ΠΑΝΕΠΙΣΤΉΜΙΟ ΠΑΤΡΏΝ

ΠΟΛΥΤΕΧΝΙΚΉ ΣΧΟΛΉ

ΤΜΉΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ & ΠΛΗΡΟΦΟΡΙΚΉΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

"Σχεδιασμός & Ανάπτυξη Περιβάλλοντος Αποδοτικής Διαχείρισης & Αναπαράστασης Δεδομένων, σε Ασύρματα Δίκτυα Αισθητήρων "

Συγγραφέας: Χατζηπρίμου Κλεοπάτρα, ΑΜ 623

Μεταπτυχιακό Δίπλωμα Ειδίκευσης: Επιστήμη και Τεχνολογία Υπολογιστών

ΕΠΙΒΛΈΠΩΝ ΚΑΘΗΓΗΤΉΣ:

Π. Σπυράκης

Ευχαριστίες

Η ολοκλήρωση της εργασίας αυτής δε θα ήταν δυνατή χωρίς την καθοριστική συμβολή ενός συνόλου ατόμων, στα οποία οφείλω τις ειλικρινείς ευχαριστίες μου.

Ευχαριστώ τον επιβλέποντα Καθηγητή κύριο Παύλο Σπυράκη για την ανάθεση αυτής της εργασίας, για τον καίριο, συμβουλευτικό ρόλο του και την παροχή των μέσων και των ανέσεων για την εκπόνησής της.

Θα ήθελα να ευχαριστήσω τον ερευνητή κύριο Χατζηγιαννάκη Ιωάννη για τη συνεισφορά στα επιτεύγματα αυτής της εργασίας, την συνεργασία, και τις δημιουργικές ιδέες.

Πρόλογος

Τα δίκτυα αισθητήρων είναι μια εξειδικευμένη κατηγορία κατανεμημένων δικτύων, η οποία τα τελευταία χρόνια έχει συγκεντρώσει το ενδιαφέρον της ερευνητικής κοινότητας, λόγω του ευρύτατου πεδίου εφαρμογών της. Τα δίκτυα αυτά αποτελούνται από συσκευές που διαθέτουν αισθητήρες (sensors) και ενδεχομένως μηχανισμούς δράσης (actuators) και είναι διασκορπισμένες στο χώρο, με δυνατότητα επικοινωνίας μεταξύ τους και επεξεργασίας σε ένα βαθμό της πληροφορίας που διακινούν στο δίκτυο. Στόχος των δικτύων αυτών είναι η παρακολούθηση της εξέλιξης ενός φαινόμενου, ή η ανίχνευση περιβαλλοντικών συνθηκών και η αποστολή των δεδομένων που συλλέγονται σε έναν κεντρικό κόμβο, το κέντρο ελέγχου. Οι δυνατότητες των συσκευών είναι κατά κανόνα περιορισμένες, λόγω του μικρού μεγέθους τους, του χαμηλού κόστους τους και ενίοτε του αναλώσιμου ρόλου τους.

Οι εφαρμογές των δικτύων αυτών ποικίλουν, από εφαρμογές στην επιστήμη γεωργίας, στην πυρανίχνευση και την παρακολούθηση συνθηκών περιβάλλοντος σε κτιριακές εγκαταστάσεις, μέχρι εφαρμογές ανίχνευσης κίνησης των εχθρικών μονάδων σε πεδίο μάχης, ή παρακολούθησης φαινόμενου που εξελίσσεται σε δυσπρόσιτη περιοχή.

Μερικές από τις βασικές προκλήσεις που προκύπτουν σε αυτά τα δίκτυα, λόγω της φύσης τους, είναι η βέλτιστη διαχείριση των ενεργειακών πόρων κάθε κόμβου, η αποδοτική συνδυαστική επεξεργασία του τεράστιου μεγέθους διακινούμενων δεδομένων και η αξιόπιστη δρομολόγηση της πληροφορίας που συλλέγεται προς το κέντρο ελέγχου, καθώς και η δυναμική και αυτόνομη οργάνωση του δικτύου. Η εξέλιξη των δικτύων αισθητήρων σε συνδυασμό με την επιθυμία υλοποίησης περίπλοκων και περισσότερο ολοκληρωμένων εφαρμογών, οδήγησε σε δίκτυα που περιέχουν ετερογενείς κόμβους, όπως για παράδειγμα κόμβους με διαφορετική αρχιτεκτονική, διαφορετικούς αισθητήρες, με διαφορετικές επεξεργαστικές και επικοινωνιακές ικανότητες. Επιπρόσθετα, σε πολλές εφαρμογές διαφαίνεται η ανάγκη για διασύνδεση πολλαπλών δικτύων αισθητήρων, και διάδοση της πληροφορίας που συλλέγουν μέσω του διαδικτύου προς απομακρυσμένους εξυπηρετητές με αυξημένες επεξεργαστικές ικανότητες προς βάσεις δεδομένων, συσκευές PDA ή κινητά τηλέφωνα, σταθμούς εργασίας τελικών χρηστών και άλλες συσκευές. Δημιουργείται με αυτόν τον τρόπο ένα ευρύτερο ετερογενές δίκτυο επικάλυψης (overlay sensor network), η διαχείριση και αποδοτική χρήση του οποίου απαιτεί νέες αρχιτεκτονικές οργάνωσης των

επιμέρους δικτύων, προσαρμοσμένα πρωτόκολλα διάδοσης πληροφορίας και πλατφόρμες που διευκολύνουν την ομαλή λειτουργία των δικτύων αυτών.

Για την πραγμάτωση του αντικειμενικού σκοπού τους, δηλαδή την αποτελεσματική επίβλεψη συγκεκριμένου γεγονότος και την τροφοδότηση των ερευνητικών ομάδων με λεπτομερείς αναφορές με τα καταγραφέντα δεδομένα για περαιτέρω επεξεργασία, τα δίκτυα αισθητήρων απαιτούν μηχανισμούς λογισμικού που θα μεριμνούν για την αποθήκευση, οργάνωση, επεξεργασία και παρουσίαση σε υψηλό επίπεδο της τεράστιας ποσότητας παραγόμενης πληροφορίας που συγκεντρώνουν. Το επιθυμητό λογισμικό χειρισμού δεδομένων πρέπει να υποστηρίζει την ετερογένεια των δικτύων αισθητήρων μεγάλης κλίμακας, αναλαμβάνοντας θέματα χαμηλού επιπέδου που δε θα πρέπει να απασχολούν τον τελικό χρήστη. Λαμβάνοντας υπόψη τις περιορισμένες επεξεργαστικές δυνατότητες των συσκευών αισθητήρων αλλά και την περιορισμένη μνήμη τους, είναι κρίσιμη η ανάπτυξη κεντρικού μηχανισμού, για την ελαχιστοποίηση της απώλειας πληροφορίας κατά τη λειτουργία τους και ταχύ χειρισμό της διακινούμενης πληροφορίας. Τα παραπάνω συμβάλλουν σημαντικά στη μείωση του overhead κατά την επικοινωνία των κόμβων και συνεπώς βελτιώνουν την συνολική απόδοση του δικτύου. Απώτερος σκοπός, είναι η παροχή στην επιστημονική κοινότητα ενός ομοιογενούς εργαλείου που θα διευκολύνει την έρευνα, επιτρέποντας άμεση και ταχεία εξαγωγή συμπερασμάτων από τον τεράστιο όγκο ανεπεξέργαστων δεδομένων που συσσωρεύεται στα ασύρματα δίκτυα αισθητήρων.

Η συνεισφορά της παρούσας μεταπτυχιακής εργασίας στο παραπάνω πεδίο, έγκειται αρχικά στη μελέτη των ήδη διαθέσιμων μοντέλων παρουσίασης υψηλού επιπέδου πληροφορίας που παράγεται στα πλαίσια λειτουργίας συστημάτων δικτύων αισθητήρων προς τους τελικούς χρήστες, με σκοπό την αξιολόγηση και ανάλυση τους. Τα υπάρχοντα μοντέλα επεκτείνονται με την εισαγωγή και υλοποίηση ενός νέου, πιο εύρωστου σχήματος εμφώλευσης δεδομένων δικτύων αισθητήρων που ονομάζεται WiseML. Επιθυμώντας την παρουσίαση μιας νέας πρότασης στο λογισμικό χειρισμού δεδομένων αισθητήρων αναπτύχθηκε η πλατφόρμα EasySense. Το EasySense αξιοποιώντας μεγάλη ποικιλία από νέες τεχνολογίες ανάπτυξης λογισμικού, προσφέρει έναν ευέλικτο, επεκτάσιμο μηχανισμό αποθήκευσης, ανανέωσης και διατήρησης μεγάλου όγκου πληροφορίας, δυνατότητα παρακολούθησης της εξέλιξης διακριτών γεγονότων του δικτύου και αλληλεπίδρασης με το χρήστη. Οι ενδιαφερόμενοι μπορούν να αναζητήσουν δεδομένα του δικτύου με διαφορετικά επίπεδα λεπτομέρειας, σε τρία διαθέσιμα μοντέλα κωδικοποίησης: GraphML, SensorML, WiseML. Βασιζόμενοι στην ευέλικτη υποδομή των peer-to-peer δικτύων, προτείνουμε ένα μοντέλο διαμοιρασμού των υπηρεσιών του EasySense ανάμεσα σε πολλαπλά ετερογενή δίκτυα αισθητήρων στο διαδίκτυο.

Η παρούσα εργασία χωρίζεται σε οκτώ κεφάλαια.

Στο πρώτο κεφάλαιο, γίνεται επισκόπηση της εξέλιξης των ασύρματων δικτύων αισθητήρων, των χαρακτηριστικών τους και των μοντέρνων εργαλείων που χρησιμοποιούνται σήμερα για την ανάπτυξη σχετικών εφαρμογών. Εξετάζονται οι τέσσερις θεμελιώδης συνιστώσες που καθορίζουν τη λειτουργία ενός δικτύου αισθητήρων, αυτές των υλικού, λογισμικού, αλγορίθμων και δεδομένων. Επίσης, τονίζεται η σημαντικότητα της συνεισφοράς και έρευνας σε θέματα δικτύων αισθητήρων μέσω μιας αναλυτική παρουσίασης ποικιλίας εφαρμογών ασύρματων δικτύων αισθητήρων που σκοπό έχουν τη βελτίωση της ποιότητας ζωής μας.

Στο δεύτερο κεφάλαιο, μελετώνται θέματα αναπαράστασης δεδομένων στα περιβάλλονται δικτύων αισθητήρων. Εξετάζεται πληθώρα από συνηθισμένα αλλά και πιο σπάνια σενάρια λειτουργίας, και μέσω αυτών εξάγεται ένα σύνολο απαιτήσεων και σταθερών κατά την αναπαράσταση πληροφορίας που θα διευκολύνει τις επιστημονικές ομάδες κατά την εξαγωγή συμπερασμάτων. Στη συνέχεια παρουσιάζονται τα δύο ήδη υπάρχοντα και επικρατέστερα μοντέλα αναπαράστασης δεδομένων γράφων και αισθητήρων, SensorML και GraphML,που χρησιμοποιούνται στα δίκτυα αισθητήρων, ενώ εισάγεται ένα νέο που εισάγει μεγαλύτερο επίπεδο λεπτομέρειας, το WiseML και επικαλύπτει τα δύο προυπάρχοντα.

Στο τρίτο κεφάλαιο παρουσιάζεται η πλατφόρμα EasySense. Θίγονται θέματα λογικής, αρχιτεκτονικής και υλοποίησης. Εξετάζονται οι συναρτήσεις που υλοποιεί η πλατφόρμα, ο τρόπος μοναδικής αναπαράστασης των διακριτών πόρων του δικτύου, ο μηχανισμός λειτουργίας του, η δομή της βάσης δεδομένων του συστήματος όπως και η γενική δομή του κώδικα.

Στο τέταρτο κεφάλαιο εξετάζεται η τεχνολογία Hibernate, η οποία συνιστά ένα από τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της πλατφόρμας EasySense. Παρουσιάζονται οι λόγοι που επιλέχθηκε, η δομή της, τα πλεονεκτήματα που εισάγει και ο τρόπος λειτουργίας της.

Στο πέμπτο κεφάλαιο εισάγεται το Simple XML Framework, το οποίο χρησιμοποιήθηκε κατά την ανάπτυξη του EasySense και ειδικότερα για τη δημιουργία μηχανισμών συντακτικής ανάλυσης και αυτόματης παραγωγής GraphML, SensorML και WiseML για την εμφώλευση των δεδομένων που αιτεί ο χρήστης. Παρουσιάζονται τα χαρακτηριστικά του framework όπως και μια εισαγωγή στον τρόπο χρήσης των εργαλείων που προσφέρει για την ανάπτυξη εφαρμογών.

Στο έκτο κεφάλαιο αναλύονται τα Java Web Services και η τεχνολογία Jax-WS. Λόγος γίνεται για τα πλεονεκτήματα του, τα τεχνικά χαρακτηριστικά του και το μηχανισμό λειτουργίας του.

Στο έβδομο κεφάλαιο, παρουσιάζεται η λογική των peer to peer συστημάτων. Συζητούνται τα τεχνικά χαρακτηριστικά τους και οι τοπολογίες τους ενώ παρουσιάζονται αναλυτικά τα πρωτόκολλα JXTA και η χρήση τους για την βελτίωση των λειτουργικοτήτων της πλατφόρμας EasySense.

Στο όγδοο κεφάλαιο γίνεται ανασκόπηση της συνολικής προσφοράς της παρούσας μεταπτυχιακής εργασίας, των θεμάτων που αναλύθηκαν και επεκτάθηκαν αλλά και ζητημάτων τα οποία παραμένουν ανοικτά στην επιστημονική κοινότητα.

Τέλος, στο παράρτημα ο αναγνώστης μπορεί να βρει τμήματα πηγαίου κώδικα που αναπτύχθηκε στα πλαίσια υλοποίησης της πλατφόρμας EasySense.

Περιεχόμενα

<u>ΚΕΦΑΛΑΙΟ 1</u>:

ΠΕΡΙΒΑΛΛΟΝΤΑ ΔΙΚΤΥΩΝ ΑΙΣΘΗΤΗΡΩΝ

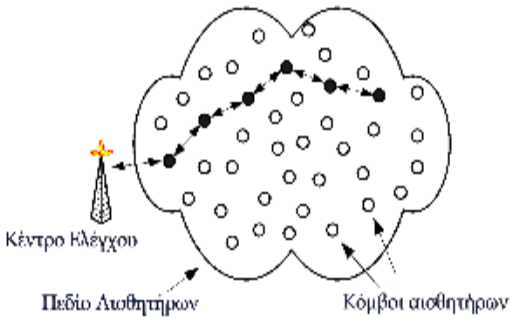
ΚΕΦΑΛΑΙΟ 1

1.1 Εισαγωγή

Μέχρι πρόσφατα, οι επιστημονικές προσπάθειες για μελέτη υπολογιστικών μεθοδολογιών ανάλυσης αποκεντρικοποιημένων σύνθετων συστημάτων υπήρξαν περιορισμένες. Μια ταχέως αναπτυσσόμενη και ιδιαίτερα ελπιδοφόρα περιοχή έρευνας στον τομέα, αποτελούν τα δίκτυα αισθητήρων, τα οποία χαίρουν ολοένα και περισσότερης προσοχής από την επιστημονική και βιομηχανική κοινότητα, προσελκύοντας ερευνητές από όλους τους διαφορετικούς κλάδους της επιστήμης των υπολογιστών, καθώς οι εφαρμογές τους έχουν αποδεδειγμένα τη δύναμη να επιφέρουν επανάσταση στην οικονομία αλλά και στο σύγχρονο τρόπο ζωής. Η έρευνα πάνω στα δίκτυα αισθητήρων παρουσιάζει ιδιαίτερο ενδιαφέρον καθώς θέτει
τις υποδομές για πολλές χρήσιμες, καινοτόμες και εντυπωσιακές εφαρμογές στην κοινωνία μας. Το πεδίο της έρευνας περιλαμβάνει αντικείμενα όπως ο
σχεδιασμός και η υλοποίηση έξυπνων συσκευών με αισθητήρες, ο σχεδιασμός λογισμικού που εκτελείται στις προγραμματιζόμενες έξυπνες συσκευές ως λειτουργικό σύστημα, προτάσεις για αρχιτεκτονικές και μοντέλα δικτύων αισθητήρων, σχεδιασμός πρωτοκόλλων για διάφορα θέματα του δικτύου (δρομολόγηση, διαχείριση ενέργειας, συγχρονισμός, ασφάλεια, διαχείριση πληροφορίας), εξομοίωση της συμπεριφοράς των δικτύων με ειδικό λογισμικό(simulations), σχεδιασμός πλατφορμών δοκιμής των δικτύων (testbeds) και προτάσεις για ειδικό λογισμικό που διευκολύνει τον προγραμματισμό εφαρμογών για τα δίκτυα αυτά.
Πριν προχωρήσουμε σε ανάλυση τεχνικών θεμάτων αναφορικά με τα δίκυα αισθητήρων και τις απαιτήσεις του, θεωρείται απαραίτητη μια εισαγωγή σχετικά με το τι συνεπάγεται και εμπεριέχει ο όρος ¨δίκτυο αισθητήρων¨ στη σύγχρονη βιβλιογραφία.

1.2 Δίκτυα Αισθητήρων

Το κλασσικό μοντέλο ενός δικτύου αισθητήρων περιλαμβάνει ένα σύνολο από κόμβους με αισθητήρες και ενεργοποιητές και ένα κέντρο ελέγχου που καθορίζει τη λειτουργία του δικτύου και στο οποίο συλλέγεται η πληροφορία που παράγουν οι κόμβοι. Το κόστος των κόμβων θεωρείται χαμηλό και οι δυνατότητές του καθενός ξεχωριστά περιορισμένες, ωστόσο μπορούν και συνεργάζονται μεταξύ τους στις λειτουργίες αίσθησης και στην επεξεργασία της διαδιδόμενης πληροφορίας δημιουργώντας ένα έξυπνο κατανεμημένο δίκτυο αισθητήρων.Το κέντρο ελέγχου θεωρείται πως διαθέτει απεριόριστους υπολογιστικούς και ενεργειακούς πόρους και αντιπροσωπεύει τις αρχές εκείνες, που είναι υπεύθυνες για τη ανάπτυξη του δικτύου και ενδιαφέρονται να αποκομίσουν πληροφορίες από αυτό. Στην παρακάτω εικόνα φαίνεται ένα γενικευμένο δίκτυο αισθητήρων.



Οι βασικές λειτουργίες των κόμβων είναι η συλλογή πληροφοριών από τον περιβάλλοντα χώρο μέσω των αισθητήρων τους και η προώθηση πληροφορίας προς το κέντρο ελέγχου. Ο κάθε κόμβος μπορεί να έχει ένα σύνολο από διαφορετικούς αισθητήρες για να αντιλαμβάνεται ποικίλα φαινόμενα που διαδραματίζονται στο άμεσο περιβάλλον του. Επιπρόσθετα ο κάθε κόμβος αναλαμβάνει να δημιουργήσει πληροφορίες σχετιζόμενες με τις μετρήσεις των αισθητήρων που διαθέτει, αλλά και να δρομολογήσει κάθε εισερχόμενη πληροφορία προς το κέντρο ελέγχου. Τα είδη των αισθητήρων που μπορεί να ενσωματώνουν οι κόμβοι των δικτύων αισθητήρων καθορίζουν τις υπηρεσίες που μπορεί να παρέχει το δίκτυο στο οποίο συμμετέχουν και συνεπώς τις εφαρμογές που μπορεί να βασιστούν σε αυτά. Μερικά από τα μεγέθη που μπορεί να ανιχνεύονται από ένα κόμβο με αισθητήρες είναι η θερμοκρασία, η υγρασία, η ατμοσφαιρική πίεση, το επίπεδο ηχητικού θορύβου, τα μαγνητικά πεδία, το επίπεδο φωτεινότητας και η επιτάχυνση ενός αντικειμένου.
Η πληροφορία που ζητάει το κέντρο ελέγχου μπορεί να σχετίζεται :

1) με τον τόπο εκδήλωσης κάποιου φαινομένου, οπότε περιμένει απαντήσεις μόνο από ένα συγκεκριμένο γεωγραφικό τμήμα του δικτύου.

2) με τις μετρήσεις ενός συγκεκριμένου κόμβου, οπότε θα απαντήσει μόνο ο καθορισμένος κόμβος.

3) με τις μετρήσεις για κάποιο φυσικό μέγεθος ή φαινόμενο γενικά, στην οποία περίπτωση θα απαντήσουν όσοι κόμβοι διαθέτουν σχετική πληροφορία.

Σημαντική παράμετρος στη φύση των δικτύων αυτών είναι ότι τυπικά οι κόμβοι των αισθητήρων έχουν περιορισμένες δυνατότητες και συνεπώς απαιτούνται ειδικά πρωτόκολλα για να διαχειριστούν την συμπεριφορά και τη λειτουργία τους. Ο κάθε κόμβος διαθέτει πεπερασμένο ποσό αρχικής ενέργειας, το οποίο καταναλώνεται κατά την αναμονή ή τη μετάδοση μηνυμάτων και κατά την επεξεργασία πληροφορίας. Πολλά πρωτόκολλα αναλαμβάνουν τη διαχείριση της ενέργειας του κόμβου, ώστε να εξασφαλίζονται περίοδοι που ο κόμβος θα λειτουργεί σε κατάσταση μειωμένης κατανάλωσης, πιθανότατα απέχοντας προσωρινά από τη λειτουργία του δικτύου (sleep mode). ΄Οσοι κόμβοι εξαντλούνται ενεργειακά καθίστανται ανενεργοί και δε συμμετέχουν πλέον στην ανίχνευση ή στη διάδοση πληροφορίας. Η εμβέλεια μετάδοσης μηνυμάτων είναι συνήθως περιορισμένη. Αυτό αποτελεί πρόβλημα ειδικά στην περίπτωση δικτύων με πολλαπλούς κόμβους που αναπτύσσονται σε μεγάλη έκταση με την πιθανή παρουσία φυσικών εμποδίων. Σε αυτές τις περιπτώσεις ο κάθε κόμβος δεν μπορεί να επικοινωνεί απευθείας με το κέντρο ελέγχου και απαιτούνται ειδικά πρωτόκολλα δρομολόγησης των μηνυμάτων σε πολλαπλά βήματα (multihop), αποφεύγοντας τα εμπόδια (obstacle avoidance) και λαμβάνοντας υπόψη τις δυναμικές αλλαγές στην τοπολογία λόγω απώλειας των εξαντλημένων κόμβων.

Ιδιαίτερα περίπλοκο σε αυτή την κατάσταση είναι και το θέμα του χρονικού συγχρονισμού των κόμβων, αλλά και του υπολογισμού της θέσης τους για τις περιπτώσεις όπου είναι απαραίτητο οι κόμβοι να γνωρίζουν την τοποθεσία όπου βρίσκονται. Τέλος, οι κόμβοι διαθέτουν μικρό αποθηκευτικό χώρο και σχετικά απλή μονάδα επεξεργασίας. ΄Ετσι υπάρχουν αυστηρά όρια στο μέγεθος του λογισμικού και την πολυπλοκότητα που μπορεί να εκτελείται σε αυτούς, το μέγεθος της ουράς (queue) για τα μηνύματα που λαμβάνονται και αυτά που αναμένουν μετάδοση και το μέγεθος του χώρου προσωρινής αποθήκευσης για την αναμετάδοση δεδομένων που χάθηκαν. Από τα παραπάνω θέματα προκύπτει πως τα πρωτόκολλα και το λογισμικό που εκτελείται στα δίκτυα αισθητήρων οφείλουν να λαμβάνουν υπόψη τις ιδιαιτερότητες της φύσης των δικτύων αισθητήρων.

Πρέπει να είναι ανθεκτικά σε σφάλματα (λόγω απώλειας ή υπολειτουργίας των κόμβων), ευέλικτα ώστε να προσαρμόζονται στη δυναμική τοπολογία των δικτύων, να χειρίζονται το θέμα της ενεργειακής κατανάλωσης και της περιορισμένης εμβέλειας μετάδοσης και να είναι απλά, ώστε να μπορεί ο κάθε κόμβος να τα εκτελέσει.

Άλλο ένα χαρακτηριστικό των δικτύων αισθητήρων είναι ο υψηλός βαθμός ετερογένειας που παρουσιάζουν. Για παράδειγμα, η σύνθεση του δικτύου μπορεί να περιλαμβάνει κινητούς και στατικούς κόμβους. Μπορεί ακόμα και ο κόμβος ελέγχου να είναι κινητός κόμβος που εκτελεί περιπάτους για την περισυλλογή πληροφορίας από τους κόμβους αισθητήρες. Οι κόμβοι αισθητήρων στο δίκτυο μπορεί ακόμα να είναι μοντέλα από διαφορετικές τεχνολογίες. Οι κόμβοι δεν είναι ομότιμοι, δεν διαθέτουν τους ίδιους αισθητήρες και έχουν διαφορετικές δυνατότητες επεξεργαστικές, αποθηκευτικές και επικοινωνιακές. Η σύνδεση με τους υπόλοιπους αισθητήρες μπορεί να είναι ενσύρματη ή ασύρματη. Επιπλέον, περισσότερα του ενός επιμέρους και τοπικά δίκτυα αισθητήρων μπορούν να συνεργάζονται δημιουργώντας ένα ευρύτερο δίκτυο επικάλυψης (overlay sensor network) με αυξημένες δυνατότητες. Μπορεί έτσι ένα δίκτυο με αποκλειστικά κινητούς ή φορητούς κόμβους αισθητήρων να συνεργάζεται με ένα δίκτυο που έχει στατική υποδομή ή πολλαπλά γεωγραφικά διάσπαρτα δίκτυα να συνεργάζονται για να παρέχουν τις υπηρεσίες τους σε ακόμα μεγαλύτερη περιοχή κάλυψης.

1.3 Τομείς Έρευνας στα Δίκτυα Αισθητήρων

Η έρευνα και ανάπτυξη στα θέματα δικτύων αισθητήρων μπορεί να χωριστεί σε τέσσερις θεμελιώδεις κατηγορίες: λογισμικού, υλικού, αλγορίθμων και χειρισμού δεδομένων. Η ταυτόχρονη εξέλιξη των παραπάνω είναι αναγκαία συνθήκη για τη ομαλή λειτουργικότητα των συστημάτων αισθητήρων. Στην παρούσα παράγραφο, θα παρουσιαστούν συνοπτικά τα πιο πρόσφατες εξελίξεις για κάθε κατηγορία.

- ***Το Hardware των nodes:*** Ενώ η πρώτη γενιά του υλικού των αισθητήρων εμφάνιζε απληστία ως προς την κατανάλωση ενέργειας, το διαθέσιμο σήμερα υλικό δεύτερης και τρίτης γενιάς έχει ωριμάσει σημαντικά. Προσφέρει αξιοπιστία, σταθεροποιημένο, υψηλό εύρος ζώνης στα 250 Kbits/sec και

γρήγορους αλλά χαμηλής κατανάλωσης μικρο ελεγκτές. Ευρέος χρησιμοποιούμενες συσκευές είναι τα TelosB, BTnodes , MSB430 με κύριο target group την ερευνητική κοινότητα, ScatterNodes για εμπορική χρήση και η πλατφόρμα iSense που προσφέρεται τόσο για ερευνητικούς σκοπούς όσο και για εμπορικούς. Παρακάτω φαίνεται δείγμα συσκευών:

- ***Το Λογισμικό των συσκευών:*** Μια βασική απαίτηση στα πλαίσια ανάπτυξης λογισμικού των συσκευών είναι η ανάπτυξη middleware. Καθώς ένα λογισμικό middleware αναλαμβάνει την απόκρυψη θεμάτων ετερογένειας του υποκείμενου συστήματος και υποβοηθά στον προγραμματισμό εφαρμογών σε αυτό, μπορεί να γίνει εύκολα αντιληπτή η χρησιμότητά τους στο πεδίο των δικτύων αισθητήρων.

Δημιουργήθηκε για να παρέχει υποστήριξη ως προς την ανάπτυξη, διατήρηση, εκτέλεση WSN εφαρμογών και μεταφορά τους σε σενάρια λειτουργίας του πραγματικού κόσμου. Γωστές προσεγγίσεις είναι οι CORBA και Enterprise Java Beans. Για τις υπάρχουσες προτάσεις middleware λογισμικού έχουν γίνει κάποιες απόπειρες κατηγοριοποίησης τους, παρόλο που τα περισσότερα middleware δεν μπορούν να καταταχθούν σε μία μόνο κατηγορία. Ετσι, μπορούμε να διακρίνουμε τα middleware που υλοποιούνται για τις ανάγκες μόνο μιας συγκεκριμένης εφαρμογής ή μιας πολύ εξειδικευμένης κλάσης εφαρμογών, ενώ αντίθετα υπάρχουν middleware που επιδιώκουν να αποτελέσουν ένα γενικευμένο υπόβαθρο το οποίο θα παρέχει τα βασικά εργαλεία και τις κατάλληλες αφαιρετικές δομές για τον προγραμματισμό μεγαλύτερου εύρους εφαρμογών. Μια άλλη διάκριση μπορεί να γίνει βάσει των διεπαφών που προσφέρει το κάθε λογισμικό στον προγραμματιστή για την προσπέλαση των υπηρεσιών από τα δίκτυα. Υπάρχουν middleware που στην ουσία διατηρούν και ενημερώνουν μια βάση δεδομένων, στην οποία ο προγραμματιστής έχει πρόσβαση με κάποια συγγενική γλώσσα της SQL. Σε άλλη κατηγορία συναντάμε middleware, όπου οι κόμβοι στα δίκτυα εκτελούν κώδικα, ο οποίος έχει γραφτεί για να ερμηνεύεται από την εικονική μηχανή που εκτελείται σε κάθε κόμβο. Τέλος, υπάρχουν middleware στα οποία ο προγραμματισμός και η ρύθμιση της λειτουργίας των κόμβων στα δίκτυα γίνεται με κινητούς πράκτορες λογισμικού που μετακινούνται από κόμβο σε κόμβο επιτελώντας τις κατάλληλες ενημερώσεις στον κώδικα που εκτελείται.

Η διαδικασία προσομοίωσης είναι ακόμα μια σημαντική πλευρά στα πλαίσια ανάπτυξης λογισμικού. Η προσομοίωση ενός πειράματος συνεισφέρει στην εξοικονόμηση πόρων όπως χρόνος, κόστος αλλά και στην αντιμετώπιση ποικίλων χρονικών και γεωγραφικών περιορισμών που εισάγουν τα κατανεμημένα συστήματα. Γνωστά εργαλεία που εξυπηρετούν τους παραπάνω σκοπούς είναι οι προσομοιωτές ns-2 , OMNeT++, OPNET Modeler, GTNetS και YANS. Οι παραπάνω, αποτελούν προσομοιωτές διακριτών γεγονότων, οι οποίοι μοντελοποιούν το network stack, MAC πρωτόκολλα, μοντέλα παρεμβολής, κατανάλωσης ενέργειας κ.λ.π. Ακόμα πιο ακριβείς προσομοιωτές είναι οι **NCTUns** που χρησιμοποιεί το διαθέσιμο TCP/IP πρωτόκολλο**, TOSSIM** που εκμεταλλεύεται τη δομή του λειτουργικού συστήματος TinyOS και Avrora

Άλλη μια κατεύθυνση ανάπτυξης λογισμικού για συστήματα αισθητήρων, είναι και τα εργαλεία απομακρυσμένης ενημέρωσης όπως τα **XNP** το οποίο αποτελεί την πρώτη υλοποίηση remote tool, **Deluge** που χρησιμοποιείται σταθερά σε εφαρμογές βασισμένες στο TinyOS και **Trickle** που χρησιμοποιείται κυρίως για περιβάλλοντα virtual machines**.**

Αναφορικά με θέματα ανάπτυξης λογισμικού αποσφαλμάτωσης, η αποσφαλμάτωση που προσφέρουν τα σύγχρονα εργαλεία αναφέρεται τόσο σε παραμέτρους λογισμικού όσο και γενικότερα του δικτύου (κυρίως θέματα συνεκτικότητας). Τα πιθανά σφάλματα που αντιμετωπίζουν ταξινομούνται στις ακόλουθες τέσσερις κατηγορίες :

i. Σφάλματα στους κόμβους, (**node errors**)
ii. Σφάλματα στις ακμές, (**link errors)**
iii. Σφάλματα στις διαδρομές, **(path errors)**
iv. Καθολικά σφάλματα, που επηρεάζουν σημαντικά τμήματα του δικτύου, **(global problems)**

Πιο συγκεκριμένα παραδείγματα εργαλείων αποσφαλμάτωσης αποτελούν τα ***Sensor Network Management System*** (SNMS) σχεδιασμένο για TinyOs εφαρμογές, ***Sympathy*** σχεδιασμένο για δίκτυα αισθητήρων που επιτρέπουν μοντέλο κεντρικού gateway, ***Memento*** το οποίο εκτός από διάγνωση σφαλμάτων προσφέρει και προειδοποιητικούς συναγερμούς, ***Marionette*** ένα περιβάλλον λογισμικού γραμμένο σε γλώσσα προγραμματισμού nesC, που επιτρέπει εύκολη αλληλεπίδραση με εφαρμογές, και τέλος το ***Sensor Network Inspection Framework*** (SNIF) το οποίο ακολουθεί μια πιο παθητική προσέγγιση, χρησιμοποιώντας ένα ξεχωριστό backbone δίκτυο.

- Τους Αλγορίθμους*:* Από την αλγοριθμική σκοπιά λοιπόν, τα χαρακτηριστικά των ασύρματων δικτύων αισθητήρων υπονοούν απουσία κεντρικής μονάδας ελέγχου, περιορισμένες υπολογιστικές ικανότητες των κόμβων και περιορισμένες δυνατότητες επικοινωνίας μεταξύ των ακμών. Τα παραπάνω, απαιτούν την ανάπτυξη νέων αλγοριθμικών προσεγγίσεων που να συνδυάζουν πρωτόκολλα δικτύων με παραδοσιακούς κεντρικοποιημένους αλγορίθμους δικτύων και νέες μεθόδους κατανεμημένου υπολογισμού. Με άλλα λόγια, η μεγάλη πρόκληση του κλάδου, σχετίζεται με το πώς μπορεί κανείς να χρησιμοποιήσει μια περιορισμένη ποσότητα αυστηρά τοπικής πληροφορίας με τέτοιο τρόπο ώστε να επιτύχει καθολική γνώση των ιδιοτήτων του δικτύου. Πλήθος αλγορίθμων έχουν αναπτυχθεί για την κάλυψη των παρακάτω παραμέτρων των δικτύων αισθητήρων:

- Εκπαίδευση των αισθητήρων και ασφάλεια,
- Δικαιοσύνη στη χρήση των πόρων
- Ενσωματωμένα (embedded) λειτουργικά συστήματα,
- Επεξεργασία σημάτων και πρόσβαση στο μέσο,
- Εντοπισμός του στόχου, καταμέτρηση, εύρεση των συντεταγμένων των αισθητήρων,
- Broadcasting, δρομολόγηση και κάλυψη της περιοχής αισθητήρων,
- Κατασκευή τοπολογίας και διατήρηση,
- Πρωτόκολλα προσανατολισμένα στα δεδομένα και συλλογή των δεδομένων του δικτύου,
- Χρονοδρομολόγηση,
- Εξοικονόμηση ενέργειας και εύρεση νέων πόρων
- Ασφάλεια

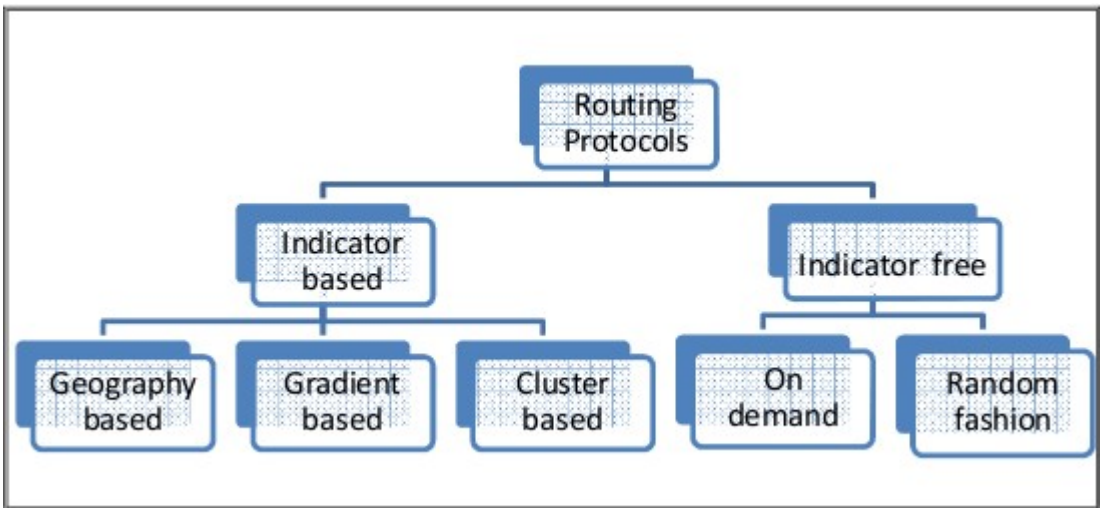
Μερικά από τα πιο γνωστά πρωτόκολλα που έχουν αναπτυχθεί για την ικανοποίηση ορισμένων από τα παραπάνω ζητημάτα περιγράφονται στη συνέχεια.

Αναφορικά με τα πρωτόκολλα δρομολόγησης λοιπόν, μπορούν να χωρισθούν σε δύο
κατηγορίες: πρωτόκολλα βάσει ενδείκτη και πρωτόκολλα χωρίς ενδείκτη. Στα πρωτόκολλα βάσει ενδείκτη υπάρχει πάντα μια αρχική φάση όπου
εφαρμόζεται ένας αλγόριθμος παραγωγής ενδείκτη. Σύμφωνα με τον αλγόριθμο, κάθε
κόμβος παράγει έναν ενδείκτη που βοηθάει στον καθορισμό του δρομολογίου. Στην
κατηγορία πρωτοκόλλων χωρίς ενδείκτη, τα δρομολόγια δημιουργούνται

αυθαίρετα.

Παρατηρούμε ότι στα πρωτόκολλα δρομολόγησης το σημαντικότερο θέμα είναι η συντήρηση της ενέργειας. Η ενέργεια είναι σημαντικότερη ακόμα και από την απόδοση του δικτύου. Όσο η ενέργεια εξαντλείται, το δίκτυο μπορεί να ελαττώσει την ποιότητα των δεδομένων για να μειώσει την απώλεια της ενέργειας και έτσι να αυξήσει τον χρόνο ζωής του δικτύου.

Μια περαιτέρω ταξινόμηση φαίνεται στο ακόλουθο σχήμα:



Στα πρωτόκολλα δρομολόγησης πρέπει να ικανοποιούνται ταυτόχρονα οι παρακάτω απαιτήσεις

1. Πλήρως κατανεμημένα.

2. Προσαρμοστικότητα στις συχνές αλλαγές τοπολογίας του ασύρματου δικτύου.

3. Ο υπολογισμός δρομολογίων και η διαχείριση πρέπει να εμπλέκει τον

4. Ελάχιστο αριθμό κόμβων.

5. Ελάχιστος αριθμός συγκρούσεων πακέτων.

6. Παροχή ικανοποιητικής ποιότητας υπηρεσιών (QoS).

7. Χρήση των ελάχιστων πόρων (όπως το εύρος ζώνης).

Το Location-Aided Routing (LAR) είναι ένα πρωτόκολλο που αντί να πλημμυρίζει όλο το δίκτυο, επιλέγει μία μικρή περιοχή για να διασπείρει τα δεδομένα. Το LAR χρησιμοποιεί πληροφορία θέσης (η οποία μπορεί να είναι ξεπερασμένη την ώρα που χρησιμοποιείται) για να μειώσει το χώρο έρευνας για ένα δρομολόγιο. Ελαχιστοποίηση του χώρου έρευνας σημαίνει λιγότερα μηνύματα εύρεσης δρομολογίου. Συνήθως η πληροφορία θέσης που χρησιμοποιείται στο LAR παρέχεται από ένα σύστημα GPS.

Ο αλγόριθμος λειτουργεί όπως παρακάτω: Καθορίζονται η αναμενόμενη ζώνη (expected zone) και η ζώνη αίτησης (request zone). Ως αναμενόμενη ζώνη θεωρείται μια γεωγραφική περιοχή όπου αναμένεται να περιέχει τον κόμβο προορισμού D σε κάποιο μελλοντικό χρόνο. Έτσι, εάν στον χρόνο t0 ο D βρίσκεται σε κάποιο σημείο και σε χρόνο t1 κινείται με μέση ταχύτητα u τότε μπορούμε να υπολογίσουμε μια τοποθεσία L στην οποία αναμένουμε να βρίσκεται ο D. Εάν ο αρχικός κόμβος S δεν ξέρει την προηγούμενη θέση του D, τότε ως αναμενόμενη ζώνη θεωρείται ολόκληρη η περιοχή που ελέγχουμε. Σε αυτήν την περίπτωση, ο αλγόριθμος LAR μεταπίπτει σε απλό αλγόριθμο πλημύρας μηνυμάτων.

Το GPSR αποτελείται από δύο μεθόδους για να καθορίσει ένα δρομολόγιο: greedy forwarding και perimeter forwarding. Με τη μέθοδο greedy forwarding (άπληστη προώθηση) κάθε πακέτο μαρκάρεται από τον κατασκευαστή του με την τοποθεσία προορισμού του. Έτσι, κάθε ενδιάμεσος κόμβος μπορεί να κάνει μια τοπική βέλτιστη επιλογή για τον επόμενο κόμβο προορισμού του πακέτου. Ειδικότερα, εάν ο κόμβος ξέρει την τοποθεσία του γειτονικού κόμβου, η πιο βέλτιστη λύση είναι να προωθήσει τα πακέτα στον πλησιέστερα γεωγραφικά κόμβο προς τον προορισμό του πακέτου. Η τακτική αυτή συνεχίζεται μέχρι τέλους. Σε περίπτωση που κάποιος κόμβος δεν λάβει ένα σήμα για κάποιο χρονικό διάστημα από γειτονικό κόμβο, υποθέτει ότι αυτός ο κόμβος έχει μετακινηθεί ή απορριφθεί και διαγράφει αυτόν τον κόμβο από τη λίστα του. Το μεγαλύτερο πλεονέκτημα αυτής της μεθόδου είναι η στήριξη μόνο στην γνώση των άμεσα γειτονικών κόμβων. Έτσι, δεν χρειάζεται πολύ πυκνά

κατανεμημένα δίκτυα για να επιτύχει την αποστολή του. Το κύριο μειονέκτημα αυτής της τεχνικής είναι ότι υπάρχουν τοπολογίες όπου εάν χρειάζεται μόνο ένα άλμα για να φτάσει το πακέτο στον προορισμό του, πιθανόν να χρειαστούν περισσότερα εγγύτερα άλματα μέχρι την ολοκλήρωση.

Για την επίτευξη εξοικονόμησης ενεργειακών πόρων, το πρωτόκολλο Low Energy Adaptive Clustering Hierarchy (LEACH)
, χρησιμοποιείται μια τυχαία αλλαγή των αρχηγών ομάδων (cluster-heads) για να διανείμει το φορτίο ενέργειας μεταξύ των κόμβων του δικτύου. Τα βασικά χαρακτηριστικά του LEACH είναι τα παρακάτω:

- Τοπική συνεργασία και έλεγχος για τη δημιουργία και λειτουργία ομάδων.

- Τυχαία εναλλαγή των σταθμών βάσεων των ομάδων ή των αρχηγών ομάδων και

- Τοπική συμπίεση δεδομένων για να μειωθεί η επιβάρυνση στην επικοινωνία.

Η χρήση ομάδων για την μετάδοση δεδομένων προς τον σταθμό βάσης δίνει ένα πλεονέκτημα στους κόμβους αισθητήρες, καθώς απαιτούνται μόνο λίγοι κόμβοι που θα μεταδώσουν σε μεγάλες αποστάσεις. Επιπλέον, το LEACH ανακατανέμει την ενέργεια που καταναλώνεται από τους αρχηγούς ομάδων και μπορεί να εκτελέσει τοπικούς υπολογισμούς σε κάθε ομάδα για να μειώσει τα δεδομένα που μεταδίδονται προς τον σταθμό βάσης.
Το LEACH υποθέτει ότι ο σταθμός βάσης είναι μακριά από τους κόμβους και ότι όλοι οι κόμβοι είναι ομοιογενείς και περιορισμένοι ενεργειακά. Η κύρια εξοικονόμηση ενέργειας προέρχεται από τον συνδυασμό της συμπίεσης δεδομένων και της δρομολόγησης. Έτσι εφαρμόζει περιορισμένες συνεργασίες για να βελτιώσει την κλιμάκωση και την ευρωστία του δικτύου. Επιπλέον, χρησιμοποιεί συγχώνευση δεδομένων για να μειώσει το ποσό της πληροφορίας που μεταδίδεται μεταξύ ενός κόμβου και της δεξαμενής και τέλος, χρησιμοποιεί δυναμικούς μηχανισμούς

επιλογής αρχηγών ομάδας για να αποφύγει την μείωση της ενέργειας των

κόμβων.
Η τυχαία επιλογή αρχηγών ομάδας γίνεται για να αποφευχθεί ο

ενεργειακός θάνατος προκαθορισμένων κόμβων.

- Τα Καταγραφέντα Δεδομένα: Στο παρελθόν οι επιστήμονες υλοποίησαν έναν αριθμό παραδειγμάτων εφαρμογών ασύρματων δικτύων αισθητήρων για να αξιολογήσουν τα ερευνητικά αποτελέσματα τους σε πραγματικού-κόσμου πειράματα, να συλλέξουν δεδομένα για offline ανάλυση των αλγορίθμων ή για να συνδράμουν στο γενικότερο πλαίσιο έρευνας στα ασύρματα δίκτυα αισθητήρων. Στην παρούσα παράγραφο θα παρουσιαστούν ορισμένες γνωστές WSNs εφαρμογές. Μια από αυτές εστιάζει στην παρατήρηση ζώων στο φυσικό τους περιβάλλοντα. Τα projects "*Great Duck Island*" και " *ZebraNet* " χρησιμοποίησαν αισθητήρες για την ανίχνευση περιβαλλοντολογικών συνθηκών, την ακριβή τοποθεσία ή παρουσία ζώων κ.λ.π. Σε άλλα projects όπως τα "*ARGO*", "*Bathymetry*", "*GLACSWEB*" και "*Wisevine*" τα WSNs χρησιμοποιήθηκαν για την περιβαλλοντολογική επίβλεψη, υδάτων ωκεανών και επιτήρηση γεωργικών εκτάσεων. Επίσης, στην Αμερική διεξάγεται σημαντικός αριθμός έργων επιτήρησης . Ενδεικτικά αναφέρονται τα " *A line in the sand project* " και "Palms project" που καταπιάνονται με τον εντοπισμό και καταμέτρηση πολλαπλών, διασκορπισμένων, ετερογενών αντικειμένων. Άλλα ερευνητικά έργα, συγκέντρωσαν δεδομένα τα οποία λειτούργησαν ως βάση για την ανάπτυξη νέων υπολογιστικών μοντέλων, ιδεών και τη βελτιστοποίηση αλγορίθμων. Τα συνολικά δεδομένα, αναλύθηκαν είτε στο χέρι, είτε με τη βοήθεια προσομοιωτή, όπως για παράδειγμα στη μελέτη "The Impact of radio irregularity" όπου εμπειρικά δεδομένα συλλέχθηκαν μέσω της πλατφόρμας MICA2. Τέλος, το "Fence monitoring" χρησιμοποιήθηκε για την αναφορά "multi-hop" γεγονότων.

Παρά την σημαντική τεχνολογική εξέλιξη των συστημάτων δικτύων αισθητήρων και του αναμφίβολου ευεργετικού αντίκτυπου των εφαρμογών τους, υπάρχουν θέματα που παραμένουν ανοικτά και προς επίλυση για την επιστημονική κοινότητα.
Τα πιο σημαντικά από αυτά έχουν να κάνουν με το γεγονός ότι η ανάπτυξη μικρού μεγέθους συσκευών ικανών να εκτελούν λειτουργίες ανίχνευσης, αποθήκευσης, επεξεργασίας δεδομένων αλλά και ασύρματης μεταξύ τους επικοινωνίας αποτελεί ιδιαίτερα χρονοβόρα διαδικασία. Η χρήση μικρής κλίμακας, ειδικού

σκοπού υπολογιστικών συσκευών, συνεπάγεται την δυσκολία ανάπτυξης του κατάλληλου λογισμικού. Μέχρι στιγμής, χρησιμοποιούνται αποκλειστικά υψηλού κόστους custom-made λύσεις. Από αλγοριθμικής άποψης, η κλασσική προσέγγιση αναφέρεται σε μια μονάδα, κεντρικοποιημένης επεξεργασίας. Συνεπώς, στα WSNs απαιτείται πλέον η μελέτη και ανάπτυξη νέων αλγορίθμων, για οργανωμένα μεγάλης κλίμακας, κατανεμημένα συστήματα. Επιπρόσθετα, ο μεγάλος όγκος πληροφορίας που ανιχνεύεται από τις συσκευές αισθητήρων αλλά και το overhead που προκύπτει από τη μεταξύ τους επικοινωνία συντελούν στη συσσώρευση τεράστιων ποσών πληροφορίας. Η αποθήκευση, οργάνωση και αξιολόγηση όλων αυτών των δεδομένων με χρήσιμο αλλά και πρακτικό τρόπο εγείρει νέες προκλήσεις. Αξίζει να σημειωθεί ότι οι παραπάνω παράμετροι είναι ισχυρά αλληλοεξαρτώμενες. Για παράδειγμα, το μικρών διαστάσεων hardware επιβάλλει νέους περιορισμούς και προκλήσεις στο σχεδιασμό του λογισμικού , των απαραίτητων αλγορίθμων αλλά και της διατήρησης δεδομένων.

1.3 Αναδρομή σε Συστήματα Αισθητήρων

Το εναρκτήριο λάκτισμα στην ανάπτυξη και στήριξη της έρευνας σε συστήματα αισθητήρων ξεκίνησε στις ΗΠΑ και χρηματοδοτήθηκε από τον ερευνητικό τομέα του Υπουργείου Άμυνας των Η.Π.Α. (US Defence Advanced Research Project Agency, DARPA) μέσω προγραμμάτων όπως το Smart Dust (Έξυπνη Σκόνη) και το SensIT (Sensor Information Technology – Τεχνολογία της Πληροφορίας από Αισθητήρες), σε συνεργασία με κορυφαία αμερικανικά πανεπιστήμια όπως τα Πανεπιστήμια της Καλιφόρνια στο Berkeley και στο LA. Στόχος των προγραμμάτων αυτών ήταν η ανάπτυξη τεχνολογίας που θα επέτρεπε στους στρατιώτες να “βλέπουν” πίσω από εμπόδια και να αντιλαμβάνονται την απειλή χημικών και βιολογικών όπλων. Ως αποτέλεσμα, η πιο πολυδιάστατη έρευνα σε ασύρματα δίκτυα αισθητήρων πραγματοποιείται επί του παρόντος στην άλλη πλευρά του Ατλαντικού. Πρόσφατες ενδεικτικές πρωτοβουλίες στον τομέα περιλαμβάνουν το κέντρο CENS του Ιδρύματος Επιστημών των Η.Π.Α. με έδρα το Πανεπιστήμιο της Καλιφόρνια στο Λος Άντζελες (UCLA) και το SensorWeb, μια πολυδιάστατη ερευνητική πρωτοβουλία του Τεχνολογικού Ινστιτούτου της Μασαχουσέτης.
Αναφορικά με άλλα σημαντικά έργα, Το “*Trio testbed*”, υπήρξε μια από τις

μεγαλύτερες πλατφόρμες δοκιμής αισθητήρων, σε εσωτερικούς αλλά και εξωτερικούς χώρους που έχει δημιουργηθεί ποτέ. Ο βασικός στόχος ήταν η ανάπτυξη, επίδειξη και αξιολόγηση ενός μεγάλης κλίμακας συστημάτων αισθητήρων για την εξυπηρέτηση πολλαπλών γενικών εφαρμογών καταμέτρησης ετερογενών αντικειμένων και εντοπισμού. Αποτελούνταν από 557 αισθητήρες οι οποίοι λειτουργούσαν με ηλιακή ενέργεια, 7 gateways και ένα καθολικό server. Το όλο πείραμα κάλυπτε σχετικά μεγάλη περιοχή ενώ τέθηκε σε λειτουργία τους τελευταίους μήνες του έτους 2005. Το testbed δεν ήταν ανοικτό στην κοινωνία της δημόσιας έρευνας καθώς εξυπηρετούσε μόνο συγκεκριμένου σκοπού εφαρμογές. Στη συνέχεια, το "*MoteLab*" ήταν ένα εσωτερικού χώρου σύστημα αισθητήρων που αναπτύχθηκε στο πανεπιστήμιο του Harvard και ήταν ανοικτό προς το δημόσιο κοινό. Στην τρέχουσα εκδοχή του, διαθέτει 190 Tmote Sky συσκευές αισθητήρων. Όλοι οι κόμβοι είναι καλωδιομένοι με προγραμματιστικά boards, επιτρέποντας άμεσο επαναπρογραμματισμό και επικοινωνία. Το έργο αυτό σχεδιάστηκε βάση της λογικής ότι πρέπει να είναι προσβάσιμο και εύκολο στη χρήση από άλλους ερευνητές έτσι ώστε πολλές επιστημονικές ομάδες να μπορούν να πειραματιστούν σε θέματα μεγάλης κλίμακας δικτύων αισθητήρων. Επιπλέον παρέχει ένα web-based interface για προγραμματισμό, αποσφαλμάτωση και πρόσβαση δεδομένων σχετιζόμενων με το δίκτυο αισθητήρων.Έπειτα, το "*TWIST*" είναι επίσης τοποθετημένο σε εσωτερικό χώρο του τεχνικού πανεπιστημίου του Βερολίνου, και εκτείνεται σε πολλούς διαφορετικούς ορόφους του. Ο συνολικός αριθμός κόμβων αισθητήρων που χρησιμοποιούνται αγγίζει τους 200 και το υλικό τους είναι κατηγορίας Tmote Sky και EyesIFXv2. Χαρακτηρίζεται από ετερογένεια hardware. Το έργο προσφέρει τρία επίπεδα ιεραρχίας: servers, απλούς nodes και super-nodes με επιπλέον επεξεργαστικές ικανότητες. Το έργο είναι ανοικτό σε μόνο σε εγγεγραμμένους χρήστες. Το "TutorNet" χρησιμοποιεί επίσης μια τριών επιπέδων τοπολογία με servers, gateway stations και απλούς κόμβους οι οποίοι χαρακτηρίζονται από τη δυνατότητα USB σύνδεσης με τους gateway stations. Κάθε σταθμός μαζί με έναν αριθμό αισθητήρων σχηματίζουν ένα cluster. Ανάμεσα στις παρεχόμενες υπηρεσίες συγκαταλέγεται και ο απομακρυσμένος προγραμματισμός των κόμβων. Εξουσιοδοτημένοι χρήστες μπορούν να συνδέονται στους εξυπηρετητές και να χρησιμοποιούν εργαλεία του command-line για τον έλεγχο των κόμβων του testbed. Ο συνολικός αριθμός αισθητήρων αγγίζει τους 104 ενώ το έργο είναι ανοικτό αποκλειστικά σε εγγεγραμμένους χρήστες. Ακόμα αξίζει να αναφερθεί το "*Intel SensorNet*" όπου χρησιμοποιήθηκαν 100 τύπου MicaZ συσκευές αισθητήρων στα ερευνητικά κέντρα της Intel, του πανεπιστημίου Berkeley. Επιτρέπει δέσμευση πόρων ανάμεσα σε πολλαπλούς χρήστες, οι οποίοι υποβάλουν και εκτελούν τις εργασίες τους στο testbed. Επίσης, το "*Kansei* " αποτελεί ένα ακόμα έργοπου απευθύνεται σε μεγάλης κλίμακας εσωτερικών χώρων δίκτυα αισθητήρων. Σήμερα, διαθέτει 210 κόμβους αισθητήρων και 210 gateway stations, μία για κάθε κόμβο αισθητήρων. Προσφέρει εξειδικευμένες

λειτουργίες στους ερευνητές, όπως διαδικτυακή πλατφόρμα εργασίας μέσω της οποίας πραγματοποιείται η υποβολή των διαφόρων εργασιών στο testbed, visualization της τοπολογίας αισθητήρων, αποσφαλμάτωση και επιτήρηση του συνολικού περιβάλλοντος. Το "*MetroSense – Bikenet*", είναι ένα έργο που δίνει έμφαση στις πτυχές των δικτύων αισθητήρων που σχετίζονται με την κινητή επικοινωνία. Παρόλο που δεν είναι ένα ολοκληρωμένη πλατφόρμα σαν τις προαναφερθέντες, υπάρχουν σήμερα αρκετά μικρής έκτασης δίκτυα αισθητήρων σαν το Bikenef. Βασίζεται και αυτό, σε μια αρχιτεκτονική τριών επιπέδων, με testbed servers, gateway stations και κόμβους αισθητήρων προσαρμοσμένους σε ανθρώπους και οχήματα. Ένα διαδικτυακό interface του έργου είναι δημόσια προσβάσιμο, εξυπηρετώντας την επίβλεψη της προσομοίωσης του testbed και την επεξεργασία των δεδομένων των αισθητήρων. Τέλος, το "*TrueMobile*" (Mobile Emulab wireless sensor net testbed) αποτελεί επέκταση του εξαιετικά δημοφιλούς *EmuLab* testbed, για ασύρματα ad-hoc δίκτυα. Το κινητό αυτό testbed, καλύπτει σήμερα συνολική περιοχή 60 τετραγωνικών μέτρων και είναι τοποθετημένο σε εσωτερικό χώρο, περιλαμβάνοντας 6 ασύρματα κινητά ρομπότ και 25 Mica2 συσκευές σειριακά προγραμματιζόμενες.

1.4 Εφαρμογές των δικτύων αισθητήρων

Το μεγάλο πλήθος εφαρμογών των δικτύων αισθητήρων, αποδίδεται κυρίως στο ότι θεωρητικά το κόστος των ίδιων των συσκευών στο εμπόριο είναι αρκετά χαμηλό και συνεπώς η ανάπτυξη (deployment) πολλών κόμβων σε ένα πεδίο και η συντήρηση (maintainance) του δικτύου μπορούν να γίνουν με σχετικά μικρό προϋπολογισμό. Επίσης, σημαντικό πλεονέκτημα αποτελεί το ότι μπορούν να αυτοοργανώνονται σε δίκτυα (self-configured) και να λαμβάνουν μετρήσεις σε όλη την έκταση που καλύπτει το δίκτυο και με ακρίβεια, ειδικά καθότι μπορούν και συνεργάζονται για να υπολογίσουν ακριβέστερα μια μέτρηση, που δεν ήταν πριν εφικτή. Η αρχιτεκτονική ενός περιβάλλοντος ανάπτυξης εφαρμογών και οι υπηρεσίες που θα πρέπει να παρέχει σε ένα προγραμματιστή πρέπει να προσανατολίζονται, ώστε να καλύπτουν αυτό το πλαίσιο. Το πεδίο των εφαρμογών είναι ευρύ αλλά, ωστόσο, άκρως ανομοιογενές και ποικιλόμορφο.

Η κάθε διαφορετική εφαρμογή έχει ιδιαίτερα χαρακτηριστικά, στόχους και λειτουργικότητα (scope), τα οποία προκύπτουν από τις απαιτήσεις της σχετικά με :

1) τα είδη των αισθητήρων και άλλων βοηθητικών συσκευών δικτύου που θα χρησιμοποιηθούν,

2) το βαθμό ακρίβειας των μετρήσεων που λαμβάνονται,

3) τη συχνότητα λήψης πληροφορίας (περιοδικά, κατά απαίτηση ή μόνο κατά την ανίχνευση έκτακτων συνθηκών),

4) τον τρόπο διάταξης των αισθητήρων και των συσκευών του δικτύου γενικότερα,

5) την κλίμακα και έκταση που θα καλύπτει το δίκτυο,

6) την πυκνότητα των κόμβων του δικτύου.

7) την παρουσία ενεργοποιητών (actuators) που θα αναλαμβάνουν δράση σύμφωνα με
συγκεκριμένες λογικές συνθήκες.

8) την παρουσία κινούμενων κόμβων, όπως κινητά τηλέφωνα ή αισθητήρες ενσωματωμένοι σε αυτοκίνητα (mobility),

9) την επιλογή κατάλληλων πρωτοκόλλων επικοινωνίας, δρομολόγησης και επεξεργασίας της πληροφορίας

10) το βαθμό μακροζωίας του δικτύου και ρύθμιση της διαχείρισης των ενεργειακών
πόρων του,

11) την ανάγκη για διασφάλιση QoS παραμέτρων όπως :

α) παράδοση πληροφορίας σε πραγματικό χρόνο (real-time),

β) ελαχιστοποίηση σφαλμάτων,

γ) αξιόπιστη λειτουργία,

δ) ασφάλεια στη διακίνηση των δεδομένων,

Οι κυριότεροι τομείς όπου μπορούν να βρουν εφαρμογή και να συνεισφέρουν τα δίκτυα αισθητήρων είναι αυτοί της υγείας, μέριμνας, ποιότητας ζωής του ατόμου, στρατιωτικός, πρόληψης καταστροφών και ατυχημάτων. Η υγεία είναι ένας βασικός τομέας που μπορεί να επωφεληθεί από τα δίκτυα αισθητήρων. Ο MSens , που είναι παράδειγμα μιας τυπικής εφαρμογής για την υγεία προσαρτώνται αισθητήρες στο σώμα ή σε ρούχα ασθενών, με σκοπό τη συνεχή παρακολούθηση της κατάστασής τους (πίεση του αίματος, τη θερμοκρασία του σώματος) και των κινήσεών τους, την ανίχνευση κρίσιμων περιστατικών και την άμεση ειδοποίηση ιατρικού προσωπικού. Οι αισθητήρες αυτοί σε συνδυασμό με μια κεντρική βάση δεδομένων μπορούν να παρέχουν και ένα ιστορικό από πρόσφατες ασθένειες ή συμπτώματα ενός ατόμου καθώς και την τρέχουσα φαρμακευτική αγωγή που ενδέχεται να ακολουθεί αυτό, ώστε να διευκολύνεται η διάγνωση της κατάστασής του ή η κατάλληλη τροποποίηση της πορείας της θεραπείας του. Το δίκτυο περικλείει και τα κινητά τηλέφωνα των ασθενών, που αφενός επικοινωνούν με τους ασύρματους αισθητήρες του ατόμου και αφετέρου μέσω της υπάρχουσας υποδομής κινητών δικτύων επικοινωνούν και με την κεντρική βάση. Η ιδέα της διεισδυτικής τεχνολογίας συνίσταται στην δημιουργία περιβαλλόντων στα οποία βρίσκονται διάσπαρτοι πολυάριθμοι κόμβοι δικτύων αισθητήρων, διακριτικά τοποθετημένοι, ώστε να μην παρεμποδίζουν ή να ενοχλούν τους ανθρώπους-χρήστες. Στην εν λόγω εφαρμογή, η εταιρεία Elite-Care, δημιούργησε ένα σύμπλεγμα διαμερισμάτων αντί ενός τυπικού γηροκομείου, για να διαμένουν οι ηλικιωμένοι φιλοξενούμενοι. Οι εγκαταστάσεις διαθέτουν ένα δίκτυο από σταθερούς αισθητήρες που είναι τοποθετημένοι τόσο στα διαμερίσματα όσο και σε πολυσύχναστους χώρους, και από φορητούς αισθητήρες που φέρουν οι ηλικιωμένοι υπό τη μορφή περιβραχιόνιου. Οι αισθητήρες συγκεντρώνουν πληροφορία σχετικά με τα ζωτικά σημεία των ηλικιωμένων και τις κινήσεις τους. Ανιχνεύουν έτσι άμεσα ποιοι ηλικιωμένοι χρειάζονται άμεση φροντίδα από το ιατρικό προσωπικό. Η πληροφορία που συγκεντρώνεται αποθηκεύεται σε βάση, όπου αποτυπώνονται η χρονική πρόοδος των τάσεων και συνηθειών του κάθε ατόμου και φυσικά η πορεία της υγείας του.

Το δίκτυο επίσης διαθέτει ενεργοποιητές (actuators) που χρησιμοποιούνται για να ανάβουν αυτόματα φωτισμό που θα καθοδηγεί τους κατοίκους προς το μπάνιο στο σκοτάδι. Αυτό που πετυχαίνει η εφαρμογή είναι να εκμεταλλεύεται το ασύρματο δίκτυο, ώστε η φροντίδα των ηλικιωμένων να παρέχεται χωρίς να χρειάζεται περιττές επισκέψεις και παρεμβάσεις από ιατρικό προσωπικό, και τελικά οι ηλικιωμένοι να ζουν όσο το δυνατόν φυσιολογική ζωή. Αρκετές εφαρμογές πάλι, στοχεύουν στη βελτίωση της ποιότητας ζωής του ατόμου. Παράδειγμα τέτοιας εφαρμογής είναι τα `έξυπνα΄ σπίτια, όπου η κεντρική ιδέα είναι η ενσωμάτωση ασύρματων κόμβων με αισθητήρες στις οικιακές συσκευές (τηλεόραση, θερμαντικές πηγές, ψυγείο, βίντεο, φούρνος, πλυντήριο) με στόχο τη δημιουργία ενός πιο άνετου και εκσυγχρονισμένου οικιακού περιβάλλοντος.

Με τον τρόπο αυτό και με τη βοήθεια κάποιων πρόσθετων ενεργοποιητών μπορεί να επιτευχθεί αυτόματη προσαρμογή της θερμοκρασίας του σπιτιού, της φωτεινότητας ή του θορύβου ενός δωματίου σε επιθυμητά επίπεδα και ανάλογα με την παρουσία ή όχι κάποιου ατόμου στο σπίτι και της δραστηριότητας που εξασκεί. Διαμορφώνοντας ένα δίκτυο μέσα στο σπίτι, το οποίο θα μπορεί να επικοινωνεί με εξωτερικά δίκτυα από το Internet, γίνεται επιπλέον εφικτή η διαχείριση των οικιακών συσκευών είτε από πρόσωπο εντός της οικίας ή εξ΄ αποστάσεως με περισσότερη ευκολία.

Μια παρόμοια ιδέα ακολουθείται και στις εφαρμογές `σκεπτόμενων εγκαταστάσεων με γραφεία΄, οι οποίες αποσκοπούν στην κατά το δυνατόν καλυτέρευση των εργασιακών συνθηκών του ατόμου με άμεση συνέπεια την ευδιαθεσία του και την αύξηση της παραγωγικότητας του. Μια τέτοια εφαρμογή, που χρησιμοποιεί ένα σχετικά μεγάλο δίκτυο από 100 κόμβους, το οποίο καλύπτει 50 δωμάτια γραφείων σε ένα κτίριο. Οι αισθητήρες και ενεργοποιητές μετρούν και ρυθμίζουν τις περιβαλλοντολογικές συνθήκες σε
κάθε δωμάτιο, ενώ επιτελούν και λειτουργίες που σχετίζονται με την ασφάλεια του κτιρίου και τον χειρισμό επειγουσών καταστάσεων.

Επίσης, πολλές είναι οι εφαρμογές των δικτύων αισθητήρων που εξυπηρετούν στην πρόληψη φυσικών καταστροφών με έμφαση κυρίως στην έγκαιρη ειδοποίηση για την αντιμετώπισή τους. Η άμεση ανίχνευση πυρκαγιών και ο υπολογισμός της πιθανότητας πλημμύρας ενός ποταμού ή έκρηξης ενός ηφαιστείου είναι μερικές τέτοιες εφαρμογές. Τυπικά τα δίκτυα για αυτές τις εφαρμογές αναπτύσσονται στη φύση, σε μεγάλη κλίμακα έκτασης και μένουν για καιρό ανεπίβλεπτα. Οι κόμβοι θεωρούνται αναλώσιμοι υπό την έννοια ότι το δίκτυο θα πρέπει να συνεχίσει να λειτουργεί ακόμα και αν εξαντληθούν ή υπολειτουργήσουν κάποιοι κόμβοι, ενώ υπάρχει και σοβαρό ενδεχόμενο να καταστραφούν εντελώς σε περίπτωση κακοκαιρίας ή κάποιας φυσικής καταστροφής. Συχνά σε τέτοια δίκτυα απαιτείται η χρήση ανανεώσιμων πηγών ενέργειας στους κόμβους και τακτικής συμπλήρωσης του δικτύου με νέους κόμβους, ώστε να διατηρείται κατά μέσο όρο σταθερή η πυκνότητά του σε όλη την έκταση που παρακολουθείται. Σημαντικό ρόλο εδώ έχουν υπηρεσίες όπως η έγκαιρη ανίχνευση κρίσιμων γεγονότων και η γρήγορη μετάδοση σήματος κινδύνου στο κέντρο ελέγχου και η απρόσκοπτη και αξιόπιστη δρομολόγηση της πληροφορίας παρά τα φυσικά εμπόδια και την ενεργειακή εξάντληση μερικών κόμβων. Επίσης, με προτεραιότητα που ποικίλει ανάλογα με την εφαρμογή, έχει νόημα και η περιοδική δειγματοληψία μετρήσεων από το περιβάλλον, που θα αποθηκεύονται σε βάση και θα μελετώνται ως ιστορικό για την αναγνώριση τυπικών μοτίβων και την πρόβλεψη της συμπεριφοράς του παρατηρούμενου φαινομένου.

Οι εφαρμογές των δικτύων αισθητήρων στο στρατιωτικό τομέα είναι επίσης σημαντικές. Είναι άλλωστε γνωστό ότι οι στρατιωτικές εφαρμογές μιας τεχνολογίας συνήθως συνεπάγονται σημαντικούς οικονομικούς πόρους διαθέσιμους για την περαιτέρω έρευνα και ανάπτυξή σε αυτή. Οι κυριότερες εφαρμογές είναι η εποπτεία ενός πεδίου μάχης και η παρακολούθηση της κίνησης εχθρικών μονάδων (tracking). Τα δίκτυα αναπτύσσονται για να καλύψουν περιοχές στρατηγικής σημασίας καθώς και δρόμους που μπορούν να αποτελέσουν διόδους μετακίνησης και πιθανής επίθεσης ή εισβολής για τα εχθρικά στρατεύματα. Ο κεντρικός στόχος είναι η ανίχνευση των δραστηριοτήτων στα εχθρικά στρατεύματα και η έγκαιρη ειδοποίηση των συμμαχικών δυνάμεων. Η πρόκληση είναι η επιβίωση των δικτύων σε αφιλόξενα περιβάλλοντα, η απόκρυψη της λειτουργίας τους από τις εχθρικές δυνάμεις, και η απρόσκοπτη και ασφαλής επικοινωνία με τα συμμαχικά κέντρα ελέγχου παρά τις όποιες προσπάθειες παρεμβολών από τις εχθρικές δυνάμεις.

ΚΕΦΑΛΑΙΟ 2:

ΘΕΜΑΤΑ ΑΝΑΠΑΡΑΣΤΑΣΗΣ ΔΕΔΟΜΕΝΩΝ ΣΕ ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ

2.1 Αναπαράσταση δεδομένων του δικτύου

Τα μοντέλα αναπαράσταση πληροφορίας απεικονίζουν τη δομή, τη συμπεριφορά τη σημασιολογία και τα χαρακτηριστικά του πεδίου ενδιαφέροντος. Παρουσιάζουν μεγάλη ποσότητα πληροφορίας με συμπιεσμένο και ιεραρχικά δομημένο τρόπο. Έτσι προσεγγίζουν το σύνθετο τρόπο που είναι οργανωμένη η ανθρώπινη σκέψη παρέχοντας έναν γρήγορο μηχανισμό εξαγωγής συμπερασμάτων από αδόμητα αρχικά δεδομένα,ιδιαίτερα χρήσιμο στην επιστημονική κοινότητα.
Στα πλαίσια της έρευνας για την ανάπτυξη μοντέλων παρουσίασης υψηλού επιπέδου δεδομένων δικτύων αισθητήρων, κρίνεται αναγκαία η καταγραφή αριθμού αντιπροσωπευτικών σεναρίων λειτουργίας συστημάτων αισθητήρων. Σκοπός είναι η συγκριτική μελέτη και αξιολόγηση των απαιτήσεων τους και των δεδομένων που παράγουν, ώστε να εντοπιστούν οι κοινοί τόποι που θα αντικατοπτρίζουν τη συμπεριφορά της πλειοψηφίας των εφαρμογών που προσφέρουν οι σύγχρονες πλατφόρμες ασύρματων αισθητήρων.
Βάσει του εντοπισμού χαρακτηριστικών σεναρίων λειτουργίας (traces), θα υπογραμμισθούν οι απαιτήσεις της διαδικασίας αναπαράστασης δεδομένων, με σεβασμό στις παραμέτρους χώρου και χρόνου.
Οι παραπάνω πληροφορίες θα χρησιμεύσουν στη δόμηση ομοιογενών μοντέλων αναπαράστασης δεδομένων δικτύων αισθητήρων, προσφέροντας στους ερευνητές ένα εργαλείο επεξεργασίας, μελέτης, υψηλού επιπέδου αποθήκευσης και παρουσίασης μεγάλου όγκου πληροφορίας.

Τα traces δίνονται ως είσοδο σε προσομοιωτές για την εξαγωγή συμπερασμάτων. Πρέπει να εμπεριέχουν υψηλό βαθμό ρεαλισμού κινούμενο στις παρακάτω συνιστώσες:

- Πιο ρεαλιστικά αποτελέσματα με σεβασμό στις πραγματικές παραμέτρους προσομοίωσης (για παράδειγμα θέματα software, hardware και επικοινωνίας των προσομοιωμένων κόμβων του δικτύου

- Πιο ρεαλιστικά σενάρια λειτουργίας. Συγκεκριμένα, αντί να χρησιμοποιούνται πιθανοτικές ή άλλες παρεμφερείς τεχνικές για την προσομοίωση των σχετιζόμενων με τις εφαρμογές παραμέτρων, αξιοποιούνται τα δεδομένα από τα πραγματικές πλατφόρμες αισθητήρων και προσομοιώνονται γεγονότα τα οποία έχουν νόημα για εφαρμογές πραγματικού κόσμου, συνεπώς επιτυγχάνεται

βελτίωση της ποιότητας των αποτελεσμάτων που λαμβάνει τη ερευνητική ομάδα. Ακόμα μια ενδιαφέρουσα τακτική εκτός από την προσομοίωση και μελέτη αναμενόμενων σεναρίων λειτουργίας και χρήσης ασύρματων δικτύων αισθητήρων, είναι η χρήση απρόβλεπτων σεναρίων, με σκοπό τον εντοπισμό των απαιτήσεων ευρωστίας των υπό μελέτη συστημάτων ακόμα και στις χειρότερες, εξαιρετικά σκληρές συνθήκες λειτουργίας.

Ο σχεδιασμός όλων αυτών των σεναρίων θα επιφέρει τελικά μια εποπτική αναφορά, σχετικά με τις παραμέτρους που κάθε πλατφόρμα αισθητήρων απαιτεί.

Τα σενάρια λειτουργίας συστημάτων αισθητήρων που έχουν μελετηθεί, μπορούν να προσομοιωθούν ως μοντέλα που εμπίπτουν σε μια από τις παρακάτω κατηγορίες:

1. Χειρισμός καθολικών οντοτήτων: Σε αυτό το σενάριο αντιστοιχίζεται η μοντελοποίηση του χειρισμού των καθολικών οντοτήτων κάποιας εταιρίας ή γενικότερα κάποιου συγκεκριμένου κτιρίου, ενσωματωμένος σε ένα κεντρικό σταθμό λειτουργίας, όπου όλα τα δεδομένα συσσωρεύονται. Στο σημείο αυτό τα δεδομένα αναλύονται και οπτικοποιούνται. Στην κατηγορία αυτή θέτονται σε λειτουργία πολλαπλά τμήματα των κατανεμημένων συστημάτων αισθητήρων και επικεντρώνεται σε σταθερούς κόμβους αισθητήρων μετρώντας στατικά φαινόμενα. Περιλαμβάνει ακόμα τις ακόλουθες πλευρές ανάπτυξης:

- Εγκατάσταση ασύρματων κόμβων αισθητήρων στα δωμάτια του κτιρίου (π.χ στα γραφεία) για τη συνεχή επίβλεψη συγκεκριμένων παραμέτρων.

- Χρήση πλήθους αισθητήρων σχετιζόμενων με θέματα κατανάλωσης ενέργειας και άλλες σημαντικές παραμέτρους όπως θερμοκρασία, φως, διοξείδιο του άνθρακα

- Χρήση backbone network infrastructure, για την υποστήριξη και καταγραφή των πειραμάτων που διεξάγονται στη συγκεκριμένη περιοχή.

- Χρήση ειδικών μηχανισμών, για την επίβλεψη των ηλεκτρομηχανικών συσκευών.

- Καταγραφή τακτικής ημερήσιας δραστηριότητας, με σκοπό τον εντοπισμό και καταγραφή ρεαλιστικών δεδομένων εισόδου χρηστών και περιβάλλοντος.

- Χρήση ελεγχόμενων, δυναμικών εισόδων και εξωτερικών παραμέτρων στο

σύστημα με σκοπό τον επηρεασμό των λειτουργιών του. Για παράδειγμα εισαγωγή κινητών ανθρώπινων χρηστών ή ρομπότ μέσα στις κτιριακές εγκαταστάσεις.

2. Καταμέτρηση αντικειμένων στα οριακά σημεία του συστήματος: Στην κατηγορία αυτή, μοντελοποιούνται συστήματα ως εταιρίες παροχής υπηρεσιών ασφαλείας όπου όταν άνθρωποι ή αντικείμενα παραβαίνουν ένα συγκεκριμένο κατώφλι ή επιτρεπόμενο όριο εντοπίζονται και καταγράφονται.
Το πλεονέκτημα που προσδίδουν τα κατανεμημένα testbeds είναι η δυνατότητα χρήσης πολλαπλών τεχνολογιών εντοπισμού και ανίχνευσης για την εκτίμηση της αποδοτικότητας των συγκεκριμένων εργασιών. Και σε αυτή την εφαρμογή εμφανίζεται μέτριου επιπέδου πολυπλοκότητα και χρησιμοποιούνται σταθεροί αισθητήρες, παρόλο που ορισμένα φαινόμενα εμφανίζουν κινητικότητα. Η κατηγορία αυτή εγείρει νέα ζητήματα, τα οποία δεν χρειάστηκε να αντιμετωπιστούν προηγουμένως, όπως παρουσιάζονται παρακάτω:

- Κινητικότητα συσκευών*:* Η χρήση συσκευών που "ταξιδεύουν" κατά μήκος του δικτύου προκαλεί αριθμό προβλημάτων σχετικών με την συνεχώς μεταβαλλόμενη τοπολογία του δικτύου. Ανάλογα με την ταχύτητα μετακίνησης των συσκευών, οι στατικοί αλγόριθμοι ενδεχομένως μπορούν να προσαρμοστούν στα νέα δεδομένα, ωστόσο νέα πρωτόκολλα πρέπει να σχεδιαστούν. Επιπλέον, από τη σκοπιά της προσομοίωσης και αναπαράστασης τίθεται το ερώτημα του τρόπου αναπαράστασης της κινητικότητας των συσκευών. Βέβαια, μεγάλη διασπορά συνεπάγεται την παραγωγή πολύ μεγάλων αρχείων εξόδου με αποτέλεσμα την επιβράδυνση της διαδικασίας προσομοίωση και την παροχή ανεπιθύμητου επιπέδου λεπτομέρειας.

- Ζητήματα ασφαλείας: Εξορισμού, η φύση της εφαρμογής είναι ιδιαίτερα επιρρεπής σε επιθέσεις ασφαλείας. Ένας εισβολέας δύναται να παρέμβει σε ευρεία περιοχή λειτουργικοτήτων, όπως επίθεση σε ανεξάρτητους κόμβους, παρεμβολή ψευδών πληροφοριών στο δίκτυο, παρακώλυση του συνολικού ασύρματου δικτύου κ.α

- Ευρωστία: Καθώς το σενάριο αυτό αντιμετωπίζει θέματα ασφαλείας, η παράμετρος ευρωστίας του συνολικού συστήματος είναι κριτικής σημασίας.

3. Παρακολούθηση αθλητικών γεγονότων: Σε αυτή την περίπτωση, είναι επιθυμητή η καταμέτρηση και εξυπηρέτηση των συμμετεχόντων σε κάποιο

μεγάλης έκτασης αθλητικό γεγονός, για παράδειγμα ένας μαραθώνιος. Το σενάριο αυτό εμφανίζει αυξημένη πολυπλοκότητα καθώς περιλαμβάνει όχι μόνο κινητές συσκευές αισθητήρων αλλά επειδή εξ' ορισμού είναι μια πειραματική κατάσταση με αυξημένες παραμέτρους κινητικότητας. Αυτό το είδος πειραματικού σεναρίου εμφανίζει τα ακόλουθα χαρακτηριστικά:

- Διαρκείς αλλαγές τοπολογίας δικτύου: Η μεγάλη πλειοψηφία των συσκευών αυτού του τύπου δικτύων είναι κινητές, έτσι όσοι αλγόριθμοι προσπαθούν να χτίσουν δέντρα δρομολόγησης θα αποτύχουν. Συνήθως υιοθετείται η προσέγγιση αποθήκευσης χρονοσημασμένων δεδομένων και μεταφορά τους στους basestations όταν είναι διαθέσιμα, παρόλο που και σε αυτή την περίπτωση εμφανίζονται δυσκολίες. Για παράδειγμα οι κόμβοι δεν είναι ομοιόμορφα κατανεμημένοι, αλλά το δίκτυο χωρίζεται σε ομάδες με υψηλή ή χαμηλή πυκνότητα αισθητήρων. Επιπλέον ο ανταγωνισμός για την πρόσβαση στο μέσο επικοινωνίας προβλέπεται δύσκολος στο χειρισμό, ειδικά για περιοχές υψηλής περιεκτικότητας συσκευών. Τέλος, άλλη μια περιοριστική παράμετρος είναι το σημαντικό κόστος συντήρησης των basestations.

- Έγκαιρη παράδοση μηνυμάτων: Είναι πιθανότατα αδύνατη η παραλαβή μηνυμάτων σε πραγματικό χρόνο, από όλες τις συσκευές του δικτύου. Οι κόμβοι αισθητήρων θα αποθηκεύουν χρονοσημασμένα δεδομένα και θα τα παραδίδουν με τον καλύτερο δυνατό τρόπο στα διαθέσιμα gateways . Ωστόσο, λόγω περιορισμών στην επικοινωνία, υπάρχει συγκεκριμένο αυστηρό άνω όριο στη συχνότητα αναφορών.

Τα παραπάνω traces, όπως έγινε κατανοητό, επεξεργάζονται διαφορετικά χαρακτηριστικά και πτυχές των απαιτήσεων και αποτελεσμάτων των συστημάτων ασύρματων δικτύων αισθητήρων , συνεπώς δρουν ως οδηγοί για την περιγραφή όλων των σχετιζόμενων με δεδομένα δικτύων εφαρμογών, καθώς και για την περιγραφή των ίδιων των δεδομένων του WSN και το σύνολο των πιθανών τύπων τους. Ωστόσο, ανεξάρτητα από τους τύπους των δεδομένων, είναι ιδιαίτερα σημαντική η παρατήρηση και καταμέτρηση των τρόπων συμπεριφοράς των δεδομένων αναφορικά με παραμέτρους που σχετίζονται με χρόνο, χώρο, διευθυνσιοδότηση, περιοδικότητα, στατικότητα, τοπικότητα, συχνότητα δειγματοληψίας, ορθότητα, επάρκεια, ένταση κ.α Από αυτή την οπτική, σημαντική είναι και η ανάπτυξη μηχανισμού που θα περιγράφει μαζί με το δεδομένο και το ακριβές περιβάλλον όπου αυτά προέκυψαν και καταγράφηκαν.

2.2 Μοντέλα Αναπαράστασης δεδομένων του δικτύου

Σε αυτή την ενότητα θα παρουσιαστούν τα διαθέσιμα μοντέλα αναπαράστασης δεδομένων δικτύων ασύρματων αισθητήρων.

2.2.1 GraphMLFormat

Η επιλογή του GraphML format , βασίστηκε μερικώς στο γεγονός ότι ένας παραδοσιακός τρόπος περιγραφής ενός δικτύου, τουλάχιστον από τη σκοπιά της επιστήμης των υπολογιστών, είναι με τη χρήση της έννοιας των γράφων. Επιπρόσθετα η GraphML είναι παράγωγο της XML, εύκολη στη χρήση, επεκτάσιμη και αρκετά γενική ώστε να μπορεί να λειτουργήσει ως κοινός παρονομαστής για όλα τα είδη λειτουργικοτήτων που παράγουν, αποθηκεύουν ή επεξεργάζονται γράφους και άρα μπορούν εύκολα να αντιστοιχισθούν σε όλα τα είδη σεναρίων δικτύων αισθητήρων. Σε γενικές γραμμές, η GraphML χρησιμοποιείται για να περιγράφει γράφους. Σχεδιάστηκε έτσι ώστε να είναι εύκολα επεκτάσιμη, γενικής φύσης, παρέχοντας δυνατότητες ορισμού και περιγραφής πρόσθετων χαρακτηριστικών των κόμβων αισθητήρων, αλλά και των ασύρματων ακμών που σχηματίζονται μεταξύ τους. Χρησιμοποιώντας GraphML, η τοπολογία ενός γράφου και απλά χαρακτηριστικά των συστατικών του γράφου μπορούν να παρουσιαστούν κατά αύξουσα σειρά. Για να αποθηκευτούν πιο σύνθετα δεδομένα στα πλαίσια του έργου, η GraphML πρέπει να επεκταθεί. Πιο συγκεκριμένα, είναι απαραίτητο να οριστούν επιπλέον γνωρίσματα αναφορικά με τις ακμές και τους κόμβους, τα οποία θα περιέχουν χρήσιμη πληροφορία σχετικά με την κατάσταση μέσα στις πειραματικές πλατφόρμες ασύρματων αισθητήρων, στα πλαίσια του κάθε, υπό εξέταση πειράματος. Στη συνέχεια της ενότητας, θα παρουσιαστεί ο τρόπος περιγραφής της τοπολογίας ενός ασύρματου δικτύου αισθητήρων και των γεγονότων που σχετίζονται με αυτό, μέσω GraphML format. Απαραιτήτως, για τους σκοπούς ενός network trace, χρειάζεται κανείς την απεικόνιση των χωροχρονικών παραμέτρων στα πλαίσια των γεγονότων επικοινωνίας, υλικού και λογισμικού. Η λίστα των δικτυακών χαρακτηριστικών που μπορούν να καταγραφούν είναι πρακτικά σχεδόν άπειρη, έτσι στην παρούσα εργασία επιλέγεται να παρουσιαστούν τα βασικότερα στοιχεία που μπορούν εύκολα να χρησιμοποιηθούν για να επανα-παραχθούν οι απαραίτητες θεμελιώδης συνθήκες για την κατηγοριοποίηση της απόδοσης του συστήματος και της ορθότητας των

τελικών αποτελεσμάτων.

2.2.1.1 Περιγραφή Τοπολογίας

Οι χωρικές παράμετροι του δικτύου είναι καλά ορισμένες εξορισμού από την GraphML, αντιστοιχίζοντας απλά κόμβους του δικτύου σε κόμβους γράφου και ακμές του δικτύου σε ακμές του γράφου. Η βάση αυτή, επιτρέπει την προσθήκη των απαραίτητων επεκτάσεων για την λεπτομερή περιγραφή των συστημάτων.

Σχετικά με τους κόμβους των δικτύων, ορίζονται οι ακόλουθες παράμετροι:

- ***isGateway:*** Το συγκεκριμένο χαρακτηριστικό μπορεί να εφαρμοστεί είτε σε κόμβους του δικτύου που χρησιμοποιούνται ως base stations είτε σε απλούς κόμβους. Σαν γνώρισμα, παρέχει τη βασική διάκριση αναφορικά με τη σημαντικότητα συγκεκριμένων κόμβων και το γενικότερο ρόλο τους στη συνολική λειτουργία του δικτύου αισθητήρων. (για παράδειγμα τα σημεία επικοινωνίας του δικτύου αισθητήρων με τον υπόλοιπο κόσμο).

- ***Location***: ορισμένο συναρτήσει των παραγόντων μήκους,πλάτους και ύψους (στο ίδιο format που χρησιμοποιείται και στα GPS). Εναλλακτικά, η τοποθεσία μπορεί να εκφραστεί και με τη βοήθεια των συντεταγμένων ***x, y, z.*** Στη δεύτερη περίπτωση ένα επιπλέον γνώρισμα, αυτό του ορισμού της αρχής των αξόνων συντεταγμένων, απαιτείται για την αναπαράσταση των δεδομένων.

Αναφορικά με τις ακμές του γράφου, πρέπει απαραιτήτως να απεικονίζουν τα μοντέλα επικοινωνίας μέσα στο δίκτυο. Βάσει αυτών, ορίζονται τα παρακάτω γνωρίσματα ακμών:

- ***rssi***: Υποδεικνύει την ένδειξη ισχύος του λαμβανόμενου σήματος αναφορικά με μια κατευθυνόμενη ακμή. Για παράδειγμα, η ισχύς του σήματος που έλαβε ο κόμβος του δικτύου έστω Α από τον κόμβο Β.

■ ***LinkQuality***: Αποτελεί μέτρο για την περιγραφή της ποιότητας του συνδέσμου ανάμεσα σε δύο κόμβους του δικτύου.

■ ***Encryption***: Υποδεικνύει αν η WSN εφαρμογή που χρησιμοποιήθηκε μέσα στο tesbed, χρησιμοποιεί κρυπτογραφημένες ή μη ακμές.

2.2.1.2 Αναπαράσταση χρόνου

Για την αναλυτική, επαρκή περιγραφή της διακινούμενης πληροφορίας είναι απαραίτητη η χρονική απεικόνισή της. Σχετικά με την καταγραφή των χρονικών παραμέτρων των σημείων ενδιαφέροντος λοιπόν, πρέπει να γίνει επέκταση των γνωρισμάτων των δεδομένων της GraphML, με σκοπό την περιγραφή προχωρημένων ιδιοτήτων κόμβων και ακμών. Στην περίπτωση αυτή, υπάρχει ενδιαφέρον για τα διακριτά ή συνεχή γεγονότων του δικτύου, που μεταβάλλονται κατά την εξέλιξη των πειραμάτων. Παραδείγματα γεγονότων, αποτελούν το άνοιγμα ή κλείσιμο φωτός ή η μετακίνηση ενός κόμβου μέσα στο δίκτυο αισθητήρων. Συνεπώς τα παρακάτω στοιχεία ορίστηκαν στα πλαίσια της GraphML:

■ ***<timespan>***: Χρησιμοποιείται για να ορίσει μια περίοδο χρόνου κατά την οποία εμφανίζονται συγκεκριμένα επιθυμητά χαρακτηριστικά (για παράδειγμα η ακριβής τοποθεσία ενός κόμβου) με συγκεκριμένες τιμές. Για την ακριβέστερη περιγραφή του χαρακτηριστικού αυτού, εισάγονται δύο πρόσθετα προαιρετικά πεδία:

- <begin>*: Είναι ένα πεδίο για τη δήλωση της πρώτης χρονικής στιγμής, όπου η τιμή του γνωρίσματος που εξετάζουμε ενεργοποιήθηκε.*

- <end>*: Το πεδίο αυτό ορίζει την τελευταία χρονική στιγμή, όπου η τιμή του γνωρίσματος που εξετάζουμε ήταν ενεργή.*

Παράδειγμα τμήματος GraphML, όπου χρησιμοποιείται το γνώρισμα timespan φαίνεται στην ακόλουθη εικόνα:

```
<node id="urn:wisebed:node:cti:gw1:n0"/>
  <data key="isGateway">true</data>
  <data key="longitude">
    <timespan>
      <begin>0</begin>
      <end>1000</end>
      <value>32.0</value>
    </timespan>
  </data>
```

■ ***<timestamp>***: Χρησιμοποιείται για τον ορισμό ενός χρονικού σημείου όπου ένα συγκεκριμένο χαρακτηριστικό (για παράδειγμα η τιμή rssi κάποιας συγκεκριμένης ακμής) στο οποίο ενδιαφερόμαστε, παίρνει συγκεκριμένη τιμή. Για την ακριβέστερη περιγραφή του χαρακτηριστικού αυτού, εισάγεται ένα πρόσθετο προαιρετικό πεδίο:

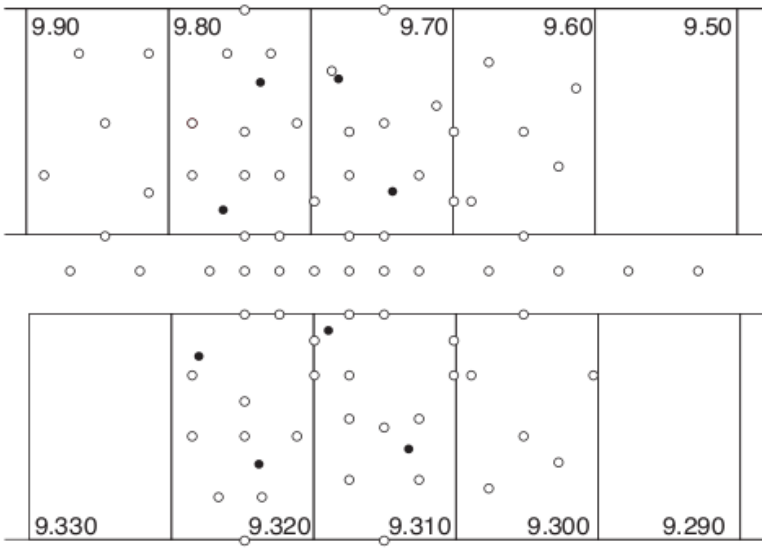
- ***<when>****: Το πεδίο δηλώνει το σημείο στο χρόνο όπου το υπό εξέταση γεγονός λαμβάνει χώρα.*

Παράδειγμα τμήματος GraphML όπου χρησιμοποιείται το γνώρισμα timestamp φαίνεται στην ακόλουθη εικόνα:

```
...
<edge source="urn:wisebed:node:cti:gw1:n0" target="urn:wisebed:node:cti:gw1:n2">
  <data key="rssi">
    <timestamp>
      <when>2009-05-29T23:59:59+03:00</when>
      <value>120</value>
    </timestamp>
  </data>
  <data key="encryption">true</data>
</edge>
```

Αξίζει να σημειωθεί, ότι και τα δύο παραπάνω γνωρίσματα, timestamp και timespan, μπορούν να εμφανιστούν σε μια GraphML περιγραφή, αρκεί να μην αλληλοσυγκρούονται οι τιμές τους. Με άλλα λόγια, δεν επιτρέπεται η περιγραφή των δύο πεδίων αναφέρεται στην ίδια χρονική στιγμή. Στην

παρακάτω εικόνα φαίνεται η εικονική προσομοίωση της τοπολογίας ενός ασύρματου δικτύου αισθητήρων, και στη συνέχεια τμήμα της μετάφρασής του σε GraphML format.




```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
     http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">

  <key id="isGateway" for="node" attr.name="isGateway" attr.type="boolean">
    <default>false</default>
  </key>

  <key id="poiName" for="node" attr.name="poiName" attr.type="string"/>
  <key id="poiDescription" for="node"
    attr.name="poiDescription" attr.type="string"/>

  <key id="longitude" for="node" attr.name="longitude" attr.type="double"/>
  <key id="latitude" for="node" attr.name="latitude" attr.type="double"/>
  <key id="altitude" for="node" attr.name="altitude" attr.type="double"/>

  <key id="rssi" for="edge"
    attr.name="rssi" attr.type="int" />
  <key id="encryption" for="edge"
    attr.name="encryption" attr.type="boolean">
    <default>false</default>
  </key>
```

2.2.2 SensorML Format

Η SensorML παρέχει μοντελοποίηση και XML κωδικοποίηση για την περιγραφή αισθητήρων, την επεξεργασία των μετρήσεων που προέρχονται από αυτούς και την παροχή συνόλου εντολών για την παραγωγή υψηλού επιπέδου πληροφορίας από τις μετρήσεις των αισθητήρων. Όλες οι διεργασίες, ορίζουν τις εισόδους,παραμέτρους, μεθόδους και εξόδους τους όπως και παρέχουν σχετικά metadata. Συνοψίζοντας λοιπόν, η SensorML μοντελοποιεί ανιχνευτές και αισθητήρες σαν διεργασίες οι οποίες μπορούν να μετατρέψουν πραγματικά φαινόμενα σε δεδομένα.

Η πορεία της κωδικοποίηση αυτής, ξεκίνησε μόλις το 1998, υπό την επίβλεψη της διεθνούς επιτροπής *"Committee for Earth Observing Satellites"* . Από τότε η SensorML διαρκώς αναπτύσσεται και εξελίσσεται ενώ λειτουργεί εξαιρετικά καλά σε μια σειρά εφαρμογών όπως αυτές που παρουσιάζονται παρακάτω:

◆ ***Electronic Specification Sheet:*** Θεωρείται η απλούστερη δυνατή εφαρμογή. Η SensorML μπορεί να χρησιμοποιηθεί για να προσφέρει αναγνωρισμένα ψηφιακά μέσα για την παροχή specifiacation sheets αναφορικά με συστατικά αισθητήρων αλλά και ολόκληρα συστήματα.

◆ ***Εύρεση Αισθητήρων, Συστημάτων Αισθητήρων και διεργασιών***: Η SensorML συνιστά ένα μέσο, με τη βοήθεια του οποίου συστήματα αισθητήρων ή διεργασίες μπορούν να τεθούν προς ανακάλυψη από άλλους χρήστες. Αυτό μπορεί να γίνει μέ τη βοήθεια της πλούσιας συλλογής metadata που προσφέρει η SensorML. Αυτά τα metadata περιλαμβάνουν αναγνωριστικά, κατηγοριοποιήσεις, περιορισμούς ως προς χρόνο, ασφάλεια και νομικά θέματα, δυνατότητες, χαρακτηριστικά κ.λ.π.

◆ ***Σειρές Παρατηρήσεων***: Η SensorML μπορεί να παρέχει πλήρη και μονοσήμαντη περιγραφή σειρών παρατηρήσεων που προκύπτουν από συστήματα αισθητήρων. Πιο συγκεκριμένα, είναι δυνατή η λεπτομερής περιγραφή μιας επιστημονικής παρατήρησης από την στιγμή απόκτησης των πρώτων δεδομένων μέχρι τη στιγμή μετάφρασης της από ειδικό αναλυτή.

◆ ***On-demand Επεξεργασία Παρατηρήσεων***: Αλυσίδες επεξεργασίας υψηλού ή χαμηλού επιπέδου των παρατηρήσεων που προκύπουν από συστήματα αισθητήρων, μπορούν να περιγραφούν σε SensorML, να διαδοθούν στο ίντερνετ και να εκτελεστούν on-demand, χωρίς να απαιτείται a priori γνώση των χαρακτηριστικών των αισθητήρων ή των επεξεργαστών. Επιπλέον, σημειώνεται οτι η SensorML επιτρέπει την κατανομή της επεξεργασίας σε οποιοδήποτε

σημείο μέσα στην αλυσίδα αισθητήρων: από τις συσκευές αισθητήρων, στο κέντρο επεξεργασίας δεδομένων ή ακόμα και στο ανεξάρτητο PDA του χρήστη. Για όλα τα παραπάνω, δε χρειάζεται η ανάπτυξη ειδικού λογισμικού για τους αισθητήρες.

◆ ***Auto-configuring & Αυτόνομα Δίκτυα Αισθητήρων***: Η SensorML επιτρέει την ανάπτυξη plug-n-play αισθητήρων, προσομοιώσεων, και διεργασιών που μπορούν εύκολα να προστεθούν στα συστήματα αποφάσεων. Τα ίδιο-περιγραφικά χαρακτηριστικά που προσφέρονται από τη SensorML, καθιστούν δυνατή την υποστήριξη αυτόνομων δικτύων αισθητήρων στα οποία οι κόμβοι μπορούν να δημοσιεύσουν συναγερμούς ή εργασίες στα οποία άλλοι αισθητήρες μπορούν να εγγραφούν και να αντιδράσουν

◆ ***Αρχειοθέτηση Παραμέτρων Αισθητήρων***: Τελικά, η SensorML προσφέρει ένα μηχανισμό για την αρχειοθέτηση θεμελιωδών παραμέτρων και υποθέσεων αναφορικά με τους αισθητήρες και τις σχετικές διεργασίες που τους αφορούν, με τέτοιο τρόπο ώστε παρατηρήσεις και συμπεράσματα από αυτά τα συστήματα να δύναται να επανεξεταστούν και να βελτιωθούν ακόμα και αν μεγάλο χρονικό διάστημα έχει περάσει από την τελευταία επιτυχή επεξεργασία τους. Στην ακόλουθη εικόνα φαίνεται ένα παράδειγμα SensorML format:

```
−<sml:identifier name="Short Name">
  −<sml:Term definition="urn:ogc:def:identifier:OGC:shortName">
     <sml:value>SBE 37-SMP</sml:value>
   </sml:Term>
 </sml:identifier>
−<sml:identifier name="Long Name">
  −<sml:Term definition="urn:ogc:def:identifier:OGC:longName">
     <sml:value>SBE 37-SMP MicroCAT CTP Recorder</sml:value>
   </sml:Term>
 </sml:identifier>
−<sml:identifier name="Manufacturer Name">
  −<sml:Term definition="urn:ogc:def:identifier:OGC:manufacturerName">
     <sml:value>Seabird Electronics</sml:value>
   </sml:Term>
 </sml:identifier>
−<sml:identifier name="Model Number">
  −<sml:Term definition="urn:ogc:def:identifier:OGC:modelNumber">
     <sml:codeSpace xlink:href="urn:ogc:def:identifier:SBE:modelNumber"/>
     <sml:value>SBE37SMP</sml:value>
   </sml:Term>
```

Τα βασικά συστατικά της SensorML περιγράφονται ως ακολούθως:

- ***Component***: Αναπαριστά φυσικές, ατομικές διεργασίες που μπορούν να

μετατρέψουν την αποθηκευμένη πληροφορίας από μια μορφή σε κάποια άλλη. Για παράδειγμα, ένας ανιχνευτής τυπικά μετατρέπει μια φυσικά ανιχνεύσιμη ιδιότητα ή φαινόμενο σε ψηφιακά νούμερα. Παραδείγματα άλλων τέτοιων components αποτελούν ανιχνευτές, ενεργοποιητές και φυσικά φίλτρα.

- ***System***: Το πεδίο αυτό συνθέτει ένα μοντέλο ομάδας ή δομής συστατικών , τα οποία μπορεί να περιλαμβάνουν ανιχνευτές, ενεργοποιητές και φυσικά φίλτρα. Το πεδίο αυτό συσχετίζει μια διεργασία με τον πραγματικό κόσμο και συνεπώς παράγει σημαντική πληροφορία για τα συστατικά του συστήματος, όπως και για διεπαφές επικοινωνίας μεταξύ τους.

- ***Process Model***: Αποτελεί ατομικό σώμα επεξεργασίας, το οποία συνήθως χρησιμοποιείται με πιο σύνθετες αλυσίδες επεξεργασίας. Συσχετίζεται με το πεδίο "***Process Method***" όπου ορίζονται οι διεπαφές διεργασιών όπως και ο τρόπος εκτέλεσης των μοντέλων επεξεργασίας. Τέλος ορίζονται παράμετροι εισόδου και εξόδου.

- ***Process Chain***: Αποτελεί σύνθετο μοντέλο επεξεργασίας, το οποία συνίσταται από δια-συνδεδεμένες και αλληλοεξαρτώμενες διεργασίες. Μια αλυσίδα διεργασιών μπορεί επίσης να περιλαμβάνει πιθανές πηγές δεδομένων όπως και συσχετίσεις των σημάτων εσόδου και εξόδου των διεργασιών.

- ***Process Method***: Το πεδίο αποτελεί ορισμό της συμπεριφοράς και διεπαφής ενός μοντέλου διεργασίας. Μπορεί να αποθηκευτεί σε βιβλιοθήκη ώστε να είναι δυνατή η επαναχρησιμοποίησή του. Απαραιτήτως, περιγράφει τη διεπαφή της διεργασίας, τον αλγόριθμο που χρησιμοποιεί όπως επίσης και παρέχει πληροφορίες στους χρήστες σχετικά με την υλοποίησή της.

- ***Detector***: Αποτελεί ατομικό συστατικό ενός σύνθετου συστήματος μετρήσεων το οποίο ορίζει δείγματα και χαρακτηριστικά απόκρισης απλών ανιχνευτικών συσκευών. Ένας ανιχνευτής έχει μόνο μια είοδο και έξοδο, οι οποίες συνιστούν κλιμακωτές οντότητες. Πιο σύνθετοι αισθητήρες που συνίστανται από πολλαπλούς ανιχνευτές μπορούν να περιγραφούν σε SensorML σαν πίνακες ή αμάδες πεδίων ανιχνευτών. Σημειώνεται οτι στη SensorML το πεδίο ανιχνευτής αποτελεί ένα ειδικό μοντέλο επεξεργασίας.

- ***Sensor***: Συνιστά ειδικό τύπο του συστήματος το οποίο αναπαριστά μια πλήρη συσκευή αισθητήρων.

2.2.3 WiseMLFormat

Το WiseML Format αποτελεί την πιο πρόσφατη εξέλιξη των μοντέλων αναπαράστασης δεδομένων δικτύων αισθητήρων. Η WiseML αποτελεί επέκταση του GraphML format. Ο βασικός στόχος πίσω από το σχεδιασμό της είναι η ανάγκη για ένα περιγραφικό format που επιτρέπει τη μοντελοποιημένη αποθήκευση πειραματικών traces. Επιπλέον παρατηρήθηκε ότι το σχήμα αυτό θα μπορούσε να χρησιμοποιηθεί για να διευκολύνει την επικοινωνία μεταξύ πολλαπλών εργαλείων που αναπτύσσονται στα πλαίσια λειτουργίας ενός δικτύου αισθητήρων. Το συνολικό μοντέλο WiseML εμφανίζεται κατά μινιμαλιστικό τρόπο προσφέροντας περιορισμένες βασικές λειτουργίες αντί για πληθώρα οι οποίες τείνουν να παραμένουν αχρησιμοποίητες.
Κάθε ξεχωριστό πειραματικό αποτύπωμα αποθηκεύεται σε διαφορετικό αρχείο, ενώ το όνομα του αρχείου αυτού πρέπει να έχει την κατάληξη ***".wiseml"*** . Το αρχείο περιέχει όλες τις απαραίτητες πληροφορίες για την αναγνώριση και επανα-παραγωγή ενός προσωμοιωμένου πειράματος, συνεπώς δεν υπάρχουν περιορισμοί στην πληροφορία που πρέπει να κωδικοποιήσει το αρχείο.
Κατά κανόνα ένα WiseML αρχείο συνίσταται από τρεις βασικές ενότητες όπως φαίνονται παρακάτω:

- ***Ενότητα Setup***: Η ενότητα περιέχει όλες της πληροφορίες που χρειάζονται προκειμένου να γίνει η εκκίνηση ενός παρόμοιου πειράματος. Απαριθμούνται τα συστατικά του δικτύου, οι δυνατότητές τους, το περιβάλλον ανάπτυξης κ.λ.π

- ***Ενότητα Scernario***: Η ενότητα αυτή, έχει προστεθεί με σκοπό να καθίσταται δυνατή η μελέτη συγκεκριμένων προσομοιωμένων πειραμάτων. Για παράδειγμα, ο χρήστης μπορεί να φιλτράρει τμήματα πληροφορίας του αρχείου στο επίπεδο αυτό όπως την απενεργοποίηση ενός κόμβου ή συνδέσμου.

- ***Δυναμικές Πληροφορίες***: Η ενότητα παρουσιάζει τα πραγματικά καταγεγραμμένα δεδομένα. Εδώ ο χρήστης μπορεί να ορίσει δυναμικά στοιχεία του δικτύου όπως καταγεγραμμένα δεδομένα σε συνάρτηση του χρόνου κ.λ.π.

Έτσι, βάσει όλων των παραπάνω και από σκοπιά υψηλού επιπέδου ένα WiseML αρχείο εμφανίζει την ακόλουθη δομή:

```
<wiseml version="1.0" xmlns="http://wisebed.eu/ns/wiseml/1.0">

      <setup>
            <!-- static information section -->
      </setup>
      <scenario id="…">
            <!-- scenario information -->
      </scenario>
      <trace id="…">
            <!-- dynamic information -->
      </trace>

</wiseml>
```

2.2.3.1 Πληροφορίες Εκκίνησης της WiseML

Η ενότητα αυτή περιγράφει όλα τα δεδομένα που συγκεντρώθηκαν κατά την εκκίνηση του πειράματος (***Setup***). Τα δεδομένα αυτής της ενότητας κατηγοριοποιούνται ως εξής:

- ***Πληροφορίες σχετιζόμενες αποκλειστικά με την εκκίνηση του πειράματος:***

Πιο συγκεκριμένα σε αυτό το τμήμα δεδομένων έχουν οριστεί οι παρακάτω ιδιότητες

<origin>: Περιέχει πληροφορίες για τις συντεταγμένες του συστήματος.
<timeinfo>: Αναφέρεται σε πληροφορίες σχετικές με τη χρονική εξέλιξη του πειράματος.
<interpolation>: Ορίζει τα είδη παρεμβολής που μπορούν να χρησιμοποιηθούν για την περιγραφή της κινητικότητας του πειράματος.
<coordinatetype>: Αναφέρεται στη μονάδα μέτρησης των συντεταγμένων
<description>: Προσφέρει μια συνοπτική παρουσίαση του πειράματος.

- ***Πληροφορίες σχετιζόμενες με τους κόμβους του δικτύου, τις εξορισμού τιμές τους και τα χαρακτηριστικά τους:***

Πιο συγκεκριμένα σε αυτό το τμήμα δεδομένων έχουν οριστεί οι παρακάτω ιδιότητες:

<position>*: Παρουσιάζει τις συνταγμένες x, y, z των κόμβων του συστηματος*
<gateway>*: Προσδιορίζει αν κάποιος συγκεκριμένος κόμβος χρησιμοποιήθηκε ως gateway ή όχι.*
<image>*: Αναφέρεται στο image λογισμικού του κόμβου*
<nodetype>*: Παρέχει πληροφορίες για τον τύπο της συσκευής των κόμβων*
<description>*: Συνοπτική περιγραφή δεδομένου κόμβου*
<capability>*: Μέσα σε αυτή την ενότητα παρουσιάζονται οι δυνατότητες και λοιπά χαρακτηριστικά των κόμβων όπως όνομα, μονάδα μέτρησης δεδομένων κ.λ.π*

- ***Πληροφορίες σχετιζόμενες με τις ακμές του δικτύου, τις εξορισμού τιμές τους και τα χαρακτηριστικά τους***

Πιο συγκεκριμένα σε αυτό το τμήμα δεδομένων έχουν οριστεί οι παρακάτω ιδιότητες:

<encrypted>: Αποτελεί μια boolean μεταβλητή που υποδηλώνει αν η ακμή έχει κρυπτογραφηθεί ή όχι
<rssi>: Προσδιορίζει την rssi τιμή της κάθε ακμής. Ο χρήστης πρέπει να προσδιορίσει τον τύπο δεδομένων που χρησιμοποιείται, όπως και τις εξορισμού τιμές του πεδίου.
<capability>: Αναφέρεται σε χαρακτηριστικά των ακμών όπως κόστος, ρυθμός λαθών ανά πακέτο

Βάσει όλων των παραπάνω στοιχείων μέσω της ενότητας setup και των πεδίων position, τα χαρακτηριστικά των κόμβων και ακμών προσφέρεται μια εποπτική ανάλυση της τοπολογίας του συστήματος. Τμήμα setup section, ενός wiseml αρχείου παρουσιάζεται παρακάτω:

```
<setup>

      <!-- setup properties section -->
      <origin>
            <x>1.0</x>
            <y>5.8</y>
            <z>2</z>
            <phi>5</phi>
            <theta>0</theta>
      </origin>
      <timeinfo>
            <start>2009-08-28T14:03:00Z</start>
            <end>2009-08-28T15:03:00Z</end>
            <!-- OR <duration>12</duration> -->
            <unit>seconds</unit>
      </timeinfo>
      <interpolation>cubic</interpolation>
      <description>this is an exmaple WiseML with all the
lements.</description>

      <!-- default values for nodes -->
      <defaults for="node">
            <position>
                  <x>1.0</x>
                  <y>2</y>
                  <z>1.879</z>
            </position>
```

...

Σημειώνεται πως υπάρχουν προαιρετικά και υποχρεωτικά πεδία. Οι πληροφορίες χαρακτηριστικών και δυνατοτήτων των περιγραφόμενων οντοτήτων παρουσιάζονται στην ακόλουθη μορφή:

```
<defaults for="node">
      <gateway>false</>
      <capability>
            <name>urn:wisebed:node:capability:temperature</name>
            <datatype>integer</datatype>
            <unit>lux</unit>
            <default>0</default>
      </capability>
</defaults>
```

2.2.3.2 Πληροφορίες Σεναρίου της WiseML

Βασιζόμενοι πια στα πραγματικά σύνολα δεδομένων που ορίζονται και παράγονται στο υπό εξέταση σύστημα αισθητήρων, μπορεί να οριστεί με τη βοήθεια της WiseML, αριθμός σεναρίων λειτουργίας. Για παράδειγμα, μια αποτυχία στο δίκτυο μπορεί να προσομοιωθεί μέσω της απενεργοποίησης ενός κόμβου ή της τροποποίησης της rssi τιμής συγκεκριμένης ακμής κάποια δεδομένη χρονική στιγμή. Δεδομένου του γεγονότος ότι τα πραγματικά δεδομένα χρησιμοποιούνται στη συνέχεια σε ποικίλες προσομοιώσεις, η WiseML μία ξεχωριστή ενότητα, που ονομάζεται *Scenario Section* και όπου η περιγραφή όλων αυτών των σεναρίων μπορεί να προσδιοριστεί.

Υπάρχουν δύο διαφορετικές ενέργειες που μπορεί να ορίσει ο χρήστης:

- **Τροποποίηση της υπάρχουσας τοπολογίας**: Αναφέρεται στην ενεργοποίηση ή απενεργοποίηση κόμβων και ακμών του δικτύου. Πιο συγκεκριμένα για την επίτευξη της αλλαγής τοπολογίας προσφέρονται από τη WiseML τα ακόλουθα πεδία:
    - ***<enablenode>:*** Χρησιμοποιείται για την ενεργοποίηση κάποιου κόμβου του συστήματος. Κατά την εκκίνηση ενός πειράματος όλοι οι κόμβοι θεωρούνται ενεργοποιημένοι.
    - ***<disablenode>:*** Χρησιμοποιείται για την απενεργοποίηση δεδομένου κόμβου. Αυτό σημαίνει πως τα παραγόμενα αποτελέσματα από τον κόμβο παραβλέπονται όπως και τα στοιχεία των ακμών που είτε καταλήγουν είτε ξεκινούν από αυτόν.
    - ***<enablelink>:*** Χρησιμοποιείται για την ενεργοποίηση ακμής του δικτύου.
    - ***<disablelink>:*** Χρησιμοποιείται για την απενεργοποίηση ακμής του δικτύου.

- **Τροποποίηση των δεδομένων που έχουν ανιχνευθεί:** Η συγκεκριμένη ενέργεια σχετίζεται κυρίως με τη διόρθωση σφαλμάτων που προέκυψαν κατά την ανάγνωση των δεδομένων του δικτύου.

Παράδειγμα της ενότητας σεναρίου σε WiseML εμφανίζεται στην ακόλουθη εικόνα:

```
<scenario id="...">

      <timestamp>0</timestamp>
      <enablenode id="..." />
      <disablenode id="..." />
      <enablelink source="..." target="..." />
      <disablelink source="..." target="..." />

      <node id="...">
            <position>
                  <x>0</x>
                  <y>1</y>
                  <z>2</z>
                  <phi>0</phi>
                  <theta>1</theta>
            </position>
            <data key="lqi">50</data>
      </node>

</scenario>
```

2.2.3.3 Δυναμικά Δεδομένα

Το συγκεκριμένο τμήμα του WiseML format, προσφέρεται για την αναπαράσταση δεδομένων του συστήματος που αλλάζουν δυναμικά με την πάροδο του χρόνου, όπως για παράδειγμα οι μετρήσεις των αισθητήρων ή η τοπολογία του κινητού δικτύου αισθητήρων. Η πληροφορία, αποθηκεύεται στα αρχεία ανά διακριτές στιγμές, χρονολογικά διατεταγμένες. Για την περιγραφή των πολλαπλών σημείων ενδιαφέροντος στο χρόνο, εισάγεται από τη wiseml το tag ***timestap*** ενώ για την περιγραφή της κινητικότητας των κόμβων και άρα των δυναμικών μεταβολών στην τοπολογία του συστήματος χρησιμοποιήθηκε το tag ***interpolation.*** Παράδειγμα τμήματος WiseML, που αναφέρεται σε δυναμικές πληροφορίες εμφανίζεται παρακάτω:

```
<timestamp>10</timestamp>
<!-- information for this moment in time -->

<timestamp>20</timestamp>
<!-- information for this moment in time -->
```

2.2.4 Future Work σε θέματα αναπαράστασης δεδομένων δικτύου

Αναφορικά με τη συσχέτιση κόμβων των testbeds με συγκεκριμένες οντότητες όπως διάφορα σημεία ενδιαφέροντος, δωμάτια κτιρίων, κ.λ.π σκοπεύεται να χρησιμοποιηθεί στο κοντινό μέλλον η δομή "***Hyperedge***". Οι Hyperedges αποτελούν γενικεύσεις των ακμών των γράφων, εκφράζοντας μια σχέση (φυσική είτε λογική) ανάμεσα σε έναν αυθαίρετο αριθμό κόμβων. Τέτοιες σχέσεις μπορούν να χρησιμοποιηθούν για να εκφράσουν γεγονότα όπως για παράδειγμα ότι ένα σύνολο κόμβων αισθητήρων βρίσκεται φυσικά στο ίδιο δωμάτιο ή ότι δύο κάμερες παρακολουθούν τον ίδιο χώρο και θα έπρεπε να έχουν τη δυνατότητα να ανιχνεύσουν τα ίδια γεγονότα όταν αυτά συμβαίνουν. Αυτές οι λειτουργίες μπορούν να φανούν ιδιαίτερα χρήσιμες με τους παρακάτω τρόπους:

- Οι ερευνητές που επιθυμούν τη χρήση ή τον ορισμό οντοτήτων μέσα σε ένα δίκτυο αισθητήρων, μπορούν να χρησιμοποιήσουν τις Hyperedges σαν βάση για τα πειράματά τους.

- Οι Hyperedges μπορούν να χρησιμοποιηθούν για την απλοποίηση της εξαγωγής δεδομένων από τα δίκτυα αισθητήρων

- Οι χρήστες μπορούν να αλλάξουν τις παραμέτρους προσομοίωσης του έργου κατά τρόπο σύμφωνο με με το πραγματικό περιβάλλον των testbed

Ένα παράδειγμα ορισμού Hyperedge ανάμεσα στους κόμβους έστω n1 και n3 εμφανίζεται παρακάτω:

```
<hyperedge>
  <endpoint node="n1"/>
  <endpoint node="n3"/>
</hyperedge>
```

ΚΕΦΑΛΑΙΟ 3:

Η ΠΛΑΤΦΟΡΑΜΑ

"EasySense"

ΚΕΦΑΛΑΙΟ 3

Στο παρόν κεφάλαιο θα παρουσιαστούν τα βασικά σχεδιαστικά χαρακτηριστικά της πλατφόρμας EasySense. Περιγράφονται τα πρωτόκολλα που χρησιμοποιήθηκαν κατά την υλοποίηση, οι λογικές λύσεις που υιοθετήθηκαν και οι βασικές τεχνολογίες στην οποίες βασίστηκε η ανάπτυξη του λογισμικού.

3.1 Η Αρχιτεκτονική της πλατφόρμας EasySense

Η πλατφόρμα EasySense βασίζεται σε μια κλασσική **3-tier** client-server αρχιτεκτονική. Γενικά, το client-server computing αναφέρεται σε μια βασική αλλαγή στο στυλ των υπολογιστών, την αλλαγή από τα συστήματα που βασίζονται στα μηχανήματα στα συστήματα που βασίζονται στον χρήστη. Ειδικότερα, ένα σύστημα client-server είναι ένα σύστημα στο οποίο το δίκτυο ενώνει διάφορους υπολογιστικούς πόρους, ώστε οι clients (ή αλλιώς front end) να μπορούν να ζητούν υπηρεσίες από έναν server (ή αλλιώς back end), ο οποίος προσφέρει πληροφορίες ή επιπρόσθετη υπολογιστική ισχύ. Με άλλα λόγια, στο client-server μοντέλο, ο client θέτει μια αίτηση και ο server επιστρέφει μια ανταπόκριση ή κάνει μια σειρά από ενέργειες. Ο server μπορεί να ενεργοποιείται άμεσα για την αίτηση αυτή ή να προσθέτει την αίτηση σε μια ουρά. Η άμεση ενεργοποίηση για την αίτηση μπορεί, για παράδειγμα, να σημαίνει ότι ο server υπολογίζει έναν αριθμό και τον επιστρέφει αμέσως στον client. Η τοποθέτηση της αίτησης σε μια ουρά μπορεί να σημαίνει ότι η αίτηση πρέπει να τεθεί σε αναμονή για να εξυπηρετηθεί. Ένα καλό παράδειγμα για αυτό είναι όταν εκτυπώνουμε ένα κείμενο σε ένα εκτυπωτή δικτύου. Ο server τοποθετεί την αίτηση σε μια ουρά μαζί με αιτήσεις εκτυπώσεων και από άλλους clients. Μετά επεξεργάζεται την αίτηση με βάση την σειρά προτεραιότητας, η οποία , σε αυτή την περίπτωση, καθορίζεται από τη σειρά με την οποία ο server

παρέλαβε την απαίτηση. Το client-server computing είναι πολύ σημαντικό, διότι επιτυγχάνει τα εξής:

- Αποτελεσματική χρήση της υπολογιστικής ισχύος.
- Μείωση του κόστους συντήρησης, δημιουργώντας συστήματα client-server που απαιτούν λιγότερη συντήρηση και κοστίζουν λιγότερο στην αναβάθμιση.
- Αύξηση της παραγωγικότητας, προσφέροντας στους χρήστες ξεκάθαρη πρόσβαση στις αναγκαίες πληροφορίες μέσω σταθερών και εύκολων στην χρήση διασυνδέσεων.
- Αύξηση της ευελιξίας και της δυνατότητας δημιουργίας συστημάτων που υποστηρίζουν πολλά περιβάλλοντα.

Με βάση αυτούς τους σκοπούς, οι οργανισμοί που κινούνται προς την κατεύθυνση της client-server τεχνολογίας αυξάνουν κατά πολύ την ανταγωνιστική τους θέση. Σχετικά με τη λειτουργία του συστήματος, η πλευρά του client πρώτα στέλνει ένα μήνυμα για να καλέσει σε ετοιμότητα τον server. Από τη στιγμή που ο client και ο server έχουν επικοινωνία μεταξύ τους, ο client μπορεί να υποβάλλει την αίτησή του. Ο client είναι ο αιτών των υπηρεσιών. Ο client δεν μπορεί παρά να είναι ένας υπολογιστής. Οι υπηρεσίες που ζητούνται από τον client μπορεί να υπάρχουν στους ίδιους σταθμούς εργασίας ή σε απομακρυσμένους σταθμούς εργασίας που συνδέονται μεταξύ τους μέσω ενός δικτύου. Ο client ξεκινάει πάντα την επικοινωνία. Τα συστατικά του client είναι πολύ απλά. Μια client μηχανή πρέπει να μπορεί να κάνει τα ακόλουθα:

- Να τρέχει το λογισμικό των γραφικών διεπαφών χρηστών (GUIs).
- Να δημιουργεί τις αιτήσεις για πληροφορίες και να τις στέλνει στον server.
- Να αποθηκεύει τις επιστρεφόμενες πληροφορίες.

Αυτές οι αιτήσεις καθορίζουν πόση μνήμη χρειάζεται, ποια ταχύτητα επεξεργασίας θα μπορούσε να βελτιώσει τον χρόνο ανταπόκρισης, και πόση χωρητικότητα αποθήκευσης απαιτείται. Από την άλλη, Ο server απαντάει στις αιτήσεις που γίνονται από τους clients. Ένας client μπορεί να ενεργεί ως server εάν λαμβάνει και επεξεργάζεται αιτήσεις όπως ακριβώς και τις στέλνει (για παράδειγμα, ένας σταθμός εργασίας που

χρησιμοποιείται και ως server εκτυπώσεων από άλλους). Οι server δεν ξεκινάνε τις επικοινωνίες -περιμένουν τις αιτήσεις των clients. Στο παράδειγμα του server εκτυπώσεων ενός δικτύου, ο client ζητάει από τον server να εκτυπώσει ένα κείμενο σε έναν συγκεκριμένο εκτυπωτή και ο server προσθέτει την εκτύπωση σε μια ουρά και ενημερώνει τον client όταν το κείμενο εκτυπωθεί επιτυχημένα. Η διαδικασία του client μπορεί να ανήκει φυσικά στον ίδιο σταθμό εργασίας με την διαδικασία του server. Στο παράδειγμα εδώ, μια εντολή εκτύπωσης μπορεί να εκδίδεται στον server του σταθμού εργασίας του δικτύου, χρησιμοποιώντας την διαδικασία του server εκτυπώσεωνσε αυτόν τον σταθμό εργασίας. Τα συστατικά του server είναι πολύ απλά. Μια server μηχανή πρέπει να μπορεί να κάνει τα ακόλουθα :

- Να επιθεωρεί τις αιτήσεις των clients.
- Να αποθηκεύει, να ανακτά και να προστατεύει πληροφορίες.
- Να δημιουργεί εφαρμογές διαχείρισης πληροφοριών, όπως δημιουργία αντιγράφων, ασφάλεια κτλ.
- Να διαχειρίζεται πληροφορίες.

Ο server πρέπει να είναι ικανός να ανταποκριθεί στην αίτηση του client αμέσως. Εάν πολλοί clients κάνουν αιτήσεις ταυτόχρονα, ο server πρέπει να είναι ικανός να βάζει σε προτεραιότητα τις αιτήσεις των clients, και να επεξεργάζεται πολλές αιτήσεις την στιγμή. Από την στιγμή, που ο server επιβεβαιώνει ότι ο χρήστης είναι εξουσιοδοτημένος να κάνει αιτήσεις στον server, ο server μπορεί να αποκαλύψει την απαίτηση και να την επεξεργαστεί. Η αίτηση μπορεί να έχει μια από τις ακόλουθες τέσσερις μορφές:

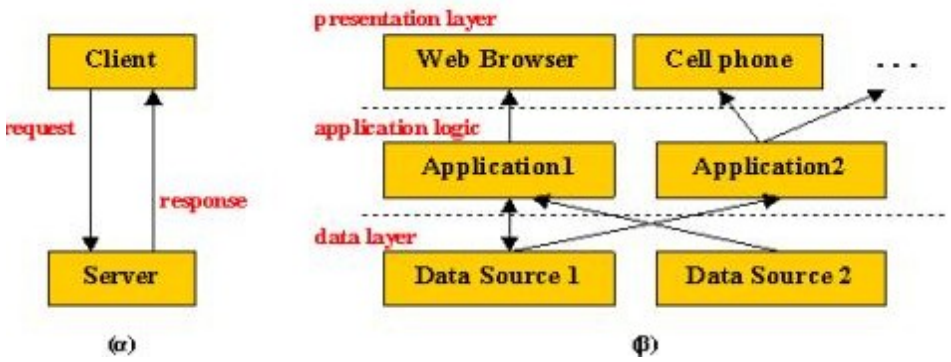
- Μια απόμακρη αίτηση είναι μια απλή αίτηση για πληροφορίες από έναν απλό client.
- Μια απόμακρη συναλλαγή περιλαμβάνει πολλαπλές αιτήσεις για πληροφορίες από έναν απλό client.
- Μια κατανεμημένη συναλλαγή περιλαμβάνει πολλαπλές αιτήσεις για πληροφορίες από έναν απλό client, οι οποίες πληροφορίες ανήκουν σε πολλούς server.
- Μια κατανεμημένη αίτηση είναι μια συναλλαγή που σχηματίζεται από πολλαπλές αιτήσεις για πληροφορίες από πολλαπλούς clients, οι οποίες πληροφορίες ανήκουν σε πολλαπλούς servers.

Αυτές οι αιτήσεις πρέπει να περάσουν από το λεγόμενο ACID τεστ: Ατομικότητα (Atomicity), Συνέπεια (Consistency), Απομόνωση(Isolation) και Αντοχή (Durability). Η ατομικότητα σημαίνει ότι ολόκληρη η συναλλαγή πρέπει να πετύχει ή να αποτύχει, δεν μπορεί να ολοκληρωθεί ως προς ένα κομμάτι της. Η συνέπεια σημαίνει ότι το σύστημα πάει από ένα σταθερό σημείο σε ένα άλλο σταθερό σημείο. Η απομόνωση σημαίνει ότι, από τη στιγμή που μια συναλλαγή ολοκληρώνεται με επιτυχία, τα αποτελέσματα της δεν είναι ορατά σε άλλες συναλλαγές. Η αντοχή σημαίνει ότι από τη στιγμή που η συναλλαγή ολοκληρώνεται με επιτυχία, δεσμεύεται μόνιμα από το σύστημα και επακόλουθες αποτυχίες δεν θα το επηρεάσουν. Εάν η συναλλαγή αποτύχει, το σύστημα οπισθοχωρεί στο σημείο που ήταν πριν προσπαθήσει να επεξεργαστεί την συναλλαγή.

Ένα καλά οργανωμένο σύστημα που χρησιμοποιεί αυτήν την κλασσική αρχιτεκτονική προσφέρει συνολικά τις ακόλουθες δυνατότητες:

- *Ένα υψηλό επίπεδο αξιοπιστίας .*
- *Κεντρικό έλεγχο και κεντρική διαχείριση των πληροφοριών.*
- *Ισχυρή διαχείριση των πληροφοριών και δυνατότητα αποθηκεύσεων*
- *Διαφάνεια .*

Στην ακόλουθη εικόνα φαίνεται σχηματικά ο τρόπος λειτουργίας του αρχιτεκτονικού μοντέλου:



Το περιβάλλον του client-server έχει πολλά πλεονεκτήματα σε σχέση με τις κλασσικές αρχιτεκτονικές. Η διαχείριση της διασύνδεσης των χρηστών και άλλες επεξεργασίες είναι αποφορτισμένα από τον «οικοδεσπότη», ενώ ο server ακόμη προσφέρει συγκεντρωμένο έλεγχο των κοινών πόρων. Επειδή ο client επικοινωνεί με τον server μέσω ενός καθορισμένου συστήματος διασύνδεσης, δεν χρειάζεται να γνωρίζει που ανήκει ο server ή πως ενεργεί.

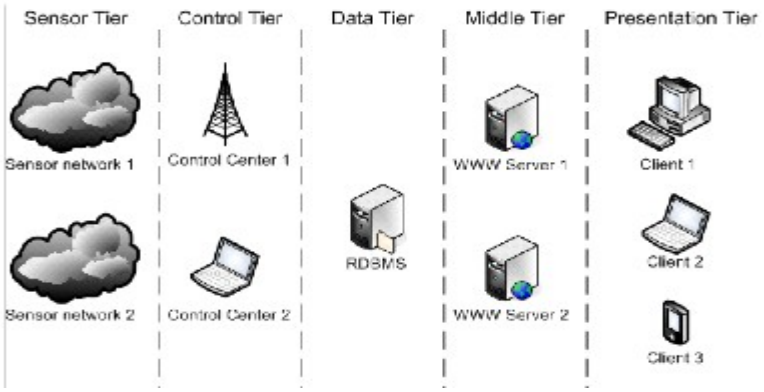
Ο σταθμός εργασίας τρέχει την εφαρμογή και εμφανίζει τις πληροφορίες στον χρήστη. Μόνο όταν ο client προσπελάζει πληροφορίες, τότε εγκαθίσταται επικοινωνία με τον server. Ο φόρτος εργασίας μειώνεται δραματικά στον υπολογιστή-«οικοδεσπότη» όσο αυξάνεται η ισχύς κάθε σταθμού εργασίας.

Ωστόσο, οι κλασικές εφαρμογές περιορίζουν την ευελιξία των τελικών χρηστών. Η διασύνδεση των χρηστών δεν είναι γραφική, κάτι που κάνει το σύστημα δυσκολότερο στη χρήση και σημαίνει ότι ο χρήστης πρέπει να μάθει πως να χρησιμοποιήσει την γλώσσα του οικοδεσπότη.

Επίσης, οι εφαρμογές εξαρτώνται από μια πλατφόρμα, που σημαίνει ότι εάν κάτι συμβεί στον υπολογιστή-«οικοδεσπότη», ο χρήστης δεν μπορεί να χρησιμοποιήσει το σύστημα, έως ότου το σύστημα αρχίσει να επαναλειτουργεί.

Πολλές φορές προτιμούνται να χρησιμοποιούνται όροι, όπως 2-tier, 3-tier client-server αρχιτεκτονικές αντί των όρων ισχυροί clients και ισχυροί servers. Αλλά ουσιαστικά αυτοί οι όροι βασίζονται στην ίδια βασική ιδέα. Έχουν να κάνουν με το πώς διαιρείται η client-server εφαρμογή σε λειτουργικές ενότητες, οι οποίες μετά μπορούν να ανατεθούν είτε στον client, είτε σε έναν ή περισσότερους servers.

Πιο συγκεκριμένα λοιπόν, η πλατφόρμα Easysense, υλοποιεί στο ανώτερο επίπεδο της 3-tier αρχιτεκτονικής της, έναν ***generic data & application portal server*** ο οποίος παρέχει το συνολικό Monitoring API και έναν client ο οποίος χρησιμοποιεί τα παρεχόμενα web-services επιστρέφοντας στο κατάλληλο XML format τα αποτελέσματα στους χρήστες . Στο χαμηλότερο επίπεδο της αρχιτεκτονικής βρίσκεται η 18 πινάκων SQL βάση δεδομένων στην οποία αποθηκεύονται όλα τα δεδομένα του χρησιμοποιούνται, παράγονται ή ανταλλάσσονται στα πλαίσια λειτουργίας του WSN. Στο μεσαίο επίπεδο της αρχιτεκτονικής, το οποίο αποκρύπτει τις λεπτομέρειες σχεδιασμού από το χρήστη, τα δεδομένα της βάσης, αντιστοιχίζονται σε HashMaps προκειμένου να διευκολύνουν την τάχιστη παραγωγή των XML παραγώγων (WiseML, SensorML, GraphML) και την άμεση εμφώλευσή τους σε XML format. Επιπλέον, κάθε πίνακας της βάσης αντιστοιχίζεται σε μια κλάση τύπου Hibernate Entity και γίνεται XML Mapped όπως υπαγορεύει η τεχνολογία. Έτσι, για την υλοποίηση Database Queries και εντοπισμό δεδομένων από τη βάση χρησιμοποιείται η πολύ πιο αποδοτική, όπως θα δούμε και σε επόμενο κεφάλαιο, γλώσσα HQL. Σχηματικά τα παραπάνω συνοψίζονται στην ακόλουθη εικόνα:



Η αρχιτεκτονική, υλοποιήθηκε με τη βοήθεια JaxWS (Java Web Services), μια τεχνολογία που θα παρουσιαστεί αναλυτικά σε επόμενο κεφάλαιο. Ο client λοιπόν, στέλνει αιτήσεις στον server, οι οποίες σχετίζονται με την αναζήτηση δεδομένων του δικτύου αισθητήρων. Τέτοιες αιτήσεις μπορεί να αφορούν την εύρεση / εντοπισμό συγκεκριμένου αριθμού χαρακτηριστικών των οντοτήτων του ασύρματου δικτύου αισθητήρων, την εύρεση δεδομένων του δικτύου που ικανοποιούν συγκεκριμένες συνθήκες, την περιγραφή της τρέχουσας τοπολογίας ή των επιστρεφόμενων αποτελεσμάτων σε κάποιο από τα XML formats που συζητήθηκαν στο κεφάλαιο 2 όπως wiseml, graphML ή sensorML, κ.λ.π. Τέλος ο server ανταποκρίνεται υλοποιώντας τις συναρτήσεις που παρουσιάστηκαν στην ενότητα 3.1 και επικοινωνώντας κατάλληλα με τη βάση δεδομένων του συστήματος.

3.2 *Θέματα Υλοποίησης*

Η πλατφόρμα Easysense, είναι υλοποιημένη σε Java v.6, και περιλαμβάνει έναν generic portal server ο οποίος παρέχει διαρκή επιτήρηση / επίβλεψη και καταγραφή των συνθηκών του ασύρματου δικτύου αισθητήρων οποιαδήποτε χρονική στιγμή. Με τη χρήση Java Web Services κατά το σχεδιασμό του λογισμικού, αιτήσεις πελατών σχετικές με:

I. Περιγραφή της τρέχουσας τοπολογίας του δικτύου,
II. Περιγραφή του πλαισίου λειτουργίας του WSN, όπως για παράδειγμα των χαρακτηριστικών του τρέχοντος πειράματος που διεξάγεται στο δίκτυο,
III. Επισκόπηση παρελθόντων πειραματικών traces και περιγραφή προηγούμενων αποτελεσμάτων που συλλέχθηκαν
IV. Εντοπισμό, αναζήτηση και ανάκτηση δεδομένων που παρήχθησαν στα πλαίσια λειτουργίας του ασύρματου δικτύου αισθητήρων
V. Αναζήτηση χαρακτηριστικών και λοιπών εξορισμού δυνατοτήτων των συστατικών του δικτύου.

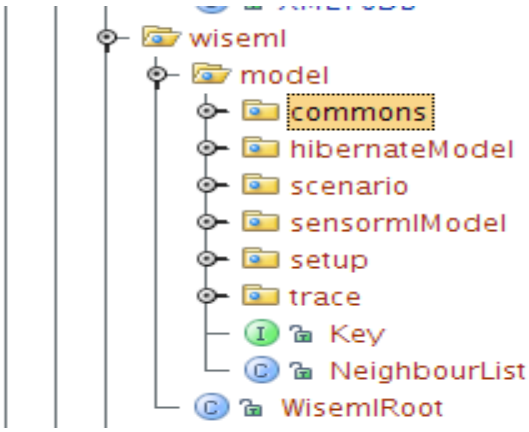
μπορούν να καταφθάνουν συνεχώς στον server και να εξυπηρετούνται σειριακά.
Η περιγραφή των ζητούμενων δεδομένων, επιστρέφεται ανάλογα με το είδος της αίτησης σε GraphML, SensorML ή WiseML format. Προκειμένου να παραχθούν τα παραπάνω XML formats, χρειάζεται ο κατάλληλος μηχανισμός παραγωγής και ανάλυσης XML, έτσι υλοποιήθηκε το ***wiseml interface***. Το Wiseml interface περιέχει τα παρακάτω επιμέρους interfaces:

I. **model:** interface που χρησιμοποιείται για την παραγωγή wiseml, sensorml και graphml αρχείων
II. **WisemlRoot:** κλάση υλοποιημένη σύμφωνα με το Simple XML Framework. Παράγει τον πυρήνα ενός wiseml αρχείου εμφωλεύοντας τα επιμέρους συστατικά της wiseml setup, trace, scenario τα οποία αναπαρίστανται από τα ομώνυμα interfaces στον κώδικα.

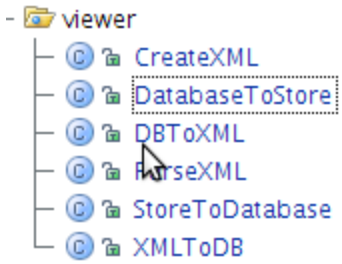
Με τη σειρά του το model interface, υλοποιεί τα παρακάτω interfaces**:**

I. **commons:** Περιλαμβάνει κλάσεις ορισμένες βάσει του Simple XML Framework, και είναι κοινές για όλα τα επιμέρους interfaces. Πιο συγκεκριμένα οι κλάσεις αυτές περιγράφουν τις ακμές και τους κόμβους του δικτύου αισθητήρων όπως επίσης και τις επιμέρους δυνατότητες των κόμβων (capabilities), το configuration και το rssi των ακμών του δικτύου αισθητήρων.
II. **HibernateModel:** Κάθε πίνακας της βάσης δεδομένων, αντιστοιχίζεται σε μια κλάση Hibernate Entity. Κάθε πεδίο του πίνακα, γίνεται τώρα πεδίο της κλάσης στο οποίο ο προγραμματιστής έχει άμεση πρόσβαση με τη βοήθεια των κατάλληλων setter/getter συναρτήσεων. Κάθε hibernate entity υπάγεται σε XML mapping όπως θα αναλυθεί λεπτομερώς και σε επόμενο κεφάλαιο Αφού διαβαστεί το XML configuration μέσω ενός και μόνο Hibernate session, δε χρειάζεται απευθείας πια πρόσβαση στη βάση για queries, και χρησιμοποιείται η HQL με αποτέλεσμα να βελτιστοποιούνται οι χρόνοι προσπέλασης των αποθηκευμένων δεδομένων του δικτύου.

III. **Scenario:** Περιέχει τις απαραίτητες κλάσεις για την παραγωγή της ενότητας scenario, όπως αυτή ορίζεται από τη wiseml.
IV. **sensormlModel:** Περιλαμβάνει τις απαραίτητες κλάσεις για την παραγωγή sensorML format.
V. **Setup:** Περιέχει τις απαραίτητες κλάσεις για την παραγωγή της ενότητας setup, όπως αυτή ορίζεται από τη wiseml.
VI. **Trace:** Περιέχει τις απαραίτητες κλάσεις για την παραγωγή της ενότητας trace, όπως αυτή ορίζεται από τη wiseml.



Ο μηχανισμός που παρέχει μετατροπή από raw data σε ένα από τα βασικά XML formats αλλά και αντίστροφα από τα XML formats σε raw data που αποθηκεύονται στη βάση δεδομένων του συστήματος βρίσκεται στο **interface viewer,** όπως δείχνει και το ακόλουθο σχήμα.



Η κλάση CreateXML χρησιμοποιεί τα αντικείμενα Serialiser και Persister που παρέχει το SimpleXML Framework. Βάσει αυτών, και των κλάσεων του wiseml interface, είναι υπεύθυνο για την παραγωγή XML στο filepath που θα προσδιορίσει ο χρήστης. Στη συνέχεια η κλάση ParseXML, διαβάζει το filepath

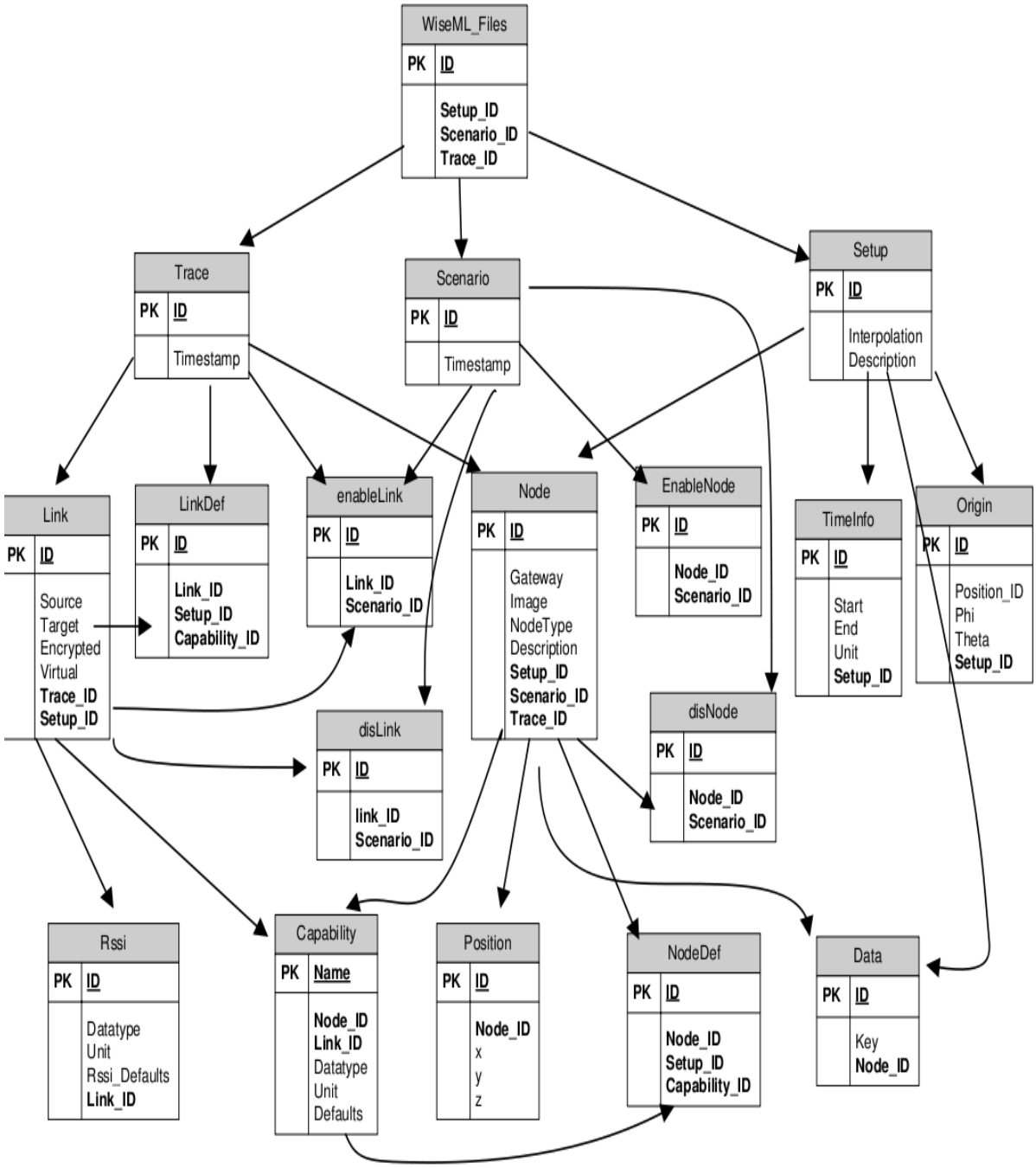
που θα προσδιορίσει ο χρήστης και είναι υπεύθυνη για την συντακτική ανάλυση του GraphML, Wiseml ή SensorML αρχείου που ορίζεται. Εφόσον το κείμενο αξιολογηθεί σημασιολογικά σωστό, η κλάση με χρήση πάλι του αντικειμένου Serialiser, διασπά τα πεδία που περιέχουν χρήσιμη πληροφορία του δικτύου και αποθηκεύει τις τιμές τους σε κατάλληλα Hashmaps. Η κλάση StoreToDatabase, είναι υπεύθυνη για την άμεση μεταφορά της XML πληροφορίας στη βάση δεδομένων και η αντίστροφη DatabaseToStore αναλαμβάνει την άμεση μετατροπή των raw data της βάσης, σε WiseML αρχεία.

Η πλατφόρμα Easysense προκειμένου να αναπαράγει WiseML αρχεία διατηρεί μια βάση δεδομένων 18 πινάκων. Εν συντομία οι πίνακες αυτοί αντιστοιχούν στους:

- *WmlFiles*: Περιέχει αναφορές στις βασικές ενότητες ενός wiseml αρχείου.
- *Trace*: Αποθηκεύει τα συστατικά της ενότητας trace ενός wiseml αρχείου
- *Senario*: Αποθηκεύει τα πολλαπλά σενάρια λειτουργίας του δικτύου αισθητήρων
- *Setup*: Αποθηκεύει τις πολλαπλές συνθήκες αρχικοποίησης μια πειραματικής διαδικασίας πάνω στο εξεταζόμενο δίκτυο αισθητήρων
- *Capability*: Αποθηκεύει τις δυνατότητες των οντοτήτων του δικτύου
- *Node*: Αποθηκεύει περιγραφές για όλους τους κόμβους του συστήματος
- *Link*: Αποθηκεύει περιγραφές για όλους τους συνδέσμους του δικτύου
- *EnableNode*: Χρησιμοποιείται για την καταγραφή των ενεργών κόμβων του συστήματος
- *EnableLink*: Χρησιμοποιείται για την καταγραφή των ενεργών συνδέσμων του συστήματος
- *DisLink*: Χρησιμοποιείται για την καταγραφή των ανενεργών συνδέσμων του συστήματος
- *DisNode*:Χρησιμοποιείται για την καταγραφή των ανενεργών κόμβων του συστήματος
- *LinkDef*: Αποθηκεύονται εξορισμού τιμές για τις ακμές του δικτύου
- *NodeDef*: Αποθηκεύονται εξορισμού τιμές για των κόμβων αισθητήρων
- *Timeinfo*: Περιλαμβάνει χρονολογικές πληροφορίες για την περιγραφή του κάθε πειράματος.
- *Rssi*: Αποθηκεύει τις rssi τιμές των ακμών του δικτύου
- *Position:* Αναφέρεται στην ακριβή τοποθεσία συγκεκριμένου κόμβου του συστήματος.
- *Origin*: Αναφέρεται στην μονάδα μέτρησης και συντεταγμένες των κόμβων του συστήματος
- *Data*: Προσφέρει περαιτέρω περιγραφή για τις οντότητες του

συστήματος

Παρακάτω φαίνεται το ER σχεσιακό διάγραμμα της βάσης, και όλες οι αλληλεπιδράσεις μεταξύ των πινάκων όπως και τα αναλυτικά πεδία έκαστου. Παρατηρούμε πως κάθε πεδίο που ορίζεται από το WiseML format, αντιστοιχίζεται στον κατάλληλο πίνακα της βάσης δεδομένων του συστήματος.



3.3 Uniform Resource Names- URNs

Η πλατφόρμα Easysense αποτελεί μηχανισμό που αναπτύχθηκε για να εξυπηρετήσει τις ανάγκες διαχείρισης, παρουσίασης και χειρισμού των δεδομένων που παράγονται στα πλαίσια λειτουργίας ενός ασύρματου δικτύου αισθητήρων, όπως επίσης και για την εξυπηρέτηση διαδικτυακών λειτουργιών των testbeds, στα πλαίσια του project WISEBED. Ωστόσο, πριν αναλυθούν λεπτομέρειες λειτουργίας, χαρακτηριστικά και αρχιτεκτονική της πλατφόρμας, παρουσιάζεται ένα σχήμα ενιαίας ονομασίας πόρων (***Uniform Resource Name - URN***) για την περιγραφή των ακόλουθων οντοτήτων των δικτύων αισθητήρων:

- **Κόμβοι αισθητήρων,** *(Sensor nodes)*
- **Ικανότητες των κόμβων,** *(Node capabilities)*
- **Χαρακτηριστικά των κόμβων,** *(Node attributes)*
- **Χαρακτηριστικά των ακμών,** *(Edge attributes)*
- **Σημεία ενδιαφέροντος,** *(Points of interest)*

3.3.1 Οι Κόμβοι Αισθητήρων

Ως κόμβοι αισθητήρων, ορίζονται οι μοναδικοί κόμβοι που συνθέτουν τα testbeds αισθητήρων του WISEBED.
Κάθε κόμβος μπορεί να ανήκει μόνο σε ένα δίκτυο αισθητήρων και του αναλογεί ένα μοναδικό αναγνωριστικό (*ID*). Βάσει των παραπάνω ένας κόμβος μπορεί να χαρακτηριστεί μοναδικά παρέχοντας το αναγνωριστικό του, το δίκτυο στο οποίο ανήκει και την επωνυμία του πανεπιστήμιου, ερευνητικού ινστιτούτου ή εταιρίας που το χρησιμοποιεί.

Γενικό παράδειγμα node URN φαίνεται ακολούθως:

<URN> ::= ''urn:wisebed:node:<site>:<network id>:<node id>''

3.3.2 Οι Ικανότητες των Κόμβων Αισθητήρων

Με τον όρο ικανότητες κόμβων, γίνεται αναφορά στις δυνατότητες ανίχνευσης και δράσης που προσφέρει κάθε ένας κόμβος, δηλαδή τα διαφορετικά φαινόμενα που μπορεί να καταγράψει και οι διαφορετικοί τρόποι με τους οποίους μπορεί να αλληλεπιδράσει με το περιβάλλον όπου βρίσκεται. Για παράδειγμα, ένας κόμβος με αισθητήρα θερμοκρασίας μπορεί να καταγράψει τη θερμοκρασία ανεξάρτητα από τις παραμέτρους ακρίβειας κ.λ.π, μπορεί να επικοινωνήσει με άλλες οντότητες χρησιμοποιώντας υπέρυθρες ή μπορεί να χρησιμοποιήσει relay switch για να ανοίξει ή κλείσει το φως του χώρου. Οι ικανότητες των κόμβων περικλείουν και λεπτομέρειες hardware.

Παραδείγματα node capability URNs φαίνονται στη συνέχεια:

urn:wisebed:node:capability:light
urn:wisebed:node:capability:temperature
urn:wisebed:node:capability:humidity

3.3.3 Τα Χαρακτηριστικά των Κόμβων Αισθητήρων

Οι κόμβοι αισθητήρων κάθε ασύρματου δικτύου αισθητήρων του WISEBED, αναπαρίστανται σαν κόμβοι αντίστοιχου γράφου. Αυτοί οι κόμβοι περιγράφονται λεπτομερώς με επιπλέον χαρακτηριστικά όπως για παράδειγμα αν λειτουργούν ως gateway/ base station. *Γενικός τύπος και παράδειγμα node capability URNs φαίνονται στη συνέχεια:*

<URN> ::= ''urn:wisebed:node:attribute:< attribute name >''

urn:wisebed:node:attribute:isGateway

3.3.4 Τα Χαρακτηριστικά των Ακμών

Είναι επιθυμητή η παροχή περιγραφή της τοπολογίας των testbeds σε επίπεδο δικτύου αισθητήρων. Χρησιμοποιείται η έννοια του γράφου για να παρουσιάσει την τοπολογία και τις συνδέσεις ανάμεσα σε διαφορετικούς κόμβους. Τα χαρακτηριστικά των ακμών, παρέχουν πληροφορίες όπως η ποιότητα ενός συνδέσμου ανάμεσα σε δύο κόμβους του ασύρματου δικτύου ασιθητήρων.

Παράδειγμα και γενικός τύπος Edge Attribute URN φαίνεται στη συνέχεια:

<URN> ::= ''urn:wisebed:edge:attribute:< attribute name >''

urn:wisebed:edge:attribute:rssi

3.3.5 Σημεία Ενδιαφέροντος

Τα σημεία ενδιαφέροντος, στα πλαίσια του WISEBED, αναφέρονται σε συγκεκριμένες περιοχές οι οποίες καλύπτονται από τα δίκτυα αισθητήρων του WISEBED. Είναι επίσης χρήσιμο, να δίνεται η δυνατότητα στους χρήστες να αναζητούν πληροφορίες χωρίς να χρειάζεται να γνωρίζουν την ακριβή τοποθεσία των testbeds και κόμβων του δικτύου.

Ο γενικός τύπος του Point of Interest URN φαίνεται στη συνέχεια:

<URN> ::= ''urn:wisebed:poi:< site >:< poi id >''

Χρησιμοποιώντας το παραπάνω σχήμα ονομασίας, μπορεί πια να οριστεί ένα interface για τη λήψη και χειρισμό των δεδομένων του δικτύου. Προφανώς, πριν ο χρήστης μπορέσει να καλέσει τις επιθυμητές συναρτήσεις που παρέχει το interface αυτό, πρέπει να πιστοποιήσει τη γνησιότητα των στοιχείων του και στη συνέχεια να δεσμεύσει έναν επιθυμητό αριθμό κόμβων προκειμένου να εκτελέσει το πείραμά του.

3.4 *Συναρτήσεις που προσφέρει η πλατφόρμα EasySense*

Στην παρούσα ενότητα, παρουσιάζεται το πακέτο των βασικότερων συναρτήσεων που υλοποιεί και προσφέρει η πλατφόρμα easysense στους χρήστες, για το χειρισμό, επεξεργασία, λήψη, παρουσίαση δεδομένων και την εξυπηρέτηση λοιπών διαδικτυακών λειτουργιών. Ως επί το πλείστον η έξοδος των εν λόγω συναρτήσεων επιστρέφεται σε GraphML, SensorML ή WiseML format.

I. ***int[] getReachableNeighborsOfNode (int ID):***

Η συνάρτηση επιστρέφει στο χρήση μια λίστα που αναπαριστά τους γείτονες ενός δεδομένου κόμβου. Τα αποτελέσματα επιστρέφονται σε Wiseml format.

II. ***string getRecords():***

Η συνάρτηση επιστρέφει ένα GraphML αρχείο, το οποίο περιέχει μια πλήρη λίστα των κόμβων αλλά και της γενικότερης τοπολογίας του δικτύου αισθητήρων

III. ***string getEdges():***

Η συνάρτηση επιστρέφει όλες τις τιμές edge URN που αντιστοιχούν στις ακμές του δικτύου αισθητήρων, περιλαμβανομένων των node URNs πηγής και προορισμού και συμβολίζουν τα δύο διακριτά άκρα μιας ακμής.

IV. ***string getNeighbourhood (string urn):***

Η συνάρτηση αυτή παίρνει ως είσοδο την τιμή του URN ενός συγκεκριμένου κόμβου του δικτύου. Σαν έξοδο, επιστρέφει όλα τα URNs των κόμβων που συνδέονται με εκείνον που δόθηκε στην είσοδο, σε WiseMl format.

V. ***string getCapabilities (string urn):***

Η συνάρτηση δέχεται ως είσοδο την τιμή URN ενός συγκεκριμένου κόμβου του δικτύου και επιστρέφει τα URNs όλων των δυνατοτήτων του σε WiseML. Τμήμα της εξόδου της συνάρτησης, εμφανίζεται παρακάτω :

```
<capabilities>
  urn:wisebed:node:capability:light
  urn:wisebed:node:capability:temperature
  urn:wisebed:node:capability:humidity
  urn:wisebed:node:capability:acceleration
  urn:wisebed:node:capability:cpu
  urn:wisebed:node:capability:memory
  urn:wisebed:node:capability:usb
  urn:wisebed:node:capability:radio
</capabilities>
```

VI. ***string getPropertyValueOf (int ID, string propertyName):***

Η συνάρτηση επιστρέφει τη συγκεκριμένη ιδιότητα, κάποιου συγκεκριμένου κόμβου του testbed που ο χρήστης αναζήτησε. Το αποτέλεσμα βρίσκεται σε GraphML format.

VII. ***string describeCapability (string urn):***

Για την συγκεκριμένη δυνατότητα (node capability), της οποίας το URN δίνεται ως είσοδος στη συνάρτηση, επιστρέφεται μια πλήρης περιγραφή σε SensorML. Η περιγραφή ενός capability περιέχει στοιχεία όπως τύπος δεδομένων, μονάδα μέτρησης δεδομένων, στοιχεία κατασκευής κόμβου κ.λ.π

Οι ακόλουθες συναρτήσεις εφαρμόζουν την αντίστροφη διαδικασία σε σχέση με τις προηγούμενες. Πιο συγκεκριμένα λαμβάνουν ως είσοδο ένα αρχείο σε μορφή GraphMl, SensorMl ή WiseM. Αρχικά το κείμενο αναλύεται συντακτικά έστε να συμφωνεί με το format των μοντέλων αναπαράστασης δεδομένων δικτύων αισθητήρων, ύστερα το επεξεργάζονται διαχωρίζοντάς το στα επιμέρους συστατικά τους και κατόπιν επιστρέφουν στο χρήστη τις επιθυμητές πληροφορίες.

VIII. ***string getEdgeList (string graphML):***

Η συνάρτηση λαμβάνει ως είσοδο ένα αρχείο GraphML. Επιστρέφει στο χρήστη λίστα αντικειμένων που αντιστοιχεί στις ακμές του γράφου όπως αυτές περιγράφηκαν στο αρχείο GraphML και παρήχθησαν από την getEdges() συνάρτηση.

IX. ***string getNodeList (string graphML):***

Η συνάρτηση λαμβάνει ως είσοδο ένα αρχείο GraphML. Επιστρέφει στο χρήστη λίστα αντικειμένων που αντιστοιχεί στους κόμβους του γράφου όπως αυτοί περιγράφηκαν στο αρχείο GraphML και παρήχθησαν από την getNeighbourhood() συνάρτηση.

X. ***string getAllRecordsList (string graphML):***

Η συνάρτηση λαμβάνει ως είσοδο ένα αρχείο GraphML. Επιστρέφει στο χρήστη τη λίστα αντικειμένων που περιγράφονται στο αρχείο GraphML και παρήχθησαν από την getRecords() συνάρτηση.

XI. ***string getCapabilitiesList (string graphML):***

Η συνάρτηση λαμβάνει ως είσοδο ένα αρχείο GraphML. Επιστρέφει στο χρήστη λίστα αντικειμένων που αντιστοιχεί στις δυνατότητες των κόμβων του γράφου όπως αυτοί περιγράφηκαν στο αρχείο GraphML, και παρήχθησαν από την getCapabilities() συνάρτηση.

Τα δεδομένα που επιστρέφονται στο χρήστη, διατηρούνται στην τοπική βάση δεδομένων του συστήματος και σε ενδιάμεσο επίπεδο λειτουργίας αντιστοιχίζονται σε HashMaps, προκειμένου η αναζήτηση και προσκόμισή τους στους τελικούς χρήστες να είναι γρήγορη και αποτελεσματική. Επιπλέον η αποθήκευση σε δομές HashMaps συνεισφέρει στη μείωση απώλειας πληροφορίας και στη διατήρηση των δεδομένων καθ όλο τον κύκλο λειτουργίας του συστήματος αισθητήρων.

ΚΕΦΑΛΑΙΟ 4:

Hibernate

ΚΕΦΑΛΑΙΟ 4

4.1 Η τεχνολογία Hibernate

Ανάμεσα στις βασικές τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της πλατφόρμας easysense είναι η ***Hibernate***. Στην παρούσα ενότητα θα παρουσιαστούν αναλυτικά ο σκοπός, οι εντολές, η αρχιτεκτονική, τα πλεονεκτήματα αλλά και τα μειονεκτήματα της συγκεκριμένης τεχνολογίας.

Οι περισσότερες εφαρμογές πρέπει να χρησιμοποιήσουν κάποιου είδους δεδομένα. Παρά το γεγονός ότι το πρόβλημα της διατήρησής τους δεν αποτελεί κάποιο καινούριο ή ασυνήθιστο θέμα για τις εφαρμογές που είναι γραμμένες σε Java ο προγραμματιστής δεν έχει την δυνατότητα απλά να αποφασίσει ποιο δρόμο να ακολουθήσει επιλέγοντας από μια γκάμα παρόμοιων και ευρέως χρησιμοποιούμενων λύσεων.

Για πολλά χρόνια η διατήρηση δεδομένων (persistence) υπήρξε θέμα συζήτησης στην κοινότητα των προγραμματιστών της Java. Πολλοί προγραμματιστές δε συμφωνούν ούτε καν με το θέμα του προβλήματος. Είναι η διατήρηση δεδομένων ένα πρόβλημα που έχει ήδη λυθεί από τη σχεσιακή τεχνολογία (relational technology) και επεκτάσεις (extensions) αυτής, όπως οι αποθηκευμένες διαδικασίες (stored procedures), ή αποτελεί ένα πιο σημαντικό πρόβλημα το οποίο χρειάζε- ται διαχείριση μέσω ειδικών μοντέλων όπως τα Enterprise Java Beans (EJB)·Πρέπει να χρησιμοποιείται SQL κώδικας ακόμη και για τις πιο κλασικές ενέργειες όπως create, read, update, delete ή πρέπει αυτές να γίνονται αυτόματα· Πώς επιτυγχάνεται μεταφερσιμότητα, όταν κάθε σύστημα διαχείρισης βάσεων δεδομένων χρησιμοποιεί τη δική του διάλεκτο· Πρέπει η SQL να εγκαταλειφθεί τελείως και να υιοθετηθεί ένα διαφορετικό σύστημα διαχείρισης βάσεων δεδομένων· Η συζήτηση συνεχίζεται. Παρ όλα αυτά, για την επίλυση του προβλήματος έχει αναπτυχθεί μία προγραμματιστική τεχνική γνωστή ως ‘Αντιστοίχηση μεταξύ Δεδομένου - Σχέσης’ — object/relational mapping (ORM) — η οποία έχει αποκτήσει ευρεία αποδοχή.

Για την διατήρηση των δεδομένων (persistence) στην Java, στα πλαίσια των Enterprice Java Beans 3, έχει οριστεί μία τεχνολογία που ονομάζεται **Java Persistence API**.

Η τεχνολογία αυτή ορίζει ένα περιβάλλον εργασίας (framework) για την αντιστοίχηση οντοκεντρικών μοντέλων (object - oriented models) σε σχήματα παραδοσιακών σχεσιακών βάσεων δεδομένων. Κεντρικός στόχος για την ανάπτυξή της υπήρξε ο ορισμός ενός ενιαίου τρόπου για την διατήρηση δεδομένων σε όλες τις Java εφαρμογές.Η **Hibernate** αποτελεί μία βιβλιοθήκη ελεύθερου λογισμικού για τη γλώσσα προγραμματισμού Java, η οποία υλοποιεί την διεπαφή που ορίζεται στα πλαίσια του Java Persistence API. Είναι μία ολοκληρωμένη λύση στο πρόβλημα της διαχείρισης δεδομένων εκτελώντας όλες τις ενέργειες μεταξύ των εφαρμογών και των σχεσιακών βάσεων δεδομένων απαλλάσσοντας τον προγραμματιστή από επιπρόσθετο κόπο και τον κίνδυνο αστοχιών στην αποτύπωση σχέσεων μεταξύ κλάσεων και σχήματος βάσης. Ο κεντρικός στόχος αυτής είναι η δημιουργία μίας διεπαφής (interface) μεταξύ των διαδεδομένων σχεσιακών βάσεων δεδομένων και του αντικειμενοστραφούς προγραμματισμού. Με άλλα λόγια, επιτρέπει τη χρήση μίας σχεσιακής βάσης δεδομένων ως αντικειμενοστραφή. Για την επίτευξη αυτού, δημιουργούνται αντιστοιχίες μεταξύ των εννοιών του αντικειμενοστραφούς προγραμματισμού, όπως οι συσχετίσεις, η κληρονομικότητα και ο πολυμορφισμός (τα οποία δεν υπάρχουν σε μία σχεσιακή βάση δεδομένων), και των πινάκων και σχέσεων μεταξύ αυτών μίας σχεσιακής βάσης. Με αυτόν τον τρόπο ο προγραμματιστής βλέπει τελικά μία αντικειμενοστραφή βάση δεδομένων, παρ όλο που στην ουσία χρησιμοποιεί μία σχεσιακή.΄Ετσι ο προγραμματιστής χρησιμοποιεί τα αντικείμενα της συγκεκριμένης εφαρμογής, τα τροποποιεί σχετικά με τη λογική της εφαρμογής που αναπτύσσει και τα αποθηκεύει (τροποποιεί, διαγράφει και αναζητά) στη βάση ως αντικείμενα. Σκέπτεται, δηλαδή, με αντικειμενοστραφείς έννοιες και όχι με βάση το σχήμα της σχεσιακής βάσης δεδομένων.΄Ετσι η Hibernate, γνωρίζοντας την αν-τιστοιχία μεταξύ βάσης και λογικής της εφαρμογής, αναλαμβάνει να κατασκευάσει τον κατάλληλο κώδικα τον οποίο στέλνει τελικά στη βάση δεδομένων.΄Επειτα τα αποτελέσματα που επιστρέφονται από τη βάση στην Hibernate δίνονται στην εφαρμογή ως αντικείμενα. Πρόκειται, δηλαδή, ένα ενδιάμεσο επίπεδο μεταξύ εφαρμογής και βάσης δεδομένων.Πιο συγκεκριμένα, η Hibernate Διαχειρίζεται όλα τα create-read-update-delete Operations, χρησιμοποιώντας απλό API και όχι SQL. Δημιουργεί DDL scripts για τη δημιουργία DB schema ενώ εισάγει ευελιξία στην συγγραφή SQL και κλήση αποθηκευμένων διαδικασιών για βελτιστοποίηση της απόδοσης. Υποστηρίζει πάνω από 20 RDBMS (Relational Database Management System). Επίσης, Η βάση μπορεί να αλλάξει με απλή αλλαγή στο χρησιμοποιούμενο configuration file. Συνέπεια των παραπάνω είναι η σημαντική μείωση του χρόνου συγγραφής κώδικα.

4.2 Πλεονεκτήματα

Υπάρχουν πάρα πολλοί λόγοι για τους οποίους κάποιος θα επέλεγε να χρησιμοποιήσει τη Hibernate για την ανάπτυξη μίας εφαρμογής που χρειάζεται κάποιου είδους πρόσβαση σε βάση δεδομένων. Οι πιο σημαντικοί από αυτούς συνοψίζονται παρακάτω :

- Η Hibernate ενδείκνυται για ανάπτυξη πολύπλοκων εφαρμογών. Παρέχει μια σειρά από εργαλεία που επιτρέπουν την εύκολη αποτύπωση αντικειμενοστραφών εννοιών σε σχήματα βάσης ασχέτως του είδους της βάσης που χρησιμοποιείται. Επιτρέπει τη χρήση κληρονομικότητας, πολυμορφισμού και σύνθεσης (composition) στις αντιστοιχιζόμενες κλάσεις και παρέχει μια πολύ ισχυρή γλώσσα ερωτημάτων, την HQL, η οποία επιτρέπει τη χρήση όλων των παραπάνω.

- Μπορεί να βοηθήσει να βελτιωθεί η απόδοση της εφαρμογής. Αυτή η βελτίωση πολύ δύσκολα θα επιτυγχανόταν δημιουργώντας απ΄ ευθείας κώδικα που να εκτελεί τις επιθυμητές λειτουργίες (hand-coding). Η Hibernate δημιουργεί πολύ αποδοτικά ερωτήματα, πράγμα που διασφαλίζει την απόδοση σε πολλές περιπτώσεις. Εκτός αυτού, όμως, υποστηρίζει μία πολύ έξυπνη και αποδοτική πολιτική πρώτου και δεύτερου επιπέδου caching.Με αυτόν τον τρόπο επιτυγχάνεται μεγάλη δυνατότητα κλιμάκωσης. Επίσης δίνει τη δυνατότητα στον προγραμματιστή να επιλέξει το επίπεδο caching που επιθυμεί, όντας πολύ έξυπνη στην πολιτική των write-backs. Επίσης,μπορεί να συνδυαστεί πολύ καλά με κάποια από τα σημαντικότερα λογισμικά caching τόσο ελεύθερου λογισμικού όσο και εμπορικά πακέτα.

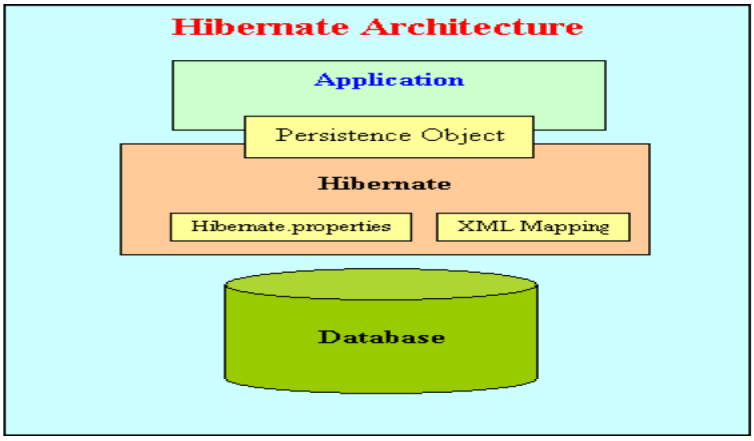
- Επιτρέπει σε μεγάλο βαθμό τη μεταφερσιμότητα των εφαρμογών μεταξύ σχεσιακών βάσεων δεδομένων. Η μόνη διαφοροποίηση στις περισσότερες των περιπτώσεων είναι η αλλαγή μίας μόνο παραμέτρου — της διαλέκτου επικοινωνίας με την βάση. Επιπλέον, οι κλάσεις που αντιστοιχίζονται είναι απλές κλάσεις Java (Plain Old Java Objects — POJOs). ΄Ετσι δε χρειάζεται να είναι απόγονοι μιας πολύπλοκης υποχρεωτικής δομής, με αποτέλεσμα να αυξάνεται ακόμη περισσότερο η μεταφερσιμότητα.

- Παρ όλο που χρειάζεται κάποιο χρόνος για την εκμάθησή της, η Hibernate τελικά αυξάνει σε μεγάλο βαθμό την παραγωγικότητα.

Ο προγραμματιστής ενδιαφέρεται μόνο για τα αντικείμενα. Αυτά αποθηκεύονται στη βάση και ανακτώνται από αυτήν με ελάχιστο κόπο.

4.3 Θέματα Αρχιτεκτονικής

Στην παρούσα παράγραφο συνοψίζονται οι βασικές συνιστώσες που συγκροτούν της αρχιτεκτονική της τεχνολογίας Hibernate. Πρώτα από όλα λοιπόν, η Hibernate ανοίγει η ίδια σύνδεση με τη βάση δεδομένων του συστήματος. Μετατρέπει HQL (Hibernate Query Language) σε database specific statements. Επιπλέον, παραλαμβάνει τα result sets και εκτελεί **mapping** των database specific data σε Java objects τα οποία χρησιμοποιούνται απευθείας από τις εφαρμογές. Η Hibernate "διαβάζει" το database specification από το αρχείο **Hibernate properties**. Βάσει αυτού, γίνεται αυτόματη αντιστοίχηση (mapping) για καθορισμένα Java Objects με τη βοήθεια των δεδομένων που δηλώνονται στο καάλληλο **.hbm** XML file. Στην ακόλουθη εικόνα φαίνεται σχηματικά η αρχιτεκτονική που ακολουθεί η τεχνολογία:

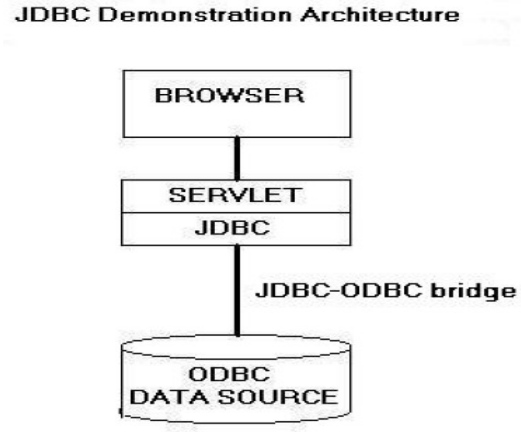


4.4 Συγκριτική παρουσίαση Hibernate και JDBC

Η τεχνολογία Hibernate εμφανίζει σημαντικές διαφορές και πλεονεκτήματα σε σύγκριση με το ευρέος χρησιμοποιούμενο JDBC. Όπως είναι γνωστό, η 'συνηθισμένη' λύση για database connectivity σε Java είναι το **JDBC** API (Java Database Connectivity). Σύμφωνα με αυτή επιτρέπεται στους developers σύνδεση, queries, και update στη βάση δεδομένων χρησιμοποιώντας SQL. Επιτρέπεται επίσης αλληλεπίδραση με διαφορετικά RDBMS, όπως και πρόσβαση στα tables από Java Applications χωρίς να είναι απαραίτητη η γνώση RDBMS λεπτομερειών, με απλή χρήση των **database specific JDBC drivers**. Το interaction του JDBC με το RDBMS είναι απλό. Για να αποκτήσει πρόσβαση ένα application στη βάση χρειάζονται τα ακόλουθα βήματα:

- Άνοιγμα σύνδεσης μς τη βάση δεδομένων
- Χρήση του JDBC driver για την αποστολή queries στη βάση
- Επεξεργασία των επιστρεφόμενων αποτελεσμάτων
- Κλείσιμο της σύνδεσης

Η παραπάνω διαδικασία αποτυπώνεται σχηματικά στην ακόλουθη εικόνα:



Πρώτα από όλα φαίνεται λοιπόν, ότι δουλεύοντας με Object-Oriented software και Relational Database είναι πολύπλοκο έργο με το JDBC, γιατί εμφανίζεται απόκλιση ανάμεσα στον τρόπο αναπαράστασης των δεδομένων σε objects και στον τρόπο που τα 'αντιλαμβάνεται' η βάση.

Έτσι ο προγραμματιστής πρέπει να γράψε παραπάνω κώδικα για να αντιστοιχίσει τα δεδομένα από relational model σε object model. Το πρόβλημα δεν εμφανίζεται στο Hibernate καθώς το mapping γίνεται αυτόματα με XML files. Η διαδικασία ονομάζεται **transparent persistence**. Επιπλέον, Το JDBC υποστηρίζει μόνο native SQL. Είναι δουλειά του προγραμματιστή να επιλέξει το πιο effective query / database access για να εκτελέσει συγκεκριμένο task. Το Hibernate παρέχει την πολύ αποδοτική Hibernate query language (ανεξάρτητη από τον τύπο DB) που έχει την ίδια σύνταξη με SQL και υποστηρίζει polymorphic queries. Ακόμα παρέχει native SQL queries και επιλέγει τον πιο αποδοτικό τρόπο για τη διαχείριση της βάσης, σύμφωνα με το εκάστοτε application. Αξίζει επίσης να σημειωθεί ότι Όταν χρησιμοποιείται JDBC, αλλαγή στη βάση δεδομένων μεταφράζεται σε αλλαγή στη δομή του κώδικα. Αντιθέτως, αλλαγές στην βάση, με χρήση Hibernate μεταφράζονται δυναμικά στον Java κώδικα με απλή τροποποίηση των XML properties files. Επιπλέον, με το JDBC ο προγραμματιστής αναλαμβάνει να μετατρέψει χειροκίνητα resultsets σε Java Objects. Το Hibernate, γλιτώνει χρόνο, κόστος, και συντομεύει τον απαιτούμενο κώδικα, διατηρώντας object-table mapping & επιστρέφοντας τα αποτελέσματα στο application σε μορφή Java Objects. Τέλος σε σύγκριση με το JDBC η Hibernate Εμφανίζει καλή κλιμάκωση με την αύξηση του όγκου των δεδομένων και υψηλή αξιοπιστία. Υποστηρίζει **database versioning** διευκολύνοντας την παράλληλη συγγραφή κώδικα από διαφορετικά άτομα όπως και cashing μειώνοντας τη χρονοβόρα διαδικασία πρόσβασης στο δίσκου

4.5 Mapping Δεδομένων

Κάθε οντότητα της βάσης δεδομένων αντιστοιχίζεται σε ένα XML αρχείο της εφαρμογής που την περιγράφει διεξοδικά. Η διαδικασία αυτή ονομάζεται mapping. Η hibernate προσφέρει τέσσερις τρόπους για την αντιστοίχηση των σχέσεων των mapping objects, ανάλογα με την πολυπλοκότητά τους όπως φαίνεται και στο παρακάτω σχήμα:

- **many-to-one**: ordinary reference to another persistent object. Child object lifetime is *independent* of parent.
- **one-to-many**: a collection of references to another class of mapped objects.
- **one-to-one**: reference to another persistent object. Child object lifetime is *dependent* on parent lifetime.
- **many-to-many**: collection of entities with its own table.
  - Intervening collection (association) table needed.

Τα mappings μπορούν να συμπεριλάβουν τύπους δεδομένων όπως <set>, <list>, <map>, <bag>, <array>, <primitive-array> όπως και κάποιο από τα ακόλουθα interfaces:

- java.util.Map
- java.util.Set
- java.util.SortedMap
  java.util.SortedSet
- java.util.List
- java.util.Collection

Το mapping και η σύνδεση με τη βάση χρησιμοποιούνται για τη δημιουργία SessionFactory αντικειμένου. Το SessionFactory αποτελεί thread-safe cache από compiled mappings για μια βάση. Δημιουργείται μόνο μια φορά, στην αρχή του application (expensive). Το session δηλώνει την επικοινωνία βάσης – application , και κρατά το πρώτο επίπεδο cache των αντικειμένων.

4.6 HQL & Βασικές Εντολές

Η σύνταξη της HQL είναι παρόμοια με της SQL. Αντίθετα με την SQL, είναι database-agnostic, δηλαδή δεν αναφέρεται άμεσα στη βάση δεδομένων του συστήματος. Αντιθέτως, *'καταλαβαίνει'* εννοιες όπως κληρονομικότητα, πολυμορφισμό, ssociations, aggregations, compositions. Παρέχει Java-based API για τη δημιουργία queries. Τα παραγόμενα queries είναι αποτελεσματικά, διότι χρησιμοποιούν conditional logic, αποφεύγοντας messy string manipulation. Χαρακτηριστικά παραδείγματα φαίνονται παρακάτω:

- Selection: *from*, *as*
- Associations and joins: *inner join, left outer join, right outer join, full join*
- Projection: *select, elements*
- Constraints: *where*
- Other constructs: aggregate functions, expressions, order by clauses, group by clauses, polymorphic selections, subqueries

```
• from quizzer.domain.Question as question
• from java.lang.Object as object
• from quizzer.domain.Question as question
    order by question.questionText
• select elements(question.answers)
    from quizzer.domain.Question as question
• select count(question)
    from quizzer.domain.Question as question
• select question.questionText
    from quizzer.domain.Question as question
```

Οι βασικές HQL εντολές παρουσιάζουν μεγάλες ομοιότητες με τις αντίστοιχες SQL και αντιστοιχούν στη δημιουργία αντικείμενου, διαγραφή αντικειμένου, επιλογή αντικειμένου από πληθώρα αντικειμένων και ενημέρωση αντικειμένου.
Ο κώδικας που αντιστοιχεί σε κάθε μια παρουσιάζεται παρακάτω:

- ***Δημιουργία αντικειμένου:***

```
Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();

AuctionItem item = new AuctionItem();
item.setDescription("Batman Begins");
item.setType("DVD");
session.save(item);

tx.commit();
session.close();
```

- ***Ενημέρωση Αντικειμένου***

```
Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();

AuctionItem item =
   (AuctionItem) session.get(ActionItem.class, itemId);
item.setDescription(newDescription);

tx.commit();
session.close();
```

- ***Διαγραφή Αντικειμένου***

```
Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();

AuctionItem item =
   (AuctionItem) session.get(ActionItem.class, itemId);
session.delete(item);

tx.commit();
session.close();
```

- ***Επιλογή Αντικειμένου***

```
Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();

List allAuctions = session.createQuery("
   select item
   from AuctionItem item
        join item.bids bid
   where item.description like 'Batman%'
        and bid.amount < 15
").list();

tx.commit();
session.close();
```

Παράδειγμα των εντολών μέσα στον κώδικα της εφαρμογής μας φαίνεται παρακάτω:

```
String url =  "jdbc:odbc:" + dbName;
List<EmployeeBean> employeeList = new ArrayList<EmployeeBean>(

/* load the jdbc-odbc driver */
class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

/* Open a connection to database */
Connection con = DriverManager.getConnection(url);

/* create Statement object */
Statement stmt = con.createStatement();

/* execute statement */
ResultSet rs = stmt.executeQuery("SELECT * FROM Sells");
while ( rs.next() )
{
      EmployeeBean eb = new Employeebean();
      eb.setName(rs.getString("name"));
      eb.setSalary(rs.getFloat("salary"));
      employeeList.add(eb);
}
```

4.7 Μειονεκτήματα της Hibernate

Παρά τα σημαντικά πλεονεκτήματα και ευκολίες που εισάγει η τεχνολογία στους απανταχού προγραμματιστές, παρουσιάζονται παράλληλα και ορισμένα εμπόδια που ενδεχομένως δυσκολεύουν την ευρεία εξάπλωση και χρησιμοποίησή της. Κάποια από αυτά παρουσιάζονται στην παράγραφο αυτή. Πρώτα από όλα λοιπόν, η Hibernate παρατηρείται οτι εμφανίζει χαμηλό learning curve. Επιπλέον, μπορεί να προκαλέσει overhead σε εφαρμογές που είναι απλές και χρησιμοποιούν βάση που δεν πρόκειται να υποστεί αλλαγές, είτε το μόνο που απαιτούν είναι η τοποθέτηση δεδομένων στα tables αλλά όχι extra SQL, είτε τέλος δεν εμφανίζουν objects mapped σε δύο διαφορετικά tables. Καθώς λοιπόν η Hibernate εισάγει πολλά διαφορετικά layers δεδομένων, για πολύ απλές εφαρμογές διασύνδεσης και αλληλεπίδρασης με βάση δεδομένων προτείνεται η χρήση JDBC. Τέλος, σημειώνεται πως η υποστήριξη Hibernate

στο Internet δεν είναι επαρκή. Όποιος δηλαδή χρειάζεται να διατηρήσει applications που χρησιμοποιούν hibernate, θα πρέπει να μάθει hibernate. Για πολύπλοκα δεδομένα, το mapping από objects σε tables και αντίστροφα μειώνει την απόδοση απαιτώντας υψηλό χρόνο. Δεν υποστηρίζει ορισμένα queries που προσφέρει το JDBC. (Για παράδειγμα δεν υποστηρίζει την τοποθέτηση multiple objects στο ίδιο table με ένα query.)

ΚΕΦΑΛΑΙΟ 5:

Simple XML Framework

ΚΕΦΑΛΑΙΟ 5

5.1 Το Simple XML Framework

Το ***Simple XML Framework*** χρησιμοποιήθηκε στα πλαίσια ανάπτυξης της πλατφόρμας easysense για την δημιουργία και ανάλυση xml αρχείων.

Το ***Simple XML Framework*** αποτελεί ουσιαστικά μια υψηλής απόδοσης πλατφόρμα για Java, η οποία εξυπηρετεί την εμφώλευση και διαμόρφωση XML κειμένων. Σκοπός του, είναι η παροχή ενός XML Framework που να επιτρέπει τάχιστη ανάπτυξη XML configuration και συστημάτων επικοινωνίας. Το συγκεκριμένο πλαίσιο εργασίας, συνεισφέρει στην ανάπτυξη XML συστημάτων με μικρή προγραμματιστική προσπάθεια και χαμηλό ρυθμό σφαλμάτων. Προσφέρει πλήρη εμφώλευση αντικειμένων όπως και αποκωδικοποίησή τους διατηρώντας παράλληλα όλες τις σχετικές αναφορές. Η κεντρική ιδέα του framework, είναι παρόμοια με την C# XML serialization για Java, αλλά συνδυάζει επιπλέον γνωρίσματα για την καταγραφή και αποδοτικό χειρισμό των δεδομένων.

5.1 .1 Χαρακτηριστικά

Τα νέα γνωρίσματα που εισάγει το Simple XML Framework σκιαγραφούνται ως εξής:

- ***Απλό πλαίσιο εργασίας με ισχυρές δυνατότητες***: Το πλαίσιο που χρησιμοποιείται για την παροχή XML serialization είναι ιδιαίτερα απλό στη χρήση ενώ συνδέει πολλαπλά annotations σε ένα μόνο persister object, το οποίο και χρησιμοποιείται για την εγγραφή και ανάγνωση αντικειμένων σε και από XML

- ***Μπορεί να χειριστεί κύκλους στον γράφο***: Η μηχανή persistence έχει τη δυνατότητα χειρισμού κύκλων στο αντικείμενο του γράφου, γεγονός το οποίο επιτρέπει την εμφώλευση σύνθετων, αναδρομικών οντοτήτων. Κατά τον τρόπο αυτό, εξασφαλίζεται και η εύκολη ανάλυση του XML αρχείου.

- ***Δεν απαιτείται configuration:*** Σε αντίθεση με πολλά XML Frameworks για Java δεν υπάρχουν mappings ούτε configuration για την εμφώλευση κατά XML των αντικειμένων, ανεξαρτήτως της πολυπλοκότητας τους. Το XML schema παρουσιάζεται με χρήση field annotations.

- ***Εξαιρετικά γρήγορη ανάπτυξη XML***: Μέσω της χρήσης του Simple XML Framework η ανάπτυξη XML configuration αλλά και συστημάτων επικοινωνίας μπορεί να γίνει εξαιρετικά γρήγορα σε σύγκριση με άλλα framworks όπως τα DOM, SAX, Digester και Xstream.

- ***Υποστηρίζει μετατροπές από και πρός human editable XML***: Ένας από τους βασικότερους στόχους του framework, είναι ότι τα XML δεδομένα που χρησιμοποιούνται για την εμφώλευση και ανάλυση αντικειμένων πρέπει να περιέχουν υψηλού επιπέδου, human readable πληροφορίες. Επιπλέον, όλα τα XML στοιχεία και ιδιότητες που παρέχει η πλατφόρμα, έχουν απλή δομή η οποία μπορεί εύκολα να δημιουργηθεί με έναν μορφοποιητή κειμένου.

- ***Περιλαμβάνει ένα XML templating system***: Για την διευκόλυνση της διαδικασίας ανάλυσης xml δεδομένων, template markers μέσα στα XML στοιχεία μπορούν να αντικατασταθούν από διάφορες μεταβλητές. Η συγκεκριμένη διαδικασία κάνει το σύστημα ιδιαίτερα προσαρμόσιμο σε διαφορετικές συνθήκες λειτουργίας. Επιπρόσθετα βάσει αυτού, το σύστημα μπορεί να χρησιμοποιηθεί ως dynamic configuration system.

5.2 Χρήση του Framework σε Java περιβάλλον

Στην παρούσα παράγραφο, παρουσιάζονται απλοί τρόποι χρήσης των δυνατοτήτων της πλατφόρμας, για γρήγορη, συντακτικά ορθή, δημιουργία και ανάλυση xml, σε περιβάλλον προγραμματισμού Java. Δομικό στοιχείο της πλατφόρμας, αποτελούν τα annotations, τα οποία ενημερώνουν το σύστημα για τον τρόπο εμφώλευσης των πεδίων ενός xml αντικειμένου.

- **<u>Εμφώλευση Απλού Αντικειμένου</u>**:

Για να επιτευχθεί η εμφώλευση (serialization) ενός αντικειμένου σε XML, μια σειρά annotations πρέπει να τοποθετηθούν εντός του συγκεκριμένου αντικειμένου. Θεωρούμε ένα παράδειγμα αντικειμένου με τρία annotations:

1. Το πρώτο στη σειρά, αναφέρεται στο Root πεδίο, και η ύπαρξή του στον κώδικα είναι υποχρεωτική
2. Τα δύο επόμενα είναι προαιρετικά και στο συγκεκριμένο παράδειγμα αναφέρονται στο περιεχόμενο ενός μηνύματος και
3. Το τελευταίο προσδιορίζει ένα μοναδικό αναγνωριστικό στο αντικείμενο

Με βάση τα παραπάνω ο κώδικας για την εμφώλευση των συγκεκριμένων πεδίων και annotations σε XML διαμορφώνεται ως εξής:

```
@Root
public class Example {

   @Element
   private String text;

   @Attribute
   private int index;

   public Example() {
      super();
   }

   public Example(String text, int index) {
      this.text = text;
      this.index = index;
   }

   public String getMessage() {
      return text;
   }

   public int getId() {
      return index;
   }
}
```

Με τον παραπάνω κώδικα, ορίστηκαν τα επιθυμητά elements και attributes του προς δημιουργία XML αρχείου, και στη συνέχεια το μόνο που απομένει για την αυτόματη δημιουργία των επιθυμητών πληροφοριών είναι η χρήση του αντικειμένου ***Serializer, που προσφέρει το Simple XML Framework*** όπως φαίνεται στο παρακάτω snippet***:***

```
Serializer serializer = new Persister();
Example example = new Example("Example message", 123);
File result = new File("example.xml");
serializer.write(example, result);
```

Με βάση τα πεδία που δηλώθηκαν μέσω των annotations, και των συναρτήσεων που διαθέτει το αντικείμενο Serializer, η παραγόμενη XML έχει την ακόλουθη μορφή:

```
<example index="123">
<text>Example message</text>
</example>
```

Ένα τυπικό WiseML αρχείο που παράγεται από το Easysense χρησιμοποιώντας το παραπάνω Framework έχει το ακόλουθο format και συνδυασμό πεδίων:

```
<wiseml version="1.0" xmlnss="http://wisebed.eu/ns/wiseml/1.0">
  <setup>
    <origin>
      <phi>5.0</phi>
      <theta>0.0</theta>
    </origin>
```

```
    <timeinfo>
      <start>9/7/2009</start>
      <end>19/12/2010</end>
      <unit>seconds</unit>
    </timeinfo>
<interpolation>cubic</interpolation>
<description>wiseml example</description>
```



```
    <defaults For="node">
      <node id="urn:wisebed:node:tud:M4FTR">
        <position>
          <x>1.23</x>
          <y>1.56</y>
          <z>1.77</z>
        </position>
        <gateway>gw1</gateway>
        <image>blinkfast.tnode</image>
        <nodetype>fast blinking node</nodetype>
        <description>TNide v4</description>
        <capability>
          <name>urn:wisebed:node:capability:temp</name>
          <datatype>integer</datatype>
          <unit>C</unit>
          <defaults>20</defaults>
         </capability>
```

```
    </node>

  </defaults>
  <link source="urn:wisebed:node:tud:330006" target="urn:wisebed:node:tud:330009">

    <encrypted>true</encrypted>

    <virtual>false</virtual>

    <rssi datatype="decimal" unit="dBam" rssiDefault="-120"/>
```



```
    <data key="urn:wisebed:node:capability:speed" value="12"/>
  </link>

</setup>

<scenario id="scenario_1">

  <timestamp value="23/09/09 00:00:01">

  <enablenode id="urn:wisebed:node:tud:M4FTR"/>

  <enablelink source="urn:wisebed:node:tud:33000" target="urn:wisebed:node:tud:330009"/>

  <enablelink source="urn:wisebed:node:tud:33000" target="urn:wisebed:node:tud:330001"/>

  <enablelink source="urn:wisebed:node:tud:33000"
  <node id="urn:wisebed:node:tud:M4FTR">

    <position>

      <x>1.23</x>

      <y>1.56</y>

      <z>1.77</z>

    </position>

    <gateway>gw1</gateway>

    <image>blinkfast.tnode</image>
```

```
<nodetype>fast blinking node</nodetype>

<description>TNide v4</description>

<data key="urn:wisebed:node:capability:temp" value="30"/>

  ...
</wiseml>
```


ΚΕΦΑΛΑΙΟ 6:

Η ΤΕΧΝΟΛΟΓΙΑ JAX-WS

ΚΕΦΑΛΑΙΟ 6

6.1 Ορισμός της Τεχνολογίας Jax-Ws

Η τεχνολογία Jax-Ws χρησιμοποιήθηκε κατά την ανάπτυξη των πλατφόρμας Easysense, εξυπηρετώντας το σχεδιασμό του data & application server όπως και του client αναζήτησης δεδομένων του δικτύου. Πριν αναλυθεί η χρήση της τεχνολογίας στον κώδικά της πλατφόρμας easysense, θα παρουσιαστούν οι δυνατότητες και ο τρόπος λετουργίας της Jax-Ws.

Αρχικά λοιπόν, σημειώνεται ότι η JAX-WS αντιστοιχεί σε Java API για XML Web Services . Είναι μια τεχνολογία για την κατασκευή web services και εφαρμογές πελατών (που θα χρησιμοποιούν την υπηρεσία) ,όπου επικοινωνούν χρησιμοποιώντας XML. Επιτρέπει στους developers να κατασκευάζουν message-oriented καθώς και RPC-oriented web services .

6.1.1 Web Services

Σύμφωνα με το W3 (World Wide Web Consortium) , Web Service ορίζεται ένα σύστημα λογισμικού σχεδιασμένο να υποστηρίζει διαλειτουργική αλληλεπίδραση μεταξύ συστημάτων σε ένα δίκτυο. Συχνά τα Web Services είναι απλά Web APIs (Application Programming Interfaces) ,στα οποία μπορουν να έχουμε πρόσβαση μέσα από ένα δίκτυο ,όπως το Internet. Επίσης είναι υπηρεσίες

που εκτελούνται σε απομακρυσμένο σύστημα που «παρέχει» τις υπηρεσίες και περιμένουν αιτήσεις για τη λειτουργία τους από τους πελάτες (client-server model) . Όταν χρησιμοποιούμε τον όρο Web Service θα αναφερόμαστε σε clients (πελάτες) και servers (εξυπηρετητές) που επικοινωνούν με XML messages που ακολουθούν το SOAP standard. Το WSDL είναι μια περιγραφή των λειτουργιών-υπηρεσιών που προσφέρει ο server (εξυπηρετητής)

6.2 Πρωτόκολλα WS

Τα πρωτόκολλα που χρησιμοποιούνται από τα web services ανήκουν σε μια από τις παρακάτω κατηγορίες:

- ***XML:*** προσφέρει ένα text-based τρόπο να περιγράφει και να εφαρμόζει μια ιεραρχική δομή δέντρου σε πληροφορίες. Στις Web Services χρησιμοποιούν την XML ,διότι είναι εύκολη στη χρήση και μπορεί να χρησιμοποιηθεί από διαφορετικές πλατφόρμες

- ***SOAP (Simple Object Access Protocol) :*** είναι ένα πρωτόκολλο για ανταλλαγή XML-based μηνυμάτων σε δίκτυα υπολογιστών και κυρίως χρησιμοποιώντας το HTTP , HTTPs. Επίσης προσφέρει ένα απλό και ελαφρύ μηχανισμό για ανταλλαγή δομημένων πληροφοριών μεταξύ εφαρμογών σε ένα κατανεμημένο περιβάλλον χρησιμοποιώντας XML. Το SOAP προσφέρει ένα τρόπο να επικοινωνούν εφαρμογές που τρέχουν σε διαφορετικά λειτουργικά συστήματα, τεχνολογίες και γλώσσες προγραμματισμού . Το SOAP αποτελείται από 3 μέρη:

    - ***Το SOAP envelope construct*** *που ορίζει τι είναι μέσα στο message , σε ποιόν αναφέρεται ,*
    - ***το SOAP encoding rules*** *και το*
    - ***SOAP RPC***

- ***WSDL (Web Services Description Language) :*** είναι μια XML- based γλώσσα για να περιγράφει τα web services καθώς και το πως θα έχουν πρόσβαση σε αυτές. Αντίστοιχα πρέπει να δημιουργείται ένα αρχείο WSDL στο οποίο θα καταγράφονται όλες οι πληροφορίες για το ίδιο το service, το πού βρίσκεται ο server (σε ποια διεύθυνση), ποιες λειτουργίες υποστηρίζει καθώς και πως δέχεται και πως επιστρέφει τα

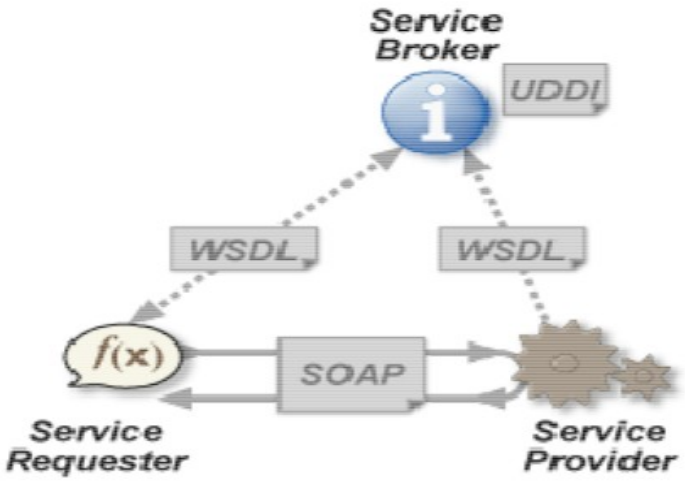
δεδομένα για κάθε λειτουργία

- ***UDDI (Universal Description , Discovery and Intergration)*** : αποτελεί ένα πρωτόκολλο καταχώρησης για web services. Στην ουσία είναι ένα πρωτόκολλο για δημοσίευση και ανακάλυψη web services ,δηλαδή επιτρέπει σε εφαρμογές να αναζητούν τις web services είτε κατά τη συγγραφή τους, είτε κατά την λειτουργία τους.

- Κάθε καταχώρηση περιέχει το WSDL αρχείο και τη διεύθυνση που λειτουργεί η υπηρεσία στο Internet. Επιπλέον σε κάθε καταχώρηση υπάρχουν και διάφορες άλλες πληροφορίες για την υπηρεσία που σχετίζονται με τον ιδιοκτήτη της , την πολιτική του κτλ.

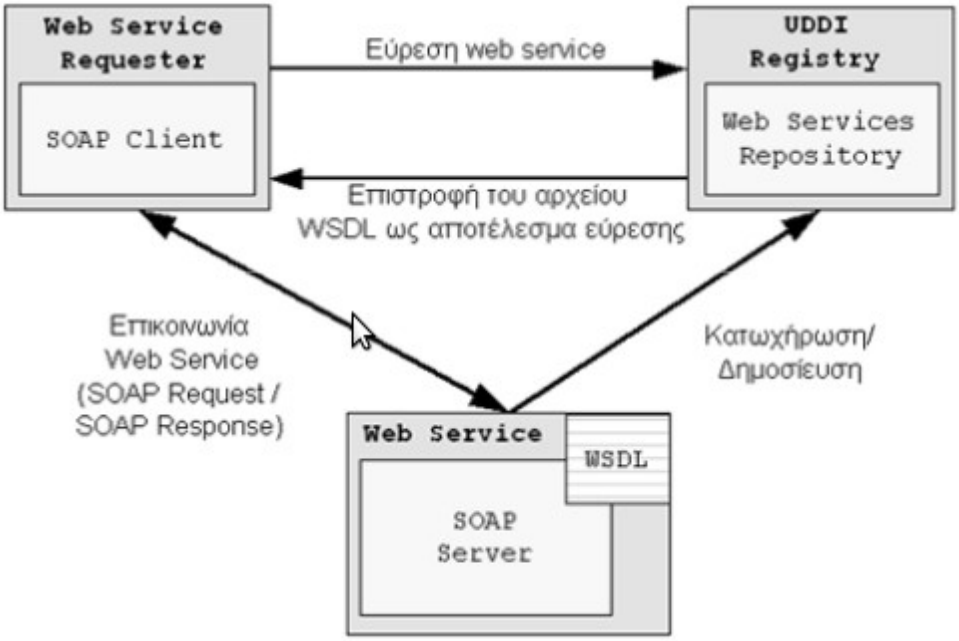
6.3 Περιγραφή λειτουργίας WS

Στο κεφάλαιο αυτό περιγράφεται ο τρόπος λειτουργίας των Java Web services. Αρχικά λοιπόν, ο Service Requester αναζητά για κάποιο Web Service στο UDDI , που στην ουσία είναι ένα interface που περιέχει πληροφορίες όλων των καταχωρημένων Web Services και επιστρέφει τα αποτελέσματα σε αυτόν . Η αναζήτηση και εύρεση των Web Services από το UDDI γίνεται με τη βοήθεια της WSDL , η οποία είναι ένας τρόπος για να περιγράφουν τα Web Services τις υπηρεσίες-λειτουργίες που προσφέρουν . Η διαδικασία, φαίνεται σχηματικά και στην παρακάτω εικόνα:



Αν ο Service Requester θελήσει να χρησιμοποιήσει κάποια λειτουργία-υπηρεσία ενός Web Service επικοινωνεί με τον Service Provider μέσω του πρωτοκόλλου SOAP , το οποίο σκοπός του είναι να στέλνει το αίτημα στον Service Provider με μια συγκεκριμένη μορφή (message envelope format).

Τέλος ο Service Provider μόλις εχτεί το αίτημα θα απαντήσει πάλι μέσω του SOAP και με την ίδια μορφή στον Service Requester . Ακόμα πιο αναλυτικά η συνολική λειτουργία ενός Java Web Service περιγράφεται στο ακόλουθο σχήμα:



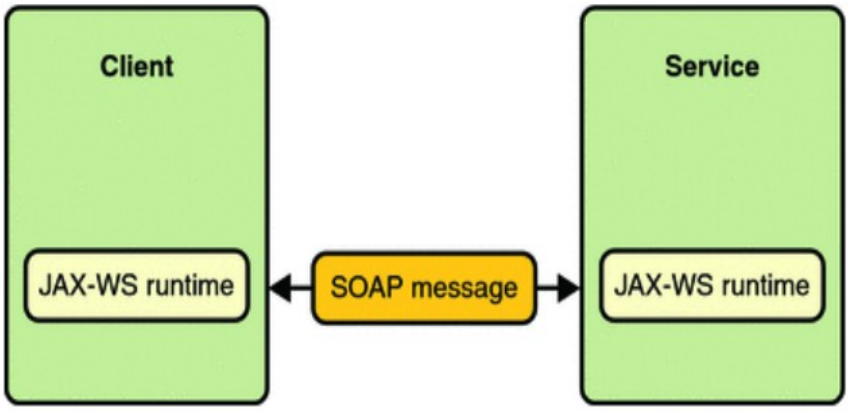
Στην JAX-WS, ένα web service operation invocation αναπαρίσταται από ένα XML-based πρωτόκολλο ,όπως το SOAP . Το SOAP specification ορίζει την envelope structure, encoding rules, and conventions για την αναπαράσταση των κλήσεων και των απαντήσεων των web services (calls and responses) . Αυτά τα calls και τα responses αποστέλλονται μέσω SOAP μηνυμάτων (XML files) over HTTP . Εξετάζοντας τις δύο διαφορετικές πλευρές server και client σημειώνονται τα παρακάτω:

- ***Server Side***: Ο developer καθορίζει τις λειτουργίες του web service ,ορίζοντας μεθόδους σε ένα interface γραμμένο σε Java . Επίσης ο

developer μπορεί να γράψει μια ή περισσότερες κλάσεις που υλοποιούν αυτές τις μεθόδους

- ***Client Side***: Ένας client δημιουργεί ένα proxy (a local object που αναπαριστά την υπηρεσία) και στη συνέχεια απλά καλεί μεθόδους στο proxy . Με την JAX-WS, ο developer δεν δημιουργεί ούτε κάνει parse σε SOAP messages. Το JAX-WS runtime system αναλαμβάνει να μετατρέψει τα API calls και responses σε και από SOAP messages.

Ωστόσο, Η αρχή για την ανάπτυξη μιας JAX-WS web service είναι η Java κλάση javax.jws.WebService .Ο **@WebService** σχολιασμός ορίζει την κλάση σαν ένα web service endpoint Ένα service endpoint interface ή μια service endpoint implementation (SEI) είναι μια Java interface ή κλάση αντίστοιχα, που δηλώνει τις μεθόδους τις οποίες ο πελάτης μπορεί να καλέσει στην υπηρεσία. Το interface δεν είναι αναγκαίο όταν δημιουργούμε ένα JAX-WS endpoint. Η web service implementation class έμμεσα ρίζει ένα SEI. Η επικοινωνία μεταξύ ενός JAX-WS Web Service και ενός πελάτη φαίνεται σχηματικά ως εξής:



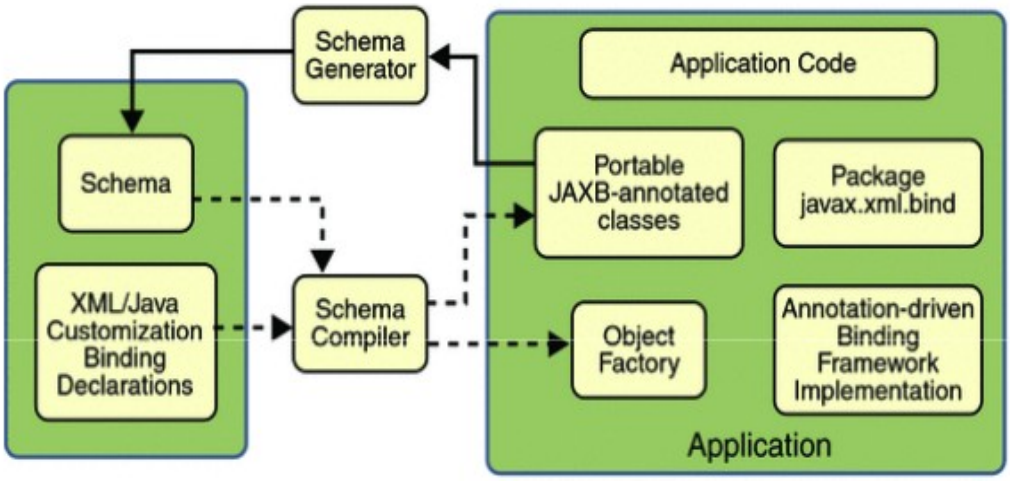
Τα βασικά βήματα για την δημιουργία ενός web service και client είναι :

- *Code the implementation class*
- *2.Compile the implementation class.*
- *3.Use wsgen to generate the artifacts required to deploy the*

- *service*
- *4.Package the files into a WAR file*
- *5.Deploy the WAR file. The web service artifacts (which are used*
- *to communicate with clients) are generated by the Application*

- *Server during deployment*
- *6.Code the client class*
- *7.Use wsimport to generate and compile the web service*
- *artifacts needed to connect to the service*
- *8.Compile the client class*
- *9.Run the client*

Η αρχιτεκτονική της JavaTM για XML Binding (JAXB) προσφέρει ένα γρήγορο και εύκολο τρόπο να κάνει κανείς σύνδεση μεταξύ XML schemas και Java representations, κάνοντας εύκολο στους Java developers να ενσωματώσουν XML data και processing functions σε εφαρμογές Java ώς ένα μέρος της διεργασίας. Η JAXB διαθέτει μεθόδους μετατροπή XML instance documents σε Java content trees και στη συνέχεια για μετατροπή τωνJava content trees πάλι σε XML instance documents. Η JAXB επίσης διαθέτει ένα τρόπο για τη δημιουργία XML schema από Java objects. Η αρχιτεκτονική της JAXB φαίνεται σχηματικά στην παρακάτω εικόνα:



Αναφορικά με τους τρόπους δημιουργίας WS, υπάρχει μεγάλη ποικιλία. Οι βασικότεροι παρουσιάζονται παρακάτω:

- **Ξεκινώντας από ένα WSDL file (top-down approach)**

    - *Δημιουργία κλάσεων χρησιμοποιώντας wsimport*
    - *WS interface*
    - *WS implementation skeleton class*
    - *Build ,deploy and test*

- **Ξεκινώντας από Plain Old Java Objects (POJO) (bottom-up approach)**

    - *Annotate POJO*
    - *Build and deploy*
- *Αυτόματη δημιουργία WSDL αρχείου*

Ένα απλό παράδειγμα κώδικα WS, εμφανίζεται παρακάτω:

6.4 Πλεονεκτήματα

Η τεχνολογία Jax-WS παρουσιάζει μια σειρά πλεονεκτημάτων που διευκολύνουν τους προγραμματιστές κατά την ανάπτυξη εφαρμογών. Τα πιο σημαντικά από αυτά, εμφανίζονται παρακάτω:

- *Με την JAX-WS, οι clients και τα web services είναι ανεξάρτητα του συστήματος (platform independence) εξαιτίας της Java*

.

- *Επίσης η JAX-WS δεν είναι restrictive , δηλαδή ένας JAX-WS client μπορεί να έχει πρόσβαση σε ένα web service ,το οποίο δεν τρέχει σε Java platform, και αντίστροφα*

- *Η ευελιξία αυτή έγκειται στο γεγονός ότι η JAX-WS χρησιμοποιεί τεχνολογίες που ορίζονται από το World Wide Web Consortium (W3C) όπως : HTTP, SOAP, WSDL*

- *Επειδή τα SOAP messages είναι πολύπλοκα, η JAX-WS API αποκρύπτει την πολυπλοκότητα από τον developer και δίνει τη δυνατότητα στον developer να ασχοληθεί με την λειτουργικότητα του web service*

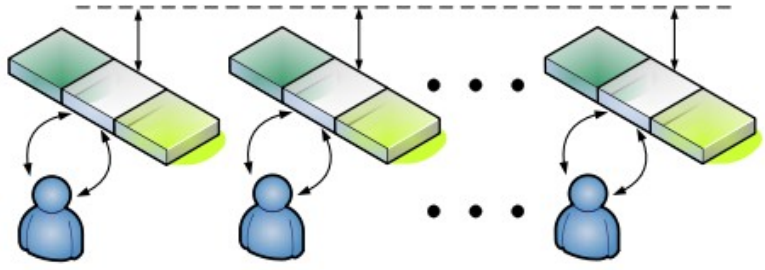
ΚΕΦΑΛΑΙΟ 7:

Peer-toPeer **συστήματα & πρωτόκολλα *JXTA***

ΚΕΦΑΛΑΙΟ 7

7.1 Συστήματα Peer-to-Peer

Ένα peer-to-peer μοντέλο βασίζεται σε ένα σύνολο από κόμβους (hosts) που μπορούν να επικοινωνήσουν μεταξύ τους άμεσα μέσω του δικτύου, όπως δείχνει και η παρακάτω εικόνα:



Ο παραπάνω ορισμός υπονοεί ότι:

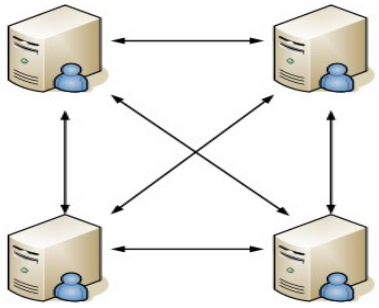
• Δεν υπάρχει ανάγκη για την ύπαρξη ενός κεντρικού κόμβου ο οποίος θα πρέπει να εξυπηρετεί τους υπόλοιπους κόμβους των χρηστών. Με άλλα λόγια, δεν απαιτείται ένας κεντρικός εξυπηρετητής.

• Οι κόμβοι σε ένα peer-to-peer μοντέλο έχουν την ίδια λειτουργικότητα και ίδια δικαιώματα. Στην περίπτωση αυτή, κάθε κόμβος διατηρεί ένα αντίγραφο της διαμοιραζόμενης εφαρμογής. Αυτό σημαίνει ότι το αντίγραφο θα πρέπει να συντονίζει τις ενέργειες που γίνονται τοπικά (local) με αυτές που γίνονται στους άλλους κόμβους (remote), και να τις συγχρονίζει.
Το βασικό πλεονέκτημα ενός peer-to-peer μοντέλου είναι ότι δεν υπάρχει κεντρικό σημείο αστοχίας (central point of failure). Ωστόσο, η επεκτασιμότητα ενός peer-to-peer συστήματος είναι μάλλον περιορισμένη, αφού όταν για παράδειγμα γίνει μια αλλαγή στην κατάσταση ενός αντικειμένου, ο υπεύθυνος κόμβος θα πρέπει να στείλει ένα μήνυμα για τον συγχρονισμό της κατάστασης του αντικειμένου σε όλους τους άλλους κόμβους (peers).

Το μειονέκτημα εδώ είναι ότι η λύση αυτή απαιτεί από τους χρήστες αρκετή υπολογιστική ισχύ. Υπάρχουν δύο τύποι μοντέλων επικοινωνίας με τη χρήση peer-to-peer τοπολογιών, ανάλογα με τον τύπο σύνδεσης:

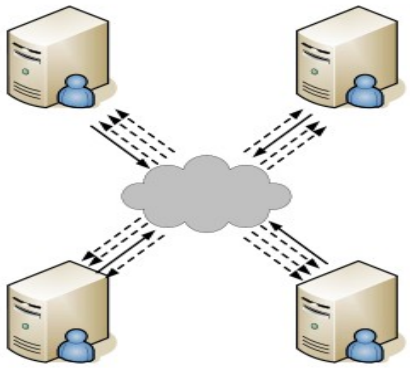
A. Peer-to-peer Τοπολογία με Unicast Δίκτυο

Σε αυτό το είδος peer-to-peer τοπολογίας (Σχήμα 21.2) ο αριθμός των μηνυμάτων που μεταδίδονται αυξάνεται με O(N2), επειδή κάθε peer θα πρέπει να στείλει ένα μήνυμα ενημέρωσης σε όλους τους υπόλοιπους κόμβους. Σχηματικά φαίνεται στην ακόλουθη εικόνα:



B. Peer-to-peer Τοπολογία με Multicast Δίκτυο

Στην περίπτωση του multicasting, τα peers μπορούν να στείλουν τα μηνύματά τους σε ένα υποσύνολο από κόμβους με μια μόνο μετάδοση. Σε αυτό το είδος peer-to-peer τοπολογίας, ο αριθμός των μηνυμάτων που μεταδίδονται αυξάνεται με O(N), που αποτελεί μια σημαντική βελτίωση σε σχέση με την προηγούμενη τοπολογία. Σχηματικά φαίνεται στο παρακάτω σχήμα:



7.2 Η Πλατφόρμα της JXTA

Ένα τελευταίο κομμάτι της αρχιτεκτονικής της πλατφόρμας **Easysense**, αποτελεί η πλατφόρμα του peer-to-peer (p2p) δικτύου της JXTA. Η JXTA, προσφέρει στο σύστημα χρήσιμες εγγενείς λειτουργίες, όπως η δημοσίευση των ιδιοτήτων και των πόρων ενός κόμβου, ο έλεγχος της κατάστασης των κόμβων και η οργάνωση των κόμβων σε ιδιωτικές και δημόσιες ομάδες με διαφορετικά προνόμια σε κάθε μια ως προς την πρόσβαση στις διαθέσιμες υπηρεσίες. Η επιλογή της ενσωμάτωσης ενός αρθρώματος peer-to-peer δικτύωσης στο Easysense, έγινε λόγω των πολύ καλών ιδιοτήτων αυτής της τεχνολογίας δικτύων, όπως είναι :

1. Η κατανεμημένη αρχιτεκτονική της, όπου οι κόμβοι μπορούν να λειτουργούν ως πελάτες και ως εξυπηρετητές. Αποφεύγονται έτσι τα προβλήματα της κεντρικοποιη-μένης αρχιτεκτονικής πελάτη-εξυπηρετητή, όπου η εσφαλμένη λειτουργία του εξυπηρετητή καθιστά το δίκτυο αναξιόπιστο ή άχρηστο.

2. Η δυνατότητα αυτό-οργάνωσης, καθώς οι κόμβοι μπορούν να ανακαλύψουν άλλους κόμβους και να συγκροτήσουν ένα αυτόνομο p2p δίκτυο.

3. Η εύκολη σύνδεση στο δίκτυο ακόμα και κόμβων που βρίσκονται πίσω από συστήματα προστασίας (firewall).

4. Ο αποδοτικός χειρισμός συχνών συνδέσεων και αποσυνδέσεων κόμβων. Αυτό κάνει το δίκτυο πολύ ευέλικτο και αξιόπιστο ακόμα και σε δίκτυα επικάλυψης μεγάλης κλίμακας ως προς το πλήθος των επιμέρους δικτύων που τα απαρτίζουν.

5. Η χρήση κατανεμημένων πινάκων για την δεικτοδότηση των διαμοιραζόμενων πόρων.

Οι δομές αυτές ευνοούν τη γρήγορη αναζήτηση και εντοπισμό επιθυμητών πόρων και χειρίζονται αποδοτικά την εξυπηρέτηση αιτήσεων από μεγάλο αριθμό κόμβων δικτύου. Η JXTA (Juxtapose) είναι ένα σύνολο από προδιαγραφές για πρωτόκολλα που ρυθμίζουν τις λειτουργίες ενός peer-to-peer δικτύου.

Ο καθορισμός των προδιαγραφών αυτών ξεκίνησε από την Sun Microsystems το 2001 και ακολουθεί το μοντέλο ανάπτυξης ανοικτού κώδικα υπό την άδεια Apache Software License. Μια δραστήρια και αφοσιωμένη κοινότητα προγραμματιστών προσφέρει στη συνεχή εξέλιξη και βελτίωσή του πλαισίου της JXTA. Στόχος της είναι να καθοριστούν τα πλαίσια λειτουργίας και οι βασικές υπηρεσίες ενός γενικού σκοπού peer-to-peer δικτύου με τέτοιο τρόπο, ώστε κάθε συσκευή με δυνατότητες δικτυακής επικοινωνίας να μπορεί να συνδεθεί στο δίκτυο αυτό ανεξάρτητα από την τοπολογία του δικτύου όπου βρίσκεται ή το λειτουργικό της σύστημα, και να συνεργαστεί αποδοτικά με τους υπόλοιπους κόμβους.

Τα πρωτόκολλα της JXTA στηρίζονται ουσιαστικά στον ορισμό της δομής και την ανταλλαγή XML μηνυμάτων.΄Ετσι, είναι εύκολο να υλοποιηθεί σε διάφορες γλώσσες προγραμματισμού και ήδη έχουν παρουσιαστεί αρκετά ώριμες υλοποιήσεις σε Java, J2ME, C και C++. Σε ένα peer-to-peer δίκτυο της JXTA οι κόμβοι του δικτύου (peers) θα πρέπει να υλοποιούν ένα βασικό υποσύνολο τουλάχιστον από τα πρωτόκολλα της JXTA για να μπορούν να επικοινωνούν και να συνεργάζονται με τους υπόλοιπους peers. Επιπρόσθετα, με την ένταξή τους στο δίκτυο της JXTA εγγράφονται στην προκαθορισμένη δημόσια ομάδα κόμβων (peer group) του δικτύου JXTA.΄Επειτα, τους δίνεται η δυνατότητα να συγκροτήσουν επιμέρους ομάδες στις οποίες η πρόσβαση στις υπηρεσίες μπορεί να είναι επίσης ανοιχτή ή να επιτρέπεται μόνο σε πιστοποιημένα μέλη της ομάδας. ΄Ενας κόμβος μπορεί να ανήκει σε περισσότερες από μία ομάδες, ενώ ενίοτε ορίζονται ομάδες με ιεραρχικό τρόπο, όπου μια ομάδα μπορεί να περιέχει μικρότερες υπό-ομάδες. Τυπικά μια εφαρμογή δημιουργεί και χρησιμοποιεί μια δική της ομάδα, ώστε να παρέχει τις υπηρεσίες της μόνο σε όσους κόμβους της

JXTA εντάσσονται σε αυτή. Οι πόροι μιας ομάδας κόμβων της JXTA διαφημίζονται και γίνονται γνωστοί σε όλους τους κόμβους της ομάδας. Ως πόρους JXTA εννοούμε τα στοιχεία των κόμβων, των διαθέσιμων peer group στα οποία μπορούν να ενταχθούν και των υπηρεσιών που μπορούν να χρησιμοποιήσουν. Κάθε ένας από τους πόρους διαθέτει ένα αναγνωριστικό ταυτότητας που εξασφαλίζεται να είναι μοναδικό ανάμεσα στα υπόλοιπα σε όλο το δίκτυο. Ειδικοί κόμβοι που ονομάζονται κόμβοι συνάντησης και αναμετάδοσης (rendezvous/relay) αναλαμβάνουν λειτουργίες δρομολόγησης, διαχείρισης των διαφημίσεων και ενημέρωσης του κατανεμημένου μητρώου για τους πόρους του δικτύου. Επιπλέον χειρίζονται θέματα συνδεσιμότητας με κόμβους που βρίσκονται πίσω από τοπικά δίκτυα τοπολογίας NAT ή πίσω από κάποιο τοίχος προστασίας firewall. Σε μια ομάδα κόμβων της JXTA είναι απαραίτητη η παρουσία τουλάχιστον ενός τέτοιου ειδικού κόμβου.

\

Οι βασικές εγγενείς υπηρεσίες της JXTA είναι οι ακόλουθες :

1. Η υπηρεσία ανακάλυψης (discovery service), την οποία οι κόμβοι χρησιμοποιούν για να ανακαλύψουν τους πόρους του δικτύου και να διαφημίσουν τους δικούς τους.

2. Η υπηρεσία ¨διασωλήνωσης¨ (pipe service) και η υπηρεσία ανάλυσης (resolver service). Αυτές οι υπηρεσίες δημιουργούν και διαχειρίζονται τις συνδέσεις για την ανταλλαγή δεδομένων μεταξύ των κόμβων.

3. Υπηρεσία συμμετοχής και πρόσβασης σε μια ομάδα (membership service). Με αυτή την βασική υπηρεσία ρυθμίζονται θέματα εμπιστοσύνης, ασφάλειας και προστασίας ιδιωτικών πόρων μέσα σε μια ομάδα κόμβων (peer group).

4. Η υπηρεσία παρακολούθησης (monitoring service). Πρόκειται για την υπηρεσία που επιτρέπει την παρακολούθηση της κατάστασης ενός κόμβου μιας ομάδας.

7.3 Χρήση JXTA στο Easysense

Η αρχιτεκτονική της πλατφόρμας Easysense, χρησιμοποιεί την υποδομή των πρωτοκόλλων Jxta ώστε να προσφέρει βασικές peer to peer υπηρεσίες στους χρήστες της. Έτσι, με τις παραπάνω λειτουργικότητες πολλά ετερογενή ασύρματα δίκτυα αισθητήρων μπορούν να ενωθούν σε ένα μεγαλύτερης έκτασης χρησιμοποιώντας από κοινού τις προσφερόμενες υπηρεσίες.
Η πλατφόρμα της JXTA αναλαμβάνει να χειριστεί το μεγαλύτερο μέρος από ζητήματα όπως δρομολόγηση των μηνυμάτων, διαφήμιση των πόρων του συστήματος και καταγραφή τους σε κατανεμημένο μητρώο, χειρισμός αποσυνδέσεων και επανασυνδέσεων, ενώ το Easysense υποβοηθούμενο από αυτή την υποδομή, αλληλεπιδρά με τις υπηρεσίες της JXTA σε ανώτερο επίπεδο.
Μέσω του δικτύου που προσφέρει το Easysense, δημιουργείται ένα ευρύτερο δίκτυο αποτελούμενο από τα επιμέρους δίκτυα, που επιθυμούν να χρησιμοποιήσουν τα web services που προσφέρει η πλατφόρμα, όπως αυτά περιγράφονται στο κεφάλαιο 3.

Οι κόμβοι που συμμετέχουν σε αυτή τη δικτύωση χωρίζονται στις ακόλουθες τρεις κατηγορίες:

I. **κόμβους δρομολόγησης**
II. κόμβους πύλης
III. κόμβους χρήστη

Οι κόμβοι πύλης αναλαμβάνουν την διαφήμιση των δυνατοτήτων και υπηρεσίες που μπορούν να προσφέρουν. Οι υπηρεσίες που διαφημίζονται είναι τα web services εντοπισμού και περιγραφής των δεδομένων του δικτύου αισθητήρων σε κάποιο κατάλληλο XML format όπως GraphML, SensorML ή WiseML.
Οι κόμβοι δρομολόγησης αναλαμβάνουν το ρόλο της προώθησης των μηνυμάτων και της εξασφάλισης της συνδεσιμότητας όλων των κόμβων που επιθυμούν να ενταχθούν στο γενικό δίκτυο. Τέλος, οι κόμβοι χρηστών είναι τα σημεία όπου εκτελούνται απλές εφαρμογές δικτύων αισθητήρων.
Όλοι οι κόμβοι ανήκουν στον ίδιο βασικό τύπο Peer και επιτελούν κάποιες κοινές λειτουργίες, κυρίως κατά την εκκίνησή τους. Αυτές, αφορούν κυρίως στην αρχικοποίηση της πλατφόρμας της JXTA, τη σύνδεση στην προκαθορισμένη ομάδα net peer group και την καταχώρηση των υπηρεσιών του.
Ο ρόλος του κόμβου πύλης στο δίκτυο Easysense, αναλαμβάνει την διαφήμιση των πόρων του, στέλνοντας περιοδικά μια περιγραφή τους. Αναλαμβάνει την αρχικοποίηση των συστατικών της Jxta όπως και τη σύνδεση και αποσύνδεση peers με το δίκτυο του Easysense. Επίσης, περιμένει να δεχτεί ερωτήσεις και εντολές και τελικά θα στείλει τις αντίστοιχες απαντήσεις και μηνύματα

καταστάσεως πίσω στους ενδιαφερόμενους peer κόμβους.

Ο κόμβος δρομολόγησης αναλαμβάνει τις λειτουργίες δρομολόγησης στο δίκτυο . Κατ΄ ουσίαν προέρχεται από τον τύπο κόμβου Rendezvous της JXTA . Οι κόμβοι αυτού του τύπου διατηρούν ένα κατανεμημένο πίνακα αντιστοιχίσεων (distributed hash tables) για να βρίσκουν τη βέλτιστη διαδρομή που πρέπει να ακολουθήσει ένα μήνυμα προς τον προορισμό του στο δίκτυο. Επιπλέον, είναι απαραίτητοι καθώς εξασφαλίζουν τη συνδεσιμότητα στο δίκτυο για κόμβους που είναι κρυμμένοι πίσω από τείχη προστασίας (firewalls) ή τοπολογίες που χρησιμοποιούν το NAT πρωτόκολλο. Οι Rendezvous κόμβοι είναι απαραίτητοι για την ύπαρξη και λειτουργία ενός δικτύου που βασίζεται στην πλατφόρμα της JXTA. Συνεπώς, για το δίκτυο του Easysense απαιτείται η ύπαρξη τουλάχιστονενός κόμβου δρομολόγησης, ο οποίος και θα δημιουργήσει το βασικό επιμέρους peer group του Easysense μέσα στο δημόσιο προκαθορισμένο peer group της JXTA. Στην υλοποίηση χρησιμοποιείται το aeolus jar, το οποίο παρέχει τα παραίτητα utilities.

Τέλος, ως κόμβος χρήστη στο Easysense μπορεί να λειτουργεί οποιαδήποτε συσκευή με δυνατότητες δικτύωσης. Κατά συνέπεια, μια εφαρμογή που χρησιμοποιεί την υποδομή του Easysense, απαιτεί μόνο την ύπαρξη ενεργού κόμβου-χρήστη στο ίδιο μηχάνημα,ο οποίος θα μπορεί να συνδεθεί με το δίκτυο του συστήματος.

ΚΕΦΑΛΑΙΟ 8:

Συμπεράσματα

ΚΕΦΑΛΑΙΟ 8

Τα ασύρματα δίκτυα αισθητήρων, έχουν μια πλειάδα σημαντικών και συνεχώς αυξανόμενων εφαρμογών στην κοινωνία του σήμερα. Η εξέλιξη των

δικτύων αισθητήρων σε συνδυασμό με την επιθυμία υλοποίησης περίπλοκων και περισσότερο ολοκληρωμένων εφαρμογών, οδήγησε σε δίκτυα που περιέχουν ετερογενείς κόμβους, όπως για παράδειγμα κόμβους με διαφορετική αρχιτεκτονική, διαφορετικούς αισθητήρες, με διαφορετικές επεξεργαστικές και επικοινωνιακές ικανότητες ή και κόμβους με δυνατότητες κίνησης ή επιτέλεσης εργασιών (actuators) σε απόκριση εντολών που λαμβάνουν από το κέντρο ελέγχου.

Πολλές από τις εφαρμογές των ασύρματων δικτύων αισθητήρων, ειδικά όσες σχετίζονται με pervasive computing, για παράδειγμα εφαρμογές υγείας και μέριμνας, παράγουν συνεχώς καταιγιστικές ποσότητες δεδομένων για μεγάλες χρονικές περιόδους. Καθώς οι συσκευές αισθητήρων έχουν περιορισμένες υπολογιστικές δυνατότητες και μνήμη, η αποδοτική διαχείριση, αξιοποίηση, αποθήκευση και δυναμική παρουσίαση της τεράστιας ποσότητας πληροφορίας που παράγεται κατά τη λειτουργία των ασύρματων δικτύων αισθητήρων είναι κρίσιμης σημασίας για την ερευνητική κοινότητα και την εξαγωγή των επιθυμητών επιστημονικών συμπερασμάτων. Βάσει των παραπάνω, απαραίτητη κρίνεται, η ύπαρξη αποδοτικού αρχιτεκτονικού μοντέλου χειρισμού δεδομένων, το οποίο θα αντισταθμίζει τους ακραίους περιορισμούς των συσκευών αισθητήρων σε πόρους όπως ενέργεια, αποθήκευση και επεξεργαστική ισχύ.

Συνεισφέροντας προς αυτή την κατεύθυνση, στα πλαίσια της παρούσας μεταπτυχιακής εργασίας, αναπτύχθηκε η πλατφόρμα Easysense, ένα κλιμακούμενο εργαλείο που επιτρέπει στους επιστήμονες και ερευνητές την on-demand δυνατότητα ανάλυσης, παρατήρησης, ανάκτησης και αποθήκευσης της πληροφορίας των γεωγραφικά κατανεμημένων κόμβων αισθητήρων, αποτυπώνοντας παράλληλα τις χωρο-χρονικές συνιστώσες της. Μελετώντας νέες τεχνολογίες όπως Hibernate, Java Web Services και αξιοποιώντας κλασσικές δομές δεδομένων π.χ hashtables, επιτυγχάνονται βελτιστοποιημένοι χρόνοι αξιόπιστων δοσοληψιών με τη βάση δεδομένων του συστήματος και γρήγορη επικοινωνία με τους χρήστες. Χρησιμοποιώντας pervasive computing τεχνικές, τα δεδομένα καταγράφονται στο σύστημα και διατηρούνται καθ όλη τη διάρκεια ζωής του WSN. Μοναδικά αναγνωριστικά (Unique Naming Resources) χρησιμοποιούνται εξυπηρετώντας την παρακολούθηση γεγονότων και πόρων του συστήματος και την άμεση επιστροφή αναλυτικής περιγραφής τους προς κάθε ενδιαφερόμενο.

Άμεση συνέπεια των παραπάνω, είναι ότι πια καθίσταται δυνατή, η δύσκολη διαδικασία καταγραφής και παρατήρησης της διαρκώς μεταβαλλόμενης τοπολογίας των κατανεμημένων ασύρματων δικτύων αισθητήρων.

Όσο αφορά στην διαδικασία παρουσίασης των δεδομένων αισθητήρων στους χρήστες σε υψηλό επίπεδο, υπάρχοντα μοντέλα όπως SensorML και GraphML αξιολογήθηκαν, ερευνήθηκαν και επεκτάθηκαν. Η ύπαρξη XML

παραγώγων με πεδία και στοιχεία που απευθύνονται αποκλειστικά σε δίκτυα αισθητήρων είναι εξαιρετικά σημαντική καθώς προσφέρει στους επιστήμονες ένα κοινό παρονομαστή για την επεξεργασία, σύγκριση, και ανάλυση δεδομένων από πολλά ετερογενή ασύρματα δίκτυα. Ωστόσο, παρά τη σημαντικότητά τους, παρόμοια εργαλεία είναι περιορισμένα. Με αυτή την αφετηρία, στα πλαίσια της παρούσας εργασίας προτάθηκε και μελετήθηκε ένα ευρύτερο μοντέλο παρουσίασης δεδομένων η WiseML. Το format αυτό, είναι ιδιαίτερα εύρωστο και επεκτάσιμο καθώς επιτρέπει πολλαπλά επίπεδα ανάλυσης πληροφορίας, συσχετίζει τα δεδομένα με χωρο-χρονικές συνιστώσες ενώ δίνει τη δυνατότητα συγκεντρωτικής, αναλυτικής παρουσίασης όλων των πειραματικών διαδικασιών, περιβαλλοντικών συνθηκών και αποτελεσμάτων που προέκυψαν κατά τη διάρκεια ζωής ενός WSN. Απεικονίζει τα δεδομένα με τρόπο άμεσο, αρθρωτό και εύληπτο. Έτσι, σε ένα μόνο αρχείο οι ερευνητές μπορούν να αποθηκεύουν όλη την ωφέλιμη πληροφορία του προς μελέτη συστήματος, μειώνοντας σημαντικά το χρόνο ανάλυσης και επεξεργασίας πολλαπλών ετερογενών δεδομένων. Η πλατφόρμα Easysense, παρέχει μηχανισμούς παραγωγής, συντακτικής ανάλυσης και παρουσίασης των επιθυμητών δεδομένων σε όλα τα διαθέσιμα μοντέλα περιγραφής WSN πληροφορίας: SensorML, GraphML, WiseML.

Τέλος, αξιοποιώντας την ευέλικτη υποδομή των peer-to-peer δικτύων και δίνοντας έμφαση στην επεκτασιμότητα της αρχιτεκτονικής του, το Easysense, δίνει τη δυνατότητα να προσφέρει τις υπηρεσίες του σε πολλαπλά ετερογενή δίκτυα αισθητήρων με συσκευές και σταθμούς εργασίας πάνω από το διαδίκτυο, διασυνδέοντάς τα ουσιαστικά σε ένα ευρείας κλίμακας.

Οι πολλές νέες εφαρμογές των δικτύων αισθητήρων εγείρουν κάποια άξια αναφοράς κοινωνικά, νομικά και θεσμικά Σητήματα. Το κυριότερο εξ αυτών είναι το ζήτημα της ασφάλειας της διαδιδόμενης πληροφορίας, με όλες τις συνιστώσες διαστάσεις του, όπως η εξασφάλιση της προστασίας ευαίσθητων δεδομένων και η εξουσιοδοτημένη πρόσβαση σε διαβαθμισμένη πληροφορία. ΄Ενα άλλο σημαντικό κοινωνικό θέμα αφορά την πρόσβαση στις νέες υπηρεσίες των δικτύων αισθητήρων ως κοινωνικό αγαθό. Είναι ακόμα ασαφές το τοπίο ως προς τις αρχές που θα διαχειρίζονται τις υπηρεσίες και την διακινούμενη πληροφορία, ως προς το πώς αυτές θα κοστολογούνται, ποιοι θα έχουν δικαίωμα να τις εμπορεύονται και τι μηχανισμοί ελέγχου θα θεσπιστούν για τις διαδικασίες αυτές. Οι εξελίξεις στο πεδίο των δικτύων αισθητήρων αναμένεται να συνεχίσουν να είναι ραγδαίες και εντυπωσιακές. Είναι ευκταίο οι εφαρμογές τους να χρησιμοποιηθούν προς όφελος του ανθρώπου παρά για την εκμετάλλευση ή την καταστροφή του.

ΠΑΡΑΡΤΗΜΑ

Στο παρών παράρτημα παρουσιάζονται χαρακτηριστικά δείγματα πηγαίου κώδικα που αναπτύχθηκε για την πλατφόρμα EasySense. Οι βιβλιοθήκες που χρησιμοποιήθηκαν μπορούν να βρεθούν στα ακόλουθα packages:

- **org.apache.log4j.Logger;**
- **org.hibernate.HibernateException;**
- **org.hibernate.Session;**
- **org.hibernate.SessionFactory;**
- **org.hibernate.Query;**
- **org.hibernate.cfg.AnnotationConfiguration;**
- **org.simpleframework.xml.Serializer;**
- **org.simpleframework.xml.load.Persister;**
- **javax.jws.WebMethod;**
- **javax.jws.WebService;**
- **javax.xml.ws.Endpoint;**
- **java.io.*;**
- **static java.lang.System.err;**
- **java.util.List;**
- **java.util.Iterator;**
- **java.util.ArrayList;**

Server Side Code Snippet:

```
@WebService()
public class FindNeighbour extends ParseXML {

    /**
     * Create a sessionFactory for Hibernate queries.
     */
    private static SessionFactory ourSessionFactory;

    /**
     * The parameters of the class.
     */
    private static String[] argv;

    /**
     * Apache Log4J logger.
     */
    protected static final Logger LOGGER = Logger.getLogger("server");

    static {

        try {
            ourSessionFactory = new AnnotationConfiguration().
                    configure("hibernate.cfg.xml").
                    buildSessionFactory();
        }
        catch (Exception ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }


    /**
     * This non-Web function creates a Hibernate Session.
     *
     * @return
     * @throws org.hibernate.HibernateException
     *
     */
    public static Session getSession() throws HibernateException {
        return ourSessionFactory.openSession();
    }
```

```
/**
 * This funtion reads an xml file, and returns its contents as a String.
 *
 * @param path
 * @return string path
 */
public final String readXml(final String path) {
    String text;
    text = "";
    try {

        final FileInputStream fstream = new FileInputStream(path);

        final DataInputStream inputStream;
        inputStream = new DataInputStream(fstream);
        final BufferedReader bufread;
        bufread = new BufferedReader(new InputStreamReader(inputStream));
        String strLine;

        //Read File Line By Line
        while ((strLine = bufread.readLine()) != null) {
            text = text + strLine + "\n";
        }
        inputStream.close();

    } catch (Exception e) { //Catch exception if any
        err.println("Error: " + e.getMessage());
    }
    return text;
}

/**
 * This function overwrites every occurence of certain substring,
 * with a new pattern in a file.
 *
 * @param fname
 * @param oldPattern
 * @param replPattern
 */
public static void readReplace(final String fname, final String oldPattern,
                               final String replPattern) {
    String line;
    final StringBuffer sbuf = new StringBuffer();
```

```
        try {
            final FileInputStream fis = new FileInputStream(fname);
                                final BufferedReader reader = new BufferedReader(new
InputStreamReader(fis));
            while ((line = reader.readLine()) != null) {
                line = line.replaceAll(oldPattern, replPattern);
                sbuf.append(line).append('\n');
            }
            reader.close();
            final BufferedWriter out;
            out = new BufferedWriter(new FileWriter(fname));
            out.write(sbuf.toString());
            out.close();
        }
        catch (Throwable e) {
            err.println("*** exception ***");
        }
    }

    public final SensormlRoot getField(final String code,
                                final String value,
                                final String definition,
                                final String fieldName) {



        /**
         * Initiallize uom.
         */

        final Unit uom = new Unit();
        uom.setCode(code);
        uom.setValue(value);

        /**
         * Initiallize QuantityRange.
         */

        final QuantityRange range = new QuantityRange();
        range.setDefinition("urn:ogc:def:property:" + definition);
        range.setUom(uom);

        /**
```

```
     * Initiallize field.
     */

    final SensormlRoot field = new SensormlRoot();
    field.setName(fieldName);
    field.setRange(range);
    field.setRole("urn:orc:property:" + definition);


    return field;

}

/**
 * This function produces a file with SensorML content,
 * describing the capabilities of a certain node.
 *
 * @param path
 * @param fieldList
 * @param definition
 * @param description
 * @param capabilityName
 */
public final void writeSensorML(final String path,
                                final List<SensormlRoot> fieldList,
                                final String definition,
                                final String description,
                                final String capabilityName) {

    final DataRecord data = new DataRecord();
    data.setDefinition("urn:orc:def:property:" + definition);
    data.setDescription(description);
    data.setFieldList(fieldList);


    final Serializer serializer = new Persister();
    final Capabilities example = new Capabilities(capabilityName, data);
    final File result = new File(path);
    try {
        serializer.write(example, result);
    }
    catch (Exception e) {
        e.printStackTrace();
    }
```

```
        readReplace(path, "F10", ":");
    }

    /**
     * The Web method getNode takes
     * as input a node id from the client and returns
     * its neighbourhood.
     */
    @WebMethod
    public String getNode(final String nodeid) {
        DatabaseToStore dbstore = new DatabaseToStore();
        List<Node> nodeList = new ArrayList<Node>();

        final Session session = getSession();
        session.beginTransaction();

        final List gwList = session.createQuery("select node.gateway "
                + "from NodeEntity node where node.id='" + nodeid + "' ").list();

        String gateway = (String) gwList.get(0);

            final Query query = session.createQuery("from NodeEntity  as node where
node.gateway ='" + gateway + "'");
        final Iterator iterator;
        for (iterator = query.iterate(); iterator.hasNext();) {
            final NodeEntity nodentity;
            nodentity = (NodeEntity) iterator.next();
            System.out.println(nodentity.getId());

            dbstore.getNodeToStore(nodentity.getId());
            Node node = NodeStore.getInstance().get(nodentity.getId());
            System.out.println(node.getID());
            nodeList.add(node);
            CreateXML create = new CreateXML();
              create.writeXMLnodeList("/home/kleopatra/Documents/xml/neighbours.xml",
nodeList);
        }

        final String result = readXml("/home/kleopatra/Documents/xml/neighbours.xml");


        return result;
```

```
    }

    /**
     * The Main function generates the web service.
     *
     * @param argv
     */
    public static void main(final String[] argv) {
        FindNeighbour.argv = argv;
        final FindNeighbour implementor = new FindNeighbour();
        final String address = "http://localhost:9000/FindNeighbour";
        Endpoint.publish(address, implementor);
    }

}
```

Client Side Code Snippet :

```
public class Client {



/** Simple Client Calling a specific web service
**/

    public static void main(final String[] argv) {

        FindNeighbour service;

        service = new FindNeighbourService().getFindNeighbourPort();

        final ParseXML parse = new ParseXML();


 final String neighbouResult = service.getNode("urn:wisebed:node:tud:M4AOP2OV");
}
```

Στην παρούσα παράγραφο, παρουσιάζεται ενδεικτικό τμήμα κώδικα όπου χρησιμοποιήθηκε η τεχνολογία Hibernate, για την υλοποίηση της πλατφόρμας EasySense. Οι βιβλιοθήκες που χρησιμοποιήθηκαν μπορούν να βρεθούν στο package:

- javax.persistence.*;

```
/**
 * Describes a Capability Entity with Hibernate.
*/

@Entity
@Table(catalog = "wisemIDb", name = "capability")
public class CapabilityEntity {

    /**
     * The name of the entity.
     */
    private String name;

    @Id
    @Column(name = "name")


    public String getName() {
        return name;
    }

    /**
     * The function sets the name value.
     *
     * @param name
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * The node id of the entity.
     */
    private String nodeid;
```

```
@Basic
@Column(name = "nodeid")

/**
 * The function retrieves the node id value.
 */
public String getNodeid() {
    return nodeid;
}

/**
 * The functions sets the node id value.
 *
 * @param nodeid
 */
public void setNodeid(String nodeid) {
    this.nodeid = nodeid;
}

/**
 * Default values of the entity.
 */
private String defaults;
/**
 * The functions retrieves the default values.
 *
 * @return string.
 */

public String getDefaults() {
    return defaults;
}

/**
 * The function sets the default values.
 *
 * @param defaults
 */

public void setDefaults(String defaults) {
    this.defaults = defaults;
}
```

```java
/**
 * The link id of the entity.
 */
private int linkid;

@Basic
@Column(name = "linkid")
/**
 * The function retrieves the link id value.
 */
public int getLinkid() {
    return linkid;
}

/**
 * The function sets the link id value.
 *
 * @param linkid
 */

public void setLinkid(int linkid) {
    this.linkid = linkid;
}


/**
 * The datatype of the entity.
 */
private String datatype;

@Basic
@Column(name = "datatype")

/**
 * The function retrieves the datatype value.
 */
public String getDatatype() {
    return datatype;
}
```

```java
/**
 * The function sets the datatype value.
 *
 * @param datatype
 */
public void setDatatype(String datatype) {
    this.datatype = datatype;
}

/**
 * The unit of the entity.
 */
private String unit;

@Basic
@Column(name = "unit")

/**
 * The function retrieves the unit value.
 */
public String getUnit() {
    return unit;
}

/**
 * The function sets the unit value.
 * @param unit
 */
public void setUnit(String unit) {
    this.unit = unit;
}


@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;

    CapabilityEntity that = (CapabilityEntity) o;

    // if (lidefaultid != that.lidefaultid) return false;
    if (linkid != that.linkid) return false;
```

```
        // if (nodefaultid != that.nodefaultid) return false;
        if (nodeid != that.nodeid) return false;
            if (datatype != null ? !datatype.equals(that.datatype) : that.datatype != null)
return false;
        if (name != null ? !name.equals(that.name) : that.name != null) return false;
        if (unit != null ? !unit.equals(that.unit) : that.unit != null) return false;

        return true;
    }
    @Override
    public int hashCode() {
        int result = name != null ? name.hashCode() : 0;
        //result = 31 * result + nodeid;
        result = 31 * result + linkid;
        result = 31 * result + (datatype != null ? datatype.hashCode() : 0);
        result = 31 * result + (unit != null ? unit.hashCode() : 0);
        // result = 31 * result + nodefaultid;
        // result = 31 * result + lidefaultid;
```

Στην παρούσα ενότητα, παρατίθεται τμήμα κώδικα όπου χρησιμοποιήθηκε το Simple XML Framework, κατά την ανάτπυξη της πλατφόρμας Easysense. Οι χρησιμοποιούμενες βιβλιοθήκες μπορούν να βρεθούν στα packages:

- ***org.simpleframework.xml.Attribute***
- ***org.simpleframework.xml.Element;***
- ***org.simpleframework.xml.ElementList;***
- ***java.util.List***
- ***wiseml.model.commons.Capability;***
- ***wiseml.model.Key;***

```
    /**
 * This class describes a Link entity, required in a wiseml file.
 */


public class Link implements Key {
    /**
     * The link identity.
     */

    private int id;
```

```
/**
 * The edge's source.
 */

@Attribute
private String source;

/**
 * The edge's target.
 */

@Attribute
private String target;


@Element
private String encrypted;

@Element
private String virtual;

@Element
private Rssi rssi

* List with the capability elements of the entity.
 */

@ElementList(inline = true)
private List<Capability> capabilityList;


/**
 * Requires function for deserializing objects.
 */
public Link() {
    super();
}
```