ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μηχανισμός Ανακάλυψης Πληροφορίας Πλαισίου με χρήση Αλγορίθμων Διάχυσης Πληροφορίας

Μιχαήλ Σ. Νικητίδης

Επιβλέπων: Ευστάθιος Χατζηευθυμιάδης

ΠΑΤΡΑ
ΑΠΡΙΛΙΟΣ, 2009

Διπλωματική Εργασία

Μηχανισμός Ανακάλυψης Πληροφορίας Πλαισίου με χρήση Αλγορίθμων Διάχυσης Πληροφορίας

Μιχαήλ Σ. Νικητίδης

Απρίλιος 2009

Μηχανισμός Ανακάλυψης Πληροφορίας Πλαισίου με χρήση Αλγορίθμων Διάχυσης Πληροφορίας

Μιχαήλ Σ Νικητίδης

Ευστάθιος Χατζηευθυμιάδης	**Αχιλλέας Καμέας**	**Βασίλειος Βίτσας**
Επιβλέπων	Μέλος 1	Μέλος 2
Επίκουρος Καθηγητής	Επίκουρος Καθηγητής	Αναπληρωτής Καθηγητής
Ε.Κ.Π.Α	Ε.Α.Π.	Α.Τ.Ε.Ι.Θ.

Περίληψη

Η έννοια επίγνωση πλαισίου (context-awareness) αναφέρεται στην ικανότητα υπηρεσιών κινητού υπολογισμού (mobile context-aware applications) να μετρούν, να ανακτούν και να συμπεραίνουν πληροφορία πλαισίου (πολυδιάστατα δεδομένα) από το περιβάλλον που ενεργεί ο χρήστης και να προσαρμόζονται σε αυτό. Ένα βασικό θέμα που σχετίζεται με την λειτουργικότητα των υπηρεσιών αυτών είναι η ανακάλυψη πληροφορίας πλαισίου (contextual information - context) σε ad-hoc ασύρματα δίκτυα κινητών κόμβων. Κύριος στόχος των κινητών κόμβων που αναφέρονται ως «οπαδοί» είναι η μετακίνησή τους σε χώρους όπου η ποιότητα υπηρεσίας μεγιστοποιείται.

Στόχος αυτής της Διπλωματικής Εργασίας είναι η εύρεση κάποιων μηχανισμών/αλγορίθμων που θα βοηθούν στην ανακάλυψη της πληροφορίας πλαισίου. Ένας τέτοιος μηχανισμός που υιοθετήθηκε προς αυτήν την κατεύθυνση και επεκτάθηκε η λειτουργικότητά του στο πεδίο της χρονικής εγκυρότητας της εκτίμησης του πλαισίου, χαρακτηρίζεται από τη «συνεργατική» συμπεριφορά των οντοτήτων (agents ή particles) που συμμετέχουν προκειμένου να επιτευχθεί ο στόχος, και είναι ο particle swarm optimization (PSO). Ουσιαστικά πρόκειται για μια βιο-μιμητική προσέγγιση του μηχανισμού όπου κοπάδια έμβιων όντων (πουλιά, ψάρια, μυρμήγκια) ανακαλύπτουν την τροφή τους.

Σύμφωνα με την θεώρηση αυτή, οι οπαδοί και οι πηγές αποτελούν κινούμενους «οργανισμούς» (particle) και κινούμενη τροφή, αντίστοιχα, σε ένα «σμήνος» (swarm) M+N οργανισμών. Η πλοήγηση των οπαδών μέσα στο πεδίο της εκάστοτε εφαρμογής κινητού υπολογισμού προκειμένου να ανακτήσουν επικαιροποιημένα πλαίσια, βασίζεται στην προσωπική εμπειρία κάθε κόμβου-οπαδού (υποκειμενικός παράγοντας) αλλά και στη γνώση που αυτός αποκομίζει από άλλα μέλη του σμήνους (αντικειμενικός παράγοντας).

Με τη βοήθεια της πλατφόρμας προσομοίωσης J-Sim έχει αναπτυχθεί ένα 2D μοντέλο που ουσιαστικά υλοποιεί μια «συνεργατική συμπεριφορά» των κόμβων-οπαδών μετατρέποντας τους σε «co-operative hunters» με τη βοήθεια του PSO, έτσι ώστε να ανακτήσουν όσο το δυνατόν πιο επίκαιρη πληροφορία από τις πηγές που

την κατέχουν. Εξετάστηκε η συμπεριφορά του μοντέλου για συντηρητική πολιτική ανακάλυψης, για δύο διαφορετικές παράμετρους ενός τέτοιου σύστηματος κινητού υπολογισμού (π.χ. αυτο-οργανώμενες ρομπομποτικές συσκευές, επισκέπτες ενός μουσείου εφοδιασμένοι με PDA's, σμήνη UUV κλπ), όπως ο βαθμός κινητικότητας των πηγών και το μέγεθος γειτονίας (ακτίνα δράσης) των κόμβων-οπαδών και βρέθηκαν κάποιες βέλτιστες τιμές βαθμού κινητικότητας και ακτίνας δράσης. Τέλος έγινε συγκριτική ανάλυση της συμπεριφοράς του μοντέλου για τυχαία κίνηση των κόμβων-οπαδών σε σχέση με την περίπτωση που αυτοί πλοηγούνται βάσει του PSO.

Λέξεις-κλειδιά: Επίγνωση Πλαισίου, Ασύρματα Δίκτυα Αισθητήρων, Ανακάλυψη Πληροφορίας Πλαισίου, Αλγόριθμος PSO, Κινητός Υπολογισμός

Περιεχόμενο:Κείμενο, πρόγραμμα σε γλώσσα Java, πρόγραμμα σε γλώσσα Tcl, σχήματα

Contextual Information Discovery Mechanism using Information Diffusion Algorithms

Michail S Nikitidis

Efstathios Hadjiefthymiades	**Achilleas Kameas**	**Vassilios Vitsas**
Supervisor	Member 1	Member 2
Assistant Professor	Assistant Professor	Associate Professor
N.K.U.A.	H.O.U.	A.T.E.I.Th.

Abstract

The concept of Context-Awareness refers to the ability of mobile context-aware applications to measure, regress and infer contextual information (mutli-dimensional data) from user environment and to adapt to it. An essential issue related to the functionality of mobile context-aware applications is the discovery of contextual information in ad-hoc wireless networks of mobile nodes. The aim of such mobile nodes refer to as ”followers” is to move to areas where the quality of service is maximised.

The aim of this thesis is the establishment of mechanisms/algorithms to utilize the discovery of contextul information. A mechanism that has been adopted towards this direction and its functionality has been extended to the field of temporal validity of context estimation, is characterised by the ”co-operative” behaviour of the participating entities (agents or particles) to achieve the goal and is known as Particle Swarm Optimisation (PSO). It is actually a bio-inspired approach of the mechanism where bird flocking, fish schooling or social insects strive to discover their food.

According to these assumptions, ”followers” and ”sources” consist mobile ”organisms” (particles) and food respectively, within a swarm of M+N ”organisms”. The navigation of the ”followers” within the field where the context-aware mobile application is executed, in order to regress valid contexual information, is based on the subjective experience of each ”follower” (cognitive factor) and the knowledge that each ”follower” retrieves from other swarm members (social factor).

A 2D model has been developed based on the simulation platform J-sim, which actually builts a ”co-operative” behaviour for the ”follower” nodes by trasnforming them into ”co-operative hunters” using PSO, utilizing them to regress valid contextual information from sources.

Model behaviour has been investigated for a conservative foraging policy, regarding two parameters of a context-aware system (e.g. self-oranised robots, museum visitors equipped with PDA’s, UUV swarms etc) like degree of source mobility and neighbourhood size (sensing radius). Optimal values have been calculated. Finally a comparative analysis was carried out regarding model behaviour for random navigation of ”follower” nodes with regards to the case where PSO was used.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα Στάθη Χατζηευθυμιάδη που μου εμπιστεύθηκε την εκπόνηση της παρούσας Διπλωματικής αλλά και για τις χρήσιμες οδηγίες του κατά τη διάρκεια της εκπόνησης.
Επίσης θα ήθελα να ευχαριστήσω και την οικογένεια μου για την υπομονή της κατά τη διάρκεια των σπουδών μου και στην οποία αφιερώνεται αυτή η εργασία.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

1 Εισαγωγή

Τα δίκτυα κινητών ασύρματων κόμβων χωρίς συγκεντρωτική δομή (mobile ad-hoc networks) βρίσκουν αρκετές εφαρμογές όπως σε στρατιωτικές επιχειρήσεις, συστήματα κατανεμημένου υπολογισμού, δίκτυα ασύρματων αισθητήρων για on - line monitoring, κ.λ.π.

Για να μπορέσουν αυτά τα συστήματα να λειτουργούν όσο το δυνατόν πιο αυτόνομα χωρίς την συμβολή και τη διαμεσολάβηση των χρηστών και να λαμβάνουν από μόνα τους αποφάσεις έτσι ώστε να μπορούν να θεωρηθούν εύχρηστα, χρήσιμα και με ανθρωποκεντρικό χαρακτήρα εργαλεία θα πρέπει να έχουν αυτό που ονομάζεται «επίγνωση πληροφορίας πλαισίου».

Ο όρος «επίγνωση πληροφορίας πλαισίου» αναφέρεται στην ικανότητα ενός συστήματος να ανακαλύπτει, να ερμηνεύει, να αξιοποιεί και να συλλογίζεται βάσει της πληροφορίας για την περιρρέουσα κατάσταση γύρω από μια οντότητα (χρήστης, εφαρμογή) με τελικό σκοπό να λάβει αποφάσεις και να προσαρμοστεί όσο το δυνατόν γρηγορότερα σε διάφορες καταστάσεις [1]. Επομένως το πρώτο στάδιο για να επιτευχθεί από ένα σύστημα η «επίγνωση της πληροφορίας πλαισίου» είναι η γρήγορη ανακάλυψή του μέσα στο περιβάλλον στο οποίο κινείται. Η ανάγκη για γρήγορη ανακάλυψη της πληροφορίας πλαισίου συμβαδίζει τόσο με την περιορισμένη ενεργειακή αυτονομία τέτοιων συστημάτων όσο και με την απαίτηση της άμεσης προσαρμογής στο μεταβαλλόμενο περιβάλλον.

Στόχος επομένως αυτής της Διπλωματικής Εργασίας είναι η εύρεση κάποιων μηχανισμών/αλγορίθμων που θα βοηθούν στην ανακάλυψη της πληροφορίας πλαισίου. Ένας τέτοιος μηχανισμός που θα ελεγχθεί προς αυτήν την κατεύθυνση χαρακτηρίζεται από τη «συνεργατική» συμπεριφορά των οντοτήτων (agents ή particles) που συμμετέχουν προκειμένου να επιτευχθεί ο στόχος. Ουσιαστικά πρόκειται για μια βιο-μιμητική προσέγγιση του μηχανισμού όπου κοπάδια έμβιων όντων (πουλιά, ψάρια, μυρμήγκια) ανακαλύπτουν την τροφή τους. Ο αλγόριθμος βελτιστοποίησης που έχει αναπτυχθεί για να αντιμετωπίζονται τέτοιου είδους προβλήματα είναι ο particle swarm optimization (PSO).

Η έννοια του «πλαισίου» αναφέρεται συνήθως στη θέση που βρίσκεται μια οντότητα, οι γειτονικές προς αυτήν οντότητες όπως επίσης και οι θέσεις των οντοτήτων που κατέχουν την πληροφορία. Όμως η θέση είναι μια μόνο από τις συνιστώσες του «πλαισίου» το οποίο χαρακτηρίζει την περιρρέουσα κατάσταση, καθώς μπορεί κανείς να αναφέρει το μέτρο της ακρίβειας της πληροφορίας, τη χρονική εγκυρότητά της (πρόσφατη ή απαρχαιωμένη) κοκ. Για τους σκοπούς αυτής της εργασίας επιλέχθηκε η αναπαράσταση της «πληροφορίας πλαισίου» με μια συνιστώσα, αυτήν της χρονικής εγκυρότητας.

Η διάρθρωση της Διπλωματικής Εργασίας είναι η ακόλουθη:

Στη δεύτερη ενότητα γίνεται λόγος για τις έννοιες ”πληροφορία πλαισίου”, ”επίγνωση πληροφορίας πλαισίου”, και παρατίθεται αναλυτικά η αρχιτεκτονική ενός

συστήματος επίγνωσης πληροφορίας πλαισίου καθώς επίσης και μέθοδοι μοντελοποίησης της πληροφορίας πλαισίου.

Οι πιο δημοφιλείς αλγόριθμοι Νοημοσύνης Σμήνους (Swarm Intelligence - SI), ο Ant Colony Optimization και ο Particle Swarm Optimization, περιγράφονται στην τρίτη ενότητα μαζί με τις εφαρμογές αυτών των αλγορίθμων σε περιβάλλοντα κινητού υπολογισμού. Γίνεται επίσης συζήτηση και για το Βιολογικό Ανάλογο αυτών των αλγορίθμων, δίνοντας στοιχεία για τα «κοινωνικά έμβια όντα» (μυρμήγκια, πουλιά, ψάρια) από την συμπεριφορά των οποίων ξεπήδησαν οι αλγόριθμοι SI.

Στην τέταρτη ενότητα δίνεται αναλυτική περιγραφή του μοντέλου που κατασκευάσθηκε αλλά και της πολιτικής ανακάλυψης πλαισίου. Τέλος στην πέμπτη ενότητα παρατίθενται τα σενάρια που εκτελέστηκαν σε συγκεκριμένης δομής περιβάλλοντα κινητού υπολογισμού μαζί με τα αποτελέσματα και στην έκτη ενότητα τα συμπεράσματα και οι προτάσεις για περαιτέρω έρευνα. Στα Παρατήματα Α και Β παρατίθενται λίστες του κώδικα που αναπτύχθηκε ή τροποποιήθηκε.

2 Συστήματα Επίγνωσης Πληροφορίας Πλαισίου

2.1 Πληροφορία Πλαισίου

Η έννοια του Πλαισίου (context) έχει χρησιμοποιηθεί κατά κόρον σε διάφορες επιστημονικές περιοχές, όπως στη γλωσσολογία, φιλοσοφία, τεχνητή νοημοσύνη, επικοινωνίες κ.α. Στη Βιβλιογραφία υπάρχουν διάφορες ερμηνείες για τον όρο Πλαίσιο.

Το λεξικό Meriam-Webster [1] ορίζει ως πλαίσιο τις «αλληλοσχετιζόμενες συνθήκες μέσα στις οποίες υπάρχει ή συμβαίνει κάτι». Σύμφωνα με τον Dey[2] «πλαίσιο αποτελεί οποιαδήποτε πληροφορία μπορεί να χρησιμοποιηθεί για να χαρακτηρίσει μια οντότητα ή μια κατάσταση. Η οντότητα μπορεί να είναι κάποιο αντικείμενο, πρόσωπο ή τοποθεσία που σχετίζεται με την αλληλεπίδραση ενός χρήστη και μιας εφαρμογής, συμπεριλαμβανομένων του χρήστη και της εφαρμογής». Κατά τους Schilit και Theimer[3]το πλαίσιο αναφέρεται «ως η θέση και η ταυτότητα γειτονικών ως προς μια οντότητα αντικειμένων και προσώπων, καθώς επίσης και οι αλλαγές που συμβαίνουν σε αυτά τα αντικείμενα». Οι Brown et al[4]ερμηνεύει το πλαίσιο ως «τα στοιχεία του περιβάλλοντος ενός χρήστη για τα οποία ο υπολογιστής του χρήστη θα πρέπει να γνωρίζει». Τέλος κατά τους Schmidt et al[5] το πλαίσιο ορίζεται ως «η γνώση σχετικά με την κατάσταση ενός χρήστη και της συσκευής του, συμπεριλαμβανόμενων του περίγυρου και των συνθηκών στις οποίες βρίσκονται».

Κοινό χαρακτηριστικό των παραπάνω ορισμών αποτελεί ο περίγυρος μιας οντότητας και επομένως το πλαίσιο είναι ουσιαστικά ό,τι μας περιβάλλει και ο όρος αυτός χρησιμοποιείται κυρίως σε σχέση με το φυσικό κόσμο που περιβάλλει μια κινητή συσκευή, μια εφαρμογή ή ένα ολόκληρο σύστημα. Η έννοια του πλαισίου αρχικά βασιζόταν σε τρία θέματα: που βρίσκεται ο χρήστης, ποιοι είναι οι γύρω του χρήστες και ποίες είναι οι γειτονικές πηγές πληροφορίας του και αντικείμενα. Αν και η πληροφορία θέσης είναι βασικό συστατικό για την περιγραφή της περιρρέουσας κατάστασης του χρήστη, δεν μπορεί όμως να εντοπίσει τυχόν αλλαγές του περιβάλλοντος του χρήστη ή δυναμικές ανακατατάξεις των γειτονικών αντικειμένων του χρήστη. Έτσι η έννοια του πλαισίου επεκτείνεται σε μια πιο γενική θεώρηση της πληροφορίας που μπορεί να περιγράψει την κατάσταση του χρήστη συμπεριλαμβανομένου εκτός της πληροφορίας θέσης και πληροφορία όπως, για παράδειγμα, επίπεδο φωτεινότητας, επίπεδο θορύβου, διαθεσιμότητα πρόσβασης σε δίκτυο και κοινωνικά γεγονότα.

Στην επιστήμη της Πληροφορικής διεξάγεται τεράστια έρευνα γύρω από την έννοια της «πληροφορίας πλαισίου» (contextual information) και το κατά πόσο μπορεί να αξιοποιηθεί στην ανάπτυξη «εφαρμογών κινητού υπολογισμού» (mobile application) και στα δίκτυα του άμεσου μέλλοντος γενικότερα. Συγκεκριμένα, στην Πληροφορική, η έννοια πληροφορία πλαισίου είναι ισοδύναμη με την έννοια πλαίσιο, εφόσον παρουσιάζει ενδιαφέρον η πληροφορία που περιγράφει την περιρρέουσα κατάσταση. Παράλληλα εμφανίζεται και η ανάγκη για τον κατάλληλο προσδιορισμό του πλαισί-

ου σε συσχέτιση με τις ανάγκες κάθε εφαρμογής.

Λόγω της σύνθετης υφής της έννοιας «πληροφορία πλαισίου» και της ανομοιογενούς χρήσης της σε διάφορα επιστημονικά πεδία, είναι αναγκαίο να κατηγοριοποιηθούν τα διάφορα χαρακτηριστικά της, έτσι ώστε να να γίνουν ευδιάκριτα εκείνα που μπορούν να χρησιμοποιηθούν στο πεδίο της Πληροφορικής και των Επικοινωνιών. Μια τέτοια κατηγοριοποίηση ακολουθεί παρακάτω.

- **Τύπος Πλαισίου**
  - ⋆ Χαρακτηριστικά πληροφορίας πλαισίου για ανθρώπινο χρήστη: Περίγυρος χρήστη (θέση, ταυτότητα, κινητικότητα, διαθέσιμες συσκευές κλπ), Φυσική οντότητα χρήστη(ταυτότητα, συνήθειες, ιστορικό κλπ)
  - ⋆ Χαρακτηριστικά πληροφορίας πλαισίου για συσκευή: Διεύθυνση IP, IP Mask
  - ⋆ Χαρακτηριστικά πληροφορίας πλαισίου για δίκτυο: Ταυτότητα δικτύου, ευρυζωνικότητα, ποιότητα υπηρεσιών, επίπεδο ασφάλειας, κάλυψη κλπ
- **Εμμονή**
  - ⋆ Σταθερό (δεν απαιτείται ενημέρωση): Πλαίσιο το οποίο δεν εξελίσσεται με το χρόνο, παραμένοντας σταθερό σε όλη τη διάρκεια της ύπαρξής του(π.χ. όνομα, Αρ. ταυτότητας κλπ)
  - ⋆ Προσωρινό (απαιτείται ενημέρωση): μέρος της πληροφορίας πλαισίου μεταβάλλεται με το χρόνο (θέση, φορτίο δεδομένων σε δρομολογητές)
- **Εξέλιξη: για προσωρινό πλαίσιο**
  - ⋆ Στατικό: Πλαίσιο το οποίο δεν εξελίσσεται με το χρόνο πολύ γρήγορα (θερμοκρασία περιβάλλοντος)
  - ⋆ Δυναμικό: Πλαίσιο το οποίο εξελίσσεται με το χρόνο πολύ γρήγορα (θέση οδηγού αυτοκινήτου)
- **Μετρησιμότητα**
  - ⋆ Φυσικό (μετρήσιμο): Πλαίσιο το οποίο δεν είναι απροσδιόριστο και μπορεί να μετρηθεί (π.χ. με αισθητήρες) (γεωγραφική θέση, υγρασία, θερμοκρασία, πόροι δικτύου)
  - ⋆ Απροσδιόριστο (μη μετρήσιμο με τη βοήθεια φυσικών μεγεθών): Παραδείγματα τέτοιων πλαισίων (όνομα, συνήθειες, hobby κλπ)
- **Συνδεσιμότητα με υπηρεσία ή εφαρμογή**
  - ⋆ Αναγκαιότητα: Μέρος της πληροφορίας πλαισίου είναι απαραίτητο να ανακτηθεί από κάποια υπηρεσία ή εφαρμογή για να λειτουργήσει σωστά.
  - ⋆ Μη αναγκαίο: Επιπρόσθετη πληροφορία πλαισίου που δεν είναι απαραίτητη για τη λειτουργία κάποιας υπηρεσίας αλλά μπορεί να αποβεί χρήσιμη.

- **Χρονικές συνθήκες**
  - ⋆ Παρελθόν: Αυτή η κατηγορία αναφέρεται σε πληροφορία πλαισίου από το παρελθόν, δηλαδή σε ένα ιστορικό πληροφορίας πλαισίου για κάποιο χρονικό διάστημα του πρόσφατου ή απώτερου παρελθόντος.
  - ⋆ Παρόν: Αυτή η κατηγορία αναφέρεται στην τρέχουσα πληροφορία πλαισίου (π.χ. θέση στην οποία βρίσκομαι αυτή τη στιγμή)
  - ⋆ Μέλλον: Αυτή η κατηγορία αναφέρεται σε πλαίσια που αναφέρονται σε προκαθορισμένα ή προβλέψιμα γεγονότα στο μέλλον (π.χ.τοποθεσία αυριανής συνάντησης) Μπορούν να χρησιμοποιηθούν σε υπηρεσίες η εφαρμογές χρηστών όπου γίνεται πρόβλεψη μελλοντικών γεγονότων και αυτού του είδους τα πλαίσια είναι ιδιαίτερα χρήσιμα σε χρήστες που αλλάζουν τοποθεσίες ή γίνονται συνδρομητές σε καινούργιες υπηρεσίες.
- **Αλληλεπίδραση πηγών πληροφορίας πλαισίου με καταβόθρες πληροφορίας πλαισίου**
  - ⋆ Ώθηση πλαισίων: Οι πηγές πλαισίων περιοδικά ενημερώνουν τις καταβόθρες με φρέσκια πληροφορία.
  - ⋆ Ανάκτηση πλαισίων: Οι καταβόθρες περιοδικά ή όταν παραστεί ανάγκη αιτούνται ενημερωμένων πλαισίων πληροφορίας από τις πηγές.

Από την παραπάνω περιγραφή των ιδιοτήτων της πληροφορίας πλαισίου και της δυνατότητας ενσωμάτωσης του σε περιβάλλοντα εφαρμογών κινητού υπολογισμού, γίνεται αντιληπτό ότι είναι απαραίτητη η συσχέτιση χρονοσφραγίδων, περιόδων εγκυρότητας ή ακόμη και κάποιου μέτρου ποιότητας.

2.2 Επίγνωση Πληροφορίας Πλαισίου

Οι ευκολίες που μπορούν να παρασχεθούν σε ένα σύστημα όταν αυτό έχει πλήρη «επίγνωση» του τι συμβαίνει γύρω του είναι τεράστιες. Η πληροφορία που το επηρεάζει είναι αχανής και εκτείνεται από χωρικές και χρονικές παραμέτρους ως και παράγοντες που δεν είναι πάντα τόσο προφανείς αλλά μπορεί να αποδειχθούν κρίσιμοι. Αν ένα σύστημα μπορέσει να φτάσει στο σημείο να «προσαρμόζεται» κάθε φορά στο περιβάλλον του κι ακόμη στους ίδιους του τους χρήστες με αξιοπιστία και προνοητικότητα, τότε θα μπορούμε να μιλάμε για ένα πραγματικά εύχρηστο, χρήσιμο, κατά το δυνατόν διαθέσιμο και ανθρωποκεντρικό προϊόν τεχνολογίας.

Μέγιστη σημασία, λοιπόν, δίνεται στην έννοια της «επίγνωσης του πλαισίου» (context awareness). Η επίγνωση πλαισίου μπορεί να οριστεί ως:
«η ικανότητα ενός συστήματος να ανακαλύπτει, να διερμηνεύει, να συμπεραίνει, να αξιοποιεί και να συλλογίζεται βάσει της περιρρέουσας πληροφορίας ώστε να λαμβάνει αποφάσεις, να προβαίνει σε προκαθορισμένες ενέργειες και να προσαρμόζεται σε διάφορες καταστάσεις» [6].

Με μια προσεκτική και δομημένη εισαγωγή της «επίγνωσης πλαισίου» στην υπολογιστική ικανότητα που διέπει την καθημερινότητά μας, πολλές δραστηριότητες μπορούν να γίνουν πολύ πιο απλές και αποδοτικές με ελάχιστη συμβολή και διαμεσολάβηση του χρήστη. Η δυνατότητα για επικοινωνία μπορεί όχι μόνο να διευκολυνθεί αλλά και να ενσωματωθεί με φυσικό τρόπο στη ζωή μας. Αυτά τα δύο δένονται με συνεχή και διάφανο τρόπο έχοντας ως συνδετικό κρίκο το πλαίσιο, την αναπαράστασή του, την επίγνωσή του, τον συμπερασμό του, την προσαρμογή του στις καταστάσεις του χρήστη / δικτύου καθώς και την έγκαιρη πρόβλεψή του.

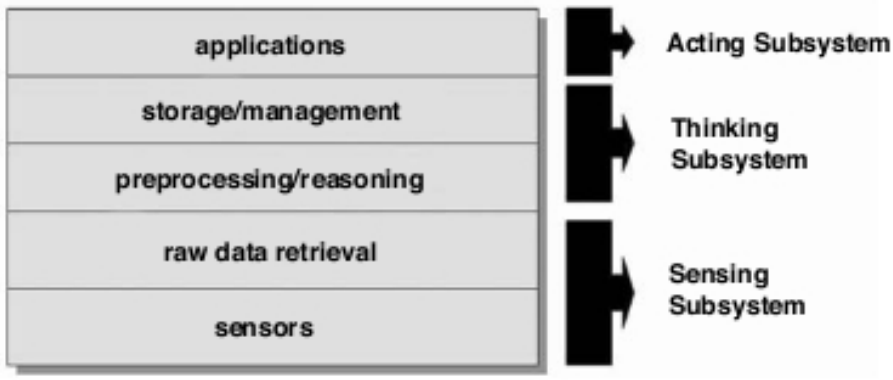
Η επίγνωση πλαισίου ακούγεται άψογη στη θεωρία και είναι αντικείμενο επιστημονικής έρευνας το πώς θα μπορέσει να ενσωματωθεί πραγματικά σε μια «Εφαρμογή Επίγνωσης Πλαισίου» -ΕΕΠ- (context-aware application). Σημαντικά είναι, για παράδειγμα, τα προβλήματα που σχετίζονται με την αλληλεπίδραση ανθρώπου και υπολογιστή. Μέσω του πλαισίου, η σχέση άνθρωπος-υπολογιστής ορίζεται σχεδόν εκ νέου. Πώς θα μπορούσε ο χρήστης να αισθάνεται ασφαλής και ικανοποιημένος απέναντι στο τρέχον σύστημά του ενώ ταυτόχρονα το τελευταίο να παίρνει όσο το δυνατόν περισσότερες και όσο το δυνατόν πιο πρώιμες αποφάσεις;

Ας δούμε ένα απλό παράδειγμα όπου μια ποικιλία τεχνολογιών αναζήτησης και επεξεργασίας πληροφορίας θα μπορούσαν να αξιοποιηθούν για να υποβοηθήσουν τη δραστηριότητα χρηστών που έχουν ανάγκη από «υπηρεσίες βασισμένες στην θέση» και την κίνησή τους (location-based services). Ας φανταστούμε μία κινητή συσκευή που επιτρέπει στον χρήστη της να κινείται σε ένα συνεδριακό χώρο από αίθουσα σε αίθουσα, να έχει πρόσβαση στις πληροφορίες που σχετίζονται άμεσα με το σημείο στο οποίο βρίσκεται (π.χ., θέμα συζήτησης, ροή παρουσιάσεων), να κρατά ιστορικό της διαδρομής και των σημείων όπου ο χρήστης έδειξε το μεγαλύτερο ενδιαφέρον (επειδή ίσως στάθηκε εκεί για περισσότερο χρόνο) και να τον ενημερώνει για τη δραστηριότητα των συναδέλφων του (τοποθεσία τους, τι θεώρησαν εκείνοι ενδιαφέρον και στάθηκαν να παρατηρήσουν). Γίνεται εύκολα αντιληπτό, λοιπόν, ότι η επίγνωση πλαισίου είναι ένα ιδιαίτερα ευρύ θέμα και γι' αυτό έχει και ευρύ φάσμα θεωρήσεων.

Όμως αντικείμενο έρευνας είναι και το ζήτημα της «αναπαράστασης» (representation) και «ερμηνείας» (interpretation) της πληροφορίας πλαισίου. Η πληροφορία που συλλέγεται από πολλές (διαφορετικές) πηγές ερμηνεύεται διαφορετικά από κάθε ΕΕΠ. Επομένως, αναπτύσσονται μηχανισμοί μέσω των οποίων το πλαίσιο ελέγχεται και ερμηνεύεται κατάλληλα βάσει της μελλοντικής του χρήστης και αξιοποίησης.

2.3 Αρχιτεκτονική Συστήματος Επίγνωσης Πληροφορίας Πλαισίου

'Ενα Σύστημα Επίγνωσης Πληροφορίας Πλαισίου (ΣΕΠ) θα πρέπει να ει τρεις βασικές λειτουργίες: της αίσθησης (sensing) της περιρέουσας κατάστασης, της σκέψης (thinking) μεταφορικά και της δράσης (acting). Στο Σχήμα 1 φαίνεται η αρχιτεκτονική ενός ΣΕΠ. Για να μπορέσουν να προσλάβουν πληροφορία από το πλαίσιο στο



Σχήμα 1: Αρχιτεκτονική ενός Συστήματος Επίγνωσης Πληροφορίας Πλαισίου

οποίο βρίσκονται (περιβάλλον, περιρρέουσα κατάσταση) έχουν την ανάγκη να εξοπλιστούν με ειδικούς αισθητήρες και πηγές πληροφόρησης που τους επιτρέπουν να έχουν πρόσβαση στην πληροφορία αυτή. Οι αισθητήρες είναι εξειδικευμένα εξαρτήματα, τα οποία δεν έχουν σαν πρώτιστο στόχο την υπολογιστική ικανότητα αλλά την πρόσληψη των παραμέτρων του περιβάλλοντος στο μοντέλο του υπολογισμού.

Οι κινητές συσκευές που έχουν ενσωματωμένες δυνατότητες άμεσης επίγνωσης πλαισίου διαθέτουν πολλαπλούς αισθητήρες και αλγόριθμους για την επεξεργασία των δεδομένων που λαμβάνουν από το περιβάλλον σε δεδομένα χρήσιμα (αξιοποιήσιμα) για τις λειτουργίες τους.

Οι αισθητήρες θέσης παρέχουν πληροφορία που σχετίζεται με την τοποθεσία σαν ανεξάρτητο αλλά και ιδιαίτερα χρήσιμο δεδομένο. Συχνά, βέβαια, δεν είναι η τοποθεσία που συνιστά μόνη της άμεσα αξιοποιήσιμο δεδομένο αλλά σε συνδυασμό με επιπρόσθετη πληροφορία που μπορεί να παραχθεί (π.χ. οι πόροι που είναι διαθέσιμοι σε κάποιο σημείο). Επίσης, οι διαφορές μετρήσεις από αισθητήρες μπορούν να ορίσουν μια πιο σύνθετη αναπαράσταση και επίγνωση του πλαισίου (π.χ., η κατάσταση πυρκαγιάς επάγεται από μετρήσεις αισθητήρων καπνού, θερμοκρασίας και υγρασίας). Ακόμη, οι κάμερες παρέχουν δυνητικά πολύ πλούσιο υλικό, από το οποίο είναι δυνατόν να εξορυχτεί η απαραίτητη πληροφορία.

Για να μπορέσει ένα ΣΕΠ να λάβει εικόνα της περιρρέουσας κατάστασης μέσα από τις μετρήσεις των αισθητήρων και να αποφασίσει για περαιτέρω ενέργειες, θα πρέπει να εξορύξει αυτές τις πληροφορίες με τη βοήθεια μαθηματικών μοντέλων και μεθόδων[7]. Τέτοια μοντέλα είναι τα ακόλουθα:

- Φυσικά μαθηματικά μοντέλα (Kalman filetring)
- Pattern recognition, neural networks,cluster algorithms
- Cognitive based models, fuzzy logic

Κατά το τρίτο στάδιο της λειτουργίας ενός ΣΕΠ, θα πρέπει να πραγματοποιηθούν διάφορες ενέργειες για να ικανοποιηθούν οι ανάγκες της εφαρμογής την οποία ε-

ξυπηρετεί το συγκεκριμένο ΣΕΠ. Αυτές οι ενέργειες μπορεί να παραπέμπουν ακόμη και σε επιπρόσθετη συλλογή δεδομένων και θα πρέπει να πραγματοποιηθούν αρκετά γρήγορα έτσι ώστε να έχουν νόημα και να συμβαδίζουν με τη περιρρέουσα κατάσταση. Για παράδειγμα, έστω μια εφαρμογή η οποία στέλνει διαφημίσεις μέσα από ένα ασύρματο δίκτυο στο κινητό τηλέφωνο ενός χρήστη, παίρνοντας υπόψιν τη θέση του χρήστη καθώς αυτός κινείται. Αν ο χρήστης κινείται αρκετά γρήγορα, ή το σύστημα είναι πολύ αργό στο να βρει και να χρησιμοποιήσει την πληροφορία θέσης που αποτελεί και την πληροφορία πλαισίου της εφαρμογής, τότε είναι πιθανόν ο χρήστης να μην λαμβάνει έγκαιρα τις διαφημίσεις.

Τα στοιχεία που πρέπει να ληφθούν υπόψιν για το σχεδιασμό ενός ΣΕΠ σχετίζονται με τη σωστή λειτουργία των τριών σταδίων που αναφέρθηκαν παραπάνω, τη σωστή επιλογή της πληροφορίας πλαισίου αλλά και την αναγνώριση της περιρρέουσας κατάστασης που θα περιβάλλει το υπό σχεδιασμό σύστημα.

Επομένως τα επιθυμητά χαρακτηριστικά για ένα ΣΕΠ είναι τα ακόλουθα:

Διεισδυτικότητα (pervasiveness). Να μπορεί να συνεργάζεται πολύ πιο αποτελεσματικά με τους χρήστες του και να δρα αυτόνομα στις απαιτούμενες ενέργειες,

Προ-δραστικότητα (proactiveness). Να μπορεί να προλαμβάνει τις ανάγκες που θα παρουσιαστούν στο κοντινό μέλλον και κατ' επέκταση να προσαρμοστεί κατάλληλα προκειμένου να τις αντιμετωπίσει,

Ικανότητα Συμπερασμού Πλαισίου (inference). Να μπορεί να επάγει συμπεράσματα από το πλαίσιό του (περαιτέρω πληροφορία και επαύξηση γνώσης), να μπορεί όχι μόνον να συλλογιστεί για την περιρρέουσα κατάσταση αλλά και να παρέχει πολύ πιο εξειδικευμένες υπηρεσίες στους χρήστες του.

Από την άλλη πλευρά, διάφορα ζητήματα αναδύονται κατά το σχεδιασμό ενός ΣΕΠ. Ενδεικτικά:

- Η πληροφορία πλαισίου πρέπει να αναπαρασταθεί κατάλληλα (context representation) με καθορισμένα «μοντέλα αναπαράστασης πλαισίου» (context model) ώστε το εκάστοτε μοντέλο γνώσης του πλαισίου να αντικατοπτρίζει το τρέχον πλαίσιο και να επιτυγχάνεται ο κατάλληλος συμπερασμός (context inference) και συλλογισμός του (context reasoning).
- Οι συσκευές / συστήματα πρέπει να διαθέτουν κατάλληλους ανιχνευτές για την πρόσληψη της αναγκαίας πληροφορίας τόσο από το περιβάλλον όσο και από τους χρήστες με έξυπνο και διάχυτο τρόπο (δηλαδή με όσο τον δυνατό λιγότερη επέμβαση του χρήστη).
- Η πληροφορία που συλλέγεται από το σύστημα και τους ανιχνευτές του είναι σε μεγάλο βαθμό ετερογενής και έτσι πολλές μέθοδοι και αλγόριθμοι «σύντηξης» (context fusion) και «συνάθροισης πλαισίου» (context aggregation) μπορούν να αναπτυχθούν.
- Η «ταξινόμηση» (classification) και «πρόβλεψη» (prediction) του πλαισίου είναι κυρίως αναγκαία σε συστήματα με υπολογιστικές ικανότητες, πράγμα το

οποίο εισάγει ένα σύνολο αλγορίθμων που μπορούν να χρησιμοποιηθούν (π.χ., αλγόριθμοι Μηχανικής Μάθησης).

- Η «εκμάθηση» (training) και η «προσαρμογή» (adaptation) από την πλευρά του συστήματος πρέπει να γίνεται σε πραγματικό χρόνο και όχι να έπεται της κατάστασης την όποία καλείται να διευκολύνει.
- Η «αλληλεπίδραση» (user interaction / user intervention) με το χρήστη πρέπει να μείνει σε χαμηλό επίπεδο και οπωσδήποτε να διεκπεραιώνεται όσο το δυνατόν πιο διακριτικά.

Συνεπώς, η επίγνωση πλαισίου είναι πλέον ένα αναπτυσσόμενο ερευνητικό πεδίο. Ένα μεγάλο ζητούμενο είναι η εξισορρόπηση ανάμεσα στην παρεμβολή από πλευράς χρήστη και την αυτονομία και προνοητικότητα που οφείλει να δείχνει το σύστημα. Καινοτόμα συστήματα που βασίζονται στο πλαίσιο δε μπορούν να κατασκευαστούν αν δεν εισάγουν στην αρχιτεκτονική τους μηχανισμούς προσαρμογής και πρόβλεψης του πλαισίου για την κατάλληλη εκμετάλλευση της παραγόμενης και επαγόμενης πληροφορίας. Επίσης, ως γνωστόν, ο άνθρωπος βασίζεται στη συνήθεια και δρα, τις περισσότερες φορές, βάσει συγκεκριμένων προτύπων συμπεριφοράς. Διάφορες μέθοδοι αναγνώρισης και προσδιορισμού καταστάσεων υιοθετούνται στην επίγνωση πλαισίου. Ένα σύστημα που εκμεταλλεύεται τέτοια πρότυπα πρέπει να μαθαίνει αυτόματα τις συνήθειες των χρηστών του και να προσαρμόζεται δυναμικά στις αλλαγές τους. Πρέπει, επομένως, να στηρίζεται στη γνώση της «συμπεριφοράς του παρελθόντος» (historical context) για να προβλέψει και να προσαρμοστεί σε συμπεριφορές του μέλλοντος. Αν πάλι η προσπάθεια πρόβλεψης αποτυγχάνει, η αποτυχία αυτή πρέπει να αναγνωρίζεται και να λαμβάνεται υπόψη για τις επόμενες απόπειρες πρόγνωσης και προσαρμογής (context adaptation).

2.4 Μοντελοποίηση Πληροφορίας Πλαισίου

Η Μοντελοποίηση Πλαισίου είναι μια περιοχή έρευνας της Επίγνωσης Πλαισίου που εστιάζει στην μελέτη των εξής προβλημάτων:

- ποια είναι η καταλληλότερη πληροφορία πλαισίου (π.χ., μεταβλητές, παράμετροι, γνωρίσματα) που μπορεί να αναπαραστήσει όσο το δυνατό καλύτερα την πληροφορία (π.χ., χρόνος, θέση, περιβαλλοντικά μεγέθη, έννοιες, δραστηριότητες) για ένα συγκεκριμένο επιστημολογικό πεδίο (π.χ., Location-based Services). Οι μεταβλητές αυτές καλούνται «συνιστώσες» (contextual ingredients) της πληροφορίας πλαισίου εφόσον φανταστούμε το πλαίσιο ως ένα πολυδιάστατο διάνυσμα (multivariate context vector ή context vector) σε ένα διανυσματικό χώρο καθοριζόμενο από μεταβλητές που αναπαριστάνουν την πληροφορία του εκάστοτε επιστημολογικού πεδίου. Αξίζει να σημειωθεί ότι στον χώρο των διανυσμάτων αυτών οι μεταξύ τους συνιστώσες δεν είναι ανεξάρτητες πάντα.
- ποιες είναι οι ενάγουσες και υπάρχουσες εξαρτήσεις – συσχετίζεις μεταξύ των συνιστωσών αυτών (π.χ., χρονικές εξαρτήσεις, σχέσεις εκλέπτυνσης / εξει-

δίκευσης και γενίκευσης, εξαρτήσεις μέρους-όλο / μερεολογικές, εξαρτήσεις συμβατότητας, εξαρτήσεις περιορισμών, έτερο-συσχετίσεις), και,

- ποιες είναι οι μέθοδοι εκμάθησης, ταξινόμησης και προσαρμογής του διανύσματος του πλαισίου εφόσον αυτό κρίνεται αναγκαίο. Ο κύριος σκοπός αυτής της Διπλωματικής εργασίας εστιάζεται στο μέρος εκείνο του σχεδιασμού ενός ΣΕΠ που αναφέρεται στην εξασφάλιση συνεχόμενης και χρονικά έγκυρης πληροφορίας πλαισίου που τροφοδοτεί κάποια εφαρμογή(π.χ. σύστημα παρακολούθησης κυκλοφορίας, παρακολούθηση δασικών εκτάσεων για πρόληψη πυρκαγιών) έτσι ώστε να αντιδρά άμεσα σε ταχύτατα μεταβαλλόμενες περιρρέουσες καταστάσεις.

3 Νοημοσύνη Σμήνους (Swarm Intelligence) σε Περιβάλλοντα Κινητού Υπολογισμού

3.1 Το Βιολογικό Ανάλογο

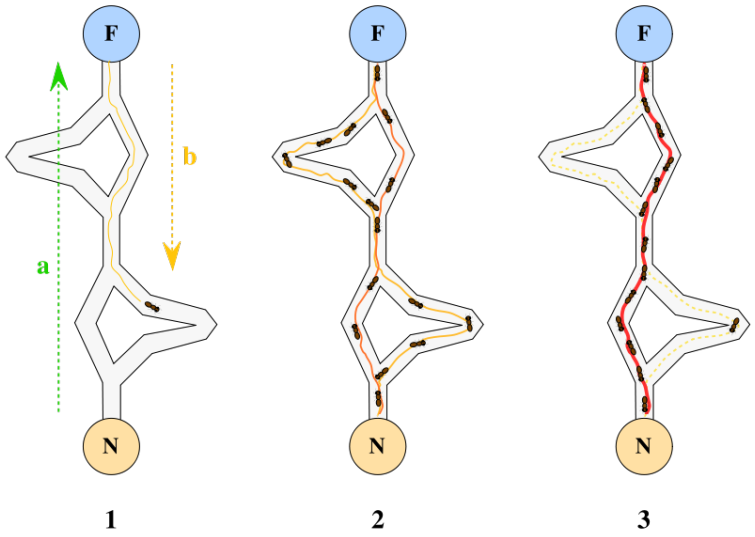
Τα έντομα που ζουν σε αποικίες (μυρμήγκια, μέλισσες, σφήκες, τερμίτες κτλ.) αλλά και τα διάφορα είδη ψαριών και πουλιών που κινούνται σε κοπάδια (flocks) ήταν πάντα αντικείμενο θαυμασμού για τον άνθρωπο. Από πού πηγάζουν και πώς αναπτύσσονται οι οργανωμένες συμπεριφορές των κοινωνικών εντόμων; Κάθε έντομο σε μια αποικία και κάθε πουλί στο σμήνος του φαίνεται να έχει τη δική του ημερήσια διάταξη και όμως σαν σύνολο δείχνουν μια υψηλού επιπέδου οργάνωση. Προφανώς αυτή η αόρατη ολοκλήρωση των μεμονωμένων δραστηριοτήτων δεν απαιτεί κάποια εποπτεία.

Η μελέτη της συμπεριφοράς των κοινωνικών εντόμων αποκάλυψαν ότι η συνεργασία σε επίπεδο αποικίας είναι σε μεγάλο βαθμό αυτο-οργανωνόμενη: στη μεγάλη πλειοψηφία των περιπτώσεων προκύπτει από αλληλεπιδράσεις μεταξύ των μεμονωμένων ατόμων. Αν και αυτές οι αλληλεπιδράσεις μπορεί να είναι απλές (π.χ. ένα μυρμήγκι απλώς ακολουθεί το μονοπάτι που χάραξε ένα άλλο), όλες μαζί μπορούν να λύσουν δύσκολα προβλήματα (όπως η εξεύρεση του συντομότερου δρόμου ανάμεσα σε αμέτρητες δυνατές διαδρομές προς την τροφή). Αυτή η συλλογική συμπεριφορά που προκύπτει από μια ομάδα κοινωνικών εντόμων ονομάστηκε «νοημοσύνη σμήνους».

Μια από τις πρώιμες μελέτες της νοημοσύνης σμήνους ασχολήθηκε με την αναζήτηση τροφής από τα μυρμήγκια. Απέδειξε, ότι οι γραμμές μυρμηγκιών που βλέπουμε συχνά στη φύση ή στην κουζίνα μας είναι αποτέλεσμα της έκκρισης από τα μυρμήγκια μιας χημικής ουσίας που προσελκύει τα άλλα μυρμήγκια, της φερομόνης (pheromone). Η μελέτη επισήμανε, επίσης, ότι η διαδικασία της έκκρισης φερομόνης για τη δημιουργία ενός μονοπατιού, που μπορούν να ακολουθήσουν τα άλλα έντομα, είναι μια καλή στρατηγική για την εύρεση της συντομότερης διαδρομής ανάμεσα στη φωλιά και στην τροφή.

Σε σχετικά πειράματα (βλ. Σχήμα 2) δημιουργήθηκαν ανάμεσα στη φωλιά και την τροφή των μυρμηγκιών τέσσερις διαδρομές διαφορετικού μήκους. Μέσα σε μερικά λεπτά η αποικία συνήθως επέλεγε το συντομότερο δρόμο. Τα πρώτα μυρμήγκια που επέστρεφαν στη φωλιά από την τοποθεσία της τροφής ήταν εκείνα που είχαν ακολουθήσει το σύντομο δρόμο τόσο στον πηγεμό, όσο και στο γυρισμό. Επειδή αυτή η διαδρομή ήταν η πρώτη που είχε διπλό ίχνος φερομόνης, τα άλλα μυρμήγκια την προτιμούσαν, με αποτέλεσμα να ενισχυθεί κι άλλο και να γίνει η οδός πρόσβασης προς την τροφή.

Το μυστικό της «επιτυχημένης» διεκπεραίωσης συλλογικών εργασιών και μάλιστα με βέλτιστο τρόπο βρίσκεται στην αυτο-οργάνωση (self-organization-SO) των εντόμων. Στα μοντέλα αυτο-οργάνωσης γίνεται η βασική θεώρηση ότι το κάθε άτομο-



Σχήμα 2: 1) Το πρώτο μυρμήγκι επιλέγει μια διαδρομή προς την τροφή (F) και επιστρέφει στη φωλιά (N) αφήνοντας ίχνη φερομόνης 2) τα άλλα μυρμήγκια ακολουθούν κάποια από τις τέσσερις διαδρομές 3) η διαδρομή με την μεγαλύτερη εναπόθεση φερορμόμης, που είναι και η συντομότερη, γίνεται προτιμητέα

έντομο είναι ένας απλός «πράκτορας» (agent) που μπορεί να διεκπεραιώσει μόνο απλές λειτουργίες. Η θεώρηση αυτή δεν λαμβάνει υπόψη την περίπλοκη δομή του μεμονωμένου εντόμου. Για παράδειγμα, στα SO μοντέλα αυτό που έχει σημασία είναι η μετακίνηση της μέλισσας μεταξύ δύο σημείων και όχι πως αυτό πραγματοποιήθηκε, δηλαδή πια αισθητήρια μέσα χρησιμοποίησε η μέλισσα για τον εντοπισμό της θέσης της και του στόχου της, αν έχει περιορισμένη ικανότητα μνήμης κτλ. Ομοίως, στις αποικίες των μυρμηγκιών το βασικό χαρακτηριστικό που λαμβάνουν υπόψη τα SO μοντέλα είναι ότι τα μυρμήγκια μπορούν να ακολουθήσουν με κάποια πιθανότητα ένα ίχνος φερομόνης και να το ενισχύσουν χωρίς να ενδιαφέρει πως ανιχνεύεται το ίχνος αυτό, πως λαμβάνεται η απόφαση από το μυρμήγκι να το ακολουθήσει ή όχι και τέλος με ποιον τρόπο ενισχύει το ήδη υπάρχον ίχνος φερομόνης.

Τα SO μοντέλα που έχουν αναπτυχθεί με βάση την παραπάνω θεώρηση δείχνουν ότι είναι δυνατόν να εμφανιστούν περίπλοκες συλλογικές συμπεριφορές σε μια αποικία εντόμων. Αναλυτικότερα, η αυτο-οργάνωση [9] είναι ένα σύνολο δυναμικών μηχανισμών με τους οποίους σχηματίζονται δομές σε ένα σύστημα από αλληλεπιδράσεις των συνιστωσών του. Οι αλληλεπιδράσεις αυτές στηρίζονται καθαρά σε τοπικές πληροφορίες χωρίς να λαμβάνουν υπόψη τους τη συνολική εικόνα.

Αυτές οι παρατηρήσεις από το ζωικό βασίλειο οδήγησαν σε υπολογιστικές προσομοιώσεις αρχικά με την τεχνική της εξάτμισης της φερομόνης, όπου δημιουργήθηκαν αποικίες από εικονικά μυρμήγκια και πηγές τροφής σε διαφορετική απόσταση από τη φωλιά τους. Στην αρχή, τα εικονικά μυρμήγκια εξερεύνησαν το γύρω χώρο με τυχαίο τρόπο. Στη συνέχεια αποκατέστησαν μονοπάτια που συνέδεαν όλες τις πηγές

τροφής με τη φωλιά. Μετά διατήρησαν μόνο τα μονοπάτια προς τις πλησιέστερες πηγές τροφής, με αποτέλεσμα τη σταδιακή εξάντληση των πηγών αυτών. Οταν η τροφή εξαντλήθηκε, τα εικονικά μυρμήγκια άρχισαν να αξιοποιούν πιο μακρινές πηγές.

Αυτό το φαινόμενο της σύγκλισης σε κάποια βέλτιστη λύση από κάποιες αρχικές χαοτικές συνθήκες μέσω της αυτο-οργάνωσης, οδήγησε πολλούς ερευνητές, κυρίως από το χώρο της Πληροφορικής και των Επικοινωνιών, να θεμελιώσουν και να χρησιμοποιήσουν τέτοιους βιο- εμπνευσμένους (bio-inspired) αλγορίθμους σε προβλήματα βελτιστοποίησης (π.χ. δρομολόγησης σε ασύρματα δίκτυα) και αυτο-οργάνωσης (π.χ. σμήνη ρομπότ) με εξαιρετική επιτυχία. Δύο από τους πιο σημαντικούς και δημοφιλείς αλγορίθμους περιγράφονται στην επόμενη ενότητα.

3.2 Αλγόριθμοι Νοημοσύνης Σμήνους

3.2.1 Ο αλγόριθμος Ant Colony Optimisation (ACO)

Ο αλγόριθμος αυτός επινοήθηκε από τον Dorigo [8] στην προσπάθειά του να επιλύσει με κάποιο εναλλακτικό αλγόριθμο το γνωστό πρόβλημα του Περιπλανώμενου Πωλητή (Travelling Salesman Problem), δηλαδή να βρει την ελάχιστη διαδρομή που πρέπει να διατρέξει ο πωλητής προκειμένου να επισκεφθεί όλες τις πόλεις από μία φορά. Για αυτό και η περιγραφή του αλγορίθμου θα βασιστεί πάνω στη λύση αυτού του προβλήματος. Τον εμπνεύστηκε από διάφορες μελέτες που αναφέρονταν στη συμπεριφορά και στην αυτο-οργάνωση αποικιών μυρμηγκιών στην προσπάθεια τους να εξασφαλίσουν τροφή.

Η κάθε λύση για το TSP σχηματίζεται με τη διαδοχική μετάβαση των «ψηφιακών» μυρμηγκιών από τη μια πόλη στην άλλη με κάποια πιθανότητα. Το «ταξίδι» ολοκληρώνεται όταν το «ψηφιακό» μυρμήγκι επιστέψει στην αρχική πόλη. Τότε βαθμολογείται η λύση που έχει επιτύχει με κριτήριο το συνολικό μήκος της διαδρομής και προστίθεται η ανάλογη φερομόνη στο μονοπάτι που ακολούθησε. Η παραπάνω διαδικασία επαναλαμβάνεται για κάθε άτομο της αποικίας μέχρι να συμπληρωθεί ο ζητούμενος αριθμός επαναλήψεων.

Τα μυρμήγκια τοποθετούνται είτε τυχαία στις πόλεις είτε το καθένα σε μια διαφορετική πόλη ως αφετηρία της διαδρομής τους. Η πόλη αυτή καταγράφεται ως νούμερο «ένα» στη λίστα της μνήμης κάθε μυρμηγκιού έτσι ώστε να αποκλειστεί από τις επόμενες επιλογές του. Η επιλογή της επόμενης πόλης την οποία θα επισκεφθεί κάθε μυρμήγκι ορίζεται από μια πιθανότητα μετάβασης η οποία εξαρτάται από την «ορατότητα» του κάθε μυρμηγκιού (ορατότητα ορίζεται ως το αντίστροφο της απόστασης μεταξύ δύο πόλεων και άρα εισάγει μεγάλη πιθανότητα στον κανόνα μετάβασης όσον αφορά τις κοντινές πόλεις) και στο ποσοστό της φερομόνης που έχει εναποτεθεί στην κάθε διαδρομή εκφράζοντας την εμπειρία της υπόλοιπης αποικίας.

Αφού όλα τα μυρμήγκια ολοκληρώσουν τα «ταξίδια» τους σε κάθε επανάληψη, τότε προστίθεται φερομόνη σε κάθε μονοπάτι ανάλογα με την επίδοση του κάθε μυρμηγκιού. Σημαντικός παράγοντας στην όλη διαδικασία είναι και η εισαγωγή ενός

παράγοντα σχετικού με την εξάτμιση της φερομόνης. Κατά αυτόν τον τρόπο αποφεύγεται η ενίσχυση των αρχικών τυχαίων διακυμάνσεων. Τα βήματα του αλγορίθμου ACO είναι τα ακόλουθα:

Βήμα 1ο: Θέτουμε τη φερομόνη στην αρχική της τιμή (είτε σε μια πολύ μικρή τιμή είτε σε μια τυχαία τιμή) σε όλα τα μονοπάτια που συνδέουν τις πόλεις μεταξύ τους. Υπολογίζουμε όλες τις αποστάσεις των πόλεων. Επιλέγουμε (τυχαία) την πόλη-αφετηρία της διαδρομής κάθε μυρμηγκιού και την τοποθετούμε στην λίστα της μνήμης του.

Βήμα 2ο: Για κάθε μυρμήγκι, επιλέγουμε την επόμενη πόλη που θα επισκεφτεί με βάση τον τυχαίο αναλογικό κανόνα μετάβασης μέχρι να ολοκληρώσει τη διαδρομή του επισκεπτόμενο κάθε πόλη μια μόνο φορά και τελικά να επιστρέψει στην αφετηρία του.

Βήμα 3ο: Καταγράφουμε την καλύτερη διαδρομή που βρέθηκε.

Βήμα 4ο: Ανανεώνουμε τη φερομόνη στα μονοπάτια που επισκέφτηκαν τα μυρμήγκια με βάση τον κανόνα ανανέωσης της φερομόνης

Βήμα 5ο: Επαναλαμβάνουμε τη διαδικασία από το βήμα 2 μέχρι έως ότου ολοκληρωθεί ένας συγκεκριμένος αριθμός επαναλήψεων ή επιτευχθεί ένα κριτήριο σύγκλισης.

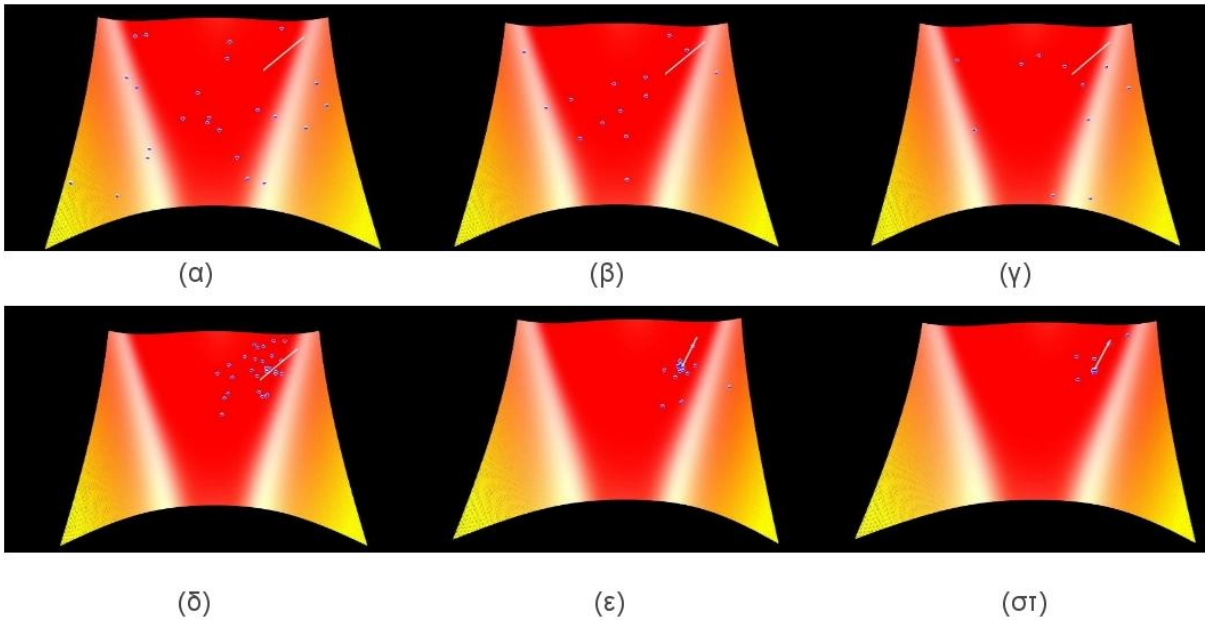
Παραλλαγές [12] ή/και τροποποιήσεις ([10],[11]) του παραπάνω αλγορίθμου έχουν επίσης αναπτυχθεί για να αντιμετωπίσουν την επίλυση συγκεκριμένων προβλημάτων ή να βελτιώσουν την απόδοση του αρχικού αλγορίθμου που μόλις περιγράφηκε.

3.2.2 Ο αλγόριθμος Partcile Swarm Optimisation (PSO)

Ο αλγόριθμος PSO χρησιμοποιείται σε πολλές εφαρμογές βελτιστοποίησης, όπως δρομολόγηση κυκλοφορίας σε τηλεπικοινωνιακά συστήματα, σχεδιασμό αλγορίθμων για έλεγχο αυτόνομων ρομπότ κλπ., βασιζόμενος στην αλληλεπίδραση οντοτήτων μεταξύ τους μέσα σε μια αγέλη σωματιδίων.

Η αρχική ιδέα δημιουργήθηκε από τους Eberhart και Kennedy [13] και βελτιώθηκε από τους Clerc και Kennedy[14]. Αποτελεί μεταφορά του βιολογικού αναλόγου της κοινωνικής συμπεριφοράς και αυτο-οργάνωσης διαφόρων ειδών οντοτήτων (αγέλες ψαριών και πουλιών). Οι οντότητες αυτές τοποθετούνται σε ένα χώρο όπου ορίζεται μια συνάρτηση και εκτελείται από κάθε οντότητα αναζήτηση των θέσεων όπου πραγματοποιείται μεγιστοποίηση αυτής της συνάρτησης και τις οποίες χρησιμοποιούν για την περαιτέρω πλοήγηση τους μέσα στο χώρο, με σκοπό να μεγιστοποιήσουν ακόμη περισσότερο τη συγκεκριμένη συνάρτηση και να τείνουν προς την/τις πηγές τροφής. Οι πληροφορίες που συλλέγονται από την κάθε οντότητα χρησιμοποιούνται όχι μόνο για τη δική τους πλοήγηση, αλλά διαχέονται και σε άλλα μέλη της αγέλης που τυχόν βρεθούν στη γειτονιά τους επιδεικνύοντας μια συνεργατική συμπεριφορά [15].

Όπως αναφέρθηκε προηγουμένως ο αλγόριθμος PSO προσομοιώνει τη συμπεριφορά μιας αγέλης πουλιών ή ψαριών. Ας φανταστούμε το ακόλουθο σενάριο: μια ομάδα πουλιών αναζητά τροφή σε μια περιοχή πετώντας με τυχαίο τρόπο. Υπάρχει μόνο μια ποσότητα τροφής στην περιοχή και τα πουλιά δεν γνωρίζουν που ακριβώς βρίσκεται αλλά γνωρίζουν πόσο μακριά από την τροφή βρίσκονται σε κάθε χρονική στιγμή. Άρα ποια είναι η καλύτερη στρατηγική για ένα μέλος του σμήνους να βρει την τροφή; Η πιο αποτελεσματική λύση είναι να ακολουθήσει το πουλί που βρίσκεται πιο κοντά στην τροφή.



Σχήμα 3: Επίδειξη του αλγορίθμου PSO για 40 σωματίδια, Rosenbrock fitness function. Η άσπρη γραμμή αντιπροσωπεύει το minimum της συνάρτησης (τροφή) στο οποίο τελικά συγκλίνουν τα σωματίδια. [16]

Ο αλγόριθμος PSO είναι εμπνευσμένος από αυτό το σενάριο και χρησιμοποιείται για τη λύση προβλημάτων βελτιστοποίησης. Στον PSO μία λύση είναι ένα «πουλί» στο χώρο αναζήτησης και το οποίο ονομάζεται «σωματίδιο» (particle) ή πράκτορας(agent). Όλα τα σωματίδια κατέχουν κάποιες τιμές καταλληλότητας (fitness values) οι οποίες επαληθεύονται από τη συνάρτηση που πρέπει να μεγιστοποιηθεί. Έχουν επίσης διανύσματα ταχυτήτων που τα κατευθύνουν στο «πέταγμα» τους. Τα σωματίδια «πετούν» μέσα στο χώρο του προβλήματος προς βελτιστοποίηση και ακολουθούν τα σωματίδια με τις βέλτιστες τιμές καταλληλότητας ως εκείνη τη στιγμή. Στο Σχήμα 3 δίνονται κάποια στιγμιότυπα από τη διαδικασία σύγκλισης του αλγορίθμου για σμήνος 40 σωματιδίων. Ως συνάρτηση καταλληλότητας χρησιμοποιείται η συνάρτηση Rosenbrock που δίνεται από την $f(x, y) = (1 - x)^2 + 100(y - x^2)^2$.

Ο αλγόριθμος αρχικοποιείται με μια ομάδα τυχαίων σωματιδίων (λύσεων) και κατόπιν αναζητεί βέλτιστες λύσεις ενημερώνοντας σε κάθε χρονική στιγμή. Σε κάθε επανάληψη ενημερώνονται δυο μεταβλητές για κάθε σωματίδιο που ονομάζονται

«καλύτερες τιμές» (best values). Η πρώτη μεταβλητή αναφέρεται στην καλύτερη λύση που ανακάλυψε ένα σωματίδιο ως εκείνη τη στιγμή (*προσωρινή εμπειρία* ή *υποκειμενικός παράγοντας*) και ονομάζεται *pbest* και η δεύτερη μεταβλητή αναφέρεται στην γειτονιά με την καλύτερη λύση (*αντικειμενικός παράγοντας*) και ονομάζεται *gbest*.

Σε κάθε επανάληψη του αλγορίθμου η ταχύτητα και η θέση κάθε σωματιδίου ενημερώνονται σύμφωνα με τις ακόλουθες εξισώσεις

$$v = v + c_1 \cdot rand() \cdot (pbest - present) + c_2 \cdot rand() \cdot (gbest - present) \quad (1)$$

$$present = present + v \quad (2)$$

όπου v η ταχύτητα των σωματιδίων, $present$ η τρέχουσα θέση, $rand()$ γεννήτρια τυχαίων αριθμών που $\in (0, 1)$ και c_1,c_2 συντελεστές βαρών. Ο αλγόριθμος PSO συνοψίζεται στον ακόλουθο ψευδοκώδικα:

∀ (σωματίδιο)
    Αρχικοποίησε σωματίδιο
END
DO
    ∀ (σωματίδιο)
    Υπολόγισε τιμή καταλληλότητας
    Αν η τιμή καταλληλότητας είναι καλύτερη από $pbest$
    Θέσε τρεχουσα τιμή καταλληλότητας ως $pbest$
END
Διάλεξε το σωματίδιο με το καλύτερο $gbest$
∀ (σωματίδιο)
    Υπολόγισε ταχύτητα σωματιδίου σύμφωνα με την (1)
    Υπολόγισε θέση σωματιδίου σύμφωνα με την (2)
END
While μέγιστος αριθμός επαναλήψεων ή επαλήθευση κριτηρίων σύγκλισης.

Ο αλγόριθμος PSO είναι πολύ δημοφιλής για την επίλυση προβλημάτων βελτιστοποίησης λόγω της απλής και εύκολης υλοποίησης, των λίγων παραμέτρων που απαιτούνται αλλά και της εύκολης λόγω της φύσης του, προσαρμογής του σε περιβάλλοντα παράλληλων αρχιτεκτονικών για την επίλυση σύνθετων προβλημάτων.

3.3 Εφαρμογές Αλγορίθμων Νοημοσύνης Σμήνους

Τα τελευταία χρόνια, αυξάνονται διαρκώς οι προσπάθειες εφεύρεσης τρόπων για την εφαρμογή της νοημοσύνης σμήνους σε ποικιλία προβλημάτων. Η αναζήτηση τροφής από τα μυρμήγκια έχει οδηγήσει σε μια νέα μέθοδο για την αναδρομολόγηση της κυκλοφορίας στα σύγχρονα τηλεπικοινωνιακά δίκτυα. Η συνεργατική αλληλεπίδραση των μυρμηγκιών που προσπαθούν να μεταφέρουν ένα μεγάλο κομμάτι τροφής μπορεί να οδηγήσει σε πιο αποτελεσματικούς αλγορίθμους για χρήση σε ρομπότ. Και ο καταμερισμός εργασίας ανάμεσα στις μέλισσες ίσως βοηθήσει σε αποτελεσματικότερες διαδικασίες στις βιομηχανίες με αλυσίδα συναρμολόγησης.

Οι κυριότερες εφαρμογές του αλγορίθμου ACO σε ασύρματα δίκτυα χωρίς συγκεντρωτική δομή (MANET) τα οποία εξάλλου αποτελούν και την κυριότερη πλατφόρμα για εφαρμογές κινητού υπολογισμού αφορούν στην εύρεση μεθόδων αποτελεσματικής δρομολόγησης. Αρκετοί αλγόριθμοι έχουν αναπτυχθεί τα τελευταία χρόνια βασισμένοι στον ACO για την εξάλειψη των προβλημάτων δρομολόγησης που εμφανίζονται στα δίκτυα MANET εξαιτίας της κινητικότητας των πηγών. Ο αλγόριθμος AntHocNet[17] είναι ένας υβριδικός αλγόριθμος που δημιουργεί αρχικά τους πίνακες δρομολόγησης και κατόπιν χρησιμοποιεί πράκτορες (μυρμήγκια) για την συνεχή αξιολόγηση των μονοπατιών. Ο αλγόριθμος ARA (Ant colony-based Routing Algorithm) [18] είναι ένας κατά απαίτηση (on demand) αλγόριθμος που αρκείται στην εύρεση του συντομότερου μονοπατιού ανάμεσα στους κόμβους όταν ζητηθεί χρησιμοποιώντας τα πακέτα δεδομένων ως πράκτορες για να ιχνηλατήσουν τα συντομότερα μονοπάτια.

Σε άλλες περιπτώσεις([19], [20]) χρησιμοποιούνται πράκτορες για να διατρέξουν το δίκτυο τυχαία, ιχνηλατώντας τα μονοπάτια που χρησιμοποίησαν από την πηγή προς τους κόμβους. Αλγόριθμοι υπολογισμού θέσης με τη βοήθεια του ACO έχουν επίσης αναφερθεί [21]όπου οι πράκτορες διαχέουν την πληροφορία δρομολόγησης που έχουν ανακτήσει. Ένας υβριδικός αλγόριθμος που συνδυάζει τον ACO με το πρωτόκολλο AODV [22] χρησιμοποιεί πράκτορες που διατρέχουν τυχαία το δίκτυο και κρατούν πληροφορίες για τους n κόμβους που επισκέφθηκαν τελευταία ενημερώνοντας τους πίνακες δρομολόγησης κατάλληλα.

Αρκετοί ακόμα αλγόριθμοι έχουν προταθεί για τη δρομολόγηση σε δίκτυα MANET. Όλοι αυτοί οι αλγόριθμοι παρεκκλίνουν από τις αρχές του ACO προσπαθώντας να αντιμετωπίσουν τις ιδιαιτερότητες αυτών των δικτύων, και οι περισσότεροι από αυτούς δεν διαφέρουν τελικά από κοινούς κατά απαίτηση αλγορίθμους δρομολόγησης. Θα πρέπει να αναφερθεί ακόμη ότι έχουν χρησιμοποιηθεί παραλλαγές του ACO για την αποτελεσματικότερη ενεργειακή διαχείριση των ασύρματων δικτύων [23] και οι οποίες αποσκοπούν σε αποτελεσματικούς αλγορίθμους δρομολόγησης για τη σωστή διαχείριση των ενεργειακών πόρων του δικτύου.

Ο αλγόριθμος PSO έχει βρει εκτεταμένες εφαρμογές σε προβλήματα βελτιστοποίησης και αυτο-οργάνωσης όσον αφορά δίκτυα MANET. Πιο συγκερκιμένα έχουν προταθεί μέθοδοι δρομολόγησης [24] αλλά χρησιμοποιώντας «bird-flocking> τεχνι-

κές [25], όπως επίσης και μέθοδοι αυτο-οργάνωσης τέτοιων δικτύων[26] εξαιτίας της συνεργατικής συμπεριφοράς του συγκεκριμένου αλγορίθμου. Μάλιστα ο βαθμός της αυτο-οργάνωσης που επιτεύχθηκε με την εφαρμογή του PSO δεν απέχει και πολύ από το βαθμό αυτο-οργάνωσης που θα είχε το ίδιο δίκτυο σε περίπτωση κεντρικής διαχείρισης. Ο αλγόριθμος PSO έχει βρει εφαρμογή και στην ενεργειακή διαχείριση των κόμβων ενός δικτύου MANET[27] με τη δημιουργία clusters για εξοικονόμηση ενέργειας όσον αφορά τη συλλογή δεδομένων από τους κόμβους-αισθητήρες.

Ένας καινούργιος τομέας στον οποίο βρίσκουν εφαρμογή οι αλγόριθμοι Νοημοσύνης Σμήνους είναι τα Σμήνη Ρομπότ (Swarm Robots). Η χρήση swarm robots για την διεκπεραίωση διαφόρων εργασιών παρουσιάζει πολλά πλεονεκτήματα όπως απλότητα κατασκευής ρομποτικών μονάδων, υψηλή πιστότητα (μικρή πιθανότητα λάθους) και μικρό κόστος. Ένα σμήνος ρομπότ μπορεί να πραγματοποιήσει πολύπλοκες εργασίες που είναι δύσκολες ακόμη και για τα παραδοσιακά ρομποτικά συστήματα. Ένα τέτοιο μοντέλο[28] έχει προταθεί για την ανακάλυψη και αποφυγή εμποδίων με τη βοήθεια του PSO. Άλλα μοντέλα swarm robots έχουν επίσης προταθεί για ανθεκτικότητα αποτυχίας (fault tolerance) [29], [30] και συνεργατική αναζήτηση [31].

Θα πρέπει να αναφερθεί τέλος και η χρησιμοποίηση του PSO σε σμήνη UAV (Unmanned Aerial Vehicles) και UUV (Unmanned Underwater Vehicles) δηλαδή μη επανδρωμένων εναέριων και υποβρύχιων οχημάτων που χρησιμοποιούνται κυρίως για στρατιωτικούς σκοπούς. Τρία τέτοια παραδείγματα είναι η εφαρμογή του PSO για της εύρεση βέλτιστης διαδρομής προς το στόχο σε τρεις διαστάσεις, ενός σμήνους από UAV έτσι ώστε να αποφεύγουν εμπόδια, εχθρικές απειλές με την ταυτόχρονη κατανάλωση ελάχιστων καυσίμων [32], της χρησιμοποίησης του PSO από ένα σμήνος UAV το οποίο μέσω της συνεργασίας και της αυτο-οργάνωσης να προστατεύει συγκεκριμένο χώρο από εχθρικές απειλές [33], όπως επίσης και της αυτο-οργάνωσης και πλοήγησης σμήνους UUV με τη βοήθεια μιας βελτιωμένης έκδοσης του αλγορίθμου PSO [34].

Άλλες τυπικές εφαρμογές είναι το covering (εξερεύνηση σε εχθρικό έδαφος), patrolling (φύλαξη μουσείων από κλοπές), environment manipulation(έλεγχος κυκλοφορίας), self-assembling(επαναρυθμιζόμενα ρομπότ) και localization(βελτίωση προσδιορισμού θέσης). Επιπλέον σημαντική ερευνητική δουλειά έχει επενδυθεί στο σχεδιασμό συστημάτων σμήνους για την εξερεύνηση περιοχών γνωστών ή άγνωστων.Στην πλειονότητα τους οι προηγούμενες ερευνητικές εργασίες σχετικές με συστήματα νοημοσύνης σμήνους θεωρούν τις πηγές πληροφορίας στατικές εκτός ελάχιστων περιπτώσεων όπου οι προσπάθειες αφορούν δυναμικά περιβάλλοντα.([35],[36], [37]). Η [38] αναφέρεται σε δυναμικά περιβάλλοντα αλλά δεν θίγει θέματα εγκυρότητας πλαισίων.

Οι εφαρμογές που μπορεί να βρει η νοημοσύνη σμήνους είναι σημαντικές και αφορούν πολλούς τομείς. Οι επιστήμονες δεν ξέρουν όλες τις λεπτομέρειες των αλληλεπιδράσεων μέσα στα σμήνη κοινωνικών εντόμων και χωρίς αυτές τις πληροφο-

ρίες δεν μπορούν να κατασκευάσουν λογισμικό που να τις προσομοιώνει. Επιπλέον, η νοημοσύνη σμήνους στερείται μιας γενικής θεώρησης των πραγμάτων και έτσι δεν μπορεί να εφαρμοστεί σε προβλήματα που απαιτούν βαθιά συλλογιστική.

Από την παραπάνω επισκόπηση γίνεται φανερό ότι δεν υπάρχει προηγούμενη εμπειρία στο θεματικό πεδίο το οποίο πραγματεύεται η παρούσα διπλωματική εργασία δηλαδή της δημιουργίας ενός μηχανισμού ανακάλυψης της περιρρέουσας κατάστασης σε ένα δίκτυο MANET με τη βοήθεια Swarm Intelligence αλγορίθμων.

3.4 Μηχανισμός Ανακάλυψης Πληροφορίας Πλαισίου

3.4.1 Κατανεμημένα συστήματα νοημοσύνης σμήνους (Swarm Intelligence (SI) distributed systems)

Η περιοχή των πολυ-πρακτορικών συστημάτων (multi-agent systems) κατανεμημένου υπολογισμού έχει αναπτυχθεί ραγδαία τα τελευταία χρόνια. Πολλές υπηρεσίες κινητού υπολογισμού χρησιμοποιούν τεχνικές με πολλαπλούς πράκτορες ή σμήνη. Η βασικη ιδέα που κρύβεται πίσω από τα πολυ-πρακτορικά συστήματα έγκειται στην αποτελεσματικότερη διεκπεραίωση των εργασιών μιας εφαρμογής κινητού υπολογισμού από πολλούς, απλούς στη δομή και αυτόνομους πράκτορες παρά από έναν μοναδικό και περίπλοκο στη λειτουργία και κατασκευή πράκτορα. Τέτοια συστήματα είναι περισσότερο προσαρμοστικά, επεκτάσιμα και ανθεκτικά από αυτά που στηρίζονται σε έναν και μοναδικό υψηλών δυνατότητων πράκτορα.

Ένα σύστημα νοημοσύνης σμήνους μπορεί να ορισθεί ως μια ομάδα αυτόνομων πρακτόρων (σωματιδίων) χωρίς κεντρική διαχείριση με απλή δομή και περιορισμένες υπολογιστικές δυνατότητες. Οι πολλαπλοί πράκτορες πρέπει να συνεργάζονται έξυπνα για την επίτευξη των στόχων του σμήνους.

Ερευνούμε για ένα μηχανισμό που θα εκμεταλλεύεται τη συνεργατική συμπεριφορά των πρακτόρων προκειμένου να αντιμετωπιστεί το πρόβλημα της ανακάλυψης πληροφορίας πλαισίου (Context Discovery Problem - CDP). Πιο συγκεκριμένα στο CDP ένας πράκτορας (π.χ. κινητός κόμβος) χρειάζεται να ανακαλύψει και να προσδιορίσει τη θέση της απαιτούμενης πληροφορίας πλαισίου (π.χ. περιβαλλοντικές παράμετροι όπως θερμοκρασία, υγρασία, καταστάσεις όπως ξέσπασμα πυρκαγιάς) προκειμένου να τροφοδοτήσει την υπηρεσία κινητού υπολογισμού με επίγνωση του περίγυρου (context-aware mobile application) όπως π.χ. ο έλεγχος μιας ομάδας ρομπότ.

Η νοημοσύνη σμήνους εισάγει μια δυναμική και καινοτόμο ιδέα για την κατασκευή κατανεμημένων συστημάτων όπου η συνολική λειτουργικότητά τους εξαρτάται από την αλληλεπίδραση των πρακτόρων μεταξύ τους αλλά και με το περιβάλλον τους. Αυτοί οι πράκτορες συντονίζονται χρησιμοποιώντας αποκεντρωμένο έλεγχο και αυτο-οργάνωση. Τα συστήματα νοημοσύνης σμήνους εμπεριέχουν από τη φύση τους παραλληλισμό στην εκτέλεση των εργασιών και παρουσιάζουν υψηλή ανθεκτικότητα και αξιοπιστία.

Τα κύρια χαρακτηριστικά ενός SI- κατανεμημένου συστήματος είναι:

- Δεν παρουσιάζουν ιεραρχική δομή εντολών και ελέγχου με αποτέλεσμα να μην υπάρχει συγκεκριμένο σημείο δυσλειτουργίας ή τρωτότητας. Οι πράκτορες είναι γενικά απλοί και το συνολικό σμήνος εκ φύσεως είναι ένα σύστημα ανθεκτικό στις αποτυχίες (failure tolerant) και αποτελείται από έναν αριθμό πανομοιότυπων μονάδων που λειτουργούν (διαισθάνονται την πληροφορία πλασίου από το περιβάλλον π.χ. αισθητήρες) και συνεργάζονται (ανταλλάσουν πλαίσια) παράλληλα. Αντιθέτως, σύνθετα συμβατικά κατανεμημένα συστήματα απαιτούν σημαντική προσπάθεια σχεδιασμού για την αποφυγή αποτυχιών.
- Η σημαντικότερη αρχή ενός SI-κατανεμημένου συστήματος είναι η απλότητα των πρακτόρων του (π.χ. ένα κινητό τηλεφωνο με αισθητήρες). Αυξάνοντας απλώς τον αριθμό των πρακτόρων που συμμετέχουν για την επίτευξη ενός στόχου δεν βελτιώνουμε απαραίτητα και την απόδοση του συστήματος (αποδοτικότητα και αξιοπιστία). Οι πράκτορες συνεργάζονται ανταλλάσοντας χρήσιμες πληροφορίες προκειμένου να ανακτήσουν το απαιτούμενο πλαίσιο.
- Σε ένα ολοκληρωτικά κατανεμημένο περιβάλλον οι πράκτορες συνεργάζονται για την ανακάλυψη πλαισίων συγκεκριμένης εγκυρότητας (σχετιζόμενης με χρονικούς ή χωρικούς περιορισμούς). Το πλαίσιο περιοδικά απαρχαιώνεται και απαιτείται η επικαιροποίησή του ανά τακτά χρονικά διαστήματα με τη διαδικασία της ανακάλυψης και της ανάκτησης. Επιπλέον οι πόροι των πρακτόρων είναι περιορισμένοι ως προς τη μνήμη - υπάρχει περιορισμένο ιστορικό καταγραφής, ως προς τις δυνατότητες αίσθησης (sensing capabilities) - για πράκτορες που κινούνται συνεχώς η ακτίνα δράσης τους μπορεί να είναι μικρή σχετικά με την περιοχή κάλυψης και έτσι να χάνονται χρήσιμες πληροφορίες ακόμη και από γειτονικούς κόμβους, και τέλος ως προς τους επικοινωνιακούς πόρους - αυτοί οι πόροι χρησιμοποιούνται αποκλειστικά για τη μεταβίβαση πληροφορίας στο σμήνος.

3.4.2 Επιλογή μηχανισμού

Η προσπάθεια για ανακάλυψη έγκυρων πλαισίων πληροφορίας σε ένα περιβάλλον κινητού υπολογισμού προσομοιάζει κατά ένα μεγάλο ποσοστό στην προσπάθεια των έμβιων όντων που προαναφέρθηκαν(έντομα, πουλιά, ψάρια) να ανακαλύψουν την τροφή τους. Επομένως η επιλογή ενός από τους δύο αλγορίθμους που περιγράφηκαν μπορεί να αποτελέσει τη βάση για να θεμελιωθεί ένας τέτοιος μηχανισμός. Στην περίπτωση που οι πηγές δεν κινούνται και έχουμε μια πλήρως επιβλεπόμενη κατάσταση όλων των κόμβων τότε ο μηχανισμός ανακάλυψης πλαισίου ομοιάζει κατά πολύ με τους αλγορίθμους Ant Colony Optimization. Παρόλα αυτά όμως υπάρχουν βασικές διαφορές με τους αλγορίθμους αυτούς (όπως αναφέρονται και παρακάτω).Στο πρόβλημα της ανακάλυψης πληροφορίας πλαισίου πρέπει να ληφθούν υπόψιν τα ακόλουθα:

1. δεν υφίσταται η έννοια του πλήρους ελέγχου όλων των κόμβων, κάτι που ισχύει σε περιβάλλοντα κατανεμημένου ελέγχου και διάχυτου υπολογισμού,
2. η πληροφορία που αναζητείται δεν μπορεί να εκτιμηθεί καθολικά εξίσου από όλους τους κόμβους, κάτι που ισχύει σε ένα σύστημα κατά το οποίο ο κάθε κόμβος αυτόνομα εκτιμά και θέτει τις δικές του προτιμήσεις και περιορισμούς εγκυρότητας πλαισίου,
3. η εκτίμηση για την πληροφορία πλαισίου μεταβάλλεται χρονικά εφόσον οι συνιστώσες πλαισίου μεταβάλλονται επίσης,
4. οι πηγές δεν είναι πάντα ακίνητες, άρα μπορεί να ανακαλυφθεί μόνον μια προσωρινά βέλτιστη λύση, κάτι που υφίσταται σε δυναμικά περιβάλλοντα στα οποία η κινητικότητα αποτελεί αναπόσπαστο μέρος της δυναμικής συμπεριφοράς των περιβαλλόντων αυτών, και,
5. οι κόμβοι δεν μπορούν να έχουν πάντα πλήρη συνεργατική συμπεριφορά. Ενδέχεται να υπάρξει μια ημι-συνεργατική συμπεριφορά.

Με βάση τα πιο πάνω χαρακτηριστικά ο καταλληλότερος αλγόριθμος για το μηχανισμό ανακάλυψης πληροφορίας πλαισίου που θέλουμε να δημιουργήσουμε είναι ο PSO.

4 Μοντέλο προσομοίωσης για την Ανακάλυψη Πληροφορίας Πλαισίου

4.1 Δομή Μοντέλου

Η πλατφόρμα προσομοίωσης που χρησιμοποιήθηκε είναι το J-sim [39] με την επέκταση του ασύρματου πακέτου (wireless package) για την προσομοίωση δικτύων MANET. Προσομοιώθηκαν δύο ειδών κόμβοι, οι κόμβοι-πηγές και οι κόμβοι-οπαδοί. Σχηματικά διαγράμματα για τα δύο είδη κόμβων φαίνονται στο Σχήμα 4.

Η δομή των πηγών και των οπαδών είναι παρόμοια και αναφέρεται στη χρήση έτοιμων συστατικών του πακέτου για την προσομοίωση των διαφόρων επιπέδων κατά OSI, του ελέγχου της κίνησης, της γειτονίας των κόμβων αλλά και του μέσου διάδοσης (ασύρματο κανάλι). Χρησιμοποιήθηκε επίσης και το πρωτόκολλο AODV (Ad-hoc On-Demand Distance-Vector) κατάλληλου για τη δρομολόγηση ανάμεσα στους κόμβους που χάνουν τις συνδέσεις μεταξύ τους λόγω της κίνησης όταν βρεθούν εκτός εμβέλειας. Όσον αφορά το μοντέλο κίνησης, για τις πηγές χρησιμοποιήθηκε το RandomWay που εμπεριέχεται στο πακέτο, ενώ για την περίπτωση των οπαδών υιοθετήθηκε ένα καινούργιο μοντέλο κίνησης βασισμένο στον αλγόριθμο Particle Swarm Optimization (PSO) ο οποίος περιγράφεται αναλυτικά σε αυτή την ενότητα και ενσωματώθηκε στο πακέτο J-sim μετά την υλοποίηση του.

Αναλυτικά η δομή του μοντέλου φαίνεται παρακάτω:

Trace file – Packet Generator

Το συγκεκριμένο συστατικό διαβάζει μια χρονοακολουθία από ένα αρχείο (trace file) και παράγει χρονοσφραγισμένα πακέτα στις συγκεκριμένες χρονικές στιγμές της προσομοίωσης. Για κάθε κόμβο-πηγή χρησιμοποιούνται διαφορετικές χρονοακολουθίες (αρχεία) για να αποφευχθεί κάποια πιθανή προκατειλημμένη (biased) προσκόλληση οπαδών σε συγκεκριμένες πηγές κατά τη διαδικασία ανακάλυψης πληροφορίας και σύγκλισης του PSO αλγορίθμου.

TCP

Υλοποιεί το Transport Protocol και εγκαθιστά μια TCP σύνδεση μεταξύ δύο κόμβων, χωρίς άνοιγμα και κλείσιμο της σύνδεσης όπως επίσης και του 3-way handshaking, τα οποία βέβαια δεν ενδιαφέρουν στη συγκεκριμένη προσομοίωση. Για N πηγές και M οπαδούς δημιουργούνται από το μοντέλο N x M TCP συνδέσεις.

Pkt Dispatcher

Υλοποιεί το IP layer.

LL

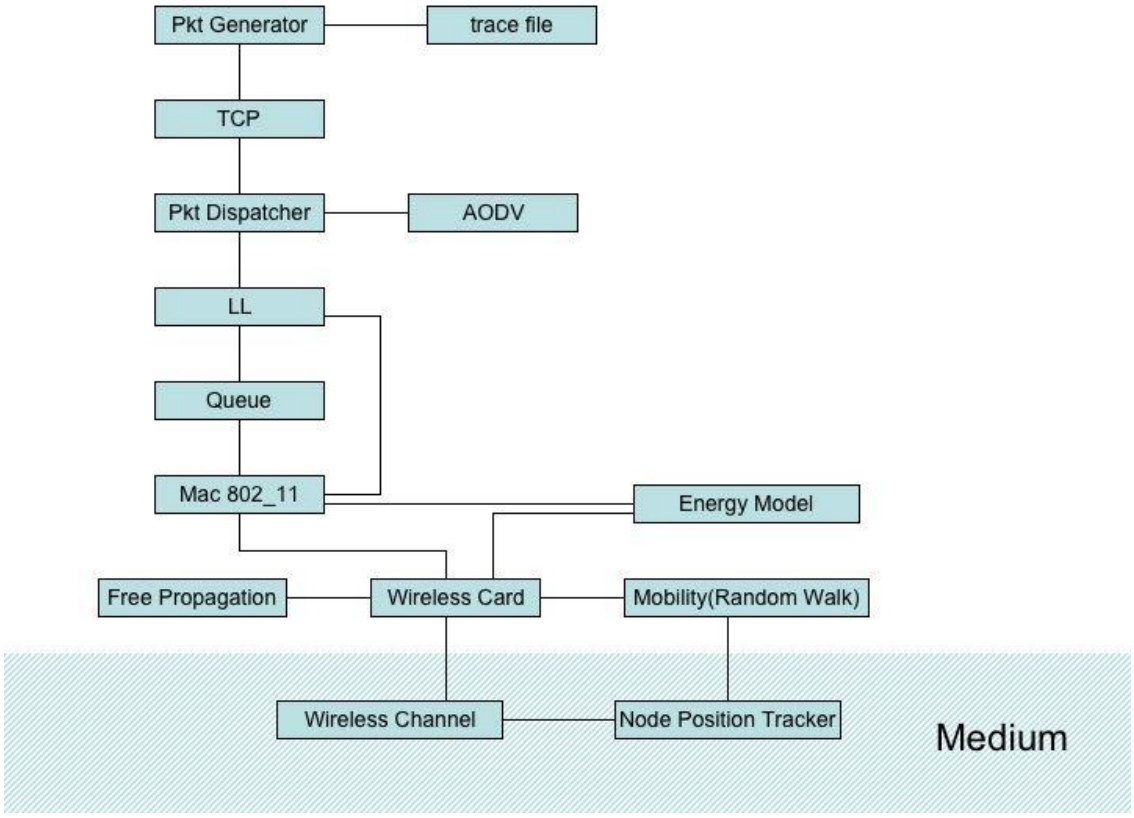
Υλοποιεί ένα μέρος του Link layer.

Queue

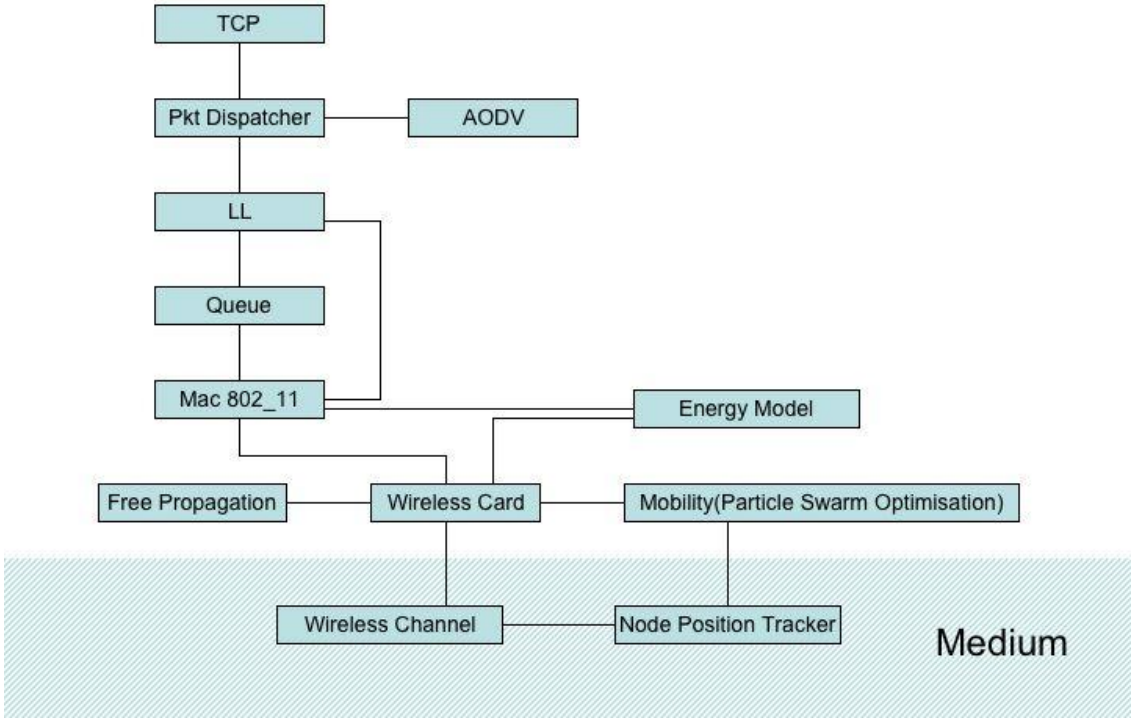
Υλοποιεί την ουρά της ασύρματης κάρτας όπου γίνονται buffered τα πακέτα.

AODV

Υλοποιεί το Ad hoc on Demand Distance Vector πρωτόκολλο δρομολόγησης που



i)



ii)

Σχήμα 4: Δομή Κόμβων - i) Πηγές ii) Οπαδοί

έχει σχεδιαστεί ειδικά για δίκτυα ασύρματων κόμβων. Υπάρχουν δύο φάσεις του πρωτοκόλλου, η φάση της ανακάλυψης διαδρομής (route discovery phase) όπου ένας κόμβος στέλνει RREQ πακέτα και αναμένει απάντηση από προσκείμενους κόμβους προκειμένου να δημιουργηθεί σύνδεση και η φάση της διατήρησης διαδρομής (route maintenance phase) όπου αφαιρούνται χαλασμένες συνδέσεις (broken links) από τα routing tables των κόμβων.

Mac 802_11

Υλοποιεί το Mac πρωτόκολλο και παραλαμβάνει πακέτα από την ουρά προκειμένου να σταλούν μέσω της κάρτας στο κανάλι, όπως επίσης και παραλαμβάνει πακέτα από το κανάλι για να τα προωθήσει στα υψηλότερα επίπεδα. Επίσης ανιχνεύει τα broken links λόγω της κίνησής των κόμβων και ειδοποιεί το AODV συστατικό για τις περαιτέρω ενέργειες. Σε αυτό το συστατικό υλοποιείται επίσης και η λειτουργία εξοικονόμησης ενέργειας (Power Saving Mode) και η οποία είναι ενεργοποιημένη στο παρόν μοντέλο. Το PSM λειτουργεί με τη βοήθεια πλαισίων beacons τα οποία εκπέμπονται στην αρχή κάποιων ομοιόμορφων χρονικών διαστημάτων (beacon intervals) και σε περίπτωση που ο κόμβος δεν πάρει απόκριση για διάστημα ίσο με το ATIM window size μπαίνει σε save mode (κοιμάται) για το υπόλοιπο του beacon interval, αφού beacon interval > ATIM.

Wireless Card

Αποτελεί την σύνδεση του κόμβου με το ασύρματο κανάλι και υλοποιεί τις λειτουργίες της ασύρματης κάρτας του κόμβου όπως την ισχύ του σήματος του πλαισίου που παραλαμβάνεται για να υπολογισθεί η εγκυρότητα του.

Energy Model

Το ενεργειακό μοντέλο είναι εγκατεστημένο στο συστατικό της ασύρματης κάρτας και παρακολουθεί την ενεργειακή κατάσταση του κόμβου. Το συγκεκριμένο ενεργειακό μοντέλο παίρνει υπόψιν μόνον τα ποσά της ενέργειας που καταναλώνονται λόγω εκπομπής και λήψης του κόμβου αλλά όχι και την κατανάλωση ενέργειας εξαιτίας της κίνησης του κόμβου.

Στο Παράρτημα Β παρατίθενται οι μεταβολές στον κώδικα του ενεργειακού μοντέλου (EnergyModel.java) και της ασύρματης κάρτας (WirelessPhy.java) προκειμένου να λαμβάνεται υπόψιν και η κατανάλωση ενέργειας λόγω της κίνησης. Αυτό έγινε για να υπάρξει νόημα στην εφαρμογή των πολιτικών ανακάλυψης πληροφορίας, έτσι ώστε ένας κόμβος με χαμηλό απόθεμα ενέργειας και απαρχαιωμένο πλαίσιο να παραμένει ακίνητος προκειμένου να μην καταναλώνει ενέργεια, ευελπιστώντας στην εμφάνιση κάποιου κόμβου με έγκυρο πλαίσιο στη γειτονιά του για να το ανακτήσει.

Free propagation

Υλοποιεί το Free Space Model για τη μετάδοση των ραδιοκυμάτων. Επιλέχθηκε γιατί είναι το απλούστερο μοντέλο και δεν επηρεάζει τη συγκεκριμένη προσομοίωση.

Mobility Model

Το συγκεκριμένο συστατικό είναι υπεύθυνο για την κίνηση των κόμβων. Υπάρχουν

δύο ειδών μοντέλα, το Random Walk όπου οι κόμβοι μετακινούνται σε τυχαίες θέσεις που παράγει το μοντέλο και το Trajectory based όπου οι κόμβοι κινούνται βάση ενός προκαθορισμένου αρχείου συντεταγμένων. Για την κίνηση των κόμβων-πηγών επιλέχθηκε το Random Walk για να υπάρχει τυχαιότητα στην κίνηση και ομοιόμορφη κάλυψη στην εκπομπή πληροφορίας στο πλέγμα της προσομοίωσης. Σε αυτό συμβάλλει και το γεγονός ότι έχουν επιλεχθεί και τυχαίες αρχικές θέσεις εκκίνησης για τους κόμβους.

Advanced Mobility Model

Όσον αφορά τους κόμβους-οπαδούς, υλοποιήθηκε μια επέκταση του προηγούμενου μοντέλου με την ονομασία AdvancedMobilityModel και το οποίο περιλαμβάνει την εφαρμογή του αλγορίθμου PSO για την πλοήγηση των οπαδών. Στο παράρτημα Β φαίνεται ο κώδικας για το συγκεκριμένο συστατικό.

Wireless Channel

Αυτό το συστατικό προσομοιώνει το διαμοιραζόμενο ασύρματο κανάλι. Όλοι οι κόμβοι πρέπει να συνδεθούν στο κανάλι προκειμένου να στέλνουν και να λαμβάνουν πακέτα. Επίσης λαμβάνει αναφορά από τον Node Position Tracker για τη γειτονιά του κάθε κόμβου δηλαδή όλων εκείνων των κόμβων που μπορούν να ακούσουν το ασύρματο interface του κόμβου. Η γειτονιά ενός κόμβου καθορίζεται σε μονάδες πλέγματος.

Node Position Tracker

Αποτελεί τον ελεγκτή της θέσης του κάθε κόμβου καθώς αυτός κινείται στο πλέγμα και επομένως είναι ενήμερος για τη θέση του οποιουδήποτε κόμβου σε οποιαδήποτε χρονική στιγμή. Διαιρεί την περιοχή προσομοίωσης σε μικρά ορθογώνια και λαμβάνει συνεχείς ενημερώσεις από το Mobility Model σχετικά με τις θέσεις των κόμβων στο πλέγμα. Αναφέρει τους κόμβους που γειτονεύουν με ένα συγκεκριμένο κόμβο βάσει της τιμής που καθορίζεται από το Wireless Channel.

Το μοντέλο που έχει αναπτυχθεί για την προσομοίωση είναι πλήρως παραμετροποιημένο όπως φαίνεται στο Παράρτημα Α. Οι παράμετροι του μοντέλου είναι:

- αριθμός πηγών
- αριθμός οπαδών
- συντεταγμένες περιοχής προσομοίωσης
- μέγεθος στοιχείου πλέγματος
- παράμετρος γειτονίας κόμβων
- ενεργειακές σταθερές που καθορίζουν την κατανάλωση για εκπομπή, λήψη και κίνηση κόμβων
- beacon interval και ATIM window size
- ταχύτητα κόμβων
- εμβέλεια ασύρματων καρτών
- seed values για τις θέσεις εκκίνησης των κόμβων.

Στο Παράρτημα Β δίνονται τα lisitngs των συστατικών του πακέτου J-sim τα οποία έχουν τροποποιηθεί για τις ανάγκες της παρούσας προσομοίωσης. Συνοπτικά αναφέρονται παρακάτω οι κυριότερες προσθήκες/μετατροπές του κάθε συστατικού.

AdvancedMobilityModel.java - Προσθήκη 6 νέων ports για την αποθήκευση των αποστάσεων των οπαδών από τις πηγές (outPort), της κατάστασης των οπαδών (statusPort), της on-line απεικόνισης των τροχιών τους (plotPort), της ενημέρωσης του ενεργειακού μοντέλου για την κίνηση των οπαδών (movhisPort), της επεξεργασίας της γειτονιάς των κόμβων (trackPort) και τέλος της λήψης των θέσεων των κόμβων (valndxPort).
Προσθήκη μεθόδων για την επεξεργασία των πλαισίων (dataArriveAtUpPort(), context_validity()), της συνεργατικής συμπεριφοράς των κόμβων με την εύρεση γειτονιάς και την απόκτηση αντικειμενικής και υποκειμενικής γνώσης (Process Neighbors()), της ενημέρωσης του διανύσματος κίνησης (updatePSO Position(), setPSODestination()), της ενημέρωσης της κατάστασης των κόμβων (setNode Status()), του υπολογισμού της απόστασης οπαδών-πηγών (Source FollowerDistance()) και της πολιτικής ανακάλυψης(UtilityFunction()).
Προσαρμογές έχουν γίνει επίσης και στις μεθόδους χρονισμού του συστατικού (timeout και reporting).

WirelessPhy.java Προσθήκη ενός νέου port (movhisPort) για την ενημέρωση της κίνησης των κόμβων.
Προσθήκη της μεθόδου calc_energy_mov() η οποία υπολογίζει το ποσοστό της κατανάλωσης ενέργειας εξαιτίας της κίνησης.

EnergyModel.java Προσθήκη της μεθόδου updateMovEnergy που ενημερώνει το ενεργειακό μοντέλο για την κατανάλωση ενέργειας λόγω της κίνησης.

NeighborQueryContract.java Αναβάθμιση στη μορφή του μηνύματος που στέλνει το συστατικό NodePositionTracker στους κόμβους, με τη δημιουργία ενός καινούργιου constructor έτσι ώστε να αποστέλλονται οι θέσεις των γειτονικών κόμβων και τα πλαίσια που κατέχουν, εκτός από τη λίστα των γειτονικών κόμβων.

NodePositionTracker.java Προσθήκη ενός νέου port (valndxPort) για την αποστολή των θέσεων των κόμβων.
Προσθήκη και της αντίστοιχης μεθόδου processValIndex() που επεξεργάζεται τα δεδομένα που διακινούνται από το συγκεκριμένο port.
Το συστατικό NodePositionTracker όπως έχει αναφερθεί και νωρίτερα σε αυτήν την ενότητα, ενημερώνει οποιονδήποτε κόμβο σε οποιαδήποτε χρονική στιγμή σχετικά με το ποιοί είναι οι γειτονικοί του κόμβοι δηλαδή εκείνοι οι κόμβοι που βρίσκονται εντός της περιοχής γύρω από τον κόμβο που ορίζει η παράμετρος του μεγέθους γειτονίας. Οι μεταβολές που έχουν γίνει ουσιαστικά αναβαθμίζουν το συγκεκριμένο συστατικό προκειμένου να ενημερώνει επιπλέον για τις θέσεις των κόμβων όπως επίσης και για τα πλαίσια που κατέχουν

σε κάθε χρονική στιγμή.

PositionReportContract.java Αναβάθμιση της μορφής του μηνύματος που χρησιμοποιείται από τους κόμβους για την αναφορά της θέσης τους στο συστατικό NodePositionTracker, με τη δημιουργία ενός καινούργιου constructor έτσι ώστε να αναφέρονται και τα πλαίσια που κατέχουν σε κάθε χρονική στιγμή.

4.2 Υλοποίηση αλγορίθμου PSO

Έχοντας σαν βάση τη γενική αρχή του αλγορίθμου PSO που περιγράφηκε στην παράγραφο §3.2.2 προσπαθούμε να δημιουργήσουμε έναν μηχανισμό ανακάλυψης πλαισίου σε περιβάλλοντα διάχυτου υπολογισμού, όπως δίκτυα ασύρματων αισθητήρων, επεκτείνοντας τη λειτουργία του στο πεδίο της χρονικής εγκυρότητας της εκτίμησης του πλαισίου. Σύμφωνα με την θεώρηση αυτή, οι οπαδοί και οι πηγές αποτελούν κινούμενους «οργανισμούς» (particles or agents) και κινούμενη τροφή, αντίστοιχα, σε μια «αγέλη» (swarm) M+N οργανισμών. Στόχος είναι: «οι κινούμενοι οργανισμοί να εντοπίσουν τις πηγές τροφής. Ο προσδιορισμός της τροφής επιτυγχάνεται με συνεχόμενες μεταβολές του διανύσματος της κατεύθυνσης κίνησης (διεύθυνση, φορά και μέτρο ταχύτητας) ενός οργανισμού προκειμένου να τείνει σε εκείνον τον οργανισμό που είναι πιο κοντά (π.χ., χωρικά) στην πηγή τροφής».

Ο PSO αλγόριθμος είναι επαναληπτικός και, σε κάθε βήμα του, εντοπίζεται η μέχρις τότε πλησιέστερη χωρική θέση της πηγής. Έτσι, η επόμενη μεταβολή του διανύσματος κίνησης ενός οργανισμού / οπαδού βασίζεται όχι μόνον στην τρέχουσα εκτίμηση θέσης, που είναι καθορισμένη από τον κάθε οργανισμό αυτόνομα, αλλά και από την εκτίμηση που προκύπτει από την (χωρική) γειτονία του κάθε οργανισμού. Η πρώτη εκτίμηση αποτελεί τη τρέχουσα «τοπικά» καλύτερη εκτίμηση που υπολογίζεται από έναν οπαδό (υποκειμενική εκτίμηση ότι πλησιάζει την πηγή). Η δεύτερη εκτίμηση υπολογίζεται από τις υποκειμενικές εκτιμήσεις των γειτονικών οπαδών (αντικειμενική εκτίμηση της γειτονιάς ενός οπαδού). Ο συνδυασμός της πρώτης και της δεύτερης εκτίμησης συνεισφέρουν στην συνολική εκτίμηση και προσδιορισμό του διανύσματος κίνησης του οπαδού.

Συγκεκριμένα, το βάρος της απόφασης για την μελλοντική κατεύθυνση του κόμβου βασίζεται τόσο στην προσωρινή υποκειμενική όσο και στην προσωρινή αντικειμενική εκτίμηση. Προσδιορίζονται έτσι δύο βάρη σημαντικότητας όπου καθορίζουν το ποσοστό μεταβολής του διανύσματος κίνησης.

Το υποκειμενικό βάρος, που αναφέρεται στην βιβλιογραφία ως ο παράγοντας «νόησης» (cognitive factor). Το βάρος αυτό θέτει την σημαντικότητα της υποκειμενικής εκτίμησης για την μελλοντική απόφαση του οπαδού ως προς τα που θα κινηθεί. Στην περίπτωση της ανακάλυψης πλαισίου, η υποκειμενική εκτίμηση είναι μεταβαλλόμενη εφόσον εξαρτάται από την κίνηση του οπαδού, την κίνηση της πηγής, την κίνηση των γειτονικών κόμβων του οπαδού και την εγκυρότητα εκτίμησης του πλαισίου.

Το αντικειμενικό βάρος, που αναφέρεται στην βιβλιογραφία ως ο «κοινωνικός» παράγοντας (social factor). Το βάρος αυτό θέτει την σημαντικότητα της εκτίμησης των γειτονικών οπαδών για την μελλοντική απόφαση. Στην περίπτωση της ανακάλυψης πλαισίου, η αντικειμενική εκτίμηση είναι μεταβαλλόμενη εφόσον εξαρτάται από την κίνηση των γειτονικών κόμβων του οπαδού και την εγκυρότητα εκτίμησης του πλαισίου, που καθορίζεται από τους γειτονικούς κόμβους του οπαδού, όπως θα αναφερθεί παρακάτω. Τα δύο αυτά βάρη και σε συνδυασμό με την τρέχουσα κατεύθυνση του ορίζουν την μελλοντική κατεύθυνση του οπαδού στον χώρο. Επίσης, ο μηχανισμός ανακάλυψης μελετά και το κομμάτι της βελτιστοποίησης της αναζήτησης του πλαισίου βάσει περιορισμών όπως ενέργειας επικοινωνίας μεταξύ κόμβων και ενέργειας διατήρησης των οπαδών κόμβων προς τις τρέχουσες πηγές πληροφορίας πλαισίου.

Ορίζεται [40] μια γραμμική συνάρτηση απόστασης χρόνου U(t) η οποία απαρχαιώνεται γραμμικά με το χρόνο και ονομάζεται βαθμός εγκυρότητας πλαισίου

$$U(t) = e^{(t-\theta_{im})} \tag{3}$$

όπου θ_{im} το χρονικό κατώφλι πέρα από το οποίο μια τιμή που μετρήθηκε τη χρονική στιγμή $t - \theta_{im}$ θεωρείται απαρχαιωμένη. Το συγκεκριμένο κατώφλι ορίζεται για ένα συγκεκριμένο βαθμό εγκυρότητας η που ονομάζεται κατώφλι εγκυρότητας, δηλ. μια τιμή θεωρείται έγκυρη για $U \leq$ η όπου το θ_{im} ορίζεται για $U = \eta$.

Ορίζεται ως δείκτης εγκυρότητας πλαισίου x ενός κόμβου i, $g_{ix}(k)$, την χρονική στιγμή k ο μέγιστος βαθμός εγκυρότητας όλων των συνιστωσών του x με κατώφλι εγκυρότητας, η_{ix}, ίσο με το ελάχιστο κατώφλι εγκυρότητας των συνιστωσών x [40].

Χωρίς βλάβη της γενικότητας και για λόγους απλότητας, στις προσομοιώσεις μας, θεωρούμε ότι κάθε πλαίσιο αποτελείται από μια συνιστώσα, αυτή της χρονικής αξιοπιστίας και επομένως ο βαθμός εγκυρότητας και ο δείκτης εγκυρότητας ταυτίζονται και μπορούν να χρησιμοποιηθούν ισοδύναμα. Επίσης θεωρούμε ότι όλοι οι κόμβοι θέτουν το ίδιο κατώφλι εγκυρότητας στην εκτίμηση της αξιοπιστίας των πλαισίων. Η συνάρτηση του δείκτη ή βαθμού εγκυρότητας αποτελεί τη συνάρτηση καταλληλότητας που πρέπει να μεγιστοποιηθεί (μεγάλη ποιότητα πλαισίων) προκειμένου να προκύψουν οι λύσεις που αναζητούνται δηλ. οι περιοχές με φρέσκα πλαίσια σύμφωνα με όσα έχουν συζητηθεί στην παράγραφο §3.2.2.

Σε οποιαδήποτε χρονική στιγμή της προσομοίωσης κάθε κόμβος συλλέγει από τους γείτονες του τα πλαίσια που κατέχουν και αποκομίζει γνώση για το καλύτερο υποκειμενικά πλαίσιο που κατέχει κάποιος γείτονας, αποθηκεύοντας παράλληλα και τις χωρικές πληροφορίες για την καλύτερα υποκειμενικά γειτονιά που αντιστοιχούν στον γείτονα με το καλύτερο πλαίσιο ως εκείνη τη στιγμή. Επιπροσθέτως συλλέγει πληροφορίες για τη μέση τιμή γης εγκυρότητας των πλαισίων από τη γειτονιά του, αποκομίζοντας αντικειμενική εκτίμηση για την καλύτερη γειτονιά και αποθηκεύει τη θέση του, ως θέση καλύτερη αντικειμενικής εκτίμησης ως εκείνη τη στιγμή. Η μετακίνηση του στην επόμενη χρονική στιγμή θα γίνει προς την κατεύθυνση της καλύτε-

ρα υποκειμενικής και καλύτερα αντικειμενικής γνώσης, αποδίδοντας τα κατάλληλα βάρη για τα δύο είδη γνώσης.

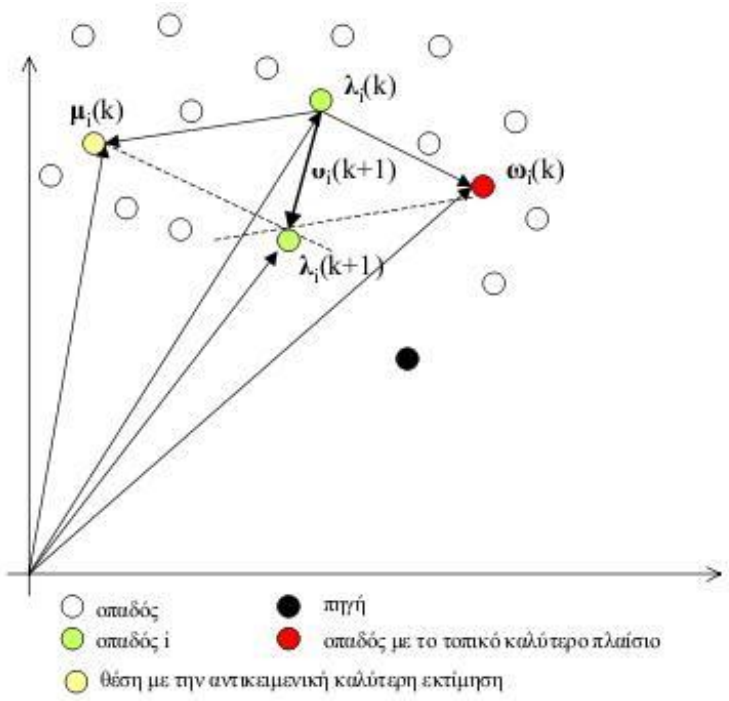
Στο Σχήμα 5 δίνεται παραστατικά ο τρόπος πλοήγησης ενός οπαδού βάσει των διανυσμάτων θέσης που αντιστοιχούν στην υποκειμενική και αντικειμενική γνώση.

Οι καταστάσεις στις οποίες μπορεί να περιέλθει ένας οπαδός κατά τη διαδικασία ανακάλυψης πλαισίου είναι οι ακόλουθες:

Έωλος – Περιέχει απαρχαιωμένο πλαίσιο ($g_i > \eta_i$)

Συμβιβάσιμος – Υπάρχει κάποιος γειτονικός οπαδός που έχει καλύτερο πλαίσιο από αυτόν αλλά όχι αυτό που πραγματικά ζητά ($\eta_i < g_i < g_n$)

Ικανοποιημένος – Έχει βρει το κατάλληλο για αυτόν πλαίσιο ($g_i < \eta_i < g_n$)



Σχήμα 5: Τρόπος πλοήγησης ενός οπαδού i βάσει των διανυσμάτων μ_i, λ_i και ω_i

Αναλυτικά τα βήματα του αλγορίθμου είναι τα ακόλουθα:

- Αρχικοποίηση των θέσεων και των ταχυτήτων των πηγών και των οπαδών στο πλέγμα προσομοίωσης. Ορισμός των σταθερών c_1 και c_2 για τα βάρη πλοήγησης της υποκειμενικής και της αντικειμενικής γνώσης αντίστοιχα. Είναι $c_1, c_2 \in (0, 1)$
- Καλύτερη αντικειμενική θέση (μ) = Καλύτερη υποκειμενική θέση (ω) = Αρχική θέση κόμβου (λ), Κατάσταση κόμβου (K) $\rightarrow$ Έωλος

- ⋆ Συλλογή από τους γείτονες των πλαισίων τους και υπολογισμός της αντικειμενικής εγκυρότητας $g_{OBS} = g_{mean}(0)$ για τη χρονική στιγμή 0, όπου το g_{mean} είναι ο μέσος όρος της εγκυρότητας των πλαισίων των γειτόνων
- ⋆ Υπολογισμός της υποκειμενικής εγκυρότητας $g_{SBS} = min(g_n(0)), n = 1 \dots j$ όπου j ο αριθμός των γειτόνων.

Επαναληπτικός βρόγχος για όλους τους οπαδούς μέχρι να εκπληρωθούν συγκεκριμένες συνθήκες (π.χ. ενεργειακές σε συνδυασμό με την κατάσταση του οπαδού). Εδώ εφαρμόζεται το Utility function που θα αναφερθεί στην παράγραφο §4.3 Τα παρακάτω βήματα για λόγους απλότητας αναφέρονται σε έναν κόμβο τη χρονική στιγμή t.

- ⋆ Δημιουργία τυχαίων διανυσμάτων βαρών θέσης $r_1, r_2 \in (0, 1)$
- ⋆ Ενημέρωση του διανύσματος κίνησης $\upsilon(t) = \upsilon(t-1) + c_1 r_1(\omega - \lambda(t)) + c_2 r_2(\mu - \lambda(t))$
- ⋆ Ενημέρωση του διανύσματος θέσης του κόμβου $\lambda(t) = \lambda(t-1) + \upsilon(t)$
- ⋆ Ενημέρωση του αντικειμενικού παράγοντα εάν $g_{OBS} > g_{mean}$ τότε $g_{OBS} = g_{mean}$ και $\mu = \lambda((t)$
- ⋆ Ενημέρωση του υποκειμενικού παράγοντα εάν $g_{SBS} > min(g_n(0)), n = 1 \dots j$ τότε $g_{SBS} = min(g_n(0)), n = 1 \dots j$ και $\omega = \lambda_n(t)$
- ⋆ Εάν $g >$ η τότε K ← έωλος
- ⋆ Εάν η $< g < g_{SBS}$ τότε K ← συμβιβάσιμος
- ⋆ Εάν $g <$ η $< g_{SBS}$ τότε K ← ικανοποιημένος
- ⋆ Ενημέρωση g, g_{SBS}, g_{OBS}

Θα πρέπει να αναφερθεί ότι αν οι κόμβοι-οπαδοί δεν βρίσκονται εντός εμβέλειας των πηγών προκειμένου να λάβουν φρέσκα πλαίσια από αυτές κατά τη διάρκεια της προσομοίωσης (απαρχαιωμένα πλαίσια) ή στην αρχή της (δεν έχει κανένα ληφθεί πλαίσιο), εφαρμόζεται το μοντέλο τυχαίας κίνησης των κόμβων-οπαδών (Random Way Point) προκειμένου να "βρεθούν" πηγές και οπαδοί μεταξύ τους για να ξεκινήσει η λήψη πλαισίων και έτσι να αναλάβει την πλοήγηση των οπαδών ο PSO προκειμένου αυτοί να οδηγηθούν σε περιοχές με φρέσκα πλαίσια.

Τέλος θα πρέπει να γίνει σαφές ότι το διάνυσμα θέσης ω_i όπου αναφέρεται στην τρέχουσα υποκειμενική εκτίμηση του οπαδού για το πλαίσιο και το διάνυσμα θέσης μ_i δεν είναι πάντα τα ίδια. Για παράδειγμα, έστω ότι σε μια γειτονιά με 3 οπαδούς, ο οπαδός 1 περιέχει φρέσκο πλαίσιο, δηλαδή, $g_1 x(k) < \eta_x$, και οι υπόλοιποι 2 οπαδοί απαρχαιωμένο, δηλαδή, $g_2 x(k)$, $g_3 x(k)$, $>$ α $\cdot\, \eta_x$ για κάποιο α > 1. Τότε, η γειτονιά αυτή ναι μεν περιέχει τον οπαδό 1 με το φρέσκο πλαίσιο, αλλά η συνολική εκτίμηση της γειτονιάς $g_1 x(k) + g_2 x(k) + g_3 x(k) \gg \eta_x$ (για α πολύ μεγάλο) αναφέρεται σε απαρχαιωμένο πλαίσιο. Φανταστείτε μια γειτονιά όπου οι τρεις οπαδοί έχουν περίπου φρέσκο πλαίσιο, δηλαδή, $g_1 x(k) = g_2 x(k) = g_3 x(k) \approx \eta_x$ Συνολικά η γειτονιά

αυτή είναι καλύτερη από την προαναφερθείσα γειτονιά, εφόσον,$g_1x(k) + g_2x(k) + g_3x(k) \approx \eta_x$. Αλλά, όμως, η πρώτη γειτονιά περιέχει τον οπαδό με το φρέσκο πλαίσιο. Έτσι, ο οπαδός i θα πρέπει να λάβει υπό όψιν του ότι μια μετακίνηση προς την πρώτη γειτονιά μπορεί να οδηγήσει σε θέσεις όπου δεν υπάρχουν οπαδοί με φρέσκο πλαίσιο εκτός από το τρέχοντα οπαδό που περιέχει το φρέσκο. Από την άλλη, μια μετακίνηση στην δεύτερη γειτονιά θα αποσκοπεί σε μια συνολικά καλύτερη γειτονιά αλλά, όμως, δεν θα ανακτήσει φρέσκια πληροφορία από κάποιον οπαδό. Συνεπώς, η αντικειμενική μετρική g_{OBS} που αναφέρεται στο διάνυσμα μ_i παίζει σημαντικό ρόλο για την πλοήγηση του οπαδού στην ανακάλυψη πλαισίου.

4.3 Πολιτική Ανακάλυψης Πλαισίου

Η κατάσταση ενός κόμβου-οπαδού όσον αφορά τις περαιτέρω ενέργειες του στο κυνήγι έγκυρου πλαισίου, χαρακτηρίζεται κυρίως από δύο παραμέτρους, το ενεργειακό απόθεμα και το δείκτη εγκυρότητας πλαισίου. Το ενεργειακό απόθεμα σε κάθε χρονική στιγμή μπορεί να εκφραστεί ως

$$U_E = (1 - e^{\frac{E}{E_{max}}}) \tag{4}$$

Χρησιμοποιείται εκθετική συνάρτηση και όχι γραμμική προκειμένου στα μεγάλα αποθέματα ενέργειας να υπάρξει scale down της συνεισφοράς στο Utility function για πιο ελεγχόμενη διαχείριση του αποθέματος. Στα χαμηλά αποθέματα υπάρχει σύμπτωση με τη γραμμική συμπεριφορά.

Επομένως το utility function μπορεί να ορισθεί ως η συνολική συνεισφορά από τους δύο προαναφερθέντες παράγοντες με τα ανάλογα βάρη μ_1 και μ_2 κατά αναλογία με το [41].

$$U = \mu_1 U_E + \mu_2 g(t) \tag{5}$$

όπου $\mu_1 + \mu_2 = 1$.

Ορίζοντας ένα κατώφλι για το U που το ονομάζουμε U_{thr} μπορούμε να ορίσουμε για το μοντέλο που έχει κατασκευασθεί στο J-sim όπου δεν λαμβάνεται υπόψιν η ταχύτητα στην κατανάλωση ενέργειας παρά μόνο η κίνηση ότι για $U < U_{thr}$ ο κόμβος – οπαδός παραμένει ακίνητος αναμένοντας κάποιο κόμβο με πιο έγκυρο πλαίσιο να βρεθεί στη γειτονιά του προκειμένου να το ανακτήσει. Σε διαφορετική περίπτωση όπου $U > U_{thr}$ ο κόμβος οπαδός συνεχίζει να κινείται για ανακάλυψη πληροφορίας σύμφωνα με τον PSO.

Οι πολιτικές ανακάλυψης που εφαρμόζει ένας κόμβος-οπαδός κατά την διάρκεια της αναζήτησης πλαισίων εξαρτώνται από την επιλογή των σταθερών μ_1 και μ_2. Για μεγάλες τιμές του μ_1 ($\mu_1 \geq 0.7$) ο κόμβος-οπαδός δίνει μεγαλύτερη έμφαση στο ενεργειακό του απόθεμα παρά στην ποιότητα των πλαισίων, προτιμώντας να μείνει ακίνητος όταν ξεπεράσει το κατώφλι $U < U_{thr}$ αναμένοντας κόμβους με φρέσκια πληροφορία στη γειτονιά του προκειμένου να τα ανακτήσει, και επομένως μπορεί να χαρακτηριστεί *συντηρητικός*.

Για μικρές τιμές του μ_1 ($\mu_1 <= 0.3$) ο κόμβος οπαδός δίνει μεγαλύτερη έμφαση στην ποιότητα των πλαισίων του και επομένως συνεχίζει να κινείται αδιαφορώντας για το ενεργειακό του απόθεμα προκειμένου να ανακαλύψει πλαίσια με ολοένα και μεγαλύτερη ποιότητα από αυτήν που έχουν τα πλαίσια που ήδη κατέχει. Ένας τέτοιος κόμβος-οπαδός χαρακτηρίζεται *άπληστος*. Τέλος υπάρχει και η ενδιάμεση κατάσταση όπου($0.3<\mu_1<0.7$) όπου ο κόμβος-οπαδός δίνει ανάλογη σημασία και στο ενεργειακό του απόθεμα αλλά και στην ποιότητα των πλαισίων.

5 Εφαρμογές Μοντέλου Ανακάλυψης Πληροφορίας Πλαισίου σε Περιβάλλοντα Κινητού Υπολογισμού

5.1 Περιγραφή Περιβάλλοντος Κινητού Υπολογισμού

Ας φανταστούμε ένα σύνολο νομαδικών κόμβων που εκτελούν εφαρμογές συνεργατικής επίγνωσης πλαισίου (collaborative context awareness). Οι κόμβοι αυτοί δεν έχουν όλοι το ίδιο ακριβώς σύνολο αισθητήρων ή οι φερόμενοι αισθητήρες εμφανίζουν συχνά διακοπές στη λειτουργία τους. Τότε οι κόμβοι «αναζητούν» επίκαιρη ή ποιοτικά καλύτερη πληροφορία πλαισίου. Η ποιοτική παράμετρος προσδιορίζεται κατά περίπτωση σύμφωνα με τις ιδιαιτερότητες του εκάστοτε προβλήματος.

Η αναζήτηση βασίζεται στη δυνατότητα εντοπισμού της θέσης του διαθέτη της πληροφορίας πλαισίου (πηγή), του ενδιαφερομένου (hunter) καθώς και σε αδόμητη δικτυακή επικοινωνία μικρής εμβέλειας (ad hoc short range communications). Σε κάποιο συγκεκριμένο χώρο όπου αναπτύσσονται όλοι οι κόμβοι, οι ενδιαφερόμενοι προσαρτώνται σε πηγές ή κόμβους με επίκαιρο περιεχόμενο και εφαρμόζουν διάφορες πολιτικές για την εξασφάλιση της καλύτερης δυνατής πληροφορίας πλαισίου. Η αναζήτηση αυτή ομοιάζει με την αναζήτηση καλύτερης ισχύος σήματος σε μία επικοινωνία κινητών τηλεφώνων ή την αναζήτηση ακριβούς πληροφορίας μεταξύ των συμμετεχόντων σε μία κοινωνική εκδήλωση. Ας εξετάσουμε διεξοδικότερα το τελευταίο παράδειγμα. Όλοι οι καλεσμένοι σε κάποια εκδήλωση βρίσκονται σε μία μεγάλη αίθουσα και μετακινούνται ελεύθερα προς όλες τις κατευθύνσεις. Ξαφνικά, εξαπλώνεται η φήμη για κάποιο συγκεκριμένο γεγονός. Οι άμεσα ενδιαφερόμενοι προσπαθούν να κατευθυνθούν σε περιοχές της αίθουσας όπου υπάρχει συγκεκριμένη πληροφορία για το γεγονός. Έτσι διαμορφώνονται θύλακες («πηγαδάκια») όπου ανταλλάσεται αυτή η πληροφορία.

Αντίστοιχο παράδειγμα μπορούμε να φανταστούμε στη λειτουργία ρομποτικών συσκευών που συνεργατικά επιλύουν κάποιο συγκεκριμένο πρόβλημα σε συγκεκριμένο χώρο. Στην αρχή κινούνται τυχαία μέχρι να εντοπίσουν τη συγκεκριμένη περιοχή ενδιαφέροντος (ROI). Όταν κάποιος από τους ρομποτικούς κόμβους εντοπίσει το ROI, μέσω επικοινωνιών βραχείας κλίμακας μεταδίδει σχετική πληροφορία σε ορισμένους από τους ανεπτυγμένους κόμβους οι οποίοι προστρέχουν σε βοήθεια. Οι υπόλοιποι κόμβοι συνεχίζουν τη τυχαία κίνησή τους μέχρι να ενημερωθούν για τον εντοπισμό του ROI ή να εντοπίσουν κάποιο νέο.

Στην Ωκεανογραφία είναι πολλές φορές απαραίτητη η εξερεύνηση του βυθού σε αρκετά μεγάλα βάθη δυσπρόσιτα για τον άνθρωπο, η χαρτογράφηση μολυσμένων θαλάσσιων περιοχών ή υποθαλάσσιων κοιτασμάτων πετρελαίου. Σμήνη μη επανδρωμένων υποβρύχιων οχημάτων (Unmanned Underwater Vehicles - UUV) εξοπλισμένων με αισθητήρες, ακουστικές συσκευές επικοινωνίας για το βυθό και ραδιοε-

πικοινωνιακά μέσα για την επιφάνεια αυτο-οργανώνονται και αναλαμβάνουν αυτές τις δύσκολες για τον άνθρωπο αποστολές. Μέσα από την επικοινωνία και τη συνεργατική συμπεριφορά σμήνη υποβρυχίων συγκεντρώνονται με την πάροδο χρόνου σε σημεία του βυθού με συγκεκριμένο ενδιαφέρον (π.χ. κοράλια με παράξενο χρώμα), σε σημεία με μεγάλη περιβαλλοντική μόλυνση ή σε περιοχές του βυθού με πιθανότητα ύπαρξης κοιτασμάτων πετρελαίου.

Όλα τα προαναφερθέντα παραδείγματα μπορούν να αποτελέσουν πεδίο έρευνας για την εφαρμογή του μηχανισμού ανακάλυψης πληροφορίας πλαισίου που προτείνεται στην παρούσα Διπλωματική. Μεταφορικά επομένως, οι κόμβοι-οπαδοί που σύμφωνα με το μοντέλο που αναπτύχθηκε στην §4 αποτελούν τους ενδιαφερόμενους (hunters) για την πληροφορία, μπορεί να είναι κινητά τηλέφωνα, ρομποτικές συσκευές, επισκέπτες ενός μουσείου εφοδιασμένοι με PDA's, σμήνη UUV κλπ

Η ανάλυση των αποτελεσμάτων από την εφαρμογή του μοντέλου/μηχανισμού ανακάλυψης επίκαιρων πλαισίων σε πεδία εφαρμογών κινητού υπολογισμού επικεντρώνεται στην αναπαράσταση της Ευκλείδιας απόστασης των οπαδών από τις πηγές όπως επίσης και στη μετάβαση καταστάσεων κάθε οπαδού καθώς κινείται στο πλέγμα προσομοίωσης προκειμένου να ανακτήσει φρέσκα πλαίσια Ως μέτρο της σύγκλισης των οπαδών σε κάποιες πηγές χρησιμοποιείται η τυπική απόκλιση από το μέσο όρο η οποία δίνει μια εικόνα του βαθμού διακύμανσης (degree of fluctuation) της απόστασης και δίνεται από τη σχέση

$$\sigma = \sqrt{\frac{\sum (x - \overline{x})^2}{N - 1}} \tag{6}$$

Η αναπαράσταση του ποσοστού του χρόνου για το οποίο ένας οπαδός βρίσκεται σε μια συγκεκριμένη κατάσταση δίνει μια εικόνα του βαθμού ικανοποίησης των οπαδών στην προσπάθεια τους να ανακτήσουν φρέσκια πληροφορία.

Η παρούσα ανάλυση επικεντρώνεται σε δύο από τους κατά τεκμήριο αρκετούς παράγοντες που μπορούν να επηρεάσουν την ανακάλυψη πληροφορίας πλαισίου σε ένα WSN που θεωρούνται ίσως πιο σημαντικοί. Πιο συγκεκριμένα στην κινητικότητα των πηγών επιλέγοντας τέσσερα σενάρια με μέγιστες ταχύτητες πηγών 2,10,20 και 50 αντίστοιχα, όπως επίσης και στο μέγεθος της γειτονιάς κάθε κόμβου από όπου αντλείται η πληροφορία για την πλοήγηση του κόμβου μέσα στο πλέγμα προσομοίωσης (αλγόριθμος PSO) και το οποίο μέγεθος γειτονίας αποτελεί και μέτρο της πυκνότητας του πλέγματος αφού καθορίζει την εμβέλεια στην οποία κάθε οπαδός αντλεί πληροφορία από πηγές ή από άλλους οπαδούς.

Επιλέχθηκαν τέσσερα σενάρια προσομοίωσης με μέγεθος γειτονίας 2,3,4 και 5.ενώ η μέγιστη ταχύτητα των πηγών σε αυτά τα σενάρια ορίσθηκε σε 2 και επιλέχθηκε ως η βέλτιστη από την ανάλυση της κινητικότητας των πηγών που προηγήθηκε και θα παρατεθεί παρακάτω.

Τέλος παρατίθεται και ένα σενάριο όπου οι κόμβοι πλοηγούνται μέσα στο πεδίο του Κινητού Υπολογισμού βάσει ενός τυχαίου μοντέλου κίνησης (Random Way

Model) για διάφορες τιμές του μεγέθους γειτονίας και το οποίο χρησιμοποιείται σαν βάση σύγκρισης για την αξιολόγηση της απόδοσης του αλγορίθμου PSO ως προς τη χρησιμότητα του για την ανακάλυψη φρέσκων πλαισίων πληροφορίας.

5.2 Συμπεριφορά Μοντέλου σε σχέση με την κινητικότητα των Πηγών Πληροφορίας

Οι κύριες παράμετροι της προσομοίωσης φαίνονται στον Πίνακα 1. Όσον αφορά τις

Πίνακας 1: Παράμετροι προσομοίωσης - Σενάριο κινητικότητας πηγών

Παράμετροι	**Τιμές**
Αριθμός πηγών	4
Αριθμός οπαδών	6
Μέγεθος πλέγματος προσομοίωσης X	100
Μέγεθος πλέγματος προσομοίωσης Y	100
Μέγεθος στοιχείου πλέγματος X	20
Μέγεθος στοιχείου πλέγματος Y	20
Μέγεθος γειτονίας	2
Κατώφλι εγκυρότητας πλαισίου	0.2
Σταθερά c1 (PSO) (Subjective)	0.8
Σταθερά c2 (PSO) (Objective)	0.2
Σταθερά μ1 (Utility function)	0.8
Σταθερά μ1 (Utility function)	0.2
Κατώφλι (Utility function)	0.2
Μέγιστη ταχύτητα οπαδών (Random way)	0.5

τιμές του Πίνακα 1 θα πρέπει να αναφερθεί ότι η μέγιστη ταχύτητα οπαδών αναφέρεται στο Random Way Model το οποίο εφαρμόζεται στην περίπτωση που κόμβοι και πηγές έχουν χαθεί μεταξύ τους (τα πλαίσια που κατέχουν έχουν απαρχαιωθεί) έτσι ώστε με την ανακατάταξη αυτή να υπάρξει ξανά επικοινωνία και να πλοηγηθούν πλέον βάσει του PSO. Οι διακοπές αυτές είναι όπως αναμένεται πιο συχνές για μικρό μέγεθος γειτονίας (μικρή εμβέλεια πηγών και οπαδών) και μεγάλα πεδία προσομοίωσης. Μπορεί εν μέρει να εμφανίζονται και για πηγές που κινούνται με μεγάλη ταχύτητα. Οι απότομες μεταβάσεις που εμφανίζονται τόσο στα διαγράμματα απόστασης όσο και σε αυτά της μετάβασης καταστάσεων (βλ. παρακάτω σχήματα) αντιστοιχούν ακριβώς σε αυτές τις διακοπές επικοινωνίας.

Οι σταθερές για το Utility function έχουν επιλεγεί να είναι biased ως προς τον ενεργειακό όρο δηλ. η πολιτική κάθε κόμβου να σταματήσει ή να συνεχίζει να κινείται για την ανακάλυψη πλαισίων να επηρεάζεται κατά μεγαλύτερο βαθμό από το ενεργειακό του απόθεμα παρά από την ποιότητα του πλαισίου που κατέχει (συντηρητική πολιτική ανακάλυψης πλαισίου). Τέλος το μέγεθος γειτονίας ορίζεται σε 2 που αντιστοιχεί σε πυκνότητα πλέγματος 50,2%. Παρακάτω φαίνονται τα διαγράμματα

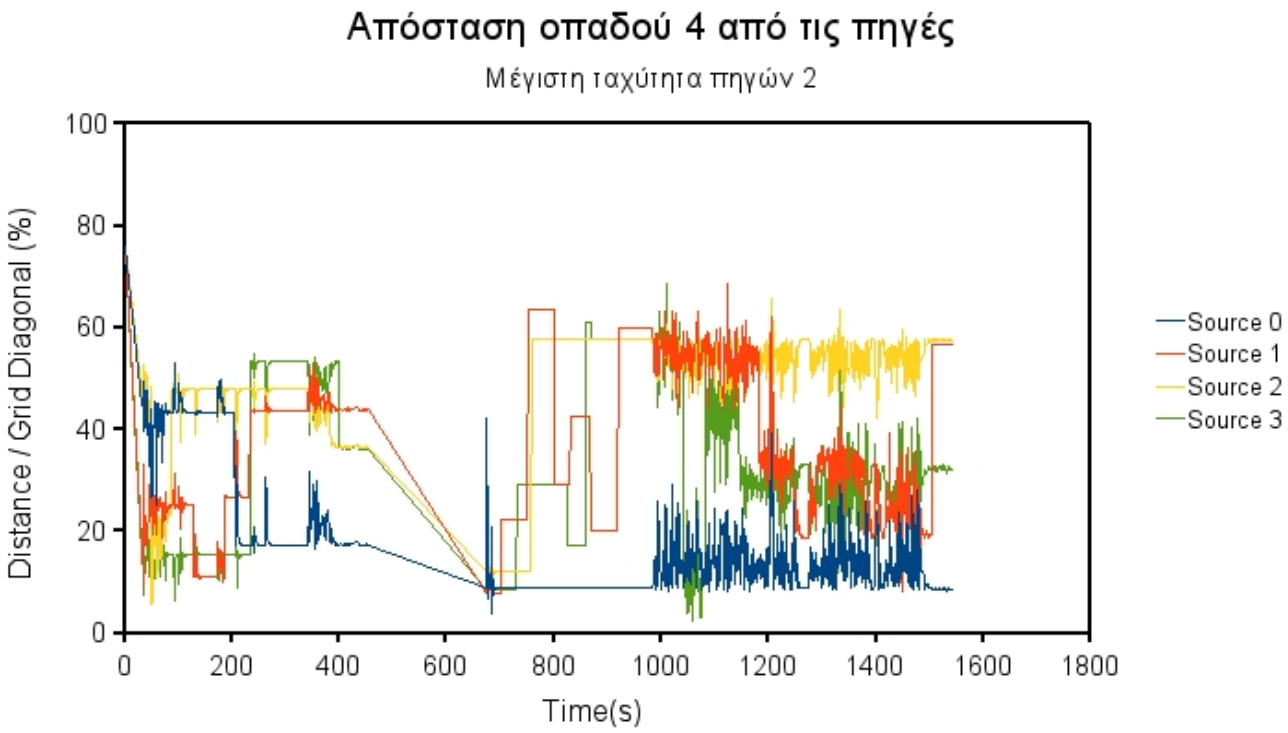
της απόστασης από τις πηγές και της μετάβασης καταστάσεων για τον κόμβο-οπαδό 4 και για κάθε τιμή της μέγιστης ταχύτητας των πηγών που προσομοιώθηκε.)

Στα Σχήματα 6, 7, 8 και 9 παρατηρούμε ότι η διακύμανση της απόστασης οπαδών-πηγών οδηγείται σε μεγαλύτερες τιμές με την αύξηση της κινητικότητας των πηγών για μέγιστη ταχύτητα πηγών ίση με 50 δυσκολεύοντας τη σύγκλιση οπαδών-πηγών. Παρόλα αυτά το ποσοστό του χρόνου που ένας οπαδός είναι ικανοποιημένος από την ποιότητα των πλαισίων που έχει ανακτήσει δεν χειροτερεύει με την αύξηση της κινητικότητας των πηγών όπως μπορούμε να συμπεράνουμε από το Σχήμα 12 για την περίπτωση του οπαδού 4. Ο λόγος έγκειται στο γεγονός ότι οι μεγάλες ταχύτητες με τις οποίες κινούνται οι πηγές επιτρέπουν την ομοιόμορφη διάχυση φρέσκιας πληροφορίας σε όλο το πλέγμα καθ'όλη τη διάρκεια της προσομοίωσης (οι πηγές κινούνται με το Random Way Model και επομένως μια τέτοια υπόθεση μπορεί να θεωρηθεί ευσταθής) με αποτέλεσμα οι οπαδοί να μπορούν να ανακτούν φρέσκα πλαίσια από τις πηγές αλλά κυρίως σε αυτές τις μεγάλες κινητικότητες από τους άλλους οπαδούς.

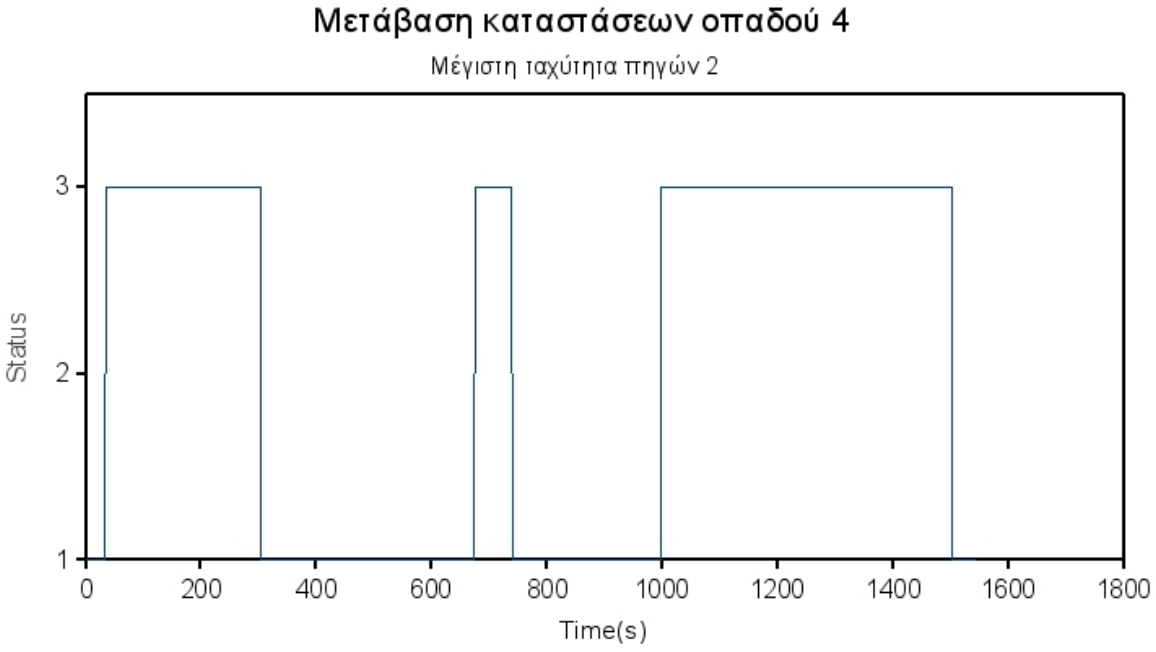
Θα πρέπει να σημειωθεί ότι στα διαγράμματα μετάβασης καταστάσεων οι τιμές 1,2 και 3 στον άξονα των Y αντιστοιχούν στις καταστάσεις Έωλος, Συμβιβάσιμος και Ικανοποιημένος αντίστοιχα. Παρατηρούμε από τα διαγράμματα των Σχημάτων 6, 7, 8 και 9 αλλά και από τα Σχήματα 12 και 13, ότι ο κόμβος δεν μεταβαίνει στην κατάσταση συμβιβάσιμος. Ο προφανής λόγος για αυτό είναι ότι οπαδοί διατηρούν έγκυρα πλαίσια κατά τη διάρκεια της προσομοίωσης, ενώ η προυπόθεση για να χαρακτηριστεί ένας κόμβος συμβιβάσιμος είναι να κατέχει μη έγκυρο πλαίσο αλλά σαφώς καλύτερο από αυτό των γειτόνων του. Κάτι τέτοιο φαίνεται να συμβαίνει μόνο στην περίπτωση όπου έχει εφαρμοστεί το Radnom Way Model για τους οπαδούς εξαιτίας της διακοπής επικοινωνίας τους με άλλες πηγές ή οπαδούς και κατέχουν μη έγκυρο πλαίσιο εκένη τη στιγμή αλλά πιθανόν καλύτερο από τα πλαίσια των γειτόνων του.

Από το Σχήμα 11 φαίνεται ότι η μικρότερη διακύμανση εμφανίζεται στην περίπτωση όπου η μέγιστη ταχύτητα των πηγών είναι 2. Συνδυάζοντας και το Σχήμα 13 όπου φαίνεται ο μέσος όρος για όλους τους οπαδούς σχετικά με το ποσοστό χρόνου που βρίσκονται σε μια κατάσταση φαίνεται ότι η επιλογή της μέγιστης ταχύτητας πηγών ίσης με 2 είναι καλή προκειμένου να μελετηθεί και η επίδραση του άλλου παράγοντα που αναφέρθηκε προηγουμένως, δηλ. του μεγέθους της γειτονιάς.

Θα πρέπει τέλος να σημειωθεί ότι η τροχιά των πηγών πληροφορίας κατά την κίνησή τους από ένα σημείο του πεδίου προσομοίωσης σε κάποιο άλλο δεν είναι συνεχής αλλά πραγματοποιείται στιγμιαία με αποτέλεσμα οι οπαδοί να χάνουν τις πηγές στη χρονική στιγμή που αυτές μετακινούνται και έτσι να παρουσιάζεται στα διαγράμματα απόστασης, η εικόνα των διαδοχικών συγκλίσεων πηγών-οπαδών. Αυτή η ιδιαιτερότητα του εργαλείου προσομοίωσης (J-sim) μπορεί να μην ανταποκρίνεται επαρκώς στην κίνηση των πηγών πληροφορίας σε ένα πραγματικό περιβάλλον κινητού υπολογισμού αλλά δε περιορίζει την δυνατότητα του προτεινόμενου μηχανισμού να οδηγεί σε σύγκλιση οπαδών-πηγών και στην ανάκτηση φρέσκιας πληροφορίας πλαισίου.

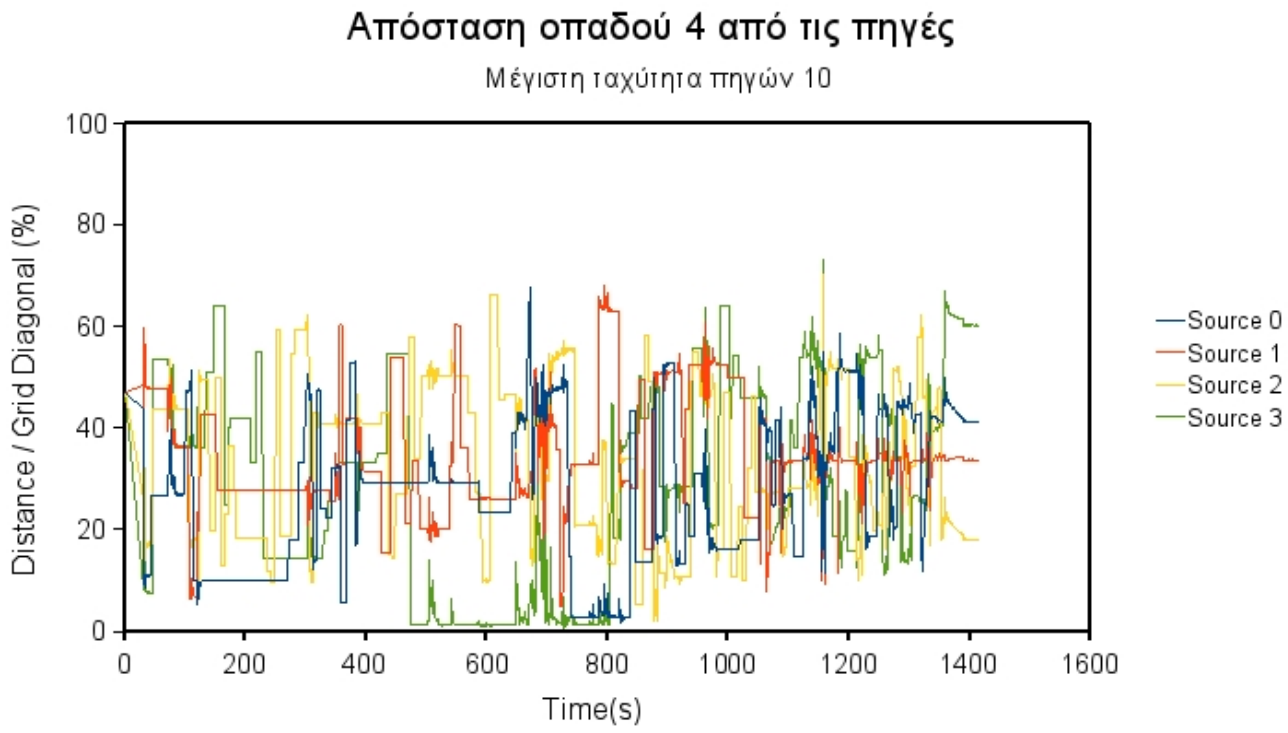


i)

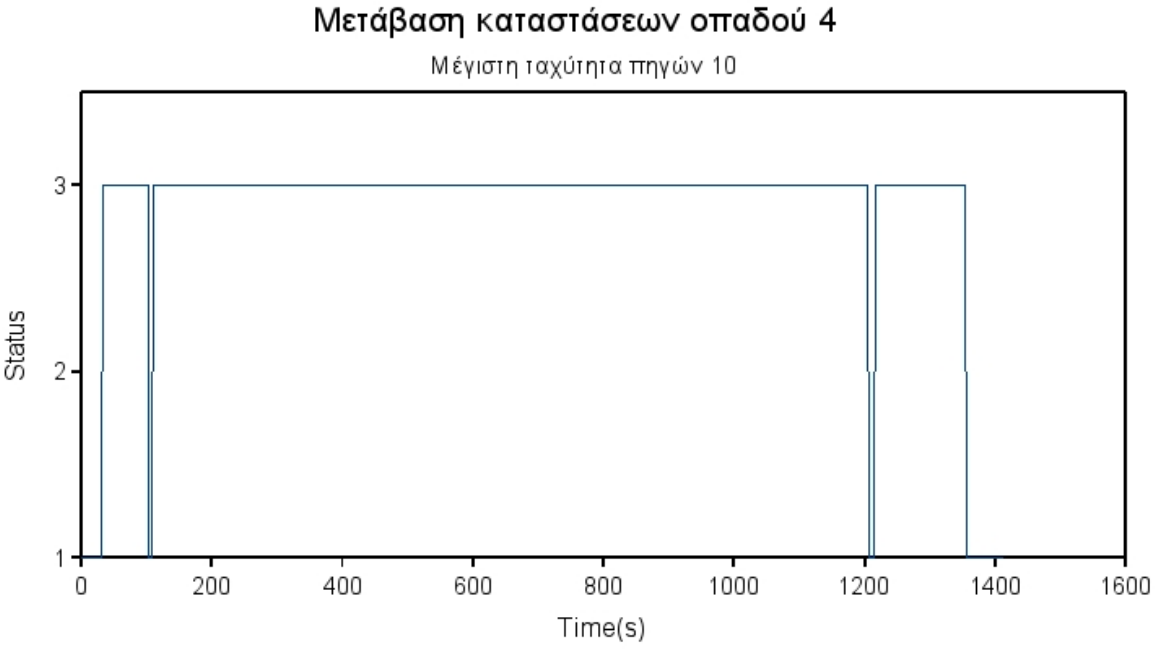


ii)

Σχήμα 6: Ταχύτητα Πηγών 2 - i) Απόσταση ii) Μετάβαση καταστάσεων

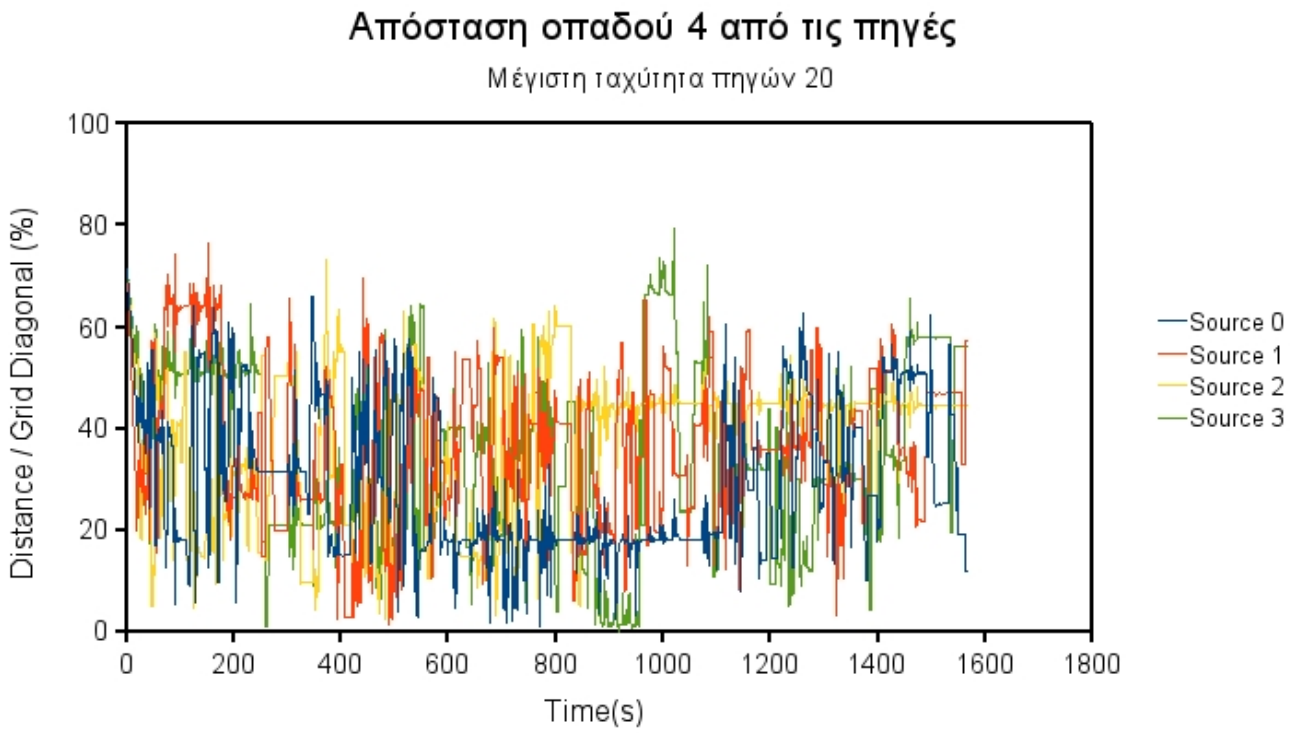


i)

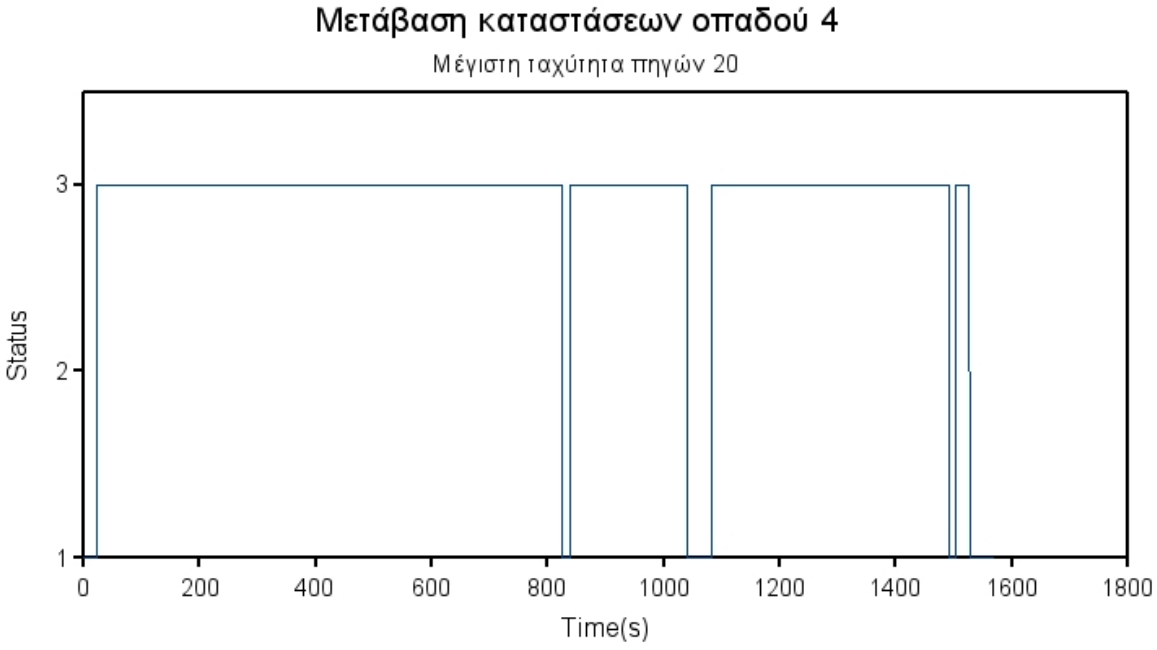


ii)

Σχήμα 7: Ταχύτητα Πηγών 10 - i) Απόσταση ii) Μετάβαση καταστάσεων

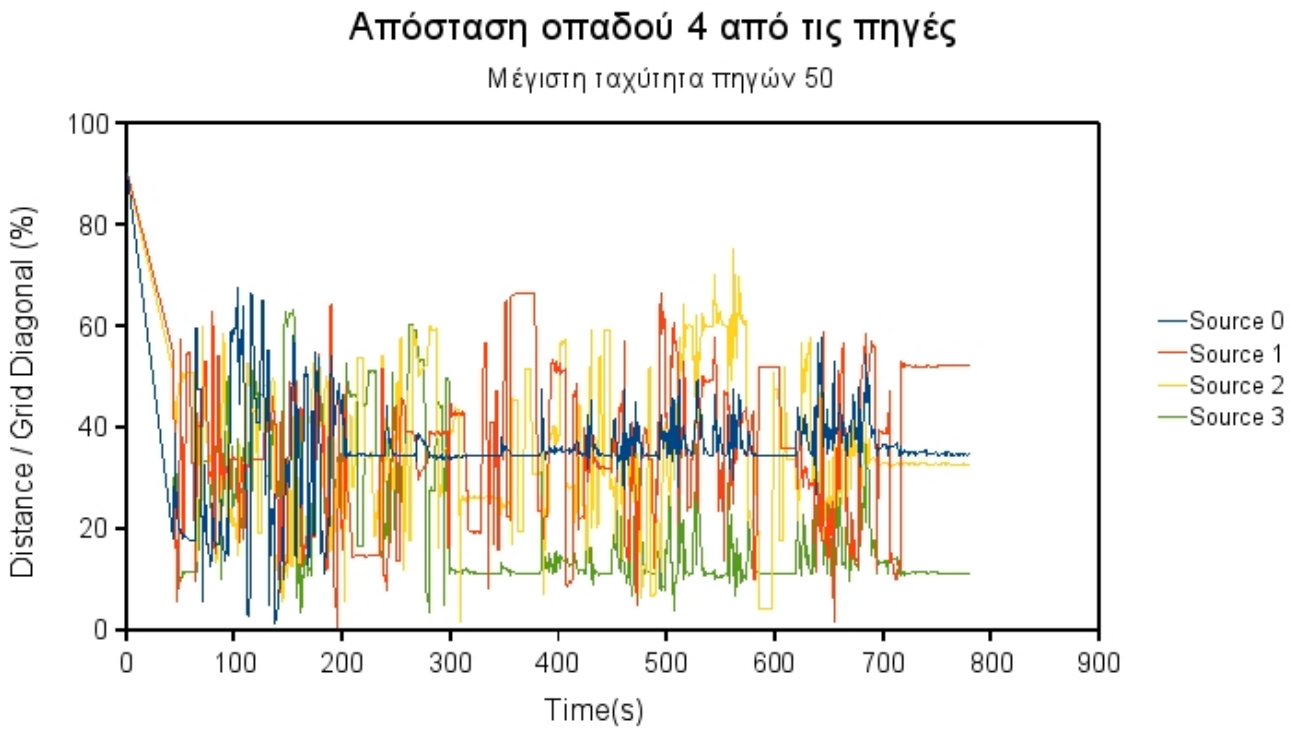


i)

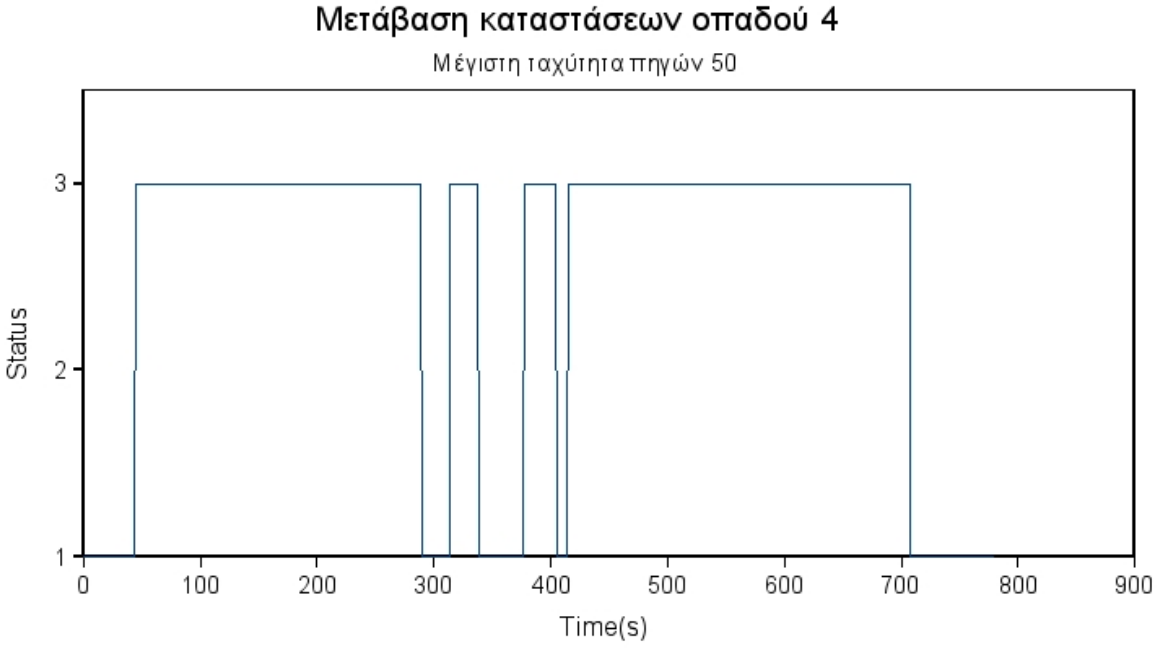


ii)

Σχήμα 8: Ταχύτητα Πηγών 20 - i) Απόσταση ii) Μετάβαση καταστάσεων

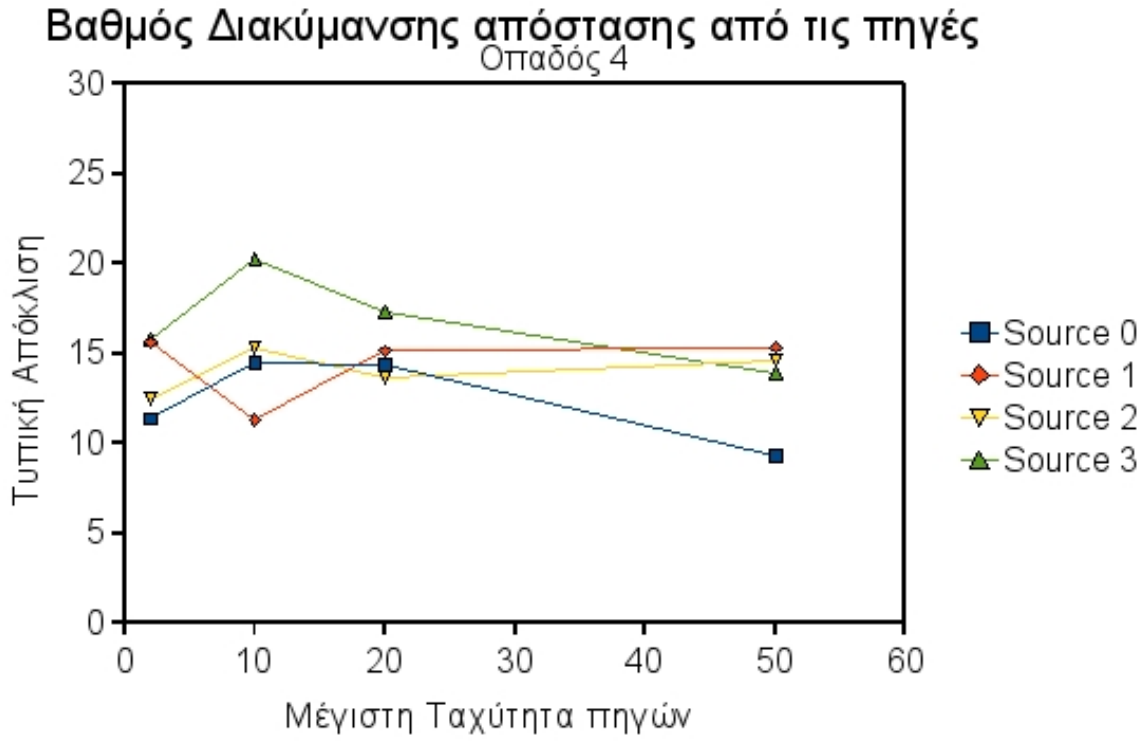


i)

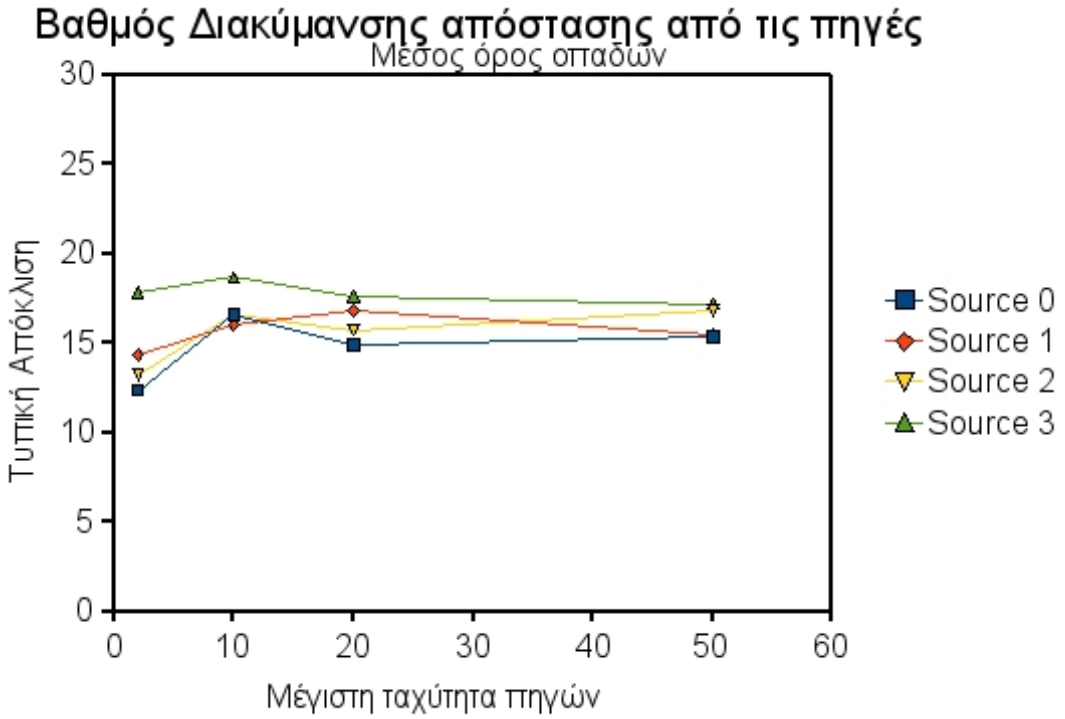


ii)

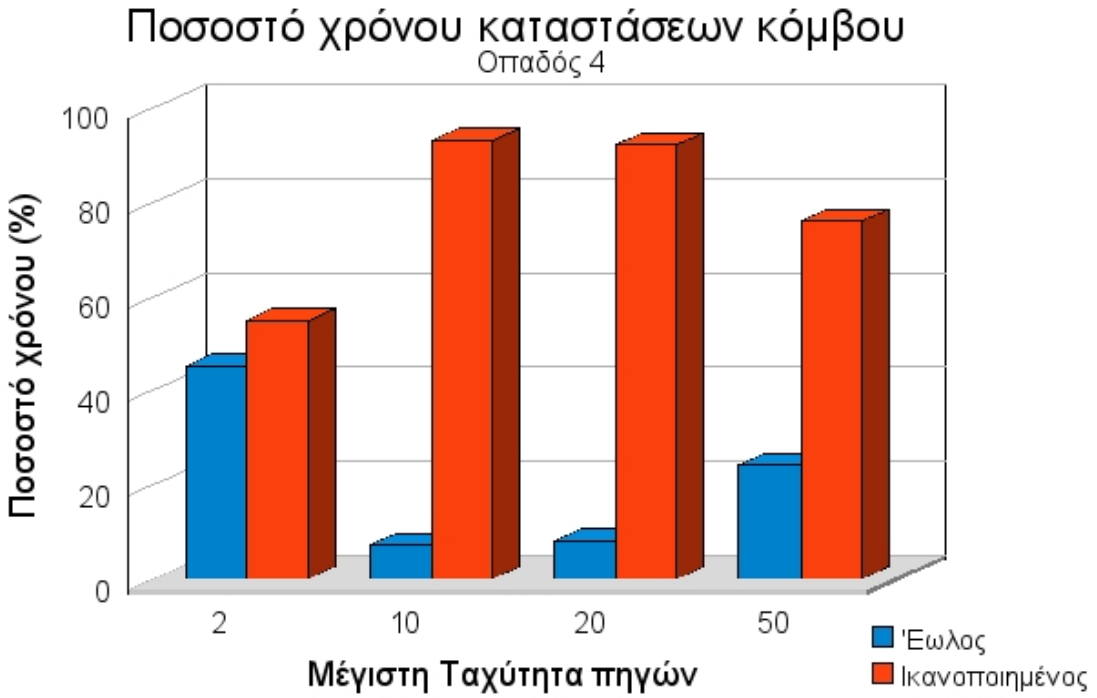
Σχήμα 9: Ταχύτητα Πηγών 50 - i) Απόσταση ii) Μετάβαση καταστάσεων



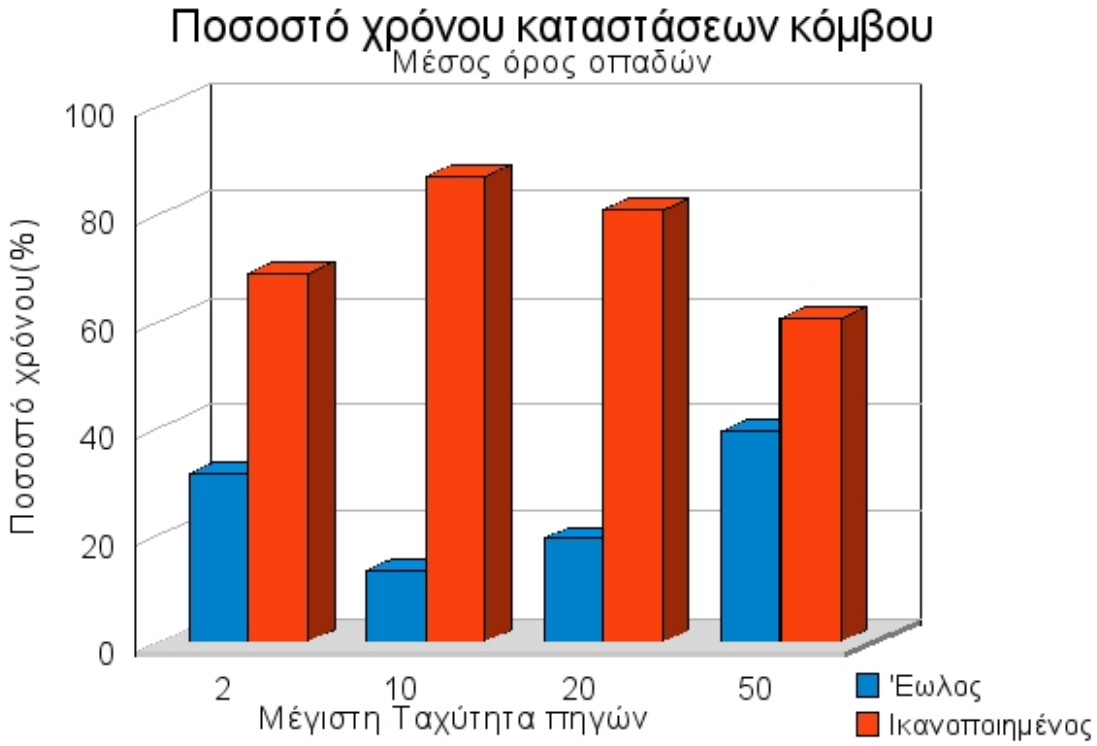
Σχήμα 10: Βαθμός Διακύμανσης απόστασης από τις πηγές για τον οπαδό 4



Σχήμα 11: Βαθμός Διακύμανσης απόστασης από τις πηγές. Μέσος όρος ο-παδών



Σχήμα 12: Ποσοστό χρόνου καταστάσεων οπαδού 4



Σχήμα 13: Ποσοστό χρόνου καταστάσεων. Μέσος όρος οπαδών

5.3 Συμπεριφορά Μοντέλου σε σχέση με το μέγεθος γειτονίας των κόμβων

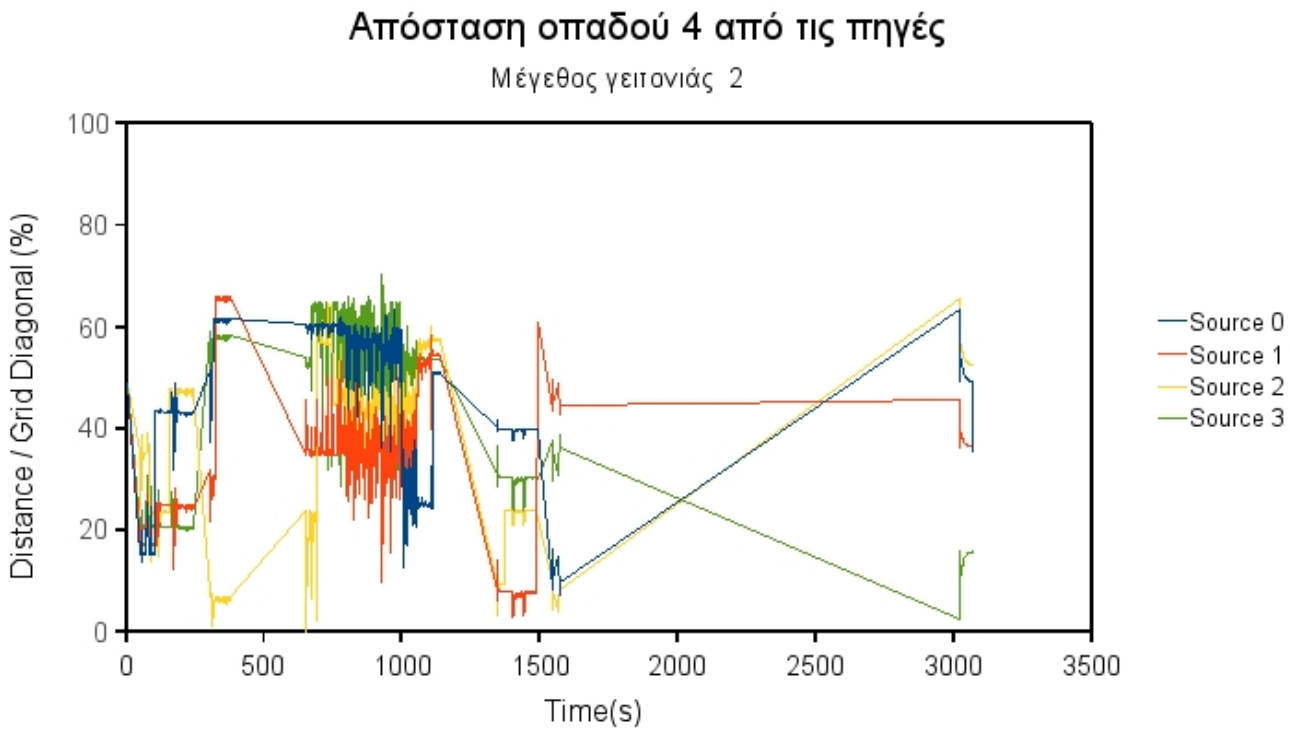
Ο συγκεκριμένος παράγοντας έχει άμεση σχέση με την πυκνότητα του πλέγματος αφού η τελευταία μεταβάλλεται είτε με τη μεταβολή του αριθμού των πηγών και των οπαδών είτε με τη μεταβολή του μεγέθους γειτονίας. Επιλέγουμε τέσσερα μεγέθη γειτονίας 2,3,4 και 5 που αντιστοιχούν σε πυκνότητες πλέγματος 12.6, 28.3, 50.2 και 78.5% σύμφωνα με τη σχέση $\pi r^2/E_{Grid}$ όπου r η ακτίνα που ορίζει τη γειτονιά και E_{Grid} το εμβαδό του πεδίου προσομοίωσης. Στον Πίνακα 2 φαίνονται οι παράμετροι της προσομοίωσης.

Στην περίπτωση του μεγέθους γειτονίας 2 υπάρχουν αρκετές διακοπές επικοινωνίας που αναγκάζουν τους οπαδούς να βρίσκονται στην έωλο κατάσταση για μεγάλο χρονικό διάστημα και το ποσοστό χρόνου κατά το οποίο ένας οπαδός είναι κατά μέσο όρο ικανοποιημένος είναι το μικρότερο και από τα τέσσερα σενάρια. (βλ. Σχ 21). Οπότε η περίπτωση αυτού του μεγέθους γειτονίας δεν είναι η ιδανικότερη για έλεγχο διαφόρων άλλων παραμέτρων της προσομοίωσης.

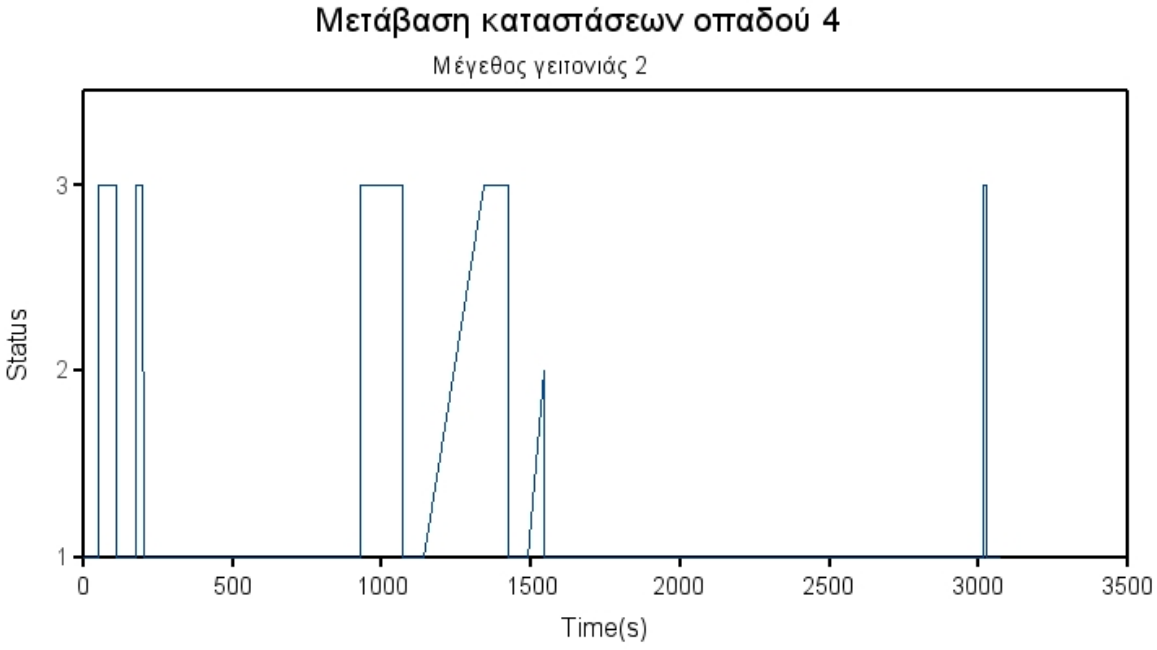
Το ίδιο ισχύει και για την περίπτωση μεγέθους γειτονίας 5 όπου ουσιαστικά καλύπτεται σχεδόν το πεδίο προσομοίωσης και ένας οπαδός έχει μεγάλη πιθανότητα να λάβει «πλαίσια από το σύνολο σχεδόν των πηγών και οπαδών αφού όλο το πεδίο προσομοίωσης περίπου θεωρείται γειτονιά του. Η διακύμανση είναι σε καλά επίπεδα όπως επίσης και ο μέσος χρόνος που ένας οπαδός είναι ικανοποιημένος (Σχήμα 21) Επομένως οι άλλες δύο περιπτώσεις φαντάζουν ιδανικότερες με την τιμή 4 να έχει προτεραιότητα λόγω του καλύτερου ποσοστού της ικανοποιημένης κατάστασης όπως φαίνεται και από το Σχήμα 21.

Πίνακας 2: Παράμετροι προσομοίωσης - Σενάριο μεγέθους γειτονίας

Παράμετροι	**Τιμές**
Αριθμός πηγών	4
Αριθμός οπαδών	6
Μέγεθος πλέγματος προσομοίωσης X	200
Μέγεθος πλέγματος προσομοίωσης Y	200
Μέγεθος στοιχείου πλέγματος X	20
Μέγεθος στοιχείου πλέγματος Y	20
Κατώφλι εγκυρότητας πλαισίου	0.2
Σταθερά c1 (PSO) (Subjective)	0.8
Σταθερά c2 (PSO) (Objective)	0.2
Σταθερά μ1 (Utility function)	0.8
Σταθερά μ1 (Utility function)	0.2
Κατώφλι (Utility function)	0.2
Μέγιστη ταχύτητα οπαδών (Random way)	0.5
Μέγιστη ταχύτητα πηγών	2

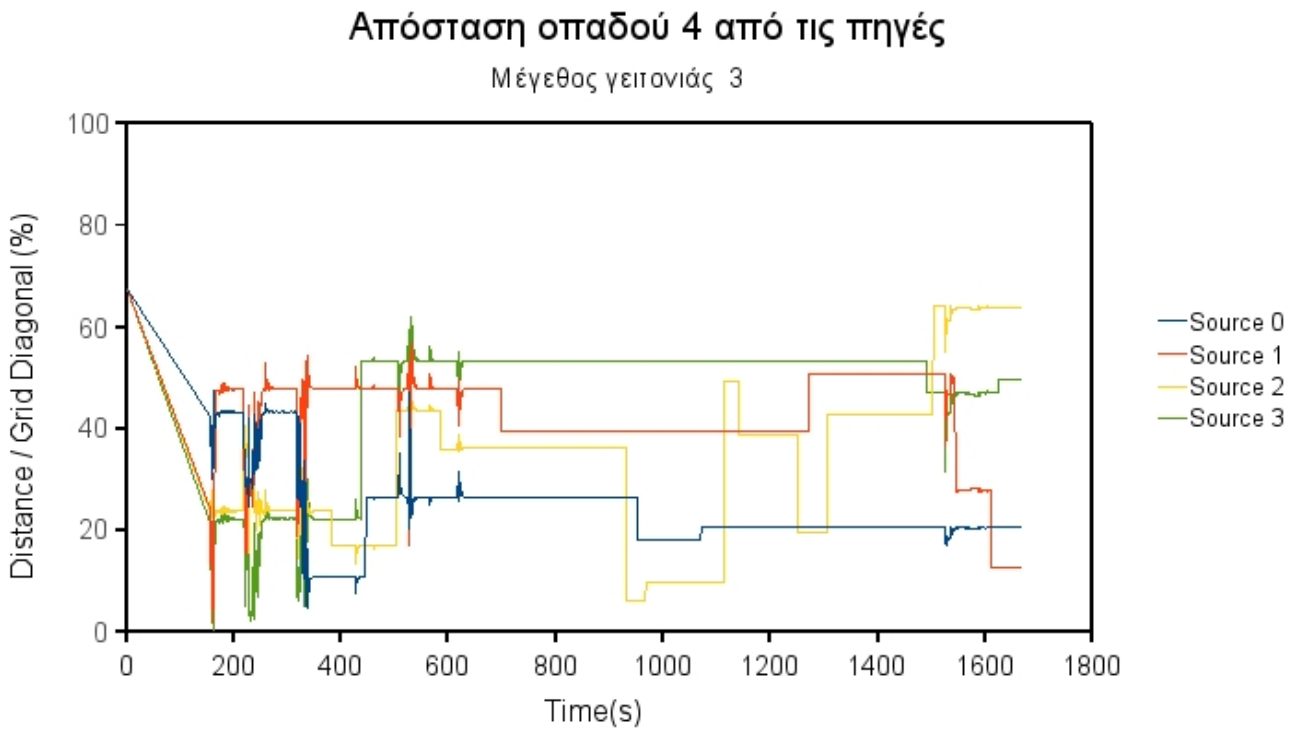


i)

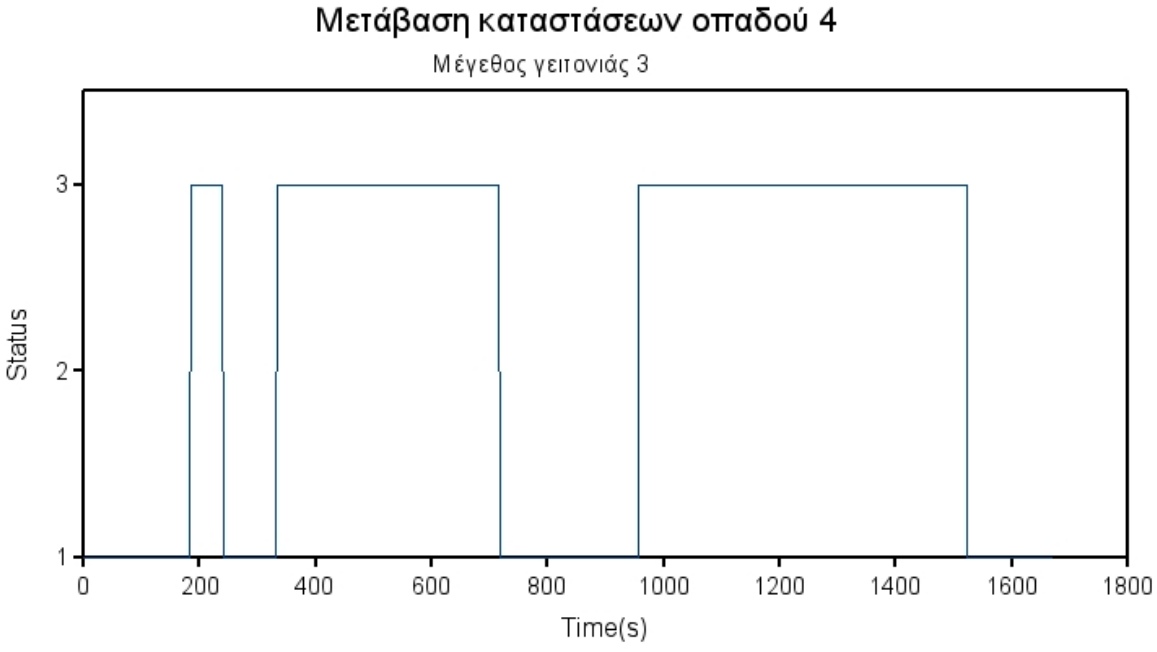


ii)

Σχήμα 14: Μέγεθος γειτονίας 2 - i) Απόσταση ii) Μετάβαση καταστάσεων

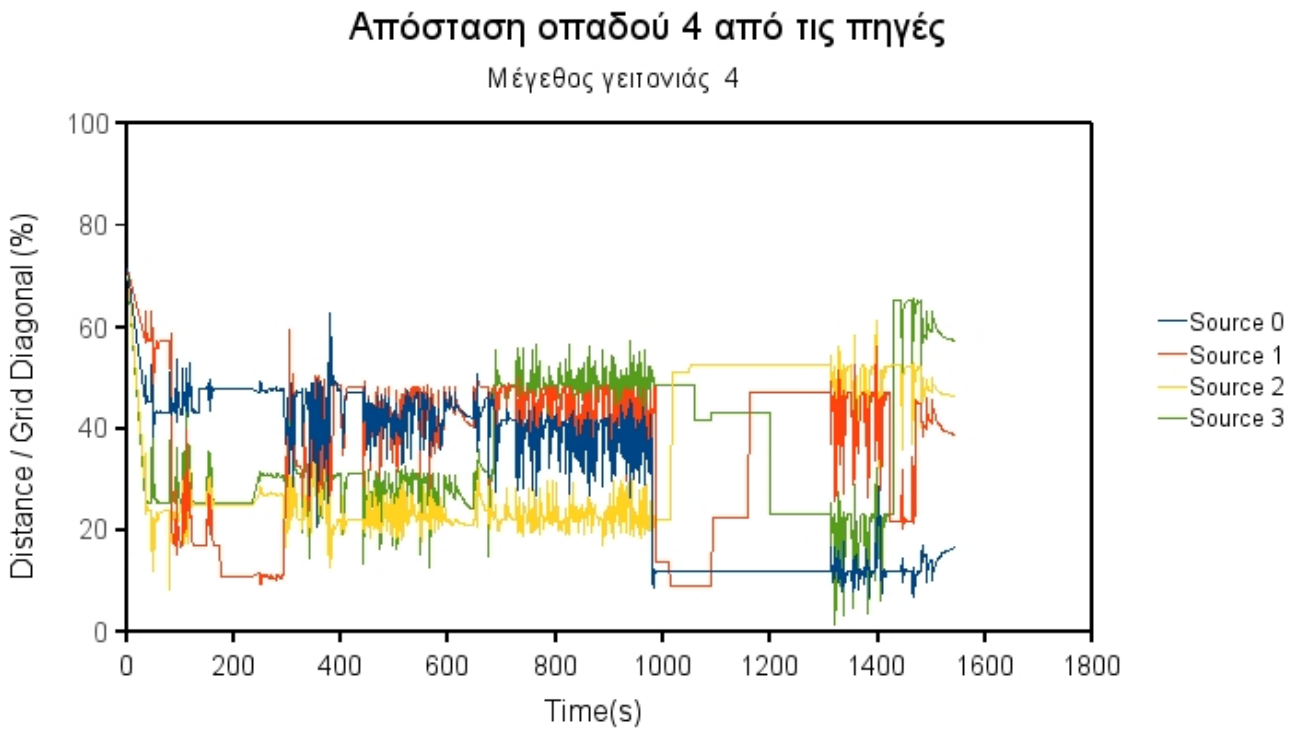


i)

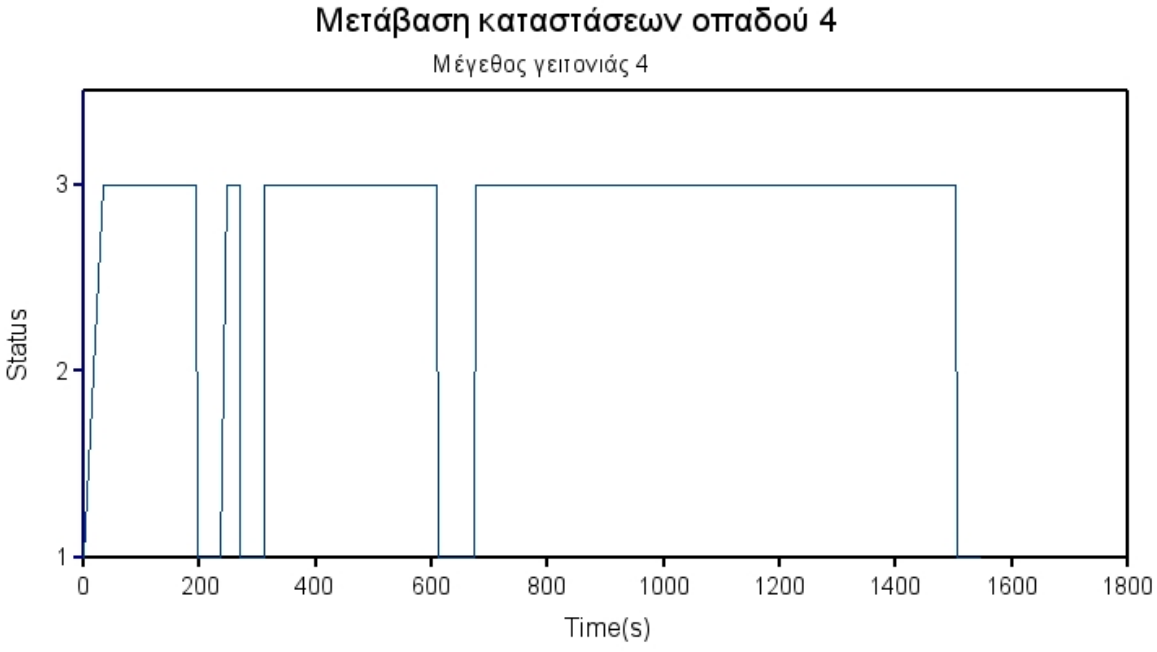


ii)

Σχήμα 15: Μέγεθος γειτονίας 3 - i) Απόσταση ii) Μετάβαση καταστάσεων

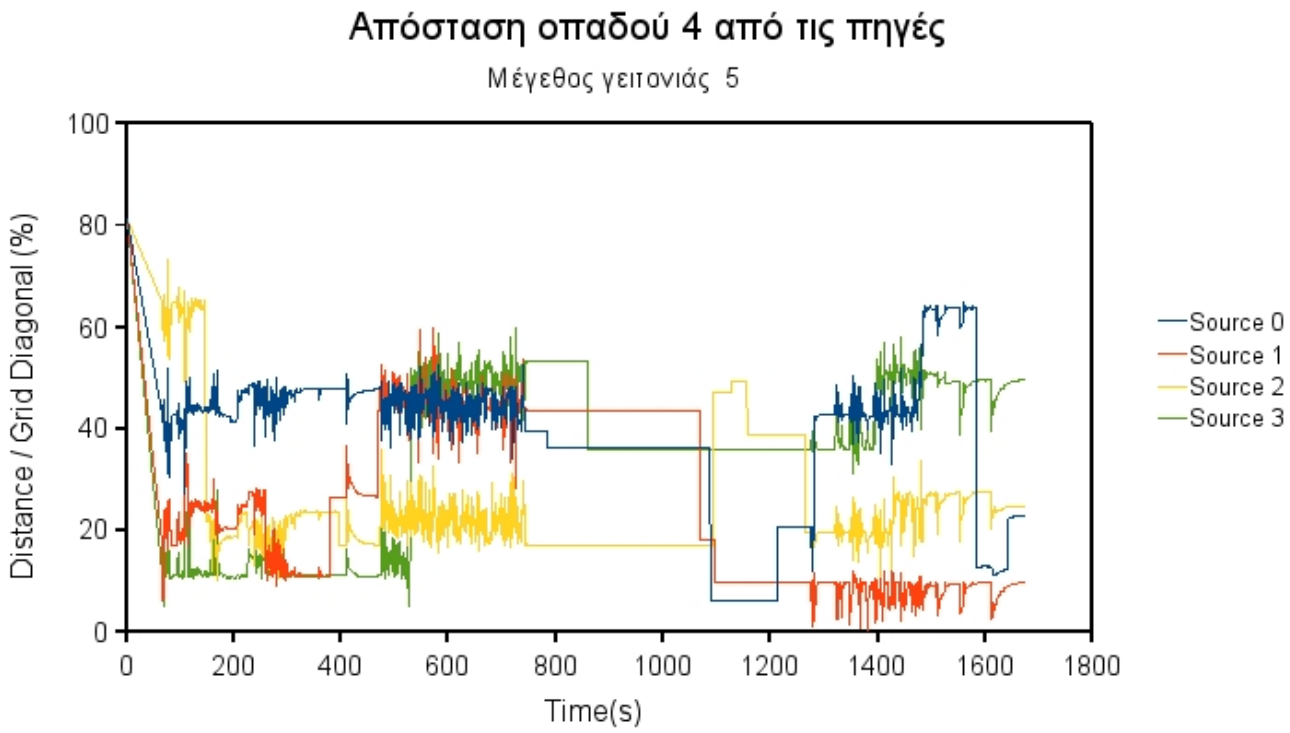


i)

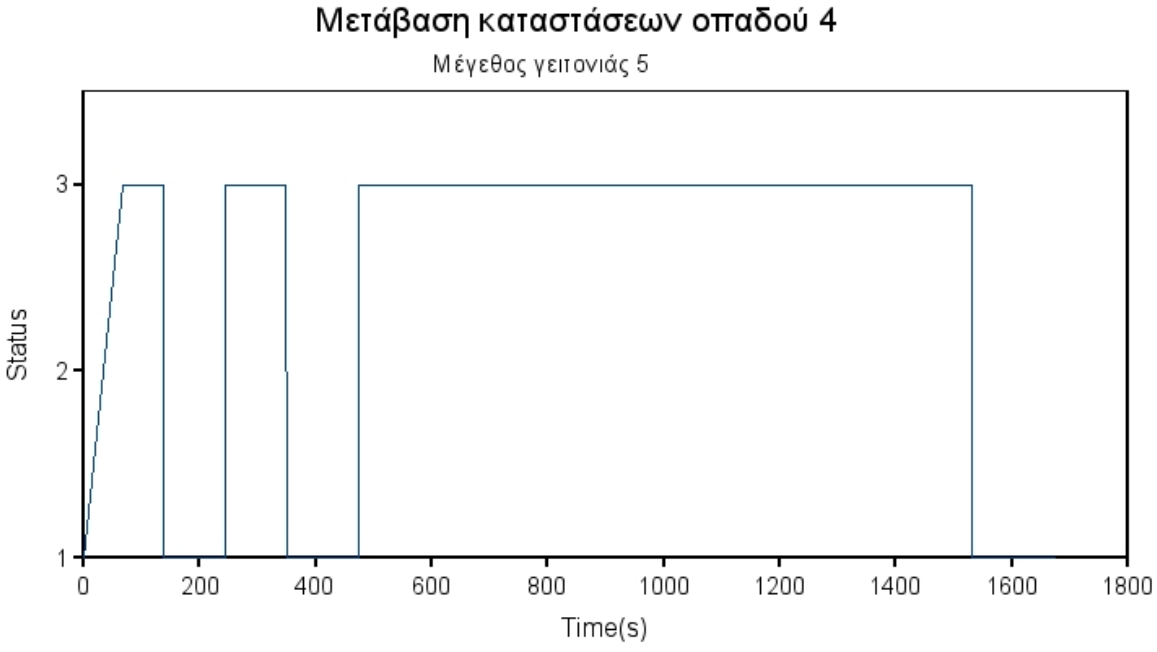


ii)

Σχήμα 16: Μέγεθος γειτονίας 4 - i) Απόσταση ii) Μετάβαση καταστάσεων

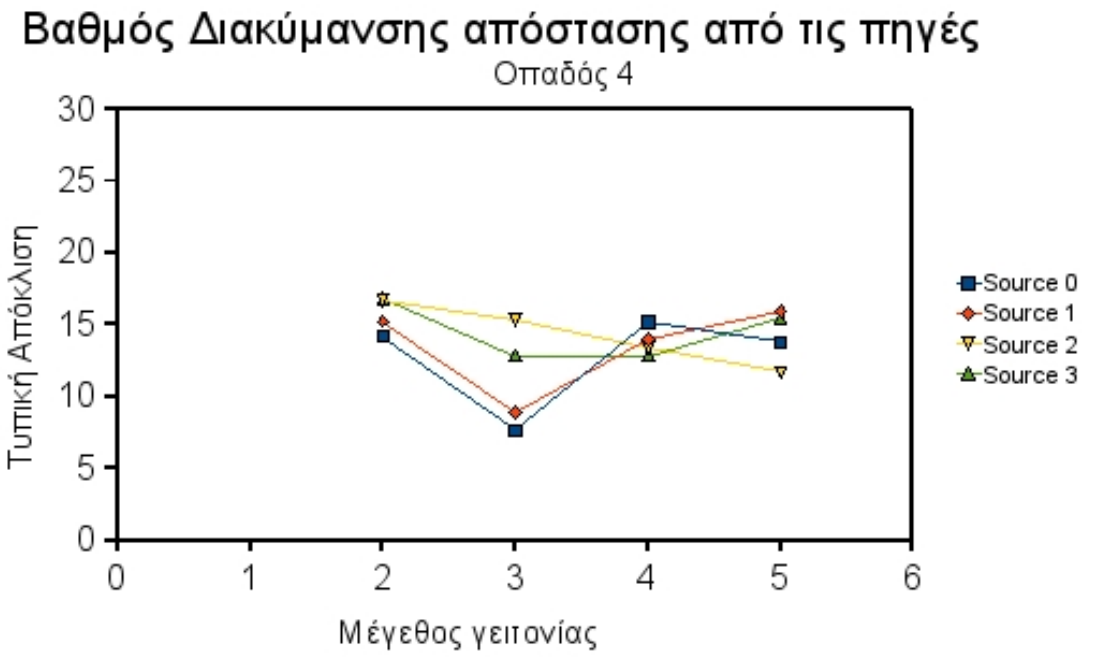


i)

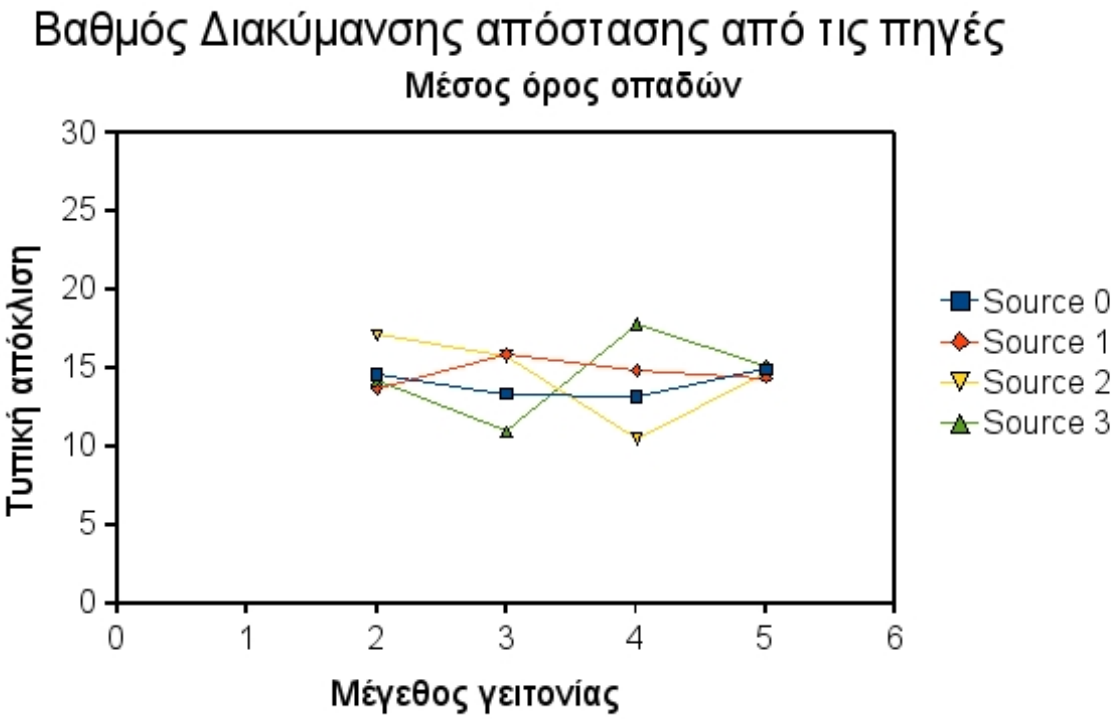


ii)

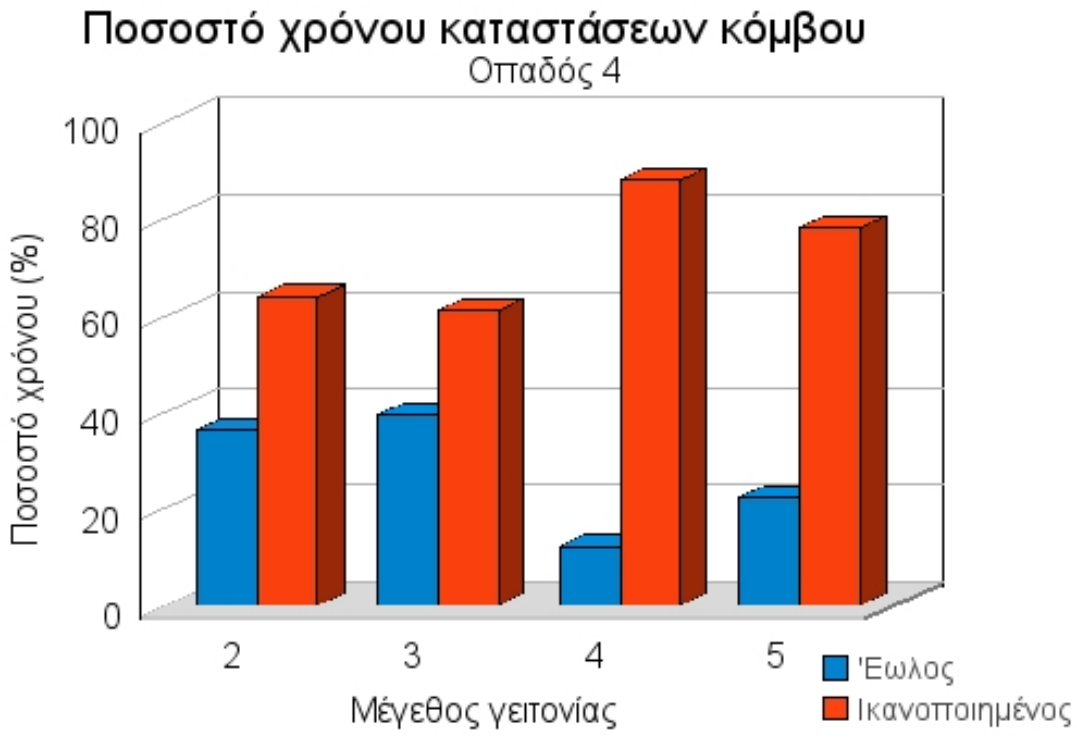
Σχήμα 17: Μέγεθος γειτονίας 5 - i) Απόσταση ii) Μετάβαση καταστάσεων



Σχήμα 18: Βαθμός Διακύμανσης απόστασης από τις πηγές για τον οπαδό 4



Σχήμα 19: Βαθμός Διακύμανσης απόστασης από τις πηγές. Μέσος όρος ο-παδών

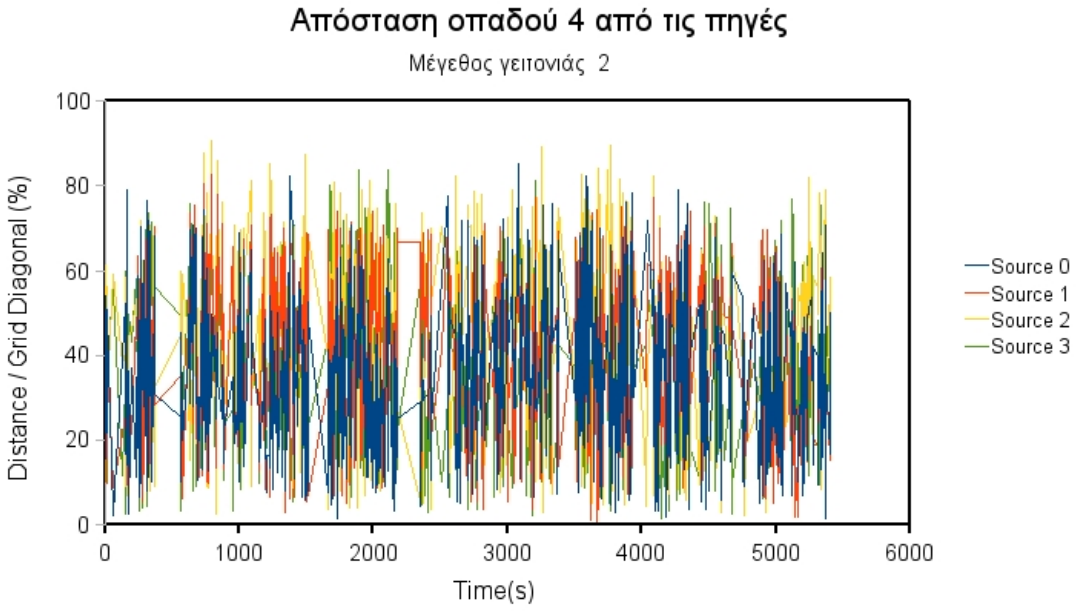


Σχήμα 20: Πσοσοστό χρόνου καταστάσεων οπαδού 4

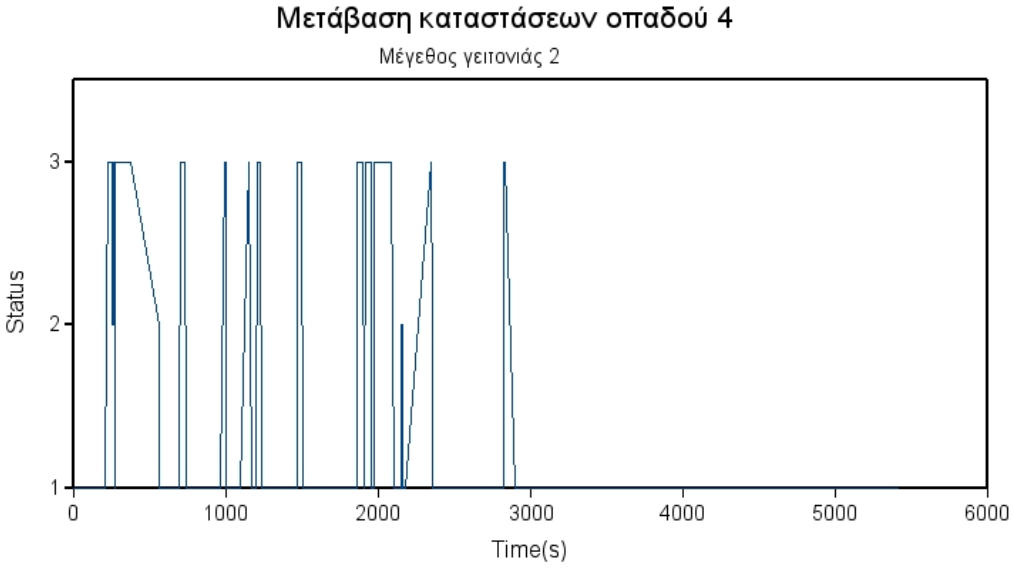
Σχήμα 21: Ποσοστό χρόνου καταστάσεων. Μέσος όρος οπαδών

5.4 Συμπεριφορά Μοντέλου για τυχαία κίνηση των κόμβων-οπαδών

Τα παρακάτω αποτελέσματα αναφέρονται σε προσομοίωση με παραμέτρους όμοιες με αυτές του Πίνακα 2 και για τα ίδια μεγέθη γειτονίας όπως στο προηγούμενο σενάριο με μόνη διαφορά ότι σε αυτήν την περίπτωση οι οπαδοί πλοηγούνται βάσει του Random Way Model σύμφωνα με το οποίο κινούνται και οι πηγές και όχι σύμφωνα με τον αλγόριθμο PSO.

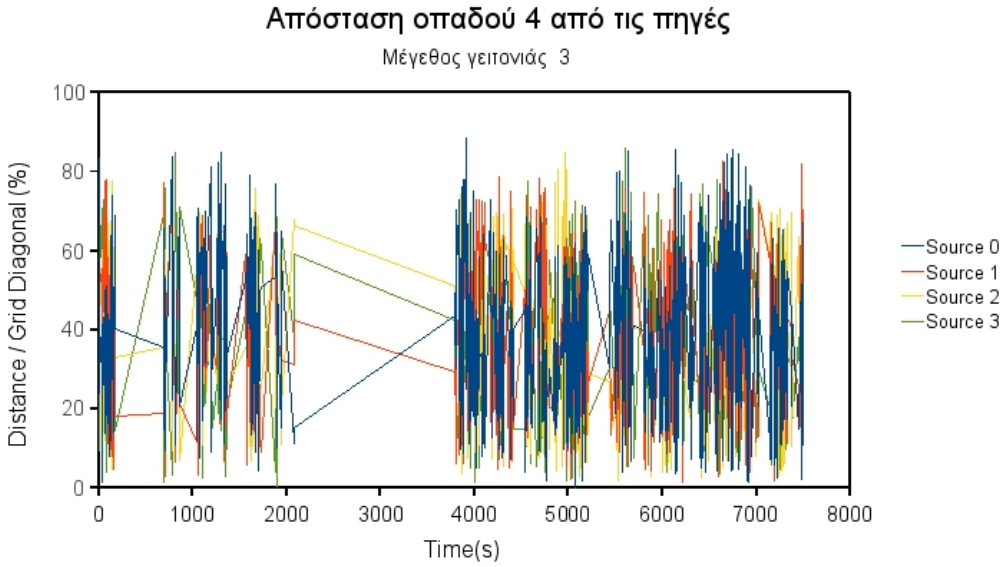


i)

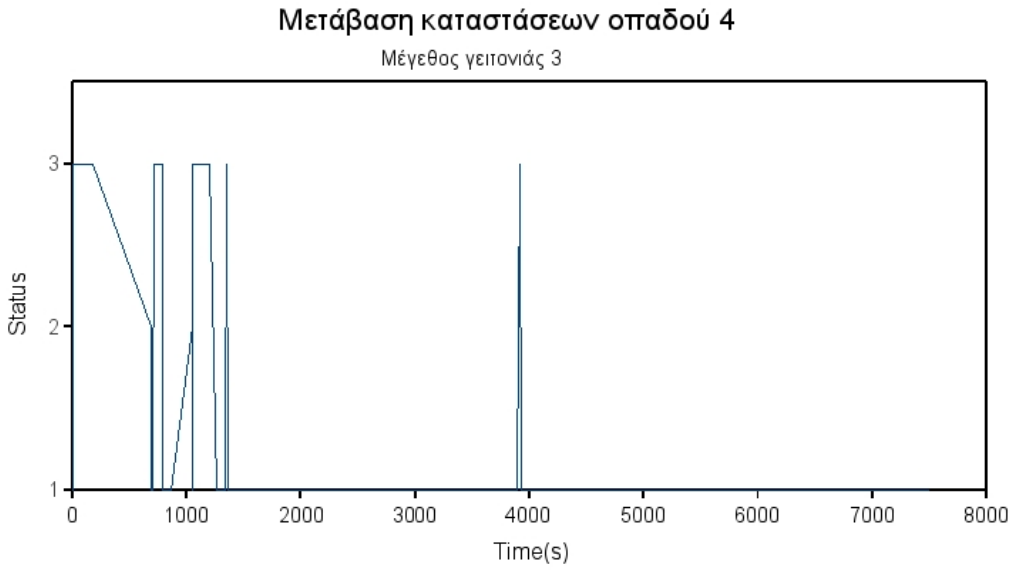


ii)

Σχήμα 22: Μέγεθος γειτονίας 2 - i) Απόσταση ii) Μετάβαση καταστάσεων



i)

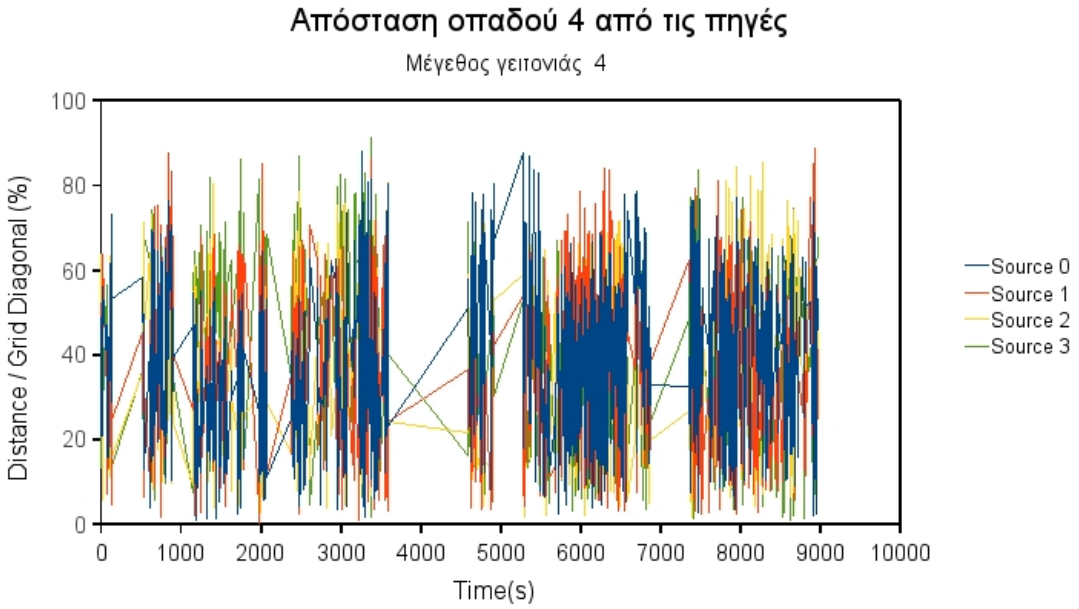


ii)

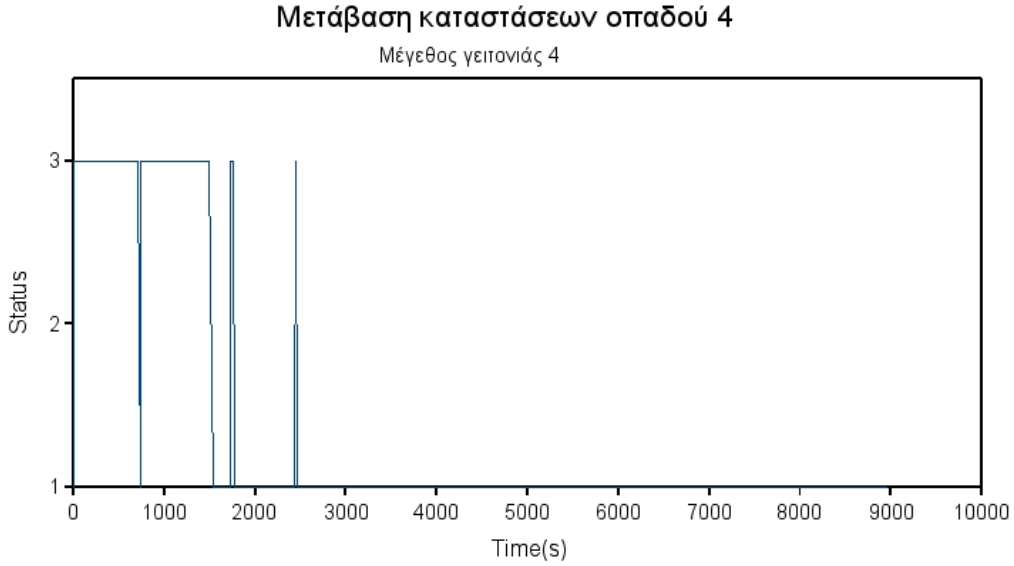
Σχήμα 23: Μέγεθος γειτονίας 3 - i) Απόσταση ii) Μετάβαση καταστάσεων

Τα κυριότερα συμπεράσματα είναι τα εξής:

1. Η διάρκεια της προσομοίωσης σε αυτό το σενάριο είναι σημαντικά μεγαλύτερη για το ίδιο ενεργειακό απόθεμα των κόμβων. Αυτό συμβαίνει γιατί στην περίπτωση που οι κόμβοι οπαδοί κινούνται βάσει τυχαιότητας βρίσκονται πολύ πιο συχνά σε περιόδους διακοπής επικοινωνίας μεταξύ τους και με τις πηγές (το ενεργειακό απόθεμα μειώνεται μόνο λόγω της κίνησης σε αυτές τις περιπτώσεις) και έτσι έχουν τη δυνατότητα να αναζητούν πλαίσια για μεγαλύτερο χρονικό διάστημα. Αντίθετα με την πλοήγηση βάσει του PSO όπου οι κόμβοι οδηγούνται σε γειτονιές με φρέσκα πλαίσια μέσα από τη συνεχή επικοινωνία

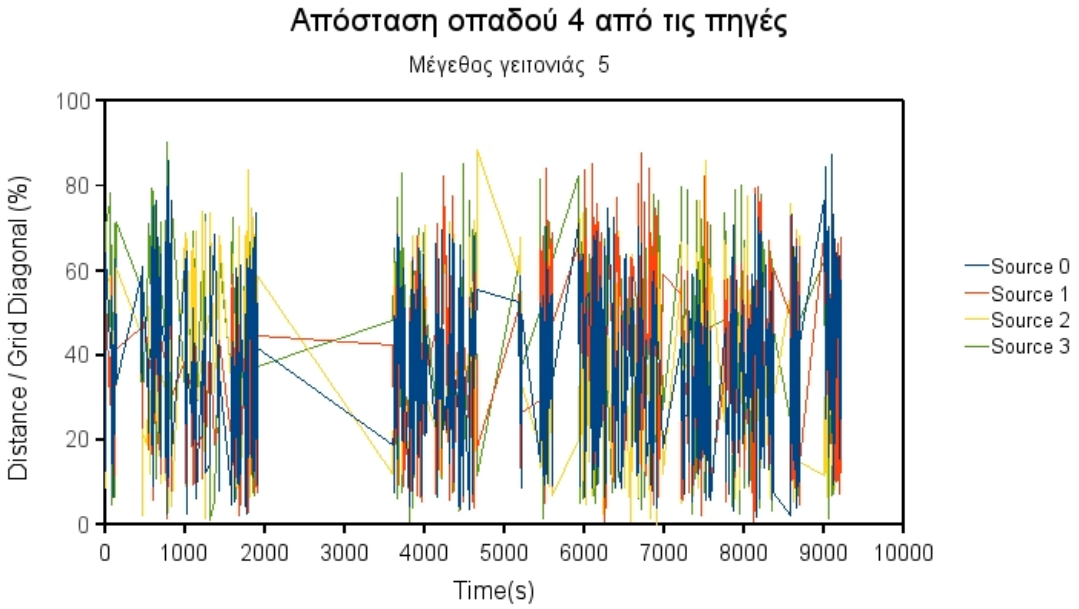


i)

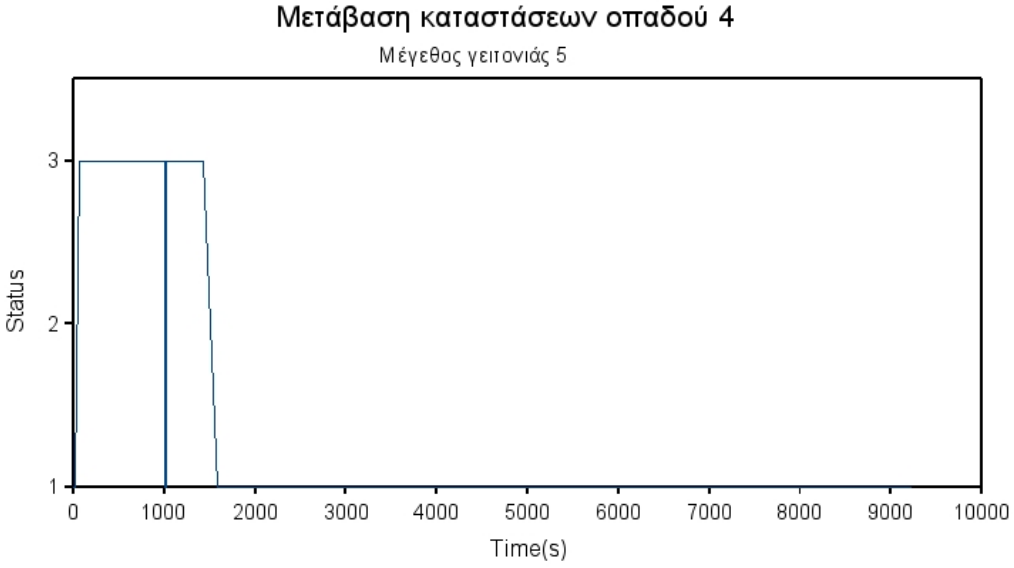


ii)

Σχήμα 24: Μέγεθος γειτονίας 4 - i) Απόσταση ii) Μετάβαση καταστάσεων



i)



ii)

Σχήμα 25: Μέγεθος γειτονίας 5 - i) Απόσταση ii) Μετάβαση καταστάσεων

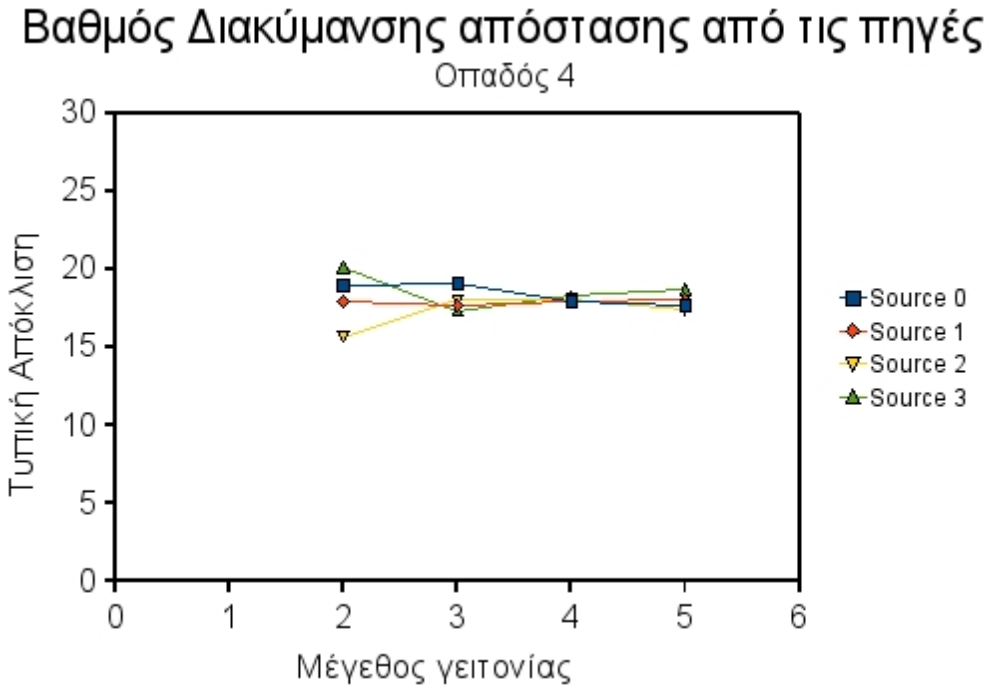
και την προσκόλλησή τους σε πηγές.

2. Όπως φαίνεται και από τα διαγράμματα αποστάσεων κόμβων-οπαδών υπάρχει μεγάλος βαθμός διακύμανσης όπως αναμενόταν σε σχέση με την περίπτωση του PSO, που δεν προδίδει την προσκόλληση κάποιων πηγών και οπαδών. Πιο συγκεκριμένα όπως φαίνεται και στο Σχήμα 30 ο βαθμός της διακύμανσης ανάμεσα στις δυο περιπτώσεις παρουσιάζει μια διαφορά της τάξης του 30%. Σε αυτό συνηγορεί και το ιστορικό κίνησης του κόμβου 4 (βλ. Σχήμα 31) όπου κατά την πλοήγηση με τον αλγόριθμο PSO φαίνεται να υπάρχουν δύο περιοχές (μια μικρότερη και μια μεγαλύτερη) με μεγάλη συγκέντρωση τροχιών που αντιστοιχούν σε περιοχές δράσης πηγών και τις οποίες φαίνεται να ακολουθεί ο οπαδός 4. Σε αντίθεση με την τυχαία πλοήγηση υπάρχει ισότροπη κατανομή τροχιών σε όλο το πεδίο όπως αναμενόταν.

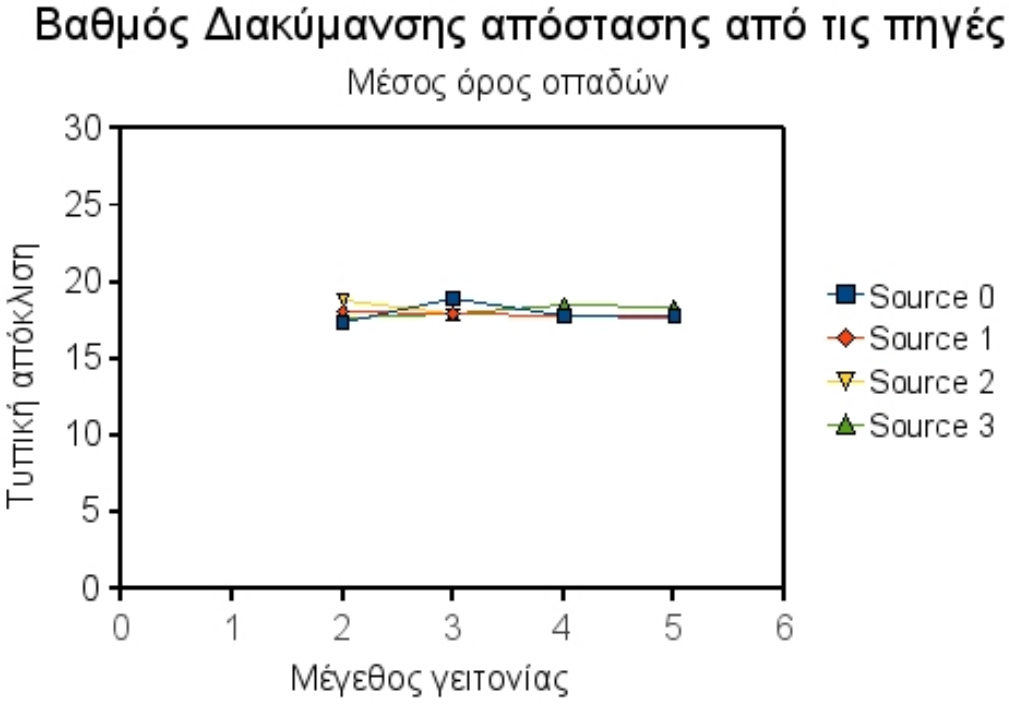
3. Όσον αφορά το ποσοστά ικανοποίησης των κόμβων όπως φαίνεται και στα Σχήματα 28 και 29, κυμαίνονται σε πολύ μεγαλύτερες τιμές για την κατάσταση έωλος συγκριτικά με την περίπτωση της πλοήγησης με τον PSO. Άρα οι κόμβοι οπαδοί μένουν ανικανοποίητοι όσον αφορά την ανάκτηση φρέσκων πλαισίων γιατί η τυχαία κίνηση τους δεν μπορεί να τους οδηγήσει με μεγάλη πιθανότητα σε γειτονιές φρέσκων πλαισίων και να παραμείνουν σε αυτές για μεγάλο χρονικό διάστημα.

4. Από τα σχήματα 21 και 29 συνάγεται ότι ο μέσος χρόνος κατά τον οποίο ένας κόμβος βρίσκεται στη συμβιβάσιμη κατάσταση είναι μεγαλύτερος στην περίπτωση του τυχαίου μοντέλου κίνησης των πηγών. Αυτό συμβαίνει γιατί η πιθανότητα να έχει ένας κόμβος μη έγκυρο πλαίσιο (απαραίτητη προϋπόθεση για τη συμβιβάσιμο κατάσταση) είναι μεγαλύτερη στην περίπτωση του τυχαίου μοντέλου.

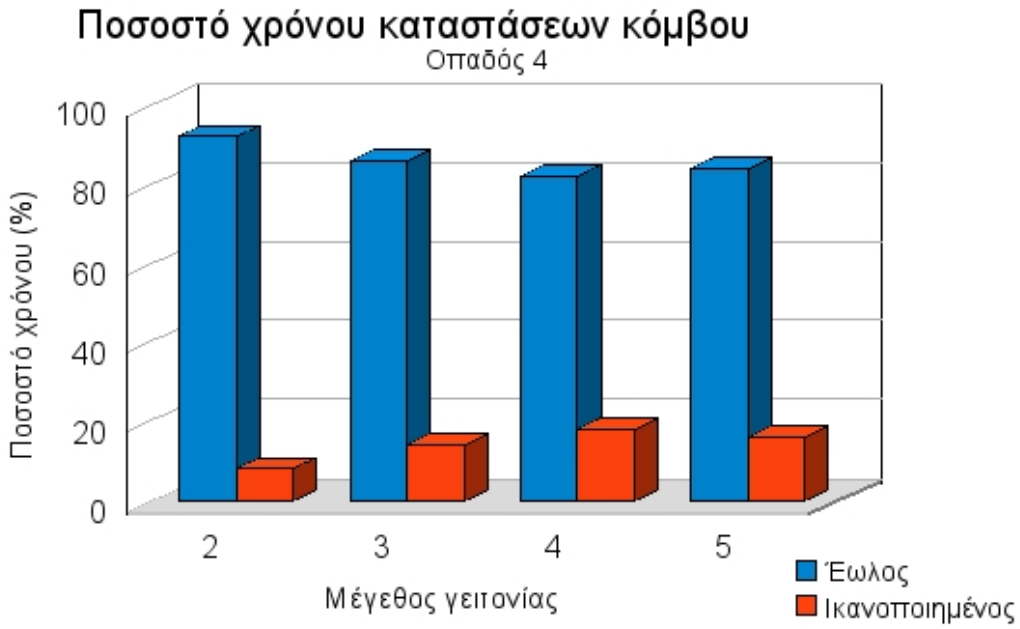
5. Τέλος πρέπει να αναφερθεί (Σχήματα 26 και 27) ότι ο βαθμός διακύμανσης παραμένει σχεδόν σταθερός για όλες τις περιπτώσεις μεγέθους γειτονίας και δείχνει ότι είναι πρακτικά ανεξάρτητος από τη γειτονιά. Αυτό μπορεί να θεωρηθεί αναμενόμενο καθώς η κίνηση και των πηγών και των οπαδών βασίζεται στο ίδιο τυχαίο μοντέλο που παρουσιάζει ισοτροπία στην κατεύθυνση της κίνησης σε αντίθεση με την περίπτωση κίνησης με τον PSO όπου η επιλογές της κίνησης είναι biased προς εκείνες τις κατευθύνσεις των γειτονιών με φρέσκα πλαίσια και εξαρτώνται και από το μέγεθος της γειτονιάς (Σχήμα 8)



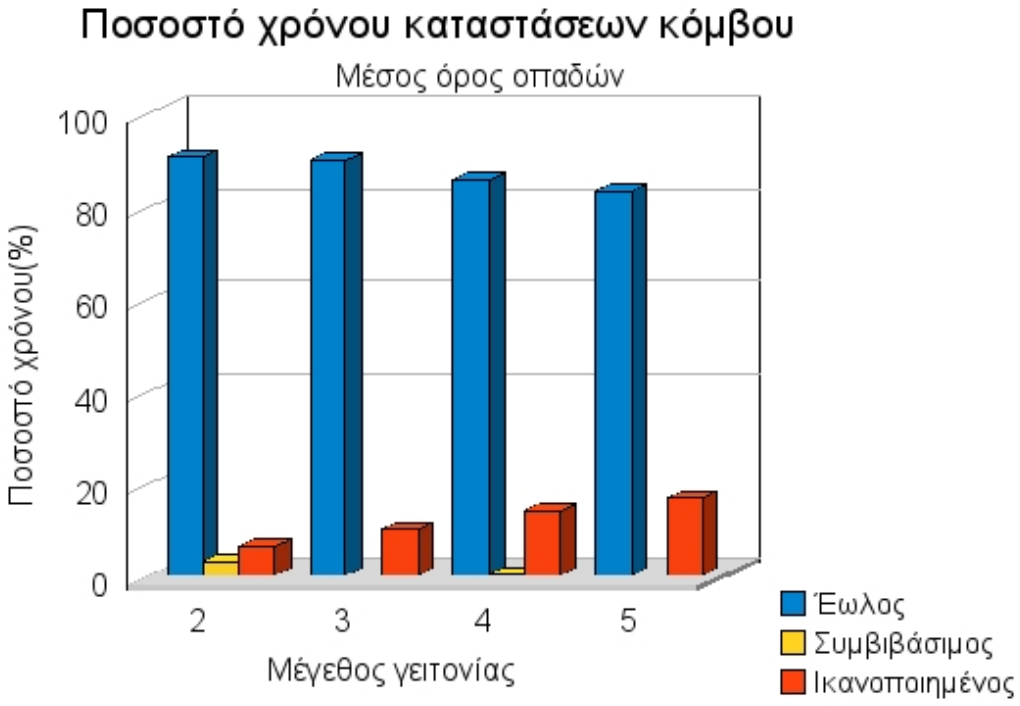
Σχήμα 26: Βαθμός Διακύμανσης απόστασης από τις πηγές για τον οπαδό 4



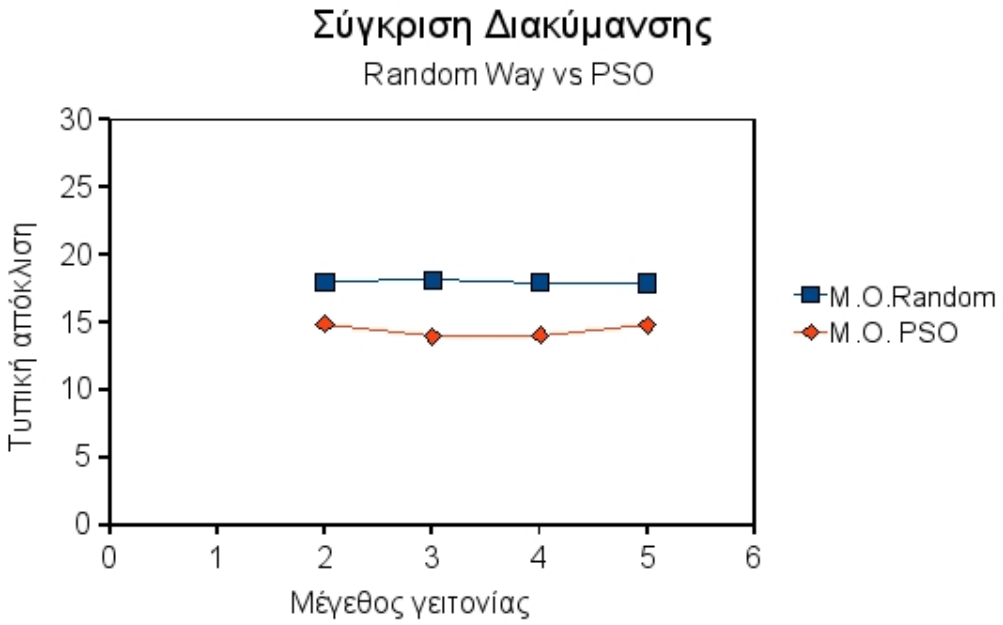
Σχήμα 27: Βαθμός Διακύμανσης απόστασης από τις πηγές. Μέσος όρος οπαδών



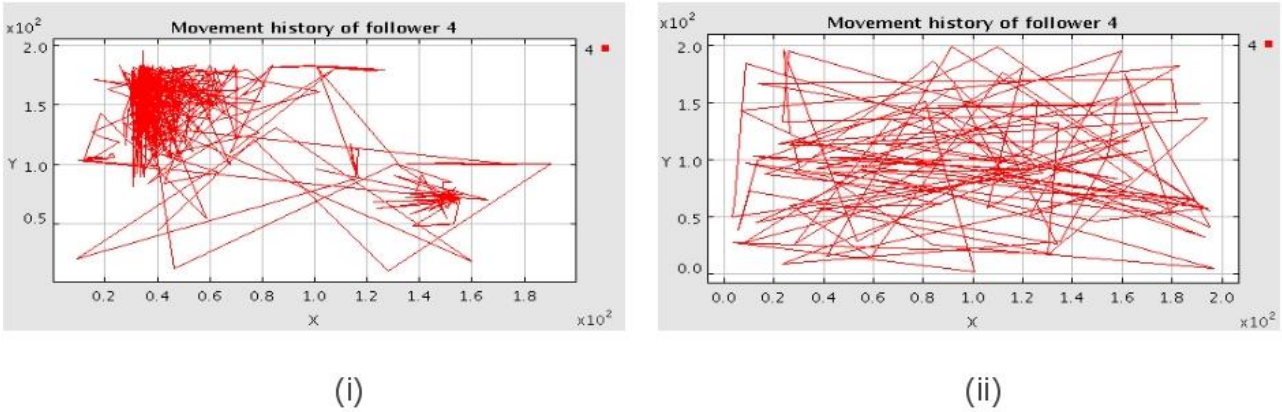
Σχήμα 28: Ποσοστό χρόνου καταστάσεων οπαδού 4



Σχήμα 29: Ποσοστό χρόνου καταστάσεων. Μέσος όρος οπαδών



Σχήμα 30: Ποσοστό χρόνου καταστάσεων. Μέσος όρος οπαδών



Σχήμα 31: Ιστορικό κίνησης κόμβου 4 i) PSO ii) RandomWay

6 Συμπεράσματα

Έχοντας σαν βάση τον αλγόριθμο Particle Swarm Optimization (PSO) που αποτελεί βιο-μιμητική προσέγγιση του τρόπου με τον οποίο «σμήνη πουλιών» (bird flocks) ανακαλύπτουν την τροφή τους και χρησιμοποιείται για την επίλυση πολύπλοκων προβλημάτων βελτιστοποίησης, θεμελιώθηκε και υλοποιήθηκε ένας μηχανισμός ανακάλυψης επικαιροποιημένης πληροφορίας πλαισίου βασιζόμενος στη «συνεργατική συμπεριφορά» κάποιων κόμβων που ονομάζονται «οπαδοί», σε περιβάλλοντα κινητού υπολογισμού (δίκτυο κινητών τηλεφώνων, ρομποτικές συσκευές, επισκέπτες ενός μουσείου εφοδιασμένοι με PDA's, σμήνη UUV κλπ) τα οποία αποτελούνται από πηγές που κατέχουν/μετρούν την πληροφορία πλαισίου και κινούνται τυχαία στο πεδίο της κινητής εφαρμογής και από κόμβους - οπαδούς οι οποίοι πασχίζουν να ανακαλύψουν όσο το δυνατόν πιο «φρέσκια» πληροφορία.

Εξετάστηκε αρχικά η συμπεριφορά του μοντέλου σε σχέση με το βαθμό κινητικότητας των πηγών και πιο συγκεκριμένα για ταχύτητες πηγών 2 ,5, 10 και 50. Ο αριθμός των πηγών και των οπαδών ορίσθηκε σε 4 κα 6 αντίστοιχα για όλες τις προσομοιώσεις, ενώ το μέγεθος γειτονίας (ακτίνα δράσης των οπαδών) ορίσθηκε σε 2 που αντιστοιχεί σε πυκνότητα πλέγματος 50.2%. Η διακύμανση της απόστασης οπαδών-πηγών οδηγείται σε μεγαλύτερες τιμές με την αύξηση της κινητικότητας των πηγών για μέγιστη ταχύτητα πηγών ίση με 50 δυσκολεύοντας τη σύγκλιση οπαδών-πηγών. Παρόλα αυτά το ποσοστό του χρόνου που ένας οπαδός είναι ικανοποιημένος από την ποιότητα των πλαισίων που έχει ανακτήσει δεν χειροτερεύει με την αύξηση της κινητικότητας των πηγών.Ο λόγος έγκειται στο γεγονός ότι οι μεγάλες ταχύτητες με τις οποίες κινούνται οι πηγές επιτρέπουν την ομοιόμορφη διάχυση φρέσκιας πληροφορίας σε όλο το πλέγμα καθ'όλη τη διάρκεια της προσομοίωσης (οι πηγές κινούνται με το Random Way Model και επομένως μια τέτοια υπόθεση μπορεί να θεωρηθεί ευσταθής) με αποτέλεσμα οι οπαδοί να μπορούν να ανακτούν φρέσκα πλαίσια από τις πηγές αλλά κυρίως σε αυτές τις μεγάλες κινητικότητες από τους άλλους οπαδούς. Η βέλτιστη τιμή για το βαθμό κινητικότητας των πηγών εκτιμήθηκε σε 2.

Η συμπεριφορά του μοντέλου σε σχέση με το μέγεθος γειτονίας εξετάσθηκε επιλέγοντας τιμές 2, 3, 4 και 5 για το μέγεθος γειτονίας που αντιστοιχούν σε πυκνότητα πλέγματος 12.6, 28.3, 50.2 και 78.5%. Η κινητικότητα των πηγών επιλέχθηκε να είναι 2 σύμφωνα με τη βέλτιστη τιμή που υπολογίσθηκε. Η βέλτιστη τιμή που υπολογίσθηκε είναι 4.

Τέλος επιλέχθηκε να πλοηγηθούν οι κόμβοι-οπαδοί βάσει του τυχαίου μοντέλου κίνησης Random Way για ακτίνες δράσης 2, 3, 4 και 5 και να γίνει σύγκριση όσον αφορά την ικανότητα ανάκτησης πλαισίων σε σχέση με την πλοήγηση με τον PSO. Όπως αναμενόταν ο βαθμός διακύμανσης της απόστασης πηγών-οπαδών είναι κατά 30% μεγαλύτερος στην περίπτωση της τυχαίας κίνησης των οπαδών ενώ τα ποσοστά ικανοποίησης των κόμβων είναι χαμηλότερα για την περίπτωση της τυχαίας κίνησης.

Στον επίλογο αυτής της Διπλωματικής εργασίας θα πρέπει να αναφερθεί ότι το μοντέλο που αναπτύχθηκε για να χρησιμοποιηθεί σαν βάση ενός μηχανισμού ανακάλυψης πληροφορίας σε περιβάλλοντα κινητού υπολογισμού μπορεί να επεκταθεί στον τρισδιάσταστο χώρο και να καλύψει επίσης και πληροφορία πλαισίου με περισσότερες από μια συνιστώσες εκτός της χρονικής, προσφέροντας έτσι τη δυνατότητα για μελλοντική δουλειά στο πεδίο που πραγματεύθηκε η συγκεκριμένη Διπλωματική.

Αναφορές

[1] http://www.merriam-webster.com/dictionary/context

[2] Dey AK. Understanding and using context. Journal of Personal and Ubiquitous Computing 5(1), 2001, pp. 4-7.

[3] Schilit BN, Theimer MM. Disseminating active map information to mobile hosts. IEEE Network 8(5), 1994, pp. 22-32.

[4] Brown PJ, Bovey JD, Chen X. Context-aware applications: from the laboratory to the marketplace. IEEE Personal Communications 4(5), 1997, pp. 58-64.

[5] Schmidt A, Beigl M, Gellersen HW. There is more to context than location. Computers & Graphics Journal, Elsevier 23(6), 1999, pp. 893-902

[6] Anagnostopoulos C., Tsounis A., Hadjiefthymiades S., Context Awareness in Mobile Computing Environments, Wireless Personal Communications, vol. 42, 2006, pp. 445-464

[7] Loke S., Context Aware Pervasive Systems - Architectures for a new breed of applications Auerbach Publications 2007 USA

[8] Dorigo M., The Ant System: Optimization by a colony of cooperating agents IEEE Transactions on Systems, Man & Cybernetics-Part B, (26)1, 1996, pp. 1-13

[9] Bonabeau E., Dorigo M., Theraulaz G., Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, Oxford, 1999.

[10] Dorigo M., Gambardella L.M., Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1), 1997, pp. 53–66.

[11] Dorigo M., Gambardella L.M.,, Ant colonies for the traveling salesman problem, BioSystems 43, 1997, pp. 73–81.

[12] Stützle T., Hoos H., MAX-MIN Ant System, Future Generation Computer Systems 16, 2000, pp. 889-914.

[13] Kennedy J., Eberhart R., Particle swarm optimization, Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ, 1995, pp. 1942–1948

[14] Clerc M., Kennedy J., The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space, IEEE Transactions on Evolutionary Computation, vol. 6, 2002, pp. 58-73

[15] Altshuler Y., Yanovsky V., Wagner I.A., Bruckstein A.M. Swarm Intelligence – Searchers, Cleaners and Hunters, Swarm Intelligent Systems, Eds. N. Nedjah, L. de Macedo Mourele, 2006, pp. 93-132

[16] http://gecco.org.chemie.uni-frankfurt.de/PsoVis/applet.html

[17] Di Caro G., Ducatelle F., and Gambardella L.M. AntHocNet: An adaptive nature-inspired algorithm for routing in Mobile Ad Hoc networks. Technical Report No. IDSIA-27-04-2004, Manno, Switzerland

[18] Gunes M., Sorges U., Bouazizi I, ARA – The Ant-colony based routing algorithm for MANETs, International workshop on Ad-hoc Networking (IWAHN 2002)

[19] Fujita K., Saito A., Matsui T., Matsuo H. An adaptive ant-based routing algorithm used routing history in dynamic networks. In Proc. of the 4th Asia-Pacific Conf. on Simulated Evolution and Learning, 2002.

[20] Matsuo H., Mori K. Accelerated ants routing in dynamic networks. In 2nd Int. Conf. on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2001.

[21] Camara D., Loureiro A.A.F. Gps/ant-like routing in ad hoc networks. Telecommunication Systems, 18(1–3), 2001, pp. 85–100

[22] Marwaha S., Tham C. K., Srinivasan D. Mobile agents based routing protocol for mobile ad hoc networks. In Proc. of IEEE Globecom, 2002.

[23] Camilo T., Carreto C., Sa Silva J., Boavida F. An energy efficient ant-based routing algorithm for wireless sensor networks, Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006, Brussels Belgium, September 2006, LNCS 4150, pp. 49-59

[24] Zhang X., Xu W. QoS Based Routing in Wireless Sensor Network with Particle Swarm Optimization. Agent Computing and Mutlti-Agent Systems, Springer, 2006, pp. 602-607

[25] Deneubourg J., Goss S.,Sandini G., Ferrari F., Dario P. Self-Organizing Collection and Transport of Objects in Unpredictable Environments, Symposium on Flexible Automation, 1990, pp. 1093-1098.

[26] Brueckner S.A., Van Dyke Parunak H. Self-organizing MANET management Engineering Self-organising Systems, 2004, Springer, pp. 20-35

[27] Liang Y., Yu H. PSO-Based Energy Efficient Gathering in Sensor Networks Mobile Ad-hoc and Sensor Networks, 2005, Springer, pp. 362-369

[28] Min H., Zhu J., Zheng X. Obstacle avoidance with multi-objective optimization by PSO in dynamic environment Proc. Int. Conf. Machine Learning and Cybernetics, Vol. 5, 2005, pp. 2950-2956.

[29] Drogoul A., Ferber J. From Tom Thumb to the Dockers: Some Experiments With Foraging Robots, 2nd Int. Conference on Simulation of Adaptive Behavior, 1992, pp. 451-459.

[30] Parker L.E. ALLIANCE: An Architecture for Fault-Tolerant Multi-Robot Cooperation, IEEE Transactions on Robotics and Automation, 14(2), 1998, pp. 220-240.

[31] LaVille S.M., Lin D., Guibas L.J.,Latombe J.C., Motwani R. Finding an Unpredictable Target in a Workspace with Objects, IEEE Int. Conference on Robotics and Automation, 1997, pp. 737-742.

[32] Foo J.L., Knutzon J.S., Oliver J.H., Winer E.H. Three-Dimensional multi-objective path planning of unammaned aerial vehicles using partcile swarm optimization 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 23 - 26 April 2007, Honolulu, Hawaii

[33] Banks A., Vincent J., Phalp K. Particle Swarm Guidance System for Autonomous Unmanned Aerial Vehicles in an Air Defence Role Journal of Navigation, vol. 61, no1, 2008, pp. 9-29

[34] Guo B., Liang X., Wang B., Wan L. Sigmoid surface control for mini underwater vehicles by improved particle swarm optimization Proceedings of the 2007 IEEE International Conference on Robotics and Biomimetics, ROBIO 2007, Sanya China , 2007, pp.1200-1205

[35] Pasino K.,. Polycarpou M, Jacques D., Pachter M.,Liu Y.,Yang Y., Flint M., Baum M. Cooperative Control for Autonomous Air Vehicles, Cooperative Control and Optimization, R. Murphy, P. Pardalos (eds.) Kluwer Academics Publishers, Boston, 2002.

[36] Polycarpou M., Yang Y., Pasino K. A Cooperative Search Framework for Distributed Agents, IEEE Int. Symposium on Intelligent Control, 2001, pp. 1-6.

[37] Stone L.D, Theory of Optimal Search, Academic Press, New York, 1975.

[38] Vincent P., Rubin I., A Framework and Analysis for Cooperative Search Using UAV Swarms, ACM Symposium on Applied Computing, 2004.

[39] http://sites.google.com/site/jsimofficial/

[40] Αναγνωστόπουλος Χ. Μοντελοποίηση και Αρχιτεκτονική Συστημάτων Κινητού Υπολογισμού, Διδακτορική Διατριβή, Ε.Κ.Π.Α (2008)

[41] Athanassoulis M., Alagiannis I., Hadjiefthymiades S. Energy Efficiency in Wireless Sensor Networks. A Utility Based Architecture, European Wireless 2007, Paris France, April 1-4, 2007

ΠΑΡΑΡΤΗΜΑ Α: Σενάριο TCL

```
puts "Particle swarm in wsn"

set ns 4; # number of sources
set nf 6; # number of followers
set maxgx 100.0; # max x coordinate of sim area
set maxgy 100.0; #max y coordinate of sim area
set mingx 0.0; # min x coordinate of sim area
set mingy 0.0; # min y coordinate of sim area
set gsizex 20.0; # grid zize along x axis
set gsizey 20.0; # grid size along y axis
set nbgrid 3; # define as neighbours, nodes that are within nbgrid grids
set seeds 1111; # seed for uniform class generating random initial node positions for sources
set seedf 1111; # seed for uniform class generating random initial node positions for followers
#set tx_range 0.00085872; # wireless card range of 40m
#set tx_range 0.007214; # wireless card range of 100m
set tx_range 0.2818; # wireless card range of 250m
set cons_tx 0.30; # amount of energy consuption during tx
set cons_rx 0.25; # amount of energy consuption during rx
set cons_idle 0.0; # amount of energy consuption when node is idle
set cons_off 0.0; # amount of energy consuption when node powers off
set cons_mov 0.2; # amount of energy consumtion at each movement
set b_int 2.0; # beacon interval
set atim 1.0; # atim window size
set src_vel 2.0; #source node velocity
set fol_vel 5.00; #follower node velocity

mkdir drcl.comp.Component sim1; # Root component
cd sim1
# Creating source nodes
for {set i 0} {$i < $ns} {incr i} {

    puts "Creating source node $i"
    mkdir drcl.comp.Component s$i
    cd s$i

    mkdir drcl.net.traffic.SimpleTraceTCP trci

    ! trci setReader [java::new java.io.FileReader [concat trace$i]]
    set PD [mkdir drcl.inet.core.PktDispatcher dispatch]
    mkdir drcl.inet.core.queue.FIFO fifo
    mkdir drcl.inet.mac.FreeSpaceModel free
    mkdir drcl.inet.mac.MobilityModel mob
    mkdir drcl.inet.mac.WirelessPhy phy

    for {set j $ns} {$j < [expr $ns + $nf]} {incr j} {
    mkdir drcl.inet.transport.TCP tcp$j
    }

    set mc [mkdir drcl.inet.mac.Mac_802_11 mac]
    mkdir drcl.inet.mac.LL ll
    set ID [mkdir drcl.inet.core.Identity id]
    set RT [mkdir drcl.inet.core.RT rt]
    $PD bind $ID
    $PD bind $RT

    ! fifo setCapacity 40
```

```
! fifo setMode ”packet”
! dispatch setRouteBackEnabled true

mkdir drcl.inet.protocol.aodv.AODV aodv
connect −c aodv/down@ −and dispatch/103@up
connect aodv/.service_rt@ −and rt/.service_rt@
connect aodv/.service_id@ −and id/.service_id@
connect aodv/.ucastquery@ −and dispatch/.ucastquery@
connect mac/.linkbroken@ −and aodv/.linkbroken@
! aodv enable_link_detection

connect mac/.linklayer@ −and ll/.mac@
connect mac/up@ −and fifo/output@
connect mac/down@ −and phy/up@
connect ll/.mac@ −and mac/.linklayer@
connect −c ll/up@ −and dispatch/0@down
connect ll/down@ −and fifo/up@
for {set j $ns} {$j < [expr $ns + $nf]} {incr j} {
connect −c trci/down@ −and tcp$j/up@
connect −c tcp$j/down@ −and dispatch/17@up
}
connect −c dispatch/0@down −and ll/up@
connect fifo/output@ −and mac/up@
connect free/.query@ −and phy/.propagation@
connect mob/.query@ −and phy/.mobility@
connect phy/up@ −and mac/down@
connect phy/.propagation@ −and free/.query@
connect phy/.mobility@ −and mob/.query@

for {set j $ns} {$j < [expr $ns + $nf]} {incr j} {
! tcp$j setPeer $j
! tcp$j setMSS 512
puts ”Peering with f$j”
}

mkdir drcl.net.tool.TrafficMonitor tm

connect −c dispatch/0@down −to tm/in@

set rndx [java::new java.util.Random $seeds]
set rndy [java::new java.util.Random $seeds]
set rndx0 [$rndx nextDouble]
set rndy0 [$rndy nextDouble]
set maxgxx [expr $maxgx / 1.0]
set maxgyy [expr $maxgy / 1.0]
set rndx1 [expr $rndx0 * [expr $maxgxx − $mingx]]
set rndy1 [expr $rndy0 * [expr $maxgyy − $mingy]]

! mob setTopologyParameters $maxgx $maxgy $mingx $mingy $gsizex $gsizey 0.0
! mob setPosition $src_vel $rndx1 $rndy1 0.0
! mac disable_PSM
#! mac enable_PSM
connect mac/.energy@ −and phy/.energy@
set nid $i
! ll setAddresses $nid $nid
! mac setMacAddress $nid
! phy setNid $nid
! mob setNid $nid
! id setDefaultID $nid
```

```
    ! mac setRTSThreshold 0
    ! mac setBeaconInterval $b_int
    ! mac setATIMWindow $atim

    set em [! phy getEnergyModel]
    $em setEnergyConsumption $tx_range $cons_tx $cons_rx $cons_idle $cons_off $cons_mov
    java::new drcl.inet.mac.TSFTimer $mc 0

    mkdir drcl.inet.mac.ARP arp
    connect ll/.arp@ −and arp/.arp@
    ! arp setAddresses $nid $nid
    ! arp setBypassARP true

    cd ..
}

# Creating follower nodes
for {set i $ns} {$i < [expr $ns + $nf]} {incr i} {

    puts ”Creating follower node $i”
    mkdir drcl.comp.Component f$i
    cd f$i

    set mc [mkdir drcl.inet.mac.Mac_802_11 mac]
    mkdir drcl.inet.mac.LL ll
    set PD [mkdir drcl.inet.core.PktDispatcher dispatch]
    mkdir drcl.inet.core.queue.FIFO fifo
    mkdir drcl.inet.mac.FreeSpaceModel free
    mkdir drcl.inet.mac.AdvancedMobilityModel mob
    mkdir drcl.inet.mac.WirelessPhy phy
    mkdir drcl.inet.transport.TCPSink sink
    mkdir drcl.inet.application.BulkSink bsink

    set ID [mkdir drcl.inet.core.Identity id]
    set RT [mkdir drcl.inet.core.RT rt]
    $PD bind $ID
    $PD bind $RT

    ! fifo setCapacity 40
    ! fifo setMode ”packet”
    ! dispatch setRouteBackEnabled true
    set nid $i
    ! ll setAddresses $nid $nid
    ! mac setMacAddress $nid
    ! phy setNid $nid
    ! mob setNid $nid
    ! id setDefaultID $nid

    connect mac/.linklayer@ −and ll/.mac@
    connect mac/up@ −and fifo/output@
    connect mac/down@ −and phy/up@
    connect ll/.mac@ −and mac/.linklayer@
    connect −c ll/up@ −and dispatch/0@down
    connect ll/down@ −and fifo/up@
    connect −c dispatch/0@down −and ll/up@
    connect fifo/output@ −and mac/up@
    connect free/.query@ −and phy/.propagation@
    connect mob/.query@ −and phy/.mobility@
    connect phy/up@ −and mac/down@
```

```
  connect phy/.propagation@ −and free/.query@
  connect phy/.mobility@ −and mob/.query@
  connect −c sink/down@ −and dispatch/17@up
  connect bsink/down@ −and sink/up@
  connect −c dispatch/17@up −and mob/17@up
  mkdir drcl.inet.protocol.aodv.AODV aodv
  connect −c aodv/down@ −and dispatch/103@up
  connect aodv/.service_rt@ −and rt/.service_rt@
  connect aodv/.service_id@ −and id/.service_id@
  connect aodv/.ucastquery@ −and dispatch/.ucastquery@
  connect mac/.linkbroken@ −and aodv/.linkbroken@
  ! aodv enable_link_detection

mkdir drcl.net.tool.TrafficMonitor tm

connect −c dispatch/17@up −to tm/in@

set rndx [java::new java.util.Random $seedf]
set rndy [java::new java.util.Random $seedf]
set rndx0 [$rndx nextDouble]
set rndy0 [$rndy nextDouble]
set maxgxx [expr $maxgx / 1.0]
set maxgyy [expr $maxgy / 1.0]
set rndx1 [expr $rndx0 * [expr $maxgxx − $mingx]]
set rndy1 [expr $rndy0 * [expr $maxgyy − $mingy]]

  ! mob setTopologyParameters $maxgx $maxgy $mingx $mingy $gsizex $gsizey 0.0
  ! mob setPosition $fol_vel $rndx1 $rndy1 0.0
  ! mac disable_PSM
  #! mac enable_PSM
connect mac/.energy@ −and phy/.energy@
  ! mac setRTSThreshold 0
  ! mac setBeaconInterval $b_int
  ! mac setATIMWindow $atim

  set em [! phy getEnergyModel]
  $em setEnergyConsumption $tx_range $cons_tx $cons_rx $cons_idle $cons_off $cons_mov

java::new drcl.inet.mac.TSFTimer $mc 0

mkdir drcl.inet.mac.ARP arp
  connect ll/.arp@ −and arp/.arp@
  ! arp setAddresses $nid $nid
  ! arp setBypassARP true
  connect mob/.movhis@ −and phy/.movhis@
  ! mob setThreshold 2.0
  ! mob setValidContextThr 0.2
  ! mob setPSO 1
  ! mob setNoSrc $ns
  ! mob setInitFactors 1.0 1.0 0
  ! mob setPSOFactors 0.8 0.2
  ! mob setnGrids $nbgrid
  ! mob setUtilityFunctionFactors 0.8 0.2 0.2
  ! mob setDset1 $i

  set file [mkdir drcl.comp.io.FileComponent fl]
  $file open out$i
  connect −c mob/.out@ −to $file/in@
```

```
    set file1 [mkdir drcl.comp.io.FileComponent fl1]
    $file1 open status$i
    connect −c mob/.status@ −to $file1/in@

    cd ..
}
# Creating Channel and Node Tracking
    puts "Creating Channel and Node Tracking"
    mkdir drcl.inet.mac.Channel Chn
    mkdir drcl.inet.mac.NodePositionTracker npt
    ! Chn setCapacity [expr $ns + $nf]
    connect Chn/.tracker@ −and npt/.channel@

    puts "Attaching source nodes to the Channel"

for {set i 0} {$i < $ns} {incr i} {
    connect /sim1/s$i/mob/.report@ −and /sim1/npt/.node@
    connect /sim1/s$i/phy/down@ −to /sim1/Chn/.node@
    ! Chn attachPort $i [! /sim1/s$i/phy getPort .channel]


}
   puts "Attaching follower nodes to the Channel"
for {set i $ns} {$i < [expr $ns + $nf]} {incr i} {

    connect /sim1/f$i/mob/.report@ −and /sim1/npt/.node@
    connect /sim1/f$i/phy/down@ −to /sim1/Chn/.node@
    connect /sim1/f$i/mob/.tracker@ −and npt/.channel@
    connect /sim1/f$i/mob/.valndx@ −and npt/.valndx@

    #! Chn attachPort [expr $m + $i + 1] [! /sim1/f$i/phy getPort .channel]
    ! Chn attachPort $i [! /sim1/f$i/phy getPort .channel]
}

    ! npt setGrid $maxgx $mingx $maxgy $mingy $gsizex $gsizey
    ! npt setNoSrc $ns
    ! Chn setnGrids $nbgrid

    mkdir drcl.comp.tool.Plotter plot


 set i 5


  ! plot setYLabel 1 "Distance / Grid Diagonal %"
  ! plot setTitle 1 [concat Source−Follower$i distance]
  ! /sim1/f$i/mob setDset1 $i

 connect −c /sim1/f$i/mob/.out@ −to plot/0@1
 connect −c /sim1/f$i/mob/.status@ −to plot/0@2
    ! plot setTitle 2 "Status history of follower $i"
    ! plot setXRange 2 0 5
    ! plot setYRange 2 0 5
    ! plot setXLabel 2 "Time"
    ! plot setYLabel 2 "Status"


    for {set i $ns} {$i < [expr $ns + $nf]} {incr i} {
```

```
  connect −c /sim1/f$i/mob/.plotpos@ −to plot/0@$i
  ! plot setTitle $i ”Movement history of follower $i”
  ! plot setXRange $i $mingx $maxgx
  ! plot setYRange $i $mingy $maxgy
  ! plot setXLabel $i ”X”
  ! plot setYLabel $i ”Y”
  }

  set sim [attach_simulator .]
  run .
rt . stop 10000
```

ΠΑΡΑΡΤΗΜΑ B: Java Code listings

AdvancedMobilityModel

```
/* Original MobilityModel modified by MS Nikitidis March 2009. Renamed to AdvancedMobilityModel
    Particle Swarm Optimization algorithm implemented in addition to random walk and trajectory*/

// @(#)MobilityModel.java 1/2004
// Copyright (c) 1998−2004, Distributed Real−time Computing Lab (DRCL)
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are met:
//
// 1. Redistributions of source code must retain the above copyright notice,
// this list of conditions and the following disclaimer.
// 2. Redistributions in binary form must reproduce the above copyright notice,
// this list of conditions and the following disclaimer in the documentation
// and/or other materials provided with the distribution.
// 3. Neither the name of "DRCL" nor the names of its contributors may be used
// to endorse or promote products derived from this software without specific
// prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
// AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
// ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR
// ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
// DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
// SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
// CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
// OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


package drcl.inet.mac;

import drcl.comp.*;
import drcl.inet.data.*;
import drcl.net.*;
import drcl.inet.transport.*;/*msn 19−11−2008*/
import drcl.inet.*; /*msn 19−11−2008*/
import drcl.data.*;/*msn 19−11−08*/

/**
* This class simulates the movement of the mobile nodes. The position of
* the node is periodically updated and reported to the <code>NodePositionTracker</code>
* component. The coordinates can be either (longitude, lagitude, height) or (X, Y, Z).
* There are two different modes. If a trajectory array is installed, the mobility
* model assume the mobile node move from one point in the trajectory to the
* next point at a constant speed. Otherwise starting from the original
* position, the node randomly chooses a position in the simulated area as the destination,
* and moves in a straight line to that destiona point at a constant speed which is
* uniformly distributed between 0 and Max_Speed. Whenever the node arrives at the destination,
* it chooses the next position and repeats the procedure again.
*
* @see WirelessPhy
* @see NodePositionTracker
```

```
* @author Ye Ge
*/


public class AdvancedMobilityModel extends drcl.net.Module implements drcl.comp.ActiveComponent
{

  /**
   * This field is defined to keep the node Id instead of requesting such kind of
   * information from other components which may incur extra simulation overhead.
   */
  public long nid;
  public double timesmp; /*msn 19−11−2008*/
  public int ds,ds1; /*msn 23−11−2008*/
  public long movcnt; /*msn 23−11−08*/
  public double threshold_validity; //msn 26−11−08
  public double VCT; // msn 18−12−08
  public double best_cognitive_val; //msn 18−12−08
  public long[] nodeList;//msn 19−12−08
  public int PSO_flag;//msn 22−12−08
  public int NoSrc; //msn 12−1−09
  public double mean_neighbor_context; //msn 12−1−09
  public double best_neighbor_validity;//msn 12−1−09
    public double OBS_validity; //msn 12−1−09
  public double SBS_validity; //msn 12−1−09
  public double Xcf; //msn 12−1−09
  public double Ycf; //msn 12−1−09
    public double Xsf; //msn 12−1−09
  public double Ysf; //msn 12−1−09
    public double c1, c2; //msn 18−1−09
    public int nGrids; // msn 18−1−09
    public double MyContext_validity; //msn 18−1−09
    public double Xsrc[] = new double[NoSrc];//msn 25−1−09
    public double Ysrc[] = new double [NoSrc];//msn 25−1−09
    public int status; //msn 26−1−09
    public double best_neighbor_context;
    public double SBS_context;
    public double OBS_context;
    public int start_sim;
    public double Xsrcp[] = new double[20];//msn 25−1−09
    public double Ysrcp[] = new double [20];//msn 25−1−09
    public double Xfol[] = new double[NoSrc];//msn 25−1−09
    public double Yfol[] = new double [NoSrc];//msn 25−1−09
    public double best_SBS_X,best_SBS_Y;
    double m1,m2,UF_thres;



  /**
  * Sets the nid.
  *
  *@param nid_ the id of the mobile node.
  */
  public void setNid(long nid_) { nid = nid_; }

  /** Gets the nid. */
  public long getNid() { return nid; }

  /* configurate the ports */
```

```
private static final String REPORT_PORT_ID = ".report"; // connect to the mobile nodes' wirelessphy
private static final String CONFIG_PORT_ID = ".config";
private static final String QUERY_PORT_ID = ".query"; // connect to the channel component
private static final String OUT_PORT_ID = ".out"; /*msn 19-11-08*/
private static final String PLOT_POS_ID = ".plotpos";/*msn 19-11-08*/
private static final String MOV_HIST_ID = ".movhis"; /*msn 23-11-08*/
private static final String TRACKER_PORT_ID = ".tracker"; //msn 19-12-08
private static final String VALINDEX_PORT_ID = ".valndx"; //msn 9-1-09
  private static final String STATUS_PORT_ID = ".status"; //msn 13-2-09

  /** the port which should be connected to the mobility tracker. */
protected Port reportPort = addPort(REPORT_PORT_ID, false);

/** the config port */
protected Port configPort = addPort(CONFIG_PORT_ID, false);

/** the query port which should be connected to the WireleePhy component. */
protected Port queryPort = addPort(QUERY_PORT_ID, false);
protected Port outPort = addPort(OUT_PORT_ID, false);/*msn 19-11-08*/ // parameters same as those in
      NodePositionTracker
protected Port plotPort = addPort(PLOT_POS_ID, false);/*msn 19-11-08*/
protected Port movhisPort = addPort(MOV_HIST_ID,false);/*msn 23-11-08*/
protected Port trackPort = addPort(TRACKER_PORT_ID,false);//msn 19-12-08
protected Port valndxPort = addPort(VALINDEX_PORT_ID,false);//msn 9-1-09
  protected Port statusPort = addPort(STATUS_PORT_ID,false); //msn 13-2-09

  /** The largest x coordinate value of the simulated area */
public double maxX;
/** The largest y coordinate value of the simulated area */
public double maxY;
/** The largest z coordinate value of the simulated area */
public double maxZ;
/** The smallest x coordinate value of the simulated area */
public double minX;
/** The smallest y coordinate value of the simulated area */
public double minY;
/** The smallest z coordinate value of the simulated area */
public double minZ;
/** The grid size along X-axle */
public double dX;
/** The grid size along Y-axle */
public double dY;
/** The grid size along Z-axle */
public double dZ;

/** The x coordinator of the node's current position */
public double X;
/** The y coordinator of the node's current position */
public double Y;
/** The z coordinator of the node's current position */
public double Z; // current position


/** The x coordinate of the node's previous position */
public double X0;
/** The y coordinate of the node's previous position */
public double Y0;
/** the z coordinate of the node's previous position */
public double Z0; // position at the last update time
```

```
/** The x coordinate of the destination position */
public double destX;
/** The y coordinate of the destination position */
public double destY;
/** The z coordinate of the destination position */
public double destZ; // destination position

/** The moving speed along the x−axle */
public double sx;
/** The moving speed along the y−axle */
public double sy;
/** The moving speed along the z−axle */
public double sz;
/** The moving speed */
public double speed; // (sx, sy, sz) is the normalized direction of the movement

/** The maximal speed for generating a random speed heading to the next random direction */
public double Max_Speed; // Max_Speed for generating a random speed heading to the next random direction

ACATimer reportPositionTimer; // the object to keep the reference to the ACATimer so that we can cancel it if
      necessary
ACATimer changeDestinationTimer;

/** The last time (X,Y,Z) is updated */
private double positionUpdateTime; // the last time (X,Y,Z) is updated
/** The interval between automatic position updates */
private double positionReportPeriod; // automatic position update period time
                        // position update may also be triggered by the query
/** The time period between two destination points */
double changeDestinationPeriod; // the time between two destination points

/** The index of the current starting position in the trajectory array. */
int nCurrentTraj; // index of the current starting position

/**
* Number of points on the trajectory.
*/
int nTrajectory;

/**
* An two dimensional array represents the trajectory of the mobile node.
*/
double[][] trajectory; // arrys of ( time, longitude, latitude, altitude )


/**
* The random number generator. The default seed is set to 11111.
*/
static java.util.Random rand = new java.util.Random(11111);

/** Installs a trajectory array. */
public void installTrajectory(double[][] trajectory_) { // the trajectory point array derived from SDF file
   trajectory = trajectory_; // set the trajectory ( arryr of states in SDF file )
   nTrajectory = trajectory.length;
   nCurrentTraj = −1;
}

/**
* Constructor.
```

```
 *
 *@param nid_ the id of the mobile node.
 */
public AdvancedMobilityModel(long nid_) {
   super();
   nid = nid_;
   X = 0.0; Y = 0.0; Z = 0.0;
   X0 = 0.0; Y0 = 0.0; Z0 = 0.0;
   sx = 0.0; sy = 0.0; sz = 0.0;
   speed = 0.0;
   queryPort.setType(drcl.comp.Port.PortType_SERVER);
   positionReportPeriod = 1.0;

   removeDefaultDownPort();
   removeDefaultUpPort();

}

/**
 * Constructor.
 */
public AdvancedMobilityModel( ) {
   super();
   nid = -1;
   X = 0.0; Y = 0.0; Z = 0.0;
   X0 = 0.0; Y0 = 0.0; Z0 = 0.0;
   sx = 0.0; sy = 0.0; sz = 0.0;
   speed = 0.0;
   queryPort.setType(drcl.comp.Port.PortType_SERVER);
   positionReportPeriod = 1.0;

   removeDefaultDownPort();
   removeDefaultUpPort();

}

/**
 * Set topology parameters
 *
 * @param maxX_ largest x coordinate value of the simulated area
 * @param minX_ smallest x coordinate value of the simulated area
 * @param maxY_ largest y coordinate value of the simulated area
 * @param minY_ smallest y coordinate value of the simulated area
 * @param dX_ grid size along X-axle
 * @param dY_ grid size along Y-axle
 * @param dZ_ grid size along Z-axle
 */
public void setTopologyParameters(double maxX_, double maxY_, double minX_, double minY_, double dX_,
        double dY_, double dZ_) {
   maxX = maxX_; maxY = maxY_; maxZ = 10.0;
   minX = minX_; minY = minY_; minZ = 0.0;
   dX = dX_; dY = dY_; dZ = dZ_;
}

/**
 * Set topology parameters
 *
 * @param maxX_ largest x coordinate value of the simulated area
 * @param minX_ smallest x coordinate value of the simulated area
```

```
* @param maxY_ largest y coordinate value of the simulated area
* @param minY_ smallest y coordinate value of the simulated area
* @param maxZ_ largest y coordinate value of the simulated area
* @param minZ_ smallest y coordinate value of the simulated area
* @param dX_ grid size along X−axle
* @param dY_ grid size along Y−axle
* @param dZ_ grid size along Z−axle
*/
public void setTopologyParameters(double maxX_, double maxY_, double maxZ_, double minX_, double
        minY_, double minZ_, double dX_, double dY_, double dZ_) {
   maxX = maxX_; maxY = maxY_; maxZ = maxZ_;
   minX = minX_; minY = minY_; minZ = minZ_;
   dX = dX_; dY = dY_; dZ = dZ_;
}

/**
* Sets the initial position of the node.
* @param Max_Speed_ − maximum moving speed
* @param X_ − initial X coordinate
* @param Y_ − initial Y coordinate
* @param Z_ − initial Z coordinate
*/
public void setPosition(double Max_Speed_, double X_, double Y_, double Z_) {
   X = X_; Y = Y_; Z = Z_;
   X0 = X_; Y0 = Y_; Z0 = Z_;
   Max_Speed = Max_Speed_;
}


//public void putPosition() {
// debug("X=" + X + " Y=" + Y + "Z = " + Z);
//}

protected void _start () {
   //randomPosition();
   //System.out.println("MobilityModel _start()");
   if ( trajectory == null ) {


        if ( PSO_flag == 1)//msn 22−12−08
   {
          status = 1;
          Xcf = X;
          Ycf = Y;
          Xsf = X;
          Ysf = Y;
        reportPSOPosition(true); // the position of the node must have been set in tcl file
        setPSODestination();
        positionUpdateTime = getTime();

                }
                else
                {
        //reportPosition(true); // the position of the node must have been set in tcl file
      reportPSOPosition(true);
            setRandomDestination();
      positionUpdateTime = getTime();

                }
```

```
        }
    else {
        setTrajectoryDestination();
        reportPosition(true);
        positionUpdateTime = getTime();
    }

}

protected void _stop() {
    if (reportPositionTimer != null) cancelTimeout(reportPositionTimer);
    if (changeDestinationTimer != null) cancelTimeout(changeDestinationTimer);
}

protected void _resume() {
    // Why comment out this?
    //if (reportPositionTimer == null) updatePosition( );
}

/**
* Sets this mobile node to a random generated position.
*/
public synchronized void setRandomPosition() {

    X = minX + rand.nextDouble() * ( maxX − minX );
    Y = minY + rand.nextDouble() * ( maxY − minY );
    Z = minZ + rand.nextDouble() * ( maxZ − minZ );

    reportPosition(true); // force it to report
    positionUpdateTime = getTime();
}

/**
* Sets a random generated destination to move to
*/
public synchronized void setRandomDestination( ) {
    double dist;

    updatePosition(); // calculate the current position

    // speed = Math.random() * Max_Speed;
    speed = rand.nextDouble() * Max_Speed;

    destX = minX + rand.nextDouble() * (maxX − minX);
    destY = minY + rand.nextDouble() * (maxY − minY);
    destZ = minZ + rand.nextDouble() * (maxZ − minZ);

    sx = destX − X;
    sy = destY − Y;
    sz = destZ − Z;

    dist = Math.sqrt((sx * sx) + (sy * sy) + (sz * sz));

    if ( destX != X || destY != Y || destZ != Z ) {
        sx = sx/dist*speed;
        sy = sy/dist*speed;
        sz = sz/dist*speed;
    }
```

```
    if ( speed > 1.0e−20 )
      changeDestinationPeriod = dist / speed;
    else
      changeDestinationPeriod = dist / 1.0e−20;

    // why cancel?
    //if ( changeDestinationTimer != null ) cancelTimeout(changeDestinationTimer);
    changeDestinationTimer = setTimeout("ChangeDestination", changeDestinationPeriod);

  }

  /**
  * Sets the next destination according to the installed trajectory.
  */
  public synchronized void setTrajectoryDestination( ) {
    double dist;
    double now = getTime();

    if ( nCurrentTraj == −1 ) { // first time calling, initiate
      nCurrentTraj = 0;
      X = trajectory[0][1]; Y = trajectory[0][2]; Z = trajectory[0][3];
      X0 = trajectory[0][1]; Y0 = trajectory[0][2]; Z0 = trajectory[0][3];
    }
    else {
      X = trajectory[nCurrentTraj][1]; Y = trajectory[nCurrentTraj][2]; Z = trajectory[nCurrentTraj][3]; // set the
          current position to the destination
    }

    positionUpdateTime = now;

    if ( nCurrentTraj < (nTrajectory−1) ) {
      sx = (trajectory[nCurrentTraj+1][1] − trajectory[nCurrentTraj][1]); // this is not direction yet
      sy = (trajectory[nCurrentTraj+1][2] − trajectory[nCurrentTraj][2]);
      sz = (trajectory[nCurrentTraj+1][3] − trajectory[nCurrentTraj][3]);
      dist = Math.sqrt(sx*sx + sy*sy + sz*sz);
      changeDestinationPeriod = (double) ( trajectory[nCurrentTraj+1][0] − trajectory[nCurrentTraj][0] ); // time
          to the next update time
      speed = dist / changeDestinationPeriod;
      sx = sx / changeDestinationPeriod; // velocity in x direction
      sy = sy / changeDestinationPeriod;
      sz = sz / changeDestinationPeriod;

      destX = trajectory[nCurrentTraj+1][1];
      destY = trajectory[nCurrentTraj+1][2];
      destZ = trajectory[nCurrentTraj+1][3];

      nCurrentTraj = nCurrentTraj + 1;

      //if ( changeDestinationTimer != null ) cancelTimeout(changeDestinationTimer);
      changeDestinationTimer = setTimeout("ChangeDestination", changeDestinationPeriod);
    }
    else {
      dist = 0.0; speed = 0.0;
      changeDestinationPeriod = 1000000000.0;
      sx = 0.0; sy = 0.0; sz = 0.0;
      // no more trajectory update timeout event scheduled
    }
  }
```

```
/**
* Calculates the current position upon timeout or query
*/
public synchronized void updatePosition() {
  double now = getTime();
  double interval = now − positionUpdateTime; // the time from now to the next timeout

  if ( interval == 0.0 ) return;

  X = X + sx * interval;
  Y = Y + sy * interval;
  Z = Z + sz * interval;

  // adjust the position if it exceed the destination point
  if ((sx > 0 && X > destX) || (sx < 0 && X < destX)) X = destX;
  if ((sy > 0 && Y > destY) || (sy < 0 && Y < destY)) Y = destY;
  if ((sz > 0 && Z > destZ) || (sz < 0 && Z < destZ)) Z = destZ;


  positionUpdateTime = now;


  if (X != X0 && Y != Y0) movcnt = movcnt + 1; /*msn 23−11−08*/
}

/**
* Sends position report to the <code>NodePositionTracker</code> component.
* If falseReport is set to true, the report is generated immediately.
*/
public synchronized void reportPosition(boolean forcedReport) {
  if (forcedReport == true || (X−X0) >= dX || (X0−X) >= dX || (Y−Y0) >= dY || (Y0−Y) >= dY || (Z−Z0) >=
        dZ || (Z0−Z) >= dZ ) { // if not forced report, report only if the change of position has exceeded the
        boundary

          reportPort.doSending(new PositionReportContract.Message(nid, X, Y, Z, X0, Y0, Z0));

          plotPort.doSending(new XYData(nid,X,Y));/*msn 20−11−08*/
    movhisPort.doSending (movcnt); /* msn 23−11−08*/}


          if (start_sim == 1)
          getSourcePosition();


  }


  public synchronized void reportPSOPosition(boolean forcedReport) {
  if (forcedReport == true || (X−X0) >= dX || (X0−X) >= dX || (Y−Y0) >= dY || (Y0−Y) >= dY || (Z−Z0) >=
        dZ || (Z0−Z) >= dZ ) { // if not forced report, report only if the change of position has exceeded the
        boundary

          reportPort.doSending(new PositionReportContract.Message(nid, X, Y, Z, X0, Y0, Z0));
          plotPort.doSending(new XYData(nid,X,Y));/*msn 20−11−08*/
    movhisPort.doSending (movcnt); /* msn 23−11−08*/}



  // reportPort.doSending(new PositionReportContract.Message(nid,timesmp));//msn 22−12−08
```

```
    X0 = X; Y0 = Y; Z0 = Z;/*}*/


  ProcessNeighbors();//msn 19−12−08


      setNodeStatus(); // mns 26−1−09


}

/**
* Handles timeout events.
*/
protected synchronized void timeout(Object data_) {

  if ( data_.equals("ChangeDestination") ) {
    if ( trajectory == null ) {
            if (PSO_flag == 1 ){ //msn 18−1−09
                if ( context_validity(timesmp) <= 2.0*VCT && context_validity(timesmp) > 0.0 &&
                        start_sim == 1)
                {
            setPSODestination();
            reportPSOPosition(false);
                }
                else if ( context_validity(timesmp) > 2.0*VCT || context_validity(timesmp) == 0.0 ||
                        start_sim == 0)
                {
             setRandomDestination();
             reportPosition(false);
                }


            }
            else{
      setRandomDestination();
            reportPSOPosition(false);}
      //reportPosition(false);}
      // timeout is scheduled again in setRandomDestination
    }
    else {
      setTrajectoryDestination();
      reportPosition(false);
      // timeout is scheduled again in setTrajectoryDestination
    }
  }
  else if ( data_.equals("ReportPosition") ) {
    if ( trajectory == null ) {
            if (PSO_flag == 1) //msn 18−1−09
            {

                if ( context_validity(timesmp) <= 2.0*VCT && context_validity(timesmp) > 0.0 &&
                        start_sim == 1 )
                {
             updatePSOPosition();
             reportPosition(false);
             reportPositionTimer = setTimeout("ReportPosition", positionReportPeriod);
                }
```

```
                    else if ( context_validity(timesmp) > 2.0*VCT || context_validity(timesmp) == 0.0 ||
                            start_sim == 0 )
                    {
                updatePosition();
        reportPosition(false);
        reportPositionTimer = setTimeout("ReportPosition", positionReportPeriod);
                    }

                }
                else {
        updatePosition();
        //reportPosition(false);
                reportPSOPosition(false);
        reportPositionTimer = setTimeout("ReportPosition", positionReportPeriod);}
      }
      else {
        updatePosition();
        reportPosition(false);
        reportPositionTimer = setTimeout("ReportPosition", positionReportPeriod);
      }
    }

}

protected synchronized void processOther(Object data_, Port inPort_) {
    String portid_ = inPort_.getID();

    if (portid_.equals(MOV_HIST_ID)){//msn 24-11-08
            NeighborQueryContract.Message msg = ( NeighborQueryContract.Message) data_;
            if (msg.getEnergy() <= 0.0 || (UtilityFunction(msg.getEnergy()) < UF_thres ) ){
        setNid(msg.getNid());
    setPosition(0.0,X0,Y0,Z0);
            }


    return ;
    }

    if (portid_.equals(QUERY_PORT_ID))
      processQuery(data_);
    else
      super.processOther(data_, inPort_);

}


/**
* Processes the position query from <code>WirelessPhy</code> component.
*/
protected synchronized void processQuery(Object data_) {
    updatePosition();
    queryPort.doSending(new PositionReportContract.Message(X, Y, Z));

}

/**
* Sets the seed of the random number generator.
*/
public void setSeed(long seed) {
```

```
    rand.setSeed(seed);
  }

  public String info()
  {
    return "current=(" + X + "," + Y + "," + Z + ")\n"
      + " prev=(" + X0 + "," + Y0 + "," + Z0 + ")\n"
      + " speed=(" + sx + "," + sy + "," + sz + ")=" + speed + "\n"
      + " toward=(" + destX + "," + destY + "," + destZ + ")\n";
  }


//**************************
//Section added by MS Nikitidis to implement PSO algorithm
//**************************
  protected synchronized void dataArriveAtUpPort(Object data_,Port inPort)
  {
    TCPPacket tcppkt_ = (TCPPacket)((InetPacket)data_).getBody();

      timesmp =tcppkt_.getTS();

      start_sim = 1;

        MyContext_validity = context_validity(timesmp);


  }

  /*Set dataset no for XYData*/
    public void setDset(int set_)
  { ds = set_; }

    /*Get dataset no for XYData*/
  public long getDset()
  { return ds; }

    public void setDset1(int set1_)
  { ds1 = set1_; }

    public long getDset1()
  { return ds1; }



  //Calculates context validity of frames for each node
  public double context_validity(double timesmp_)//msn 2-12-08
  {

  double time_elapsed;
  double con_val_indx;

  time_elapsed =( getTime() - timesmp_ ) / 60.0;

  con_val_indx = Math.exp(time_elapsed - threshold_validity);
  return con_val_indx;

  }
```

```
//sets and gets threshold time for context validity
  public void setThreshold(double threshold) { threshold_validity = threshold; }//msn 2−12−08


public double getThreshold() { return threshold_validity; }


  public void setValidContextThr(double VCT_) { VCT = VCT_; }//msn 18−12−08
  //sets initial value for cognitive factor context validity


  public void ProcessNeighbors()
  {
    //msn 19−12−2008
    int i,jj;

                  NeighborQueryContract.Message msg1;
     NeighborQueryContract.Message msg;//msn 19−12−08



     long nl[];// msn 19−12−08
     double cff[];
              int i_best=0;



   msg = (NeighborQueryContract.Message)trackPort.sendReceive(new NeighborQueryContract.Message(nid
         , X, Y, Z, nGrids));
   nl = msg.getNodeList();
   msg1 = (NeighborQueryContract.Message)valndxPort.sendReceive(new NeighborQueryContract.Message(
         nid, X, Y, Z, nGrids));
   cff = msg1.getcf();

          Xfol = msg1.getXfol();
          Yfol = msg1.getYfol();

   mean_neighbor_context = 0.0;
         if (nl.length > 0)
   best_neighbor_validity = 1.0;//context_validity(cff[0]);




         OBS_validity = context_validity(OBS_context);
         SBS_validity = context_validity(SBS_context);/*}*/

   jj = 0;

   for ( i = 0; i < nl.length; i ++ ) {

         if ( nl[i] >= NoSrc && cff[i] != 0.0 ){

                        mean_neighbor_context = mean_neighbor_context + cff[i];
         if ( context_validity(cff[i]) < best_neighbor_validity )
                  {
         best_neighbor_validity = context_validity(cff[i]);
                  best_neighbor_context = cff[i];
                  best_SBS_X = Xfol[i];
```

```
                    best_SBS_Y = Yfol[i];
                    i_best = i;}
            jj ++ ;}


            }
            if ( jj > 0 )
            mean_neighbor_context = mean_neighbor_context / jj ;

                    if (context_validity(mean_neighbor_context) < OBS_validity && jj != 0 ){

                        OBS_context = mean_neighbor_context;

                        OBS_validity = context_validity(OBS_context);
              Xsf = X;
              Ysf= Y; }

            if (best_neighbor_validity < SBS_validity && jj != 0 ){

                        SBS_context = best_neighbor_context;

                       SBS_validity = context_validity(SBS_context);
              Xcf = best_SBS_X;
              Ycf= best_SBS_Y; }



                if (context_validity(timesmp) > best_neighbor_validity && best_neighbor_context !=0.0 )
            {
               // MyContext = context_validity(best_neighbor_context);
                    timesmp = best_neighbor_context;

            }

                else if (context_validity(timesmp) <= best_neighbor_validity && best_neighbor_context !=0.0
                     )
                {
                    Xcf = X;
                    Ycf = Y;
                }

                reportPort.doSending(new PositionReportContract.Message(nid,timesmp));


                    SourceFollowerDistance();

}

public void setPSO(int PSO_flag_)//msn 22−12−08
{ PSO_flag = PSO_flag_;}

public void setNoSrc(int NoSrc_)//msn 12−1−09
{ NoSrc = NoSrc_ ; }

public void setInitFactors ( double bsf, double bcf , int start_sim_)//msn 12−1−09
{ SBS_validity = bcf;
  OBS_validity = bsf;
        start_sim = start_sim_;}
```

```
        public synchronized void updatePSOPosition() //msn 18−1−09
        {
            double now = getTime();
            double interval = now − positionUpdateTime;

            if ( interval == 0.0 ) return;

            X = X + sx;
            Y = Y + sy;


            positionUpdateTime = now;
            if (X != X0 && Y != Y0) movcnt = movcnt + 1;
        }

        public synchronized void setPSODestination()//msn 18−1−09
        {
            double dist;
            updatePSOPosition();
            speed = rand.nextDouble() * Max_Speed;

            sx = rand.nextDouble() * c1 * (Xcf − X) + rand.nextDouble() * c2 * (Xsf − X);
            sy = rand.nextDouble() * c1 * (Ycf − Y) + rand.nextDouble() * c2 * (Ysf − Y);


            dist = Math.sqrt((sx * sx) + (sy * sy));


            if ( speed > 1.0e−20)
            changeDestinationPeriod = 1.0; //dist / speed;
            else
            changeDestinationPeriod = dist / 1.0e−20;

            changeDestinationTimer = setTimeout("ChangeDestination", changeDestinationPeriod);
        }

        public void setPSOFactors ( double c1_, double c2_ )//msn 18−1−09
    { c1 = c1_;
      c2 = c2_; }

        public void setnGrids ( int nGrids_) //mns 18−1−09/home/msn/jsim/src/drcl/inet/mac/
                AdvancedMobilityModel.java:653: cannot find symbol
        {
        nGrids = nGrids_;
        }

        public void SourceFollowerDistance()
        {
            int jj;
            double distance[]= new double[30];
            double griddiagonal = Math.sqrt((maxX − minX)*(maxX − minX) + (maxY − minY)*(maxY −
                    minY));
            double travel_distance;

            travel_distance = getSourcePosition();

             for ( jj = 0; jj < NoSrc; jj++)
             {
```

```
                distance[jj] = Math.sqrt((Xsrc[jj] − X)*(Xsrc[jj] − X) + (Ysrc[jj] − Y)*(Ysrc[jj] − Y));
                distance[jj] = (distance[jj] / griddiagonal) * 100;}


        if (nid == ds1 )//&& travel_distance != 0.0 )
            for ( jj = 0; jj < NoSrc; jj++)
          outPort.doSending(new XYData(jj,getTime(),distance[jj]));


    }

    public void setNodeStatus() //msn 26−1−09
    {

        if ( context_validity(timesmp) > VCT) // node eolos
            status = 1;


        if ( context_validity(timesmp) > VCT && context_validity(timesmp) < context_validity(
              SBS_context)) //best_neighbor_validity ) //node compromised
           status = 2;

         if ( context_validity(timesmp) < VCT && (VCT < context_validity(SBS_context) ||
              context_validity(timesmp) < context_validity(SBS_context) ))//best_neighbor_validity ) //
              node satisfied
            status = 3;


         if (nid == ds1 )

       statusPort.doSending(new XYData(ds1,getTime(),status));
    }

    public double getSourcePosition()
    {
        int jj;
        double travel_distance = 0.0;

NeighborQueryContract.Message msg1;

 msg1 = (NeighborQueryContract.Message)valndxPort.sendReceive(new NeighborQueryContract.Message(
       nid, X, Y, Z, nGrids));

        Xsrc = msg1.getXsrc();//msn 25−1−09
        Ysrc = msg1.getYsrc();//msn 25−1−09

        for ( jj = 0; jj < NoSrc; jj++)
          {
             travel_distance = travel_distance + Math.abs(Xsrc[jj] − Xsrcp[jj]) + Math.abs(Ysrc[jj] − Ysrcp
                   [jj]);
        }

         for ( jj = 0; jj < NoSrc; jj++)
         {
              Xsrcp[jj] = Xsrc[jj];
              Ysrcp[jj] = Ysrc[jj];
         }

         return travel_distance;
```

```
        }

        public double UtilityFunction(double Energy_)
        {
            double U1,U;


            U1 = (1 − Math.exp(−(Energy_/1000.0)));

            U = m1 * U1 + m2 * context_validity(timesmp);

            return U;
         }

         public void setUtilityFunctionFactors ( double m1_, double m2_,double UF_thres_)//msn 18−1−09
{ m1 = m1_;
  m2 = m2_;
        UF_thres = UF_thres_; }


   }/*end of class*/
```

WirelessPhy

```
/* Modified by MS Nikitidis March 2009*/


// @(#)WirelessPhy.java 1/2004
// Copyright (c) 1998−2004, Distributed Real−time Computing Lab (DRCL)
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are met:
//
// 1. Redistributions of source code must retain the above copyright notice,
// this list of conditions and the following disclaimer.
// 2. Redistributions in binary form must reproduce the above copyright notice,
// this list of conditions and the following disclaimer in the documentation
// and/or other materials provided with the distribution.
// 3. Neither the name of ”DRCL” nor the names of its contributors may be used
// to endorse or promote products derived from this software without specific
// prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ”AS IS”
// AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
// ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR
// ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
// DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
// SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
// CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
// OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//

package drcl.inet.mac;

import drcl.data.*;
import drcl.comp.*;
import drcl.net.*;
import drcl.inet.*;
import drcl.inet.contract.*;
import java.util.*;
import drcl.comp.Port;
import drcl.comp.Contract;
import java.io.*;
import java.util.StringTokenizer;
import java.lang.Math;

/**
 * This class simulates many functions of a wireless physical card. It piggy−backs
 * various information (ie. location of the sending node, the transmission power
 * of data frame etc.) to the mac layer data frame and passes that fram to the channel.
 * While receiving a data frame from the channel component, it determines whether
 * that frame can be decoded correctly by consulting <code>RadioPropagationModel</code>
 * and passes the decodable frame to the mac layer. It also contains a <code>EnergyModel</code>
 * component to track the energy consumption.
 * @author Ye Ge
 */
public class WirelessPhy extends drcl.net.Module implements ActiveComponent {
```

```
long nid; //this may be removed later. right now, it is same at the node id.

private static String STATUS_SEND = "SEND";
private static String STATUS_RECEIVE = "RECEIVE";
private static String STATUS_IDLE = "IDLE";
//private static int totalCachePutNum = 0;

private static int numAODV = 0;
private static int numACK = 0;
private static int numRTS = 0;
private static int numCTS = 0;
private static int numUDP = 0;
private static int numOthers = 0;

ACATimer idleenergytimer_ = null;

private String status;
private long cur_mov_cnt; // msn 24-11-08
private long prev_mov_cnt;// msn 24-11-08

// Assume AT&T's Wavelan PCMCIA card -- Chalermek
// Pt = 8.5872e-4; // For 40m transmission range.
// Pt = 7.214e-3; // For 100m transmission range.
// Pt = 0.2818; // For 250m transmission range.
// Pt = pow(10, 2.45) * 1e-3; // 24.5 dbm, ~ 281.8mw
//  Pt_consume = 0.660; // 1.6 W drained power for transmission
// Pr_consume = 0.395; // 1.2 W drained power for reception
// P_idle = 0.035; // 1.15 W drained power for idle

/** The energy model installed */
EnergyModel em;

/** Transmitting power */
double Pt;

/** The last time the node sends somthing. */
double last_send_time;

/** When the channel be idle again. */
double channel_become_idle_time;
/** The last time we update energy. */
double last_energy_update_time;

/** Frequency. */
double freq; // frequency

/** Wavelength (m) */
double Lambda; // wavelength (m)
// double L_ = 1.0;  // system loss factor

/** receive power threshold (W) */
double RXThresh; // receive power threshold (W)
/** carrier sense threshold (W) */
double CSThresh; // carrier sense threshold (W)
/** capture threshold (db) */
double CPThresh; // capture threshold (db)

/* configurate the ports */
```

```
public static final String CONFIG_PORT_ID = ".config";
public static final String CHANNEL_PORT_ID = ".channel";
public static final String PROPAGATION_PORT_ID = ".propagation";
public static final String MOBILITY_PORT_ID = ".mobility";
public static final String ENERGY_PORT_ID = ".energy";
public static final String ANTENNA_PORT_ID = ".antenna"; // Chunyu
public static final String MOVHIS_PORT_ID = ".movhis";/*msn 24-11-08*/

protected Port configPort = addPort(CONFIG_PORT_ID, false);
/** the port receiving packets from the channel */
protected Port channelPort = addPort(CHANNEL_PORT_ID, false); // the port receiving packets from the
    channel
/** the port to query the path loss */
protected Port propagationPort = addPort(PROPAGATION_PORT_ID, false); // the port to query the path
    loss
/** the port to query the current position of myself */
protected Port mobilityPort = addPort(MOBILITY_PORT_ID, false); // the port to query the current position
    of myself

protected Port energyPort = addPort(ENERGY_PORT_ID, false);

/** antenna port */
protected Port antennaPort = addPort(ANTENNA_PORT_ID, false); //Chunyu
protected Port movhisPort = addPort(MOVHIS_PORT_ID, false);/*msn 24-11-08*/



/* antenna -- Chunyu*/
Antenna antenna = new Antenna();
ACATimer lockTimer;

// simulate the energy model and antenna model
double Gt = 1.0; // transmitting antenna gain, should be moved to attenna component later
double Gr = 1.0; // receiving antenna gain, should be moved to attenna component later

/** bandwidth */
double bandwidth;

/** use this cache to speed up the simulation by avoiding unnecessary propagation loss calculation */
private Hashtable pathLossCache; // use this cache to speed up the simulation by avoiding unnecessary
    propagation loss calculation
private double Xc = 0.0; // my previous position
private double Yc = 0.0;
private double Zc = 0.0;
private double tc = -1.0; // my cached position and the last time I consult the mobility model.
private static double tp = -1.0; //last time I print the statistics

/** Below are the tolerance of coordinate for hashing,
   * i.e., if any distance change (in any one dimension) exceeds the
   * tolerance below, the path loss will be recalculated,
   * otherwise, it will be just pick from the cache.
   * Notice the tolerance for Cardesian system and longitude-latitude
   * system is different. */
// At the equator, the circumference of the earth is 40,003 kilometers
// (10.0/40003000.0)*360.0 = 9e-5
private double xyz_tol = 10.0;
private double long_lat_tol = 0.00009;

/** A sample card */
```

```
static class SampleCard {
    static double freq = 900000000;
    static double bandwidth = 2000000.0;
    static double Pt = 0.2818; // for 260m transmission range?
    static double Pt_consume = 0.660;
    static double Pr_consume = 0.395;
    static double P_idle = 0.0;
    static double P_sleep = 0.130;
    static double P_off = 0.043;
    static double RXThresh = 0.2818 * (1/100.0) * (1/100.0) * (1/100.0) * (1/100.0);
    static double CSThresh = 0.2818 * (1/100.0) * (1/100.0) * (1/100.0) * (1/100.0) / 8.0;
    // static double RXThresh = Math.pow(10.0, −20.0/10.0) * 0.001;;
    // static double CSThresh = Math.pow(10.0, −30.0/10.0) * 0.001;;
    static double CPThresh = 10; // (db)
    SampleCard() {};
}

/** tank to soldier card for NMS demo */
static class Demo_TSCard { // tank to soldier card for NMS demo
    static double freq = 2400000000.0;
    static double bandwidth = 6000000.0;
    static double Pt = Math.pow(10.0, (21.94 + 0.15 )/10.0) * 0.001; // for 260m transmission range?
                                        // added 0.15 is Gt − cable loss
    static double Pt_consume = 0.660;
    static double Pr_consume = 0.395;
    static double P_idle = 0.0;
    static double P_sleep = 0.130;
    static double P_off = 0.043;

    static double RXThresh = Math.pow(10.0, −68.0/10.0) * 0.001;;
    static double CSThresh = Math.pow(10.0, −78.0/10.0) * 0.001;;
    // static double RXThresh = Math.pow(10.0, −20.0/10.0) * 0.001;;
    // static double CSThresh = Math.pow(10.0, −30.0/10.0) * 0.001;;
    static double CPThresh = 10; // (db)
    Demo_TSCard() {};
}

/** tank to tank card for NMS demo */
static class Demo_TTCard { // tank to tank card for NMS demo
    static double freq = 2400000000.0;
    static double bandwidth = 2000000.0;
    static double Pt = Math.pow(10.0, (40.0 + 0.15)/10.0) * 0.001; // for 260m transmission range?
                                        //the added 0.15 is Gt − cableloss = 2.15 − 2.0
    static double Pt_consume = 0.660;
    static double Pr_consume = 0.395;
    static double P_idle = 0.0;
    static double P_sleep = 0.130;
    static double P_off = 0.043;

    static double RXThresh = Math.pow(10.0, −68.0/10.0) * 0.001;;
    static double CSThresh = Math.pow(10.0, −78.0/10.0) * 0.001;;
    // static double RXThresh = Math.pow(10.0, −20.0/10.0) * 0.001;;
    // static double CSThresh = Math.pow(10.0, −30.0/10.0) * 0.001;;
    static double CPThresh = 10; // (db)
    Demo_TTCard() {};
}

/** tank to uav card for NMS demo */
static class Demo_TUCard { // tank to uav card for NMS demo
```

```
        static double freq = 2400000000.0;
        static double bandwidth = 3000000.0;
        static double Pt = Math.pow(10.0, 42.0/10.0) * 0.001; // for 260m transmission range?
        static double Pt_consume = 0.660;
        static double Pr_consume = 0.395;
        static double P_idle = 0.0;
        static double P_sleep = 0.130;
        static double P_off = 0.043;
        static double RXThresh = Math.pow(10.0, −68.0/10.0) * 0.001;;
        static double CSThresh = Math.pow(10.0, −78.0/10.0) * 0.001;;
        // static double RXThresh = Math.pow(10.0, −20.0/10.0) * 0.001;;
        // static double CSThresh = Math.pow(10.0, −30.0/10.0) * 0.001;;
        static double CPThresh = 10; // (db) xxxxxxxxok
        Demo_TUCard() {};
}

public void _start() {
        idleenergytimer_ = setTimeout(”IdleEnergyUpdateTimeout”, 10.0);
}

/**
 * Constructor. Sets some parameters according to a simple card.
 */
public WirelessPhy() {
        super();
        pathLossCache = new Hashtable();
        freq = SampleCard.freq;
        Pt = SampleCard.Pt;
        bandwidth = SampleCard.bandwidth;
        Lambda = 300000000.0 / freq; // wavelength (m)
        RXThresh = SampleCard.RXThresh;
        CSThresh = SampleCard.CSThresh; // carrier sense threshold (W) //
        CPThresh = SampleCard.CPThresh; // capture threshold (db)
        last_send_time = 0.0; // the last time the node sends somthing.
        channel_become_idle_time = 0.0; // channel idle time.
        last_energy_update_time = 0.0; // the last time we update energy.
        em = new EnergyModel();
}

/**
 * Configures the card parameters.
 */
public void configureCard(String card_) {
        if ( card_.equals(”Demo_TSCard”) ) {
                freq = Demo_TSCard.freq;
                Pt = Demo_TSCard.Pt;
                bandwidth = Demo_TSCard.bandwidth;
                RXThresh = Demo_TSCard.RXThresh;
                CSThresh = Demo_TSCard.CSThresh; // carrier sense threshold (W) //
                CPThresh = Demo_TSCard.CPThresh; // capture threshold (db)
                Lambda = 300000000.0 / freq; // wavelength (m)
        }
        else if ( card_.equals(”Demo_TTCard”) ) {
                freq = Demo_TTCard.freq;
                Pt = Demo_TTCard.Pt;
                bandwidth = Demo_TTCard.bandwidth;
                RXThresh = Demo_TTCard.RXThresh;
                CSThresh = Demo_TTCard.CSThresh; // carrier sense threshold (W) //
                CPThresh = Demo_TTCard.CPThresh; // capture threshold (db)
```

```
            Lambda = 300000000.0 / freq; // wavelength (m)
        }
        else if ( card_.equals("Demo_TUCard") ) {
            freq = Demo_TUCard.freq;
            Pt = Demo_TUCard.Pt;
            bandwidth = Demo_TUCard.bandwidth;
            RXThresh = Demo_TUCard.RXThresh;
            CSThresh = Demo_TUCard.CSThresh; // carrier sense threshold (W) //
            CPThresh = Demo_TUCard.CPThresh; // capture threshold (db)
            Lambda = 300000000.0 / freq; // wavelength (m)
        }
    }

public EnergyModel getEnergyModel()
{ return em; }

    public String getName() { return "WirelessPhy"; }

    public void duplicate(Object source_) {
        super.duplicate(source_);
        WirelessPhy that_ = (WirelessPhy) source_;
        Pt = that_.Pt;
        RXThresh = that_.RXThresh;
        CSThresh = that_.CSThresh;
        CPThresh = that_.CPThresh;
        freq = that_.freq;
        Lambda = that_.Lambda;
        bandwidth = that_.bandwidth;

        // need to duplicate the energy model?
    }

    public Port getChannelPort() { return channelPort; }

    /**
     * Sets the node id.
     */
    public void setNid(long nid_) { nid = nid_; }

    /**
     * Gets the node id.
     */
    public long getNid(long nid_) { return nid; }

    /** Sets the transmission power */
    public void setPt(double Pt_) { Pt = Pt_; }

    /** Sets the power level. */
    public void setPwl(int pwl_) {Pt = Pt/pwl_;} /* power level in decending order, 1 — maximum */

    public void setRxThresh(double RXThresh_) { RXThresh = RXThresh_; }
    public void setCSThresh(double CSThresh_) { CSThresh = CSThresh_; }
    public void setCPThresh(double CPThresh_) { CPThresh = CPThresh_; }

    /** Sets the frequency */
    public void setFreq(double freq_ ) { freq = freq_; Lambda = 300000000.0 / freq; }

public void setBandwidth(double bw_) { bandwidth = bw_; }
```

```
/**
 * Processes data frame coming from MAC component.
 */
protected synchronized void dataArriveAtUpPort(Object data_, drcl.comp.Port upPort_) {

    if ( !em.getOn() || em.getSleep() )
        return; // packet can not be transmitted, drop siliently

    //Decreases node's energy
    if ( em.energy > 0 ) {
        double txtime = ((Packet) data_).size * 8.0 / bandwidth;
        double start_time = Math.max(channel_become_idle_time, getTime());
        double end_time = Math.max(channel_become_idle_time, getTime()+txtime);
        double actual_txtime = end_time - start_time;

        // decrease the energy consumed during the period from the last energy updating time to the start of
            the this transmission
        if (start_time > last_energy_update_time) {
            em.updateIdleEnergy(start_time - last_energy_update_time);
            last_energy_update_time = start_time;
        }
        double temp = Math.max(getTime(), last_send_time);
        double begin_adjust_time = Math.min(channel_become_idle_time, temp);
        double finish_adjust_time = Math.min(channel_become_idle_time, getTime()+txtime);
        double gap_adjust_time = finish_adjust_time - begin_adjust_time;
        if (gap_adjust_time < 0.0) {
            drcl.Debug.error("Negative gap time. Check WirelessPhy.java! finish=" + finish_adjust_time +
                ", begin=" + begin_adjust_time + "\n");
        }

        if ((gap_adjust_time > 0.0) && (status == STATUS_RECEIVE)) {
            em.updateTxEnergy(-gap_adjust_time);
            em.updateRxEnergy(gap_adjust_time);

        }
        em.updateTxEnergy(actual_txtime);

        if (end_time > channel_become_idle_time) {
            status = STATUS_SEND;
        }
        last_send_time = getTime() + txtime;
        channel_become_idle_time = end_time;
        last_energy_update_time = end_time;
        if (!em.getOn()) {
            // logEnergy();
        }
    }
    else {
        // siliently discards the packet
        return;
    }

    double t;
    t = this.getTime();
    if ( Math.abs(t - tc) > 1.0 ) { // the least position check interval is one second to speed up simulation
        PositionReportContract.Message msg = new PositionReportContract.Message();
        msg = (PositionReportContract.Message) mobilityPort.sendReceive(msg);
        Xc = msg.getX();
        Yc = msg.getY();
```

```
            Zc = msg.getZ();
            tc = t;

        }
        downPort.doSending(new NodeChannelContract.Message(nid, Xc, Yc, Zc, Pt, Gt, data_));

        //Add by Honghai for debugging
        if (isDebugEnabled() ) {
          if (t − tp > 1.0) {
            printPktStat();
            tp = t;
          }
          String pktType = ((Packet)data_).getPacketType();
          if (pktType.equals("AODV") ) numAODV ++;
          else if (pktType.equals("MAC−802.11_ACK_Frame") ) numACK++;
          else if (pktType.equals("MAC−802.11_RTS_Frame") ) numRTS++;
          else if (pktType.equals("MAC−802.11_CTS_Frame") ) numCTS++;
          else if (pktType.equals("UDP") ) numUDP++;
          else {
            numOthers++;
            System.out.println("type <" + pktType + ">" );
      }
    }
    }

void printPktStat() {
    StringBuffer sb_ = new StringBuffer(toString());
    sb_.append("AODV packet: " + numAODV);
    sb_.append("\tRTS packet: " + numRTS);
    sb_.append("\tCTS packet: " + numCTS);
    sb_.append("\tACK packet: " + numACK);
    sb_.append("\tUDP packet: " + numUDP);
    sb_.append("\tOther packet: " + numOthers);
    debug(sb_.toString() );
}

    void logEnergy() {
        ;// drcl.Debug.debug("At time: "+ getTime() + " Node" + nid + " remaining energy = " + em.getEnergy
              () + "movements" + (mh−mh1) + "enred" + em.geten() + "\n");
    }

    /**
     * Processes the received frame.
     */
    protected synchronized void dataArriveAtChannelPort(Object data_) {
        double Pr;
        double Loss;
        double Pt_received; // Pt of the received packet
        double Gt_received; // Gt of the received packet
        double Xs, Ys, Zs; // position of the sender

        boolean incorrect = false;

        Packet pkt;

        double t;
        t = this.getTime();
        if ( Math.abs(t − tc) > 1.0 ) { // the least position check interval is one second to speed up simulation
            PositionReportContract.Message msg = new PositionReportContract.Message();
```

```
        msg = (PositionReportContract.Message) mobilityPort.sendReceive(msg);
        Xc = msg.getX();
        Yc = msg.getY();
        Zc = msg.getZ();
        tc = t;
    }

    NodeChannelContract.Message msg2 = (NodeChannelContract.Message) data_;
    Xs = msg2.getX();
    Ys = msg2.getY();
    Zs = msg2.getZ();
    Gt_received = msg2.getGt();
    Pt_received = msg2.getPt();

    String type = antenna.QueryType();
    Antenna.Orientation incomingOrient = new Antenna.Orientation(0, 0);
    if (!type.equals("OMNIDIRECTIONAL ANTENNA")) {
            // add by Chunyu -- calculate the gain from uni-/omni-directional antenna gain
            // 1. calculate the incoming angle
            incomingOrient = CalcOrient (Xc, Yc, Zc, Xs, Ys, Zs);
            // 2. get the antenna gain in dBi and convert it to absolute value
            Gr = Math.exp (0.1 * antenna.getGain_dBi(incomingOrient) );
    }

    Long sid = new Long(msg2.getNid());


    boolean cacheHit = false;
    Loss = 1.0; // Loss will be set to proper value below, here I set it to 1.0 to avoid the compiler's complaint

    if ( pathLossCache.containsKey(sid) ) {
        CachedPathLoss c = (CachedPathLoss) (pathLossCache.get(sid));
    if (RadioPropagationModel.isCartesianCoordinates()) {
    if (Math.abs(c.xs - Xs) <= xyz_tol &&
      Math.abs(c.ys - Ys) <= xyz_tol &&
      Math.abs(c.zs - Zs) <= xyz_tol &&
      Math.abs(c.xr - Xc) <= xyz_tol &&
      Math.abs(c.yr - Yc) <= xyz_tol &&
      Math.abs(c.zr - Zc) <= xyz_tol ) {
        cacheHit = true;
        Loss = c.loss;
    }
  } else {
    if (Math.abs(c.xs - Xs) <= long_lat_tol &&
      Math.abs(c.ys - Ys) <= long_lat_tol &&
      Math.abs(c.zs - Zs) <= xyz_tol &&
      Math.abs(c.xr - Xc) <= long_lat_tol &&
      Math.abs(c.yr - Yc) <= long_lat_tol &&
      Math.abs(c.zr - Zc) <= xyz_tol )  {
        cacheHit = true;
                Loss = c.loss;
    }
  }
    }

    if ( cacheHit == false ) {
        RadioPropagationQueryContract.Message msg3 = (RadioPropagationQueryContract.Message)
              propagationPort.sendReceive(new RadioPropagationQueryContract.Message(Lambda, Xs, Ys,
               Zs, Xc, Yc, Zc ));
```

```
        Loss = msg3.getLoss();
        CachedPathLoss c = new CachedPathLoss(Xc, Yc, Zc, Xs, Ys, Zs, Loss);
        pathLossCache.put(sid, c);
    }

    Pr = Pt_received * Gt_received * Gr * Loss;

    // if the node is in sleeping mode, drop the packet simply
    //Rong's comment: it is possible that the interface overheard other's communication
    if ( !em.getOn() ) {
        //debug("Packet is dropped at node" + nid + " because the node is not on");
        return;
    }
    if ( em.getSleep() ) {
        //debug("Packet is dropped at node" + nid + " because the node is sleeping");
        return;
    }

    pkt = (Packet) msg2.getPkt();

    if ( Pr < CSThresh/1000 ) {
        return;
    }

    if ( Pr < RXThresh) {
        // can detect, but not successfully receive this packet.
        // marks the packet erro;
        incorrect = true;
    }

    //if modulation is simulated, mark packek decoding error here

    /*
     * The MAC layer must be notified of the packet reception
     * now — ie; when the first bit has been detected — so that
     * it can properly do Collision Avoidance / Detection.
     */

    /*
     * Decrease energy if packet successfully received
     */
    double rcvtime = (8. * ((Packet) pkt).size) / bandwidth;
    // no way to reach here if the energy level < 0

    double start_time = Math.max(channel_become_idle_time, getTime());
    double end_time = Math.max(channel_become_idle_time, getTime() + rcvtime);
    double actual_rcvtime = end_time−start_time;

    if (start_time > last_energy_update_time) {
        em.updateIdleEnergy(start_time−last_energy_update_time);
        last_energy_update_time = start_time;
    }

    em.updateRxEnergy(actual_rcvtime);
    if (end_time > channel_become_idle_time) {
        status = STATUS_RECEIVE;
    }

    channel_become_idle_time = end_time;
```

```
        last_energy_update_time = end_time;

        if (em.getOn()) {
            // logEnergy();
        }

        /* added by Chunyu Aug. 05, 2002
         * 1. lock on the signal if the antenna is not locked
         * 2. if lock succeeds, recalculate Graphics= antenna.getGain() and Pr
         * 3. set a timer to unlock the antenna, which times out at end_time
         */
        if ( !type.equals("OMNIDIRECTIONAL ANTENNA")
       && Pr >= RXThresh && !antenna.isLocked()) {
            antenna.lockAtSignal (incomingOrient);
            Gr = Math.exp (0.1 * antenna.getGain_dBi (incomingOrient) );
            // incomingOrient.azimuth + ". Gr = " + Gr); //for debug
            Pr = Pt_received * Gt_received * Gr * Loss;
            lockTimer = setTimeout ("AntennaLockSignal_TimeOut", end_time-getTime());
        } //endif


        // MacPhyContract.Message is defined to convey all necessary information to the MAC component.
        MacPhyContract.Message msg4 = new MacPhyContract.Message(incorrect, Pr, CPThresh, CSThresh,
            RXThresh, pkt);
        upPort.doSending(msg4);
    }

    /* added by Chunyu -- 08.08.2002
     * Calculate the orientation of the sender (Xt, Yt, Zt) in regards to
     * the receiver's position (Xr, Yr, Zr)
     */
    /**
     * Calculates the orientation of the sender (Xt, Yt, Zt) in regards to
     * the receiver's position (Xr, Yr, Zr)
     *
     */
    protected Antenna.Orientation CalcOrient (
        double Xr, double Yr, double Zr,
        double Xt, double Yt, double Zt)
    {
        double delta_x, delta_y, delta_z, delta_xy;
        double alfa = 0, beta = 0;
        Antenna.Orientation orient;

        delta_x = Xt-Xr;
        delta_y = Yt-Yr;
        delta_z = Zt-Zr;
        delta_xy = Math.sqrt(delta_x*delta_x + delta_y*delta_y);

        if (delta_x==0) {
            if (delta_y==0) alfa = 0;
            else if(delta_y>0) alfa = 90;
            else alfa = 270;
        }else {
            alfa = Math.toDegrees (Math.abs (Math.atan(delta_y/delta_x)));
            if (delta_x>0 && delta_y>=0) ;
            else if (delta_x<0 && delta_y >=0) alfa = 180-alfa;
            else if (delta_x<0 && delta_y < 0) alfa = 180+alfa;
            else if (delta_x>0 && delta_y < 0) alfa = 360-alfa;
```

```
        }

        if (delta_xy==0){
            if (delta_z==0) beta = 0;
            else if (delta_z>0) beta = 90;
            else beta = 270;
        } else {
            beta = Math.toDegrees (Math.abs (Math.atan(delta_z/delta_xy)));
            if (delta_xy>0 && delta_z>=0) ;
            else if (delta_xy<0 && delta_z >=0) beta = 180 – beta;
            else if (delta_xy<0 && delta_z >=0) beta = 180 + beta;
            else if (delta_xy>0 && delta_z < 0) beta = 360 – beta;
        }

        return new Antenna.Orientation ((int)alfa, (int)beta);
    } //end CalcOrient

    /**
     * Configures the node's antenna from assigned port.
     */
    protected void configAntenna (Object data_)
    {
        String args = ((String)data_).toLowerCase(), value;

        /* create an antenna */
        if (args.startsWith("create")) {
            String ant = args.substring (args.indexOf("create")+ 6);
            ant = ant.trim();

            if (ant.equals("antenna")) {
                //antenna = new Antenna();
                return;
            }

            if (ant.equals("switchedbeam antenna")) {
                antenna = new SwitchedBeamAntenna();
                return;
            }

            if (ant.equals("adaptive antenna")) {
                antenna = new AdaptiveAntenna();
                return;
            }

            System.out.println ("FORMAT erorr! shall be <create antenna/switchedbeam antenna/adaptive
                antenna>");
            return;
        } //endif "create"

        /* Query the antenna type*/
        if (args.startsWith ("querytype")) {
            System.out.println (antenna.QueryType());
            return;
        } //endif "QueryType"

        /* initialization work */
        int index;
        if ( (index = args.indexOf ('=')) != –1) {
            value = (args.substring(index+1)).trim();
```

```
        if (value.equals(null)) {
        System.out.println (this + ":: pls. use the format such as <height = 1.5>");
        return;
    } //endif value

    if (args.indexOf ("height")!=-1) {
        float height = Float.parseFloat(value);
        antenna.setHeight (height);
        System.out.println ("set height = " + antenna.setHeight(height));
        return;
    } //endif "height"

    if (args.indexOf ("omnigain_dbi")!=-1) {
        float omniGain_dBi = Float.parseFloat(value);
        antenna.setOmniGain_dBi(omniGain_dBi);
        System.out.println ("set omniGain_dBi = " + antenna.getGain_dBi());
        return;
    } //endif "omniGain_dBi"

    if (args.indexOf ("azimuthpatterns")!=-1) {
        try {
            BufferedReader in =
                new BufferedReader (new FileReader(value));
            in.close();
        } catch (java.io.IOException e) {
            System.out.println (this + ":: error in opening " + value);
            return;
        } //endtry

        if (antenna.initAzimuthPatterns (value))
            ;//System.out.println (this + "Successfully initialize the azimuth pattern file!");
        else
            System.out.println (this + "Failure in initializing the azimuth pattern file!");
        return;
    } //endif "azimuthPatterns"

    if (args.indexOf ("elevationpatterns")!=-1) {
        try {
            BufferedReader in =
                new BufferedReader (new FileReader(value));
            in.close();
        } catch (java.io.IOException e) {
            System.out.println (this + ":: error in opening " + value);
            return;
        } //endtry

        if (antenna.initElevationPatterns (value))
            System.out.println (this + "Successfully initialize the elevation pattern file!");
        else
            System.out.println (this + "Failure in initializing the evlation pattern file!");
        // antenna transmission gain, should be moved to attenna component later
        return;
    } //endif "elevationPatterns"

    System.out.println (" Wrong format: no such initialization item!");

    } //endif ' ... = ... '

    System.out.println ("Wrong format to communicate with the Antenna component!");
```

```
} //end configAntenna

protected synchronized void processOther(Object data_, Port inPort_) {
    String portid_ = inPort_.getID();

    if (portid_.equals(CHANNEL_PORT_ID)) {
        if (!(data_ instanceof NodeChannelContract.Message)) {
            error(data_, "processOther()", inPort_, "unknown object");
            return;
        }
        dataArriveAtChannelPort(data_); // a packet is "heard" from the channel
        return;
    }

    if (portid_.equals(ENERGY_PORT_ID)) {
        if (data_ == null) // this is a query message
     energyPort.doSending(em);
        else if (data_ instanceof BooleanObj) {
            em.setSleep(((BooleanObj) data_).getValue());
            updateIdleEnergy();
        }
        else
            drcl.Debug.error("WirelessPhy.processOther()", "Unrecognized contract", true);
        return;
    }

    /* antenna -- Chunyu */
    if (portid_.equals (ANTENNA_PORT_ID)) {
        configAntenna (data_);
        return;
    }

   if (portid_.equals(MOVHIS_PORT_ID)) {//msn 24-11-08
      //drcl.Debug.debug("At time: "+ data_ + "\n");
   calc_energy_mov(data_);
   return;
                }


    super.processOther(data_, inPort_);
}

/**
 * Preriodically timeout to update energy consumption even if it is in idle state.
 */
public synchronized void timeout(Object data_) {
    if ( data_ instanceof String ){
        if ( ((String) data_).equals("IdleEnergyUpdateTimeout") ) {
            if ( em.getEnergy() > 0 ) logEnergy();
            updateIdleEnergy();
            idleenergytimer_ = setTimeout("IdleEnergyUpdateTimeout", 10.0);
        }
        else if ( ((String) data_).equals("AntennaLockSignal_TimeOut") ) {
            antenna.unlock();
        }
    }
}
```

```
    /**
     * updates energy consumption during the idle state.
     */
    protected void updateIdleEnergy() {
        if ( getTime() > last_energy_update_time) {
            if(em.getOn()) {
                if(em.getSleep())
                    em.updateSleepEnergy(getTime() − last_energy_update_time);
                else
                    em.updateIdleEnergy(getTime() − last_energy_update_time);
            }
            last_energy_update_time = getTime();
        }
        //if ( em.getEnergy() > 0 ) logEnergy();
    }

  protected void calc_energy_mov(Object data_) // msn 24−11−08
    {

    prev_mov_cnt= cur_mov_cnt;
    cur_mov_cnt = (Long) data_;
    em.updateMovEnergy(cur_mov_cnt−prev_mov_cnt);//msn 23−11−08
  ///// if (em.getEnergy() <= 0.0) movhisPort.doSending(nid);
        movhisPort.doSending(new NeighborQueryContract.Message(nid, em.getEnergy()));

  // if (nid ==5) drcl.Debug.debug("At time: "+ getTime() + " Node" + nid + " remaining energy = " + em.
        getEnergy() + "movements" + (mh−mh1) + "enred" + em.geten() + "\n");


    }

}
```

EnergyModel

```
/* Modified by MS Nikitidis March 2009*/

// @(#)EnergyModel.java 1/2004
// Copyright (c) 1998−2004, Distributed Real−time Computing Lab (DRCL)
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are met:
//
// 1. Redistributions of source code must retain the above copyright notice,
// this list of conditions and the following disclaimer.
// 2. Redistributions in binary form must reproduce the above copyright notice,
// this list of conditions and the following disclaimer in the documentation
// and/or other materials provided with the distribution.
// 3. Neither the name of "DRCL" nor the names of its contributors may be used
// to endorse or promote products derived from this software without specific
// prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
// AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
// ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR
// ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
// DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
// SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
// CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
// OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//

package drcl.inet.mac;

/**
 * The class implements a simple energy model.
 * @author Rong Zheng
 */
public class EnergyModel extends drcl.DrclObj {
    /** Creates a new instance of EnergyInfo */
    double Pt = 0.2818;  // transmitter signal power (W)
    double Pt_consume = 0.660; // power consumption for transmission (W)
    double Pr_consume = 0.395; // power consumption for reception (W)
    double P_idle = 0.0; // idle power consumption (W)
    double P_off = 0.043;
    double P_sleep = 0.130;
   double P_mov = 0.362; // msn 23−11−08

    boolean isOn = true;
    boolean isSleep = false;

    double energy = 1000;
   long m1;
    public EnergyModel() {
    }

  /** Set energy consumption
    * @param Pt_ signal power
```

```
    * @param Pt_consume_ power consumption of transmission
    * @param Pr_consume_ power consumption for reception
    * @param P_idle_ idle power consumption
    * @param P_off_ shutdown energy consumption
    */
    public void setEnergyConsumption(double Pt_, double Pt_consume_, double Pr_consume_, double P_idle_,
            double P_off_, double P_mov_) {//msn 23-11-08
        Pt = Pt_;
        Pt_consume = Pt_consume_;
        Pr_consume = Pr_consume_;
        P_idle = P_idle_;
        P_off = P_off_;
      P_mov = P_mov_;//msn 23-11-08
    }

public String toString()
{
    return "EnergyModel:t_signal_power=" + Pt + ",t_consume=" + Pt_consume + ",r_consume=" +
            Pr_consume + ",idle_consume=" + P_idle + ",P_off=" + P_off + ",P_mov=" + P_mov;
}//msn 24-11-08

    protected void setOn(boolean is_on) {
        isOn = is_on;
    }

    protected void setSleep(boolean is_sleep) {
        isSleep = is_sleep;
    }

    protected void setEnergy(double energy_) {
        energy = energy_;
    }

    protected boolean updateIdleEnergy(double time) {
        energy -= P_idle*time;
        if (energy <= 0) {
            energy = 0;
            isOn = false;
        }

        return isOn;
    }

    protected boolean updateTxEnergy(double time) {
        energy -= Pt_consume*time;
        if (energy <= 0) {
            energy = 0;
            isOn = false;
        }

        return isOn;
    }

    protected boolean updateRxEnergy(double time) {
        energy -= Pr_consume*time;
        if (energy <= 0) {
            energy = 0;
            isOn = false;
        }
```

```
        return isOn;
    }

    protected boolean updateSleepEnergy(double time) {
        energy −= P_sleep*time;
        if (energy <= 0) {
            energy = 0;
            isOn = false;
        }

        return isOn;
    }

    protected boolean getOn() {
        return isOn;
    }

    protected boolean getSleep() {
        return isSleep;
    }

    protected double getEnergy() {
        return energy;
    }


    protected boolean updateMovEnergy(long mov_) {// msn 19−11−08
        energy −= P_mov * mov_;
      m1=mov_;
        if (energy <= 0) {
            energy = 0;
            isOn = false;
        }

        return isOn;
    }

// public double geten()
// {
// return (P_mov * m1);
// }

}
```

NeighborQueryContract

```
/* Modified by MS Nikitidis March 2009*/


// @(#)NeighborQueryContract.java 1/2004
// Copyright (c) 1998−2004, Distributed Real−time Computing Lab (DRCL)
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are met:
//
// 1. Redistributions of source code must retain the above copyright notice,
// this list of conditions and the following disclaimer.
// 2. Redistributions in binary form must reproduce the above copyright notice,
// this list of conditions and the following disclaimer in the documentation
// and/or other materials provided with the distribution.
// 3. Neither the name of ”DRCL” nor the names of its contributors may be used
// to endorse or promote products derived from this software without specific
// prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ”AS IS”
// AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
// ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR
// ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
// DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
// SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
// CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
// OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//

package drcl.inet.mac;

import drcl.data.*;
import drcl.net.*;
import drcl.comp.*;
import drcl.util.ObjectUtil;
import drcl.util.StringUtil;

/**
 * The NeightbourQueryContract contract. This contract defines the commands
 * and the message format between <code>Channel</code> component
 * and <code>NodePositionTracker</code> component.
 *
 * @see Channel
 * @see NodePositionTracker
 * @author Ye Ge
 */
public class NeighborQueryContract extends Contract
{
  public static final NeighborQueryContract INSTANCE = new NeighborQueryContract();

  public NeighborQueryContract()
  { super(); }

  public NeighborQueryContract(int role_)
```

```
{ super(role_); }

public String getName()
{ return "Position Report Contract"; }

public Object getContractContent()
{ return null; }
  /**
   * Queries the neighbours of the sender whose current position is (X_, Y_, Z_).
   */
public static Object query(long nid_, double X_, double Y_, double Z_, Port out_)
{ return out_.sendReceive(new Message(nid_, X_, Y_, Z_)); }

  /**
   * Sends back a reply through the out_ port
   *
   *@param nodeList_ an array of the nid's of neighboring nodes.
   *@out_ the port through which a reply message should be sent back.
   */
public static void reply(long[] nodeList_, Port out_)
{ out_.doSending(new Message(nodeList_)); }

  public static class Message extends drcl.comp.Message
{
      long nid;
      double X, Y, Z;
      long[] nodeList;
    double[] cf; //msn 9-1-09
        double[] Xsrc;//msn25-1-09
        double[] Ysrc;//msn 25-1-09
        double[] Xfol;//msn25-1-09
        double[] Yfol;//msn 25-1-09
        double Energy;

      // All nodes in nGrids away subareas are considered as neighbours.
      // By default, nGrids is set to 1, which means all nodes in 9 adjacent
      // subareas (including the subarea where the sender is located) are neighbours.
      int nGrids = 1;

  public Message () { }

      /**
       * Constructor. Assumes nGrids is 1.
       *
       *
       *@param nid_ the id of the querying node.
       *@param X_ the x coordinate of the node current position.
       *@param Y_ the y coordinate of the node current position.
       *@param Z_ the z coordinate of the node current position.
       *
       */
  public Message (long nid_, double X_, double Y_, double Z_)
  {
    nid = nid_;
          X = X_; Y = Y_; Z = Z_;
          nodeList = null;
          nGrids = 1;
  }
      /**
```

```
        * Constructor.
        *
        *
        *@param nid_ the id of the querying node.
        *@param X_ the x coordinate of the node current position.
        *@param Y_ the y coordinate of the node current position.
        *@param Z_ the z coordinate of the node current position.
        *@nGrids_ the nGrid value.
        */
public Message (long nid_, double X_, double Y_, double Z_, int nGrids_)
{
  nid = nid_;
          X = X_; Y = Y_; Z = Z_;
          nodeList = null;
          nGrids = nGrids_;
}

     /**
       * Construct. Constructs a reply message containning only a node list.
       *
       *@param nodeList_ the list of all neighbour node id.
       *
       */
     public Message (long[] nodeList_)
     {
          nid = -1;
          X = 0; Y = 0; Z = 0;
          nGrids = nGrids;
          nodeList = nodeList_;
     }

/**
        * Constructor. Constructs a message
        *
  * @param nid_ the node id
  * @param X_ the X coordinate of the querying node
  * @param Y_ the Y coordinate of the querying node
  * @param Z_ the Z coordinate of the querying node
  * @param nodeList_ the id list of all neighbour nodes
  * @param nGrids_ the value of nGrids
  */
public Message (long nid_, double X_, double Y_, double Z_, long[] nodeList_, int nGrids_)
{
          nid = nid_;
  X = X_; Y = Y_; Z = Z_;
          if ( nodeList_ != null ) {
               int n = nodeList_.length;
               nodeList = new long[n];
               for ( int i = 0; i < n; i ++ )
                    nodeList[i] = nodeList_[i];
          }
          else
               nodeList = null;
          nGrids = nGrids_;
}

  public Message (double[] cf_, double[] Xsrc_, double[] Ysrc_, double[] Xfol_,double[] Yfol_) //msn
       9-1-09 //revised 25-1-09
     {
```

```
            nid = −1;
            X = 0; Y = 0; Z = 0;
            nGrids = nGrids;
            cf = cf_;
            Xsrc = Xsrc_;
            Ysrc = Ysrc_;
            Xfol = Xfol_;
            Yfol = Yfol_;
        }

        public Message (long nid_, double Energy_)
        {
            nid = nid_;
            Energy = Energy_;
        }
    /** Gets the node id */
        public long getNid() { return nid; }
    /** Gets the X coordinate */
    public double getX() { return X; }
    /** Gets the Y coordinate */
    public double getY() { return Y; }
    /** Gets the Z coordinate */
    public double getZ() { return Z; }
    /** Gets nGrids */
    public int getnGrids() { return nGrids; }
    /**Gets cf*/
    public double[] getcf()  { return cf; }//msn 9−1−09
        public double[] getXsrc()  { return Xsrc; }//msn 25−1−09
        public double[] getYsrc()  { return Ysrc; }//msn 25−1−09
        public double[] getXfol()  { return Xfol; }//msn 25−1−09
        public double[] getYfol()  { return Yfol; }//msn 25−1−09
        public double getEnergy() { return Energy; }

        /** Gets node list in the message */
        public long[] getNodeList() {
            return nodeList;
        }

    public Object clone()
    { return new Message(nid, X, Y, Z, nodeList, nGrids); }

    public Contract getContract()
    { return INSTANCE; }

    public String toString(String separator_)
    {
            String str;
            str = ”Neighbor Query:” + separator_ + ”nid=” + nid + separator_ + ”X=” + X + separator_ + ”Y=”
                + Y + separator_ + ”Z=” + Z + separator_ + ”nodeList =”;
            if ( nodeList != null )
                if ( nodeList.length != 0 )
                    for ( int i = 0; i < nodeList.length; i ++ )
                        str = str + nodeList[i] + ” ”;
            str = str + separator_ + ”nGrids=” + nGrids;
      return str;
    }
  }
}
```

NodePositionTracker

```
/* Modified by MS Nikitidis March 2009*/


// @(#)NodePositionTracker.java 7/2003
// Copyright (c) 1998−2003, Distributed Real−time Computing Lab (DRCL)
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are met:
//
// 1. Redistributions of source code must retain the above copyright notice,
// this list of conditions and the following disclaimer.
// 2. Redistributions in binary form must reproduce the above copyright notice,
// this list of conditions and the following disclaimer in the documentation
// and/or other materials provided with the distribution.
// 3. Neither the name of ”DRCL” nor the names of its contributors may be used
// to endorse or promote products derived from this software without specific
// prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ”AS IS”
// AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
// ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR
// ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
// DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
// SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
// CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
// OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//

package drcl.inet.mac;

import drcl.comp.*;
import drcl.net.*;
import drcl.inet.*;
import drcl.inet.contract.*;
import java.util.*;
import drcl.comp.Port;
import drcl.comp.Contract;

/**
* This component divides the X−Y plane of the simulated space area into multiple
* subareas. Each mobile node’s <code>MobiltiyModel</code> periodically reports
* its position to <code>NodePositionTracker</code>.
* When a <code>NodePositionTracker</code> receives a neighboring node inquery from
* the <node>Channel</code> component, the <code>NodePositionTracker</code>
* first finds out which subarea that sender node is located and
* returns the nid’s of all mobile nodes which are currently located in all neighbouring
* subareas.
*
* @author Ye Ge
* @see Channel
* @see MobilityModel
*
*/
```

```
public class NodePositionTracker extends drcl.net.Module {

  /* configurate the ports */
  private static final String NODE_PORT_ID = ".node"; // connect to the mobile nodes' wirelessphy
  private static final String CONFIG_PORT_ID = ".config";
  private static final String CHANNEL_PORT_ID = ".channel"; // connect to the channel component
  private static final String VALINDEX_PORT_ID = ".valndx"; //msn 9-1-09
  private Port nodePort = addPort(NODE_PORT_ID, false);
  private Port configPort = addPort(CONFIG_PORT_ID, false);
  private Port channelPort = addServerPort(CHANNEL_PORT_ID);
   private Port valndxPort = addServerPort(VALINDEX_PORT_ID);//msn 9-1-09
  {
     removeDefaultUpPort();
     removeDefaultDownPort();
     removeTimerPort();
  }

  /** The grid size along X-axle */
  public double dX;

  /** The grid size along Y-axle */
  public double dY;

  /** The largest x coordinate value of the simulated area */
  public double maxX;
  /** The largest y coordinate value of the simulated area */
  public double maxY;
  /** The smallest x coordinate value of the simulated area */
  public double minX;
  /** The smallest y coordinate value of the simulated area */
  public double minY;

  /** Number of subareas along the X-axle. */
  public int m;
  /** Number of subareas along the X-axle. */
  public int n;

  /**
  * Each element of this two dimensional array is
  * a vector to hold the ids of all mobile nodes
  * currently are located in that subarea.
  */
  public Vector g[][];
  double cf[] = new double[30]; //msn 22-12-08
  long[] nodeList;//msn 9-1-09
     public int NoSrc; //msn 25-1-09
     double Xsrc[] = new double[30];//msn 25-1-09
     double Ysrc[] = new double [30];//msn 25-1-09
  double Xfol[] = new double[30];//msn 25-1-09
     double Yfol[] = new double [30];//msn 25-1-09
  /**
  * Construction function.
  *
  * @param maxX_ the largest x coordinate value of the simulated area
  * @param minX_ the smallest x coordinate value of the simulated area
  * @param maxY_ the largest y coordinate value of the simulated area
  * @param minY_ the smallest y coordinate value of the simulated area
  * @param dX_ the grid size along X-axle
  * @param dY_ the grid size along Y-axle
```

```
*/
public NodePositionTracker(double maxX_, double minX_, double maxY_, double minY_, double dX_,
        double dY_) {
   super();
   maxX = maxX_; minX = minX_; dX = dX_;
   maxY = maxY_; minY = minY_; dY = dY_;
   m = ((int)((maxX − minX)/dX)) + 1;
   n = ((int)((maxY − minY)/dY)) + 1;
   g = new Vector[m][n];
   //System.out.println("size of g "+m+" "+n);
   for (int i = 0; i < m; i ++ )
      for (int j = 0; j < n; j ++ )
         g[i][j] = new Vector();
}

public NodePositionTracker( ) {
   super();
}

public String info()
{
   StringBuffer sb0= new StringBuffer();
   sb0.append("Xrange/dX = " + minX + "−−" + maxX + "/" + dX + "\n");
   sb0.append("Yrange/dY = " + minY + "−−" + maxY + "/" + dY + "\n");
   StringBuffer sb= new StringBuffer();
   for (int i=0; i<m; i++)
      for (int j=0; j<n; j++)
         if (!g[i][j].isEmpty()) {
            sb.append(" (" + i + "," + j + "):");
            for (Iterator it=g[i][j].iterator(); it.hasNext(); ) {
               sb.append(it.next() + " ");
            }
            sb.append("\n");
         }
   if (sb.length() == 0)
      return sb0 + "No node has reported position.\n";
   else
      return sb0 + "Node Positions:\n" + sb;
}

/**
* Sets the grid parameters to divide the simulated area into subareas.
*
* @param maxX_ the largest x coordinate value of the simulated area
* @param minX_ the smallest x coordinate value of the simulated area
* @param maxY_ the largest y coordinate value of the simulated area
* @param minY_ the smallest y coordinate value of the simulated area
* @param dX_ the grid size along X−axle
* @param dY_ the grid size along Y−axle
*/
public void setGrid(double maxX_, double minX_, double maxY_, double minY_, double dX_, double dY_) {
   //System.out.println("setGrid maxX_ = " + maxX_ + " minX_ = " + minX_ + " maxY_ = " + maxY_ + "
         minY_ = " + minY_ + "dX = " + dX_ + " dY_ = " + dY_);
   maxX = maxX_; minX = minX_; dX = dX_;
   maxY = maxY_; minY = minY_; dY = dY_;
   m = ((int)((maxX − minX)/dX)) + 1;
   n = ((int)((maxY − minY)/dY)) + 1;
   g = new Vector[m][n];
   for (int i = 0; i < m; i ++ )
```

```
      for (int j = 0; j < n; j ++ )
         g[i][j] = new Vector();
}

public String getName() { return ”NodePositionTracker”; }

public void duplicate(Object source_)
{
   super.duplicate(source_);
   NodePositionTracker that_ = (NodePositionTracker)source_;
   maxX = that_.maxX; minX = that_.minX; dX = that_.dX;
   maxY = that_.maxY; minY = that_.minY; dY = that_.dY;

   m = ((int)((maxX − minX)/dX)) + 1;
   n = ((int)((maxY − minY)/dY)) + 1;
   g = new Vector[m][n];
   for (int i = 0; i < m; i ++ )
      for (int j = 0; j < n; j ++ )
         g[i][j] = new Vector();
}

protected void processOther(Object data_, Port inPort_)
{
   String portid_ = inPort_.getID();




   if (portid_.equals(NODE_PORT_ID)) {
   /*
      if (!(data_ instanceof PositionReportContract.Message)) {
         error(data_, ”processOther()”, inPort_, ”unknown object”);
         return;
      }
      */
      processReport(data_, inPort_);
   }
   else if (portid_.equals(CHANNEL_PORT_ID)) {
   /*
      if (!(data_ instanceof NeighborQueryContract.Message)) {
         error(data_, ”processOther()”, inPort_, ”unknown object”);
         return;
      }
      */
      processQuery(data_, inPort_);
   }
   else if (portid_.equals(VALINDEX_PORT_ID))//msn 9−1−09
   {
   processValIndex(data_,inPort_);
   }

   else
      super.processOther(data_, inPort_);
}

/**
* Processes the position update reports sent from the <code>MobilityModel</code>
* of the mobile nodes
*/
```

```
protected void processReport(Object data_, Port inPort_) {
  //debug("received position report" + data_.toString());

  if ( !(data_ instanceof PositionReportContract.Message) ) {
    error(data_, "processReport()", inPort_, "unknown object");
    return;
  }
  PositionReportContract.Message msg = (PositionReportContract.Message) data_;

  long id;
  double X, Y, Z, X0, Y0, Z0;

  id = msg.getNid();
  X = msg.getX();
  Y = msg.getY();
  Z = msg.getZ();
  X0 = msg.getX0();
  Y0 = msg.getY0();
  Z0 = msg.getZ0();
  cf[(int)id] = msg.getCF();//msn 22−12−08
      // drcl.Debug.debug ("node" + id + "X" + X + "Y" + Y + "\n");
      if ((int)id < NoSrc && X != 0.0 && Y!= 0.0)//msn 25−1−09
      {
      Xsrc[(int)id] = X;
      Ysrc[(int)id] = Y;
      }

      if ((int)id >=NoSrc && X != 0.0 && Y != 0.0)
      {
      Xfol[(int)id] = X;
      Yfol[(int)id] = Y;
      //for (jj = 0; jj < nodeList.length; jj ++)
      //drcl.Debug.debug ("node" + id + "xfol" + Xfol[(int)id] + "\n");
      }

  int i, j, i0, j0;

  if ( X == X0 && Y == Y0 ) { // first time or not moved yet
    //System.out.println("minX "+minX+ " X "+X+" dX "+dX);
    //System.out.println("minY "+minY+ " Y "+Y+" dY "+dY);
    i = (int)((X−minX)/dX);
    j = (int)((Y−minY)/dY);
    //System.out.println("i "+i+" j "+j);
    if ( g[i][j].contains(new Long(id)) == false )
      g[i][j].insertElementAt(new Long(id), 0);
  }
  else {
    i = (int)((X−minX)/dX);
    j = (int)((Y−minY)/dY);
    i0 = (int)((X0−minX)/dX);
    j0 = (int)((Y0−minY)/dY);

    g[i0][j0].remove(new Long(id));
    g[i][j].insertElementAt(new Long(id), 0);
  }

          // drcl.Debug.debug ("Node" + id + "cf=" + cf[(int)id] + "\n");//msn 22−12−08
}
```

```
/**
* Processes the neighbouring node inquery.
*/
protected void processQuery(Object data_, Port inPort_) {
   if ( !(data_ instanceof NeighborQueryContract.Message) ) {
      error(data_, "processQuery()", inPort_, "unknown object");
      return;
   }

   NeighborQueryContract.Message msg = (NeighborQueryContract.Message) data_;

   long id;
   double X, Y, Z;
// long[] nodeList; msn 9-1-09
   int nGrids;

   X = msg.getX();
   Y = msg.getY();
   Z = msg.getZ();
   nGrids = msg.getnGrids();

   int i, j, il, ir, jl, jr;
   i = (int)((X-minX)/dX);
   j = (int)((Y-minY)/dY);
   il = Math.max(0, i-nGrids);
   ir = Math.min(m-1, i+nGrids);
   jl = Math.max(0, j-nGrids);
   jr = Math.min(n-1, j+nGrids);

   int nn = 0;
   int ki, kj, kk, kv;
   for ( ki = il; ki <= ir; ki ++ )
      for ( kj = jl; kj <= jr; kj ++ )
         nn = nn + g[ki][kj].size();
   nodeList = new long[nn];
   kk = 0;
   for ( ki = il; ki <= ir; ki ++ ) {
      for ( kj = jl; kj <= jr; kj ++ ) {
         for ( kv = 0; kv < g[ki][kj].size(); kv ++ ) {
            nodeList[kk] = ((Long)(g[ki][kj].elementAt(kv))).longValue();
            kk = kk + 1;
         }
      }
   }
   // debug("query X = " + X + " Y = " + Y + " il = " + il + " i = " + i + " ir = " + ir + " jl = " + jl + " j =
         " + j + " jr = " + jr + " nodeList size = " + nn);
   channelPort.doSending(new NeighborQueryContract.Message(nodeList));


}

protected void processValIndex(Object data_, Port inPort_)//msn 9-1-09
{
   int jj; //msn 9-1-09
   double[] cff, //msn 9-1-09
         //double[] Xsrc1;
         //double[] Ysrc1;
         double[] Xfol1;
```

```
            double[] Yfol1;



        cff = new double[nodeList.length];//msn 9-1-09
              // Xsrc1 = new double[nodeList.length];
                 //Ysrc1 = new double[nodeList.length];
                 Xfol1 = new double[nodeList.length];
                 Yfol1 = new double[nodeList.length];

     for (jj = 0; jj < nodeList.length; jj ++)
          {
     cff[jj] = cf[(int)nodeList[jj]];
          //Xsrc1[jj] = Xsrc[(int)nodeList[jj]];
          //Ysrc1[jj] = Ysrc[(int)nodeList[jj]];
          Xfol1[jj] = Xfol[(int)nodeList[jj]];
          Yfol1[jj] = Yfol[(int)nodeList[jj]];

          }

     valndxPort.doSending(new NeighborQueryContract.Message(cff, Xsrc, Ysrc, Xfol1, Yfol1));//revised
            25-1-09




   }

     public void setNoSrc(int NoSrc_)//msn 25-1-09
     { NoSrc = NoSrc_ ; }



}
```

PositionReportContract

```
/* Modified by MS Nikitidis March 2009*/

// @(#)PositionReportContract.java 1/2004
// Copyright (c) 1998−2004, Distributed Real−time Computing Lab (DRCL)
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are met:
//
// 1. Redistributions of source code must retain the above copyright notice,
// this list of conditions and the following disclaimer.
// 2. Redistributions in binary form must reproduce the above copyright notice,
// this list of conditions and the following disclaimer in the documentation
// and/or other materials provided with the distribution.
// 3. Neither the name of "DRCL" nor the names of its contributors may be used
// to endorse or promote products derived from this software without specific
// prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
// AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
// ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR
// ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
// DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
// SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
// CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
// OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//

package drcl.inet.mac;

import drcl.comp.*;
import drcl.data.*;
import drcl.net.*;
import drcl.util.ObjectUtil;
import drcl.util.StringUtil;

/**
 * The PositionReportContract contract. This contract defines the message format
 * between <code>MobileModel</code> and <code>NodePositionTracker</code> as well as
 * between <code>MobileModel</code> and <code>WirelessPhy</code>.
 *
 * @see MobilityModel
 * @see NodePositionTracker
 * @see WirelessPhy
 * @author Ye Ge
 */
public class PositionReportContract extends Contract
{
  public static final PositionReportContract INSTANCE = new PositionReportContract();

  public PositionReportContract()
  { super(); }

  public PositionReportContract(int role_)
```

```
{ super(role_); }

public String getName()
{ return "Position Report Contract"; }

public Object getContractContent()
{ return null; }


  /**
   * Constructs a position report message and sends it through out_ port.
   *
   *@param X_ the x coordinate of the reporting node current position.
   *@param Y_ the x coordinate of the reporting node current position.
   *@param Z_ the x coordinate of the reporting node current position.
   *@out_ the port through which to send the constructed message.
   *
   */
public static void report(double X_, double Y_, double Z_, Port out_)
{ out_.doSending(new Message(X_, Y_, Z_)); }

/** The position report message. */
  public static class Message extends drcl.comp.Message
{
        long nid; // node id
        double X, Y, Z; // new position
        double X0, Y0, Z0; // original position
      double cur_frame; // msn 22−12−08 best frame by node

  public Message ()
  {}

        /** Constructor
          *
          *@param nid_ the node Id
          *@param X_ the x coordinate of the node current position
          *@param Y_ the y coordinate of the node current position
          *@param Z_ the z coordinate of the node current position
          *@param X0_ the x coordinate of the node previous position
          *@param Y0_ the y coordinate of the node previous position
          *@param Z0_ the z coordinate of the node previous position
          */
  public Message (long nid_, double X_, double Y_, double Z_, double X0_, double Y0_, double Z0_)
  {
    nid = nid_; X = X_; Y = Y_; Z = Z_; X0 = X0_; Y0 = Y0_; Z0 = Z0_;
  }

        /**
          * This is only used while MobilityModel responding WirelessPhy's query and reporting its own position.
          */
        public Message (double X_, double Y_, double Z_)
  {
    nid = −1;
            X = X_; Y = Y_; Z = Z_;
            X0 = X_; Y0 = Y_; Z0 = Z_;
  }

  public Message (long nid_, double cur_frame_)//msn 22−12−08
  {
```

```
        nid = nid_;

        cur_frame = cur_frame_;
        }

          public long getNid() { return nid; }
    public double getX() { return X; }
    public double getY() { return Y; }
    public double getZ() { return Z; }
    public double getX0() { return X0; }
    public double getY0() { return Y0; }
    public double getZ0() { return Z0; }
    public double getCF() { return cur_frame; }//msn 22−12−08

    /*
    public void duplicate(Object source_)
    {
      Message that_ = (Message)source_;
            nid = that_.nid; X = that_.X; Y = that_.Y; Z = that_.Z; X0 = that_.X0; Y0 = that_.Y0; Z0 = that_.Z0
                ;
    }
    */

    public Object clone()
    { return new Message(nid, X, Y, Z, X0, Y0, Z0); }

    public Contract getContract()
    { return INSTANCE; }

    public String toString(String separator_)
    {
      return ”Position Report:” + separator_ + ”nid=” + nid + separator_ + ”X=” + X + separator_ + ”Y=” + Y +
          separator_ + ”Z=” + Z + separator_ + ”X0=” + X0 + separator_ + ”Y0=” + Y0 + separator_ + ”Z0=”
          + Z0;
    }
  }
}
```