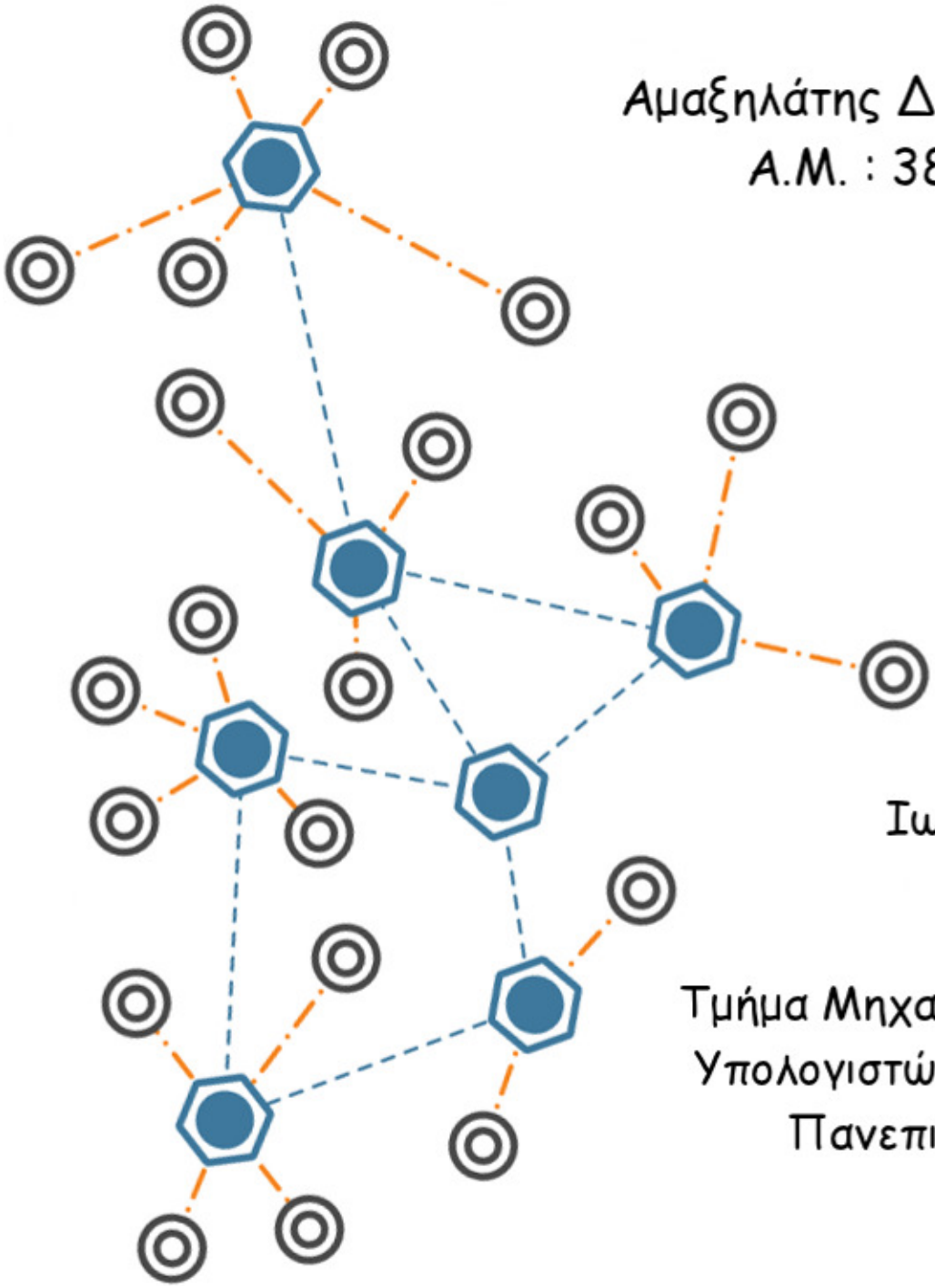
Κατανεμημένοι Αλγόριθμοι Ετερογενών Αδόμητων Ασύρματων Δικτύων

Διπλωματική Εργασία

Αμαξηλάτης Δημήτριος
Α.Μ. : 3891



Επιβλέπων:
Χρηστος Ζαρολιάγκης
Καθηγητής

Συνεπιβλέπων:
Ιωάννης Χατζηγιαννάκης

Τμήμα Μηχανικών Ηλεκτρονικών
Υπολογιστών και Πληροφορικής
Πανεπιστήμιο Πατρών

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τους καθηγητές μου κ. Χρήστο Ζαρολιάγκη και κ. Ιωάννη Χατζηγιαννάκη, τόσο για την διεκπεραίωση της παρούσας εργασίας όσο και για την πολύτιμη καθοδήγηση που μου παρείχαν τα τελευταία χρόνια των σπουδών μου. Τους ευχαριστώ ιδιαίτερα για τις συμβουλές που μου έδωσαν και για τις γνώσεις που κέρδισα από την πολυετή εμπειρία τους στο χώρο της Πληροφορικής.

Επίσης, τους συναδέλφους μου με τους οποίους συνεργάστηκα για θέματα λιγότερο ή περισσότερο σχετικά με την διπλωματική αυτή κατά την διάρκεια της υλοποίησής της.

Τους φίλους μου με τους οποίους πέρασα τα τελευταία 5 χρόνια της ζωής μου στην Πάτρα για όλα όσα περάσαμε και ελπίζω η φιλία μας να διατηρηθεί και στο μέλλον.

Τέλος, ευχαριστώ μέσα από τα βάθη της καρδιάς μου την οικογένειά μου για όλα όσα έχουν κάνει για μένα. Η υποστήριξη που παρέχουν σε κάθε βήμα της ζωής μου είναι ανεκτίμητη και ελπίζω κάποια μέρα να τους το ανταποδώσω.

Αμαξηλάτης Δημήτριος

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο και σημασία

Μεσα στα τελευταία χρόνια τα Ασύρματα Δίκτυα Αισθητήρων (Wireless Sensor Networks) έχουν κερδίσει το ενδιαφέρον στην επιστήμη της πληροφορικής τόσο στην βιομηχανία όσο και στον ακαδημαϊκό χώρο όχι μόνο από την θεωρητική πλευρά αλλά και από πρακτικές εφαρμογές. Συνήθως αποτελούνται από μεγάλο αριθμό τυχαία και αραιά τοποθετημένων αυτόνομων συσκευών με περιορισμένους πόρους και υπολογιστική ισχύ. Είναι ικανοί να αναγνωρίσουν χαρακτηριστικά του περιβάλλοντος τους όπως θερμοκρασία ή υγρασία με ειδικούς μηχανισμούς που διαθέτουν και μπορούν να επικοινωνήσουν μεταξύ τους χρησιμοποιώντας ασύρματες συνδέσεις κοντινών αποστάσεων (όπως Bluetooth ή Zigbee Radios) ή ακόμη και σε νεότερες συσκευές να συνδεθούν σε μεγαλύτερα δίκτυα ή ακόμη και στο Διαδίκτυο (πχ με χρήση Ethernet).

Τέτοια δίκτυα μπορούν να τοποθετηθούν παντού και μπορούν να δώσουν πληροφορίες για μετρήσεις και στοιχεία τα οποία συλλέγουν κατά τη λειτουργία τους ή να εκτελέσουν εντολές που τους μεταβιβάζονται από μεγάλες αποστάσεις ή να τις αποφασίσουν αυτόνομα ανάλογα με τις συνθήκες που αντιλαμβάνονται. Παραδείγματα χρήσης τέτοιων συσκευών μπορούν να είναι :

- η χρήση μέσα σε ένα κτίριο για τον έλεγχο του φωτισμού ή του κλιματισμού με στόχο τον περιορισμό των συνολικών απαιτήσεων ενέργειας,
- η χρήση τους μέσα σε ένα κλειστό ή ανοιχτό χώρο για λόγους α-

σφαλείας (άνοιγμα πόρτας, αναγνώριση κίνησης ή έλεγχος πρόσβασης με ηλεκτρονικές κλειδαριές),

- η χρήση τους μέσα σε μεγάλες εκτάσεις όπως δάση για την έγκαιρη ειδοποίηση των αρχών σε περίπτωση φωτιάς,

- η χρήση τους σε περιβάλλοντα όπου ο άνθρωπος δεν μπορεί να επιβιώσει ή δεν μπορεί να προσεγγίσει όπως ηφαίστεια, εμπόλεμες ζώνες ή τον βυθό της θάλασσας για την παρακολούθηση την κατασκοπία ή την συλλογή πληροφοριών για το περιβάλλον και τις συνθήκες που επικρατούν.

Τα δίκτυα αισθητήρων αποτελούν λοιπόν ένα πολύ χρήσιμο εργαλείο για ένα πολύ μεγάλο εύρος εφαρμογών και απλοποιούν σε μεγάλο βαθμό την χρήση ανθρώπινης επέμβασης. Ωστόσο δεν ισχύει το ίδιο και για την εγκατάσταση , τον προγραμματισμό και την διαχείριση τέτοιων δικτύων. Ο μεγάλος αριθμός των συσκευών που υπάρχουν, η χρήση τεχνολογιών όπως οι συνδέσεις κοντινών αποστάσεων (οι οποίες συνήθως είναι αναξιόπιστες) καθώς και οι ανάγκες συντονισμού και προγραμματισμού αυτών συσκευών δίνουν μεγάλη σημασία στην δημιουργία αποδοτικών και σταθερών σε όλη την περίοδο λειτουργίας τους αλγορίθμων. Οι ανάγκες αυτές είναι ένα ιδιαίτερα σημαντικό κίνητρο για τους προγραμματιστές καθώς πρόκειται για ένα νέο και ενδιαφέρον πεδίο εργασίας.

1.2 Στόχος

Στόχος της Διπλωματικής αυτής εργασίας είναι η μελέτη της σύγχρονης βιβλιογραφίας σχετικής με την υλοποίηση και την κατηγοριοποίηση των αλγορίθμων clustering. Με βάση τη βιβλιογραφία αυτή στόχος είναι σχεδιαστεί ένα πλαίσιο υλοποίησης των αλγορίθμων το οποίο τυποποιεί την δομή και την λειτουργία τους. Να δημιουργηθεί δηλαδή μια διεπαφή για την την επικοινωνία των αλγορίθμων αυτών με άλλους αλγορίθμους καθώς και ο προσδιορισμός των δομικών συστατικών που κάθε αλγόριθμος διαθέτει.

1.3 Συνεισφορά

Η συνεισφορά αυτής της εργασίας στο πεδίο των WSN είναι ένα πλαίσιο υλοποίησης αλγορίθμων clustering για την γρηγορότερη και ευκολότερη ανάπτυξη τους αλλά και για μια τυποποίηση των συναρτήσεων διεπαφής τόσο με τον έξω κόσμο και άλλους αλγορίθμους όσο και ανάμεσα στα υποσυστήματα που απαρτίζουν τον κάθε αλγόριθμο.

Επίσης ένα ακόμη σημαντικό χαρακτηριστικό του πλαισίου της υλοποίησης είναι η δυνατότητα εναλλαγής των αντίστοιχων τμημάτων διαφορετικών αλγορίθμων για την δημιουργία αλγορίθμων υβριδίων για να εκμεταλλευτούμε θετικά χαρακτηριστικά άλλων αλγορίθμων που μπορεί να μας βοηθούν στην εκάστοτε εφαρμογή.

1.4 Οργάνωση της Εργασίας

Το **2ο κεφάλαιο** αποτελεί μια εισαγωγή στην περιοχή των ετερογενών ασύρματων δικτύων και της Ομαδοποίησης (Clustering). Παρουσιάζεται ο σκοπός της δημιουργίας clusters μέσα σε ένα δίκτυο αισθητήρων, η εσωτερική δομή των clusters και η συνεισφορά τους στην επίλυση πραγματικών περισσότερο συνδυαστικών προβλημάτων. Επίσης γίνεται αναφορά σε σχετική βιβλιογραφία που χρησιμοποιήθηκε ως βάση για την ανάπτυξη των θεμάτων αυτής της διπλωματικής εργασίας και των αλγορίθμων που θα υλοποιηθούν στην συνέχεια.

Στο **3ο κεφάλαιο** παρουσιάζονται τα συστατικά μέρη στα οποία διαχωρίζουμε τους clustering αλγορίθμους και τα οποία θα χρησιμοποιήσουμε στην συνέχεια για να τους διασπάσουμε σε ανεξάρτητα κομμάτια τα οποία συνεργαζόμενα θα αποτελούν υλοποιήσεις των διαφορετικών αλγορίθμων. Επίσης παρουσιάζουμε με τον τρόπο με τον οποίο αυτές οι ανεξάρτητες μονάδες μπορούν να χρησιμοποιηθούν για την υλοποίηση νέων αλγορίθμων με την ελάχιστη δυνατή επιπλέον εργασία από τον προγραμματιστή. Επίσης λόγω της χρήσης της βιβλιοθήκης Wiselib δείχνουμε ότι οι υλοποιήσεις αυτές είναι δυνατόν να εναλλάσσονται μέσα σε ήδη υλοποιημένους αλγορίθμους για να μεταβάλλεται η συμπεριφορά τους. Τέλος παρουσιάζονται οι διεπαφές που το κάθε κομμάτι ενός υλοποιημένου αλγορίθμου πρέπει να περιέχει έτσι ώστε να είναι

συμβατό με τα πρότυπα που απαιτεί η Wiselib και να είναι δυνατή η εναλλαγή ολόκληρων των αλγορίθμων μέσα στα πλαίσια μιας σύνθετης εφαρμογής χωρίς καμία αλλαγή στον κώδικα παρά μόνο στην δήλωση των αλγορίθμων.

Στο **4ο κεφάλαιο** παρουσιάζουμε πειραματικά αποτελέσματα που αποδεικνύουν ότι η υλοποιήσεις των αλγορίθμων σύμφωνα με την αρχιτεκτονική δομή που προτείνουμε είναι απόλυτα σύμφωνες με τις αρχικές περιγραφές των δημιουργών τους. Για να το πετύχουμε αυτό επαναλαμβάνουμε όσα πειράματα είχαν εκτελέσει οι δημιουργοί τους και περιγράφουν στις εργασίες τους.

Στο **5ο κεφάλαιο** παρουσιάζουμε όλα τα πειραματικά αποτελέσματα που έχουν προκύψει από διάφορες κατηγορίες πειραμάτων που επιλέχθηκαν για την ορθότερη περιγραφή των αλγορίθμων. Τα πειράματα αυτά έχουν γίνει τόσο σε προσομοιωτές ασύρματων δικτύων όσο και σε πραγματικές συσκευές που ήταν διαθέσιμες. Στα πειράματα με την χρήση προσομοιωτών μελετάται η κλιμακοσιμότητα των υλοποιήσεων. Στις πραγματικές συσκευές προσπαθούμε να καταλάβουμε πως αντιδρά ο αλγόριθμος σε αλλαγές του περιβάλλοντός του αλλά και σε προβλήματα που είναι αναμενόμενα στον χώρο των ασυρμάτων δικτύων αισθητήρων.

Στο **6ο κεφάλαιο** παραθέτονται τα συμπεράσματα της διπλωματικής εργασίας. Επίσης γίνεται μια αναφορά σε μελλοντικούς στόχους και εφαρμογές των αλγορίθμων που έχουν ήδη υλοποιηθεί ή θα υλοποιηθούν στο μέλλον.

Κεφάλαιο 2

Βασικές Ενοιες

2.1 Ετερογενή Ασύρματα δίκτυα αισθητήρων

Στα σύνχρονα δίκτυα αισθητήρων που περιγράψαμε και στην Εισαγωγή παρατηρούμε αρκετά συχνά το φαινόμενο της ετερογένειας. Με τον όρο αυτό εννοούμε ότι το σύνολο του δικτύου δεν αποτελείται από συσκευές ίδιου τύπου. Δηλαδή από συσκευές που χρησιμοποιούν διαφορετικά μέσα επικοινωνίας, διαφορετικά λειτουργικά συστήματα , προέρχονται συχνά από διαφορετικούς κατασκευαστές και γενικά έχουν διαφορετικές δυνατότητες μεγέθη και υπολογιστική ισχύ. Ωστόσο συνήθως αυτό που θέλουμε είναι να μην χρειάζεται να υλοποιήσουμε τις εφαρμογές μας για κάθε μία από τις συσκευές ξεχωριστά, αλλά να ακολουθούμε τις αρχές των εφαρμογών κατανεμημένων συστημάτων (δηλαδή μια υλοποίηση για όλο το δίκτυο). Στόχος λοιπόν είναι να θεωρούμε το διαθέσιμο δίκτυο ως ένα γράφο στον οποίο οι κορυφές αντιστοιχούν σε αισθητήρες και οι ακμές αντιστοιχούν σε συνδέσεις ανάμεσα σε αυτούς μέσω κάποιου ασύρματου δικτύου για τα χαρακτηριστικά του οποίου αδιαφορούμε πλήρως.

2.2 Clustering

Με τον όρο Clustering στα ασύρματα δίκτυα αισθητήρων αναφερόμαστε στον σχηματισμό ομάδων (clusters) με συγκεκριμένα προκαθορισμένα χαρακτηριστικά. Οι ομάδες αυτές σχηματίζονται μετά από επικοινωνία ανάμεσα στους αισθητήρες και ανταλλαγή πληροφορίας για

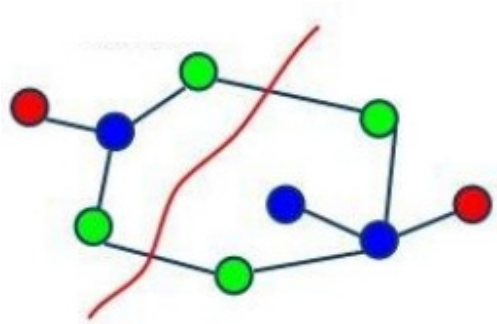
τις δυνατότητες και την κατάστασή τους. Μέσα στις ομάδες που δημιουργούνται υπάρχουν 3 διακριτοί ρόλου που μπορεί να αναλάβει ο κάθε κόμβος. Αυτοί είναι:

- **Αρχηγός (ClusterHead)** Υπάρχει ένας μονάχα σε κάθε cluster και είναι υπεύθυνος για να εκπροσωπεί όλους τους υπόλοιπους στο δίκτυο.

- **Σύνδεσμος (Gateway)** Έτσι ονομάζουμε τους κόμβους που συνδέονται με τουλάχιστον ένα κόμβο διαφορετικού cluster.

- **Απλός (Member)** Κάθε κόμβος ο οποίος έχει απλά γίνει μέλος του cluster.

Οι ρόλοι και μια πιθανή ανάθεσή τους σε ένα τυχαίο δίκτυο όπου έχουν διαμορφωθεί 2 ομάδες φαίνονται στο Σχήμα 2.1 όπου με κόκκινο χρώμα εμφανίζονταιοι 2 εκλεγμένοι ClusterHeads με πράσινο οι Gateways και με μπλε τα απλά Members.



Σχήμα 2.1: Παράδειγμα δικτύου όπου έχουν διαμορφωθεί 2 clusters

Η δημιουργία των clusters διαμορφώνει μια ιεραρχία μέσα στο δίκτυο. Η ιεραρχία αυτή είναι ιδιαίτερα σημαντική για πληθώρα εφαρμογών όπως την συγκέντρωση μετρήσεων (Data Aggregation), την παρακολούθηση στόχων(Target Tracking) και την ασφαλή επικοινωνία (κρυπτογράφηση μηνυμάτων μέσα στο cluster). Επίσης βοηθά στον περιορισμό των απαιτήσεων επικοινωνίας τόσο εντός όσο και εκτός των ομάδων καθώς τα μηνύματα προωθούνται μέσω συγκεκριμένων διαδρομών (από τους κόμβους προς τους αρχηγούς των ομάδων και αντίστροφα) αλλά και στην εξισορρόπηση φόρτου εργασίας. Καθώς οι αισθητήρες αυτοί είναι συχνά τοποθετημένοι σε δυσπρόσιτες περιοχές με

ακραίες συνθήκες (όπως δάση, πεδία στρατιωτικών επιχειρήσεων, βουνά) και λειτουργούν με περιορισμένους πόρους είναι ιδιαίτερα σημαντικό να περιορίσουμε όσο περισσότερο γίνεται την οποιαδήποτε σπατάλη ενέργειας. Η μείωση του όγκου της επικοινωνίας και η καλύτερη οργάνωση του δικτύου έχει παρατηρηθεί οδηγεί σε σημαντική αύξηση της διάρκειας ζωής των αισθητήρων και συνεπώς της συνολικής εγκατάστασης.

Ο τρόπος με τον οποίο σχηματίζονται τα clusters είναι και αυτός που προσδίδει ιδιαίτερα χαρακτηριστικά σε κάθε αλγόριθμο. Οι αλγόριθμοι μπορεί να είναι είτε ντετερμινιστικοί (πχ σχηματίζουν ομάδες χρησιμοποιώντας κάποια ντετερμινιστική ποσότητα όπως την απόσταση μεταξύ τους), πιθανοτικοί (σχηματίζουν τις ομάδες χρησιμοποιώντας σε κάθε απόφαση γεννήτριες τυχαίων αριθμών) ή και να συνδυάζουν χαρακτηριστικά και των 2 κατηγοριών (πχ Εκλογή του κόμβου με την περισσότερη ενέργεια ως ClusterHead και αποδοχή κάθε πρόσκλησης για συμμετοχή στην ομάδα με 50% πιθανότητα).

2.3 Σχετική Βιβλιογραφία

Τα τελευταία χρόνια έχει παρουσιαστεί και μελετηθεί μια μεγάλη συλλογή από αλγορίθμους clustering. Στην σχετική βιβλιογραφία υπάρχουν αρκετές έρευνες (π.χ. [2, 7, 20, 12]) που συγκρίνουν αρκετούς αλγορίθμους και καταλήγουν ότι υπάρχουν αρκετές ομοιότητες σε επιμέρους διαδικασίες τους. Ακόμη συχνά καταλήγουν σε μια κατηγοριοποίηση των αλγορίθμων ως προς τον τρόπο λειτουργίας και σχεδίασης.

Για παράδειγμα, στο [2] εξετάζονται 54 αλγόριθμοι ως προς τον αρχιτεκτονικό τους σχεδιασμό και τις παραμέτρους που χρησιμοποιούνται στα εξεταζόμενα πρωτόκολλα όπως την δυναμική του δικτύου, τις δυνατότητες των κόμβων και την υπολογιστική τους ισχύ. Καταλήγουν λοιπόν σε μια κατηγοριοποίηση των αλγορίθμων ως προς τον στόχο που έχουν σε αλγορίθμους: Εξισορρόπησης φόρτου, Ανοχής σε σφάλματα, Αυξημένης επικοινωνίας, Ελαχιστοποίησης αριθμού cluster και Μεγιστοποίησης της ζωής του δικτύου.

Με αντίστοιχο τρόπο στο [7] παρουσιάζονται 44 αλγόριθμοι και επικεντρώνονται στον αριθμό των clusters, στην επικάλυψη των clusters, στον τρόπο εκλογής των ClusterHeads, στην κινητικότητα των κόμβων και στην χρονική πολυπλοκότητα. Επίσης καταλήγουν σε ένα διαχωρισμό των αλγορίθμων σε Πιθανοτικούς και Ντετερμινιστικούς ως προς τον τρόπο εκλογής των ClusterHeads και σχηματισμού των clusters.

Μελετώντας συνολικά 8 ([2, 7, 20, 12, 13, 17, 9, 16]) ερευνητικές εργασίες που παρουσιάζουν συνολικά 90 αλγορίθμους αναγνωρίζουμε 2 βασικές περιοχές που εμφανίζονται σε κάθε αλγόριθμο. Την εκλογή των ClusterHeads και την μέθοδο σχηματισμού των clusters. Αυτός ο διαχωρισμός αποτελεί τον βασικό πυλώνα της γενικής σχεδιαστικής μεθόδου διαχωρισμού των clustering αλγορίθμων σε ανεξάρτητες υπομονάδες. Ωστόσο η ανάπτυξη των υπομονάδων βασίζεται στον κάθε αλγόριθμο ξεχωριστά, ανάλογα με τις ιδιαιτερότητες που αυτός παρουσιάζει. Στην συνέχεια παρουσιάζονται κάποιοι αλγόριθμοι που συναντώνται συχνά στην βιβλιογραφία, θεωρούνται βασικοί σε αυτόν τον τομέα και ο τρόπος με τον οποίο ενσωματώνονται στην προτεινόμενη αρχιτεκτονική.

2.4 Κατηγοριοποίηση Αλγορίθμων

Τα 2 αυτά τμήματα των αλγορίθμων που συναντήσαμε στην μελέτη όλης αυτής της βιβλιογραφίας τα κατηγοριοποιήσαμε στο σύνολο της βιβλιογραφίας και παρουσιάζονται στους Πίνακες 2.1 και 2.2. Ο διαχωρισμός αυτός είναι γενικός και στην συνέχεια θα εξειδικευτεί σε μεγαλύτερο βαθμό καθώς θα δοθούν συγκεκριμένα παραδείγματα υλοποίησης.

Τρόπος Υλοποίησης	Εργασία
Τυχαίος-Πιθανοτικός	[19], [33], [5], [23], [32], [25], [22]
Συνδεσιμότητα	[24], [27], [10]
Με χρήση Ταυτοτήτων	[4], [1], [6]
Ενέργεια	[32], [25], [30], [29], [28], [18]
Μέτρηση Αισθητήρα	[31]
Ζυγισμένου Συνδιασμού	[11], [15], [26]

Πίνακας 2.1: Τρόπος Εκλογής Αρχηγών

2.5 Επιλεγμένοι αλγόριθμοι

Στα πλαίσια αυτής της διπλωματικής επιλέχθηκαν και υλοποιήθηκαν από το σύνολο της βιβλιογραφίας οι παρακάτω αλγόριθμου που παρου-

Μέθοδος Υλοποίησης	Εργασία
Απόσταση από την Αρχηγό	[19], [33], [6], [27], [4], [5], [32], [6], [24], [3], [23]
Ενέργεια Αρχηγού	[30], [29], [28], [18]
Ποιότητα σύνδεσης	[22]
Χρόνος Ζωής Αρχηγού	[25]
Μέρτηση Αρχηγού	[31]
Βάρος Αρχηγού	[11], [15], [26]

Πίνακας 2.2: Τρόπος Σχηματισμού Ομάδων

σιάζουν τα χαρακτηριστικά του πλαισίου υλοποίησης που θα προταθεί στο Κεφάλαιο 3:

LCA [4] Κάθε κόμβος κατά την αρχικοποίηση του αλγορίθμου εκλέγεται αρχηγός με πιθανότητα p. Στην συνέχεια όλοι οι εκλεγμένοι ως αρχηγοί κόμβοι γίνονται ClusterHeads στο δικό τους cluster και διαφημίζουν στο δίκτυο την παρουσία τους. Οι υπόλοιποι μη εκλεγμένοι κόμβοι αποφασίζουν να μπουν στον πλησιέστερο σε αυτούς ClusterHead. Η εξάπλωση της πληροφορίας για την ύπαρξη του ClusterHead στο δίκτυο γίνεται από τους νέους κόμβους που τον αποδέχτηκαν και συνεχίζεται μέχρι το πολύ k συνδέσεις μακριά, σχηματίζοντας έτσι clusters k συνδέσεων. Οι παράμετροι p και k είναι σταθερές σε όλη τη διάρκεια εκτέλεσης του αλγορίθμου.

Leach [19] Πρόκειται για ένα αλγόριθμου που σε προκαθορισμένα διαστήματα εναλλάσσει τους εκλεγμένους ClusterHeads με σκοπό την εξισορρόπηση φόρτου που προκύπτει από τον ρόλο του ClusterHead μέσα στο δίκτυο. Έτσι κάθε κόμβος εκλέγεται αρχηγός με μια πιθανότητα p η οποία όμως μεταβάλλεται σε κάθε φάση επανεκλογής αρχηγών με διαφορετικό τρόπο για κάθε κόμβο. Για τον υπολογισμό του p χρησιμοποιείται μια ευρετική συνάρτηση που πετυχαίνει την εναλλαγή αυτή. Μετά την εκλογή των ClusterHeads ο σχηματισμός των clusters γίνεται με τον ίδιο τρόπο σε σχέση με πριν απλά ως επιπλέον κριτήριο για την αποδοχή χρησιμοποιείται η ποιότητα της σύνδεσης για την ελαχιστοποίηση των απαιτήσεων σε ενέργεια. Εδώ η παράμετρος k είναι σταθερή αλλά το p μεταβάλλεται διαφορετικά για κάθε κόμβο.

Moca [33] Ο αλγόριθμος αυτός έχει ως στόχο την δημιουργία επικαλυπτόμενων cluster. Πέρα από αυτή την διαφορά λειτουργεί σχεδόν με πανομοιότυπο τρόπο σε σχέση με τον LCA καθώς οι ClusterHeads εκλέγονται με πιθανότητα p κατά την αρχικοποίηση, και ο σχηματισμός των ομάδων γίνεται με προώθηση της πληροφορίας k συνδέσεις μακριά από τον ClusterHead. Η ιδιότητά του να δημιουργεί επικαλυπτόμενα clusters δημιουργεί την ανάγκη ύπαρξης σε κάθε κόμβο δομών για αποθήκευση πληροφορίας για όλα τα clusters που συμμετέχει ο κόμβος και όχι μόνο ενός. Οι παράμετροι p και k είναι σταθερές σε όλη τη διάρκεια εκτέλεσης του αλγορίθμου.

Maxmind [3] Αυτός ο αλγόριθμος είναι κάπως διαφορετικός σε σχέση με τους προηγούμενους. Διαθέτει μια ολοκληρωμένη λογική για την τοποθέτηση των ClusterHeads σε βέλτιστες θέσεις μέσα στο δίκτυο. Για τον σκοπό αυτό πραγματοποιεί μια ανταλλαγή πληροφοριών για 2 k φάσεις μέσα στο δίκτυο και στην συνέχεια αποφασίζει ταυτόχρονα την θέση των ClusterHeads και τον σχηματισμό των clusters. Η μορφή αυτή του προσδίδει κάποια ιδιαίτερα χαρακτηριστικά σε σχέση με τους υπόλοιπους αλγορίθμους που συναντήσαμε στην βιβλιογραφία. Παρόλα αυτά ακόμη και αλγόριθμοι όπως ο Maxmind μπορούν να ενσωματωθούν στο προτεινόμενο πλαίσιο.

Fronts [14] Επίσης υλοποιήθηκε και ένας επιπλέον αλγόριθμος ο οποίος δεν αντιστοιχεί σε κάποιο συγκεκριμένο αλγόριθμο της βιβλιογραφίας αλλά έχει ιδιαίτερα χαρακτηριστικά που απαιτούνταν στα πλαίσια του έργου FRONTS. Για τον αλγόριθμο αυτό απαιτούνταν η επιλογή των ClusterHead κόμβων με μοναδικό κριτήριο την ελάχιστη μοναδική ταυτότητά στην κ γειτονιά των κόμβων. Στην συνέχεια οι υπόλοιποι μη εκλεγμένοι κόμβοι απαιτούνταν να συνδέονται με τον κόμβο αρχηγό που είχε την μικρότερη ταυτότητα μέσα σε αυτή την κ γειτονιά. Ο αλγόριθμος αυτός αναπτύχθηκε σε συνδυασμό με άλλους αλγορίθμους του έργου για τον έλεγχο και την λειτουργία ενός ασύρματου δικτύου 40 αισθητήρων (iSense και Xbee).

Κεφάλαιο 3

Προτεινόμενη Αρχιτεκτονική

3.1 Αρχιτεκτονική Δομή

Όπως αναφέρθηκε πριν σε όλους τους αλγορίθμους παρατηρούνται 2 βασικά κομμάτια που σε πολλούς από αυτούς παρουσιάζουν αρκετές ομοιότητες. Τα 2 μέρη αυτά είναι:

Η επιλογή των κόμβων αρχηγών (Cluster Head Decision, CHD) που συνήθως γίνεται στην αρχικοποίηση του αλγορίθμου ή μετά από την ανταλλαγή κάποιας πληροφορίας (όπως τις μοναδικές ταυτότητες κάθε κόμβου, την υπάρχουσα ενέργεια, κάποια είσοδο από κάποιο αισθητήρα). Το τμήμα αυτό είναι ανεξάρτητο από την υπόλοιπη λειτουργία του αλγορίθμου (στο κομμάτι της διαχείρισης του cluster και της αποφυγής σφαλμάτων) και καλείται μόνον όταν υπάρχει ανάγκη εκλογής νέων αρχηγών που σε ορισμένους αλγορίθμους είναι απαραίτητο για την εξισορρόπηση του φόρτου εργασίας.

Ο σχηματισμός των ομάδων (Cluster Formation ή Cluster Join Decision, JD) ξεκινά συνήθως μετά την εκλογή των ClusterHeads και γίνεται μέσω ανταλλαγής μηνυμάτων πρόσκλησης ή διαφήμισης (JOIN μηνύματα) από τους κόμβους αρχηγούς προς το υπόλοιπο δίκτυο. Κάποιες φορές λόγο ορισμένων ιδιόμορφων αλγορίθμων μπορεί να χρειαστεί επίσης κάποια επιπλέον επικοινωνία για την πλήρη ενημέρωση των δομών των ClusterHeads και την διόρθωση λάθος τοποθετημένων κόμβων. Στους περισσότερους αλγορίθμους το κομμάτι του σχηματισμού μοιάζει (με εξαίρεση κάποιες παραμέτρους) αλλά υπάρχουν και εδώ ειδικές περιπτώσεις όπως ο Maxmind που ο σχηματισμός των ομάδων δεν απαιτεί

καμία διαδικασία αλλά στην συνέχεια χρειάζονται κάποιες επιπλέον ενέργειες για την διόρθωση συγκεκριμένων ανωμαλιών.

Για να δοθεί η δυνατότητα υλοποίησης των αλγορίθμων για επικαλυπτόμενα και μη clusters υπάρχει και ένα 3^{o} τμήμα που ονομάζεται Iterator στον οποίο αποθηκεύονται σε κατάλληλες δομές οι πληροφορίες που προκύπτουν από την λειτουργία των άλλων τμημάτων. Στην περίπτωση των επικαλυπτόμενων clusters οι δομές του Iterator είναι περισσότερο πολύπλοκες για να πετύχουν τον στόχο αυτό. Επίσης η ύπαρξη του Iterator τμήματος διευκολύνει την ανάπτυξη αλγορίθμων που χρησιμοποιούν ειδικές δομές για την διατήρηση κάποιων επιπλέον δεδομένων όπως ο Maxmind.

Για την σύνδεση αυτών των τμημάτων χρησιμοποιείται ένα κεντρικό τμήμα το Core Component (CC) που είναι ουσιαστικά η υλοποίηση του αλγορίθμου και με αυτό επικοινωνούν οι χρήστες ή κάποιος άλλος αλγόριθμος για να πάρει την οποιαδήποτε πληροφορία για την κατάσταση του αλγορίθμου όπως το cluster στο οποίο ανήκει, τον ρόλο του στο cluster και την απόσταση από τον ClusterHead. Επίσης αυτό το τμήμα είναι επιφορτισμένο με την διαχείριση της κατάστασης του κόμβου και πραγματοποιεί όλες τις απαραίτητες ενέργειες για την προσαρμογή στις μεταβολές του δικτύου (Cluster Maintenance). Το CC ακολουθεί μια συγκεκριμένη μορφή και διαθέτει μια προκαθορισμένη δημόσια διεπαφή. Σε γενικές γραμμές οι διεργασίες που επιτελεί το CC είναι:

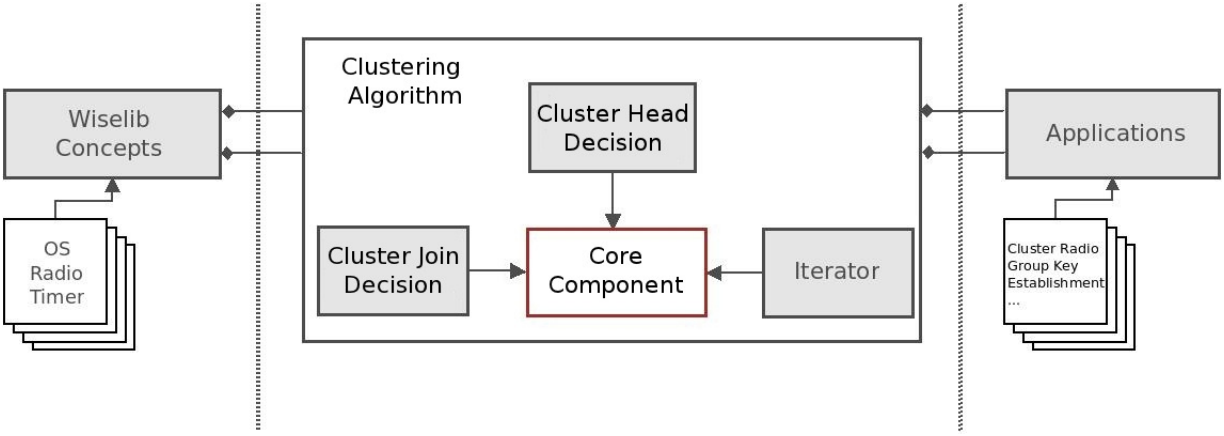
1. Χρησιμοποιεί το CHD για να καθοριστεί αν ο κόμβος είναι ClusterHead ή όχι.

2. Αν ο κόμβος είναι ClusterHead στέλνει JOIN μηνύματα με την βοήθεια του JD για να προσκαλέσει κόμβους στο cluster του.

3. Αν ο κόμβος δεν είναι ClusterHead περιμένει μέχρι να λάβει τα μηνύματα των ClusterHeads, τα προωθεί στο JD που λαμβάνει μια την απόφαση.

4. Όταν έχουν διαμορφωθεί τα cluster το CC ακούει τα γεγονότα που συμβαίνουν στην άμεση γειτονιά του (κίνηση κόμβων, απώλεια σύνδεσης) και κάνει αντίστοιχες διορθωτικές κινήσεις (αλλαγή cluster, δημιουργία νέου cluster).

5. Αν υπάρχει ανάγκη για περιοδική αναδιαμόρφωση των clusters το CC χρόνο-προγραμματίζει την επανεκτέλεση του αλγορίθμου από την αρχή.



Σχήμα 3.1: Σχηματική αναπαράσταση της Modular Αρχιτεκτονικής

Για το κομμάτι της διαχείρισης του cluster και της προσαρμογής στις αλλαγές του δικτύου χρησιμοποιείται ένας Neighbor Discovery αλγόριθμος της Wiselib (παρουσιάζεται στην ενότητα 3.2) ο οποίος με χρήση beacons (μηνυμάτων που επανεκπέμπονται περιοδικά) προσφέρει πληροφορίες για την ποιότητα των συνδέσεων με κάθε άλλο κόμβο της γειτονιάς, ενημέρωση για αλλαγές σε αυτήν (απώλεια κόμβων) καθώς και τη δυνατότητα αποστολής πληροφορίας (μέσω piggybacking) άλλων εφαρμογών στα beacons του.

Στο Σχήμα 3.1 φαίνεται η δομή των clustering αλγορίθμων με αυτή την Modular Αρχιτεκτονική. Το Core Component είναι το μόνο προσβάσιμο στοιχείο από άλλους αλγορίθμους (Εφαρμογές) και επίσης το μόνο που έχει άμεση πρόσβαση σε τμήματα του υλικού όπως το Radio για την αποστολή και λήψη μηνυμάτων ή τους Timers μέσω των αντίστοιχων Concepts της Wiselib. Τα υπόλοιπα τμήματα του αλγορίθμου επικοινωνούν μόνο με αυτό και όχι άμεσα μεταξύ τους. Για την επικοινωνία ανάμεσα στα modules χρησιμοποιούνται callbacks που θα αναλυθούν στην συνέχεια.

3.2 Wiselib: μια βιβλιοθήκη αλγορίθμων για ετερογενή δίκτυα Αισθητήρων

Τα δίκτυα αισθητήρων σήμερα αποτελούνται από διαφορετικών τύπων αισθητήρες και συνεπώς υπάρχει μεγάλη ανάγκη για την υλοποίηση αλγορίθμων που να μπορούν να εκτελεστούν όσο πιο εύκολα γίνεται σε

```
typedef wiselib::OSMODEL Os;
class ExampleApplication{
 void init(Os::AppMainParameter& value){
  radio_ = &wiselib::FacetProvider<Os, Os::Radio>::get_facet( value );
 }
 void start( void* ){
  Os::Radio::block_data_t message[2];
  radio_->send(0xffff, 2, message);
 }
 Os::Radio::self_pointer_t radio_;
};
```

Σχήμα 3.2: Παράδειγμα Generic Εφαρμογής στην Wiselib

μεγάλο πλήθος των διαθέσιμων πλατφορμών. Επίσης η υλοποίηση της modular αρχιτεκτονικής για τους αλγορίθμους clustering απαιτεί την υλοποίηση των τμημάτων με αντίστοιχο τρόπο έτσι ώστε να προσφέρουν το ίδιο interface και να είναι συμβατά μεταξύ τους.

Για να επιλύσουμε τα προβλήματα που προκύπτουν από την ετερογένεια των δικτύων αισθητήρων και την τμηματική αρχιτεκτονική χρησιμοποιούμε την Wiselib [8] . Είναι υλοποιημένη σε C++ και χρησιμοποιεί templates για την δημιουργία γενικών και παραμετροποιήσημων συναρτήσεων. Η επιλογή του επιθυμητού λειτουργικού συστήματος γίνεται την ώρα της μεταγλώττισης αυτόματα ανάλογα με την πλατφόρμα στην οποία θέλουμε να εκτελέσουμε τον αλγόριθμο. Η δυνατότητα αυτή προσφέρεται με τη χρήση των template κλάσεων.

Η Wiselib είναι μια βιβλιοθήκη αλγορίθμων για δίκτυα αισθητήρων αντίστοιχη των Boost και CGAL. Προσφέρει μια γενική διεπαφή για το λειτουργικό σύστημα που διευκολύνει την ανάπτυξη και μειώνει την ανάγκη επίλυσης προβλημάτων χαμηλού επιπέδου λόγω των διαφορετικών πλατφορμών. Έτσι η υλοποίηση γίνεται ανεξάρτητα της πλατφόρμας στόχου σε όλες τις διαθέσιμες πλατφόρμες της βιβλιοθήκης (iSense, Contiki, TinyOs, iPhone, Android και τον προσομοιωτή δικτύων αισθητήρων Shawn). Για παράδειγμα, για την ανταλλαγή μηνυμάτων ανάμεσα στις συσκευές υπάρχει υλοποιημένη η συνάρτηση send για κάθε μια από της πλατφόρμες με διαφορετικό τρόπο, αλλά διαθέσιμη στον προγραμματιστή μέσω μιας template διεπαφής Radio που δέχεται ως παράμετρο την υλοποίηση της πλατφόρμας που επιθυμούμε.

3.3 Ανάλυση των τμημάτων και διεπαφής

3.3.1 Core Component

Το βασικό μέρος που καθορίζει τον κάθε αλγόριθμο και του προσδίδει τα ιδιαίτερα χαρακτηριστικά που τον διαφοροποιούν από τους υπόλοιπους είναι το Core Component και έχει την γενική μορφή που περιγράφεται από το παρακάτω template:

```
template<typename OsModel,
  typename Radio, typename Timer, typename Debug,
  typename HeadDecision, typename JoinDecision, typename Iterator,
  typename NeighborDiscovery>
class CoreComponent {
public:
  void init(Radio&,Timer&,Debug&,CHD&,JD&,IT&,ND&);
  void enable(void);
  void disable(void);

  void set_XX_parameter(xx_parameter_t);
  void form_cluster(void * p);

  template<typename T, void(T::*TMethod)(uint8_t)>
    int reg_changed_callback(T* obj);

  node_id_t parent(void);
  cluster_id_t cluster_id(void);
  bool is_cluster_head(void);
  int hops(void);
  bool is_gateway(void);
  ...
};
```

Σχήμα 3.3: Διεπαφή CC

3.3.2 Cluster Head Decision

Το τμήμα εκλογής αρχηγών περιγράφεται από το επόμενο template. Βασικότερη συνάρτηση είναι η $calculate\_head$ που χρησιμοποιείται για την επανεκλογή αρχηγών.

3.3.3 Join Decision

Το τμήμα απόφασης συμμετοχής περιγράφεται περιέχει συναρτήσεις για την αποστολή μηνυμάτων για πρόσκληση σε ομάδα και αποδοχή ή α-

```
template<typename Radio, typename Debug>
class ClusterheadDecision {
public:
  void init(Radio& , Debug& );
  void enable(void);
  void disable(void);

  void set_xx_parameter(xx_parameter_t);
  bool is_cluster_head(void);
  bool calculate_head(void);
  size_t get_payload_length(int);
  XX_Msg_t get_xx_payload(void);
};
```

Σχήμα 3.4: Διεπαφή CHD

πόρριψη των προσκλήσεων ενώ κάθε φορά που λαμβάνεται μια πρόσκληση προωθείται στην συνάρτηση $join$ που χρησιμοποιώντας το μήνυμα αποφασίζει για την επόμενη ενέργεια (αποδοχή ή απόρριψη)ś

```
template<typename Radio, typename Debug>
class JoinDecision {
public:
  void init(Radio& , Debug& );
  void enable(void);
  void disable(void);

  void set_xx_parameter(xx_parameter_t);
  bool join(node_id , message_size , message);
  size_t get_payload_length(int);
  XX_Msg_t get_xx_payload(void);
};
```

Σχήμα 3.5: Διεπαφή JD

3.3.4 Iterator

Ο Iterator περιέχει κατάλληλες συναρτήσεις για να πάρει το Core Component την πληροφορία της τρέχουσας κατάστασης και να ένα τύπο μηνύματος που ενημερώνει την ομάδα και συγκεκριμένα τον για την τελική μορφή και δομή της ομάδας που σχηματίστηκε.

```
template<typename Radio, typename Debug>
class Iterator {
public:
  void init(Radio& , Debug& );
  void enable(void);
  void disable(void);

  void set_xx_parameter(xx_parameter_t);
  size_t get_payload_length(int);
  XX_Msg_t get_xx_payload(void);
  clusted_id_t cluster(void);
  int hops(void);
  node_id_t parent();
  bool gateway();
};
```

Σχήμα 3.6: Διεπαφή IT

3.4 Υλοποιημένα Components

Τα υλοποιημένα μέχρι την στιγμή αυτή components παρουσιάζονται με μια σύντομη περιγραφή στους Πίνακες 3.1και3.2. Τα components αυτά υλοποιήθηκαν για την υλοποίηση των αλγορίθμων που παρουσιάστηκαν στην ενότητα 2.5 και στην συνέχεια θα περιγραφεί το πως συνδυάζονται για την συνολική υλοποίηση των αλγορίθμων.

3.4.1 Συνδυασμός Components για την δημιουργία αλγορίθμων

Για τον σχηματισμό αλγορίθμων αυτό που απαιτείται από τον προγραμματιστή είναι να σχεδιάσει τον CC του αλγορίθμου του χρησιμοποιώντας είτε αυτούσιο είτε με μικρές τροποποιήσεις τα ήδη διαθέσιμα components. Για παράδειγμα ο LCA αλγόριθμος υλοποιήθηκε φτιάχνοντας τα 3 components του Πίνακα 3.3. Στην συνέχεια για την υλοποίηση του Moca δεν χρειάστηκε να υλοποιηθεί το CHD component ενώ για το JD και τον IT χρειάστηκε να γίνουν κάποιες τροποποιήσεις για να υποστηρίζονται τα επικαλυπτόμενα clusters και προέκυψαν τα Moca JD και Overlapping IT που μπορούν πλέον να χρησιμοποιηθούν για αλγορίθμους που αναφέρονται σε επικαλυπτόμενα clusters. Επίσης για την υλοποίηση του αλγορίθμου Fronts χρησιμοποιήθηκε το Attr CHD το οποίο είχε υλοποιηθεί αρχικά αλλά δεν είχε χρησιμοποιηθεί σε κάποιον αλγόριθμο. Το Bfs JD προέκυψε από τροποποιήσεις του LCA JD ενώ ο Iterator που χρησιμοποιείται είναι ο ίδιος. Για την υλοποίηση του LEAC η μόνη διαφοροποίηση σε σχέση με

Κατηγορία	Όνομα	Περιγραφή
Chd	Probabilistic (Prob)	Αποφασίζει σύμφωνα με ένα τυχαίο αριθμό και μια δοσμένη πιθανότητα p
	Leach	Αποφασίζει χρησιμοποιώντας ένα συνδυασμό πληροφορίας για την κατάσταση του κόμβου και την ενέργεια που του απομένει για να εξάγει την πιθανότητα με την οποία πρέπει να εκλεγεί p
	Attribute (Attr)	Ανταλλάσσει μηνύματα με μια τιμή και αποφασίζει με βάση την μικρότερη από αυτές
	Maxmind (Mmd)	Ανταλλάσσει μηνύματα με τα winner-ids των κόμβων και αποφασίζει με βάση την ευρετική συνάρτηση του Mmd
JD	Bfs	Διαμορφώνει Clusters με την αποστολή Broadcast μηνυμάτων και επιλέγει τον κοντινότερο ClusterHead
	Fronts	Διαμορφώνει Clusters με την αποστολή Broadcast μηνυμάτων και επιλέγει τον μικρότερο ClusterHead με βάση το μοναδικό id
	Maxmind (Mmd)	Αποφασίζει την συμμετοχή στην ομάδα που έχει ήδη επιλέξει Mmd CHD component
	Moca	Λειτουργεί όπως ακριβώς το Bfs JD αποδεχόμενο όλα τα clusters που απέχουν το πολύ k συνδέσεις

Πίνακας 3.1: Υλοποιημένα Components CHD, JD και τα βασικά τους χαρακτηριστικά

Κατηγορία	Όνομα	Περιγραφή
IT	Normal (Norm)	Παρέχει τις δομές για την συμμετοχή σε ένα cluster κάθε φορά
	Maxmind (Mmd)	Παρέχει τις δομές του Norm και κάποια επιπλέον στοιχεία για τον Mmmd αλγόριθμο
	Overlapping (Over)	Παρέχει τις δομές για την συμμετοχή σε ένα cluster πολλαπλά clusters
CC	Lca	Τα CC για τους υλοποιημένους αλγορίθμους
	Leach	
	Moca	
	Fronts	
	Maxmind (Mmd)	

Πίνακας 3.2: Υλοποιημένα Components IT, CC και τα βασικά τους χαρακτηριστικά

τον LCA είναι ο διαφορετικός τρόπος εκλογής αρχηγών και συνεπώς οι αλλαγές περιορίζονται μονάχα στο κομμάτι του CHD component.

Αλγόριθμος	CHD	JD	IT
Lca	Prob	Lca	Norm
Leach	Leach	Lca	Norm
Fronts	Attr	Bfs	Norm
Maxmind	Mmd	Mmd	Mmd
Moca	Prob	Moca	Over

Πίνακας 3.3: Σχηματισμός Αλγορίθμων από Components

Κεφάλαιο 4

Επιβεβαίωση Ορθότητας της Υλοποίησης

4.1 Προσομοιώσεις

Προσομοιωτής Shawn Για να εξεταστεί η λειτουργία των υλοποιήσεων σε μεγάλα δίκτυα, τάξης εκατοντάδων χιλιάδων κόμβων που δεν υπάρχουν άμεσα διαθέσιμα για χρήση απαιτείται η χρήση προσομοιωτών όπως του Shawn Sensor Network Simulator. Το Shawn [21] δίνει την δυνατότητα να δημιουργηθούν τοπολογίες δικτύων με ιδιαίτερα ελεγχόμενα χαρακτηριστικά και να προσδιοριστεί η συμπεριφορά του αλγορίθμου σε συγκεκριμένες καταστάσεις που δύσκολα μπορούν να επιτευχθούν σε επίπεδο πραγματικών συσκευών. Οι στόχοι των προσομοιώσεων είναι:

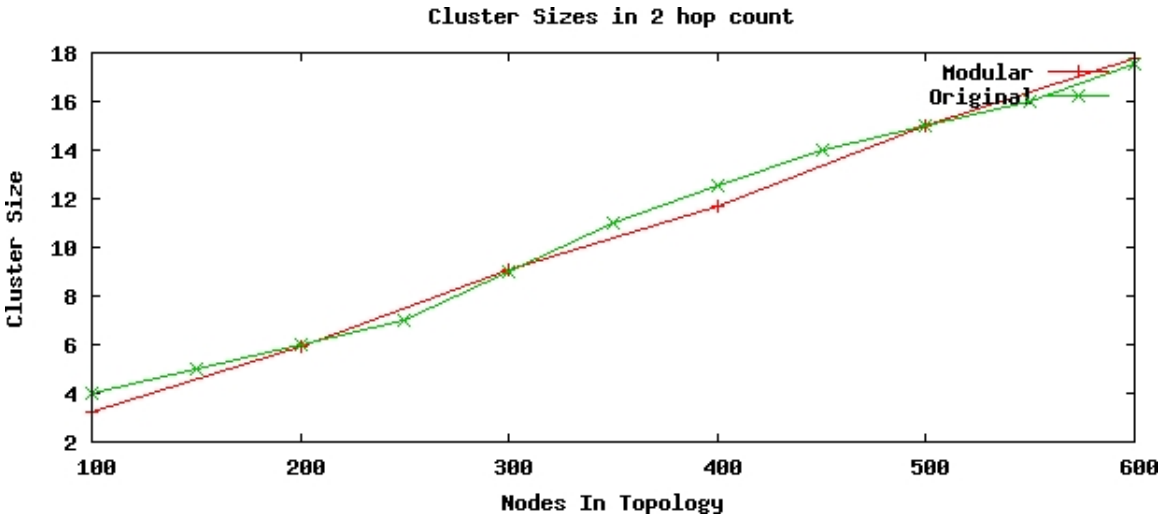
- Εκτέλεση πειραμάτων που αποδεικνύουν την ορθή υλοποίηση των αλγορίθμων με την modular αρχιτεκτονική
- Μελέτη της κλιμακοσιμότητας ως προς συγκεκριμένες παραμέτρους

4.1.1 Επιβεβαίωση ορθής υλοποίησης

Στη βιβλιογραφία κάποιων από τους αλγορίθμους που υλοποιήθηκαν υπήρχαν αποτελέσματα προσομοιώσεων από την χρήση των αλγορίθμων αυτών που παρουσίαζαν συγκεκριμένα χαρακτηριστικά για κάθε αλγόριθμο. Οι τοπολογίες πάνω στις οποίες είχαν εκτελεστεί τα πειράματα περιγράφονταν και έτσι για να αποδείξουμε ότι η υλοποίηση αυτών των αλγορίθμων είναι ορθή και σύμφωνη με όσα περιέγραφαν οι

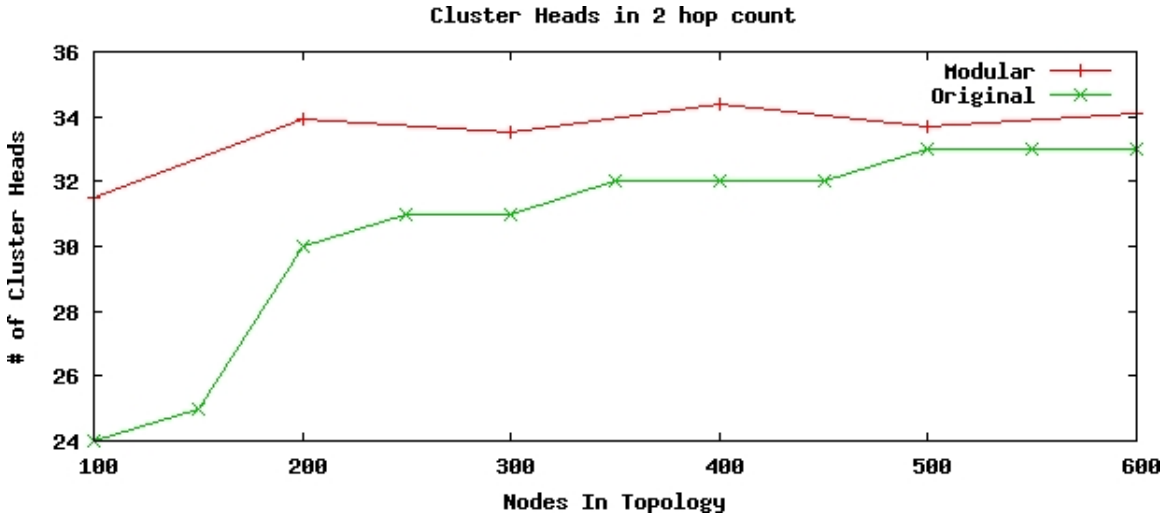
δημιουργοί τους αρκεί να επανεκτελέσουμε τα πειράματα και να συγκρίνουμε τα αποτελέσματα που παρατηρούμε.

Maxmind Για τον αλγόριθμο αυτό υπήρχαν αποτελέσματα για το μέγεθος των ομάδων που σχηματίζει και τον αριθμό των αρχηγών που εκλέγει σε συγκεκριμένες τοπολογίες. Συγκεκριμένα οι τοπολογίες που χρησιμοποιήθηκαν είναι τοπολογίες με το πολύ 600 κόμβους μέσα σε επιφάνεια μεγέθους 200x200 μονάδες και ακτίνα μετάδοσης 20 μονάδων. Η παράμετρος του αλγορίθμου d είχε την τιμή 2.



Σχήμα 4.1: Μέσο Μέγεθος ομάδων που προκύπτουν από την υλοποίηση και την βιβλιογραφία

Οπως φαίνεται από το Σχήμα 4.1 η υλοποίηση του αλγορίθμου δημιουργεί ομάδες με σχεδόν το ίδιο μέσο μέγεθος σε σχέση με τον αρχικό αλγόριθμο. Επίσης στο Σχήμα 4.2 όσον αφορά τον αριθμό των κόμβων που επιλέγονται ως αρχηγοί για μικρές τοπολογίες υπάρχει μια διαφορα της τάξης των 7 κόμβων η οποία φαίνεται να περιορίζεται σημαντικά καθώς το μέγεθος της τοπολογίας αυξάνει. Η διαφορά αυτή πιθανότατα δεν οφείλεται στην υλοποίηση του αλγορίθμου αλλά σε κάποια ιδιαιτερότητα των τοπολογιών που χρησιμοποιήθηκαν για την εκτέλεση των πειραμάτων είτε στην αρχική υλοποίηση είτε στην modular.

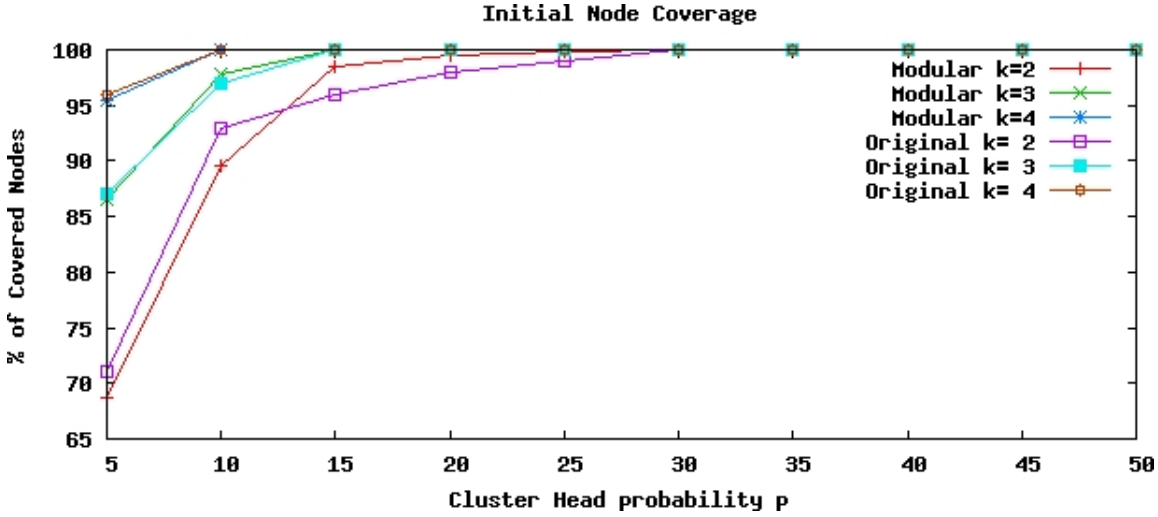


Σχήμα 4.2: Αριθμός ομάδων που προκύπτουν από την υλοποίηση και την βιβλιογραφία

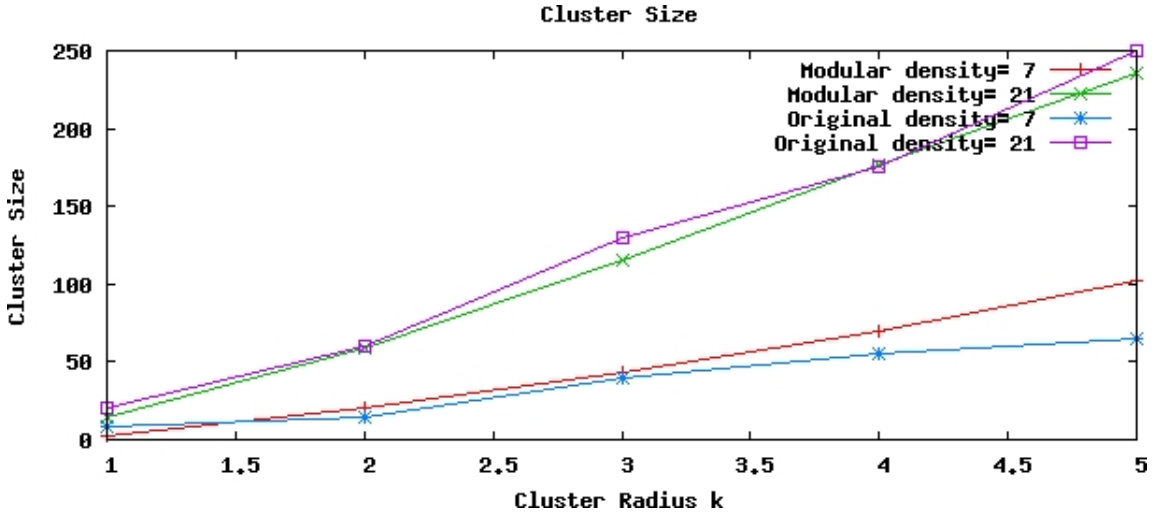
MOCA Για τον αλγόριθμο αυτό υπήρχαν αποτελέσματα για τα ιδιαίτερα χαρακτηριστικά των επικαλυπτόμενων ομάδων, όπως το ποσοστό κάλυψης όλων των κόμβων μέσα στην τοπολογίας στην πρώτη φάση εκτέλεσης του αλγορίθμου και το μέσο μέγεθος των ομάδων που δημιουργούνται σε σχέση με τον βαθμό των κόμβων. Συγκεκριμένα χρησιμοποιήθηκαν τοπολογίες με 400 κόμβους με μέσο αριθμό γειτόνων 7 και 21.

Στο Σχήμα 4.3 καταγράφεται το κατά πόσον η εκτέλεση της πρώτης φάσης του αλγορίθμου είναι ικανή να καλύψει κάθε κόμβο του δικτύου. Παρατηρούμε όπως και στις προηγούμενες περιπτώσεις ότι η υλοποίηση του αλγορίθμου πετυχαίνει σχεδόν πανομοιότυπα αποτελέσματα σε σχέση με τα στοιχεία που παρουσιάστηκαν. Επίσης παρατηρούμε ότι ακόμη και για χαμηλές τιμές στην παράμετρο p το ποσοστό των κόμβων που έχουν ήδη ενταχθεί σε κάποια ομάδα αγγίζει το 95%. Αντίστοιχη ομοιότητα παρατηρείται και στο μέσο μέγεθος ομάδας όπου διατηρώντας σταθερή την παράμετρο p μεταβάλουμε το μέγιστο ύψος της ομάδας (Σχήμα 4.4). Όπως ήταν αναμενόμενο και σύμφωνα και με προηγούμενα στοιχεία (καθώς επαναχρησιμοποιείται το Bfs JD) το μέσο μέγεθος ομάδας αυξάνεται αναλογικά με την παράμετρο k.

Για τους υπόλοιπους αλγορίθμους είτε δεν υπήρχαν είτε δεν ήταν δυνατό να επαναληφθούν πειράματα για την επιβεβαίωση της αρτιότητας της υλοποίησης.



Σχήμα 4.3: Ποσοστό κόμβων που μετέχουν σε κάποια ομάδα μετά από το βασικό στάδιο εκτέλεσης του αλγορίθμου



Σχήμα 4.4: Μέσο μέγεθος ομάδων που δημιουργούνται από τον αλγόριθμο Moca

Κεφάλαιο 5

Πειραματικά Αποτελέσματα

5.1 Πειραματικά αποτελέσματα με χρήση προσομοιωτών

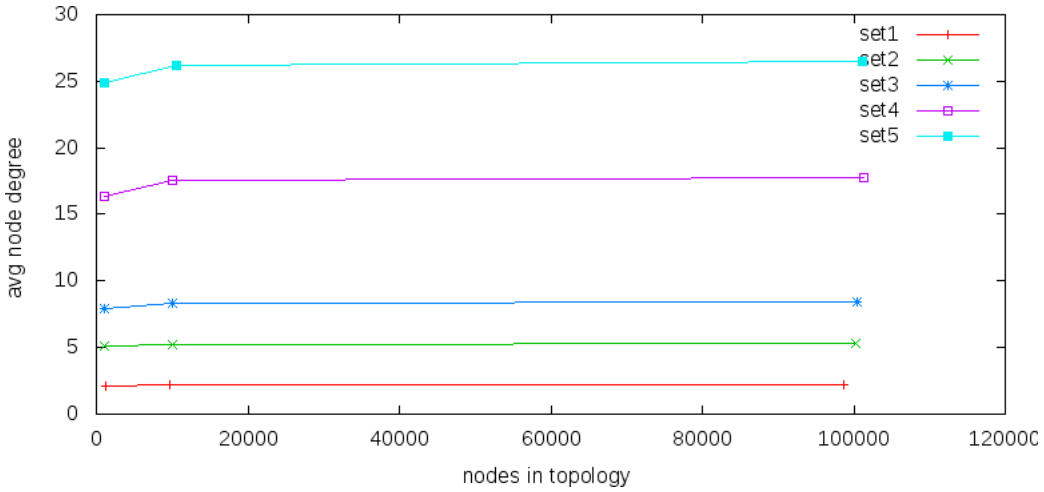
5.1.1 Κλιμακοσιμότητα

Στις προσομοιώσεις αυτές ψάχνουμε να βρούμε το πως επηρεάζεται η λειτουργία του αλγορίθμου καθώς αυξάνεται ο αριθμός των κόμβων που υπάρχουν στο δίκτυο. Υπάρχουν 2 τρόποι με τους οποίους μπορεί να αυξηθεί το μέγεθος του δικτύου:

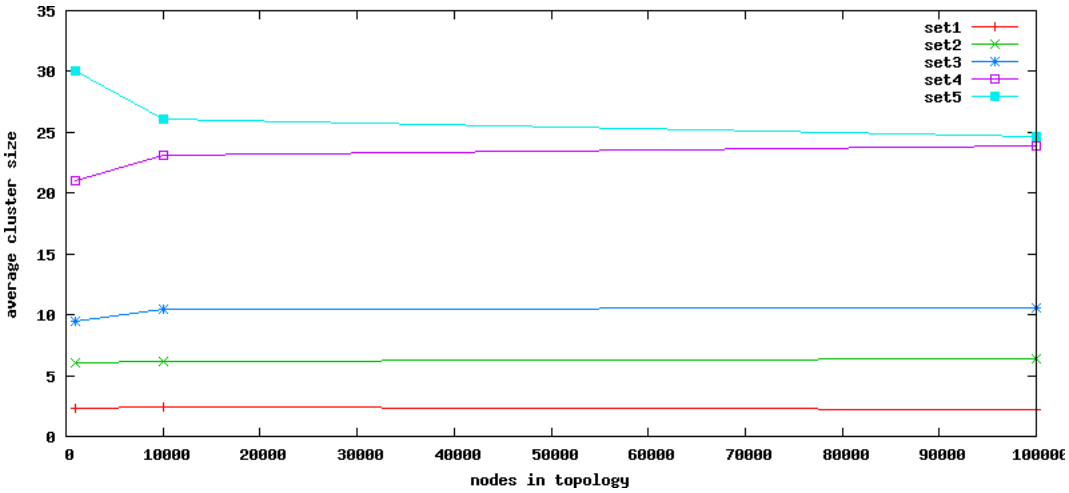
- τοποθετώντας μέσα στην ίδια περιοχή περισσότερους κόμβους αυξάνοντας την πυκνότητα του δικτύου
- είτε τοποθετώντας νέους κόμβους εκτός της υπάρχουσας περιοχής διατηρώντας σταθερή την πυκνότητα και αυξάνοντας την διάμετρο του δικτύου.

Κατασκευή Τοπολογιών

Στις προσομοιώσεις χρησιμοποιούνται 5 σετ τοπολογιών με μεγέθη 10 έως 100.000 κόμβους. Σε κάθε σετ η πυκνότητα είναι σταθερή σε όλες τις τοπολογίες η πυκνότητα και μεταβάλλεται η διάμετρος του δικτύου που σχηματίζεται. Οι διαφορετικές τιμές που επιλέγονται για τον αριθμό γειτόνων κάθε κόμβου φαίνονται στο Σχήμα 5.1 και κυμαίνονται από 2.5 έως 26.



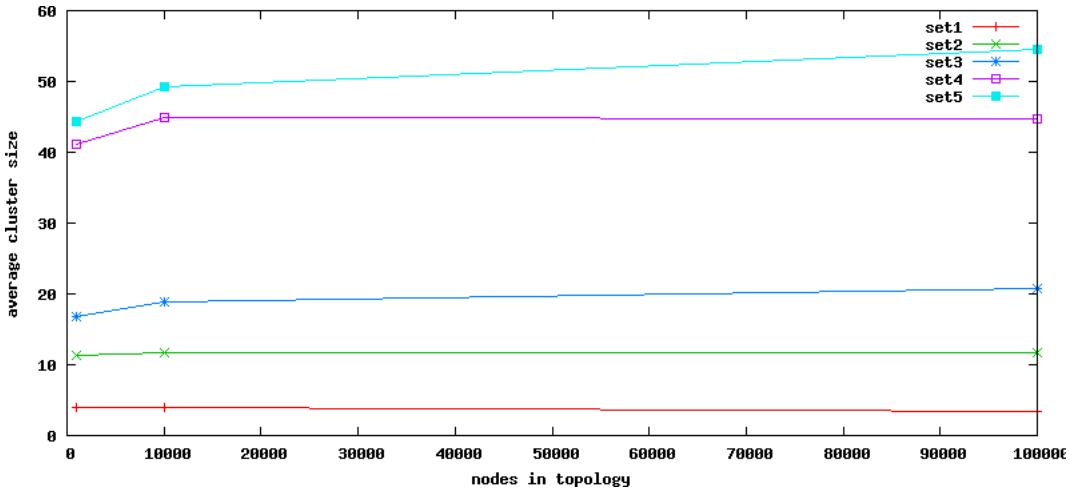
Σχήμα 5.1: Πυκνότητα Κόμβων στα 5 σετ Τοπολογιών



Σχήμα 5.2: Μέσο μέγεθος cluster για $k = 2$

Σταθερή Πυκνότητα - Αύξηση Διαμέτρου

Στόχος αυτών των πειραμάτων είναι να ελεγχθεί το κατά πόσο ο αλγόριθμος συμπεριφέρεται με τον ίδιο τρόπο ανεξάρτητα από την έκταση (διάμετρο) του δικτύου στο οποίο εκτελείται. Όπως φαίνεται και στα Σχήματα 5.2,5.3 το μέσο μέγεθος των clusters που σχηματίζονται είναι ανεξάρτητο του μεγέθους του δικτύου στο οποίο εκτελείται ο αλγόριθμος. Ωστόσο αυξάνοντας την παράμετρο k του αλγορίθμου

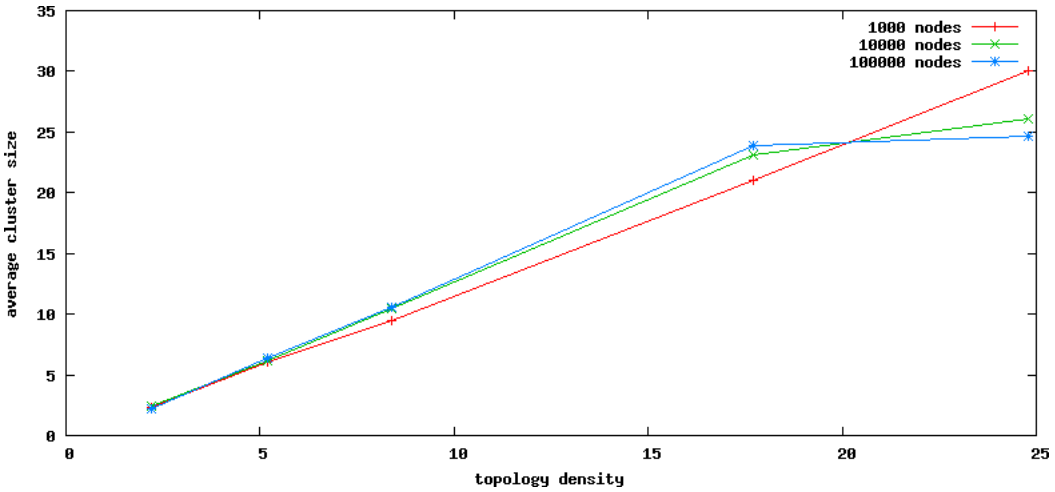


Σχήμα 5.3: Μέσο μέγεθος cluster για $k = 4$

οι περιοχές επιρροής των ClusterHeads επεκτείνονται καταλαμβάνοντας τους αντίστοιχους κόμβους. Έτσι μεταβάλλοντας το k μπορούμε να πετύχουμε clusters με διαφορετικές ιδιότητες και αποστάσεις των Gateway κόμβων από τους ClusterHeads που επιθυμούμε.

Σταθερός Αριθμός Κόμβων - Αύξηση Πυκνότητας

Εξετάζοντας την σχέση πυκνότητας - μεγέθους cluster παρατηρούμε στο Σχήμα 5.4 ότι η σχέση είναι γραμμική και όπως είδαμε πριν ανεξάρτητη του συνολικού μεγέθους του δικτύου.



Σχήμα 5.4: Σχέση πυκνότητας Δικτύου και Μεγέθους Cluster

5.2 Πειραματικά αποτελέσματα με χρήση πραγματικού υλικού

Τα πειράματα σε πραγματικές συσκευές έχουν ως στόχο τον έλεγχο της λειτουργίας του αλγορίθμου στον πραγματικό κόσμο. Σε αντίθεση με τις προσομοιώσεις υπάρχουν προβλήματα περισσότερο περίπλοκα και ανεξέλεγκτα, όπως παρεμβολές από άλλες συσκευές, πτώσεις τάσης, φυσικά εμπόδια που δυσκολεύουν την επικοινωνία των συσκευών και κατασκευαστικές ατέλειες που δίνουν διαφορετικά χαρακτηριστικά.

5.2.1 iSense Sensors

Σχήμα 5.5: iSense Sensor

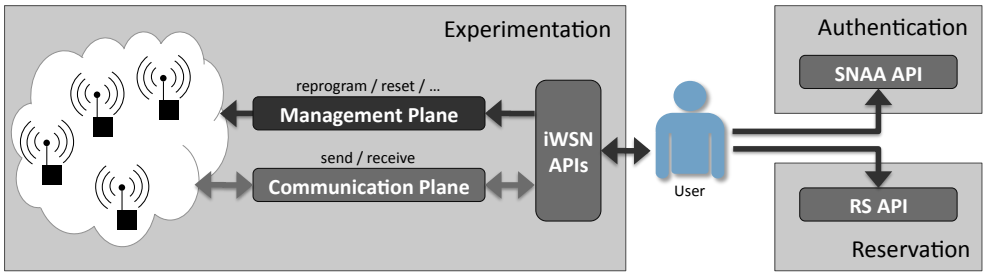
Για την πραγματοποίηση των πειραμάτων χρησιμοποιήθηκαν iSense sensors. Πρόκειται για sensors που διαθέτουν ένα 32-bit RISC επεξεργαστή, ασύρματη επικοινωνία τύπου IEEE 802.15.4 και 2 τύπων κεραί-

ες (ενσωματωμένες κεραμικές ή εξωτερικές). Για τον προγραμματισμό τους απαιτείται κώδικας σε C++. Τα iSense διαθέτουν επίσης μια μεγάλη συλλογή από πλακέτες αισθητήρων (modules) που συνδέονται πάνω στην πλακέτα του επεξεργαστή (core module) όπως αισθητήρες θερμοκρασίας, φωτεινότητας, σχετικής υγρασίας, βαρομετρικής πίεσης, επιτάχυνσης, υπέρυθρης ακτινοβολίας για τον εντοπισμό κίνησης, κάμερας για λήψη φωτογραφιών και δέκτη gps για προσδιορισμό της ακριβούς θέσης.

5.2.2 Wisebed

Το πρόγραμμα WISEBED είναι μια προσπάθεια 9 ακαδημαϊκών και ερευνητικών Ινστιτούτων από όλη την Ευρώπη. Το πρόγραμμα ξεκίνησε το Μάιο του 2008 και ολοκληρώθηκε το 2011. Στόχος του προγράμματος είναι να προσφέρει μια υποδομή διασυνδεδεμένων Testbed δικτύων ασύρματων αισθητήρων για ερευνητικούς σκοπούς. Αυτό δείχνει το πως πολλά μικρά δίκτυα διαφορετικών αισθητήρων (ετερογενών) μπορούν να συνδεθούν και να σχηματίσουν μεγάλες οργανωμένες δομές σε αντίθεση με την ανάγκη δημιουργίας ενός μεγάλου δικτύου. Επίσης δίνει στους ερευνητές την δυνατότητα να δουλέψουν με διαφορετικές καταστάσεις και δομές όπως και εγκαταστάσεις. Ακόμη ως μέρος του προγράμματος θα δημιουργηθούν ένα σύνολο αλγορίθμων για ετερογενή ασύρματα δίκτυα με χρήση της Wiselib οι οποίοι θα υλοποιηθούν και θα δοκιμαστούν στις εγκαταστάσεις που προσφέρονται από την υποδομή του Wisebed. Οι αλγόριθμοι που θα αναπτυχθούν θα είναι κυρίως στο πεδίο της αυτο-οργάνωσης και λειτουργίας του δικτύου. Συγκεκριμένα υπάρχουν αλγόριθμοι:

- Εύρεσης Γειτονιάς (Neighbor Discovery)
- Δρομολόγησης (Routing)
- Ομαδοποίησης (Clustering)
- Χρωματισμού (Coloring)
- Κρυπτογραφημένης Επικοινωνίας (Crypto)
- Εντοπισμού (Localization)
- Παρακολούθησης (Tracking)



Σχήμα 5.6

Πραγματοποίηση πειραμάτων στο WISEBED Για την πραγματοποίηση πειραμάτων σε οποιοδήποτε από τα 9 διαθέσιμα Testbeds χρειάζονται διαπιστευτήρια από κάποιο από τα Ιδρύματα που συμμετέχουν. Κάθε Testbed διατηρεί 3 διαφορετικούς εξυπηρετητές:

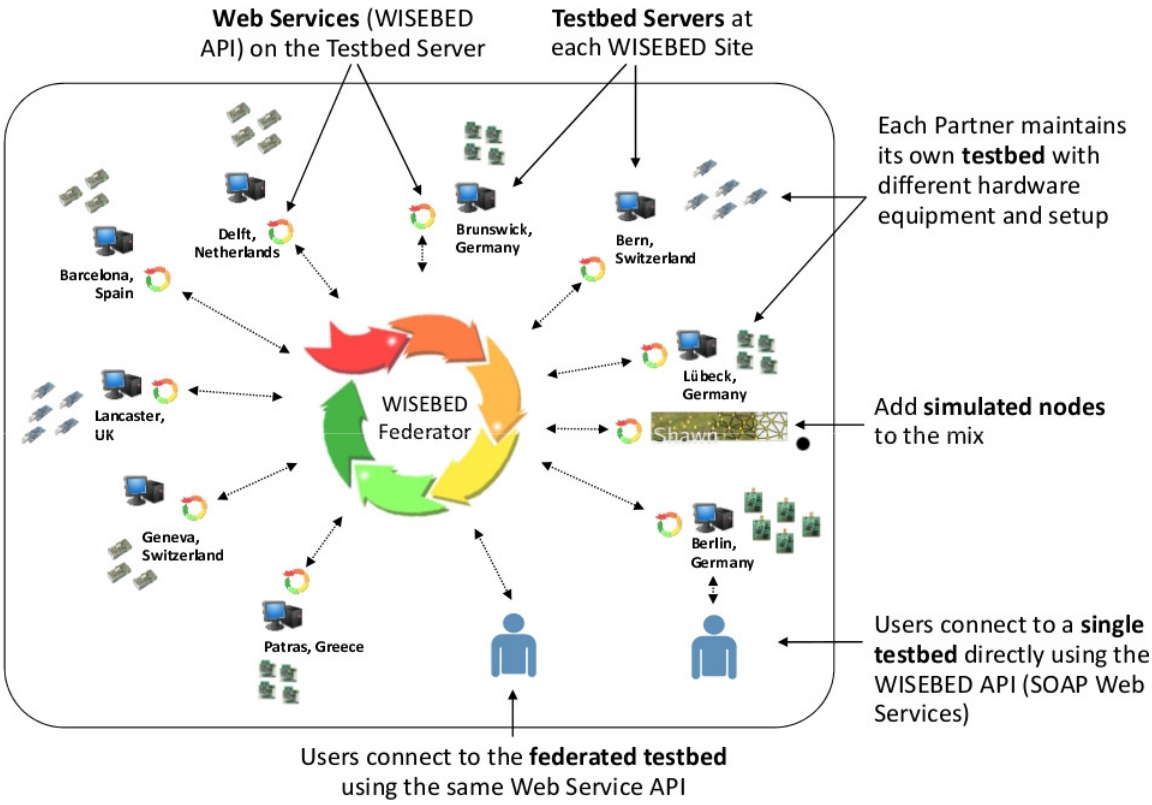
SNAA: για την πιστοποίηση των δικαιωμάτων των χρηστών

RS: Για την διαχείριση των κρατήσεων που κάνουν οι πιστοποιημένοι χρήστες έτσι ώστε μόνο ένας χρήστης να έχει κάθε φορά δικαίωμα σε κάθε κόμβο του συστήματος

IWSN: Για την πραγματική διαχείριση των αισθητήρων, τον προγραμματισμό, την συλλογή δεδομένων και ότι έχει να κάνει με την εκτέλεση του πειράματος του χρήστη.

Όλοι οι εξυπηρετητές ακολουθούν στην λειτουργία τους ένα προκαθορισμένο API το οποίο είναι κοινό για όλα τα Testbed του WISEBED, πράγμα που δίνει την δυνατότητα στον διαχειριστή κάθε Testbed να υλοποιήσει εσωτερικά τις διαδικασίες και τις λειτουργίες των εξυπηρετητών όπως επιθυμεί. Όπως φαίνεται και στην εικόνα του Σχήματος 5.6 ο χρήστης πρέπει ακολουθιακά να αποκτήσει πρόσβαση στο σύστημα μέσω του SNAA , να πραγματοποιήσει μια κράτηση για κάποιους ή όλους τους κόμβους ενός Testbed και στην συνέχεια να συνδεθεί στον IWSN εξυπηρετητή για να πραγματοποιήσει το πείραμά του.

Επίσης όπως αναφέραμε προηγουμένως στα πλαίσια του Wisebed είναι δυνατόν να υπάρξουν και πειράματα συνδυάζοντας περισσότερα του ενός Testbed και συνδέοντας τους κόμβους τους χρησιμοποιώντας εικονικές συνδέσεις, στις οποίες όλα τα μηνύματα που ανταλλάσσονται μέσω της ασύρματης επικοινωνίας μεταδίδονται και μέσω των IWSN εξυπηρετητών και σε άλλες συσκευές που βρίσκονται σε διαφορετικό χώρος και ο κόμβος τις αντιλαμβάνεται ως πραγματική ασύρματη επικοινωνία από κάποιο γειτονικό του κόμβο. Η δυνατότητα αυτή



Σχήμα 5.7

παρέχεται από τους Federator εξυπηρετητές. Αυτοί δίνουν την δυνατότητα να συνδέεται κάποιος χρήστης ταυτόχρονα σε περισσότερα από 1 Testbeds εκτελώντας τις ενέργειες που απαιτούνται μονάχα στον Federator ο οποίος στην συνέχεια αναλαμβάνει να τις προωθήσει προς τον κάθε κόμβο του συστήματος. Ο Federator προσφέρει αντίστοιχα τα Testbeds 3 διαφορετικούς εξυπηρετητές (SNAA, RS, IWSN) και περιγράφεται στην εικόνα του Σχήματος 5.7. Η ύπαρξη του κοινού API κάνει τον Federator να λειτουργεί με τον ίδιο τρόπο (ως ένα ενιαίο Testbed) αγνοώντας τις πραγματικές υλοποιήσεις των επιμέρους εξυπηρετητών.

5.2.3 TestbedRuntime

Για την πραγματοποίηση των πειραμάτων σε πραγματικές συσκευές απαιτείται συγχρονισμός όλων των συσκευών (τουλάχιστον στην εκκίνηση του αλγορίθμου), συλλογή αποτελεσμάτων και λήψη πληροφοριών

κατά την διάρκεια της εκτέλεσης του πειράματος για την εξασφάλιση της σωστής εκτέλεσής του. Για τον σκοπό αυτό χρησιμοποιήθηκε το λογισμικό TestbedRuntime (TR) που προσφέρει την δυνατότητα έλεγχου πολλών αισθητήρων. Το TR υποστηρίζει αισθητήρες τύπου iSense, Pacemate, TelosB και SunSpot. Προσφέρει πλήρη έλεγχο στις συσκευές και διευκολύνει σε μεγάλο βαθμό την εκτέλεση των πειραμάτων. Με την χρήση του λαμβάνεται έξοδος από όλες τις συσκευές σε πραγματικό χρόνο, στέλνονται εντολές (όπως μηνύματα εκκίνησης, αλλαγής κατάστασης και λήψης αναφοράς) και προσομοιώνονται ευκολότερα βλάβες και άλλες ειδικές καταστάσεις. Για την λειτουργία του TR οι συσκευές είναι συνδεδεμένες σε υπολογιστές που ελέγχονται από ένα κεντρικό Portal Server προσβάσιμο από τους χρήστες του συστήματος για την πραγματοποίηση των πειραμάτων. Προσφέρει ένα σύστημα ταυτοποίησης χρηστών ανάλογα με της ανάγκες του διαχειριστή καθώς και ένα σύστημα πραγματοποίησης κρατήσεων για οποιοδήποτε αριθμό κόμβων του συστήματος. Ο χρήστης στην συνέχεια έχοντας το δικαίωμα να διαχειριστεί τους κόμβους του συστήματος μπορεί να περάσει στους αισθητήρες τον αλγόριθμο που θέλει να δοκιμάσει και να πραγματοποιήσει το πείραμά του. Το σύστημα προσφέρει επίσης δυνατότητες όπως εικονικές συνδέσεις μέσω federated Testbeds όπου μέσω λογισμικού μπορούν να συνδεθούν 2 διαφορετικές τοποθεσίες και να επικοινωνούν με προκαθορισμένα μονοπάτια σχηματίζοντας έτσι ακόμη μεγαλύτερα Testbeds για δοκιμές.
Τα πειράματα σε πραγματικές συσκευές εκτελέστηκαν σε 3 διαφορετικές εγκαταστάσεις με διαφορετικά χαρακτηριστικά που παρουσιάζονται στην συνέχεια.

Τρόπος εκτέλεσης πειραμάτων χρησιμοποιώντας το TestbedRuntime
Για να εκτελέσουμε κάποιο πείραμα με την χρήση του TestbedRuntime χρησιμοποιούμε κάποια Beanshell scripts που εκτελούν την επικοινωνία ανάμεσα στον εξυπηρετητή του Testbed και στον τοπικό πελάτη τον οποίο πρέπει να εκτελέσουμε. Στα scripts αυτά πρέπει αρχικά να συνδεθούμε στο Testbed δίνοντας τα διαπιστευτήρια μας (όνομα χρήστη και κωδικός). Μετά την σύνδεση λαμβάνουμε κάποια κλειδιά τα οποία χρησιμοποιούμε στην συνέχει για να επιλέξουμε τους κόμβους που επιθυμούμε και να πραγματοποιήσουμε μια κράτηση για αυτούς. Με την επιτυχημένη κράτηση λαμβάνουμε ένα προσωπικό κωδικό για την συγκεκριμένη αυτήν κράτηση. Αυτός ο κωδικός είναι το μόνο που χρειαζόμαστε στην συνέχεια για να επικοινωνήσουμε με τον IWSN εξυπηρετητή για όλο το διάστημα που διαρκεί η κράτησή μας. Ενα παράδειγμα

τέτοιου πειράματος φαίνεται στο Σχήμα 5.8.

```
//Σύνδεση στον SNAA εξυπηρετητή
SNAA snaaService = SNAAServiceHelper.getSNAAService
  ("http ://testbed.wisebed.eu:8890/snaa");
AuthenticationTriple credentials = new AuthenticationTriple ();
//Διαπιστευτήρια χρήστη
credentials.setUrnPrefix ("urn:wisebed:cti:");
credentials.setUsername ("username@wisebed1.university.gr");
credentials.setPassword ("secretpassword");
List < AuthenticationTriple > credentialsList = new ArrayList ();
credentialsList.add( credentials );
//Αποστολή κωδικών
List < SecretAuthenticationKey > secretAuthenticationKeys =
   snaaService.authenticate ( credentialsList );

//Λήψη απότον IWSN τωνδιαθέσιμωνσυσκευώντύπου isense
SessionManagement smService =
WSNServiceHelper.getSessionManagementService
   ("http ://testbed.wisebed.eu:8888/sessions");
String wiseml = smService.getNetwork ();
List <String > nodeUrns = WiseMLHelper.getNodeUrns (wiseml , " isense ");

//Πραγματοποίηση Κράτησης
RS rsService = RSServiceHelper.getRSService
   ("http ://testbed.wisebed.eu:8889/rs");
XMLGregorianCalendar from = datatypeFactory.newXMLGregorianCalendar (
new GregorianCalendar (2011 , 04, 12, 15, 00));
XMLGregorianCalendar to = datatypeFactory.newXMLGregorianCalendar (
new GregorianCalendar (2011 , 04, 13, 16, 00));
ConfidentialReservationData reservationData = new ConfidentialReservationData ();
reservationData.setFrom (from);
reservationData.setTo(to);
reservationData.getNodeURNs().addAll(nodeUrns);
//Λήψη κωδικούγιατηνΚράτηση
List < SecretReservationKey > secretReservationKeys = rsService . makeReservation (
secretAuthenticationKeys ,reservationData);

//Σύνδεση στον IWSN εξυπηρετητή
WSN wsnService = WSNServiceHelper.getWSNService(wsnEndpointUrl);
//Επανεκίνηση τωνκόμβων
wsnService.resetNodes(nodeUrns);
Thread.sleep(3000) ;
//Αποστολή μηνύματοσεόλουςτουςκόμβους
Message startCommand = new Message ();
startCommand.setBinaryData(" start ".getBytes ());
wsnService.send(nodeUrns , startCommand );
```

Σχήμα 5.8: Παράδειγμα script για την εκτέλεση ενός πειράματος στο TestbedRuntime

Testbed UZL (Σχήμα 5.9)

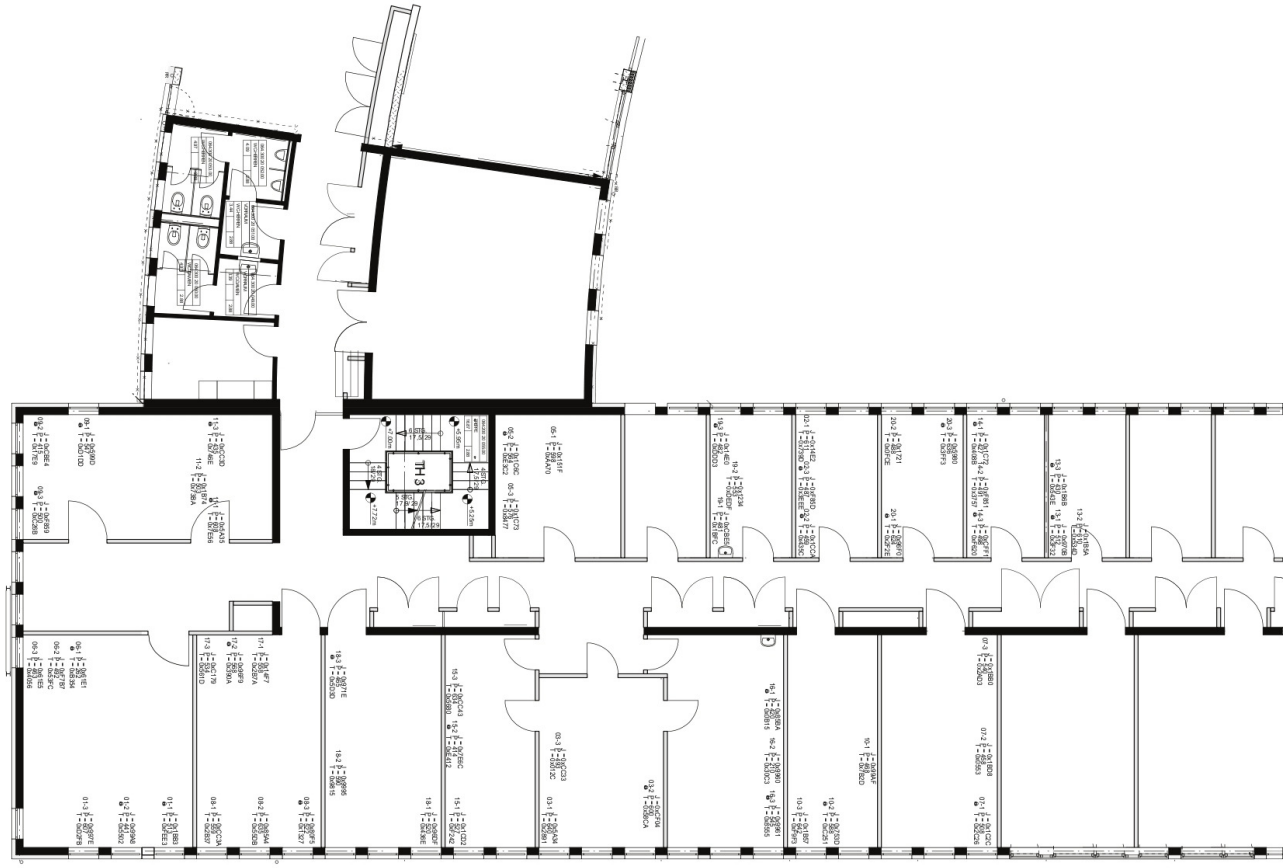
Βρίσκεται στις εγκαταστάσεις του Πανεπιστημίου του Luebeck στην Γερμανία, είναι το μεγαλύτερο από τα 3 διαθέσιμα testbeds και αποτελείται από 54 κόμβους iSense. Παρά το γεγονός ότι είναι το μεγαλύτερο σε έκταση οι κόμβοι του είναι τοποθετημένοι με τέτοιο τρόπο που η πυκνότητα και ο μέσος αριθμός γειτόνων ανά κόμβο είναι χαμηλότερη από τα υπόλοιπα.

Testbed EAITY (Σχήμα 5.10)

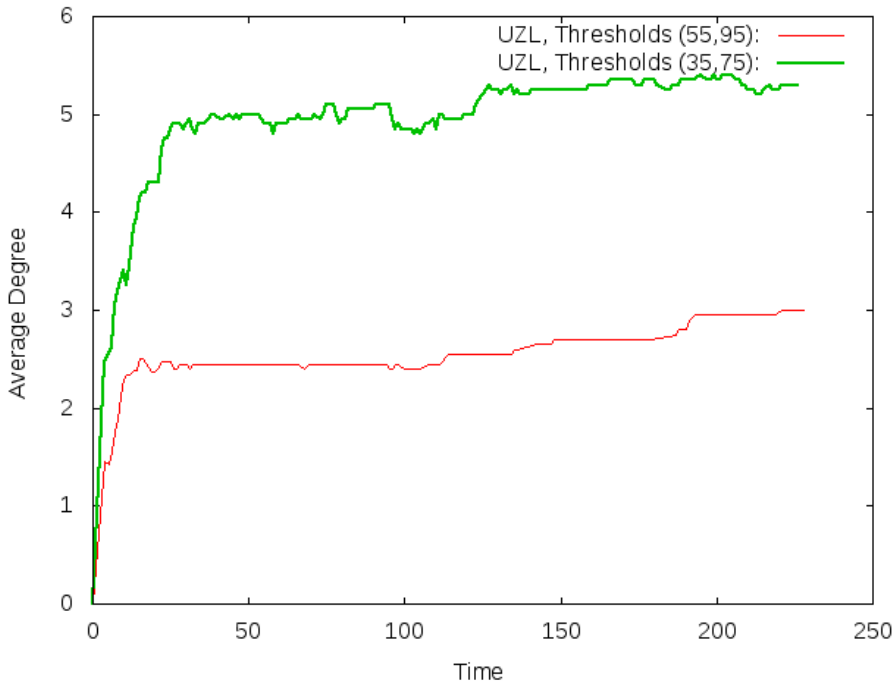
Αποτελείται από 18 αισθητήρες iSense τοποθετημένους σε 9 συνολικά γραφεία. Λόγω της μορφής του κτηρίου παρουσιάζει μια ιδιορρυθμία με τους αισθητήρες να είναι χωρισμένοι σε 2 ομάδες, από 8 και 10 αισθητήρες αντίστοιχα οι οποίες συνδέονται μέσω ενός διαδρόμου. Συγκριτικά είναι το μικρότερο σε μέγεθος αλλά παρουσιάζει πυκνότητα μεγαλύτερη του UZL και μικρότερη του UNIGE

Testbed UNIGE (Σχήμα 5.11)

Αποτελείται από 25 κόμβους συγκεντρωμένους σε μικρότερο χώρο από τα προηγούμενα. Με αυτό τον τρόπο η πυκνότητα του δικτύου είναι μεγαλύτερη. Επίσης οι κόμβοι είναι ομοιόμορφα τοποθετημένοι μέσα σε ένα τετράγωνο χώρο 5 γραφείων.

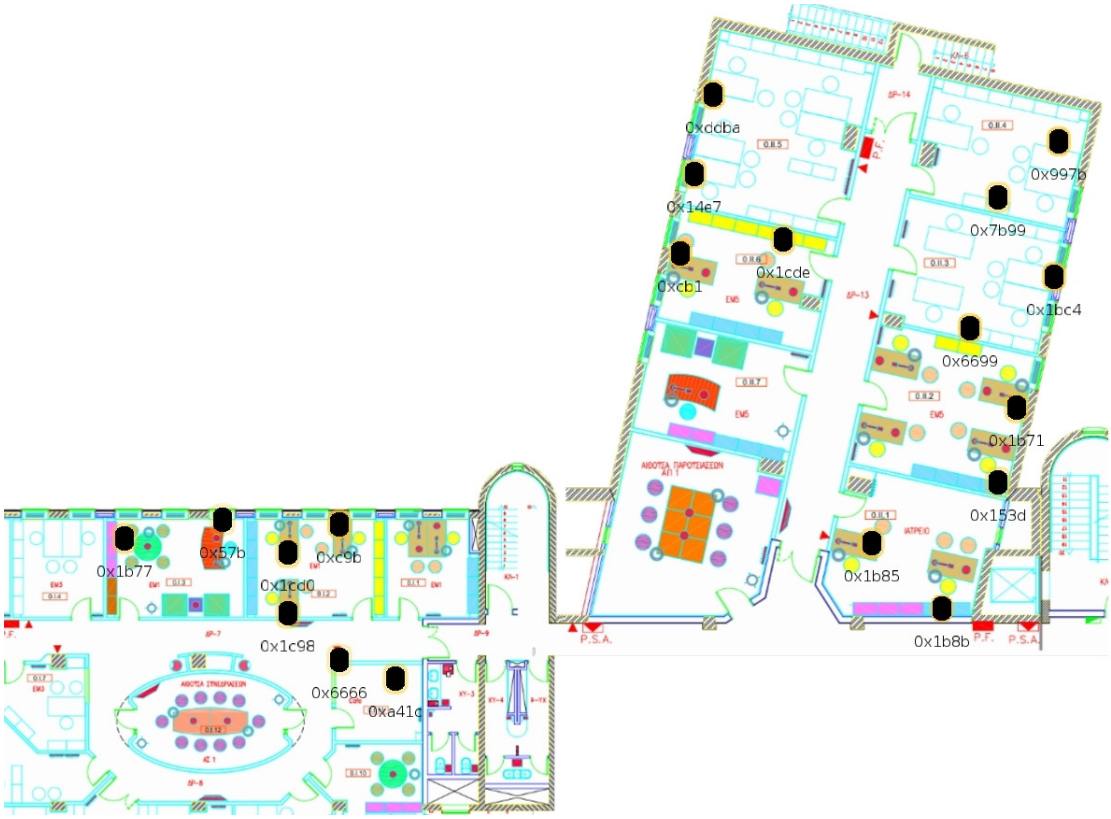


(α') Χάρτης Συσκευών

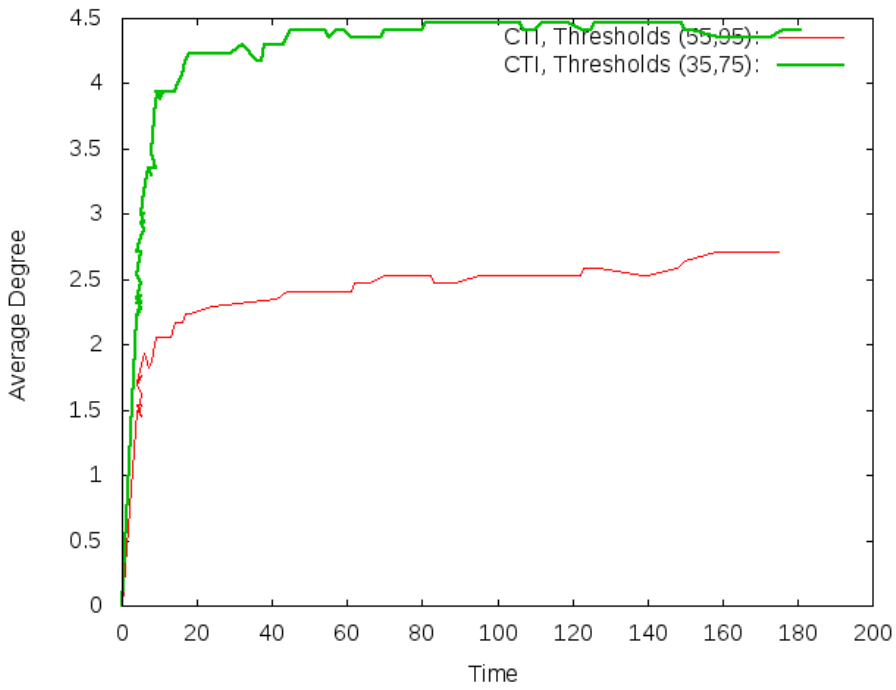


(β') Ποιότητα Συνδέσεων

Σχήμα 5.9: Testbed UZL

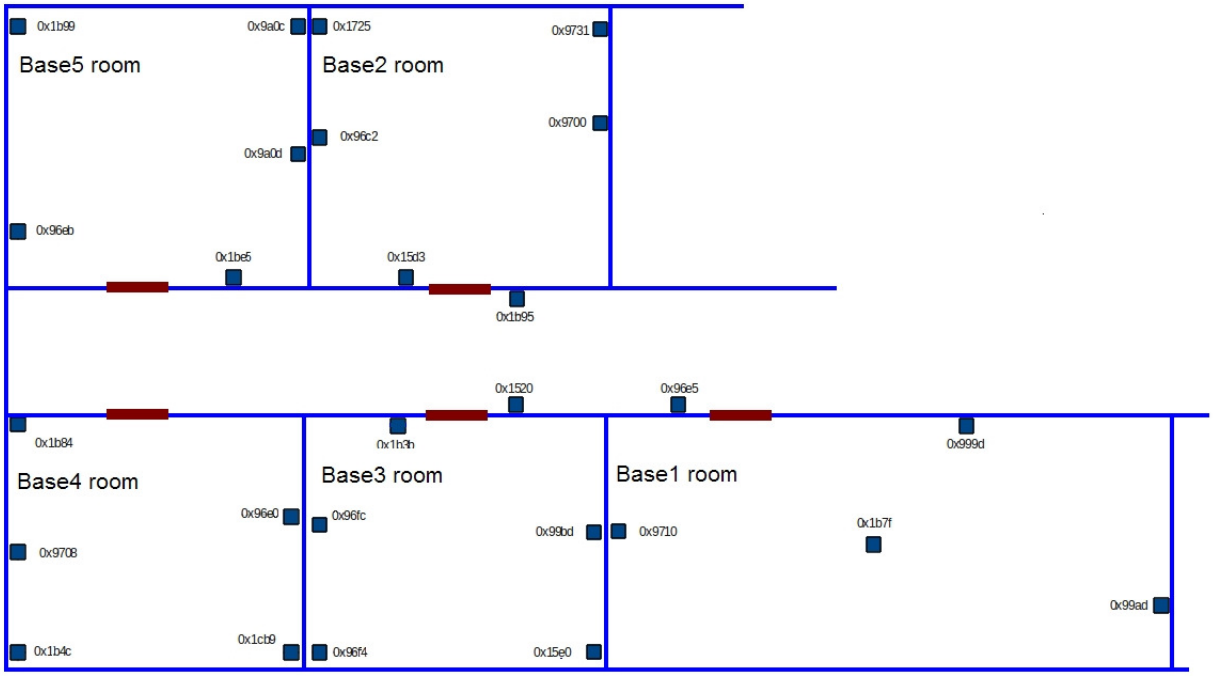


(α') Χάρτης Συσκευών

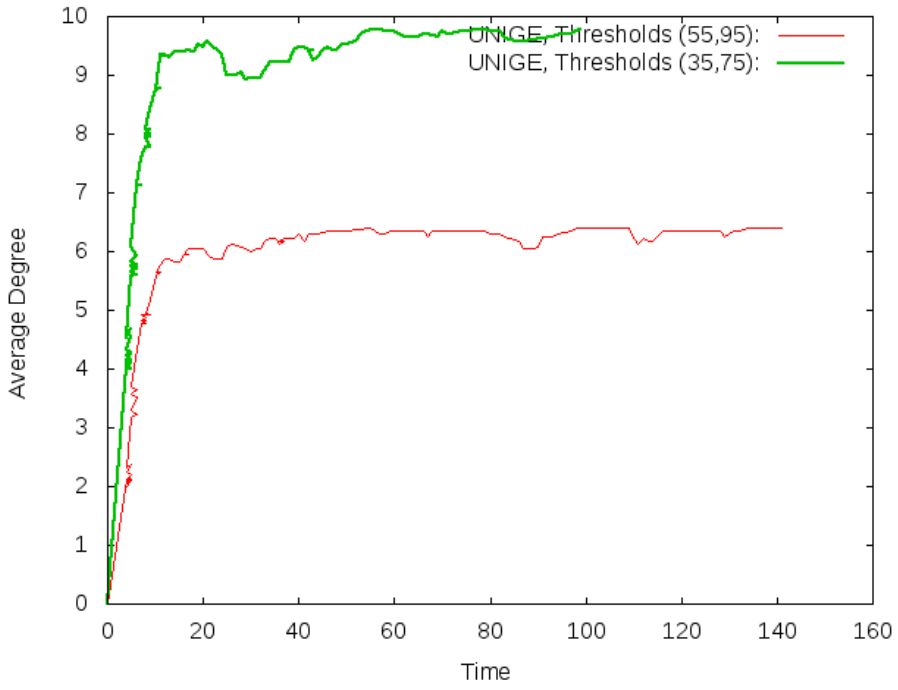


(β') Ποιότητα Συνδέσεων

Σχήμα 5.10: Testbed EAITY



(α′) Χάρτης Συσκευών



(β′) Ποιότητα Συνδέσεων

Σχήμα 5.11: Testbed UNIGE

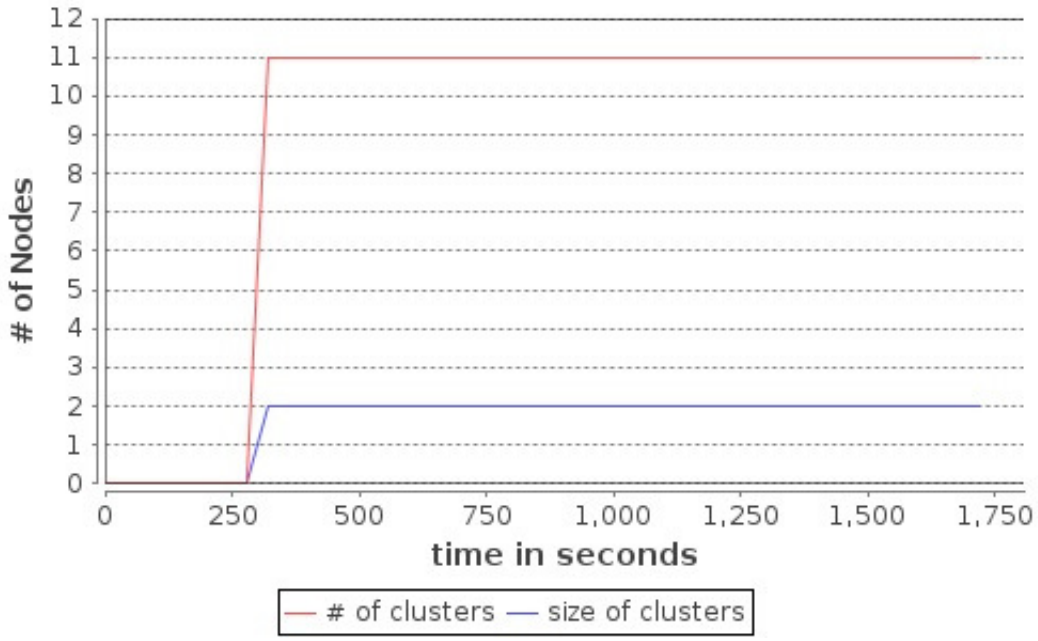
5.2.4 Σταθερά Δίκτυα

Στα πειράματα που παρουσιάζονται στις επόμενες σελίδες προσπαθήσαμε να μελετήσουμε την συμπεριφορά ενός από τους αλγορίθμους, επιλέχθηκε ο αλγόριθμος FRONTS, στα 3 διαθέσιμα δίκτυα τόσο σε κατάσταση ηρεμίας όσο και σε κατάσταση αστάθειας μεταβάλλοντας την βασική παράμετρο του αλγορίθμου, το μέγεθος του cluster.

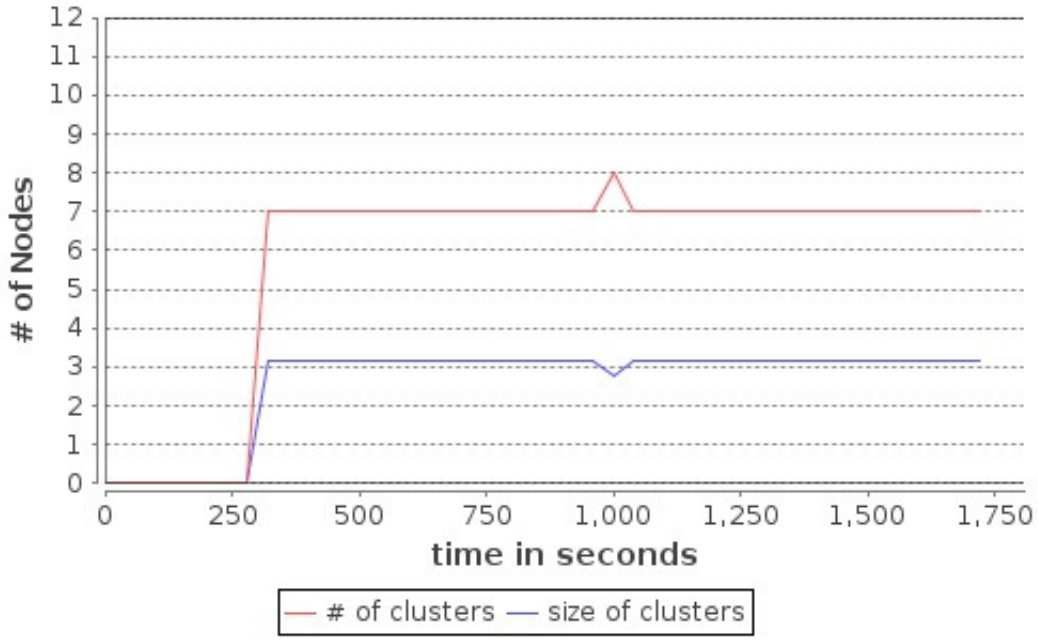
Αρχικά ξεκινάμε από το μεγαλύτερο διαθέσιμο δίκτυο (όπου είχαμε διαθέσιμους τους 22 από τους 54 κόμβους. Παρατηρούμε από το Σχήμα 5.12 ότι για επιλογή $k = 1$ το δίκτυο χωρίζεται σε 11 clusters με μέσο αριθμό μελών ανά cluster 2. Ο διαχωρισμός αυτός καθώς το δίκτυο παραμένει σταθερό συνεχίζεται κατά την διάρκεια όλου του πειράματος ενώ αντίστοιχα στοιχεία παρατηρούμε αν αυξήσουμε την παράμετρο k στην τιμή 2 όπου τα clusters από 11 πέφτουν σε 7, καθώς οι clusters με τα μικρότερα ids απορροφούν clusters με μεγαλύτερα και το μέγεθος αυξάνει κατά 1 περίπου για να φτάσουμε τους συνολικά 22 κόμβους που διαθέτουμε.

Το δίκτυο του EAITY είναι το επόμενο στο οποίο εκτελέστηκε ο ίδιος αλγόριθμος. Παρατηρούμε στο Σχήμα 5.13 ότι για αντίστοιχη εκτέλεση αρχικά με μέγεθος $k = 1$ και στην συνέχεια $k = 2$ ο αλγόριθμος παρουσιάζει τα ίδια χαρακτηριστικά. Για την πρώτη περίπτωση διαμορφώνονται clusters ίδιου μεγέθους αλλά λιγότερες στον αριθμό (όπως αναμένεται καθώς είναι λιγότεροι οι διαθέσιμοι κόμβοι). Ωστόσο στην δεύτερη εκτέλεση βλέπουμε ότι σχηματίζονται λιγότερα και μεγαλύτερα clusters. Όπως επιθυμούσαμε η πυκνότητα του δικτύου είναι καθοριστικός παράγοντας για την συμπεριφορά και την ομαδοποίηση που προκύπτει από τον αλγόριθμο.

Το τελευταίο από τα 3 δίκτυα είναι και το πιο πυκνό. Αυτό παρουσιάζεται χαρακτηριστικά στο Σχήμα 5.14 όπου για την εκτέλεση με $k = 1$ η συμπεριφορά είναι αντίστοιχη. Για την περίπτωση του $k = 2$ εμφανίζεται λόγω της εγκατάστασης τόσων κόμβων σε αρκετά μικρό χώρο ένα ιδιαίτερο αποτέλεσμα. Σχηματίζονται 2 μεγάλα clusters με 11 κόμβους το κάθε ένα. Το χαρακτηριστικό αυτό οφείλεται κυρίως στο ότι η διάμετρος αυτού του δικτύου είναι 4 συνδέσεις (λόγω του περιορισμένου χώρου) και επομένως αναμενόμενα 2 clusters αρκούν για να καλύψουν το σύνολο των κόμβων. Επίσης η διακύμανση που παρατηρείται ανάμεσα στα 7 και 8 clusters οφείλεται στην ύπαρξη μιας προβληματικής συσκευής όπως ανακαλύψαμε στην συνέχεια που βρισκόταν σε τέτοιο σημείο ώστε να χάνει συνεχώς τις συνδέσεις της με τις υπόλοιπες συσκευές και πρόσθετε αυτό το επιπλέον cluster στο αποτέλεσμα.

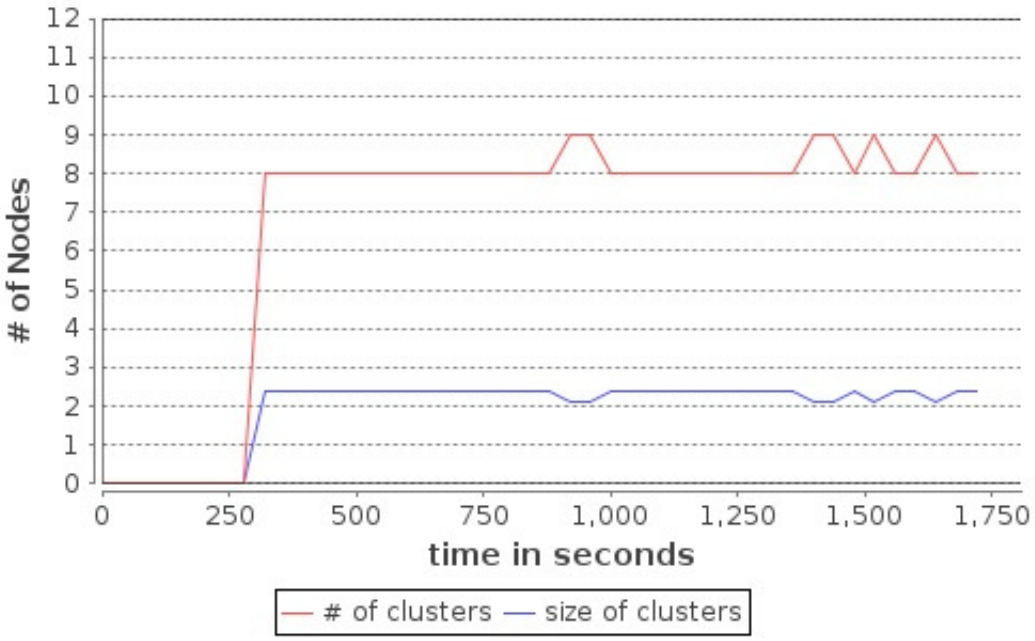


(α′) εκτέλεση 30 λεπτών για k=1

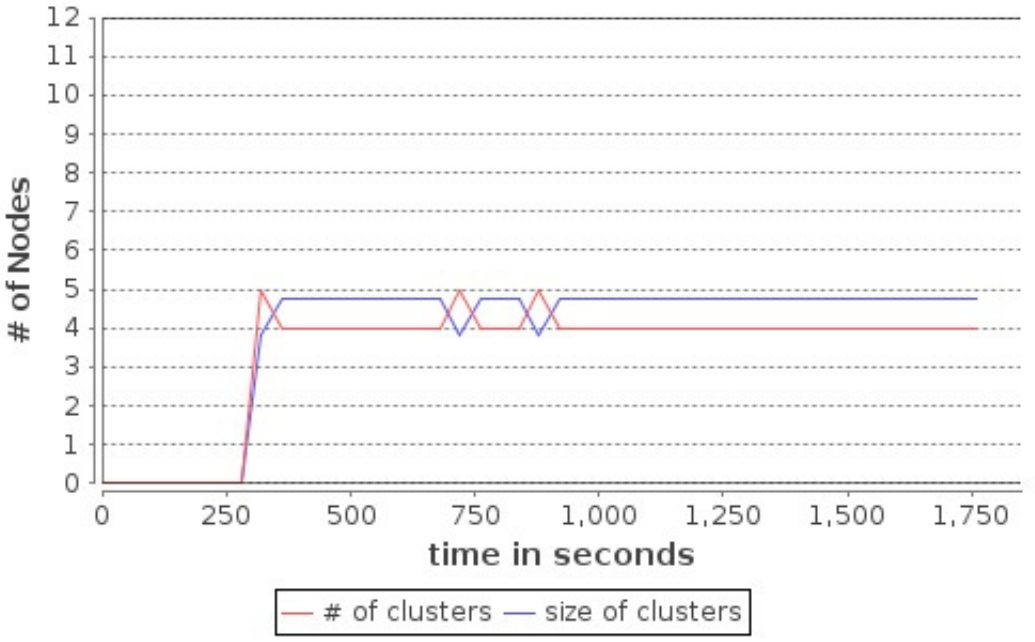


(β′) εκτέλεση 30 λεπτών για k=2

Σχήμα 5.12: Αριθμός Clusters σε Σταθερό Δίκτυο UZL

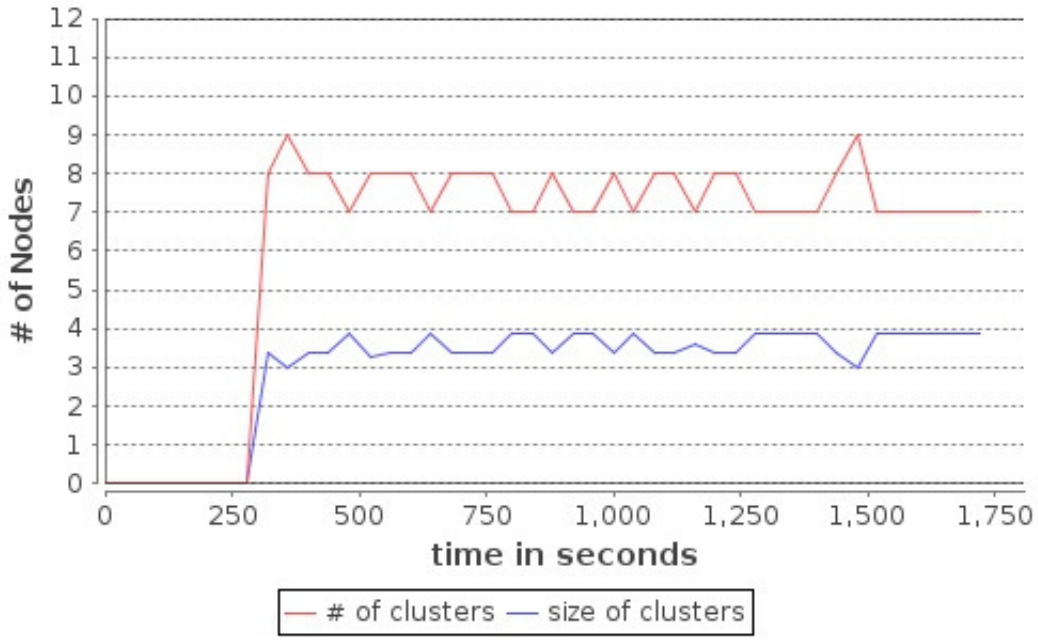


(α′) εκτέλεση 30 λεπτών για k=1

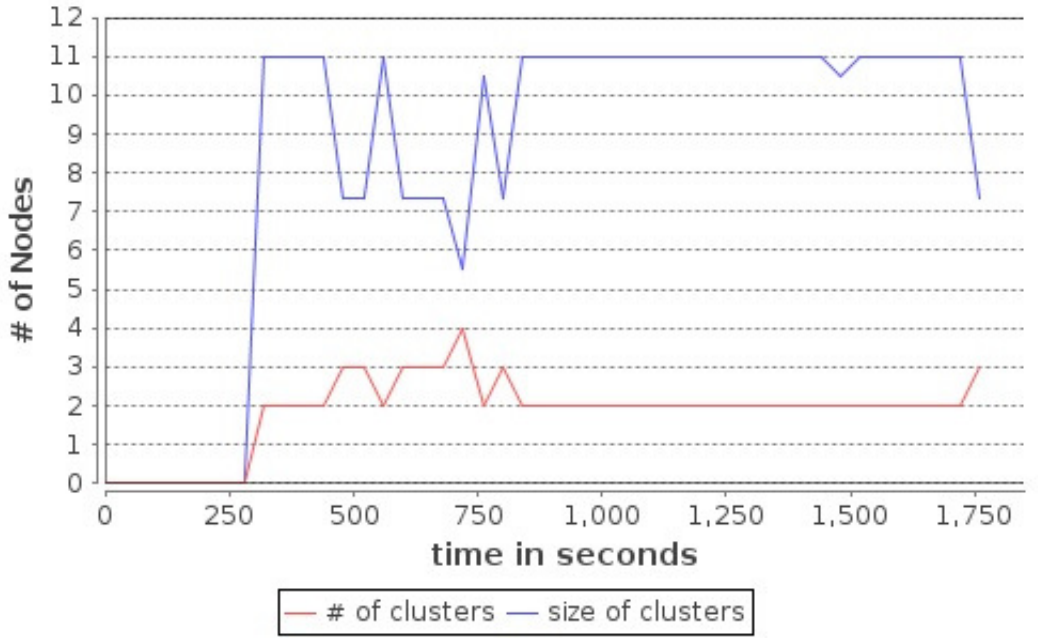


(β′) εκτέλεση 30 λεπτών για k=2

Σχήμα 5.13: Αριθμός Clusters σε Σταθερό Δίκτυο EAITY



(α′) k=1



(β′) k=2

Σχήμα 5.14: Αριθμός Ομάδων Σε Σταθερό δίκτυο UNIGE

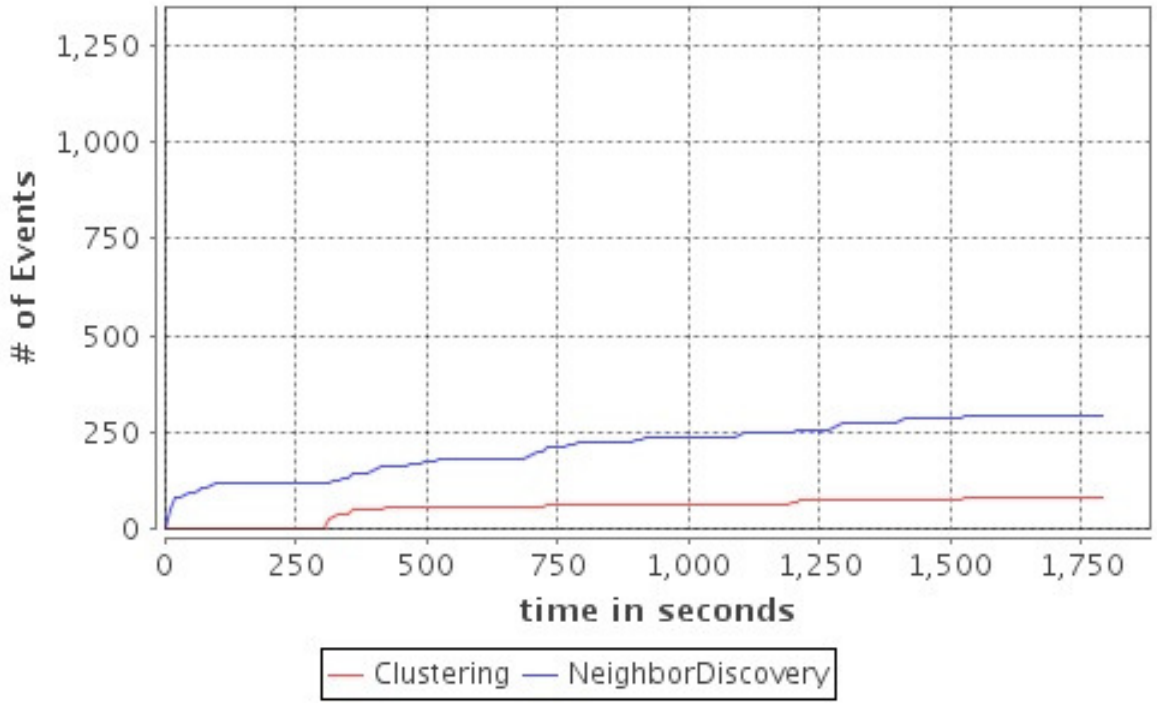
5.2.5 μη Σταθερά Δίκτυα

Στην συνέχεια εκτελούμε τα ίδια περάματα με πριν αλλά θα προκαλέσουμε αστάθεια στο δίκτυο. Για να το πετύχουμε αυτό τροποποιούμε τον αλγόριθμο Neighbor Discovery που χρησιμοποιούμε για να μεταδίδει περισσότερα μηνύματα από όσα απαιτούνται και συνεπώς να χάνει κάποια από αυτά και να μεταβάλλει την κατάσταση των συνδέσεων εσφαλμένα σε σχέση με πριν. Αυτή η αστάθεια θα προκαλέσει περισσότερες μεταβολές στην γειτονιά των κόμβων άλλα ο αλγόριθμος θα τείνει να διατηρήσει της ομάδες και να τις προσαρμόσει στην κατάσταση που υπάρχει κάθε φορά στο δίκτυο.

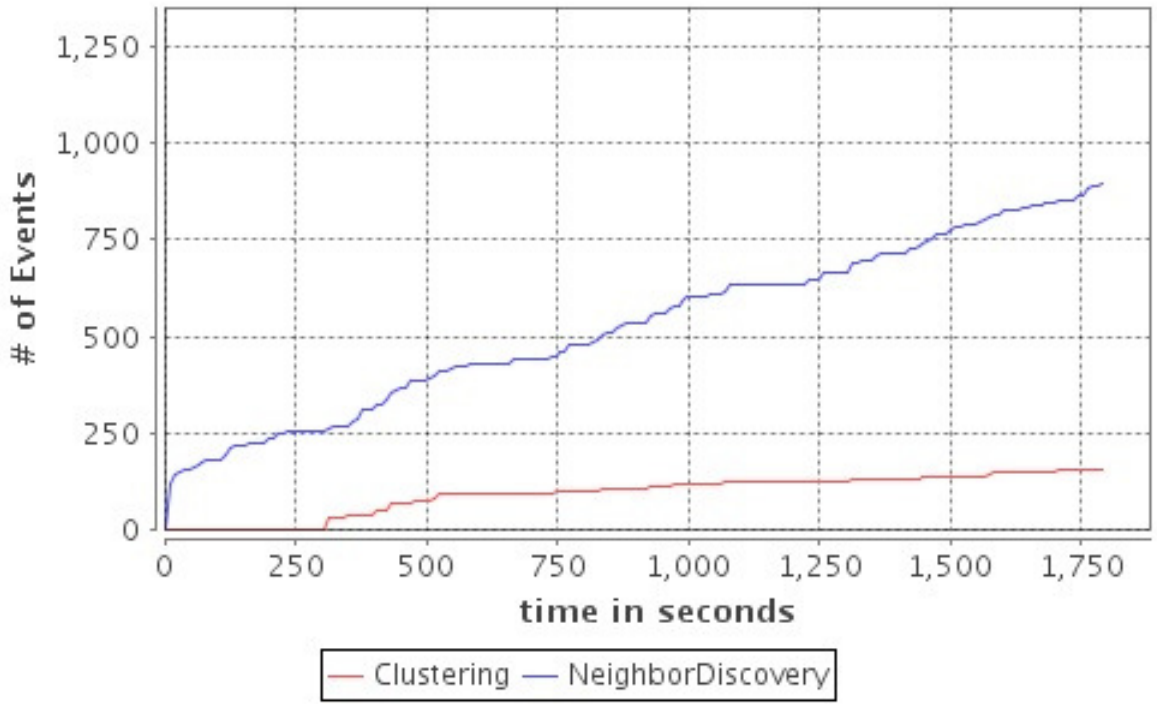
Στα Σχήματα 5.15,5.16 παρουσιάζονται με μπλε γραμμή οι μεταβολές που υπάρχουν στο δίκτυο και το χαρακτηρίζουν ως ευσταθές ή αστάθές.

Παρατηρούμε ότι για ενώ στο ευσταθές δίκτυο οι μεταβολές μέσα σε 30 λεπτά που διάρκεια το πείραμα είναι περίπου 250 στην δεύτερη περίπτωση τετραπλασιάζονται και αυτό το γεγονός αναγκάζει τον αλγόριθμο να προσαρμοστεί στις αλλαγές αυτές και να παράγει και αυτός με τη σειρά του περισσότερες μεταβολές (κόκκινη γραμμή). Στο Σχήμα 5.17 φαίνεται ο αντίκτυπος όλων αυτών των μεταβολών στην κίνηση του δικτύου και συγκεκριμένα στους 3 τύπους μηνυμάτων που ανταλλάσσουν οι κόμβοι για τον σχηματισμό και την προσαρμογή των ομάδων.

Στις γραφικές παραστάσεις του Σχήματος 5.18,5.19 βλέπουμε τις ομάδες που δημιουργούνται για τα 2 δίκτυα (UZL και EAITY) στην περίπτωση του μη Σταθερού δικτύου που παρουσιάσαμε πιο πάνω. Βλέπουμε ότι ο αλγόριθμος σχηματίζει αριθμό clusters και μέσο μέγεθος cluster αντίστοιχο με πριν αλλά υπάρχουν κάποιες αυξομοιώσεις που οφείλονται στην γενικότερη αστάθεια των συνδέσεων ανάμεσα στους κόμβους.

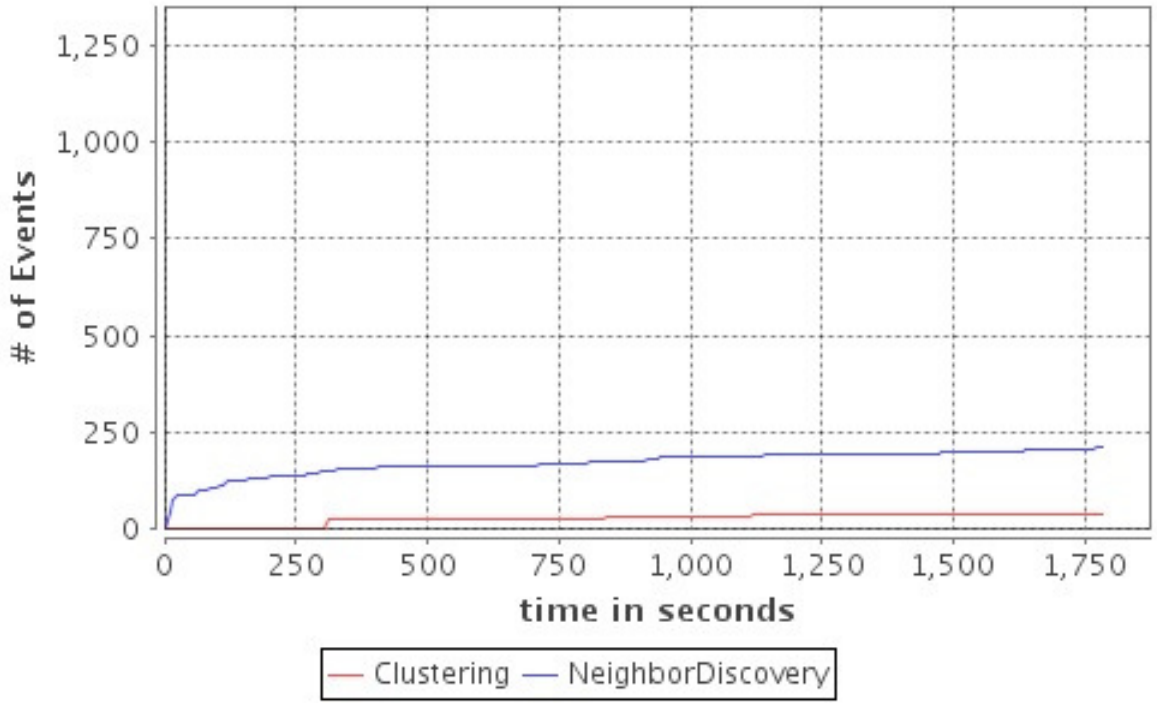


(α') Σταθερό Δίκτυο

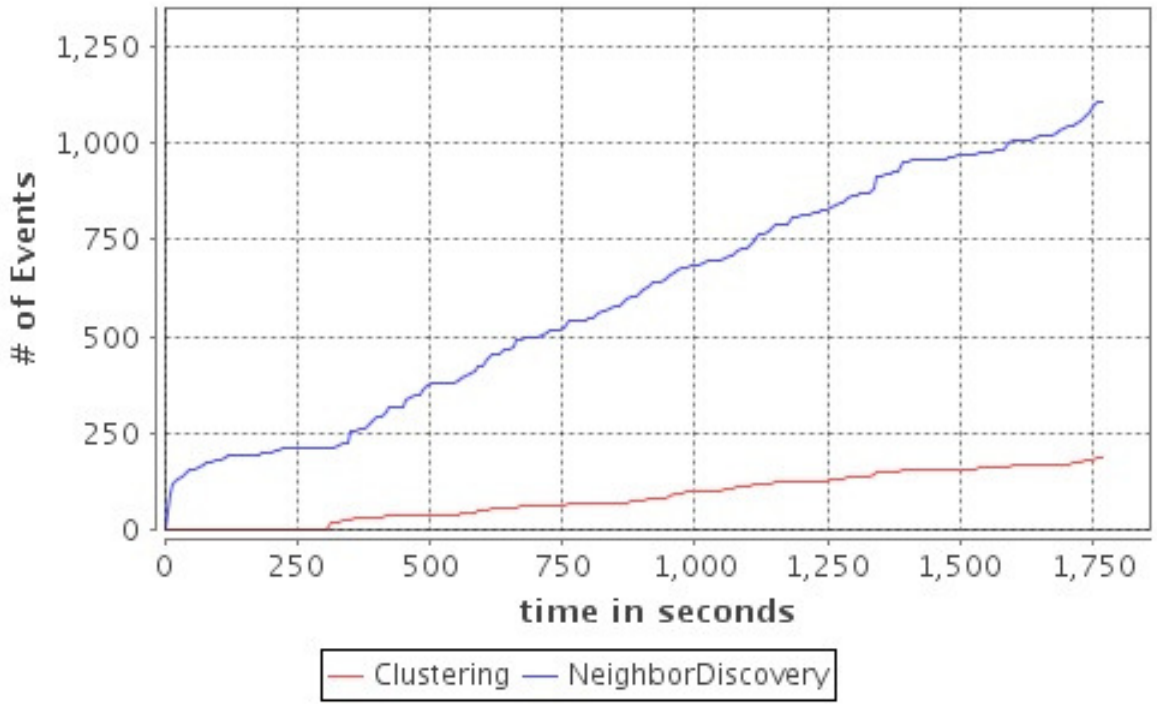


(β') Μη Σταθερό Δίκτυο

Σχήμα 5.15: Σύγκριση Σταθερού και μη δικτύου ως προς μεταβολές & ανταλλαγή μηνυμάτων

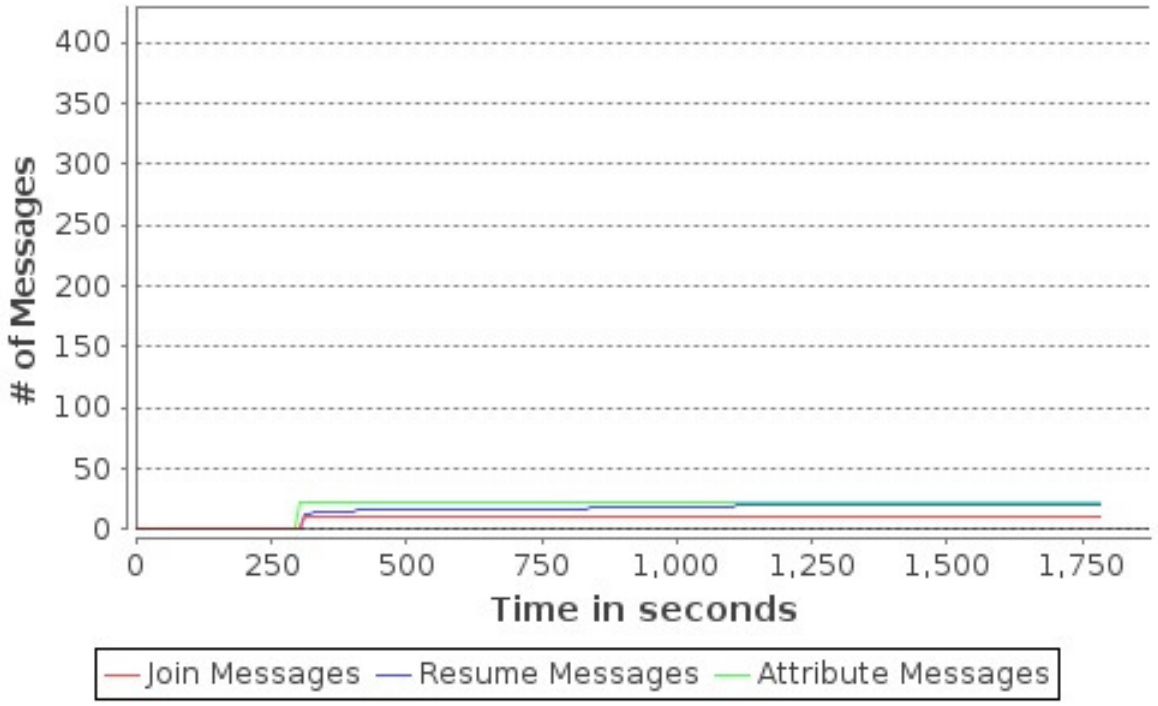


(α') Σταθερό Δίκτυο

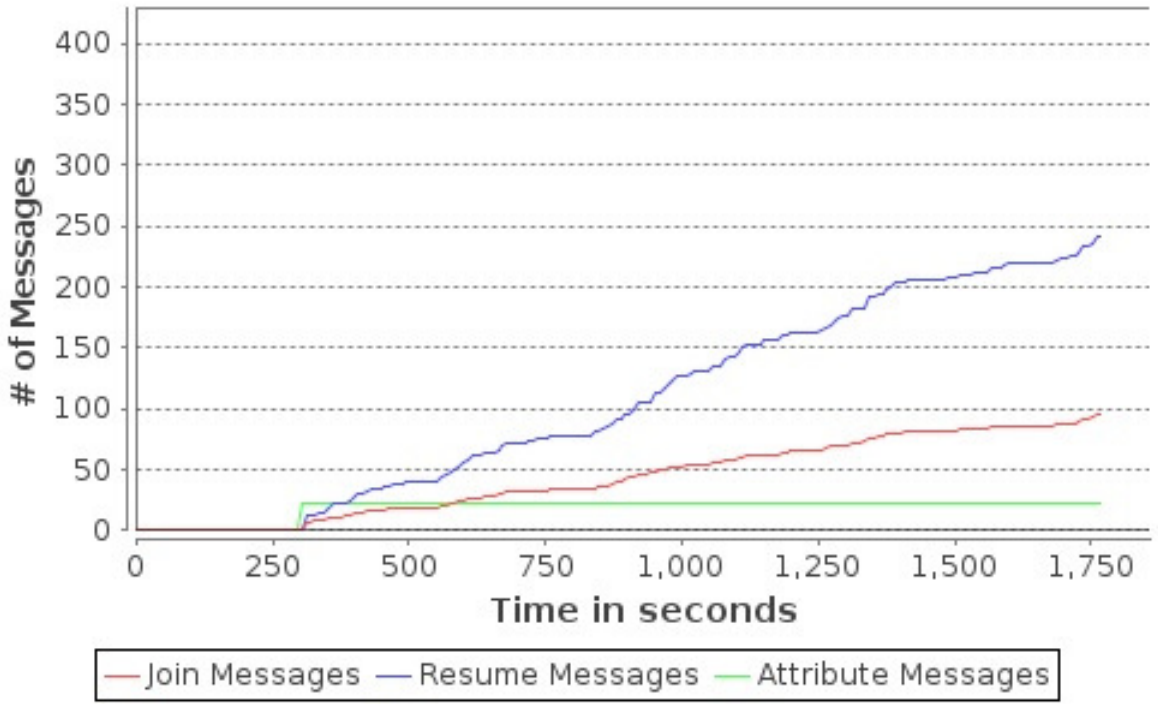


(β') Μη Σταθερό Δίκτυο

Σχήμα 5.16: Σύγκριση Σταθερού και μη δικτύου ως προς μεταβολές & ανταλλαγή μηνυμάτων

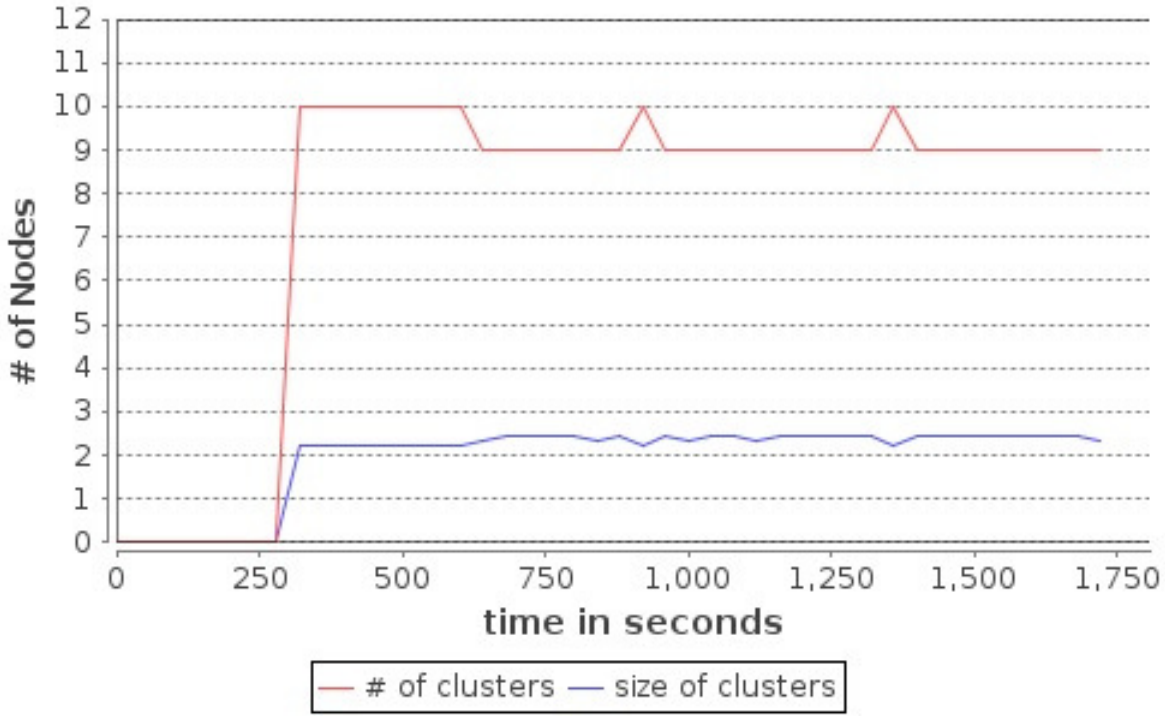


(α') Σταθερό Δίκτυο

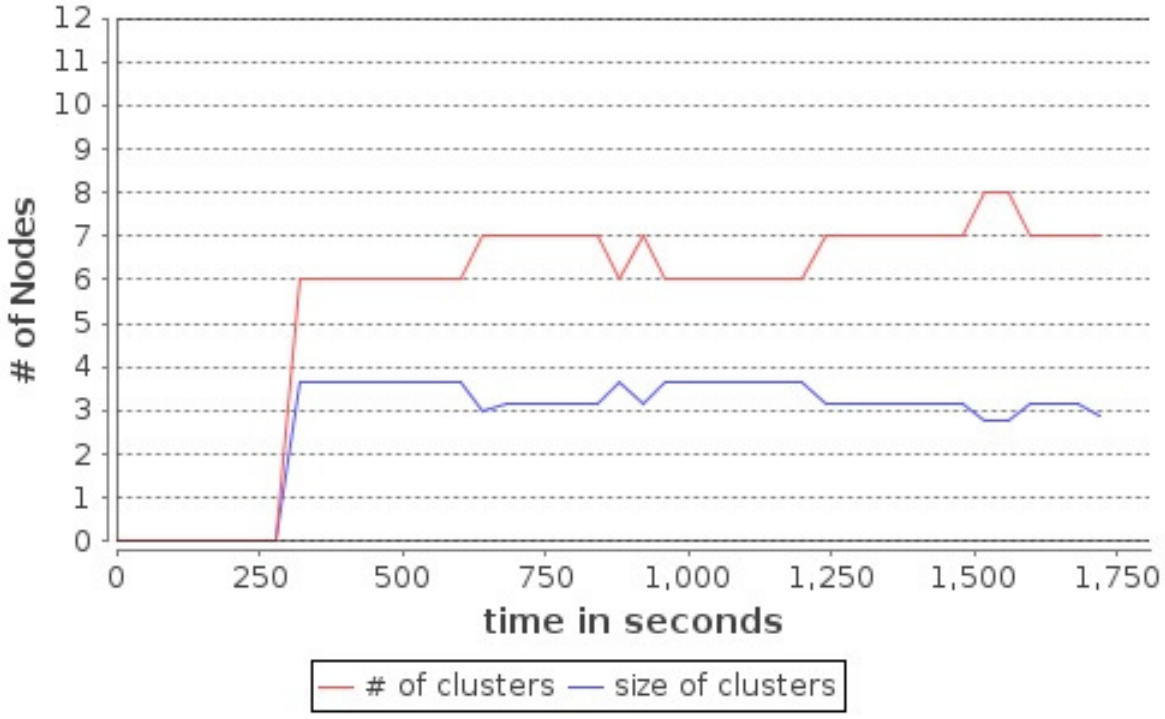


(β') Μη Σταθερό Δίκτυο

Σχήμα 5.17: Σύγκριση Σταθερού και μη δικτύου ως προς μεταβολές & ανταλλαγή μηνυμάτων

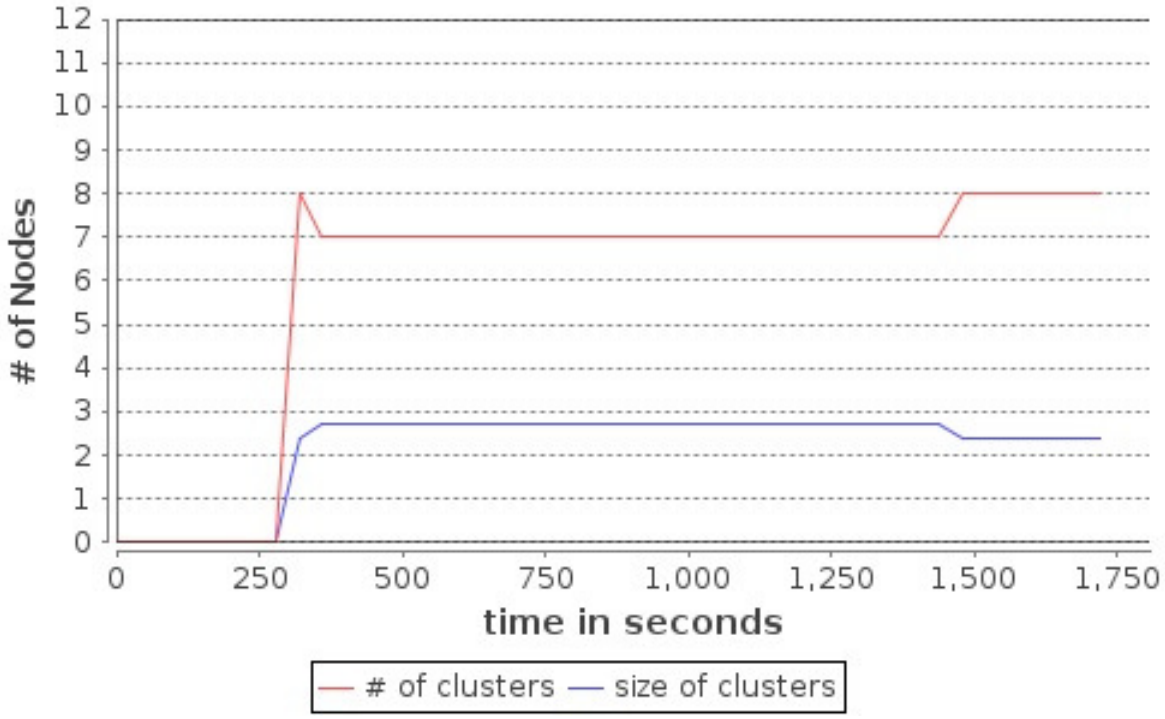


(α′) k=1

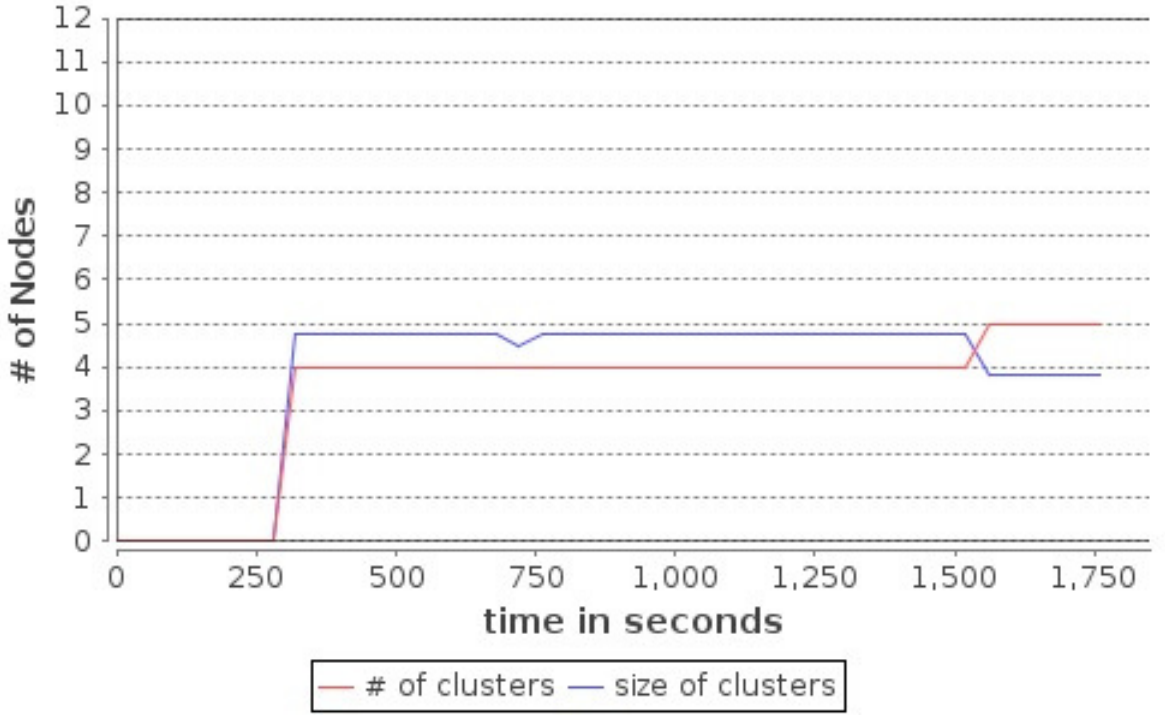


(β′) k=2

Σχήμα 5.18: εκτέλεση 30 λεπτών για μη Σταθερά δίκτυα



(α′) k=1



(β′) k=2

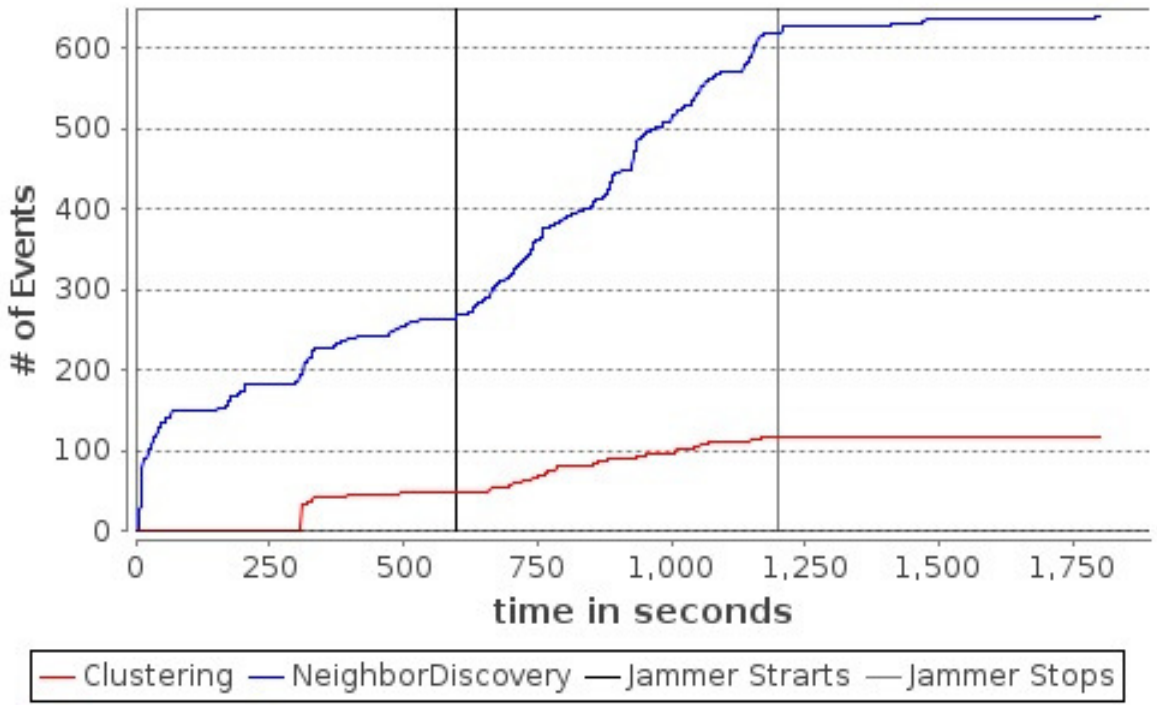
Σχήμα 5.19: εκτέλεση 30 λεπτών για μη Σταθερά δίκτυα

5.2.6 Πειράματα με χρήση Jammer

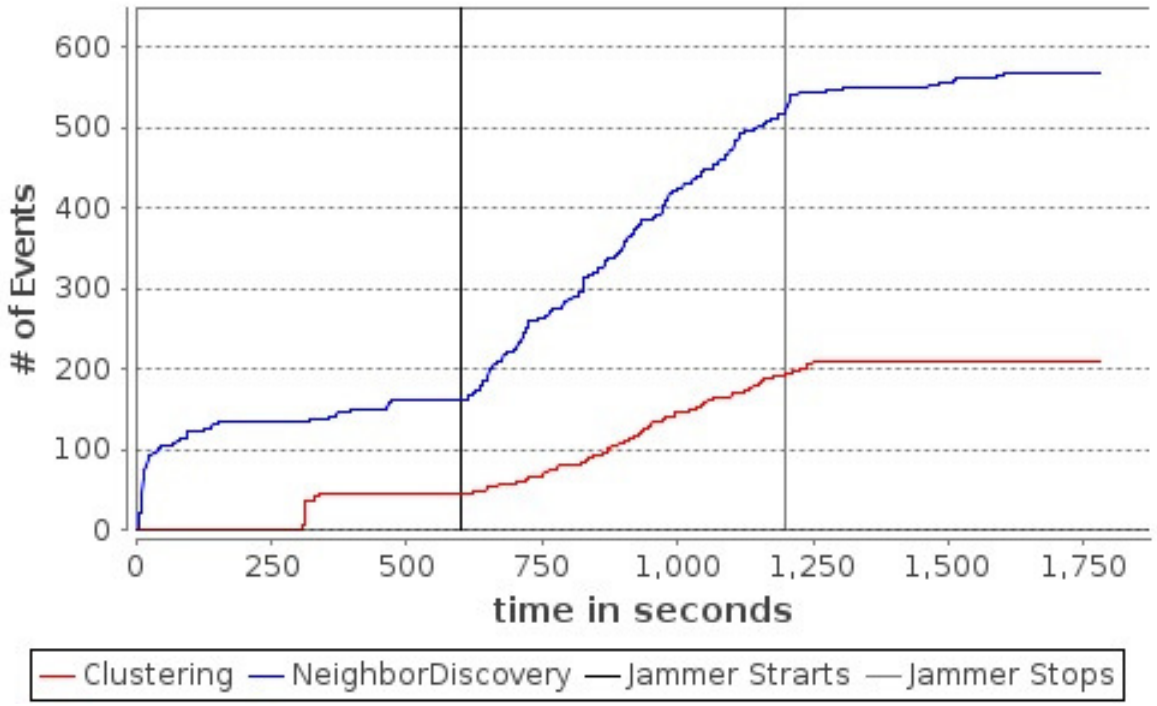
Στο επόμενο πείραμα γίνεται χρήση ενός ειδικού κόμβου που έχει ως στόχο να εμποδίσει την επικοινωνία των υπολοίπων. Τον κόμβο αυτό τον αποκαλούμε Jammer και στόχος του είναι να δημιουργεί συνεχώς κίνηση στο δίκτυο στέλνοντας μεγάλα πακέτα προς όλους τους κόμβους. Με αυτό τον τρόπο ο jammer χρησιμοποιεί συνεχώς το ασύρματο μέσο επικοινωνίας και παρεμποδίζει την επικοινωνία των υπολοίπων προκαλώντας συγκρούσεις μεταξύ των πακέτων και υποβαθμίζοντας την συνολική ποιότητα της επικοινωνίας. Κύριο χαρακτηριστικό της χρήσης αυτού του κόμβου είναι ότι δεν επηρεάζεται το συνολικό δίκτυο αλλά ένα μέρος αυτού. Συγκεκριμένα καθώς ο jammer έχει εμβέλεια ίση με την εμβέλεια των υπολοίπων κόμβων υπάρχει διαταραχή μόνο στην άμεση γειτονιά του. Τα πειράματα με την χρήση του jammer εκτελέστηκαν στο δίκτυο του EAITY. Τα πειραματικά αποτελέσματα επιβεβαιώνουν όπως φαίνεται και στο Σχήματα 5.20,5.21, 5.22,5.23 τις παραπάνω υποθέσεις.

Πιο συγκεκριμένα το πείραμα που εκτελούμε χωρίζεται σε 3 φάσεις. Αρχικά το ο Jammer είναι ανενεργός και αφήνουμε τις συσκευές να ομαδοποιηθούν σύμφωνα με τον αλγόριθμο. Μετά από 10 λεπτά και ενώ βλέπουμε ότι το δίκτυο έχει φτάσει σε στάσιμη κατάσταση ενεργοποιούμε τον Jammer για 10 λεπτά. Τέλος απενεργοποιούμε τον Jammer και αφήνουμε το δίκτυο να σταθεροποιηθεί εκ νέου. Οι 3 αυτές φάσεις φαίνονται χαρακτηριστικά στις γραφικές παραστάσεις του Σχήματος 5.20. Επίσης είναι εμφανές το γεγονός ότι η ύπαρξη του jammer μέσα στο δίκτυο επιδρά σημαντικά στην λειτουργία του αλγορίθμου καθώς ο αριθμός των αλλαγών τόσο στην γειτονιά όσο και στα clusters αυξάνεται γραμμικά, περισσότερο απότομα στην περίπτωση της άμεσης γειτονιάς του Jammer και λιγότερο στις ομάδες. Η διαφορά αυτή οφείλετε στο γεγονός ότι οι αλλαγές στη γειτονιά μπορεί να σημαίνουν και πτώση συνδέσεων ενός κόμβου με το γειτονικό cluster στο οποίο δεν ανήκει, κάτι που δεν προκαλεί μεταβολή στην εικόνα των clusters.

Αντίστοιχα χαρακτηριστικά παρατηρούμε και στα διαγράμματα που παρουσιάζουν τα clusters και τα μηνύματα που ανταλλάσσονται. Κατά την διάρκεια λειτουργίας του jammer βλέπουμε πως ο αριθμός των clusters που υπάρχουν μεταβάλλονται συνεχώς ενώ στην πρώτη και τρίτη φάση παραμένουν σταθεροί. Η διακύμανση αυτή είναι περισσότερο εμφανής στην περίπτωση που χρησιμοποιούμε $k = 2$ καθώς οι αλλαγές προωθούνται προς τον αρχηγό του cluster και προκαλούν μεγαλύτερη αναστάτωση.

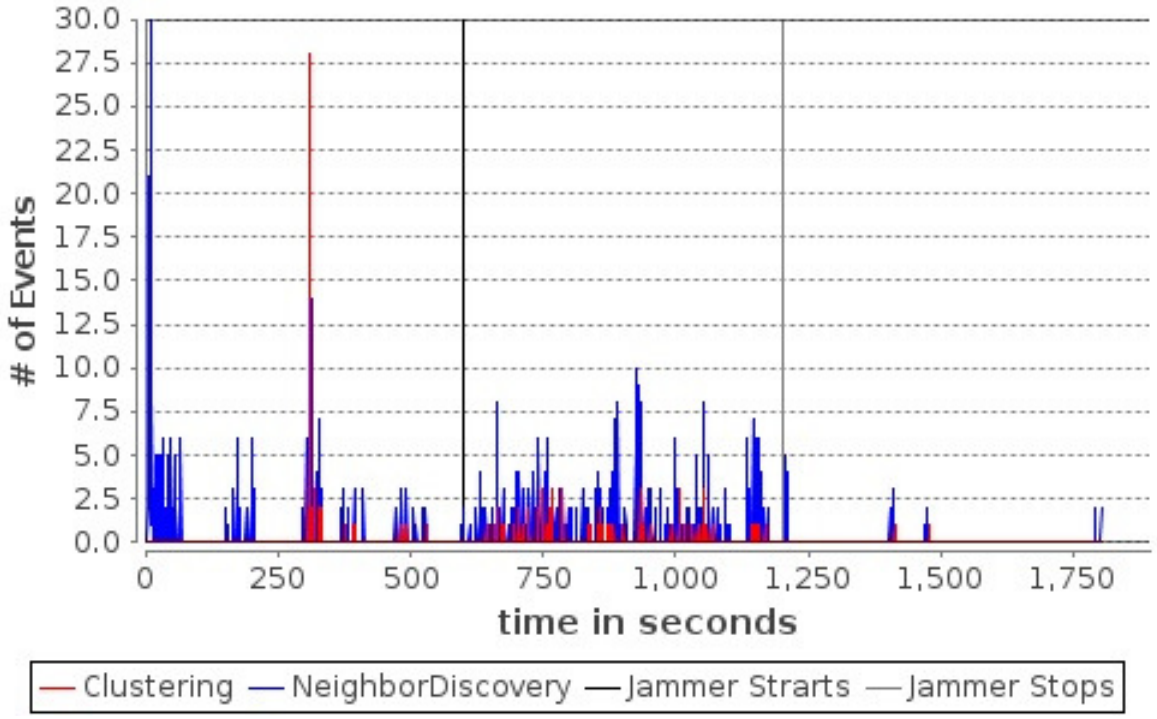


(α′) k=1

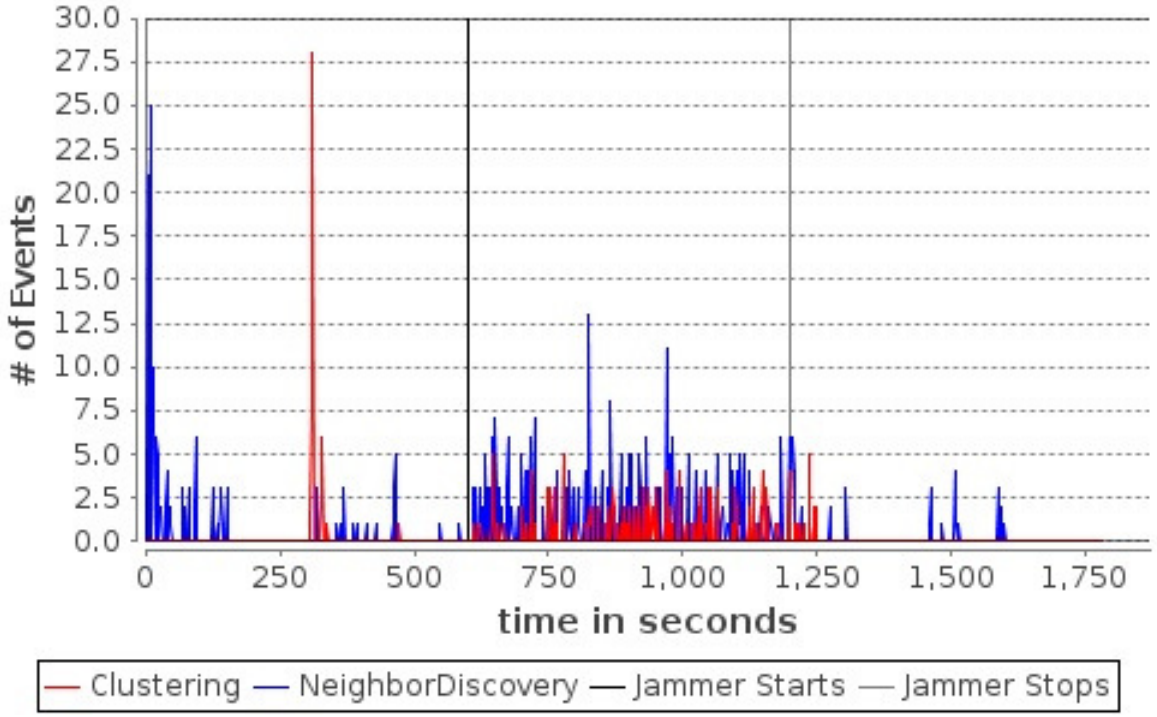


(β′) k=2

Σχήμα 5.20: Μεταβολές που προκαλούνται για εκτέλεση 30 λεπτών με χρήση Jammer

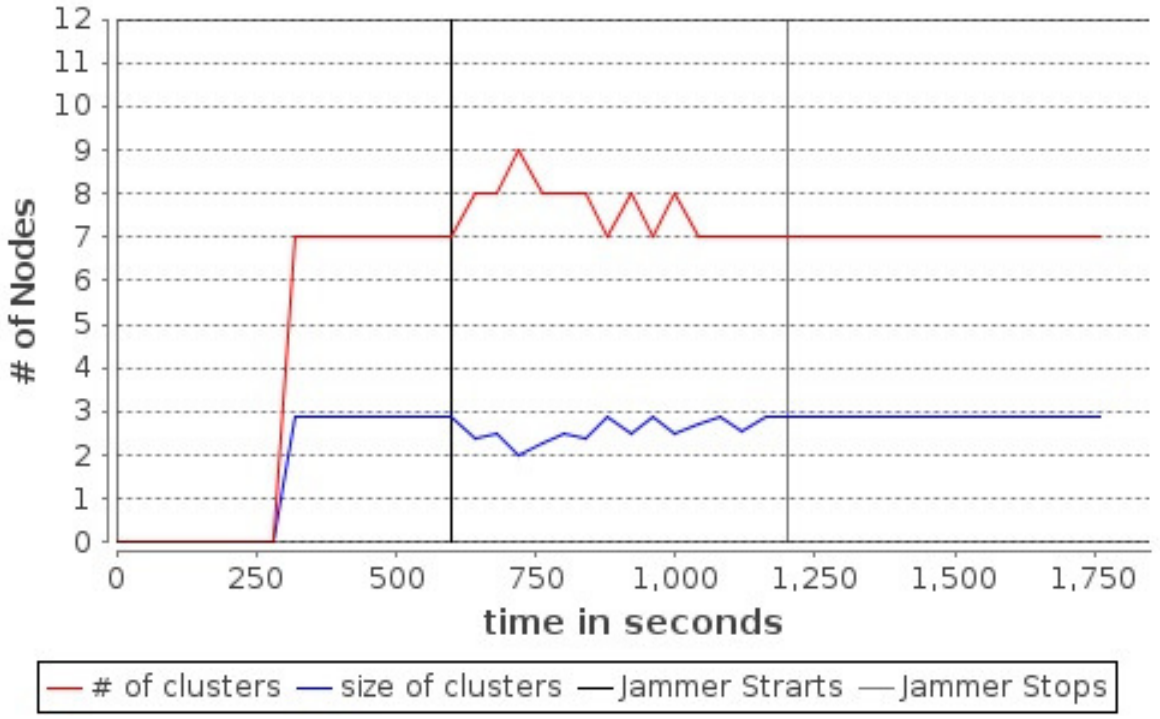


(α′) k=1

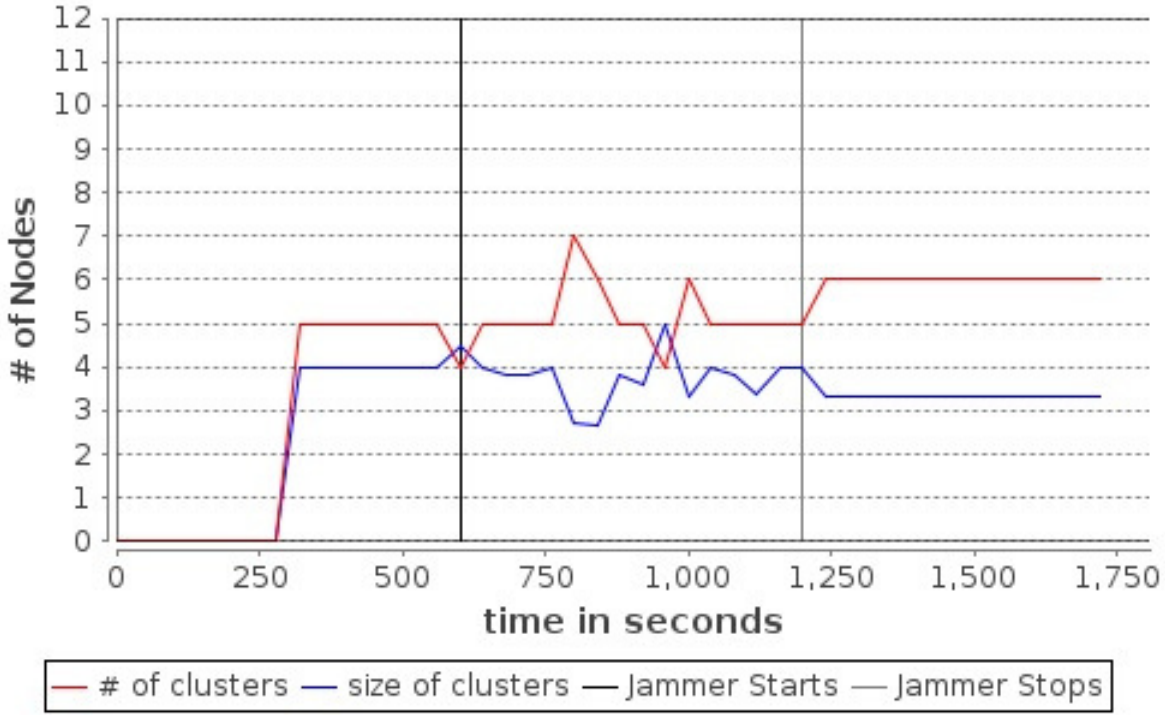


(β′) k=2

Σχήμα 5.21: Μεταβολές που προκαλούνται για εκτέλεση 30 λεπτών με χρήση Jammer



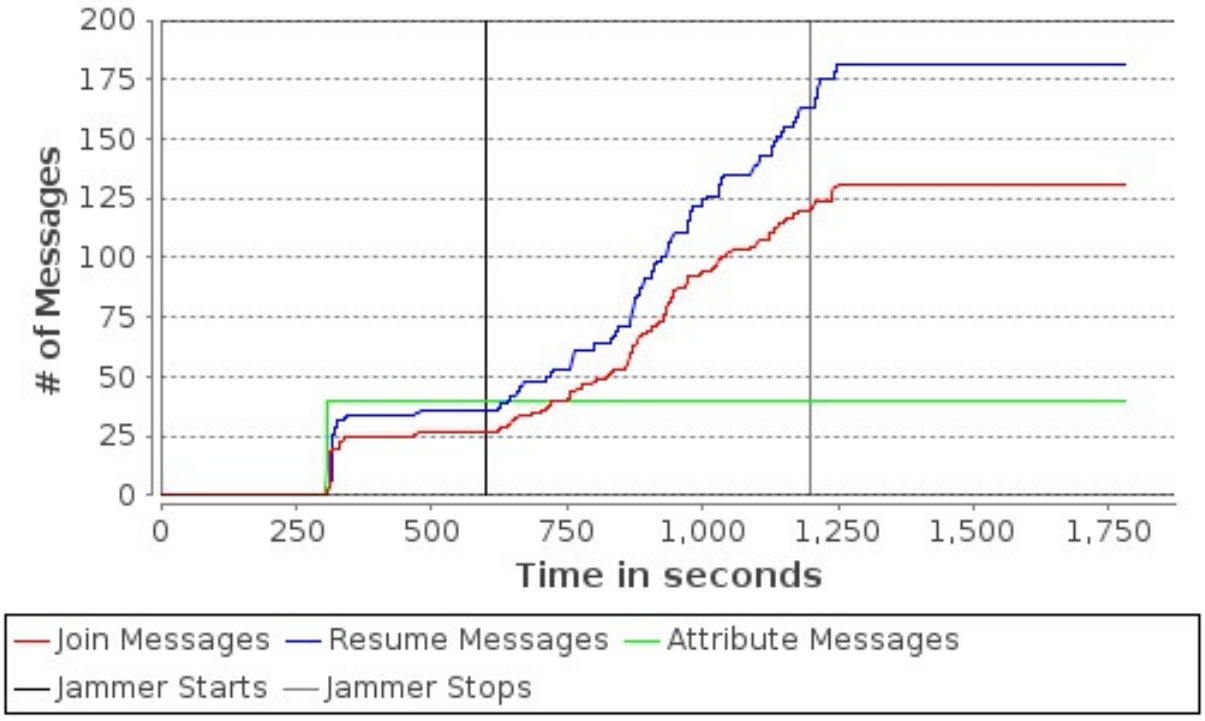
(α′) k=1



(β′) k=2

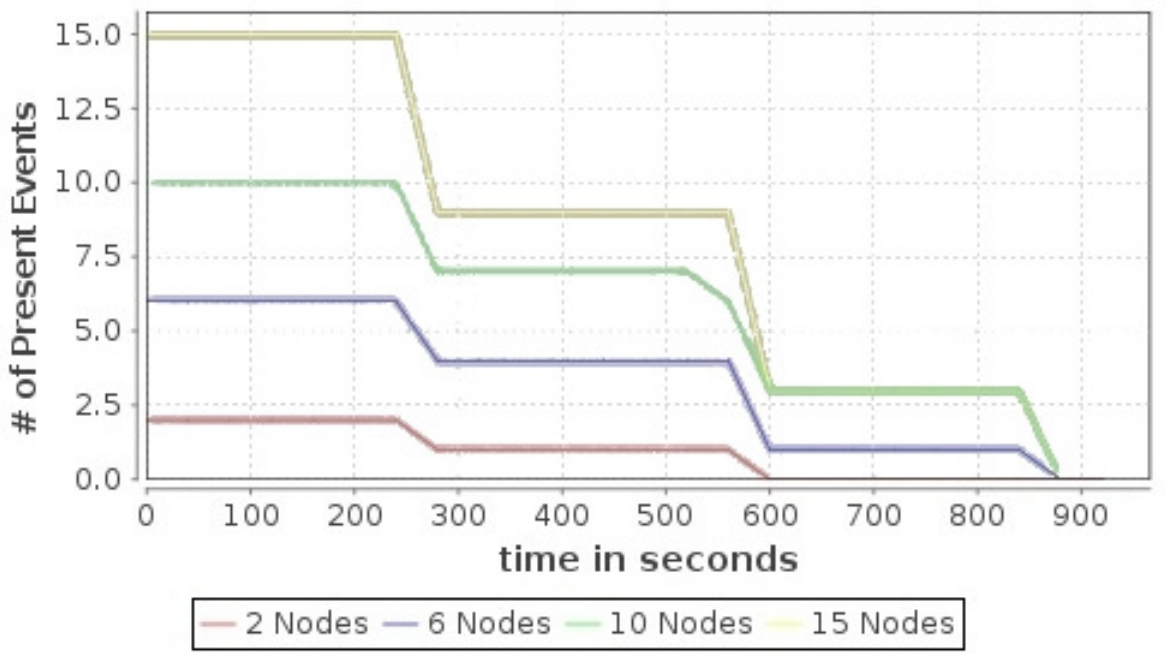
Σχήμα 5.22: Clusters που Δημιουργούνται για εκτέλεση 30 λεπτών με χρήση Jammer

(α′) k=1



(β′) k=2

Σχήμα 5.23: Μηνύματα που ανταλλάσονται για εκτέλεση 30 λεπτών με χρήση Jammer



Σχήμα 5.24: Ενεργοί κόμβοι στο δίκτυο εκτέλεση 15 λεπτών με εμφάνιση αστοχίας υλικού

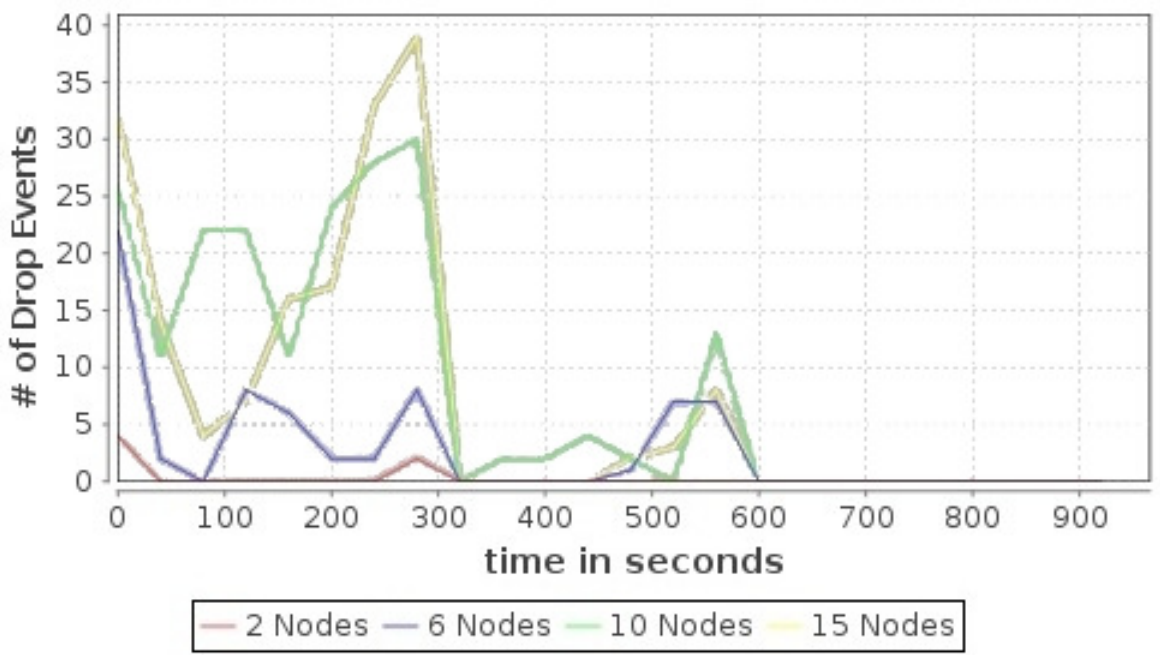
5.2.7 Πειράματα με βλάβες συσκευών

Στην συνέχεια γίνεται προσπάθεια να μελετηθεί η συμπεριφορά του αλγορίθμου απέναντι σε βλάβες που μπορεί να συμβούν στις ίδιες τις συσκευές. Αυτό μπορεί να οφείλεται είτε σε απώλεια ενέργειας μετά από μεγάλο χρονικό διάστημα λειτουργίας, βλάβη υλικού, ανθρώπινη παρέμβαση ή κάποιο άλλο τυχαίο συμβάν.

Στα πειράματα αυτά στόχος είναι να δείξουμε πως ο αλγόριθμος προσαρμόζεται σε αυτές τις δύσκολες καταστάσεις. Για να πετύχουμε και να προσημειώσουμε μια τέτοια κατάσταση βλάβης απενεργοποιούμε ένα ποσοστό των συσκευών μετά από ένα διάστημα λειτουργίας του συστήματος.

Σε αυτά τα πειράματα έχουμε και πάλι 3 φάσεις λειτουργίας. Στην πρώτη φάση διάρκειας 5 λεπτών οι συσκευές λειτουργούν κανονικά για να φέρουν το δίκτυο σε μια σταθερή κατάσταση. Στην δεύτερη φάση απενεργοποιείται ένα ποσοστό των ενεργών ClusterHeads και 5 λεπτά μετά απενεργοποιούμε και πάλι ένα ποσοστό των ενεργών ClusterHeads.

Ο τρόπος με τον οποίο αφαιρούνται οι κόμβοι από το δίκτυο είναι σημαντικός για την κατανόηση της συμπεριφοράς του αλγορίθμου. Στο

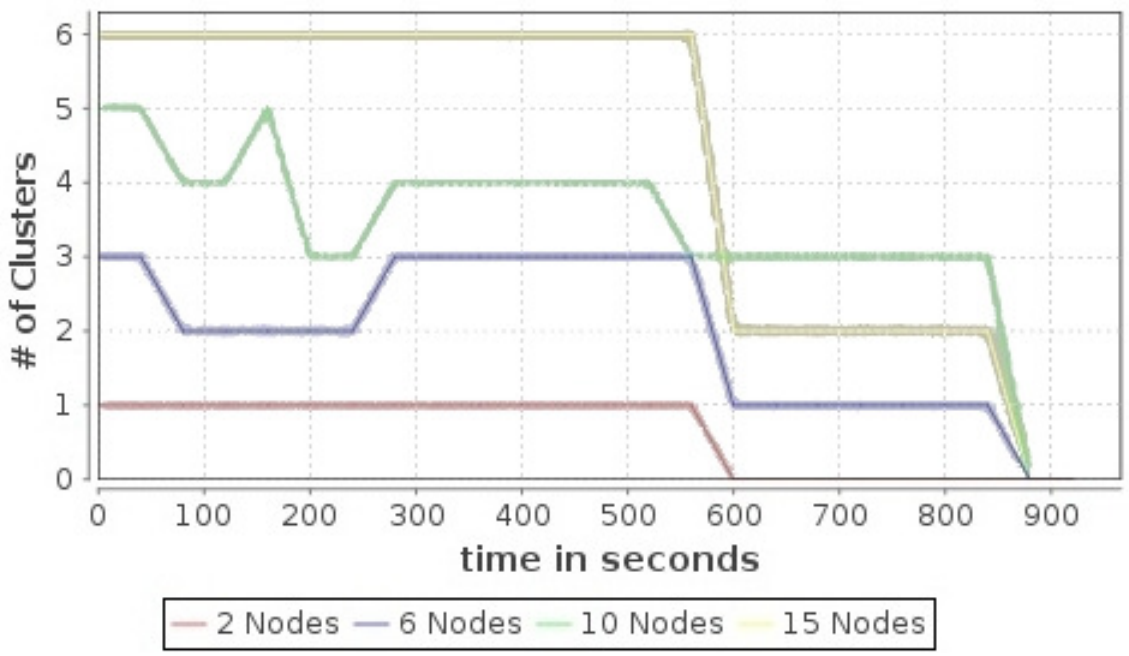


Σχήμα 5.25: Οι απώλειες επικοινωνίας με κόμβους στο δίκτυο εκτέλεση 15 λεπτών με εμφάνιση αστοχίας υλικού όπως αναφέρονται από τον Neighbor Discovery αλγόριθμο

Σχήμα 5.24 παρατηρούμε τον αριθμό των ενεργών κόμβων ως προς το χρόνο.

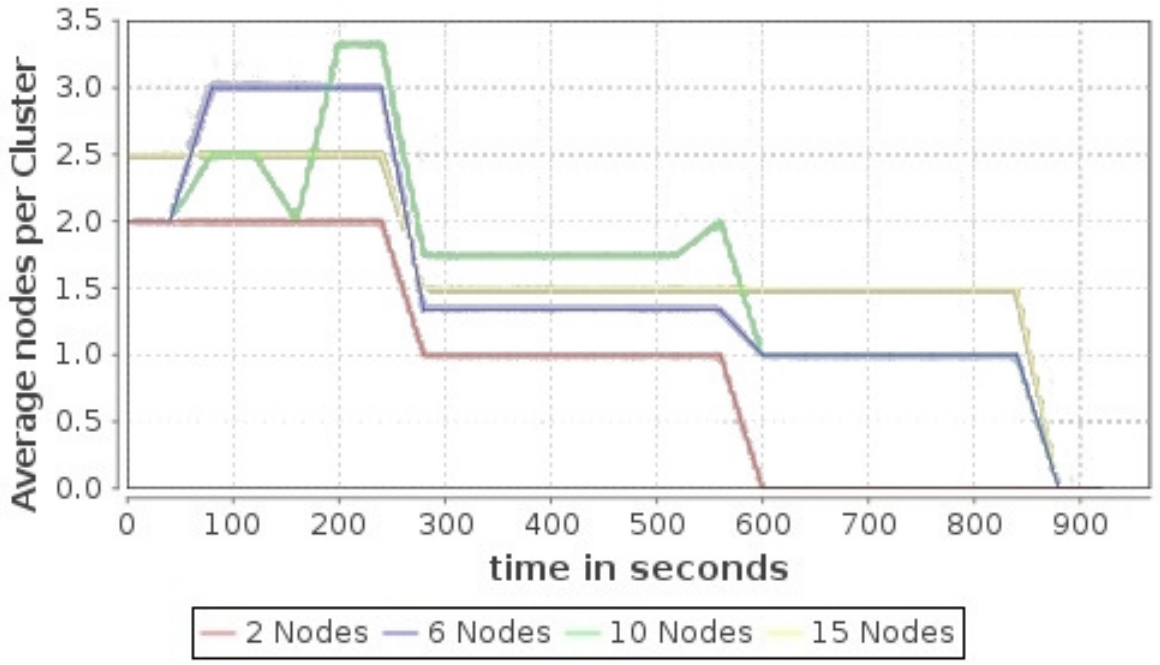
Επίσης στο Σχήμα 5.25 βλέπουμε τον τρόπο με τον οποίο ο Neighbor Discovery αλγόριθμος αναφέρει αυτές τις μεταβολές για την προσαρμογή του clustering σε αυτές.

Αρχικά παρουσιάζουμε το πως μεταβάλλεται ο αριθμός των σχηματισμένων clusters ως προς το χρόνο ενώ συμβαίνουν οι βλάβες των συσκευών. Όπως φαίνεται στο Σχήμα 5.26 για την περίπτωση των 2 συσκευών μετά την πρώτη αποτυχία ο αριθμός των clusters παραμένει 1 καθώς ο κόμβος ο οποίος είχε συνδεθεί στο πρώτο cluster γίνεται ClusterHead. Στην περίπτωση των 6 και 10 κόμβων παρατηρούμε μια αύξηση του αριθμού των clusters. Αυτό δεν είναι απόλυτα μη αναμενόμενο ή αφύσικο. Καθώς βγαίνουν κάποιες από τις συσκευές εκτός λειτουργίας το δίκτυο διασπάται και πολλές συνδέσεις του βγαίνουν εκτός λειτουργίας με αποτέλεσμα περισσότεροι κόμβοι να αυτοανακηρύσσονται ClusterHeads λόγω έλλειψης γειτόνων.



Σχήμα 5.26: Clusters που σχηματίζονται για εκτέλεση 15 λεπτών με εμφάνιση αστοχίας υλικού

Το πιο χαρακτηριστικό παράδειγμα για την συμπεριφορά των αλγορίθμων είναι αυτό των 15 συσκευών όπου σύμφωνα και με τις πληροφορίες του Σχήματος 5.27 μετά τις βλάβες που συμβαίνουν τα clusters παραμένουν τα ίδια σε αριθμό αλλά μειώνεται σημαντικά το μέγεθός τους.



Σχήμα 5.27: Μέγεθος Clusters που σχηματίζονται για εκτέλεση 15 λεπτών με εμφάνιση αστοχίας υλικού

Κεφάλαιο 6

Μελλοντικοί Στόχοι

Στόχος είναι στο μέλλον να προστεθούν στους ήδη υπάρχοντες αλγορίθμους και νέοι έτσι ώστε να καλυφθεί μεγαλύτερο μέρος της βιβλιογραφίας και να υπάρχουν διαθέσιμοι που να παρουσιάζουν όλα τα χαρακτηριστικά που μπορεί κάποιος να αναζητά. Συγκεκριμένα άμεσος στόχος είναι η ανάπτυξη αλγορίθμων clustering που χρησιμοποιούν και σημασιολογικά χαρακτηριστικά για τον σχηματισμό των clusters για να προκύπτει άμεση σχέση ανάμεσα στου αισθητήρες και στα αντικείμενα πάνω στα οποία είναι προσκολλημένοι.

Βιβλιογραφία

[1] *Max-min d-cluster formation in wireless ad hoc networks*, volume 1, 2000.

[2] A. A. Abbasi and M. Younis. A survey on clustering algorithms for wireless sensor networks. *Comput. Commun.*, 30(14-15):2826–2841, 2007.

[3] A. D. Amis, R. Prakash, D. Huynh, and T. Vuong. Max-min d-cluster formation in wireless ad hoc networks. In *INFOCOM*, pages 32–41, 2000.

[4] D. J. Baker and A. Ephremides. A distributed algorithm for organizing mobile radio telecommunication networks. In *ICDCS*, pages 476–483, 1981.

[5] S. Bandyopadhyay and E. J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *INFOCOM*, 2003.

[6] S. Banerjee and S. Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks. In *INFOCOM*, pages 1028–1037, 2001.

[7] C. K. Basilis Mamalis, Damianos Gavalas and G. Pantziou. *Clustering in Wireless Sensor Networks*, chapter in Book RFID and Sensor Networks: Architectures, Protocols, Security and Integrations. Taylor & Francis Group, 2009.

[8] T. Baumgartner, I. Chatzigiannakis, S. P. Fekete, C. Koninis, A. Kröller, and A. Pyrgelis. Wiselib: A generic algorithm library for heterogeneous sensor networks. In *Proceedings of the Seventh European Conference on Wireless Sensor Networks (EWSN 2010)*, pages 162–177, Coimbra, Portugal, February 2010. Springer-Verlag LNCS 5970.

[9] L. M. A. C. and N. Nasser. Comparison of clustering algorithms and protocols for wireless sensor networks. In *CCECE*, pages 1787–1792, 2006.

[10] H. Chan and A. Perrig. Ace: An emergent algorithm for highly uniform cluster formation. In *EWSN*, pages 154–171, 2004.

[11] M. Chatterjee, S. K. Das, and D. Turgut. Wca: A weighted clustering algorithm for mobile ad hoc networks. *Cluster Computing*, 5(2):193–204, 2002.

[12] Y. P. Chen, A. Liestman, and J. Liu. Clustering algorithms for ad hoc wireless networks. *Ad Hoc and Sensor Networks*, 30:2826–2841, 2007.

[13] S. Chinara and S. K. Rath. A survey on one-hop clustering algorithms in mobile ad hoc networks. *J. Netw. Syst. Manage.*, 17(1-2):183–207, 2009.

[14] S. D. C. K. A. P. Dimitrios Amaxilatis, Ioannis Chatzigiannakis and P. G. Spirakis. Adaptive hierarchical network structures for wireless sensor networks. In *ADHOCNETS 2011*, 2011.

[15] P. Ding, J. Holliday, and A. Celik. Distributed energy-efficient hierarchical clustering for wireless sensor networks. In *DCOSS*, pages 322–339, 2005.

[16] m. L. D.J. Dechene, A. El Jardali and A. Sauer. A survey of clustering algorithms for wireless sensor networks, 2006.

[17] K. Erciyes, O. Dagdeviren, D. Cokuslu, and D. Ozsoyeller. Graph theoretic clustering algorithms in mobile ad hoc networks and wireless sensor networks survey, 2007.

[18] B. Guo and Z. Li. A dynamic-clustering reactive routing algorithm for wireless sensor networks. *Wireless Networks*, 15(4):423–430, 2009.

[19] W. R. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *33rd IEEE Hawaii International Conference on System Sciences (HICSS 2000)*, page 8020, 2000.

[20] C. Jiang, D. Yuan, and Y. Zhao. Towards clustering algorithms in wireless sensor networks: a survey. In *WCNC'09: Proceedings of the 2009 IEEE conference on Wireless Communications & Networking Conference*, pages 2009–2014, Piscataway, NJ, USA, 2009. IEEE Press.

[21] A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer. Shawn: A new approach to simulating wireless sensor networks. In *Proceedings of the Design, Analysis, and Simulation of Distributed Systems Symposium 2005 (DASD' 05)*, pages 117–124, Apr. 2005.

[22] G. Li and T. Znati. Reca: a ring-structured energy-efficient clustering architecture for robust communication in wireless sensor networks. *IJSNet*, 2(1/2):34–43, 2007.

[23] A. Manjeshwar and D. P. Agrawal. Teen: Arouting protocol for enhanced efficiency in wireless sensor networks. In *IPDPS*, page 189, 2001.

[24] A. Parekh. Selecting routers in ad hoc wireless networks. In *Proccedings of ITS, Rio-de-Janeiro,Brazil*, pages 420–424, 1994.

[25] S. Selvakennedy and S. Sinnappan. An adaptive data dissemination strategy for wireless sensor networks. *IJDSN*, 3(1):23–40, 2007.

[26] R. Virrankoski, D. Lymberopoulos, and A. Savvides. Tasc: topology adaptive spatial clustering for sensor networks. In *in IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 605–614, 2005.

[27] C.-Y. Wen and W. A. Sethares. Automatic decentralized clustering for wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.*, 2005(5):686–697, 2005.

[28] Y. Xu and H. Qi. Decentralized reactive clustering for collaborative processing in sensor networks. In *ICPADS*, pages 54–, 2004.

[29] K. Yanagihara, J. Taketsugu, K. Fukui, S. Fukunaga, S. Hara, and K.-I. Kitayama. Eacle: Energy-aware clustering scheme with transmission power control for sensor networks. *Wirel. Pers. Commun.*, 40(3):401–415, 2007.

[30] M. Ye, C. Li, G. Chen, and J. Wu. An energy efficient clustering scheme in wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, 3(2-3):99–119, 2007.

[31] S. Yoon and C. Shahabi. The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks. *TOSN*, 3(1):3, 2007.

[32] O. Younis and S. Fahmy. Heed: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Trans. Mob. Comput.*, 3(4):366–379, 2004.

[33] A. M. Youssef, M. F. Younis, M. Youssef, and A. K. Agrawala. Distributed formation of overlapping multi-hop clusters in wireless sensor networks. In *GLOBECOM*, 2006.