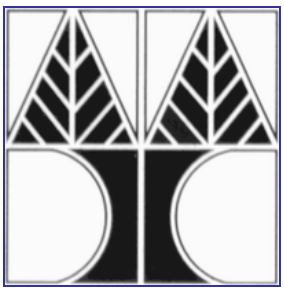
ΔΙΑΤΡΙΒΗ ΜΑΣΤΕΡ

Κινητικότητα για Έλεγχο Κάλυψης σε Ασύρματα Δίκτυα Αισθητήρων

Έλενα Κακουλλή

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ



ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2009

ΠΕΡΙΛΗΨΗ

Η παρούσα Διατριβή Μάστερ παρουσιάζει μια πληθώρα επιστημονικών άρθρων στο τομέα των ασύρματων δικτύων αισθητήρων με κύρια κατεύθυνση τον έλεγχο της κάλυψης με τη βοήθεια της κινητικότητας. Μέσα από αυτή την μελέτη έχουν κατηγοριοποιηθεί και αναλυθεί αλγόριθμοι οι οποίοι δημιουργήθηκαν για να επιλύσουν το πρόβλημα της κάλυψης σε διαφορετικούς τύπους ασύρματων δικτύων αισθητήρων.

Υπάρχουν πάρα πολλά χαρακτηριστικά που διακρίνουμε μέσα από τις τεχνικές/πρωτόκολλα και μπορούμε έτσι να παρατηρήσουμε ομοιότητες και διαφορές μεταξύ τους. Για κάθε συγκεκριμένη κατάσταση σε μια εφαρμογή ενός ασύρματου δικτύου αισθητήρων μπορεί να κρίνεται διαφορετικός καταλληλότερος αλγόριθμος. Αυτή η επιλογή υποβοηθείται όταν μπορούμε να διαχωρίσουμε τα μοντέλα που υπάρχουν σε ορισμένες κατηγορίες και συνδυασμό διαφόρων χαρακτηριστικών.

Επίσης μέσα στην μελέτη αυτή έγινε υλοποίηση δύο αλγορίθμων με βάση ορισμένα κριτήρια επιλογής και σύγκριση τους μέσα από διάφορες εκτελέσεις τους. Οι αλγόριθμοι αυτοί, ο AM (Active Model) και ο SR (Snake-like cascading Replacement process), εφαρμόζονται σε κινητά ασύρματα δίκτυα αισθητήρων και στόχος τους είναι η επίτευξη της πλήρης κάλυψης και συνδετικότητας μέσα στο δίκτυο. Η σύγκριση των δύο αυτών αλγορίθμων έδειξε πως ο SR μπορεί να κριθεί καταλληλότερος αλγόριθμος για αυτού του τύπου δίκτυα, γιατί μπορεί με ένα αποτελεσματικότερο και αποδοτικότερο τρόπο να επιτύχει τον στόχο του.

Κινητικότητα για Έλεγχο Κάλυψης σε Ασύρματα Δίκτυα Αισθητήρων

Έλενα Κακουλλή

Η Διατριβή αυτή

Υποβλήθηκε προς Μερική Εκπλήρωση των

Απαιτήσεων για την Απόκτηση

Τίτλου Σπουδών Master

σε Προηγμένες Τεχνολογίες Πληροφορικής

στο

Πανεπιστήμιο Κύπρου

Συστήνεται προς Αποδοχή

από το Τμήμα Πληροφορικής

Μάιος, 2009

ΣΕΛΙΔΑ ΕΓΚΡΙΣΗΣ

Διατριβή Master

Κινητικότητα για Έλεγχο Κάλυψης σε Ασύρματα Δίκτυα Αισθητήρων

Παρουσιάστηκε από

Έλενα Κακουλλή

Ερευνητικός Σύμβουλος

_____

Δρ. Βάσος Βασιλείου

Μέλος Επιτροπής

_____

Δρ. Ανδρέας Πιτσιλλίδης

Μέλος Επιτροπής

_____

Δρ. Γιώργος Χρυσάνθου

Πανεπιστήμιο Κύπρου

Μάιος, 2009

ΕΥΧΑΡΙΣΤΙΕΣ

Θέλω να εκφράσω τις βαθύτερες ευχαριστίες μου στον επιβλέποντα καθηγητή της διπλωματικής αυτής εργασίας, Δρ. Βάσο Βασιλείου, για την υποστήριξη του και την καθοδήγηση του καθ' όλη την διάρκεια της εκπόνησης της παρούσας εργασίας. Η συνεχής παρακολούθηση και βοήθεια που μου πρόσφερε, το αμείωτο ενδιαφέρον του, οι συμβουλές, οι οδηγίες και οι κατευθύνσεις του συνέβαλαν καταλυτικά στην επιτυχή εκπόνηση της διπλωματικής μου εργασίας.

Και τέλος, να ευχαριστήσω θερμά την οικογένεια μου και τους φίλους μου για την ψυχολογική στήριξη και ενθάρρυνση που μου παρείχαν καθ' όλη τη διάρκεια της εκπόνησης της διπλωματικής μου εργασίας.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Κεφάλαιο 1

Εισαγωγή

Τα τελευταία χρόνια προωθείται πολύ η χρήση μικροσκοπικών συσκευών σε όλες τις πτυχές της τεχνολογικής ανάπτυξης. Απλές και χαμηλής ισχύος συσκευές που ενσωματώνουν ασύρματες τηλεπικοινωνιακές ζεύξεις υψηλής απόδοσης και βελτιωμένες μικρής κλίμακας πηγές ενέργειας έχουν συνδυαστεί με μειωμένα κόστη παραγωγής για να κάνουν ένα νέο τεχνολογικό όνειρο πραγματικότητα: τα Ασύρματα Δίκτυα Αισθητήρων (WSNs).

Τα Ασύρματα Δίκτυα Αισθητήρων (WSNs) αποτελούνται από ένα ή περισσότερα sink (ή base station) και από μερικές δεκάδες ή χιλιάδες κόμβους αισθητήρες (sensor nodes), οι οποίοι διασκορπίζονται σε ένα χώρο. Τα Ασύρματα Δίκτυα Αισθητήρων είτε λειτουργούν αυτοτελώς, είτε είναι διασυνδεδεμένα σε μεγαλύτερα δίκτυα τηλεπικοινωνιών ή στο διαδίκτυο. Τα δίκτυα αισθητήρων αποτελούνται από μεγάλο αριθμό μικρών ηλεκτρονικών διατάξεων (αισθητήρων), κινητών ή μη, που αποστέλλουν σε μια κεντρική μονάδα πληθώρα δεδομένων προς επεξεργασία και λήψη αποφάσεων. Οι κόμβοι αυτοί συλλέγουν πληροφορίες από το περιβάλλον και ανάλογα με την εφαρμογή, είτε επεξεργάζονται τις πληροφορίες και τις στέλνουν, είτε τις στέλνουν χωρίς καμιά επεξεργασία. Οι κόμβοι αυτοί, συνήθως αισθάνονται τη θερμοκρασία, το φως, τη δόνηση, τον ήχο, την ακτινοβολία κ.α. Οι πληροφορίες αυτές "ταξιδεύουν" μέσα στο δίκτυο, έχοντας σαν

τελικό προορισμό τους κόμβους sink. Ανάλογα με την εφαρμογή, τα sink ενδέχεται να αποστείλουν κάποια επερωτήματα (queries) προς τους κόμβους, με σκοπό να μαζέψουν χρήσιμες πληροφορίες.

Τα ασύρματα δίκτυα αισθητήρων θα μπορούσαν να προωθήσουν πολλές επιστημονικές αναζητήσεις παρέχοντας ένα όχημα για τις διάφορες μορφές παραγωγικότητας, συμπεριλαμβανομένης της κατασκευής, της γεωργίας, της κατασκευής, και της μεταφοράς. Τα δίκτυα αισθητήρων βρίσκουν εφαρμογές σε ποικίλους τομείς όπως ο περιβαλλοντικός έλεγχος, οι στρατιωτικοί σκοποί και η συγκέντρωση των πληροφοριών αντίληψης στις αφιλόξενες θέσεις. Οι κόμβοι αισθητήρων έχουν περιορισμένη ταχύτητα επεξεργασίας, ικανότητα αποθήκευσης, έυρος ζώνης επικοινωνίας και ενέργειας.

Μερικές κύριες προκλήσεις που υπερνικούνται με τη διευκόλυνση τέτοιων τεχνολογιών (WSNs) είναι οι παρακάτω:

- Low Power consumption
- Limited Resources
- Highly distributed
- Deployment and Coverage
- Need for Calibration
- Basic System Services
- Network Dynamics

- Spatio-Temporal Irregularity
- Data-centric Naming
- Concurrency intensive

Το πρόβλημα της κινητικότητας φαίνεται να είναι επίσης κρίσιμο για το ασύρματο περιβάλλον στο οποίο τοποθετούνται οι αισθητήρες, μιας και επηρεάζεται από περισσότερους παράγοντες από ότι στα ενσύρματα δίκτυα όπου δεν υπάρχει καθόλου κίνηση στους κόμβους μας. Σε αυτό το περιβάλλον οι παρεμβολές αυξάνονται, αυξάνοντας παράλληλα και τις πιθανότητες προβλημάτων λόγω της κίνησης των κόμβων. Η κατάσταση είναι χειρότερη στα δίκτυα αισθητήρων τα οποία έχουν περιορισμένες υπολογιστικές δυνατότητες με αποτέλεσμα οι συσκευές μας να είναι περισσότερο ευαίσθητες στα χαρακτηριστικά του περιβάλλοντος.

1.1 Επισκόπηση Ιδέας

Η μελέτη μου αποσκοπεί στην έρευνα στον τομέα της κινητικότητας για τον έλεγχο κάλυψης στα ασύρματα δίκτυα αισθητήρων. Πιο συγκεκριμένα έχει γίνει εκτεταμένη μελέτη στη κινητικότητα στα ασύρματα δίκτυα αισθητήρων με απώτερο σκοπό την εξασφάλιση της καλύτερης δυνατής κάλυψης στα δίκτυα αυτά. Μέσα από διάφορα άρθρα προσπαθήσαμε να

απομονώσουμε τα θέματα τα οποία έχουν ερευνηθεί και μπορούν σε συνδυασμό μεταξύ τους ή και το καθένα ξεχωριστά να βοηθήσουν για την έρευνα πάνω στο θέμα αυτό που ασχοληθήκαμε. Υπάρχουν πολλές ερευνητικές δουλειές που έχουν γίνει και είχαν ως αποτέλεσμα τη δημιουργία τεχνικών, αλγορίθμων, προσομοιώσεων και αναλύσεων που σχετίζονται με το πιο πάνω ερευνητικό θέμα. Αφού μελετήθηκαν όλες οι προηγούμενες έρευνες που έχουν γίνει στον τομέα των ασύρματων δικτύων αισθητήρων και πιο συγκεκριμένα με το θέμα της παρουσίασης της κινητικότητας στον έλεγχο της κάλυψης στα WSNs, συγκεντρώθηκαν και ταξινομήθηκαν έτσι ώστε να γίνει μια σωστή τοποθέτηση επί του θέματος και να παραχθούν συμπεράσματα τα οποία θα μας βοηθήσουν στην καλύτερη δυνατή επιλογή των καταλληλότερων αλγορίθμων ανάλογα με την κατάσταση που έχουμε να αντιμετωπίσουμε σε ένα ασύρματο δίκτυο αισθητήρων.

Κεφάλαιο 2

Σχετική Ερευνητική Εργασία

2.1 Επισκόπηση ασύρματων δικτύων αισθητήρων

Η πρόσφατη πρόοδος στις ασύρματες επικοινωνίες και την ηλεκτρονική έχει επιτρέψει την ανάπτυξη των χαμηλού κόστους δικτύων αισθητήρων. Τα δίκτυα αισθητήρων μπορούν να χρησιμοποιηθούν σε διάφορες εφαρμογές. Για κάθε είδος εφαρμογής, υπάρχουν διαφορετικά τεχνικά ζητήματα τα οποία χρειάζονται έρευνα. Τα Ασύρματα Δίκτυα Αισθητήρων ακολουθούν παρόμοια διαστρωμάτωση (layering) όπως και τα σταθερά δίκτυα. Σε ένα δίκτυο αισθητήρων multi-hop, οι κόμβοι επικοινωνίας συνδέονται από ένα ασύρματο μέσο. Αυτές οι συνδέσεις μπορούν να διαμορφωθούν μέσα από ραδιοσυχνότητες, υπέρυθρες ή οπτικές ζεύξεις. Το φυσικό στρώμα είναι αρμόδιο για την επιλογή συχνότητας, την παραγωγή συχνότητας μεταφορέων, την ανίχνευση σημάτων, τη διαμόρφωση και την κρυπτογράφηση στοιχείων. Το πρωτόκολλο MAC σε ένα ασύρματο δίκτυο αισθητήρων πρέπει να επιτύχει δύο στόχους. Ο πρώτος είναι η δημιουργία της υποδομής δικτύων. Ο δεύτερος στόχος είναι ο δίκαιος και αποδοτικός διαμοιρασμός των πόρων επικοινωνίας μεταξύ των κόμβων αισθητήρων. Μια

σημαντική λειτουργία του στρώματος δικτύων είναι να παρασχεθεί η σύνδεση μέσω δικτύων με τα εξωτερικά δίκτυα όπως άλλα δίκτυα αισθητήρων, τα συστήματα εντολής και ελέγχου και το διαδίκτυο. Τα πρωτόκολλα στρώματος μεταφορών είναι ακόμα ανεξερεύνητα. Μπορούν να είναι καθαρά πρωτόκολλα UDP-type, επειδή κάθε κόμβος αισθητήρων έχει περιορίσει τη μνήμη και τη δύναμη (power). Η ευελιξία, η ανοχή σφαλμάτων, το χαμηλότερο κόστος και τα γρήγορα χαρακτηριστικά επέκτασης των δικτύων αισθητήρων δημιουργούν πολλούς νέους και συναρπαστικούς τομείς εφαρμογών για δίκτυα αισθητήρων. Στο μέλλον, αυτό το ευρύ φάσμα των τομέων εφαρμογής θα καταστήσει τα δίκτυα αισθητήρων αναπόσπαστο τμήμα των ζωών μας. [1]

Διάφορα "automotive" πρωτόκολλα έχουν αναπτυχθεί, και μερικά από αυτά είναι χρήσιμα επίσης για τον έλεγχο. CAN είναι ένα τμηματικό πρωτόκολλο επικοινωνιών που αναπτύσσεται για τα "automotive" πολλαπλά συστήματα καλωδίωσης, και έχει υιοθετηθεί στις βιομηχανικές εφαρμογές από τους κατασκευαστές όπως Allen-Bradley (στο σύστημα DeviceNET) και Honeywell (στο SDS). Το CAN υποστηρίζει κατανεμημένο πραγματικού χρόνου έλεγχο με ένα υψηλό επίπεδο ασφάλειας, και είναι ένα πρωτόκολλο "multimaster" που επιτρέπει σε οποιοδήποτε κόμβο στο δίκτυο για να επικοινωνήσει με οποιοδήποτε άλλο κόμβο [11]. Υποστηρίζονται ο καθορισμένος από το χρήστη καθορισμός προτεραιοτήτων μηνυμάτων, το "multiple access/collision resolution", και η ανίχνευση λάθους. Το πρωτόκολλο LonWorks, που αναπτύχθηκε από το Echelon Corp είναι πολύ κατάλληλο για τις βιομηχανικές και καταναλωτικές εφαρμογές. Υποστηρίζει και τα επτά στρώματα του προτύπου OSI/RM, και υποστηρίζει τις απαιτήσεις "fieldbus", τη διαιτησία, και την κωδικοποίηση μηνυμάτων. Το

LonWorks λειτουργεί σε peer-to-peer “bus network basis”. Οι συσκευές σε ένα δίκτυο LonWorks επικοινωνούν με τη χρησιμοποίηση LonTalk. Αυτή η γλώσσα παρέχει ένα σύνολο υπηρεσιών που επιτρέπουν το πρόγραμμα εφαρμογής σε μια συσκευή για να στείλουν και να λάβουν τα μηνύματα από άλλες συσκευές πέρα από το δίκτυο χωρίς να πρέπει να είναι γνωστή η τοπολογία του δικτύου ή των ονομάτων, των διευθύνσεων, ή των λειτουργιών άλλων συσκευών. Το πρωτόκολλο LonWorks μπορεί προαιρετικά να παρέχει end-to-end αναγνώριση των μηνυμάτων, την επικύρωση των μηνυμάτων, και την προτεραιότητα της παράδοσης για να παρέχει τους οριακούς χρόνους συναλλαγής. Η υποστήριξη για τις διαχειριστικές υπηρεσίες δικτύων επιτρέπει τα μακρινά διαχειριστικά εργαλεία δικτύων να αλληλεπιδράσουν με τις συσκευές πέρα από το δίκτυο, συμπεριλαμβανομένου του επανασχηματισμού των διευθύνσεων δικτύων και των παραμέτρων, μεταφόρτωση των προγραμμάτων εφαρμογής, υποβολή έκθεσης των προβλημάτων δικτύων, και start/stop/reset των προγραμμάτων εφαρμογής συσκευών. Τα δίκτυα LonWorks μπορούν να εφαρμοστούν πέρα από βασικά οποιοδήποτε μέσο, συμπεριλαμβανομένων των ηλεκτροφόρων καλωδίων, το twisted pair, ραδιοσυχνότητα (FR), υπέρυθρο (IR), ομοαξονικό καλώδιο, και οπτικές ίνες.

Έχει υπάρξει μια δραματική μετατόπιση στα δίκτυα αισθητήρων προς τη μελέτη των κινητών συστημάτων. Εξετάστηκε [13] η κλασική εφαρμογή στόχων ενός τέτοιου δικτύου (monitoring). Το Ragobot είναι μικρότερο από τα περισσότερα πλήρως-πλεύσιμα ρομπότ και δομείται βαριά με συλλογή βίντεο, σύλληψη ήχου, επεξεργασία, και αναπαραγωγή ήχου, αποφυγή σύγκρουσης IR, (cliff) ανίχνευση IR, RFID ανάγνωσης/γραφής, “inertial navigation” και άλλα. Για να διατηρήσει μια ισορροπία μεταξύ της ικανότητας, της ενεργειακής αποδοτικότητας, και του

διαμορφώσιμου (modularity), το Ragobot υιοθετεί ένα παράδειγμα σχεδίου που ισορροπεί το διαμορφώσιμο και την αποδοτικότητα. Αυτή η προσέγγιση αποτελείται από δύο κύριες έννοιες: διαφοροποιημένη επικοινωνία και επαναλαμβανόμενες ενότητες (modules). Λόγω αυτής της τοποθετημένης στη σειρά διαμόρφωσης ενότητας, η επικοινωνία μπορεί να βελτιστοποιηθεί και τοπικά και συνολικά μέσα στην ιεραρχία. Οι συμπληρωματικές πληροφορίες για το Ragobot μπορούν να βρεθούν στο http://www.ragobot.com

Η συνδυασμένη και χωρισμένη ερευνητική πρόοδος στους τομείς της επικοινωνίας, και του υπολογισμού έχει οδηγήσει τις ερευνητικές προσπάθειες στα ασύρματα δίκτυα αισθητήρων (WSN) τα τελευταία χρόνια. Σύμφωνα με τον παραδοσιακό ορισμό WSNs, μια πυκνή, στατική, και απαγορευτικά ακριβή επέκταση αισθητήρων απαιτείται "implicitly", η οποία έχει περιορίσει τη δυνατότητα πραγματοποίησης ασύρματων δικτύων αισθήτηρων στις μη στρατιωτικές προσεγγίσεις. Εκτός από τη στάσιμη επέκταση αισθητήρων και sink, η εισαγωγή της κινητικότητας των κινητών τερματικών σε WSNs μπορεί να είναι για να ανακουφίσει το φορτίο ορισμένων στατικών sinks για την αποδοτική αποθήκη, τη διάδοση και τη διανομή πληροφοριών. Μελετήθηκε μια αρχιτεκτονική [15] του πολυ-ραδιόφωνο κινητού ασύρματου δικτύου αισθητήρων (MEMOSEN) που έχει εισαχθεί, το οποίο έχει λάβει την ετερογένεια και την κινητικότητα των αισθητήρων. Το κέρδος ικανότητας λόγω της κινητικότητας σε MEMOSEN έχει εξεταστεί, και οι προκλήσεις σχεδίου και οι πιθανές εφαρμογές MEMOSEN έχουν αναλυθεί επίσης. "Multiradio" και εμπνευσμένη ετερογενείς αρχιτεκτονική σχεδίαση για το κινητό ασύρματο δίκτυο αισθητήρων έχουν μελετηθεί. Η θεωρητική ανάλυση έχει επισημάνει τα κύρια χαρακτηριστικά γνωρίσματα της εφαρμογής της πολυ-τοποθετημένης στη σειρά δομής

γιατί η δικτύωση είναι η εξελιξιμότητα και η ανθεκτικότητα για την επέκταση μεγάλης κλίμακας. Επιπλέον, το κινητό στρώμα πρακτόρων που παρεμβάλλεται μεταξύ του στρώματος sink και του στρώματος αισθητήρων θα ωφελήσει για να αυξήσει την χωρητικότητα δικτύων. Παραδοσιακά, η ενεργειακή συντήρηση έχει κριθεί ως κύριος κανόνας σχεδιασμού για WSN. Εντούτοις, σε MEMOSEN όπου τα κινητά τηλέφωνα μπορούν να επαναφορτιστούν κατά περιόδους, η σχετική αξιολόγηση επίδοσης παράδοσης μηνυμάτων μπορεί να τονιστεί περισσότερο στο μέλλον.

Διάφορες ερευνητικές εργασίες προτείνουν λύσεις σε ένα ή περισσότερα από τα προβλήματα που υπάρχουν σε δίκτυα αισθητήρων. Κάποια από αυτά τα προβλήματα είναι:

Ενεργειακή αποδοτικότητα (Energy Efficiency): Η ενεργειακή αποδοτικότητα είναι μια κυρίαρχη ανάγκη, επειδή οι κόμβοι αισθητήρων έχουν μόνο μια μικρή και πεπερασμένη πηγή ενέργειας. Πολλές λύσεις, και υλικό και λογισμικό σχετικό, έχουν προταθεί για να βελτιστοποιήσουν την ενεργειακή χρήση.

Τα πρωτόκολλα έχουν ενεργειακό κόστος κατά τη μετάδοση στοιχείων για να διατηρήσουν τις δομές δρομολόγησης και να ενισχύσουν την αξιοπιστία των πληροφοριών. Τα δίκτυα αισθητήρων μπορούν να αποφύγουν τα ρητά μηνύματα πρωτοκόλλου με “piggybacking” πληροφορίες ελέγχου για τα μηνύματα δεδομένων και με “overhearing” τα πακέτα που προορίζονται για άλλους κόμβους. Μπορούν να χρησιμοποιήσουν προπρογραμματισμένο χρόνο για να μειωθεί το “contention” και ο χρόνος που το radio παραμένει ζωντανό. Αυτό μπορεί να

συντονιστεί με την υψηλού επιπέδου συμπεριφορά εφαρμογής, παραδείγματος χάριν, την περιοδική “low-rate” δειγματοληψία δεδομένων. Εναλλακτικά, το δίκτυο θα μπορούσε να εφαρμόσει την ενεργειακή συντήρηση γενικά μέσα σε χαμηλότερα στρώματα, για παράδειγμα, TDMA (time division multiple access). Το δίκτυο μπορεί να απορρίψει τα χωρίς ενδιαφέρον πακέτα με το να κλείσει το radio μετά από την παραλαβή μόνο ενός μέρους. Εντούτοις, επειδή αυτές οι πολλές βελτιστοποιήσεις μπορούν να είναι αμοιβαία συγκρουόμενες, ένας πλούσιος και αυξανόμενος όγκος της βιβλιογραφίας υιοθετεί διαφορετικούς συνδυασμούς τεχνικών στις διαφορετικές περιπτώσεις εφαρμογής και πλατφορμών.

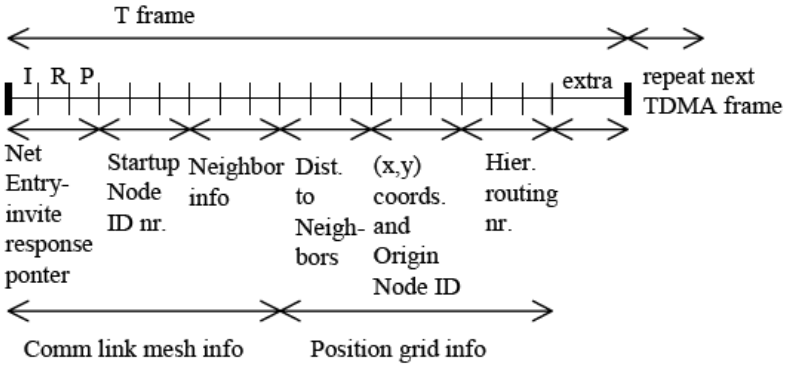
Εντοπισμός (Localization): Στις περισσότερες από τις περιπτώσεις, οι κόμβοι αισθητήρων επεκτείνονται κατά τρόπο ειδικό. Η διαδικασία μέχρι οι κόμβοι να προσδιοριστούν σε κάποιο χωρικό ισότιμο σύστημα, αυτό το πρόβλημα αναφέρεται ως εντοπισμός.

Στα δίκτυα αισθητήρων, οι κόμβοι επεκτείνονται σε μια μη σχεδιασμένη υποδομή όπου δεν υπάρχει καμία “a priori” γνώση θέσης. Το πρόβλημα για τις χωρικές-συντεταγμένες του κόμβου αναφέρεται ως εντοπισμός. Μια άμεση λύση που έρχεται να απασχολήσει, είναι το GPS ή το Global Positioning System. Εντούτοις, υπάρχουν μερικοί ισχυροί παράγοντες ενάντια στη χρήση του GPS. Το GPS μπορεί να λειτουργήσει μόνο υπαίθρια. Αφετέρου, οι δέκτες GPS είναι ακριβοί και μη κατάλληλοι στην κατασκευή των μικρών φτηνών κόμβων αισθητήρων. Ένας τρίτος παράγοντας είναι ότι δεν μπορεί να λειτουργήσει παρουσία οποιασδήποτε παρεμπόδισης όπως το dense foliage κ.λ.π. Κατά συνέπεια, οι κόμβοι αισθητήρων θα πρέπει να έχουν άλλα μέσα τις θέσεις τους και σε ένα ισότιμο σύστημα χωρίς στήριξη σε μια υπάρχουσα υποδομή. Τα

δίκτυα αισθητήρων είναι πολύ υποσχόμενα στις εφαρμογές όπου απαιτείται συγκέντρωση απομακρυσμένων πληροφοριών.

Relative Layout Positioning- Localization

Ο σχετικός προσδιορισμός θέσης ή ο εντοπισμός απαιτεί τις επικοινωνίες ενδιάμεσων κόμβων, και ένα πλαίσιο επιγραφών μηνυμάτων TDMA που έχει και τους τομείς επικοινωνιών και εντοπισμού, παρουσιάζεται στο σχήμα 1.



TDMA frame for communication protocols and localization

Σχήμα 1 : TDMA πλαίσιο [11]

Υπάρχουν διάφορα μέσα για έναν να μετρηθεί η απόσταση ενός κόμβου απο τους γείτονές του, συνήθως βασισμένη στις πληροφορίες “time-of-flight” RF. Στον αέρα, η ταχύτητα διάδοσης είναι γνωστή, έτσι οι χρονικές διαφορές μπορούν να μετατραπούν στις αποστάσεις.

Λαμβάνοντας υπόψη τις σχετικές αποστάσεις μεταξύ των κόμβων οργάνωσαν τον ιστό σε “grid specified”, σχετικές θέσεις.

Δρομολόγηση (Routing): Οι δαπάνες επικοινωνίας διαδραματίζουν έναν μεγάλο ρόλο στην απόφαση της τεχνικής δρομολόγησης που χρησιμοποιείται. Τα παραδοσιακά πρωτόκολλα δρομολόγησης δεν είναι πλέον χρήσιμα δεδομένου ότι οι ενεργειακές εκτιμήσεις απαιτούν μόνο να γίνει η ουσιαστική ελάχιστη δρομολόγηση.

Οι μικρές συσκευές για να καλύψουν τις μεγάλες αποστάσεις, το δίκτυο πρέπει να δρομολογήσει τη πληροφορία από hop σε hop μέσω των κόμβων, όπως οι δρομολογητές κινούν τις πληροφορίες σε ολόκληρο το διαδίκτυο. Ακόμα κι έτσι, η επικοινωνία παραμένει μια από τις πιο ενεργοβόρες διαδικασίες, με κάθε bit να κοστίζει τόση ενέργεια όπως περίπου 1.000 εντολές. Για να μειώσουν την κατανάλωση ενέργειας, τα ασύρματα δίκτυα αισθητήρων επεξεργάζονται τα δεδομένα μέσα στο δίκτυο οπουδήποτε είναι δυνατόν. Οι πιθανές διασυνδέσεις μεταξύ των συσκευών πρέπει να ανακαλυφθούν και οι πληροφορίες να δρομολογηθούν αποτελεσματικά από όπου παράγονται σε όπου χρησιμοποιούνται.

Το στρώμα συνδέσεων (link layer) μεταδίδει μια δομημένη σειρά από bits που διαμορφώνουν ένα πακέτο που κωδικοποιείται στο ραδιο σήμα (radio signal). Σε ένα συνδεδεμένο με καλώδιο δίκτυο όπως το Διαδίκτυο, κάθε δρομολογητής συνδέει με ένα συγκεκριμένο σύνολο άλλων δρομολογητών, που διαμορφώνει μια γραφική παράσταση δρομολόγησης. Σε WSNs, κάθε κόμβος έχει ένα radio που παρέχει ένα σύνολο συνδέσεων επικοινωνίας με τους κοντινούς

κόμβους. Με την ανταλλαγή των πληροφοριών, οι κόμβοι μπορούν να ανακαλύψουν τους γείτονές τους και να εκτελέσουν έναν κατανεμημένο αλγόριθμο για να καθορίσουν πώς να δρομολογήσουν τα στοιχεία σύμφωνα με τις ανάγκες της εφαρμογής. Αν και η φυσική τοποθέτηση καθορίζει πρώτιστα τη συνδετικότητα, οι μεταβλητές όπως οι παρεμποδίσεις, η παρεμβολές, οι περιβαλλοντικοί παράγοντες, ο προσανατολισμός κεραιών, και η κινητικότητα καθιστούν την καθοριστική συνδετικότητα πρωτίστως δύσκολη. Παρόλο αυτά, το δίκτυο ανακαλύπτει και προσαρμόζεται σε οτιδήποτε συνδετικότητα είναι παρούσα.

Όλο και περισσότερο, τα δίκτυα αισθητήρων θα επεκτείνουν τις "*disruption-tolerant*" προσεγγίσεις δικτύωσης στις οποίες μεταφέρουν τις δέσμες των στοιχείων αξιόπιστα, από hop σε hop, σε αντίθεση στο Διαδίκτυο, το οποίο ιδρύει μια end-to-end σύνδεση χρησιμοποιώντας το ταίριασμα ψηφιολέξεων (byte) ή πακέτων μεταξύ της αρχικής πηγής και του προορισμού για να καθορίσει την αξιοπιστία. Το DTN μοντέλο χρησιμοποιεί καλύτερα την μεταβλητή συνδετικότητα που προκύπτει από τα δυναμικά περιβάλλοντα και την ανάγκη duty-cycle. [3]

Στην εργασία [5], έχουν εξετάσει την ασυμπτωτική ρυθμοαπόδοση χωρητικότητας των μεγάλων κινητών ασύρματων ειδικών δικτύων. Τα αποτελέσματά μας δείχνουν ότι η άμεση επικοινωνία μεταξύ των πηγών και των προορισμών δεν μπορεί μόνο να επιτύχει την υψηλή ρυθμοαπόδοση, επειδή είναι πάρα πολύ μακριά τις περισσότερες φορές. Προτείνουν να διαδώσουν την κυκλοφορία στους ενδιάμεσους κόμβους αναμεταδότες για να εκμεταλλευτούν των πολλών χρηστών ποικιλομορφίας οφέλη έχοντας πρόσθετες «διαδρομές» μεταξύ μιας πηγής και ενός

προορισμού. Οι διαδρομές δύο-hops είναι επαρκείς για να επιτύχουν τη μέγιστη ρυθμοαπόδοση χωρητικότητας του δικτύου μέσα στα όρια που επιβάλλονται από το παρεμβαλλόμενο μοντέλο.

Μέσα από το άρθρο [4] παρουσιάζεται μια έρευνα που αφορά τα σχέδια κινητικότητας και τα πρότυπα κινητικότητας για τα ασύρματα δίκτυα. Τα σχέδια κινητικότητας είναι ταξινομημένα στους ακόλουθους τύπους: πεζοί, οχήματα, εναέριο, δυναμικό μέσο, ρομπότ, και κίνηση εξωτερικού διαστήματος.

Η βελτίωση στη ρυθμοαπόδοση είναι δραματική, αλλά υπογραμμίζουν ότι αυτό το αποτέλεσμα επιτυγχάνεται σε διάφορες ιδεαλιστικές υποθέσεις. Συγκεκριμένα υποθέτουν την πλήρη μίξη των τροχιών των κόμβων στο δίκτυο. Θα ήταν ενδιαφέρον να μελετηθεί πόση ρυθμοαπόδοση μπορεί να επιτευχθεί όταν έχουν οι κόμβοι τα λιγότερα τυχαία σχέδια κινητικότητας. Τα πρόσφατα αποτελέσματα προτείνουν ότι η υψηλή ρυθμοαπόδοση ανά ζευγάρι S-D είναι ακόμα επιτεύξιμη και όταν η κινητικότητα των κόμβων είναι περισσότερο περιορισμένη. Συγκεκριμένα, αποδείχθηκε ότι εάν κάθε κόμβος είναι περιορισμένος στην κίνηση κατά μήκος ενός τυχαία τοποθετημένου τμήματος γραμμών, η ρυθμοαπόδοση χωρητικότητα ανά-κόμβων είναι ακόμα Θ(1). Κατά συνέπεια, το δυσδιάστατο σχέδιο κινητικότητας που υπέθεσαν στο παρόν έγγραφο δεν είναι μια απαραίτητη συνθήκη για το αποτέλεσμα που κρατούμε.

Στο άρθρο [5] εστιάζεται η μετρική επίδοσης της ρυθμοαπόδοσης (throughput) χωρίς να λάβει υπόψη την καθυστέρηση. Η καθυστέρηση που βιώνεται από τα πακέτα κάτω από τη στρατηγική που προτείνεται σε αυτό το έγγραφο είναι μεγάλη, αυξάνεται με το μέγεθος του συστήματος.

Υπό αυτήν τη μορφή, το αποτέλεσμα πρέπει να αντιμετωπισθεί ως ένα θεωρητικό. Η θεωρία προτείνει ότι για τις ανεκτικές εφαρμογές καθυστέρησης, υπάρχει άφθονη ευκαιρία να ανταλλαχτεί η καθυστέρηση και η ρυθμοαπόδοση με την εκμετάλλευση της κινητικότητας. Το αποτέλεσμα αυτού του εγγράφου μπορεί να θεωρηθεί ως ακραίο σημείο στην ανταλλαγή, χωρίς οποιοδήποτε περιορισμό στην καθυστέρηση. Με έναν σφιχτότερο περιορισμό καθυστέρησης, η μέγιστη επιτεύξιμη ρυθμοαπόδοση πρέπει να μειωθεί. Θα ήταν ενδιαφέρον να χαρακτηριστεί η βέλτιστη ανταλλαγή μεταξύ της ρυθμοαπόδοσης και της καθυστέρησης και να καθοριστεί το είδος στρατηγικών που επιτυγχάνει αυτήν την ανταλλαγή.

2.2 Η κινητικότητα στα ασύρματα δίκτυα αισθητήρων

Προηγούμενη εργασία για την κάλυψη των κινητών δικτύων αισθητήρων εστιάζει στους αλγορίθμους για να επανατοποθετήσει τους αισθητήρες προκειμένου να επιτευχθεί μια στατική διαμόρφωση με μια διευρυμένη καλυμμένη περιοχή. Στο άρθρο [6] μελετήθηκαν οι δυναμικές πτυχές της κάλυψης ενός κινητού δικτύου αισθητήρων που εξαρτώνται από τη διαδικασία της μετακίνησης αισθητήρων. Με το πέρασμα του χρόνου, μια θέση είναι πιθανότερο να καλυφθεί, οι στόχοι ότι η δύναμη δεν ανιχνεύεται ποτέ σε ένα στάσιμο δίκτυο αισθητήρων μπορούν τώρα να ανιχνευθούν με την κίνηση των αισθητήρων. Χαρακτηρίζουμε την κάλυψη περιοχής στις στιγμές συγκεκριμένου χρόνου και κατά τη διάρκεια των χρονικών διαστημάτων, καθώς επίσης

και το χρόνο που παίρνει για να ανιχνεύσει έναν τυχαία τοποθετημένο στάσιμο στόχο. Τα αποτελέσματά δείχνουν ότι η κινητικότητα αισθητήρων μπορεί να χρησιμοποιηθεί για να αντισταθμίσει την έλλειψη αισθητήρων και να βελτιώσει την κάλυψη δικτύων. Για τους κινητούς στόχους, παίρνουμε ένα παράδειγμα και αντλούμε τις βέλτιστες στρατηγικές κινητικότητας για τους αισθητήρες και τους στόχους από τις προοπτικές τους. Πιο συγκεκριμένα, χαρακτηρίσθηκε η κάλυψη περιοχής στις στιγμές συγκεκριμένου χρόνου και κατά τη διάρκεια ενός χρονικού διαστήματος, και το χρόνο ανίχνευσης ενός τυχαία τοποθετημένου στόχου. Αυτά τα μέτρα κάλυψης εξαρτώνται από τη διαδικασία της μετακίνησης αισθητήρων και είναι μοναδικές ιδιότητες των κινητών δικτύων αισθητήρων. Για την τυχαία αρχική στρατηγική επέκτασης και το πρότυπο κινητικότητας αισθητήρων υπό εξέταση, έχουν δείξει ότι ενώ οι στιγμές κάλυψης περιοχής οποιαδήποτε στιγμή παραμένουν αμετάβλητες, περισσότερη περιοχή θα καλυφθεί κατά τη διάρκεια ενός χρονικού διαστήματος, επίσης, στόχοι που θα ανιχνευθούν σε ένα στάσιμο δίκτυο αισθητήρων μπορούν τώρα να ανιχνευθούν με την κίνηση των αισθητήρων. Το «tradeoff» είναι ότι μια θέση είναι μόνο καλυμμένη μέρος του χρόνου. Αυτό το σενάριο είναι σημαντικό για τις εφαρμογές που δεν απαιτούν την ταυτόχρονη κάλυψη στις στιγμές συγκεκριμένου χρόνου αλλά μάλλον επιθυμεί να καλύψει μια μεγάλη μερίδα της περιοχής κατά τη διάρκεια ενός ορισμένου χρονικού διαστήματος. Χρησιμοποιείται ο χρόνος ανίχνευσης για να μετρηθεί πόσο γρήγορα οι αισθητήρες ανιχνεύουν έναν τυχαία τοποθετημένο στόχο. Τα αποτελέσματα προτείνουν ότι η κινητικότητα αισθητήρων μπορεί να χρησιμοποιηθεί για να μειώσει αποτελεσματικά το χρόνο ανίχνευσης ενός στάσιμου στόχου όταν περιορίζεται ο αριθμός αισθητήρων. Για τους κινητούς στόχους, ο χρόνος ανίχνευσης στόχων εξαρτάται από τις στρατηγικές κινητικότητας αισθητήρων και στόχων. Πήραν ένα

παράδειγμα και μελέτησαν την καλύτερη στη χειρότερη περίπτωση απόδοση ενός κινητού δικτύου αισθητήρων από άποψη του χρόνου ανίχνευσης στόχου. Για μια δεδομένη συμπεριφορά κινητικότητας αισθητήρων, υποθέτουμε ότι ένας στόχος μπορεί να επιλέξει τη στρατηγική κινητικότητάς του ώστε να μεγιστοποιηθεί ο χρόνος ανίχνευσής του, ενώ οι αισθητήρες επιλέγουν μια στρατηγική κινητικότητας που ελαχιστοποιεί το μέγιστο χρόνο ανίχνευσης. Έχουμε διαβάσει ότι η βέλτιστη στρατηγική κινητικότητας αισθητήρων είναι ότι κάθε αισθητήρας επιλέγει τη μετακίνησή του ομοιόμορφα σε όλες τις κατευθύνσεις. Η αντίστοιχη στρατηγική κινητικότητας στόχων είναι να μείνει στάσιμη προκειμένου να μεγιστοποιηθεί ο χρόνος ανίχνευσής της.

Μια βασική πρόκληση στα δίκτυα αισθητήρων είναι η εξασφάλιση της ικανότητας υποστήριξης του συστήματος στο απαραίτητο επίπεδο απόδοσης, κατά τρόπο αυτόνομο. Η ικανότητα υποστήριξης είναι μια σημαντική ανησυχία λόγω των αυστηρών ικανοτήτων ενέργειας, εύρους ζώνης και αντίληψης περιορισμών των πόρων από άποψη στο σύστημα. Μελέτησαν [7] τη χρήση μιας νέας διάστασης σχεδίου για να ενισχύσουμε την ικανότητα υποστήριξης στα δίκτυα αισθητήρων, τη χρήση της ελεγχόμενης κινητικότητας. Επίσης είδαν πως αυτή η ικανότητα μπορεί να ανακουφίσει τους περιορισμούς των πόρων και να βελτιώσει την απόδοση συστημάτων με το να προσαρμοστεί στις απαιτήσεις επέκτασης. Ενώ η καιροσκοπική χρήση της εξωτερικής κινητικότητας έχει εξεταστεί προηγουμένως, η χρήση της ελεγχόμενης κινητικότητας είναι κατά ένα μεγάλο μέρος ανεξερεύνητη. Είδαν ακόμη τα ερευνητικά ζητήματα που συνδέονται με αποτελεσματική χρήση αυτής της νέας διάστασης σχεδίου. Δύο πρωτότυπα συστημάτων, είδαμε [7], που παρουσιάζουν τα πρώτα βήματα προς την πραγματοποίηση του

προτεινόμενου οράματος. Βλέπουμε ότι η εισαγωγή της ελεγχόμενης κινητικότητας έχει το σημαντικό όφελος για την απόδοση και την ικανότητα υποστήριξης των δικτύων αισθητήρων. Η προσέγγιση Morph ενεργά να μετατρέψει τα δικτυωμένα συστήματα και παρέχει μια νέα μέθοδο για τα άμεσα διευθυνσιοδοτημένα σύνθετα προβλήματα που συνδέονται με τα δίκτυα που έχουν σχέση στενά με το φυσικό περιβάλλον τους. Εντούτοις, υπάρχουν ανοικτά ζητήματα που αντιμετωπίζονται προτού να μπορέσουν να πραγματοποιηθούν τα πιθανά πλεονεκτήματα. Παραδείγματος χάριν, έχουν δείξει ότι η εφαρμογή των συμβατικών πρωτοκόλλων επικοινωνίας στα δίκτυα Morph θα μπορούσε να επιδεινωθεί, αντί να βελτιωθεί, η απόδοση. Τα θεμελιώδη ζητήματα που οδηγούν στις προκλήσεις Morph επισημάνθηκαν και διάφορες συγκεκριμένες προκλήσεις συζητήθηκαν. Αυτές οι προκλήσεις εμπνέουν την έρευνα σε όλα τα στρώματα της στοίβας δικτύου, που περιλαμβάνουν τη φυσική προσαρμογή στρώματος στο εξωτερικό περιβάλλον, τις αλλαγές στρώματος μεταφορών στην εργασία με Morph και τη συγκεκριμένη εκμετάλλευση εφαρμογής της ελεγχόμενης κινητικότητας. Είδαμε επίσης τα αρχικά πρωτότυπα που εκμεταλλεύονται την ελεγχόμενη κινητικότητα και καταδεικνύουν τα πλεονεκτήματα και τη δυνατότητα πραγματοποίησης Morph. Πιστεύουν ότι είναι έγκαιρο και ελκυστικό να αναπτυχθεί η λειτουργία δικτύων για την εκμετάλλευση της ελεγχόμενης ώθησης που εκτείνεται στους ποικίλους βαθμούς ελευθερίας στα διαφορετικά χρονοδιαγράμματα.

2.3 Το πρόβλημα των τρυπών στα ασύρματα δίκτυα αισθητήρων

Διάφορες ανωμαλίες μπορούν να εμφανιστούν στα ασύρματα δίκτυα αισθητήρων που εξασθενίζουν τις επιθυμητές λειτουργίες τους δηλ., την αντίληψη και την επικοινωνία. Τα διαφορετικά είδη τρυπών μπορούν να διαμορφωθούν σε τέτοια δίκτυα δημιουργώντας γεωγραφικά συσχετισμένες προβληματικές περιοχές όπως οι τρύπες κάλυψης, οι τρύπες δρομολόγησης, οι "jamming" τρύπες, οι sink/μαύρες τρύπες, οι "worm" τρύπες, κ.λπ. Γίνετε περιγραφή στο άρθρο [28] των διαφορετικών τύπων τρυπών, τονίζονται τα χαρακτηριστικά τους και μελετούνται τα αποτελέσματά τους στην επιτυχή εργασία ενός δικτύου αισθητήρων. Παρουσιάζεται η κατάσταση προόδου στην έρευνα για την εξέταση των σχετικών με τις τρύπες προβλημάτων στα ασύρματα δίκτυα αισθητήρων και συζητούνται προτεινόμενες λύσεις για την καταπολέμηση των διαφορετικών ειδών τρυπών. Ολοκληρώνοντας δίνουν έμφαση στις μελλοντικές ερευνητικές κατευθύνσεις.

Περιγράφουν τις διάφορες προτεινόμενες λύσεις για την εύρεση και τον καθορισμό των τρυπών κάλυψης στα δίκτυα αισθητήρων. Τα προτεινόμενα πρωτόκολλα που συζητούνται εδώ είναι ταξινομημένα με βάση τον αριθμό κινητών κόμβων στο δίκτυο αισθητήρων ως (ι) κινητά δίκτυα αισθητήρων (ii) υβριδικά δίκτυα αισθητήρων (iii) στατικά δίκτυα αισθητήρων. Για κάθε ένα από τα πρωτόκολλα που περιγράφονται μια περαιτέρω διαφοροποίηση γίνεται σύμφωνα με το εάν είναι σχεδιασμένα για να εξετάσουν την ενιαία (single) ή την πολλαπλάσια (multiple) απαίτηση κάλυψης.

Η βασική υπόθεση στα στατικά δίκτυα αισθητήρων, ότι οι κόμβοι είναι διαθέσιμοι για να καλύψουν κάθε σημείο στη περιοχή στόχου για να παρέχουν το επιθυμητό επίπεδο κάλυψης είναι υπερβολικά απλοϊκή. Αυτό ισχύει ιδιαίτερα για εχθρικά ή σκληρά περιβάλλοντα, όπως τα πεδία μάχης και την πυρκαγιά ή χημική έκχυση, όπου το δίκτυο δεν μπορεί να είναι προσεκτικά επεκταμένο σε μια προκαθορισμένη κανονική τοπολογία. Τυχαία εναέρια επέκταση των κόμβων αισθητήρων είναι μια πιθανή λύση, αλλά σε αυτήν την περίπτωση αυτό είναι πολύ δύσκολο να εγγυηθεί ότι η απαραίτητη πολλαπλάσια κάλυψη επιτυγχάνεται.

Έχουν παρέχει ένα στιγμιότυπο της τρέχουσας κατάστασης προόδου της έρευνας στα δίκτυα αισθητήρων που εξετάζουν τα διάφορα προβλήματα σχετικά με τις τρύπες. Μέσα από το άρθρο [28] παρουσιάστηκαν υπάρχουσες λύσεις περιγράφοντας τη συμπεριφορά τους με των διαφόρων ειδών τρύπες που αναφέρθηκαν στο έγγραφο αυτό (coverage, routing, jamming, sink/black, worm). Η έρευνα αυτή έχει αποκαλύψει επίσης μερικές πιθανές μελλοντικές ερευνητικές κατευθύνσεις, όπως την ανάπτυξη των πρωτοκόλλων κάλυψης βασισμένα στις ρεαλιστικές υποθέσεις, χρησιμοποίηση κινητών αισθητήρων για την επίτευξη της διαφοροποιημένης πολλαπλάσιας κάλυψης, ανακάλυψη μιας αποδοτικής λύσης για την καταδίωξη των κινούμενων jamming τρυπών και της ανάγκης για ασφαλέστερα και ελαστικά πρωτόκολλα στα ασύρματα δίκτυα αισθητήρων.

2.4 Η κάλυψη δικτύου από αισθητήρες

Η επέκταση αισθητήρων είναι ένα σημαντικό ζήτημα στο σχεδιασμό των δικτύων αισθητήρων. Στο άρθρο [8], σχεδιάζουν και αξιολογούν τα διανεμημένα πρωτόκολλα αυτο-επέκτασης για τους κινητούς αισθητήρες. Μετά την ανακάλυψη μιας τρύπας κάλυψης, τα προτεινόμενα πρωτόκολλα υπολογίζουν τις θέσεις στόχων των αισθητήρων όπου πρέπει να κινηθούν. Χρησιμοποιούν τα διαγράμματα Voronoi για να ανακαλύψουν τις τρύπες κάλυψης και σχεδίασαν τρία " movement-assisted" πρωτόκολλα επέκτασης αισθητήρων, το VEC (VECtor based), το VOR (vORonoi based), και το Minimax βασισμένο στην αρχή της μετακίνησης αισθητήρων από τις πυκνά επεκταμένες περιοχές προς τις αραιά επεκταμένες περιοχές. Τα αποτελέσματα προσομοίωσης δείχνουν ότι τα πρωτόκολλά αυτά μπορούν να παρέχουν την υψηλή κάλυψη μέσα σε έναν σύντομο χρόνο ανάπτυξης και με μια περιορισμένη μετακίνηση.

Τα ασύρματα δίκτυα αισθητήρων αναμένονται να χρησιμοποιηθούν εντατικά στο μέλλον δεδομένου ότι μπορούν πολύ να ενισχύσουν την ικανότητά μας για παρακολούθηση και έλεγχο του φυσικού περιβάλλοντος. Τα δίκτυα αισθητήρων ξεσηκώνουν τις παραδοσιακές μεθόδους συλλογής δεδομένων, γεφυρώνοντας το χάσμα μεταξύ του φυσικού κόσμου και του εικονικού κόσμου πληροφοριών. Λόγω της λαβυρινθώδης σχέσης με το φυσικό κόσμο, η κατάλληλη επέκταση των αισθητήρων είναι πολύ σημαντική για την επιτυχή ολοκλήρωση των στόχων αντίληψης που εκδίδονται. Οι περισσότερες εργασίες υποθέτουν ότι το περιβάλλον είναι αρκετά γνωστό και υπό έλεγχο. Εντούτοις, όταν το περιβάλλον είναι άγνωστο ή εχθρικό όπως οι απομακρυσμένοι σκληροί τομείς, καταστρεμμένες περιοχές και τοξικές αστικές περιοχές, η

επέκταση αισθητήρων δεν μπορεί να εκτελεσθεί “manual”. Έχουν υπάρξει μερικές ερευνητικές προσπάθειες στην ανάπτυξη των κινητών αισθητήρων, αλλά οι περισσότερες από αυτές είναι βασισμένες στις συγκεντρωμένες προσεγγίσεις.

Σχεδιάστηκαν και αξιολογήθηκαν [8] τρία διανεμημένα πρωτόκολλα αυτό-επέκτασης για τα δίκτυα αισθητήρων. Η δήλωση προβλήματός μας είναι: λαμβάνοντας υπόψη την περιοχή στόχου, πώς να μεγιστοποιήσουμε την κάλυψη αισθητήρων με το λιγότερο χρόνο, την απόσταση μετακίνησης και την πολυπλοκότητα μηνυμάτων. Λαμβάνοντας υπόψη μια περιοχή που ελέγχεται, τα διανεμημένα πρωτόκολλα αυτό-επέκτασής ανακαλύπτουν αρχικά την ύπαρξη των τρυπών κάλυψης (η περιοχή που δεν καλύπτεται από οποιοδήποτε αισθητήρα) στην περιοχή στόχου βασισμένη στην αισθανόμενη υπηρεσία που απαιτείται από την εφαρμογή. Μετά από την ανακάλυψη μιας τρύπας κάλυψης, τα προτεινόμενα πρωτόκολλα υπολογίζουν τις θέσεις στόχων αυτών των αισθητήρων, όπου πρέπει να κινηθούν. Χρησιμοποιούνται τα διαγράμματα Voronoi για την ανακάλυψη των τρυπών και σχεδιάζονται τρία πρωτόκολλα βοηθημένα στην μετακίνηση αισθητήρων για να επιτευχθεί η κάλυψη της επέκτασης, VEC (VECtor-based), VOR (VORonoi-based), και Minimax βασισμένο στην αρχή της κινητικότητας των αισθητήρων από τις πυκνά επεκταμένες περιοχές προς τις αραιά επεκταμένες περιοχές. Από τις εντατικές προσομοιώσεις, γίνεται αξιολόγηση των πρωτοκόλλων από διάφορες πτυχές: κάλυψη, χρόνος επέκτασης, κινούμενη απόσταση, εξελιξιμότητα στην αρχική επέκταση και κυμαινόμενη επικοινωνία, κλπ, και δείχνουν ότι τα πρωτόκολλά είναι πολύ αποτελεσματικά από την άποψη της κάλυψης, του χρόνου επέκτασης, και της κινούμενης απόστασης.

Το διάγραμμα Voronoi είναι μια σημαντική δομή δεδομένων στην υπολογιστική γεωμετρία. Αντιπροσωπεύει τις πληροφορίες εγγύτητας για ένα σύνολο κόμβων. Το διάγραμμα Voronoi μιας συλλογής των γεωμετρικών κόμβων χωρίζει το διάστημα στα κύτταρα-cells, κάθε ένα από τα οποία αποτελείται από τα σημεία που είναι πιο κοντά σε έναν ιδιαίτερο κόμβο από ότι σε οποιαδήποτε άλλα. Τα πρωτόκολλα επέκτασης αισθητήρων είναι βασισμένα στα διαγράμματα Voronoi. Κάθε αισθητήρας, που αντιπροσωπεύεται από έναν αριθμό, εσωκλείεται από ένα κύτταρο Voronoi. Αυτά τα πολύγωνα καλύπτουν μαζί τον τομέα στόχων. Τα σημεία μέσα σε ένα πολύγωνο είναι πιο κοντά στον αισθητήρα μέσα σε αυτό το πολύγωνο σε σύγκριση με τους αισθητήρες που τοποθετούνται αλλού. Εάν αυτός ο αισθητήρας δεν μπορεί να ανιχνεύσει το αναμενόμενο φαινόμενο, κανένας άλλος αισθητήρας δεν μπορεί να το ανιχνεύσει. Κατά συνέπεια, κάθε αισθητήρας είναι αρμόδιος για το στόχο αντίληψης στο κύτταρο Voronoi του και επιλέγουμε το διάγραμμα Voronoi ως δομή δεδομένων μας για να εξετάσουμε τις τρύπες κάλυψης. Κατ' αυτό τον τρόπο, κάθε κόμβος μπορεί να εξετάσει την τρύπα κάλυψης τοπικά, και πρέπει μόνο να ελέγξει μια μικρή περιοχή γύρω από αυτήν. Επίσης, για να κατασκευαστεί το κύτταρο Voronoi, κάθε αισθητήρας πρέπει μόνο να ξέρει την ύπαρξη των γειτόνων Voronoi του, η οποία μειώνει την πολυπλοκότητα επικοινωνίας δεδομένου ότι είναι ικανοποιητικό να επικοινωνήσει τοπικά με τους γείτονες Voronoi του.

Το πρωτόκολλο επέκτασης αισθητήρων μας τρέχει επαναληπτικά έως ότου τερματιστεί ή φθάσει στο διευκρινισμένο μέγιστο γύρο. Σε κάθε γύρο, οι αισθητήρες αρχικά μεταδίδουν ραδιοφωνικά τις θέσεις τους και κατασκευάζουν τα τοπικά πολύγωνα Voronoi τους βασισμένα στις λαμβανόμενες πληροφορίες των γειτόνων. Για να κατασκευάσει το πολύγωνο Voronoi του, κάθε

αισθητήρας υπολογίζει αρχικά τους διχοτόμους των εξεταζόμενων αισθητήρων και τις δικές του βασισμένος στις πληροφορίες θέσης. Αυτοί οι διχοτόμοι και το όριο του τομέα στόχου διαμορφώνουν διάφορα πολύγωνα. Το μικρότερο πολύγωνο που περικυκλώνει τον αισθητήρα είναι το πολύγωνο Voronoi αυτού του αισθητήρα. Αφότου έχουν κατασκευαστεί τα πολύγωνα Voronoi, εξετάζονται για να καθορίσουν την ύπαρξη τρυπών κάλυψης. Εάν οποιαδήποτε τρύπα κάλυψης υπάρχει, οι αισθητήρες αποφασίζουν πού να κινηθούν για να αποβάλουν ή να μειώσουν το μέγεθος της τρύπας κάλυψης, διαφορετικά παραμένουν στη θέση τους.

Το VEC πρωτόκολλο σπρώχνει αισθητήρες μακριά από μια πυκνά καλυμμένη περιοχή, το VOR τραβά τους αισθητήρες στην αραιά καλυμμένη περιοχή, και το Minimax κινεί τους αισθητήρες προς την τοπική κεντρική περιοχή τους.

Ο VEC (VECtor-based Algorithm) παρακινείται από τις ιδιότητες των ηλεκτρομαγνητικών μορίων: όταν δύο ηλεκτρομαγνητικά μόρια είναι το ένα πολύ κοντά στο άλλο, μια δύναμη απωθών τα ωθεί χώρια. Η τελική γενική δύναμη στους αισθητήρες είναι το διανυσματικό άθροισμα των εικονικών δυνάμεων από το όριο και όλους τους γείτονες Voronoi. Αυτές οι εικονικές δυνάμεις θα ωθήσουν τους αισθητήρες από την πυκνά καλυμμένη περιοχή στην αραιά καλυμμένη περιοχή. Κατά συνέπεια, VEC είναι ένας "δυναμικός" αλγόριθμος, ο οποίος προσπαθεί να επανεντοπίσει τους αισθητήρες που διανέμονται ομοιόμορφα. Σαν εμπλουτισμό, προσθέτουμε ένα σχέδιο μετακίνησης-ρύθμισης για να μειώσουμε το λάθος της εικονικής-δύναμης. Όταν ένας αισθητήρας καθορίζει τη θέση στόχου του, ελέγχει εάν η τοπική κάλυψη θα αυξηθεί από τη μετακίνηση. Η τοπική κάλυψη ορίζεται ως η κάλυψη του τοπικού πολυγώνου

Voronoi και μπορεί να υπολογιστεί από τη διατομή του πολυγώνου και του αισθανόμενου κύκλου. Εάν η τοπική κάλυψη δεν αυξάνεται, ο αισθητήρας δεν πρέπει να κινηθεί προς τη θέση στόχου. Αν και η γενική κατεύθυνση της μετακίνησης είναι σωστή, η τοπική κάλυψη δεν μπορεί να αυξηθεί επειδή η θέση στόχου είναι πάρα πολύ μακριά. Για να εξετάσει αυτό το πρόβλημα, ο αισθητήρας θα επιλέξει το μεσαίο σημείο μεταξύ της θέσης στόχου του και της τρέχουσας θέσης του ως νέα θέση στόχου του. Εάν η τοπική κάλυψη αυξάνεται στη νέα θέση στόχου, ο αισθητήρας θα κινηθεί διαφορετικά, θα παραμείνει στην θέση του.

Σε σύγκριση με τον αλγόριθμο VEC, ο VOR (VORonoi-based Algorithm) είναι ένας αλγόριθμος βασισμένος στο τράβηγμα, δηλαδή τραβά τους αισθητήρες στις τοπικές μέγιστες τρύπες κάλυψής τους. Στον αλγόριθμο VOR, εάν ένας αισθητήρας ανιχνεύσει την ύπαρξη των τρυπών κάλυψης, θα κινηθεί το πολύ-πολύ προς την κορυφή Voronoi. Ο VOR είναι ένας άπληστος αλγόριθμος που προσπαθεί να καθορίσει τη μεγαλύτερη τρύπα. Οι κινούμενες ταλαντώσεις μπορούν να εμφανιστούν εάν οι νέες τρύπες παράγονται λόγω της αναχώρησης του αισθητήρα. Για να εξετάσουμε αυτό το πρόβλημα, προσθέτουμε τον έλεγχο ταλάντωσης που δεν επιτρέπει στους αισθητήρες να κινηθούν αμέσως προς τα πίσω. Κάθε φορά που θέλει να κινηθεί ένας αισθητήρας, πρώτα ελέγχει εάν η κινούμενη κατεύθυνσή του είναι αντίθετη από αυτήν που είχε στον προηγούμενο γύρο. Εάν ναι, σταματά για ένα γύρο. Επιπλέον, η ρύθμιση μετακίνησης που αναφέρεται στον αλγόριθμο VEC εφαρμόζεται επίσης εδώ.

Παρόμοιος με τον αλγόριθμο VOR, ο Minimax αλγόριθμος καθορίζει τις τρύπες όταν κινηθεί πιο κοντά προς την κορυφή Voronoi, αλλά δεν κινείται τόσο μακριά όσο ο VOR για να αποφύγει

την κατάσταση στην οποία η κορυφή που ήταν αρχικά κοντά γίνεται νέα κορυφή. Ο Minimax επιλέγει τη θέση στόχου ως σημείο μέσα στο πολύγωνο Voronoi του οποίου η απόσταση από την κορυφή Voronoi ελαχιστοποιείται. Καλούμε αυτό το σημείο Minimax σημείο. Αυτός ο αλγόριθμος είναι βασισμένος στην πεποίθηση ότι ένας αισθητήρας δεν πρέπει να είναι πάρα πολύ μακριά από οποιαδήποτε κορυφή Voronoi του όταν διανέμονται ομοιόμορφα οι αισθητήρες. Ο αλγόριθμος Minimax μπορεί να μειώσει τη διαφορά των αποστάσεων από τις κορυφές Voronoi, με συνέπεια ένα κανονικότερο διαμορφωμένο πολύγωνο Voronoi, το οποίο χρησιμοποιεί καλύτερα τον αισθανόμενο κύκλο του αισθητήρα. Σε αντίθεση με τον αλγόριθμο VOR, ο Minimax εξετάζει περισσότερες πληροφορίες και είναι πιο συντηρητικός. Σε σύγκριση με τον αλγόριθμο VEC, ο Minimax είναι "αντιδραστικός", καθορίζει την τρύπα αμεσότερα με την κίνηση προς την κορυφή Voronoi.

Μέτρησαν [8] την απόδοση των προτεινόμενων πρωτοκόλλων από δύο πτυχές: ποιότητα και κόστος επέκτασης. Η ποιότητα επέκτασης μετριέται μέχρι την κάλυψη αισθητήρων και το χρόνο να επιτευχθεί αυτή η κάλυψη. Ο χρόνος επέκτασης καθορίζεται μέχρι τον αριθμό γύρων που απαιτούνται και το χρόνο κάθε γύρου. Η διάρκεια κάθε γύρου καθορίζεται πρώτιστα από την κινούμενη ταχύτητα των αισθητήρων, η οποία είναι η μηχανική ιδιότητα των αισθητήρων. Κατά συνέπεια, χρησιμοποιούμε μόνο τον αριθμό γύρων για να μετρήσουμε το χρόνο επέκτασης. Το κόστος έχει δύο συστατικά. Το πρώτο είναι το κόστος αισθητήρων για να φθάσει σε μια ορισμένη κάλυψη, το προτεινόμενο πρωτόκολλο χρειάζεται λιγότερους αισθητήρες από την τυχαία επέκταση των στατικών αισθητήρων. Ο δεύτερος είναι η κατανάλωση ενέργειας της επέκτασης. Η μηχανική μετακίνηση και η ηλεκτρονική επικοινωνία καταναλώνουν την ενέργεια,

της οποίας η μηχανική κίνηση είναι το σημαντικότερο μέρος. Ως εκ τούτου επιλέγουμε την κινούμενη απόσταση ως μετρική αξιολόγησης.

Μπορούμε να συμπεράνουμε ότι ο αλγόριθμος Minimax εκτελείται καλύτερα, ενώ ο VEC εκτελείται χειρότερα. Ο Minimax χρησιμοποιεί πλήρως το τοπικό πολύγωνο Voronoi. Καθορίζει την τρύπα κάλυψης άμεσα με την κίνηση προς τη μεγαλύτερη τρύπα, αποφεύγοντας την πιθανή αρνητική επίπτωση από τη μετακίνηση του αισθητήρα. Επιπλέον, με την εντόπιση των αισθητήρων στο minimax σημείο, ο Minimax χαμηλώνει τη διαφορά των αποστάσεων ενός αισθητήρα από τις κορυφές Voronoi του, με συνέπεια ένα κανονικότερο διαμορφωμένο πολύγωνο Voronoi. Ο VOR καθορίζει την τρύπα κάλυψης πιο άπληστα, αλλά στερείται μια περιεκτική εκτίμηση. Είναι αναμενόμενο ότι ο VOR αποδίδει χειρότερα από τον Minimax. Κάτι που συμβαίνει στον πρώτο γύρο είναι ότι ο VOR είναι καλύτερος από τον Minimax. Αυτό προκύπτει επειδή αρχικά οι τρύπες κάλυψης είναι κανονικά μεγαλύτερες από ότι στη μέση της διαδικασίας επέκτασης, κατά συνέπεια μπορεί άπληστα να οδηγήσει σε μια υψηλότερη αύξηση κάλυψης. Ο VEC εκτελείται χειρότερα για διάφορους λόγους. Ο κύριος λόγος είναι ότι ο VEC είναι ευαίσθητος στην αρχική επέκταση. Σε μια ακραία κατάσταση, όπου οι αισθητήρες βρίσκονται στην ίδια γραμμή με την ίδια “inter-distance”, κανένας αισθητήρας δεν θα κινηθεί, δεδομένου ότι οι εικονικές δυνάμεις αντισταθμίζουν η μια την άλλη, αν και υπάρχουν μεγάλες τρύπες κάλυψης. Εάν οι αισθητήρες βρίσκονται σε παρόμοια σχετική θέση αρχικά, ο VEC δεν αποδίδει καλά. Επιπλέον, ο VEC ούτε εξετάζει τις τρύπες κάλυψης ούτε χρησιμοποιεί οποιεσδήποτε γεωμετρικές πληροφορίες από το πολύγωνο Voronoi κατά την επιλογή της θέσης στόχου. Προσπαθεί να φθάσει σε μια σχετικά ισορροπημένη θέση μεταξύ των αισθητήρων, που

είναι πολύ σκληρή, προκύπτει δυσκολία ακόμη και για την διανομή μόνο των τοπικών πληροφοριών.

Για τον VEC, η κινούμενη απόσταση είναι παρόμοια κάτω από τις διαφορετικές πυκνότητες αισθητήρων. Αυτό προκύπτει επειδή ο VEC καθορίζει τις τρύπες κάλυψης με την ώθηση των αισθητήρων σε μια σχετική διανομή. Στον VEC, οι αισθητήρες ωθούνται από τις εικονικές δυνάμεις, οι οποίες καθορίζονται από τη διαφορά μεταξύ της μέσης απόστασης των αισθητήρων όταν διανέμονται ομοιόμορφα και οι μεμονωμένες "inter-distances". Και οι δύο τιμές αυξάνονται με μια χαμηλή πυκνότητα και μειώνονται με μια υψηλότερη πυκνότητα. Κατά συνέπεια, η διαφορά δεν είναι τόσο ευαίσθητη στην πυκνότητα αισθητήρων. Σε αντίθεση, ο Minimax και ο VOR επανεντοπίζουν τους αισθητήρες με τη μέτρηση των τρυπών κάλυψης, οι οποίες είναι μεγαλύτερες κάτω από τη χαμηλότερη πυκνότητα και τη μικρότερη κατώτερη υψηλή πυκνότητα. Μεταξύ του Minimax και του VOR, το πρώτο κινεί πάντα μια μεγαλύτερη απόσταση. Ο Minimax όχι μόνο προσπαθεί να καθορίσει τις τρύπες, αλλά προσπαθεί να φθάσει στα κανονικότερα διαμορφωμένα πολύγωνα Voronoi. Κατά συνέπεια, μετά από τους πρώτους δύο γύρους και τις υπόλοιπες τρύπες είναι σχετικά μικρός, ο VOR μετακινεί τους αισθητήρες ελαφρώς ενώ ο Minimax κάνει τον αισθητήρα να κινηθεί περισσότερο προς τα minimax σημεία.

Όταν η εμβέλεια επικοινωνίας είναι πάρα πολύ χαμηλή, οι περισσότεροι αισθητήρες δεν ξέρουν όλους τους γείτονες Voronoi τους, και το κατασκευασμένο διάγραμμα Voronoi δεν είναι πολύ ακριβές. Συνεπώς, οι αισθητήρες μπορούν να πάρουν μερικές ψεύτικες τρύπες κάλυψης και να λάβουν τις λανθασμένες αποφάσεις για τη θέση στόχου. Μεταξύ αυτών των τριών

πρωτοκόλλων, ο VEC επηρεάζεται σε λιγότερο από τη χαμηλή εμβέλεια επικοινωνίας και από την άποψη της κινούμενης απόστασης και της κάλυψης, δεδομένου ότι χρησιμοποιεί μόνο το πολύγωνο Voronoi για να αποφασίσει εάν θα κινηθεί ή όχι, αλλά όχι να αποφασίσει τη θέση στόχου.

Σε πολλά πιθανά εργασιακά περιβάλλοντα, όπως οι απομακρυσμένοι σκληροί τομείς, οι περιοχές καταστροφής και οι τοξικές αστικές περιοχές, η επέκταση αισθητήρων δεν μπορεί να εκτελεσθεί "manual". Μια πιθανή λύση είναι να διασκορπιστούν οι αισθητήρες με τα αεροσκάφη. Εντούτοις, χρησιμοποιώντας αυτήν την τεχνική, η πραγματική θέση προσγείωσης δεν μπορεί να ελεγχθεί λόγω της ύπαρξης του αέρα και των εμποδίων, όπως τα δέντρα και τα κτήρια. Συνεπώς, η κάλυψη μπορεί να είναι κατώτερη από τις απαιτήσεις εφαρμογής ανεξάρτητα από το πόσους αισθητήρες πέφτουν. Επιπλέον, σε πολλές περιπτώσεις, όπως κατά τη διάρκεια των τοξικών διαρροών, οι χημικοί αισθητήρες πρέπει να τοποθετηθούν μέσα σε ένα κτήριο από το εξωτερικό μέρος. Σε αυτά τα σενάρια, είναι απαραίτητο να χρησιμοποιηθούν οι κινητοί αισθητήρες, οι οποίοι μπορούν να κινηθούν προς τις σωστές θέσεις για να παρέχουν την απαραίτητη κάλυψη. Σε μια προηγούμενη εργασία [8], σχεδιάστηκαν τρία διανεμημένα πρωτόκολλα αυτό-επέκτασης που βοηθούν τους κινητούς αισθητήρες στην κίνηση από τις πυκνά καλυμμένες περιοχές προς τις αραιά καλυμμένες περιοχές, και καταδείχτηκαν ότι αυτά τα πρωτόκολλα μπορούν να παρέχουν μια καλή κάλυψη μέσα σε έναν σύντομο χρόνο κρατώντας μια χαμηλή επέκταση κόστους από την άποψη της κίνησης της απόστασης και της πολυπλοκότητας μηνυμάτων. Εντούτοις, να εξοπλιστεί κάθε αισθητήρας με μια μηχανή που αυξάνει το κόστος δικτύου και είναι περιττός όταν η απαίτηση κάλυψης δεν είναι πολύ αυστηρή,

ή εάν οι αισθητήρες μπορούν να διασκορπιστούν στον τομέα στόχων σχετικά ομοιόμορφα. Για να επιτευχθεί μια ισορροπία μεταξύ του αισθητήρα κόστους και κάλυψης, μπορούμε να επεκτείνουμε ένα μίγμα κινητών αισθητήρων και στατικών αισθητήρων για να κατασκευάσουμε τα δίκτυα αισθητήρων. Παρουσιάστηκε [9] ένα νέο διανεμημένο πρωτόκολλο προσφοράς για την επέκταση των κινητών αισθητήρων, που σχεδιάζεται ειδικά για τα δίκτυα αισθητήρων στα οποία μόνο ένα υποσύνολο των επεκταμένων αισθητήρων είναι κινητό. Σε αυτό το πρωτόκολλό, οι κινητοί αισθητήρες αντιμετωπίζονται ως κεντρικοί υπολογιστές για να θεραπεύσουν τις τρύπες κάλυψης, οι οποίες είναι θέσεις που δεν καλύπτονται από οποιοδήποτε αισθητήρα. Κάθε κινητός αισθητήρας έχει μια ορισμένη βασική αξία για να εξυπηρετήσει μια τρύπα στον αισθανόμενο τομέα. Η τιμή συσχετίζεται με το μέγεθος οποιασδήποτε νέας τρύπας που παράγεται από τη μετακίνησή τους. Οι στατικοί αισθητήρες θα ανιχνεύσουν τις τρύπες κάλυψης τοπικά, θα υπολογίσουν τα μεγέθη τους ως προσφορές, και θα προσφέρουν τους κινητούς αισθητήρες με μια βασική τιμή χαμηλότερη από τις προσφορές τους. Οι κινητοί αισθητήρες επιλέγουν τις υψηλότερες προσφορές και κινούνται για να θεραπεύσουν τις μεγαλύτερες τρύπες κάλυψης. Αυτή η διαδικασία επαναλαμβάνει έως ότου δεν μπορεί να δώσει κανένας στατικός αισθητήρας μια προσφορά υψηλότερη από την βασική τιμή οποιουδήποτε κινητού αισθητήρα.

Στο πρωτόκολλο που θα παρουσιάσουμε πιο κάτω [9], οι στατικοί αισθητήρες ανιχνεύουν ελλείψεις στην τοπική κάλυψη χρησιμοποιώντας διαγράμματα Voronoi, και προσφέρει κινητούς αισθητήρες που βασίζονται στο μέγεθος της ανιχνευμένης έλλειψης. Οι κινητοί αισθητήρες διαλέγουν ποιες ελλείψεις στην κάλυψη θα περιθάλψουν βασιζόμενοι στην προσφορά.

Όταν μια μερίδα των επεκταμένων αισθητήρων είναι κινητή, το πρόβλημα επέκτασης μπορεί να περιγραφεί ως εξής: λαμβάνοντας υπόψη έναν τομέα στόχων που καλύπτεται από διάφορους κύκλους (ο αισθανόμενος κύκλος των στατικών αισθητήρων), αλλά ακόμα υπάρχουν μερικές ακάλυπτες περιοχές, πώς να τοποθετήσει ορισμένους πρόσθετους κύκλους (ο αισθανόμενος κύκλος των κινητών αισθητήρων) για να μεγιστοποιήσει τη γενική κάλυψη.

Greedy Approximation Algorithm

Αν και το πρόβλημά μας είναι ένα πλήρως δύσκολο πρόβλημα (NP-hard), και δεν υπάρχει καμία βέλτιστη λύση, μπορούμε ακόμα να βρούμε μερικές πρακτικές λύσεις για να προσεγγίσουμε τη βέλτιστη λύση βασισμένη σε heuristics. Παρόμοιο με τον άπληστο αλγόριθμο, που είναι ένας συνήθης χρησιμοποιημένος ευρετικός αλγόριθμος για το καθορισμένο πρόβλημα κάλυψης, το πρόβλημα επέκτασής μας μπορεί να λυθεί ως εξής:

Για ι = 1 μέχρι N:

1. ορίστε σε κάθε άσπρο τετράγωνο ένα βάρος, το οποίο είναι ο αριθμός άσπρων τετραγώνων που καλύπτονται εάν ένας κινητός αισθητήρας τοποθετείται μέσα.

2. ταξινομήστε τα άσπρα τετράγωνα κατά τα βάρη τους.

3. τοποθετήστε ένα αισθητήρα στο άσπρο τετράγωνο με το μεγαλύτερο βάρος, αλλάξτε το χρώμα των πρόσφατα καλυμμένων τετραγώνων στο μαύρο.

Bidding Protocol

Σύμφωνα με τον άπληστο ευρετικό αλγόριθμο για αυτό το NP-hard πρόβλημα, οι κινητοί αισθητήρες πρέπει να κινηθούν προς την περιοχή όπου μπορεί να ληφθεί η πιο πρόσθετη

κάλυψη. Μετά από τους κινητούς αισθητήρες η αρχική θέση για να καλύψει (να θεραπεύσει) μια άλλη τρύπα κάλυψης, μπορεί να παραγάγει μια νέα τρύπα στην αρχική θέση της. Κατά συνέπεια, ένας κινητός αισθητήρας κινείται μόνο για να καλύψει μια άλλη τρύπα εάν η αναχώρησή της δεν θα παραγάγει μια μεγαλύτερη τρύπα από αυτή που καλύπτεται. Εντούτοις, λόγω έλλειψης των σφαιρικών πληροφοριών, οι κινητοί αισθητήρες μπορούν να μην ξέρουν όπου υπάρχει η τρύπα κάλυψης. Ακόμη και με τη θέση της τρύπας κάλυψης, υπάρχει ακόμα μια μεγάλη πρόκληση, να βρεθεί η θέση στόχων μέσα στην τρύπα κάλυψης, η οποία μπορεί να φέρει την πιο πρόσθετη κάλυψη όταν τοποθετείται εκεί ένας κινητός αισθητήρας σε σύγκριση με άλλες θέσεις. Προτείνεται να αφήσουμε τους στατικούς αισθητήρες να ανιχνεύσουν τις τρύπες κάλυψης τοπικά, να υπολογίσουν το μέγεθος αυτών των τρυπών, και να καθορίσουν τη θέση στόχων μέσα στην τρύπα. Με βάση τις ιδιότητες του διαγράμματος Voronoi, οι στατικοί αισθητήρες μπορούν να βρουν τις τρύπες κάλυψης τοπικά και να παρέχουν έναν καλό τρόπο να υπολογιστεί η θέση στόχων των κινητών αισθητήρων.

Οι ρόλοι των κινητών και στατικών αισθητήρων μας παρακινούν στη σχεδίαση ενός πρωτόκολλο προσφοράς για να βοηθήσουν τη μετακίνηση των κινητών αισθητήρων. Βλέπουμε έναν κινητό αισθητήρα ως ένα κεντρικό υπολογιστή θεραπείας τρυπών. Η υπηρεσία αυτή έχει ορισμένη μια βασική τιμή, η οποία είναι η εκτίμηση της παραγόμενης τρύπας κάλυψης αφότου αφήνει την τρέχουσα θέση. Οι στατικοί αισθητήρες είναι οι πλειοδότες των υπηρεσιών κάλυψης τρυπών. Οι προσφορές τους είναι τα κατ' εκτίμηση μεγέθη των τρυπών που ανιχνεύουν. Οι κινητοί αισθητήρες επιλέγουν τις υψηλότερες προσφορές και κινούνται προς τις θέσεις στόχων που παρέχονται από τους στατικούς αισθητήρες. Το πρωτόκολλο προσφοράς τρέχει γύρο με

γύρο μετά από την περίοδο αρχικοποίησης. Κατά τη διάρκεια της περιόδου αρχικοποίησης, όλοι οι στατικοί αισθητήρες μεταδίδουν ραδιοφωνικά τις θέσεις και τις ταυτότητές τους τοπικά. Επιλέγουμε την ακτίνα ραδιοφωνικής μετάδοσης να είναι δύο hops, με την οποία οι αισθητήρες μπορούν να κατασκευάσουν το διάγραμμα Voronoi στις περισσότερες περιπτώσεις. Μετά από την περίοδο αρχικοποίησης, οι στατικοί αισθητήρες μεταδίδουν ραδιοφωνικά αυτές τις πληροφορίες πάλι μόνο όταν φθάνουν οι νέοι κινητοί αισθητήρες και χρειάζονται αυτές τις πληροφορίες για να κατασκευάσουν τα κύτταρα Voronoi τους.

Κάθε γύρος αποτελείται από τρεις φάσεις: διαφήμιση υπηρεσιών, προσφορά, εξυπηρέτηση. Στη φάση διαφημίσεων, οι κινητοί αισθητήρες μεταδίδουν ραδιοφωνικά τις βασικές τους τιμές και τις θέσεις τους σε μια τοπική περιοχή. Η βασική τιμή τίθεται αρχικά να είναι μηδέν. Στη φάση προσφοράς, οι στατικοί αισθητήρες ανιχνεύουν τις τρύπες κάλυψης τοπικά με την εξέταση των κυττάρων Voronoi τους. Εάν τέτοιες τρύπες υπάρχουν, υπολογίζουν τις προσφορές και τις θέσεις στόχων για τους κινητούς αισθητήρες. Με βάση τις λαμβανόμενες πληροφορίες από τους κινητούς αισθητήρες, ο στατικός αισθητήρας μπορεί να βρει έναν πιο κοντινό κινητό αισθητήρα ο οποίος έχει βασική τιμή χαμηλότερη από την προσφορά της, και στέλνει ένα μήνυμα προσφοράς σε αυτόν τον κινητό αισθητήρα. Στην φάση εξυπηρέτησης , ο κινητός αισθητήρας επιλέγει την υψηλότερη προσφορά και τις κινήσεις για να καλύψει-θεραπεύσει εκείνη την τρύπα κάλυψης. Η αποδεκτή προσφορά θα γίνει η νέα βασική τιμή του κινητού αισθητήρα. Μετά από την φάση εξυπηρέτησης, ένας άλλος νέος γύρος μπορεί να αρχίσει αφότου μεταδώσουν ραδιοφωνικά οι κινητοί αισθητήρες τις νέες θέσεις και τις νέες βασικές τιμές τους. Δεδομένου ότι η βασική τιμή αυξάνεται μονοτονικά, όταν δεν μπορεί να δώσει κανένας στατικός

αισθητήρας μια προσφορά υψηλότερη από την βασική τιμή των κινητών αισθητήρων, το πρωτόκολλο τερματίζεται.

Στο μήνυμα προσφοράς, οι στατικοί αισθητήρες δίνουν το κατ' εκτίμηση μέγεθος των τρυπών κάλυψης και τη θέση των στόχων που πρέπει να κινηθεί ο κινητός αισθητήρας. Αυτές οι πληροφορίες υπολογίζονται με βάση τα κύτταρα Voronoi τους. Οι στατικοί αισθητήρες κατασκευάζουν τα κύτταρα Voronoi εξετάζοντας μόνο τους στατικούς γείτονες και τους κινητούς γείτονες που δεν είναι πιθανό να κινηθούν. Αυτοί οι κινητοί αισθητήρες ανιχνεύονται με την εξέταση των βασικών τιμών τους. Εάν η βασική τιμή ενός κινητού αισθητήρα είναι μηδέν, αυτός ο αισθητήρας δεν έχει κινηθεί ακόμα και πιθανότατα θα κινηθεί σύντομα για να θεραπεύσει μερικές τρύπες κάλυψης. Κατά συνέπεια, κατά την ανίχνευση των τρυπών κάλυψης, οι στατικοί αισθητήρες δεν εξετάζουν εκείνους τους κινητούς αισθητήρες που είναι έτοιμοι να φύγουν. Για να κατασκευάσει το κύτταρο Voronoi του, κάθε αισθητήρας υπολογίζει αρχικά τους διχοτόμους των εξεταζόμενων αισθητήρων και τις δικές του βασισμένος στις πληροφορίες θέσης. Αυτοί οι διχοτόμοι διαμορφώνουν διάφορα πολύγωνα. Το μικρότερο πολύγωνο που περικυκλώνει τον αισθητήρα είναι το κύτταρο Voronoi αυτού του αισθητήρα. Κατασκευάζοντας τα κύτταρα Voronoi, οι στατικοί αισθητήρες εξετάζουν αυτά τα κύτταρα. Εάν υπάρχει μια τρύπα κάλυψης, ο στατικός αισθητήρας επιλέγει το πολύ-πολύ κορυφές Voronoi ως θέση στόχου του ερχόμενου κινητού αισθητήρα. Μέσα σε μια τρύπα κάλυψης, υπάρχουν πολλές θέσεις που ένας κινητός αισθητήρας μπορεί να βρεθεί. Εάν ο κινητός αισθητήρας τοποθετείται σε μια θέση πάρα πολύ μακριά από οποιουσδήποτε κοντινούς αισθητήρες, η κερδισμένη κάλυψη είναι η

υψηλότερη δεδομένου ότι η επικάλυψη των αισθανόμενων κύκλων μεταξύ αυτού του νέου ερχόμενου κινητού αισθητήρα και των υπαρχόντων αισθητήρων είναι η χαμηλότερη.

Η αξιολόγηση του πρωτόκολλο προσφοράς [9] γίνεται από δύο πτυχές: η ποιότητα επέκτασης που μετριέται από το χρόνο κάλυψης και επέκτασης, και το κόστος επέκτασης που μετριέται από το κόστος των αισθητήρων και της κατανάλωσης ενέργειας. Η κάλυψη αισθητήρων είναι η αρχική ανησυχία του αλγορίθμου. Ο χρόνος επέκτασης είναι μια λειτουργία του αριθμού γύρων που απαιτούνται για να επιτύχουν ορισμένη κάλυψη και του χρόνο κάθε γύρου. Η διάρκεια κάθε γύρου καθορίζεται πρώτιστα από την κινούμενη ταχύτητα των αισθητήρων, η οποία είναι μια μηχανική ιδιότητα. Κατά συνέπεια, χρησιμοποιούμε τον αριθμό γύρων για να μετρήσουμε το χρόνο επέκτασης. Η μηχανική κίνηση και η ηλεκτρική επικοινωνία καταναλώνουν την ενέργεια, αλλά η μηχανική μετακίνηση είναι ο κυρίαρχος παράγοντας. Επομένως, χρησιμοποιούν την κινούμενη απόσταση ως μετρική αξιολόγησης για την κατανάλωση ενέργειας [9]. Το κόστος αισθητήρων καθορίζεται από το συνολικό αριθμό χρησιμοποιούμενων αισθητήρων και το ποσοστό των κινητών αισθητήρων. Δεδομένου ότι όσο πιο πολλοί κινητοί αισθητήρες χρησιμοποιούνται, μπορεί να ληφθεί μια καλύτερη κάλυψη, αλλά το κόστος αισθητήρων θα αυξηθεί. Κατά συνέπεια, αξιολογούμε επίσης την ανταλλαγή μεταξύ του κόστους και της κάλυψης.

Η αξιολόγηση απόδοσης [9] δείχνει ότι το πρωτόκολλο προσφοράς μπορεί να αυξήσει την κάλυψη σημαντικά με την χαμηλή επικοινωνία, την χαμηλή πολυπλοκότητα υπολογισμού και την περιορισμένη μετακίνηση. Υπολογίζονται επίσης οι δαπάνες αισθητήρων για να φθάσουμε

σε ορισμένη κάλυψη με τη χρησιμοποίηση όλων των στατικών αισθητήρων, όλων των κινητών αισθητήρων και του μιχτού τύπου με το πρωτόκολλο προσφοράς, και διαπιστώνεται ότι ο μιχτός τύπος μπορεί να επιτύχει μια καλή ισορροπία μεταξύ της κάλυψης και του κόστους αισθητήρων.

2.5 Επανεντοπισμός αισθητήρων σε ένα WSN

Λόγω των πολλών ελκυστικών χαρακτηριστικών των κόμβων αισθητήρων όπως το μικρό μέγεθος και το χαμηλότερο κόστος, τα δίκτυα αισθητήρων έχουν υιοθετηθεί σε πολλές στρατιωτικές και αστικές εφαρμογές συμπεριλαμβανομένης της στρατιωτικής επιτήρησης, των έξυπνων σπιτιών, του μακρινού ελέγχου περιβάλλοντος, και στον έλεγχο και καθοδήγηση της ρομποτικής. Προκειμένου να αισθανθούν κατάλληλα τα φαινόμενα ενδιαφέροντος, οι κόμβοι αισθητήρων πρέπει να επεκταθούν κατάλληλα για να φθάσουν σε ένα επαρκές επίπεδο κάλυψης για την επιτυχή ολοκλήρωση των στόχων αντίληψης. Επιπλέον, μόλις επεκταθούν, οι κόμβοι αισθητήρων μπορούν να αποτύχουν, απαιτώντας τους κόμβους για να κινηθούν για να υπερνικήσουν την τρύπα κάλυψης που δημιουργείται από τον αποτυχημένο αισθητήρα. Σε αυτά τα σενάρια, είναι απαραίτητο να χρησιμοποιηθούν οι κινητοί αισθητήρες, οι οποίοι μπορούν να κινηθούν για να παρέχουν την απαραίτητη κάλυψη. Ένα παράδειγμα ενός κινητού αισθητήρα είναι το Robomote. Αυτοί οι αισθητήρες είναι μικρότεροι από $0.000047m^3$ και το κόστος τους είναι λιγότερο από 150 δολάρια.

Στα WSN υπάρχει το πρόβλημα του επανεντοπισμού αισθητήρων, δηλ., κινώντας τους προηγουμένως επεκταμένους αισθητήρες για να υπερνικήσουμε την αποτυχία άλλων κόμβων, ή για να αποκριθούμε σε ένα εμφανιζόμενο γεγονός που απαιτεί ότι ένας αισθητήρας κινείται προς τη θέση του. Αυτός ο επανεντοπισμός αισθητήρων είναι διαφορετικός από την υπάρχουσα εργασία για τους κινητούς αισθητήρες που επικεντρώνονται στην επέκταση αισθητήρων π.χ., κινούμενοι αισθητήρες για να παρέχει ορισμένη αρχική κάλυψη. Σε σύγκριση με την επέκταση αισθητήρων, ο επανεντοπισμός αισθητήρων έχει πολλές ειδικές δυσκολίες. Κατ' αρχάς, ο επανεντοπισμός αισθητήρων μπορεί να έχει μια ακριβή απαίτηση χρόνου απόκρισης. Παραδείγματος χάριν, εάν ο αισθητήρας που ελέγχει μια ασφαλισμένη, ευαίσθητη περιοχή πεθάνει, ένας άλλος αισθητήρας πρέπει να κινηθεί για να την αντικαταστήσει το συντομότερο δυνατόν. Δεύτερον, ο επανεντοπισμός δεν πρέπει να έχει επιπτώσεις στην εφαρμογή χρησιμοποιώντας αυτήν την περίοδο το δίκτυο αισθητήρων, το οποίο σημαίνει ότι ο επανεντοπισμός πρέπει να ελαχιστοποιήσει την επίδρασή του στην τρέχουσα τοπολογία αντίληψης. Τέλος, δεδομένου ότι η μετακίνηση μπορεί να είναι ακριβότερη από τον υπολογισμό και την επικοινωνία, από την άποψη της ενέργειας, οποιοσδήποτε αλγόριθμος πρέπει να ισορροπήσει τις ενεργειακές δαπάνες με το χρόνο απόκρισης. Ειδικότερα, η προσοχή πρέπει να ληφθεί για να ισορροπήσει τις ενεργειακές δαπάνες ενός μεμονωμένου κόμβου με το γενικό ενεργειακό κόστος δικτύων για να εξασφαλίσει μέγιστη διάρκεια ζωής δικτύων. Προτάθηκε [10] ένα πλαίσιο για τους κινητούς αισθητήρες κατά τρόπο έγκαιρο, αποδοτικό, και ισορροπημένο, και συγχρόνως, που διατηρεί την αρχική τοπολογία αντίληψης όσο το δυνατόν περισσότερο. Ο επανεντοπισμός αισθητήρων αποτελείται από δύο φάσεις: η πρώτη πρόκειται να βρει τους περιττούς αισθητήρες στο δίκτυο αισθητήρων, η δεύτερη είναι για να επανεντοπιστούν στη θέση

στόχου. Για την πρώτη φάση, προτείνεται σαν λύση ένα Grid-Quorum για να εντοπιστούν γρήγορα οι περιττοί αισθητήρες με τα χαμηλά γενικά έξοδα μηνυμάτων. Για τη δεύτερη φάση, προτείνονται αποδοτικά heuristics για να επιτευχθεί η καλή ισορροπία μεταξύ της ενεργειακής - αποδοτικότητας και του χρόνου επανεντοπισμού κατά τον καθορισμό της πορείας επανεντοπισμού αισθητήρων.

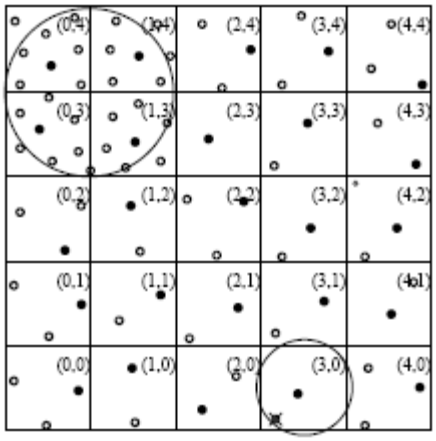
Έχουν υπάρξει διάφορες ερευνητικές προσπάθειες στην ανάπτυξη των κινητών αισθητήρων. Παραδείγματος χάριν, ένα ισχυρό cluster είναι διαθέσιμο για να συλλέξει τις πληροφορίες και να καθορίσει τη θέση στόχων των κινητών αισθητήρων. Η επέκταση αισθητήρων έχει εξεταστεί επίσης στον τομέα της ρομποτικής, όπου οι αισθητήρες επεκτείνονται ένας-ένας, χρησιμοποιώντας τις πληροφορίες θέσης των προηγουμένως επεκταμένων αισθητήρων. Αυτή η μέθοδος δεν είναι κατάλληλη στο πρόβλημα επανεντοπισμού επειδή δεν θα καλύψει τις απαιτήσεις χρόνου απόκρισης σε πολλές περιπτώσεις. Πρόσφατα, προτάθηκαν τρία πρωτόκολλα mobility-assisted για επέκταση αισθητήρων όπου οι κινητοί αισθητήρες κινούνται από τις πυκνά επεκταμένες περιοχές προς τις αραιές περιοχές για να αυξήσουν την κάλυψη. Τα πρωτόκολλα που οργανώνονται επαναληπτικά [8]. Για να επιτύχει μια καλή ισορροπία μεταξύ του αισθητήρα κόστους και αισθητήρα κάλυψης, σχεδιάστηκε ένα πρωτόκολλο προσφοράς για την ανάπτυξη των κινητών αισθητήρων στα δίκτυα αισθητήρων που αποτελούνται και από τους κινητούς και στατικούς αισθητήρες [9]. Επειδή όλοι αυτοί οι αλγόριθμοι παίρνουν ενδεχομένως διάφορες επαναλήψεις που τερματίζονται, μπορούν να μην καλύψουν τις απαιτήσεις χρόνου απόκρισης του προβλήματος επανεντοπισμού.

Θεωρητικά, τα τέσσερα πρωτόκολλα που αναφέραμε προηγουμένως [8], [9] μπορούν να χρησιμοποιηθούν και για τον επανεντοπισμό αισθητήρων. Παραδείγματος χάριν, μετά από μια αποτυχία αισθητήρων, οι γειτονικοί αισθητήρες του αποτυχημένου κόμβου μπορούν να εκτελέσουν τους αλγορίθμους. Μετά από διάφορους γύρους, οι γειτονικοί αισθητήρες θα κινηθούν για να καλύψουν την περιοχή που καλύπτεται αρχικά από τον αποτυχημένο αισθητήρα. Εντούτοις, οι κινούμενοι γειτονικοί αισθητήρες μπορούν να δημιουργήσουν τις νέες τρύπες σε εκείνη την περιοχή. Για να θεραπεύσουν αυτές τις νέες τρύπες, πρέπει να κινηθούν οι περισσότεροι αισθητήρες. Αυτή η διαδικασία συνεχίζεται έως ότου κάποια περιοχή να έχει τους περιττούς αισθητήρες και οι αισθητήρες που αφήνουν αυτήν την περιοχή δεν δημιουργούν τις νέες τρύπες. Χρησιμοποιώντας τη μέθοδο, οι αισθητήρες μπορούν να κινηθούν αρκετές φορές, σπαταλώντας την ενέργεια. Επιπλέον, δεδομένου ότι πολλοί αισθητήρες περιλαμβάνονται, μπορεί να πάρει έναν μακροχρόνιο χρόνο για να τερματιστεί ο αλγόριθμος. Με βάση αυτήν την παρατήρηση, προτείνεται να βρίσκουμε αρχικά τις θέσεις των περιττών αισθητήρων, και να σχεδιάζουμε έπειτα μια αποδοτική διαδρομή προς τον προορισμό. Για να καθοριστεί ποιος αισθητήρας είναι περιττός είναι ένα προκλητικό πρόβλημα. Είναι δύσκολο για έναν ενιαίο αισθητήρα να αποφασίσει ανεξάρτητα εάν η μετακίνησή του θα παραγάγει μια τρύπα κάλυψης. Για να λάβει μια τέτοια απόφαση, ο αισθητήρας επιθυμεί τις πληροφορίες εάν οι γείτονές του θα κινηθούν ή όχι. Πιο συγκεκριμένα, διάφοροι αισθητήρες που είναι τοποθετημένοι πολύ κοντινά πρέπει να καθορίσουν τους περιττούς αισθητήρες μεταξύ τους.

Μια αρχιτεκτονική βασισμένη στο πλέγμα είναι μια φυσική λύση για αυτό το πρόβλημα [10]. Μπορούμε να διαιρέσουμε τον τομέα στόχου σε πλέγματα. Ο αρχηγός του πλαισίου (grid head)

είναι αρμόδιος για τη συλλογή των πληροφοριών των μελών του, και τον καθορισμό της ύπαρξης των περιττών αισθητήρων βασισμένων στις θέσεις τους. Για τους περιττούς αισθητήρες που βρίσκονται στο όριο του πλέγματος, ο "grid head" πρέπει να λάβει τις αποφάσεις. Ο "grid head" μπορεί επίσης να ελέγξει τα μέλη της ομάδας του και να κινήσει μια διαδικασία επανεντοπισμού σε περίπτωση νέας αποτυχίας γεγονότος ή αισθητήρων. Μια αρχιτεκτονική βασισμένη στο πλέγμα είναι εφικτή σε ένα δίκτυο στο οποίο οι κόμβοι σχετικά τακτικά επεκτείνονται, παραδείγματος χάριν όπως θα συνέβαινε μετά από τη λήξη των προηγουμένως προτεινόμενων αλγορίθμων επέκτασης αισθητήρων [8], [9]. Αυτό προκύπτει επειδή, αντίθετα από την περίπτωση ενός δικτύου στο οποίο οι κόμβοι επεκτείνονται τυχαία, το κόστος για να οργανωθούν οι αισθητήρες στα πλέγματα είναι χαμηλό. Περαιτέρω, αυτή η οργάνωση μπορεί να διευκολύνει τη συνάθροιση στοιχείων, τη δρομολόγηση, κλπ, εκτός από την εύρεση των περιττών αισθητήρων. Με το μοντέλο βασισμένο στο πλέγμα, το πρόβλημα επανεντοπισμού αισθητήρων μπορεί να περιοριστεί σε δύο υποπροβλήματα: στην εύρεση των περιττών αισθητήρων και έπειτα στον επανεντοπισμό τους στη θέση στόχου. Το Σχήμα 2 επεξηγεί το πρόβλημα επανεντοπισμού αισθητήρων όταν χρησιμοποιούνται τα πλέγματα, οι μαύροι κόμβοι χρησιμοποιούνται για να αντιπροσωπεύσουν τους "grid heads". Κάθε πλέγμα συντάσσεται από δυάδες (tuple), της οποίας ο πρώτος αριθμός χρησιμοποιείται για να αντιπροσωπεύσει τη στήλη και ο δεύτερος αριθμός χρησιμοποιείται για να αντιπροσωπεύσει τη σειρά. Τα πλέγματα (1.3), (0.3), (1,4) και (0,4) έχουν τους περιττούς αισθητήρες. Όταν ένας αισθητήρας στο πλέγμα (3,0) πεθάνει, με συνέπεια μια τρύπα κάλυψης, ο "grid head" πρώτος πρέπει να εντοπίσει τον περιττό αισθητήρα και να επανεντοπίσει έπειτα κάποιο αισθητήρα για να καθορίσει την τρύπα κάλυψης. Για το πρώτο πρόβλημα, προτείνεται μια λύση Grid-Quorum για να προσδιοριστούν γρήγορα οι

περιττοί αισθητήρες. Για το δεύτερο πρόβλημα, προτείνεται μια σε σειρά λύση μετακίνησης για να επανεντοπιστούν οι αισθητήρες με έναν έγκαιρο και ενεργειακό αποδοτικό τρόπο.



Σχήμα 2. Το μοντέλο του συστήματος [10]

Distributed Protocol

Προτείνεται να χρησιμοποιηθούν οι γεωγραφικές πληροφορίες για να εξασφαλιστεί ότι οι κόμβοι λαμβάνουν τις σωστές αποφάσεις πριν μεταδώσουν ραδιοφωνικά με μια υψηλή πιθανότητα. Κατά συνέπεια, στις περισσότερες περιπτώσεις, κάθε αισθητήρας μεταδίδει ραδιοφωνικά μόνο μια φορά. Η λύση μας χρειάζεται δύο δομές δεδομένων: η αρχική περιοχή αναζήτησης και ο κατάλογος αναμονής. Η αρχική περιοχή αναζήτησης καθορίζεται βασισμένα στη θέση του περιττού αισθητήρα και του γεγονότος, και πρέπει να έχει μια υψηλή πιθανότητα για να καλύψει όλους τους διαδοχικούς κόμβους. Οι διαδοχικοί κόμβοι πρέπει να είναι κοντά

στη γραμμή που συνδέει τον περιττό αισθητήρα και τη θέση γεγονότος. Επομένως, δεν είναι απαραίτητο να περιληφθούν οι μακρινοί κόμβοι. Κάθε κόμβος αισθητήρων καθορίζει έναν κατάλογο αναμονής αφότου λάβει το αίτημα επανεντοπισμού για πρώτη φορά.

Ένα άλλο ζήτημα αυτού του broadcast-based πρωτοκόλλου είναι ότι ο πιθανός διαδοχικός κόμβος μπορεί να είναι εκτός της εμβέλειας επικοινωνίας του αποστολέα. Λόγω του περιορισμού της σειράς επικοινωνίας, ο διάδοχος δεν μπορεί να είναι γείτονας επικοινωνίας.

Η αξιολόγηση περιλαμβάνει τρία μέρη [10]: Κατ' αρχάς, η αποτελεσματικότητα της προτεινόμενης λύσης συγκρίνεται με το σχέδιο VOR [8]. Δεύτερον, η αποτελεσματικότητα της σειράς μετακίνησης αξιολογείται. Τέλος, η αποτελεσματικότητα του μετρικού για την επιλογή του προγράμματος απότομου πεσίματος αξιολογείται.

Η λύση επανεντοπισμού αισθητήρων που παρουσιάζεται πρώτα ψάχνει τον πιο κοντινό περιττό αισθητήρα και τον επανεντοπίζει έπειτα στη θέση στόχου. Άλλες λύσεις μπορούν να βασιστούν στην ιδέα του επανεντοπισμού του κοντινού αισθητήρα στη θέση στόχων. Το αντιπροσωπευτικό σχέδιο είναι ο VOR [8]. Ο VOR είναι επαναληπτικός. Σε κάθε γύρο, ανιχνεύει την τρύπα κάλυψης και κινεί τους κοντινούς αισθητήρες για να την θεραπεύσει. Αυτή η διαδικασία συνεχίζεται έως ότου καλυπτεί καλά ο τομέας στόχου. Επιλέγουμε τυχαία έναν αισθητήρα, μειώνουμε την ενέργειά του, και παράγουμε μια τρύπα κάλυψης. Κατόπιν, ο προτεινόμενος επανεντοπισμός αισθητήρων και ο VOR εκτελούνται για να ανακτήσουν την αποτυχία αισθητήρων. Μετριέται η απόδοση και των δύο προσεγγίσεων με τρεις μετρικές: ο αριθμός

αισθητήρων που κινήθηκε, η συνολική κατανάλωση ενέργειας, και η ελάχιστη υπόλοιπη ενέργεια.

Στον αλγόριθμο VOR, οι κοντινοί αισθητήρες κινούνται για να θεραπεύσουν την τρύπα κάλυψης. Δεδομένου ότι η μετακίνησή τους οδηγεί στις νέες τρύπες, όλο και περισσότεροι αισθητήρες περιλαμβάνονται. Η διάδοση της μετακίνησης αισθητήρων δεν κατευθύνεται στον περιττό αισθητήρα, αλλά σε όλες τις κατευθύνσεις, με συνέπεια την κίνηση της ταλάντωσης. Επίσης, στο VOR, όταν ανιχνεύεται μια τρύπα κάλυψης, οι κοντινοί αισθητήρες κινούνται ακόμα και όταν η υπόλοιπη ενέργειά τους είναι χαμηλή. Αυτό οδηγεί σε μια πολύ χαμηλότερη ελάχιστη υπόλοιπη ενέργεια. Εν περίληψη, αν και VOR είναι αποτελεσματικός αλγόριθμος στην ανάπτυξη των κινητών αισθητήρων, βρίσκοντας πρώτα τον περιττό αισθητήρα και έπειτα να επανεντοπίσει τη θέση στόχου, είναι πολύ καλύτερος για τον επανεντοπισμό αισθητήρων.

Cascaded Movement vs. Direct Movement

Ο χρόνος επανεντοπισμού μπορεί να μειωθεί σημαντικά στη προσέγγιση Cascaded Movement. Όσον αφορά στην κατανάλωση ενέργειας, η άμεση μετακίνηση είναι καλύτερη, αλλά το πλεονέκτημά του πέρα από τη Cascaded Movement είναι πολύ περιορισμένο. Αυτό αποδεικνύει ότι η Cascaded Movement είναι ενεργειακά αποδοτική. Αφ' ετέρου, η ελάχιστη υπόλοιπη ενέργεια της χρησιμοποίησης Cascaded Movement είναι πολύ καλύτερη από αυτή της Direct Movement. Εάν ο περιττός αισθητήρας έχει τη σχετικά υψηλή δύναμη, η κίνηση της άμεσα προς τη θέση στόχου μπορεί να μην έχει επιπτώσεις στην ελάχιστη υπόλοιπη δύναμη διαφορετικά,

μπορεί σημαντικά να μειώσει την ελάχιστη υπόλοιπη δύναμη, ειδικά όταν η κινούμενη απόσταση είναι μεγάλη. Αυτό εξηγεί γιατί η ελάχιστη υπόλοιπη ενέργεια μειώνεται αναλογικά όταν η απόσταση αυξάνεται στην άμεση προσέγγιση μετακίνησης.

Η μετρική που χρησιμοποιείται για να πάρει το καλύτερο πρόγραμμα cascading είναι να ελαχιστοποιήσει τη διαφορά μεταξύ της συνολικής κατανάλωσης ενέργειας και της ελάχιστης υπόλοιπης δύναμης. Η συνολική κατανάλωση ενέργειας της προσέγγισής [10] είναι παρόμοια με την άμεση προσέγγιση μετακίνησης, η οποία είναι βέλτιστη, συγκρίνεται μόνο η προσέγγισή με μια άλλη εναλλακτική λύση που μεγιστοποιεί την ελάχιστη υπόλοιπη δύναμη. Μεταξύ αυτών των δύο τοποθετήσεων, η προσέγγισή Cascaded Movement σώζει περισσότερη ενέργεια όταν η υπόλοιπη ενέργεια είναι παρόμοια. Ο λόγος είναι ο ακόλουθος. Οι αισθητήρες με σχετικά περισσότερη ενέργεια πρέπει να περιληφθούν για να μεγιστοποιήσουν την ελάχιστη υπόλοιπη ενέργεια. Όταν η υπόλοιπη ενέργεια είναι παρόμοια, δεν είναι πιθανό να βρει τους κοντινούς αισθητήρες με την υψηλή ενέργεια. Κατόπιν, οι μακρινοί αισθητήρες είναι πιθανότερο να περιληφθούν, και περισσότερη ενέργεια θα καταναλωθεί σε αντίθεση με τη λύση. Αφ' ετέρου, όταν η υπόλοιπη ενέργεια είναι παρόμοια, το μειονέκτημα της λύσης [10] είναι λίγο μεγαλύτερο δεδομένου ότι μόνο οι κοντινοί αισθητήρες περιλαμβάνονται στον επανεντοπισμό. Αυτοί οι αισθητήρες μπορούν να γίνουν οι αισθητήρες με την ελάχιστη υπόλοιπη ενέργεια μετά από τον επανεντοπισμό και η ελάχιστη υπόλοιπη ενέργεια μεταξύ όλων των αισθητήρων μειώνεται συνεπώς. Όταν η υπόλοιπη ενέργεια είναι πολύ διαφορετική, και οι δύο προσεγγίσεις έχουν την παρόμοια ελάχιστη παραμονή ενέργειας δεδομένου ότι ένας αισθητήρας με την ελάχιστη υπόλοιπη ενέργεια είναι πιθανόν να μην περιλαμβάνεται στον επανεντοπισμό και η ελάχιστη υπόλοιπη ενέργεια του δικτύου δεν αλλάζει μετά από τον επανεντοπισμό. [10]

Προτείνεται [20] ένα μοντέλο για να προσδιορίσει τη διάρκεια ζωής του εντοπισμού των στόχων σε ασύρματο δίκτυο αισθητήρων. Το μοντέλο είναι στατικής "cluster-based" αρχιτεκτονικής και στοχεύει να παρέχει δύο παράγοντες. Κατ' αρχάς, είναι η αύξηση της διάρκειας ζωής του ασύρματου δικτύου αισθητήρων εντοπισμού των στόχων. Αφετέρου, πρόκειται να επιτρέψει ένα καλό αποτέλεσμα εντοπισμού με τη μικρή κατανάλωση ενέργειας για κάθε αισθητήρα στο δίκτυο. Το μοντέλο αποτελείται από ετερογενείς αισθητήρες και κάθε αισθανόμενος κόμβος μέλος σε μια συστάδα χρησιμοποιεί 2 στάδια λειτουργίας – το ενεργό στάδιο και το στάδιο ύπνου. Τα αποτελέσματα απόδοσης διευκρινίζουν ότι η προτεινόμενη αρχιτεκτονική καταναλώνει τη λιγότερη ενέργεια και αυξάνει τη διάρκεια ζωής από τις συγκεντρωμένες και δυναμικές αρχιτεκτονικές συγκέντρωσης, για το δίκτυο αισθητήρων εντοπισμού των στόχων.

Με τη χρησιμοποίηση της ομοιόμορφης διανομής για τοποθέτηση αισθητήρων, διαίρεσαν ολόκληρο το δίκτυο αισθητήρων σε ίσες περιοχές. Το σύστημα χρησιμοποιεί τον αποκεντρωμένο εντοπισμό των στόχων έτσι υπάρχει μόνο ένας αρχηγός συστάδων σε κάθε περιοχή. Ο αρχηγός συστάδων που καλείται επίσης και κόμβος επεξεργασίας (PN), συλλέγει τα αισθανόμενα στοιχεία από τους ακουστικούς και τους ανιχνευτές φωτογραφίας αισθητήρες στην περιοχή τους. Κατόπιν επεξεργάζεται τα λαμβανόμενα στοιχεία και στέλνει την υπογραφή στόχων στο σταθμό βάσης.

Το προτεινόμενο σύστημα εντοπισμού των στόχων αποτελείται από τρεις κύριες διαδικασίες:

• Ανίχνευση στόχων (Target detection)

• Ακουστικός εντοπισμός πηγής (Acoustic source localization)

• Στάδιο εκτίμησης και καταδίωξης στόχου (Target state estimation and tracking)

Στο έγγραφο [20], παρουσιάζεται ένα ενεργειακά αποδοτικό ετερογενές ασύρματο δίκτυο αισθητήρων για τον εντοπισμό των στόχων. Τα αποτελέσματα επίδοσης δείχνουν ότι η κατανάλωση ενέργειας για ολόκληρο το δίκτυο αισθητήρων μειώνεται σημαντικά από τη στατική αρχιτεκτονική συστάδων σε σύγκριση με τη δυναμική συγκέντρωση και τις κεντρικοποιημένες (centralized) αρχιτεκτονικές. Αυτό παρέχει στο ασύρματο δίκτυο αισθητήρων βελτίωση στη διάρκεια ζωής για την καταδίωξη του στόχου.

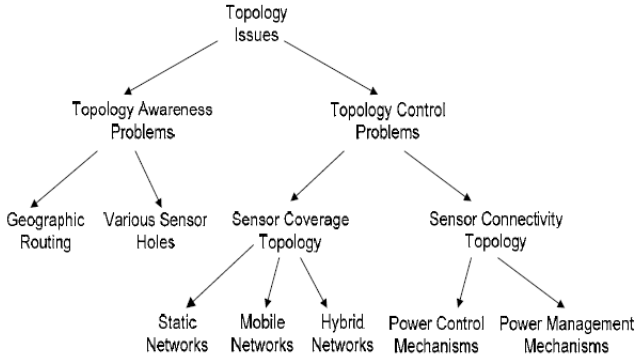
Έλεγχος μιας μεγάλης περιοχής με τα στάσιμα δίκτυα αισθητήρων απαιτεί έναν πολύ μεγάλο αριθμό κόμβων που με την τρέχον τεχνολογία υπονοεί ένα απαγορευτικό κόστος. Το κίνητρο της εργασίας [24] είναι να αναπτυχθεί μια αρχιτεκτονική όπου ένα σύνολο κινητών αισθητήρων θα συνεργαστεί με τους στάσιμους αισθητήρες προκειμένου να ανιχνευθεί αξιόπιστα και να βρεθεί ένα γεγονός. Η κύρια ιδέα αυτής της συνεργάσιμης αρχιτεκτονικής είναι ότι οι κινητοί αισθητήρες πρέπει να επιλέξουν τις περιοχές που καλύπτονται πιο ελάχιστα (λιγότερο ελεγχόμενες) από τους στάσιμους αισθητήρες. Επιπλέον, όταν οι στάσιμοι αισθητήρες έχουν μια «υποψία» ότι ένα γεγονός μπορεί να είχε εμφανιστεί, την εκθέτουν σε έναν κινητό αισθητήρα που μπορεί να κινηθεί πιο κοντά προς την πιθανή περιοχή και μπορεί να επιβεβαιώσει εάν το γεγονός έχει εμφανιστεί ή όχι. Ένα σημαντικό συστατικό της προτεινόμενης αρχιτεκτονικής είναι ότι οι κινητοί κόμβοι αυτόνομα αποφασίζουν την πορεία τους βασισμένη μόνο στις τοπικές πληροφορίες (οι πεποιθήσεις και οι μετρήσεις τους καθώς επίσης και πληροφορίες που συλλέγονται από τους στάσιμους αισθητήρες στο πεδίο επικοινωνίας τους). Πιστεύουν ότι αυτή

η προσέγγιση είναι κατάλληλη στα πλαίσια των ασύρματων δικτύων αισθητήρων δεδομένου ότι δεν είναι εφικτό να έχει μια ακριβή σφαιρική άποψη της κατάστασης του περιβάλλοντος.

Στο έγγραφο [24] προτείνουν μια αρχιτεκτονική συνεργασίας ανίχνευσης γεγονότος για WSNs που αποτελείται από έναν μεγάλο αριθμό στάσιμων κόμβων και μερικών κινητών κόμβων. Το όφελος αυτής της αρχιτεκτονικής είναι ότι οι κινητοί κόμβοι συνεργάζονται με τους στάσιμους κόμβους έτσι ώστε επιλέγουν τις περιοχές που ελάχιστα καλύπτονται από τους στάσιμους κόμβους. Κατά αυτόν τον τρόπο, τα γεγονότα που θα είχαν παραμείνει μη ανιχνευθέντα μπορούν τώρα να ανιχνευθούν. Στην προτεινόμενη αρχιτεκτονική διάφορα ζητήματα είναι ακόμα ανοικτά για έρευνα. Κατ' αρχάς, πότε οι κινητοί διακόπτουν τους στόχους (από το κέντρο της τρύπας στην πιθανή περιοχή). Σημειώστε ότι εάν το κατώτατο όριο υποψίας τίθεται πολύ χαμηλό, υπάρχει ένας κίνδυνος ότι οι περισσότεροι αισθητήρες θα εκθέσουν μια υποψία και κατά συνέπεια ο κινητός κόμβος θα σπαταλήσει το χρόνο του που αποκρίνεται στους ψεύτικους συναγερμούς. Δεύτερον, να ερευνηθούν δυναμικότερα περιβάλλοντα όπου οι πηγές εμφανίζονται και εξαφανίζονται δυναμικά και στάσιμοι αισθητήρες αποτυγχάνουν επίσης ή ανακατανέμονται τυχαία. Σε αυτήν την περίπτωση, το G πρέπει να πολλαπλασιαστεί με τα κατάλληλα ποσοστά απόρριψης έτσι ώστε να απεικονίζει εκείνη την αβεβαιότητα. Τέλος, υπάρχει ανάγκη αλγορίθμων για τους χάρτες των κινητών σε μια συνολικά κατανεμημένη μόδα καθώς επίσης και αλγορίθμων για τους χάρτες δύο κινητών κόμβων μαζί (όταν έρχονται μέσα στο πεδίο επικοινωνίας).

2.6 Ζητήματα τοπολογίας στα ασύρματα δίκτυα αισθητήρων

Τα ζητήματα τοπολογίας έχουν γίνει θέμα προσοχής στα ασύρματα δίκτυα αισθητήρων (WSN). Ενώ οι εφαρμογές WSN βελτιστοποιούνται κανονικά από τη δεδομένη υπάρχουσα τοπολογία δικτύων, μια άλλη τάση είναι να βελτιστοποιηθούν τα ασύρματα δίκτυα αισθητήρων με τη βοήθεια του ελέγχου τοπολογίας. Διάφορες προσεγγίσεις έχουν επενδυθεί σε αυτήν την περιοχή, όπως κατευθυνόμενη τοπολογία δρομολόγησης, σχέδια συνεργασίας, κάλυψη αισθητήρων βασισμένη στον έλεγχο τοπολογίας και συνδετικότητα δικτύων βασισμένη στον έλεγχο τοπολογίας. Τα περισσότερα από τα σχέδια έχουν αποδείξει να είναι σε θέση να παρέχουν ένα καλύτερο έλεγχο δικτύου και επίδοση επικοινωνίας με την παρατεταμένη διάρκεια ζωής συστημάτων. Στο έγγραφο ερευνών [14], παρέχουν ένα σύνολο απόψεων των μελετών σε αυτήν την περιοχή. Με τη συνόψιση των προηγούμενων επιτευγμάτων και την ανάλυση υφιστάμενων προβλημάτων, επισημαίνουν επίσης τις πιθανές ερευνητικές κατευθύνσεις για μελλοντική εργασία.



Σχήμα 3: Ταξινόμηση θεμάτων τοπολογίας στα WSNs [14]

Στο άρθρο [14] ερευνών, έχουν αναθεωρήσει δύο σημαντικά ζητήματα τοπολογίας σε WSNs, δηλαδή τη συνειδητοποίηση τοπολογίας και τον έλεγχο τοπολογίας. Προβλήματα συνειδητοποίησης τοπολογίας κατασκευάζουν εφαρμογές ή ανώτερα πρωτόκολλα για να προσαρμοστεί η υπάρχουσα τοπολογία. Χαρακτηριστικές προσεγγίσεις που εφαρμόζονται μέσα σε αυτήν την κατηγορία δεν εξετάζουν ενεργά τη βελτίωση της τοπολογίας από μόνη της για τις συγκεκριμένες εφαρμογές. Μηχανισμοί ελέγχου τοπολογίας εστιάζουν περισσότερο στην κατασκευή ενεργειακά αποδοτικής και αξιόπιστης τοπολογίας δικτύου και κανονικά να μην αγγίζει τις μεμονωμένες εφαρμογές. Έτσι πρώτο σημαντικό ερώτημα που προβάλλουν είναι πώς να αφορά ο μηχανισμός ελέγχου τοπολογίας την ανώτερη τοπολογία ενήμερων εφαρμογών πιο στενά σε WSNs. Για τα προβλήματα ελέγχου τοπολογίας, τοπολογία κάλυψης αισθητήρων και τοπολογία συνδετικότητας αισθητήρων ήταν ξεχωριστά συζητημένα στην περισσότερη βιβλιογραφία. Εντούτοις, ενώ η τοπολογία κάλυψης αντίληψης αντιπροσωπεύει τη δυνατότητα αντίληψης δικτύων, η τοπολογία συνδετικότητας εάν είναι τόσο καλά διατηρημένη όσο μια ανάγκη για επιτυχής παράδοση πληροφοριών, συμπεριλαμβανομένων των ερωτήσεων, αισθανόμενων στοιχείων και μηνυμάτων ελέγχου. Πώς να κατασκευαστεί μια βελτιστοποιημένη τοπολογία κάλυψης διατηρώντας τη συνδετικότητα αποδοτικού και χαμηλότερου κόστους δεν είναι καλά κατανοητό και αξίζει περαιτέρω μελέτη. Ο έλεγχος δύναμης και η διαχείριση δύναμης είναι δύο διαφορετικοί τύποι μεθόδων ελέγχου τοπολογίας. Ο συνδυασμός των δύο ακόμα καλά δεν έχει μελετηθεί. Αυτοί θεωρούν ότι με την ενσωμάτωση του ελέγχου δύναμης και της διαχείρισης δύναμης, είναι δυνατό να παρέχει αξιοπρόσεχτες βελτιώσεις στην τοπολογία δικτύων και αποδοτικότητες της ενεργειακής χρήσης. Αυτό είναι ένα άλλο ενδιαφέρον ερευνητικό θέμα για τους ερευνητές στον τομέα. Τέλος, παρουσιάσαν μια περιεκτική έρευνα στα

ζητήματα τοπολογίας για WSNs και μας έχουν παρέχει ταξινομήσεις προβλημάτων και προσεγγίσεων. Κάτω από αυτό το πλαίσιο, απαριθμούν, αναθεωρούν και συγκρίνουν μερικές κλασσικές εργασίες στον τομέα. Δίνουν έμφαση στις προκλήσεις σε αυτό το θέμα και επισημάνουν κάποιες μελλοντικές κατευθύνσεις έρευνας.

Category	*Approach*	*Proposed Solution*	*Main Assumptions*	*Characteristics*
Static Network	Partial Coverage	PEAS [47]	Power dynamic adjustment	Distributed sleeping schedule
		Rotating coverage[7]	synchronized clocks, sensing range	Distributed sleeping schedule, guarantee finite delay bound
	Single Coverage	OGDC[51]	Location info, uniform sensing disk	Residual energy consideration
		Sponsored Area[36]	Location info.	Sector based coverage calculations
		Extended-Sponsored Area[18]	Location info, synchronized clock	Uniform disk sensing model
	Multiple Coverage	CCP[41]	Location info	Configurable degree of coverage.
		k-UC, k-NC[17]	Location info	Non-unit disk model supported
		Differentiated [46]	Location info, synchronized clock	Grid based differentiated degree of coverage
Mobile Networks	Computational Geometry	VEC, VOR, Minmax [40]	Location info	Localized, Scalable, Distributed.
		Co-Fi [13]	Location info, Nodes predict its death	Single coverage based. Residual energy considerations.
	Virtual Forces	Potential Fields[15]	Range and bearing	Scalable, Distributed. No local communication required.
		DSS[14]	Location info	Scalable, Distributed. Residual energy based.
Hybrid Networks	Single Mobile sensor	Single Robot[2]	Location info	Distributed. No multi-hop communications.
	Multiple Mobile Sensor	Bidding Protocol[39]	Location info	Voronoi diagram is used for single coverage requirement.

Πίνακας 1: Μια σύγκριση των διαφορετικών προσεγγίσεων κάλυψης αισθητήρων [14]

Ο πίνακας 1 συνοψίζει πιο πάνω τους μηχανισμούς διαχείρισης δύναμης, και μας δίνει μια σε βάθος γνώση των χαρακτήρων των προτεινόμενων μηχανισμών.

2.7 Έλεγχος κινητικότητας για την κάλυψη στα WSNs

Η ύπαρξη της κινητικότητας στα Ασύρματα Δίκτυα Αισθητήρων είναι πλέον ένα συνήθες χαρακτηριστικό τους το οποίο μπορεί να συμβάλει και στην επίτευξη του ελέγχου κάλυψης σε τέτοια δίκτυα. Ακολουθούν πιο κάτω αλγόριθμοι για τον έλεγχο της κάλυψης σε ασύρματα δίκτυα αισθητήρων που χρησιμοποιούν την κινητικότητα για να ολοκληρώσουν τις διαδικασίες τους και να επιτύχουν τον στόχο τους.

Ο αλγόριθμος SR – Snake like cascading Replacement

Στο έγγραφο [16], προτείνουν μια νέα μέθοδο ελέγχου για να καλύψουν τις «τρύπες» στα ασύρματα δίκτυα αισθητήρων. Πολλές εφαρμογές αντιμετωπίζουν συχνά το πρόβλημα των τρυπών όταν τίθενται εκτός λειτουργίας μερικοί κόμβοι αισθητήρων από τη συνεργασία λόγω των αποτυχιών και της μη σωστής συμπεριφοράς τους. Αυτές οι τρύπες μπορούν να εμφανιστούν δυναμικά, και ένα τέτοιο πρόβλημα δεν μπορεί να λυθεί εντελώς με την απλή επέκταση περισσότερων περιττών αισθητήρων. Με έναν συγχρονισμό γύρω από κάθε τρύπα που χρησιμοποιεί τον κατευθυνόμενο κύκλο του Χάμιλτον (Hamilton cycle), μια (και μόνο μια) φιδωτή cascading διαδικασία αντικατάστασης θα αρχικοποιηθεί στην τοπική περιοχή προκειμένου να συμπληρωθεί εκείνη η κενή περιοχή με έναν εφεδρικό κόμβο. Κατά αυτόν τον τρόπο, η συνδετικότητα δικτύου και η κάλυψη μπορούν να εγγυηθούν. Τα αναλυτικά και πειραματικά αποτελέσματά τους παρουσιάζουν ουσιαστικές βελτιώσεις μιας τέτοιας αντικατάστασης έναντι του καλύτερου γνωστού μέχρι σήμερα αποτελέσματος.

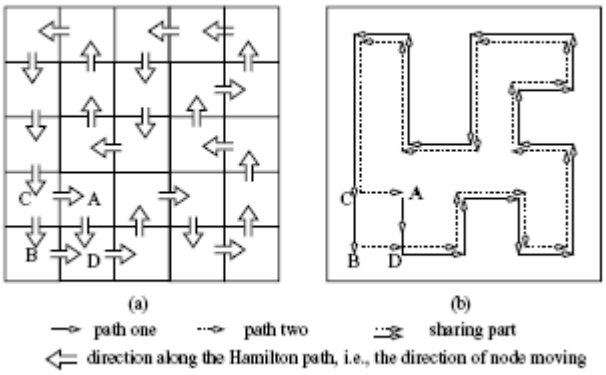
Περιγραφή της τεχνικής

Αλγόριθμος 1: Σχέδιο ελέγχου κινητικότητας βασισμένο στον κατευθυνόμενο κύκλο του Χάμιλτον.

1. Στον επικεφαλή (head) u, η ακόλουθη διαδικασία αντικατάστασης θα αρχικοποιηθεί όταν δεν μπορεί να βρει το u τον επικεφαλή στο πλέγμα διαδόχων κατά μήκος του κατευθυνόμενου κύκλου του Χάμιλτον δηλ., ένα κενό πλέγμα σε μια τέτοια κατεύθυνση ανιχνεύεται.
2. Βρείτε έναν εφεδρικό κόμβο στο πλέγμα του u, κόμβος β, για να κινηθεί σε εκείνη την κενή περιοχή προτού να αρχίσει ο επόμενος κύκλος.
3. Εάν το πιο πάνω βήμα αποτυγχάνει, επαναλάβετε τα ακόλουθα βήματα έως ότου μπορεί ο δηλωμένος κόμβος u να βρει έναν εφεδρικό κόμβο στο πιο πάνω βήμα: (α) στείλτε την ανακοίνωση στο προηγούμενο πλέγμα που ρωτά για μια αντικατάσταση για το u το ίδιο. (β) Περιμένετε έως ότου λαμβάνει ο αντίστοιχος επικεφαλής w αυτήν την ανακοίνωση. (γ) Μετακίνησε το u στο κενό πλέγμα πριν από τον επόμενο κύκλο που αρχίζει, δηλ., αφήνοντας το τρέχον πλέγμα κενό για την cascading αντικατάσταση.

Αλγόριθμος 2: Σχέδιο ελέγχου κινητικότητας για ένα σύστημα πλέγματος με το διπλό μονοπάτι (dual-path) του κύκλου Χάμιλτον.

1. Καθορίστε τα πλέγματα Α, Β, Γ, και Δ.



Σχήμα 4: Διπλές πορείες στην κατασκευή του κύκλου του Χάμιλτον σε ένα σύστημα πλέγματος 5 × 5 [16]

2. Όταν το πλέγμα Α γίνεται κενό, το Γ θα κινήσει μια διαδικασία αντικατάστασης. Μια τέτοια αντικατάσταση θα τεντώσει κατά μήκος της διπλής πορείας με τον Αλγόριθμο 1. Ομοίως, όποτε το πλέγμα Β γίνεται κενό, η αντικατάστασή της που αρχίζει στο Γ θα τεντώσει κατά μήκος μιας πορείας .

3. Όταν το πλέγμα Δ γίνεται κενό, μόνο το Β θα κινήσει τη διαδικασία αντικατάστασης και μια τέτοια αντικατάσταση θα τεντώσει κατά μήκος μιας πορείας. Εντούτοις, στο πλέγμα Γ, προτιμάται πάντα το πλέγμα Α με τους εφεδρικούς κόμβους προτού να συνεχίσει η αντικατάσταση που τεντώνει κατά μήκος μιας πορείας.

4. Όταν οποιοδήποτε άλλο πλέγμα γίνεται κενό, η αντικατάστασή της θα ακολουθήσει την κατεύθυνση του κύκλου του Χάμιλτον έως ότου φθάνει στο πλέγμα Δ. Από το Δ, είτε Α είτε Β θα δηλωθεί όταν έχει οποιοσδήποτε από αυτούς τουλάχιστον έναν εφεδρικό κόμβο.

Παρατηρήσεις και Αποτελέσματα (N είναι ο αριθμός των εφεδρικών κόμβων)

1. Ένα από τα έγγραφα (Z. Jiang, J. Wu, A. Agah, and B. Lu. Topology control for secured coverage in wireless sensor networks. *Proc. of* 3rd *Workshop on Wireless Sensor Networks Security (WSNS'07)*) αναφοράς υποστηρίζει ότι μεταξύ όλων των υπαρχόντων μεθόδων για τον καθορισμό της τρύπας, το σχέδιο AR έχει την καλύτερη απόδοση όσον αφορά στο συνολικό αριθμό μετακινήσεων κόμβων και τη συνολική κινούμενη απόσταση. Όμως, μερικές περιττές διαδικασίες και μετακινήσεις κόμβων απαιτούνται σε αυτήν την μέθοδο. Λόγω της χρήσης της κατευθυνόμενης πορείας του Χάμιλτον, οι διαδικασίες αντικατάστασης που κινούνται γύρω από το κενό πλέγμα μπορούν να συγχρονιστούν. Κατά συνέπεια, λιγότερο από 50% των διαδικασιών αντικατάστασης απαιτούνται στο SR. Ο συνολικός αριθμός μετακινήσεων κόμβων και η αντίστοιχη συνολική κινούμενη απόσταση μπορούν να μειωθούν ουσιαστικά.

2. Όταν $N < 55$, SR απαιτούν μια μακριά πορεία κατά μήκος του κύκλου του Χάμιλτον για να πλησιάσουν τον εφεδρικό κόμβο. Περισσότερες μετακινήσεις κόμβων και κινούμενη απόσταση απαιτούνται στη μέθοδο SR. Όμως, η μέθοδος του AR έχει 10% ~ 20% αποτυχίες στις διαδικασίες αντικατάστασης ενώ το ποσοστό επιτυχίας είναι πάντα 100% στη μέθοδο SR, δηλαδή η κάλυψη επιτήρησης είναι λιγότερο γερή (εύρωστη) στη μέθοδο AR στα δίκτυα με τη χαμηλότερη πυκνότητα κόμβων.

3. Όταν $N \geq 55$ (δηλ., > 1.22 ενεργοί κόμβοι ανά πλέγμα) που είναι πιο κοινό στις πραγματικές εφαρμογές, το SR απαιτεί τις λιγότερες μετακινήσεις κόμβων και λιγότερη κινούμενη απόσταση

κρατώντας το ποσοστό επιτυχίας υψηλότερο από το AR. Όπως φαίνεται στα αποτελέσματα, η νέα τους προσέγγιση SR είναι οικονομικώς πιο αποδοτική από την AR σε συνηθισμένες περιπτώσεις.

4. Η μετακίνηση cascading υιοθετείται και στον AR και στον SR αλγόριθμο. Η διαδικασία αντικατάστασης SR είναι ακριβώς μια από τις περιπτώσεις των διαδικασιών αντικατάστασης SR που είναι κατά μήκος ενός ειδικού μονοπατιού. Το SR δεσμεύει το ίδιο όριο συγκλίνουσας ταχύτητας όπως το AR, το οποίο έχει παρουσιαστεί.

5. Ένας συντομότερος δρόμος κατά μήκος του κύκλου του Χάμιλτον μπορεί να μειώσει το μήκος της πορείας για τη διαδικασία αντικατάστασης για να πλησιάσει έναν εφεδρικό κόμβο. Η κατασκευή ενός τέτοιου συντομότερου δρόμου θα είναι η μελλοντική εργασία τους και περαιτέρω αύξηση της σύγκλισης της ταχύτητας του SR. Κατά συνέπεια, το κόστος του SR θα μειωθεί πολύ στις περιπτώσεις όταν N <55.

Στο έγγραφο [16], έχουν παρουσιάσει μια πιο οικονομικώς αποδοτική, φιδωτή διαδικασία αντικατάστασης για να καλύψουν τις τρύπες επιτήρησης των WSNs όπου όλοι οι αισθητήρες που επεκτείνονται σε ορισμένες περιοχές αντίληψης είναι εκτός λειτουργίας από τη συνεργασία. Κατά συνέπεια, η συνδετικότητα και η κάλυψη των WSNs μπορούν να εγγυηθούν, ακόμα και όταν αλλάζει δυναμικά η θέση εργασίας των κόμβων. Στις μεθόδους τους, μόνο 1-hop γειτονιά χρησιμοποιείται, και η ρύθμιση των κόμβων μπορεί να ελεγχθεί μέσα στην τοπική περιοχή κάτω από ένα μοντέλο συγχρονισμού βασισμένο στον κύκλο του Χάμιλτον (Hamilton cycle). Τα

αναλυτικά και πειραματικά αποτελέσματα παρουσιάζουν την προτεινόμενη μέθοδο να είναι εύρωστη και εξελικτική με τις ελαχιστοποιημένες δαπάνες. Στη μελλοντική εργασία τους, ένα αποτελεσματικότερο πρότυπο συγχρονισμού θα μελετηθεί για να μειώσει περαιτέρω το μήκος της πορείας σε κάθε αντικατάσταση.

Το άρθρο [17] παρουσιάζει αλγόριθμους του ελέγχου και συντονισμού για τις ομάδες οχημάτων. Η εστίαση είναι στα αυτόνομα δίκτυα οχημάτων εκτελώντας τους διανεμημένους στόχους αντίληψης όπου κάθε όχημα διαδραματίζει το ρόλο ενός κινητού “tunable” αισθητήρα. Το έγγραφο προτείνει τους αλγορίθμους καθόδου κλίσης για μια κατηγορία λειτουργιών χρησιμότητας που κωδικοποιούν τη βέλτιστη κάλυψη και “sensing” πολιτικές. Η προκύπτουσα συμπεριφορά κλειστών βρόγχων (closed-loop) είναι προσαρμοστική, διανεμημένη, ασύγχρονη, και επαληθεύσιμη σωστά.

Συγκεκριμένα έχουν παρουσιάσει [17] μια νέα προσέγγιση στο συντονισμό αλγορίθμων για τα δίκτυα πολυ-οχημάτων. Το σχήμα μπορεί να θεωρηθεί ως νόμος αλληλεπίδρασης μεταξύ των πρακτόρων και δεδομένου τέτοιο που είναι εκτελέσιμο σε μια διανεμημένη ασύγχρονη μόδα. Οι πολυάριθμες επεκτάσεις εμφανίζονται να αξίζουν την επιδίωξή τους. Προγραμματίζουν να ερευνήσουν τον καθορισμό των “non-convex” περιβαλλόντων και των μη-ισοτροπικών αισθητήρων. Εφαρμόζουν αυτήν την περίοδο αυτούς τους αλγορίθμους σε ένα δίκτυο για όλα τα εδάφη οχημάτων. Επίσης, προγραμματίζουν να επεκτείνουν τους αλγορίθμους για να παρέχουν εγγυημένα αποφυγή των συγκρούσεων και στη δυναμική οχημάτων που δεν είναι τοπικά ελεγχόμενη.

Ο αλγόριθμος Passive Model (PM) & Active Model (AM)

Στο άρθρο [18], παρουσιάζουν μια νέα μέθοδο ελέγχου που διενεργεί μερικές προσαρμογές κόμβων στις τοπικές περιοχές σε μια προσπάθεια να καλύψουν τις «τρύπες» στα ασύρματα δίκτυα αισθητήρων. Πολλές εφαρμογές ασφάλειας αντιμετωπίζουν συχνά το πρόβλημα των τρυπών όταν μερικοί κόμβοι αισθητήρων είναι μη ενεργοί στην συνεργασία λόγω των αποτυχιών τους και της μη σωστής λειτουργίας τους. Επηρεάζονται από κακόβουλες επιθέσεις, αυτές οι τρύπες μπορούν να εμφανιστούν δυναμικά και ένα τέτοιο πρόβλημα δεν μπορεί να λυθεί εντελώς με απλά την επέκταση περισσότερων περιττών αισθητήρων. Προτείνουν μια φιδωτή διαδικασία cascading αντικατάστασης σε μια τοπική περιοχή για να γεμίσουν την κενή περιοχή με τους εμπιστευμένους / ενεργούς (trusted) κόμβους. Μόνο η γειτονιά ενός hop χρησιμοποιείται στην προσέγγισή τους. Οι εφαρμογές είναι κάτω από ένα παθητικό πρότυπο (passive model) και κάτω από ένα ενεργητικό πρότυπο (active model) και περιγράφονται τα μοντέλα αυτά στο έγγραφο. Τα αποτελέσματα της προσομοίωση της νέα τους μέθοδος ελέγχου παρουσιάζουν ουσιαστικές βελτιώσεις στη συνολική κινούμενη απόσταση, στο συνολικό αριθμό κινήσεων, και στη διαδικασία σύγκλισης ταχυτήτων, έναντι του καλύτερου γνωστού μέχρι σήμερα αποτελέσματος.

Στο παθητικό μοντέλο, το άδειο πλέγμα ανιχνεύεται μόνο όταν η ροή επικοινωνίας χρειάζεται να περάσει από αυτό το πλέγμα. Ενώ στο ενεργητικό μοντέλο, κάθε αρχηγός του πλέγματος θα ελέγχει ολόκληρη την περιοχή των πλεγμάτων των γειτόνων του. Κάθε φορά που ένα άδειο πλέγμα επέρχεται, μια διαδικασία αντικατάστασης θα κινηθεί αμέσως στα γειτονικά πλέγματα.

Σημειώνεται ότι η κίνηση ενός κόμβου κατά την διαδικασία της αντικατάστασης μπορεί να προκαλέσει μια άλλη αντικατάσταση για αυτόν τον συγκεκριμένο κόμβο. Στα πειραματικά αποτελέσματα τους έδειξαν ότι η cascading κίνηση μπορεί να συγκλίνει γρήγορα κάτω και από τα δύο μοντέλα (passive model και active model).

Αλγόριθμος 1: Σχήμα ελέγχου κάτω από το παθητικό μοντέλο

1. Σε ένα αναμεταδότη κόμβο α, η ακόλουθη διαδικασία αντικατάστασης θα κινηθεί όταν δεν μπορεί να βρει το α τον διάδοχο κόμβο για να διασχίσει το επόμενο πλέγμα κατά μήκος της πορείας δηλ., ένα κενό πλέγμα στην κατεύθυνση αποστολής ανιχνεύεται.

2. Βρείτε έναν εφεδρικό εμπιστευμένο (spare trusted node) κόμβο στο πλέγμα του α, κόμβος β, για να κινηθείτε σε εκείνη την κενή περιοχή προτού να αρχίσει ο επόμενος γύρος.

3. Εάν το πιο πάνω βήμα αποτυγχάνει, επαναλάβετε ότι ακολουθεί έως ότου μπορεί να βρει ο δηλωμένος κόμβος α έναν εφεδρικό εμπιστευμένο κόμβο β στο πιο πάνω βήμα: (α) επιλέξτε ένα γειτονικό πλέγμα εκτός από το κενό. Ένα πλέγμα με εφεδρικούς εμπιστευμένους κόμβους προτιμάται πάντα. (β) Στείλετε την ανακοίνωση που συνδέει μια τέτοια επιλογή που ρωτά για μια αντικατάσταση του α. (γ) Μετακινείται το α στο κενό πλέγμα πριν αρχίσει ο επόμενος γύρος δηλ., αφήνοντας το τρέχον πλέγμα κενό για την cascading αντικατάσταση.

Αλγόριθμος 2: Σχήμα ελέγχου κάτω από το ενεργητικό μοντέλο

1. Για οποιοδήποτε επικεφαλή πλέγματος α που ανιχνεύει ένα κενό γειτονικό πλέγμα, εάν καμία ανακοίνωση δεν παραλαμβάνεται στον προηγούμενο κύκλο από εκείνη την περιοχή (για να αποφύγει το overreaction), μια διαδικασία αντικατάστασης κινείται αφότου καθορίζεται η αντίστοιχη περιοχή Δ.

2. Για οποιοδήποτε επικεφαλή πλέγματος α που έχει αρχίσει την διαδικασία αντικατάστασης ή αυτή δηλώνεται στην διαδικασία cascading αντικατάστασης, βρίσκει ένα από τους γειτονικούς εφεδρικούς εμπιστευμένους κόμβους στο πλέγμα του, για παράδειγμα κόμβος β, για να κινηθεί σε εκείνο το γειτονικό κενό πλέγμα πριν αρχίσει ο επόμενος γύρος.

3. Εάν ένας τέτοιος κόμβος β δεν μπορεί να βρεθεί, επιλέξτε οποιοδήποτε γειτονικό πλέγμα εκτός από το κενό αλλά να είναι ακόμα στην περιοχή Δ. Κατόπιν, στείλετε την ανακοίνωση για την αντικατάσταση του α, που συνδέει τις πληροφορίες της Δ και μια τέτοια επιλογή. Θα επιλέξει πάντα έναν με εφεδρικό εμπιστευμένο κόμβο(ους) πρώτα. Μετά από αυτό, το α μετακινείται στο κενό πλέγμα πριν αρχίσει ο επόμενος γύρος.

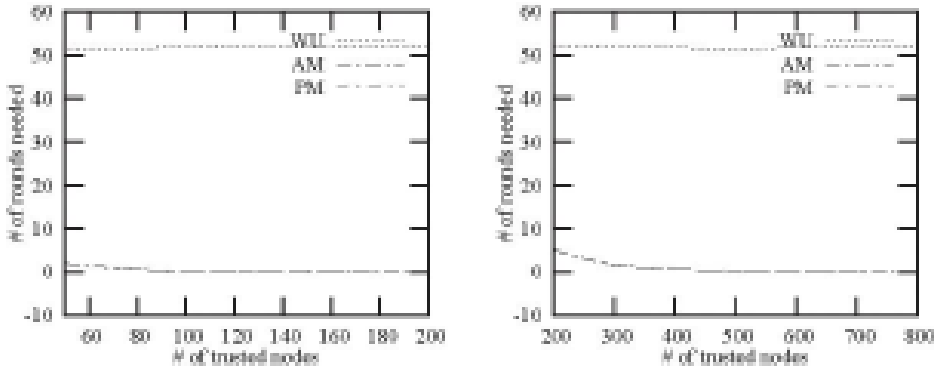
Αποτελέσματα προσομοίωσης

Τα αποτελέσματα μπορούν να συνοψιστούν ως εξής:

1. Αναφέρεται μέσα από ένα άρθρο [33] ότι μεταξύ όλων των υπαρχόντων βοηθημένων μετακίνησης ισορροπίας μεθόδων, το σχέδιο WU έχει την καλύτερη απόδοση όσον αφορά

στη διαδικασία που συγκλίνει. Όμως, αυτή η μέθοδος απαιτεί μια ανίχνευση πριν από τη μετακίνηση κόμβων. Εξετάζοντας τις δαπάνες επικοινωνίας στη διαδικασία ανίχνευσής της, ο αριθμός κύκλων που απαιτούνται, δηλ., η συγκλίνουσα ταχύτητα, είναι O(n). Μαζί τα σχέδια PM και AM χρησιμοποιούν την 1-hop γειτονιά, και θα συγκλίνουν πολύ γρηγορότερα από το σχέδιο WU. Επιπλέον, τα αποτελέσματά τους δείχνουν ότι δεν υπάρχει καμία μεγάλη διαφορά μεταξύ των σχεδίων PM και AM στη συγκλίνουσα ταχύτητα.

2. Λόγω της τοπικής ρύθμισης της διαδικασίας αντικατάστασης, οι μοιάζοντας φιδωτές διαδικασίες αντικατάστασης, σχέδια PM και AM, είναι πιο εξελικτικά από το σχέδιο WU, όπως φαίνεται στο πιο κάτω σχήμα.



Σχήμα 5: Γραφικές παραστάσεις αποτελεσμάτων [18]

3. Επίσης αναφέρεται μέσα από ένα άρθρο [33] ότι το σχέδιο WU έχει το χαμηλότερο κόστος σε σχέση με τη συνολική κινούμενη απόσταση και το συνολικό αριθμό των κινήσεων. Εντούτοις, απαιτεί μια ρύθμιση μέσα σε ολόκληρο το δίκτυο. Με τον περιορισμό της ρύθμισης μέσα σε μια τοπική περιοχή, ο αριθμός συνολικών κινήσεων και η αντίστοιχη συνολική κινούμενη απόσταση μπορούν να μειωθούν πολύ στη φιδωτή διαδικασία αντικατάστασής τους (και στα 2 σχέδια PM και AM). Η διαδικασία αντικατάστασης κινείται στο σχέδιο PM μόνο όταν απαιτείται. Έτσι με αυτό που συμβαίνει, το σχέδιο PM αναλαμβάνει το χαμηλότερο κόστος. Όμως, απαιτεί τη διαδικασία αντικατάστασης προκειμένου να κινηθεί ένας εφεδρικός κόμβος τόσο γρήγορα όσο η διάδοση επικοινωνίας. Το σχέδιο AM βρίσκει μια ανταλλαγή μεταξύ αυτού του σκληρού περιορισμού και του κόστους, με την επίδοση του στη συγκλίνουσα ταχύτητα να μένει ακόμα αποδεκτή. Επομένως, είναι πρακτικότερο.

Στο άρθρο [18], έχουν παρουσιάσει μια φιδωτή διαδικασία αντικατάστασης για να καλύψουν τις τρύπες επιτήρησης WSNs όπου όλοι οι αισθητήρες που επεκτείνονται σε ορισμένες περιοχές αντίληψης είναι εκτός λειτουργίας από τη συνεργασία. Κατά συνέπεια, η συνδετικότητα και η κάλυψη των WSNs μπορούν να εγγυηθούν και τέτοια δίκτυα γίνονται εφαρμόσιμα για τις εφαρμογές ασφάλειας, ακόμα και όταν αλλάζει δυναμικά η θέση εργασίας των κόμβων. Η εφαρμογή της κάτω από δύο διαφορετικά πρότυπα συζητήθηκε: ένα κάτω από το παθητικό πρότυπο και άλλο κάτω από το ενεργητικό πρότυπο. Στις μεθόδους τους, μόνο ενός hop γειτονιά χρησιμοποιείται και η ρύθμιση των κόμβων μπορεί να ελεγχθεί μέσα στην τοπική περιοχή. Τα πειραματικά αποτελέσματα παρουσιάζουν την προτεινόμενη μέθοδο να είναι εξελικτική και ότι

συγκλίνει γρήγορα με ένα ελαχιστοποιημένο κόστος. Σε μελλοντική δουλειά, η κατανάλωση ενέργειας θα εξεταστεί στην προσαρμογή κόμβων έτσι ώστε η διάρκεια ζωής της πλήρους κάλυψης να μπορεί να επεκταθεί.

Ο αλγόριθμος Scan-based Movement-Assisted sensoR deployment (SMART)

Η αποδοτικότητα των δικτύων αισθητήρων εξαρτάται από την κάλυψη της περιοχής ελέγχου. Αν και γενικά ένας ικανοποιητικός αριθμός αισθητήρων χρησιμοποιείται για να εξασφαλίσει έναν ορισμένο βαθμό πλεονασμού στην κάλυψη έτσι ώστε οι αισθητήρες μπορούν να περιστραφούν μεταξύ των ενεργών (active) και κοιμισμένων (sleep) σταδίων, ένας καλός αισθητήρας τοποθετημένος παραμένει απαραίτητος για να ισορροπήσει το φόρτο εργασίας των αισθητήρων. Σε ένα δίκτυο αισθητήρων με ευκολίες μετακίνησης, οι αισθητήρες μπορούν να μετακινηθούν γύρω για να αυτο-επεκταθούν. Η βοηθημένη μετακίνηση (movement-assisted) επέκτασης αισθητήρων εξετάζει τους κινούμενους αισθητήρες από ένα αρχικό μη ισορροπημένο στάδιο προς ένα ισορροπημένο στάδιο. Επομένως, τα διάφορα προβλήματα βελτιστοποίησης μπορούν να καθορίσουν την ελαχιστοποίηση των διαφόρων παραμέτρων, συμπεριλαμβανομένης της συνολικής κινούμενης απόστασης, του συνολικού αριθμού των κινήσεων, του κόστους επικοινωνίας / υπολογισμού, και του ποσοστού σύγκλισης. Στο έγγραφο [19], προτείνουν μια (Scan-based Movement-Assisted sensoR deploymenT) ανίχνευση βασισμένη μετακίνηση-βοηθημένη μέθοδο επέκτασης αισθητήρων (SMART) που χρησιμοποιεί την ανίχνευση και ανταλλαγή διάστασης για να επιτύχει ένα ισορροπημένο στάδιο.

Το SMART εξετάζει επίσης ένα μοναδικό πρόβλημα αποκαλούμενο τρύπες επικοινωνίας στα δίκτυα αισθητήρων. Χρησιμοποιώντας την έννοια της εξισορρόπησης φορτίων, το SMART επιτυγχάνει την καλή επίδοση ειδικά όταν εφαρμόζεται στα ανομοιόμορφα κατανεμημένα δίκτυα αισθητήρων, και μπορεί να είναι ένα συμπλήρωμα στις υπάρχουσες μεθόδους επέκτασης αισθητήρων. Η εκτενής προσομοίωση έχει γίνει για να ελέγξει την αποτελεσματικότητα του προτεινόμενου σχεδίου. Στο SMART, ένας δεδομένος ορθογώνιος τομέας αισθητήρων χωρίζεται αρχικά σε ένα 2-D πλέγμα μέσω της συγκέντρωσης (clustering). Κάθε συστάδα αντιστοιχεί σε μια τετραγωνική περιοχή και έχει ένα αρχηγό (clusterhead) που είναι υπεύθυνο για τη λογιστική (bookkeeping) και την επικοινωνία με παρακείμενα clusterheads. Η συγκέντρωση είναι μια ευρέως χρησιμοποιημένη προσέγγιση στα δίκτυα αισθητήρων για την υποστήριξή και για την απλοποίηση του σχεδιασμού. Έχει ήδη παρουσιαστεί σε άρθρο ότι η συγκέντρωση είναι η αποδοτικότερη για το δίκτυο αισθητήρων όπου τα δεδομένα διαβιβάζονται συνεχώς. Μια υβριδική προσέγγιση χρησιμοποιείται για την εξισορρόπηση φορτίων, όπου το 2-D πλέγμα χωρίζεται σε 1-D πίνακες από σειρά και από στήλη. Δύο ανιχνεύσεις χρησιμοποιούνται στη σειρά: μια για όλες τις σειρές, που ακολουθείται από μια άλλη για όλες τις στήλες. Μέσα σε κάθε σειρά και στήλη, η λειτουργία ανίχνευσης χρησιμοποιείται για να υπολογίσει το μέσο φορτίο και έπειτα για να καθορίσει το ποσό υπερφόρτωσης και υποφόρτωσης στις συστάδες. Το φορτίο μετατοπίζεται από τις υπερφορτωμένες συστάδες στις υποφορτωμένες συστάδες με έναν βέλτιστο τρόπο να επιτευχθεί ένα ισορροπημένο στάδιο. Βέλτιστο εννοούμε τον ελάχιστο αριθμό κινήσεων και ελάχιστη συνολική κινούμενη απόσταση. Με το ισορροπημένο στάδιο, αναφερόμαστε σε ένα στάδιο με το μέγιστο μέγεθος συστάδων (ο αριθμός αισθητήρων σε μια συστάδα) και το ελάχιστο μέγεθος συστάδων που είναι διαφορετικό από το πολύ 1.

Χρησιμοποίηση αυτής της 2-διαστάσεις ανίχνευσης χωρίς σφαιρικές πληροφορίες, κάθε αισθητήρας κινείται το πολύ-πολύ δύο φορές, αν και μπορεί να μην είναι συνολικά βέλτιστος από την άποψη της συνολικής κινούμενης απόστασης σε 2-D πλέγματα.

Το πρόβλημα τρυπών επικοινωνίας σε ένα 2-D πλέγμα αντιστοιχεί σε μια συστάδα με μέγεθος συστάδων μηδέν. Σαφώς, η προσέγγιση ανίχνευσης δεν μπορεί να χρησιμοποιηθεί σε μια σειρά ή μια στήλη με τις τρύπες, αφού οι clusterheads χωρισμένοι από μια ή περισσότερες τρύπες δεν μπορούν να επικοινωνήσουν ο ένας με τον άλλον για να εκτελέσουν μια λειτουργία ανίχνευσης. Η λύση τους [19] στο ζήτημα τρυπών είναι βασισμένη στη φύτευση ενός «seed» από μια μη κενή συστάδα σε μια παρακείμενη κενή συστάδα. Διάφορες λύσεις προτείνονται κατά τέτοιο τρόπο ώστε αυτή η διαδικασία seed-planting (επίσης αποκαλούμενη προεπεξεργασία) μπορεί εύκολα να ενσωματωθεί με την κανονική 2-D διαδικασία ανίχνευσης που επιτυγχάνει μια καλή ισορροπία των διάφορων στόχων.

Τα αποτελέσματα προσομοίωσης μπορούν να συνοψιστούν ως εξής:

1. Το SMART επιτυγχάνει ένα πιο ισορροπημένο στάδιο από τη διάχυση, την ανταλλαγή διάστασης και τις μεθόδους βασισμένες στο σχήμα Voronoi επέκτασης αισθητήρων μέσα στα άνισα επεκταμένα δίκτυα αισθητήρων.
2. Το SMART χρειάζεται λίγους κύκλους. Οι οποίοι είναι οριακοί από 8, για εξισορρόπηση φορτίων.
3. Το βελτιστοποιημένο SMART έχει τον ελάχιστο αριθμό κινήσεων σε σύγκριση με άλλους αλγορίθμους.

4. Το SMART μπορεί να είναι αποτελεσματικό όταν χρησιμοποιείται σχετικά σε πυκνά δίκτυα αισθητήρων ως συμπλήρωμα για τις υπάρχοντες μεθόδους επέκτασης αισθητήρων.
5. Όταν ο αριθμός επεκταμένων κόμβων είναι λιγότερος από $4n^2$, η απόδοση του SMART μειώνεται, αφού απαιτεί περισσότερους κύκλους και το ισορροπημένο τελικό στάδιο δεν μπορεί να επιτευχθεί.

Έχουν προτείνει [19] ένα (Scan-based Movement-Assisted sensoR deploymenT) ανίχνευσης βασισμένο μετακίνηση-βοηθημένο αλγόριθμο επέκτασης αισθητήρων, ο οποίος είναι μια υβριδική προσέγγιση των τοπικών και σφαιρικών μεθόδων. Έχουν εξετάσει ένα μοναδικό ζήτημα αποκαλούμενο τρύπα επικοινωνίας, όπου ορισμένες περιοχές αντίληψης δεν έχουν κανέναν επεκταμένο αισθητήρα. Μια μέθοδος με seed-planting διαδικασία έχει προταθεί για να κινήσει έναν αισθητήρα προς κάθε αποκαλυμμένη περιοχή πριν από τη διαδικασία ανίχνευσης. Τα αποτελέσματα προσομοίωσης δείχνουν ότι η προτεινόμενη μέθοδος μπορεί να επιτύχει ακόμη και την επέκταση των αισθητήρων με τις μέτριες δαπάνες. Μελλοντικά, θα αποδώσουν στην προσομοίωση βάθους στην κατανάλωση ενέργειας των αλγορίθμων επέκτασης αισθητήρων και θα σχεδιάσουν μερικούς "intra-cluster" ισορρόπησης αλγορίθμους για να επιτύχουν την εξισορρόπηση φορτίων υψηλής ανάλυσης.

Ο αλγόριθμος SPAN

Το άρθρο [22] παρουσιάζει το Span, μια τεχνική αποταμίευσης δύναμης για τα ειδικά ασύρματα δίκτυα multi-hop που μειώνει την κατανάλωση ενέργειας χωρίς σημαντικά να μικρύνει την

χωρητικότητα ή τη συνδετικότητα του δικτύου. Το Span στηρίζεται στην παρατήρηση ότι όταν έχει μια περιοχή ενός ασύρματου δικτύου μοιραζόμενου καναλιού (shared channel) μια ικανοποιητική πυκνότητα των κόμβων, μόνο ένας μικρός αριθμός από αυτούς χρειάζεται να είναι ενεργός οποιαδήποτε στιγμή για να προωθήσει τη κυκλοφορία για τις ενεργές συνδέσεις. Το Span είναι ένας κατανεμημένος, τυχαίος αλγόριθμος όπου οι κόμβοι λαμβάνουν τις τοπικές αποφάσεις σχετικά με εάν θα «κοιμηθούν», ή αν θα λάβουν μέρος στη προώθηση αποστολής ως συντονιστής. Κάθε κόμβος στηρίζει τις αποφάσεις του σχετικά με μια εκτίμηση του πόσοι από τους γείτονές του θα ωφεληθούν από το να μείνει αυτός ξύπνιος, και στο ποσό ενέργειας που είναι διαθέσιμο. Δίνουν έναν τυχαίο αλγόριθμο όπου οι συντονιστές περιστρέφονται με το χρόνο, που καταδεικνύει πώς οι εντοπισμένες αποφάσεις κόμβων οδηγούν σε σύνδεση, συντηρώντας τη χωρητικότητα σφαιρικής τοπολογίας.

Η βελτίωση στη διάρκεια ζωής συστημάτων λόγω του Span αυξάνεται καθώς η αναλογία της κατανάλωσης ενέργειας από μη απασχόληση σε ύπνο (idle-to-sleep) αυξάνεται, και αυξάνεται καθώς η πυκνότητα του δικτύου αυξάνεται. Οι προσομοιώσεις τους δείχνουν ότι με ένα πρακτικό ενεργειακό πρότυπο, η διάρκεια ζωής συστημάτων ενός δικτύου 802.11 στον τρόπο αποταμίευσης δύναμης (in power saving mode) με το Span είναι ένας παράγοντας του δύο, καλύτερο από ότι χωρίς αυτό. Το Span ενσωματώνει ωραία με το 802.11, όταν οργανώνεται από κοινού με τον τρόπο αποταμίευσης δύναμης 802.11. Επίσης το Span βελτιώνει τη καθυστέρηση επικοινωνίας, την χωρητικότητα, και τη διάρκεια ζωής συστημάτων.

Το Span προσαρμοστικά εκλέγει τους συντονιστές από όλους τους κόμβους στο δίκτυο, και τους περιστρέφει εγκαίρως. Οι συντονιστές του Span μένουν άγρυπνοι και εκτελούν multi-hop

δρομολόγηση πακέτων μέσα στο ειδικό δίκτυο, ενώ άλλοι κόμβοι παραμένουν σε power-saving mode και περιοδικά ελέγχουν εάν θα ξυπνήσουν και να γίνουν συντονιστές. Με το Span, κάθε κόμβος χρησιμοποιεί έναν τυχαίο backoff delay για να αποφασίσει εάν θα γίνει συντονιστής. Αυτή η καθυστέρηση είναι μια λειτουργία του αριθμού άλλων κόμβων στη γειτονιά που μπορεί να γεφυρωθεί χρησιμοποιώντας αυτόν τον κόμβο, και το ποσό ενέργειας που θα του έχει παραμείνει. Τα αποτελέσματά τους δείχνουν ότι το Span όχι μόνο συντηρεί τη συνδετικότητα δικτύου, συντηρεί επίσης την χωρητικότητα, μειώνει τη καθυστέρηση, και παρέχει τη σημαντική ενέργεια αποταμίευσης. Για παράδειγμα, για ένα πρακτικό πεδίο πυκνοτήτων κόμβων και ενός πρακτικού ενεργειακού προτύπου, οι προσομοιώσεις τους δείχνουν ότι η διάρκεια ζωής συστημάτων με το Span είναι περισσότερη από έναν παράγοντα δύο καλύτερο από ότι χωρίς αυτό.

Το ποσό ενέργειας αποταμίευσης του Span παρέχει αυξήσεις μόνο ελαφρώς καθώς η πυκνότητα αυξάνεται. Αυτό οφείλεται κατά ένα μεγάλο μέρος στο γεγονός ότι η τρέχουσα εφαρμογή του Span χρησιμοποιεί τα χαρακτηριστικά γνωρίσματα αποταμίευσης δύναμης 802.11, από τους κόμβους περιοδικά ξυπνούν και ακούνε τις διαφημίσεις κυκλοφορίας. Επίσης έδειξαν ότι αυτή η προσέγγιση μπορεί να είναι εξαιρετικά ακριβή. Αυτό επιτρέπει την έρευνα σχετικά με ένα πιο γερό/εύρωστο και αποδοτικό στρώμα MAC αποταμίευσης δύναμης, ένα που ελαχιστοποιεί το χρονικό διάστημα που κάθε κόμβος σε στάδιο αποταμίευσης δύναμης πρέπει να μείνει ενεργό.

Ο αλγόριθμος CCP & CCP+SPAN

Μια αποτελεσματική προσέγγιση [21] για τη διατήρηση της ενέργειας στα ασύρματα δίκτυα αισθητήρων σχεδιάζει τα διαστήματα ύπνου για τους εξωτερικούς κόμβους, ενώ οι υπόλοιποι κόμβοι μένουν ενεργοί για να παρέχουν συνεχή υπηρεσία. Για το δίκτυο αισθητήρων για να λειτουργήσει επιτυχώς, οι ενεργοί κόμβοι πρέπει να διατηρήσουν μαζί και την κάλυψη αντίληψης και τη συνδετικότητα δικτύου. Επιπλέον, το δίκτυο πρέπει να είναι σε θέση να διαμορφωθεί από μόνο του σε οποιοδήποτε εφικτό βαθμό κάλυψης και συνδετικότητας σε σειρά προκειμένου να υποστηρίξει διαφορετικές εφαρμογές και περιβάλλοντα με διαφορετικές απαιτήσεις. Το έγγραφο [21] παρουσιάζει το σχέδιο και την ανάλυση των νέων πρωτοκόλλων που μπορούν δυναμικά να διαμορφώσουν ένα δίκτυο για να επιτύχουν τους εγγυημένους βαθμούς κάλυψης και συνδετικότητας. Αυτή η εργασία [21] διαφέρει από τα υπάρχοντα πρωτόκολλα συντήρησης συνδετικότητας ή κάλυψης με διάφορους βασικούς τρόπους: 1) παρουσιάζουν ένα πρωτόκολλο διαμόρφωσης κάλυψης (CCP) που μπορεί να παρέχει τους διαφορετικούς βαθμούς κάλυψης που ζητούνται από τις εφαρμογές. Αυτή η ευελιξία επιτρέπει στο δίκτυο να αυτοδιαμορφωθεί για ένα ευρύ φάσμα των εφαρμογών και (των ενδεχομένως δυναμικών) περιβαλλόντων. 2) Παρέχουν μια γεωμετρική ανάλυση της σχέσης μεταξύ της κάλυψης και της συνδετικότητας. Αυτή η ανάλυση παράγει τις βασικές ιδέες για τη μεταχείριση της κάλυψης και της συνδετικότητας σε ένα ενοποιημένο πλαίσιο: αυτό είναι σε αιχμηρή αντίθεση με διάφορες υπάρχουσες προσεγγίσεις που εξετάζουν τα δύο προβλήματα ξεχωριστά. 3) Ενσωματώνουν το CCP με το SPAN για να παρέχουν μαζί εγγυημένα και κάλυψη και συνδετικότητα. Καταδεικνύουν την ικανότητα των πρωτοκόλλων τους για να παρέχουν τις

εγγυημένες διαμορφώσεις κάλυψης και συνδετικότητας, και μέσω της γεωμετρικής ανάλυσης και μέσω των εκτενών προσομοιώσεων.

Συνοπτικά, τα βασικά αποτελέσματα των πειραμάτων τους είναι τα ακόλουθα:

- ⇨ Αποδοτικότητα κάλυψης: Το CCP μπορεί να παρέχει την ενός-κάλυψη κρατώντας έναν σημαντικά μικρότερο αριθμό ενεργών κόμβων από το πρωτόκολλο Ottawa. Ο αριθμός ενεργών κόμβων παραμένει σταθερά όσον αφορά την πυκνότητα δικτύου για τον ίδιο ζητούμενο βαθμό κάλυψης.
- ⇨ Διαμόρφωση κάλυψης: Ο αλγόριθμος επιλεξιμότητας CCP μπορεί αποτελεσματικά να επιβάλει τους διαφορετικούς βαθμούς κάλυψης που διευκρινίζονται από την εφαρμογή. Ο αριθμός ενεργών κόμβων παραμένει ανάλογος προς το ζητούμενο βαθμό κάλυψης.
- ⇨ Ενσωματωμένη διαμόρφωση κάλυψης και συνδετικότητας: Όταν Rc/Rs >= 2, όλα τα πρωτόκολλα που υιοθετούν CCP αποδίδουν καλά από την άποψη της αναλογίας παράδοσης πακέτων, της κάλυψης, και του αριθμού ενεργών κόμβων. Όταν Rc/Rs < 2, CCP+SPAN-2Hop είναι το πιο αποτελεσματικότερο πρωτόκολλο που παρέχει και την ικανοποιητική κάλυψη και την επικοινωνία. Το SPAN δεν μπορεί να εγγυηθεί την κάλυψη κάτω από όλες τις δοκιμασμένες συνθήκες. Τα εμπειρικά αποτελέσματα ταιριάζουν με την γεωμετρική τους ανάλυση [21].

Αυτό το έγγραφο [21] ερευνά το πρόβλημα της διατήρησης της ενέργειας ενώ διατηρείται η επιθυμητή κάλυψη και συνδετικότητα στα ασύρματα δίκτυα αισθητήρων. Παρείχαν μια

γεωμετρική ανάλυση που 1) αποδεικνύει η κάλυψη αντίληψης υπονοεί τη συνδετικότητα δικτύων όταν το πεδίο αντίληψης δεν είναι περισσότερο από το μισό του πεδίου επικοινωνίας και 2) ποσολόγησαν τη σχέση μεταξύ του βαθμού κάλυψης και της συνδετικότητας. Ανέπτυξαν το πρωτόκολλο διαμόρφωσης κάλυψης (CCP) το οποίο μπορεί να επιτύχει τους διαφορετικούς βαθμούς κάλυψης που ζητούν οι εφαρμογές. Αυτή η ευελιξία επιτρέπει στο δίκτυο για να αυτό-διαμορφωθεί για ένα ευρύ φάσμα εφαρμογών και (ενδεχομένως δυναμικών) περιβαλλόντων. Ενσωματώνουν επίσης το CCP με το SPAN για να παρέχουν εγγυήσεις κάλυψης και συνδετικότητας όταν το πεδίο αντίληψης είναι υψηλότερο από το μισό πεδίο επικοινωνίας. Αποτελέσματα προσομοίωσης καταδεικνύουν ότι το CCP και το CCP+SPAN+2Hop μπορούν αποτελεσματικά να διαμορφώσουν το δίκτυο για να επιτύχουν και τους δύο μαζί ζητούμενους βαθμούς κάλυψης και ικανοποιητική χωρητικότητα επικοινωνίας κάτω από τις διαφορετικές αναλογίες αντίληψης/επικοινωνίας πεδίων όπως προβλέπεται από τη γεωμετρική τους ανάλυση. Στο μέλλον, θα επεκτείνουν τη λύση τους για να χειριστούν περιπλοκότερα πρότυπα κάλυψης και τη διαμόρφωση συνδετικότητας και να αναπτύξουν τον προσαρμοστικό επανασχηματισμό κάλυψης για ενεργειακά αποδοτικές κατανεμημένες τεχνικές ανίχνευσης και καταδίωξης. [21]

Ένα από τα θεμελιώδη ζητήματα στα δίκτυα αισθητήρων είναι το πρόβλημα κάλυψης, το οποίο απεικονίζει πόσο καλά ένα δίκτυο αισθητήρων ελέγχεται ή ακολουθείται από τους αισθητήρες. Στο έγγραφο [23], διατυπώνουν αυτό το πρόβλημα ως πρόβλημα απόφασης, του οποίου στόχος είναι να καθορίσει εάν κάθε σημείο στην περιοχή υπηρεσιών του δικτύου αισθητήρων καλύπτεται από τουλάχιστον κ αισθητήρες, όπου το κ είναι μια δεδομένη παράμετρος. Τα αισθανόμενα πεδία των αισθητήρων μπορούν να είναι δίσκοι μονάδων ή δίσκοι μη-μονάδων. Παρουσιάζουν τους πολυωνυμικού χρόνου αλγορίθμους, από την άποψη του αριθμού

αισθητήρων, οι οποίοι μπορούν να μεταφραστούν εύκολα στα κατανεμημένα πρωτόκολλα. Το αποτέλεσμα είναι μια γενίκευση μερικών προηγούμενων αποτελεσμάτων όπου υποθέτουν μόνο *κ=1*. Οι εφαρμογές του αποτελέσματος περιλαμβάνουν τον καθορισμό των ανεπαρκώς καλυμμένων περιοχών σε ένα δίκτυο αισθητήρων, την ενίσχυση της ανεκτικής σε λάθη χωρητικότητας στις εχθρικές περιοχές, και τη συντήρηση των ενεργειών των περιττών αισθητήρων σε ένα τυχαία επεκταμένο δίκτυο. Οι λύσεις τους μπορούν να μεταφραστούν εύκολα στα κατανεμημένα πρωτόκολλα για να λύσουν το πρόβλημα κάλυψης.

Οι αλγόριθμοι K-UC & K-NC

Μέσα από το άρθρο [23] προτείνονται πολυωνυμικού χρόνου αλγόριθμοι για να ελεγχθεί εάν κάθε σημείο στην περιοχή στόχων καλύπτεται από τουλάχιστον τον απαραίτητο αριθμό κόμβων. Ο αλγόριθμος K-UC υποθέτει μια ομοιόμορφη κυκλική αντίληψη δίσκων ενώ ο K-NC υποθέτει μια αισθανόμενη σειρά μη-δίσκων για κάθε κόμβο αισθητήρα. Ο προτεινόμενος αλγόριθμος απαιτεί μόνο πληροφορίες θέσης κάθε επεκταμένου κόμβου για να αξιολογήσει την επιθυμητή πολλαπλάσια κάλυψη. Για το K-UC, κάθε κόμβος υπολογίζει την κάλυψη του γείτονά του μόνο εάν ο γείτονας βρίσκεται μέσα δύο φορές στο αισθανόμενο πεδίο του κόμβου. Για το K-NC, οι διαφορετικοί κανόνες επιλογής γειτόνων καθορίζονται για τις περιπτώσεις όταν είναι ένας από τους κόμβους μέσα στο αισθανόμενο πεδίο άλλων κόμβων και όταν είναι ο ένας ή και οι δύο από τους κόμβους μέσα στην αντίληψη του πεδίου ο ένας τον άλλον. Ο κόμβος υπολογίζει έπειτα την κάλυψη περιμέτρου του με την εύρεση του τομέα της περιοχής κάλυψής του που καταλαμβάνεται από το αισθανόμενο πεδίο γειτόνων. Ο κόμβος έτσι ελέγχει εάν ολόκληρη η

περίμετρός 2π καλύπτεται από τους υπάρχοντες γείτονες στον απαραίτητο βαθμό ή όχι. Για να ανιχνεύσει την κάλυψη τρυπών, οι συντάκτες του άρθρου προτείνουν μια κεντρική οντότητα ελεγκτών που μπορεί να συλλέξει τις λεπτομέρειες των ανεπαρκώς καλυμμένων τμημάτων από κάθε τέτοιο κόμβο και μπορεί να αποστείλει νέο κόμβο για να καλύψει εκείνη την υπάρχουσα τρύπα. Εντούτοις, αυτή η συγκεντρωμένη προσέγγιση στερείται της εξελιξιμότητας.

Στο έγγραφο [23], έχουν προτείνει τις λύσεις σε δύο εκδόσεις του προβλήματος κάλυψης, δηλαδή K-UC και K-NC, σε ένα ασύρματο δίκτυο αισθητήρων. Διαμορφώνουν το πρόβλημα κάλυψης ως ένα πρόβλημα απόφασης, του οποίου ο στόχος είναι να καθοριστεί εάν κάθε θέση του στόχου περιοχής αντίληψης καλύπτεται αρκετά ή όχι. Παρά τον καθορισμό του επιπέδου κάλυψης κάθε θέσης, οι λύσεις τους είναι βασισμένες στον έλεγχο της περιμέτρου κάθε αισθανόμενου πεδίου αισθητήρα. Αν και το πρόβλημα φαίνεται να είναι πολύ δύσκολο με μια πρώτη ματιά, το σχέδιό τους μπορεί να δώσει μια ακριβή απάντηση σε χρόνο O(ndlogd). Με τις προτεινόμενες τεχνικές [23], συζητούν επίσης διάφορες εφαρμογές, τέτοιες σαν ανακάλυψη των ανεπαρκώς καλυμμένων περιοχών και διάσωση των ενεργειών, και των επεκτάσεων (όπως τα σενάρια με τα καυτά σημεία - hot spots - και ακανόνιστα αισθανόμενα πεδία) των αποτελεσμάτων τους. Ένα εργαλείο λογισμικού που εφαρμόζει τους προτεινόμενους αλγορίθμους είναι διαθέσιμο στο διαδίκτυο (http://hscc.csie.nctu.edu.tw/download/coverage.zip).

Ο αλγόριθμος Grid Scan

Το έγγραφο [25] περιγράφει ένα βασικό ζήτημα κάλυψης, και προτείνει ένα σχέδιο που ονομάζεται Grid Scan που εφαρμόζεται για να υπολογίσει το βασικό ποσοστό κάλυψης με την αυθαίρετη ακτίνα αντίληψης κάθε κόμβου. Με βάση την ανίχνευση πλέγματος (Grid Scan), μια προσέγγιση ανακατανομής προτείνεται για να συναντήσει οποιοδήποτε K-καλυμμένο ποσοστό σε κάποια περιοχή σύμφωνα με τις απαιτήσεις της εφαρμογής. Ο στόχος του σχεδίου ανακατανομής είναι να αποκτηθεί το ισοδύναμο ποσοστό κάλυψης που χρησιμοποιεί το λιγότερο αριθμό κόμβων αισθητήρων ή να επιτευχθεί το υψηλότερο ποσοστό κάλυψης με τον ίδιο αριθμό κόμβων αισθητήρων. Τα αποτελέσματα των πειραμάτων προσομοίωσης που έχουν γίνει υποστηρίζουν ότι η βασισμένη στην ανίχνευση ανακατανομή πλέγματος (Grid Scan based re-deployment) είναι αποτελεσματικότερη να καλύψει την ελεγχόμενη περιοχή από την τυχαία διάδοση (random spread).

Η προτεινόμενη προσέγγιση ανακατανομής πρέπει να ξέρει τη κάθε θέση κόμβου, η οποία το καθιστά μη εύκολο να εφαρμοστεί μέσα στα μεγάλης κλίμακας ασύρματα δίκτυα αισθητήρων. Στο μέλλον, μια κατανεμημένη προσέγγιση ανακατανομής χωρίς οποιεσδήποτε πληροφορίες θέσης θα αναπτυχθεί. Επιπλέον, θα πάρουν την ενεργειακή αποδοτικότητα και την ευρωστία στο μη κανονικό (irregular) radio μοντέλο ως άλλες δύο μετρικές για να αξιολογηθούν τα σχέδια / σχήματα.

Παρουσιάστηκε ένα διανεμημένο σχέδιο ελέγχου κάλυψης για τα συνεργαζόμενα κινητά δίκτυα αισθητήρων [26]. Το διάστημα αποστολής διαμορφώνεται χρησιμοποιώντας μια λειτουργία πυκνότητας που αντιπροσωπεύει τη συχνότητα των τυχαίων γεγονότων που πραγματοποιούνται,

με τους κινητούς αισθητήρες που λειτουργούν πέρα από ένα περιορισμένο πεδίο που καθορίζεται από ένα πιθανολογικό μοντέλο. Ένας gradient-based αλγόριθμος σχεδιάζεται επιθυμώντας τις τοπικές πληροφορίες σε κάθε αισθητήρα και μεγιστοποιώντας τις κοινές πιθανότητες ανίχνευσης των τυχαίων γεγονότων. Ενσωματώνουν επίσης τις δαπάνες επικοινωνίας στο πρόβλημα ελέγχου κάλυψης, που βλέπει το δίκτυο αισθητήρων ως δίκτυο πολλών πηγών, single-basestations συλλογής δεδομένων. Το κόστος επικοινωνίας διαμορφώνεται ως η κατανάλωση ισχύος που απαιτείται για να παραδώσει τα συλλεχθήσα στοιχεία από τους κόμβους αισθητήρων, κατά συνέπεια trading off της αισθανόμενης κάλυψης και του κόστους επικοινωνίας. Το σχήμα ελέγχου εξετάζεται σε ένα περιβάλλον προσομοίωσης για να επεξηγήσει τις προσαρμοστικές, διανεμημένες, και ασύγχρονες ιδιότητές του.

Το διάστημα αποστολής διαμορφώνεται χρησιμοποιώντας μια λειτουργία πυκνότητας που αντιπροσωπεύει τη συχνότητα της πραγματοποίησης τυχαίων γεγονότων. Υποθέτουμε ότι ένας κινητός ο αισθητήρας έχει ένα περιορισμένο πεδίο που καθορίζεται από ένα πιθανολογικό μοντέλο. Αυτός ο κατανεμημένος αλγόριθμος επέκτασης [26] έχει εξεταστεί εκτενώς σε ένα περιβάλλον προσομοίωσης. Τα πειραματικά αποτελέσματα δείχνουν ότι αυτό το σχέδιο είναι αποδοτικό και μπορεί να παραγάγει ένα σχήμα ποιοτικής επέκτασης. Επιπλέον, με την εφαρμογή των μεθόδων κλίσης και των γεωγραφικών τεχνικών δρομολόγησης, ο αλγόριθμος αποφεύγει τη λύση των σφαιρικών προβλημάτων βελτιστοποίησης, το οποίο εγγυάται στη συνέχεια την επίδοση σε πραγματικό χρόνο. Η μελλοντική εργασία περιλαμβάνει μια θεωρητική ανάλυση της σφαιρικής βελτιστοποίησης του “gradient-based” αλγορίθμου. Μπορεί να είναι απαραίτητο να

προσδιοριστούν οι ιδιότητες της λειτουργίας πυκνότητας γεγονότος και του μοντέλου αισθητήρα που εγγυώνται τέτοια σφαιρική βελτιστοποίηση. [26]

Ο αλγόριθμος COVEN

Η κάλυψη περιοχής είναι ένα από τα πιο θεμελιώδη προβλήματα στα ειδικά ασύρματα δίκτυα αισθητήρων επειδή είναι άμεσα συσχετιζόμενο με τη βελτιστοποίηση των πόρων σε έναν αισθανόμενο τομέα/πεδίο. Μεγιστοποίηση της περιοχής κάλυψης διατηρώντας ένα χαμηλότερο κόστος επέκτασης είναι πάντα μια πρόκληση, ειδικά όταν η περιοχή ελέγχου είναι άγνωστη και ενδεχομένως επικίνδυνη. Στο έγγραφο [27], παρουσιάζουν μια μέθοδο που ντετερμινιστικά υπολογίζετε το ακριβές ποσό τρυπών κάλυψης κάτω από τυχαία επέκταση που χρησιμοποιεί τα διαγράμματα Voronoi και χρησιμοποιεί στατικούς κόμβους για να συνεργαστούν και να υπολογίσουν τον αριθμό πρόσθετων κινητών κόμβων που χρειάζονται για να επεκταθούν και να επανεντοπιστούν στις βέλτιστες θέσεις για να μεγιστοποιηθεί η κάλυψη. Ακολουθούν μια διαδικασία επέκτασης με 2 βήματα σε ένα μικτό δίκτυο αισθητήρων και υποστηρίζουν με προσομοιώσεις και με ανάλυση ότι ο αλγόριθμος τους με συνεργαζόμενη ενίσχυση κάλυψης (COVEN) μπορεί να επιτύχει μια ανταλλαγή μεταξύ του κόστους της επέκτασης και του ποσοστού της περιοχής που καλύπτεται.

Notations:
s_i, m_j, R_s, $N(s_i)$, V_i are as defined in Table I. N_s: total number of static nodes deployed randomly
L_s: $\{\{id_{s_i}, loc_{s_i}\} \mid s_i \in N_s\}$, the list of ids and locations of the static nodes
L_m: $\{\{id_{s_i}, loc_{s_i}\}\}$, the list of ids and locations of the additional mobile nodes
$V_{coven(s_i)}$: the set of coverage-enhancing vertices for s_i, $V_{contention}$: the set of contention vertices
N_{m_i}: number of additional mobile nodes estimated by s_i
$A_{C(\Delta'_{ik})}$: hole lying inside part of Δ_{ik}. In Fig. 7, ΔAVP and ΔBVP are two such triangles and the holes within them are referred as $A_{C(\Delta'_{AB})}$ and $A_{C(\Delta'_{BA})}$ respectively. Note that, by construction $A_{C(\Delta'_{ik})} = A_{C(\Delta'_{ki})}$.
Initialization phase:
Choose a random static sensor as the *anchor* node, which starts the Coven algorithm by being the first member to calculate the hole within its Voronoi polygon and to estimate the number of additional mobile nodes required.
$V_{contention} \leftarrow \Phi$, Initialise the set of contention vertices as the NULL set.
At each static node s_i: (Coverage hole discovery and estimation phase)
$N_{m_i} \leftarrow 0$, $V_{coven(s_i)} \leftarrow \Phi$
Construct the Voronoi polygon $V(s_i)$ from the list L_s.
Calculate the amount of coverage hole $A_{C(\Delta_{ik})}$ existing inside each of the Voronoi triangles Δ_{ik}, $(k = 1, 2, ..., |N(s_i)|)$, that constitute its Voronoi polygon $V(s_i)$, by following the procedure described in section 4.
for $i = 0$ to $|N(s_i)|$ do
  if $V_i \notin V_{contention}$ AND $(V_{i+1 \bmod |N(s_i)|} \notin V_{coven(s_i)}$ OR $l_{ik} \geq R_s)$ then
    $A_{C(V_i)} \leftarrow 2 * (A_{C(\Delta'_{ik})} + A_{C(\Delta'_{ik+1})})$, $A_{C(\Delta'_{ik})}$ and $A_{C(\Delta'_{ik+1})}$ are calculated from $A_{C(\Delta_{ik})}$ and $A_{C(\Delta_{ik})}$.
    if $A_{C(V_i)} \geq \mu\pi R_s^2$ then
      Decide to place an additional mobile node m_i nearby vertex V_i.
      Calculate the optimal position for m_i at point p_i such that:
        1. p_i lies on the line that bisects the angle formed by V_i and the adjacent edges of $V(s_i)$
        2. p_i lies within $V(s_i)$, 3. $d(s_i, p_i) = \min\{2R_s, d(s_i, V_i)\}$
      $V_{coven(s_i)} \leftarrow V_{coven(s_i)} \cup V_i$, put the vertex V_i in set $V_{coven(s_i)}$
      Update $A_{C(\Delta_{ik})}$ and $A_{C(\Delta_{ik})}$ assuming a hypothetical sensor m_i is placed at p_i.
      if $d(V_i, p_i) \leq \epsilon R_s$ then
        $V_{contention} \leftarrow V_{contention} \cup V_i$, mark vertex V_i as a contention vertex.
      end if
      $N_{m_i} \leftarrow N_{m_i} + 1$
    end if
  end if
end for
if $V_{contention} \neq \Phi$ then
  Broadcast a message to $N(s_i)$ with the content of $V_{contention}$.
end if
A total $\sum_{i=1}^{N(s)}$ additional mobile nodes are randomly deployed.
At each static node s_i: (Mobile node placement phase)
Broadcast a message which contains all the $A_{C(V_i)}$'s and p_i's, in an attempt to discover N_{m_i} number of mobile sensors in its neighborhood, and a request to move the mobile nodes to locations p_i's.
At each mobile node m_i: (One-time movement phase)
Construct Voronoi polygon considering both the list L_s and L_m.
Calculate the area it is presently covering, call this A_{m_i}.
If not marked as *moved-once*, upon receiving a broadcast message from a static sensor, it finds out to which of the p_i's, it is closest.
if $A_{m_i} < A_{C(V_i)}$ then
  Move to point p_i.
  Mark itself as *moved-once*.
end if

Σχήμα 6: Λεπτομερής περιγραφή του αλγορίθμου COVEN [27]

Ο αλγόριθμος COVEN ακολουθεί μια διαδικασία επέκτασης με δύο στάδια, όπου αρχικά επεκτείνεται τυχαία οι στατικοί κόμβοι ανακαλύπτουν τις τρύπες μέσα στο αισθανόμενο πεδίο και υπολογίζουν έπειτα τον αριθμό των επιπρόσθετων κόμβων που απαιτείται και τις βέλτιστες θέσεις τους για να μεγιστοποιήσουν την κάλυψη. Τα αποτελέσματα προσομοίωσης [27] δείχνουν ότι ο αλγόριθμος συνεργασίας μπορεί να επιτύχει τη σημαντική βελτίωση απόδοσης πέρα από την τυχαία επέκταση. Αντί του περιορισμού του αριθμού κινητών κόμβων από μια συνάρτηση κόστους, αφήνουν τους στατικούς κόμβους να αποφασίζουν ανάλογα με τις θέσεις

τους και να υπολογίζουν τον αριθμό πρόσθετων κόμβων που χρειάζεται να επεκταθεί. Ο πρόσθετος αριθμός κινητών κόμβων είναι οριακός από τον αριθμό των «Voronoi vertices». Ως μελλοντική εργασία τους, θα επιθυμούσαν να υπολογίσουν το ακριβές ποσό τρυπών κάλυψης σε περίπτωση που οι κόμβοι έχουν διαφορετικά αισθανόμενα πεδία. Πιστεύουν ότι η συνεργασία στο δίκτυο αισθητήρων είναι ουσιαστικό για οποιοδήποτε κατανεμημένο αλγόριθμο για να αποδώσει αποτελεσματικά και εφαρμόζοντας το σε έναν μικτό δίκτυο αισθητήρων θα παρείχε πολλές σημαντικές προκλήσεις και άνοιγμα σε νέες ερευνητικές περιοχές. [27]

Ο αλγόριθμος Self-Deployment by density Control (SDDC)

Η κάλυψη αντίληψης είναι ένα σημαντικό ζήτημα στα ασύρματα κινητά δίκτυα αισθητήρων. Η στρατηγική για το πώς να επεκτείνεις τους κόμβους αισθητήρων σε ένα περιβάλλον, ειδικά μέσα σε άγνωστο μεγάλο περιβάλλον, θα έχει επιπτώσεις στη χρησιμότητα του δικτύου ακριβώς όπως την ποιότητα της επικοινωνίας. Παρουσιάζετε μέσα στο άρθρο [29] μια αποδοτική μέθοδος για την επέκταση αισθητήρων υποθέτοντας ότι οι σφαιρικές πληροφορίες δεν είναι διαθέσιμες. Ο αλγόριθμός που παρουσιάζεται (Self-Deployment by density Control, SDDC), χρησιμοποιεί έλεγχο πυκνότητας από κάθε κόμβο για να επεκτείνει τους κόμβους αισθητήρων ταυτόχρονα. Φτιάχνουν επίσης τους κόμβους για να δημιουργηθούν συστάδες για να επιτευχθεί ισορροπία πυκνότητας περιοχής. Τα χαρακτηριστικά του SDDC είναι η ταυτόχρονη κίνηση των πολυαισθητήρων, διανεμημένη λειτουργία, τοπικός υπολογισμός, και αυτοεπέκταση. Οι προσομοιώσεις παρουσιάζουν καλές επιδόσεις έναντι του επαυξητικού αυτοεπεκτανόμενου αλγορίθμου.

Οι 4 φάσεις που διαχωρίζεται ο έλεγχος της πυκνότητας:

1. Αρχικοποίηση (Initialization)

Κάθε κόμβος έχει δύο είδη θέσης. Μία είναι η θέση συστάδων και άλλη είναι η θέση κόμβων. Η θέση συστάδων που περιλαμβάνει «undecided», «cluster-head» και «member» χρησιμοποιείται για τη διαμόρφωση των συστάδων. Η θέση κόμβων που περιλαμβάνει «not-deployed» και «deployed» χρησιμοποιείται για την ένδειξη εάν πρέπει να προσπαθήσει να κινηθεί πάλι. Στη φάση αρχικοποίησης, ένας κόμβος αισθητήρων γίνεται ενεργός. Η θέση του κόμβου είναι «not-deployed» και η θέση συστάδων είναι «undecided». Θα ανιχνεύσει ότι τον περιβάλλει μέσα στο πεδίο επικοινωνίας του. Ένας κόμβος μεταδίδει μηνύματα «Hello» στους κόμβους γείτονες του. Τα μηνύματα «Hello» περιλαμβάνουν την ταυτότητα μηχανής και θέση συστάδων. Οι κόμβοι αισθητήρων χρησιμοποιούν τα μηνύματα «Hello» για να κατασκευάσουν τη συστάδα και να γεμίσουν τον πίνακα γειτόνων τους με τη θέση συστάδων.

2. Επιλογή στόχου (Goal selection)

Σε αυτήν την φάση, κάθε κόμβος αισθητήρας υπολογίζει την τοπική πυκνότητά του και υπολογίζει την επόμενη θέση. Όταν έχει τα στοιχεία της γωνίας και της απόστασης, μπορεί να υπολογίσει την πυκνότητα σε κάθε μικρότερη περιοχή επικοινωνίας.

3. Ψήφισμα στόχου (Goal resolution)

Για την αποφυγή δημιουργίας ενός τεμαχισμένο δίκτυο αισθητήρων, η μελλοντική θέση του κόμβου πρέπει να ελεγχθεί πριν κινηθεί. Ο κόμβος μεταδίδει την πιθανή επόμενη θέση του στους γείτονές του και περιμένει για ένα «ok» ή «suggested» μήνυμα. Εάν αυτός ο κόμβος δεν λαμβάνει οποιοδήποτε μήνυμα σε ένα χρονικό διάστημα λ, η θέση του παραμένει «not-deployed». Μετά από το χρονικό διάστημα λ, θα στείλει τα μηνύματα «ask» στη γειτονιά του σε κάθε χρονικό διάστημα τ έως ότου λάβει το μήνυμα. Η αξία του τ ορίζεται τυχαία μέσα στο διάστημα ενός χρόνου. Εάν ο αισθητήρας δεν μπορεί να πάει οπουδήποτε λόγω εμποδίου ή θα αφήσει το πεδίο επικοινωνίας ενός άλλου κόμβου ή η απόσταση μεταξύ της παρούσας και επόμενης θέσης είναι μικρότερη από ένα κατώτατο όριο ω, η θέση του αισθητήρα γίνεται επίσης «deployed». Αφότου επεκτείνονται όλοι οι αισθητήρες, ο αρχηγός της συστάδας (cluster-head) υπολογίζει τη μέση πυκνότητά μέσα στη συστάδα του και στέλνει αυτό το μήνυμα στους γειτονικούς του αρχηγούς συστάδων. Ένας γείτονας αρχηγός συστάδας πρέπει να είναι το μέγιστο τρία hops μακριά. Αφότου έχει λάβει την πυκνότητα της συστάδας του γείτονα, ο αρχηγός συστάδας θα την συγκρίνει με την τοπική μέση πυκνότητά του. Εάν μια πυκνότητα της ομάδας είναι μεγαλύτερη από άλλη από ένα κατώτατο όριο ε, η υψηλότερη πυκνότητα αρχηγού συστάδας θα διατάξει έναν κόμβο αισθητήρων μέλος που είναι ο πιο κοντινός στον αρχηγό συστάδας με τη χαμηλότερη πυκνότητα να κινηθεί προς αυτόν. Για να καθορίσουν ποιος κόμβος να κινηθεί, υπάρχουν δύο περιπτώσεις. Στη 1η περίπτωση, ο αρχηγός συστάδας λαμβάνει το μήνυμα πυκνότητας από τον κόμβο μέλος του. Κατόπιν ο κόμβος μέλος διατάζεται για να κινηθεί. Στη 2η περίπτωση, ο αρχηγός συστάδας λαμβάνει το μήνυμα από κόμβο μέλος ενός άλλου αρχηγού συστάδας και έτσι ο ίδιος ο αρχηγός συστάδας πρέπει να κινηθεί.

4. Εκτέλεση (Execution)

Οι αισθητήρες ρυθμίζουν τις θέσεις τους έως ότου έχουν επεκτείνει τη θέση και η διαφορά πυκνότητας οποιοδήποτε δύο αρχηγών συστάδας είναι μικρότερη από ένα κατώτατο όριο *ε*. Αυτός ο αλγόριθμος επαναλαμβάνει μέσω των φάσεων επιλογής, ψηφίσματος και εκτέλεσης. Θα ολοκληρώσει όταν έχουν όλοι οι κόμβοι θέση «deployed» και η διαφορά πυκνότητας οποιοδήποτε δύο αρχηγών συστάδων είναι μικρότερη από ένα κατώτατο όριο *ε*. Ο χρόνος σύγκλισης που επεκτείνεται συσχετίζεται με το κατώτατο όριο *ε*. Το μικρότερο *ε* είναι, πιο ομοιόμορφα αισθητήρες που διαδίδονται αλλά πιο μακροχρόνιος χρόνος να κινηθεί.

Συγκρίναμε τον αλγόριθμό αυτόν με τον ISD και LAMRC μέχρι το χρόνο που καλύφθηκε για την επέκταση και κάλυψη της περιοχής. Οι περιοχές κάλυψης είναι πλησιέστερες στο ISD και SDDC. Επειδή η τακτική επέκτασης της ISD είναι διαδοχική, ο περισσότερος αριθμός κόμβων είναι, ο περισσότερος χρόνος που χρειάζεται για την επέκταση. Ο χρόνος να ανταλλαχθούν οι πληροφορίες μεταξύ των κόμβων στο LAMRC κάνει το συνολικό χρόνο της κάλυψης πιο μεγάλο από το SDDC. Η ιδέα αυτού του εγγράφου είναι ότι οι κόμβοι αισθητήρων μπορούν να επεκταθούν μόνοι τους όπως γρήγορα εξαπλωμένα μικρά μόρια. Τα πειράματα καθιερώνουν σαφώς τη χρησιμότητα του αλγορίθμου SDDC. [29]

Το έγγραφο [30] εξετάζει το πρόβλημα επέκτασης ενός κινητού δικτύου αισθητήρων μέσα σε ένα άγνωστο περιβάλλον. Ένα κινητό δίκτυο αισθητήρων αποτελείται από μια διανεμημένη συλλογή κόμβων, κάθε ένας από τους οποίους έχει τις ικανότητες αντίληψης, υπολογισμού,

επικοινωνίας και μετακίνησης. Τέτοια δίκτυα είναι ικανά αυτοεπέκτασης δηλ., αρχίζοντας από κάποια συμπαγή αρχική διαμόρφωση, οι κόμβοι στο δίκτυο μπορούν να απλωθούν έτσι ώστε η καλυμμένη περιοχή από το δίκτυο να μεγιστοποιείται. Σε αυτό το έγγραφο, παρουσιάζετε μια "potential-field-based" προσέγγιση στην επέκταση. Οι τομείς κατασκευάζονται έτσι ώστε κάθε κόμβος αποκρούεται και από τα εμπόδια και από άλλους κόμβους, με αυτόν τον τρόπο αναγκάζοντας το δίκτυο για να διαδοθεί σε όλο το περιβάλλον. Η προσέγγιση είναι κατανεμημένη και εξελικτική.

Τα πειράματα που περιγράφονται στο έγγραφο [30] είναι, εντούτοις, αρκετά επαρκή για να καταδείξουν ότι μια πιθανή προσέγγιση τομέων (potential field) μπορεί να χρησιμοποιηθεί για να επεκτείνει τα κινητά δίκτυα αισθητήρων. Η προσέγγιση έχει το πλεονέκτημα ότι δεν απαιτεί κεντρικοποιημένο έλεγχο, εντοπισμό ή επικοινωνία, και επομένως θα επεκταθεί σε πολύ μεγάλα δίκτυα. Επιπλέον, όπως καταδεικνύεται μέσα στο άρθρο αυτό, αυτή η προσέγγιση έχει τα αποδείξιμα χαρακτηριστικά σύγκλισης. Υπάρχουν διάφορες κατευθύνσεις στις οποίες θα επιθυμούσαν να επεκτείνουμε αυτήν την έρευνα. Παραδείγματος χάριν, για το πώς κάποιος να εφαρμόσει αυτήν την προσέγγιση στα προβλήματα κάλυψης στα οποία η συνδετικότητα οπτικής επαφής (line-of-sight) είναι σημαντική. Για αυτά τα προβλήματα, θα επιθυμούσαν την επέκταση για να προχωρήσει έτσι ώστε το δίκτυο να συνδέεται πλήρως πάντα με τις σχέσεις οπτικής επαφής. Σε γενικές γραμμές, αυτό απαιτεί τη μορφή επικοινωνίας μεταξύ των κόμβων, στην πράξη, μπορεί να συμβεί εκείνη η συνδετικότητα, όπως η κάλυψη περιοχής, μπορεί να προκύψει από έναν συνδυασμό καθαρώς τοπικών κανόνων.

Η δυνατότητα ώθησης εισάγει πλήρως μιας νέας διάστασης σχέδιο στα ασύρματα ειδικά δίκτυα αισθητήρων, επιτρέποντας στο δίκτυο για να μετατρέψει προσαρμοστικά και να επισκευαστεί ως αντίδραση στην απρόβλεπτη δυναμική χρόνου εκτέλεσης. Ένα από το κλειδί των πόρων του δικτύου μέσα σε αυτά τα συστήματα είναι η ενέργεια και αρκετοί ανεξέλεγκτοι παράγοντες οδηγούν στις καταστάσεις όπου ένα ορισμένο τμήμα του δικτύου γίνεται περιορισμένο σε ενέργεια πριν από το υπόλοιπο δίκτυο. Η απόδοση γίνεται περιορισμένη λόγω περιορισμένων τμημάτων. Υποστηρίζετε ότι σε αυτό το σενάριο, αντί να καταστήσουν το πλήρες δίκτυο άχρηστο, οι υπόλοιποι πόροι ενέργειας πρέπει να αναδιοργανωθούν για να διαμορφώσουν μια νέα λειτουργική τοπολογία στο δίκτυο [31]. Έτσι παρουσιάζουν μεθόδους για το δίκτυο για να γνωρίζει την δική του ακεραιότητα και να χρησιμοποιήσει ώθηση «actuation» για να βελτιώσει την απόδοση όταν απαιτείται. Αυτή η ικανότητα του συστήματος αναφέρεται ως «self aware actuation».

Ο αλγόριθμος Coverage Fidelity Maintenance (Co-Fi)

Στο έγγραφο [31], εξετάζετε ένα δίκτυο όπου κόμβοι ή ένα υποσύνολο από τους κόμβους έχει τη δυνατότητα έλξης. Το δίκτυο χρησιμοποιεί την κινητικότητα για να επισκευάσει την απώλεια κάλυψης στην περιοχή που ελέγχεται από αυτό. Παρουσιάζετε ένας εντελώς διανεμημένος ενεργειακός ενήμερος αλγόριθμος (αναφερόμενος ως Co-Fi) για τη συντονισμένη κάλυψη συντήρησης στα δίκτυα αισθητήρων. Τα ενεργειακά γενικά έξοδα της κινητικότητας ενσωματώνονται στον αλγόριθμο, χωρίς να αφήνουν κατά συνέπεια καμία κρυμμένη δαπάνη. Η

προκαταρκτική ανάλυσή τους δείχνει ότι το Co-Fi μπορεί σημαντικά να βοηθήσει στη βελτίωση της χρησιμοποιήσιμης διάρκειας ζωής αυτών των δικτύων.

Αλγόριθμος Coverage Fidelity Maintenance

Ο Co-Fi λειτουργεί με τα ακόλουθα βήματα:

- Φάση έναρξης (Initialization): Κάθε κόμβος μαθαίνει για τους αισθανόμενους γείτονές του και υπολογίζει την περιοχή κάλυψής του.
- Φάση αιτήματος πανικού (Panic Request): Ένας κόμβος που θα πεθάνει στέλλει αίτημα για ανανέωση της τοπολογίας δικτύου.
- Φάση απάντησης πανικού (Panic Reply): Οι αισθανόμενοι γείτονες του κόμβου που θα πεθάνει ανταποκρίνονται σε αυτό το αίτημα.
- Φάση απόφασης (Decision): Η τοπολογία δικτύου ενημερώνεται αναλόγως.

Στο έγγραφο [31], υποστηρίζετε ότι η δυνατότητα ώθησης επιτρέπει στο δίκτυο αισθητήρων να μετατρέψει προσαρμοστικά και να επισκευαστεί από μόνο του σε σειρά για να βελτιωθεί η απόδοσή του. Με βάση αυτήν την έννοια αυτοδιοργάνωσης, αναπτύσσουν ένα σχέδιο για τη διατήρηση πιστής κάλυψης στα δίκτυα αισθητήρων που χρησιμοποιούν την κινητικότητα των κόμβων αισθητήρων. Όταν μερικά τμήματα του δικτύου γίνονται περιορισμένα από πόρους, αντί να θεωρηθεί το πλήρες δίκτυο άχρηστο, το Co-Fi κινεί ρητά τους κόμβους για να διαμορφώσει μιαν νέα αποδοτική τοπολογία στο δίκτυο. Το Co-Fi είναι ένας πλήρως διανεμημένος και εντοπισμένος αλγόριθμος. Δείχνουν μέσα από το άρθρο ότι ακόμα κι αν η κατανομή ενέργειας

είναι ομοιόμορφη στο δίκτυο, δηλαδή στη χειρότερη περίπτωση σενάριο για τον αλγόριθμο, η απόδοση του συστήματος βελτιώνεται από έναν παράγοντα 2-6 ανάλογα με την πυκνότητα του δικτύου. Η σχετική απόδοση από το Co-Fi παραμένει ανεπηρέαστη με την εισαγωγή εμποδίων ή διαφορετικών αισθανόμενων πεδίων μεταξύ των κόμβων αισθητήρων, δηλαδή κάποιες διαφορετικές συνθήκες στο μοντέλο κάλυψης ενός κόμβου. Πιστεύουν ότι έχουν ακουμπήσει μόλις την επιφάνεια της σφαίρα που αφορά το θέμα «self-aware actuation». Στο έγγραφο [31], χρησιμοποιήθηκε προσέγγιση της «self-aware actuation» ώθησης για τη συντήρηση κάλυψης μέσα σε δίκτυα αισθητήρων. Όμως, τα μελλοντικά συστήματα μπορούν να χρησιμοποιήσουν αυτή την προσέγγιση που βελτιώνεται σε άλλες πτυχές όπως η συνδετικότητα, το «sensor calibration» ή η ασφάλεια δεδομένων. Πιστεύουν ότι η «self-aware actuation» ώθηση παρουσιάζει μια πλήρως νέα διάσταση στο σχέδιο των ασύρματων δικτύων αισθητήρων και μπορεί να βοηθήσει στην επίλυση πολλών προβλημάτων για την πραγματοποίηση του μεγίστου των δυνατοτήτων συστημάτων που είναι περιορισμένα με βάση τους πόρους.

Στο έγγραφο [32], εξετάζεται ένα από τα θεμελιώδη προβλήματα των ασυρμάτων ειδικών δικτύων, η κάλυψη. Η κάλυψη γενικά, έχει να κάνει με την ποιότητα εξυπηρέτησης (επιτήρηση) που μπορεί να παρασχεθεί από ένα ιδιαίτερο δίκτυο αισθητήρων. Καθορίζεται αρχικά η κάλυψη ως πρόβλημα από διάφορες απόψεις συμπεριλαμβανομένης της ντετερμινιστικής, στατιστικής, χειρότερης και καλύτερης περίπτωσης, και παρουσιάζονται παραδείγματα σε κάθε περιοχή. Συνδυάζοντας την υπολογιστική γεωμετρία και τις γραφικές θεωρητικών τεχνικών, συγκεκριμένα τα διαγράμματα Voronoi και τους γραφικούς αλγόριθμους αναζήτησης, καθιερώνουν το πιο κύριο σημείο του εγγράφου – τον βέλτιστο πολυωνυμικό σε χρόνο

χειρότερο και μέσης περίπτωσης αλγόριθμο για τον υπολογισμό κάλυψης. Παρουσιάζουν επίσης τα περιεκτικά πειραματικά αποτελέσματα και συζητούν τις μελλοντικές ερευνητικές κατευθύνσεις σχετικές με την κάλυψη στα δίκτυα αισθητήρων.

Στα περισσότερα δίκτυα αισθητήρων, υπάρχουν δύο φαινομενικά αντιφατικά, όμως σχετιζόμενα όσον αφορά την κάλυψη: χειρότερη και καλύτερη περίπτωση κάλυψης. Στη χειρότερη περίπτωση κάλυψης, οι προσπάθειες γίνονται για ποσολογισμό της ποιότητας εξυπηρέτησης με την εύρεση των περιοχών χαμηλότερης αίσθησης από τους κόμβους αισθητήρες και την ανίχνευση παραβίασης περιοχών. Στην κάλυψη καλύτερης περίπτωσης, όπου βρίσκει τις περιοχές υψηλής αίσθησης από τους αισθητήρες και προσδιορίζει την καλύτερη υποστήριξη και περιοχές καθοδήγησης είναι αρχικής ανησυχίας. Ο κεντρικός στόχος τους είναι ο σχεδιασμός εύρωστων, αποδοτικών και εξελικτικών αλγορίθμων που θα χρησιμοποιηθούν στην ασύρματη ολοκλήρωση πολυαισθητήρων. Από την εννοιολογική και αλγοριθμική άποψη, η κύρια συμβολή είναι ευαπόδεικτα ένας βέλτιστος πολυωνυμικού χρόνου αλγόριθμος για την κάλυψη στα δίκτυα αισθητήρων. Συνδυάζουν υπάρχοντες υπολογιστικές τεχνικές και κατασκευάσματα γεωμετρίας όπως το διάγραμμα Voronoi, με τις γραφικές θεωρητικές αλγοριθμικές τεχνικές. Η χρήση του διαγράμματος Voronoi, αποτελεσματικά και χωρίς απώλεια βελτιστοποίησης, μετασχηματίζει το συνεχή γεωμετρικό πρόβλημα σε ένα διακριτό γραφικών πρόβλημα. Επιπλέον, επιτρέπει την άμεση εφαρμογή των τεχνικών αναζήτησης μέσα στην προκύπτουσα γραφικών αντιπροσώπευση. Μελετούν επίσης την ασυμπτωτική συμπεριφορά κάλυψης των τυχαίων ασύρματων ειδικών δικτύων.

Παρουσιάστηκαν διάφορες ερμηνείες και διατυπώσεις της κάλυψης στα ασύρματα ειδικά δίκτυα αισθητήρων. Ένας βέλτιστος πολυωνυμικός χρονικός αλγόριθμος που χρησιμοποιεί τη γραφική θεωρητική και τα υπολογιστικά κατασκευάσματα γεωμετρίας προτάθηκε [32] για την επίλυση της καλύτερης και της χειρότερης περίπτωσης καλύψεων. Τα πειραματικά αποτελέσματα παρουσιάζουν διάφορες εφαρμογές των θεωρητικών διατυπώσεων και των αλγορίθμων κάλυψης συγκεκριμένα για την επίλυση για μέγιστο μονοπάτι παραβίασης (Maximal Breach Path), μέγιστο μονοπάτι υποστήριξης (Maximal Support Path), επιπρόσθετα "heuristics" επέκτασης αισθητήρων για να βελτιώσουν την κάλυψη, και το στοχαστικό πεδίο κάλυψης.

Στο έγγραφο [33], μελετήθηκε το πρόβλημα μέγιστης διάρκειας ζωής κάλυψης στόχων (MTC), το οποίο είναι να μεγιστοποιήσει τη διάρκεια ζωής του δικτύου ενώ εγγυείται την πλήρη κάλυψη όλων των στόχων. Πολλοί κεντρικοποιημένοι αλγόριθμοι έχουν προταθεί για να λύσουν αυτό το πρόβλημα. Πολύ λίγες διανεμημένες εκδόσεις έχουν παρουσιαστεί επίσης αλλά καμία από αυτές δεν πετυχαίνει μια καλή αναλογία προσέγγισης. Σε αυτό το έγγραφο, παρουσιάζονται δύο O(log n) τοπικοί αλγόριθμοι. Συγκεκριμένα, περιορίζουν αρχικά το MTC πρόβλημα στο «domatic number» πρόβλημα μέσα σε κατευθυνόμενους γράφους. Αυτή η σχέση δείχνει ότι μια εφικτή λύση στο «domatic number» πρόβλημα είναι επίσης μια εφικτή λύση στο MTC πρόβλημα. Αποδεικνύουν έπειτα το χαμηλότερο και ανώτερο όριο αυτού του «domatic number». Με βάση αυτήν την απόδειξη, παρουσιάζουν δύο O(log n)-τοπικούς αλγόριθμους για τη λύση του MTC προβλήματος.

Επίσης μέσα στο έγγραφο [33], μελετήθηκε το πρόβλημα της κάλυψης με έναν στόχο την μεγιστοποίηση της διάρκειας ζωής των δικτύων, η οποία είναι ένα κρίσιμο θέμα στα ασύρματα δίκτυα αισθητήρων. Αφού περιόρισαν αυτό το πρόβλημα σε ένα «domatic» πρόβλημα, το οποίο είναι να βρει τον μέγιστο αριθμό των «disjoint dominating sets» στους κατευθυνόμενους γράφους. Ακόμη παρουσιάστηκαν δύο τοπικοί αλγόριθμοι, των οποίων η αναλογία προσέγγισης είναι μέσα σε έναν παράγοντα $O(\log n)$ όπου το n είναι ο αριθμός των κόμβων αισθητήρων. Μετασχημάτισαν τη σχέση κάλυψης μεταξύ των κόμβων αισθητήρων και των στόχων σε ένα κατευθυνόμενο γράφο, του οποίου οι κορυφές αποτελούνται από μόνο τους κόμβους αισθητήρων (κανένας στόχος σχετικός). Η κύρια ιδιότητα αυτού του κατευθυνόμενου γράφου G' είναι ότι κάθε «dominating set» του G' είναι μια λύση του προβλήματος της κάλυψης. Το «dominating set» καλύπτει εντελώς όλους τους στόχους. Χρησιμοποιώντας το πιθανολογικό επιχείρημα και ορίζοντας το χρώμα σχεδίου, οι τοπικοί αλγόριθμοι επιστρέφουν ένα σύνολο «dominating sets» που ενεργοποιούνται διαδοχικά για να παρατείνουν τη διάρκεια ζωής των δικτύων. Αφού οι προτεινόμενες λύσεις είναι τυχαιοποιημένες, ενδιαφέρονται για τη μελέτη του πώς να γίνουν "de-randomize" αυτοί οι αλγόριθμοι. Επιπλέον, λόγω της σχέσης μεταξύ του MTC προβλήματος και του προβλήματος MDS, προγραμματίζουν να ερευνήσουν περαιτέρω πώς να εφαρμοστούν λύσεις στα MTC προβλήματα στον έλεγχο της τοπολογίας και των προβλημάτων συγκέντρωσης με έναν στόχο τη μεγιστοποίηση της διάρκειας ζωής δικτύου.

Οι αλγόριθμοι “Distributed Self-Spreading Algorithm” (DSSA) & Intelligent Deployment and Clustering Algorithm (IDCA)

Θεωρείται η ενέργεια σε ένα ασύρματο δίκτυο αισθητήρων (WSN) ένας πολύτιμος πόρος. Η επέκταση των κινητών αισθητήρων σε ένα WSN είναι μια διαδικασία ενεργειακής κατανάλωσης και γι’αυτό πρέπει να σχεδιαστεί προσεκτικά. Μέσα στο έγγραφο [34], προτείνουν έναν ευφυή ενεργειακά αποδοτικό επεκτατικό αλγόριθμο, βασισμένο στις συστάδες ασύρματων δικτύων αισθητήρων από έναν συνδυασμό συνεργασίας δόμησης συστάδων και ένα peer-to-peer σχέδιο επέκτασης. Η απόδοση του αλγορίθμου αξιολογείται από την άποψη της κάλυψης, της ομοιομορφίας, και του χρόνου και της απόστασης που διανύονται μέχρι να συγκλίνει ο αλγόριθμος. Ο αλγόριθμός που παρουσιάζεται μέσα στο έγγραφο αυτό αποδεικνύεται ότι παρέχει άριστη επίδοση.

Ένας αλγόριθμος βασισμένος στο peer-to-peer mode, που καλείται κατανεμημένος αυτοδιαδιδόμενος αλγόριθμος “Distributed Self-Spreading Algorithm” (DSSA) εμπνέεται από την ισορροπία των μορίων, η οποία ελαχιστοποιεί τη μοριακή ηλεκτρονική ενέργεια και τη διαπυρηνική απέχθεια. Κάθε μόριο καθορίζει το χαμηλότερο ενεργειακό σημείο του με έναν διανεμημένο τρόπο και το διάστημα του αποτελέσματος του από τα άλλα μόρια είναι σχεδόν ίδιο. Το βέλτιστο διάστημα μεταξύ των αισθητήρων από την άποψη της κάλυψης μπορεί να βρεθεί με μια διαδικασία παρόμοια με την ισορροπία των μορίων. Αυτός ο διανεμημένος αλγόριθμος εκτελείτε σε κάθε κόμβο. Ο αλγόριθμος περιέχει τα πιο κάτω τέσσερα μέρη:

1. Initialization
2. Partial force calculation
3. Oscillation-check
4. Stability-check

Κάθε κόμβος αποφασίζει τον τρόπο του για να είναι είτε σε έναν τρόπο συγκέντρωσης (clustering mode) είτε σε "peer-to-peer mode" βασισμένο στην τοπική του πυκνότητα και. στο επίπεδο υπολειπόμενης ενέργειας με ένα διανεμημένο και προσαρμοστικό τρόπο. Αυτός ο αλγόριθμος καλείται Intelligent Deployment and Clustering Algorithm (IDCA). Αυτός ο διανεμημένος αλγόριθμος εκτελείται σε κάθε κόμβο και περιέχει επίσης τέσσερα μέρη όπως τον αλγόριθμο DSSA και είναι τα πιο κάτω:

1. Initialization
2. Mode determination and partialforce calculation
3. Oscillation-check
4. Stability-check

Και οι δύο οι αλγόριθμοι εκθέτουν μια παρόμοια απόδοση πέρα από τα διαφορετικά μεγέθη δικτύων. Η κάλυψη που επιτυγχάνεται από όλους τους αλγορίθμους αυξάνεται όσο το μέγεθος δικτύων ανεβαίνει. Δεδομένου ότι ο αριθμός κόμβων αυξάνεται, η βελτίωση στην κάλυψη μικραίνει. Τόσο ο αλγόριθμος DSSA όσο και ο αλγόριθμος IDCA επιτυγχάνουν καλύτερη ομοιομορφία από την αρχική και ο DSSA ξεπερνά τον IDCA ελαφρώς. Η βελτίωση στην

ομοιομορφία δεν είναι αυτή ευαίσθητη στην πυκνότητα δικτύου. Μέσα από τα αποτελέσματα των προσομοιώσεων βλέπουμε ότι ο IDCA οδηγεί στη γρηγορότερη επέκταση από τον DSSA σε έναν μέσο όρο. Επίσης οι χρόνοι λήξης του IDCA είναι σχεδόν ίδιοι πέρα από ένα ευρύ φάσμα του αριθμού κόμβων. Αυτό σημαίνει ότι ο IDCA είναι λιγότερο ευαίσθητος στον αριθμό κόμβων. Με τον αριθμό κόμβων σε αυτό το πεδίο, ο χρόνος που απαιτείται να συγκλίνει είναι σχεδόν ο ίδιος και η απόσταση ταξιδιού του IDCA είναι πολύ μικρότερη από του DSSA. Επειδή η παραλλαγή απαιτείται εγκαίρως για να συγκλίνει και η απόσταση ταξιδιού είναι μικρότερη πέρα από αυτό το πεδίο των πυκνοτήτων των κόμβων, είναι ευκολότερο να υπολογιστεί η απαραίτητη ενέργεια για την επέκταση.

Έχουν εξετάσει το πρόβλημα επέκτασης για τα κινητά ασύρματα δίκτυα αισθητήρων στο άρθρο αυτό. Μια περιοχή ενδιαφέροντος (ROI) πρέπει να καλυφθεί από αριθμό κόμβων με το περιορισμένο πεδίο αντίληψης και επικοινωνίας. Αρχίζουν με μια «τυχαία» διανομή των κόμβων. Αν και πολλά σενάρια υιοθετούν την τυχαία επέκταση λόγω των πρακτικών λόγων όπως το κόστος και ο χρόνος επέκτασης, η τυχαία επέκταση ίσως δεν παρέχει μια ομοιόμορφη διανομή που είναι επιθυμητή για μια πιο μακροχρόνια διάρκεια ζωής συστημάτων πάνω από την περιοχή ενδιαφέροντος. Σε αυτό το έγγραφο, έχουν προτείνει ένα ευφυή ενεργειακά αποδοτικό αλγόριθμο επέκτασης για βασισμένα σε συστάδες ασύρματα δίκτυα αισθητήρων. Μετά την εφαρμογή του αλγορίθμου αυτού η περιοχή ενδιαφέροντος καλύπτεται από ομοιόμορφα διανεμημένους κόμβους. Η απόδοση του αλγορίθμου καθορίζεται από το ποσοστό της περιοχής που καλύπτεται, τον χρόνο υπολογισμού / επέκτασης, τον μέσο όρο απόστασης που απαιτείται για την επέκταση, και την ομοιομορφία των δικτύων. Τα αποτελέσματα προσομοίωσης δείχνουν

ότι ο αλγόριθμός που παρουσιάστηκε μέσα από το άρθρο αυτό επιτυγχάνει μια ομοιόμορφη κατανομή από αρχικές ανομοιόμορφες κατανομές με ένα ενεργειακά αποδοτικό τρόπο.

Το πρόβλημα της κάλυψης χωρίς αρχικές σφαιρικές πληροφορίες για το περιβάλλον είναι ένα στοιχείο κλειδί του γενικού προβλήματος εξερεύνησης. Οι εφαρμογές ποικίλλουν από την εξερεύνηση της επιφάνειας του Άρη στην αστική περιοχή αναζήτησης και διάσωσης (USAR), όπου ούτε ένας χάρτης, ούτε ένα σύστημα παγκόσμιας πλοήγησης (GPS) δεν είναι διαθέσιμος. Προτείνουν μέσα από το άρθρο [35] δύο αλγορίθμους για το 2D πρόβλημα κάλυψης χρησιμοποιώντας πολλαπλά κινητά ρομπότ. Η βασική προϋπόθεση και των δύο αλγορίθμων είναι ότι η τοπική διασπορά είναι ένας φυσικός τρόπος να επιτευχθεί η σφαιρική κάλυψη. Κατά συνέπεια, και οι δύο αλγόριθμοι είναι βασισμένοι στην τοπική, αμοιβαία διασπορά αλληλεπίδρασης μεταξύ των ρομπότ όταν είναι μέσα στο πεδίο αντίληψης ο καθένας στον άλλον. Οι προσομοιώσεις έδειξαν ότι οι προτεινόμενοι αλγόριθμοι λύνουν το πρόβλημα μέσα σε 5-7% των (manually generated) βέλτιστων λύσεων. Έδειξαν ότι η φύση της αλληλεπίδρασης που απαιτείται μεταξύ των ρομπότ είναι πολύ απλή, πράγματι η ανώνυμη αλληλεπίδραση ξεπερνά ελαφρώς δηλαδή περισσότερο περίπλοκη τοπική τεχνική βασισμένη στον εφήμερο προσδιορισμό.

Η πρώτη προσέγγιση, που καλούμε πληροφοριακή (Informative), είναι βασισμένη στην ιδέα της ανάθεσης των τοπικών ταυτοτήτων στα ρομπότ όταν είναι μέσα στο πεδίο αισθητήρα ο ένας τον άλλον. Αυτή η προσέγγιση στηρίζεται στον εφήμερο προσδιορισμό όπου οι προσωρινές τοπικές ταυτότητες ορίζονται και οι αμοιβαίες σχετικές πληροφορίες θέσης ανταλλάσσονται μεταξύ των

αλληλεπιδρώντας ρομπότ, επιτρέποντας σε αυτά για να εξαπλωθούν έξω με έναν συντονισμένο τρόπο. Η δεύτερη προσέγγιση, αποκαλούμενη μοριακή (Molecular), είναι απλούστερη από την πρώτη. Τα ρομπότ δεν εκτελούν οποιαδήποτε κατευθυνόμενη επικοινωνία, και κανένας τοπικός προσδιορισμός δεν γίνεται. Κάθε ρομπότ επιλέγει μια κατεύθυνση «away» από όλους τους άμεσους αισθανόμενους γείτονές του και κινείται σε εκείνη την κατεύθυνση χωρίς επικοινωνία με τους γείτονές του. Και οι δύο αυτές προσεγγίσεις εξαρτώνται από τη δυνατότητα ενός ρομπότ να διακρίνει ένα άλλο ρομπότ από άλλα αντικείμενα στο περιβάλλον του. Μια τρίτη προσέγγιση (που καλείται βασική "Basic"), στην οποία δεν υπάρχει καμία αλληλεπίδραση «inter-robot» (εκτός από την αποφυγή εμποδίων), παρουσιάζεται επίσης και συγκρίνεται με τις δύο προτεινόμενες τεχνικές. Σε αυτήν την προσέγγιση τα ρομπότ δεν κάνουν καμία διάκριση μεταξύ των ρομπότ και άλλων αντικειμένων στο περιβάλλον. Και στις τρεις προσεγγίσεις η κίνηση κάθε ρομπότ καθοδηγείται από την αντιληπτή περιοχή κάλυψής του. Η μεγαλύτερη διαφορά είναι ότι η Πληροφοριακή (Informaitive) και η Μοριακή (Molecular) τεχνική εξετάζουν την αλληλεπίδραση μεταξύ των ρομπότ, ενώ η βασική (Basic) τεχνική είναι βασισμένη μόνο στη συγκεκριμένη μεγιστοποίηση κάλυψης. Οι προσομοιώσεις δείχνουν ότι η Πληροφοριακή (Informaitive) και η Μοριακή (Molecular) τεχνική λύνουν το πρόβλημα σε 5-7% των βέλτιστων λύσεων και ξεπερνούν σημαντικά τη βασική τεχνική. Δείχνουν ότι η φύση της αλληλεπίδρασης που απαιτείται μεταξύ των ρομπότ είναι πολύ απλή πράγματι η ανώνυμη αλληλεπίδραση (Molecular) ελαφρώς ξεπερνά τον εφήμερο προσδιορισμό (Informative).

Παρά τις διαφορές μεταξύ των δύο τεχνικών που παρουσιάζονται, τα αποτελέσματά τους είναι ποσοτικά παρόμοια. Και οι δύο προσεγγίσεις ξεπερνούν τη βασική προσέγγιση και οι δύο

διαποτίζουν γρήγορα σχεδόν στην πλήρη κάλυψη. Ακόμα κι αν η μοριακή προσέγγιση είναι απλούστερη, ξεπερνά ελαφρώς την πληροφοριακή προσέγγιση. Υποθέτουν ότι αυτό οφείλεται στα πρόσθετα γενικά έξοδα της διακοπής της Πληροφοριακή προσέγγιση για το σχηματισμό «coalition». Δεν είναι σαφές ότι ο εφήμερος προσδιορισμός βοηθά πραγματικά σε τέτοιες περιπτώσεις αν και αυτό επιτρέπει περαιτέρω έρευνα. Αυτό που είναι σαφώς προφανές είναι ότι η δυνατότητα των ρομπότ να επικοινωνήσουν μεταξύ τους χωρίς εμπόδια είναι κρίσιμη, και η μοριακή και πληροφοριακή προσεγγίση χρησιμοποιούν αυτό και ξεπερνά σημαντικά τη βασική προσέγγιση. Η μοριακή προσέγγιση αποδίδει καλύτερα από την άποψη του ISC μετρικού επίσης, παρά το γεγονός ότι το ISC μειώνεται με τον αυξανόμενο αριθμό ρομπότ. [35]

Η καταδίωξη των κινητών στόχων είναι μια σημαντική εφαρμογή των δικτύων αισθητήρων. Στο έγγραφο [36], το ζήτημα της καταδίωξης αντιμετωπίζεται αρχικά μέσω του προσδιορισμού μιας μειωμένης κάλυψης για την περιοχή ενδιαφέροντος. Οι αλγόριθμοι καταδίωξης αναπτύσσονται έπειτα χρησιμοποιώντας ένα μειωμένο σύνολο κόμβων αισθητήρων. Οι ανταλλαγές που περιλαμβάνονται στην ενεργειακά αποδοτική καταδίωξη του στόχου μελετούνται και η απόδοση των διανεμημένων αλγορίθμων καταδίωξης είναι συγκρινόμενη με πολύ γνωστές στρατηγικές από τη βιβλιογραφία. Αποδεικνύεται ότι το κέρδος στην ενέργεια αποταμίευσης έρχεται δαπάνη της μειωμένης ποιότητας της καταδίωξης. Οι αλγόριθμοι εγγυώνται την ευρωστία και την ακρίβεια της καταδίωξης καθώς επίσης και την επέκταση της γενικής διάρκειας ζωής συστημάτων. Οι αριθμητικές προσομοιώσεις παρουσιάζονται για να επικυρώσουν την απόδοση των προτεινόμενων αλγορίθμων.

Το πρόβλημα κάλυψης στα 3 διαστάσεων ασύρματα δίκτυα αισθητήρων (WSNs) διατυπώθηκε και αναλύθηκε. Οι αλγόριθμοι προτάθηκαν για να υπολογίσουν τον ελάχιστο αριθμό αισθητήρων που απαιτείται για την πλήρη κάλυψη μιας δεδομένης περιοχής. Επίσης προτάθηκε ένας διανεμημένος αλγόριθμος καταδίωξης που χρησιμοποιεί τα ασύρματα δίκτυα αισθητήρων. Τα θεωρητικά καθώς επίσης και πειραματικά αποτελέσματα αναπτύχθηκαν. Αντίθετα από προηγούμενη δουλειά σε αυτήν την περιοχή, οι αλγόριθμοί τους χρησιμοποιούν ένα ελάχιστο υποσύνολο των κόμβων αισθητήρων προκειμένου να ακολουθηθεί ένας στόχος που ελαχιστοποιεί τη γενική κατανάλωση ενέργειας και επομένως τη διάρκεια ζωής του δικτύου. Η εργασία τους σε αυτό το έγγραφο είναι μια συνέχεια της προηγούμενης δουλειάς τους όπου το τρισδιάστατο πρόβλημα ολικής κάλυψης και κάλυψης συνόρων ήταν αναλυμένο.

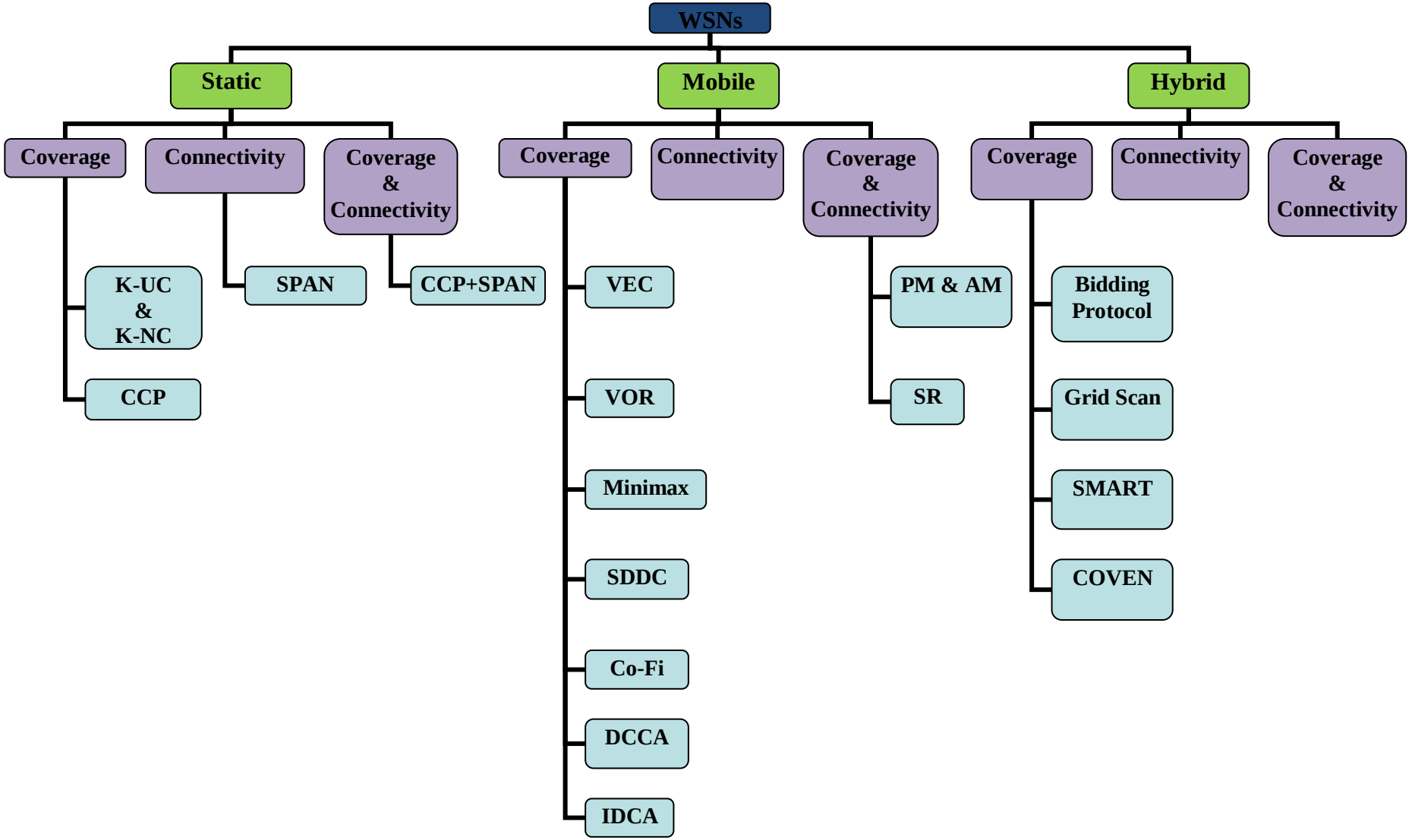
Κεφάλαιο 3

Κατηγοριοποίηση και Ανάλυση Αλγορίθμων που μελετήθηκαν

Μετά από τη μελέτη πολλών εργασιών, οι οποίες έχουν ήδη περιγραφεί στο προηγούμενο κεφάλαιο, συγκεντρώσαμε ένα πλήθος αλγορίθμων. Οι αλγόριθμοι αυτοί ασχολήθηκαν με κινητά ή στατικά ή υβριδικά δίκτυα αισθητήρων και είχαν κύριο στόχο τους να επιτύχουν τη πλήρη κάλυψη του δικτύου ή τη πλήρη συνδετικότητα του δικτύου ή και τα δύο μαζί. Παίρνοντας τον καθένα αλγόριθμο ξεχωριστά μπορούμε να διακρίνουμε κάποια χαρακτηριστικά τα οποία μπορούν να θεωρηθούν σημεία σύγκρισης μεταξύ αλγορίθμων που έχουν κοινό στόχο και εφαρμόζονται στον ίδιο τύπο δικτύου αισθητήρων. Έτσι συνοψίζοντας τα κύρια χαρακτηριστικά των αλγορίθμων που μελετήθηκαν δημιουργήσαμε τον πιο κάτω πίνακα για να μπορούμε να διακρίνουμε τις ομοιότητες και διαφορές τους.

3.1 Ταξινόμηση αλγορίθμων ελέγχου κάλυψης

Μετά την εκτενή μελέτη επιστημονικών άρθρων γύρω από το θέμα της κάλυψης σε ασύρματα δίκτυα αισθητήρων συγκεντρώσαμε μια πληθώρα από διαφορετικές τεχνικές και πρωτόκολλα αλγορίθμων κάλυψης ασύρματων δικτύων και τα έχουμε διαχωρίσει με βάση ορισμένα χαρακτηριστικά. Το σχήμα 7 παρουσιάζει τους αλγορίθμους ταξινομημένους με βάση την κατηγορία των δικτύων και τον κύριο στόχο τους και ο πίνακας 2 παρουσιάζει τις λεπτομέρειες και τα κυριότερα χαρακτηριστικά του κάθε αλγορίθμου.



Σχήμα 7: Σχηματική αναπαράσταση αλαγορίθμων ελέγχου κάλυψης ή/και συνδετικότητας

Διαφορετικές τεχνικές / πρωτόκολλα κάλυψης ασύρματων δικτύων αισθητήρων

		Τεχνική / πρωτόκολλο																
		VEC	VOR	Minimax	Bidding Protocol	SR	PM & AM	SMART	K-UC & K-NC	CCP	SPAN	CCP with SPAN	Grid Scan	COVEN	SDDC	Co-Fi	DSSA	IDCA
Κατηγορία δικτύου		Κινητά δίκτυα	Κινητά δίκτυα	Κινητά δίκτυα	Υβριδικά Δίκτυα	Κινητά δίκτυα	Κινητά δίκτυα	Υβριδικά Δίκτυα	Στατικά Δίκτυα	Στατικά Δίκτυα	Στατικά Δίκτυα	Στατικά Δίκτυα	Υβριδικά Δίκτυα	Υβριδικά Δίκτυα	Κινητά Δίκτυα	Υβριδικά Δίκτυα	Κινητά Δίκτυα	Κινητά Δίκτυα
	Χρειάζεται πληροφορία τοποθεσίας	√	√	√	√	√	√	√	√	√	X	√	X	√	√	√	√	√
	Single or multiple radius	single	single	single	single	single	single	single	Single/ multiple	single	single	single	single	single	single	Single/ multiple	single	Single
	Χρήση Υπολογιστικής Γεωμετρίας	√	√	√	√	X	X	X	X	X	X	X	X	√	X	X	X	X

		Τεχνικά/ πρωτόκολλα																
		VEC	VOR	Minimax	Bidding Protocol	SR	PM & AM	SMART	K-UC & K-NC	CCP	SPAN	CCP with SPAN	Grid Scan	COVEN	SDDC	Co-Fi	DSSA	IDCA
Απαιτήσεις/ Υποθέσεις/ χαρακτηριστικά	Νοητές Δυνάμεις	√	√	X	X	X	X	X	X	X	X	X	X	X	√	X	X	X
	Localized	√	√	√							√	√				√		
	Distributed	√	√	√	√	√	√	√	X	√	√	√	√	√	√	√	√	√
	Hamilton cycle	X	X	X	X	X	√	X	X	X	X	X	X	X	X	X	X	X
	Voronoi diagram	√	√	√	√	X	X	X	X	X	X	X	X	√	X	X	X	X
	Type of Network	Small/ Large	Small/ Large	Small/ Large	Small/ Large	Small/ Large	Small/ Large	Small/ Large	Small	Small/ Large	Small/ Large	Small/ Large	Small	Small/ Large	Small/ Large	Small/ Large	Small/ Large	Small/ Large
	Συγχρονισμός	X	X	X	√	√	√	X	√	√	√	√		√	√			
	Single or Multiple mobile sensor	Mult.	Mult.	Mult.	Mult.	Mult.	Mult.	Mult.						Mult.	Mult.	Mult.	Mult.	Mult.
	Reconfiguration	√	√	√	√	√	√	√						√	√	√		

		Τεχνική / πρωτόκολλα																
		VEC	VOR	Minimax	Bidding Protocol	SR	PM & AM	SMART	K-UC & K-NC	CCP	SPAN	CCP with SPAN	Grid Scan	COVEN	SDDC	Co-Fi	DSSA	IDCA
Coverage	Partial	√	√	√	√					√	√							
	Single																	
	Multiple					√	√		√			√						
Πλεονεκτήματα	Scalable	√	√	√	√	√	√	√	X				X			√		
	Incr. System Lifetime (Energy efficiency)					√					√	√				√	√	√

Πίνακας 2: Διαφορετικές τεχνικές/πρωτόκολλα ασύρματων δικτύων Αισθητήρων

3.2 Κριτήρια σύγκρισης

Πιο πάνω έγινε μια μικρή περιγραφή αλγορίθμων μέσα από διάφορα άρθρα, οι οποίοι παρουσιάστηκαν μέχρι σήμερα και σκοπό τους ήταν η επίλυση του προβλήματος της κάλυψης σε ασύρματα δίκτυα αισθητήρων. Αφού ειπώθηκαν κάποια βασικά χαρακτηριστικά τους στη λειτουργία και επίδοση τους μπορούμε να τους ταξινομήσουμε με βάση 2 κατηγορίες που μας ενδιαφέρουν έτσι ώστε στη συνέχεια να γίνει επιλογή αυτών που με βάση τον συνδυασμό διαφόρων χαρακτηριστικών μπορούν να επιτύχουν την καλύτερη επίδοση για την επίλυση του προβλήματος της κάλυψης για να υλοποιηθούν και να προσομοιωθούν ως συνέχεια της μελέτης αυτής. Οι κατηγορίες που θα χωρίσουμε τους αλγορίθμους και θα θεωρήσουμε μέσα από τη λειτουργία και επίδοση τους ότι μπορούν να επιτύχουν το στόχο τους με το καλύτερο δυνατό τρόπο είναι οι πιο κάτω:

1. **Coverage Maximization**: Αλγόριθμοι που ο σκοπός τους είναι η αύξηση της κάλυψης, δηλαδή η μείωση των περιοχών που δεν καλύπτονται από αρκετό αριθμό αισθητήρων και η ύπαρξη ομοιόμορφης τοποθέτησης αισθητήρων για την καλύτερη κάλυψη όλων των περιοχών ή τουλάχιστον των περισσοτέρων.

2. **Connectivity Maintenance**: Αλγόριθμοι που ο στόχος τους είναι να διατηρήσουν την καλύτερη δυνατή επικοινωνία μεταξύ των αισθητήρων, δηλαδή η ομοιόμορφη τοποθέτηση

3. των αισθητήρων έτσι ώστε να επιτυγχάνεται η καλύτερη δυνατή επικοινωνία ανάμεσα σε όλους τους αισθητήρες.

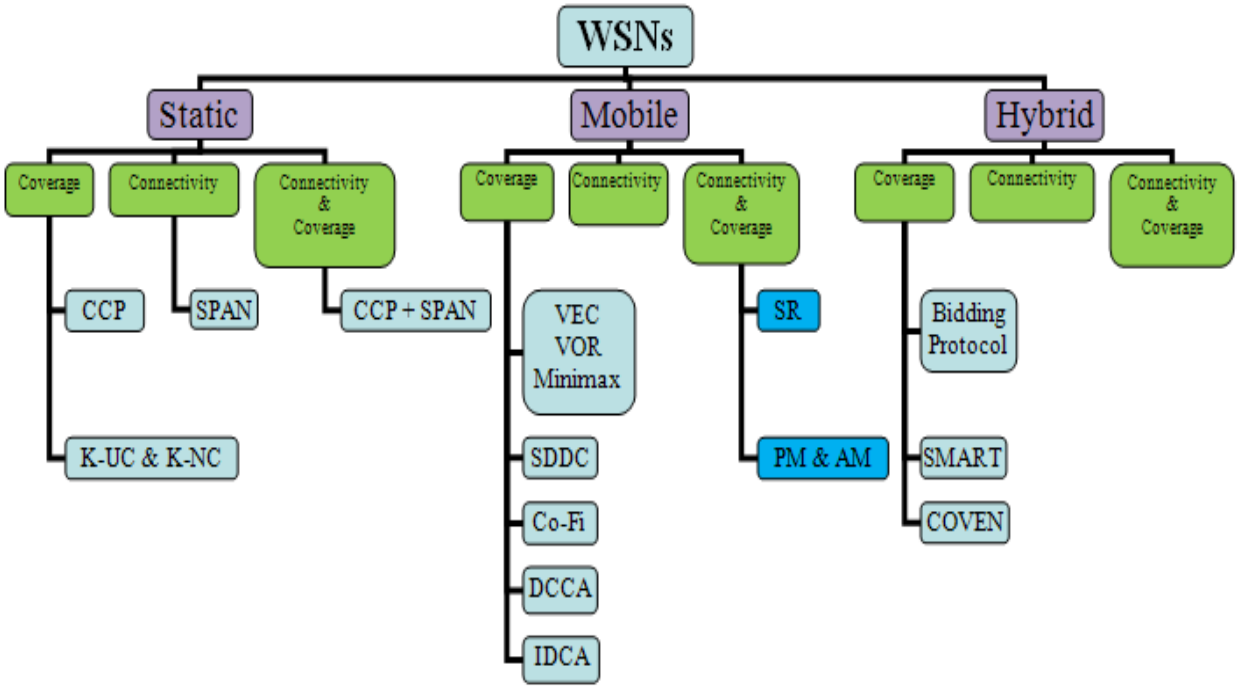
Υπάρχουν πάρα πολλά χαρακτηριστικά που διακρίνουμε μέσα από τους αλγορίθμους που μελετήθηκαν, έτσι παίρνοντας ένα συνδυασμό από αυτά ώστε να τοποθετηθεί ένα κριτήριο επιλογής τους για να προχωρήσουμε στην υλοποίηση των επιλεγμένων αλγορίθμων και την προσομοίωση τους για να εξάγουμε κάποια αποτελέσματα που θα είναι χρήσιμα στο εγγύς μέλλον σε αυτό τον ερευνητικό τομέα. Ο συνδυασμός των χαρακτηριστικών που θέσαμε είναι ο πιο κάτω:

Συνδυασμός χαρακτηριστικών

- **Energy Efficiency:** Χαρακτηρίζει την ενεργειακή αποδοτικότητα του αλγορίθμου, δηλαδή το ποσό ενέργειας που καταναλώνεται κατά την εκτέλεση του αλγορίθμου.
- **Convergence Time:** Χαρακτηρίζει το χρόνο ολοκλήρωσης των διαδικασιών για την πλήρη επίτευξη του στόχου του αλγορίθμου.
- **Distance of Movement:** Είναι η συνολική απόσταση μετακίνησης των κόμβων για την επιτυχή ολοκλήρωση του αλγορίθμου.

- **Communication Cost (Overhead):** Το κόστος επικοινωνίας, συγκεκριμένα των μηνυμάτων που ανταλλάσσονται μεταξύ των κώμβων κατά την εκτέλεση του αλγορίθμου.
- **Complexity:** Χαρακτηρίζει την πολυπλοκότητα του αλγορίθμου.
- **Distributed:** Εάν ο αλγόριθμος είναι κατανεμημένος δηλαδή μπορεί να εφαρμοστεί σε κατανεμημένο σύστημα.
- **Adaptability:** Το χαρακτηριστικό αυτό εκφράζει την προσαρμοστικότητα που έχει ο κάθε αλγόριθμος, για παράδειγμα στη διαδικασία του επανασχηματισμού (reconfiguration).

Στην πιο κάτω σχηματική αναπαράσταση διαχωρίζονται μερικοί αλγόριθμοι, που πληρούν τον πιο πάνω συνδυασμό χαρακτηριστικών και μελετήθηκαν μέσα από τα επιστημονικά άρθρα, με βάση το τύπο δικτύου αισθητήρων που εφαρμόζονται και το σκοπό που επιτυγχάνουν.



Σχήμα 8: Διαχωρισμός μερικών αλγορίθμων κάλυψης ή/και συνδετικότητας στα ασύρματα δίκτυα αισθητήρων

3.2 Αλγόριθμοι επιλογής για υλοποίηση με το λογισμικό Matlab

Μετά από την μελέτη αρκετών αλγορίθμων μέσα από επιστημονικά άρθρα επιλέξαμε 2 αλγορίθμους για να τους υλοποιήσουμε με τη βοήθεια του λογισμικού Matlab και μέσα από προσομοιώσεις να γίνει η σύγκριση τους. Οι αλγόριθμοι που επιλέξαμε είναι 1) Ενεργό Μοντέλο (AM) και 2) Snake-like cascading Replacement process (SR) και εφαρμόζονται σε κινητά δίκτυα με στόχο να επιτύχουν την πλήρη κάλυψη και συνδετικότητα του δικτύου.

3.2.1 Ενεργό μοντέλο - Active Model (AM)

Ο Αλγόριθμος αυτός επιλέχτηκε γιατί εγγυάται την κάλυψη και την συνδετικότητα στα κινητά δίκτυα με ένα γρήγορο και αποδοτικό τρόπο. Είναι ένα μοντέλο το οποίο μπορεί να υλοποιηθεί με τη βοήθεια του λογισμικού Matlab και να μελετηθεί και πειραματικά σε σύγκριση με κάποιο άλλο μοντέλο που μπορεί να εφαρμοστεί σε κινητά δίκτυα και αυτό με τη σειρά του να μπορεί να εγγυηθεί τόσο την κάλυψη όσο και την συνδετικότητα.

Ο αλγόριθμος του Ενεργού Μοντέλου βασίζεται στα πιο κάτω στάδια:

Στάδιο 1^{ο}

Για οποιοδήποτε επικεφαλή πλέγματος α που ανιχνεύει ένα κενό γειτονικό πλέγμα, εάν καμία ανακοίνωση δεν παραλαμβάνεται στον προηγούμενο κύκλο από εκείνη την περιοχή (για να αποφύγει το overreaction), μια διαδικασία αντικατάστασης κινείται αφότου καθορίζεται η αντίστοιχη περιοχή.

Στάδιο 2^{ο}

Για οποιοδήποτε επικεφαλή πλέγματος α που έχει αρχίσει την διαδικασία αντικατάστασης ή αυτή δηλώνεται στην διαδικασία cascading αντικατάστασης, βρίσκει ένα από τους γειτονικούς εφεδρικούς εμπιστευμένους κόμβους στο πλέγμα του, για παράδειγμα κόμβος β, για να κινηθεί σε εκείνο το γειτονικό κενό πλέγμα πριν αρχίσει ο επόμενος γύρος.

Στάδιο 3^{ο}

Εάν ένας τέτοιος κόμβος β δεν μπορεί να βρεθεί, επιλέξτε οποιοδήποτε γειτονικό πλέγμα εκτός από το κενό αλλά να είναι ακόμα στην περιοχή. Κατόπιν, στείλετε την ανακοίνωση για την αντικατάσταση του α, που συνδέει τις πληροφορίες της περιοχής και μια τέτοια επιλογή. Θα σταλεί η ανακοίνωση σε έναν από τους γειτονικούς του επικεφαλής πλέγματος και επιλέγεται πάντα αυτός με τους περισσότερους εφεδρικούς εμπιστευμένους κόμβους. Μετά από αυτό, το α μετακινείται στο κενό πλέγμα πριν αρχίσει ο επόμενος γύρος.

3.2.2 Snake-like cascading Replacement process (SR)

Ο Αλγόριθμος αυτός επιλέχτηκε γιατί εγγυάται την κάλυψη και την συνδετικότητα στα κινητά δίκτυα με ένα γρήγορο και αποδοτικό τρόπο, χρησιμοποιώντας την τεχνική του κατευθυνόμενου κύκλου του Χάμιλτον (“Hamilton cycle”) που δεν την είχαμε δει στους προηγούμενους αλγορίθμους που μελετήσαμε. Είναι ένα μοντέλο το οποίο μπορεί να υλοποιηθεί προγραμματιστικά με τη βοήθεια του λογισμικού Matlab και να μελετηθεί και πειραματικά σε σύγκριση με κάποιο άλλο μοντέλο που μπορεί να εφαρμοστεί σε κινητά δίκτυα και αυτό με τη σειρά του να μπορεί να εγγυηθεί τόσο την κάλυψη όσο και την συνδετικότητα, αλλά να διαφέρουν στον τρόπο που το επιτυγχάνουν αυτό μέσα από την τεχνική που χρησιμοποιούν.

Περιγραφή του μοντέλου

Στάδιο 1^{ο}

Στον επικεφαλή (head) u, η ακόλουθη διαδικασία αντικατάστασης θα αρχικοποιηθεί όταν δεν μπορεί να βρει το u τον επικεφαλή στο πλέγμα διαδόχων κατά μήκος του κατευθυνόμενου κύκλου του Χάμιλτον δηλ., ένα κενό πλέγμα σε μια τέτοια κατεύθυνση ανιχνεύεται.

Στάδιο 2^{ο}

Βρείτε έναν εφεδρικό κόμβο στο πλέγμα του u, κόμβος β, για να κινηθεί σε εκείνη την κενή περιοχή προτού να αρχίσει ο επόμενος κύκλος.

Στάδιο 3^{ο}

Εάν το πιο πάνω βήμα αποτυγχάνει, επαναλάβετε τα ακόλουθα βήματα έως ότου μπορεί ο δηλωμένος κόμβος u να βρει έναν εφεδρικό κόμβο στο πιο πάνω βήμα: (α) Στείλτε την ανακοίνωση στο προηγούμενο πλέγμα που ρωτά για μια αντικατάσταση για το u το ίδιο. (β) Περιμένετε έως ότου λαμβάνει ο αντίστοιχος επικεφαλής w αυτήν την ανακοίνωση. (γ) Μετακίνησε το u στο κενό πλέγμα πριν από τον επόμενο κύκλο που αρχίζει, δηλ., αφήνοντας το τρέχον πλέγμα κενό για την cascading αντικατάσταση.

3.3 Σύγκριση AM με SR μέσω διαφόρων προσομοιώσεων

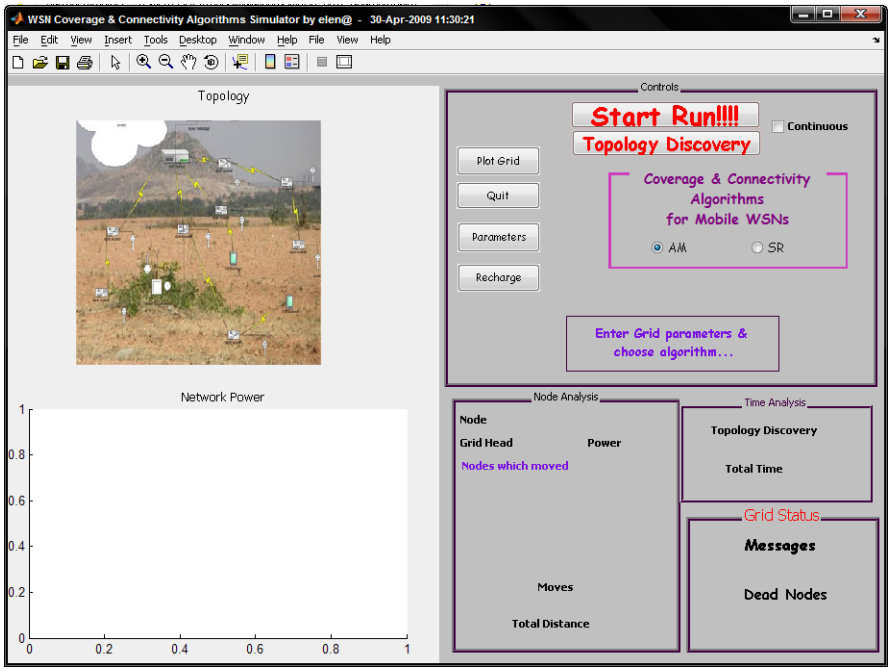
Οι πιο πάνω αλγόριθμοι υλοποιήθηκαν μέσω του λογισμικού Matlab και έτρεξαν για διαφορετικές τοπολογίες και πλήθος κινητών αισθητήρων. Επίσης η αρχική τοποθέτηση/τοπολογία των κινητών αισθητήρων ήταν τυχαία εκτός από τους αρχηγούς του κάθε πλαισίου (grid) όπου τοποθετούνταν στο κέντρο του κάθε πλαισίου (grid) στα αρχικά πειράματα ενώ στην συνέχεια έγιναν κάποιες αλλαγές στην επιλογή της μετακίνησης των αρχηγών κάθε πλαισίου και έτσι δημιουργήθηκαν τρεις διαφορετικές περιπτώσεις για κάθε αλγόριθμο, AM και SR, και με βάση τα πειραματικά αποτελέσματα που πήραμε έγιναν στην συνέχεια συγκρίσεις μεταξύ τους.

1η αλλαγή

Η αρχική τοποθέτηση όλων των κινητών αισθητήρων να είναι τυχαία και στη συνέχεια να μετακινείται ένας τυχαίος κινητός αισθητήρας από κάθε πλαίσιο στην κεντρική θέση του πλαισίου ως αρχηγός.

2η αλλαγή

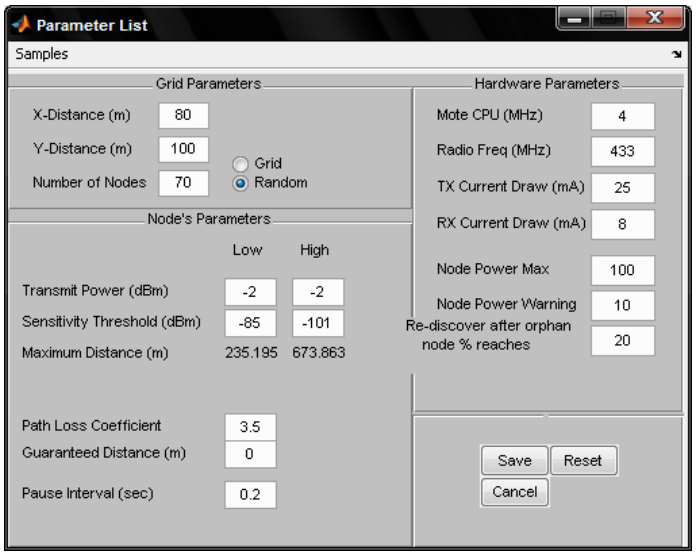
Η αρχική τοποθέτηση όλων των κινητών αισθητήρων να είναι τυχαία και στη συνέχεια να μετακινείται ο πιο κοντινός κινητός αισθητήρας από κάθε πλαίσιο στην κεντρική θέση του κάθε πλαισίου ως αρχηγός.



Σχήμα 9: Η οθόνη λειτουργίας των αλγορίθμων που υλοποιήθηκαν

3.3.1 Αποτελέσματα πειραμάτων

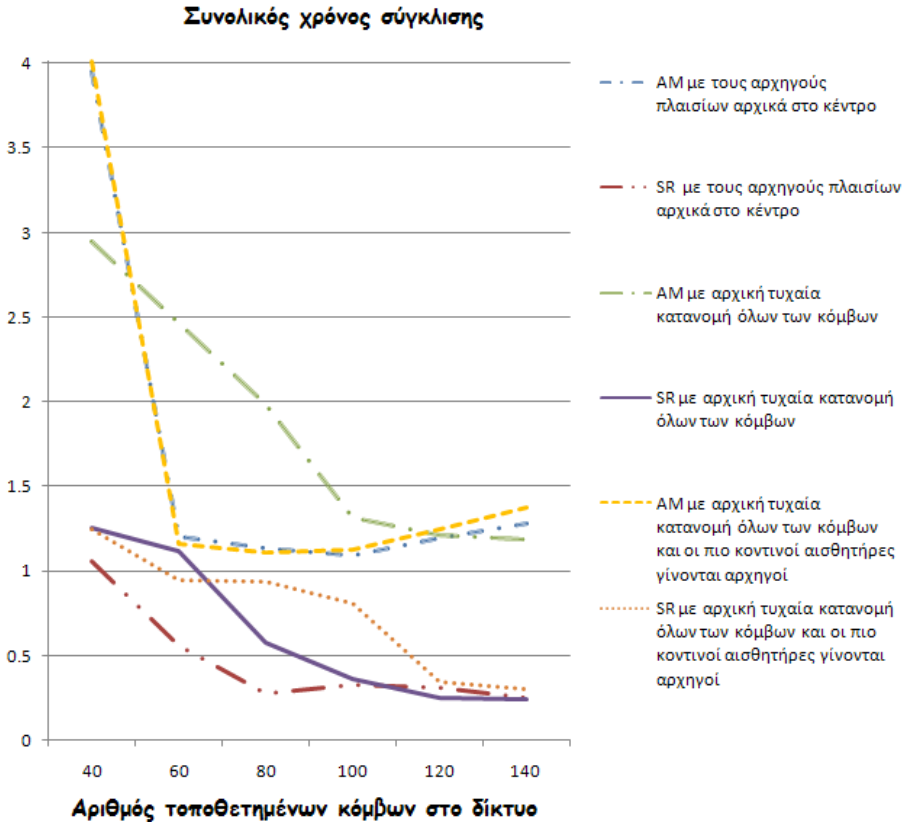
Εκτελέστηκαν διάφορες προσομοιώσεις μέσω του λογισμικού Matlab. Στα πιο κάτω πειράματα χρησιμοποιήθηκε τοπολογία 80x100 μέτρα και τυχαία τοποθετημένοι κινητοί αισθητήρες με το κάθε πλαίσιο (Grid) να έχει διαστάσεις 20x20 μέτρα. Πιο κάτω, στο σχήμα 10, ακολουθεί η λίστα παραμέτρων που χρησιμοποιήθηκε.



Σχήμα 10: Η λίστα των παραμέτρων που χρησιμοποιεί το πρόγραμμα με τους αλγορίθμους που υλοποιήθηκε στη Matlab

Η πιο κάτω γραφική παράσταση 1 παρουσιάζει τους χρόνους σύγκλισης για τους αλγορίθμους AM και SR κατά την αρχική τοποθέτηση των αρχηγών κάθε πλαισίου στο κέντρο του κάθε πλαισίου, για τους αλγορίθμους AM και SR κατά την αρχική τοποθέτηση όλων των αισθητήρων τυχαία και μετά την μετακίνηση ενός τυχαίου αισθητήρα μέσα σε κάθε πλαίσιο ως αρχηγός κάθε πλαισίου στο κέντρο του κάθε πλαισίου, και για τους αλγορίθμους AM και SR κατά την αρχική τοποθέτηση όλων των αισθητήρων τυχαία και μετά την μετακίνηση του πιο κοντινού αισθητήρα μέσα σε κάθε πλαίσιο ως αρχηγός κάθε πλαισίου στο κέντρο του κάθε πλαισίου. Παρατηρούμε αρχικά ότι οι χρόνοι σύγκλισης του SR είναι μικρότεροι σε σύγκριση με αυτούς του AM που βρίσκονται σε ψηλότερα επίπεδα. Υπάρχει και μια ελαφριά διαφορά μεταξύ των AM και SR με

την αρχική τοποθέτηση των αρχηγών κάθε πλαισίου στο κέντρο του κάθε πλαισίου σε σχέση με τους AM και SR με την αρχική τοποθέτηση όλων των αισθητήρων τυχαία και μετά την μετακίνηση των αρχηγών κάθε πλαισίου στο κέντρο του κάθε πλαισίου με βάση τον αισθητήρα με την πιο κοντινή απόσταση ή τυχαία, η οποία οφείλεται στον επιπρόσθετο χρόνο υπολογισμού για την μετακίνηση των τυχαία τοποθετημένων αισθητήρων που είναι πιο κοντά στο κέντρο του κάθε πλαισίου για να γίνουν οι αρχηγοί του καθενός πλαισίου αντίστοιχα.

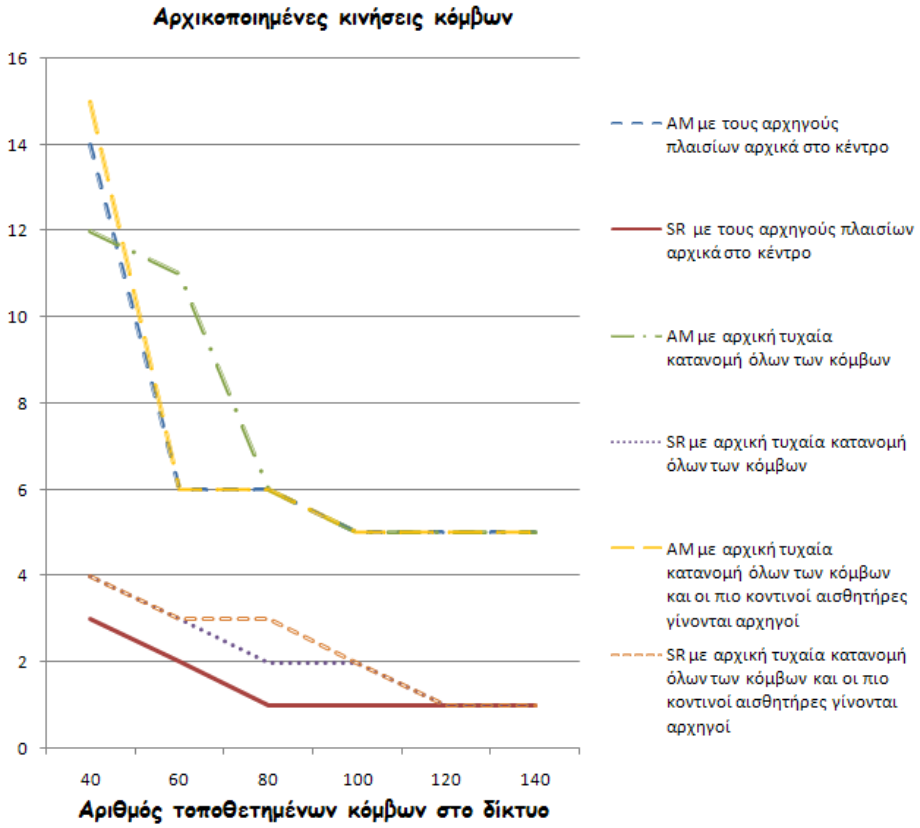


Γραφική παράσταση 1: Συνολικός χρόνος σύγκλισης των αλγορίθμων AM και SR

Είναι σημαντικό να τονίσουμε πως στις γραφικές παραστάσεις 2 και 3 που αφορούν τις κινήσεις των κόμβων δεν περιλάβαμε τις κινήσεις που χρειάζονται οι αισθητήρες που μετακινούνται ως αρχηγοί στο κάθε πλαίσιο, δηλαδή για τις διαστάσεις δικτύου 80x100 m που χρησιμοποιήσαμε για τα πειράματα χρειάζονται επιπρόσθετα για τους αλγορίθμους που αρχικά τοποθετούν όλους τους αισθητήρες τυχαία ακόμη 20 μετακινήσεις κόμβων για κάθε εκτέλεση του αλγορίθμου. Εστιάσαμε τις παρατηρήσεις μας για τις γραφικές παραστάσεις 2 και 3 στην κύρια διαδικασία του αλγορίθμου και αφήσαμε εκτός την μεγάλη διαφορά που θα έχουν κατά την τυχαία αρχική τοποθέτηση όλων των κόμβων και την δημιουργία των αρχηγών κάθε πλαισίου.

Η πιο κάτω γραφική παράσταση 2 παρουσιάζει τις αρχικοποιημένες κινήσεις των αισθητήρων για τους αλγορίθμους AM και SR κατά την αρχική τοποθέτηση των αρχηγών κάθε πλαισίου στο κέντρο του κάθε πλαισίου, για τους αλγορίθμους AM και SR κατά την αρχική τοποθέτηση όλων των αισθητήρων τυχαία και μετά την μετακίνηση ενός τυχαίου αισθητήρα μέσα σε κάθε πλαίσιο ως αρχηγός κάθε πλαισίου στο κέντρο του κάθε πλαισίου, και για τους αλγορίθμους AM και SR κατά την αρχική τοποθέτηση όλων των αισθητήρων τυχαία και μετά την μετακίνηση του πιο κοντινού αισθητήρα μέσα σε κάθε πλαίσιο ως αρχηγός κάθε πλαισίου στο κέντρο του κάθε πλαισίου. Συγκριμένα αρχικοποιημένες κινήσεις είναι οι κινήσεις που έχουν ενεργοποιηθεί από τους αρχηγούς των πλαισίων για κάποιο άδειο πλαίσιο και στο επόμενο χρονικό διάστημα κάποιες από αυτές θα εκτελεστούν ενώ μπορεί κάποιες από αυτές να μην εκτελεστούν εάν το άδειο πλαίσιο γεμίσει από τουλάχιστον ένα αισθητήρα προτού αρχίσει η εκτέλεση της μετακίνησης, αυτό είναι τυχαίο και δεν παίζει κανένα σημαντικό ρόλο στη σύγκριση. Είναι

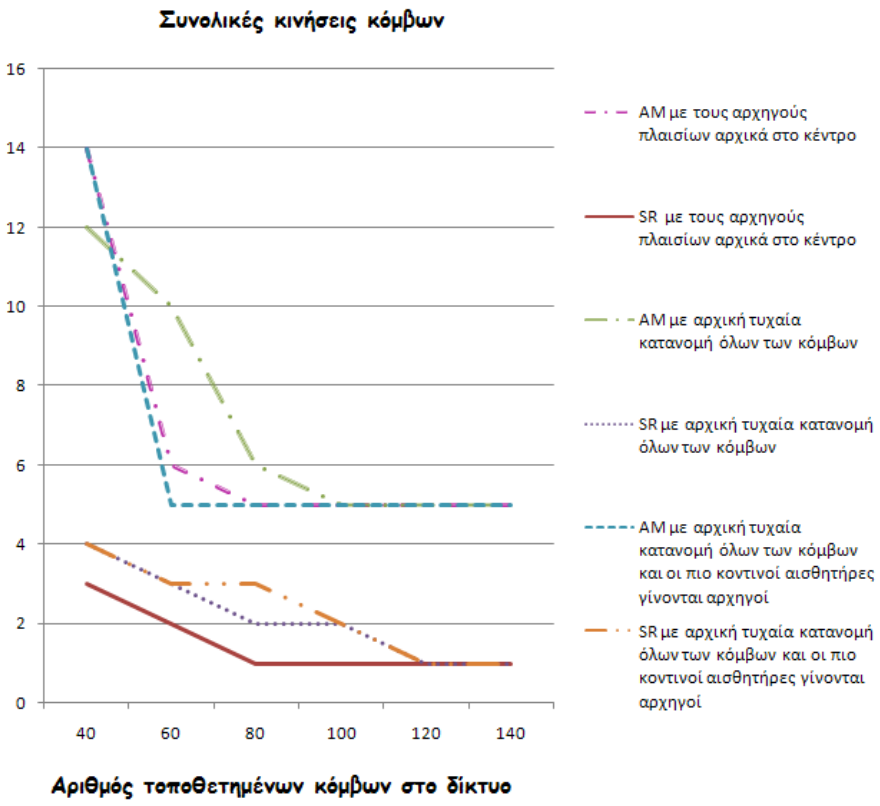
αναμενόμενο ο SR να έχει μικρότερο αριθμό αρχικοποιημένων κινήσεων αφού για κάθε άδειο πλαίσιο είναι υπεύθυνο ένα και μονάχα ένα πλαίσιο, ο "preceding" του, ενώ στον AM μπορεί όλοι οι γειτονικοί αρχηγοί πλαισίων του άδειου πλαισίου να αρχικοποιήσουν μια μετακίνηση και όλες να εκτελεστούν με αποτέλεσμα το άδειο πλαίσιο να γεμίσει με περισσότερους από έναν αισθητήρα. Υπάρχει μια πολύ μικρή διαφορά μεταξύ των τριών διαφορετικών περιπτώσεων που δημιουργήσαμε για τον κάθε αλγόριθμο, AM και SR, αφού η κύρια διαδικασία τους είναι όμοια και αυτό οφείλεται στο ότι για κάθε διαφορετική εκτέλεση ο αριθμός των άδειων πλαισίων όπως και το πλαίσιο είναι τυχαίος και από αυτό εξαρτάται και ο αριθμός των αρχικοποιημένων κινήσεων. Έγιναν πολλές εκτελέσεις για τους αλγορίθμους και παρατηρούμε ότι η συμπεριφορά των διαφορετικών περιπτώσεων που αφορούν την αρχική τοποθέτηση των κόμβων δεν επηρεάζει την κύρια διαδικασία του κάθε αλγορίθμου και συγκεκριμένα ο αριθμός των αρχικοποιημένων κινήσεων των αισθητήρων είναι παρόμοιος για κάθε διαφορετική περίπτωση της αρχικής κατανομής των αισθητήρων του ίδιου αλγορίθμου.



Γραφική παράσταση 2: Οι αρχικοποιημένες κινήσεις των αισθητήρων των αλγορίθμων AM και SR

Στην συνέχεια υπάρχει η γραφική παράσταση 3 που παρουσιάζει τις συνολικές κινήσεις των αισθητήρων για τους αλγορίθμους AM και SR κατά την αρχική τοποθέτηση των αρχηγών κάθε πλαισίου στο κέντρο του κάθε πλαισίου, και για τους αλγορίθμους AM και SR κατά την αρχική τοποθέτηση όλων των αισθητήρων τυχαία και μετά την μετακίνηση των αρχηγών κάθε πλαισίου στο κέντρο του κάθε πλαισίου με βάση τον αισθητήρα με την πιο κοντινή απόσταση. Ο αριθμός συνολικών κινήσεων εξαρτάται από τον αριθμό αρχικοποιημένων κινήσεων και στο επόμενο χρονικό διάστημα εκτελούνται παράλληλα όλες οι κινήσεις των κόμβων και μετά γίνονται οι

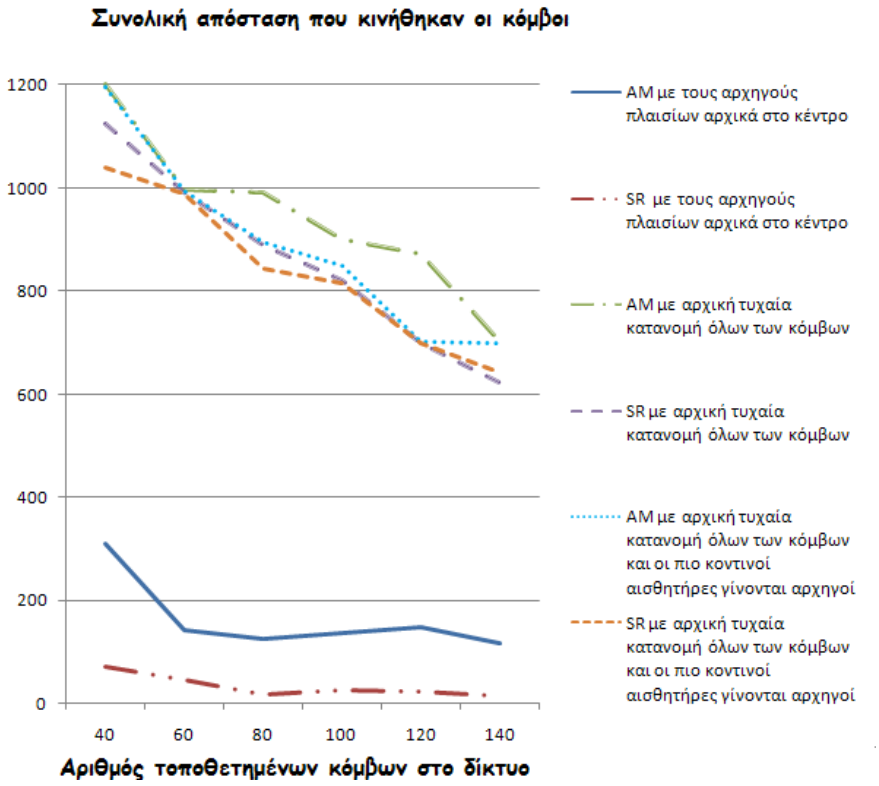
αλλαγές στα πλαίσια με βάση τις κινήσεις που πραγματοποιήθηκαν. Παρατηρούμε ότι οι λιγότερες κινήσεις γίνονται στον αλγόριθμο SR και αυτό είναι αναμενόμενο αφού σε αυτόν τον αλγόριθμο αρχικοποιείται μία και μόνο μία κίνηση για κάθε άδειο πλαίσιο και έτσι εκτελείται μία και μόνο μία κίνηση για να συμπληρωθεί το άδειο πλαίσιο με τουλάχιστον ένα αισθητήρα. Ενώ στον AM αλγόριθμο μπορούν όλοι οι γειτονικοί αρχηγοί κάθε άδειου πλαισίου να εκτελέσουν μία μετακίνηση η οποία αρχικοποιήθηκε και αυτό έχει ως αποτέλεσμα περισσότερες συνολικές κινήσεις αισθητήρων για να επιτύχουν την πλήρη κάλυψη και συνδετικότητα και μπορούμε να παρατηρήσουμε ότι σχεδόν είναι οι διπλάσιες από αυτές στον SR. Βλέπουμε όπως είδαμε και προηγουμένως ότι αριθμός των συνολικών κινήσεων των αισθητήρων είναι παρόμοιος για κάθε διαφορετική περίπτωση της αρχικής κατανομής των αισθητήρων του ίδιου αλγορίθμου.



Γραφική παράσταση 3: Οι συνολικές κινήσεις των αισθητήρων σύγκλισης των αλγορίθμων AM και SR

Και τέλος, πιο κάτω παρουσιάζεται η γραφική παράσταση 4 με την συνολική απόσταση που κινήθηκαν όλοι οι αισθητήρων, για τους αλγορίθμους AM και SR κατά την αρχική τοποθέτηση των αρχηγών κάθε πλαισίου στο κέντρο του κάθε πλαισίου, και για τους αλγορίθμους AM και SR κατά την αρχική τοποθέτηση όλων των αισθητήρων τυχαία και μετά την μετακίνηση των αρχηγών κάθε πλαισίου στο κέντρο του κάθε πλαισίου με βάση τον αισθητήρα με την πιο κοντινή απόσταση, κατά την διάρκεια της διαδικασίας κάθε αλγορίθμου για την επίτευξη της

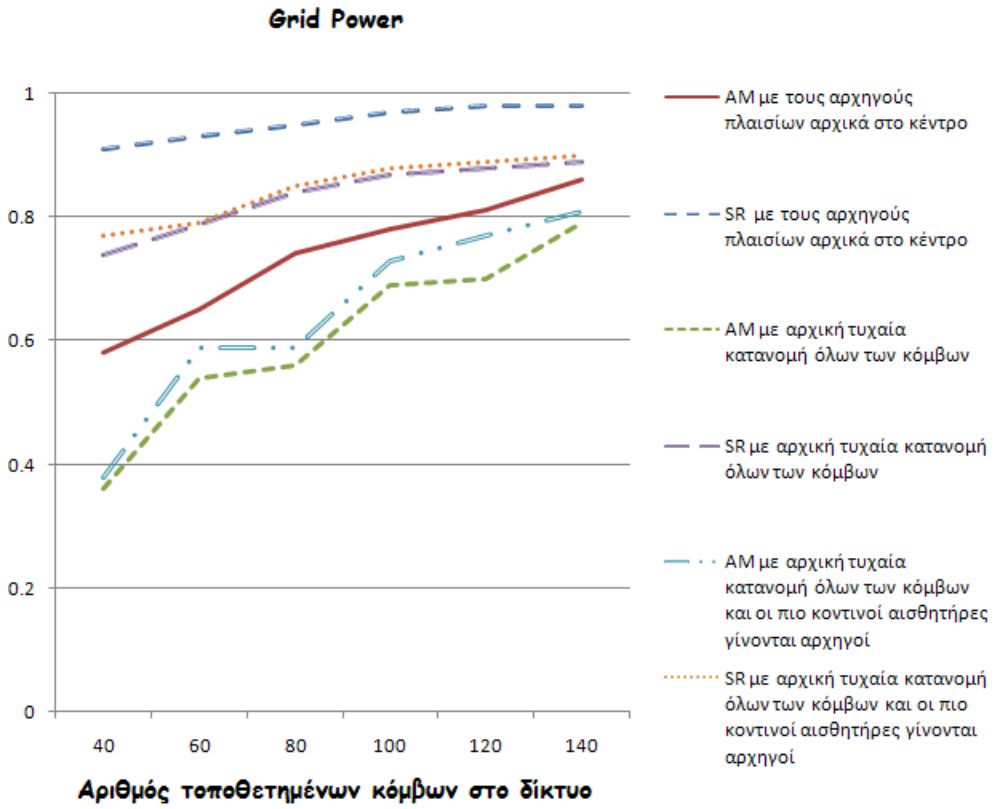
πλήρης κάλυψης και συνδετικότητας. Σημαντική παρατήρηση είναι η μεγάλη διαφορά, περίπου η διπλάσια, στη συνολική απόσταση που κινήθηκαν οι αισθητήρες στον αλγόριθμο AM σε σχέση με τον SR, συγκεκριμένα στον SR η συνολική απόσταση που κινήθηκαν οι κόμβοι είναι μικρότερη από ότι στον AM. Αυτό δεν οφείλεται στον αριθμό των αρχικοποιημένων κινήσεων ή στον αριθμό των συνολικών κινήσεων ή στον αριθμό των άδειων πλαισίων γιατί αυτή η διαφορά πάντα θα υπάρχει εάν λάβουμε υπόψη ότι στον AM μπορούμε να έχουμε για ένα άδειο πλαίσιο περισσότερες από μία μετακινήσεις και αυτό έχει ως επακόλουθο να αυξάνεται και η συνολική απόσταση μετακίνησης. Επίσης βλέπουμε πως η συνολική απόσταση των κινήσεων των αισθητήρων είναι πολύ μεγαλύτερη και για τους δύο αλγορίθμους, AM και SR, στις περιπτώσεις που η αρχική κατανομή όλων των αισθητήρων είναι τυχαία και στη συνέχεια μετακινούνται είτε τυχαία είτε ο αισθητήρας με την πιο κοντινή απόσταση στο κέντρο του κάθε πλαισίου για να γίνει ο αρχηγός του πλαισίου.



Γραφική παράσταση 4: Συνολική απόσταση των κινήσεων των αισθητήρων για τους αλγορίθμους AM και SR

Είναι σημαντικό να σημειώσουμε ότι με βάση όλες τις πιο πάνω παρατηρήσεις θα έχουμε ακόμη μια σημαντική παράμετρο να επηρεάζεται αφού είναι εξαρτώμενη όλων των πιο πάνω παραμέτρων που έχουν καταμετρηθεί. Η συνολική ενέργεια του δικτύου (Network Power), η οποία εάν λάβουμε υπόψη όλες τις γραφικές παραστάσεις που προήλθαν από τα αποτελέσματα σύγκρισης διαφορετικών εκτελέσεων, έχει και αυτή μια μεγάλη διαφορά μεταξύ των δύο

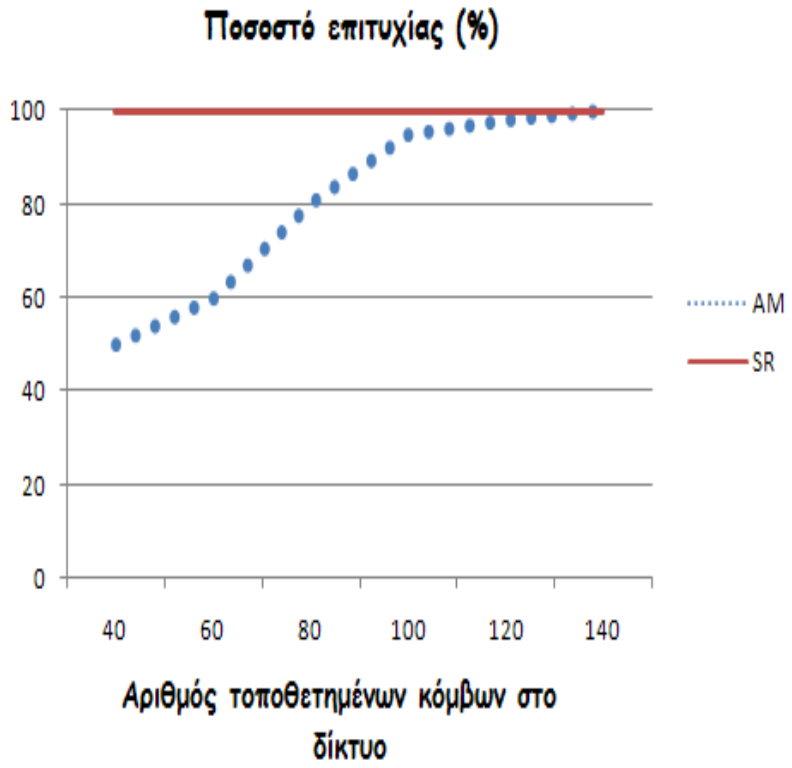
αλγορίθμων SR και AM, όπου στον SR το υπολειπόμενο ποσό συνολικής ενέργειας του δικτύου είναι μεγαλύτερο αφού η συνολική απόσταση μετακίνησης των αισθητήρων και των μηνυμάτων που αποστέλλονται είναι μικρότερη. Ενώ στον AM το υπολειπόμενο ποσό συνολικής ενέργειας του δικτύου είναι λιγότερο λόγω των περισσότερων μετακινήσεων αισθητήρων και μηνυμάτων που αποστέλλονται. Υπάρχει διαφορά μεταξύ των τριών διαφορετικών περιπτώσεων τοποθέτησης των αισθητήρων, δηλαδή ο AM και ο SR με την αρχική τοποθέτηση των αρχηγών κάθε πλαισίου στο κέντρο του κάθε πλαισίου σε σχέση με τον AM και τον SR με την αρχική τοποθέτηση όλων των αισθητήρων τυχαία και μετά την μετακίνηση των αρχηγών κάθε πλαισίου στο κέντρο του κάθε πλαισίου είτε με βάση τον αισθητήρα με την πιο κοντινή απόσταση είτε τυχαία, η οποία οφείλεται στις επιπρόσθετες μετακινήσεις των τυχαία τοποθετημένων αισθητήρων που είναι πιο κοντά ή στην περίπτωση που παίρνουμε κάποιον τυχαίο στο κέντρο του κάθε πλαισίου για να γίνουν οι αρχηγοί του καθενός πλαισίου αντίστοιχα. Έτσι στους αλγορίθμους AM και SR με την αρχική τοποθέτηση όλων των αισθητήρων τυχαία και μετά την μετακίνηση των αρχηγών κάθε πλαισίου στο κέντρο του κάθε πλαισίου είτε τυχαία είτε με βάση τον αισθητήρα με την πιο κοντινή απόσταση παρατηρούμε λιγότερη υπολειπόμενη συνολική ενέργεια του δικτύου σε σχέση με τους αντίστοιχους αλγορίθμους AM και SR με την αρχική τοποθέτηση των αρχηγών κάθε πλαισίου στο κέντρο του κάθε πλαισίου.



Γραφική παράσταση 5: Το Grid Power κατά την εκτέλεση των αλγορίθμων AM και SR

Ο SR κατάφερνε να ολοκληρώσει επιτυχώς ακόμη και με την τοποθέτηση πολύ μικρού αριθμού αισθητήρων, δηλαδή μικρότερης πυκνότητας αισθητήρων, για παράδειγμα με αριθμό αισθητήρων κάτω από το τριπλάσιο των πλαισίων, με μόνο μειονέκτημα την αύξηση του χρόνου σύγκλισης, λόγο των μεγαλύτερων μονοπατιών «Hamilton» που δημιουργούνταν, ενώ ο AM κάτω από τέτοιες συνθήκες, χαμηλής πυκνότητας αισθητήρων, υπάρχει πιθανότητα αποτυχίας του αφού χρειάζεται περισσότερους κόμβους για την ολοκλήρωση των διαδικασιών του. Δηλαδή όταν ο αριθμός των αισθητήρων που τοποθετούνταν ήταν κάτω από το τριπλάσιο των πλαισίων

ένα μεγάλο ποσοστό των εκτελέσεων του αλγορίθμου AM αποτύγχανε. Αυτό μπορούμε να το παρατηρήσουμε και μέσα από τη γραφική παράσταση 6.



Γραφική παράσταση 6: Το ποσοστό επιτυχίας (%) κατά την εκτέλεση των αλγορίθμων AM και SR

Κεφάλαιο 4

Συμπεράσματα

Μετά από τη μελέτη όλων αυτών των επιστημονικών άρθρων και την υλοποίηση δύο αλγορίθμων του AM και του SR και την εξαγωγή αποτελεσμάτων, μπορούμε να συμπεράνουμε ότι για την εφαρμογή ενός αλγορίθμου σε ένα ασύρματο δίκτυο κινητών αισθητήρων θα ήταν μια πολύ καλή επιλογή ο αλγόριθμος SR. Επειδή ο SR μπορεί να εγγυηθεί την επίτευξη της πλήρης κάλυψης και συνδετικότητας με τον λιγότερο δυνατό φόρτο στο δίκτυο, διατηρώντας έτσι σε όσο το δυνατό υψηλότερα επίπεδα την υπολειπόμενη ενέργεια του δικτύου.

Τόσο στον αριθμό των συνολικών κινήσεων των αισθητήρων όσο και στη συνολική απόσταση μπορούμε να συμπεράνουμε ότι η διαφορά μεταξύ των δύο αλγόριθμων έφτανε περίπου το 50%, δηλαδή στον SR έχουμε περίπου τις μισές συνολικές κινήσεις αισθητήρων και συνολική απόσταση που κινήθηκαν σε σύγκριση με τον AM.

Όσον αφορά τις διαφορετικές περιπτώσεις αρχικής τοποθέτησης των αισθητήρων που υλοποιήσαμε και συγκρίναμε μεταξύ τους μπορούμε να συμπεράνουμε πως χρησιμοποιώντας τους αλγόριθμους με την τυχαία αρχική τοποθέτηση όλων των κόμβων υπάρχει

περισσότερος χρόνος υπολογισμού, περισσότερες μετακινήσεις και μεγαλύτερη συνολική απόσταση και έτσι μειώνεται περισσότερο η συνολική ενέργεια του δικτύου (network power), λόγω του ότι οι αρχηγοί των πλαισίων αρχικά τοποθετούνται τυχαία και μετά μετακινούνται στο κέντρο του κάθε πλαισίου. Συγκριτικά μεταξύ τους οι αλγόριθμοι αρχικής τυχαίας τοποθέτησης όλων των κόμβων τόσο του AM όσο και του SR, κατείχε ελαφριά μεγαλύτερο χρόνο υπολογισμού αυτός που μετακινούσε τους πιο κοντινούς αισθητήρες του κάθε πλαισίου στο κέντρο του για να γίνουν αρχηγοί και αυτός που έπαιρνε τυχαία ένα αισθητήρα μέσα στο πλαίσιο και τον μετακινούσε στο κέντρο είχε λίγο μεγαλύτερη συνολική απόσταση μετακινήσεων των κόμβων.

Επίσης κατά της εκτελέσεις των αλγορίθμων με πολύ λίγους κινητούς αισθητήρες ο SR είχε πολύ μεγάλο χρόνο σύγκλισης μιας και το μονοπάτι «Hamilton» ήταν πολύ μεγαλύτερο για να επιτευχθεί η κάλυψη στο δίκτυο, ενώ στον AM υπήρξαν και φορές που δεν ολοκλήρωνε όταν ο αριθμός των κινητών αισθητήρων ήταν πάρα πολύ μικρός και αποτύγχανε να ολοκληρώσει τις διαδικασίες του. Το ποσοστό επιτυχίας του αλγορίθμου AM μειώνεται αισθητά όταν μειώνεται η πυκνότητα των αισθητήρων, ενώ του αλγορίθμου SR παραμένει πάντα σε ψηλότερα επίπεδα.

Κεφάλαιο 5

Μελλοντική εργασία

Θα ήταν καλό σε μελλοντική εργασία να γίνει υλοποίηση και άλλων αλγορίθμων που περιγράφηκαν πιο πάνω και μελετήθηκαν μέσα από επιστημονικά άρθρα και να υπάρξει έτσι η δυνατότητα σύγκρισης όλων αυτών των αλγορίθμων με βάση τον τύπο ασύρματου δικτύου αισθητήρων κάτω από το οποίο μπορούν να ενεργήσουν.

Ακόμη θα μπορούσε να μελετηθεί βαθύτερα η μετακίνηση των αισθητήρων, δηλαδή η απόφαση για μετακίνηση κάποιου αισθητήρα να βασίζετε και σε άλλα κριτήρια που μπορεί να είναι εξίσου σημαντικά. Για παράδειγμα η μετακίνηση των κινητών αισθητήρων να γίνετε μόνο από περιοχές που δεν υπάρχει μεγάλη ανάγκη για μεγάλη πυκνότητα κόμβων και όχι από περιοχές που έχουν περισσότερη ανάγκη να έχουν μεγάλο αριθμό αισθητήρων.

Επίσης μπορεί να γίνει και μια εκτενής μελέτη κατά πόσο θα μπορούσαν αλγόριθμοι που εκτελούνται σε ένα ορισμένο τύπο ασύρματου δικτύου να εφαρμοστούν και σε διαφορετικό. Για παράδειγμα αλγόριθμοι που λειτουργούν σε κινητά ασύρματα δίκτυα αισθητήρων εάν μπορούν να λειτουργήσουν και σε υβριδικά ασύρματα δίκτυα αισθητήρων με το ίδιο ποσοστό επιτυχίας.

Βιβλιογραφία

[1] Akyildic, W.Su, Y. Sankarasubramaniam, E. Cayirci "A survey on sensor networks",IEEE Communication Magazine, 2002

[2] A. Bharathidasan, VAS Ponduru "Sensor Networks: An Overview", IEEE INFOCOM'04, 2004

[3] D. Culler, D. Estrin, M. Srivastava "Overview of Sensor Networks", Computer, vol.37, no. 8, pp.41—49, 2004

[4] C. Schindelhaue "Mobility in Wireless Networks", SOFSEM 2006, LNCS 3831, pp. 100—116, 2006

[5] M. Grossglauser, DNC Tse "Mobility increases the capacity of ad hoc wireless networks", IEEE/ACM Transactions on Networking (TON), 2002

[6] B Liu, P Brass, O Dousse, P Nain, D Towsley, "Mobility Improves Coverage of Sensor Networks", MobiHoc'05, May 25-27, 2005

[7] A Kansal, M Rahimi, WJ Kaiser, MB Srivastava,GJ Pottie, D Estrin "Controlled Mobility for Sustainable Wireless", IEEE SECON, 2004

[8] G. Wang, G. Cao, and T. La Porta, "Movement-Assisted Sensor Deployment", IEEE INFOCOM '04, 2004

[9] G. Wang, G. Cao, and T. La Porta, "A Bidding Protocol for Sensor Deployment", IEEE ICNP'03, Nov., 2003

[10] G. Wang, G. Cao, and T. La Porta, “Sensor Relocation in Mobile Sensor Networks”, IEEE INFOCOM’05, 2005

[11] F. L. Lewis, “Wireless Sensor Networks”, Smart Environments: Technologies, Protocols, and Applications, 2004

[12] K Dantu, M Rahimi, H Shah, S Babel, A Dhariwal, GS Sukhatme “Robomote: Enabling Mobility In Sensor Networks”, Proceedings of the 4th international symposium on Information processing in sensor networks, 2005

[13] J. Friedman et al, “Ragobot a new hardware platform for research in wireless mobile sensor networks” University of California, LA

[14] Mo Li, Baijian Yang: A Survey on Topology issues in Wireless Sensor Network. ICWN 2006

[15] C Chen, J Ma, “MEMOSEN Multi-radio Enabled Mobile Wireless Sensor Network”, Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA’06), 2006

[16] Zhen Jiang, Jie Wu, Robert Kline, “Mobility Control for Coverage in Wireless Sensor Networks”, The 28th International Conference on Distributed Computing Systems Workshops 2008

[17] Jorge Cortes, Sonia Martinez, Timur Karatas, Francesco Bullo, “Coverage control for mobile sensing networks”, November 4, 2002, IEEE Transactions on robotics and automation.

[18] Zhen Jiang, Jie Wu, Afrand Agah, Bin Lu, “Topology Control for Secured Coverage in Wireless Sensor Networks”, 2007 IEEE

[19] Jie Wu, Shuhui Yang, “SMART: A Scan-Based Movement-Assisted Sensor Deployment Method in Wireless Sensor Networks”, 2005 IEEE

[20] Khin Thanda Soe, “Increasing Lifetime of Target Tracking Wireless Sensor Networks”, PWASET VOLUME 32 August 2008 ISSN 2070-3740

[21] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, Christopher Gill, “Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks”, SenSys’03, November 5-7, 2003, Los Angeles, California, USA

[22] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks", ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001), Rome, Italy, July 16-21, 2001.

[23] Chi-Fu Huang, Yu-Chee Tseng, “The Coverage Problem in a Wireless Sensor Network”. In Proceedings of the 2nd ACM WSNA’03, Sep 2003, San Diego, CA, USA.

[24] Theofanis P. Lambrou, Christos G. Panayiotou, “Collaborative Event Detection Using Mobile and Stationary Nodes in Sensor Networks”

[25] Xingfa Shen, Jiming Chen, Youxian Sun, “Grid Scan: A Simple and Effective Approach for Coverage Issue in Wireless Sensor Networks”, 2006 IEEE.

[26] Wei Li , Christos G. Cassandras, “Distributed Cooperative Coverage Control of Sensor Networks”, Dept. of Manufacturing Engineering and Center for Information and Systems Engineering Boston University, Brookline, MA 02446

[27] Amitabha Ghosh, «Estimating Coverage Holes and Enhancing Coverage in Mixed Sensor Networks»

[28] Nadeem Ahmed, Salil S. Kanhere, Sanjay Jha, "The Holes Problem in Wireless Sensor Networks: A Survey", Mobile Computing and Communications Review, Volume 9, Number 2, Sydney, Australia.

[29] Ruay-Shiung Chang, Shuo-Hung Wang, "Deploying Sensors for Maximum Coverage in Sensor Networks", IWCMC'07, August 12-16, 2007, Honolulu, Hawaii, USA.

[30] Andrew Howard, Maja J Mataric, Gaurav S Sukhatme, «Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem», In Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02) Fukuoka, Japan, June 25-27,
2002

[31] Saurabh Ganeriwal, Aman Kansal, Mani B. Srivastava, "Self Aware Actuation for Fault Repair in Sensor Networks", Proceedings of the 2004 IEEE, international Conference on Robotics & Automation New Orleans, LA, April 2004

[32] S. Meguerdichian, K. Farinaz, P. Miodrag, M. B. Srivastava, "Coverage Problems in Wireless Ad Hoc Sensor Networks," EEE ICC, 2001.

[33] My T. Thai, Yingshu Li, Feng Wang, "O(log n)-Localized Algorithms on the Coverage Problem in Heterogeneous Sensor Networks", 2007 IEEE.

[34] Nojeong Heo, Pramod K. Varshney, "An Intelligent Deployment and Clustering Algorithm for a Distributed Mobile Sensor Network*"

[35] Maxim A. Batalin, Gaurav S. Sukhatme, “Spreading Out: A Local Approach to Multi-robot Coverage”, In *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems* pp. 373-382, Fukuoka, Japan, June 25-27, 2002

[36] Mohamed K. Watfa, Sesh Commuri, “An Energy Efficient Approach to D ynamic Coverage in Wireless Sensor Networks”, JOURNAL OF NETWORKS, VOL. 1, NO. 4, AUGUST 2006

Παραρτήματα

Παράρτημα Α

Κώδικας υλοποίησης των αλγορίθμων AM και SR που κατά την αρχική τυχαία τοποθέτηση των αισθητήρων κάθε πλαισίου τοποθετούνται στο κέντρο του κάθε πλαισίου οι αρχηγοί του κάθε πλαισίου αντίστοιχα χωρίς τυχαιότητα στην τοποθέτηση τους

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      WSN_COVandCONN_Sim                %
%            Coverage & Connectivity Control in WSNs      %
%                         Kakoulli Elena                  %
%                     Computer Science, UCY               %
%                                                         %
%                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = WSN_COVandCONN_Sim(varargin)
    gui_Singleton = 1;
    gui_State = struct('gui_Name', mfilename, 'gui_Singleton', gui_Singleton, 'gui_OpeningFcn', @WSN_COVandCONN_Sim_OpeningFcn, 'gui_OutputFcn', @WSN_COVandCONN_Sim_OutputFcn, 'gui_LayoutFcn', [], 'gui_Callback', []);
    if (nargin && ischar(varargin{1}))
        gui_State.gui_Callback = str2func(varargin{1});
    end
    if (nargout)
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
```

```
end % End initialization code


% --- Executes just before WSN_CovANDConn_Sim is made visible.
function    WSN_COVandCONN_Sim_OpeningFcn(hObject,    eventdata,    handles,
varargin)
    GlobalVars();
    %clear workspace; clc;
    handles.output = hObject; guidata(hObject, handles);
    axes(handles.GridAxes); cla;
    sensorLogoHandle = imshow('sensorLogo.jpg');
    axis([get(sensorLogoHandle, 'YData') get(sensorLogoHandle, 'XData')]);
    SetSamples(handles, 'initAll', 'hide'); % Set default values
    if (nodesDispersedF)
        ClearHandles(handles);
    end

    set(handles.COVandCONN_AM, 'Value',0);
    set(handles.COVandCONN_SR, 'Value',0);

    switch (COVandCONNAlg)
        case 'AM'
            set(handles.COVandCONN_AM, 'Value',1);
        case 'SR'
            set(handles.COVandCONN_SR, 'Value',1);
    end

    simFileInfo = dir('WSN_COVandCONN_Sim.m');
    set(handles.clearGrid, 'String','Quit');
    set(handles.statusText,  'String','Enter  Grid  parameters  &  choose
algorithm...');
    set(handles.analyzedNode, 'String','');
    set(handles.pwrRem, 'String','');
    set(handles.WSN_COVandCONN_Sim,   'Name',   sprintf('WSN   Coverage   &
Connectivity Algorithms Simulator by elen@  -   %s', simFileInfo.date));
end

% --- Outputs from this function are returned to the command line.
```

```
function varargout = WSN_COVandCONN_Sim_OutputFcn(hObject, eventdata,
handles)
    varargout{1} = handles.output;
    reqVer = '7.1.0.19920 (R14)';
    if (version ~= reqVer) % Incorrect Matlab version detected
        msgbox('Matlab R14 SP3 is required to ensure proper GUI
performance.');
    end
end

% ================ BUTTON FUNCTIONS ================
function editParams_Callback(hObject, eventdata, handles)
    ParamEdit();
end

function recharge_Callback(hObject, eventdata, handles)
    GlobalVars();
    if (nodesDispersedF)
        nodePower = nodePowerMax .* ones(1, nodeCount); % Maximize the
power of each node
        %UpdateAnalyzedNode(handles, closestNode);
        UpdatePowerAxes(handles, 'noCalc');
        axes(handles.GridAxes);
        PlotGrid(); % Redraw the nodes to refresh their color
    end
end

function clearGrid_Callback(hObject, eventdata, handles)
    GlobalVars();
    if (networkSyncF)
        set(handles.source_sinkTime, 'String','');
        set(handles.numRX, 'String','');
        set(handles.droppedTX, 'String','');
        UpdateGridAxes(handles);
        DrawTopology(0);
        PlotGrid();
        networkSyncF = 0;
    elseif (networkLvlDiscF) % Clear the lvl-disc lines if no sync has been
performed
```

```
        ClearHandles(handles);
        UpdateGridAxes(handles);
        PlotGrid();
        set(handles.NodesMove, 'String','');
        set(handles.statusText, 'String','Enter network parameters & choose
algorithm...');
        networkLvlDiscF = 0;
    elseif (nodesDispersedF) % Clear nodes and reset to the opening screen
        WSN_COVandCONN_Sim_OpeningFcn(hObject, eventdata, handles, 1);
        axes(handles.PwrAxes); cla; axis([0 1 0 1]);
        set(handles.Moves, 'String','');
        set(handles.Distance, 'String','');
        set(handles.analyzedNode, 'String','');
        set(handles.pwrRem, 'String','');
        set(handles.clearGrid, 'String','Quit');
        nodesDispersedF = 0;
    else
        close;
    end
end

function plotGrid_Callback(hObject, eventdata, handles)
    GlobalVars(); clc;
    set(handles.clearGrid, 'String','Clear Grid');
    set(handles.statusText,    'String','Initializing    Network,    please
waiting...');
    if  (~dataLoadedF)  %  Set  variables  here  to  avoid  disrupting  loaded
values
        SetSamples(handles, 'initArrays', 'hide'); % Set default values
        networkChangedF = 0;
        nodesDispersedF = 1;
        networkLvlDiscF = 0;
        networkSyncF = 0;


    else
        dataLoadedF = 0;
    end
    UpdateGridAxes(handles);
```

```
    PlotGrid();
    if (~networkChangedF && networkLvlDiscF)
    % Needed in case the data has been loaded and we want the discovering
lines...NetworkChangedF flag must be 0 to redraw the discovering lines
        DrawTopology(0);
    end
    UpdatePowerAxes(handles, 'noCalc');

    ClearHandles(handles);
    set(handles.Distance, 'String',sprintf('%g m', moveDistances));
    set(handles.Moves,          'String',sprintf('%g          /          %g',
movesNodes,ProcessesInit));

    set(handles.statusText,  'String','Enter  network  parameters  &  choose
algorithm...');
    %set(handles.analyzedNode, 'String',sourceNode);
    set(handles.pwrRem, 'String',nodePower(sourceNode));
    set(handles.topolDiscovery, 'Enable','on');
    set(handles.run, 'Enable','on');
end

function topolDiscovery_Callback(hObject, eventdata, handles)
    GlobalVars(); clc;
    if (networkChangedF || ~nodesDispersedF || dataLoadedF)
        plotGrid_Callback(hObject, eventdata, handles);
    end

    maxDistanceLow = nthroot(PtWLow/PrThresholdWLow, pathLossCoeff);
    maxDistanceHigh = nthroot(PtWHigh/PrThresholdWHigh, pathLossCoeff);

    set(handles.statusText, 'String','Simulating Topology Discovery, please
waiting...');
    set(handles.plotGrid, 'Enable','off');
    set(handles.clearGrid, 'Enable','off');
    set(handles.topolDiscovery, 'Enable','off');
    set(handles.run, 'Enable','off');
    set(handles.COVandCONN_AM, 'Enable','off');
    set(handles.COVandCONN_SR, 'Enable','off');
    set(handles.editParams, 'Enable','off');
```

```
    set(handles.recharge, 'Enable','off');
    pause(0.001);
    UpdateGridAxes(handles);
    PlotGrid();

    s = cputime;
    discoverTime = cputime - s;

    DrawTopology(pauseInt);
    ClearHandles(handles);
    PlotGrid();
    UpdatePowerAxes(handles);
    %UpdateAnalyzedNode(handles, closestNode);
    UpdateNodeStatus(handles);
    set(handles.plotGrid, 'Enable','on');
    set(handles.clearGrid, 'Enable','on');
    set(handles.topolDiscovery, 'Enable','on');
    set(handles.run, 'Enable','on');
    set(handles.COVandCONN_AM, 'Enable','on');
    set(handles.COVandCONN_SR, 'Enable','on');
    set(handles.editParams, 'Enable','on');
    set(handles.recharge, 'Enable','on');
    set(handles.discoverTime, 'String',discoverTime);
    set(handles.numRX, 'String',numRx);
    networkLvlDiscF = 1;
    networkSyncF = 0;
    set(handles.statusText, 'String','Topology Discovery completed!!');
end

function run_Callback(hObject, eventdata, handles)
    GlobalVars();
    networkSyncF = 1;
   if (networkChangedF || ~networkLvlDiscF)
        topolDiscovery_Callback(hObject, eventdata, handles);
        if (continuousF)
            %pause(1);
        end
    end
```

```
set(handles.plotGrid, 'Enable','off');
set(handles.clearGrid, 'Enable','off');
set(handles.topolDiscovery, 'Enable','off');
set(handles.run, 'Enable','off');
set(handles.COVandCONN_AM, 'Enable','off');
set(handles.COVandCONN_SR, 'Enable','off');
set(handles.editParams, 'Enable','off');
set(handles.recharge, 'Enable','off');
set(handles.statusText, 'String','Grid Heads discovering...');
pause(0.001);
UpdateGridAxes(handles);
DrawTopology(0);
PlotGrid();

numTx = 0;
numRx = 0;
dropTx = 0;
s = cputime;

switch (COVandCONNAlg)
    case 'AM'
        COVandCONNAlgorithmAM();
    case 'SR'
        COVandCONNAlgorithmSR();
end

source_sinkTime = cputime - s;

% Continuously run
    while (continuousF && source_sinkTime <= 10)
        pause(1);
        UpdateGridAxes(handles);
        DrawTopology(0);
        PlotGrid();
        UpdatePowerAxes(handles);
        pause(1);
```

```
            numTx = 0;
            numRx = 0;
            dropTx = 0;
            UpdateGridAxes(handles);
            DrawTopology(0);
            PlotGrid();
            s = cputime;

            switch (COVandCONNAlg)
                case 'AM'
                    COVandCONNAlgorithmAM();
                case 'SR'
                    COVandCONNAlgorithmSR();
            end

            source_sinkTime = cputime - s;
            %UpdateAnalyzedNode(handles, closestNode);
            UpdateNodeStatus(handles);
            networkSyncF = 1;
        end

    UpdatePowerAxes(handles);
    UpdateAnalyzedNode(handles, 1);
    UpdateNodeStatus(handles);
    set(handles.plotGrid, 'Enable','on');
    set(handles.clearGrid, 'Enable','on');
    set(handles.topolDiscovery, 'Enable','on');
    set(handles.run, 'Enable','on');
    set(handles.COVandCONN_AM, 'Enable','on');
    set(handles.COVandCONN_SR, 'Enable','on');
    set(handles.editParams, 'Enable','on');
    set(handles.recharge, 'Enable','on');
    set(handles.source_sinkTime, 'String',source_sinkTime);
    set(handles.numRX, 'String',numRx);
    set(handles.droppedTX, 'String',DeadNodes);
    set(handles.statusText, 'String','Simulation completed!!!');
end

% ------------------- NON-BUTTON CALLBACKS -------------------
```

```
function continuous_Callback(hObject, eventdata, handles)
    GlobalVars();
    continuousF = get(gco,'Value');
end

function COVandCONN_AM_Callback(hObject, eventdata, handles)
    GlobalVars();
    set(handles.COVandCONN_AM, 'Value',1);
    set(handles.COVandCONN_SR, 'Value',0);
    COVandCONNAlg = 'AM';
end

function COVandCONN_SR_Callback(hObject, eventdata, handles)
    GlobalVars();
    set(handles.COVandCONN_AM, 'Value',0);
    set(handles.COVandCONN_SR, 'Value',1);
    COVandCONNAlg = 'SR';
end

function GridAxes_ButtonDownFcn(hObject, eventdata, handles)
    GlobalVars();
% Check to make sure that the nodes have been plotted and the user has
either performed network level discovery or wants to change the source node
    if (networkChangedF || dataLoadedF)
        plotGrid_Callback(hObject, eventdata, handles);
        tempStr = get(handles.statusText, 'String');
        tempColor = get(handles.statusText, 'BackgroundColor');
        set(handles.statusText, 'String','Network parameters changed!!');
        set(handles.statusText, 'BackgroundColor','red');
        pause(2);
        set(handles.statusText, 'String',tempStr);
        set(handles.statusText, 'BackgroundColor',tempColor);
        networkChangedF = 0;
        dataLoadedF = 0;
    elseif (nodesDispersedF)
        pt = get(gca,'currentpoint'); % Find the nearest node to the mouse-
click
        closestDist = xDistance;
        for i = 1 : nodeCount
```

```
            d   =   CalcDistance(pt(1,1),   pt(1,2),   neighborTable(i,1),
neighborTable(i,2));
            if  (d < closestDist)
                closestNode = i; % Nearest node to mouse-click after loop
is complete
                closestDist = d;
            end
        end

        if (changeSourceNodeF) % Change source node
            sourceNode = closestNode;
            closestNode = sourceNode;
            ClearHandles(handles);

            axes(handles.GridAxes);
            cla; axis([0 xDistance 0 yDistance]); axis on; axis xy; axis
fill; axis manual; hold all;
            PlotGrid();
            set(handles.Moves, 'String','');
            set(handles.statusText,     'String','Enter     topolDiscovery
parameters...');
            networkLvlDiscF = 0;
            networkSyncF = 0;
            UpdateAnalyzedNode(handles, closestNode);
        else % Analyze node's properties
            PlotGrid();
            UpdateAnalyzedNode(handles, closestNode);
            if (networkLvlDiscF)
                PlotGrid();
                if (tracebackF && gridHeads(closestNode) ~= 0)
                    UpdateGridAxes(handles);
                    DrawTopology(0);
                    PlotGrid();

                end
            end
        end
    end
end
```

```
function PwrAxes_ButtonDownFcn(hObject, eventdata, handles)
    GlobalVars();
    pt = get(gca,'currentpoint'); % Find the nearest node to the mouse-
click
    closestDist = length(gridPower);
    for i = 1 : length(gridPower)
        d = CalcDistance(pt(1,1), pt(1,2), i, gridPower(i));
        if  (d < closestDist)
            closestPt = i; % Nearest event to mouse-click after loop is
complete
            closestDist = d;
        end
    end
end

% ================== MENU FUNCTIONS ==================
function Menu_LoadPar_Callback(hObject, eventdata, handles) % FILE
FUNCTIONS
    GlobalVars();
    [fname pname]= uigetfile('*.mat', 'Open');
    if ((ischar(fname))& (ischar(pname)))
        curdir = pwd;
        cd(pname);
        load(fname); % Load the entire workspace
        cd(curdir);
        SetSamples(handles, 'load','hide');
        dataLoadedF = 1;
        msgbox('Parameters loaded succesfully!');
    end
end

function Menu_SavePar_Callback(hObject, eventdata, handles)
    GlobalVars();
    [fname pname] = uiputfile('parameters.mat', 'Save As');
    if ((ischar(fname))& (ischar(pname)))
        curdir = pwd;
        cd(pname);
        fid = fopen(fname,'wt');
```

```
        save(fname, '*'); % Save the entire workspace
        fclose(fid);
        cd(curdir);
        networkLvlDiscF
        msgbox(sprintf('%s%s',  'Parameters  successfully  saved  to  ',
fname));
    end
end

function Menu_Exit_Callback(hObject, eventdata, handles)
    %clear all;
    if (isdeployed)
        close all;
        quit force;
    else
        close all;
    end
end

function  Menu_View_NodeIDs_Callback(hObject,  eventdata,  handles)  %  VIEW
FUNCTIONS
    GlobalVars();
    if strcmp(get(gcbo,'Checked'),'on')
        viewNodeIDF = 0;
        set(gcbo, 'Checked','off');
    else
        viewNodeIDF = 1;
        set(gcbo, 'Checked','on');
    end
    if (nodesDispersedF) % Update the grid axes only if the nodes have been
dispersed
        UpdateGridAxes(handles);
        if (networkLvlDiscF) % Draw discovering lines only if network has
already been lvl-discovered
            DrawTopology(0);
        end
        PlotGrid();
    end
end
```

```
function Menu_MatlabVer_Callback(hObject, eventdata, handles) % HELP
FUNCTIONS
    msgbox(sprintf('Current Matlab Version:\n%s', version), 'Matlab
Version');
end

function Menu_Info_Callback(hObject, eventdata, handles)
    titleStr = 'WSN Coverage & Connectivity Control Simulator';
    authorStr = sprintf('%s\n%s', 'Elena Kakoulli, UCY');
    simFileInfo = dir('WSN_COVandCONN_Sim.m');
    msgbox(sprintf('%s\n%s\n\nLast Updated on %s', titleStr, authorStr,
simFileInfo.date), 'Info');
end

% ================== OTHER FUNCTIONS ==================

function UpdateAnalyzedNode(handles, node)
    GlobalVars();
    set(handles.analyzedNode, 'String',node);
    set(handles.pwrRem, 'String',nodePower(node));
    if (networkLvlDiscF) % Output discovering details

        if (gridHeads(node) == 0)
            set(handles.GridHead, 'String','Orphan');
        else
            set(handles.GridHead, 'String',num2str(gridHeads(node)));
        end

        set(handles.NodesMove,
'String',num2str(NodesMove(1:movesNodes,1)));

        if (nodePower(node) <= 0)
            set(handles.pwrRem, 'String','Depleted');
        else
            set(handles.pwrRem, 'String',num2str(nodePower(node)));
        end
    end
end
```

```
function UpdateNodeStatus(handles)
    GlobalVars();

    for i = 1 : nodeCount
        if (nodePower(i) <= 0)
            gridHeads(i) = 0;
        end
    end

    set(handles.Distance, 'String',sprintf('%g m', moveDistances));
    set(handles.Moves,          'String',sprintf('%g          /          %g',
movesNodes,ProcessesInit));

end

function UpdateGridAxes(handles)
    GlobalVars();
    axes(handles.GridAxes); cla;
    axis([0 xDistance 0 yDistance]);
    axis on; axis xy; axis fill; axis manual; hold all;
end

function UpdatePowerAxes(handles, x)
    GlobalVars();
    if (nargin == 2 & x == 'noCalc')
        gridPower = [1];
    else
        gridPower  =  [gridPower  sum(nodePower)/(nodePowerMax*nodeCount)]  %
Calculate new normalized grid power
    end
    axes(handles.PwrAxes); cla;
    axis([1 length(gridPower)+1 0 1]);
    %axis([0 source_sinkTime 0 1]);
    set(handles.PwrAxes, 'YGrid','on', 'YMinorGrid','on');
    axis on; axis xy; axis fill; axis manual; hold all;
    plot(gridPower, '-b*');
end
```

```
% --- Executes during object creation, after setting all properties.
function WSN_COVandCONN_Sim_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WSN_COVandCONN_Sim (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Update Node Power                              %
%             Coverage & Connectivity Control in WSNs                  %
%                         Kakoulli Elena                               %
%                    Computer Science, UCY                             %
%                                                                      %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function UpdateNodePower()
    GlobalVars();
    for i = 1 : nodeCount
        if (nodePower(i) <= 0) % Node i has run out of energy
            DeadNodes = DeadNodes + 1;
            nodeLevel(i) = -1;
            gridHeads(i) = 0;
        end
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                         Set Samples                                  %
%             Coverage & Connectivity Control in WSNs                  %
%                         Kakoulli Elena                               %
%                    Computer Science, UCY                             %
%                                                                      %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function SetSamples(handles, sampleID, visibility)
    GlobalVars();
    switch sampleID
        case 'initAll'
            SetSamples(handles, 'mica2dot433', 'hide'); % Set default
values
            propSpd = 2.998e8;
            pauseInt = 0.2;
            xDistance = 80;
            yDistance = 100;
            nodeCount = 70;
            nodeDistribMode = 'rand';
            pathLossCoeff = 3.5;
```

```
            nodePowerMax = 100;
            nodePowerWarn = 10;
            rediscoverLimit = 20;
            COVandCONNAlg = 'AM';

        % Set flags and source node
            NodesMove = zeros(nodeCount,1);
            dataLoadedF = 0;
            nodesDispersedF = 0;
            networkChangedF = 0;
            networkLvlDiscF = 0;
            networkSyncF = 0;
            viewNodeIDF = 0;
            changeSourceNodeF = 0;
            selectNodeF = 0;
            tracebackF = 0;
            continuousF = 0;
            sourceNode = 1;

            SetSamples(handles, 'initArrays', 'hide');
        case 'initArrays'
            DeadNodes = 0;
            trustOfHead = zeros(nodeCount,2);
            trustedNodes = zeros(nodeCount,1);
            ProcessesInit = 0;
            Neighbors = [];
            NeighboringHeads = [];
            NumGridHeads = 0;
            movesNodes= 0;
            moveDistances = 0;
            VacantGrid = [];
            gridHeads = zeros(nodeCount,1);
            FoundTrustedNode = 0;
            neighborTable = [];
            parent = [];
            numChildren = [];
            children = [];
            deadNodeList = [];
            deadNodes = 0;
            orphanNodes = 0;
            numTx = 0;
            numRx = 0;
            dropTx = 0;
            RXtoTXratio = RXCurDraw / TXCurDraw;
            receiver_threshold = (3 + sqrt(9 + 8/RXtoTXratio)) / 2; %
Calculate point at which hybrid sync'ing switches from RBS to TPSN
            nodePower = nodePowerMax .* ones(1, nodeCount); % Set the power
of each node to max
            gridPower = [1]; % Set the grid's initial normalized power to 1
(sum of all fully charged nodes)
            closestNode = sourceNode;
            PtWLow = 10 ^ (PtLow/10); % Convert to watts
            PrThresholdWLow = 10 ^ (PrThresholdLow/10); % Convert to watts
            maxDistanceLow = nthroot(PtWLow/PrThresholdWLow,
pathLossCoeff);
            PtWHigh = 10 ^ (PtHigh/10); % Convert to watts
            PrThresholdWHigh = 10 ^ (PrThresholdHigh/10); % Convert to
watts
```

```
            maxDistanceHigh = nthroot(PtWHigh/PrThresholdWHigh,
pathLossCoeff);

            if (nodeDistribMode == 'rand') % Random distribution

                 neighborTable = rand([nodeCount 2]);

                %%%%%%%%% HOW MANY GRIDS SO HOW MANY GRIDHEADS %%%%%%%%%%
                NumGridHeads = floor(xDistance / 20) * floor(yDistance /
20);
                if(NumGridHeads==0)
                    NumGridHeads = 1;
                end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %FOR GRIDHEADS
                xCount = floor(xDistance / 20);
                yCount = floor(yDistance / 20);

                dist = 20;
                num = 1;
                for j = 1 : yCount
                    for i = 1 : xCount
                        neighborTable(num,1) = i*dist - 0.5*dist;
                        neighborTable(num,2) = j*dist - 0.5*dist;
                        gridHeads(num)=num;
                        Neighbors(num)=0;
                        num = num + 1;

                    end
                end

              %%%%%%%%%%%%%%%% DIMIOURGIA GEITONWN %%%%%%%%%%%%%%%%%%%
               if(strcmp(COVandCONNAlg, 'AM'))
                 for i=1 : NumGridHeads

                     for j=1 : NumGridHeads
                         if(j~=i)
                             dist = CalcDistance(neighborTable(i,1),
neighborTable(i,2), neighborTable(j,1), neighborTable(j,2));

                             if(neighborTable(j,1) <=
(neighborTable(i,1)+20) && neighborTable(j,2)<=(neighborTable(i,2)+20) &&
neighborTable(j,1)>=(neighborTable(i,1)-20) &&
neighborTable(j,2)>=(neighborTable(i,2)-20))
                                 Neighbors(i) = Neighbors(i)+1;
                                 NeighboringHeads(i,Neighbors(i))=j;
                             end
                         end
                     end
                 end
               end

               %%%%%%%%%%%%%%% DIMIOURGIA HAMILTON CYCLE %%%%%%%%%%%%%%%%%%
                if(strcmp(COVandCONNAlg, 'SR'))
                   if(xCount>=2 || yCount>=2) % proipo8esi gia na ypar3ei
Hamilton Cycle
                    %HAMILTON CYCLE
```

```
                    if(rem(xCount,2)==0)     % The columns have even
number of GridHeads so we have a Hamilton cycle path
                        start = NumGridHeads-(xCount-1);
                        successors(start) =  start-xCount;
                        precedings(start) = start+1;
                        start=start-xCount;
                        for i=2: yCount-1      %pros ta katw
                            successors(start) =  start-xCount;
                            precedings(start) = start+xCount;
                            start=start-xCount;
                        end
                        successors(1) =  2;
                        precedings(1) = xCount+1;
                        for i=2: xCount-1      %PROS TA DE3IA
                            successors(i) = i+1;
                            precedings(i) = i-1;
                        end
                         successors(xCount) = 2*xCount;
                         precedings(xCount) = xCount-1;

                        start=2*xCount;
                        for i=2 : yCount-1 %pros ta panw
                            successors(start) = start+xCount;
                            precedings(start) = start-xCount;
                            start = start+xCount;
                        end

                        successors(NumGridHeads) = NumGridHeads-1;
                        precedings(NumGridHeads) = NumGridHeads-xCount;


                        if(xCount>2)   % yparxoun mesaies grammes
                         stop = (xCount/2)-1;
                         start = NumGridHeads-1;
                          for point=1 : stop
                               successors(start) = start-xCount;
                               precedings(start) = start+1;
                               start = start-xCount;
                               for i=1 : yCount-3   %pros ta katw
                               successors(start) =  start-xCount;
                               precedings(start) = start+xCount;
                               start = start-xCount;
                               end
                               successors(start) =  start-1;
                               precedings(start) = start+xCount;
                               successors(start-1) =  start-1+xCount;
                               precedings(start-1) = start;
                               start = start-1 + xCount;

                               for i=1 : yCount-3   %pros ta panw
                               successors(start) =  start+xCount;
                               precedings(start) = start-xCount;
                               start = start+xCount;
                               end
                               successors(start) =  start-1;
                               precedings(start) = start-xCount;
                               start = start-1;
                            end
```

```
                        end
                    elseif(rem(yCount,2)==0) % The rows have even number
of GridHeads so we have a Hamilton cycle path
                        successors(1) = 2;
                        precedings(1) = 1+xCount;

                        for i=2: xCount-1  %pros ta de3ia
                            successors(i) =  i+1;
                            precedings(i) = i-1;
                        end
                        successors(xCount) =  xCount*2;
                        precedings(xCount) = xCount-1;
                        start = 2*xCount;
                        for i=2: yCount-2      %pros ta panw
                            successors(start) = start+xCount;
                            precedings(start) = start-xCount;
                            start = start+xCount;
                        end

                        successors(NumGridHeads) =  NumGridHeads-1;
                        precedings(NumGridHeads) = NumGridHeads-xCount;
                        start = start-1;

                        for i=2 : xCount-2     %pros ta aristera

                            successors(start) = start-1;
                            precedings(start) = start+1;
                            start = start-1;
                        end

                        start=NumGridHeads-xCount+1;
                        successors(start) = start-xCount;
                        precedings(start) = start+1;


                        if(yCount>2)   % yparxoun mesaies grammes
                         stop = (yCount/2)-1;
                         start = start-xCount;

                           for point=1 : stop
                               successors(start) = start+1;
                               precedings(start) = start+xCount;

                               for i=1 : xCount-3   %pros ta de3ia
                                   start = start+1;
                               successors(start) =  start+1;
                               precedings(start) = start-1;
                               end

                               successors(start+1) = start+1-xCount;
                               precedings(start+1) = start;

                               start=start+1-xCount;
                               successors(start) =  start-1;
                               precedings(start) = start+xCount;
                               start = start-1;
```

```
                                for i=1 : xCount-3   %pros aristera
                                successors(start) =  start-1;
                                precedings(start) = start+1;
                                start = start-1;
                                end
                                successors(start) =  start-xCount;
                                precedings(start) = start+1;
                                start = start-xCount;
                            end

                        end


                    else
                        % The MxN network don't have even columns or rows
                        % so we have DUAL Hamilton cycle paths
                    end
                end
            end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
            %trustedNodes = zeros(NumGridHeads,1);

            for i = 1+NumGridHeads : nodeCount
                neighborTable(i,1) = neighborTable(i,1) * xDistance;
                neighborTable(i,2) = neighborTable(i,2) * yDistance;

               for j=1 : NumGridHeads
                   if(j==1)
                    if(neighborTable(i,1) <= 20 && neighborTable(i,2)<=
20)
                        gridHeads(i) = j; % belongs to gridHead j
                        trustedNodes(j) = trustedNodes(j) + 1;
                        break;
                    end

                   else
                       if(neighborTable(i,1) <= (neighborTable(j,1)+10)
&& neighborTable(i,2)<=(neighborTable(j,2)+10) &&
neighborTable(i,1)>=(neighborTable(j,1)-10) &&
neighborTable(i,2)>=(neighborTable(j,2)-10))
                          gridHeads(i) = j; % belongs to gridHead j
                          trustedNodes(j) = trustedNodes(j) + 1;
                          break;
                       end
                   end
               end
            end
        end


      if (nodeDistribMode == 'grid') % Uniform grid distribution
            dist = sqrt(xDistance*yDistance / nodeCount);
            xCount = floor(xDistance / dist);
            yCount = floor(yDistance / dist);
```

```
                if (xCount*yCount < nodeCount)
                    if (xDistance > yDistance)
                        xCount = xCount + 1;
                        if (xCount*yCount < nodeCount)
                            yCount = yCount + 1;
                        end
                    else
                        yCount = yCount + 1;
                        if (xCount*yCount < nodeCount)
                            xCount = xCount + 1;
                        end
                    end
                end
                dist = min(xDistance/xCount, yDistance/yCount);
                for j = 1 : yCount
                    for i = 1 : xCount
                        neighborTable = [neighborTable; i*dist - 0.5*dist
j*dist - 0.5*dist];
                    end
                end
          end


        case 'mica2dot433'
            moteClockSpd = 4;
            radioFreq = 433;
            TXCurDraw = 25;
            RXCurDraw = 8;
            PtLow = -2;
            PrThresholdLow = -85;
            PtHigh = -2;
            PrThresholdHigh = -101;
            guarDistance = 0;
        case 'mica2dot916'
            moteClockSpd = 4;
            radioFreq = 916;
            TXCurDraw = 27;
            RXCurDraw = 10;
            PtLow = -2;
            PrThresholdLow = -85;
            PtHigh = -2;
            PrThresholdHigh = -98;
            guarDistance = 0;
        case 'micaz'
            moteClockSpd = 4;
            radioFreq = 2400;
            TXCurDraw = 17.4;
            RXCurDraw = 19.7;
            PtLow = -5;
            PrThresholdLow = -85;
            PtHigh = -5;
            PrThresholdHigh = -94;
            guarDistance = 0;
        case 'load' % Leave variables as they are
        otherwise
    end

    if (visibility == 'show')
```

```
        set(handles.xDist, 'String', xDistance);
        set(handles.yDist, 'String', yDistance);
        set(handles.nodeCount, 'String', nodeCount);
        if (nodeDistribMode == 'grid')
            set(handles.gridMode, 'Value',1);
            set(handles.randMode, 'Value',0);
        elseif (nodeDistribMode == 'rand')
            set(handles.randMode, 'Value',1);
            set(handles.gridMode, 'Value',0);
        end

        set(handles.TxPwrLow, 'String',PtLow);
        set(handles.RxThreshLow, 'String',PrThresholdLow);
        set(handles.TxPwrHigh, 'String',PtHigh);
        set(handles.RxThreshHigh, 'String',PrThresholdHigh);
        set(handles.pathLossCoeff, 'String',pathLossCoeff);
        set(handles.guarDist, 'String',guarDistance);
        set(handles.pauseInt, 'String',pauseInt);
        set(handles.maxTxDistLow, 'String',maxDistanceLow);
        set(handles.maxTxDistHigh, 'String',maxDistanceHigh);

        set(handles.moteCPU, 'String',moteClockSpd);
        set(handles.radioFreq, 'String',radioFreq);
        set(handles.TXCurDraw, 'String',TXCurDraw);
        set(handles.RXCurDraw, 'String',RXCurDraw);
        set(handles.nodePowerMax, 'String',nodePowerMax);
        set(handles.nodePowerWarn, 'String',nodePowerWarn);
        set(handles.rediscoverLimit, 'String',rediscoverLimit);
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Plot Grid                                      %
%             Coverage & Connectivity Control in WSNs                   %
%                          Kakoulli Elena                               %
%                     Computer Science, UCY                             %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function PlotGrid()
% Plot each of the nodes according to their role and status in the network
    GlobalVars();

    for i=1 : NumGridHeads
         % s = sprintf('%d', i);

            %nodeColor = checkNodePwrLevel(i, 'b');
            if (viewNodeIDF)
                plot(neighborTable(i,1), neighborTable(i,2), '.',
'MarkerEdgeColor','b', 'LineWidth',2);
                text(neighborTable(i,1) + xDistance/100,
neighborTable(i,2), s, 'FontSize',8, 'FontWeight','bold',
'HorizontalAlignment','left', 'Color',nodeColor);
            else
                plot(neighborTable(i,1), neighborTable(i,2), '^',
'MarkerEdgeColor','b', 'LineWidth',2);
```

```
            end
    end

    for i = 1+NumGridHeads : nodeCount
        %s = sprintf('%d', i);

       if(gridHeads(i)>0) %den metakini8ike allou
            nodeColor = checkNodePwrLevel(i, 'k');
            %if(gridHeads(i)==1)
            %    plot(neighborTable(i,1), neighborTable(i,2), 'x',
'MarkerEdgeColor','r', 'LineWidth',4);
            %else
                if (viewNodeIDF)
                plot(neighborTable(i,1), neighborTable(i,2), '.',
'MarkerEdgeColor',nodeColor, 'LineWidth',2);
                text(neighborTable(i,1) + xDistance/100,
neighborTable(i,2), s, 'FontSize',8, 'HorizontalAlignment','left',
'Color',nodeColor);
                else
                plot(neighborTable(i,1), neighborTable(i,2), 'o',
'MarkerEdgeColor',nodeColor, 'LineWidth',2);
                end
       end
    end
end

function [color] = checkNodePwrLevel(curNode, stdColor)
% Change the node's color if its power level has decreased below the
warning level
    GlobalVars();
    if (nodePower(curNode) > nodePowerWarn)
        color = stdColor;
    elseif (nodePower(curNode) <= nodePowerWarn && nodePower(curNode) > 0)
        color = 'y'; % Yellow for warning
    else
        color = 'r'; % Red for depleted
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Edit Parameters                                 %
%            Coverage & Connectivity Control in WSNs                    %
%                         Kakoulli Elena                                %
%                    Computer Science, UCY                              %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = ParamEdit(varargin)
    gui_Singleton = 1;
    gui_State = struct('gui_Name', mfilename, 'gui_Singleton',
gui_Singleton, 'gui_OpeningFcn', @ParamEdit_OpeningFcn, 'gui_OutputFcn',
@ParamEdit_OutputFcn, 'gui_LayoutFcn', [], 'gui_Callback', []);
    if (nargin && ischar(varargin{1}))
        gui_State.gui_Callback = str2func(varargin{1});
    end
    if (nargout)
```

```
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
end % End initialization code

% --- Executes just before the editor is made visible.
function ParamEdit_OpeningFcn(hObject, eventdata, handles, varargin)
    GlobalVars();
    handles.output = hObject; guidata(hObject, handles);
    SetSamples(handles, 'load', 'show');
end

% --- Outputs from this function are returned to the command line.
function varargout = ParamEdit_OutputFcn(hObject, eventdata, handles)
    varargout{1} = handles.output;
end

function gridParams_Callback(hObject, eventdata, handles)
    GlobalVars();
    if (str2double(get(handles.xDist, 'String')) == 0)
        set(hObject, 'String', xDistance);
    end
    if (str2double(get(handles.yDist, 'String')) == 0)
        set(hObject, 'String', yDistance);
    end
    if (str2double(get(handles.nodeCount, 'String')) == 0)
        set(hObject, 'String', nodeCount);
    end

    if (str2double(get(handles.xDist, 'String')) ~= xDistance || ...
            str2double(get(handles.yDist, 'String')) ~= yDistance || ...
            str2double(get(handles.nodeCount, 'String')) ~= nodeCount)
        networkChangedF = 1;
    end
end

function gridMode_Callback(hObject, eventdata, handles)
    GlobalVars();
    set(handles.gridMode, 'Value', 1);
    set(handles.randMode, 'Value', 0);
end

function randMode_Callback(hObject, eventdata, handles)
    GlobalVars();
    set(handles.gridMode, 'Value', 0);
    set(handles.randMode, 'Value', 1);
end

function signalParams_Callback(hObject, eventdata, handles)
    GlobalVars();
    pathLossCoeffTemp = str2double(get(handles.pathLossCoeff, 'String'));
    PtLTemp = str2double(get(handles.TxPwrLow, 'String'));
    PtWLTemp = 10^(PtLTemp/10) / 1000; % Convert to watts
    PrThresholdLTemp = str2double(get(handles.RxThreshLow, 'String'));
    PrThresholdWLTemp = 10^(PrThresholdLTemp/10) / 1000; % Convert to watts
    set(handles.maxTxDistLow, 'String', nthroot(PtWLTemp/PrThresholdWLTemp,
pathLossCoeffTemp));
```

```
    PtHTemp = str2double(get(handles.TxPwrHigh, 'String'));
    PtWHTemp = 10^(PtHTemp/10) / 1000; % Convert to watts
    PrThresholdHTemp = str2double(get(handles.RxThreshHigh, 'String'));
    PrThresholdWHTemp = 10^(PrThresholdHTemp/10) / 1000; % Convert to watts
    set(handles.maxTxDistHigh, 'String',
nthroot(PtWHTemp/PrThresholdWHTemp, pathLossCoeffTemp));
end

function resetParams_Callback(hObject, eventdata, handles)
    GlobalVars();
    SetSamples(handles, 'mica2dot433', 'show');
    close;
end

function cancelParams_Callback(hObject, eventdata, handles)
    networkChangedF = 0;
    close;
end

function saveParams_Callback(hObject, eventdata, handles)
    GlobalVars();
    if (xDistance ~= str2double(get(handles.xDist, 'String')) ...
            || yDistance ~= str2double(get(handles.yDist, 'String')) ...
            || nodeCount ~= str2double(get(handles.nodeCount, 'String')))
        networkChangedF = 1;
    else
        networkChangedF = 0;
    end

    xDistance = str2double(get(handles.xDist, 'String'));
    yDistance = str2double(get(handles.yDist, 'String'));
    nodeCount = str2double(get(handles.nodeCount, 'String'));
    if (get(handles.gridMode, 'Value'))
        nodeDistribMode = 'grid';
    else
        nodeDistribMode = 'rand';
    end
    PtLow = str2double(get(handles.TxPwrLow, 'String'));
    PtWLow = 10^(PtLow/10) / 1000;
    PrThresholdLow = str2double(get(handles.RxThreshLow, 'String'));
    PrThresholdWLow = 10^(PrThresholdLow/10) / 1000;
    PtHigh = str2double(get(handles.TxPwrHigh, 'String'));
    PtWHigh = 10^(PtHigh/10) / 1000;
    PrThresholdHigh = str2double(get(handles.RxThreshHigh, 'String'));
    PrThresholdWHigh = 10^(PrThresholdHigh/10) / 1000;
    pathLossCoeff = str2double(get(handles.pathLossCoeff, 'String'));
    maxDistanceLow = nthroot(PtWLow/PrThresholdWLow, pathLossCoeff);
    maxDistanceHigh = nthroot(PtWHigh/PrThresholdWHigh, pathLossCoeff);
    guarDistance = str2double(get(handles.guarDist, 'String'));
    pauseInt = str2double(get(handles.pauseInt, 'String'));
    moteClockSpd = str2double(get(handles.moteCPU, 'String'));
    radioFreq = str2double(get(handles.radioFreq, 'String'));

    TXCurDraw = str2double(get(handles.TXCurDraw, 'String'));
    RXCurDraw = str2double(get(handles.RXCurDraw, 'String'));
    RXtoTXratio = RXCurDraw / TXCurDraw;
```

```
    receiver_threshold = (3 + sqrt(9 + 8/RXtoTXratio)) / 2; % Calculate
point at which hybrid sync'ing switches from RBS to TPSN

    nodePowerMax = str2double(get(handles.nodePowerMax, 'String'));
    nodePowerWarn = str2double(get(handles.nodePowerWarn, 'String'));
    rediscoverLimit = str2double(get(handles.rediscoverLimit, 'String'));
    close;
end

% ------------------- SAMPLE FUNCTIONS -------------------
function Menu_mica2dot433_Callback(hObject, eventdata, handles)
    SetSamples(handles, 'mica2dot433', 'show');
    SetSamples(handles, 'initArrays', 'hide');
end

function Menu_mica2dot916_Callback(hObject, eventdata, handles)
    SetSamples(handles, 'mica2dot916', 'show');
    SetSamples(handles, 'initArrays', 'hide');
end

function Menu_micaz_Callback(hObject, eventdata, handles)
    SetSamples(handles, 'micaz', 'show');
    SetSamples(handles, 'initArrays', 'hide');
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                           Is Received                                 %
%            Coverage & Connectivity Control in WSNs                    %
%                       Kakoulli Elena                                  %
%                   Computer Science, UCY                               %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [r] = IsReceived(dist, packetType)
% Return 1 for successful TX or 0 for failed TX
    GlobalVars();
    r = 0;
    if (strcmp(packetType, 'topolDiscovery'))
        maxDist = maxDistanceLow;
    else
        maxDist = maxDistanceHigh;
    end
    gDist = min(guarDistance, maxDist); % Guaranteed TX distance

    if (dist <= gDist)
        r = 1;
    elseif (dist <= maxDist)
        prob = 1 - (dist - gDist) / (maxDist - gDist);
        if (rand() < prob)
            r = 1;
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Global Variables                                %
```

```
%             Coverage & Connectivity Control in WSNs      %
%                        Kakoulli Elena                    %
%                    Computer Science, UCY                 %
%                                                          %
%                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global verDate; % Version number for the simulator
global propSpd; % Speed of light (in m/s)

% Grid Parameters
global xDistance; % Length of x-axis on grid
global yDistance; % Length of y-axis on grid
global nodeCount; % Number of nodes in the grid
global nodeDistribMode; % Type of grid distribution for nodes (grid or
random)

% Hardware Parameters
global moteClockSpd; % Speed of mote's CPU (in MHz)
global radioFreq; % Throughput of mote's transceiver (in Kbps)
global TXCurDraw; % Current draw for a transmission (in mA)
global RXCurDraw; % Current draw for a reception (in mA)
global receiver_threshold; % Number of receivers needed where HTC becomes
more efficient than APC
global nodePowerMax; % Max power for each node
global nodePowerWarn; % Power where node shows low-power warning
global nodePower; % [1 x nodeCount] array holding power for each node
global nodeBuffer;% [1 x nodeBuffer] array holding buffer for each node
global gridPower; % [1 x *] array holding the grid's total power (index is
number of energy-consuming events)

% discover Parameters
global PtLow; % Transmission power (in dB) for discover packets
global PtWLow; % Transmission power (in W) for discover packets
global PrThresholdLow; % Threshold power (in dB) for discover packets
global PrThresholdWLow; % Threshold power (in W) for discover packets
global PtHigh; % Transmission power (in dB) for sync packets
global PtWHigh; % Transmission power (in W) for sync packets
global PrThresholdHigh; % Threshold power (in dB) for sync packets
global PrThresholdWHigh; % Threshold power (in W) for sync packets
global pathLossCoeff; % Path loss coefficient for transmission equation
global guarDistance; % Guaranteed distance for successful transmission
global maxDistanceLow; % Maximum distance nodes are capable of transmitting
a discover packet
global maxDistanceHigh; % Maximum distance nodes are capable of
transmitting a sync packet
global RXtoTXratio; % Ratio of reception power to transmission power
global pauseInt; % Length of time between plotting generations

% topolDiscovery Variables
global neighborTable; % nodeCount x nodeCount array holding x and y
coordinates for each node
global nodeLevel; % [1 x nodeCount] array showing the generation of a given
node (-1 means it is an orphan)
global maxGens; % Maximum number of generations possible in the
topolDiscovery
global sourceNode; % Root node for transmissions
global closestNode; % Closest node user clicked to
```

```
global discoverTime; % Time used to topolDiscovery the network

% Coverage or Connectivity Variables
global COVandCONNAlg;   % Type of coverage or connectivity Algorithm being
used
global numTx;           % Number of transmissions used to perform
congestionAlgorithm
global numRx;           % Number of receptions used to perform
congestionAlgorithm
global dropTx;          % Number of transmissions used to perform
congestionAlgorithm
global rediscoverLimit; % Percentage of dead nodes at which network must
rediscover
global source_sinkTime; % The total time
global DeadNodes;

% Grid Heads Variables & flags
global gridHeads;     % periexei gia ka8e node, diladi ka8e 8esi tou
antistoixei se ka8e node kai to periexomeno einai o gridHead pou anikei
global VacantGrid;    % periexei to adeio grid
global NoVacant;      % flag when the grid is no vacant
global FoundTrustedNode; %periexei ton node pou 8a kini8ei
global NumGridHeads;  % o ari8mos twn gridHeads
global NeighboringHeads;  % [NumGridHeads x NumGridHeads] array showing
each node's children
global Neighbors;     % o ka8e gridHead me ton ari8mo geitonikwn gridHeads
tou
global movesNodes;    % o ari8mos twn nodes pou kini8ikan
global moveDistances; % h synoliki apostasi kinisis twn nodes
global ProcessesInit; % ta processes pou ginontai initiated
global trustedNodes;  % pinakas ka8e 8esi tou opoiou einai o ari8mos twn
trusted nodes ka8e gridHead
global trustOfHead;   % pinakas pou periexei olous tous gridHead me tous
trusted Nodes tous
global NodesMove;     % Oi komboi pou kini8ikan kata tin diarkeia twn
algori8mwn

% FOR SR Algorithm
global successors;  %[NumGridHeads 2]
global precedings; %[NumGridHeads 2]


% Flags
global dataLoadedF; % =1 when the parameters are loaded from a .mat file
global nodesDispersedF; % =1 when the nodes have been plotted
global networkChangedF; % =1 when the xDist, yDist, or nodeCount are
changed and the plot has not been updated
global continuousF; % =1 when the "Continuous" checkbox is selected
global changeSourceNodeF; % =1 when the "Change Source" checkbox is
selected
global tracebackF; % =1 when the "Trace back to Root" checkbox is selected
global viewNodeIDF; % =1 when the "View Node IDs" menu item is selected
global networkLvlDiscF; % =1 when the network has done the level discovery
(discovering)
global networkSyncF; % =1 when the network has been completely synchronized
global congestedF; %=1 when the node is congested
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Draw Topology                                  %
%             Coverage & Connectivity Control in WSNs                   %
%                          Kakoulli Elena                               %
%                    Computer Science, UCY                              %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function DrawTopology(delay)
% Draw the discovering lines
    GlobalVars;
         for j = 1 : nodeCount
            UpdateNodePower();
            x1 = neighborTable(gridHeads(j),1);
            y1 = neighborTable(gridHeads(j),2);
            x2 = neighborTable(j,1);
            y2 = neighborTable(j,2);
            line([x1 x2], [y1 y2], 'Color', 'g', 'LineWidth', 2);
         end

        if (delay ~= 0)
            pause(delay);
        end

end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Coverage & Connectivity Algorithm SR                     %
%             Coverage & Connectivity Control in WSNs                   %
%                          Kakoulli Elena                               %
%                    Computer Science, UCY                              %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function COVandCONNAlgorithmSR()
% Function implements the SR algorithm for coverage and connectivity
control

    GlobalVars();
    VacantGrid = zeros(NumGridHeads);
    finished = 0;    %flag for end of algorithm
    gridHeads(3)=0;  % for sample - this must be change later

    while(finished==0)
         GridHeadsCommunicSR();

        for i=1 : NumGridHeads
            if(VacantGrid(i)~=0)
                finished = 0;
                break;
            else
                finished = 1;
```

```
            end
        end

        if(finished==0)
            CascadingMovementSR();

        end

    end

    %% For GridHeads connection
    for i=1 : NumGridHeads
        if(VacantGrid(i)==0)
                    x1 = neighborTable(i,1);
                    y1 = neighborTable(i,2);
                    x2 = neighborTable(successors(i),1)
                    y2 = neighborTable(successors(i),2);
                    line([x1 x2], [y1 y2], 'Color', 'cyan', 'LineWidth',
2);
        end
    end
      UpdateNodePower();
end

%% STEP 1: Oloi oi gridHeads elegxoun ean yparxei connection meta3y tous
%% alliws yparxei vacant grid kai 8a 3ekinisei cascading movement apo ton
%% 1o gridHead pou to anixneyse kai 8a enimerwsei tous ypoloipous me ena
%% notification kai to STEP 1 8a ginete epanaliptika efoson den pairnei
%% kanenas kapoio notification msg

function GridHeadsCommunicSR()

    GlobalVars();
    NoVacant = -1;
    RandomHeads = randperm(NumGridHeads);

    for i=1 : NumGridHeads     % o ka8e gridHead elegxei

       if(gridHeads(RandomHeads(i))~=0 && successors(RandomHeads(i))~=0)
        dist = CalcDistance(neighborTable(RandomHeads(i),1),
neighborTable(RandomHeads(i),2),
neighborTable(successors(RandomHeads(i)),1),
neighborTable(successors(RandomHeads(i)),2));

            % sensing, so lose power
            nodePower(RandomHeads(i)) = nodePower(RandomHeads(i)) -
(dist/4);

        if(dist~=0)
                PrWLow = PtWLow / (dist ^ pathLossCoeff); % Calculate
reception power
            end
            %IsReceived(dist, 'topolDiscovery') == 0

            if(gridHeads(successors(RandomHeads(i)))==0)
                NoVacant = 0;
```

```
            for k=5 : nodeCount       %elegxos ean einai vacant to grid
                   if(gridHeads(k)==successors(RandomHeads(i)))
                       NoVacant = 1;
                       break;
                   end
            end

           end
       end

         if(NoVacant==0)         %bre8ike vacant grid
            VacantGrid(RandomHeads(i)) = successors(RandomHeads(i));
            ProcessesInit = ProcessesInit + 1;
            break;
         else
            VacantGrid(RandomHeads(i)) = 0;
         end


        NoVacant = -1;

    end

end

%% STEP 2: Oi gridHeads pou exo un anixneysei kapoio vacant grid 8a prepei
%% na psaxoun na broun kapoio trusted node gia na metakini8ei sto vacant
grid
%% kai na ginei aytos o gridHead tou vacant grid, alliws an den bre8ei na
steiloun
%% ena notification msg stous geitonikous gridHeads gia na arxikopoihsoun
%% kai oi idioi mia tetoia diadikasia kai na metakini8ei o idios sto vacant
grid
%% kai na ginei se ayto o gridHead

function CascadingMovementSR();
    GlobalVars();
    RandomHeads = randperm(NumGridHeads);

    for i=1 : NumGridHeads
        if(VacantGrid(RandomHeads(i))>0)

            movesNodes = movesNodes + 1; %ay3anw tous kombous pou kini8ikan

            x1 = neighborTable(RandomHeads(i),1);
            y1 = neighborTable(RandomHeads(i),2);
            x2 = neighborTable(VacantGrid(RandomHeads(i)),1);
            y2 = neighborTable(VacantGrid(RandomHeads(i)),2);
            plot(neighborTable(VacantGrid(RandomHeads(i)),1),
neighborTable(VacantGrid(RandomHeads(i)),2), 'x', 'MarkerEdgeColor','r',
'LineWidth',10);
            drawLineStatus(x1, x2, y1, y2, 'NoConnection');



          for j=1+NumGridHeads : nodeCount  %elegxos ean yparxei trusted
node mesa sto grid pou einai arxigos
              FoundTrustedNode = RandomHeads(i);
```

```
            if(gridHeads(j)==RandomHeads(i))
               FoundTrustedNode = j;
               break;
            end
         end


          % ypologismos apostasis pou metakineitai o trusted node 'i o
gridHead
            dist = CalcDistance(neighborTable(FoundTrustedNode,1),
neighborTable(FoundTrustedNode,2),
neighborTable(VacantGrid(RandomHeads(i)),1),
neighborTable(VacantGrid(RandomHeads(i)),2));
            moveDistances = moveDistances + dist; %ay3anw tin synoliki
apostasis metakinisis pou egine
            NodesMove(movesNodes) = FoundTrustedNode;

            if(dist~=0)
              PrWLow = PtWLow / (dist ^ pathLossCoeff); % Calculate
reception power
            end

            % moving, so lose power
            nodePower(FoundTrustedNode) = nodePower(FoundTrustedNode) -
(dist/2);


         if(FoundTrustedNode == RandomHeads(i)) % ean den bre8ike trusted
node
            %enimerwsi tou geitonikou gridHead precending
            NotificationSR(RandomHeads(i));

            %move and change gridhead
            drawLineStatus(x1, x2, y1, y2, 'CascadingMovement');
            %neighborTable(VacantGrid(i),1) =
            %neighborTable(VacantGrid(i),2) =

gridHeads(VacantGrid(RandomHeads(i)))=VacantGrid(RandomHeads(i));
            gridHeads(FoundTrustedNode)=0;

         else   % tote bre8ike trusted node k prepei na metakini8ei sto
vacant grid k na ginei o gridHead tou
            x1 = neighborTable(FoundTrustedNode,1);
            y1 = neighborTable(FoundTrustedNode,2);
            x2 = neighborTable(VacantGrid(RandomHeads(i)),1);
            y2 = neighborTable(VacantGrid(RandomHeads(i)),2);
            drawLineStatus(x1, x2, y1, y2, 'CascadingMovement');

            %metakinisi tou trusted node sto vacant grid kai ginete o
            %gridHead tou vacant grid
            neighborTable(FoundTrustedNode,1) =
neighborTable(VacantGrid(RandomHeads(i)),1);
            neighborTable(FoundTrustedNode,1) =
neighborTable(VacantGrid(RandomHeads(i)),2);
            gridHeads(FoundTrustedNode)=VacantGrid(RandomHeads(i));
            gridHeads(VacantGrid(RandomHeads(i)))=
VacantGrid(RandomHeads(i));
            %gridHeads(VacantGrid(i))=FoundTrustedNode;
```

```
            %gridHeads(FoundTrustedNode)=0;
            %....enimerwsi pws einai o new gridHead tou vacant grid

          end

           plot(neighborTable(VacantGrid(RandomHeads(i)),1),
neighborTable(VacantGrid(RandomHeads(i)),2), '^', 'MarkerEdgeColor','b',
'LineWidth',2);

        end
    end

end

function NotificationSR(gridHead)
% Send notification msg from the gridHead to its precending gridHead
% gridHead to inform it for his cascading movement to the vacant grid

    GlobalVars();

        if (precedings(gridHead)~=VacantGrid(gridHead) &&
nodePower(gridHead) >= 1 && nodePower(precedings(gridHead)) > 0)

            x1 = neighborTable(gridHead,1);
            y1 = neighborTable(gridHead,2);
            x2 = neighborTable(precedings(gridHead),1);
            y2 = neighborTable(precedings(gridHead),2);

            drawLineStatus(x1, x2, y1, y2, 'MsgNotification');
            numTx = numTx + 1;
            nodePower(gridHead) = nodePower(gridHead) - 1;
            numRx = numRx + 1;
            nodePower(precedings(gridHead)) =
nodePower(precedings(gridHead)) - RXtoTXratio;

        end

end



function drawLineStatus(startX, endX, startY, endY, type)
    GlobalVars();
    switch (type)
        case 'MsgNotification'
            lineColor = 'b';
            style = '-';
        case 'CascadingMovement'
            lineColor = 'red';
            style = '--';
        case 'NoConnection'
            lineColor = 'm';
            style = '--';
        otherwise
    end
    line([startX endX], [startY endY], 'Color',lineColor, 'LineWidth',2,
'LineStyle',style);
    pause(0.8);
```

```
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Coverage & Connectivity Algorithm AM       %
%            Coverage & Connectivity Control in WSNs      %
%                       Kakoulli Elena                    %
%                    Computer Science, UCY                %
%                                                         %
%                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Control scheme under active model:
% 1. For any grid head u that detects a vacant neighboring
% grid, if no notification is received in the previous round
% from that area (to avoid overreaction), a replacement
% process is initiated after the corresponding region area is determined.
% 2. For any grid head u that has started the replacement
% process or that is notified in the cascading replacement
% process, find one of neighboring spare trusted nodes
% in its grid, say node v, to move into that neighboring
% vacant grid before the next round starts.
% 3. If such a node v cannot be found, select any neighboring
% grid that is other than the vacant one but still in
% region. Then, send out the notification for the replacement
% of u, attaching the information of region and
% such a selection. It will always select the one with
% spare trusted node(s) first. After that, move u to the
% vacant grid before the next round starts.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function COVandCONNAlgorithmAM()
% Implements coverage and connectivity control algorithm Active Model

    GlobalVars();
    VacantGrid = zeros(NumGridHeads);
    finished = 0;    %flag for end of algorithm
    gridHeads(3)=0;  %for sample - this must be change later

    while(finished==0)
         GridHeadsCommunicAM();

        for i=1 : NumGridHeads
            if(VacantGrid(i)~=0)
                finished = 0;
                break;
            else
                finished = 1;
            end
        end

        if(finished==0)
            CascadingMovementAM();
```

```
        end

    end

    %% For GridHeads connection
    for i=1 : NumGridHeads
        if(VacantGrid(i)==0)
            for j = 1 : Neighbors(i)
                    x1 = neighborTable(i,1);
                    y1 = neighborTable(i,2);
                    x2 = neighborTable(NeighboringHeads(i,j),1);
                    y2 = neighborTable(NeighboringHeads(i,j),2);
                    line([x1 x2], [y1 y2], 'Color', 'cyan', 'LineWidth',
2);
            end
        end
    end

    UpdateNodePower();
end

%% STEP 1: Oloi oi gridHeads elegxoun ean yparxei connection meta3y tous
%% alliws yparxei vacant grid kai 8a 3ekinisei cascading movement apo ton
%% 1o gridHead pou to anixneyse kai 8a enimerwsei tous ypoloipous me ena
%% notification kai to STEP 1 8a ginete epanaliptika efoson den pairnei
%% kanenas kapoio notification msg

function GridHeadsCommunicAM()

    GlobalVars();
    NoVacant = -1;
    RandomHeads = randperm(NumGridHeads);

    for i=1 : NumGridHeads     % o ka8e gridHead elegxei

          NoVacant = -1;

       if(gridHeads(RandomHeads(i))~=0)
        for j=1 : Neighbors(RandomHeads(i)) % ean yparxei connection me ton
ka8e geitoniko tou gridHead

            dist = CalcDistance(neighborTable(RandomHeads(i),1),
neighborTable(RandomHeads(i),2),
neighborTable(NeighboringHeads(RandomHeads(i),j),1),
neighborTable(NeighboringHeads(RandomHeads(i),j),2));
            %sensing, so lose power
            nodePower(RandomHeads(i)) = nodePower(RandomHeads(i)) -
(dist/4);

         if(VacantGrid(NeighboringHeads(RandomHeads(i),j))~=RandomHeads(i))

            if(dist~=0)
                PrWLow = PtWLow / (dist ^ pathLossCoeff); % Calculate
reception power
            end
            %IsReceived(dist, 'topolDiscovery') == 0

            if(gridHeads(NeighboringHeads(RandomHeads(i),j))==0)
```

```
                NoVacant = 0;

             for k=5 : nodeCount       %elegxos ean einai vacant to grid
                    if(gridHeads(k)==NeighboringHeads(RandomHeads(i),j))
                        NoVacant = 1;
                        break;
                    end
             end

            end

         end

         if(NoVacant==0)         %bre8ike vacant grid
            VacantGrid(RandomHeads(i)) =
NeighboringHeads(RandomHeads(i),j);
            ProcessesInit = ProcessesInit + 1;
            break;
         else
            VacantGrid(RandomHeads(i)) = 0;
         end

        end
       end
    end
end

%% STEP 2: Oi gridHeads pou exo un anixneysei kapoio vacant grid 8a prepei
%% na psaxoun na broun kapoio trusted node gia na metakini8ei sto vacant
grid
%% kai na ginei aytos o gridHead tou vacant grid, alliws an den bre8ei na
steiloun
%% ena notification msg se ena geitona gridHead kai na metakini8ei o idios
sto vacant grid
%% kai na ginei se ayto o gridHead

function CascadingMovementAM();
    GlobalVars();
    RandomHeads = randperm(NumGridHeads);


    for i=1 : NumGridHeads
        if(VacantGrid(RandomHeads(i))>0 && gridHeads(RandomHeads(i))~=0)

            movesNodes = movesNodes + 1; %ay3anw tous kombous pou kini8ikan

            x1 = neighborTable(RandomHeads(i),1);
            y1 = neighborTable(RandomHeads(i),2);
            x2 = neighborTable(VacantGrid(RandomHeads(i)),1);
            y2 = neighborTable(VacantGrid(RandomHeads(i)),2);
            plot(neighborTable(VacantGrid(RandomHeads(i)),1),
neighborTable(VacantGrid(RandomHeads(i)),2), 'x', 'MarkerEdgeColor','r',
'LineWidth',10);
            drawLineStatus(x1, x2, y1, y2, 'NoConnection');


          if(trustedNodes(RandomHeads(i))>0) %elegxos ean yparxei trusted
node mesa sto grid pou einai arxigos
```

```
             for j=1+NumGridHeads : nodeCount  %briskw kapoion apo ta
trusted nodes tou
              if(gridHeads(j)==RandomHeads(i))
                 FoundTrustedNode = j;
                 break;
              end
             end
          else
              FoundTrustedNode = RandomHeads(i);
              %enimerwsi enos apo tous geitonikoys gridHeads me ta
              %perissotera trusted nodes
              NotificationAM(RandomHeads(i));
          end


           % ypologismos apostasis pou metakineitai o trusted node 'i o
gridHead
              dist = CalcDistance(neighborTable(FoundTrustedNode,1),
neighborTable(FoundTrustedNode,2),
neighborTable(VacantGrid(RandomHeads(i)),1),
neighborTable(VacantGrid(RandomHeads(i)),2));
              moveDistances = moveDistances + dist; %ay3anw tin synoliki
apostasis metakinisis pou egine

              NodesMove(movesNodes) = FoundTrustedNode;

              if(dist~=0)
                PrWLow = PtWLow / (dist ^ pathLossCoeff); % Calculate
reception power
              end


              % moving, so lose power
              nodePower(FoundTrustedNode) = nodePower(FoundTrustedNode)-
(dist/2);


          if(FoundTrustedNode == RandomHeads(i)) % ean den bre8ike trusted
node



           %for w=1 :4
            %  if(RandomHeads(w)~=i)
             %   if(VacantGrid(RandomHeads(w))==0)
              %   VacantGrid(RandomHeads(w))=i;
               %  CascadingMovementAM();
                % break;
                %end
              %end
           %end

              %move and change gridhead
              drawLineStatus(x1, x2, y1, y2, 'CascadingMovement');
              %neighborTable(VacantGrid(i),1) =
              %neighborTable(VacantGrid(i),2) =

gridHeads(VacantGrid(RandomHeads(i)))=VacantGrid(RandomHeads(i));
```

```
              gridHeads(FoundTrustedNode)=0;

          else   % tote bre8ike trusted node k prepei na metakini8ei sto
vacant grid k na ginei o gridHead tou
              x1 = neighborTable(FoundTrustedNode,1);
              y1 = neighborTable(FoundTrustedNode,2);
              x2 = neighborTable(VacantGrid(RandomHeads(i)),1);
              y2 = neighborTable(VacantGrid(RandomHeads(i)),2);
              drawLineStatus(x1, x2, y1, y2, 'CascadingMovement');

              %metakinisi tou trusted node sto vacant grid kai ginete o
              %gridHead tou vacant grid
              neighborTable(FoundTrustedNode,1) =
neighborTable(VacantGrid(RandomHeads(i)),1);
              neighborTable(FoundTrustedNode,1) =
neighborTable(VacantGrid(RandomHeads(i)),2);
              gridHeads(FoundTrustedNode)=VacantGrid(RandomHeads(i));
              gridHeads(VacantGrid(RandomHeads(i)))=
VacantGrid(RandomHeads(i));
              trustedNodes(RandomHeads(i)) = trustedNodes(RandomHeads(i)) -
1; %feygei o trusted node kai ara meiwnete kai o ari8mos twn trusted nodes
toy gridHead
              %gridHeads(VacantGrid(i))=FoundTrustedNode;
              %gridHeads(FoundTrustedNode)=0;
              %....enimerwsi pws einai o new gridHead tou vacant grid


              %plotGrid();

          end

           plot(neighborTable(VacantGrid(RandomHeads(i)),1),
neighborTable(VacantGrid(RandomHeads(i)),2), '^', 'MarkerEdgeColor','b',
'LineWidth',2);

        end
          VacantGrid(RandomHeads(i)) = 0;
    end


end


function NotificationAM(gridHead)
% Send notification msg from the gridHead to one of its neighbor's
% gridHeads, that with more trusted nodes, to inform it for his cascading
movement to the vacant
% gridHead

    GlobalVars();
    tnodes = 0;
    headChoose = 0;

    for h = 1 : Neighbors(gridHead)

        if(NeighboringHeads(gridHead,h)~=VacantGrid(gridHead) &&
nodePower(gridHead) >= 1 && nodePower(NeighboringHeads(gridHead,h)) > 0 &&
tnodes<trustedNodes(NeighboringHeads(gridHead,h)))
```

```
            tnodes = trustedNodes(NeighboringHeads(gridHead,h));
            headChoose = NeighboringHeads(gridHead,h);
        end

    end

    if(headChoose~=0)
    x1 = neighborTable(gridHead,1);
    y1 = neighborTable(gridHead,2);
    x2 = neighborTable(headChoose,1);
    y2 = neighborTable(headChoose,2);

    % for power
    numTx = numTx + 1;
    nodePower(gridHead) = nodePower(gridHead) - 1;
    numRx = numRx + 1;
    nodePower(h) = nodePower(h) - RXtoTXratio;

    drawLineStatus(x1, x2, y1, y2, 'MsgNotification');
    VacantGrid(headChoose) = gridHead;
    ProcessesInit = ProcessesInit + 1;


    end

end

function drawLineStatus(startX, endX, startY, endY, type)
    GlobalVars();
    switch (type)
        case 'MsgNotification'
            lineColor = 'b';
            style = '-';
        case 'CascadingMovement'
            lineColor = 'red';
            style = '--';
        case 'NoConnection'
            lineColor = 'm';
            style = '--';
        otherwise
    end
    line([startX endX], [startY endY], 'Color',lineColor, 'LineWidth',2,
'LineStyle',style);
    pause(0.8);
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Clear Handles                                   %
%            Coverage & Connectivity Control in WSNs                    %
%                        Kakoulli Elena                                 %
%                   Computer Science, UCY                               %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ClearHandles(handles)
```

```
% Auxiliary function which clear the handles of the main function
    GlobalVars();
    set(handles.analyzedNode, 'String','');
    set(handles.GridHead, 'String','');
    set(handles.NodesMove, 'String','');
    set(handles.pwrRem, 'String','');
    set(handles.discoverTime, 'String','');
    set(handles.source_sinkTime, 'String','');
    set(handles.numRX, 'String','');
    set(handles.droppedTX, 'String','');
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Calculate Distance                                  %
%             Coverage & Connectivity Control in WSNs                   %
%                        Kakoulli Elena                                 %
%                    Computer Science, UCY                              %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [d] = dist(x1, y1, x2, y2)
% Function for calculating the distance between two nodes
    d = sqrt(double((x1-x2)^2 + (y1-y2)^2));
end
```

Παράρτημα Β

Κώδικας υλοποίησης των αλγορίθμων AM και SR που κατά την αρχική τυχαία τοποθέτηση όλων των αισθητήρων γίνεται μετά μετακίνηση ενός τυχαίου αισθητήρα μέσα σε κάθε πλαίσιο ως αρχηγός κάθε πλαισίου στο κέντρο του κάθε πλαισίου

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        WSN_COVandCONN_Sim                             %
%             Coverage & Connectivity Control in WSNs                   %
%                        Kakoulli Elena                                 %
```

```
%                   Computer Science, UCY                   %
%                                                           %
%                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = WSN_COVandCONN_Sim(varargin)
    gui_Singleton = 1;
    gui_State = struct('gui_Name', mfilename, 'gui_Singleton',
gui_Singleton, 'gui_OpeningFcn', @WSN_COVandCONN_Sim_OpeningFcn,
'gui_OutputFcn', @WSN_COVandCONN_Sim_OutputFcn, 'gui_LayoutFcn', [],
'gui_Callback', []);
    if (nargin && ischar(varargin{1}))
        gui_State.gui_Callback = str2func(varargin{1});
    end
    if (nargout)
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
end % End initialization code


% --- Executes just before WSN_CovANDConn_Sim is made visible.
function WSN_COVandCONN_Sim_OpeningFcn(hObject, eventdata, handles,
varargin)
    GlobalVars();
    %clear workspace; clc;
    handles.output = hObject; guidata(hObject, handles);
    axes(handles.GridAxes); cla;
    sensorLogoHandle = imshow('sensorLogo.jpg');
    axis([get(sensorLogoHandle, 'YData') get(sensorLogoHandle, 'XData')]);
    SetSamples(handles, 'initAll', 'hide'); % Set default values
    if (nodesDispersedF)
        ClearHandles(handles);
    end

    set(handles.COVandCONN_AM, 'Value',0);
    set(handles.COVandCONN_SR, 'Value',0);

    switch (COVandCONNAlg)
        case 'AM'
            set(handles.COVandCONN_AM, 'Value',1);
        case 'SR'
            set(handles.COVandCONN_SR, 'Value',1);
    end

    simFileInfo = dir('WSN_COVandCONN_Sim.m');
    set(handles.clearGrid, 'String','Quit');
    set(handles.statusText, 'String','Enter Grid parameters & choose
algorithm...');
    set(handles.analyzedNode, 'String','');
    set(handles.pwrRem, 'String','');
    set(handles.WSN_COVandCONN_Sim, 'Name', sprintf('WSN Coverage &
Connectivity Algorithms Simulator by elen@  -   %s', simFileInfo.date));
end

% --- Outputs from this function are returned to the command line.
```

```
function varargout = WSN_COVandCONN_Sim_OutputFcn(hObject, eventdata,
handles)
    varargout{1} = handles.output;
    reqVer = '7.1.0.19920 (R14)';
    if (version ~= reqVer) % Incorrect Matlab version detected
        msgbox('Matlab R14 SP3 is required to ensure proper GUI
performance.');
    end
end

% ================ BUTTON FUNCTIONS ================
function editParams_Callback(hObject, eventdata, handles)
    ParamEdit();
end

function recharge_Callback(hObject, eventdata, handles)
    GlobalVars();
    if (nodesDispersedF)
        nodePower = nodePowerMax .* ones(1, nodeCount); % Maximize the
power of each node
        %UpdateAnalyzedNode(handles, closestNode);
        UpdatePowerAxes(handles, 'noCalc');
        axes(handles.GridAxes);
        PlotGrid(); % Redraw the nodes to refresh their color
    end
end

function clearGrid_Callback(hObject, eventdata, handles)
    GlobalVars();
    if (networkSyncF)
        set(handles.source_sinkTime, 'String','');
        set(handles.numRX, 'String','');
        set(handles.droppedTX, 'String','');
        UpdateGridAxes(handles);
        DrawTopology(0);
        PlotGrid();
        networkSyncF = 0;
    elseif (networkLvlDiscF) % Clear the lvl-disc lines if no sync has been
performed
        ClearHandles(handles);
        UpdateGridAxes(handles);
        PlotGrid();
        set(handles.NodesMove, 'String','');
        set(handles.statusText, 'String','Enter network parameters & choose
algorithm...');
        networkLvlDiscF = 0;
    elseif (nodesDispersedF) % Clear nodes and reset to the opening screen
        WSN_COVandCONN_Sim_OpeningFcn(hObject, eventdata, handles, 1);
        axes(handles.PwrAxes); cla; axis([0 1 0 1]);
        set(handles.Moves, 'String','');
        set(handles.Distance, 'String','');
        set(handles.analyzedNode, 'String','');
        set(handles.pwrRem, 'String','');
        set(handles.clearGrid, 'String','Quit');
        nodesDispersedF = 0;
    else
        close;
    end
```

```
end

function plotGrid_Callback(hObject, eventdata, handles)
    GlobalVars(); clc;
    set(handles.clearGrid, 'String','Clear Grid');
    set(handles.statusText, 'String','Initializing Network, please
waiting...');
    if (~dataLoadedF) % Set variables here to avoid disrupting loaded
values
        SetSamples(handles, 'initArrays', 'hide'); % Set default values
        networkChangedF = 0;
        nodesDispersedF = 1;
        networkLvlDiscF = 0;
        networkSyncF = 0;


    else
        dataLoadedF = 0;
    end
    UpdateGridAxes(handles);
    PlotGrid();
    if (~networkChangedF && networkLvlDiscF)
    % Needed in case the data has been loaded and we want the discovering
lines...NetworkChangedF flag must be 0 to redraw the discovering lines
        DrawTopology(0);
    end
    UpdatePowerAxes(handles, 'noCalc');

    ClearHandles(handles);
    set(handles.Distance, 'String',sprintf('%g m', moveDistances));
    set(handles.Moves, 'String',sprintf('%g / %g',
movesNodes,ProcessesInit));

    set(handles.statusText, 'String','Enter network parameters & choose
algorithm...');
    %set(handles.analyzedNode, 'String',sourceNode);
    set(handles.pwrRem, 'String',nodePower(sourceNode));
    set(handles.topolDiscovery, 'Enable','on');
    set(handles.run, 'Enable','on');
end

function topolDiscovery_Callback(hObject, eventdata, handles)
    GlobalVars(); clc;
    if (networkChangedF || ~nodesDispersedF || dataLoadedF)
        plotGrid_Callback(hObject, eventdata, handles);
    end

    maxDistanceLow = nthroot(PtWLow/PrThresholdWLow, pathLossCoeff);
    maxDistanceHigh = nthroot(PtWHigh/PrThresholdWHigh, pathLossCoeff);

    set(handles.statusText, 'String','Simulating Topology Discovery, please
waiting...');
    set(handles.plotGrid, 'Enable','off');
    set(handles.clearGrid, 'Enable','off');
    set(handles.topolDiscovery, 'Enable','off');
    set(handles.run, 'Enable','off');
    set(handles.COVandCONN_AM, 'Enable','off');
    set(handles.COVandCONN_SR, 'Enable','off');
```

```
    set(handles.editParams, 'Enable','off');
    set(handles.recharge, 'Enable','off');
    pause(0.001);
    UpdateGridAxes(handles);
    PlotGrid();

    s = cputime;
    discoverTime = cputime - s;

    DrawTopology(pauseInt);
    ClearHandles(handles);
    PlotGrid();
    UpdatePowerAxes(handles);
    %UpdateAnalyzedNode(handles, closestNode);
    UpdateNodeStatus(handles);
    set(handles.plotGrid, 'Enable','on');
    set(handles.clearGrid, 'Enable','on');
    set(handles.topolDiscovery, 'Enable','on');
    set(handles.run, 'Enable','on');
    set(handles.COVandCONN_AM, 'Enable','on');
    set(handles.COVandCONN_SR, 'Enable','on');
    set(handles.editParams, 'Enable','on');
    set(handles.recharge, 'Enable','on');
    set(handles.discoverTime, 'String',discoverTime);
    set(handles.numRX, 'String',numRx);
    networkLvlDiscF = 1;
    networkSyncF = 0;
    set(handles.statusText, 'String','Topology Discovery completed!!');
end

function run_Callback(hObject, eventdata, handles)
    GlobalVars();
    networkSyncF = 1;
   if (networkChangedF || ~networkLvlDiscF)
        topolDiscovery_Callback(hObject, eventdata, handles);
        if (continuousF)
            %pause(1);
        end
    end


    set(handles.plotGrid, 'Enable','off');
    set(handles.clearGrid, 'Enable','off');
    set(handles.topolDiscovery, 'Enable','off');
    set(handles.run, 'Enable','off');
    set(handles.COVandCONN_AM, 'Enable','off');
    set(handles.COVandCONN_SR, 'Enable','off');
    set(handles.editParams, 'Enable','off');
    set(handles.recharge, 'Enable','off');
    set(handles.statusText, 'String','Grid Heads discovering...');
    pause(0.001);
    UpdateGridAxes(handles);
    DrawTopology(0);
    PlotGrid();

    numTx = 0;
    numRx = 0;
    dropTx = 0;
```

```
    s = cputime;

    switch (COVandCONNAlg)
        case 'AM'
            COVandCONNAlgorithmAM();
        case 'SR'
            COVandCONNAlgorithmSR();
    end

    source_sinkTime = cputime - s;

    % Continuously run
        while (continuousF && source_sinkTime <= 10)
            pause(1);
            UpdateGridAxes(handles);
            DrawTopology(0);
            PlotGrid();
            UpdatePowerAxes(handles);
            pause(1);


            numTx = 0;
            numRx = 0;
            dropTx = 0;
            UpdateGridAxes(handles);
            DrawTopology(0);
            PlotGrid();
            s = cputime;

            switch (COVandCONNAlg)
                case 'AM'
                    COVandCONNAlgorithmAM();
                case 'SR'
                    COVandCONNAlgorithmSR();
            end

            source_sinkTime = cputime - s;
            %UpdateAnalyzedNode(handles, closestNode);
            UpdateNodeStatus(handles);
            networkSyncF = 1;
        end

    UpdatePowerAxes(handles);
    UpdateAnalyzedNode(handles, 1);
    UpdateNodeStatus(handles);
    set(handles.plotGrid, 'Enable','on');
    set(handles.clearGrid, 'Enable','on');
    set(handles.topolDiscovery, 'Enable','on');
    set(handles.run, 'Enable','on');
    set(handles.COVandCONN_AM, 'Enable','on');
    set(handles.COVandCONN_SR, 'Enable','on');
    set(handles.editParams, 'Enable','on');
    set(handles.recharge, 'Enable','on');
    set(handles.source_sinkTime, 'String',source_sinkTime);
    set(handles.numRX, 'String',numRx);
    set(handles.droppedTX, 'String',DeadNodes);
    set(handles.statusText, 'String','Simulation completed!!!');
end
```

```
% ------------------- NON-BUTTON CALLBACKS -------------------
function continuous_Callback(hObject, eventdata, handles)
    GlobalVars();
    continuousF = get(gco,'Value');
end

function COVandCONN_AM_Callback(hObject, eventdata, handles)
    GlobalVars();
    set(handles.COVandCONN_AM, 'Value',1);
    set(handles.COVandCONN_SR, 'Value',0);
    COVandCONNAlg = 'AM';
end

function COVandCONN_SR_Callback(hObject, eventdata, handles)
    GlobalVars();
    set(handles.COVandCONN_AM, 'Value',0);
    set(handles.COVandCONN_SR, 'Value',1);
    COVandCONNAlg = 'SR';
end

function GridAxes_ButtonDownFcn(hObject, eventdata, handles)
    GlobalVars();
% Check to make sure that the nodes have been plotted and the user has
either performed network level discovery or wants to change the source node
    if (networkChangedF || dataLoadedF)
        plotGrid_Callback(hObject, eventdata, handles);
        tempStr = get(handles.statusText, 'String');
        tempColor = get(handles.statusText, 'BackgroundColor');
        set(handles.statusText, 'String','Network parameters changed!!');
        set(handles.statusText, 'BackgroundColor','red');
        pause(2);
        set(handles.statusText, 'String',tempStr);
        set(handles.statusText, 'BackgroundColor',tempColor);
        networkChangedF = 0;
        dataLoadedF = 0;
    elseif (nodesDispersedF)
        pt = get(gca,'currentpoint'); % Find the nearest node to the mouse-
click
        closestDist = xDistance;
        for i = 1 : nodeCount
            d = CalcDistance(pt(1,1), pt(1,2), neighborTable(i,1),
neighborTable(i,2));
            if  (d < closestDist)
                closestNode = i; % Nearest node to mouse-click after loop
is complete
                closestDist = d;
            end
        end

        if (changeSourceNodeF) % Change source node
            sourceNode = closestNode;
            closestNode = sourceNode;
            ClearHandles(handles);

            axes(handles.GridAxes);
            cla; axis([0 xDistance 0 yDistance]); axis on; axis xy; axis
fill; axis manual; hold all;
```

```
            PlotGrid();
            set(handles.Moves, 'String','');
            set(handles.statusText, 'String','Enter topolDiscovery
parameters...');
            networkLvlDiscF = 0;
            networkSyncF = 0;
            UpdateAnalyzedNode(handles, closestNode);
        else % Analyze node's properties
            PlotGrid();
            UpdateAnalyzedNode(handles, closestNode);
            if (networkLvlDiscF)
                PlotGrid();
                if (tracebackF && gridHeads(closestNode) ~= 0)
                    UpdateGridAxes(handles);
                    DrawTopology(0);
                    PlotGrid();

                end
            end
        end
    end
end

function PwrAxes_ButtonDownFcn(hObject, eventdata, handles)
    GlobalVars();
    pt = get(gca,'currentpoint'); % Find the nearest node to the mouse-
click
    closestDist = length(gridPower);
    for i = 1 : length(gridPower)
        d = CalcDistance(pt(1,1), pt(1,2), i, gridPower(i));
        if  (d < closestDist)
            closestPt = i; % Nearest event to mouse-click after loop is
complete
            closestDist = d;
        end
    end
end

% ================== MENU FUNCTIONS ==================
function Menu_LoadPar_Callback(hObject, eventdata, handles) % FILE
FUNCTIONS
    GlobalVars();
    [fname pname]= uigetfile('*.mat', 'Open');
    if ((ischar(fname))& (ischar(pname)))
        curdir = pwd;
        cd(pname);
        load(fname); % Load the entire workspace
        cd(curdir);
        SetSamples(handles, 'load','hide');
        dataLoadedF = 1;
        msgbox('Parameters loaded succesfully!');
    end
end

function Menu_SavePar_Callback(hObject, eventdata, handles)
    GlobalVars();
    [fname pname] = uiputfile('parameters.mat', 'Save As');
    if ((ischar(fname))& (ischar(pname)))
```

```
        curdir = pwd;
        cd(pname);
        fid = fopen(fname,'wt');
        save(fname, '*'); % Save the entire workspace
        fclose(fid);
        cd(curdir);
        networkLvlDiscF
        msgbox(sprintf('%s%s', 'Parameters successfully saved to ',
fname));
    end
end

function Menu_Exit_Callback(hObject, eventdata, handles)
    %clear all;
    if (isdeployed)
        close all;
        quit force;
    else
        close all;
    end
end

function Menu_View_NodeIDs_Callback(hObject, eventdata, handles) % VIEW
FUNCTIONS
    GlobalVars();
    if strcmp(get(gcbo,'Checked'),'on')
        viewNodeIDF = 0;
        set(gcbo, 'Checked','off');
    else
        viewNodeIDF = 1;
        set(gcbo, 'Checked','on');
    end
    if (nodesDispersedF) % Update the grid axes only if the nodes have been
dispersed
        UpdateGridAxes(handles);
        if (networkLvlDiscF) % Draw discovering lines only if network has
already been lvl-discovered
            DrawTopology(0);
        end
        PlotGrid();
    end
end

function Menu_MatlabVer_Callback(hObject, eventdata, handles) % HELP
FUNCTIONS
    msgbox(sprintf('Current Matlab Version:\n%s', version), 'Matlab
Version');
end

function Menu_Info_Callback(hObject, eventdata, handles)
    titleStr = 'WSN Coverage & Connectivity Control Simulator';
    authorStr = sprintf('%s\n%s', 'Elena Kakoulli, UCY');
    simFileInfo = dir('WSN_COVandCONN_Sim.m');
    msgbox(sprintf('%s\n%s\n\nLast Updated on %s', titleStr, authorStr,
simFileInfo.date), 'Info');
end

% ================== OTHER FUNCTIONS ==================
```

```
function UpdateAnalyzedNode(handles, node)
    GlobalVars();
    set(handles.analyzedNode, 'String',node);
    set(handles.pwrRem, 'String',nodePower(node));
    if (networkLvlDiscF) % Output discovering details

        if (gridHeads(node) == 0)
            set(handles.GridHead, 'String','Orphan');
        else
            set(handles.GridHead, 'String',num2str(gridHeads(node)));
        end

        set(handles.NodesMove,
'String',num2str(NodesMove(1:movesNodes,1)));

        if (nodePower(node) <= 0)
            set(handles.pwrRem, 'String','Depleted');
        else
            set(handles.pwrRem, 'String',num2str(nodePower(node)));
        end
    end
end

function UpdateNodeStatus(handles)
    GlobalVars();

    for i = 1 : nodeCount
        if (nodePower(i) <= 0)
            gridHeads(i) = 0;
        end
    end

    set(handles.Distance, 'String',sprintf('%g m', moveDistances));
    set(handles.Moves, 'String',sprintf('%g / %g',
movesNodes,ProcessesInit));

end

function UpdateGridAxes(handles)
    GlobalVars();
    axes(handles.GridAxes); cla;
    axis([0 xDistance 0 yDistance]);
    axis on; axis xy; axis fill; axis manual; hold all;
end

function UpdatePowerAxes(handles, x)
    GlobalVars();
    if (nargin == 2 & x == 'noCalc')
        gridPower = [1];
    else
        gridPower = [gridPower sum(nodePower)/(nodePowerMax*nodeCount)] %
Calculate new normalized grid power
    end
    axes(handles.PwrAxes); cla;
    axis([1 length(gridPower)+1 0 1]); set(handles.PwrAxes, 'YGrid','on',
'YMinorGrid','on');
    axis on; axis xy; axis fill; axis manual; hold all;
```

```
    plot(gridPower, '-b*');
end


% --- Executes during object creation, after setting all properties.
function WSN_COVandCONN_Sim_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WSN_COVandCONN_Sim (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Update Node Power                              %
%             Coverage & Connectivity Control in WSNs                  %
%                         Kakoulli Elena                               %
%                    Computer Science, UCY                             %
%                                                                      %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function UpdateNodePower()
    GlobalVars();
    for i = 1 : nodeCount
        if (nodePower(i) <= 0) % Node i has run out of energy
            DeadNodes = DeadNodes + 1;
            nodeLevel(i) = -1;
            gridHeads(i) = 0;
        end
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Set Samples                                 %
%             Coverage & Connectivity Control in WSNs                  %
%                         Kakoulli Elena                               %
%                    Computer Science, UCY                             %
%                                                                      %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function SetSamples(handles, sampleID, visibility)
    GlobalVars();
    switch sampleID
        case 'initAll'
            SetSamples(handles, 'mica2dot433', 'hide'); % Set default
values
            propSpd = 2.998e8;
            pauseInt = 0.2;
            xDistance = 80;
            yDistance = 100;
            nodeCount = 40;
            nodeDistribMode = 'rand';
            pathLossCoeff = 3.5;
```

```
            nodePowerMax = 100;
            nodePowerWarn = 10;
            rediscoverLimit = 20;
            COVandCONNAlg = 'AM';

        % Set flags and source node
            dataLoadedF = 0;
            nodesDispersedF = 0;
            networkChangedF = 0;
            networkLvlDiscF = 0;
            networkSyncF = 0;
            viewNodeIDF = 0;
            changeSourceNodeF = 0;
            selectNodeF = 0;
            tracebackF = 0;
            continuousF = 0;
            sourceNode = 1;


            SetSamples(handles, 'initArrays', 'hide');
        case 'initArrays'
            DeadNodes = 0;
            NodesMove = zeros(nodeCount,1);
            trustOfHead = [];
            Heads = [];
            trustedNodes = [];
            ProcessesInit = 0;
            Neighbors = [];
            NeighboringHeads = [];
            NumGridHeads = 0;
            movesNodes= 0;
            moveDistances = 0;
            VacantGrid = [];
            gridHeads = zeros(nodeCount,1);
            FoundTrustedNode = 0;
            neighborTable = [];
            parent = [];
            numChildren = [];
            children = [];
            deadNodeList = [];
            deadNodes = 0;
            orphanNodes = 0;
            numTx = 0;
            numRx = 0;
            dropTx = 0;
            RXtoTXratio = RXCurDraw / TXCurDraw;
            receiver_threshold = (3 + sqrt(9 + 8/RXtoTXratio)) / 2; %
Calculate point at which hybrid sync'ing switches from RBS to TPSN
            nodePower = nodePowerMax .* ones(1, nodeCount); % Set the power
of each node to max
            gridPower = [1]; % Set the grid's initial normalized power to 1
(sum of all fully charged nodes)
            closestNode = sourceNode;
            PtWLow = 10 ^ (PtLow/10); % Convert to watts
            PrThresholdWLow = 10 ^ (PrThresholdLow/10); % Convert to watts
            maxDistanceLow = nthroot(PtWLow/PrThresholdWLow,
pathLossCoeff);
            PtWHigh = 10 ^ (PtHigh/10); % Convert to watts
```

```
            PrThresholdWHigh = 10 ^ (PrThresholdHigh/10); % Convert to
watts
            maxDistanceHigh = nthroot(PtWHigh/PrThresholdWHigh,
pathLossCoeff);

            if (nodeDistribMode == 'rand') % Random distribution

                 neighborTable = rand([nodeCount 2]);

                %%%%%%%%% HOW MANY GRIDS SO HOW MANY GRIDHEADS %%%%%%%%%%
                NumGridHeads = floor(xDistance / 20) * floor(yDistance /
20);
                if(NumGridHeads==0)
                    NumGridHeads = 1;
                end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %FOR GRIDHEADS
                xCount = floor(xDistance / 20);
                yCount = floor(yDistance / 20);

                Heads = zeros(NumGridHeads,1);
                trustedNodes = zeros(nodeCount,1);
                trustOfHead = zeros(nodeCount,nodeCount);

                for i = 1 : nodeCount
                    neighborTable(i,1) = neighborTable(i,1) * xDistance;
                    neighborTable(i,2) = neighborTable(i,2) * yDistance;
                end

                dist = 20;
                num = 1;
                isHead = 0;
                for j = 1 : yCount
                    for i = 1 : xCount
                        Heads(num)=0;
                        for k = 1 : nodeCount
                        if(neighborTable(k,1)<=i*dist &&
neighborTable(k,2)<=j*dist && neighborTable(k,1)>=((i*dist)-20) &&
neighborTable(k,2)>=((j*dist)-20))

                            if(Heads(num)==0)
                                Heads(num) = k;
                                gridHeads(k) = k;
                                Neighbors(k)=0;
                                neighborTable(k,1) = i*dist - 0.5*dist;
                                neighborTable(k,2) = j*dist - 0.5*dist;

                                FromCenter = CalcDistance(i*dist - 0.5*dist
, j*dist - 0.5*dist, neighborTable(k,1), neighborTable(k,2));

                                % moving to center, so lose power
                                nodePower(k) = nodePower(k) -
(FromCenter/2);
                                moveDistances = moveDistances + FromCenter;
%ay3anw tin synoliki apostasis metakinisis pou egine

                            else
```

```
                                    gridHeads(k)=Heads(num);
                                    trustedNodes(Heads(num)) =
trustedNodes(Heads(num)) + 1;

trustOfHead(Heads(num),trustedNodes(Heads(num))) = k;
                                end
                            end

                            end

                            num = num + 1;
                        end
                    end

                    num = 1;
                       for j = 1 : yCount
                        for i = 1 : xCount

                            if(Heads(num)==0)
                                for w=1 : NumGridHeads
                                    if(Heads(w)~=0 && trustedNodes(Heads(w))>0)
                                            Heads(num) =
trustOfHead(Heads(w),trustedNodes(Heads(w)));

gridHeads(trustOfHead(Heads(w),trustedNodes(Heads(w)))) =
trustOfHead(Heads(w),trustedNodes(Heads(w)));

Neighbors(trustOfHead(Heads(w),trustedNodes(Heads(w))))=0;

neighborTable(trustOfHead(Heads(w),trustedNodes(Heads(w))),1) = i*dist -
0.5*dist;

neighborTable(trustOfHead(Heads(w),trustedNodes(Heads(w))),2) = j*dist -
0.5*dist;
                                            trustedNodes(Heads(w)) =
trustedNodes(Heads(w)) - 1;

                                            FromCenter = CalcDistance(i*dist -
0.5*dist , j*dist - 0.5*dist, Heads(num), Heads(num));

                                            % moving to center, so lose power
                                            nodePower(Heads(num)) =
nodePower(Heads(num)) - (FromCenter/2);
                                            moveDistances = moveDistances +
FromCenter; %ay3anw tin synoliki apostasis metakinisis pou egine

                                            break;
                                    end
                                end
                            end
                           num = num + 1;
                        end
                       end


                %%%%%%%%%%%%%%%% DIMIOURGIA GEITONWN %%%%%%%%%%%%%%%%%%%%
                 if(strcmp(COVandCONNAlg, 'AM'))
```

```
                for i=1 : NumGridHeads

                    for j=1 : NumGridHeads
                        if(j~=i)
                            %dist = CalcDistance(neighborTable(Heads(i),1),
neighborTable(Heads(i),2), neighborTable(Heads(j),1),
neighborTable(Heads(j),2));

                            if(neighborTable(Heads(j),1) <=
(neighborTable(Heads(i),1)+20) &&
neighborTable(Heads(j),2)<=(neighborTable(Heads(i),2)+20) &&
neighborTable(Heads(j),1)>=(neighborTable(Heads(i),1)-20) &&
neighborTable(Heads(j),2)>=(neighborTable(Heads(i),2)-20))
                                Neighbors(Heads(i)) =
Neighbors(Heads(i))+1;

NeighboringHeads(Heads(i),Neighbors(Heads(i)))=Heads(j);
                            end
                        end
                    end
                end
            end

            %%%%%%%%%%%%%% DIMIOURGIA HAMILTON CYCLE %%%%%%%%%%%%%%%%%%%
        if(strcmp(COVandCONNAlg, 'SR'))
                if(xCount>=2 || yCount>=2) % proipo8esi gia na ypar3ei
Hamilton Cycle
                 %HAMILTON CYCLE
                  if(rem(xCount,2)==0)     % The columns have even
number of GridHeads so we have a Hamilton cycle path
                      start = NumGridHeads-(xCount-1);
                      successors(start) =  start-xCount;
                      precedings(start) = start+1;
                      start=start-xCount;
                      for i=2: yCount-1      %pros ta katw
                          successors(start) =  start-xCount;
                          precedings(start) = start+xCount;
                          start=start-xCount;
                      end
                      successors(1) =  2;
                      precedings(1) = xCount+1;
                      for i=2: xCount-1      %PROS TA DE3IA
                          successors(i) = i+1;
                          precedings(i) = i-1;
                      end
                       successors(xCount) = 2*xCount;
                       precedings(xCount) = xCount-1;

                      start=2*xCount;
                      for i=2 : yCount-1 %pros ta panw
                          successors(start) = start+xCount;
                          precedings(start) = start-xCount;
                          start = start+xCount;
                      end

                      successors(NumGridHeads) = NumGridHeads-1;
                      precedings(NumGridHeads) = NumGridHeads-xCount;
```

```
                        if(xCount>2)   % yparxoun mesaies grammes
                         stop = (xCount/2)-1;
                         start = NumGridHeads-1;
                          for point=1 : stop
                               successors(start) = start-xCount;
                               precedings(start) = start+1;
                               start = start-xCount;
                               for i=1 : yCount-3   %pros ta katw
                               successors(start) =  start-xCount;
                               precedings(start) = start+xCount;
                               start = start-xCount;
                               end
                               successors(start) =  start-1;
                               precedings(start) = start+xCount;
                               successors(start-1) =  start-1+xCount;
                               precedings(start-1) = start;
                               start = start-1 + xCount;

                               for i=1 : yCount-3   %pros ta panw
                               successors(start) =  start+xCount;
                               precedings(start) = start-xCount;
                               start = start+xCount;
                               end
                               successors(start) =  start-1;
                               precedings(start) = start-xCount;
                               start = start-1;
                            end

                        end
                      elseif(rem(yCount,2)==0) % The rows have even number
of GridHeads so we have a Hamilton cycle path
                        successors(1) = 2;
                        precedings(1) = 1+xCount;

                        for i=2: xCount-1  %pros ta de3ia
                            successors(i) =  i+1;
                            precedings(i) = i-1;
                        end
                        successors(xCount) =  xCount*2;
                        precedings(xCount) = xCount-1;
                        start = 2*xCount;
                        for i=2: yCount-2      %pros ta panw
                            successors(start) = start+xCount;
                            precedings(start) = start-xCount;
                            start = start+xCount;
                        end

                        successors(NumGridHeads) =  NumGridHeads-1;
                        precedings(NumGridHeads) = NumGridHeads-xCount;
                        start = start-1;

                        for i=2 : xCount-2     %pros ta aristera

                            successors(start) = start-1;
                            precedings(start) = start+1;
                            start = start-1;
```

```
                    end

                    start=NumGridHeads-xCount+1;
                    successors(start) = start-xCount;
                    precedings(start) = start+1;


                    if(yCount>2)   % yparxoun mesaies grammes
                     stop = (yCount/2)-1;
                     start = start-xCount;

                       for point=1 : stop
                           successors(start) = start+1;
                           precedings(start) = start+xCount;

                           for i=1 : xCount-3   %pros ta de3ia
                               start = start+1;
                           successors(start) =  start+1;
                           precedings(start) = start-1;
                           end

                           successors(start+1) = start+1-xCount;
                           precedings(start+1) = start;

                           start=start+1-xCount;
                           successors(start) =  start-1;
                           precedings(start) = start+xCount;
                           start = start-1;

                           for i=1 : xCount-3   %pros aristera
                           successors(start) =  start-1;
                           precedings(start) = start+1;
                           start = start-1;
                           end
                           successors(start) =  start-xCount;
                           precedings(start) = start+1;
                           start = start-xCount;
                        end

                    end


                else
                    % The MxN network don't have even columns or rows
                    % so we have DUAL Hamilton cycle paths

                end
              end
          end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
       end

     if (nodeDistribMode == 'grid') % Uniform grid distribution
           dist = sqrt(xDistance*yDistance / nodeCount);
           xCount = floor(xDistance / dist);
```

```
                yCount = floor(yDistance / dist);
                if (xCount*yCount < nodeCount)
                    if (xDistance > yDistance)
                        xCount = xCount + 1;
                        if (xCount*yCount < nodeCount)
                            yCount = yCount + 1;
                        end
                    else
                        yCount = yCount + 1;
                        if (xCount*yCount < nodeCount)
                            xCount = xCount + 1;
                        end
                    end
                end
                dist = min(xDistance/xCount, yDistance/yCount);
                for j = 1 : yCount
                    for i = 1 : xCount
                        neighborTable = [neighborTable; i*dist - 0.5*dist
j*dist - 0.5*dist];
                    end
                end
          end


        case 'mica2dot433'
            moteClockSpd = 4;
            radioFreq = 433;
            TXCurDraw = 25;
            RXCurDraw = 8;
            PtLow = -2;
            PrThresholdLow = -85;
            PtHigh = -2;
            PrThresholdHigh = -101;
            guarDistance = 0;
        case 'mica2dot916'
            moteClockSpd = 4;
            radioFreq = 916;
            TXCurDraw = 27;
            RXCurDraw = 10;
            PtLow = -2;
            PrThresholdLow = -85;
            PtHigh = -2;
            PrThresholdHigh = -98;
            guarDistance = 0;
        case 'micaz'
            moteClockSpd = 4;
            radioFreq = 2400;
            TXCurDraw = 17.4;
            RXCurDraw = 19.7;
            PtLow = -5;
            PrThresholdLow = -85;
            PtHigh = -5;
            PrThresholdHigh = -94;
            guarDistance = 0;
        case 'load' % Leave variables as they are
        otherwise
    end
```

```
    if (visibility == 'show')
        set(handles.xDist, 'String', xDistance);
        set(handles.yDist, 'String', yDistance);
        set(handles.nodeCount, 'String', nodeCount);
        if (nodeDistribMode == 'grid')
            set(handles.gridMode, 'Value',1);
            set(handles.randMode, 'Value',0);
        elseif (nodeDistribMode == 'rand')
            set(handles.randMode, 'Value',1);
            set(handles.gridMode, 'Value',0);
        end

        set(handles.TxPwrLow, 'String',PtLow);
        set(handles.RxThreshLow, 'String',PrThresholdLow);
        set(handles.TxPwrHigh, 'String',PtHigh);
        set(handles.RxThreshHigh, 'String',PrThresholdHigh);
        set(handles.pathLossCoeff, 'String',pathLossCoeff);
        set(handles.guarDist, 'String',guarDistance);
        set(handles.pauseInt, 'String',pauseInt);
        set(handles.maxTxDistLow, 'String',maxDistanceLow);
        set(handles.maxTxDistHigh, 'String',maxDistanceHigh);

        set(handles.moteCPU, 'String',moteClockSpd);
        set(handles.radioFreq, 'String',radioFreq);
        set(handles.TXCurDraw, 'String',TXCurDraw);
        set(handles.RXCurDraw, 'String',RXCurDraw);
        set(handles.nodePowerMax, 'String',nodePowerMax);
        set(handles.nodePowerWarn, 'String',nodePowerWarn);
        set(handles.rediscoverLimit, 'String',rediscoverLimit);
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Plot Grid                                      %
%            Coverage & Connectivity Control in WSNs                    %
%                        Kakoulli Elena                                 %
%                    Computer Science, UCY                              %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function PlotGrid()
% Plot each of the nodes according to their role and status in the network
    GlobalVars();

    for i=1 : NumGridHeads
          s = sprintf('%d', Heads(i));

            %nodeColor = checkNodePwrLevel(i, 'b');
            if (viewNodeIDF)
                plot(neighborTable(Heads(i),1), neighborTable(Heads(i),2),
'.', 'MarkerEdgeColor','b', 'LineWidth',2);
                text(neighborTable(Heads(i),1) + xDistance/100,
neighborTable(Heads(i),2), s, 'FontSize',8, 'FontWeight','bold',
'HorizontalAlignment','left', 'Color',nodeColor);
            else
```

```
                plot(neighborTable(Heads(i),1), neighborTable(Heads(i),2),
'^', 'MarkerEdgeColor','b', 'LineWidth',2);
            end
    end

    for i = 1 : nodeCount
        s = sprintf('%d', i);
        isHead = 0;
       if(gridHeads(i)>0) %den metakini8ike allou
            nodeColor = checkNodePwrLevel(i, 'k');
            %if(gridHeads(i)==1)
            %    plot(neighborTable(i,1), neighborTable(i,2), 'x',
'MarkerEdgeColor','r', 'LineWidth',4);
            %else
            for j=1 : NumGridHeads
                if(Heads(j)==i)
                    isHead = 1;
                    break;
                end
            end
            if(isHead==0)
                if (viewNodeIDF)
                plot(neighborTable(i,1), neighborTable(i,2), '.',
'MarkerEdgeColor',nodeColor, 'LineWidth',2);
                text(neighborTable(i,1) + xDistance/100,
neighborTable(i,2), s, 'FontSize',8, 'HorizontalAlignment','left',
'Color',nodeColor);
                else
                plot(neighborTable(i,1), neighborTable(i,2), 'o',
'MarkerEdgeColor',nodeColor, 'LineWidth',2);
                end
            end
       end
    end
end

function [color] = checkNodePwrLevel(curNode, stdColor)
% Change the node's color if its power level has decreased below the
warning level
    GlobalVars();
    if (nodePower(curNode) > nodePowerWarn)
        color = stdColor;
    elseif (nodePower(curNode) <= nodePowerWarn && nodePower(curNode) > 0)
        color = 'y'; % Yellow for warning
    else
        color = 'r'; % Red for depleted
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Edit Parameters                                 %
%             Coverage & Connectivity Control in WSNs                   %
%                         Kakoulli Elena                                %
%                     Computer Science, UCY                             %
%                                                                       %
%                                                                       %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = ParamEdit(varargin)
    gui_Singleton = 1;
    gui_State = struct('gui_Name', mfilename, 'gui_Singleton',
gui_Singleton, 'gui_OpeningFcn', @ParamEdit_OpeningFcn, 'gui_OutputFcn',
@ParamEdit_OutputFcn, 'gui_LayoutFcn', [], 'gui_Callback', []);
    if (nargin && ischar(varargin{1}))
        gui_State.gui_Callback = str2func(varargin{1});
    end
    if (nargout)
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
end % End initialization code

% --- Executes just before the editor is made visible.
function ParamEdit_OpeningFcn(hObject, eventdata, handles, varargin)
    GlobalVars();
    handles.output = hObject; guidata(hObject, handles);
    SetSamples(handles, 'load', 'show');
end

% --- Outputs from this function are returned to the command line.
function varargout = ParamEdit_OutputFcn(hObject, eventdata, handles)
    varargout{1} = handles.output;
end

function gridParams_Callback(hObject, eventdata, handles)
    GlobalVars();
    if (str2double(get(handles.xDist, 'String')) == 0)
        set(hObject, 'String', xDistance);
    end
    if (str2double(get(handles.yDist, 'String')) == 0)
        set(hObject, 'String', yDistance);
    end
    if (str2double(get(handles.nodeCount, 'String')) == 0)
        set(hObject, 'String', nodeCount);
    end

    if (str2double(get(handles.xDist, 'String')) ~= xDistance || ...
            str2double(get(handles.yDist, 'String')) ~= yDistance || ...
            str2double(get(handles.nodeCount, 'String')) ~= nodeCount)
        networkChangedF = 1;
    end
end

function gridMode_Callback(hObject, eventdata, handles)
    GlobalVars();
    set(handles.gridMode, 'Value', 1);
    set(handles.randMode, 'Value', 0);
end

function randMode_Callback(hObject, eventdata, handles)
    GlobalVars();
    set(handles.gridMode, 'Value', 0);
    set(handles.randMode, 'Value', 1);
```

```
end

function signalParams_Callback(hObject, eventdata, handles)
    GlobalVars();
    pathLossCoeffTemp = str2double(get(handles.pathLossCoeff, 'String'));
    PtLTemp = str2double(get(handles.TxPwrLow, 'String'));
    PtWLTemp = 10^(PtLTemp/10) / 1000; % Convert to watts
    PrThresholdLTemp = str2double(get(handles.RxThreshLow, 'String'));
    PrThresholdWLTemp = 10^(PrThresholdLTemp/10) / 1000; % Convert to watts
    set(handles.maxTxDistLow, 'String', nthroot(PtWLTemp/PrThresholdWLTemp,
pathLossCoeffTemp));

    PtHTemp = str2double(get(handles.TxPwrHigh, 'String'));
    PtWHTemp = 10^(PtHTemp/10) / 1000; % Convert to watts
    PrThresholdHTemp = str2double(get(handles.RxThreshHigh, 'String'));
    PrThresholdWHTemp = 10^(PrThresholdHTemp/10) / 1000; % Convert to watts
    set(handles.maxTxDistHigh, 'String',
nthroot(PtWHTemp/PrThresholdWHTemp, pathLossCoeffTemp));
end

function resetParams_Callback(hObject, eventdata, handles)
    GlobalVars();
    SetSamples(handles, 'mica2dot433', 'show');
    close;
end

function cancelParams_Callback(hObject, eventdata, handles)
    networkChangedF = 0;
    close;
end

function saveParams_Callback(hObject, eventdata, handles)
    GlobalVars();
    if (xDistance ~= str2double(get(handles.xDist, 'String')) ...
            || yDistance ~= str2double(get(handles.yDist, 'String')) ...
            || nodeCount ~= str2double(get(handles.nodeCount, 'String')))
        networkChangedF = 1;
    else
        networkChangedF = 0;
    end

    xDistance = str2double(get(handles.xDist, 'String'));
    yDistance = str2double(get(handles.yDist, 'String'));
    nodeCount = str2double(get(handles.nodeCount, 'String'));
    if (get(handles.gridMode, 'Value'))
        nodeDistribMode = 'grid';
    else
        nodeDistribMode = 'rand';
    end
    PtLow = str2double(get(handles.TxPwrLow, 'String'));
    PtWLow = 10^(PtLow/10) / 1000;
    PrThresholdLow = str2double(get(handles.RxThreshLow, 'String'));
    PrThresholdWLow = 10^(PrThresholdLow/10) / 1000;
    PtHigh = str2double(get(handles.TxPwrHigh, 'String'));
    PtWHigh = 10^(PtHigh/10) / 1000;
    PrThresholdHigh = str2double(get(handles.RxThreshHigh, 'String'));
    PrThresholdWHigh = 10^(PrThresholdHigh/10) / 1000;
    pathLossCoeff = str2double(get(handles.pathLossCoeff, 'String'));
```

```
    maxDistanceLow = nthroot(PtWLow/PrThresholdWLow, pathLossCoeff);
    maxDistanceHigh = nthroot(PtWHigh/PrThresholdWHigh, pathLossCoeff);
    guarDistance = str2double(get(handles.guarDist, 'String'));
    pauseInt = str2double(get(handles.pauseInt, 'String'));
    moteClockSpd = str2double(get(handles.moteCPU, 'String'));
    radioFreq = str2double(get(handles.radioFreq, 'String'));

    TXCurDraw = str2double(get(handles.TXCurDraw, 'String'));
    RXCurDraw = str2double(get(handles.RXCurDraw, 'String'));
    RXtoTXratio = RXCurDraw / TXCurDraw;
    receiver_threshold = (3 + sqrt(9 + 8/RXtoTXratio)) / 2; % Calculate
point at which hybrid sync'ing switches from RBS to TPSN

    nodePowerMax = str2double(get(handles.nodePowerMax, 'String'));
    nodePowerWarn = str2double(get(handles.nodePowerWarn, 'String'));
    rediscoverLimit = str2double(get(handles.rediscoverLimit, 'String'));
    close;
end

% ------------------- SAMPLE FUNCTIONS -------------------
function Menu_mica2dot433_Callback(hObject, eventdata, handles)
    SetSamples(handles, 'mica2dot433', 'show');
    SetSamples(handles, 'initArrays', 'hide');
end

function Menu_mica2dot916_Callback(hObject, eventdata, handles)
    SetSamples(handles, 'mica2dot916', 'show');
    SetSamples(handles, 'initArrays', 'hide');
end

function Menu_micaz_Callback(hObject, eventdata, handles)
    SetSamples(handles, 'micaz', 'show');
    SetSamples(handles, 'initArrays', 'hide');
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Is Received                                  %
%            Coverage & Connectivity Control in WSNs                    %
%                      Kakoulli Elena                                   %
%                  Computer Science, UCY                                %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [r] = IsReceived(dist, packetType)
% Return 1 for successful TX or 0 for failed TX
    GlobalVars();
    r = 0;
    if (strcmp(packetType, 'topolDiscovery'))
        maxDist = maxDistanceLow;
    else
        maxDist = maxDistanceHigh;
    end
    gDist = min(guarDistance, maxDist); % Guaranteed TX distance

    if (dist <= gDist)
```

```
        r = 1;
    elseif (dist <= maxDist)
        prob = 1 - (dist - gDist) / (maxDist - gDist);
        if (rand() < prob)
            r = 1;
        end
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Global Variables                                %
%             Coverage & Connectivity Control in WSNs                   %
%                         Kakoulli Elena                                %
%                     Computer Science, UCY                             %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global verDate; % Version number for the simulator
global propSpd; % Speed of light (in m/s)

% Grid Parameters
global xDistance; % Length of x-axis on grid
global yDistance; % Length of y-axis on grid
global nodeCount; % Number of nodes in the grid
global nodeDistribMode; % Type of grid distribution for nodes (grid or
random)

% Hardware Parameters
global moteClockSpd; % Speed of mote's CPU (in MHz)
global radioFreq; % Throughput of mote's transceiver (in Kbps)
global TXCurDraw; % Current draw for a transmission (in mA)
global RXCurDraw; % Current draw for a reception (in mA)
global receiver_threshold; % Number of receivers needed where HTC becomes
more efficient than APC
global nodePowerMax; % Max power for each node
global nodePowerWarn; % Power where node shows low-power warning
global nodePower; % [1 x nodeCount] array holding power for each node
global nodeBuffer;% [1 x nodeBuffer] array holding buffer for each node
global gridPower; % [1 x *] array holding the grid's total power (index is
number of energy-consuming events)
global deadNodes; % Number of total dead nodes in the network

% discover Parameters
global PtLow; % Transmission power (in dB) for discover packets
global PtWLow; % Transmission power (in W) for discover packets
global PrThresholdLow; % Threshold power (in dB) for discover packets
global PrThresholdWLow; % Threshold power (in W) for discover packets
global PtHigh; % Transmission power (in dB) for sync packets
global PtWHigh; % Transmission power (in W) for sync packets
global PrThresholdHigh; % Threshold power (in dB) for sync packets
global PrThresholdWHigh; % Threshold power (in W) for sync packets
global pathLossCoeff; % Path loss coefficient for transmission equation
global guarDistance; % Guaranteed distance for successful transmission
global maxDistanceLow; % Maximum distance nodes are capable of transmitting
a discover packet
```

```
global maxDistanceHigh; % Maximum distance nodes are capable of
transmitting a sync packet
global RXtoTXratio; % Ratio of reception power to transmission power
global pauseInt; % Length of time between plotting generations

% topolDiscovery Variables
global neighborTable; % nodeCount x nodeCount array holding x and y
coordinates for each node
global parent; % [1 x nodeCount] array showing parent node (i.e. parent(6)
returns the parent node of node 6)
global numChildren; % [1 x nodeCount] array showing a node's number of
children
global children; % [nodeCount x nodeCount] array showing each node's
children
global nodeLevel; % [1 x nodeCount] array showing the generation of a given
node (-1 means it is an orphan)
global maxGens; % Maximum number of generations possible in the
topolDiscovery
global sourceNode; % Root node for transmissions
global closestNode; % Closest node user clicked to
global orphanNodes; % Variable to keep track of the number of orphaned
nodes
global discoverTime; % Time used to topolDiscovery the network

% Coverage or Connectivity Variables
global COVandCONNAlg;   % Type of coverage or connectivity Algorithm being
used
global numTx;           % Number of transmissions used to perform
congestionAlgorithm
global numRx;           % Number of receptions used to perform
congestionAlgorithm
global dropTx;          % Number of transmissions used to perform
congestionAlgorithm
global rediscoverLimit; % Percentage of dead nodes at which network must
rediscover
global source_sinkTime; % Time used to send data from the source to the
sink
global DeadNodes;
% Grid Heads Variables & flags
global gridHeads;     % periexei gia ka8e node, diladi ka8e 8esi tou
antistoixei se ka8e node kai to periexomeno einai o gridHead pou anikei
global VacantGrid;    % periexei to adeio grid
global NoVacant;      % flag when the grid is no vacant
global FoundTrustedNode; %periexei ton node pou 8a kini8ei
global NumGridHeads;  % o ari8mos twn gridHeads
global NeighboringHeads;  % [NumGridHeads x NumGridHeads] array showing
each node's children
global Neighbors;     % o ka8e gridHead me ton ari8mo geitonikwn gridHeads
tou
global movesNodes;    % o ari8mos twn nodes pou kini8ikan
global moveDistances; % h synoliki apostasi kinisis twn nodes
global ProcessesInit; % ta processes pou ginontai initiated
global trustedNodes;  % pinakas ka8e 8esi tou opoiou einai o ari8mos twn
trusted nodes ka8e gridHead
global Heads;         % pinakas pou periexei olous tous grid heads
global trustOfHead;   % pinakas me tous trusted nodes
global NodesMove;     % Oi komboi pou kini8ikan kata tin diarkeia twn
algori8mwn
```

```
% FOR SR Algorithm
global successors;  %[NumGridHeads 2]
global precedings; %[NumGridHeads 2]


% Flags
global dataLoadedF; % =1 when the parameters are loaded from a .mat file
global nodesDispersedF; % =1 when the nodes have been plotted
global networkChangedF; % =1 when the xDist, yDist, or nodeCount are
changed and the plot has not been updated
global continuousF; % =1 when the "Continuous" checkbox is selected
global changeSourceNodeF; % =1 when the "Change Source" checkbox is
selected
global tracebackF; % =1 when the "Trace back to Root" checkbox is selected
global viewNodeIDF; % =1 when the "View Node IDs" menu item is selected
global networkLvlDiscF; % =1 when the network has done the level discovery
(discovering)
global networkSyncF; % =1 when the network has been completely synchronized
global congestedF; %=1 when the node is congested


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Draw Topology                                  %
%            Coverage & Connectivity Control in WSNs                    %
%                         Kakoulli Elena                                %
%                    Computer Science, UCY                              %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function DrawTopology(delay)
% Draw the discovering lines
    GlobalVars;
         for j = 1 : nodeCount
            UpdateNodePower();
            x1 = neighborTable(gridHeads(j),1);
            y1 = neighborTable(gridHeads(j),2);
            x2 = neighborTable(j,1);
            y2 = neighborTable(j,2);
            line([x1 x2], [y1 y2], 'Color', 'g', 'LineWidth', 2);
         end


        if (delay ~= 0)
            pause(delay);
        end

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Coverage & Connectivity Algorithm SR                      %
%            Coverage & Connectivity Control in WSNs                    %
%                         Kakoulli Elena                                %
%                    Computer Science, UCY                              %
%                                                                       %
```

```
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function COVandCONNAlgorithmSR(method, startNode)
% Function implements the SR algorithm for coverage and connectivity
control

    GlobalVars();
    VacantGrid = zeros(NumGridHeads);
    finished = 0;    %flag for end of algorithm
    gridHeads(Heads(3))=0; % for sample - this must be change later

    while(finished==0)
         GridHeadsCommunicSR();

        for i=1 : NumGridHeads
            if(VacantGrid(Heads(i))~=0)
                finished = 0;
                break;
            else
                finished = 1;
            end
        end

        if(finished==0)
            CascadingMovementSR();
         %gridHeads(3)=3;
        end

    end

    %% For GridHeads connection
    for i=1 : NumGridHeads

        if(VacantGrid(Heads(i))==0)
                    x1 = neighborTable(Heads(i),1);
                    y1 = neighborTable(Heads(i),2);
                    x2 = neighborTable(Heads(successors(i)),1)
                    y2 = neighborTable(Heads(successors(i)),2);
                    line([x1 x2], [y1 y2], 'Color', 'cyan', 'LineWidth',
2);
        end
    end
    UpdateNodePower();
end

%% STEP 1: Oloi oi gridHeads elegxoun ean yparxei connection meta3y tous
%% alliws yparxei vacant grid kai 8a 3ekinisei cascading movement apo ton
%% 1o gridHead pou to anixneyse kai 8a enimerwsei tous ypoloipous me ena
%% notification kai to STEP 1 8a ginete epanaliptika efoson den pairnei
%% kanenas kapoio notification msg

function GridHeadsCommunicSR()

    GlobalVars();
    NoVacant = -1;
    RandomHeads = randperm(NumGridHeads);
```

```
    for i=1 : NumGridHeads     % o ka8e gridHead elegxei

       if(gridHeads(Heads(RandomHeads(i)))~=0 &&
successors(RandomHeads(i))~=0)
        dist = CalcDistance(neighborTable(Heads(RandomHeads(i)),1),
neighborTable(Heads(RandomHeads(i)),2),
neighborTable(Heads(successors(RandomHeads(i))),1),
neighborTable(Heads(successors(RandomHeads(i))),2));

        %sensing, so lose power
        nodePower(RandomHeads(i)) = nodePower(RandomHeads(i)) - (dist/4);


            if(dist~=0)
                PrWLow = PtWLow / (dist ^ pathLossCoeff); % Calculate
reception power
            end
            %IsReceived(dist, 'topolDiscovery') == 0


            if(gridHeads(Heads(successors(RandomHeads(i))))==0 &&
trustedNodes(Heads(successors(RandomHeads(i)))) == 0) %elegxos gia vacant
grid

                VacantGrid(Heads(RandomHeads(i))) =
Heads(successors(RandomHeads(i)));
                ProcessesInit = ProcessesInit + 1;
            else
                VacantGrid(Heads(RandomHeads(i))) = 0;
            end
       end
    end
end

%% STEP 2: Oi gridHeads pou exo un anixneysei kapoio vacant grid 8a prepei
%% na psaxoun na broun kapoio trusted node gia na metakini8ei sto vacant
grid
%% kai na ginei aytos o gridHead tou vacant grid, alliws an den bre8ei na
steiloun
%% ena notification msg stous geitonikous gridHeads gia na arxikopoihsoun
%% kai oi idioi mia tetoia diadikasia kai na metakini8ei o idios sto vacant
grid
%% kai na ginei se ayto o gridHead

function CascadingMovementSR();
    GlobalVars();
    RandomHeads = randperm(NumGridHeads);

    for i=1 : NumGridHeads

        if(VacantGrid(Heads(RandomHeads(i)))>0)

            movesNodes = movesNodes + 1; %ay3anw tous kombous pou kini8ikan

            x1 = neighborTable(Heads(RandomHeads(i)),1);
            y1 = neighborTable(Heads(RandomHeads(i)),2);
            x2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),1);
            y2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),2);
```

```
            plot(neighborTable(VacantGrid(Heads(RandomHeads(i))),1),
neighborTable(VacantGrid(Heads(RandomHeads(i))),2), 'x',
'MarkerEdgeColor','r', 'LineWidth',10);
            drawLineStatus(x1, x2, y1, y2, 'NoConnection');

            if(trustedNodes(Heads(RandomHeads(i)))>0) %elegxos ean yparxei
trusted node mesa sto grid pou einai arxigos
                %briskw ton 1o apo ta trusted nodes tou
                FoundTrustedNode =
trustOfHead(Heads(RandomHeads(i)),trustedNodes(Heads(RandomHeads(i))));
            else
              FoundTrustedNode = Heads(RandomHeads(i));
              %enimerwsi tou preceding node tou
              NotificationSR(Heads(RandomHeads(i)), RandomHeads(i));
            end


           % ypologismos apostasis pou metakineitai o trusted node 'i o
gridHead
              dist = CalcDistance(neighborTable(FoundTrustedNode,1),
neighborTable(FoundTrustedNode,2),
neighborTable(VacantGrid(Heads(RandomHeads(i))),1),
neighborTable(VacantGrid(Heads(RandomHeads(i))),2));
              moveDistances = moveDistances + dist; %ay3anw tin synoliki
apostasis metakinisis pou egine
              NodesMove(movesNodes) = FoundTrustedNode;

              if(dist~=0)
                PrWLow = PtWLow / (dist ^ pathLossCoeff); % Calculate
reception power
              end

               % moving, so lose power
              nodePower(FoundTrustedNode) = nodePower(FoundTrustedNode) -
(dist/2);

          if(FoundTrustedNode == Heads(RandomHeads(i))) % ean den bre8ike
trusted node

              %move & change gridhead
              drawLineStatus(x1, x2, y1, y2, 'CascadingMovement');
              %neighborTable(VacantGrid(i),1) =
              %neighborTable(VacantGrid(i),2) =

gridHeads(VacantGrid(Heads(RandomHeads(i))))=VacantGrid(Heads(RandomHeads(i
)));
              gridHeads(FoundTrustedNode)=0;

          else   % tote bre8ike trusted node k prepei na metakini8ei sto
vacant grid k na ginei o gridHead tou
              x1 = neighborTable(FoundTrustedNode,1);
              y1 = neighborTable(FoundTrustedNode,2);
              x2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),1);
              y2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),2);
              drawLineStatus(x1, x2, y1, y2, 'CascadingMovement');

              %metakinisi tou trusted node sto vacant grid kai ginete o
              %gridHead tou vacant grid
```

```
              neighborTable(FoundTrustedNode,1) =
neighborTable(VacantGrid(Heads(RandomHeads(i))),1);
              neighborTable(FoundTrustedNode,1) =
neighborTable(VacantGrid(Heads(RandomHeads(i))),2);
              gridHeads(FoundTrustedNode)=0;
              gridHeads(VacantGrid(Heads(RandomHeads(i))))=
VacantGrid(Heads(RandomHeads(i)));
              trustedNodes(Heads(RandomHeads(i))) =
trustedNodes(Heads(RandomHeads(i))) - 1; %feygei o trusted node kai ara
meiwnete kai o ari8mos twn trusted nodes toy gridHead
              %gridHeads(VacantGrid(i))=FoundTrustedNode;
              %gridHeads(FoundTrustedNode)=0;
              %....enimerwsi pws einai o new gridHead tou vacant grid

          end

           plot(neighborTable(VacantGrid(Heads(RandomHeads(i))),1),
neighborTable(VacantGrid(Heads(RandomHeads(i))),2), '^',
'MarkerEdgeColor','b', 'LineWidth',2);

           VacantGrid(Heads(RandomHeads(i))) = 0;
        end
    end
end


function NotificationSR(gridHead, position)
% Send notification msg from the gridHead to its precending gridHead
% gridHead to inform it for his cascading movement to the vacant grid

    GlobalVars();

        if (Heads(precedings(position))~=VacantGrid(gridHead) &&
nodePower(gridHead) >= 1 && nodePower(Heads(precedings(position))) > 0)

            x1 = neighborTable(gridHead,1);
            y1 = neighborTable(gridHead,2);
            x2 = neighborTable(Heads(precedings(position)),1);
            y2 = neighborTable(Heads(precedings(position)),2);

            drawLineStatus(x1, x2, y1, y2, 'MsgNotification');
            numTx = numTx + 1;
            nodePower(gridHead) = nodePower(gridHead) - 1;
            numRx = numRx + 1;
            nodePower(Heads(precedings(position))) =
nodePower(Heads(precedings(position))) - RXtoTXratio;

        end

end


function drawLineStatus(startX, endX, startY, endY, type)
    GlobalVars();
    switch (type)
        case 'MsgNotification'
            lineColor = 'b';
            style = '-';
```

```
        case 'CascadingMovement'
            lineColor = 'red';
            style = '--';
        case 'NoConnection'
            lineColor = 'm';
            style = '--';
        otherwise
    end
    line([startX endX], [startY endY], 'Color',lineColor, 'LineWidth',2,
'LineStyle',style);
    pause(0.8);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Coverage & Connectivity Algorithm AM       %
%            Coverage & Connectivity Control in WSNs      %
%                       Kakoulli Elena                    %
%                    Computer Science, UCY                %
%                                                         %
%                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Control scheme under active model:
% 1. For any grid head u that detects a vacant neighboring
% grid, if no notification is received in the previous round
% from that area (to avoid overreaction), a replacement
% process is initiated after the corresponding region area is determined.
% 2. For any grid head u that has started the replacement
% process or that is notified in the cascading replacement
% process, find one of neighboring spare trusted nodes
% in its grid, say node v, to move into that neighboring
% vacant grid before the next round starts.
% 3. If such a node v cannot be found, select any neighboring
% grid that is other than the vacant one but still in
% region. Then, send out the notification for the replacement
% of u, attaching the information of region and
% such a selection. It will always select the one with
% spare trusted node(s) first. After that, move u to the
% vacant grid before the next round starts.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function COVandCONNAlgorithmAM()
% Implements coverage and connectivity control algorithm Active Model

    GlobalVars();
    VacantGrid = zeros(NumGridHeads);
    finished = 0;    %flag for end of algorithm
gridHeads(Heads(2))=0; % for sample - this must be change later

    while(finished==0)
         GridHeadsCommunicAM();

        for i=1 : NumGridHeads
            if(VacantGrid(Heads(i))~=0)
                finished = 0;
                break;
```

```
            else
                finished = 1;
            end
        end

        if(finished==0)
            CascadingMovementAM();
        end

    end

    %% For GridHeads connection
    for i=1 : NumGridHeads
        if(VacantGrid(Heads(i))==0)
            for j = 1 : Neighbors(Heads(i))
                    x1 = neighborTable(Heads(i),1);
                    y1 = neighborTable(Heads(i),2);
                    x2 = neighborTable(NeighboringHeads(Heads(i),j),1);
                    y2 = neighborTable(NeighboringHeads(Heads(i),j),2);
                    line([x1 x2], [y1 y2], 'Color', 'cyan', 'LineWidth',
2);
            end
        end
    end

      UpdateNodePower();
end

%% STEP 1: Oloi oi gridHeads elegxoun ean yparxei connection meta3y tous
%% alliws yparxei vacant grid kai 8a 3ekinisei cascading movement apo ton
%% 1o gridHead pou to anixneyse kai 8a enimerwsei tous ypoloipous me ena
%% notification kai to STEP 1 8a ginete epanaliptika efoson den pairnei
%% kanenas kapoio notification msg

function GridHeadsCommunicAM()

    GlobalVars();
    RandomHeads = randperm(NumGridHeads);

    for i=1 : NumGridHeads     % o ka8e gridHead elegxei

       if(gridHeads(Heads(RandomHeads(i)))~=0 &&
VacantGrid(Heads(RandomHeads(i)))==0)
        for j=1 : Neighbors(Heads(RandomHeads(i))) % ean yparxei connection
me ton ka8e geitoniko tou gridHead
         dist = CalcDistance(neighborTable(Heads(RandomHeads(i)),1),
neighborTable(Heads(RandomHeads(i)),2),
neighborTable(NeighboringHeads(Heads(RandomHeads(i)),j),1),
neighborTable(NeighboringHeads(Heads(RandomHeads(i)),j),2));

        %sensing, so lose power
        nodePower(RandomHeads(i)) = nodePower(RandomHeads(i)) - (dist/4);


if(VacantGrid(NeighboringHeads(Heads(RandomHeads(i)),j))~=Heads(RandomHeads
(i)))
```

```
            if(dist~=0)
                PrWLow = PtWLow / (dist ^ pathLossCoeff); % Calculate
reception power
            end
            %IsReceived(dist, 'topolDiscovery') == 0
         end

            if(gridHeads(NeighboringHeads(Heads(RandomHeads(i)),j))==0 &&
trustedNodes(NeighboringHeads(Heads(RandomHeads(i)),j))==0) %elegxos gia
vacant grid

                VacantGrid(Heads(RandomHeads(i))) =
NeighboringHeads(Heads(RandomHeads(i)),j);
                ProcessesInit = ProcessesInit + 1;
                break;
            else
                VacantGrid(Heads(RandomHeads(i))) = 0;
            end

        end
       end
    end
end

%% STEP 2: Oi gridHeads pou exo un anixneysei kapoio vacant grid 8a prepei
%% na psaxoun na broun kapoio trusted node gia na metakini8ei sto vacant
grid
%% kai na ginei aytos o gridHead tou vacant grid, alliws an den bre8ei na
steiloun
%% ena notification msg se ena geitona gridHead kai na metakini8ei o idios
sto vacant grid
%% kai na ginei se ayto o gridHead

function CascadingMovementAM();
    GlobalVars();
    RandomHeads = randperm(NumGridHeads);


    for i=1 : NumGridHeads

        if(VacantGrid(Heads(RandomHeads(i)))>0)

            movesNodes = movesNodes + 1; %ay3anw tous kombous pou kini8ikan

            x1 = neighborTable(Heads(RandomHeads(i)),1);
            y1 = neighborTable(Heads(RandomHeads(i)),2);
            x2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),1);
            y2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),2);
            plot(neighborTable(VacantGrid(Heads(RandomHeads(i))),1),
neighborTable(VacantGrid(Heads(RandomHeads(i))),2), 'x',
'MarkerEdgeColor','r', 'LineWidth',10);
            drawLineStatus(x1, x2, y1, y2, 'NoConnection');


          if(trustedNodes(Heads(RandomHeads(i)))>0) %elegxos ean yparxei
trusted node mesa sto grid pou einai arxigos
                %briskw ton 1o apo ta trusted nodes tou
```

```
                FoundTrustedNode =
trustOfHead(Heads(RandomHeads(i)),trustedNodes(Heads(RandomHeads(i))));
          else
              FoundTrustedNode = Heads(RandomHeads(i));
              %enimerwsi enos apo tous geitonikoys gridHeads me ta
              %perissotera trusted nodes
              NotificationAM(Heads(RandomHeads(i)));
          end


           % ypologismos apostasis pou metakineitai o trusted node 'i o
gridHead
              dist = CalcDistance(neighborTable(FoundTrustedNode,1),
neighborTable(FoundTrustedNode,2),
neighborTable(VacantGrid(Heads(RandomHeads(i))),1),
neighborTable(VacantGrid(Heads(RandomHeads(i))),2));
              moveDistances = moveDistances + dist; %ay3anw tin synoliki
apostasis metakinisis pou egine
              NodesMove(movesNodes) = FoundTrustedNode;

              if(dist~=0)
                PrWLow = PtWLow / (dist ^ pathLossCoeff); % Calculate
reception power
              end


              % moving, so lose power
              nodePower(FoundTrustedNode) = nodePower(FoundTrustedNode) -
(dist/2);

          if(FoundTrustedNode == Heads(RandomHeads(i))) % ean den bre8ike
trusted node



           %for w=1 :4
            %  if(RandomHeads(w)~=i)
             %   if(VacantGrid(RandomHeads(w))==0)
              %   VacantGrid(RandomHeads(w))=i;
               %  CascadingMovementAM();
                % break;
                %end
              %end
           %end

              %move & change gridhead
              drawLineStatus(x1, x2, y1, y2, 'CascadingMovement');
              %neighborTable(VacantGrid(i),1) =
              %neighborTable(VacantGrid(i),2) =

gridHeads(VacantGrid(Heads(RandomHeads(i))))=VacantGrid(Heads(RandomHeads(i
)));
              gridHeads(FoundTrustedNode)=0;

          else   % tote bre8ike trusted node k prepei na metakini8ei sto
vacant grid k na ginei o gridHead tou
              x1 = neighborTable(FoundTrustedNode,1);
              y1 = neighborTable(FoundTrustedNode,2);
```

```
              x2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),1);
              y2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),2);
              drawLineStatus(x1, x2, y1, y2, 'CascadingMovement');

              %metakinisi tou trusted node sto vacant grid kai ginete o
              %gridHead tou vacant grid
              neighborTable(FoundTrustedNode,1) =
neighborTable(VacantGrid(Heads(RandomHeads(i))),1);
              neighborTable(FoundTrustedNode,1) =
neighborTable(VacantGrid(Heads(RandomHeads(i))),2);
              gridHeads(FoundTrustedNode)=0;
              gridHeads(VacantGrid(Heads(RandomHeads(i))))=
VacantGrid(Heads(RandomHeads(i)));
              trustedNodes(Heads(RandomHeads(i))) =
trustedNodes(Heads(RandomHeads(i))) - 1; %feygei o trusted node kai ara
meiwnete kai o ari8mos twn trusted nodes toy gridHead
              %gridHeads(VacantGrid(i))=FoundTrustedNode;
              %gridHeads(FoundTrustedNode)=0;
              %....enimerwsi pws einai o new gridHead tou vacant grid


              %plotGrid();

          end

           plot(neighborTable(VacantGrid(Heads(RandomHeads(i))),1),
neighborTable(VacantGrid(Heads(RandomHeads(i))),2), '^',
'MarkerEdgeColor','b', 'LineWidth',2);
           VacantGrid(Heads(RandomHeads(i))) = 0;
        end

    end
end


function NotificationAM(gridHead)
% Send notification msg from the gridHead to one of its neighbor's
% gridHeads, that with more trusted nodes, to inform it for his cascading
movement to the vacant
% gridHead

    GlobalVars();
    tnodes = 0;
    headChoose = 0;

    for h = 1 : Neighbors(gridHead)

        if(NeighboringHeads(gridHead,h)~=VacantGrid(gridHead) &&
nodePower(gridHead) >= 1 && nodePower(NeighboringHeads(gridHead,h)) > 0 &&
tnodes<trustedNodes(NeighboringHeads(gridHead,h)))
            tnodes = trustedNodes(NeighboringHeads(gridHead,h));
            headChoose = NeighboringHeads(gridHead,h);
        end

    end

    if(headChoose~=0)
    x1 = neighborTable(gridHead,1);
```

```
    y1 = neighborTable(gridHead,2);
    x2 = neighborTable(headChoose,1);
    y2 = neighborTable(headChoose,2);

    drawLineStatus(x1, x2, y1, y2, 'MsgNotification');
    numTx = numTx + 1;
    nodePower(gridHead) = nodePower(gridHead) - 1;
    numRx = numRx + 1;
    nodePower(headChoose) = nodePower(headChoose) - RXtoTXratio;
    VacantGrid(headChoose) = gridHead;
    ProcessesInit = ProcessesInit + 1;

    end
end


function drawLineStatus(startX, endX, startY, endY, type)
    GlobalVars();
    switch (type)
        case 'MsgNotification'
            lineColor = 'b';
            style = '-';
        case 'CascadingMovement'
            lineColor = 'red';
            style = '--';
        case 'NoConnection'
            lineColor = 'm';
            style = '--';
        otherwise
    end
    line([startX endX], [startY endY], 'Color',lineColor, 'LineWidth',2,
'LineStyle',style);
    pause(0.8);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Clear Handles                                  %
%             Coverage & Connectivity Control in WSNs                   %
%                          Kakoulli Elena                               %
%                     Computer Science, UCY                             %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ClearHandles(handles)
% Auxiliary function which clear the handles of the main function
    GlobalVars();
    set(handles.analyzedNode, 'String','');
    set(handles.GridHead, 'String','');
    set(handles.NodesMove, 'String','');
    set(handles.pwrRem, 'String','');
    set(handles.discoverTime, 'String','');
    set(handles.source_sinkTime, 'String','');
    set(handles.numRX, 'String','');
    set(handles.droppedTX, 'String','');
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Calculate Distance                                  %
%             Coverage & Connectivity Control in WSNs                    %
%                        Kakoulli Elena                                  %
%                     Computer Science, UCY                              %
%                                                                        %
%                                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [d] = dist(x1, y1, x2, y2)
% Function for calculating the distance between two nodes
    d = sqrt(double((x1-x2)^2 + (y1-y2)^2));
end
```

Παράρτημα Γ

Κώδικας υλοποίησης των αλγορίθμων AM και SR που κατά την αρχική τοποθέτηση όλων των αισθητήρων τυχαία γίνεται μετά μετακίνηση του πιο κοντινού αισθητήρα μέσα σε κάθε πλαίσιο ως αρχηγός κάθε πλαισίου στο κέντρο του κάθε πλαισίου

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        WSN_COVandCONN_Sim                              %
%             Coverage & Connectivity Control in WSNs                    %
%                        Kakoulli Elena                                  %
%                     Computer Science, UCY                              %
%                                                                        %
%                                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = WSN_COVandCONN_Sim(varargin)
    gui_Singleton = 1;
```

```
    gui_State = struct('gui_Name', mfilename, 'gui_Singleton',
gui_Singleton, 'gui_OpeningFcn', @WSN_COVandCONN_Sim_OpeningFcn,
'gui_OutputFcn', @WSN_COVandCONN_Sim_OutputFcn, 'gui_LayoutFcn', [],
'gui_Callback', []);
    if (nargin && ischar(varargin{1}))
        gui_State.gui_Callback = str2func(varargin{1});
    end
    if (nargout)
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
end % End initialization code


% --- Executes just before WSN_CovANDConn_Sim is made visible.
function WSN_COVandCONN_Sim_OpeningFcn(hObject, eventdata, handles,
varargin)
    GlobalVars();
    %clear workspace; clc;
    handles.output = hObject; guidata(hObject, handles);
    axes(handles.GridAxes); cla;
    sensorLogoHandle = imshow('sensorLogo.jpg');
    axis([get(sensorLogoHandle, 'YData') get(sensorLogoHandle, 'XData')]);
    SetSamples(handles, 'initAll', 'hide'); % Set default values
    if (nodesDispersedF)
        ClearHandles(handles);
    end

    set(handles.COVandCONN_AM, 'Value',0);
    set(handles.COVandCONN_SR, 'Value',0);

    switch (COVandCONNAlg)
        case 'AM'
            set(handles.COVandCONN_AM, 'Value',1);
        case 'SR'
            set(handles.COVandCONN_SR, 'Value',1);
    end

    simFileInfo = dir('WSN_COVandCONN_Sim.m');
    set(handles.clearGrid, 'String','Quit');
    set(handles.statusText, 'String','Enter Grid parameters & choose
algorithm...');
    set(handles.analyzedNode, 'String','');
    set(handles.pwrRem, 'String','');
    set(handles.WSN_COVandCONN_Sim, 'Name', sprintf('WSN Coverage &
Connectivity Algorithms Simulator by elen@  -   %s', simFileInfo.date));
end

% --- Outputs from this function are returned to the command line.
function varargout = WSN_COVandCONN_Sim_OutputFcn(hObject, eventdata,
handles)
    varargout{1} = handles.output;
    reqVer = '7.1.0.19920 (R14)';
    if (version ~= reqVer) % Incorrect Matlab version detected
        msgbox('Matlab R14 SP3 is required to ensure proper GUI
performance.');
    end
```

```
end

% ================ BUTTON FUNCTIONS ================
function editParams_Callback(hObject, eventdata, handles)
    ParamEdit();
end

function recharge_Callback(hObject, eventdata, handles)
    GlobalVars();
    if (nodesDispersedF)
        nodePower = nodePowerMax .* ones(1, nodeCount); % Maximize the
power of each node
        %UpdateAnalyzedNode(handles, closestNode);
        UpdatePowerAxes(handles, 'noCalc');
        axes(handles.GridAxes);
        PlotGrid(); % Redraw the nodes to refresh their color
    end
end

function clearGrid_Callback(hObject, eventdata, handles)
    GlobalVars();
    if (networkSyncF)
        set(handles.source_sinkTime, 'String','');
        set(handles.numRX, 'String','');
        set(handles.droppedTX, 'String','');
        UpdateGridAxes(handles);
        DrawTopology(0);
        PlotGrid();
        networkSyncF = 0;
    elseif (networkLvlDiscF) % Clear the lvl-disc lines if no sync has been
performed
        ClearHandles(handles);
        UpdateGridAxes(handles);
        PlotGrid();
        set(handles.NodesMove, 'String','');
        set(handles.statusText, 'String','Enter network parameters & choose
algorithm...');
        networkLvlDiscF = 0;
    elseif (nodesDispersedF) % Clear nodes and reset to the opening screen
        WSN_COVandCONN_Sim_OpeningFcn(hObject, eventdata, handles, 1);
        axes(handles.PwrAxes); cla; axis([0 1 0 1]);
        set(handles.Moves, 'String','');
        set(handles.Distance, 'String','');
        set(handles.analyzedNode, 'String','');
        set(handles.pwrRem, 'String','');
        set(handles.clearGrid, 'String','Quit');
        nodesDispersedF = 0;
    else
        close;
    end
end

function plotGrid_Callback(hObject, eventdata, handles)
    GlobalVars(); clc;
    set(handles.clearGrid, 'String','Clear Grid');
    set(handles.statusText, 'String','Initializing Network, please
waiting...');
```

```
    if (~dataLoadedF) % Set variables here to avoid disrupting loaded
values
        SetSamples(handles, 'initArrays', 'hide'); % Set default values
        networkChangedF = 0;
        nodesDispersedF = 1;
        networkLvlDiscF = 0;
        networkSyncF = 0;


    else
        dataLoadedF = 0;
    end
    UpdateGridAxes(handles);
    PlotGrid();
    if (~networkChangedF && networkLvlDiscF)
    % Needed in case the data has been loaded and we want the discovering
lines...NetworkChangedF flag must be 0 to redraw the discovering lines
        DrawTopology(0);
    end
    UpdatePowerAxes(handles, 'noCalc');

    ClearHandles(handles);
    set(handles.Distance, 'String',sprintf('%g m', moveDistances));
    set(handles.Moves, 'String',sprintf('%g / %g',
movesNodes,ProcessesInit));

    set(handles.statusText, 'String','Enter network parameters & choose
algorithm...');
    %set(handles.analyzedNode, 'String',sourceNode);
    set(handles.pwrRem, 'String',nodePower(sourceNode));
    set(handles.topolDiscovery, 'Enable','on');
    set(handles.run, 'Enable','on');
end

function topolDiscovery_Callback(hObject, eventdata, handles)
    GlobalVars(); clc;
    if (networkChangedF || ~nodesDispersedF || dataLoadedF)
        plotGrid_Callback(hObject, eventdata, handles);
    end

    maxDistanceLow = nthroot(PtWLow/PrThresholdWLow, pathLossCoeff);
    maxDistanceHigh = nthroot(PtWHigh/PrThresholdWHigh, pathLossCoeff);

    set(handles.statusText, 'String','Simulating Topology Discovery, please
waiting...');
    set(handles.plotGrid, 'Enable','off');
    set(handles.clearGrid, 'Enable','off');
    set(handles.topolDiscovery, 'Enable','off');
    set(handles.run, 'Enable','off');
    set(handles.COVandCONN_AM, 'Enable','off');
    set(handles.COVandCONN_SR, 'Enable','off');
    set(handles.editParams, 'Enable','off');
    set(handles.recharge, 'Enable','off');
    pause(0.001);
    UpdateGridAxes(handles);
    PlotGrid();

    s = cputime;
```

```
    discoverTime = cputime - s;

    DrawTopology(pauseInt);
    ClearHandles(handles);
    PlotGrid();
    UpdatePowerAxes(handles);
    %UpdateAnalyzedNode(handles, closestNode);
    UpdateNodeStatus(handles);
    set(handles.plotGrid, 'Enable','on');
    set(handles.clearGrid, 'Enable','on');
    set(handles.topolDiscovery, 'Enable','on');
    set(handles.run, 'Enable','on');
    set(handles.COVandCONN_AM, 'Enable','on');
    set(handles.COVandCONN_SR, 'Enable','on');
    set(handles.editParams, 'Enable','on');
    set(handles.recharge, 'Enable','on');
    set(handles.discoverTime, 'String',discoverTime);
    set(handles.numRX, 'String',numRx);
    networkLvlDiscF = 1;
    networkSyncF = 0;
    set(handles.statusText, 'String','Topology Discovery completed!!');
end

function run_Callback(hObject, eventdata, handles)
    GlobalVars();
    networkSyncF = 1;
   if (networkChangedF || ~networkLvlDiscF)
        topolDiscovery_Callback(hObject, eventdata, handles);
        if (continuousF)
            %pause(1);
        end
    end


    set(handles.plotGrid, 'Enable','off');
    set(handles.clearGrid, 'Enable','off');
    set(handles.topolDiscovery, 'Enable','off');
    set(handles.run, 'Enable','off');
    set(handles.COVandCONN_AM, 'Enable','off');
    set(handles.COVandCONN_SR, 'Enable','off');
    set(handles.editParams, 'Enable','off');
    set(handles.recharge, 'Enable','off');
    set(handles.statusText, 'String','Grid Heads discovering...');
    pause(0.001);
    UpdateGridAxes(handles);
    DrawTopology(0);
    PlotGrid();

    numTx = 0;
    numRx = 0;
    dropTx = 0;
    s = cputime;

    switch (COVandCONNAlg)
        case 'AM'
            COVandCONNAlgorithmAM();
        case 'SR'
            COVandCONNAlgorithmSR();
```

```
    end

    source_sinkTime = cputime - s;

    % Continuously run
        while (continuousF && source_sinkTime <= 10)
            pause(1);
            UpdateGridAxes(handles);
            DrawTopology(0);
            PlotGrid();
            UpdatePowerAxes(handles);
            pause(1);


            numTx = 0;
            numRx = 0;
            dropTx = 0;
            UpdateGridAxes(handles);
            DrawTopology(0);
            PlotGrid();
            s = cputime;

            switch (COVandCONNAlg)
                case 'AM'
                    COVandCONNAlgorithmAM();
                case 'SR'
                    COVandCONNAlgorithmSR();
            end

            source_sinkTime = cputime - s;
            %UpdateAnalyzedNode(handles, closestNode);
            UpdateNodeStatus(handles);
            networkSyncF = 1;
        end

    UpdatePowerAxes(handles);
    UpdateAnalyzedNode(handles, 1);
    UpdateNodeStatus(handles);
    set(handles.plotGrid, 'Enable','on');
    set(handles.clearGrid, 'Enable','on');
    set(handles.topolDiscovery, 'Enable','on');
    set(handles.run, 'Enable','on');
    set(handles.COVandCONN_AM, 'Enable','on');
    set(handles.COVandCONN_SR, 'Enable','on');
    set(handles.editParams, 'Enable','on');
    set(handles.recharge, 'Enable','on');
    set(handles.source_sinkTime, 'String',source_sinkTime);
    set(handles.numRX, 'String',numRx);
    set(handles.droppedTX, 'String',DeadNodes);
    set(handles.statusText, 'String','Simulation completed!!!');
end

% ------------------- NON-BUTTON CALLBACKS -------------------
function continuous_Callback(hObject, eventdata, handles)
    GlobalVars();
    continuousF = get(gco,'Value');
end
```

```
function COVandCONN_AM_Callback(hObject, eventdata, handles)
    GlobalVars();
    set(handles.COVandCONN_AM, 'Value',1);
    set(handles.COVandCONN_SR, 'Value',0);
    COVandCONNAlg = 'AM';
end

function COVandCONN_SR_Callback(hObject, eventdata, handles)
    GlobalVars();
    set(handles.COVandCONN_AM, 'Value',0);
    set(handles.COVandCONN_SR, 'Value',1);
    COVandCONNAlg = 'SR';
end

function GridAxes_ButtonDownFcn(hObject, eventdata, handles)
    GlobalVars();
% Check to make sure that the nodes have been plotted and the user has
either performed network level discovery or wants to change the source node
    if (networkChangedF || dataLoadedF)
        plotGrid_Callback(hObject, eventdata, handles);
        tempStr = get(handles.statusText, 'String');
        tempColor = get(handles.statusText, 'BackgroundColor');
        set(handles.statusText, 'String','Network parameters changed!!');
        set(handles.statusText, 'BackgroundColor','red');
        pause(2);
        set(handles.statusText, 'String',tempStr);
        set(handles.statusText, 'BackgroundColor',tempColor);
        networkChangedF = 0;
        dataLoadedF = 0;
    elseif (nodesDispersedF)
        pt = get(gca,'currentpoint'); % Find the nearest node to the mouse-
click
        closestDist = xDistance;
        for i = 1 : nodeCount
            d = CalcDistance(pt(1,1), pt(1,2), neighborTable(i,1),
neighborTable(i,2));
            if  (d < closestDist)
                closestNode = i; % Nearest node to mouse-click after loop
is complete
                closestDist = d;
            end
        end

        if (changeSourceNodeF) % Change source node
            sourceNode = closestNode;
            closestNode = sourceNode;
            ClearHandles(handles);

            axes(handles.GridAxes);
            cla; axis([0 xDistance 0 yDistance]); axis on; axis xy; axis
fill; axis manual; hold all;
            PlotGrid();
            set(handles.Moves, 'String','');
            set(handles.statusText, 'String','Enter topolDiscovery
parameters...');
            networkLvlDiscF = 0;
            networkSyncF = 0;
            UpdateAnalyzedNode(handles, closestNode);
```

```
        else % Analyze node's properties
            PlotGrid();
            UpdateAnalyzedNode(handles, closestNode);
            if (networkLvlDiscF)
                PlotGrid();
                if (tracebackF && gridHeads(closestNode) ~= 0)
                    UpdateGridAxes(handles);
                    DrawTopology(0);
                    PlotGrid();

                end
            end
        end
    end
end

function PwrAxes_ButtonDownFcn(hObject, eventdata, handles)
    GlobalVars();
    pt = get(gca,'currentpoint'); % Find the nearest node to the mouse-
click
    closestDist = length(gridPower);
    for i = 1 : length(gridPower)
        d = CalcDistance(pt(1,1), pt(1,2), i, gridPower(i));
        if  (d < closestDist)
            closestPt = i; % Nearest event to mouse-click after loop is
complete
            closestDist = d;
        end
    end
end

% ================== MENU FUNCTIONS ==================
function Menu_LoadPar_Callback(hObject, eventdata, handles) % FILE
FUNCTIONS
    GlobalVars();
    [fname pname]= uigetfile('*.mat', 'Open');
    if ((ischar(fname))& (ischar(pname)))
        curdir = pwd;
        cd(pname);
        load(fname); % Load the entire workspace
        cd(curdir);
        SetSamples(handles, 'load','hide');
        dataLoadedF = 1;
        msgbox('Parameters loaded succesfully!');
    end
end

function Menu_SavePar_Callback(hObject, eventdata, handles)
    GlobalVars();
    [fname pname] = uiputfile('parameters.mat', 'Save As');
    if ((ischar(fname))& (ischar(pname)))
        curdir = pwd;
        cd(pname);
        fid = fopen(fname,'wt');
        save(fname, '*'); % Save the entire workspace
        fclose(fid);
        cd(curdir);
        networkLvlDiscF
```

```
        msgbox(sprintf('%s%s', 'Parameters successfully saved to ',
fname));
    end
end

function Menu_Exit_Callback(hObject, eventdata, handles)
    %clear all;
    if (isdeployed)
        close all;
        quit force;
    else
        close all;
    end
end

function Menu_View_NodeIDs_Callback(hObject, eventdata, handles) % VIEW
FUNCTIONS
    GlobalVars();
    if strcmp(get(gcbo,'Checked'),'on')
        viewNodeIDF = 0;
        set(gcbo, 'Checked','off');
    else
        viewNodeIDF = 1;
        set(gcbo, 'Checked','on');
    end
    if (nodesDispersedF) % Update the grid axes only if the nodes have been
dispersed
        UpdateGridAxes(handles);
        if (networkLvlDiscF) % Draw discovering lines only if network has
already been lvl-discovered
            DrawTopology(0);
        end
        PlotGrid();
    end
end

function Menu_MatlabVer_Callback(hObject, eventdata, handles) % HELP
FUNCTIONS
    msgbox(sprintf('Current Matlab Version:\n%s', version), 'Matlab
Version');
end

function Menu_Info_Callback(hObject, eventdata, handles)
    titleStr = 'WSN Coverage & Connectivity Control Simulator';
    authorStr = sprintf('%s\n%s', 'Elena Kakoulli, UCY');
    simFileInfo = dir('WSN_COVandCONN_Sim.m');
    msgbox(sprintf('%s\n%s\n\nLast Updated on %s', titleStr, authorStr,
simFileInfo.date), 'Info');
end

% ================== OTHER FUNCTIONS ==================

function UpdateAnalyzedNode(handles, node)
    GlobalVars();
    set(handles.analyzedNode, 'String',node);
    set(handles.pwrRem, 'String',nodePower(node));
    if (networkLvlDiscF) % Output discovering details
```

```
        if (gridHeads(node) == 0)
            set(handles.GridHead, 'String','Orphan');
        else
            set(handles.GridHead, 'String',num2str(gridHeads(node)));
        end

        set(handles.NodesMove,
'String',num2str(NodesMove(1:movesNodes,1)));

        if (nodePower(node) <= 0)
            set(handles.pwrRem, 'String','Depleted');
        else
            set(handles.pwrRem, 'String',num2str(nodePower(node)));
        end
    end
end

function UpdateNodeStatus(handles)
    GlobalVars();

    for i = 1 : nodeCount
        if (nodePower(i) <= 0)
            gridHeads(i) = 0;
        end
    end

    set(handles.Distance, 'String',sprintf('%g m', moveDistances));
    set(handles.Moves, 'String',sprintf('%g / %g',
movesNodes,ProcessesInit));

end

function UpdateGridAxes(handles)
    GlobalVars();
    axes(handles.GridAxes); cla;
    axis([0 xDistance 0 yDistance]);
    axis on; axis xy; axis fill; axis manual; hold all;
end

function UpdatePowerAxes(handles, x)
    GlobalVars();
    if (nargin == 2 & x == 'noCalc')
        gridPower = [1];
    else
        gridPower = [gridPower sum(nodePower)/(nodePowerMax*nodeCount)] %
Calculate new normalized grid power
    end
    axes(handles.PwrAxes); cla;
    axis([1 length(gridPower)+1 0 1]); set(handles.PwrAxes, 'YGrid','on',
'YMinorGrid','on');
    axis on; axis xy; axis fill; axis manual; hold all;
    plot(gridPower, '-b*');
end


% --- Executes during object creation, after setting all properties.
function WSN_COVandCONN_Sim_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WSN_COVandCONN_Sim (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      Update Node Power                               %
%             Coverage & Connectivity Control in WSNs                  %
%                        Kakoulli Elena                                %
%                    Computer Science, UCY                             %
%                                                                      %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function UpdateNodePower()
    GlobalVars();
    for i = 1 : nodeCount
        if (nodePower(i) <= 0) % Node i has run out of energy
            DeadNodes = DeadNodes + 1;
            nodeLevel(i) = -1;
            gridHeads(i) = 0;
        end
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Set Samples                                 %
%             Coverage & Connectivity Control in WSNs                  %
%                          Kakoulli Elena                              %
%                    Computer Science, UCY                             %
%                                                                      %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function SetSamples(handles, sampleID, visibility)
    GlobalVars();
    switch sampleID
        case 'initAll'
            SetSamples(handles, 'mica2dot433', 'hide'); % Set default
values
            propSpd = 2.998e8;
            pauseInt = 0.2;
            xDistance = 80;
            yDistance = 100;
            nodeCount = 70;
            nodeDistribMode = 'rand';
            pathLossCoeff = 3.5;
            nodePowerMax = 100;
            nodePowerWarn = 10;
            rediscoverLimit = 20;
            COVandCONNAlg = 'AM';

        % Set flags and source node
            dataLoadedF = 0;
```

```
            nodesDispersedF = 0;
            networkChangedF = 0;
            networkLvlDiscF = 0;
            networkSyncF = 0;
            viewNodeIDF = 0;
            changeSourceNodeF = 0;
            selectNodeF = 0;
            tracebackF = 0;
            continuousF = 0;
            sourceNode = 1;
            gridHead1 = 1;
            gridHead2 = 2;
            gridHead3 = 3;
            gridHead4 = 4;


            SetSamples(handles, 'initArrays', 'hide');
        case 'initArrays'
            DeadNodes = 0;
            NodesMove = zeros(nodeCount,1);
            trustOfHead = [];
            Heads = [];
            trustedNodes = [];
            ProcessesInit = 0;
            Neighbors = [];
            NeighboringHeads = [];
            NumGridHeads = 0;
            movesNodes= 0;
            moveDistances = 0;
            VacantGrid = [];
            gridHeads = zeros(nodeCount,1);
            FoundTrustedNode = 0;
            neighborTable = [];
            parent = [];
            numChildren = [];
            children = [];
            deadNodeList = [];
            deadNodes = 0;
            orphanNodes = 0;
            numTx = 0;
            numRx = 0;
            dropTx = 0;
            RXtoTXratio = RXCurDraw / TXCurDraw;
            receiver_threshold = (3 + sqrt(9 + 8/RXtoTXratio)) / 2; %
Calculate point at which hybrid sync'ing switches from RBS to TPSN
            nodePower = nodePowerMax .* ones(1, nodeCount); % Set the power
of each node to max
            gridPower = [1]; % Set the grid's initial normalized power to 1
(sum of all fully charged nodes)
            closestNode = sourceNode;
            PtWLow = 10 ^ (PtLow/10); % Convert to watts
            PrThresholdWLow = 10 ^ (PrThresholdLow/10); % Convert to watts
            maxDistanceLow = nthroot(PtWLow/PrThresholdWLow,
pathLossCoeff);
            PtWHigh = 10 ^ (PtHigh/10); % Convert to watts
            PrThresholdWHigh = 10 ^ (PrThresholdHigh/10); % Convert to
watts
```

```
            maxDistanceHigh = nthroot(PtWHigh/PrThresholdWHigh,
pathLossCoeff);

            if (nodeDistribMode == 'rand') % Random distribution

                 neighborTable = rand([nodeCount 2]);

                %%%%%%%%% HOW MANY GRIDS SO HOW MANY GRIDHEADS %%%%%%%%%%
                NumGridHeads = floor(xDistance / 20) * floor(yDistance /
20);
                if(NumGridHeads==0)
                    NumGridHeads = 1;
                end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %FOR GRIDHEADS
                xCount = floor(xDistance / 20);
                yCount = floor(yDistance / 20);

                Heads = zeros(NumGridHeads,1);
                trustedNodes = zeros(nodeCount,1);
                trustOfHead = zeros(nodeCount,nodeCount);

                for i = 1 : nodeCount
                    neighborTable(i,1) = neighborTable(i,1) * xDistance;
                    neighborTable(i,2) = neighborTable(i,2) * yDistance;
                end

                dist = 20;
                num = 0;

                for j = 1 : yCount
                    for i = 1 : xCount
                        num = num + 1;
                        Heads(num) = 0;
                        maxDist = CalcDistance(0, 0, xDistance, yDistance);

                        for k = 1 : nodeCount

                            if(neighborTable(k,1)<=i*dist &&
neighborTable(k,2)<=j*dist && neighborTable(k,1)>=((i*dist)-20) &&
neighborTable(k,2)>=((j*dist)-20))
                                FromCenter = CalcDistance(i*dist - 0.5*dist
, j*dist - 0.5*dist, neighborTable(k,1), neighborTable(k,2));

                                if(maxDist > FromCenter)    % evresi tou
pio kontinou sensor
                                    maxDist = FromCenter;
                                    Heads(num) = k;
                                end
                            end
                        end
                        if(Heads(num)~=0)
                            gridHeads(Heads(num)) = Heads(num);  % bre8ike
o pio kontinos sensor
                            Neighbors(Heads(num))=0;
                            neighborTable(Heads(num),1) = i*dist -
0.5*dist;
```

```
                            neighborTable(Heads(num),2) = j*dist -
0.5*dist;
                            FromCenter = CalcDistance(i*dist - 0.5*dist ,
j*dist - 0.5*dist, neighborTable(k,1), neighborTable(k,2));

                            % moving to center, so lose power
                            nodePower(Heads(num)) = nodePower(Heads(num)) -
(FromCenter/2);
                            moveDistances = moveDistances + FromCenter;
%ay3anw tin synoliki apostasis metakinisis pou egine
                          for k = 1 : nodeCount
                             if(k~=Heads(num) &&
neighborTable(k,1)<=i*dist && neighborTable(k,2)<=j*dist &&
neighborTable(k,1)>=((i*dist)-20) && neighborTable(k,2)>=((j*dist)-20))
                                gridHeads(k)=Heads(num);
                                trustedNodes(Heads(num)) =
trustedNodes(Heads(num)) + 1;

trustOfHead(Heads(num),trustedNodes(Heads(num))) = k;
                             end
                          end
                        end

                    end
                end

                 num = 1;

                   for j = 1 : yCount
                    for i = 1 : xCount

                        if(Heads(num)==0)
                            bestMove = CalcDistance(0, 0,
xDistance,yDistance);
                            apostasi = 0;
                            HeadChoice = 0;
                            trustedNode = 0;
                            for w=1 : NumGridHeads
                                if(Heads(w)~=0 && trustedNodes(Heads(w))>0)

                                   for n = 1: trustedNodes(Heads(w))

                                    apostasi = CalcDistance(i*dist -
0.5*dist, j*dist - 0.5*dist, neighborTable(trustOfHead(Heads(w),n),1),
neighborTable(trustOfHead(Heads(w),n),2));

                                    if(bestMove > apostasi) %ean exei tin
mikroteri apostasi
                                        bestMove = apostasi;
                                        Heads(num) =
trustOfHead(Heads(w),n);
                                        HeadChoice = Heads(w);
                                        trustedNode = n;


                                    end

                                   end
```

```
                                end
                            end

                            if(Heads(num)~=0)
                                gridHeads(Heads(num)) = Heads(num);
                                Neighbors(Heads(num))=0;
                                neighborTable(Heads(num),1) = i*dist -
0.5*dist;
                                neighborTable(Heads(num),2) = j*dist -
0.5*dist;

                                trustOfHead(HeadChoice,trustedNode) =
trustOfHead(HeadChoice,trustedNodes(HeadChoice));
                                trustedNodes(HeadChoice) =
trustedNodes(HeadChoice) - 1;

                                % moving to center, so lose power
                                nodePower(Heads(num)) =
nodePower(Heads(num)) - (bestMove/2);
                                 moveDistances = moveDistances + bestMove;
%ay3anw tin synoliki apostasis metakinisis pou egine
                            end
                        end

                          num = num + 1;
                    end
                   end

             %%%%%%%%%%%%%%%% DIMIOURGIA GEITONWN %%%%%%%%%%%%%%%%%%%%
            if(strcmp(COVandCONNAlg, 'AM'))

                for i=1 : NumGridHeads

                    for j=1 : NumGridHeads
                        if(j~=i)
                            %dist = CalcDistance(neighborTable(Heads(i),1),
neighborTable(Heads(i),2), neighborTable(Heads(j),1),
neighborTable(Heads(j),2));

                            if(neighborTable(Heads(j),1) <=
(neighborTable(Heads(i),1)+20) &&
neighborTable(Heads(j),2)<=(neighborTable(Heads(i),2)+20) &&
neighborTable(Heads(j),1)>=(neighborTable(Heads(i),1)-20) &&
neighborTable(Heads(j),2)>=(neighborTable(Heads(i),2)-20))
                                Neighbors(Heads(i)) =
Neighbors(Heads(i))+1;

NeighboringHeads(Heads(i),Neighbors(Heads(i)))=Heads(j);
                            end
                        end
                    end
                end
              end

              %%%%%%%%%%%%%%% DIMIOURGIA HAMILTON CYCLE %%%%%%%%%%%%%%%%%%%%
            if(strcmp(COVandCONNAlg, 'SR'))
```

```
                    if(xCount>=2 || yCount>=2) % proipo8esi gia na ypar3ei
Hamilton Cycle
                     %HAMILTON CYCLE
                      if(rem(xCount,2)==0)     % The columns have even
number of GridHeads so we have a Hamilton cycle path
                           start = NumGridHeads-(xCount-1);
                           successors(start) =  start-xCount;
                           precedings(start) = start+1;
                           start=start-xCount;
                           for i=2: yCount-1      %pros ta katw
                               successors(start) =  start-xCount;
                               precedings(start) = start+xCount;
                               start=start-xCount;
                           end
                           successors(1) =  2;
                           precedings(1) = xCount+1;
                           for i=2: xCount-1      %PROS TA DE3IA
                               successors(i) = i+1;
                               precedings(i) = i-1;
                           end
                            successors(xCount) = 2*xCount;
                            precedings(xCount) = xCount-1;

                           start=2*xCount;
                           for i=2 : yCount-1 %pros ta panw
                               successors(start) = start+xCount;
                               precedings(start) = start-xCount;
                               start = start+xCount;
                           end

                           successors(NumGridHeads) = NumGridHeads-1;
                           precedings(NumGridHeads) = NumGridHeads-xCount;


                           if(xCount>2)   % yparxoun mesaies grammes
                            stop = (xCount/2)-1;
                            start = NumGridHeads-1;
                             for point=1 : stop
                                  successors(start) = start-xCount;
                                  precedings(start) = start+1;
                                  start = start-xCount;
                                  for i=1 : yCount-3   %pros ta katw
                                  successors(start) =  start-xCount;
                                  precedings(start) = start+xCount;
                                  start = start-xCount;
                                  end
                                  successors(start) =  start-1;
                                  precedings(start) = start+xCount;
                                  successors(start-1) =  start-1+xCount;
                                  precedings(start-1) = start;
                                  start = start-1 + xCount;

                                  for i=1 : yCount-3   %pros ta panw
                                  successors(start) =  start+xCount;
                                  precedings(start) = start-xCount;
                                  start = start+xCount;
                                  end
                                  successors(start) =  start-1;
```

```
                precedings(start) = start-xCount;
                start = start-1;
            end

        end
    elseif(rem(yCount,2)==0) % The rows have even number
of GridHeads so we have a Hamilton cycle path
        successors(1) = 2;
        precedings(1) = 1+xCount;

        for i=2: xCount-1  %pros ta de3ia
            successors(i) =  i+1;
            precedings(i) = i-1;
        end
        successors(xCount) =  xCount*2;
        precedings(xCount) = xCount-1;
        start = 2*xCount;
        for i=2: yCount-2      %pros ta panw
            successors(start) = start+xCount;
            precedings(start) = start-xCount;
            start = start+xCount;
        end

        successors(NumGridHeads) =  NumGridHeads-1;
        precedings(NumGridHeads) = NumGridHeads-xCount;
        start = start-1;

        for i=2 : xCount-2     %pros ta aristera

            successors(start) = start-1;
            precedings(start) = start+1;
            start = start-1;
        end

        start=NumGridHeads-xCount+1;
        successors(start) = start-xCount;
        precedings(start) = start+1;


        if(yCount>2)   % yparxoun mesaies grammes
         stop = (yCount/2)-1;
         start = start-xCount;

           for point=1 : stop
               successors(start) = start+1;
               precedings(start) = start+xCount;

               for i=1 : xCount-3   %pros ta de3ia
                   start = start+1;
               successors(start) =  start+1;
               precedings(start) = start-1;
               end

               successors(start+1) = start+1-xCount;
               precedings(start+1) = start;

               start=start+1-xCount;
               successors(start) =  start-1;
```

```
                            precedings(start) = start+xCount;
                            start = start-1;

                            for i=1 : xCount-3   %pros aristera
                            successors(start) =  start-1;
                            precedings(start) = start+1;
                            start = start-1;
                            end
                            successors(start) =  start-xCount;
                            precedings(start) = start+1;
                            start = start-xCount;
                         end

                     end


                 else
                     % The MxN network don't have even columns or rows
                     % so we have DUAL Hamilton cycle paths

                 end
               end
           end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
            end

          if (nodeDistribMode == 'grid') % Uniform grid distribution
                dist = sqrt(xDistance*yDistance / nodeCount);
                xCount = floor(xDistance / dist);
                yCount = floor(yDistance / dist);
                if (xCount*yCount < nodeCount)
                    if (xDistance > yDistance)
                        xCount = xCount + 1;
                        if (xCount*yCount < nodeCount)
                            yCount = yCount + 1;
                        end
                    else
                        yCount = yCount + 1;
                        if (xCount*yCount < nodeCount)
                            xCount = xCount + 1;
                        end
                    end
                end
                dist = min(xDistance/xCount, yDistance/yCount);
                for j = 1 : yCount
                    for i = 1 : xCount
                        neighborTable = [neighborTable; i*dist - 0.5*dist
j*dist - 0.5*dist];
                    end
                end
          end


        case 'mica2dot433'
            moteClockSpd = 4;
```

```
        radioFreq = 433;
        TXCurDraw = 25;
        RXCurDraw = 8;
        PtLow = -2;
        PrThresholdLow = -85;
        PtHigh = -2;
        PrThresholdHigh = -101;
        guarDistance = 0;
    case 'mica2dot916'
        moteClockSpd = 4;
        radioFreq = 916;
        TXCurDraw = 27;
        RXCurDraw = 10;
        PtLow = -2;
        PrThresholdLow = -85;
        PtHigh = -2;
        PrThresholdHigh = -98;
        guarDistance = 0;
    case 'micaz'
        moteClockSpd = 4;
        radioFreq = 2400;
        TXCurDraw = 17.4;
        RXCurDraw = 19.7;
        PtLow = -5;
        PrThresholdLow = -85;
        PtHigh = -5;
        PrThresholdHigh = -94;
        guarDistance = 0;
    case 'load' % Leave variables as they are
    otherwise
end

if (visibility == 'show')
    set(handles.xDist, 'String', xDistance);
    set(handles.yDist, 'String', yDistance);
    set(handles.nodeCount, 'String', nodeCount);
    if (nodeDistribMode == 'grid')
        set(handles.gridMode, 'Value',1);
        set(handles.randMode, 'Value',0);
    elseif (nodeDistribMode == 'rand')
        set(handles.randMode, 'Value',1);
        set(handles.gridMode, 'Value',0);
    end

    set(handles.TxPwrLow, 'String',PtLow);
    set(handles.RxThreshLow, 'String',PrThresholdLow);
    set(handles.TxPwrHigh, 'String',PtHigh);
    set(handles.RxThreshHigh, 'String',PrThresholdHigh);
    set(handles.pathLossCoeff, 'String',pathLossCoeff);
    set(handles.guarDist, 'String',guarDistance);
    set(handles.pauseInt, 'String',pauseInt);
    set(handles.maxTxDistLow, 'String',maxDistanceLow);
    set(handles.maxTxDistHigh, 'String',maxDistanceHigh);

    set(handles.moteCPU, 'String',moteClockSpd);
    set(handles.radioFreq, 'String',radioFreq);
    set(handles.TXCurDraw, 'String',TXCurDraw);
    set(handles.RXCurDraw, 'String',RXCurDraw);
```

```
        set(handles.nodePowerMax, 'String',nodePowerMax);
        set(handles.nodePowerWarn, 'String',nodePowerWarn);
        set(handles.rediscoverLimit, 'String',rediscoverLimit);
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Plot Grid                                      %
%            Coverage & Connectivity Control in WSNs                    %
%                        Kakoulli Elena                                 %
%                    Computer Science, UCY                              %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function PlotGrid()
% Plot each of the nodes according to their role and status in the network
    GlobalVars();

    for i=1 : NumGridHeads
          s = sprintf('%d', Heads(i));

            %nodeColor = checkNodePwrLevel(i, 'b');
            if (viewNodeIDF)
                plot(neighborTable(Heads(i),1), neighborTable(Heads(i),2),
'.', 'MarkerEdgeColor','b', 'LineWidth',2);
                text(neighborTable(Heads(i),1) + xDistance/100,
neighborTable(Heads(i),2), s, 'FontSize',8, 'FontWeight','bold',
'HorizontalAlignment','left', 'Color',nodeColor);
            else
                plot(neighborTable(Heads(i),1), neighborTable(Heads(i),2),
'^', 'MarkerEdgeColor','b', 'LineWidth',2);
            end
    end

    for i = 1 : nodeCount
        s = sprintf('%d', i);
        isHead = 0;
       if(gridHeads(i)>0) %den metakini8ike allou
            nodeColor = checkNodePwrLevel(i, 'k');
            %if(gridHeads(i)==1)
            %    plot(neighborTable(i,1), neighborTable(i,2), 'x',
'MarkerEdgeColor','r', 'LineWidth',4);
            %else
            for j=1 : NumGridHeads
                if(Heads(j)==i)
                    isHead = 1;
                    break;
                end
            end
            if(isHead==0)
                if (viewNodeIDF)
                plot(neighborTable(i,1), neighborTable(i,2), '.',
'MarkerEdgeColor',nodeColor, 'LineWidth',2);
                text(neighborTable(i,1) + xDistance/100,
neighborTable(i,2), s, 'FontSize',8, 'HorizontalAlignment','left',
'Color',nodeColor);
```

```
                else
                plot(neighborTable(i,1), neighborTable(i,2), 'o',
'MarkerEdgeColor',nodeColor, 'LineWidth',2);
                end
            end
        end
    end
end

function [color] = checkNodePwrLevel(curNode, stdColor)
% Change the node's color if its power level has decreased below the
warning level
    GlobalVars();
    if (nodePower(curNode) > nodePowerWarn)
        color = stdColor;
    elseif (nodePower(curNode) <= nodePowerWarn && nodePower(curNode) > 0)
        color = 'y'; % Yellow for warning
    else
        color = 'r'; % Red for depleted
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      Edit Parameters                                  %
%            Coverage & Connectivity Control in WSNs                    %
%                        Kakoulli Elena                                 %
%                    Computer Science, UCY                              %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = ParamEdit(varargin)
    gui_Singleton = 1;
    gui_State = struct('gui_Name', mfilename, 'gui_Singleton',
gui_Singleton, 'gui_OpeningFcn', @ParamEdit_OpeningFcn, 'gui_OutputFcn',
@ParamEdit_OutputFcn, 'gui_LayoutFcn', [], 'gui_Callback', []);
    if (nargin && ischar(varargin{1}))
        gui_State.gui_Callback = str2func(varargin{1});
    end
    if (nargout)
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
end % End initialization code

% --- Executes just before the editor is made visible.
function ParamEdit_OpeningFcn(hObject, eventdata, handles, varargin)
    GlobalVars();
    handles.output = hObject; guidata(hObject, handles);
    SetSamples(handles, 'load', 'show');
end

% --- Outputs from this function are returned to the command line.
function varargout = ParamEdit_OutputFcn(hObject, eventdata, handles)
    varargout{1} = handles.output;
end
```

```
function gridParams_Callback(hObject, eventdata, handles)
    GlobalVars();
    if (str2double(get(handles.xDist, 'String')) == 0)
        set(hObject, 'String', xDistance);
    end
    if (str2double(get(handles.yDist, 'String')) == 0)
        set(hObject, 'String', yDistance);
    end
    if (str2double(get(handles.nodeCount, 'String')) == 0)
        set(hObject, 'String', nodeCount);
    end

    if (str2double(get(handles.xDist, 'String')) ~= xDistance || ...
            str2double(get(handles.yDist, 'String')) ~= yDistance || ...
            str2double(get(handles.nodeCount, 'String')) ~= nodeCount)
        networkChangedF = 1;
    end
end

function gridMode_Callback(hObject, eventdata, handles)
    GlobalVars();
    set(handles.gridMode, 'Value', 1);
    set(handles.randMode, 'Value', 0);
end

function randMode_Callback(hObject, eventdata, handles)
    GlobalVars();
    set(handles.gridMode, 'Value', 0);
    set(handles.randMode, 'Value', 1);
end

function signalParams_Callback(hObject, eventdata, handles)
    GlobalVars();
    pathLossCoeffTemp = str2double(get(handles.pathLossCoeff, 'String'));
    PtLTemp = str2double(get(handles.TxPwrLow, 'String'));
    PtWLTemp = 10^(PtLTemp/10) / 1000; % Convert to watts
    PrThresholdLTemp = str2double(get(handles.RxThreshLow, 'String'));
    PrThresholdWLTemp = 10^(PrThresholdLTemp/10) / 1000; % Convert to watts
    set(handles.maxTxDistLow, 'String', nthroot(PtWLTemp/PrThresholdWLTemp,
pathLossCoeffTemp));

    PtHTemp = str2double(get(handles.TxPwrHigh, 'String'));
    PtWHTemp = 10^(PtHTemp/10) / 1000; % Convert to watts
    PrThresholdHTemp = str2double(get(handles.RxThreshHigh, 'String'));
    PrThresholdWHTemp = 10^(PrThresholdHTemp/10) / 1000; % Convert to watts
    set(handles.maxTxDistHigh, 'String',
nthroot(PtWHTemp/PrThresholdWHTemp, pathLossCoeffTemp));
end

function resetParams_Callback(hObject, eventdata, handles)
    GlobalVars();
    SetSamples(handles, 'mica2dot433', 'show');
    close;
end

function cancelParams_Callback(hObject, eventdata, handles)
    networkChangedF = 0;
```

```
    close;
end

function saveParams_Callback(hObject, eventdata, handles)
    GlobalVars();
    if (xDistance ~= str2double(get(handles.xDist, 'String')) ...
            || yDistance ~= str2double(get(handles.yDist, 'String')) ...
            || nodeCount ~= str2double(get(handles.nodeCount, 'String')))
        networkChangedF = 1;
    else
        networkChangedF = 0;
    end

    xDistance = str2double(get(handles.xDist, 'String'));
    yDistance = str2double(get(handles.yDist, 'String'));
    nodeCount = str2double(get(handles.nodeCount, 'String'));
    if (get(handles.gridMode, 'Value'))
        nodeDistribMode = 'grid';
    else
        nodeDistribMode = 'rand';
    end
    PtLow = str2double(get(handles.TxPwrLow, 'String'));
    PtWLow = 10^(PtLow/10) / 1000;
    PrThresholdLow = str2double(get(handles.RxThreshLow, 'String'));
    PrThresholdWLow = 10^(PrThresholdLow/10) / 1000;
    PtHigh = str2double(get(handles.TxPwrHigh, 'String'));
    PtWHigh = 10^(PtHigh/10) / 1000;
    PrThresholdHigh = str2double(get(handles.RxThreshHigh, 'String'));
    PrThresholdWHigh = 10^(PrThresholdHigh/10) / 1000;
    pathLossCoeff = str2double(get(handles.pathLossCoeff, 'String'));
    maxDistanceLow = nthroot(PtWLow/PrThresholdWLow, pathLossCoeff);
    maxDistanceHigh = nthroot(PtWHigh/PrThresholdWHigh, pathLossCoeff);
    guarDistance = str2double(get(handles.guarDist, 'String'));
    pauseInt = str2double(get(handles.pauseInt, 'String'));
    moteClockSpd = str2double(get(handles.moteCPU, 'String'));
    radioFreq = str2double(get(handles.radioFreq, 'String'));

    TXCurDraw = str2double(get(handles.TXCurDraw, 'String'));
    RXCurDraw = str2double(get(handles.RXCurDraw, 'String'));
    RXtoTXratio = RXCurDraw / TXCurDraw;
    receiver_threshold = (3 + sqrt(9 + 8/RXtoTXratio)) / 2; % Calculate
point at which hybrid sync'ing switches from RBS to TPSN

    nodePowerMax = str2double(get(handles.nodePowerMax, 'String'));
    nodePowerWarn = str2double(get(handles.nodePowerWarn, 'String'));
    rediscoverLimit = str2double(get(handles.rediscoverLimit, 'String'));
    close;
end

% ------------------- SAMPLE FUNCTIONS -------------------
function Menu_mica2dot433_Callback(hObject, eventdata, handles)
    SetSamples(handles, 'mica2dot433', 'show');
    SetSamples(handles, 'initArrays', 'hide');
end

function Menu_mica2dot916_Callback(hObject, eventdata, handles)
    SetSamples(handles, 'mica2dot916', 'show');
    SetSamples(handles, 'initArrays', 'hide');
```

```
end

function Menu_micaz_Callback(hObject, eventdata, handles)
    SetSamples(handles, 'micaz', 'show');
    SetSamples(handles, 'initArrays', 'hide');
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Is Received                                  %
%            Coverage & Connectivity Control in WSNs                    %
%                      Kakoulli Elena                                   %
%                  Computer Science, UCY                                %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [r] = IsReceived(dist, packetType)
% Return 1 for successful TX or 0 for failed TX
    GlobalVars();
    r = 0;
    if (strcmp(packetType, 'topolDiscovery'))
        maxDist = maxDistanceLow;
    else
        maxDist = maxDistanceHigh;
    end
    gDist = min(guarDistance, maxDist); % Guaranteed TX distance

    if (dist <= gDist)
        r = 1;
    elseif (dist <= maxDist)
        prob = 1 - (dist - gDist) / (maxDist - gDist);
        if (rand() < prob)
            r = 1;
        end
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      Global Variables                                 %
%            Coverage & Connectivity Control in WSNs                    %
%                        Kakoulli Elena                                 %
%                  Computer Science, UCY                                %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global verDate; % Version number for the simulator
global propSpd; % Speed of light (in m/s)

% Grid Parameters
global xDistance; % Length of x-axis on grid
global yDistance; % Length of y-axis on grid
global nodeCount; % Number of nodes in the grid
```

```
global nodeDistribMode; % Type of grid distribution for nodes (grid or
random)

% Hardware Parameters
global moteClockSpd; % Speed of mote's CPU (in MHz)
global radioFreq; % Throughput of mote's transceiver (in Kbps)
global TXCurDraw; % Current draw for a transmission (in mA)
global RXCurDraw; % Current draw for a reception (in mA)
global receiver_threshold; % Number of receivers needed where HTC becomes
more efficient than APC
global nodePowerMax; % Max power for each node
global nodePowerWarn; % Power where node shows low-power warning
global nodePower; % [1 x nodeCount] array holding power for each node
global nodeBuffer;% [1 x nodeBuffer] array holding buffer for each node
global gridPower; % [1 x *] array holding the grid's total power (index is
number of energy-consuming events)
global deadNodes; % Number of total dead nodes in the network

% discover Parameters
global PtLow; % Transmission power (in dB) for discover packets
global PtWLow; % Transmission power (in W) for discover packets
global PrThresholdLow; % Threshold power (in dB) for discover packets
global PrThresholdWLow; % Threshold power (in W) for discover packets
global PtHigh; % Transmission power (in dB) for sync packets
global PtWHigh; % Transmission power (in W) for sync packets
global PrThresholdHigh; % Threshold power (in dB) for sync packets
global PrThresholdWHigh; % Threshold power (in W) for sync packets
global pathLossCoeff; % Path loss coefficient for transmission equation
global guarDistance; % Guaranteed distance for successful transmission
global maxDistanceLow; % Maximum distance nodes are capable of transmitting
a discover packet
global maxDistanceHigh; % Maximum distance nodes are capable of
transmitting a sync packet
global RXtoTXratio; % Ratio of reception power to transmission power
global pauseInt; % Length of time between plotting generations

% topolDiscovery Variables
global neighborTable; % nodeCount x nodeCount array holding x and y
coordinates for each node
global parent; % [1 x nodeCount] array showing parent node (i.e. parent(6)
returns the parent node of node 6)
global numChildren; % [1 x nodeCount] array showing a node's number of
children
global children; % [nodeCount x nodeCount] array showing each node's
children
global nodeLevel; % [1 x nodeCount] array showing the generation of a given
node (-1 means it is an orphan)
global maxGens; % Maximum number of generations possible in the
topolDiscovery
global sourceNode; % Root node for transmissions
global closestNode; % Closest node user clicked to
global orphanNodes; % Variable to keep track of the number of orphaned
nodes
global discoverTime; % Time used to topolDiscovery the network

% Coverage or Connectivity Variables
global COVandCONNAlg;   % Type of coverage or connectivity Algorithm being
used
```

```
global numTx;           % Number of transmissions used to perform
congestionAlgorithm
global numRx;           % Number of receptions used to perform
congestionAlgorithm
global dropTx;          % Number of transmissions used to perform
congestionAlgorithm
global rediscoverLimit; % Percentage of dead nodes at which network must
rediscover
global source_sinkTime; % Time used to send data from the source to the
sink
global DeadNodes;

% Grid Heads Variables & flags
global gridHeads;     % periexei gia ka8e node, diladi ka8e 8esi tou
antistoixei se ka8e node kai to periexomeno einai o gridHead pou anikei
global VacantGrid;    % periexei to adeio grid
global NoVacant;      % flag when the grid is no vacant
global FoundTrustedNode; %periexei ton node pou 8a kini8ei
global NumGridHeads;  % o ari8mos twn gridHeads
global NeighboringHeads;  % [NumGridHeads x NumGridHeads] array showing
each node's children
global Neighbors;     % o ka8e gridHead me ton ari8mo geitonikwn gridHeads
tou
global movesNodes;    % o ari8mos twn nodes pou kini8ikan
global moveDistances; % h synoliki apostasi kinisis twn nodes
global ProcessesInit; % ta processes pou ginontai initiated
global trustedNodes;  % pinakas ka8e 8esi tou opoiou einai o ari8mos twn
trusted nodes ka8e gridHead
global Heads;         % pinakas pou periexei olous tous grid heads
global trustOfHead;   % pinakas me tous trusted nodes
global NodesMove;     % Oi komboi pou kini8ikan kata tin diarkeia twn
algori8mwn

% FOR SR Algorithm
global successors;  %[NumGridHeads 2]
global precedings; %[NumGridHeads 2]


% Flags
global dataLoadedF; % =1 when the parameters are loaded from a .mat file
global nodesDispersedF; % =1 when the nodes have been plotted
global networkChangedF; % =1 when the xDist, yDist, or nodeCount are
changed and the plot has not been updated
global continuousF; % =1 when the "Continuous" checkbox is selected
global changeSourceNodeF; % =1 when the "Change Source" checkbox is
selected
global tracebackF; % =1 when the "Trace back to Root" checkbox is selected
global viewNodeIDF; % =1 when the "View Node IDs" menu item is selected
global networkLvlDiscF; % =1 when the network has done the level discovery
(discovering)
global networkSyncF; % =1 when the network has been completely synchronized
global congestedF; %=1 when the node is congested



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Draw Topology                                  %
```

```
%            Coverage & Connectivity Control in WSNs      %
%                       Kakoulli Elena                    %
%                   Computer Science, UCY                 %
%                                                         %
%                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function DrawTopology(delay)
% Draw the discovering lines
    GlobalVars;
         for j = 1 : nodeCount
            UpdateNodePower();
            x1 = neighborTable(gridHeads(j),1);
            y1 = neighborTable(gridHeads(j),2);
            x2 = neighborTable(j,1);
            y2 = neighborTable(j,2);
            line([x1 x2], [y1 y2], 'Color', 'g', 'LineWidth', 2);
         end

        if (delay ~= 0)
            pause(delay);
        end

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Coverage & Connectivity Algorithm SR        %
%            Coverage & Connectivity Control in WSNs      %
%                       Kakoulli Elena                    %
%                   Computer Science, UCY                 %
%                                                         %
%                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function COVandCONNAlgorithmSR(method, startNode)
% Function implements the SR algorithm for coverage and connectivity
control

    GlobalVars();
    VacantGrid = zeros(NumGridHeads);
    finished = 0;    %flag for end of algorithm
    gridHeads(Heads(3))=0; %for sample - this must be change later

    while(finished==0)
         GridHeadsCommunicSR();

        for i=1 : NumGridHeads
            if(VacantGrid(Heads(i))~=0)
                finished = 0;
                break;
            else
                finished = 1;
            end
        end

        if(finished==0)
            CascadingMovementSR();
```

```
         %gridHeads(3)=3;
        end

    end

    %% For GridHeads connection
    for i=1 : NumGridHeads

        if(VacantGrid(Heads(i))==0)
                    x1 = neighborTable(Heads(i),1);
                    y1 = neighborTable(Heads(i),2);
                    x2 = neighborTable(Heads(successors(i)),1)
                    y2 = neighborTable(Heads(successors(i)),2);
                    line([x1 x2], [y1 y2], 'Color', 'cyan', 'LineWidth',
2);
        end
    end
    UpdateNodePower();
end

%% STEP 1: Oloi oi gridHeads elegxoun ean yparxei connection meta3y tous
%% alliws yparxei vacant grid kai 8a 3ekinisei cascading movement apo ton
%% 1o gridHead pou to anixneyse kai 8a enimerwsei tous ypoloipous me ena
%% notification kai to STEP 1 8a ginete epanaliptika efoson den pairnei
%% kanenas kapoio notification msg

function GridHeadsCommunicSR()

    GlobalVars();
    NoVacant = -1;
    RandomHeads = randperm(NumGridHeads);

    for i=1 : NumGridHeads     % o ka8e gridHead elegxei

       if(gridHeads(Heads(RandomHeads(i)))~=0 &&
successors(RandomHeads(i))~=0)
        dist = CalcDistance(neighborTable(Heads(RandomHeads(i)),1),
neighborTable(Heads(RandomHeads(i)),2),
neighborTable(Heads(successors(RandomHeads(i))),1),
neighborTable(Heads(successors(RandomHeads(i))),2));

        %sensing, so lose power
        nodePower(RandomHeads(i)) = nodePower(RandomHeads(i)) - (dist/4);

            if(dist~=0)
                PrWLow = PtWLow / (dist ^ pathLossCoeff); % Calculate
reception power
            end
            %IsReceived(dist, 'topolDiscovery') == 0


            if(gridHeads(Heads(successors(RandomHeads(i))))==0 &&
trustedNodes(Heads(successors(RandomHeads(i)))) == 0) %elegxos gia vacant
grid

                VacantGrid(Heads(RandomHeads(i))) =
Heads(successors(RandomHeads(i)));
                ProcessesInit = ProcessesInit + 1;
```

```
            else
                VacantGrid(Heads(RandomHeads(i))) = 0;
            end
        end
    end
end

%% STEP 2: Oi gridHeads pou exo un anixneysei kapoio vacant grid 8a prepei
%% na psaxoun na broun kapoio trusted node gia na metakini8ei sto vacant
grid
%% kai na ginei aytos o gridHead tou vacant grid, alliws an den bre8ei na
steiloun
%% ena notification msg stous geitonikous gridHeads gia na arxikopoihsoun
%% kai oi idioi mia tetoia diadikasia kai na metakini8ei o idios sto vacant
grid
%% kai na ginei se ayto o gridHead

function CascadingMovementSR();
    GlobalVars();
    RandomHeads = randperm(NumGridHeads);

    for i=1 : NumGridHeads

        if(VacantGrid(Heads(RandomHeads(i)))>0)

            movesNodes = movesNodes + 1; %ay3anw tous kombous pou kini8ikan

            x1 = neighborTable(Heads(RandomHeads(i)),1);
            y1 = neighborTable(Heads(RandomHeads(i)),2);
            x2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),1);
            y2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),2);
            plot(neighborTable(VacantGrid(Heads(RandomHeads(i))),1),
neighborTable(VacantGrid(Heads(RandomHeads(i))),2), 'x',
'MarkerEdgeColor','r', 'LineWidth',10);
            drawLineStatus(x1, x2, y1, y2, 'NoConnection');

            if(trustedNodes(Heads(RandomHeads(i)))>0) %elegxos ean yparxei
trusted node mesa sto grid pou einai arxigos
                %briskw ton 1o apo ta trusted nodes tou
                FoundTrustedNode =
trustOfHead(Heads(RandomHeads(i)),trustedNodes(Heads(RandomHeads(i))));
            else
              FoundTrustedNode = Heads(RandomHeads(i));
              %enimerwsi tou preceding node tou
              NotificationSR(Heads(RandomHeads(i)), RandomHeads(i));
            end


          % ypologismos apostasis pou metakineitai o trusted node 'i o
gridHead
              dist = CalcDistance(neighborTable(FoundTrustedNode,1),
neighborTable(FoundTrustedNode,2),
neighborTable(VacantGrid(Heads(RandomHeads(i))),1),
neighborTable(VacantGrid(Heads(RandomHeads(i))),2));
              moveDistances = moveDistances + dist; %ay3anw tin synoliki
apostasis metakinisis pou egine
              NodesMove(movesNodes) = FoundTrustedNode;
```

```
              if(dist~=0)
                PrWLow = PtWLow / (dist ^ pathLossCoeff); % Calculate
reception power
              end

               % moving, so lose power
                nodePower(FoundTrustedNode) = nodePower(FoundTrustedNode) -
(dist/2);

          if(FoundTrustedNode == Heads(RandomHeads(i))) % ean den bre8ike
trusted node

              %move & change gridhead
              drawLineStatus(x1, x2, y1, y2, 'CascadingMovement');
              %neighborTable(VacantGrid(i),1) =
              %neighborTable(VacantGrid(i),2) =

gridHeads(VacantGrid(Heads(RandomHeads(i))))=VacantGrid(Heads(RandomHeads(i
)));
              gridHeads(FoundTrustedNode)=0;

          else   % tote bre8ike trusted node k prepei na metakini8ei sto
vacant grid k na ginei o gridHead tou
              x1 = neighborTable(FoundTrustedNode,1);
              y1 = neighborTable(FoundTrustedNode,2);
              x2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),1);
              y2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),2);
              drawLineStatus(x1, x2, y1, y2, 'CascadingMovement');

              %metakinisi tou trusted node sto vacant grid kai ginete o
              %gridHead tou vacant grid
              neighborTable(FoundTrustedNode,1) =
neighborTable(VacantGrid(Heads(RandomHeads(i))),1);
              neighborTable(FoundTrustedNode,1) =
neighborTable(VacantGrid(Heads(RandomHeads(i))),2);
              gridHeads(FoundTrustedNode)=0;
              gridHeads(VacantGrid(Heads(RandomHeads(i))))=
VacantGrid(Heads(RandomHeads(i)));
              trustedNodes(Heads(RandomHeads(i))) =
trustedNodes(Heads(RandomHeads(i))) - 1; %feygei o trusted node kai ara
meiwnete kai o ari8mos twn trusted nodes toy gridHead
              %gridHeads(VacantGrid(i))=FoundTrustedNode;
              %gridHeads(FoundTrustedNode)=0;
              %....enimerwsi pws einai o new gridHead tou vacant grid

          end

           plot(neighborTable(VacantGrid(Heads(RandomHeads(i))),1),
neighborTable(VacantGrid(Heads(RandomHeads(i))),2), '^',
'MarkerEdgeColor','b', 'LineWidth',2);

           VacantGrid(Heads(RandomHeads(i))) = 0;
        end
    end
end


function NotificationSR(gridHead, position)
```

```
% Send notification msg from the gridHead to its precending gridHead
% gridHead to inform it for his cascading movement to the vacant grid

    GlobalVars();

        if (Heads(precedings(position))~=VacantGrid(gridHead) &&
nodePower(gridHead) >= 1 && nodePower(Heads(precedings(position))) > 0)

            x1 = neighborTable(gridHead,1);
            y1 = neighborTable(gridHead,2);
            x2 = neighborTable(Heads(precedings(position)),1);
            y2 = neighborTable(Heads(precedings(position)),2);

            drawLineStatus(x1, x2, y1, y2, 'MsgNotification');
            numTx = numTx + 1;
            nodePower(gridHead) = nodePower(gridHead) - 1;
            numRx = numRx + 1;
            nodePower(Heads(precedings(position))) =
nodePower(Heads(precedings(position))) - RXtoTXratio;

        end

end


function drawLineStatus(startX, endX, startY, endY, type)
    GlobalVars();
    switch (type)
        case 'MsgNotification'
            lineColor = 'b';
            style = '-';
        case 'CascadingMovement'
            lineColor = 'red';
            style = '--';
        case 'NoConnection'
            lineColor = 'm';
            style = '--';
        otherwise
    end
    line([startX endX], [startY endY], 'Color',lineColor, 'LineWidth',2,
'LineStyle',style);
    pause(0.8);
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Coverage & Connectivity Algorithm AM       %
%            Coverage & Connectivity Control in WSNs      %
%                       Kakoulli Elena                    %
%                    Computer Science, UCY                %
%                                                         %
%                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Control scheme under active model:
% 1. For any grid head u that detects a vacant neighboring
```

```
% grid, if no notification is received in the previous round
% from that area (to avoid overreaction), a replacement
% process is initiated after the corresponding region area is determined.
% 2. For any grid head u that has started the replacement
% process or that is notified in the cascading replacement
% process, find one of neighboring spare trusted nodes
% in its grid, say node v, to move into that neighboring
% vacant grid before the next round starts.
% 3. If such a node v cannot be found, select any neighboring
% grid that is other than the vacant one but still in
% region. Then, send out the notification for the replacement
% of u, attaching the information of region and
% such a selection. It will always select the one with
% spare trusted node(s) first. After that, move u to the
% vacant grid before the next round starts.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function COVandCONNAlgorithmAM()
% Implements coverage and connectivity control algorithm Active Model

    GlobalVars();
    VacantGrid = zeros(NumGridHeads);
    finished = 0;    %flag for end of algorithm
gridHeads(Heads(2))=0; % for sample - this must be change later

    while(finished==0)
         GridHeadsCommunicAM();

        for i=1 : NumGridHeads
            if(VacantGrid(Heads(i))~=0)
                finished = 0;
                break;
            else
                finished = 1;
            end
        end

        if(finished==0)
            CascadingMovementAM();
        end

    end

    %% For GridHeads connection
    for i=1 : NumGridHeads
        if(VacantGrid(Heads(i))==0)
            for j = 1 : Neighbors(Heads(i))
                    x1 = neighborTable(Heads(i),1);
                    y1 = neighborTable(Heads(i),2);
                    x2 = neighborTable(NeighboringHeads(Heads(i),j),1);
                    y2 = neighborTable(NeighboringHeads(Heads(i),j),2);
                    line([x1 x2], [y1 y2], 'Color', 'cyan', 'LineWidth',
2);
            end
        end
    end

      UpdateNodePower();
```

```
end

%% STEP 1: Oloi oi gridHeads elegxoun ean yparxei connection meta3y tous
%% alliws yparxei vacant grid kai 8a 3ekinisei cascading movement apo ton
%% 1o gridHead pou to anixneyse kai 8a enimerwsei tous ypoloipous me ena
%% notification kai to STEP 1 8a ginete epanaliptika efoson den pairnei
%% kanenas kapoio notification msg

function GridHeadsCommunicAM()

    GlobalVars();
    RandomHeads = randperm(NumGridHeads);

    for i=1 : NumGridHeads     % o ka8e gridHead elegxei

       if(gridHeads(Heads(RandomHeads(i)))~=0 &&
VacantGrid(Heads(RandomHeads(i)))==0)
        for j=1 : Neighbors(Heads(RandomHeads(i))) % ean yparxei connection
me ton ka8e geitoniko tou gridHead

             dist = CalcDistance(neighborTable(Heads(RandomHeads(i)),1),
neighborTable(Heads(RandomHeads(i)),2),
neighborTable(NeighboringHeads(Heads(RandomHeads(i)),j),1),
neighborTable(NeighboringHeads(Heads(RandomHeads(i)),j),2));

        %sensing, so lose power
        nodePower(RandomHeads(i)) = nodePower(RandomHeads(i)) - (dist/4);


if(VacantGrid(NeighboringHeads(Heads(RandomHeads(i)),j))~=Heads(RandomHeads
(i)))


            if(dist~=0)
                PrWLow = PtWLow / (dist ^ pathLossCoeff); % Calculate
reception power
            end
            %IsReceived(dist, 'topolDiscovery') == 0
         end

            if(gridHeads(NeighboringHeads(Heads(RandomHeads(i)),j))==0 &&
trustedNodes(NeighboringHeads(Heads(RandomHeads(i)),j))==0) %elegxos gia
vacant grid

                VacantGrid(Heads(RandomHeads(i))) =
NeighboringHeads(Heads(RandomHeads(i)),j);
                ProcessesInit = ProcessesInit + 1;
                break;
            else
                VacantGrid(Heads(RandomHeads(i))) = 0;
            end

        end
       end
    end
end

%% STEP 2: Oi gridHeads pou exo un anixneysei kapoio vacant grid 8a prepei
```

```
%% na psaxoun na broun kapoio trusted node gia na metakini8ei sto vacant
grid
%% kai na ginei aytos o gridHead tou vacant grid, alliws an den bre8ei na
steiloun
%% ena notification msg se ena geitona gridHead kai na metakini8ei o idios
sto vacant grid
%% kai na ginei se ayto o gridHead

function CascadingMovementAM();
    GlobalVars();
    RandomHeads = randperm(NumGridHeads);


    for i=1 : NumGridHeads

        if(VacantGrid(Heads(RandomHeads(i)))>0)

            movesNodes = movesNodes + 1; %ay3anw tous kombous pou kini8ikan

            x1 = neighborTable(Heads(RandomHeads(i)),1);
            y1 = neighborTable(Heads(RandomHeads(i)),2);
            x2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),1);
            y2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),2);
            plot(neighborTable(VacantGrid(Heads(RandomHeads(i))),1),
neighborTable(VacantGrid(Heads(RandomHeads(i))),2), 'x',
'MarkerEdgeColor','r', 'LineWidth',10);
            drawLineStatus(x1, x2, y1, y2, 'NoConnection');


          if(trustedNodes(Heads(RandomHeads(i)))>0) %elegxos ean yparxei
trusted node mesa sto grid pou einai arxigos
                %briskw ton 1o apo ta trusted nodes tou
                FoundTrustedNode =
trustOfHead(Heads(RandomHeads(i)),trustedNodes(Heads(RandomHeads(i))));
          else
              FoundTrustedNode = Heads(RandomHeads(i));
              %enimerwsi enos apo tous geitonikoys gridHeads me ta
              %perissotera trusted nodes
              NotificationAM(Heads(RandomHeads(i)));
          end


           % ypologismos apostasis pou metakineitai o trusted node 'i o
gridHead
              dist = CalcDistance(neighborTable(FoundTrustedNode,1),
neighborTable(FoundTrustedNode,2),
neighborTable(VacantGrid(Heads(RandomHeads(i))),1),
neighborTable(VacantGrid(Heads(RandomHeads(i))),2));
              moveDistances = moveDistances + dist; %ay3anw tin synoliki
apostasis metakinisis pou egine
              NodesMove(movesNodes) = FoundTrustedNode;

              if(dist~=0)
                PrWLow = PtWLow / (dist ^ pathLossCoeff); % Calculate
reception power
              end
```

```
              % moving, so lose power
              nodePower(FoundTrustedNode) = nodePower(FoundTrustedNode) -
(dist/2);

          if(FoundTrustedNode == Heads(RandomHeads(i))) % ean den bre8ike
trusted node



           %for w=1 :4
            %  if(RandomHeads(w)~=i)
             %   if(VacantGrid(RandomHeads(w))==0)
              %   VacantGrid(RandomHeads(w))=i;
               %  CascadingMovementAM();
                % break;
                %end
              %end
           %end

              %move & change gridhead
              drawLineStatus(x1, x2, y1, y2, 'CascadingMovement');
              %neighborTable(VacantGrid(i),1) =
              %neighborTable(VacantGrid(i),2) =

gridHeads(VacantGrid(Heads(RandomHeads(i))))=VacantGrid(Heads(RandomHeads(i
)));
              gridHeads(FoundTrustedNode)=0;

          else   % tote bre8ike trusted node k prepei na metakini8ei sto
vacant grid k na ginei o gridHead tou
              x1 = neighborTable(FoundTrustedNode,1);
              y1 = neighborTable(FoundTrustedNode,2);
              x2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),1);
              y2 = neighborTable(VacantGrid(Heads(RandomHeads(i))),2);
              drawLineStatus(x1, x2, y1, y2, 'CascadingMovement');

              %metakinisi tou trusted node sto vacant grid kai ginete o
              %gridHead tou vacant grid
              neighborTable(FoundTrustedNode,1) =
neighborTable(VacantGrid(Heads(RandomHeads(i))),1);
              neighborTable(FoundTrustedNode,1) =
neighborTable(VacantGrid(Heads(RandomHeads(i))),2);
              gridHeads(FoundTrustedNode)=0;
              gridHeads(VacantGrid(Heads(RandomHeads(i))))=
VacantGrid(Heads(RandomHeads(i)));
              trustedNodes(Heads(RandomHeads(i))) =
trustedNodes(Heads(RandomHeads(i))) - 1; %feygei o trusted node kai ara
meiwnete kai o ari8mos twn trusted nodes toy gridHead
              %gridHeads(VacantGrid(i))=FoundTrustedNode;
              %gridHeads(FoundTrustedNode)=0;
              %....enimerwsi pws einai o new gridHead tou vacant grid


              %plotGrid();

          end
```

```
            plot(neighborTable(VacantGrid(Heads(RandomHeads(i))),1),
neighborTable(VacantGrid(Heads(RandomHeads(i))),2), '^',
'MarkerEdgeColor','b', 'LineWidth',2);
            VacantGrid(Heads(RandomHeads(i))) = 0;
        end

    end
end


function NotificationAM(gridHead)
% Send notification msg from the gridHead to one of its neighbor's
% gridHeads, that with more trusted nodes, to inform it for his cascading
movement to the vacant
% gridHead

    GlobalVars();
    tnodes = 0;
    headChoose = 0;

    for h = 1 : Neighbors(gridHead)

        if(NeighboringHeads(gridHead,h)~=VacantGrid(gridHead) &&
nodePower(gridHead) >= 1 && nodePower(NeighboringHeads(gridHead,h)) > 0 &&
tnodes<trustedNodes(NeighboringHeads(gridHead,h)))
            tnodes = trustedNodes(NeighboringHeads(gridHead,h));
            headChoose = NeighboringHeads(gridHead,h);
        end

    end

    if(headChoose~=0)
    x1 = neighborTable(gridHead,1);
    y1 = neighborTable(gridHead,2);
    x2 = neighborTable(headChoose,1);
    y2 = neighborTable(headChoose,2);

    drawLineStatus(x1, x2, y1, y2, 'MsgNotification');
    numTx = numTx + 1;
    nodePower(gridHead) = nodePower(gridHead) - 1;
    numRx = numRx + 1;
    nodePower(headChoose) = nodePower(headChoose) - RXtoTXratio;
    VacantGrid(headChoose) = gridHead;
    ProcessesInit = ProcessesInit + 1;

    end
end


function drawLineStatus(startX, endX, startY, endY, type)
    GlobalVars();
    switch (type)
        case 'MsgNotification'
            lineColor = 'b';
            style = '-';
        case 'CascadingMovement'
            lineColor = 'red';
            style = '--';
```

```
        case 'NoConnection'
            lineColor = 'm';
            style = '--';
        otherwise
    end
    line([startX endX], [startY endY], 'Color',lineColor, 'LineWidth',2,
'LineStyle',style);
    pause(0.8);
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Clear Handles                                  %
%             Coverage & Connectivity Control in WSNs                   %
%                          Kakoulli Elena                               %
%                    Computer Science, UCY                              %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ClearHandles(handles)
% Auxiliary function which clear the handles of the main function
    GlobalVars();
    set(handles.analyzedNode, 'String','');
    set(handles.GridHead, 'String','');
    set(handles.NodesMove, 'String','');
    set(handles.pwrRem, 'String','');
    set(handles.discoverTime, 'String','');
    set(handles.source_sinkTime, 'String','');
    set(handles.numRX, 'String','');
    set(handles.droppedTX, 'String','');
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Calculate Distance                                 %
%             Coverage & Connectivity Control in WSNs                   %
%                          Kakoulli Elena                               %
%                     Computer Science, UCY                             %
%                                                                       %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [d] = dist(x1, y1, x2, y2)
% Function for calculating the distance between two nodes
    d = sqrt(double((x1-x2)^2 + (y1-y2)^2));
end
```