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

Συσσώρευση και Επεξεργασία Δεδομένων Δικτύων Αισθητήρων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλέξανδρος Κ. Μηλαίος

Επιβλέπων : Νικόλαος Μήτρου
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2010

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

Συσσώρευση και Επεξεργασία Δεδομένων Δικτύων Αισθητήρων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλέξανδρος Κ Μηλαίος

Επιβλέπων : Νικόλαος Μήτρου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30η Ιουνίου 2010.

.....	.....	.....
Νικόλαος Μήτρου Καθηγητής Ε.Μ.Π.	Μιχάλης Θεολόγου Καθηγητής Ε.Μ.Π.	Συμεών Παπαβασιλείου Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2010

.....
Αλέξανδρος Κ. Μηλαίος

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

Περίληψη

Τα δίκτυα αισθητήρων κατά τα τελευταία χρόνια έχουν αναπτυχθεί θεαματικά. Το γεγονός αυτό οφείλεται στην ανάπτυξη νέων τεχνολογιών υλικού και στην υιοθέτηση νέων προτύπων και πρωτοκόλλων για την επικοινωνία, όπως το IPv6. Ωστόσο, μέχρι τώρα έμφαση έχει δοθεί στην δικτύωση των αισθητήρων και όχι στην διαχείριση και στην επεξεργασία των δεδομένων που συλλέγονται. Η ανάπτυξη αποδοτικών τεχνικών τόσο για την συσσώρευση όσο και για την επεξεργασία των δεδομένων είναι επιβεβλημένη. Επιπλέον, τα δεδομένα που παράγονται από τα δίκτυα αισθητήρων θα πρέπει να παρέχονται σε μια πρότυπη μορφή που θα τα καθιστά εκμεταλλεύσιμα και από περαιτέρω εφαρμογές. Το Open Geospatial Consortium (OGC) κινούμενο προς αυτή την κατεύθυνση δημιούργησε μια σειρά από πρότυπα, που συνθέτουν το Sensor Web Enablement (SWE). Λαμβάνοντας υπόψη τις παραπάνω εξελίξεις, στην παρούσα εργασία μελετάται η κατάλληλη και αποδοτική συσσώρευση, επεξεργασία και διαχείριση δεδομένων δικτύων αισθητήρων. Αρχικά, καταγράφονται τα θεωρητικά χαρακτηριστικά των δικτύων αισθητήρων και οι τεχνικές συσσώρευσης και επεξεργασίας δεδομένων, ενώ στη συνέχεια παρουσιάζεται μια υλοποίηση που πραγματοποιήθηκε με σκοπό την κατάλληλη επεξεργασία δεδομένων που συλλέγονται από ετερογενή δίκτυα αισθητήρων. Πιο αναλυτικά, αναπτύχθηκαν τα θέματα της συσσώρευσης, διαχείρισης και επεξεργασίας δεδομένων σε δίκτυα αισθητήρων ενώ παράλληλα μελετήθηκαν και θέματα δικτύωσης.

Λέξεις Κλειδιά: δίκτυο αισθητήρων, επεξεργασία δεδομένων, αναπαράσταση δεδομένων, SensorML, IPv6, 6LowPAN, 802.15.4

Abstract

Sensor networks have been developed dramatically during the last years. This is due to the fast development of modern technologies of hardware and due to the fact that revolutionary protocols and standards have been adopted. Nevertheless, focus has been given in the internetworking of sensors and not in the management of the data that sensors networks collect. Due to this fact, the development of efficient methods for data gathering and data processing is imperative. Moreover, the data that is produced by sensor networks, should be offered in a standardized format, in order to be exploitable from other applications. Towards this direction, the Open Geospatial Consortium (OGC) has created several standards, which compose the Sensoe Web Enablement (SWE). Taking into consideration the above developments, this diploma thesis focuses on appropriate and efficient methods for gathering, processing and managing of data in a wireless sensor network. Initially, the theoretical characteristics of sensors networks are presented in detail. Furthermore, all the techniques of data collection and processing have been studied and analyzed. At the same time, an implementation has been developed, which aims to efficiently gather and process the data within a wireless sensor network. Emphasis is given on data management isuues, but also at the same time networking aspects have been studied.

Keywords: sensor network, data processing, data representation, SensorML, IPv6, 6LowPAN, 802.15.4

Περιεχόμενα

1. Εισαγωγή

Τα δίκτυα αισθητήρων έχουν προσελκύσει το ενδιαφέρον τον τελευταίο καιρό και έχουν υιοθετηθεί από ένα ευρύ φάσμα εφαρμογών σε διάφορους τομείς όπως η υγεία, η διαχείριση της κίνησης, η πρόγνωση του καιρού και η παρακολούθηση δορυφόρων. Επιπλέον, ο αριθμός των δικτύων από αισθητήρες έχει αυξηθεί, κάτι που οφείλεται στην ύπαρξη μικρών, φθηνών και αξιόπιστων αισθητήρων που είναι ενεργειακά αυτάρκεις και έχουν επιπλέον την δυνατότητα ασύρματης επικοινωνίας και δικτύωσης. Η επικοινωνία ανάμεσα στους αισθητήρες είναι σημαντική, διότι παρέχει την δυνατότητα για τη διαχείριση και τη διαμόρφωση των δικτύων αισθητήρων ενώ συμβάλει και στη διασύνδεση των δικτύων αισθητήρων με το διαδίκτυο και την κατάλληλη παρουσίαση των δεδομένων σε αυτό, όπου είναι αναγκαίο. Επίσης, η υιοθέτηση του IPv6 παρέχει ένα τεράστιο χώρο διευθύνσεων για σκοπούς δικτύωσης, κάτι που χρειάζεται στα μεγάλα δίκτυα αισθητήρων σε παγκόσμια κλίμακα, ενώ ταυτόχρονα οδηγεί στην ταχεία ανάπτυξη πολλών χρήσιμων εφαρμογών. Έτσι, δεν είναι παράλογο να περιμένουμε ότι στο εγγύς μέλλον, σε πολλούς τομείς, θα αναπτυχθούν δίκτυα αισθητήρων τα οποία θα είναι προσβάσιμα μέσω του διαδικτύου. Έτσι τα αποθηκευμένα και πραγματικού χρόνου δεδομένα από αισθητήρες θα είναι διαθέσιμα σε όλο τον κόσμο, μέσα από διαμορφωμένες διεπαφές προγραμματισμού εφαρμογών (Application Programming Interfaces - APIs).

Ωστόσο, πάρα πολλή προσοχή έχει διατεθεί για τη δικτύωση των αισθητήρων ενώ πολύ μικρότερη για την ανάπτυξη εργαλείων που παρέχουν διαχείριση, ανάλυση και κατανόηση των δεδομένων που συλλέγονται. Για τον λόγο αυτό, πρέπει να αναπτυχθούν και να εφαρμοστούν κατάλληλες τεχνικές διαχείρισης δεδομένων, για να είναι εκμεταλλεύσιμα τα δεδομένα που συλλέγονται και να βελτιωθεί η διαλειτουργικότητα και η αποτελεσματική συνεργασία μεταξύ των κόμβων των δικτύων αισθητήρων. Προς αυτή την κατεύθυνση, η ομαδοποίηση των δεδομένων και η επεξεργασία τους που πρέπει να γίνεται με τρόπο που τα καθιστά πολύτιμα για τις εφαρμογές και τις παρεχόμενες υπηρεσίες σε ένα δίκτυο αισθητήρων. Οι αναπτυσσόμενες εφαρμογές θα πρέπει να είναι σε θέση να προβούν σε κατάλληλες ενέργειες ανάλογα με τα δεδομένα που συλλέγονται. Τέλος, κατά τον σχεδιασμό των συστημάτων διαχείρισης δεδομένων πρέπει να λαμβάνεται υπόψη ως περιοριστικός παράγοντας οι περιορισμένοι πόροι που έχουν οι αισθητήρες (χαμηλή μπαταρία, περιορισμένη δυνατότητα επεξεργασίας σήματος, περιορισμένη υπολογιστική ισχύς και δυνατότητες επικοινωνίας, μικρή ποσότητα μνήμης κ.α.) .

Προκειμένου να επιτευχθεί η κατάλληλη επεξεργασία των δεδομένων, αυτά πρέπει να συλλέγονται και να αποθηκεύονται πριν συσσωρευτούν. Διάφορες τεχνικές για ομαδοποίηση των δεδομένων έχουν προταθεί, οι οποίες διαχωρίζονται σύμφωνα με τον τύπο του δικτύου και με βάση τις απαιτήσεις που υπάρχουν. Ωστόσο, τα συγκεντρωτικά στοιχεία είναι ανεπεξέργαστα δεδομένα που έχουν μικρό νόημα από μόνα τους και ως εκ τούτου, είναι σημαντικό να ερμηνεύουν σύμφωνα με τις πληροφορίες που είναι σχετικές με τις προοριζόμενες εφαρμογές . Η σωστή αναπαράσταση και ερμηνεία των δεδομένων είναι σημαντική γιατί μπορεί να αυξήσει τη διαλειτουργικότητα μεταξύ των διαφόρων τύπων αισθητήρων και να παρέχει γενικές πληροφορίες για την κατάσταση του συνολικού δικτύου αισθητήρων. Επιπλέον, οι κατάλληλες μέθοδοι επεξεργασίας των δεδομένων μπορούν να αποδειχθούν εξαιρετικά χρήσιμες, ιδιαίτερα σε περιπτώσεις όπου τα δεδομένα προέρχονται από πολλές ετερογενείς πηγές και ενδέχεται να χρειαστεί να συγκριθούν ή να συνδυαστούν από διάφορες εφαρμογές. Για τον λόγο αυτό το Open Geospatial Consortium (OGC) ίδρυσε πρόσφατα το Sensor Web Enablement (SWE) με σκοπό να πετύχει τον παραπάνω στόχο. Στην προσπάθεια αυτή καθορίστηκαν οι προδιαγραφές που πρέπει να πληρούνται από τους κόμβους αισθητήρων, τυποποιήθηκε η μορφή αναπαράστασης των δεδομένων, και σχεδιάστηκαν οι υπηρεσίες διαδικτύου που θα επιτρέψουν την προσβασιμότητα και δυνατότητα ελέγχου των δεδομένων μέσω του Διαδικτύου. Επομένως, το Sensor Web αποτελεί μια ειδικού τύπου υποδομή, βασισμένη στο Διαδίκτυο, για την συλλογή, μοντελοποίηση, αποθήκευση, ανάκληση, διανομή, ανάλυση και οπτικοποίηση των δεδομένων και των φαινομένων που περιγράφουν.

Λαμβάνοντας υπόψη τις παραπάνω εξελίξεις, στην παρούσα εργασία μελετάται η κατάλληλη και αποδοτική συσσώρευση, επεξεργασία και διαχείριση δεδομένων δικτύων αισθητήρων. Αρχικά καταγράφονται τα θεωρητικά χαρακτηριστικά των δικτύων αισθητήρων και οι τεχνικές συσσώρευσης και επεξεργασίας δεδομένων, ενώ στη συνέχεια παρουσιάζεται μια υλοποίηση που πραγματοποιήθηκε με σκοπό την κατάλληλη επεξεργασία δεδομένων που συλλέγονται από ετερογενή δίκτυα αισθητήρων.

Αναλυτικότερα, στο πρώτο μέρος παρουσιάζονται τα χαρακτηριστηκά των δικτύων αισθητήρων, οι περιοχές των εφαρμογών τους, οι τοπολογίες των δικτύων των αισθητήρων και αναπτύσσονται θέματα που σχετίζονται με τις τρέχουσες εξελίξεις στην τεχνολογία των δικτύων αισθητήρων. Ιδιαίτερη προσοχή έχει δοθεί στα θέματα της διαχείρισης δεδομένων, στις υπάρχουσες τεχνικές επεξεργασίας, το Sensor Web Enablement και την προτεινόμενη

γλώσσα μοντελοποίησης SensorML. Τέλος, γίνεται μια εκτεταμένη παρουσίαση των προτεινόμενων πρωτοκόλλων δρομολόγησης και συγκέντρωσης δεδομένων. Στο δεύτερο μέρος, αναπτύσσεται και αναλύεται η υλοποίηση που έγινε στα πλαίσια της διπλωματικής εργασίας. Αρχικά παρουσιάζεται η προτεινόμενη αρχιτεκτονική πάνω στην οποία βασίστηκε ο σχεδιασμό της υλοποίησης του δικτύου αισθητήρων. Στη συνέχεια δίνεται μια περιγραφή της υλοποίησης και αναλύονται οι κόμβοι και το πρότυπο της SensorML που χρησιμοποιήθηκε. Παρουσιάζονται οι τοπολογίες διασύνδεσης αισθητήρων και το λογισμικό που αναπτύχθηκε, καθώς και οι μετρήσεις και τα αποτελέσματα που προέκυψαν από την υλοποίηση.

2. Χαρακτηριστικά των Δικτύων Αισθητήρων

Ένα ασύρματο δίκτυο αισθητήρων αποτελείται από κόμβους που μπορούν να συλλέγουν πληροφορίες για το περιβάλλον που βρίσκονται, ενώ παράλληλα μέσα από τις ασύρματες συνδέσεις, που διαθέτουν, επικοινωνούν μεταξύ τους και δρομολογούν τα δεδομένα τους. Τα δεδομένα τους προωθούνται με ένα ή περισσότερα βήματα στους κόμβους συγκέντρωσης (sink nodes), οι οποίοι έχουν τη δυνατότητα να συνδέονται με άλλα δίκτυα, όπως το Διαδίκτυο. Τα χαρακτηριστικά των κόμβων που αποτελούν τα δίκτυα αισθητήρων διαφέρουν. Για παράδειγμα μπορεί να είναι στατικοί ή να κινούνται, μπορεί να ξέρουν ή όχι την θέση που βρίσκονται. Επίσης μπορεί ένα δίκτυο να είναι ομοιογενές ή να αποτελείται από διαφορετικού είδους κόμβους. Παρακάτω αναλύονται πιο εκτεταμένα τα χαρακτηριστικά των κόμβων.

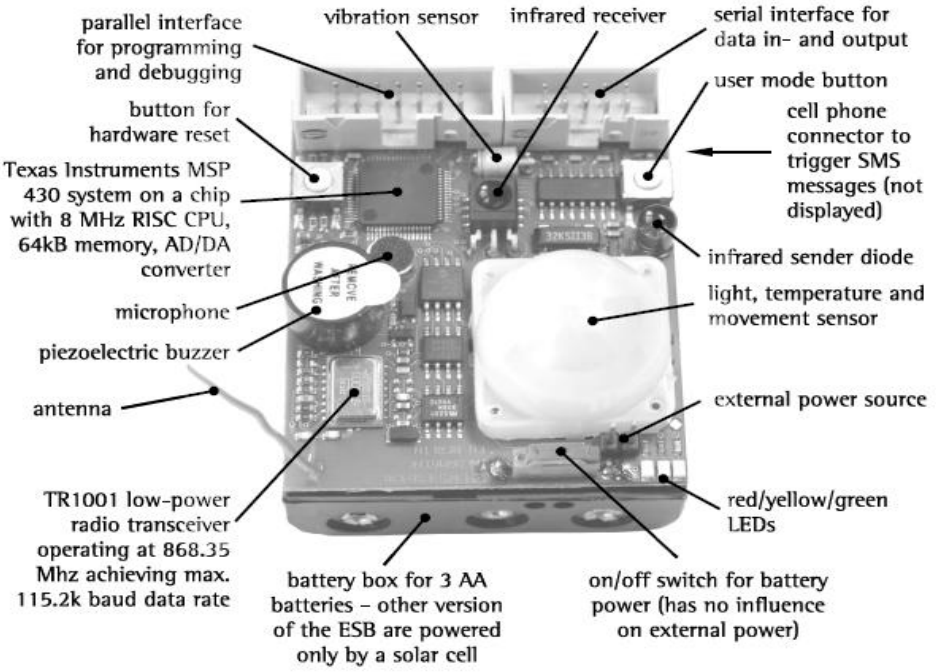
Οι σύγχρονοι κόμβοι μπορούν να συλλέγουν δεδομένα, να τα επεξεργάζονται , να τα αποθηκεύουν και να επικοινωνούν μεταξύ τους ή με κάποια κεντρική μονάδα. Αυτά μπορούν να τα κάνουν διότι διαθέτουν υπολογιστικές δυνατότητες και συγκεκριμένους ενεργειακούς πόρους. Τα μέρη που αποτελούν έναν αισθητήρα είναι και αυτά επιπλέον φαινονται στις εικόνες ένα και δυο:

- Η μονάδα επεξεργασίας η οποία όχι μόνο επεξεργάζεται τα δεδομένα άλλα και συντονίζει όλες τις άλλες μονάδες του αισθητήρα για να παρέχει την επιθυμητή λειτουργία. Συνήθως υποστηρίζεται με ένα chip μνήμης.
- Η μονάδα επικοινωνίας η οποία συμβάλλει στην συγκέντρωση και την συλλογή των δεδομένων σε κάποιους κεντρικούς κόμβους (sink nodes). Η μονάδα αυτή λειτουργεί

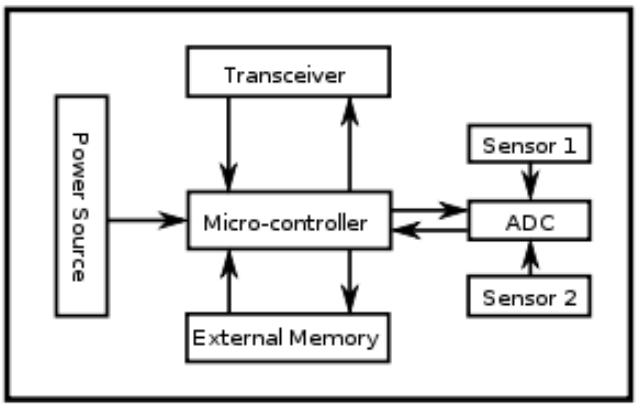
είτε μέσω των RF συχνοτήτων που είναι ελεύθερες είτε με υπέρυθρες που όμως απαιτούν οπτική επαφή και η επικοινωνία είναι ευαίσθητη στις ατμοσφαιρικές συνθήκες.

- Η μονάδα παροχής ενέργειας η οποία παρέχει ενέργεια στον αισθητήρα. Το μεγαλύτερο ποσοστό ενέργειας καταναλώνεται για την επικοινωνία του αισθητήρα με γειτονικούς αισθητήρες η με την κεντρική υποδομή. Ο κάθε αισθητήρας συλλέγει τα δεδομένα από το περιβάλλον και μετατρέπει τα αναλογικά σήματα σε ψηφιακά δεδομένα με τη χαμηλότερη δυνατή κατανάλωση ενέργειας.

Οι σύγχρονοί κόμβοι θα πρέπει να έχουν χαμηλό κόστος, να κάνουν πολλές λειτουργίες, να συνεργάζονται και να έχουν μικρό μέγεθος. Τα παραπάνω χαρακτηριστικά θα πρέπει να γίνουν με χαμηλή κατανάλωση ενεργείας. Το ζήτημα της ενεργειακής επάρκειας είναι το πιο σημαντικό στην σχεδίαση και την λειτουργία ενός κόμβου. Εξαιτίας της τυχαίας τοποθέτησης των κόμβων στην προς παρακολούθηση περιοχή δεν είναι εύκολη η ανανέωση της μπαταρίας των κόμβων και έτσι είναι προτιμότερο να τοποθετηθούν νέοι κόμβοι. Η χρήση νέων τεχνολογιών για την αποθήκευση ενεργείας αυξάνουν το κόστος και έτσι η έρευνα έχει στραφεί προς τις ανανεώσιμες πηγές ενεργείας, που όμως ακόμα δεν είναι ευρέως διαδεδομένες.



Εικόνα 1: Δομικά Μέρη ενός Κόμβου



Εικόνα 2: Διάγραμμα Λειτουργίας Κόμβου

Στη συνέχεια αναλύονται τα γενικά χαρακτηριστικά και οι δυνατότητες που πρέπει να προσφέρουν τα δίκτυα αισθητήρων:

- Οι κόμβοι έχουν περιορισμένους ενεργειακούς πόρους. Αυτό έχει σαν αποτέλεσμα την ανάγκη για βελτιστοποίηση των λειτουργιών της συγκέντρωσης και της αποθήκευσης δεδομένων και κάνει αυτές τις λειτουργίες καθοριστικές για την επιτυχία του δικτύου.
- Τα δίκτυα θα πρέπει να είναι ανθεκτικά, σε περίπτώση απώλειας κόμβων, δηλαδή θα πρέπει να ολοκληρώνονται οι λειτουργίες του δικτύου παρά την απώλεια ενός η περισσότερων εξ' αυτών. Η απώλεια των κόμβων μπορεί να οφείλεται είτε στις άσκημες καιρικές συνθήκες που προκαλούν πρόβλημα στις τηλεπικοινωνίες είτε στην περιορισμένη ισχύ που διαθέτουν οι κόμβοι.
- Τα δίκτυα των αισθητήρων θα πρέπει να αντιμετωπίσουν και την κινητικότητα των κόμβων. Θα πρέπει να αναπτυχθούν αλγόριθμοι για την αποδοτική συγκέντρωση δεδομένων, όταν οι κόμβοι δεν είναι σταθεροί αλλά κινούνται. Όταν αλλάζει η τοπολογία του δικτύου, είναι απαραίτητο το δίκτυο να προσαρμόζεται σε κάθε εξέλιξη που συμβαίνει.
- Η ετερογένεια των κόμβων είναι ένα χαρακτηριστικό αρκετά σημαντικό, που καθορίζει τη μορφή, τη δομή και την λειτουργία των δικτύων αισθητήρων. Με αλλά λόγια είναι σημαντικό να ενσωματωθούν σε ένα δίκτυο αισθητήρες διαφορετικής μορφής και ακόμα είναι απαραίτητο να υπάρχει συμβατότητα μεταξύ δικτύων που αποτελούνται από διαφορετικούς κόμβους. Ένα άλλο χαρακτηριστικό είναι και η μεγάλη διαφοροποίηση των κόμβων που αποτελούν το δίκτυο. Με άλλα λόγια χρειάζεται να

αναπτυχθούν πολλοί κόμβοι με διαφορετικές ικανότητες για διαφορετικές λειτουργίες που απαιτεί το δίκτυο.

- Τα δίκτυα των αισθητήρων θα πρέπει να είναι αυτό-διαχειριζόμενα. Οι λειτουργίες που εξυπηρετούν τα δίκτυα αυτά απαιτούν αυτό το χαρακτηριστικό και ο σχεδιασμός τους πρέπει να δίνει αυτή τη δυνατότητα
- Τέλος ένα άλλο χαρακτηριστικό των δικτύων αισθητήρων είναι ο μικρός ρυθμός μεταφοράς δεδομένων που πρέπει να δρομολογούνται μέσα στο δίκτυο. Τόσο η απαίτηση για χαμηλή κατανάλωση ενέργειας, όσο και το μικρό εύρος ζώνης που έχουν στη διάθεση τους είναι οι αιτίες για την ύπαρξη αυτού του χαρακτηριστικού.

Τα δίκτυα αισθητήρων διαφέρουν από τα κλασσικά δίκτυα επικοινωνιών καθώς ο βασικός τους στόχους είναι η μεγιστοποίηση του χρόνου ζωής τους και όχι η παροχή βέλτιστης ποιότητας υπηρεσίας (QoS). Υπάρχουν ακόμα όμως και αισθητήρες πολυμέσων που απαιτούν από το δίκτυο αισθητήρων χαρακτηριστικά όμοια με αυτά των δικτύων επικοινωνιών. Τα δίκτυα αυτά πρέπει να είναι ανθεκτικά σε προβλήματα που μπορούν να προκύψουν. Τα προβλήματα αυτά συμβαίνουν λόγω αλλαγών στο φυσικό περιβάλλον και του δυναμικού τρόπου που συνδέονται και αποσυνδέονται οι κόμβοι στο δίκτυο. Επιπλέον, πρόβλημα στο δίκτυο μπορεί να δημιουργηθεί όταν τελειώσουν οι ενεργειακοί πόροι σε κάποιο αισθητήρα. Αυτά τα προβλήματα μπορούν να ξεπεραστούν από την χρήση κόμβων νέας γενιάς που έχουν ήδη αναπτυχθεί και από την κατάλληλη εγκατάσταση (τοπολογία και πυκνότητα) των κόμβων στα δίκτυα αισθητήρων. Σε πυκνά δίκτυα οι αποστάσεις μεταξύ κόμβων είναι μικρές και έτσι επιτυγχάνεται μικρότερη κατανάλωση ενέργειας λόγω μετάδοσης των δεδομένων.

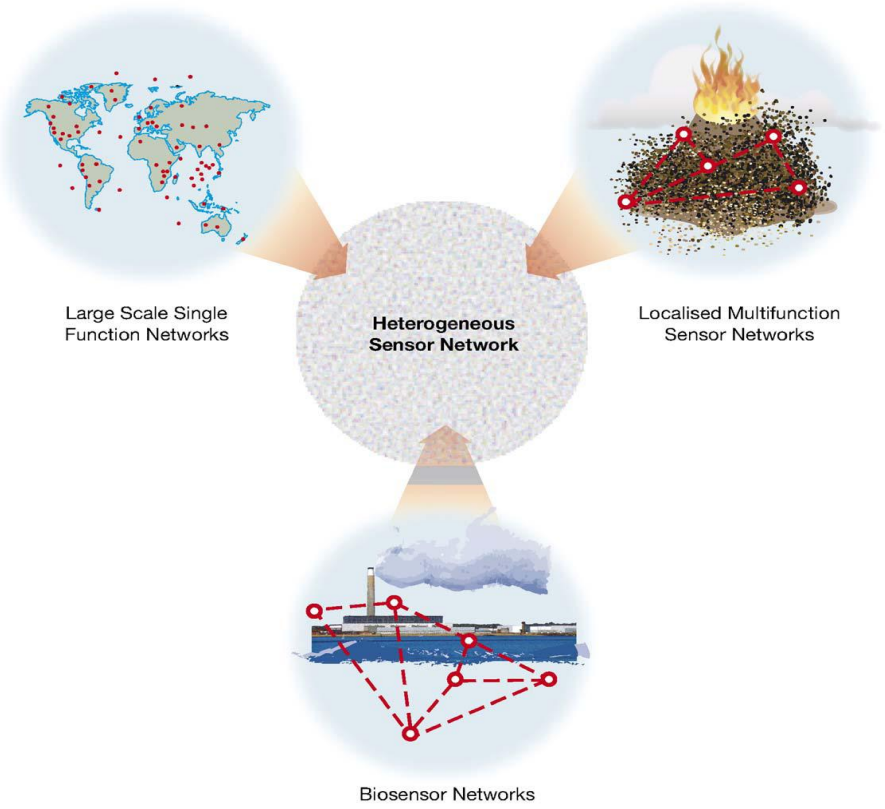
2.1 Περιοχές Εφαρμογών

Στο παρακάτω κεφάλαιο αναλύονται πιο εκτεταμένα οι περιοχές των εφαρμογών των δικτύων των αισθητήρων. Τα δίκτυα αισθητήρων έχουν ευρεία εφαρμογή σε διάφορα πεδία όπως αυτά που φαίνονται στην εικόνα 3 όπου η διαχείριση δεδομένων είναι σημαντική. Κάποια από τα πεδία εφαρμογής είναι τα επόμενα :

- Χρησιμοποιούνται στον τομέα της υγείας. Υπάρχουν αισθητήρες που συλλέγουν βιομετρικά δεδομένα και παρακολουθούν την εξέλιξη της υγείας των ασθενών. Ακόμα, με τα δίκτυα των αισθητήρων είναι δυνατόν να παρακολουθούνται οι επιδημίες και να βοηθούν τους γιατρούς να λάβουν τις σωστές αποφάσεις. Τέλος, μπορούν να

συμβάλουν στην αποτελεσματική διαχείριση των πόρων των νοσοκομείων.

- Ένας άλλος τομέας, στον οποίο βρίσκουν εφαρμογή τα δίκτυα αισθητήρων, είναι οι μετεωρολογικές και περιβαλλοντολογικές μετρήσεις. Οι περιβαλλοντολογικοί αισθητήρες χρησιμοποιούνται για την πρόγνωση του καιρού, την αντιμετώπιση των πυρκαγιών και της μόλυνσης του περιβάλλοντος, όπως επίσης για την παρακολούθηση της εξέλιξης του φαινομένου του θερμοκηπίου. Με τα δεδομένα που συλλέγονται λαμβάνονται κρίσιμες αποφάσεις.
- Στις βιομηχανικές εφαρμογές: διάφορα ήδη αισθητήρων εγκαθίστανται σε διάφορους τομείς της βιομηχανίας όπως αεροναυπηγική, οικοδομικές κατασκευές, επεξεργασία τροφίμων και αυτοκινητοβιομηχανία. Οι αισθητήρες αυτοί παρακολουθούν τα προϊόντα, από τη στιγμή της παραγωγής τους μέχρι την κατανάλωσή τους, ενώ υπάρχουν και άλλες περιοχές εφαρμογών των δικτύων των αισθητήρων στη βιομηχανία.
- Τα έξυπνα σπίτια είναι μια εξέλιξη που οφείλεται στα δίκτυα των αισθητήρων. Οι οικιακοί αυτοματισμοί δημιουργούν έξυπνες τεχνολογικές καινοτομίες που κάνουν τη ζωή του χρήστη πιο εύκολη. Ειδικοί αισθητήρες εγκαθίστανται στις οικιακές συσκευές για την παρακολούθησή τους και ίσως την, μέσω διαδικτύου, διαχείρισή τους. Επιπλέον, τα δίκτυα των αισθητήρων μπορούν να συμβάλουν στην παρακολούθηση και βελτίωση της ποιότητας ζωής των ηλικιωμένων.
- Ακόμα τα δίκτυα των αισθητήρων έχουν χρησιμότητα σε αμυντικές εφαρμογές. Τα δίκτυα αισθητήρων μπορούν να παρακολουθούν τις εχθρικές κινήσεις, ενώ έχουν και άλλες πολλές πιθανές στρατιωτικές εφαρμογές. Για παράδειγμα, ήδη χρησιμοποιούνται για τη διαχείριση συγκρούσεων, την αναγνώριση στόχων και την αξιολόγηση των καταστροφών.



Εικόνα 3: Περιοχές Εφαρμογών Δικτύων Αισθητήρων

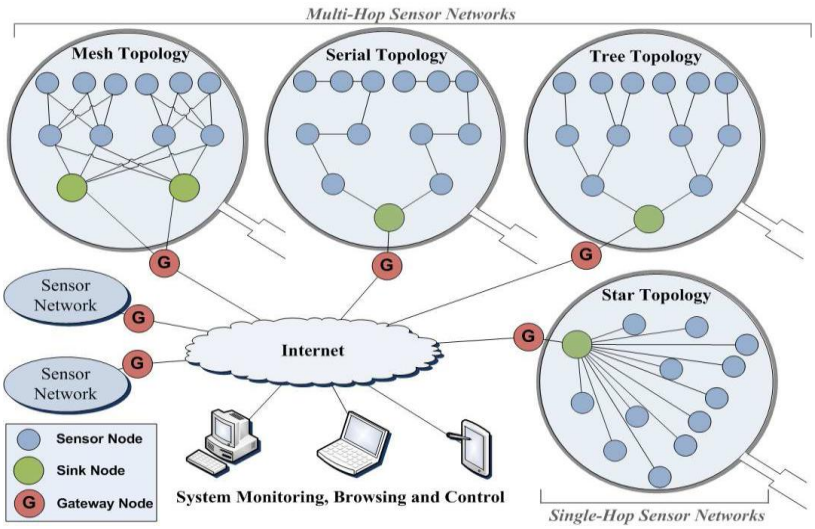
Επιπλέον οι εφαρμογές διαχωρίζονται σε δύο είδη ανάλογα με την μορφή των δεδομένων που συλλέγουν.

- Στις εφαρμογές που αναγνωρίζουν γεγονότα (event detection). Σε αυτές τις εφαρμογές οι κόμβοι εξετάζουν τα δεδομένα που λαμβάνουν και αν οι τιμές ξεπερνούν κάποιο όριο, τότε αποστέλλονται στους κόμβους συγκέντρωσης και στη συνέχεια αναλαμβάνει μια κεντρική δομή να εξετάσει τα συγκεκριμένα δεδομένα. Τέτοιες εφαρμογές χρησιμοποιούνται στην ανίχνευση συμβάντων, όπως για παράδειγμα φωτιά σε ένα δάσος.
- Στις εφαρμογές που χρησιμοποιούνται για την προσέγγιση χωρικών διαδικασιών (παρακολούθηση περιοχών και φαινομένων). Αυτές οι διαδικασίες έχουν τη μορφή στοχαστικών διαδικασιών. Δηλαδή, οι κόμβοι σε αυτήν την περίπτωση λαμβάνουν δεδομένα τα οποία επεξεργάζονται είτε από αυτούς είτε από κάποιο κεντρικό διαχειριστή με σκοπό να παραχθεί γνώση με βάση τα επεξεργασμένα δεδομένα. Η ακρίβεια των μετρήσεων εξαρτάται από την πυκνότητα των κόμβων.

Οι τομείς, που δύνανται να εκμεταλλευτούν προς όφελός τους τα δίκτυα των αισθητήρων, είναι αρκετοί. Όμως ο κάθε τομέας έχει τις δικές του απαιτήσεις από την εγκατάσταση και διαχείριση του εκάστοτε δικτύου αισθητήρων. Προκειμένου να βελτιστοποιηθούν τα χαρακτηριστικά των δικτύων αισθητήρων και να υπάρξει μια σύγκλιση μεταξύ των ετερογενών δικτύων απαιτείται περαιτέρω έρευνα.

2.2 Τοπολογίες Δικτύων Αισθητήρων

Στο επόμενο κεφάλαιο παρουσιάζεται μια σειρά από πιθανές τοπολογίες εγκατάστασης ενός δικτύου αισθητήρων (Εικόνα 4). Αρχικά, όταν ένα σύμπλεγμα από κόμβους αισθητήρων αναπτύσσεται και επικοινωνία εγκαθίσταται ανάμεσα σε αυτούς τους κόμβους, τότε προκύπτει ένα Ασύρματο Δίκτυο Αισθητήρων (Wireless Sensor Network – WSN). Τα δίκτυα αυτά αποτελούνται από απλούς κόμβους (nodes), κόμβους συγκέντρωσης δεδομένων (sink nodes) και κόμβους διασύνδεσης (Gateway Nodes) που συνδέουν τους κόμβους συγκέντρωσης με άλλα δίκτυα όπως το Διαδίκτυο. Συχνά, ο κόμβος διασύνδεσης και ο κόμβος συγκέντρωσης ταυτίζονται. Στη συνέχεια και ανάλογα με την τοπολογία, οι απλοί κόμβοι στέλνουν αμέσως (single-hop network) δεδομένα στο κόμβο συγκέντρωσης (τοπολογία αστέρα) ή επικοινωνούν με τους γειτονικούς κόμβους και έπειτα με τους κόμβους συγκέντρωσης (multi-hop network) πχ. Σειριακή τοπολογία γραμμή, τοπολογία δέντρου και mesh. Οι κόμβοι των δικτύων ενδέχεται να παρουσιάζουν και κινητικότητα. Στην επόμενη εικόνα φαίνονται κάποιες από αυτές τις τοπολογίες:



Εικόνα 4: Πιθανές Τοπολογίες Δικτύων Αισθητήρων

Στην τοπολογία αστέρα όλοι οι κόμβοι στέλνουν τα δεδομένα τους σε ένα κεντρικό κόμβο ο οποίος τα αποθηκεύει και είναι αυτός υπεύθυνος για την διαχείριση των δεδομένων και την απάντηση των ερωτημάτων που υποβάλλουν οι χρήστες. Από την άλλη πλευρά, στην τοπολογία δένδρου υπάρχει μια ιεράρχηση των κόμβων. Αυτοί επικοινωνούν μεταξύ τους και τελικά συγκεκριμένοι κόμβοι επικοινωνούν με τον κεντρικό κόμβο. Στην τοπολογία σειράς οι κόμβοι επικοινωνούν μεταξύ τους διαδοχικά και ο τελευταίος με τον κεντρικό κόμβο. Τέλος, υπάρχουν και οι ανακατεμένες τοπολογίες, όπου μπορεί να υπάρχει ή όχι ιεράρχηση των κόμβών,ενώ δεν υπάρχει καθορισμένος τρόπος μετάδοσης των δεδομένων στους κεντρικούς κόμβους. Στην περίπτωση αυτή δυναμικά πρωτόκολλα δρομολόγησης και διαχείρισης δεδομένων πρέπει να εφαρμοσθούν, ώστε να γίνονται αντιληπτές οι αλλαγές στην τοπολογία του δικτύου αισθητήρων.

2.3 Εξελίξεις στα Δίκτυα Αισθητήρων

Διαδίκτυο των Πραγμάτων - Internet of Things

Στο επόμενο κεφάλαιο αναλύεται ο όρος «Διαδίκτυο των Πραγμάτων» (Internet of Things) και πως αυτός συνδέεται με τα δίκτυα των αισθητήρων. Παράλληλα δίνεται και η πιθανή μορφή που θα έχουν τα δίκτυα, ενώ επίσης αναλύονται τα θέματα που ανακύπτουν από την ενσωμάτωση των δικτύων των αισθητήρων στο Διαδίκτυο. Τέλος, δίνονται γενικές πληροφορίες για το «Διαδίκτυο των Πραγμάτων» και της πιθανές εφαρμογές του.

Το «Διαδίκτυο των Πραγμάτων» είναι εξέλιξη του σημερινού Διαδικτύου και αποτελείται από παγκόσμια διασυνδεδεμένα δίκτυα. Τα μέρη αυτών των δικτύων θα είναι αντικείμενα που θα έχουν μοναδικές διευθύνσεις και θα βασίζονται πάνω σε συγκεκριμένα πρωτόκολλα επικοινωνίας. Για να δοθούν τόσες διευθύνσεις είναι απαραίτητο να αυξηθεί ο χώρος των διευθύνσεων και άρα είναι σημαντική η υιοθέτηση του πρωτοκόλλου IPv6. Η ενσωμάτωση των δικτύων των αισθητήρων στο νέο αυτό είδος του διαδικτύου δίνει νέες δυνατότητες στις προσφερόμενες υπηρεσίες από τα δίκτυα των αισθητήρων και το διαδίκτυο αλλά παράλληλα δημιουργεί κάποια θέματα που χρειάζονται αντιμετώπιση.

Για την ενσωμάτωση των δικτύων των αισθητήρων στο «Διαδίκτυο των Πραγμάτων» θα πρέπει να ληφθούν υπόψη τα ποικίλα σενάρια εφαρμογών των δικτύων αισθητήρων, δηλαδή θα πρέπει να βρεθεί ο κατάλληλος τρόπος για την ενσωμάτωση όλων των ειδών των δικτύων αισθητήρων στο Διαδίκτυο. Σήμερα υπάρχουν τρεις διαφορετικές τεχνικές για την επικοινωνία των δικτύων των αισθητήρων με το διαδίκτυο. Αρχικά, η

πρώτη προσέγγιση προβλέπει την ύπαρξη μόνο μιας διεξόδου από το δίκτυο των αισθητήρων προς το διαδίκτυο (gateway). Σύμφωνα με την δεύτερη προσέγγιση υπάρχουν περισσότερες από μια πύλες επικοινωνίας με το διαδίκτυο. Τέλος, η τρίτη προσέγγιση προβλέπει την άμεση πρόσβαση από τους κόμβους του δικτύου των αισθητήρων προς το διαδίκτυο με ένα μόλις βήμα, δηλαδή προβλέπει την άμεση επικοινωνία των κόμβων με τις πύλες εξόδου προς το διαδίκτυο. Κάθε μια από τις παραπάνω τεχνικές είναι κατάλληλη για κάποιο συγκεκριμένο είδος εφαρμογών.

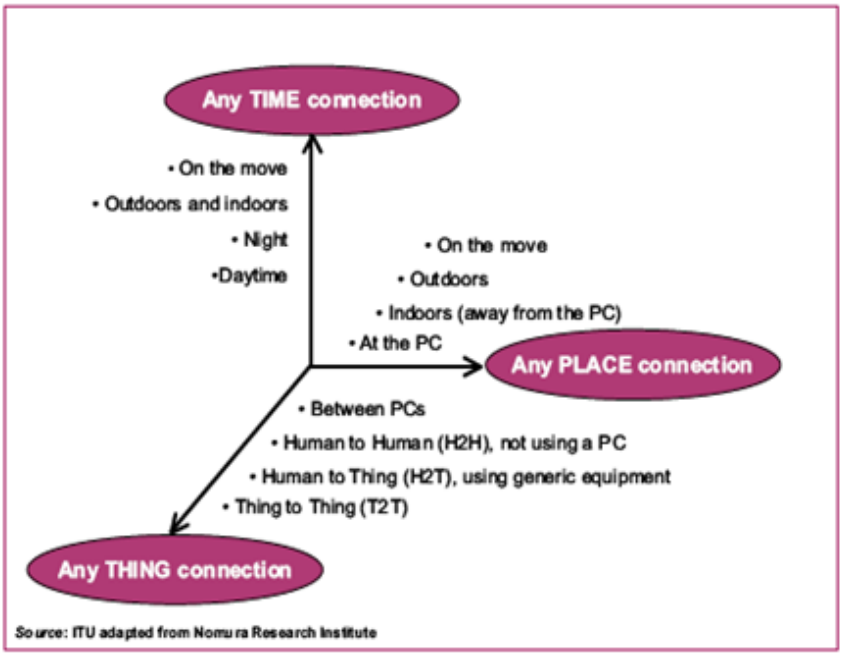
Για να καταστεί δυνατή η ενσωμάτωση των δικτύων των αισθητήρων στο «Διαδίκτυο των Πραγμάτων» θα πρέπει οι όλοι κόμβοι των δικτύων να αναβαθμιστούν έτσι ώστε να είναι δυνατή η άμεση επικοινωνία τους και η αυτόνομη σύνδεση τους με το διαδίκτυο. Επιπλέον, θα πρέπει να πάρουν διευθύνσεις και να αναπτύξουν προηγμένες δυνατότητες δρομολόγησης κάτι που δεν τους το επιτρέπει το υλικό τους σήμερα. Αυτή όμως η αναβάθμιση έχει ως αποτέλεσμα τη δημιουργία μερικών άλλων θεμάτων. Αρχικά, οι κόμβοι θα πρέπει να αναπτύξουν κάποιους μηχανισμούς ασφάλειας και προστασίας από επιθέσεις. Στη συνέχεια, θα πρέπει οι κόμβοι να αναβαθμιστούν έτσι ώστε να συνεργάζονται και να εκτελούν σύνθετες εργασίες προκειμένου να βελτιωθεί η ποιότητα των υπηρεσιών που προσφέρονται (QoS). Τέλος, θα πρέπει να βρεθούν μέθοδοι ώστε να αυτό-οργανώνεται το δίκτυο. Για παράδειγμα να μπορεί ο κάθε κόμβος να αντιλαμβάνεται πιθανές δυσλειτουργίες και να προβαίνει σε ενέργειες ώστε να επιδιορθώνει το δίκτυο.

Από τα παραπάνω είναι κατανοητό ότι απαιτείται περισσότερη μελέτη για τον τρόπο με τον οποίο θα γίνει η ενσωμάτωση των δικτύων των αισθητήρων στο «Διαδίκτυο των Πραγμάτων». Στη συνέχεια ακολουθεί μία πιο γενική ανάλυση του όρου «Διαδίκτυο των Πραγμάτων».

Με τον όρο «Διαδίκτυο των Πραγμάτων» αναφερόμαστε σε ένα δίκτυο που αποτελείται από διάφορα αντικείμενα, όπως οι οικιακές συσκευές. Αυτό το δίκτυο είναι αυτό-οργανωμένο και ασύρματο. Η έννοια «Διαδίκτυο των Πραγμάτων» αρχικά αποδίδεται στο Auto-ID Center που δημιούργησε το MIT. Η ιδέα είναι απλή αλλά και πολύ δύσκολη στην εφαρμογή της. Αν όλα τα αντικείμενα ήταν εξοπλισμένα με κάποιου είδους αισθητήρες, τότε θα έκανε την καθημερινότητα των ανθρώπων να αλλάξει προς το καλύτερο. Δεν θα υπήρχε ποτέ πρόβλημα με τα αποθέματα πόρων και προϊόντων καθώς και με τη διαχείρισή τους, αφού θα είναι δυνατόν να γνωρίζει κάποιος τί καταναλώνεται παγκοσμίως.

Σε τελική ανάλυση το «Διαδίκτυο των Πραγμάτων» είναι ένα μη ντετερμινιστικό, ανοικτό και αυτό-οργανωμένο δίκτυο με έξυπνους κόμβους που αλληλεπιδρούν με το περιβάλλον τους. Η μορφή με την οποία θα λειτουργήσει το «Διαδίκτυο των Πραγμάτων» είναι παρόμοια με την τεχνολογία «Σημασιολογικού Ιστού» (Semantic Web) και βασίζεται στην αναγνώριση του περιβάλλοντος από τις συσκευές και τον καθορισμό των γεγονότων που συμβαίνουν σε αυτό.

Το «Διαδίκτυο των Πραγμάτων» προσθέτει μια νέα διάσταση στον κόσμο της πληροφορικής και στις τεχνολογίες επικοινωνίας. Η συνδεσιμότητα επεκτείνεται και στα αντικείμενα τα οποία μπορούν να αλληλεπιδρούν με τους ανθρώπους ή και μεταξύ τους (εικόνα 5) .



Εικόνα 5: Διαστάσεις στον Κόσμο των Επικοινωνιών

Το Internet of Things για να αναπτυχθεί βασίζεται στην ανάπτυξη της τεχνολογίας των αισθητήρων που τους δίνει τόσο την ικανότητα να αντιλαμβάνονται το περιβάλλον τους όσο και την δυνατότητα να έχουν μεγαλύτερη υπολογιστική ισχύ. Τέλος, με την χρήση της νανοτεχνολογίας παράγονται όλο και μικρότερες συσκευές με αυξημένες δυνατότητες. Οι παραπάνω τεχνολογίες καθιστούν δυνατή την ανάπτυξη των εφαρμογών όπως τα έξυπνα σπίτια που διαθέτουν ικανότητες αυτοδιαχείρισης (εικόνα 6).

Εικόνα 6: Εφαρμογή του IoT στα Εξύπνα Σπίτια

Πρωτόκολλο IPv6

Όπως αναφέρθηκε και παραπάνω το πρωτόκολλο IPv6 έχει ως στόχο την αύξηση του χώρου των διευθύνσεων και κατά συνέπεια την ενσωμάτωση των δικτύων των αισθητήρων στο Διαδίκτυο. Το πρωτόκολλο IPv6 έχει σχεδιαστεί για να αντικαταστήσει το IPv4 και να επιτρέψει στο Διαδίκτυο την περαιτέρω ανάπτυξη για τις επόμενες δεκαετίες. Οι συσκευές και οι εφαρμογές αναμένεται τα επόμενα χρόνια να υπερβούν τον αριθμό των διαθέσιμων διευθύνσεων κάνοντας επιτακτική την ανάγκη για διεύρυνση του χώρου των διευθύνσεων. Επιπλέον, αυξάνεται και το ελάχιστο μήκους του πακέτου που μεταφέρεται από το στρώμα δικτύου κάτι που επιτρέπει την υιοθέτηση μεγαλύτερων διευθύνσεων. Επίσης, το γεγονός ότι ο κατακερματισμός των πακέτων πραγματοποιείται στο τελικό σημείο προορισμού και όχι στους ενδιάμεσους δρομολογητές, απλοποιεί τη διαδικασία της δρομολόγησης και επιτρέπει την εφαρμογή του IPv6. Ακόμα, για να βελτιωθεί η απόδοση του IPv6 ενσωματώθηκε σε αυτό η δυνατότητα στοχευμένης multicast εκπομπής. Τέλος, στο πρωτόκολλο αυτό δίνεται η δυνατότητα στους κόμβους να προσδιορίζουν αυτοί δυναμικά τις διευθύνσεις τους.

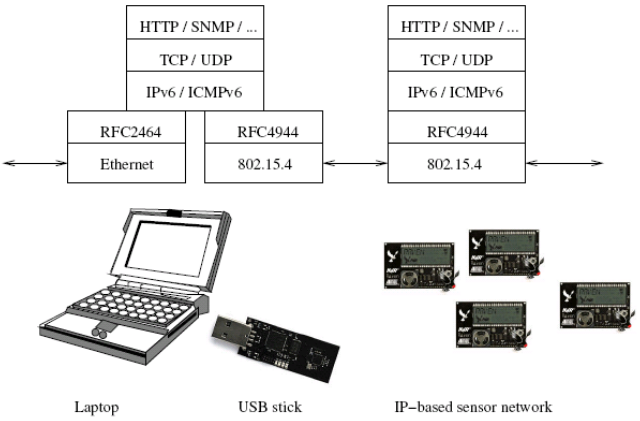
Για να εφαρμοστεί το IPv6 στα δίκτυα των αισθητήρων θα πρέπει να αντιμετωπιστούν κάποια θέματα. Στα δίκτυα αισθητήρων εφαρμόζεται το πρωτόκολλο IEEE 802.15.4, το οποίο είναι ειδικά σχεδιασμένο για να εξυπηρετεί τις ανάγκες των LoWPAN δικτύων, όπως τα δίκτυα αισθητήρων. Τα δίκτυα αυτά είναι σχεδιασμένα να έχουν μεγάλη περίοδο ζωής .Οι κόμβοι τους δεν πρέπει να καταναλώνουν πολύ ενέργεια, οι

συνδέσεις τους έχουν χαμηλούς ρυθμούς μετάδοσης και τα πλαίσια τους έχουν μικρό μήκος για να υπάρχει χαμηλός αριθμός λαθών. Οι κόμβοι έχουν μικρή υπολογιστική ισχύ και δεν μπορούν να κάνουν σύνθετες διαδικασίες, ενώ οι διευθύνσεις τους είναι 16 bit για να μειωθεί το μήκος της επικεφαλίδας. Τέλος, η ακτίνα διάδοσης είναι πολύ μικρή και τα πακέτα προωθούνται σε πολλά βήματα (multi-hop). Λόγω των παραπάνω περιορισμών, η εφαρμογή του IPv6 πάνω από το IEEE 802.15.4 απαιτεί την αντιμετώπιση των παρακάτω θεμάτων :

- Τα πακέτα του IPv6 είναι μεγάλα για τον χειρισμό τους από τα LoWPAN δίκτυα.
- Το 802.15.4 πρέπει να προσαρμόζεται σε ένα δυναμικά μεταβαλλόμενο περιβάλλον εξαιτίας του μικρού ρυθμού μετάδοσης και της χαμηλής κατανάλωσης ενέργειας.
- Ένα LoWPAN δίκτυο παρουσιάζει μια σύνθετη τοπολογία που αποτελείται από μικρής εμβέλειας συνδέσεις.

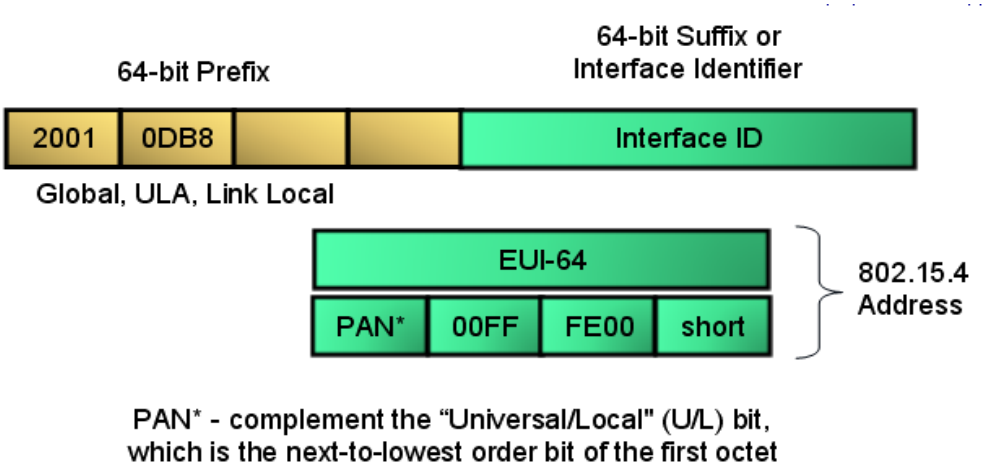
Για να γίνει δυνατή η εφαρμογή του IPv6 στα LoWPAN δίκτυα, δημιουργήθηκε το πρωτόκολλο 6LoWPAN. Αυτό εισαγάγει ένα στρώμα προσαρμογής ανάμεσα στο IEEE 802.15.4 και το στρώμα δικτύου με σκοπό να γίνει δυνατή η μεταφορά πακέτων IPv6 από τα LoWPAN δίκτυα(εικόνα 7). Το στρώμα αυτό πρέπει να κάνει τις ακόλουθες διαδικασίες, ώστε να καταστήσει τα δύο πρωτόκολλα συμβατά.

- Συμπίεση επικεφαλίδας : Η επικεφαλίδα του IPv6 παραλείπεται από τα πακέτα, όταν το στρώμα προσαρμογής μπορεί να αντλήσει τις απαραίτητες πληροφορίες από το επίπεδο της ζεύξης και από τα πακέτα του IEEE 802.15.4.
- Κατακερματισμός πακέτων: Τα πακέτα IPv6 τεμαχίζονται σε πολλαπλά πακέτα επιπέδου ζεύξης για να προσαρμοστούν στις απαιτήσεις της ελάχιστης MTU (Maximum Transmission Unit) του IPv6.
- Δρομολόγηση στο επίπεδο της ζεύξης: Για να μπορέσει το επίπεδο της ζεύξης να δρομολογήσει IPv6 πακέτα, το στρώμα προσαρμογής πρέπει να μπορεί να φέρει τις διευθύνσεις επιπέδου ζεύξης κατά την διάρκεια ενός IP βήματος.



Εικόνα 7:Ενδιάμεσο Στρώμα Προσαρμογής (6LoWPAN)

Στην Εικόνα 8 παρουσιάζονται τα τμήματα που αποτελούν τη διεύθυνση των συσκευών που χρησιμοποιούν το IPv6 σε συνδυασμό με το 6LoWPAN. Πιο συγκεκριμένα υπάρχουν δυο τμήματα. Το πρώτο τμήμα έχει μήκος 64 bits. Το δεύτερο αποτελείται από 64 bits και είναι η διεύθυνση του πρωτοκόλλου 802.15.4.



Εικόνα 8: Επικεφαλίδα IPv6

«Διαδίκτυο Αισθητήρων» (Sensor Web)

Ο όρος «Διαδίκτυο Αισθητήρων» (Sensor Web) χρησιμοποιείται από το Open Geospatial Consortium (OGC) για να περιγράψει ένα σύστημα αισθητήρων που αναφέρουν τα δεδομένα τους μέσα από το Διαδίκτυο. Αυτά τα συστήματα μπορεί να είναι και ολόκληρα δίκτυα αισθητήρων που θεωρούνται σαν απλοί κόμβοι και μπορούν να επικοινωνούν και να διαχειρίζονται μέσα από το Διαδίκτυο. Επιπλέον, το «Διαδίκτυο

Αισθητήρων» επικεντρώνεται στην ανταλλαγή των δεδομένων των δικτύων αισθητήρων και στην κατάλληλη επεξεργασία τους, έτσι ώστε να καταστούν αυτά εκμεταλλεύσιμα και να συμβάλουν καλύτερα στην αντίληψη του περιβάλλοντός τους. Το γεγονός αυτό, όπως έχει αναφερθεί και προηγουμένως, καθιστά την επεξεργασία των δεδομένων σαν ένα από τα σημαντικότερα θέματα στα δίκτυα αισθητήρων.

Διαθέσιμες Συσκευές και Λειτουργικά Συστήματα Δικτύων Αισθητήρων

Στον Πίνακα 1 παρουσιάζονται τα χαρακτηριστικά συσκευών δικτύων αισθητήρων που είναι διαθέσιμες σήμερα. Σύμφωνα με αυτά, παρατηρούμε πως οι κόμβοι αισθητήρων εξελίσσονται με προηγμένες δυνατότητες αποθήκευσης και επεξεργασίας δεδομένων.

Αισθητήρας	Μνήμη	Επεξεργαστής	Παροχή Ενέργειας	Πρωτόκολλο Επικοινωνίας
Sunspot	512K Ram-4 M Flash	ARM920T-180MHz -32bit	3,7 V 750mAh	IEEE 802.15.4
Iris Mote	Ram 8 K-512K Flash	AtMega128L	2 AA μπαταρίες	IEEE 802.15.4
Imote2	32MB SRAM 32MB Flash	Marvell PXA271 ARM11-400MHz	3 AAA 3,2-4,5 V	TI CC2420 802.15.4
Micaz	4K RAM 128K Flash	ATMEGA 128	2 AA μπαταρίες	TI CC2420 802.15.4
EcoWizard	Ram 8 K-512K Flash	AtMega128L	6AA μπαταρίες	TI CC2420 802.15.4

Πίνακας 1: Διαθέσιμες Συσκευές Δικτύων Αισθητήρων

Επιπλέον, είναι γνωστό πως όλα τα ενσωματωμένα συστήματα για να λειτουργήσουν είναι απαραίτητο να υποστηρίζονται από ένα λειτουργικό σύστημα. Αυτό είναι υπεύθυνο για το συγχρονισμό όλων των λειτουργιών του κόμβου. Το λειτουργικό σύστημα παρεμβάλλεται μεταξύ του υλικού των κόμβων και των εφαρμογών. Με τον τρόπο αυτό διαχειρίζεται τα επιμέρους τμήματα του κόμβου. Επίσης, το λειτουργικό σύστημα θα πρέπει να σχεδιαστεί με βάση τις υπολογιστικές δυνατότητες των κόμβων. Οι κόμβοι βασίζονται σε μικρουπογιστές και η μνήμη που έχουν στη διάθεσή τους είναι πολύ μικρή.

Τέλος, το λειτουργικό σύστημα θα πρέπει να εξοικονομεί όσο το δυνατό περισσότερη ενέργεια. Στη συνέχεια παρουσιάζονται τα πιο γνωστά από αυτά.

Το TinyOS είναι ένα από τα κυριότερα λειτουργικά σύστηματα για ενσωματωμένα συστήματα και προορίζεται κυρίως για τους κόμβους των δικτύων των αισθητήρων. Το συγκεκριμένο λειτουργικό είναι γραμμένο σε C και στοχεύει στην καλύτερη δυνατή εκμετάλλευση της μνήμης των κόμβων. Αποτελείται από διαφορετικά στοιχεία λογισμικού, τα οποία συνεργάζονται με σκοπό ένα σταθερό και αποδοτικό σύστημα. Μέσα από τις διεπαφές που προσφέρονται, τα διάφορα στοιχεία επικοινωνούν μεταξύ τους καθώς και με το υλικό του κόμβου.

Το επόμενο λειτουργικό που χρησιμοποιείται από αρκετούς κόμβους είναι το Contiki. Το λειτουργικό αυτό σύστημα έχει ως πλεονεκτήματά του τη φορητότητα του και την ικανότητα να διαχειρίζεται πολλές εφαρμογές ταυτόχρονα. Παράλληλα, είναι σχεδιασμένο για ενσωματωμένα συστήματα με περιορισμένη μνήμη και άρα κατάλληλο για τα ασύρματα δίκτυα αισθητήρων. Ένα ακόμα σημαντικό στοιχείο, που έχει το Contiki, είναι ότι έχει υλοποιημένη τη στοίβα των πρωτοκόλλων TCP/IP και υποστηρίζει και το πρωτόκολλο IPv6.

Στην παρούσα εργασία χρησιμοποιήθηκε το SunSPOT σαν κόμβος αισθητήρα. Το λειτουργικό που χρησιμοποιήθηκε είναι το Squawk, που είναι μια Java Micro Edition εικονική μηχανή για ενσωματωμένα συστήματα και μικρές συσκευές. Η εικονική αυτή μηχανή αναλαμβάνει την εκτέλεση των εφαρμογών. Στο αντίστοιχο κεφάλαιο υπάρχει διεξοδική ανάλυση.

3. Διαχείριση Δεδομένων σε Δίκτυα Αισθητήρων

Η διαχείριση των δεδομένων των δικτύων αισθητήρων έχει το σημαντικότερο ρόλο σε αυτά και είναι η κυριότερη αιτία ύπαρξής τους. Με άλλα λόγια, τα δίκτυα των αισθητήρων στοχεύουν στην καλύτερη κατανόηση του περιβάλλοντος, την συλλογή πληροφοριών και την εκμετάλλευση αυτών. Οι χρήστες βασίζονται πάνω σε αυτά τα δίκτυα προκειμένου να αντλήσουν γνώση για το περιβάλλον τους. Ο παραπάνω λόγος είναι ο πιο αποφασιστικός για την κατάλληλη και αποδοτική διαχείριση δεδομένων. Υπάρχουν μερικοί περιορισμοί που καθορίζουν τον τρόπο προσέγγισης της λειτουργίας της διαχείρισης

δεδομένων. Αρχικά, θα πρέπει ο τρόπος διαχείρισης των δεδομένων να ελαχιστοποιεί την κατανάλωση ενέργειας, ώστε να μεγιστοποιείται ο χρόνος ζωής του δικτύου. Ακόμα, θα πρέπει τα δεδομένα να παρέχονται όσο το δυνατόν πιο γρήγορα προς το χρήστη που τα ζήτησε και θα πρέπει τα δεδομένα αυτά να απεικονίζουν την πραγματικότητα όσο πιο πιστά γίνεται. Τέλος, οι δυνατότητες των κόμβων επηρεάζουν σημαντικά την λειτουργία της διαχείρισης δεδομένων, καθώς απαιτείται υπολογιστική και αποθηκευτική ισχύς.

Η διαχείριση των δεδομένων αποτελείται από τρεις ξεχωριστές λειτουργίες. Η πρώτη εξ αυτών είναι η συλλογή των δεδομένων σε κάποιους κόμβους συγκέντρωσης. Στη συνέχεια ακολουθεί η αποθήκευσή τους και τέλος η παρουσίαση τους στους χρήστες, όταν αυτό ζητηθεί από το δίκτυο αισθητήρων. Για τον καθορισμό των παραπάνω λειτουργιών, έχουν προταθεί διάφορες προσεγγίσεις. Κάθε μια από αυτές έχει συγκεκριμένα πλεονεκτήματα και μειονεκτήματα και αντιπαρέρχεται συγκεκριμένους από τους παραπάνω περιορισμούς.

3.1 Επεξεργασία Δεδομένων

Κατά κύριο λόγο έχουν προταθεί δύο τεχνικές για την διαχείριση των δεδομένων των δικτύων αισθητήρων. Η πρώτη βασίζεται στα συστήματα κατανεμημένων βάσεων δεδομένων. Η ερεύνα σε αυτόν τομέα έχει προοδεύσει και είναι σε θέση να παρέχει λύσεις σε σχέση με τα προβλήματα που ανακύπτουν. Από την άλλη πλευρά, οι δυνατότητες των κόμβων των δικτύων αισθητήρων είναι αρκετά πιο περιορισμένες και δεν μπορούν να συγκριθούν με τις κατανεμημένες βάσεις και άρα θα πρέπει να ληφθούν υπόψη οι περιορισμοί που θέτουν τα δίκτυα των αισθητήρων. Για τον λόγο αυτό αναπτύχθηκε η δεύτερη προσέγγιση, η οποία σχεδιάστηκε με σκοπό να αντιμετωπίσει τους περιορισμούς των δικτύων των αισθητήρων και να παρέχει ένα σταθερό σύστημα διαχείρισης των δεδομένων. Αυτή βασίζεται σε ασύγχρονες και τοπικές προσεγγίσεις που εκμεταλλεύονται τις συνδέσεις που υπάρχουν μεταξύ των κόμβων. Στη επόμενη παράγραφο περιγράφονται πιο αναλυτικά οι παραπάνω προσεγγίσεις σε σχέση με κάθε τομέα της συγκέντρωσης δεδομένων, όπως η αποθήκευση, η συλλογή και η παρουσίαση των δεδομένων.

3.2 Τεχνικές επεξεργασίας

Στην προηγούμενη ενότητα περιγράφηκαν σε γενικές γραμμές οι δυο προσεγγίσεις που υπάρχουν για την επεξεργασία των δεδομένων. Στην συνέχεια του κεφαλαίου αναλύονται οι ξεχωριστές λειτουργίες της επεξεργασίας δεδομένων και περιγράφονται οι προσεγγίσεις που ακολουθούνται για αυτές.

Η αποθήκευση των δεδομένων αποτελεί την πρώτη λειτουργία που πρέπει να γίνει, πριν την επεξεργασία τους. Η αποθήκευση των δεδομένων είναι σημαντική διότι μέσα από αυτή απαντώνται τα ερωτήματα των χρηστών, επηρεάζεται η απόδοση του δικτύου και μπορεί να εξοικονομηθεί ενέργεια. Τρεις είναι οι στρατηγικές που έχουν προταθεί. Η πρώτη προβλέπει την αποθήκευση σε μια κεντρική δομή εκτός του δικτύου αισθητήρων. Το μειονέκτημα αυτής της προσέγγισης είναι ότι καταναλώνει αρκετή ενέργεια κατά την αποστολή των δεδομένων από τους κόμβους. Η δεύτερη προβλέπει την αποθήκευση των δεδομένων κάθε κόμβου τοπικά σε αυτόν. Σε αυτή την περίπτωση κάθε κόμβος δρομολογεί τα δεδομένα του μόνο όταν χρειάζεται να απαντήσει σε κάποιο ερώτημα κάποιου χρηστή (Data-Centric Routing - DCR). Τέλος, έχει προταθεί και η αποθήκευση δεδομένων σε συγκεκριμένους κόμβους ανάλογα με τον τύπο τους (Data-Centric Storage DCS).

Η συλλογή των δεδομένων είναι μια άλλη λειτουργία που εντάσσεται στην επεξεργασία των δεδομένων. Η συλλογή προβλέπει την συγκέντρωση των δεδομένων σε έναν κόμβο συγκέντρωσης (Sink-Node). Η ύπαρξη περισσότερων κόμβων που μετρούν την ίδια ιδιότητα προσφέρει μεγαλύτερη ασφάλεια και καλύτερη ποιότητα στις μετρήσεις. Όμως για την αποδοτικότερη συλλογή των δεδομένων δυο θέματα που αφορούν την επιλογή του κόμβου συγκέντρωσης πρέπει να λυθούν. Αρχικά, η επιλογή του κόμβου συγκέντρωσης των δεδομένων θα πρέπει να γίνεται με βάση την υπολογιστική ισχύ των κόμβών. Στη συνέχεια, θα πρέπει η επιλογή να υλοποιηθεί με βάση μια στρατηγική που θα ελαχιστοποιεί την κατανάλωση της ενέργειας στο δίκτυο. Για να είναι δυνατή η καλύτερη επιλογή υπάρχουν αρκετοί αλγόριθμοι για την επιλογή κόμβου συγκέντρωσης δεδομένων που λαμβάνουν υπόψη τους όλους τους περιορισμούς και δίνουν περισσότερο χρόνο ζωής στο δίκτυο.

Ακόμα, κατά την επεξεργασία των δεδομένων πρέπει να γίνουν και κάποιες υπολογιστικές διαδικασίες. Αυτές μπορεί να γίνουν είτε σε μια κεντρική δομή που καταλήγουν τα δεδομένα είτε γίνονται τμηματικά στους κόμβους. Συμφώνα με την δεύτερη

περίπτωση θα πρέπει ο κάθε κόμβος να κάνει τις απαραίτητες διαδικασίες και στη συνέχεια θα πρέπει να προωθεί τα δεδομένα του για επεξεργασία στους κόμβους που ακολουθούν, σύμφωνα με το δέντρο δρομολόγησης των δεδομένων. Με την δεύτερη διαδικασία εισάγονται και θέματα συγχρονισμού των κόμβων, διότι θα πρέπει οι κόμβοι που ακολουθούν να συγχρονίζονται με αυτούς που προηγούνται. Για να επιτευχθεί ο συγχρονισμός έχουν προταθεί κάποιες στοχαστικές λύσεις που μπορούν να μειώσουν την καθυστέρηση του δικτύου. Οι μέθοδοι συγκέντρωσης δεδομένων αξιολογούνται με βάση την ακρίβειά τους, την καθυστέρηση, την πληρότητά τους και το μέγεθος της επικεφαλίδας που εισάγουν σε κάθε πακέτο.

Τέλος, η απάντηση των ερωτημάτων του χρήστη περιλαμβάνεται στη διαχείριση των δεδομένων και συνδέεται με την αποθήκευση και την συλλογή τους. Τα ερωτήματα διαχωρίζονται με βάση το είδος και το μέγεθος των δεδομένων που ζητούν. Για παράδειγμα η συλλογή των δεδομένων μπορεί να θεωρηθεί σαν απάντηση σε ένα ερώτημα που απαιτεί το σύνολο των δεδομένων. Από την άλλη πλευρά το ερώτημα για την θερμοκρασία ενός αισθητήρα απαιτεί λιγότερα δεδομένα, αφού η θερμοκρασία μεταβάλλεται σχετικά αργά και δημιουργεί λιγότερη κίνηση. Παρόλα αυτά ο σημαντικότερος στόχος της παρουσίασης των αποτελεσμάτων είναι η μεγαλύτερη δυνατή ακρίβεια με την χαμηλότερη δυνατή κατανάλωση ενέργειας. Για τον λόγο αυτό έχουν γίνει προσπάθειες να βελτιστοποιηθούν όλες οι λειτουργίες που αποτελούν τη διαδικασία παρουσίασης των αποτελεσμάτων όπως ο τρόπος απάντησης των ερωτημάτων, η μορφή των ερωτημάτων και ο τρόπος δρομολόγησης των απαντήσεων. Τελικά, για την μείωση της κατανάλωσης της ενέργειας των κόμβων έχει προταθεί η βελτιστοποίηση των ερωτημάτων και η χρήση κατάλληλων αλγόριθμων δρομολόγησης για τις απαντήσεις προς τους χρήστες.

Επιπλέον, ένα άλλο χαρακτηριστικό των δικτύων των αισθητήρων είναι ότι τα δεδομένα εισέρχονται συνέχεια από τους κόμβους και χρειάζεται η συνεχής παρακολούθησή τους. Με βάση αυτά τα δεδομένα θα πρέπει να ενεργοποιούν κάποιους συναγερμούς και να ειδοποιούν τους χρήστες για τα πιθανά γεγονότα. Αυτό το χαρακτηριστικό των δικτύων των αισθητήρων κάνει τη διαχείριση και την επεξεργασία των δεδομένων πιο απαιτητική και πιο σημαντική για τη σωστή τους λειτουργία.

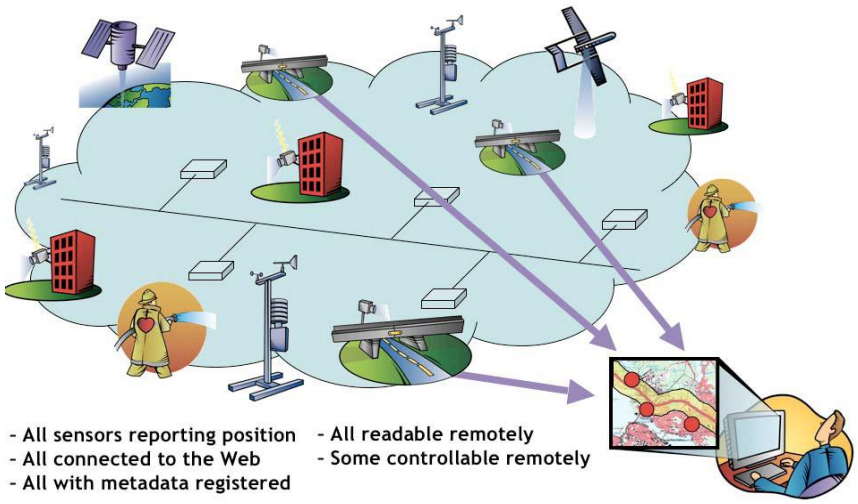
Τέλος, στα δίκτυα των αισθητήρων, που οι κόμβοι είναι σε κίνηση, η διαχείριση των δεδομένων γίνεται πιο απαιτητική και οι στρατηγικές που εφαρμόζονται για την

αποθήκευση, τη συλλογή και την απάντηση ερωτημάτων θα πρέπει να προσαρμοστούν ανάλογα με τη συνεχή αλλαγή της τοπολογίας του δικτύου.

Συμπερασματικά, τα παραπάνω θέματα ανακύπτουν λόγω των περιορισμών που υπάρχουν στα δίκτυα των αισθητήρων. Επιπλέον, με την εισαγωγή του Sensor Web δημιουργήθηκαν νέες απαιτήσεις και έγινε επιτακτική η ανάγκη για αποδοτικότερη διαχείριση και επεξεργασία των δεδομένων και η ανάγκη για διασύνδεση των ετερογενών δικτύων. Αυτή η διασύνδεση θα κάνει το Sensor Web πιο ευέλικτο. Για τον λόγο αυτό το OGC δημιούργησε Sensor Web Enablement (SWE), το οποίο είναι μια σειρά από πρότυπα που περιγράφουν τα δίκτυα των αισθητήρων και καταστούν δυνατή τη διασύνδεση διαφορετικών δικτύων αισθητήρων. Στην επόμενη ενότητα αναλύονται διεξοδικά τα πρότυπα και δίνεται ιδιαίτερη έμφαση στην SensorML.

3.3 Sensor Web Enablement

Μια δραστηριότητα του Open Geospatial Consortium (OGC) είναι το Sensor Web Enablement (SWE). Το SWE έχει ως σκοπό την οριοθέτηση ενός επαναστατικού πλαισίου από πρότυπα για την αποδοτικότερη εκμετάλλευση των δικτύων των αισθητήρων όλων των ειδών (εικόνα 9). Το SWE προσφέρει τη δυνατότητα στο διαδίκτυο να έχει μια πιο πραγματικού χρόνου προσέγγιση και ακόμα προσφέρει την εφαρμογή των δικτύων των αισθητήρων σε διαφόρους τομείς.



Εικόνα 9: Δίκτυα Ετερογενών Κόμβων

Τα μοντέλα, οι κωδικοποιήσεις και οι υπηρεσίες της αρχιτεκτονικής του SWE συμβάλλουν στην ανάπτυξη και την διαλειτουργικότητα ετερογενών δικτύων αισθητήρων. Το SWE στοχεύει στην ανάπτυξη προτύπων για την ανακάλυψη, στην ανταλλαγή και την επεξεργασία των παρατηρήσεων. Επιπλέον, στοχεύει στο σαφή καθορισμό των εργασιών των δικτύων.

Βασικός στόχος του SWE είναι ο χειρισμός όλων των αισθητήρων μέσω του διαδικτύου. Σε επόμενο βήμα θα πρέπει να δημιουργηθούν τα πρότυπα που θα επιτρέπουν την εγκατάσταση δικτύων αισθητήρων συνδεδεμένων με το διαδίκτυο. Για αυτό το λόγο οι προδιαγραφές του SWE πρέπει να είναι καλά ορισμένες και οι ομάδες που δουλεύουν για την υλοποίηση των προτύπων πρέπει να τα εναρμονιστούν με τις προδιαγραφές του SWE.

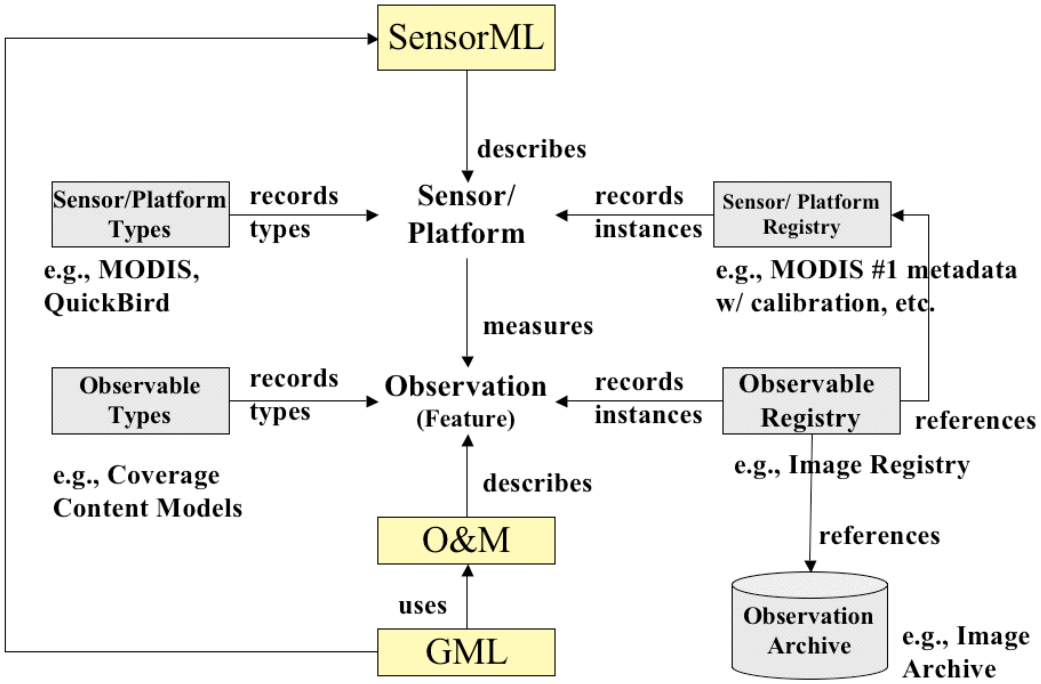
Όταν τα δίκτυα αισθητήρων δικτυώνονται με βάση τις τεχνολογίες και τα πρωτόκολλα του διαδικτύου, τότε είναι δυνατόν να χρησιμοποιηθούν XML σχήματα για τη δημοσιοποίηση και την περιγραφή των δυνατοτήτων των αισθητήρων, τη θέσης τους και των διεπαφών τους. Με την βοήθεια των τεχνολογιών του διαδικτύου είναι δυνατό να αναλυθούν τα XML δεδομένα των δικτύων των αισθητήρων ανεξάρτητα από τη μορφή τους και την τεχνολογία τους.

Οι πληροφορίες για τους αισθητήρες που βρίσκονται στα XML σχήματα περιγράφουν τα δομικά χαρακτηριστικά των συσκευών και κάνουν ευκολότερη την ανάπτυξη νέων εφαρμογών. Αυτή η δομή της XML έχει ως εξέλιξη την SensorML. Τέλος, υπάρχει και μια πιο αποδοτική αντικειμενοστραφής προσέγγιση του αισθητήρα.

Τα επτά πρότυπα του OGC για το SWE είναι τα επόμενα (εικόνα 10).

- Observations & Measurements (O&M): Αυτό το πρότυπο στοχεύει στο να παρέχει ένα πλαίσιο για την αναπαράσταση και την ανταλλαγή των δεδομένων.
- Sensor Model Language (SensorML): Χρησιμοποίειται για την μοντελοποίηση και κωδικοποίηση δεδομένων και συστημάτων.
- TransducerML: Είναι μια μέθοδος και ένα πρότυπο μηνυμάτων για την περιγραφή πληροφοριών για τους αισθητήρες και για τα δίκτυα των αισθητήρων με την ευρύτερη έννοια.
- Sensor Observation Service (SOS): Στόχος του είναι να δώσει μια σταθερής μορφής διεπαφή προς τα δεδομένα των αισθητήρων.

- Sensor Planning Service (SPS): Έχει σχεδιαστεί και αναπτυχθεί έτσι ώστε να είναι δυνατή μια διαλειτουργική υπηρεσία κατά την οποία ένας client δύναται να καταθέσει ένα σύνολό από αιτήσεις σε ένα σύστημα αισθητήρων.
- Sensor Alert Service (SAS): Καθορίζει τις διεπαφές για τη συγκέντρωση δεδομένων και περιγράφει τις δυνατότητες μιας υπηρεσίας για τον καθορισμό των συναγερμών.
- Web Notification Service (WNS): Περιγράφει μια διεπαφή για υπηρεσία, όπου ένας client έχει τη δυνατότητα ασύγχρονης επικοινωνία με άλλες υπηρεσίες των συστημάτων των αισθητήρων.



Εικόνα 10: Τα Πρότυπα του OGC για το SWE

3.4 Sensor Model Language (SensorML)

Η SensorML παρέχει ένα πληροφοριακό μοντέλο-μια κωδικοποίηση που κάνει δυνατή την ανακάλυψη και την χρησιμοποίηση των αισθητήρων που βρίσκονται στο διαδίκτυο και την εκμετάλλευση των παρατηρήσεών τους.

Η μέτρηση ενός φαινομένου, που γίνεται με βάση κάποιες παρατηρήσεις, αποτελείται από ορισμένες διαδικασίες. Αρχικά υπάρχει η διαδικασία της ανίχνευσης και της δειγματοληψίας και στη συνέχεια ακολουθεί η διαδικασία της επεξεργασίας των δεδομένων. Ο διαχωρισμός μεταξύ της μέτρησης και της επεξεργασίας που ακολουθεί έχει γίνει δύσκολη εξαιτίας των σύνθετων και έξυπνων αισθητήρων και της χρησιμοποίησης εφαρμογών που απαιτούν την επεξεργασία των παρατηρήσεων από τους κόμβους. Τα GPS είναι ένα πρώτο παράδειγμα μιας συσκευής που περιέχει τους βασικούς ανιχνευτές μαζί με τις σύνθετες διαδικασίες που τελικά δίνουν τα αναμενόμενα αποτελέσματα.

Η SensorML ορίζει μοντέλα και XML σχήματα για την περιγραφή όλων των διαδικασιών συμπεριλαμβανομένων των μετρήσεων από έναν αισθητήρα και της μετέπειτα επεξεργασίας.

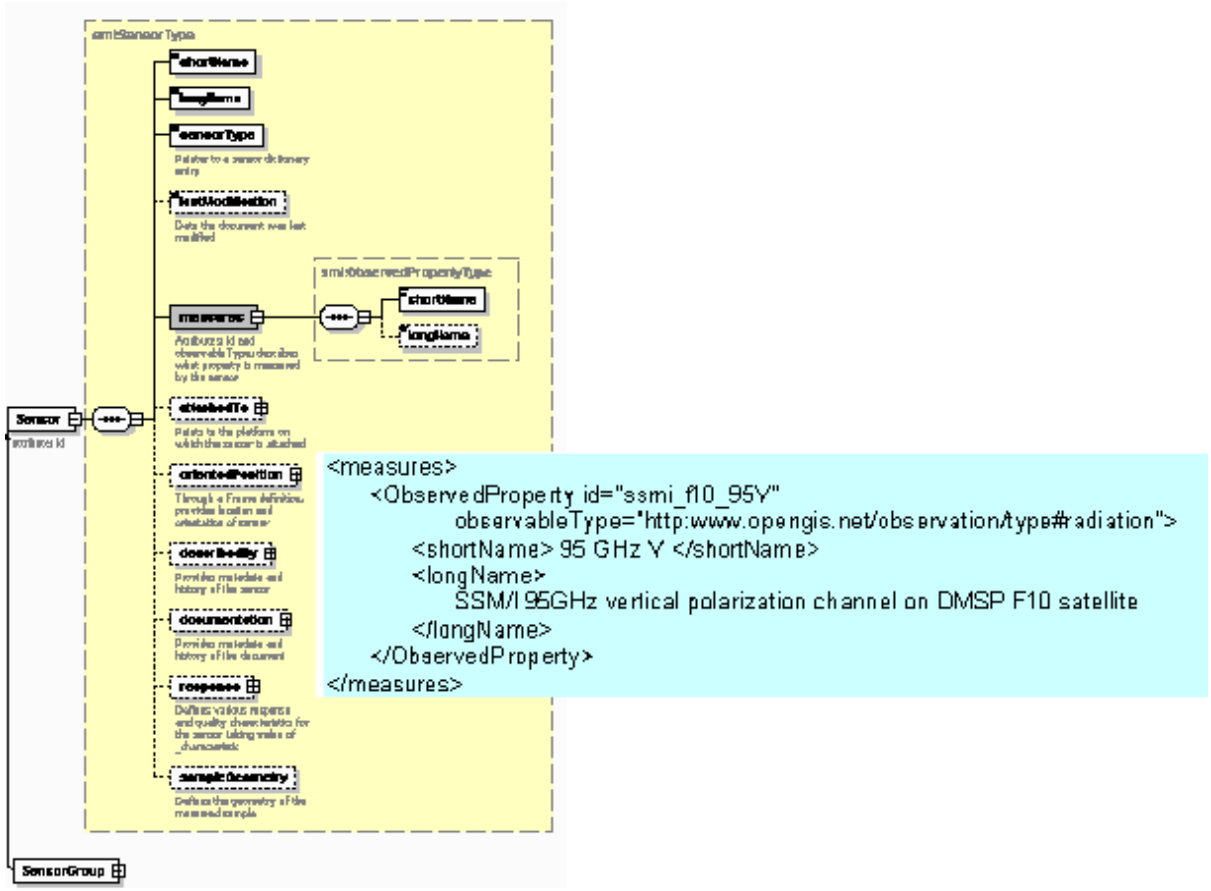
Με την SensorML όλα τα στοιχεία, όπως οι ανιχνευτές, οι ενεργοποιητές, τα φίλτρα και οι διαχειριστές, ορίζονται ως μοντέλα διαδικασιών. Ένα μοντέλο διαδικασίας ορίζει τις μεταβλητές εισόδου, τις μεταβλητές εξόδου, τις παραμέτρους και τη μέθοδο για τη διαδικασία, ενώ ακόμα καθορίζει μια σειρά από meta-data απαραίτητα, τόσο για μετέπειτα επεξεργασία, όσο και για τη χρησιμοποίησή τους από ανθρώπους. Οι μεταβλητές εισόδου, οι μεταβλητές εξόδου και οι παράμετροι είναι ορισμένες από τους τύπους δεδομένων του SWE. Η επεξεργασία των meta-data περιλαμβάνει αναγνωριστικά, ταξινομητές, διάφορους περιορισμούς (χρόνος, νομιμότητα, και ασφάλεια), ικανότητες, χαρακτηριστικά, επαφές, και αναφορές, εκτός από τις εισροές, τις εκροές, τις παραμέτρους, και τη θέση του συστήματος.

Η SensorML παρέχει ένα λειτουργικό μοντέλο του συστήματος και όχι μια λεπτομερή περιγραφή του υλικού του συστήματος. Η SensorML θεωρεί το δίκτυο των αισθητήρων και όλα τα εξαρτήματα του σαν διαδικασίες. Για αυτό κάθε εξάρτημα μπορεί να πάρει μέρος σε μια ή περισσότερες αλυσίδες διαδικασιών που μπορούν να περιγράψουν την καταγωγή των παρατηρήσεων ή να παρέχουν μια

διαδικασία για τον εντοπισμό και την επεξεργασία των μετρήσεων σε υψηλότερο επίπεδο.

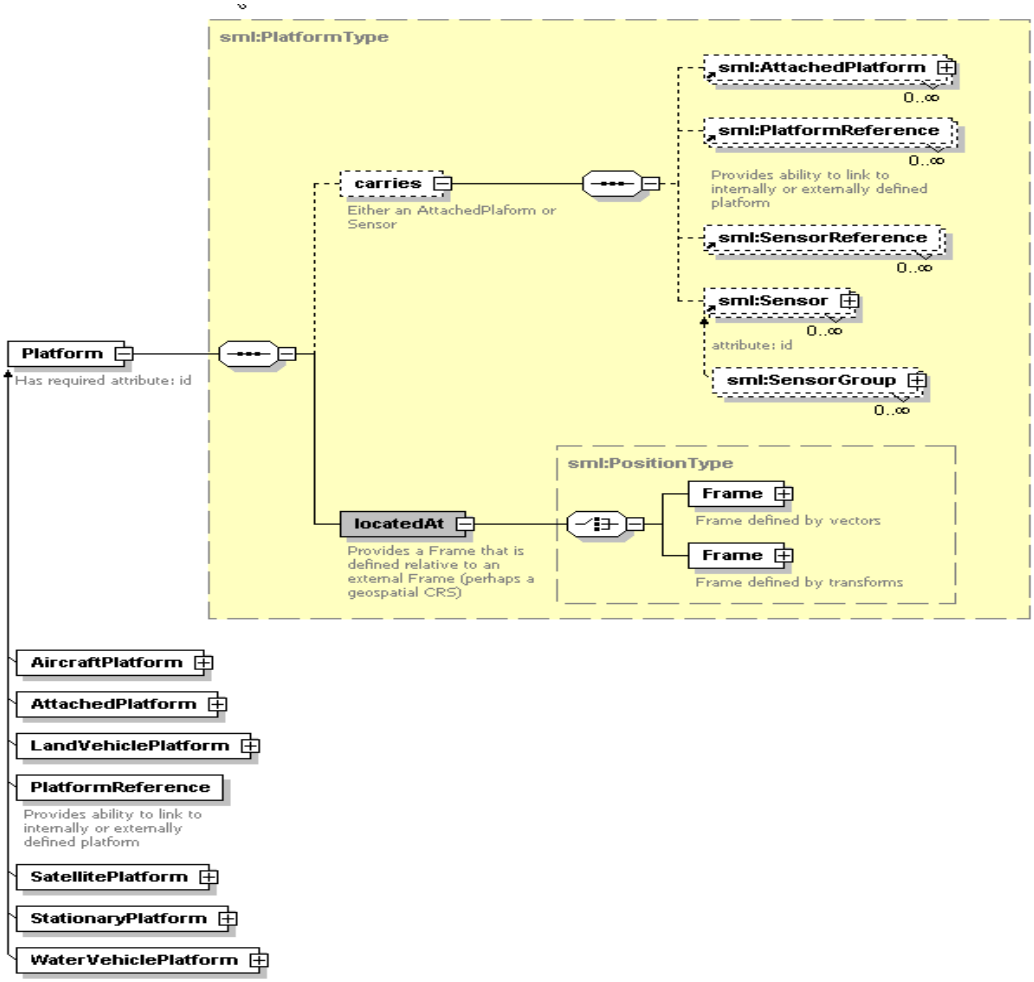
Στην SensorML όλα τα μέρη ,συμπεριλαμβανομένων των αισθητήρων και των δικτύων, έχουν μεταβλητές εισόδου, μεταβλητές εξόδου ,παραμέτρους και μεθόδους που μπορούν να χρησιμοποιηθούν από εφαρμογές, ώστε να εκμεταλλευτούν τις παρατηρήσεις από οποιοδήποτε σύστημα αισθητήρων. Επιπλέον, η SensorML παρέχει meta-data τα οποία συμβάλλουν στην ανακάλυψη, στον προσδιορισμό των περιορισμών του συστήματος (π.χ. ασφάλεια ή νομικούς περιορισμούς χρήσης), για την παροχή επαφών και αναφορών, και για την περιγραφή των ιδιοτήτων των πινάκων εργασιών, των διεπαφών.

Στην εικόνα 11 παρουσιάζεται το σχήμα για την περιγραφή ενός αισθητήρα.



Εικόνα 11: Περιγραφή ενός Κόμβου

Ακολουθεί το σχήμα 12 της πλατφόρμας πάνω στην οποία στήνεται ένα δίκτυο αισθητήρων.



Εικόνα 12: Περιγραφή μιας Πλατφόρμας

Τέλος δίνεται το XML αρχείο για την ιδιότητα describedBy(εικόνα 14)

```
<describedBy>
    <SensorSpecification>
        <manufacturedBy> YSI </manufacturedBy>
        <model> 58 </model>
        <identifyingNumber type="serialNumber"> s454165 </identifyingNumber>
        <history>
            <SensorCreation>
                <byWhom>
                    <Person>
                        <fullName> Some Guy </fullName>
                    </Person>
                </byWhom>
                <when> 2001-12-04 </when>
                <supportingDocuments>
                    <Reference id="YSI-12">
                        <authors>
                            <Person>
                                <fullName> Some Author </fullName>
                            </Person>
                        </authors>
                        <publishingDate> 2001-01-01 </publishingDate>
                        <documentTitle> Blueprints for This Sensor </documentTitle>
                        <documentNumber> YSI-12345-678 </documentNumber>
                    </Reference>
                </supportingDocuments>
            </SensorCreation>
            <SensorDeployment>
                <byWhom>
                    <Person>
                        <fullName>SomeOther Guy</fullName>
                    </Person>
                </byWhom>
                <where> Someplace, Somewhere </where>
                <when> 2001-12-04 </when>
                <description> Attached to northern most footing on Some Bridge at a depth
                    of 5 meters </description>
            </SensorDeployment>
        </history>
        <properties>
            <genericProperty name="sensorTechnology" dataType="xs:string"> rapid
                pulse </genericProperty>
            <genericProperty name="measurementMethod" dataType="xs:string"> EPA
                accepted </genericProperty>
            <genericProperty name="membraneThickness" dataType="xs:double"
                uom="#mil"> 1.0 </genericProperty>
            <genericProperty name="membraneType" dataType="xs:string"> Teflon
                </genericProperty>
            <genericProperty name="probeSolutionType" dataType="xs:string"> Na2SO4
                </genericProperty>
        </properties>
    </SensorSpecification>
</describedBy>
```

Εικόνα 14: Το XML σχήμα για την ιδιότητα describedBy

4. Συγκέντρωση και Δρομολόγηση Δεδομένων στα Δικτυα Αισθητήρων

4.1 Συγκέντρωση Δεδομένων

Η αποτελεσματική συγκέντρωση των δεδομένων είναι σημαντική για τη μείωση του κόστους της επικοινωνίας και άρα την επέκταση του χρόνου ζωής του δικτύου. Με βάση την τοπολογία του δικτύου, τη θέση των πηγών και τη λειτουργία συγκέντρωσης, μια βέλτιστη δομή συνάθροισης είναι εφικτό να κατασκευαστεί. Η βέλτιστη συνάθροιση καθορίζεται από τη συνολική κατανάλωση ενέργειας, τη χρησιμοποίηση του εύρους ζώνης και την καθυστέρηση για τη μεταφορά των συλλεχθεισών πληροφοριών από τους απλούς κόμβους στους κόμβους

συγκέντρωσης. Η συλλογή δεδομένων μπορεί να πραγματοποιηθεί μέσα από δομημένες ή ελεύθερες από δομή προσεγγίσεις.

Οι δομημένες προσεγγίσεις ταιριάζουν σε εφαρμογές συγκέντρωσης δεδομένων στις οποίες εφαρμόζεται συγκεκριμένο σχέδιο δρομολόγησης των δεδομένων στους κόμβους συγκέντρωσης και οι κόμβοι, που απαρτίζουν το δίκτυο, έχούν προκαθορισμένους ρόλους. Λόγω του αμετάβλητου σχεδίου κυκλοφορίας, οι δομημένες τεχνικές συνάθροισης έχουν μικρή επικεφαλίδα διατήρησης της τοπολογίας και είναι επομένως κατάλληλες για τέτοιες εφαρμογές. Αλλά, σε περίπτωση δυναμικών τοπολογιών είναι απαραίτητο η επικεφαλίδα να περιέχει επιπλέον πληροφορίες , οι οποίες είναι απαραίτητες για την διατήρηση του δικτύου, με αποτέλεσμα την αύξηση του μεγέθους της επικεφαλίδας και άρα εξαλείφονται τα πλεονεκτήματα που προσφέρει η σύντομη επικεφαλίδα. Επιπλέον, οι δομημένες προσεγγίσεις είναι ευαίσθητες στην καθυστέρηση που επιβάλλεται από τους ενδιάμεσους κόμβους, τη συχνότητα της μετάδοσης δεδομένων και το μέγεθος του δικτύου αισθητήρων. Μια κεντρική οντότητα είναι αρμόδια για την ανακάλυψη των νέων κόμβων και την προδιαγραφή της πολιτικής αποκτήσεων δεδομένων. Η απόκτηση στοιχείων μπορεί να βασίζεται σε γεγονότα, όπου τα στοιχεία στέλνονται από την πηγή και μια μέθοδος καλείται για να τα συλλέξει ή σε polling-based μέθοδο, όπου ο κεντρικός κόμβος ρωτά περιοδικά για δεδομένα τους διοικούμενους αισθητήρες.

Η μεταφορά των δεδομένων στους κόμβους συγκέντρωσης στις δομημένες προσεγγίσεις μπορεί να γίνει :

- Address-centric, όπου κάθε κόμβος – πηγή δεδομένων στέλνει τα δεδομένα του κατευθείαν στον κόμβο συγκέντρωσης (end to end routing). Αυτή η μέθοδος έχει την χαμηλότερη δυνατή καθυστέρηση αλλά καταναλώνει πολύ ενέργεια.
- Data-centric: Η δρομολόγηση γίνεται με βάση δεδομένα που ζητούνται. Οι κόμβοι ενσωματώνουν τα δεδομένα τους σε ένα μεγαλύτερο πακέτο που περιέχει δεδομένα και από άλλους κόμβους και έτσι επιτυγχάνεται μικρότερος αριθμός μεταδόσεων και άρα χαμηλότερη κατανάλωση ενέργειας αλλά και μεγαλύτερη καθυστέρηση. Το σχήμα που πετυχαίνει την καλύτερη επίδοση από πλευράς λιγότερων εκπομπών είναι το άπληστα αυξανόμενο δέντρο(Greedy Incremental

Tree , GIT). Σε αυτό το σχήμα τα δεδομένα από τους κόμβους που είναι πιο κοντά στην πηγή αρχικά προσκολλούνται στο πακέτο και στην συνεχεία σε κάθε βήμα το πακέτο μεγαλώνει.

Στις ελεύθερες, από δομή, προσεγγίσεις δεν υπάρχει καμία προκαθορισμένη δομή και οι αποφάσεις δρομολόγησης για την αποδοτική συνάθροιση των πακέτων πρέπει να ληφθούν άμεσα. Δεδομένου ότι οι κόμβοι δεν ξέρουν ρητά τους προς τα πάνω κόμβους τους, δεν μπορούν να περιμένουν τα στοιχεία από οποιοδήποτε κόμβο πριν διαβιβάσουν τα δικά τους στοιχεία προς τον προορισμό. Αυτές οι προσεγγίσεις μπορούν να εφαρμοστούν στα δυναμικά περιβάλλοντα και στα ειδικά δίκτυα αισθητήρων, όπου οι κόμβοι διέρχονται συνεχώς στο δίκτυο και έτσι μια προκαθορισμένη στατική δρομολόγηση και ένα σχέδιο συνάθροισης θα ήταν ενδεχομένως ακατάλληλο.

Για τη συλλογή δεδομένων από σχήματα χωρίς προκαθορισμένη δομή έχει προταθεί ένα σχήμα που δρα στο MAC επίπεδο με το όνομα Data-Aware Anycast (DAA). Στο DAA ένα ανεξάρτητο σύνολο από κόμβους-πηγές δεδομένων δημιουργείται αυτόματα κατά τη διάρκεια αποστολής των δεδομένων προς τους κόμβους συγκέντρωσης. Οι κόμβοι δρουν σαν σημεία συγκέντρωσης. Με την παραπάνω τεχνική είναι δυνατό να μειωθούν οι μεταδόσεις για τη συγκρότηση και διαχείριση του δικτύου. Αν το DAA εμπλουτιστεί και με την ιδέα της τυχαίας αναμονής (Randomized Waiting ,RW), τότε γίνεται πιο αποδοτικό το σύστημα συγκεντρώσης των δεδομένων. Αυτή η ιδέα καθορίζει ένα χρονικό περιθώριο πριν ένας κόμβος στείλει τα δεδομένα του προς τον κόμβο συγκέντρωσης. Σε αυτό το περιθώριο μπορεί να λάβει δεδομένα από άλλους κόμβους και έτσι να τα ενσωματώσει στα δικά του και να αποφευχθούν περιττές μεταδόσεις δεδομένων.

Τέλος η ασφάλεια είναι ένας σημαντικός τομέας .Όλες οι απαιτήσεις και τα εμπόδια έχουν οριστεί, ενώ έχουν καταγραφεί όλες οι επιθέσεις που έχουν γίνει και οι μέθοδοι για την αντιμετώπιση τους. Για μια βέλτιστη δρομολόγηση και συλλογή δεδομένων δεδομένου του τύπου των αισθητήρων και του σεναρίου ανάπτυξης του δικτύου, θα πρέπει να γίνει ένας συμβιβασμός ανάμεσα στην ασφάλεια και την αποτελεσματικότητα.

4.2 Πρωτόκολλα Δρομολόγησης

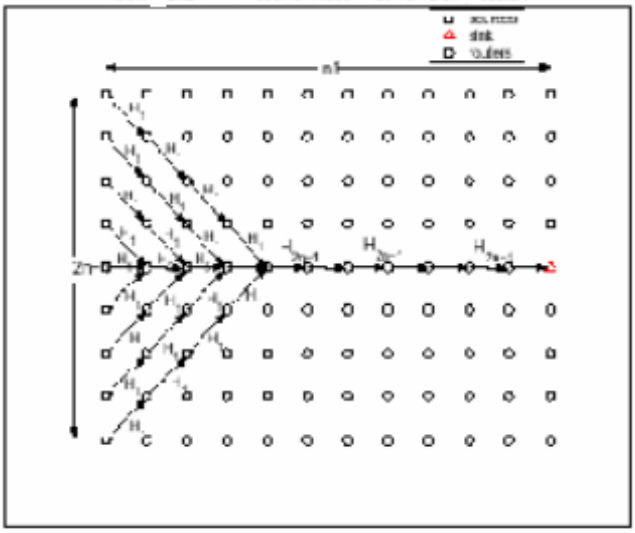
Στην παράγραφο αυτή αναλύονται τόσο οι τεχνικές και οι ιδέες δρομολόγησης όσο και τα πρωτόκολλα που έχουν σχεδιαστεί και εφαρμόζονται από τα σύγχρονα δίκτυα αισθητήρων. Στην αρχή περιγράφονται οι ιδέες της δρομολόγησης. Στα ασύρματα δίκτυα αισθητήρων (WSN) τρεις τεχνικές δρομολόγησης παρουσιάζουν ιδιαίτερο ενδιαφέρον.

- Η από κοινού δρομολόγηση και συμπίεση μέσα στο δίκτυο. Αυτό το σχήμα ελαχιστοποιεί το συνολικό αριθμό των bits που μεταδίδονται μέσα στο δίκτυο.
- Η χαμηλής καθυστέρησης δρομολόγηση με προγραμματισμένες διακοπές. Αυτό το σχήμα έχει ως στόχο να ελαχιστοποιήσει την, από άκρη σε άκρη, καθυστέρηση του δικτύου με δεδομένο έναν κύκλο λειτουργίας (duty cycle).
- Η σταθερή γεωγραφική δρομολόγηση πάνω από πραγματικές ασύρματες συνδέσεις. Το σχήμα στοχεύει στη μεγιστοποίηση του λόγου παράδοσης των δεδομένων (delivery ratio) και στην αποδοτική ενεργειακή διαχείριση.

Δρομολόγηση με Συμπίεση

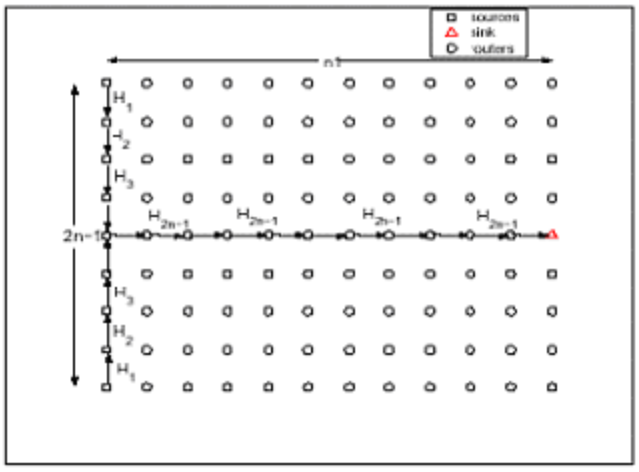
Για τη δρομολόγηση με συμπίεση υπάρχουν οι τρεις παρακάτω τεχνικές, ώστε να επιτευχθεί ταυτόχρονα η απαιτούμενη συμπίεση.

- Δρομολόγηση από το συντομότερο μονοπάτι (Εικόνα 15): Η εν λόγω δρομολόγηση γίνεται από κάθε κόμβο προς τον κόμβο συγκέντρωσης από το συντομότερο μονοπάτι και όταν τα μονοπάτια συμπίπτουν, τότε ενώνονται και υπάρχει συμπίεση των δεδομένων και μείωση των μεταδόσεων.



Εικόνα 15: Δρομολόγηση από το Συντομότερο Μονοπάτι

- Δρομολόγηση οδηγούμενη από την συμπίεση (Εικόνα 16): Αυτή η δρομολόγηση γίνεται με σκοπό τη μεγιστοποίηση της συμπίεσης των δεδομένων που ενσωματώνονται σε ένα πακέτο και με αυτόν τον τρόπο δημιουργούνται μεγαλύτερα μονοπάτια.

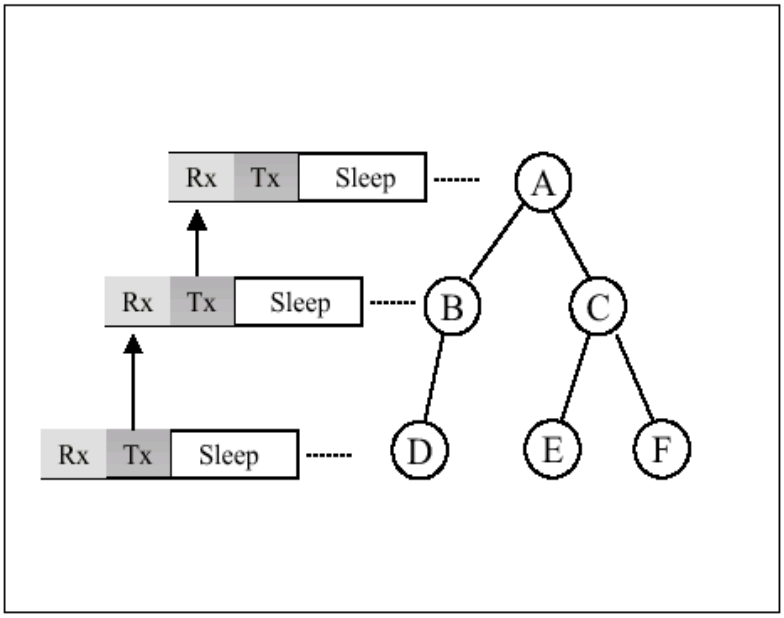


Εικόνα 16:Δρομολόγηση με Βάση την Συμπίεση

- Κατανεμημένη κωδικοποίηση δεδομένων στους κόμβους: Ο κάθε κόμβος συμπιέζει τα δεδομένα του με κάποια μορφή κωδικοποίησης και τα δρομολογεί μέσα από τη συντομότερη διαδρομή προς τον κόμβο συγκέντρωσης. Αν εξαιρεθεί η διαδικασία κατά την οποία ο κόμβος συγκέντρωσης μαθαίνει την μορφή της κωδικοποίησης, αυτή η μέθοδος έχει αποδειχθεί η αποδοτικότερη.

Δρομολόγηση με Χαμηλή Καθυστέρηση

Για να περιοριστεί η κατανάλωση ενέργειας από τους κόμβους προτάθηκε η ύπαρξη περιόδων που δεν θα λειτουργούν οι κόμβοι (Duty Cycle) και δεν θα χρειάζεται να είναι ανοικτή η κεραία του κόμβου που καταναλώνει ενέργεια σε εφαρμογές με χαμηλό ρυθμό αποστολής δεδομένων (Εικόνα 17). Αυτή όμως η μέθοδος έχει ως αποτέλεσμα την αύξηση της καθυστέρησης. Για να λυθεί αυτό το πρόβλημα προτάθηκε η χρήση κλιμακωτών διαστημάτων λειτουργίας για την ελαχιστοποίηση της καθυστέρησης κατά τη συγκέντρωση των δεδομένων.



Εικόνα 17: Χρονικές Σχισμές Δρομολόγησης

Η διαδικασία αυτή προβλέπει την ανάθεση σε κάθε κόμβο μιας χρονοσχισμής από τις k, ώστε να λαμβάνει δεδομένα σε αυτή. Στη συνέχεια η χρονοσχισμή ανακοινώνεται στους γειτονικούς κόμβους που μπορεί να στείλουν δεδομένα στον κόμβο που της αντιστοιχεί.. Οι κόμβοι δεν λειτουργούν τις άλλες χρονοσχισμές, εκτός αν έχουν δεδομένα προς αποστολή. Η χρονική διάρκεια της μη λειτουργίας είναι και ο κύριος παράγοντας καθυστέρησης. Η καθυστέρηση ανάμεσα σε δύο κόμβους διαφέρει και εξαρτάται από τις χρονοσχισμές παραλαβής που έχει ο καθένας. Τα δεδομένα δρομολογούνται από το μονοπάτι με την χαμηλότερη καθυστέρηση. Τελικός στόχος είναι η καλύτερη δυνατή κατανομή των χρονοσχισμών, έτσι ώστε να επιτευχθεί η χαμηλότερη καθυστέρηση από άκρη σε άκρη του δικτύου. Αυτό είναι το πρόβλημα που πρέπει να λυθεί και είναι NP δυσκολίας. Υπάρχουν τοπολογίες που για

τις οποίες αποδοτικές λύσεις έχουν δοθεί και αυτές είναι οι τοπολογίες δέντρου, δαχτυλιδιού και πλέγματος .

Αποδοτικότερη λύση μπορεί να δοθεί αν εφαρμοστεί στους κόμβους η πολλαπλή ανάθεση χρονοσχισμών. Η κλιμάκωση των σχισμών θα πρέπει να σχεδιαστεί έξυπνα, ώστε να γίνει δυνατή η δρομολόγηση με χαμηλή καθυστέρηση. Στο μέλλον θα πρέπει να προσαρμόζεται το πρόγραμμα των σχισμών στις ροές που υπάρχουν στο δίκτυο και έτσι να είναι αποδοτικότερος ο τρόπος δρομολόγησης.

Δρομολόγηση πάνω από Πραγματικές Ασύρματες Συνδέσεις.

Σε αυτό το σχήμα λαμβάνονται υπόψη η ποιότητα των ασύρματων συνδέσεων και γίνεται η δρομολόγηση των πακέτων με βάση τα δύο παρακάτω σχήματα.

- Δρομολόγηση προς το γειτονικό κόμβο, ο οποίος οδηγεί πιο γρήγορα στον κόμβο συγκέντρωσης. Αυτή η μέθοδος έχει ως αποτέλεσμα λιγότερα βήματα (hops), αλλά λόγω πιθανής χαμηλής ποιότητας της σύνδεσης, να απαιτούνται περισσότερες μεταδόσεις, ανά βήμα (hop).
- Δρομολόγηση προς τον κοντινότερο κόμβο που βρίσκεται στην επιθυμητή κατεύθυνση. Αυτή η τεχνική προκαλεί χαμηλή πιθανότητα να χαθεί το πακέτο και επομένως μεγαλύτερο ρυθμό λήψης δεδομένων, αλλά απαιτούνται περισσότερα βήματα μέχρι τον τελικό προορισμό.

Για την αποτελεσματική δρομολόγηση πρέπει να λαμβάνεται υπόψη η ποιότητα των ασυρμάτων συνδέσεων, ώστε να είναι πιο σταθερό το σύστημα. Στα επόμενα βήματα προβλέπεται η ύπαρξη ενός μηχανισμού που θα κάνει τη δρομολόγηση να προσαρμόζεται στις συνθήκες που επικρατούν και έτσι να είναι αποδοτικότερο το σύστημα.

Ένας μείζων στόχος της έρευνας, για το μέλλον, είναι να καθοριστεί, εάν αυτές και άλλες τέτοιες βελτιστοποιήσεις μπορούν να ενοποιηθούν σε μία ενιαία αρχιτεκτονική δρομολόγησης, που θα ήταν κατάλληλη για ένα αρκετά μεγάλο σύνολο εφαρμογών.

Κατηγορίες Πρωτοκόλλων Δρομολόγησης

Τα πρωτόκολλα δρομολόγησης χωρίζονται σε τρεις κατηγορίες με βάση τον τρόπο, που η πηγή βρίσκει το δρόμο προς τον προορισμό. Υπάρχει η δυνατότητα όλα τα μονοπάτια να είναι υπολογισμένα από πριν (proactive), τα μονοπάτια να υπολογίζονται όταν χρησιμοποιούνται (reactive) και ένας συνδυασμός των δύο παραπάνω (hybrid).

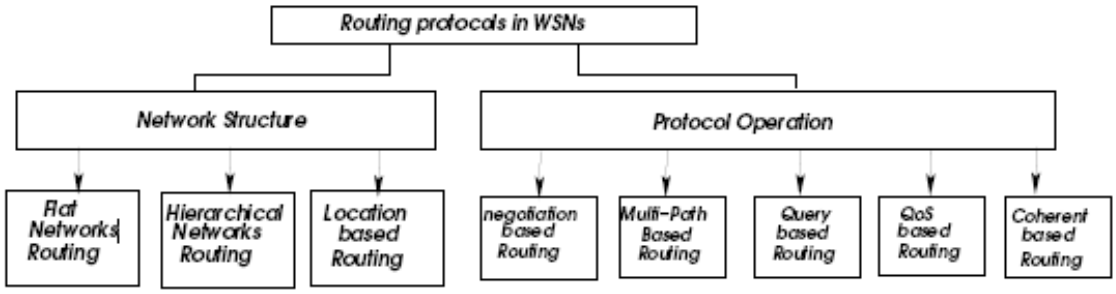
Τα πρωτόκολλα δρομολόγησης χωρίζονται σε τρεις κατηγορίες με βάση τη δομή του δικτύου .

- Τα επίπεδα σχήματα δρομολόγησης (flat)
- Τα ιεραρχικά σχήματα δρομολόγησης (hierarchical)
- Δρομολόγηση με βάση τη θέση (location-based routing)

Επιπλέον, χωρίζονται με βάση τον τρόπο λειτουργίας του πρωτοκόλλου στις επόμενες κατηγορίες :

- Σχήματα βασισμένα στις πολλαπλές διαδρομές
- Σχήματα βασισμένα στα ερωτήματα για δεδομένα
- Σχήματα βασισμένα στη διαπραγμάτευση
- Σχήματα βασισμένα στην ποιότητα εξυπηρέτησης του χρήστη (QoS)
- Σχήματα βασισμένα στη συνεκτικότητα του δικτύου

Στην εικόνα 18 φαίνεται η κατηγοριοποίηση των πρωτοκόλλων.

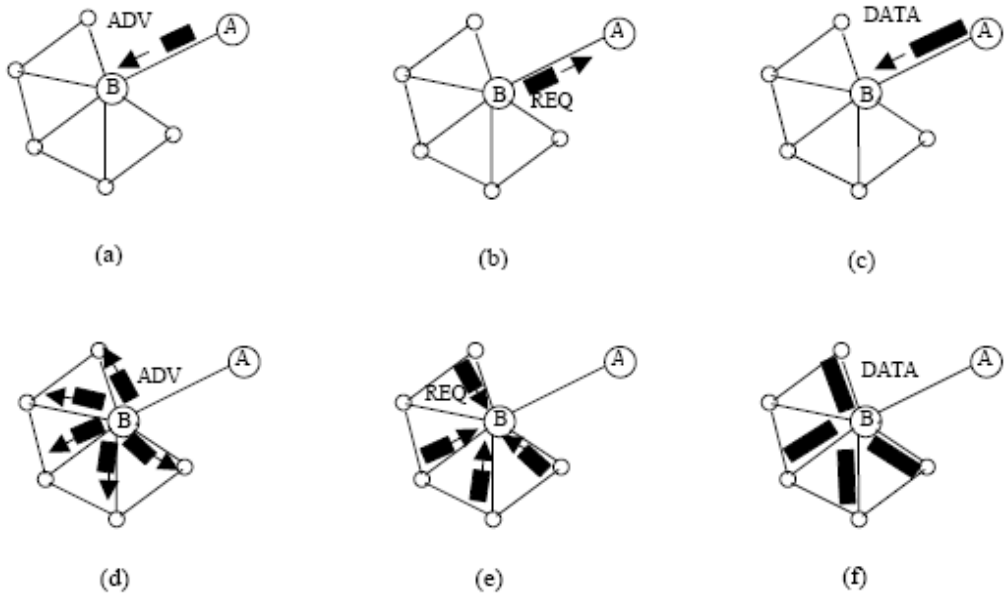


Εικόνα 18; Κατηγορίες Πρωτοκόλλων

Flat Routing

Στα επίπεδα σχήματα δρομολόγησης όλοι οι κόμβοι έχουν τον ίδιο ρόλο και συνεργάζονται, ώστε να πραγματοποιηθεί η δρομολόγηση. Η δρομολόγηση είναι της μορφής Data-centric, δηλαδή, όταν ένας σταθμός βάσης ζητά δεδομένα από κάποια περιοχή με βάση αυτά (τα δεδομένα) δρομολογούνται πίσω σε αυτόν οι απαντήσεις. Στην συνέχεια παρουσιάζονται τα πιο γνωστά πρωτόκολλα της κατηγορίας αυτής.

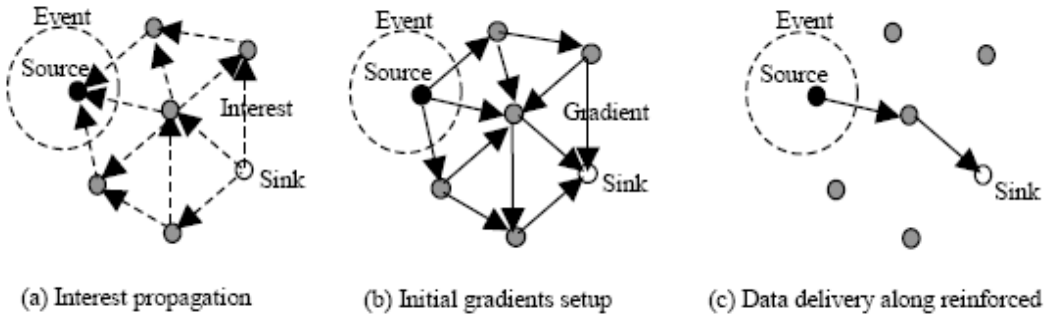
- Το Sensor Protocols of Information via Negotiation (SPIN)(Εικόνα 19) είναι ένα από τα πρώτα σχήματα δρομολόγησης που πραγματοποιούν την ιδέα της Data-Centric δρομολόγησης. Στο σχήμα αυτό όλοι οι κόμβοι είναι εν δυνάμει σταθμοί-βάσεις. Η δρομολόγηση βασίζεται στα meta-data που ανταλλάσσουν οι κόμβοι πριν από τη μεταφορά των δεδομένων. Ο κάθε κόμβος ανακοινώνει στους γειτονικούς του κόμβους τα δεδομένα που έχει λάβει με ένα πακέτο ADV και αυτοί αν ενδιαφέρονται απαντούν με ένα μήνυμα (REQ). Στη συνέχεια λαμβάνουν τα δεδομένα. Τα πλεονεκτήματά του είναι ότι, ο κάθε κόμβος χρειάζεται να γνωρίζει όσους κόμβους είναι γειτονικοί και επιπλέον δεν σπαταλά ενέργεια.



Εικόνα 19:Τρόπος λειτουργίας του SPIN

- Το Directed Diffusion είναι ένα Data-Centric πρωτόκολλο δρομολόγησης(Εικόνα 20). Για να αποφευχθούν οι ενεργειακά ανεπιθύμητες λειτουργίες του στρώματος

δικτύου, το πρωτόκολλο αυτό χρησιμοποιεί ζευγάρια attribute-value για τη δρομολόγηση των δεδομένων. Πιο αναλυτικά, ο κόμβος συγκέντρωσης (sink node) με βάση μια λίστα από ζευγάρια attribute-value στέλνει σε όλους κόμβους ένα μήνυμα ότι ενδιαφέρεται για κάποια δεδομένα(interest). Οι κόμβοι αποθηκεύουν το μήνυμα αυτό και όταν παρουσιαστεί το επιθυμητό γεγονός, τότε μέσα από τους γειτονικούς τους κόμβους στέλνουν ένα πακέτο gradient, που περιέχει δεδομένα για τον τρόπο δρομολόγησης των πακέτων ενδιαφέροντός τους, προς τον κόμβο συγκέντρωσης με σκοπό τον καθορισμό της καλύτερης διαδρομής για την παράδοση των δεδομένων. Στη συνέχεια με βάση αυτή την διαδρομή γίνεται η μεταφορά των δεδομένων. Τα πλεονεκτήματα είναι ότι, οι κόμβοι χρειάζεται να ξέρουν μόνο τους γειτονικούς τους κόμβους και κάθε κόμβος μπορεί να συλλέγει και να αποθηκεύει δεδομένα. Από την άλλη πλευρά δεν είναι κατάλληλο για όλες τις εφαρμογές και τα ζευγάρια attribute-value πρέπει να προσαρμόζονται σε κάθε εφαρμογή.



Εικόνα 20: Λειτουργία Directed Diffusion

- Στο Rumor Routing οι κόμβοι, όταν αντιληφθούν ένα γεγονός (event), το ανακοινώνουν στο δίκτυο με ένα μήνυμα agent. Οι κόμβοι που λαμβάνουν αυτά τα μηνύματα αποθηκεύουν σε έναν πίνακα το μονοπάτι προς τον κόμβο και το γεγονός που έχει ο καθένας. Έτσι όταν ο κόμβος συγκέντρωσης ζητά δεδομένα για ένα γεγονός πρέπει να βρεθεί η κατάλληλη εγγραφή σε κάποιο κόμβο. Στη συνέχεια, με βάση την εγγραφή δρομολογούνται τα δεδομένα από την πηγή προς τον προορισμό. Με αυτόν τον τρόπο δεν υπάρχει ανάγκη να πλημμυρίσει το δίκτυο με ερωτήματα για κάποια δεδομένα. Υπάρχει μόνο μια διαδρομή μεταξύ πηγής και προορισμού. Τέλος, αυτό το πρωτόκολλο δουλεύει καλά μόνο για μικρό αριθμό γεγονότων.

- Ο αλγόριθμος MCFA (Minimum Cost Forwarding Algorithm) εκμεταλλεύεται το γεγονός ότι η κατεύθυνση της δρομολόγησης είναι γνωστή και πάντα προς τον σταθερό εξωτερικό σταθμό βάσης. Άρα, ένας κόμβος δεν χρειάζεται να έχει μια μοναδική ταυτότητα ούτε να διατηρεί έναν πίνακα δρομολόγησης. Αντ΄ αυτού, κάθε κόμβος διατηρεί την καλλίτερη εκτίμηση από πλευράς κόστους για μια διαδρομή από αυτόν προς το σταθμό βάσης. Κάθε μήνυμα από τον κόμβο μεταδίδεται στους γείτονες του. Όταν ένας κόμβος λαμβάνει το μήνυμα, ελέγχει αν είναι στη διαδρομή ελαχίστου κόστους μεταξύ του κόμβου πηγή και του σταθμούς βάσης. Εάν αυτό ισχύει, τότε μεταδίδει το μήνυμα στους γειτονικούς του κόμβους και αυτό γίνεται μέχρι να φτάσουν τα δεδομένα στον προορισμό τους. Τέλος, υπάρχει ένα μηχανισμός για τον καθορισμό του κόστους των διαδρόμων.
- Η δρομολόγηση βασισμένη στην gradient είναι μια παραλλαγή της Direct Diffusion. Ο κάθε κόμβος αποθηκεύει τον αριθμό των βημάτων (hops) από τον BS προς αυτόν κατά την διάχυση των interest μηνυμάτων. Ως ύψος (height) ενός κόμβου ορίζεται ο ελάχιστος αριθμός των βημάτων (hops) προς τον BS. Η διαφορά ύψους (height) μεταξύ ενός κόμβου και των γειτονικών του είναι το gradient που υπάρχει στην σύνδεση τους. Τα πακέτα προωθούνται από την σύνδεση με το μεγαλύτερο gradient. Επιπλέον, όταν από κάποιο κόμβο περνούν περισσότερα μονοπάτια, τότε αυτά ενώνονται. Για να επιταχυνθεί ο ρυθμός μετάδοσης και να εξομαλυνθεί η κίνηση υπάρχουν τρία σχήματα. Το πρώτο είναι η τυχαία επιλογή επόμενου κόμβου, όταν υπάρχουν ισότιμα gradient. Το δεύτερο σχήμα προβλέπει ότι, οι κόμβοι θα αυξάνουν το ύψος τους, όταν έχουν χαμηλή ενέργεια για την εξοικονομήσουν. Σύμφωνα με το τελευταίο σχήμα τα νέα data-streams δεν δρομολογούνται από κόμβους που ήδη έχουν να δρομολογήσουν αλλά ροές δεδομένων (streams). Τέλος, το πρωτόκολλο αυτό είναι ενεργειακά αποδοτικό.
- Το COUGAR είναι ένα Data-Centric πρωτόκολλο που βλέπει όλο WSN σαν μια κατανεμημένη βάση δεδομένων. Μέσα από δηλωτικά ερωτήματα προς τους κόμβους συλλέγονται τα δεδομένα in-network και αυτό γίνεται για να αποφευχθεί η επεξεργασία μηνυμάτων σε επίπεδο δικτύου. Για τον λόγο αυτό εισαγάγεται ένα ενδιάμεσο στρώμα. Ο BS είναι αυτός που καθορίζει τα

ερωτήματα και τον leader κόμβο από τον οποίο λαμβάνει όλα τα δεδομένα. Μεγαλύτερες επικεφαλίδες και μεγαλύτερη κατανάλωση ενέργειας είναι το αποτέλεσμα του επιπλέον στρώματος. Ακόμα απαιτείται συγχρονισμός για τη συλλογή των δεδομένων. Τέλος, η επιλογή των leader κόμβων πρέπει να γίνεται δυναμικά για την εξομάλυνση της κίνησης .

- Το ACQUIRE βλέπει και αυτό το WSN σαν μια κατανεμημένη βάση δεδομένων. Όμως, τα ερωτήματα είναι σύνθετα και διαχωρίζονται σε δυο κατηγορίες. Ο BS στέλνει ένα ερώτημα. Ο κάθε κόμβος προσπαθεί να το απαντήσει με βάση τις αποθηκευμένες πληροφορίες και το προωθεί σε άλλους κόμβους. Εάν οι αποθηκευμένες πληροφορίες δεν είναι νέες, οι κόμβοι συγκεντρώνουν πληροφορίες από τους γείτονές τους, εντός ενός ορίου d βημάτων (hops). Μόλις το ερώτημα έχει επιλυθεί, θα σταλεί πίσω στον BS η απάντηση, μέσω της αντίστροφης διαδρομής ή της συντομότερης διαδρομής. Το ACQUIRE μπορεί να ασχοληθεί με σύνθετα ερωτήματα, επιτρέποντας σε πολλούς κόμβους να αποστείλουν ταυτόχρονα τις απαντήσεις τους.

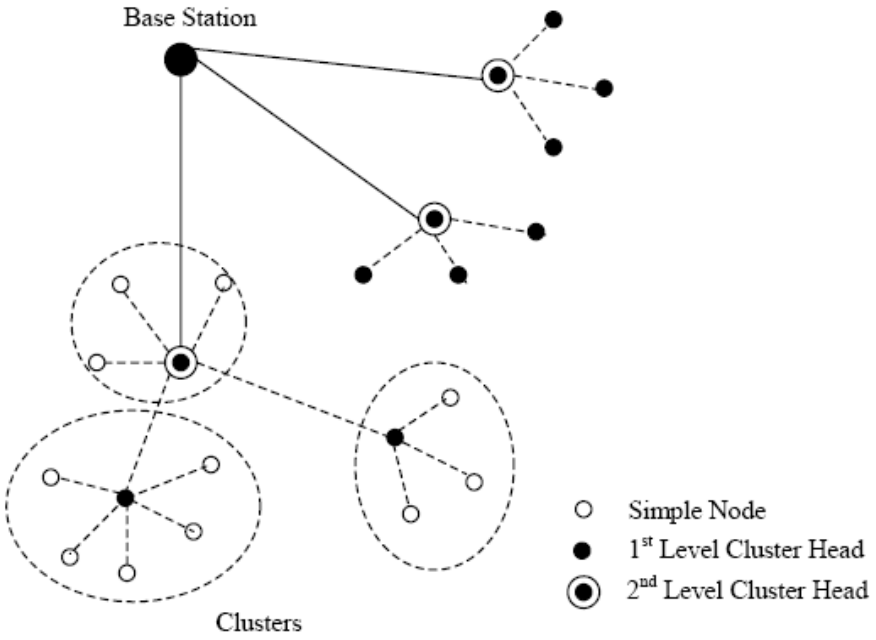
Ιεραρχικά Πρωτόκολλα Δρομολόγησης

Τα ιεραρχικά πρωτόκολλα δρομολόγησης προέρχονται από τα ενσύρματα δίκτυα και έχουν ως πλεονέκτημα τους την επεκτασιμότητα και την αποδοτικότητα τους. Στα WSNs προσδίδουν το γεγονός ότι είναι αποδοτικά και από ενεργειακή άποψη. Οι κόμβοι, που έχουν μεγαλύτερες ενεργειακά δυνατότητες, αναλαμβάνουν λειτουργίες επεξεργασίας και μεταφοράς των δεδομένων, ενώ οι άλλοι κόμβοι περιορίζονται στην παρακολούθηση μιας περιοχής. Επειδή οι μεταδόσεις προς τον BS περιορίζονται γίνεται και περιορισμός της ενεργειακής σπατάλης. Μερικά πιο γνωστά ιεραρχικά πρωτόκολλα είναι τα επόμενα.

- Το Low Energy Clustering Hierarchy (LEACH) είναι ένα πρωτόκολλο βασισμένο στις ομάδες που δημιουργούν οι κόμβοι. Ο κάθε κόμβος τυχαία αποφασίζεται να ηγηθεί μιας ομάδας (Cluster Head ,CH) για κάποιο χρονικό διάστημα. Αυτός αποφασίζει τη μορφή της κωδικοποίησης στην ομάδα του π.χ. CDMA. Ο κάθε CH στέλνει ένα μήνυμα ADV, έτσι ώστε οι κόμβοι που το λαμβάνουν, να αποφασίζουν με βάση την ισχύ του κάθε μηνύματος σε πια ομάδα

ανήκουν. Στη συνέχεια, ο CH καθορίζει ένα πρόγραμμα μετάδοσης προς αυτόν TDMA, δηλαδή, χωρίζει τον χρόνο σε σχισμές και κάθε σχισμή ένας κόμβος μπορεί να στείλει προς αυτόν ενώ οι άλλοι παραμένουν ανενεργοί. Ο CH συλλέγει, συμπιέζει τα δεδομένα και τα στέλνει στο σταθμό βάσης (BS). Τα πλεονεκτήματά του είναι ότι δεν χρειάζεται γενική γνώση του δικτύου και ότι είναι ενεργειακά αποδοτικός. Από την άλλη πλευρά υποθέτει ότι όλοι κόμβοι μπορεί να γίνουν CH και όλοι οι κόμβοι πρέπει να υποστηρίζουν CDMA και TDMA.

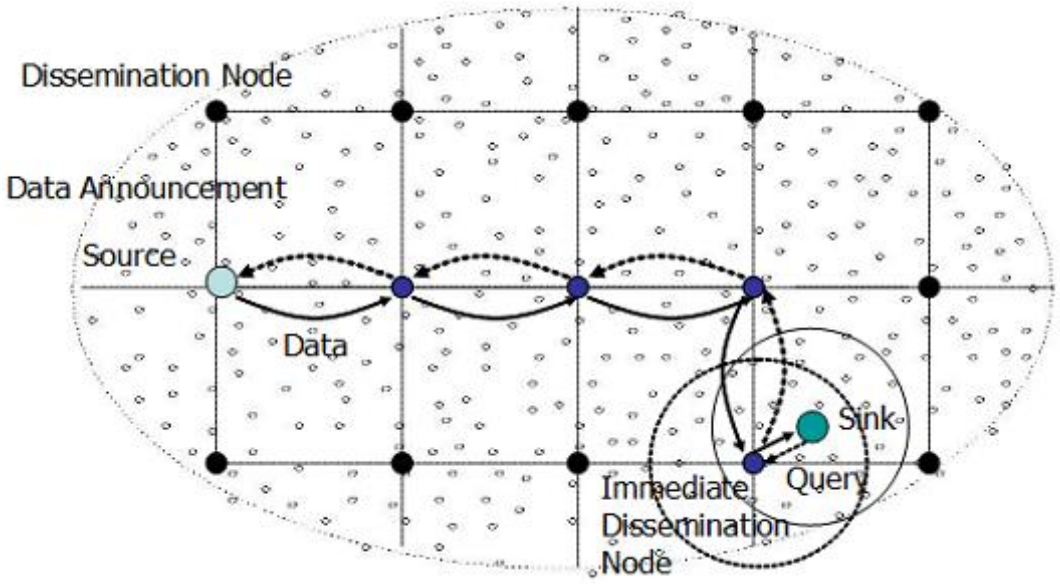
- To Threshold sensitive Energy Efficient Network protocol (TEEN) είναι ένα reactive πρωτόκολλο που οδηγείται από την εμφάνιση των γεγονότων και είναι κατάλληλο εφαρμογές που απαιτούν χρονική ακρίβεια. Η ιδέα είναι ότι, ο κάθε κόμβος στέλνει δεδομένα στο CH μόνο όταν οι τιμές αλλάξουν δραματικά. Οι εκπομπές δεν είναι περιοδικές και έτσι εξοικονομείται ενέργεια. Ο CH στέλνει στην ομάδα του ένα αυστηρό όριο ή ένα πιο χαλαρό όριο. Στην περίπτωση του αυστηρού ο κόμβος στέλνει δεδομένα μόνο, όταν οι τιμές βρίσκονται σε κάποιο συγκεκριμένο διάστημα. Στο χαλαρό όριο κάποιος κόμβος στέλνει δεδομένα, όταν αλλάξει η τιμή περισσότερο από το χαλαρό όριο. Σαν CH είναι όλοι οι κόμβοι της ομάδας περιοδικά. Στις τοπολογίες μπορεί να υπάρχει και ιεράρχηση των ομάδων. Τα μειονεκτήματα του είναι ότι το εν λόγω πρωτόκολλο δεν είναι κατάλληλο για όλες τις εφαρμογές και υπάρχει ασάφεια μεταξύ της χρήσιμης και της πλεονάζουσας πληροφορίας. Στην εικόνα 21 παρουσιάζεται ο τρόπος λειτουργίας του συγκεκριμένου πρωτοκόλλου.



Εικόνα 21: Τρόπος λειτουργίας TEEN

- Το Adaptive Threshold sensitive Energy Efficient Network protocol (APTEEN) είναι μια επέκταση του TEEN με στόχο την εξυπηρέτηση εφαρμογών που απαιτούν περιοδικές μετρήσεις, αλλά και εφαρμογές που ενεργοποιούνται από γεγονότα. Η ιδέα βασίζεται στην αποστολή δεδομένων από έναν κόμβο, όταν έχει περάσει ένα διάστημα CT χωρίς αυτός (ο κόμβος) να έχει στείλει δεδομένα. Το διάστημα CT καθορίζεται από τον CH. Σε σύγκριση με το LEACH το TEEN και το APTEEN καταναλώνουν λιγότερη ενέργεια και άρα έχουν μεγαλύτερο χρόνο ζωής. Από την άλλη πλευρά η ύπαρξή επικεφαλίδας, η σύνθετη διαδικασία δημιουργίας ιεραρχικών ομάδων και η υλοποίηση των λειτουργιών ορίου κάνουν δύσκολη την εφαρμογή του APTEEN και TEEN.

- Στο Two Tier Data Dissemination(TTDD) οι BS μετακινούνται ενώ οι απλοί κόμβοι είναι σταθεροί (Εικόνα 22). Υπάρχουν δύο βαθμίδες¨: η ερώτηση και η προώθηση των συγκεκριμένων δεδομένων. Όταν συμβαίνει ένα γεγονός η πηγή σχηματίζει ένα πλέγμα από κόμβους. Μερικοί κόμβοι έχουν την ικανότητα να αποθηκεύουν δεδομένα σχετικά με την πηγή και το γεγονός (Dissemination Node). Έτσι, όταν ο BS κάνει ένα ερώτημα σε κάποιο κόμβο του πλέγματος, τότε αυτός (ο κόμβος) απευθύνεται στον κοντινότερο Dissemination Node, ο οποίος τον οδηγεί στην πηγή.



Εικόνα 22:Τρόπος Λειτουργίας TTDD

Στα μειονεκτήματα του περιλαμβάνεται το μεγάλο φορτίο για τη διατήρηση της τοπολογίας του πλέγματος και ότι δεν χρησιμοποιούνται αποδοτικά συστήματα εντοπισμού θέσης στα σύγχρονα WSNs.

Δρομολόγηση με βάση τη θέση

Στα πρωτόκολλα αυτά, οι κόμβοι έχουν διευθύνσεις και έχουν συστήματα εντοπισμού τη θέσης τους. Οι κόμβοι γνωρίζουν τους γειτονικούς τους κόμβους με βάση την ισχύ του σήματος που λαμβάνουν. Επειδή αυτοί οι κόμβοι καταναλώνουν πολύ ενέργεια, πρέπει να απενεργοποιούνται, όταν δεν στέλνουν δεδομένα. Μερικά πρωτόκολλα δρομολόγησης παρουσιάζονται παρακάτω.

- Το Sequential Assignment Routing (SAR) είναι ένα πρωτόκολλο βασισμένο στον πίνακα δρομολόγησης κάθε κόμβου με τον οποίο δημιουργούνται πολλαπλά μονοπάτια και έτσι γίνεται ανθεκτική και ενεργειακά συμφέρουσα η δρομολόγηση. Υπάρχουν δένδρα με ρίζα τους γειτονικούς κόμβους του κόμβου συγκέντρωσης (sink node,SN) και έτσι δημιουργούνται μονοπάτια από τους κόμβους προς τον SN. Η ύπαρξη των μονοπατιών εγγυάται τη διόρθωση των λαθών. Το μονοπάτι για τη δρομολόγηση επιλέγεται με βάση τους ενεργειακού

πόρους και το QoS του κάθε μονοπατιού. Το μειονεκτήματά του είναι η δυσκολία να διατηρηθούν οι πίνακες δρομολόγησης σε κάθε κόμβο.

- Το Energy Aware QoS Routing Protocol αποτελείται από τον BS, τους Gateway κόμβους (GN), που είναι οι μόνοι που επικοινωνούν με τους κόμβους μιας ομάδας και διαχειρίζονται την ομάδα. Οι GN επιπλέον επικοινωνούν μεταξύ τους. Η δρομολόγηση μέσα σε μια ομάδα γίνεται με βάση το QoS και τα δεδομένα είναι είτε real-time είτε όχι. Το πρωτόκολλο αυτό βρίσκει τις συνδέσεις με το ελάχιστο κόστος και τις λιγότερες ενεργειακά απώλειες που ικανοποιούν την από άκρη σε άκρη καθυστέρηση της σύνδεσης. Μοντέλα από τα συστήματα αναμονής χρησιμοποιούνται για την καλύτερη δυνατή δρομολόγηση της κίνησης των κόμβων. Μειονέκτημά του είναι ότι, οι GN πρέπει να έχουν αυξημένες δυνατότητες σε σχέση με τους άλλους κόμβους.
- Τέλος, υπάρχει και το SPEED, όπου κάθε κόμβος διατηρεί πληροφορίες για τους γείτονές του και με άπληστη γεωγραφική δρομολόγηση φτιάχνει τις διαδρομές. Προσπαθεί να διατηρήσει μια συγκεκριμένη ταχύτητα για κάθε πακέτο μέσα στο δίκτυο και προσπαθεί να αποφύγει τη συμφόρηση. Η δρομολόγηση είναι επίπεδη και δεν απαιτούνται ιδιαίτερες δυνατότητες από κάποιους κόμβους.

Ακολουθεί ο πίνακας 2 που περιέχει τα περισσότερα γνωστά πρωτόκολλα, τις κατηγορίες που ανήκουν και τα χαρακτηριστικά τους.

	Classification	Mobility	Position Awareness	Power Usage	Negotiation based	Data Aggregation	Localization	QoS	State Complexity	Scalability	Multipath	Query based
SPIN	Flat	Possible	No	Limited	Yes	Yes	No	No	Low	Limited	Yes	Yes
Directed Diffusion	Flat	Limited	No	Limited	Yes	Yes	Yes	No	Low	Limited	Yes	Yes
Rumor Routing	Flat	Very Limited	No	N/A	No	Yes	No	No	Low	Good	No	Yes
GBR	Flat	Limited	No	N/A	No	Yes	No	No	Low	Limited	No	Yes
MCFA	Flat	No	No	N/A	No	No	No	No	Low	Good	No	No
CADR	Flat	No	No	Limited	No	Yes	No	No	Low	Limited	No	No
COUGAR	Flat	No	No	Limited	No	Yes	No	No	Low	Limited	No	Yes
ACQUIRE	Flat	Limited	No	N/A	No	Yes	No	No	Low	Limited	No	Yes
EAR	Flat	Limited	No	N/A	No	No		No	Low	Limited	No	Yes
LEACH	Hierarchical	Fixed BS	No	Maximum	No	Yes	Yes	No	CHs	Good	No	No
TEEN & APTEEN	Hierarchical	Fixed BS	No	Maximum	No	Yes	Yes	No	CHs	Good	No	No
PEGASIS	Hierarchical	Fixed BS	No	Maximum	No	No	Yes	No	Low	Good	No	No
MECN & SMECN	Hierarchical	No	No	Maximum	No	No	No	No	Low	Low	No	No
SOP	Hierarchical	No	No	N/A	No	No	No	No	Low	Low	No	No
HPAR	Hierarchical	No	No	N/A	No	No	No	No	Low	Good	No	No
VGA	Hierarchical	No	No	N/A	Yes	Yes	Yes	No	CHs	Good	Yes	No
Sensor aggregate	Hierarchical	Limited	No	N/A	No	Yes	No	No	Low	Good	No	Possible
TTDD	Hierarchical	Yes	Yes	Limited	No	No	No	No	Moderate	Low	Possible	Possible
GAF	Location	Limited	No	Limited	No	No	No	No	Low	Good	No	No
GEAR	Location	Limited	No	Limited	No	No	No	No	Low	Limited	No	No
SPAN	Location	Limited	No	N/A	Yes	No	No	No	Low	Limited	No	No
MFR, GEDIR	Location	No	No	N/A	No	No	No	No	Low	Limited	No	No
GOAFR	Location	No	No	N/A	No	No	No		Low	Good	No	No
SAR	QoS	No	No	N/A	Yes	Yes	No	Yes	Moderate	Limited	No	Yes
SPEED	QoS	No	No	N/A	No	No	No	Yes	moderate	Limited	No	Yes

Πίνακας 2: Πρωτόκολλα Δρομολόγησης στα Δίκτυα Αισθητήρων

5. Προτεινόμενη Αρχιτεκτονική **Διαχείρισης Δεδομένων**

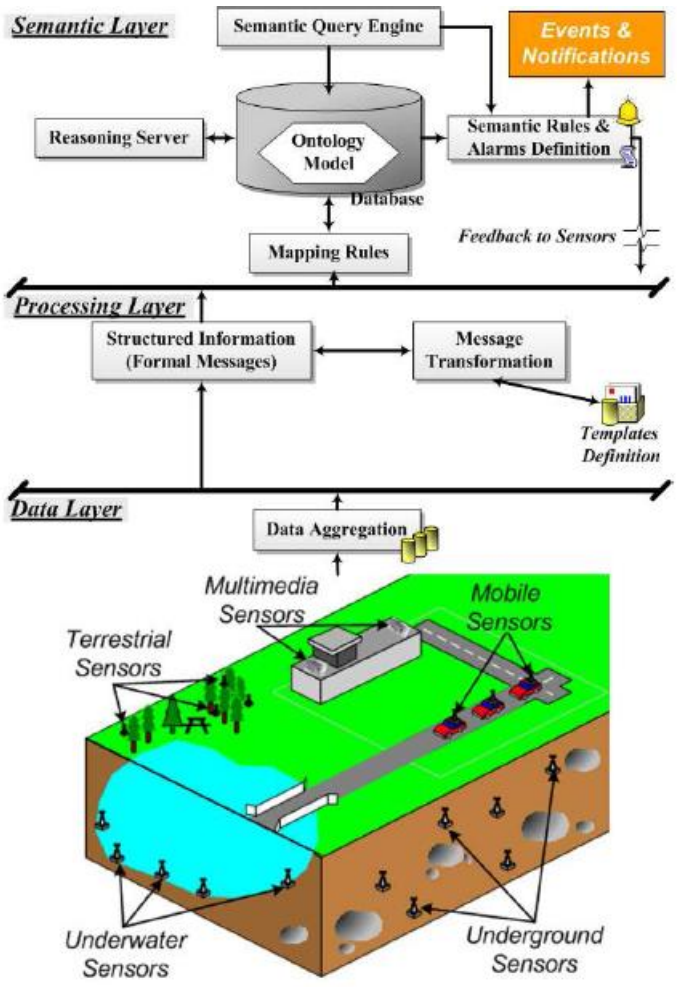
5.1 Ενοποιημένη Γενική Αρχιτεκτονική Διαχείρισης Δεδομένων

Αν και οι σημερινές προσεγγίσεις των δικτύων αισθητήρων αποκλίνουν, υπάρχουν κάποια κοινά χαρακτηριστικά που κάνουν δυνατή τη δημιουργία ενός γενικού σχήματος με ευρεία εφαρμογή για μια αποτελεσματική διαχείριση των δεδομένων, συνδυάζοντας, όσο το δυνατό, περισσότερα επιθυμητά χαρακτηριστικά. Αυτό το σχήμα αντιμετωπίζει τους περιορισμούς που έθεταν μέχρι τώρα τα υπάρχοντα δίκτυα και απαντά στα θέματα της semantic αναπαράστασης των δεδομένων, της διασύνδεσης και της επέκτασης των ετερογενών συστημάτων και του απομακρυσμένου ελέγχου των πόρων των δικτύων.

Η αρχιτεκτονική που προτείνεται και αντιμετωπίζει τα παραπάνω θέματα περιέχει τρία ανεξάρτητα στρώματα. Αυτά είναι:

- Το επίπεδο δεδομένων (Data Layer) που ασχολείται με τη συλλογή και τη συγκέντρωση των δεδομένων.
- Το επίπεδο επεξεργασίας (Processing Layer) που ασχολείται με την ενσωμάτωση και επεξεργασία των δεδομένων μέσω της χρήσης κατάλληλων προτύπων.
- Το επίπεδο σημασιολογικού ιστού (Semantic Layer) που ασχολείται με την εξαγωγή σημασιολογικής πληροφορίας μέσω της αναπαράστασης δεδομένων με οντολογίες και την υποβολή κατάλληλων ερωτημάτων σε αυτές.

Λόγω της ανεξαρτησίας των επιπέδων, οι στρατηγικές που εφαρμόζονται σε κάθε επίπεδο δεν επηρεάζουν τα άλλα και έτσι μπορεί να είναι εκμεταλλεύσιμη η δουλειά που έχει γίνει ξεχωριστά σε κάθε επίπεδο.



Εικόνα 23: Αρχιτεκτονική Διαχείρισης Δεδομένων

Επίπεδο Δεδομένων

Αυτό το στρώμα χειρίζεται την ανακάλυψη των δεδομένων, τη συλλογή και συγκέντρωση σε ένα κεντρικό φορέα. Η αποτελεσματική συγκέντρωση των δεδομένων είναι σημαντική για τη μείωση του κόστους της επικοινωνίας και άρα την επέκταση του χρόνου ζωής του δικτύου. Με βάση την τοπολογία του δικτύου, τη θέση των πηγών και τη λειτουργία συγκέντρωσης, μια βέλτιστη δομή συνάθροισης μπορεί να κατασκευαστεί. Η βέλτιστη συνάθροιση μπορεί να είναι καθορισμένη από άποψη συνολική κατανάλωση ενέργειας, χρησιμοποίηση του εύρους ζώνης και της καθυστέρηση για τη μεταφορά των συλλεχθεισών πληροφοριών από τους απλούς κόμβους στους κόμβους συγκέντρωσης. Η συλλογή δεδομένων μπορεί να πραγματοποιηθεί μέσα από δομημένες ή ελεύθερες από δομή προσεγγίσεις.

Οι δομημένες προσεγγίσεις ταιριάζουν σε εφαρμογές συγκέντρωσης δεδομένων που υπάρχει συγκεκριμένη στρατηγική προώθησης των δεδομένων στους κόμβους συγκέντρωσης. Λόγω του αμετάβλητου σχεδίου κυκλοφορίας, οι δομημένες τεχνικές συνάθροισης υφίστανται τη μικρή επικεφαλίδα διατήρησης της τοπολογίας και είναι επομένως κατάλληλες για τέτοιες εφαρμογές. Αλλά, σε περίπτωση δυναμικών περιβαλλόντων, το μέγεθος της επικεφαλίδα για την κατασκευή και την συντήρηση της δομής μπορούν να αντισταθμίσουν τα οφέλη της δομημένης συνάθροισης δεδομένων. Επιπλέον, οι δομημένες προσεγγίσεις είναι ευαίσθητες στην καθυστέρηση που επιβάλλεται από τους ενδιάμεσους κόμβους, τη συχνότητα της μετάδοσης δεδομένων και το μέγεθος του δικτύου αισθητήρων. Μια κεντρική οντότητα είναι αρμόδια για την ανακάλυψη των νέων κόμβων και την προδιαγραφή της πολιτικής αποκτήσεως των δεδομένων. Η απόκτηση στοιχείων μπορεί να βασίζεται σε γεγονότα, όπου τα στοιχεία στέλνονται από την πηγή και μια μέθοδος καλείται για να τα συλλέξει ή η polling-based, όπου ο κεντρικός κόμβος ρωτά περιοδικά για δεδομένα τους διοικούμενους αισθητήρες.

Στις ελεύθερες, από δομή, προσεγγίσεις δεν υπάρχει καμία προκαθορισμένη κατάσταση και οι αποφάσεις δρομολόγησης για την αποδοτική συνάθροιση των πακέτων πρέπει να ληφθούν άμεσα. Δεδομένου ότι, οι κόμβοι δεν ξέρουν ρητά τους, προς τα πάνω, κόμβους τους, δεν μπορούν να περιμένουν τα στοιχεία από οποιοδήποτε κόμβο πριν διαβιβάζουν τα δικά τους στοιχεία. Αυτές οι προσεγγίσεις

μπορούν να εφαρμοστούν στα δυναμικά περιβάλλοντα και στα ειδικά δίκτυα αισθητήρων, όπου οι κόμβοι ενώνονται συνεχώς και εγκαταλείπουν το δίκτυο και έτσι μια προκαθορισμένη στατική δρομολόγηση και ένα σχέδιο συνάθροισης θα είναι ενδεχομένως ακατάλληλα.

Τέλος, η ασφάλεια είναι ένας σημαντικός τομέας .Όλες οι απαιτήσεις και τα εμπόδια έχουν οριστεί, ενώ έχουν καταγραφεί όλες οι επιθέσεις που έχουν γίνει και οι μέθοδοι για την αντιμετώπισή τους. Για μια βέλτιστη δρομολόγηση και συλλογή δεδομένων, δεδομένου του τύπου των αισθητήρων και σεναρίου ανάπτυξης του δικτύου, θα πρέπει να γίνει ένας συμβιβασμός ανάμεσα στην ασφάλεια και την αποτελεσματικότητα.

Επίπεδο Επεξεργασίας

Τα δεδομένα που λαμβάνει το επίπεδο αυτό χρειάζονται περαιτέρω διεργασία για να είναι προσπελάσιμα από εφαρμογές και γιατί προέρχονται από ετερογενείς πήγες. Αυτό γίνεται από κάποια μοντέλα βασισμένα σε XML. Τα δεδομένα πρέπει να ενσωματώνονται και να μετασχηματίζονται σε μια XML διάταξη (όπως η SensorML), ώστε να είναι προσιτά στον τελικό χρήστη.

Αρχικά, το στρώμα επεξεργασίας ενσωματώνει τον όγκο των εισερχόμενων στοιχείων. Δεν είναι απαραίτητο, ούτε βέλτιστο, σε ορισμένες περιπτώσεις, να διατηρήσει το συνολικό ποσό των δεδομένων. Αυτό θα υπερφόρτωνε το δίκτυο και θα αύξανε τις ανάγκες του σε σχέση με τη διατήρησή του, θα απαιτούσε περισσότερο χώρο για αποθήκευση και στο τέλος θα αμφισβητούσε την αυτονομία του. Οι αθροιστικές εκθέσεις (όπως ένας μέσος όρος των τιμών των αναφερόμενων) μπορούν να είναι επαρκείς για να περιγράψουν τα μεγέθη προς παρακολούθηση.

Για να προωθηθούν τα δεδομένα στο Semantic στρώμα θα πρέπει να ενσωματωθούν σε μηνύματα κατάλληλα για αυτόματη επεξεργασία. Η SensorML είναι ένα πρότυπο μηνύματος που μπορεί να μεταφέρει τα δεδομένα στο επόμενο επίπεδο καθώς ακολουθεί το Semantic Web και τις έννοιες του Object-association-Object.

Για να γίνουν τα παραπάνω μηνύματα χρειάζεται περισσότερη υπολογιστική ισχύς από αυτή που έχουν οι κόμβοι και άρα είναι απαραίτητη μια κεντρική δομή. Η μορφή των πρότυπων μηνυμάτων και ο μετασχηματισμός τους αποτελούν μια κοινή

συμφωνία για τα μηνύματα που ανταλλάσσονται στο δίκτυο. Ο καθορισμός προτύπων και ο αντίστοιχος μετασχηματισμός τους μπορούν να θεωρηθούν ως ο καθορισμός των σχημάτων XML (XSD) και ένας μετασχηματισμός XSLT (Extensible Stylesheet Language Transformation) προκειμένου να παραχθεί ένα μήνυμα XML.

Επίπεδο Σημασιολογικού Ιστού

Το επίπεδο σημασιολογικού ιστού παίρνει τα δεδομένα από το χαμηλότερο επίπεδο και μέσα από οντολογίες καθορισμένες από τις εφαρμογές δημιουργεί τα context annotations χωρίς ανθρώπινη παρέμβαση. Τα πλεονεκτήματα των τεχνολογιών σημασιολογικού ιστού είναι πολλά και οδηγούν στη διασύνδεση συστημάτων και υπηρεσιών. Τα μέρη του επιπέδου είναι οι κανόνες, το μοντέλο οντολογίας, ο εξυπηρετητής συλλογισμού και τα ερωτήματα.

Οι κανόνες περιγράφουν την επιθυμητή συμπεριφορά των συστημάτων που αλληλεπιδρούν με το περιβάλλον τους. Το μοντέλο οντολογίας χρησιμοποιείται για τον σχεδιασμό των γραφημάτων των μοντέλων, τα οποία αναλαμβάνουν να κάνουν τις απαραίτητες επόμενες ενέργειες. Στη συνέχεια ο εξυπηρετητής συλλογισμού είναι σημαντικός για την λειτουργία του σημασιολογικού ιστού, γιατί είναι απαραίτητη η ύπαρξη μια εικονικής μηχανής για να ξεχωρίζει τα γεγονότα με βάση τις οντολογίες και τελικά δημιουργεί νέα γνώση. Τέλος, τα ερωτήματα σημασιολογικού ιστού είναι ο τρόπος μέσα από τον οποίο ο χρήστης μπορεί να αντλήσει πληροφορίες από την βάση γνώσεων ή να ειδοποιηθεί για κάποιο μήνυμα συναγερμού.

6. Υλοποίηση

6.1 Περιγραφή

Στο κεφάλαιο της υλοποίησης παρουσιάζεται το σύστημα που κατασκευάστηκε στα πλαίσια αυτής της εργασίας. Το σύστημα αυτό βασίζεται στην αρχιτεκτονική που έχει περιγραφεί στο προηγούμενο κεφάλαιο και στα πρότυπα που προσφέρει το SWE. Το γεγονός αυτό είναι αρκετά σημαντικό διότι δίνει στο σύστημα μια καθορισμένη και μια τυποποιημένη μορφή. Με αυτό τον τρόπο το σύστημα είναι δυνατόν να απολαμβάνει τα πλεονεκτήματα που προσφέρει η συγκεκριμένη

αρχιτεκτονική και η SensorML, τα οποία αναπτύχθηκαν στα αντίστοιχα προηγούμενα κεφάλαια. Το σύστημα που δίνεται έχει ενσωματώσει τα δύο πρώτα στρώματα της προτεινόμενης αρχιτεκτονικής (το στρώμα δεδομένων και το στρώμα επεξεργασίας). Στη συνέχεια δίνεται μια σύντομη περιγραφή της υλοποίησης και αναδεικνύονται τα σημαντικά κομμάτια που την απαρτίζουν.

Αρχικά, χρησιμοποιήθηκαν οι αισθητήρες SunSPOT, οι οποίοι έχουν την ικανότητα να επικοινωνούν μεταξύ τους και έτσι να σχηματίζονται δίκτυα. Επιπλέον, έχουν αυξημένες υπολογιστικές δυνατότητες και είναι εύκολα να διαχειριστούν και να προγραμματιστούν. Παράλληλα παρέχονται τα κατάλληλα εργαλεία για την εύκολη συντήρηση τους και τον εύκολο χειρισμό τους. Το SunSPOT Manager είναι το εργαλείο συντήρησης τους, όμως το σημαντικότερο εργαλείο που παρέχεται είναι το solarium. Το εργαλείο αυτό είναι ένας εξομοιωτής που παρέχει εικονικούς κόμβους, στους οποίους μπορούν να φορτωθούν όλες οι εφαρμογές που κατασκευάστηκαν. Ακόμα επιτρέπει την αλληλεπίδραση των εικονικων κόμβων με τα πραγματικά SPOTs και άρα είναι δυνατόν να κατασκευαστούν οι επιθυμητές τοπολογίες. Με άλλα λόγια επιτρέπεται η ανταλλαγή πακέτων μεταξύ των πραγματικών κόμβων και των εικονικών. Με τον τρόπο αυτό δεν είναι απαραίτητο να έχει κάποιος μεγάλο αριθμό κόμβων για να δει πως μπορεί να λειτουργούν οι διάφορες τοπολογίες του συστήματος.

Ένας ακόμα στόχος της υλοποίησης που δίνεται είναι η εφαρμογή των στρωμάτων της προτεινόμενης αρχιτεκτονικής. Πιο συγκεκριμένα, για το στρώμα δεδομένων γίνεται η συλλογή των δεδομένων από τους κόμβους σε μια κεντρική δομή που αποθηκεύει όλα τα δεδομένα. Κάθε κόμβος χρησιμοποιεί για την αποστολή των δεδομένων μικρά πακέτα, τα οποία σαν επικεφαλίδα περιέχουν μόνο τη διεύθυνση του κόμβου που κατέγραψε τα συγκεκριμένα δεδομένα. Επίσης, κατασκευάστηκαν τέσσερις ξεχωριστές τοπολογίες από τις οποίες αντλούνται τα δεδομένα. Για κάθε τοπολογία αναπτύχθηκαν οι αντίστοιχες εφαρμογές και χρησιμοποιήθηκαν επιλεγμένες πόρτες επικοινωνίας. Τέλος, μέσα από καθορισμένους κόμβους τα δεδομένα κατέληγαν σε έναν κεντρικό υπολογιστή και αποθηκεύονταν σε μια βάση δεδομένων. Σε επόμενα αντίστοιχα κεφαλαία αναπτύσσονται τα παραπάνω.

Το σύστημα προβλέπει και την επεξεργασία των δεδομένων, κατά αντιστοιχία, με την αρχιτεκτονική που παρουσιάστηκε παραπάνω. Το στρώμα επεξεργασίας των δεδομένων του συστήματος, που υλοποιήθηκε, προβλέπει την ενσωμάτωση των μετρήσεων σε ένα XML σχήμα, το οποίο είναι της μορφής της SensorML. Τα XML αρχεία που παράγονται είναι σημαντικά, διότι μπορεί άλλες εφαρμογές να τα χρησιμοποιήσουν για περαιτέρω επεξεργασία ή να προωθηθούν σε ένα στρώμα με semantic δυνατότητες.

Επιπλέον, σε αυτό το επίπεδο παρουσιάζονται τα αποτελέσματα με βάση τις μετρήσεις που υπάρχουν στην βάση δεδομένων του κεντρικού υπολογιστή. Στον υπολογιστή τρέχει ένας εξυπηρετητής, ο οποίος μπορεί να απαντάει σε αιτήσεις των χρηστών και έτσι να παρουσιάζονται σε τα αποτελέσματα που αυτοί επιθυμούν. Πιο αναλυτικά στους χρήστες δίνονται δυο εφαρμογές για να επιλέξουν αυτήν που επιθυμούν. Στο αντίστοιχο επόμενο κεφάλαιο περιγράφονται τα παραπάνω πιο αναλυτικά.

Συμπερασματικά, το σύστημα που υλοποιήθηκε στοχεύει στην εφαρμογή της προτεινόμενης αρχιτεκτονικής, ώστε να αναδειχθούν τα πλεονεκτήματα της. Παράλληλα θα πρέπει να υπάρχουν τυποποιημένες διεπαφές, ώστε να είναι δυνατή η διασύνδεση των ετερογενών δικτύων των αισθητήρων. Για τον λόγο αυτό παράγονται τα XML αρχεία. Τέλος, έγινε προσπάθεια, ώστε το σύστήμα να είναι όσο πιο αποδοτικό γίνεται, άλλα ταυτόχρονα να είναι και απλό στη λειτουργία του.

6.2 Χαρακτηριστικά συσκευών Sun SPOT

Το Sun SPOT (Sun Small Programmable Object Technology) (Εικόνα 24) είναι ένας κόμβος ενός ασύρματου δικτύου αισθητήρων και έχει κατασκευαστεί από την Sun Microsystems. Βασίζεται στο πρότυπο IEEE 802.15.4 για την επικοινωνία του με τους άλλους κόμβους. Σε αντίθεση με άλλου είδους κόμβους που έχουν αναπτυχθεί, το Sun SPOT χρησιμοποιεί την Squawk Java Virtual Machine.

Εικόνα 24:SunSPOT

Στόχος των κατασκευαστών ήταν να πετύχουν τη κατασκευή μιας συσκευής που να χωράει στην παλάμη ενός ανθρωπινου χεριού και να είναι ,όσο το δυνατό, πιο εύκολη στον προγραμματισμό της.

Σε επίπεδο Hardware αποτελείται από (Εικόνα 25):

Processing:

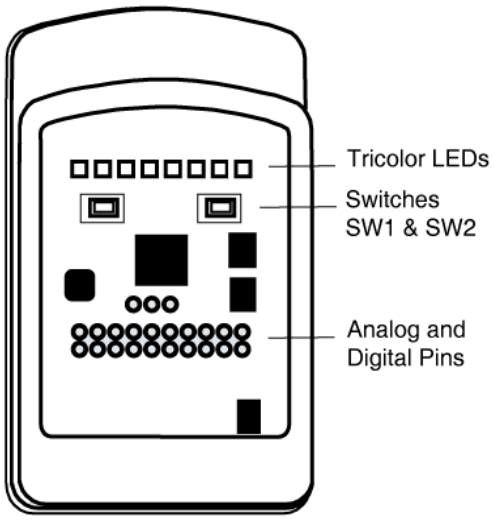
- Επεξεργαστής 180 MHz 32 bit ARM920T- 512K RAM – 4M Flash
- 2.4 GHz IEEE 802.15.4 radio με ενσωματωμένη κεραία
- AT91 timer chip
- Θύρα USB

Sensor Board:

- 2G/6G μετρητές επιτάχυνσης και στις τρεις διαστάσεις
- αισθητήρα θερμοκρασίας
- αισθητήρα φωτεινότητας
- 8 tri-color LEDs
- 6 αναλογικές εισόδους
- 2 διακόπτες
- 5 γενικού σκοπού I/O pins και 4 pins εξόδου υψηλής έντασης

Μπαταρία

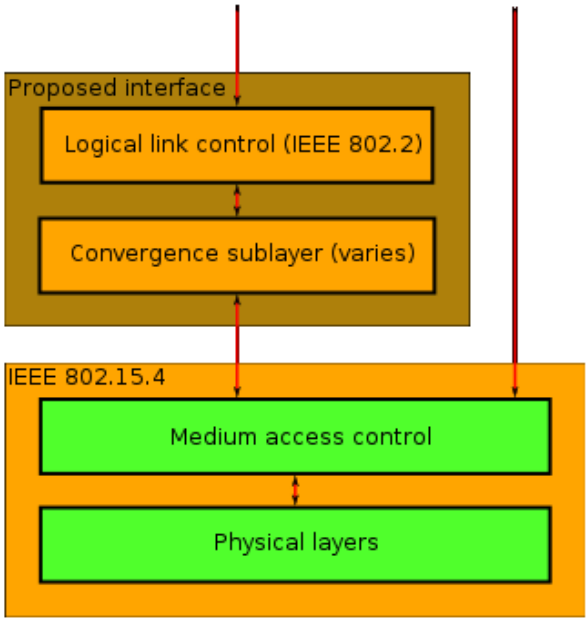
- 3.7 Volt επαναφορτιζόμενη 750 mAh lithium-ion battery
- 30 uA deep sleep mode
- Αυτόματη διαχείριση μπαταρίας μέσω του λογισμικού



Εικόνα 25: Τα Μέρη του SunSPOT

IEEE 802.15.4

Το πρότυπο IEEE 802.15.4 καθορίζει το Φυσικό Επίπεδο(Physical Layer) και το επίπεδο/πρωτόκολλο Ελέγχου Πρόσβασης Μέσου (Media Access Control , MAC) για χαμηλού ρυθμού μετάδοσης ασύρματα προσωπικά δίκτυα (Low Rate Wireless Personal Area Networks LR-WPANs) (Εικόνα 26). Πάνω από αυτό το πρότυπο μπορούν να αναπτυχθούν άλλα πρότυπα με σκοπό να ενσωματώσουν και παραπάνω επίπεδα της διαστρωμάτωσης OSI έτσι ώστε να παραταχθεί ένα ολοκληρωμένο πρότυπο π.χ. ZigBee.



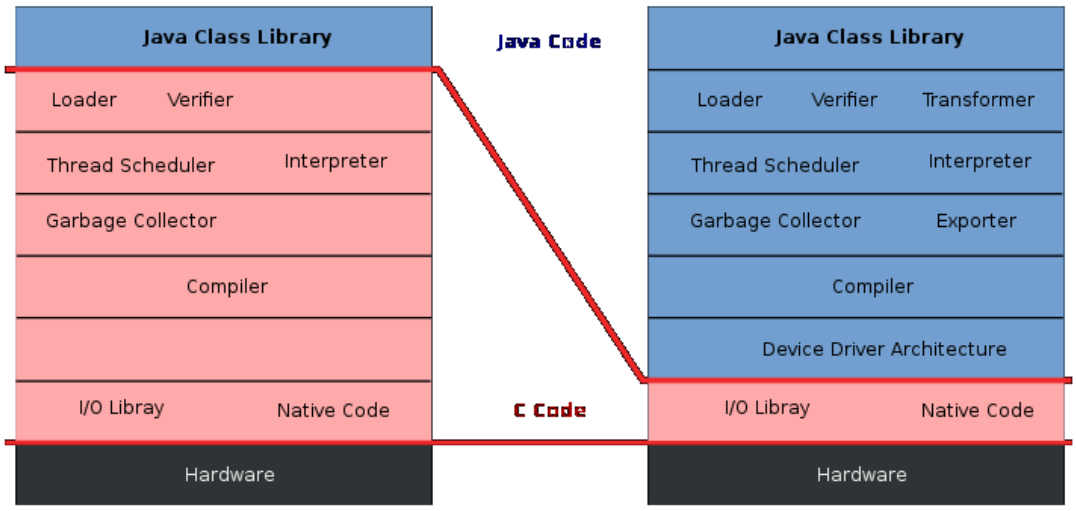
Εικόνα 26: Τα στρώματα του 802.15.4

Στόχος του πρότυπου είναι να προσφέρει ασύρματη επικοινωνία μεταξύ συσκευών με χαμηλό κόστος και χαμηλή ταχύτητα ώστε τελικά να είναι μειωμένη η κατανάλωση ενέργειας .

Squawk Virtual Machine

Το Squawk είναι μια Java Micro Edition εικονική μηχανή για ενσωματωμένα συστήματα και μικρές συσκευές. Οι περισσότερες εικονικές μηχανές για την Java είναι γραμμένες σε χαμηλού επιπέδου γλώσσες όπως η C/C++ και assembler. Αυτό που κάνει διαφορετικό το Squawk είναι ότι, ο πυρήνας του είναι γραμμένος κυρίως σε Java. Η υλοποίηση της VM σε Java δίνει τη δυνατότητα της φορητότητας και την εύκολη διαχείριση των πόρων κάθε εφαρμογής. Οι διαφορές του Squawk από μια συνηθισμένη VM φαίνονται στην εικόνα 27.

Standard Java VM Squawk Java VM

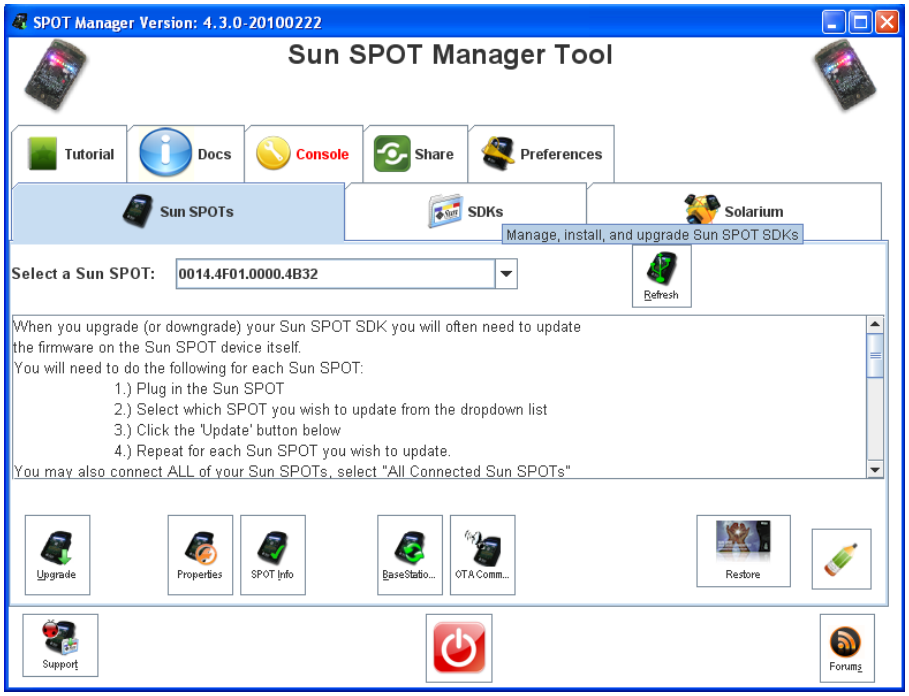


Εικόνα 27: Διαφορές Squawk Java VM και Standard Java VM

Το Squawk παρέχει ένα μηχανισμό απομόνωσης με την βοήθεια του οποίου κάθε εφαρμογή που τρέχει στην VM αναπαριστάται σαν ένα αντικείμενο. Στο Squawk μια ή περισσότερες εφαρμογές μπορούν να τρέχουν σε μια και μόνο Java VM και άρα κάθε εφαρμογή είναι ανεξάρτητη από τις υπόλοιπες.

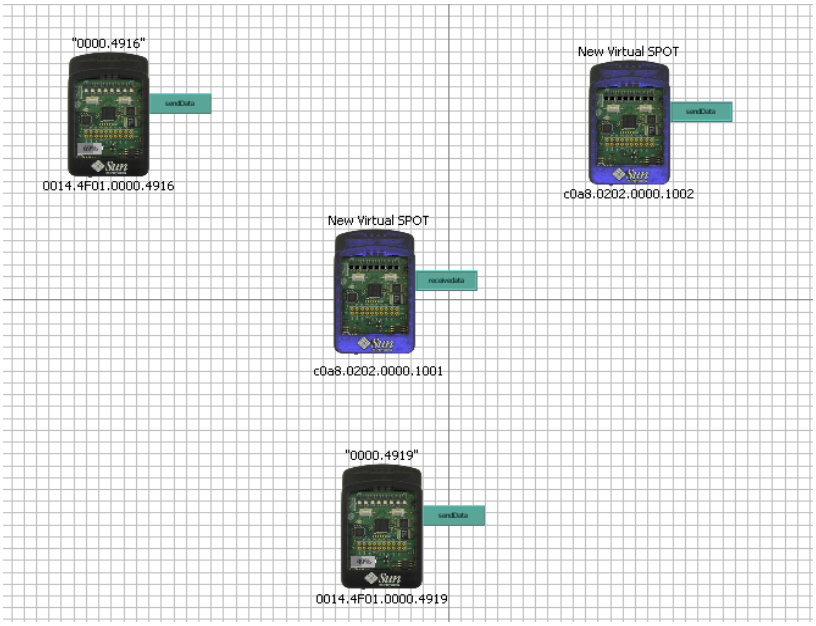
Sun SPOT Manager Tool

Το εργαλείο αυτό παρέχει τη δυνατότητα του ευκολότερου και αμεσότερου χειρισμού των SPOTs. Μέσα από αυτό μπορεί κάποιος να συλλέξει πληροφορίες για τους κόμβους και επιπλέον να ενεργοποιήσει διάφορες λειτουργίες που έχουν τα SPOTs (Εικόνα 28). Μπορεί επίσης να ενημερώνει το λογισμικό της VM και να εγκαθιστά τις αρχικές ρθμίσεις, όταν αυτό είναι αναγκαίο.



Εικόνα 28:SunSPOT Manager

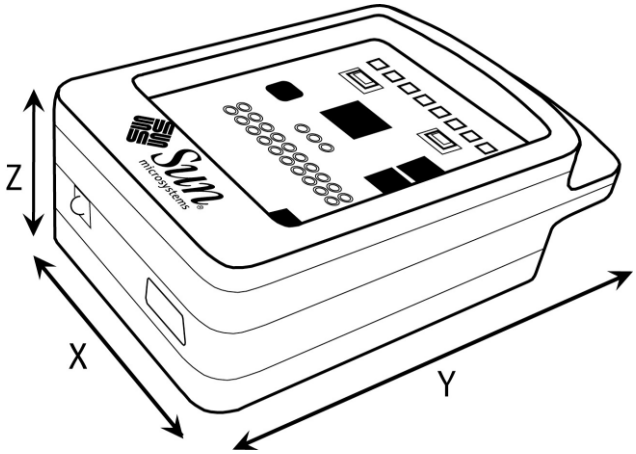
Ακόμα το εργαλείο αυτό προσφέρει και το Solarium, το οποίο είναι ένας εξομοιωτής (Εικόνα 29). Μέσα από αυτό το πρόγραμμα μπορεί κάποιος να δημιουργήσει εικονικούς κόμβούς και να φορτώσει το επιθυμητό λογισμικό. Ακόμα μπορεί να χειριστεί και τα πραγματικά SPOTs. Μπορεί να εγκαταστήσει και να απεγκαταστήσει οποιαδήποτε εφαρμογή επιθυμεί. Τέλος, τα εικονικά και τα πραγματικά SPOTs μπορούν να αλληλεπιδρούν και να σχηματίζουν δίκτυα.



Εικόνα 29: Το Solarium

Αισθητήρες Επιτάχυνσης , Φωτεινότητας και Θερμοκρασίας

Οι αισθητήρες των SPOTs έχουν τη δυνατότητα να καταγράφουν την επιτάχυνση που τους ασκείται και στους τρεις άξονες . Κάθε SPOT περιέχει ένα αδρανειακό αισθητήρα LIS3L02AQ της ST-Microsystems που μετράει σε τρεις διαστάσεις από 2g/6g και δίνει ένα σήμα τάσης . Η κατεύθυνση του άξονα Z είναι κάθετη στην επιφάνεια του SPOT, η κατεύθυνση του άξονα Y είναι παράλληλη στην επιφάνεια του SPOT και κάθετη στην γραμμή των LEDs και τέλος η κατεύθυνση του άξονα X είναι παράλληλη και στην επιφάνεια του SPOT και στη γραμμής των LEDs (Εικόνα 30).



Εικόνα 30:Οι άξονες του SunSPOT

Επιπλέον, τα SPOTs μπορούν να μετρήσουν και τη φωτεινότητα που υπάρχει στο περιβάλλον τους. Για να είναι σε θέση να το κάνουν αυτό διαθέτουν τον αισθητήρα μετατροπής του φωτός σε τάση Toshiba TPS851.

Για να μετατραπούν τα παραπάνω σήματα τάσης σε ψηφιακά δεδομένα χρειάζεται μια συσκευή μετατροπής των αναλογικών δεδομένων σε ψηφιακές τιμές. Αυτή είναι η ADT7411. Η συσκευή αυτή περιέχει και ένα αισθητήρα θερμοκρασίας με δυνατότητα να μετρήσει τιμές από -40 μέχρι 125 C^o .

6.3 SensorML Πρότυπο

Από την υλοποίηση προκύπτουν XML αρχεία που περιέχουν τα δεδομένα που καταχωρήθηκαν στη βάση. Σε κάθε αισθητήρα αντιστοιχεί και ένα αρχείο XML με τις μετρήσεις που έγιναν στο τελευταίο πείραμα από το συγκεκριμένο αισθητήρα. Η μορφή της XML είναι δομημένη κατά SensorML, όπως παρουσιάζεται παρακάτω για τον αισθητήρα με διεύθυνση 0014.4F01.0000.4916.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<data>
  <sensor address="0014.4F01.0000.4916" noofMeasurment="1">
    <time>11 1 110 1265887804671</time>
    <acceleration_z>0.947905477980666</acceleration_z>
```

```
    <temperature>27.5</temperature>
    <light>24.0</light>
  </sensor>
  <sensor address="0014.4F01.0000.4916" noofMeasurment="2">
    <time>11 1 110 1265887834609</time>
    <acceleration_z>0.9532760472610097</acceleration_z>
    <temperature>27.75</temperature>
    <light>34.0</light>
  </sensor>
  <sensor address="0014.4F01.0000.4916" noofMeasurment="3">
    <time>11 1 110 1265887864796</time>
    <acceleration_z>0.9640171858216972</acceleration_z>
    <temperature>27.5</temperature>
    <light>28.0</light>
  </sensor>
  <sensor address="0014.4F01.0000.4916" noofMeasurment="4">
    <time>11 1 110 1265887894937</time>
    <acceleration_z>0.9371643394199786</acceleration_z>
    <temperature>27.75</temperature>
    <light>452.0</light>
  </sensor>
  <sensor address="0014.4F01.0000.4916" noofMeasurment="5">
    <time>11 1 110 1265887925031</time>
    <acceleration_z>0.9425349087003223</acceleration_z>
    <temperature>27.25</temperature>
    <light>391.0</light>
  </sensor>
</data>
```

Από το κώδικα προκύπτει ότι κάθε αρχείο XML αποτελείται από ένα στοιχείο ρίζα το οποίο είναι το data. Το στοιχείο αυτό περιέχει στοιχεία sensor, που αντιστοιχούν στα πακέτα που στέλνει ο κάθε αισθητήρας με τις μετρήσεις τους. Το

στοιχείο sensor έχει δυο χαρακτηριστικά (attributes) το address που είναι η διεύθυνση-αναγνωριστικό του κάθε αισθητήρα και το αύξοντα αριθμό του πακέτου που περιέχει τις μετρήσεις (noofMeasurment). Κάθε στοιχείο sensor περιέχει τέσσερα άλλα στοιχεία τα οποία είναι τα δεδομένα των αισθητήρων και ο χρόνος που έγιναν.

- Το στοιχείο time περιχεί το χρόνο στον οποίο λήφθηκαν τα δεδομένα. Συγκεκριμένα δίνεται ημέρα , ο μήνας (0-11), ο χρόνος(από το 1900) και ο αριθμό τον millisecond από το 1970.
- Το στοιχείο acceleration_z δίνει την τιμή της επιτάχυνσης σε g στον άξονα Z.
- Η θερμοκρασία δίνεται σε βαθμούς Κέλσιου από το στοιχείο temperature.
- Τέλος, η φωτεινότητα δίνεται από το στοιχείο light και η μονάδα μετρήσης είναι το lux.

6.4 Τοπολογίες Διασύνδεσης Αισθητήρων

Τοπολογία Αστέρα

Η τοπολογία αυτή αποτελείται από ένα κεντρικό SunSPOT (1001), το οποίο συνδέεται με τον κεντρικό (Host) υπολογιστή και προωθεί σε αυτόν τα πακέτα που παράλληλα δέχεται από τα αλλά τερματικά Sun SPOT(1002 , 4916 , 4919) και τα οποία λαμβάνουν δεδομένα από τους αισθητήρες τους (Εικόνα 31).

Εικόνα 31: Τοπολογία Αστέρα

Τα περιφερειακά Spots λαμβάνουν δεδομένα, ανά καθορισμένο χρονικό διάστημα και τα στέλνουν στον κεντρικό spot μέσω μιας broadcast σύνδεσης στην πόρτα 37. Αυτό γίνεται μέσα από την εφαρμογή sendData του project simple-leaf που τρέχει στα περιφερειακά spots. Μπορούν να τερματίζουν την αποστολή δεδομένων μέσα από το διακόπτη ένα. Αυτά φαίνονται και στον κώδικα sendDataThread.java.

Στο simple-main SPOT τρέχει η εφαρμογή receiveData, η οποία λαμβάνει όλα τα πακέτα από την πόρτα 37 και τα προωθεί στην πόρτα 67 με μια broadcast σύνδεση, όπου ακούει ο κεντρικός υπολογιστής (Host). Σε αυτόν τρέχει η εφαρμογή xml-databasehost, που δημιουργεί τα xml αρχεία ,ενώ ενημερώνει τη βάση και με τις μετρήσεις.

Δενδρική Τοπολογία

Η τοπολογία αυτή αποτελείται από 7 κόμβους. Ο κεντρικός από αυτούς συσσωρεύει τα δεδομένα και τα προωθεί στο κεντρικό υπολογιστή, ο οποίος τα αποθηκεύει τα επεξεργάζεται και τα παρουσιάζει. Υπάρχουν τέσσερις κόμβοι που συλλέγουν δεδομένα και τα στέλνουν σε δυο ενδιάμεσους κόμβους δηλαδή δυο σε κάθε ενδιάμεσο. Αυτό φαίνεται και στηνεικόνα 32.

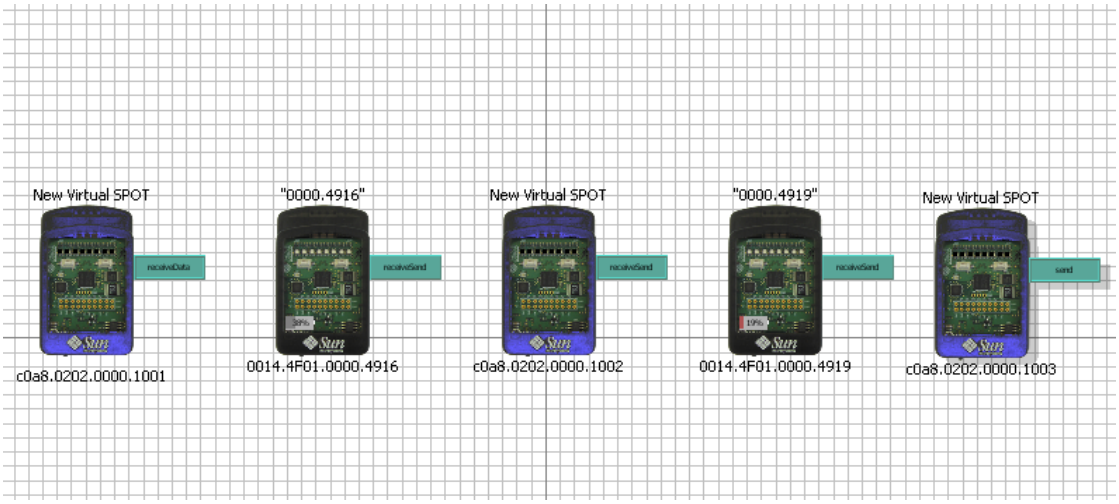
Εικόνα 32:Τοπολογία Δέντρου

Οι κόμβοι 1002 και 1004 δρομολογούν τα πακέτα που φτάνουν σε αυτούς ή που παράγουν οι ίδιοι στον 1001 που είναι ο κεντρικός κόμβος. Αυτό γίνεται με μια broadcast σύνδεση στην πόρτα 47 που ακούει ο 1001. Όμοια οι 4916 και 1003 στέλνουν τα δεδομένα στην πόρτα 37 που ακούει ο 1002 με broadcasting, ενώ οι 1005 και 4919 στέλνουν στην πόρτα 38 που ακούει ο 1004.

Ο 1001 προωθεί όλη την κίνηση που λαμβάνει στον κεντρικό υπολογηστή μέσα από μια broadcast σύνδεση στην πόρτα 67 του host. Σε αυτόν τρέχει η εφαρμογή xml-databasehost, που παράγει xml αρχεία και ενημερώνει τη βάση δεδομένων η οποία περιέχει όλες της μετρήσεις που έχουν γίνει.

Σειριακή Τοπολογία

Σε αυτήν την τοπολογία συνδέονται οι κόμβοι διαδοχικά μεταξύ τους και ο τελευταίος με τον Host, όπου αποθηκεύονται και επεξεργάζονται τα δεδομένα. Στην εικόνα 33 φαίνεται η διάταξη τους.



Εικόνα 33:Τοπολογία Γραμμής

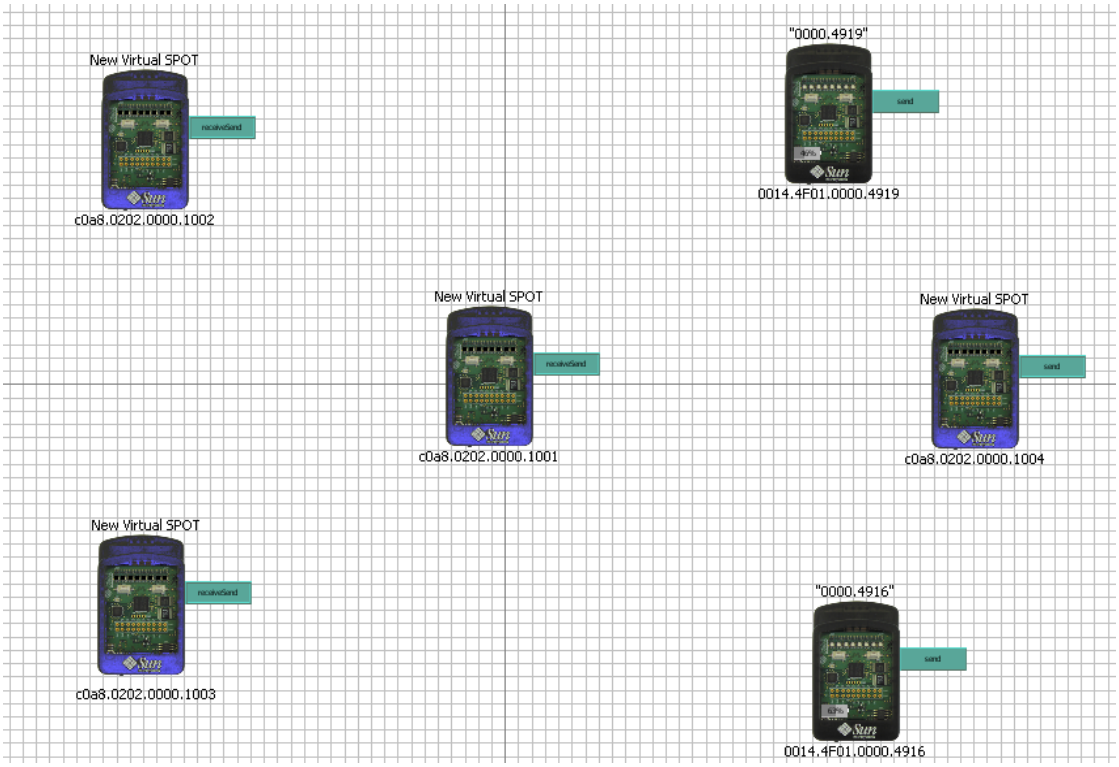
Σε αυτή την τοπολογία ο κόμβος 1001 συνδέεται με μια broadcast σύνδεση στην πόρτα 67 του Host, ενώ ακούει στην πόρτα 37. Ο 4916 δρομολογεί τα πακέτα που λαμβάνει από την πόρτα 38 και τα πακέτα που παράγει στην 37 . Όμοια ο 1002 ακούει από την πόρτα 39 και δρομολογεί στην πόρτα 38, ενώ ο 4919 ακούει από την πόρτα 40 και στέλνει τα πακέτα του στην 39 πόρτα. Τέλος, ο 1003 στέλνει τα

δεδομένα του στην πόρτα 40 και δεν λαμβάνει πακέτα. Όλες οι συνδέσεις είναι broadcast.

Όμοια με τις άλλες τοπολογίες στον κεντρικό υπολογιστή τρέχει η εφαρμογή xml-databasehost και ένας server με δυο εφαρμογές παρουσίασης των αποτελεσμάτων .

Τοπολογία Mesh

Σε αυτήν την τοπολογία υπάρχουν δυο κόμβοι που συνδέονται με τον Host και του δίνουν τα δεδομένα της τοπολογίας. Σε αυτούς συνδέεται ο επόμενος κόμβος σε κάθε έναν ξεχωριστά κάθε φορά που λαμβάνει πακέτο διαλέγοντας με έναν τυχαίο τρόπο. Τέλος σε αυτόν συνδέονται άλλοι κόμβοι, που κάνουν μόνο μετρήσεις. Στην εικόνα 34 οι κόμβοι 1002 και 1003 είναι οι ακραίοι κόμβοι συγκέντρωσης που επικοινωνούν με τον Host. Ο 1001 είναι ο ενδιάμεσος κόμβος που δρομολογεί τα πακέτα είτε στον 1002 είτε στο 1003 με τυχαίο τρόπο. Οι 4916, 4919 και 1004 είναι οι κόμβοι που συλλέγουν δεδομένα. Επιπλέον και οι κόμβοι 1002 και 1003 συλλέγουν μετρήσεις.



Εικόνα 34: Τοπολογία Mesh

Οι 4916, 4919 και 1004 συνδέονται στον 1001 στην πόρτα 37 και αυτός με βάση τη συνάρτηση random δρομολογεί τα πακέτα που φτάνουν είτε στην πόρτα 37 και στον κόμβο 1002 είτε στην πόρτα 38 και στον κόμβο 1003. Τέλος αυτοί επικοινωνούν με τον Host που ακούει στην πόρτα 67. Όλες οι συνδέσεις είναι broadcast.

Στον κεντρικό υπολογιστή τρέχει η εφαρμογή xml-databasehost που παράγει τα xml αρχεία με τις μετρήσεις του πειράματος και ενημερώνει την βάση του Host. Η βάση αυτή χρησιμεύει και για την παρουσίαση των αποτελεσμάτων από τις εφαρμογές του server που τρέχει στον Host.

6.5 Ανάπτυξη Λογισμικού

Η Εφαρμογή Στον Κεντρικό Υπολογιστή

Στο Host τρέχει η εφαρμογή xml-databasehost, η οποία παραλαμβάνει τα πακέτα από κάθε τοπολογία, παράγει κάποια xml αρχεία και ενημερώνει τη βάση δεδομένων, που υπάρχει στον Host για τις μετρήσεις που παρέλαβε. Επίσης παράγει και ένα XML αρχείο για κάθε αισθητήρα που πήρε μέρος στο πείραμα. Η εφαρμογή αποτελείται από τις κλάσεις SunSpotHostApplication και χρησιμοποιεί μεθόδους από τις κλάσεις UsefulMethods και databasehandle.

Η εφαρμογή παραλαμβάνει πακέτα από την πόρτα 67. Τα ελέγχει αν έχει δεχτεί μετρήσεις από το συγκεκριμένο αισθητήρα και είτε ενημερώνει το xml αρχείο που αντιστοιχεί στον συγκεκριμένο αισθητήρα είτε παράγει ένα καινούργιο αρχείο εάν θα ήταν η πρώτη φορά που έστειλε μετρήσεις ο αισθητήρας. Μετά ενημερώνεται η βάση δεδομένων και εισάγονται τα δεδομένα των πακέτων σε αυτήν.

Η βάση δεδομένων είναι μια απλή βάση που ονομάζεται sensorsbase και είναι γραμμένη σε MySQL. Αποτελείται από δυο πίνακες τους IDtable και datatable . Ο πρώτος περιέχει όλες τις διευθύνσεις– αναγνωριστικά των αισθητήρων που στέλνουν δεδομένα και άλλος έχει όλες τις μετρήσεις που έχουν ληφθεί.
Οι εντολές για την δημιουργία της βάσης είναι οι παρακάτω:

- `create database sensorsbase ;`
- `use sensorsbase`
- `CREATE TABLE iDtable (sensorId INT NOT NULL AUTO_INCREMENT PRIMARY KEY, address VARCHAR(200));`
- `CREATE TABLE datatable (address VARCHAR(200), date VARCHAR(200),month VARCHAR(200) ,year VARCHAR(200),time VARCHAR(200), acceleration VARCHAR(200) , temperature VARCHAR(200), light VARCHAR(200) );`

Ο Κώδικας της εφαρμογής μαζί με τις μεθόδους που χρησιμοποιούνται δίνονται στο παράρτημα στο αντίστοιχο σημείο.

Τοπολογία Αστέρα

Για την τοπολογία αστέρα εχουμε δυο είδη SPOTs: το main και το leaf. Οι εφαρμογές που υλοποιούν τις λειτουργίες που προβλέπονται για κάθε κόμβο είναι η receiveData και η sendData. Η sendData καλεί το sendDataThread, το οποίο είναι ένα νήμα που λαμβάνει τα δεδομένα από τους αισθητήρες και τα ενσωματώνει σε πακέτα, τα οποία στέλνει στο main SPOT. Τα πακέτα που στέλνονται τόσο σε αυτή την τοπολογία όσο και στις υπόλοιπες έχουν τη μορφή 'διεύθυνση' 'αριθμός μέτρησης' 'επιτάχυνση' 'θερμοκρασία' 'φωτεινότητα'. Οι τιμές έχουν ένα κενό ανάμεσα τους για να διαχωρίζονται από το πακέτο. Στη συνέχεια στο main SPOT τρέχει η εφαρμογή receiveData και από αυτή καλείται το νήμα receiveDataThread που λαμβάνει τα πακέτα και το προωθεί στον host. Στο παράρτημα δίνονται οι αντίστοιχοι κώδικες στο κεφάλαιο με τις εφαρμογές των SPOT.

Τοπολογία Δέντρου

Στην τοπολογία δένδρου χρησιμοποιούνται πέντε είδη εφαρμογών για τα SPOTs, που συμμετέχουν. Αυτά είναι το simple-main SPOT , το simple-intermediate_1 ,το simple-intermediate_2 , το simple-leaf_1 και το simple-leaf_2. Το simple-main έχει την ίδια λειτουργία και τον ίδιο κώδικα με το main του αστέρα αλλά

ακούει σε διαφορετική πόρτα, όπως περιγράφεται και στο προηγούμενο κεφάλαιο. Όμοια, τα simple-leaf_1 και 2 κάνουν την ίδια λειτουργία με το leaf του αστέρα με τη διαφορά ότι στέλνουν τα πακέτα σε διαφορετικές πόρτες. Ο αντίστοιχος κώδικας δίνεται στο παράτημα. Στη συνεχεία υπάρχουν οι ενδιάμεσοι κόμβοι που συλλέγουν δεδομένα από τους αισθητήρες τους, αλλά και δρομολογούν εισερχόμενα πακέτα προς τον main, όπως έχει αναλυθεί και σε προηγούμενο κεφαλαίο. Ο κώδικας τους είναι ίδιος με τη διάφορα ότι ένας ακούει στην πόρτα 37 και ο άλλος στην πόρτα 38. Η εφαρμογή λέγεται receiveSendData και από αυτήν καλούνται δυο threads. Το πρώτο είναι το SendThread, που παίρνει τα δεδομένα από τον αισθητήρα και τα στέλνει και το δεύτερο είναι το receiveSendThread που δέχεται τα πακέτα και τα προωθεί στον main κόμβο. Ο αντίστοιχος κώδικας υπάρχει στο παράρτημα.

Τοπολογία Σειράς

Στο κεφαλαίο για τις τοπολογίες αναλύθηκε η συγκεκριμένη τοπολογία και επισημάνθηκαν οι πόρτες που χρησιμοποιούνται για κάθε broadcasting σύνδεση. Συμμετέχουν 5 SPOTs από τα οποία το πρώτο έχει το ρόλο του main(gateway, δηλαδή επικοινωνεί με τον Host και είναι σαν κώδικας όμοιος με τον κώδικα για τα main SPOTs των παραπάνω τοπολογιών και ακούει στην πόρτα 37 και στέλνει στη 67 πόρτα του Host. Υπάρχει ένα SPOT , στο οποίο τρέχει η εφαρμογή serial-main και αυτή είναι όμοια με τη main εφαρμογή του αστέρα.

Ακόμα, υπάρχουν τρία ενδιάμεσα (intermediate) SPOTs, που συνδέονται διαδοχικά μεταξύ τους και το πρώτο συνδέεται με το main ενώ το τελευταίο συνδέεται με ένα leaf Spot . Ο κώδικας είναι ο ίδιος με τον κώδικα των ενδιάμεσων, που υπάρχει στην τοπολογία δένδρο με διαφορά στις πόρτες, που χρησιμοποιούνται κάθε φορά.

Τέλος, υπάρχει ένα SPOT leaf, το οποίο στέλνει μόνο δεδομένα και ο κώδικας του είναι ο ίδιος με το κώδικα των leaf SPOTs των παραπάνω τοπολογιών. Οι αντίστοιχοι κώδίκες δίνονται στο παράρτημα.

Τοπολογία Mesh

Στην τοπολογία αυτή, όπως αναλύθηκε και στο κεφαλαίο με τις τοπολογίες, υπάρχουν τρία είδη SPOTs με αντίστοιχα τρεις εφαρμογές. Υπάρχουν δυο SPOTs που τρέχουν την εφαρμογή receiveSend και συνδέονται και επικοινωνούν με τον Host. Το ένα ακούει στην πόρτα 37 και το άλλο στην 38. Ο κώδικας τους δίνεται στο παράρτημα. Υπάρχει ένα SPOT με ρόλο 'διακόπτη' που δρομολογεί τα εισερχόμενα πακέτα σε ένα από τους δυο τελικούς κόμβους. Ο κώδικας του είναι στο παράρτημα με το όνομα mesh-inter. Τέλος,υπάρχουν και τα leaf SPOTs, που είναι όμοια με τα leaf SPOTs των άλλων τοπολογιών παραπάνω.

6.6 Υποστήριξη Αυτόνομων Χαρακτηριστικών

Για το σενάριο αυτό σχεδιάστηκαν δυο εφαρμογές με σκοπό να σχηματιστεί μια τοπολογία της μορφής του αστέρα. Το νέο χαρακτηριστικό που προστίθεται στο σενάριο αυτό είναι η ύπαρξη κάποιων πακέτων, που έχουν τη μορφή σηματοδοσίας και ανταλλάσσονται με σκοπό τη δημιουργία της τοπολογίας. Με βάση αυτά τα πακέτα, ένας νέος αισθητήρας, που εισέρχεται στο δίκτυο, μπορεί να αναγνωρίσει αυτόματα τον πλησιέστερο κόμβο συγκέντρωσης και να εγκαταστήσει επικοινωνία μαζί του. Πιο αναλυτικά, ο κόμβος, που επικοινωνεί με τον κεντρικό υπολογιστή, στέλνει μηνύματα, ανά καθορισμένα χρονικά διάστημα, διαφημίζοντας τη παρουσία του. Από την άλλη πλευρά, οι απλοί κόμβοι δεν στέλνουν μετρήσεις μέχρι να αποκαταστήσουν επικοινωνία με κάποιο κεντρικό κόμβο. Για το λόγο αυτό και αυτά στέλνουν κάποια διαφημιστικά μηνύματα. Όταν κάθε κόμβος λάβει το αντίστοιχο διαφημιστικό μήνυμα αποθηκεύει την διεύθυνση του παραλήπτη και ανοίγεται απευθείας unicast σύνδεση μεταξύ τους. Ο απλός κόμβος αρχίζει τότε και στέλνει τα δεδομένα του και σταματά τα διαφημίστηκα μηνύματα, αντίθετα ο κεντρικός κόμβος συνεχίζει ώστε να συνδεθεί με άλλους κόμβους που πιθανόν δεν έχουν αρχίσει να στέλνουν δεδομένα. Πιο κάτω περιγράφονται αναλυτικά τα παραπάνω και δίνονται και οι αναφορές προς τους κώδικες που τα υλοποιούν.

Αναπτύχθηκαν δυο εφαρμογές. Η πρώτη λέγεται Leaf-Spot και έχει ως σκοπό την αποστολή των δεδομένων προς την βάση των δεδομένων. Η δεύτερη εφαρμογή,

που ονομάζεται Main-Spot, έχει σκοπό τη συλλογή των πακέτων από τα Leaf-Spots και τη μεταφορά τους στον host.

Η εφαρμογή Leaf-Spot υλοποιείται από την κλάση leaf_spot. Από αυτήν καλούνται δύο νήματα, τα Talk_Broadcast και Hear_Broadcast με σκοπό την ανταλλαγή πακέτων πάνω από Broadcast συνδέσεις για την εξασφάλιση της διεύθυνσης του Main-Spot. Πιο συγκεκριμένα κάθε 0,5 δευτερόλεπτα αποστέλλεται ένα πακέτο από το νήμα TalkBroadcast προς την πόρτα 37 με broadcast που ακούει τα Main-Spots. Ακόμα, όταν τα Leaf-Spots λαμβάνουν από την πόρτα 37 πακέτα από τα Main-Spot τότε καλείται το νήμα Talk_Unicast και σταματούν τα άλλα νήματα. Το τελευταίο νήμα είναι αυτό που στέλνει τα δεδομένα στο Main-Spot μέσα από unicast συνδέσεις στην πόρτα 123. Στο παράρτήμα δίνεται ο κώδικας της εφαρμογής για να γίνουν κατανοητές οι λεπτομέρειες.

Η εφαρμογή Main-Spot αποτελείται από την κύρια κλάση Main_Spot από την οποία καλούνται δύο νέα νήματα το Talk_Broadcast και το HearBroadcast. Το πρώτο στέλνει τα απαραίτητα πακέτα προς το Leaf–Spot για να έχει αυτό την διεύθυνσή του. Ενώ το δεύτερο ακούει το πακέτο σηματοδοσίας από τα Leaf-Spots. Στη συνέχεια αποθηκεύει τις διευθύνσείς και καλεί το νήμα Talk_Unicast_M, που είναι υπεύθυνο για την διαχείριση της unicast σύνδεσης με το Leaf-Spot. Για κάθε Leaf-Spot δημιουργείται και ένα νέο νήμα Talk_Unicast_M . Από αυτό τα δεδομένα μεταβιβάζονται στον host μέσα από μια σύνδεση broadcast στην πόρτα 67,. δηλαδή, όπως και στις παραπάνω τοπολογίες. Στο παράρτημα δίνεται ο κώδικας της εφαρμογής Main-Spot.

Με βάση το παραπάνω λογισμικό για την τοπολογία του αστέρα αναπτύχθηκαν εφαρμογές δημιουργίας της τοπολογίας σε μορφή δέντρου. Ο κώδικας των Leaf-Spots και Main-Spots έμεινε ο ίδιος, αλλά με αλλαγές στις πόρτες που ανοίγουν οι συνδέσεις. Ανάμεσα τους υπάρχει μια νέα εφαρμογή που παραλαμβάνει τα πακέτα των Leaf-Spots και τα προωθεί προς το Main-Spot. Η δομή της τοπολογίας είναι η ίδια με αυτή του δέντρου σε προηγούμενο κεφάλαιο. Όμοια, υπάρχουν δύο ενδιάμεσοι κόμβοι και ο καθένας χρησιμοποιεί διαφορετικές πόρτες. Στο παράρτημα δίνεται ο κώδικας της εφαρμογής intermediate_1, ενώ ο κώδικας των Main και των Leaf-Spots είναι ανάλογος με τις εφαρμογές που αναπτυχθηκαν για την τοπολογία του αστέρα (απαιτούνται αλλαγές στις πόρτες).

Οι εφαρμογές αυτές δεν δοκιμάστηκαν στα πραγματικά SPOTs αλλά μόνο στο Solarium, όπου και λειτουργούν, όπως προβλέπεται.

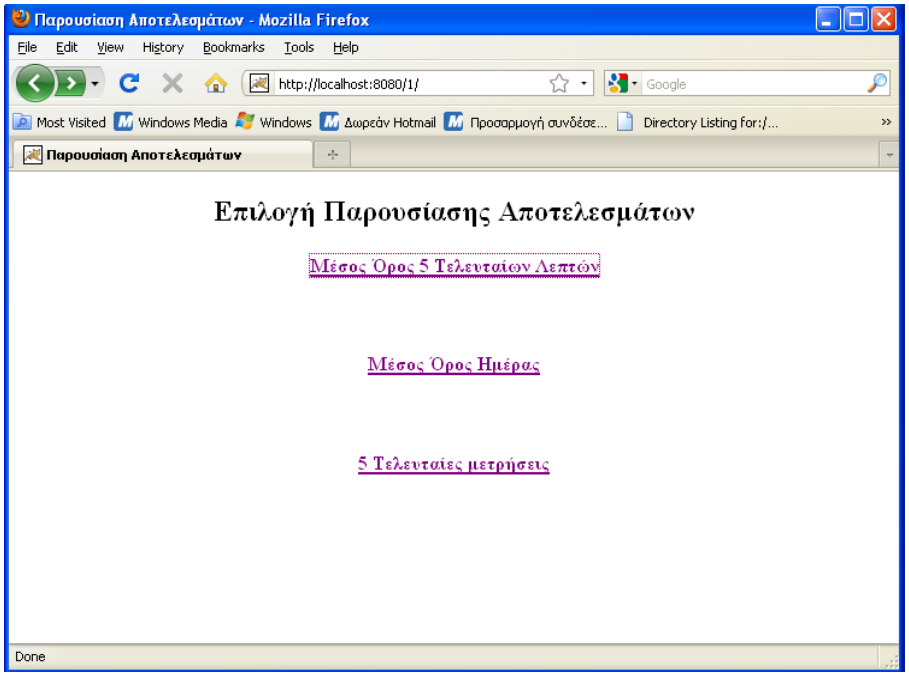
7. Μετρήσεις-Αποτελέσματα

7.1 Δικτυακές Διεπαφές (Web Interface)

Στο τελικό στάδιο χρειάζεται ένας τρόπος, ώστε κάθε χρηστής να έχει τη δυνατότητα να δει τα αποτελέσματα του δικτύου των αισθητήρων, έτσι ώστε να μπορεί να καταλήξει σε κάποιο συμπέρασμα με βάση τις μετρήσεις που έγιναν. Με άλλα λόγια χρειάζεται ένας μηχανισμός παρουσίασης των αποτελεσμάτων.

Για τον παραπάνω λόγο στον κεντρικό υπολογιστή, όπου υπάρχει η βάση, τρέχει και ένας εξυπηρετητής (server). Σε αυτόν υπάρχουν δυο υπηρεσίες, που επικοινωνούν με την βάση, στην οποία υπάρχουν οι μετρήσεις και παρουσιάζουν τα αποτελέσματα. Οι υπηρεσίες αυτές χρησιμοποιούν την τεχνολογία JSP (Java Service Pages).

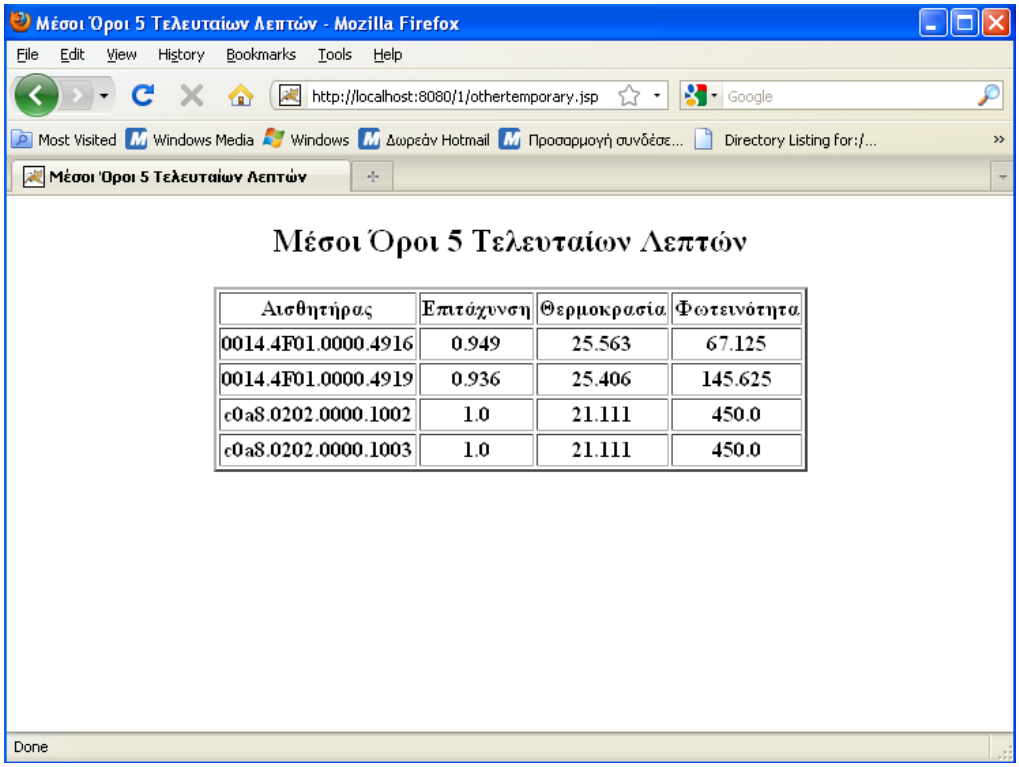
Η πρώτη υπηρεσία παρουσιάζει τους μέσους ορούς των μετρήσεων κάθε αισθητήρα τα τελευταία 5 λεπτά, την τελευταία ημέρα και των τελευταίων πέντε μετρήσεων μέσα στη συγκεκριμένη μέρα, που υποβλήθηκε η αίτηση. Η αρχική σελίδα (index) της υπηρεσίας είναι η εικόνα 35.



Εικόνα 35:Αρχική Σελίδα Πρώτης Υπηρεσίας

Η σελίδα αυτή είναι μια απλή σελίδα σε HTML και ο κώδικας της βρίσκεται στο παράρτημα.

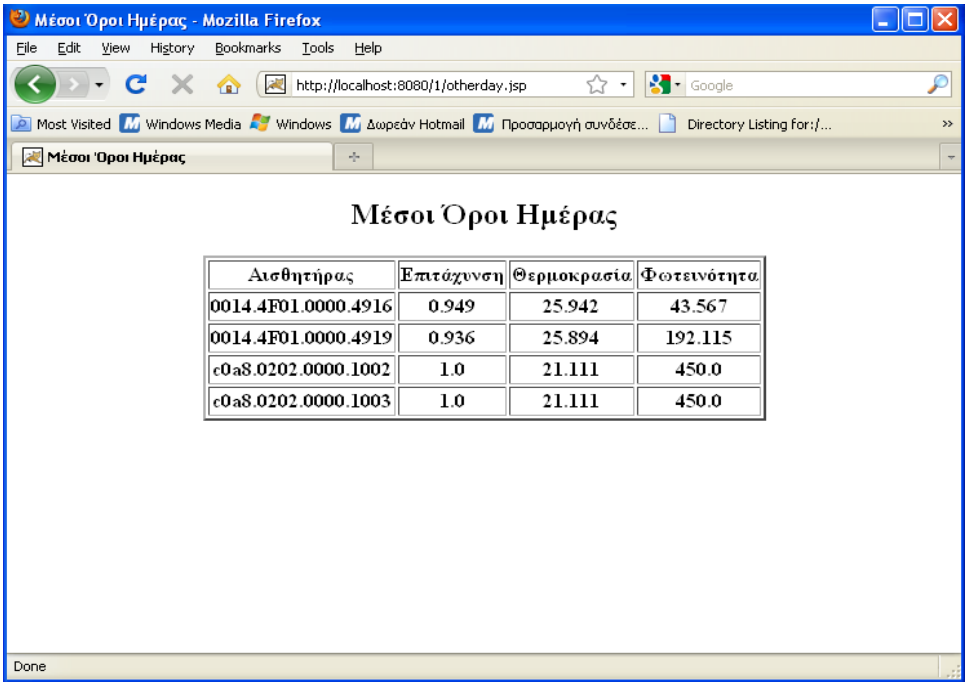
Όπως φαίνεται και παραπάνω, ο χρηστής πατώντας το πρώτο link πηγαίνει στην εφαρμογή othertemporary.jsp, η οποία του δίνει τα αποτελέσματα κατά τα τελευταία πέντε λεπτά από τους αισθητήρες που έχουν στείλει μέτρηση. Αυτό φαίνεται και στην εικόνα 36.



Εικόνα 36: Αποτελέσματα τα Τελευταία 5 Λεπτά

Ο κώδικας για την εφαρμογή αυτής είναι στο παράρτημα και επιπλέον μαζί δίνονται και οι μέθοδοι που χρησιμοποιούνται για τις ερωτήσεις στη βάση για όλη την υπηρεσία.

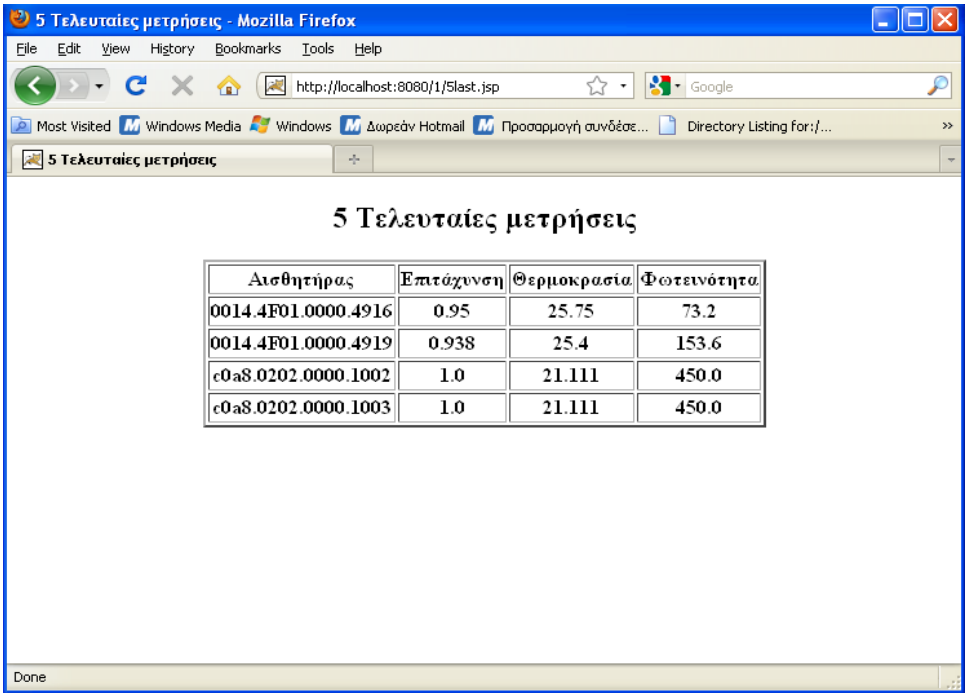
Το δεύτερο link οδηγεί στην εφαρμογή otherday.jsp που φαίνονται οι μέσοι όροι ημέρας για τους αισθητήρες που υπάρχει μέτρηση. Η μορφή των αποτελεσμάτων φαίνεται στην εικόνα 37.



Εικόνα 37: Αποτελέσματα την Τελευταία Μέρα

Ο κώδικας αυτής της εφαρμογής είναι στο παράρτημα, ενώ η μέθοδος που παρουσιάζει τα αποτελέσματα που έχει ήδη δοθεί στο παράρτημα.

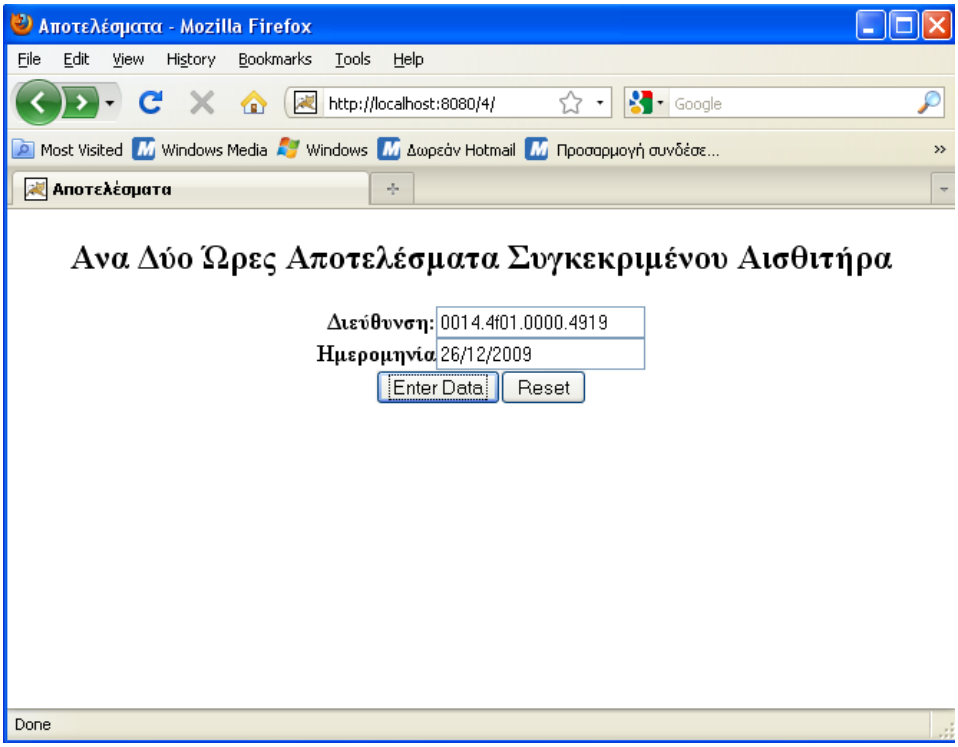
Το τρίτο link οδηγεί στην εφαρμογή 5last.jsp που παρουσιάζει τις πέντε τελευταίες μετρήσεις, όποτε και αν αυτές έγιναν, κατά την τελευταία ημέρα (Εικόνα 38).



Εικόνα 38:5 Τελευταίες Μετρήσεις

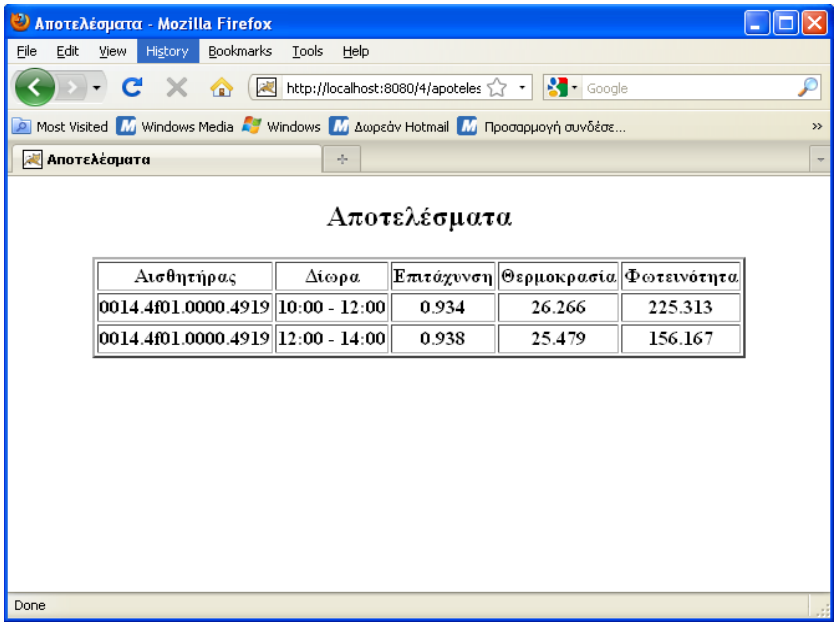
Ο αντίστοιχος κώδικας δίνεται στο παράρτημα, ενώ η μέθοδος που καλείται και δίνει τα αποτελέσματα μετά την ερώτηση στην βάση είναι η ίδια που χρησιμοποιείται και από τις άλλες εφαρμογές.

Στην δεύτερη εφαρμογή ζητούνται η διεύθυνση–ID του αισθητήρα και η ημερομηνία και η εφαρμογή επιστρέφει τους μέσους όρους, σε διάστημα δύο ωρών, όταν έχουμε μετρήσεις για κάποιο δίωρο. Στη εικόνα 39 παρουσιάζεται η αρχική σελίδα της εφαρμογής.



Εικόνα 39:Αρχική Σελίδα Δεύτερης Υπηρεσίας

Τα αποτελέσματα της παραπάνω αίτησης βρίσκονται στη εικόνα 40 και προβάλλουν τον τρόπο παρουσίασης των αποτελεσμάτων.



Εικόνα 40: Αποτελέσματα δεύτερης Υπηρεσίας

Οι κώδικες των παραπάνω φορμών και οι μέθόδοι που χρησιμοποιούνται παρατίθενται στο παράρτημα στα αντίστοιχα σημεία.

Η παρουσίαση των αποτελεσμάτων είναι καθοριστική για την χρησιμότητα του συστήματος. Θα πρέπει να παρέχονται τα αποτελέσματα στους χρήστες όσο τον δυνατόν πιο γρήγορα, αλλά ταυτόχρονα θα πρέπει να είναι και ακριβή. Οι παραπάνω εφαρμογές έχουν ως στόχο την παρουσίαση των μετρήσεων όσο πιο αναλυτικά γίνεται και για το λόγο αυτό παρέχονται περισσότερες από μια υπηρεσίες. Ειδικότερα, η δεύτερη εφαρμογή δίνει τη δυνατότητα να αναλυθούν τα δεδομένα ενός αισθητήρα καθ΄όλη τη διάρκεια της ημέρας που ζητείται και έτσι ο χρήστης μπορεί να παρακολουθήσει την εξελίξη ενός φαινομένου σε μια περιοχή, που παρακολουθείται από ένα συγκεκριμένο αισθητήρα.

Τέλος, μπορούν να αναπτυχθούν και άλλες εφαρμογές σαν επέκταση του συστήματος με βάση την βάση που έχει αναπτυχθεί, ώστε να παρακολουθούνται μεγαλύτερες περιοχές. Με συνδυασμό αισθητήρων μπορεί να επιτευχθεί η παρακολούθηση μεγαλύτερο περιοχών. Επιπλέον, η χρησιμοποίηση των XML αρχείων είναι δυνατό να συμβάλλει στην παρουσίαση των αποτελεσμάτων.

Επίσης, στο κεφάλαιο αυτό έγιναν μια σειρά από μετρήσεις για να δείξουμε τον αριθμό των χαμένων πακέτων, όταν η κίνηση αυξάνεται. Για να αυξηθεί η κίνηση υπάρχουν δυο τρόποι. Ο πρώτος τρόπος προβλέπει την αύξηση των SPOTs έτσι περισσότεροι κόμβοι στέλνουν δεδομένα και αυξάνεται ο αριθμός των πακέτων που

διακινούνται προς τον Host. Ο δεύτερος τρόπος προβλέπει την αύξηση του αριθμού των πακέτων που στέλνει ένας κόμβος σε κάθε λεπτό. Είναι δυνατόν να συνδυαστούν οι δύο τρόποι για να αυξηθεί η κίνηση. Για την τοπολογία του αστέρα και την τοπολογία Mesh είναι δυνατόν να χρησιμοποιηθούν και οι δύο τρόποι. Αντίθετα στις τοπολογίες δέντρο και γραμμής θα πρέπει να αναπτυχθούν και άλλες εφαρμογές για την αύξηση του αριθμού των SPOTs και άρα μόνο με την αύξηση των αριθμό των πακέτων ανά λεπτό μπορεί να μεταβληθεί η κίνηση στο δίκτυο. Το κάθε πείραμα διαρκούσε τριάντα λεπτά και καταγράφηκε ο αριθμός των πακέτων που μεταβιβάστηκαν και ο αριθμός των πακέτων που χάθηκαν.

Τα πειράματα περιλαμβάνουν τις παραπάνω τοπολογίες. Το κάθε πείραμα διαρκούσε τριάντα λεπτά και ο αριθμός των κόμβων ήταν σταθερός. Έτσι εξετάζοντας το XML κείμενο είναι δυνατόν να βρεθούν ο αριθμός των πακέτων που έστειλε ο κάθε κόμβος και των πακέτων που χάθηκαν. Για να επιτευχθεί αυτό χρησιμοποιήθηκε η ιδιότητα “noofMeasurment”. Στο κάθε πείραμα ο αριθμός των πακέτων είναι σταθερός.

Τοπολογία Αστέρα

Με αυτήν την τοπολογία έγιναν πειράματα με βάση τα οποία προκύπτει το διάγραμμα 1. Εξετάστηκαν δυο τοπολογίες. Στην πρώτη τρία SPOTs συνδέονταν στον κεντρικό κόμβο ενώ στην δεύτερη πέντε SPOTs συνδέονταν στον κεντρικό κόμβο και μετά στον Host. Παράλληλα μεταβάλλεται και ο αριθμός των πακέτων ανά λεπτό. Τα δεδομένα βρίσκονται στους πίνακες 3 και 4. Από τα διαγράμματα προκύπτει ότι όσο η κίνηση αυξάνει τόσο αυξάνονται και τα χαμένα πακέτα. Στο κεφάλαιο με τα συμπεράσματα αναλύονται τα αποτελέσματα των μετρήσεων.

3 SPOTs

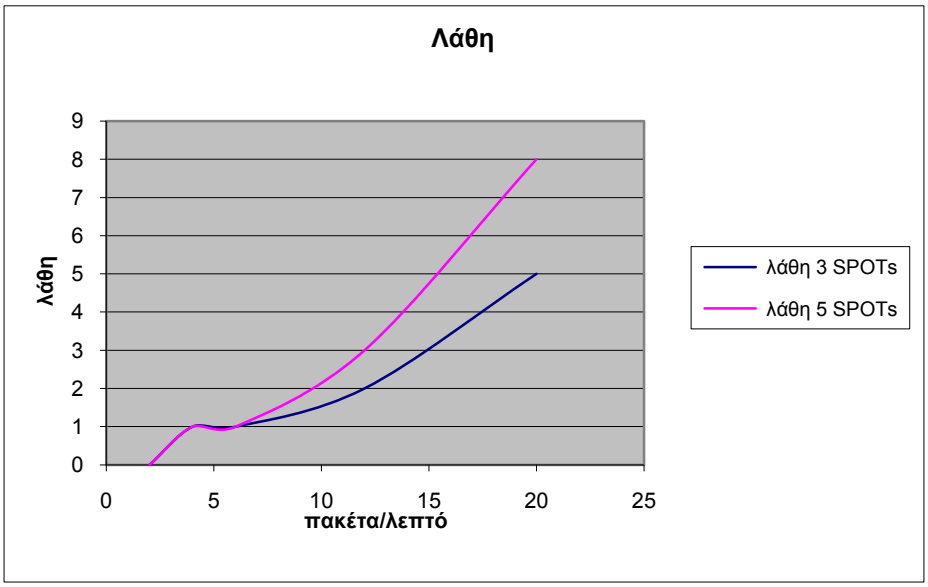
πακέτα / λεπτό	συνολικά λάθη	συνολικά πακέτα
2	0	185
4	1	368
6	1	548
12	2	1087
20	5	1815

Πίνακας 3:Δεδομένα Πειράματος

5 SPOTs

πακέτα / λεπτό	συνολικά λάθη	συνολικά πακέτα
2	0	245
4	1	612
6	1	916
12	3	1820
20	8	3030

Πίνακας 4:Δεδομένα Πειράματος



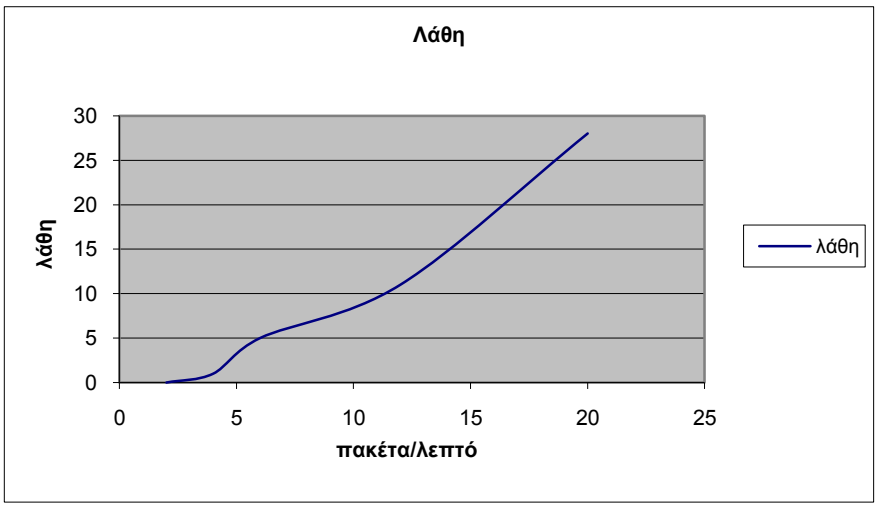
Διάγραμμα 1: Χαμένα Πακέτα στην Τοπολογία Αστέρα

Τοπολογία Δέντρο

Το πείραμα αυτό προβλέπει την καταγραφή των λαθών καθώς η τοπολογία μένει σταθερή και μεταβάλλεται μόνο ο αριθμός των πακέτων, ανά λεπτό. Η τοπολογία μένει σταθερή και είναι αυτή που περιγράφεται στο αντίστοιχο κεφάλαιο. Από τα αποτελέσματα προκύπτει, όπως αναμενόταν, ότι, οι απώλειες των πακέτων αυξάνονται, καθώς η κίνηση αυξάνεται. Τα δεδομένα βρίσκονται στον πίνακα 5 και στο διάγραμμα 2 δίνονται τα αποτελέσματα του πειράματος.

πακέτα / λεπτό	συνολικά λάθη	συνολικά πακέτα
2	0	368
4	1	740
6	5	1113
12	11	2190
20	28	3650

Πίνακας 5:Δεδομένα Πειράματος



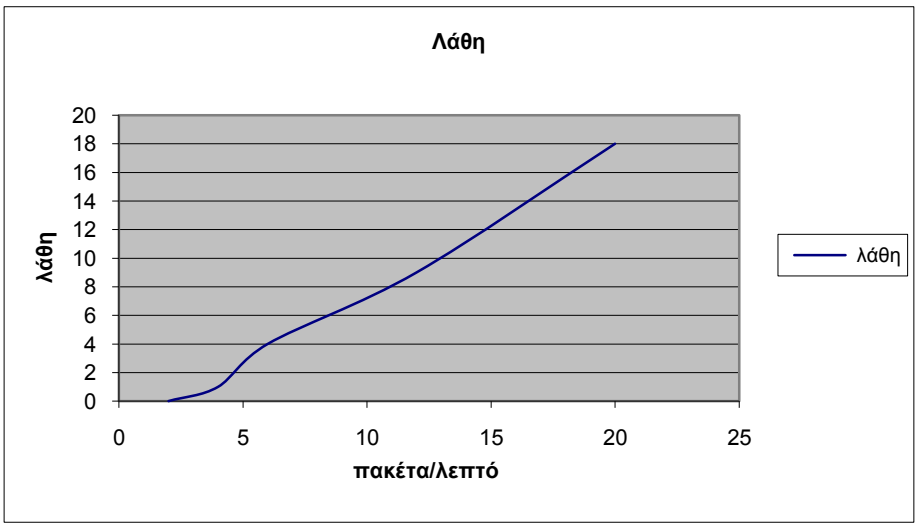
Διάγραμμα 2: Χαμένα Πακέτα στην Τοπολογία Δέντρο

Τοπολογία Σειράς

Στην τοπολογία αυτή οι κόμβοι είναι σταθεροί και έτσι δεν είναι δυνατόν να μεταβληθεί ο αριθμός των SPOTs, ώστε να αυξηθεί η κίνηση του δικτύου. Άρα ο μόνος τρόπος για την αλλαγή της κίνησης είναι να αυξηθούν τα πακέτα, ανά λεπτό. Τα δεδομένα από τα πειράματα της τοπολογίας αυτής βρίσκονται στον πίνακα 6. Στο διάγραμμα 3 απεικονίζονται η τάση της αύξησης του αριθμού των λαθών, καθώς ο αριθμός των πακέτων αυξάνεται, ανά λεπτό. Στη παράγραφο «Συμπεράσματα» αναλύονται περισσότερο τα αποτελέσματα των πειραμάτων.

πακέτα / λεπτό	συνολικά λάθη	συνολικά πακέτα
2	0	243
4	1	487
6	4	730
12	9	1468
20	18	2440

Πίνακας 6:Δεδομένα Πειράματος



Διάγραμμα 3: Χαμένα Πακέτα στην Τοπολογία Σειράς

Τοπολογία Mesh

Στο πείραμα αυτό εξετάστηκε η τοπολογία mesh. Για το λόγο αυτό έγιναν μετρήσεις τόσο με 4 SPOTs όσο και με 6 SPOTs που έστέλναν δεδομένα. Τα αποτελέσματα φαίνονται στους πίνακες 7 και 8 και είναι τα αναμενόμενα, όπως προβλέπονταν και τα λάθη αυξάνονται με την, προς τα πάνω, μεταβολή της κίνησης. Στο διάγραμμα 4 παρουσιάζονται τα αποτελέσματα του πειράματος.

4SPOTs

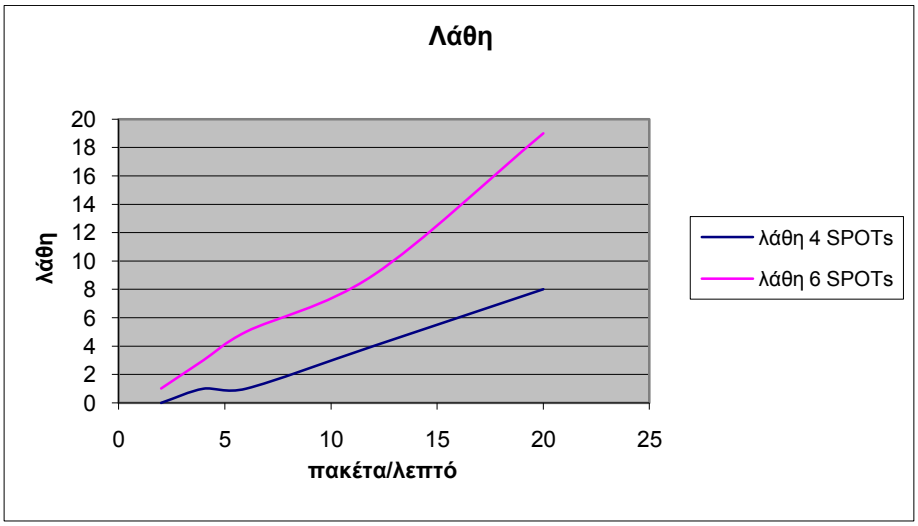
πακέτα / λεπτό	Συνολικά λάθη	συνολικά πακέτα
2	0	243
4	1	484
6	1	725
12	4	1462
20	8	2430

Πίνακας 7:Δεδομένα Πειράματος

6 SPOTs

πακέτα / λεπτό	Συνολικά λάθη	συνολικά πακέτα
2	1	365
4	3	729
6	5	1094
12	9	2184
20	19	3658

Πίνακας 8:Δεδομένα Πειράματος



Διάγραμμα 4: Χαμένα Πακέτα στην Τοπολογία Mesh

Η απώλεια πακέτων παρουσιάζεται, όταν η κίνηση αυξάνεται ή όταν εξωτερικοί παράγοντες αποτρέπουν την ασύρματη επικοινωνία των κόμβων. Επειδή

είναι αδύνατο να μελετηθούν οι εξωτερικοί παράγοντες, μελετήθηκε η συμπεριφορά του δικτύου, όταν μεταβάλλεται η κίνηση που διεκπεραιώνει αυτό. Αυτό γίνεται είτε με αύξηση των κόμβων, είτε με αύξηση των πακέτων που στέλνει ο κάθε κόμβος, είτε με συνδυασμό αυτών. Με βάση τα δεδομένα που συλλέχθηκαν, εξάγεται το συμπέρασμα ότι, στο σύστημα όσο ο αριθμός των πακέτων αυξάνεται τόσο και ο ρυθμός απώλειας τους μεγαλώνει. Πιθανή αιτία για αυτές τις απώλειες ίσως είναι το λογισμικό των κόμβων και το γεγονός ότι υπάρχουν συγκρούσεις κατά την παραλαβή των πακέτων από αυτούς (κόμβους). Δηλαδή υπάρχει η πιθανότητα δύο κόμβοι που προωθούν τα δεδομένα τους στον ίδιο κόμβο στην ίδια πόρτα να στείλουν ταυτόχρονα πακέτα και έτσι κάποιο να χαθεί. Τέλος με την ανάθεση σε κάθε κόμβο διαφορετικών χρονικών διαστημάτων, κατά τα οποία θα μπορούν να εκπέμπουν, είναι δυνατόν να μειωθεί ο κίνδυνος απώλειας πακέτων λόγο συγκρούσεων.

8. Συμπεράσματα – Επεκτάσεις

Το σύστημα που αναπτύχθηκε αποτελεί μια μικρογραφία των βασικών μονάδων ενός μεγαλύτερου και πληρέστερου δικτύου αισθητήρων. Ιδιαίτερη προσοχή δόθηκε στο κομμάτι της συλλογής, αναπαράστασης και επεξεργασίας των δεδομένων των αισθητήρων σε διάφορες τοπολογίες. Τα δεδομένα αποθηκεύονται τόσο σε μια βάση δεδομένων όσο και σε XML αρχεία, τα οποία μπορούν να χρησιμοποιηθούν στη συνέχεια από άλλες εφαρμογές. Μέσω της κατάλληλης αναπαράστασης των δεδομένων, αυτά μπορούν να επεξεργασθούν εύκολα και αποδοτικά από τις εφαρμογές που μπορεί να αναπτύξει ο εκάστοτε διαχειριστής.

Στην παρούσα εργασία, χρησιμοποιήθηκαν αισθητήρες SunSPOT ως κόμβοι αισθητήρων για την μελέτη των μηχανισμών συγκέντρωσης και επεξεργασίας δεδομένων που σχεδιάστηκαν. Οι αισθητήρες αυτοί είναι απλοί στον προγραμματισμό καθώς οι εφαρμογές αναπτύσσονται σε Java και η παρουσία της Squawk Java Virtual Machine δίνει την δυνατότητα στο σχεδιαστή να μην ασχοληθεί με την ανάπτυξη και τον προγραμματισμό του λειτουργικού συστήματος. Προτυποποιήθηκε ένα XML template για κατάλληλη αναπαράσταση δεδομένων και μελετήθηκαν μηχανισμοί συγκέντρωσης και επεξεργασίας δεδομένων σε διάφορες

τοπολογίες. Για τον σκοπό αυτό χρησιμοποιήθηκαν πραγματικές αλλά και εικονικές συσκευές SunSPOT.

Συμπερασματικά, μπορούμε να αναφέρουμε πως με χρήση των υφιστάμενων διεθνών προτύπων για κατάλληλη αναπαράσταση δεδομένων που συλλέγονται από δίκτυα αισθητήρων και με την εφαρμογή σύγχρονων τεχνικών επεξεργασίας, παρέχεται στους διαχειριστές η δυνατότητα αποδοτικής παρακολούθησης και διαχείρισης του δικτύου αισθητήρων και η εξαγωγή γεγονότων σε πραγματικό χρόνο.

Πιθανές επεκτάσεις του προτεινόμενου συστήματος αφορούν την ανάπτυξη ενός δυναμικού μοντέλου το οποίο θα επεξεργάζεται τις μετρήσεις και θα παρέχει μηνύματα με βάση τα οποία θα ενεργοποιούνται κάποιοι μηχανισμοί διαχείρισης και θα ενημερώνονται οι χρήστες του συστήματος. Για τον σκοπό αυτό θα πρέπει να εκμεταλλευθούν τα αρχεία XML (με βάση την SensorML) και επιπλέον να γίνεται ένα φιλτράρισμα των δεδομένων που φτάνουν στο τελικό σύστημα. Προς την κατεύθυνση αυτή μπορούν να χρησιμοποιηθούν και τεχνολογίες σημασιολογικού ιστού έτσι ώστε με τη χρήση κατάλληλων οντολογιών να αυξηθεί η επίγνωση περιβάλλοντος στο δίκτυα αισθητήρων και να εξαχθούν χρήσιμα γεγονότα.

Βιβλιογραφία

[1] Anastasios Zafeiropoulos, Dimitrios-Emmanuel Spanos, Stamatios Arkoulis, Nikolaos Konstantinou, Nikolas Mitrou, Data Management in Sensor Networks using Semantic Web Technologies (2009).

[2] Jane K. Hart, Kirk Martinez, Environmental Sensor Networks: A revolution in the earth system science, Earth-Science Reviews (2006).

[3] Chiara Buratti, Andrea Conti, Davide Dardarl, Roberto Verdone, An Overview on Wireless Sensor Network Technology and Evolution, www.mdpi.org/sensors/ (2009).

[4] ITU Internet Reports 2005 :The Internet of Things Executive Summary

[5] Jonathan W. Hui**,** David E. Culler, Extending IP to Low-Power, Wireless Personal Area Networks, Mesh Networking (2008).

[6] Carl Reed, Mike Botts, John Davidson, George Percivall, OGC® Sensor Web Enablement: Overview and High Level Architecture (2007).

[7] Mike Botts, SensorML and Sensor Web Enablement (2002).

[8] Bhaskar Krishnamachari, Towards Efficient Routing in Wireless Sensor Networks , SenMetrics (2005).

[9] Bhaskar Krishnamachari, Deborah Estrin, Stephen Wicker The Impact of Data Aggregation in Wireless Sensor Networks

[10] Kai-Wei Fan, Sha Liu, Prasun Sinha, Structure-free Data Aggregation in Sensor Networks.

[11] Kemal Akkaya, Mohamed Younis, A Survey on Routing Protocols for Wireless Sensor Networks.
[12] Jamal N. Al-Karaki, Ahmed E. Kamal, Routing Techniques in Wireless Sensor Networks: A Survey.
[13] Lei Shu, Manfred Hauswirth, Long Chen, Jian Ma, Vinny Reynolds, Lin Zhang, Sharing Worldwide Sensor Networks Section 4: Framework for Networked Sensing Systems SWDMNSS 2008 in SAINT 2008, Turku, Finland..
[14] Sun™ SPOT Owner's Manual Red Release 5.0
[15] Sun™ SPOT Theory of Operation Red Release 5.0
[16] Delphine Christin, Andreas Reinhardt, Parag S. Mogre, Ralf Steinmetz, Wireless Sensor Networks and the Internet ofThings: Selected Challenges

Παράρτημα

Εφαρμογή Στον Host

```
/*
 * SunSpotHostApplication.java
 *
 * Created on 11 Δεκ 2009 9:38:57 πμ;
 */

package org.sunspotworld;

import com.sun.spot.peripheral.radio.RadioFactory;
import com.sun.spot.peripheral.radio.IRadioPolicyManager;
import com.sun.spot.io.j2me.radiostream.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.IEEEAddress;
import java.util.Date;
import java.util.Stack;
import java.io.*;
import javax.microedition.io.*;
import nu.xom.*;
import java.sql.DriverManager;
import java.sql.Connection;
/**
 * H Host application dexetai ta paketa apo tin porta 67 kai
 * paragei ta xml arxeia me tin voithia kapoiwn methodwn apo
 * kai  stin  sinexia  apothikeuei ta dedomena kathe paketou
 * stin  vasi dedomenwn pou vrisketai ston Host.
 * Gia na ginoyn auta xreisimopoiountai methodoi apo tis
 * UsefulMethods kai databaseHandle.
 */
public class SunSpotHostApplication {
 private static final int HOST_PORT = 67;
    /**
     * Print out our radio address.
     */
    public void run()throws Exception {
        long ourAddr =
RadioFactory.getRadioPolicyManager().getIEEEAddress();
        RadiogramConnection rCon;
        Datagram dg;
        Connection con;
        Stack s =new Stack();
        Element data = new Element("data");
        Document doc =new Document(data);
        try {
            rCon = (RadiogramConnection) Connector.open("radiogram://:" +
HOST_PORT);
            dg = rCon.newDatagram(rCon.getMaximumLength());
            Class.forName("com.mysql.jdbc.Driver");
            String url = "jdbc:mysql://localhost/sensorsbase";
            con = DriverManager.getConnection(url,"root","");
        } catch (Exception e) {
```

```
             System.err.println("setUp caught " + e.getMessage());
             throw e;
        }
        while (true) {
            try {
                rCon.receive(dg);
                String output = dg.readUTF();
                String[] mesurs = UsefulMethods.measures(output);
                Date date = new Date();
             if(!s.contains(mesurs[0])){
                File file = new File("data-"+mesurs[0]+".xml");
                FileOutputStream fos=new FileOutputStream(file);
                Serializer out= new Serializer(fos,"ISO-8859-1");
                out.setIndent(2);
                data.appendChild(UsefulMethods.xmlCreator(mesurs, date));
                out.write(doc);
                data.removeChildren();
                s.push(mesurs[0]);
            }else{
                Builder parser = new Builder();
                doc = parser.build("data-"+mesurs[0]+".xml");
                data = doc.getRootElement();
                if(!UsefulMethods.packet_arrived(data, mesurs[1])){
                    data.appendChild(UsefulMethods.xmlCreator(mesurs,
date));
                    File file =new File("data-"+mesurs[0]+".xml");
                    FileOutputStream fos=new FileOutputStream(file);
                    Serializer out= new Serializer(fos,"ISO-8859-1");
                    out.setIndent(2);
                    out.write(doc);
                    data.removeChildren();
            }
            }
            if(!UsefulMethods.packet_arrived(data, mesurs[1])){
                if(!(databasehandle.ismember(mesurs[0],con))){
                    databasehandle.insert1(mesurs[0], con);
                }
                if(databasehandle.ismember(mesurs[0], con)){
                    databasehandle.insert2(mesurs, date, con);
                }
            }
            }catch (Exception e) {
                System.err.println("Caught " + e +  " while reading sensor
samples.");
                throw e;
            }
        }
    }

    /**
     * Start up the host application.
     *
     * @param args any command line arguments
     */
    public static void main(String[] args) throws Exception{
        SunSpotHostApplication app = new SunSpotHostApplication();
        app.run();
```

```
    }
}
```

Οι μέθοδοι που χρησιμοποιούνται για την δημιουργία των xml αρχείων είναι οι επόμενες

```
package org.sunspotworld;

import java.io.*;
import java.util.*;
import nu.xom.*;
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author user1
 */
public class UsefulMethods {

    public static String []measures(String input){
        StringTokenizer st = new StringTokenizer(input);
        String [] tokens = new String[st.countTokens()];
        for(int i=0;i<tokens.length;i++){
            tokens[i]=st.nextToken();
        }
        return tokens;
    }

    public static Element xmlCreator(String[] data, Date date){
        Element sensor = new Element("sensor");
        Attribute address = new Attribute("address",data[0]);
        sensor.addAttribute(address);
        Attribute currentMeasur = new Attribute("noofMeasurment",data[1]);
        sensor.addAttribute(currentMeasur);
        Element time = new Element("time");
        time.appendChild(""+date.getDate()+" "+date.getMonth()+" "
                +date.getYear()+" "+date.getTime());
        sensor.appendChild(time);
        Element acceleration = new Element("acceleration_z");
        acceleration.appendChild(data[2]);
        sensor.appendChild(acceleration);
        Element temperature = new Element("temperature");
        temperature.appendChild(data[3]);
        sensor.appendChild(temperature);
        Element light = new Element("light");
        light.appendChild(data[4]);
        sensor.appendChild(light);
        return sensor;
    }
public static boolean packet_arrived(Element data,String noofme){
    boolean packet_found = false;
    Elements sensors = data.getChildElements();
    for(int i=0 ;i<sensors.size();i++){
```

```
        Element sensor =sensors.get(i);
        String noof = sensor.getAttributeValue("noofMeasurment");
        if(noof.equals(noofme)){
            packet_found = true;
        break;
        }
    }
    return packet_found;
}

}
```

Οι μέθοδοι για την ενημέρωση της βάσης είναι οι επόμενες.

```
package org.sunspotworld;
import java.util.*;
import java.sql.*;
import java.util.Date;
/**
 * @author user1
 */
public class databasehandle {

    public static String []measures(String input){
        StringTokenizer st = new StringTokenizer(input);
        String [] tokens = new String[st.countTokens()];
        for(int i=0;i<tokens.length;i++){
            tokens[i]=st.nextToken();
        }
        return tokens;
    }
static public boolean ismember(String address , Connection C ){
      try {
            Statement stmt =C.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM iDtable ");

            while(rs.next()){
                  String col2=rs.getString("address");
            if(col2.equals(address))
                  return true;
            }
      } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      }

      return false;
}
static public  void insert1(String address,Connection C ){
       try {
            PreparedStatement ps = C.prepareStatement("INSERT INTO
iDtable(address) VALUES (?)");
            ps.setString(1, address);
            ps.executeUpdate();
       } catch (Exception e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
      }
 }
static public  void insert2(String[]indata,Date date,Connection C ){
       try {
            PreparedStatement ps = C.prepareStatement("INSERT INTO
datatable(address , date , month , year , time , " +
               "acceleration , temperature , light) VALUES
(?,?,?,?,?,?,?,?)");
            ps.setString(1,indata[0] );
            ps.setString(2, ""+date.getDate() );
        ps.setString(3, ""+date.getMonth() );
        ps.setString(4, ""+date.getYear() );
        ps.setString(5, ""+date.getTime() );
        ps.setString(6, indata[2] );
        ps.setString(7, indata[3] );
        ps.setString(8, indata[4] );
            ps.executeUpdate();
       } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      }
 }
static public  ResultSet getAddresses(Connection C)throws  Exception{
    Statement stm = C.createStatement();
    ResultSet rs = stm.executeQuery("SELECT * FROM iDtable");
    return rs;
}
}
```

Simple-leaf

H sendData

```
package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.io.j2me.radio.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.Utils;

import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
* @author user1
 */
/*
 * Stin efarmogi auti kalitai to sendDataThread gia tis sillogi
 * kai tin apostoli twn dedomenwn
 */
public class sendData extends MIDlet {

    protected void startApp() throws MIDletStateChangeException {
```

```
        new sendDataThread();
    }

    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }
      protected void destroyApp(boolean unconditional) throws
      MIDletStateChangeException {
    }
}
```

και το Thread που καλείται είναι το παρακάτω με σημαντικότερο κομμάτι εκεί που ανοίγεται η σύνδεση στην πόρτα 37 και στέλνει τα πακέτα .

```
package org.sunspotworld;

import com.sun.spot.io.j2me.radiogram.*;

import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.peripheral.*;
import java.io.IOException;
import javax.microedition.io.*;
/**
 *
 * @author user1
 */
/*
 * Sto thread auto silegontai ta dedomena apo toys aisthitires
 * kai stelnontai mesa apo mia Broadcast sindesi stin porta 37
 * sto main SPOT kai akoma elenxei ton diakopti 1 trematismo
*/
public class sendDataThread implements Runnable ,ISwitchListener{

    private EDemoBoard demo ;
    private IAccelerometer3D accel;
    private ILightSensor  light;
    private ITemperatureInput temp;
    private ISwitch [] sw ;
    private ITriColorLED []leds;
    private ITriColorLED led;
    RadiogramConnection con;
    Radiogram outgram;
    public  boolean synch= false;
    private Thread runner ;
    private int numofMesurments =0;

    public sendDataThread() {

        demo = EDemoBoard.getInstance();
        accel = demo.getAccelerometer();
        light = demo.getLightSensor();
        temp = demo.getADCTemperature();
        leds = demo.getLEDs();
        led = leds[4];
        led.setColor(LEDColor.GREEN);
        sw = demo.getSwitches();
```

```
        sw[0].addISwitchListener(this);
        if(runner == null){
            runner = new Thread(this);
            runner.start();
        }
    }

     synchronized  public void run(){
       try{
            String mesur = "";
             con =(RadiogramConnection)
Connector.open("radiogram://broadcast:37");
             outgram = (Radiogram)con.newDatagram(con.getMaximumLength());
        while(!synch){
            try{
                led.setOn();
                outgram.reset();
                double accelz = accel.getAccelZ();
                double tem = temp.getCelsius();
                double ligh = light.getValue();
                numofMesurments++;
                mesur =System.getProperty("IEEE_ADDRESS")+"
"+numofMesurments+" "+accelz+" "+
                        tem+" "+ligh;
                outgram.writeUTF(mesur);
                con.send(outgram);
               runner.sleep(1000);
                led.setOff();
                runner.sleep(29*1000);
        }catch(IOException e){
            System.out.println(e.getMessage());
            return;
        }catch(NullPointerException e){
            System.out.println("provlima sto yield "+e.getMessage());
        }catch(Exception e){
            System.out.println(e.getMessage());
        }
       }
             con.close();
    }catch(IOException e){
        System.out.println("To Leaf exei provlima stin unicast sindesi
"+e.getMessage());
    }
    }

    public void switchReleased(ISwitch sw){
        if(sw==this.sw[0]){
            synch=true;
            try{
                outgram.reset();
                outgram.writeUTF("telos apo to
"+System.getProperty("IEEE_ADDRESS"));
                con.send(outgram);
                con.close();
            }catch(Exception e){
                System.out.println("Provlima sto leaf oten xrisimopiw to
switch"
```

```
                        +e.getMessage());
            }
        }
    }
    public void switchPressed(ISwitch sw){

    }
}
```

Simple-Main

```
package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.io.j2me.radio.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.Utils;

import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
 * @author user1
*/
public class receivedata extends MIDlet {
    //Stin methodo startApp() kalitai ena neo Thraed
    protected void startApp() throws MIDletStateChangeException {
       new receiveDataThread();
    }

    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }
    protected void destroyApp(boolean unconditional) throws
MIDletStateChangeException {
    }
}
```

Το thread που καλείται είναι

```
package org.sunspotworld;

import com.sun.squawk.util.StringTokenizer;
import com.sun.spot.io.j2me.radiogram.*;
import java.io.IOException;
import javax.microedition.io.*;
/**
 *
 * @author user1
 */
/*
```

```
 * Sto Thread auto paralamvanotai ta peketa apo tin porta 37
 * kai proothountai ston Host stin port 67
 * ektos apo tin periptosi poy stelnei "telos" to leaf-SPOT
 */
public class receiveDataThread implements Runnable {

    private Thread runner ;
    private RadiogramConnection con = null;
    private RadiogramConnection BHcon = null;
    private static final int HOST_PORT = 67;

    public receiveDataThread() {
        if(runner == null){
            runner = new Thread(this);
            runner.start();
        }
    }
    synchronized public void run(){
       try{
         con =(RadiogramConnection) Connector.open("radiogram://:37");
        Radiogram ingram =
(Radiogram)con.newDatagram(con.getMaximumLength());
        while(true){

                con.receive(ingram);
                String in = ingram.readUTF();
                 String [] tok = token(in);
               if(tok[0].equals("telos")){
                    System.out.println(in);
               }else{
                    sendDatatoHost(in);
               }

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

        }

    private synchronized  void  sendDatatoHost(String input)throws
Exception{
        BHcon =
(RadiogramConnection)Connector.open("radiogram://broadcast:"+HOST_PORT);
        Datagram sendtohost = BHcon.newDatagram(BHcon.getMaximumLength());

        sendtohost.reset();
        sendtohost.writeUTF(input);
        BHcon.send(sendtohost);
    }
    public static String[] token(String str) {

          StringTokenizer tk = new StringTokenizer(str);
          String tokens[] = new String[tk.countTokens()];
          for (int i = 0; i < tokens.length; i++) {
                tokens[i] = tk.nextToken();
          }
```

```
            return tokens;
      }
}
```

Simple-intermediate

receiveSendData

```
package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.io.j2me.radio.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.Utils;

import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
 * The startApp method of this class is called by the VM to start the
 * application.
 *
 * The manifest specifies this class as MIDlet-N, which means it will not
 * be automatically selected for execution.
 *
 * @author user1

 */
public class receiveSendData extends MIDlet {

    protected void startApp() throws MIDletStateChangeException {
        try{
           RadiogramConnection  sendConnection = (RadiogramConnection)
Connector.open("radiogram://broadcast:47");
            new receiveSendThread(sendConnection);
            new SendThread(sendConnection);
    }catch(Exception e ){
        e.printStackTrace();
    }
    }
    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }
    /**
     * Called if the MIDlet is terminated by the system.
     * I.e. if startApp throws any exception other than
MIDletStateChangeException,
     * if the isolate running the MIDlet is killed with Isolate.exit(), or
     * if VM.stopVM() is called.
     *
     * It is not called if MIDlet.notifyDestroyed() was called.
     *
```

```
     * @param unconditional If true when this method is called, the MIDlet
must
     *    cleanup and release all resources. If false the MIDlet may throw
     *    MIDletStateChangeException  to indicate it does not want to be
destroyed
     *    at this time.
     */
    protected void destroyApp(boolean unconditional) throws
MIDletStateChangeException {
    }
}
```

Το thread που παίρνει τις μετρήσεις είναι το SendThread και είναι το επόμενο .

```
package org.sunspotworld;
import com.sun.spot.io.j2me.radiogram.*;

import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.peripheral.*;
import java.io.IOException;
import javax.microedition.io.*;
/**
 *
 * @author user1
 */
public class SendThread implements Runnable {
private EDemoBoard demo ;
    private IAccelerometer3D accel;
    private ILightSensor  light;
    private ITemperatureInput temp;
    private ISwitch [] sw ;
    private ITriColorLED []leds;
    private ITriColorLED led;
    RadiogramConnection con;
    Radiogram outgram;
    public  boolean synch= false;
    private Thread runner ;
    private int numofMesurments =0;

    public SendThread(RadiogramConnection sCon) {
        con=sCon;
        demo = EDemoBoard.getInstance();
        accel = demo.getAccelerometer();
        light = demo.getLightSensor();
        temp = demo.getADCTemperature();
        leds = demo.getLEDs();
        led = leds[4];
        led.setColor(LEDColor.GREEN);
        sw = demo.getSwitches();
        if(runner == null){
            runner = new Thread(this);
            runner.start();
        }
    }

     synchronized  public void run(){
       try{
```

```
            String mesur = "";
             outgram = (Radiogram)con.newDatagram(con.getMaximumLength());
        while(!synch){
            try{
                led.setOn();
                outgram.reset();
                double accelz = accel.getAccelZ();
                double tem = temp.getCelsius();
                double ligh = light.getValue();
                numofMesurments++;
                mesur =System.getProperty("IEEE_ADDRESS")+"
"+numofMesurments+" "+accelz+" "+
                        tem+" "+ligh;
                outgram.writeUTF(mesur);
                con.send(outgram);
               runner.sleep(1000);
                led.setOff();
                runner.sleep(24*1000);
        }catch(IOException e){
            System.out.println(e.getMessage());
            return;
        }catch(NullPointerException e){
            System.out.println("provlima sto yield "+e.getMessage());
        }catch(Exception e){
            System.out.println(e.getMessage());
        }
       }
             con.close();
    }catch(IOException e){
        System.out.println("To Leaf exei provlima stin unicast sindesi
"+e.getMessage());
    }
    }
}
```

Το thread που δρομολογεί τα εισερχόμενα πακέτα είναι το receiveSendThread.

```
package org.sunspotworld;
import com.sun.spot.sensorboard.peripheral.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.sensorboard.EDemoBoard;
import java.io.IOException;
import javax.microedition.io.Connector;
/**
 *
 * @author user1
 */
public class receiveSendThread implements Runnable {

    private Thread runner ;
    private RadiogramConnection receiveConnection = null;
    private RadiogramConnection sendConnection = null;
    private EDemoBoard demo ;
    private ITriColorLED []leds;
    private ITriColorLED led;
```

```
    public receiveSendThread( RadiogramConnection sCon) {
        sendConnection = sCon;
        demo=EDemoBoard.getInstance();
        leds = demo.getLEDs();
        led = leds[4];
        led.setColor(LEDColor.BLUE);
        if(runner == null){
            runner = new Thread(this);
            runner.start();
        }
    }
    synchronized public void run(){
        try{
            receiveConnection =(RadiogramConnection)
Connector.open("radiogram://:38");
            Radiogram ingram = (Radiogram)
receiveConnection.newDatagram(receiveConnection.getMaximumLength());
            Radiogram outgram = (Radiogram)
sendConnection.newDatagram(sendConnection.getMaximumLength());

            while(true){
                receiveConnection.receive(ingram);
                led.setOn();
                String in = ingram.readUTF();
                outgram.reset();
                outgram.writeUTF(in);
                sendConnection.send(outgram);
                led.setOff();
            }
        }catch(IOException e){
            System.out.println(e.getMessage());
        }

    }
}
```

Mesh-main (&2)

```
package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.io.j2me.radio.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.Utils;

import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
 * The startApp method of this class is called by the VM to start the
 * application.
 *
 * The manifest specifies this class as MIDlet-N, which means it will not
```

```
 * be automatically selected for execution.
 *
 * @author user1

 */
public class receiveSend extends MIDlet {

    protected void startApp() throws MIDletStateChangeException {
       try{
           RadiogramConnection  sendConnection = (RadiogramConnection)
Connector.open("radiogram://broadcast:67");
            new receiveSendThread(sendConnection);
            new SendThread(sendConnection);
    }catch(Exception e ){
        e.printStackTrace();
    }
    }

    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }

    /**
     * Called if the MIDlet is terminated by the system.
     * I.e. if startApp throws any exception other than
MIDletStateChangeException,
     * if the isolate running the MIDlet is killed with Isolate.exit(), or
     * if VM.stopVM() is called.
     *
     * It is not called if MIDlet.notifyDestroyed() was called.
     *
     * @param unconditional If true when this method is called, the MIDlet
must
     *    cleanup and release all resources. If false the MIDlet may throw
     *    MIDletStateChangeException  to indicate it does not want to be
destroyed
     *    at this time.
     */
    protected void destroyApp(boolean unconditional) throws
MIDletStateChangeException {
    }
}
```

Τα threads που καλούνται είναι τα επόμενα.

```
package org.sunspotworld;
import com.sun.spot.sensorboard.peripheral.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.sensorboard.EDemoBoard;
import java.io.IOException;
import javax.microedition.io.Connector;
/**
 *
 * @author user1
 */
public class receiveSendThread implements Runnable {
```

```
    private Thread runner ;
    private RadiogramConnection receiveConnection = null;
    private RadiogramConnection sendConnection = null;
    private EDemoBoard demo ;
    private ITriColorLED []leds;
    private ITriColorLED led;

    public receiveSendThread( RadiogramConnection sCon) {
        sendConnection = sCon;
        demo=EDemoBoard.getInstance();
        leds = demo.getLEDs();
        led = leds[4];
        led.setColor(LEDColor.BLUE);
        if(runner == null){
            runner = new Thread(this);
            runner.start();
        }
    }
    synchronized public void run(){
        try{
            receiveConnection =(RadiogramConnection)
Connector.open("radiogram://:37");
            Radiogram ingram = (Radiogram)
receiveConnection.newDatagram(receiveConnection.getMaximumLength());
            Radiogram outgram = (Radiogram)
sendConnection.newDatagram(sendConnection.getMaximumLength());

            while(true){
                receiveConnection.receive(ingram);
                led.setOn();
                String in = ingram.readUTF();
                outgram.reset();
                outgram.writeUTF(in);
                sendConnection.send(outgram);
                led.setOff();
            }
        }catch(IOException e){
            System.out.println(e.getMessage());
        }

    }
}

public class SendThread implements Runnable {
private EDemoBoard demo ;
    private IAccelerometer3D accel;
    private ILightSensor  light;
    private ITemperatureInput temp;
    private ISwitch [] sw ;
    private ITriColorLED []leds;
    private ITriColorLED led;
    RadiogramConnection con;
    Radiogram outgram;
    public  boolean synch= false;
    private Thread runner ;
    private int numofMesurments =0;
```

```
    public SendThread(RadiogramConnection sCon) {
        con=sCon;
        demo = EDemoBoard.getInstance();
        accel = demo.getAccelerometer();
        light = demo.getLightSensor();
        temp = demo.getADCTemperature();
        leds = demo.getLEDs();
        led = leds[4];
        led.setColor(LEDColor.GREEN);
        if(runner == null){
            runner = new Thread(this);
            runner.start();
        }
    }

     synchronized  public void run(){
       try{
            String mesur = "";
             outgram = (Radiogram)con.newDatagram(con.getMaximumLength());
        while(!synch){
            try{
                led.setOn();
                outgram.reset();
                double accelz = accel.getAccelZ();
                double tem = temp.getCelsius();
                double ligh = light.getValue();
                numofMesurments++;
                mesur =System.getProperty("IEEE_ADDRESS")+"
"+numofMesurments+" "+accelz+" "+
                        tem+" "+ligh;
                outgram.writeUTF(mesur);
                con.send(outgram);
               runner.sleep(1000);
                led.setOff();
                runner.sleep(24*1000);
        }catch(IOException e){
            System.out.println(e.getMessage());
            return;
        }catch(NullPointerException e){
            System.out.println("provlima sto yield "+e.getMessage());
        }catch(Exception e){
            System.out.println(e.getMessage());
        }
       }
             con.close();
    }catch(IOException e){
        System.out.println("To Leaf exei provlima stin unicast sindesi
"+e.getMessage());
    }
    }
}
```

Mesh-inter

```
package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.io.j2me.radio.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.Utils;

import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
 * The startApp method of this class is called by the VM to start the
 * application.
 *
 * The manifest specifies this class as MIDlet-N, which means it will not
 * be automatically selected for execution.
 *
 * @author user1

 */
public class receiveSend extends MIDlet {

    protected void startApp() throws MIDletStateChangeException {
    new receiveSendThread();
    }

    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }

    /**
     * Called if the MIDlet is terminated by the system.
     * I.e. if startApp throws any exception other than
MIDletStateChangeException,
     * if the isolate running the MIDlet is killed with Isolate.exit(), or
     * if VM.stopVM() is called.
     *
     * It is not called if MIDlet.notifyDestroyed() was called.
     *
     * @param unconditional If true when this method is called, the MIDlet
must
     *    cleanup and release all resources. If false the MIDlet may throw
     *    MIDletStateChangeException  to indicate it does not want to be
destroyed
     *    at this time.
     */
    protected void destroyApp(boolean unconditional) throws
MIDletStateChangeException {
    }
}
```

Το thread είναι

```
package org.sunspotworld;
import com.sun.spot.sensorboard.peripheral.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.sensorboard.EDemoBoard;
import java.io.IOException;
import javax.microedition.io.Connector;
import java.util.Date;
/**
 *
 * @author user1
 */
public class receiveSendThread implements Runnable {

    private Thread runner ;
    private RadiogramConnection receiveConnection = null;
    private RadiogramConnection sendConnection = null;
    private EDemoBoard demo ;
    private ITriColorLED []leds;
    private ITriColorLED led;

    public receiveSendThread() {
        demo=EDemoBoard.getInstance();
        leds = demo.getLEDs();
        led = leds[4];
        led.setColor(LEDColor.BLUE);
        if(runner == null){
            runner = new Thread(this);
            runner.start();
        }
    }
    synchronized public void run(){
         while(true){
        try{
              int port =37;
             if(random()) {
                 port=38;
             }
            sendConnection = (RadiogramConnection)
Connector.open("radiogram://broadcast:"+port);
            receiveConnection =(RadiogramConnection)
Connector.open("radiogram://:39");
            Radiogram ingram = (Radiogram)
receiveConnection.newDatagram(receiveConnection.getMaximumLength());
            Radiogram outgram = (Radiogram)
sendConnection.newDatagram(sendConnection.getMaximumLength());
            receiveConnection.receive(ingram);
            led.setOn();
            String in = ingram.readUTF();
            outgram.reset();
            outgram.writeUTF(in);
            sendConnection.send(outgram);
            led.setOff();
            sendConnection.close();
            receiveConnection.close();
```

```
        }catch(IOException e){
            System.out.println(e.getMessage());
        }
      }
    }
public boolean random(){
        Date date = new Date();
        long time = date.getTime();
        long choise = (long) time % 10;
        if(choise>4) return true;
        else
        return false;
    }
}
```

Leaf-Spot

Κώδικας της εφαρμογής Leaf-Spot είναι ο επόμενος

```
package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.io.j2me.radio.*;
import com.sun.spot.io.j2me.radiogram.*;


import com.sun.spot.util.BootloaderListener;
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
 * The startApp method of this class is called by the VM to start the
 * application.
 *
 * The manifest specifies this class as MIDlet-N, which means it will not
 * be automatically selected for execution.
 *
 * @author user1

 */
public class leaf_Spot extends MIDlet {

    protected void startApp() throws MIDletStateChangeException {
        System.out.println("Start app...");
        new BootloaderListener().start();
        Talk_Broadcast tbrc = new Talk_Broadcast();
        Hear_Broadcast hbrc = new Hear_Broadcast();
        while(hbrc.address.equals(""));
        tbrc.synch=true;
        Talk_Unicast tunc = new Talk_Unicast(hbrc.address);
    }
```

```
    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }

    /**
     * Called if the MIDlet is terminated by the system.
     *    I.e.    if    startApp    throws    any    exception    other    than
MIDletStateChangeException,
     * if the isolate running the MIDlet is killed with Isolate.exit(), or
     * if VM.stopVM() is called.
     *
     * It is not called if MIDlet.notifyDestroyed() was called.
     *
     * @param unconditional If true when this method is called, the MIDlet
must
     *    cleanup and release all resources. If false the MIDlet may throw
     *     MIDletStateChangeException  to indicate it does not want to be
destroyed
     *    at this time.
     */
    protected     void     destroyApp(boolean     unconditional)     throws
MIDletStateChangeException {
    }
}
```

Ακολουθεί το νήμα Talk_Broadcast

```
package org.sunspotworld;

import java.io.IOException;
import javax.microedition.io.*;

public class Talk_Broadcast implements  Runnable {

    public boolean synch = false;
    private Thread runner;

    public Talk_Broadcast() {
        if(runner == null){
            runner = new Thread(this);
            runner.start();
        }

    }
    synchronized public void run(){
        DatagramConnection dgConn = null;
        Datagram dg = null;
        try{
            dgConn                  =                  (DatagramConnection)
Connector.open("radiogram://broadcast:37");
            dg = dgConn.newDatagram(dgConn.getNominalLength());
        }catch(IOException e){
            System.out.println("  Spot  Leaf  Not  OPEN  Send  Connection  "+
e.getMessage());
            return;
        }
```

```
        while(!synch){
            try{
                dg.reset();
                dg.writeUTF("HI");
                dgConn.send(dg);
                runner.sleep(500);
            }catch(Exception e){
                System.out.println("Leaf Spot Can send "+ e.getMessage());
            }

        }
        try{
        dgConn.close();
        }catch(IOException e){
            System.out.println(e.getMessage());
        }

    }
}
```

Το νήμα Hear_Broadcast

```
package org.sunspotworld;

import com.sun.spot.io.j2me.radiogram.*;
import java.io.IOException;
import javax.microedition.io.Connector;


public class Hear_Broadcast implements Runnable{

    public String  address="";
    private Thread  runner;
    public boolean findAddress = false;



    Hear_Broadcast(){
            if(runner == null){
                runner = new Thread(this);
                runner.start();
            }
        }
   synchronized public void run(){
        String  iden = "";
        RadiogramConnection dgConn = null;
        Radiogram dg = null;

        try{
            dgConn                  =                  (RadiogramConnection)
Connector.open("radiogram://:37");
            dg = (Radiogram)dgConn.newDatagram(dgConn.getMaximumLength());
        }catch(IOException e){
            System.out.println("Leaf    Spot    Not    receiver    Connection
"+e.getMessage());
            return;
```

```
        }

        while(address.equals("")){
            try{
                dg.reset();
                dgConn.receive(dg);
                iden = dg.readUTF();
                if(iden.equals("isMajor")){
                     address=dg.getAddress();
                }

            }catch(IOException e){
                System.out.println("Leaf      Spot      Nothing      Receive
"+e.getMessage());
            }
        }
        try{
            dgConn.close();
        }catch(IOException e){
            System.out.println(e.getMessage());
        }
    }
}
```

Το νήμα Talk_Unicast

```
package org.sunspotworld;


import com.sun.spot.util.IEEEAddress;
import com.sun.spot.io.j2me.radiogram.*;

import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.peripheral.*;
import java.io.IOException;
import javax.microedition.io.*;

public class Talk_Unicast implements Runnable ,ISwitchListener {

    private EDemoBoard demo ;
    private IAccelerometer3D accel;
    private ILightSensor  light;
    private ITemperatureInput temp;
    private ISwitch [] sw ;
    private ITriColorLED []leds;
    private ITriColorLED led;
    RadiogramConnection con;
    Radiogram outgram;
    public  boolean synch= false;
    private String address = "";
    private Thread runner ;

    public Talk_Unicast(String address) {
        this.address = address;
        demo = EDemoBoard.getInstance();
        accel = demo.getAccelerometer();
```

```
        light = demo.getLightSensor();
        temp = demo.getADCTemperature();
        leds = demo.getLEDs();
        led = leds[4];
        led.setColor(LEDColor.GREEN);
        sw = demo.getSwitches();
        sw[0].addISwitchListener(this);
        if(runner == null){
            runner = new Thread(this);
            runner.start();
        }
    }

     synchronized  public void run(){
       try{
            String mesur = "";
             con                                        =(RadiogramConnection)
Connector.open("radiogram://"+address+":123");
             outgram = (Radiogram)con.newDatagram(con.getMaximumLength());
        while(!synch){
            try{
                led.setOn();
                outgram.reset();
                double accelz = accel.getAccelZ();
                double tem = temp.getCelsius();
                double ligh = light.getValue();
                mesur
=System.getProperty("IEEE_ADDRESS")+"\n"+"acceleration                    =
"+accelz+"\n"+"temperature = "+
                        tem+"\n"+"light = "+ligh+"\n";
                outgram.writeUTF(mesur);
                con.send(outgram);
               runner.sleep(1000);
                led.setOff();
                runner.sleep(24*1000);
        }catch(IOException e){
            System.out.println(e.getMessage());
            return;
        }catch(NullPointerException e){
            System.out.println("provlima sto yield "+e.getMessage());
        }catch(Exception e){
            System.out.println(e.getMessage());
        }
       }
             con.close();
    }catch(IOException e){
        System.out.println("To  Leaf  exei  provlima  stin  unicast  sindesi
"+e.getMessage());
    }
    }

    public void switchReleased(ISwitch sw){
        if(sw==this.sw[0]){
            synch=true;
            try{
                outgram.reset();
                outgram.writeUTF("telos ");
```

```
                con.send(outgram);
                con.close();
            }catch(Exception e){
                System.out.println("Provlima  sto  leaf  oten  xrisimopiw  to
switch"
                        +e.getMessage());
            }
        }
    }
    public void switchPressed(ISwitch sw){

    }
}
```

Main-Spot

Ο κώδικας τη εφαρμογής είναι ο επόμενος.

```
package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.io.j2me.radio.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.Utils;

import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
 * The startApp method of this class is called by the VM to start the
 * application.
 *
 * The manifest specifies this class as MIDlet-N, which means it will not
 * be automatically selected for execution.
 *
 * @author user1

 */
public class Main_Spot extends MIDlet {

    protected void startApp() throws MIDletStateChangeException {
       System.out.println("Starting app...");

        Talk_Broadcast tbrc = new Talk_Broadcast();
        Hear_Broadcast hbrc = new Hear_Broadcast();
        //while(hbrc.s2.empty());
        //while(!hbrc.s2.empty()){
          //  System.out.println("To main spot exei tin add = " +
(String)hbrc.s2.pop());
        //}
    }
```

```
    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }

    /**
     * Called if the MIDlet is terminated by the system.
     * I.e. if startApp throws any exception other than
MIDletStateChangeException,
     * if the isolate running the MIDlet is killed with Isolate.exit(), or
     * if VM.stopVM() is called.
     *
     * It is not called if MIDlet.notifyDestroyed() was called.
     *
     * @param unconditional If true when this method is called, the MIDlet
must
     *    cleanup and release all resources. If false the MIDlet may throw
     *    MIDletStateChangeException  to indicate it does not want to be
destroyed
     *    at this time.
     */
    protected void destroyApp(boolean unconditional) throws
MIDletStateChangeException {
    }
}
```

το νήνα Talk_Broadcast

```
package org.sunspotworld;


import com.sun.spot.util.Utils;
import java.io.IOException;
import javax.microedition.io.*;

public class Talk_Broadcast implements  Runnable {

    private Thread runner;

    public Talk_Broadcast() {
        if(runner == null){
            runner = new Thread(this);
            runner.start();
        }

    }
    synchronized public void run(){
        DatagramConnection dgConn = null;
        Datagram dg = null;
        try{
            dgConn = (DatagramConnection)
Connector.open("radiogram://broadcast:37");
            dg = dgConn.newDatagram(dgConn.getNominalLength());
        }catch(IOException e){
            System.out.println(" Spot Main Not OPEN Send Connection "+
e.getMessage());
            return;
```

```
        }
        while(true){
            try{
                dg.reset();
                dg.writeUTF("isMajor");
                dgConn.send(dg);
            runner.sleep(1000);
            }catch(Exception e){
                System.out.println("Main Spot Can send "+ e.getMessage());
            }
        }
    }


}
```

το νήμα Hear_Broadcast

```
package org.sunspotworld;

import com.sun.spot.io.j2me.radiogram.RadiogramConnection;

import java.io.IOException;
import java.util.Stack;
import javax.microedition.io.Connector;
import javax.microedition.io.Datagram;
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.peripheral.*;

public class Hear_Broadcast implements Runnable , ISwitchListener{

   private Stack s1 ;
   private ISwitch [] sw ;
   private Thread  runner;
   private EDemoBoard demo ;

    Hear_Broadcast(){
            s1 = new Stack();
            demo = EDemoBoard.getInstance();
            sw = demo.getSwitches();
            sw[0].addISwitchListener(this);
            if(runner == null){
                runner = new Thread(this);
                runner.start();
            }
        }


      synchronized public void run(){
        String  iden = "";
        RadiogramConnection dgConn = null;
        Datagram dg = null;

        try{
            dgConn = (RadiogramConnection)
Connector.open("radiogram://:37");
```

```
            dg = dgConn.newDatagram(dgConn.getMaximumLength());
        }catch(IOException e){
            System.out.println("Main Spot Not receiver Connection
"+e.getMessage());
            return;
        }
        while(true){
            try{
                dg.reset();
                dgConn.receive(dg);
                iden = dg.readUTF();
                if((iden.equals("HI"))&& (s1.search(dg.getAddress())==-1)){
                s1.push(dg.getAddress());
                //System.out.println("To main spot exei tin add = " +
dg.getAddress());
                Talk_Unicast_M tunc = new Talk_Unicast_M(dg.getAddress());
                }
            }catch(IOException e){
                System.out.println("Main Spot Nothing Receive
"+e.getMessage());
            }
        }

    }
public void switchReleased(ISwitch sw){
    if(sw == this.sw[0]){
        s1.removeAllElements();
    }
}
public void switchPressed(ISwitch sw){

    }
}
```

το νήμα Talk_Unicast_M

```
package org.sunspotworld;
import com.sun.spot.io.j2me.radiogram.*;
import java.io.IOException;
import javax.microedition.io.*;

public class Talk_Unicast_M implements Runnable {

    private boolean synch= false;
    private String address = "";
    private Thread runner ;
    private RadiogramConnection con = null;
    private RadiogramConnection BHcon = null;
    private static final int HOST_PORT = 67;

    public Talk_Unicast_M(String address) {
        this.address = address;
        if(runner == null){
            runner = new Thread(this);
            runner.start();
        }
```

```
    }

      synchronized public void run(){
       try{
         con =(RadiogramConnection)
Connector.open("radiogram://"+address+":123");
        Radiogram ingram =
(Radiogram)con.newDatagram(con.getMaximumLength());
        while(!synch){
            try{
                con.receive(ingram);
                String in = ingram.readUTF();
                System.out.print(in);
               if(in.equals("telos ")){
                    System.out.println("apo to " +ingram.getAddress());
                    synch=true;
                    con.close();
               }else{
                    //HTTPConn.postUpdate(in);
                    sendDatatoHost(in);
                    //System.out.println(in);
               }
        }catch(IOException e){
            System.out.println(e.getMessage());

        }catch(Exception e){
            System.out.println("provlima sto yield "+e.getMessage());
        }
       }
        //con.close();
    }catch(IOException e){
        System.out.println("To Main exei provlima stin unicast sindesi
"+e.getMessage());
    }
    }
    private synchronized  void  sendDatatoHost(String input)throws
Exception{
        BHcon =
(RadiogramConnection)Connector.open("radiogram://broadcast:"+HOST_PORT);
        Datagram sendtohost = BHcon.newDatagram(BHcon.getMaximumLength());

        sendtohost.reset();
        sendtohost.writeUTF(input);
        BHcon.send(sendtohost);
    }
}
```

intermediate_1-Spot

Ο κώδικας της εφαρμογής

```
package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.io.j2me.radio.*;
```

```
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.Utils;

import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
 * The startApp method of this class is called by the VM to start the
 * application.
 *
 * The manifest specifies this class as MIDlet-N, which means it will not
 * be automatically selected for execution.
 *
 * @author user1

 */
public class inter extends MIDlet {

    protected void startApp() throws MIDletStateChangeException {
        System.out.println("Start app ...");
        Talk_Broadcast_l tb_l = new Talk_Broadcast_l();
        Hear_Broadcast_l hb_l = new Hear_Broadcast_l();
        while(hb_l.address.equals(""));
        tb_l.synch = true;
        System.out.println(hb_l.address);
        Talk_Broadcast_M tb_m = new Talk_Broadcast_M();
        Hear_Broadcast_M hb_m =new Hear_Broadcast_M(hb_l.address);
    }

    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }

    protected void destroyApp(boolean unconditional) throws
MIDletStateChangeException {
    }
}
```

Τα νήματα με την σειρά που καλούνται.

Talk_Broadcast_l

```
package org.sunspotworld;

import com.sun.spot.util.Utils;
import java.io.IOException;
import javax.microedition.io.*;

public class Talk_Broadcast_l implements  Runnable {

    public boolean synch = false;
    private Thread runner;

    public Talk_Broadcast_l() {
        if(runner == null){
```

```
            runner = new Thread(this);
            runner.start();
        }

    }
    synchronized public void run(){
        DatagramConnection dgConn = null;
        Datagram dg = null;
        try{
            dgConn = (DatagramConnection)
Connector.open("radiogram://broadcast:47");
            dg = dgConn.newDatagram(dgConn.getNominalLength());
        }catch(IOException e){
            System.out.println(" Spot Leaf Not OPEN Send Connection "+
e.getMessage());
            return;
        }
        while(!synch){
            try{
                dg.reset();
                dg.writeUTF("HI");
                dgConn.send(dg);
            }catch(IOException e){
                System.out.println("Leaf Spot Can send "+ e.getMessage());
            }
            Utils.sleep(500);
        }
        try{
        dgConn.close();
        }catch(IOException e){
            System.out.println(e.getMessage());
        }

    }
}
```

Hear_Broadcast_l

```
package org.sunspotworld;

import com.sun.spot.io.j2me.radiogram.*;
import java.io.IOException;
import javax.microedition.io.Connector;


public class Hear_Broadcast_l implements Runnable{

    public String  address="";
    private Thread  runner;

    Hear_Broadcast_l(){
            if(runner == null){
                runner = new Thread(this);
                runner.start();
```

```
            }
        }

   synchronized public void run(){
        String  iden = "";
        RadiogramConnection dgConn = null;
        Radiogram dg = null;

        try{
            dgConn = (RadiogramConnection)
Connector.open("radiogram://:47");
            dg = (Radiogram)dgConn.newDatagram(dgConn.getMaximumLength());
        }catch(IOException e){
            System.out.println("Leaf Spot Not receiver Connection
"+e.getMessage());
            return;
        }

        while(address.equals("")){
            try{
                dg.reset();
                dgConn.receive(dg);
                iden = dg.readUTF();
                if(iden.equals("isMajor")){
                     address=dg.getAddress();
                }

            }catch(IOException e){
                System.out.println("Leaf Spot Nothing Receive
"+e.getMessage());
            }
        }
        try{
            dgConn.close();
        }catch(IOException e){
            System.out.println(e.getMessage());
        }
    }

}
```

Talk_Broadcast_M

```
package org.sunspotworld;


import com.sun.spot.util.Utils;
import java.io.IOException;
import javax.microedition.io.*;

public class Talk_Broadcast_M implements  Runnable {

    private Thread runner;

    public Talk_Broadcast_M() {
        if(runner == null){
```

```
            runner = new Thread(this);
            runner.start();
        }

    }
    synchronized public void run(){
        DatagramConnection dgConn = null;
        Datagram dg = null;
        try{
            dgConn = (DatagramConnection)
Connector.open("radiogram://broadcast:37");
            dg = dgConn.newDatagram(dgConn.getNominalLength());
        }catch(IOException e){
            System.out.println(" Spot Main Not OPEN Send Connection "+
e.getMessage());
            return;
        }
        while(true){
            try{
                dg.reset();
                dg.writeUTF("isMajor");
                dgConn.send(dg);
            }catch(IOException e){
                System.out.println("Main Spot Can send "+ e.getMessage());
            }
            Utils.sleep(1000);
        }
    }


}
```

Hear_Broadcast_M

```
package org.sunspotworld;

import com.sun.spot.io.j2me.radiogram.RadiogramConnection;

import java.io.IOException;
import java.util.Stack;
import javax.microedition.io.Connector;
import javax.microedition.io.Datagram;
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.peripheral.*;

public class Hear_Broadcast_M implements Runnable , ISwitchListener{

   private Stack s1 ;
   private ISwitch [] sw ;
   private Thread  runner;
   private EDemoBoard demo ;
   public RadiogramConnection mConn  = null;

    Hear_Broadcast_M(String mAddress){
        try{
            s1 = new Stack();
```

```
            mConn = (RadiogramConnection)
Connector.open("radiogram://"+mAddress+":133");
            demo = EDemoBoard.getInstance();
            sw = demo.getSwitches();
            sw[0].addISwitchListener(this);
            if(runner == null){
                runner = new Thread(this);
                runner.start();
            }
        }catch(IOException e){
            System.out.println(e.getMessage());
        }
        }


      synchronized public void run(){
        String  iden = "";
        RadiogramConnection dgConn = null;
        Datagram dg = null;

        try{
            dgConn = (RadiogramConnection)
Connector.open("radiogram://:37");
            dg = dgConn.newDatagram(dgConn.getMaximumLength());
        }catch(IOException e){
            System.out.println("Main Spot Not receiver Connection
"+e.getMessage());
            return;
        }
        while(true){
            try{
                dg.reset();
                dgConn.receive(dg);
                iden = dg.readUTF();
                if((iden.equals("HI"))&&(s1.search(dg.getAddress())==-
1)&&(s1.size()<=2)){
                    s1.push(dg.getAddress());
                    Talk_Unicast_M tunc = new
Talk_Unicast_M(dg.getAddress(),mConn);
                }
            }catch(IOException e){
                System.out.println("Main Spot Nothing Receive
"+e.getMessage());
            }
        }

    }
public void switchReleased(ISwitch sw){
    if(sw == this.sw[0]){
        s1.removeAllElements();
        try{
        mConn.close();
        }catch(IOException e){
            System.out.println(e.getMessage());
        }
    }
}
```

```
public void switchPressed(ISwitch sw){

    }
}
```

Από το τελευταίο νήμα καλείται το νήμα Talk_Unicast_M

```
package org.sunspotworld;
import com.sun.spot.io.j2me.radiogram.*;
import java.io.IOException;
import javax.microedition.io.*;

public class Talk_Unicast_M implements Runnable {

    private boolean synch= false;
    private String address = "";
    private Thread runner ;
    private RadiogramConnection con = null;
    private RadiogramConnection Mcon = null;



    public Talk_Unicast_M(String address,RadiogramConnection C) {
        this.address = address;
        Mcon = C ;
        if(runner == null){
            runner = new Thread(this);
            runner.start();
        }
    }

      synchronized public void run(){
       try{
         con = (RadiogramConnection)
Connector.open("radiogram://"+address+":123");
         Radiogram ingram =
(Radiogram)con.newDatagram(con.getMaximumLength());
         Radiogram outgram =
(Radiogram)Mcon.newDatagram(Mcon.getMaximumLength());
         while(!synch){
            try{
                con.receive(ingram);
                String in = ingram.readUTF();
                outgram.reset();
                outgram.writeUTF(in);
                Mcon.send(outgram);
                if(in.equals("telos ")){
                    System.out.println("apo to " +ingram.getAddress());
                    synch=true;
                    con.close();
               }
        }catch(IOException e){
            System.out.println(e.getMessage());

        }catch(Exception e){
            System.out.println("provlima sto yield "+e.getMessage());
```

```
        }
       }
    }catch(IOException e){
        System.out.println("To Main exei provlima stin unicast sindesi
"+e.getMessage());
    }
    }
}
```

Index.html 1

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-7"
    pageEncoding="ISO-8859-7"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>Παρουσίαση Αποτελεσμάτων</title>
</head>
<body>
<h2 align="center">Επιλογή Παρουσίασης Αποτελεσμάτων</h2>
<h4 align="center"><a href="othertemporary.jsp" >Μέσος Όρος 5
Τελευταίων Λεπτών</a></h4><br>
<h4 align="center"><a href="otherday.jsp">Μέσος Όρος
Ημέρας</a></h4><br>
<h4 align="center"><a href="5last.jsp">5 Τελευταίες
μετρήσεις</a></h4><br>
</body>
</html>
```

othertemporary.jsp

```
<%@ page import ="java.sql.*,pack.*" %>
<%@ page import = "java.util.Date" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-7"
    pageEncoding="ISO-8859-7"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>Μέσοι Όροι 5 Τελευταίων Λεπτών</title>
</head>
<body>
<h2 align="center">Μέσοι Όροι 5 Τελευταίων Λεπτών</h2>
<table align="center" border="2">
<tr>
<th>Αισθητήρας</th>
<th>Επιτάχυνση</th>
<th>Θερμοκρασία</th>
<th>Φωτεινότητα</th>
</tr>
```

```
<%
String servername = "localhost";
String database = "sensorsbase";
String dbusername = "root";
String dbpassword = "";
String []addreses ;

Class.forName("com.mysql.jdbc.Driver").newInstance();
Connection C = DriverManager.getConnection("jdbc:mysql://"
            + servername + "/" + database, dbusername, dbpassword);
Date date =new Date();
addreses = databasepresentor.getAddresses(C);
for(int i=0 ; i<addreses.length; i++){
      double [] apot =
databasepresentor.meanresults(C,date,addreses[i]);
      if(apot[2]==-1){
      continue;
      }else
%>
<tr>
<th><%= addreses[i]%></th>
<th><%=Double.parseDouble(Math.round(apot[0]*1000)+"")/1000%></th>
<th><%=Double.parseDouble(Math.round(apot[1]*1000)+"")/1000%></th>
<th><%=Double.parseDouble(Math.round(apot[2]*1000)+"")/1000%></th>
</tr>
<%
}
C.close();
%>
</table>
</body>
</html>
```

Ο κώδικας των μεθόδων

```
package pack;
import java.sql.*;
import java.util.Date;
public class databasepresentor {

      public static String[] getAddresses(Connection c)throws
Exception{
            String sql = "SELECT * FROM iDtable";
            PreparedStatement ps = c.prepareStatement(sql);
            ResultSet rs = ps.executeQuery();
            String[] apot = new String[findnumofAddresses(c)];
            int i=0;
            while(rs.next()){
                  apot[i]=rs.getString("address");
                  i++;
            }
      return apot;
      }
      public static int findnumofAddresses(Connection c)throws
Exception{
            String sql = "SELECT * FROM iDtable";
            PreparedStatement ps = c.prepareStatement(sql);
            ResultSet rs = ps.executeQuery();
```

```
            int i=0;
            while(rs.next()){
                  i++;
            }
      return i;
      }
      public static double[] meanresults(Connection c, Date date,
String add)throws Exception {
            try{
            String sql = "SELECT * FROM datatable WHERE address=? AND
time >?";
            PreparedStatement ps = c.prepareStatement(sql);
            ps.setString(1, add);
            ps.setString(2, ""+(date.getTime()-300000));
            ResultSet rs = ps.executeQuery();
            double [] mean =  new double[3];
            mean[0]=0;mean[1]=0;mean[2]=0;
            int num = 0;
            boolean q= false;
            while(rs.next()){

      mean[0]+=Double.parseDouble(rs.getString("acceleration"));

      mean[1]+=Double.parseDouble(rs.getString("temperature"));

      mean[2]+=Double.parseDouble(rs.getString("light"));
                        num++;
                        q=true;
            }
            if(q){
            for(int i =0 ;i<mean.length;i++){
                  mean[i]/=num;
            }
            return mean;
            }else{
                  mean[2]=-1;
                  return mean;
            }
      }catch(Exception e){
            e.printStackTrace();
            throw e;
      }
      }

      public static double[] meanresults3(Connection c, Date date,
String add)throws Exception {
            try{
            String sql = "SELECT * FROM datatable WHERE address = ?
AND time > ?";
            PreparedStatement ps = c.prepareStatement(sql);
            ps.setString(1, add);
            ps.setString(2,""+(date.getTime()-86400000));
            ResultSet rs = ps.executeQuery();
            double [] mean =  new double[3];
            mean[0]=0;mean[1]=0;mean[2]=0;
            int num = 0;
            boolean q= false;
            while(rs.next()){
```

```
	mean[0]+=Double.parseDouble(rs.getString("acceleration"));

	mean[1]+=Double.parseDouble(rs.getString("temperature"));
			mean[2]+=Double.parseDouble(rs.getString("light"));
			num++;
			q=true;
		}
		if(q){
		for(int i =0 ;i<mean.length;i++){
			mean[i]/=num;
		}
		return mean;
		}else{
			mean[2]=-1;
			return mean;
		}
	}catch(Exception e){
		e.printStackTrace();
		throw e;
	}
	}
	public static double[] meanresults4(Connection c, String
add)throws Exception {
		try{
		Date date =new Date();
		String sql = "SELECT * FROM datatable WHERE address = ?
AND time > ?";
		PreparedStatement ps = c.prepareStatement(sql);
		ps.setString(1, add);
		ps.setString(2,""+(date.getTime()-86400000));
		ResultSet rs = ps.executeQuery();
		double [] mean =  new double[3];
		mean[0]=0;mean[1]=0;mean[2]=0;
		int num = 0;
		boolean q= false;
		if(rs.last()){
			do{

	mean[0]+=Double.parseDouble(rs.getString("acceleration"));

	mean[1]+=Double.parseDouble(rs.getString("temperature"));
			mean[2]+=Double.parseDouble(rs.getString("light"));
			num++;
			q=true;
			}while(rs.previous()&& num<5);
		}
		if(q){
		for(int i =0 ;i<mean.length;i++){
			mean[i]/=num;
		}
		return mean;
		}else{
			mean[2]=-1;
			return mean;
		}
	}catch(Exception e){
		e.printStackTrace();
		throw e;
```

```
        }
        }
}
```

otherday.jsp

```
<%@ page import ="java.sql.*,pack.*" %>
<%@ page import = "java.util.Date" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-7"
    pageEncoding="ISO-8859-7"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>Μέσοι Όροι Ημέρας</title>
</head>
<body>
<h2 align="center">Μέσοι Όροι Ημέρας</h2>
<table align="center" border="2">
<tr>
<th>Αισθητήρας</th>
<th>Επιτάχυνση</th>
<th>Θερμοκρασία</th>
<th>Φωτεινότητα</th>
</tr>
<%
String servername = "localhost";
String database = "sensorsbase";
String dbusername = "root";
String dbpassword = "";
String []addreses ;

Class.forName("com.mysql.jdbc.Driver").newInstance();
Connection C = DriverManager.getConnection("jdbc:mysql://"
            + servername + "/" + database, dbusername, dbpassword);
Date date =new Date();
addreses = databasepresentor.getAddresses(C);
for(int i=0 ; i<addreses.length; i++){
      double [] apot =
databasepresentor.meanresults3(C,date,addreses[i]);
      if(apot[2]==-1){
            continue;
      }else
%>
<tr>
<th><%= addreses[i]%></th>
<th><%=Double.parseDouble(Math.round(apot[0]*1000)+"")/1000%></th>
<th><%=Double.parseDouble(Math.round(apot[1]*1000)+"")/1000%></th>
<th><%=Double.parseDouble(Math.round(apot[2]*1000)+"")/1000%></th>
</tr>
<%
}
C.close();
%>
```

```
</table>
</body>
</html>
```

5last.jsp

```
<%@ page import ="java.sql.*,pack.*" %>
<%@ page import = "java.util.Date" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-7"
    pageEncoding="ISO-8859-7"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>5 Τελευταίες μετρήσεις</title>
</head>
<body>
<h2 align="center">5 Τελευταίες μετρήσεις</h2>
<table align="center" border="2">
<tr>
<th>Αισθητήρας</th>
<th>Επιτάχυνση</th>
<th>Θερμοκρασία</th>
<th>Φωτεινότητα</th>
</tr>
<%
String servername = "localhost";
String database = "sensorsbase";
String dbusername = "root";
String dbpassword = "";
String []addreses ;

Class.forName("com.mysql.jdbc.Driver").newInstance();
Connection C = DriverManager.getConnection("jdbc:mysql://"
            + servername + "/" + database, dbusername, dbpassword);
addreses = databasepresentor.getAddresses(C);
for(int i=0 ; i<addreses.length; i++){
      double [] apot = databasepresentor.meanresults4(C,addreses[i]);
      if(apot[2]==-1){
      continue;
      }else
%>
<tr>
<th><%= addreses[i]%></th>
<th><%=Double.parseDouble(Math.round(apot[0]*1000)+"")/1000%></th>
<th><%=Double.parseDouble(Math.round(apot[1]*1000)+"")/1000%></th>
<th><%=Double.parseDouble(Math.round(apot[2]*1000)+"")/1000%></th>
</tr>
<%
}
C.close();
%>
</table>
</body>
```

```
</html>
```

Index.html 2

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-7"
    pageEncoding="ISO-8859-7"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>Αποτελέσματα</title>
</head>
<body>
<h2 align="center">Ανα Δύο Ώρες Αποτελέσματα Συγκεκριμένου
Αισθιτήρα</h2>
<form method="Post" action="apotelesmata.jsp">
<table border="0" align="center" cellpadding="0" cellspacing="0">
      <tr>
            <td align="right"><strong>Διεύθυνση:</strong></td>
            <td align="center"><input name="address" type="text"
size="20"
                  maxlength="255"></TD>
      </tr>
      <tr>
            <td align="right"><strong>Ημερομηνία:</strong></td>
            <td align="center"><input name="day" type="text"
                  size="20" maxlength="255"></td>
      </tr>
</table>
<table border="0" align="center" cellpadding="0" cellspacing="0">
      <tr>
            <td align="center"><input name="submit" type="submit"
                  value="Enter Data"></td>
            <td align="center"><input type="reset" value="Reset"
name="B2">
            </td>
      </tr>
</table>
</form>
</body>
</html>
```

apotelesmata.jsp

```
<%@ page import ="java.sql.*,pack.*" %>
<%@ page import = "java.util.Date" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-7"
    pageEncoding="ISO-8859-7"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>Αποτελέσματα</title>
</head>
<body>
<h2 align="center" style="azimuth: deg">Αποτελέσματα</h2>
<table align="center" border="2" style="azimuth: deg">
<tr>
<th>Αισθητήρας</th>
<th>Δίωρα</th>
<th>Επιτάχυνση</th>
<th>Θερμοκρασία</th>
<th>Φωτεινότητα</th>
</tr>
<%
String servername = "localhost";
String database = "sensorsbase";
String dbusername = "root";
String dbpassword = "";
String []addreses ;

Class.forName("com.mysql.jdbc.Driver").newInstance();
Connection C = DriverManager.getConnection("jdbc:mysql://"
+ servername + "/" + database, dbusername, dbpassword);
String address =(String)request.getParameter("address");
String day = (String) request.getParameter("day");
String [] dayMonYear = databasamethods.token(day);
Long mera = (Long.parseLong(dayMonYear[0]));
int minas = Integer.parseInt(dayMonYear[1]);
Long etoi =  Long.parseLong(dayMonYear[2])-1970;
Long meres = 10*366 + (etoi-10)*365;
switch(minas){
      case 1:
            break;
      case 2:
            meres+=31;
            break;
      case 3:
            meres+=59;
            break;
      case 4:
            meres+=90;
            break;
      case 5:
            meres+=120;
            break;
      case 6:
            meres+=151;
            break;
      case 7:
            meres+=181;
            break;
      case 8:
            meres+=212;
            break;
      case 9:
            meres+=243;
            break;
```

```
		case 10:
			meres+=273;
			break;
		case 11:
			meres+=304;
			break;
		case 12:
			meres+=334;
			break;
}
meres += mera;
meres--;
Long time = meres*24*3600*1000-2*3600000;
int gap=24;
Long stime ,etime;
for(int i=0;i<gap;i+=2){
stime=time + i*3600000;
etime=stime + 2*3600000;
double[]  apot = databasamethods.apotel(C,address,stime,etime);
if(apot[2]!=-1){
%>
<tr>
<th><%= address%></th>
<th><%=i%>:00 - <%=i+2%>:00</th>
<th><%=Double.parseDouble(Math.round(apot[0]*1000)+"")/1000%></th>
<th><%=Double.parseDouble(Math.round(apot[1]*1000)+"")/1000%></th>
<th><%=Double.parseDouble(Math.round(apot[2]*1000)+"")/1000%></th>
</tr>
<%
}
}
C.close();
%>
</table>
</body>
</html>
```

Τέλος ο κώδικας της μεθόδου που καλείται από την σελίδα apotelesmata.jsp .

```
package pack;
import java.sql.*;
import java.util.*;
public class databasamethods {
	public static double[] apotel (Connection C, String addr,long
starttime,long endtime)
	throws Exception{
		try{
			String sql = "SELECT * FROM datatable WHERE
address=? AND time <? AND time >?";

		PreparedStatement ps = C.prepareStatement(sql);
		ps.setString(1, addr);
		ps.setString(2, ""+endtime);
		ps.setString(3, ""+starttime);
		ResultSet rs = ps.executeQuery();
		double [] mean =  new double[3];
		mean[0]=0;mean[1]=0;mean[2]=0;
		int num = 0;
```

```
            boolean q= false;
            while(rs.next()){

        mean[0]+=Double.parseDouble(rs.getString("acceleration"));

        mean[1]+=Double.parseDouble(rs.getString("temperature"));

        mean[2]+=Double.parseDouble(rs.getString("light"));
                        num++;
                        q=true;
            }
            if(q){
            for(int i =0 ;i<mean.length;i++){
                  mean[i]/=num;
            }
            return mean;
            }else{
                  mean[2]=-1;
                  return mean;
            }
            }catch (Exception e) {
                  e.printStackTrace();
                  throw e;
            }
      }
      public static String[] token(String str) {

            StringTokenizer tk = new StringTokenizer(str,"/");
            String tokens[] = new String[tk.countTokens()];
            for (int i = 0; i < tokens.length; i++) {
                  tokens[i] = tk.nextToken();
            }
            return tokens;
      }
}
```