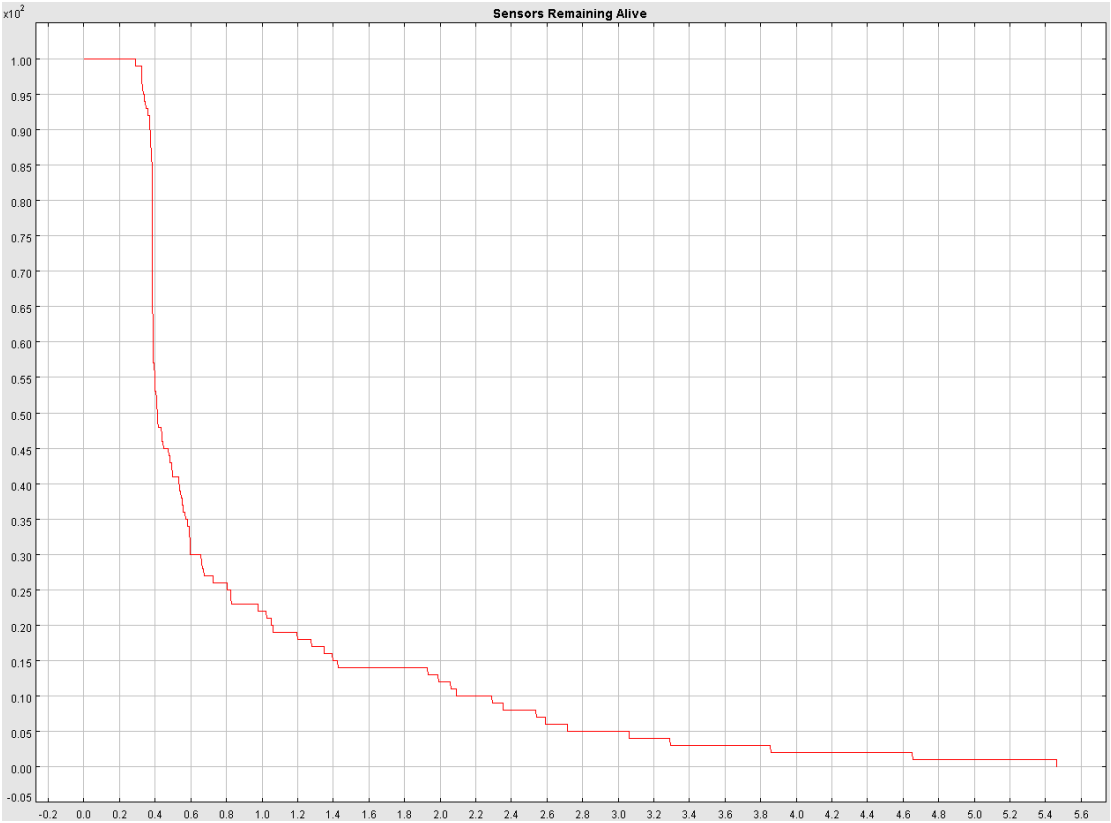
Total Packets sent from all nodes: 2111

Number of Dropped Packets: 39

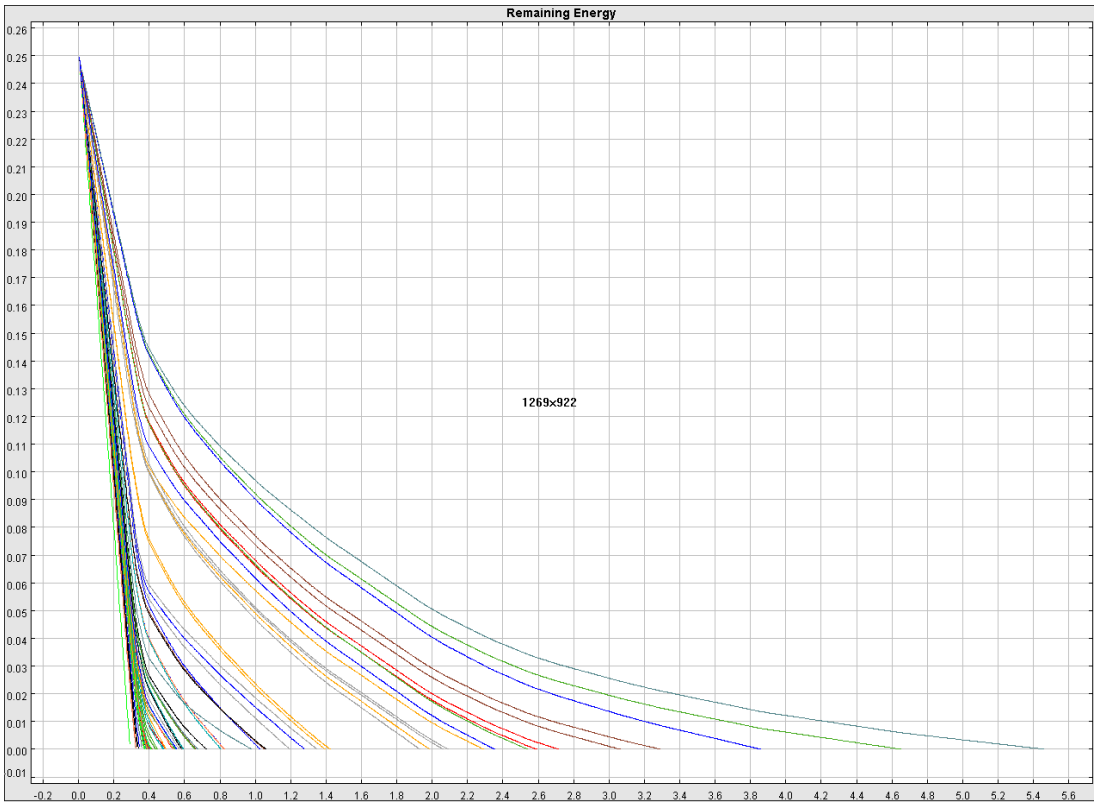
Success Rate: 71.9090478446



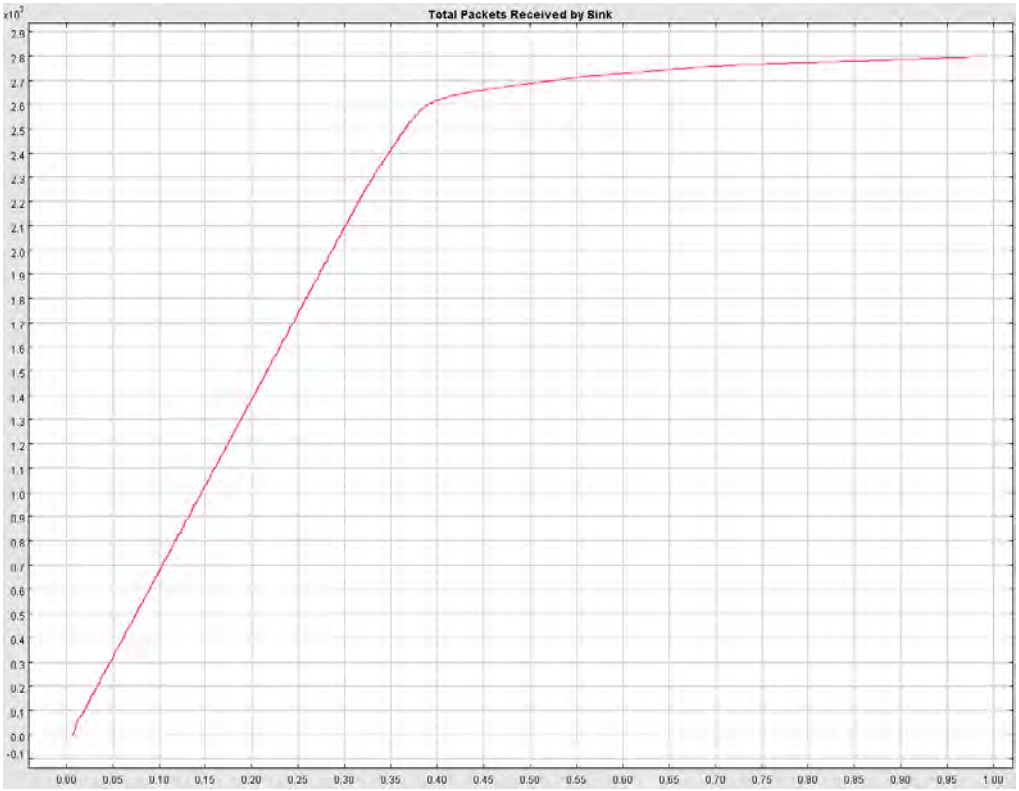
Εικόνα 63 - Γράφος MultiHop δικτύου με metric την ενέργεια



Εικόνα 64 - Ζωντανοί κόμβοι στο χρόνο



Εικόνα 65 - Ενέργεια κόμβων στο χρόνο



Εικόνα 66 - Αριθμός ληφθέντων πακέτων στο χρόνο

Στατιστικά Στοιχεία

100 out of 100 nodes dead at 5464.0

Total packets dropped at Application layer: 0

Total packets dropped at physical layer: 30

Average Latency on BS is: 0.0105698548653

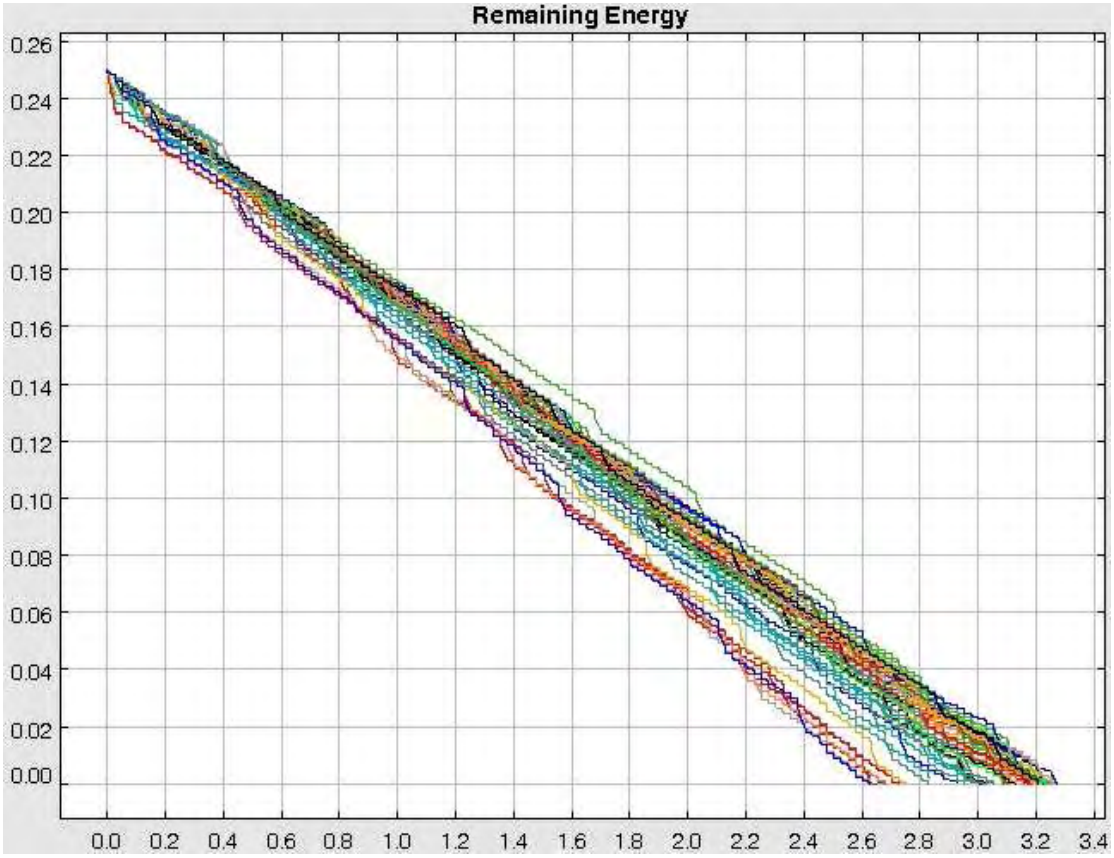
Total Packets sent from all nodes: 8704

Number of Dropped Packets: 30

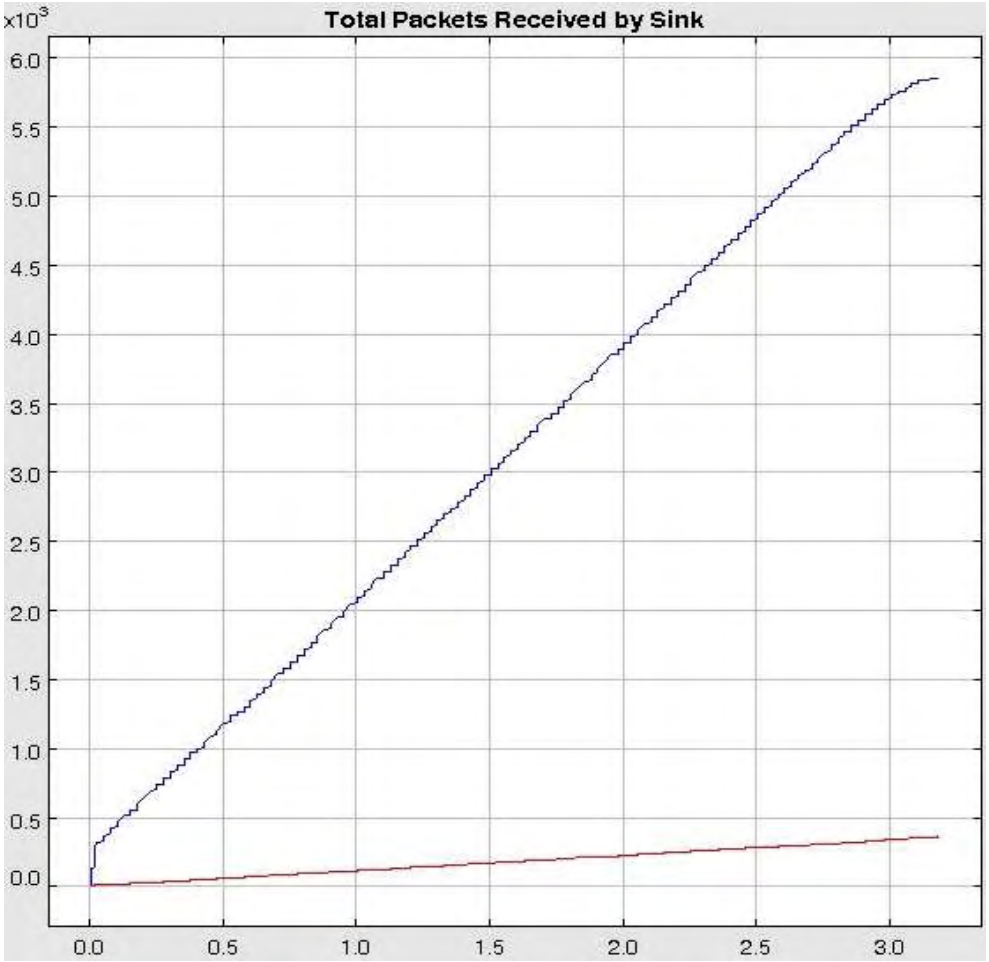
Success Rate: 38.1001838235

5.2.3 LEACH

1. Δίκτυο 100 κόμβων σε περιοχή 100m x 100m



Εικόνα 67 - Ενέργεια κόμβων στο χρόνο



Εικόνα 68 - Αριθμός πακέτων που έστειλαν οι κόμβοι / αριθμός πακέτων που έλαβε ο σταθμός βάσης

Στατιστικά Στοιχεία

Χρόνος ζωής δικτύου 3271.0

Total packets dropped at Application layer: 0

Total packets dropped at physical layer: 0

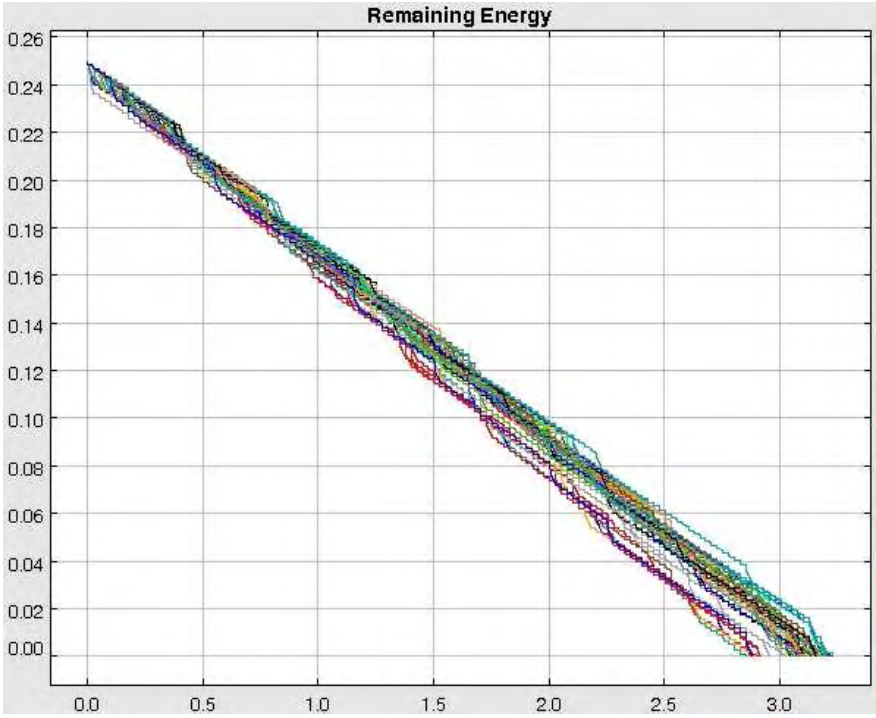
Drops due to collisions (discovered at MAC layer: 37

Total Packets sent from all nodes: 5176

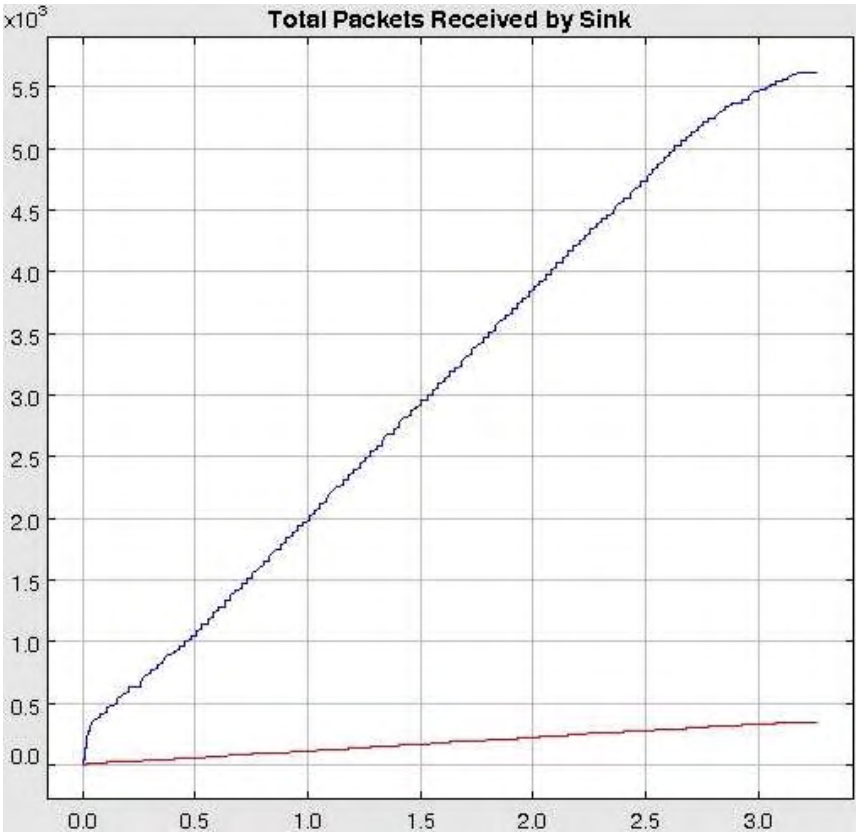
Number of Dropped Packets: 37

Packet Sink/Node ratio (Success Rate): 5.56414219474

2. Δίκτυο 100 κόμβων σε περιοχή 200m x 200m



Εικόνα 69 - Ενέργεια κόμβων στο χρόνο



Εικόνα 70 - Αριθμός πακέτων που έστειλαν οι κόμβοι / αριθμός πακέτων που έλαβε ο σταθμός βάσης

Στατιστικά Στοιχεία

Χρόνος ζωής δικτύου 3229.0

Base Station Received 235

Total packets dropped at Application layer: 0

Total packets dropped at physical layer: 0

Drops due to collisions (discovered at MAC layer): 18

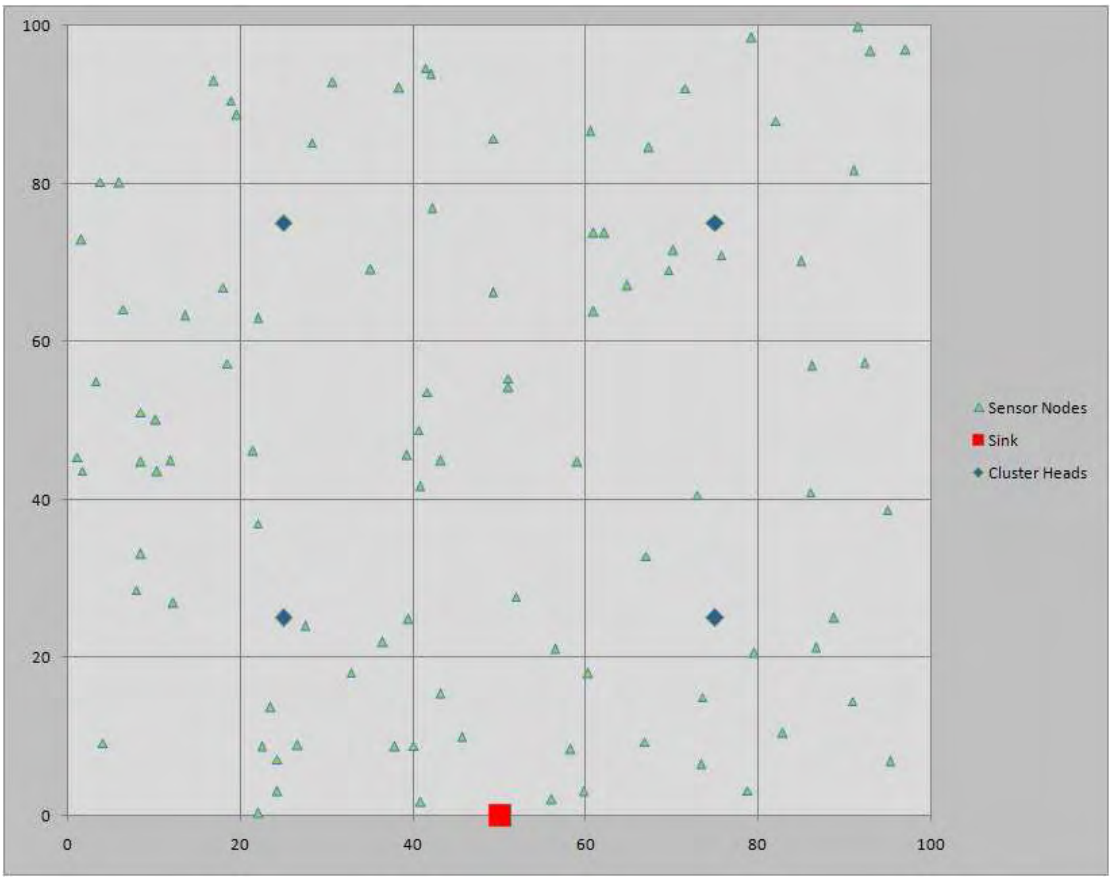
Total Packets sent from all nodes: 5138

Number of Dropped Packets: 18

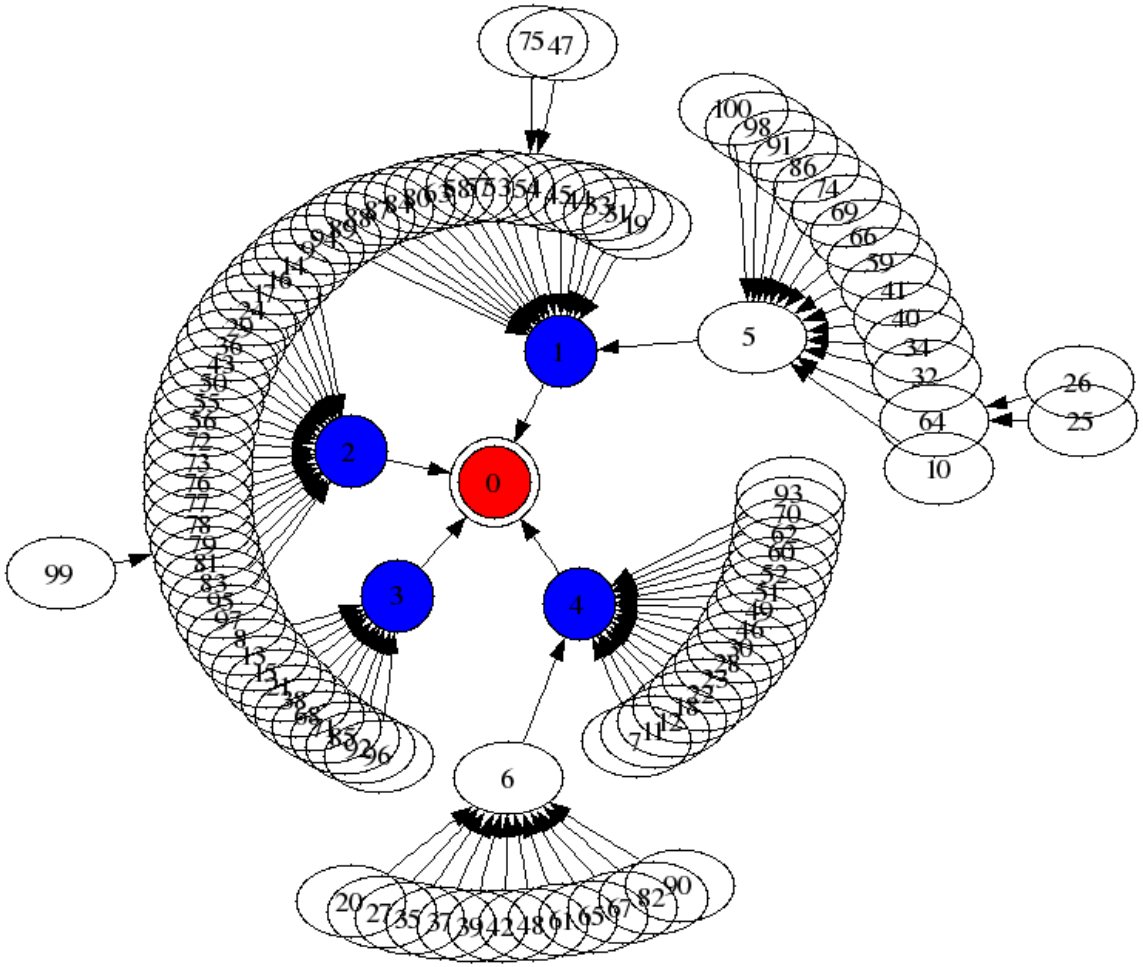
Packet Sink/Node ratio: 2.34110380554

5.2.4 Static Clustering with optimal energy routing

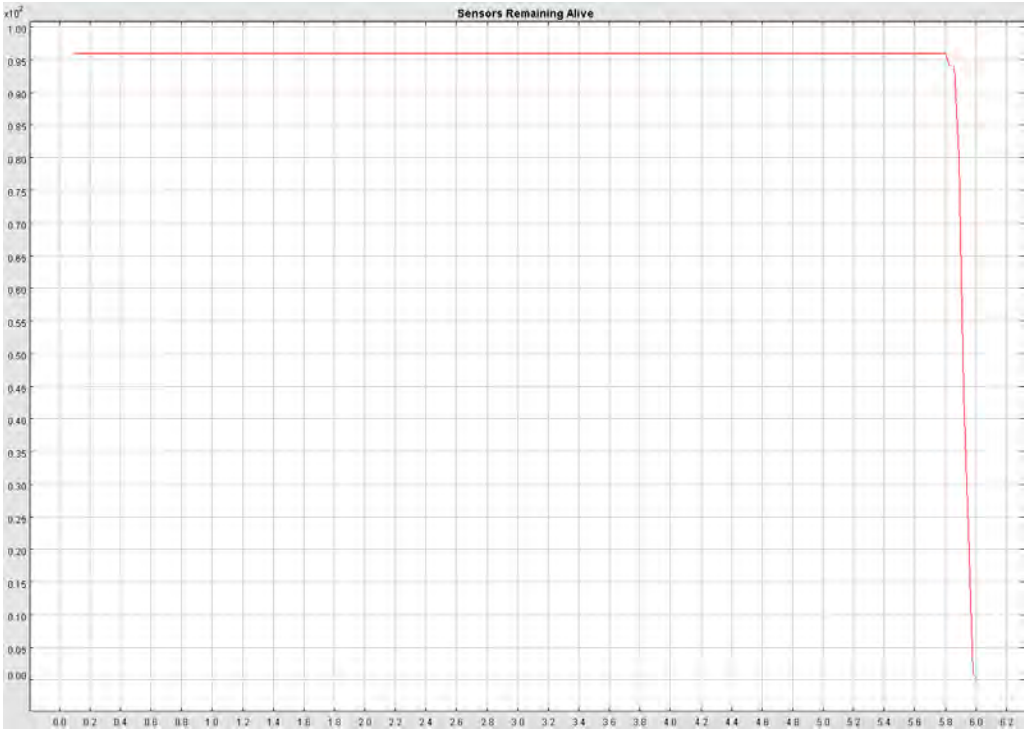
1. Δίκτυο 100 κόμβων (4 Cluster Heads) σε περιοχή 100m x 100m



Εικόνα 71 - Τοπολογία δικτύου αισθητήρων σε πλέγμα 100x100

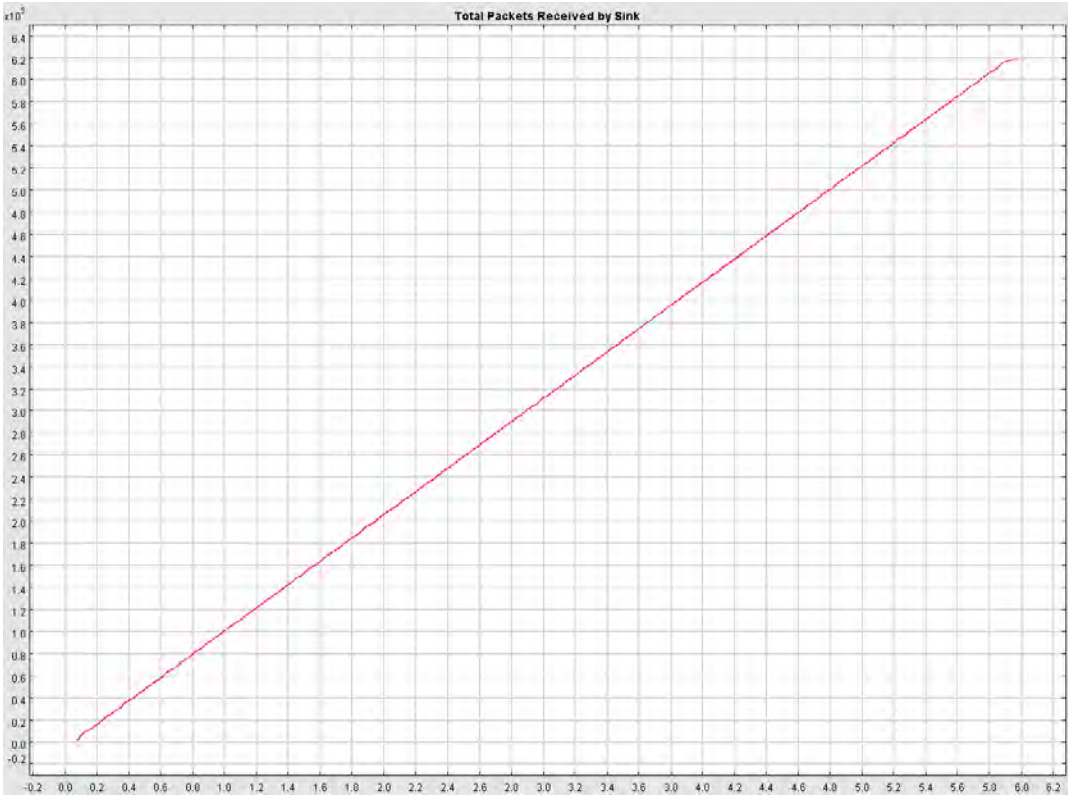


Εικόνα 72 - Γράφος Clustered δικτύου (100x100)



Εικόνα 73 - Ζωντανοί κόμβοι στο χρόνο

Εικόνα 74 - Ενέργεια κόμβων στο χρόνο



Εικόνα 75 - Αριθμός ληφθέντων πακέτων στο χρόνο

Στατιστικά Στοιχεία

94 out of 94 nodes dead at 601.0

Total packets dropped at Application layer: 0

Total packets dropped at physical layer: 14

Average Latency on BS is: 0.00490970182542

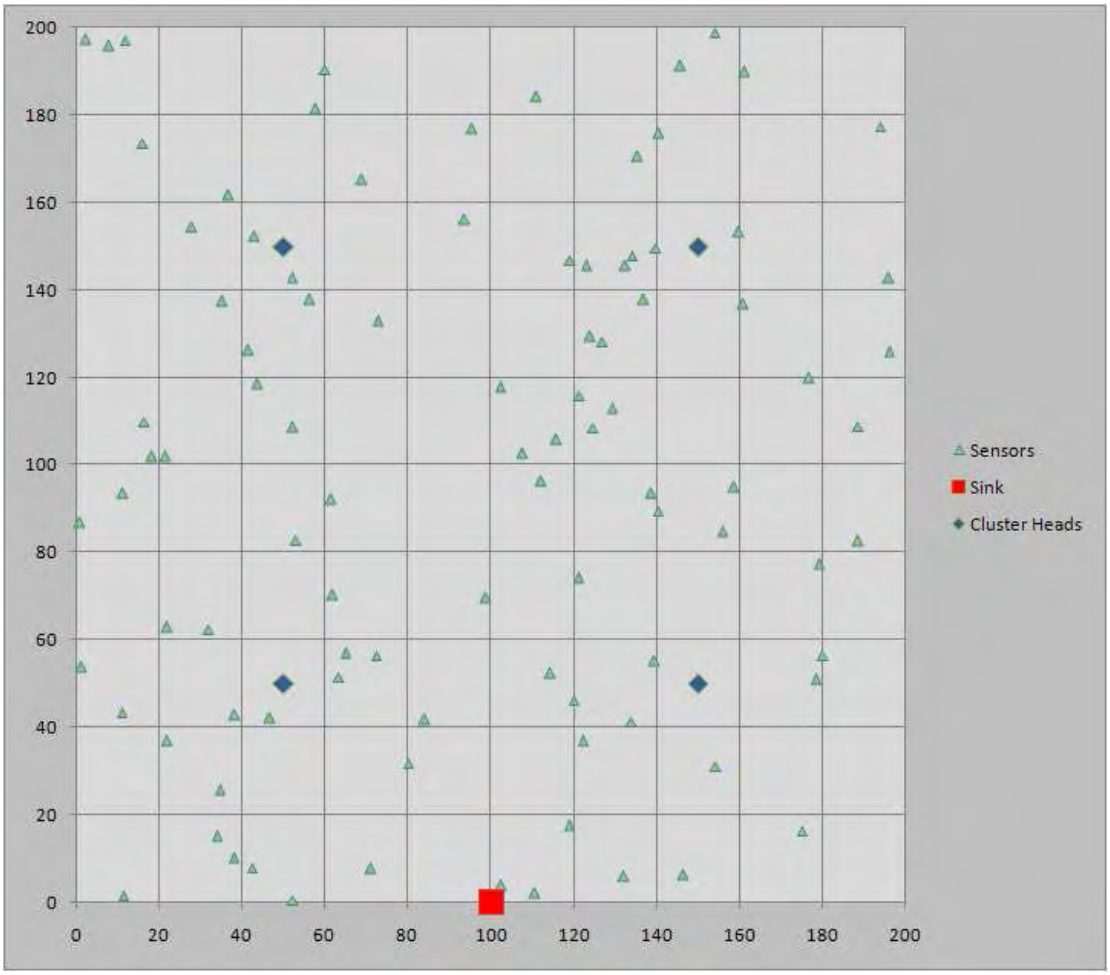
Total Packets sent from all nodes: 5927

Total Packets sent from all Cluster Heads: 257

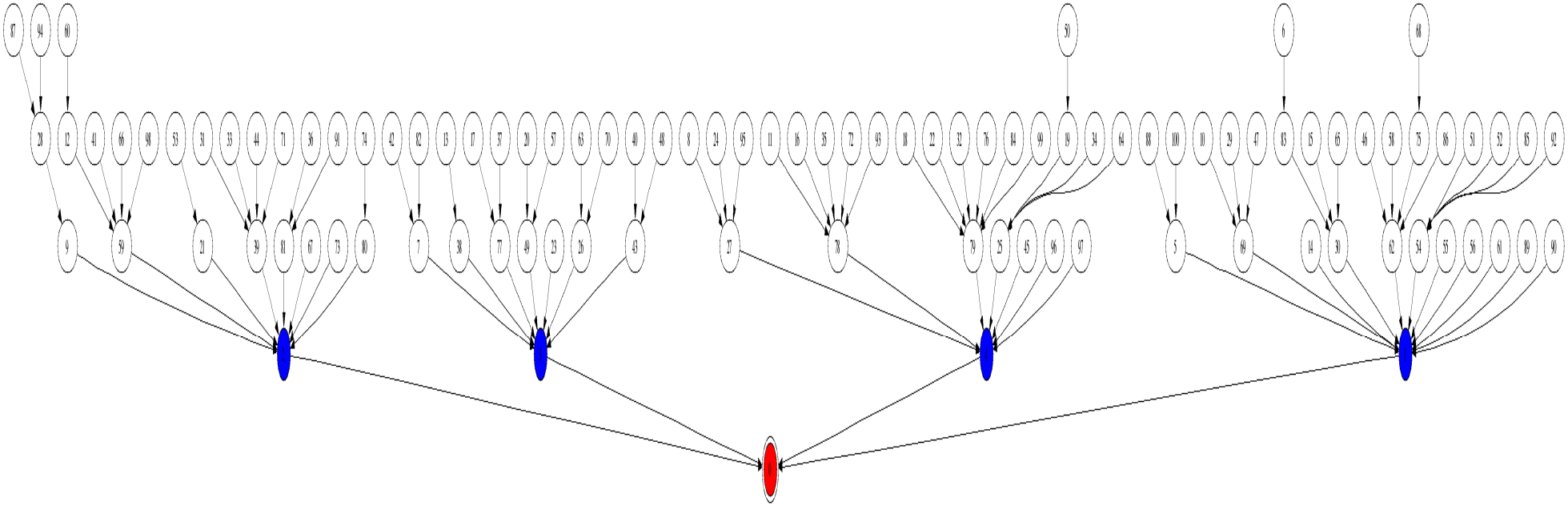
Number of Dropped Packets: 14

Success Rate: 100.0

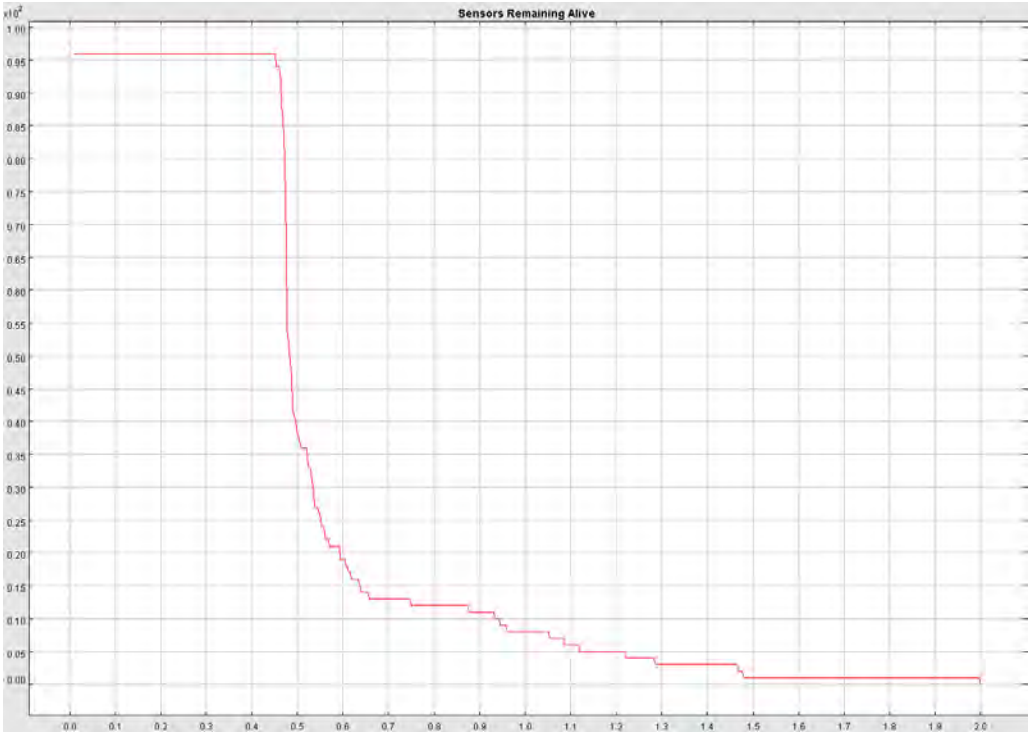
2. Δίκτυο 100 κόμβων (4 Cluster Heads) σε περιοχή 200m x 200m



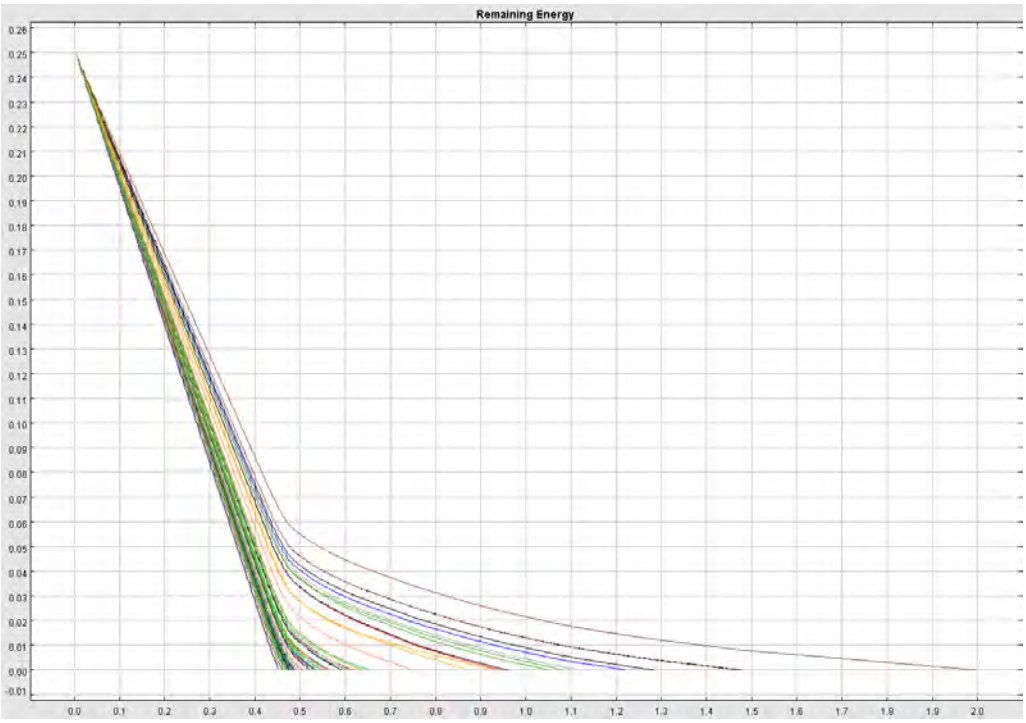
Εικόνα 76 - Τοπολογία δικτύου αισθητήρων σε πλέγμα 200x200



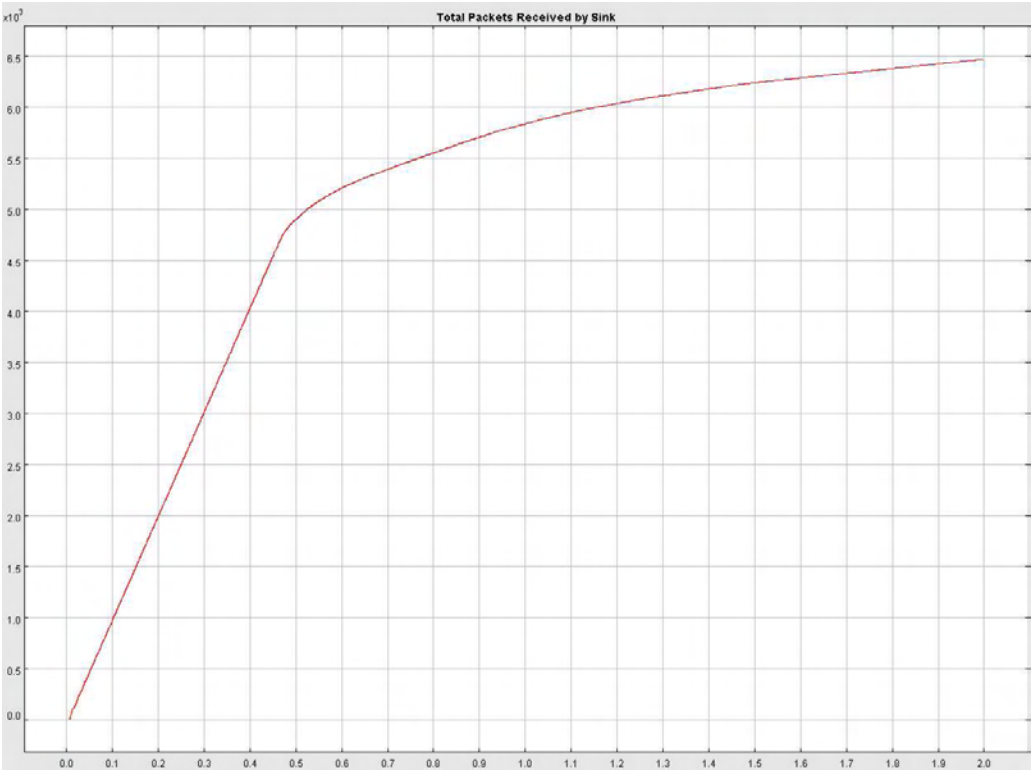
Εικόνα 77 - Γράφος Clustered δικτύου (200x200)



Εικόνα 78 - Ζωντανοί κόμβοι στο χρόνο



Εικόνα 79 - Ενέργεια κόμβων στο χρόνο



Εικόνα 80 - Αριθμός ληφθέντων πακέτων στο χρόνο

Στατιστικά Στοιχεία

94 out of 94 nodes dead at 1999.0

Total packets dropped at Application layer: 0

Total packets dropped at physical layer: 14

Average Latency on BS is: 0.00719606781881

Total Packets sent from all nodes: 5806

Total Packets sent from all Cluster Heads: 667

Number of Dropped Packets: 14

Success Rate: 99.9691024255

100 κόμβοι σε πλέγμα 100x100			
	Χρόνος ζωής (sec)	Latency (sec)	Ποσοστό Επιτυχίας (%)[1]
One-Hop	1247.0	0.0021967	97,0435
Multi-Hop (distance metric)	229.0	0.021656	98.5039
Multi-Hop (energy metric)	544.0	0.005695	99.1894
LEACH	3271.0	N/A	5.5641
Optimal Energy Clustering	601.0	0.0049097	100.0

[1] Για το LEACH δεν μετράμε ποσοστό επιτυχίας αλλά την αναλογία αριθμού πακέτων που έστειλαν οι κόμβοι προς αριθμό πακέτων που έλαβε ο σταθμός βάσης από τους Cluster Heads

100 κόμβοι σε πλέγμα 200x200			
	Χρόνος ζωής (sec)	Latency (sec)	Ποσοστό Επιτυχίας (%)
One-Hop	879,0	0,00233	40,2714
Multi-Hop (distance metric)	241.0	0.023999	71.9090
Multi-Hop (energy metric)	5464.0	0.01067	38.1002
LEACH	3229.0	N/A	2.3411
Optimal Energy Clustering	1999.0	0.00719	99.9691

Κεφάλαιο 6 - ΣΥΜΠΕΡΑΣΜΑΤΑ – ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

6.*1 Συμπεράσματα Προσομοιώσεων*

Όπως είδαμε στις προσομοιώσεις, σε ένα δίκτυο ανεπτυγμένο σε μικρή σχετικά έκταση, ακόμη και το πρωτόκολλο της απευθείας μετάδοσης μπορεί να αποτελέσει μια πολύ καλή λύση με εξαιρετικά απλή υλοποίηση. Σε πιο αραιά όμως δίκτυα, όπου αρκετοί κόμβοι μπορεί να απέχουν σημαντικά από το σταθμό βάσης, σε αποστάσεις μεγαλύτερες της ακτίνας εκπομπής τους, η αξιοπιστία του πρωτοκόλλου πέφτει σημαντικά.

Σε περιπτώσεις όπως πιο πάνω έρχεται να δώσει λύση το multi-hoping, μειώνοντας την απόσταση στην οποία πρέπει να εκπέμψει ο κάθε κόμβος. Εδώ σημαντικό ρόλο παίζει το κριτήριο επιλογής του επόμενου-γείτονα (του κόμβου που θα αναλάβει την αναμετάδοση των δεδομένων). Το να βρεθεί απλά ένας κόμβος με απόσταση από το σταθμό βάσης μικρότερη από αυτή του ενδιαφερόμενου προς εκπομπή κόμβου αποδεικνύεται ουσιαστικά ασύμφορη, με ενεργειακή απόδοση χειρότερη της απευθείας μετάδοσης, με τους κόμβους που βρίσκονται κοντά στο σταθμό βάσης να «πεθαίνουν» πολύ γρήγορα αφού καλούνται να εξυπηρετήσουν την προώθηση πολλών δεδομένων άλλων κόμβων. Η αναπαράσταση του γράφου αυτού του δικτύου είδαμε ότι έχει τη μορφή δέντρου με πολύ μεγάλο βάθος, το οποίο και επηρεάζει τη διάρκεια ζωής των κόμβων. Επιπλέον, ο μέσος όρος της καθυστέρησης άφιξης των δεδομένων είναι αυξημένος σε σχέση με την απευθείας μετάδοση αφού χρησιμοποιείται η αναμετάδοση προσθέτοντας επιπλέον χρόνο σε κάθε βήμα.

Αντίθετα, αν θέσουμε σαν κριτήριο αναμετάδοσης το ενεργειακό συμφέρον, τα αποτελέσματα που λαμβάνουμε είναι πολύ καλά, με ικανοποιητική αξιοπιστία δικτύου και πολύ καλή διάρκεια ζωής. Ο γράφος που κατασκευάζεται έχει όπως και πριν τη μορφή δένδρου, αναπτυσσόμενος όμως σε πλάτος και όχι σε βάθος. Ακόμη, αφού αποφεύγονται οι άσκοπες αναμεταδόσεις, πετυχαίνουμε και πολύ καλό μέσο όρο χρόνου άφιξης δεδομένων στο σταθμό βάσης. Ακόμη κι έτσι όμως, τα αποτελέσματα των προσομοιώσεων, ενώ έδειξαν εξαιρετική αξιοπιστία, δεν εντυπωσίασαν με την ενεργειακή απόδοση. Αυτό συνέβη λόγω του φαινομένου του overhearing, της ακούσιας δηλαδή λήψης από κάποιο κόμβο, σήματος που δεν τον αφορά. Αυτό βάζει τους κόμβους που λαμβάνουν «κατά λάθος» σήμα, σε κατάσταση κατανάλωσης ενέργειας. Δοκιμάζοντας να μειώσουμε την ευαισθησία του ασύρματου συστήματος μεγαλώνοντας το CSThreshold, είδαμε θεαματική μείωση στην κατανάλωση ενέργειας, αλλά μειωμένη αξιοπιστία λόγω αύξησης του πλήθους των συγκρούσεων. Στο πρωτόκολλο της απευθείας μετάδοσης το φαινόμενο αυτό δεν συμβαίνει γιατί οι κόμβοι δεν περιμένουν να λάβουν

δεδομένα, αλλά «ακούν» το μέσο μόνο όταν πρέπει να μεταδώσουν, προκειμένου να αποφύγουν τις συγκρούσεις.

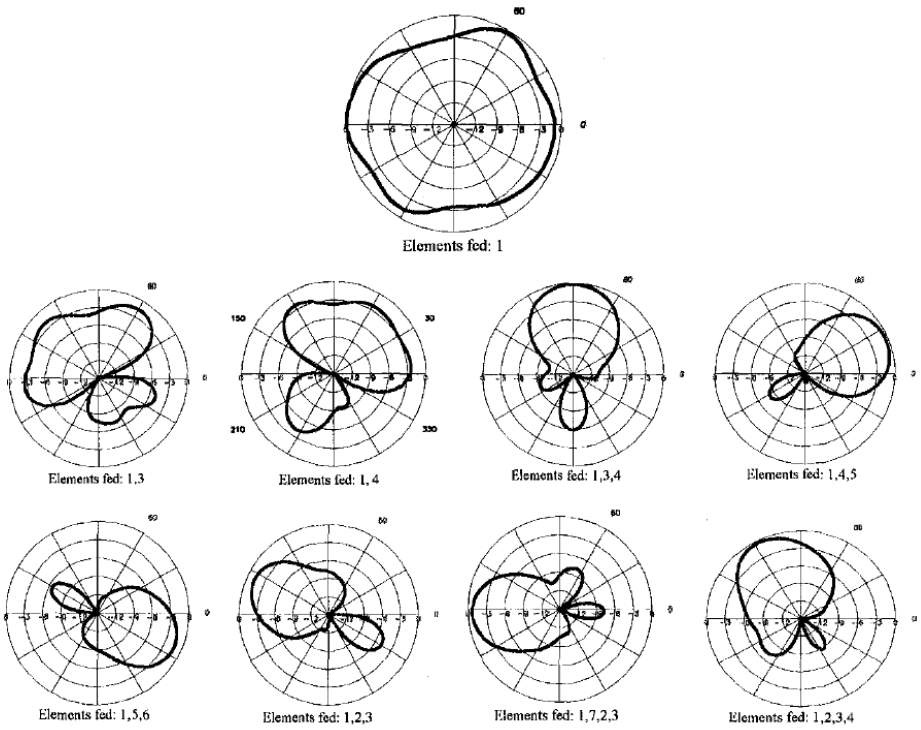
Στο πρωτόκολλο LEACH, βλέπουμε ότι γίνεται πολύ καλή εκμετάλλευση της ενέργειας του δικτύου το οποίο και έχει μεγαλύτερη διάρκεια ζωής από τις άλλες περιπτώσεις. Βέβαια το γεγονός ότι γίνεται συνάθροιση δεδομένων κάθε συστάδας πριν αυτά μεταδοθούν στο σταθμό βάσης, κάνει το πρωτόκολλο ακατάλληλο για εφαρμογές όπου ο χρόνος απόκρισης του δικτύου στα αισθητηριακά δεδομένα είναι σημαντικός. Επιπλέον σε αραιό δίκτυο όπου η απόσταση των κόμβων (πιο συγκεκριμένα των ορισμένων cluster heads, που είναι υπεύθυνοι για τη μετάδοση των δεδομένων) από το σταθμό βάσης είναι μεγάλη, βλέπουμε ότι το δίκτυο έχει μειωμένη αξιοπιστία. Αυτό συμβαίνει γιατί το LEACH δεν υποστηρίζει multi-hoping. Παρόλα αυτά, έγινε εμφανές ότι η συσταδοποίηση (clustering) αποτελεί εξαιρετική τεχνική για πολύ καλή διαχείριση και εξοικονόμηση ενέργειας σε ένα ασύρματο δίκτυο αισθητήρων.

Τέλος, οι προσομοιώσεις του εξεταζόμενου πρωτοκόλλου δείχνουν ότι έχει πολύ καλή ενεργειακή συμπεριφορά, πολύ καλό χρόνο άφιξης δεδομένων στο σταθμό βάσης και εξαιρετική αξιοπιστία, είτε πρόκειται για πυκνό είτε για αραιό δίκτυο. Ειδικότερα σε αραιό δίκτυο μεγάλης περιοχής κάλυψης (200x200) η ζωή του δικτύου είναι πολύ ικανοποιητική και επιπλέον έχουμε πολύ χαμηλό latency και σχεδόν 100% αξιοπιστία, με μόλις 4 Cluster Heads για σύνολο 100 κόμβων.

6.2 Βελτιώσεις στο πρωτόκολλο

Ένας από τους τρόπους που θα μπορούσε να μειωθεί η καταναλισκομένη ενέργεια στο δίκτυο είναι η μείωση της απαιτούμενης ισχύος εκπομπής για τη μετάδοση δεδομένων. Κάτι τέτοιο θα ήταν εφικτό με τη χρήση «έξυπνων κεραιών» (smart antennas).

Καθώς οι κόμβοι αισθητήρων μικραίνουν ολοένα σε μέγεθος θα γίνεται πιο δύσκολο να πετύχουμε αποδοτική ασύρματη επικοινωνία λόγω των παραγόντων του μεγέθους, του κόστους και των ενεργειακών περιορισμών. Το μικρό μέγεθος των κόμβων, περιορίζει και το μέγεθος των χρησιμοποιούμενων κεραιών. Αυτό σημαίνει ότι είναι επιθυμητό να μειώσουμε το μήκος κύματος και κατά συνέπεια να αυξήσουμε τη συχνότητα λειτουργίας των ασύρματων συστατικών των κόμβων. Έχουμε δει ότι είναι δυνατός ο σχεδιασμός και η κατασκευή έξυπνων κεραιών πάνω σε πλακέτες τυπωμένων κυκλωμάτων, με μέγεθος κάποιων χιλιοστών ανά στοιχείο και πολύ καλά χαρακτηριστικά κέρδους [29].



Εικόνα 81 - Λοβοί ακτινοβολίας έξυπνης κεραίας 7 στοιχείων

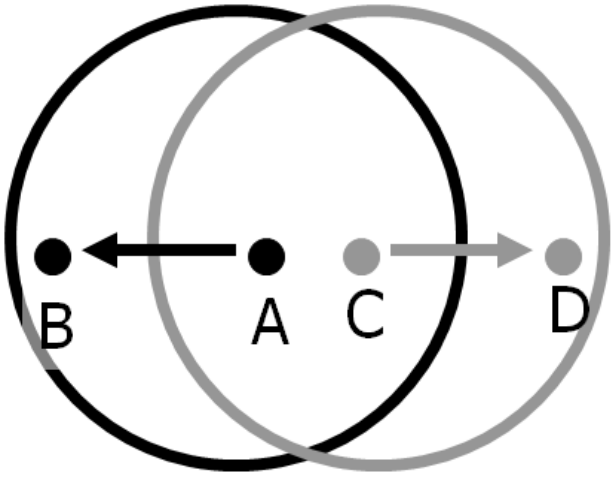
Table 1: Examples of beams at different feeding patterns

Elements Fed	Elevation peak (deg)	Azimuth peak (deg)	Directivity (dBi)
1	90	177	5.5
1,3	90	57	6.23
1,4	90	330	6.37
1,3,4	57	87	7.21
1,4,5	63	21	7.91
1,5,6	81	336	8.56
1,4,5,6	63	0	7.2

Εικόνα 82 - Κέρδη λοβών της πιο πάνω κεραίας

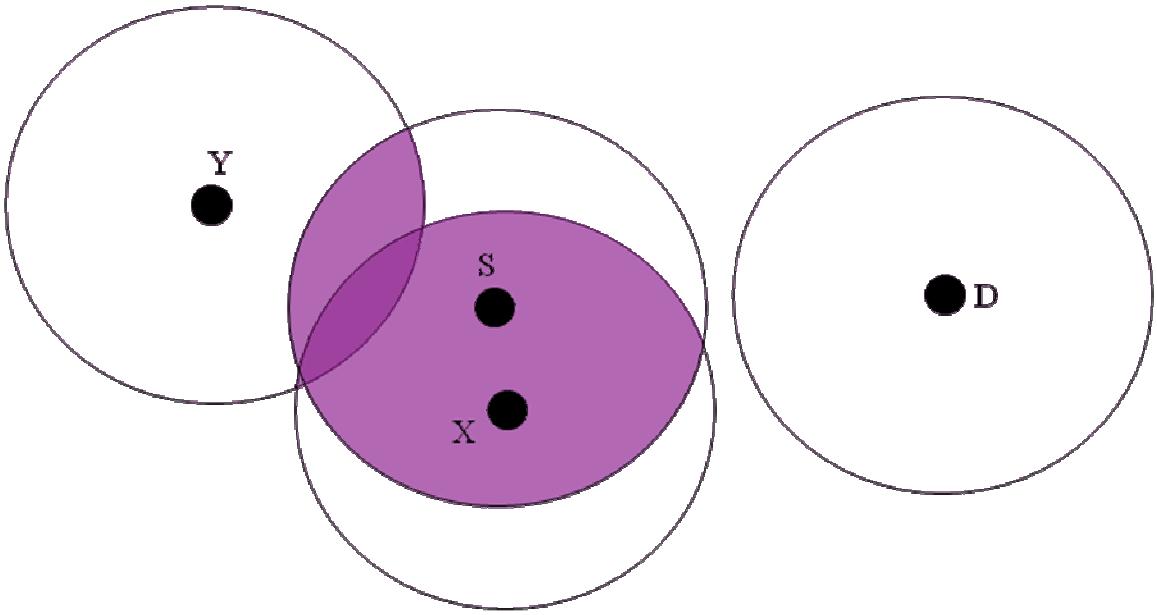
Γενικά, τα υπάρχοντα πρωτόκολλα θεωρούν τη χρήση παντοκατευθυντικών κεραιών. Αυτό μπορεί να είναι περιοριστικό, ιδιαίτερα στα multi-hop πρωτόκολλα όπου μπορεί να είναι επιθυμητή η παράλληλη εκπομπή από πολλούς κόμβους. Επιπλέον η omni εκπομπή συνεισφέρει στο φαινόμενο του overhearing. Αυτό

προκαλεί την επιπλέον κατανάλωση ενέργειας αφού το ασύρματο σύστημα μπαίνει σε κατάσταση λήψης για τους κόμβους που το σήμα έχει περάσει το κατώφλι ευαισθησίας. Το φαινόμενο αυτό, βέβαια δεν διερευνήθηκε στα πλαίσια της παρούσας εργασίας καθώς ξέφευγε από τους στόχους της. Άλλο ένα φαινόμενο που αποτελεί και αδυναμία των ασύρματων CSMA δικτύων (ειδικά τόσο πυκνών όσο τα δίκτυα αισθητήρων) είναι και το φαινόμενο του εκτεθειμένου κόμβου (exposed terminal problem) όπου λόγω του MAC πρωτοκόλλου, ενώ μπορεί να εκπέμψει προς ένα κόμβο, δεν το κάνει γιατί «ακούει» ότι το μέσο είναι κατειλημμένο από μια επικοινωνία που δεν μπορεί να τον επηρεάσει (Εικόνα 83, ο κόμβος C δεν μεταδίδει στον κόμβο D γιατί «ακούει» την εκπομπή του A στον B). Αποτέλεσμα είναι η κακή εκμετάλλευση του διαθέσιμου bandwidth.

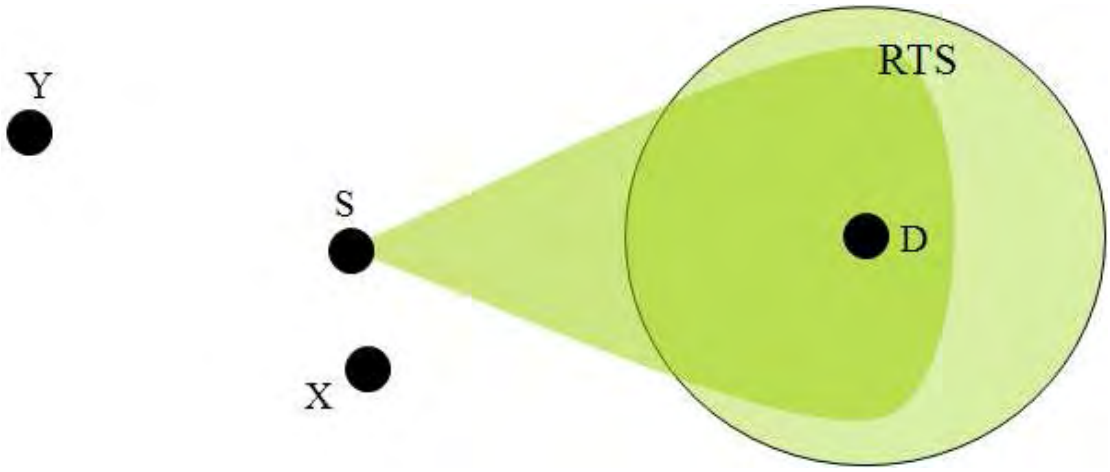


Εικόνα 83 - Exposed Terminal problem

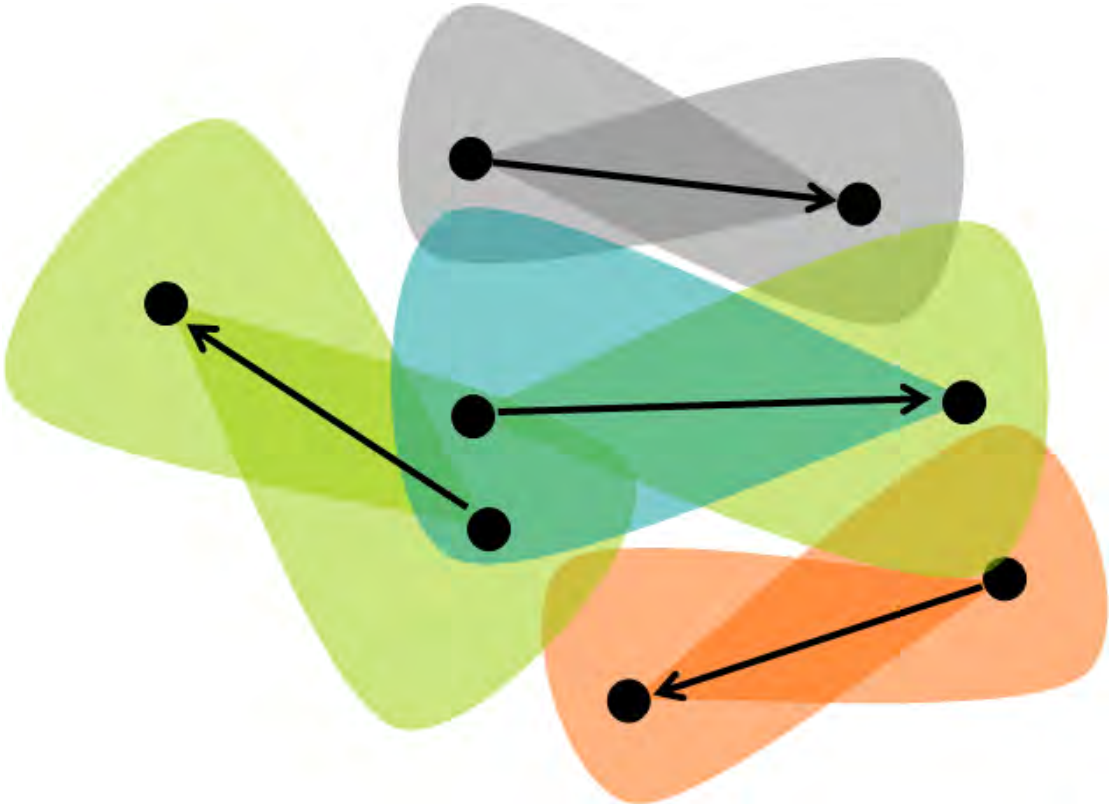
Και τα δύο προβλήματα που αναφέρθηκαν πιο πάνω μπορούν να αντιμετωπιστούν με τη χρήση έξυπνων κατευθυντικών κεραιών [28]. Ένα πιθανό σενάριο χρήσης θα ήταν το πιο κάτω:



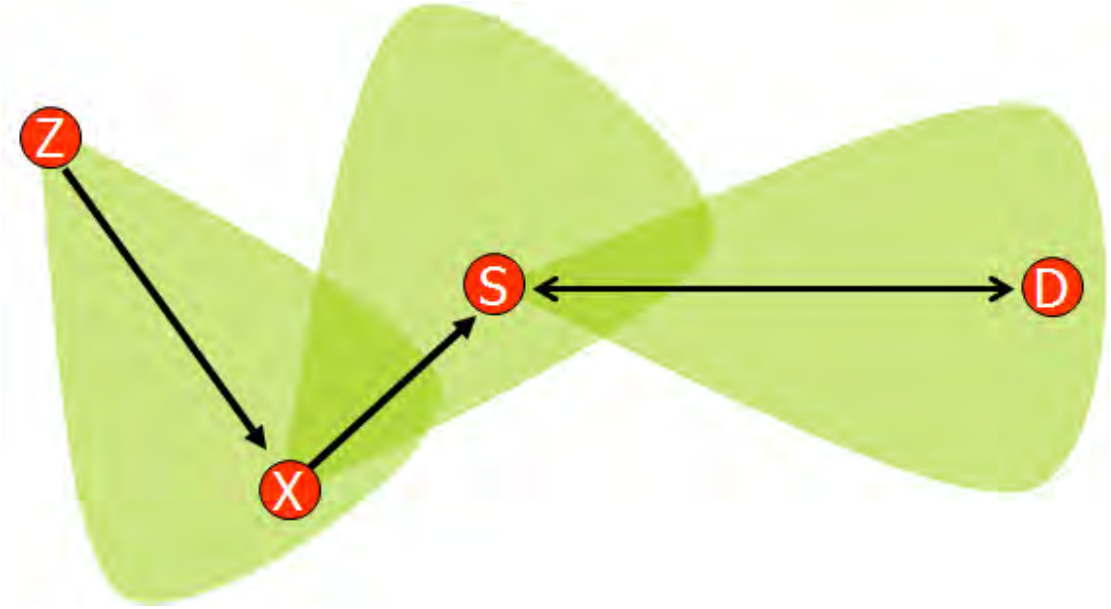
Εικόνα 84 - Idle operation Omni (οι κόμβοι δεν γνωρίζουν την κατεύθυνση του σήματος)



Εικόνα 85 - Έστω ότι ο S γνωρίζει την κατεύθυνση του D

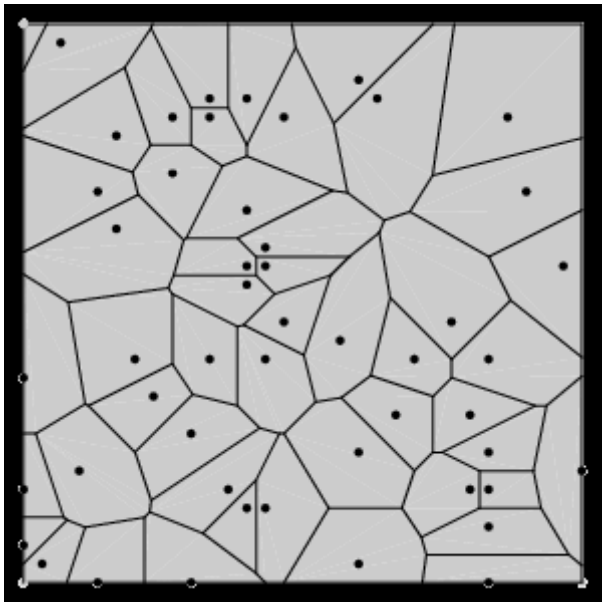


Εικόνα 86 - Επικοινωνία πολλών κόμβων (point-to-point)



Εικόνα 87 - Multihop επικοινωνία

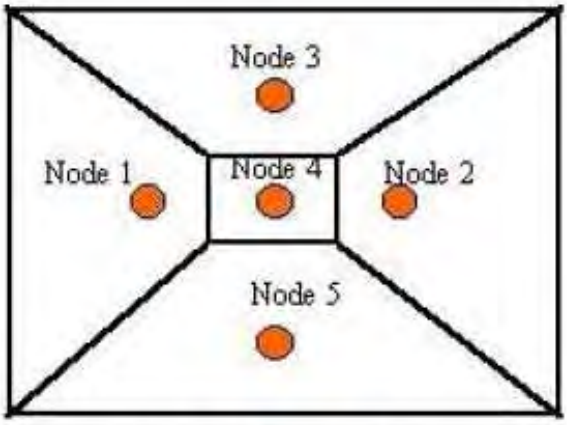
Τέλος, λόγω της τυχαίας ανάπτυξης ενός δικτύου ασύρματων αισθητήρων, μια προσεκτική μελέτη των περιοχών κάλυψης κάθε κόμβου, μπορεί να μας αποκαλύψει ότι η ίδια περιοχή μπορεί να εξυπηρετείται από δύο ή και περισσότερους κόμβους. Αυτό μπορεί να αποτελεί σπατάλη ενέργειας για το δίκτυο. Θα μπορούσαμε να δούμε σε μελλοντική εργασία την ανάπτυξη κάποιου αλγόριθμου προγραμματισμού (scheduling), ώστε από τη μια να μην υπάρχει θέμα κάλυψης της περιοχής παρατήρησης και από την άλλη να επιτυγχάνεται βέλτιστη οικονομία ενέργειας. Το κριτήριο για το πρόγραμμα ενός πιθανού αλγορίθμου, μπορεί να είναι είτε η περιοχή κάλυψης κάθε κόμβου ως προς την ασύρματη επικοινωνία (ακτίνα εκπομπής) είτε ως προς την περιοχή κάλυψης του προς παρατήρηση φαινομένου (ακτίνα αίσθησης). Σημαντικά εργαλεία σε μια τέτοια προσπάθεια μπορεί να αποτελέσουν τα διαγράμματα Voronoi[2] [25]. Παράδειγμα ενός διαγράμματος Voronoi φαίνεται στην πιο κάτω εικόνα.



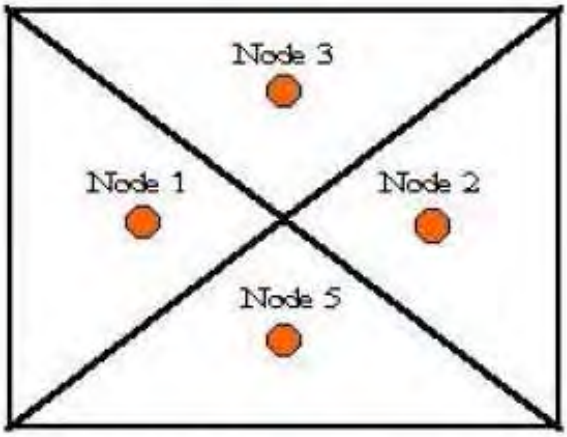
Εικόνα 88 - Διάγραμμα Voronoi

Ένα παράδειγμα χρήσης αλγόριθμου προγραμματισμού θα μπορούσε να είναι το πιο κάτω.

[2] Έστω $S = \{p_1, p_2, \ldots p_i, \ldots, p_n\}$ σύνολο σημείων σε Ευκλείδιο διδιάστατο πεδίο. Τα σημεία αυτά ονομάζονται τοποθεσίες. Το διάγραμμα Voronoi δημιουργεί περιοχές γύρω από κάθε τοποθεσία ώστε όλα τα σημεία στην περιοχή γύρω από την τοποθεσία p_i να βρίσκονται πιο κοντά στο p_i από κάθε άλλο σημείο του S.



(a) Initial Voronoi diagram



(b) New Voronoi diagram

Εικόνα 89 - Παράδειγμα scheduling

Όπως φαίνεται πιο πάνω, με την αρχική ανάπτυξη του δικτύου χρησιμοποιούνται πέντε (5) κόμβοι για την κάλυψη μιας περιοχής. Μετά την εκτέλεση όμως κάποιου αλγορίθμου κάλυψης, η ίδια περιοχή καλύπτεται από τέσσερις (4) με τον κόμβο 4 να μπαίνει σε κατάσταση αδράνειας, είτε για λόγους εφεδρείας είτε μέχρι το πρόγραμμα του αλγορίθμου να τον ενεργοποιήσει. Με βάση μια τέτοια φιλοσοφία θα μπορούσε να αναπτυχθεί αλγόριθμος που να ελέγχει τη λειτουργία ή μη κάθε κόμβου, με στόχο την κάλυψη της περιοχής παρατήρησης από τη μια και την εξοικονόμηση ενέργειας από την άλλη.

Βιβλιογραφία

[1] N.Bulusu, D.Estrin, L.Girod, and J.Heidemann "Scalable Coordination forWireless Sensor Networks: Self-Configuring Localization Systems", *Proc. of the 6th International Symposium on Communication Theory and applications (ISCTA'01), Ambleside, UK, July 2001*

[2] B.Sundararaman, U.Buy , A.Kshemkalyani "Clock synchronization for wireless sensor networks: A survey." *Ad Hoc Networks, Article in Press.*

[3] W.Ye, J. Heidemann "Medium Access control in Wireless Sensor Networks" *book chapter in "Wireless sensor networks", Kluwer 2004.*

[4] T.Dimitriou and A.Kalis "Efficient Delivery of Information in Sensor Networks Using Smart Antennas", *in S. Nikoletseas and J. Rolim (Eds.): ALGOSENSORS 2004, LNCS 3121, pp. 109–122, 2004.*

[5] W.R.Heinzelman, A.Chandrakasan, H.Balakrishman. "Energy-Efficient Communication Protocol for Wireless Microsensor Networks". IEEE, January 2000.

[6] S.Lindsey, C.S.Raghavendra."PEGASIS: Power-Efficient Gathering in Sensor Information Systems" IEEEAC Paper #242, Sept. 29 2001

[7] S.Lindsey, C.Raghavendra, K.Sivalingam."Data Gathering in Sensor Networks using the Energy*Delay Metric".

[8] A.Manjeshwar, D.P.Agrawal. "TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks".

[9] A,Manjeshwar, D.P.Agrawal. "APTTEN: A Hybrid Protocol for Efficient Routing and Complrehensive Information Retrieval in Wireless Sensor Networks".

[10] K.Akkaya, M.Younis. "A survey on Routing Protocols for Wireless Sensor Networks".

[11] J.N. Al-Karaki, A.E.Karnal. "Routing Techniques in Wireless Sensor Networks: A Survey".

[12] Y.Yu, R.Govindan, D.Estrin. “Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks”, UCLA Computer Science Department Technical Report, 08/14/2001.

[13] Y.Xu, J.Heidemann, D.Estrin. “Geography-informed Energy Conservation for Ad Hoc Routing”, Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking 2001.

[14] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks", in the Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00), Boston, MA, August 2000.

[15] Aivo Anier, ’’Routing in Wireless Sensor Networks”, Tallinn University of Technology, Department of Computer Science, General Informatics, 2005

[16] K. Sohrabi, et al., "Protocols for self-organization of a wireless sensor network,” IEEE PersonalCommunications, Vol. 7, No. 5, pp. 16-27, October 2000.

[17] Chalermek Intanagonwiwat, “Directed Diffussion: An Application-specific and Data-centric Communication Paradigm for Wireless Sensor Networks”, University of Southern California, December 2002

[18] W. Heinzelman, J. Kulik, and H. Balakrishnan, “Adaptive protocols for information dissemination in wireless sensor networks,” in the Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom’99), Seattle, WA, August 1999.

[19] S. Hedetniemi and A. Liestman, “A survey of gossiping and broadcasting in communication networks,” Networks, Vol. 18, No. 4, pp. 319-349, 1988.

[20] T. He et al., “SPEED: A stateless protocol for real-time communication in sensor networks,” in the Proceedings of International Conference on Distributed Computing Systems, Providence, RI, May 2003.

[21] D. Bertsekas, R. Gallager “Data Networks”. *Prentice-Hall, 1987, pp. 297-421*

[22] Jamil Ibriq and Imad Mahgoub "Cluster-Based Routing in Wireless Sensor Networks: Issues and Challenges", SPECTS 04

[23] N. Vlajic and D. XIA, "Wireless Sensor Networks: To Cluster or Not To Cluster?", Proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)

[24] Jichuan Zhao, Ahmet T. Erdogan, "A Novel Self-organizing Hybrid Network Protocol for Wireless Sensor Networks", Proceedings of the First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06)

[25] Marcos Augusto M. Vieira, Luiz Filipe M. Vieira, Linnyer B. Ruiz, Antonio A.F. Loureiro, Antonio O. Fernandes, Jose Marcos S. Nogueira, "Scheduling Nodes in Wireless Sensor Networks: A Voronoi Approach", Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN'03)

[26] Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan and Honghai Zhang, "J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks".

[27] http://www.j-sim.org

[28] *Dee Leang, Antonis Kalis,* "Smart SensorDVBs: Sensor Network Development Boards with Smart Antennas"

[29] Kalis et al.: "A Printed Circuit Switched Array Antenna for Indoor Communications", IEEE Transactions on Consumer Electronics, Vol. 46, No, 3, AUGUST 2000

[30] Παναγιώτης Κ. Κίκιρας, «Δίκτυα Αισθητήρων για Διάχυτο Υπολογισμό», Διδακτορική Διατριβή

[31] Hannes Stratil, "Voronoi supported communication in Wireless ad-hoc Networks"

[32] Seapahn Meguerdichian, Farinaz Koushanfar, Miodrag Potkonjak, Mani B. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Networks"

[33] Shugong Xu and Tarek Saadawi, “Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks?”, IEEE Communications Magazine, June 2001

[34] Wei Ye and John Heidemann, “Medium Access Control in Wireless Sensor Networks”, USC/ISI TECHNICAL REPORT ISI-TR-580, OCTOBER 2003

[35] WeiZhao Wang, Wen-Zhan Song, Xiang-Yang Li, Kousha Moaveni-Nejad, “Optimal Cluster Association in Two-Tiered Wireless Sensor Networks”

[36] Tamer Nadeem, Ashok Agrawala, “Performance of IEEE 802.11 based Wireless Sensor Networks in Noisy Environments”

[37] Qingjiang Tian, Seema Bandyopadhyay, and Edward J. Coyle, “Effect of Directional Antennas on Spatiotemporal Sampling in Clustered Sensor Networks”

[38] Antonis Kalis and Tassos Dimitriou, “Fast routing in wireless sensor networks using directional transmissions”

[39] Jaekyu Cho, Jeongkeun Lee, Taekyoung Kwon and Yanghee Choi, “Directional Antenna at Sink (DAaS) to prolong network lifetime inWireless Sensor Networks”

[40] Matt Welsh and Geoff Mainland, “Programming Sensor Networks Using Abstract Regions”

[41] Chris Savarese, Jan M. Rabaey, Jan Beutel, “LOCATIONING IN DISTRIBUTED AD-HOCWIRELESS SENSOR NETWORKS”

[42] Xiaojiang Du, Fengjing Lin, “Designing Efficient Routing Protocol for Heterogeneous Sensor Networks”

[43] http://www.acsg.mcmaster.ca/merizzn/

[44] TaeSuk Kim, Hyuk Lim, Jennifer C. Hou, “Improving Spatial Reuse through Tuning Transmit Power, Carrier Sense Threshold, and Data Rate in Multihop Wireless Networks”, MobiCom’06

ΠΑΡΑΡΤΗΜΑ Α – Κώδικας TCL (simulation scripts)

One-Hop

```
##############################################################
#
# Sample tcl file of a wireless sensor network simulation.
#
##############################################################

source "../../../../test/include.tcl"

cd [mkdir -q drcl.comp.Component /oneHop80211]

# TOTAL number of nodes (sensor nodes + target nodes)
set node_num 103

# Number of TARGET nodes ONLY
set target_node_num 2
# Hence, number of SENSORS = node_num - target_node_num

set sink_id 0
set sinkX 50.0
set sinkY 0.0
set sinkZ 0.0

#Maximum simulation time
set maxsimtime 15000

#FOR FILE WRITTING
set count 0

# create the sensor channel
mkdir drcl.inet.sensorsim.SensorChannel chan

# Capacity of the sensor channel is total number of nodes (sensors +
targets)
# make simulation for $node_num nodes
! chan setCapacity $node_num

# create the propagation model
mkdir drcl.inet.sensorsim.SeismicProp seismic_Prop
! seismic_Prop setD0 0.2

# Grid dimensions
set gx 100
set gy 100

# create the sensor node position tracker
mkdir drcl.inet.sensorsim.SensorNodePositionTracker nodetracker
                              # maxX  minX maxY minY
! nodetracker setGrid $gx 0.0 $gy 0.0

# connect the sensor channel to the sensor node position tracker
connect chan/.tracker@ -and nodetracker/.channel@

# create the wireless channel
```

```
mkdir drcl.inet.mac.Channel channel

# Capacity of the wireless channel is number of sensors and sinks
ONLY
# which is equal to $node_num - $target_node_num
! channel setCapacity [expr $node_num - $target_node_num]

# create the node position tracker
mkdir drcl.inet.mac.NodePositionTracker tracker
#the dx and dy below represent 'how far' my signal travels
#so in this case any node located in my 100x100m grid will hear
#what a sensor broadcasts
#                 maxX minX maxY minY  dX   dY
! tracker setGrid $gx 0.0 $gy 0.0 $gx $gy

connect channel/.tracker@ -and tracker/.channel@

#*****************
#     In order to graph total # of sensors still
#     alive we created a new component that will
#     have a connected port to a plotter
mkdir drcl.inet.sensorsim.AliveSensors    liveSensors
set numNodesPlot_ [mkdir drcl.comp.tool.Plotter .numNodesPlot]
connect -c liveSensors/.plotter@ -to $numNodesPlot_/0@0



#****************************************************
# FOR THE SINKs ONLY, do the following
# SINKs have only a network protocol stack
for {set i 0} {$i < [expr $sink_id + 1]} {incr i} {
      puts "create sink $i"
      set node($i) [mkdir drcl.comp.Component n$i]

      cd n$i
      mkdir drcl.inet.sensorsim.OneHop.SinkAppOH app
      ! app setNid $i
      ! app setSinkNid $sink_id
      ! app setCoherentThreshold 1000.0

      # connect the sensor application to the wireless agent
      # so that sinks can send through the wireless network protocol
stack
      # create wireless agent layers
      mkdir drcl.inet.sensorsim.WirelessAgent wireless_agent

      # connect the sensor application to the wireless agent
      # so that sensors can send through the wireless network
protocol stack
      connect app/down@ -to wireless_agent/up@

      # connect the wireless agent to the sensor application
      # so that sensors can receive thru the wireless network
protocol stack
      connect wireless_agent/.toSensorApp@ -to
app/.fromWirelessAgent@

      mkdir drcl.inet.mac.LL ll
      mkdir drcl.inet.mac.ARP arp
      mkdir drcl.inet.core.queue.FIFO queue
      mkdir drcl.inet.mac.Mac_802_11 mac
```

```
    # added 09-04-04
    ! mac disable_PSM

    mkdir drcl.inet.mac.WirelessPhy wphy


    mkdir drcl.inet.mac.FreeSpaceModel propagation
    mkdir drcl.inet.mac.MobilityModel mobility

    set PD [mkdir drcl.inet.core.PktDispatcher      pktdispatcher]
  set RT [mkdir drcl.inet.core.RT                 rt]
    set ID [mkdir drcl.inet.core.Identity           id]

    ! pktdispatcher setRouteBackEnabled 1

    $PD bind $RT
    $PD bind $ID

    # present if using 802.11 power-saving mode
    connect mac/.energy@ -and wphy/.energy@

    connect wphy/.mobility@    -and mobility/.query@
    connect wphy/.propagation@ -and propagation/.query@

    connect mac/down@ -and wphy/up@
    connect mac/up@   -and queue/output@

    connect ll/.mac@ -and mac/.linklayer@
    connect ll/down@ -and queue/up@
    connect ll/.arp@ -and arp/.arp@

    connect -c pktdispatcher/0@down -and ll/up@

    set nid $i

    ! arp setAddresses  $nid $nid
    ! ll  setAddresses  $nid $nid
    ! mac setMacAddress $nid
    ! wphy setNid       $nid
    ! mobility setNid   $nid
    ! id setDefaultID   $nid

    ! queue setMode      "packet"
    ! queue setCapacity  40

    # disable ARP
    ! arp setBypassARP  [ expr 2>1]


    ! mac setRTSThreshold 10

    connect mobility/.report@ -and /oneHop80211/tracker/.node@

    connect wphy/down@ -to /oneHop80211/channel/.node@

    ! /oneHop80211/channel attachPort $i [! wphy getPort .channel]

    #                                     maxX maxY  maxZ minX minY
minZ dX   dY   dZ
```

```
    ! mobility setTopologyParameters $gx $gy 0.0 $gx $gy 0.0 $gx $gy
0.0

      ! mac  disable_MAC_TRACE_ALL

      connect -c  wireless_agent/down@ -and pktdispatcher/1111@up

      cd ..
}

#*******************************************************************
*
# FOR THE SENSORS ONLY , do the following
#
for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {

      puts "create sensor $i"
      set node($i) [mkdir drcl.comp.Component n$i]

      cd n$i

      mkdir drcl.inet.sensorsim.OneHop.OneHopApp app
      ! app setNid $i
      ! app setSinkNid $sink_id
      ! app setCoherentThreshold 1000.0
      ! app setIs_uAMPS 1

      # create nodes
      mkdir drcl.inet.sensorsim.SensorAgent agent

      ! agent setDebugEnabled 0

      # create sensor physical layers
      mkdir drcl.inet.sensorsim.SensorPhy phy
      ! phy setRxThresh 0.0
      ! phy setDebugEnabled 0

      # create mobility models
      mkdir drcl.inet.sensorsim.SensorMobilityModel mobility

      ! phy setNid $i
      ! phy setRadius [expr sqrt($gx*$gx+$gy*$gy)]
      #! phy setRadius 100.0

      # connect phyiscal layers to sensor agents so that nodes can
receive
      connect phy/.toAgent@ -to agent/.fromPhy@

      # connect sensor agent and sensor application
      connect agent/.toSensorApp@ -to app/.fromSensorAgent@

      # connect the sensor channel to the nodes so that they can
receive
      ! /oneHop80211/chan attachPort $i [! phy getPort .channel]

      # connect the nodes to the propagation model
      connect phy/.propagation@ -and
/oneHop80211/seismic_Prop/.query@

      ! mobility setNid $i
```

```
      # create wireless agent layers
      mkdir drcl.inet.sensorsim.WirelessAgent wireless_agent

      # connect the sensor application to the wireless agent
      # so that sensors can send through the wireless network
protocol stack
      connect app/down@ -to wireless_agent/up@

      # connect the wireless agent to the sensor application
      # so that sensors can receive thru the wireless network
protocol stack
      connect wireless_agent/.toSensorApp@ -to
app/.fromWirelessAgent@

      mkdir drcl.inet.mac.LL ll
      mkdir drcl.inet.mac.ARP arp
      mkdir drcl.inet.core.queue.FIFO queue
      mkdir drcl.inet.mac.Mac_802_11 mac

      # added 09-04-04
      ! mac disable_PSM

      #*************
      #Required to let the MAC802.11 layer
      #be in charge of maintaining the
      #radio modes and energy levels.
      ! mac setIs_uAMPS 1

      mkdir drcl.inet.mac.WirelessPhy wphy
      mkdir drcl.inet.mac.FreeSpaceModel propagation

      set PD [mkdir drcl.inet.core.PktDispatcher      pktdispatcher]
      set RT [mkdir drcl.inet.core.RT                 rt]
      set ID [mkdir drcl.inet.core.Identity           id]

      ! pktdispatcher setRouteBackEnabled 1

      $PD bind $RT
      $PD bind $ID

      #**************
      #NICHOLAS
      # create route configuration request for testing
      #this is to define the interfaces. So in this case each sensor
      #only has 1 interface (hence array size 1) and its eth0.
      #another example is (which has 3 interfaces 0, 2, and 4:
      #set ifs [java::new drcl.data.BitSet [java::new {int[]} 3 {0 2
4}]]

      set ifs [java::new drcl.data.BitSet [java::new {int[]} 1 {0}]]
      set base_entry [java::new drcl.inet.data.RTEntry $ifs]

      set key [java::new drcl.inet.data.RTKey $i 0 -1]
      set entry_ [!!! [$base_entry clone]]

      ! rt add $key $entry_
      #******************************

      # present if using 802.11 power-saving mode
      connect mac/.energy@ -and wphy/.energy@
```

```
      #****************************
      connect app/.energy@ -and wphy/.appEnergy@
      mkdir drcl.inet.sensorsim.CPUAvr cpu

      connect app/.cpu@ -and cpu/.reportCPUMode@
      connect cpu/.battery@ -and wphy/.cpuEnergyPort@
      #****************************

      connect wphy/.mobility@    -and mobility/.query@
      connect wphy/.propagation@ -and propagation/.query@

      connect mac/down@ -and wphy/up@
      connect mac/up@   -and queue/output@


      connect ll/.mac@ -and mac/.linklayer@
      connect ll/down@ -and queue/up@
      connect ll/.arp@ -and arp/.arp@

      connect -c pktdispatcher/0@down -and ll/up@

      set nid $i

      ! arp setAddresses  $nid $nid
      ! ll  setAddresses  $nid $nid
      ! mac setMacAddress $nid
      ! wphy setNid       $nid
      ! id setDefaultID   $nid

      ! queue setMode      "packet"
      ! queue setCapacity  40

      # disable ARP
      ! arp setBypassARP  [ expr 2>1]

      ! mac setRTSThreshold 40

      connect mobility/.report@ -and /oneHop80211/tracker/.node@
      connect wphy/down@ -to /oneHop80211/channel/.node@

      ! /oneHop80211/channel attachPort $i [! wphy getPort .channel]

      #                                maxX maxY  maxZ minX minY minZ
dX  dY    dZ
    ! mobility setTopologyParameters $gx $gy 0.0 $gx $gy 0.0 $gx $gy
0.0

      ! mac  disable_MAC_TRACE_ALL

      connect -c  wireless_agent/down@ -and pktdispatcher/1111@up

      cd ..
}

#*************************************************************
# FOR THE TARGET NODES ONLY , do the following
if { $target_node_num == 0 } {
      puts "No target agents .... "
} else {
```

```
      for {set i [expr $node_num - $target_node_num]} {$i <
$node_num} {incr i} {
            puts "create target $i"

            set node$i [mkdir drcl.comp.Component n$i]

            cd n$i

            # create target agents
            mkdir drcl.inet.sensorsim.TargetAgent agent
            ! agent setBcastRate 1.0
            ! agent setSampleRate 1.0

            # create sensor physical layers
            mkdir drcl.inet.sensorsim.SensorPhy phy
            ! phy setRxThresh 0.0
            ! phy setNid $i
            #! phy setRadius 100.0
            ! phy setRadius [expr sqrt($gx*$gx+$gy*$gy)]

            ! phy setDebugEnabled 0

            # create mobility models
            mkdir drcl.inet.sensorsim.SensorMobilityModel mobility

            # connect target agents to phy layers so that nodes can
send
            connect agent/down@ -to phy/up@

            # connect phy layers to sensor channel so that nodes can
send
            connect phy/down@ -to /oneHop80211/chan/.node@

            # connect the nodes to the propagation model
            connect phy/.propagation@ -and
/oneHop80211/seismic_Prop/.query@

            ! mobility setNid $i

            # set the topology parameters
            ! mobility setTopologyParameters $gx $gy 0.0 $gx $gy 0.0

            cd ..
      }
}

#*********************************************************************
# for SENSORs and TARGETs only. Not SINKs
for {set i [expr $sink_id + 1]} {$i < $node_num} {incr i} {
      # connect the mobility model of each node to the node position
tracker
      connect n$i/mobility/.report_sensor@ -and
/oneHop80211/nodetracker/.node@
      connect n$i/phy/.mobility@ -and n$i/mobility/.query@
}


#**********************************************
#Positioning
#
```

```
# set the position of sink nodes args=> (speed(m/sec),
xCoord,yCoord,zCoord
! $node(0)/mobility setPosition 0.0 $sinkX $sinkY $sinkZ


! n101/mobility setPosition 0.0 0.001 [expr $gy-0.001] 0.0
! n102/mobility setPosition 0.0 [expr $gx-0.001] [expr $gy-0.001] 0.0


# for the sensors They will be randomly placed on the grid (2D only)
# set the position of sensor nodes args=> (speed(m/sec),
xCoord,yCoord,zCoord
for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
   ! n$i/mobility setPosition 0.0 [expr rand()*$gx] [expr rand() *
$gy] 0.0
}

# for the target we can include random mobility They will be randomly
# placed on the grid (2D only)
#set the position of target nodes args=> (speed(m/sec),
xCoord,yCoord,zCoord
#for {set i [expr $node_num - $target_node_num]} {$i < $node_num}
{incr i} {
#   ! n$i/mobility setPosition 0.0 [expr rand()*$gx] [expr rand() *
$gy] 0.0
#}

#****************************************************
#fmakis
#make the sensors aware of the position of the sink in case it is not
0,0,0 (not taken into account in previous implementations)
for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
   ! n$i/app setSinkLocation $sinkX $sinkY $sinkZ
}


#***************************************************
#routeInfo() to execute do:
#     script "routeInfo" -at 0.35 -period 4.0 -on $sim
proc routeInfo { } {
      global sim n1
      puts "Current Route Table\n [! n1/rt info]"
}


#***************************************************
#Output remaining energy levels of the sensors to a plotter
set plot_ [mkdir drcl.comp.tool.Plotter .plot]
for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
   connect -c n$i/app/.plotter@ -to $plot_/$i@0
}

#***************************************************
#Plotters for the SINK node

#Graph # 1
#To plot the total number of received packets at the sink
set sinkPlot1_ [mkdir drcl.comp.tool.Plotter .sinkPlot1]
```

```
connect -c n0/app/.PacketsReceivedPlot@ -to $sinkPlot1_/0@0

#Graph # 2
#Calculate the avg latency when the sink finally receives it
set sinkPlot2_ [mkdir drcl.comp.tool.Plotter .sinkPlot2]
connect -c n0/app/.latencyPlot@ -to $sinkPlot2_/0@0


#Graph # 3
#plot the actual phenomena being sensed.
! n$sink_id/app createSnrPorts $node_num $target_node_num
set sinkPlot3_ [mkdir drcl.comp.tool.Plotter .sinkPlot3]

for {set i 0} {$i < $target_node_num} {incr i} {

   connect -c n$sink_id/app/.snr$i@ -to $sinkPlot3_/$i@$i
   if { $testflag } {
       attach -c $testfile/in@ -to n$sink_id/app/.snr$i@
   }
}
#************************************************
#wsnLoop()
#
#     This method is called periodically to check
#     if the simulation should continue or not. If
#     all nodes are dead or simtime>maxsimtime then stop the
simulator and
#     display the cummulative statistics otherwise keep running

proc wsnLoop { } {
      global sim node_num node sink_id target_node_num maxsimtime

      #reset variables
      set live_sensors 0
      set dead_sensors 0
      set total_packets 0
      set dropped_packets 0

      #check how many are still alive
      for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
            if { [! $node($i)/app isSensorDead] == 0 } {
                  incr live_sensors
            } else {
                  incr dead_sensors
            }

      }
      script [! liveSensors setLiveNodes $live_sensors]
      script [! liveSensors updateGraph]

      #display statistics if they are all dead.
      if { $dead_sensors == [expr $node_num - $target_node_num - 1]
|| [! $sim getTime] > $maxsimtime} {
         #puts "All nodes dead at [! $sim getTime]"
         puts "$dead_sensors nodes dead at [! $sim getTime]"
         $sim stop
         puts "----------------------------------------------"
         puts "Simulation Terminated\n"
         puts "Results:"
         puts "Base Station Received [! n0/app getTotalINPackets]"
```

```
         puts "Collisions at Base Station: [! n0/mac getCollision]"

           for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
            set curr_packets [! n$i/app geteID]
            puts "Sensor$i Sent $curr_packets Packets to BS"
            set total_packets [expr $total_packets + $curr_packets]
         }

         set app_dropped [! n1/app getDropped_packets]
         set wphy_dropped [! n1/wphy getDropped_packets]
         #set mac_dropped [! n1/mac getDropped_packets]

         puts "Total packets dropped at Application layer:
$app_dropped"
         puts "Total packets dropped at physical layer:
$wphy_dropped"
         #puts "Drops due to collisions (discovered at MAC layer:
$mac_dropped"
         #set dropped_packets [expr $app_dropped + $wphy_dropped +
$mac_dropped]
         set dropped_packets [expr $app_dropped + $wphy_dropped]
            puts "Average Latency on BS is: [! n0/app getAvgLatency]"
         puts "Total Packets sent from all nodes: $total_packets"
         puts "Number of Dropped Packets: $dropped_packets"
         puts "Success Rate: [expr ([! n0/app getTotalINPackets].0 /
$total_packets.0) * 100]"
      }
}


#***********************************************
#sensorLocPrintOut()
#     Goes throught all sensors and prints their
#     (X,Y,Z) Coordinates
proc sensorLocPrintOut { } {
      global sink_id node_num target_node_num target_node_num
      for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
            script [! n$i/app printNodeLoc]
      }
}


#***********************************************
#Output sensor location and status
#to a file for GUI to read from
proc file_output { } {
   global sink_id node_num count target_node_num

   #open a file for writting
   set filename "sensorInfo$count.log"
   set out [open $filename w]

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
     #get its x and y location
     set Xloc [! n$i/app getX]
     set Yloc [! n$i/app getY]

     #get its energy
```

```
      set status [expr [! n$i/app isSensorDead] - 0]

      #write it to the file
      puts $out "$Xloc $Yloc $status"
   }
   incr count

   close $out
}


#**********************************************
#Output sensor location and status
#to a file for GUI to read from
proc energy_dist { } {
   global sink_id node_num count target_node_num

   #open a file for writting
   set filename "energyDistrNode.log"
   set out [open $filename w]

   puts $out "TxCost     RxCost    IdleCost    SleepCost    CPUcost"

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
      #collect the data
      set txCost [! n$i/wphy getRadioTotalTX]
      set rxCost [! n$i/wphy getRadioTotalRX]
      set idleCost [! n$i/wphy getRadioTotalidle]
      set sleepCost [! n$i/wphy getRadioTotalsleep]
      set cpuCost [! n$i/wphy getTotalCPU]

      #write it to the file
      puts $out "$txCost $rxCost $idleCost $sleepCost $cpuCost"
   }

   close $out
}



#**********************************************
#Strictly used for validation purposes. This method
#can be called to get the energy at any random
#point of a particular node.
proc getCurrEnergy { } {

   puts "\n\n"
   global sink_id node_num count
   for {set i [expr $sink_id + 1]} {$i < $node_num} {incr i} {
      puts "-----------------------------------------------"

      puts "Sensor$i radio in Idle mode used: [! n$i/wphy
getRadioTotalidle] joules"
      puts "Sensor$i radio in Rx mode used: [! n$i/wphy
getRadioTotalRX] joules"
      puts "Sensor$i radio in Tx mode used: [! n$i/wphy
getRadioTotalTX] joules\n"

      puts "Sensor$i Radio was idle for: [! n$i/wphy
getTotalRadioIdleTime] sec"
```

```
      puts "Sensor$i Radio was Tx for: [! n$i/wphy
getTotalRadioActiveTime] sec"
      puts "Sensor$i Radio was Rx for: [! n$i/wphy
getTotalRadioRxTime] sec\n"

      puts "Sensor$i CPU active used: [! n$i/wphy getTotalCPUactive]
joules"
      puts "Sensor$i CPU sleep used: [! n$i/wphy getTotalCPUsleep]
joules\n"

      puts "Sensor$i CPU was active for: [! n$i/wphy
getTotalCPUactiveTime] sec"
      puts "Sensor$i CPU was sleep for: [! n$i/wphy
getTotalCPUsleepTime] sec\n"


      puts "Sensor$i CPU Total used: [! n$i/wphy getTotalCPU] joules
\n"

      puts "Sensor$i has [! n$i/wphy getRemEnergy] joules remaining"

   }
}

#-------------------------

#************************************************
puts "Simulation begins...\n"
set sim [attach_simulator .]
$sim stop


#******************start the sink************************
script {run n0} -at 0.001 -on $sim

#***************start the sensors************************
for {set i [expr $sink_id + 1]} {$i < $node_num} {incr i} {
      script puts "run n$i" -at 0.1 -on $sim
}

#*********print out all the node locations**************
script "sensorLocPrintOut" -at 0.002 -on $sim

#*******Check if Sensor Status*************************
script "wsnLoop" -at 1.0 -period 2.0 -on $sim


#************For Matlab plotting************************
#script "file_output" -at 200.0 -period 150.0 -on $sim

#************One-time Capture of where energy went******
#script "energy_dist" -at 3499.0 -on $sim

#the following line is only used in validation!
#script "getCurrEnergy" -at 10.0 -period 10.0 -on $sim

$sim resumeTo [expr $maxsimtime+5]
```

Multi-Hop

```
# ##############################################################
# Sample tcl file of a wireless sensor network simulation.
# This simulation uses a multi-hop method to communicate
# information back to the base station. The MAC layer is based
# on the 802.11 protocol.
# ############################################################

source "../../../../test/include.tcl"



cd [mkdir -q drcl.comp.Component /multiHop80211]

# TOTAL number of nodes (sensor nodes + target nodes + cluster heads)
set node_num 103

# Number of TARGET nodes ONLY
set target_node_num 2
# Hence, number of SENSORS = node_num - target_node_num

set sink_id 0
set sinkX 100.0
set sinkY 0.0
set sinkZ 0.0

#Maximum simulation time
set maxsimtime 10000

#to keep track of all the sensors neighbors and their
#location used in a subroutine further-on
set sensorList []
set delNodes []

#FOR FILE WRITTING
set count 0

# create the sensor channel
mkdir drcl.inet.sensorsim.SensorChannel chan

# Capacity of the sensor channel is total number of nodes (sensors +
targets)
# make simulation for $node_num nodes
! chan setCapacity $node_num

# create the propagation model
mkdir drcl.inet.sensorsim.SeismicProp seismic_Prop
! seismic_Prop setD0 0.2
#! seismic_Prop setAtnFactor 1.0

# Grid dimensions
set gx 200
set gy 200

# create the sensor node position tracker (for target nodes)
mkdir drcl.inet.sensorsim.SensorNodePositionTracker nodetracker
                        #   maxX  minX maxY minY
! nodetracker setGrid $gx   0.0  $gy  0.0
```

```
# connect the sensor channel to the sensor node position tracker
connect chan/.tracker@ -and nodetracker/.channel@

# create the wireless channel
mkdir drcl.inet.mac.Channel channel

# Capacity of the wireless channel is number of sensors and sinks
ONLY
# which is equal to $node_num - $target_node_num
! channel setCapacity [expr $node_num - $target_node_num]

# create the node position tracker
mkdir drcl.inet.mac.NodePositionTracker tracker

#the dx and dy below represent 'how far' my signal travels
#so in this case any node located in my 100x100m grid will hear
#what a sensor broadcasts
#                 maxX minX maxY minY  dX   dY
! tracker setGrid $gx  0.0  $gy  0.0  $gx   $gy

connect channel/.tracker@ -and tracker/.channel@

#*****************
#     In order to graph total # of sensors still
#     alive we created a new component that will
#     have a connected port to a plotter
mkdir drcl.inet.sensorsim.AliveSensors    liveSensors
set numNodesPlot_ [mkdir drcl.comp.tool.Plotter .numNodesPlot]
connect -c liveSensors/.plotter@ -to $numNodesPlot_/0@0



#****************************************************
# FOR THE SINKs ONLY, do the following
# SINKs have only a network protocol stack
for {set i 0} {$i < [expr $sink_id + 1]} {incr i} {
      puts "create sink $i"
      set node($i) [mkdir drcl.comp.Component n$i]

      cd n$i
      mkdir drcl.inet.sensorsim.MultiHop.SinkAppMH app
      ! app setNid $i
      ! app setSinkNid $sink_id

      mkdir drcl.inet.sensorsim.WirelessAgent wireless_agent


      # connect the sensor application to the wireless agent
      # so that sensors can send through the wireless network
protocol stack
      connect app/down@ -to wireless_agent/up@

      # connect the wireless agent to the sensor application
      # so that sensors can receive thru the wireless network
protocol stack
      connect wireless_agent/.toSensorApp@ -to
app/.fromWirelessAgent@

      mkdir drcl.inet.mac.LL ll
      mkdir drcl.inet.mac.ARP arp
```

```
      mkdir drcl.inet.core.queue.FIFO queue
      mkdir drcl.inet.mac.Mac_802_11 mac

      # added 09-04-04
      ! mac disable_PSM

      mkdir drcl.inet.mac.WirelessPhy wphy

      inject “set omnigain_dbi = 9?wphy/.antenna@

#************
      ! wphy setMultiHopMode 1      ;#turn on MH mode settings
#************

      mkdir drcl.inet.mac.FreeSpaceModel propagation
      mkdir drcl.inet.mac.MobilityModel mobility

      set PD [mkdir drcl.inet.core.PktDispatcher      pktdispatcher]
    set RT [mkdir drcl.inet.core.RT                 rt]
      set ID [mkdir drcl.inet.core.Identity           id]

      ! pktdispatcher setRouteBackEnabled 1

      $PD bind $RT
      $PD bind $ID

#**************
#           Above we inserted a permanent entry in the
#         table going to the sink. Here this contract is
#         needed so we can send directly to neighbors.

      connect app/.setRoute@ -to rt/.service_rt@
#**************************

      # present if using 802.11 power-saving mode
      connect mac/.energy@ -and wphy/.energy@

      connect wphy/.mobility@    -and mobility/.query@
      connect wphy/.propagation@ -and propagation/.query@


      connect mac/down@ -and wphy/up@
      connect mac/up@   -and queue/output@

      connect ll/.mac@ -and mac/.linklayer@
      connect ll/down@ -and queue/up@
      connect ll/.arp@ -and arp/.arp@

      connect -c pktdispatcher/0@down -and ll/up@

      set nid $i

      ! arp setAddresses  $nid $nid
      ! ll  setAddresses  $nid $nid
      ! mac setMacAddress $nid
      ! wphy setNid       $nid
      ! mobility setNid   $nid
      ! id setDefaultID   $nid

      ! queue setMode      "packet"
      ! queue setCapacity  40
```

```
      # disable ARP
      ! arp setBypassARP  [ expr 2>1]

      ! mac setRTSThreshold 0

      connect mobility/.report@ -and /multiHop80211/tracker/.node@

      connect wphy/down@ -to /multiHop80211/channel/.node@

      ! /multiHop80211/channel attachPort $i [! wphy getPort
.channel]

      #                                          maxX maxY  maxZ minX
minY minZ  dX  dY   dZ
      ! mobility setTopologyParameters $gx  $gy   0.0  $gx   $gy  0.0
$gx $gy  0.0

      ! mac  disable_MAC_TRACE_ALL

      connect -c  wireless_agent/down@ -and pktdispatcher/1111@up

      cd ..
}


#*******************************************************************
*
# FOR THE SENSORS ONLY , do the following

for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {

      puts "create sensor $i"
      set node($i) [mkdir drcl.comp.Component n$i]

      cd n$i

      #*****************
      mkdir drcl.inet.sensorsim.MultiHop.MultiHopApp app
      ! app setNid $i
      ! app setSinkNid $sink_id
      ! app setCoherentThreshold 1000.0
      ! app setNn_ [expr $node_num - 1]
      #*************************
      #fmakis
      #Routing metric
      #metric 1 = raw distance
      #metric 2 = energy
      ! app setmetric 2

      connect app/.getNeighbor@ -and
/multiHop80211/nodetracker/.multiHop@
      #*****************

      # create nodes
      mkdir drcl.inet.sensorsim.SensorAgent agent

      ! agent setDebugEnabled 0

      # create sensor physical layers
```

```
      mkdir drcl.inet.sensorsim.SensorPhy phy
      ! phy setRxThresh 0.0
      ! phy setDebugEnabled 0

      # create mobility models
      mkdir drcl.inet.sensorsim.SensorMobilityModel mobility

      ! phy setNid $i
      ! phy setRadius [expr sqrt($gx*$gx + $gy*$gy)]

      # connect phyiscal layers to sensor agents so that nodes can
receive
      connect phy/.toAgent@ -to agent/.fromPhy@

      # connect sensor agent and sensor application
      connect agent/.toSensorApp@ -to app/.fromSensorAgent@

      # connect the sensor channel to the nodes so that they can
receive
      ! /multiHop80211/chan attachPort $i [! phy getPort .channel]

      # connect the nodes to the propagation model
      connect phy/.propagation@ -and
/multiHop80211/seismic_Prop/.query@

      ! mobility setNid $i

      # create wireless agent layers
      mkdir drcl.inet.sensorsim.WirelessAgent wireless_agent

      # connect the sensor application to the wireless agent
      # so that sensors can send through the wireless network
protocol stack
      connect app/down@ -to wireless_agent/up@

      # connect the wireless agent to the sensor application
      # so that sensors can receive thru the wireless network
protocol stack
      connect wireless_agent/.toSensorApp@ -to
app/.fromWirelessAgent@

      mkdir drcl.inet.mac.LL ll
      mkdir drcl.inet.mac.ARP arp
      mkdir drcl.inet.core.queue.FIFO queue
      mkdir drcl.inet.mac.Mac_802_11 mac

      # added 09-04-04
      ! mac disable_PSM

      #*************
      #Required to let the MAC802.11 layer
      #be in charge of maintaining the
      #radio modes and energy levels.
      ! mac setIs_uAMPS 1
      #*************

      mkdir drcl.inet.mac.WirelessPhy wphy

      #*************
      #fmakis
      #set sensing area and bandwidth with the new method
```

```
      ! wphy setsensingArea $gx $gy
      ! wphy setGr $Gr
      ! wphy setGt $Gt


      #*************
      ! wphy setMultiHopMode 1      ;#turn on MH mode settings
      #************

      mkdir drcl.inet.mac.FreeSpaceModel propagation


      set PD [mkdir drcl.inet.core.PktDispatcher      pktdispatcher]
    set RT [mkdir drcl.inet.core.RT                 rt]
    set ID [mkdir drcl.inet.core.Identity           id]

      ! pktdispatcher setRouteBackEnabled 1

      $PD bind $RT
      $PD bind $ID

      #**************
      #NICHOLAS
      # create route configuration request for testing
      #this is to define the interfaces. So in this case each sensor
      #only has 1 interface (hence array size 1) and its eth0.
      #another example is (which has 3 interfaces 0, 2, and 4:
      #set ifs [java::new drcl.data.BitSet [java::new {int[]} 3 {0 2
4}]]

      set ifs [java::new drcl.data.BitSet [java::new {int[]} 1 {0}]]
      set base_entry [java::new drcl.inet.data.RTEntry $ifs]

      set key [java::new drcl.inet.data.RTKey $i 0 -1]
      set entry_ [!!! [$base_entry clone]]

      ! rt add $key $entry_

      #**************
      #          Above we inserted a permanent entry in the
      #          table going to the sink. Here this contract is
      #          needed so we can send directly to neighbors.

      connect app/.setRoute@ -to rt/.service_rt@
      #*************************

      # present if using 802.11 power-saving mode
      connect mac/.energy@ -and wphy/.energy@

      #****************************
      connect app/.energy@ -and wphy/.appEnergy@
      mkdir drcl.inet.sensorsim.CPUAvr cpu

      connect app/.cpu@ -and cpu/.reportCPUMode@
      connect cpu/.battery@ -and wphy/.cpuEnergyPort@
      #*****************************

      connect wphy/.mobility@    -and mobility/.query@
      connect wphy/.propagation@ -and propagation/.query@

      connect mac/down@ -and wphy/up@
```

```
      connect mac/up@   -and queue/output@

      connect ll/.mac@ -and mac/.linklayer@
      connect ll/down@ -and queue/up@
      connect ll/.arp@ -and arp/.arp@

      connect -c pktdispatcher/0@down -and ll/up@

      set nid $i

      ! arp setAddresses  $nid $nid
      ! ll  setAddresses  $nid $nid
      ! mac setMacAddress $nid  ;#set MAC

      ! wphy setNid       $nid
      ! id setDefaultID   $nid

      ! queue setMode      "packet"
      ! queue setCapacity  40

      # disable ARP
      ! arp setBypassARP  [ expr 2>1]

      ! mac setRTSThreshold 0

      connect mobility/.report@ -and /multiHop80211/tracker/.node@
      connect wphy/down@ -to /multiHop80211/channel/.node@

      ! /multiHop80211/channel attachPort $i [! wphy getPort
.channel]

      #                                   maxX maxY maxZ minX minY
minZ dX  dY  dZ
      ! mobility setTopologyParameters $gx  $gy  0.0  $gx  $gy  0.0
$gx $gy 0.0

      ! mac  disable_MAC_TRACE_ALL

      connect -c  wireless_agent/down@ -and pktdispatcher/1111@up

      cd ..
}


#*************************************************************
# FOR THE TARGET NODES ONLY , do the following
if { $target_node_num == 0 } {
      puts "No target agents .... "
} else {
      for {set i [expr $node_num - $target_node_num]} {$i <
$node_num} {incr i} {
            puts "create target $i"

            set node$i [mkdir drcl.comp.Component n$i]

            cd n$i

            # create target agents
            mkdir drcl.inet.sensorsim.TargetAgent agent
            ! agent setBcastRate 1.0
            ! agent setSampleRate 1.0
```

```
            # create sensor physical layers
            mkdir drcl.inet.sensorsim.SensorPhy phy
            ! phy setRxThresh 0.0
            ! phy setNid $i
            ! phy setRadius [expr sqrt($gx*$gx + $gy*$gy)]


            ! phy setDebugEnabled 0

            # create mobility models
            mkdir drcl.inet.sensorsim.SensorMobilityModel mobility

            # connect target agents to phy layers so that nodes can
send
            connect agent/down@ -to phy/up@

            # connect phy layers to sensor channel so that nodes can
send
            connect phy/down@ -to /multiHop80211/chan/.node@

            # connect the nodes to the propagation model
            connect phy/.propagation@ -and
/multiHop80211/seismic_Prop/.query@

            ! mobility setNid $i

            # set the topology parameters
            ! mobility setTopologyParameters $gx  $gy  0.0  $gx  $gy
0.0

            cd ..
      }
}

#*********************************************************************
# for SENSORs and TARGETs only. Not SINKs
for {set i [expr $sink_id + 1]} {$i < $node_num} {incr i} {
      # connect the mobility model of each node to the node position
tracker
      connect n$i/mobility/.report_sensor@ -and
/multiHop80211/nodetracker/.node@
      connect n$i/phy/.mobility@ -and n$i/mobility/.query@
}


#**********************************************
#Positioning
#

# set the position of sink nodes args=> (speed(m/sec),
xCoord,yCoord,zCoord
#! $node(0)/mobility setPosition 0.0 0.0 0.0 0.0
! $node(0)/mobility setPosition 0.0 $sinkX $sinkY $sinkZ

#*************************************
#fmakis
#workaround for the target nodes: if we place them randomly they
might be elected as next hop neighbors
#and they cannot forward packets. Therefore we place them in
positions that cannot affect routing
```

```
! n101/mobility setPosition 0.0 0.001 [expr $gy-0.001] 0.0
! n102/mobility setPosition 0.0 [expr $gx-0.001] [expr $gy-0.001] 0.0

# for the sensors They will be randomly placed on the grid (2D only)
# set the position of sensor nodes args=> (speed(m/sec),
xCoord,yCoord,zCoord
for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
   ! n$i/mobility setPosition 0.0 [expr rand()*$gx] [expr
rand()*($gy-0.01)] 0.0
}

# for the target we can include random mobility They will be randomly
# placed on the grid (2D only)
#set the position of target nodes args=> (speed(m/sec),
xCoord,yCoord,zCoord
#for {set i [expr $node_num - $target_node_num]} {$i < $node_num}
{incr i} {
#   ! n$i/mobility setPosition 0.0 [expr rand()*$gx] [expr $gy-0.005]
0.0
#}


#****************************************************
#fmakis
#make the sensors aware of the position of the sink in case it is not
0,0,0 (not taken into account in previous implementations)
for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
   ! n$i/app setSinkLocation $sinkX $sinkY $sinkZ
}

#***************************************************
#routeInfo() to execute do:
#     script "routeInfo" -at 0.35 -period 4.0 -on $sim
proc routeInfo { } {
   global sim n1
   puts "Current Route Table\n [! n1/rt info]"
}

#***************************************************
#Output remaining energy levels of the sensors to a plotter
set plot_ [mkdir drcl.comp.tool.Plotter .plot]
for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
   connect -c n$i/app/.plotter@ -to $plot_/$i@0
}

#***************************************************
#Plotters for the SINK node

#Graph # 1
#To plot the total number of received packets at the sink
set sinkPlot1_ [mkdir drcl.comp.tool.Plotter .sinkPlot1]
connect -c n0/app/.PacketsReceivedPlot@ -to $sinkPlot1_/0@0

#Graph # 2
#Calculate the avg latency when the sink finally receives it
set sinkPlot2_ [mkdir drcl.comp.tool.Plotter .sinkPlot2]
connect -c n0/app/.latencyPlot@ -to $sinkPlot2_/0@0
```

```
#Graph # 3
#plot the actual phenomena being sensed.
! n$sink_id/app createSnrPorts $node_num $target_node_num
set sinkPlot3_ [mkdir drcl.comp.tool.Plotter .sinkPlot3]

for {set i 0} {$i < $target_node_num} {incr i} {

   connect -c n$sink_id/app/.snr$i@ -to $sinkPlot3_/$i@$i
   if { $testflag } {
       attach -c $testfile/in@ -to n$sink_id/app/.snr$i@
   }
}


#*************************************************
#wsnLoop()
#
#     This method is called periodically to check
#     if the simulation should continue or not. If
#     all nodes are dead then stop the simulator and
#     display the cummulative statistics... o.w keep
#     running
proc wsnLoop { } {
      global sim node_num node sink_id target_node_num maxsimtime
ch_num

      #reset variables
      set live_sensors 0
      set dead_sensors 0
      set total_packets 0
      set total_packetsCH 0
      set dropped_packets 0


      #check how many are still alive
      for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num ]} {incr i} {
            if { [! $node($i)/app isSensorDead] == 0 } {
                  incr live_sensors
            } else {
                  incr dead_sensors
            }

      }
      script [! liveSensors setLiveNodes $live_sensors]
      script [! liveSensors updateGraph]

      #display statistics if they are all dead.
      if { $dead_sensors >= [expr 1*($node_num - $target_node_num -
1)] || [! $sim getTime] > $maxsimtime} {
         #puts "All nodes dead at [! $sim getTime]"
         puts "$dead_sensors out of [expr $node_num -
$target_node_num - 1] nodes dead at [! $sim getTime]"
         $sim stop
         puts "-----------------------------------------------"
         puts "Simulation Terminated\n"
         puts "Results:"
         puts "Base Station Received [! n0/app getTotalINPackets]"
         puts "Collisions at Base Station: [! n0/mac getCollision]"
```

```
        for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num ]} {incr i} {
            set curr_packets [! n$i/app geteID]
            puts "Sensor$i Sent $curr_packets Packets to the Sink"
            set total_packets [expr $total_packets + $curr_packets]
         }

         set app_dropped [! n1/app getDropped_packets]
         set wphy_dropped [! n1/wphy getDropped_packets]
         #set mac_dropped [! n1/mac getDropped_packets]

         puts "Total packets dropped at Application layer:
$app_dropped"
         puts "Total packets dropped at physical layer:
$wphy_dropped"
         #puts "Drops due to collisions (discovered at MAC layer:
$mac_dropped"
         #set dropped_packets [expr $app_dropped + $wphy_dropped +
$mac_dropped]
         set dropped_packets [expr $app_dropped + $wphy_dropped]
            puts "Average Latency on BS is: [! n0/app getAvgLatency]"
         #set total_packets [expr $total_packets + $total_packetsCH]
         puts "Total Packets sent from all nodes: $total_packets"
         #puts "Total Packets sent from all Cluster Heads:
$total_packetsCH"
         puts "Number of Dropped Packets: $dropped_packets"
         puts "Success Rate: [expr ([! n0/app getTotalINPackets].0 /
$total_packets.0) * 100]"
      }
}



#***********************************************
#sensorLocPrintOut()
#     Goes throught all sensors and prints their
#     (X,Y,Z) Coordinates
proc sensorLocPrintOut { } {
      global sink_id node_num target_node_num
      for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
            script [! n$i/app printNodeLoc]
      }
}

#***********************************************
#Output sensor location and status
#to a file for GUI to read from
proc file_output { } {
   global sink_id node_num count target_node_num ch_num

   #open a file for writting
   set filename "sensorInfo$count.log"
   set out [open $filename w]

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
     #get its x and y location
     set Xloc [! n$i/app getX]
     set Yloc [! n$i/app getY]
```

```
      #get its energy
      set status [expr [! n$i/app isSensorDead] - 0]

      #write it to the file
      puts $out "$Xloc $Yloc $status"
   }
   incr count

   close $out
}

#*************************************************
#Output sensor location and status
#to a file for GUI to read from
proc energy_dist { } {
   global sink_id node_num count target_node_num ch_num

   #open a file for writting
   set filename "energyDistrNode.log"
   set out [open $filename w]

   puts $out "TxCost     RxCost    IdleCost    SleepCost    CPUcost"

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
      #collect the data
      set txCost [! n$i/wphy getRadioTotalTX]
      set rxCost [! n$i/wphy getRadioTotalRX]
      set idleCost [! n$i/wphy getRadioTotalidle]
      set sleepCost [! n$i/wphy getRadioTotalsleep]
      set cpuCost [! n$i/wphy getTotalCPU]

      #write it to the file
      puts $out "node$i - $txCost $rxCost $idleCost $sleepCost
$cpuCost"
   }

   close $out
}

#*************************************************
#Prints out each sensors neighbor at the present time
#
proc printNeighborList { } {
   global sink_id node_num count target_node_num ch_num

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
      script [! n$i/app printNeighborID]
   }
   puts "\n"

}

#*************************************************
#To call this do:
# script "getQueueSize" -at 20.0 -period 1.0 -on $sim

proc getQueueSize { } {
   global sink_id node_num count target_node_num ch_num
```

```
   puts "\n-------\nQUEUE SIZES\n-------"
   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
      puts "Queue size of sensor$i is:  [! n$i/queue getSize]"
   }

}

#***********************************************
#Find next hop neighbor-- node to whom always send data.
#Choose closest node that is in the direction of the base station.
#NOTE!  This algorithm assumes nodes know the location of all nodes
#near them.  In practice, this would require an initial set-up
#phase where this information is disseminated throughout the network
#and that each node has a GPS receiver or other location-tracking
#algorithms to determine node locations.
proc setNeighbor { } {
   global sink_id node_num count sensorList target_node_num ch_num

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num ]} {incr i} {
      script [! n$i/app setNeighbor]

      #we also need to store it in the script
      #to maintain their next neighbor when a
      #node dies.

    set selfID [! n$i/app getNid]
      set tempID [! n$i/app getNeighbor_id]
      set tempX  [! n$i/app getNeighborX]
      set tempY  [! n$i/app getNeighborY]
      set tempZ  [! n$i/app getNeighborZ]
      set tempDist [! n$i/app getNeighbor_dist]

      #inner list: [nid, neighborID, neigh_X, neigh_Y, neighZ,
tempDist]
      set iThSensor [list $selfID $tempID $tempX $tempY $tempZ
$tempDist]

      #append it to the main list.
      set sensorList [lappend SensorNeighbors $iThSensor]
   }
   #puts "Size of List is: [llength $sensorList]"
}

#***********************************************
proc neighborUpdate { } {
   global sink_id node_num count sensorList delNodes

   for {set i 0 } {$i < [llength $sensorList]} {incr i} {
     #extract the current element
      set current_node [lindex $sensorList $i]

      set newSID [lindex $current_node 0]
      set newNID [lindex $current_node 1]
      set newX  [lindex $current_node 2]
      set newY  [lindex $current_node 3]
      set newZ  [lindex $current_node 4]
      set newDist [lindex $current_node 5]

      #collect the data
```

```
      set remEnergy [! n$newSID/wphy getRemEnergy]

      #****************************
      #fmakis
      #leave just enough energy to the node so that it wont drop it's
own data
      if { $remEnergy <= 0.001 } {

      for {set j 0 } {$j < [llength $sensorList]} {incr j} {

       set currentID [lindex [lindex $sensorList $j] 0]
         set neighborID [lindex [lindex $sensorList $j] 1]

         #puts "Checking Sensor$currentID Neighbor Settings"
        if { $currentID != $newSID } {
            if { $neighborID == $newSID} {
                 #update the nodes neighbor in Java
           script [! n$currentID/app setNewNeighborID $newNID]
             script [! n$currentID/app setNewNeighborX $newX]
             script [! n$currentID/app setNewNeighborY $newY]
             script [! n$currentID/app setNewNeighborY $newZ]
             script [! n$currentID/app setNewNeighborDist $newDist]

        #update the nodes neighbor w/in script
            set iThSensor [list $currentID $newNID $newX $newY $newZ
$newDist]
            #append it to the main list.
               set sensorList [lreplace $sensorList $j $j $iThSensor]
            }
            }
      };#end inner for
        #puts " inserting number $newSID into delNodes"
        set delNodes [lappend delNodes $newSID]

      };#end if energy = 0

   } ;#end outer for
   #Now remove the elements from sensorList
   for {set k 0 } {$k < [llength $delNodes]} {incr k} {

      set delID [lindex $delNodes $k] ;#which node is it
  #    puts "getting index of $delID"
      for {set z 0 } {$z < [llength $sensorList]} {incr z} {
         set currentID [lindex [lindex $sensorList $z] 0]
       if { $currentID == $delID } {
            set index  $z
       }
      }
      set sensorList [lreplace $sensorList $index $index]
   }
   set delNodes [] ;#clear that temp list.
}


#*********************************************
#Strictly used for validation purposes. This method
#can be called to get the energy at any random
#point of a particular node.
proc getCurrEnergy { } {

   global sink_id node_num count target_node_num
```

```
   puts "\n\n"

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
      #puts "-----------------------------------------------"

      #puts "Sensor$i radio in Idle mode used: [! n$i/wphy
getRadioTotalidle] joules"
      puts "Sensor$i radio in Rx mode used: [! n$i/wphy
getRadioTotalRX] joules"
      #puts "Sensor$i radio in Tx mode used: [! n$i/wphy
getRadioTotalTX] joules\n"

      #puts "Sensor$i Radio was idle for: [! n$i/wphy
getTotalRadioIdleTime] sec"
      #puts "Sensor$i Radio was Tx for: [! n$i/wphy
getTotalRadioActiveTime] sec"
      puts "Sensor$i Radio was Rx for: [! n$i/wphy
getTotalRadioRxTime] sec\n"

      #puts "Sensor$i CPU active used: [! n$i/wphy getTotalCPUactive]
joules"
      #puts "Sensor$i CPU sleep used: [! n$i/wphy getTotalCPUsleep]
joules\n"

      #puts "Sensor$i CPU was active for: [! n$i/wphy
getTotalCPUactiveTime] sec"
      #puts "Sensor$i CPU was sleep for: [! n$i/wphy
getTotalCPUsleepTime] sec\n"


      #puts "Sensor$i CPU Total used: [! n$i/wphy getTotalCPU] joules
\n"

      puts "Sensor$i has [! n$i/wphy getRemEnergy] joules remaining"

   }
}

#-------------------------


#*********************************************
puts "simulation begins...\n"
set sim [attach_simulator .]
#$sim setDebugEnabled 1
$sim stop

#******************start the sink************************
script {run n0} -at 0.001 -on $sim

#***************start the sensors************************
for {set i [expr $sink_id + 1]} {$i < $node_num} {incr i} {
      script puts "run n$i" -at 0.01 -on $sim
}

#*********print out all the node locations**************
script "sensorLocPrintOut" -at 0.03 -on $sim

#********determine who their neighbors are**************
```

```
script "setNeighbor" -at 0.2 -on $sim

#*******print off each of their respective neighbors****
script "printNeighborList" -at 0.3 -on $sim

#*******Check if Sensor Status*************************
script "wsnLoop" -at 1.0 -period 3.0 -on $sim

#*******To maintain neighbor settings******************
#script "neighborUpdate" -at 15.0 -period 0.5 -on $sim
script "neighborUpdate" -at 15.0 -period 0.5 -on $sim

#************For Matlab plotting************************
#script "file_output" -at 1.0 -period 100.0 -on $sim

#the following line is only used in validation!
#script "getCurrEnergy" -at 150.0 -on $sim

#$sim resumeTo 10000.1
#Maximum simulation time as defined in wsnLoop()

$sim resumeTo [expr $maxsimtime+5]
```

LEACH

```
# ************************************************************
# Sample tcl file of a wireless sensor network simulation.

# Author: Nicholas Merizzi
# Date: 12/23/2003
#
# This script will simulate a sensor network which utilizes
# the hierarchichal routing protocol called LEACH. J-Sim was
# modified so that clustering can occur, as well as complete
# power control over the radio components.
#
#
#************************************************************

source "../../../../test/include.tcl"

cd [mkdir -q drcl.comp.Component /LEACH]

# TOTAL number of nodes (sensor nodes + target nodes)
set node_num 103

# Number of TARGET nodes ONLY
set target_node_num 2
# Hence, number of SENSORS = node_num - target_node_num

set sink_id 0
set sinkX 50.0
set sinkY 0.0
set sinkZ 0.0
#*************************
#NICHOLAS
#FOR FILE WRITTING
set count 0

# create the sensor channel
mkdir drcl.inet.sensorsim.SensorChannel chan

# Capacity of the sensor channel is total number of nodes (sensors +
targets)
# make simulation for $node_num nodes
! chan setCapacity $node_num

# create the propagation model
mkdir drcl.inet.sensorsim.SeismicProp seismic_Prop
! seismic_Prop setD0 0.2

# Grid dimensions
set gx 100
set gy 100


# create the sensor node position tracker
mkdir drcl.inet.sensorsim.SensorNodePositionTracker nodetracker
                              # maxX  minX maxY minY
! nodetracker setGrid $gx    0.0  $gy   0.0

# connect the sensor channel to the sensor node position tracker
```

```
connect chan/.tracker@ -and nodetracker/.channel@

# create the wireless channel
mkdir drcl.inet.mac.Channel channel

# Capacity of the wireless channel is number of sensors and sinks
ONLY
# which is equal to $node_num - $target_node_num
! channel setCapacity [expr $node_num - $target_node_num]

# create the node position tracker
mkdir drcl.inet.mac.NodePositionTracker tracker
#the dx and dy below represent 'how far' my signal travels
#so in this case any node located in my 100x100m grid will hear
#what a sensor broadcasts
#                 maxX minX maxY minY  dX   dY
! tracker setGrid $gx 0.0 $gy 0.0 $gx $gy

connect channel/.tracker@ -and tracker/.channel@

#*****************
#NICHOLAS
#     In order to graph total # of sensors still
#     alive we created a new component that will
#     have a connected port to a plotter
mkdir drcl.inet.sensorsim.AliveSensors    liveSensors
set numNodesPlot_ [mkdir drcl.comp.tool.Plotter .numNodesPlot]
connect -c liveSensors/.plotter@ -to $numNodesPlot_/0@0



#****************************************************
# FOR THE SINKs ONLY, do the following
# SINKs have only a network protocol stack
for {set i 0} {$i < [expr $sink_id + 1]} {incr i} {
      puts "create sink $i"
      set node($i) [mkdir drcl.comp.Component n$i]

      cd n$i
      mkdir drcl.inet.sensorsim.LEACH.SinkAppLEACH app
      ! app setNid $i
      ! app setSinkNid $sink_id
      ! app setCoherentThreshold 1000.0

      # connect the sensor application to the wireless agent
      # so that sinks can send through the wireless network protocol
stack
      # create wireless agent layers
      mkdir drcl.inet.sensorsim.LEACH.WirelessLEACHAgent
wireless_agent

      # connect the sensor application to the wireless agent
      # so that sensors can send through the wireless network
protocol stack
      connect app/down@ -to wireless_agent/up@

      # connect the wireless agent to the sensor application
      # so that sensors can receive thru the wireless network
protocol stack
      connect wireless_agent/.toSensorApp@ -to
app/.fromWirelessAgent@
```

```
mkdir drcl.inet.mac.LL ll
mkdir drcl.inet.mac.ARP arp
mkdir drcl.inet.core.queue.FIFO queue
mkdir drcl.inet.mac.CSMA.Mac_CSMA mac
mkdir drcl.inet.mac.WirelessPhy wphy

#*************
#NICHOLAS
! wphy setLEACHMode 1
connect wphy/.channelCheck@ -and mac/.wphyRadioMode@
#************

mkdir drcl.inet.mac.FreeSpaceModel propagation
mkdir drcl.inet.mac.MobilityModel mobility

set PD [mkdir drcl.inet.core.PktDispatcher      pktdispatcher]
set RT [mkdir drcl.inet.core.RT                 rt]
set ID [mkdir drcl.inet.core.Identity           id]

! pktdispatcher setRouteBackEnabled 1

$PD bind $RT
$PD bind $ID

#**************
#Nicholas --> Here this contract is
#     needed so we can send directly to neighbors.

connect app/.setRoute@ -to rt/.service_rt@
#*************************

connect wphy/.mobility@    -and mobility/.query@
connect wphy/.propagation@ -and propagation/.query@

connect mac/down@ -and wphy/up@
connect mac/up@   -and queue/output@

connect ll/.mac@ -and mac/.linklayer@
connect ll/down@ -and queue/up@
connect ll/.arp@ -and arp/.arp@

connect -c pktdispatcher/0@down -and ll/up@

set nid $i

! arp setAddresses  $nid $nid
! ll  setAddresses  $nid $nid
! mac setNode_num_  $nid       ;#set the MAc address

#let it know we are running LEACH for SS
! mac setLEACHmode 1
! mac setMacAddress $nid  ;#set MAC
! wphy setNid       $nid
! mobility setNid   $nid
! id setDefaultID   $nid

! queue setMode      "packet"
! queue setCapacity  40

# disable ARP
```

```
      ! arp setBypassARP  [ expr 2>1]

      connect mobility/.report@ -and /LEACH/tracker/.node@
      connect wphy/down@ -to /LEACH/channel/.node@

      ! /LEACH/channel attachPort $i [! wphy getPort .channel]

      #                                          maxX maxY  maxZ minX
minY minZ  dX  dY   dZ
      ! mobility setTopologyParameters $gx $gy 0.0 $gx $gy 0.0 $gx
$gy 0.0

      connect -c  wireless_agent/down@ -and pktdispatcher/1111@up

      cd ..
}

#*******************************************************************
*
# FOR THE SENSORS ONLY , do the following

for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {

      puts "create sensor $i"
      set node($i) [mkdir drcl.comp.Component n$i]

      cd n$i

      mkdir drcl.inet.sensorsim.LEACH.LEACHApp app
      ! app setNid $i
      ! app setSinkNid $sink_id
      ! app setCoherentThreshold 1000.0

      #******
      #Nicholas
      ! app setNn_ [expr $node_num - 1]
      ! app setNum_clusters 5
      ! app setTotal_rounds 5


      # create nodes
      mkdir drcl.inet.sensorsim.SensorAgent agent

      ! agent setDebugEnabled 0

      # create sensor physical layers
      mkdir drcl.inet.sensorsim.SensorPhy phy
      ! phy setRxThresh 0.0
      ! phy setDebugEnabled 0

      # create mobility models
      mkdir drcl.inet.sensorsim.SensorMobilityModel mobility

      ! phy setNid $i
      ! phy setRadius 200.0

      # connect phyiscal layers to sensor agents so that nodes can
receive
      connect phy/.toAgent@ -to agent/.fromPhy@
```

```
      # connect sensor agent and sensor application
      connect agent/.toSensorApp@ -to app/.fromSensorAgent@

      # connect the sensor channel to the nodes so that they can
receive
      ! /LEACH/chan attachPort $i [! phy getPort .channel]

      # connect the nodes to the propagation model
      connect phy/.propagation@ -and /LEACH/seismic_Prop/.query@

      ! mobility setNid $i

      # create wireless agent layers
      mkdir drcl.inet.sensorsim.LEACH.WirelessLEACHAgent
wireless_agent

      # connect the sensor application to the wireless agent
      # so that sensors can send through the wireless network
protocol stack
      connect app/down@ -to wireless_agent/up@

      # connect the wireless agent to the sensor application
      # so that sensors can receive thru the wireless network
protocol stack
      connect wireless_agent/.toSensorApp@ -to
app/.fromWirelessAgent@

      mkdir drcl.inet.mac.LL ll
      mkdir drcl.inet.mac.ARP arp
      mkdir drcl.inet.core.queue.FIFO queue
      mkdir drcl.inet.mac.CSMA.Mac_CSMA mac
      mkdir drcl.inet.mac.WirelessPhy wphy

      #***************
      #NICHOLAS
      ! mac setLEACHmode 1    ;#let it know we are running LEACH for
SS
      ! wphy setLEACHMode 1   ;#let it know we are running LEACH for
SS
      ! wphy setMIT_uAMPS 1   ;#turn on MH mode settings
      connect wphy/.channelCheck@ -and mac/.wphyRadioMode@

      mkdir drcl.inet.mac.FreeSpaceModel propagation

      set PD [mkdir drcl.inet.core.PktDispatcher      pktdispatcher]
      set RT [mkdir drcl.inet.core.RT                 rt]
      set ID [mkdir drcl.inet.core.Identity           id]

      ! pktdispatcher setRouteBackEnabled 1

      $PD bind $RT
      $PD bind $ID

      #**************
      #NICHOLAS
      # create route configuration request for testing
      #this is to define the interfaces. So in this case each sensor
      #only has 1 interface (hence array size 1) and its eth0.
      #another example is (which has 3 interfaces 0, 2, and 4:
      #set ifs [java::new drcl.data.BitSet [java::new {int[]} 3 {0 2
4}]]
```

```
set ifs [java::new drcl.data.BitSet [java::new {int[]} 1 {0}]]
set base_entry [java::new drcl.inet.data.RTEntry $ifs]

set key [java::new drcl.inet.data.RTKey $i 0 -1]
set entry_ [!!! [$base_entry clone]]

! rt add $key $entry_

#**************
#Nicholas -->Above we inserted a permanent entry in the
#          table going to the sink. Here this contract is
#          needed so we can send directly to neighbors.

connect app/.setRoute@ -to rt/.service_rt@

#*****************************
#NICHOLAS
connect app/.energy@ -and wphy/.appEnergy@
mkdir drcl.inet.sensorsim.CPUAvr cpu

connect app/.cpu@ -and cpu/.reportCPUMode@
connect cpu/.battery@ -and wphy/.cpuEnergyPort@
#******************************

#******************************
#NICHOLAS
connect mac/.sensorApp@ -and app/.macSensor@
#******************************

connect wphy/.mobility@    -and mobility/.query@
connect wphy/.propagation@ -and propagation/.query@

connect mac/down@ -and wphy/up@
connect mac/up@   -and queue/output@

connect ll/.mac@ -and mac/.linklayer@
connect ll/down@ -and queue/up@
connect ll/.arp@ -and arp/.arp@

connect -c pktdispatcher/0@down -and ll/up@

set nid $i

! arp setAddresses  $nid $nid
! ll  setAddresses  $nid $nid
! mac setNode_num_  $nid            ;#set the MAc address
! mac setMacAddress $nid      ;#same as above
! wphy setNid       $nid
! id setDefaultID   $nid

! queue setMode      "packet"
! queue setCapacity  40

# disable ARP
! arp setBypassARP  [ expr 2>1]


connect mobility/.report@ -and /LEACH/tracker/.node@

connect wphy/down@ -to /LEACH/channel/.node@
```

```
      ! /LEACH/channel attachPort $i [! wphy getPort .channel]

      #                                 maxX maxY maxZ minX minY minZ
dX dY dZ
      ! mobility setTopologyParameters $gx $gy 0.0 $gx $gy 0.0 $gx
$gy 0.0


      connect -c  wireless_agent/down@ -and pktdispatcher/1111@up

      cd ..
}

#*************************************************************
# FOR THE TARGET NODES ONLY , do the following
if { $target_node_num == 0 } {
      puts "No target agents .... "
} else {
      for {set i [expr $node_num - $target_node_num]} {$i <
$node_num} {incr i} {
            puts "create target $i"

            set node$i [mkdir drcl.comp.Component n$i]

            cd n$i

            # create target agents
            mkdir drcl.inet.sensorsim.TargetAgent agent
            ! agent setBcastRate 1.0
            ! agent setSampleRate 1.0

            # create sensor physical layers
            mkdir drcl.inet.sensorsim.SensorPhy phy
            ! phy setRxThresh 0.0
            ! phy setNid $i
            ! phy setRadius 200.0

            ! phy setDebugEnabled 0

            # create mobility models
            mkdir drcl.inet.sensorsim.SensorMobilityModel mobility

            # connect target agents to phy layers so that nodes can
send
            connect agent/down@ -to phy/up@

            # connect phy layers to sensor channel so that nodes can
send
            connect phy/down@ -to /LEACH/chan/.node@

            # connect the nodes to the propagation model
            connect phy/.propagation@ -and
/LEACH/seismic_Prop/.query@

            ! mobility setNid $i

            # set the topology parameters
            ! mobility setTopologyParameters $gx $gy 0.0 $gx $gy 0.0

            cd ..
```

```
      }
}

#*******************************************************************
# for SENSORs and TARGETs only. Not SINKs
for {set i [expr $sink_id + 1]} {$i < $node_num} {incr i} {
      # connect the mobility model of each node to the node position
tracker
      connect n$i/mobility/.report_sensor@ -and
/LEACH/nodetracker/.node@
      connect n$i/phy/.mobility@ -and n$i/mobility/.query@
}

#**********************************************
#Positioning
#

# set the position of sink nodes args=> (speed(m/sec),
xCoord,yCoord,zCoord
! $node(0)/mobility setPosition 0.0 $sinkX $sinkY $sinkZ

! n101/mobility setPosition 0.0 0.001 [expr $gy-0.001] 0.0
! n102/mobility setPosition 0.0 [expr $gx-0.001] [expr $gy-0.001] 0.0

# for the sensors They will be randomly placed on the grid (2D only)
# set the position of sensor nodes args=> (speed(m/sec),
xCoord,yCoord,zCoord

for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
   ! n$i/mobility setPosition 0.0 [expr rand()*$gx] [expr rand() *
$gy] 0.0
}

# for the target we can include random mobility They will be randomly
# placed on the grid (2D only)
#set the position of target nodes args=> (speed(m/sec),
xCoord,yCoord,zCoord
#for {set i [expr $node_num - $target_node_num]} {$i < $node_num}
{incr i} {
#   ! n$i/mobility setPosition 0.0 [expr rand()*$gx] [expr rand() *
$gy] 0.0
#}

# make the sensors aware of the position of the sink
for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
   ! n$i/app setSinkLocation $sinkX $sinkY $sinkZ
}

#**********************************************
#routeInfo() to execute do:
#     script "routeInfo" -at 0.35 -period 4.0 -on $sim
proc routeInfo { } {
   global sim n1
   puts "Current Route Table\n [! n1/rt info]"
}

#**********************************************
#Output remaining energy levels of the sensors to a plotter
set plot_ [mkdir drcl.comp.tool.Plotter .plot]
```

```
for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
   connect -c n$i/app/.plotter@ -to $plot_/$i@0
}

#*************************************************
#Plotters for the SINK node

#Graph # 1
#To plot the total number of received packets at the sink
#red line-> actual packets received
#blue line-> what the sink actually would have received if it
#          wasn't for CH aggregation
set sinkPlot1_ [mkdir drcl.comp.tool.Plotter .sinkPlot1]
connect -c n0/app/.PacketsReceivedPlot@ -to $sinkPlot1_/0@0
connect -c n0/app/.theo_PacketsReceivedPlot@ -to $sinkPlot1_/1@0

#Graph # 2
#Calculate the avg latency when the sink finally receives it
set sinkPlot2_ [mkdir drcl.comp.tool.Plotter .sinkPlot2]
connect -c n0/app/.latencyPlot@ -to $sinkPlot2_/0@0


#Graph # 3
#plot the actual phenomena being sensed.
! n$sink_id/app createSnrPorts $node_num $target_node_num
set sinkPlot3_ [mkdir drcl.comp.tool.Plotter .sinkPlot3]

for {set i 0} {$i < $target_node_num} {incr i} {

   connect -c n$sink_id/app/.snr$i@ -to $sinkPlot3_/$i@$i
   if { $testflag } {
       attach -c $testfile/in@ -to n$sink_id/app/.snr$i@
   }
}

#Graph # 4 --Instead this graph is combined into Graph#1 to
#see exactly how fewer the number of incomming packets are
#at the base station.
#set sinkPlot4_ [mkdir drcl.comp.tool.Plotter .sinkPlot4]
#connect -c n0/app/.theo_PacketsReceivedPlot@ -to $sinkPlot4_/0@0

#*************************************************
#wsnLoop()
#
#     This method is called periodically to check
#     if the simulation should continue or not. If
#     all nodes are dead or simtime>maxsimtime then stop the
simulator and
#     display the cummulative statistics otherwise keep running

proc wsnLoop { } {
      global sim node_num node sink_id target_node_num maxsimtime

      #reset variables
      set live_sensors 0
      set dead_sensors 0
      set total_packets 0
      set dropped_packets 0

      #check how many are still alive
```

```
      for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
            if { [! $node($i)/app isSensorDead] == 0 } {
                  incr live_sensors
            } else {
                  incr dead_sensors
            }

      }
      script [! liveSensors setLiveNodes $live_sensors]
      script [! liveSensors updateGraph]

      #display statistics if they are all dead.
      if { $dead_sensors == [expr $node_num - $target_node_num - 1]
|| [! $sim getTime] > $maxsimtime} {
         #puts "All nodes dead at [! $sim getTime]"
         puts "$dead_sensors nodes dead at [! $sim getTime]"
         $sim stop
         puts "-----------------------------------------------"
         puts "Simulation Terminated\n"
         puts "Results:"
         puts "Base Station Received [! n0/app getTotalINPackets]"
         puts "Collisions at Base Station: [! n0/mac getCollision]"

           for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
            set curr_packets [! n$i/app geteID]
            puts "Sensor$i Sent $curr_packets Packets to BS"
            set total_packets [expr $total_packets + $curr_packets]
         }

         set app_dropped [! n1/app getDropped_packets]
         set wphy_dropped [! n1/wphy getDropped_packets]
         set mac_dropped [! n1/mac getDropped_packets]

         puts "Total packets dropped at Application layer:
$app_dropped"
         puts "Total packets dropped at physical layer:
$wphy_dropped"
         puts "Drops due to collisions (discovered at MAC layer:
$mac_dropped"
         set dropped_packets [expr $app_dropped + $wphy_dropped +
$mac_dropped]

         puts "Total Packets sent from all nodes: $total_packets"
         puts "Number of Dropped Packets: $dropped_packets"
         puts "Success Rate: [expr ([! n0/app getTotalINPackets].0 /
$total_packets.0) * 100]"
      }
}


#*********************************************
#sensorLocPrintOut()
#     Goes throught all sensors and prints their
#     (X,Y,Z) Coordinates
proc sensorLocPrintOut { } {
      global sink_id node_num target_node_num
      for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
            script [! n$i/app printNodeLoc]
```

```
      }
}

#**********************************************
#Output sensor location and status
#to a file for GUI to read from
proc file_output { } {
   global sink_id node_num count target_node_num

   #open a file for writting
   set filename "sensorInfo$count.log"
   set out [open $filename w]

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
     #get its x and y location
     set Xloc [! n$i/app getX]
     set Yloc [! n$i/app getY]

     #get its energy
     set status [expr [! n$i/app isSensorDead] - 0]

     #write it to the file
     puts $out "$Xloc $Yloc $status"
   }
   incr count

   close $out
}


#**********************************************
#Output sensor location and status
#to a file for GUI to read from
proc energy_dist { } {
   global sink_id node_num count target_node_num

   #open a file for writting
   set filename "energyDistrNode.log"
   set out [open $filename w]

   puts $out "TxCost     RxCost    IdleCost    SleepCost    CPUcost"

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
      #collect the data
      set txCost [! n$i/wphy getRadioTotalTX]
      set rxCost [! n$i/wphy getRadioTotalRX]
      set idleCost [! n$i/wphy getRadioTotalidle]
      set sleepCost [! n$i/wphy getRadioTotalsleep]
      set cpuCost [! n$i/wphy getTotalCPU]

      #write it to the file
      puts $out "$txCost $rxCost $idleCost $sleepCost $cpuCost"
   }

   close $out
}


#**********************************************
```

```
#Strictly used for validation purposes. This method
#can be called to get the energy at any random
#point of a particular node.
proc getCurrEnergy { } {

   puts "\n\n"
   global sink_id node_num count target_node_num

   puts "last update time was at: [! n1/wphy getLastUpdateTime]
seconds"

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
      puts "-----------------------------------------------"
      puts "Sensor$i has [! n$i/wphy getRemEnergy] joules
remaining\n"

      puts "Sensor$i idle radio used: [! n$i/wphy getRadioTotalidle]
joules"
      puts "Sensor$i Radio was in idle mode for: [! n$i/wphy
getTotalRadioIdleTime] sec\n"

      puts "Sensor$i CPU active used: [! n$i/wphy getTotalCPUactive]
joules"
      puts "Sensor$i CPU was active for: [! n$i/wphy
getTotalCPUactiveTime] sec\n"

      puts "Sensor$i CPU sleep used: [! n$i/wphy getTotalCPUsleep]
joules"
      puts "Sensor$i CPU was sleep for: [! n$i/wphy
getTotalCPUsleepTime] sec\n"

      puts "Sensor$i CPU Total used: [! n$i/wphy getTotalCPU] joules
\n"

   }
}


#***********************************************
puts "Simulation begins...\n"
set sim [attach_simulator .]
$sim stop


#******************start the sink************************
script {run n0} -at 0.001 -on $sim

#***************start the sensors************************
for {set i [expr $sink_id + 1]} {$i < $node_num} {incr i} {
      script puts "run n$i" -at 0.1 -on $sim
}

#*********print out all the node locations**************
script "sensorLocPrintOut" -at 0.002 -on $sim

#*******Check if Sensor Status*************************
script "wsnLoop" -at 1.0 -period 2.0 -on $sim


#************For Matlab plotting************************
```

```
#script "file_output" -at 200.0 -period 100.0 -on $sim

#************One-time Capture of where energy went******
#script "energy_dist" -at 500.0 -period 200.0 -on $sim

#the following line is only used in validation!
#script "getCurrEnergy" -at 4.99 -on $sim

$sim resumeTo 100000.0
```

Static Clustering with optimal energy routing

```
# ##########################################################
# Sample tcl file of a wireless sensor network simulation
# with static clustering
# ##########################################################

source "../../../../test/include.tcl"



cd [mkdir -q drcl.comp.Component /fmRP]

# TOTAL number of nodes (sensor nodes + target nodes + cluster heads)
set node_num 103

# Number of TARGET nodes ONLY
set target_node_num 2
# Hence, number of SENSORS = node_num - target_node_num

# Number of Cluster Heads
set ch_num 6

set sink_id 0
set sinkX 50.0
set sinkY 0.0
set sinkZ 0.0

# Grid dimensions
set gx 100
set gy 100

#Maximum simulation time
set maxsimtime 10000

#for the sensors
set Gt 1.0
set Gr 1.0

#to keep track of all the sensors neighbors and their
#location used in a subroutine further-on
set sensorList []
set delNodes []

for {set i [expr $sink_id +1]} {$i < [expr $ch_num + 1]} {incr i} {
      set chlist$i []
      set tchlist$i []
}



#*************************
#FOR FILE WRITTING
set count 0

# create the sensor channel
mkdir drcl.inet.sensorsim.SensorChannel chan
```

```
# Capacity of the sensor channel is total number of nodes (sensors +
targets)
# make simulation for $node_num nodes
! chan setCapacity $node_num

# create the propagation model
mkdir drcl.inet.sensorsim.SeismicProp seismic_Prop
! seismic_Prop setD0 0.2
#! seismic_Prop setAtnFactor 1.0


# create the sensor node position tracker (for target nodes)
mkdir drcl.inet.sensorsim.SensorNodePositionTracker nodetracker
                        #   maxX  minX maxY minY
! nodetracker setGrid $gx   0.0  $gy  0.0

# connect the sensor channel to the sensor node position tracker
connect chan/.tracker@ -and nodetracker/.channel@

# create the wireless channel
mkdir drcl.inet.mac.Channel channel

# Capacity of the wireless channel is number of sensors and sinks
ONLY
# which is equal to $node_num - $target_node_num
! channel setCapacity [expr $node_num - $target_node_num]

# create the node position tracker
mkdir drcl.inet.mac.NodePositionTracker tracker

#the dx and dy below represent 'how far' my signal travels
#so in this case any node located in my 100x100m grid will hear
#what a sensor broadcasts
#                 maxX minX maxY minY  dX   dY
! tracker setGrid $gx  0.0  $gy  0.0  $gx   $gy

connect channel/.tracker@ -and tracker/.channel@

#*****************
#     In order to graph total # of sensors still
#     alive we created a new component that will
#     have a connected port to a plotter
mkdir drcl.inet.sensorsim.AliveSensors    liveSensors
set numNodesPlot_ [mkdir drcl.comp.tool.Plotter .numNodesPlot]
connect -c liveSensors/.plotter@ -to $numNodesPlot_/0@0



#*****************************************************
# FOR THE SINKs ONLY, do the following
# SINKs have only a network protocol stack
for {set i 0} {$i < [expr $sink_id + 1]} {incr i} {
      puts "create sink $i"
      set node($i) [mkdir drcl.comp.Component n$i]

      cd n$i
      mkdir drcl.inet.sensorsim.MultiHop.SinkAppMH app
      ! app setNid $i
      ! app setSinkNid $sink_id

      mkdir drcl.inet.sensorsim.WirelessAgent wireless_agent
```

```
      # connect the sensor application to the wireless agent
      # so that sensors can send through the wireless network
protocol stack
      connect app/down@ -to wireless_agent/up@

      # connect the wireless agent to the sensor application
      # so that sensors can receive thru the wireless network
protocol stack
      connect wireless_agent/.toSensorApp@ -to
app/.fromWirelessAgent@

      mkdir drcl.inet.mac.LL ll
      mkdir drcl.inet.mac.ARP arp
      mkdir drcl.inet.core.queue.FIFO queue
      mkdir drcl.inet.mac.Mac_802_11 mac

      # added 09-04-04
      ! mac disable_PSM

      mkdir drcl.inet.mac.WirelessPhy wphy
      #! wphy setBandwidth 250e3
      inject “set omnigain_dbi = 9?wphy/.antenna@

#************
      ! wphy setMultiHopMode 1      ;#turn on MH mode settings
#************

      ! wphy setRxThresh 1e-10
      ! wphy setCSThresh 1e-10

      mkdir drcl.inet.mac.FreeSpaceModel propagation
      mkdir drcl.inet.mac.MobilityModel mobility

      set PD [mkdir drcl.inet.core.PktDispatcher      pktdispatcher]
    set RT [mkdir drcl.inet.core.RT                 rt]
      set ID [mkdir drcl.inet.core.Identity           id]

      ! pktdispatcher setRouteBackEnabled 1

      $PD bind $RT
      $PD bind $ID

#**************
#            Above we inserted a permanent entry in the
#          table going to the sink. Here this contract is
#          needed so we can send directly to neighbors.

      connect app/.setRoute@ -to rt/.service_rt@
#**************************

      # present if using 802.11 power-saving mode
      connect mac/.energy@ -and wphy/.energy@

      connect wphy/.mobility@    -and mobility/.query@
      connect wphy/.propagation@ -and propagation/.query@


      connect mac/down@ -and wphy/up@
      connect mac/up@   -and queue/output@
```

```
      connect ll/.mac@ -and mac/.linklayer@
      connect ll/down@ -and queue/up@
      connect ll/.arp@ -and arp/.arp@

      connect -c pktdispatcher/0@down -and ll/up@

      set nid $i

      ! arp setAddresses  $nid $nid
      ! ll  setAddresses  $nid $nid
      ! mac setMacAddress $nid
      ! wphy setNid       $nid
      ! mobility setNid   $nid
      ! id setDefaultID   $nid

      ! queue setMode      "packet"
      ! queue setCapacity  40

      # disable ARP
      ! arp setBypassARP  [ expr 2>1]

      ! mac setRTSThreshold 0

      connect mobility/.report@ -and /fmRP/tracker/.node@

      connect wphy/down@ -to /fmRP/channel/.node@

      ! /fmRP/channel attachPort $i [! wphy getPort .channel]

      #                                          maxX maxY  maxZ minX
minY minZ  dX  dY   dZ
      ! mobility setTopologyParameters $gx  $gy   0.0  $gx   $gy  0.0
$gx $gy  0.0

      ! mac  disable_MAC_TRACE_ALL

      connect -c  wireless_agent/down@ -and pktdispatcher/1111@up

      cd ..
}

#######################################################################
##############################

#*****************************************************
# FOR THE Cluster Heads ONLY, do the following
# CHs have only a network protocol stack
for {set i [expr $sink_id +1]} {$i < [expr $ch_num + 1]} {incr i} {
      puts "create CH $i"
      set node($i) [mkdir drcl.comp.Component n$i]

      cd n$i

      #*****************
      #fmakis
      mkdir drcl.inet.sensorsim.MultiHop.MultiHopApp app
      ! app setNid $i
      ! app setSinkNid $sink_id
      ! app setCoherentThreshold 1000.0
      ! app setNn_ [expr $node_num - 1]
```

```
      ! app setCH
      ! app setfmRP
      ! app setNewNeighborID $sink_id
      ! app setIs_uAMPS 0
      ! app setmetric 2


      connect app/.getNeighbor@ -and /fmRP/nodetracker/.multiHop@
      #*****************

      # create nodes
      mkdir drcl.inet.sensorsim.SensorAgent agent

      ! agent setDebugEnabled 0

      # create sensor physical layers
      mkdir drcl.inet.sensorsim.SensorPhy phy
      ! phy setRxThresh 0.0

      ! phy setDebugEnabled 0

      # create mobility models
      mkdir drcl.inet.sensorsim.SensorMobilityModel mobility

      ! phy setNid $i
      ! phy setRadius [expr sqrt($gx*$gx+$gy*$gy)]

      #connect phyiscal layers to sensor agents so that nodes can
receive
      connect phy/.toAgent@ -to agent/.fromPhy@

      #connect sensor agent and sensor application
      connect agent/.toSensorApp@ -to app/.fromSensorAgent@

      #connect the sensor channel to the nodes so that they can
receive
      ! /fmRP/chan attachPort $i [! phy getPort .channel]

      #connect the nodes to the propagation model
      connect phy/.propagation@ -and /fmRP/seismic_Prop/.query@

      ! mobility setNid $i

      # create wireless agent layers
      mkdir drcl.inet.sensorsim.WirelessAgent wireless_agent

      # connect the sensor application to the wireless agent
      # so that sensors can send through the wireless network
protocol stack
      connect app/down@ -to wireless_agent/up@

      # connect the wireless agent to the sensor application
      # so that sensors can receive thru the wireless network
protocol stack
      connect wireless_agent/.toSensorApp@ -to
app/.fromWirelessAgent@

      mkdir drcl.inet.mac.LL ll
      mkdir drcl.inet.mac.ARP arp
      mkdir drcl.inet.core.queue.FIFO queue
      mkdir drcl.inet.mac.Mac_802_11 mac
```

```
      # added 09-04-04
      ! mac disable_PSM

      #*************
      #Required to let the MAC802.11 layer
      #be in charge of maintaining the
      #radio modes and energy levels.
      ! mac setIs_uAMPS 1
      #*************

      mkdir drcl.inet.mac.WirelessPhy wphy
      inject “set omnigain_dbi = 3?wphy/.antenna@

      #*************
      #fmakis
      #set sensing area with the new method
      ! wphy setsensingArea $gx $gy
      #inject "create SwitchedBeam Antenna" wphy/.antenna@
      #inject "azimuthpatterns = azimuth-simple" wphy/.antenna@
      #*************
      ! wphy setMultiHopMode 1      ;#turn on MH mode settings
      #************

      ! wphy setPt 0.5
      ! wphy setRxThresh 1e-9
      ! wphy setCSThresh 1e-9

      mkdir drcl.inet.mac.FreeSpaceModel propagation

      set PD [mkdir drcl.inet.core.PktDispatcher      pktdispatcher]
    set RT [mkdir drcl.inet.core.RT                 rt]
    set ID [mkdir drcl.inet.core.Identity           id]

      ! pktdispatcher setRouteBackEnabled 1

      $PD bind $RT
      $PD bind $ID

      #**************
      #NICHOLAS
      # create route configuration request for testing
      #this is to define the interfaces. So in this case each sensor
      #only has 1 interface (hence array size 1) and its eth0.
      #another example is (which has 3 interfaces 0, 2, and 4:
      #set ifs [java::new drcl.data.BitSet [java::new {int[]} 3 {0 2
4}]]

      set ifs [java::new drcl.data.BitSet [java::new {int[]} 1 {0}]]
      set base_entry [java::new drcl.inet.data.RTEntry $ifs]

      set key [java::new drcl.inet.data.RTKey $i 0 -1]
      set entry_ [!!! [$base_entry clone]]

      ! rt add $key $entry_

      #**************
      #            Above we inserted a permanent entry in the
      #          table going to the sink. Here this contract is
      #          needed so we can send directly to neighbors.
```

```
      connect app/.setRoute@ -to rt/.service_rt@
      #*************************

      # present if using 802.11 power-saving mode
      connect mac/.energy@ -and wphy/.energy@

      #****************************
      connect app/.energy@ -and wphy/.appEnergy@
      mkdir drcl.inet.sensorsim.CPUAvr cpu

      connect app/.cpu@ -and cpu/.reportCPUMode@
      connect cpu/.battery@ -and wphy/.cpuEnergyPort@
      #*****************************

      connect wphy/.mobility@    -and mobility/.query@
      connect wphy/.propagation@ -and propagation/.query@

      connect mac/down@ -and wphy/up@
      connect mac/up@   -and queue/output@

      connect ll/.mac@ -and mac/.linklayer@
      connect ll/down@ -and queue/up@
      connect ll/.arp@ -and arp/.arp@

      connect -c pktdispatcher/0@down -and ll/up@

      set nid $i

      ! arp setAddresses  $nid $nid
      ! ll  setAddresses  $nid $nid
      ! mac setMacAddress $nid  ;#set MAC

      ! wphy setNid       $nid
      ! id setDefaultID   $nid

      ! queue setMode      "packet"
      ! queue setCapacity  40

      # disable ARP
      ! arp setBypassARP  [ expr 2>1]

      ! mac setRTSThreshold 0

      connect mobility/.report@ -and /fmRP/tracker/.node@
      connect wphy/down@ -to /fmRP/channel/.node@

      ! /fmRP/channel attachPort $i [! wphy getPort .channel]

      #                                    maxX maxY maxZ minX minY
minZ dX  dY  dZ
      ! mobility setTopologyParameters $gx  $gy  0.0  $gx  $gy  0.0
$gx $gy 0.0

      ! mac  disable_MAC_TRACE_ALL

      connect -c  wireless_agent/down@ -and pktdispatcher/1111@up

      cd ..

}
```

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


#*******************************************************************
*
# FOR THE SENSORS ONLY , do the following

for {set i [expr $sink_id + $ch_num + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {

      puts "create sensor $i"
      set node($i) [mkdir drcl.comp.Component n$i]

      cd n$i

      #*****************
      mkdir drcl.inet.sensorsim.MultiHop.MultiHopApp app
      ! app setNid $i
      ! app setSinkNid $sink_id
      ! app setCoherentThreshold 1000.0
      ! app setNn_ [expr $node_num - 1]
      ! app setfmRP
      #*************************
      #Routing metric
      #metric 1 = raw distance
      #metric 2 = energy
      ! app setmetric 2



      connect app/.getNeighbor@ -and /fmRP/nodetracker/.multiHop@
      #*****************

      # create nodes
      mkdir drcl.inet.sensorsim.SensorAgent agent

      ! agent setDebugEnabled 0

      # create sensor physical layers
      mkdir drcl.inet.sensorsim.SensorPhy phy
      ! phy setRxThresh 0.0
      ! phy setDebugEnabled 0

      # create mobility models
      mkdir drcl.inet.sensorsim.SensorMobilityModel mobility

      ! phy setNid $i
      ! phy setRadius [expr sqrt($gx*$gx+$gy*$gy)]

      # connect phyiscal layers to sensor agents so that nodes can
receive
      connect phy/.toAgent@ -to agent/.fromPhy@

      # connect sensor agent and sensor application
      connect agent/.toSensorApp@ -to app/.fromSensorAgent@

      # connect the sensor channel to the nodes so that they can
receive
      ! /fmRP/chan attachPort $i [! phy getPort .channel]
```

```
      # connect the nodes to the propagation model
      connect phy/.propagation@ -and /fmRP/seismic_Prop/.query@

      ! mobility setNid $i

      # create wireless agent layers
      mkdir drcl.inet.sensorsim.WirelessAgent wireless_agent

      # connect the sensor application to the wireless agent
      # so that sensors can send through the wireless network
protocol stack
      connect app/down@ -to wireless_agent/up@

      # connect the wireless agent to the sensor application
      # so that sensors can receive thru the wireless network
protocol stack
      connect wireless_agent/.toSensorApp@ -to
app/.fromWirelessAgent@

      mkdir drcl.inet.mac.LL ll
      mkdir drcl.inet.mac.ARP arp
      mkdir drcl.inet.core.queue.FIFO queue
      mkdir drcl.inet.mac.Mac_802_11 mac

      # added 09-04-04
      ! mac disable_PSM

      #*************
      #Required to let the MAC802.11 layer
      #be in charge of maintaining the
      #radio modes and energy levels.     ! mac setIs_uAMPS 1
      #*************

      mkdir drcl.inet.mac.WirelessPhy wphy

      # create a switched beam antenna for the node and initialize
the azimuth patter
      #inject "create SwitchedBeam Antenna" wphy/.antenna@
      #inject "azimuthpatterns = azimuth-simple-g4db" wphy/.antenna@

      #*************
      #fmakis
      #set sensing area and bandwidth with the new method
      ! wphy setsensingArea  $gx $gy

      ! wphy setMultiHopMode 1      ;#turn on MH mode settings
      #************

      mkdir drcl.inet.mac.FreeSpaceModel propagation


      set PD [mkdir drcl.inet.core.PktDispatcher      pktdispatcher]
    set RT [mkdir drcl.inet.core.RT                 rt]
    set ID [mkdir drcl.inet.core.Identity           id]

      ! pktdispatcher setRouteBackEnabled 1

      $PD bind $RT
      $PD bind $ID
```

```
#**************
#NICHOLAS
# create route configuration request for testing
#this is to define the interfaces. So in this case each sensor
#only has 1 interface (hence array size 1) and its eth0.
#another example is (which has 3 interfaces 0, 2, and 4:
#set ifs [java::new drcl.data.BitSet [java::new {int[]} 3 {0 2
4}]]

set ifs [java::new drcl.data.BitSet [java::new {int[]} 1 {0}]]
set base_entry [java::new drcl.inet.data.RTEntry $ifs]

set key [java::new drcl.inet.data.RTKey $i 0 -1]
set entry_ [!!! [$base_entry clone]]

! rt add $key $entry_

#**************
#            Above we inserted a permanent entry in the
#          table going to the sink. Here this contract is
#          needed so we can send directly to neighbors.

connect app/.setRoute@ -to rt/.service_rt@
#**************************

# present if using 802.11 power-saving mode
connect mac/.energy@ -and wphy/.energy@

#*****************************
connect app/.energy@ -and wphy/.appEnergy@
mkdir drcl.inet.sensorsim.CPUAvr cpu

connect app/.cpu@ -and cpu/.reportCPUMode@
connect cpu/.battery@ -and wphy/.cpuEnergyPort@
#******************************

connect wphy/.mobility@    -and mobility/.query@
connect wphy/.propagation@ -and propagation/.query@

connect mac/down@ -and wphy/up@
connect mac/up@   -and queue/output@

connect ll/.mac@ -and mac/.linklayer@
connect ll/down@ -and queue/up@
connect ll/.arp@ -and arp/.arp@

connect -c pktdispatcher/0@down -and ll/up@

set nid $i

! arp setAddresses  $nid $nid
! ll  setAddresses  $nid $nid
! mac setMacAddress $nid  ;#set MAC

! wphy setNid       $nid
! id setDefaultID   $nid

! queue setMode      "packet"
! queue setCapacity  40

# disable ARP
```

```
      ! arp setBypassARP  [ expr 2>1]

      ! mac setRTSThreshold 0

      connect mobility/.report@ -and /fmRP/tracker/.node@
      connect wphy/down@ -to /fmRP/channel/.node@

      ! /fmRP/channel attachPort $i [! wphy getPort .channel]

      #                                maxX maxY maxZ minX minY minZ
dX  dY  dZ
    ! mobility setTopologyParameters $gx  $gy  0.0  $gx  $gy  0.0
$gx $gy 0.0

      ! mac  disable_MAC_TRACE_ALL

      connect -c  wireless_agent/down@ -and pktdispatcher/1111@up

      cd ..
}


#*************************************************************
# FOR THE TARGET NODES ONLY , do the following
if { $target_node_num == 0 } {
      puts "No target agents .... "
} else {
      for {set i [expr $node_num - $target_node_num]} {$i <
$node_num} {incr i} {
            puts "create target $i"

            set node$i [mkdir drcl.comp.Component n$i]

            cd n$i

            # create target agents
            mkdir drcl.inet.sensorsim.TargetAgent agent
            ! agent setBcastRate 1.0
            ! agent setSampleRate 1.0

            # create sensor physical layers
            mkdir drcl.inet.sensorsim.SensorPhy phy
            ! phy setRxThresh 0.0
            ! phy setNid $i
            ! phy setRadius [expr sqrt($gx*$gx+$gy*$gy)]

            ! phy setDebugEnabled 0

            # create mobility models
            mkdir drcl.inet.sensorsim.SensorMobilityModel mobility

            # connect target agents to phy layers so that nodes can
send
            connect agent/down@ -to phy/up@

            # connect phy layers to sensor channel so that nodes can
send
            connect phy/down@ -to /fmRP/chan/.node@

            # connect the nodes to the propagation model
            connect phy/.propagation@ -and /fmRP/seismic_Prop/.query@
```

```
            ! mobility setNid $i

            # set the topology parameters
            ! mobility setTopologyParameters $gx  $gy  0.0  $gx  $gy
0.0

            cd ..
      }
}

#*********************************************************************
# for Cluster Heads SENSORs and TARGETs only. Not SINKs
for {set i [expr $sink_id + 1]} {$i < $node_num} {incr i} {
      # connect the mobility model of each node to the node position
tracker
      connect n$i/mobility/.report_sensor@ -and
/fmRP/nodetracker/.node@
      connect n$i/phy/.mobility@ -and n$i/mobility/.query@
}


#**********************************************
#Positioning
#

# set the position of sink nodes args=> (speed(m/sec),
xCoord,yCoord,zCoord

! $node(0)/mobility setPosition 0.0 $sinkX $sinkY $sinkZ

! n1/mobility setPosition 0.0 [expr 0.50*$gx] [expr 0.80*$gy] 0.0
! n2/mobility setPosition 0.0 [expr 0.50*$gx] [expr 0.22*$gy] 0.0
! n3/mobility setPosition 0.0 [expr 0.80*$gx] [expr 0.75*$gy] 0.0
! n4/mobility setPosition 0.0 [expr 0.80*$gx] [expr 0.33*$gy] 0.0
! n5/mobility setPosition 0.0 [expr 0.22*$gx] [expr 0.75*$gy] 0.0
! n6/mobility setPosition 0.0 [expr 0.22*$gx] [expr 0.25*$gy] 0.0

! n101/mobility setPosition 0.0 0.001 [expr $gy-0.001] 0.0
! n102/mobility setPosition 0.0 [expr $gx-0.001] [expr $gy-0.001] 0.0




# for the sensors They will be randomly placed on the grid (2D only)
# set the position of sensor nodes args=> (speed(m/sec),
xCoord,yCoord,zCoord
for {set i [expr $sink_id + $ch_num + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
   ! n$i/mobility setPosition 0.0 [expr rand()*$gx] [expr rand()*$gy]
0.0
}


# make all sensors aware of the position of the sink
for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
   ! n$i/app setSinkLocation $sinkX $sinkY $sinkZ
}

#**************************************************
```

```
#for sensor nodes only find the nearest cluster head
#
for {set i [expr $sink_id + $ch_num +1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {

            set nX [! n$i/mobility getX]
            set nY [! n$i/mobility getY]
            set nZ [! n$i/mobility getZ]
            set tempX [! n1/mobility getX]
            set tempY [! n1/mobility getY]
            set tempZ [! n1/mobility getZ]
            set mindist [expr sqrt(($tempX-$nX)*($tempX-$nX) +
($tempY-$nY)*($tempY-$nY) + ($tempZ-$nZ)*($tempZ-$nZ))]
            set destCH 1
            #puts "$i $mindist"

            for {set j [expr $sink_id + 2]} {$j < $ch_num+1} {incr j}
{
                  set tempX1 [! n$j/mobility getX]
                  set tempY1 [! n$j/mobility getY]
                  set tempZ1 [! n$j/mobility getZ]
                  set mindist1 [expr sqrt(($nX-$tempX1)*($nX-$tempX1)
+ ($nY-$tempY1)*($nY-$tempY1) + ($nZ-$tempZ1)*($nZ-$tempZ1))]

                  if {$mindist1 < $mindist} {
                  set mindist $mindist1
                  set destCH  $j
                  }
            }


            set chX [! n$destCH/mobility getX]
            set chY [! n$destCH/mobility getY]
            set chZ [! n$destCH/mobility getZ]
            ! n$i/app setch_nid $destCH
            ! n$i/app setdistanceToCH $mindist
            ! n$i/app setchPosX $chX
            ! n$i/app setchPosY $chY
            ! n$i/app setchPosZ $chZ
            ! n$i/app setNewNeighborID $destCH
            ! n$i/app setNewNeighborX $chX
            ! n$i/app setNewNeighborY $chY
            ! n$i/app setNewNeighborZ $chZ
            ! n$i/app setNewNeighborDist $mindist


            switch $destCH {
                  1 {lappend chlist1 $i}
                  2 {lappend chlist2 $i}
                  3 {lappend chlist3 $i}
                  4 {lappend chlist4 $i}
                  5 {lappend chlist5 $i}
                  6 {lappend chlist6 $i}
            }

            #lappend chlist$destCH $i
            #puts "appending $i to list chlist$destCH"
```

```
            #puts "node $i's destination is Cluster Head CH$destCH
with coordinates $chX $chY $chZ"


            #puts "node $i's neighbor is Cluster Head CH$destCH"
            #puts "node $i's distance to Cluster Head CH$destCH is
$mindist"

}

lappend tchlist1 1 $chlist1
lappend tchlist2 2 $chlist2
lappend tchlist3 3 $chlist3
lappend tchlist4 4 $chlist4
lappend tchlist5 5 $chlist5
lappend tchlist6 6 $chlist6
set chlist []
lappend chlist $tchlist1 $tchlist2 $tchlist3 $tchlist4 $tchlist5
$tchlist6

for {set i 0} {$i < [llength $chlist]} {incr i} {
      puts "Cluster [expr $i+1] members: [lindex $chlist $i]"
}

#***********************************************
#routeInfo() to execute do:
#     script "routeInfo" -at 0.35 -period 4.0 -on $sim
proc routeInfo { } {
   global sim n1
   puts "Current Route Table\n [! n1/rt info]"
}

#***********************************************
#Output remaining energy levels of the sensors to a plotter
set plot_ [mkdir drcl.comp.tool.Plotter .plot]
for {set i [expr $sink_id + $ch_num +1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
   connect -c n$i/app/.plotter@ -to $plot_/$i@0
}

#***********************************************
#Plotters for the SINK node

#Graph # 1
#To plot the total number of received packets at the sink
set sinkPlot1_ [mkdir drcl.comp.tool.Plotter .sinkPlot1]
connect -c n0/app/.PacketsReceivedPlot@ -to $sinkPlot1_/0@0

#Graph # 2
#Calculate the avg latency when the sink finally receives it
set sinkPlot2_ [mkdir drcl.comp.tool.Plotter .sinkPlot2]
connect -c n0/app/.latencyPlot@ -to $sinkPlot2_/0@0


#Graph # 3
#plot the actual phenomena being sensed.
! n$sink_id/app createSnrPorts $node_num $target_node_num
set sinkPlot3_ [mkdir drcl.comp.tool.Plotter .sinkPlot3]

for {set i 0} {$i < $target_node_num} {incr i} {
```

```
   connect -c n$sink_id/app/.snr$i@ -to $sinkPlot3_/$i@$i
   if { $testflag } {
       attach -c $testfile/in@ -to n$sink_id/app/.snr$i@
   }
}


#*************************************************
#wsnLoop()
#
#     This method is called periodically to check
#     if the simulation should continue or not. If
#     all nodes are dead or simtime>maxsimtime then stop the
simulator and
#     display the cummulative statistics otherwise keep running
proc wsnLoop { } {
      global sim node_num node sink_id target_node_num maxsimtime
ch_num

      #reset variables
      set live_sensors 0
      set dead_sensors 0
      set total_packets 0
      set total_packetsCH 0
      set dropped_packets 0


      #check how many are still alive
      for {set i [expr $sink_id + $ch_num + 1]} {$i < [expr $node_num
- $target_node_num ]} {incr i} {
            if { [! $node($i)/app isSensorDead] == 0 } {
                  incr live_sensors
            } else {
                  incr dead_sensors
            }

      }
      script [! liveSensors setLiveNodes $live_sensors]
      script [! liveSensors updateGraph]

      #display statistics if they are all dead.
      if { $dead_sensors >= [expr 1*($node_num - $target_node_num -
$ch_num - 1)] || [! $sim getTime] > $maxsimtime} {
         #puts "All nodes dead at [! $sim getTime]"
         puts "$dead_sensors out of [expr $node_num -
$target_node_num - $ch_num - 1] nodes dead at [! $sim getTime]"
         $sim stop
         puts "-----------------------------------------------"
         puts "Simulation Terminated\n"
         puts "Results:"
         puts "Base Station Received [! n0/app getTotalINPackets]"
         puts "Collisions at Base Station: [! n0/mac getCollision]"

        for {set i [expr $sink_id + $ch_num + 1]} {$i < [expr
$node_num - $target_node_num ]} {incr i} {
              set curr_packets [! n$i/app geteID]
            puts "Sensor$i Sent $curr_packets Packets to the Cluster
Head"
            set total_packets [expr $total_packets + $curr_packets]
         }
```

```
         for {set i [expr $sink_id + 1]} {$i < [expr $ch_num + 1 ]}
{incr i} {
            set curr_packets [! n$i/app geteID]
            puts "CH$i Sent $curr_packets Packets to BS"
            set total_packetsCH [expr $total_packetsCH +
$curr_packets]
              }

         set app_dropped [! n1/app getDropped_packets]
         set wphy_dropped [! n1/wphy getDropped_packets]
         #set mac_dropped [! n1/mac getDropped_packets]

         puts "Total packets dropped at Application layer:
$app_dropped"
         puts "Total packets dropped at physical layer:
$wphy_dropped"
         #puts "Drops due to collisions (discovered at MAC layer:
$mac_dropped"
         #set dropped_packets [expr $app_dropped + $wphy_dropped +
$mac_dropped]
         set dropped_packets [expr $app_dropped + $wphy_dropped]
         puts "Average Latency on BS is: [! n0/app getAvgLatency]"
         #set total_packets [expr $total_packets + $total_packetsCH]
         puts "Total Packets sent from all nodes: $total_packets"
         puts "Total Packets sent from all Cluster Heads:
$total_packetsCH"
         puts "Number of Dropped Packets: $dropped_packets"
         puts "Success Rate: [expr ([! n0/app getTotalINPackets].0 /
[expr $total_packets.0 + $total_packetsCH.0]) * 100]"
      }
}



#***********************************************
#sensorLocPrintOut()
#     Goes throught all sensors and prints their
#     (X,Y,Z) Coordinates
proc sensorLocPrintOut { } {
      global sink_id node_num target_node_num
      for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
            script [! n$i/app printNodeLoc]
      }
}

#***********************************************
#Output sensor location and status
#to a file for GUI to read from
proc file_output { } {
   global sink_id node_num count target_node_num ch_num

   #open a file for writting
   set filename "sensorInfo$count.log"
   set out [open $filename w]

   for {set i [expr $sink_id + $ch_num + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
     #get its x and y location
     set Xloc [! n$i/app getX]
     set Yloc [! n$i/app getY]
```

```
     #get its energy
     set status [expr [! n$i/app isSensorDead] - 0]

     #write it to the file
     puts $out "$Xloc $Yloc $status"
   }
   incr count

   close $out
}

#**********************************************
#Output sensor location and status
#to a file for GUI to read from
proc energy_dist { } {
   global sink_id node_num count target_node_num ch_num

   #open a file for writting
   set filename "energyDistrNode.log"
   set out [open $filename w]

   puts $out "TxCost     RxCost    IdleCost    SleepCost    CPUcost"

   for {set i [expr $sink_id + $ch_num + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
      #collect the data
      set txCost [! n$i/wphy getRadioTotalTX]
      set rxCost [! n$i/wphy getRadioTotalRX]
      set idleCost [! n$i/wphy getRadioTotalidle]
      set sleepCost [! n$i/wphy getRadioTotalsleep]
      set cpuCost [! n$i/wphy getTotalCPU]

      #write it to the file
      puts $out "$txCost $rxCost $idleCost $sleepCost $cpuCost"
   }

   close $out
}

#**********************************************
#fmakis: find the nearest Cluster Head
proc findCH { } {
      global node_num node sink_id target_node_num ch_num

      for {set i [expr $sink_id + $ch_num +1]} {$i < [expr $node_num
- $target_node_num]} {incr i} {

            set nX [! n$i/mobility getX]
            set nY [! n$i/mobility getY]
            set nZ [! n$i/mobility getZ]
            set tempX [! n1/mobility getX]
            set tempY [! n1/mobility getY]
            set tempZ [! n1/mobility getZ]
            set mindist [expr sqrt(($tempX-$nX)*($tempX-$nX) +
($tempY-$nY)*($tempY-$nY) + ($tempZ-$nZ)*($tempZ-$nZ))]
            set destCH 1
            #puts "$i $mindist"

            for {set j [expr $sink_id + 2]} {$j < $ch_num+1} {incr j}
{
```

```
                  set tempX1 [! n$j/mobility getX]
                  set tempY1 [! n$j/mobility getY]
                  set tempZ1 [! n$j/mobility getZ]
                  set mindist1 [expr sqrt(($nX-$tempX1)*($nX-$tempX1)
+ ($nY-$tempY1)*($nY-$tempY1) + ($nZ-$tempZ1)*($nZ-$tempZ1))]

                  if {$mindist1 < $mindist} {
                  set mindist $mindist1
                  set destCH  $j
                  }
            }
            script [! n$i/app setch_nid $destCH]

            #script [! n$i/app setNewNeighborID $destCH]

            set nX [! n$destCH/mobility getX]
            set nY [! n$destCH/mobility getY]
            set nZ [! n$destCH/mobility getZ]

            #script [! n$i/app setCHLocation $nX $nY $nZ]


            #script [! n$i/app setNewNeighborX $nX]
            #script [! n$i/app setNewNeighborY $nY]
            #script [! n$i/app setNewNeighborZ $nZ]
            #script [! n$i/app setNewNeighborDist $mindist]

            #script [! n$i/app setch_nid $destCH]
            #script [! n$i/app setdistanceToCH $mindist]
            #puts "node $i's destination is Cluster Head CH$destCH"
            #puts "node $i's neighbor is Cluster Head CH$destCH"
            #puts "node $i's distance to Cluster Head CH$destCH is
$mindist"

      }
}


#*************************************************
#Prints out each sensors neighbor at the present time
#
proc printNeighborList { } {
   global sink_id node_num count target_node_num ch_num

   for {set i [expr $sink_id + $ch_num + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
      script [! n$i/app printNeighborID]
   }
   puts "\n"
}


#*************************************************
#To call this do:
# script "getQueueSize" -at 20.0 -period 1.0 -on $sim

proc getQueueSize { } {
   global sink_id node_num count target_node_num ch_num

   puts "\n-------\nQUEUE SIZES\n-------"
```

```
   for {set i [expr $sink_id + $ch_num + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
      puts "Queue size of sensor$i is:  [! n$i/queue getSize]"
   }

}

#***********************************************
#Find next hop neighbor-- node to whom always send data.
#Choose closest node that is in the direction of the base station.
#NOTE!  This algorithm assumes nodes know the location of all nodes
#near them.  In practice, this would require an initial set-up
#phase where this information is disseminated throughout the network
#and that each node has a GPS receiver or other location-tracking
#algorithms to determine node locations.
proc setNeighbor { } {
   global sink_id node_num count sensorList target_node_num ch_num

   for {set i [expr $sink_id + $ch_num + 1]} {$i < [expr $node_num -
$target_node_num ]} {incr i} {
      script [! n$i/app setNeighborCH]

      #we also need to store it in the script
      #to maintain their next neighbor when a
      #node dies.

    set selfID [! n$i/app getNid]
      set tempID [! n$i/app getNeighbor_id]
      set tempX  [! n$i/app getNeighborX]
      set tempY  [! n$i/app getNeighborY]
      set tempZ  [! n$i/app getNeighborZ]
      set tempDist [! n$i/app getNeighbor_dist]

      #inner list: [nid, neighborID, neigh_X, neigh_Y, neighZ,
tempDist]
      set iThSensor [list $selfID $tempID $tempX $tempY $tempZ
$tempDist]

      #append it to the main list.
      set sensorList [lappend SensorNeighbors $iThSensor]
   }
   #puts "Size of List is: [llength $sensorList]"
}

#***********************************************
proc neighborUpdate { } {
   global sink_id node_num count sensorList delNodes

   for {set i 0 } {$i < [llength $sensorList]} {incr i} {
     #extract the current element
      set current_node [lindex $sensorList $i]

      set newSID [lindex $current_node 0]
      set newNID [lindex $current_node 1]
      set newX  [lindex $current_node 2]
      set newY  [lindex $current_node 3]
      set newZ  [lindex $current_node 4]
      set newDist [lindex $current_node 5]

      #collect the data
      set remEnergy [! n$newSID/wphy getRemEnergy]
```

```
      if { $remEnergy <= 0.01 } {

      for {set j 0 } {$j < [llength $sensorList]} {incr j} {

       set currentID [lindex [lindex $sensorList $j] 0]
         set neighborID [lindex [lindex $sensorList $j] 1]

      #   puts "Checking Sensor$currentID Neighbor Settings"
           if { $currentID != $newSID } {
            if { $neighborID == $newSID} {
                 #update the nodes neighbor in Java
               script [! n$currentID/app setNewNeighborID $newNID]
             script [! n$currentID/app setNewNeighborX $newX]
             script [! n$currentID/app setNewNeighborY $newY]
             script [! n$currentID/app setNewNeighborY $newZ]
             script [! n$currentID/app setNewNeighborDist $newDist]

                 #update the nodes neighbor w/in script
             set iThSensor [list $currentID $newNID $newX $newY $newZ
$newDist]
             #append it to the main list.
               set sensorList [lreplace $sensorList $j $j $iThSensor]
            }
         }
      };#end inner for
   #     puts " inserting number $newSID into delNodes"
        set delNodes [lappend delNodes $newSID]

      };#end if energy = 0

   } ;#end outer for
   #Now remove the elements from sensorList
   for {set k 0 } {$k < [llength $delNodes]} {incr k} {

      set delID [lindex $delNodes $k] ;#which node is it
  #    puts "getting index of $delID"
      for {set z 0 } {$z < [llength $sensorList]} {incr z} {
         set currentID [lindex [lindex $sensorList $z] 0]
       if { $currentID == $delID } {
            set index  $z
       }
      }
      set sensorList [lreplace $sensorList $index $index]
   }
   set delNodes [] ;#clear that temp list.
}


#*********************************************
#Strictly used for validation purposes. This method
#can be called to get the energy at any random
#point of a particular node.
proc getCurrEnergy { } {

   global sink_id node_num count target_node_num

   puts "\n\n"

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
```

```
      #puts "-----------------------------------------------"

      #puts "Sensor$i radio in Idle mode used: [! n$i/wphy
getRadioTotalidle] joules"
      #puts "Sensor$i radio in Rx mode used: [! n$i/wphy
getRadioTotalRX] joules"
      #puts "Sensor$i radio in Tx mode used: [! n$i/wphy
getRadioTotalTX] joules\n"

      #puts "Sensor$i Radio was idle for: [! n$i/wphy
getTotalRadioIdleTime] sec"
      #puts "Sensor$i Radio was Tx for: [! n$i/wphy
getTotalRadioActiveTime] sec"
      #puts "Sensor$i Radio was Rx for: [! n$i/wphy
getTotalRadioRxTime] sec\n"

      #puts "Sensor$i CPU active used: [! n$i/wphy getTotalCPUactive]
joules"
      #puts "Sensor$i CPU sleep used: [! n$i/wphy getTotalCPUsleep]
joules\n"

      #puts "Sensor$i CPU was active for: [! n$i/wphy
getTotalCPUactiveTime] sec"
      #puts "Sensor$i CPU was sleep for: [! n$i/wphy
getTotalCPUsleepTime] sec\n"


      #puts "Sensor$i CPU Total used: [! n$i/wphy getTotalCPU] joules
\n"

      puts "Sensor$i has [! n$i/wphy getRemEnergy] joules remaining"

   }
}

#*****************************************************
#fmakis: for validation print the transmiting power
proc getPt { } {
   global sink_id node_num count target_node_num ch_num

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
      puts "node$i TX power = [! n$i/wphy getPt]"
   }
   puts "-----------------------------------------------"
}


#-------------------------


#***********************************************
puts "simulation begins...\n"
set sim [attach_simulator .]
#$sim setDebugEnabled 1
$sim stop

#******************start the sink************************
script {run n0} -at 0.001 -on $sim

#***************start the sensors************************
```

```
for {set i [expr $sink_id + 1]} {$i < $node_num} {incr i} {
      script puts "run n$i" -at 0.01 -on $sim
}

#*********print out all the node locations**************
script "sensorLocPrintOut" -at 0.03 -on $sim

#*********fmakis: find nearest CH**********************
#script "findCH" -at -0.1 -on $sim

#********determine who their neighbors are**************
script "setNeighbor" -at 0.2 -on $sim

#*******print off each of their respective neighbors****
script "printNeighborList" -at 0.3 -on $sim

#script "getPt" -at 20 -on $sim

#*******Check if Sensor Status*************************
script "wsnLoop" -at 10.0 -period 3.0 -on $sim

#*******To maintain neighbor settings******************
script "neighborUpdate" -at 100.0 -period 0.5 -on $sim

#************For Matlab plotting***********************
#script "file_output" -at 1.0 -period 100.0 -on $sim

#the following line is only used in validation!
#script "getCurrEnergy" -at $maxsimtime -on $sim


#$sim resumeTo 10000.1
#Maximum simulation time as defined in wsnLoop()

$sim resumeTo [expr $maxsimtime+5]
```