ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΔΙΔΑΚΤΙΚΗ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ»
ΚΑΤΕΥΘΥΝΣΗ : ΔΙΚΤΥΟΚΕΝΤΡΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

"Μελέτη τεχνολογιών μεσισμικού (middleware) για Ασύρματα Δίκτυα Αισθητήρων. Υλοποίηση μιας αντίστοιχης πλατφόρμας βάσει επιλογής."

ΓΙΑΝΤΣΕΛΙΔΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ (ΜΕ0667)

Επιβλέπων: Βέρα-Αλεξάνδρα Σταυρουλάκη, Επίκουρη Καθηγήτρια

ΠΕΙΡΑΙΑΣ 09/2011

ΠΕΡΙΛΗΨΗ

Αυτή η διπλωματική διατριβή εκπονήθηκε στα πλαίσια του μεταπτυχιακού προγράμματος σπουδών του Τμήματος Διδακτικής της Τεχνολογίας και Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς στην κατεύθυνση Δικτυοκεντρικά Συστήματα.

Στα πλαίσια αυτής γίνεται μια μελέτη στα Ασύρματα Δίκτυα Αισθητήρων (Wireless Sensor Networks, WSN) που αποτελούν, τα τελευταία χρόνια, μία περιοχή με μεγάλη ερευνητική δραστηριότητα.

Οι ιδιαιτερότητες αυτών των δικτύων καθιστούν τη μελέτη τους ξεχωριστή από τις ήδη υπάρχουσες τεχνολογίες ασύρματων δικτύων (όπως Ad-Hoc ή IEEE 802.11). Οι ιδιαιτερότητες των δικτύων αυτών δημιουργούν νέα πεδία εφαρμογής (ιατρικά, επιστημονικά, επιχειρηματικά κ.α.).

Η εργασία επεκτείνεται και αναλύει τα διάφορα πρωτόκολλα επικοινωνίας και δρομολόγησης καθώς και τις τεχνολογίες ενδιάμεσου λογισμικού που χρησιμοποιούνται, ή μπορούν να χρησιμοποιηθούν, προκειμένου τα δίκτυα αυτά να λειτουργούν απρόσκοπτα με τη μικρότερη κατανάλωση ενέργειας.

Κατόπιν παρουσιάζεται η πλατφόρμα του OSGi, περιγράφοντας κάποιες λειτουργίες του και τονίζοντας συγκεκριμένα χαρακτηριστικά και πλεονεκτήματά του. Όλα αυτά γίνονται εμφανή και πιο κατανοητά με μια σχετική υλοποίηση, σε OSGi, μιας βασικής μορφής middleware για την ανάκτηση και παρουσίαση δεδομένων από ένα ή περισσότερα ασύρματα δίκτυα αισθητήρων.

Λέξεις-κλειδιά: Ασύρματα Δίκτυα Αισθητήρων

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στην επιβλέπουσα της εργασίας, Επίκουρη Καθηγήτρια κα. Βέρα Σταυρουλάκη, η οποία μου εμπιστεύτηκε την εκπόνηση της διπλωματικής εργασίας. Την ευχαριστώ επίσης για την πολύ σημαντική και υποδειγματική καθοδήγησή της, τις εύστοχες παρατηρήσεις της, το χρόνο που αφιέρωσε καθώς και την υπομονή της. Η συμβολή της υπήρξε καθοριστική και χωρίς τη βοήθειά της, θα ήταν αδύνατη η ολοκλήρωση της εργασίας.

Επίσης θα ήθελα να ευχαριστήσω τη σύζυγο μου Ελευθερία για την στήριξη που μου προσέφερε, όπως και τους γονείς μου για τις πολύτιμες συμβουλές τους.

Αφιερώνεται στην νεογέννητη κόρη μου

Κωνσταντίνος Γιαντσελίδης

ΠΕΡΙΕΧΟΜΕΝΑ

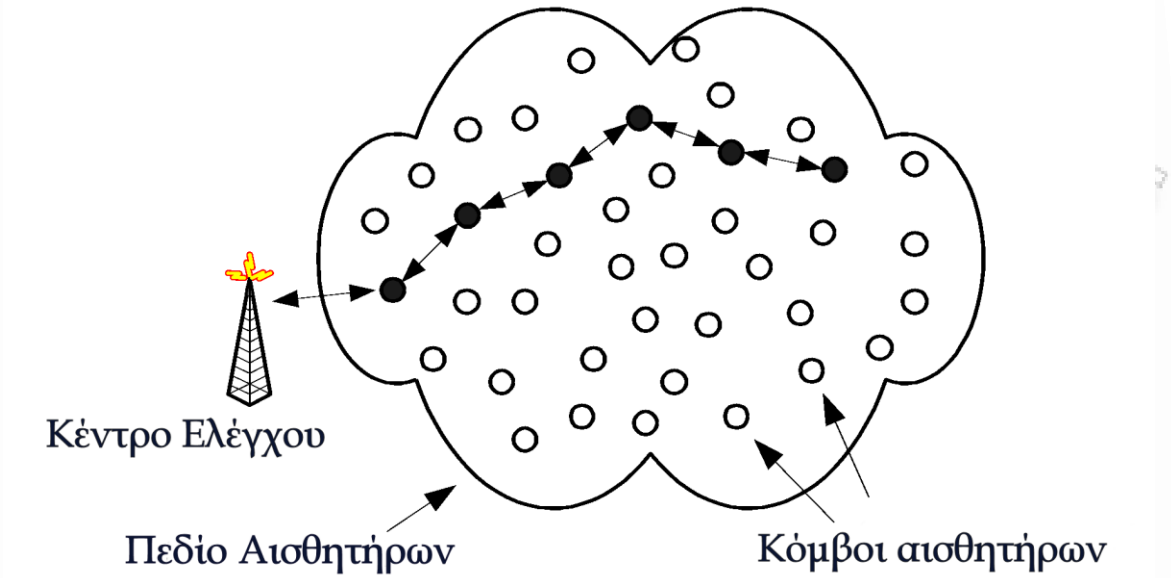
ΛΙΣΤΑ ΕΙΚΟΝΩΝ

1

ΕΙΣΑΓΩΓΗ

Η πρόσφατη πρόοδος στην τεχνολογία των μικρό-ηλεκτρομηχανικών συστημάτων (ΜΗΜΣ), στην ασύρματη επικοινωνία και στην στα ψηφιακά ηλεκτρονικά έχει δώσει την δυνατότητα για την ανάπτυξη κόμβων αισθητήρων χαμηλού-κόστους, χαμηλής κατανάλωσης ενέργειας και πολλών λειτουργιών, οι οποίοι είναι μικροί σε μέγεθος και επικοινωνούν ,χωρίς ανθρώπινη παρέμβαση ή επιτήρηση, μεταξύ τους σε μικρές αποστάσεις. Αυτοί οι μικροσκοπικοί κόμβοι αισθητήρων, που αποτελούνται από στοιχεία αίσθησης, επεξεργασίας δεδομένων και επικοινωνιών, οδηγούν στην ιδέα δικτύων αισθητήρων που βασίζονται στην συνεργατική λειτουργία ενός μεγάλου συνόλου κόμβων. Τα δίκτυα αισθητήρων αντιπροσωπεύουν μια σημαντική εξέλιξη έναντι των παραδοσιακών αισθητήρων, οι οποίοι εγκαθίστανται με τους ακόλουθους δύο τρόπους:

- Εγκατάσταση αισθητήρων μακριά από το πραγματικό φαινόμενο. Με αυτήν την προσέγγιση απαιτούνται μεγάλοι αισθητήρες που χρησιμοποιούν πολύπλοκες τεχνικές για να διακρίνουν τους στόχους από τον θόρυβο του περιβάλλοντος.
- Εγκατάσταση αισθητήρων που εκτελούν μόνο αισθητήρια εργασία. Η θέση και η επικοινωνιακή τοπολογία σχεδιάζονται προσεκτικά. Αυτοί εκπέμπουν μια σειρά του υπό παρακολούθηση φαινομένου στους κεντρικού κόμβους όπου εκτελούνται οι υπολογισμοί και συγχωνεύονται τα δεδομένα.



Εικόνα 1: Απλό μοντέλο δικτύου αισθητήρων. [1]

Ένα δίκτυο αισθητήρων αποτελείται από ένα μεγάλο αριθμό κόμβων αισθητήρων, οι οποίοι αναπτύσσονται πυκνά, είτε μέσα στο φαινόμενο είτε πολύ κοντά σε αυτό (Εικόνα 1). Η θέση των κόμβων αισθητήρων δεν είναι ανάγκη να προσχεδιαστεί ή να προαποφασιστεί. Αυτό επιτρέπει την τυχαία εξάπλωσή σε μη προσβάσιμα εδάφη ή σε επιχειρήσεις για την αντιμετώπιση καταστροφών. Από την άλλη πλευρά, αυτό σημαίνει ότι τα πρωτοκολλά και οι αλγόριθμοι των δικτύων αισθητήρων πρέπει να διαθέτουν αυτό-οργανωτικές δυνατότητες. Ένα άλλο μοναδικό χαρακτηριστικό των δικτύων αισθητήρων είναι η συνεργατική λειτουργία των κόμβων αισθητήρων. Οι αισθητήριοι κόμβοι εξοπλίζονται με έναν on-board επεξεργαστή. Αντί να στέλνουν ακατέργαστα δεδομένα στους κόμβους που είναι υπεύθυνοι για την συγχώνευση των δεδομένων, οι αισθητήριοι κόμβοι χρησιμοποιούν τις δυνατότητες επεξεργασίας που διαθέτουν προκείμενου να εκτελέσουν τοπικά απλούς υπολογισμούς και να εκπέμψουν μόνο τα επεξεργασμένα δεδομένα.

Το παραπάνω χαρακτηριστικό εξασφαλίζει ένα μεγάλο πλήθος εφαρμογών για δίκτυα αισθητήρων. Μερικές από τις περιοχές εφαρμογής είναι η υγεία, ο στρατός και η ασφάλεια. Για παράδειγμα, μια στρατιωτική εφαρμογή των δικτύων αισθητήρων είναι η εμφύτευση τους στα συστήματα διαταγών, ελέγχου, επικοινωνιών, πληροφορικής, πληροφοριών, επιτήρησης, αναγνωρίσεων και σκόπευσης, εκμεταλλευόμενοι τις ιδιότητες τους όπως η ταχεία εγκατάσταση, η

αυτό-οργάνωση και η αντοχή σε λάθη. Οι αισθητήριοι κόμβοι μπορούν επίσης να χρησιμοποιηθούν για να εντοπίσουν ξένα χημικά στοιχεία στον αέρα και το νερό. Μπορούν να βοηθήσουν για να ανακαλύψουν τον τύπο, την συγκέντρωση και την θέση μολυσματικών ουσιών.

Προκειμένου να υλοποιηθούν οι παραπάνω αλλά και άλλες εφαρμογές των δικτύων αισθητήρων απαιτούνται τεχνικές ad-hoc δικτύωσης (καθόσον έχουν ομοιότητες με τα δίκτυα αισθητήρων). Παρόλο που αρκετοί αλγόριθμοι και πρωτοκολλά έχουν προταθεί για τα παραδοσιακά ad-hoc ασύρματα δίκτυα, δυστυχώς δεν είναι δυνατόν να χρησιμοποιηθούν προκειμένου να καλύψουν τα μοναδικά χαρακτηριστικά και τις απαιτήσεις των εφαρμογών των δικτύων αισθητήρων. Παρακάτω αναφέρονται περιληπτικά οι διαφορές μεταξύ των δύο αυτών δικτύων που δικαιολογεί το παραπάνω πρόβλημα:

- Ο αριθμός των κόμβων σε ένα δίκτυο αισθητήρων μπορεί να είναι πολλές φορές πιο μεγάλος από ότι σε ένα ad hoc δίκτυο.
- Η χωρική πυκνότητα των δικτύων αισθητήρων είναι συχνά μεγάλη.
- Οι αισθητήριοι κόμβοι είναι εύκολο να καταστραφούν.
- Η τοπολογία ενός δικτύου αισθητήρων αλλάζει πολύ συχνά.
- Οι αισθητήριοι κόμβοι χρησιμοποιούν κυρίως επικοινωνία broadcast ενώ τα περισσότερα δίκτυα ad hoc βασίζονται στην επικοινωνία σημείου προς σημείο.
- Οι αισθητήριοι κόμβοι διακρίνονται για τους σημαντικούς περιορισμούς που έχουν, από κατασκευής, στους τομείς της ενέργειας, της υπολογιστικής ισχύος και της μνήμης.
- Οι αισθητήριοι κόμβοι συνήθως δεν έχουν κάποιο παγκόσμιο αναγνωριστικό (ID), εξαιτίας του μεγάλου μεγέθους της επικεφαλίδας που απαιτεί μια τέτοια ιδιότητα, καθώς και του μεγάλου αριθμού των κόμβων.

Επειδή ένας μεγάλος αριθμός κόμβων αισθητήρων αναπτύσσεται με πυκνή διάταξη, οι γειτονικοί κόμβοι μπορεί να βρίσκονται πολύ κοντά ο ένας στον άλλο. Έτσι η επικοινωνία μεταξύ πολλαπλών διαδοχικών κόμβων (multi-hop communication) στα δίκτυα αισθητήρων αναμένεται να απαιτεί λιγότερη ενέργεια από ότι η παραδοσιακή επικοινωνία μεταξύ γειτονικών κόμβων (single-hop communication). Η επικοινωνία μεταξύ πολλαπλών διαδοχικών κόμβων (multi-hop) μπορεί να αντιμετωπίσει αποτελεσματικά κάποια από τα προβλήματα διάδοσης του σήματος σε μακρινές αποστάσεις.

Ένας από τα πιο σημαντικούς περιορισμούς στα δίκτυα ασύρματων αισθητήρων είναι η απαίτηση για χαμηλή κατανάλωση ενέργειας. Οι αισθητήριοι κόμβοι έχουν περιορισμένες και συνήθως αναντικατάστατες πηγές ενέργειας. Έτσι ενώ τα παραδοσιακά δίκτυα στοχεύουν να παρέχουν υπηρεσίες υψηλής ποιότητας, τα δίκτυα ασύρματων αισθητήρων έχουν ως πρωταρχικό στόχο την διατήρηση της ενέργειας. Επίσης θα πρέπει να έχουν ένα μηχανισμό που θα δίνει στον χρήστη του δικτύου την επιλογή να παρατείνει την ζωή του δικτύου με αντάλλαγμα την μικρότερη διαμεταγωγή ή την μεγαλύτερη καθυστέρηση στην μετάδοση.

2

ΕΦΑΡΜΟΓΕΣ ΔΙΚΤΥΩΝ ΑΙΣΘΗΤΗΡΩΝ

Τα δίκτυα αισθητήρων μπορούν να αποτελούνται από πολλούς διαφορετικούς τύπους αισθητήρων όπως σεισμικών, μαγνητικών χαμηλού ρυθμού δειγματοληψίας, θερμικών, οπτικών, υπερύθρων, ακουστικών και ραντάρ, οι οποίοι είναι ικανοί να παρακολουθούν μια ευρεία ποικιλομορφία περιβαλλοντολογικών συνθηκών που περιλαμβάνουν τα ακόλουθα:

- Θερμοκρασία
- Υγρασία
- Κίνηση οχημάτων
- Συνθήκες φωτός.
- Πίεση.
- Διάρθρωση εδάφους.
- Επίπεδα θορύβου.
- Την παρουσία ή απουσία προκαθορισμένων ειδών αντικειμένων.
- Επίπεδα μηχανικής πίεσης σε προσκολλημένα αντικείμενα και
- Τα τρέχοντα χαρακτηριστικά όπως ταχύτητα, κατεύθυνση και μέγεθος ενός αντικειμένου.

Οι αισθητήριοι κόμβοι μπορούν να χρησιμοποιηθούν για συνεχή ανίχνευση, ανίχνευση συμβάντων, ανίχνευση ταυτοτήτων γεγονότων, ανίχνευση θέσης και τοπικό έλεγχο μηχανισμών κίνησης. Η ιδέα της μικρό-ανίχνευσης και της ασύρματης σύνδεσης αυτών των κόμβων υπόσχεται πολλές νέες περιοχές εφαρμογών. Οι εφαρμογές των δικτύων αισθητήρων μπορούν να ομαδοποιηθούν σε στρατιωτικές, υγείας, περιβάλλοντος, οικιακές και εμπορικές. Είναι δυνατόν να επεκτείνουμε την ομαδοποίηση με περισσότερες κατηγορίες όπως εξερεύνηση του διαστήματος, χημική επεξεργασία, αντιμετώπιση καταστροφών κ.α.

2.1 Στρατιωτικές Εφαρμογές.

Τα ασύρματα δίκτυα αισθητήρων μπορούν να είναι ένα ενσωματωμένο κομμάτι των στρατιωτικών συστημάτων διαταγών, ελέγχου, επικοινωνιών, υπολογισμού, ευφυΐας, παρακολούθησης, αναγνωρίσεων και στόχευσης. Τα χαρακτηριστικά των δικτύων αισθητήρων, όπως η ταχεία εγκατάσταση, η αυτό-οργάνωση και η αντοχή σε λάθη, τους κατατάσσουν σε ένα πολύ υποσχόμενο αισθητήριο μέσο για τα παραπάνω συστήματα. Καθώς τα δίκτυα αισθητήρων βασίζονται στην πυκνή χωρική εγκατάσταση, η καταστροφή μερικών κόμβων από εχθρικές δυνάμεις δεν επηρεάζει μια στρατιωτική επιχείρηση σε τέτοιο βαθμό όσο η καταστροφή των παραδοσιακών αισθητήρων, κάνοντας την χρήση των δικτύων αισθητήρων ιδανική για τα πεδία των μαχών. Κάποιες από τις στρατιωτικές εφαρμογές των δικτύων αισθητήρων είναι η παρακολούθηση των φιλικών δυνάμεων, του εξοπλισμού και των πυρομαχικών τους, η παρακολούθηση του πεδίου της μάχης, η αναγνώριση των εχθρικών δυνάμεων και του εδάφους, η στόχευση, η αποτίμηση των ζημιών της μάχης, καθώς και η ανίχνευση και αναγνώριση μιας ΡαδιοΒιολογικής Χημικής και Πυρηνικής (ΡΒΧΠ) απειλής.

Παρακολούθηση του εξοπλισμού και των πυρομαχικών των φίλιων δυνάμεων

Οι ηγέτες και οι διοικητές μπορούν χρησιμοποιώντας τα δίκτυα αισθητήρων να παρακολουθούν την κατάσταση των τμημάτων τους καθώς και του εξοπλισμού και των πυρομαχικών τους. Κάθε στρατιώτης, όχημα, εξοπλισμός και κρίσιμο οπλικό σύστημα μπορεί να εξοπλιστεί με αισθητήρες που θα αναφέρουν την κατάστασή του. Αυτές οι αναφορές συγκεντρώνονται σε κεντρικούς κόμβους και προωθούνται προς τους διοικητές των τμημάτων. Τα δεδομένα μπορούν επίσης να προωθηθούν και σε μεγαλύτερα ιεραρχικά κλιμάκια αθροισμένα με δεδομένα από άλλες μονάδες του ίδιου επιπέδου.

Παρακολούθηση του πεδίου της μάχης

Κρίσιμα εδάφη, δρομολόγια προσέγγισης, μονοπάτια και στενωποί μπορούν γρήγορα να καλυφθούν με δίκτυα αισθητήρων και να παρακολουθούνται στενά για εχθρικές δραστηριότητες. Καθώς οι επιχειρήσεις θα εξελίσσονται και θα ετοιμάζονται νέα επιχειρησιακά σχέδια, μπορούν κάθε φορά να εγκαθίστανται νέα δίκτυα αισθητήρων που θα καλύπτουν τις νέες ανάγκες.

Αναγνώριση των εχθρικών δυνάμεων και του εδάφους

Τα δίκτυα αισθητήρων μπορούν να εγκατασταθούν σε κρίσιμα εδάφη και να συγκεντρώνουν έγκαιρα πολύτιμες και λεπτομερείς πληροφορίες για τις εχθρικές δυνάμεις και το έδαφος σε ελάχιστα λεπτά, προτού οι εχθρικές δυνάμεις να μπορέσουν να αναχαιτίσουν τα δίκτυα.

Στόχευση

Τα δίκτυα αισθητήρων μπορούν να εμφυτευτούν σε συστήματα πλοήγησης των έξυπνων πυρομαχικών.

Εκτίμηση των ζημιών μάχης

Πριν ή μετά από κάποια επίθεση δίκτυα αισθητήρων μπορούν να εγκατασταθούν στην περιοχή του στόχου ή των στόχων για να συγκεντρώσουν πληροφορίες προκειμένου να γίνει εκτίμηση των ζημιών.

Ανίχνευση και αναγνώριση ΡΒΧΠ

Στον ΡΒΧΠ πόλεμο, όταν είσαι κοντά στο σημείο μηδέν (σημείο έκρηξης ΡΒΧΠ όπλου) είναι σημαντικό να διαθέτεις ακριβή και έγκαιρη πληροφορία για την ύπαρξη μόλυνσης. Τα δίκτυα αισθητήρων τα οποία εγκαθίστανται στην φίλια περιοχή και χρησιμοποιούνται σαν συστήματα αναγνώρισης και προειδοποίηση ΡΒΧΠ ουσιών, μπορούν να παρέχουν στις φίλιες δυνάμεις κρίσιμο χρόνο για να αντιδράσουν, και να μειώσουν δραστικά τις απώλειες. Τα δίκτυα αισθητήρων μπορούν επίσης να χρησιμοποιηθούν για την αναγνώριση μιας περιοχής που προσβλήθηκε από ΡΒΧΠ επίθεση χωρίς να είναι αναγκαίο να εκτεθεί μια ομάδα ανίχνευσης στην ραδιενέργεια.

2.2 Περιβαλλοντολογικές Εφαρμογές.

Μερικές περιβαλλοντολογικές εφαρμογές των ασύρματων δικτύων αισθητήρων περιλαμβάνουν την καταγραφή των μετακινήσεων πουλιών, μικρών ζώων και εντόμων, την παρακολούθηση περιβαλλοντολογικών συνθηκών που επηρεάζουν τη χλωρίδα και την πανίδα, τους υδροφόρους ορίζοντες, την ανίχνευση

βιολογικών και χημικών στοιχείων, την ανίχνευση πυρκαγιών σε δασικές εκτάσεις, μετεωρολογικές ή γεωφυσικές έρευνες, την ανίχνευση πλημμύρων, τη χαρτογράφηση της βιοποικιλότητας του περιβάλλοντος καθώς και τη μελέτη της μόλυνσης του περιβάλλοντος.

Ανίχνευση πυρκαγιών

Από τη στιγμή που οι κόμβοι του δικτύου μπορούν να διασκορπιστούν τυχαία ή με βάση κάποια στρατηγική και με μεγάλη πυκνότητα μέσα σε κάποιο δάσος, οι αισθητήρες μπορούν να αποκαλύψουν την ακριβή εστία της φωτιάς πριν η διάδοσή της είναι ανεξέλεγκτη. Εκατομμύρια αισθητήρες-κόμβοι μπορούν να χρησιμοποιηθούν μέσω ραδιοσυχνοτήτων / οπτικών συστημάτων. Επίσης, μπορούν να εξοπλιστούν με αποτελεσματικές μεθόδους για την εξοικονόμηση ενέργειας, όπως ηλιακά κελιά, επειδή οι αισθητήρες μπορεί να παραμείνουν χωρίς επιτήρηση για μήνες ή ακόμα και για χρόνια. Οι αισθητήρες θα αλληλεπιδρούν μεταξύ τους προκειμένου να πραγματοποιούν κατανεμημένες λειτουργίες αισθήσεων και προκειμένου να παρακάμψουν εμπόδια, όπως δέντρα και πέτρες που θα εμπόδιζαν την επικοινωνία μεταξύ τους.

Χαρτογράφηση της βιοποικιλότητας του περιβάλλοντος

Η χαρτογράφηση της βιοποικιλότητας του περιβάλλοντος προϋποθέτει ανεπτυγμένες προσεγγίσεις προκειμένου να συνδυάσουν τις πληροφορίες μεταξύ χωρικής και χρονικής κλίμακας. Η πρόοδος της τεχνολογίας στον τομέα της απομακρυσμένης αίσθησης και της αυτοματοποιημένης συλλογής δεδομένων έχει δώσει τη δυνατότητα για μεγαλύτερη χωρική και χρονική ανάλυση, με ένα γεωμετρικά φθίνον κόστος ανά κόμβο. Παράλληλα με αυτή την τεχνολογική πρόοδο, οι αισθητήρες έχουν τη δυνατότητα να συνδέονται με το internet, το οποίο επιτρέπει σε απομακρυσμένους χρήστες να ελέγχουν, να παρακολουθούν και να παρατηρούν τη βιοποικιλότητα του περιβάλλοντος. Παρόλο που οι δορυφορικοί και αεροπορικοί αισθητήρες είναι χρήσιμοι στην παρατήρηση μεγάλης κλίμακας βιοποικιλότητας, π.χ. χωρική πολυπλοκότητα κάποιων κυρίαρχων ειδών φυτών, δεν είναι αρκετά ικανοποιητικοί στην παρατήρηση μικρής κλίμακας βιοποικιλότητας που ουσιαστικά αποτελεί και το βασικό κομμάτι ενός οικοσυστήματος. Αυτό έχει ως αποτέλεσμα να υπάρχει ανάγκη για χρήση ασύρματου δικτύου αισθητήρων στην επιφάνεια του εδάφους για την πιο αποτελεσματική παρατήρηση της βιοποικιλότητας.

Ανίχνευση πλημμύρων

Ένα χαρακτηριστικό παράδειγμα ανίχνευσης πλημμύρων είναι το σύστημα ALERT εφαρμόστηκε στις Ηνωμένες Πολιτείες της Αμερικής. Μερικοί τύποι αισθητήρων που χρησιμοποιήθηκαν στο σύστημα ALERT είναι αισθητήρες βροχής, ύδατος και κλίματος. Αυτοί οι αισθητήρες παρέχουν πληροφορίες σε μία κεντρική βάση δεδομένων με έναν προδιαγεγραμμένο τρόπο. Μερικά ερευνητικά προγράμματα, όπως το COUGAR, εξετάζουν κατανεμημένες προσεγγίσεις αναφορικά με την αλληλεπίδραση με τους αισθητήρες σε ένα πεδίο αισθητήρων προκειμένου να παρέχουν άμεσες αλλά μακροχρόνιες επερωτήσεις.

Προβλέψεις χρήσιμες για τη γεωργία

Μερικά από τα πλεονεκτήματα των δικτύων αισθητήρων είναι η παρακολούθηση των επιπέδων του πόσιμου νερού, της διάβρωσης του εδάφους και της μόλυνσης του αέρα σε πραγματικό χρόνο.

2.3 Εφαρμογές Υγείας.

Μερικές από τις εφαρμογές των δικτύων αισθητήρων στο κλάδο της υγείας είναι η παροχή διεπαφών προς τους ανάπηρους, η συνεχής παρακολούθηση ασθενών, η διάγνωση, η σύσταση συγκεκριμένων φαρμάκων στα νοσοκομεία, η παρακολούθηση των κινήσεων και των εσωτερικών διεργασιών εντόμων και άλλων μικρών ζώων, η τηλεπαρακολούθηση δεδομένων της ανθρώπινης φυσιολογίας καθώς και ο εντοπισμός και η παρακολούθηση γιατρών και ασθενών σε ένα νοσοκομείο.

Τηλεπαρακολούθηση των δεδομένων της ανθρώπινης φυσιολογίας

Τα δεδομένα της φυσιολογίας που συλλέγονται από τους αισθητήρες μπορούν να αποθηκεύονται για ένα μεγάλο χρονικό διάστημα και να χρησιμοποιούνται για λόγους ιατρικής έρευνας. Οι αισθητήρες μπορούν επίσης να ανιχνεύουν τη συμπεριφορά ηλικιωμένων ανθρώπων, όπως για παράδειγμα κάποιο πέσιμο. Αυτοί οι μικροί κόμβοι δίνουν μια μεγάλη ελευθερία κινήσεων στους εξεταζόμενους ενώ παράλληλα δίνουν στους γιατρούς τη δυνατότητα να ανιχνεύουν προδιαγεγραμμένα για το άτομο συμπτώματα εγκαίρως. Επιπλέον

εξασφαλίζουν μια υψηλότερης ποιότητας ζωή στον εξεταζόμενο, σε σύγκριση με την αντίστοιχη ενός κέντρου νοσηλείας.

Ανίχνευση και παρακολούθηση των κινήσεων των γιατρών και των ασθενών στο νοσοκομείο

Κάθε ασθενής έχει προσαρτημένους επάνω του μικρούς και ελαφρούς αισθητήρες. Κάθε αισθητήρας έχει το δικό του ξεχωριστό ρόλο. Για παράδειγμα ένας αισθητήρας μπορεί να παρακολουθεί τους χτύπους της καρδιάς, ενώ κάποιος άλλος να ελέγχει την πίεση του αίματος. Οι γιατροί μπορούν επίσης να μεταφέρουν τέτοιους αισθητήρες, κάτι που επιτρέπει σε άλλους γιατρούς να τους εντοπίζουν μέσα στο νοσοκομείο.

Σύσταση συγκεκριμένων φαρμάκων στα νοσοκομεία

Αν ένας ασθενής φέρει κάποιους αισθητήρες, τότε είναι εύκολο να ανιχνευθούν οι αλλεργίες του και αντίστοιχα οι κατάλληλες φαρμακευτικές αγωγές για την αντιμετώπιση κάποιου προβλήματος υγείας, αποφεύγοντας έτσι την χορήγηση φαρμάκων στα οποία τελικά ο ασθενής είναι αλλεργικός.

2.4 Οικιακές Εφαρμογές.

Οικιακός αυτοματισμός

Καθώς η τεχνολογία προοδεύει, έξυπνοι αισθητήρες μπορούν να τοποθετηθούν πάνω σε οικιακές συσκευές, όπως ηλεκτρικές σκούπες, φούρνοι μικροκυμάτων, ψυγεία κ.α. Αυτοί οι αισθητήρες μπορούν να επικοινωνούν μεταξύ τους, καθώς και με εξωτερικά δίκτυα μέσω Internet η δορυφόρου. Έτσι ο χρήστης μπορεί πολύ εύκολα να χρησιμοποιεί και να ελέγχει αυτές τις συσκευές όντας απομακρυσμένος από αυτές.

Έξυπνο περιβάλλον

Ο σχεδιασμός του έξυπνου περιβάλλοντος μπορεί να έχει δύο διαφορετικές οπτικές. Την ανθρωποκεντρική και την τεχνοκεντρική. Στη περίπτωση της ανθρωποκεντρικής, ένα έξυπνο περιβάλλον πρέπει να προσαρμόζεται άριστα

στις ανάγκες του χρήστη αναφορικά με τις δυνατότητες εισόδου εξόδου. Στη περίπτωση της τεχνοκεντρικής, πρέπει να αναπτυχθούν καινούργιες τεχνολογίες υλικού, δικτυακές λύσεις καθώς και υπηρεσίες ενδιάμεσου λογισμικού. Ένα σενάριο του πως οι αισθητήρες μπορούν να χρησιμοποιηθούν για την ανάπτυξη ενός έξυπνου περιβάλλοντος είναι το εξής: Οι αισθητήρες εφάπτονται σε έπιπλα και συσκευές και επικοινωνούν μεταξύ τους, καθώς και με έναν server του δωματίου. Κάθε server δωματίου μπορεί να επικοινωνεί με άλλους παρόμοιους servers και να μαθαίνει τι υπηρεσίες προσφέρουν (π.χ. scanning, printing και faxing).Άλλες Εμπορικές Εφαρμογές Κάποιες επιπλέον εμπορικές εφαρμογές είναι η ανίχνευση αδυναμιών σε υλικά, η δημιουργία εικονικών πληκτρολογίων, η κατηγοριοποίηση αγαθών, ο έλεγχος ποιότητας αγαθών, η κατασκευή έξυπνων χώρων σε γραφεία, ο έλεγχος του περιβάλλοντος σε κτίρια γραφείων, ο έλεγχος και η καθοδήγηση ρομπότ σε αυτοματοποιημένα κατασκευαστικά περιβάλλοντα, η κατασκευή διαδραστικών παιχνιδιών, η κατασκευή διαδραστικών μουσείων, ο έλεγχος διεργασιών στη βιομηχανία και η αυτοματοποίηση τους, η παρακολούθηση κατεστραμμένων περιοχών, η κατασκευή έξυπνων δομών με ενσωματωμένους αισθητήρες, η διάγνωση μηχανημάτων, η μετακίνηση, η ανίχνευση κλεμμένων αυτοκινήτων, ο εντοπισμός κινούμενων οχημάτων, καθώς και άλλες πολλές.

2.5 Άλλες εμπορικές Εφαρμογές.

Μερικές από τις εμπορικές εφαρμογές είναι η παρακολούθηση της καταπόνησης των υλικών, η κατασκευή κάθετων κατασκευών, η διαχείριση αποθεμάτων, η παρακολούθηση της ποιότητας της παραγωγής, η κατασκευή έξυπνων χώρων γραφείου, ο περιβαλλοντολογικός έλεγχος σε συγκροτήματα γραφείων, ο έλεγχος των ρομπότ και η καθοδήγηση σε περιβάλλοντα αυτόματης κατασκευής, αλληλεπιδραστικά παιχνίδια, αλληλεπιδραστικά μουσεία, ο έλεγχος των βιομηχανικών διεργασιών και αυτοματισμών, η παρακολούθηση περιοχών καταστροφής, οι έξυπνες κατασκευές με αισθητήριους κόμβους εμφυτευμένους σε αυτές, η διάγνωση μηχανών, οι μεταφορές, η εγκατάσταση βιομηχανικών οργάνων, ο τοπικός έλεγχος μηχανισμών κίνησης, η ανίχνευση και παρακολούθηση κλεφτών αυτοκινήτων, ο εντοπισμός και ανίχνευση κινούμενων οχημάτων .

Έλεγχος περιβάλλοντος σε κτίρια γραφείων

Ο κλιματισμός και η θέρμανση στα περισσότερα κτίρια ελέγχεται από κάποιο κεντρικό σημείο. Συνεπώς η θερμοκρασία μέσα σε ένα δωμάτιο μπορεί να διαφέρει μερικούς βαθμούς. Για παράδειγμα αν υπάρχει μόνο ένα σώμα θέρμανσης και η κατανομή της θερμότητας δεν γίνεται ισομερώς, η μια πλευρά θα είναι πιο ζεστή από την άλλη. Ένα κατανεμημένο ασύρματο δίκτυο αισθητήρων θα μπορούσε να χρησιμοποιηθεί για να ελέγχει τη ροή του αέρα και της θερμοκρασίας σε διαφορετικά μέρη του δωματίου.

Κατασκευή αλληλεπιδραστικών μουσείων

Μελλοντικά τα παιδιά θα μπορούν να αλληλεπιδρούν με αντικείμενα σε μουσεία προκειμένου να μάθουν περισσότερα για αυτά. Αυτά τα αντικείμενα θα μπορούν να ανταποκρίνονται στο άγγιγμα τους και στο λόγο τους. Επίσης τα παιδιά θα μπορούν να συμμετέχουν σε πραγματικού χρόνου πειράματα που δικαιολογούν την αιτία και το αποτέλεσμα του πειράματος, το οποίο θα τους διδάξει πολλά για την επιστήμη.

Ανίχνευση και παρακολούθηση κλοπής αυτοκινήτων

Οι αισθητήρες που εφάπτονται σε ένα αυτοκίνητο μπορούν να προσδιορίσουν την ακριβή του θέση κάθε στιγμή. Έτσι αν το αυτοκίνητο κλαπεί, θα είναι εύκολο για την αστυνομία να εξακριβώσει γρήγορα και αποτελεσματικά που το έχουν μεταφέρει οι κλέφτες.

Διαχείριση και έλεγχος αποθεμάτων

Κάθε αντικείμενο σε μια αποθήκη μπορεί να έχει ένα αισθητήριο κόμβο προσκολλημένο πάνω του . Ο τελικός χρήστης μπορεί να εντοπίσει την ακριβή θέση του αντικειμένου και να μετρήσει τα αντικείμενα της ίδιας κατηγορίας. Αν οι τελικοί χρήστες επιθυμούν να εισάγουν νέα αποθέματα, το μόνο που χρειάζεται να κάνουν είναι να προσκολλήσουν τους κατάλληλους αισθητήριους κόμβους στα αποθέματα αυτά. Οι τελικοί χρήστες μπορούν να εντοπίσουν και να παρακολουθήσουν που βρίσκονται τα αποθέματα κάθε χρονική στιγμή.

Παρακολούθηση και ανίχνευση οχημάτων

Υπάρχουν δύο προσεγγίσεις στην παρακολούθηση και ανίχνευση ενός οχήματος. Η πρώτη είναι ότι η κατεύθυνση του οχήματος αποφασίζεται τοπικά εντός των κόμβων και κατόπιν στέλνεται κεντρικά στον σταθμό βάσης και δεύτερη είναι ότι τα δεδομένα όπως συλλέγονται από τους αισθητήριους κόμβους προωθούνται στον σταθμό βάσης για να αποφασιστεί η θέση του οχήματος.

3

ΠΑΡΑΓΟΝΤΕΣ ΠΟΥ ΕΠΗΡΕΑΖΟΥΝ ΤΟ ΣΧΕΔΙΑΣΜΟ ΕΝΟΣ ΔΙΚΤΥΟΥ ΑΙΣΘΗΤΗΡΩΝ

Ο σχεδιασμός ενός δικτύου αισθητήρων επηρεάζεται από πολλούς παράγοντες. Παρακάτω αναφέρονται μερικοί από αυτούς. Η μελέτη αυτών των παραγόντων (που πρέπει ή δεν πρέπει να διαθέτουν τα δίκτυα αισθητήρων και οι αισθητήριοι κόμβοι) είναι πρωταρχικής σημασίας γιατί παρέχουν τις κατευθύνσεις γύρω από τις οποίες πρέπει να σχεδιαστεί ένα πρωτόκολλο ή αλγόριθμός για δίκτυα αισθητήρων.

3.1 Αντοχή σε σφάλματα.

Είναι δυνατόν, ορισμένοι αισθητήριοι κόμβοι να μπλοκάρουν ή να αποτύχουν κατά τη λειτουργία τους, λόγω εξωτερικών περιβαλλοντολογικών παρεμβολών, έλλειψης ενέργειας ή φυσικής καταστροφής. Η αποτυχία ή καταστροφή (παροδική ή μόνιμη) μερικών αισθητήριων κόμβων δεν θα πρέπει να επηρεάζει τον συνολικό σκοπό του δικτύου αισθητήρων. Αυτό το θέμα αναφέρεται ως αξιοπιστία ή αντοχή σε σφάλματα. Η αντοχή σε σφάλματα είναι η δυνατότητα του δικτύου αισθητήρων να διατηρεί τη λειτουργικότητά του χωρίς διακοπές που να οφείλονται στις αποτυχίες των κόμβων του. Οι αλγόριθμοι και τα πρωτόκολλα μπορούν να σχεδιαστούν, ώστε να εμπεριέχουν τα επίπεδα αντοχής σε λάθη που απαιτούνται από τα δίκτυα αισθητήρων. Αν το περιβάλλον στο οποίο πρόκειται να αναπτυχθεί ένα δίκτυο αισθητήρων δημιουργεί μικρές παρεμβολές τότε τα πρωτόκολλα μπορούν ανάλογα να είναι πιο ελαστικά. Για παράδειγμα, αν ένα δίκτυο αισθητήρων βρίσκεται εγκατεστημένο σε μια οικία προκειμένου να παρακολουθεί τα επίπεδα υγρασίας και θερμοκρασίας, η αντοχή σε σφάλματα

μπορεί να είναι χαμηλή αφού τέτοιου είδους αισθητήριοι κόμβοι δεν καταστρέφονται και δεν επηρεάζονται εύκολα από το περιβάλλον. Αντιθέτως, σε ένα πεδίο μάχης, το δίκτυο αισθητήρων που θα εγκατασταθεί πρέπει να έχει μεγάλη αντοχή σε σφάλματα διότι είναι πολύ εύκολο να καταστραφούν αρκετοί κόμβοι του από εχθρικές επιχειρήσεις. Από τα παραπάνω διαπιστώνεται ότι η αντοχή σε σφάλματα εξαρτάται και από την εφαρμογή για την οποία προορίζεται το δίκτυο. Συνεπώς αυτό πρέπει να λαμβάνεται υπόψη στο σχεδιασμό του δικτύου αισθητήρων αλλά και των ίδιων των κόμβων.

3.2 Δυνατότητα κλιμάκωσης.

Ο αριθμός των αισθητήριων κόμβων που έχουν αναπτυχθεί για την μελέτη ενός φαινομένου μπορεί να είναι της τάξης των εκατοντάδων ή χιλιάδων. Ανάλογα με την εφαρμογή, ο αριθμός αυτός μπορεί να φτάσει και την τάξη των εκατομμυρίων. Ότι πρωτόκολλο σχεδιαστεί θα πρέπει να μπορεί να χειριστεί αυτόν τον αριθμό των κόμβων. Πρέπει επίσης να χρησιμοποιήσουν την υψηλή πυκνότητα με την οποία εγκαθίστανται οι αισθητήριοι κόμβοι. Η πυκνότητα μπορεί να διαφέρει από μερικούς μέχρι εκατοντάδες κόμβους σε μια περιοχή η οποία μπορεί να είναι μικρότερη σε διάμετρο από 10m Επιπλέον, ο αριθμός των κόμβων σε μια περιοχή μπορεί να χρησιμοποιηθεί για να δείξει την πυκνότητα των κόμβων. Η πυκνότητα αυτή εξαρτάται από την εφαρμογή για την οποία εγκαταστάθηκαν οι αισθητήριοι κόμβοι.

3.3 Κόστος παραγωγής.

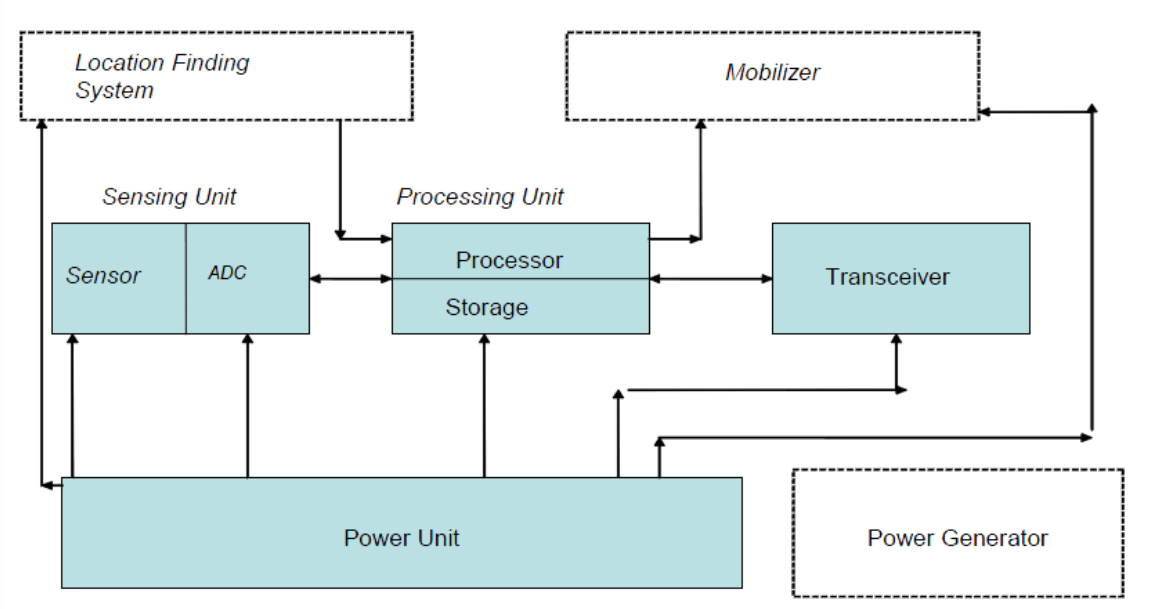
Αφού τα δίκτυα αισθητήρων αποτελούνται από ένα μεγάλο αριθμό κόμβων, το κόστος ενός μόνο αισθητήριου κόμβου είναι πολύ σημαντικό για ένα τέτοιο δίκτυο. Αν το κόστος του δικτύου είναι πιο ακριβό από το να εγκατασταθούν οι παραδοσιακοί αισθητήρες, τότε τα δίκτυα αισθητήρων δεν θα συμφέρουν οικονομικά. Αποτέλεσμα του παραπάνω είναι ότι το κόστος του κάθε αισθητήριου κόμβου πρέπει να είναι όσο το δυνατόν μικρότερο.

Το κόστος ενός ραδιοσυστήματος τεχνολογίας Bluetooth, η οποία είναι μια χαμηλού κόστους συσκευή, είναι 10 φορές πιο ακριβή από την στοχευόμενη τιμή ενός αισθητήριου κόμβου. Επιπλέον ο αισθητήριος κόμβος διαθέτει αισθητήρες και συσκευές επεξεργασίας. Επιπλέον αυτών, είναι δυνατόν να εξοπλιστεί με σύστημα εντοπισμού θέσης με σύστημα κίνησης ή με σύστημα παραγωγής ενέργειας, ανάλογα με την εφαρμογή την οποία πρόκειται να εκτελέσει.

Αποτέλεσμα όλων αυτών είναι ότι, η κατασκευή ενός αισθητήριου κόμβου με τόσο χαμηλό κόστος αποτελεί ένα πολύ ενδιαφέρον πρόβλημα.

3.4 Περιορισμοί υλικού.

Ένας κόμβος ενός ασύρματου δικτύου αισθητήρων, όπως φαίνεται και στην παρακάτω εικόνα (Εικόνα 2), αποτελείται κατά βάση από τέσσερα τμήματα: 1) μια μονάδα αίσθησης, 2) μια μονάδα επεξεργασίας, 3) ένα πομποδέκτη και 4) μια μονάδα ενέργειας.



Εικόνα 2: Τμήματα κόμβου ασύρματου δικτύου αισθητήρων. [2]

Ανάλογα με την εφαρμογή για την οποία προορίζεται μπορεί να διαθέτει επιπλέον τμήματα, όπως σύστημα εντοπισμού θέσης, μονάδα παραγωγής ενέργειας και μονάδα κίνησης. Η μονάδα αίσθησης συνήθως αποτελείται από δύο υπομονάδες: α) τους αισθητήρες και β) τους αναλογικό-ψηφιακούς μετατροπείς. Τα αναλογικά σήματα που παράγονται από τα αισθητήρια όργανα και βασίζονται στα παρατηρούμενα φαινόμενα μετατρέπονται σε ψηφιακά σήματα από τους αναλογικό-ψηφιακούς μετατροπείς και κατόπιν μεταφέρονται στην μονάδα επεξεργασίας. Αυτή η μονάδα, που γενικά σχετίζεται με μια μικρή μονάδα αποθήκευσης, διαχειρίζεται τις διαδικασίες που κάνουν τον αισθητήριο κόμβο να συνεργάζεται με άλλους κόμβους για να φέρει εις πέρας τους

προσδιορισμένους στόχους. Η μονάδα του πομποδέκτη συνδέει τον αισθητήριο κόμβο στο δίκτυο. Ένα από τα πιο σημαντικά τμήματα του αισθητήριου κόμβου είναι η μονάδα ενέργειας. Οι μονάδες ενέργειας είναι δυνατόν να υποστηρίζονται από μια μονάδα εξαγωγής και παραγωγής ενέργειας (scavenging energy) από το περιβάλλον όπως οι ηλιακές κυψέλες. Υπάρχουν όμως και άλλες μικρότερες μονάδες, των οποίων η χρήση εξαρτάται από την εφαρμογή για την οποία χρησιμοποιούνται οι αισθητήριοι κόμβοι.

Οι περισσότερες από τις τεχνικές δρομολόγησης και οι εφαρμογές παρακολούθησης των δικτύων αισθητήρων απαιτούν την γνώση της θέσης με μεγάλη συνήθως ακρίβεια. Έτσι είναι σύνηθες για ένα αισθητήριο κόμβο να έχει προσαρτημένη και μια μονάδα εύρεσης θέσης. Μια μονάδα κίνησης είναι δυνατόν να χρησιμοποιηθεί όταν απαιτείται να κινηθούν οι αισθητήριοι κόμβοι προκειμένου να παρακολουθήσουν καλύτερα το παρατηρούμενο φαινόμενο.

Όλες αυτές οι μικρότερες μονάδες πρέπει να μπορούν να χωρέσουν σε ένα χώρο μεγέθους σπιρτόκουτου. Το απαιτούμενο μέγεθος μπορεί να απαιτείται να είναι μικρότερο από ένα κυβικό εκατοστό και να είναι αρκετά ελαφρύ για να παραμένει αιωρούμενο στον αέρα. Εκτός από το μέγεθος, υπάρχουν ακόμα πιο αυστηροί περιορισμοί για τους αισθητήριους κόμβους όπως:

- ✓ Πρέπει να καταναλώνουν εξαιρετικά χαμηλή ενέργεια.
- ✓ Πρέπει να λειτουργούν ακόμα και σε πολύ πυκνή χωρική τοποθέτηση.
- ✓ Πρέπει να έχουν χαμηλό κόστος παραγωγής και να είναι αναλώσιμοι.
- ✓ Πρέπει να είναι αυτόνομοι και να λειτουργούν χωρίς παρακολούθηση.
- ✓ Πρέπει να προσαρμόζονται στο περιβάλλον που θα λειτουργούν.

Αφού οι αισθητήριοι κόμβοι είναι συνήθως δύσχρηστοι σε ότι αφορά τις απαιτήσεις τους και τη λειτουργικότητά τους , η διάρκεια ζωής ενός δικτύου αισθητήρων εξαρτάται άμεσα από την διάρκεια ζωής των πηγών ενέργειας των κόμβων. Η ενέργεια είναι ένας σπάνιος πόρος του συστήματος εξαιτίας των περιορισμών του μεγέθους. Για το σύστημα του ολοκληρωμένου ασύρματου δικτύου αισθητήρων (Wireless Integrated Network Sensors WINS) η ολική ενέργεια που πρέπει να παρέχεται πρέπει να είναι μικρότερη των 30μA προκειμένου να έχει μεγάλη διάρκεια λειτουργίας. Οι κόμβοι στο παραπάνω σύστημα παίρνουν ενέργεια από μια τυπική μπαταρία Λιθίου (Li) τύπου νομίσματος (2.5 cm διάμετρος και 1 cm πάχος). Είναι δυνατόν να επεκτείνουμε την διάρκεια ζωής των δικτύων αισθητήρων χρησιμοποιώντας τεχνικές εξαγωγής και παραγωγής ενέργειας από το περιβάλλον.

3.5 Τοπολογία δικτύου αισθητήρων.

Ένας μεγάλος αριθμός μη προσβάσιμων και χωρίς παρακολούθηση αισθητήριων κόμβων, οι οποίοι εύκολα μπορούν να χαλάσουν, κάνει την διατήρηση της τοπολογίας του δικτύου μια μεγάλη πρόκληση. Η πυκνότητα μπορεί να φθάνει και τους 20 κόμβους/m^3, κάτι που δυσκολεύει ακόμα περισσότερο την διαχείριση της τοπολογίας. Μπορούμε να εξετάσουμε την διατήρηση της τοπολογίας του δικτύου αισθητήρων σε 3 φάσεις.

Φάση πριν την εγκατάσταση και φάση εγκατάστασης

Οι αισθητήριοι κόμβοι μπορούν είτε να διασπαρθούν μαζικά είτε να τοποθετηθούν ένας-ένας στο χώρο. Μπορούν να εγκατασταθούν με τους εξής τρόπους :

- Να πεταχτούν από ένα αεροπλάνο
- Να βρίσκονται σε ένα βλήμα πυροβολικού (ή πύραυλο) το οποίο εκρήγνυται και τους διασπείρει στην περιοχή.
- Να ριφθούν με ένα καταπέλτη π.χ. από το κατάστρωμα ενός πλοίου.
- Να τοποθετηθούν ένας – ένας από ένα άνθρωπο ή ένα ρομπότ.

Αν και ο μεγάλος αριθμός των αισθητήρων καθώς και η χωρίς παρακολούθηση εγκατάστασή τους συνήθως περιλαμβάνει την τοποθέτησή τους σύμφωνα με ένα προσεχτικά μελετημένο σχέδιο, η αρχική εγκατάσταση πρέπει να πληροί κάποια κριτήρια :

- Μείωση του κόστους εγκατάστασης.
- Εξαφάνιση της ανάγκης για οποιαδήποτε προ-οργάνωση ή προ-σχεδιασμό.
- Αύξηση της ευελιξίας τοποθέτησης.
- Προώθηση της αυτό-οργάνωσης και της αντοχής σε σφάλματα.

Φάση μετά την εγκατάσταση

Μετά την εξάπλωση, οι αλλαγές στην τοπολογία οφείλονται σε αλλαγές στους αισθητήριους κόμβους όπως :

- Θέση.
- Δυνατότητα επικοινωνίας.
- Διαθέσιμη ενέργειας.
- Δυσλειτουργία.

- Λεπτομέρειες στο σκοπό για τον οποίο εγκαταστάθηκαν.

Οι αισθητήριοι κόμβοι μπορούν να εγκατασταθούν και στατικά. Οι αποτυχίες είναι ένα σύνηθες φαινόμενο λόγω έλλειψης ενέργειας ή καταστροφής. Είναι επίσης πιθανό να έχουμε δίκτυα αισθητήρων των οποίων οι κόμβοι συνεχώς κινούνται. Εκτός από τα προβλήματα τα οποία είναι φυσικό να αντιμετωπίζουν εξαιτίας των χαρακτηριστικών τους είναι δυνατόν ακόμα να έχουμε και δολιοφθορές. Αποτέλεσμα όλων των παραπάνω είναι οι τοπολογίες των δικτύων αισθητήρων να υπόκεινται σε συχνές αλλαγές.

Φάση εγκατάστασης επιπλέον κόμβων

Επιπλέον κόμβοι είναι δυνατόν να εγκατασταθούν οποιαδήποτε χρονική στιγμή για να αντικαταστήσουν τους κόμβους που παρουσιάζουν δυσλειτουργίες ή λόγω αλλαγών στον αρχικό σκοπό για τον οποίο εγκαταστάθηκαν. Η πρόσθεση νέων κόμβων στο δίκτυο δημιουργεί την ανάγκη για αναδιοργάνωση. Προκειμένου να αντιμετωπίσουμε τις συχνές αλλαγές στην τοπολογία ενός ασύρματου δικτύου αισθητήρων, το οποίο αποτελείται από ένα μεγάλο αριθμό κόμβων με μεγάλους περιορισμούς στην κατανάλωση ενέργειας χρειαζόμαστε ειδικά σχεδιασμένα πρωτόκολλα δρομολόγησης.

3.6 Περιβάλλον.

Οι αισθητήριοι κόμβοι τοποθετούνται , είτε πολύ κοντά στο υπό παρατήρηση φαινόμενο, είτε ακριβώς μέσα σε αυτό. Έτσι συνήθως εργάζονται χωρίς παρακολούθηση σε απομακρυσμένες γεωγραφικές περιοχές. Για παράδειγμα, είναι δυνατόν να εργάζονται:

- Στο εσωτερικό ενός μεγάλου μηχανήματος,
- Στα βάθη του ωκεανού,
- Μέσα σε ένα κυκλώνα,
- Στην επιφάνεια ενός ωκεανού στην διάρκεια μια καταιγίδας,
- Σε μια περιοχή μολυσμένη από ραδιενέργεια ή χημικές ουσίες,
- Στο πεδίο της μάχης, πίσω από τις γραμμές του εχθρού,
- Σε ένα σπίτι ή σε ένα μεγάλο κτίριο,
- Σε μια μεγάλη αποθήκη,
- Εμφυτευμένοι σε ζώα,
- Ενσωματωμένοι σε ταχέως κινούμενα οχήματα,
- Στα νερά ενός ποταμού.

Η παραπάνω λίστα μας δίνει μια περιγραφή των συνθηκών, υπό τις οποίες δουλεύουν οι κόμβοι ενός ασύρματου δικτύου αισθητήρων.

3.7 Μέσα μετάδοσης.

Σ' ένα ασύρματο δίκτυο αισθητήρων, οι αισθητήριοι κόμβοι συνδέονται συνήθως μέσω του ασύρματου μέσου. Αυτού του είδους οι συνδέσεις μπορούν να πραγματοποιηθούν με τη βοήθεια ραδιοσυχνοτήτων, υπέρυθρων ή οπτικών μέσων. Προκειμένου να γίνει εφικτή μια παγκόσμια χρήση αυτών των δικτύων, το επιλεγμένο μέσο πρέπει να είναι διαθέσιμο παγκοσμίως.

Ευρέως χρησιμοποιούμενη συχνότητα στις ασύρματες ζεύξεις είναι η μπάντα ISM (Industrial Scientific Medical ISM Band), η οποία προσφέρεται και χωρίς άδεια χρήσης στις περισσότερες χώρες. Στην Ευρώπη χρησιμοποιούνται οι συχνότητες των 433 MHz και στην Β. Αμερική οι συχνότητες των 915 MHz. Στις περισσότερες περιπτώσεις δικτύων αισθητήρων χρησιμοποιούμε την επικοινωνία η οποία γίνεται μέσω ραδιοσυχνοτήτων. Μεταξύ αυτών διακρίνουμε τις εξής περιπτώσεις :

- Το μAMPS το οποίο χρησιμοποιεί πομποδέκτη συμβατό με Bluetooth στα 2.4GHz, με ένα ενσωματωμένο συνθέτη συχνοτήτων,
- Ένας αισθητήρας χαμηλής ενέργειας ο οποίος χρησιμοποιεί πομποδέκτη ραδιοσυχνότητας ενός καναλιού, το οποίο βρίσκεται σε συχνότητα λειτουργίας στα 916 MHz,
- Η αρχιτεκτονική WINS χρησιμοποιεί ραδιοσυχνότητες μεταξύ των κόμβων.

Ένας άλλος τρόπος επικοινωνίας στα δίκτυα αισθητήρων είναι η χρήση υπερύθρων, για την οποία δεν απαιτείται άδεια χρήσης, και επιπλέον είναι ανθεκτική στις παρεμβολές από ηλεκτρικές συσκευές. Το μοναδικό μειονέκτημα που εντοπίζουμε είναι η απαίτηση για οπτική επαφή των επικοινωνούντων συσκευών. Το μειονέκτημα αυτό αποτρέπει τη χρήση υπερύθρων ως μέσο μετάδοσης στα ασύρματα δίκτυα αισθητήρων. Οι συνήθεις απαιτήσεις των εφαρμογών για τις οποίες χρησιμοποιούνται τα δίκτυα αισθητήρων δημιουργούν μεγάλη πρόκληση στην επιλογή ενός μέσου μετάδοσης. Για παράδειγμα, σε εφαρμογές που μπορούν να είναι υποθαλάσσιες μπορεί να απαιτείται η χρήση του νερού ως μέσου μετάδοσης. Επιπλέον, λόγω του ότι η κεραία ενός αισθητήρα μπορεί να μην έχει το απαιτούμενο ύψος, ή η ισχύ εκπομπής να είναι περιορισμένη, εκτός από την επιλογή του μέσου, μεγάλο ρόλο παίζει η χρήση ισχυρής κωδικοποίησης και η επιλογή της κατάλληλης συχνότητας προκειμένου να γίνει στο έπακρο εκμετάλλευση των χαρακτηριστικών του καναλιού.

3.8 Κατανάλωση ενέργειας.

Δεδομένου ότι ο κάθε αισθητήριος κόμβος αποτελεί αυτοτελή μικροηλεκτρονική συσκευή, μπορεί να εφοδιαστεί με μια περιορισμένη πηγή ενέργειας (<0.5 Ah, 1.2V). Δεδομένου ότι η αντικατάσταση αυτής της πηγής ενέργειας είναι αδύνατη, η ζωή του αισθητήριου κόμβου εξαρτάται από αυτήν. Σε κάθε δίκτυο αισθητήρων ο κάθε κόμβος παίζει το ρόλο του αποστολέα και του δρομολογητή, επομένως εάν παρουσιαστεί κάποια βλάβη σε έναν από τους κόμβους του δικτύου, απαιτείται συνολική αναδιοργάνωση του δικτύου και επαναδρομολόγηση των μηνυμάτων. Η σωστή διαχείριση της ενέργειας των κόμβων παίζει μεγάλο ρόλο, και για αυτό το λόγο αποδίδεται σε τρεις λειτουργίες:

- Επικοινωνία,
- Αίσθηση,
- Επεξεργασία δεδομένων.

Επικοινωνία

Η πιο απαιτητική λειτουργία από άποψη κατανάλωσης ενέργειας είναι η επικοινωνία. Συνήθως για τις μικρές αποστάσεις που λειτουργούν οι αισθητήριοι κόμβοι η κατανάλωση είναι ίδια κατά την εκπομπή και την λήψη. Βεβαίως, εκτός από αυτό, σοβαρό ρόλο παίζει και το άνοιγμα και κλείσιμο του κυκλώματος του πομποδέκτη.

Αίσθηση

Το είδος της αίσθησης το οποίο καλείται να προσομοιώσει ένας αισθητήρας αναμφισβήτητα επηρεάζει το ποσό ενέργειας που θα απαιτηθεί για την λειτουργία αυτή. Για παράδειγμα, ένας αισθητήρας μέτρησης ραδιενέργειας απαιτεί μεγαλύτερη ισχύ από έναν απλό αισθητήρα μέτρησης θερμοκρασίας.

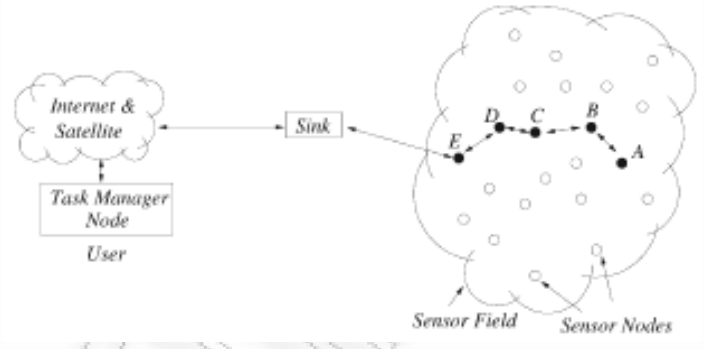
Επεξεργασία δεδομένων

Η κατανάλωση ενέργειας κατά τη φάση της επεξεργασίας δεδομένων είναι μικρότερη από αυτή κατά τη φάση της επικοινωνίας. Συνεπώς, ο κόμβος κατά τη φάση της επεξεργασίας θα έχει ενσωματωμένο ένα κύκλωμα επεξεργασίας προκειμένου να επεξεργάζεται τα δεδομένα με απώτερο σκοπό την αποστολή λιγότερων πακέτων κατά τη φάση της επικοινωνίας

4

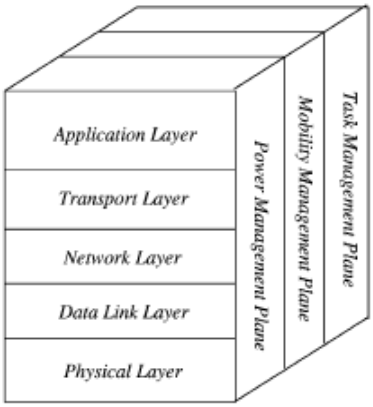
ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΑΣΥΡΜΑΤΩΝ ΔΙΚΤΥΩΝ ΑΙΣΘΗΤΗΡΩΝ

Οι ασύρματοι κόμβοι διασπείρονται σε ένα πεδίο όπως φαίνεται και στην Εικόνα 3. Κάθε ένας από αυτούς συλλέγει δεδομένα, τα επεξεργάζεται και τα στέλνει πίσω σε



Εικόνα 3: Διασπορά ασύρματων κόμβων σε ένα πεδίο. [2]

ένα κεντρικό σημείο και από εκεί καταλήγουν στους ενδιαφερόμενους χρήστες. Η στοίβα πρωτοκόλλου που χρησιμοποιείται από το κεντρικό σημείο αλλά και από όλους τους κόμβους φαίνεται στην Εικόνα 4. Όπως φαίνεται αποτελείται από τα εξής επίπεδα: φυσικό, ζεύξης δεδομένων, δικτύου, μεταφοράς και εφαρμογής, καθώς και από τα κάτωθι επίπεδα διαχείρισης (management planes) ενέργειας, κινήσεως και στόχου.



Εικόνα 4: Η στοίβα πρωτοκόλλου των δικτύων αισθητήρων. [2]

Τα τρία τελευταία επίπεδα διαχείρισης βοηθούν τους αισθητήριους κόμβους να συνεργαστούν καλύτερα ο ένας με τον άλλο προκειμένου να φέρουν σε πέρας τον σκοπό για τον οποίο εγκαταστάθηκαν καταναλώνοντας όσο το δυνατόν λιγότερη ενέργεια. Τα υπόλοιπα επίπεδα λειτουργούν ανάλογα με αυτά του προτύπου OSI. Τρία υπάρχοντα σχέδια που χρησιμοποιούν αυτά τα επίπεδα είναι το WINS, το smart dust motes, και το μAMPS.

4.1 Φυσικό επίπεδο.

Το φυσικό επίπεδο είναι υπεύθυνο για την επιλογή της συχνότητας, την δημιουργία του φέροντος, την ανίχνευση του σήματος, την διαμόρφωση και την κρυπτογράφηση των δεδομένων. Βασικός παράγοντας στον σχεδιασμό του επιπέδου παραμένει η ενέργεια που καταναλώνεται στην επικοινωνία. Βέβαια εξαιτίας της πυκνής χωρικά ανάπτυξης των αισθητήρων και της δυνατότητας επικοινωνίας μέσω πολλαπλών κόμβων (multi-hop communication) έχουμε σημαντική εξοικονόμηση στην ενέργεια αλλά και μικρές απώλειες στο σήμα, άρα δυνατότητα για μικρότερη εκπεμπόμενη ενέργεια. Ακόμα το όλο πεδίο είναι ανεξερεύνητο και μερικά θέματα που μπορούν να αναφερθούν είναι η επιλογή της διαμόρφωσης που βοηθά στην μείωση της καταναλισκόμενης ενέργειας. Επίσης άλλο θέμα αφορά την επιλογή της συχνότητας. Φαίνεται ότι η χρήση μιας ευρείας συχνότητας (Ultra Wide Band) βοηθά στην αντιμετώπιση των απωλειών και στην εξοικονόμηση της ενέργειας.

4.2 Επίπεδο ζεύξης δεδομένων.

Το επίπεδο αυτό είναι υπεύθυνο για την πολυπλεξία των δεδομένων, την ανίχνευση των πλαισίων δεδομένων, την πρόσβαση στο μέσο και τον έλεγχο λαθών.

4.2.1 Έλεγχος πρόσβασης στο μέσο.

Το επίπεδο ζεύξης δεδομένων πρέπει να πετύχει 2 σκοπούς : α) την κατασκευή της δομής του δικτύου προκειμένου να έχουμε επικοινωνία από σημείο- προς-σημείο και να δοθεί στο δίκτυο μια αυτό-οργανωτική ικανότητα και β) να μοιραστεί το μέσο μετάδοσης ισότιμα και αποτελεσματικά μεταξύ των αισθητήριων κόμβων.

Τα υπάρχοντα πρωτόκολλα ζεύξης δεδομένων είναι ανεπαρκή. Αυτό συμβαίνει γιατί έχουν ως πρωταρχικό σκοπό την παροχή ποιότητας υπηρεσίας χωρίς να υπολογίζουν σε μεγάλο βαθμό το θέμα της καταναλισκόμενης ενέργειας αφού θεωρούν ότι είναι δυνατόν να την αναπληρώνουμε. Επίσης ένας άλλος παράγοντας είναι ο αριθμός των κόμβων σε κάθε δίκτυο καθώς και οι συχνές αλλαγές στην τοπολογία αλλά και η απουσία κάποιου κεντρικού κόμβου ο οποίος παίζει το ρόλο του συντονιστή. Μερικά προτεινόμενα πρωτόκολλα του επιπέδου ζεύξης είναι :

SMACS και ο αλγόριθμος EAR.

Το πρωτόκολλο SMACS επιτυγχάνει την έναρξη του δικτύου καθώς και την οργάνωση του επιπέδου ζεύξης δεδομένων και ο αλγόριθμος EAR δίνει την δυνατότητα για διαφανή σύνδεση των ασύρματων κόμβων στο δίκτυο αισθητήρων. Το SMACS είναι ένα κατανεμημένο (πρωτόκολλο) χτισίματος δομής το οποίο δίνει την δυνατότητα στους κόμβους να ανακαλύψουν τους γείτονές τους και να προγραμματίσουν χρονικά την εκπομπή και την λήψη χωρίς την ανάγκη ύπαρξης κόμβων που θα παίζουν το ρόλο του κεντρικού οργανωτή, είτε σε τοπικό είτε σε καθολικό επίπεδο. Σε αυτό το πρωτόκολλο, η ανακάλυψη των γειτόνων και η ανάθεση των φάσεων του καναλιού συνδυάζονται έτσι ώστε από την ώρα που οι κόμβοι θα ακούσουν τους γείτονές τους θα έχουν σχηματίσει ένα ολοκληρωμένο δίκτυο. Ένας επικοινωνιακός δεσμός αποτελείται από ένα ζευγάρι χρονοθυρίδων που λειτουργούν με την τυχαία επιλογή συχνότητας (σταθερής ή με αναπηδήσεις). Αυτό είναι εφικτό αφού το διαθέσιμο εύρος συχνοτήτων είναι πολύ μεγαλύτερο από τον αναμενόμενο ρυθμό μετάδοσης. Με τον τρόπο αυτό δεν είναι αναγκαίο να υπάρχει συγχρονισμός όλων των κόμβων του δικτύου αλλά μόνο αυτών που επικοινωνούν. Η διατήρηση της ενέργειας επιτυγχάνεται με την χρήση ενός τυχαίου προγράμματος ξυπνήματος κατά την φάση της σύνδεσης και με κλείσιμο του πομποδέκτη κατά την φάση των κενών χρονοθυρίδων.

Το πρωτόκολλο EAR**Error! Reference source not found.** προσπαθεί να προσφέρει συνεχή υπηρεσία στους κινούμενους κόμβους, είτε κάτω από συνθήκες κίνησης είτε σταθερότητας. Εδώ οι κινούμενοι κόμβοι έχουν τον πλήρη έλεγχο της επικοινωνίας και αποφασίζουν επίσης για το πότε να διακόψουν την σύνδεση μειώνοντας έτσι την επικεφαλίδα. Το EAR είναι διαφανές προς το SMACS και έτσι το τελευταίο είναι λειτουργικό μέχρι την είσοδο κινούμενων κόμβων στο δίκτυο. Σε αυτό το μοντέλο το δίκτυο υπολογίζεται να είναι γενικά στατικό, δηλ. κάθε κινούμενος κόμβος έχει ένα αριθμό στατικών σταθμών στην εμβέλειά του. Ένα μειονέκτημα ενός τέτοιου σχήματος ανάθεσης χρονοθυρίδων είναι η πιθανότητα μέλη που ανήκουν σε διαφορετικά υποδίκτυα να μην συνδεθούν ποτέ.

Μέσο πρόσβασης βασισμένο στο CSMA .

Τα παραδοσιακά πρωτόκολλα που βασίζονται στο CSMA για πρόσβαση στο μέσο δεν είναι κατάλληλα για τα δίκτυα αισθητήρων, αφού πρέπει να υποστηρίζουν μεταβλητή αλλά πολύ συσχετιζόμενη και κατά κύριο λόγο περιοδική κίνηση. Τα πρωτόκολλα αυτού του τύπου έχουν δύο πολύ σημαντικά τμήματα : το μηχανισμό παρακολούθησης του μέσου (listening mechanism) και το σχήμα οπισθοχώρησης. Από προσομοιώσεις αποδείχτηκε ότι οι συνεχείς περίοδοι παρακολούθησης του μέσου (listen periods) είναι αποτελεσματικές ενεργειακά και η εισαγωγή των τυχαίων καθυστερήσεων μειώνει την πιθανότητα ύπαρξης συνεχών συγκρούσεων

Ένας προσαρμοζόμενος ρυθμός μετάδοσης επιτυγχάνει την ισοτιμία στην πρόσβαση του μέσου εξισορροπώντας τους ρυθμούς της κίνησης που παράγεται στους κόμβους είτε που διέρχεται από αυτούς. Ελέγχεται ο ρυθμός των δεδομένων που αποστέλλεται για να επιτραπεί η μετάδοση των διερχόμενων δεδομένων (ο κάθε κόμβος αποστέλλει τα δικά του δεδομένα αλλά λειτουργεί και σαν δρομολογητής για τα δεδομένα των γειτονικών του κόμβων). Μέσω μιας λειτουργίας δίνεται προτεραιότητα στην διερχόμενη κίνηση από ότι στην παραγόμενη. Η υπολογιστική φύση αυτού του ελέγχου τον κάνει πιο αποτελεσματικό ενεργειακά σε σχέση με την ύπαρξη αρχικής φάσης συνεννόησης (handshaking) για επικοινωνία μεταξύ των κόμβων. Επίσης γίνεται προσπάθεια να μειωθεί το πρόβλημα των κρυμμένων κόμβων με τον συνεχή συντονισμό των ρυθμών εκπομπής και με την εκτέλεση αλλαγών στην φάση, έτσι ώστε τα περιοδικά κύματα δεδομένων να είναι λιγότερο πιθανό να συγκρουστούν συνεχόμενα.

4.2.2 Καταστάσεις λειτουργίας εξοικονόμησης ενέργειας .

Ανεξάρτητα από ποιο πρωτόκολλο πρόσβασης στο μέσο θα χρησιμοποιήσουμε πρέπει οπωσδήποτε να υποστηρίζει κατάσταση λειτουργίας για εξοικονόμηση ενέργειας. Ο πιο προφανής τρόπος είναι η απενεργοποίηση του πομποδέκτη όταν αυτός δεν είναι αναγκαίος. Βέβαια το παραπάνω μπορεί και να έχει τα αντίθετα αποτελέσματα αφού οι αισθητήριοι κόμβοι ανταλλάσσουν μικρά μηνύματα με αποτέλεσμα αν σε κάθε μικρό χρονικό διάστημα που δεν έχουμε δραστηριότητα ανταλλαγής πακέτων απενεργοποιούμε τον πομποδέκτη να καταναλώνουμε περισσότερη ενέργεια λόγω της ανάγκης να τον ενεργοποιήσουμε ξανά. Θα πρέπει να εντοπίσουμε το σύνολο των καταστάσεων λειτουργίας ενός ασύρματου αισθητήρα που εξαρτώνται από τις καταστάσεις του μικρό-επεξεργαστή, της μνήμης, του αναλογοψηφιακού μετατροπέα και του πομποδέκτη. Κάθε μία από αυτές τις καταστάσεις χαρακτηρίζεται από την ενέργεια που καταναλώνει καθώς και την ενέργεια που καταναλώνει για να μεταβεί από μια κατάσταση σε μία άλλη.

4.2.3 Έλεγχος Λαθών.

Υπάρχουν δύο κατηγορίες ελέγχου και διόρθωσης λαθών. Η μία είναι η αυτόματη αίτηση για επανάληψη (Automatic Repeat Request ARQ) και η δεύτερη είναι η διόρθωση των λαθών στον δέκτη (Forward Error Correction FEC).

ARQ.

Η χρησιμότητα της μεθόδου είναι πολύ μικρή εξαιτίας του κόστους των επανεκπομπών και της μεγάλης επικεφαλίδας. Συνεπώς η χρήση του FEC είναι καλύτερη επιλογή λαμβάνοντας υπόψη και το γεγονός ότι τα μηνύματα που ανταλλάσσονται είναι μικρά και κατ' επέκταση η επιπλέον κωδικοποίηση είναι μικρή.

FEC.

Η αξιοπιστία του καναλιού είναι αναγκαία στα δίκτυα ασύρματων αισθητήρων. Ένας τρόπος μέτρησης της είναι ο ρυθμός εμφάνισης λαθών στο κανάλι (Bit Error Rate BER). Το BER μπορεί να μειωθεί με δύο τρόπους, είτε με την αύξηση της ισχύος στον πομπό είτε με την χρήση κατάλληλου κώδικα διόρθωσης λαθών. Λόγω της δυνατότητας του κόμβου να εκτελεί επεξεργασία δεδομένων και αποκωδικοποίηση με μικρότερο κόστος σε ενέργεια από ότι η επανεκπομπή των ίδιων δεδομένων, η μέθοδος αυτή φαίνεται ιδανική για την περίπτωση των

δικτύων αυτών. Βέβαια, πρέπει να ληφθεί υπόψη πόση ενέργεια ξοδεύεται στην επεξεργασία, γιατί αν αυτή είναι μεγαλύτερη από ότι να στέλνεται χωρίς κωδικοποίηση το σήμα, τότε η όλη διαδικασία είναι αντιοικονομική.

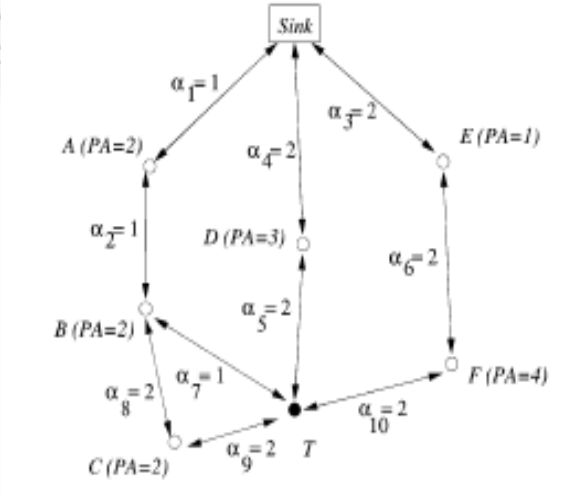
4.3 Επίπεδο Δικτύου.

Στην Εικόνα 3 φαίνεται ένας τρόπος εγκατάστασης των ασύρματων αισθητήρων για την παρατήρηση ενός φαινόμενου. Απαιτούνται ειδικά πρωτόκολλα δρομολόγησης προκειμένου η πληροφορία από το φαινόμενο να φτάσει στους τελικούς χρήστες. Τα υπάρχοντα πρωτόκολλα δεν επαρκούν και απαιτείται η χρήση άλλων. Οι αρχές σύμφωνα με τις οποίες πρέπει να σχεδιάζεται το επίπεδο δικτύου ενός ασύρματου δικτύου αισθητήρων είναι :

- Αποτελεσματική χρήση της ενέργειας.
- Τα δίκτυα αισθητήρων είναι συνήθως δεδομένο-κεντρικά
- Ο συγκερασμός των δεδομένων είναι χρήσιμος όταν δεν εμποδίζει την συνεργατική προσπάθεια των ασύρματων κόμβων.
- Ένα ιδανικό δίκτυο αισθητήρων έχει διευθυνσιοδότηση βασισμένη σε χαρακτηριστικά (attributes) και γνώση της θέσης.

Υπάρχουν οι ακόλουθες τεχνικές που μπορούν να χρησιμοποιηθούν προκειμένου να επιλεχτεί η πιο αποτελεσματική διαδρομή από άποψη οικονομίας ενέργειας. Προκειμένου να περιγραφούν καλύτερα ακολουθεί το σενάριο στην Εικόνα 5:

Ο κόμβος Τ είναι αυτός ο οποίος στέλνει τα δεδομένα τα οποία κατευθύνονται προς τον κόμβο συγκεντρωτή (sink). Οι τέσσερις εναλλακτικές διαδρομές και τα κόστη σε ενέργεια φαίνονται παρακάτω :



Εικόνα 5: Σενάριο δρομολόγησης. [5]

Διαδρομή 1 : Sink-A-B-T συνολική ΔΕ=4 συνολικό α=3.

Διαδρομή 2 : Sink-A-B-C-T συνολική ΔΕ=6 συνολικό α=6.

Διαδρομή 3 : Sink-D-T συνολική ΔΕ=3 συνολικό α=4.

Διαδρομή 4 : Sink-E-F-T συνολική ΔΕ=5 συνολικό α=6.

Όπου ΔΕ είναι η Διαθέσιμη Ενέργεια (Available Power AP) και το α η ενέργεια που απαιτείται για την εκπομπή ενός πακέτου από τον ένα στον άλλο κόμβο.

- *Διαδρομή βάση της μέγιστης Διαθέσιμης Ενέργειας (ΔΕ) :* Προτιμάται η διαδρομή με την μέγιστη διαθέσιμη ενέργεια. Η ολική ΔΕ υπολογίζεται από το άθροισμα των ΔΕ κάθε κόμβου κατά μήκους της διαδρομής. Όταν μέρος μιας διαδρομής αποτελεί μια άλλη διαφορετική διαδρομή (π.χ. η διαδρομή 2 περιλαμβάνει και την διαδρομή 1) τότε αυτή απορρίπτεται και προτιμάται η αμέσως επόμενη. Συνεπώς η διαδρομή 2 απορρίπτεται και επιλέγεται η διαδρομή 4.
- *Διαδρομή ελάχιστης ενέργειας :* Είναι αυτή η διαδρομή της οποίας η χρήση προκειμένου να αποσταλούν τα δεδομένα από την πηγή στον προορισμό καταναλώνει την λιγότερη ενέργεια. Στο σχήμα η διαδρομή αυτή είναι η 1.
- *Διαδρομή των ελάχιστων αλμάτων (hop) :* Είναι αυτή που έχει τα λιγότερα βήματα από τον κόμβο πηγή προς τον κόμβο προορισμό. Στο σχήμα η διαδρομή αυτή είναι η 3. αν οι κόμβοι εκπέμπουν όλοι με την ίδια ενέργεια τότε η διαδρομή αυτή είναι ίση με την διαδρομή ελάχιστης ενέργειας.
- *Διαδρομή μέγιστης ελάχιστης Διαθέσιμης Ενέργειας :* Προτιμάται η διαδρομή κατά μήκος της οποίας η ελάχιστη ΔΕ είναι μεγαλύτερη από την ελάχιστη ΔΕ των κόμβων στις άλλες διαδρομές. Στο σχήμα αυτή η διαδρομή είναι η 3. Με αυτό τον τρόπο επιλογής αποκλείεται να επιλεγεί νωρίς κάποια διαδρομή της οποίας οι κόμβοι μπορεί να έχουν χαμηλότερη ΔΕ από ότι σε άλλες διαδρομές.

Τα πρωτόκολλα δρομολόγησης διαχωρίζονται σε κατηγορίες ανάλογα με κάποιες παραμέτρους :

<u>Ανάλογα με τον τρόπο δρομολόγησης</u> :

1. Προδραστική δρομολόγηση (proactive routing): Το επίπεδο δικτύου ανανεώνει όλες τις διαδρομές περιοδικά και έχει με αυτόν τον τρόπο μια ενημερωμένη εικόνα του δικτύου και των καλύτερων διαδρομών.
2. Αντιδραστική δρομολόγηση (Reactive routing): Το δίκτυο βρίσκει την ζητούμενη διαδρομή μόνο όταν την χρειάζεται. Έτσι δεν δημιουργείται επιπλέον κίνηση όταν αλλάζει το δίκτυο αλλά για κάθε δεδομένο που μετακινείται υπάρχει μεγαλύτερη επικεφαλίδα.

3. Υβριδική δρομολόγηση : Εκμεταλλευόμενοι τα χαρακτηριστικά των δικτύων αισθητήρων χρησιμοποιούμε άλλους τρόπους δρομολόγησης που βασίζονται στην διάδοση ερωτημάτων είτε βασιζόμενα σε πληροφορίες περιεχομένου είτε σε πληροφορία θέσης.

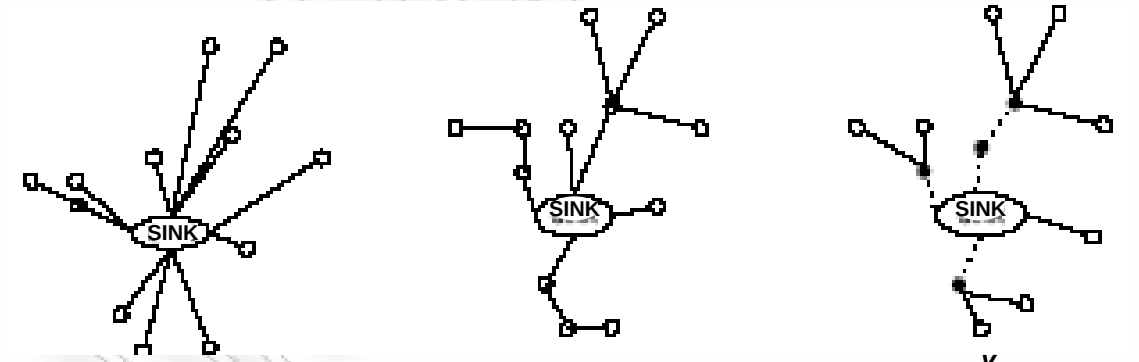
Ανάλογα με τον γνώση της θέσης :

1. Δρομολόγηση με γνώση της θέσης.

2. Δρομολόγηση χωρίς γνώση της θέσης.

Ανάλογα με το τρόπο συμμετοχής των κόμβων :

1. Άμεση επικοινωνία (direct communication): η οποία δεν είναι εφικτή μιας και η απαιτήσεις σε ενέργεια αυξάνουν με την έκταση του δικτύου.(Εικόνα 6α)
2. Επίπεδη δρομολόγηση (flat routing): στην οποία χρησιμοποιείται επικοινωνία μεταξύ γειτονικών κόμβων για την διασπορά της πληροφορίας. Οι κόμβοι κοντά στον κόμβο δεξαμενή (sink) έχουν μεγάλη απαίτηση σε ενέργεια αφού διακινούν όλη την πληροφορία μεταξύ του δικτύου και του κόμβου δεξαμενή (sink). (Εικόνα 6β)
3. Πρωτόκολλα δρομολόγησης με ομάδες (clustering routing protocols) :Είναι τα καταλληλότερα για τα δίκτυα αισθητήρων αφού έχουν αρκετά πλεονεκτήματα : α) Οι κόμβοι χρειάζεται να αποθηκεύουν πληροφορία μόνο για τον επικεφαλής της ομάδας. Συνεπώς είναι επεκτάσιμα! β) Τα δρομολόγια ανακαλύπτονται και συντηρούνται εύκολα, και γ) έχουν μικρή κατανάλωση ενέργειας αφού τα δεδομένα συγκεντρώνονται στους επικεφαλείς των ομάδων όπου και γίνεται η επεξεργασία τους. (Εικόνα 6γ)



α. *β.* *γ.*

Εικόνα 6: Πρωτόκολλα δρομολόγησης ανάλογα με τον τρόπο συμμετοχής των κόμβων. [3]

Όλα τα πρωτόκολλα δρομολόγησης μπορούν να ομαδοποιηθούν σε :

1. Δεδομένο-κεντρικά.
2. Ιεραρχικά.

3. Βασισμένα στην θέση των κόμβων.

Υπάρχουν και δύο άλλες κατηγορίες που βασίζονται στην ροή των δεδομένων στο δίκτυο και στην παρεχόμενη ποιότητα υπηρεσίας.

Τα δεδομένο-κεντρικά πρωτόκολλα βασίζονται σε ερωτήματα (παρόμοια με αυτά μιας βάσης δεδομένων) και εξαρτώνται από την ονομασία επιθυμητών δεδομένων, το οποίο βοηθά να εξαλειφθούν οι πλεονάζουσες εκπομπές. Τα ιεραρχικά πρωτόκολλα στοχεύουν στην ομαδοποίηση των κόμβων έτσι ώστε οι επικεφαλείς κόμβοι των ομάδων να εκτελούν κάποιο συγκερασμό και μείωση των δεδομένων προς εκπομπή με σκοπό την εξοικονόμηση ενέργειας. Τα πρωτόκολλα που βασίζονται στην γνώση της θέσης των κόμβων, χρησιμοποιούν αυτή την πληροφορία για να αναμεταδώσουν τα δεδομένα σε επιθυμητές περιοχές, αντί σε όλο το δίκτυο.

4.3.1 Δεδομένο-κεντρικά πρωτόκολλα δρομολόγησης.

Σ' αυτή την κατηγορία δρομολόγησης, ο κόμβος συγκεντρωτής (sink) στέλνει ερωτήματα σε συγκεκριμένες περιοχές και περιμένει τα δεδομένα από τους αισθητήρες που βρίσκονται σε αυτές τις περιοχές. Αφού τα δεδομένα ζητούνται μέσω ερωτημάτων, είναι απαραίτητο να υπάρχει ονοματοδοσία βασισμένη σε χαρακτηριστικά προκειμένου να καθοριστούν λεπτομερώς οι ιδιότητες των δεδομένων.

Flooding and Gossiping.

Δύο πολύ αντιπροσωπευτικά παραδείγματα αυτής της κατηγορίας είναι το Flooding και το Gossiping. Το flooding έχει πολύ εύκολη υλοποίηση αλλά έχει 3 μειονεκτήματα. Πρώτον, έχουμε μεγάλη συγκέντρωση από αντίγραφα μηνυμάτων που στέλνονται από τον ίδιο κόμβο (implosion). Δεύτερο υπάρχει επικάλυψη (overlap) όταν δύο ή περισσότεροι κόμβοι που ανιχνεύουν το ίδιο γεγονός στέλνουν παρόμοια πακέτα πληροφορίας στο ίδιο γείτονα, και τρίτο καταναλώνει πολύ ενέργεια χωρίς να λαμβάνει υπόψη τους ενεργειακούς περιορισμούς.

Το Gossiping λειτουργεί όπως το flooding αλλά δημιουργήθηκε για να διορθώσει τα προβλήματα του τελευταίου. Έτσι αποφεύγει το πρόβλημα της μεγάλης συγκέντρωσης (implosion) γιατί ο κάθε κόμβος αντί να στέλνει το πακέτο σε όλους το στέλνει σε ένα τυχαίο γείτονα που κι' αυτός με τη σειρά του σε άλλον κ.ο.κ. Το μειονέκτημα του πρωτοκόλλου είναι ότι έχει προβλήματα καθυστερήσεων στην διάδοση της πληροφορίας.

SPIN (Sensor Protocols for Information via Negotiation).

Το SPIN δημιουργήθηκε για να διορθώσει τα προβλήματα του flooding και πετυχαίνει αποτελεσματική χρήση της ενέργειας.

Πετυχαίνει 3.5 φορές μικρότερη απώλεια ενέργειας σε σχέση με το flooding και μείωση στο μισό (50%) των περιττών δεδομένων (μικρότερος πλεονασμός).Οι οποιεσδήποτε αλλαγές στην τοπολογία παραμένουν σε τοπικό επίπεδο αφού ο κάθε κόμβος είναι απαραίτητο να γνωρίζει τους γείτονές του και μόνο.

Ο μηχανισμός που χρησιμοποιεί το SPIN (διαφήμιση των υπαρχόντων δεδομένων αλλά και των αναγκών σε νέα δεδομένα), δεν μπορεί να εγγυηθεί την παράδοση των δεδομένων, γιατί μεταξύ των κόμβων που ενδιαφέρονται για συγκεκριμένη πληροφορία και αυτών που την διαθέτουν μπορεί να παρεμβάλλονται κόμβοι οι οποίοι δεν ενδιαφέρονται για την συγκεκριμένη πληροφορία. Γενικώς δεν ενδείκνυται η χρήση του σε κρίσιμες εφαρμογές όπως, σύστημα συναγερμού για ανίχνευση παραβιάσεων όπου απαιτείται αξιόπιστη παράδοση των δεδομένων σε τακτά χρονικά διαστήματα.

Directed Diffusion.

Σε σχέση με το SPIN που οι κόμβοι διαφημίζουν για διαθέσιμα δεδομένα επιτρέποντας στους ενδιαφερόμενους κόμβους να ζητήσουν τα δεδομένα αυτά στο Directed Diffusion ο κεντρικός κόμβος συγκεντρωτής (sink) ρωτά τους αισθητήριους κόμβους αν συγκεκριμένα δεδομένα είναι διαθέσιμα.

Όλες οι επικοινωνίες είναι γείτονα προς γείτονα χωρίς να την ανάγκη για ύπαρξη μηχανισμού που θα απευθύνεται σε συγκεκριμένο κόμβο. Ο κάθε κόμβος, εκτός από την εργασία της αίσθησης (sensing), μπορεί να κάνει συγκερασμό (aggregate) των δεδομένων και προσωρινή αποθήκευση (caching). Η προσωρινή αποθήκευση (caching) προσφέρει μεγάλο πλεονέκτημα λόγω της αποτελεσματικής χρήσης της ενέργειας και της μικρής καθυστέρησης. Επίσης η κατανάλωση της ενέργειας μειώνεται αφού η πληροφορία δίνεται κατ' απαίτηση και δεν υπάρχει ανάγκη για την διατήρηση γνώσης για την τοπολογία του δικτύου.

Το πρωτόκολλο αυτό παρουσιάζει και κάποια μειονεκτήματα όπως το γεγονός ότι από τη φύση του δεν μπορεί να χρησιμοποιηθεί σε όλες τις εφαρμογές γιατί βασίζεται σε ένα μοντέλο παράδοσης της πληροφορίας κατ' απαίτηση. Συνεπώς δεν είναι κατάλληλο για συνεχή παρακολούθηση του περιβάλλοντος. Άλλο

πρόβλημα είναι η ονοματοδοσία των ερωτημάτων αφού εξαρτάται από την εφαρμογή και πρέπει κάθε φορά τα ερωτήματα να καθορίζονται εξ' αρχής. Τέλος η διαδικασία του ταιριάσματος δηλαδή ποιες απαντήσεις απευθύνονται σε ποια ερωτήματα, μπορεί να απαιτεί επιπλέον χώρο στην επικεφαλίδα των ερωτημάτων.

Energy Aware Routing.

Το πρωτόκολλο Energy Aware Routing είναι παρόμοιο με το Directed Diffusion γιατί ανακαλύπτονται πολλαπλά μονοπάτια από την πηγή μέχρι τον προορισμό (sink). Τα μονοπάτια αυτά επιλέγονται από μια συνάρτηση πιθανοτήτων, η οποία εξαρτάται από την ενεργειακή κατανάλωση στον κάθε κόμβο. Το πρωτόκολλο αυτό θεωρεί ότι χρησιμοποιώντας συνέχεια το μονοπάτι με το μικρότερο κόστος συνεχώς θα έχει ως αποτέλεσμα να εξαντληθεί η ενέργεια των κόμβων. Για να αποφευχθεί αυτό, το κάθε ένα από τα πολλαπλά μονοπάτια χρησιμοποιούνται με μια συγκεκριμένη πιθανότητα έτσι ώστε να αυξηθεί η ζωή του δικτύου.

Το Directed Diffusion στέλνει τα δεδομένα μέσα από πολλαπλά μονοπάτια, και ένα από αυτά εξαναγκάζεται να στέλνει δεδομένα σε υψηλότερους ρυθμούς. Σε αντίθεση το πρωτόκολλο αυτό διαλέγει ένα τυχαίο μονοπάτι προκειμένου να εξοικονομήσει ενέργεια. Αποτέλεσμα να υπάρχει 21.5% οικονομία ενέργειας παραπάνω από ότι το Directed Diffusion και 44% αύξηση στην ζωή του δικτύου.

Αυτή η μέθοδος όμως έχει και μειονεκτήματα, όπως είναι η ικανότητα επαναφοράς (recover) μετά από κάποιο σφάλμα ενός κόμβου, αφού χρησιμοποιείται ένα μονοπάτι σε αντίθεση με το Directed Diffusion.

Rumor Routing.

Το Rumor Routing είναι μια παραλλαγή του Directed Diffusion για εφαρμογή σε περιβάλλοντα όπου δεν εφαρμόζονται κριτήρια γεωγραφικής δρομολόγησης. Όταν ένας κόμβος ανιχνεύσει ένα συμβάν στέλνει στο δίκτυο ένα πακέτο πληροφορίας (agent), που αφορά αυτό το συμβάν, προς τους απομακρυσμένους κόμβους. Με αυτόν τον τρόπο αποφεύγεται το κόστος της «πλημμύρας» (flooding) προς όλο το δίκτυο. Αποτελέσματα προσομοίωσης έδειξαν ότι σε σχέση με το flooding έχουμε σημαντική εξοικονόμηση ενέργειας και επίσης μπορεί να αντιμετωπιστεί αποτελεσματικά η αστοχία (καταστροφή ή βλάβη) κάποιων κόμβων.

Η διαφορά του από το Directed Diffusion είναι ότι το πρωτόκολλο Rumor Routing διατηρεί ένα μονοπάτι μεταξύ πηγής και προορισμού σε αντίθεση με το πρώτο όπου τα δεδομένα μπορούν να σταλούν μέσω διαφορετικών μονοπατιών σε χαμηλούς ρυθμούς μετάδοσης.

Το πρωτόκολλο αυτό λειτουργεί αποτελεσματικά όταν το πλήθος των συμβάντων είναι μικρό. Αν έχουμε μεγάλο πλήθος συμβάντων και δεν υπάρχει το απαιτούμενο ενδιαφέρον από μέρους της πηγής, τότε το κόστος διατήρησης των πακέτων πληροφορίας (agent) καθώς και των πινάκων με τα δεδομένα είναι μεγάλο, χωρίς να προσφέρει αντίστοιχο όφελος.

Gradient Based Routing.

Το Gradient Based Routing αποτελεί μια μικρή παραλλαγή του Directed Diffusion. Η ιδέα είναι ο κάθε κόμβος να κρατάει τον αριθμό των αλμάτων προς τους γείτονές του κατά την φάση της διάχυσης ενδιαφέροντος. Έτσι ο κάθε κόμβος γνωρίζει πόσο απέχει από τον κόμβο δεξαμενή (sink) σε άλματα το οποίο ονομάζεται και ύψος του κόμβου. Η διαφορά ύψους ενός κόμβου από ένα γειτονικό του κόμβο ονομάζεται κλίση του κόμβου. Τα πακέτα προωθούνται σε μια ζεύξη με τη μεγαλύτερη κλίση. Χρησιμοποιώντας τεχνικές όπως ο συγκερασμός των δεδομένων και η εξάπλωση τους πετυχαίνει να εξισορροπήσει την κίνηση ομοιόμορφα. Όσον αφορά την εξάπλωση των δεδομένων χρησιμοποιούνται 3 τεχνικές : *α)* *Stochastic Scheme* όταν υπάρχουν περισσότερες ζεύξεις με την ίδια κλίση, ο κόμβος επιλέγει μία τυχαία. *β)* *Energy-Based Routing* όταν η ενέργεια ενός κόμβου μειωθεί κάτω από μια τιμή , τότε αυτός αυξάνει την κλίση του έτσι ώστε να αποτρέψει άλλους κόμβους στο να τον επιλέξουν για αποστολή δεδομένων. και *γ)* *Stream-Based scheme* όπου η ιδέα είναι να εκτρέπονται οι νέες ροές δεδομένων μακριά από κόμβους που διακινούν κάποια ροή δεδομένων.

CADR (Constrained Anisotropic Diffusion Routing).

Το CADR αποτελεί και αυτό μια παραλλαγή του Directed Diffusion. Με την δρομολόγηση αυτή προτείνονται 2 τεχνικές. Η μία είναι ερωτήσεις κατευθυνόμενες προς τους αισθητήρες με βάση την πληροφορία και η άλλη είναι η περιορισμένη μη ισοτροπική διάχυση. Η όλη ιδέα αφορά στην ερώτηση των αισθητήρων και στην δρομολόγηση των δεδομένων στο δίκτυο προκειμένου να έχουμε κέρδος στην πληροφορία και ταυτόχρονα μείωση των καθυστερήσεων και του χρησιμοποιούμενου εύρους συχνοτήτων. Αυτή είναι και η μεγάλη

διαφορά από το Directed Diffusion. Κάθε κόμβος αξιολογεί την πληροφορία και το κόστος διάδοσής της βασιζόμενος σε πληροφορίες που διαθέτει τοπικά, αλλά και στις απαιτήσεις των χρηστών. Επίσης χρησιμοποιείται και μια άλλη συμπληρωματική τεχνική, σύμφωνα με την οποία ο κάθε κόμβος μπορεί να επιλέγει ποιος κόμβος μπορεί να παρέχει την ζητούμενη πληροφορία με το λιγότερο ενεργειακό κόστος.

4.3.2 Ιεραρχικά πρωτόκολλα δρομολόγησης.

LEACH Low Energy Adaptive Clustering Hierarchy.

Το LEACH είναι ένα από τα πιο διάσημα ιεραρχικά πρωτόκολλα δρομολόγησης για δίκτυα αισθητήρων. Η λειτουργία του βασίζεται στην δημιουργία ομάδων (clusters) αισθητήριων κόμβων, που βασίζονται στην ένταση του λαμβανόμενου σήματος και στην χρήση των επικεφαλών των ομάδων σαν δρομολογητών μεταξύ των κόμβων και του κόμβου δεξαμενή (sink). Με αυτόν τον τρόπο γίνεται εξοικονόμηση ενέργειας, μιας και εκπομπή δεδομένων προς τον κόμβο δεξαμενή (sink) γίνεται μόνο από τους επικεφαλείς κόμβους (cluster heads) και όχι από όλους τους κόμβους. Ο βέλτιστος (optimal) αριθμός των επικεφαλείς κόμβων είναι το 5% των συνολικών κόμβων. Η επεξεργασία των δεδομένων (συγκερασμός) γίνεται στους κόμβους επικεφαλείς των ομάδων. Η λειτουργία του πρωτοκόλλου έχει 2 φάσεις: α) την φάση εγκατάστασης και β) την σταθερή φάση. Κατά την α) φάση επιλέγονται οι επικεφαλείς των κόμβων. Προκειμένου να γίνει εξισορρόπηση της απώλειας ενέργειας μεταξύ των κόμβων, οι επικεφαλείς κόμβοι αλλάζουν τυχαία στο χρόνο. Ο τρόπος της αλλαγής γίνεται με τον κόμβο να επιλέγει ένα τυχαίο αριθμό μεταξύ του 0 και του 1. Ο κόμβος γίνεται επικεφαλής μιας ομάδας, αν ο αριθμός που επέλεξε είναι μικρότερος από την τιμή κατωφλίου.

Μόλις επιλεγούν οι επικεφαλείς των ομάδων, στέλνουν μηνύματα (advertise) προς όλους τους κόμβους του δικτύου ότι είναι επικεφαλείς κόμβοι πλέον. Όταν οι αισθητήριοι κόμβοι λαμβάνουν αυτό το μήνυμα, αποφασίζουν για την ομάδα στην οποία θέλουν να ανήκουν βασιζόμενοι στην ένταση του σήματος του μηνύματος που έλαβαν. Κατόπιν ειδοποιούν τον αντίστοιχο επικεφαλής ομάδας ότι θα ανήκουν στην ομάδα του. Τέλος, ο επικεφαλής της ομάδας ορίζει τα χρονικά διαστήματα κατά τα οποία οι αισθητήριοι κόμβοι μπορούν να στέλνουν δεδομένα, βασιζόμενος σε μια προσέγγιση TDMA.

Κατά τη διάρκεια της σταθερής φάσης, οι αισθητήριοι κόμβοι μπορούν να λαμβάνουν δεδομένα από το περιβάλλον και να τα εκπέμπουν προς τους επικεφαλείς των ομάδων. Το πρωτόκολλο LEACH επιτυγχάνει μείωση ως και 7 φορές στην κατανάλωση της ενέργειας σε σχέση με την απ' ευθείας μετάδοση και μείωση 4-8 φορές σε σχέση με την μέθοδο μετάδοσης της μικρότερης ενέργειας.

Το LEACH χρησιμοποιεί δρομολόγηση single-hop όπου ο κάθε κόμβος μπορεί να εκπέμψει κατευθείαν στον επικεφαλής της ομάδας και στον κόμβο sink. Το τελευταίο αποτελεί και μειονέκτημα διότι δεν μπορεί να εφαρμοστεί σε δίκτυα που εγκαθίστανται σε μεγάλες περιοχές. Ένα άλλο πρόβλημα αποτελεί και η δυναμική αλλαγή των ομάδων που επιφέρει επιπλέον κατανάλωση ενέργειας λόγω αλλαγής του επικεφαλής, της επιπλέον διαφήμισης κ.α.

PEGASIS & Ιεραρχικό PEGASIS.

- *PEGASIS Power Efficient Gathering in Sensor Information Systems*

Πρόκειται για μια βελτίωση του πρωτοκόλλου LEACH. Η διαφορά του με το πρωτόκολλο LEACH είναι η χρήση δρομολόγησης multi-hop με την δημιουργία αλυσίδων και με την επιλογή ενός μόνο κόμβου που θα εκπέμπει προς το σταθμό βάσης αντί της χρήσης πολλαπλών κόμβων. Τα συλλεγμένα δεδομένα κινούνται από κόμβο σε κόμβο, αθροίζονται (aggregated) και τελικώς στέλνονται προς το σταθμό βάσης (sink). Η κατασκευή της αλυσίδας γίνεται με άπληστο τρόπο (greedy way).

Πλεονέκτημα του είναι ότι έχει αποδειχτεί ότι ξεπερνάει σε απόδοση το LEACH περίπου 100-300% για διάφορα μεγέθη δικτύου και διάφορες τοπολογίες. Μειονέκτημα του είναι ότι επιφέρει μεγάλη καθυστέρηση για απομακρυσμένους κόμβους στην αλυσίδα. Επίσης ο μοναδικός αρχηγός στην αλυσίδα μπορεί να γίνει μία στενωπός στην οποία θα συνωστίζεται όλη η κίνηση του δικτύου και θα επιφέρει μεγάλη καθυστέρηση στην μετάδοση των δεδομένων.

- *Ιεραρχικό PEGASIS*

Είναι μια επέκταση του απλού PEGASIS. Προκειμένου να μειωθεί η καθυστέρηση και να προταθεί μια λύση στην συγκέντρωση των δεδομένων. Προκειμένου να αποφύγει συγκρούσεις και πιθανές παρεμβολές από κόμβους που εκπέμπουν σε κοντινή απόσταση έχουν ερευνηθεί 2 τεχνικές : 1) κωδικοποίηση σήματος π.χ. CDMA και 2) να μπορούν να εκπέμπουν ταυτόχρονα μόνο οι σε απόσταση κόμβοι.

Το Ιεραρχικό PEGASIS έχει αποδειχτεί ότι λειτουργεί καλύτερα από ότι η απλή έκδοση κατά ένα παράγοντα 60%. **TEEN & APTEEN.**

- *TEEN Threshold Sensitive Energy Efficient Sensor Network protocol*

Σχεδιάστηκε για να ανταποκρίνεται σε ξαφνικές αλλαγές στα χαρακτηριστικά των παρατηρούμενων γεγονότων, όπως είναι η θερμοκρασία. Είναι σημαντικό να υπάρχει άμεση ανταπόκριση για εφαρμογές πραγματικού χρόνου. Η αρχιτεκτονική του δικτύου αισθητήρων βασίζεται σε μια ιεραρχική ομαδοποίηση όπου οι κοντινότεροι κόμβοι δημιουργούν ομάδες (clusters) και αυτή η λειτουργία συνεχίζεται και σε 2^{o} επίπεδο μέχρις ότου φθάσουμε στον σταθμό βάσης (sink node). Χρησιμοποιεί 2 τιμές κατωφλίου την σκληρή και την μαλακή (hard and soft threshold) με αποτέλεσμα να έχουμε συνεχή παρακολούθηση του γεγονότος αλλά και μείωση των εκπομπών και διατήρηση της ενέργειας.

- *APTEEN AdaPtive TEEN*.

Πρόκειται για μια επέκταση του TEEN που στοχεύει στο να μπορεί να λειτουργήσει και για εφαρμογές λήψης δεδομένων περιοδικά και να αντιδρά σε χρήσιμα χρονικά γεγονότα. Η χρησιμοποιούμενη αρχιτεκτονική είναι ίδια με του απλού TEEN. Σε σχέση με το απλό TEEN, υποστηρίζει 3 διαφορετικούς τύπους ερωτημάτων 1) ιστορικά για να αναλύσει περασμένες τιμές δεδομένων. 2) Μιας χρονικής στιγμής για να λάβει άποψη των παρατηρούμενων γεγονότων εκείνη τη στιγμή. 3) Συνεχόμενων για να παρακολουθήσει ένα γεγονός για συγκεκριμένη χρονική περίοδο.

4.3.3 Πρωτόκολλα δρομολόγησης βασισμένα στην θέση.

MECN Minimum Energy Communication Network

Φτιάχνει και διατηρεί ένα δίκτυο ελάχιστης ενέργειας για ασύρματα δίκτυα χρησιμοποιώντας ένα χαμηλής ισχύος GPS. Το MECN βρίσκει μια περιοχή αναμετάδοσης (relay region) για κάθε κόμβο. Αυτή η αποτελείται από κόμβους στην γύρω από τον κόμβο περιοχή, μέσω των οποίων η μετάδοση είναι πιο αποτελεσματική ενεργειακά, από ότι η απ' ευθείας μετάδοση. Στη συνέχεια δημιουργείται η «εμβέλεια» του κόμβου i από τις ενώσεις όλων των περιοχών αναμετάδοσης τις οποίες μπορεί να φθάσει ο κόμβος i. Η κύρια ιδέα του MECN είναι η εύρεση ενός υπό-δικτύου, το οποίο έχει μικρότερο αριθμό κόμβων και

απαιτεί λιγότερη ενέργεια για εκπομπή μεταξύ δύο οποιοδήποτε κόμβων. Με αυτόν τον τρόπο κατασκευάζονται μονοπάτια ελάχιστης ενέργειας (global minimum paths), χωρίς να λαμβάνει υπόψη όλους τους κόμβους του δικτύου. Αυτό είναι δυνατόν χρησιμοποιώντας μια τοπική έρευνα για κάθε κόμβο λαμβάνοντας υπόψη την περιοχή αναμετάδοσης του. Το MECN είναι αυτορρυθμιζόμενο και έτσι μπορεί δυναμικά να προσαρμοστεί σε αποτυχίες (αστοχίες) κάποιων κόμβων ή στην εξάπλωση νέων κόμβων.

SMECN Small MECN

Είναι μια επέκταση του MECN. Στο MECN δεν είναι δυνατόν κάθε κόμβος να εκπέμψει σε άλλον κόμβο κάθε στιγμή. Το SMECN υποθέτει ότι μπορεί να υπάρχουν πιθανά εμπόδια μεταξύ των κόμβων αλλά ότι το δίκτυο παραμένει πλήρως συνδεδεμένο. Το υποδίκτυο που κατασκευάζεται από το SMECN είναι μικρότερο από ότι στο MECN με την προϋπόθεση ότι μια καθολική εκπομπή είναι ικανή να φτάσει προς όλους τους κόμβους σε μια κυκλική περιοχή γύρω από τον κόμβο που εκπέμπει προς όλους. Αποτέλεσμα είναι ο αριθμός των αλμάτων για επικοινωνία να είναι μειωμένος. Αποτελέσματα προσομοίωσης έδειξαν ότι το SMECN χρησιμοποιεί λιγότερη ενέργεια από το MECN και το κόστος συντήρησης των δεσμών είναι μικρότερο. Το μειονέκτημά του είναι ότι το να βρεις ένα υποδίκτυο με μικρότερο αριθμό άκρων εισάγει επιπλέον κόστος στον αλγόριθμο.

GAF Geographical Adaptive Fidelity.

Είναι ένας αλγόριθμός δρομολόγησης που λαμβάνει υπόψη την καταναλισκόμενη ενέργεια και την θέση που βρίσκεται ο αισθητήρας, ο οποίος σχεδιάστηκε αρχικά για κινούμενα ad-hoc δίκτυα, αλλά μπορεί να εφαρμοστεί και σε δίκτυα αισθητήρων. Το GAF διατηρεί την ενέργεια θέτοντας εκτός λειτουργίας τους μη αναγκαίους κόμβους του δικτύου χωρίς όμως να επηρεάζει το επίπεδο πιστότητας της δρομολόγησης. Δημιουργεί ένα εικονικό πλέγμα της καλυπτόμενης περιοχής. Οι κόμβοι που ανήκουν στο ίδιο σημείο στο πλέγμα θεωρούνται ισοδύναμοι χρησιμοποιώντας ως μέτρο την καταναλισκόμενη ενέργεια για την δρομολόγηση ενός πακέτου. Έτσι μόνο ένας κόμβος από όλους μένει ενεργός σε κάθε σημείο του πλέγματος, ενώ όλοι οι υπόλοιποι τίθενται εκτός λειτουργίας (κατάσταση ύπνου).

GEAR Geographically and Energy Aware Routing.

Το πρωτόκολλο αυτό χρησιμοποιεί πληροφορίες ενέργειας και θέσης των γειτονικών κόμβων για να δρομολογήσει τα πακέτα προς την περιοχή του στόχου. Σκοπός είναι να μειωθεί ο αριθμός των ενδιαφερόμενων κόμβων όπως συμβαίνει με την δρομολόγηση Directed Diffusion, χρησιμοποιώντας μια μόνο περιοχή για αποστολή των μηνυμάτων , από το να στέλνει το ενδιαφέρον σε όλο το δίκτυο. Το GEAR με αυτόν τον τρόπο συμπληρώνει το πρωτόκολλο Directed Diffusion καταναλώνοντας λιγότερη ενέργεια.

Τεχνικές Εύρεσης Θέσης.

Οι τρεις βασικές τεχνικές για εύρεση της θέσης είναι ο τριγωνισμός (triangulation), η ανάλυση του σκηνικού (scene analysis) και η εγγύτητα (proximity).

- Τριγωνισμός (Triangulation).

Η τεχνική αυτή χρησιμοποιεί τις γεωμετρικές ιδιότητες των τριγώνων για να υπολογίσει τις θέσεις των αντικειμένων. Οι δύο μεγάλες κατηγορίες τριγωνισμού είναι το lateration που χρησιμοποιεί μέτρηση αποστάσεων, και γωνιακή θέση (angulation) που χρησιμοποιεί μέτρηση γωνιών ή αζιμούθιων (bearing measurement).

Στο Lateration προκειμένου να βρεθεί η θέση ενός αντικειμένου πρέπει να μετρηθούν οι αποστάσεις από πολλαπλά γνωστά σημεία ή σημεία αναφοράς. Προκειμένου να βρεθεί η θέση σε 2 διαστάσεις απαιτείται η μέτρηση αποστάσεων από 3 μη ομοαξονικά σημεία, ενώ για εύρεση θέσης σε 3 διαστάσεις απαιτούνται 4 μη ομοεπίπεδα σημεία. Είναι δυνατόν αυτά τα σημεία να μειωθούν με τη βοήθεια ενός συστήματος εύρεσης θέσης χρησιμοποιώντας χαρακτηριστικά του περιβάλλοντος.

Γενικά χρησιμοποιούνται 3 τεχνικές για την μέτρηση αποστάσεων:

α) Απευθείας. Αυτό γίνεται με τη χρήση μια φυσικής πράξης ή κίνησης. Για παράδειγμα ένα ρομπότ μπορεί να μετρήσει μια απόσταση εκτείνοντας ένα μεταλλικό άκρο μέχρι να έχει φυσική επαφή με το προς μέτρηση αντικείμενο.

β) Χρόνος «Πτήσης» (Time-of-Flight). Είναι τεχνική που βασίζεται στο χρόνο και μετρά το χρόνο άφιξης ή τη διαφορά μεταξύ των χρόνων άφιξης σημάτων. Τη χρήση της τεχνικής αυτής τη χρησιμοποιούν αρκετά συστήματα όπως το GPS.

γ) Εξασθένιση (Attenuation). Η τεχνική αυτή βασίζεται στο γεγονός ότι η ένταση ενός εκπεμπόμενου σήματος μειώνεται καθώς η απόσταση από το σημείο εκπομπής αυξάνεται.

Το Angulation αποτελεί μια παρόμοια τεχνική με την Lateration μόνο που αντί για αποστάσεις μετρώνται γωνίες, προκειμένου να βρεθεί η θέση ενός αντικειμένου. Συνήθως απαιτείται η μέτρηση 2 γωνιών και μιας απόστασης για εύρεση θέσης σε 2 διαστάσεις ενώ για τις 3 διαστάσεις χρειάζεται επιπρόσθετα και η μέτρηση ενός αζιμούθιου. Συνήθως, σε αυτή την τεχνική, χρησιμοποιείται και ένα άνυσμα αναφοράς (π.χ. ο μαγνητικός βορράς) ως η θέση 0^o. Στην βιβλιογραφία η τεχνική ονομάζεται και angle of arrival (AoA).

- Ανάλυση του πεδίου (Scene Analysis).

Σύμφωνα με την τεχνική αυτή χρησιμοποιώντας τα χαρακτηριστικά ενός περιβάλλοντος που παρατηρήθηκαν από κάποιο κατάλληλα επιλεγμένο σημείο, είναι δυνατόν να εξαχθούν συμπεράσματα για την θέση του παρατηρητή ή των αντικειμένων στο περιβάλλον. Συνήθως τα παρατηρούμενα περιβάλλοντα απλοποιούνται προκειμένου να είναι εύκολο να αναπαρασταθούν και να συγκριθούν (Εικόνα 7)**Error! Reference source not found.**.

Στα στατικά περιβάλλοντα η ανάλυση του πεδίου βασίζεται σε προσχεδιασμένα σύνολα αντικειμένων που έχουν παρατηρηθεί και έχουν αναπαρασταθεί σε τοποθεσίες αντικειμένων. Σε αντίθεση στα δυναμικά περιβάλλοντα η ανάλυση του πεδίου παρακολουθεί τις διαφορές μεταξύ των διαδοχικών σκηνών προκειμένου να κάνει εκτίμηση της τοποθεσίας. Οι διαφορές στο πεδίο ανταποκρίνονται στον τρόπο που ο παρατηρητής βλέπει το περιβάλλον καθώς αυτός κινείται. Αν υπάρχουν κάποια χαρακτηριστικά σημεία των οποίων οι θέσεις είναι γνωστές, ο παρατηρητής μπορεί να υπολογίσει τη θέση του σε σχέση με αυτά.

Το πλεονέκτημα της μεθόδου είναι ότι η θέση των αντικειμένων μπορεί να συναχθεί χρησιμοποιώντας παθητική παρατήρηση και τα χαρακτηριστικά του περιβάλλοντος, χωρίς να είναι αναγκαία η χρήση γεωμετρικών αποστάσεων και γωνιών. Σε περίπτωση που γίνει κάποια αλλαγή στα χαρακτηριστικά του περιβάλλοντος συνήθως απαιτείται η επανάκτηση κάποιων σημείων του συνόλου ή η ανάκτηση ενός καινούριου συνόλου σημείων. Ένα σύστημα που χρησιμοποιεί την τεχνική αυτή είναι το RADAR.

Εικόνα 7: Σχήμα του ορίζοντα όπως φαίνεται από μια κάμερα. [29]

- Εγγύτητα (Proximity).

Η συγκεκριμένη τεχνική απαιτεί την ανίχνευση της παρουσίας ενός αντικειμένου με τη χρήση ενός φυσικού φαινόμενου περιορισμένης ακτίνας. Υπάρχουν 3 διαφορετικές προσεγγίσεις γι' αυτή την τεχνική :

α) Ανίχνευση φυσικής επαφής. Είναι η πιο βασική τεχνική και χρησιμοποιεί αισθητήρες πίεσης, αίσθησης και ανιχνευτές χώρου.

β) Παρακολούθηση ασύρματων κυψελωτών σημείων πρόσβασης. Συνήθως χρησιμοποιείται στα κινητά τηλέφωνα χωρίς να αποκλείει τη χρήση της από δίκτυα αισθητήρων. Ο τρόπος λειτουργίας είναι ανίχνευση της κινητής συσκευής όταν αυτή εισέρχεται στην ακτίνα κάλυψης της κυψέλης.

γ) Παρακολούθηση συστημάτων αυτόματου αριθμού αναγνώρισης (automatic ID systems). Όπως προδίδει και η ονομασία η εύρεση της θέσης χρησιμοποιεί συστήματα αυτόματης αναγνώρισης όπως τερματικά συναλλαγών με πιστωτικές κάρτες.

Συνήθως τέτοια συστήματα αναγνώρισης θέσης με χρήση της εγγύτητας μπορεί να χρειάζεται να συνδυαστούν με κάποιο σύστημα αναγνώρισης, αν δεν διαθέτουν τέτοια δυνατότητα.

4.3.4 Πρωτόκολλα δρομολόγησης βασισμένα στην ροή του δικτύου και στην ποιότητα της υπηρεσίας.

Αν και τα περισσότερα πρωτόκολλα ανταποκρίνονται στην κατάταξη των παραπάνω παραγράφων, υπάρχουν και κάποια που απαιτούν διαφορετική κατάταξη όπως η ροή του δικτύου και ποιότητα υπηρεσίας.

Maximum lifetime energy routing.

Είναι πρωτόκολλο δρομολόγησης που αφορά στην ροή του δικτύου. Ο κύριος σκοπός αυτής της προσέγγισης είναι να μεγιστοποιηθεί η διάρκεια ζωής του δικτύου με τον προσεκτικό σχεδιασμό του κόστους ζεύξης σαν συνάρτηση της εναπομένουσας ενέργειας του κόμβου και της απαιτούμενης ενέργειας εκπομπής χρησιμοποιώντας αυτή τη ζεύξη. Από εκεί προέρχεται και το όνομα του πρωτοκόλλου

Maximum lifetime data gathering.

Ένας άλλος προτεινόμενος αλγόριθμος αναφέρεται στη μέγιστη ζωή του δικτύου για τη συλλογή δεδομένων. Η ζωή «Τ» του συστήματος ορίζεται ως ο αριθμός των γύρων ή των περιοδικών λήψεων δεδομένων από τους αισθητήρες μέχρις ότου βγει εκτός λειτουργίας ο πρώτος αισθητήρας.

Minimum Cost Forwarding.

Το πρωτόκολλο αυτό στοχεύει στο να βρει τη διαδρομή με το ελάχιστο δυνατό κόστος σε ένα μεγάλο δίκτυο αισθητήρων. Δεν είναι ακριβώς βασισμένο στην ροή του δικτύου, αλλά επειδή τα δεδομένα ρέουν στο μονοπάτι με το μικρότερο κόστος και οι πόροι των αισθητήριων κόμβων ανανεώνονται σε κάθε ροή, μπορεί να θεωρηθεί ότι ανήκει σε αυτήν την ομάδα.

Sequential Assignment Routing SAR.

Είναι το πρώτο πρωτόκολλο για δίκτυα αισθητήρων το οποίο περιλαμβάνει την έννοια της ποιότητας υπηρεσίας (QoS), στις αποφάσεις της δρομολόγησης. Το πρωτόκολλο αυτό δημιουργεί πολλά δέντρα των οποίων η ρίζες είναι οι άμεσοι γείτονες προς τον κόμβο δεξαμενή (sink). Το κάθε δέντρο επεκτείνεται μακριά από τον κόμβο δεξαμενή (sink), αποφεύγοντας να συμπεριλάβει σε αυτό κόμβους με πολύ χαμηλή ποιότητα υπηρεσίας (QoS) ή μικρή εναπομένουσα ενέργεια. Στο τέλος αυτής της διαδικασίας οι περισσότεροι κόμβοι ανήκουν σε πολλαπλά δέντρα. Αυτό επιτρέπει στον κόμβο να επιλέγει το καταλληλότερο δέντρο προκειμένου να αναμεταδώσει προς τον κόμβο δεξαμενή (sink).

Energy-Aware QoS Routing Protocol.

Πρόκειται για ένα σχετικά νέο πρωτόκολλο το οποίο βρίσκει ένα μονοπάτι το οποίο έχει το μικρότερο κόστος και είναι και αποτελεσματικό ενεργειακά, ενώ ταυτόχρονα εγγυάται συγκεκριμένη καθυστέρηση από άκρη σε άκρη. Το κόστος της ζεύξης που χρησιμοποιείται είναι μια συνάρτηση που λαμβάνει υπόψη την εναπομένουσα ενέργεια του κόμβου, την ενέργεια εκπομπής, τον ρυθμό των λαθών και άλλες επικοινωνιακές παραμέτρους.

Προκειμένου να έχουμε ταυτόχρονα κίνηση με την καλύτερη προσπάθεια και πραγματικού χρόνου κίνηση, χρησιμοποιείται ένα μοντέλο ταξινόμησης ουράς. Το μοντέλο αυτό επιτρέπει την ταξινόμηση της κίνησης σε ροή πραγματικού και μη χρόνου.

SPEED.

Το πρωτόκολλο αυτό απαιτεί από τον κάθε κόμβο να διατηρεί πληροφορία για τους γείτονές του και χρησιμοποιεί γεωγραφική προώθηση των πακέτων για να βρει τα μονοπάτια. Επιπλέον το πρωτόκολλο αυτό προσπαθεί να επιτύχει μια σταθερή ταχύτητα για κάθε πακέτο που κινείται στο δίκτυο, έτσι ώστε η κάθε εφαρμογή να μπορεί να εκτιμήσει την από άκρη σε άκρη καθυστέρηση. Επίσης το πρωτόκολλο μπορεί να αποφύγει την συμφόρηση όταν συμβαίνει αυτό στο δίκτυο

4.4. Επίπεδο Μεταφοράς.

Το επίπεδο μεταφοράς είναι αναγκαίο να υπάρχει όταν το σύστημα πρόκειται να είναι προσβάσιμο μέσω του Διαδικτύου ή άλλων εξωτερικών δικτύων. Βέβαια αυτό είναι μάλλον σύνηθες να συμβαίνει αφού τα δίκτυα αισθητήρων εγκαθίστανται προκειμένου να παρακολουθήσουν γεγονότα και να μεταδώσουν πληροφορίες. Στη σημερινή εποχή που επικρατεί η ιδέα της «δικτύωσης παντού», μια πληροφορία η οποία δεν μπορεί να μεταδοθεί έγκαιρα σε οποιονδήποτε ενδιαφερόμενο, θεωρείται παρωχημένη και χωρίς αξία. Συνεπώς η ανάγκη σύνδεσης ενός δικτύου ασύρματων αισθητήρων με άλλα δίκτυα είναι επιβεβλημένη. Το πρωτόκολλο TCP όπως είναι σχεδιασμένο μπορεί να ταιριάξει με τα δίκτυα ασύρματων αισθητήρων, με κάποια αλλαγή όπως είναι ο τερματισμός του πρωτοκόλλου στους κόμβους «δεξαμενές» (sink nodes) όπου θα τερματίζεται και η σύνδεση TCP. Από εκεί και πέρα κάποιο ειδικό

πρωτόκολλο μεταφοράς μπορεί να αναλάβει τη διακίνηση της πληροφορίας μεταξύ των ασύρματων κόμβων και του κόμβου δεξαμενή (sink node). Αυτή η διαφοροποίηση είναι αναγκαία λόγω των χαρακτηριστικών των δικτύων αισθητήρων, καθώς και στο διαφορετικό τρόπο διευθυνσιοδότησης βασισμένο στα χαρακτηριστικά της πληροφορίας και όχι σε συγκεκριμένους αισθητήρες, όπως αναφέρθηκαν στις προηγούμενες παραγράφους.

4.5. Επίπεδο Εφαρμογής.

Το επίπεδο εφαρμογής παραμένει ένας ανεξερεύνητος τομέας για τα ασύρματα δίκτυα αισθητήρων, αν και έχουν οριστεί και προταθεί πολλές περιοχές εφαρμογής των δικτύων αυτών. Τρία πιθανά επίπεδα εφαρμογής εξετάζονται στις επόμενες παραγράφους.

4.5.1. Sensor Management Protocol (SMP).

Ο σχεδιασμός ενός πρωτόκολλο διαχείρισης επιπέδου εφαρμογής έχει αρκετά πλεονεκτήματα μιας και τα δίκτυα αισθητήρων θα συνδέονται με άλλα δίκτυα. Είναι απαραίτητο λοιπόν να υπάρχει ένα τέτοιο πρωτόκολλο που θα αναλαμβάνει να κάνει διαφανώς την εργασία της μεταφοράς των δεδομένων των χαμηλότερων επιπέδων προς το επίπεδο της εφαρμογής. Το πρωτόκολλο SMP είναι ένα πρωτόκολλο διαχείρισης που προσφέρει το αναγκαίο λογισμικό προκειμένου να εκτελεστούν κάποιοι διαχειριστικοί στόχοι όπως :

- Εισαγωγή των κανόνων που σχετίζονται με τον συγκερασμό των δεδομένων, την διευθυνσιοδότηση βασισμένη στα χαρακτηριστικά και την ομαδοποίηση των αισθητήριων κόμβων.
- Ανταλλαγή των δεδομένων που σχετίζονται με αλγόριθμους εύρεσης θέσης.
- Χρονικός συγχρονισμός των αισθητήριων κόμβων.
- Να κινήσει τους αισθητήριους κόμβους.
- Να τους θέσει εντός και εκτός λειτουργίας.
- Να θέσει ερωτήματα για την τρέχουσα κατάσταση και τις ρυθμίσεις των αισθητήριων κόμβων, καθώς και να επαναρυθμίσει το ασύρματο δίκτυο.
- Να εκτελέσει την αυθεντικοποίηση, την διανομή των κλειδιών και την ασφάλεια στις επικοινωνίες

4.5.2. Task assignment and data advertisement protocol.

Μια άλλη σημαντική λειτουργία στα ασύρματα δίκτυα αισθητήρων είναι η διάχυση είτε του ενδιαφέροντος των χρηστών για κάποιες πληροφορίες, είτε της γνωστοποίησης των υπαρχόντων πληροφοριών από τους αισθητήριους κόμβους. Το ενδιαφέρον των χρηστών μπορεί να αφορά σε ένα χαρακτηριστικό ή ένα φαινόμενο ή ενός γεγονότος που προκάλεσε κάποιο ερέθισμα. Από την άλλη πλευρά η γνωστοποίηση των κατεχομένων δεδομένων και πληροφοριών αφορά στην μετάδοση αυτής της πληροφορίας προς τους χρήστες ώστε αυτοί τελικά να θέσουν ερωτήματα για πληροφορίες που τους ενδιαφέρουν. Ένα πρωτόκολλο επιπέδου εφαρμογής που παρέχει τα παραπάνω δύο χαρακτηριστικά καθώς και το λογισμικό στο χρήστη με επαρκείς διεπαφές είναι χρήσιμο για τις λειτουργίες των κατωτέρων επιπέδων όπως η δρομολόγηση.

4.5.3. Sensor query and data dissemination protocol.

Το πρωτόκολλο SQDDP παρέχει στους χρήστες εφαρμογές με διεπαφές για να μπορούν να θέτουν ερωτήματα, απαντήσεις στα ερωτήματα και συλλογή των απαντήσεων. Τα ερωτήματα αυτά δεν τίθενται σε συγκεκριμένους κόμβους αλλά χρησιμοποιείται διευθυνσιοδότηση βασισμένη σε χαρακτηριστικά ή στην θέση των κόμβων. Παράδειγμα ερωτήματος είναι : «βρες την θέση των κόμβων που ανιχνεύουν θερμοκρασία άνω των 35^{o}C», ή «ποιες είναι οι θερμοκρασίες που ανιχνεύουν οι κόμβοι στην περιοχή Α».

5

ΥΠΑΡΧΟΥΣΕΣ ΤΕΧΝΟΛΟΓΙΕΣ MIDDLEWARE

Σε αυτό το κεφάλαιο θα γίνει αναφορά σε μερικά από τα πιο γνωστά υπάρχοντα συστήματα ενδιάμεσου λογισμικού που έχουν παρουσιαστεί τα τελευταία χρόνια για ασύρματα δίκτυα αισθητήρων. Οι περισσότερες σύγχρονες εφαρμογές είναι συσχετισμένες με συγκεκριμένες τεχνολογίες ενδιάμεσου λογισμικού ασύρματων δικτύων αισθητήρων.

Τα σημερινά δίκτυα ασύρματων αισθητήρων στοχεύουν σε εφαρμογές συλλογής δεδομένων και στις περισσότερες περιπτώσεις υποστηρίζουν τουλάχιστον μία τέτοια εφαρμογή ανά εγκατεστημένο δίκτυο. Γι' αυτό το λόγο μέχρι σήμερα ο σχεδιασμός των πρωτοκόλλων του δικτύου είναι άρρηκτα συνδεδεμένος με τις εφαρμογές που θα εκτελούνται σε αυτό . Τέτοιες όμως διαδικασίες είναι ad-hoc και επιβάλλουν άμεση αλληλεπίδραση με το ενσωματωμένο λειτουργικό σύστημα, ή ακόμα και το υλικό των κόμβων.

Όπως είναι προφανές κάτι δεν βοηθά στην κατασκευή αισθητήριων κόμβων οι οποίοι θα μπορούν να λειτουργήσουν όπως οι σημερινοί υπολογιστές, δηλαδή ανεξάρτητοι από το δίκτυο και τις εφαρμογές. Συνεπώς η εξάπλωση τους παραμένει μια διαδικασία δύσκολη και ακριβή. Θα πρέπει να δημιουργηθούν μέθοδοι σχεδιασμού εφαρμογών που να μη βασίζονται στα πρωτόκολλα και στο υλικό των δικτύων αισθητήρων. Επιπροσθέτως θα πρέπει να προβλεφθεί η περίπτωση όπου πολλαπλές εφαρμογές θα εκτελούνται ταυτόχρονα σε ένα δίκτυο αισθητήρων.

Από τα παραπάνω καθίσταται αναγκαία η ύπαρξη ενός ενδιάμεσου λογισμικού (middleware) το οποίο θα βρίσκεται μεταξύ των εφαρμογών και των λειτουργικών συστημάτων καθώς και του υλικού του δικτύου. Αυτό θα πρέπει να παρέχει :

- Τυποποιημένες υπηρεσίες συστήματος σε διάφορες εφαρμογές
- ένα περιβάλλον εκτέλεσης που θα υποστηρίζει και θα συντονίζει πολλαπλές εφαρμογές και

- μηχανισμούς που θα επιτυγχάνουν την προσαρμόσιμη και αποτελεσματική χρησιμοποίηση των πόρων του εκάστοτε συστήματος.

Τέτοιο ενδιάμεσο λογισμικό είναι ιδιαιτέρως χρήσιμο σε δίκτυα ασύρματων αισθητήρων που φιλοξενούν πολύπλοκες εφαρμογές με μεγάλο όγκο επεξεργασίας πληροφοριών και με αυστηρούς περιορισμούς στην κατανάλωση της ενέργειας.

Αν και έχουν γίνει αρκετές προσπάθειες για την ανάπτυξη διαφόρων πρωτοκόλλων επικοινωνίας και δρομολόγησης το θεμελιώδες πρόβλημα της αναγνώρισης, σχεδίασης και ανάπτυξης ενός κατάλληλου ενδιάμεσου λογισμικού (middleware) που θα εξυπηρετεί πλήρως τις ικανότητες και εφαρμογές των δικτύων αισθητήρων, δεν έχει λυθεί ακόμα. Τα παραδοσιακά κατανεμημένα ενδιάμεσα λογισμικά (distributed middleware) (όπως το DCOM, CORBA, PVM) είναι συνήθως «βαριά» σε κατανάλωση μνήμης και απαιτήσεων υπολογιστικής δύναμης, οπότε και καθίστανται ακατάλληλα για το περιβάλλον των δικτύων ασύρματων αισθητήρων. Για την περίπτωση μας είναι αναγκαίος ένας σχεδιασμός απλός, εύκολα υλοποιήσιμος, και «ελαφρύς» όσον αφορά στην κατανάλωση πόρων του συστήματος (ενέργεια, μνήμη, υπολογιστική ισχύς). Επιπλέον πρέπει να ανταποκρίνεται στα χαρακτηριστικά των δικτύων ασύρματων αισθητήρων που διαφέρουν από τα παραδοσιακά δίκτυα, όπως είναι η μη σχεδιασμένη από πριν εγκατάσταση (ad-hoc deployment), η μη παρακολουθούμενη λειτουργία και η λειτουργία σε δυναμικά περιβάλλοντα.

5.1 Χαρακτηριστικά του ενδιάμεσου λογισμικού.

Το ενδιάμεσο λογισμικό είναι συνήθως λογισμικό που ενώνει τους παρόχους υπηρεσιών (service suppliers) και τους χρήστες των υπηρεσιών αυτών (service consumers). Πάροχος υπηρεσιών θεωρείται οποιοσδήποτε τύπος δικτυακού κόμβου (υλικό ή λογισμικό) που μπορεί να προσφέρει υπηρεσίες (π.χ. εκτυπωτές, αισθητήρες, βάσεις δεδομένων και εφαρμογές). Αντίστοιχα, χρήστης υπηρεσιών είναι οποιοσδήποτε τύπος δικτυακού κόμβου που απαιτεί υπηρεσίες από ένα πάροχο. Οι εφαρμογές επιπλέον μπορούν να συμπεριφέρονται και σαν παροχείς υπηρεσιών αλλά και σαν χρήστες των παρεχομένων υπηρεσιών ταυτόχρονα.

Χαρακτηριστικά θα μπορούσαμε να αναφέρουμε πως το ενδιάμεσο λογισμικό χαρακτηρίζεται από τα παρακάτω: παρουσιάζει τα εξής χαρακτηριστικά:

5.1.1 Ανεξαρτησία από το δίκτυο

Το ενδιάμεσο λογισμικό (middleware) συνήθως λειτουργεί σαν μια γέφυρα μεταξύ πολλαπλών δικτυακών εφαρμογών, ή πρέπει να προσαρμοστεί σε πολλαπλά υποκείμενα δίκτυα. Αυτό περικλείει την ικανότητα να συνδέονται ενσύρματες δικτυακές τεχνολογίες, όπως τοπικά δίκτυα Ethernet και ATM δίκτυα κορμού μεταξύ τους ή με άλλα ασύρματα δίκτυα όπως Bluetooth δίκτυα, 802.11 ή υπέρυθρα ασύρματα δίκτυα. Πέρα από όλα αυτά ένα ενδιάμεσο λογισμικό για να είναι ευέλικτο θα πρέπει να είναι ανεξάρτητο από το δίκτυο στο οποίο εφαρμόζεται.

5.1.2 Λειτουργίες αυτόματης προσαρμογής για εκτέλεση(plug and play).

Ένα άλλο ζήτημα κλειδί, είναι η ικανότητα του ενδιάμεσου λογισμικού να προσαρμόζεται καθώς αλλάζει το περιβάλλον. Μέρος αυτής της ικανότητας εξαρτάται από τους μηχανισμούς που διαθέτει για ανίχνευση των υπηρεσιών που είναι ενσωματωμένες στο σύστημα. Οι μηχανισμοί μπορεί να είναι εντελώς κατανεμημένοι, ελεγχόμενοι κεντρικά ή ένα μίγμα και των δύο. Η επιλογή τους εξαρτάται από το μέγεθος του δικτύου, το μέγεθος της ανεκτικότητας στο επιπλέον κόστος της επικοινωνίας καθώς και το πόσο συχνά αλλάζουν τα συστατικά του δικτύου. Μια άλλη προσέγγιση είναι να επιτρέπεται στον μηχανισμό ανίχνευσης της υπηρεσίας να προσαρμόζεται στο τρέχων περιβάλλον, επιλέγοντας μια κατανεμημένη ή κεντρικού ελέγχου επιλογή, ανάλογα με χαρακτηριστικά του δικτύου όπως η πυκνότητα ή η κίνηση.

Η ανίχνευση της υπηρεσίας μπορεί να αυξήσει την ευελιξία του ενδιάμεσου λογισμικού, προσφέροντας μια περίληψη της διεπαφής με την μορφή των σημασιολογικών γλωσσών όπως η XML.

5.1.3 Ποιότητα Υπηρεσίας (QoS).

Ένα άλλο σημαντικό χαρακτηριστικό πολλών ενδιάμεσων συστημάτων (middleware systems) είναι η ικανότητά τους να χειρίζονται την ποιότητα υπηρεσίας του συστήματος (System QoS) μεταξύ των παρόχων υπηρεσιών (service supplier) και των χρηστών (service consumer), λαμβάνοντας υπόψη τους περιορισμούς του δικτύου. Το επίπεδο ποιότητας υπηρεσίας των παροχών υπηρεσιών (service supplier) περιλαμβάνει : απαραίτητα στοιχεία για σύνδεση, έλεγχο πρόσβασης, πιστοποίηση και κρυπτογράφηση δεδομένων, περιορισμούς ενέργειας (για την περίπτωση των χειριζόμενων συστημάτων με μπαταρία), καθώς και περιορισμούς διαθεσιμότητας υπηρεσιών (δηλ. μια υπηρεσία μπορεί

να μην είναι διαθέσιμη κάθε στιγμή). Η ποιότητα υπηρεσίας από την άποψη του χρήστη (service consumer) πρέπει να περιλαμβάνει την υπηρεσία και τα αναγκαία χαρακτηριστικά γνωρίσματα στο χρόνο και το χώρο βασισμένα σε αξιόπιστες μετρήσεις. Πρέπει επίσης να περιλαμβάνει τους χρονικούς περιορισμούς της ποιότητας υπηρεσίας (δηλ. μια εφαρμογή μπορεί να είναι ευαίσθητη σε χρονικές καθυστερήσεις (π.χ. παρακολούθηση χώρου), ενώ κάποια άλλη όχι (π.χ. παρακολούθηση του καιρού). Μια άλλη πιθανότητα είναι η χωρική ποιότητα υπηρεσίας δηλαδή αυτή που συνδέεται με την φυσική τοποθεσία. Ένα παράδειγμα είναι να ζητηθούν πληροφορίες για ένα θέμα που μπορούν να δοθούν από πολλούς κόμβους άλλα να επιθυμούμε να τις πάρουμε από εκείνους που διαθέτουν καλύτερη ακρίβεια ή/και ταχύτητα. Οποιοδήποτε χαρακτηριστικό ποιότητας υπηρεσίας θα πρέπει επίσης να παρέχει στο ενδιάμεσο λογισμικό εργαλεία για να μπορεί να αντιμετωπίσει με προοδευτική υποβάθμιση την παρουσία λαθών στο δίκτυο, ούτως ώστε το δίκτυο να έχει μεγαλύτερη αντοχή σε λάθη και αστοχίες.

5.1.4 Εύρεση θέσης και δρομολόγηση

Σε πολλά συστήματα, ειδικά αν πρόκειται για κινητά συστήματα, το ενδιάμεσο λογισμικό απαιτεί γνώση της θέσης του συστήματος και την ύπαρξη μηχανισμού για επικοινωνία μέσω διαδοχικών κόμβων(multiple network hops). Αντί να ενσωματωθούν τέτοιες υπηρεσίες στην κάθε εφαρμογή θα ήταν καλύτερο να ενσωματωθούν στο ενδιάμεσο λογισμικό. Αν και τα πρωτόκολλα ανίχνευσης της υπηρεσίας μπορούν να προσφέρουν λειτουργίες εύρεσης θέσης, μπορεί να μην έχουν επαρκή πληροφορία για τα κατώτερα επίπεδα του δικτύου προκειμένου να πάρουν τις βέλτιστες αποφάσεις. Γι’ αυτό, το ενδιάμεσο λογισμικό πρέπει να μπορεί να εκμεταλλεύεται την πληροφορία από τα κατώτερα επίπεδα του δικτύου για εύρεση της θέσης και δρομολόγηση όταν απαιτούνται τέτοιοι έλεγχοι.

5.1.5 Δοσοληψίες (Transactions).

Ακόμα και στον πιο βασικό σχεδιασμό, το ενδιάμεσο λογισμικό επιτρέπει τη επικοινωνία μεταξύ κατανεμημένων κόμβων. Με τον όρο δοσοληψίες (transactions) εννοούμε την αλληλεπίδραση μεταξύ ενός παροχέα υπηρεσιών (service supplier) και ενός χρήστη (service consumer) των υπηρεσιών αυτών. Μια δοσοληψία πρέπει να εγκαθιδρυθεί από το ενδιάμεσο λογισμικό βασισμένη σε συγκεκριμένα χαρακτηριστικά, μεταξύ του παροχέα και του χρήστη, συμπεριλαμβανομένου του περιορισμού της ποιότητας υπηρεσίας. Οι δοσοληψίες μπορούν να κατηγοριοποιηθούν σαν συνεχείς, διακοπτόμενες, ή

προγραμματισμένες κατ' απαίτηση. Οι δοσοληψίες αντιπροσωπεύουν ένα σημαντικό θέμα στο ενδιάμεσο λογισμικό, που σχετίζεται με τον τρόπο που ανταλλάσσονται τα δεδομένα και οι λειτουργίες μεταξύ του παροχέα και του χρήστη υπηρεσιών.

5.1.6 Χρονοπρογραμματισμός

Μερικά περιβάλλοντα έχουν αυστηρούς περιορισμούς σε λειτουργίες που μπορούν να εκτελεστούν σε μια συγκεκριμένη χρονική στιγμή. Το ενδιάμεσο λογισμικό μπορεί να αποφασίσει για την χρονική σειρά των αλληλεπιδράσεων, βάσει προτεραιοτήτων ή περιορισμών του εύρους ζώνης. Για παράδειγμα, αν μια υπηρεσία πρόκειται να διακοπεί (π.χ. παροχή υπηρεσίας σε κινητό το οποίο κινείται εκτός εμβελείας), τότε η δοσοληψία που την αφορά θα πρέπει είτε να ολοκληρωθεί είτε να μεταφερθεί σε άλλη υπηρεσία που ταιριάζει με τους περιορισμούς της αρχικής. Αυτές οι αλληλεπιδράσεις μπορούν να χρόνο-προγραμματιστούν με υψηλή προτεραιότητα και πιθανώς να τους αποδοθεί περισσότερο εύρος ζώνης. Παρόμοια ζητήματα χρόνο-προγραμματισμού τίθενται στο grid computing όπου το ενδιάμεσο λογισμικό πρέπει να καθορίζει σειρά εκτέλεσης των διεργασιών στους επεξεργαστές.

5.1.7 Σύστημα ανάκαμψης (Recovery System).

Αν το ενδιάμεσο λογισμικό χειρίζεται κρίσιμες δοσοληψίες , θα πρέπει να περιλαμβάνει ένα σύστημα ανάκαμψης που θα ασχολείται με τυχόν αστοχίες. Μερικές φορές μπορεί να χρησιμοποιηθεί ένα απλό ημερολόγιο καταγραφής γεγονότων, ενώ άλλες πρέπει να ενσωματωθούν ειδικοί μηχανισμοί ανάκαμψης βασισμένοι σε βάσεις δεδομένων.

5.1.8 Διαλειτουργικότητα

Σε μερικά συστήματα το ενδιάμεσο λογισμικό δίνει έμφαση στην ανάγκη σύνδεσης μεταξύ πολλαπλών γλωσσών και /ή μεταξύ πλατφόρμων ενδιάμεσου λογισμικού. Το CORBA μπορεί να συνδεθεί με τις περισσότερες κοινές εφαρμογές που βασίζονται σε διαφορετικές γλώσσες προγραμματισμού, αλλά πρέπει να υπάρχει ένα αντικείμενο CORBA, το οποίο μερικές φορές είναι δύσκολο να υλοποιηθεί σε μικρά συστήματα. Υπάρχει γενικότερος προβληματισμός αν θα είναι επιτυχές το παραπάνω σε ένα μεγάλης κλίμακας δίκτυο όπως το Διαδίκτυο. Θα πρέπει ταυτόχρονα με την διαλειτουργικότητα να ζυγιστεί προσεκτικά το κόστος της, ειδικά όταν μιλάμε για ενσωματωμένα

(embedded) συστήματα. Για ένα οποιοδήποτε σύστημα, είναι αναγκαία η χρήση μιας σημασιολογικής γλώσσας (markup language) όπως είναι η XML ή οποιασδήποτε άλλης η οποία παρέχει σημασιολογική ανεξαρτησία (semantics independence), για να έχουμε εγγυημένη διαλειτουργικότητα (guarantee interoperability).

5.2 Χαρακτηριστικά των εφαρμογών.

Αν και οι εφαρμογές των δικτύων αισθητήρων καλύπτουν ένα μεγάλο εύρος δραστηριοτήτων μπορούν να χαρακτηριστούν από αρκετά κοινά χαρακτηριστικά. Συνήθως οι αισθητήρες λειτουργούν με μπαταρίες και εγκαθίστανται σε μια περιοχή προκειμένου να παρακολουθήσουν διάφορα φαινόμενα. Προκειμένου να εξοικονομήσουν ενέργεια, οι αισθητήρες μπορούν να έχουν διάφορες καταστάσεις λειτουργίας, και συνεπώς διαφορετική ενεργειακή κατανάλωση.

Οι μηχανισμοί που αλλάζουν τις καταστάσεις λειτουργίας ενός αισθητήρα καλούνται «μοχλοί» του συστήματος (system “knobs”). Αρχικά, προκειμένου να εξοικονομήσουν ενέργεια οι αισθητήρες μπορούν να βρίσκονται σε κατάσταση «ύπνου», εκτός από μερικούς οι οποίοι αναλαμβάνουν το ρόλο της «φύλαξης». Όταν οι ενεργοί (αυτοί που δεν βρίσκονται σε κατάσταση «ύπνου») αισθητήριοι κόμβοι διαπιστώσουν την ύπαρξη κάποιου ανιχνεύσιμου φαινόμενου, είναι υπεύθυνοι να ενεργοποιήσουν τον απαιτούμενο αριθμό κόμβων προκειμένου να ανιχνευθεί καλύτερα το φαινόμενο αυτό.

Προκειμένου να εκπληρωθεί η παραπάνω διαδικασία πρέπει να ληφθούν υπόψη κάποια ζητήματα, όπως :

- Η δημιουργία μηχανισμών οι οποίοι θα θέτουν σε λειτουργία επιλεκτικά κάποιους αισθητήριους κόμβους, οι οποίοι βρίσκονται σε κατάλληλες θέσεις γύρω από την περιοχή του εκάστοτε στόχου και ταυτόχρονα έχουν αρκετή εναπομένουσα ενέργεια.
- Η συνεργασία μεταξύ των ενεργοποιημένων κόμβων για να κατανέμουν το συνολικό επεξεργαστικό φόρτο, να συγκεράσουν (aggregate) τα αποτελέσματα και τέλος να δρομολογήσουν το τελικό αποτέλεσμα στο σταθμό βάσης.
- Τυχόν απαιτήσεις ποιότητας υπηρεσίας που θα πρέπει να ικανοποιηθούν. Πιο συγκεκριμένα, πρόκειται για απαιτήσεις οι οποίες είναι σχετικές με την ακρίβεια της απόφασης, και οι οποίες καθορίζουν την απαιτούμενη ποσότητα της ενέργειας καθώς και των πόρων για υπολογισμούς, επικοινωνίες, και ανίχνευση.

- Η διαχείριση του δικτύου, που μπορεί να γίνει αρκετά πολύπλοκη λόγω της ανομοιογένειας των αισθητήριων κόμβων (στο δίκτυο μπορεί να υπάρχουν κόμβοι με διαφορετικά επίπεδα ενέργειας, δυνατότητες επεξεργασίας και ανίχνευσης).

Τα παραπάνω ζητήματα δεν μπορούν να παραβλεφθούν, αν αναλογιστούμε α) σε κάποιες περιπτώσεις, τα ad-hoc χαρακτηριστικά του δικτύου, β) την περιορισμένη ενέργεια και τις δυνατότητες υπολογιστικής ισχύος, επικοινωνίας και ανίχνευσης των αισθητήρων, γ) τις πιθανές μεταβολές στις συνθήκες του συστήματος και του περιβάλλοντος και δ) το γεγονός ότι οι αισθητήριοι κόμβοι λειτουργούν στο περιβάλλον χωρίς κάποια ανθρώπινη παρακολούθηση. Προκειμένου να περιοριστεί η πολυπλοκότητα για τους σχεδιαστές των εφαρμογών, το ενδιάμεσο λογισμικό πρέπει να παρέχει ένα περιβάλλον εκτέλεσης που θα λαμβάνει υπόψη τα παραπάνω χαρακτηριστικά και περιορισμούς. Επίσης για να εγγυηθεί η σωστή λειτουργία πολλαπλών εφαρμογών κάτω από τους αυστηρούς ενεργειακούς περιορισμούς, το ενδιάμεσο λογισμικό πρέπει να είναι ικανό να «θυσιάζει» αποτελεσματικά την ποιότητα υπηρεσίας μίας εφαρμογής προς όφελος κάποιας άλλης και αντίστροφα.

5.3 Αρχές σχεδίασης του ενδιάμεσου λογισμικού (Middleware).

Προκειμένου το ενδιάμεσο λογισμικό να είναι σε θέση να ανταποκριθεί στα χαρακτηριστικά των δικτύων ασύρματων αισθητήρων θα πρέπει κι αυτό με τη σειρά του να ακολουθεί κάποιες βασικές αρχές, όπως οι παρακάτω :

- Δεδομένο-κεντρικοί Μηχανισμοί (Data-centric mechanisms): Το ενδιάμεσο λογισμικό πρέπει να παρέχει δεδομένο-κεντρικούς μηχανισμούς για την επεξεργασία των δεδομένων και τις ερωτήσεις στο δίκτυο. Λόγω της απλότητας, ευελιξίας, και ευρωστίας (robustness) οι αρχιτεκτονικές που βασίζονται στην ομαδοποίηση έχουν ευρεία χρήση στο σχεδιασμό και την υλοποίηση των πρωτοκόλλων του δικτύου.
- Γνώση της εφαρμογής (application knowledge): Μπορεί να χρησιμοποιηθεί προκειμένου να συνδέσει το σχεδιασμό και την υλοποίηση του λογισμικού. Είναι, λοιπόν, σημαντικό να ενοποιηθεί η γνώση της εφαρμογής με τις προσφερόμενες υπηρεσίες από το ενδιάμεσο λογισμικό. Όμως είναι δυνατόν, λόγω της αναγκαιότητας για υποστήριξη και βελτιστοποίηση ενός μεγάλου αριθμού εφαρμογών, να γίνεται μια ανταλλαγή μεταξύ του βαθμού υποστήριξης μιας συγκεκριμένης εφαρμογής και της γενικότητας του ενδιάμεσου λογισμικού. Μια εφαρμόσιμη πρακτική είναι να ενσωματώνονται τα μοναδικά χαρακτηριστικά της εφαρμογής μέσα στον κώδικά της, ο οποίος

θα μπορεί να διερμηνευτεί από το ενδιάμεσο λογισμικό. Αποτέλεσμα είναι οι πληροφορίες αυτές για τα χαρακτηριστικά και τις δυνατότητες που έχει η εφαρμογή να μπορούν αξιοποιηθούν και να χρησιμοποιηθούν για να κατευθύνουν τις λειτουργίες του ενδιάμεσου λογισμικού.

- Τοπικοί αλγόριθμοι (localized algorithms): θα πρέπει να χρησιμοποιούνται ώστε να επιτυγχάνουν συλλογικά ένα επιθυμητό μαζικό στόχο, ενώ ταυτόχρονα να προσφέρουν επεκτασιμότητα και ευρωστία στο σύστημα. Δεδομένου ότι η ομαδοποιημένη αρχιτεκτονική κάνει την αλληλεπίδραση των αισθητήριων κόμβων τοπική και ως εκ τούτου και το συντονισμό και έλεγχο των επιπλέον απαιτήσεων σε μια συγκεκριμένη περιοχή, είναι δικαιολογημένο να θεωρείται η κάθε ομάδα σαν μια βασική οντότητα του ενδιάμεσου λογισμικού. Συμπερασματικά, το ενδιάμεσο λογισμικό δρα σαν ένα κατανεμημένο λογισμικό αποτελούμενο από πολλαπλές ομάδες.
- Αλγόριθμοι Προσαρμόσιμης Πιστότητας (Adaptive fidelity algorithms): Αυτοί επιτρέπουν το αντιστάθμισμα μεταξύ της ποιότητας του αποτελέσματος και της χρήσης πόρων του συστήματος, γι' αυτό και πρόκειται για μια αρχή σχεδιασμού η οποία συμβάλει καθοριστικά στην αποτελεσματική χρήση των πόρων. Μια ιδανική περίπτωση είναι η εφαρμογή να μπορεί να διαλέξει μεταξύ πολλών διαφορετικών αλγορίθμων οι οποίοι επιλύουν το ίδιο πρόβλημα με διαφορετικές απαιτήσεις στην ποιότητα του αποτελέσματος και στην χρήση πόρων των κόμβων του δικτύου
- Ελαφρύ στη λειτουργία (lightweight): Αφού οι διαθέσιμες πηγές ενέργειας των αισθητήριων κόμβων είναι μικρές, ένα άλλο χαρακτηριστικό που πρέπει να διαθέτει το ενδιάμεσο λογισμικό είναι να είναι «ελαφρύ» σε απαιτήσεις υπολογισμών και επικοινωνίας. Το παραπάνω χαρακτηριστικό απαιτεί την χρήση απλών και αποτελεσματικών μεθόδων για λιγότερο βέλτιστες λύσεις.
- Λόγω των περιορισμένων πόρων, είναι πολύ πιθανό ότι δεν θα μπορούν να ικανοποιηθούν ταυτόχρονα οι απαιτήσεις απόδοσης των εκτελέσιμων εφαρμογών. Γι' αυτό και είναι αναγκαίο το ενδιάμεσο λογισμικό να μπορεί να διαπραγματεύεται την ποιότητα υπηρεσίας μεταξύ των εφαρμογών.

Στα συμβατικά υπολογιστικά συστήματα, κάθε συσκευή έχει και ένα κάτοχο, ο οποίος είναι υπεύθυνος για την ρύθμιση, τη συντήρηση και τον χειρισμό της σε περίπτωση λάθους ή άλλης αστοχίας. Σε αντίθεση, τα δίκτυα ασύρματων αισθητήρων πρέπει να μπορούν να λειτουργούν χωρίς παρακολούθηση, γεγονός που σημαίνει ότι το ενδιάμεσο λογισμικό πρέπει να παρέχει επίπεδα υποστήριξης για αυτόματη ρύθμιση και χειρισμό λαθών. Άλλο θέμα αποτελεί η χρήση των δικτύων ασύρματων αισθητήρων για την παρακολούθηση γεγονότων του πραγματικού κόσμου με αποτέλεσμα ο χρόνος και η θέση να παίζουν πολύ σημαντικό ρόλο. Επίσης μερικές εφαρμογές μπορεί να έχουν απαιτήσεις για λειτουργία και εξαγωγή αποτελεσμάτων σε πραγματικό χρόνο. Αυτό έχει ως

αποτέλεσμα το ενδιάμεσο λογισμικό να πρέπει να έχει από κατασκευής του, ενσωματωμένα τα χαρακτηριστικά για τη διαχείριση της έννοιας του χρόνου και της θέσης.

5.4 Κατηγορίες ενδιάμεσου λογισμικού.

Οι υπάρχουσες προσεγγίσεις σε ότι αφορά το ενδιάμεσο λογισμικό, ομαδοποιούνται στις επόμενες τρεις κατηγορίες:

- Κινητών πρακτόρων (Mobile Agents): Η κατηγορία αυτή προσεγγίζει το δίκτυο εμπνευσμένη από τους κινητούς πράκτορες και τον κινητό κώδικα (mobile code) και λειτουργεί εισάγοντας ένα πρόγραμμα στο δίκτυο. Το πρόγραμμα αυτό μπορεί να συγκεντρώσει δεδομένα από τους κόμβους, να διαδώσει αντίγραφα του εαυτού του στο δίκτυο και να επικοινωνεί με αυτά.
- Βάσεων Δεδομένων: Η κατηγορία αυτή χειρίζεται το δίκτυο σαν μια κατανεμημένη βάση δεδομένων όπου οι χρήστες μπορούν να θέτουν ερωτήματα τύπου SQL προκειμένου το δίκτυο να εκτελέσει κάποια αισθητήρια λειτουργία.
- Συμβάντων (Events): Η κατηγορία αυτή βασίζεται στην έννοια των συμβάντων. Σύμφωνα με αυτή τη προσέγγιση οι εφαρμογές επικεντρώνουν το ενδιαφέρον τους σε αλλαγές της κατάστασης των αντικειμένων που παρακολουθούν στον πραγματικό κόσμο δηλαδή στα συμβάντα που λαμβάνουν χώρα. Όταν ένας αισθητήρας αντιληφθεί ένα τέτοιο γεγονός, στέλνει μια ειδοποίηση συμβάντος προς την εφαρμογή. Η εφαρμογή μπορεί όμως να θέσει περιορισμούς στις ειδοποιήσεις θέτοντας κάποια κριτήρια τα οποία πρέπει να πληρούνται προκειμένου οι αισθητήρες να στείλουν τέτοιου είδους ειδοποιήσεις.

5.5 Προσέγγιση κινητών πρακτόρων.

Στα περισσότερα δίκτυα θεωρείται ότι οι προτεινόμενοι αλγόριθμοι, είναι ενσωματωμένοι στην μνήμη του κάθε κόμβου. Σε μερικές πλατφόρμες, όπως το TinyOS, ο σχεδιαστής εφαρμογών χρησιμοποιεί ένα λειτουργικό σύστημα με επίπεδα στον κάθε κόμβο, το οποίο έχει πολλά πλεονεκτήματα καθώς προσφέρει συναρμολογησιμότητα, πολυδιεργασία, και μία αφαιρετική άποψη του υλικού. Ακόμα και σε αυτήν την περίπτωση ο σχεδιαστής πρέπει να δημιουργήσει ένα εκτελέσιμο πρόγραμμα το οποίο θα το φορτώσει χειροκίνητα σε κάθε ένα κόμβο.

Μια προσέγγιση, η οποία κερδίζει μεγάλη απήχηση τελευταία, είναι η προσέγγιση του ενεργού αισθητήρα («active sensor» approach). Ο όρος χρησιμοποιείται για να περιγράψει μια ομάδα πλαισίων (frameworks) τα οποία προσπαθούν να προγραμματίσουν τους αισθητήρες με ένα συνηθισμένο τρόπο. Ο τρόπος αυτός είναι παρόμοιος με αυτόν που τα πλαίσια (frameworks) προγραμματίζουν τους κόμβους ενός οποιουδήποτε δικτύου. Η διαφορά ανάμεσα στα δύο είναι το γεγονός ότι στο πρώτο δίκτυο πρέπει να προγραμματίσουμε να αντιδρά μόνο όταν λαμβάνει ένα πακέτο δεδομένων, ενώ στα ασύρματα δίκτυα αισθητήρων έχουμε αντίδραση σε πολλών ειδών γεγονότα, όπως αίσθησης, δικτύου, και εκπνοής χρόνου.

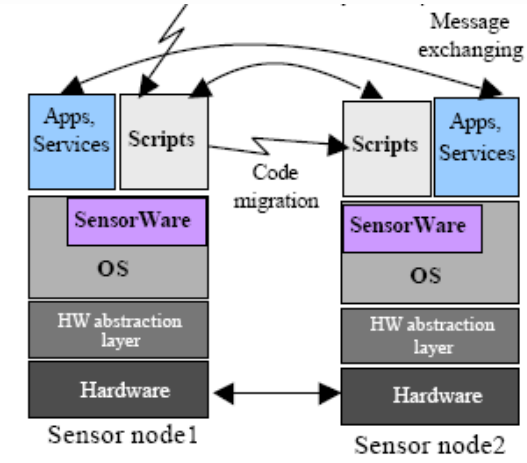
5.5.1 SENSOR WARE

Η αρχιτεκτονική του SensorWare βασίζεται σε ένα περιβάλλον εκτέλεσης (run-time) με δυνατότητα εγγραφής μικρών προγραμμάτων (scriptable). Το περιβάλλον αυτό έχει βελτιστοποιηθεί για χρήση σε αισθητήριους κόμβους με περιορισμένη ενέργεια και μνήμη. Είναι συμβατό για ένα ή περισσότερα απλά, συμπτυγμένα, και ανεξάρτητα πλατφόρμας σενάρια (script) ελέγχου αισθητήριων κόμβων. Οι πόροι για ανίχνευση, επικοινωνία και επεξεργασία σήματος ενός κόμβου, εκτίθενται στα προγράμματα ελέγχου που οργανώνουν την ροή των δεδομένων ώστε να δημιουργήσουν τα συνηθισμένα πρωτοκολλά και στοίβες επεξεργασίας σήματος. Το ενδιάμεσο λογισμικό πρέπει επίσης να προάγει την δημιουργία κατανεμημένων προδραστικών αλγορίθμων που βασίζονται στη γλώσσα προγραμματισμού που περιγράφηκε παραπάνω. Όταν λέμε προδραστικό αλγόριθμο εννοούμε ένα αυτόνομο και κατανεμημένο αλγόριθμο. Γι' αυτό το λόγο τα σενάρια (scripts) γίνονται «κινητά» χρησιμοποιώντας ειδικές εντολές και οδηγίες. Ένα τέτοιο script μπορεί να αναπαραχθεί (replicate) ή να μεταναστεύσει (migrate) τον κώδικά και τα δεδομένα του σε άλλους κόμβους, επηρεάζοντας άμεσα την συμπεριφορά τους. Αυτή τη διαδικασία θα την ονομάζουμε «αναπαραγωγή» ("population").

Αρχιτεκτονική.

Το SensorWare όπως φαίνεται και στην Εικόνα 8 τοποθετείται μέσα στην αρχιτεκτονική του αισθητήριου κόμβου. Η αρχιτεκτονική αυτή μπορεί να περιγραφεί σε επίπεδα. Στα κατώτερα επίπεδα βρίσκεται το υλικό (Hardware) καθώς και το επίπεδο αφαίρεσης του υλικού (Hardware Abstraction Layer) (δηλ οι οδηγοί των συσκευών). Το λειτουργικό σύστημα (OS) βρίσκεται στο πιο υψηλό επίπεδο.

Περιστασιακοί εξωτερικοί χρήστες μπορούν να εισάγουν script



Εικόνα 8: Η γενική αρχιτεκτονική ενός αισθητήρα. [22]

Το λειτουργικό σύστημα (OS) παρέχει όλες τις τυποποιημένες λειτουργίες και υπηρεσίες ενός πολύ-νηματικού περιβάλλοντος που είναι αναγκαίες στα επόμενα επίπεδα. Το λογισμικό SensorWare, για παράδειγμα, χρησιμοποιεί αυτές τις λειτουργίες και υπηρεσίες για να παρέχει το περιβάλλον εκτέλεσης (run-time environment) για τα script ελέγχου. Αυτά εξαρτώνται άμεσα από το λογισμικό SensorWare καθώς «αναπαράγονται» στο δίκτυο.

Γλώσσα προγραμματισμού.

Μια γλώσσα προγραμματισμού (scripting language) απαιτεί τις σωστές συναρτήσεις και εντολές, προκειμένου να τις χρησιμοποιήσει ως δομικά στοιχεία. Κάθε μία από αυτές τις εντολές πρέπει να επιτελεί ένα συγκεκριμένο σκοπό για τον αισθητήριο κόμβο, όπως επικοινωνία με άλλους κόμβους, ή ανάκτηση των αισθητήριων δεδομένων. Μια δεύτερη απαίτηση για τη γλώσσα προγραμματισμού (scripting language) είναι η ύπαρξη βασικών δομών για την οικοδόμηση δομικών στοιχείων σε script ελέγχου. Αρκετές από τις βασικές εντολές/ συναρτήσεις του SensorWare είναι ομαδοποιημένες σε API's με σχετικό θέμα. Ο όρος API χρησιμοποιείται για να δείξει μια συλλογή σχετιζόμενων συναρτήσεων που προσφέρουν μια διεπαφή προγραμματισμού προς κάποιο πόρο ή υπηρεσία.

Το μοντέλο προγραμματισμού που χρησιμοποιείται είναι ισοδύναμο με το εξής : Ένα συμβάν περιγράφεται, και συσχετίζεται με τον ορισμό ενός χειριστή

συμβάντων (event hander). Ο χειριστής συμβάντων, σύμφωνα με την τρέχουσα κατάσταση, θα εκτελέσει μερική επεξεργασία και πιθανώς να δημιουργήσει κάποια νέα συμβάντα ή/και θα τροποποιήσει την κατάσταση του. Παραδείγματα συμβάντων είναι : α) λήψη ενός μηνύματος με συγκεκριμένη μορφοποίηση, β) λήψη μιας τιμής που είναι πάνω από ένα δοθέν όριο από μια αισθητήρια συσκευή, γ) γέμισμα ενός καταχωρητή (buffer) με αισθητήρια δεδομένα συγκεκριμένου ρυθμού δειγματοληψίας και δ) Μηδενισμός κάποιων χρονομετρητών.

Περιβάλλον εκτέλεσης (Run-time environment).

Τα scripts ελέγχου που αναφέρθηκαν παραπάνω είναι τόσο σημαντικά για την πλατφόρμα του SensorWare, όσο και το περιβάλλον εκτέλεσης που τα υποστηρίζει. Η Εικόνα 9 δείχνει τις βασικές λειτουργίες που εκτελούνται από το περιβάλλον εκτέλεσης. Οι λειτουργίες διαχωρίζονται στις πάγιες (fixed) και τις ειδικευμένες στην πλατφόρμα (platform-specific). Οι πάγιες λειτουργίες περιλαμβάνονται πάντα στην υλοποίηση του SensorWare, ενώ οι ειδικευμένες στην πλατφόρμα εξαρτώνται από τα υπάρχοντα αρθρώματα και υπηρεσίες στην πλατφόρμα του κάθε κόμβου.

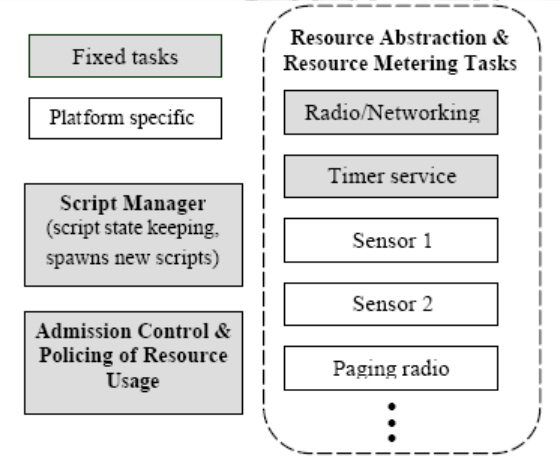
Ο διαχειριστής των Script (Script Manager task) είναι η λειτουργία που δέχεται όλες τις αιτήσεις για την παραγωγή νέων scripts. Προωθεί την αίτηση προς τη λειτουργία Ελέγχου Αποδοχής (Admission Control task) και μόλις λάβει θετική απάντηση, ξεκινάει την εκτέλεση ενός νέου κώδικα (script) και του διερμηνευτή του. Τηρεί κάθε κατάσταση που σχετίζεται με τον κώδικα αυτό, όπως τα δεδομένα του κώδικα (script-data), για όσο διάστημα είναι ενεργός. Επίσης τηρεί μια προσωρινή μνήμη του κώδικα του προγράμματος (script-code cache) με σκοπό να μειώσει τις εκπομπές του στο ασύρματο κανάλι.

Οι λειτουργίες του Ελέγχου Αποδοχής (Admission Control) και Ελέγχου (Αστυνόμευσης) της Διαχείρισης Πόρων (Policing of Resource Usage), όπως και το όνομά τους προδίδει, λαμβάνουν όλες τις αποφάσεις για την αποδοχή ενός script. Η σημαντικότερη λειτουργία τους είναι ο έλεγχος της κατανάλωσης ενέργειας. Αν η συνολική κατανάλωση ξεπερνάει τα ορισθέντα όρια, τότε η λειτουργία αυτή τερματίζει επιλεκτικά την εκτέλεση μερικών script σύμφωνα με τις συγκεκριμένες πολιτικές του SensorWare.

Το περιβάλλον εκτέλεσης περιλαμβάνει επίσης τις λειτουργίες «Resource Abstraction and Resource Metering» Κάθε λειτουργία υποστηρίζει τις εντολές του συγκεκριμένου API και διαχειρίζεται συγκεκριμένους πόρους. Υπάρχουν όμως και δύο πάγιες λειτουργίες σε αυτήν την κατηγορία αφού κάθε πλατφόρμα

θεωρείται ότι έχει τουλάχιστον μια υπηρεσία για τον ασύρματο και μια υπηρεσία χρονομετρητή.

Η λειτουργία του ασυρμάτου (“Radio” task) διαχειρίζεται τον ασύρματο δηλ. α) δέχεται τις αιτήσεις από τα scripts σχετικά με τη μορφή των μηνυμάτων του δικτύου που αναμένονται, β) δέχεται όλα τα μηνύματα του δικτύου και τα διαχωρίζει προς τα κατάλληλα scripts, σύμφωνα με τις ανάγκες τους και γ) μετράει την χρήση του ασυρμάτου για κάθε script. Η δεύτερη πάγια λειτουργία, που αφορά στην υπηρεσία του χρονομετρητή (timer service), δέχεται τις διάφορες αιτήσεις για λειτουργία χρονομετρητών από όλα τα scripts και προσπαθεί να τις εξυπηρετήσει χρησιμοποιώντας ένα χρονομετρητή τον οποίο παρέχει το ενσωματωμένο σύστημα



Εικόνα 9:Λειτουργίες στο περιβάλλον εκτέλεσης του SensorWare. [22]

Φορητότητα και Επεκτασιμότητα (Portability and Expandability).

Υπάρχουν δύο είδη παραλλαγής της πλατφόρμας (platform variability) :

α) Μεταβλητότητα των δυνατοτήτων : Διαφορετικές πλατφόρμες μπορεί να έχουν και διαφορετικές δυνατότητες. Για παράδειγμα, μια πλατφόρμα Α έχει ένα ασύρματο και ένα μαγνητόμετρο, ενώ μια άλλη Β έχει δύο ασύρματους (ένα κανονικό και ένα μεγαλύτερο) και μια φωτογραφική μηχανή. Το SensorWare χρησιμοποιεί μια αρθρωτή λύση προκειμένου να μπορεί να χειριστεί τις 2 αυτές πλατφόρμες. Δηλώνει, ορίζει και υποστηρίζει εικονικές συσκευές. Οποιοδήποτε

άρθρωμα ή υπηρεσία αναπαρίσταται σαν μία εικονική συσκευή π.χ. ασύρματος, αισθητήρια συσκευή, συσκευή χρονομέτρου, πρωτόκολλο εύρεσης θέσης.

β) Μεταβλητότητα του υλικού και λογισμικού (HW/SW variability) : Ακόμα κι αν δύο πλατφόρμες έχουν τις ίδιες δυνατότητες (δηλ. τα ίδια αρθρώματα/ υπηρεσίες), είναι δυνατόν να βασίζονται σε διαφορετικά υλικά ή/ και λειτουργικά συστήματα. Με σκοπό να διευκολυνθεί η διαδικασία μεταφοράς (porting) είναι επιθυμητό να διαχωριστούν ξεκάθαρα οι κώδικες (codes) του λειτουργικού συστήματος και του υλικού από τον πάγιο κώδικα (fixed code) και τον κώδικα των δυνατοτήτων (capabilities code). Για να επιτευχθεί το παραπάνω πρέπει να δημιουργηθούν αφηρημένες συναρτήσεις wrapper (abstracted wrapper functions), οι οποίες ορίζονται σε διαφορετικά τμήματα του κώδικα (δηλ σε διαφορετικά αρχεία c) έτσι ώστε ο κατασκευαστής να μπορεί εύκολα να αναγνωρίσει τα σημεία της αλλαγής για μία διαδικασία porting (porting procedure).

5.6 Προσέγγιση βάσεων δεδομένων.

Σύμφωνα με αυτήν την προσέγγιση, χειριζόμαστε τα δίκτυα αισθητήρων σαν μια κατανεμημένη βάση δεδομένων, στην οποία οι χρήστες μπορούν να θέτουν ερωτήματα τύπου SQL, προκειμένου το δίκτυο να εκτελέσει κάποια συγκεκριμένη λειτουργία. Τέτοιου τύπου ενδιάμεσο λογισμικό είναι το TinyDB και το SINA (*Sensor Information Networking Architecture and Applications).* Αυτές οι προσεγγίσεις χρησιμοποιούν κατανεμημένο έλεγχο και προσφέρουν επεξεργασία των δεδομένων από τους αισθητήρες προτού μεταδοθούν, επιτυγχάνοντας μείωση του όγκου δεδομένων προς μετάδοση. Με απλά λόγια, αυτό σημαίνει εξοικονόμηση ενέργειας, αφού η επεξεργασία καταναλώνει ελάχιστη ενέργεια σε σχέση με την επικοινωνία.

5.6.1 TinyDB

Το TinyDB, όπως και το Cougar αντιμετωπίζουν το δίκτυο σαν μια κατανεμημένη βάση δεδομένων. Θεωρούν την ύπαρξη ενός εικονικού πίνακα (Sensors) μιας βάσης δεδομένων, η κάθε στήλη του οποίου είναι και ένας συγκεκριμένος υποστηριζόμενος τύπος (π.χ., Θερμοκρασία, Υγρασία). Οι τιμές κάθε αισθητήρα καταγράφονται σε μια γραμμή του πίνακα αυτού. Οι χρήστες επιλέγουν τα δεδομένα που επιθυμούν μέσα από συγκεκριμένες επερωτήσεις, παρόμοιες με αυτές της SQL (για την ακρίβεια είναι ένα υποσύνολό τους με κάποιους περιορισμούς, (π.χ. δεν υποστηρίζεται το λογικό “or”) και κάποιες επεκτάσεις

που επιτρέπουν σε μια επερώτηση να είναι περιοδικά επαναλαμβανόμενη). Και το TinyDB και το Cougar χρησιμοποιούν μια αποκεντρωμένη προσέγγιση, όπου ο κάθε κόμβος έχει το δικό του επεξεργαστή επερωτήσεων που επεξεργάζεται και αποστέλλει τα δεδομένα που συλλέγει κατευθείαν στο χρήστη. Παρόλο που το TinyDB θεωρείται μια “de-facto” λύση, η SQL-like φύση των επερωτήσεων που δέχεται δεν είναι πάντοτε ευέλικτη. Επίσης δεν υποστηρίζει τον “on the-fly” ορισμό συμβάντων, αλλά μόνο ορισμένα από πριν συμβάντα τα οποία έχουν προγραμματιστεί στο υλικό του αισθητήρα κατά τη διάρκεια της μεταγλώττισης.

5.6.2 SINA

Το SINA είναι ένα ενδιάμεσο λογισμικό που επιτρέπει σε εφαρμογές των αισθητήρων να πραγματοποιούν επερωτήσεις και εργασίες, να συλλέγουν απαντήσεις και αποτελέσματα και να παρακολουθούν αλλαγές μέσα στο δίκτυο. Τα βασικά χαρακτηριστικά του περιλαμβάνουν την ιεραρχική ομαδοποίηση των κόμβων των αισθητήρων, την ονοματοδοσία βάσει μεταβλητών των κόμβων, καθώς και ένα παράδειγμα οργάνωσης των δεδομένων στους κόμβους. Το SINA χρησιμοποιεί SQLLike επερωτήσεις καθώς και SQTL (Sensor Query and Tasking Language) διαδικαστικά scripts. Οι SQL-Like επερωτήσεις χρησιμοποιούν τα προαναφερθέντα χαρακτηριστικά για να εκτελέσουν απλές εργασίες επερωτήσεων και παρακολούθησης, ενώ για πιο προχωρημένες εφαρμογές η SQTL παίζει το ρόλο της διεπαφής μεταξύ των αισθητήρων και του SINA. Ένα SQTL μήνυμα που περιέχει ένα script, βρίσκεται μέσα σε ένα XML-Like SQTL Wrapper και προορίζεται ώστε να διαβαστεί και να εκτελεστεί από ένα περιβάλλον εκτέλεσης των αισθητήρων (Sensor Execution Environment - SEE), που τρέχει σε κάθε αισθητήρα. Εν αντιθέσει με το TinyDB και το Cougar, επειδή παρέχει μια εναλλακτική scripting διεπαφή, θεωρείται πιο ευέλικτη προσέγγιση. Παρόλα αυτά ο ουσιαστικός προγραμματισμός των εργασιών μπορεί να αποδειχθεί αρκετά δύσκολος.

5.6.3. Cougar.

Το COUGAR είναι ένα κατανεμημένο σύστημα διαχείρισης δεδομένων. Εγκαθίσταται κατευθείαν στους αισθητήριους κόμβους και δημιουργεί ένα ιδεατό κεντρικό κόμβο χωρίς όμως να υπάρχει κεντρικός έλεγχος δεδομένων ή υπολογισμοί. Έχει μια 3 επιπέδων αρχιτεκτονική :

QueryProxy : Ένα συστατικό μιας μικρής Βάσης Δεδομένων που εκτελείται στους αισθητήριους κόμβους για την μεταγλώττιση και την εκτέλεση των ερωτημάτων.

FronEnd : Πρόκειται για ένα πιο ισχυρό QueryProxy που λειτουργεί σαν πύλη (gateway) για τις συνδέσεις με τον έξω κόσμο του δικτύου αισθητήρων.

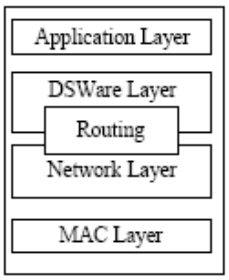
GUI : Ένα γραφικό περιβάλλον μέσω του οποίου οι χρήστες μπορούν να θέτουν ερωτήματα στο δίκτυο, αλλά και να λαμβάνουν τις απαντήσεις.

5.7 Προσέγγιση βασισμένη στην έννοια των συμβάντων (Events).

Μια άλλη προσέγγιση στο ενδιάμεσο λογισμικό για δίκτυα αισθητήρων βασίζεται στην έννοια των συμβάντων. Σύμφωνα με αυτή, η εφαρμογή δηλώνει το ενδιαφέρον της για συγκεκριμένες αλλαγές της κατάστασης των παρακολουθούμενων γεγονότων (βασικά συμβάντα) στον πραγματικό κόσμο. Μόλις ένα τέτοιο συμβάν ανιχνευθεί, ο αισθητήριος κόμβος, που το ανίχνευσε, στέλνει μια ειδοποίηση (event notification) προς την εφαρμογή που έχει εκδηλώσει το ανάλογο ενδιαφέρον. Μπορεί να καθοριστεί από την εφαρμογή και ένα συγκεκριμένο σχήμα συμβάντων (pattern of events), έτσι ώστε να έχουμε ειδοποίηση μόνο αν λάβει χώρα η συγκεκριμένη σειρά γεγονότων.

5.7.1. DSWare Data Service Middleware.

Το DSWare έχει τη δυνατότητα να αποφεύγει τη επανάληψη της υλοποίησης των τμημάτων υπηρεσιών δεδομένων (data service part) των διάφορων εφαρμογών.



Εικόνα 10: Τοποθέτηση DSWare. [30]

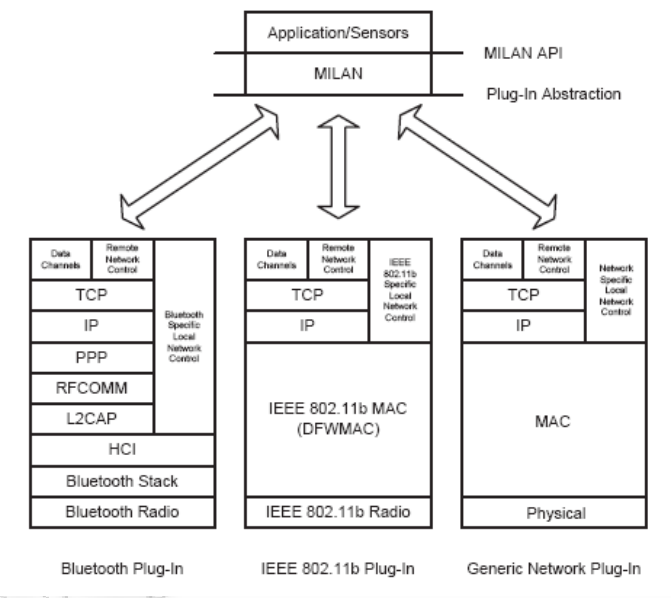
Το DSWare βρίσκεται μεταξύ του επιπέδου της εφαρμογής και του επιπέδου του δικτύου (Εικόνα 10). Σε αυτήν την αρχιτεκτονική, η δρομολόγηση έχει διαχωριστεί

και από το DSWare και από το επίπεδο του δικτύου, αφού τα τμήματα διαχείρισης των ομάδων και του χρόνο-προγραμματισμού (group management and scheduling components) του DSWare, μπορούν να χρησιμοποιηθούν για να βελτιώσουν την απόδοση στην εξοικονόμηση ενέργειας και στην παροχή υπηρεσιών πραγματικού χρόνου (enhance the power-awareness and real-time-awareness) των πρωτοκόλλων δρομολόγησης.

5.8. Προσέγγιση βασισμένη στις εφαρμογές.

5.8.1. MiLAN.

Το MiLAN (Middleware Linking Applications and Networks) είναι μια άλλη προσέγγιση στο χώρο των middleware. Δέχεται τις ανάγκες απόδοσης της εφαρμογής, παρακολουθεί τις συνθήκες του δικτύου και ρυθμίζει τη διαμόρφωση του δικτύου προς όφελος της εφαρμογής.



Εικόνα 11: Στοίβα πρωτοκόλλου του MILAN. [31]

Το MiLAN διαφοροποιείται από τα κλασσικά middleware, που βρίσκονται μεταξύ της εφαρμογής και του επιπέδου του δικτύου. Σκοπεύει στον έλεγχο του δικτύου

και έχει μια αρχιτεκτονική που εξαπλώνεται στη στοίβα πρωτοκόλλου του δικτύου (Εικόνα 11).

Η διεπαφή του MiLAN (MiLAN API) προς την εφαρμογή και τα χαμηλού επιπέδου συστατικά (low-level components) εκτελεί δύο εργασίες : α) επιτρέπει στην εφαρμογή να παρέχει μια γραφική παράσταση (graph) όπου καθορίζονται οι ιδιότητες (utilities) των συστατικών χαμηλού επιπέδου, που είναι διαθέσιμες στην διάρκεια του χρόνου και β) ένα μηχανισμό ώστε το MiLAN να μπορεί να ελέγχει τα συστατικά αυτά.

Το MiLAN θα πρέπει να έχει μια απεικόνιση του κόστους για την χρήση ενός συνόλου αισθητήριων κόμβων δεδομένου του πρωτοκόλλου επικοινωνίας που χρησιμοποιεί. Αυτό επιτυγχάνεται με την χρήση δικτυακών προτύπων που καθορίζουν υπαρκτά σύνολα κόμβων για κάθε ένα από τα οποία έχει υπολογιστεί το επικοινωνιακό κόστος, ανάλογα με το πρωτόκολλο επικοινωνίας που χρησιμοποιείται και τον τύπο του δικτύου που έχει εγκατασταθεί. Παράλληλο θέμα που διαχειρίζεται το MiLAN αποτελεί η συνεχής παρακολούθηση του δικτύου για την εύρεση νέων κόμβων και για να γνωρίζει πότε κάποιοι κόμβοι δεν είναι διαθέσιμοι (λόγω κινητικότητας ή λόγω έλλειψης ενέργειας).

5.9 Χωρίς ταξινόμηση.

5.9.1. Role-Based Middleware.

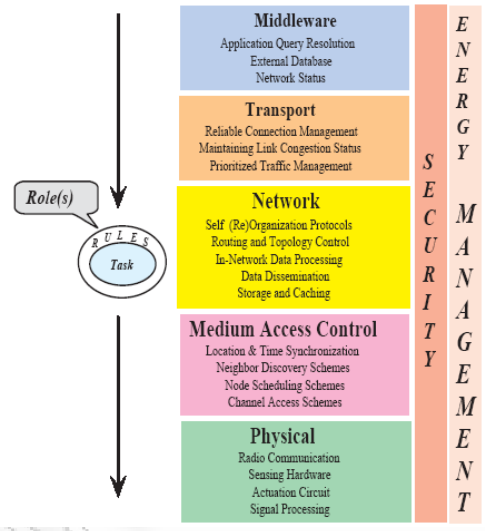
Το πλαίσιο που βασίζεται σε ρόλους αντιστοιχεί ένα χαρακτηριστικό μιας υπηρεσίας εφαρμογής, που αναπαρίσταται σαν App{Service,QoS}, σε διάφορους ρόλους που αναπαρίστανται σαν Roles{ProtocolTask,Rules}.

Η ευέλικτη φύση της αφαίρεσης μεταξύ των υπηρεσιών της εφαρμογής και των ενδοδικτυακών ρόλων, επιτρέπει σε διαφορετικές εφαρμογές να περιγράψουν σημασιολογικά την ίδια υπηρεσία με διάφορες λεπτομέρειες στο βασισμένο σε ρόλους ενδιάμεσο λογισμικό (role-based middleware). Από την άποψη των χαρακτηριστικών της υπηρεσίας, απαιτούνται κάποια τμήματα μιας ισχυρής γλώσσας που να επιτρέπουν ταυτόχρονα την γενικότητα και την λεπτομέρεια, αλλά και είναι καθολικές (universal) όσον αφορά στην χρήση τους από πολλαπλές εφαρμογές και υπηρεσίες.

Αυτή η απαίτηση δεν σταματά στο επίπεδο της εφαρμογής. Το ενδιάμεσο λογισμικό πρέπει να παρέχει αποτελεσματικά ανταλλάγματα μεταξύ των πόρων

και της ακρίβειας των αποτελεσμάτων, με την μέθοδο των *tuning knobs*. Για παράδειγμα το περιβάλλον δηλώσεων της εφαρμογής του πεδίου της μάχης, για μια υπηρεσία συγκερασμού δεδομένων που αφορά σε ένα συμβάν ανίχνευσης ενός εχθρικού άρματος μάχης, μπορεί να εκφραστεί σημασιολογικά σαν ένα πρόγραμμα εφαρμογής. Αυτές οι προκαταρκτικές σημασιολογίες της εφαρμογής έχουν ουσιαστικό περιβάλλον δηλώσεων για το περιβάλλον, τον αισθητήριο στόχο (ή το συμβάν), τις σχετικές παραμέτρους του δικτύου και μετρήσεις ποιότητας ειδικά για την λειτουργία του συγκερασμού των δεδομένων.

Μια βασική υπόθεση που σχετίζεται με το σχεδιασμό ενός γενικού πλαισίου βασισμένου σε ρόλους (role-based framework) είναι η απαίτηση για ύπαρξη ανοιχτών διεπαφών μεταξύ των διάφορων επιπέδων της στοίβας πρωτοκόλλου των δικτύων αισθητήρων. Οι ανοικτές διεπαφές απαιτούνται για το διαμοιρασμό : α) της κατάστασης του δικτύου από την άποψη των δυναμικών δυνατοτήτων των κόμβων και β) των διαθέσιμων πρωτοκόλλων του δικτύου που χρησιμοποιούνται από το ενδιάμεσο λογισμικό (middleware) για να τυποποιήσει τους ρόλους. Οι τεχνολογικοί περιορισμοί των κόμβων αισθητήρων, προς το παρόν, απαιτούν τα πρωτόκολλα να χρησιμοποιούν αποτελεσματικά τους πόρους. Αυτό έχει οδηγήσει στην ανάπτυξη λειτουργικών συστημάτων όπως TinyOS που υιοθετούν ένα σχεδιασμό βασισμένο σε συστατικά (component oriented design) για να επιτευχθούν οι παραπάνω στόχοι.



Εικόνα 12: Κατάσταση του δικτύου από την άποψη του ρόλου. [37]

Η Εικόνα 12 παρουσιάζει έναν ρόλο ο οποίος συγκεντρώνει για έναν κόμβο και τους γείτονες του σημαντικές πληροφορίες που αφορούν στο δίκτυο καθώς διαπερνά τα διάφορα στρώματα μιας υποθετικής στοίβας πρωτοκόλλου.

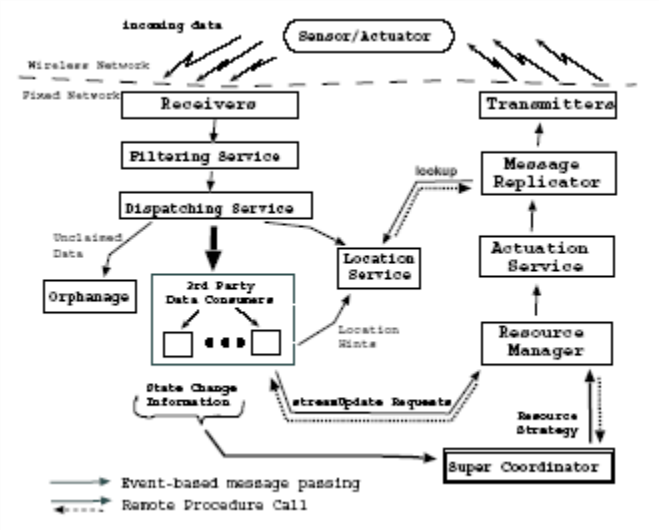
5.9.2. Garnet Middleware.

Το Garnet είναι ένα κατανεμημένο ενδιάμεσο λογισμικό, με αρχιτεκτονική κατασκευασμένη σε επίπεδα με έμφαση στην διαχείριση ροών δεδομένων. Επιτρέπει στον πλέον ανειδίκευτο να ασκήσει δυναμικό έλεγχο των αισθητήρων και να επηρεάσει την διαδικασία παράδοσης δεδομένων. Αυτό είναι επιθυμητό αφού η γνώση στο επίπεδο της εφαρμογής μπορεί να χρησιμοποιηθεί για να βελτιώσει την όλη λειτουργία του δικτύου. Σε περίπτωση που οι καταναλωτές εκμεταλλεύονται περισσότερο από το επιτρεπτό τις υπηρεσίες χειρισμού των δεδομένων, εφαρμόζεται κατάλληλη πολιτική από το ενδιάμεσο λογισμικό. Το Garnet έχει δημιουργηθεί προκειμένου να επιτυγχάνει τις κάτωθι λειτουργίες :

- α) Αφαίρεση (abstraction) για την οντότητα κόμβος και τη θέση
- β) Μηχανισμούς για την αναμετάδοση των μηνυμάτων ελέγχου για να επιτύχει τις δυναμικές αλλαγές στις ρυθμίσεις και την συμπεριφορά του συστήματος.
- γ) Απλούς, εύκαμπτους και ασφαλείς μηχανισμούς για την πρόσβαση στα δεδομένα.
- δ) Μικρό κόστος απόδοσης, εξελίξιμος σχεδιασμός και μηχανισμός που υποστηρίζει ποιότητα υπηρεσίας.

Αρχιτεκτονική στο Garnet.

Η Εικόνα 13 απεικονίζει ένα απλοϊκό διάγραμμα των υπηρεσιών που ορίζονται από το Garnet για την επεξεργασία των ροών δεδομένων των αισθητήρων.



Εικόνα 13: Αρχιτεκτονική του Garnet. [32]

Οι αισθητήρες εκπέμπουν δεδομένα μέσω ενός περιορισμένης εμπιστοσύνης ασύρματου δικτύου προς ένα σταθερό δίκτυο. Στο σταθερό δίκτυο, οι ροές των δεδομένων παραλαμβάνονται και χρησιμοποιούνται (θα χρησιμοποιηθεί ο όρος καταναλώνονται όπως και στο πρωτότυπο) από τις εφαρμογές οι οποίες χρησιμοποιούν συνήθεις μηχανισμούς (διαφήμισης, ανακάλυψης, εγγραφής, πιστοποίησης και διανομής), για να αναγνωρίσουν, να εγγραφούν και να παραλάβουν τις ροές δεδομένων που τους ενδιαφέρουν. Οι διαδικασίες κατανάλωσης (consumer processes) μπορούν να επηρεάσουν το μελλοντικό περιεχόμενο των ροών δεδομένων, κάνοντας επίκληση σε υπηρεσίες του ενδιάμεσου λογισμικού.

Συστατικά του Garnet.

Αισθητήρες και μηχανισμοί κίνησης (Sensors/ Actuators) : Οι αισθητήρες εκπέμπουν μηνύματα δεδομένων τα οποία λαμβάνονται από ένα πίνακα αποδεκτών. Λόγω της κίνησης τους έξω από την περιοχή λήψης είναι πιθανό να έχουμε απώλεια δεδομένων.

Αποδέκτες(Receivers) : Αυτοί τοποθετούνται έτσι ώστε να επικαλύπτονται οι περιοχές αποτελεσματικής λήψης. Αυτό βελτιώνει την λήψη των μηνυμάτων αλλά προκαλεί πιθανό διπλασιασμό τους.

Υπηρεσία Φιλτραρίσματος και υπηρεσία διανομής (Filtering Service, Dispatching Service) : Η υπηρεσία φιλτραρίσματος (filtering service) φιλτράρει τις ροές των δεδομένων εξαφανίζοντας τυχόν διπλά μηνύματα. Κατόπιν προωθεί τα δεδομένα στην υπηρεσία διανομής (dispatching service) η οποία φροντίζει για την παράδοση τους στις εγγεγραμμένες διεργασίες κατανάλωσης.

Διεργασίες και υπηρεσίες κατανάλωσης(Consumer Processes and Services): Οι διεργασίες κατανάλωσης (consumer processes) χρησιμοποιούν ένα μηχανισμό δημοσίευσης/ εγγραφής (publish/ subscribe mechanism) για να έχουν πρόσβαση στις ροές των δεδομένων (data streams), ο οποίος επιτρέπει να ανιχνευθούν οι μη-διαμορφωμένες ροές των δεδομένων (un-configured data streams).

Υπέρ συντονιστής (Super Coordinator): Οι πολύπλοκες διαδικασίες κατανάλωσης (consumer processes) μπορούν να προωθήσουν λεπτομέρειες για την αλλαγή της κατάστασης προς τον υπέρ-συντονιστή (super coordinator), ο οποίος τελικά συσσωρεύει μια συνολική άποψη γι' αυτές. Σε απάντηση αυτής της άποψης ο υπέρ-συντονιστής μπορεί να επικαλεσθεί αλλαγές στην χρησιμοποιούμενη από τον διαχειριστή των πόρων πολιτική.

Διαχειριστής Πόρων, Υπηρεσία Κίνησης (Resource manager, Actuation Service) : Υπάρχει ένα συγκεκριμένο μονοπάτι, προκειμένου οι διαδικασίες κατανάλωσης (consumer processes) να εκπέμψουν μηνύματα ελέγχου προς τους αισθητήρες με τρόπο ουδέτερο προς την θέση. Κατ' αρχάς, επιδιώκεται η έγκριση από το διαχειριστή των πόρων (resource manager) που ασκεί τον έλεγχο των επιτρεπόμενων ενεργειών που μπορεί να ζητήσει ένα σύνολο καταναλωτών. Κατόπιν η υπηρεσία κίνησης (Actuation Service) επεξεργάζεται τις αιτήσεις με χρονοσφραγίδες (timestamps) και ελέγχους αθροισμάτων (check-sums), προτού τις προωθήσει στον αντιγραφέα μηνυμάτων.

Αντιγραφέας Μηνυμάτων, Πομποί (Message Replicator, Transmitters) : Ο αντιγραφέας μηνυμάτων (message replicator) καθορίζει την αναμενόμενη περιοχή του αισθητήρα στόχου. Βασιζόμενοι στην περιοχή αυτή, ένα κατάλληλο σύνολο των πομπών (transmitters) εκπέμπουν καθολικά (broadcast) την αίτηση, που μπορεί να παραληφθεί από τον αισθητήριο κόμβο.

5.9.3. MIRES.

Το Mires ενσωματώνει χαρακτηριστικά ενός προσανατολισμένου στα μηνύματα ενδιάμεσου λογισμικού το οποίο επιτρέπει στις εφαρμογές να επικοινωνούν με ένα τρόπο διαφήμισης/ εγγραφής (in a publish/subscribe way). Επίσης το mires ενσωματώνει πρωτοκολλά του επιπέδου δικτύου (πρωτόκολλα ελέγχου δρομολόγησης και τοπολογίας) και παρέχει ένα υψηλού επιπέδου API το οποίο διευκολύνει την ανάπτυξη των εφαρμογών.

Τα στοιχεία κλειδιά στην επικοινωνία διαφήμισης/ εγγραφής (publish/ subscribe) είναι η *υπηρεσία ενημέρωσης* (notification service) και η *προσωρινή μνήμη* (buffer), στην οποία αποθηκεύονται τα μηνύματα σε ουρά προτού διανεμηθούν στους συνδρομητές. Η υπηρεσία ενημέρωσης είναι υπεύθυνη για να ενημερώσει τους συνδρομητές ότι έφθασε ένα νέο μήνυμα. Με αυτόν τον τρόπο έχουμε μια πλήρως ασύγχρονη επικοινωνία μεταξύ των αποστολέων και των παραληπτών των μηνυμάτων, που είναι και το κύριο πλεονέκτημα αυτού του είδους επικοινωνίας.

Η επικοινωνία μεταξύ των κόμβων εξελίσσεται σε 3 φάσεις : α) Αρχικά, οι κόμβοι στο δίκτυο διαφημίζουν τα διαθέσιμα θέματα (available topics) (π.χ. θερμοκρασία, υγρασία κτλ.) τα οποία έχουν συλλέξει. β) Τα διαφημιστικά μηνύματα δρομολογούνται προς τον κόμβο συγκεντρωτή (sink) χρησιμοποιώντας ένα αλγόριθμο δρομολόγησης πολλαπλών αλμάτων (multi-hop). Η εφαρμογή του χρήστη που είναι συνδεδεμένη στον συγκεντρωτή (sink) μπορεί να επιλέξει μεταξύ των διαφημιστικών θεμάτων (advertising topics) ποια θα παρακολουθήσει. γ) Τα εγγεγραμμένα μηνύματα εκπέμπονται καθολικά (broadcasted) στους κόμβους του δικτύου. Αφού οι κόμβοι συλλέξουν τα εγγεγραμμένα θέματα, οι κόμβοι μπορούν να διανείμουν τα συλλεγμένα δεδομένα στο δίκτυο.

5.9.4. IMPALA.

Πρόκειται για μια αρχιτεκτονική που καθιστά δυνατή την συναρμολογησιμότητα, την προσαρμοστικότητα, και την δυνατότητα επισκευής στις εφαρμογές που εκτελούνται σε ασύρματα δίκτυα αισθητήρων. Το Impala επιτρέπει να λαμβάνονται ενημερώσεις λογισμικού μέσω του πομποδέκτη του κόμβου και να εφαρμόζονται δυναμικά στο εκτελούμενο σύστημα. Επιπλέον προσφέρει προσαρμογή on-the-fly προκειμένου να βελτιωθεί η απόδοση, η αποτελεσματικότητα χρήσης ενεργείας και η αξιοπιστία του λογισμικού του συστήματος.

Το Impala είναι μέρος του σχήματος ZebraNet το οποίο βοηθά στην βελτίωση της τεχνολογίας παρακολούθησης μέσω ενεργώς αποτελεσματικά κόμβων και τεχνικές επικοινωνίας peer-to-peer.

Το Impala έχει έτοιμες βιβλιοθήκες με προγραμματιστικές εφαρμογές (programming utilities) όπως εφαρμογές δικτύου (network utilities), χρονομετρών (timer utilities). Οι εφαρμογές δικτύου (network utilities) επιτρέπουν στις εφαρμογές (applications) να στέλνουν ασύγχρονα μηνύματα και να λαμβάνουν ειδοποίηση όταν αυτά ολοκληρώνονται με ένα κατάλληλο συμβάν (`send done`) από το φίλτρο συμβάντων (event filter). Οι εφαρμογές χρονομέτρη (timer utilities) επιτρέπουν στις εφαρμογές (applications) να θέσουν κάποιους χρονομέτρες για διάφορους σκοπούς π.χ. να στείλουν κάποιο μήνυμα σε κάποια καθορισμένη ώρα ή σε συγκεκριμένα διαστήματα. Ένας χρονομέτρης μπορεί να οριστεί, να μηδενιστεί ή να ακυρωθεί πολλές φορές. Οι εφαρμογές των συσκευών (device utilities) επιτρέπουν στις εφαρμογές (applications) να έχουν κάποιο βαθμό ελέγχου στις συσκευές του υλικού, π.χ. να θέτουν εντός και εκτός λειτουργίας τον πομποδέκτη.

5.9.5. DFuse.

Το DFuse είναι μια αρχιτεκτονική για προγραμματισμό συγχωνευμένων εφαρμογών (fusion applications). Υποστηρίζει κατανεμημένη συγχώνευση δεδομένων με αυτόματη διαχείριση της τοποθέτησης και μετανάστευσης των τμημάτων συγχώνευσης (fusion point), για την βελτιστοποίηση μιας δοθείσας συνάρτησης κόστους (όπως η μακροζωία του δικτύου). Χρησιμοποιώντας το DFuse οι προγραμματιστές των εφαρμογών πρέπει μόνο να υλοποιήσουν τις συναρτήσεις συγχώνευσης (fusion functions) και να παρέχουν τον γράφο ροής των δεδομένων (dataflow graph). Τα κύρια κομμάτια του DFuse είναι :

α) *API συγχώνευσης (Fusion API)* : Έχει υλοποιηθεί ένα API που επιτυγχάνει τον εύκολο προγραμματισμό για την ανάπτυξη πολύπλοκων εφαρμογών συγχώνευσης για αισθητήρες. Επιτρέπει οποιαδήποτε λειτουργία σύνθεσης να οριστεί στη ροή των δεδομένων σαν συνάρτηση συγχώνευσης, από απλό συγκερασμό (min, max, avg) ως πιο σύνθετες διαδικασίες.

β) *Κατανεμημένος αλγόριθμος (distributed algorithm)* για την τοποθέτηση και δυναμική μετεγκατάσταση συναρτήσεων συγχώνευσης : Υπάρχει ένας μεγάλος αριθμός επιλογών για την τοποθέτηση των συναρτήσεων συγχώνευσης στο δίκτυο. Έτσι είναι δύσκολη η εύρεση της βέλτιστης συνάρτησης τοποθέτησης που να ελαχιστοποιεί το κόστος επικοινωνίας. Επίσης η τοποθέτηση πρέπει να

ξανά-αποτιμηθεί αρκετά συχνά αν λάβουμε υπόψη το δυναμικό περιβάλλον των δικτύων αισθητήρων. Έχει αναπτυχθεί ένας ευρετικός αλγόριθμος ο οποίος

χρησιμοποιείται για να βρει μια «καλή» απεικόνιση των συναρτήσεων κόστους προς τους αισθητήριους κόμβους. Η απεικόνιση αυτή αποτιμάται σε συχνά διαστήματα ώστε να ανταποκρίνεται στις αλλαγές της ενεργείας των κόμβων αλλά και της τοπολογίας του δικτύου. Το DFuse βλέπει την εφαρμογή σαν ένα γράφο στόχων (task graph) του οποίου οι κορυφές μπορεί να είναι μια *πηγή δεδομένων* (*data source*), ένα *σημείο συγκέντρωσης δεδομένων* (*data sink*) ή ένα *σημείο συγχώνευσης* (*fusion point*).

Μια πηγή (*data source*) είναι ένας αισθητήρας ή μια εφαρμογή που παράγει δεδομένα, ένα σημείο συγκέντρωσης (*data sink*) είναι ο τελικός χρήστης (εφαρμογή ή άνθρωπος) και ένα σημείο συγχώνευσης (*fusion point*) κάποια συσκευή η πρόγραμμα που εκτελεί κάποια επεξεργασία συγχώνευσης στα δεδομένα και τα δρομολογεί.

5.9.6. EnviroTrack.

Το EnviroTrack είναι ένα κατανεμημένο αντικειμενοστραφές ενδιάμεσο λογισμικό το οποίο παρέχει μια δυναμική διεπαφή προς τους σχεδιαστές που παράγουν εφαρμογές για την παρακολούθηση του φυσικού περιβάλλοντος. Καινοτομεί στον τρόπο ενσωμάτωσης των φυσικών αντικειμένων που λειτουργούν στον πραγματικό χρόνο και χώρο, μέσα στο υπολογιστικό περιβάλλον της εφαρμογής. Το EnviroTrack εξάγει μια καινούρια κλάση αφαίρεσης που ονομάζεται *παρακολουθούμενα αντικείμενα* (*tracking objects*), που μπορούν να επικολληθούν λογικά σε επιλεγμένες οντότητες του φυσικού περιβάλλοντος για σκοπούς ανίχνευσης (tracking purposes). Κάθε τέτοιο αντικείμενο ενσωματώνει την συνολική κατάσταση της παρακολουθούμενης οντότητας. Καθώς η παρακολουθούμενη οντότητα κινείται, η ταυτότητα και η θέση των αισθητήριων κόμβων αλλάζει, αλλά το παρακολουθούμενο αντικείμενο (tracking object) παραμένει το ίδιο. Έτσι ο προγραμματιστής αλληλεπιδρά με ένα σύνολο αισθητήρων που αλλάζει στο χρόνο αλλά μέσω μιας απλής διεπαφής ενός αντικειμένου.

Τα αντικείμενα παρακολούθησης (tracking objects) που συνδέονται με μια ετικέτα περιεχόμενου αποτελούνται από μεθόδους των οποίων η επίκληση γίνεται είτε χρονικά (time-triggered) είτε με την άφιξη κάποιων μηνυμάτων που περιέχουν αιτήσεις επίκλησης μεθόδων. Ο κώδικας του αντικειμένου εκτελείται σε ένα κόμβο συνήθως στον αρχηγό της ομάδας. Ο κώδικας μπορεί να αναφέρεται στην κατάσταση συγκερασμού (aggregate state) που συντηρείται στο περιεχόμενο.

Αυτή η κατάσταση συλλέγεται με τη χρήση του κατανεμημένου πρωτοκόλλου συλλογής δεδομένων.

Η κατάσταση συγκερασμού (aggregate state) που συλλέχθηκε από μια ομάδα αισθητήρων διατηρείται με ένα σύνολο μεταβλητών που καλούνται *μεταβλητές κατάστασης συγκερασμού* (*aggregate state variables)*. Αυτές οι μεταβλητές και οι τύποι τους δηλώνονται στον ορισμό του τύπου περιεχομένου. Ο ορισμός μιας μεταβλητής κατάστασης ορίζει 3 σημαντικές πληροφορίες :

α) *Συνάρτηση συγκερασμού (aggregate function)* : παράγει βαθμιδωτές τιμές από τις ενδείξεις ενός συνόλου αισθητήρων. Στο EnviroTrack παρέχονται αρκετές συναρτήσεις συγκερασμού καθώς και μηχανισμοί για τη δημιουργία καινούριων.

β) *Νεότητα (Freshness)* :Το όριο νεότητας ενημερώνει το σύστημα για πόσο διάστημα οι ενδείξεις των αισθητήρων μπορούν να χρησιμοποιούνται. Κατόπιν αυτές θεωρούνται παρωχημένες.

γ) *Κρίσιμη Μάζα (Critical mass)* :Η κρίσιμη μάζα είναι ένας ακέραιος που δηλώνει τον ελάχιστο αριθμό κόμβων που πρέπει να αναμιχθούν στον υπολογισμό της συγκερασμένης τιμής προκειμένου να θεωρείται έγκυρη. Θα πρέπει όμως πρώτα οι ενδείξεις των αισθητήρων να έχουν περάσει το όριο νεότητας.

Χρησιμοποιώντας τους παραπάνω 3 περιορισμούς ο αρχηγός κόμβος κάθε ομάδας λαμβάνει τις ενδείξεις από τους κόμβους τις οποίες συγκεράζεται και σηματοδοτεί με μια σημαία (flag) ως έγκυρες. Τα παραπάνω έχουν ως αποτέλεσμα η τιμή που εξάγει ο αρχηγός κόμβος να έχει τις παρακάτω ιδιότητες :

- Οι κόμβοι που συμμετείχαν για τον υπολογισμό αυτής της τιμής ήταν όλοι μέλη της ομάδας.
- Υπολογίστηκε εντός του χρονικού ορίου της νεότητας.
- Ο αριθμός των κόμβων που έλαβαν μέρος στον υπολογισμό ήταν μεγαλύτερος από το όριο της κρίσιμης μάζας.

5.10 Σύνοψη και συγκρίσεις.

Το SensorWare απευθύνεται σε πλούσιες πλατφόρμες τόσο υπολογιστικά όσο και ενεργειακά. Βρίσκεται σε λογικό επίπεδο μεταξύ των εφαρμογών και των προγραμμάτων και του λειτουργικού συστήματος του κάθε κόμβου. Συνεργάζεται με το λειτουργικό σύστημα χωρίς να το υποκαθιστά.

Το TinyDB δημιουργήθηκε για να χρησιμοποιηθεί μαζί με το TinyOS και αντιμετωπίζει το δίκτυο σαν έναν πίνακα. Χρησιμοποιεί γλώσσα περίπου σαν την SQL για τον προγραμματισμό αλλά και την λειτουργία. Δημιουργείται μια ιεραρχία δέντρου όπου στέλνονται τα ερωτήματα και επιστρέφουν οι απαντήσεις. Κάθε κόμβος προκειμένου να μειώσει τον όγκο των εκπεμπόμενων δεδομένων κάνει τοπική επεξεργασία προτού μεταδώσει τα δεδομένα. Λόγω των μεγάλων κενών χρονικών διαστημάτων στην λειτουργία των αισθητήρων γίνεται μεγάλη οικονομία ενέργειας. Το COUGAR χρησιμοποιεί μια 3 επιπέδων αρχιτεκτονική. Μέσα στο δίκτυο δημιουργούνται ομάδες των οποίων ο κάθε αρχηγός εκτελεί συγκερασμό δεδομένων προτού εκπέμψει τις απαντήσεις, αλλά έχει και την ευθύνη διάχυσης των ερωτημάτων. Το SINA χρησιμοποιεί κι αυτό ομαδοποίηση με τον αρχηγό της ομάδας να συμπεριφέρεται όπως και στο COUGAR. Η διαφορά είναι ότι οι ομάδες στο COUGAR δημιουργούνται κατά την φάση που τίθεται ένα ερώτημα και με βάση τα όρια που θέτει ο χρήστης. Στο SINA υπάρχουν 2 τρόποι. Ο πρώτος είναι να δημιουργηθούν οι ομάδες με βάση το παρακολουθούμενο αντικείμενο ενώ ο δεύτερος που χρησιμοποιείται για γεωγραφικές εφαρμογές είναι να δημιουργηθούν οι ομάδες με την εγκατάσταση του δικτύου. Στο SINA χρησιμοποιείται η πληροφορία της θέσης των αισθητήρων κάτι που μπορεί να γίνει και στο COUGAR χωρίς να είναι και αναγκαίο. Επίσης στο SINA χρησιμοποιείται μια ιεραρχική ομαδοποίηση, σαν δέντρο, και ονοματοδοσία με βάση χαρακτηριστικά (attribute-based naming). Τα δεδομένα στο δίκτυο οργανώνονται σαν φύλλα δεδομένων (spreadsheet) με κάθε κελί (cell) να φιλοξενεί και ένα χαρακτηριστικό γνώρισμα. Παρέχει ευρωστία και διόρθωση λαθών κάτι που δεν έχει καταφέρει ακόμα το COUGAR και έχει αφεθεί για μελλοντικές βελτιστοποιήσεις. Μια ακόμα διαφορά είναι η γλώσσα προγραμματισμού που ενώ στο SINA είναι η SQTL, στο COUGAR είναι η SQL όταν τίθενται ερωτήματα και η XML για να διαχυθούν και να απαντηθούν από το δίκτυο. Το TinyDB είναι ένα middleware που παρέχει υποστήριξη προς δίκτυα αισθητήρων που κάνουν χρήση της διαδεδομένης πλατφόρμας TinyOS. Το γεγονός αυτό μπορεί να προσφέρει μια μικρή υπεροχή έναντι άλλου ενδιάμεσου λογισμικού. Βέβαια και το COUGAR τυγχάνει ευρείας αποδοχής αν και απευθύνεται σε πιο πλούσιες πλατφόρμες. Μειονέκτημα όμως μπορεί να θεωρηθεί ο τρόπος δημιουργίας των ομάδων (με παρέμβαση του χρήστη), διότι υπάρχει πιθανότητα ο χρήστης που θα θέσει το ερώτημα να μην γνωρίζει σε ποιους αισθητήρες θα πρέπει να απευθυνθεί για να λάβει τις απαιτούμενες απαντήσεις. Μειονέκτημα όμως μπορεί να θεωρηθεί και για το SINA το γεγονός ότι η γνώση της θέσης είναι αναγκαία για τη λειτουργία του. Αυτό μπορεί να δημιουργήσει δυσκολίες στην υλοποίηση του, μιας και η χρήση του GPS σε δίκτυα αισθητήρων μπορεί να φέρει πολλά προβλήματα. Στα θέματα της επίγνωσης ενέργειας (power awareness), της επεκτασιμότητας (scalability), της

κινητικότητας (mobility), της ανοικτότητας (openness), και της ευκολίας χρήσης, το TinyDB υπερτερεί των άλλων δυο, υστερώντας όμως στην της ανομοιογένειας (heterogeneity), με το COUGAR να ακολουθεί.

Συνολικά τα παραπάνω middleware χρησιμοποιούν μια πολλά υποσχόμενη και με πολλές εφαρμογές προσέγγιση. Οι βάσεις δεδομένων είναι πολύ διαδεδομένες με αποτέλεσμα να υπάρχει μεγάλη υποστήριξη και συνεπώς να αναπτύσσονται νέες τεχνικές για αναζήτηση και ανάκτηση πληροφοριών από τα δίκτυα αισθητήρων.

Τα DSWare και MILAN είναι οι κύριοι εκφραστές των κατηγοριών όπου ανήκουν γι αυτό και αποτελούν κατά κάποιο τρόπο, μονόδρομο στις αντίστοιχες προσεγγίσεις middleware. Άλλωστε η σύγκριση middlewares που ανήκουν σε διαφορετική κατηγορία δεν μπορεί να προσφέρει βάσιμα αποτελέσματα. Αυτό που μπορούμε να πούμε συνοπτικά είναι ότι το DSWare υπερέχει στο θέμα του power awareness, αλλά υστερεί στο scalability και στο openness, ενώ το MILAN μειονεκτεί στην επίγνωση ενέργειας και πλεονεκτεί στην ανοικτότητα και την επεκτασιμότητα.

Για τα υπόλοιπα middleware που αναφέρθηκαν μπορούμε να πούμε συνοπτικά ότι:

- Το IMPALA παίρνει σχεδόν άριστα σε όλα τα παραπάνω χαρακτηριστικά που αναφέρθηκαν για τα άλλα middleware.
- Το MIRES ακολουθεί από κοντά το IMPALA, αλλά χάνει λίγο στο θέμα της κινητικότητας και της ανομοιογένειας.
- Το EnviroTrack μπορούμε να πούμε ότι είναι στα ίδια επίπεδα με το IMPALA σε όλα τα χαρακτηριστικά.

6

ΠΛΑΤΦΟΡΜΑ OSGi - ΥΛΟΠΟΙΗΣΗ

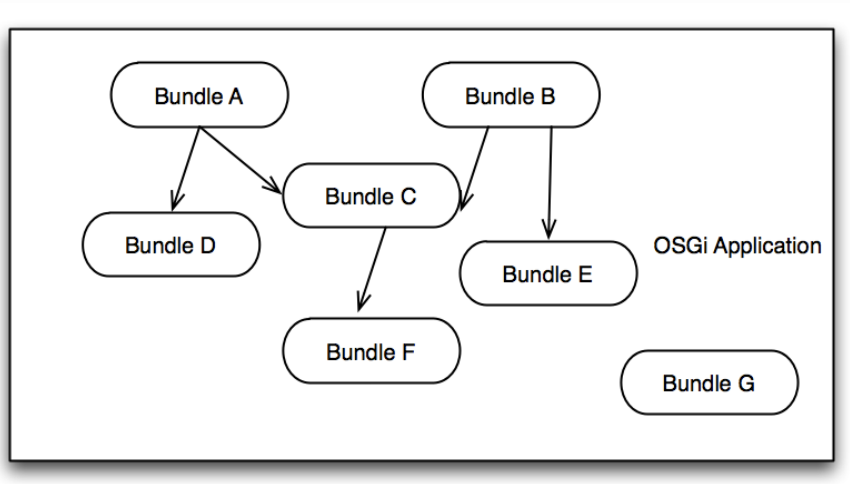
6.1 OSGI - Open Services Gateway Initiative

6.1.1 Εισαγωγή

Το OSGi (Open Services Gateway initiative) ορίζει ένα πλήρες, δυναμικό σύστημα και μια πλατφόρμα υπηρεσιών, για την γλώσσα προγραμματισμού Java.

Ένα OSGi σύστημα είναι μια συλλογή από στοιχεία που ονομάζονται bundles.Τα bundles που τρέχουν στο σύστημα είναι ανεξάρτητα αλλά μπορούν να συνεργάζονται μεταξύ τους. Η συνεργασία τους γίνεται μέσω υπηρεσιών οι οποίες μπορούν να δηλωθούν είτε στατικά και να αναζητηθούν μέσω τους ονόματος τους, είτε δυναμικά κατά την λειτουργία του συστήματος με τους μηχανισμούς που περιγράφονται στην Εικόνα 14. Το κάθε ξεχωριστό bundle περιγράφει πλήρως τον εαυτό του, ορίζει την ανοικτή προγραμματιστική διεπαφή του, τις εξαρτήσεις του σε άλλα bundles και κρύβει την εσωτερική του υλοποίηση.

Όπως φαίνεται στην , μια εφαρμογή OSGi δεν έχει αρχή και τέλος αλλά αποτελεί μια συλλογή από bundles που συνεργάζονται. Σε μια εφαρμογή βασισμένη στο OSGi δεν υπάρχει κύρια μέθοδος από την οποία γίνεται η έναρξη της. Το κάθε ξεχωριστό bundle που υπάρχει στο σύστημα μπορεί να παρομοιαστεί με ένα ομότιμο χρήστη σε ένα συνεργατικό σύστημα.



Εικόνα 14: Εφαρμογή OSGi. [44]

Τα συστήματα που χρησιμοποιούν OSGi είναι τελείως δυναμικά αφού τα bundles που τα αποτελούν μπορούν να αλλάξουν στην διάρκεια ζωής της εφαρμογής. Κάθε ένα από αυτά μπορεί δυναμικά να εγκατασταθεί στο σύστημα, να απεγκατασταθεί και να ενημερωθεί οποιαδήποτε στιγμή. Με αυτά τα χαρακτηριστικά τα συστήματα που χρησιμοποιούν OSGi έχουν τεράστια πλεονεκτήματα ειδικά σε περιπτώσεις που υπάρχουν περισσότερα από ένα άτομα που συνεισφέρουν όπως π.χ. στις εφαρμογές ανοικτού κώδικα. Με την χρήση του OSGi αυξάνεται δραματικά η ευκαιρία για επαναχρησιμοποίηση κώδικα και έτσι ο μειώνεται ο χρόνος ανάπτυξης των εφαρμογών.

Σε ένα σύστημα OSGi μπορούν να εκτελεστούν οι εντολές που παρουσιάζονται στην Εικόνα 15.

ss	Παρουσιάζει μια λίστα με τα bundles που είναι εγκατεστημένα με την πληροφορία για ταυτότητα, όνομα και κατάσταση.
Start <bundleid>	Ξεκινά ένα bundle
Stop <bundleid>	Σταματά ένα bundle
Update <bundleid>	Ανανεώνει ένα με καινούργιο bundle με καινούργιο JAR αρχείο
Install <bundleURL>	Εγκατάσταση νέου bundle στο OSGi container
Uninstall <bundleid>	Απεγκατάσταση ενός εγκατεστημένου bundle

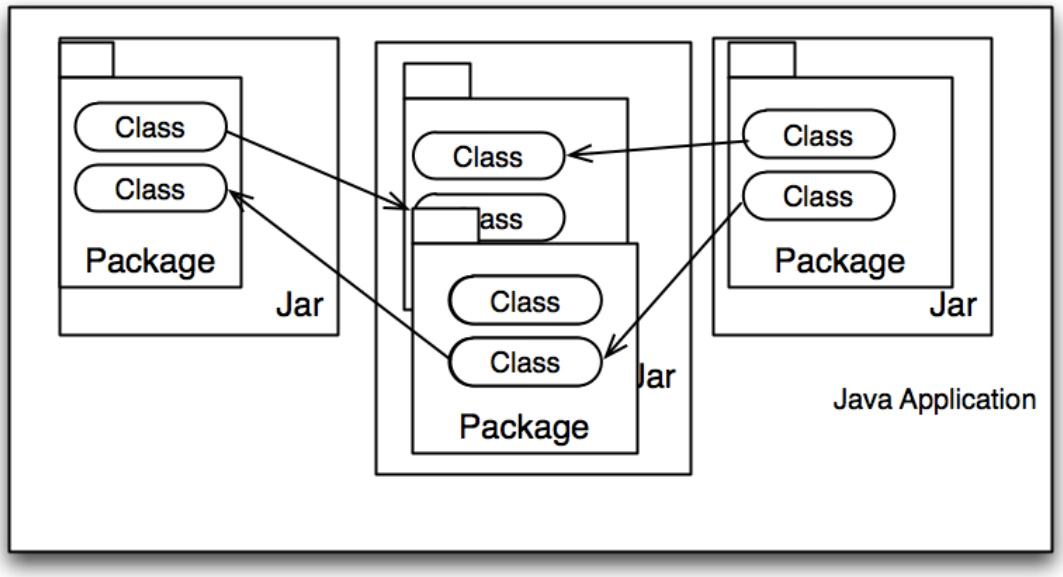
Εικόνα 15: Εντολές κονσόλας OSGi. [46]

6.1.2 Πλεονεκτήματα

Σε μια συνηθισμένη Java εφαρμογή υπάρχουν κλάσεις και διεπαφές (interfaces). Κάθε ένα από αυτά μπορεί να αποτελείται από χαρακτηριστικά και μεθόδους και οργανώνονται σε πακέτα.

Το σύνολο των πακέτων ορίζει ένα παγκόσμιο χώρο ονομάτων και η Java ορίζει τους κανόνες ορατότητας για να διαχειρίζεται τις αλληλεπιδράσεις μεταξύ των τύπων και των μελών τους.

Όπως φαίνεται στην Εικόνα 16, πολλά πακέτα συνήθως κτίζονται και οργανώνονται μαζί σε Java Archives(Jars). Τα jars συλλέγονται σε ένα ενιαίο χώρο ονομάτων ο οποίος αναζητείται σειριακά από το Java Virtual Machine για να ανακαλυφθούν και να φορτωθούν οι κλάσεις,.



Εικόνα 16: Εφαρμογή Java. [47]

Η έννοια των Jars είναι βασική και χρήσιμη αφού ένας παραγωγός μπορεί να δημιουργήσει ένα jar που να υλοποιεί κάποιες λειτουργίες και κάποιος άλλος να το χρησιμοποιήσει αλλά ουσιαστικά είναι ένας απλός μηχανισμός παράδοσης και δεν έχει καμιά επίπτωση με το σύστημα που τρέχει.

Με τα χαρακτηριστικά όμως αυτά δεν υπάρχει υποστήριξη για λειτουργικότητα ορισμού και επιβολής εξαρτήσεων και χωρίς εξαρτήσεις δεν μπορεί να υπάρξει αρθρωτός προγραμματισμός. Όπως φαίνεται από το σχήμα το αποτέλεσμα είναι η δημιουργία μονολιθικών εφαρμογών στις οποίες δεν είναι ξεκάθαρη η σχέση των Jars. Υπάρχει ψηλή σύζευξη μεταξύ τους, πολλαπλές πολυκατευθυνόμενες

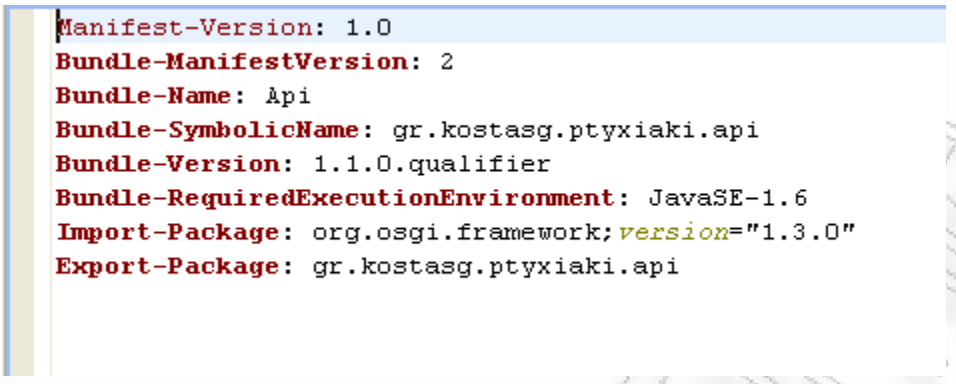
εξαρτήσεις και επιπλέον μπορεί να υπάρχουν κυκλικές εξαρτήσεις. Αυτό έχει ως αποτέλεσμα η συνεργασία και ο διαμερισμός μεταξύ ομάδων να παρεμποδίζεται.

Το OSGi είναι γραμμένο σε java αλλά προσθέτει μερικά βασικά στοιχεία. Συγκεκριμένα αντί για Jars to OSGi μιλά για bundles. Τυπικά ένα bundle υλοποιείται ως ένα Jar αλλά έχει επιπλέον ταυτότητα και πληροφορίες για τις εξαρτήσεις του δηλαδή τα bundles είναι αυτοπεριγραφικά jars.

Αυτό δίνει την ευκαιρία στους δημιουργούς να καθορίσουν τις προθέσεις τους και όταν τρέχει η εφαρμογή αφού υπάρχουν οι απαραίτητες πληροφορίες μπορούν να χρησιμοποιηθούν για να επιβληθούν. Για να είναι διαθέσιμο ένα bundle σε άλλα πρέπει να γίνει εξαχθεί (export) και για να χρησιμοποιηθεί από άλλα πρέπει να εισαχθεί (import). Το OSGi runtime επιβάλει αυτούς τους περιορισμούς, αλλά η εισαγωγή ενός bundle δηλώνει την εξάρτηση σου σε λειτουργικότητα ασχέτως με το bundle που την υλοποιεί. Αυτό δημιουργεί συστήματα με χαμηλή σύζευξη. Κατά τον χρόνο εκτέλεσης του συστήματος επιλύονται οι εξαρτήσεις και συσχετίζονται τα εξαρτώμενα bundles,αυτό επιλύει όλα τα προβλήματα του χώρου ονομάτων και ταυτόχρονα βελτιστοποιεί την απόδοση της εφαρμογής. Σχετικά με τις εξαρτήσεις μεταξύ συνεργαζόμενων bundles γίνεται εισαγωγή και εξαγωγή πακέτων για τις στατικές και για τις δυναμικές χρησιμοποιούνται οι υπηρεσίες. Μια υπηρεσία είναι ένα αντικείμενο που υλοποιεί μια σύμβαση και ένα τύπο και εγγράφεται στο μητρώο των υπηρεσιών του OSGi. Τα bundles που ψάχνουν για μια υπηρεσία ψάχνουν στο μητρώο για να βρουν υλοποιήσεις της. Βασικό πλεονέκτημα είναι ότι το bundle που θα καταναλώσει την υπηρεσία δεν γνωρίζει το bundle που την παράγει ή τον τύπο της υλοποίησης έτσι το σύστημα ενώ είναι σύστημα συνεργασίας παραμένει χαλαρά συνδεδεμένο. Το OSGi μπορεί να θεωρηθεί ως επέκταση στη γλώσσα προγραμματισμού Java που επιβάλει τις εξαρτήσεις και την ορατότητα των πακέτων δυναμικά. Μέσω αυτών των περιορισμών είναι ευκολότερο να υλοποιηθούν εφαρμογές που έχουν χαλαρά συνδεδεμένα τμήματα και με ψηλή συνεκτικότητα.

6.1.3 OSGi Bundle

Ένα bundle είναι μια αυτοπεριγραφική συλλογή από αρχεία. Ο ορισμός των περιεχομένων του και των απαιτήσεων του βρίσκονται μέσα στο αρχείο manifest.mf. Χρησιμοποιεί την σύνταξη του jar manifest και προσθέτει επιπλέον ειδικές OSGi κεφαλίδες. Παράδειγμα του αρχείου manifest.mf του bundle φαίνεται στην Εικόνα 17.

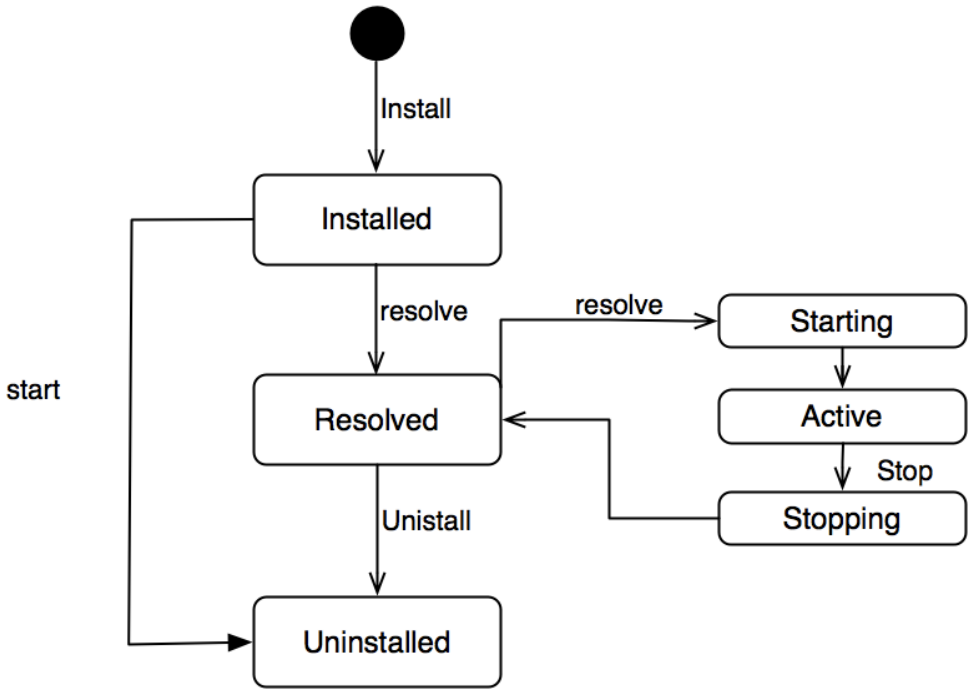


Εικόνα 17: Περιεχόμενα αρχείου MANIFEST.MF.

Υποχρεωτικά πρέπει να υπάρχουν το Bundle-SymbolicName και το Bundle-Version. Οι δύο αυτές επικεφαλίδες χαρακτηρίζουν μοναδικά ένα bundle. Μέσω της επικεφαλίδας Import-Package δηλώνονται οι εξαρτήσεις μεταξύ των bundles.

6.1.4 Κύκλος ζωής

Για να υπάρχει η δυναμικότητα στο σύστημα τα bundles πρέπει να έχουν ξεκάθαρο κύκλο ζωής. Η Εικόνα 18 παρουσιάζει τις φάσεις και τις μεταβάσεις κατά την διάρκεια ζωής ενός bundle. Η ζωή ενός bundle αρχίζει από την κατάσταση installed. Εάν ικανοποιούνται όλες οι εξαρτήσεις που ορίζει μεταφέρεται στην κατάσταση resolved. Εάν γίνει resolved μπορεί να αρχίσει να τρέχει αφού φορτωθούν όλες οι κλάσεις του. Εάν αρχίσει να τρέχει μεταφέρεται στην κατάσταση active. Όταν σταματήσει για οποιοδήποτε λόγο προχωρά στην κατάσταση stopping και μετά επιστρέφει ξανά στην κατάσταση Resolved.



Εικόνα 18: Κύκλος ζωής ενός OSGi bundle. [44]

Όλες οι αλλαγές στις καταστάσεις συμβαίνουν δυναμικά και τα bundles μπορούν να τις παρακολουθούν και να αντιδρούν ανάλογα π.χ. όταν ένα bundle γίνει installed άλλα bundles μπορεί να ενδιαφέρονται για τις λειτουργίες του.

Πριν να μπορέσει ένα bundle να προσφέρει υπηρεσίες ανιχνεύσιμες από άλλα δομοστοιχεία πρέπει:

- Να εγκατασταθεί επιτυχώς.
- Όλες οι κλάσεις Java επάνω στις οποίες βασίζεται να έχουν επιλυθεί.
- Να εκκινήσει και τερματίσει η διαδικασία ενεργοποίησης της υλοποίησης του interface BundleActivator που ισχύει για το bundle (βλ. Εικόνα 19).
- Να έχει ικανοποιηθεί η πολιτική ενεργοποίησης του (ένα σύνολο κανόνων που ορίζει πότε ενεργοποιείται το bundle) .

Όταν ένα bundle είναι ενεργό μπορεί να προσφέρει τις υπηρεσίες που ορίζονται στις ρυθμίσεις του.

```java
package com.example.helloworld;

import org.osgi.framework.BundleActivator;

public class Activator implements BundleActivator {

    /*
     * (non-Javadoc)
     * @see org.osgi.framework.BundleActivator#start(org.osgi.framework.BundleContext)
     */
    public void start(BundleContext context) throws Exception {
        System.out.println("Hello World!!");
    }

    /*
     * (non-Javadoc)
     * @see org.osgi.framework.BundleActivator#stop(org.osgi.framework.BundleContext)
     */
    public void stop(BundleContext context) throws Exception {
        System.out.println("Goodbye World!!");
    }

}
```

Εικόνα 19: Παράδειγμα υλοποίησης του interface BundleActivator.

6.1.5 Συνεργασία μεταξύ bundles.

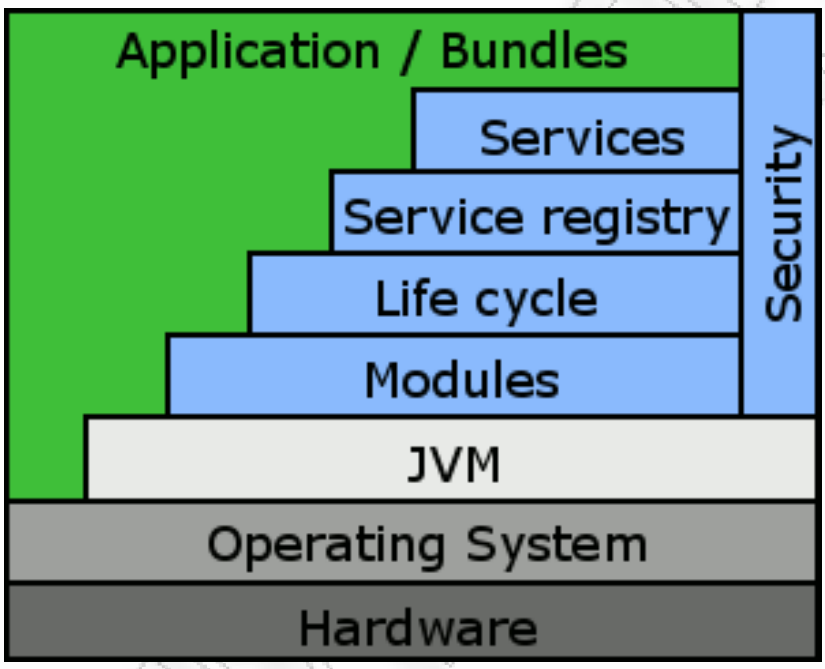
Υπηρεσίες

Για την συνεργασία μεταξύ των bundles χρησιμοποιούνται υπηρεσίες (services) που επιτρέπουν στα bundles να συνεργάζονται χωρίς να εξαρτώνται από συγκεκριμένο πακέτο ή υλοποίηση κάποιου άλλου bundle. Αυτό κάνει τα συστήματα ακόμη πιο ευέλικτα και δυναμικά.

Το OSGi μητρώο υπηρεσιών λειτουργεί ως συντονιστής τριών ομάδων bundles, των bundles που ορίζουν interfaces, των bundles που υλοποιούν τα interfaces και των bundles που αναζητούν, ανακαλύπτουν και χρησιμοποιούν τις υπηρεσίες. Ανάμεσα στις τρεις αυτές ομάδες υπάρχει ανωνυμία αφού το bundle που προσφέρει την υπηρεσία δεν γνωρίζει ποιος την καταναλώνει και αυτός που την καταναλώνει δεν γνωρίζει ποιός την προσφέρει. Για παράδειγμα το bundle C ορίζει ένα interface που υλοποιείται από το B. Το bundle A βρίσκει την υπηρεσία και την χρησιμοποιεί χωρίς να γνωρίζει οτιδήποτε για το bundle B.

Οι υπηρεσίες OSGi προτιμώνται από τις ενδοεξαρτήσεις κλάσεων. Όταν χρησιμοποιείται μια υπηρεσία το bundle εξαρτάται μόνο από το interface της υπηρεσίας και όχι από οποιαδήποτε κλάση υλοποίησης. Μια υπηρεσία δηλαδή μπορεί να αναφερθεί μόνο μέσω του interface της. Με αυτό τον τρόπο τα OSGi services επιτρέπουν σε ένα bundle να μειώσει δραματικά τις εξαρτήσεις του.

Οι στατικές και δυναμικές υπηρεσίες αναφέρονται σε συνεργασία bundles που τρέχουν στο τοπικό σύστημα και η αναζήτηση γίνεται μόνο στο τοπικό μητρώο υπηρεσιών όπως ορίζεται από την αρχιτεκτονική του OSGi και φαίνεται στην Εικόνα 20, σε αντίθεση με τις απομακρυσμένες υπηρεσίες που επεκτείνουν την χρήση και αναζήτηση υπηρεσιών από απομακρυσμένα συστήματα μέσω πρωτοκόλλων που υλοποιούν πρόσβαση σε απομακρυσμένα μητρώα υπηρεσιών και κατάλληλες μεθόδους πρόσβασης στις απομακρυσμένες υπηρεσίες.



Εικόνα 20: Επίπεδα τοπικής αρχιτεκτονικής OSGi. [45]

Στατικές Υπηρεσίες

Η πρώτη κατηγορία μηχανισμού για εξαρτήσεις αφορά τις στατικές εξαρτήσεις υπηρεσιών. Το bundle που προσφέρει την υπηρεσία δηλώνει ένα αντικείμενο ως υπηρεσία σε ένα ή περισσότερα interfaces. Με αυτό τον τρόπο εγγράφονται στο κιβώτιο OSGi τα interfaces και οι υπηρεσίες που τις υλοποιούν. Το bundle που αναζητά μια υπηρεσία για να χρησιμοποιήσει την υλοποίηση της, αναζητά στο κιβώτιο OSGi για υπηρεσίες κάτω από ένα συγκεκριμένο interface. Όταν βρεθεί το interface, το bundle προσδένεται μαζί του και μπορεί να αρχίσει να καλεί τις μεθόδους του.

Δυναμικές Υπηρεσίες

Η δεύτερη κατηγορία μηχανισμού για εξαρτήσεις αφορά τις δυναμικές υπηρεσίες δηλαδή όταν το bundle-καταναλωτής χρειάζεται δυναμικά να γνωρίζει την ύπαρξη αντικείμενου κάτω από ένα συγκεκριμένο interface. Για τις δυναμικές υπηρεσίες οι βασικοί μηχανισμοί είναι οι εξής: Service Tracker_Λειτουργεί αποδοτικά όταν οι υπηρεσίες χρησιμοποιούνται περιοδικά και όχι εάν όλες σε κάποια στιγμή θα χρειαστούν. Service Activator Toolkit –SAT Όταν δεν είναι πολύ μεγάλος ο αριθμός των bundles είναι αποδοτικό. Όταν είναι πολλά επειδή πρέπει να γίνει φόρτωση και εκκίνηση όλων των activators για να διαπιστώσει εάν υπάρχουν οι υπηρεσίες που χρειάζεται δεν είναι αποδοτικό. Από μελέτη αποδείχτηκε ότι ο αριθμός των bundles πρέπει να είναι περίπου 100.

OSGi Declarative Services

Είναι ο πιο χρησιμοποιημένος μηχανισμός. Λειτουργεί αποδοτικά όταν τα bundles ξεκινούν περιοδικά. Στον μηχανισμό αυτό γίνεται δηλωτική περιγραφή των μεθόδων που χρησιμοποιεί το SAT. Τα Declarative Services αποτελούνται από συνθετικά μέρη (components) τα οποία περιέχουν ένα XML αρχείο(component.xml) και μία κλάση που υλοποιεί τις υπηρεσίες services και παραλαμβάνει referenced services.

Απομακρυσμένες Υπηρεσίες

Υπάρχει επιπλέον η δυνατότητα για χρήση απομακρυσμένων υπηρεσιών μέσω του interface Remote OSGi Service. Μέσω ακροατών υπηρεσιών (listeners) όταν βρεθεί μια υπηρεσία, το interface μεταφέρεται στον κόμβο. Ο προεπιλεγμένος τρόπος είναι η δημιουργία ενδιάμεσου εξυπηρετητή(proxy) μέσω του οποίου μπορούν να καλούνται οι μέθοδοι της υπηρεσίας στον απομακρυσμένο κόμβο. Εσωτερικά οι κλήσεις γίνονται χρησιμοποιώντας απομακρυσμένη κλήση μεθόδων.

Επεκτάσεις

Οι υπηρεσίες για την συνεργασία μεταξύ bundles είναι αποδοτικός και ελαφρύς μηχανισμός αλλά υπάρχουν και άλλοι μηχανισμοί με τα δικά τους πλεονεκτήματα όπως οι επεκτάσεις.

Για τον μηχανισμό των επεκτάσεων υπάρχει το μητρώο επεκτάσεων (ανάλογο του μητρώου υπηρεσιών).Τα bundles είναι ανοικτά για επέκταση (extension) ή ρύθμιση δηλώνοντας ένα σημείο επέκτασης (extension point). Δηλώνοντας μια επέκταση το bundle δίνει πληροφορία για το τι θα κάνει ένα του δοθεί κάποιο είδος πληροφορίας. Άλλα bundles μπορούν να κάνουν επεκτάσεις δίνοντας την

κατάλληλη πληροφορία. Στο αρχείο plugin.xml που έχει κάθε ένα από τα bundles ορίζονται όλες οι σχέσεις επέκτασης. Όταν γίνει η επίλυση των bundles οι σχέσεις φορτώνονται στο Extension registry για να είναι προσβάσιμες από τα υπόλοιπα bundles.

Υλοποιήσεις OSGi

Υπάρχουν αρκετές υλοποιήσεις του OSGi, οι πιο διαδεδομένες από αυτές είναι:

- Equinox, η πιο διαδομένη υλοποίηση. Σε αυτήν είναι βασισμένο το εργαλείο Eclipse.
- Felix, κάτω από Apache license
- Knopflerfish, κάτω από BSD license

6.1.6 Πλεονεκτήματα του OSGi.

Το OSGi ήδη εξετάζεται ως η επόμενη γενιά της τεχνολογίας Java Middleware. Αυτό δεν μπορεί να αποτελεί έκπληξη, λόγω της υιοθέτησης του OSGi από τους παρόχους JEE Servers αλλά και την Java κοινότητα εν γένει.

Πιο αναλυτικά η OSGi τεχνολογία:

1. Εφαρμόζει καλύτερο componentization από ό, τι οι περισσότερες υφιστάμενες τεχνικές JEE (με ένα classloader ανά bundle).

2. Παρέχει εκδόσεις των στοιχείων και τη δυνατότητα να αναπτυχθούν πολλαπλές εκδόσεις για το ίδιο στοιχείο με τον ίδιο VM, σε αντίθεση με την JEE.

3. Εφαρμόζει SOA για τα στοιχεία που χρησιμοποιούν δυναμικές υπηρεσίες (dynamic services).

4. Εφαρμόζει χαλαρή σύνδεση (loose coupling) των στοιχείων (με τη ενός publish/find/bind μοντέλου υπηρεσία).

5. Επιβάλει την επίλυση των προβλημάτων εξάρτησης εφαρμογών (dependency problems), απαιτώντας τη ρητή δήλωση των εξαρτήσεων μεταξύ των στοιχείων.

6. Ξεπερνάει το πρόβλημα «hot-deployment functionality » επιτρέποντας το deploy/undeploy στοιχείων λογισμικού χωρίς να απαιτείται ο Server / JVM).

7. Υπάρχουν διαθέσιμες πολλές εφαρμογές OSGi ανοικτού κώδικα:

- Equinox/Eclipse
- Felix/Apache
- Knopflerfish/Makewave

8. Οι OSGi υλοποιήσεις είναι αρκετά σταθερές με πολλές εφαρμογές / servers που τις χρησιμοποιούν ή τρέχουν σε αυτές:

- ✓ Η Ricoh παρέχει εκτυπωτές με OSGi από το 2005.
- ✓ Η BMW 5series παρέχεται με OSGi.
- ✓ JBoss, IBM WAS και BEA Weblogic servers υιοθετούν OSGi.
- ✓ Ο Glassfish της Sun χρησιμοποιεί OSGi
- ✓ Το SpringSource DM Server είναι χτισμένο πάνω στο OSGi.
- ✓ Η Siemens Enterprise Communications χρησιμοποιεί το OSGi,και μάλιστα έχει χτίσει επιπλέον framework (Symphonia) πάνω του, για τις ανάγκες της σουίτας της OpenScape Unified Communications.

9. Οι OSGi εφαρμογές έχουν συνήθως ένα μικρό αποτύπωμα (Με OSGi Jars που είναι λιγότερο από 500 kb).

10. Το OSGi έχει αποδείξει την αξία του στον κόσμο των embedded /desktop applications. Το, ανοιχτού κώδικα, Eclipse IDE βασίζεται σε OSGi από το 2004.

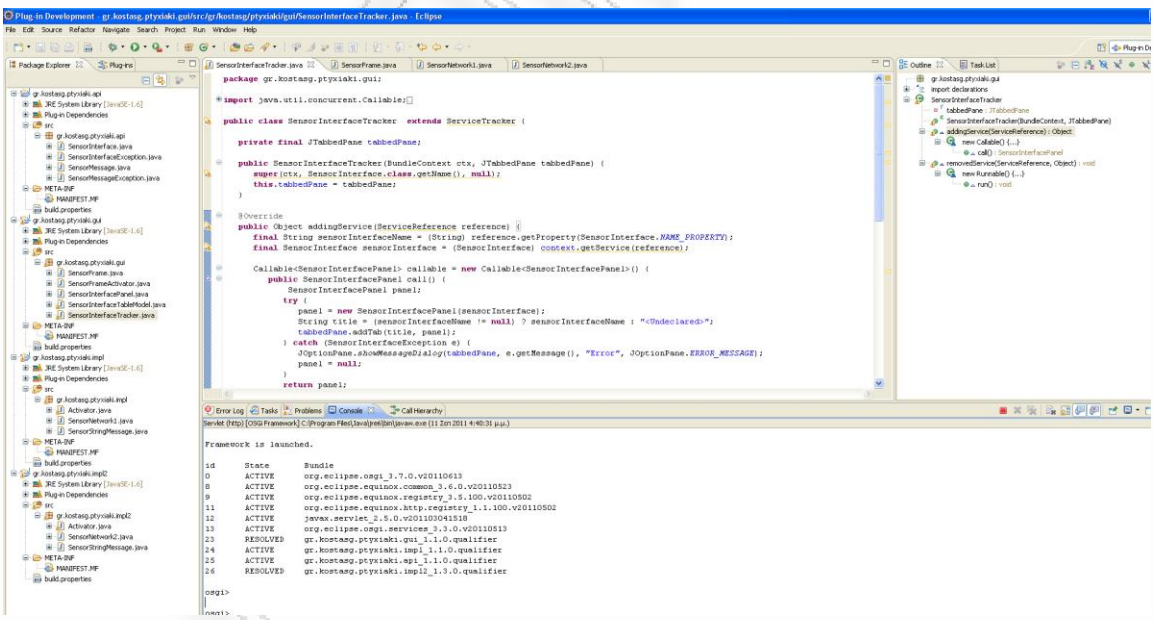
6.2 Υλοποίηση

Η υλοποίηση στα πλαίσια αυτής της εργασίας περιλαμβάνει την κατασκευή ενός ενδιάμεσου λογισμικού το οποίο θα είναι υπεύθυνο να καταγράφει τις διάφορες μετρήσεις αισθητήρων που είναι συνδεδεμένοι στο ασύρματο δίκτυο αισθητήρων και να τις παρουσιάζει στον χρήστη σε ένα απλό και ευέλικτο περιβάλλον.

Επιπλέον, μέσα από αυτήν την υλοποίηση, θα αναδειχτούν όλα τα πλεονεκτήματα της πλατφόρμας OSGi, τα οποία παρουσιάστηκαν σε θεωρητικό επίπεδο στην προηγούμενη ενότητα. Επιπρόσθετα, θα περιγραφούν οι δυνατότητες εξέλιξης αλλά και οι ευκολίες παραμετροποίησης και επέκτασης της εφαρμογής. Ως συνήθως σε κάθε παρόμοια υλοποίησης, γίνονται κάποιες παραδοχές και προσομοιώσεις, οι οποίες δεν επηρεάζουν την γενικότητα.

6.2.1 Εργαλεία ανάπτυξης.

Το κύριο εργαλείο ανάπτυξης που χρησιμοποιήθηκε είναι το Eclipse IDE (Version: Indigo Release, Build id: 20110615-0604), του οποίου η εμφάνιση φαίνεται στην Εικόνα 21.



Εικόνα 21: Το εργαλείο ανάπτυξης Eclipse IDE .

Ο κύριος λόγος επιλογής του Eclipse IDE είναι η άρρηκτη σχέση του με το OSGi, η ποικιλία λειτουργιών και η απλότητα με την οποία τις προσφέρει και φυσικά το ότι αποτελεί το δημοφιλέστερο εργαλείο ανάπτυξης σε Java / OSGi. Αυτό φαίνεται από την πλειάδα plug-ins που υπάρχουν διαθέσιμα στο διαδίκτυο και εξυπηρετούν οποιαδήποτε ανάγκη και τα οποία ουσιαστικά αποτελούν ξεχωριστά bundles τα οποία εγκαθίστανται στον Equinox container του Eclipse.

Ο Equinox είναι η επιλογή του Eclipse για έναν OSGi container ο οποίος θα καλύπτει τις αδιάλειπτες και ποικίλες ανάγκες της προγραμματιστικής κοινότητας.

6.2.2 Γενικός σχεδιασμός και παραδοχές.

Η υλοποίηση στα πλαίσια της εργασίας αυτής, έχει ως βασικό στόχο την κατασκευή μιας βασικής μορφής middleware για την ανάκτηση και παρουσίαση δεδομένων από ένα ή περισσότερα ασύρματα δίκτυα αισθητήρων, αναδεικνύοντας παράλληλα τα οφέλη και τα πλεονεκτήματα της χρήσης της πλατφόρμας OSGi.

Αρχικά είναι αναγκαίες κάποιες παραδοχές, οι οποίες θα διευκολύνουν την υλοποίηση, χωρίς όμως να επηρεάζουν στο ελάχιστο την γενικότητα αυτής.

Για παράδειγμα, το πώς γίνεται η μέτρηση από έναν αισθητήρα και ο τρόπος διασύνδεσής του με το υπολογιστικό σύστημα, είναι θέματα που ξεφεύγουν από το πλαίσιο αυτής της υλοποίησης. Γι’ αυτό και θεωρούμε ότι οι τιμές μέτρησης διάφορων αισθητήρων είναι διαθέσιμες σε ένα αρχείο κειμένου, από όπου διαβάζονται, επεξεργάζονται και παρουσιάζονται στον τελικό χρήστη.

Επιπλέον, χάριν ευκολίας διαχείρισης των δεδομένων, θεωρούμε ότι οι μετρήσεις (ή τα μηνύματα) των αισθητήρων περιέχουν βασικές πληροφορίες, όπως την τιμή μέτρησης και το είδος μέτρησης κ.α.

Για την βέλτιστη διεπαφή με τον τελικό χρήστη, κατασκευάστηκε ένα απλό, λειτουργικό γραφικό περιβάλλον, το οποίο παρουσιάζει σε ευανάγνωστη μορφή όλες τις απαιτούμενες πληροφορίες που στέλνονται από τον ίδιο τον αισθητήρα. Με την εκκίνησή του διαβάζονται όλες οι υπάρχουσες μετρήσεις για κάθε δίκτυο εγγεγραμμένο στο σύστημα, που όπως είπαμε είναι αποθηκευμένες σε συγκεκριμένα αρχεία. Η ανανέωση του γίνεται αυτόματα με την κάθε είσοδο ενός νέου δικτύου αισθητήρων, ή με οποιαδήποτε νέα μέτρηση. Δεδομένης της απουσίας hardware το οποίο θα μας έδινε «ζωντανές» μετρήσεις, προσομοιώνουμε αυτή τη λειτουργία με την τοποθέτηση ενός καινούριου αρχείου

κειμένου σε συγκεκριμένο φάκελο. Το καινούριο αυτό αρχείο θεωρείται ότι περιέχει τις «ζωντανές» μετρήσεις, οι οποίες θα διαβάζονται και θα ανανεώνεται αυτόματα και το περιβάλλον του χρήστη.

Ένα δίκτυο αισθητήρων μπορεί να εισαχθεί στο σύστημα ως ένα bundle οποίο θα υλοποιεί μια συγκεκριμένη διεπαφή του συστήματος και φυσικά όλες τις μεθόδους της. Το πώς όμως θα υλοποιούνται οι μέθοδοι, είναι κάτι που δεν ενδιαφέρει το κεντρικό σύστημα και δεν επηρεάζει τη λειτουργικότητα του middleware, θεωρείται δηλαδή “transparent”.

Αυτό είναι ένα πάρα πολύ σημαντικό πλεονέκτημα, διότι το σύστημα εν δυνάμει μπορεί να υποστηρίξει οποιοδήποτε δίκτυο αισθητήρων, αλλά και μεμονωμένους αισθητήρες, αρκεί να υλοποιηθεί η κεντρική διεπαφή και οι μέθοδοι της.

Όταν εισάγεται ένα δίκτυο αισθητήρων στο σύστημα, πρέπει να δηλώσει την υπηρεσία που προσφέρει στο middleware με τη χρήση του OSGi framework. Κατόπιν ενημερώνεται αυτόματα ο client ότι ένα νέο δίκτυο εισήχθη στο σύστημα, διαβάζει τα δεδομένα του και τα παρουσιάζει στο γραφικό περιβάλλον του χρήστη σε νέα καρτέλα.

Όταν ένα δίκτυο αισθητήρων σταματά να παρέχει την υπηρεσία στο κεντρικό σύστημα, τότε αυτόματα η αντίστοιχη καρτέλα αποσύρεται από το γραφικό περιβάλλον. Αυτό μπορεί να συμβεί για διάφορους λόγους, όπως π.χ. απόσυρση του δικτύου από το σύστημα, απώλεια ενέργειας ή οποιοδήποτε άλλο σφάλμα.

6.2.3 Κλάσεις, διεπαφές και λειτουργικότητα.

Για την υλοποίηση δημιουργήθηκαν τέσσερα διαφορετικά projects στο Eclipse, το κάθε ένα με διαφορετική λειτουργία και σκοπό. Μέσα στο αρχείο MANIFEST.MF του κάθε project, δηλώθηκαν όλες οι απαραίτητες εξαρτήσεις μεταξύ των bundles, ώστε να είναι δυνατή η εκτέλεσή του. Ειδικότερα το «gr.kostasg.ptyxiaki.api» το δηλώνουμε ως «Export-Package», ενώ σε όλα τα άλλα projects το δηλώνουμε ως «Import-Package».

- **gr.kostasg.ptyxiaki.api:**

Το πρώτο και βασικό project (gr.kostasg.ptyxiaki.api) περιέχει όλα τα απαραίτητα interfaces που πρέπει να υλοποιηθούν από τις κλάσεις.

Έτσι έχουμε το SensorMessage, που ουσιαστικά περιγράφει τι πεδία και τι πληροφορίες πρέπει να περιέχουν τα μηνύματα που έρχονται από τους αισθητήρες, με τις αντίστοιχες μεθόδους για την ανάκτησή τους. Έχει γίνει η παραδοχή ότι ένα τέτοιο μήνυμα θα περιέχει ένα μοναδικό id του μηνύματος, την τιμή μέτρησης, τον τύπο της μέτρησης, το πότε έγινε η μέτρηση και πληροφορίες σχετικά με τον αισθητήρα όπως μάρκα, μοντέλο κλπ.

```
package gr.kostasg.ptyxiaki.api;

/**
 *
 * @author Konstantinos Giantselidis
 */

public interface SensorMessage {
    /**
     * @return The unique sensor ID.
     */
    long getSensorMessageId();

    /**
     * @return The value the sensor measured.
     */
    long getSensorValue();

    /**
     * @return The type of what the sensor measured.
     */
    String getSensorType();

    /**
     * @return The date-time that the sensor measurement took place.
     */
    String getSensorTimestamp();

    /**
     *  @return The info of the sensor (usually its model type).
     *
     */
    String getSensorInfo();

}
```

“SensorMessage.java”

Στη συνέχεια ορίζουμε το SensorInterface, το οποίο αποτελεί ένα είδος κεντρικής διαχείρισης των παραπάνω μηνυμάτων, περιγράφοντας πέντε μεθόδους (ένα για κάθε πεδίο του μηνύματος) που παρέχουν τα δεδομένα των μηνυμάτων στις κλάσεις που θα τις υλοποιήσουν.

Τέλος περιγράφουμε και την κλάση SensorInterfaceException, για την πιο εύκολη και κατανοητή παρουσίαση των Exceptions που τυχόν παρουσιαστούν.

```
package gr.kostasg.ptyxiaki.api;
/**
*
* @author Konstantinos Giantselidis
*/
public interface SensorInterface {

   public static final String NAME_PROPERTY = "sensorNetworkName";

   /**
    * Retrieve all messages available.
    *
    * @return An array of message IDs.
    * @throws SensorInterfaceException
    */
   long[] getAllSensorMessages() throws SensorInterfaceException;

   /**
    * Retrieve all messages received after the specified message.
    *
    * @param id The message ID.
    * @return An array of message IDs.
    * @throws SensorInterfaceException
    */
   long[] getSensorMessagesSince(long id) throws
SensorInterfaceException;

   /**
    * Retrieve the specified messages.
    *
    * @param ids The IDs of the messages to be retrieved.
    * @return An array of Messages.
    * @throws SensorInterfaceException
    */
   SensorMessage[] getSensorMessages(long[] ids) throws
SensorInterfaceException;

}
```

“SensorInterface.java”

```java
package gr.kostasg.ptyxiaki.api;

/**
*
* @author Konstantinos Giantselidis
*/

public class SensorInterfaceException extends Exception {

   private static final long serialVersionUID = 1L;

   public SensorInterfaceException(String message) {
      super(message);
   }

   public SensorInterfaceException(Throwable cause) {
      super(cause);
   }

   public SensorInterfaceException(String message, Throwable cause) {
      super(message, cause);
   }

}
```

”SensorInterfaceException.java”

- **gr.kostasg.ptyxiaki.impl:**

Σε αυτό το project έχουμε μια υλοποίηση των interfaces που ορίστηκαν παραπάνω. Καταρχήν, στην SensorStringMessage υλοποιείται η διεπαφή SensorMessage και όλες τις οι μέθοδοι, ενώ είναι εμφανές πως όλες οι τιμές των πεδίων του μηνύματος μεταφέρονται μέσω του constructor της κλάσης.

```java
package gr.kostasg.ptyxiaki.impl;

/**
*
* @author Konstantinos Giantselidis
*/

import gr.kostasg.ptyxiaki.api.SensorMessage;

public class SensorStringMessage implements SensorMessage {

   private final long id;
   private final long value;
   private final String type;
   private final String timestamp;
   private final String info;

   public SensorStringMessage(long id, long value, String type,
                              String timestamp, String info) {
      this.id = id;
      this.value = value;
      this.timestamp = timestamp;
      this.type = type;
      this.info = info;
   }

   public String getSensorInfo() {
      return info;
   }

   public long getSensorMessageId() {
      return id;
   }

   public String getSensorType() {
      return type;
   }

   public long getSensorValue() {
      return value;
   }

   public String getSensorTimestamp() {
      return timestamp;
   }
}
```

“SensorStringMessage.java”

Στην συνέχεια έχουμε την κλάση Activator, η οποία ουσιαστικά υλοποιεί την διεπαφή BundleActivator του OSGi, και είναι υπεύθυνη για την εκκίνηση αλλά και τον τερματισμό του ίδιου του project ως bundle.

Μια άλλη βασική λειτουργία που εκτελείται είναι η εγγραφή της υπηρεσίας που προσφέρει το bundle, δηλαδή η εισαγωγή του δικτύου αισθητήρων στο OSGi framework.

Αυτό γίνεται με κλήση της registerService(), πάνω στο context που έχει επιστραφεί από το ίδιο το OSGi framework, όταν ξεκίνησε το bundle. Ιδιαίτερη αναφορά πρέπει να γίνει στην μεταβλητή props, που περνάει σαν όρισμα μέσα στην registerService(), αφού μας επιτρέπει να ορίσουμε διάφορες πληροφορίες για την υπηρεσία που προσπαθούμε να εγγράψουμε. Έτσι μπορούμε να δηλώσουμε ότι η υπηρεσία τύπου SensorNetwork1 θα έχει την ονομασία Kostas_Network. Με αυτόν τον τρόπο μπορούμε και ξεχωρίζουμε τα εγγεγραμμένα δίκτυα στο σύστημα και έχουμε τη δυνατότητα να τα διαχειριστούμε με διαφορετική προσέγγιση.

Η κλήση της μεθόδου SensorNetwork1, που υλοποιεί το interface SensorInterface, ουσιαστικά προετοιμάζει τα δεδομένα των μηνυμάτων για χρήση από τον client. Για το συγκεκριμένο project επιλέχθηκε τα δεδομένα να υπάρχουν ήδη στη μνήμη, χάριν ευκολίας προσομοίωσης των υπολοίπων λειτουργιών. Για το λόγο αυτό δημιουργείται ένα ArrayList όπου αποθηκεύονται οι πληροφορίες πέντε μηνυμάτων.

Επίσης υλοποιούνται οι μέθοδοι που περιγράφηκαν στην SensorInterface, για την ανάκτηση όλων των μηνυμάτων (getAllSensorMessages), ή συγκεκριμένων μηνυμάτων (getSensorMessages), ή μηνύματα μετά από κάποιο συγκεκριμένο ID μηνύματος (getSensorMessagesSince).

```
package gr.kostasg.ptyxiaki.impl;

/**
*
* @author Konstantinos Giantselidis
*/

import java.util.Dictionary;
import java.util.Hashtable;

import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceRegistration;

import gr.kostasg.ptyxiaki.api.SensorInterface;

public class Activator implements BundleActivator {
      private ServiceRegistration sr;
      private static BundleContext context;

      static BundleContext getContext() {
            return context;
      }

      public void start(BundleContext bundleContext) throws Exception {
          this.context = bundleContext;
          SensorInterface sensorInterface = new SensorNetwork1();
          Dictionary props = new Hashtable();
          props.put(SensorInterface.NAME_PROPERTY, "Kostas_Network");

          sr = context.registerService(SensorInterface.class.getName(),
                                       sensorInterface, props);

          System.out.println("Sensor Network <<Kostas_Network>>
                              has come online.");
      }

      public void stop(BundleContext bundleContext) throws Exception {
            System.out.println("Sensor Network <<Kostas_Network>>
                                has gone offline.");
            sr.unregister();
      }

}
```

”Activator.java”

```
package gr.kostasg.ptyxiaki.impl;

/**
*
* @author Konstantinos Giantselidis
*/

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import gr.kostasg.ptyxiaki.api.SensorInterface;
import gr.kostasg.ptyxiaki.api.SensorInterfaceException;
import gr.kostasg.ptyxiaki.api.SensorMessage;

public class SensorNetwork1 implements SensorInterface {

        protected final List<SensorMessage> messages;

        public SensorNetwork1() {
           messages = new ArrayList<SensorMessage>(5);
           messages.add(new SensorStringMessage(0, 25, "Temperature",
now("dd.MM.yyyy HH:MM:SS"), "Motorola HMTS1050"));
           messages.add(new SensorStringMessage(1, 70, "Humidity",
now("dd.MM.yyyy HH:MM:SS"),"Crossbow MDA300CA"));
           messages.add(new SensorStringMessage(2, 30, "WindSpeed",
now("dd.MM.yyyy HH:MM:SS"), "Climatronics F460"));
           messages.add(new SensorStringMessage(3, 23, "Temperature",
now("dd.MM.yyyy HH:MM:SS"), "Motorola HMTS1050"));
           messages.add(new SensorStringMessage(4, 74, "Humidity",
now("dd.MM.yyyy HH:MM:SS"),"Crossbow MDA300CA"));
        }

        public synchronized long[] getAllSensorMessages() {
           long[] ids = new long[messages.size()];
           for (int i = 0; i < ids.length; i++) {
              ids[i] = i;
           }
           return ids;
        }

        public synchronized SensorMessage[] getSensorMessages(long[]
ids) throws SensorInterfaceException {
           SensorMessage[] result = new SensorMessage[ids.length];
           for (int i = 0; i < ids.length; i++) {
              long id = ids[i];
              if (id < 0 || id >= messages.size()) {
                 throw new SensorInterfaceException("Invalid message
ID: " + id);
              }
              result[i] = messages.get((int) id);
           }
           return result;
        }
```

```
        public synchronized long[] getSensorMessagesSince(long id)
              throws SensorInterfaceException {
           int first = (int) (id + 1);
           if (first < 0) {
              throw new SensorInterfaceException("Invalid message ID:
" + first);
           }
           int length = Math.max(0, messages.size() - first);
           long[] ids = new long[length];
           for (int i = 0; i < length; i++) {
              ids[i] = i + first;
           }
           return ids;
        }

        public static String now(String dateFormat) {
              Calendar cal = Calendar.getInstance();
              SimpleDateFormat sdf = new
SimpleDateFormat(dateFormat);
              return sdf.format(cal.getTime());
        }
}
```

"SensorNetwork1.java"

- **gr.kostasg.ptyxiaki.impl2:**

Αυτό το project αποτελεί μια παραλλαγή του προηγούμενο, με τη διαφορά ότι τα δεδομένα των μηνυμάτων τώρα διαβάζονται από συγκεκριμένο αρχείο, ενώ αλλάζει και το όνομα του δικτύου που εισάγεται στο σύστημα (Dimos_Network).

Έτσι και εδώ έχουμε την κλάση SensorStringMessage που υλοποιεί την διεπαφή SensorMessage και όλες τις μεθόδους της.

```java
package gr.kostasg.ptyxiaki.impl2;

/**
*
* @author Konstantinos Giantselidis
*/

import gr.kostasg.ptyxiaki.api.SensorMessage;

public class SensorStringMessage implements SensorMessage {

   private final long id;
   private final long value;
   private final String type;
   private final String timestamp;
   private final String info;


   public SensorStringMessage(long id, long value, String type,
                              String timestamp, String info) {
      this.id = id;
      this.value = value;
      this.timestamp = timestamp;
      this.type = type;
      this.info = info;
   }

   public String getSensorInfo() {
      return info;
   }

   public long getSensorMessageId() {
      return id;
   }

   public String getSensorType() {
      return type;
   }

   public long getSensorValue() {
      return value;
   }

   public String getSensorTimestamp() {
      return timestamp;
   }
}
```

“SensorInterface.java”

Τα άλλα δυο αρχεία όμως θα αλλάξουν σημαντικά. Καταρχήν ο Activator αλλάζει μορφή, μιας και τώρα η εγγραφή της υπηρεσίας θα γίνει όταν θα βρεθεί το αρχείο που θα περιέχει τα απαραίτητα δεδομένα από τα μηνύματα των αισθητήρων.

Η κλάση έχει φτιαχτεί έτσι ώστε να κάνει συνεχόμενο polling, μέχρι να βρεθεί το αρχείο. Αυτός είναι και ένας τρόπος προσομοίωσης των live μετρήσεων, αφού μπορούμε να εισάγουμε το αρχείο όταν το επιθυμούμε, για να δούμε τον client αυτόματα να παρουσιάζει τα νέα δεδομένα σε ξεχωριστή καρτέλα. Η χρήση κάποιων ιδιαίτερων κλάσεων (Runnable, Future, Callable) είναι αναγκαία, διότι με την εκκίνηση του bundle ξεκινάμε ένα thread, από το οποίο αργότερα θέλουμε να ενημερωθούμε εάν βρέθηκε το αρχείο. Και λόγω του τρόπου λειτουργίας των threads, οι παραπάνω μέθοδοι μας βοηθάνε να πάρουμε την πληροφορία που θέλουμε από το σωστό thread.

```
package gr.kostasg.ptyxiaki.impl2;

/**
*
* @author Konstantinos Giantselidis
*/

import java.util.Dictionary;
import java.util.Hashtable;
import java.io.File;

import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceRegistration;

import gr.kostasg.ptyxiaki.api.SensorInterface;

public class Activator implements BundleActivator {
      private Thread thread;

      public void start(BundleContext bundleContext) throws Exception {
            Dictionary props = new Hashtable();
            File file = new File("messages.txt");

            props.put(SensorInterface.NAME_PROPERTY, "Dimos_Network");
            System.out.println("Sensor Network <<Dimos_Network>> has
come online.");
            System.out.println("Will search for file " +
file.getAbsolutePath());
            RegistrationRunnable runnable = new
      RegistrationRunnable(bundleContext, file, props);
```

```
            thread = new Thread(runnable);
            thread.start();
        }

        public void stop(BundleContext bundleContext) throws Exception {
            System.out.println("Sensor Network <<Dimos_Network>> has
gone offline.");
            thread.interrupt();
        }

}

class RegistrationRunnable implements Runnable {

        private final BundleContext context;
        private final File file;
        private final Dictionary props;

        public RegistrationRunnable(BundleContext context, File file,
Dictionary props) {
            this.context = context;
            this.file = file;
            this.props = props;
        }

        public void run() {
            ServiceRegistration registration = null;

            try {
                while (!Thread.currentThread().isInterrupted()) {
                    if (file.exists()) {
                        if (registration == null) {
                            registration = context.registerService(
                                    SensorInterface.class.getName(),
                                    new SensorNetwork2(file), props);
                            break;  // Remove this for continuous polling.
                        }
                    } else {
                        if (registration != null) {
                            registration.unregister();
                            registration = null;
                        }
                    }
                    Thread.sleep(10000);
                }
            } catch (InterruptedException e) {
                System.out.println("Thread exited... Exception=" + e);
            }
        }
}
```

“Activator.java”

Η κλήση της μεθόδου SensorNetwork2, που υλοποιεί το interface SensorInterface, ουσιαστικά προετοιμάζει τα δεδομένα των μηνυμάτων για χρήση από τον client, αυτή τη φορά διαβάζοντάς τα από ένα αρχείο. Και πάλι δημιουργείται ένα ArrayList όπου αποθηκεύονται οι πληροφορίες πέντε μηνυμάτων που διαβάζονται από το αρχείο messages.txt (το οποίο ΠΡΕΠΕΙ να βρίσκεται στο φάκελο C:\eclipse).

Φυσικά υλοποιούνται και πάλι οι μέθοδοι που περιγράφηκαν στην SensorInterface, για την ανάκτηση όλων των μηνυμάτων (getAllSensorMessages), ή συγκεκριμένων μηνυμάτων (getSensorMessages), ή για μηνύματα μετά από κάποιο συγκεκριμένο ID μηνύματος (getSensorMessagesSince).

```
package gr.kostasg.ptyxiaki.impl2;

/**
*
* @author Konstantinos Giantselidis
*/

import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

import gr.kostasg.ptyxiaki.api.SensorInterface;
import gr.kostasg.ptyxiaki.api.SensorInterfaceException;
import gr.kostasg.ptyxiaki.api.SensorMessage;


public class SensorNetwork2 implements SensorInterface {

          protected final List<SensorMessage> messages;

          public SensorNetwork2(File file) {
               messages = new ArrayList<SensorMessage>(5);

               getTextFile(file);
          }

          public synchronized long[] getAllSensorMessages() {
             long[] ids = new long[messages.size()];
             for (int i = 0; i < ids.length; i++) {
                ids[i] = i;
             }
             return ids;
          }

          public synchronized SensorMessage[] getSensorMessages(long[]
ids)
                throws SensorInterfaceException {
             SensorMessage[] result = new SensorMessage[ids.length];
             for (int i = 0; i < ids.length; i++) {
                long id = ids[i];
                if (id < 0 || id >= messages.size()) {
                   throw new SensorInterfaceException("Invalid message
ID: " + id);
                }
                result[i] = messages.get((int) id);
             }
             return result;
          }
```

```
        public synchronized long[] getSensorMessagesSince(long id)
               throws SensorInterfaceException {
            int first = (int) (id + 1);
            if (first < 0) {
               throw new SensorInterfaceException("Invalid message
                                                   ID: " + first);
            }
            int length = Math.max(0, messages.size() - first);
            long[] ids = new long[length];
            for (int i = 0; i < length; i++) {
               ids[i] = i + first;
            }
            return ids;
         }

         private void getTextFile(File file) {
               try
               {
                 FileReader fr = new FileReader(file);
                 BufferedReader br = new BufferedReader( fr );
                 // declare String variable and prime the read
                 String stringRead = br.readLine( );
                 while( stringRead != null ) // end of the file?
                 {
                   // process the line read
                   StringTokenizer st = new StringTokenizer(
stringRead, "#" );
                   String sensId = st.nextToken( );
                   String sensValue = st.nextToken( );
                   String sensType = st.nextToken( );
                   String sensTimestamp = st.nextToken( );
                   String sensInfo = st.nextToken( );
                   messages.add(new SensorStringMessage(new
Long(sensId), new Long(sensValue), sensType, sensTimestamp, sensInfo));

                   stringRead = br.readLine( );
                 }
                 // release associated resources
                 br.close( );
               }

               catch( FileNotFoundException fnfe )
               {
                     System.out.println("Cannot find file " +
file.getAbsolutePath());
               }
               catch( IOException ioe )
               {
                     System.out.println("IOException occured.  ");
                     ioe.printStackTrace();
               }
            }
      }
```

”SensorNetwork2.java”

- **gr.kostasg.ptyxiaki.gui:**

Όπως προδίδει και η ονομασία του (GUI), αυτό το project δημιουργήθηκε για την κατασκευή ενός απλού και φιλικού προς τον χρήστη γραφικού περιβάλλοντος, το οποίο θα παρουσιάζει σωστά τις πληροφορίες που αντλήθηκαν από τα μηνύματα των αισθητήρων.

Η περιγραφή και επεξήγηση του κώδικα που έχει να κάνει με την δημιουργία του γραφικού περιβάλλοντος δεν είναι της παρούσης εργασίας, γι αυτό θα επικεντρωθούμε στην λειτουργία κάθε αυτή.

Αρχικά, όπως και σε κάθε bundle το οποίο θέλουμε να τρέξουμε, υπάρχει μια κλάση που υλοποιεί την διεπαφή του συστήματος BundleActivator. Σε αυτό το project είναι η SensorFrameActivator. Βέβαια τώρα στην μέθοδο start(), έχουμε όλες τις απαραίτητες εντολές για την εκκίνηση του γραφικού περιβάλλοντος του client.

Επίσης έχει προστεθεί και ένας listener, ο οποίος θα τερματίζει το bundle όταν ο χρήστης κλείσει το παράθυρο του client. Αν και ακόμη δεν έχει καλεστεί κάποια μέθοδος για την ανεύρεση υπηρεσιών, η διαδικασία αυτή ουσιαστικά ξεκινάει με το κάλεσμα της μεθόδου openTracking().

Στη συνέχεια έχουμε την SensorFrame, η οποία δημιουργεί το κεντρικό γραφικό περιβάλλον και την πρώτη καρτέλα με το μήνυμα υποδοχής. Επίσης υλοποιείται η μέθοδος openTracking(), που αναφέρθηκε παραπάνω, η οποία δημιουργεί ένα αντικείμενο SensorInterfaceTracker (θα περιγραφεί στη συνέχεια) καλώντας τον αντίστοιχο constructor.

```
package gr.kostasg.ptyxiaki.gui;

/**
*
* @author Konstantinos Giantselidis
*/

import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.UIManager;

import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.framework.BundleException;

public class SensorFrameActivator implements BundleActivator {

   private SensorFrame frame;

   public void start(final BundleContext context) throws Exception {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
      frame = new SensorFrame();
      frame.pack();

      frame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent e) {
            try {
               context.getBundle().stop();
            } catch (BundleException e1) {
               // Ignore
            }
         }
      });

      frame.openTracking(context);

      frame.setVisible(true);
   }

   public void stop(BundleContext context) throws Exception {
      frame.setVisible(false);

      frame.closeTracking();
   }

}
```

“SensorFrameActivator.java”

```
package gr.kostasg.ptyxiaki.gui;
/**
* @author Konstantinos Giantselidis
*/
import java.awt.BorderLayout; import java.awt.Color;
import java.awt.Component;    import java.awt.Dimension;
import java.awt.Font;         import javax.swing.SwingConstants;
import javax.swing.JFrame;    import javax.swing.JLabel;
import javax.swing.JPanel;    import javax.swing.JTabbedPane;
import org.osgi.framework.BundleContext;

public class SensorFrame extends JFrame {
   private static final long serialVersionUID = 1L;
   private JTabbedPane tabbedPane;
   private SensorInterfaceTracker tracker;

   public SensorFrame() {
      super("Wireless Sensor Networks");
      tabbedPane = new JTabbedPane();
      tabbedPane.addTab("Sensor Networks", createIntroPanel());
      tabbedPane.setPreferredSize(new Dimension(800, 600));
      getContentPane().add(tabbedPane, BorderLayout.CENTER);
   }

   private Component createIntroPanel() {
      JPanel panel = new JPanel();
      JLabel label1 = new JLabel("Select the tab with the preferred
Network");
      label1.setHorizontalAlignment(SwingConstants.CENTER);
      label1.setVerticalTextPosition(SwingConstants.BOTTOM);
      label1.setFont(new Font("Arial", Font.BOLD, 24));
      label1.setForeground(Color.RED);
      JLabel label2 = new JLabel("WSNs");
      label2.setHorizontalAlignment(SwingConstants.CENTER);
      label2.setVerticalTextPosition(SwingConstants.TOP);
      label2.setFont(new Font("Arial", Font.BOLD, 226));
      label2.setForeground(Color.BLUE);
      JLabel label0 = new JLabel("                ");
      label0.setFont(new Font("Arial", Font.BOLD, 124));
      panel.add(label2); panel.add(label0); panel.add(label1);
      return panel;
   }

   protected void openTracking(BundleContext context) {
      tracker = new SensorInterfaceTracker(context, tabbedPane);
      tracker.open();
   }
   protected void closeTracking() {
      tracker.close();
   }
}
```

“SensorFrame.java”

Η βασική λειτουργία του client μπορούμε να πούμε ότι γίνεται στην κλάση SensorInterfaceTracker, η οποία επεκτείνει την κλάση ServiceTracker του OSGi framework. Αυτό της δίνει τη δυνατότητα να καλέσει τις μεθόδους που θα επιτρέψουν την αναζήτηση και ανεύρεση των υπηρεσιών που έχουν ήδη δηλωθεί στο OSGi framework. Επιπλέον δίνει τη δυνατότητα να γίνει "track" μια συγκεκριμένη υπηρεσία, ώστε να ενημερωθεί η εφαρμογή άμεσα από το framework όταν η συγκεκριμένη υπηρεσία εγγραφεί στο σύστημα.

Έτσι με την είσοδο της υπηρεσίας στο σύστημα, ο client αμέσως δημιουργεί μια νέα καρτέλα και την προετοιμάζει ώστε να παρουσιάσει τα δεδομένα που αντιστοιχούν σε αυτήν την υπηρεσία (δηλαδή το δίκτυο αισθητήρων).

```
package gr.kostasg.ptyxiaki.gui;

/**
*
* @author Konstantinos Giantselidis
*/

import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.Future;
import java.util.concurrent.FutureTask;
import javax.swing.JOptionPane;
import javax.swing.JTabbedPane;
import javax.swing.SwingUtilities;

import gr.kostasg.ptyxiaki.api.SensorInterface;
import gr.kostasg.ptyxiaki.api.SensorInterfaceException;
import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceReference;
import org.osgi.util.tracker.ServiceTracker;

public class SensorInterfaceTracker  extends ServiceTracker {

   private final JTabbedPane tabbedPane;

   public SensorInterfaceTracker(BundleContext ctx, JTabbedPane
tabbedPane) {
      super(ctx, SensorInterface.class.getName(), null);
      this.tabbedPane = tabbedPane;
   }

   @Override
   public Object addingService(ServiceReference reference) {
      final String sensorInterfaceName = (String)
reference.getProperty(SensorInterface.NAME_PROPERTY);
      final SensorInterface sensorInterface = (SensorInterface)
context.getService(reference);
```

```
        Callable<SensorInterfacePanel> callable = new
                        Callable<SensorInterfacePanel>() {
           public SensorInterfacePanel call() {
               SensorInterfacePanel panel;
              try {
                 panel = new SensorInterfacePanel(sensorInterface);
                 String title = (sensorInterfaceName != null) ?
sensorInterfaceName : "<Undeclared>";
                 tabbedPane.addTab(title, panel);
              } catch (SensorInterfaceException e) {
                 JOptionPane.showMessageDialog(tabbedPane,
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                 panel = null;
              }
              return panel;
           }
        };

        FutureTask<SensorInterfacePanel> future = new
FutureTask<SensorInterfacePanel>(callable);
        SwingUtilities.invokeLater(future);

        return future;
     }

     @Override
     public void removedService(ServiceReference reference, Object svc) {

        @SuppressWarnings("unchecked")
        final Future<SensorInterfacePanel> panelRef =
(Future<SensorInterfacePanel>) svc;

        SwingUtilities.invokeLater(new Runnable() {
           public void run() {
              try {
                    SensorInterfacePanel panel = panelRef.get();
                 if (panel != null) {
                    tabbedPane.remove(panel);
                 }
              } catch (ExecutionException e) {
                 // The SensorInterfacePanel was not successfully created
              } catch (InterruptedException e) {
                 // Restore interruption status
                 Thread.currentThread().interrupt();
              }
           }
        });

        context.ungetService(reference);
     }
}
```

“SensorInterfaceTracker.java”

```
package gr.kostasg.ptyxiaki.gui;

/**
*
* @author Konstantinos Giantselidis
*/

import java.awt.Dimension;

import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;

import gr.kostasg.ptyxiaki.api.SensorInterface;
import gr.kostasg.ptyxiaki.api.SensorInterfaceException;

public class SensorInterfacePanel  extends JPanel {
   private static final long serialVersionUID = 1L;

   private final SensorInterfaceTableModel tableModel;

   public SensorInterfacePanel(SensorInterface sensorInterface) throws
SensorInterfaceException {
      tableModel = new SensorInterfaceTableModel(sensorInterface);
      JTable table = new JTable(tableModel);
      JScrollPane scrollPane = new JScrollPane(table);
      scrollPane.setPreferredSize(new Dimension(800, 600));

      add(scrollPane);
   }
}
```

“SensorInterfacePanel.java”

```
package gr.kostasg.ptyxiaki.gui;
/**
*
* @author Konstantinos Giantselidis
*/
import java.util.ArrayList;
import java.util.List;
import javax.swing.table.AbstractTableModel;

import gr.kostasg.ptyxiaki.api.SensorInterface;
import gr.kostasg.ptyxiaki.api.SensorInterfaceException;
import gr.kostasg.ptyxiaki.api.SensorMessage;

public class SensorInterfaceTableModel extends AbstractTableModel {
   private static final String ERROR = "ERROR";
   private final SensorInterface sensorInterface;
   private final List<SensorMessage> sensorMessages;
```

```
   public SensorInterfaceTableModel(SensorInterface sensorInterface)
throws SensorInterfaceException {
      this.sensorInterface = sensorInterface;
      long[] messageIds = sensorInterface.getAllSensorMessages();
      sensorMessages = new ArrayList<SensorMessage>(messageIds.length);
      SensorMessage[] messageArray =
sensorInterface.getSensorMessages(messageIds);
      for (SensorMessage message : messageArray) {
        sensorMessages.add(message);
      }
   }

   public synchronized int getRowCount() {
      return sensorMessages.size();
   }

   public int getColumnCount() {
      return 5;
   }

   @Override
   public String getColumnName(int column) {
      switch (column) {
      case 0:          return "ID";
      case 1:          return "Value";
      case 2:          return "Type";
      case 3:          return "Timestamp";
      case 4:          return "Info";
      }
      return ERROR;
   }

   public synchronized Object getValueAt(int row, int column) {
      SensorMessage sensorMessage = sensorMessages.get(row);
      switch (column) {
      case 0:
         return Long.toString(sensorMessage.getSensorMessageId());
      case 1:
         return sensorMessage.getSensorValue();
      case 2:
         return sensorMessage.getSensorType();
      case 3:
         return sensorMessage.getSensorTimestamp();
      case 4:
         return sensorMessage.getSensorInfo();
      }
      return ERROR;
   }
}
```

“SensorInterfaceTableModel.java”

Επίσης έχουμε την κλάση SensorInterfacePanel, που περιέχει εντολές για τη δημιουργία του γραφικού περιβάλλοντος της καρτέλας που θα υποδεχθεί τα δεδομένα.

Τέλος, στην κλάση SensorInterfaceTableModel περιέχεται η λειτουργία ανάγνωσης των δεδομένων, καλώντας την getSensorMessages(), μέθοδος την οποία απαιτείται να υλοποιεί κάθε κλάση που υλοποιεί την διεπαφή SensorInterface. Επίσης, δεδομένου ότι η κλάση αυτή επεκτείνει την κλάση AbstractTableModel, προστέθηκαν δυο μέθοδοι που είναι υπεύθυνες για την ονομασία των στηλών και την τοποθέτηση των σωστών τιμών σε αυτές.

- **Χρήση της εφαρμογής.**

Η εφαρμογή μπορεί να εκτελεστεί σχετικά εύκολα. Το μόνο που απαιτείται είναι το Eclipse IDE, μιας και τα projects φτιάχτηκαν σε αυτό και είναι πολύ εύκολο να εισαχθούν πάλι.

Αυτό δε σημαίνει ότι η εφαρμογή δεν μπορεί να σε τρέξει με άλλο τρόπο. Ο κώδικας δεν στηρίζεται αποκλειστικά στο Eclipse. Έχει υλοποιηθεί πάνω σε ένα γενικό OSGi framework και μπορεί να μεταφερθεί και να εκτελεστεί σε οποιοδήποτε container, αρκεί να τηρηθούν κάποιοι κανόνες σχετικά με τις ρυθμίσεις των εξαρτήσεων μεταξύ των bundles.

Εάν λοιπόν τα τέσσερα αυτά projects εισαχθούν σε ένα Eclipse IDE, τότε αρχικά επιλέγουμε “Project -> Build All” για να σιγουρευτούμε ότι όλα τα projects έγιναν build. Στη συνέχεια επιλέγουμε το project “gr.kostasg.ptyxiaki.api”, με δεξί click, “Run As -> Run Configurations” (Εικόνα 25). Στο παράθυρο που ανοίγει έχουμε τη δυνατότητα ν ορίσουμε εμείς τα απαραίτητα bundles που θέλουμε να τρέξουν με την εφαρμογή μας. Καλό είναι βέβαια να το αφήσουμε ως έχει.

Με το που πατήσουμε “Run” θα δούμε στο κάτω μέρος του Eclipse, την κονσόλα του OSGi, όπου και εκτυπώνονται όλα τα μηνύματα του συστήματος. Από εκεί μπορούμε ακόμη να διαχειριστούμε όλα τα bundles του συστήματος, να δούμε ποια είναι εγκατεστημένα μόνο, ποια είναι και ενεργοποιημένα, να ξεκινήσουμε ή να σταματήσουμε ένα bundle, και πολλές άλλες λειτουργίες.

Λογικά ήδη θα έχει εμφανιστεί το γραφικό παράθυρο με την καρτέλα εισόδου (Εικόνα 22), και εάν τα δυο bundles υλοποίησης λειτουργούν σωστά, θα πρέπει να βλέπουμε και δυο ακόμη καρτέλες, μια για κάθε δίκτυο αισθητήρων που εισήλθε στο σύστημα (Εικόνα 23).

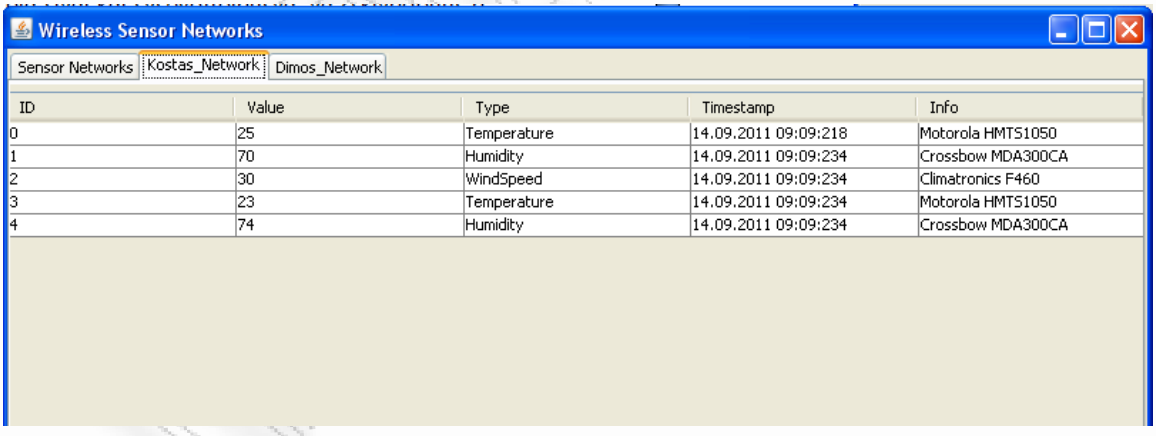
Εάν πάμε στην κονσόλα και εκτελέσουμε την εντολή ”ss” θα δούμε την λίστα με όλα τα bundles στο framework, το κάθε ένα με το δικό μοναδικό αριθμό. Εάν π.χ. το bundle “gr.kostasg.ptyxiaki.impl2” έχει αντιστοιχηθεί στον αριθμό 22, τότε μπορούμε να σταματήσουμε τη λειτουργία του δίνοντας στην κονσόλα την εντολή “stop 22”. Τότε το bundle θα σταματήσει κάθε λειτουργία και θα διακόψει την υπηρεσία που προσφέρει, κάτι που θα το καταλάβει ο client και θα αφαιρέσει την αντίστοιχη καρτέλα του δικτύου αισθητήρων.

Κάπως έτσι προσομοιώνεται η αφαίρεση ενός δικτύου από το σύστημα. Ομοίως βέβαια, με την εντολή “start 22” το bundle εκκινεί και πάλι ενώ και αντίστοιχη καρτέλα επανεμφανίζεται στον client (Εικόνα 24).

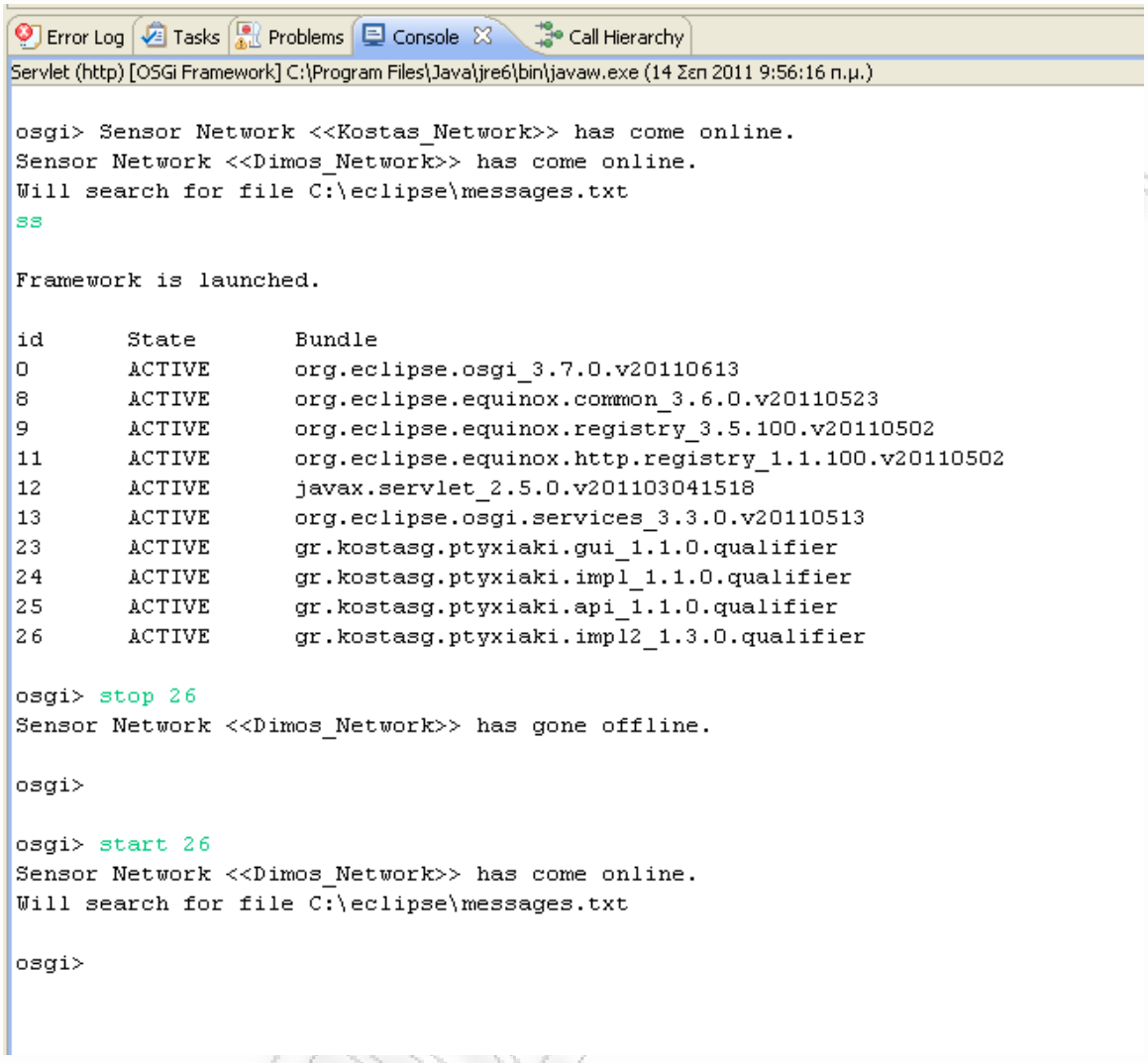
Να σημειώσουμε και πάλι ότι για να λειτουργήσει σωστά το bundle “gr.kostasg.ptyxiaki.impl2” πρέπει να έχουμε τοποθετήσει στον φάκελο C:\eclipse το αρχείο “messages.txt” (Εικόνα 26).

Στη συνέχεια παραθέτονται μερικές εικόνες από την χρήση της εφαρμογής αλλά και όσων περιγράφηκαν παραπάνω.

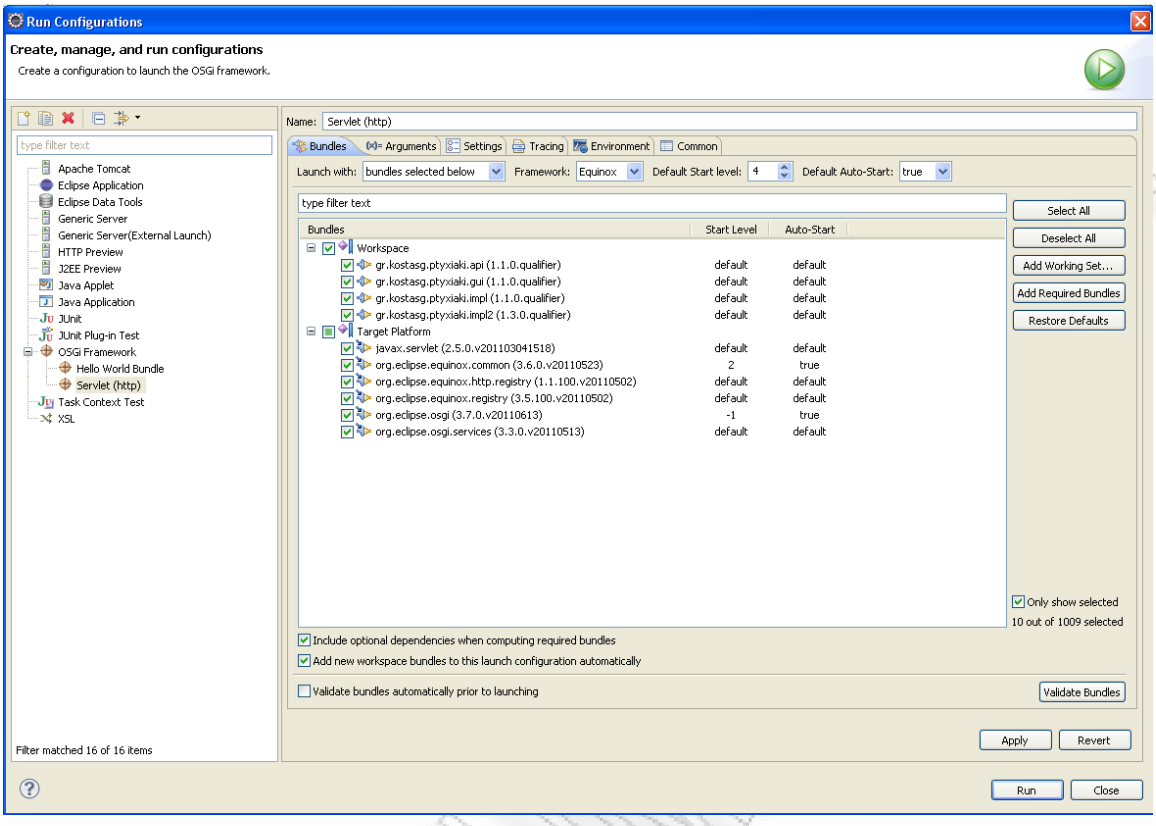
Εικόνα 22: Κεντρική καρτέλα του client.



Εικόνα 23: Καρτέλα με τα δεδομένα ενός δικτύου αισθητήρων.



Εικόνα 24: Εντολές στην κονσόλα OSGi του Eclipse.



Εικόνα 25: Τρόπος εκτέλεσης της εφαρμογής (Run Configurations).

```
0#38#Temperature#21.05.2011 12:04:01#Motorola HMTS1050
1#92#Humidity#09.05.2011 11:45:00#Crossbow MDA300CA
2#60#WindSpeed#08.02.2011 16:03:01#Climatronics F460
3#13#Temperature#11.01.2011 22:03:31#Motorola HMTS1050
4#88#Humidity#15.05.2011 08:30:21#Crossbow MDA300CA
```

Εικόνα 26: Περιεχόμενα αρχείου "messages.txt".

6.2.4 Σύνοψη και μελλοντικές επεκτάσεις.

Στα πλαίσια αυτής της διπλωματικής επιτεύχθηκε η ανάπτυξη ενός ενδιάμεσου λογισμικού για τη βασικότερη εφαρμογή που άπτεται στα ασύρματα δίκτυα αισθητήρων και αφορά στην ανάκτηση και παρουσίαση των δεδομένων που ανιχνεύονται από τον αισθητήρα, σε ευανάγνωστη μορφή.

Η συγκεκριμένη υλοποίηση αποτελεί τη βάση στην οποία θα στηριχτεί ένας αριθμός διαδικτυακών υπηρεσιών (web services) τα οποία απαιτούν άμεση πρόσβαση στα δεδομένα από τους αισθητήρες.

Ο τρόπος κατασκευής του συστήματος οδήγησε σε μια σειρά πλεονεκτημάτων, όπως η ανεξαρτησία από τον τύπο του αισθητήρα και τις πλατφόρμες υλοποίησης, η ευελιξία, η επεκτασιμότητα και η εύκολη αποδόμηση του συστήματος σε επιμέρους υποσυστήματα. Ισχυρό εργαλείο στην προαναφερθείσα κατεύθυνση αποτελεί το OSGi framework, το οποίο δίνει τη δυνατότητα για περιγραφή της πληροφορίας με τέτοιο τρόπο που να συμβαδίζει με τα ανωτέρω στοιχεία του συστήματος. Η μεθοδολογία που ακολουθεί η παρούσα διπλωματική εργασία μπορεί να λειτουργήσει ως οδηγός για την επίλυση παρόμοιων προβλημάτων.

Η πλατφόρμα και η εφαρμογή της παρούσας υλοποίησης επιδέχονται αρκετές βελτιώσεις αλλά και επεκτάσεις. Μια πρόταση είναι η δημιουργία μιας πιο πολύπλοκης και εκτενούς διεπαφής που θα μπορεί να καλύψει κάθε είδος δικτύου αισθητήρων (π.χ. dependency injection και delegates). Επίσης μπορεί να εξελιχθεί και η επικοινωνία με το ίδιο το OSGi framework σε κάτι πιο ευέλικτο και επεκτάσιμο (π.χ. Eventing).

Επιπλέον, η ευκολίες που μας δίνει το OSGi framework, επιτρέπει την δημιουργία bundles για την εκτέλεση συγκεκριμένων λειτουργιών που θα μπορούν να συντηρούνται και να αναβαθμίζονται χωρίς να επηρεάζεται το κεντρικό σύστημα (π.χ. bundles για την επικοινωνία με άλλα middleware, ή με άλλες πλατφόρμες, ή ακόμη και bundles τύπου plug-in για την κάλυψη συγκεκριμένων αναγκών).

Τέλος μεγάλη βελτίωση επιδέχεται και ο client, ο οποίος θα μπορούσε να εκτελεί πιο πολύπλοκες λειτουργίες και ελέγχους (π.χ. ανάλογα το όνομα της υπηρεσίας), να δίνει πιο πολλές επιλογές στον χρήστη (π.χ. λίστες επιλογής, κουμπιά κλπ.) αλλά και να είναι πιο διαδραστικό. Τέλος θα μπορούσε να υλοποιηθεί και μια web-based έκδοση του client (π.χ. με Java Server Pages) ώστε να είναι διαθέσιμη η υπηρεσία και στο διαδίκτυο.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]. I.F. Akyildiz,W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless Sensor Networks: A Survey”. Computer Networks (Elsevier), March 2002.

[2]. I.F. Akyildiz,W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A Survey on Sensor Networks”, IEEE Communications Magazine, August 2002

[3]. Malik Tubaishat and Sanjay Madria, “Sensor networks: an overview”, IEEE Potentials, 22(2):20--23, April-May 2003.

[4]. Ning Xu, “A Survey of Sensor Network Applications”, Computer Science Department, University of Southern California.

[5]. C. Perkins, “Ad Hoc Networks”, Addison-Wesley, Reading, MA, 2000.

[6]. G. Hoblos, M. Staroswiecki, A. Aitouche, “Optimal design of fault tolerant sensor networks”, IEEE International Conference on Control Applications, Anchorage, AK, September 2000.

[7]. D. Nadig, S.S. Iyengar, “A new architecture for distributed sensor integration”, Proceedings of IEEE Southeastcon'93, Charlotte, NC, April 1993.

[8]. C. Shen, C. Srisathapornphat, C. Jaikaeo, “Sensor information networking architecture and applications”, IEEE Personal Communications, August 2001.

[9]. S. Cho, A. Chandrakasan, “Energy-efficient protocols for low duty cycle wireless microsensor”, Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Maui, HI Vol. 2 (2000).

[10]. N. Bulusu, D. Estrin, L. Girod, J. Heidemann, “Scalable coordination for wireless sensor networks: self-configuring localization systems”, International Symposium on Communication Theory and Applications (ISCTA 2001), Am-bleside, UK, July 2001.

[11]. A. Sinha, A. Chandrakasan, “Dynamic power management in wireless sensor networks”, IEEE Design and Test of Computers, March/April 2001.

[12]. E.M. Petriu, N.D. Georganas, D.C. Petriu, D. Makrakis, V.Z. Groza, “Sensor-based information appliances”, IEEE Instrumentation and Measurement Magazine (December 2000).

[13]. Salem Hadim, Nader Mohamed, “Middleware Challenges and Approaches for Wireless Sensor Networks”, IEEE Computer Society, March 2006.

[14]. Vivek Mhatre,Catherine Rosenberg,´´Design guidelines for Wireless sensor networks: communication,clustering and aggregation”, Computer Networks (Elsevier),July 2003

[15]. C. Intanagonwiwat, R. Govindan, D. Estrin, “Directed diffusion: a scalable and robust communication paradigm for sensor networks”, Proceedings of the ACM Mobi-Com'OO, Boston, MA, 2000.

[16]. G.J. Pottie, W.J. Kaiser, “Wireless integrated network sensors”, Communications of the ACM 43 (5) (2000).

[17]. Richard S. Hall, Karl Pauls, Stuart McCulloch, David Savage, “OSGi in Action”, Maning Publications, July 2010.

[18]. Xingchen Chu, Rajkumar Buyya", “Service Oriented Sensor Web”, May 2007

[19]. David E.Culler, Gilman Tolle, “Embedded Web Services Making Sense out of Diverse Sensors”, May 2007

[20]. Jing Deng , Richard Han ,and Shivakant Mishra, “A Performance Evaluation of Intrusion-Tolerant Routing in Wireless Sensor Networks”, Springer-Verlag Berlin Heidelberg 2003

[21]. L. Li, J.Y. Halpern, “Minimum-energy mobile wireless networks revisited”, IEEE International Conference on Communications ICC'Ol, Helsinki, Finland, June 2001.

[22]. A. Boulis, C.C. Han, and M. B. Srivastava, "Design and Implementation of a Framework for Efficient and Programmable Sensor Networks", In MobiSys, San Francisco, CA, 2003.

[23]. A. Savvides, C. Han, M. Srivastava, “Dynamic fine-grained localization in ad-hoc networks of sensors”, Proceedings of ACM MobiCom'Ol, Rome, Italy, July 2001.

[24]. S. Meguerdichian, F. Koushanfar, G. Qu, M. Potkonjak, “Exposure in wireless ad-hoc sensor networks”, Proceedings of ACM MobiCom'Ol, Rome, Italy, 2001.

[25]. B. Warneke, B. Liebowitz, K.S.J. Pister, “Smart dust: communicating with a cubic-millimeter computer”, IEEE Computer (January 2001).

[26]. Chris Karlof, David Wagner, “Secure routing in wireless sensor networks:attacks and countermeasures”, Computer Science (Elsevier), 2003

[27]. K. Govil, E. Chan, H. Wasserman, “Comparing algorithms for dynamic speed-setting of a low-power CPU”, Proceedings of ACM MobiCom'95, Berkeley, CA, November 1995.

[28]. J. Lorch, A. Smith, “Reducing processor power consump-tion by improving processor time management in a single-user operating system”, Proceedings of ACM MobiCom'96, 1996.

[29]. J. Ross Beveridge, Christopher R. Graves, and Christopher E. Lesher. Lo-cal search as a tool for horizon line matching. In Image Understanding Workshop, Los Altos, CA, February 1996. ARPA, Morgan Kaugmann.

[30]. Shuoqi Li, Sang H. Son, and John A. Stankovic, "Event Detection Services Using Data Service Middleware in Distributed Sensor Networks", In IPSN 2003, Palo Alto, USA, April 2003.

[31]. Amy Murphy Wendi Heinzelman, "Milan: Middleware Linking Applications and Networks", Technical Report TR-795, Univ. of Rochester, CS dept., Nov. 2002.

[32]. Lyndell St. Ville and Peter Dickman, "Garnet: A Middleware Architecture for Distributing Data Streams Originating in Wireless Sensor Networks", 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03), May 2003.

[33]. M. Weiser et al., “Scheduling for reduced CPU energy”, Proceedings of 1st USENIX Symposium on Operating System Design and Implementation, November 1994.

[34]. D. Estrin, R. Govindan, J. Heidemann, S. Kumar, “Next century challenges: scalable coordination in sensor networks”, ACM MobiCom'99, Washingtion, USA, 1999.

[35]. J. Agre, L. Clare, “An integrated architecture for cooperative sensing networks”, IEEE Computer Magazine (May 2000).

[36]. M. Bhardwaj, T. Garnett, A.P. Chandrakasan, “Upper bounds on the lifetime of sensor networks”, IEEE International Conference on Communications ICC'Ol, Helsinki, Finland, June 2001.

[37]. Manish Kochhal Loren Schwiebert Sandeep Gupta, "Role-based Middleware for Sensor Networks", Wayne State University, WSU-CSC-NEWS/04-TR01, May 2004.

[38]. P. Bonnet, J. Gehrke, P. Seshadri, “Querying the physical world”, IEEE Personal Communications (October 2000).

[39]. A.S. Tanenbaum, “Computer Networks”, Prentice Hall, New Jersey, 1996.

[40]. W.R. Heinzelman, J. Kulik, H. Balakrishnan, “Adaptive protocols for information dissemination in wireless sensor networks”, Proceedings of the ACM MobiCom'99, Seattle, Washington, 1999.

[41]. S. Slijepcevic, M. Potkonjak, “Power efficient organization of wireless sensor networks”, IEEE International Conference on Communications ICC'Ol, Helsinki, Finland, June 2001.

[42]. A. Chandrakasan, R. Amirtharajah, S. Cho, J. Goodman, G. Konduri, J. Kulik, W. Rabiner, A. Wang, “Design considerations for distributed micro-sensor systems”, Proceedings of the IEEE 1999 Custom Integrated Circuits Conference, San Diego, CA, May 1999.

[43]. Kemal Akkaya, Mohamed Younis, " A Survey on Routing Protocols for Wireless Sensor Networks”,

[44]. OSGi Alliance –http://www.osgi.org

[45]. OSGi Service Platform Release 4.3 (April 2011) - http://www.osgi.org/Specifications/HomePage

[46]. Apache Felix - http://felix.apache.org

[47]. The Eclipse Foundation - http://www.eclipse.org/

[48]. *http://en.wikipedia.org/wiki/Wireless_sensor_network*