ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
Ακαδημαϊκό Έτος 2006-2007

Σχεδιασμός & Υλοποίηση Ενδιάμεσου Λογισμικού για εφαρμογές συστημάτων Έξυπνης Σκόνης με χρήση της JXTA

Απόστολος Παπαγεωργίου – Α.Μ. 2703
papagap@ceid.upatras.gr

Επιβλέπων: Νικολετσέας Σωτήρης
Επίκουρος Καθηγητής Πανεπιστημίου Πατρών

Πάτρα, Μάιος 2007

ΠΡΟΛΟΓΟΣ

Κάθε εξέλιξη είναι αποτέλεσμα πάρα πολλών συνιστωσών και αυτός είναι ένας από τους λόγους που το μέλλον είναι, ευτυχώς, τόσο απρόβλεπτο. Η εξελίξεις που οδηγούν στην ανάγκη για μελέτες και εργασίες σαν την παρούσα είναι σύνθετες. Είναι πολλοί οι τομείς που με την πρόοδό τους θα συνεισφέρουν ώστε η «Έξυπνη Σκόνη», ή αλλιώς τα «Σύγχρονα Συστήματα Δικτύων Αισθητήρων», να γίνει ένα σημαντικό κομμάτι της τεχνολογίας, της επιστήμης αλλά και της καθημερινής μας ζωής.

Η ραγδαία εξέλιξη στον τομέα είναι βέβαιη και βασίζεται τόσο στην πρόοδο των ασύρματων επικοινωνιών και των τεχνολογιών κατασκευής ολοκληρωμένων κυκλωμάτων όσο και στην πρόοδο του τομέα των δικτύων και την ανάπτυξη του σχετικού λογισμικού. Αλλά και σε πολλές άλλες τεχνολογικές εξελίξεις. Η παρούσα μελέτη, φιλοδοξώντας να γίνει και αυτή ένα μέλος μιας ελάχιστης συνιστώσας, πραγματεύεται το σχεδιασμό και την ανάπτυξη λογισμικού που μπορεί να βοηθήσει την εξέλιξη του τομέα των εφαρμογών Έξυπνης Σκόνης. Το συνοδευτικό CD περιέχει το αποτέλεσμα της υλοποίησης και είναι ένα αρκετά ολοκληρωμένο και λειτουργικό ενδιάμεσο λογισμικό που περιγράφεται στο κεφάλαιο 4. Η μελέτη, όμως, ολόκληρης της εργασίας είναι καίρια για την κατανόηση και την ολοκληρωμένη τεκμηρίωσή του.

Η ανάγκη για τέτοιο λογισμικό είναι μεγάλη αφού ο συγκεκριμένος τομέας βρίσκεται σε πρώιμα στάδια. Έχουν γίνει κάποιες ενδιαφέρουσες προσπάθειες, τις οποίες πραγματεύεται η παρούσα μελέτη, ενώ αναμένεται να ακολουθήσουν και άλλες. Ο περαιτέρω σχολιασμός της κατάστασης βρίσκεται στη δικαιοδοσία της εισαγωγής που θα ακολουθήσει και γι' αυτό σε αυτό το σημείο αρκεί να αναφέρω ορισμένα σημαντικά πράγματα και στη συνέχεια να ευχηθώ καλή ανάγνωση.

Πριν δώσω ευχαριστίες σε συγκεκριμένους ανθρώπους, οφείλω να αναφέρω τη σημασία που έχει η ανταλλαγή απόψεων και η αλληλοβοήθεια που προσφέρεται μέσα από τις επιστημονικές κοινότητες. Χωρίς αυτήν πιστεύω πως η εργασία, και κάθε σημαντική εργασία, θα ήταν πολύ δύσκολο ως αδύνατο να ολοκληρωθεί με επιτυχία. Συγχαρητήρια, λοιπόν, σε όλα τα μέλη των επιστημονικών forum και, ειδικότερα για την παρούσα εργασία, συγχαρητήρια και ευχαριστίες σε όλα τα μέλη των JXTA mailing lists. Τόσο σε αυτούς που «με βοήθησαν» όσο και σε αυτούς που «βοήθησα». Το κέρδος από την αλληλοβοήθεια είναι πάντα το ίδιο για όλους και εύχομαι αυτού του είδους η «ανθρώπινη» συμπεριφορά να υπάρχει πάντα και, αν είναι δυνατόν, από όλους.

Οι άμεσες ευχαριστίες μου πηγαίνουν στο Γιάννη Χατζηγιαννάκη και το Γιώργο Μυλωνά για την κατεύθυνση που έδωσαν στην εργασία μου και για τη συνεπίβλεψή της, καθώς και στον κ. Σωτήρη Νικολετσέα για την επίβλεψη και τις συμβουλές του. Ευχαριστώ επίσης τους φίλους μου (και όσους ήταν στη ζωή μου) γιατί μόνο όσο η ζωή είναι τόσο ενδιαφέρουσα όσο την κάνουν μπορώ να εργάζομαι με θέληση. Στους γονείς μου δε θα πω ένα «ευχαριστώ», γιατί θα ήταν προσβλητικά λίγο για αυτούς.

Καλή ανάγνωση.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ (WSN)

Εισαγωγή

Παραδοσιακά, κάθε υπολογιστικό σύστημα ή δίκτυο συστημάτων χρησιμοποιείται για την επεξεργασία, αποθήκευση, διακίνηση κτλ. κάποιων δεδομένων. Τα δεδομένα, αυτά καθεαυτά, παραμένουν μια παράμετρος. Μια παράμετρος που στη συντριπτική πλειοψηφία των περιπτώσεων προέρχεται από (ή σχετίζεται με) τον πραγματικό κόσμο. Η επανάσταση που ξεκινά με τα συστήματα αισθητήρων βασίζεται στο γεγονός ότι πλέον το σύστημα είναι υπεύθυνο και για τη συλλογή / παραγωγή των δεδομένων κατευθείαν από την παρατήρηση του περιβάλλοντος, με ακρίβεια και χωρίς ανθρώπινη παρέμβαση στα δεδομένα.

Ένα Ασύρματο Δίκτυο Αισθητήρων (Wireless Sensor Network) είναι μια συλλογή από κόμβους που είναι οργανωμένοι ώστε να συνεργάζονται. Κάθε κόμβος έχει επεξεργαστική δυνατότητα (π.χ. μικροελεγκτή, CPU κτλ.), διάφορα είδη μνήμης (π.χ. μνήμη προγράμματος, μνήμη δεδομένων κτλ.), ένα πομποδέκτη ραδιοσυχνοτήτων ή / και οποιοδήποτε άλλο μέσο ασύρματης επικοινωνίας, μια πηγή ενέργειας και διάφορους αισθητήρες (όργανα μέτρησης και καταγραφής τιμών παραμέτρων του περιβάλλοντος – π.χ. θερμόμετρο, κάμερα και άλλα πολλά που θα αναφερθούν στη συνέχεια).

Η τεχνολογική εξέλιξη στο επιστημονικό πεδίο των Ασύρματων Δικτύων Αισθητήρων βρίσκεται σε σχετικά πρώιμο στάδιο, τόσο όσον αφορά το υλικό όσο και όσον αφορά την ανάπτυξη πρωτοκόλλων, αρχιτεκτονικών και λογισμικού για αυτά. Παρόλ' αυτά, το φάσμα των εφαρμογών που μπορούν να αναπτυχθούν χάρη σε αυτά διαφαίνεται πολύ ευρύ και ήδη έχει ενταθεί το σχετικό ενδιαφέρον, όχι μόνο στην έρευνα αλλά και στην αγορά. Η περαιτέρω ανάπτυξη αναμένεται ότι θα οδηγήσει σύντομα σε ένα κόσμο γεμάτο με δίκτυα αισθητήρων, τα οποία πιθανότατα θα συνεργάζονται και θα είναι προσβάσιμα από το διαδίκτυο. Οι τεχνολογικές εξελίξεις που καθιστούν τα δίκτυα αισθητήρων τόσο αξιοποιήσιμα και η παρούσα κατάσταση στον τομέα περιγράφονται στην ενότητα 1.2.

Η αιτία αυτής της ανάπτυξης είναι το γεγονός ότι τα δίκτυα αισθητήρων είναι στενά συνδεδεμένα με το περιβάλλον και συνεπώς οι σχετικές εφαρμογές μπορούν να καλύψουν ανάγκες που καμία άλλη κλάση εφαρμογών δε μπορεί. Αυτή η «σχέση» των δικτύων αισθητήρων με τον πραγματικό κόσμο τα καθιστά κατάλληλα για εφαρμογές σε ένα πλήθος τομέων, όπως το περιβάλλον, η ιατρική, ο στρατός, οι μεταφορές, η διασκέδαση, η ασφάλεια χώρων κ.α. Τα παραδείγματα από εφαρμογές που έχουν ήδη αναπτυχθεί ή σχεδιαστεί είναι αρκετά για να καταλάβουμε τις δυνατότητες που υπάρχουν αλλά πολλές είναι και οι εφαρμογές που μπορούμε να φανταστούμε. Περιγραφή τέτοιων παραδειγμάτων γίνεται στην ενότητα 1.3.

Τα ζητήματα και τα ερωτήματα που ανακύπτουν είναι πάρα πολλά. Πως πρέπει να γίνει ο σχεδιασμός τέτοιων δικτύων και τι αλγόριθμοι μπορούν να εφαρμοστούν σε αυτά; Τι ομοιότητες έχουν με άλλα δίκτυα (π.χ. γνωστά κατανεμημένα συστήματα,

αδόμητα δίκτυα κτλ.) και ποιες είναι οι νέες απαιτήσεις και προκλήσεις; Η αλήθεια είναι ότι τα ασύρματα δίκτυα αισθητήρων συνδυάζουν τεχνικές από πολλά πεδία (κλασσικά δίκτυα, κατανεμημένα συστήματα, ασύρματες τηλεπικοινωνίες, λειτουργικά συστήματα) και αυτό δημιουργεί ένα νέο σύνολο απαιτήσεων. Για παράδειγμα, τα δίκτυα αισθητήρων είναι ασύρματα, έχουν περιορισμούς ενέργειας, έχουν περιορισμένους πόρους, είναι χωρο-εξαρτώμενα κτλ. Όλες αυτές οι προκλήσεις δεν έχουν αντιμετωπιστεί συχνά ταυτόχρονα. Στην ενότητα 1.4 περιγράφονται συνοπτικά τα καίρια ζητήματα των ασύρματων δικτύων αισθητήρων και κάποιες προσεγγίσεις για την επίλυσή τους.

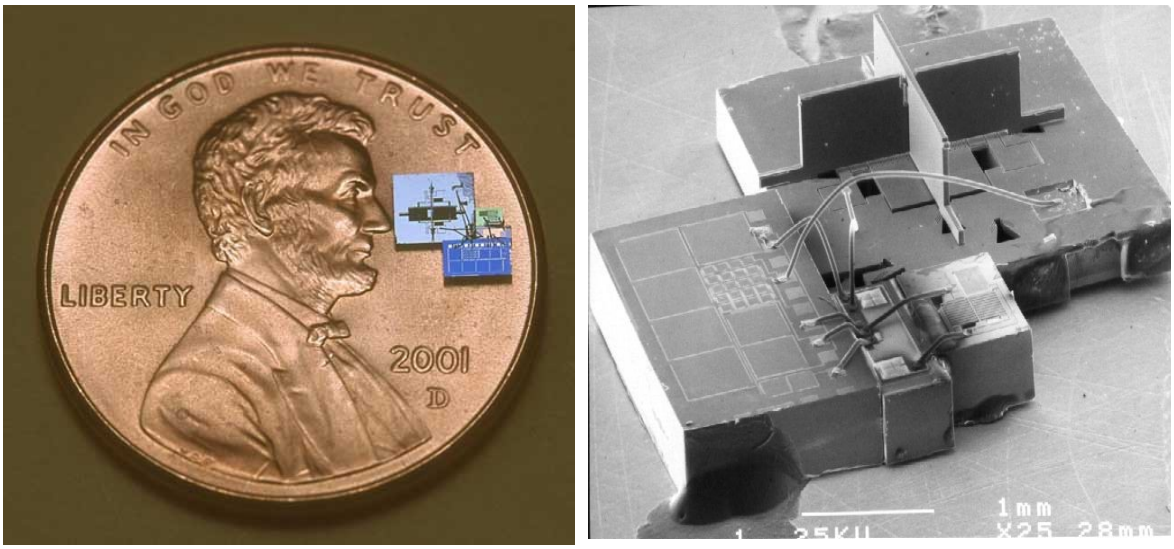
Όσο βέλτιστα κι αν αντιμετωπιστούν τα παραπάνω γενικά ζητήματα, η δημιουργία εφαρμογών για ασύρματα δίκτυα αισθητήρων θα εξακολουθεί να φαντάζει πολύπλοκη και δύσκολη. Για τη διευκόλυνση της δημιουργίας τέτοιων εφαρμογών, γίνεται μελέτη των αρχιτεκτονικών που μπορεί να έχει ένα σύστημα αισθητήρων και ανάλυση των ιδιαιτεροτήτων της κάθε αρχιτεκτονικής. Αυτό με τη σειρά του οδηγεί στην ανάπτυξη κατάλληλων πρωτοκόλλων και πεπατημένων τακτικών για τη συγγραφή εφαρμογών ή ακόμα και στην ανάπτυξη ενδιάμεσου λογισμικού (overlay software) που είναι επαναχρησιμοποιήσιμο και απλοποιεί υπερβολικά τη δημιουργία εφαρμογών. Κάθε τέτοιο overlay μπορεί να χρησιμεύσει ακόμα κι αν δε χρησιμοποιείται ως έχει, διότι δείχνει το δρόμο για την υλοποίηση κρίσιμων τμημάτων των εφαρμογών. Μια ανάλυση των αρχιτεκτονικών και του ενδιάμεσου λογισμικού για ασύρματα δίκτυα αισθητήρων γίνεται στην ενότητα 1.5.

Το [1] και το [2] πραγματεύονται γενικά τα ασύρματα δίκτυα αισθητήρων και τα εξετάζουν από πολλές πλευρές τους, με το [1] να είναι από τα πιο πλήρη εγχειρίδια για τα βασικά ζητήματα των ασύρματων δικτύων αισθητήρων. Στο υπόλοιπο της εργασίας, τα ασύρματα δίκτυα αισθητήρων θα συντομογραφούνται συνήθως ως WSN (Wireless Sensor Networks), ενώ το «ενδιάμεσο λογισμικό», που αποτελεί και κεντρικό θέμα της εργασίας, θα αναφέρεται ως overlay. Στην επόμενη ενότητα θα ξεκαθαριστεί και ο όρος «Έξυπνη Σκόνη» (Smart Dust), που κατά κάποιο τρόπο είναι εναλλακτικός του όρου «Δίκτυο Αισθητήρων».

Σύγχρονοι Αισθητήρες και «Έξυπνη Σκόνη»

Η ιδέα της χρήσης των WSN δεν εμφανίστηκε ξαφνικά. Η μέτρηση μεγεθών του περιβάλλοντος ήταν πάντοτε απαραίτητη, ενώ απλοί, ανεξάρτητοι αισθητήρες χρησιμοποιούνται στη μηχανική και στην αυτοματοποίηση (ιδιαίτερα στη μηχανολογία, αρχικά) από τη δεκαετία του ’60. Σύμφωνα με το [3], η αρχική επιστημονική ονομασία ήταν «διάταξη μετατροπής» (transducer), αφού έπαιρνε μετρήσεις και τις μετέτρεπε σε μορφή αξιοποιήσιμη από το αυτοματοποιημένο σύστημα, αλλά η λέξη «αισθητήρας» (sensor) επικράτησε όταν μπήκε η αγορά στο «παιχνίδι».

Για να αποκτήσει αξία η χρήση των αισθητήρων μέσα σε δίκτυα και να οδηγηθούμε έτσι στα WSN έπρεπε πρώτα αφ’ ενός να αναπτυχθούν και να αποκτήσουν ευρεία χρήση τα δίκτυα, αλλά ακόμα πιο καθοριστική ήταν η εξέλιξη των τεχνικών κατασκευής ολοκληρωμένων συστημάτων, που έχει επιτρέψει να ενσωματωθούν τα όργανα των αισθητήρων σε ολοκληρωμένα κυκλώματα που συνιστούν αυτόνομους κόμβους, όπως αυτοί περιγράφηκαν στην εισαγωγή. Και το μέγεθος των κόμβων να είναι, φυσικά, τόσο μικρό που να επιτρέπει πλήθος εφαρμογών. Σήμερα, τέτοιοι κόμβοι με το μέγεθος ενός σπιρτόκουτου είναι αρκετά διαδεδομένοι, ενώ η έρευνα έχει παρουσιάσει δυνατότητες ολοκλήρωσης που μπορούν να οδηγήσουν σε μέγεθος της τάξης χιλιοστών, όπως φαίνεται παρακάτω.



ΕΙΚΟΝΑ 1.1 – Κόμβος ηλιακής ενέργειας με δυνατότητες επικοινωνίας διπλής κατεύθυνσης και αισθητήρες επιτάχυνσης και φωτός. Στα αριστερά βλέπουμε το σχετικό του μέγεθος πάνω σε ένα κέρμα και στα δεξιά μια μεγέθυνσή του.

Τα επιτεύγματα αυτά είναι που οδήγησαν στο όραμα δικτύων πάρα πολλών τέτοιων κόμβων που θα συνεργάζονται και θα «ελέγχουν» περιοχές. Ίσως αυτό το όραμα να «γέννησε» και την ονομασία «Έξυπνη Σκόνη» (Smart Dust), η οποία αναφέρεται σε ένα WSN με πολύ μεγάλο πλήθος κόμβων μεγέθους σκόνης. Η έρευνα και η εφαρμογή, όμως, έχουν προχωρήσει προς όλες τις κατευθύνσεις και πλέον μελετούνται WSN διαφορετικών ειδών (διαφορετικού πλήθους κόμβων και διαφορετικών αρχιτεκτονικών)

και ολοένα ενισχύεται μάλιστα η άποψη ότι τα WSN με μικρό πλήθος συνεργαζόμενων κόμβων θα αποκτήσουν πιο ευρεία χρήση, δεδομένου ότι πολλά WSN μπορούν να συνδέονται μεταξύ τους. Ο όρος Smart Dust, που ξεκίνησε από σχετική έρευνα του πανεπιστημίου Berkeley, εξακολουθεί βέβαια να είναι δόκιμος, είτε μιλάμε για δίκτυα πολλών κόμβων είτε όχι. Οι διαφορές που ανακύπτουν στην αντιμετώπιση καίριων ζητημάτων θα εξηγηθούν σε επόμενη ενότητα.

Αυτές οι εξελίξεις, βέβαια, δεν καθιστούν αυτόματα ελκυστική τη δημιουργία εφαρμογών για WSN. Καθοριστικό ρόλο στην πρόοδο του τομέα έπαιξε επίσης η ανάπτυξη σχετικού λογισμικού χαμηλού επιπέδου. Όπως επισημάνθηκε, οι κόμβοι (motes, όπως θα αναφέρονται εναλλακτικά) έχουν επεξεργαστικές δυνατότητες και δυνατότητες δικτυακής επικοινωνίας. Συνεπώς, η ανάπτυξη λειτουργικού συστήματος ειδικού για αυτούς δίνει τεράστια ώθηση στον προγραμματισμό και στην αξιοποίησή τους. Όπως υπάρχουν πολλοί κατασκευαστές αισθητήρων, έτσι έχουν αναπτυχθεί και διάφορα σχετικά λογισμικά. Χαρακτηριστικότερο παράδειγμα είναι το λειτουργικό σύστημα TinyOS [17] καθώς και το σχετικό σύστημα Βάσης Δεδομένων TinyDB [18]. Πρόκειται για ελεύθερο λογισμικό ανοιχτού κώδικα (open-source) και δίνει μεγάλη ευελιξία στη χρήση των motes για τη δημιουργία εφαρμογών έξυπνης σκόνης. Χρησιμοποιεί τη νέα γλώσσα προγραμματισμού nesC (βασισμένη σε C, αναπτύχθηκε επίσης στο Berkeley), που είναι ειδικά σχεδιασμένη για αυτό το σκοπό. Οι δυνατότητες που παρέχονται από ένα τέτοιο λειτουργικό δεν είναι πολυποίκιλες αλλά είναι οι κατάλληλες για την περίσταση και δίνουν ό, τι χρειάζεται για τη δημιουργία εφαρμογών. Σκοπός της εργασίας δεν είναι η περαιτέρω ανάλυση ενός τέτοιου λειτουργικού, όμως για μια συνοπτική περιγραφή ο αναγνώστης παραπέμπεται στο κεφ. 6 του [4], ενώ για περισσότερες λεπτομέρειες υπάρχουν οι επίσημες ιστοσελίδες των TinyOS και nesC. Αυτό που πρέπει να γίνει σαφές είναι ότι αυτό το λογισμικό προσφέρει μια προγραμματιστική διεπαφή (API) που διευκολύνει τη λήψη περιοδικών ή κατ' αίτηση μετρήσεων για την καταγραφή τιμών σε βάση δεδομένων κτλ.

Επίσης σημαντική για την εξέλιξη των WSN ήταν και η πρόοδος των ασύρματων επικοινωνιών. Μελέτες για την εύρεση βέλτιστων τρόπων υλοποίησης της ασύρματης επικοινωνίας των motes ([5]), είτε αυτή γίνεται με ραδιοσυχνότητες, όπως συνηθίζεται, είτε με οποιονδήποτε άλλο τρόπο, βοήθησαν στο να επιτευχθούν επιθυμητοί στόχοι όσον αφορά το ελάχιστο bandwidth αυτής της επικοινωνίας, που πρέπει να είναι της τάξης μερικών kbps. Τέλος, η αντιμετώπιση ενεργειακών προβλημάτων καθώς και η επίλυση άλλων ζητημάτων, όπως αυτά που θα περιγραφούν σε επόμενη ενότητα, έδωσαν ώθηση στην εφαρμογή των WSN.

Εφαρμογές με WSN

Η παράμετρος που επηρεάζει περισσότερο από όλες το είδος των εφαρμογών που μπορούν να αναπτυχθούν με τη χρήση των WSN είναι τα είδη των αισθητήρων που μπορούν να ενσωματωθούν στους κόμβους. Οι κλάδοι που μπορούν να επωφεληθούν από την ιδέα της λήψης και διακίνησης δεδομένων σε πραγματικό χρόνο είναι αμέτρητες. Το θέμα είναι τι είδους δεδομένα μπορεί να είναι αυτά. Οι δημοφιλέστερες φυσικές παράμετροι για τις οποίες υπάρχουν αισθητήρες που μπορούν να ενσωματωθούν στα ολοκληρωμένα των motes είναι η θερμοκρασία, η ορατή ακτινοβολία (εικόνα), η υπέρυθρη ακτινοβολία, ο ήχος, η ταχύτητα, η επιτάχυνση, η υγρασία, η δόνηση, η πίεση, η ποσότητα χημικών ουσιών, ο μαγνητισμός κ.α.

Τα παραπάνω ορίζουν ένα άπειρο πλήθος πιθανών εφαρμογών. Ο σκοπός των παραδειγμάτων που θα περιγραφούν στη συνέχεια δεν είναι να βάλουν ένα περίγραμμα στις δυνατότητες που δημιουργούνται χάρη στα WSN αλλά, αντιθέτως, είναι να δείξουν τη δυναμική του τομέα, να δώσουν ιδέες για νέες εφαρμογές και να καταστήσουν κατανοητό το πώς αξιοποιούνται οι δυνατότητες των WSN. Πρέπει να σημειωθεί επίσης ότι η εξέλιξη και η ανάπτυξη νέων αρχιτεκτονικών και νέου ενδιάμεσου λογισμικού «γεννούν» νέες δυνατότητες για εφαρμογή των WSN. Επίσης, τα παραδείγματα που ακολουθούν είναι γενικά και κάποια από αυτά (αλλά και κάποια άλλα) ίσως γίνουν πιο κατανοητά στα επόμενα κεφάλαια, κατά την περιγραφή συγκεκριμένων overlays.

Ακόμα, σημαντικό ρόλο στην εφαρμογή των WSN παίζει το κόστος. Μέχρι στιγμής, το κόστος της υποδομής (infrastructure) που απαιτείται για να στηθεί ένα WSN είναι αρκετά υψηλό για τα δεδομένα μικρών (π.χ. προσωπικού χαρακτήρα) εφαρμογών. Και από τις εφαρμογές που θα παρατεθούν γίνεται άλλωστε φανερό ότι επί των πλείστων αφορούν κλάδους ή οργανισμούς με αρκετά μεγάλη υποδομή. Όμως η εξέλιξη των motes δείχνει ότι σύντομα η τεχνολογία των WSN θα είναι διαθέσιμη -και όχι απαγορευτικά ακριβή- για την ανάπτυξη οποιουδήποτε είδους σχετικής εφαρμογής.

Έλεγχος Περιβάλλοντος και Βιοπληροφορική

Τα δίκτυα αισθητήρων μπορούν να τοποθετηθούν σε περιοχές όπου χρειάζεται να ελέγχεται η μόλυνση του περιβάλλοντος (π.χ. χωματερές, εργοστασιακοί χώροι) για να μετρούν ή/και να ρυθμίζουν τα επίπεδα επικίνδυνων ουσιών. Επίσης μπορούν να τοποθετηθούν σε περιοχές όπου υπάρχει ενδιαφέρουσα βιοποικιλότητα, για να μελετηθούν διάφορα είδη φυτών και ζώων. Οι αισθητήρες μπορεί να είναι οσοδήποτε κοντά στα παρατηρούμενα φαινόμενα/ζώα χωρίς να επηρεάζουν ή να γίνονται αντιληπτοί.

Αυτοματισμός Κτιρίων

Η καταγραφή και η παρακολούθηση της θερμοκρασίας, της υγρασίας, του εξαερισμού, των δονήσεων και άλλων φυσικών παραμέτρων εντός ενός κτιρίου μπορεί να οδηγήσει στη δημιουργία συστημάτων ελέγχου με τεράστιο όφελος. Για παράδειγμα, πολύ μεγάλη ποσότητα ενέργειας μπορεί να μη σπαταλάται άσκοπα αν ο έλεγχος των συσκευών που καταναλώνουν ενέργεια γίνεται με βάση τις μετρούμενες τιμές και κάποιες προδιαγραφές, παρά με ανθρώπινες εκτιμήσεις. Έχει αποδειχθεί, άλλωστε, ότι, ειδικά σε μεγάλα κτίρια (π.χ. επιχειρήσεις, οργανισμούς), η σπατάλη ενέργειας είναι πολύ μεγάλη. Ακόμα και το περιβάλλον εργασίας θα μπορούσε να είναι πολύ πιο φιλικό

αν ο αυτοματισμός των κτιρίων γινόταν με κατάλληλα μελετημένες προδιαγραφές. Η χρήση των αισθητήρων στα κτίρια επεκτείνεται με μεγάλο ενδιαφέρον και στον τομέα της ασφάλειας. Συναγερμοί προερχόμενοι είτε από σεισμικές δονήσεις είτε από πιθανές εστίες φωτιάς, είτε από διαρρήξεις θα μπορούσαν να γίνουν πολύ πιο αποδοτικοί και «έξυπνοι» με τη χρήση WSN. Εκτός από την ενσωμάτωση των WSN σε υπάρχοντα κτίρια, υπάρχει η ιδέα της τοποθέτησής τους κατά την κατασκευή των κτιρίων, κάτι που όχι μόνο μπορεί να τα κάνει πιο κομψά και αποδοτικά, αλλά κατά κάποιο τρόπο συμψηφίζει και τα έξοδα κατασκευής (άλλωστε τα έξοδα ενός WSN θα είναι μικρά σε σχέση με τα έξοδα κατασκευής του κτιρίου).

Ιατρική και Νοσοκομειακή Περίθαλψη

Οι αισθητήρες μπορούν να είναι ενσωματωμένοι σε ασθενείς ώστε να μετρούν ουσίες στον οργανισμό τους και ενδεχομένως να τους παρέχουν φάρμακα αυτόματα, όταν η κατάσταση το απαιτεί. Έτσι επιτρέπουν και στους γιατρούς να είναι ενήμεροι για την κατάσταση των ασθενών τους συνεχώς, ακόμα και όταν βρίσκονται μακριά από αυτούς. Ένας άλλος τρόπος χρήσης των WSN στην ιατρική παρουσιάζεται και στο Hourglass [19], όπου αναλύεται ο τρόπος χρησιμοποίησής τους για τη διευκόλυνση των σωστών επιλογών σε καταστάσεις ανάγκης. Δηλαδή, τα ασθενοφόρα αποφασίζουν με βάση τη διαθεσιμότητα δωματίων και γιατρών σε ποιο νοσοκομείο να προσκομίσουν τους ασθενείς, οι γιατροί καλούνται αυτόματα με τις αφίξεις ασθενών ή επιδείνωσης της κατάστασής τους κτλ. Το απόρρητο, βέβαια, των ιατρικών δεδομένων, πρέπει να ληφθεί υπόψη στο σχεδιασμό τέτοιων εφαρμογών.

Έλεγχος Κίνησης και Κυκλοφοριακού Προβλήματος

Πρόκειται για έναν από τους τομείς που έχουν προσελκύσει έντονο ενδιαφέρον από ερευνητικές εφαρμογές (π.χ. CarTel [15]). Αισθητήρες τοποθετημένοι σε σταθερά σημεία του οδικού δικτύου ή/και σε αυτοκίνητα μπορούν να συλλέγουν πληροφορίες σχετικά με την κυκλοφορία και να βοηθούν τους ενδιαφερόμενους χρήστες (οδηγούς) στις αποφάσεις τους. Κάτι τέτοιο θα μπορούσε να βοηθήσει στη λύση του κυκλοφοριακού προβλήματος. Άλλωστε, κάποια από τα στοιχεία που προέκυψαν από σχετική εφαρμογή [15] διέψευσαν κάποιες εμπειρικές απόψεις και απέδειξαν ότι συχνά κάνουμε λάθος επιλογές στο δρόμο.

Αντιμετώπιση Κρίσεων

Η τοποθέτηση πλήθους αισθητήρων σε εκτάσεις όπου έχει συμβεί ή μπορεί να συμβεί μια καταστροφή (π.χ. φωτιές σε δάση, συντρίμμια κτιρίων που έχουν καταρρεύσει) μπορούν να κάνουν πολύ πιο αποδοτική την αντιμετώπιση. Ο στιγμιαίος εντοπισμός των εστιών μιας φωτιάς ή των σημείων ζωής εγκλωβισμένων ανθρώπων μπορεί να βοηθήσει π.χ. την πυροσβεστική υπηρεσία να ανταπεξέλθει σε καταστάσεις που ειδάλλως θα ήταν καταστροφικές.

Μεταφορές και Τροφοδοτική Αλυσίδα

Πολλές φορές, πολύτιμα ή σημαντικά αγαθά/αντικείμενα μεταφέρονται από εταιρίες μεταφορών. Τα αντικείμενα αυτά, μπορεί εκτός των άλλων να είναι ευπαθή ή να απαιτούν συγκεκριμένες συνθήκες. Η τοποθέτηση κατάλληλων αισθητήρων μαζί με τα εν λόγω αντικείμενα μπορεί να επιτρέψει ανά πάσα στιγμή τον έλεγχο της κατάστασης

και της θέσης τους, κάνοντας έτσι ασφαλέστερες τις μεταφορές. Ακόμα και για μικρότερης αξίας αντικείμενα, η γνώση της θέσης τους κατά τη μεταφορά τους ή όταν βρίσκονται σε μεγάλες αποθήκες είναι χρήσιμη.

Έλεγχος Γεωργικών Καλλιεργειών

Πρόκειται για κλάδο που κατεξοχήν χρησιμοποιεί και έχει ανάγκη από τις μετρήσεις φυσικών παραμέτρων. Εκτός αυτού, οι εκτάσεις που πρέπει να παρακολουθούνται στο συγκεκριμένο κλάδο συνηθίζουν να είναι μεγάλες, ως μη ελέγξιμες από ανθρώπους και μόνο. Οι αισθητήρες υγρασίας, θερμοκρασίας, σύνθεσης του εδάφους και ανίχνευσης ουσιών είναι οι κύριοι αισθητήρες που αξιοποιούνται σε τέτοιες εφαρμογές και, εκτός από την προφανή υπηρεσία παρακολούθησης της καλλιέργειας που μπορεί να βοηθήσει στη λήψη αποφάσεων, μπορούν να οδηγήσουν ακόμα και στην κατασκευή συστημάτων που φροντίζουν αυτόματα τις καλλιέργειες (χορηγούν λιπάσματα και άλλες ουσίες όταν και όπου πρέπει κτλ.).

Έμπειρα Συστήματα, Ρομποτική και Μηχανολογία

Στη ρομποτική χρειάζονται εκ των πραγμάτων αισθητήρες και ανάλογα με την έκταση μιας εφαρμογής ρομποτικής, μπορεί να χρειάζονται πολλοί αισθητήρες, είτε στατικοί είτε κινητοί, και η διασύνδεσή τους σε ένα WSN. Παρεμφερής θα ήταν και οποιαδήποτε προσπάθεια αυτοματοποίησης της οδήγησης οχημάτων (κατασκευή αυτόματων πιλότων), με την αναγκαιότητα ύπαρξης αισθητήρων στα οχήματα και στους δρόμους. Εκτός από αυτά τα φανταστικά σενάρια, τα WSN μπορούν ήδη να χρησιμοποιηθούν στη μηχανολογία. Αυτό γίνεται στον τομέα της συντήρησης μηχανημάτων, όπου κατάλληλοι αισθητήρες (π.χ. πίεσης) μπορούν να «ενημερώνουν» για το πότε ένα μηχάνημα χρειάζεται επιδιόρθωση.

Καίρια Ζητήματα σε WSN

Τα ασύρματα δίκτυα αισθητήρων συνδυάζουν πολλές από τις απαιτήσεις παραδοσιακών συστημάτων, αλλά εμφανίζουν και πολλές νέες. Στα περισσότερα από τα ζητήματα δεν υπάρχει μια ενιαία βέλτιστη λύση. Η καλύτερη προσέγγιση εξαρτάται κάθε φορά από το σχεδιασμό και τις ανάγκες του συστήματος. Για παράδειγμα, ανάλογα με το πλήθος των κόμβων, δύο WSN μπορεί να έχουν τελείως διαφορετικές απαιτήσεις όσον αφορά τη δρομολόγηση, την ενέργεια κ.α. Τα δύο ζητήματα που μόλις αναφέρθηκαν (δρομολόγηση και ενέργεια) δεν είναι τυχαία. Είναι αυτά που έχουν απασχολήσει το κύριο μέρος της ερευνητικής δραστηριότητας στον τομέα και έχουν προταθεί πάρα πολλές προσεγγίσεις. Υπάρχουν, όμως, και άλλα ζητήματα τα οποία είναι κοινά σχεδόν για όλα τα WSN και λιγότερο ή περισσότερο πρέπει πάντα να λαμβάνονται υπόψη και συνεπώς να τυγχάνουν και γενικής αντιμετώπισης από τους ερευνητές. Ο σκοπός της ενότητας δεν είναι να περιγράψει συγκεκριμένες προσεγγίσεις (ίσως δοθούν απλά κάποια παραδείγματα) αλλά να σκιαγραφήσει τα ίδια τα προβλήματα. Με τον ένα ή τον άλλο τρόπο πάντως, που θα είναι διαφορετικός ανάλογα με την περίσταση, τα ζητήματα που ακολουθούν θα πρέπει να αντιμετωπίζονται από τους κατασκευαστές των WSN ή τους κατασκευαστές ενδιάμεσου λογισμικού για αυτά.

1.4.1 Δρομολόγηση

Η δρομολόγηση είναι από τα καθοριστικότερα ζητήματα και στα κλασσικά δίκτυα και ορισμένοι αλγόριθμοι από αυτά μπορούν να φανούν χρήσιμοι στα WSN. Παρόλ' αυτά, το γεγονός ότι η επικοινωνία μεταξύ των κόμβων αισθητήρων στα WSN είναι ασύρματη και ότι τα WSN πρέπει να λειτουργούν κατά κανόνα σε «φυσικούς χώρους» (όπου μπορεί να υπάρχουν διάφορα εμπόδια) και όχι π.χ. σε εργαστήρια, αναγκάζει τα WSN να μπορούν να «δανειστούν» αλγόριθμους δρομολόγησης σχεδόν μόνο από τα αδόμητα (ad-hoc) δίκτυα κινητών κόμβων. Επίσης, υπάρχουν και άλλοι λόγοι (π.χ. διαφορετικές δυνατότητες για bandwidth, ο ρόλος της γεωγραφικής θέσης) που οδηγούν στην ανάγκη μελέτης εντελώς νέων αλγορίθμων δρομολόγησης.

Κοινό στοιχείο στις περισσότερες από αυτές τις μελέτες είναι η ανάγκη για δρομολόγηση πολλών βημάτων (multihoping), όπου τα δεδομένα φτάνουν από τον αποστολέα στον παραλήπτη δρομολογούμενα μέσω πολλών άλλων κόμβων αισθητήρων, επιδιώκοντας να ακολουθήσουν μια διαδρομή που είναι βέλτιστη τόσο ως προς την απόδοση όσο και ως προς τη σπαταλούμενη ενέργεια. Ακόμα όμως και το multihoping, που είναι κεντρική έννοια στη δρομολόγηση σε WSN, μπορεί να μην είναι απαραίτητο σε δίκτυα περιορισμένου αριθμού κόμβων.

Το [6] είναι μια πολύ καλή σύνοψη των τεχνικών δρομολόγησης σε WSN και κάνει και μια κατηγοριοποίηση που θα παρουσιάσουμε και εδώ για να γίνει κατανοητός ο τρόπος με τον οποίο πρέπει να «κινηθεί» ο σχεδιαστής της δρομολόγησης που θα υλοποιείται από ένα WSN. Μια τέτοια ανάλυση βοηθά στην επιλογή κατάλληλου πρωτοκόλλου δρομολόγησης για κάθε WSN ή στην υλοποίηση ενός νέου, όποτε αυτό απαιτείται. Μια και αυτή η εργασία ασχολείται περισσότερο με την ανάπτυξη εφαρμογών και ενδιάμεσου λογισμικού, πρέπει να τονιστεί ότι συνήθως το ζήτημα της δρομολόγησης αντιμετωπίζεται από τους δημιουργούς εφαρμογών με τη χρήση έτοιμων πρωτοκόλλων ή ενδιάμεσων λογισμικών χαμηλού επιπέδου. Η γνώση του ζητήματος,

όμως, είναι απαραίτητη ώστε να γίνουν οι κατάλληλες επιλογές και οι πιθανές προσαρμογές.

Η κατηγοριοποίηση των πρωτοκόλλων δρομολόγησης για WSN μπορεί να γίνει είτε με βάση τη δομή του δικτύου (Network Structure) στην οποία αναφέρεται το πρωτόκολλο είτε με βάση τη λειτουργία του πρωτοκόλλου (Protocol Operation).

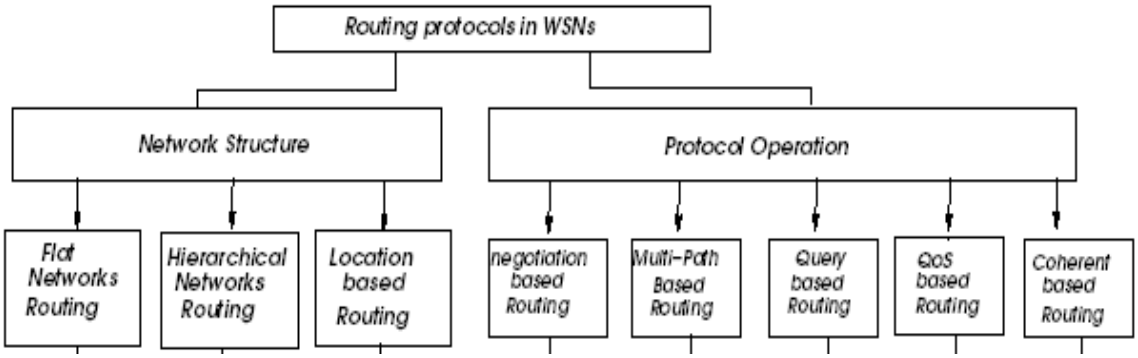
- Με βάση τη δομή του δικτύου υπάρχουν οι εξής κατηγορίες:

- Πρωτόκολλα «επίπεδων» δικτύων (Flat Networks Routing): Τα πρωτόκολλα αυτά θεωρούν όλους τους κόμβους του δικτύου ισοδύναμους, με την ίδια λειτουργικότητα (π.χ. Directed Diffusion [7], [4, κεφ.2], LTP [8], PFR [9]).
- Πρωτόκολλα ιεραρχημένων δικτύων (Hierarchical Networks Routing): Σε αυτά τα πρωτόκολλα οι κόμβοι διαχωρίζονται ανάλογα με τις δυνατότητές τους και έχουν διαφορετικό ρόλο ο καθένας (π.χ. πρωτόκολλο TEEN [4, κεφ.3]).
- Πρωτόκολλα «Γεωγραφικής» δρομολόγησης (Location-based Routing ή Geographic Routing): Σε αυτά τα πρωτόκολλα χρησιμοποιείται η γνώση της θέσης κάθε κόμβου (θεωρείται, δηλαδή, δεδομένη η ύπαρξη των αισθητήρων που παρέχουν τη σχετική γνώση) ώστε η δρομολόγηση να γίνει πιο αποδοτική.

- Με βάση τη λειτουργία που ορίζουν, τα πρωτόκολλα χωρίζονται στις εξής κατηγορίες:

- Πρωτόκολλα που χρησιμοποιούν «διαπραγμάτευση» (Negotiation-based Routing): Αυτά χρησιμοποιούν δεδομένα υψηλού επιπέδου για να διακινούν μόνο χρήσιμα δεδομένα (και όχι π.χ. πολλαπλά ή «νεκρά» δεδομένα), αφότου επικοινωνήσουν («διαπραγματευτούν») με το δίκτυο.
- Πρωτόκολλα πολλαπλών διαδρομών (Multipath-based Routing): Αυτά χρησιμοποιούν πολλαπλά μονοπάτια, δηλαδή πολλαπλή μετάδοση των ίδιων δεδομένων, για να αυξήσουν την αξιοπιστία.
- Πρωτόκολλα βασισμένα σε ερωτήσεις (Query-based Routing): Σε αυτά, η διακίνηση των δεδομένων ξεκινά με τη διάδοση σχετικών queries, έτσι ώστε οι κόμβοι που «μπορούν», να απαντήσουν.
- Πρωτόκολλα ποιότητας υπηρεσιών (“Quality of Service”-based Routing): Σε αυτά κάθε κόμβος εξετάζει συγκεκριμένες μετρικές (ενέργεια, bandwidth κτλ.) πριν από κάθε προώθηση μηνύματος και με βάση αυτές τις μετρικές λαμβάνονται οι αποφάσεις.
- Πρωτόκολλα βασισμένα στη συνέπεια (Coherent-based Routing): Σε αυτά η δρομολόγηση γίνεται με βάση το κατά πόσο οι κόμβοι επεξεργάζονται τα δεδομένα που διακινούν (coherent protocols αν γίνεται επεξεργασία σε κάθε κόμβο, non-coherent protocols αλλιώς).

Στο [6] γίνεται μεγαλύτερη ανάλυση και υποδεικνύονται παραδείγματα για κάθε κατηγορία ενώ από εκεί προέρχεται και η σχηματική κατηγοριοποίηση που ακολουθεί. Το [4] περιγράφει και συγκεκριμένα πρωτόκολλα και η μελέτη κάποιων βασικών από αυτά (όπως π.χ. αυτά των [7], [8], [9]) είναι απαραίτητη για την κατανόηση της δρομολόγησης σε WSN.



ΣΧΗΜΑ 1.1 – Κατηγοριοποίηση των πρωτοκόλλων δρομολόγησης για WSN

1.4.2 **Ενέργεια και Διάρκεια**

Αν και έχουν αρχίσει να αναπτύσσονται ιδιαίτερα οι εφαρμογές μικρών, διασυνδεδεμένων δικτύων αισθητήρων λίγων κόμβων, όπου οι κόμβοι μπορούν να φορτίζονται εύκολα, όπως π.χ. τα κινητά τηλέφωνα, στην πλειοψηφία των εφαρμογών WSN οι κόμβοι έχουν περιορισμένα αποθέματα ενέργειας. Η τοποθέτηση των κόμβων γίνεται με τέτοιο τρόπο και σε τέτοια σημεία ώστε η αντικατάσταση των πηγών ενέργειας (μπαταριών) ή των ίδιων των κόμβων να μην είναι πρακτική. Συνήθως, το τέλος αποθεμάτων ενέργειας ενός κόμβου σημαίνει το τέλος της λειτουργίας του κόμβου μέσα στο WSN και έτσι λέμε ότι τα WSN έχουν περιορισμένη διάρκεια ζωής.

Υπάρχουν δύο βασικές παράμετροι που καθορίζουν το ποσό της καταναλισκόμενης ενέργειας. Η πρώτη έχει σχέση με το υλικό και είναι η ποσότητα ενέργειας για τη μεταφορά ενός μηνύματος από έναν κόμβο σε ένα άλλο. Μετριέται σε Joule/bit. Η δεύτερη έχει να κάνει με τους αλγόριθμους που εφαρμόζονται και αφορά κατά κάποιο τρόπο το ποσό της «περιττής» πληροφορίας που διακινείται. Ως περιττή εννοείται εδώ η ποσότητα των bits της οποίας η διακίνηση θα μπορούσε να έχει αποφευχθεί με τη χρήση ενός «καλύτερου» αλγόριθμου δρομολόγησης ή κάποια μηνύματα που θα μπορούσαν να μην έχουν σταλεί καθόλου αν χρησιμοποιούταν ένας «καλύτερος» αλγόριθμος λειτουργίας. Επίσης σημαντικό είναι να υπάρχει ομοιόμορφη κατανάλωση στους κόμβους έτσι ώστε η διάρκεια ζωής τους να είναι στο τέλος περίπου ίδια. Αυτό αυξάνει προφανώς τη συνολική διάρκεια ζωής του δικτύου. Χαρακτηριστικές μελέτες που οδήγησαν σε αλγόριθμους αποδοτικότερης συμπεριφοράς όσον αφορά την ενέργεια είναι αυτές που βρίσκονται στα [10] (NRG Protocol) και [11].

Όσον αφορά την ενέργεια για τη μετάδοση πληροφορίας από κόμβο σε κόμβο, υπάρχουν πολλές δυνατότητες και η τελική επιλογή για το πόση ενέργεια θα καταναλώνεται με κάθε μήνυμα ανήκει στο σχεδιαστή αλλά όσο λιγότερη επιλεχθεί να είναι αυτή η ενέργεια τόσο χαμηλότερη θα είναι και η ποιότητα και η αξιοπιστία του δικτύου. Άρα, η επιλογή εξαρτάται κυρίως από την εφαρμογή μια και η ένταση των ραδιοκυμάτων (που χρησιμοποιούνται συνήθως) είναι ρυθμιζόμενη. Χαρακτηριστικά αναφέρεται ότι ένα τα Mica motes, ένας πολύ γνωστός τύπος κόμβων, καταναλώνει 21mW κατά την αποστολή και 15mW κατά τη λήψη ενός μηνύματος. Τα σχετικά στοιχεία, καθώς και αρκετές σχετικές έρευνες για διάφορα είδη κόμβων και διάφορες συχνότητες μετάδοσης δείχνουν ότι αυτοί οι ραδιοπομποί έχουν αποδοτικότητα περί του 10%, δηλαδή για να μεταφέρουν κύματα ενός συγκεκριμένου ποσού ενέργειας, χρειάζεται να καταναλώσουν δεκαπλάσια ενέργεια.

Όσον αφορά το σχεδιασμό αλγορίθμων για μείωση της καταναλισκόμενης ενέργειας, οι στόχοι είναι πιο ξεκάθαροι. Όσο λιγότερα τα βήματα του multihoping, τόση λιγότερη ενέργεια καταναλώνεται. Βέβαια, στην επιλογή αλγορίθμου δρομολόγησης υπάρχουν και άλλα κριτήρια εκτός από την κατανάλωση ενέργειας. Ο σχεδιασμός της εφαρμογής πρέπει να γίνει επίσης με προσοχή ώστε να μη συλλέγονται και διακινούνται δεδομένα από τους αισθητήρες συχνότερα από όσο χρειάζεται (π.χ. να γίνει κατάλληλη επιλογή μεταξύ *περιοδικής* (periodic) ή *καθοδηγούμενης από γεγονότα* (event-driven) λήψης δεδομένων από τους αισθητήρες. Ο διαχωρισμός αυτός θα αναλυθεί περισσότερο στην επόμενη ενότητα). Τέλος, η μορφή των μηνυμάτων που ανταλλάσσονται από τους κόμβους, η οποία εξαρτάται και από το λειτουργικό τους, παίζει ρόλο στη συνολική ποσότητα της ενέργειας που καταναλώνεται.

Πρέπει να τονιστεί ότι οι περιορισμοί λόγω της ενέργειας είναι πολύ μεγάλοι και ότι η επίλυση του προβλήματος θα έδινε τεράστια ώθηση στην ανάπτυξη εφαρμογών WSN. Γίνονται συνεχώς νέες προσπάθειες για αξιοποίηση της ηλιακής ενέργειας (βλ. και εικόνα 1.1) καθώς και εφαρμογή άλλων ιδεών για συνεχή φόρτιση των μπαταριών των κόμβων και η ιδανική λύση θα οδηγεί σε πρακτικά άπειρο χρόνο ζωής.

1.4.3 Αναπαράσταση Δεδομένων και Ετερογένεια

Όπως έχει ήδη αναφερθεί, ένας από τους βασικούς στόχους είναι και η διασύνδεση πολλών WSN μεταξύ τους, πιθανόν μέσα από το διαδίκτυο. Όπως και να έχουν τα πράγματα, είτε ο σκοπός είναι να γίνει κάτι τέτοιο άμεσα είτε όχι, ένα WSN πρέπει να διαθέτει τους μηχανισμούς για να υποστηρίξει την ετερογένεια. Η ετερογένεια αυτή μπορεί να υπάρχει είτε σε επίπεδο δικτύου και πρωτοκόλλων ασύρματης επικοινωνίας, είτε σε επίπεδο δεδομένων. Όσον αφορά την επικοινωνία, ένα επιμέρους WSN απαρτίζεται από ίδιας τεχνολογίας κόμβους άρα για να γίνει η διασύνδεση με άλλα δίκτυα αρκεί η ύπαρξη κάποιου gateway που «γνωρίζει» το πρωτόκολλο επικοινωνίας αυτού του WSN.

Η ετερογένεια των δεδομένων είναι κάτι που μπορεί να αντιμετωπιστεί με πολλές προσεγγίσεις. Διαφορετικά είδη κόμβων μπορεί να παράγουν και να υποστηρίζουν διαφορετικούς τύπους δεδομένων, διαφορετικές μονάδες μέτρησης κτλ. Όλες οι προσεγγίσεις «συμφωνούν» ότι μεταξύ της εφαρμογής και των κόμβων των WSN πρέπει να μεσολαβεί ένα «σχήμα» που στα μάτια του προγραμματιστή εφαρμογών να ενοποιεί τις προδιαγραφές των WSN. Έτσι, γνωρίζοντας αυτό το σχήμα, ο προγραμματιστής μπορεί να αγνοεί την τεχνολογία του κάθε επιμέρους WSN. Δε θα γίνει εδώ περαιτέρω ανάλυση μια και ένα τέτοιο σχήμα υπάρχει σχεδόν σε κάθε ένα από τα overlays που θα περιγραφούν στο υπόλοιπο της εργασίας, καθώς και σε αυτό που υλοποιήθηκε στα πλαίσια της εργασίας.

Είναι πλέον αποδεκτό ότι για τη δημιουργία τέτοιων σχημάτων, η καλύτερη προσέγγιση είναι οι περιγραφές με γλώσσες βασισμένες σε XML. Αυτό προσδίδει σε ένα WSN αυτό που ονομάζεται αυτό-περιγραφικότητα (auto-configuration) και, εκτός του ότι απαιτείται για την υποστήριξη της ετερογένειας, βοηθά και στην επεκτασιμότητα του WSN, που σχολιάζεται στη συνέχεια.

Η ετερογένεια, όμως, δεν περιορίζεται σε αυτές τις λεπτομέρειες. Είναι κάτι που πρέπει να λαμβάνεται υπόψη από γενική σκοπιά και με όλες τις παραμέτρους της. Για παράδειγμα, ένα σύνολο διασυνδεδεμένων WSN μπορεί να περιλαμβάνει δίκτυα αραιών

κόμβων και δίκτυα πυκνών κόμβων, δίκτυα πολλών κόμβων και δίκτυα λίγων κόμβων, δίκτυα κινούμενων κόμβων και δίκτυα στατικών κόμβων, δίκτυα με επαρκή ενέργεια και άλλα με περιορισμούς ενέργειας. Αυτά δεν αρκεί να λαμβάνονται υπόψη από το σχεδιαστή των πρωτοκόλλων δρομολόγησης ή από αυτόν που επιλέγει ένα τέτοιο για κάθε WSN. Για προφανείς λόγους παίζουν κρίσιμο ρόλο και στην υλοποίηση εφαρμογών.

1.4.4 Συντήρηση και Επεκτασιμότητα

Τα WSN είναι κατά κανόνα δυναμικά συστήματα με την έννοια ότι ανά πάσα στιγμή μπορεί να προστεθούν ή να αφαιρεθούν κόμβοι, είτε οικειοθελώς είτε όχι. Οι περιορισμοί ενέργειας, ο χρόνος ζωής αλλά και η πιθανή προσθήκη κόμβων μπορούν να αλλάξουν την κατάσταση έτσι ώστε επιλογές που είχαν γίνει όσον αφορά τα ζητήματα που συζητήθηκαν στις τρεις προηγούμενες παραγράφους να πρέπει να προσαρμοστούν. Άρα τα πρωτόκολλα πρέπει να είναι ευέλικτα και να μην είναι σχεδιασμένα «σφικτά» συνδεδεμένα με τις ανάγκες μιας συγκεκριμένης δομής.

Επίσης, η δυνατότητα επέκτασης μπορεί να αφορά είτε τα επιμέρους WSN (με την προσθήκη νέων κόμβων) είτε ένα σύνολο διασυνδεδεμένων WSN (με την προσθήκη νέων WSN). Συνεπώς, αυτές οι προοπτικές πρέπει να λαμβάνονται υπόψη από τα χρησιμοποιούμενα σχήματα.

1.4.5 Συνδεσιμότητα και Κίνηση

Από το πιο καίρια ζητήματα που επίσης θα συζητηθούν σχεδόν σε κάθε overlay είναι το πρόβλημα της πιθανής ελλιπούς / διακοπτόμενης σύνδεσης (intermittent connectivity). Και εδώ, το πρόβλημα μπορεί να αφορά είτε τους κόμβους αισθητήρων (κυρίως), είτε τους ίδιους τους gateways που διασυνδέουν τα WSN. Το ζήτημα είναι άμεσα συνδεδεμένο με την ύπαρξη ή όχι κινητών κόμβων, μια και οι στατικοί συνήθως δεν εμφανίζουν τέτοια προβλήματα.

Το ζήτημα της κίνησης επεκτείνεται και σε άλλα θέματα εκτός της ελλιπούς συνδεσιμότητας (κυρίως όταν η γεωγραφική θέση παίζει ρόλο στην εφαρμογή) και είναι ως εκ τούτου απαραίτητο, πριν γίνει οποιοσδήποτε σχεδιασμός μιας εφαρμογής ή ενός overlay, να εξετάζεται αν στόχος είναι η υποστήριξη της κίνησης των κόμβων και/ή των gateways.

Οι συχνότερες προσεγγίσεις για την αντιμετώπιση διακοπτόμενης συνδεσιμότητας είναι η προσωρινή αποθήκευση (buffering), τα πολλαπλά αντίγραφα (replication) ή οι διάφορες τεχνικές εντοπισμού των αποσυνδεδεμένων κόμβων και η αγνόηση του προβλήματος. Αυτά θα φανούν και στα συστήματα των επόμενων κεφαλαίων ενώ μια ιδέα για εξειδικευμένα πρωτόκολλα αντιμετώπισης του προβλήματος της κίνησης μπορεί να πάρει ο αναγνώστης από τα [12] και [13].

Υπάρχουν και άλλα ζητήματα που είναι σημαντικά για την πλειοψηφία των WSN αλλά αυτά είναι που πρέπει να λαμβάνονται υπόψη καταρχήν σε οποιοδήποτε τέτοιο σύστημα. Άλλα ζητήματα (π.χ. τοπικότητα, αξιοποίηση ισχυρών δομών) θα ανακύψουν και θα συζητηθούν σε επιμέρους παραδείγματα.

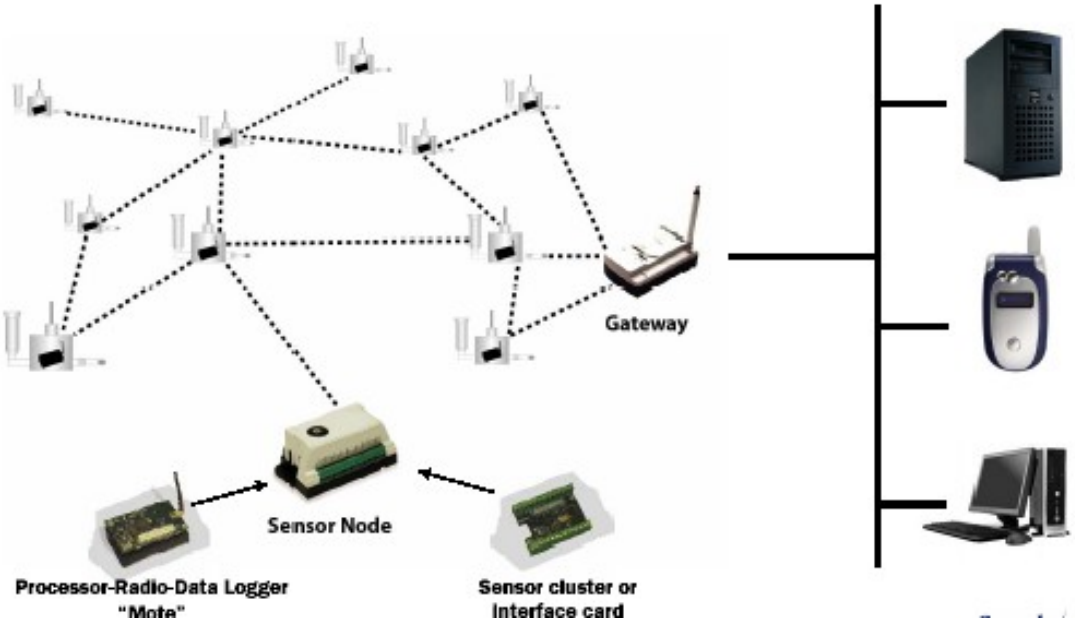
Αρχιτεκτονικές για WSN και Overlays

Οι αρχιτεκτονικές των WSN μπορούν να οριστούν και να κατηγοριοποιηθούν με βάση πολλά διαφορετικά κριτήρια. Μια και αυτή η εργασία ασχολείται με την κατασκευή ενδιάμεσου λογισμικού (overlay) για τη διασύνδεση πολλών επιμέρους WSN, οι αρχιτεκτονικές που έχουν ιδιαίτερη σημασία είναι οι αρχιτεκτονικές διασύνδεσης των επιμέρους WSN. Αξίζει όμως, αφού ξεκαθαρίσαμε τη διαφορά, να δούμε λίγο και τις αρχιτεκτονικές που μπορεί να έχει ένα επιμέρους WSN ανάλογα με την εσωτερική δομή των κόμβων του. Άλλωστε, ένα ολοκληρωμένο overlay πρέπει μεν να μπορεί να υποστηρίξει οποιαδήποτε από αυτές τις δομές ή και όλες ταυτόχρονα αλλά η γνώση των πιθανών αυτών δομών μπορεί να είναι καθοριστική για το σχεδιασμό του. Τόσο όσον αφορά τις αρχιτεκτονικές των κόμβων όσο και όσον αφορά τις αρχιτεκτονικές διασύνδεσης των WSN, η αυστηρή κατηγοριοποίηση θα ήταν χονδροειδής, μια και οι τελικές υλοποιήσεις είναι συνήθως είτε υβριδικές είτε ενδιάμεσες λύσεις.

Η εισαγωγή των overlays επιλέχθηκε να γίνει στη συνέχεια της ίδιας ενότητας, μια και είναι άμεσα συνδεδεμένα με τις αρχιτεκτονικές διασύνδεσης και στην ουσία αποτελούν μια αφαίρεση της αρχιτεκτονικής του συστήματος «στα μάτια» του προγραμματιστή εφαρμογών για WSN.

1.5.1 **Αρχιτεκτονικές Κόμβων ενός WSN**

Πρόκειται για τους τρόπους με τους οποίους μπορεί να γίνει η διάταξη και η επικοινωνία των κόμβων αισθητήρων (sensor nodes). Στην παρακάτω εικόνα φαίνεται το τμήμα του δικτύου το οποίο αφορούν αυτές οι αρχιτεκτονικές.



ΕΙΚΟΝΑ 1.2 – Γενική εικόνα ενός μεμονωμένου WSN. Οι διάφορες προτεινόμενες αρχιτεκτονικές κόμβων αφορούν τη διάταξη και επικοινωνία των sensor nodes.

- Δίκτυα λίγων (ή/και ενσωματωμένων) κόμβων:

Υπάρχει το ενδεχόμενο (που θα δούμε τελικά ότι είναι πολύ πρακτικό και μπορεί να υποστηρίξει πολλές εφαρμογές) οι κόμβοι να επικοινωνούν όλοι απευθείας με το σταθμό-βάση (gateway) ή ακόμα και να είναι κατά κάποιο τρόπο ενσωματωμένοι σε αυτόν. Αυτή η περίπτωση λύνει το πρόβλημα της ενέργειας (με επιτόπου φόρτιση) και εξαλείφει την ανάγκη για πολύπλοκη δρομολόγηση αλλά δεν παρέχει την κάλυψη που είναι απαραίτητη για ορισμένες εφαρμογές. Η διασύνδεση, όμως, πολλών τέτοιων μικρών δικτύων μπορεί, όπως θα δούμε στα επόμενα κεφάλαια, να υποστηρίξει εξαιρετικά ενδιαφέρουσες εφαρμογές.

- Δίκτυα κινούμενων κόμβων:

Εδώ πρέπει να τονιστεί ότι οι ασύρματοι κόμβοι είναι ούτως ή άλλως κινητοί. Το ζήτημα είναι αν αναμένεται να κινούνται συνεχώς για τις ανάγκες του δικτύου ή όχι. Στην περίπτωση αυτή χρησιμοποιούνται και τεχνικές των ad-hoc δικτύων αλλά αυτό δε σημαίνει ότι και αυτά τα δίκτυα δεν ανήκουν σε μια από τις αρχιτεκτονικές mesh, star ή υβριδικές τους, που περιγράφονται στη συνέχεια. Άλλωστε, επαναλαμβάνουμε ότι οι αρχιτεκτονικές αυτές δε σχετίζονται τόσο με την τοπολογία όσο με τον τρόπο επικοινωνίας και διάδοσης μηνυμάτων.

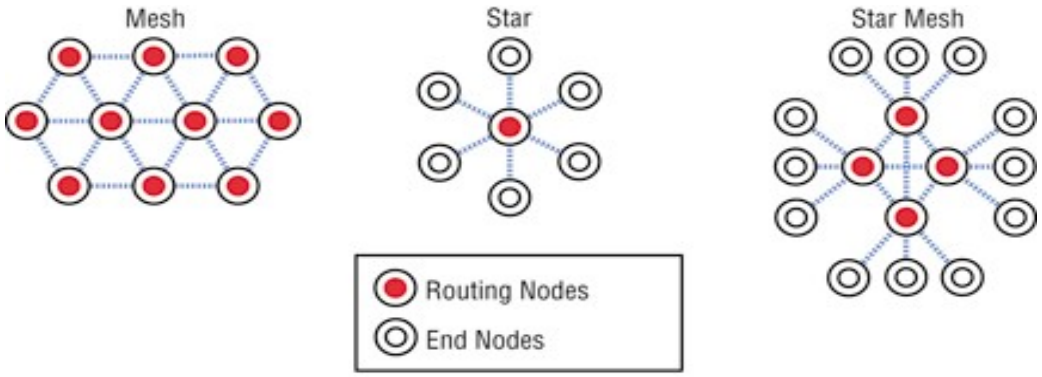
- Δίκτυα αρχιτεκτονικής mesh:

Σε αυτά, η διάδοση των μηνυμάτων μπορεί να γίνει μεταξύ οποιωνδήποτε επικοινωνούντων κόμβων, των οποίων οι συνδέσεις σχηματίζουν κάτι σαν πλέγμα, ανάλογα με τις ανάγκες της δρομολόγησης.

- Δίκτυα αρχιτεκτονικής star:

Σε αυτά, μόνο κάποιοι κόμβοι υλοποιούν τη δρομολόγηση ενώ οι υπόλοιποι επικοινωνούν με έναν από αυτούς κάνοντας απλά sensing. Αυτό, όπως φαίνεται και στην εικόνα, δημιουργεί μια τοπολογία αστέρα.

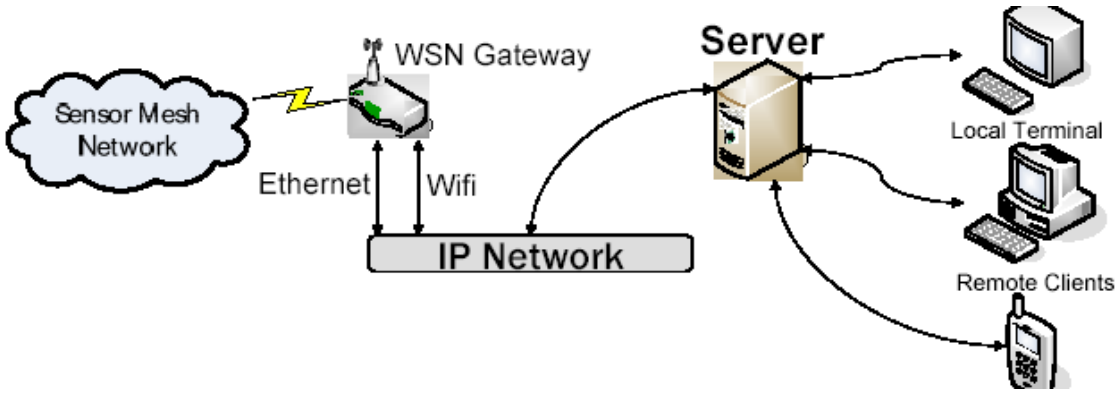
Φυσικά, συναντώνται και άλλες αρχιτεκτονικές, γνωστές από τα γενικά δίκτυα αλλά αυτές είναι οι πιο συχνά χρησιμοποιούμενες στα WSN. Το σχήμα που ακολουθεί προέρχεται, άλλωστε, από το site της εταιρίας κατασκευής εμπορικών WSN «Dust Networks», όπου αυτές αναφέρονται ως κύριες αρχιτεκτονικές.



ΣΧΗΜΑ 1.2 – Mesh, Star και Star-Mesh αρχιτεκτονικές κόμβων

1.5.2 **Αρχιτεκτονικές Διασύνδεσης WSN**

Πρόκειται για τον τρόπο με τον οποίο μπορούν να συνδεθούν μεταξύ τους πολλά WSN, όπως αυτό της εικόνας 1.2. Οι αρχιτεκτονικές, αυτές, λοιπόν, βρίσκονται ένα επίπεδο παραπάνω. Ακολουθεί μια πολύ γενική εικόνα, που καθιστά κατανοητό το επίπεδο για το οποίο μιλάμε.



ΕΙΚΟΝΑ 1.3 – Γενική εικόνα σύνδεσης WSN σε IP δίκτυο (π.χ. Internet). Φυσικά, μπορούν να διασυνδεθούν οσαδήποτε WSN ενώ το δεξί μέλος είναι απλά ένα παράδειγμα για τη δημιουργία ολοκληρωμένου συστήματος χρήσης WSN.

Είναι σαφές ότι ένα ολοκληρωμένο δίκτυο που χρησιμοποιεί WSN μπορεί να έχει πολλές διαφορετικές μορφές και η αυστηρή κατηγοριοποίηση θα ήταν εδώ άστοχη. Δύο σημεία που διαχωρίζουν τα συστήματα αυτά και σηματοδοτούν διαφορετικές προσεγγίσεις κατά τη δημιουργία σχετικών overlays, είναι ο τρόπος οργάνωσης των μελών του δικτύου (είτε peer-to-peer είτε «πολύ-επίπεδος») και η ανάκτηση των δεδομένων (είτε περιοδική είτε οδηγούμενη από γεγονότα).

Στις αρχιτεκτονικές Ν-επιπέδων, ο ρόλος των διαφόρων μερών του συστήματος είναι καθαρά διαχωρισμένος (επίπεδο αισθητήρων, επίπεδο server, επίπεδο client, επίπεδο βάσης δεδομένων κτλ. ανάλογα με την περίπτωση) και οι ευθύνες του κάθε επιπέδου είναι σαφείς. Στις peer-to-peer αρχιτεκτονικές, τα διάφορα μέρη οργανώνονται σε οντότητες που η κάθε μία έχει σχετικά ολοκληρωμένη λειτουργικότητα («ισότιμη», όπως λέγεται, με κάθε άλλη οντότητα). Ο διαχωρισμός των δύο αυτών προσεγγίσεων είναι ήδη ευρέως γνωστός. Οι περαιτέρω διαφορές τους θα φανούν μέσω των παραδειγμάτων των επόμενων κεφαλαίων. Άλλωστε, ελάχιστες αρχιτεκτονικές ακολουθούν ξεκάθαρα ή τη μία ή την άλλη προσέγγιση. Τα στοιχεία τους συχνά αναμιγνύονται και ο διαχωρισμός δεν είναι τόσο ξεκάθαρος.

Όσον αφορά την ανάκτηση των δεδομένων, τα πράγματα είναι λίγο πιο ξεκάθαρα. Μπορεί οι κόμβοι αισθητήρων να είναι προγραμματισμένοι ώστε να παίρνουν μετρήσεις περιοδικά και να τις αναφέρουν έτσι ώστε να γίνεται η καταγραφή τους σε μια βάση δεδομένων. Τα υπόλοιπα μέρη του συστήματος δεν έχουν καμία αλληλεπίδραση με τους αισθητήρες και ό, τι δεδομένα χρειάζονται, τα διαβάζουν μόνο από τις βάσεις δεδομένων. Αυτή είναι η περιοδική (periodic) λήψη δεδομένων. Εναλλακτικά, μπορεί οι ανάγκες του προγράμματος να πυροδοτούν τη λήψη δεδομένων. Να στέλνονται, δηλαδή, εντολές στους κόμβους αισθητήρων, ώστε να γίνουν επιτόπου οι επιθυμητές μετρήσεις. Αυτή η προσέγγιση λέγεται οδηγούμενη από τα γεγονότα (event-driven) και είναι πιο

δύσκολη στην υλοποίηση αλλά αποδοτικότερη όσον αφορά την ενέργεια και ίσως αποτελεσματικότερη για κάποιες real-time εφαρμογές. Υπάρχουν, όμως, και άλλοι παράγοντες που καθορίζουν τα παραπάνω.

1.5.3 Overlays

Τόσο την επικοινωνία μεταξύ των μερών του συστήματος και τον τρόπο ανάκτησης των δεδομένων όσο και πολλές άλλες λεπτομέρειες, ιδίως τις τεχνικές λεπτομέρειες των WSN, ο προγραμματιστής εφαρμογών προτιμά να μην τις αντιμετωπίζει άμεσα κάθε φορά. Το ενδιάμεσο λογισμικό (overlay) για WSN είναι επαναχρησιμοποιήσιμο λογισμικό που ενθυλακώνει την περιγραφή των τεχνικών λεπτομερειών και του τρόπου επικοινωνίας ώστε να δώσει στο χρήστη του μια προγραμματιστική διεπαφή (API) με κλήσεις υψηλότερου επιπέδου.

Λίγα overlays έχουν αναπτυχθεί για συστήματα WSN και κάθε ένα ταιριάζει περισσότερο σε συγκεκριμένες χρήσεις. Στο κεφάλαιο 2 περιγράφονται κάποια από αυτά που χρησιμοποιούν την αρχιτεκτονική Ν-επιπέδων, στο κεφάλαιο 3 κάποια που χρησιμοποιούν peer-to-peer προσέγγιση ενώ στη συνέχεια θα παρουσιαστεί μια νέα πρόταση που υλοποιήθηκε στα πλαίσια της εργασίας και έχει κάπως διαφορετική λογική από τα υπάρχοντα overlays.

2. ΕΝΔΙΑΜΕΣΟ ΛΟΓΙΣΜΙΚΟ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ Ν – ΕΠΙΠΕΔΩΝ

Εισαγωγή

Η συνύπαρξη πολλών τεχνολογιών και συσκευών μέσα σε ένα σύστημα ασύρματων δικτύων αισθητήρων καθιστά το σύστημα πολύ-επίπεδο από τη φύση του. Παρά το γεγονός ότι τα overlays μπορούν να «οργανώσουν» το σύστημα έτσι ώστε η διασύνδεση των WSN να γίνεται με peer-to-peer λογική, η πιο φυσική και «αυθόρμητη» προσέγγιση είναι αυτή των πολλών επιπέδων. Παραδείγματα επιπέδων θα δοθούν μέσα από τις περιγραφές συγκεκριμένων overlays. Τα επίπεδα έχουν σε κάθε περίπτωση διαφορετικό νόημα και στόχος τους είναι πάντα η ενθυλάκωση εννοιών και ευθυνών που είναι κοινές για κάποια μέρη του συστήματος.

Στη συνέχεια του κεφαλαίου θα περιγραφούν τρία συστήματα που το καθένα έχει τις δικές του ιδιαιτερότητες. Το CarTel [15] αναπτύχθηκε στο MIT και είναι στην πράξη από τα πληρέστερα συστήματα, το MetroSense [16] αναπτύχθηκε από το Dartmouth College και το πανεπιστήμιο της Columbia και είναι από τα πιο ενδιαφέροντα, με μια ιδιαίτερη θεωρητική προσέγγιση ενώ το jWebDust [17, 18] αναπτύχθηκε στο Πανεπιστήμιο Πατρών και είναι από τα χαρακτηριστικότερα παραδείγματα της συγκεκριμένης αρχιτεκτονικής, με ξεκάθαρο διαχωρισμό των επιπέδων και των ευθυνών / λειτουργιών που έχει το καθένα από αυτά.

Ο τρόπος παρουσίασής τους επιλέχθηκε ώστε να ευνοεί την κριτική προσέγγιση και μελέτη του καθενός και να οδηγεί σε εύκολη σύγκριση και συμπεράσματα. Κάθε ένα από αυτά, όπως και κάθε σύστημα του επόμενου κεφαλαίου, παρουσιάζεται σε τέσσερεις ενότητες. Τη γενική ιδέα, τις ιδιαιτερότητες-καινοτομίες του, την περιγραφή του συστήματος και την εφαρμογή του, με τα σχετικά συμπεράσματα. Αυτό ευνοεί την κατανόησή τους και βάζει και τις βάσεις για μια σύντομη ανακεφαλαίωση και σύγκριση που θα γίνει στο κεφάλαιο 4, πριν την παρουσίαση του συστήματος που υλοποιήθηκε.

CarTel

Η Γενική Ιδέα

Το CarTel είναι χαρακτηριστικό παράδειγμα N-tier αρχιτεκτονικής, αφού ολόκληρη η λειτουργικότητα της κάθε εφαρμογής, αλλά και η σχετική βάση δεδομένων, εγκαθίστανται στη συσκευή χρήστη (portal, στην ορολογία του CarTel). Όλα τα υπόλοιπα τμήματα της αρχιτεκτονικής ασχολούνται με τη μεταφορά των δεδομένων προς τις βάσεις δεδομένων των εφαρμογών και δε συνεργάζονται με τις ίδιες τις εφαρμογές. Δεν προβλέπεται, λοιπόν, ούτε η ύπαρξη κατανεμημένων εφαρμογών, ούτε κατανεμημένων βάσεων δεδομένων για τις εφαρμογές του CarTel, αν και φυσικά όλα αυτά μπορούν να κατασκευαστούν on top, με τη χρήση άλλων τεχνολογιών.

Η αρχιτεκτονική του CarTel περιλαμβάνει μικρά και απλά δίκτυα αισθητήρων, δεδομένου ότι κάθε δίκτυο αισθητήρων βρίσκεται ολόκληρο σε μια μικρή περιοχή (κατά προτίμηση σε ένα αυτοκίνητο). Και αυτό ισχύει όχι μόνο για τους αισθητήρες αλλά και για το σταθμό-βάση τους (gateway), γεγονός που σημαίνει ότι κάλλιστα μπορεί να μη χρειάζεται καν ασύρματη επικοινωνία μεταξύ αισθητήρων και σταθμού-βάσης, αν και φυσικά υποστηρίζεται. Η ασύρματη επικοινωνία στην οποία επικεντρώνεται το σύστημα είναι η επικοινωνία των σταθμών-βάσεων με σημεία πρόσβασης στο διαδίκτυο.

Η ιδέα πίσω από το CarTel δεν είναι τόσο η διασύνδεση υπαρχόντων, πιθανώς στατικών δικτύων αισθητήρων (άλλωστε η αρχιτεκτονική του μοιάζει κάπως «ακριβή», όσον αφορά τους απαιτούμενους αισθητήρες, τους απαιτούμενους gateways και το απαιτούμενο λογισμικό) όσο η αξιοποίηση της δυνατότητας που παρέχουν τα οχήματα για την κάλυψη ευρέων περιοχών. Πρόκειται για μια επαναχρησιμοποιήσιμη πλατφόρμα για την κατασκευή εφαρμογών κινητών (σε μεγάλες αποστάσεις και ταχύτητες) αισθητήρων που παρέχει τα εξής:

- Απλή προγραμματιστική διεπαφή. Στόχος είναι η *κεντρικοποίηση* και απλοποίηση της δημιουργίας εφαρμογών με κινητούς αισθητήρες. Η κίνηση και η κατανεμημένη μορφή συλλογής των δεδομένων αποκρύπτεται από το δημιουργό εφαρμογών.
- Χειρισμό πολλών, ετερογενών αισθητήρων. Δεν υπάρχουν περιορισμοί στους τύπους δεδομένων που χρησιμοποιούν οι αισθητήρες και νέοι τύποι αισθητήρων μπορούν να ενσωματωθούν εύκολα. Παρόλ' αυτά, ο χειρισμός αυτός, δηλαδή η προσωρινή αποθήκευση (buffering) και η επεξεργασία δεδομένων που σχετίζονται με τους αισθητήρες πρέπει να γίνεται στους κινητούς κόμβους, αλλιώς το bandwidth των συνδέσεων δε θα ήταν αρκετό. Γι' αυτό και, όπως προαναφέρθηκε, οι gateways βρίσκονται στους κινητούς κόμβους.
- Χειρισμό διακοπτόμενης σύνδεσης (intermittent connectivity), εφόσον ο κύριος τρόπος σύνδεσης των κινητών κόμβων στο διαδίκτυο είναι μέσω τυχαίων ασύρματων σημείων πρόσβασης (Wi-Fi, Bluetooth). Η ιδέα του CarTel βασίζεται στην υπόθεση ότι τα Wireless Access Points χρηστών από το σπίτι θα είναι στο μέλλον αρκετά ώστε να προσφέρουν επαρκώς συχνή σύνδεση στους κινητούς κόμβους του CarTel, τουλάχιστον σε αστικές περιοχές.

Ιδιαιτερότητες – Καινοτομίες

Το CarTel είναι μία από τις λίγες πλατφόρμες που αντιμετωπίζουν συνολικά (δηλαδή σε όλα τα επίπεδά του) το ζήτημα της κατασκευής κινητών δικτύων αισθητήρων. Σε σχέση με τα γενικά ζητήματα του τομέα, το CarTel παρουσιάζει καινοτομικές υλοποιήσεις όσον αφορά την ανεκτική σε καθυστερήσεις δικτύωση (delay-tolerant networking) και την επεξεργασία ερωτημάτων (query processing), ενώ σε σχέση με άλλες πλατφόρμες (overlays) υλοποίησης εφαρμογών δικτύων αισθητήρων παρουσιάζει κυρίως τρεις ιδιαιτερότητες. Ακολουθεί περιγραφή, πρώτα του delay-tolerant networking και του query processing και στη συνέχεια αυτών των ιδιαιτεροτήτων.

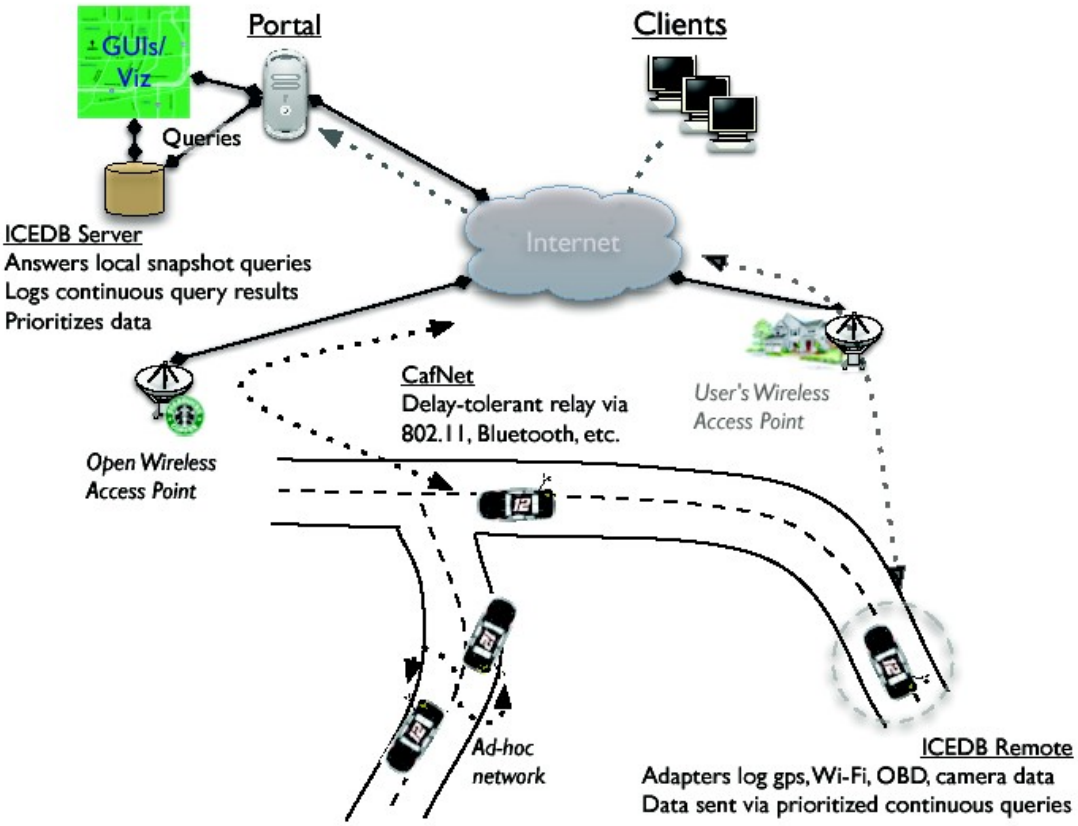
Το delay-tolerant networking συχνά γίνεται προσπάθεια να επιτευχθεί με τη χρήση πολλαπλών μονοπατιών στην αποστολή των δεδομένων ή με το να ζητούνται τα ίδια δεδομένα από πολλές διαφορετικές πηγές (replication). Αυτές οι προσεγγίσεις δεν παρέχουν ούτε απόλυτη βεβαιότητα ολοκλήρωσης της αποστολής / παραλαβής δεδομένων ούτε την εξασφαλισμένη εγκυρότητά τους. Η προσέγγιση που ακολουθεί το CarTel είναι η προσωρινή αποθήκευση (buffering) και θα αναλυθεί στην περιγραφή του πρωτοκόλλου CafNet από το οποίο και υλοποιείται.

Όσον αφορά το query processing, το CarTel προσφέρει ουσιαστικά μια νέα γλώσσα επικοινωνίας με τη βάση (τροποποίηση της SQL) για να καλύψει τις ιδιαιτερότητες των κινητών δικτύων αισθητήρων. Περιγράφεται στο πρωτόκολλο ICEDB.

Οι τρεις ιδιαιτερότητες του CarTel σε σχέση με παρόμοιου σκοπού overlays είναι η έντονη τοπική επεξεργασία των δεδομένων (γίνεται σχεδόν εξ' ολοκλήρου «κοντά» στους αισθητήρες, δηλαδή από το σταθμό-βάση τους.), η πρωτότυπη οπτικοποίηση των δεδομένων (χρήση GoogleMaps και αλληλεπιδραστικού GUI) καθώς και η λειτουργία του συστήματος όταν οι ταχύτητες των δικτύων αισθητήρων είναι τόσο υψηλές (ταχύτητες οχημάτων, που σημαίνει συνεχείς αποσυνδέσεις και επανασυνδέσεις με το δίκτυο).

Το Σύστημα

Το σύστημα CarTel έχει τρία κύρια συστατικά. Το ICEDB, το CafNet και το Portal. Τα συστατικά αυτά δεν πρέπει να συγχέονται με τα επίπεδα της αρχιτεκτονικής του δικτύου, μια και υλοποιούνται σε πολλά επίπεδα (π.χ. το ICEDB και το CafNet υλοποιούνται τόσο στον host της εφαρμογής όσο και στους σταθμούς-βάσεις των κινητών δικτύων αισθητήρων). Η επόμενη εικόνα είναι μια γενική εικόνα της αρχιτεκτονικής του δικτύου ενός συστήματος CarTel ενώ στη συνέχεια ακολουθεί περιγραφή των τριών κύριων συστατικών.



ΕΙΚΟΝΑ 2.1 – Η αρχιτεκτονική του CarTel. Τα αυτοκίνητα συλλέγουν δεδομένα καθώς κινούνται και τα καταγράφουν στις τοπικές βάσεις δεδομένων ICEDB. Όταν η σύνδεση γίνεται εφικτή, τα δεδομένα μεταφέρονται μέσω του CafNet στο Portal, όπου οι χρήστες μπορούν να τα προσπελάσουν μέσω οπτικών διεπαφών κτλ.

ICEDB

Το ICEDB κατανέμει την εκτέλεση των ερωτήσεων (queries) και την αποστολή των δεδομένων μεταξύ του ICEDB Server που βρίσκεται στο Portal και του ICEDB Remote που βρίσκεται στα αυτοκίνητα. Ο Server δέχεται συνεχείς ερωτήσεις από την εφαρμογή και τις προωθεί μέσω του CafNet στα ICEDB Remotes των οχημάτων. Εκεί γίνεται η επεξεργασία των δεδομένων και η αποστολή τελικών αποτελεσμάτων πίσω στον Server, πάλι με χρήση του CafNet. Ιδιαίτερη σημασία έχει ότι η -βασισμένη σε SQL- γλώσσα που χρησιμοποιεί το ICEDB επιτρέπει τον καθορισμό προτεραιοτήτων για τα αποτελέσματα, έτσι ώστε αν κατά την επόμενη σύνδεση δεν προλάβουν να σταλούν όλα, να σταλούν τα πιο σημαντικά. Οι προτεραιότητες αυτές μπορούν, μάλιστα, να αλλάζουν δυναμικά.

Για να υποστηρίζεται ετερογένεια των τύπων δεδομένων που χρησιμοποιούν οι αισθητήρες και για να είναι εύκολη η προσθαφαίρεση αισθητήρων, χρησιμοποιούνται πακέτα μετα-δεδομένων (meta-data) για την περιγραφή των αισθητήρων. Μια τέτοια περιγραφή μαζί με ένα εκτελέσιμο πρόγραμμα που (αφότου κάνει ανάλυση (parsing) των

μετα-δεδομένων και καταχωρήσει στη βάση δεδομένων του κατάλληλους πίνακες και πεδία) εκκινεί τη συλλογή δεδομένων ονομάζεται προσαρμογέας (adapter). Αυτοί οι προσαρμογείς μπορούν να οριστούν και να φορτωθούν προγραμματιστικά, από τις εφαρμογές στους σταθμούς-βάσεις των οχημάτων.

Η γλώσσα που χρησιμοποιείται από το ICEDB είναι SQL με επεκτάσεις για να υποστηριχθούν οι συνεχείς ερωτήσεις (continuous queries) και ο καθορισμός προτεραιοτήτων (prioritization). Για να υποστηριχθούν οι συνεχείς ερωτήσεις έχει προστεθεί μια πρόταση RATE, που δηλώνει το ρυθμό λήψης δεδομένων, όπως φαίνεται στο παρακάτω παράδειγμα:

```
SELECT carid,traceid,time,location FROM gps
  WHERE gps.time BETWEEN now()-1 mins AND now()
  RATE 5 mins
```

Ο καθορισμός προτεραιοτήτων είναι κάτι πιο πολύπλοκο και γίνεται σε δύο επίπεδα. Τόσο κατά τη δημιουργία και εκτέλεση ενός query από τον Server, όσο και, δυναμικά, σε κάθε σύνδεση του οχήματος με το δίκτυο. Η πρώτη περίπτωση ονομάζεται τοπικός καθορισμός προτεραιοτήτων (local prioritization) ενώ η δεύτερη ονομάζεται καθολικός καθορισμός προτεραιοτήτων (global prioritization) αφού λαμβάνονται υπόψη όλα τα αποτελέσματα των οποίων η αποστολή εκκρεμεί και όχι μόνο τα αποτελέσματα ενός συγκεκριμένου query.

Το local prioritization προσθέτει στο query δύο προτάσεις:
- Η πρόταση PRIORITY αναθέτει στο query μια προτεραιότητα. Είναι απλά ένας αριθμός που δείχνει τη βαρύτητα του query.
- Η πρόταση DELIVERY ORDER μοιάζει με την παραδοσιακή ORDER BY της SQL και καθορίζει το πώς θα καταταχθούν τα αποτελέσματα του συγκεκριμένου query.

Επαναλαμβάνεται ότι αυτά καθορίζονται εξ' αρχής, στον Server του Portal, και αφορούν ένα συγκεκριμένο query.

Το global prioritization γίνεται με κάπως πιο περίεργο τρόπο. Η διαδικασία του ξεκινά από τα οχήματα και όχι από το Portal. Όταν ένα όχημα συνδέεται, στέλνει μια περίληψη των αποτελεσμάτων που έχει συγκεντρώσει προς αποστολή (πρόκειται για αποτελέσματα πολλών διαφορετικών queries). Ο Server, λαμβάνοντας αυτές τις περιλήψεις από όλα τα οχήματα, έχει γενική εικόνα των αποτελεσμάτων που πρόκειται να παραλάβει και αποφασίζει τότε τη σειρά με την οποία τα θέλει. Στέλνει, λοιπόν, νέες, καθολικές προτεραιότητες στα οχήματα, τα οποία με τη σειρά τους ανακατατάσσουν με βάση τα νέα κριτήρια τα αποτελέσματα που εκκρεμούν.

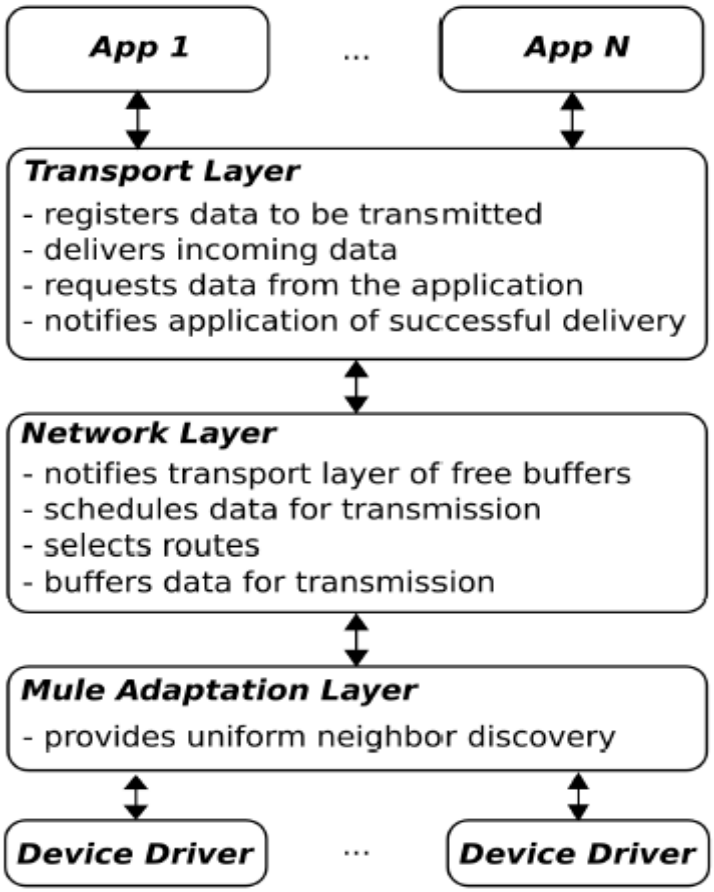
CafNet

Το CafNet είναι μια γενικής χρήσης δικτυακή στοίβα (network stack) για την επικοινωνία εν μέσω διακοπτόμενων συνδέσεων. Όλοι οι κόμβοι που χρησιμοποιούν το CafNet πρέπει να έχουν ξεχωριστό ID. Το CafNet, σε αντίθεση με πρωτόκολλα συνδέσεων που χρησιμοποιούν ροή δεδομένων (π.χ. TCP), χρησιμοποιεί ένα API που δίνει στις εφαρμογές τη δυνατότητα να μεταφέρουν την πληροφορία με μηνύματα. Αυτή η προσέγγιση ταιριάζει περισσότερο σε περιπτώσεις όπου η σύνδεση (και συνεπώς η ροή των δεδομένων) ενδέχεται να διακόπτεται συχνά και για μεγάλα διαστήματα.

Το μήνυμα που χρησιμοποιεί το CafNet ονομάζεται «μονάδα» δεδομένων εφαρμογής (Application Data Unit – ADU). Όλη η ιδέα της υλοποίησης κρύβεται στον

τρόπο με τον οποίο γίνεται η προσωρινή αποθήκευση (buffering) και η αποστολή των μηνυμάτων. Δεν αποστέλλονται μηνύματα όταν απλώς υπάρχουν τέτοια προς αποστολή. Αυτό γίνεται γιατί σε περίπτωση αδυναμίας σύνδεσης τα μηνύματα θα αποθηκεύονταν προσωρινά στους buffer του επιπέδου μεταφοράς (transport layer), όπου δεν υπάρχουν πολλά περιθώρια χώρου αλλά ούτε και δυνατότητα ανακατάταξης των μηνυμάτων που βρίσκονται ήδη εκεί. Έτσι, ο δυναμικός καθορισμός προτεραιοτήτων δε θα ήταν δυνατός. Η στοίβα δικτύου (network stack), λοιπόν, δεν αποθηκεύει δεδομένα, αλλά ενημερώνει την εφαρμογή όταν η σύνδεση είναι εφικτή. Μέχρι τότε, όλοι οι buffers με τα δεδομένα συντηρούνταν από την εφαρμογή και την «τελευταία στιγμή», και ενώ υπάρχει σύνδεση, η εφαρμογή αποφασίζει ποια δεδομένα να στείλει.

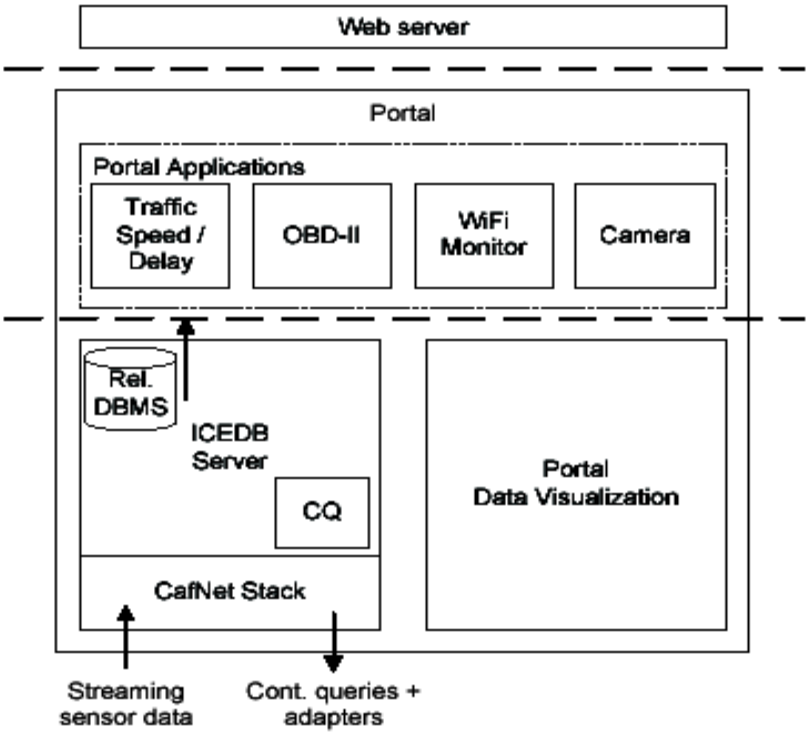
Το CafNet έχει τρία επίπεδα. Το επίπεδο μεταφοράς (Transport Layer) παρέχει στις εφαρμογές την ειδοποίηση συνδεσιμότητας που περιγράφηκε παραπάνω. Το API της αποτελείται μόνο από την «callback» συνάρτηση cb_get_adu(), που κάνει την εφαρμογή να επιστρέψει ένα μήνυμα (ADU) για άμεση αποστολή και την input(), η οποία φέρνει στο επίπεδο μεταφοράς μηνύματα από τα χαμηλότερα επίπεδα. Τα χαμηλότερα επίπεδα είναι το επίπεδο δικτύου (Network Layer) και το επίπεδο για την επικοινωνία με τους γειτονικούς κόμβους (Mule Adaptation Layer). Τα επίπεδα αυτά αποκρύπτουν τις λεπτομέρειες του επικοινωνιακού μέσου (Wi-Fi, Bluetooth κτλ.).



ΣΧΗΜΑ 2.1 – Η στοίβα επικοινωνίας του CafNet

Portal

Πρόκειται για ένα περιβάλλον πελάτη (client environment) που υποστηρίζει ό, τι χρειάζεται για να στηθεί η άκρη (edge) μιας εφαρμογής στο CarTel. Δηλαδή έχει εγκατεστημένη την εφαρμογή του πελάτη, το τμήμα της βάσης δεδομένων που χρειάζεται στον πελάτη (δηλαδή μια σχεσιακή βάση δεδομένων συνδεδεμένη με τον ICEDB Server), τη γραφικό περιβάλλον διεπαφής με το χρήστη κτλ. Όλα αυτά φαίνονται καλύτερα στο σχήμα που ακολουθεί.



ΣΧΗΜΑ 2.2 – Τα συστατικά του portal, όπου εγκαθίσταται το client τμήμα της εφαρμογής

Οι χρήστες παρακολουθούν τα δεδομένα των αισθητήρων χρησιμοποιώντας διαδικτυακές εφαρμογές που στεγάζονται στο portal και μπορεί να έχουν οποιαδήποτε μορφή ανάλογα με την εφαρμογή. Ένα παράδειγμα φαίνεται στις εικόνες της επόμενης παραγράφου.

Το ζητούμενο από ένα portal είναι να παρέχονται τα θεμέλια για την κατασκευή εφαρμογών που έχουν κοινό τρόπο ταυτοποίησης χρηστών (user authentication) και παρόμοιο γραφικό περιβάλλον. Αν υπάρχουν περιορισμοί πρόσβασης στα δεδομένα (privacy), οι χρήστες μπορούν να παρακολουθούν μόνο δεδομένα που υπάρχουν αποθηκευμένα στην βάση δεδομένων του συγκεκριμένου portal, αλλά είναι δυνατή και η υλοποίηση ανταλλαγής δεδομένων από portal σε portal. Όμως αυτό, που όπως θα δούμε αποτελεί και έναν από τους κύριους στόχους του λογισμικού που αναπτύχθηκε στα πλαίσια της εργασίας, είναι κάτι που μπορεί να υλοποιηθεί on top αλλά δεν προβλέπεται από τα πρωτόκολλα του CarTel.

Εφαρμογή – Συμπεράσματα

Η αλήθεια είναι ότι το CarTel, αν και αρκετά ολοκληρωμένο, είναι κατά κάποιο τρόπο ειδικού σκοπού. Όπως έχει ήδη γίνει σαφές και όπως μαρτυρά και το όνομά του, είναι σχεδιασμένο για εφαρμογές όπου οι κόμβοι αισθητήρων βρίσκονται σε οχήματα και κινούνται συνεχώς, με υψηλές μάλιστα ταχύτητες. Μπορεί να υποστηρίξει ευρεία γκάμα εφαρμογών αλλά για κάποιες άλλες εφαρμογές επιφέρει πολλή αχρείαστη πολυπλοκότητα. Επίσης, όπως έχει ήδη αναφερθεί, βασίζεται στην υπόθεση ότι η περιοχή είναι γεμάτη με σημεία ασύρματης πρόσβασης στο διαδίκτυο (Wi-Fi Access Points).

Στις εφαρμογές οδικής κυκλοφορίας (Road-Traffic Analysis Applications) χρησιμοποιήθηκε με αρκετούς τρόπους και με αρκετά ενδιαφέροντα γραφικά περιβάλλοντα, βασισμένα στα Google Maps. Παρακάτω φαίνονται δύο παραδείγματα ενώ ακολουθεί πίνακας με μετρήσεις που φανερώνουν πως λειτούργησαν στην πράξη κάποια ενδιαφέροντα ζητήματα που προσπαθεί να αντιμετωπίσει το CarTel και έχουν να κάνουν με τη διακοπτόμενη συνδεσιμότητα (intermittent connectivity).

ΕΙΚΟΝΑ 2.2 – Παραδείγματα από τα GUI εφαρμογών CarTel. Αριστερά: Φωτογραφία από την επόμενη στροφή. Δεξιά: Ενημέρωση του χρήστη για τα σημεία συμφόρησης στην πόλη.

Μέση διάρκεια σύνδεσης οχήματος	25 δευτερόλεπτα
Μέσο διάστημα μεταξύ συνδέσεων	260 δευτερόλεπτα
Μέση ταχύτητα μεταφοράς (upload) δεδομένων	30 Kbytes / δευτερόλεπτο

ΠΙΝΑΚΑΣ 2.1 – Σύνοψη δεδομένων συνδεσιμότητας εφαρμογών του CarTel

MetroSense

Η Γενική Ιδέα

Το MetroSense έχει στηριχτεί και αυτό στην διαδεδομένη πλέον υπόθεση ότι το θεμέλιο για τη δημιουργία μεγάλης κλίμακας και ευρείας χρήσης διαδικτυακών εφαρμογών αισθητήρων δε θα είναι τα μεγάλα και εξειδικευμένα δίκτυα αισθητήρων που υλοποιούν πολύπλοκους αλγόριθμους δρομολόγησης και λειτουργικότητας, για τα οποία έχει υπάρξει έντονη ερευνητική δραστηριότητα. Αντίθετα, για να μπορούν να προσαρτηθούν στο Internet (ως edges) πολλά ασύρματα δίκτυα αισθητήρων, απαιτείται μια νέα αρχιτεκτονική που να υποστηρίζει τις μεγάλης κλίμακας εφαρμογές και να εστιάζει στη χρήση άνθρωπο-κεντρικών αισθητήρων (people-centric sensing), αισθητήρων, δηλαδή, οι οποίοι είναι άμεσα συνδεδεμένοι με τον άνθρωπο-χρήστη τους, τις συνήθειές τους και την κίνησή τους. Κι αυτό γιατί οι άνθρωποι που φέρουν αισθητήρες, όχι μόνο είναι πιθανό ότι θα παίζουν τόσο το ρόλο του παραγωγού δεδομένων όσο και του καταναλωτή αλλά επίσης, μέσω της κίνησής τους, καθιστούν δυνατή την κάλυψη εκτάσεων και σημείων που με στατικούς αισθητήρες δε θα ήταν ποτέ δυνατόν να καλυφθούν. Επίσης, καταργείται το πολύ περιοριστικό ως τώρα πρόβλημα της ενέργειας των αισθητήρων αφού οι άνθρωποι φορείς τους μπορούν να τους φορτίζουν πολύ απλά και εύκολα, όπως γίνεται π.χ. με τα κινητά τηλέφωνα.

Η περιγραφή αυτή του people-centric sensing οδηγεί κατευθείαν σε κάποιες ιδέες για τη μορφή των δικτύων αισθητήρων. Μπορεί, λοιπόν, να πρόκειται είτε για κινητά τηλέφωνα εξοπλισμένα με αισθητήρες (βλ. και κεφ.3 - Skylark) είτε για άλλες μικροσυσκευές που θα φέρουν μικρο-αισθητήρες τύπου mote ή κάποια πιο εξελιγμένη μορφή έξυπνης σκόνης. Φυσικά, αφού το MetroSense αποσκοπεί σε διαδικτυακές εφαρμογές, πρέπει να θεωρηθεί δεδομένη η ύπαρξη και άλλων αξιοποιήσιμων δικτύων αισθητήρων, διαφορετικού μεγέθους, τεχνολογίας και λογικής. Συνεπώς, το MetroSense υποστηρίζει αυτήν την ετερογένεια με ένα κοινό σύνολο διεπαφών (interfaces) που πρέπει να υποστηρίζονται από κάθε πλατφόρμα ώστε αυτή να μπορεί να συμμετάσχει στις εφαρμογές.

Ιδιαιτερότητες – Καινοτομίες

Τα συστήματα που αποσκοπούν στο να εξυπηρετήσουν την ανάπτυξη εφαρμογών για μεγάλης κλίμακας και ανθρωπο-κεντρικά δίκτυα αισθητήρων είναι ούτως ή άλλως λίγα. Οι ιδιαιτερότητες του MetroSense έγκεινται στη γενικότητά του και στον προσανατολισμό του στο διαδίκτυο ενώ καινοτομικός είναι και ο ορισμός του opportunistic sensing και η προσέγγιση σε αυτό το ζήτημα.

Η «γενικότητα» του συστήματος φαίνεται από το γεγονός ότι οι περιπτώσεις χρήσης και οι εφαρμογές χωρίζονται σε τρία επίπεδα και επιδιώκεται η βέλτιστη υποστήριξη όλων. Έτσι, λοιπόν, έχουμε ατομικές εφαρμογές (personal sensing), ομαδικές εφαρμογές (peer sensing) και καθολικές εφαρμογές (utility sensing) και τα use-cases του συστήματος τις λαμβάνουν υπόψη όλες.

Ο προσανατολισμός στο διαδίκτυο, εκτός του ότι δηλώνεται ρητά και φαίνεται και στις δοκιμαστικές εφαρμογές, διακρίνεται και από τις τρεις αρχές στις οποίες βασίστηκε ολόκληρη η αρχιτεκτονική του συστήματος. 1) Συμβίωση δικτύων: Τα νέα

δίκτυα θα πρέπει να μην καταργούν αλλά να αναβαθμίζουν τα παλιά και η επιβάρυνση των υπαρχόντων δικτύων να είναι αμελητέα. 2) Μη συμμετρικός σχεδιασμός: Η ύπαρξη μηχανημάτων διαφορετικών δυνατοτήτων και τεχνολογίας πρέπει να λαμβάνεται υπόψη και, όταν είναι δυνατόν, να αξιοποιείται. Να μην αγνοείται, δηλαδή, το γεγονός ότι στο διαδίκτυο υπάρχουν και κόμβοι μεγάλης ισχύος, η οποία μπορεί να αξιοποιηθεί ούτε και ότι κάποιοι σύνδεσμοι (π.χ. οι ασύρματοι) είναι «αδύναμοι» και διακοπτόμενοι. 3) Τοπικότητα: Οι επικοινωνίες που θα γίνονται με βάση τα interfaces του MetroSense πρέπει να είναι τοπικές γιατί αυτό σε τελική ανάλυση, αν και μειώνει την ελαστικότητα του συστήματος, επιφέρει καλύτερη απόδοση, δεδομένης της τεχνολογίας του διαδικτύου.

Το opportunistic sensing έχει να κάνει με την τυχαιότητα της θέσης και της κατάστασης των αισθητήρων ανά πάσα στιγμή και θα αναλυθεί στην περιγραφή του συστήματος.

Το Σύστημα

Τα επίπεδα που διακρίνονται στην αρχιτεκτονική του MetroSense είναι τρία. Το επίπεδο εξυπηρετητών (Server tier), το επίπεδο σημείων πρόσβασης αισθητήρων (Sensor Access Point tier - SAP) και επίπεδο αισθητήρων (Sensor tier). Άρα πρόκειται για αρχιτεκτονική τριών επιπέδων (3-tier architecture).

Το επίπεδο εξυπηρετητών αποτελείται από μηχανήματα με πρακτικά άπειρη υπολογιστική και αποθηκευτική ισχύ, που συνδέονται μεταξύ τους μέσω Ethernet, έχουν δηλαδή καλή και σταθερή σύνδεση. Αυτοί οι εξυπηρετητές προσφέρουν στους αλγόριθμους του MetroSense δύο είδη υπηρεσιών. Τις υπηρεσίες πυρήνα (core components) και τις κοινές υπηρεσίες (common components). Οι υπηρεσίες πυρήνα είναι αυτές που επιτρέπουν τη διαχείριση και την επίβλεψη της συλλογής και προώθησης δεδομένων αισθητήρων. Παραδείγματα είναι ο έλεγχος προσβάσεων, η αποθήκευση της κατάστασης του συστήματος, ο συγχρονισμός και η επιβολή της τοπικότητας στη μετακίνηση των δεδομένων κτλ. Οι κοινές υπηρεσίες είναι κοινά εργαλεία που χρησιμοποιούνται για την αποθήκευση των ίδιων των δεδομένων και τους υπολογισμούς πάνω σε αυτά. Παραδείγματα είναι η αποθήκευση (repository), η «εξόρυξη» (mining) και ο έλεγχος ορθότητας (anomaly detection) των δεδομένων των αισθητήρων.

Το επίπεδο SAP επιτελεί τη διαδικασία για τη λήψη των δεδομένων των αισθητήρων, λειτουργεί σαν σταθμός-βάση για τα δεδομένα που συγκεντρώνονται από το επίπεδο αισθητήρων και προγραμματίζει τους αισθητήρες φορτώνοντας μικρά κομμάτια λογισμικού σε αυτούς. Μέσω των μελών αυτού του επιπέδου προσφέρεται υψηλή επίδοση, αξιοπιστία και ασφάλεια, τίποτα από τα οποία δεν είναι εφικτό αν απλοί αισθητήρες συλλέγουν και διακινούν δεδομένα αυτόνομα. Τηρώντας την αρχή της συμβίωσης δικτύων, μέλη του SAP πρέπει να μπορούν να είναι συστατικά που ήδη υπάρχουν στο διαδίκτυο, όπως PCs/Laptops, κινητά τηλέφωνα, σημεία πρόσβασης WiFi κτλ.

Το επίπεδο αισθητήρων αποτελείται από τους γνωστούς αισθητήρες, αρκεί αυτοί να υλοποιούν τα πρωτόκολλα που ορίζει το MetroSense για να επικοινωνούν με τα μέλη του επιπέδου SAP. Υποστηρίζεται φυσικά ασύρματη επικοινωνία και οι αισθητήρες μπορεί να είναι είτε στατικοί είτε μετακινούμενοι.

Το χαρακτηριστικό των αλγορίθμων και των πρωτοκόλλων του MetroSense είναι αυτό το μοντέλο της τυχαιότητας (opportunistic model) που χρησιμοποιεί ως προσέγγιση όσον αφορά το sensing. Ακολουθεί περιγραφή των χαρακτηριστικών του.

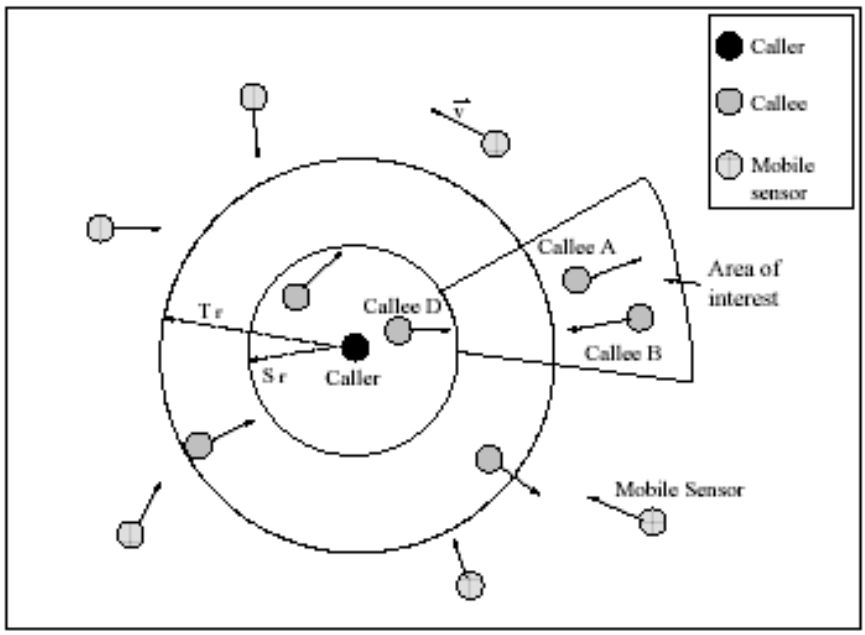
Χαρακτηριστικά του opportunistic model

Ο σκοπός είναι η ανεξέλεγκτη κίνηση των κινητών αισθητήρων, αντί να αποτελεί πρόβλημα, να αξιοποιηθεί ώστε να καλυφθούν τα αναπόφευκτα κενά των στατικών δικτύων αισθητήρων. Ο ορισμός του opportunistic tasking είναι η πληρότητα των τριών προϋποθέσεων που πρέπει να καλύπτονται για να γίνει επιτυχής μέτρηση σε ένα περιβάλλον σαν το MetroSense. Για να γίνει, λοιπόν, πρέπει ο αισθητήρας να έχει τα κατάλληλα όργανα για να μετρήσει τα ζητούμενα δεδομένα, να έχει φορτωθεί σε αυτόν η σχετική εφαρμογή και να περνά από το σημείο ενδιαφέροντος την κατάλληλη χρονική στιγμή. Με δεδομένη, όμως, την αρχιτεκτονική του MetroSense γίνεται σαφές ότι για να ολοκληρωθεί η οποιαδήποτε μέτρηση πρέπει τα δεδομένα να περάσουν και σε κάποιο μέλος του SAP. Άρα ο αισθητήρας πρέπει να βρίσκεται στην εμβέλεια ενός τέτοιου τόσο όταν λαμβάνει την εντολή να συλλέξει τα δεδομένα όσο και όταν πρέπει να στείλει πίσω τις τιμές. Έχοντας αυτές τις ανάγκες κατά νου μπορούμε να προχωρήσουμε στην ανάλυση πιο συγκεκριμένων χαρακτηριστικών.

- Τυχαία μεταβίβαση ευθύνης (Opportunistic Delegation Model)

Πρόκειται για περιορισμένη μεταβίβαση της ευθύνης λήψης και συλλογής μετρήσεων από αισθητήρα σε αισθητήρα, η οποία γίνεται μόνο όταν μπορεί να υπάρξει κάποιο όφελος και είναι περιορισμένη με την έννοια ότι έχει χρονικό όριο και αφορά συγκεκριμένο στόχο. Η λειτουργία αυτού του χαρακτηριστικού και το κέρδος από αυτήν γίνονται κατανοητά μέσα από ένα σενάριο:
Υποθέτουμε ότι μια εφαρμογή χρειάζεται δεδομένα τύπου «γ» από την περιοχή Α στο χρονικό διάστημα [t1, t2]. Υποθέτουμε επίσης ότι, μέσω κάποιας προηγούμενης ανάθεσης από κάποιο SAP, ο στατικός αισθητήρας y έχει αναλάβει την ευθύνη για αυτή τη μέτρηση, αλλά η εμβέλεια του αισθητήρα τύπου «γ» που διαθέτει είναι τέτοια που η περιοχή Α είναι εκτός εμβέλειας. Υπάρχει, όμως, ένας κινητός αισθητήρας z που διαθέτει ένα αισθητήρα τύπου «γ» και που έχει διάνυσμα κίνησης τέτοιο που διασχίζει την περιοχή Α κατά το χρονικό διάστημα [t1, t2]. Ιδανικά, ο z «συναντιέται» με τον y πριν την τομή του διανύσματος κίνησής του με την περιοχή Α. Σε αυτήν την περίπτωση, ο y μεταβιβάζει την ευθύνη για τη μέτρηση στον z. Επίσης, αν ο z δε συναντά τον y, αυτό μπορεί να γίνει σε περισσότερα βήματα. Δηλαδή ο y να μεταβιβάσει την ευθύνη σε κάποιον w και αυτός στον z. Αφότου αναληφθεί η ευθύνη, τα δεδομένα που θα μετρηθούν πρέπει να επιστραφούν στον αισθητήρα που μεταβίβασε την ευθύνη. Και αυτό μπορεί, αν χρειαστεί, να γίνει με μεταβίβαση των δεδομένων. Στο παράδειγμα, δηλαδή, αν ο z δε σκοπεύει να επιστρέψει στην περιοχή εμβέλειας του y μπορεί να μεταβιβάσει τις μετρήσεις σε κάποιον αισθητήρα με διάνυσμα κίνησης προς τον y.

Το παρακάτω σχήμα δείχνει ένα παράδειγμα αύξησης της εικονικής εμβέλειας ενός στατικού αισθητήρα (Caller) χάρη στους κινητούς αισθητήρες (Callees).



ΣΧΗΜΑ 2.3 – Αύξηση εμβέλειας αισθητήρα χάρη σε opportunistic delegation

Σε αυτό το σημείο πρέπει να τονιστεί ότι η «κάλυψη» που παρέχουν οι αισθητήρες με αυτόν τον τρόπο είναι ίσως αυξημένη, αλλά εξακολουθεί να είναι πιθανοτική. Δεν υπάρχουν, δηλαδή, εγγυήσεις ότι θα πληρούνται οι προϋποθέσεις και ως εκ τούτου πρέπει να λαμβάνεται υπόψη από τις εφαρμογές η πιθανότητα αποτυχίας ή μεγάλης καθυστέρησης.

- Επιλογή αισθητήρων

Τόσο κατά την ανάθεση της ευθύνης για μια μέτρηση σε έναν αισθητήρα όσο και κατά τη μεταβίβαση αυτής της ευθύνης με τον τρόπο που αναλύθηκε, μπορεί να υπάρχουν πολλοί υποψήφιοι αισθητήρες που να πληρούν τις προϋποθέσεις. Στη γενική περίπτωση αυτός που πρέπει να επιλέξει, δε γνωρίζει όλα τα χαρακτηριστικά των υποψηφίων αισθητήρων και μπορεί να μην καταλήξει και στην καλύτερη επιλογή.

Μια λύση που προβλέπει το MetroSense είναι η επιλογή πολλών εκ των υποψηφίων αισθητήρων, αυξάνοντας έτσι και την πιθανότητα επιτυχούς ολοκλήρωσης της εργασίας. Απλά πρέπει να προσεχθεί το γεγονός ότι μια τέτοια τακτική, σε συνδυασμό με μια πιθανή αλυσιδωτή μεταβίβαση ευθύνης πολλών βημάτων, μπορεί να οδηγήσει σε εκθετικά μεγάλη επιβάρυνση χωρίς λόγο. Γι' αυτό το MetroSense μπορεί να διατηρεί πληροφορίες για να ελέγχει την αύξηση του αριθμού των αισθητήρων στους οποίους έχει ανατεθεί μια εργασία, διατηρώντας την σταθερή ή γραμμική.

Μια άλλη προσέγγιση η οποία μπορεί να χρησιμοποιηθεί από μια εφαρμογή φτιαγμένη πάνω στο MetroSense, είναι η χρήση του δείκτη ποιότητας συνδέσμων (Link Quality Indicator - LQI). Με τη βοήθεια περιοδικών τετριμμένων μηνυμάτων υπολογίζεται η ικανότητα / ισχύς του σχετικού αισθητήρα. Ανάλογα, με βάση και άλλες πληροφορίες, γίνεται η επιλογή του καταλληλότερου.

- Αξιοπιστία των δεδομένων

Η επίδραση της πιθανοτικής αυτής προσέγγισης του opportunistic delegation στην αξιοπιστία των δεδομένων εξαρτάται από την ευαισθησία των οργάνων των αισθητήρων. Όταν τίθενται θέματα ευαισθησίας του οργάνου, τότε η μέτρηση είναι τόσο πιο ακριβής όσο πιο κοντά είναι ο αισθητήρας στο στόχο και το σφάλμα αυξάνεται μονοτονικά με την απόσταση. Λογικά δε θα ανατίθετο σε κάποιον αισθητήρα να κάνει οποιαδήποτε μέτρηση έχει ως στόχο ένα σημείο που ναι μεν είναι στην εμβέλειά του αλλά πέρα από το όριο της απόστασης μέσα στην οποία η μέτρηση είναι αξιόπιστη.

Όταν όμως γίνεται μεταβίβαση ευθύνης, η απόσταση που θα έχει τελικά ο αισθητήρας από το σημείο ενδιαφέροντος τη στιγμή της μέτρησης δε μπορεί να είναι γνωστή με ακρίβεια. Η λύση που προτείνει το MetroSense είναι απλά η ύπαρξη στατικών αισθητήρων που καλύπτουν πλήρως περιοχές ιδιαίτερου ενδιαφέροντος, κάτι που όμως δεν είναι εφικτό σε μεγάλη κλίμακα. Η πρώτη σκέψη που έρχεται σε κάποιον ίσως είναι ο περιορισμός των αισθητήρων ώστε να μη μετρούν ποτέ σημεία σε απόσταση που δεν υπάρχει απόλυτη αξιοπιστία. Όταν όμως τα όργανα είναι ευαίσθητα, μάλλον δε μπορεί να οριστεί απόσταση απόλυτης ασφάλειας και ο υπολογισμός της είναι κάτι μάλλον υποκειμενικό (εξαρτάται από την εφαρμογή κτλ.). Άλλωστε τέτοιος περιορισμός μπορεί να μείωνε δραματικά την κάλυψη του συστήματος.

Εφαρμογή – Συμπεράσματα

Η μία από τις πειραματικές εφαρμογές του MetroSense είναι ενδεικτική όσον αφορά το πλεονέκτημα που δίνει το opportunistic μοντέλο. Ένα άλλο συμπέρασμα που προκύπτει από αυτήν την εφαρμογή είναι η αναγκαιότητα για μεγάλη πυκνότητα και μεγάλο πλήθος αισθητήρων που υπάρχει για να λειτουργήσει αποτελεσματικά μια εφαρμογή βασισμένη στο MetroSense.

Πρόκειται για μια εφαρμογή χιονοδρομικού κέντρου. Στατικοί αισθητήρες τοποθετούνται σε σταθερά σημεία της ευρύτερης περιοχής (π.χ. στήλες φώτων, σημαίες διαδρομών κτλ.). Επίσης αισθητήρες είναι προσαρμοσμένοι στους σκιέρ και το προσωπικό. Τα παραπάνω αποτελούν το επίπεδο αισθητήρων, όπως περιγράφηκε νωρίτερα, ενώ το επίπεδο SAP αποτελείται από σταθμούς τοποθετημένους στη βάση του βουνού και το επίπεδο εξυπηρετητών είναι στα γραφεία του χιονοδρομικού κέντρου. Συλλογή δεδομένων γίνεται από όλους τους αισθητήρες, μεταβίβαση ευθύνης από τους στατικούς στους κινητούς (αλλά και μεταξύ των κινητών) ώστε δεδομένα από όλο το βουνό να φτάνουν συνεχώς στη βάση. Τα οφέλη είναι προφανή. Οι σκιέρ μπορούν να επιλέγουν διαδρομές ανάλογα με τη διαθεσιμότητα, οι συντηρητές μπορούν να γνωρίζουν την κατάσταση κάθε σημείο, το ιατρικό προσωπικό μπορεί να σπεύδει κατευθείαν σε κάθε έκτακτη ανάγκη κτλ.

ΕΙΚΟΝΑ 2.3 – Κόμβοι εφαρμογής σκι βασισμένης στο MetroSense. Οι κόκκινες τελείες είναι οι στατικοί αισθητήρες, ενώ τα μαύρα τετράγωνα αποτελούν το επίπεδο SAP.

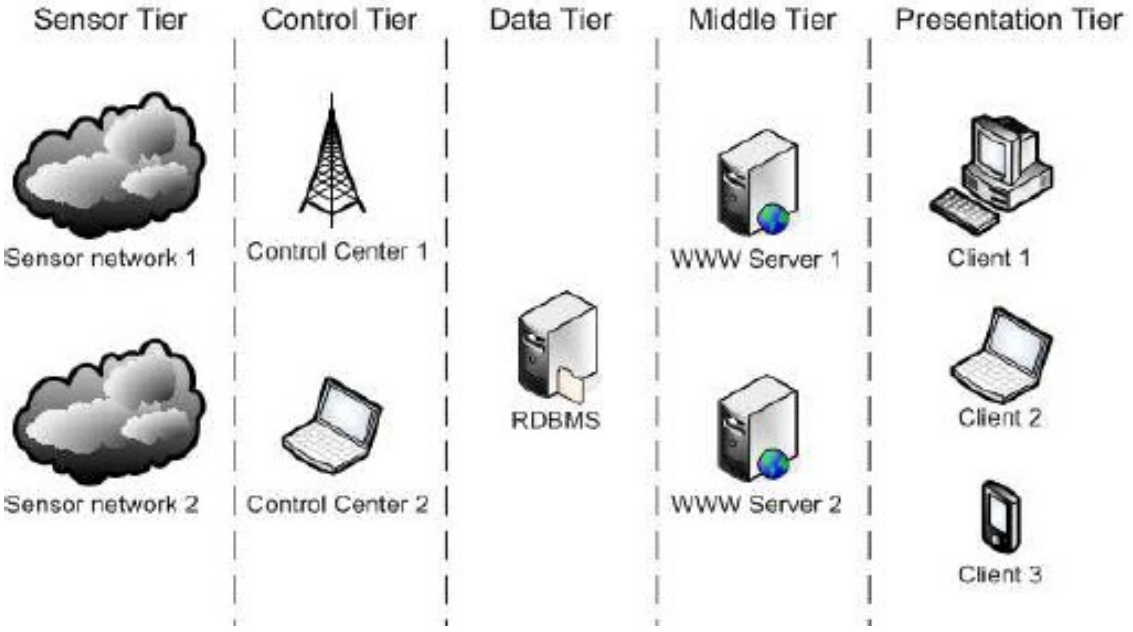
Γίνεται σαφές ότι και το MetroSense αντιμετωπίζει αποτελεσματικά την κίνηση και άρα μπορεί να χρησιμοποιηθεί και αυτό για εφαρμογές για οχήματα, όπως και το CarTel. Χαρακτηριστικό είναι ότι άλλη πειραματική εφαρμογή έγινε με αισθητήρες προσαρμοσμένους σε ποδήλατα. Βέβαια, μια και η διάδοση των δεδομένων βασίζεται και στην τύχη, οι εφαρμογές είναι αξιόπιστες μόνο αν το σύστημα είναι πυκνό οπότε και η πιθανότητα εύρεσης κόμβων κατάλληλων για τη μεταβίβαση ευθύνης είναι μεγάλη. Το MetroSense έχει έλλειψη στην αντιμετώπιση της ετερογένειας (δεν περιγράφεται μέθοδος για το configuration των -πιθανώς ετερογενών- δεδομένων των αισθητήρων) ενώ δε γίνονται και αναφορές στον τρόπο διασύνδεσης πολλών WSN (κάτι που θα υλοποιούταν στο επίπεδο εξυπηρετητών). Τα δύο παραπάνω ζητήματα αντιμετωπίζονται εν μέρει από κάποια άλλα overlays ενώ αποτελούν κύριο στόχο της εργασίας που περιγράφεται στο κεφάλαιο 4.

jWebDust

Η Γενική Ιδέα

Το JWebDust είναι χαρακτηριστικό παράδειγμα της αρχιτεκτονικής μιας και τα επίπεδά του φαίνονται σαφώς διαχωρισμένα μεταξύ τους. Σε σχέση, λοιπόν, με τα δύο συστήματα που παρουσιάστηκαν προηγουμένως είναι πιο τμηματοποιημένο (π.χ. στο CarTel όλα τα επίπεδα υλοποιούσαν τμήματα του συνολικού πρωτοκόλλου ICEDB κτλ.), πιο γενικό όσον αφορά το είδος των WSN που επιχειρεί να υποστηρίξει (μεγάλα/μικρά, κινητά/στατικά κτλ.), ενώ είναι και λιγότερο προσανατολισμένο στην αντιμετώπιση της κίνησης και της διακοπτόμενης συνδεσιμότητας, αν και η τμηματικότητά του επιτρέπει και το χειρισμό τέτοιων περιπτώσεων. Στόχος του δεν είναι τόσο η υλοποίηση νέων πρωτοκόλλων όσο η δημιουργία ενός ολοκληρωμένου overlay που θα παραλείπει να αντιμετωπίσει όσο το δυνατόν λιγότερα ζητήματα των WSN και θα αποκρύπτει όσο το δυνατόν περισσότερες λεπτομέρειες από τον προγραμματιστή εφαρμογών για WSN. Επίσης, το jWebDust δίνει βάρος και στη δυνατότητα υλοποίησης διαφόρων περιβαλλόντων χρήστη, μια και η αρχιτεκτονική του βολεύει ώστε αυτό το κομμάτι να σχεδιαστεί ανεξάρτητα. Εκτός αυτού, η java, που χρησιμοποιείται στα υψηλά επίπεδα, διευκολύνει τη σύνδεση του συστήματος με διάφορες εφαρμογές χρήστη που χρησιμοποιούν ποικίλα γραφικά περιβάλλοντα.

Για να γίνει καλύτερα κατανοητή η ιδέα του jWebDust παρατίθεται εδώ η εικόνα με τα επίπεδα της αρχιτεκτονικής του. Τα επίπεδα θα περιγραφούν, το καθένα ξεχωριστά, στην ανάλυση του συστήματος.



ΕΙΚΟΝΑ 2.4 – Τα 5 επίπεδα της αρχιτεκτονικής του jWebDust

Ιδιαιτερότητες – Καινοτομίες

Η συνεισφορά του jWebDust στον τομέα των overlays για WSN δεν είναι κάποιος νέος αλγόριθμος ή κάποια νέα πρωτόκολλα για την υλοποίηση βασικών λειτουργιών (όπως π.χ. οι αλγόριθμοι για opportunistic sensing του MetroSense ή τα πρωτόκολλα ICEDB και CafNet του CarTel). Αυτό που παρουσιάζει το jWebDust είναι μια νέα αρχιτεκτονική για τη δημιουργία τέτοιων συστημάτων και ο βέλτιστος τρόπος για να ενθυλακωθούν και να αποκρυφτούν οι λειτουργίες του κάθε επιπέδου. Σε πολλά σημεία της αρχιτεκτονικής, ο τελικός δημιουργός της εφαρμογής αφήνεται να επιλέξει τον τρόπο και την τεχνολογία υλοποίησης ανάμεσα από υπάρχουσες τεχνολογίες και πρωτόκολλα, γνωρίζοντας ότι όποια κι αν είναι η επιλογή του, το jWebDust θα την υποστηρίζει και θα του παρέχει το κατάλληλο API. Θα φανεί, άλλωστε, πώς το jWebDust θα μπορεί να χρησιμοποιηθεί σε συνδυασμό με το σύστημα που υλοποιήθηκε στα πλαίσια της εργασίας και παρουσιάζεται στο κεφάλαιο 4.

Οι βασικότερες από τις επιμέρους ιδιαιτερότητες, που βασίζονται στις αρχές της *γενικότητας* και της υποστήριξης της *ετερογένειας*, είναι οι ακόλουθες:

- Παρέχει ένα περιβάλλον για την υποστήριξη, το «πακετάρισμα» και το χειρισμό της πληθώρας των χαμηλότερου επιπέδου πρωτοκόλλων, χωρίς να απαιτείται πολύς κόπος κατά την υλοποίηση και τη διαχείριση του συστήματος.
- Επιτρέπει την υλοποίηση επιπλέον λειτουργικότητας που μπορεί εύκολα να ενσωματωθεί με την υπόλοιπη αρχιτεκτονική σε όλα τα επίπεδα της ιεραρχίας, ανάλογα με τις απαιτήσεις κάθε εφαρμογής.
- Υποστηρίζει τη διασύνδεση πολλών WSN (ανεξαρτήτως της γεωγραφικής τους θέσης), που το καθένα έχει ξεχωριστό κέντρο ελέγχου. Έτσι, χωρίς να υπάρχει η απαίτηση για διαφορετικά ID και λοιπά χαρακτηριστικά των αισθητήρων, το σύνολο των WSN φαίνεται στα μάτια του τελικού χρήστη σαν ένα, ενιαίο δίκτυο αισθητήρων.
- Μπορεί να χειριστεί δίκτυα αισθητήρων με ετερογενή χαρακτηριστικά, όπως άλλωστε αναμένεται να είναι τα δίκτυα, αν πρόκειται για ήδη υπάρχοντα, απομακρυσμένα δίκτυα του πραγματικού κόσμου.
- Μπορεί να υποστηρίξει κίνηση/διακοπές σύνδεσης, χειριζόμενο με ειδικό τρόπο τις αποσυνδέσεις των επιμέρους κέντρων ελέγχου (μαζί με τα αντίστοιχα WSN). Αυτό μπορεί να γίνει εύκολα σε ένα κεντρικοποιημένο σύστημα σαν το jWebDust, αφού η αποσύνδεση μερών δεν επηρεάζει την πρόσβαση στη βάση δεδομένων και στο υπόλοιπο δίκτυο, και έτσι δε χάνονται πληροφορίες, αφού το αποσυνδεδεμένο κέντρο ενημερώνει τη βάση δεδομένων με τα νέα δεδομένα όταν επανασυνδεθεί.
- Υποστηρίζει την πρόσβαση στα δεδομένα του δικτύου από πολλούς χρήστες ταυτόχρονα, μια και οι χρήστες μπορούν να στέλνουν queries στο jWebDust με τη χρήση διαφόρων διεπαφών ή και μέσω Web Services.

Το Σύστημα

Η συνοπτική περιγραφή του jWebDust, που ακολουθεί, γίνεται σε δέκα παραγράφους. Στις πέντε πρώτες περιγράφονται τα πέντε επίπεδα της αρχιτεκτονικής του συστήματος (επίπεδο αισθητήρων, επίπεδο ελέγχου, επίπεδο πληροφορίας, ενδιάμεσο επίπεδο, επίπεδο παρουσίασης) ενώ στις επόμενες πέντε αναλύεται η αντιμετώπιση κάποιων βασικών ζητημάτων από το jWebDust (πρωτόκολλο επικοινωνίας δικτύου αισθητήρων, μέτρηση και διάδοση δεδομένων, ανακάλυψη νέων κόμβων, έλεγχος κατάστασης κόμβων, χρονικός συγχρονισμός). Πιο ολοκληρωμένη περιγραφή γίνεται στο [17] ενώ ακόμα περισσότερες λεπτομέρειες μπορούν να μελετηθούν στο Μέρος ΙΙΙ του [18].

Το επίπεδο αισθητήρων (Sensor tier)

Αποτελείται από το σύνολο των κόμβων αισθητήρων (motes), που είναι κατάλληλα τοποθετημένοι ώστε να παίρνουν μετρήσεις από μια συγκεκριμένη περιοχή, και όλοι οι κόμβοι μιας περιοχής αναφέρουν τα αποτελέσματα σε έναν σταθμό-βάση (gateway). Το πρωτόκολλο που χρησιμοποιείται από τους κόμβους σε ένα σύστημα jWebDust είναι βασισμένο στο λειτουργικό σύστημα TinyOS (που περιγράφηκε συνοπτικά στην εισαγωγή, ενότητα 1.2) και αποτελείται από πέντε επίπεδα, σε αντιστοιχία με το πασίγνωστο μοντέλο δικτύων OSI. Η λειτουργικότητα του φυσικού επιπέδου (Physical Layer)και του επιπέδου σύνδεσης δεδομένων (Data Link Layer) καλύπτονται πλήρως από το TinyOS, ενώ το firmware του jWebDust, που εκτελείται στα motes, δίνει στα επίπεδα δικτύου (Network Layer), μεταφοράς (Transport Layer) και εφαρμογής (Application Layer) μια λειτουργικότητα που θα περιγραφεί καλύτερα στις παραγράφους «πρωτόκολλο επικοινωνίας δικτύου αισθητήρων» και «μέτρηση και διάδοση δεδομένων».

Το επίπεδο ελέγχου (Control tier)

Αποτελείται από τα κέντρα ελέγχου όλων των επιμέρους WSN. Υπάρχουν δύο είδη κέντρων ελέγχου. Τα γενικά κέντρα ελέγχου και τα μικρά κέντρα ελέγχου.

Τα γενικά κέντρα ελέγχου είναι υπεύθυνα για τη συλλογή όλων των μετρήσεων από το δίκτυο αισθητήρων και για την προώθηση των «ερωτήσεων» (queries) από το επίπεδο δεδομένων προς το δίκτυο αισθητήρων. Αυτό υλοποιείται με περιοδικό έλεγχο του επιπέδου δεδομένων για την ύπαρξη νέων queries. Έτσι αποφεύγεται περιττή επικοινωνία με το δίκτυο αισθητήρων, κάτι που θα σπαταλούσε πολύτιμη ενέργεια. Οι μετρήσεις που παραλαμβάνουν τα κέντρα ελέγχου καταγράφονται απευθείας στο επίπεδο δεδομένων, σε μια βάση δεδομένων που θα μελετηθεί στην επόμενη παράγραφο. Ένα σύνηθες γενικό κέντρο ελέγχου αποτελείται από ένα PC ή ένα Laptop με ένα mote προσκολλημένο σε αυτό. Το mote είναι απαραίτητο για την επικοινωνία με τους υπόλοιπους κόμβους. Φυσικά, το κέντρο ελέγχου μπορεί να είναι και ένα PDA με την απαραίτητη λειτουργικότητα (και πάλι η ύπαρξη ενός προσκολλημένου mote είναι, φυσικά, απαραίτητη).

Τα μικρά κέντρα ελέγχου χρησιμοποιούνται για τον έλεγχο της κατάστασης των κόμβων εντός ενός δικτύου αισθητήρων. Ελέγχουν δηλαδή τα επίπεδα διαθέσιμης

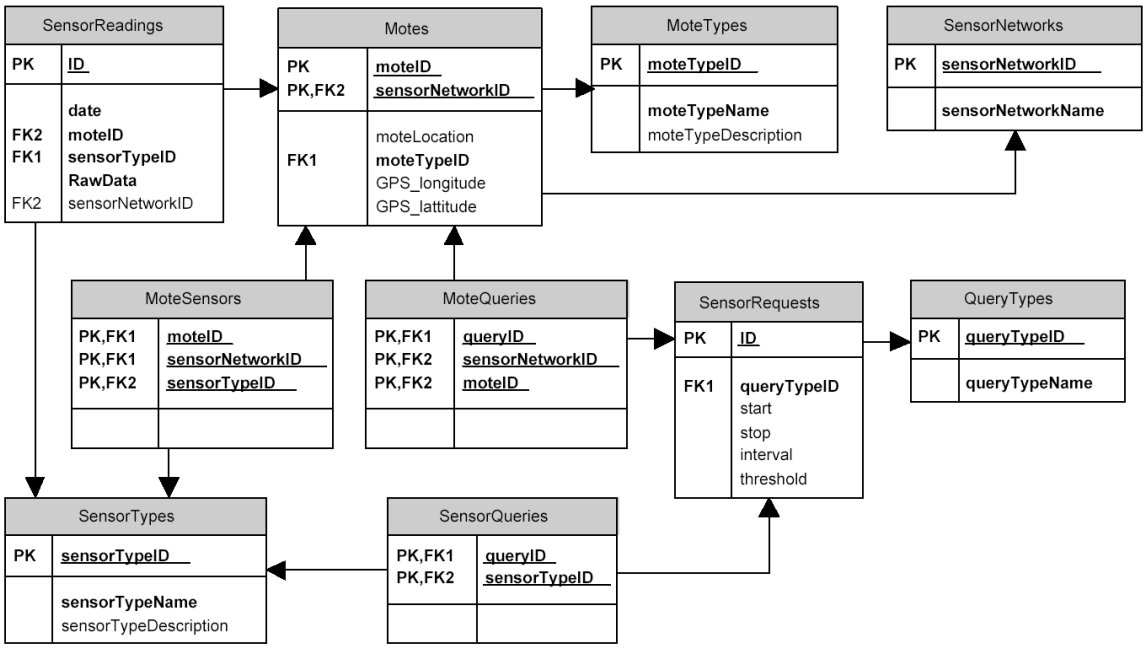
ενέργειας και τη δυνατότητα σύνδεσης των κόμβων με το υπόλοιπο δίκτυο. Μπορούν να χρησιμοποιηθούν και για άλλους παρόμοιους ελέγχους, αλλά δεν προωθούν δεδομένα στο δίκτυο και δεν επικοινωνούν με τα υψηλότερα επίπεδα της ιεραρχίας.

Στα κέντρα ελέγχου βρίσκεται και η λειτουργικότητα του συστήματος που χειρίζεται τις διακοπές συνδέσεων (intermittent connectivity). Η λειτουργία των κέντρων ελέγχου και η επικοινωνία τους με το δίκτυο αισθητήρων συνεχίζεται και όταν τα κέντρα ελέγχου δεν έχουν επικοινωνία με το υπόλοιπο δίκτυο. Αυτό είναι σημαντικό γιατί τα κέντρα ελέγχου μπορεί να έχουν και αυτά ασύρματη επικοινωνία με το υπόλοιπο δίκτυο. Με την αρχιτεκτονική του jWebDust δε χάνονται ούτε ερωτήσεις (τις παραλαμβάνει το κέντρο ελέγχου όταν αποκατασταθεί η σύνδεση) ούτε αποστολές δεδομένων (παραμένουν στο κέντρο ελέγχου και καταγράφονται στο επίπεδο δεδομένων όταν αποκατασταθεί η σύνδεση με αυτό).

Επίσης, το γεγονός ότι τα κέντρα ελέγχου παραλαμβάνουν «με δική τους πρωτοβουλία» (περιοδικά) τα queries και το ότι το κάθε κέντρο ελέγχου έχει ένα διαφορετικό ID, αποκρύπτει στα ανώτερα επίπεδα την ύπαρξη πολλών επιμέρους WSN. Τα ανώτερα επίπεδα απλά δημιουργούν queries και το κατάλληλο κέντρο ελέγχου μπορεί να απαντήσει.

Το επίπεδο δεδομένων (Data tier)

Το επίπεδο δεδομένων βασίζεται σε σχεσιακές βάσεις δεδομένων που χρησιμοποιούν ANSI SQL και η λειτουργικότητα που προσθέτει το jWebDust έχει να κάνει με τις υπηρεσίες που χρειάζονται στο ενδιάμεσο επίπεδο και στο επίπεδο ελέγχου. Ανεξαρτήτως εφαρμογής, δημιουργείται ένα συγκεκριμένο σχήμα βάσης δεδομένων, που είναι απαραίτητο για τη λειτουργικότητα που προσφέρει το jWebDust. Κάθε εφαρμογή μπορεί να έχει, προφανώς, και άλλα στοιχεία στη βάση ή να χρησιμοποιεί και άλλες βάσεις δεδομένων, αλλά το παρακάτω σχήμα είναι απαραίτητο:



ΣΧΗΜΑ 2.4 – Το σχήμα της βάσης δεδομένων του jWebDust

Περιέχει 10 πίνακες, που χωρίζονται σε τρεις κατηγορίες. Κάποιοι σχετίζονται με τα τεχνικά χαρακτηριστικά των motes, άλλοι με τα queries και άλλοι με τα αποτελέσματα των μετρήσεων των αισθητήρων.

Οι πίνακες των κόμβων (mote related tables) περιέχουν πληροφορίες για το υλικό των κόμβων, οι οποίες είναι απαραίτητες για να μπορούν να υποστηριχθούν ετερογενή δίκτυα. Ελέγχοντας πρώτα τους συγκεκριμένους πίνακες, το σύστημα θα μπορεί να αποφασίσει τι πρωτόκολλα επικοινωνίας κτλ. να χρησιμοποιήσει ανάλογα με την τεχνολογία του εκάστοτε χρησιμοποιούμενου κόμβου. Φυσικά, μια ελάχιστη απαίτηση είναι η εξής: οι κόμβοι του ίδιου δικτύου να υλοποιούν το ίδιο πρωτόκολλο επικοινωνίας, ώστε να μπορούν να επικοινωνούν μεταξύ τους. Οι πίνακες MoteTypes και SensorTypes περιέχουν τα είδη επεξεργαστών / ραδιοσυχνοτήτων που αναμένεται να χρησιμοποιηθούν στους κόμβους του δικτύου και τους αισθητήρες που μπορούν να υπάρχουν σε αυτά. Ο πίνακας Motes περιέχει μία εγγραφή για κάθε κόμβο του δικτύου και έχει σχέση ένα-προς-ένα με τον MoteTypes (κάθε κόμβος ανήκει σε έναν και μόνο τύπο) και σχέση n-προς-n με τον SensorTypes (κάθε κόμβος μπορεί να έχει πολλά είδη αισθητήρων). Ο πίνακας που υλοποιεί αυτήν την τελευταία σχέση και δείχνει τι αισθητήρες έχει ο κάθε κόμβος είναι ο πίνακας MoteSensors. Τέλος, ο πίνακας SensorNetworks περιέχει μία εγγραφή για κάθε επιμέρους WSN που υπάρχει σε ολόκληρο το σύστημα που έχει στηθεί με το jWebDust.

Οι πίνακες των ερωτήσεων (query related tables) περιέχουν πληροφορίες σχετικά με τα queries, τα οποία μπορεί να είναι και σύνθετα, απευθυνόμενα σε πολλαπλούς κόμβους / αισθητήρες. Εκτός από τα κλασσικά queries της SQL, το jWebDust ορίζει και νέους τύπους queries που μπορούν να απευθύνονται στα δίκτυα αισθητήρων. Ο πίνακας QueryTypes περιέχει τις δυνατές επιλογές. Ο πίνακας SensorRequests περιέχει όλα τα queries («σπασμένα» σε κατάλληλα πεδία) που έχουν τεθεί και δεν έχουν εξυπηρετηθεί. Οι πίνακες SensorQueries και MoteQueries χρησιμοποιούνται για να υλοποιήσουν τη σχέση n-προς-n των queries με τους πίνακες Mote και SensorTypes.

Ο πίνακας μετρήσεων αισθητήρων (sensor readings table) είναι ο SensorReadings και περιέχει τις πληροφορίες που έχουν παραληφθεί από τα δίκτυα αισθητήρων. Κάθε εγγραφή είναι μια μέτρηση που αφορά έναν κόμβο και έναν συγκεκριμένο αισθητήρα. Ως εκ τούτου, ο πίνακας αυτός σχετίζεται με τους Motes, SensorTypes και SensorRequests.

Το ενδιάμεσο επίπεδο (Middle tier)

Επειδή η θέση του ενδιάμεσου επιπέδου στο σχήμα της ιεραρχίας μπορεί να προκαλεί κάποια παρερμηνεία, τονίζεται ότι το ενδιάμεσο επίπεδο συνδέει το επίπεδο ελέγχου με τη βάση δεδομένων και το επίπεδο παρουσίασης με τη βάση δεδομένων. Δηλαδή, το επίπεδο ελέγχου παραδίδει στο ενδιάμεσο επίπεδο τις μετρήσεις από το επίπεδο αισθητήρων χωρίς να νοιάζεται για το πώς αυτές θα καταγραφούν στη βάση δεδομένων. Παρομοίως, το επίπεδο παρουσίασης χρησιμοποιεί ένα απλό API για να προμηθευτεί δεδομένα από τη βάση, χωρίς να ενδιαφέρεται για τον τρόπο με τον οποίο αυτό υλοποιείται από το ενδιάμεσο επίπεδο.

Η εκτέλεση του λογισμικού του ενδιάμεσου επιπέδου πρέπει να είναι αυτόνομη από τη λειτουργία κάθε άλλου τμήματος της εφαρμογής. Για την επίτευξη αυτού του σκοπού χρησιμοποιείται η τεχνολογία των Java Servlets. Πρόκειται για ανεξάρτητες

συνιστώσες λογισμικού που η καθεμία έχει καθορισμένη λειτουργία. Προφανώς, η λειτουργία των servlets εξαρτάται αφενός από τη βάση δεδομένων, που έχει σταθερό σχήμα. Εξαρτάται, όμως, και από τη λειτουργικότητα που είναι επιθυμητή για κάθε επίπεδο που επικοινωνεί με το ενδιάμεσο επίπεδο. Η χρήση των ανεξάρτητων συνιστωσών λογισμικού (που τροποποιούνται και μεταγλωττίζονται και ανεξάρτητα με ευκολία και κατά βούληση) προσδίδει στο jWebDust ευελιξία σε αλλαγές της λειτουργικότητας βασικών τμημάτων. Επειδή, όπως προκύπτει άλλωστε και από τη συζήτηση για πιθανή συνεργασία του jWebDust με το λογισμικό που παρουσιάζεται στο κεφάλαιο 4, το ενδιάμεσο επίπεδο μπορεί να υλοποιηθεί με διάφορους τρόπους (χάρη και στα servlets), δε θα επεκταθούμε. Αναφέρεται, απλά, ότι στη συγκεκριμένη υλοποίηση με servlets, υπάρχουν τριών κατηγοριών κλάσεις σε Java. Αυτές που αντιστοιχούν στους πίνακες της βάσης δεδομένων, αυτές που τις διαχειρίζονται και αυτές που αναλαμβάνουν τη διασύνδεση με τη βάση.

Το επίπεδο παρουσίασης (Presentation tier)

Το επίπεδο παρουσίασης του συστήματος είναι άμεσα συνδεδεμένο με την εφαρμογή. Είναι κυρίως το γραφικό περιβάλλον μέσω του οποίου ο χρήστης μπορεί να δίνει την είσοδο με βάση την οποία θα επιλέγονται οι κατάλληλες μέθοδοι από τα χαμηλότερα επίπεδα και θα γίνονται οι απαραίτητες λειτουργίες ώστε να παρουσιαστούν τα αποτελέσματα με έναν πρακτικό και ωραίο τρόπο.

Εξαιτίας της φύσης των εφαρμογών δικτύων αισθητήρων, το επίπεδο παρουσίασης συνηθίζει να χρησιμοποιεί «γεωγραφικά» χαρακτηριστικά ή ακόμα και χάρτες. Επίσης, αν και γενικώς πρέπει να λαμβάνεται υπόψη ότι το επίπεδο παρουσίασης ενός WSN μπορεί να υλοποιείται και σε συσκευές περιορισμένων δυνατοτήτων (κινητά, κάποια PDA's), το jWebDust είναι περισσότερο προσανατολισμένο σε ικανούς PC Browsers. Φυσικά, η αρχιτεκτονική του επιτρέπει ανάλογες προσθήκες, με σχετικά εύκολο τρόπο.

Για να είναι πληρέστερη η περιγραφή του συστήματος, ακολουθεί, όπως αναφέρθηκε και νωρίτερα, η παρουσίαση πέντε σημαντικών σημείων της λειτουργικότητας του jWebDust. Πρόκειται για τα σημεία όπου αντιμετωπίζονται ζητήματα που είναι βασικά για τη λειτουργία κάθε δικτύου αισθητήρων.

Πρωτόκολλο επικοινωνίας δικτύου αισθητήρων

Όπως έχει αναφερθεί, το jWebDust έχει σχεδιαστεί για motes που εκτελούν το λειτουργικό σύστημα TinyOS. Ως εκ τούτου, οι αλγόριθμοι δρομολόγησης τους οποίους μπορεί να χρησιμοποιήσει το jWebDust είναι αυτοί που μπορεί να υλοποιηθούν από το συγκεκριμένο λειτουργικό. Τα πρωτόκολλα δρομολόγησης για το TinyOS χωρίζονται σε δύο κατηγορίες. Η πρώτη, που περιλαμβάνει και την πλειοψηφία των υλοποιούμενων αλγορίθμων, συμπεριλαμβάνει τους αλγόριθμους όπου τελικός προορισμός κάθε μηνύματος που δημιουργείται στα motes είναι το κέντρο ελέγχου. Φυσικά, υπάρχει επικοινωνία μεταξύ των motes για χάρη της υλοποίησης του multi-hop routing, αλλά ο τελικός προορισμός ενός μηνύματος δεν είναι ποτέ ένα mote. Σε αυτή την κατηγορία

ανήκει και τι κλασσικό «πρωτόκολλο δρομολόγησης πολλών βημάτων με δενδρική δομή» (multi-hop tree-based routing protocol). Στη δεύτερη κατηγορία ανήκουν τα υπόλοιπα πρωτόκολλα, αυτά δηλαδή που επιτρέπουν δρομολόγηση με σχεδόν οποιονδήποτε πιθανό τελικό προορισμό (many-to-many routing).

Η βέλτιστη επιλογή του αλγορίθμου δρομολόγησης εξαρτάται πάντα από την εφαρμογή και τα κριτήρια είναι η ταχύτητα, το πλήθος των μηνυμάτων και η ενέργεια. Συνεπώς, το πρωτόκολλο δρομολόγησης που χρησιμοποιεί ένα overlay πρέπει να είναι κατάλληλο για το είδος των εφαρμογών που αναμένεται να υλοποιηθούν με αυτό το overlay, σχετικά γενικό και, αν είναι δυνατόν, προσαρμόσιμο στις ανάγκες της εκάστοτε εφαρμογής. Προς αυτήν την κατεύθυνση, για το jWebDust έχει επιλεγεί (αλλά ακόμα αποτελεί απλώς μια προδιαγραφή, δεν έχει υλοποιηθεί πλήρως) ένα πρωτόκολλο επικοινωνίας βασισμένο στον απλό αλγόριθμο Αναζήτησης Πρώτα-Σε-Πλάτος (Breadth First Search - BFS) για τη δημιουργία ενός δέντρου. Για την αναλυτική περιγραφή τέτοιων δέντρων δρομολόγησης, ο αναγνώστης παραπέμπεται στη σχετική βιβλιογραφία των κατανεμημένων συστημάτων [21], [22]. Ο απλός αυτός αλγόριθμος ανήκει στην πρώτη από τις δύο κατηγορίες που αναφέρθηκαν προηγουμένως και έχει επεκταθεί ώστε να είναι πιο αποδοτικός όσον αφορά την ενέργεια. Αυτό είναι εφικτό χάρη στη δυνατότητα του TinyOS να προσαρμόζει την ισχύ με την οποία τα motes εκπέμπουν τα ραδιοκύματα. Η επέκταση του αλγορίθμου είναι η εξής:

Το κέντρο ελέγχου στέλνει περιοδικά ορισμένα τετριμμένα μηνύματα «hello» τα οποία διαχέονται σε όλο το δίκτυο χάρη στην ενδοεπικοινωνία των motes. Αν η συνδεσιμότητα των motes είναι επαρκής, τα μηνύματα αυτά θα φτάσουν σε όλα τα motes του WSN. Τα motes διατηρούν σε πίνακες τα γειτονικά τους motes, από τα οποία θα πρέπει να λάβουν ή στα οποία θα πρέπει να στείλουν μηνύματα «hello». Αν κάποιο από αυτά τα μηνύματα δεν παραληφθεί, τότε τα αντίστοιχα motes αποφασίζουν να αυξήσουν τη χρησιμοποιούμενη εμβέλεια εκπομπής ραδιοκυμάτων, ώστε να ξεπεραστεί το πρόβλημα συνδεσιμότητας που υπάρχει. Έτσι, χρησιμοποιείται πάντα όση ακριβώς ενέργεια απαιτείται.

Μέτρηση και διάδοση δεδομένων

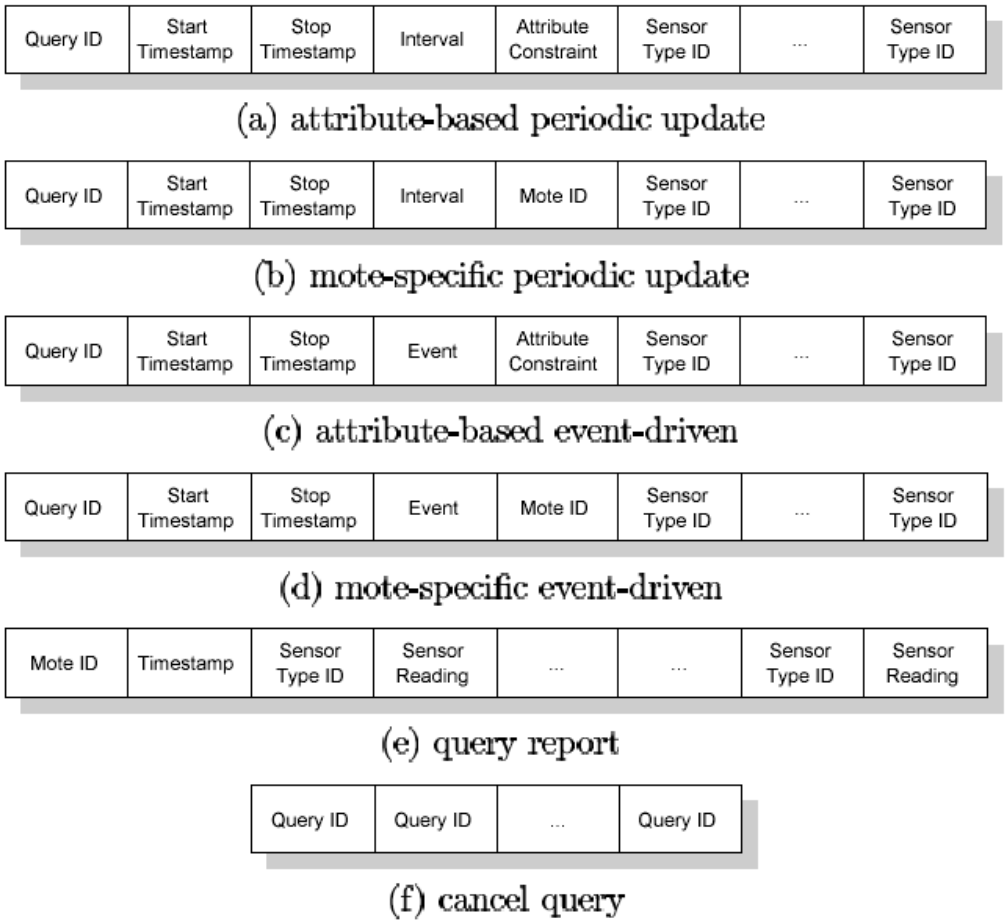
Τα queries που θα περιγραφούν είναι αυτά που χρησιμοποιούνται εντός ενός επιμέρους WSN για τη μέτρηση και τη διάδοσή των δεδομένων από τους αισθητήρες προς το κέντρο ελέγχου. Δεν πρέπει να συγχέονται με τα queries που κάνει μια εφαρμογή ή αυτά που τίθενται στη βάση δεδομένων του jWebDust. Αυτά που μόλις αναφέρθηκαν αναλύονται από το ενδιάμεσο επίπεδο και το επίπεδο ελέγχου για να δημιουργηθούν τα κατάλληλα queries που απευθύνονται στα motes. Τα τελευταία είναι αυτά που περιγράφονται εδώ.

Το jWebDust προσφέρει έξι τύπους τέτοιων μηνυμάτων/queries. Ακολουθεί η περιγραφή τους και στη συνέχεια ένα σχήμα με τα πεδία τους. Τονίζεται ότι το μέγεθος του πακέτου που χρησιμοποιεί το TinyOS είναι 36 bytes, επαρκές για την αποστολή των μηνυμάτων που θα περιγραφούν.

a) Μηνύματα για περιοδική ενημέρωση βασισμένα σε χαρακτηριστικό (Attribute-based periodic update). Τα μηνύματα αυτά απευθύνονται σε όλους τους κόμβους του δικτύου και όσοι κόμβοι πληρούν τους περιορισμούς που τίθενται στο

μήνυμα, απαντούν στο κέντρο ελέγχου. Παράδειγμα: «Να επιστρέφεται κάθε y λεπτά η θερμοκρασία όσων κόμβων έχουν τιμή φωτός μεγαλύτερη από x»

b) Μηνύματα για περιοδική ενημέρωση βασισμένα σε κόμβο (Mote-based periodic update). Τα μηνύματα αυτά απευθύνονται σε συγκεκριμένο κόμβο. Παράδειγμα: «Να επιστρέφεται κάθε 5 λεπτά η τιμή της υγρασίας του κόμβου με ID=74».
c) Μηνύματα οδηγημένα από γεγονότα βασισμένα σε χαρακτηριστικό (Attribute-based event-driven). Όπως στο a αλλά τώρα ζητείται μόνο μια τρέχουσα τιμή και όχι περιοδική αποστολή τιμών. Παράδειγμα: «Να επιστραφεί η θερμοκρασία όσων κόμβων βρίσκονται σε γεωγραφικό μήκος μεγαλύτερο του z».
d) Μηνύματα οδηγημένα από γεγονότα βασισμένα σε κόμβο (Mote-based event-driven). Όπως στο b, αλλά μόνο για την τρέχουσα τιμή. Παράδειγμα: «Να επιστραφεί η τιμή της θερμοκρασίας του κόμβου με ID=74».
e) Μηνύματα απάντησης (Query report). Δημιουργούνται από τους κόμβους με βάση τα queries που έχουν παραληφθεί και δρομολογούνται προς το κέντρο ελέγχου, περιέχοντας τις μετρημένες τιμές.
f) Μηνύματα ακύρωσης query (Cancel query).



ΣΧΗΜΑ 2.5 – Η δομή των μηνυμάτων που χρησιμοποιούνται για μέτρηση και διάδοση τιμών αισθητήρων

Ανακάλυψη νέων κόμβων

Ο στόχος του jWebDust είναι να παρέχεται στο χρήστη μια πλήρης εικόνα όσον αφορά τους υπάρχοντες κόμβους σε ένα WSN. Να είναι, δηλαδή, αποθηκευμένα ανά πάσα στιγμή όλα τα απαραίτητα χαρακτηριστικά (είδος και τεχνολογία, συχνότητες και εμβέλεια εκπομπής, διαθέσιμοι αισθητήρες κτλ.) κάθε συνδεδεμένου στο δίκτυο mote. Το jWebDust το επιτυγχάνει αυτό με τον κατάλληλο προγραμματισμό των motes, χωρίς να είναι απαραίτητος ο περιοδικός έλεγχος για νέους κόμβους. Όλα τα απαραίτητα χαρακτηριστικά του υλικού ενός κόμβου δίνονται ως παράμετροι κατά την εγκατάσταση του firmware στον κόμβο. Πιο συγκεκριμένα και με βάση αυτό το γεγονός, χρησιμοποιείται το παρακάτω πρωτόκολλο.

Όταν ένας κόμβος ενεργοποιείται για πρώτη φορά, αφότου ολοκληρωθεί η εκκίνηση και η αρχικοποίησή του, το πρωτόκολλο ανίχνευσης κόμβων στέλνει από τον κόμβο προς το κέντρο ελέγχου ένα μικρό μήνυμα που περιέχει το ID, τον τύπο και μια λίστα με όλους τους διαθέσιμους αισθητήρες του κόμβου. Το κέντρο ελέγχου αποθηκεύει στο επίπεδο δεδομένων τα νέα στοιχεία. Αυτό γίνεται με τρόπο που είναι προφανής αν μελετήσουμε το σχήμα της βάσης του επιπέδου δεδομένων του jWebDust (Σχήμα 2.4). Ο νέος κόμβος περιμένει επιβεβαιωτικό μήνυμα από το κέντρο ελέγχου. Αν δεν το λάβει, δηλαδή αν δεν είναι βέβαιο ότι έχουν αποθηκευτεί επιτυχώς τα δεδομένα του, ξαναστέλνει το μήνυμα.

Η τεχνική που περιγράφηκε χρησιμοποιεί ένα «εξειδικευμένο» είδος μηνύματος. Οι περισσότερες τεχνικές για την ανίχνευση νέων κόμβων βασίζονται, όπως και η πλειοψηφία των overlays που περιγράφονται σε αυτήν την εργασία, στη χρήση μηνυμάτων με περιγραφές βασισμένες σε XML. Αυτό βοηθά στην καλύτερη αντιμετώπιση της ετερογένειας και τείνει να γίνει κοινή πρακτική, αλλά απαιτεί τη χρήση μεγαλύτερων μηνυμάτων.

Έλεγχος κατάστασης κόμβων

Είναι χρήσιμο να είναι ανά πάσα στιγμή γνωστό στο κέντρο ελέγχου αν όλοι οι κόμβοι είναι ενεργοί και αν όχι, ποιοι δε λειτουργούν ή βρίσκονται σε κατάσταση «sleep». Αυτό μπορεί να οδηγήσει στην αποφυγή αποστολής άσκοπων μηνυμάτων και συνεπώς στην εξοικονόμηση ενέργειας.

Αν κάθε κόμβος στέλνει περιοδικά (με περίοδο που ορίζεται από το χρήστη) μήνυμα που επιβεβαιώνει ότι λειτουργεί και μπορεί να απαντήσει σε queries και αν ενημερώνει, αντίστοιχα, όταν «μπαίνει» σε κατάσταση «sleep», τότε οι παραπάνω στόχοι μπορούν να επιτευχθούν. Φυσικά, για να μην γίνεται σπατάλη μηνυμάτων και ενέργειας, οποιοδήποτε μήνυμα από έναν κόμβο προς το κέντρο ελέγχου θα πρέπει να θεωρείται ως μήνυμα που επιβεβαιώνει το ότι βρίσκεται σε λειτουργία.

Χρονικός Συγχρονισμός

Ήδη έχει φανεί σε πολλά σημεία η ανάγκη για περιοδική αποστολή μετρήσεων, για μετρήσεις που αφορούν και συγκεκριμένες χρονικές στιγμές κτλ. Αυτό καθιστά σαφές ότι χρειάζεται μια τεχνική για το χρονικό συγχρονισμό των κόμβων. Έχουν προταθεί πολλά σχετικά πρωτόκολλα. Το jWebDust, λόγω των χαλαρών απαιτήσεών του

στο συγκεκριμένο ζήτημα, χρησιμοποιεί τον όχι υπερβολικά ακριβή αλγόριθμο της TinyDB. Η μέγιστη απόκλιση που μπορεί να προκύψει με τη χρήση αυτού είναι περίπου της τάξης των millisecond ή μικρότερη. Περισσότερες λεπτομέρειες μπορούν να βρεθούν στο σχετικό documentation, μέσω της επίσημης ιστοσελίδας της TinyDB [24].

Εφαρμογή – Συμπεράσματα

Το jWebDust, παρότι δεν ορίζει ή υλοποιεί συγκεκριμένα πρωτόκολλα σε όλες τις βαθμίδες, είναι από τις πιο ολοκληρωμένες προτάσεις για τη δημιουργία διαδικτυακών εφαρμογών με WSN, είτε θέτοντας προδιαγραφές είτε δίνοντας λύσεις σε όλα τα επίπεδα της αρχιτεκτονικής ενός τέτοιου συστήματος. Είναι αρκετά γενικό και προσανατολισμένο στην παρακολούθηση περιβάλλοντος και χώρων, ενώ δίνει έμφαση και στη δημιουργία διαδικτυακών εφαρμογών με υψηλού επιπέδου γραφικά περιβάλλοντα (βλ. χρήση servlets).

Προβλέπει μια κεντρικοποιημένη δομή, χωρίς αυτό να σημαίνει ότι δε μπορούν να χτιστούν peer-to-peer εφαρμογές που να βασίζονται στα πρωτόκολλά του. Αναλύθηκε σχετικά εκτενέστερα από τα υπόλοιπα και θα αναφερθεί και σε επόμενο κεφάλαιο, όπου σε παραλληλία με την περιγραφή νέου λογισμικού θα γίνει ακόμα πιο κατανοητό το είδος των εφαρμογών που μπορούν να βασιστούν στο overlay του jWebDust.

3. ΕΝΔΙΑΜΕΣΟ ΛΟΓΙΣΜΙΚΟ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ PEER – TO – PEER

Εισαγωγή

Σε αυτό το σημείο πρέπει να τονιστεί και πάλι ότι ο διαχωρισμός των συστημάτων σε peer-to-peer και κεντρικοποιημένα δεν είναι σχεδόν ποτέ απόλυτος και ότι στην περίπτωση των συστημάτων που εξετάζουμε, συμπεριλαμβάνουμε στα peer-to-peer συστήματα αυτά που χρησιμοποιούν τη λογική peer-to-peer για να υλοποιήσουν ορισμένες λειτουργίες. Αυτό δεν πρέπει να παρερμηνεύεται και να θεωρείται ότι αποκλείεται να υπάρχουν κάποιες κεντρικοποιημένες δομές ούτε ότι τα μηχανήματα που συμμετέχουν σε ένα τέτοιο σύστημα είναι ισότιμα. Η λογική είναι συνήθως η δημιουργία οντοτήτων, που μπορεί να είναι και σύνθετες (π.χ. ένα κέντρο ελέγχου μαζί με τους αισθητήρες του ή ένα οποιοδήποτε μηχάνημα που υλοποιεί συγκεκριμένη λειτουργικότητα), που συνεργάζονται χρησιμοποιώντας μεθόδους και αλγόριθμους της τεχνολογίας peer-to-peer. Αν και λαμβάνεται δεδομένο ότι η peer-to-peer λογική, γενικά ως προσέγγιση για την κατασκευή οποιουδήποτε συστήματος, είναι γνωστή στον αναγνώστη αυτής της εργασίας, αξίζει να ανακεφαλαιωθούν σε αυτήν την εισαγωγή ορισμένα πράγματα. Περισσότερες αναφορές στα peer-to-peer συστήματα θα γίνουν και στο κεφάλαιο 4.

Τα peer-to-peer δίκτυα (p2p) είναι μια προσέγγιση εναλλακτική του μοντέλου πελάτη-εξυπηρετητή (client-server model). Ένα p2p δίκτυο παρέχει ένα κλιμακούμενο και ανεκτικό σε σφάλματα μηχανισμό για τον εντοπισμό κόμβων οπουδήποτε σε ένα δίκτυο χωρίς τη διατήρηση ενός μεγάλου σε μέγεθος καταστατικού δρομολόγησης. Τα Peer to Peer συστήματα έγιναν πάρα πολύ δημοφιλή κυρίως γιατί προσέφεραν έναν τρόπο στους ανθρώπους να αποκτούν διάφορα αρχεία χωρίς πληρώνουν για αυτά. Χάρη στον τεράστιο αριθμό ομότιμων συστημάτων, τα αντικείμενα μπορούν να αναπαραχθούν ευρέως, παρέχοντας τη δυνατότητα υψηλής διαθεσιμότητας και κλιμάκωσης, παρά την έλλειψη κεντρικής υποδομής. Μια καλή εισαγωγή για τα p2p συστήματα μπορεί να μελετηθεί στο [25].

Και για αυτά τα συστήματα έχουν καθοριστεί δομημένα overlays όπως τα CAN, Chord, Pastry και Tapestry, παρέχοντας ένα οργανωμένο υπόστρωμα, για τις μεγάλης κλίμακας p2p εφαρμογές. Αυτά ασχολούνται περισσότερο με τη δρομολόγηση, ενώ βασικά ζητήματα για κάθε peer-to-peer σύστημα είναι η ασφάλεια και η διαμοίραση των πόρων, μια και η έλλειψη κεντρικής δομής τα καθιστά πιο απρόβλεπτα και ίσως και πιο ευάλωτα. Μια συνοπτική περιγραφή των overlays και των παραπάνω ζητημάτων υπάρχει στο [26]. Όλα αυτά πρέπει να μελετώνται πριν την κατασκευή ενός peer-to-peer συστήματος. Βέβαια, τα συστήματα που θα παρουσιαστούν εδώ και στο κεφάλαιο 4 αντιμετωπίζουν αυτά τα ζητήματα βασιζόμενα σε κάποια πάγια τακτική ή αφήνουν να τα χειριστεί το χρησιμοποιούμενο overlay (π.χ. JXTA). Για το λόγο αυτό, δε θα δούμε να αναλύονται ιδιαίτερα. Έχουν εξεταστεί και αναλυθεί κατά την επιλογή του overlay.

Μέσα από τα συστήματα που θα περιγραφούν θα γίνει περισσότερο κατανοητή η έννοια του peer-to-peer και κυρίως το πώς αυτή προσαρμόζεται στην περίπτωση των εφαρμογών για WSN και του σχετικού ενδιάμεσου λογισμικού. Τα συστήματα που παρουσιάζονται είναι το Hourglass [19], ένα σχετικά γενικό overlay που σχεδιάστηκε στο πανεπιστήμιο του Harvard και στις πειραματικές του εφαρμογές δίνει έμφαση στον τομέα της υγείας και το Skylark [20], μια πρόταση των ερευνητών της Ericsson για την αξιοποίηση των δυνατοτήτων των WSN στην κινητή τηλεφωνία. Στο τελευταίο γίνεται μάλιστα μια πρώτη αναφορά στη JXTA, που είναι και η τεχνολογία στην οποία βασίζεται η δημιουργία του λογισμικού που θα περιγραφεί στο κεφάλαιο 4.

Hourglass

Η Γενική Ιδέα

Ο στόχος του Hourglass είναι να οριστεί μια εσωτερική δομή (infrastructure) που να ενώνει πολλά, διαφορετικά και διάσπαρτα στο διαδίκτυο δίκτυα αισθητήρων με τις εφαρμογές που επιθυμούν να τα χρησιμοποιήσουν και να κατασκευαστεί ένα σύστημα που να κάνει εφικτή τη δημιουργία εφαρμογών που να βασίζονται στη συγκεκριμένη εσωτερική δομή. Δεν επικεντρώνεται τόσο στη συλλογή και προώθηση δεδομένων εντός ενός δικτύου αισθητήρων και στην επικοινωνία των αισθητήρων με τον σταθμό-βάση (gateway), όσο στη διασύνδεση πολλών δικτύων αισθητήρων μεταξύ τους και στην αρχιτεκτονική του ενδιάμεσου λογισμικού (overlay) που θα επιτρέπει την ανάπτυξη εφαρμογών που θα τα χρησιμοποιούν ταυτόχρονα και με αδιαφάνεια (σαν να πρόκειται για ένα).

Δεν ορίζονται καινούρια πρωτόκολλα για την επικοινωνία με τους αισθητήρες, αλλά φυσικά εννοείται η χρήση κάποιου πρωτοκόλλου ώστε να επιτευχθεί αυτή. Παρόλ' αυτά, δε γίνεται αναφορά σε κάποιο συγκεκριμένο και θεωρούμε ότι σε ένα σύστημα βασισμένο στο Hourglass, η επιλογή για το πώς θα υλοποιείται αυτή η επικοινωνία αφήνεται στον κάθε σταθμό-βάση. Πολύ λογικό, δεδομένου ότι το Hourglass «θέλει» να ενώσει και ήδη υπάρχοντα δίκτυα αισθητήρων.

Η εσωτερική δομή είναι βασισμένη στη δομή του διαδικτύου (Internet-based) και τα κύρια ζητήματα που αντιμετωπίζει είναι η ονομασία (naming), η ανίχνευση (discovery), η διαχείριση της διαφορετικότητας του ορισμού (schema management), η δρομολόγηση (routing) και η προώθηση δεδομένων από πολλά διαφορετικά δίκτυα αισθητήρων. Η εσωτερική αυτή δομή του Hourglass ονομάζεται δίκτυο συλλογής δεδομένων (Data Collection Network - DCN) και παρέχει λειτουργικότητα τόσο για απλά δίκτυα αισθητήρων όσο και για πολύπλοκα (π.χ. αυτά που επιτελούν πολύπλοκη δρομολόγηση, multihop routing κτλ.).

Οι κύριες ιδέες όσον αφορά την αρχιτεκτονική του συστήματος είναι η ύπαρξη κυκλωμάτων (circuits) και υπηρεσιών (services). Τα κυκλώματα είναι κάτι σαν δικτυακές συνδέσεις που ενώνουν τις πηγές δεδομένων δικτύων αισθητήρων με έναν ή περισσότερους παραλήπτες. Η ιδέα του κυκλώματος διαχωρίζει και τα μηνύματα που κυκλοφορούν στο δίκτυο σε βραχύβια (short-lived) και μακρόβια (long-lived). Η σημασία αυτών θα γίνει κατανοητή στη συνέχεια ενώ τα δύο άλλα βασικά συστατικά του Hourglass (που μπορούμε να τα θεωρήσουμε και υπηρεσίες χάριν του τρόπου με τον οποίο υλοποιούνται, αν και διαφέρουν από τις υπόλοιπες υπηρεσίες) είναι το registry (κατανεμημένη βάση δεδομένων όπου καταχωρούνται οι πληροφορίες για τη διαθεσιμότητα των πόρων) και ο διαχειριστής κυκλωμάτων (circuit manager).

Η βασική πρόκληση που αντιμετωπίζει επιτυχώς το Hourglass είναι η αντιμετώπιση των προβλημάτων που μπορούν να προκύψουν από την ύπαρξη διακοπτόμενων, μη αξιόπιστων συνδέσεων. Αυτό γίνεται με μεθόδους προσωρινής αποθήκευσης (buffering).

Ιδιαιτερότητες – Καινοτομίες

Υπάρχουν τέσσερα βασικά πεδία έρευνας από τα οποία το Hourglass χρησιμοποιεί στοιχεία και σε σχέση με τα οποία εμφανίζει καινοτομίες. Τα δίκτυα αισθητήρων, η επεξεργασία συνεχών επερωτήσεων σε βάσεις δεδομένων (continuous query processing), τα συστήματα ομότιμων κόμβων και τα παρόμοια με αυτά (Peer-to-Peer and publish / subscribe systems) και το «πλέγμα» (Grid). Πιο συγκεκριμένα:

- Στον τομέα των δικτύων αισθητήρων:

Το Hourglass δίνει έμφαση στην ροή των δεδομένων μέσα στο διαδίκτυο και προς απομακρυσμένες εφαρμογές και όχι στη δρομολόγηση και επεξεργασία δεδομένων εντός ενός δικτύου αισθητήρων. Σε αυτά τα ζητήματα (ανταλλαγή μηνυμάτων μεταξύ αισθητήρων, μειωμένη κατανάλωση ενέργειας κτλ.), των οποίων η έρευνα σταματά στο σημείο που τα δεδομένα συγκεντρωθούν στο σταθμό-βάση, το Hourglass προσθέτει την έρευνα για την αντιμετώπιση των προβλημάτων που ανακύπτουν κατά την περαιτέρω αδιάφανη διακίνηση των δεδομένων στο δίκτυο και την κατανάλωσή τους από πολλές, διαφορετικές εφαρμογές.

- Στον τομέα του continuous query processing:

Το Hourglass προσθέτει δύο χαρακτηριστικά. Την υποστήριξη του continuous query processing δεδομένης της πιθανότητας διακοπτόμενων συνδέσεων (intermittent connectivity) και τη δυνατότητα επεξεργασίας των δεδομένων όχι με κεντρικοποιημένο τρόπο αλλά οπουδήποτε (κατά προτίμηση στον κόμβο-παραγωγό των δεδομένων ή κοντά σε αυτόν). Κάποια κατανεμημένα συστήματα συλλογής δεδομένων (IrisNet, PIER, Medusa / Aurora) έχουν αντιμετωπίσει παρόμοια ζητήματα αλλά σε σχέση με αυτά, το Hourglass έχει τρεις βασικές διαφορές, λόγω του σχεδιασμού του ειδικά για δίκτυα αισθητήρων: 1) Παρέχει ένα πολύ πιο πλούσιο σύνολο υπηρεσιών για τη συλλογή, το φιλτράρισμα, την προώθηση και την επεξεργασία δεδομένων κατά τη ροή τους μέσα στο δίκτυο. 2) Αντιμετωπίζει το πρόβλημα της κίνησης (mobility) τόσο των αισθητήρων όσο και των συσκευών των σταθμών-βάσεων αλλά και των εφαρμογών. 3) Υποστηρίζει τη δυναμική ενσωμάτωση ετερογενών δικτύων.

- Στον τομέα των Peer-to-Peer συστημάτων:

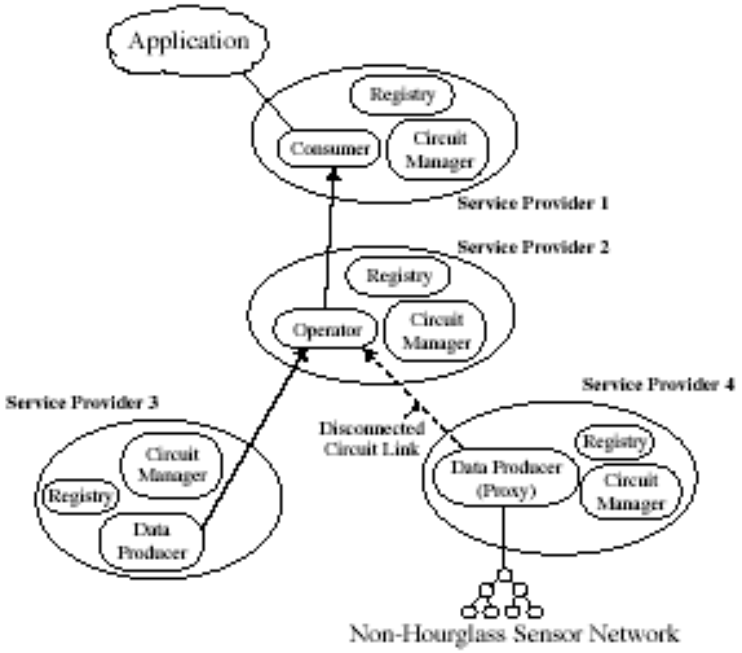
Το Hourglass ακολουθεί καινοτόμα τακτική όσον αφορά τη διαθεσιμότητα των πόρων και την αξιοπιστία. Η γενική αντιμετώπιση του προβλήματος της διαθεσιμότητας των πόρων από τα peer-to-peer συστήματα είναι η πολλαπλή αποθήκευση (replication). Πολλαπλά αντίτυπα των δεδομένων, αποθηκευμένα σε διαφορετικούς κόμβους, εξασφαλίζουν ως ένα βαθμό τη διαθεσιμότητά τους. Σε αντίθεση με αυτήν την τακτική, το Hourglass εφαρμόζει την τακτική της προσωρινής αποθήκευσης (buffering). Τα δεδομένα, όποτε υπάρχει διακοπή σύνδεσης με κάποιο κόμβο από τον οποίο πρέπει να περάσουν, δεν αναζητούνται από αλλού και με διαφορετικές δρομολογήσεις όπως στο replication, αλλά παραμένουν προσωρινά αποθηκευμένα μέχρι να γίνει εφικτή η απαραίτητη σύνδεση για να συνεχίσουν την πορεία τους.

- Σε σχέση με το πλέγμα:

Το Hourglass δεν επικεντρώνεται τόσο στη διανομή των υπολογιστικών πόρων (αν και επιδιώκει παρόμοιο στόχο και χρησιμοποιεί παρόμοιες τακτικές με το να κάνει δυνατή την επεξεργασία των δεδομένων σε οποιοδήποτε σημείο του δικτύου) και δεν υποθέτει την ύπαρξη κόμβων με υψηλές επεξεργαστικές δυνατότητες. Αντίθετα είναι προσανατολισμένο σε δίκτυο πολύ διαφορετικών μεταξύ τους κόμβων, πολλοί από τους οποίους έχουν περιορισμένες δυνατότητες επεξεργασίας ή / και συνδεσιμότητας στο δίκτυο.

Το Σύστημα

Ακολουθεί ένα σχήμα όπου φαίνεται μέσα από ένα απλό παράδειγμα η δομή ενός συστήματος Hourglass:



ΣΧΗΜΑ 3.1 – Παράδειγμα συστήματος Hourglass με ένα υλοποιημένο κύκλωμα

Κάθε ροή δεδομένων στο Hourglass βασίζεται σε ένα κύκλωμα (circuit), το οποίο είναι ένα μονοπάτι δεδομένων μέσα από το σύστημα που σιγουρεύει ότι η εφαρμογή θα λάβει τα δεδομένα για τα οποία ενδιαφέρεται. Το κύκλωμα περιλαμβάνει ενδιάμεσες υπηρεσίες που επιτελούν πράξεις και άλλες λειτουργίες πάνω στα δεδομένα. Οι υπηρεσίες οργανώνονται μέσα σε διακριτούς παροχείς υπηρεσιών (service providers). Ο κάθε παροχέας υπηρεσιών περιλαμβάνει υποχρεωτικά έναν διαχειριστή κυκλωμάτων (circuit manger), που είναι υπεύθυνος για την εγκαθίδρυση και τη διαχείριση των κυκλωμάτων, και ένα registry, το οποίο βοηθά στην ανίχνευση των διαθέσιμων υπηρεσιών.

Οι υπηρεσίες του Hourglass έχουν καλά ορισμένες διεπαφές (interfaces) για να επικοινωνούν με τις άλλες υπηρεσίες, τους διαχειριστές κυκλωμάτων και τα registries. Μια υπάρχουσα οντότητα μπορεί να εισχωρήσει σε ένα σύστημα Hourglass αν υλοποιεί

τη λειτουργικότητα που απαιτείται από αυτές τις διεπαφές. Κάποια δίκτυα αισθητήρων ή κάποιες εφαρμογές μπορούν να αποφασίσουν να συνδεθούν στο σύστημα μέσω ενδιάμεσων εξυπηρετητών (proxy servers) για να αποφύγουν το κόστος του να εκτελούν εγγενώς μια «βαριά» υπηρεσία του συστήματος.

Ακολουθεί περιγραφή των συστατικών μερών της αρχιτεκτονικής του Hourglass.

- Το Κύκλωμα (Circuit):

Το κύκλωμα είναι μια αφαιρετική έννοια που συνδέει ένα σύνολο παραγωγών δεδομένων με έναν καταναλωτή μέσω κόμβων που υλοποιούν ενδο-δικτυακές υπηρεσίες. Αυτά ενοποιούνται σαν μια ροή δεδομένων. Το κύκλωμα επιτρέπει στις εφαρμογές να εκφράσουν τις ανάγκες τους σε «γλώσσα υψηλού επιπέδου», διευκολύνοντας έτσι τη δημιουργία εφαρμογών δικτύων αισθητήρων. Ένα κύκλωμα μπορεί να αναπαρασταθεί με ένα δέντρο που έχει τον καταναλωτή σαν ρίζα και τους παραγωγούς σαν φύλλα. Πολλά κυκλώματα μπορούν να μοιράζονται κοινές φυσικές συνδέσεις, αποφεύγοντας έτσι άσκοπη πολλαπλή μετάδοση δεδομένων τα οποία είναι χρήσιμα σε παραπάνω από ένα κυκλώματα.

Τα κυκλώματα δημιουργούνται από τους διαχειριστές κυκλωμάτων με βάση τις απαιτήσεις των εφαρμογών. Έχουν κάποιο μέγιστο χρόνο ζωής και πρέπει να ανανεώνονται περιοδικά, αλλιώς απομακρύνονται από το σύστημα. Έτσι αποφεύγεται η ύπαρξη κυκλωμάτων-φαντασμάτων μετά από αποτυχίες εφαρμογών. Προφανώς, η δομή των κυκλωμάτων μετά τη δημιουργία τους είναι γνωστή στις εφαρμογές με κατανεμημένο τρόπο.

Για την περιγραφή των προς δημιουργία κυκλωμάτων ορίστηκε μια βασισμένη σε XML γλώσσα περιγραφής κυκλωμάτων, η HCDL (Hourglass Circuit Description Language). Μια περιγραφή κυκλώματος σε αυτή τη γλώσσα μπορεί να υπάρχει σε τρεις μορφές: 1) Η μη ορισμένη μορφή (unrealized circuit) δεν καθορίζει στιγμιότυπα υπηρεσιών (δηλαδή τις διευθύνσεις των παροχέων τους) που θα χρησιμοποιηθούν. 2) Η μερικώς ορισμένη (partially-realized) καθορίζει μερικά από αυτά ενώ 3) Η πλήρως ορισμένη (fully-realized) όλα. Οι δύο πρώτες κατηγορίες πρέπει στην περιγραφή να περιέχουν περιορισμούς για τις υπηρεσίες που πρέπει να χρησιμοποιηθούν, ώστε όταν έρθει η ώρα, ο διαχειριστής κυκλωμάτων να αποφασίσει ποιοι παροχείς υπηρεσιών είναι οι καταλληλότεροι να χρησιμοποιηθούν. Μπορούμε, με λίγα λόγια, μέσω της HCDL, είτε να ορίσουμε από πού ακριβώς θα περνάει ένα κύκλωμα, είτε να δώσουμε περιορισμούς στον διαχειριστή κυκλωμάτων ώστε να αποφασίσει αυτός, είτε, τέλος, να κάνουμε κάτι ενδιάμεσο. Οι περιορισμοί δίνονται με τη βοήθεια των topics και των predicates, τα οποία εξηγούνται κατά την περιγραφή των υπηρεσιών. Ακολουθεί ένα χαρακτηριστικό παράδειγμα μιας μερικώς ορισμένης περιγραφής κυκλώματος, όπου ο καταναλωτής στη διεύθυνση 192.168.0.1:16800 ζητά δεδομένα από μια συγκεκριμένη διεύθυνση εντός του LAN ενός νοσοκομείου (192.168.15.4:16800), αφότου πρώτα τα δεδομένα υποστούν επεξεργασία από οποιαδήποτε υπηρεσία καταχωρημένη με topic BostonEMSDispatch και predicate company=”HMSAmbulance”:

```
<circuit>
  <consumer endpoint="192.168.0.1:16800">
    <operator topic="BostonEMSDispatch">
      <outschema v="bems:dispatchDecision1.0"/>
      <inschema0 v="bems:hospitalAvail1.0"/>
        <producer topic="BostonHospAvail"
                  endpoint="192.168.15.4:16800"/>
          <outschema v="bems:hospitalAvail1.0"/>
        </producer>
      <predicate company="HMSAmbulance"/>
    </operator>
  </consumer>
</circuit>
```

ΣΧΗΜΑ 3.2 – Παράδειγμα μερικώς ορισμένης περιγραφής κυκλώματος σε HCDL

- Η Υπηρεσία (Service):

Οι κόμβοι ενός κυκλώματος υλοποιούνται σαν υπηρεσίες του Hourglass. Μια υπηρεσία μπορεί να λειτουργήσει σαν παραγωγός δεδομένων, σαν καταναλωτής δεδομένων ή και τα δύο μαζί. Η δημιουργία μιας υπηρεσίας δε χρειάζεται να λαμβάνει υπόψιν τη δρομολόγηση και την πολύπλεξη των δεδομένων που παράγει ή καταναλώνει, αφού αυτά γίνονται εξ' ολοκλήρου από ένα Hourglass service layer. Όταν ένα δίκτυο αισθητήρων συνδέεται με το σύστημα μέσω ενδιάμεσου εξυπηρετητή (proxy), τότε η υπηρεσία υλοποιείται εξ' ολοκλήρου στον proxy και απλά απαιτείται επιπλέον ένα πρωτόκολλο επικοινωνίας του proxy με το δίκτυο αισθητήρων (όπως έχει ήδη αναφερθεί αυτό το πρωτόκολλο δεν ορίζεται από το Hourglass).

Οι υπηρεσίες του Hourglass χωρίζονται στις βασικές υπηρεσίες (generic services) και σε αυτές που χρειάζονται από μια συγκεκριμένη εφαρμογή (application-specific services). Παραδείγματα βασικών υπηρεσιών είναι η υπηρεσία προσωρινής αποθήκευσης (buffer service), η υπηρεσία φιλτραρίσματος XML δεδομένων (filter service) και η υπηρεσία μόνιμης αποθήκευσης (persistent storage service).

- Η υπηρεσία προσωρινής αποθήκευσης (buffer service) είναι υπεύθυνη για τη διατήρηση δεδομένων σε κόμβούς όταν δεν υπάρχει σύνδεση (disconnection) με τον κόμβο-προορισμό των δεδομένων και για την σωστή διανομή τους στο κύκλωμα όταν υπάρξει επανασύνδεση (reconnection). Προφανώς αυτή υπηρεσία είναι απαραίτητο να υλοποιείται, τουλάχιστον στους κόμβους όπου η διακοπή σύνδεσης είναι πιθανή (ασύρματες συνδέσεις κτλ.).
- Η υπηρεσία φιλτραρίσματος XML δεδομένων (filter service) εξαρτάται από το μοντέλο δεδομένων που χρησιμοποιεί η εφαρμογή και χρησιμοποιείται προφανώς για να μειώσει την άσκοπη κατανάλωση της χωρητικότητας (bandwidth) του δικτύου. Είναι καλό να υλοποιείται στους παραγωγούς δεδομένων ή κοντά σε αυτούς αλλιώς δε δίνει ιδιαίτερο κέρδος.
- Η υπηρεσία μόνιμης αποθήκευσης (persistent storage service) επιτρέπει στις εφαρμογές να διατηρούν το ιστορικό της ροής των δεδομένων που έχουν περάσει μέσα από ένα κύκλωμα. Αυτό επιτρέπει την πιθανή επανάληψη αποστολής δεδομένων.

Ένα κάπως «ανοιχτό» ζήτημα όσον αφορά τις υπηρεσίες του Hourglass είναι η ανίχνευσή τους, το πώς, δηλαδή, γίνεται γνωστή η ύπαρξή τους στις εφαρμογές. Η προσέγγιση που έχει γίνει είναι η καταχώρηση κάθε υπηρεσίας σε ένα θεματικό πεδίο (topic) και ο περαιτέρω προσδιορισμός της μέσω των predicates. Απαιτείται όμως η σύμβαση ότι οι ενδιαφερόμενοι γνωρίζουν τα topics και τα predicates των υπηρεσιών που επιθυμούν να χρησιμοποιήσουν. Αυτό συμβαίνει γιατί, όπως αναφέρθηκε, οι εφαρμογές πρέπει να δώσουν περιγραφές των κυκλωμάτων που χρειάζονται. Για να γίνει όμως αυτό έχει καταστεί σαφές ότι πρέπει να έχουν γνώση των topics και των predicates (αν όχι των ακριβών διευθύνσεων όπου υλοποιούνται οι επιθυμητές υπηρεσίες).

- Ο Παροχέας Υπηρεσίας (Service Provider):

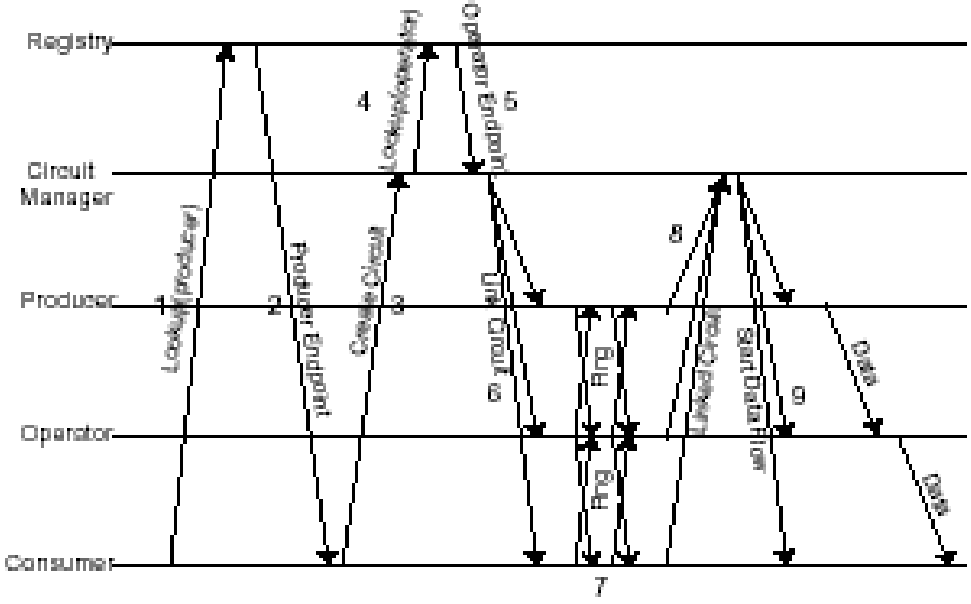
Οι υπηρεσίες είναι οργανωμένες σε παροχείς υπηρεσιών. Κάθε παροχέας μπορεί να περιλαμβάνει έναν ή περισσότερους κόμβους του συστήματος αλλά μπορεί να εισέρχεται και να απέρχεται από το σύστημα μόνο σαν μία ενότητα. Η ελάχιστη λειτουργικότητα που πρέπει να υλοποιεί ένας παροχέας είναι ένα στιγμιότυπο του διαχειριστή κυκλωμάτων συν ένα στιγμιότυπο του registry. Προτείνεται βέβαια και ένα επιπλέον σύνολο υπηρεσιών που είναι χρήσιμες για κάθε παροχέα. Για παράδειγμα, παροχείς που θέλουν να αντιμετωπίζουν το πρόβλημα της ελλιπούς συνδεσιμότητας πρέπει, όπως έχει αναφερθεί, να υλοποιούν και το buffer service. Οι παροχείς υπηρεσιών μπορεί σε τεχνικό επίπεδο να είναι πολύ ετερογενείς μεταξύ τους. Κάποιοι είναι σταθεροί, άλλοι αποσυνδέονται συχνά. Κάποιοι έχουν μεγαλύτερη υπολογιστική ισχύ, άλλοι μικρότερη.

Ρίχνοντας μια ματιά στο σχήμα 3.1 και αναλογιζόμενοι τη λειτουργικότητα των service providers μπορούμε να κατανοήσουμε γιατί το Hourglass είναι ένα peer-to-peer σύστημα. Οι κόμβοι είναι ομότιμοι και συνεργάζονται μεταξύ τους τόσο όσον αφορά την κατανεμημένη υλοποίηση των circuit manager και registry, όσο και όσον αφορά την παροχή των υπηρεσιών. Αυτή η ελάχιστη λειτουργικότητα και η δυνατότητα των παροχέων να λειτουργήσουν είτε σαν παραγωγοί είτε σαν καταναλωτές είτε και τα δύο μαζί είναι κοινή είτε πρόκειται για gateways είτε για clients είτε για οποιονδήποτε ενδιάμεσο κόμβο. Εξ' ου και η ομοτιμία. Άλλωστε, οι αισθητήρες είναι μέλη ενός δικτύου στημένου με το Hourglass αλλά δεν είναι μέλη της αρχιτεκτονικής του συστήματος.

- Ο Διαχειριστής Κυκλωμάτων (Circuit Manager):

Ο διαχειριστής κυκλωμάτων παίρνει σαν είσοδο μια γραμμένη σε HCDL αίτηση για δημιουργία κυκλώματος και επιτελεί τη διαδικασία της δημιουργίας του κυκλώματος. Πρέπει είτε να πάρει μια πλήρως ορισμένη (fully-realized) περιγραφή είτε να παράγει μία με τη βοήθεια του registry. Όταν δημιουργηθούν όλες οι συνδέσεις, ο διαχειριστής σηματοδοτεί την εκκίνηση της παραγωγής και επεξεργασίας των δεδομένων. Ο διαχειριστής κυκλωμάτων μπορεί να αλλάζει τα κυκλώματα και δυναμικά, όποτε αυτό βελτιώνει την ποιότητα των παρεχόμενων υπηρεσιών. Για παράδειγμα, η τροποποίηση ενός κυκλώματος ώστε να περνά από κόμβους που υλοποιούν το buffer service είναι απαραίτητη αν παρατηρηθούν συχνές διακοπές σύνδεσης κτλ.

Ακολουθεί σχηματική αναπαράσταση και στη συνέχεια αναλυτική περιγραφή της διαδικασίας εγκαθίδρυσης ενός κυκλώματος, για την κατανόηση της οποίας χρήσιμη είναι και η κατανόηση της λειτουργίας του registry.



ΣΧΗΜΑ 3.3 – Διαδικασία εγκαθίδρυσης κυκλώματος Hourglass

Η εγκαθίδρυση ενός νέου κυκλώματος ξεκινά από μια εφαρμογή. Για να αλληλεπιδράσει με το Hourglass, μια εφαρμογή πρέπει να υλοποιήσει μια υπηρεσία Hourglass. Αυτό γίνεται είτε άμεσα, αν η εφαρμογή βρίσκεται σε μηχάνημα με επαρκείς πόρους, είτε έμμεσα με τη βοήθεια ενός ενδιάμεσου εξυπηρετητή (proxy) που αναλαμβάνει την αλληλεπίδραση με το Hourglass για λογαριασμό της εφαρμογής.

Ο καταναλωτής που επιθυμεί να εγκαθιδρύσει ένα κύκλωμα φτιάχνει πρώτα μια μη ορισμένη περιγραφή (unrealized description) του επιθυμητού κυκλώματος σε HCDL. Έπειτα, ο καταναλωτής πρέπει να ανακτήσει από το registry ένα κατάλληλο σύνολο παραγωγών (βήμα 1 στο σχήμα). Ο καταναλωτής κάνει την ερώτηση στο registry και παίρνει ως απάντηση ένα σύνολο ταιριαστών παραγωγών (βήμα 2). Αν το σύνολο που επιστραφεί είναι πολύ μεγάλο, ο καταναλωτής μπορεί να «σφίξει» τους περιορισμούς με χρήση αυστηρότερων predicates. Μπορεί επίσης, αν θέλει, να ορίσει αυστηρά ορισμένους από τους ενδιάμεσους κόμβους (operators) δίνοντας ακριβείς διευθύνσεις. Μετά από όλα αυτά, ο καταναλωτής ανανεώνει την περιγραφή του κυκλώματος ώστε να περιλαμβάνει τους κατάλληλους παραγωγούς και στέλνει στον διαχειριστή κυκλωμάτων μια μερικώς ορισμένη περιγραφή, δίνοντάς του εντολή να δημιουργήσει το κύκλωμα (βήμα 3).

Είναι ευθύνη του διαχειριστή κυκλωμάτων να ορίσει τις υπόλοιπες υπηρεσίες (αυτές, δηλαδή, για τις οποίες δεν έχουν οριστεί ακριβείς διευθύνσεις). Το πραγματοποιεί μέσω περαιτέρω ερωτήσεων προς το registry (βήματα 4,5). Αφότου η περιγραφή γίνει

πλήρως ορισμένη, ο διαχειριστής κυκλωμάτων επικοινωνεί με τις επιλεγμένες υπηρεσίες (βήμα 6). Οι υπηρεσίες παραλαμβάνουν αντίγραφα της περιγραφής του κυκλώματος και δημιουργούν συνδέσμους με τους κόμβους που είναι «παιδιά» τους και με αυτους που είναι «γονείς» τους στο δέντρο του κυκλώματος (βήμα 7). Ύστερα, οι υπηρεσίες αναφέρουν πίσω στον διαχειριστή την επιτυχή δημιουργία των εν λόγω συνδέσμων (βήμα 8). Όταν έχει λάβει μηνύματα επιτυχούς δημιουργίας για όλους τους συνδέσμους, ο διαχειριστής κυκλωμάτων εκκινεί τη ροή των δεδομένων (βήμα 9).

- Το registry:

Το registry είναι μια κατανεμημένη αποθήκη πληροφορίας σχετικά με τις υπάρχουσες υπηρεσίες και τα υπάρχοντα κυκλώματα. Μπορεί να υλοποιηθεί με διάφορους τρόπους και με διάφορες τεχνολογίες. Χαρακτηριστικά, αναφέρονται τα πρωτόκολλα Pastry και Chord.

Οι πληροφορίες για τις υπάρχουσες υπηρεσίες αποθηκεύονται στο registry για να μπορούν οι εφαρμογές να εντοπίσουν τις διευθύνσεις των παραγωγών που τις ενδιαφέρουν ενώ οι πληροφορίες για τα υπάρχοντα κυκλώματα αποθηκεύονται στο registry για να επιτρέπουν στους διαχειριστές κυκλωμάτων να επιτελούν δυναμική βελτιστοποίηση.

Κάθε παροχέας υπηρεσιών πρέπει να έχει ένα registry, τόσο για να αντιμετωπίζονται τα προβλήματα των διακοπών σύνδεσης μερικών κόμβων όσο και για να μοιράζεται ο φόρτος της δουλειάς που επιτελεί το registry. Το registry πρέπει οπωσδήποτε να περιέχει όλες τις απαραίτητες πληροφορίες για τις τοπικές υπηρεσίες, μαζί με ό,τι άλλες πληροφορίες περιέχει για υπηρεσίες που βρίσκονται σε άλλους παροχείς υπηρεσιών.

Όταν μια υπηρεσία ενεργοποιείται πρέπει να δηλώσει την ύπαρξή της στο registry. Αυτό γίνεται με τα announce messages, μηνύματα που πρέπει να περιέχουν μια διεύθυνση (endpoint) η οποία παρέχει την υπηρεσία, το topic με το οποίο συνδέεται θεματικά η υπηρεσία και ένα χρόνο ζωής (lease time), που δηλώνει το χρονικό διάστημα για το οποίο οι πληροφορίες για την υπηρεσία θα παραμείνουν στο registry. Επίσης, το announce message μπορεί να περιλαμβάνει κάποια predicates.

Εφαρμογή – Συμπεράσματα

Η εφαρμογή του Hourglass αφορούσε τον τομέα της υγείας (ενδο-νοσοκομειακή επικοινωνία και συντονισμός ασθενοφόρων και επειγόντων περιστατικών) αλλά έγινε στο περιβάλλον προσομοίωσης ModelNet, για να μπορούν να δοκιμαστούν περιπτώσεις μεγάλων συστημάτων. Από τις δυνατές υλοποιήσεις για τα συστατικά που περιγράφηκαν (π.χ. registry και circuit manager) χρησιμοποιήθηκαν οι απλούστερες.

Το κυριότερο που ελέγχθηκε ήταν οι αποκρίσεις του συστήματος σε περιπτώσεις διακοπών σύνδεσης και τα αποτελέσματα ήταν θετικά αφού η επανασυνδέσεις των αποσυνδεδεμένων μελών ανιχνεύονταν εντός 5-10 δευτερολέπτων και η διαδικασία προώθησης των δεδομένων συνεχιζόταν επιτυχώς.

Χαρακτηριστικά σενάρια λειτουργικότητας της εφαρμογής είναι η ενημέρωση των PDA των γιατρών για τυχόν αλλαγές στην κατάσταση των ασθενών τους, επιλογή νοσοκομείου από τους οδηγούς ασθενοφόρων με βάση την κίνηση, τη διαθεσιμότητα κτλ

Skylark

Η Γενική Ιδέα

Το σύστημα αυτό έχει ως στόχο να δημιουργήσει μια πλατφόρμα που να υποστηρίζει την ανάπτυξη peer-to-peer εφαρμογών πάνω στο δίκτυο κινητής τηλεφωνίας, με απώτερο σκοπό την εκμετάλλευση των δυνατοτήτων που μπορούν να προσφέρουν οι εφαρμογές δικτύων αισθητήρων. Κάθε συνδρομητής είναι μια εν δυνάμει πηγή πληροφοριών (αν στα κινητά τηλέφωνα ενσωματωθούν οι κατάλληλοι αισθητήρες), απλά οι πληροφορίες αυτές θα είναι τυχαίες (ad-hoc) χωρίς να υπάρχει, δηλαδή, έλεγχος στο τι αναφέρουν, από πού το αναφέρουν και πότε είναι συνδεδεμένοι. Το αποτέλεσμα όμως που θα προέκυπτε θα ήταν παρόλαυτά η πληρέστερη λύση για τις κλασσικές εφαρμογές δικτύων αισθητήρων γενικού ενδιαφέροντος (έλεγχος κίνησης, παρατήρηση περιβάλλοντος κτλ.).

Το πρόβλημα με τα υπάρχοντα δίκτυα κινητής τηλεφωνίας τρίτης γενιάς (3G) είναι ότι είναι ιεραρχικά οργανωμένα και πολύ σφιχτά ελεγχόμενα από τους χειριστές τους. Οι σχετικές εφαρμογές βρίσκονται επίσης υπό αυστηρό έλεγχο και παρέχονται στους συνδρομητές από τα κέντρα υπηρεσιών. Οι χρήστες δε μπορούν να παρέχουν υπηρεσίες σε άλλους χρήστες απευθείας. Πρέπει πρώτα να κάνουν συμφωνία με το χειριστή του δικτύου και να στηθεί η απαραίτητη εσωτερική δομή (infrastructure) μέσα στο κέντρο υπηρεσιών. Γίνεται εύκολα κατανοητό γιατί όλα αυτά καθιστούν απαγορευτική τη δημιουργία peer-to-peer εφαρμογών και την υλοποίηση του οράματος δικτύων αισθητήρων όπου κάθε κινητό τηλέφωνο θα είναι σταθμός-βάση (gateway).

Άλλα χαρακτηριστικά που οδήγησαν στην ανάπτυξη του συγκεκριμένου συστήματος βασίζονται στη μορφή των δικτύων αισθητήρων που επιθυμεί να ενώσει. Το σύστημα έχει ως στόχο να χρησιμοποιεί μικρού μεγέθους (small-scale) δίκτυα αισθητήρων και όχι μεγάλα δίκτυα όπως αυτά στα οποία έχει επικεντρωθεί η έρευνα των δικτύων αισθητήρων (δηλαδή με πολλούς, αυτό-οργανούμενους αισθητήρες, με multihop δρομολόγηση και άλλα πολύπλοκα πρωτόκολλα). Τα μικρά δίκτυα αισθητήρων (αυτά, δηλαδή, με ένα σταθμό-βάση και μικρό αριθμό αισθητήρων που επικοινωνούν με αυτόν) είναι εύκολα στην υλοποίηση και προσφέρονται για εφαρμογές που ενδιαφέρουν τους απλούς συνδρομητές (π.χ. για εφαρμογές στον τομέα της υγείας κτλ.). Χαρακτηριστικό αυτών των δικτύων είναι η κίνηση (mobility – και μάλιστα τυχαία) και το γεγονός ότι η τοπολογία του δικτύου και των υπηρεσιών που αυτό προσφέρει (δηλαδή το ποιες περιοχές μπορούν να παρακολουθηθούν και με τι αισθητήρες δε μπορεί να είναι γνωστό εκ των προτέρων). Επίσης σημαντικό είναι να ληφθεί υπόψη ότι κάποια από τα δεδομένα είναι προσωπικής φύσεως. Συνοψίζοντας, τα ζητήματα που πρέπει να αντιμετωπίζονται από το σύστημα είναι:

- Η δυνατότητα δυναμικής ανίχνευσης των τυχαίων υπηρεσιών (ad-hoc services) που θα προσφέρονται και η «παρακολούθησή» τους σε ένα περιβάλλον κίνησης.
- Η δημιουργία συνδέσεων μεταξύ των δικτύων αισθητήρων με βάση το είδος και της σχετικότητας της πληροφορίας που αυτά μπορούν να παρέχουν.
- Η κατανεμημένη οργάνωση του συνολικού δικτύου, έτσι ώστε κάθε μέλος του δικτύου να μπορεί να είναι χρήστης και παροχέας υπηρεσιών ταυτόχρονα.

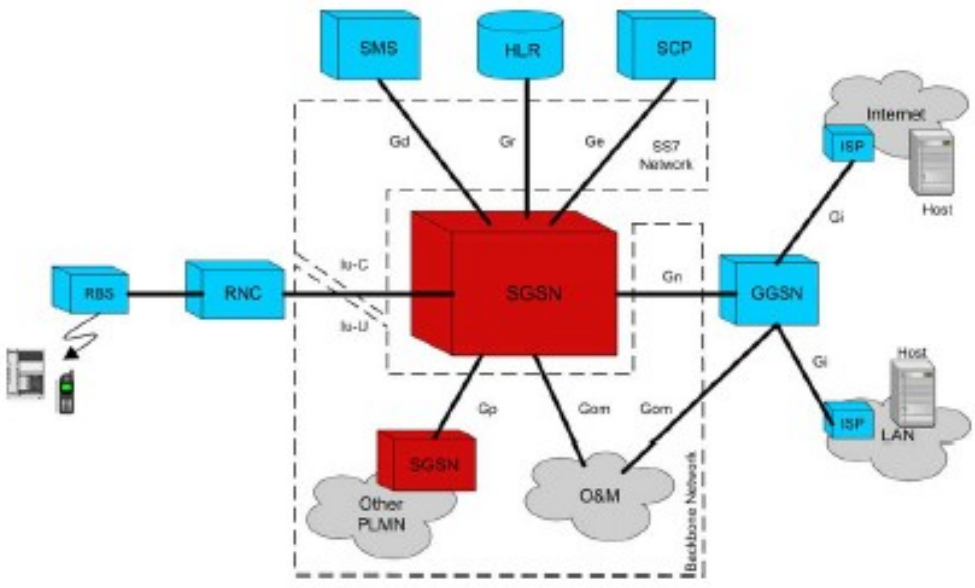
Έχοντας κατά νου όλα τα παραπάνω, η ιδέα είναι η ύπαρξη μιας πλατφόρμας ενδιάμεσου λογισμικού (overlay) η οποία θα προσφέρει peer-to-peer λειτουργικότητα

πάνω στο ήδη υπάρχον (ή λίγο τροποποιημένο) δίκτυο κινητής τηλεφωνίας, καθώς και ο καθορισμός των πρωτοκόλλων με τα οποία θα γίνεται η επικοινωνία των συσκευών με τους αισθητήρες τους. Για το πρώτο ζήτημα χρησιμοποιείται ως βάση η τεχνολογία JXTA, ενώ για το δεύτερο ορίζονται κάποια πρωτόκολλα όπως το XMLSens. Λεπτομέρειες θα αναλυθούν στην περιγραφή του συστήματος.

Ιδιαιτερότητες – Καινοτομίες

Η δυνατότητα δημιουργίας peer-to-peer εφαρμογών δικτύων αισθητήρων στο δίκτυο κινητής τηλεφωνίας είναι εκ των πραγμάτων κάτι εξ' ολοκλήρου καινοτόμο. Τεχνικά, οι βασικές ιδιαιτερότητες του συστήματος είναι η τροποποίηση της πλατφόρμας JXTA και η προσαρμογή της στις ανάγκες του δικτύου κινητής τηλεφωνίας και η δημιουργία των πρωτοκόλλων για την επικοινωνία συσκευών – αισθητήρων. Αυτά θα αναλυθούν στην περιγραφή του συστήματος.

Το πρόγραμμα όμως αυτό παρουσιάζει και προτείνει καινοτομικές αλλαγές στη δομή του δικτύου κινητής τηλεφωνίας έτσι ώστε να μπορεί να υποστηριχθεί με τον καλύτερο δυνατό τρόπο η ανάπτυξη εφαρμογών δικτύων αισθητήρων. Αποφεύγοντας τις πολλές τεχνικές λεπτομέρειες του δικτύου κινητής τηλεφωνίας, παρουσιάζεται στο παρακάτω σχήμα η γενική μορφή του και μετά από μια πολύ συνοπτική περιγραφή, παρατίθενται οι προτεινόμενες καινοτομικές αλλαγές.



ΣΧΗΜΑ 3.4 – Τα συστατικά του δικτύου ενός WCDMA συστήματος

Τα κύρια συστατικά είναι το SGSN (Serving GPRS Support Node) και το GGSN (Gateway GPRS Support Node). Το SGSN παρέχει τη δυνατότητα του χειρισμού της κίνησης (mobility) των χρηστών και του ελέγχου των συνδέσεων (sessions) των συσκευών των χρηστών. Το GGSN είναι η πύλη προς τα εξωτερικά IP δίκτυα (π.χ. Internet). Οι υπόλοιπες υπηρεσίες χωρίζονται σε αυτές που χρησιμοποιούν το IP πρωτόκολλο (π.χ. radio network) και αυτές που χρησιμοποιούν το SS7 (Signaling System No. 7 – π.χ. υπηρεσία SMS, HLR κ.α.). Ο λόγος που γίνεται αναφορά σε αυτά θα

καταστεί σαφής στην περιγραφή του συστήματος. Η ανάπτυξη εφαρμογών κινητής τηλεφωνίας με τον παραδοσιακό τρόπο απαιτεί επικοινωνία της εφαρμογής με κάποιους κόμβους του δικτύου μέσω του SS7. Αυτό το γεγονός μαζί με την γενικότερη κεντρικοποιημένη προσέγγιση που έχει ακολουθηθεί ως τώρα καθιστούν δύσκολη τη δημιουργία εφαρμογών μαζικού ενδιαφέροντος, όπως αυτές που μας ενδιαφέρουν. Η προσέγγιση του συστήματος Skylark προτείνει τις εξής καινοτομίες στα δίκτυα κινητής τηλεφωνίας επόμενων γενεών:
- Να μην περιορίζεται η λειτουργικότητα της συσκευής χρήστη (edge device) με τον περιορισμό του ελέγχου όλων των λειτουργιών της από τον πυρήνα του δικτύου (core network).
- Να επιτραπεί στις εφαρμογές να εκμεταλλεύονται τη δυνατότητα ασύρματης επικοινωνίας μεταξύ των συσκευών των χρηστών.
- Να εμπλουτιστούν οι συσκευές με το απαραίτητο λογισμικό ώστε να μπορεί η λογική των εφαρμογών να στεγαστεί σε αυτές και να υπάρχει έτσι αυτόνομο υπολογιστικό περιβάλλον (Viral / Agent model of software creation).
- Να είναι όλες οι συσκευές ικανές για εντοπισμό θέσης, δυνατότητα πολύ κρίσιμη για τις εφαρμογές κινητών δικτύων αισθητήρων.
- Να αναπτυχθούν νέες, ευέλικτες τεχνικές διευθυνσιοδότησης για να μπορούν να εντοπίζονται σε πραγματικό χρόνο οι πόροι από τους ενδιαφερόμενους.
- Να τροποποιηθεί η τεχνική με την οποία γίνονται οι συνδέσεις κλήσεων (call sessions) γιατί ο τρόπος με τον οποίο υλοποιούνται τώρα είναι πολύ περιοριστικός για ένα δίκτυο με προσαρτημένα δίκτυα αισθητήρων στα άκρα του.

Τα παραπάνω είναι απλές προτάσεις. Η αρχιτεκτονική του συστήματος Skylark, του οποίου ακολουθεί η περιγραφή, είναι βασισμένη στο υπάρχον δίκτυο.

Το Σύστημα

Το Skylark είναι μια πλατφόρμα για ασύρματα peer-to-peer δίκτυα που αναπτύχθηκε για να καταστήσει εφικτή τη συνεργασία των δικτύων αυτών με δίκτυα αισθητήρων. Η πλατφόρμα έχει δύο κύρια συστατικά. Το σταθμό-βάση (gateway) και το τροποποιημένο ενδιάμεσο λογισμικό (middleware) JXTA.

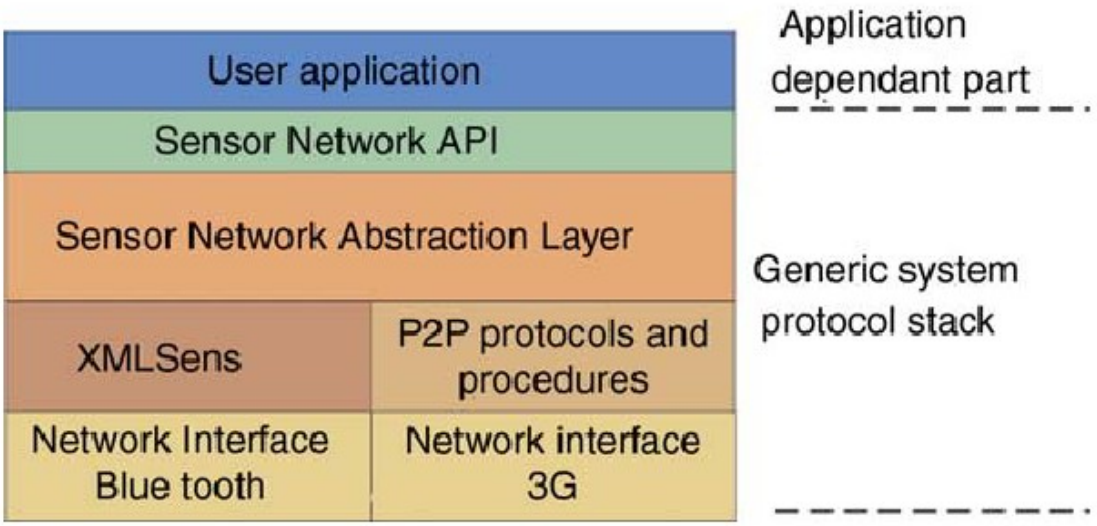
Η Αρχιτεκτονική του σταθμού-βάση

Υπενθυμίζεται ότι το σύστημα αποσκοπεί στην ένωση μικρών και απλών δικτύων αισθητήρων, όπου για κάθε δίκτυο υπάρχει ένας σταθμός-βάση που αναλαμβάνει εξ' ολοκλήρου την ευθύνη της οργάνωσης των αισθητήρων, της δρομολόγησης των μηνυμάτων σε αυτούς και την επικοινωνία με το υπόλοιπο δίκτυο. Οι αισθητήρες δεν επιτελούν καμία από αυτές τις λειτουργίες (όπως γίνεται στα συμβατικά ασύρματα δίκτυα αισθητήρων, όπου θεωρείται δεδομένη η πιθανότητα ύπαρξης μεγάλου αριθμού αισθητήρων). Επίσης, ο υπολογιστικός φόρτος για πολλούς υπολογισμούς μεταφέρεται από τους αισθητήρες στους σταθμούς-βάση, κάνοντας έτσι εφικτή τη δημιουργία φθηνότερων και απλούστερων δικτύων αισθητήρων (βλ. και κεφ.1 – Εισαγωγή στα δίκτυα αισθητήρων). Δεν πρέπει άλλωστε να ξεχνάμε ότι το συγκεκριμένο σύστημα αποσκοπεί στην εκμετάλλευση και των κινητών τηλεφώνων σαν σταθμούς-βάσεις,

συνεπώς η συντριπτική πλειοψηφία των χρηστών δε μπορεί να έχει συσκευές με πολλούς και ακριβούς αισθητήρες.

Οι σταθμοί-βάσεις επικοινωνούν με τους αισθητήρες μέσω ασύρματων διεπαφών (wireless interfaces) μικρής εμβέλειας. Αφότου γίνει η σύνδεση, ο σταθμός-βάση συλλέγει όλες τις πληροφορίες σχετικά με τους αισθητήρες και τις παρέχει με λογικό τρόπο (αποκρύπτοντας τεχνικές πληροφορίες) στους ενδιαφερόμενους. Για την επικοινωνία σταθμών-βάσεων – αισθητήρων χρησιμοποιείται το πρωτόκολλο XMLSens, που θα περιγραφεί στη συνέχεια. Αυτό κάνει την αρχιτεκτονική ευέλικτη και ικανή να υποστηρίξει διάφορες τεχνολογίες επικοινωνίας (στην πειραματική υλοποίηση χρησιμοποιείται Bluetooth), διάφορα είδη αισθητήρων και διάφορες εφαρμογές, χωρίς να χρειάζονται ιδιαίτερες μετατροπές.

Το παρακάτω σχήμα απεικονίζει τα επίπεδα της αρχιτεκτονικής του σταθμού-βάσης (και στην ουσία ολόκληρη την ιδέα του Skylark) με τα δύο χαμηλότερα να χωρίζονται σε αυτά που αφορούν την επικοινωνία σταθμού-βάσης – αισθητήρων (αριστερά) και αυτά που αφορούν την επικοινωνία των σταθμών-βάσεων μεταξύ τους και με το υπόλοιπο δίκτυο.



ΣΧΗΜΑ 3.5 – Η Αρχιτεκτονική ενός σταθμού-βάσης σε ένα p2p δίκτυο

Το πρωτόκολλο XMLSens και το επίπεδο αφαίρεσης

Χρήση τέτοιου είδους πρωτοκόλλων (βασισμένων σε XML) για την επικοινωνία αισθητήρων – σταθμών βλέπουμε και σε άλλα σχετικά συστήματα και κατά κοινή ομολογία αποτελεί την καλύτερη λύση όσον αφορά την ευελιξία και επεκτασιμότητα του συστήματος. Η διαδικασία που ακολουθεί το συγκεκριμένο πρωτόκολλο είναι η εξής:
Ο σταθμός-βάση, όταν ανιχνεύσει την ύπαρξη ενός νέου αισθητήρα, του στέλνει ένα μήνυμα τύπου GetSensorList και ο αισθητήρας απαντά με ένα μήνυμα τύπου GetSensorListResponse. Η μορφή αυτών είναι η παρακάτω:

- GetSensorList

```
<?xml version="1.0" encoding="UTF-8"?>
  <XMLSensCommand>
    <name>
      GetSensorList
    </name>
  </XMLSensCommand>
```

- GetSensorListResponse

```
<?xml version="1.0" encoding="UTF-8"?>
  <XMLSensCommand>
    <name>
      GetSensorListResponse
    </name>
  <sensor>
    <attributeProfile>
    ...
    </attributeProfile>
    <dataProfile>
      <dataAttribute>
      ...
      </dataAttribute>
    </dataProfile>
  </sensor>
  <sensor>
  ...
  </sensor>
  </XMLSensCommand>
```

Το attribute profile περιέχει πληροφορίες για τα βασικά χαρακτηριστικά του αισθητήρα, όπως ο τύπος, η τοποθεσία, ο κατασκευαστής, η ακρίβεια, η εμβέλεια, η ημερομηνία ενεργοποίησης, η A/D ανάλυσή του κτλ.
Το data profile περιέχει πληροφορίες για τον τύπο των δεδομένων (δηλαδή των αποστελλόμενων τιμών) που παράγει. Αν, δηλαδή, είναι ακέραιος ή κινητής υποδιαστολής, πόσα byte καταλαμβάνει κτλ.
Ακολουθεί χαρακτηριστικό παράδειγμα που περιγράφει αισθητήρα θερμοκρασίας που βρίσκεται στο δωμάτιο 22 του δεύτερου ορόφου.

```
<?xml version="1.0" encoding="UTF-8"?>
<sensor>
  <attributeProfile>
    <attribute name = "type">
      temperature
    </attribute>
  <attribute name="location">
    <attribute name="building">
      Radio house
    </attribute>
    <attribute name="floor">
      2
    </attribute>
    </attribute name="room">
      22
    </attribute>
  </attribute>
  </attributeProfile>
  <dataProfile>
    <dataAttribute>
      <Adresolution>
        12
      </Adresolution>
      <samplingRate>
        200
      </samplingRate>
      <sampleSize>

      4
    </samplesize>
    </dataAttribute>
  </dataProfile>
</sensor>
```

Τέλος, η αποστολή δεδομένων ξεκινά με την αποστολή ενός μηνύματος GetData από τους σταθμούς στους αισθητήρες. Αυτό έχει την παρακάτω μορφή καθορίζοντας χαρακτηριστικά όπως η περίοδος αποστολής τιμών κτλ.

```
<?xml version="1.0" encoding="UTF-8"?>
<XMLSensCommand>
    <name>
        GetData
    </name>
    <sensor>
        <sensorID>
            2
        </sensorID>
    </sensor>
    <transmitMode>
    <attribute name="periodical">
        <attribute name="period">
            5
        </attribute>
    <attribute name="periodUnit">
        min
    </attribute>
    </transmitMode>
    <dataFormat>
        <attribute name="type">
            byte
        </attribute>
        <attribute name="totalNumber">
            10000
        </attribute>
        <attribute name="sampleRate">
            200
        </attribute>
    </dataFormat>
</XMLSensCommand>
```

Υπάρχουν επίσης και κάποιοι άλλοι τύποι μηνυμάτων.

Όλες αυτές οι λεπτομέρειες όμως δεν είναι ορατές στην εφαρμογή. Το επίπεδο αφαίρεσης (Abstraction Layer) που υλοποιείται από το σταθμό-βάση προσφέρει στο δημιουργό της εφαρμογής μια διεπαφή προγραμματισμού εφαρμογών (API) που διευκολύνει τη δημιουργία εφαρμογών. Οι μέθοδοι του εν λόγω API δίνουν στους χρήστες τη δυνατότητα να προσκομίσουν γενικές περιγραφές των δικτύων αισθητήρων και να ζητήσουν από αυτά τα επιθυμητά δεδομένα. Οι μέθοδοι αυτές είναι ανεξάρτητες του τύπου των αισθητήρων και άλλων ιδιαίτερων χαρακτηριστικών του δικτύου. Έτσι οι εφαρμογές που θα έχουν γραφτεί με αυτό το API είναι μεταφέρσιμες σε οποιαδήποτε πλατφόρμα. Τα μηνύματα που θα ανταλλάσσονται από τις εφαρμογές για να γίνεται η ανταλλαγή των δεδομένων είναι και αυτά βασισμένα σε XML (Το απαιτεί άλλωστε η JXTA) αλλά περιέχουν πληροφορίες σχετικές μόνο με τα δεδομένα και όχι με τα δίκτυα, την τοπολογία τους και τις λεπτομέρειες υλοποίησης τους.

Η Αρχιτεκτονική του ενδιάμεσου P2P λογισμικού

Η JXTA είναι ένα σύνολο πρωτοκόλλων που ορίζουν έναν τρόπο με τον οποίο υλοποιούνται τα βασικά συστατικά ενός peer-to-peer δικτύου. Έτσι, κάθε υλοποίηση της JXTA (αυτή τη στιγμή υπάρχουν υλοποιήσεις σε Java και C) είναι ένα εργαλείο για την κατασκευή peer-to-peer δικτύων τα οποία θα είναι μεταφέρσιμα σε οποιαδήποτε πλατφόρμα με την ελάχιστη απαιτούμενη λειτουργικότητα για να μπορεί να φιλοξενήσει την υλοποίηση. Γίνεται αφαίρεση όλων των εννοιών που μπορεί να είναι περιοριστικές (π.χ. οι διευθύνσεις απεικονίζονται σε JXTA Endpoints, το κάθε μηχάνημα απεικονίζεται

σε έναν Peer που έχει ίδιες δυνατότητες με τους υπόλοιπους κτλ.). Αναλυτικότερη περιγραφή του JXTA Overlay θα γίνει στο επόμενο κεφάλαιο, αφού η JXTA χρησιμοποιείται και για την ανάπτυξη της δικής μας πειραματικής εφαρμογής. Ως σύνολο πρωτοκόλλων που αποσκοπούν στη θέσπιση προτύπων για τα peer-to-peer δίκτυα, η JXTA δεν είναι τόσο λογικό να τροποποιηθεί παρά μόνο με προσθήκες. Αυτό έγινε και για την ανάπτυξη του Skylark. Έγιναν προσθήκες στην Java υλοποίηση της JXTA. Οι προσθήκες αυτές ήταν δύο ειδών. Αυτές που σχετίζονται με τον πυρήνα της JXTA και αυτές που έγιναν «περιφερειακά» με σκοπό απλά την υποστήριξη χαρακτηριστικών που σχετίζονται με τους αισθητήρες.

Προσθήκες στον πυρήνα της JXTA

- Υποστήριξη των συνδέσεων μέσω του πρωτοκόλλου GPRS.

Προστέθηκε ένα νέο Endpoint πρωτόκολλο για να μπορεί μια διεύθυνση Endpoint να αντιστοιχεί και σε διεύθυνση GPRS (εκτός από IP κτλ.). Η διεύθυνση Endpoint της JXTA είναι μια αφαίρεση της έννοιας της διεύθυνσης που είναι ανεξάρτητη του μηχανισμού που χρησιμοποιείται για τη μεταφορά των δεδομένων. Βασίζεται στον τύπο διευθύνσεων URI και έχει τη μορφή:
πρωτόκολλο://διεύθυνση_δικτύου/όνομα_υπηρεσίας/παράμετροι_υπηρεσίας
Ένας περιορισμός του GPRS, για παράδειγμα, είναι το γεγονός ότι δεν υποστηρίζει IP multicasting. Έτσι, υλοποιήθηκαν τα πρωτόκολλα GPRSEndpointProtocol, GPRSEndpointMessenger και GPRSEndpointAddress για να κατασκευαστεί η κλάση GPRSAddress, η οποία κάνει δυνατή την αναπαράσταση των διευθύνσεων με τη μορφή:
gprs://ip:θύρα/προαιρετικές_παράμετροι

Η κατασκευή, βέβαια, μιας τέτοιας GPRS διεύθυνσης απαιτούσε πολλές προσθήκες καθώς και την αποθήκευση επιπλέον πληροφοριών για τα χαρακτηριστικά της GPRS μετάδοσης. Τα μηνύματα XML που χρησιμοποιούνται από την JXTA περιέχουν πολύ περιττή πληροφορία για αυτό το σκοπό, αλλά η εναλλακτική λύση, δηλαδή η χρήση της JXME (έκδοση της JXTA για μικρο-συσκευές) απορρίφθηκε αμέσως για λόγους ασφάλειας. Κι αυτό γιατί η JXME βασίζεται σε μια proxy αρχιτεκτονική, όπου ο σταθμός-βάση θα έπρεπε να «τρέχει» έναν απλό client, του οποίου η λειτουργικότητα θα υλοποιείτο σε κάποιον ενδιάμεσο εξυπηρετητή (proxy), ο οποίος, δεδομένης της αρχιτεκτονικής του δικτύου κινητής τηλεφωνίας, θα έπρεπε να τρέχει μέσα στον πυρήνα του δικτύου, πράγμα πολύ επικίνδυνο.

- Αυτόνομη ρύθμιση των rendezvous κόμβων

Η JXTA χρησιμοποιεί την προσέγγιση των rendezvous κόμβων για την προώθηση των μηνυμάτων (βλ. κεφ. 4). Μια και, όπως αναφέρθηκε, το GPRS πρωτόκολλο δεν υποστηρίζει broadcasting, πρέπει στο σύστημα Skylark να γίνει χρήση τέτοιων κόμβων. Κάθε χρήστης όμως (που στην περίπτωσή μας είναι και gateway) πρέπει να συνδέεται με έναν τουλάχιστον τέτοιο κόμβο με το που συνδέεται στο δίκτυο. Δεδομένων των περιορισμών που υπάρχουν στα ασύρματα δίκτυα, ο φόρτος για αυτήν την επικοινωνία θα ήταν ιδιαίτερα επιβαρυντικός. Έτσι, επινοήθηκε ένας τρόπος σύνδεσης των χρηστών με τους rendezvous μέσω της υπηρεσίας SMS, κάτι που δεν επιβαρύνει. Για να υποστηριχθεί αυτό, έπρεπε να γίνουν οι σχετικές προσθήκες στην υλοποίηση της JXTA.

Προσθήκες στη JXTA για υπηρεσίες αισθητήρων

- Ευέλικτη υπηρεσία εντοπισμού θέσης.

Υπάρχουν πολλοί τρόποι για τον εντοπισμό θέσης (π.χ. GPS, MPS, RFID κτλ.). Η προσέγγιση που ακολουθήθηκε είναι ένας κατανεμημένος p2p αλγόριθμος που ορίζει ένα interface σε επίπεδο εφαρμογής, το οποίο είναι ανεξάρτητο από τον τρόπο που χρησιμοποιείται «από κάτω». Έτσι, μια αίτηση για εντοπισμό θέσης προωθείται πάντα στο δίκτυο και αυτό επιστρέφει τον πιο ταιριαστό κόμβο για να απαντήσει. Σε αυτόν τον αλγόριθμο κάθε συσκευή κρατά πληροφορίες για τη θέση της ίδιας και κάποιων «γειτόνων» της και τις παρέχει στους ενδιαφερόμενους,

- Επέκταση του μοντέλου διευθύνσεων της JXTA.

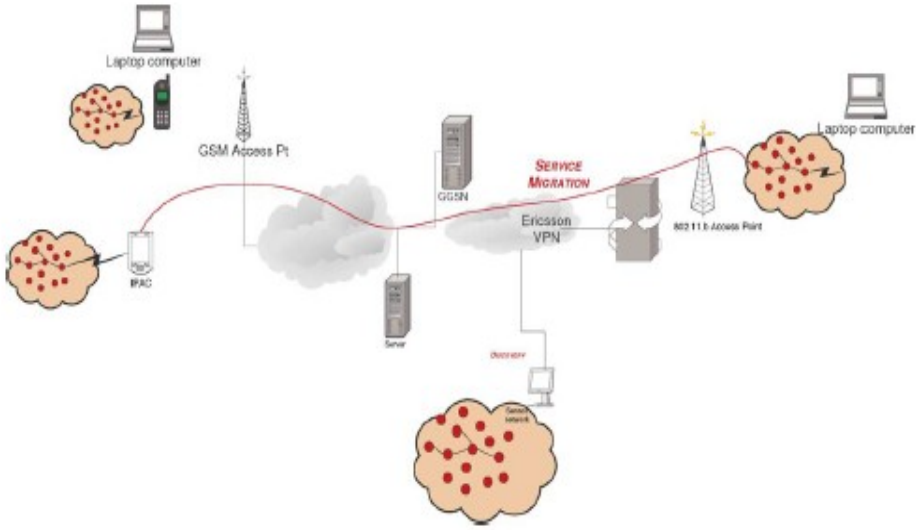
Υλοποιήθηκε μια σειρά αλγορίθμων για την αντιστοίχηση των διευθύνσεων των συνδρομητών σε JXTA διευθύνσεις. Αυτό δεν πρέπει να συγχέεται με τις GPRS διευθύνσεις, που αναφέρθηκαν νωρίτερα και αφορούν τον πυρήνα του δικτύου.

- Κατανομή λογισμικού

Έγιναν προσθήκες για να καταστεί εφικτή η απομακρυσμένη φόρτωση (remote loading) κλάσεων στις συσκευές με p2p τρόπο. Αυτό επιτρέπει τη δυναμική διαχείριση εφαρμογών δικτύων αισθητήρων χωρίς μια κεντρική αποθήκη κώδικα και καθιστά ευκολότερη τη μετακίνηση αισθητήρων από περιοχή σε περιοχή κτλ.

Εφαρμογή – Συμπεράσματα

Το σύστημα χρησιμοποιήθηκε σε μια απλή εφαρμογή παρακολούθησης περιβάλλοντος, δηλαδή με απλούς αισθητήρες που μετρούν θερμοκρασία, υγρασία και ιξώδες. Παρακάτω φαίνεται το σχετικό σχήμα. Τα laptops, τα IPAC και οι συσκευές κινητών τηλεφώνων αποτελούν σταθμούς-βάσεις ενώ τα άλλα μέρη έχουν λίγο-πολύ αναλυθεί. Στη συνέχεια παρατίθενται οι στόχοι που επιτεύχθηκαν.



ΕΙΚΟΝΑ 3.1 – Παράδειγμα δομής μιας εφαρμογής με το Skylark

- Δυναμική εισαγωγή νέων σταθμών-βάσεων

- Δυναμική απομάκρυνση σταθμών-βάσεων και αισθητήρων

- Δυναμική επαναρύθμιση των αισθητήρων

- Δυναμική ανάθεση αισθητήρων σε άλλους σταθμούς-βάσεις αν οι δικοί τους αποτύχουν

- Δυναμική μετακίνηση εφαρμογών μεταξύ των σταθμών-βάσεων

- Εφαρμογές όπου οι αισθητήρες συνεργάζονται (ανεξαρτήτως θέσης, τεχνολογίας κτλ.)

4. ΤΟ ΝΕΟ ΕΝΔΙΑΜΕΣΟ ΛΟΓΙΣΜΙΚΟ ShareSense

Σύνοψη και σύγκριση υπαρχόντων overlays

Στα πλαίσια της εργασίας υλοποιήθηκε λογισμικό για χρήση σε συστήματα δικτύων αισθητήρων. Το μεγαλύτερο μέρος αυτού του λογισμικού μπορεί να χρησιμοποιηθεί σαν overlay για τις υψηλότερου επιπέδου λειτουργίες ενός συστήματος δικτύων αισθητήρων, ενώ υλοποιήθηκε και λογισμικό επιπέδου εφαρμογής, με κατάλληλο γραφικό περιβάλλον, για να γίνουν οι απαραίτητες δοκιμές, πειράματα και επιδείξεις της λειτουργίας του. Πριν γίνει η υλοποίηση ενός τέτοιου συστήματος είναι απαραίτητο να μελετηθούν υπάρχοντα συστήματα που αντιμετωπίζουν παρόμοια ζητήματα. Τα κυριότερα από αυτά είναι αυτά που παρουσιάστηκαν στα δύο προηγούμενα κεφάλαια και σε αυτό το σημείο είναι χρήσιμο να γίνει μια ανασκόπηση και εξαγωγή συμπερασμάτων. Η ανασκόπηση θα γίνει με βάση τον τρόπο αντιμετώπισης ορισμένων καίριων ζητημάτων, όπως είναι η ετερογένεια των WSN, η διακοπτόμενη συνδεσιμότητα, η επεκτασιμότητα, η υποστήριξη διαδικτυακών εφαρμογών και η ενέργεια. Στην επόμενη ενότητα θα σχολιαστεί και το πού τοποθετείται το νέο λογισμικό ShareSense μέσα σε αυτό το «χάρτη» των overlays.

Η *ετερογένεια*, δηλαδή η ύπαρξη δικτύων με διαφορετικό πλήθος ή/και είδος κόμβων (δηλ. κόμβους που παράγουν μετρήσεις με διαφορετικούς τύπους δεδομένων κτλ.), αντιμετωπίζεται συνήθως με τη χρήση μετα-δεδομένων (meta-data). Έτσι, με μηνύματα τύπου XML, οι αισθητήρες αυτο-περιγράφονται και τα κέντρα ελέγχου (έτσι θα ονομάζουμε εδώ τα σημεία πρόσβασης ή τα sinks ή τους gateways των επιμέρους WSN) ξέρουν αφενός πώς να επικοινωνήσουν με αυτούς και αφετέρου μπορούν να έχουν το δικό τους σχήμα για να είναι δυνατή η μεταξύ τους επικοινωνία και ανταλλαγή δεδομένων. Τέτοια σχήματα χρησιμοποιούνται στο CarTel μεταξύ κέντρων ελέγχου και αισθητήρων ενώ στο Skylark χρησιμοποιούνται για την επικοινωνία μεταξύ κέντρων ελέγχου. Το Skylark απαιτεί, ούτως ή άλλως, ενσωματωμένους αισθητήρες, περιοριζόμενο έτσι να μην υποστηρίζει δίκτυα πολλών κόμβων με multihop-routing κτλ. Το jWebDust περιορίζει τους κόμβους στη χρήση συγκεκριμένου λειτουργικού και συγκεκριμένων μηνυμάτων, για να αποφύγει τη χρήση της XML, η οποία απαιτεί μεγάλα σε μέγεθος μηνύματα. Έτσι υποστηρίζει ετερογένεια μεταξύ αισθητήρων αλλά υπό προϋποθέσεις, ενώ η ετερογένεια στα υψηλότερα επίπεδα του συστήματος αντιμετωπίζεται χάρη στο συγκεκριμένο σχήμα της βάσης δεδομένων. Το MetroSense εκ των πραγμάτων προορίζεται για δίκτυα που είναι ετερογενή όσον αφορά το μέγεθος και την κίνηση, δεν δίνει όμως λύση στην ετερογένεια της τεχνολογίας των αισθητήρων και απαιτεί να υλοποιούν κοινά πρωτόκολλα. Το Hourglass δεν πραγματεύεται την ετερογένεια κόμβων άλλα υποστηρίζει ετερογένεια στα υψηλότερα επίπεδα, αφού ένας peer μπορεί να αποτελείται από διάφορα είδη και πλήθη συσκευών.

Η *διακοπτόμενη συνδεσιμότητα* αφορά συνήθως κινούμενα κέντρα ελέγχου, που ανά πάσα στιγμή μπορεί να μην έχουν σύνδεση μεταξύ τους ή με άλλα μέρη του

συστήματος. Η καλύτερη αντιμετώπιση είναι η προσωρινή αποθήκευση (buffering) για όσο δεν υπάρχει σύνδεση και η αυτόματη προώθηση όταν αυτή αποκατασταθεί. Τα πρωτόκολλα των CarTel και Hourglass υλοποιούν αυτήν την τεχνική προσφέροντας ικανοποιητικά αποτελέσματα, ενώ το Skylark αφήνει το υπόστρωμα της jxta να χειριστεί το ζήτημα, δίνοντας έτσι μια από τις πιο αξιόπιστες λύσεις. Το jWebDust δεν προβλέπει την άμεση επικοινωνία του καταναλωτή δεδομένων με τα κέντρα ελέγχου, αλλά μεσολαβεί το επίπεδο δεδομένων. Έτσι, δεν τίθεται θέμα αποτυχίας στην επικοινωνία, αλλά ένα query για δεδομένα που δεν έχουν καταγραφεί ακόμα λόγω έλλειψης συνδεσιμότητας δεν είναι σχεδιασμένο ώστε να περιμένει την αποκατάσταση της σύνδεσης, οπότε η έλλειψη συνδεσιμότητας δεν αντιμετωπίζεται άμεσα. Το MetroSense δεν αναφέρεται σε συγκεκριμένες τεχνικές για τα επίπεδα από το κέντρο ελέγχου και πάνω.

Η *επεκτασιμότητα* αντιμετωπίζεται εκ των πραγμάτων καλύτερα από τα peer-to-peer συστήματα. Έτσι, το Skylark και το Hourglass πλεονεκτούν. Παρόλ' αυτά, και τα υπόλοιπα συστήματα αξιοποιούν τη δομή του διαδικτύου για να είναι επεκτάσιμα. Η χρήση κάθε κεντρικοποιημένης δομής είναι μειονέκτημα (π.χ. βάσεις δεδομένων στο jWebDust), ενώ από την άλλη τίθεται και θέμα επεκτασιμότητας όσον αφορά τα ίδια τα επιμέρους WSN (το πλήθος, δηλαδή, των κόμβων που είναι σχεδιασμένα να υποστηρίξουν), όπου το jWebDust πλεονεκτεί, με το Hourglass και το MetroSense να έχουν επίσης θεωρητικά τη δυνατότητα χρήσης WSN ευρείας κάλυψης.

Οι χρηστικές *διαδικτυακές εφαρμογές* αποτελούν κοινό στόχο όλων αυτών των συστημάτων, από τη στιγμή που τα συστήματα έχουν περάσει από τη μελέτη των επιμέρους WSN στη μελέτη της διαδικτυακής διασύνδεσής τους. Τα συστήματα που είναι προσανατολισμένα να διευκολύνουν περισσότερο τη δημιουργία τέτοιων εφαρμογών, με χρηστικότερα και υψηλότερου επιπέδου γραφικά περιβάλλοντα, είναι το CarTel και το jWebDust. Τα υπόλοιπα φυσικά υποστηρίζουν τη δημιουργία τους αλλά δεν κάνουν κάποιο βήμα για τη διευκόλυνση της υλοποίησης, με εξαίρεση το Skylark, που αφορά το δίκτυο κινητής τηλεφωνίας και έχει το πλεονέκτημα του ήδη υπάρχοντος σχετικού λογισμικού, αφού υπάρχουν διάφορα λογισμικά που να διευκολύνουν την υλοποίηση εφαρμογών με χρηστικά GUI για κινητά τηλέφωνα.

Το πρόβλημα της *ενέργειας* αντιμετωπίζεται βέλτιστα αν οι κόμβοι μπορούν να φορτίζονται ή να ανανεώνεται η πηγή ενέργειάς τους με ευκολία. Αυτό είναι εφικτό μόνο σε συστήματα λίγων κόμβων. Όλα τα overlays έχουν σχεδιαστεί λαμβάνοντας υπόψη ότι και με τέτοια δίκτυα λίγων κόμβων μπορούν να γίνουν εντυπωσιακές εφαρμογές, αλλά αυτό που προηγείται είναι το Skylark, μια και αναμένεται να προσφέρει ευρύτατη κάλυψη με όλους τους αισθητήρες να βρίσκονται σε ένα κινητό (και συνεπώς να φορτίζονται εύκολα). Ακολουθεί το MetroSense, και μάλιστα χωρίς την ανάγκη ύπαρξης λίγων κόμβων, που επικεντρώνεται σε άνθρωπο-κεντρικά WSN, πράγμα που σημαίνει εύκολη ανανέωση της ενέργειας, με προσωπική και άμεση ευθύνη του χρήστη. Το CarTel είναι επίσης σχεδιασμένο για μικρά δίκτυα. Σε συστήματα όπου η τεχνική της φόρτισης/ανανέωσης δεν είναι εφικτή, ο στόχος είναι η ελάχιστη δυνατή επικοινωνία των κόμβων, ώστε να προκύπτει η ελάχιστη κατανάλωση. Προς αυτήν την κατεύθυνση, το jWebDust έχει επιλέξει τη χρήση μικρών μηνυμάτων (αποφυγή XML) ενώ όσα συστήματα υποστηρίζουν την οδηγούμενη από γεγονότα (event-based) επικοινωνία, μπορούν να αποφύγουν σε ορισμένες περιπτώσεις την άσκοπη περιοδική επικοινωνία. Ακολουθούν πίνακες που συνοψίζουν τη σύγκριση.

	ετερογενεια	ενεργεια
CarTel	+ XML-based adapters μεταξύ κέντρου ελέγχου και αισθητήρων	+ Κατά κανόνα οι κόμβοι είναι εντός οχημάτων (εύκολη φόρτιση)
MetroSense	+ Πρωτόκολλο για συνεργασία πολλών αισθητήρων, στατικών ή κινητών - Απαίτηση για παρόμοια τεχν/γία αισθητήρων (Υλοποίηση κοινού πρωτοκόλλου επικοινωνίας)	+ Προοπτική για δυνατότητα ανανέωσης της πηγής ενέργειας, μια και οι αισθητήρες είναι άμεσα συνδεδεμένοι με χρήστες - Συχνή και έντονη επικοινωνία μεταξύ κόμβων για την υλοποίηση της μεταβίβασης ευθύνης (opportunistic delegation)
jWebDust	+ Χρήση ειδικών μηνυμάτων για αυτό-περιγραφή των αισθητήρων - Απαίτηση για συγκεκριμένο λειτουργικό (TinyOS) και μηνύματα αυστηρού format	+ Αξιοποίηση δυνατότητας λειτουργικού για προσαρμογή συχνοτήτων εκπομπής των motes + Sleep modes και event-based sensing - Δυσκολία φόρτισης των κόμβων στις εφαρμογές για τις οποίες προορίζεται
Hourglass	+ Ένας peer μπορεί να αποτελείται από οσεσδήποτε και οποιεσδήποτε συσκευές - Καμία αντιμετώπιση της ετερογένειας των κόμβων των WSN	- Δεν πραγματεύεται καθόλου το ζήτημα
Skylark	+ XML-based μηνύματα μεταξύ κέντρων ελέγχου για ανταλλαγή δεδομένων - Απαίτηση για συγκεκριμένα είδη αισθητήρων, ενσωματωμένων στα κινητά τηλέφωνα.	+ Κόμβοι ενσωματωμένοι στο κινητό (πολύ εύκολη φόρτιση)

ΠΙΝΑΚΑΣ 4.1 – Σύγκριση των overlays σε ζητήματα που σχετίζονται περισσότερο με το επίπεδο αισθητήρων

	διακοπτομενη συνδεση	επεκτασιμοτητα	διαδικτυακεσ εφαρμογεσ
CarTel	+ Προσωρινή αποθήκευση χάρη στη στοίβα CafNet	- Βασίζεται στα Wi-Fi Access Points των αστικών περιοχών και απαιτεί διαθέσιμα οχήματα	+ Χρήση GoogleMaps και άλλων υψηλού επιπέδου GUI
MetroSense	- Δεν πραγματεύεται το ζήτημα	+ Υλοποίηση επέκτασης στο Open-WRT Linux για χρήση των Wi-Fi Access Points ως κέντρα ελέγχου των WSN, διευκολύνοντας τη διαδικτύωση	- Δεν πραγματεύεται τη μορφή που θα έχουν για το χρήστη οι διαδικτυακές εφαρμογές του
jWebDust	+ Παρεμβολή του επιπέδου δεδομένων μεταξύ εφαρμογής και κέντρου ελέγχου - Όχι αυτόματη ενημέρωση της εφαρμογής (αλλά μόνο της βάσης δεδομένων) με την αποκατάσταση της σύνδεσης	- Χρήση κεντρικοποιημένων δομών, όπως η βάση δεδομένων jWebDust	+ Χρήση Java Servlets για τη δυνατότητα εύκολης και εντελώς ανεξάρτητης από το υπόλοιπο σύστημα κατασκευής GUI υψηλού επιπέδου
Hourglass	+ Πειραματικά αποδεδειγμένη αντιμετώπιση χάρη στο buffering service που υλοποιούν οι κρίσιμοι peers	+ Οποιαδήποτε οντότητα υλοποιήσει τη βασική λειτουργικότητα μπορεί να εισαχθεί στο σύστημα ως ένας νέος peer, χωρίς καμία επιβάρυνση του συστήματος. Αντίθετα, το αναβαθμίζει.	- Δεν πραγματεύεται τη μορφή που θα έχουν για το χρήστη οι διαδικτυακές εφαρμογές του
Skylark	+ Χρήση της JXTA, που έχει πολλές δομές για αξιόπιστη επικοινωνία	+ Αξιοποιεί την τεράστια δυνατότητα επεκτασιμότητας του δικτύου κινητής τηλεφωνίας	+ Δυνατότητα αξιοποίησης υπάρχοντος λογισμικού για χρηστικές εφαρμογές κινητών τηλεφώνων

ΠΙΝΑΚΑΣ 4.2 – Σύγκριση των overlays σε ζητήματα που σχετίζονται περισσότερο με τη διαδικτύωση

Τι είναι το ShareSense

Όπως και τα άλλα overlays μεταξύ τους, έτσι και το νέο λογισμικό που υλοποιήθηκε έχει σε πολλά σημεία διαφορετικό προσανατολισμό και διαφορετικό στόχο από τα υπόλοιπα.

Το ShareSense δεν παρέχει υπόστρωμα για την επικοινωνία μεταξύ κέντρων ελέγχου και αισθητήρων, αλλά υποθέτει την ύπαρξη κάποιων άλλων πρωτοκόλλων για αυτή τη λειτουργία. Για παράδειγμα, θα μπορούσε να χρησιμοποιεί τα επίπεδα ελέγχου και αισθητήρων του jWebDust ή οποιαδήποτε άλλη τεχνική. Στη συνέχεια θα γίνει πιο ξεκάθαρο το σημείο στο οποίο το ShareSense δίνει τα σκήπτρα σε λειτουργίες χαμηλότερου επιπέδου.

Αυτό που επιτυγχάνει είναι: η ενοποίηση οσωνδήποτε επιμέρους WSN (δηλ. η ενοποίηση των κέντρων ελέγχου τους) σε ένα peer-to-peer σύστημα μαζί με όλους τους χρήστες/πελάτες (κάθε κέντρο ελέγχου είναι ένας peer και κάθε πελάτης είναι, επίσης, ένας peer) και η παροχή μιας προγραμματιστικής διεπαφής (API) για να δέχεται από το χρήστη queries που αποκρύπτουν τις λεπτομέρειες του δικτύου και να τα αναλύει σε συγκεκριμένα queries που απευθύνονται στα κατάλληλα κέντρα ελέγχου.

Απαιτεί από τα κέντρα ελέγχου να διαθέτουν μια βασισμένη σε XML περιγραφή του WSN που «ελέγχουν». Έτσι οι χρήστες αποκτούν μια (κρυμμένη από το χρήστη) γνώση της δομής του δικτύου και αποφεύγεται η περιττή επικοινωνία. Αποστέλλονται, δηλαδή, στα κέντρα ελέγχου μόνο ερωτήματα που μπορούν να απαντηθούν από αυτά, χωρίς να χρειάζεται κάποια κεντρική δομή, όπως εξυπηρετητής ή βάση δεδομένων.

Το ShareSense είναι το πιο γενικού σκοπού, ευέλικτο και εύκολο σε τροποποιήσεις λογισμικό διασύνδεσης WSN και παρέχει από τις αποδοτικότερες λύσεις όσον αφορά την αντιμετώπιση κίνησης / διακοπτόμενης σύνδεσης και επεκτασιμότητας του δικτύου. Χάρη στην πλατφόρμα υλοποίησης JXTA, μπορεί επίσης να παρέχει ασφάλεια, έλεγχο πρόσβασης και ταυτοποίηση, αν η εφαρμογή το απαιτεί, μια και όπως είναι γνωστό, τα δεδομένα αισθητήρων είναι συχνά είτε προσωπικής φύσεως είτε απόρρητα. Αν και δεν εξασφαλίζει το ίδιο την υποστήριξη ετερογενών κόμβων αισθητήρων (δεν ασχολείται με αυτό το επίπεδο), δεν εμποδίζει κιόλας την αντιμετώπιση του προβλήματος, αφού οι ενθυλακωμένες λειτουργίες του μπορούν να συνδυαστούν με οποιαδήποτε τεχνική για την υποστήριξη ετερογενών WSN.

Επιστρέφοντας, λοιπόν, στα κριτήρια της προηγούμενης ενότητας, το ShareSense μπορεί, χάρη στις βασισμένες σε XML περιγραφές, να διασυνδέσει ετερογενή (όσον αφορά τους τύπους δεδομένων κτλ.) δίκτυα, δεδομένης της ύπαρξης πρωτοκόλλου που εξασφαλίζει την επικοινωνία μεταξύ κέντρου ελέγχου και αισθητήρων. Αφήνει ανοιχτό το ενδεχόμενο χρήσης είτε περιοδικών είτε οδηγούμενων από γεγονότα μετρήσεων, δίνοντας ευελιξία στη διαχείριση της ενέργειας. Προχωρώντας στα προβλήματα του επιπέδου διασύνδεσης, που είναι και αυτά που στην ουσία πραγματεύεται και αντιμετωπίζει, πετυχαίνει τα εξής:

- Αντιμετώπιση διακοπτόμενης σύνδεσης από το επίπεδο επικοινωνίας της JXTA
- Απόλυτη επεκτασιμότητα με ελάχιστο κόστος, όπως κάθε peer-to-peer δομή.
- Παροχή Java API υψηλού επιπέδου, ό, τι πιο ευέλικτο για την υλοποίηση χρηστικών διαδικτυακών εφαρμογών.

Πριν την ανάλυση του συστήματος, περιγράφεται η πλατφόρμα JXTA, στην οποία βασίστηκε η υλοποίηση.

Η πλατφόρμα JXTA

4.1.1 **Η JXTA γενικά**

Η λέξη jxta δεν είναι αρκτικόλεξο. Είναι συντόμευση της αγγλικής λέξης «juxtapose» και σημαίνει «τοποθετώ δίπλα». Το project JXTA ξεκίνησε από μια ομάδα της εταιρίας Sun Microsystems υπό τη διεύθυνση των Bill Joy και Mike Clary. Στόχος ήταν αρχικά η δημιουργία μιας γλώσσας για τη συγγραφή peer-to-peer εφαρμογών που σύντομα εξελίχθηκε σε στόχο για τον καθορισμό προδιαγραφών και standards που θα διευκόλυναν την ανάπτυξη τέτοιων εφαρμογών. Η JXTA πέρασε το 2001 στα χέρια της κοινότητας των προγραμματιστών και αποτελεί πλέον ελεύθερο λογισμικό με ανοικτό κώδικα.

Ο δημιουργός δε χρειάζεται πλέον να απασχολείται με την υλοποίηση της επικοινωνίας των μερών του συστήματός του, αφού αυτό το αναλαμβάνει η jxta. Στον πυρήνα βρίσκεται μια συλλογή πρωτοκόλλων που καθορίζει έναν συγκεκριμένο τρόπο peer-to-peer επικοινωνίας. Επειδή τα μηνύματα που ανταλλάσσονται είναι καθορισμένης μορφής και βασίζονται αποκλειστικά σε XML σχήματα, προκύπτουν πολλά πλεονεκτήματα: Η επικοινωνία στη JXTA δεν εξαρτάται από τη χρησιμοποιούμενη γλώσσα προγραμματισμού και έτσι ο προγραμματισμός της εφαρμογής μπορεί να γίνει με οποιαδήποτε γλώσσα ή και με πολλές διαφορετικές, αρκεί να χρησιμοποιείται κάθε φορά η κατάλληλη υλοποίηση της πλατφόρμας (JXTA Platform Reference Implementation). Αυτή τη στιγμή υπάρχουν σταθερές εκδόσεις (stable releases) σε Java και C, ενώ υπό κατασκευή είναι αρκετές ακόμα (π.χ. σε Perl). Κάθε έκδοση παρέχει την αντίστοιχη προγραμματιστική διεπαφή (API), αλλά οι λειτουργίες που παρέχονται είναι πάντα οι ίδιες και τα API έχουν πολλές ομοιότητες. Ετερογένεια, λοιπόν, υποστηρίζεται και ως προς το υλικό και ως προς το λογισμικό, αφού ένας peer μπορεί να στεγάζεται σε οποιαδήποτε συσκευή (π.χ. PC, PDA κτλ., αρκεί να έχει εγκατασταθεί κάποια υλοποίηση του πυρήνα της JXTA) και να είναι προγραμματισμένος σε οποιαδήποτε γλώσσα (αρκεί να έχει εγκατασταθεί η αντίστοιχη υλοποίηση). Η υλοποίηση που χρησιμοποιείται στη συντριπτική πλειοψηφία των περιπτώσεων είναι η JXTA J2SE, δηλαδή η υλοποίηση σε Java.

Η JXTA αποτελείται από έξι πρωτόκολλα, τα κείμενα των οποίων μπορούν να βρεθούν μέσα από την επίσημη ιστοσελίδα της κοινότητας jxta [27]. Εκεί μπορούν να βρεθούν επίσης οι πλατφόρμες (και ο πηγαίος κώδικας) όλων των υλοποιήσεων, όπως επίσης και πολύτιμοι οδηγοί, βοηθήματα, demos καθώς και οι λίστες αλληλογραφίας (mailing lists), των οποίων η σημασία είναι καταλυτική (βλ. Πρόλογο). Στις επόμενες δύο υποενότητες περιγράφονται πολύ συνοπτικά έννοιες και πρωτόκολλα, τα οποία όπως ξεκαθαρίστηκε αποτελούν προδιαγραφές για κάθε υλοποίηση, ενώ στην υποενότητα 4.3.4 γίνεται μια πρώτη αναφορά στα πολύ βασικά σημεία του JXTA J2SE API [28], που αφορούν πολύ βασικές λειτουργίες, κοινές σχεδόν για όλες τις εφαρμογές. Το πώς όλα αυτά αξιοποιούνται στην πράξη θα φανεί στον κώδικα του ShareSense. Εκτός από τα παραπάνω, οι κυριότερες πηγές που χρησιμοποιήθηκαν κατά το σχεδιασμό και τον προγραμματισμό του ShareSense ήταν το [29], που αποτελεί το πιο ολοκληρωμένο (και

διαθέσιμο online) βιβλίο για τη JXTA, το [30], που είναι ο επίσημος προγραμματιστικός οδηγός, καθώς και τα [31], [32] ως γενικές μελέτες.

4.1.2 **Βασικές έννοιες της JXTA**

Κόμβος (Peer)

Ένα δίκτυο βασισμένο στη JXTA αποτελείται από ένα σύνολο κόμβων που αντιπροσωπεύονται από μια IP διεύθυνση και ένα port και ονομάζονται Peers. Η διεύθυνση αυτή αντιστοιχίζεται σε μια αφαιρετική έννοια της JXTA που ονομάζεται Peer Endpoint. Έτσι όλοι οι κόμβοι είναι, όπως λέμε στα peer-to-peer συστήματα, ομότιμοι και δεν έχει σημασία σε τι συσκευή «στεγάζονται», ενώ σε μια συσκευή μπορεί να βρίσκονται και πολλοί κόμβοι. Κάθε Peer λειτουργεί ανεξάρτητα από τους άλλους, χαρακτηρίζεται από ένα μοναδικό ID, το PeerID, και μπορεί, χάρη στα Endpoints, να επικοινωνεί με άλλους κόμβους είτε απευθείας είτε, όπως θα δούμε, μέσω άλλων Peers. Κάθε Peer, όταν εκτελείται, δημιουργεί μια τοπική μνήμη (την cache, by default σε ένα φάκελο με το όνομα cm) και εκεί αποθηκεύει όλα όσα χρειάζεται να αποθηκεύσει (τα Advertisements, όπως θα δούμε). Η JXTA δεν δημιουργεί «καθαρά» peer-to-peer συστήματα, με την έννοια ότι όσον αφορά τη λειτουργικότητά τους δεν είναι ακριβώς ομότιμοι, αλλά χωρίζονται στις εξής κατηγορίες:
- Edge peers: Αυτοί είναι οι κόμβοι που αποτελούν την πλειοψηφία και υλοποιούν τη λειτουργικότητα της εφαρμογής. Το όνομά τους μαρτυρά ότι βρίσκονται στην «άκρη» του δικτύου, με την έννοια ότι δεν χρησιμοποιούνται για να διασυνδέουν άλλους κόμβους. Χωρίζονται περαιτέρω σε simple peers και minimal peers, ανάλογα με το αν εκτός από το να ανταλλάσουν μηνύματα μπορούν και να αποθηκεύουν και να χειρίζονται πληροφορίες για το υπόλοιπο δίκτυο. Οι minimal peers δε μπορούν να το κάνουν αυτό γιατί βρίσκονται σε συσκευές περιορισμένων πόρων (βλ. την έκδοση της JXTA για μικροσυσκευές, JXTA Micro Edition [33]).
- Rendezvous peers: Η δουλειά αυτών των κόμβων είναι η προώθηση μηνυμάτων αναζήτησης των υπολοίπων κόμβων. Προωθούν τα μηνύματα σε όσους Edge peers αλλά και σε όσους άλλους Rendezvous peers γνωρίζουν. Κάθε peer μπορεί να λειτουργήσει ως τέτοιος.
- Relay peers: Αυτοί περιέχουν πληροφορίες για τη διαδρομή από την οποία μπορεί να γίνει η επικοινωνία με κάποιον άλλο peer και χρησιμοποιούν τις πληροφορίες αυτές για την προώθηση μηνυμάτων. Χαρακτηριστικότερο παράδειγμα είναι η χρήση τους για να συνδέουν με το υπόλοιπο δίκτυο τους peers που βρίσκονται πίσω από κάποιο NAT ή Firewall.

Ομάδα κόμβων (Peer Group)

Οι peers ενός δικτύου JXTA οργανώνονται σε ομάδες, τα Peer Groups, τα οποία επίσης χαρακτηρίζονται από μοναδικά ID, τα PeerGroupID, και χρησιμοποιούνται για να ομαδοποιήσουν τους κόμβους είτε ανάλογα με τα «ενδιαφέροντα» και τη λειτουργικότητά τους (αφού κάθε Group μπορεί να παρέχει διαφορετικές υπηρεσίες), είτε ανάλογα με την ασφάλεια και την ταυτοποίηση που απαιτείται σε κάθε Group. Αρχικά, κάθε κόμβος ανήκει στο Net Peer Group (το Group που περιέχει όλους τους

κόμβους ενός δικτύου JXTA) και στη συνέχεια μπορεί να εισχωρήσει (join) σε οσαδήποτε Groups.

Μήνυμα (Message)

Το μήνυμα είναι η ελάχιστη «ενότητα» που μπορεί να αποσταλεί ή να παραληφθεί από έναν κόμβο. Αποτελείται από ένα σύνολο ζευγών, τα στοιχεία του μηνύματος (Message Elements). Κάθε τέτοιο στοιχείο έχει ένα όνομα και το αντίστοιχο περιεχόμενο. Τα πρωτόκολλα της JXTA προβλέπουν στοιχεία που το περιεχόμενό τους είναι είτε τύπου XML είτε δυαδικά δεδομένα (binary data). Κάθε υπηρεσία (service) χρησιμοποιεί αυτά που χρειάζεται, αρκεί στην περίπτωση των δυαδικών δεδομένων να χρησιμοποιείται η κωδικοποίηση Base64. Τα μηνύματα μεταδίδονται με τη χρήση των pipes ή των sockets.

Advertisement

Το Advertisement είναι μια δομή μετα-δεδομένων, δηλαδή μια περιγραφή τύπου XML, που χρησιμοποιείται από τη JXTA για να αναπαραστήσει τους πόρους ενός δικτύου, για παράδειγμα Peers, Peer Groups, Pipes, Υπηρεσίες κτλ. Τα πρωτόκολλα της JXTA χρησιμοποιούν τα Advertisements για να περιγράψουν τους πόρους και να τους «δημοσιεύσουν» στο δίκτυο. Τα Advertisements έχουν συγκεκριμένο χρόνο ζωής ώστε να μη «διαφημίζουν» μη υπαρκτούς πόρους και για να έχουν ισχύ αφότου λήξει ο χρόνος ζωής τους, πρέπει να ξαναδημοσιεύονται. Υπάρχουν πολλά διαφορετικά είδη Advertisements αλλά οι θεμελιώδεις τύποι είναι τρεις. Ο πρώτος είναι τα PeerAdvertisements (περιέχουν το όνομα, το ID, το Endpoint, μια περιγραφή κτλ. ενός Peer), ο δεύτερος είναι τα PeerGroupAdvertisements (περιέχει αντίστοιχες πληροφορίες για ένα PeerGroup), ενώ ο τρίτος τύπος περιλαμβάνει όλα τα υπόλοιπα, συνηθέστερα των οποίων είναι τα PipeAdvertisements αλλά και τα custom Advertisements, αυτά, δηλαδή, που κατασκευάζει και χρησιμοποιεί ο προγραμματιστής για τις ανάγκες των υπηρεσιών της εφαρμογής του.

Υπηρεσία (Service)

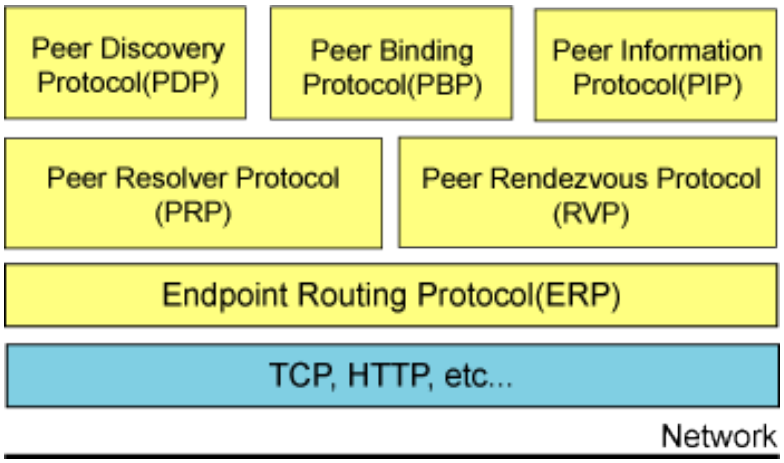
Για να επιτελούν τις επιθυμητές λειτουργίες μέσα στο δίκτυο, οι Peers χρησιμοποιούν τις υπηρεσίες. Για να μπορεί ένας Peer να προσφέρει ή να χρησιμοποιήσει μια υπηρεσία πρέπει υλοποιεί την υπηρεσία. Οι υπηρεσίες χωρίζονται στις υπηρεσίες πυρήνα (core services), αυτές δηλαδή που υπάρχουν στον πυρήνα της JXTA (και άρα κάθε peer μπορεί –προαιρετικά- να τις υλοποιήσει/χρησιμοποιήσει) και στις υπηρεσίες χρήστη, που υλοποιούνται με βάση τις ανάγκες της εφαρμογής. Και οι μεν και οι δε είναι άρρηκτα συνδεδεμένες με την έννοια του Group. Μια υπηρεσία, δηλαδή, ενός Group δε συνεργάζεται με την ίδια υπηρεσία ενός άλλου Group. Για παράδειγμα, αν ένας Peer θέλει να αναζητήσει πόρους μέσα σε ένα Group χρησιμοποιώντας την υπηρεσία αναζήτησης πόρων (Discovery Service), πρέπει πρώτα να «φορτώσει» το Discovery Service του συγκεκριμένου Group. Οι κυριότερες υπηρεσίες πυρήνα είναι οι παρακάτω:

- Rendezvous Service: Υλοποιεί την επικοινωνία με τους rendezvous peers ενώ μια κύρια υπηρεσία που παρέχει το API της είναι ο έλεγχος για το αν υπάρχει σύνδεση με τους επιθυμητούς rendezvous peers.
- Discovery Service: Χρησιμοποιείται για την ανεύρεση νέων πόρων μέσα στο Group. Πιο συγκεκριμένα, για την εύρεση των σχετικών Advertisements.
- Resolver Service: Χρησιμοποιείται για την αποστολή ερωτημάτων ή αιτημάτων. Είναι πολύ γενική υπηρεσία και σε αυτήν βασίζονται και άλλες υπηρεσίες. Μπορεί να χρησιμοποιηθεί για την επικοινωνία των Peers, εναλλακτικά του Pipe service ή παράλληλα με αυτό.
- Pipe Service: Αυτή η υπηρεσία είναι υπεύθυνη για τη δημιουργία συνδέσεων για την αποστολή μηνυμάτων μεταξύ των μελών ενός Group.
- Membership Service: Χρησιμοποιείται από τα μέλη ενός Group για την ταυτοποίηση νέων μελών, τον έλεγχο πρόσβασης σε υπηρεσίες και την ασφάλεια.

Υπάρχουν και άλλες υπηρεσίες (π.χ. η Endpoint Service) και στην επόμενη υποενότητα θα φανεί πόσο στενά συνδεδεμένες είναι οι υπηρεσίες πυρήνα με τα έξι πρωτόκολλα της JXTA.

4.1.3 **Τα πρωτόκολλα της JXTA**

Η JXTA ορίζει συνολικά έξι πρωτόκολλα, που το καθένα είναι ανεξάρτητο από τα υπόλοιπα και συνεπώς μπορεί να επανεξεταστεί ή να τροποποιηθεί αυτόνομα. Τα πρωτόκολλα αυτά σχεδιάστηκαν για να εξασφαλίσουν τη διαλειτουργικότητα όσων peer-to-peer εφαρμογών έχουν σχεδιαστεί με βάση αυτά και οι γνώμονες κατά το σχεδιασμό τους ήταν να θέτουν τις ελάχιστες δυνατές απαιτήσεις που πρέπει να πληρούνται, να δημιουργούν την ελάχιστη δυνατή επιβάρυνση αλλά και να κάνουν όσο το δυνατόν λιγότερες υποθέσεις για το χρησιμοποιούμενο σύστημα και τη δομή των δικτύων. Οι peers χρησιμοποιούν αυτά τα πρωτόκολλα για να επικοινωνούν μεταξύ τους, για να μοιράζονται τους πόρους τους και για να οργανώνονται, χωρίς να απαιτείται γνώση των πολύπλοκων δικτυακών δομών που τους συνδέουν. Ακολουθεί απεικόνιση των πρωτοκόλλων και στη συνέχεια συνοπτική περιγραφή τους. Τα δύο βασικότερα, που πρέπει να υλοποιούνται από κάθε peer, είναι το Peer Endpoint Protocol και το Peer Resolver Protocol, τα οποία καθιστούν δυνατή τη διευθυνσιοδότηση του peer, δηλαδή τον καθιστούν, όπως λέμε, addressable μέσα στο peer-to-peer δίκτυο. Τα υπόλοιπα πρωτόκολλα, αν και όπως είπαμε σχεδιάζονται ανεξάρτητα, βασίζονται σε αυτά τα δύο με την έννοια ότι χρησιμοποιούν λειτουργίες που καθορίζονται από αυτά.



ΣΧΗΜΑ 4.1 – Τα έξι πρωτόκολλα της JXTA

Endpoint Routing Protocol (ERP)

Το πρωτόκολλο αυτό χρησιμοποιείται για την αποστολή ενός μηνύματος από έναν peer σε έναν άλλο, ακόμα και όταν η μεταξύ τους σύνδεση δεν είναι εφικτή π.χ. επειδή εμποδίζεται από κάποιο firewall. Σε αυτήν την περίπτωση χρησιμοποιούνται ενδιάμεσοι κόμβοι (relay peers) για να προωθήσουν το μήνυμα στον επιθυμητό προορισμό και ακόμα κι αν η τοπολογία του δικτύου αλλάξει, το πρωτόκολλο είναι υπεύθυνο για να βρει τον τρόπο να διεκπεραιώνεται η επικοινωνία μεταξύ δύο οποιωνδήποτε peers. Όταν ένας peer πρόκειται να στείλει ένα μήνυμα, αναζητά πρώτα τις πληροφορίες δρομολόγησης στην τοπική του μνήμη. Αν δεν τις βρει, τότε τις ζητά από όσους relay peers γνωρίζει και αυτοί του απαντούν, του στέλνουν, δηλαδή, μια λίστα από τα router endpoints μέσω των οποίων θα επικοινωνήσει με τον peer αποδέκτη. Το μήνυμα περνάει από όλους αυτούς τους δρομολογητές (routers) και καθένας από αυτούς προσθέτει επιπλέον πληροφορίες που χρησιμοποιούνται για την αποφυγή πολλαπλής αποστολής του ίδιου μηνύματος. Με τη βοήθεια αυτού του πρωτοκόλλου, λοιπόν, καθίσταται η δρομολόγηση ανεξάρτητη από τις φυσικές θέσεις των peers μέσα στο δίκτυο και αποκρύβεται η τοπολογία.

Peer Resolver Protocol (PRP)

Το πρωτόκολλο αυτό παρέχει το μηχανισμό με τον οποίο ένας peer μπορεί να στείλει ένα μήνυμα σε έναν ή περισσότερους peers. Ένα προς αποστολή μήνυμα σχετίζεται με μια διαδικασία αποστολής αλλά όχι με μια συγκεκριμένη διεύθυνση ενός peer. Έτσι, όταν στέλνεται ένα μήνυμα χρησιμοποιώντας τις τεχνικές που ορίζει αυτό το πρωτόκολλο, το μήνυμα μπορεί να παραληφθεί από πολλούς, ή και όλους, τους peers ενός peer group. Όμως, μόνο οι peers που σχετίζονται με την αντίστοιχη διαδικασία θα επεξεργαστούν το μήνυμα και θα στείλουν, πιθανώς, κάποια απάντηση. Το ID του peer που στέλνει το αρχικό μήνυμα περιέχεται και στο μήνυμα (που για το συγκεκριμένο πρωτόκολλο ονομάζεται ReolverQueryMsg) και έτσι οι παραλήπτες του μπορούν να απαντήσουν κατευθείαν σε αυτόν. Αυτού του είδους η ερωταπάντηση που βασίζεται στο Resolver Service χρησιμοποιείται συχνά, τόσο από τους προγραμματιστές για τις ανάγκες μιας εφαρμογής όσο και από τις υπόλοιπες υπηρεσίες του πυρήνα για να παρέχουν υπηρεσίες υψηλότερου επιπέδου (π.χ. κάποιες λειτουργίες του Discovery

Service βασίζονται στο Resolver Service). Προαιρετικά, μπορεί να υλοποιείται και multicasting, δηλαδή η αποστολή ενός μηνύματος προς όλους τους peers που βρίσκονται εντός του τοπικού δικτύου του αποστολέα.

Peer Discovery Protocol (PDP)

Αυτό το πρωτόκολλο χρησιμοποιείται για τη «δημοσίευση» των πόρων ενός peer στο δίκτυο και για την ανακάλυψη των πόρων που διατίθενται από άλλους peers. Αυτό γίνεται πάντα στα πλαίσια ενός group. Η περιγραφή οποιουδήποτε πόρου γίνεται με τα Advertisements, τα οποία έχουν μια καθορισμένη μορφή μετα-δεδομένων και έτσι είναι ανεξάρτητα γλώσσας προγραμματισμού και πλατφόρμας. Δεν υπάρχει εγγύηση ότι κάθε μήνυμα αναζήτησης πόρων θα απαντηθεί (αυτό γίνεται μόνο αν βρεθούν σχετικοί πόροι), ενώ μπορεί και να παραληφθούν και πολλά μηνύματα που περιέχουν τα ίδια Advertisements. Επίσης, ένας peer μπορεί ανά πάσα στιγμή να παραλάβει και μήνυμα με Advertisements χωρίς να έχει πρώτα επιτελέσει τη σχετική αναζήτηση. Αυτό χρειάζεται ώστε να είναι δυνατή η δημοσίευση πόρων με πρωτοβουλία του peer που τους διαθέτει. Η υλοποίηση του πρωτοκόλλου αυτού από τον πυρήνα της JXTA αναβαθμίζεται από «εξυπνότερους» μηχανισμούς, χάρη στην υπερφόρτωση σχετικών μεθόδων που παρέχονται από το API.

Pipe Binding Protocol (PBP)

Εκτός από την τεχνική που παρέχεται από το Resolver Protocol, και το Pipe Binding Protocol παρέχει έναν μηχανισμό για την επικοινωνία μεταξύ δύο ή περισσότερων peers. Βασίζεται στη δημιουργία ενός εικονικού καναλιού μεταξύ των endpoints. Αυτά τα κανάλια είναι μονής κατεύθυνσης (unidirectional) και η μετάδοση των δεομένων γίνεται ασύγχρονα. Το άκρο του αποστολέα ονομάζεται output pipe και το άκρο του παραλήπτη ονομάζεται input pipe. Ένα κανάλι μεταξύ δύο peers ονομάζεται point-to-point ενώ ένα κανάλι με περισσότερους από έναν παραλήπτες ονομάζεται propagate. Ένα pipe περιγράφεται και αυτό από ένα Advertisement, το οποίο περιέχει ένα μοναδικό ID αλλά όχι και τη διεύθυνση κάποιου peer, αφού το κανάλι επικοινωνίας είναι ανεξάρτητο από το δίκτυο και τη φυσική του δομή. Ένας peer που επιθυμεί να δημιουργήσει κανάλι επικοινωνίας με έναν άλλο απλά δημοσιεύει το σχετικό Pipe Advertisement και ο δεύτερος το ανακαλύπτει χρησιμοποιώντας το Discovery Protocol.

Peer Rendezvous Protocol (PRP)

Αυτό το πρωτόκολλο είναι υπεύθυνο για τη διάδοση μηνυμάτων από έναν rendezvous peer προς όλους τους peers που γνωρίζει. Όσοι υλοποιούν αυτό το πρωτόκολλο λέγονται και rendezvous peers και δεν το αναφέραμε ως βασικό πρωτόκολλο μαζί με τα Endpoint και Resolver γιατί δεν είναι υποχρεωτικό να υλοποιείται από κάθε peer. Παρόλ' αυτά, έχει θεμελιώδη λειτουργικότητα και το Resolver Protocol βασίζεται και σε αυτό για να αποστείλει μηνύματα με πολλαπλούς αποδέκτες. Επίσης, το PRP είναι υπεύθυνο για τον έλεγχο του χρόνου ζωής των Advertisements.

Peer Information Protocol (PIP)

Αυτό το πρωτόκολλο προβλέπει τον τρόπο με τον οποίο οι peers μπορούν να ερωτώνται για την κατάσταση λειτουργίας τους (status), π.χ. την «κίνηση» μηνυμάτων στον συγκεκριμένο peer, τις χρονικές στιγμές αποστολής/παραλαβής μηνυμάτων κτλ. Πρόκειται, όμως, για ένα πρωτόκολλο πολύ λίγο ανεπτυγμένο και ελάχιστα υλοποιημένο από τις υπάρχουσες υλοποιήσεις.

4.1.4 **Ο προγραμματισμός με την υλοποίηση JXTA για J2SE**

Η γλώσσα προγραμματισμού που χρησιμοποιείται στις περισσότερες εφαρμογές που βασίζονται στη JXTA είναι η Java. Έτσι, η έκδοση της JXTA για την Java 2 Standard Edition είναι με τεράστια διαφορά η πιο διαδεδομένη. Για τον προγραμματισμό με αυτήν απαιτείται η σχετική πλατφόρμα, η οποία είναι διαθέσιμη μέσω του [27]. Επίσης διαθέσιμος είναι και ο πηγαίος κώδικας της πλατφόρμας για τον χρήστη που επιθυμεί είτε να τον μελετήσει για καλύτερη κατανόηση είτε να τον τροποποιήσει για τις ανάγκες της εφαρμογής του είτε για να συμβάλλει στην εξέλιξή του από την κοινότητα, μια και πρόκειται για λογισμικό ανοικτού κώδικα. Το API που αυτή παρέχει είναι επίσημα τεκμηριωμένο στο [28] και είναι και αυτό που χρησιμοποιήθηκε για την ανάπτυξη του ShareSense και τη χρήση αυτού θα σχολιάσουμε σε αυτήν την υποενότητα.

Το *πρώτο* πράγμα που πρέπει να κάνει ένας peer είναι να ρυθμίσει τις δικτυακές παραμέτρους του, δηλαδή το port και το πρωτόκολλο επικοινωνίας που θα χρησιμοποιήσει (TCP/HTTP), τις διευθύνσεις των peers που θα χρησιμοποιεί ως relays/rendezvous, το αν θα χρησιμοποιεί multicasting κτλ. Για αυτό το σκοπό υπάρχει η κλάση NetworkConfigurator που παρέχει τις σχετικές μεθόδους (setPeerID, setName, setTcpPort, addRdvSeedingURI, addRelaySeedingURI κτλ.) ενώ, για αρχάριους χρήστες, υπάρχει ένα γραφικό περιβάλλον ενός αυτόματου network configurator που «εκτοξεύεται» αν δε ρυθμίζονται προγραμματιστικά οι παράμετροι. Η διαδικασία αυτή οδηγεί στη δημιουργία του αρχείου PlatformConfig, που αν υπάρχει κατά την εκκίνηση του peer δε χρειάζεται να ξαναδημιουργηθεί, αλλά οι παράμετροι διαβάζονται από εκεί.

Το *δεύτερο* βήμα στον προγραμματισμό ενός peer είναι η εκκίνηση της πλατφόρμας JXTA. Αυτή γίνεται με τη δημιουργία του καθολικού group (σ.σ. του NetPeerGroup) με τη βοήθεια της κλάσης NetPeerGroupFactory. Αυτό εκκινεί αυτόματα τη λειτουργία της πλατφόρμας και καθιστά τον peer μέλος του Net Peer Group δικτύου. Σε αυτό το σημείο, πριν προχωρήσει στην λειτουργία του, ο peer συνηθίζει να ελέγχει αν έχει σύνδεση με κάποιον rendezvous peer (που συχνά είναι και relay peer), με τη βοήθεια της μεθόδου isConnectedToRendezVous της διεπαφής (interface) RendezvousService.

Το *τρίτο* μέρος του προγραμματισμού ενός peer είναι η υλοποίηση της λειτουργικότητάς του. Παρότι αυτό είναι καθαρά εξαρτώμενο από την εφαρμογή, υπάρχουν κάποια πράγματα που είναι κοινά (ή παρόμοια) σχεδόν σε κάθε εφαρμογή της

JXTA. Αυτά θα τα δούμε στην υλοποίηση του ShareSense αλλά ας κάνουμε μια αναφορά στις πολύ βασικές ενέργειες που κάνουν σχεδόν όλοι οι peers:
- Δημιουργία Group (*net.jxta.peergroup.PeerGroup.newGroup()*)
- Εισχώρηση σε Group (*net.jxta.membership.MembershipService.join()*)
- Ανακάλυψη πόρων (υλοποίηση του *DiscoveryListener*, υπερφόρτωση της μεθόδου *net.jxta.discovery.DiscoveryListener.discoverEvent()* και κλήση της μεθόδου *net.jxta.discovery.DiscoveryService.getRemoteAdvertisements()*)
- χρήση των pipes και του resolver για επικοινωνία (χρησιμοποιώντας τα interfaces και τις σχετικές μεθόδους που βρίσκονται στα *net.jxta.pipe* και *net.jxta.resolver* αντίστοιχα)
- και άλλα πολλά…

Το Σύστημα ShareSense

4.1.5 Δομή και συνολική λειτουργικότητα

Το ενδιάμεσο λογισμικό που αναπτύχθηκε δημιουργεί peer-to-peer δίκτυα που διασυνδέουν δίκτυα αισθητήρων και τους χρήστες αυτών. Τα δίκτυα αυτά, που βασίζονται στη JXTA, αποτελούνται από τρεις τύπους peers:
- Οι gateway peers είναι αυτοί που εκτελούνται στα κέντρα ελέγχου των επιμέρους WSN. Τα κέντρα ελέγχου συνδέουν ένα WSN με άλλα WSN καθώς και με το υπόλοιπο δίκτυο. Γνωρίζουν τους αισθητήρες που ελέγχουν και μπορούν να επικοινωνούν με αυτούς.
- Οι user peers είναι αυτοί που εκτελούνται στα μηχανήματα των χρηστών των εφαρμογών που βασίζονται στο ShareSense. Εκεί, δηλαδή, όπου εγκαθίσταται και η εκάστοτε εφαρμογή χρήστη. Για τις ανάγκες της εργασίας δημιουργήθηκαν και μια-δυο εφαρμογές με γραφικό περιβάλλον για queries στο δίκτυο αλλά το λογισμικό αυτών δεν πρέπει να συγχέεται με το στάνταρ λογισμικό του user peer.
- Οι routing peers εκτελούν τα καθήκοντα των rendezvous και relay peers της JXTA και παρακολουθούν τις συνδέσεις των υπολοίπων peers στο δίκτυο.

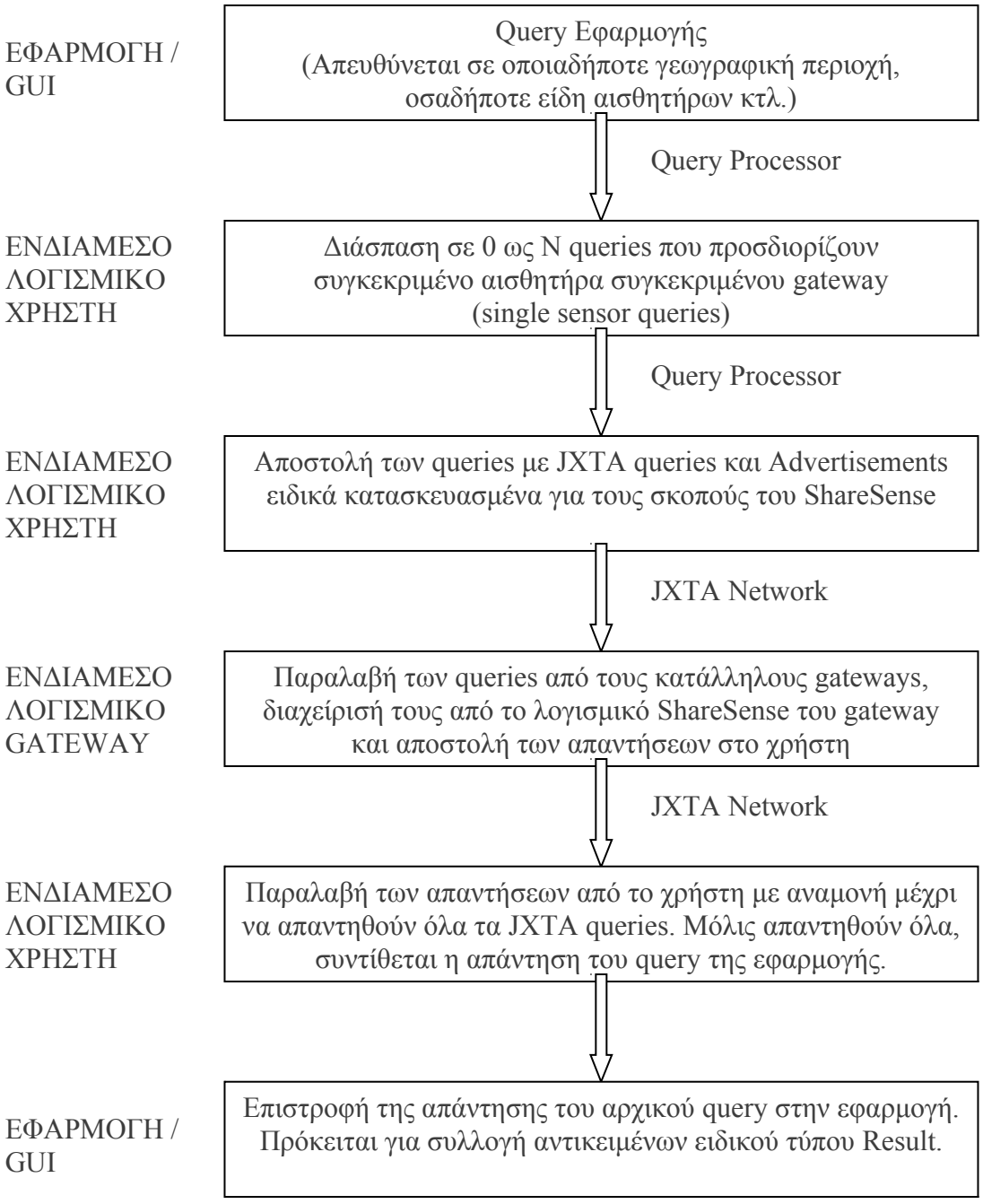
Το CD που συνοδεύει αυτήν την εργασία περιέχει το λογισμικό για καθένα από τα είδη των peers στους φακέλους ShareSenseGateway, ShareSenseUser και ShareSenseRdv αντίστοιχα. Οι routing peers πρέπει να εκτελούνται σε μηχανήματα με IP διεύθυνση που είναι γνωστή και προσβάσιμη από τα μηχανήματα όπου θα εκτελεστούν οι υπόλοιποι peers. Αυτή η διεύθυνση πρέπει να αποθηκεύεται στο αρχείο seeds.txt, που είναι διαθέσιμο σε κάθε έναν από τους peers. Επίσης, στο φάκελο ShareSenseUser περιέχεται και το λογισμικό της εφαρμογής χρήστη που υλοποιήθηκε. Οι δοκιμές έγιναν με τους gateways και τους users να εκτελούνται σε σταθμούς εργασίας (PC's και laptops) και τους routing peers να εκτελούνται είτε σε μηχάνημα στο ίδιο τοπικό δίκτυο με τους υπόλοιπους peers είτε απομακρυσμένα, σε εξυπηρετητή του Ινστιτούτου Τεχνολογίας Υπολογιστών (ΙΤΥ) με public IP. Είναι σαφές ότι όσο περισσότεροι γίνουν οι peers του δικτύου συνολικά, τόσο περισσότεροι routing peers θα χρειάζονται.

Πριν προχωρήσουμε στην περιγραφή του κώδικα και της λειτουργίας καθενός από τους peers, είναι απαραίτητο να εξηγηθεί από γενική σκοπιά το πώς αλληλεπιδρούν

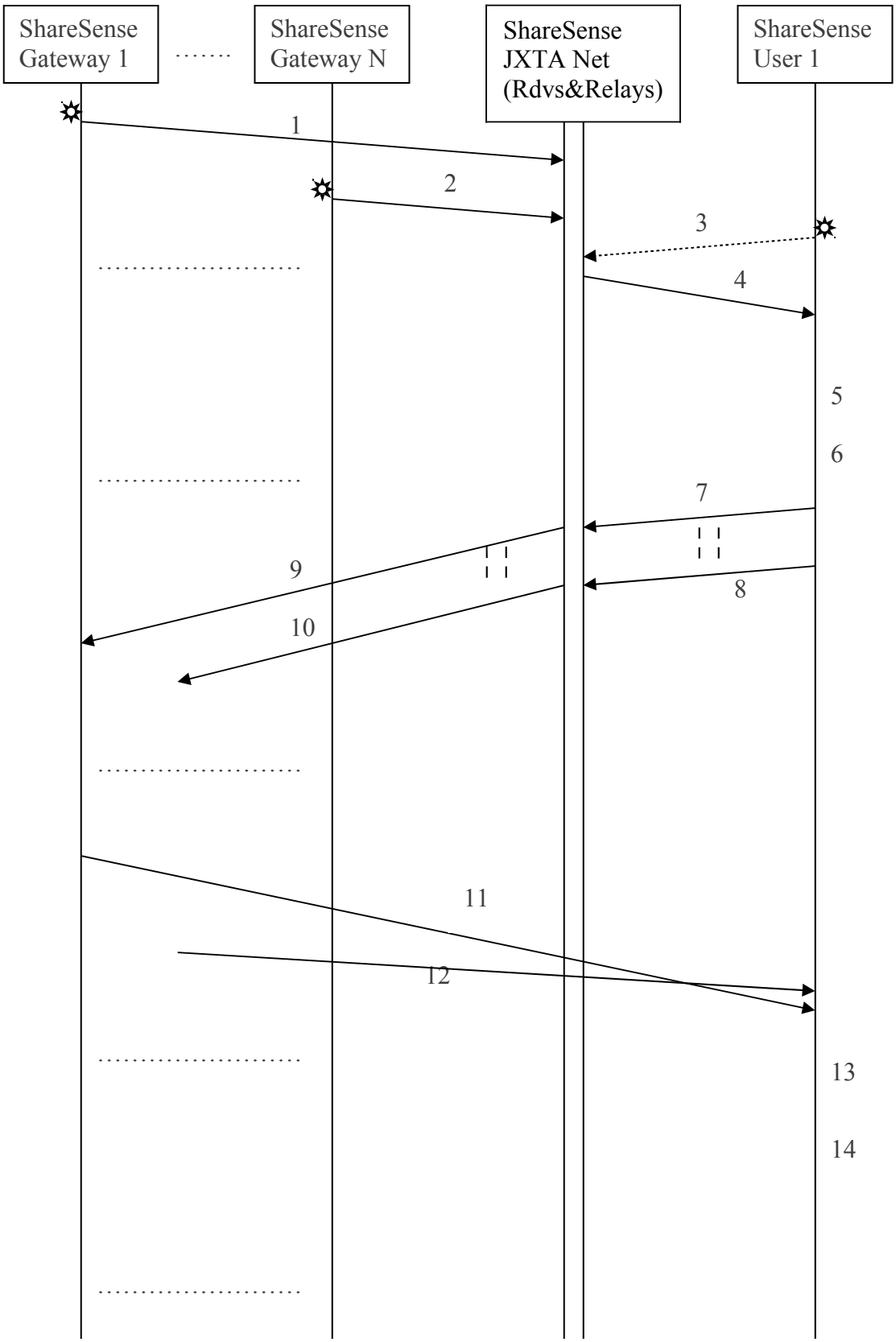
οι peers για την εκτέλεση των queries και να γίνει αυτή η διαδικασία πιο κατανοητή και με τη βοήθεια ορισμένων σχημάτων.

Όπως έχει αναφερθεί και κατά την ανάλυση άλλων overlays, στόχος είναι η εφαρμογή να μπορεί να βλέπει τα WSN που βρίσκονται «από κάτω» σαν ένα, ενιαίο δίκτυο αισθητήρων, στο οποίο μπορεί να απευθύνει queries, χωρίς να προσδιορίζει λεπτομέρειες που αφορούν τη δομή του δικτύου. Κάθε εφαρμογή θέτει στο δίκτυο τα δικά της queries, με τη μορφή που ορίζεται από την εφαρμογή και στις χρονικές στιγμές που το επιθυμεί ο χρήστης της εφαρμογής.

Έτσι και στις εφαρμογές που φτιάχνονται με το ShareSense, ο χρήστης της εφαρμογής δε χρειάζεται να γνωρίζει τη δομή του δικτύου. Κάθε query της εφαρμογής αναλύεται από το λογισμικό του ShareSense σε queries που είναι επαρκώς προσδιορισμένα ώστε να αποσταλούν στον κατάλληλο gateway. Αυτή η ανάλυση γίνεται από το τμήμα του ShareSense που βρίσκεται στη μεριά του χρήστη. Το πώς θα το δούμε στην περιγραφή του ShareSenseUser. Εδώ πρέπει να σημειωθεί ότι αυτό είναι εφικτό χάρη στη γνώση για τη δομή του δικτύου που αποκτά ο User κατά τη σύνδεσή του στο δίκτυο, οπότε και εντοπίζει τα σχετικά Advertisements με χρήση του Discovery Service. Μια εφαρμογή περιμένει μέχρι να απαντηθεί το query της, δηλαδή μέχρι να απαντηθεί κάθε ένα από τα εξειδικευμένα queries στα οποία έχει αναλυθεί. Η διαδικασία αυτή γίνεται πιο κατανοητή με τα σχήματα που ακολουθούν. Το πρώτο δείχνει τον κύκλο ζωής ενός query μιας εφαρμογής, ενώ το επόμενο επεκτείνεται λίγο περισσότερο για να κάνει την όλη διαδικασία κατανοητή μέσα στο πλαίσιο του χώρου και του χρόνου.



ΣΧΗΜΑ 4.2 – Κύκλος ζωής των queries στο ShareSense



(εξηγήσεις αριθμών και λεζάντα στην επόμενη σελίδα)

Λειτουργίες και μηνύματα του σχήματος:

(Το μικρό σχήμα που μοιάζει με ήλιο δείχνει το σημείο εκκίνησης λειτουργίας)

1. Δημοσίευση του Advertisement με την περιγραφή του WSN που ελέγχει ο gateway 1
2. Δημοσίευση του Advertisement με την περιγραφή του WSN που ελέγχει ο gateway N (τέτοιο Advertisement στέλνουν, λοιπόν, όλοι οι gateways μόλις συνδεθούν στο δίκτυο)
3. Αναζήτηση πόρων του δικτύου (δηλ. των περιγραφών των WSN) από το χρήστη
4. Μήνυμα(-τα) με όλα τα Advertisements με τις περιγραφές των gateways
5. Δημιουργία ενός query της εφαρμογής
6. Διάσπαση (split) του query της εφαρμογής σε m single-sensor queries
7. Αποστολή του μηνύματος του πρώτου single-sensor query (προς το δίκτυο, αλλά έχοντας προσδιορίσει το ID του gateway peer παραλήπτη)
8. Αποστολή του μηνύματος του m-οστού single-sensor query (προς το δίκτυο, αλλά έχοντας προσδιορίσει το ID του gateway peer παραλήπτη). Οι διακεκομμένες γραμμές δείχνουν ότι έχει μεσολαβήσει η αποστολή όλων των single-sensor queries.
9. Προώθηση του μηνύματος του 7 από τον rendezvous/relay peer στον κατάλληλο αποδέκτη (στο παράδειγμα του σχήματος αποδέκτης είναι ο gateway 1)
10. Προώθηση του μηνύματος του 8 από τον rendezvous/relay peer στον κατάλληλο αποδέκτη (στο παράδειγμα του σχήματος αποδέκτης είναι ένας οποιοσδήποτε gateway)
11, 12. Αποστολή των απαντήσεων πίσω στο χρήστη. Βλέπουμε ότι η αποστολή των απαντήσεων γίνεται πλέον με απευθείας σύνδεση των δύο peers, αφού ο gateway, εκτός από το ID του user peer, έχει «μάθει» από τους rendezvous/relays και το πώς να επικοινωνήσει με αυτόν (βλ. και εισαγωγή του κεφαλαίου 5 του [29]). Όπως δείχνει και το σχήμα, η αποστολή αυτών των απαντήσεων γίνεται, φυσικά, ασύγχρονα ενώ μεσολαβούν και οι απαντήσεις για τα υπόλοιπα queries, που δεν έχουν σημειωθεί. Οι απαντήσεις αυτές μπορούν να αποτελούνται και από πολλά μηνύματα, τα οποία το ενδιάμεσο λογισμικό χειρίζεται κατάλληλα.
13. Εντοπισμός του γεγονότος ότι όλα τα queries έχουν απαντηθεί και σύνθεση της απάντησης του αρχικού query της εφαρμογής.
14. Επιστροφή της απάντησης στην εφαρμογή χρήστη.

ΣΧΗΜΑ 4.3 – Αναπαράσταση του κύκλου ζωής των queries του ShareSense στο χώρο και στο χρόνο.

Οι παραπάνω λειτουργίες παρουσιάζονται από γενική σκοπιά και σε πολλά σημεία γίνονται γενικεύσεις (π.χ. η αποστολή των queries μπορεί να γίνει με διάφορα είδη μηνυμάτων και υπηρεσιών ανάλογα με την περίσταση, η παραλαβή και σύνθεση των αποτελεσμάτων είναι πιο περίπλοκη από όσο φαίνεται κτλ.). Επίσης, ο κύκλος ζωής που φαίνεται στο σχήμα μπορεί να είναι διαφορετικός (π.χ. το ενδιάμεσο λογισμικό χρήστη δεν ανιχνεύει πόρους μόνο κατά την εκκίνησή του, αλλά παρέχει και μεθόδους για να γίνει αυτό σε οποιαδήποτε στιγμή επιθυμεί η εφαρμογή), ενώ φυσικά οι χρήστες είναι πολλοί και χρησιμοποιούν το δίκτυο ταυτόχρονα. Βλέπουμε ότι ο τρόπος με τον οποίο στέλνονται τα queries παρέχει τη δυνατότητα για αντιμετώπιση του προβλήματος της διακοπτόμενης συνδεσιμότητας. Περισσότερα θα γίνουν κατανοητά στη συνέχεια.

4.1.6 **Λογισμικό Διασυνδετικών Κόμβων (ShareSenseRdv)**

Ανάλογα με την έκταση του δικτύου, πρέπει να υπάρχουν ένας ή περισσότεροι κόμβοι που εκτελούν αυτό το λογισμικό. Πρόκειται για κόμβους που ρυθμίζονται προγραμματιστικά ώστε να λειτουργούν σαν rendezvous και relays (ταυτόχρονα) στο JXTA δίκτυο που δημιουργείται για τις ανάγκες του ShareSense. Το πώς αυτοί χειρίζονται και προωθούν τα μηνύματα, υλοποιείται από την πλατφόρμα της JXTA. Το μόνο που χρειάζεται κατά τον προγραμματισμό τους είναι να δηλωθεί ότι λειτουργούν σαν τέτοιοι κόμβοι και να προγραμματιστούν και κάποιες βασικές λειτουργίες. Στην περίπτωσή μας απλά εκκινούν την πλατφόρμα, γίνονται μέλη του group CeidSN (στο οποίο γίνονται μέλη όλοι οι κόμβοι του δικτύου), και μετά απλώς αναμένουν συνδέσεις, εκτυπώνοντας σχετικά μηνύματα όταν τις εντοπίζουν και εκτελώντας μια περιοδική αναζήτηση για Advertisements εντός του group.

Το λογισμικό είναι απλό, βρίσκεται σε ένα μόνο αρχείο πηγαίου κώδικα (Rendezvous.java) και, εξαιρουμένων των απαραίτητων τροποποιήσεων για να λειτουργεί στο δικό μας σύστημα και με το δικό μας group, βασίζεται σε κώδικα που βρίσκεται ανηρτημένος και στο site της κοινότητας, αναφερόμενος ως private-net-demo. Το αρχείο seeds.txt πρέπει να περιέχει πάντα τη διεύθυνση του μηχανήματος στο οποίο εκτελείται το λογισμικό.

Επειδή οι βασικές λειτουργίες που γίνονται και οι μέθοδοι που χρησιμοποιούνται για αυτές υπάρχουν με σχεδόν πανομοιότυπη μορφή και στα ShareSenseGateway και ShareSenseUser, δε θα σχολιαστούν εδώ. Στα πειράματα χρησιμοποιήθηκε μόνο ένας peer που εκτελούσε αυτό το λογισμικό και βρισκόταν, όπως αναφέρθηκε, είτε σε τοπικό δίκτυο μαζί με όλους τους υπόλοιπους peers είτε στον em1server.cti.gr (150.140.5.5) του Ινστιτούτου Τεχνολογίας Υπολογιστών (ΙΤΥ).

Οι υπόλοιποι peers συνδέονται απευθείας σε κάποιους από αυτούς (προσθέτοντας τις διευθύνσεις αυτών στο δικό τους seeds.txt αρχείο) και μέσω αυτών συνδέονται στο υπόλοιπο δίκτυο. Για το πώς γίνεται αυτό, ο αναγνώστης παραπέμπεται στη σχετική βιβλιογραφία της JXTA ([29],[30] κτλ.). Η ανάμιξή τους δε θα μας απασχολήσει από εδώ και στο εξής και θα μιλάμε για -βασισμένη σε ID- επικοινωνία των υπόλοιπων peers μεταξύ τους, χωρίς να μας ενδιαφέρει πώς γίνεται αυτή και πόσοι ενδιάμεσοι κόμβοι μεσολαβούν. Αυτό το αναλαμβάνει η πλατφόρμα JXTA.

4.1.7 **Λογισμικό Κέντρων Ελέγχου (ShareSenseGateway)**

Οι peers που εκτελούνται σε μηχάνημα που αποτελεί κέντρο ελέγχου ενός WSN πρέπει να διαθέτουν αυτό το λογισμικό.

Υποχρεωτικά, κάθε τέτοιος peer διαθέτει ένα αρχείο που περιέχει την περιγραφή του αντίστοιχου WSN. Πρόκειται για το αρχείο PeerDescription.txt. Περιέχει μια περιγραφή βασισμένη σε XML που περιλαμβάνει όλες τις απαραίτητες πληροφορίες που χρειάζονται οι χρήστες ώστε να μπορούν, έχοντας συλλέξει τις σχετικές πληροφορίες από όλα τα κέντρα ελέγχου, να θέτουν queries και να χειρίζονται τα δεδομένα με το σωστό τρόπο. Για παράδειγμα, σε αυτές τις περιγραφές περιέχεται η γνώση για τις τοποθεσίες και τις δυνατότητες των αισθητήρων και αυτό βοηθά στην αποδοτική επικοινωνία, αφού οι χρήστες θέτουν μόνο queries που μπορούν να απαντηθούν, και μόνο σε αυτούς που μπορούν να τα απαντήσουν. Επίσης, περιέχονται πληροφορίες για τους τύπους δεδομένων που υποστηρίζει κάθε αισθητήρας, γεγονός που, σε συνδυασμό με το κατάλληλο ενδιάμεσο λογισμικό από την πλευρά του χρήστη, παρέχει πλήρη υποστήριξη της ετερογένειας. Φυσικά, το «σχήμα» και η σημασία των μετα-δεδομένων που περιέχονται στο PeerDescription.txt είναι καθορισμένη και έχει την παρακάτω μορφή (description format):

```
<peer>

	<type>Gateway</type>

	<capabilities>
		<capability>
			<id>……</id>
			<type>……</type>
			<dataType>……</dataType>
		</capability>
		<capability>
			……
		</capability>
		……
		……
	</capabilities>

	<sensors>
		<sensor>
			<id>……</id>
			<name>……</name>
			<accuracy>……</accuracy>
			<units>……</units>
			<longitude>……</longitude>
			<latitude>……</latitude>
			<capabilities>
				<id>……</id>
				<id>……</id>
			</capabilities>
		</sensor>
		<sensor>
			……
		</sensor>
		……
		……
	</sensors>

</peer>
```

Μπορούν, φυσικά, να προστεθούν οποιαδήποτε και οσαδήποτε tags πρώτης τάξεως (όπως είναι παραπάνω τα type, capabilities και sensors), με οποιοδήποτε περιεχόμενο. Στα συγκεκριμένα όμως tags πρέπει να διατηρείται η εμφώλευση που φαίνεται παραπάνω. Τα capabilities είναι ουσιαστικά τα φυσικά μεγέθη που μπορούν να μετρηθούν σε αυτό το WSN ενώ οι sensors αντιστοιχούν στους κόμβους του WSN. Βλέπουμε ότι ένας κόμβος μπορεί να έχει τη δυνατότητα να μετρά πολλά μεγέθη (σ.σ. πολλά capabilities για κάθε sensor), όπως ισχύει στα σύγχρονα motes. Πιο διαφωτιστικό είναι το παράδειγμα του PeerDescription.txt που βρίσκεται στο CD.

Η τακτική αυτή και οι περιγραφές αυτού του είδους καταργούν την ανάγκη για IDs των sensors που είναι μοναδικά σε ολόκληρο το δίκτυο. Αρκεί να είναι μοναδικά εντός του WSN που ελέγχεται από το συγκεκριμένο κέντρο ελέγχου. Όπως θα δούμε αργότερα, οι περιγραφές αυτές θα αποτελούν το τμήμα <Description> του PeerAdvertisement του gateway peer που εκτελείται στο κέντρο ελέγχου. Το PeerAdvertisement είναι αυτό που, περιλαμβάνοντας και άλλες πληροφορίες για τον peer, θα «δημοσιεύεται στο JXTA δίκτυο (βλ. βήματα 1 και 2 του σχήματος 4.2).

Ο κώδικας του gateway peer αποτελείται από 6 κλάσεις. Η κλάση Initiator υλοποιεί τη βασική λειτουργικότητα του peer, η κλάση ProviderListener ορίζει τον τρόπο λειτουργίας των pipes που θα χρησιμοποιεί ο gateway, οι κλάσεις PublicQueryMsg, PublicResponseMsg και PublicQueryHandler ορίζουν τον τρόπο λειτουργίας του ResolverService για την επικοινωνία με resolver queries ενώ, τέλος, η κλάση RequestAdvertisement είναι ένας custom τύπος Advertisement (σ.σ. υπάρχουν και οι core τύποι Advertisements που ορίζονται από τον πυρήνα της JXTA, όπως π.χ. το PeerAdvertisement ή το PeerGroupAdvertisement κτλ.) που κατασκευάστηκε για να «δημοσιεύει» στο δίκτυο την ύπαρξη των χρησιμοποιούμενων pipes (ο core τύπος PipeAdvertisement δεν ήταν κατάλληλος για το συγκεκριμένο σκοπό).

- Η κλάση **Initiator** υλοποιεί, όπως είπαμε τη βασική λειτουργικότητα του peer, που συνοπτικά είναι η εξής (σύγκρινε και με 4.3.4):

- Ρύθμιση παραμέτρων του δικτύου (μέθοδος configureJxta)
- Εκκίνηση της πλατφόρμας JXTA, με αναμονή μέχρι τον εντοπισμό σύνδεσης με κάποιον rendezvous peer (μέθοδος startJxta).
- Έλεγχος για το Group του (δια)δικτύου αισθητήρων του οποίου επιθυμεί να γίνει μέλος

  Αν το Group είναι ήδη γνωστό (σ.σ. υπάρχει στην cache του peer), τότε γίνεται μέλος του Group (μέθοδος joinGroup*). Αλλιώς πρώτα δημιουργεί το Group (μέθοδος createGroup) και μετά γίνεται μέλος του (μέθοδος joinGroup*).

  ΣΗΜΕΙΩΣΗ: Ο έλεγχος γίνεται με τη μέθοδο checkGroupCreated, που όμως δεν έχει νόημα αν η configureJxta καλεί την clearCache νωρίτερα, όπως και γίνεται στον παρών κώδικα.

* Η joinGroup περιλαμβάνει ανάγνωση της XML περιγραφής, ενσωμάτωσή της στο PeerAdvertisement και δημοσίευση του PeerAdvertisement στο Group.

- Καταχώρηση του PublicQueryHandler ως χειριστή των queries που θα γίνονται στο Resolver Service του Group για αυτόν (μέθοδος registerPublicQueryHandler).
- Ατέρμονη αναμονή με περιοδική αναζήτηση για Advertisements τύπου RequestAdvertisement (μέθοδος searchPipes). Ο εντοπισμός οποιουδήποτε τέτοιου Advertisement πυροδοτεί την κλήση της μεθόδου discoveryEvent, η οποία με τη σειρά της εκκινεί την εξυπηρέτηση της υπηρεσίας που ζητείται από το Advertisement, χρησιμοποιώντας τον ProviderListener.

Οι μέθοδοι που αναφέρθηκαν είναι μέθοδοι της κλάσης Initiator και ενθυλακώνουν κλήσεις βασικών μεθόδων του API της JXTA J2SE, όπως αυτές που αναφέρθηκαν στο 4.3.4, με τη σειρά, τη μορφή και τη λειτουργικότητα που απαιτείται από το ShareSense. Επίσης, υπάρχουν και κάποιες άλλες μέθοδοι για περαιτέρω ενθυλάκωση ή για βελτίωση των παρεχόμενων υπηρεσιών. Ήδη πρέπει να έχει γίνει σαφές ότι για την εξυπηρέτηση των queries, οι gateways χρησιμοποιούν δύο τρόπους. Είτε το Resolver Service είτε τα pipes. Η αναμονή για queries οποιουδήποτε από τους δύο τύπους είναι παράλληλη. Στην παρούσα υλοποίηση, τα pipes χρησιμοποιούνται για την εξυπηρέτηση των queries που αφορούν δυαδικά δεδομένα, ενώ το Resolver Service για τους υπόλοιπους τύπους δεδομένων. Οι διαφορές, τα πλεονεκτήματα και τα μειονεκτήματα θα φανούν στη συνέχεια. Υπενθυμίζεται ότι οι τρεις επόμενες κλάσεις που θα περιγραφούν (PublicQueryMsg, PublicQueryResponse, PublicQueryHandler) αφορούν τη χρήση του Resolver Service ενώ οι δύο τελευταίες (RequestAdvertisement, ProviderListener) αφορούν τη χρήση των pipes.

- Η κλάση **PublicQueryMsg** αντιπροσωπεύει το μήνυμα που στέλνεται όταν πρέπει να εξυπηρετηθεί ένα query από το Resolver Service του Group. Προφανώς, λοιπόν, τέτοια μηνύματα θα στέλνονται στο δίκτυο από το λογισμικό του χρήστη. Όμως και ένας gateway peer πρέπει να διαθέτει την κλάση, για να γνωρίζει τη δομή του μηνύματος και να μπορεί να το παραλαμβάνει και να το αναλύει.

Ένα τέτοιο μήνυμα χαρακτηρίζεται από πέντε πεδία:

- sensorId: Το id του κόμβου (mote) στον οποίο αναφέρεται το query.
- capabilityId: Το id του φυσικού μεγέθους (π.χ. θερμοκρασία κτλ.) στον οποίο αναφέρεται το query.
- isHistory: Το αν απαιτείται και ιστορικό μετρήσεων ή όχι.
- function: Η συνάρτηση που καθορίζει ποια μετρημένη τιμή του συγκεκριμένου μεγέθους ζητείται από το query (π.χ. η τελευταία, η μέγιστη, η μέση κτλ.).
- queryCount: Αριθμός που δείχνει τη σειρά του query εντός μιας ομάδας από queries (της ομάδας των queries στην οποία έχει διασπαστεί ένα query εφαρμογής).

Τα μηνύματα αυτά, όταν κυκλοφορούν στο δίκτυο, έχουν και αυτά τη μορφή μετα-δεδομένων. Έτσι, οι βασικές μέθοδοι αυτής της κλάσης είναι αυτές που μετατρέπουν τα μετα-δεδομένα σε αντικείμενα και το αντίστροφο. Ο constructor, λοιπόν, PublicQueryMessage(InputStream) αναλύει (parsing) τα μετα-δεδομένα του InputStream θέτοντας με βάση αυτά τις τιμές των πεδίων του μηνύματος. Αντίστροφα, η μέθοδος getDocument(MimeMediaType) παίρνει τις τιμές των πεδίων του μηνύματος (από τις αντίστοιχες μεταβλητές) και κατασκευάζει την κατάλληλη ακολουθία μετα-δεδομένων, για να σταλούν μέσα από το δίκτυο. Οι υπόλοιπες μέθοδοι επιστρέφουν τις τιμές των πεδίων ή ενθυλακώνουν λειτουργίες.

Ο τρόπος χρήσης των αντικειμένων PublicQueryMsg (όπως και των PublicResponseMsg) γίνεται πιο κατανοητός με τη μελέτη του PublicQueryHandler.

- Η κλάση **PublicResponseMsg** αντιπροσωπεύει το μήνυμα που στέλνεται ως απάντηση σε ένα PublicQueryMsg, και πάλι με χρήση του Resolver Service. Η λογική του constructor από InputStream και της μεθόδου getDocument είναι ίδια με πριν και το ίδιο ισχύει και για τις υπόλοιπες μεθόδους. Αυτό που διαφέρει είναι μόνο τα πεδία του μηνύματος. Η απάντηση, περιλαμβάνει, λοιπόν, έξι πεδία και είναι τα παρακάτω:

- sensorId: Το id του κόμβου (mote), όπως προηγουμένως.
- capabilityId: Το id του φυσικού μεγέθους, όπως προηγουμένως.
- value: Η τιμή της απάντησης.
- respondeName: Το όνομα του gateway peer που στέλνει την απάντηση.
- queryCount: Πρόκειται για την αντίστοιχη τιμή του query στο οποίο απαντά αυτό το μήνυμα. Έτσι, ο παραλήπτης ξέρει σε ποιο query αναφέρεται κάθε απάντηση που παραλαμβάνει, μια και οι απαντήσεις λαμβάνονται ασύγχρονα και συνεπώς πιθανότατα με διαφορετική σειρά από αυτήν που στάλθηκαν τα αντίστοιχα queries.
- fileVec: Δομή που χρησιμοποιείται για τη μεταφορά ιστορικού μετρήσεων.

- Η έννοια του handler στη java είναι λίγο-πολύ γνωστή. Η κλάση **PublicQueryHandler** υλοποιεί τις λειτουργίες που εκτελεί ένας peer κατά την παραλαβή ενός PublicQueryMsg ή ενός PublicResponseMsg. Υλοποιεί (implements) τη διεπαφή QueryHandler, που υπαγορεύει τη βασική λειτουργικότητα ενός handler του Resolver Service της JXTA. Όπως κάθε κλάση που υλοποιεί αυτή τη διεπαφή, η PublicQueryHandler υλοποιεί τις μεθόδους processQuery και processResponse. Η πρώτη καλείται όταν παραληφθεί ένα αίτημα (Query) και συνεπώς είναι αυτή που στο ShareSense χρησιμοποιείται από έναν gateway. Η δεύτερη καλείται όταν παραληφθεί μια απάντηση (Response) και άρα στο ShareSense καλείται από το χρήστη. Γι' αυτό η λειτουργία της θα γίνει πιο κατανοητή κατά την περιγραφή του λογισμικού του χρήστη.

Η processQuery χρησιμοποιεί τον constructor της PublicQueryMsg που αναφέρθηκε νωρίτερα για να δημιουργήσει από το ληφθέν μήνυμα μετα-δεδομένων ένα αντικείμενο. Με χρήση των ειδικών μεθόδων της PublicQueryMsg προσκομίζει τις τιμές των πεδίων του αντικειμένου και τις χρησιμοποιεί για να δημιουργήσει την απάντηση. Όταν ολοκληρώσει, επιστρέφει την απάντηση στον peer που είχε στείλει το query. Το σημείο στο οποίο κατασκευάζεται η απάντηση (μέθοδος findAttrValue) είναι πολύ σημαντικό. Πρόκειται για το σημείο στο οποίο το ShareSense πρέπει να συνδεθεί με το επίπεδο που θα βρίσκεται από κάτω του (π.χ. να διαβάζει από μια βάση δεδομένων συγκεκριμένου σχήματος, να καλεί μεθόδους ενός άλλου API ή οτιδήποτε παρόμοιο). Στην υλοποίηση, βλέπουμε να παράγονται τυχαίες τιμές. Οι readFile (και storeFile) χρησιμοποιούνται για να διαβάζουν από (και να γράφουν σε) αρχεία ιστορικού μετρήσεων. Η πρώτη χρησιμοποιείται στην processQuery (δηλαδή από τον gateway) ενώ η δεύτερη στην processResponse (δηλαδή από το χρήστη).

- Η κλάση **RequestAdvertisement** ορίζει Advertisements που προορίζονται να «διαφημίζουν» την ύπαρξη pipes που είναι ανοιχτά περιμένοντας να εξυπηρετήσουν κάποια queries.

Ενώ για να εξυπηρετήσει τα queries του Resolver Service ο gateway αρκεί να έχει καταχωρήσει (register) το σχετικό handler (και στη συνέχεια κάθε query που φτάνει υφίσταται επεξεργασίας αυτόματα), η κατάσταση για τα queries που εξυπηρετούνται από pipes είναι κάπως διαφορετική. Ο gateway εξυπηρετεί ένα pipe όταν μαθαίνει ότι ένα pipe είναι ανοιχτό (περιμένει εξυπηρέτηση) από τη μεριά κάποιου χρήστη και, φυσικά, μπορεί και να το εξυπηρετήσει (δηλ. είναι αυτός που έχει στο WSN του τους κατάλληλους αισθητήρες για να απαντήσει στα ζητούμενα). Η ύπαρξη των pipes μαθαίνεται με τον εντοπισμό συγκεκριμένων Advertisements που «δημοσιεύονται» στο δίκτυο από τους χρήστες όταν οι τελευταίοι θέλουν να εξυπηρετηθεί κάποιο pipe που έχουν ανοίξει. Οι gateways δεν παραλαμβάνουν αυτόματα αυτά τα Advertisements, όπως συνέβαινε με τα queries του Resolver Service, αλλά είναι αναγκασμένοι να τα αναζητούν

περιοδικά. Το γεγονός αυτό, όμως, δεν προκαλεί ιδιαίτερα μεγάλες καθυστερήσεις στον εντοπισμό τους, δεδομένου ότι οι gateways βρίσκονται ούτως ή άλλως σε αναμονή και αναζητήσεις με περίοδο της τάξης του δευτερολέπτου είναι λογικές.

Μια κλάση ενός τύπου Advertisement, όπως η RequestAdvertisement, μοιάζει λίγο με τις κλάσεις των μηνυμάτων του Resolver Service (PublicQueryMsg και PublicResponseMsg). Έχει, δηλαδή, μεθόδους που από XML Advertisements δημιουργούν αντικείμενα και αντιστρόφως. Επίσης έχει διάφορα πεδία και τις αντίστοιχες μεθόδους για να τα επιστρέφει. Η διαφορά είναι ότι τα Advertisements έχουν «ταυτότητα» μέσα στο δίκτυο. Θεωρούνται διαμοιραζόμενοι πόροι του δικτύου (που αποθηκεύονται και σε διάφορους rendezvous/relay peers) και ως εκ τούτου χαρακτηρίζονται από μοναδικά ID. Αυτό καθιστά απαραίτητη την ύπαρξη ενός μηχανισμού για τη «γέννηση» ενός νέου, και μοναδικού, ID για κάθε τέτοιο Advertisement που δημιουργείται (instantiation). Για αυτό το σκοπό υπάρχουν οι μέθοδοι setSeed και getID. Και οι δύο καλούνται μία φορά για κάθε νέο Advertisement, κατά τη «γένεσή» του. Η πρώτη εξασφαλίζει ότι το ID που θα ανατεθεί θα είναι τυχαίο, ενώ η δεύτερη το αναθέτει χρησιμοποιώντας το IDFactory της JXTA.

Ένα άλλο πολύ σημαντικό χαρακτηριστικό, που στην ουσία αποτελεί και το λόγο που το ShareSense χρησιμοποιεί τα RequestAdvertisements και όχι απλά τα -ήδη ορισμένα από την πλατφόρμα της JXTA- PipeAdvertisements, είναι τα «πεδία δεικτοδότησης» (index fields). Πρόκειται για τα πεδία με βάση τις τιμές των οποίων μπορεί η αναζήτηση να περιοριστεί. Δηλαδή, ο gateway, στην περιοδική αναζήτησή του για Advertisements (δηλαδή στην κλήση της getRemoteAdvertisements), αναζητά μόνο Advertisements που έχουν μια συγκεκριμένη τιμή (“Binary Query”) σε ένα συγκεκριμένο πεδίο (“name”). Αλλιώς θα του επιστρεφόταν και ένα πλήθος άλλων, «άχρηστων» Advertisements. Στο RequestAdvertisement, λοιπόν, τέτοια πεδία έχουν οριστεί να είναι τα πεδία “name”, “ID” και “pipeID”. Τοποθετούνται στον πίνακα indexFields και επιστρέφονται από την getIndexFields. Περισσότερα μπορούν να γίνουν κατανοητά με τη μελέτη του κώδικα τόσο του RequestAdvertisement όσο και της διεπαφής (interface) Advertisement του πυρήνα, την οποία η RequestAdvertisement υλοποιεί.

- Η κλάση **ProviderListener** υλοποιεί (implements) την OutputPipeListener. Οι μέθοδοι της ProviderListener καλούνται όταν ο peer εντοπίσει κάποιο ανοιχτό input pipe που μπορεί να το εξυπηρετήσει. Τότε δημιουργείται το αντίστοιχο output pipe (στη μέθοδο provide). Η δημιουργία ενός output pipe πυροδοτεί (triggers) αυτόματα την κλήση της outputPipeEvent, μεθόδου που πρέπει να υλοποιείται σε κάθε κλάση που υλοποιεί την OutputPipeListener. Στην ProviderListener, η outputPipeEvent υλοποιείται ώστε να στέλνει τα επιθυμητά μηνύματα και, αφότου το κάνει, να κλείσει το output pipe.

Τα μηνύματα που στέλνονται περιέχουν τα δυαδικά δεδομένα που έχουν ζητηθεί από το χρήστη (το τι έχει ζητηθεί μαρτυρείται από τις τιμές των πεδίων του RequestAdvertisement που είχε εντοπιστεί). Και σε αυτό το σημείο το ShareSense πρέπει να συνδεθεί με το κατώτερο επίπεδο ώστε να διαβάζει τα δεδομένα. Προς το παρόν, τα διαβάζει από αρχεία με σημασιολογικές ονομασίες. Επειδή το μέγιστο μέγεθος ενός μηνύματος που στέλνεται μέσω pipe είναι περιορισμένο, το ShareSense χρησιμοποιεί πακέτα μεγέθους PACKET_SIZE, ώστε η σταθερά PACKET_SIZE να μπορεί να τροποποιηθεί εύκολα. Γενικά, πρέπει να είναι μικρότερο των 16Kbyte. Η JXTA προβλέπει μηνύματα μέχρι και 64Kbyte αλλά συχνά προδίδουν οι buffers. Στην παρούσα υλοποίηση το PACKET_SIZE είναι 10Kbyte. Επειδή τα pipes δεν είναι αξιόπιστα,

χρησιμοποιείται συντονισμός και έλεγχος, με τη χρήση αριθμών ακολουθίας στα μηνύματα κτλ. (βλ. και σχολιασμό κώδικα). Εναλλακτικά του ζεύγους input/output pipe θα μπορούσαν να χρησιμοποιηθούν JxtaBiDiPipes (pipes διπλής κατεύθυνσης, με δυνατότητες για αξιόπιστη μεταφορά) ή JxtaSockets. Θα επέφεραν, όμως, αχρείαστη επιβάρυνση, ενώ στη JXTA τα sockets δεν έχουν αποδειχτεί και τόσο αξιόπιστα.

Σύγκριση των δύο μεθόδων εξυπηρέτησης query (Resolver Service vs Pipe Service)

Ανακεφαλαιώνοντας, και αφού έχουν αναλυθεί αρκετά οι δύο τεχνικές, υπενθυμίζεται ότι, στο ShareSense, το Pipe Service χρησιμοποιείται όταν τα δεδομένα που ζητούνται είναι δυαδικά ενώ το Resolver Service σε κάθε άλλη περίπτωση. Τα pipes προσφέρονται εκ των πραγμάτων περισσότερο για τη μεταφορά δυαδικών δεδομένων, αλλά ο βασικός λόγος για αυτήν την επιλογή είναι άλλος και έχει να κάνει με την πιθανότητα διακοπτόμενης συνδεσιμότητας.

Η λύση που έχει υλοποιηθεί με τη χρήση των pipes είναι απόλυτα αξιόπιστη και αποδοτική στην περίπτωση που ο gateway δεν έχει διαρκή και αξιόπιστη σύνδεση, πιθανόν γιατί είναι μια κινητή συσκευή. Τα Advertisements που δημοσιεύονται από τον χρήστη υπάρχουν στο δίκτυο μέχρι να λήξει ο χρόνος ζωής τους και έτσι ο gateway θα τα εντοπίσει και θα τα εξυπηρετήσει οποιαδήποτε στιγμή κι αν αποκατασταθεί η σύνδεσή του και όποτε και αν είχε διακοπεί. Αν, λοιπόν, ο μέγιστος χρόνος ζωής επιλεχθεί κατάλληλα, δε θα χάνεται κανένα query. Αυτή η ιδιότητα έχει μεγαλύτερη σημασία όταν πρόκειται για δυαδικά δεδομένα διότι είναι συνήθως μεγαλύτερα σε μέγεθος (σ.σ. ένα XML μήνυμα χωρίς δυαδικά δεδομένα ποτέ δε θα είναι της τάξης των πολλών Kbyte ή Mbyte). Αντίθετα, το πλεονέκτημα της χρήσης του Resolver Service είναι ότι εξυπηρετούνται άμεσα (χωρίς αναζήτηση κτλ.) και γρηγορότερα, καθώς και ότι μπορούν να απευθύνονται σε πολλούς peers (ακόμα και σε όλους τους peers ενός Group) ταυτόχρονα. Όμως, τέτοια queries μπορεί και να χαθούν. Γι' αυτό, όπως θα δούμε, τίθενται προθεσμίες (timeouts) και από την πλευρά του χρήστη.

4.1.8 **Λογισμικό Χρηστών (ShareSenseUser)**

Το λογισμικό που βρίσκεται από τη μεριά του χρήστη δίνει τη δυνατότητα υλοποίησης εφαρμογών που χρησιμοποιούν το δίκτυο JXTA που έχει περιγραφεί, για να εκτελούν queries. Οι εφαρμογές δεν ενδιαφέρονται για το πώς γίνεται η σύνδεση, ποια είναι η δομή του δικτύου και τι αισθητήρες υπάρχουν. Αυτά υλοποιούνται από το ενδιάμεσο λογισμικό χρήστη, το οποίο τελικά παρέχει στην εφαρμογή έναν query processor, δηλαδή μια κλάση που χρησιμοποιείται για να απαντά στα ερωτήματα της εφαρμογής χρησιμοποιώντας κατάλληλα το δίκτυο. Εν τέλει, δηλαδή, η εφαρμογή χρησιμοποιεί από το ενδιάμεσο λογισμικό ShareSenseUser ένα απλό API για να συνδεθεί στο δίκτυο και τον query processor.

Η λειτουργικότητα που απαιτείται για τη σύνδεση του user peer στο δίκτυο έχει πολλές ομοιότητες με τη σχετική λειτουργικότητα του gateway peer. Επίσης, όπως έχει ήδη αναφερθεί, ορισμένες κλάσεις που σχετίζονται με την επικοινωνία (είτε χρησιμοποιώντας το Resolver Service είτε το Pipe Service) είναι κοινές. Αυτά τα σημεία δε θα αναλυθούν ξανά και στη θέση τους θα υπάρχουν παραπομπές στην περιγραφή της προηγούμενης υποενότητας. Στο λογισμικό ShareSenseUser του συνοδευτικού CD εμπεριέχεται και το λογισμικό μιας εφαρμογής γραφικού περιβάλλοντος για queries. Πρόκειται για το SenseGUI.java και θα περιγραφεί στην ενότητα 4.5. Προς το παρόν θα αναφερθούμε στις υπόλοιπες κλάσεις. Είναι 11 κλάσεις και η βασική λειτουργικότητα του peer υλοποιείται στην κλάση Querier, με την οποία και θα ξεκινήσει η περιγραφή.

- Η κλάση **Querier** έχει, όσον αφορά βασικές λειτουργίες ενός JXTA peer, ομοιότητα με την κλάση Initiator του gateway peer. Συνοπτικά, έχει την εξής λειτουργικότητα:

- Ρύθμιση παραμέτρων του δικτύου (μέθοδος configureJxta).
- Εκκίνηση της πλατφόρμας JXTA, με αναμονή μέχρι τον εντοπισμό σύνδεσης με κάποιον rendezvous peer (μέθοδος startJxta).
- Εύρεση του επιθυμητού Group στο δίκτυο (αναζήτηση με τη μέθοδο findGroup, έλεγχος των «ευρημάτων» στην «αυτόματα καλούμενη» discoveryEvent). Και εδώ, όπως γίνεται και στο λογισμικό του gateway peer, θα μπορούσε να χρησιμοποιηθεί η προσέγγιση του «γνωστού γκρουπ» (well-known Group-ID technique). Να μην αναζητήσει, δηλαδή, ο peer το Group αλλά να γίνει κατευθείαν μέλος του, χρησιμοποιώντας την υπόθεση ότι γνωρίζει σε ποιο Group θέλει να εισχωρήσει. Προτιμήθηκε όμως η αναζήτηση, αφενός γιατί δεν έχει νόημα να δημιουργήσει ο χρήστης ένα Group, αν αυτό δεν υπάρχει ήδη και αφετέρου για να υλοποιηθεί ως εναλλακτική τακτική, μια και μελλοντικά μπορεί να προστεθούν έλεγχοι ταυτοποίησης για την εισχώρηση στο Group.
- Εισχώρηση στο Group (μέθοδος joinGroup).
- Εύρεση των gateway peers, δηλαδή των σχετικών PeerAdvertisements (μέθοδος αναζήτηση με τη μέθοδο findPeers, έλεγχος των «ευρημάτων» στην «αυτόματα καλούμενη» discoveryEvent). Για κάθε εύρεση ενός νέου Advertisement που «διαφημίζει» έναν gateway peer, καλείται η analyzeDescription για να αναλύσει την XML περιγραφή για την οποία έγινε λόγος στην προηγούμενη υποενότητα και να αποθηκευτούν οι πληροφορίες σε κατάλληλες δομές. Αυτές οι πληροφορίες θα είναι απαραίτητες στον query processor για την σωστή διάσπαση των queries της εφαρμογής και τη σωστή αποστολή των εξειδικευμένων queries που θα προκύπτουν.
- Καταχώρηση του PublicQueryHandler ως χειριστή των queries που θα γίνονται στο

Resolver Service του Group για αυτόν (μέθοδος registerPublicQueryHandler).
- Αδράνεια, αναμένοντας κλήση του μεθόδων του από την εφαρμογή ή τον query processor. Βασικές μέθοδοι είναι οι εξής:
findPeers: αναφέρθηκε ήδη, μπορεί να χρησιμοποιηθεί ανά πάσα στιγμή για ανανέωση των πληροφοριών σχετικά με τα υπάρχοντα δίκτυα αισθητήρων.
ceidSendQuery: στέλνει ένα query χρησιμοποιώντας το Resolver Service.
ceidGetImage: εκκινεί τη διαδικασία εξυπηρέτησης ενός query χρησιμοποιώντας pipes.

Ο Querier διαθέτει, φυσικά, και άλλες μεθόδους, όπως για παράδειγμα αυτές που επιστρέφουν τις δομές όπου αποθηκεύονται οι πληροφορίες από τις περιγραφές των gateways.

- Οι κλάσεις **PublicQueryMsg**, **PublicResponseMsg**, **PublicQueryHandler** και **RequestAdvertisement** υπάρχουν και εδώ, όπως περιγράφηκαν στην προηγούμενη υποενότητα. Αντί για την ProviderListener, για να προσδοθεί σε συνδυασμό με την RequestAdvertisement η απαραίτητη λειτουργικότητα των pipes, υπάρχει εδώ η κλάση QuerierListener, που είναι και η επόμενη που περιγράφεται.
- Η κλάση **QuerierListener** υλοποιεί (implements) την PipeMsgListener. Για κάθε query που πρόκειται να εξυπηρετηθεί από pipes δημιουργείται ένα αντικείμενο της QuerierListener και καλείται η μέθοδος listen αυτού. Αυτή η μέθοδος δημιουργεί ένα input pipe το οποίο σχετίζεται με τον παρόντα listener, ενώ στη συνέχεια δημιουργεί και δημοσιεύει ένα RequestAdvertisement το οποίο «διαφημίζει» την ύπαρξη του εν λόγω pipe. Στη συνέχεια, αν ο κατάλληλος gateway εντοπίσει και εξυπηρετήσει αυτό το RequestAdvertisement, τότε η μέθοδος pipeMsgEvent (που πρέπει να υπάρχει σε κάθε κλάση που υλοποιεί την PipeMsgListener) διαχειρίζεται τα λαμβανόμενα μηνύματα (τα οποία, όπως έχει ήδη αναφερθεί, έχουν δυαδικά δεδομένα, αριθμό ακολουθίας μηνύματος, καθώς και άλλα πεδία), ώστε όταν έχουν παραληφθεί όλα τα μηνύματα να καταγραφούν οι πληροφορίες που είναι απαραίτητες στον Query Processor για να συντεθεί η απάντηση, και στη συνέχεια να κλείσει το pipe.

Σε αυτήν την κλάση υπάρχει και η σταθερά PACKET_SIZE, που καθορίζει το μέγεθος του μηνύματος (που με τη σειρά του καθορίζει, προφανώς, και το πλήθος των μηνυμάτων που απαιτείται να αποσταλούν για την εξυπηρέτηση ενός query, παράμετρος σημαντική για την αποδοτικότητα του δικτύου). Η τιμή του PACKET_SIZE πρέπει να συμφωνεί με την αντίστοιχη που υπάρχει στην κλάση ProviderListener των gateways. Επίσης, και οι υπόλοιπες λεπτομέρειες όσον αφορά το πώς αποστέλλονται τα μηνύματα από τον ProviderListener πρέπει να είναι γνωστές ώστε να γίνεται σωστά η παραλαβή και σύνθεση της πληροφορίας που περιέχεται στα μηνύματα. Η βοηθητική μέθοδος storeBinFile καταγράφει τα δυαδικά δεδομένα ώστε να «διαβαστούν» από τον Query Processor.

- Η κλάση **QueryProcessor** είναι αυτή που υλοποιεί τη διάσπαση (split) των queries της εφαρμογής σε jxta queries, όπως εξηγήθηκε στο 4.4.1. Για να γίνει αυτό αξιοποιείται η γνώση της δομής του δικτύου, η οποία βρίσκεται αποθηκευμένη σε πίνακες (Hashtables). Τα δεδομένα που καθορίζουν τις τιμές των μεταβλητών ενός αντικειμένου QueryProcessor προέρχονται από την εφαρμογή και καθορίζουν τη πορεία που θα ακολουθήσει η υπόλοιπη διαδικασία (δηλαδή ποιες μέθοδοι της QueryProcessor θα κληθούν και τι queries θα δημιουργηθούν). Πριν από εκκίνηση της διαδικασίας

διάσπασης του query, γίνεται έλεγχος εγκυρότητας των παραπάνω δεδομένων, ώστε αν δεν είναι έγκυρα να μη γίνουν περιττές διαδικασίες.

Η μέθοδος findResults είναι αυτή που καλείται από τον «χρήστη της κλάσης» και αναλαμβάνει να καλέσει τις υπόλοιπες μεθόδους για να εξυπηρετήσει το query της εφαρμογής και, αφότου ολοκληρωθεί η διαδικασία, να επιστρέψει ένα (πιθανώς κενό) σύνολο αποτελεσμάτων. Η κλήση, λοιπόν, της findResults πυροδοτεί την εκτέλεση τριών λειτουργιών, που εκτελούνται με την κλήση τριών μεθόδων (πιθανώς διαφορετικών ανα περίσταση). Πρώτον γίνεται ο έλεγχος της εγκυρότητας των δεδομένων, δεύτερον η διάσπαση του query και η αποστολή των επιμέρους queries στο δίκτυο και τρίτον η σύνθεση των παραληφθέντων αποτελεσμάτων. Η πρώτη λειτουργία γίνεται από τη μέθοδο checkInputValid, η δεύτερη γίνεται από μία από τις μεθόδους rangeAnalysis, specificAnalysis, pointAnalysis και regionAnalysis, ενώ η τρίτη γίνεται από την readResults. Η τελευταία από τις παραπάνω περιέχει και δύο ενδιαφέροντα σημεία. Τη διαχείριση των χρονικών προθεσμιών (timeouts) των queries και την αντιμετώπιση της διαφορετικότητας των τύπων δεδομένων των απαντήσεων.

Οι χρονικές προθεσμίες (timeouts) τίθενται κυρίως για το ενδεχόμενο πολύ καθυστερημένης παραλαβής μιας απάντησης. Να σημειωθεί ότι η προθεσμία (timeout) δεν αφορά το query της εφαρμογής αλλά κάθε ένα από τα jxta queries ξεχωριστά, γεγονός που δίνει ευελιξία, μια και αν δημιουργηθεί πρόβλημα μόνο σε κάποια από αυτά, τα υπόλοιπα θα απαντηθούν κανονικά. Στην παρούσα υλοποίηση, το timeout έχει οριστεί στα 5 δευτερόλεπτα και δεν ισχύει για τα queries που απαιτούν δυαδικά δεδομένα (δηλαδή αυτά που εξυπηρετούνται από pipes), αφού αυτά χρησιμοποιούνται και για τη μεταφορά απαντήσεων με σχετικά μεγάλο όγκο δεδομένων. Αυτό, βέβαια, μπορεί να τροποποιηθεί και να προσαρμοστεί εύκολα στις εκάστοτε ανάγκες.

Η διαφορετικότητα των τύπων δεδομένων των απαντήσεων αφορά, προφανώς, μόνο απαντήσεις που διακινούνται με το Resolver Service, αφού οι υπόλοιπες έχουν έτσι κι αλλιώς κοινό τύπο δεδομένων, δηλαδή δυαδικά δεδομένα. Για την αντιμετώπιση του ζητήματος δημιουργείται ένα αντικείμενο DataTypeAdapter, το οποίο βοηθά να τοποθετηθεί στη δομή των επιστρεφόμενων αποτελεσμάτων ένα αντικείμενο κατάλληλου τύπου δεδομένων (π.χ. Integer για ακεραίους, String για αλφαριθμητικά κτλ.). Η κλάση DataTypeAdapter περιγράφεται ακολούθως.

- Η κλάση **DataTypeAdapter** έχει στόχο να είναι επεκτάσιμη ώστε να μπορεί να προσαρμοστεί σε πιθανές εξειδικευμένες ανάγκες κάποιων εφαρμογών. Χρησιμοποιείται για να μεταμορφώσει τις απαντήσεις που παραλαμβάνονται σε αντικείμενα διαφόρων τύπων, δηλαδή σε αντικείμενα που είναι κατάλληλα, ανάλογα με τον τύπο δεδομένων της εκάστοτε απάντησης. Η παρούσα υλοποίηση επιστρέφει ακεραίους, δεκαδικούς (μονής ή διπλής ακρίβειας), αλφαριθμητικά κ.α. ενώ μπορεί να επεκταθεί για να εξυπηρετήσει οποιοδήποτε τύπο (π.χ. πίνακες, μεικτούς τύπους ή οποιαδήποτε άλλη μορφή απάντησης), αρκεί ο τύπος δεδομένων να είναι γνωστός εκ των προτέρων και να χρησιμοποιείται η κατάλληλη «ονοματολογία» τύπων δεδομένων. Οι κλάσεις που ακολουθούν είναι μικρές και απλές και αντιστοιχούν σε βασικές οντότητες του συστήματος.
- Η κλάση **GeoPoint** έχει ένα πεδίο για το γεωγραφικό μήκος (longitude) και ένα για το γεωγραφικό πλάτος (latitude) και αντιπροσωπεύει ένα γεωγραφικό σημείο. Είναι χρήσιμο για την περιγραφή της κάλυψης που παρέχουν τα επιμέρους WSN και για την εκτέλεση queries βασισμένων σε γεωγραφική θέση/περιοχή.

- Η κλάση **Sensor** αντιπροσωπεύει έναν αισθητήρα και τα πεδία της τον περιγράφουν πλήρως (ID, όνομα, μονάδες που χρησιμοποιεί, ακρίβεια που παρέχει, φυσικά μεγέθη που μπορεί να μετρήσει κτλ.). Τέτοια αντικείμενα χρησιμοποιούνται κυρίως κατά τη διάσπαση των queries για να καθοριστεί τι queries μπορούν να σταλούν και σε ποιους αισθητήρες πρέπει να σταλούν.
- Η κλάση **Result** αντιπροσωπεύει ένα αποτέλεσμα ενός jxta query ενώ ένα query εφαρμογής απαντάται, όπως πρέπει να έχει γίνει κατανοητό, από ένα σύνολο αντικειμένων τύπου Result. Τα πεδία του είναι τα εξής: τοποθεσία, είδος αισθητήρα, χρησιμοποιούμενη συνάρτηση, τύπος δεδομένων, απάντηση, απάντηση σε μορφή κειμένου. Η χρήση των αντικειμένων είναι σχετικά προφανής.

Τα αρχεία "run" (.sh και .bat), που βρίσκονται τόσο στον ShareSenseGateway όσο και στον ShareSenseUser, χρησιμοποιούνται για την εκτέλεση του λογισμικού σε διάφορες πλατφόρμες, στις οποίες έγιναν και δοκιμές. Για διαφορετικές πλατφόρμες θα χρειάζονται διαφορετικά "run" αρχεία. Υπενθυμίζεται ότι ο κάθε peer μπορεί να εκτελείται σε οποιαδήποτε πλατφόρμα, χωρίς να επηρεάζεται η επικοινωνία του με τους υπόλοιπους.

Πιλοτική εφαρμογή για το ShareSense

Στα πλαίσια της εργασίας, ως απαραίτητο συστατικό για τις δοκιμές όλων των επιμέρους λειτουργιών του ShareSense αλλά και του συστήματος ως σύνολο, αναπτύχθηκε και λογισμικό εφαρμογής. Αυτό είχε διάφορες μορφές στα διάφορα στάδια της δημιουργίας του συστήματος, μόνο ορισμένες από τις οποίες θα σχολιαστούν, ενώ στο συνοδευτικό CD υπάρχει μόνο μία τελική έκδοση. Η τελική αυτή έκδοση είναι ένα γραφικό περιβάλλον για την παρακολούθηση ενός συστήματος βασισμένου στο ShareSense και βρίσκεται μαζί με το λογισμικό χρήστη (ShareSenseUser). Ο κώδικας βρίσκεται εξ' ολοκλήρου στο αρχείο SenseGUI.java. Η ανάπτυξη έγινε με τα ευρέως χρησιμοποιούμενα πακέτα γραφικών awt και swing της java, αλλά ο περαιτέρω σχολιασμός λεπτομερειών της υλοποίησης δεν αφορά την παρούσα εργασία.

Στόχος κάθε ενδιάμεσου λογισμικού είναι να παρέχει στο δημιουργό εφαρμογών μια προγραμματιστική διεπαφή (API) και, από εκεί και πέρα, πλήρη ευελιξία όσον αφορά τον τύπο και τη μορφή της εφαρμογής. Το ShareSense το επιτυγχάνει αυτό, όντας μάλιστα αρκετά απλό, με την έννοια ότι οι λειτουργίες που εκτελούν οι peers δε θα διαφέρουν πολύ από εφαρμογή σε εφαρμογή. Σε κάθε περίπτωση αναμένεται να προηγείται η χρήση ενός στάνταρ API για τη σύνδεση των peers στο δίκτυο, ενώ θα ακολουθεί η κατά βούληση χρήση και αξιοποίηση των δυνατοτήτων του Query Processor.

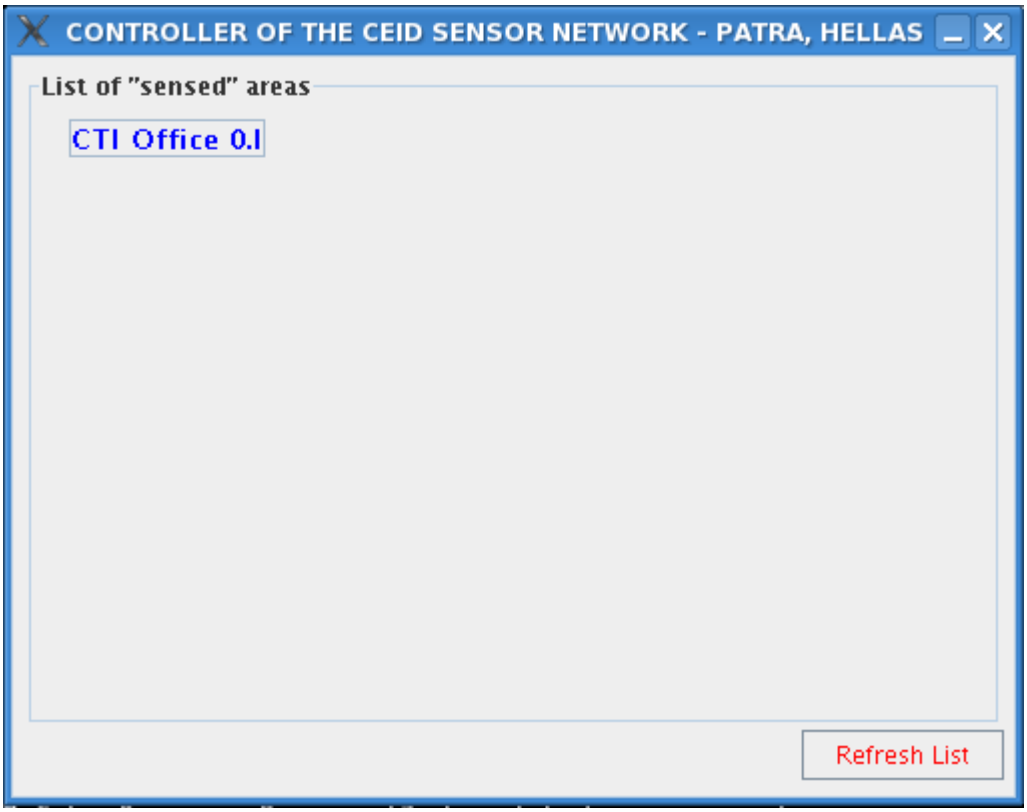
Ποικιλία Δυνατοτήτων

Οι εφαρμογές αισθητήρων ποικίλουν από εφαρμογές που απλώς παρουσιάζουν δεδομένα σε εφαρμογές που εμπεριέχουν πολύπλοκη ανάλυση και επεξεργασία των δεδομένων, με εξαγωγή συμπερασμάτων, λήψη αποφάσεων και πυροδότηση δραστηριοτήτων. Μπορεί να είναι αλληλεπιδραστικές με το χρήστη ή όχι και η μορφή τους μπορεί να είναι απλή ή να περιέχει μεγάλα και εντυπωσιακά γραφικά περιβάλλοντα. Μπορεί να βασίζονται σε τεράστιες βάσεις δεδομένων ή να χρησιμοποιούν ελάχιστη μνήμη βασιζόμενοι σε στοιχειώδη δεδομένα. Μπορεί, επίσης, να ποικίλουν όσον αφορά ένα πλήθος από άλλες παραμέτρους. Το ShareSense, ως overlay, δεν επιβάλει κανέναν σχετικό περιορισμό και μπορεί να υποστηρίξει τη δημιουργία κάθε λογής εφαρμογής.

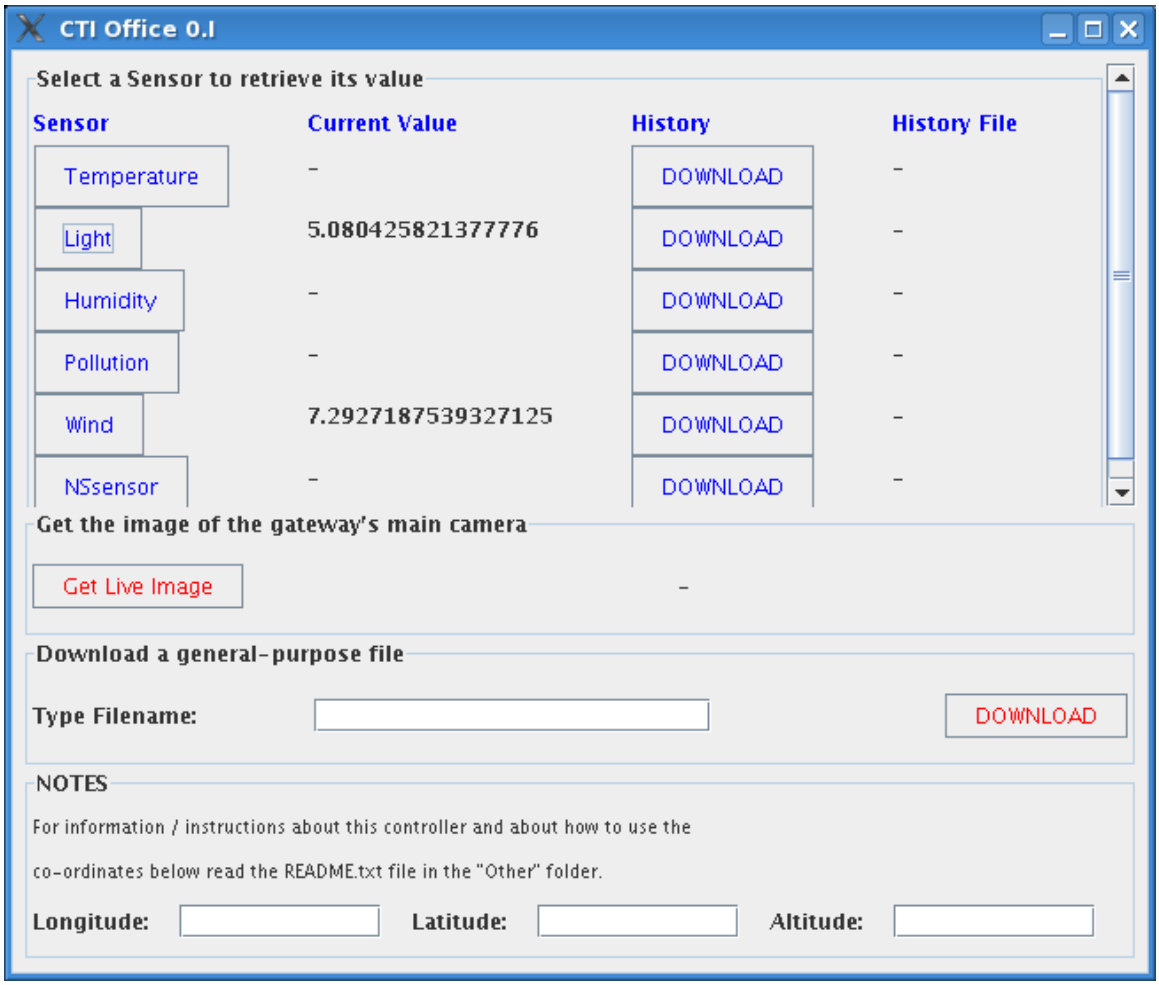
Κατά τις δοκιμές, για παράδειγμα, χρησιμοποιήθηκαν αρχικά απλές εφαρμογές χωρίς γραφικά περιβάλλοντα και χωρίς αλληλεπίδραση με το χρήστη, για να υλοποιηθεί στη συνέχεια ένα γραφικό περιβάλλον που έχει όμως τελείως διαφορετική λογική (και δυνατότητες) από την τελική έκδοση (την οποία θα δούμε στην υποενότητα 4.5.1), ενώ σε οποιοδήποτε σημείο θα μπορούσε να προστεθεί και επεξεργασία των δεδομένων κτλ. Όλα τα παραπάνω εξακολουθούν να επιδεικνύουν μόνο σε πολύ μικρό βαθμό τη διαφορετικότητα των εφαρμογών αισθητήρων που μπορούν να «χτιστούν» πάνω στο ShareSense.

Μέσω των γραφικών περιβαλλόντων υλοποιήθηκε και μια υποτυπώδης υπηρεσία ανταλλαγής αρχείων που δείχνει τη σχετικότητα μιας λειτουργίας όπως η ανταλλαγή αρχείων με την ανταλλαγή δεδομένων αισθητήρων, καθώς και το σημαντικό ρόλο που διαδραματίζουν τα αρχεία στα συστήματα που είναι βασισμένα στη JXTA. Ακολουθούν εικόνες από ένα πρώιμο γραφικό περιβάλλον που υλοποιήθηκε και έχει την εξής λογική:

- Παρουσίαση των υπαρχόντων παρακολουθούμενων περιοχών (δηλαδή των υπαρχόντων gateways ή, αλλιώς, των υπαρχόντων αυτόνομων WSN) και επιλογή ενός εξ' αυτών από το χρήστη.
- Αλληλεπίδραση και επικοινωνία με το επιλεγμένο αυτόνομο WSN. Το είδος της αλληλεπίδρασης που μπορεί να επιτευχθεί είναι προφανές από την εικόνα και βασίζεται και σε υποθέσεις που είναι σφιχτές και κατά κάποιο τρόπο βλάπτουν τη γενικότητα (π.χ. ύπαρξη μόνο ενός αισθητήρα ενός συγκεκριμένου τύπου σε ένα αυτόνομα WSN. Αυτά, τα προβλήματα δεν υπάρχουν, βέβαια, στην έκδοση που παρουσιάζεται στο 4.5.1). Συνεπώς, αυτό το GUI θα μπορούσε να εξυπηρετήσει περιορισμένο σύνολο δικτύων, συγκεκριμένου τύπου.



ΕΙΚΟΝΑ 4.1 – Παρουσίαση των παρακολουθούμενων περιοχών (σ.σ. gateways) με δυνατότητα επιλογής οποιουδήποτε εξ' αυτών. Στο παράδειγμα της εικόνας υπάρχει μόνο μία παρακολουθούμενη περιοχή. Μετά την επιλογή εμφανίζεται ένα γραφικό περιβάλλον «χειρισμού» όπως αυτό της επόμενης εικόνας

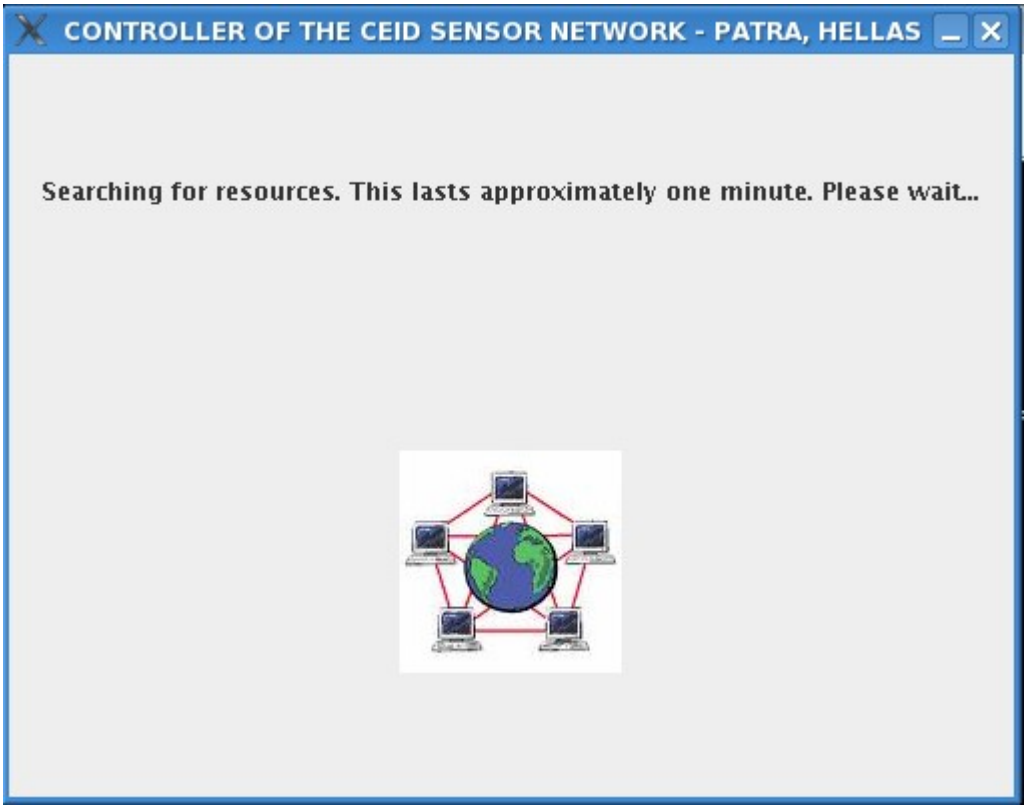


ΕΙΚΟΝΑ 4.2 – Γραφικό περιβάλλον που παρουσιάζει λίστα των διαθέσιμων αισθητήρων με δυνατότητα επιλογής καθενός από αυτούς ώστε να γίνει το αντίστοιχο query και να ανανεωθεί live η αντίστοιχή τιμή. Το γραφικό περιβάλλον παρέχει, όπως φαίνεται, και άλλες δυνατότητες, όπως αιτήσεις για μεταφορά ιστορικού μετρήσεων / μεταφορά εικόνων / μεταφορά αρχείων

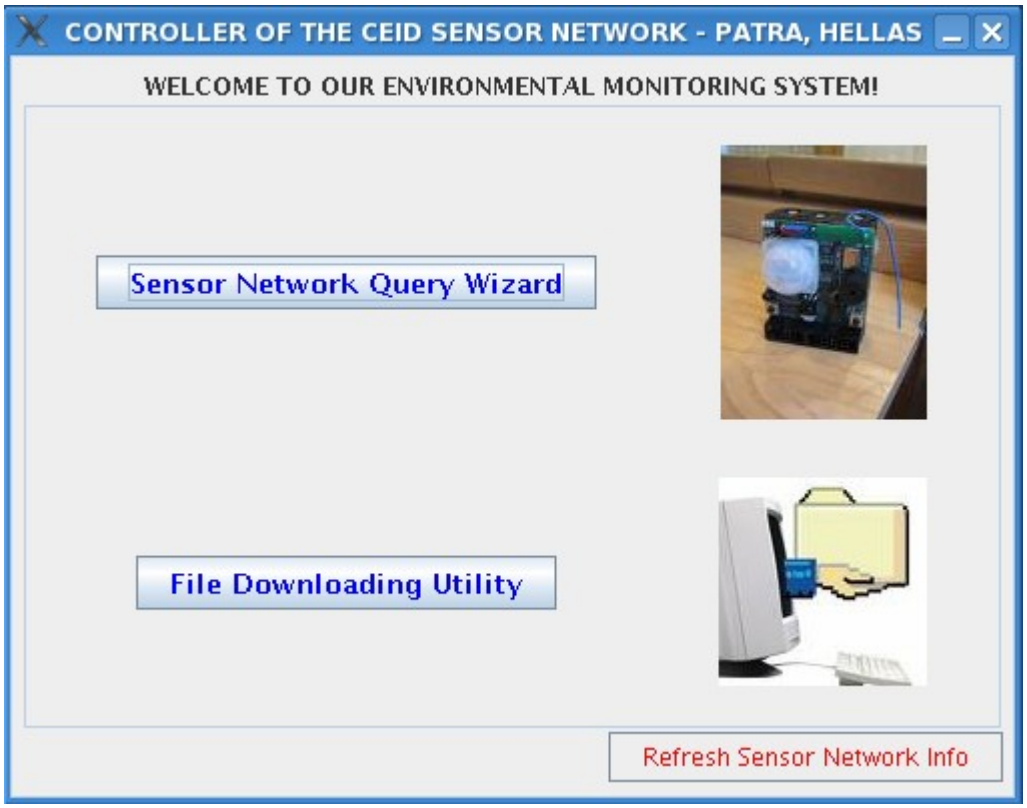
4.1.9 Η τελική έκδοση της πιλοτικής εφαρμογής

Η εφαρμογή γραφικού περιβάλλοντος που παραδίδεται μαζί με την εργασία είναι ολοκληρωμένη και ενδεικτική όσον αφορά τον τρόπο με τον οποίο γίνονται τα queries στο δίκτυο. Δεν είναι πολύπλοκη στη χρήση, παρέχει πολλούς δυνατούς τρόπους για να γίνουν τα queries στο δίκτυο και, όσον αφορά τα δεδομένα, δεν υλοποιεί κάτι περισσότερο από απλή παρουσίαση των απαντήσεων. Είναι απόλυτα γενική, με την έννοια ότι δεν κάνει καμία υπόθεση για τα υποκείμενα δίκτυα αισθητήρων και άρα μπορεί να χρησιμοποιηθεί με οποιαδήποτε από αυτά, ενώ ενσωματώνει και αυτή μια υποτυπώδη υπηρεσία ανταλλαγής αρχείων.

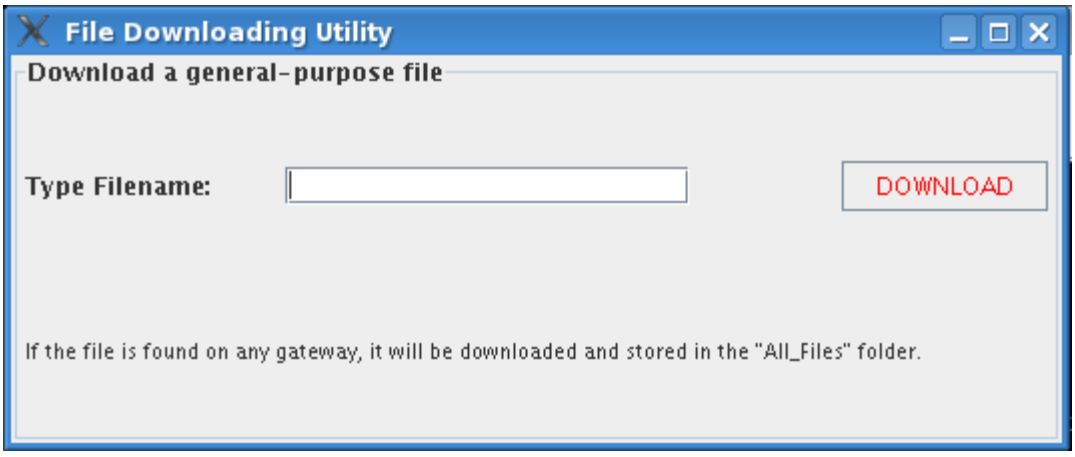
Ενδεικτικό είναι ότι η εφαρμογή είναι προσανατολισμένη σε μεγάλο βαθμό στην αναζήτηση δεδομένων με γεωγραφικά κριτήρια. Το ίδιο ισχύει εν γένει και για το API που παρέχεται από το ShareSense. Αυτό είναι σημαντικό για δύο λόγους. Πρώτον, στις περισσότερες περιπτώσεις, οι χρήστες εφαρμογών αισθητήρων ενδιαφέρονται όντως για γεωγραφικά σημεία ή περιοχές, αδιαφορώντας πλήρως για τη δομή των υποκείμενων δικτύων. Δεύτερον, αυτός ο προσανατολισμός είναι ένα βήμα προς τη σύνδεση του ShareSense (ή εφαρμογών όπως η παρούσα) με πιο ενδιαφέροντα και εντυπωσιακά γραφικά περιβάλλοντα (π.χ. διαδικτυακά προσβάσιμους γεωγραφικούς χάρτες κτλ.). Ακολουθούν εικόνες που μαρτυρούν τις δυνατότητες και τον τρόπο χρήσης της εφαρμογής.



ΕΙΚΟΝΑ 4.3 – Σημείο εκκίνησης. Το σύστημα του χρήστη κάνει μια πρώτη αναζήτηση (και καταγραφή) πόρων (βλ. βήματα 3,4 του σχήματος 4.3) πριν αρχίσει η αλληλεπίδραση με το χρήστη

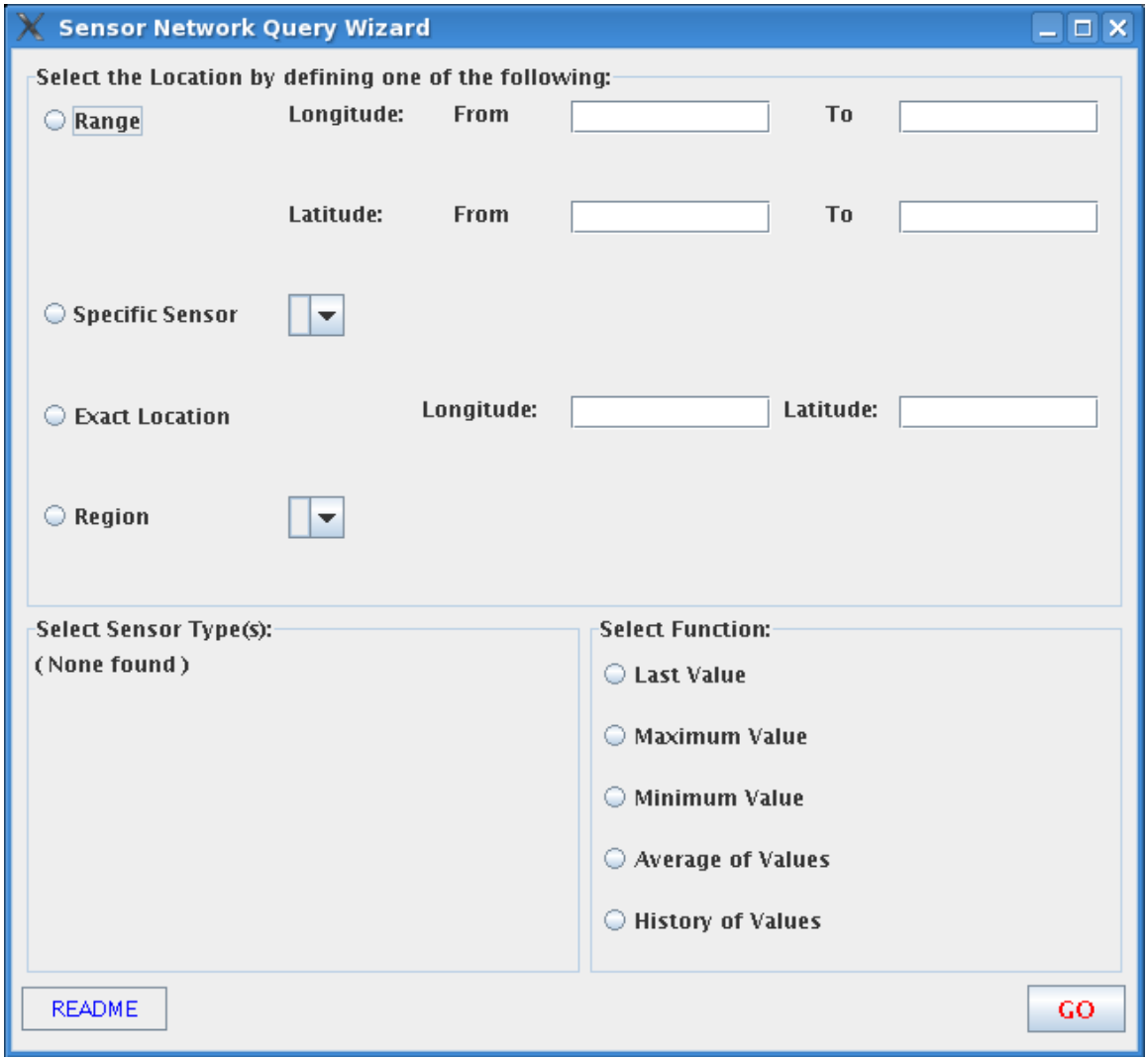


ΕΙΚΟΝΑ 4.4 – Κεντρικό μενού. Ο χρήστης μπορεί να επιλέξει μεταξύ της κύριας λειτουργίας (queries για μετρήσεις αισθητήρων) και της υποτυπώδους υπηρεσίας ανταλλαγής αρχείων. Η επιλογή "refresh" ανανεώνει τις πληροφορίες σχετικά με τη δομή του δικτύου.

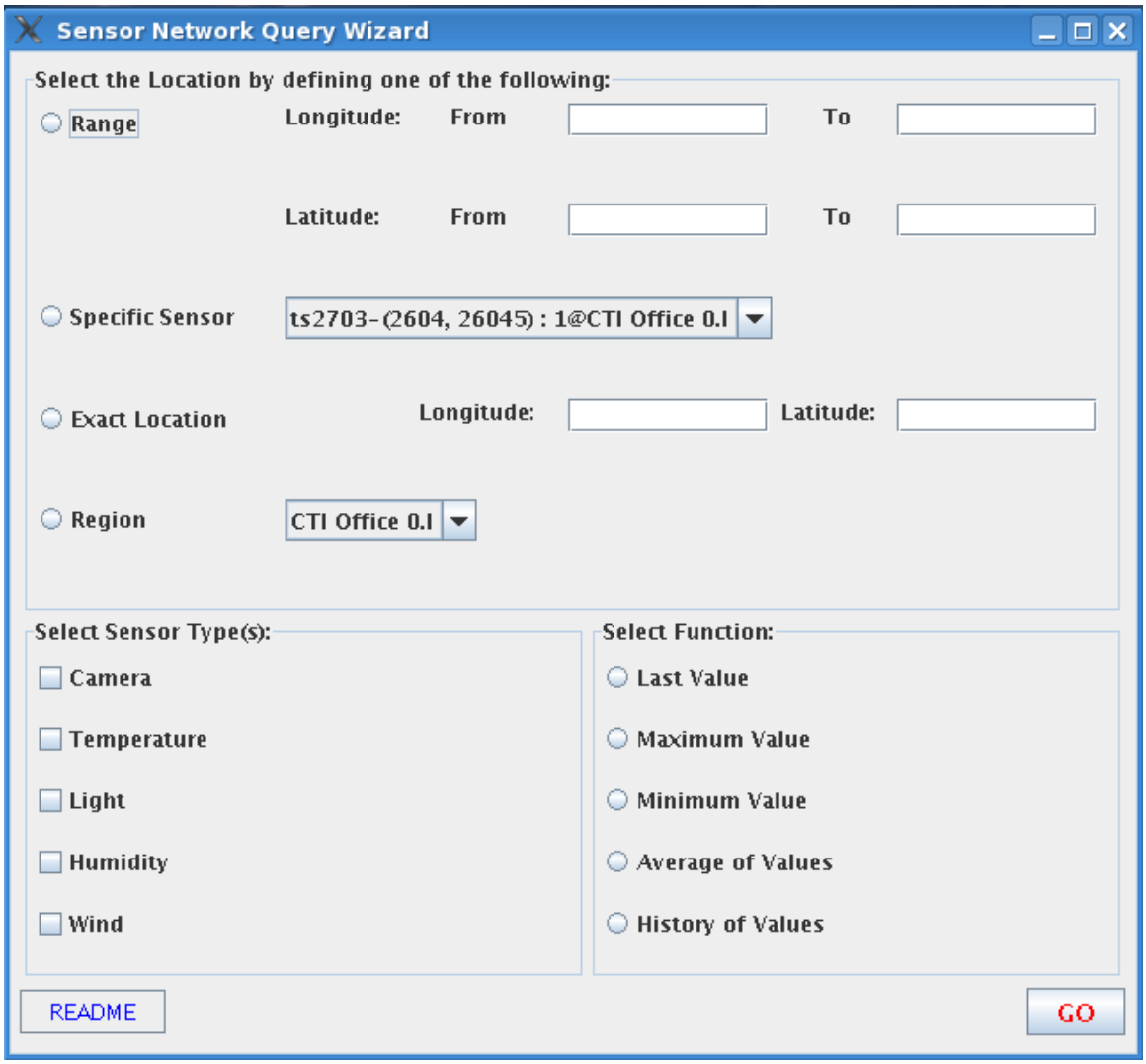


ΕΙΚΟΝΑ 4.5 – Υποτυπώδης υπηρεσία ανταλλαγής αρχείων

Έστω ότι αρχικά εκτελείται το λογισμικό χωρίς κανένας gateway να είναι συνδεδεμένος στο δίκτυο και επιλέγεται από το χρήστη ο Sensor Network Query Wizard. Τότε ανοίγει το παράθυρο της εικόνας 4.6. Αν στη συνέχεια συνδεθεί στο δίκτυο ο πειραματικός gateway (σ.σ. αυτός που βρίσκεται και στο συνοδευτικό CD), ο χρήστης κάνει "refresh" από το κεντρικό μενού και στη συνέχεια ανοίξει ξανά τον Sensor Network Query Wizard, τότε ανοίγει το παράθυρο της εικόνας 4.7. Αν υπήρχαν και άλλοι gateways με επιπλέον δυνατότητες, τότε θα φαίνονταν, φυσικά, όλες οι δυνατότητες.



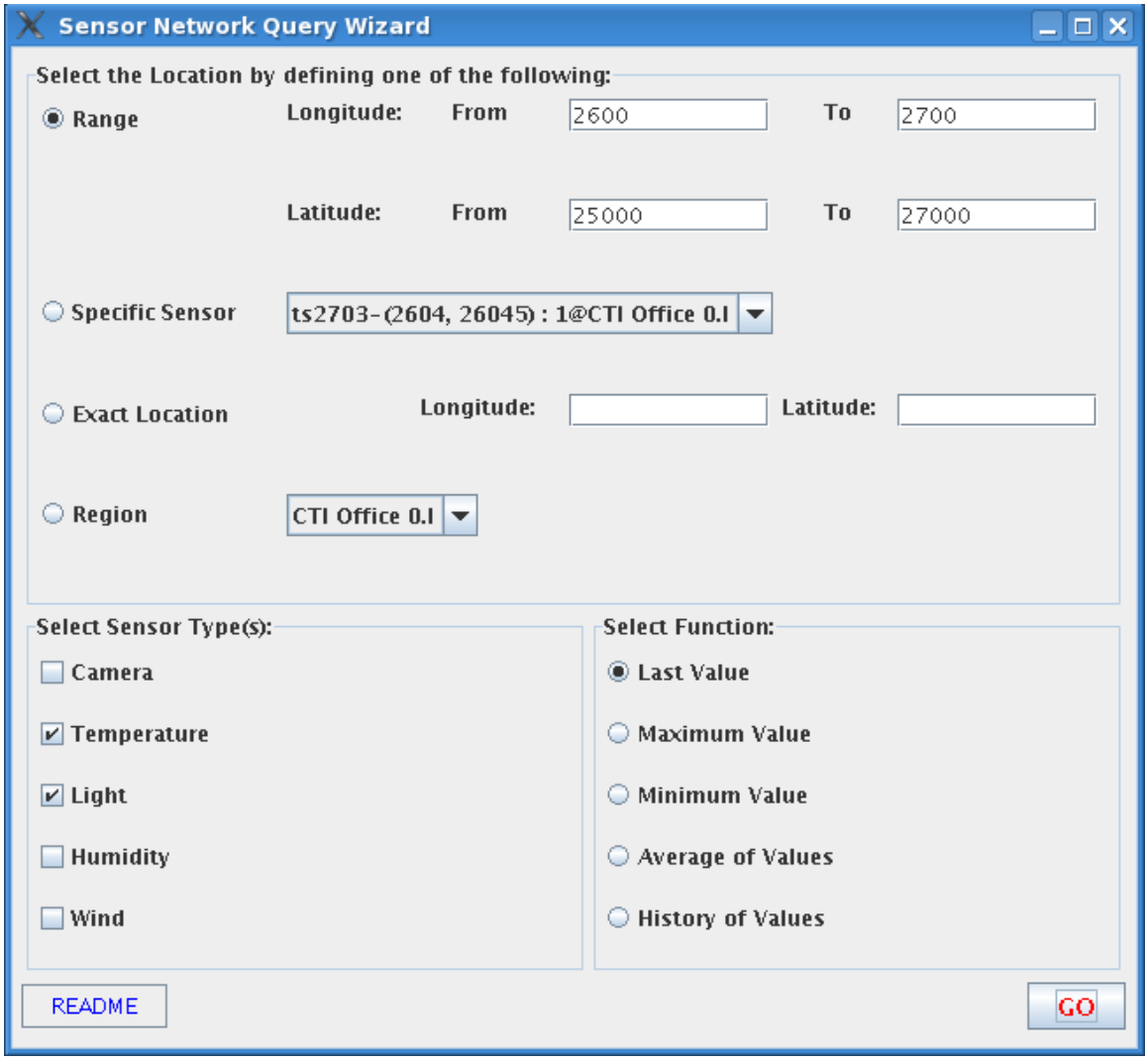
ΕΙΚΟΝΑ 4.6 – Ο Sensor Network Query Wizard με την ανυπαρξία gateways



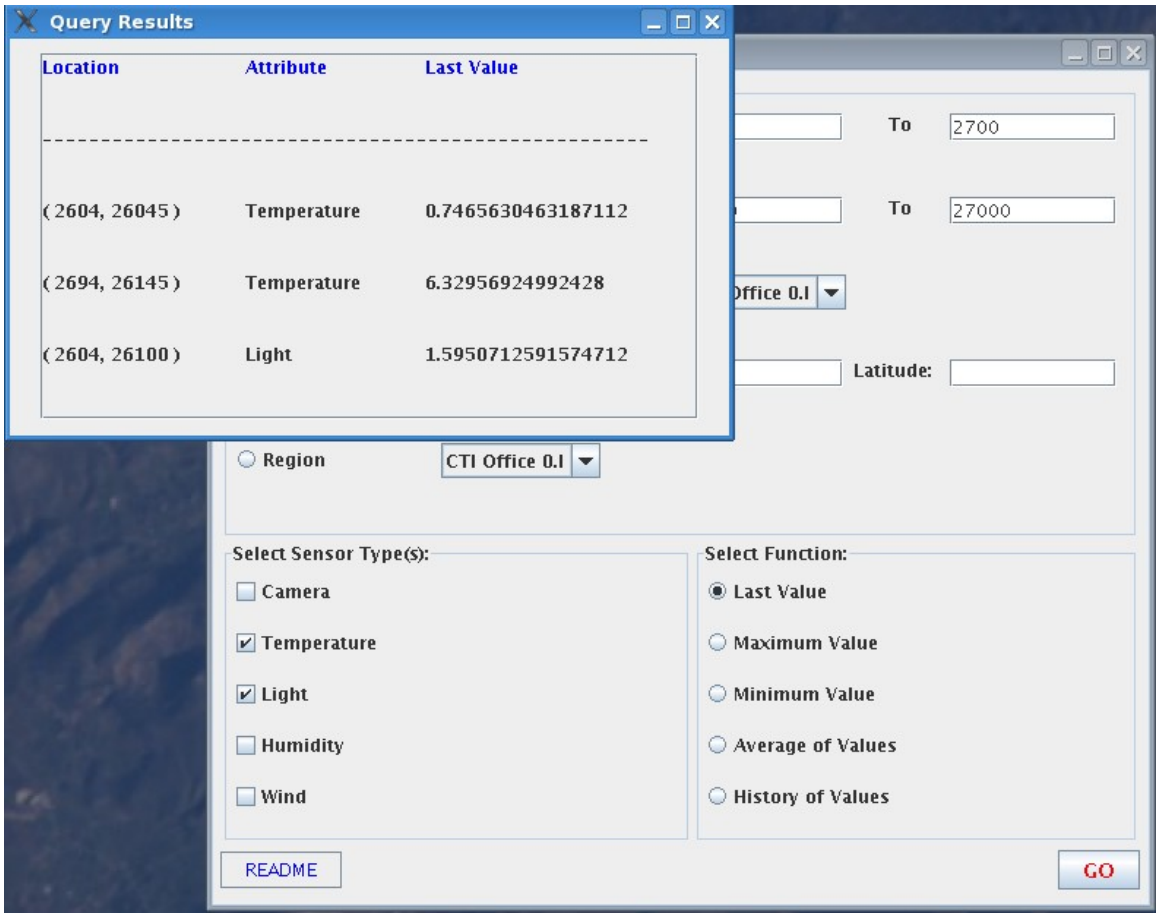
ΕΙΚΟΝΑ 4.7 – Ο Sensor Network Query Wizard μετά τη σύνδεση του πειραματικού gateway και την ανανέωση των πληροφοριών

Υπενθυμίζεται ότι τα στοιχεία που φαίνονται βασίζονται στα όσα περιγράφονται στην XML περιγραφή των gateways. Αν υποθέσουμε ότι γίνεται κάποια αλλαγή στη δομή του WSN που ελέγχεται από έναν gateway (π.χ. προστίθεται ένας αισθητήρας), τότε πρέπει να ενημερωθεί η XML περιγραφή και να επανεκκινηθεί το λογισμικό του gateway. Στη συνέχεια, όταν ένας χρήστης αναζητήσει πόρους θα βρει τη νέα περιγραφή και άρα και ο Sensor Network Query Wizard της εφαρμογής θα παρουσιάζει διαφορετικές δυνατότητες. Με το παρόν λογισμικό, αυτό μπορεί να δοκιμαστεί με τροποποιήσεις της περιγραφής του gateway (αρχείο PeerDescription.txt), επανεκκίνηση κτλ.

Ακολουθεί ένα παράδειγμα επιλογών του χρήστη βασισμένο σε γεωγραφική περιοχή οριοθετημένη από συντεταγμένες. Στο παράδειγμα ζητούνται οι πιο πρόσφατες τιμές δύο συγκεκριμένων δυνατοτήτων (θερμοκρασίας και φωτός) εντός αυτής της περιοχής. Οι επιλογές αυτές φαίνονται στην εικόνα 4.8. Αν αφότου οριστούν αυτές οι επιλογές πατηθεί το κουμπί GO, τότε γίνεται η απαραίτητη επικοινωνία και μετά την ολοκλήρωσή της και την παραλαβή των απαντήσεων, εμφανίζονται τα σχετικά αποτελέσματα, όπως φαίνεται στην εικόνα 4.9.



ΕΙΚΟΝΑ 4.8 – Ο Sensor Network Query Wizard αφότου οριστούν κάποιες επιλογές από το χρήστη



ΕΙΚΟΝΑ 4.9 – Αποτελέσματα μετά τη λήψη απαντήσεων για τις επιλογές της προηγούμενης εικόνας

5. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΣΚΕΨΕΙΣ

Αξιολογώντας την εργασία συνολικά, κεντρικό θέμα είναι το ενδιάμεσο λογισμικό για συστήματα δικτύων αισθητήρων. Τόσο η μελέτη όσο και ο σχεδιασμός και η υλοποίηση που έγιναν στα πλαίσια της εργασίας επικεντρώνονται σε αυτό. Έχοντας φτάσει σε αυτό το σημείο, μπορούν να εξαχθούν κάποια γενικότερα συμπεράσματα που αφορούν τον τομέα. Μπορεί να αξιολογηθεί το πόσο χρήσιμα είναι τελικά αυτά τα λογισμικά, το πώς φαντάζει από γενική σκοπιά η παρούσα κατάσταση, καθώς και το τι είδους εξελίξεις αναμένεται να υπάρχουν στο μέλλον. Αφού σχολιαστούν αυτά, θα γίνει μια πιο συγκεκριμένη αναφορά στη μελλοντική δουλειά που θα μπορούσε να γίνει σχετικά με αυτή τη μελέτη και το λογισμικό ShareSense.

Αξιολόγηση της χρησιμότητας των overlays

Γενικά, το ενδιάμεσο λογισμικό (overlay software) έχει τόσο μεγαλύτερη αξία:
-Πρώτον, όσο πιο συνηθισμένη είναι μεταξύ των εφαρμογών η απαίτηση για τη λειτουργικότητα που το λογισμικό αυτό υλοποιεί και
-Δεύτερον, όσο πιο δύσκολο είναι να υλοποιηθεί αυτή η λειτουργικότητα.
Δεν έχει, δηλαδή, ιδιαίτερο νόημα η ανάπτυξη ενδιάμεσου λογισμικού που πρόκειται να χρησιμοποιηθεί από ελάχιστες εφαρμογές. Μέχρι στιγμής, παρότι το ενδιαφέρον για τα δίκτυα αισθητήρων είναι πολύ μεγάλο, το πλήθος των σχετικών εφαρμογών δεν είναι τέτοιο που να καθιστά αυτόματα προφανή την τεράστια ανάγκη για τη συνεχή έρευνα σχετικά με τέτοιο λογισμικό. Είναι, όμως, μαθηματικά βέβαιο ότι η ανάπτυξη του τομέα θα είναι τέτοια που η έρευνα και η ανάπτυξη σχετικού ενδιάμεσου λογισμικού είναι καίρια για τη διευκόλυνση της ανάπτυξης πολλών και αξιόλογων εφαρμογών.

Το πόσο διευκολύνεται και προωθείται η ανάπτυξη εφαρμογών χάρη στο ενδιάμεσο λογισμικό έχει γίνει σαφές και η εμπειρία το δείχνει μέσα από διάφορους τομείς. Για παράδειγμα, πόσες δικτυακές εφαρμογές θα μπορούσαν να αναπτυχθούν αν δεν υπήρχαν έτοιμα overlays που αναλαμβάνουν να υλοποιήσουν τα πρωτόκολλα της δικτυακής επικοινωνίας; Ή πόσο εύκολη θα ήταν η κατασκευή αξιόλογων ιστοσελίδων αν δεν υπήρχαν εργαλεία με ενσωματωμένο ενδιάμεσο λογισμικό για τη δημιουργία βασικών modules; Η μελέτη και η συζήτηση που έγινε κατά την παρουσίαση των ενδιάμεσων λογισμικών καθιστά σαφές ότι η δουλειά που «κρύβεται» πίσω από ένα σύστημα δικτύων αισθητήρων είναι πολλή και βαριά. Χωρίς τη χρήση κάποιου ενδιάμεσου λογισμικού, η κατασκευή ενός τέτοιου συστήματος αποτελεί ένα μεγάλο έργο. Αυτό το γεγονός, σε συνδυασμό με την πρόβλεψη ότι οι εφαρμογές αισθητήρων θα είναι πάρα πολλές στο μέλλον (χάρη και στη μείωση του κόστους του απαιτούμενου υλικού) δικαιολογεί το ενδιαφέρον και την αναγκαιότητα για την ανάπτυξη τέτοιου λογισμικού.

Η παρούσα κατάσταση

Ο τομέας των εφαρμογών συστημάτων δικτύων αισθητήρων βρίσκεται σε σχετικά πρώιμο στάδιο. Ως εκ τούτου, δεν έχει αποδειχθεί το πώς ακριβώς θα είναι και το τι ανάγκες θα πρέπει να καλύπτει ένα βέλτιστο ενδιάμεσο λογισμικό για τέτοιες εφαρμογές. Είναι επίσης γεγονός ότι τα ενδιάμεσα λογισμικά που μελετήθηκαν και παρουσιάστηκαν βρίσκονται ακόμα σε πειραματικό στάδιο και κανένα δεν έχει χρησιμοποιηθεί ευρέως. Δε θα μπορούσε άλλωστε. Την παρούσα στιγμή, ο χώρος έχει να επιδείξει μόνο ορισμένες εφαρμογές μεμονωμένων και συνήθως μεγάλων σε έκταση WSN, αλλά ελάχιστα πράγματα όσον αφορά τη διασύνδεση πολλών WSN (που αποτελεί και κεντρική πρόκληση για το ενδιάμεσο λογισμικό).

Το συμπέρασμα που προκύπτει αν επιχειρήσουμε να συμψηφίσουμε τους στόχους όλων των αναπτυσσόμενων ενδιάμεσων λογισμικών είναι ότι η κύρια κατεύθυνση που ακολουθείται είναι περισσότερο προς τη μελλοντική δυνατότητα για δίκτυα πολύ μεγάλου (ή και παγκόσμιου) εύρους και λιγότερο προς τα μικρά, ιδιωτικά δίκτυα αισθητήρων. Λογικό άλλωστε, μια και οι περισσότερες προκλήσεις που αντιμετωπίζει το ενδιάμεσο λογισμικό αφορούν ανάγκες που συνήθως δεν υπάρχουν καν στα μικρά δίκτυα. Αυτό δε σημαίνει, φυσικά, ότι τα τελευταία δεν έχουν αντίστοιχη ανάπτυξη. Απλώς, ένας από τους στόχους των ενδιάμεσων λογισμικών είναι και η διασύνδεση των μικρών δικτύων. Ακόμα, δε σημαίνει ούτε ότι ο τομέας αφορά μόνο μεγάλους οργανισμούς. Αντιθέτως, τα μεγάλα/παγκόσμια αυτά δίκτυα σχεδιάζονται για να εξυπηρετούν και κοινές ανάγκες και για να βρίσκονται στη διάθεση και στην καθημερινή ζωή του καθενός.

Έτσι, λοιπόν, τα ενδιάμεσα λογισμικά που σχεδιάζονται την παρούσα στιγμή είναι όσο το δυνατόν γενικότερα και δεν θέτουν πολλούς περιορισμούς. Αυτό είναι καίριο για να μπορέσουν πιθανώς κάποια στιγμή να τεθούν standards και να είναι δυνατή η ενοποίηση δικτύων αισθητήρων με όσο το δυνατόν λιγότερα προβλήματα. Το τι θα χρησιμοποιηθεί ευρέως και το τι θα επικρατήσει ως ενδιάμεσο λογισμικό για συστήματα δικτύων αισθητήρων δε μπορεί να διαφανεί προς το παρόν.

Το μέλλον

Κάποια σχόλια για το μέλλον του τομέα έγιναν ήδη, αναπόφευκτα, στις προηγούμενες παραγράφους. Το βασικό είναι ότι, όπως σε κάθε περίπτωση, η πρακτική εφαρμογή θα δείξει το δρόμο για τη μελλοντική ανάπτυξη ενδιάμεσου λογισμικού. Πολλά ζητήματα έχουν αναλυθεί και πολλά έχουν προβλεφθεί αλλά μόνο όταν οι εφαρμογές αισθητήρων κατακλύσουν την αγορά θα γίνουν προφανή τα προβλήματα των ενδιάμεσων λογισμικών και θα «ζυμωθεί» μία βέλτιστη λύση. Κανένα από τα ενδιάμεσα λογισμικά που παρουσιάστηκαν δε φιλοδοξεί να γίνει η βάση για την ανάπτυξη μεγάλου πλήθους εφαρμογών. Όλα μαζί, αποσκοπούν στο να θέσουν τις βάσεις για την εκκίνηση της ευρείας ανάπτυξης εφαρμογών αισθητήρων. Μόνο όταν ξεκινήσει αυτή η ανάπτυξη και γίνουν σαφέστεροι οι στόχοι θα εμφανιστούν ευρέως χρησιμοποιούμενα, «εμπορικά» overlays. Επίσης, δεν πρέπει να αμελείται το γεγονός ότι οι εφαρμογές αισθητήρων αναμένεται να έχουν τόσες διαφορετικές μορφές (π.χ. με κίνηση ή όχι, με συνεργασία με

το διαδίκτυο ή όχι κτλ.) που δεν αναμένεται να εμφανιστούν κάποια αυστηρά standards, ούτε και να επικρατήσει ενδιάμεσο λογισμικό μίας μόνο συγκεκριμένης μορφής.

Μελλοντικές σκέψεις για το ShareSense

Οι μελλοντικές δραστηριότητες που μπορούν να γίνουν μελλοντικά σε σχέση με το ShareSense χωρίζονται σε τρεις κατηγορίες:
- Τη βελτίωση / επέκταση του λογισμικού.
- Τη σύνδεση του λογισμικού με το «από κάτω» επίπεδο.
- Την ανάπτυξη εφαρμογών βασισμένων στο ShareSense.

Και προς τις τρεις αυτές κατευθύνσεις υπάρχουν διάφορες σκέψεις και δυνατότητες, μια περίληψη των οποίων είναι οι τρεις παράγραφοι που ακολουθούν.

Όσον αφορά βελτιώσεις, κάποιες δομές της JXTA που χρησιμοποιούνται για την επικοινωνία μπορούν να συγκριθούν με άλλες, εναλλακτικές δομές και να επιλεγούν οι αποδοτικότερες (π.χ. απλά pipes VS σύνθετα pipes VS sockets κτλ.). Επίσης, πειράματα και δοκιμές μεγαλύτερης κλίμακας μπορούν να οδηγήσουν σε βελτιωμένες επιλογές κάποιων σταθερών και κάποιων τεχνικών. Το API που παρέχεται μπορεί να επεκταθεί ή/και να απλοποιηθεί (π.χ. υπερφορτώνοντας μεθόδους ώστε να δέχονται και λιγότερες παραμέτρους κτλ.). Βάσεις δεδομένων μπορούν να συνδεθούν με το σύστημα για την αποθήκευση πιθανώς μεγάλου όγκου πληροφοριών σχετικών με το δίκτυο και άλλα πολλά.

Το ShareSense απαιτείται, όπως έχει ήδη τονιστεί σε άλλα σημεία της εργασίας, να συνδεθεί με κάποιο άλλο λογισμικό, το οποίο επιτελεί τις λειτουργίες του χαμηλότερου επιπέδου (δηλαδή του επιπέδου αισθητήρων). Ένα παράδειγμα που έχει αναφερθεί είναι το επίπεδο αισθητήρων του jWebDust, ενώ μια πιθανότητα θα μπορούσε να είναι η ανάπτυξη ενός νέου λογισμικού, πιο κατάλληλου να επιτελέσει αυτό το ρόλο. Ούτως ή άλλως, το ShareSense μπορεί να συνδεθεί με οποιοδήποτε τέτοιο λογισμικό που είναι επαρκώς τμηματοποιημένο, αφού δε θέτει περιορισμούς για το χαμηλότερο επίπεδο. Συνεπώς, μελλοντικές δοκιμές πολλών διαφορετικών επιλογών θα μπορούσαν να οδηγήσουν σε συμπεράσματα για το ποια λογισμικά είναι κατάλληλα να συνδυαστούν με το ShareSense.

Η ανάπτυξη εφαρμογών διαφορετικών και πιθανώς πολυπλοκότερων της πιλοτικής μπορεί να αναδείξει δυνατότητες και αδυναμίες του ShareSense και να οδηγήσει σε χρήσιμα συμπεράσματα όσον αφορά τις δυνατότητες εξέλιξής του.

6. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Holger Karl, Andreas Willig: *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons Editions, May 2005

[2] John Stankovic: *Wireless Sensor Networks*. Chapter in "Handbook of Real-Time and Embedded Systems", CRC Press, June 19, 2006

[3] Dennis Dauenhauer: *Sensor History Newsletter* at the All Sensors company official site, August 2006

[4] Γεώργιος Μυλωνάς: *Δίκτυα Έξυπνης Σκόνης: Τεχνολογική μελέτη υπαρχόντων συστημάτων και συγκριτική αξιολόγηση πρωτοκόλλων για αποδοτικό εντοπισμό και διάδοση πληροφορίας. Ανάπτυξη και λειτουργία πειραματικού δικτύου.* Διπλωματική εργασία, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, 2003

[5] V.S. Hsu, J.M. Kahn, K.S.J. Pister: *Wireless Communications for Smart Dust.* Electronics Research Laboratory Memorandum Number M98/2, 1998

[6] Jamal N. Al-Karaki, Ahmed E. Kamal: *Routing Techniques in Wireless Sensor Networks: A Survey.* 2002

[7] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin: *Directed Diffusion: A scalable and robust communication paradigm for sensor networks.* In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MOBICOM), Boston, Massachusetts, USA, pages 56-67, August 2000

[8] I. Chatzigiannakis, P. Spirakis, S. Nikoletseas: *Efficient and Robust Protocols for Local Detection and Propagation in Smart Dust Networks*, accepted in the ACM/Baltzer Mobile Networks and Applications (MONET) Journal, Special Issue on Algorithmic Solutions for Wireless, Mobile, Ad Hoc and Sensor Networks, in MONE 10 (1), pp. 133-149, 2005

[9] I. Chatzigiannakis, T. Dimitriou, S. Nikoletseas, P. Spirakis: *A Probabilistic Algorithm for Efficient and Robust Data Propagation in Smart Dust Networks*, in the Ad-Hoc Networks Journal, Elsevier, accepted, 2005

[10] C. Efthymiou, S. Nikoletseas, J. Rolim: *Energy Balanced Data Propagation in Wireless Sensor Networks*, in the Wireless Networks (WINET) Journal, Special Issue on "Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks", accepted, appeared as *Article ID*05-NO00006529, Springer Verlag, 2006

[11] I. Chatzigiannakis, A. Kinalis, S. Nikoletseas: *Adaptive Energy Management for Incremental Deployment of Heterogeneous Wireless Sensors*, invited paper to the Theory of Computing Systems (TOCS) Journal, Special Issue on the Seventeenth (17th) Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), accepted, to appear in 2006

[12] I.Chatzigiannakis, A.Kinalis, S.Nikoletseas: *Sink Mobility Protocols for Data Collection in Wireless Sensor Networks*. In Proc. 4th ACM Workshop on Mobility Management and Wireless Access (MOBIWAC'06), Torremolinos, Malaga, Spain, October 2, 2006, pp. 52-59

[13] A. Kinalis, S. Nikoletseas: *Scalable Data Collection Protocols for Wireless Sensor Networks with Multiple Mobile Sinks,* anss, pp. 60-72, 40th Annual Simulation Symposium (ANSS'07), 2007

[14] John Suh: *How to build Wireless Sensor Networks.* Presented for the Crossbow company, 19 September 2006

[15] B. Hull, V. Bychkovsky, K. Chen, M. Goraczko, A. Miu, E. Shih, Y. Zhang, H. Balakrishnan, S. Madden: *CarTel: A distributed mobile sensor computing system.* SensSys 06, 2006

[16] S. Eisenman, N. Lane, E. Miluzzo, R. Peterson, G. Ahn, A. Campbell: *MetroSense project: People-centric sensing at scale.* In Workshop on World-Sensor-Web (WSW 2006), Boulder, October 31, 2006

[17] Ioannis Chatzigiannakis, Georgios Mylonas, Sotiris Nikoletseas: *The Design of an Environment for Monitoring and Controlling Remote Sensor Networks.* In the International Journal of Distributed Sensor Networks (IJSDN), Taylor&Francis. January 26, 2007

[18] Γεώργιος Μυλωνάς: *Σχεδιασμός και Ανάπτυξη ενός γενικού περιβάλλοντος για υλοποίηση εφαρμογών σε ασύρματα δίκτυα αισθητήρων.* Μεταπτυχιακή εργασία, Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής, 2005

[19] J. Schneidman, P. Pietzuch, J. Leddlie, M. Roussopoulos, M. Seltzer, M. Welsh: *Hourglass: An infrastructure for connecting sensor networks and applications.* Harvard Technical Report TR-21-04, 2004

[20] S. Krco, D. Cleary, D. Parker: *Enabling ubiquitous sensor networking over mobile mobile networks through peer-to-peer overlay networking.* Computer communications 28, 1586-1601, 2005

[21] Nancy Lynch: *Distributed Algorithms.* Morgan Kaufmann Publishers, San Mateo, CA, 1996.

[22] Αθανάσιος Σ. Πουλακίδας: *Συμπληρωματικές Σημειώσεις στα Κατανεμημένα Συστήματα.* Έκδοση Πανεπιστημίου Πατρών, 2004

[23] *http://webs.cs.berkeley.edu/tos/*: Official site of the TinyOS operating system, a component-based OS for the Network Sensor Regime

[24] *http://telegraph.cs.berkeley.edu/tinydb/:* Official site of TinyDB, a declarative database for sensor networks

[25] Daniel Brookshier, Juan Carlos Soto: *What is P2P?* From "JXTA: Java P2P Programming", Chapter 1, Sams Publishers, pub. March 22, 2002

[26] Γεώργιος Στριμπάκος, Απόστολος Παπαγεωργίου, Αθανάσιος Μπάμης: *Μια μελέτη ζητημάτων ασφαλείας σε δίκτυα ομότιμων.* ΤΜΗΥΠ, 2006

[27] *http://www.jxta.org*: Official site of the JXTA project open community

[28] *http://platform.jxta.org/nonav/java/api/index.htm/*: JXTA J2SE API Documentation

[29] Brendon Wilson: *JXTA.* Available online by O' Reilly, published by "Que" on June 6, 2002

[30] Sun Microsystems, Inc.: *JXTA v2.3.x: Java Programmer's Guide.* April 7, 2005

[31] Jürgen Nützel, Dirk Michael: *Entwurf von P2P Anwendungen mit Hilfe der JXTA-Technologie.* Hauptseminar, Wintersemester 2002-2003

[32] Emir Halepovic, Ralph Deters: *The costs of using JXTA.* The Third IEEE International Conference on Peer-to-Peer Computing, Linköping, Sweden, 2003

[33] Akhil Arora, Carl Haywood, Kuldip Singh Pabla: *JXTA for J2ME – Extending the Reach of Wireless with JXTA Technology,* White Paper, http://www.jxta.org/project/www/docs/JXTA4J2ME.pdf

ΤΕΛΟΣ