ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

# "Σχεδιασμός και ανάπτυξη ενός γενικού περιβάλλοντος για υλοποίηση εφαρμογών σε ασύρματα δίκτυα αισθητήρων"

Γεώργιος Χ. Μυλωνάς, ΑΜ 352
Μεταπτυχιακό Δίπλωμα Ειδίκευσης: *Επιστήμη και Τεχνολογία Υπολογιστών*
Τμήμα Μηχανικών Η/Υ και Πληροφορικής
mylonasg@ceid.upatras.gr

**Επιβλέπων:** Σωτήρης Νικολετσέας

**Τριμελής Επιτροπή:** Μάνος Βαρβαρίγος, Καθηγητής
Σωτήρης Νικολετσέας, Λέκτορας
Παύλος Σπυράκης, Καθηγητής

18 Ιουλίου 2005

Κάθε αυθεντικό αντίτυπο φέρει την υπογραφή του συγγραφέα...

*Αφιερωμένο στους γονείς μου, στον αδερφό μου,*
*στους φίλους μου και σε όλους όσους με βοήθησαν να βρω τον εαυτό μου*
*(και ειδικά στη Β.).*

# Πρόλογος

Τα ασύρματα δίκτυα αισθητήρων (wireless sensor networks) αποτελούν μια πολύ νέα κατηγορία δικτύων υπολογιστών. Αυτά τα δίκτυα αποτελούνται από ένα μεγάλο πλήθος υπολογιστικών κόμβων μικροσκοπικού μεγέθους, εφοδιασμένων με πλήθος αισθητήρων και μονάδων ελέγχου. Σκοπός τους είναι η επίτευξη μιας δύσκολης, για τα δεδομένα του κάθε κόμβου, αποστολής μέσω της συνεργασίας μεταξύ όλων των κόμβων του δικτύου.

Τα δίκτυα αυτά αντιμετωπίζονται με μεγάλο ενδιαφέρον από την ερευνητική κοινότητα τα τελευταία χρόνια. Έτσι, έχει προταθεί πλήθος από πρωτόκολλα διάδοσης πληροφορίας, πιθανές εφαρμογές, έχει υλοποιηθεί πλήθος λογισμικού, κτλ. Υπάρχει όμως περιορισμός στα διαθέσιμα εργαλεία για την ανάπτυξη εφαρμογών σε τέτοια δίκτυα, το οποίο σημαίνει από τη μια περιορισμός στις διαθέσιμες στους τελικούς χρήστες δυνατότητες και από την άλλη αυξημένη δυσκολία υλοποίησης κάποιας εφαρμογής.

Σκοπός της εργασίας αυτής είναι ο σχεδιασμός και η ανάπτυξη ενός γενικού περιβάλλοντος, το οποίο θα προσφέρει τη δυνατότητα στους χρήστες να υλοποιούν γρήγορα μια εφαρμογή σε ένα δίκτυο ασύρματων αισθητήρων. Το περιβάλλον αυτό, το οποίο ονομάζουμε jWebDust, εκτείνεται σε όλα τα επίπεδα που μπορεί να περιλαμβάνει μια τέτοια εφαρμογή, από το επίπεδο δικτύου ως το επίπεδο παρουσίασης, και επιτρέπει στο χρήστη να αναπτύξει μια εφαρμογή χωρίς να χρειαστεί να υλοποιήσει ο ίδιος τα επίπεδα αυτά. Η υλοποίηση του περιβάλλοντος αυτού βασίζεται σε υπάρχουσες τεχνολογίες, όπως το λειτουργικό σύστημα TinyOS, και απευθύνεται στις πλατφόρμες που υποστηρίζουν το σύστημα αυτό, όπως είναι τα MICA motes.

Η συνεισφορά της παρούσας εργασίας συνοψίζεται σε δύο συνιστώσες:

1. Το πρωτόκολλο VTRP [6], το οποίο προσπαθεί να λύσει το πρόβλημα του εντοπισμού και διάδοσης πολλαπλών γεγονότων (*multiple event detection and propagation*) στα ασύρματα δίκτυα αισθητήρων. Το πρόβλημα αυτό συνίσταται στον εντοπισμό ενός πλήθους από γεγονότα μέσα στο δίκτυο και της διάδοσης των αντίστοιχων αναφορών σε ένα κέντρο ελέγχου, με έναν αποδοτικό τρόπο όσον αφορά στην κατανάλωση ενέργειας και στην ανοχή σε σφάλματα μετάδοσης. Το στοιχείο που διαφοροποιεί το VTRP σε σχέση με τα υπόλοιπα συναφή πρωτόκολλα είναι η μεταβολή της εμβέλειας μετάδοσης των κόμβων του ασύρματου δικτύου αισθητήρων,

όταν αυτό επιβάλλεται από τις συνθήκες που επικρατούν στο δίκτυο.

2. Το περιβάλλον jWebDust [8], το οποίο έχει σαν στόχο να βοηθήσει τον τελικό χρήστη να υλοποιήσει μια εφαρμογή για ένα δίκτυο ασύρματων αισθητήρων εύκολα και γρήγορα. Το jWebDust παρέχει στο χρήστη ένα απλό interface μέσω του οποίου μπορεί να εκτελέσει τις απαραίτητες ενέργειες για τη διαχείριση ενός ασύρματου δικτύου αισθητήρων και να συλλέξει πληροφορίες από τους κόμβους του δικτύου αυτού.

Από πλευράς οργάνωσης, η εργασία περιλαμβάνει τα ακόλουθα μέρη:

**1. Έρευνα αιχμής στα ασύρματα δίκτυα αισθητήρων (κεφάλαια $1-3$):** Στο πρώτο μέρος κάνουμε μια εισαγωγή των βασικών εννοιών που συναντάμε στα ασύρματα δίκτυα αισθητήρων και μια μικρή αναδρομή στην πορεία τους μέχρι στιγμής. Ακολουθεί μια παρουσίαση της έρευνας αιχμής στα ασύρματα δίκτυα αισθητήρων, όσον αφορά τις σχετικές hardware πλάτφορμες και τις λύσεις που υπάρχουν σε λογισμικό ως τώρα.

**2. Παρουσίαση του πρωτοκόλλου VTRP (κεφάλαιο $4$):** παρουσιάζουμε αναλυτικά τη λειτουργία του VTRP και παραθέτουμε πειραματικά αποτελέσματα από εξομοιώσεις που κάναμε μέσω του εξομοιωτή simDust [1], [2].

**3. Αναλυτική παρουσίαση του συστήματος jWebDust (κεφάλαια $5-6$):** Παρουσιάζουμε συνολικά την αρχιτεκτονική του jWebDust και παρουσιάζουμε αναλυτικά τις προδιαγραφές λειτουργίας για κάθε επίπεδο του συστήματος.

Καλή ανάγνωση!

# Ευχαριστίες

Θα ήθελα πρώτα από όλους να ευχαριστήσω τους γονείς μου, για την υπομονή και την κατανόηση που έδειξαν κατά τη διάρκεια της εκπόνησης της εργασίας που κρατάτε στα χέρια σας (ή βλέπετε στην οθόνη μπροστά σας). Ελπίζω να απόκτησω και εγώ κάποτε την ίδια ιώβεια υπομονή.

Κατόπιν, θα ήθελα να ευχαριστήσω το φίλο και συνεργάτη Δρ. Ιωάννη Χατζηγιαννάκη. Κάποια από τα αποτελέσματα της συνεργασίας μας περιέχονται στην παρούσα εργασία. Ο Γιάννης αποτελεί ένα ακόμα λαμπρό παράδειγμα καρτερίας, καθώς μοιραστήκαμε το ίδιο γραφείο για ένα χρόνο.

Θα ήθελα επίσης να ευχαριστήσω το Δρ. Σωτήρη Νικολετσέα, λέκτορα του τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Πατρών, για την επίβλεψη της παρούσας εργασίας και τη συνεργασία μας κατά τη διάρκεια της εκπόνησής της.

Τέλος, θα ήθελα να ευχαριστήσω το φίλο και συνεργάτη Θανάση Αντωνίου, για την πολύτιμη συνεισφορά του σχετικά με το πρωτόκολλο δρομολόγησης VTRP για ασύρματα δίκτυα αισθητήρων, το οποίο παρουσιάζεται στην εργασία αυτή.

"How many roads must a man walk down? 42" - Douglas Adams,
*Γυρίστε το Γαλαξία με Ωτοστόπ*

# Περιεχόμενα

# Μέρος Ι

# Ασύρματα δίκτυα αισθητήρων

# Κεφάλαιο 1

# Εισαγωγή στα ασύρματα δίκτυα αισθητήρων

*"Freedom is free of the need to be free"*
George Clinton

## 1.1 Γενικά εισαγωγικά στοιχεία

Μια ιδέα η οποία χρησιμοποιείται όλο και περισσότερο τα τελευταία χρόνια, σε διάφορα πεδία της επιστήμης των υπολογιστών και με διάφορες μορφές, είναι η χρήση ενός πλήθους υπολογιστικών συστημάτων για την επίτευξη ενός κοινού στόχου. Σε συνδυασμό με τις εξελίξεις στον τομέα του υλικού (hardware) και των ασύρματων τηλεπικοινωνιών, με τις οποίες μπορούμε πλέον να έχουμε υπολογιστές σε πολύ μικρό μέγεθος που να επικοινωνούν μεταξύ τους ασύρματα, προκύπτει ένα πλήθος από νέες δυνατότητες και εφαρμογές.

Πάνω στην απλή αυτή ιδέα λοιπόν βασίζονται τα *ασύρματα δίκτυα αισθητήρων* (wireless sensor networks), ή αλλιώς *δίκτυα έξυπνης σκόνης* (smart dust networks). Τα δίκτυα αυτά αποτελούνται από ένα πολύ μεγάλο αριθμό κόμβων, οι οποίοι έχουν πολύ μικρό μέγεθος, και έχουν κάποιους αισθητήρες ενσωματωμένους συν κάποιες δυνατότητες επεξεργασίας και ασύρματης επικοινωνίας. Οι κόμβοι των δικτύων αυτών συνεργάζονται για να φέρουν εις πέρας μια δύσκολη, για τα δεδομένα κάθε τέτοιου κόμβου, αποστολή.

Επειδή αφενός η ονομασία "δίκτυα έξυπνης σκόνης" αναφέρεται σε συγκεκριμένη hardware πλατφόρμα και αφετέρου δεν είναι ακόμα ευρέως διαθέσιμη κάποια πλατφόρμα σε διαστάσεις "σκόνης", χρησιμοποιούμε την ονομασία "ασύρματα δίκτυα αισθητήρων" στην παρούσα εργασία. Στη σχετική βιβλιογραφία πάντως συναντάμε και τις δύο ονομασίες.

Τα ασύρματα δίκτυα αισθητήρων έχουν ένα αρκετά ευρύ πεδίο εφαρμογών, το οποίο είναι ακόμα ανοιχτό, δηλαδή συνεχώς γίνονται νέες προτάσεις για εφαρμογές στις οποίες τα ασύρματα δίκτυα αισθητήρων αντικαθιστούν τις υπάρχουσες μεθόδους ή μας δίνουν δυνατότητες που πριν ήταν δύσκολο να τις φανταστούμε στην πράξη. Όλες αυτές οι εφαρμογές βασίζονται σε ένα κοινό πρότυπο, δηλαδή έχουμε ένα μεγάλο πλήθος από κόμβους για ασύρματα δίκτυα αισθητήρων, τους οποίους τοποθετούμε σε κάποια συγκεκριμένη περιοχή, και οι οποίοι με βάση το λογισμικό που τρέχουν παίρνουν μετρήσεις από το περιβάλλον μέσω των αισθητήρων τους και τις παραδίδουν με κάποιο δικτυακό πρωτόκολλο σε ένα *κέντρο ελέγχου* (control center ή αλλιώς sink node).

Σε αντίθεση με την απλότητα της αρχικής ιδέας, η υλοποίηση των δικτύων αυτών δεν είναι καθόλου εύκολη υπόθεση. Μάλιστα οι προκλήσεις που παρουσιάζονται σε ερευνητικό επίπεδο είναι πάρα πολλές και επίσης, σε πολλά πεδία. Αρκεί να αναλογιστούμε ότι επιδιώκουμε να κατασκευάσουμε κόμβους δικτύου τόσο μικρούς, ώστε να μοιάζουν με σκόνη, να επικοινωνούν σε αποστάσεις ακόμα και των εκατοντάδων μέτρων, να αντέχουν σε αντίξοες συνθήκες, να καταναλώνουν ελάχιστη ενέργεια. Μόλις τα τελευταία χρόνια έγινε δυνατόν να κατασκευαστούν ολοκληρωμένα που να πληρούν κάποιες από αυτές τις προϋποθέσεις, χάρη στην πρόοδο στους τομείς του VLSI και των ασύρματων επικοινωνιών.

Το συγκριτικό πλεονέκτημα των ασύρματων δικτύων αισθητήρων σε σχέση με τα παραδοσιακά δίκτυα αισθητήρων έγκειται αφενός στο ότι αποτελούνται από μεγάλο πλήθος μικροσκοπικών κόμβων που μπορούν να σχηματίσουν ένα δίκτυο και να ρυθμίσουν τη λειτουργία τους από μόνοι τους (με το κατάλληλο λογισμικό) και αφετέρου στο μικρό τους κόστος.

Ένα πολύ παραστατικό παράδειγμα σε σχέση με το κόστος των ασύρματων δικτύων και τις δυνατότητές τους, αναφέρεται στο [17]. Αν υποθέσουμε ότι ένας κόμβος ασύρματου δικτύου αισθητήρων έχει εμβέλεια μετάδοσης 50 μέτρα και ότι σε λίγα χρόνια από τώρα ένας τέτοιος κόμβος θα κοστίζει 1 δολάριο, ένα τέτοιο δίκτυο θα μπορούσε να απλωθεί κατά μήκος του Ισημερινού της Γης με κόστος μικρότερο από ένα εκατομμύριο δολάρια. Το παράδειγμα αυτό μπορεί να μην έχει κάποια πρακτική χρησιμότητα, παρ' όλα αυτά όμως είναι άκρως εντυπωσιακό.

Ακολουθεί μια σύντομη περιγραφή των διαφορών ανάμεσα στα δίκτυα έξυπνης σκόνης και τα ήδη υπάρχοντα δίκτυα, και των παραγόντων που επιδρούν συνολικά στη σχεδίαση ενός τέτοιου δικτύου. Ακολουθεί η γενική αρχιτεκτονική του δικτύου. Στη συνέχεια κάνουμε μια αναφορά στις μελλοντικές προοπτικές των δικτύων αυτών, καθώς και σε σενάρια εφαρμογής τους. Τέλος, περιγράφουμε δύο εφαρμογές ασυρμάτων δικτύων αισθητήρων για να δείξουμε τη χρησιμότητά τους σε πρακτικό επίπεδο.

Ένα εξαιρετικό survey, το οποίο μπορεί να χρησιμοποιήσει ο αναγνώστης ως εισαγωγή στο πεδίο των ασύρματων δικτύων αισθητήρων, είναι το [5], μαζί με το [4]. Επίσης, οι εργασίες [1] και [2] θα φανούν χρήσιμες σαν πηγές στον αναγνώστη, οι οποίες επιπλέον είναι γραμμένες στα Ελληνικά.

## 1.2 Διαφορές των ασύρματων δικτύων αισθητήρων σε σχέση με τους κλασσικούς αισθητήρες και τα δίκτυα ad hoc

Τα ασύρματα δίκτυα αισθητήρων αποτελούν σημαντική βελτίωση σε σχέση με τους παραδοσιακούς αισθητήρες, οι οποίοι χρησιμοποιούνται με τους παρακάτω δύο τρόπους:

1. Τοποθετούνται σχετικά μακριά από το φαινόμενο προς παρατήρηση και απαιτούνται ακριβοί αισθητήρες προκειμένου να έχουν τη δυνατότητα να φιλτράρουν το θόρυβο από το περιβάλλον, ο οποίος υπεισέρχεται ακριβώς λόγω της απόστασης από το φαινόμενο προς μέτρηση, ώστε να δίνουν αξιόπιστα αποτελέσματα.

2. Τοποθετούνται σε *μικρούς αριθμούς* με κάποιο *προσχεδιασμένο* τρόπο και μεταδίδουν συνεχώς μετρήσεις με κάποιο ενσύρματο, συνήθως, δίκτυο σε προκαθορισμένες χρονικές στιγμές επικοινωνόντας απ' ευθείας με κάποιο κεντρικό σταθμό.

Σε αντίθεση με τους κλασσικούς αισθητήρες, ένα δίκτυο έξυπνης σκόνης:

1. Αποτελείται από *πολύ μεγάλο αριθμό κόμβων* και η ανάπτυξή του στο πεδίο ενδιαφέροντος μπορεί να γίνεται με εντελώς τυχαίο τρόπο.

2. Οι κόμβοι του δικτύου βρίσκονται *μέσα* στο πεδίο ενδιαφέροντος. Οι μετρήσεις που λαμβάνουμε είναι συνεπώς ακριβείς.

3. Αντί να στέλνει ο κάθε κόμβος κατευθείαν τις μετρήσεις που παίρνει σε κάποιο κεντρικό (βασικό) σταθμό, χρησιμοποιείται ένα πιο σύνθετο πρωτόκολλο δικτύου για να προωθηθεί η πληροφορία προς το βασικό σταθμό, σε περισσότερα από ένα βήματα.

4. Εκτός από τους αισθητήρες που φέρουν οι κόμβοι αυτών των δικτύων, μπορούν να συνδεθούν με *μηχανισμούς κίνησης και ελέγχου* (actuators), προσδίδοντας με τον τρόπο αυτό επιπλέον δυνατότητες.

5. Τα ασύρματα δίκτυα αισθητήρων μπορούν να μειώσουν δραματικά το κόστος υλοποίησης μιας εφαρμογής σε σχέση με το αντίστοιχο κόστος εγκατάστασης και συντήρησης των απλών αισθηρήτων.

6. Έχουν την ικανότητα να προσαρμόζονται δυναμικά στις συνθήκες που επικρατούν στο περιβάλλον στο οποίο λειτουργούν. Μπορούν να ανταποκριθούν σε περιπτώσεις όπου αλλάζει η τοπολογία και η συνδεσιμότητα του δικτύου.

Διαβάζοντας κάποιος τα προήγουμενα χαρακτηριστικά, μπορεί να συμπεράνει ότι τα ασύρματα δίκτυα αισθητήρων είναι απλά ad hoc δίκτυα και να αναρωτηθεί: "γιατί δεν εφαρμόζουμε απλά τις μεθόδους και τα πρωτόκολλα που

έχουμε αναπτύξει τα τελευταία χρόνια για τα ad hoc δίκτυα;''. Η απάντηση είναι ότι τα ασύρματα δίκτυα αισθητήρων, αν και παρουσιάζουν ομοιότητες με τα κλασσικά ad hoc δίκτυα, δεν ταυτίζονται μαζί τους, ενώ και οι ήδη υπάρχουσες μέθοδοι για ad hoc δίκτυα δεν επαρκούν για τα ασύρματα δίκτυα αισθητήρων. Παραθέτουμε τους κυριότερους λόγους για αυτό:

- Ο αριθμός των κόμβων σε ένα ασύρματο δίκτυο αισθητήρων αναμένεται να είναι πολύ μεγαλύτερος από ότι σε ένα ad hoc δίκτυο.
- Οι κόμβοι είναι επιρρεπείς σε αστοχίες (failures), κυρίως σε επίπεδο υλικού (hardware), και επομένως και η τοπολογία μπορεί να αλλάζει με μεγαλύτερη συχνότητα από ότι στα ad hoc δίκτυα.
- Υπάρχουν πολύ μεγαλύτεροι περιορισμοί σε επεξεργαστικές δυνατότητες, μνήμη και ενέργεια.
- Τα ασύρματα δίκτυα αισθητήρων καλούνται να καλύψουν διαφορετικές εφαρμογές σε σχέση με τα ad hoc δίκτυα, με διαφορετικές απαιτήσεις (π.χ. quality of service, διάρκεια ζωής, κ.α).

Αυτές ήταν εν συντομία οι κύριες διαφορές των δικτύων έξυπνης σκόνης με τους κλασσικούς αισθητήρες και τα ad hoc δίκτυα. Στη συνέχεια, θα δούμε πως αυτές οι διαφορές μεταφράζονται σε παράγοντες που επηρεάζουν τη σχεδίαση ενός τέτοιου δικτύου.

## 1.3 Παράγοντες οι οποίοι επηρεάζουν συνολικά το σχεδιασμό ενός ασύρματου δικτύου αισθητήρων

**Προσαρμοστικότητα του δικτύου:** Ένα ασύρματο δίκτυο αισθητήρων έχει να αντιμετωπίσει δυσκολίες όπως συχνές αστοχίες υλικού, συχνά προβλήματα στην επικοινωνία μεταξύ κόμβων και μεγάλο μέγεθος και πυκνότητα δικτύου. Οι αλγόριθμοι, οι οποίοι ρυθμίζουν την επικοινωνία μεταξύ των κόμβων πρέπει να έχουν μεγάλες ανοχές σε λάθη, να αντιλαμβάνονται γρήγορα πότε μια σύνδεση δε λειτουργεί σωστά και να δημιουργούν καινούριες. Επιπλέον, πρέπει να αντιμετωπίσουν τη μεγάλη πυκνότητα του δικτύου και να μπορούν να εφαρμόζονται σε περιπτώσεις στις οποίες έχουμε αρκετές χιλιάδες κόμβων.

**Τοπολογία δικτύου:** Η τοπολογία ενός ασύρματου δικτύου αισθητήρων μπορεί να είναι εντελώς τυχαία ή να ακολουθεί κάποιο πρότυπο. Μπορεί να γίνει ρίψη συσκευών από αεροπλάνο (η συγκεκριμένη μέθοδος έχει ήδη δοκιμαστεί), από καταπέλτη, με τα χέρια, κτλ. Μετά την τοποθέτηση των κόμβων η τοπολογία μπορεί να αλλάξει λόγω θέσης, συνδεσιμότητας, έλλειψης ενέργειας. Επιπλέον, νέοι κόμβοι μπορεί να τοποθετηθούν για να αντικαταστήσουν παλιούς που τέθηκαν εκτός λειτουργίας ή για να επεκτείνουν το σύστημα. Επομένως, το σύστημα πρέπει να αντιδρά δυναμικά απέναντι σε όλους αυτούς τους παράγοντες.

**Μέσο μετάδοσης:** Η επικοινωνία μεταξύ των κόμβων του δίκτυου γίνεται μέσω ραδιοσυχνοτήτων, αν και έχουν παρουσιαστεί σενάρια για επικοινωνία μέσω laser. Για τις ραδιοσυχνότητες πρέπει να επιλεγεί ζώνη μετάδοσης, η οποία θα είναι διαθέσιμη για εμπορική χρήση, π.χ οι συχνότητες που χρησιμοποιούνται από τα wireless LAN. Επιπλέον, πρέπει να υπάρχουν μεγάλα περιθώρια ανοχής σε θόρυβο και παρεμβολές, δεδομένων των συνθηκών που επικρατούν στους χώρους που ενδέχεται να χρησιμοποιηθούν τα ασύρματα δίκτυα αισθητήρων. Υπάρχει η επιλογή μεταξύ narrowband και wideband ραδιοσυχνοτήτων. Στην πρώτη περίπτωση έχουμε το πλεονέκτημα της χαμηλής κατανάλωσης ενέργειας και της απλής υλοποίησης σε hardware, ενώ στη δεύτερη περίπτωση έχουμε το πλεόνεκτημα της μεγαλύτερης ανοχής σε θόρυβο και του μεγαλύτερου data rate. Μια απόπειρα για ένα πρότυπο για ασύρματη επικοινωνία με τέτοια χαρακτηριστικά περιγράφεται στο κεφάλαιο 3 (IEEE 802.15.4).

**Περιορισμοί υλικού και κόστους παραγωγής:** Ένας κόμβος ασύρματου δικτύου αισθητήρων αποτελείται από τέσσερα βασικά τμήματα, έναν αισθητήρα(ή και περισσότερους), μία CPU, ένα πομποδέκτη και μια μονάδα ισχύος. Επίσης, είναι δυνατόν να υπάρχουν κι άλλα τμήματα, όπως μονάδα GPS ή κινητήρας. Όλες αυτές οι μονάδες πρέπει να χωρούν σε μια συσκευασία μεγέθους σπιρτόκουτου ή, ιδανικά, ακόμα μικρότερη. Ακόμα, πρέπει να καταναλώνουν ελάχιστη ενέργεια, να είναι αυτόνομες και να λειτουργούν ανεξάρτητες, να προσαρμόζονται στο περιβάλλον τους, να αντέχουν σε αντίξοες συνθήκες. Επομένως, υπάρχουν πολύ μεγάλες απαιτήσεις στον τομέα του υλικού, και πρέπει η έρευνα στον τομέα της κατασκευής ολοκληρωμένων κυκλωμάτων να υποστηρίξει αυτές τις ανάγκες.

Στα παραπάνω έρχεται να προστεθεί η απαίτηση για πολύ μικρό κόστος παραγωγής ενός κόμβου. Ένα ασύρματο δίκτυο αισθητήρων προφανώς είναι συμφέρον, αν κάθε κόμβος στοιχίζει τόσο λίγο ώστε το συνολικό κόστος να μην είναι απαγορευτικό. Εφόσον μιλάμε για χιλιάδες κόμβων, το κόστος του κάθε κόμβου πρέπει να είναι της τάξης του ενός ευρώ. Συγκριτικά, μια συσκευή Bluetooth πρόσφατα κόστιζε κοντά στα 10$.

**Κατανάλωση ενέργειας:** Η λειτουργία ενός κόμβου-αισθητήρα συνοψίζεται ως εξής: ανίχνευση γεγονότος, επεξεργασία πληροφορίας, μετάδοση ή λήψη μηνύματος. Επομένως, η ενέργεια ενός κόμβου ξοδεύεται με τρεις τρόπους.

- *Επικοινωνία:* Η περισσότερη ενέργεια καταναλώνεται εδώ, στην οποία περιλαμβάνεται τόσο η μετάδοση όσο και η λήψη δεδομένων. Γενικά, για μικρές αποστάσεις και για τις δύο περιπτώσεις καταναλώνεται περίπου η ίδια ενέργεια. Επίσης, λαμβάνουμε υπόψη μας και την ενέργεια για την αρχικοποίηση του πομποδέκτη. Επομένως, πρέπει να βρούμε έναν έξυπνο τρόπο να διαχειριστούμε τον πομποδέκτη και να ελαχιστοποιήσουμε μεταδόσεις και λήψεις μηνυμάτων. Τυπικές τιμές για αυτήν την περιοχή είναι 50 $nJ/bit$.

- *Επεξεργασία πληροφορίας:* Εδώ, η κατανάλωση ενέργειας είναι πολύ μικρότερη σε σχέση με την ενέργεια για επικοινωνία. Μπορούμε να εκμεταλλευτούμε την ισχύ του επεξεργαστή και να κάνουμε συμπίεση δεδομένων ή μπορούμε ακόμα και να επεξεργαστούμε τα μηνύματα που λαμβάνουμε και να αφαιρέσουμε περιττή πληροφορία (π.χ το ίδιο μήνυμα έχει φθάσει από δύο κόμβους), προκειμένου να γλιτώσουμε bit μεταδιδόμενης πληροφορίας. Τυπική τιμή για τους σύγχρονους επεξεργαστές είναι 1 pJ/εντολή.

- *Μέτρηση πεδίου:* Η ενέργεια που καταναλώνεται εδώ είναι σταθερή και περιλαμβάνει την ενέργεια για λήψη δείγματος από τους αισθητήρες ενός κόμβου. Προφανώς, μπορούμε να ελαττώσουμε την ενέργεια που καταναλώνουμε, όσο πιο αραιά κάνουμε μετρήσεις της ποσότητας που μας ενδιαφέρει (π.χ θερμοκρασία). Τυπικές τιμές σε αυτό το πεδίο είναι 4 nJ ανά δείγμα (ακρίβεια 10 bit/δειγμα).

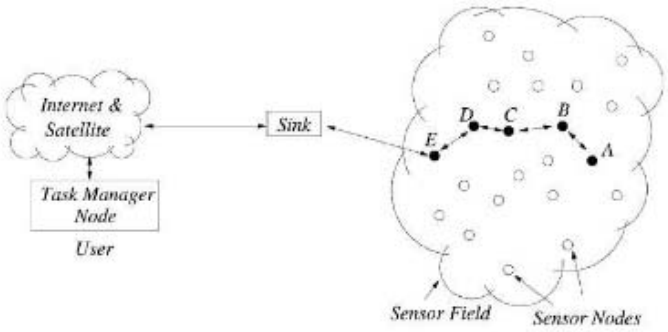
**Πηγές ενέργειας:** Υπάρχουν αρκετές τεχνολογικές προτάσεις ως υποψήφιες πηγές ενέργειας για χρήση σε ασύρματα δίκτυα αισθητήρων, καθεμιά με τα πλεονεκτήματα και τα μειονεκτήματά της. Προφανώς αυτό που ενδιαφέρει περισσότερο είναι η διάρκεια ζωής και η αυτονομία που προσφέρουν στους κόμβους του δικτύου, αλλά υπάρχουν και άλλες σημαντικές παράμετροι όπως το μέγεθος, η ευκολία στη σύνδεση με τους κόμβους, κ.α. Προς το παρόν, οι υπάρχουσες λύσεις προσφέρουν περιορισμένη αυτονομία.

Η τεχνολογία η οποία χρησιμοποιείται ως επί το πλείστον είναι οι μπαταρίες λιθίου. Οι τελευταίες γενιές των μπαταριών αυτών είναι επαναφορτιζόμενες, γεγονός το οποίο είναι πολύ χρήσιμο. Μια άλλη πρόταση από το χώρο των μπαταριών, είναι οι νέες μπαταρίες λεπτού φιλμ οξειδίου βαναδίου και οξειδίου μολυβδαινίου. Οι χωρητικότητες τους είναι ανάλογες των μπαταριών λιθίου και επιπλέον έχουν το πλεονέκτημα ότι μπορούν να ενσωματωθούν στους κόμβους πιο εύκολα.

Μια άλλη πρόταση είναι η χρήση φωτοηλεκτρικών κυττάρων, τα οποία παράγουν ενέργεια από το φως. Το φως αυτό μπορεί να προέρχεται είτε από τον Ήλιο, είτε από κάποια άλλη φωτεινή πηγή, όπως ο εσωτερικός φωτισμός. Βέβαια, στη δεύτερη περίπτωση η παραγόμενη ενέργεια είναι κλάσμα της πρώτης. Η λύση αυτή μπορεί να χρησιμοποιηθεί περισσότερο ως ένας τρόπος να αναπληρώνεται η ενέργεια μιας μπαταρίας, η οποία θα είναι η κύρια πηγή ενέργειας. Αν αυτό μπορεί να γίνεται με έναν ικανοποιητικό ρυθμό, τότε μπορεί να αυξηθεί αρκετά ο χρόνος ζωής ενός ασύρματου δικτύου αισθητήρων.

## 1.4 Αρχιτεκτονική ενός ασύρματου δικτύου αισθητήρων

Ένα ασύρματο δίκτυο αισθητήρων σε γενικές γραμμές αποτελείται από τους απλούς κόμβους του δικτύου (sensor nodes ή particles ή motes), οι οποίοι βρίσκονται διασκορπισμένοι σε μία περιοχή που μας ενδιαφέρει (sensor field), και από ένα *κέντρο ελέγχου*, το οποίο συγκεντρώνει και αποθηκεύει ή προωθεί

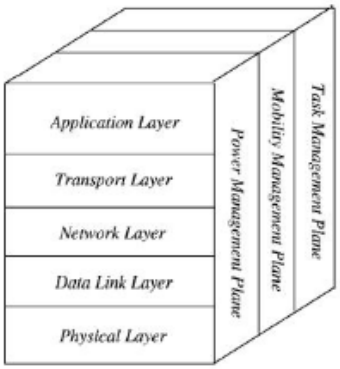



Σχήμα 1.1: Γενική άποψη ενός ασύρματου δικτύου αισθητήρων

περαιτέρω τις πληροφορίες που συλλέγουν οι κόμβοι μέσα στο πεδίο. Το κέντρο ελέγχου μπορεί να είναι τοποθετημένο σε κάποια μικρή απόσταση από το πεδίο και θεωρούμε ότι έχει πολύ μεγαλύτερα αποθέματα ενέργειας, σε σχέση με τους απλούς κόμβους του δικτύου. Η ροή πληροφορίας προς το κέντρο ελέγχου επιτυγχάνεται μέσω της συνεργασίας των κόμβων του δικτύου και κάποιου δικτυακού πρωτοκόλλου. Επίσης, θεωρούμε ότι το κέντρο ελέγχου μπορεί να είναι συνδεδεμένο και με κάποιο άλλο δίκτυο, π.χ το Internet, με κάποιο τρόπο (ενσύρματο ή ασύρματο). Η οργάνωση αυτή φαίνεται στο σχήμα 1.1.

Η στοίβα με τα επίπεδα δικτύου ενός ασύρματου δικτύου αισθητήρων, σύμφωνα με το [5], φαίνεται στο σχήμα 1.2. Αποτελείται από τα επίπεδα δικτύου, όπως αυτά ορίζονται στο μοντέλο OSI αλλά χωρίς επίπεδο session, συν τρία νέα επίπεδα, τα οποία διατρέχουν ολόκληρη την ιεραρχία επιπέδων, τα power, mobility και task management επίπεδα. Εκτός των τριών τελευταίων, τα υπόλοιπα επίπεδα επιτελούν ακριβώς τις γνωστές λειτουργίες όπως τις γνωρίζουμε από τα IP δίκτυα. Τα τρία τελευταία παρακολουθούν την κατανομή κατανάλωσης ενέργειας και φόρτου εργασίας σε ένα ασύρματο δίκτυο αισθητήρων, καθώς και την διατήρηση της συνεκτικότητας του δικτύου.

Το *power management plane* διαχειρίζεται τον τρόπο με τον οποίο ένας κόμβος εκμεταλλεύεται τα αποθέματα ενέργειάς του. Π.χ ένας κόμβος μπορεί να κλείνει το δέκτη του για ένα διαστήμα αφού λάβει ένα μήνυμα, προκειμένου να μην το λάβει δύο φορές. Το *mobility management plane* παρακολουθεί την ύπαρξη γειτόνων στο δίκτυο ανά πάσα στιγμή, έτσι ώστε να υπάρχουν συνδέσεις μέσω των οποίων θα προωθηθεί πληροφορία προς το κέντρο ελέγχου. Γνωρίζοντας ποιοι είναι οι γείτονες ενός κόμβου μπορεί να κατανεμηθεί κι ο φόρτος εργασίας συνολικά στο δίκτυο. Δεν χρειάζεται να είναι ενεργοί όλοι οι κόμβοι που βρίσκονται σε μια μικρή περιοχή και να παίρνουν δείγματα ή να προωθούν μηνύματα συνεχώς. Το *task management plane* εξισορροπεί το φόρτο εργασίας και καθορίζει ποιος κόμβος είναι ενεργός ανά πάσα στιγμή. Π.χ οι κόμβοι που έχουν περισσότερη ενέργεια, αναλαμβάνουν περισσότερο ενεργό



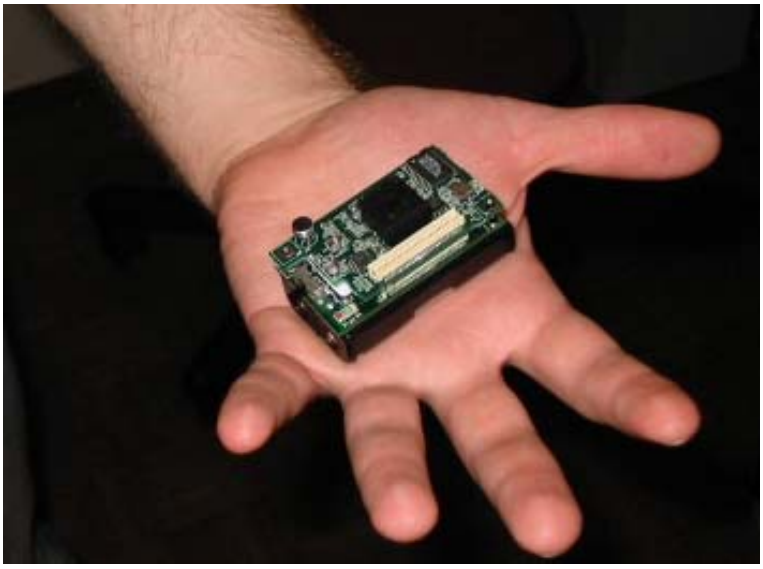

Σχήμα 1.2: Τα επίπεδα δικτύου σε ένα ασύρματο δίκτυο αισθητήρων

ρόλο από άλλους. Αυτά τα τρία επίπεδα χρειάζονται για τη σωστή συνεργασία μεταξύ των κόμβων και το διαμοιρασμό των πόρων, έτσι ώστε να επιμηκυνθεί η σωστή λειτουργία και η διάρκεια ζωής του δικτύου συνολικά.

## 1.5 Ιστορική αναδρομή στην εξέλιξη ως σήμερα των ασύρματων δικτύων αισθητήρων

Τα ασύρματα δίκτυα αισθητήρων έχουν γίνει αποδεκτά με μεγάλο ενδιαφέρον από την ερευνητική κοινότητα και αποτελούν ένα πεδίο όπου οι εξελίξεις τρέχουν. Η πρώτη πειραματική πλατφόρμα ασύρματων δικτύων αισθητήρων σχεδιάστηκε και υλοποιήθηκε στο Πανεπιστήμιο της Καλιφόρνια στο Berkeley το 1999. Έκτοτε, έχει ακολουθήσει ένα πλήθος από νέες προτάσεις για τους κόμβους αυτών των δικτύων:

- Cots Dust mote (1999)
- weC mote (1999)
- Rene mote (2000)
- Rene2 mote (2001)
- Dot mote (2001)
- Mica mote (2002)
- Mica2 mote (2003)
- Mica2Dot mote (2003)
- MicaZ mote(2004)



Σχήμα 1.3: Ένας κόμβος ασύρματου δικτύου αισθητήρων (MICA mote)

- Telos mote (2004)
- Tmote Sky mote (2005)

Από τις πλατφόρμες αυτές, οι πρώτες ήταν πειραματικές υλοποιήσεις του Πανεπιστημίου Berkeley, ενώ από το Mica mote και έπειτα κατασκευάζονται από την εταιρία Crossbow[11]. Η συγκεκριμένη εταιρία αποτελεί πλέον οδηγό στις εξελίξεις στα ασύρματα δίκτυα αισθητήρων, ενώ έχουν αρχίσει να εμφανίζονται και άλλες εταιρίες στο χώρο, όπως η Moteiv, της οποίας προϊόν είναι το Tmote Sky mote.

Τα motes λοιπόν είναι ουσιαστικά οι κόμβοι του ασύρματου δικτύου αισθητήρων και το μέγεθος τους είναι λίγο μεγαλύτερο από τις διαστάσεις δύο μπαταριών AA, από τις οποίες τροφοδοτούνται με ενέργεια. Στο σχήμα 1.3 φαίνεται ένα Mica mote. Όπως φαίνεται στο σχήμα, το μέγεθος της συσκευής είναι πολύ μικρό, παρ' όλα αυτά δεν έχει τις μικροσκοπικές διαστάσεις που οραματίζομαστε για τα ασύρματα δίκτυα αισθητήρων. Από την άλλη όμως δεν ήταν αυτός ο σκοπός των συσκευών αυτών, αλλά να δοθούν στην επιστημονική κοινότητα συσκεύες με τα οποία θα μπορεί να κάνει έρευνα στο πεδίο των ασύρματων δικτύων αισθητήρων, οι οποίες θα έχουν τα απαραίτητα για αυτά τα δίκτυα χαρακτηριστικά ενώ παράλληλα θα έχουν και ένα λογικό κόστος.

Κάθε τέτοιο mote συνοπτικά διαθέτει έναν επεξεργαστή, ένα πομποδέκτη, μία μικρή ποσότητα μνήμης, μετατροπείς αναλογικού σήματους σε ψηφιακό, δυνατότητες διασύνδεσης με sensor board και και μία μνήμη flash. Η μνήμη flash η οποία είναι διαθέσιμη, μπορεί να χρησιμεύσει ως χώρος αποθήκευσης

δειγμάτων από τις μετρήσεις που κάνει το mote. Τα sensor board είναι μικροσκοπικές κάρτες επέκτασης που φέρουν διάφορους αισθητήρες τους οποίους μπορεί να χρησιμοποιήσει το mote. Ανάμεσα στα φαινόμενα που μπορούμε να παρακολουθήσουμε με τους υπάρχοντες αισθητήρες, ανήκουν τα παρακάτω:

- Θερμοκρασία
- Υγρασία
- Ταχύτητα και κατεύθυνση ανέμου
- Πίεση
- Θόρυβος (ήχος γενικά)
- Ανίχνευση χημικών ουσιών

Η εταίρια Crossbow, παράλληλα με τα Mica mote, διάθετει στην αγορά ένα πλήθος από sensor board τα οποία μπορούν να συνεργαστούν με τα mote της. Οι δυνατότητες που προσφέρουν ποικίλλουν, ενώ τα πιο εξελιγμένα από αυτά διαθέτουν και GPS δυνατότητες. Για περισσότερες πληροφορίες, ο αναγνώστης μπορεί να ανατρέξει στο [11].

Αυτό που δίνει ζωή σε ένα ασύρματο δίκτυο αισθητήρων είναι το λειτουργικό σύστημα TinyOS [29], το οποίο σχεδιάστηκε επίσης στο Πανεπιστήμιο της Καλιφόρνια στο Berkeley, και του οποίου ο κώδικας είναι open source. Η επιτυχία που γνωρίζουν τα ασύρματα δίκτυα αισθητήρων οφείλεται σε μεγάλο βαθμό και στο TinyOS. Είναι το σύστημα που επιτρέπει τη λειτουργία των motes δίνοντάς μας μεγάλη ευελιξία και ευκολία στη δημιουργία εφαρμογών για ασύρματα δίκτυα αισθητήρων, παρά τις περιορισμένες δυνατότητές τους.

Αυτό όμως που έχει μεγαλύτερη σημασία είναι όχι τόσο οι υπηρεσίες τις οποίες μας προσφέρει το TinyOS, οι οποίες εκ των πραγμάτων δεν μπορούν να είναι πολλές, αλλά το γεγονός ότι η φιλοσοφία και ο σχεδιασμός του είναι εναρμονισμένα με το πλαίσιο στο οποίο καλούνται να λειτουργήσουν τα motes. Αυτό έχει ως αποτέλεσμα τη δημιουργία ενός περιβάλλοντος κομμένου και ραμμένου για χρήση σε τέτοια δίκτυα.

Η τελευταία έκδοση του TinyOS είναι η 1.1, με την έκδοση 2.0 να αναμένεται με μεγάλο ενδιαφέρον. Η έκδοση 1.0 ήταν μια μεγάλη βελτίωση σε σχέση με τις προηγούμενες και κυκλοφόρησε για την καλύτερη υποστήριξη των Mica2 mote, τα οποία κυκλοφόρησαν την ίδια περίπου εποχή. Η βασικότερη αλλαγή ήταν ότι ολόκληρο το σύστημα γράφτηκε ξανά στη γλώσσα nesC, τη γλώσσα προγραμματισμού η οποία χρησιμοποιείται έκτοτε σε συνδυασμό με το TinyOS για τον προγραμματισμό των συσκευών των ασύρματων δικτύων αισθητήρων. Για περισσότερες πληροφορίες σχετικά με το TinyOS και τη nesC, ο αναγνώστης μπορεί να ανατρέξει στα [1], [2] και [14].

## 1.6 Μελλοντικές προοπτικές των ασύρματων δικτύων αισθητήρων

Για να δώσουμε μια εικόνα των προοπτικών των ασύρματων δικτύων αισθητήρων θα επιχειρήσουμε μια σύγκριση μεταξύ της πορείας των ασύρματων δικτύων αισθητήρων μέχρι σήμερα και της πορείας του Διαδικτύου.

Το Διαδίκτυο γνώρισε εκρηκτική ανάπτυξη από τα μέσα της δεκαετίας του '90 και έπειτα, αλλά χρειάστηκε πρώτα να περάσουν πάνω από 20 χρόνια για να γίνει αυτή η επανάσταση. Όλη αυτή η πορεία μπορεί να χωριστεί σε 4 στάδια:

1. Ανάπτυξη της απαραίτητης τεχνολογίας (hardware, πρωτόκολλα δικτύου, λογισμικό)
2. Ανάπτυξη και λειτουργία των πρώτων πειραματικών δικτύων και των πρώτων εφαρμογών (ARPANET, e-mail).
3. Καθιέρωση κοινά αποδεκτών προτύπων (TCP/IP).
4. Εξάπλωση της χρήσης του Διαδικτύου παράλληλα με την αύξηση της εμπορικής του χρήσης (WWW).

Από τα παραπάνω μπορεί κάποιος να κάνει την αναγωγή στα ασύρματα δίκτυα αισθητήρων και να συμπεράνει πως σήμερα βρισκόμαστε κάπου ανάμεσα στη δεύτερη και την τρίτη φάση. Πλέον, έχει αρχίσει να ωριμάζει η τεχνολογία και να καθιερώνονται κοινά αποδεκτά πρότυπα, όπως θα δούμε στα επόμενα κεφάλαια της εργασίας αυτής, ενώ επίσης έχουν γίνει κάποιες πρώτες δοκιμές με ασύρματα δίκτυα αισθητήρων με μεγάλο πλήθος κόμβων.

Αφου λοιπόν συμβαίνουν όλα αυτά, έχει αρχίσει και η βιομηχανία να δείχνει μεγαλύτερο ενδιαφέρον για τα ασύρματα δίκτυα αισθητήρων. Αυτό γίνεται γιατί εκτός της ωριμότερης τεχνολόγίας και της καθιέρωσης προτύπων σε υλικό και λογισμικό, έχουν μειωθεί αισθητά οι τιμές των κόμβων των ασύρματων δικτύων αισθητήρων. Έτσι, ενώ πριν από 3 χρόνια ένας τέτοιος κόμβος κόστιζε γύρω στα 1000 δολάρια, σήμερα το κόστος του έχει μειωθεί στο ένα δέκατο, δηλαδή 100 δολάρια.

Έστω και σε αυτό το κόστος, τα ασύρματα δίκτυα αισθητήρων είναι μια οικονομικότερη επιλογή σε σχέση με τους παραδοσιακούς αισθητήρες που χρησιμοποιούνται σήμερα. Οι αισθητήρες αυτοί έχουν υψηλό κόστος εγκατάστασης και συντήρησης, το οποίο ξεπερνά το κόστος απόκτησης ενός mote για την ίδια εφαρμογή. Επιπλέον, το κόστος των ασύρματων δικτύων αισθητήρων ακολουθεί μία συνεχώς καθοδική πορεία. Ήδη, τα δίκτυα αυτά αποτελούν μια αγορά της τάξης των εκατομμυρίων δολαρίων και αναμένεται σύντομα να ξεπεράσουν τα 100 εκατομμύρια δολάρια το χρόνο.

## 1.7 Εφαρμογές δικτύων έξυπνης σκόνης

Στη συνέχεια παραθέτουμε κάποιες χαρακτηριστικές εφαρμογές στις οποίες μπορούν να χρησιμοποιηθούν τα ασύρματα δίκτυα αισθητήρων:

**Μικρο-γεωργία:** Με τη χρήση ενός δικτύου έξυπνης σκόνης, μπορούμε να παρακολουθούμε τις συνθήκες που επικρατούν στις καλλιέργειες σε πραγματικό χρόνο, ενώ παράλληλα μπορούμε να μετράμε και τις ποσότητες μικροβιοκτόνων που περνούν στο πόσιμο νερό, το επίπεδο της διάβρωσης του εδάφους, αν σμήνη εντόμων επιτίθενται στις καλλιέργειες και άλλα πολλά χρήσιμα.

**Διαχείριση αποθήκης:** η διαχείριση αποθήκης είναι από τις πιο σημαντικές πιθανές εφαρμογές των ασύρματων δικτύων αισθητήρων, και αφορα την . Η Hewlett-Packard δοκιμάζει ένα σύστημα με βιντεοκάμερες προσαρτημένες σε κάθε κόμβο για την οπτική αναγνώριση των προϊόντων που βρίσκονται στα ράφια μιας αποθήκης σε συνδυασμό με τη χρήση RFID, έτσι ώστε να γνωρίζουμε ποια προϊοντα είναι αποθηκευμένα και σε ποιο σημείο. Μπορούμε επίσης να γνωρίζουμε με αυτό τον τρόπο αν μετακινούνται προϊόντα μέσα στην αποθήκη, κτλ.

**Καταγραφή της βιοποικιλότητας:** Οι δορυφόροι που χρησιμοποιούνται για την παρατήρηση των δασικών εκτάσεων και του μεγέθους τους, δεν έχουν τη δυνατότητα να παρατηρήσουν τι γίνεται σε σχετικά μικρή κλίμακα. Με ένα ασύρματο δίκτυο αισθητήρων με δυνατότητες ανίχνευσης και αναγνώρισης αντικειμένων σε μικρή κλίμακα είναι δυνατή η καταγραφή της βιοποικιλότητας ακόμα και των πιο μικροσκοπικών ζώων, π.χ των εντόμων. Κάθε κινούμενο αντικείμενο παράγει κάποια ηλεκτρομαγνητική διαταραχή στην περιοχή που κινείται καθώς και ένα ελαφρύ σεισμικό αποτύπωμα, από τα οποία είναι δυνατό έως βαθμό να γίνει μια αναγνώριση του αντικειμένου. Π.χ ένας ελέφαντας όταν βαδίζει προκαλεί διαφορετική διαταραχή από μια καμηλοπάρδαλη.

**Ανίχνευση πυρκαγιάς:** Είναι δυνατόν να γίνει ρίψη κόμβων έξυπνης σκόνης από ένα αεροπλάνο ή ελικόπτερο ή και από ανθρώπους στο έδαφος, με κάποιο τυχαίο τρόπο σε μια δασική περιοχή. Δίνεται έτσι η δυνατότητα να ανιχνευθεί με ακρίβεια η τοποθεσία του μετώπου μιας πιθανής πυρκαγιάς σχεδόν αμέσως. Το πρόβλημα στην υλοποίηση ενός τέτοιου σεναρίου στη χώρα μας είναι ότι τα δάση μας είναι κυρίως πευκοδάση, που σημαίνει ότι οι πυρκαγιές παίρνουν πολύ γρήγορα μεγάλες διαστάσεις και επομένως είναι αμφισβητήσιμη η χρησιμότητα ενός τέτοιου δικτύου έξυπνης σκόνης. Επιπλέον, υπάρχει και το θέμα της ρύπανσης που προκαλούν οι κόμβοι του δικτύου όταν αχρηστευθούν, με τις μπαταρίες που φέρουν.

**Έλεγχος περιβάλλοντος σε κτίρια γραφείων:** Σήμερα ο κλιματισμός και ο εξαερισμός μεγάλων κτιρίων με γραφεία γίνεται συνήθως κεντρικά. Η θερμοκρασία μέσα στο ίδιο γραφείο μπορεί να έχει σημαντικές διακυμάνσεις και η κυκλοφορία του αέρα να μην γίνεται ικανοποιητικά. Τα κτίρια αυτά επιπλέον έχουν προβλήματα λόγω του μεγέθους και του σχεδιασμού τους, ειδικά οι ουρανοξύστες παρουσιάζουν πολλά λειτουργικά προβλήματα και απαιτούν να είναι ανοιχτά τα φώτα και να λειτουργεί ο κλιματισμός καθ΄ όλη τη διάρκεια της ημέρας. Ένα ασύρματο δίκτυο αισθητήρων θα μπορούσε να χρησιμοποιηθεί για τη σωστότερη μέτρηση των συνθηκών του περιβάλλοντος.

## 1.8 Παρουσίαση πραγματικών εφαρμογών των ασύρματων δικτύων αισθητήρων

Σε αυτήν την ενότητα θα δώσουμε δύο παραδείγματα πραγματικών ασύρματων δικτύων αισθητήρων, με στόχο να δείξουμε τη χρησιμότητα ενός τέτοιου δικτύου σε ρεαλιστική εφαρμογή, καθώς και τη γενικότερη λογική που διέπει ένα τέτοιο σύστημα, χώρις να προχωρήσουμε στην αναφορά πολλων τεχνικών λεπτομερειών. Τα παραδείγματα τα οποία θα δούμε είναι τα εξής:

1. Η εφαρμογή παρακολουθήσης κλιματικών και μικροκλιματικών συνθηκών στο νησί Great Duck (University of California, Berkeley).

2. Το σύστημα ESS (*Extensible Sensing System*) στο San Jacinto της Καλιφόρνια (University of California, Los Angeles).

### 1.8.1 Παρακολούθηση κλιματικών και μικροκλιματικών συνθηκών στο νησί Great Duck

Το πρώτο παράδειγμα εφαρμογής ασύρματου δικτύου αισθητήρων αφορά στη μελέτη του πληθυσμού ενός είδους θαλασσοπουλιών και της κατανομής του πληθυσμού αυτού, στο νησί Great Duck στο Maine [26],[25]. Στη συγκεκριμένη περίπτωση ο στόχος ήταν η μέτρηση των συνθηκών μέσα στις φωλιές των πουλιών αυτών, καθώς και η συγκέντρωση στοιχείων για τη γεωγραφική κατανομή του πληθυσμού. Για το σκοπό αυτό επιλέχθηκε η χρήση ενός ασύρματου δικτύου αισθητήρων, στο οποίο υπήρχαν δύο τύποι κόμβων:

1. κόμβοι μέσα στις φωλιές για να μετρούν τις συνθήκες που επικρατούν σε αυτές,

2. κόμβοι στην επιφάνεια του εδάφους του νησιού για να μετρούν τις συνθήκες του εξωτερικού περιβάλλοντος.

Οι κόμβοι του δικτύου έφεραν τους εξής αισθητήρες:

1. αισθητήρες υπέρυθρων ακτίνων (passive infrared sensors): οι αισθητήρες αυτοί μπορούν να ανιχνεύσουν τη θερμοκρασία που εκπέμπει το σώμα των πουλιών, και επομένως, τοποθετώντας τους μέσα σε μια φωλιά μπορούμε να δούμε αν κάποιο πουλί βρίσκεται σε αυτή ή όχι.

2. αισθητήρες υγρασίας και θερμοκρασίας: μπορούμε με τους αισθητήρες αυτούς να παρακολουθήσουμε τις συνθήκες που που επικρατούν μέσα στις φωλιές και πώς αυτές εξελίσσονται στην πορεία του χρόνου.

Οι κόμβοι που χρησιμοποιήθηκαν ήταν τύπου Mica2Dot της εταιρίας Crossbow. Ο συγκεκριμένος τύπος κόμβων προτιμηθήκε λόγω του μικρού του μεγέθους, σε σχέση με τους υπόλοιπους τύπους. Χρησιμοποιηθήκαν συσκευασίες

από πλαστικό για την προφύλαξη των κόμβων από τις περιβαλλοντικές συνθήκες (υγρασία κτλ).

Η αρχιτεκτονική του συστήματος περιλαμβάνει διάφορα επίπεδα. Στο κατώτερο επίπεδο συναντάμε κάποιες ομάδες από κόμβους, οι οποίοι κάνουν τις μετρήσεις που μας ενδιαφέρουν. Κάθε τέτοια ομάδα κόμβων επικοινωνεί με ένα gateway, το οποίο με τη σειρά επικοινωνεί με το κέντρο ελέγχου το οποίο συγκεντρώνει τις μετρήσεις από όλο το δίκτυο. Το κέντρο ελέγχου με τη σειρά του επικοινωνεί του Διαδικτύου με τον υπόλοιπο κόσμο, και τις εφαρμογές που αναλαμβάνουν να πάρουν τις μετρήσεις από το κέντρο ελέγχου και να τις επεξεργαστούν ή να στείλουν εντολές διαχείρισης στο ασύρματο δίκτυο αισθητήρων.

Παράλληλα με το ασύρματο δίκτυο αισθητήρων υπήρχαν και ένα δίκτυο επαλήθευσης (verification network) το οποίο περιελάμβανε συμβατικούς αισθητήρες που μάζευαν μετρήσεις από το περιβάλλον. Οι μετρήσεις αυτές χρησίμευαν στη σωστή βαθμονόμηση (calibration) των αισθητήρων των κόμβων του ασύρματου δικτύου αισθητήρων και στην επαλήθευση των μετρήσεων που έρχονταν μέσα από το δίκτυο.

Σχετικά με τη διάρκεια ζωής των κόμβων του δικτύου, χρησιμοποιώντας single-hop δρομολόγηση για την επικοινωνία των κόμβων με τα gateway, η μέση διάρκεια ζωής για τους κόμβους που βρίσκονται μέσα στις φωλιές ήταν 52 μέρες, ενώ για τους κόμβους που βρίσκονταν στην επιφάνεια του εδάφους ήταν 120 μέρες. Με τη χρήση multihop επικοινωνίας με τα gateway, οι αντίστοιχες τιμές ήταν 34 και 63 μέρες.

### 1.8.2 Το σύστημα ESS

Η δεύτερη εφαρμογή ασύρματων δικτύων αισθητήρων που παρουσιάζουμε είναι το σύστημα ESS (Extensible Sensing System) του UCLA. Και εδώ ο στόχος είναι η παρακολούθηση μικροκλιματικών συνθηκών, περιλαμβανομένης της θερμοκρασίας του περιβάλλοντος, της υγρασίας, της ταχύτητας και της διεύθυνσης του ανέμου.

Όσον αφορά την αρχιτεκτονική του συστήματος, οι κόμβοι που χρησιμοποιούνται στο σύστημα ανήκουν σε δύο εντελώς διαφορετικές κατηγορίες, από άποψης υπολογιστικών δυνατοτήτων και λειτουργίας. Στην πρώτη κατηγορία έχουμε τους κόμβους του δικτύου που κάνουν τις μετρήσεις του περιβάλλοντος και οι οποίοι είναι τύπου Mica2 της εταιρίας Crossbow. Στη δεύτερη κατηγορία έχουμε έναν αριθμό από κόμβους Stargate (βλ. κεφάλαιο 3) της Crossbow και πάλι, οι οποίοι τρέχουν λειτουργικό σύστημα Linux και συλλέγουν τις μετρήσεις που τους στέλνουν οι κόμβοι Mica2.

Οι κόμβοι της δεύτερης κατηγορίας μάλιστα έχουν και κάποιες δυνατότητες κίνησης για να μπορούν να μετακινούνται από περιοχή σε περιοχή και να συλλέγουν μετρήσεις από διαφορετικούς κόμβους. Με τη σειρά τους, στέλνουν τις μετρήσεις αυτές σε ένα κέντρο ελέγχου το οποίο είναι προσβάσιμο μέσω διαδικτύου από τον υπόλοιπο κόσμο.

Όσον αφορα το συνολικό πλήθος των κόμβων του δικτύου, ξεπερνά τους

100 και είναι από τις πρώτες εφαρμογές για ασύρματα δίκτυα αισθητήρων που το καταφέρνει αυτό. Η συνολική έκταση που καλύπτεται από το σύστημα φτάνει τα 25 εκτάρια (δηλαδή 250 στρέματα).

Επίσης, σε μία επέκταση του σύστηματος στη διάρκεια του φθινοπώρου του 2003, προκειμένου να μελετηθεί ο ρυθμός ανάπτυξης ενός συγκεκριμένου τύπου (πολύ ψηλού) δέντρου (συγκεκριμένα, του δέντρου California Redwood), προστέθηκαν 70 κόμβοι συνολικά, οι οποίοι τοποθετήθηκαν στα κλαδιά και τον κορμό κάποιου αντιπροσωπευτικού δέντρου.

## 1.9 Συνεισφορά και οργάνωση της εργασίας

Η συνεισφορά της παρούσας εργασίας συνοψίζεται σε δύο συνιστώσες:

1. Το πρωτόκολλο VTRP [6], το οποίο προσπαθεί να λύσει το πρόβλημα του εντοπισμού και διάδοσης πολλαπλών γεγονότων (*multiple event detection and propagation*) στα ασύρματα δίκτυα αισθητήρων. Το πρόβλημα αυτό συνίσταται στον εντοπισμό ενός πλήθους από γεγονότα μέσα στο δίκτυο και της διάδοσης των αντίστοιχων αναφορών σε ένα κέντρο ελέγχου, με έναν αποδοτικό τρόπο όσον αφορά στην κατανάλωση ενέργειας και στην ανοχή σε σφάλματα μετάδοσης. Το στοιχείο που διαφοροποιεί το VTRP σε σχέση με τα υπόλοιπα συναφή πρωτόκολλα είναι η μεταβολή της εμβέλειας μετάδοσης των κόμβων του ασύρματου δικτύου αισθητήρων, όταν αυτό επιβάλλεται από τις συνθήκες που επικρατούν στο δίκτυο.

2. Το περιβάλλον jWebDust [8], το οποίο έχει σαν στόχο να βοηθήσει τον τελικό χρήστη να υλοποιήσει μια εφαρμογή για ένα δίκτυο ασύρματων αισθητήρων εύκολα και γρήγορα. Το jWebDust παρέχει στο χρήστη ένα απλό interface μέσω του οποίου μπορεί να εκτελέσει τις απαραίτητες ενέργειες για τη διαχείριση ενός ασύρματου δικτύου αισθητήρων και να συλλέξει πληροφορίες από τους κόμβους του δικτύου αυτού.

Από πλευράς οργάνωσης, η εργασία περιλαμβάνει τα ακόλουθα μέρη:

**1. Έρευνα αιχμής στα ασύρματα δίκτυα αισθητήρων (κεφάλαια $1-3$):** Στο πρώτο μέρος κάνουμε μια εισαγωγή των βασικών εννοιών που συναντάμε στα ασύρματα δίκτυα αισθητήρων και μια μικρή αναδρομή στην πορεία τους μέχρι στιγμής. Ακολουθεί μια παρουσίαση της έρευνας αιχμής στα ασύρματα δίκτυα αισθητήρων, όσον αφορά τις σχετικές hardware πλάτφορμες και τις λύσεις που υπάρχουν σε λογισμικό ως τώρα.

**2. Παρουσίαση του πρωτοκόλλου VTRP (κεφάλαιο $4$):** παρουσιάζουμε αναλυτικά τη λειτουργία του VTRP και παραθέτουμε πειραματικά αποτελέσματα από εξομοιώσεις που κάναμε μέσω του εξομοιωτή simDust [1], [2].

**3. Αναλυτική παρουσίαση του συστήματος jWebDust (κεφάλαια $5-6$):** Παρουσιάζουμε συνολικά την αρχιτεκτονική του jWebDust και παρουσιάζουμε αναλυτικά τις προδιαγραφές λειτουργίας για κάθε επίπεδο του συστήματος.

# Κεφάλαιο 2

# Επιλεγμένη σχετική έρευνα στο λογισμικό για εφαρμογές σε ασύρματα δίκτυα αισθητήρων

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τις λύσεις σε λογισμικό που υπάρχουν αυτή τη στιγμή για τη διαχείριση και την ανάπτυξη εφαρμογών σε ένα ασύρματο δίκτυο αισθητήρων. Οι εφαρμογές αυτές έχουν ως στόχο να βοηθήσουν τον διαχειριστή ενός ασύρματου δικτύου αισθητήρων με πολλούς τρόπους, μερικοί από από τους οποίους είναι οι παρακάτω:

- Απλοποίηση της διαδικασίας και προσφορά εργαλείων για τη γρήγορη και εύκολη εγκατάσταση και την έναρξη της λειτουργίας ενός ασύρματου δικτύου αισθητήρων.
- Παροχή έτοιμων διαδικασιών για την ανάπτυξη μιας καινούριας εφαρμογής για ασύρματα δίκτυα αισθητήρων.
- Παροχή εργαλείων για την παρακολούθηση της κατάστασης και της σωστής λειτουργίας των συσκευών μέσα σε ένα τέτοιο δικτύο.
- Αποθήκευση των δεδομένων από το δίκτυο και δυνατότητα διασύνδεσης με ευρέως χρησιμοποιούμενα εργαλεία ανάλυσης δεδομένων.
- Δυνατότητα εκτίμησης της διάρκειας του χρόνου λειτουργίας που απομένει στο ασύρματο δίκτυο αισθητήρων (για κάθε κόμβο και συνολικά για το δίκτυο).

Λόγω του ότι τα ασύρματα δίκτυα αισθητήρων διανύουν ουσιαστικά τα πρώτα τους χρόνια, δεν υπάρχει μεγάλος αριθμός από έτοιμες λύσεις στον τομέα αυτό ακόμα. Επιπλέον, τα περιβάλλοντα που υπάρχουν αυτή τη στιγμή ακόμα δεν έχουν φτάσει σε ένα αρκετά ικανοποιητικό βαθμό ωριμότητας, αν και έχουν γίνει κάποια βήματα στη σωστή κατεύθυνση, καθώς η χρήση και η διαχείρισή

τους είναι αρκετά πολύπλοκες και απαιτούν αρκετές εξειδικευμένες γνώσεις πάνω στα ασύρματα δίκτυα αισθητήρων.

Οι υπάρχουσες λύσεις σε αυτό τον τομέα είναι οι εξής τέσσερις:

1. TinyDB (*University of California, Berkeley*)
2. TASK (*University of California, Berkeley και Intel*)
3. Moteview (*Crossbow*)
4. EmStar (*University of California, Los Angeles*)

Στη συνέχεια κάνουμε μια σύντομη παρουσίαση για κάθεμιά από τις πλατφόρμες αυτές, δίνοντας τα βασικά χαρακτηριστικά τους και αναλύοντας ως ένα βαθμό τη λειτουργία και τη φιλοσοφία τους. Ο αναγνώστης που επιθυμεί να μάθει περισσότερα για κάποια από αυτές τις πλατφόρμες, μπορεί να ανατρέξει στα [28], [27], [19].

## 2.1 Το περιβάλλον TinyDB

Το TinyDB είναι ένα περιβάλλον μέσω του οποίου είναι δυνατή η συγκέντρωση πληροφορίας από ένα ασύρματο δίκτυο αισθητήρων, μέσω του οποίου ο χρήστης βλέπει το δίκτυο αυτό με τρόπο που θυμίζει βάση δεδομένων. Το TinyDB παρέχει ένα SQL-like interface ώστε ο χρήστης να μπορεί να ορίζει με ευκολία και σαφήνεια την πληροφορία που τον ενδιαφέρει να πάρει από το δίκτυο, με άλλα λόγια τον τύπο του αισθητήρα (θερμοκρασία, φως, κτλ) μαζί με άλλα στοιχεία όπως π.χ. η περίοδος δειγματοληψίας.

Κύριος στόχος του TinyDB είναι να επιτρέψει στους χρήστες να εκμεταλλευτούν τις δυνατότητες που προσφέρει ένα ασύρματο δίκτυο αισθητήρων, χωρίς να χρειαστεί να γράψουν οι ίδιοι κώδικα για τα motes του δικτύου και να αναπτύξουν μια καινούρια εφαρμογή. Αυτό μειώνει σε πολύ μεγάλο βαθμό τον απαιτούμενο χρόνο και φόρτο εργασίας στην περίπτωση που θα έπρεπε να αναπτυχθεί μια εφαρμογή για ασύρματα δίκτυα αισθητήρων από την αρχή.

Το TinyDB βασίζεται πάνω στο TinyOS και επομένως, απαιτείται η εγκατάστασή του για να μπορεί ο χρήστης να χρησιμοποιήσει το TinyDB (η επίσημη διανομή του TinyOS περιέχει το TinyDB). Παράλληλα, ο χρήστης χρειάζεται να έχει εγκατεστημένο το περιβάλλον JDK της Java στο σύστημά του. Επομένως, με την εγκατάσταση του TinyOS, ο χρήστης μπορεί κατευθείαν να χρησιμοποιήσει το TinyDB.

Το TinyDB γενικά μπορεί να χωριστεί σε δύο υποσυστήματα:

1. Το λογισμικό που βρίσκεται στον υπολογιστή του χρήστη.
2. Το λογισμικό που τρέχουν οι κόμβοι του ασύρματου δικτύου αισθητήρων.

Στην πρώτη κατηγορία λογισμικού ανήκει το γραφικό περιβάλλον που παρέχει το TinyDB και το λογισμικό που είναι υπεύθυνο για την αποστολή των μηνυμάτων στο δίκτυο και τη συλλογή των απαντήσεων που έρχονται από αυτό. Το TinyDB από μόνο του δεν εμπεριέχει κάποια βάση δεδομένων για την αποθήκευση των μετρήσεων, οπότε χρειάζεται μια σχεσιακή βάση δεδομένων μαζί με το TinyDB για να μην χάνονται τα δεδομένα από το δίκτυο (βλ. και την επόμενη ενότητα, σχετικά με το περιβάλλον TASK).

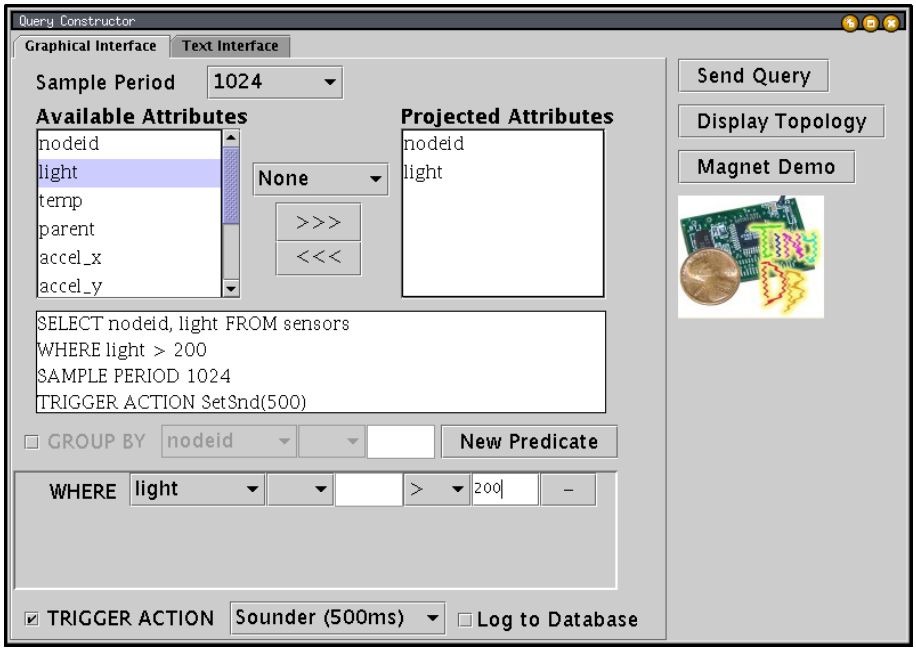
Στη δεύτερη κατηγορία λογισμικού ανήκει το λογισμικό που είναι αποθηκευμένο στη flash μνήμη των mote μέσα στο ασύρματο δίκτυο αισθητήρων και εκτελείται από αυτά. Σε γενικές γραμμές, ο χρήστης δε θα χρειαστεί να πειράξει καθόλου τον κώδικα για το επίπεδο αυτό, εκτός κι αν θέλει να επεκτείνει το σύστημα (π.χ. να προσθέσει κάποιο καινούριο χαρακτηριστικό ή να προσθέσει υποστήριξη για κάποιο καινούριο αισθητήρα). Τα βασικά μέρη του λογισμικού αυτού είναι τα εξής:

1. *Διαχειριστής καταλόγου χαρακτηριστικών:* τα χαρακτηριστικά του κάθε mote αποτυπώνονται στα components που αντιπροσωπεύουν το hardware που φέρει το mote (δηλαδή τύποι αισθητήρων) και στις εντολές για τη διαχείρισή τους (των software components). Μπορούμε να δούμε το σύνολο αυτών των χαρακτηριστικών και των εντολών σαν το schema μιας βάσης δεδομένων. Ο διαχειριστής του καταλόγου αυτού προσφέρει ένα interface στο υπόλοιπο λογισμικό του mote, ώστε να μπορεί να χρησιμοποιήσει τα components αυτά.

2. *Διαχειριστής Query:* ο διαχειριστής αυτός (*TupleRouter*) φροντίζει για το χειρισμό των νέων query που φθάνουν από το δίκτυο, για την εκτέλεσή τους και την απόστολή των αποτελεσμάτων από τις μετρήσεις που κάνει. Κάθε query φέρει ένα μοναδικό ID και βάση αυτού μπαίνει σε μια ουρά προς εκτέλεση (μπορούμε να έχουμε πολλαπλά queries εκτελούμενα ταυτόχρονα). Ο διαχειριστής καλεί έπειτα μια συνάρτηση για να υπολογίσει πότε θα είναι η επόμενη φορά που θα χρειαστεί να πάρει δείγματα από κάποιον αισθητήρα, με βάση τη συχνότητα δειγματοληψίας κάθε query. Αν εκτελείται μόνο ένα query αυτό θα γίνει στη συχνότητα δειγματοληψίας του query αυτού. Αν υπάρχουν παραπάνω από ένα query, αυτό θα γίνει στο μέγιστο κοινό διαιρέτη των περιόδων δειγματοληψίας των διαφόρων query.

3. *Διαχειριστής μνήμης:* αυτός ο διαχειριστής (*TinyAlloc*) είναι ένας απλός διαχειριστής μνήμης, ο οποίος κατανέμει byte από ένα frame μνήμης σταθέρου μεγέθους και επιστρέφει δείκτες προς περιοχές μνήμης του frame αυτού.

4. *Διαχειριστής δρομολόγησης:* ο διαχειριστής αυτός βασίζεται σε ένα multihop αλγόριθμο δρομολόγησης, και "πάνω" από αυτόν προσφέρει ένα interface με μια σειρά από εντολές, όπως αποστολή query μηνύματος, αποστολή μηνύματος μετρήσεων, κτλ.




Σχήμα 2.1: Το γραφικό περιβάλλον του TinyDB.

### TinySQL και queries στο TinyDB

Το TinyDB παρέχει μια γλώσσα για τον ορισμό των queries που τρέχουν στο ασύρματο δίκτυο αισθητήρων, η οποία μοιάζει με τη γλώσσα SQL, η οποία χρησιμοποιείται στις σχεσιακές βάσεις δεδομένων. Η γλώσσα αυτή ονομάζεται TinySQL. Ο χρήστης μπορεί να υποβάλλει queries στο δίκτυο μέσω του γραφικού περιβάλλοντος εκφρασμένα στην TinySQL. Τα queries στην TinySQL αποτελούνται από ένα σύνολο από attributes προς επιλογή (π.χ. ID κόμβου, θερμοκρασία, φως, κ.α.), ένα σύνολο από συνθήκες για την επιλογή των κόμβων που θα απαντήσουν στα queries (π.χ. η θερμοκρασία να είναι πάνω από κάποιο ορισμένο επίπεδο) και προαιρετικά κανόνες για data aggregation. Να σημειώσουμε ότι μπορούν να "τρέχουν" πολλαπλά queries σε κάθε κόμβο του δικτύου. Το γραφικό περιβάλλον του TinyDB μέσω του οποίου μπορούμε να στείλουμε queries στο δίκτυο, φαίνεται στο σχήμα 2.1.

Τα queries με τη σειρά τους μεταδίδονται σε ολόκληρο το δίκτυο, μέσω ενός flooding αλγορίθμου, και κάθε mote εξετάζει αν ένα query που θα λάβει προορίζεται για αυτό και ανάλογα αρχίζει να το εκτελεί ή το αγνοεί. Μετά από κάποιο χρονικό διάστημα, τα motes αρχίζουν και στέλνουν απαντήσεις στα queries του χρήστη πίσω στο κέντρο ελέγχου. Η δρομολόγηση των πακέτων με τις απαντήσεις αυτές γίνεται μέσω ενός multihop αλγόριθμου δρομολόγησης, ο οποίος δημιουργεί ένα δέντρο δρομολόγησης (με ρίζα το κέντρο ελέγχου, αφού είναι ο μοναδικός προορισμός για τα πακέτα).

Οι απαντήσεις στα queries που στέλνει ο χρήστης μέσω του graphical inter-

face του TinyDB στην ουσία αποθηκεύονται σε ένα μοναδικό πίνακα, ο οποίος λέγεται *Sensors*. Αυτός ο πίνακας περιέχει μια στήλη για κάθε attribute που μπορεί να υπάρχει στο ασύρματο δίκτυο αισθητήρων, δηλαδή node ID, τύποι αισθητήρων, κτλ., ή χαρακτηριστικά για την εσωτερική κατάσταση των motes, π.χ. timestamp, πατέρας στο δένδρο δρομολόγησης, κ.α. Στον server όλα αυτά τα πιθανά χαρακτηριστικά καθορίζονται σε ένα αρχείο XML.

**Σύνοψη - Ιδιαίτερα χαρακτηριστικά του TinyDB**

**Multihop δρομολόγηση μέσα στο δίκτυο:** τα μηνύματα μέσα σε ένα ασύρματο δίκτυο αισθητήρων που χρησιμοποιεί το TinyDB δρομολογούνται μέσω ενός multihop αλγορίθμου, ο οποίος κατασκευάζει ένα δέντρο δρομολόγησης με ρίζα το κέντρο ελέγχου του δικτύου, αφού αυτός είναι ο μοναδικός προορισμός για όλα τα μηνύματα που έρχονται από το δίκτυο. Για τα μηνύματα που στέλνει το κέντρο ελέγχου προς το δίκτυο, χρησιμοποιείται ένας αλγόριθμος flooding για τη διάδοσή τους μέσα στο δίκτυο.

**Δυνατότητα χρησιμοποίησης Triggers:** ένα ενδιαφέρον χαρακτηριστικό που περιλαμβάνεται στο TinyDB και στην TinySQL είναι τα *triggers*, δηλαδή queries τα οποία εκτελούν κάποια εντολή όταν συμβεί κάποιο ορισμένο από το χρήστη γεγονός. Παράδειγμα τέτοιου γεγονότος είναι το εξής: "όταν η θερμοκρασία του περιβάλλοντος ξεπεράσει τους 30 βαθμούς". Όταν συμβεί αυτό το γεγονός, υπάρχουν δύο δυνατότητες (προς το παρόν) που παρέχει το TinyDB: να αναβοσβήσουν τα LED του Mica mote ή να τεθεί σε λειτουργία το μεγαφωνάκι τους.

**Event-based queries:** ένα άλλο ενδιαφέρον χαρακτηριστικό του TinyDB είναι η δυνατότητα υποβολής *event based queries*. Η λογική εδώ είναι ότι περιγράφοντας κάποιο συγκεκριμένο γεγονός (event) με ένα component στο λογισμικό των motes, μπορούμε να καθορίσουμε πότε θα αρχίσει να εκτελείται κάποιο query.

**Καταγραφή αποτελεσμάτων στη flash μνήμη:** εκτός από τη δυνατότητα μετάδοσης των μετρήσεων μέσω ασύρματου δικτύου, τα αποτελέσματα από τις μετρήσεις μπορούν να αποθηκευτούν στη flash μνήμη του Mote. Έτσι, ο διαχειριστής του δικτύου μπορεί να επιλέξει να αποθηκεύονται με αυτόν τον τρόπο οι μετρήσεις, και όταν εξαντληθούν οι μπαταρίες του mote να συλλέξει τα mote από το χώρο που βρίσκονται και να ανακτήσει τις μετρήσεις από τη flash. Πρέπει να σημειωθεί όμως ότι η απαιτούμενη ενέργεια για αποθήκευση ενός bit στη flash μνήμη είναι λίγο μεγαλύτερη από αυτή που απαιτείται για τη μετάδοσή του πάνω μέσω ασύρματου δικτύου.

**Χρονικός συγχρονισμός και περίοδος δειγματοληψίας:** τα mote που βρίσκονται μέσα στο ασύρματο δίκτυο αισθητήρων είναι χρονικά συγχρονισμένα με έναν απλό μηχανισμό και επιπλέον εκμεταλλεύονται τον συγχρονισμό αυτό για την εξοικονόμηση ενέργειας. Αυτό επιτυγχάνεται με το να έχουν ανοικτούς τους πομποδέκτες τους και να κάνουν μετρήσεις ταυτόχρονα όλοι οι κόμβοι του δικτύου για ένα ορισμένο χρονικό διάστημα σε κάθε περίοδο δειγματοληψίας.

Το διάστημα αυτό είναι 4 δευτερόλεπτα (για queries με περίοδο μεγαλύτερη των 4 δευτερολέπτων) και καθορίζεται από το TinyDB (ο χρήστης μπορεί να το αλλάξει πριν προγραμματίσει τα motes αλλάζοντας την αντίστοιχη σταθερά). Αν π.χ. η περίοδος δειγματοληψίας είναι 30 δευτερόλεπτα, τα motes θα "ξυπνήσουν" ταυτόχρονα για 4 δευτερόλεπτα, θα κάνουν την αντίστοιχη μέτρηση και θα μεταδώσουν έπειτα τα αντίστοιχα αποτελέσματα από τις μετρήσεις τους. Το υπόλοιπο χρονικό διάστημα θα κλείσουν τους πομποδέκτες και τους αισθητήρες τους.

**Προσφερόμενο Java API:** Το TinyDB προσφέρει ένα Java API για τους προγραμματιστές που θέλουν να δημιουργήσουν τη δική τους εφαρμογή, μέσω της οποίας να στέλνουν TinySQL queries και να επεξεργάζονται τα αντίστοιχα αποτελέσματα από το ασύρματο δίκτυο αισθητήρων. Το γραφικό περιβάλλον το οποίο παρέχεται με τη διανομή του TinyDB χρησιμοποιεί το συγκεκριμένο API.

## 2.2 Το περιβάλλον TASK

Το σύστημα **TASK** (Tiny Application Sensor Kit) είναι μια προσπάθεια δημιουργίας ενός περιβάλλοντος για τη διαχείριση ασύρματων δικτύων αισθητήρων. Το TASK είναι αποτέλεσμα της συνεργασίας μεταξύ της *Intel* (Intel Research Berkeley) και των πανεπιστήμιων *Berkeley* στην Καλιφόρνια και του *MIT*. Η διάθεση και χρήση του TASK είναι δωρεάν, σύμφωνα με την άδεια χρήσης που περιέχεται στο αντίστοιχο διαθέσιμο πακέτο λογισμικού. Το TASK βασίζεται σε μεγάλο βαθμό στο TinyDB, παρέχοντας επιπρόσθετα κάποια εργαλεία για να κάνει πιο εύκολη τη δουλειά του διαχειριστή ενός ασύρματου δικτύου αισθητήρων.

Αν και η ανάπτυξη του TASK είναι ακόμα σε ενδιάμεσο στάδιο, ήδη προσφέρει αυξημένη λειτουργικότητα στον τελικό χρήστη. Το γεγονός αυτό οφείλεται σε μεγάλο βαθμό στην ωριμότητα του TinyDB, το οποίο είναι εξειδικευμένο λογισμικό για ασύρματα δίκτυα αισθητήρων και το οποίο έχει δοκιμαστεί σχεδόν από όλες τις ερευνητικές ομάδες που ασχολούνται με το συγκεκριμένο αντικείμενο.

Η ανάπτυξη ενός ασύρματου δικτύου αισθητήρων, έστω και με τη χρήση κάποιων έτοιμων εφαρμογών, δεν είναι μια εύκολη υπόθεση. Τα εργαλεία που υπήρχαν μέχρι πολύ πρόσφατα, απαιτούσαν εξειδικευμένους προγραμματιστές-διαχειριστές για την εγκατάσταση του λογισμικού και των συσκευών του δικτύου, καθώς και τη ρύθμιση της επικοινωνίας μεταξύ λογισμικού και του φυσικού δικτύου. Επίσης, ακόμα και μετά την επιτυχή έναρξη της λειτουργίας του σύστήματος, λόγω της φύσης των ασύρματων δικτύων αισθητήρων εμφανίζονται προβλήματα τα οποία επιβάλλουν την επίβλεψη του δικτύου.

Οι πιο σημαντικοί στόχοι που έθεσαν οι σχεδιαστές του TASK, από την άποψη του τελικού χρήστη, είναι οι εξής:

- Ευκολία στην εγκατάσταση του λογισμικού.

- Προσφορά ενός πλήθους από εργαλεία για την εγκατάσταση και την έναρξη της λειτουργίας του συστήματος.
- Παροχή ενός πλήθους από εργαλεία για την παρακολούθηση της κατάστασης και της σωστής λειτουργίας του δικτύου.
- Μετάφραση των μετρήσεων από το δίκτυο σε μονάδες οι οποίες είναι εύκολα κατανοητές από τον τελικό χρήστη (π.χ. βαθμοί Κελσίου για τις μετρήσεις θερμοκρασίας).
- Λειτουργία του δικτύου κάτω από αντίξοες συνθήκες.
- Αποθήκευση των δεδομένων από το δίκτυο και δυνατότητα διασύνδεσης με ευρέως χρησιμοποιούμενα εργαλεία ανάλυσης δεδομένων (π.χ. Excel) ή interface (π.χ. ODBC ή JDBC).
- Δυνατότητα εκτίμησης της διάρκειας του χρόνου λειτουργίας που απομένει στο ασύρματο δίκτυο αισθητήρων (για κάθε κόμβο και συνολικά για το δίκτυο).

Από την άλλη, εκτός από τον τελικό χρήστη, υπάρχουν και οι προγραμματιστές, οι οποίοι αναλαμβάνουν να κάνουν ορισμένες αλλαγές ή προσθήκες στο σύστημα, προκειμένου να το παραμετροποιήσουν για κάποια συγκεκριμένη εφαρμογή (π.χ. παρακολούθηση συνθηκών σε βιομηχανικές εγκαταστάσεις). Από τη σκοπιά αυτή υπάρχουν οι ακόλουθες ανάγκες:

1. Χρήση ενός συγκεκριμένου application interface για την προσθήκη ή αλλαγή εργαλείων οπτικοποίησης του δικτύου χωρίς να χρειάζεται να αλλάξει το ήδη υπάρχον λογισμικό του συστήματος.
2. Δυνατότητα προσθήκης νέων χαρακτηριστικών στο λογισμικό που τρέχουν οι συσκευές στο ασύρματο δίκτυο αισθητήρων (π.χ. νέοι τύποι αισθητήρων).
3. Κατ' επέκταση, δυνατότητα αλλαγής των υπηρεσιών του δικτύου (π.χ. δρομολόγηση ή παροχή στοιχείων για τη λειτουργία του δικτύου (Health Monitoring)).

**Αρχιτεκτονική και χαρακτηριστικά του TASK**

Το TASK έχει υλοποιηθεί σαν ένα σύνολο από επίπεδα που επικοινωνούν μεταξύ τους με κάποιο συγκεκριμένο interface. Υπάρχει το επίπεδο στο οποίο βρίσκεται ο TASK Server με τον οποίο επικοινωνεί το επίπεδο του client και των GUI εφαρμογών για να έχουν πρόσβαση στα δεδομένα που έρχονται από το ασύρματο δίκτυο αισθητήρων. Ο TASK server επικοινωνεί με το επίπεδο του ασύρματου δικτύου αισθητήρων μέσω ενός απλού και σχετικά περιορισμένου interface, το οποίο επιτρέπει τη χρήση διαφόρων μηχανισμών συλλογής

δεδομένων από το δίκτυο. Το επίπεδο αυτό έχει υλοποιηθεί με τρόπο που να επιτρέπει την εύκολη αλλαγή στη χρησιμοποιούμενη λογική του δικτύου, π.χ. αλγόριθμους δρομολόγησης, χρήση αλγορίθμων για περιορισμό κατανάλωσης ενέργειας, κτλ.

Εφόσον ήδη υπάρχει ένας αριθμός από διαθέσιμα εργαλεία για τα ασύρματα δίκτυα αισθητήρων, οι σχεδιαστές του TASK πήραν την απόφαση να εκμεταλλευτούν το λογισμικό αυτό. Η σχεδίαση του συστήματος σε επίπεδα, ουσιαστικά εξυπηρετεί το μοντέλο αυτό και η συνολική προσπάθεια επικεντρώθηκε στο να ενοποιηθούν οι εφαρμογές που αντιστοιχούν σε καθένα από τα επίπεδα αυτά και στην βελτίωση επιμέρους χαρακτηριστικών τους.

Γνώμονας στην υλοποίηση της αρχιτεκτονικής του συστήματος ήταν η δυνατότητα παροχής remote management ευκολιών στον τελικό χρήστη καθ' όλη τη διάρκεια ζωής του ασύρματου δικτύου αισθητήρων. Οι κόμβοι τέτοιων δικτύων ειναι γενικά δύσκολο να ελεγχθούν για δυσλειτουργίες ένας προς έναν. Ακόμα, επειδή τα δίκτυα αυτά αναπτύσσονται σε αντίξοες συνθήκες (παρεμβολές, συνθήκες περιβάλλοντος, κτλ), πρέπει το σύστημα να επιδεικνύει αντοχή σε καταστάσεις που έχουμε αστοχία υλικού ή λογισμικού.

Γενικά, ένα ασύρματο δίκτυο αισθητήρων στο οποίο τρέχει το σύστημα TASK αποτελείται από τα εξής τμήματα:

1. Ένα σύνολο από κόμβους (motes).

2. Ένα *sensor network appliance*, το οποίο τρέχει τον TASK Server, μια σχεσιακή βάση δεδομένων (συγκεκριμένα την Postgres) και λειτουργεί ως gateway μεταξύ του δικτύου και του υπόλοιπου κόσμου.

3. Ένα σύνολο από εργαλεία (*field tools*) τα οποία χρησιμεύουν στην παρακολούθηση των συνθηκών μέσα στο ασύρματο δίκτυο αισθητήρων και τα οποία τρέχουν σε PDA.

4. Ένα σύνολο από *εργαλεία client* για την οπτικοποίηση της κατάστασης μέσα στο δίκτυο (π.χ. δέντρο δρομολόγησης μέσα στο δίκτυο) τα οποία επικοινωνούν με το *sensor network appliance* για να πάρουν την αντίστοιχη πληροφορία.

**Λογισμικό των motes**

Οι τύποι των Motes που υποστηρίζονται από το TASK είναι mica2 και mica2dot. Τα motes αυτά τρέχουν το *TinyDB* (βλ. προηγούμενο κεφάλαιο) με μερικές προσθήκες και βελτιώσεις. Οι σχεδιαστές του *TASK* "πείραξαν" το *TinyDB* στους παρακάτω τομείς:

- εξοικονόμηση ενέργειας - κύκλος λειτουργίας,
- χρονικός συγχρονισμός (time synchronization),
- μηχανισμός query sharing,

- προσθήκη watchdog μετρητή για την ανίχνευση σφαλμάτων λογισμικού,
- καταγραφή μετρήσεων στη μνήμη flash των motes.

Όσον αφορά στη διαχείριση ενέργειας, το TinyDB δεν χρησιμοποιούσε κάποια ιδιαίτερη λογική για τη μείωση της κατανάλωσης ενέργειας στους κόμβους του ασύρματου δικτύου αισθητήρων, πριν ενσωματωθεί στο TASK. Αντίθετα, οι κόμβοι ήταν συνεχώς σε πλήρη λειτουργία, περιορίζοντας κατά πολύ τη διάρκεια ζωής του δικτύου. Οι δύο λύσεις που δοκιμάστηκαν ήταν (και πέρασαν αργότερα στη διανομή του TinyDB):

1. Η συγχρονισμένη παύση λειτουργίας και αφύπνιση για ένα μικρό χρονικό διάστημα (μέσω του χρονικού συγχρονισμού) όλων των κόμβων (δηλαδή όλοι οι κόμβοι αφυπνίζονται ταυτόχρονα για το ίδιο χρονικό διάστημα).
2. Ο κάθε κόμβος δειγματοληπτεί περιοδικά με τον τρόπο που αναλύεται στην ενότητα για τα χαρακτηριστικά του TinyDB (2.1).

Ο *χρονικός συγχρονισμός* του δικτύου επιτυγχάνεται με έναν απλό μηχανισμό. Το TinyDB βάζει timestamp αρχικά σε κάθε πακέτο που φεύγει από το κέντρο ελέγχου προς το ασύρματο δίκτυο αισθητήρων. Όταν κάποιος κόμβος λάβει ένα πακέτο από τον πατέρα του στο δέντρο δρομολόγησης, ελέγχει το timestamp του πακέτου και ρυθμίζει ανάλογα το δικό του ρολόι. Αν μεταδώσει το πακέτο στο υπόλοιπο δίκτυο, βάζει το δικό του timestamp. Με αυτό τον τρόπο σταδιακά συγχρονίζεται χρονικά ολόκληρο το δίκτυο. Αν κάποιος κόμβος δεν λάβει πακέτο από τον πατέρα του για κάποια ορισμένη χρονική περίοδο, θα ανοίξει τον πομποδέκτη του για να λάβει κάποιο πακέτο και να συγχρονιστεί μαζί του. Αυτή η μέθοδος συγχρονισμού έχει ακρίβεια στην πράξη μερικών millisecond.

Το TinyDB χρησιμοποιεί ένα μηχανισμό *watchdog* για να εξασφαλιστεί η σωστή λειτουργία των motes και η ανοχή τους σε σφάλματα λογισμικού (*fault tolerance*). Αν για $k$ περιόδους δειγματοληψίας ο κάθε κόμβος δεν λάβει κάποιο μήνυμα από τους γείτονές του, γίνεται ένα software reset στον κόμβο. Αυτή η μέθοθος προφυλάσσει από σφάλματα στη radio stack του TinyOS. Όταν ξαναξεκινήσει κανονικά τη λειτουργία του ο κόμβος, θα πάρει τα query που τρέχουν στο δίκτυο μέσω του μηχανισμού *query sharing*.

Ο μηχανισμός query sharing χρησιμεύει για τη διάδοση των query μέσα στο ασύρματο δίκτυο αισθητήρων και επιτρέπει σε κόμβους που έχουν κάνει software reset (όπως αναφέραμε στην προηγούμενη παράγραφο) να επανακτήσουν τα query που "τρέχουν" στο δίκτυο. Έτσι, όταν ένας κόμβος του δικτύου "ακούσει" κάποιο γειτονικό του κόμβο να στέλνει ένα μήνυμα ως απάντηση σε κάποιο query, θα ελέγξει το query ID που περιέχεται σε αυτό το μήνυμα. Αν το ID δεν αντιστοιχεί σε κάποιο query από αυτά που έχει στη λίστα του, θα στείλει ένα μήνυμα στο γειτονικό κόμβο για να του αποσταλλεί το query που αντιστοιχεί στο συγκεκριμένο ID. Στον μηχανισμό αυτό προστίθενται κάποιοι περιορισμοί

για το πότε και πώς θα αποσταλλούν τα αντίστοιχα μηνύματα για να μην υπάξει υπερβολική κίνηση στο δίκτυο.

**Sensor Network Appliance (SNA)**

Το SNA είναι ουσιαστικά ένα gateway ανάμεσα στο ασύρματο δίκτυο αισθητήρων και των εφαρμογών-εργαλείων που χρησιμοποιεί ο τελικός χρήστης. Το SNA τρέχει τον TASK Server, μια σχεσιακή βάση δεδομένων (RDBMS) και έναν web server. Ιδανικά, το SNA είναι ένα Stargate board (βλ. αντίστοιχο κεφάλαιο), αλλά μπορεί να είναι οποιοδήποτε laptop ή PC διαθέτει ένα mote συνδεδεμένο σε αυτό και σύνδεση με τον υπόλοιπο κόσμο μέσω Internet (αν και η λύση του Stargate ενδέχεται να είναι προτιμότερη).

Ο TASK Server λειτουργεί ως interface για να υποβάλλουν οι clients queries και εντολές στο ασύρματο δίκτυο αισθητήρων, να λαμβάνουν πληροφορίες για την κατάσταση του δικτύου και μετρήσεις ως απάντηση στα queries αυτά. Ο TASK Server είναι υλοποιημένος στη γλώσσα προγραμματισμού Java. Όταν κάποιος client στείλει ένα query, ο TASK Server το αποθηκεύει στη βάση δεδομένων και το προωθεί στο ασύρματο δίκτυο αισθητήρων. Όταν έρχονται απαντήσεις από το δίκτυο, ο server τις αποθηκεύει στη βάση δεδομένων και ειδοποιεί τους client, οι οποίοι είναι τυχόν συνδεδεμένοι. Μπορούν να εκτελελούνται ταυτόχρονα πολλά queries στο δίκτυο. Οι μετρήσεις, επίσης, μπορούν να αποστέλλονται σε κάποια άλλη βάση δεδομένων (π.χ. σε ένα PC με μεγάλο αποθηκευτικό χώρο σε σχέση με τη μνήμη του SNA).

**Field και Client εργαλεία**

Το field tool είναι μια εφαρμογή η οποία τρέχει σε κάποιο PDA (όπως το Zaurus της Sharp) και χρησιμεύει για το debugging και την επίβλεψη ενός ασύρματου δικτύου αισθητήρων. Η ιδιαιτερότητα της εφαρμογής αυτής είναι ότι ο χρήστης χρησιμοποιεί την εφαρμογή αυτή *μέσα* στο πεδίο που βρίσκεται το δίκτυο, οπότε μπορεί να εντοπίσει τυχόν προβλήματα επί τόπου και σε ποιο σημείο αυτά βρίσκονται.

Η εφαρμογή αυτή μας δίνει τη δυνατότητα να κάνουμε "ping" σε μεμονωμένα motes, με άλλα λόγια να διαπιστώσουμε αν ένα mote βρίσκεται σε λειτουργία και ποια queries εκτελεί, ενώ μπορούμε να κάνουμε reset και σε οποιαδήποτε συσκευή. Όταν ο διαχειριστής του δικτύου χρησιμοποιεί το εργαλείο αυτό με ένα PDA, στέλνει περιοδικά σήματα στα motes που βρίσκονται γύρω του για να απαντήσουν, οπότε ο χρήστης δημιουργεί μια λίστα με τα Motes που λειτουργούν κανονικά μέσα στο δίκτυο.

Υπάρχουν επίσης και δύο client εφαρμογές, μία *Web-based* εφαρμογή και μία εφαρμογή που ονομάζεται *TASKView*. Η πρώτη εφαρμογή επιτρέπει την εκτέλεση κάποιων βασικών διαχειριστικών ενεργειών μέσω του TASK Server, δηλαδή συνδέεται στον Web Server που βρίσκεται στο SNA και μέσω αυτού επικοινωνεί με τον TASK Server. Το TASKView προσφέρει προσφέρει περισσότερες λειτουργίες σε ένα δικό του γραφικό περιβάλλον, όπως αναπαράσταση

της τοπολογίας του δικτύου, εξαγωγή γραφικών παραστάσεων από τις μετρήσεις του δικτύου, κ.α.

## 2.3 Το σύστημα Moteview

Το Moteview αποτελεί μια προσπάθεια της εταιρίας Crossbow, δηλαδή της εταιρίας που κατασκευάζει τα Mica mote, για τη δημιουργία μιας εφαρμογής που λειτουργεί ως interface μεταξύ του τελικού χρήστη και ενός ασύρματου δικτύου αισθητήρων. Το Moteview προσφέρει στους τελικούς του χρήστες αρκετές δυνατότητες και εργαλεία για να απλοποιήσει τη διαδικασία εγκατάστασης και επίβλεψης ενός τέτοιου δικτύου.

Το Moteview υποστηρίζει τις πλατφόρμες mote της Crossbow από το Mica2 και έπειτα (δηλαδή Mica2, Mica2dot και MicaZ). Δεν υποστηρίζεται η πρώτη γενιά των Mica mote. Όλα τα sensor board της Crossbow υποστηρίζονται από το σύστημα, ακόμα και τα πιο σύνθετα MTS400 και MTS420. Όσον αφορά στα programming board και γενικότερα στη διασύνδεση με το ασύρματο δίκτυο αισθητήρων, υποστηρίζονται τα programming board MIB510 και MIB600 της Crossbow και επίσης το Stargate board.

Όσον αφορά στις εφαρμογές που θα τρέχουν στους κόμβους του ασύρματου δικτύου αισθητήρων και με τις οποίες είναι συμβατό το Moteview, αυτές είναι σχετικά περιορισμένες, και περιλαμβάνονται στη διανομή του Moteview. Πιο συγκεκριμένα, είναι οι Surge Reliable, Surge Reliable dot και οι εφαρμογές XMesh.

**Surge Reliable και Surge Reliable dot:** αυτές οι δυο εφαρμογές σχεδιάστηκαν αρχικά για να συνεργάζονται με την εφαρμογή Surgeview. Υπάρχουν δύο διαθέσιμες εκδόσεις του Surge Reliable, εκ των οποίων η μία δεν χρησιμοποιεί καθόλου power management για να μειώσει την κατανάλωση ενέργειας στους κόμβους του δικτύου, ενώ η άλλη χρησιμοποιεί power management και χρονικό συγχρονισμό. Η εφαρμογή Surgeview προσφέρει αρκετές δυνατότητες αλλά δεν προσφέρει ένα ολοκληρωμένο περιβάλλον, όπως αυτό του Moteview.

**XMesh εφαρμογές:** Οι εφαρμογές αυτές χρησιμοποιούν το πρωτόκολλο XMesh της Crossbow μέσα στο ασύρματο δίκτυο αισθητήρων. Υπάρχει μια διαφορετική εκδοχή διαθέσιμη (εκτελέσιμο αρχείο) για όλους τους συνδυασμούς mote και sensor board της Crossbow που υποστηρίζει το Moteview. Το XMesh έχει τα εξής χαρακτηριστικά:

- Multihop δρομολόγηση μηνυμάτων από τους κόμβους του δίκτυου προς το κέντρο ελέγχου.
- Επιλογή επομενου hop σε κάθε κόμβο με βάση την ποιότητα των συνδέσεων με τους γειτονικούς κόμβους.
- Χρονικός συγχρονισμός όλων των κόμβων του δικτύου, με ακρίβεια 1 millisecond.

| Περίοδος δειγματοληψίας (λεπτά) | Μέση ένταση ρεύματος σε Mica2 και Mica2Dot (μAmps) |
|---|---|
| 1 | 677 |
| 2 | 315 |
| 3 | 196 |

Πίνακας 2.1: Μέση ένταση ρεύματος σε σχέση με την περίοδο δειγματοληψίας

- Low power listening, με άλλα λόγια οι κόμβοι ανοίγουν τους δέκτες τους 8 φορές το δευτερόλεπτο για να ακούσουν αν μεταδίδονται μηνύματα στο δίκτυο από τους γειτονικούς τους κόμβους.

- Πολύ χαμηλή κατανάλωση ενέργειας.

Η Crossbow ισχυρίζεται πως στα δίκτυα που τρέχουν τις XMesh εφαρμογές της, το τελικό ποσοστό παραλαβής από το κέντρο ελέγχου των μηνυμάτων που στέλνονται από τους κόμβους του δικτύου ξεπερνά το 90% (και μάλιστα χωρίς τη χρήση end-to-end επιβεβαιώσεων (acknowledgements), δηλαδή λογικής παρόμοιας με του TCP/IP).

Η περίοδος δειγματοληψίας ορίζεται by default να είναι 3 λεπτά. Ο χρήστης μπορεί να αλλάξει αυτή την περίοδο δειγματοληψίας στο χρονικό διάστημα που επιθυμεί. Ένας ενδεικτικός πίνακας για τη μέση ένταση ρεύματος σε σχέση με την περίοδο δειγματοληψίας σε ένα δίκτυο που χρησιμοποιεί το XMesh, είναι ο πίνακας 2.1.

Οι εφαρμογές αυτές περιέχονται στη διανομή του Moteview ως εκτελέσιμα αρχεία (δηλαδή όχι σε μορφή πηγαίου κώδικα). Ο πηγαίος κώδικας για τις εφαρμογές είναι διαθέσιμος στο Διαδίκτυο και μπορεί οποιοσδήποτε να τον κατεβάσει και να τον χρησιμοποιήσει. Ο χρήστης μπορεί να προγραμματίσει τα mote του δικτύου του μέσω της εφαρμογής Moteconfig με τα αρχεία αυτά. Αυτό σημαίνει με λίγα λόγια ότι δεν χρειάζεται να υπάρχει εγκατεστημένο το TinyOS στο σύστημα του χρήστη για να προγραμματίσει τα mote (ούτε και για να λειτουργήσει το Moteview συνολικά), σε αντίθεση με τα υπόλοιπα συστήματα τα οποία περιγράφουμε στο κεφάλαιο αυτό. Αυτό μπορεί να ιδωθεί ως πλεονέκτημα, αφού ο χρήστης αποφεύγει όλη τη διαδικασία εγκατάστασης του TinyOS αφενός, και αφετέρου την πολυπλοκότητα του προγραμματισμου των motes του δικτύου μέσω της γραμμής εντολών, αφού το Moteconfig παρέχει ένα εύχρηστο γραφικό interface.

Τέλος, πρέπει να αναφέρουμε ότι το Moteview διατίθεται δωρέαν από το site της Crossbow αποκλειστικά για τα λειτουργικά συστήματα Windows XP και Windows 2000. Δεν υποστηρίζεται καθόλου το λειτουργικό σύστημα Linux, σε αντίθεση με τις υπόλοιπες πλατφόρμες που παρουσιάζονται σε αυτό το κεφάλαιο.

### Γενική αρχιτεκτονική του Moteview

Σε ένα ασύρματο δίκτυο αισθητήρων στο οποίο χρησιμοποιείται για τη διαχείρισή του το Moteview, διακρίνουμε τα παρακάτω τρία αρχιτεκτονικά επίπεδα:

1. *Επίπεδο mote* (mote layer): σε αυτό το επίπεδο βρίσκονται οι κόμβοι του ασύρματου δικτύου αισθητήρων, οι οποίοι τρέχουν το λογισμικό το οποίο είναι συμβατό με το Moteview.

2. *Επίπεδο εξυπηρετητή* (server layer): σε αυτό το επίπεδο βρίσκεται το λογισμικό που λειτουργεί ως gateway για το δίκτυο και το λογισμικό για την αποθήκευση των μετρήσεων από το δίκτυο (βάση δεδομένων).

3. *Επίπεδο client* (client layer): εδώ ανήκει ο client του Moteview, ο οποίος συνδέεται στη βάση δεδομένων του επιπέδου εξυπηρετητή για να παρέχει τις υπηρεσίες του στο χρήστη.

Η διανομή του Moteview αποτελείται από τα εξής μέρη:

- Το εκτελέσιμο αρχείο για το Moteview (επίπεδο client).
- Τη σχεσιακή βάση δεδομένων Postgres, στην οποία αποθηκεύουμε την πληροφορία από το δίκτυο (επίπεδο server). Η Postgres εγκαθίσταται μαζί με τον client του Moteview.
- Τα εκτελέσιμα αρχεία για τις εφαρμογές XListen, XServe (επίπεδο server) και τις εφαρμογές XMesh που είναι συμβατές με το Moteview (επίπεδο mote).

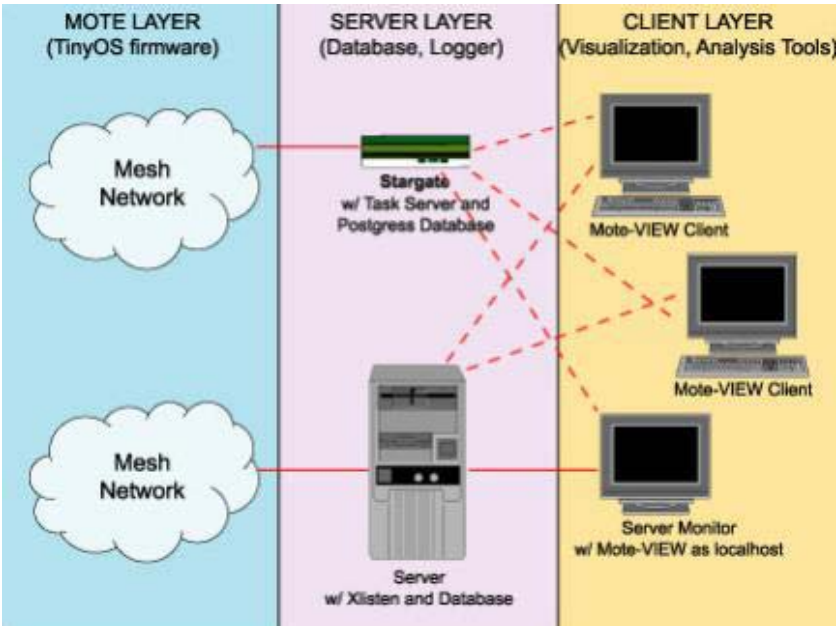
Το XListen είναι μια εφαρμογή που τρέχει στο Stargate ή στο PC που χρησιμεύει ως gateway μεταξύ του ασύρματου δικτύου αισθητήρων και του υπόλοιπου συστήματος. Έτσι λοιπόν, το XListen "ακούει" για πακέτα από το δίκτυο σε ένα συγκεκριμένο format, και όταν λάβει ένα τέτοιο πακέτο μεταφράζει την πληροφορία από τους αισθητήρες, η οποία είναι περιέχει μια αναλογική ή ψηφιακή μέτρηση, σε μονάδες κατανοητές από τον τελικό χρήστη. Από το XListen η πληροφορία μπορεί να προωθηθεί στο Moteview ή στη βάση δεδομένων του συστήματος (την Postgres).

Σύμφωνα με τα όσα αναφέραμε παραπάνω, υπάρχουν διαφορετικοί πιθανοί τρόποι λειτουργίας του Moteview. Στο σχήμα 2.2 φαίνονται οι διαφορετικές δυνατότητες που έχουμε στο στήσιμο ενός ασύρματου δικτύου αισθητήρων.

### Χαρακτηριστικά και δυνατότητες του Moteview

Το Moteview έχει ένα πλήθος από χρήσιμα χαρακτηριστικά και δίνει αρκετές δυνατότητες στην χρήστη, όσον αφορά τη διαχείριση και την επίβλεψη ενός ασύρματου δικτύου αισθητήρων. Από αυτά, τα πιο βασικά είναι τα εξής:

- Οπτικοποίηση μετρήσεων από το δίκτυο.



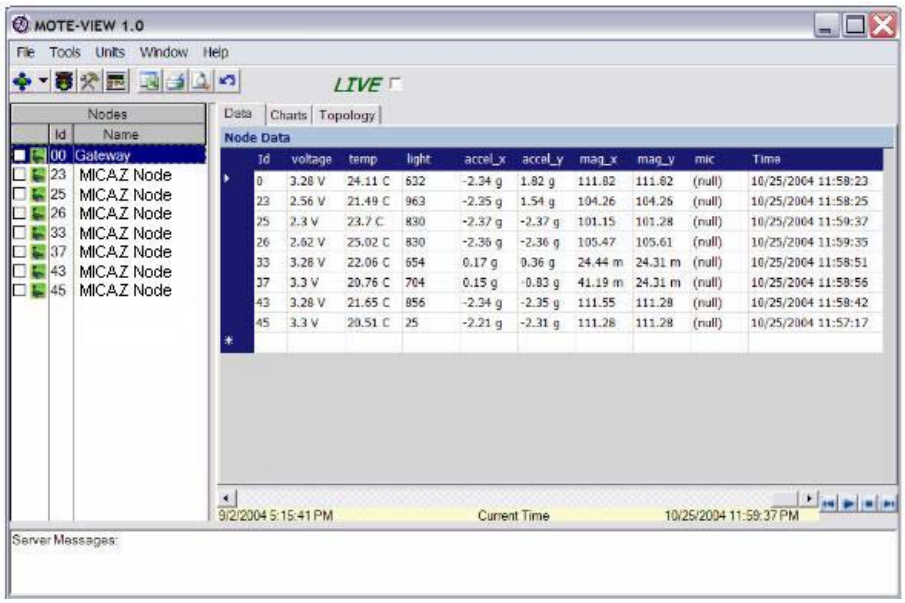

Σχήμα 2.2: Γενική άποψη της αρχιτεκτονικής του Moteview.

- Εξαγωγή μετρήσεων από το δίκτυο σε χρήσιμα format.
- Παρακολούθηση κατάστασης κόμβων (health monitoring).
- Πρόβλεψη χρόνου λειτουργίας που απομένει στο δικτύο.

Όσον αφορά την οπτικοποίηση των μετρήσεων από το δίκτυο (οι οποίες γίνονται μέσω των XMesh εφαρμογών που τρέχουν στα mote), έχουμε τις εξής δυνατότητες:

- Μετρήσεις σε μορφή απλού κειμένου σε πίνακα.
- Μετρήσεις σε μορφή γραφικής παράστασης σε σχέση με το χρόνο.
- Συνολική εικόνα του δικτύου, με τους κόμβους να απεικονίζονται ως κόμβοι ενός δένδρου του οποίου οι ακμές είναι οι διαδρομές που επιλέγει ο αλγόριθμος δρομολόγησης, με απεικονισμένες τις τελευταίες μετρήσεις του κάθε κόμβου.

Το Moteview μας δίνει τη δυνατότητα να επιλέγουμε μέσω του γραφικού του περιβάλλοντος πλήθος κόμβων και να απεικονίζουμε τις μετρήσεις από τους κόμβους αυτούς ταυτόχρονα. Παρέχει μια λίστα με όλους τους κόμβους που βρίσκονται στο δίκτυο, στους οποίους μπορούμε να δώσουμε ένα όνομα και να τους εντάξουμε σε κάποιο γκρουπ. Παράλληλα, οι μετρήσεις που παίρνουμε από το δίκτυο μπορούν να αποθηκευτούν και σε μορφή XML ή CSV (comma




Σχήμα 2.3: Το γραφικό περιβάλλον του Moteview.

separated text). Αυτό μας δίνει τη δυνατότητα να επεξεργαστούμε τις μετρήσεις και με άλλες εφαρμογές, εκτός του Moteview (όπως π.χ. το Excel).

Η λίστα με τους κόμβους του δικτύου παράλληλα μας πληροφορεί για την κατάσταση του κάθε κόμβου. Ανάλογα με το πόσος χρόνος έχει περάσει από τότε που ήρθε το τελευταίο μήνυμα από τον κάθε κόμβο του δικτύου, χρωματίζεται ο συγκεκριμένος κόμβος στη λίστα που βρίσκεται στο γραφικό περιβάλλον του Moteview. Έτσι,με πράσινο χρώμα απεικονίζονται οι κόμβοι από τους οποίους ήρθε κάποιο μήνυμα μέσα στα προηγούμενα 20 λεπτά, με κίτρινο στα τελευταία 40 λεπτά, πορτοκαλί μέσα στην τελευταία ώρα και με κόκκινο οι κόμβοι που έχουν να επικοινωνήσουν με το κέντρο ελέγχου πάνω από μια μέρα.

Ένα πολύ ενδιαφέρον χαρακτηριστικό του Moteview είναι ότι δίνει μια προβλεψη για το χρόνο λειτουργίας που απομένει σε κάθε κόμβο του δικτύου. Η πρόβλεψη αυτή παρακολουθώντας σε κάθε κόμβο την κατανάλωση ενέργειας (κάθε κόμβος περιοδικά στέλνει σχετικά μηνύματα με την πληροφορία αυτή).

Σαν ένα τελικό σχόλιο πάνω στο Moteview, θα λέγαμε ότι οι δυνατότητες που προσφέρει είναι αρκετές, αλλά υστερούν συνολικά σε σχέση με τις δυνατότητες που προσφέρουν οι υπόλοιπες πλατφόρμες. Έτσι π.χ., δεν υπάρχει η δυνατότητα για query στο δίκτυο όπως στο TinyDB, και γενικά διαθέτει περιορισμένη ευελιξία και επεκτασιμότητα. Παρ' όλα αυτά είναι μια ολοκληρωμένη πρόταση, η οποία δεν έχει ως προαπαιτούμενο την εγκατάσταση του TinyOS και διαθέτει το πιο φιλικό περιβάλλον από τις υπόλοιπες προτάσεις που παρουσιάζονται στο παρόν κεφάλαιο. Επίσης, ένα μέρος του λογισμικού, δηλαδή το λογισμικό που τρέχουν τα mote και το XListen, είναι διαθέσιμο δωρεάν και μπορεί οποιοσδήποτε να το κατεβάσει και να το χρησιμοποιήσει για να γράψει

τη δική του εφαρμογή.

## 2.4 Η πλατφόρμα EmStar (Em* )

Μια διαφορετική φιλοσοφία, σε σχέση με τα υπόλοιπα συστήματα για ασύρματα δίκτυα αισθητήρων που περιγράφονται σε αυτή την εργασία, ακολουθεί η πλατφόρμα EmStar [15],[16]. Το EmStar απευθύνεται, από πλευράς hardware, σε πλατφόρμες που μπορούν να τρέξουν το λειτουργικό σύστημα Linux, σε αντίθεση με τις υπόλοιπες προτάσεις που ο τελικός σκοπός τους είναι να υποστηρίξουν δίκτυα που τρέχουν το TinyOS. Ένας από τους βασικούς λόγους για αυτή τη διαφοροποίηση είναι ότι οι σχεδιαστές του EmStar είχαν ως στόχο τη δημιουργία ενός συστήματος για ετερογενή (heterogeneous) ασύρματα δίκτυα αισθητήρων.

Με τον όρο ετερογενή ασύρματα δίκτυα αισθητήρων, στο EmStar εννοούμε δίκτυα τα οποία αποτελούνται όχι μόνο από κόμβους με διαφορετικές δυνατότητες αίσθησης του περιβάλλοντος (διαφορετικούς αισθητήρες δηλαδή), αλλά δίκτυα που αποτελούνται από κόμβους που ανήκουν σε διαφορετικές κλάσεις, από άποψη επεξεργαστικών δυνατοτήτων. Το EmStar κάνει διάκριση σε δύο τέτοιες τάξεις:

1. Τα Mica motes που μπορούν να τρέξουν το TinyOS, και
2. τους *Microservers*, οι οποίοι τυπικά είναι PDAs (όπως το iPAQ ή το Zaurus) ή πλατφόρμες όπως το Stargate της Crossbow.

Το EmStar απευθύνεται στη δεύτερη κατηγορία κόμβων, δηλαδή τους microserver. Αυτό σημαίνει ότι σχεδιάστηκε για ασύρματα δίκτυα αισθητήρων στα οποία υπάρχουν και οι δύο τύποι κόμβων, για να εξυπηρετήσει τις ανάγκες που προκύπτουν για εργαλεία σε τέτοιου είδους εφαρμογές που να εκμεταλλεύονται τις δυνατότητες των microservers.

Παράδειγμα μιας τέτοιας εφαρμογής είναι το Extensible Sensing System (ESS) στο James Reserve στην Καλιφόρνια. Στη συγκεκριμένη εφαρμογή χρησιμοποιούνται 50 motes για τη συλλογή δεδομένων από το περιβάλλον και κάποιοι ρομποτικοί κόμβοι που έχουν δυνατότητα κίνησης (και βασίζονται στην πλατφόρμα Stargate) για τη συλλογή των δεδομένων από τα motes.

### Εργαλεία και υπηρεσίες που προσφέρει το EmStar

Το EmStar προσφέρει ένα πλήθος από εργαλεία και υπηρεσίες με σκοπό να διευκολύνει τον προγραμματιστή στην ανάπτυξη ενός ασύρματου δικτύου αισθητήρων. Τα εργαλεία αυτά αφορούν στην εξομοίωση (simulation), προσομοίωση (emulation) και οπτικοποίηση (visualization) τέτοιων δικτύων, ενώ οι υπηρεσίες που προσφέρει το EmStar έχουν σχέση με την επικοινωνία (σε φυσικό επίπεδο) στο δίκτυο, στην χρήση διάφορων αισθητήρων και στο χρονικό συγχρονισμό (time synchronization) των microservers.

**Εξομοίωση με το EmSim:** Το EmSim προσφέρει ένα περιβάλλον για εξομοίωση με μερικά πολύ χρήσιμα χαρακτηριστικά. Ένα από αυτά είναι το γεγονός ότι ο κώδικας που γράφουμε για τον εξομοιωτή είναι ο ίδιος που θα χρησιμοποιήσουμε και στην πραγματική εφαρμογή, όπως και στον TOSSIM, τον εξομοιωτή που υπάρχει για το περιβάλλον TinyOS/NesC. Μια διαφορά μεταξύ του TOSSIM και του EmSim είναι ότι κάθε εικονικός κόμβος στον EmSim αποτελεί μια ξεχωριστή διαδικασία (process) του συστήματος, στην οποία προφανώς έχουμε πολύ μεγαλύτερο έλεγχο και επίσης μπορούμε με αυτό τον τρόπο να ελέγξουμε καλυτερα την επικοινωνία μεταξύ των εικονικών κόμβων. Υπάρχει όμως το μειονέκτημα ότι λόγω των ξεχωριστών διεργασιών είναι δύσκολη η εξομοίωση μεγάλων δικτύων.

Ο κώδικας που προορίζεται για το EmStar μπορεί να είναι γραμμένος οποιαδήποτε γλώσσα προγραμματισμού μπορεί να χρησιμοποιήσει τις βιβλιοθήκες του EmStar (οι οποίες είναι γραμμένες σε C). Επίσης, μπορεί να χρησιμοποιηθεί κώδικας γραμμένος σε NesC. Υπάρχει διαθέσιμη μια βιβλιοθήκη (EmTOS) η οποία χρησιμεύει ακριβώς για αυτή την περίπτωση, αν και η NesC δεν είναι η καλύτερη επιλογή για τις εφαρμογές που προορίζονται για το περιβάλλον του Emstar.

**Οπτικοποίηση με το EmView:** Το EmView είναι μια εφαρμογή η οποία παρέχει μια γραφική αναπάρασταση ενός ασύρματου δικτύου αισθητήρων (πραγματικού ή σε εξομοίωση) του οποίου οι κόμβοι χρησιμοποιούν το EmStar. Σε κάθε κόμβο τρέχει ένας proxy server (ο οποίος έχει το πρωτότυπο όνομα EmProxy) και ο οποίος χειρίζεται την επικοινωνία μέσω μηνυμάτων UDP με τον κόμβο στον οποίο τρέχει το EmView.

**Ασύρματη επικοινωνία:** Εφόσον το EmStar προορίζεται να τρέξει σε πλατφόρμες όπως το Stargate, προσφέρει interface για ασύρματη επικοινωνία μέσω δικτύου IEEE 802.11b και μέσω Mica mote. Συγκεκριμένα, όσον αφορά στην επικοινωνία με τα Mica motes, στο EmStar υπάρχει κάτι αντίστοιχο με το SerialForwarder που υπάρχει στο TinyOS και το οποίο είναι μία εφαρμογή για επικοινωνία μέσω USB ή RS-232 θύρας από και προς ένα Mica mote, το οποίο είναι συνδεδεμένο στον Microserver. Το HostMote αντίστοιχα είναι μια εφαρμογή που χρησιμεύει ως Gateway μεταξύ του mote και των client εφαρμογών που θέλουν να επικοινωνήσουν μαζί του.

Επίσης, προσφέρονται υπηρεσίες καταγραφής γειτονικών κόμβων σε λίστα (δηλαδή ποιοι κόμβοι βρίσκονται μέσα στην εμβέλεια μετάδοσης του κόμβου) και εκτίμησης ποιότητας της επικοινωνίας με καθέναν από τους κόμβους αυτούς ( η πληροφορία αυτή χρησιμεύει στους αλγόριθμους δρομόλογησης για την επιλογή του επόμενου βήματος για μετάδοση προς κάποιον κόμβο ο οποίος δεν είναι γειτονικός).

**Χρονικός συγχρονισμός:** Ο χρονικός συγχρονισμός σε ένα ασύρματο δίκτυο αισθητήρων που τρέχει το EmStar επιτυγχάνεται μέσω μιας υλοποίησης του αλγορίθμου RBS [12], [13]. Εν συντομία, με τον συγκεκριμένο αλγόριθμο οι κόμβοι του δικτύου έχουν σχετικό χρονικό συγχρονισμό μεταξύ τους, και όχι

απόλυτο, δεν υπάρχει δηλαδή κάποιος κεντρικός κόμβος βάση του οποίου συγχρονίζονται όλοι οι κόμβοι του δικτύου. Αντίθετα, σε κάθε κόμβο γίνεται μια εκτίμηση για το πώς θα μετατραπεί ένα timestamp προερχόμενο από κάποιο γειτονικό κόμβο στη σχετική τιμή για τον συγκεκριμένο κόμβο.

# Κεφάλαιο 3

# Εξελίξεις στον τομέα του hardware και των προτύπων ασύρματης επικοινωνίας στα ασύρματα δίκτυα αισθητήρων

*"Μια καινούρια κλάση υπολογιστών εμφανίζεται ανά δεκαετία"*
Gordon Bell

*"Ο αριθμός των τρανζίστορ σε ένα chip διπλασιάζεται κάθε δύο χρόνια"*
Gordon Moore

Στην ενότητα αυτή παρουσιάζουμε τα πρότυπα 802.15.4 της IEEE [3] και ZigBee[30], τα οποία αποτελούν την πρόταση της IEEE για την επικοινωνία στα *ασύρματα PAN δίκτυα* (Wireless Personal Area Network), και τα οποία έχουν αρχίσει να κατακτούν έδαφος στο χώρο των ασύρματων δικτύων αισθητήρων, επομένως παρουσιάζουν ιδιαίτερο ενδιαφέρον. Παρουσιάζουμε συνοπτικά τα κύρια χαρακτηριστικά τους, ενώ παραθέτουμε και τα σχετικά πλεονεκτήματά τους.

Στη συνέχεια παρουσιάζουμε συνοπτικά δυο από τις hardware πλατφόρμες που χρησιμοποιούν τα πρότυπα αυτά μέσα σε ασύρματα δίκτυα αισθητήρων, την πλατφόρμα MicaZ της Crossbow και την πλατφόρμα Tmote Sky της Moteiv [18]. Οι πλατφόρμες αυτές παρουσιάζουν ιδιαίτερο ενδιαφέρον, καθώς είναι οι πρώτες που υλοποιούν το πρότυπο IEEE 802.15.4.

Τέλος, παρουσιάζουμε συνοπτικά την πλατφόρμα Spec [23], η οποία είναι μια προσπάθεια για τη δημιουργία ενός πραγματικού smart dust κόμβου, αφού το μέγεθός της είναι μικροσκοπικό. Ο αναγνώστης που ενδιαφέρεται να βρει

στοιχεία για παλαιότερες πλατφόρμες mote, όπως τα Mica και Mica2 motes, μπορεί να ανατρέξει στις εργασίες [2] και [1].

## 3.1 Τα πρότυπα IEEE 802.15.4 και ZigBee

Το πρότυπο 802.15.4 της IEEE παρουσιάστηκε πρόσφατα (2003) και περιγράφει το Physical και MAC layer ενός πρωτοκόλλου για ασύρματη επικοινωνία σε δίκτυα με μεγάλο αριθμό κόμβων, μεγάλους περιορισμούς στην κατανάλωση ενέργειας από τους κόμβους αυτούς, οι οποίοι τρέχουν εφαρμογές που δεν έχουν μεγάλες απαιτήσεις σε ταχύτητα μεταφοράς δεδομένων.

Ο όρος ZigBee αναφέρεται σε ένα κλειστό πρότυπο το οποίο ανέπτυξε ένα consortium εταιριών (όπως η Philips και η Motorola) και το οποίο χρησιμοποιεί το IEEE 802.15.4 μαζί με μια στοίβα πρωτοκόλλων για δρομολόγηση, κρυπτογράφηση, συγχρονισμό του δικτύου, ενώ παρέχει ένα API για τις εφαρμογές που τρέχουν στους κόμβους του δίκτυου.

Το πρότυπο ZigBee σχεδιάστηκε για να λειτουργεί συμπληρωματικά με τα άλλα πρότυπα για ασύρματη δικτύωση, όπως το 802.11a, b ή g και το Bluetooth. Αυτό γιατί καθένα από αυτά καλύπτει διαφορετικό πεδίο εφαρμογών και αναγκών. Το ZigBee προορίζεται για οικιακούς αυτοματισμούς, συστήματα ελέγχου κλιματισμού και εξαερισμού, ασύρματα δίκτυα αισθητήρων, και γενικά για εφαρμογές όπου είναι πολύ βασικό να υπάρχει η ελάχιστη δυνατή κατανάλωση ενέργειας. Το γεγονός αυτό, σε συνδυασμό με το ότι το κόστος των συσκευών σε ένα τέτοιο δίκτυο πρέπει να είναι πολύ μικρό, καθιστά ακατάλληλο για τις εφαρμογές για τις οποίες προορίζεται το ZigBee αφενός το Bluetooth (αφού δεν έχει σχεδιαστεί για μεγάλα δίκτυα και δεν εξασφαλίζει ελάχιστη κατανάλωση ενέργειας) και αφετέρου το 802.11 (λόγω του μεγάλου κόστους και της μεγάλης κατανάλωσης ενέργειας).

Στον πίνακα 3.1 φαίνονται τα βασικά χαρακτηριστικά των τριών προτύπων. Ακολουθούν συνοπτικές περιγραφές του IEEE 802.15.4 και του ZigBee.

| **Χαρακτηριστικό** | **802.11** | **Bluetooth** | **ZigBee** |
|---|---|---|---|
| Διάρκεια ζώης | Ώρες | Μέρες | Χρόνια |
| Πλήθος κόμβων | 32 | 7 | 65536 |
| Εμβέλεια | 100 μέτρα | 10 μέτρα | 100 μέτρα |
| Bandwidth | >11Mbps | 1 Mbps | 250 Kbps |
| Κρυπτογράφηση | Διάφορες | 64 ή 128 bit | 128 bit |

### 3.1.1 Το πρότυπο IEEE 802.15.4 συνοπτικά

Το IEEE 802.15.4 είναι ένα ανοικτό πρότυπο (σε αντίθεση με το ZigBee) το οποίο ανακοινώθηκε το Μάιο του 2003. Είναι ένα απλό πρωτόκολλο για ασύρματα δίκτυα, το οποίο στοχεύει στην ελάχιστη δυνατή κατανάλωση ενέργειας με το ελάχιστο δυνατό κόστος υλοποίησης. Ακόμα, επειδή σχεδιάστηκε για να λειτουργεί σε περιβάλλοντα με υψηλά επίπεδα θορύβου, έχει αρκετά μεγάλη ανοχή σε παρεμβολές και θόρυβο (καλύτερη από την αντίστοιχη του 802.11).

Αυτό επιτυγχάνεται με τη χρήση σε φυσικό επίπεδο Direct Sequence Spread Spectrum (DSSS).

Το πρότυπο καλύπτει τα επίπεδα Physical και MAC, καθώς και μέρος του Link Layer Control. Στο φυσικό επίπεδο υποστήριζονται τρεις ζώνες συχνοτήτων, μία στα 868.3 MHz με ταχύτητα 20 Kbps, ανάμεσα στα 902 και 928 MHz με ταχύτητα 40 Kbps και στα 2.4 GHz με ταχύτητα 250 Kbps. Συνολικά υπάρχουν διαθέσιμα 27 κανάλια. Το MAC επίπεδο του 802.15.4 είναι αρκετά απλούστερο σε σχέση με το αντίστοιχο επίπεδο του Bluetooth. Υποστηρίζονται τοπολογίες *δικτύου αστέρα* (star) και *peer-to-peer*.

Οι κόμβοι ενός δικτύου 802.15.4 έχουν από μια μοναδική 64-bit διεύθυνση, η οποία ανατίθεται με τρόπο που ορίζεται από την IEEE (παρόμοια με τον τρόπο που κάθε κάρτα δικτύου Ethernet έχει μια MAC διεύθυνση). Αυτή η διεύθυνση είναι η *extended* διέυθυνση της συσκευής. Υπάρχει και μια πιο σύντομη 16-bit διεύθυνση, η οποία ανατίθεται από τον coordinator του δικτύου όταν η κάθε συσκευή ξεκινά τη λειτουργία της μέσα στο δίκτυο. Η extended διεύθυνση χρησιμοποιείται για την επικοινωνία μεταξύ γειτονικών κόμβων, ενώ η πιο σύντομη διεύθυνση για την end-to-end επικοινωνία μέσα στο δίκτυο. Κάτι αντίστοιχο με τις MAC (extended) και IP (σύντομες) διευθύνσεις του Internet. Με αυτόν τον τρόπο μπορούμε να έχουμε συνολικά $2^{16} = 65536$ κόμβους σε ένα 802.15.4 δίκτυο, οι οποίοι είναι μάλλον υπεραρκετοί.

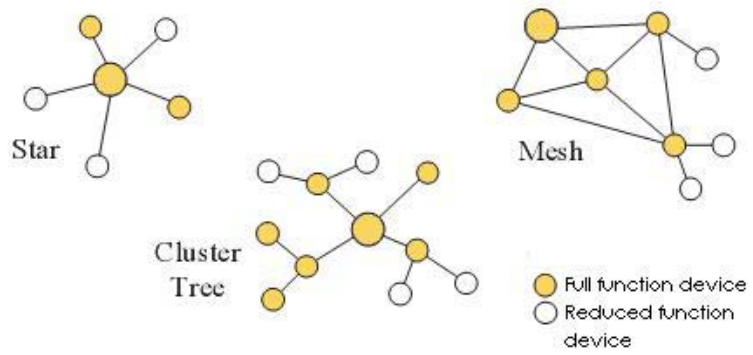
Στο πρότυπο ορίζονται δύο τύποι συσκευών:

1. Κόμβοι *πλήρους λειτουργίας* (full function devices).

2. Κόμβοι *περιορισμένης λειτουργίας* (reduced function devices).

Λίγα λόγια τώρα για τη λειτουργία των δύο τύπων κόμβων. Η διάκριση μεταξύ τους γίνεται τόσο βάση της λειτουργίας τους, όσο και των δυνατοτήτων τους. Έτσι, οι κόμβοι πλήρους λειτουργίας τυπικά έχουν περισσότερες επεξεργαστικές δυνατότητες και πιθανότατα περισσότερα αποθέματα ενέργειας, γιατί έχουν περισσότερα καθήκοντα από τους κόμβους περιορισμένης λειτουργίας. Οι κόμβοι πλήρους λειτουργίας μπορούν να λειτουργήσουν με peer-to-peer τρόπο μεταξύ τους, δηλαδή μπορούν να δρομολογήσουν κίνηση που λαμβάνουν από κάποιο γειτονικό τους κόμβο προς κάποιον άλλον (οποιονδήποτε) κόμβο του δικτύου, επιλέγοντας κάποιο γειτονικό τους κόμβο για την περαιτέρω προώθηση της πληροφορίας. Υπάρχει ένας κόμβος που παίζει το ρόλο του κέντρου ελέγχου (*coordinator*) στο δίκτυο.

Οι κόμβοι περιορισμένης λειτουργίας από την άλλη, μπορούν να επικοινωνήσουν μόνο με μια συσκευή πλήρους λειτουργίας, όπως οι κόμβοι ενός δικτύου με τοπολογία αστέρα (με τον κόμβο πλήρους λειτουργίας στο ρόλο του κεντρικού κόμβου). Επομένως, σε ένα ασύρματο δίκτυο αισθητήρων οι κόμβοι περιορισμένης λειτουργίας αναλαμβάνουν να κάνουν τις απαιτούμενες μετρήσεις και τις προωθούν στους κόμβους πλήρους λειτουργίας για περαιτέρω δρομολόγηση μέσα στο δίκτυο προς το κέντρο ελέγχου.




Σχήμα 3.1: Τοπολογίες δικτύου σε δίκτυα ZigBee.

### 3.1.2 Το πρότυπο ZigBee συνοπτικά

Το πρότυπο ZigBee χρησιμοποιεί το IEEE 802.15.4 στα κατώτερα επίπεδα και υλοποιεί μέρος του Data Link επιπέδου, και τα επίπεδα μέχρι και το interface για τις εφαρμογές που τρέχει ο κάθε κόμβος του δικτύου. Το ZigBee σχεδιάστηκε για να εκτελείται σε 8-bit μικροεπεξεργαστές με μικρό μέγεθος μνήμης. Έτσι, η στοίβα του πρωτοκόλλου καταλαμβάνει μόλις 32 KB για τις συσκευές πλήρους λειτουργίας, ενώ για τις συσκευές περιορισμένης λειτουργίας η απαιτούμενη μνήμη περιορίζεται στα 6 KB.

Το ZigBee παρέχει λειτουργίες εγκατάστασης ενός δικτύου, σύνδεσης και αποσύνδεσης από αυτό, ανάθεσης διευθύνσεων σε κόμβους που συνδέονται σε αυτό, δημιουργίας προγράμματος μεταδόσεων (transmission scheduling), δρομολόγησης, κρυπτογράφησης, κ.α.

Στα ZigBee δίκτυα μπορούμε να έχουμε επιπλέον τύπους τοπολογίας, σε σχέση με τις τοπολογίες που προσφέρει το 802.15.4, όπως η cluster tree και η mesh. Στο σχήμα 3.1 φαίνονται οι δυνατοί τύποι τοπολογίας δικτύου σε ένα 802.15.4 δίκτυο και ο ρόλος των δύο τύπων κόμβων. Αναμένεται όμως λόγω των συνθηκών που επικρατούν στις εφαρμογές για τις οποίες έχει σχεδιασθεί για να χρησιμοποιηθεί το ZigBee, ότι η mesh τοπολογία είναι αυτή που θα χρησιμοποιηθεί περισσότερο. Αυτό διότι με τη mesh τοπολογία υπάρχουν πολλές πιθανές διαδρομές προς το κέντρο ελέγχου (control center) του δικτύου από τους διάφορους κόμβους (πλήρους λειτουργίας) του δικτύου. Το γεγονός αυτό καθιστά το δίκτυο περισσότερο ανθεκτικό σε μεμονωμένες αστοχίες (failures) κόμβων.

Σε αυτή την τοπολογία ο coordinator και οι κόμβοι-δρομολογητές (οι κόμβοι πλήρους λειτουργίας) έχουν ανοιχτούς τους πομποδέκτες τους ενώ οι υπόλοιποι κόμβοι τους ανοίγουν μόνο αν θέλουν να μεταδώσουν ή να λάβουν μηνύματα. Αυτή η προσέγγιση προφανώς βοηθά τους κόμβους περιορισμένης λειτουργίας να καταναλώνουν πολύ λίγη ενέργεια, αλλά οι κόμβοι πλήρους λειτουργίας

χρειάζεται να έχουν (σχετικά) μεγάλα αποθέματα ενέργειας για να μπορέσουν να λειτουργούν για μεγάλα χρονικά διαστήματα (πιθανώς να είναι συνδεδεμένοι σε σταθερές πηγές ενέργειας).

### 3.1.3 Παρατηρήσεις πάνω στη χρήση των IEEE 802.15.4 και ZigBee σε ασύρματα δίκτυα αισθητήρων

Όσον αφορά στη χρήση του IEEE 802.15.4 και του ZigBee σε ασύρματα δίκτυα αισθητήρων, έχουμε να παρατηρήσουμε τα παρακάτω. Τα πρότυπα αυτά είναι ένα σημαντικό βήμα για την καθιέρωση προτυποποιημένων λύσεων στα ασύρματα δίκτυα αισθητήρων. Ως τώρα, ακόμα και στα προϊόντα της Crossbow, από γενιά σε γενιά Mica mote είχαμε χρήση διαφορετικών πομποδεκτών (διαφορετικη ζώνη συχνοτήτων, διαμόρφωση, κτλ). Το γεγονός αυτό καθιστούσε αδύνατη την επικοινωνία μεταξύ motes διαφορετικών τύπων. Τώρα, με την καθιέρωση των προτύπων αυτών, μπορούμε να αναμένουμε ότι οι επόμενες γενιές motes θα μπορούν να επικοινωνήσουν μεταξύ τους, αλλά και επίσης και με συσκευές από διαφορετικούς κατασκευαστές.

Επίσης, τα πρότυπα αυτά σε μεγάλο βαθμό ικανοποιούν τις απαιτήσεις που υπάρχουν στα ασύρματα δίκτυα αισθητήρων και απλοποιούν την επικοινωνία στο δίκτυο και τον προγραμματισμό των εφαρμογών για τέτοια δίκτυα. Τέλος, προσφέρουν ικανοποιητικό (και αρκετά μεγαλύτερο) bandwidth, σε σχέση με τις ως τώρα συσκευές, που φτάνει στα 250 Kbps.

Από την άλλη μεριά, υπάρχουν και κάποια μειονεκτήματα σε αυτή την προσέγγιση. Η στοίβα του πρωτοκόλλου (στους κόμβους πλήρους λειτουργίας) είναι αρκετά μεγάλη για τα δεδομένα των σημερινών συσκευών για ασύρματα δίκτυα αισθητήρων (περιορίζει αρκετά τη διαθέσιμη μνήμη για εφαρμογές). Επιπλέον, το IEEE 802.15.4 παρέχει πολλά χαρακτηριστικά, τα οποία δεν είναι χρήσιμα ή απαραίτητα στο σύνολό τους στα ασύρματα δίκτυα αισθητήρων. Ακόμα, το πρωτόκολλο απαιτεί πολλούς υπολογισμούς λόγω των χαρακτηριστικών που παρέχει, και επόμενως ενέργεια.

Μια λύση σε αυτά τα προβλήματα είναι η υποστήριξη μόνο του υποσυνόλου των χαρακτηριστικών που είναι χρήσιμα, χρησιμοποιώντας hardware συμβατό με το 802.15.4. Τέλος, η επικοινωνία μεταξύ διαφορετικών τύπων motes δεν είναι μια εύκολη υπόθεση, καθώς μπαίνουν στη μέση διάφορα θέματα, όπως η διαφορετική αρχιτεκτονική (π.χ. τα Mica motes είναι 8-bit ενώ τα Telos motes είναι 16-bit), η διάταξη των bytes στις λέξεις (little ή big endian), κτλ.

## 3.2 Πλατφόρμες για ασύρματα δίκτυα αισθητήρων που χρησιμοποιούν τα πρότυπα IEEE 802.15.4 και ZigBee

Στη συνέχεια παρουσιάζουμε συνοπτικά δύο τύπους mote που χρησιμοποιούν τα δύο αυτά πρότυπα για την επικοινωνία σε ένα δίκτυο ασύρματων αισθητήρων.

Σχήμα 3.2: Το MicaZ mote.

### 3.2.1 MicaZ

Το MicaZ mote είναι η τέταρτη γενιά των MICA motes που ακολούθησε μετά τα Mica2 και Mica2Dot. Η ουσιαστική διαφορά με την προηγούμενη γενιά είναι η χρήση ενός πομποδέκτη (ChipCon CC2420) που υλοποιεί το πρότυπο IEEE 802.15.4. Το υπόλοιπο hardware είναι όμοιο με το hardware του Mica2. Στο σχήμα 3.2 φαίνεται ένα MicaZ mote.

Συνοπτικά, τα χαρακτηριστικά του MicaZ είναι:

- Επεξεργαστής 8-bit Atmel AVR ATmega128L στα 8 MHz.
- 128 KB flash μνήμης για την αποθήκευση εφαρμογών.
- 4 KB SRAM μνήμης για χρήση από την εφαρμογή που εκτελείται στο Mote.
- 512 KB serial flash μνήμης για την αποθήκευση μετρήσεων.

Όλα τα programming και sensor board για Mica και Mica2 mote μπορούν να χρησιμοποιηθούν και με το MicaZ.

Τα χαρακτηριστικά του ChipCon 2420 συνοπτικά είναι τα εξής:

- Υποστήριξη του προτύπου IEEE 802.15.4 για ασύρματα δίκτυα.
- Συχνότητα εκπομπής στα 2.4 GHz με χρήση DSSS (Direct Sequence Spread Spectrum) για αυξημένη ανοχή σε παρεμβολές και θόρυβο.
- Ταχύτητα μετάδοσης δεδομένων 250 Kbps.

Σχήμα 3.3: Το Tmote Sky mote.

- Υποστήριξη από το λειτουργικό σύστημα TinyOS (από την έκδοση 1.1.7 και μετά).

Το MicaZ mote σε συνδυασμό με την ενσωματωμένη κεραία την οποία διαθέτει έχει εμβέλεια μετάδοσης μέχρι και 30 μέτρα σε εσωτερικούς χώρους και 100 μέτρα σε εξωτερικούς χώρους.

### 3.2.2 Telos και Tmote Sky

Η πλατφόρμα Telos σχεδιάστηκε στο Πανεπιστήμιο Berkeley στην Καλιφόρνια, ως εναλλακτική πρόταση στα Mica motes. Πάνω σε αυτή την πλατφόρμα βασίστηκε η εταιρία Moteiv και το αποτέλεσμα, μετά από κάποιες αλλαγές στον αρχικό σχεδιασμό, ήταν το *Tmote Sky* mote. Στο σχήμα 3.3 φαίνεται ένα TmoteSky mote.

Τα βασικά χαρακτηριστικά του Tmote Sky είναι:

- Χρήση καθιερωμένων προτύπων για τη διασύνδεση με PC και την επικοινωνία μέσα στο ασύρματο δίκτυο αισθητήρων, δηλαδή χρήση USB θύρας και πομποδέκτη IEEE 802.15.4 (ChipCon CC2420).
- Επεξεργαστής 16-bit Texas Instruments MSP430 στα 8 MHz.
- 10 KB SRAM μνήμη για χρήση από την εφαρμογή που εκτελείται στο mote.
- 48 KB flash μνήμη για αποθήκευση εφαρμογών.
- 1 MB serial μνήμη για αποθήκευση μετρήσεων.
- Ενσωματωμένη κεραία με εμβέλεια μετάδοσης μέχρι 50 μέτρα σε εσωτερικούς χώρους και μέχρι 125 μέτρα σε εξωτερικούς χώρους.

- Εξαιρετικά χαμηλή κατανάλωση ενέργειας (χαμηλότερη από όλα τα Mica motes).
- Ενσωματωμένοι αισθητήρες για υγρασία, θερμοκρασία και φως, χωρίς την ανάγκη ξεχωριστού sensor board.

Για τη διασύνδεση με το PC χρησιμοποιείται ειδικό λογισμικό που επιτρέπει στο PC να βλέπει μια USB θύρα ως σειριακή θύρα. Εφόσον γίνει αυτό, είναι εύκολη η διασύνδεση με το TinyOS και όλα τα περιβάλλοντα (TinyDB, TASK) που χρησιμοποιούν τη σειριακή θύρα ως interface με τα motes.

### 3.2.3 Σύγκριση MicaZ και Tmote Sky

Είναι γεγονός ότι και οι δύο πλατφόρμες χρησιμοποιούν το 802.15.4 για την ασύρματη επικοινωνία μέσα στο δίκτυο (μάλιστα χρησιμοποιούν τον ίδιο ακριβώς πομποδέκτη), οπότε, θεωρητικά τουλάχιστον, μπορούν να συνυπάρξουν στο ίδιο ασύρματο δίκτυο αισθητήρων, αν και στην πράξη μπορούν να παρουσιαστούν προβλήματα και ασυμβατότητες στην επικοινωνία μεταξύ τους.

Από τη μια μεριά, το Tmote Sky έχει σχεδιαστεί έπειτα από το Mica2 mote (το οποίο είναι σχεδόν όμοιο με το MicaZ όπως αναφέρθηκε). Η διαφορετική του σχεδίαση πρακτικά έχει ως αποτέλεσμα την μικρότερη κατανάλωση ενέργειας σε σχέση με το MicaZ. Επίσης, θεωρητικά, αξιοποιεί για μεγαλύτερο χρονικό διάστημα τις μπαταρίες που χρησιμοποιεί ως πηγή ενέργειας, γιατί μπορεί να λειτουργήσει σε χαμηλότερη ελάχιστη τάση από ότι το MicaZ. Ακόμα, έχει μικρότερους χρόνους μετάβασης από τη μια κατάσταση στην άλλη, δηλαδή από κατάσταση sleep σε active, από την κατάσταση που ο πομποδέκτης είναι κλειστός μέχρι να λειτουργήσει, κτλ, αν και οι χρόνοι αυτοί είναι ελάχιστοι και στο MicaZ.

Ένα άλλο πλεονέκτημα του Tmote Sky είναι ότι δεν χρειάζεται ειδικό programming board για να συνδεθεί σε κάποιο PC, αφού αυτό μπορεί να γίνει πολύ εύκολα μέσω της USB θύρας που διαθέτει, σε αντίθεση με το MicaZ.

Από την άλλη, το πλεονέκτημα του MicaZ είναι ότι μπορεί να αξιοποιήσει το υπάρχον hardware που έχει τυχόν αγοραστεί για τις προηγούμενες γενιές mote, δηλαδή τα sensor boards, το κόστος των οποίων δεν είναι διόλου ευκαταφρόνητο και επιπλέον διαθέτουν αισθητήρες διαφόρων τύπων με αρκετές δυνατότητες. Τα board αυτά δεν μπορούν να χρησιμοποιηθούν με το Tmote Sky.

## 3.3 Η πλατφόρμα StarGate

Η πλατφόρμα Stargate είναι αποτέλεσμα της συνεργασίας των εταιριών Intel και Crossbow, και κατασκευάζεται από την τελευταία. Είναι μια πλατφόρμα hardware ειδικά σχεδιασμένη για να καλύπτει την ανάγκη για κέντρο ελέγχου ενός ασύρματου δικτύου αισθητήρων και πιο συγκεκριμένα, είναι σχεδιασμένη για να συνεργάζεται εύκολα με την οικογένεια motes Mica της Crossbow. Το Stargate διατίθεται από την Crossbow με μια προεγκατεστημένη διανομή Linux μαζί με κάποιους εξειδικευμένους drivers.

Το Stargate διαθέτει αρκετά μεγάλες επεξεργαστικές δυνατότητες, εφόσον διαθέτει ένα RISC επεξεργαστή XScale της Intel στα 400 MHz. Παράλληλα, διαθέτει πλήθος διασυνδέσεων, δηλαδή θύρες RS-232, Ethernet, USB, Mica GPIO (δηλαδή θύρα μέσω της οποίας μπορεί να συνδεθεί με ένα Mica mote χωρίς την παρεμβολή άλλου board, καθώς και δυνατότητα ασύρματης επικοινωνίας μέσω δικτύου IEEE 802.11a ή b. Σε συνδυασμό με το πολύ μικρό μέγεθός του, μπορεί να χαρακτηριστεί ως single board computer, και είναι ιδανικό για κέντρο ελέγχου (sink, control center) σε ένα ασύρματο δίκτυο αισθητήρων. Επιπλέον, έχει ήδη δοκιμαστεί σε πραγματικά πειραματικά ασύρματα δίκτυα αισθητήρων (όπως το *ESS*).

Το Stargate αποτελείται από δύο board, το processor board και μια daughter κάρτα. Στο processor board βρίσκεται ο επεξεργαστής μαζί με μια μνήμη SDRAM 64 MB και μια μνήμη flash 32 MB. Ακόμα, υπάρχουν οι θύρες PCMCIA, Compact Flash και η θύρα διασύνδεσης με Mica mote. Στη daughter καρτα βρίσκονται οι θύρες Ethernet (10 Base-T), RS-232, USB, JTAG. Τέλος, η διανομή Linux η οποία βρίσκεται προεγκατεστημένη στη flash μνήμη του Stargate καταλαμβάνει περίπου 10 MBytes.

Στον πίνακα που ακολουθεί φαίνονται συνοπτικά τα χαρακτηριστικά του Stargate.

| **Stargate Processor Board** | |
|---|---|
| Intel XScale | 400 MHz RISC CPU |
| Intel SA1111, StrongARM | Multiple I/O Chip |
| 64 MB SDRAM | Μνήμη συστήματος |
| 32 MB Flash | Για δεδομένα και εφαρμογές |
| PCMCIA slot | Type II |
| Compact Flash slot | Type II |
| 51-pin MICA slot | Για σύνδεση με MICA mote |
| **Stargate Daughter Board** | |
| 10 Base-T port | Δίκτυο Ethernet |
| RS-232 port | Σειριακή θύρα |
| USB port | Έκδοση 1.1 |

## 3.4 Spec: μια πραγματική πλατφόρμα "έξυπνης σκόνης"

Η πλατφόρμα Spec είναι μια προσπάθεια υλοποίησης σε επίπεδο ASIC της γενικευμένης αρχιτεκτονικής για κόμβους ασύρματων δικτύων αισθητήρων πάνω στην οποία βασίστηκε η υλοποίηση των Mica mote. Σκοπός της σχεδίασης και υλοποίησης του Spec ήταν να αποδειχθεί το εφικτό της υλοποίσης ενός κόμβου για τέτοια δίκτυα σε μικροσκοπικό μέγεθος, τέτοιο που να δικαιολογεί το χαρακτηρισμό *smart dust*. Περισσότερες λεπτομέρειες για την πλατφόρμα Spec μπορεί κάποιος να βρει στην εργασία [17]. Το μέγεθος του Spec mote φαίνεται στο σχήμα 3.4.

Συνοπτικά, το Spec είναι ένα chip μικροσκοπικού μεγέθους 2.5 mm επί 2.5 mm, το οποίο περιλαμβάνει έναν 8-bit RISC επεξεργαστή (με σύνολο εντολών


Σχήμα 3.4: Το SPEC mote επάνω στην επιφάνεια ενός επεξεργαστή AVR που χρησιμοποιείται στα Mica mote.

16-bit), έναν RF πομποδέκτη που λειτουργεί στα 900 MHz, 3 KB SRAM μνήμης, μαζί με έναν ελεγκτή μνήμης (memory controller). Εκτός αυτών, ο επεξεργαστής είναι συνδεδεμένος με ένα μετατροπέα αναλογικού σήματος σε ψηφιακό, ο οποίος έχει σχεδιαστεί για να καταναλώνει ελάχιστη ενέργεια, I/O port γενικού σκοπού, ένα υποσύστημα κρυπτογράφησης και μερικούς system timers. Το γεγονός ότι το Spec είναι υλοποιημένο ως ένα μοναδικό chip δίνει το πλεονέκτημα, σε σχέση με τα Mica mote, ότι δεν χρειάζονται εξωτερικές συνδέσεις μεταξύ των διαφόρων υποσυστημάτων του mote, οι οποίες αφενός περιορίζουν την απόδοση του συστήματος, αφετέρου αυξάνουν το κόστος κατασκευής του.

Το RF υποσύστημα αντί να περιλαμβάνει την υλοποίηση σε hardware κάποιου συγκεκριμένου πρωτοκόλλου, περιέχει κάποια υποσυστήματα (communication primitives) που βοηθούν στην τέλεση βασικών λειτουργιών στα επικοινωνιακά συστήματα, όπως συγχρονισμός (δηλαδή της ανίχνευσης εισερχόμενου μηνύματος), κρυπτογράφηση, serialization της πληροφορίας που θα μεταδοθεί, κτλ.

Μέσω της υλοποίησης σε chip, όπως αναφέραμε, μειώνεται δραστικά το κόστος κατασκευής του mote. Έτσι, ενώ το Mica mote (το 2003) είχε κόστος κατασκεύης γύρω στα 60 δολάρια, *το Spec είχε κόστος κατασκευής κάτω από 1 δολάριο*. Πιο συγκεκριμένα, το κόστος για το ολοκληρωμένο ήταν 0.25 δολάρια, μαζί με 0.16 δολάρια για μια μικρή μπαταρία, 0.1 δολάριο για ένα κρύσταλλο ως εξωτερικό ρολόι και κάποια άλλα μικροκόστη. Το Spec mote λοιπόν, αποδεικνύει την αξία και την επιτευξιμότητα της υλοποίησης σε ένα chip ενός κόμβου για ασύρματα δίκτυα αισθητήρων.

# Μέρος ΙΙ

# Το πρωτόκολλο VTRP

# Κεφάλαιο 4

# Το πρωτόκολλο VTRP για τη διάδοση πληροφορίας σε δίκτυα έξυπνης σκόνης

Σε αυτή την ενότητα θα παρουσιαστεί αναλυτικά το πρωτόκολλο VTRP [6], το οποίο προσπαθεί να λύσει το πρόβλημα του εντοπισμού και διάδοσης πολλαπλών γεγονότων (multiple event detection and propagation) στα δίκτυα έξυπνης σκόνης. Το πρόβλημα αυτό συνίσταται στον εντοπισμό ενός πλήθους από καίρια συμβάντα μέσα στο δίκτυο και της διάδοσης των αναφορών για τα συμβάντα αυτά σε ένα κέντρο ελέγχου, με έναν αποδοτικό τρόπο όσον αφορά στην κατανάλωση ενέργειας και στην ανοχή σε σφάλματα μετάδοσης. Το πρόβλημα αυτό αποτελεί γενίκευση του προβλήματος διάδοσης ενός μοναδικού συμβάντος, όπως αυτό παρουσιάζεται στα [9], [10], [7].

Το στοιχείο που διαφοροποιεί το VTRP σε σχέση με τα υπόλοιπα συναφή πρωτόκολλα είναι η μεταβολή της εμβέλειας μετάδοσης των κόμβων του δικτύου έξυπνης σκόνης, όταν αυτό επιβάλλεται από τις συνθήκες που επικρατούν στο δίκτυο. Αυτό συμβάλλει στην καλύτερη αντιμετώπιση, σε σχέση με τα πρωτόκολλα που χρησιμοποιούν σταθερή εμβέλεια μετάδοσης, καταστάσεων όπως:

- Δίκτυα έξυπνης σκόνης με χαμηλή πυκνότητα κόμβων (βλέπε ενότητα 4.1). Σε αυτή την περίπτωση, τα πρωτόκολλα με σταθερή εμβέλεια μετάδοσης μπορεί να αποτύχουν να βρουν το επόμενο βήμα προς την κατεύθυνση του κέντρου ελέγχου. Με την αύξηση της εμβέλειας μετάδοσης μπορεί να βρεθούν κόμβοι για να προωθηθεί η πληροφορία προς το κέντρο ελέγχου.

- Περιπτώσεις όπου εμπόδια δυσχεραίνουν την επικοινωνία μεταξύ των κόμβων ή κόμβοι οι οποίοι έχουν κλείσει τους πομποδέκτες τους. Επίσης, με την αύξηση της εμβέλειας μετάδοσης είναι πιθανόν να υπερπηδηθούν κόμβοι που αλλιώς θα είχαν πολύ μεγάλύτερο φόρτο επικοινωνίας γενικότερα και προώθησης μηνυμάτων ειδικότερα, όπως οι κόμβοι που βρίσκονται κοντά στο κέντρο ελέγχου.

Για να δείξουμε τη συμπεριφορά του VTRP σε τέτοιες περιπτώσεις, γίνεται παράλληλα η σύγκρισή του με το LTP ([10], [20]), ένα αντίστοιχο πρωτόκολλο. Το LTP χρησιμοποιεί μια πιθανοτική λογική για την προώθηση της πληροφορίας μέσα στο δίκτυο έξυπνης σκόνης, και η λειτουργία του στηρίζεται σε μεγάλο βαθμό στην πυκνότητα του δικτύου. Μεγάλη πυκνότητα δικτύου σημαίνει σημαίνει μεγάλο ποσοστό επιτυχών μεταδόσεων πληροφορίας στο κέντρο ελέγχου. Πειραματικά αποτελέσματα δείχνουν ότι η χρήση του VTRP οδηγεί έως και σε 100% βελτιώση, σε σχέση με το LTP. Επίσης, παρουσιάζονται και αναλύονται πειραματικά τέσσερις διαφορετικοί μηχανισμοί μεταβολής της εμβέλειας μετάδοσης.

## 4.1 Το μοντέλο δικτύου που χρησιμοποιείται στο VTRP

Για τις ανάγκες της παρουσίασης του VTRP, υποθέτουμε ένα δίκτυο έξυπνης σκόνης, του οποίου οι κόμβοι είναι σταθεροί (και όχι κινητοί), ο καθένας εκ των οποίων φέρει ένα πλήθος αισθητήρων μέσω των οποίων αντιλαμβάνονται συμβάντα σε ένα δισδιάστατο επίπεδο στο οποίο έχουν τοποθετηθεί με κάποιο τρόπο, και χαρακτηρίζονται από το ενεργειακό κόστος μετάδοσης και λήψης πληροφορίας.

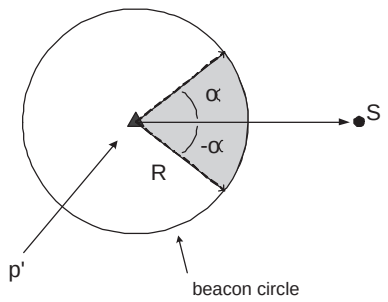
Ορίζουμε ως *πυκνότητα* $d$ (density) ενός δικτύου έξυπνης σκόνης τον λόγο του πλήθους $n$ των κόμβων του δικτύου προς την επιφάνειά του. Επίσης, ονομάζουμε $S$ ένα σημείο στην επιφάνεια που βρίσκεται το δίκτυο, το οποίο σημείο αντιπροσωπεύει το κέντρο ελέγχου στο οποίο θα πρέπει να προωθηθεί με κάποιο τρόπο η πληροφορία από κάθε κόμβο του δικτύου. Υποθέτουμε ακόμα, ότι υπάρχει μια φάση αρχικοποίησης, στη διάρκεια της οποίας οι κόμβοι επικοινωνούν μεταξύ τους και μαθαίνουν την γενική κατεύθυνση προς την οποία βρίσκεται το σημείο $S$ (υποθέτουμε ότι οι κόμβοι έχουν γενικά αίσθηση προσανατολισμού).

Οι κόμβοι του δικτύου είναι εφοδιασμένοι με πλήθος αισθητήρων για φως, υγρασία, θερμοκρασία, κτλ., και διαθέτουν δύο τρόπους μετάδοσης πληροφορίας:

1. *Broadcast beacon* μετάδοση, η οποία είναι κατευθυνόμενη μετάδοση γύρω από μια γωνία $\alpha$, όπως φαίνεται στο σχήμα 4.1, χρησιμοποιώντας κάποια ειδική για το σκοπό αυτό κεραία.

2. *Κατευθυνόμενη προς σημείο* (directed to a point) μετάδοση, πιθανώς μέσω οπτικής επικοινωνίας (π.χ. laser), η οποία καταναλώνει λιγότερη ενέργεια από τον Broadcast beacon τρόπο μετάδοσης.

Στο μοντέλο του VTRP, γίνεται η υπόθεση πως η εμβέλεια μετάδοσης $R$ των κόμβων του δικτύου μπορεί να μεταβληθεί, όπως μπορεί να γίνει στα MICA motes, στα οποία μπορούμε να μεταβάλλουμε μέσω του software την τιμή της ισχύος του πομποδέκτη της συσκευής. Η γωνία $\alpha$ αντίθετα θεωρείται σταθερή




Σχήμα 4.1: Κατευθυνόμενη μετάδοση γωνίας $\alpha$

και δε μπορεί να αλλάξει. Προφανώς, η γωνία $\alpha$ μπορεί να είναι ίση με $\pi$, οπότε δεν υπάρχει πλέον κατευθυνόμενη μετάδοση.

Κάθε κόμβος του δικτύου, καθ' όλη τη διάρκεια της λειτουργίας του, μπορεί να βρίσκεται σε μια από τις παρακάτω καταστάσεις:

- Μετάδοση μηνύματος.
- Λήψη μηνύματος.
- Χρήση των αισθητήρων για έλεγχο του περιβάλλοντος πεδίου.

Όσον αφορά στην κατανάλωση ενέργειας ακολουθούμε το παρακάτω μοντέλο. Έστω $E_{elec}$ η ενέργεια που απαιτείται για να λειτουργήσει ο πομποδέκτης και $\epsilon_{amp}$ η ενέργεια για την ενίσχυση του σήματος προκειμένου να επιτευχθεί αξιοπρεπές signal-to-noise ratio (SNR). Επίσης, υποθέτουμε ότι η ενέργεια για τη μετάδοση σε απόσταση $r$ αυξάνεται με το τετράγωνο της απόστασης ($r^2$). Επομένως, για τη μετάδοση ενός μηνύματος μεγέθους $k$-bit σε απόσταση $r$, η ενέργεια που καταναλώνεται είναι:

$$E_T(k,r) = E_{T-elec}(k) + E_{T-amp}(k,r)$$
$$E_T(k,r) = E_{elec} \cdot k + \epsilon_{amp} \cdot k \cdot r^2$$

ενώ για τη λήψη του ίδιου μηνύματος η ενέργεια που καταναλώνεται είναι:

$$E_R(k) = E_{R-elec}(k)$$
$$E_R(k,r) = E_{elec} \cdot k$$

Υπάρχουν τρεις διαφορετικές πηγές κατανάλωσης ενέργειας:

1. $E_T$: ενέργεια για τη μετάδοση μηνύματος.
2. $E_R$: ενέργεια για τη λήψη μηνύματος.
3. $E_{idle}$: ενέργεια που κατανάλωνεται στην κατάσταση όπου ο πομποδέκτης είναι κλειστός. Υποθέτουμε ότι η ενέργεια στην κατάσταση είναι σταθερή για κάποια μονάδα χρόνου και ισουται με $E_{elec}$.

Στα πειράματα που έγιναν μέσω εξομοίωσης και περιγράφονται στη σύγκριση μεταξύ VTRP και LTP, χρησιμοποιείται αυτό το μοντέλο κατανάλωσης ενέργειας. Αν και απλό, το μοντέλο αυτό θεωρείται επαρκές για να περιγράψει τι συμβαίνει στην πράξη στον τομέα αυτό σε ένα δίκτυο έξυπνης σκόνης.

## 4.2 Το πρόβλημα της διάδοσης πολλαπλών γεγονότων

Έστω ότι μέσα στο δίκτυο έξυπνης σκόνης συμβαίνει ένα πλήθος από $k$ γεγονότα $\mathcal{E}_i$, $i = 1, \ldots, k$, όπου κάθε γεγονός γίνεται αντιληπτό από ένα κόμβο $p_i$ ($i = 1, 2, \ldots, K$). Τότε, το πρόβλημα της διάδοσης πολλαπλών γεγονότων $\mathcal{P}$ ορίζεται ως εξής:

> "Πώς μπορεί κάθε κόμβος του δικτύου $p_i$ ($i = 1, 2, \ldots, K$), συνεργαζόμενος με τους υπόλοιπους κόμβους να μεταδώσει την πληροφορία *info*($\mathcal{E}_i$) σχετικά με την εμφάνιση κάποιου γεγονότος $\mathcal{E}_i$ πίσω στο κέντρο ελέγχου $\mathcal{S}$ με έναν αποδοτικό τρόπο, σε ότι αφορά στην κατανάλωση ενέργειας και τον χρόνο μετάδοσης από το $p_i$ ως το $\mathcal{S}$;"

Σε περιπτώσεις δικτύων με μεγάλη πυκνότητα, στα οποία οι κόμβοι βρίσκονται κοντά ο ένας στον άλλο, η επικοινωνία σε μικρές αποστάσεις είναι πιο αποδοτική από την απ' ευθείας μετάδοση στο $\mathcal{S}$ (η οποία λόγω απόστασης ενδέχεται και να είναι ανέφικτη). Επιπλέον, είναι δυνατή η υπερπήδηση εμποδίων η οποία αλλιώς δεν θα μπορούσε να γίνει. Επίσης, από τη μικρή εμβέλεια μετάδοσης μπορεί να ωφεληθεί η ασφάλεια των πληροφοριών που κινούνται μέσα στο δίκτυο.

Η μετάδοση σε μεγάλες αποστάσεις, από την άλλη, σχετίζεται με τη συμμετοχή μικρού πλήθους κόμβων κατά τη διάρκεια της μεταφοράς της πληροφορίας προς το κέντρο ελέγχου. Ως αποτέλεσμα αυτού, υπάρχει λιγότερο overhead στους κόμβους και μείωση του συνολικού χρόνου μετάδοσης της πληροφορίας. Επίσης, με τον τρόπο αυτό μπορούν να χρησιμοποιηθούν τεχνικές clustering για την οργάνωση του δικτύου και τη διάδοση της πληροφορίας.

Επομένως, μπορούν να υπάρξουν πολλές προσεγγίσεις στο πρόβλημα της διάδοσης πολλαπλών γεγονότων $\mathcal{P}$. Επιπλέον, εκτός της μικρής ή μεγάλης εμβέλειας μετάδοσης, υπάρχουν κάποιες επιπρόσθετες παράμετροι που πρέπει να ληφθούν υπόψιν στη διαδικασία σχεδίασης ενός πρωτοκόλλου για το πρόβλημα $\mathcal{P}$, και οι οποίες είναι οι εξής:

(α) *Υπερπήδηση εμποδίων:* Μία τεχνική για την αντιμετώπιση εμποδίων είναι η αύξηση της εμβέλειας μετάδοσης.

(β) *Ανοχή σε λάθη:* Η αυξημένη εμβέλεια μετάδοσης βοηθά στο να ανακαλύπτονται κόμβοι με τους οποίους δεν είναι δυνατόν να γίνει επικοινωνία χρησιμοποιώντας μια σταθερή μικρή εμβέλεια μετάδοσης. Η αδυναμία εύρεσης κόμβων για επικοινωνία και προώθηση της πλοηροφορίας μπορεί να οφείλεται σε "χαλασμένους" ή ανενεργούς κόμβους (που έχουν

κλείσει τους πομποδέκτες τους δηλαδή) ή στην περίπτωση δικτύων με πολύ χαμηλή πυκνότητα.

(γ) *Διάρκεια ζωής του δικτύου:* Με τον όρο αυτό εννοούμε τον συνολικό χρόνο στον οποίο οι κόμβοι του δικτύου είναι *ενεργοί* (active), αφού αυτό επιρρεάζει την ικανότητα μεταφοράς της πληροφορίας προς το κέντρο ελέγχου. Όσο πιο μεγάλο πλήθος κόμβων εξαντλούν τα αποθέματα ενέργειάς τους, τόσο μειώνονται οι πιθανές δίοδοι προς το κέντρο ελέγχου. Πιο συγκεκριμένα υπάρχει το πρόβλημα με τους κόμβους πλησίον του κέντρου ελέγχου, των οποίων η ενέργεια εξαντλείται με γρήγορους ρυθμούς.

## 4.3 Το πρωτόκολλο VTRP αναλυτικά

Στο VTRP (Varying Transmission Range Protocol) κάθε κόμβος $p'$ του δικτύου ο οποίος έχει λάβει μια πληροφορία info($\mathcal{E}$) από κάποιον άλλο κόμβο $p$ (πιθανώς με τη συνδρομή ενδιάμεσων κόμβων), κάνει τα παρακάτω:

**1. Φάση αναζήτησης (Search phase):** Ο $p'$ περιοδικά εκπέμπει ένα broadcast beacon σήμα για να ανακαλύψει άλλους κόμβους στην κατεύθυνση του $\mathcal{S}$ (οι οποίοι βρίσκονται πιο κοντά στο κέντρο ελέγχου). Δημιουργεί μια λίστα με τους κόμβους που ανακαλύπτει και από αυτήν διαλέγει κάποιο κόμβο $p''$, ο οποίος είναι ο κόμβος που βρίσκεται πιο κοντά στο κέντρο ελέγχου από τους κόμβους της λίστας (σχήμα 4.1).

**2. Φάση μετάδοσης (Direct transmission phase):** Ο $p'$ στέλνει την πληροφορία *info($\mathcal{E}$)* στον $p''$ και έπειτα στέλνει ένα μήνυμα επιτυχίας (*success* message) πίσω στον κόμβο από τον οποίο έλαβε την πληροφορία αυτή.

**3. Φάση μεταβολής της εμβέλειας μετάδοσης (Transmission Range Variation phase):** Αν η φάση αναζήτησης δε βρει κάποιον κόμβο πιο κοντά στο $\mathcal{S}$, ο κόμβος $p'$ μπαίνει στην τρίτη φάση του πρωτοκόλλου. Κάθε κόμβος διατηρεί ένα counter $\tau$, ο οποίος αρχικοποιείται στην τιμή $\tau = 0$. Κάθε φορά που αποτυγχάνει η φάση αναζήτησης, η τιμή του $\tau$ αυξάνεται κατά 1. Ο κάθε κόμβος αποφασίζει αν θα μεταβάλλει την εμβέλεια μετάδοσής του $\mathcal{R}$ με βάση την τιμή του $\tau$. Η συνάρτηση μεταβολής $\mathcal{F}(\tau)$ της εμβέλειας μετάδοσης μπορεί να είναι μια από τις εξής:

(α) *Σταθερή αύξηση.* Αυτή η συνάρτηση μεταβολής είναι κατάλληλη για περιπτώσεις δικτύων με μεγάλο πλήθος κόμβων, στα οποία μικρή μεταβολή στην εμβέλεια μετάδοσης σημαίνει μεγάλη πιθανότητα έυρεσης νέων γειτονικών κόμβων. Η συνάρτηση μεταβολής $\mathcal{F}(\tau)$ ορίζεται ως εξής:

$$\mathcal{F}(\tau) = \mathcal{R}_{new} = \mathcal{R}_{init} + c \cdot \tau \ ,$$

όπου $c$ είναι μία σταθερά με μικρή τιμή, π.χ. $c = 10$.

Αυτή θεωρείται η βασική εκδοχή του VTRP και συμβολίζεται ως $\mathsf{VTRP_c}$.

(β) *Πολλαπλασιαστική αύξηση.* Με αυτή τη συνάρτηση μεταβολής, η εμβέλεια μετάδοσης αυξάνεται πιο δραστικά. Αυτή η παραλλαγή του πρωτοκόλλου συμβολίζεται ως $\mathsf{VTRP_m}$. Η συνάρτηση μεταβολής $\mathcal{F}(\tau)$ ορίζεται ως εξής:

$$\mathcal{F}(\tau) \ = \mathcal{R}_{new} \ = \ \mathcal{R}_{init} \ + \ \mathcal{R}_{init} \cdot m \cdot \tau \ ,$$

όπου $m$ είναι μία σταθερά με μικρή τιμή, π.χ. $m = 3$.

Προφανώς, μεγαλύτερη αύξηση στην εμβέλεια μετάδοσης σημαίνει μεγαλύτερη πιθανότητα εύρεσης νέων γειτόνων αλλά και μεγαλύτερη κατανάλωση ενέργειας.

(γ) *Εκθετική αύξηση.* Η αύξηση στην εμβέλεια μετάδοσης εδώ γίνεται με εκθετικό τρόπο. Η συνάρτηση μεταβολής $\mathcal{F}(\tau)$ ορίζεται ως εξής:

$$\mathcal{F}(\tau) \ = \mathcal{R}_{new} \ = \ \mathcal{R}_{init} \ + \ \mathcal{R}_{init}^{\sqrt{(\tau+1)}}$$

Αυτή η παραλλαγή του πρωτοκόλλου συμβολίζεται ως $\mathsf{VTRP_p}$.

(δ) *Τυχαιοποιημένη αύξηση.* Όταν δεν υπάρχει εκ των προτέρων κάποια πληροφόρηση για την πυκνότητα του δικτύου, εφαρμόζουμε μια τυχαιοποιημένη αύξηση στην εμβέλεια μετάδοσης του κόμβου για να αντιμετωπίσουμε όλες τις πιθανές περιπτώσεις πυκνότητας δικτύου. Αυτή η παραλλαγή του πρωτοκόλλου συμβολίζεται ως $\mathsf{VTRP_r}$ και η συνάρτηση μεταβολής $\mathcal{F}(\tau)$ ορίζεται ως εξής:

$$\mathcal{F}(0) \ = \ \mathcal{R}_{init}$$

$$\mathcal{F}(\tau) \ = \ \mathcal{F}(\tau - 1) \ + \ \mathcal{R}_{init} \cdot r \ ,$$

όπου $r \in (0, 8]$ είναι μία τυχαία μεταβλητή.

Ανά πάσα χρονική στιγμή μέσα στο δίκτυο έξυπνης σκόνης μπορούν να συμβαίνουν πολλά γεγονότα παράλληλα. Προκειμένου να αποφευχθούν επαναμεταδόσεις μηνυμάτων για το ίδιο γεγονός κτλ., κάθε γεγονός σχετίζεται με ένα *event ID* και κάθε κόμβος διατηρεί μια ποσότητα cache μνήμης. Τα event IDs είναι μοναδικά για το κάθε γεγονός και αποτελούνται από το timestamp του γεγονότος (δηλαδή το πότε ανιχνεύθηκε από τον κόμβο το συγκεκριμένο γεγονός) και το ID του κόμβου ο οποίος ανίχνευσε το γεγονός. Κατά τη λήψη ενός μηνύματος, ο κόμβος ελέγχει την cache του για το αν έχει προωθήσει ξανά κάποιο μήνυμα για το ίδιο γεγονός. Αν όχι, ακολουθείται η καθιερωμένη διαδικασία προώθησης του μηνύματος. Αν έχει ήδη προωθηθεί, το μήνυμα αγνοείται. Το μέγεθος της cache μνήμης δεν χρειάζεται να είναι μεγάλο.

## 4.4 Το πρωτόκολλο LTP

Το πρωτόκολλο LTP (Local Target Protocol) είναι παρόμοιο με το VTRP, εκτός της τρίτης φάσης, δηλαδή της μεταβολής της εμβέλειας μετάδοσης. Όταν λοιπόν δε βρεθεί κανένας κόμβος στην κατεύθυνση του κέντρου ελέγχου, στο LTP χρησιμοποιείται ένας μηχανισμός οπισθοδρόμησης. Έστω η πληροφορία *info($\mathcal{E}$)*, η οποία μεταδόθηκε από τον κόμβο $p$ στον κόμβο $p'$. Ο $p'$ γνωρίζει από ποιον κόμβο δέχθηκε το μήνυμα αυτό και έστω ότι στη συνέχεια δεν βρίσκει κάποιον κόμβο στην κατεύθυνση του $\mathcal{S}$. Τότε, η τρίτη φάση του LTP είναι η ακόλουθη.

**Η φάση οπισθοδρόμησης στο LTP:** Αν η φάση αναζήτησης αποτύχει για έναν ορισμένο αριθμό επαναλήψεων, τότε ο κόμβος $p'$ θα στείλει πίσω στον $p$ ένα μήνυμα αποτυχίας προώθησης και την πληροφορία *info($\mathcal{E}$)*. Αν ο $p$ είναι η πηγή της *info($\mathcal{E}$)*, θα αποφασίσει ότι δεν είναι δυνατή η προώθηση πληροφορίας προς το παρόν και θα διαγράψει την *info($\mathcal{E}$)* από την cache μνήμη του.

## 4.5 Εξομοίωση και συγκριτική αξιολόγηση VTRP και LTP

Προκειμένου να γίνει η σύγκριση μεταξύ VTRP και LTP, έγινε η σχετική υλοποίηση και επέκταση στον εξομοιωτή simDust. Ο συγκεκριμένος εξομοιωτής σχεδιάστηκε για τη διεξαγωγή εξομοιώσεων δικτύων έξυπνης σκόνης και ένας από τους βασικούς στόχους κατά τη σχεδίασή του ήταν η ικανότητα να εξομοιώνονται δίκτυα με χιλιάδες κόμβων. Οι ευρέως χρησιμοποιούμενοι εξομοιωτές όπως ο *ns-2* δεν μπορούν να χειριστούν τόσο μεγάλο πλήθος κόμβων. Η σχεδίαση και υλοποίηση του simDust αναλύεται στα [1] και [2].

Τα κύρια χαρακτηριστικά του simDust είναι τα εξής:

**1. Λειτουργία σε γύρους:** Η λειτουργία του simDust γίνεται σε διακριτούς γύρους. Ένας γύρος αντιπροσωπεύει το χρονικό διάστημα στο οποίο κάποιος κόμβος μπορεί να λάβει ή να μεταδώσει κάποιο μήνυμα και να το επεξεργαστεί.

**2. MAC layer:** Δε γίνεται χρήση κάποιου συγκεκριμένου μοντέλου για το χειρισμό των συγκρούσεων στο επίπεδο MAC. Γίνεται η υπόθεση ότι με κάποιο τρόπο γίνεται η διευθέτησή τους.

**3. Μοντέλο κατανάλωσης ενέργειας:** Το χρησιμοποιούμενο μοντέλο κατανάλωσης ενέργειας είναι αυτό που περιγράφθηκε στην ενότητα 4.1. Θεωρούμε ότι οι κόμβοι μπορούν να μεταβούν περιοδικά σε μια κατάσταση *αδράνειας* (sleep) στην οποία ουσιαστικά δεν καταναλώνεται ενέργεια.

**4. Μέγεθος μηνυμάτων:** Γίνεται η υπόθεση ότι τα μηνύματα έχουν μέγεθος 1 KByte συν 40 bit ως header (ένα πεδίο 32 bit για το ID του κόμβου και 8 bit για τον τύπο του μηνύματος).

## 4.6 Μέτρα εκτίμησης απόδοσης

Έστω τώρα ότι σε μια εκτέλεση του εξομοιωτή συμβαίνουν συνολικά $K$ γεγονότα $(\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_K)$ μέσα στο δίκτυο και τελικά $k$ από αυτά αναφέρονται

επιτυχώς στο κέντρο ελέγχου (δηλαδή φτάνει σε αυτό η πληροφορία). Ορίζουμε το *ποσοστό επιτυχίας* (success rate) ως εξής:

> Το *ποσοστό επιτυχίας* $\mathcal{P}_S$ είναι ο λόγος του πλήθους $k$ των γεγονότων που αναφέρθηκαν επιτυχώς στο κέντρο ελέγχου προς το συνολικό πλήθος $K$ γεγονότων μέσα στο δίκτυο.

Επίσης, μας ενδιαφέρει η συνολική ενέργεια που υπάρχει στο δίκτυο ανά πάσα χρονική στιγμή. Έστω $E_i$ η διαθέσιμη ενέργεια στον κόμβο $i$. Τότε, η συνολική ενέργεια που περιέχεται στο δίκτυο είναι $E_{tot} = \sum_i^n E_i$, όπου $n$ το πλήθος των κομβων του δικτύου. Η συνολική ενέργεια του δικτύου είναι χρήσιμη για την εξαγωγή στατιστικών στοιχείων, αν και από μόνη της δεν μπορεί να δώσει σαφή εικόνα για το τι συμβαίνει στο δίκτυο.

Ακόμα, ένα μέτρο για την εκτίμηση της κατάστασης του δικτύου είναι το πλήθος των ενεργών κόμβων, δηλαδή των κόμβων που δεν έχουν εξαντλήσει ακόμα τα αποθέματα ενέργειάς τους. Εκτός από το πλήθος τους, μας ενδιαφέρει επίσης και η γεωγραφική κατανομή τους και ο χρόνος που εξαντλούνται τα αποθέματα ενέργειάς τους. Συμβολίζουμε το πλήθος των ενεργών κόμβων με $h_A$.

## 4.7 Πειραματικά αποτελέσματα και ανάλυσή τους

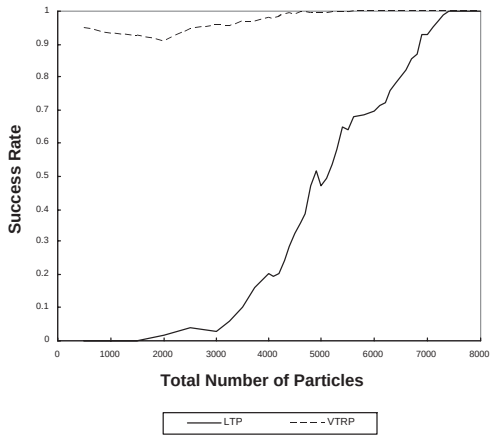
Θέτουμε στον εξομοιωτή ότι έχουμε ένα πεδίο 2000 επί 2000 μέτρα και ότι το πλήθος των κόμβων του δικτύου κυμαίνεται μεταξύ 1000 και 8000 κόμβων. Θέλουμε να μελετήσουμε την επίδραση της πυκνότητας του δικτύου στο ποσοστό επιτυχίας για τα δύο πρωτόκολλα. Οι κόμβοι τοποθετούνται με μια ομοιόμορφη κατανομή στο επίπεδο. Σε κάθε εκτέλεση του πειράματος δημιουργούμε ένα μοναδικό γεγονός διαλέγοντας έναν τυχαίο κόμβο για να το ανιχνεύσει και να μεταδώσει τη σχετική πληροφορία. Θεωρούμε ότι η μέγιστη εμβέλεια μετάδοσης στο VTRP είναι μεγαλύτερη από τη σταθερή εμβέλεια μετάδοσης που χρησιμοποιείται στο LTP, ενώ αρχικά έχουν την ίδια ακριβώς εμβέλεια μετάδοσης.

Τα αποτελέσματα του πρώτου σετ πειραμάτων φαίνονται στο σχήμα 4.2, από τα οποία μπορούν να εξαχθούν τα εξής συμπεράσματα:

- Η πυκνότητα του δικτύου έχει πολύ μεγάλη επίδραση στη λειτουργία του LTP. Πιο συγκεκριμένα, για να αρχίσει το πρωτόκολλο να παραδίδει επιτυχώς την πληροφορία στο κέντρο ελέγχου πρέπει το πλήθος των κόμβων να ξεπεράσει τις 3000.

- Το VTRP αντίθετα, ακόμα και για μικρό, σχετικά, πλήθος κόμβων έχει ποσοστό επιτυχίας κοντά στο 1, αφού ο μηχανισμός μεταβολής της εμβέλειας μετάδοσης ξεπερνά τις δυσκολίες στην προώθηση της πληροφορίας.

Προχωρώντας στο δεύτερο σετ πειραμάτων, προκαλούνται πολλαπλά γεγονότα σε κάθε εκτέλεση του εξομοιωτή. Πιο συγκεκριμένα, σε ένα πεδίο διαστάσεων 2000 επί 2000 μέτρων υπάρχουν $n = 5000$ κόμβοι και σε κάθε γύρο του



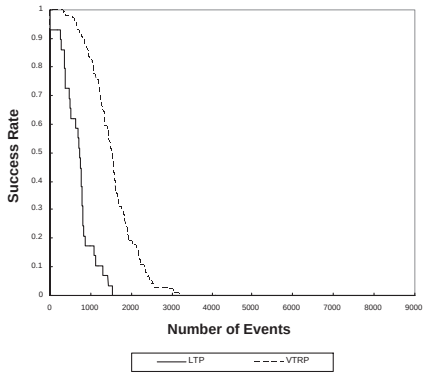

Σχήμα 4.2: Ποσοστό επιτυχίας ($\mathrm{Pr}_s$) για το LTP και το VTRP για διάφορες πυκνότητες δικτύου ($n \in [1000, 8000]$).

εξομοιωτή με κάποια πιθανότητα συμβαίνει ένα μοναδικό γεγονός, το οποίο γίνεται αντιληπτό από κάποιο συγκεκριμένο κόμβο. Αυτό γίνεται μέχρι να έχει δημιουργηθεί ένα σύνολο από 9000 τέτοια γεγονότα. Τα ποσοστά επιτυχίας για αυτό το σετ πειραμάτων φαίνονται στο σχήμα 4.3.

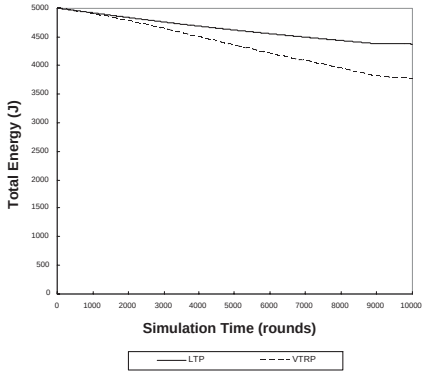
Από τα αποτελέσματα φαίνεται ότι και πάλι το VTRP υπερέχει έναντι του LTP. Τα πειράματα έδειξαν ότι μπορεί να αναφέρει συνολικά μέχρι και 2 φορές περισσότερα γεγονότα σε σχέση με το LTP. Το γεγονός αυτό μπορεί να εξηγηθεί ως εξής: στα δίκτυα έξυπνης σκόνης οι κόμβοι που βρίσκονται κοντα στο κέντρο ελέγχου αναγκαστικά εξαντλούν τα αποθέματα ενέργειάς τους γρηγορότερα από τους υπόλοιπους κόμβους. Αυτό γίνεται γιατί μεταφέρουν την πληροφορία από όλο το υπόλοιπο δίκτυο στο κέντρο ελέγχου, οπότε τους αναλογεί πολύ μεγαλύτερος φόρτος επικοινωνίας από τον μέσο όρο. Στο LTP όταν ''πεθάνουν'' αυτοί οι κόμβοι δεν υπάρχουν άλλοι που να μπορούν να μεταδώσουν απ' ευθείας στο κέντρο ελέγχου οπότε, μοιραία, όλα τα μηνύματα από το δίκτυο χάνονται.

Στο VTRP αντίθετα, ο μηχανισμός μεταβολής της εμβέλειας μετάδοσης μπορεί να προσφέρει λύση, έστω και προσωρινά, στο πρόβλημα αυτό. Το κέντρο θα εξακολουθήσει να λαμβάνει πληροφορία από το δίκτυο μέχρι να εξαντλήσουν τα αποθέματα ενέργειάς τους, με τη σειρά τους, οι κόμβοι που φτάνουν το κέντρο ελέγχου με τη μέγιστη εμβέλεια μετάδοσης.

Ο τρόπος που ''πεθαίνουν'' πρόωρα οι κόμβοι που βρίσκονται κοντά στο κέντρο ελέγχου στο LTP, φαίνεται στο σχήμα 4.5. Όσο προχωρούμε στο πεδίο του χρόνου, βλέπουμε ότι η πυκνότητα του δικτύου αρχίζει να αραιώνει γύρω από το κέντρο ελέγχου. Όταν πλέον δεν υπάρχει κόμβος με το κέντρο ελέγχου μέσα στην εμβέλεια μετάδοσής του, το κέντρο ελέγχου σταμάτα να λαμβάνει μηνύματα. Παράλληλα, το VTRP επιτυγχάνει καλύτερα αποτελέσματα κατανα-




Σχήμα 4.3: Ποσοστό επιτυχίας ($\mathrm{Pr}_s$) για το LTP και το VTRP για πολλαπλά γεγονότα ($n = 5000$).



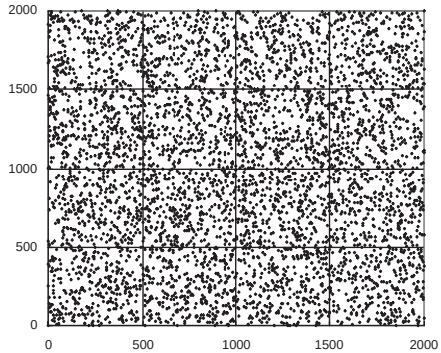

Σχήμα 4.4: Συνολική ενέργεια ($E_{tot}$) για το LTP και το VTRP για πολλαπλά γεγονότα ($n = 5000$).

λώνοντας λίγη παραπάνω ενέργεια, όπως φαίνεται στο σχήμα 4.4.

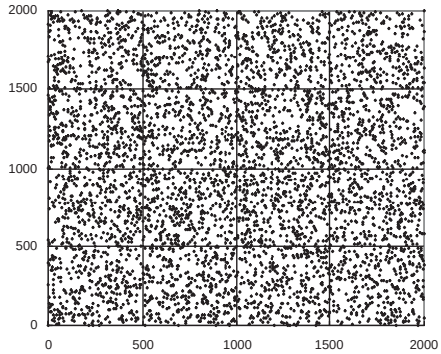
Μια πιο ξεκάθαρη εικόνα για τον αριθμό και τη γεωγραφική θέση δίνουν τα σχήματα 4.7 και 4.8, στα οποία φαίνεται το πλήθος των ενεργών κόμβων σε συνάρτηση με την απόσταση από το κέντρο ελέγχου. Στα σχήματα αυτά, οι κόμβοι του δικτύου έχουν χωριστεί σε 32 ομάδες, βάση του διαχωρισμού του επιπέδου σε 32 τμήματα πάνω στη γραμμή που ενώνει το σημείο (0,0) με το σημείο (2000, 2000). Παρατηρούμε ότι οι κόμβοι που βρίσκονται κοντά στο κέντρο ελέγχου σταδιακά πεθαίνουν, αλλά μόνο στο VTRP συνεχίζουν να πεθαίνουν κόμβοι μέχρι το τέλος του πειράματος ενώ στο LTP το πλήθος των κόμβων μετά από κάποια στιγμή παραμένει πρακτικά το ίδιο.

Στο τελευταίο σετ πειραμάτων, γίνεται μια σύγκριση μεταξύ των τεσσάρων

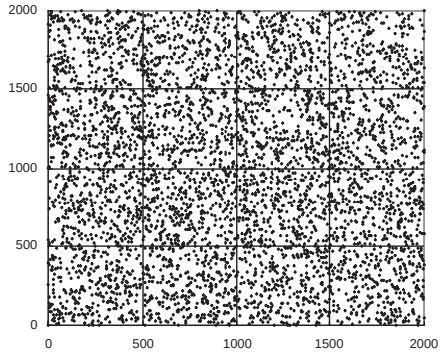
παραλλαγών του VTRP. Τα αποτελέσματα αυτού του σετ φαίνονται στα σχήματα 4.9-4.14. Χρησιμοποιούνται πάλι 5000 κόμβοι και ένα πεδίο 2000 επί 2000 μέτρα στο οποίο προκαλούνται 9000 γεγονότα. Όσον αφορά στο ποσοστό επιτυχίας, από το σχήμα 4.9 φαίνεται ότι η βασική παραλλαγή του VTRP είναι η λιγότερο αποδοτική. Αυτό μάλλον συμβαίνει γιατί απαιτούνται αρκετά βήματα μέχρι να αλλάξει σε ικανοποιητικό βαθμό η εμβέλεια μετάδοσης και να βρεθούν νέοι κόμβοι-γείτονες, σε σχέση με τις άλλες τρεις παραλλαγές, οι οποίες εμφανίζουν παρόμοια μεταξύ τους συμπεριφορά. Τέλος, στα σχήματα 4.11-4.14 φαίνεται η κατανομή των ενεργών κόμβων σε σχέση με την απόσταση από το κέντρο ελέγχου.



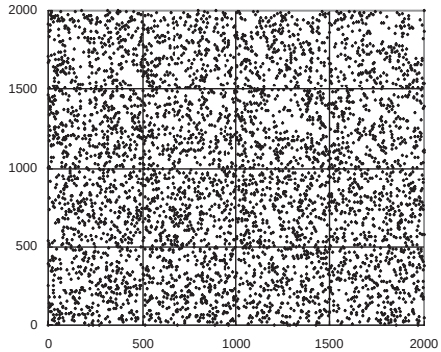

$t = 1$




$t = 2500$

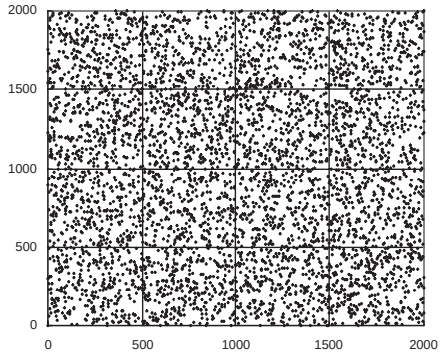



$t = 5000$

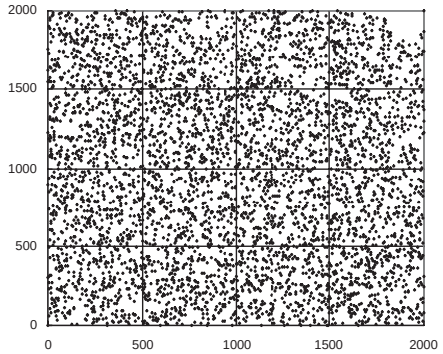



$t = 7500$

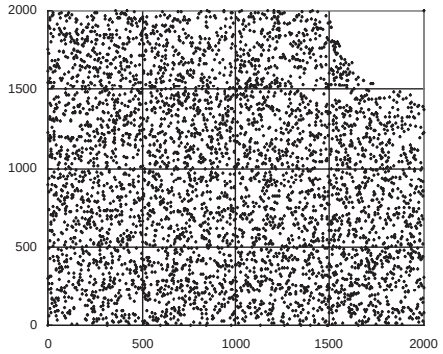
Σχήμα 4.5: Στιγμιότυπο του δικτύου που δείχνει το πλήθος των ενεργών κόμβων κατά την εκτέλεση του LTP σε συνάρτηση με το χρόνο ($n = 5000$).



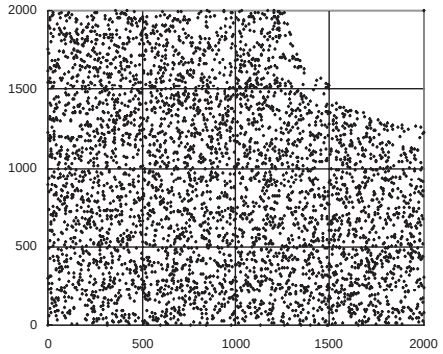

$t = 1$




$t = 2500$

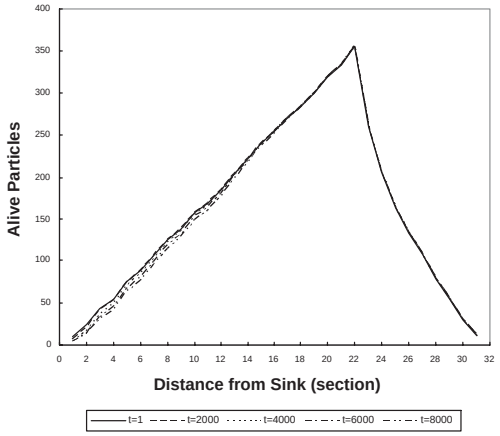



$t = 5000$

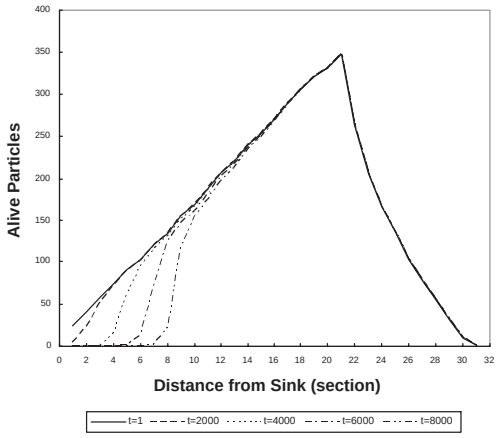



$t = 7500$

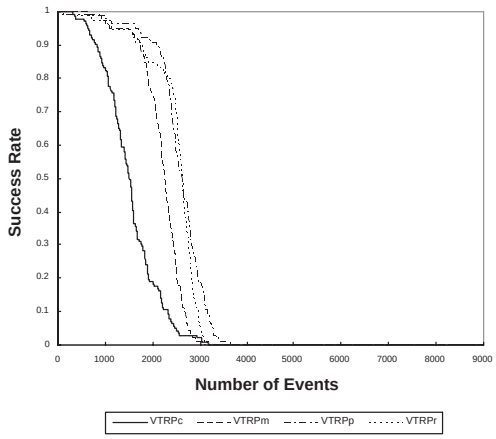
Σχήμα 4.6: Στιγμιότυπο του δικτύου που δείχνει το πλήθος των ενεργών κόμβων κατά την εκτέλεση του VTRP σε συνάρτηση με το χρόνο ($n = 5000$).



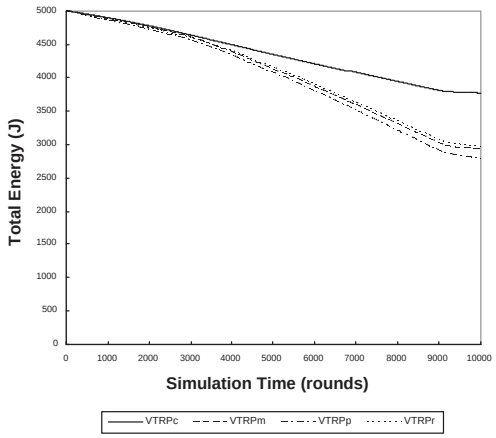

Σχήμα 4.7: Πλήθος ενεργών κόμβων ($h_A$) για το LTP σε συνάρτηση με το χρόνο ($n = 5000$).



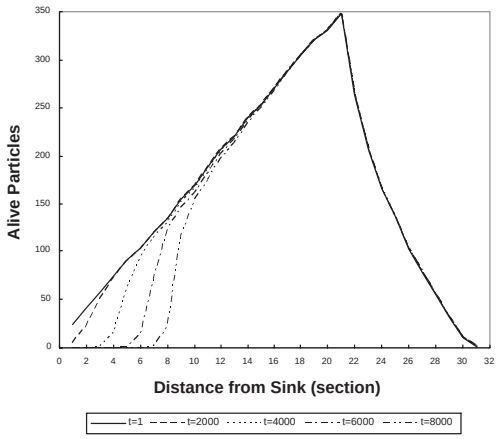

Σχήμα 4.8: Πλήθος ενεργών κόμβων ($h_A$) για το VTRP σε συνάρτηση με το χρόνο ($n = 5000$).



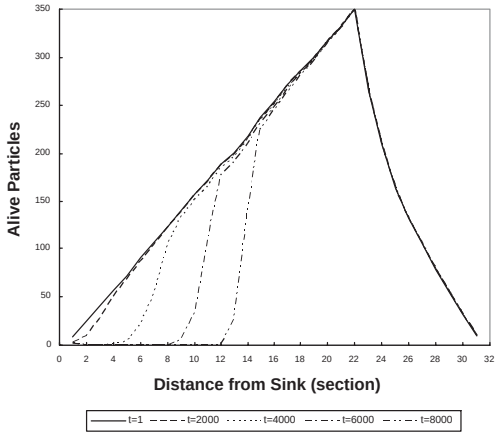

Σχήμα 4.9: Ποσοστό επιτυχίας ($\mathrm{Pr}_s$) για τις παραλλαγές του VTRP για πολλαπλά γεγονότα ($n = 5000$).



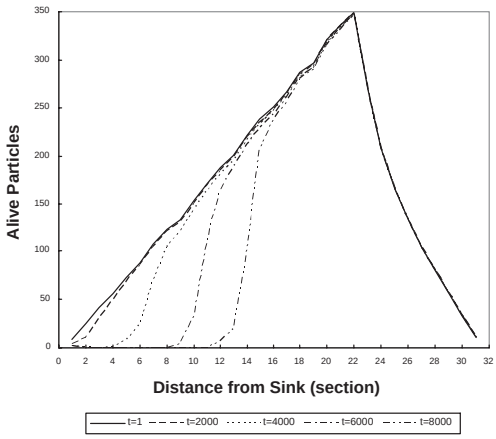

Σχήμα 4.10: Συνολική ενέργεια ($E_{tot}$) για τις διάφορες παραλλαγές του VTRP με πολλαπλά γεγονότα ($n = 5000$).



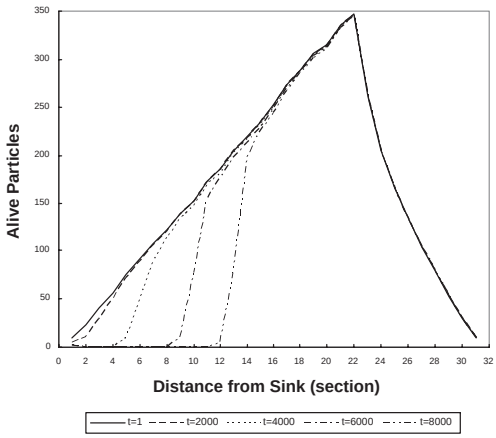

Σχήμα 4.11: Πλήθος ενεργών κόμβων ($h_A$) για το $\mathsf{VTRP_c}$ σε συνάρτηση με το χρόνο ($n = 5000$).




Σχήμα 4.12: Πλήθος ενεργών κόμβων ($h_A$) για το $\mathsf{VTRP_m}$ σε συνάρτηση με το χρόνο ($n = 5000$).




Σχήμα 4.13: Πλήθος ενεργών κόμβων ($h_A$) για το $\mathsf{VTRP_p}$ σε συνάρτηση με το χρόνο ($n = 5000$).




Σχήμα 4.14: Πλήθος ενεργών κόμβων ($h_A$) για το $\mathsf{VTRP_r}$ σε συνάρτηση με το χρόνο ($n = 5000$).

# Μέρος ΙΙΙ

# Το σύστημα jWebDust

# Κεφάλαιο 5

# Η αρχιτεκτονική του jWebDust

Το περιβάλλον jWebDust [8] έχει σαν στόχο να βοηθήσει τον τελικό χρήστη να υλοποιήσει μια εφαρμογή για ένα δίκτυο ασύρματων αισθητήρων εύκολα και γρήγορα. Συνοπτικά, το JWebDust:

- Παρέχει στον χρήστη ένα γενικό και επεκτάσιμο περιβάλλον, το οποίο περιέχει εργαλεία με τα οποία μπορεί να ρυθμίσει τις λειτουργίες που επιτελεί ο κάθε κόμβος του ασύρματου δικτύου αισθητήρων και να επεξεργαστεί τις πληροφορίες που έρχονται μέσα από το δίκτυο.

- Οι κόμβοι του ασύρματου δικτύου αισθητήρων τρέχουν λογισμικό που παρέχει ένα πλήθος από υπηρεσίες στο χρήστη, μέσω των οποίων μπορεί να εκμεταλλευτεί τις δυνατότητες των κόμβων ενός τέτοιου δικτύου. Ανάλογα με την εφαρμογή, ο χρήστης επιλέγει ποιες από αυτές τις υπηρεσίες θα χρησιμοποιήσει σε κάθε κόμβο.

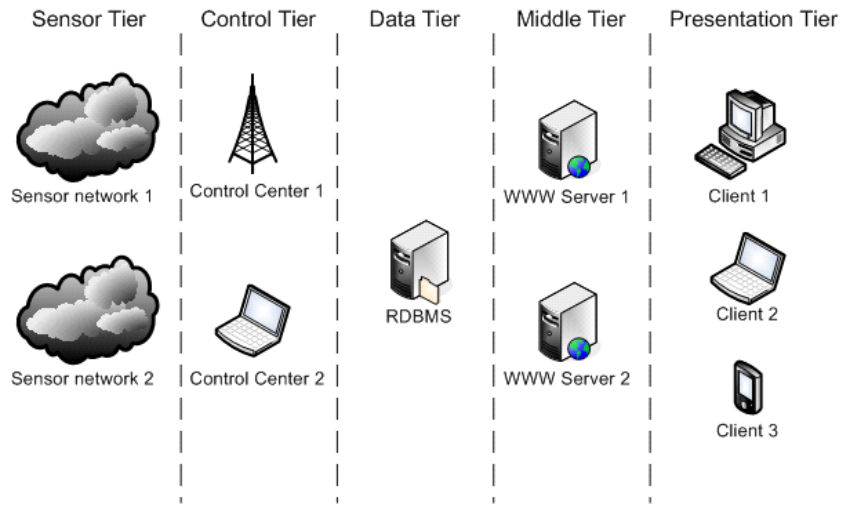
- Όλα αυτά γίνονται μέσω ένος απλού και ευέλικτου interface που παρέχεται στο χρήστη, με το οποίο δίνεται έμφαση στον τρόπο με τον οποίο παρουσιάζονται οι μετρήσεις από το ασύρματο δίκτυο αισθητήρων και με τον οποίο ο χρήστης ρυθμίζει τις παραμέτρους του συστήματος.

- Μειώνεται ο χρόνος και ο κόπος που πρέπει να καταβάλλει κάποιος προκειμένου να δημιουργήσει μια εφαρμογή με ένα ασύρματο δίκτυο αισθητήρων.

Στην ενότητα αυτή θα δούμε την αρχιτεκτονική του jWebDust και τις δυνατότητες που παρέχει στον τελικό χρήστη.

Ο σχεδιασμός της αρχιτεκτονικής του jWebDust έγινε έχοντας κατά νου αρκετά κριτήρια. Πιο συγκεκριμένα, στα κριτήρια αυτά συγκαταλέγονται τα εξής:

1. **Αυτονομία (autonomy)**: θέλουμε τα διάφορα μέρη του συστήματος να λειτουργούν αυτόνομα σε σχέση με τα υπόλοιπα, επιτρέποντας έτσι την




Σχήμα 5.1: Τα επίπεδα της αρχιτεκτονικής του jWebDust.

υλοποίηση των διαφόρων μερών του συστήματος σε όποια πλατφόρμα επιθυμούμε.

2. **Αξιοπιστία (reliability)**: πρέπει το σύστημα να είναι αξιόπιστο, και να βασίζεται σε δοκιμασμένες αρχιτεκτονικές, οι οποίες αυξάνουν την συνολική αξιοπιστία του συστήματος.

3. **Διαθεσιμότητα (availability)**: θέλουμε τα διάφορα μέρη να λειτουργούν σε μεγάλο βαθμό ανεξάρτητα το ένα από το άλλο, ώστε το σύστημα να είναι όσο το δυνατόν διαθέσιμο για μεγαλύτερο χρονικό διάστημα.

4. **Προσαρμοστικότητα (adaptability)**: πρέπει τα μέρη του συστήματος να έχουν οριστεί με τέτοιο τρόπο, ώστε να επιτρέπεται η υλοποίησή τους με διαφορετικές πλατφόρμες.

Η βασική επιλογή που έγινε, όσον αφορά στη σχεδίαση της αρχιτεκτονικής του jWebDust, ήταν να χρησιμοποιηθεί το μοντέλο εφαρμογών **Ν-επιπέδων** (*N-tier application model*). Με τον όρο αυτό εννοούμε το μοντέλο στο οποίο χωρίζουμε μια εφαρμογή σε επιμέρους επίπεδα (*tiers*), τα οποία επικοινωνούν μεταξύ τους με έναν καλά ορισμένο τρόπο και τους έχουν ανατεθεί συγκεκριμένες λειτουργίες. Στην περίπτωση του jWebDust χρησιμοποιούμε μια αρχιτεκτονική 5 επιπέδων.

Τα επίπεδα από τα οποία αποτελείται το jWebDust φαίνονται στο σχήμα 5.1, και είναι:

1. **Το επίπεδο αισθητήρων (Sensor Tier)**, το οποίο αποτελείται από τα δίκτυα έξυπνης σκόνης τα οποία διαχειρίζεται το jWebDust.

2. **Το επίπεδο ελέγχου (Control Tier)**, το οποίο αποτελείται από τα κέντρα ελέγχου για κάθε δίκτυο έξυπνης σκόνης.

3. **Το επίπεδο πληροφορίας (Data Tier)**, το οποίο αποτελείται από μια βάση δεδομένων στην οποία αποθηκεύονται όλες οι πληροφορίες του συστήματος.

4. **Το ενδιάμεσο επίπεδο (Middle Tier)**, το οποίο χρησιμεύει για την επεξεργασία της πληροφορίας που περιέχεται στη βάση δεδομένων και παράλληλα παρέχει ένα interface στα υπόλοιπα επίπεδα, ώστε να έχουν πρόσβαση στη βάση δεδομένων.

5. **Το επίπεδο παρουσίασης (Presentation Tier)**, το οποίο είναι το interface του συστήματος με τον τελικό χρήστη.

Εκτός από την διάκριση σε 5 επίπεδα, υπάρχει και μια επιπρόσθετη διάκριση των μερών του jWebDust σε δύο ομάδες.

- Τα motes, τα οποία χρησιμοποιούν το TinyOS ως λειτουργικό σύστημα.

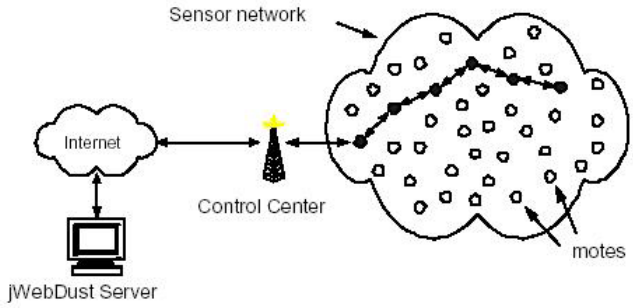
- Το υπόλοιπο σύστημα (π.χ. κέντρο ελέγχου, βάση δεδομένων, application server, κτλ), το οποίο έχει τη δυνατότητα να εκτελέσει κώδικα γραμμένο σε Java.

Και οι δύο αυτές ομάδες χρησιμοποιούν σε μεγάλο βαθμό λογισμικό που έχει ήδη δοκιμαστεί τοιουτοτρόπως (TinyOS, application server, βάση δεδομένων), οι οποίες από τη μία βοηθούν στην καλύτερη αξιοπιστία του όλου συστήματος, και από την άλλη βοηθούν στην απαίτηση για προσαρμοστικότητα. Με άλλα λόγια, για το μέρος του συστήματος που τρέχει κώδικα σε Java, μπορούμε να υλοποιήσουμε κάθε επιμέρους επίπεδο με όποια αρχιτεκτονική και υποσύστημα θέλουμε. Μπορούμε δηλαδή να διαλέξουμε όποια βάση δεδομένων θέλουμε, όποιον application server θέλουμε, κ.ο.κ.

Σε αυτό βοηθά και η οργάνωση του συστήματος σε επίπεδα. Αυτό σημαίνει ότι υπάρχει σαφής διαχωρισμός των λειτουργιών του συστήματος και ορισμός του επιπέδου στο οποίο αντιστοιχεί κάθε τέτοια λειτουργία. Παράλληλα, υπάρχει ένα καλά ορισμένο interface με το οποίο κάθε επίπεδο επικοινωνεί με τα υπόλοιπα. Επιπλέον, όταν χρειαστεί να γίνει κάποια αλλαγή σε ένα ορισμένο επίπεδο, αυτή θα περιοριστεί μόνο στο συγκεκριμένο επίπεδο, χωρίς να επιρρεάσει την υλοποίηση και τη λειτουργία των άλλων επιπέδων.

### 5.0.1 Το επίπεδο αισθητήρων (Sensor Tier)

Το *επίπεδο αισθητήρων* αποτελείται από τα motes που τοποθετούνται μέσα στο πεδίο που ενδιαφερόμαστε να εποπτεύσουμε, μαζί με το λογισμικό που τρέχουν οι συσκευές αυτές και τα σχετικά πρωτόκολλα επικοινωνίας. Καθένα από αυτά




Σχήμα 5.2: Γενική άποψη του επιπέδου αισθητήρων.

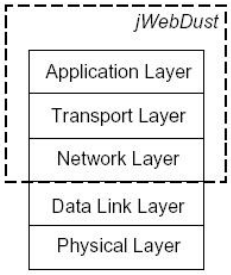
τα motes φέρει ένα πλήθος αισθητήρων και έναν πομποδέκτη για την επικοινωνία με τα άλλα motes. Οι συσκευές αυτές στο σύνολό τους χρησιμοποιούν ασύρματη ραδιοεπικοινωνία.

Η κεντρική ιδέα στη λειτουργία αυτού του επιπέδου είναι ότι καθένα από τα motes, καθοδηγούμενο από κάποιο *κέντρο ελέγχου* (control center), δειγματοληπτεί με κάποια συχνότητα το περιβάλλον στο οποίο βρίσκεται με τους αισθητήρες του. Κατόπιν, βάση κάποιων κριτηρίων αποφασίζει αν θα στείλει αναφορά για τη δραστηριότητά που ανιχνεύει πίσω στο κέντρο ελέγχου. Η προώθηση της πληροφορίας προς το κέντρο ελέγχου γίνεται σε συνεργασία με τα υπόλοιπα motes του δικτύου, δηλαδή μέσω διαδοχικών προωθήσεων από το ένα mote στο άλλο (multihop forwarding). Αυτή η λειτουργία είναι εφικτή μέσω της χρήσης ενός multihop πρωτοκόλλου δρομολόγησης. Όταν φτάσει στο κέντρο ελέγχου η πληροφορία μπορεί να αξιοποιηθεί περαιτέρω. Η δομή του επιπέδου αισθητήρων φαίνεται στο σχήμα 5.2.

Το λογισμικό που εκτελούν τα motes βασίζεται στο λειτουργικό σύστημα TinyOS και χρησιμοποιεί τις βιβλιοθήκες του για να υλοποιήσει την στοίβα πρωτοκόλλων (protocol stack) του jWebDust. Η στοίβα πρωτοκόλλων του jWebDust φαίνεται στο σχήμα 5.3 και περιλαμβάνει τα επίπεδα εφαρμογής, μεταφοράς και δικτύου (application, transport και network layer). Το λογισμικό του κάθε mote υλοποιεί έναν tree based multihop αλγόριθμο δρομολόγησης, ο οποίος περιλαμβάνεται στην επίσημη διανομή του TinyOS, με κάποιες προσθήκες.

Με τον όρο *query* εννοούμε ένα σύνολο από οδηγίες από τον χρήστη προς τους κόμβους του δικτύου για το *πώς* και *πότε* θα δειγματοληπτούν το περιβάλλον, με *ποιους* αισθητήρες θα παίρνουν τα δείγματα αυτά, μαζί με κάποιες συνθήκες για τον τρόπο με τον οποίο θα στέλνουν αναφορές (δηλαδή πώς επιλέγουν τη χρονική στιγμή που θα στείλουν αναφορά) πίσω στο κέντρο ελέγχου με τα δείγματα από τις μετρήσεις που έχουν γίνει.

Γενικά, μπορούμε να ταξινομήσουμε τα query που μπορεί κάποιος να στεί-




Σχήμα 5.3: Η στοίβα πρωτοκόλλων του jWebDust.

λει σε ένα ασύρματο δίκτυο αισθητήρων χρησιμοποιώντας δύο κριτήρια, δηλαδή (*i*) τους κόμβους στους οποίους απευθύνονται, και (*ii*) τον τρόπο με τον οποίο στέλνουν αναφορές στο κέντρο ελέγχου του δικτύου. Η πρώτη κατηγορία περιλαμβάνει τα *mote-specific* και τα *attribute-based* query, ενώ η δεύτερη κατηγορία περιλαμβάνει τα *periodic sensing* και τα *event-driven* query.

Όσον αφορά τα *Mote-based* query, απευθύνονται σε συγκεκριμένους κόμβους, με βάση το ID του κάθε κόμβου. Παράδειγμα ενός τέτοιου περιορισμού είναι το εξής: "Δώσε μου τη θερμοκρασία από το mote με ID 5". Τα *attribute-based* queries αντίθετα, δεν απευθύνονται σε συγκεκριμένα query αλλά σε ολόκληρο το ασύρματο δίκτυο αισθητήρων. Σε αυτή την περίπτωση, μόνο οι κόμβοι που ικανοποιούν κάποια συνθήκη θα απαντήσουν σε ένα τέτοιο query. Παράδειγμα τέτοιου περιορισμού είναι το "δώσε μου τις μετρήσεις θερμοκρασίας από όλους τους κόμβους που έχουν μετρήσει φως πάνω από 200 μονάδες".

Προφανώς, μαζί με τους περιορισμούς με τους οποίους επιλέγουμε ποιοι κόμβοι του δίκτυου θα εκτελέσουν κάποιο query , χρειαζόμαστε και χρονικούς περιορισμούς για το πότε πρέπει να αρχίσει και να σταματήσει η εκτέλεση του query. Τα *periodic-update* queries εισάγουν επιπλέον περιορισμούς, σε σχέση με το πότε στέλνονται αναφορές στο κέντρο ελέγχου του δικτύου. Πιο συγκεκριμένα, ορίζουν ένα χρονικό διάστημα με βάση το οποίο στέλνονται οι αναφορές στο κέντρο ελέγχου. Παράδειγμα τέτοιου περιορισμού είναι το "δώσε μου μετρήσεις θερμοκρασίας κάθε 10 δευτερόλεπτα". Τα *event-driven* queries αντί να εισάγουν τέτοια χρονικά κριτήρια, χρησιμοποιούν την έννοια του *γεγονότος* (events) για να καθορίζουν το πότε θα σταλεί αναφορά στο κέντρο ελέγχου. Παράδειγμα τέτοιου περιορισμού είναι το "όταν η θερμοκρασία αγγίξει τους 100°C".

Αν συνδυάσουμε τους τύπους query που αναφέραμε, έχουμε συνολικά τέσσερις πιθανούς συνδυασμούς:

1. attribute-based periodic update queries,

2. attribute-based event-driven queries,

3. mote-specific periodic update queries,

4. mote-specific event-driven queries.

Στόχος του jWebDust είναι να υποστηρίξει σταδιακά όλους αυτούς τους τύπους query. Παραδείγματα ολοκληρωμένων query είναι τα "δώσε μου μετρήσεις θερμοκρασίας και υγρασίας από όλους τους κόμβους που έχουν φως πάνω από 200 ξεκινώντας από τις 10:15 μέχρι τις 13:30 κάθε 5 λεπτά" και "δώσε μου το επίπεδο του φωτός από τον κόμβο με ID 4 όταν η θερμοκρασία αγγίξει τους 30°C".

Η υλοποίηση του επιπέδου αισθητήρων αφορά στη λειτουργία μερικών πρωτοκόλλων επικοινωνίας με το κέντρο ελέγχου, τα οποία αφορούν στις εξής λειτουργίες:

1. *Διανομή queries από το κέντρο ελέγχου στα motes*: το πρωτόκολλο καθορίζει τα είδη και το format των μηνυμάτων που στέλνονται από το κέντρο ελέγχου στα motes για να δηλώσουν τα queries που ορίζει ο χρήστης του συστήματος.

2. *Αποστολή αναφορών από τα motes προς το κέντρο ελέγχου*: το πρωτόκολλο καθορίζει τα είδη και το format των μηνυμάτων που στέλνονται από τα motes στο κέντρο ελέγχου ως απάντηση στα queries του χρήστη.

3. *Πρωτόκολλο δήλωσης motes*: τα motes χρησιμοποιούν έναν απλό πρωτόκολλο για να δηλώνουν στο κέντρο ελέγχου τον τύπο τους, καθώς και τους αισθητήρες που διαθέτουν.

4. *Χρονικός συγχρονισμός*: κάθε mote διαθέτει ένα ρολόι συστήματος για να καταγράφει την ακριβή ώρα που παίρνει μετρήσεις. Το ρολόι αυτό πρέπει να ειναι συγχρονισμένο με το αντίστοιχο ρολόι του κέντρου ελέγχου.

5. *Αναφορά κατάστασης mote*: επιθυμούμε να γνωρίζουμε αν κάποιο mote είναι ακόμα ενεργό, αν έχει εξαντλήσει τα αποθέματα ενέργειάς του, κτλ. Ένα απλό πρωτόκολλο φροντίζει για αυτά τα θέματα.

6. *Μεταβολή εμβέλειας μετάδοσης*: ανάλογα με τις συνθήκες που επικρατούν στο δίκτυο, το κάθε mote ενδέχεται να αποφασίσει να αλλάξει την εμβέλεια μετάδοσής του.

### 5.0.2 Το επίπεδο ελέγχου (Control Tier)

Το *επίπεδο ελέγχου* αποτελείται από τα *κέντρα ελέγχου* για το κάθε δίκτυο έξυπνης σκόνης. Τα κέντρα ελέγχου ενεργούν ως πύλες (gateways) μεταξύ του επιπέδου αισθητήρων και των υπολοίπων επιπέδων, δηλαδή του δικτύου έξυπνης σκόνης και του υπόλοιπου κόσμου. Τα καθήκοντα ενός κέντρου ελέγχου συνοψίζονται στα παρακάτω:

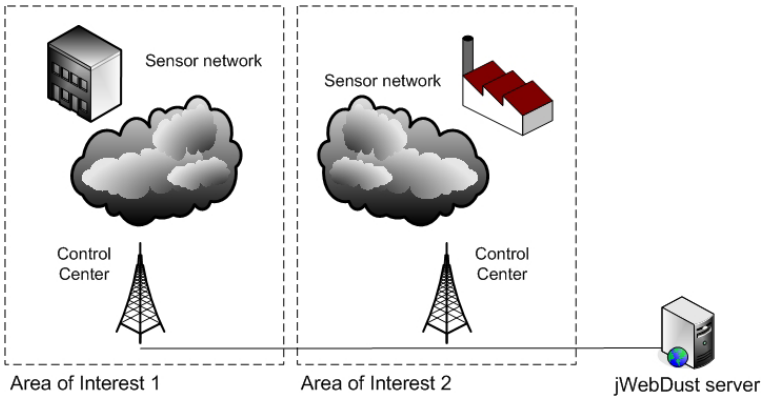
- Προώθηση των queries που θέλει να κάνει ο χρήστης στο δίκτυο έξυπνης σκόνης, στο επίπεδο αισθητήρων.
- Προώθηση των μετρήσεων που έρχονται από το επίπεδο αισθητήρων ως απάντηση στα queries που έχουν σταλεί στο δίκτυο, προς το επίπεδο πληροφορίας (Data Tier). Στην πραγματικότητα, δεν προωθούνται άμεσα στο επίπεδο πληροφορίας, αλλά μεσολαβεί το Ενδιάμεσο επίπεδο (Middle Tier) για την ενημέρωση της βάσης δεδομένων του συστήματος.
- Διαχείριση του δίκτυου έξυπνης σκόνης. Με άλλα λόγια αυτό σημαίνει την υλοποίηση του μέρους του κέντρου ελέγχου όσον αφορά στις διαδικασίες χρονικού συγχρονισμού των motes, δήλωσής τους, αναφορών της κατάστασής τους, κτλ.

Στο επίπεδο συσκευών, ένα κέντρο ελέγχου αποτελείται από ένα PC (laptop ή desktop) και ένα mote το οποίο είναι προσαρτημένο σε αυτό (συνδεδεμένο μέσω κάποιας σειριακής ή USB θύρας). Το PC διατηρεί μια δικτυακή σύνδεση σε μια βάση δεδομένων, η οποία αποτελεί στην ουσία το επίπεδο πληροφορίας (data tier). Προφανώς, η βάση δεδομένων μπορεί να βρίσκεται στο ίδιο PC με το κέντρο ελέγχου αν το επιθυμεί ο χρήστης. Το προσαρτημένο mote χρησιμεύει για την επικοινωνία σε φυσικό επίπεδο με τα motes του δικτύου έξυπνης σκόνης και απλά προωθεί μηνύματα από και προς το κέντρο ελέγχου. Εκτός από κάποιο PC, μπορεί να χρησιμοποιηθεί κάποια embedded πλατφόρμα ως κέντρο ελέγχου, όπως η πλατφόρμα *StarGate* [24] της Intel (βλ. αντίστοιχο κεφάλαιο στην εισαγωγή).

Ένα χρήσιμο και σημαντικό χαρακτηριστικό του επιπέδου ελέγχου του jWebDust είναι ότι επιτρέπει την λειτουργία του ακόμα και στην περίπτωση που διακοπεί για κάποιο σύντομο χρονικό διάστημα η σύνδεση με το επίπεδο πληροφορίας. Αυτό είναι εφικτό μέσω δύο μηχανισμών:

1. Η προώθηση των queries στο επίπεδο αισθητήρων δεν γίνεται με την απ' ευθείας αποστολή τους από το επίπεδο πληροφορίας στα κέντρα ελέγχου, αλλά μέσω περιοδικών ερωτήσεων από τα κέντρα ελέγχου στο επίπεδο πληροφορίας του συστήματος για νέα queries. Αν υπάρχουν τέτοια, θα ληφθούν από το κέντρο ελέγχου και θα προωθηθούν μέσω του προσαρτημένου mote προς το επίπεδο αισθητήρων. Αν δεν είναι εφικτή η σύνδεση του κέντρου ελέγχου με το επίπεδο πληροφορίας, θα επαναληφθεί η διαδικασία ερωτήσεων μετά από κάποιο χρονικό διάστημα.
2. Το κέντρο ελέγχου διαθέτει έναν buffer για να αποθηκεύει τις μετρήσεις που έρχονται από το επίπεδο αισθητήρων στην περίπτωση που δεν είναι εφικτή η σύνδεση με το επίπεδο πληροφορίας. Όταν επανέλθει η σύνδεση, τότε θα προωθηθούν και οι μετρήσεις.

Η σημασία του χαρακτηριστικού αυτού οφείλεται στο γεγονός ότι το κέντρο ελέγχου μπορεί να συνδέεται ασύρματα με τα υπόλοιπα επίπεδα του συστήμα-




Σχήμα 5.4: Δύο δίκτυα έξυπνης σκόνης αντιμετωπίζονται από τα ανώτερα επίπεδα του συστήματος ως ένα "εικονικό" δίκτυο έξυπνης σκόνης.

τος, το οποίο με τη σειρά του εμπεριέχει την πιθανότητα πρόσκαιρων διακοπών στη σύνδεση μαζί τους.

Τέλος, ένα χαρακτηριστικό του επιπέδου ελέγχου του jWebDust που το διαφοροποιεί σε σχέση με τα υπόλοιπα συστήματα για δίκτυα έξυπνης σκόνης, είναι η δυνατότητα διαχείρισης πολλαπλών δικτύων μέσω των αντίστοιχων πολλαπλών κέντρων ελέγχου (ένα για το κάθε δίκτυο). Μέσω του συστήματος ο τελικός χρήστης βλέπει όλα τα δίκτυα έξυπνης σκόνης ως ένα ενιαίο *"εικονικό"* (virtual) δίκτυο. Για κάθε δίκτυο έξυπνης σκόνης και το αντίστοιχο κέντρο ελέγχου υπάρχει ένα μοναδικό ID, οπότε με αυτόν τον τρόπο μπορεί το σύστημα να ξεχωρίζει ποια motes ανήκουν σε ποιο δίκτυο, να αποθηκεύει σωστά τις μετρήσεις από motes με το ίδιο ID αλλά από διαφορετικά δίκτυα, κτλ.

### 5.0.3 Το επίπεδο πληροφορίας (Data Tier)

Το *επίπεδο πληροφορίας* είναι το μέρος όπου το σύστημα αποθηκεύει όλη την πληροφορία που απαιτείται για τη λειτουργία του ως σύνολο. Ουσιαστικά αποτελείται από μια σχεσιακή βάση δεδομένων, η οποία περιέχει έναν αριθμό από πίνακες, ο ορισμός των οποίων ικανοποιεί ένα σχεσιακό σχήμα για το jWebDust. Το σχήμα της βάσης του jWebDust είναι γραμμένο σε ANSI SQL (SQL89), οπότε αυτόματα μπορεί να μεταφερθεί σε οποιαδήποτε βάση δεδομένων. Στη δική μας υλοποίηση, χρησιμοποιήθηκε η βάση δεδομένων *Postgres* [22].

Οι πληροφορίες που χρειάζεται να αποθηκευτούν στο επίπεδο πληροφορίας είναι σχετικές με:

- Τα queries που θέλουμε να σταλούν στο δίκτυο έξυπνης σκόνης.
- Τους τύπους των motes και των αισθητήρων που *μπορούν να υπάρξουν* μέσα στο δίκτυο έξυπνης που διαχειριζόμαστε με το jWebDust.

- Το πλήθος και τους τύπους των motes και των αισθητήρων που *υπάρχουν* μέσα σε αυτό το δίκτυο έξυπνης σκόνης.

- Δεδομένα σχετικά με τη διαχείριση του δικτύου, όπως πότε έστειλε δεδομένα κάποιος συγκεκριμένος κόμβος για τελευταία φορά.

- Τις μετρήσεις που στέλνουν τα motes ως απάντηση στα queries που έχει υποβάλλει ο χρήστης του συστήματος.

Από τη μια μεριά, η συμβολή του επιπέδου πληροφορίας είναι καθοριστική για τη λειτουργία του συστήματος, αφού στην ουσία καθορίζει τις δυνατότητές του. Είναι το σημείο όπου συναντιούνται .

Από την άλλη, το επίπεδο πληροφορίας δεν διεκπεραιώνει κάποια σημαντικά καθήκοντα, όπως ίσως θα περίμενε ο χρήστης του συστήματος, τα οποία έχουν ανατεθεί σε άλλα επίπεδα. Αυτά τα καθήκοντα περιλαμβάνουν την προώθηση queries και μετρήσεων από και προς το επίπεδο ελέγχου, την ευθύνη των οποίων έχει αναλάβει το κέντρο ελέγχου στο επίπεδο ελέγχου, και την μετατροπή των δεδομένων που περιέχονται στις μετρήσεις των motes σε μονάδες κατανοητές στον τελικό χρήστη, καθήκον το οποίο έχει αναλάβει το ενδιάμεσο επίπεδο.

Το σχεσιακό σχήμα της βάσης δεδομένων αποτελείται από 10 πίνακες, οι οποίοι μπορούν να χωριστούν σε τρεις κατηγορίες, ανάλογα με τη λειτουργία την οποία υπηρετούν:

1. *Σχετικά με τα motes*: αυτοί οι πίνακες περιέχουν πληροφορίες για τους τύπους motes και αισθητήρων που μπορούν να υπάρχουν μέσα στο δίκτυο, τα motes που υπάρχουν μέσα στο δίκτυο, τα δίκτυα έξυπνης σκόνης που διαχειρίζεται το σύστημα, τους αισθητήρες του κάθε mote και το δίκτυο στο οποίο ανήκει.

2. *Σχετικά με τα queries*: αυτοί οι πίνακες περιέχουν τους τύπους query που μπορεί ο χρήστης να υποβάλλει στο σύστημα (και κατ' επέκταση στο δίκτυο έξυπνης σκόνης), τα queries που έχει ήδη υποβάλλει, τα motes και τους αισθητήρες τους οποίους αφορούν.

3. *Σχετικά με τις μετρήσεις από το δίκτυο έξυπνης σκόνης*: αποθηκεύονται οι μετρήσεις που παίρνουμε από το δίκτυο έξυπνης σκόνης για να μπορούμε να τις ανακτήσουμε και επεξεργαστούμε αργότερα.

Η διαχείριση όλων αυτών των πινάκων γίνεται μέσω του interface που προσφέρει το ενδιάμεσο επίπεδο, όπως εξηγείται στην επόμενη ενότητα, οπότε ο τελικός χρήστης δε χρειάζεται να προγραμματίσει ή να ασχοληθεί με την SQL προκειμένου π.χ. να προσθέσει νέους τύπους motes, αισθητήρων, κτλ.

### 5.0.4 Το ενδιάμεσο επίπεδο (Middle Tier)

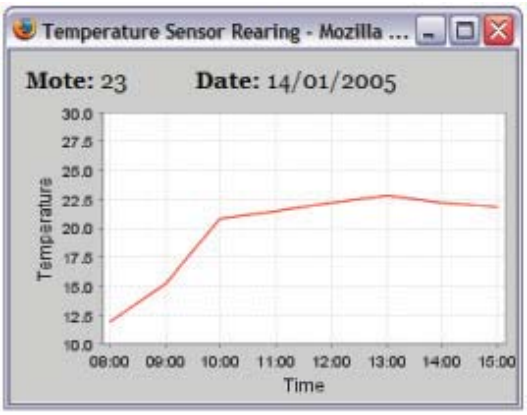
Το *ενδιάμεσο επίπεδο* είναι αυτό το οποίο υλοποιεί τη λογική που ενώνει όλα τα υπόλοιπα επίπεδα σε ένα σύστημα. Το βασικό καθήκον του επιπέδου αυτού, όπως φανερώνει το όνομά του, είναι να προσφέρει ένα interface με το οποίο να αλληλεπιδρούν τα άλλα επίπεδα με το επίπεδο πληροφορίας, και πιο συγκεκριμένα, το επίπεδο ελέγχου και το επίπεδο παρουσίασης με τη βάση δεδομένων. Αντί να χρησιμοποιήσουν απ' ευθείας τη βάση δεδομένων, τα επίπεδα αυτά επικοινωνούν με το ενδιάμεσο επίπεδο και:

- Το επίπεδο ελέγχου παραδίδει τις μετρήσεις από το επίπεδο αισθητήρων στο ενδιάμεσο επίπεδο και δεν νοιάζεται για το πώς θα αποθηκευθούν στη βάση δεδομένων.
- Το επίπεδο παρουσίασης προμηθεύεται δεδομένα από τη βάση χωρίς να νοιάζεται για τις λεπτομέρειες της μετατροπής τους σε μονάδες καταληπτές από το χρήστη ή για το πώς υλοποιούνται προσθήκες και γενικά αλλαγές στο σύστημα.

Το ενδιάμεσο επίπεδο αποτελείται από κάποιες συνιστώσες λογισμικού ( Java *servlets*), οι οποίες μπορούν να θεωρηθούν ως αυτόνομες εφαρμογές, και οι οποίες εκτελούνται σε έναν *application server*. Καθένα από τα servlets έχει αυστηρά καθορισμένη λειτουργία και εκτελούνται ανεξάρτητα το ένα από το άλλο, με σκοπό να επεξεργαστούν δεδομένα που περιέχονται στο επίπεδο πληροφορίας και να παράξουν κάποιο output (όπως, μια σελίδα HMTL με μετρήσεις από το δίκτυο έξυπνης σκόνης).

Ένας application server είναι ένας web server, ο οποίος έχει την ικανότητα να εκτελεί κώδικα γραμμένο σε Java. Η υλοποίηση του ενδιάμεσου επιπέδου με servlets έχει ως αποτέλεσμα τον καλύτερο ορισμό των υπερεσιών που προσφέρει και την καλύτερη κατανομή του φόρτου εργασίας του επιπέδου συνολικά. Επίσης, το επίπεδο παρουσίασης μπορεί με τον τρόπο αυτό να λειτουργεί παράλληλα και ανεξάρτητα από το ενδιάμεσο επίπεδο.

Ο κώδικας για το ενδιάμεσο επίπεδο βασίζεται από τη μια στους πίνακες που περιέχονται στη βάση δεδομένων του συστήματος και από την άλλη στη λειτουργικότητα που θέλουμε να υλοποίησουμε στο κάθε επίπεδο. Με άλλα λόγια, υπάρχουν κλάσεις που αντιστοιχούν στους πίνακες που περιέχονται στη βάση, άλλες κλάσεις που αναλαμβάνουν τη διαχείρισή τους και κάποιες άλλες κλάσεις που αναλαμβάνουν την διασύνδεση με τη βάση μέσω των προηγούμενων κλάσεων. Η διαχείριση παρόμοιων τύπων γίνεται από μία κλάση για τους τύπους αυτούς, π.χ. υπάρχει μια κλάση για τη διαχείριση των τύπων (motes, αισθητήρων, queries, κτλ) συνολικά. Τα servlet που υλοποιούν το interface με τα άλλα επίπεδα παρέχουν ένα καλά ορισμένο σύνολο μεθόδων για το σκοπό αυτό. Όλες αυτές οι μέθοδοι δέχονται παραμέτρους από το επίπεδο παρουσίασης και το επιπεδο ελέγχου. Εφόσον το interface είναι ορισμένο, είναι δυνατόν να υπάρξουν αλλαγές στην υλοποίηση του ενδιάμεσου επιπέδου, χωρίς να υπάρξουν αλλαγές στα υπόλοιπα επίπεδα.



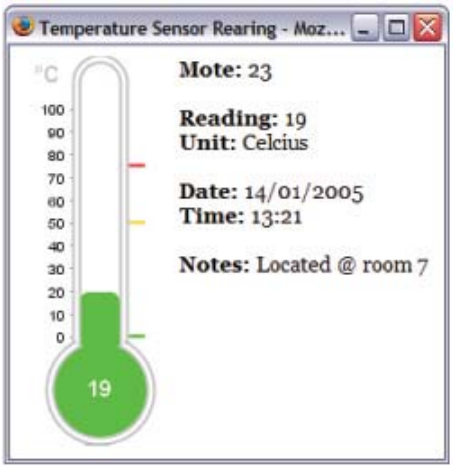

Σχήμα 5.5: Παράθυρο που απεικονίζει τις μετρήσεις θερμοκρασίας σε συνάρτηση με το χρόνο από κάποιον κόμβο του δικτύου.

Ένα επιπρόσθετο όφελος από την αρχιτεκτονική του ενδιάμεσου επιπέδου είναι το ακόλουθο. Αν είχε ακολουθηθεί μια προσέγγιση στην οποία θα υπήρχε μόνο μία συνιστώσα λογισμικού για να αναλάβει όλη τη λειτουργικότητα του επιπέδου στην περίπτωση που θέλαμε να αλλάξουμε π.χ. το χρώμα που εμφανίζεται στο background ενός πίνακα κατόπιν κάποιας αίτητης από το επίπεδο παρουσίασης, θα έπρεπε να επαναμεταγλωττίσουμε τον κώδικα για όλο το επίπεδο. Αντίθετα, στην παρούσα αρχιτεκτονική τέτοιες αλλαγές αφορούν μικρό μέρος του κώδικα, και πιο συγκεκριμένα κάποιο servlet το οποίο υλοποιεί τη λογική για τη συγκεκριμένη λειτουργία, όποτε απαιτείται επαναμεταγλώττιση μόνο του συγκεκριμένου servlet.

### 5.0.5 Το επίπεδο παρουσίασης (Presentation Tier)

Το *επίπεδο παρουσίασης* ουσιαστικά είναι το interface με τον τελικό χρήστη και είναι το επίπεδο του συστήματος με το οποίο θα χρειαστεί να αλληλεπιδράσουν οι τελικοί χρήστες ως επί το πλείστον έπειτα από την έναρξη λειτουργίας του. Το επίπεδο παρουσίασης μπορεί να υλοποιηθεί κατά πολλούς διαφορετικούς τρόπους, ανάλογα με το περιβάλλον του τελικού χρήστη. Χρησιμοποιούνται οι όροι rich και thin client, ανάλογα με τις δυνατότητες του περιβάλλοντος του χρήστη. Π.χ. ως rich client μπορεί να χαρακτηριστεί ο web browser ενός σύγχρονου PC, ενώ ως thin client ο browser ενός κινητού τηλεφώνου.

Το επίπεδο παρουσίασης του jWebDust βρίσκεται σε φάση ανάπτυξης την στιγμή που γράφεται αυτή η εργασία. Ένα από τα χαρακτηριστικά του είναι η χρήση *portlets* για την παρουσίαση των αποτελεσμάτων από το δίκτυο. Τα portlet προσφέρουν στον τελικό χρήστη τη δυνατότητα να δημιουργήσει ο ίδιος το interface με το οποίο θα χειρίζεται το ασύρματο δίκτυο αισθητήρων του. Π.χ. ο χρήστης μπορεί να ενδιαφέρεται κάθε φορά που χρησιμοποιεί το σύστημα να βλέπει τις μετρήσεις θερμοκρασίας από κάποιους συγκεκριμένους




Σχήμα 5.6: Παράθυρο που απεικονίζει την τελευταία μέτρηση θερμοκρασίας από κάποιον κόμβο του δικτύου.

κόμβους. Μπορεί να ρυθμίσει το σύστημα ώστε να παρουσιάζεται κατευθείαν αυτή η πληροφορία στην οθόνη του. Δύο παραδείγματα χρήσης portlets φαίνονται στα σχήματα 5.5 και 5.6. Τα portlet χρησιμοποιούνται παράλληλα με ένα standard interface μέσω του οποίου είναι διαθέσιμες οι βασικές διαχειριστικές λειτουργίες του συστήματος.

Μέσω του επιπέδου παρουσίασης, ο χρήστης έχει πρόσβαση σε υπηρεσίες όπως:

1. Αποστολή queries στο δίκτυο έξυπνης σκόνης
2. Διαχείριση των δικτύων έξυπνης σκόνης
3. Προσθήκες στο σύστημα, π.χ. νέοι τύποι αισθητήρων
4. Να επιλέξει μετρήσεις που βρίσκονται στη βάση δεδομένων με διάφορα κριτήρια όπως ID mote, χρόνο αναφοράς, τύπο αισθητήρα, κτλ, και να επιλέξει τη μορφή που θα του παρουσιαστούν τα αποτελέσματα (π.χ. γραφική παράσταση, πίνακας, κ.α.).

Η υλοποίηση του επιπέδου παρουσίασης του jWebDust απευθύνεται προς το παρόν σε rich clients, αν και είναι πιθανό (και εφικτό) να υλοποιηθούν κάποια components για thin clients στο μέλλον. Αυτό γιατί οι rich clients προσφέρουν μεγαλύτερη ευκολία χρήσης του συστήματος (π.χ. έχοντας ένα PC η μεγάλη οθόνη είναι πλεονέκτημα) και μεγαλύτερη ταχύτητα εκτέλεσης και απόκρισης (π.χ. ο browser ενός PC σε σχέση με το browser ενός κινητού τηλεφώνου).

# Κεφάλαιο 6

# Προδιαγραφές λειτουργίας

Στην ενότητα αυτή περιγράφονται οι προδιαγραφές λειτουργίας (*functional specifications*) του πληροφοριακού συστήματος jWebDust, αναλυτικά για κάθε αρχιτεκτονικό επίπεδο, όπως αυτά ορίστηκαν στην προηγούμενη ενότητα. Με τον όρο προδιαγραφές λειτουργίας εννοούμε τους κανόνες που ορίζουν την λειτουργία των διάφορων υποσυστημάτων του κάθε επιπέδου αρχιτεκτονικής του συστήματος, και οι οποίες περιλαμβάνουν περιγραφές των υποσυστημάτων και των λειτουργιών τους, οι οποίες γίνονται με αρκετά γενικό τρόπο, χωρίς να υπεισέρχονται σε πολλές τεχνικές λεπτομέρειες. Οι προδιαγραφές λειτουργίας απευθύνονται:

(i) στον χρήστη του εκάστοτε επιπέδου του συστήματος,

(ii) τον μηχανικό ο οποίος είναι υπέυθυνος για την παραγωγή του λογισμικού του εκάστοτε επιπέδου και ο οποίος μπορεί να διαφέρει, ανάλογα με το κάθε επίπεδο.

Ακολουθεί μια περιγραφή των προδιαγραφών λειτουργίας για κάθε επίπεδο, η οποία παρέχει τις παραπάνω πληροφορίες, μαζί με κάποιο σενάριο σχετικά με τον χρήστη κάθε τέτοιου επιπέδου. Το σενάριο έχει ως στόχο να βοηθήσει στην κατανόηση του κειμένου και όχι να υποδείξει συγκεκριμένες λειτουργίες.

Καθώς ο σχεδιασμός του συστήματος είναι ακόμα σε εξέλιξη (αν και βρίσκεται σε αρκετά προχωρημένο στάδιο για αυτή τουλάχιστον την έκδοση), ενδεχομένως να γίνουν κάποιες αλλάγες ή προσθήκες στις προδιαγραφές λειτουργίας που περιλαμβάνονται σε αυτή την ενότητα.

## 6.1 Προδιαγραφές λειτουργίας του επιπέδου αισθητήρων (Sensor Tier)

**Ορισμός:** Το *επίπεδο αισθητήρων* αποτελείται από τα motes που τοποθετούνται μέσα στο πεδίο που ενδιαφερόμαστε να εποπτεύσουμε, μαζί με το λογισμικό που τρέχουν οι συσκευές αυτές και τα σχετικά πρωτόκολλα επικοινωνίας. Καθένα

από αυτά τα motes φέρει ένα πλήθος αισθητήρων και έναν πομποδέκτη για την επικοινωνία με τα άλλα motes. Οι συσκευές αυτές επικοινωνούν μεταξύ τους χρησιμοποιώντας ασύρματη ραδιοεπικοινωνία.

**Περιγραφή:** Η κεντρική ιδέα στη λειτουργία αυτού του επιπέδου είναι ότι καθένα από τα motes, καθοδηγούμενο από κάποιο *κέντρο ελέγχου* (control center), δειγματοληπτεί με κάποια συχνότητα το περιβάλλον στο οποίο βρίσκεται με τους αισθητήρες του. Το πότε γίνεται η δειγματοληψία του περιβάλλοντος από τους αισθητήρες των motes καθορίζεται από τα queries που στέλνονται από το κέντρο ελέγχου. Κάθε mote που λαμβάνει ένα τέτοιο query δειγματοληπτεί με κάποια συχνότητα το περιβάλλον του. Κατόπιν, βάση κάποιων κριτηρίων αποφασίζει αν θα στείλει αναφορά για τη δραστηριότητά του πίσω στο κέντρο ελέγχου. Η προώθηση της πληροφορίας προς το κέντρο ελέγχου γίνεται μέσω των υπολοίπων motes του δικτύου (multihop forwarding). Από το κέντρο ελέγχου η πληροφορία μπορεί να αξιοποιηθεί περαιτέρω.

Η λειτουργία αυτού του επιπέδου επίσης περιλαμβάνει τον χρονικό συγχρονισμό με το κέντρο ελέγχου, τη δήλωση των motes στο κέντρο ελέγχου, καθώς και τη δήλωση της κατάστασής τους και τη μεταβολή της εμβέλειας μετάδοσής τους. Στη συνέχεια της ενότητας αυτής παρουσιάζονται αναλυτικά οι δραστηριότητες αυτές.

**Σενάριο:** έστω ένα mote $x$, το οποίο θέλει να φανεί χρήσιμο στο δίκτυο έξυπνης σκόνης στο οποίο βρίσκεται. Γνωρίζει ότι διαθέτει κάποιους συγκεκριμένους αισθητήρες, ότι ανήκει σε κάποιο συγκεκριμένο τύπο mote και ότι πρέπει να ακούει εισερχόμενα μηνύματα που περιέχουν queries, σχετικά με μετρήσεις που πρέπει να σταλούν σε κάποιο συγκεκριμένο mote, το κέντρο ελέγχου, το οποίο έχει ID SINK_ID (μια σταθερά γνωστή σε όλα τα motes του δικτύου).

Ακολουθεί μια περιγραφή των σημαντικότερων υποσυστημάτων από τα οποία πρέπει να αποτελείται το λογισμικό του mote $x$ (έχοντας πάντα υπόψιν ότι τις υπόλοιπες λειτουργίες τις αναλαμβάνει το TinyOS [29]).

### 6.1.1 Διαχειριστής multihop δρομολόγησης (multihop routing manager)

**Ορισμός:** Ο διαχειριστής δρομολόγησης στο mote $x$ είναι αυτός μέσω του οποίου γίνεται η αποστολή και λήψη μηνυμάτων από το δίκτυο και είναι αυτός ο οποίος αναλαμβάνει να καθορίσει ποιο είναι το επόμενο βήμα (mote) στη δρομολόγηση ενός πακέτου προς το κέντρο ελέγχου.

**Λειτουργία:** Το λογισμικό του mote περιέχει ένα διαχειριστή, ο οποίος αναλαμβάνει την υλοποίηση ενός multihop πρωτοκόλλου δρομολόγησης μέσα στο δίκτυο έξυπνης σκόνης. Ο διαχειριστής του multihop πρωτοκόλλου δρομολόγησης είναι επιφορτισμένος με τα εξής καθήκοντα (βάση πάντα του πρωτοκόλλου δρομολόγησης το οποίο υλοποιεί):

1. Εύρεση γειτονικών motes μέσα στην εμβέλεια μετάδοσης του mote.

2. Δημιουργία και διατήρηση ενός πίνακα δρομολόγησης. Η ενημέρωση του πίνακα γίνεται δυναμικά, βάση των μετρικών που χρησιμοποιεί το εκάστοτε πρωτόκολλο δρομολόγησης και χρησιμοποιείται την αποστολή και προώθηση μηνυμάτων μέσα στο δίκτυο. Η ενημέρωση του πίνακα θέλουμε να γίνεται δυναμικά για να μπορεί να ανταποκρίνεται στις αλλαγές που πιθανόν συμβαίνουν μέσα στο δίκτυο έξυπνης σκόνης (π.χ. καινούρια motes στην περιοχή, motes που είναι καταχωρημένα στον πίνακα δρομολόγησης αλλά έχουν πάθει κάποια βλάβη, κ.ο.κ.).

3. Προώθηση (*forwarding*) μηνυμάτων που λαμβάνει το mote και δεν απευθύνονται σε αυτό (έχουν δηλαδή άλλο παραλήπτη).

4. Λήψη και αποστολή μηνυμάτων στο δίκτυο έξυπνης σκόνης.

Με τον όρο "γειτονικά motes" εννοούμε τα motes που βρίσκονται μέσα στην εμβέλεια μετάδοσης του πομποδέκτη του εκάστοτε mote και θα λάβουν οποιοδήποτε μήνυμα μετάδωσει το mote αυτό. Εκτός κι αν το δίκτυο είναι πολύ μικρό ή τα motes έχουν δυνατότητες μετάδοσης σε μεγάλες αποστάσεις, δεν μπορούν όλα τα motes του δικτύου να είναι γείτονες μεταξύ τους.

Η προώθηση των μηνύματων που δεν αφορούν το συγκεκριμένο mote γίνεται χωρίς την εμπλοκή του υπόλοιπου λογισμικού. Αυτό γιατί θεωρούμε ότι δεν γίνεται κάποιου είδους data aggregation στο σύστημα και επιλέγουμε να αγνοήσουμε την πληροφορία που περιέχουν τα μηνύματα αυτά. Με τον όρο data aggregation εννοούμε τη διαδικασία κατά την οποία πληροφορία που περιέχεται σε διαφορετικά μηνύματα μπορεί να συνεκτιμηθεί με κάποιον αλγόριθμο, και κατόπιν να προωθηθεί ένα μήνυμα που να περιέχει την συνεπτυγμένη πληροφορία, αντί για πολλαπλά ξεχωριστά μηνύματα. Έγινε η επιλογή να μην υλοποιηθεί αυτό το χαρακτηριστικό, γιατί από τη μια απαιτείται η σχεδίαση και υλοποίηση ειδικών για το σκοπό αυτό αλγορίθμων, το οποίο συνεπάγεται αυξημένη πολυπλοκότητα στην επεξεργασία των μηνυμάτων, και από την άλλη για τις απλές περιπτώσεις δικτύων έξυπνης σκόνης αυτό το χαρακτηριστικό δεν είναι απαραίτητο.

Όσον αφορά στο υπόλοιπο λογισμικό του mote, ο διαχειριστής αυτός παρέχει δύο μηχανισμούς στο για να σταλεί κάποιο μήνυμα:

1. Χωρίς τη χρήση συγκεκριμένου παραλήπτη, δηλαδή να κάνει broadcast κάποιο μήνυμα. Ο μηχανισμός αυτός χρησιμεύει για την ενημέρωση του κέντρου ελέγχου σχετικά με την κατάσταση του κόμβου (βλέπε και υποενότητα 6.1.5).

2. Με χρήση συγκεκριμένου παραλήπτη, ο οποίος είναι πάντοτε το control center. Ο διαχειριστής πρέπει να επιλέγει πάντοτε το επόμενο βήμα στη δρομολόγηση, δηλαδή τον γείτονα στον οποίο θα σταλεί το μήνυμα και μέσω του οποίου θα προχωρήσει προς το control center, χωρίς να ξέρει το υπόλοιπο λογισμικό του mote ποιο είναι αυτό.

Τέλος, ο διαχειριστής πρέπει να παρέχει ένα interface στο υπόλοιπο λογισμικό, μέσω του οποίου θα είναι εφικτή η επανεκκίνηση της διαδικασίας εύρεσης γειτόνων ανά πάσα στιγμή. Αυτή η δυνατότητα είναι χρήσιμη για την περίπτωση που αποφασίσει το mote να αλλάξει την εμβέλεια μετάδοσής του, οπότε θα πρέπει να ενημερώσει τον πίνακα δρομολόγησής του για να περιλαμβάνει τους νέους γείτονες που μπορεί να προκύψουν ή να βγάλει γείτονες που πλέον δεν είναι μέσα στην εμβέλεια μετάδοσης του mote.

**Παρεχόμενα interfaces**

- **Initialize**: Ο διαχειριστής αρχίζει να εκτελεί τις διαδικασίες που προβλέπονται από τον αλγόριθμο δρομολόγησης.
- **SendMessage(messageBuffer)**: Αποστολή μηνύματος προς το κέντρο ελέγχου (δεν απαιτείται διεύθυνση παραλήπτη). Περνάμε ένα buffer που περιέχει το μήνυμα προς αποστολή ως παράμετρο. Ο buffer αυτός έχει το μέγεθος του μέγιστου επιτρεπτού μηνύματος (TOS_MSG_SIZE).
- **BroadcastMessage(messageBuffer)**: Αποστολή μηνύματος στους γείτονες του mote. Με άλλα λόγια μετάδοση μηνύματος χωρίς παραλήπτη. Όποιος είναι μέσα στην εμβέλεια του mote και έχει ανοικτό τον πομποδέκτη του, θα λάβει το μήνυμα αυτό.
- **ReceiveMessage(messageBuffer)**: Περνά τον buffer που χρησιμεύει για την αποθήκευση των λαμβανόμενων μηνυμάτων.
- **ResetRoutingTable**: Ο διαχειριστής πρέπει να αρχίσει ξανά τη διαδικασία δρομολόγησης.

### 6.1.2 Διαχειριστής Query (query manager)

**Προαπαιτούμενα:** Να έχει ξεκινήσει τη λειτουργία του ο διαχειριστής δρομολόγησης, να έχει ολοκληρωθεί η διαδικασία δήλωσης του mote (βλ. αντίστοιχη υποενότητα) και να έχει γίνει χρονικός συγχρονισμός.

**Ορισμός:** Ο διαχειριστής query στο mote $x$ αναμένει για νέα queries από το δίκτυο. Όταν λάβει ένα μήνυμα που περιέχει κάποιο query, ο διαχειριστής αναλαμβάνει την εκτέλεσή του, καθώς και την αποστολή αναφορών στο κέντρο ελέγχου.

**Λειτουργία:** Τα είδη των queries (από αυτά που ορίσαμε στο κεφάλαιο της αρχιτεκτονικής του jWebDust) που μπορούμε να υποβάλλουμε μέσω του jWebDust σε αυτή την έκδοση του συστήματος είναι:

- mote-specific periodic update query
- mote-specific event-based query

Το λογισμικό του mote γνωρίζει τους τύπους των υποστηριζόμενων query και με βάση αυτούς αποφασίζει το πώς θα τα διαχειριστεί.

Ο διαχειριστής των queries πρέπει να υλοποιεί έναν *διαιτητή*, βάση του οποίου να γίνεται η εκτέλεση των queries. Θεωρούμε ότι μπορούμε να έχουμε πολλαπλά queries, αλλά το πλήθος τους είναι περιορισμένο λόγω των περιορισμένων δυνατοτήτων των motes. Ορίζουμε το πλήθος των ταυτόχρονων query σε MAX_NUMBER_OF_QUERIES. Βάζουμε ένα πάνω όριο στο πλήθος των ταυτόχρονων queries, γιατί οι δυνατότητες των mote είναι περιορισμένες και θεωρούμε ότι ένας μεγάλο πλήθος από queries θα επηρεάσει αρνητικά τη λειτουργία ολόκληρου του δικτύου (κατανάλωση ενέργειας, πολλά μηνύματα για προώθηση, κτλ).

Ο διαιτητής βάζει σε μια ουρά τις μετρήσεις που πρέπει να γίνουν ανάλογα με το κάθε query, και αποφασίζει:

- Αν υπάρχουν queries που απαιτούν μετρήσεις από τον ίδιο αισθητήρα, να γίνεται μόνο μία φορά η δειγματοληψία και να χρησιμοποιείται αυτή η μέτρηση από τα συγκεκριμένα queries. Π.χ. αν υπάρχουν δύο queries από τα οποία το ένα να θέλει μέτρηση θερμοκρασίας και φωτός κάθε 125 ms και το άλλο μέτρηση θερμοκρασίας και πίεσης κάθε 125 ms, μπορεί να εξοικονομηθεί ενέργεια χρησιμοποιώντας μια κοινή μέτρηση θερμοκρασίας. Ο διαιτητής διατηρεί μια καταχώρηση για μετρήσεις από τον κάθε αισθητήρα και συμβουλεύεται το ρολόι του συστήματος για το αν χρειάζεται να γίνει ξανά μέτρηση ή όχι.

- Ανάλογα με την περίπτωση αν θα σταλεί απάντηση στο κέντρο ελέγχου ή όχι (αν ικανοποιούνται δηλαδή τα κριτήρια που έχει θέσει ο χρήστης με το query ή όχι).

Όλα τα παραπάνω γίνονται αυτόματα από το διαιτήτη και μόνο. Εάν φτάσει κάποιο query και η ουρά του διαιτητή είναι πλήρης, το query αυτό αγνοείται. Επίσης, ο διαχειριστής των queries πρέπει να παρέχει τη δυνατότητα ακύρωσης (cancel) οποιουδήποτε query. Το mote δεν στέλνει κάποιο μήνυμα σε καμία από τις δύο περιπτώσεις. Όταν γίνει μέτρηση από το mote και ικανοποιείται η αντίστοιχη συνθήκη, πρέπει να σταλεί ένα μήνυμα που να περιέχει τη μέτρηση αυτή πίσω στο κέντρο ελέγχου.

Λίγα λόγια τώρα σχετικά με τα πακέτα που χρησιμοποιούνται για τα query που στέλνει το κέντρο ελέγχου και τις μετρήσεις που στέλνουν πίσω οι κόμβοι του δικτύου. Το standard μέγεθος πακέτου στο TinyOS είναι 36 bytes, που σημαίνει το πολύ 29 bytes διαθέσιμα για μεταφορά πληροφορίας (στην πραγματικότητα λιγότερα). Παρ' όλα αυτά, αυτό το μέγεθος πακέτου επαρκεί για τις ανάγκες των query που θέλουμε να χρησιμοποιήσουμε στο σύστημα. Ένα timestamp καταλαμβάνει 5 bytes, ένα moteID 2bytes και τα sensortypeID 1 byte. Τα events και οι περιορισμοί για τα attribute μπορούν και τα δύο να εκφραστούν με τον ίδιο τρόπο, καταλαμβάνοντας 4 bytes:

- 1 byte για τον τύπο του αισθητήρα,
- 1 byte για τη σχέση που χρησιμοποιείται (ίσο, μικρότερο από, μεγαλύτερο από, κτλ) ,
- 2 bytes για την τιμή που χρησιμοποιείται στον περιορισμό.

Επειδή ένα query μπορεί να απαιτεί μετρήσεις από πολλοούς αισθητήρες, είναι πιθανό ότι ένα πακέτο δεν επαρκεί για να χωρέσει όλες τις μετρήσεις για κάποια δεδομένη δειγματοληψία. Στην περίπτωση αυτή απλά χρησιμοποιείται κι ένα δεύτερο μήνυμα, το οποίο περιέχει τις υπόλοιπες μετρήσεις. Επίσης, υπάρχει η περίπτωση που ο χρήστης επιθυμεί να ακύρωσει ένα query που είναι ενεργό. Το κέντρο ελέγχου στέλνει ένα μήνυμα που περιέχει το ID του query προς ακύρωση. Ο διαχειριστής όταν λάβει ένα τέτοιο μήνυμα, διαγράφει το αντίστοιχο query από τα ενεργά query.

**Παρεχόμενα interfaces**

- **Initialize**: Αρχικοποίηση του διαχειριστή (αρχικοποίηση της ουράς με queries, κτλ).
- **InsertQueryInQueue(query)**: Εισαγωγή ενός νέου query προς εκτέλεση στην ουρά των queries.
- **CancelQuery(queryID)**: Παύση εκτέλεσης και διαγραφή κάποιου συγκεκριμένου query από τη μνήμη του mote.

### 6.1.3 Διαχειριστής χρονικού συγχρονισμού (Time synchronization)

**Προαπαιτούμενα:** Να έχει ξεκινήσει τη λειτουργία του ο διαχειριστής δρομολόγησης.

**Ορισμός:** Ο διαχειριστής χρονικού συγχρονισμού ενός mote αναλμβάνει το συγχρονισμό του ρολογιού συστήματος του mote με το ρολόι συστήματος του κέντρου ελέγχου, μέσω κάποιου ειδικού πρωτοκόλλου.

**Λειτουργία:** Το mote πρέπει να είναι χρονικά συγχρονισμένο με το υπόλοιπο δίκτυο για να είναι αξιόπιστα τα timestamp που υπάρχουν στις μετρήσεις που στέλνει στο κέντρο ελέγχου. Ο διαχειριστής χρονικού συγχρονισμού:

- Κρατάει ένα ρολόι συστήματος (system clock) για το mote , το οποίο είναι συγχρονισμένο με το κέντρο ελέγχου.
- Ο συγχρονισμός με το κέντρο ελέγχου γίνεται μέσω της χρήσης κάποιου κατάλληλου πρωτοκόλλου.
- Η διαδικασία του συγχρονισμού πρέπει να επαναλαμβάνεται σε τακτά χρονικά διαστήματα.

Το ρολόι συστήματος του κάθε mote μπορεί να χάσει την ακρίβεια για διάφορους λόγους: π.χ. ένα reset, αρρυθμία στον κρύσταλλο του συστήματος, η CPU δεν ανανεώνει το ρολόι συστήματος ακριβώς στα διαστήματα που πρέπει, κ.α. Επίσης, πρέπει να υπάρχει χρονικός συγχρονισμός για να μπορούν να εκτελεούνται σωστά queries που έχουν χρονικούς περιορισμούς.

**Παρεχόμενα interfaces**

- **Initialize**: Αρχικοποίηση του ρολογιού του mote στην τιμή 0 και έναρξη της διαδικασίας χρονικού συγχρονισμού.
- **GetTime**: Ένα interface για να μπορεί το υπόλοιπο firmware να μαθαίνει την τρέχουσα τιμή του ρολογιού του συστήματος.
- **ResetTime**: Ένα interface για να αρχικοποιείται το ρολόι του συστήματος στο 0, όταν ξεκινά τη λειτουργία του το σύστημα.

### 6.1.4 Διαχειριστής πρωτοκόλλου δήλωσης motes (Mote discovery manager)

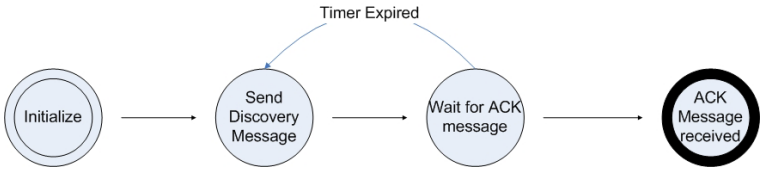
**Ορισμός:** Ο διαχειριστής του πρωτοκόλλου δήλωσης του mote $x$ αναλαμβάνει να ενημερώσει το κέντρο ελέγχου του δικτύου έξυπνης σκόνης για την ύπαρξη του mote, καθώς και για τις δυνατότητές του (πλήθος και είδος αισθητήρων).

**Λειτουργία:** Όταν ξεκινά το mote τη λειτουργία του, πριν να δεχθεί query ή οποιοδήποτε άλλο μήνυμα και αρχίσει την ενεργό συμμετοχή του στο δίκτυο έξυπνης σκόνης, πρέπει να γινει η δήλωση του mote και των δυνατοτήτων του στο κέντρο ελέγχου. Με τον όρο δήλωση του εαυτού του και των δυνατοτήτων του, εννοούμε να στείλει ένα μήνυμα με το mote ID του, μαζί με τα ID των αισθητήρων τους οποίους φέρει.

Το mote επιλέγει ένα τυχαίο χρονικό διάστημα αναμονής και μετά αρχίζει τη διαδικασία δήλωσής του. Το διάστημα αναμονής είναι μοναδικό για κάθε mote και μπορεί να υπολογιστεί χρησιμοποιώντας το mote ID, το οποίο είναι εγγυημένα μοναδικό στο δίκτυο. Αυτή η διαδικασία ακολουθείται προκειμένου να αποφύγουμε την περίπτωση που ενεργοποιούνται πολλά motes σε μικρό χρονικό διάστημα και υπάρχουν συγκρούσεις μεταξύ τους. Εφόσον το διάστημα αναμονής είναι μοναδικό για το κάθε mote μειώνεται η πιθανότητα συγκρούσεων στην έναρξη της λειτουργίας του δικτύου. Στέλνει ένα μήνυμα με την απαραίτητη πληροφορία και περιμένει ένα acknowledge μήνυμα από το κέντρο ελέγχου. Αν δεν έρθει η απάντηση από το κέντρο ελέγχου σε ένα ορισμένο παράθυρο χρόνου (REGISTRATION_TIME), η διαδικασία επαναλαμβάνεται από τον διαχειριστή μέχρι να γίνει σωστά η δήλωση του mote.

**Παρεχόμενα interfaces**

- **Initialize**: Αρχικοποίηση του διαχειριστή με τα στοιχεία που χρειάζεται να δηλώσει.




Σχήμα 6.1: Διάγραμμα ροής για τη λειτουργία του διαχειριστή του πρωτοκόλλου δήλωσης motes.

- **StartMoteRegistration**: Ο διαχειριστής ξεκινά την δήλωση του mote ακολουθώντας τα βήματα του αλγορίθμου που περιγράψαμε πιο πάνω.

Η διαδικασία δήλωσης Motes γίνεται μόνο μια φορά, στο startup του mote. Μόνο αν γίνει reset στο mote θα ξανασταλεί μήνυμα δήλωσης από το ίδιο mote στο κέντρο ελέγχου. Το κέντρο ελέγχου, όπως αναλύεται στη σχετική ενότητα για το επιπεδο ελέγχου, "θυμάται" ποια motes έχουν δηλωθεί στο δίκτυο. Στο σχήμα 6.1 φαίνεται το διάγραμμα ροής για τη λειτουργία του διαχειριστή του πρωτοκόλλου δήλωσης motes.

### 6.1.5 Διαχειριστής κατάστασης mote και εμβέλειας μετάδοσης (Liveness and Transmission range manager)

**Προαπαιτούμενα:** Να έχει τελειώσει η διαδικασία της δήλωσης του mote.

**Ορισμός:** Ο διαχειριστής κατάστασης του mote $x$ είναι υπεύθυνος για την ενημέρωση του κέντρου ελέγχου (υπό κάποιες συγκεκριμένες συνθήκες) για την κατάστασή του, καθώς και για την μεταβολή της εμβέλειας μετάδοσης του πομποδέκτη όταν το επιβάλλουν οι συνθήκες που επικρατούν στο δίκτυο.

**Λειτουργία:** Θέλουμε έναν μηχανισμό τέτοιο ώστε από τη μια να γνωρίζουμε αν τα motes που βρίσκονται μέσα στο δίκτυο έξυπνης σκόνης είναι σε λειτουργία, και από την άλλη να εξασφαλίζουμε τη συνεκτικότητα του δικτύου (δηλαδή να φτάνουν τα μηνύματα από όλα τα motes του δικτύου στο κέντρο ελέγχου και αντίστροφα). Η πρώτη συνθήκη εξασφαλίζεται μέσω ενός απλού πρωτοκόλλου και η δεύτερη μέσω του μηχανισμού αλλαγής της εμβέλεια μετάδοσης.

Αν δεν έχει σταλεί οποιοδήποτε μήνυμα στο κέντρο ελέγχου από το mote μέσα σε κάποιο χρονικό διάστημα (το οποίο ορίζεται ως "NO_ACTION_EXPIRE" και η διάρκεια του οποίου καθορίζεται από το χρήστη ανάλογα με το δίκτυο έξυπνης σκόνης), τότε στέλνεται ένα μήνυμα "I_AM_ALIVE" από το mote προς το κέντρο ελέγχου. Αυτό γίνεται για να γνωρίζουμε αν το κάθε mote βρίσκεται σε λειτουργία και να έχουμε μια πιο σαφή εικόνα του τι συμβαίνει μέσα στο δίκτυο έξυπνης σκόνης.

Όσον αφορά στην εμβέλεια μετάδοσης, το οποίο ανάγεται στο πρόβλημα της συνεκτικότητας του δικτύου, το κέντρο ελέγχου περιοδικά μεταδίδει μήνυματα

"HELLO". Τα motes που βρίσκονται κοντά στο κέντρο ελέγχου λαμβάνουν πρώτα το μήνυμα αυτό και κατόπιν το μεταδίδουν στο υπόλοιπο δίκτυο με μια διαδικασία flooding. Με τον όρο flooding εννούμε μια διαδικασία μετάδοσης μηνύματος σε ολόκληρο το δίκτυο. Όταν ένα mote λάβει ένα τέτοιο μήνυμα το προωθεί σε όλους τους γείτονές του (broadcast). Η προώθηση του ίδιου μηνύματος γίνεται μόνο μια φορά.

Αν κάποιο mote δεν λάβει κανένα μήνυμα (δηλαδή κανονικό ή "HELLO") μέσα σε κάποιο χρονικό διάστημα, θεωρεί ότι έχει αποκοπεί από το υπόλοιπο δίκτυο και αποφασίζει να αλλάξει την εμβέλεια μετάδοσης του για να επανακτήσει τη σύνδεση με το υπόλοιπο δίκτυο έξυπνης σκόνης.

**Παρεχόμενα interfaces**

- **Initialize**: Αρχικοποίηση των δύο μηχανισμών.
- **SendAliveMessage**: Αποστολή ενός μηνύματος τύπου "I_AM_ALIVE" στο κέντρο ελέγχου.
- **BroadcastHelloMessage**: Αποστολή στους γείτονες του mote ενός μηνύματος τύπου "Hello".
- **ChangeTransmissionRange**: Αλλαγή της εμβέλειας μετάδοσης του mote.

Στο σχήμα 6.2 φαίνεται το διάγραμμα ροής για τη λειτουργία του διαχειριστή κατάστασης motes και εμβέλειας μετάδοσης.

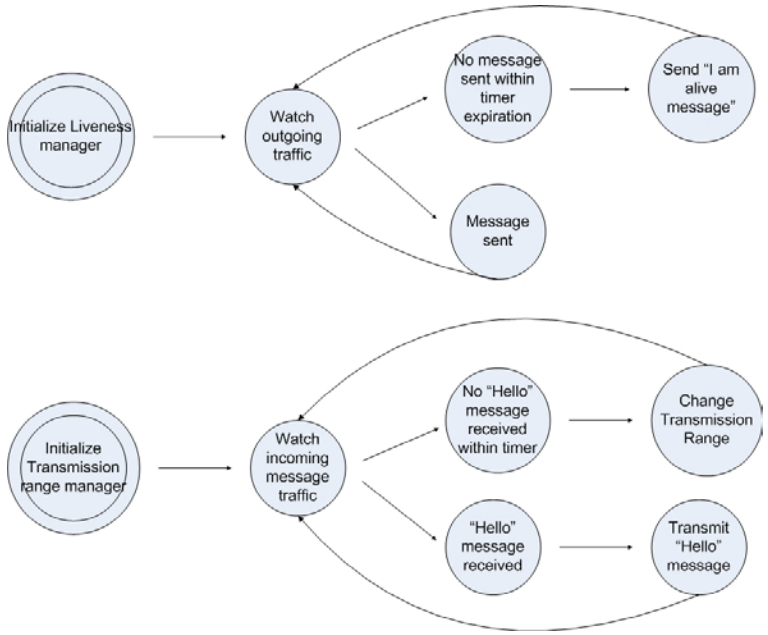
## 6.2 Προδιαγραφές λειτουργίας του επιπέδου ελέγχου (Control Tier)

**Ορισμός:** Το *επίπεδο ελέγχου* αποτελείται από τα *κέντρα ελέγχου* για το κάθε δίκτυο έξυπνης σκόνης. Τα κέντρα ελέγχου ενεργούν ως πύλες (gateways) μεταξύ του επιπέδου αισθητήρων και των υπολοίπων επιπέδων, δηλαδή του δικτύου έξυπνης σκόνης και του υπόλοιπου κόσμου.

**Περιγραφή:** Σε επίπεδο συσκευών, ένα κέντρο ελέγχου αποτελείται από ένα PC (laptop ή desktop) και ένα mote το οποίο είναι προσαρτημένο σε αυτό. Το PC διατηρεί μια δικτυακή σύνδεση σε μια βάση δεδομένων , η οποία αποτελεί στην ουσία το επίπεδο πληροφορίας (data tier). Προφανώς, η βάση δεδομένων μπορεί να βρίσκεται στο ίδιο PC με το κέντρο ελέγχου αν το επιθυμεί ο χρήστης. Το προσαρτημένο mote χρησιμεύει για την επικοινωνία σε φυσικό επίπεδο με τα motes του δικτύου έξυπνης σκόνης και απλά προωθεί μηνύματα από και προς το κέντρο ελέγχου.

Τα καθήκοντα ενός κέντρου ελέγχου συνοψίζονται στα παρακάτω:

- Προώθηση των queries που θέλει να κάνει ο χρήστης στο δίκτυο έξυπνης σκόνης, στο επίπεδο αισθητήρων.




Σχήμα 6.2: Διάγραμμα ροής για τη λειτουργία του διαχειριστή κατάστασης motes και εμβέλειας μετάδοσης.

- Προώθηση των μετρήσεων που έρχονται από το επίπεδο αισθητήρων ως απάντηση στα queries που έχουν σταλεί στο δίκτυο, προς το επίπεδο πληροφορίας (Data Tier). Στην πραγματικότητα, δεν προωθούνται άμεσα στο επίπεδο πληροφορίας, αλλά μεσολαβεί το Ενδιάμεσο επίπεδο (Middle Tier) για την ενημέρωση της βάσης δεδομένων του συστήματος.
- Διαχείριση του δίκτυου έξυπνης σκόνης. Με άλλα λόγια αυτό σημαίνει την υλοποίηση του μέρους του κέντρου ελέγχου όσον αφορά στις διαδικασίες χρονικού συγχρονισμού των motes, δήλωσής τους, αναφορών της κατάστασής τους, κτλ.

**Σενάριο:** Έστω ένα laptop PC με ένα mote προσαρτημένο σε αυτό και έστω ότι βρίσκεται αρκετά κοντά σε ένα δίκτυο έξυπνης σκόνης ώστε να μπορεί να επικοινωνήσει με τα motes που ανήκουν στο δίκτυο αυτό και ότι παράλληλα διαθέτει σύνδεση με το διαδίκτυο (ασύρματη ή ενσύρματη). Το laptop αυτό παίζει το ρόλο του κέντρου ελέγχου του συγκεκριμένου δικτύου έξυπνης σκόνης. Το mote που βρίσκεται προσαρτημένο σε αυτό έχει διαφορετικό firmware από τα motes που βρίσκονται μέσα στο δίκτυο έξυπνης σκόνης και αναλαμβάνει το ρόλο του προωθητή μηνυμάτων από και προς το δίκτυο έξυπνης σκόνης.

Το κέντρο ελέγχου ενός δικτύου έξυπνης σκόνης στο jWebDust χρειάζεται να επικοινωνεί με δύο επίπεδα, το επίπεδο αισθητήρων και το επίπεδο πληρο-

φορίας. Η επικοινωνία με το πρώτο επίπεδο γίνεται μέσω του προσαρτημένου mote και με το δεύτερο επίπεδο μέσω της σύνδεσης με το διαδίκτυο.

Πρέπει να σημειώσουμε ότι δεν αντιμετωπίζουμε περιπτώσεις στις οποίες υπάρχουν motes με το ίδιο mote ID στο ίδιο δίκτυο έξυπνης σκόνης. Αυτό γιατί δεν θέλουμε να εισάγουμε επιπλέον πολυπλοκότητα στο σύστημα και επιπλέον το σύστήμα μπορεί να λειτουργεί και στην περίπτωση που έχουμε διπλά IDs, αν και πιθανόν να εμφανιστούν κάποιες αναντιστοιχίες στις μετρήσεις που αποθηκεύονται στο επίπεδο πληροφορίας.

### 6.2.1 Διαχειριστής επικοινωνίας με το προσαρτημένο mote

**Προαπαιτούμενα:** Να έχει γίνει η δήλωση του κέντρου ελέγχου στο επίπεδο πληροφορίας.

**Ορισμός:** Ο διαχειριστής επικοινωνίας με το προσαρτημένο mote αναλαμβάνει να προωθεί τα πακέτα που λαμβάνει το mote από το δίκτυο έξυπνης σκόνης και να στέλνει πακέτα προς το δίκτυο έξυπνης σκόνης.

**Λειτουργία:** Όσον αφορά στο προσαρτημένο mote, χρειάζεται μόνο ένα απλο interface με το οποίο το κέντρο ελέγχου θα λαμβάνει και θα αποστέλλει μηνύματα μέσω του προσαρτημένου mote από το δίκτυο έξυπνης σκόνης. Στην ουσία πρόκειται για μια εφαρμογή που διαβάζει από κάποια σειριακή θύρα τα πακέτα που στέλνει το προσαρτημένο mote ή να προωθεί στο mote πακέτα που πρέπει να φτάσουν στο επίπεδο αισθητήρων.

**Παρεχόμενα interfaces**

- **SendPacket(Packet)**: Αποστολή πακέτου στο δίκτυο έξυπνης σκόνης.
- **ReceivePacket(Buffer)**: Λήψη των πακέτων από το προσαρτημένο mote σε κάποιο buffer.

### 6.2.2 Δήλωση του κέντρου ελέγχου στο επίπεδο πληροφορίας

**Ορισμός:** Το κέντρο ελέγχου δηλώνει την ύπαρξή του στο επίπεδο πληροφορίας για να μπορεί να συνεργαστεί μαζί του στη συνέχεια.

**Λειτουργία:** Όταν το κέντρο ελέγχου ξεκινήσει τη λειτουργία του πρέπει να δηλώσει στη βάση δεδομένων του συστήματος το δίκτυο έξυπνης σκόνης το οποίο αντιπροσωπεύει. Το network ID που του ανήκει μπορεί να περαστεί ως παράμετρος στο λογισμικό. Το λογισμικό συνδέεται στη βάση δεδομένων μέσω JDBC, κάνει τη δήλωσή του, και αν δεν υπάρχει εγγραφή κάποιου δικτύου έξυπνης σκόνης με το συγκεκριμένο ID ολοκληρώνεται η διαδικασία. Κατόπιν, το κέντρο ελέγχου αρχίζει την κανονική λειτουργία του. Αν δεν έχει περαστεί το network ID ως παράμετρος, το κέντρο ελέγχου παίρνει το ID που θα του επιστρέψει η βάση σε αυτή την περίπτωση.

**Παρεχόμενα interfaces**

- **Initialize**: Συνδεέται με το middle tier για να γίνει η δήλωση του δικτύου έξυπνης σκόνης.
- **RegisterSensorNetwork**: Δήλωση του δικτύου έξυπνης σκόνης σύμφωνα με τον προαναφερθέντα αλγόριθμο.

### 6.2.3 Διαχειριστής πρωτοκόλλου δήλωσης motes (Mote discovery protocol manager)

**Προαπαιτούμενα:** Να έχει δηλωθεί το κέντρο ελέγχου στο επίπεδο πληροφορίας.

**Ορισμός:** Ο διαχειριστής αυτός δηλώνει τα motes από τα οποία λαμβάνει αντίστοιχα μηνύματα δήλωσης, στο επίπεδο πληροφορίας.

**Λειτουργία:** Όπως αναφέρεται στις προδιαγραφές λειτουργίας του επιπέδου αισθητήρων, κάθε mote αφού ξεκινήσει τη λειτουργία του δηλώνει τον εαυτό του και τις δυνατότητές του στο κέντρο ελέγχου για να μπορεί να συμμετέχει στο σύστημα(βλέπε σχετική ενότητα).

Ο διαχειριστής δήλωσης motes στο κέντρο ελέγχου ακούει συνεχώς για τέτοια μηνύματα δήλωσης από τα motes. Όταν λάβει τέτοια μηνύματα από το δίκτυο έξυπνης σκόνης στέλνει ACK πακέτα και κατόπιν επικοινωνεί με το data tier για να γίνει η δήλωση του mote στη βάση δεδομένων. Τα πακέτα αυτά λειτουργούν ως επιβεβαίωση ότι έχει ολοκληρωθεί η δήλωση του mote και μπορεί να αρχίσει κανονικά τη λειτουργία του (να δέχεται queries κτλ).

Για την περίπτωση που γίνει reset σε κάποιο mote και αρχίσει ξανά τη διαδικασία δήλωσής του, κρατάμε ένα πίνακα με τα motes που έχουν ήδη δηλωθεί. Αν έχει δηλωθεί στέλνουμε ένα ACK πακέτο χωρίς να γίνει ξανά δήλωση του mote στη βάση δεδομένων. Επίσης, για την περίπτωση που γίνει reset στο κέντρο ελέγχου όταν ξεκινά τη λειτουργία του και τελειώσει η διαδίκασία δήλωσής του, βλέπει αν υπάρχουν ήδη δηλωμένα motes για το δίκτυο έξυπνης σκόνης που επιβλέπει και τροποποιεί αναλόγως τον πίνακα των δηλωμένων motes. Αν δεν μπορεί να δηλωθεί το mote λόγω προσωρινής απώλειας σύνδεσης με τα ανώτερα επίπεδα του jWebDust, τότε μπαίνει σε μια λίστα η οποία περιέχει motes που πρέπει να δηλωθούν (παράλληλα στέλνουμε ACK μήνυμα στο mote). Αυτό γίνεται για να μην εξαντλείται η ενέργεια των motes από την προσπάθεια να δηλωθούν στο κέντρο ελέγχου.

Να σημειώσουμε ότι δεν διαχειριζόμαστε περιπτώσεις που έχουμε δύο motes με το ίδιο ID στο δίκτυο έξυπνης σκόνης. Οπότε στην περίπτωση που υπάρχουν τέτοια motes θα σταλούν πολλαπλά ACK μηνύματα που θα αφορούν το ίδιο mote ID. Ίσως σε επόμενη έκδοση του συστήματος να αντιμετωπιστεί το ζήτημα αυτό.

**Παρεχόμενα interfaces**

- **Initialize**: Ο διαχειριστής ελέγχει αν υπάρχουν ήδη δηλωμένα motes στη βάση δεδομένων και ενημερώνει τον πίνακά του.

- **ReceiveDiscoveryMessage(buffer)**: Λήψη μηνύματος δήλωσης από κάποιο mote.

- **RegisterMote(moteID)**: Αν δεν έχει ήδη δηλωθεί το συγκεκριμένο mote, δήλωσή του στο επίπεδο πληροφορίας.

- **SendACKMessage(moteID)**: Αποστολή μηνύματος ACK σε κάποιο mote μετά την επιτυχή δήλωσή του στη βάση.

- **InsertIntoMotesHeard(moteID)**: Ενημέρωση του πίνακα με τα δηλωμένα motes.

### 6.2.4 Διαχειριστής χρονικού συγχρονισμού

**Προαπαιτούμενα:** Να έχει δηλωθεί το κέντρο ελέγχου στο επίπεδο πληροφορίας και να έχει αρχίσει τη λειτουργία του ο διαχειριστής δήλωσης motes.

**Ορισμός:** Ο διαχειριστής χρονικού συγχρονισμού περιοδικά αναλαμβάνει να συγχρονίζειτο ρολόι συστήματος του κέντρου ελέγχου με τα ρολόγια συστήματος των motes του δικτύου έξυπνης σκόνης.

**Λειτουργία:** Ο διαχειριστής του χρονικού συγχρονισμού είναι συνολικά υπεύθυνος για το συχρονισμό των ρολογιών ανάμεσα στο επίπεδο ελέγχου και το επίπεδο αισθητήρων. Καταρχάς, πρέπει με κάποιο τρόπο το κέντρο ελέγχου να είναι συχρονισμένο, όσον αφόρα στο χρονικό πεδίο, με τον υπόλοιπο κόσμο και βάση αυτού να συγχρονίζουν τα ρολόγια τους τα motes που βρίσκονται στο αντίστοιχο δίκτυο έξυπνης σκόνης.

Ο διαχειριστής αυτός υλοποιεί τον server του πρωτοκόλλου χρονικού συγχρονισμού που χρησιμοποιείται από τα motes μέσα στο δίκτυο έξυπνης σκόνης. Ανάλογα με το πρωτόκολλο, η διαδικασία μπορεί να ξεκινήσει είτε από το κέντρο ελέγχου είτε από τα motes μέσα στο δίκτυο, όμως το σημείο αναφοράς είναι πάντοτε το κέντρο ελέγχου.

**Παρεχόμενα interfaces**

- **StartTimeSynchProtocol()**: Έναρξη της λειτουργίας του πρωτοκόλλου χρονικού συγχρονισμού. Ανάλογα με το πρωτόκολλο, το κέντρο ελέγχου ακούει για αιτήσεις συγχρονισμού από το δίκτυο έξυπνης σκόνης ή περιοδικά αρχίζει αυτό τη διαδικασία συγχρονισμού.

- **Synchronize()**: Ανάλογα με το χρησιμοποιούμενο πρωτόκολλο, σε κάποια φάση της λειτουργίας του κέντρου ελέγχου πρέπει να γίνει ο συγχρονισμός των ρολογιών συστήματος των motes. Αποστολή μηνύματος που περιέχει την τιμή του ρολογιού συστήματος του κέντρου ελέγχου.

### 6.2.5 Διαχειριστής query

**Προαπαιτούμενα:** Να έχει γίνει δήλωση του κέντρου ελέγχου στο επίπεδο πληροφορίας και να έχουν αρχίσει τη λειτουργία τους οι διαχειριστές δήλωσης motes και χρονικού συγχρονισμού.

**Ορισμός:** Ο διαχειριστής query στο κέντρο ελέγχου είναι υπεύθυνος για την προώθηση των query που θέλει να κάνει ο χρήστης στο επίπεδο αισθητήρων.

**Λειτουργία:** Για να περάσουν στο επίπεδο αισθητήρων τα query που κάνει ο χρήστης μέσω του jWebDust και βρίσκονται αποθηκευμένα στη βάση δεδομένων του συστήματος (data tier), πρέπει να ζητηθούν από το κέντρο ελέγχου. Με άλλα λόγια δεν στέλνονται τα queries από τα ανώτερα επίπεδο στο επίπεδο ελέγχου. Έτσι λοιπόν, ο διαχειριστής των query στο επίπεδο ελέγχου κάνει polling στη βάση δεδομένων του jWebDust, θυμάται το τελευταίο query ID που έχει πάρει και παίρνει τα νέα queries για να τα προωθήσει στο δίκτυο έξυπνης σκόνης.

Η διαδικασία αυτή επαναλαμβάνεται ανά τακτά χρονικά διαστήματα, τα οποία μπορεί να καθορίσει ο χρήστης. Ακόμα κι αν χαθεί η σύνδεση με τη βάση δεδομένων για μικρό χρονικό διάστημα, δεν δημιουργείται κάποιο πρόβλημα αφού ο συγκεκριμένος διαχειριστής όταν επανέλθει η σύνδεση θα επιχειρήσει να ανακτήσει τα νέα queries.

**Παρεχόμενα interfaces**

- **Initialize**: Αρχικοποίηση του διαχειριστή.
- **PullQueries**: Αναζήτηση για τυχόν νέα queries μέσω του σχετικού interface.
- **ForwardQuery**: Προώθηση κάποιου query στο δίκτυο έξυπνης σκόνης μέσω του προσαρτημένου mote.

### 6.2.6 Διαχειριστής προώθησης και προσωρινής αποθήκευσης μετρήσεων (Readings buffering and forwarding manager)

**Προαπαιτούμενα:** Να έχει αρχίσει η λειτουργία των υπολοίπων διαχειριστών εκτός του διαχειριστή κατάστασης motes και εμβέλειας μετάδοσης.

**Ορισμός:** Ο διαχειριστής αυτός είναι υπεύθυνος για την προώθηση των μετρήσεων από το επίπεδο αισθητήρων στο επίπεδο πληροφορίας και για την προσωρινή αποθήκευσή τους όταν δεν είναι δυνατή η προώθησή τους.

**Λειτουργία:** Εδώ ακολουθείται η αντίστροφη λογική με τον προηγούμενο διαχειριστή. Εδώ επιθυμούμε να προωθήσουμε (forward) τις μετρήσεις που παίρνουμε από το δίκτυο έξυπνης σκόνης. Χρησιμοποιούμε το interface που δίνει το ενδιάμεσο επίπεδο (Middle tier) για να αποθηκεύσουμε τις μετρήσεις στο επίπεδο πληροφορίας (data tier).

Αν έχει χαθεί η σύνδεση με το ενδιάμεσο επίπεδο, ο διαχειριστής χρησιμοποιεί κάποιο buffer ώστε να αποθηκεύει προσωρινά τις μετρήσεις έως ότου να επανέλθει η σύνδεση και να τις προωθήσει.

**Παρεχόμενα interfaces**

- **Initialize**: Αρχικοποίηση του buffer.
- **ForwardReading**: Προώθηση μιας μέτρησης στη βάση δεδομένων. Αν δεν υπάρχει σύνδεση με το ενδιάμεσο επίπεδο (που αναλαμβάνει το ρόλο του interface) στη συγκεκριμένη περίπτωση, η διαδικασία επαναλαμβάνεται μετά από σύντομο χρονικό διάστημα.

### 6.2.7 Διαχειριστής κατάστασης motes και εμβέλειας μετάδοσης (Liveness and Transmission Range manager)

**Προαπαιτούμενα:** Να έχει αρχίσει η εκτέλεση των υπολοίπων διαχειριστών.

**Ορισμός:** Ο διαχειριστής αυτός υλοποιεί το μέρος του κέντρου έλεγχου όσον αφορά στο στο πρωτόκολλο δήλωσης motes και της μεταβολής της εμβέλειας μετάδοσής τους.

**Λειτουργία:** Ο διαχειριστής αυτός αναλαμβάνει να ενημερώνει τη βάση δεδομένων για την κατάσταση των motes μέσα στο δίκτυο έξυπνης σκόνης. Αυτό περιλαμβάνει την ενημέρωση του αντίστοιχου πέδίου στη βάση δεδομένων κάθε φορά που λαμβάνει κάποιο μήνυμα από ένα mote του δικτύου έξυπνης σκόνης. Το μήνυμα αυτό είναι είτε απάντηση σε κάποιο query είτε ένα "I_AM_ALIVE" μήνυμα.

Επίσης, είναι υπεύθυνος για την ενημέρωση των motes του δικτύου με μηνύματα "hello", όπως αναλύεται στη σχετική ενότητα στις προδιαγραφές λειτουργίας του επιπέδου αισθητήρων. Κάνει broadcast ένα τέτοιο μήνυμα στο δίκτυο έξυπνης σκόνης σε ένα περιοδικό χρονικό διάστημα, το οποίο μεταδίδεται σε όσα motes είναι δυνατόν μέσω flooding. Τα motes ανάλογα με το αν λάβουν ή όχι το μήνυμα αυτό αποφασίζουν για το αν θα αλλάξουν την εμβέλεια μετάδοσής τους.

**Παρεχόμενα interfaces**

- **Initialize**: Αρχικοποίηση του διαχειριστή.
- **UpdateMoteState(moteID)**: Όταν λαμβάνεται οποιοδήποτε μήνυμα (μέτρηση ή "Hello") από κάποιο mote, ανανεώνεται η κατάστασή του στη βάση δεδομένων.
- **BroadcastHelloMessage**: Broadcast μετάδοση ενός μηνύματος του τύπου "Hello" στους γείτονες του προσαρτημένου mote.

## 6.3 Data tier - Η βάση δεδομένων του jWebDust και η οργάνωσή της

Το θέμα της ενότητας αυτής είναι η βάση δεδομένων που χρησιμοποιείται στα πλαίσια του jWebDust. Σκοπός αυτής της βάσης δεδομένων είναι να κρατάμε τις μετρήσεις που παίρνουμε από το δίκτυο έξυπνης σκόνης που διαθέτουμε, ώστε να μπορούμε αργότερα να τις ανακτήσουμε, καθώς και να κρατάμε αποθηκευμένα τα queries που υποβάλλουμε στο δίκτυο έξυπνης σκόνης μέσω του interface του jWebDust, ώστε να μπορούμε να παρακολουθούμε την εξέλιξή τους, δηλαδή το πότε υποβλήθηκαν, τις παραμέτρους τους, σε ποια motes αναφέρονται, τις απαντήσεις που ήρθαν από το δίκτυο έξυπνης σκόνης, κ.ο.κ.

Για την υλοποίηση της βάσης δεδομένων επιλέχθηκε η PostgreSQL[22], μια σχεσιακή βάση δεδομένων, η οποία ξεκίνησε αρχικά από το Πανεπιστήμιο Μπέρκλεϋ στην Καλιφόρνια, και η οποία διανέμεται υπό την άδεια Artistic License (το οποίο πρακτικά σημαίνει ότι η διανομή της είναι δωρεάν και ότι μπορεί να χρησιμοποιήθει για την υλοποίηση οποιασδήποτε εφαρμογής). Είναι διαθέσιμη για πολλές πλατφόρμες, μεταξύ των οποίων Windows και Linux, και επίσης υπάρχουν γραφικά εργαλεία (pgAdmin III[21]), τα οποία κάνουν εύκολη τη διαχείρισή της. Ακόμα, η PostgreSQL εκτός από το γεγονός ότι διατίθεται δωρεάν, διαθέτει σημαντικά πλεονεκτήματα όσον αφορά στην απόδοσή της σε κάποιους τομείς που την καθιστούν ανταγωνιστική πολλών βάσεων δεδομένων που διατίθενται στην αγορά.

Στο σχήμα 6.3 παραθέτουμε το γενικό σχήμα της βάσης δεδομένων σε μορφή UML. Η σχεδίαση της βάσης δεδομένων (όπως και όλοκληρης της αρχιτεκτονικής του jWebDust) έγινε έχοντας υπόψιν τους εξής δύο βασικούς παράγοντες:

- Την υποστήριξη ετερογενών (heterogeneous) δικτύων έξυπνης σκόνης.
- Τη δυνατότητα διαχείρισης πολλαπλών δικτύων έξυπνης σκόνης μέσω του jWebDust.

Αυτοί οι δύο παράγοντες διαφοροποιούν σημαντικά το jWebDust από τις άλλες παραπλήσιες εφαρμογές (όπως τοTinyDB [28] και το MoteView[19]), όσον αφορά στις δυνατότητες διαχείρισης ενός δικτύου έξυπνης σκόνης. Το σχήμα που χρησιμοποιείται σε αυτές τις εφαρμογές είναι πιο "επίπεδο" και δεν επιτρέπει να εκφραστούν αφενός πιο σύνθετες σχέσεις μεταξύ των motes που περιέχονται στο δίκτυο έξυπνης σκόνης και των δυνατοτήτων τους (δηλαδή πλήθος και είδος αισθητήρων) και αφετέρου δεν υπάρχει η δυνατότητα καταχώρισης στη βάση motes τα οποία να ανήκουν σε διαφορετικά δίκτυα έξυπνης σκόνης με έναν ενιαίο και κομψό τρόπο, ώστε να μπορεί ο χρήστης να βλέπει όλους τα motes σαν να ανήκουν σε ένα ενιαίο "εικονικό" (virtual) δίκτυο έξυπνης σκόνης.

Συνολικά, η βάση αποτελείται από 10 πίνακες, στους οποίους περιέχονται πληροφορίες για τα Motes που περιέχονται στα δίκτυα έξυπνης σκόνης τα οποία διαχειριζόμαστε με το jWebDust, για τους αισθητήρες που έχει το καθένα από

τα motes που περιέχονται στα δίκτυα αυτά, για τις μετρήσεις που παίρνουμε, για τα query που κάνουμε στα motes, κτλ. Στο σχήμα φαίνονται οι περιορισμοί για τα πρωτεύοντα και τα ξένα κλειδιά για τον κάθε πίνακα. Ακόμα, τα πεδία που είναι υποχρεωτικά σε κάθε πίνακα εμφανίζονται με έντονη γραμματοσειρά.

Ακολουθεί μία παρουσίαση κάθε οντότητας (πίνακα) του σχήματος της βάσης, μαζί με την αναλυτική περιγραφή των πεδίων που περιέχονται σε καθένα από αυτά. Γενικά, οι πίνακες της βάσης του jWebDust μπορούν να χωριστούν σε τρεις κατηγορίες:

1. Πίνακες σχετικοί με τα motes, δηλαδή τα motes που διαχειρίζεται το σύστημα (`Motes`), οι τύποι των motes που υπάρχουν μέσα στο δίκτυο έξυπνης σκόνης (`MoteTypes`), τα δίκτυα έξυπνης σκόνης που διαχείριζεται το jWebDust (`SensorNetworks`), οι αισθητήρες που έχει το κάθε mote (`MoteSensors`) και οι τύποι των αισθητήρων που περιλαμβάνονται μέσα στο σύστημα (`SensorTypes`).

2. Πίνακες σχετικοί με τα queries που υποβάλλονται στα δίκτυα έξυπνης σκόνης που διαχείριζεται το jWebDust, δηλαδή τα υποβληθέντα queries (`SensorRequests`), οι τύποι των queries (`QueryTypes`), η αντιστοίχιση motes και queries (`MoteQueries`), η αντιστοίχιση αισθητήρων και queries (`SensorQueries`).

3. Ο πίνακας `SensorReadings` με τις ενδείξεις των αισθητήρων που έρχονται από τα motes του δικτύου, ως απάντηση σε κάποιο query που υποβλήθηκε μέσω του jWebDust σε κάποια χρονική στιγμή.

Στη συνέχεια, παρατίθενται οι εντολές SQL για τη δημιουργία και διαγραφή της βάσης δεδομένων, καθώς και κάποια παραδείγματα εντολών SQL για την εισαγωγή δεδομένων στους πίνακες της βάσης του jWebDust (ο χρήστης γενικά διαζειρίζεται το σύστημα μέσω του interface του jWebDust και δεν χρειάζεται να ασχοληθεί με τα ενδότερα του συστήματος, αλλά ενδεχομένως να χρειαστεί να κάνει κάποιες άλλαγες οι οποίες να απαιτούν τη χρήση της SQL). Τέλος, παραθέτουμε μια ενότητα που περιγράφουμε διαδικασίες για να εξακριβώσουμε την ορθή λειτουργία του κώδικά μας (testing) για την εισαγωγή δεδομένων στη βάση.

### 6.3.1 Οι πίνακες της βάσης δεδομένων του jWebDust

#### Motes

Ο πίνακας αυτός περιλαμβάνει εγγραφές για όλα τα motes που υπάρχουν στο δίκτυα έξυπνης σκόνης που διαχείριζεται το jWebDust. Κάθε mote ανήκει σε κάποιο δίκτυο έξυπνης σκόνης, έχει ένα μοναδικό ID μέσα στο δίκτυο αυτό, ενώ μπορεί να υπάρχει κάποια πληροφορία σχετικά με τη θέση του μέσα στο πεδίο στο οποίο βρίσκεται το δίκτυο έξυπνης σκόνης.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| moteID | integer | Το ID του κάθε κόμβου στο δίκτυο (primary key) |
| sensorNetworkID | integer | Το ID του δικτύου έξυπνης σκόνης στο οποίο ανήκει ο κάθε κόμβος (primary key) |
| moteLocation | varchar(40) | Μια περιγραφή της θέσης του mote στο δίκτυο, π.χ. Κτήριο Β. |
| moteTypeID | integer | Ο τύπος mote που αντιστοιχεί στο συγκεκριμένο mote. Οι δυνατές επιλογές είναι mica, mica2, mica2dot, micaZ (βλ. επόμενη παράγραφο). |
| GPS_longitude | varchar(40) | Η θέση του mote στο δίκτυο (προαιρετικό), αν έχουμε GPS στο sensor board |
| GPS_lattitude | varchar(40) | Η θέση του mote στο δίκτυο (προαιρετικό), αν έχουμε GPS στο sensor board |
| status | varchar(10) | Η κατάσταση του κόμβου (alive, asleep, dead, κτλ) |
| lastUpdate | timestamp | Το timestamp για την τελευταία επαφή με τον κόμβο |

**MoteTypes**

Ο πίνακας αυτός περιλαμβάνει εγγραφές για τους τύπους motes που μπορεί να υπάρχουν στο δίκτυο (Mica, Mica2, κτλ). Είναι σημαντικό να κάνουμε διάκριση στους τύπους των motes, αφού δεν έχουν όλοι τις ίδιες δυνατότητες και θέλουμε να έχουμε μια σαφή εικόνα του τι περιέχεται μέσα στο δίκτυο έξυπνης σκόνης, ώστε να εκμεταλλευτούμε τις δυνατότητες του. Ο πίνακας αυτός έχει μια σχέση one-to-many με τον πίνακα Motes (μπορούν να υπάρχουν πολλά ίδιου τύπου motes).

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| moteTypeID | integer | Το ID του τύπου του mote (primary key). |
| moteTypeName | varchar(40) | Ένα string για τον τύπο του mote. Οι δυνατές επιλογές είναι mica, mica2, mica2dot, micaZ. |
| moteTypeDescription | text | Εδώ βάζουμε μια σύντομη περιγραφή του συγκεκριμένου τύπου. |

Οι διαφορετικοί τύποι motes που έχουν κυκλοφορήσει μέχρι σήμερα από την εταιρεία Crossbow, και οι οποίοι μας ενδιαφέρουν, είναι οι:

- mica
- mica2
- mica2dot
- micaZ

Η πρώτη γενιά κόμβων smart dust ήταν οι Rene, οι οποίοι όμως δεν γνώρισαν μεγάλη διάδοση και δεν παρουσιάζουν πρακτικό ενδιαφέρον για μας, αφού δεν χρησιμοποιούνται πλέον από κάποια ερευνητική ομάδα. Ο αναγνώστης που ενδιαφέρεται να μάθει περισσότερα για τις διαφορές μεταξύ των τύπων motes μπορεί να ανατρέξει στα [1],[2],[11],[29].

**SensorNetworks**

Ο πίνακας αυτός περιλαμβάνει εγγραφές για όλα τα δίκτυα έξυπνης σκόνης που διαχειρίζεται το jWebDust. Αυτός ο πίνακας χρησιμοποιείται για να γίνεται εύκολα η διάκριση μεταξύ κόμβων που ανήκουν σε διαφορετικά δίκτυα έξυπνης σκόνης αλλά μπορεί να έχουν το ίδιο moteID.

| **Πεδίο** | **Τύπος δεδομένων** | **Περιγραφή** |
|---|---|---|
| sensorNetworkID | integer | Το ID του δικτύου στο οποίο ανήκει ο κόμβος (primary key) |
| sensorNetworkName | text | Μια περιγραφή του δικτύου |

Ο πίνακας αυτός έχει σχέση one-to-many με τον πίνακα Motes, αφού πολλά motes μπορούν να ανήκουν στο ίδιο δίκτυο έξυπνης σκόνης.

**SensorReadings**

Ο πίνακας αυτός περιλαμβάνει εγγραφές για όλες τις μετρήσεις που λαμβάνονται από το δίκτυο έξυπνης σκόνης που διαχειριζόμαστε με το jWebDust ως απαντήσεις στα queries που υποβάλλουμε.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| ID | integer | Το ID για τις μετρήσεις. Κάθε μέτρηση πρέπει να έχει διαφορετικό ID για να γίνεται η διάκριση μεταξύ τους (primary key) |
| date | timestamp | Η ακριβής ώρα και ημερομηνία στην οποία έγινε η συγκεκριμένη μέτρηση |
| moteID | integer | Το ID του κόμβου του δικτύου από τον οποίο προήλθε η μέτρηση |
| sensorNetworkID | integer | Το ID του δικτύου στο οποίο ανήκει ο κόμβος από τον οποίο προήλθε η μέτρηση |
| sensorTypeID | integer | Ο τύπος του αισθητήρα από τον οποίο έγινε η μέτρηση |
| RawData | long | Η πληροφορία από τη μέτρηση, σε binary μορφή. Η μετάφρασή της σε μορφή που να γίνεται κατανοητή εναπόκειται στην εφαρμογή που χρησιμοποιεί τη βάση. |

Οι πραγματικές μετρήσεις που γίνονται από τους κόμβους του δικτύου πρέπει να μεταφραστούν ανάλογα με τον τύπο του αισθητήρα που χρησιμοποιείται για τη μέτρηση. Έτσι, τα δεδομένα από τη μέτρηση ενός thermistor πρέπει να ερμηνευτούν διαφορετικά από αυτά ενός αισθητήρα φωτός. Επίσης, μερικά ολοκληρωμένα τοποθετούν στην ίδια μέτρηση δεδομένα από διαφορετικούς αισθητήρες, π.χ. βαρομετρική πίεση και θερμοκρασία. Οπότε, είναι απαραίτητο να γνωρίζουμε τον τύπο του αισθητήρα από τον οποίο προήλθε η μέτρηση. Ο πίνακας αυτός έχει μια many-to-one σχέση με τον πίνακα Motes, καθώς πολλές μετρήσεις μπορούν να προέλθουν από το ίδιο mote.

**MoteSensors**

Ο πίνακας αυτός περιλαμβάνει εγγραφές για όλους τους αισθητήρες που μπορεί να έχει ένα mote. Η διάκριση των αισθητήρων γίνεται σύμφωνα με το sensor board που βρίσκεται κάθε τέτοιος αισθητήρας (βλ. επόμενη παράγραφο, SensorTypes).

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| moteID | integer | Το ID του κόμβου στο οποίο αναφερόμαστε (primary key) |
| sensorNetworkID | integer | Το ID του δικτύου στο οποίο ανήκει ο κόμβος (primary key) |
| sensorTypeID | integer | Το ID του αισθητήρα ο οποίος περιλαμβάνεται στον κόμβο που έχει το moteID (primary key σε συνδυασμό με το moteID) |

Ο πίνακας αυτός έχει μια many-to-one σχέση με τον πίνακα motes, καθώς ένα mote μπορεί να φέρει πολλούς αισθητήρες.

**SensorTypes**

Στον πίνακα αυτό έχουμε εγγραφές για τους τύπους αισθητήρων που περιλαμβάνονται σε ένα δίκτυο smart dust, με άλλα λόγια συγκεκριμένες πληροφορίες για τους αισθητήρες που δηλώνονται στον προηγούμενο πίνακα (MoteSensors). Η πληροφορία αυτή είναι απαραίτητη για να γίνει η σωστή μετατροπή των δεδομένων που περιέχονται στις μετρήσεις του πίνακα SensorReadings σε μονάδες κατανοητές από τους χρήστες του συστήματος.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| sensorTypeID | integer | Το ID για τον τύπο αισθητήρα(primary key) |
| sensorTypeName | varchar(40) | Ένα string, το οποίο δίνει το όνομα του αισθητήρα (βλ. ακόλουθη λίστα) |
| sensorTypeDescription | text | Μια αναλυτική περιγραφή του ολοκληρωμένου του αισθητήρα |

Μέχρι στιγμής, οι πιθανές επιλογές που έχουμε είναι οι εξής (η διάκριση βασίζεται στα διαφορετικά sensor boards που κυκλοφορούν επειδή αισθητήρες για τον ίδιο σκοπό, π.χ. thermistors, αποτελούνται από διαφορετικά ολοκληρωμένα σε κάθε sensor board):

- MTS101CA Light sensor (αισθητήρας φωτός)
- MTS101CA Thermistor (αισθητήρας θερμοκρασίας)
- MTS300-310CA Microphone (μικρόφωνο)
- MTS300-310CA Thermistor (αισθητήρας θερμοκρασίας)
- MTS310CA Accelerometer (επιταχυνσιόμετρο 2 αξόνων)

- MTS310CA Magnetometer (μαγνητόμετρο 2 αξόνων)
- MTS300-310CA Light sensor (αισθητήρας φωτός)
- MTS400-420CA Humidity-temperature sensor (αισθητήρας υγρασίας και θερμοκρασίας)
- MTS400-420CA Pressure-temperature sensor (αισθητήρας βαρομετρικής πίεσης και θερμοκρασίας)
- MTS400-420CA Light sensor (αισθητήρας φωτός)
- MTS400-420CA Accelerometer (επιταχυνσιόμετρο 2 αξόνων)
- MTS420CA GPS
- MTS510CA Microphone (μικρόφωνο)
- MTS510CA Light sensor (αισθητήρας φωτός)
- MTS510CA Accelerometer (επιταχυνσιόμετρο 2 αξόνων)

Τα sensor boards που κυκλοφορούν στην αγορά από την Crossbow μέχρι στιγμής είναι τα εξής:

- MTS101CA
- MTS300CA
- MTS310CA
- MTS400CA
- MTS420CA
- MTS510CA

**QueryTypes**

Στον πίνακα αυτό, δηλώνουμε τους πιθανούς τύπους query που μπορούμε να υποβάλλουμε στο δίκτυο μέσω του interface του jWebDust. Δυο πιθανοί τύποι είναι:

- Constant-update queries, δηλαδή queries όπου τα motes του δίκτυο έξυπνης σκόνης στέλνουν τις απαντήσεις τους σε περιοδικό χρονικό διάστημα, το οποίο καθορίζεται από το χρήστη του συστήματος.
- Event-driven queries, δηλαδή queries στα οποία τα motes στέλνουν τις απαντήσεις τους μόνο όταν συμβεί κάποιο προκαθορισμένο γεγονός (event), όπως π.χ. η θερμοκρασία να ανέβει πάνω από τους 30°C.

Ο χρήστης μπορεί να προσθέσει τους δικούς του τύπους query, οι οποίοι θα έχουν ίσως διαφορετικές παραμέτρους, ενώ παράλληλα θα πρέπει να τροποποίησει το firmware των Motes και το λογισμικό που πραγματοποιεί την επικοινωνία με το δίκτυο έξυπνης σκόνης, ώστε να περιλαμβάνει τις τροποποιήσεις.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| queryTypeID | integer | Ο τύπος του query (primary key) |
| queryTypeName | varchar(40) | Εδώ μπορούμε να βάλουμε constant update ή event-driven update, αν θέλουμε δύο τύπους query. |

**SensorRequests**

Στον πίνακα αυτό περιέχονται τα queries που κάνουμε στους κόμβους του δικτύου. Όσον αφορά στο πλήθος των κόμβων και των αισθητήρων που σχετίζονται με κάποιο συγκεκριμένο query, μπορούν να αφορούν πολλά διαφορετικά motes (ακόμα και ολόκληρο το δίκτυο) και καθένα από αυτά τα motes να χρειάζεται να χρησιμοποιήσει παραπάνω από έναν από τους αισθητήρες που διαθέτει. Λόγω αυτής της many-to-many σχέσης, επιλέξαμε να χρησιμοποιήσουμε δύο παραπάνω πίνακες στο σχήμα της βάσης του jWebDust που να εκφράζουν τις σχέσεις μεταξύ queries, motes και αισθητήρων.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| requestID | integer | Το ID του query (primary key) |
| queryTypeID | integer | Ο τύπος του query, σύμφωνα με τον πίνακα QueryTypes |
| start | timestamp | Ακριβής ώρα εκκίνησης του query, αν είναι τύπου constant update |
| stop | timestamp | Ακριβής ώρα λήξης του query, αν είναι τύπου constant update |
| interval | integer | Το διάστημα μεταξύ διαδοχικών αναφορών του κόμβου, αν είναι τύπου constant update |
| threshold | long | Το όριο που θέτουμε για αναφορά, αν είναι τύπου event-driven |

**SensorQueries**

Ο πίνακας αυτός περιέχει την αντιστοίχιση μεταξύ των SensorRequests και των αισθητήρων από τους οποίους ζητούνται δεδομένα για το συγκεκριμένο SensorRequest. Γιά κάθε τέτοιο αισθητήρα υπάρχει ξεχωριστή καταχώρηση, με άλλα λόγια ένα query μπορεί να ζητά πληροφορία από 4 αισθητήρες, οπότε θα υπάρχουν 4 ξεχωριστές καταχωρήσεις.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| queryID | integer | Το ID του query σύμφωνα με τον πίνακα SensorRequests (primary key) |
| sensorTypeID | integer | Το ID του αισθητήρα στον οποίο αντιστοιχεί το συγκεκριμένο query |

Ο πίνακας αυτός έχει μία many-to-many σχέση με τον πίνακα SensorRequests.

**MoteQueries**

Ο πίνακας αυτός περιέχει την αντιστοίχιση μεταξύ των SensorRequests και των motes από τα οποία ζητούνται δεδομένα για το συγκεκριμένο SensorRequest. Υπάρχει ξεχωριστή καταχώριση για κάθε mote απο το οποίο ζητείται να απαντήσει στο query, εκτός κι αν το query είναι attribute-based, δηλαδή απευθύνεται σε όλους τους κόμβους του δικτύου.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| queryID | integer | Το ID του query σύμφωνα με τον πίνακα SensorRequests (primary key) |
| moteID | integer | Το ID του κόμβου στον οποίο αντιστοιχεί το συγκεκριμένο query |
| sensorNetworkID | integer | Το ID του δικτύου στο οποίο ανήκει ο κόμβος στον οποίο αντιστοιχεί το συγκεκριμένο query |

Ο πίνακας αυτός έχει μία many-to-many σχέση με τον πίνακα SensorRequests.

### 6.3.2 Κώδικας σε SQL για τη δημιουργία της βάσης

Στην ενότητα αυτή δίνουμε τον κώδικα σε SQL για τη δημιουργία των πινάκων που αντιστοιχούν στο σχήμα που παρουσιάσαμε στις προηγούμενες ενότητες. Οι ονομασίες για τους τύπους δεδομένων που χρησιμοποιούμε είναι σε κάποιες περιπτώσεις συγκεκριμένες μόνο για την PostgreSQL.

Να σημειωθεί ότι όπου χρησιμοποιείται ο τύπος δεδομένων `serial`, αυτό γίνεται για να δίνονται μοναδικές τιμές ID στα αντίστοιχα πεδία στους πίνακες που χρησιμοποιείται αυτός ο τύπος. Π.χ. όταν δηλώνουμε έναν νεό τύπο αισθητήρα θέλουμε το ID του να είναι σίγουρα διαφορετικό από των υπολοίπων και παράλληλα να είναι μέλος μιας ακουλουθίας. Δηλαδή, αν έχουμε ήδη 5 τύπους αισθητήρων οι οποίοι έχουν ID 1,2,3,4,5 αντίστοιχα, θέλουμε ο καινούριος

τύπος να έχει ID 6. Ο τύπος δεδομένων `serial` αυτοματοποιεί τη διαδικασία αυτή.

Επίσης, στα πεδία όπου θέτουμε περιορισμούς ξένου κλειδιού (foreign key constraints), θέτουμε κάποιες οδηγίες για τη συμπεριφορά της βάσης σε περίπτωση διαγραφής κάποιων καταχωρίσεων. Έτσι λοιπόν, θέτουμε τον περιορισμό όταν ο χρήστης επιχειρεί να αλλάξει τιμή στον Α πίνακα η οποία χρησιμοποιείται ως ξένο κλειδί στον Β πίνακα, να μην επιτρέπεται η αλλαγή αυτή, και όταν επιχειρήσει τη διαγραφή της αυτόματα να διαγράφονται και οι αντίστοιχες τιμές στον πίνακα Β.

**MoteTypes**

```
CREATE TABLE "MoteTypes"
(
"moteTypeID" serial,
"moteTypeDescription" text,
"moteTypeName" varchar(40),
CONSTRAINT "moteTypeID" PRIMARY KEY ("moteTypeID")
)
WITHOUT OIDS;
```

**SensorTypes**

```
CREATE TABLE "SensorTypes"
(
"sensorTypeID" serial,
"sensorTypeName" varchar(40),
"SensorTypeDescription" text,
CONSTRAINT "sensorTypeID" PRIMARY KEY ("sensorTypeID")
)
WITHOUT OIDS;
```

**Motes**

```
CREATE TABLE "Motes"
(
"moteID" serial,
"moteLocation" varchar(40),
"moteTypeID" int4,
"GPS_longitude" varchar(40),
"GPS_lattitude" varchar(40),
CONSTRAINT "moteID" PRIMARY KEY ("moteID"),
CONSTRAINT "moteType" FOREIGN KEY ("moteTypeID") REFERENCES "Mote-
Types" ("moteTypeID") ON UPDATE RESTRICT ON DELETE RESTRICT
)
WITHOUT OIDS;
```

**SensorReadings**

```
CREATE TABLE "SensorReadings"
(
"ID" serial,
date timestamp DEFAULT ('now'::text)::timestamp(6) with time zone,
"moteID" int4,
"sensorTypeID" int4,
"RawData" int8,
CONSTRAINT "ID" PRIMARY KEY ("ID"),
CONSTRAINT "moteID" FOREIGN KEY ("moteID") REFERENCES "Motes"
("moteID") ON UPDATE RESTRICT ON DELETE RESTRICT,
CONSTRAINT "sensorTypeID" FOREIGN KEY ("sensorTypeID") REFERENCES
"SensorTypes" ("sensorTypeID") ON UPDATE RESTRICT ON DELETE RESTRICT
)
WITHOUT OIDS;
```

**MoteSensors**

```
CREATE TABLE "MoteSensors"
(
"sensorTypeID" serial,
"moteID" int4 NOT NULL,
CONSTRAINT "key" PRIMARY KEY ("sensorTypeID", "moteID"),
CONSTRAINT "moteID" FOREIGN KEY ("moteID") REFERENCES "Motes"
("moteID") ON UPDATE RESTRICT ON DELETE RESTRICT,
CONSTRAINT "sensorTypeID" FOREIGN KEY ("sensorTypeID") REFERENCES
"SensorTypes" ("sensorTypeID") ON UPDATE RESTRICT ON DELETE RESTRICT
)
WITHOUT OIDS;
```

**QueryTypes**

```
CREATE TABLE "QueryTypes"
(
"queryTypeID" serial,
"queryTypeName" varchar(40),
CONSTRAINT "qtID" PRIMARY KEY ("queryTypeID")
)
WITHOUT OIDS;
```

**SensorRequests**

```
CREATE TABLE "SensorRequests"
(
"requestID" serial,
"queryTypeID" int4,
"start" timestamp,
stop timestamp,
"interval" int4,
"readingInterval" int8,
CONSTRAINT "rID" PRIMARY KEY ("requestID"),
CONSTRAINT "qtID" FOREIGN KEY ("queryTypeID") REFERENCES "Query-
Types" ("queryTypeID") ON UPDATE RESTRICT ON DELETE RESTRICT
)
WITHOUT OIDS;
```

**SensorQueries**

```
CREATE TABLE "SensorQueries"
(
"queryID" int4 NOT NULL,
"sensorTypeID" int4 NOT NULL,
CONSTRAINT "qID" PRIMARY KEY ("queryID", "sensorTypeID"),
CONSTRAINT "sTypeID" FOREIGN KEY ("sensorTypeID") REFERENCES "Sen-
sorTypes" ("sensorTypeID") ON UPDATE RESTRICT ON DELETE RESTRICT,
CONSTRAINT "srID" FOREIGN KEY ("queryID") REFERENCES "SensorRequests"
("requestID") ON UPDATE RESTRICT ON DELETE
RESTRICT
)
WITHOUT OIDS;
```

**MoteQueries**

```
CREATE TABLE "MoteQueries"
(
"queryID" int4 NOT NULL,
"moteID" int4 NOT NULL,
CONSTRAINT mquery PRIMARY KEY ("queryID", "moteID"),
CONSTRAINT "mID" FOREIGN KEY ("moteID") REFERENCES "Motes"
("moteID") ON UPDATE RESTRICT ON DELETE RESTRICT,
CONSTRAINT "srID" FOREIGN KEY ("queryID") REFERENCES "SensorRequests"
("requestID") ON UPDATE RESTRICT ON DELETE RESTRICT
)
WITHOUT OIDS;
```

**Διαγραφή ολόκληρης της βάσης**

Για να διαγράψουμε όλους τους πίνακες της βάσης δεδομένων του jWebDust, εισάγουμε τις παρακάτω SQL εντολές.

```
drop table "MoteTypes" CASCADE;
drop table "Motes" CASCADE;
drop table "MoteSensors" CASCADE;
drop table "SensorReadings" CASCADE;
drop table "SensorTypes" CASCADE;
drop table "QueryTypes" CASCADE;
drop table "SensorRequests" CASCADE;
drop table "MoteQueries" CASCADE;
drop table "SensorQueries" CASCADE;
```

**Κώδικας για την εισαγωγή δεδομένων**

Στη γενική περίπτωση ο χρήστης του jWebDust δε θα χρειαστεί να εισάγει δεδομένα στη βάση δεδομένων μέσω χρήσης εντολών SQL, αλλά η διαδικασία αυτή είναι χρήσιμη για λόγους debugging ή για την περίπτωση που κάποιος θέλει να επεκτείνει το σύστημα και θέλει ένα ''εικονικό'' δίκτυο έξυπνης σκόνης, ενώ χρησιμεύει και για το testing της ορθής λειτουργίας της βάσης δεδομένων.

Έστω λοιπόν ότι θέλουμε να καταχωρίσουμε ένα δίκτυο έξυπνης σκόνης το οποίο περιέχει 5 motes τύπου MICA, τα οποία έχουν 5 αισθητήρες το καθένα, ενώ μπορούμε να έχουμε 7 τύπους αισθητήρων συνολικά. Αρχικά, εισάγουμε τους πιθανούς τύπους motes στο δίκτυο. Έστω ότι μπορούμε να έχουμε τους εξής τύπους motes και τις αντίστοιχες περιγραφές:

- Mica, First generation MICA motes.
- Mica2, Second generation MICA motes.
- Mica2Dot, Second generation MICA motes, compact size, integrated sensors.
- MicaZ, Third generation MICA motes.

Ο κώδικας για να εισάγουμε τα στοιχεία αυτά στη βάση ειναι ο εξής:

```
insert into "MoteTypes" values ( DEFAULT, 'First generation MICA motes', 'mica');
insert into "MoteTypes" values ( DEFAULT, 'Second generation MICA motes', 'mica2');
insert into "MoteTypes" values ( DEFAULT, 'Second generation MICA motes, compact
size, integrated sensors', 'mica2dot');
insert into "MoteTypes" values ( DEFAULT, 'Third generation MICA motes', 'micaz');
```

Έστω τώρα ότι θέλουμε να εισάγουμε δύο κομβους στο δίκτυο, οι οποίοι είναι τύπου Mica2 (αφού προηγουμένως εισάγαμε δεύτερο κατά σειρά το συγκεκριμένο τύπο, το ID του θα είναι 2).

```
insert into "Motes" values ( DEFAULT, 'Somewhere...', 2);
insert into "Motes" values ( DEFAULT, 'Somewhere...', 2);
```

Έστω τώρα ότι θέλουμε να έχουμε 7 τύπους αισθητήρων στο δίκτυο έξυπνης σκόνης. Εισάγουμε τις παρακάτω εντολές στη βάση δεδομένων:

```
insert into "SensorTypes" values ( DEFAULT, 'MTS101CA Light sensor', NULL);
insert into "SensorTypes" values ( DEFAULT, 'MTS101CA Thermistor', NULL);
insert into "SensorTypes" values ( DEFAULT, 'MTS300-310CA Microphone', NULL);
insert into "SensorTypes" values ( DEFAULT, 'MTS300-310CA Thermistor', NULL);
insert into "SensorTypes" values ( DEFAULT, 'MTS310CA Accelerometer', NULL);
insert into "SensorTypes" values ( DEFAULT, 'MTS310CA Magnetometer', NULL);
insert into "SensorTypes" values ( DEFAULT, 'MTS300-310CA Light sensor', NULL);
```

```
insert into "MoteSensors" values ( 3, 1);
insert into "MoteSensors" values ( 4, 1);
insert into "MoteSensors" values ( 5, 1);
insert into "MoteSensors" values ( 6, 1);
insert into "MoteSensors" values ( 7, 1);
```

```
insert into "QueryTypes" values ( DEFAULT, 'constant update driven');
insert into "QueryTypes" values ( DEFAULT, 'event driven report');
```

Για να εισάγουμε κάποια εικονικά δεδομένα από το δίκτυο έξυπνης σκόνης στη βάση δίνουμε τις παρακάτω εντολές.

```
insert into "SensorRequests" values ( DEFAULT, 1, NOW(), NOW(), 600, NULL);
insert into "SensorRequests" values ( DEFAULT, 1, NOW(), NOW(), 600, NULL);
insert into "SensorRequests" values ( DEFAULT, 1, NOW(), NOW(), 600, NULL);
insert into "SensorRequests" values ( DEFAULT, 1, NOW(), NOW(), 600, NULL);
insert into "SensorRequests" values ( DEFAULT, 1, NOW(), NOW(), 600, NULL);
```

## 6.4 Προδιαγραφές λειτουργίας για το ενδιάμεσο επιπέδο (Middle Tier)

**Ορισμός:** Το *ενδιάμεσο επίπεδο* είναι αυτό το οποίο υλοποιεί τη λογική που ενώνει όλα τα υπόλοιπα επίπεδα σε ένα σύστημα. Το βασικό καθήκον του επιπέδου αυτού, όπως φανερώνει το όνομά του, είναι να προσφέρει ένα interface με το οποίο να αλληλεπιδρούν τα άλλα επίπεδα με το επίπεδο πληροφορίας, και πιο συγκεκριμένα, το επίπεδο ελέγχου και το επίπεδο παρουσίασης με τη βάση δεδομένων.

**Περιγραφή:** Καθώς η αρχιτεκτονική σχεδόν όλου του συστήματος υλοποιείται σε γλώσσα Java, το ίδιο συμβαίνει κι εδώ. Ο προγραμματιστής πρέπει να προσφέρει ένα interface που να βασίζεται σε αντικείμενα που αντιστοιχούν στους πίνακες της βάσης δεδομένων (*Logic module*), να προσφέρει διαχειριστές των αντικειμένων αυτών (*Manager module*) τέτοιους ώστε η διαχείριση σχετιζόμενων αντικειμένων να γίνεται από τον ίδιο διαχειριστή με ομοιόμορφο τρόπο και ένα module για όλη την υπόλοιπη λογική του συστήματος (*Business logic module*), το οποίο περιλαμβάνει τη λογική δημιουργίας query και τη διασύνδεση του συστήματος με τον τελικό χρήστη.

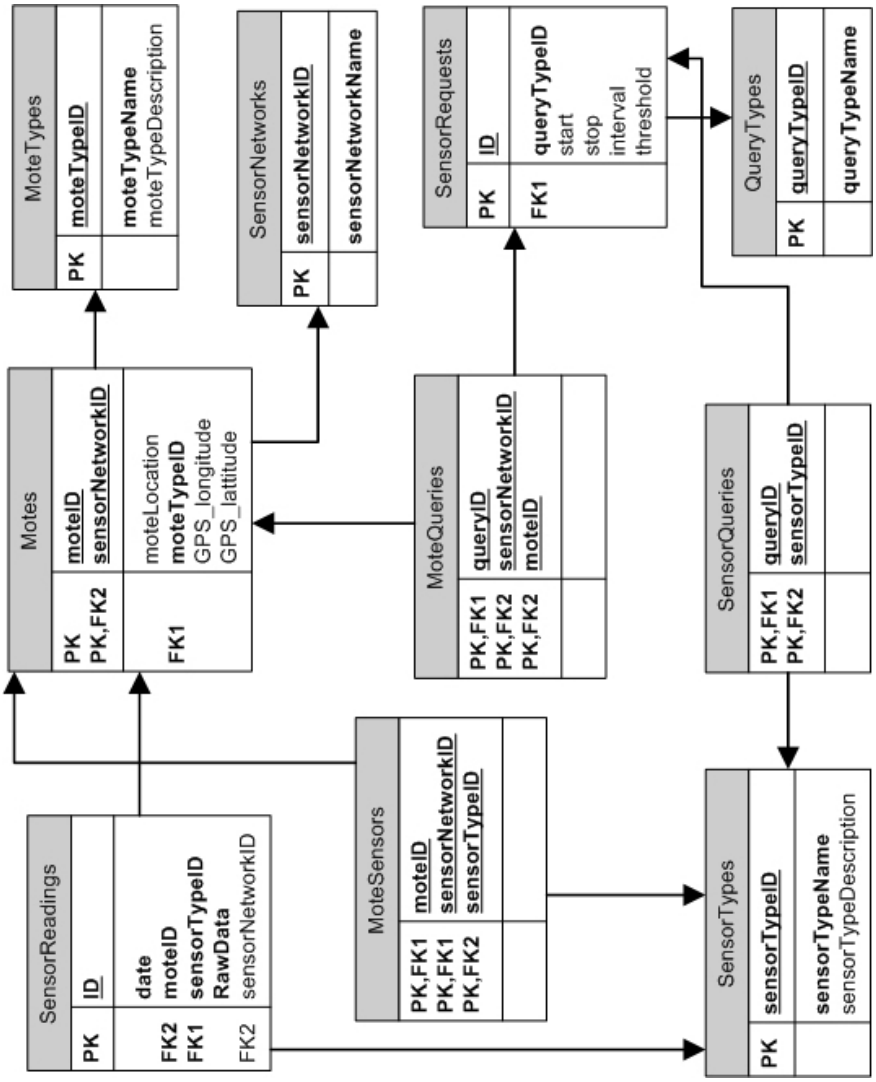
Η λογική ύπαρξης του ενδιάμεσου επιπέδου είναι ότι:

- Απλοποιεί τον τρόπο που μπορούν τα υπόλοιπα επίπεδα να προσπελάσουν και να χρησιμοποιήσουν τις εγγραφές που βρίσκονται στο επίπεδο πληροφορίας.
- Απομακρύνει από τον χρήστη την ανάγκη να ασχοληθεί προγραμματιστικά με το επίπεδο πληροφορίας, και συνεπώς δε χρειάζεται να εμπλακεί με τον προγραμματισμό σε SQL.
- Προσφέρει ένα πλήθος από έτοιμες δυνατότητες μέσω ενός συγκεκριμένου interface, τις οποίες ο χρήστης μπορεί να εκμεταλλευτεί και να χτίσει τη δική του εφαρμογή με ευκολία.

**Σενάριο:** Έστω ένας προγραμματιστής, ο κύριος Τζαβέλας, ο οποίος αναλαμβάνει να γράψει κώδικα για το ενδιάμεσο επίπεδο. Σκοπός του ενδιάμεσου επιπέδου είναι να προσφέρει ένα interface στα υπόλοιπα επίπεδα μέσω του οποίου να έχουν πρόσβαση στη βάση δεδομένων του συστήματος και να επιτελούν τις λειτουργίες που είναι απαραίτητες για την ενημέρωσή της, αλλά και να εξάγουν πληροφορία με μια οργανωμένη και περισσότερη αξιοποιήσιμη μορφή.

### 6.4.1 Logic module

Σε αυτό το κομμάτι του ενδιάμεσου επιπέδου, ο κύριος Τζαβέλας πρέπει να δημιουργήσει αντικείμενα που να αντιστοιχούν στις εγγραφές των πινάκων της βάσης δεδομένων του συστήματος, μαζί με ένα interface το οποίο επιτρέπει




Σχήμα 6.3: Το σχήμα της βάσης του jWebDust.

την αλλαγή των τιμών των μελών των αντικειμένων αυτών, καθώς και την επιστροφή των τιμών αυτών. Το interface αυτό χρησιμοποιείται από το Manager module (βλέπε επόμενη υποενότητα). Χρησιμοποιούμε αυτά τα αντικείμενα για να μπορούμε να επεξεργαστούμε την πληροφορία που περιέχεται στο επίπεδο πληροφορίας ή να πριν εισαχθούν δεδομένα από τα άλλα επίπεδα στο επίπεδο πληροφορίας, αντί να επενεργούμε απευθείας στα δεδομένα που περιέχονται σε αυτό.

Έχοντας υπόψιν ότι ο κώδικας για τα αντικείμενα αυτά θα γραφεί σε γλώσσα Java, τα ονόματα για τους τύπους δεδομένων είναι τα αντίστοιχα που χρησιμοποιούνται στη Java. Όπου αναφέρεται ο τύπος δεδομένων Timestamp, εννοείται ο τύπος *java.sql.Timestamp*. Ακολουθεί μια παρουσίαση των αντικειμένων αυτών.

**Motes**

**Ορισμός:** Αυτή η κλάση είναι μία άμεση αντιστοίχιση στον πίνακα Motes της βάσης δεδομένων στο επίπεδο πληροφορίας.

**Μέλη:** Τα μέλη της κλάσης αυτής περιέχονται στον ακόλουθο πίνακα.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| moteID | int | Το ID του κάθε κόμβου στο δίκτυο |
| sensorNetworkID | int | Το ID του δικτύου έξυπνης σκόνης στο οποίο ανήκει ο κάθε κόμβος |
| moteLocation | String | Μια περιγραφή της θέσης του mote στο δίκτυο, π.χ. Κτήριο Β. |
| moteTypeID | int | Ο τύπος mote που αντιστοιχεί στο συγκεκριμένο mote. Οι δυνατές επιλογές είναι mica, mica2, mica2dot, micaZ (βλ. επόμενη παράγραφο). |
| GPS_longitude | String | Η θέση του mote στο δίκτυο (προαιρετικό), αν έχουμε GPS στο sensor board |
| GPS_lattitude | String | Η θέση του mote στο δίκτυο (προαιρετικό), αν έχουμε GPS στο sensor board |
| status | String | Η κατάσταση του κόμβου (alive, asleep, dead, κτλ) |
| lastUpdate | Timestamp | Το Timestamp για την τελευταία επαφή με τον κόμβο |

**Περιγραφή συναρτήσεων κατασκευής:** Η κλάση αυτή περιέχει τις παρακάτω συναρτήσεις κατασκευής:

(i) Motes(): δημιουργεί ένα αντικείμενο mote με moteID ίσο με 0, με moteLocation "Somewhere" και moteTypeID ίσο με 1, δηλαδή MICA mote με τις default ρυθμίσεις.

(ii) Motes(int mID, String mLocation, int mTID, String lon, String lat, String stat, int sNID, Timestamp lUpdate): συνάρτηση κατασκευής μέσω της οποίας μπορούμε να δημιουργήσουμε ένα αντικείμενο της κλάσης και να θέσουμε τιμές για κάθε ένα από τα μέλη της.

**Υπόλοιπες συναρτήσεις-μέλη:** Οι υπόλοιπες συναρτήσεις αυτής της κλάσης είναι απλές getter και setter συναρτήσεις, δηλαδή χρησιμεύουν για να επιστρέφουν και να αλλάζουν τις τιμές των μελών της κλάσης. Η ονομασία των συναρτήσεων αυτών είναι του τύπου getAttribute() και setAttribute(type), όπου Attribute είναι το αντίστοιχο όνομα του μέλους της κλάσης.

**MoteTypes**

**Ορισμός:** Αυτή η κλάση είναι μία άμεση αντιστοίχιση στον πίνακα MoteTypes της βάσης δεδομένων στο επίπεδο πληροφορίας.

**Μέλη:** Τα μέλη της κλάσης αυτής περιέχονται στον ακόλουθο πίνακα.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| moteTypeID | int | Το ID του τύπου του mote. |
| moteTypeName | String | Ένα string για τον τύπο του mote. Οι δυνατές επιλογές είναι mica, mica2, mica2dot, micaZ. |
| moteTypeDescription | String | Εδώ βάζουμε μια σύντομη περιγραφή του συγκεκριμένου τύπου. |

**Περιγραφή συναρτήσεων κατασκευής:** Η κλάση αυτή περιέχει τις παρακάτω συναρτήσεις κατασκευής:

(i) MoteTypes(): δημιουργεί ένα αντικείμενο moteTypes με moteTypeID ίσο με $-1$, moteTypeDescription "unknown mote type" και moteTypeName το ίδιο.

(ii) MoteTypes(int mTID, String mTDescription, String mTName): συνάρτηση κατασκευής μέσω της οποίας μπορούμε να δημιουργήσουμε ένα αντικείμενο της κλάσης και να θέσουμε τιμές για κάθε ένα από τα μέλη της.

**Υπόλοιπες συναρτήσεις-μέλη:** Οι υπόλοιπες συναρτήσεις αυτής της κλάσης είναι απλές getter και setter συναρτήσεις, δηλαδή χρησιμεύουν για να επιστρέφουν και να αλλάζουν τις τιμές των μελών της κλάσης. Η ονομασία των συναρτήσεων αυτών είναι του τύπου getAttribute() και setAttribute(type), όπου Attribute είναι το αντίστοιχο όνομα του μέλους της κλάσης.

**SensorNetworks**

**Ορισμός:** Αυτή η κλάση είναι μία άμεση αντιστοίχιση στον πίνακα SensorNetworks της βάσης δεδομένων στο επίπεδο πληροφορίας.

**Μέλη:** Τα μέλη της κλάσης αυτής περιέχονται στον ακόλουθο πίνακα.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| sensorNetworkID | int | Το ID του δικτύου στο οποίο ανήκει ο κόμβος |
| sensorNetworkName | String | Μια περιγραφή του δικτύου |

**Περιγραφή συναρτήσεων κατασκευής:** Η κλάση αυτή περιέχει τις παρακάτω συναρτήσεις κατασκευής:

(i) SensorNetworks(): δημιουργία ενός SensorNetworks αντικειμένου με sensorNetworkID ίσο με 0 και sensorNetworkName "Standard Sensor Network".

(ii) SensorNetworks(int sensorNetworkID, String sensorNetworkName): συνάρτηση κατασκευής μέσω της οποίας μπορούμε να δημιουργήσουμε ένα αντικείμενο της κλάσης και να θέσουμε τιμές για κάθε ένα από τα μέλη της.

**Υπόλοιπες συναρτήσεις-μέλη:**

**SensorReadings**

**Ορισμός:** Αυτή η κλάση είναι μία άμεση αντιστοίχιση στον πίνακα SensorReadings της βάσης δεδομένων στο επίπεδο πληροφορίας.

**Μέλη:**

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| ID | int | Το ID για τις μετρήσεις. Κάθε μέτρηση πρέπει να έχει διαφορετικό ID για να γίνεται η διάκριση μεταξύ τους |
| date | Timestamp | Η ακριβής ώρα και ημερομηνία στην οποία έγινε η συγκεκριμένη μέτρηση |
| moteID | int | Το ID του κόμβου του δικτύου από τον οποίο προήλθε η μέτρηση |
| sensorNetworkID | int | Το ID του δικτύου στο οποίο ανήκει ο κόμβος από τον οποίο προήλθε η μέτρηση |
| sensorTypeID | int | Ο τύπος του αισθητήρα από τον οποίο έγινε η μέτρηση |
| RawData | long | Η πληροφορία από τη μέτρηση, σε binary μορφή. Η μετάφρασή της σε μορφή που να γίνεται κατανοητή εναπόκειται στην εφαρμογή που χρησιμοποιεί τη βάση. |

**Περιγραφή συναρτήσεων κατασκευής:** Η κλάση αυτή περιέχει τις παρακάτω συναρτήσεις κατασκευής:

(i) SensorReadings(): δημιουργία ενός αντικειμένου SensorReadings με ID ίσο με $-1$ (προς το παρόν δεν έχει σημασία η τιμή του ID γιατί όταν εισάγεται κάποια μέτρηση στη βάση δεδομένων παίρνει νέα τιμή από τη βάση), date ίση με 0 (και εδώ η βάση αρχικοποιεί στη συνέχεια), moteID ίσο με 1, sensorTypeID ίσο με 1 και rawData ίσο με 0.

(ii) SensorReadings(int id, Timestamp dte, int mID, int sNID, int sTID, long data): συνάρτηση κατασκευής μέσω της οποίας μπορούμε να δημιουργήσουμε ένα αντικείμενο της κλάσης και να θέσουμε τιμές για κάθε ένα από τα μέλη της.

**Υπόλοιπες συναρτήσεις-μέλη:** Οι υπόλοιπες συναρτήσεις αυτής της κλάσης είναι απλές getter και setter συναρτήσεις, δηλαδή χρησιμεύουν για να επιστρέφουν και να αλλάζουν τις τιμές των μελών της κλάσης. Η ονομασία των συναρτήσεων αυτών είναι του τύπου getAttribute() και setAttribute(type), όπου Attribute είναι το αντίστοιχο όνομα του μέλους της κλάσης.

**MoteSensors**

**Ορισμός:** Αυτή η κλάση είναι μία άμεση αντιστοίχιση στον πίνακα MoteSensors της βάσης δεδομένων στο επίπεδο πληροφορίας.

**Μέλη:** Τα μέλη της κλάσης αυτής περιέχονται στον ακόλουθο πίνακα.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| moteID | int | Το ID του κόμβου στο οποίο αναφερόμαστε |
| sensorNetworkID | int | Το ID του δικτύου στο οποίο ανήκει ο κόμβος |
| sensorTypeID | int | Το ID του αισθητήρα ο οποίος περιλαμβάνεται στον κόμβο που έχει το moteID |

**Περιγραφή συναρτήσεων κατασκευής:** Η κλάση αυτή περιέχει τις παρακάτω συναρτήσεις κατασκευής:

(i) MotesSensors(): δημιουργεί μια default καταχώρηση αισθητήρα με sensorTypeID ίσο με 3 για έναν κόμβο με moteID ίσο με 1 και sensorNetworkID 0.

(ii) MoteSensors(int sTID, int mID, int sNID): συνάρτηση κατασκευής μέσω της οποίας μπορούμε να δημιουργήσουμε ένα αντικείμενο της κλάσης και να θέσουμε τιμές για κάθε ένα από τα μέλη της.

**Υπόλοιπες συναρτήσεις-μέλη:** Οι υπόλοιπες συναρτήσεις αυτής της κλάσης είναι απλές getter και setter συναρτήσεις, δηλαδή χρησιμεύουν για να επιστρέφουν και να αλλάζουν τις τιμές των μελών της κλάσης. Η ονομασία των συναρτήσεων αυτών είναι του τύπου getAttribute() και setAttribute(type), όπου Attribute είναι το αντίστοιχο όνομα του μέλους της κλάσης.

**SensorTypes**

**Ορισμός:** Αυτή η κλάση είναι μία άμεση αντιστοίχιση στον πίνακα SensorTypes της βάσης δεδομένων στο επίπεδο πληροφορίας.

**Μέλη:** Τα μέλη της κλάσης αυτής περιέχονται στον ακόλουθο πίνακα.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| sensorTypeID | integer | Το ID για τον τύπο αισθητήρα |
| sensorTypeName | String | Ένα string, το οποίο δίνει το όνομα του αισθητήρα |
| sensorTypeDescription | String | Μια αναλυτική περιγραφή του ολοκληρωμένου του αισθητήρα |

**Περιγραφή συναρτήσεων κατασκευής:** Η κλάση αυτή περιέχει τις παρακάτω συναρτήσεις κατασκευής:

(i) SensorTypes(): δημιουργεί ένα SensorTypes αντικείμενο με sensorTypesID ίσο με $-1$ (για να πάρει τη default τιμή όταν εισαχθεί στη βάση δεδομένων), και sensorTypeName και sensorTypeDescription "Unknown sensor type".

(ii) SensorTypes(int sTID, String sTName, String sTDescription): συνάρτηση κατασκευής μέσω της οποίας μπορούμε να δημιουργήσουμε ένα αντικείμενο της κλάσης και να θέσουμε τιμές για κάθε ένα από τα μέλη της.

**Υπόλοιπες συναρτήσεις-μέλη:** Οι υπόλοιπες συναρτήσεις αυτής της κλάσης είναι απλές getter και setter συναρτήσεις, δηλαδή χρησιμεύουν για να επιστρέφουν και να αλλάζουν τις τιμές των μελών της κλάσης. Η ονομασία των συναρτήσεων αυτών είναι του τύπου getAttribute() και setAttribute(type), όπου Attribute είναι το αντίστοιχο όνομα του μέλους της κλάσης.

**Querytypes**

**Ορισμός:** Αυτή η κλάση είναι μία άμεση αντιστοίχιση στον πίνακα QueryTypes της βάσης δεδομένων στο επίπεδο πληροφορίας.

**Μέλη:** Τα μέλη της κλάσης αυτής περιέχονται στον ακόλουθο πίνακα.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| queryTypeID | int | Ο τύπος του query |
| queryTypeName | String | Εδώ μπορούμε να βάλουμε constant update ή event-driven update |

**Περιγραφή συναρτήσεων κατασκευής:** Η κλάση αυτή περιέχει τις παρακάτω συναρτήσεις κατασκευής:

(i) QueryTypes(): δημιουργεί ένα αντικείμενο QyeryTypes με queryTypeID ίσο με $-1$ και queryTypeName "unknown type query".

(ii) QueryTypes(int qID, String qTName): συνάρτηση κατασκευής μέσω της οποίας μπορούμε να δημιουργήσουμε ένα αντικείμενο της κλάσης και να θέσουμε τιμές για κάθε ένα από τα μέλη της.

**Υπόλοιπες συναρτήσεις-μέλη:** Οι υπόλοιπες συναρτήσεις αυτής της κλάσης είναι απλές getter και setter συναρτήσεις, δηλαδή χρησιμεύουν για να επιστρέφουν και να αλλάζουν τις τιμές των μελών της κλάσης. Η ονομασία των συναρτήσεων αυτών είναι του τύπου getAttribute() και setAttribute(type), όπου Attribute είναι το αντίστοιχο όνομα του μέλους της κλάσης.

**SensorRequests**

**Ορισμός:** Αυτή η κλάση είναι μία άμεση αντιστοίχιση στον πίνακα SensorRequests της βάσης δεδομένων στο επίπεδο πληροφορίας.

**Μέλη:** Τα μέλη της κλάσης αυτής περιέχονται στον ακόλουθο πίνακα.

| **Πεδίο** | **Τύπος δεδομένων** | **Περιγραφή** |
|---|---|---|
| requestsID | int | Το ID του query |
| queryTypeID | int | Ο τύπος του query, σύμφωνα με τον πίνακα QueryTypes |
| start | Timestamp | Ακριβής ώρα εκκίνησης του query, αν είναι τύπου constant update |
| stop | Timestamp | Ακριβής ώρα λήξης του query, αν είναι τύπου constant update |
| interval | int | Το διάστημα μεταξύ διαδοχικών αναφορών του κόμβου, αν είναι τύπου constant update |
| threshold | long | Το όριο που θέτουμε για αναφορά, αν είναι τύπου event-driven |

**Περιγραφή συναρτήσεων κατασκευής:** Η κλάση αυτή περιέχει τις παρακάτω συναρτήσεις κατασκευής:

(i) SensorRequests(): δημιουργεί ένα αντικείμενο SensorRequests με requestID ίσο με -1, που σημαινεί DEFAULT τιμή όταν εισαχθεί αργότερα στη βάση δεδομένων, queryTypeID ίσο με 1, δηλαδή constant update

reporting query, start ίσο με Timestamp(0) , δηλαδή NOW() τιμή όταν εισαχθεί στη βάση δεδομένων, stop ίσο με Timestamp(start + 60000), δηλαδή το query έχει διάρκεια ζωής 1 λεπτό, interval ίσο με 1000ms, δηλαδή μέτρηση κάθε δευτερόλεπτο και threshold ίσο με 0, αφού είναι constant update reporting query.

(ii) SensorRequests(int rID, int qTID, Timestamp strt, Timestamp stp, int val, long rval): συνάρτηση κατασκευής μέσω της οποίας μπορούμε να δημιουργήσουμε ένα αντικείμενο της κλάσης και να θέσουμε τιμές για κάθε ένα από τα μέλη της.

**Υπόλοιπες συναρτήσεις-μέλη:** Οι υπόλοιπες συναρτήσεις αυτής της κλάσης είναι απλές getter και setter συναρτήσεις, δηλαδή χρησιμεύουν για να επιστρέφουν και να αλλάζουν τις τιμές των μελών της κλάσης. Η ονομασία των συναρτήσεων αυτών είναι του τύπου getAttribute() και setAttribute(type), όπου Attribute είναι το αντίστοιχο όνομα του μέλους της κλάσης.

**SensorQueries**

**Ορισμός:** Αυτή η κλάση είναι μία άμεση αντιστοίχιση στον πίνακα SensorQueries της βάσης δεδομένων στο επίπεδο πληροφορίας.

**Μέλη:** Τα μέλη της κλάσης αυτής περιέχονται στον ακόλουθο πίνακα.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| queryID | int | Το ID του query σύμφωνα με τον πίνακα SensorRequests |
| sensorTypeID | int | Το ID του αισθητήρα στον οποίο αντιστοιχεί το συγκεκριμένο query |

**Περιγραφή συναρτήσεων κατασκευής:** Η κλάση αυτή περιέχει τις παρακάτω συναρτήσεις κατασκευής:

(i) SensorQueries(): δημιουργεί ένα αντικείμενο SensorQueries με sensorTypeID ίσο με 3 και ID ίσο με $-1$.

(ii) SensorQueries(int qID, int sTID): συνάρτηση κατασκευής μέσω της οποίας μπορούμε να δημιουργήσουμε ένα αντικείμενο της κλάσης και να θέσουμε τιμές για κάθε ένα από τα μέλη της.

**Υπόλοιπες συναρτήσεις-μέλη:** Οι υπόλοιπες συναρτήσεις αυτής της κλάσης είναι απλές getter και setter συναρτήσεις, δηλαδή χρησιμεύουν για να επιστρέφουν και να αλλάζουν τις τιμές των μελών της κλάσης. Η ονομασία των συναρτήσεων αυτών είναι του τύπου getAttribute() και setAttribute(type), όπου Attribute είναι το αντίστοιχο όνομα του μέλους της κλάσης.

**MoteQueries**

**Ορισμός:** Αυτή η κλάση είναι μία άμεση αντιστοίχιση στον πίνακα MoteQueries της βάσης δεδομένων στο επίπεδο πληροφορίας.

**Μέλη:** Τα μέλη της κλάσης αυτής περιέχονται στον ακόλουθο πίνακα.

| Πεδίο | Τύπος δεδομένων | Περιγραφή |
|---|---|---|
| queryID | int | Το ID του query σύμφωνα με τον πίνακα SensorRequests |
| moteID | int | Το ID του κόμβου στον οποίο αντιστοιχεί το συγκεκριμένο query |
| sensorNetworkID | int | Το ID του δικτύου στο οποίο ανήκει ο κόμβος στον οποίο αντιστοιχεί το συγκεκριμένο query |

**Περιγραφή συναρτήσεων κατασκευής:** Η κλάση αυτή περιέχει τις παρακάτω συναρτήσεις κατασκευής:

(i) MoteQueries(): δημιουργεί ένα αντικείμενο MoteQueries με queryID και moteID ίσα με $-1$.

(ii) MoteQueries(int qID, int mID, int sNID): συνάρτηση κατασκευής μέσω της οποίας μπορούμε να δημιουργήσουμε ένα αντικείμενο της κλάσης και να θέσουμε τιμές για κάθε ένα από τα μέλη της.

**Υπόλοιπες συναρτήσεις-μέλη:** Οι υπόλοιπες συναρτήσεις αυτής της κλάσης είναι απλές getter και setter συναρτήσεις, δηλαδή χρησιμεύουν για να επιστρέφουν και να αλλάζουν τις τιμές των μελών της κλάσης. Η ονομασία των συναρτήσεων αυτών είναι του τύπου getAttribute() και setAttribute(type), όπου Attribute είναι το αντίστοιχο όνομα του μέλους της κλάσης.

### 6.4.2 Manager module

Το module αυτό περιλαμβάνει τους διαχειριστές των αντικειμένων των οποίες τις προδιαγραφές δώσαμε στην προηγούμενη υποενότητα. Υπάρχουν τέσσερις τέτοιοι διαχειριστές:

1. **MotesManager**: είναι υπεύθυνος για τις λειτουργίες πάνω στα αντικείμενα που αντιστοιχούν σε εγγραφές στους πίνακες `Motes` και `MoteSensors`.

2. **ReadingsManager**: είναι υπεύθυνος για τις λειτουργίες πάνω στα αντικείμενα που αντιστοιχούν σε εγγραφές στον πίνακα `SensorReadings`.

3. **RequestsManager**: είναι υπεύθυνος για τις λειτουργίες πάνω στα αντικείμενα που αντιστοιχούν σε εγγραφές στους πίνακες `SensorRequests`, `MoteQueries` και `SensorQueries`.

4. **TypesManager**: είναι υπεύθυνος για τις λειτουργίες πάνω στα αντικείμενα που αντιστοιχούν σε εγγραφές στους πίνακες `MoteTypes`, `Query Types` και `SensorTypes`.

Σε αυτό το σημείο να σημειώσουμε ότι και στους τέσσερις διαχειριστές που περιγράφουμε σε αυτή την υποενότητα υπάρχει μόνο ένα μέλος, το οποίο είναι τύπου *java.sql.Connection* και είναι μια σύνδεση σε μια σχεσιακή βάση δεδομένων, για να μπορέσουμε να αλληλεπιδράσουμε με τη βάση αυτή μέσω των διαχειριστών και του JDBC interface της. Ακολουθεί μια συνοπτική περιγραφή των διαχειριστών και των λειτουργιών τους.

**MotesManager**

**Ορισμός:** Ο MotesManager είναι ο υπεύθυνος διαχειριστής για τις λειτουργίες πάνω στα αντικείμενα που ανήκουν στις κλάσεις Motes και MoteSensors. Οι λειτουργίες που θέλουμε να μπορούμε να κάνουμε σε αυτά τα αντικείμενα είναι:

- εισαγωγή αντικειμένων Motes και MoteSensors, τα οποία έχουν αρχικοποιηθεί εκ των προτέρων, στη βάση δεδομένων του συστήματος (μαζί με έλεγχο για την εγκυρότητά τους).
- επιλογή ξεχωριστών καταχωρήσεων ή του συνόλου των καταχωρήσεων Motes και MoteSensors από τη βάση δεδομένων του συστήματος.
- διαγραφή ξεχωριστών καταχωρήσεων ή του συνόλου των καταχωρήσεων Motes και MoteSensors από τη βάση δεδομένων του συστήματος.

**Περιγραφή διαθέσιμων μεθόδων:**

**MotesManager(Connection conn):** συνάρτηση κατασκευής του διαχειριστή που αρχικοποιεί τη σύνδεση με τη βάση δεδομένων.

**void insertMote(Motes mote, MoteSensors[] moteSensors):** εισαγωγή ενός νέου κόμβου στη βάση δεδομένων μαζί με τους αισθητήρες τους οποίους φέρει.

**Motes[] selectAllMotes():** επιλογή όλων των καταχωρήσεων κόμβων του δικτύου που υπάρχουν στη βάση δεδομένων.

**Motes selectMote(int mID):** επιλογή συγκεκριμένου κόμβου από τη βάση δεδομένων με κλειδί το ID του κόμβου.

**void deleteMote(int mID):** διαγραφή συγκεκριμένου κόμβου από τη βάση δεδομένων με κλειδί το ID του κόμβου.

**void updateMote(Motes mote):** ενημέρωση των στοιχείων συγκεκριμένου κόμβου από τη βάση δεδομένων με κλειδί το ID του κόμβου.

**MoteSensors[] selectAllMoteSensors():** επιλογή του συνόλου των αισθητήρων που υπάρχουν στους κόμβους του δικτύου, δηλαδή τους διαφορετικούς τύπους αισθητήρων που περιέχονται στο δίκτυο.

**MoteSensors[] selectMoteSensors(int mID):** επιλογή του συνόλου των αισθητήρων που φέρει κάποιος συγκεκριμένος κόμβος.

**void deleteMoteSensor(int mID, int stID):** διαγραφή ενός συγκεκριμένου αισθητήρα από συγκεκριμένο κόμβο.

**ReadingsManager**

**Ορισμός:** Ο ReadingsManager είναι ο υπεύθυνος διαχειριστής για τις λειτουργίες πάνω στα αντικείμενα που ανήκουν στην κλάση SensorReadings. Οι λειτουργίες που θέλουμε να μπορούμε να κάνουμε σε αυτά τα αντικείμενα είναι:

- εισαγωγή αντικειμένων SensorReadings (δηλαδή μετρήσεων από το δίκτυο) στη βάση δεδομένων του συστήματος.
- διαγραφή μεμονωμένων αντικειμένων SensorReadings από τη βάση δεδομένων του συστήματος
- επιλογή καταχωρήσεων από τη βάση δεδομένων με κλειδί το moteID του κόμβου για τον οποίο ενδιαφερόμαστε ή επιλογή όλων των καταχωρήσεων της βάσης.
- επιλογή καταχωρήσεων από τη βάση δεδομένων με κλειδί το networkID των κόμβων για τους οποίους ενδιαφερόμαστε.
- επιλογή καταχωρήσεων από τη βάση δεδομένων με κλειδί τη χρονική περίοδο για την οποία ενδιαφερόμαστε.

**Περιγραφή διαθέσιμων μεθόδων:**

**ReadingsManager(Connection conn):** συνάρτηση κατασκευής του διαχειριστή που αρχικοποιεί τη σύνδεση με τη βάση δεδομένων.

**void insertReading(SensorReadings reading):** εισαγωγή μέτρησης από το δίκτυο στη βάση δεδομένων. Η μέτρηση θα πάρει αυτόματα ID.

**void deleteSensorReading(SensorReadings reading):** διαγραφή μέτρησης από τη βάση δεδομένων.

**void updateSensorReading(SensorReadings reading):** ενημέρωση των στοιχείων μέτρησης στη βάση δεδομένων.

**SensorReadings[] selectAll():** επιλογή όλων των μετρήσεων που περιέχονται στη βάση δεδομένων.

**SensorReadings[] selectReadingByMote(int mid):** επιλογή μετρήσεων από τη βάση δεδομένων με κλειδί το ID του κόμβου που τις έστειλε.

**RequestsManager**

**Ορισμός:** Ο RequestsManager είναι ο υπεύθυνος διαχειριστής για τις λειτουργίες πάνω στα αντικείμενα που ανήκουν στις κλάσεις SensorRequests, MoteQueries και SensorQueries. Οι λειτουργίες που θέλουμε να μπορούμε να κάνουμε σε αυτά τα αντικείμενα είναι:

- εισαγωγή ενός νέου query (αντικείμενο SensorRequest) στη βάση δεδομένων του συστήματος.
- επιλογή του συνόλου ή μεμονωμένου query από τη βάση του συστήματος.
- επιλογή καταχωρήσεων MoteQueries με κλειδί το moteID ή το SensorRequestID.
- διαγραφή καταχωρήσων MoteQueries με κλειδί το SensorRequestID.
- επιλογή καταχωρήσεων SensorQueries με κλειδί το sensorTypeID ή το SensorRequestID.
- διαγραφή καταχωρήσων SensorQueries με κλειδί το SensorRequestID.

**Περιγραφή διαθέσιμων μεθόδων:**

**RequestsManager(Connection conn):** συνάρτηση κατασκευής του διαχειριστή που αρχικοποιεί τη σύνδεση με τη βάση δεδομένων.

**void insertRequest(SensorRequests request, SensorQueries[] squeries, MoteQueries[] mqueries):** εισαγωγή νέου query στη βάση δεδομένων μαζί με τους αισθητήρες και τους κόμβους τους οποίους αφορά.

**SensorRequests[] selectAllSensorRequests():** επιλογή όλων των SensorRequest που βρίσκονται αποθηκευμένα στη βάση δεδομένων.

**SensorRequests selectSensorRequest(int rID):** επιλογή συγκεκριμένου query με κλειδί το ID του.

**MoteQueries[] selectAllMoteQueries():** επιλογή όλων των καταχωρήσεων MoteQueries από τη βάση δεδομένων.

**MoteQueries[] selectMoteQueriesByID(int qID):** επιλογή συγκεκριμένης καταχώρησης MoteQueries με κλειδί το ID της.

**MoteQueries[] selectMoteQueriesByMote(int mID):** επιλογή συγκεκριμένης καταχώρησης MoteQueries με κλειδί το ID του Mote στο οποίο αναφέρεται.

**SensorQueries[] selectAllSensorQueries():** επιλογή όλων των καταχωρήσεων SensorQueries από τη βάση δεδομένων.

**SensorQueries[] selectSensorQueriesByID(int qID):** επιλογή συγκεκριμένης καταχώρησης SensorsQueries με κλειδί το ID της.

**SensorQueries[] selectSensorQueriesBySensor(int sID):** επιλογή συγκεκριμένης καταχώρησης SensorsQueries με κλειδί το ID του αισθητήρα στον οποίο αναφέρεται.

**void deleteMoteQueries(int qID):** διαγραφή καταχώρησης MoteQueries με βάση το ID της.

**void deleteSensorQueries(int qID):** διαγραφή καταχώρησης SensorQueries με βάση το ID της.

**TypesManager**

**Ορισμός:**Ο TypesManager είναι ο υπεύθυνος διαχειριστής για τις λειτουργίες πάνω στα αντικείμενα που ανήκουν στις κλάσεις MoteTypes, SensorTypes και QueryTypes. Οι λειτουργίες που θέλουμε να μπορούμε να κάνουμε σε αυτά τα αντικείμενα είναι:

- εισαγωγή νέων τύπων mote, sensor, query στη βάση δεδομένων του συστήματος.
- διαγραφή τύπων mote, sensor, query από τη βάση δεδομένων του συστήματος.
- ενημέρωση τύπων mote, sensor, query από τη βάση δεδομένων του συστήματος.

**Περιγραφή διαθέσιμων μεθόδων:**

**TypesManager(Connection conn):** συνάρτηση κατασκευής του διαχειριστή που αρχικοποιεί τη σύνδεση με τη βάση δεδομένων.

**void insertMoteType(MoteTypes mType):** εισαγωγή νέου τύπου Motes στη βάση δεδομένων του συστήματος.

**void insertQueryType(QueryTypes qType):** εισαγωγή νέου τύπου query στη βάση δεδομένων του συστήματος.

**void insertSensorType(SensorTypes sType):** εισαγωγή νέου τύπου αισθητήρα στη βάση δεδομένων του συστήματος.

**void deleteMoteType(int mtID):** διαγραφή συγκεκριμένου τύπου mote από τη βάση δεδομένων του συστήματος.

**void deleteQueryType(int qtID): διαγραφή συγκεκριμένου τύπου query από τη βάση δεδομένων του συστήματος.**

**void deleteSensorType(int stID):** διαγραφή συγκεκριμένου τύπου αισθητήρα από τη βάση δεδομένων του συστήματος.

**MoteTypes[] selectAllMoteTypes():** επιλογή όλων των τύπων mote που βρίσκονται καταχωρημένοι στο σύστημα.

**MoteTypes selectMoteType(int mtID):** επιλογή συγκεκριμένου τύπου mote με κλειδί το moteID του.

**QueryTypes[] selectAllQueryTypes():** επιλογή όλων των τύπων query που βρίσκονται καταχωρημένοι στο σύστημα.

**QueryTypes selectQueryType(int qtID):** επιλογή συγκεκριμένου τύπου query με κλειδί το ID του.

**SensorTypes[] selectAllSensorTypes():** επιλογή όλων των τύπων αισθητήρων που βρίσκονται καταχωρημένοι στο σύστημα.

**SensorTypes selectSensorType(int stID):** επιλογή συγκεκριμένου τύπου αισθητήρα με κλειδί το ID του.

**void updateMoteType(MoteTypes mType):** ενημέρωση των στοιχείων της καταχώρησης για κάποιο συγκεκριμένο τύπο mote.

**void updateSensorType(SensorTypes sType):** ενημέρωση των στοιχείων της καταχώρησης για κάποιο συγκεκριμένο τύπο αισθητήρα.

**void updateQueryType(QueryTypes qType):** ενημέρωση των στοιχείων της καταχώρησης για κάποιο συγκεκριμένο τύπο query.

## 6.5 Προδιαγραφές λειτουργίας του επιπέδου παρουσίασης (Presentation Tier)

**Ορισμός:** Το *επίπεδο παρουσίασης* είναι το interface με τον τελικό χρήστη και είναι το επίπεδο του συστήματος μέσω του οποίου γίνεται το σύνολο των διαχειριστικών λειτουργιών του συστήματος και το οποίο έχει τη δυνατότητα παρουσίασης των μετρήσεων που λαμβάνονται από το δίκτυο έξυπνης σκόνης.

**Περιγραφή:** Το *επίπεδο παρουσίασης* αποτελεί το επίπεδο το οποίο είναι απ' ευθείας διαθέσιμο στον τελικό χρήστη, μετά την έναρξη λειτουργίας των κατώτερων επιπέδων του συστήματος (επίπεδο ελέγχου, επίπεδο πληροφορίας, ενδιάμεσο επίπεδο).

Οι λειτουργίες που αντιστοιχούν στο επίπεδο παρουσίασης είναι οι εξής:

- Διαχείριση του ασύρματου δικτύου αισθητήρων. Με τον όρο διαχείριση του δικτύου εννοούμε την ρύθμιση παραμέτρων του δίκτυου, όπως η προσθήκη, ενημέρωση ή διαγραφή για τους τύπους κόμβων, αισθητήρων, κτλ, των διαθέσιμων δικτύων αισθητήρων, κ.α.

- Υποβολή query στο ασύρματο δίκτυο αισθητήρων. Μέσω ενός απλού interface ο χρήστης μπορεί να στείλει query προς εκτέλεση στους κόμβους του ασύρματου δικτύου αισθητήρων, καθορίζοντας έτσι την πληροφορία που επιθυμεί από το δίκτυο.

- Παρουσίαση των μετρήσεων από το ασύρματο δίκτυο αισθητήρων. Ο χρήστης μπορεί να καθορίσει την πληροφορία που επιθυμεί να δει από τις μετρήσεις που βρίσκονται αποθηκευμένες στο επίπεδο πληροφορίας, καθώς και τη μορφή που θέλει να τη δει. Μπορεί να επιλέξει μεταξύ διαφορετικών μονάδων και διαφορετικών αναπαραστάσεων (π.χ. γραφικές παραστάσεις ή ραβδογράμματα).

Στην ενότητα αυτή παρουσιάζουμε μια αρκετά πρώιμη μορφή του επιπέδου παρουσίασης του jWebDust, καθώς το συγκεκριμένο επίπεδο είναι ακόμα σε φάση σχεδίασης και ανάπτυξης.

**Σενάριο:** Έστω ο κύριος Χιψίδης, ο οποίος προτίθεται να χρησιμοποιήσει το jWebDust για να διαχειρίζεται ένα ασύρματο δίκτυο αισθητήρων που θέλει να εγκαταστήσει για τους σκοπούς της ερευνητικής του ομάδας. Αφού τα υπόλοιπα μέλη της ερευνητικής του ομάδας στήσουν τα κατώτερα επίπεδα του jWebDust, ο κύριος Χιψίδης θέλει να διαχειριστεί το ασύρματο δίκτυο αισθητήρων του εργαστηρίου του χρησιμοποιώντας ένα web-browser στο PC του, με άλλα λόγια θέλει ένα απλό και εύχρηστο περιβάλλον διαχείρισης (επίσης, αυτό τον διευκολύνει αρκετά, αφού δεν διαθέτει laptop και κάποιες φορές θέλει να βλέπει τι γίνεται μέσα στο δίκτυο από το PC στο σπίτι του).

Έστω λοιπόν ότι ο κύριος Χιψίδης διαθέτει ένα σύνολο από 20 συσκευές MICA2 και θέλει να χρησιμοποιήσει κάποιες από αυτές για να μετρήσει τη

θερμοκρασία σε κάποιους χώρους και τις υπόλοιπες για να μετρήσει το φώς σε κάποιους άλλους χώρους. Θα περιγράψουμε τις λειτουργίες που πρέπει να διαθέτει το web interface του συστήματος για να υλοποιηθεί το σενάριο αυτό.

Η όλη διαδικασία περιλαμβάνει τις εξής φάσεις:

1. Ρύθμιση παραμέτρων του δικτύου
2. Εγκατάσταση του ασύρματου δικτύου (στην οποία δε θα αναφερθούμε)
3. Αποστολή query
4. Οπτικοποίηση των μετρήσεων από το δίκτυο

**Ρύθμιση και προβολή παραμέτρων:** Καθώς οι συσκευές MICA2 είναι ανάμεσα σε αυτές που υποστηρίζονται από το σύστημα, ο διαχειριστής δεν χρειάζεται να εισάγει νέους τύπους κόμβων, αισθητήρων, κτλ. Εφόσον υπάρχει το πρωτόκολλο δήλωσης κόμβων, ο διαχειριστής δεν χρειάζεται να εκτελέσει κάποια ενέργεια. Μετά την έναρξη της λειτουργίας των κόμβων του ασύρματου δικτύου αισθητήρων, οι κόμβοι δηλώνονται αυτόματα στο σύστημα και είναι έτοιμοι να δεχτούν τα query του χρήστη.

Στην περίπτωση όμως χρήσης συσκευών που δεν υποστηρίζονται από το σύστημα, πρέπει να δηλωθούν οι νέοι τύποι κόμβων και αισθητήρων που θα υπάρχουν στο ασύρματο δίκτυο αισθητήρων. Νέοι τύποι query δεν μπορούν να οριστούν προς το παρόν. Η φάση αυτή πρέπει να γίνει πριν από την εγκατάσταση του λογισμικού στους κόμβους του δικτύου.

Επομένως, υπάρχει μια web σελιδα στην οποία ο χρήστης έχει τις εξής δυνατότητες:

- *Δήλωση νέου τύπου κόμβων:* Ο χρήστης πρέπει να δηλώσει όνομα και περιγραφή του νέου τύπου κόμβων.
- *Δήλωση νέου τύπου αισθητήρων:* Ο χρήστης πρέπει να δηλώσει όνομα και περιγραφή του νέου τύπου αισθητήρων, μαζί με μια συνάρτηση μετατροπής των μετρήσεων που γίνονται από αισθητήρες αυτού του τύπου σε μονάδες κατανοητές από το χρήστη. Η συνάρτηση αυτή και κάποια τάση αναφοράς (από τον συγκεκριμένο κόμβο ή ίδια για όλους τους κόμβους αυτού του τύπου αν δεν υπάρχει δυνατότητα μέτρησης τάσης αναφοράς, όπως π.χ στα MICA motes)

Παράλληλα με τη δυνατότητα δήλωσης νέων τύπων κόμβων και αισθητήρων, υπάρχει η δυνατότητα αλλαγής (edit) για τα πεδία των εγγραφών για τους τύπους αυτούς. Ο χρήστης μπορεί επίσης να αλλάξει το πεδίο που αναφέρει την τοποθεσία που βρίσκεται ο κάθε κόμβος, καθώς και τις συντεταγμένες GPS για τον κόμβο αυτό.

Επίσης, προς ευκολία του διαχειριστή, υπάρχουν οι επιλογές για την προβολή συγκεκριμένων χαρακτηριστικών του συστήματος:

- *Προβολή των διαθέσιμων τύπων κόμβων στο σύστημα:* επιστρέφει μια σελίδα η οποία περιέχει τις εγγραφές που υπάρχουν στη βάση δεδομένων για τους τύπους κόμβων στο σύστημα.

- *Προβολή των διαθέσιμων ασύρματων δικτύων αισθητήρων:* επιστρέφει μια σελίδα η οποία περιέχει τις εγγραφές που υπάρχουν στη βάση δεδομένων για τα ασύρματα δίκτυα που υπάρχουν στο σύστημα.

- *Προβολή χαρακτηριστικών συγκεκριμένου κόμβου:* επιστρέφει μια σελίδα η οποία περιέχει τις εγγραφές που υπάρχουν στη βάση δεδομένων για κάποιον συγκεκριμένο κόμβο, τον οποίο έχει επιλέξει ο διαχειριστής από μια λίστα με όλους τους κόμβους που υπάρχουν καταγεγραμμένοι στο σύστημα.

- *Προβολή των κόμβων που ανήκουν σε συγκεκριμένο ασύρματο δίκτυο αισθητήρων:* επιστρέφει μια σελίδα η οποία περιέχει τις εγγραφές που υπάρχουν στη βάση δεδομένων για τους κόμβους που ανήκουν σε κάποιο ασύρματο δίκτυο αισθητήρων από αυτά που έχουν δηλωθεί στο σύστημα.

- *Προβολή των τύπων αισθητήρων που είναι διαθέσιμοι στο σύστημα:* επιστρέφει μια σελίδα η οποία περιέχει τις εγγραφές που υπάρχουν στη βάση δεδομένων για τους τύπους αισθητήρων στο σύστημα.

- *Προβολή των αισθητήρων που φέρουν συγκεκριμένοι κόμβοι:* επιστρέφει μια σελίδα η οποία περιέχει τις εγγραφές που υπάρχουν στη βάση δεδομένων για τους αισθητήρες που φέρουν κόμβοι οι οποίοι έχουν επίλεγεί από μια λίστα με όλους τους κόμβους που υπάρχουν καταγεγραμμένοι στο σύστημα.

- *Προβολή των κόμβων του δικτύου που διαθέτουν συγκεκριμένους αισθητήρες:* επιστρέφει μια σελίδα η οποία περιέχει τις εγγραφές που υπάρχουν στη βάση δεδομένων για τους κόμβους που φέρουν ένα σύνολο συγκεκριμένων αισθητήρων, οι οποίοι αισθητήρες έχουν επιλεγεί από μια λίστα με όλους τους τύπους αισθητήρων που υπάρχουν καταγεγραμμένοι στο σύστημα.

- *Προβολή των τύπων query που είναι διαθέσιμοι στο σύστημα:* επιστρέφει μια σελίδα η οποία περιέχει τις εγγραφές που υπάρχουν στη βάση δεδομένων για τους τύπους query στο σύστημα.

- *Προβολή όλων των query που εκτελούνται στο σύστήμα:* επιστρέφει μια σελίδα η οποία περιέχει τις εγγραφές που υπάρχουν στη βάση δεδομένων για τα query που είναι ενεργά (δεν έχουν ακυρωθεί ή ολοκληρώσει τον κύκλο τους) τη συγκεκριμένη χρονική στιγμή που γίνεται το αίτημα.

**Αποστολή query:** Ο χρήστης θα πρέπει να υποβάλλει κάποια query στο σύστημα για να αρχίσουν οι κόμβοι του δικτύου να κάνουν τις επιθυμητές μετρήσεις. Στη

συγκεκριμένη περίπτωση θέλουμε να υποβάλλουμε δύο query (υποθέτουμε ότι έχουμε δώσει ID στους κόμβους του δικτύου από 1 έως 20):

1. Ένα periodic sensing query για τους κόμβους 1 έως 10, για μέτρηση της θερμοκρασίας ανά 10 λεπτά από τις 12:00 μμ στις 1 Ιουλίου 2005 έως τις 12:00 μμ στις 20 Ιουλίου 2005.

2. Ένα periodic sensing query για τους κόμβους 11 έως 20, για μέτρηση του φωτός ανά 10 λεπτά από τις 12:00 μμ στις 1 Ιουλίου 2005 έως τις 12:00 μμ στις 20 Ιουλίου 2005.

Υπάρχει μια σελίδα στην οποία ο χρήστης μπορεί να υποβάλλει query στο σύστημα χρησιμοποιώντας τα εξής πεδία:

- *Τύπος query:* προς το παρόν διαθέσιμη επιλογή είναι η periodic sensing.
- *Επιλογή ασύρματου δικτύου αισθητήρων:* ο χρήστης επιλέγει το ασύρματο δίκτυο αισθητήρων που τον ενδιαφέρει.
- *Επιλογή κόμβων:* ο χρήστης επιλέγει από μια λίστα με κόμβους αυτούς στους οποίους επιθυμεί να σταλεί το query. Η λίστα αυτή περιέχει όλους τους κόμβους που υπάρχουν στο σύστημα για το ασύρματο δίκτυο αισθητήρων που έχει επιλεγεί.
- *Επιλογή αισθητήρων:* ο χρήστης επιλέγει από μια λίστα με αισθητήρες αυτούς από τους οποίους επιθυμεί να λάβει μετρήσεις. Η λίστα περιέχει όλους τους αισθητήρες που περιέχονται στο σύστημα.
- *Περίοδος δειγματοληψίας:* επιλογή της περιόδου δειγματοληψίας από μία λίστα με τιμές. Η μικρότερη διαθέσιμη τιμή είναι 5 λεπτά και η μεγαλύτερη μία ώρα.
- *Ημερομηνία έναρξης και λήξης:* καθορισμός των χρονικών στιγμών που αρχίζει ή σταματά να εκτελείται το query. Η ημερομηνία προσδιορίζεται με την χρήση μήνα, ημέρας, ώρας και λεπτών. Η default τιμή για τις δύο ημερομηνίες είναι "τώρα".

Με την υποβολή ενός query, μέσω των επιπέδων πληροφορίας και ελέγχου, αυτό αποστέλλεται στο επίπεδο αισθητήρων. Επίσης, ο χρήστης έχει τη δυνατότητα να ακυρώσει κάποιο συγκεκριμένο query.

**Οπτικοποίηση μετρήσεων:** Ο χρήστης μπορεί να βλέπει τις μετρήσεις από το δίκτυο μέσω του jWebDust.

- *Επιλογή ασύρματου δικτύου αισθητήρων:* ο χρήστης επιλέγει το ασύρματο δίκτυο αισθητήρων (από τα διαθέσιμα) από το οποίο θέλει να δει μετρήσεις.

- *Επιλογή τύπου αισθητήρων:* ο χρήστης επιλέγει τους αισθητήρες για τους οποίους ενδιαφέρεται να δεί μετρήσεις (από τους διαθέσιμους στο συγκεκριμένο ασύρματο δίκτυο αισθητήρων που έχει επιλέξει προηγουμένως).

- *Επιλογή συγκεκριμένων κόμβων:* ο χρήστης επιλέγει τους κόμβους από τους οποίους επιθυμεί να δει τις μετρήσεις που τον ενδιαφέρουν. Η επιλογή γίνεται από μια λίστα με κόμβους, η οποία επιρεάζεται από τις δύο προηγούμενες επιλογές. Αρχικά, η λίστα αυτή αποτελείται από όλους τους κόμβους που υπάρχουν στο σύστημα, και ανάλογα με τις επιλογές που κάνει ο χρήστης για το ασύρματο δίκτυο αισθητήρων και τους τύπους αισθητήρων, εισάγονται στη λίστα οι κόμβοι που ικανοποιούν τις επιλογές του χρήστη.

- *Επιλογή timestamp για τις μετρήσεις:* ο χρήστης πρέπει να επιλέξει τις ημερομηνίες ανάμεσα στις οποίες έχουν καταγραφεί οι μετρήσεις που θέλει να δει. Η ημερομηνία προσδιορίζεται με την χρήση μήνα, ημέρας, ώρας και λεπτών.

- *Επιλογή τύπου γραφικής παράστασης:* ο χρήστης επιλέγει τον τύπο γραφικής παράστασης για την οπτικοποίηση των μετρήσεων. Διαθέσιμες επιλογές είναι το ραβδογράφημα (bar graph) και το απλό γράφημα (graph).

# Βιβλιογραφία


[1] Γ. Μυλωνάς, *Δίκτυα έξυπνης σκόνης*, Διπλωματική εργασία, Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής, Πανεπιστήμιο Πατρών, Πάτρα, 2003.

[2] Α. Αντωνίου, *Δίκτυα έξυπνης σκόνης*, Διπλωματική εργασία, Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής, Πανεπιστήμιο Πατρών, Πάτρα, 2003.

[3] *Η ιστοσελίδα του task group της IEEE για το πρότυπο 802.15.4*, http://www.ieee802.org/15/pub/TG4.html.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, *A survey on sensor networks*, IEEE Communications Magazine 40 (8) (2002) 102--114. (2002), 102--114.

[5] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, *Wireless sensor networks: a survey*, Journal of Computer Networks **38** (2002), 393--422.

[6] T. Antoniou, A. Boukerche, I. Chatzigiannakis, G. Mylonas, and S. Nikoletseas, *A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range*, 37th Annual ACM/IEEE Simulation Symposium (ANSS 2004), 2004, pp. 43--52.

[7] I. Chatzigiannakis, T. Dimitriou, M. Mavronicolas, S. Nikoletseas, and P. Spirakis, *A comparative study of protocols for efficient data propagation in smart dust networks*, Journal of Parallel Processing Letters (PPL) **13** (2003), no. 4, 615--627.

[8] I. Chatzigiannakis, G. Mylonas, and S. Nikoletseas, *jWebDust: A java-based generic application environment for wireless sensor networks*, International Conference on Distributed Computing in Sensor Systems (DCOSS '05), June 2005.

[9] I. Chatzigiannakis and S. Nikoletseas, *A sleep-awake protocol for information propagation in smart dust networks*, 3rd International Workshop on Mobile, Ad-hoc and Sensor Networks (WMAN 2003), 2003, IPDPS Workshops, p. 225.

[10] I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis, *Smart dust protocols for local detection and propagation*, 2nd ACM International Annual Workshop on Principles of Mobile Computing (POMC 2002), 2002, pp. 9--16.

[11] *Το επίσημο site της εταιρίας CrossBow*, http://www.xbow.com.

[12] J. Elson and D. Estrin, *Time synchronization for wireless sensor networks*, International Parallel and Distributed Processing Symposium (IPDPS, Workshop on Parallel and Distributed Computing Issues in Wireless and Mobile Computing (USA), 2001, pp. 186--186.

[13] J. Elson, L. Girod, and D. Estrin, *Fine-grained network time synchronization using reference broadcasts*, Fifth Symposium on Operating System Design and Implementation (OSDI 2002), 2002.

[14] D. Gay, R. von Behren, M. Welsh, E. Brewer, and D. Culler, *The nesC Language: a holistic approach to networked embedded systems*, Tech. report, Intel Research Berkeley, 2002.

[15] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin, *Em*: a software environment for developing and deploying wireless sensor networks*, 2004 USENIX technical conference (USA), December 2003, pp. 186--186.

[16] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer, *A system for simulation, emulation, and deployment of heterogeneous sensor networks*, 2nd ACM Conference on Embedded Networked Sensor Systems SenSys '04 (Baltimore, Maryland, USA), November 2004, pp. 186--186.

[17] J. Hill, *System architecture for wireless sensor networks*, Ph.D. thesis, University of California, Berkeley, 2003.

[18] *Το site της εταιρίας Moteiv*, http://www.moteiv.com.

[19] *Mote-VIEW monitoring software, Crossbow Technology Inc.*, http://www.xbow.com/Products/productsdetails.aspx?sid=88.

[20] S. Nikoletseas, I. Chatzigiannakis, H. Euthimiou, A. Kinalis, T. Antoniou, and G. Mylonas, *Energy efficient protocols for sensing multiple events in smart dust networks*, 37th Annual ACM/IEEE Simulation Symposium (ANSS 2004), 2004, pp. 15--24.

[21] *Το επίσημο site για το εργαλείο διαχείρισης* pgAdmin *της βάσης δεδομένων* PostgreSQL, http://www.pgadmin.org.

[22] *PostgreSQL official web-site*, http://www.postgresql.org.

[23] *Περιγραφή της πλατφόρμας Spec*, http://www.jlhlabs.com/jhill_cs/spec/.

[24] *Πληροφορίες για την πλατφόρμα stargate*, http://www.xbow.com.

[25] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, *An analysis of a large scale habitat monitoring application*, Proceedings of the 2nd international conference on Embedded networked sensor systems (Baltimore, Maryland, USA), November 2004.

[26] R. Szewczyk, J. Polastre, Alan Mainwaring, and David Culler, *Lessons from a sensor network expedition*, Proceedings of the First European Workshop on Sensor Networks (EWSN), January 2004.

[27] *Tiny application sensor kit (TASK), Intel Research, Berkeley*, http://berkeley.intel-research.net/task/.

[28] *TinyDB: A declarative database for sensor networks*, http://telegraph.cs.berkeley.edu/tinydb/.

[29] *Το επίσημο site για το TinyOS*, http://www.tinyos.net.

[30] *Η ιστοσελίδα του Zigbee Alliance*, http://www.zigbee.org.