Ατομική Διπλωματική Εργασία

ΑΝΑΠΤΥΞΗ, ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΠΕΙΡΑΜΑΤΙΚΗ ΑΞΙΟΛΟΓΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΕΛΕΓΧΟΥ ΣΥΜΦΟΡΗΣΗΣ ΣΕ ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ ΜΕ ΧΡΗΣΗ ΚΙΝΗΤΩΝ ΑΙΣΘΗΤΗΡΩΝ

Αντωνία Νικολάου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΠΤΥΞΗ, ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΠΕΙΡΑΜΑΤΙΚΗ ΑΞΙΟΛΟΓΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΕΛΕΓΧΟΥ ΣΥΜΦΟΡΗΣΗΣ ΣΕ ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ ΜΕ ΧΡΗΣΗ ΚΙΝΗΤΩΝ ΑΙΣΘΗΤΗΡΩΝ

Αντωνία Νικολάου

Επιβλέποντες Καθηγητές
Γεωργίου Χρύσης
Βασιλείου Βάσος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου
Μάιος 2015

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους επιβλέποντες καθηγητές μου Δρ Γεωργίου Χρύση και Δρ Βασιλείου Βάσο, για την υποστήριξη και την καθοδήγηση κατά την εκπόνηση της παρούσας διπλωματικής εργασίας, αλλά και για την επίτευξη μιας άψογης συνεργασίας. Ο χρόνος που αφιέρωσαν, το ενδιαφέρον και οι γνώσεις τους ήταν καθοριστικές για την πραγμάτωση της εργασίας αυτής.

Ακόμη θα ήθελα να ευχαριστήσω τον κ. Αριστόδημο Παφίτη και τον Δρ Χαράλαμπο Σεργίου για τις χρήσιμες πληροφορίες που μου έδωσαν σχετικά με τον προσομοιωτή που χρησιμοποίησα και την υλοποίηση του αλγορίθμου DAlPaS.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου που με στήριξε καθ' όλη τη διάρκεια των σπουδών μου.

Περίληψη

Τα ασύρματα δίκτυα αισθητήρων έχουν υποστεί μεγάλη ανάπτυξη τα τελευταία χρόνια. Ένα από τα δύσκολα θέματα στα ασύρματα δίκτυα αισθητήρων είναι ο έλεγχος συμφόρησης.

Στην παρούσα διπλωματική εργασία έχουν αναπτυχθεί τρεις αλγόριθμοι ελέγχου συμφόρησης οι οποίοι εκτελούνται όταν αποτυγχάνει ο αλγόριθμος ελέγχου συμφόρησης που χρησιμοποιεί το δίκτυο να αντιμετωπίσει τα θέματα συμφόρησης που παρουσιάζονται. Ο στόχος των αλγορίθμων είναι να παρέχουν στο δίκτυο επιπλέον πόρους για να καταφέρει να αντιμετωπίσει άμεσα τη συμφόρηση και να συνεχίσει να λειτουργεί κανονικά. Για αυτό το λόγο τοποθετούνται κινητοί κόμβοι αισθητήρων σε κατάλληλες θέσεις ώστε να δημιουργηθούν εναλλακτικά μονοπάτια. Ο πρώτος αλγόριθμος, ο Dynamic MobileCC, λύνει τοπικά κάθε πρόβλημα συμφόρησης τοποθετώντας ένα κινητό κόμβο να εξυπηρετεί κάποιους από τους κόμβους που στέλνουν δεδομένα στον κόμβο που παρουσιάζει συμφόρηση. Ο δεύτερος αλγόριθμος, ο Direct Path MobileCC, δημιουργεί όσα μονοπάτια απαιτούνται για να καλύψει τις ανάγκες του δικτύου χρησιμοποιώντας μόνο κινητούς κόμβους αισθητήρων. Ο τρίτος αλγόριθμος, ο Global MobileCC, θεωρείται ότι γνωρίζει εκ των προτέρων όλα τα θέματα συμφόρησης που θα εμφανιστούν στο δίκτυο και δημιουργεί τα βέλτιστα μονοπάτια, χρησιμοποιώντας τόσο κινητούς κόμβους αισθητήρων όσο και στατικούς κόμβους αισθητήρων. Αυτός ο αλγόριθμος είναι ο βέλτιστος δυνατός, αλλά δε μπορεί να υλοποιηθεί σε πραγματικά δίκτυα, εφόσον δεν υπάρχει εκ των πρότερων η γνώση για τους κόμβους που θα παρουσιάσουν συμφόρηση. Ο σχεδιασμός του έγινε για σκοπούς σύγκρισης του αριθμού των κινητών κόμβων που χρησιμοποιεί με τους άλλους δυο αλγορίθμους.

Οι τρεις αυτοί αλγόριθμοι υλοποιήθηκαν στον προσομοιωτή Prowler-Rmase. Τα αποτελέσματα της προσομοίωσης έχουν δείξει ότι οι αλγόριθμοι Dynamic MobileCC και Direct Path MobileCC, καταφέρνουν να διατηρήσουν το ποσοστό επιτυχίας, δηλαδή το συνολικό αριθμό των πακέτων που λαμβάνει ο κόμβος προορισμού έναντι του συνολικού αριθμού των πακέτων που αποστέλλονται από όλους τους πηγαίους κόμβους, καθ' όλη τη διάρκεια της προσομοίωσης

σταθερό σε ψηλό βαθμό, σε αντίθεση με τους συνηθισμένους αλγορίθμους ελέγχου συμφόρησης, στους οποίους τείνει να μειώνεται σταδιακά το success rate, λόγω αποτυχίας της αντιμετώπισης των θεμάτων συμφόρησης. Ο αλγόριθμος Direct Path MobileCC έχει ελαφρώς ψηλότερο ποσοστό επιτυχίας και ελαφρώς μικρότερο μέσο χρόνο αποστολής ενός μηνύματος από ένα πηγαίο κόμβο στον κόμβο προορισμού, σε σχέση με τον αλγόριθμο Dynamic MobileCC, αλλά χρησιμοποιεί τουλάχιστον τους διπλάσιους κινητούς κόμβους σε σχέση με τους κινητούς κόμβους που χρησιμοποιεί ο αλγόριθμος Dynamic MobileCC. Τονίζεται ότι ο αριθμός των κινητών κόμβων που χρησιμοποιεί ο αλγόριθμος Dynamic MobileCC σε κάποια σενάρια είναι ο ίδιος με του αλγόριθμου Global MobileCC, ενώ σε κάποια άλλα είναι ελάχιστα μεγαλύτερος. Επίσης ο αριθμός των κινητών κόμβων που χρησιμοποιούν οι αλγόριθμοι Dynamic MobileCC και Direct Path MobileCC αυξάνεται υπογραμμικά, καθώς αυξάνεται το πλήθος των κόμβων του δικτύου.

Τέλος καταλήγουμε στο συμπέρασμα ότι ο αλγόριθμος Dynamic MobileCC επιτυχαίνει τον σκοπό του με τη χρήση σχεδόν όσον κινητών κόμβων χρησιμοποιεί ο βέλτιστος αλγόριθμος Global MobileCC, ενώ ο αλγόριθμος Direct Path MobileCC κάνει σπατάλη κινητών κόμβων, οι οποίοι συνήθως κοστίζουν.

Περιεχόμενα

Κεφάλαιο 1

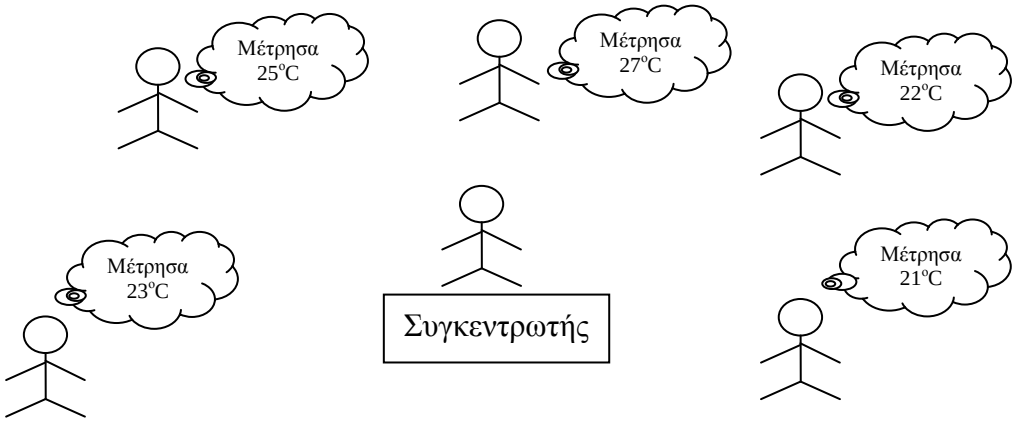
Εισαγωγή

Στο κεφάλαιο αυτό θα μελετηθούν τα κίνητρα που ώθησαν στην εκπόνηση της παρούσας διπλωματικής εργασίας, το σκοπό της, τη μεθοδολογία που ακολουθήθηκε και η δομή των επόμενων κεφαλαίων.

1.1 Κίνητρα

Ένα ασύρματο δίκτυο αισθητήρων είναι ένα δίκτυο υπολογιστών το οποίο αποτελείται από αυτόνομες συσκευές κατανεμημένες στο χώρο οι οποίες χρησιμοποιούν αισθητήρες με σκοπό τη συλλογή μετρήσεων φυσικών μεγεθών, όπως τη μέτρηση της θερμοκρασίας, της υγρασίας, του ήχου, της εικόνας, της δόνησης, της πίεσης και της κίνησης καθώς και τη μετάδοση των μετρήσεων αυτών σε ένα κεντρικό σημείο. Ένας κόμβος προορισμού συλλέγει τις πληροφορίες που δημιουργήθηκαν σε όλους τους υπόλοιπους κόμβους αισθητήρων του δικτύου. Αυτό το δίκτυο ονομάζεται single-sink γιατί υπάρχει μόνο ένας κόμβος προορισμού [5]. Η επικοινωνία είναι αμφίδρομη, δηλαδή μπορούν επίσης να μεταδίδονται δεδομένα από τον κόμβο προορισμού στους υπόλοιπους κόμβους του δικτύου.

Την ιδέα στο Σχήμα 1.1 θα μπορούσαμε να την υλοποιήσουμε με τη χρήση ασύρματων κόμβων αισθητήρων στη θέση των ανθρώπων με τα θερμόμετρα τους και του κόμβου προορισμού στη θέση του συγκεντρωτή.



Σχήμα 1.1: Σχεδιάγραμμα περιγραφής των ασύρματων δικτύων αισθητήρων

Τα ασύρματα δίκτυα αισθητήρων είναι μια πολύ σημαντική και συναρπαστική νέα τεχνολογία με τεράστιες προοπτικές για τη βελτίωση πολλών από τις υπάρχουσες εφαρμογές στην ιατρική, στη γεωργία, στον έλεγχο της βιομηχανικής διαδικασίας, στο στρατό και σε άλλους τομείς [6].

Τα ασύρματα δίκτυα αισθητήρων αποτελούν ένα πεδίο έρευνας που αναπτύσσεται ολοένα και με γρηγορότερους ρυθμούς τα τελευταία χρόνια. Αυτή η ανάπτυξη έχει καταστήσει απαραίτητη τη δημιουργία νέων εφαρμογών και το σχεδιασμό πρωτοκόλλων που να είναι ειδικά σχεδιασμένα για τις ιδιαίτερες απαιτήσεις των ασύρματων δικτύων αισθητήρων. Λόγω των περιορισμών των πόρων των κινητών αισθητήρων και της φύσης του ασύρματου περιβάλλοντος, που είναι επιρρεπής σε λάθη, τα πρωτόκολλα ελέγχου μετάδοσης δεδομένων που χρησιμοποιούνται στα ενσύρματα δίκτυα ή ακόμα και στα ασύρματα δίκτυα με υποδομή (infrastructure based wireless networks) δεν μπορούν να εφαρμοστούν στα ασύρματα δίκτυα αισθητήρων [20].

Παρόλο που έχουν αντιμετωπιστεί πολλά από τα προβλήματα των ασύρματων δικτύων αισθητήρων, το πρόβλημα της συμφόρησης παραμένει και έχει προσελκύσει πολλή προσοχή των ερευνητών. Αυτό είναι και το πιο σημαντικό πρόβλημα στα ασύρματα δίκτυα αισθητήρων που πρέπει να αντιμετωπιστεί για την αποδοτική λειτουργία του δικτύου (αναλύεται περαιτέρω στο Κεφάλαιο 2.2).

Διάφορες λύσεις έχουν προταθεί για να λυθεί το πρόβλημα της συμφόρησης. Μια δημοφιλής στρατηγική που κέρδισε πολύ ενδιαφέρον είναι ο μετριασμός της συμφόρησης (congestion mitigation). Ο μετριασμός της συμφόρησης γίνεται είτε με μείωση του ρυθμού μετάδοσης δεδομένων (traffic control) ή με αύξηση των πόρων (resource control). Συχνά όμως οι αλγόριθμοι αυτοί αποτυγχάνουν να επιλύσουν κάποια θέματα συμφόρησης. Οι αλγόριθμοι που αναπτύσσονται στην παρούσα διπλωματική εργασία μπορούν να χρησιμοποιηθούν ως επέκταση οποιουδήποτε αλγορίθμου που στοχεύει στην άμβλυνση της συμφόρησης.

1.2 Σκοπός της Διπλωματικής Εργασίας

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη, η υλοποίηση και η πειραματική αξιολόγηση τριών αλγορίθμων ελέγχου συμφόρησης που χρησιμοποιούν κινητούς κόμβους αισθητήρων και εκτελούνται όταν αποτυγχάνει ο αλγόριθμος ελέγχου συμφόρησης να δημιουργήσει εναλλακτικά μονοπάτια για να μεταδίδονται τα δεδομένα από τους πηγαίους κόμβους αισθητήρων στον κόμβο προορισμού. Ο αλγόριθμος που θα χρησιμοποιηθεί (ένας από τους τρεις αλγορίθμους που αναπτύχτηκαν στην παρούσα διπλωματική εργασία) δεν αντικαθιστά τους υπάρχων αλγόριθμους ελέγχου τοπολογίας ή τους αλγόριθμους ελέγχου συμφόρησης ή τους αλγόριθμους δρομολόγησης, αλλά τρέχει παράλληλα με αυτούς.

Ο σκοπός των αλγόριθμων που αναπτύχθηκαν είναι να παρέχουν στο δίκτυο επιπλέον πόρους για να καταφέρει να αντιμετωπίσει άμεσα τη συμφόρηση και να συνεχίσει να λειτουργεί κανονικά. Για το λόγο αυτό τοποθετούνται κινητοί κόμβοι αισθητήρων σε κατάλληλες θέσεις ώστε να δημιουργούνται εναλλακτικά μονοπάτια για τη μετάδοση των δεδομένων στον κόμβο προορισμού.

Έχουν αναπτυχθούν τρεις αλγόριθμοι ελέγχου συμφόρησης (MobileCC – Mobile Congestion Control) με διαφορετικά χαρακτηριστικά ο καθένας. Και οι τρεις αλγόριθμοι θεωρούν ότι ένας ικανοποιητικός αριθμός κινητών αισθητήρων, ανάλογα με το μέγεθος του δικτύου, είναι τοποθετημένοι αρχικά

κοντά στον κόμβο προορισμού. Ο πρώτος αλγόριθμος λύνει τοπικά το πρόβλημα τοποθετώντας ένα κινητό κόμβο να εξυπηρετεί κάποιους από τους κόμβους που στέλνουν τα δεδομένα τους στον κόμβο που παρουσιάζει συμφόρηση (congested node). Αυτός ο αλγόριθμος ονομάζεται Dynamic MobileCC. Ο δεύτερος αλγόριθμος δημιουργεί όσα μονοπάτια απαιτούνται σύμφωνα με τις ανάγκες του δικτύου χρησιμοποιώντας μόνο κινητούς κόμβους αισθητήρων. Αυτός ο αλγόριθμος ονομάζεται Direct Path MobileCC. Ο τρίτος αλγόριθμος μαζεύει όλες τις πληροφορίες για όλους τους κόμβους του δικτύου και δημιουργεί τα βέλτιστα μονοπάτια χρησιμοποιώντας τόσο κινητούς κόμβους αισθητήρων όσο και στατικούς κόμβους αισθητήρων. Αυτός ο αλγόριθμος ονομάζεται Global MobileCC. Αυτός ο αλγόριθμος δεν μπορεί να υλοποιηθεί σε πραγματικά συστήματα, εφόσον δεν υπάρχει εκ των πρότερων η γνώση για τους κόμβους που θα παρουσιάσουν συμφόρηση.

Υπάρχουν περιπτώσεις όπου ο αλγόριθμος ελέγχου συμφόρησης αποτυγχάνει να δημιουργήσει εναλλακτικά μονοπάτια για τη μετάδοση των δεδομένων στον κόμβο προορισμού, όμως είναι σημαντικό να αντιμετωπιστεί κάθε θέμα συμφόρησης που εμφανίζεται γιατί επιφέρει μείωση στο χρόνο ζωής του δικτύου, απώλεια πακέτων (άρα οι πληροφορίες δεν φθάνουν άμεσα στον κόμβο προορισμού), σπατάλη της ενέργειας των κόμβων αισθητήρων και μείωση του πλήθους των πακέτων που φθάνουν στον κόμβο προορισμού σε σχέση με το πλήθος των πακέτων που δημιουργούνται από τους πηγαίους κόμβους (success rate). Συνεπώς αυτή η επέκταση του αλγορίθμου ελέγχου συμφόρησης που χρησιμοποιείται από το δίκτυο, είναι πολύ σημαντικό να γίνει, εφόσον χρησιμοποιώντας την ικανότητα της κίνησης μερικών κόμβων δεν αποτυγχάνει η δημιουργία εναλλακτικών μονοπατιών για την μετάδοση των δεδομένων στον κόμβο προορισμού και άρα η συμφόρηση αντιμετωπίζεται αποδοτικά οποιαδήποτε χρονική στιγμή εμφανιστεί και σε οποιοδήποτε μέρος του δικτύου. Συνεπώς ο αλγόριθμος ελέγχου συμφόρησης με την προσθήκη του νέου αλγόριθμου που δημιουργήθηκε μπορεί να χρησιμοποιηθεί για την αποδοτική αντιμετώπιση της συμφόρησης σε οποιαδήποτε τοπολογία δικτύου.

1.3 Μεθοδολογία

Ένα σύστημα άμβλυνσης της συμφόρησης δημιουργώντας εναλλακτικές διαδρομές από τους πηγαίους κόμβους, στον κόμβο προορισμού είναι ο αλγόριθμος DAlPaS [1]. Η συμπεριφορά του έχει σχεδιαστεί και έχει αξιολογηθεί στον προσομοιωτή Prowler [4] μέσω της εφαρμογής Rmase [3]. Ο DAlPaS όμως δεν έχει επεκταθεί ούτως ώστε όταν δεν είναι πλέον εφικτό να δημιουργηθούν εναλλακτικά μονοπάτια για την προώθηση των πακέτων από τους πηγαίους κόμβους στον κόμβο προορισμού, να δημιουργούνται εναλλακτικά μονοπάτια χρησιμοποιώντας κινητούς κόμβους αισθητήρων. Για την αξιολόγηση των αλγορίθμων που ανατήχθηκαν στην παρούσα διπλωματική εργασία χρησιμοποιείται ο αλγόριθμος DAlPaS ως ο αλγόριθμος ελέγχου συμφόρησης που χρησιμοποιεί το δίκτυο. Σημειώνεται ότι οι αλγόριθμοι που αναπτύχθηκαν μπορούν να χρησιμοποιηθούν ως επέκταση οποιουδήποτε αλγόριθμου ελέγχου συμφόρησης.

Για την παρούσα διπλωματική εργασία, ακολουθήθηκε η εξής μεθοδολογία: Αρχικά, μελετήθηκε το αναγκαίο υπόβαθρο για την κατανόηση του αλγόριθμου DAlPaS [1], ο οποίος επεκτάθηκε στην εργασία και για την κατανόηση της κινητικότητας στα ασύρματα δίκτυα αισθητήρων. Ακολούθως, μελετήθηκαν τα χαρακτηριστικά στα οποία θα διαφέρουν οι εκδοχές του νέου αλγορίθμου που θα αναπτυχθεί και αποφασίστηκε να αναπτυχθούν τρεις εκδοχές του αλγορίθμου, για να είναι δυνατή η σύγκριση μεταξύ τους (εφόσον δεν υπάρχει προηγούμενη εργασία για να γίνουν συγκρίσεις). Στη συνέχεια, αναπτύχθηκαν οι τρεις αλγόριθμοι (dynamic algorithm , direct path algorithm, global algorithm) με βάση τις λειτουργίες που πρέπει να υποστηρίζουν και τους περιορισμούς που πρέπει να ικανοποιούν. Το επόμενο στάδιο ήταν η μελέτη του προσομοιωτή Rmase και της υλοποίησης του αλγορίθμου DAlPaS στον προσομοιωτή αυτό. Το τελικό στάδιο ήταν η υλοποίηση των τριών αλγορίθμων στον προσομοιωτή Rmase και η πειραματική αξιολόγηση τους.

1.4 Οργάνωση Κειμένου

Στο επόμενο Κεφάλαιο περιγράφονται αναλυτικά τα χαρακτηριστικά των ασύρματων δικτύων αισθητήρων, η συμφόρηση και η χρήση της κινητικότητας στα δίκτυα αυτά και γίνεται μια γενική περιγραφή των σχετικών άρθρων που μελετήθηκαν, συμπεριλαμβανομένου του αλγόριθμου DAlPaS ο οποίος χρησιμοποιείται για την υλοποίηση των αλγορίθμων που αναπτύσσονται στην παρούσα διπλωματική εργασία. Στο Κεφάλαιο 3 παρουσιάζεται αναλυτικά ο αλγόριθμος Dynamic MobileCC και παρουσιάζονται κάποια παραδείγματα της λειτουργιάς του αλγορίθμου. Στο Κεφάλαιο 4 επεξηγείται ο αλγόριθμος Direct Path MobileCC και κάποια παραδείγματα της λειτουργίας του αλγόριθμου. Στο Κεφάλαιο 5 περιγράφεται ο αλγόριθμος Global MobileCC και κάποια παραδείγματα της λειτουργίας του αλγορίθμου. Στο Κεφάλαιο 6 περιγράφονται ο προσομοιωτής Prowler και η εφαρμογή RMASE, επεξηγείται το περιβάλλον προσομοίωσης που χρησιμοποιήθηκε για την αξιολόγηση των αλγορίθμων και παρουσιάζεται η στοχευμένη και τυχαιοποιημένη αξιολόγηση που έγινε. Κλείνοντας στο Κεφάλαιο 7 παρουσιάζονται τα τελικά συμπεράσματα.

Κεφάλαιο 2

Υπόβαθρο

Στο κεφάλαιο αυτό αναλύονται τα κυριότερα χαρακτηριστικά των ασύρματων δικτύων αισθητήρων, επεξηγείται ο ορισμός της συμφόρησης και οι δυο τύποι της, καθώς επίσης γίνεται μια εισαγωγή για τις χρήσεις της κινητικότητας στα ασύρματα δίκτυα αισθητήρων και επεξηγούνται οι τρεις μορφές κινητικότητας στα δίκτυα αυτά. Στο τελευταίο υποκεφάλαιο περιγράφονται περιληπτικά τα σχετικά άρθρα που έχουν εκπονηθεί.

2.1 Χαρακτηριστικά των Ασύρματων Δικτύων Αισθητήρων

Τα ασύρματα δίκτυα αισθητήρων ανήκουν στην οικογένεια των ad-hoc δικτύων [8]. Το κύριο χαρακτηριστικό των ad-hoc δικτύων είναι η έλλειψη υποδομής και η δυνατότητα της αυτόνομης διαμόρφωσης των δικτύων που ανήκουν. Τα ασύρματα δίκτυα αισθητήρων έχουν επίσης αυτά τα χαρακτηριστικά, αλλά έχουν και κάποια άλλα χαρακτηριστικά.

Τα πιο σημαντικά από αυτά είναι τα εξής:

- Περιορισμένη ενέργεια των κόμβων αισθητήρων και Χρόνος ζωής του δικτύου:

  Ως πηγή ενέργειας χρησιμοποιούνται μπαταρίες, οι οποίες αδειάζουν, ή ανανεώσιμες πηγές ενέργειας όταν είναι εφικτό (για παράδειγμα ένα δίκτυο που μελετά συγκεκριμένα ψάρια στον ωκεανό δεν μπορεί να βασιστεί στον ήλιο ως ανανεώσιμη πηγή ενέργειας). Λόγω του μικρού μεγέθους και του χαμηλού κόστους, η ενέργεια σε αυτούς τους κόμβους είναι ζωτικής σημασίας. Όσο πιο χαμηλή είναι η κατανάλωση ενέργειας κάθε κόμβου ξεχωριστά, τόσο μεγαλύτερος είναι ο χρόνος ζωής του δικτύου και τόσο μικρότερο είναι το κόστος συντήρησης του δικτύου.

- Ρυθμός μετάδοσης δεδομένων :

  Λόγω της δυνατότητας των κόμβων αισθητήρων να αλληλεπιδρούν με το περιβάλλον, τα ασύρματα δίκτυα αισθητήρων έχουν εντελώς διαφορετικά χαρακτηριστικά μετάδοσης δεδομένων σε σχέση με άλλα δίκτυα. Για αυτό το λόγο, ο ρυθμός μετάδοσης δεδομένων για μεγάλο χρονικό διάστημα μπορεί να είναι πολύ χαμηλός και όταν συμβεί κάποιο γεγονός, να γίνει πολύ υψηλός.

  Η κύρια κατεύθυνση μετάδοσης δεδομένων είναι η κατεύθυνση από τους κόμβους αισθητήρων στον κόμβο προορισμού, αλλά μερικές φορές ο κόμβος προορισμού μπορεί να μεταδώσει δεδομένα προς τους υπόλοιπους κόμβους για έλεγχο.

- Αυτόνομη και προγραμματιζόμενη λειτουργία

  Κάθε κόμβος λειτουργεί αυτόνομα, δηλαδή ξέρει τον τρόπο που πρέπει να λειτουργήσει, όπως για παράδειγμα τη μέτρηση που πρέπει να πάρει, τη συχνότητα που πρέπει να το κάνει (συχνότητα δειγματοληψίας) και που πρέπει να στείλει τη μέτρηση

που πήρε, όπως για παράδειγμα Broadcast σε όλους τους κόμβους που βρίσκονται στο εύρος αίσθησης του. Παράλληλα όμως κάθε κόμβος έχει τη δυνατότητα να προγραμματιστεί δυναμικά, όπως για παράδειγμα μπορεί ο κόμβος προορισμού να διαδώσει στο δίκτυο καινούρια δεδομένα λειτουργίας για τον κάθε κόμβο και συνεπώς το δίκτυο να αναπρογραμματιστεί δυναμικά.

- Αυτόνομη - Διαμόρφωση (Self - Configuration):
  Οι περισσότερες από τις εφαρμογές δικτύων αισθητήρων απαιτούν την Αυτόνομη Διαμόρφωση της τοπολογίας των κόμβων του δικτύου. Η Αυτόνομη Διαμόρφωση είναι επίσης μια απαίτηση που προκύπτει, λόγω των περιορισμών που αφορούν τις πηγές ενέργειας των κόμβων αισθητήρων. Αυτοί οι περιορισμοί των πηγών ενέργειας συχνά οδηγούν στην απώλεια κάποιων κόμβων του δικτύου. Σε τέτοιες περιπτώσεις το δίκτυο δεν καταστρέφεται αλλά προσαρμόζεται και δημιουργεί νέα μονοπάτια μεταξύ των κόμβων για να διατηρηθεί η συνένωση τους.

- Συνεργασία μεταξύ των κόμβων:
  Τα ασύρματα δίκτυα αισθητήρων λειτουργούν όπως ακριβώς τα ad-hoc δίκτυα όσο αφορά τη μετάδοση των δεδομένων, δηλαδή κάθε κόμβος μπορεί να αισθανθεί και να μεταδίδει δεδομένα που δημιουργεί ή μπορεί να ενεργεί ως δρομολογητής, δηλαδή να μεταδίδει δεδομένα από άλλους κόμβους (hop by hop).

- Είναι ιδανικά για συγκεκριμένη εφαρμογή:
  Ένα ασύρματο δίκτυο αισθητήρων δημιουργείται για μια συγκεκριμένη εφαρμογή. Ανάλογα με την εφαρμογή τα χαρακτηριστικά του δικτύου αλλάζουν. Για παράδειγμα, η τοπολογία του δικτύου μπορεί να είναι διαφορετική, από πολύ πυκνή σε πολύ αραιή, το πρωτόκολλο επικοινωνίας που χρησιμοποιείται μπορεί να διαφέρει, ακόμη και το μέγεθός του

δικτύου μπορεί να διαφέρει. Είναι αδύνατο να υπάρξει ένα δίκτυο που να ταιριάζει σε όλες τις εφαρμογές.

- Ποιότητα υπηρεσίας:
  Η ποιότητα των παρεχόμενων υπηρεσιών στα ασύρματα δίκτυα αισθητήρων εξαρτάται από την εφαρμογή. Αυτό σημαίνει ότι η ποιότητα υπηρεσίας καθορίζεται από την επιτυχία του σκοπού της εφαρμογής και όχι από τον αριθμό των πακέτων που χάθηκαν κλπ.

2.2 Συμφόρηση στα Ασύρματα Δίκτυα Αισθητήρων

2.2.1 Γενικά

Η συμφόρηση μπορεί να παρουσιαστεί κατά τη μετάδοση των πακέτων από τους πηγαίους κόμβους στον κόμβο προορισμού. Η συμφόρηση στα ασύρματα δίκτυα αισθητήρων έχει αρνητικές επιπτώσεις στην απόδοση του δικτύου και στο σκοπό της εφαρμογής, εφόσον αυξάνει τον αριθμό των πακέτων που χάνονται, την καθυστέρηση αποστολής των πακέτων, τη σπατάλη της ενέργειας των κόμβων κ.α. [7].

Η παρούσα διπλωματική εργασία επικεντρώνεται σε δίκτυα βάσει γεγονότων (event-based networks). Το χαρακτηριστικό αυτών των δικτύων είναι η δημιουργία και η μετάδοση ενός μεγάλου αριθμού πακέτων δεδομένων στον κόμβο προορισμού όταν συμβεί κάποιο γεγονός, το οποίο ικανοποίει μια συνθήκη η οποία έχει προκαθοριστεί, δηλαδή όταν το δίκτυο βρίσκεται σε κατάσταση κρίσης [19]. Σε αυτή την κατάσταση, ο αριθμός των πακέτων που δημιουργούνται και πρέπει να μεταδοθούν στον κόμβο προορισμού είναι τόσο μεγάλος που εάν δεν χρησιμοποιηθεί κάποιος μηχανισμός ελέγχου, μπορεί εύκολα να οδηγήσει σε συμφόρηση [2].

Η συμφόρηση σε αυτό τον τύπο δικτύων είναι άκρως ανεπιθύμητη, όχι μόνο λόγω της σπατάλης της ενέργειας των κόμβων που επιφέρει, αλλά επίσης λόγω του γεγονότος ότι μπορεί να εμποδίσει το δίκτυο να πετύχει το σκοπό του (σκοπό της εφαρμογής). Τα πακέτα δεδομένων τα οποία δημιουργούνται όταν

ένα δίκτυο τύπου event-based βρίσκεται σε κρίσιμη κατάσταση είναι σημαντικά για την ικανοποίηση του στόχου του δικτύου και συνεπώς για την επιτυχία της εφαρμογής.

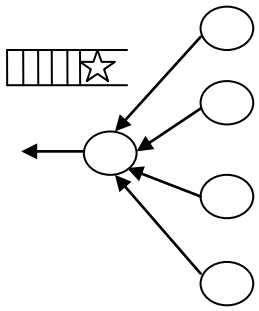
Ένα χαρακτηριστικό παράδειγμα είναι η ανίχνευση πυρκαγιάς στο δάσος μέσω ασύρματων κόμβων αισθητήρων, οι οποίοι προγραμματίζονται να στέλνουν πακέτα στον κόμβο προορισμού όταν η θερμοκρασία στο δάσος είναι μεγαλύτερη από μία προκαθορισμένη τιμή. Σε περίπτωση πυρκαγιάς, οι κόμβοι αισθητήρων οι οποίοι είναι κοντά στο συμβάν, αισθάνονται τη ψηλή θερμοκρασία κι έτσι αρχίζει η μετάδοση μιας μεγάλης ποσότητας πακέτων ταυτόχρονα. Ο σκοπός του δικτύου αυτού είναι η συνεχής αποστολή πακέτων δεδομένων στον κόμβο προορισμού (π.χ. σε τοπικούς πυροσβεστικούς σταθμούς), προκειμένου να ενημερώνονται για την εξάπλωση της πυρκαγιάς. Εάν η μεγάλη ποσότητα πακέτων δεδομένων, παραμείνει χωρίς κανένα έλεγχο συμφόρησης, το δίκτυο θα παρουσιάσει συμφόρηση και τα πακέτα δεν θα φθάνουν στον κόμβο προορισμού. Αυτό θα έχει ως αποτέλεσμα την αποτυχία του στόχου του δικτύου. Λόγω της μεγάλης σημασίας αυτών των πακέτων και των αυστηρών περιορισμών του χρόνου, η μείωση του ρυθμού μετάδοσης των πακέτων (data rate) μπορεί να επιφέρει την αποτυχία του στόχου του δικτύου (π.χ. σε ένα δάσος το οποίο καίγεται, σχεδόν κάθε πακέτο δεδομένων παρέχει νέες πληροφορίες για την εξάπλωση της πυρκαγιάς).

2.2.2 Τύποι Συμφόρησης

Η συμφόρηση μπορεί να ταξινομηθεί σε δύο κατηγορίες ανάλογα με τον τρόπο που χάνονται τα πακέτα.

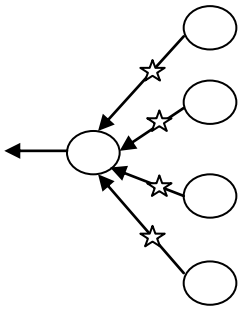
Η πρώτη κατηγορία ονομάζεται συμφόρηση στον κόμβο (node-level congestion). Συμβαίνει όταν ο αριθμός των πακέτων που λαμβάνει ένας κόμβος-αισθητήρας υπερβαίνει τη χωρητικότητα του εξερχόμενου καναλιού του και έχει ως αποτέλεσμα τα πακέτα να συσσωρεύονται στο buffer του κόμβου-αισθητήρα, όπως φαίνεται στο Σχήμα 2.1. Εάν το πρόβλημα επιμένει, το μήκος της ουράς υπερβαίνει τη χωρητικότητα του buffer, και αυτό έχει ως αποτέλεσμα

την απώλεια πακέτων και την αύξηση της καθυστέρησης στην ουρά του κόμβου (queuing delay) [7].



Σχήμα 2.1: Σχεδιάγραμμα περιγραφής node-level congestion

Η δεύτερη κατηγορία ονομάζεται συμφόρηση συνδέσμου (link-level congestion) και συμβαίνει λόγω της φύσης των ασύρματων δικτύων αισθητήρων, το κοινό μέσο επικοινωνίας και το περιορισμένο εύρος ζώνης. Συμβαίνει όταν πολλοί γειτονικοί κόμβοι αισθητήρων (που είναι στην εμβέλεια ο ένας του άλλου) προσπαθούν να έχουν πρόσβαση στο ασύρματο μέσο ταυτόχρονα, όπως φαίνεται στο Σχήμα 2.2. Αυτό το είδος της συμφόρησης μειώνει τη χρήση του συνδέσμου και τη συνολική απόδοση του δικτύου και παράλληλα αυξάνει την καθυστέρηση της αποστολής των πακέτων και τη σπατάλη της ενέργειας των κόμβων [7]. Το πρόβλημα επιδεινώνεται όταν η τοπολογία του δικτύου είναι πυκνή [7].



Σχήμα 2.2: Σχεδιάγραμμα περιγραφής link-level congestion

2.3 Κινητικότητα στα ασύρματα δίκτυα αισθητήρων

2.3.1 Γενικά

Η εξελιγμένη τεχνολογία δημιούργησε την ανάγκη για δημιουργία πιο σύνθετων εφαρμογών, που απαιτούν την κινητικότητα των κόμβων του δικτύου [21]. Όλο και πιο σύγχρονοι κόμβοι αισθητήρων αναπτύσσονται, όμως η εισαγωγή της κινητικότητας σε αυτά τα δίκτυα δίνει ακόμη περισσότερες δυνατότητες χρήσης των ασύρματων δικτύων αισθητήρων και μας ωθεί να βάλουμε περισσότερο στις ζωές μας αυτά τα δίκτυα. Η κινητικότητα των κόμβων μπορεί να διευρύνει τις εφαρμογές που χρησιμοποιούνται τα ασύρματα δίκτυα αισθητήρων [22].

Οι κινητοί κόμβοι μπορούν για παράδειγμα να εφαρμόζονται πάνω σε ζώα, ρομπότ, αυτοκίνητα, ακόμη και ανθρώπους και έτσι να παρατηρούνται διάφορα περιβαλλοντικά μεγέθη. Επίσης μπορούμε να παρατηρήσουμε ένα κινούμενο στόχο που δεν μπορούμε να πλησιάσουμε, αλλά θέλουμε να συλλέξουμε πληροφορίες από αυτόν. Επιπλέον η κινητικότητα μπορεί να αυξήσει τη συνδεσιμότητα μεταξύ των κόμβων, δεδομένου ότι οι κινητοί κόμβοι μπορούν να βοηθήσουν την επικοινωνία μεταξύ δύο απομονωμένων κόμβων [24]. Μια άλλη εφαρμογή που μπορεί να υλοποιηθεί με τη χρήση κινητών κόμβων είναι η επέκταση της περιοχής κάλυψης του ενδιαφέροντος [25] [26]. Οι αισθητήρες μπορούν επίσης να συνδέονται με μη επανδρωμένα εναέρια οχήματα (UAV) για την επιτήρηση ή τη χαρτογράφηση του περιβάλλοντος [23].

2.3.2 Μορφές Κινητικότητας στα ασύρματα δίκτυα αισθητήρων

Η κινητικότητα στα ασύρματα δίκτυα αισθητήρων μπορεί να παρουσιαστεί σε τρεις μορφές [17]:

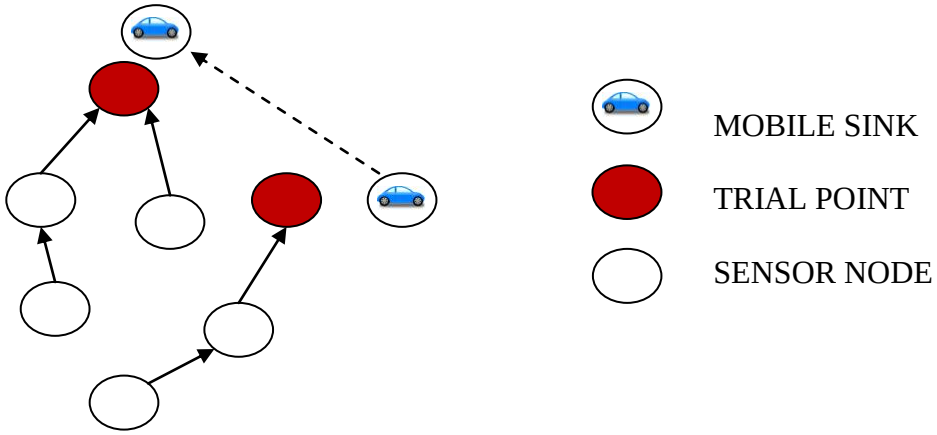
1. Κινητικότητα του κόμβου (Node mobility):

Σε αυτή τη μορφή κινητικότητας οι ασύρματοι κόμβοι αισθητήρων είναι καθεαυτό κινητοί. Τα πλεονεκτήματα της χρήσης αυτού του είδους της κινητικότητας εξαρτούνται σε μεγάλο βαθμό από την εφαρμογή. Για παράδειγμα σε εφαρμογές που ασχολούνται με τον έλεγχο του περιβάλλοντος, η κινητικότητα των κόμβων δεν πρέπει να χρησιμοποιείται, ενώ σε εφαρμογές που ασχολούνται με την επιτήρηση της κτηνοτροφίας (κόμβοι που συνδέονται με τα βοοειδή, για παράδειγμα), είναι απαραίτητο να χρησιμοποιείται αυτή η μορφή κινητικότητας. Τα δίκτυα που αποτελούνται μόνο από κινητούς κόμβους αισθητήρων πρέπει να αναδιοργανώνονται αρκετά συχνά για να μπορέσουν να λειτουργούν σωστά.

Ένα απλό παράδειγμα αυτής της μορφής κίνησης είναι χρησιμοποιώντας λιγότερους κινητούς κόμβους αισθητήρων, με κάποιες προκαθορισμένες τροχιές, για να καλύψουμε ένα χώρο.
Στην παρούσα διπλωματική εργασία χρησιμοποιείται αυτή η μορφή κινητικότητας αλλά μόνο οι κόμβοι που τοποθετούνται από τον αλγόριθμο (οι οποίοι αρχικά βρίσκονται σε sleep state, δηλαδή δεν είναι ενεργοί και είναι τοποθετημένοι κοντά στον κόμβο προορισμού) έχουν αυτήν την ιδιότητα, όλοι οι υπόλοιποι κόμβοι αισθητήρων είναι στατικοί.

2. Κινητικότητα του κόμβου προορισμού (Sink mobility):

Σε αυτή τη μορφή κινητικότητας μόνο οι κόμβοι προορισμού είναι κινητοί. Με αυτό τον τρόπο εξοικονομείται ενέργεια εφόσον δεν πρέπει να αποστέλλονται μηνύματα σε μακρινές αποστάσεις. Κάθε κόμβος στέλνει το μήνυμα του μόνο όταν φτάσει ο κόμβος προορισμού κοντά του, όπως φαίνεται στο Σχήμα 2.3. Στις απλές περιπτώσεις ο κόμβος προορισμού πρέπει να αλληλεπιδρά με το δίκτυο και να τελειώνει την αλληλεπίδραση του προτού να προχωρήσει. Ένα παράδειγμα είναι ένας χρήστης να ζητά πληροφορίες καθώς περπατά σε ένα έξυπνο κτίριο.



Σχήμα 2.3 Κινητικότητα του κόμβου προορισμού στα Ασύρματα Δίκτυα Αισθητήρων

3. Κινητικότητα του γεγονότος (Event mobility):

Η κινητικότητα του γεγονότος εμφανίζεται σε ένα δίκτυο ασύρματων αισθητήρων όταν μετακινείται το “πρόβλημα” που παρακολουθείται. Αυτή η μορφή κινητικότητας χρησιμοποιείται σε εφαρμογές όπως η ανίχνευση γεγονότος και γενικά σε εφαρμογές παρακολούθησης, όπου μετακινείται η αιτία που προκαλεί το γεγονός ή το αντικείμενο που παρακολουθείται. Σε τέτοιου είδους σενάρια, είναι σημαντικό να παρακολουθείται συνεχώς το γεγονός ή το αντικείμενο που παρακολουθείται, από ένα ικανοποιητικό αριθμό κόμβων αισθητήρων και να ενημερώνεται ο κόμβος προορισμού με τις σχετικές πληροφορίες. Οι κόμβοι αισθητήρων γύρω από το αντικείμενο ενεργοποιούνται για να το παρατηρήσουν. Ακολούθως, αφού τελειώσουν, αλλάζουν την κατάσταση τους από ενεργή σε κατάσταση ύπνου, ώστε να μην σπαταλείται η ενέργεια τους. Αυτό το μοντέλο όπου οι κόμβοι που δημιουργούν δεδομένα (πηγαίοι κόμβοι) μετακινούνται μέσα στο δίκτυο και συνεπώς μετακινείται η περιοχή των κόμβων που είναι ενεργοί, ονομάστηκε frisbee model [18].

Ένα χαρακτηριστικό παράδειγμα στο οποίο χρησιμοποιείται αυτή η μορφή κινητικότητας είναι η συνεχής παρακολούθηση ενός ζώου σε μια μεγάλη περιοχή που καλύπτεται από πάρα πολλούς κόμβους αισθητήρων. Το ζώο σε

οποιαδήποτε χρονική στιγμή μπορεί να κινηθεί και να βρεθεί σε άλλο σημείο, όμως δεν μπορεί να βρίσκεται σε ολόκληρη την περιοχή την ίδια χρονική στιγμή. Για αυτό ενεργοί είναι μόνο οι κόμβοι που βρίσκονται κοντά στο ζώο, ενώ όλοι οι υπόλοιποι είναι σε κατάσταση ύπνωσης (sleep state) και έτσι δεν σπαταλείται η ενέργεια τους.

2.4 Παραδοχές

Στην παρούσα εργασία γίνονται οι πιο κάτω παραδοχές:

1. Όλοι οι κόμβοι αισθητήρων (στατικοί και κινητοί) είναι πανομοιότυποι δηλαδή έχουν τα ίδια χαρακτηριστικά, όπως ακτίνα αίσθησης (sensing radius), ενέργεια κλπ, με τη διαφορά ότι οι κινητοί αισθητήρες έχουν επίσης τη δυνατότητα να κινούνται.
2. Όλοι οι κόμβοι χρησιμοποιούν το “carrier sense multiple access with collision avoidance” [CSMA/CA] ως το MAC πρωτόκολλο.
3. Όλοι οι κόμβοι αισθητήρων ξέρουν την ακριβή τοποθεσία τους (π.χ. μέσω GPS συστήματος ή μέσω κάποιου πρωτοκόλλου εντοπισμού τοποθεσίας) και μεταδίδουν αυτή τη γνώση στον κόμβο προορισμού.
4. Θεωρούμε ότι οι κινητοί κόμβοι κινούνται με μεγάλη ταχύτητα, ώστε ο χρόνος να μετακινηθεί ένας κινητός κόμβος στη θέση που του καθορίζεται να είναι πολύ μικρός.
5. Θεωρούμε ότι οι κινητοί κόμβοι δεν σπαταλούν επιπλέον ενέργεια όταν κινούνται.

2.5 Προηγούμενη Εργασία

Congestion Control in Wireless Sensor Networks Through Dynamic Alternative Path Selection [1]:

Σε αυτό το άρθρο παρουσιάζεται ένα σύστημα για έλεγχο και αποφυγή της συμφόρησης στα ασύρματα δίκτυα αισθητήρων που ονομάζεται Dynamic Alternative Path Selection Scheme (DAlPaS). Ο DAlPaS είναι ένα πολύ απλό αλλά αποτελεσματικό σύστημα, το οποίο ελέγχει τη συμφόρηση καθώς διατηρεί

επίσης το overhead στο ελάχιστο. Η λειτουργία αυτού του συστήματος βασίζεται στον έλεγχο των πόρων (resource control method) μέσω εναλλακτικών μονοπατιών, αντί στον έλεγχο του ρυθμού αποστολής δεδομένων στην πηγή (traffic control method).

Οι παράμετροι που εξετάζονται κατά την επιλογή μιας εναλλακτικής διαδρομής σε πυκνές τοπολογίες σε περίπτωση συμφόρησης περιλαμβάνουν (α) την εναπομείναντα ενέργεια στον κόμβο και στο δίκτυο, (β) τη συμφόρηση, από την άποψη της διαθεσιμότητας του buffer και τις παρεμβολές, (γ) τη χρονική καθυστέρηση για τη μετάδοση δεδομένων από ένα πηγαίο κόμβο σε ένα κόμβο προορισμού, και (δ) το ρυθμό απώλειας πακέτων στο δίκτυο.

Ο αλγόριθμος DAIPaS αποτελείται από τις ακόλουθες φάσεις και συστήματα: φάση εγκατάστασης, σύστημα ελέγχου τοπολογίας, σύστημα απαλής φάσης και σύστημα δύσκολης φάσης. Η φάση εγκατάστασης εκτελείται μόνο μία φορά, κατά την προετοιμασία του δικτύου. Σε αυτή τη φάση, οι κόμβοι ανακαλύπτουν ο ένας τον άλλο και κτίζουν τους πίνακες γειτνίασης τους, όπου κρατούν πληροφορίες για κάθε κόμβο του δικτύου που μπορούν να επικοινωνήσουν αμφίδρομα. Πιο συγκεκριμένα αυτή η φάση ξεκινά από τον κόμβο προορισμού ο οποίος στέλνει ένα μήνυμα με το επίπεδο του ίσο με μηδέν σε όλους τους γείτονες του. Το επίπεδο δείχνει την απόσταση ενός κόμβου σε βήματα (hops) από τον κόμβο προορισμού. Κάθε κόμβος βρίσκει στον πίνακα γειτνίασης του τον κόμβο με το μικρότερο επίπεδο, αυξάνει αυτό το επίπεδο κατά ένα και στέλνει μήνυμα στους γείτονες του με το επίπεδο του. Όσον αφορά το σύστημα ελέγχου τοπολογίας, ο αλγόριθμος DAIPaS χρησιμοποιεί ένα δυναμικό τρόπο για τον έλεγχο τοπολογίας που δεν προσθέτει επιπλέον overhead στο δίκτυο, χρησιμοποιώντας δεδομένα από τους πίνακες γειτνίασης που έχουν δημιουργηθεί κατά τη φάση εγκατάστασης. Έτσι, κάθε κόμβος που πρόκειται να μεταδώσει δεδομένα αναζητεί στον πίνακα γειτνίασης του και βρίσκει τον κόμβο με το μικρότερο επίπεδο, (δηλαδή αυτόν που είναι πιο κοντά στον κόμβο προορισμού), και μεταδίδει τα δεδομένα μέσω του συγκεκριμένου κόμβου.

Στο σύστημα απαλής φάσης (soft stage) εισέρχεται κάποιος κόμβος όταν λαμβάνει πακέτα από περισσότερες από μία ροές κι έτσι ο κόμβος προσπαθεί να σταματήσει να λαμβάνει δεδομένα από περισσότερες από μία ροές. Για λόγους απόδοσης, κάθε κόμβος επιλέγει να συνεχίσει να εξυπηρετεί τη ροή με τον υψηλότερο ρυθμό μετάδοσης δεδομένων, και ενημερώνει τις υπόλοιπες ροές για να ψάξουν για εναλλακτικό μονοπάτι. Ο κόμβος που λαμβάνει το μήνυμα αυτό ελέγχει για ένα εναλλακτικό μονοπάτι. Ένας κόμβος εισέρχεται στο σύστημα δύσκολης φάσης (hard stage) εάν δεν μπορεί να συνεχίσει να εξυπηρετεί τους κόμβους που του στέλνουν δεδομένα, λόγω μικρής εναπομείναντας μπαταρίας ή μη διαθεσιμότητας του buffer ή μη διαθεσιμότητας κόμβων σε μικρότερο επίπεδο για να προωθεί τα πακέτα που λαμβάνει. Ο κόμβος που εισέρχεται σε αυτό το σύστημα ενημερώνει τους γείτονες του, έτσι ώστε να σταματήσουν να στέλνουν τα δεδομένα τους σε αυτόν τον κόμβο. Επίσης ενεργοποιείται ο αλγόριθμος δημιουργίας εναλλακτικής διαδρομής ο οποίος επιλέγει τον επόμενο κόμβο για να μεταδίδει τα δεδομένα του ο κόμβος που βρίσκεται στο σύστημα δύσκολης φάσης, με βάση τη διαθεσιμότητα του και τον αριθμό των βημάτων (hops) για να φθάσει στον κόμβο προορισμού. Ταξινομεί μόνο τους διαθέσιμους κόμβους σε αύξουσα σειρά με βάση τον αριθμό των βημάτων (hops) από τον κόμβο προορισμού και προωθεί τα πακέτα στον κόμβο που χρειάζεται τα λιγότερα βήματα (hops) για να φθάσει στον κόμβο προορισμού. Εάν ο πρώτος κόμβος καθίσταται μη διαθέσιμος για να μεταδίδει, επιλέγει τον επόμενο κόμβο στον πίνακα.

Σε αυτό το άρθρο γίνονται οι εξής υποθέσεις: υποθέτει ότι το δίκτυο είναι πυκνά ανεπτυγμένο και οι κόμβοι έχουν τοποθετηθεί σε ένα ομοιόμορφα τυχαίο τρόπο. Υπάρχει μόνο μια πηγή στην τοπολογία του δικτύου, ενώ ο αριθμός των πηγαίων κόμβων είναι μεταβλητός. Υποθέτει επίσης ότι κάθε κόμβος γνωρίζει τη θέση του και τη θέση της πηγής. Όλοι οι κόμβοι εκτός από την πηγή είναι πανομοιότυποι και χρησιμοποιείται το CSMA ως το MAC πρωτόκολλο. Η απόδοση του DAlPaS έχει αξιολογηθεί σε σχέση με συγκρίσιμα συστήματα με καλά αποτελέσματα.

Mobile CC: Introduction Mobility to WSNs for Congestion Mitigation in Heavily Congested Areas [9]**:**

Το πιο πάνω άρθρο λύνει το πρόβλημα της συμφόρησης στα Ασύρματα Δίκτυα Αισθητήρων (Wireless Sensor Networks) με τη χρήση κινητών κόμβων αισθητήρων. Παρουσιάζει τα πλεονεκτήματα της δημιουργίας εναλλακτικών μονοπατιών με την τοποθέτηση κινητών κόμβων αισθητήρων, μεταξύ της περιοχής των κόμβων που παρουσιάζουν συμφόρηση και του κόμβου προορισμού. Εφαρμόζεται σε περιοχές όπου η συμφόρηση είναι μόνιμη ή είναι μεγάλης διάρκειας ή επαναλαμβάνεται. Δεν υπάρχει μεγάλο όφελος αν η συμφόρηση είναι για πολύ μικρό διάστημα, εκτός αν υπάρχει μεγάλη πιθανότητα να συμβεί ξανά στην ίδια τοποθεσία στο μέλλον. Ο αλγόριθμος Mobile CC δεν αντικαταστεί τους υπάρχων αλγόριθμους ελέγχου συμφόρησης ή τους αλγόριθμους δρομολόγησης, αλλά τρέχει παράλληλα με αυτούς.

Ο αλγόριθμος αποτελείται από τέσσερις μηχανισμούς. Ο μηχανισμός ανίχνευσης συμφόρησης ελέγχει την χωρητικότητα του buffer κάθε κόμβου και αν αυτή είναι μικρότερη από κάποιο προκαθορισμένο threshold τότε ελέγχεται ο ρυθμός δημιουργίας δεδομένων και ο ρυθμός με τον οποίο λαμβάνει δεδομένα από κάθε ένα από τους γείτονες του. Αν μετά από κάποιο χρονικό διάστημα η χωρητικότητα του buffer και το συνολικό data rate εξακολουθούν να μην είναι κατάλληλα τότε θεωρείται ότι ο κόμβος παρουσιάζει συμφόρηση και ο αλγόριθμος συνεχίζει με τον μηχανισμό ενημέρωσης συμφόρησης, αλλιώς αν το buffer occupancy και το total data rate είναι μικρότερα από κάποια thresholds σταματά ο έλεγχος του data rate. Ο μηχανισμός επιλογής κόμβου που παρουσιάζει συμφόρηση είναι υπεύθυνος επίσης για την επιλογή των κόμβων που θα συνεχίσει να εξυπηρετεί ο κόμβος που παρουσιάζει συμφόρηση, λαμβάνοντας υπόψη το ρυθμό με τον οποίο στέλνουν πληροφορίες σε αυτόν, την ποιότητα της σύνδεσης μεταξύ του κόμβου που παρουσιάζει συμφόρηση και του κόμβου αποστολέα και το χρόνο που οι δυο κόμβοι είναι ενωμένοι. Όταν ολοκληρωθεί αυτή η διαδικασία ο μηχανισμός ενημέρωσης συμφόρησης αποστέλλει τις πληροφορίες για τη συμφόρηση στον κόμβο προορισμού. Τέλος ο μηχανισμός αντίδρασης αποφασίζει τον αριθμό των μονοπατιών, που αποτελούνται μόνο από κινητούς κόμβους αισθητήρων, που πρέπει να κτιστούν

καθώς επίσης τον αριθμό και τις τοποθεσίες των κινητών κόμβων αισθητήρων και ενημερώνει τους κινητούς κόμβους αισθητήρων να μετακινηθούν στις κατάλληλες θέσεις.

Σε αυτό το άρθρο γίνεται η υπόθεση ότι τόσο οι στατικοί κόμβοι αισθητήρων όσο και οι κινητοί κόμβοι αισθητήρων ξέρουν τη θέση τους και όλοι οι κόμβοι μαθαίνουν τη θέση του κόμβου προορισμού κατά την αρχικοποίηση του δικτύου. Επίσης γίνεται η υπόθεση ότι κάθε κόμβος μπορεί να λαμβάνει πακέτα από πολλούς κόμβους αλλά στέλνει μόνο σε ένα κόμβο, για να μην δημιουργούνται βρόγχοι. Έχει αποδειχτεί, μέσω προσομοιώσεων στον προσομοιωτή COOJA, ότι αν οι κινητοί κόμβοι τοποθετηθούν σωστά είναι δυνατό να μειωθούν οι επιπτώσεις της συμφόρησης. Ο αλγόριθμος που αναπτύσσεται προσπαθεί να λύσει το πρόβλημα της εύρεσης του κατάλληλου αριθμού κινητών κόμβων αισθητήρων και της κατάλληλης τοποθεσίας για κάθε ένα από αυτούς.

Repairing Sensor Network Using Mobile Robots [10]:
Αυτό το άρθρο ασχολείται με το πρόβλημα της αντικατάστασης των κόμβων που δεν μπορούν να συνεχίσουν να λειτουργούν (failed nodes) με χρήση ενός μικρού αριθμού κινητών ρομπότ για μεγάλης κλίμακας στατικά δίκτυα αισθητήρων. Περιγράφονται τρεις αλγόριθμοι, ο κεντρικοποιημένος, ο σταθερά κατανεμημένος και ο δυναμικά κατανεμημένος αλγόριθμος, οι οποίοι ασχολούνται με την ανίχνευση και την αναφορά των αποτυχιών των κόμβων αισθητήρων και το συντονισμό της κίνησης των ρομπότ ώστε να ελαχιστοποιείται η ενέργεια που χρησιμοποιείται και τα μηνύματα που ανταλλάσσονται στο δίκτυο. Ένα ρομπότ μπορεί να είναι μάνατζερ ή/και συντηρητής. Μάνατζερ ονομάζεται το ρομπότ που λαμβάνει αναφορές αποτυχιών κόμβων και καθορίζει ποιο ρομπότ θα διαχειριστεί την αποτυχία. Συντηρητής ονομάζεται το ρομπότ που μεταφέρει και αντικαταστεί τους κόμβους αισθητήρων που δεν μπορούν να συνεχίσουν να λειτουργούν.

Στον κεντρικοποιημένο αλγόριθμο υπάρχει μόνο ένας μάνατζερ στο κέντρο και όλες οι αποτυχίες αναφέρονται σε αυτό, έτσι είναι υπεύθυνος να τις προωθεί σε

διαφορετικούς ρομπότ-συντηρητές. Στον σταθερά κατανεμημένο αλγόριθμο η περιοχή που καλύπτει το δίκτυο χωρίζεται σε υπό-περιοχές ίσου μεγέθους και κάθε ρομπότ είναι υπεύθυνο για μια από αυτές. Στο δυναμικά κατανεμημένο αλγόριθμο δεν υπάρχουν όρια μεταξύ δυο περιοχών χρησιμοποιούνται Voronoi graphs και όταν αλλάζει θέση το ρομπότ προσαρμόζεται ανάλογα και το voronoi graph, ώστε κάθε αισθητήρας που βρίσκεται σε μια περιοχή να είναι πιο κοντά στο ρομπότ της περιοχής του παρά σε οποιοδήποτε άλλο ρομπότ. Στους κατανεμημένους αλγόριθμους τα ρομπότ έχουν τον ρόλο τόσο του μάνατζερ όσο και του συντηρητή. Μέσω των προσομοιώσεων έχει αποδειχτεί ότι ο κεντρικοποιημένος αλγόριθμος και ο δυναμικά κατανεμημένος αλγόριθμος χρειάζονται λιγότερο κόστος για την μετακίνηση των ρομπότ σε σχέση με τον σταθερά κατανεμημένο αλγόριθμο και επίσης ότι οι δυο κατανεμημένοι αλγόριθμοι έχουν ψηλότερο κόστος ανταλλαγής μηνυμάτων.

Extending k-Coverage Lifetime of Wireless Sensor Networks Using Mobile Sensor Nodes [11]:
Αυτό το άρθρο προτείνει μια μέθοδο για τη λύση του προβλήματος της k-κάλυψης του πεδίου αίσθησης και της επέκτασης του χρόνου ζωής του δικτύου, μετακινώντας κινητούς κόμβους αισθητήρων στις κατάλληλες θέσεις. Ένα πεδίο είναι k-καλυμμένο αν κάθε σημείο στο πεδίο περιλαμβάνεται στην ακτίνα αίσθησης τουλάχιστον k κόμβων αισθητήρων. Το πρόβλημα είναι NP-δύσκολο για αυτό χρησιμοποιείται ένας αλγόριθμος βασισμένος στο σύστημα των γενετικών αλγορίθμων για την εύρεση μιας σχεδόν βέλτιστης λύσης σε μικρό χρονικό διάστημα. Για την επιτάχυνση των υπολογισμών, επινοήθηκε μια ευρετική μέθοδος που χωρίζεται σε δυο βήματα.

Το πρώτο βήμα του αλγορίθμου είναι η επίλυση του προβλήματος της εύρεσης των θέσεων των κινητών κόμβων και το δέντρο συλλογής δεδομένων για τη μεγιστοποίηση του χρόνου ζωής του δικτύου. Και το δεύτερο βήμα καθορίζει ότι σε περίπτωση που η μπαταρία κάποιου κόμβου σύντομα θα εξαντληθεί, επαναλαμβάνεται το πρώτο βήμα. Για τη δημιουργία των υποψήφιων λύσεων χρησιμοποιούνται τυχαίες μεταβλητές. Οι γωνίες και οι αποστάσεις των κινητών κόμβων είναι ομοιόμορφα κατανεμημένες τυχαίες τιμές μεταξύ 0 και 2π, και 0

και dist, όπου η σταθερά dist είναι η μεγαλύτερη απόσταση που μπορεί να κινηθεί ένας κόμβος στο πεδίο. Για κάθε κόμβο επιλέγεται τυχαία, ένας κόμβος γεωγραφικά πιο κοντά στον κόμβο προορισμού, ως πατέρας του. Επιλέγονται οι τρεις υποψήφιες λύσεις με το μικρότερο κόστος δέντρου συλλογής. Το κόστος δέντρου συλλογής δημιουργείται με την μέθοδο του minimum cost spanning tree με κόστος την απόσταση. Οι τρεις αυτές λύσεις προσθέτονται στις αρχικές υποψήφιες λύσεις. Οι υποψήφιες λύσεις αξιολογούνται ανάλογα με το χρονικό διάστημα που θα κρατήσουν το πεδίο k-καλυμμένο.

Σε αυτό το άρθρο γίνεται η υπόθεση ότι κάθε κινητός και στατικός αισθητήρας ξέρει τη θέση του και ενημερώνει τον κόμβο προορισμού για την θέση του. Επίσης γίνεται η υπόθεση ότι το μέγεθος των δεδομένων που λαμβάνει κάθε αισθητήρας, κάθε χρονική στιγμή είναι σταθερό και τα δεδομένα αποστέλλονται στην πηγή χωρίς συμπίεση ή ενοποίηση. Οι προσομοιώσεις απέδειξαν ότι με αυτή τη μέθοδο επιτυγχάνεται μεγαλύτερο διάστημα k-κάλυψης από άλλες μεθόδους για δίκτυα με 100 μέχρι 300 κόμβους

Extending k-Coverage Lifetime of Wireless Sensor Networks with Surplus Nodes [12]:
Αυτό το άρθρο περιγράφει μια μέθοδο που μεγιστοποιεί τη διάρκεια ζωής του ασύρματου δικτύου αισθητήρων, με τυχαία τοποθέτηση επιπλέον κόμβων στο πεδίο. Σε αυτό το άρθρο γίνεται η υπόθεση ότι κάθε αισθητήρας έχει τρεις τρόπους λειτουργίας: αίσθηση, αναμετάδοση και ύπνου. Κάθε κόμβος στη λειτουργία της αίσθησης ανιχνεύει δεδομένα του περιβάλλοντος και στέλνει / αναμεταδίδει τα δεδομένα στην πηγή μέσω multi-hop επικοινωνία. Κάθε κόμβος στη λειτουργία της αναμετάδοσης απλά μεταδίδει τα δεδομένα που λαμβάνει από το uplink κόμβο του στους downlink κόμβους του. Κάθε κόμβος στη λειτουργία της ύπνωσης δεν κάνει τίποτα και κρατά την μπαταρία του.

Ο αλγόριθμος που περιγράφεται αλλάζει δυναμικά την λειτουργία του κάθε κόμβου αισθητήρα έτσι ώστε η διάρκεια ζωής του δικτύου να γίνεται όσο το δυνατόν πιο μεγάλη με την αλλαγή του μικρότερου αριθμού κόμβων για επίτευξη k-κάλυψης του πεδίου των κόμβων αισθητήρων σε λειτουργία

αίσθησης. Αρχικά όλοι οι κόμβοι είναι στη λειτουργία του ύπνου. Επιλέγεται ο κόμβος που έχει τη μεγαλύτερη περιοχή που δεν είναι k-καλυμμένη, αλλά περιλαμβάνεται στο εύρος ανίχνευσης του, να αλλάξει τη λειτουργία του σε λειτουργία αίσθησης. Επαναλαμβάνεται αυτή η διαδικασία μέχρι το πεδίο να είναι k-καλυμμένο ή να μην υπάρχουν άλλοι κόμβοι αισθητήρων σε λειτουργία ύπνου. Μέσω προσομοιώσεων επιβεβαιώθηκε ότι η μέθοδός αυτή μπορεί να παρατείνει τη διάρκεια ζωής του δικτύου σχεδόν ανάλογα με τον αριθμό των κόμβων που χρησιμοποιούνται.

Mitigate Funnel Effect in Sensor Networks with Multi-Interface Relay Nodes [13]:
Παρουσιάζεται ένας αλγόριθμος που διαχειρίζεται τη γεωγραφική τοποθέτηση κόμβων αναμετάδοσης στην περιοχή όπου παρουσιάζεται το funnel effect, λόγω των υπερβολικών ροών δεδομένων. Με αυτόν τον τρόπο ελαχιστοποιούνται οι επιπτώσεις που θα παρουσιαστούν στο δίκτυο λόγω του funnel effect.

Ο αλγόριθμος που παρουσιάζεται έχει πολυπλοκότητας O(mlog(h)) όπου m είναι το μέγεθος της περιοχής που παρουσιάζει συμφόρηση και h το μέγεθος του υπολογιζόμενου κυρτού σχήματος. Το κυρτό σχήμα βρίσκει την τοποθέτηση του ελάχιστου αριθμού κόμβων αναμετάδοσης που να καλύπτουν την περιοχή που παρουσιάζει συμφόρηση. Σε άπληστο τρόπο τοποθετείται ένας κόμβος αναμετάδοσης στην κοντινότερη θέση έτσι ώστε να καλύπτει το μεγαλύτερο αριθμό κόμβων, μέχρι μια παράμετρο που ορίζεται με βάση το εύρος μετάδοσης του κόμβου και τη θέση του σε σχέση με τον κόμβο προορισμού. Πιο συγκεκριμένα, ο αλγόριθμος αρχίζει με τον υπολογισμό του κυρτού σχήματος δυο διαστάσεων που περιλαμβάνει όλους τους κόμβους στη περιοχή της συμφόρησης, το οποίο ταξινομείται σε δεξιόστροφα τοπολογία δακτυλίου. Ακολούθως ξεκινώντας από τον πιο αριστερό κόμβο επισκέπτεται κάθε κόμβο e που δεν έχει καλυφθεί. Αν ο e μπορεί να επικοινωνήσει με τον κόμβο προορισμού, θεωρείται καλυμμένος και ο βρόγχος συνεχίζεται, αλλιώς βρίσκει την καλύτερη τοποθεσία έστω q για την τοποθέτηση κόμβου μεταξύ του e και της πηγής και ο e θεωρείται καλυμμένος. Μετά επισκέπτεται κάθε κόμβο e΄ μετά

τον e που δεν είναι καλυμμένος και αν ο e΄ δεν είναι στο εύρος μετάδοσης του e σταματά, αλλιώς βρίσκει την καλύτερη τοποθεσία έστω p για να καλύπτει και τους δυο κόμβους ο κόμβος που θα τοποθετηθεί και να είναι όσο γίνεται πιο κοντά στον κόμβο προορισμού, δημιουργώντας ένα τρίγωνο μεταξύ των δυο κόμβων και του κόμβου προορισμού και κόβοντας το με βάση το εύρος μετάδοσης. Τέλος επιστρέφεται η καλύτερη τοποθεσία μεταξύ της p και της q.

Τα αποτελέσματα των προσομοιώσεων έχουν δείξει ότι χρησιμοποιώντας τον ελάχιστο αριθμό κόμβων αναμετάδοσης, μπορεί να αποθηκευτεί έως το 43% των κόμβων συγκρίνοντας το με απλές στρατηγικές τοποθέτησης, το δίκτυο αυξάνει τον ρυθμό απόδοσης του (throughput) και έχει τη δυνατότητα της εξισορρόπησης του φορτίου.

Deploying mobile nodes to connect wireless sensor networks using novel algorithms [14]:
Στο πιο πάνω άρθρο προτείνεται η χρήση κινητών κόμβων αισθητήρων στα ασύρματα δίκτυα αισθητήρων για την επανασύνδεση του δικτύου. Προτείνονται δύο νέοι αλγόριθμοι: ένας προσανατολισμένος από γράφημα αλγόριθμος (graph-oriented) και ένας αλγόριθμος διαίρει και βασίλευε (divide-and-conquer) για να συνδεθούν τα αποσυνδεδεμένα δίκτυα, χρησιμοποιώντας όσο το δυνατόν λιγότερους κινητούς κόμβους.

Ο πρώτος αλγόριθμος εκμεταλλεύεται παραδοσιακές γραφικές και γεωμετρικές τεχνικές συμπεριλαμβανομένων των εξής: το σημείο Fermat, το κυρτό σχήμα, το πλησιέστερο γείτονα, το minimum cost spanning tree και τη συρρίκνωση του γραφήματος. Υιοθετώντας μια εντελώς διαφορετική προσέγγιση, ο δεύτερος αλγόριθμος επιλύει το πρόβλημα με τη διαίρεση της περιοχής και τη συγχώνευση των υπολύσεων αναδρομικά. Όσον αφορά το θέμα της πολυπλοκότητας, ο αλγόριθμος προσανατολισμένος από το γράφημα παίρνει $O(n^3)$ χρόνο, ενώ ο αλγόριθμος διαίρει και βασίλευε απαιτεί O(n log n) χρόνο, όπου n είναι ο αριθμός των κορυφών του γραφήματος, συνεπώς μπορούν να εφαρμοστούν σε ένα κεντρικοποιημένο δίκτυο αισθητήρων.

Σε αυτό το άρθρο γίνεται η υπόθεση ότι κάθε αισθητήρας είναι εξοπλισμένος με μια συσκευή GPS και γνωρίζει πάντα τη θέση του. Επίσης γίνεται η υπόθεση ότι όλοι οι κόμβοι στο ίδιο συνδεδεμένο δίκτυο μπορούν να βρουν ο ένας τον άλλο με στοιχειώδη τρόπο μετάδοσης και ότι σε κάθε υποδίκτυο υπάρχει τουλάχιστον ένας κόμβος που ονομάζεται κεφαλή του δικτύου, ο οποίος μπορεί να μεταδώσει πληροφορίες για τη θέσης του κάθε κόμβου στο συνδεδεμένο υποδίκτυο σε κάποιο καθορισμένο διακομιστή (server). Μετά τον υπολογισμό και την ανάλυση των πληροφοριών που αφορούν τις θέσεις των κόμβων του δικτύου, ο διακομιστής θα καλέσει τους κινητούς κόμβους για να μετακινηθούν στις κατάλληλες θέσεις για να συνδεθούν τα αποσυνδεδεμένα υπο-δίκτυα.

Ο αλγόριθμος προσανατολισμένος από το γράφημα είναι πιο περίπλοκος από τον αλγόριθμο διαιρεί-και-βασίλευε, αλλά απαιτεί λιγότερους κινητούς κόμβους, ιδίως όταν η πυκνότητα της τοπολογίας των κόμβων του δικτύου είναι αραιή.

Congestion Avoidance and Energy Efficient Routing Protocol for Wireless Sensor Networks with a Mobile Sink [15]:
Αυτό το άρθρο εισάγει ένα κινητό κόμβο προορισμού στο σύστημα δρομολόγησης των ασύρματων δικτύων αισθητήρων με στόχο την αποφυγή της συμφόρησης και την αποδοτική χρήση της ενέργειας των κόμβων. Το προτεινόμενο σύστημα χρησιμοποιεί ένα κινητό κόμβο προορισμού και ένα μοντέλο αποθήκευσης των δεδομένων στο δίκτυο χρησιμοποιώντας πολλούς στατικούς κόμβους που έχουν το ρόλο του τοπικού κόμβου προορισμού (mini-sink), κατά μήκος της τροχιάς της κινητικότητας του κόμβου προορισμού.

Στο σύστημα οι κόμβοι mini-sink είναι υπεύθυνοι για τη συλλογή δεδομένων από τους κόμβους που βρίσκονται στην περιοχή τους, αποφεύγοντας έτσι τα δεδομένα να ρέουν προς ένα μόνο σημείο συλλογής δεδομένων, δηλαδή ένα μόνο στατικό κόμβο προορισμού, που είναι η κύρια αιτία της συμφόρησης, της απώλειας δεδομένων και της μειωμένης διάρκειας ζωής των δικτύων αισθητήρων. Επίσης, στον προτεινόμενο αλγόριθμο τα δεδομένα έχουν να ταξιδέψουν μόνο περιορισμένο αριθμό βημάτων (hops) για να φθάσουν στο

πλησιέστερο κόμβο mini- sink, το οποίο βοηθά στη βελτίωση της κατανάλωσης ενέργειας των κόμβων αισθητήρων.

Σε αυτό το άρθρο γίνεται η υπόθεση ότι οι κόμβοι αισθητήρων είναι ομοιόμορφα αλλά τυχαία κατανεμημένοι σε ένα απομακρυσμένο πεδίο σε πυκνούς αριθμούς. Μέσω προσομοιώσεων έχει αποδειχτεί η αποτελεσματικότητα του συγκεκριμένου συστήματος δρομολόγησης όσο αφορά την αποφυγή της συμφόρησης και την αυξημένη διάρκεια ζωής του δικτύου.

Design and Implementation of Mobile Robot for Nodes Replacement in Wireless Sensor Networks [16]:
Σε αυτό το άρθρο περιγράφεται η σχεδίαση ενός έξυπνου κινητού ρομπότ και ένας αλγόριθμος αντικατάστασης των κόμβων με χαμηλή ενέργεια και αποδεικνύονται τα πλεονεκτήματα της χρήσης των κινητών ρομπότ. Τα κινητά ρομπότ μπορούν να πλοηγηθούν προς τους κόμβους αισθητήρων που έχουν χαμηλή εναπομείναντα ενέργεια και να τους αντικαταστήσουν με νέους κόμβους αισθητήρων.

Ο αλγόριθμος πλοήγησης βασίζεται στο λαμβανόμενο σήμα (RSSI) μεταξύ του κινητού ρομπότ και του κόμβου επικοινωνίας, δεδομένου ότι δεν υπάρχουν πληροφορίες τοποθεσίας στους στατικούς κόμβους αισθητήρων, τα κινητά ρομπότ δεν γνωρίζουμε τη θέση του κόμβου που πρέπει να φθάσουν. Κατά την αρχικοποίηση του δικτύου, ο κόμβος προορισμού μεταδίδει ένα πακέτο στο υπόλοιπο δίκτυο. Όταν ένας αισθητήρας λάβει ένα τέτοιο πακέτο καταγράφει το αναγνωριστικό (ID) του αποστολέα, ο οποίος είναι το επόμενο hop προς την πηγή. Εάν ένας αισθητήρας λάβει αρκετά τέτοια πακέτα από διαφορετικούς κόμβους, κρατάει το αναγνωριστικό του αποστολέα που έχει το μικρότερο αριθμό βημάτων (hops) για να φθάσει στον κόμβο προορισμού και αναμεταδίδει το πακέτο. Μετά τη δημιουργία των διαδρομών δρομολόγησης, κάθε κόμβος μπορεί να στείλει ένα «πακέτο βοήθειας» στον κόμβο προορισμού για να ζητήσει την αντικατάσταση του, μέσου της διαδρομής που καθορίστηκε κατά την αρχικοποίηση του δικτύου. Το «πακέτο βοήθειας» καταγράφει τα αναγνωριστικά κάθε κόμβου που πέρασε μέχρι να φθάσει στον κόμβο προορισμού. Στη

συνέχεια το κινητό ρομπότ λαμβάνει τη διαδρομή που πρέπει να ακολουθήσει από τον κόμβο προορισμού. Το ρομπότ αρχίζει να κινείται για να αντικαταστήσει τον κόμβο, ξεκινώντας από τον πρώτο κόμβο της διαδρομής, χρησιμοποιώντας την ισχύ του λαμβανόμενου σήματος (RSSI).

Τα πειραματικά αποτελέσματα επιβεβαιώνουν ότι τα κινητά ρομπότ καταφέρνουν επιτυχώς να εκτελέσουν τα καθήκοντα τους.

Κεφάλαιο 3

Ο Αλγόριθμος Dynamic MobileCC

Ο πρώτος αλγόριθμος που αναπτύχθηκε είναι ο αλγόριθμος Dynamic MobileCC, ο οποίος λύνει τοπικά το πρόβλημα τοποθετώντας ένα κινητό κόμβο να εξυπηρετεί κάποιους από τους κόμβους που στέλνουν δεδομένα στον κόμβο που παρουσιάζει συμφόρηση (congested node). Στο κεφάλαιο αυτό περιγράφεται η δομή του αλγορίθμου, παρουσιάζεται το διάγραμμα ροής δεδομένων του αλγορίθμου, και ακολούθως περιγράφεται αναλυτικά ο αλγόριθμος. Τέλος περιγράφονται μερικά παραδείγματα εκτέλεσης του.

3.1 Εισαγωγή

Ο Dynamic MobileCC είναι ένας αλγόριθμος ελέγχου συμφόρησης ο οποίος μπορεί να χρησιμοποιηθεί ως επέκταση οποιουδήποτε αλγορίθμου ελέγχου συμφόρησης. Σημειώνεται ότι ο Dynamic MobileCC δεν αντικαταστεί τους υπάρχον αλγόριθμους ελέγχου τοπολογίας ή τους αλγόριθμους ελέγχου συμφόρησης ή τους αλγόριθμους δρομολόγησης, αλλά τρέχει παράλληλα με αυτούς. Ο Dynamic MobileCC δημιουργεί εναλλακτικά μονοπάτια τοποθετώντας κινητούς κόμβους αισθητήρες, σε περίπτωση υπερφόρτωσης του δικτύου και αποτυχίας του αλγόριθμου ελέγχου συμφόρησης που χρησιμοποιείται ήδη να δημιουργήσει εναλλακτικά μονοπάτια. Ο αλγόριθμος θεωρεί ότι ένας ικανοποιητικός αριθμός κινητών αισθητήρων, ανάλογα με το μέγεθος του

δικτύου, είναι τοποθετημένοι αρχικά κοντά στο κόμβο προορισμού σε κατάσταση ύπνου (sleep state) ώστε να μην σπαταλείται η ενέργεια τους.

Ο Dynamic MobileCC λύνει δυναμικά κάθε πρόβλημα συμφόρησης, δηλαδή διαχειρίζεται μεμονωμένα κάθε θέμα συμφόρησης του δικτύου, χωρίς να λαμβάνει υπόψη του την κατάσταση των κόμβων του υπόλοιπου δικτύου και για κάθε θέμα τοποθετεί μόνο ένα κινητό κόμβο. Για παράδειγμα δεν τοποθετεί επιπλέον κινητός κόμβος όταν δεν υπάρχει κάποιος διαθέσιμος κόμβος (που να μην παρουσιάζει συμφόρηση) για να στέλνει τα δεδομένα του ο κινητός κόμβος όταν θα τοποθετηθεί στη θέση που υπολογίστηκε από τον αλγόριθμο. Με αυτόν τον τρόπο δεν γίνεται σπατάλη επιπλέον κινητών κόμβων, οι οποίοι συνήθως κοστίζουν, για τη δημιουργία μονοπατιού από τον κινητό κόμβο στον κόμβο προορισμού. Βέβαια ο αλγόριθμος πρώτα ελέγχει αν υπάρχει κάποια κατάλληλη θέση για να τοποθετηθεί ο κινητός κόμβος από την οποία θα μπορεί να στέλνει τα δεδομένα του σε κάποιο κόμβο που δεν είναι σε συμφόρηση.

Αν όμως δεν υπάρχει κάποια θέση για να τοποθετηθεί ο κινητός κόμβος στην οποία να μπορεί να στέλνει τα δεδομένα του σε κάποιο κόμβο που να μην είναι σε συμφόρηση, τότε όταν τοποθετηθεί ο κινητός κόμβος στη θέση που του καθορίζεται θα αντιληφτεί την απουσία κάποιου κόμβου που να μην βρίσκεται σε συμφόρηση για να μεταδίδει τα δεδομένα που λαμβάνει. Τότε ξαναεκτελείται ο αλγόριθμος για το νέο πρόβλημα συμφόρησης και τοποθετηθεί ένας άλλος κινητός κόμβος. Η διαδικασία αυτή συνεχίζεται μέχρι ο κινητός κόμβος που θα τοποθετηθεί να μπορεί να μεταδώσει τα δεδομένα του σε κάποιο κόμβο του δικτύου που να μην παρουσιάζει συμφόρηση ή να μπορεί να επικοινωνήσει απευθείας με τον κόμβο προορισμού. Στη δεύτερη περίπτωση το μονοπάτι που δημιουργεί ο αλγόριθμος είναι το ίδιο με το μονοπάτι που δημιουργεί ο αλγόριθμος direct path MobileCC, ο οποίος περιγράφεται στο επόμενο κεφάλαιο, αλλά η δημιουργία του παίρνει περισσότερο χρόνο λόγω της καθυστέρησης της τοποθέτησης του κινητού κόμβου και της ενημέρωσης του κόμβου προορισμού για τη νέα συμφόρηση (για την ανάγκη για την τοποθέτηση ενός επιπλέον κινητός κόμβος).

Επιπλέον λόγω αυτής της άγνοιας του αλγορίθμου για την κατάσταση των υπόλοιπων κόμβων του δικτύου, δεν εξετάζεται αν μόνο ένας κινητός κόμβος μπορεί να τοποθετηθεί σε τέτοια θέση ώστε να επιλύει δυο ή περισσότερα θέματα συμφόρησης ταυτόχρονα. Άρα σε περιπτώσεις όπου πολλά θέματα συμφόρησης μπορούν να επιλυθούν τοποθετώντας ένα μόνο κινητό κόμβο σε κατάλληλη θέση, ο Dynamic MobileCC σπαταλάει επιπλέον κινητούς κόμβους αφού όπως προαναφέρθηκε λύνει μεμονωμένα κάθε πρόβλημα συμφόρησης τοποθετώντας ένα κινητό κόμβο για το καθένα.

Ο αλγόριθμος Dynamic MobileCC προσπαθεί να διαχειριστεί θέματα συμφόρησης με όσο το δυνατόν πιο λίγη πολυπλοκότητα γίνεται, έτσι ώστε να είναι παράλληλα αποδοτικός ως προς την ενέργεια αλγορίθμου και να τοποθετούνται όσο το δυνατό γρηγορότερα οι κινητοί κόμβοι, έτσι ώστε οι συνέπειες της συμφόρησης, όπως απώλεια πακέτων, μεγάλη καθυστέρηση αποστολής πακέτων κλπ, να μην οξυνθούν.

3.2 Περιγραφή της Δομής του Αλγορίθμου

Ο αλγόριθμος Dynamic MobileCC αποτελείται από τα εξής βήματα:

1) Υπολογισμός του μέσου όρου του αριθμού των πακέτων που λαμβάνει ο κόμβος που βρίσκεται σε συμφόρηση, ανά δευτερόλεπτο, και δεν μπορεί να τα προωθήσει λόγω έλλειψης buffer.

2) Εύρεση των κόμβων αισθητήρων που στέλνουν τα δεδομένα τους στον κόμβο που βρίσκεται σε συμφόρηση.

3) Εύρεση βέλτιστης θέσης για να τοποθετηθεί ο κινητός κόμβος και εύρεση των κόμβων που θα στέλνουν τα δεδομένα τους στον κινητό κόμβο.

   3Α) Έλεγχος ύπαρξης τουλάχιστον ενός κόμβου (που στέλνει τα δεδομένα του στον κόμβο που είναι σε συμφόρηση), ο οποίος

αλλάζοντας πατέρα θα λυθεί το θέμα συμφόρησης που εξετάζεται. Αν υπάρχουν τέτοιοι κόμβοι τότε τοποθετείται ο κινητός κόμβος στη βέλτιστη θέση για να εξυπηρετεί ένα από αυτούς.

3Β) Έλεγχος ύπαρξης κοινού σημείου για κάποιο υποσύνολο των κόμβων που στέλνουν τα δεδομένα τους στον κόμβο που βρίσκεται σε συμφόρηση, το οποίο αλλάζοντας πατέρα θα λυθεί το θέμα συμφόρησης που εξετάζεται. Εύρεση βέλτιστης θέσης για να τοποθετηθεί ο κινητός κόμβος εξυπηρετώντας περισσότερους από ένα κόμβο – connectivity algorithm.

3.3 Αναλυτική Περιγραφή του Αλγορίθμου

Πιο κάτω περιγράφεται βηματικά η λειτουργία του αλγορίθμου για ένα θέμα συμφόρησης (ένα congested node). Η ίδια λειτουργία ακολουθείται για κάθε θέμα συμφόρησης.

1) Υπολογισμός επιπλέον πόρων
Αρχικά ο αλγόριθμος υπολογίζει τους επιπλέον πόρους που χρειάζεται ο κόμβος που είναι σε συμφόρηση (additional_resources), δηλαδή το μέσο όρο του αριθμού των πακέτων που λαμβάνει ανά δευτερόλεπτο ο κόμβος που είναι σε συμφόρηση, και δεν μπορεί να τα προωθήσει λόγω έλλειψης του buffer του. Το additional_resources του κόμβου i, ο οποίος είναι σε συμφόρηση, υπολογίζεται ως εξής: additional_resources[i]= ($\sum_{i=1}^{n} \text{data\_rate\_received[i]}$ - data_rate_send [congested_node]) / (current_time - time_start_send_data)

Ο κινητός κόμβος που θα τοποθετηθεί θα πρέπει να λαμβάνει και να προωθεί φορτίο όσο το additional_resources από κόμβους που στέλνουν τα δεδομένα τους μέσω του congested node και δεν μπορούν να βρουν εναλλακτική διαδρομή για την προώθηση των δεδομένων τους στον κόμβο προορισμού. Έτσι ο congested node θα σταματήσει να λαμβάνει μεγαλύτερο φορτίο από

αυτό που μπορεί να στείλει και άρα δε θα είναι πλέον σε συμφόρηση (congested).

2) Εύρεση κόμβων που στέλνουν τα δεδομένα τους στον κόμβο που βρίσκεται σε συμφόρηση

Στη συνέχεια ο αλγόριθμος βρίσκει τους γείτονες του κόμβου που βρίσκεται σε συμφόρηση (congested node) που στέλνουν τα δεδομένα τους σε αυτόν, δηλαδή τους κόμβους που ο πατέρας τους είναι ο congested node.

3) Εύρεση βέλτιστης θέσης για να τοποθετηθεί ο κινητός κόμβος και εύρεση των κόμβων που θα στέλνουν τα δεδομένα τους στον κινητό κόμβο

Η επόμενη φάση είναι η επιλογή της βέλτιστης θέσης για να τοποθετηθεί ο κινητός κόμβος και η επιλογή των κόμβων που θα σταματήσουν να στέλνουν τα δεδομένα τους στον κόμβο που βρίσκεται σε συμφόρηση (congested node) και θα στέλνουν τα δεδομένα τους στον κινητό κόμβο που θα τοποθετηθεί. Ο κύριος στόχος του αλγορίθμου είναι να αλλάξουν πατέρα όσο το δυνατό λιγότεροι κόμβοι, αλλά ταυτόχρονα οι κόμβοι αυτοί να έχουν συνολικό sending rate όσο το additional_resources. Ο δευτερεύων στόχος του είναι στη θέση που θα τοποθετηθεί ο κινητός κόμβος να υπάρχει τουλάχιστον ένας κόμβος στο εύρος αίσθησης του, ο οποίος να μην είναι σε συμφόρηση (άρα από αυτόν να υπάρχει μονοπάτι στον κόμβο προορισμού) και έτσι να μπορεί όταν τοποθετηθεί ο κινητός κόμβος να στέλνει τα δεδομένα που λαμβάνει σε αυτόν. Ο τελευταίος στόχος του αλγορίθμου είναι η θέση του κινητού κόμβου να είναι όσο το δυνατόν πιο κοντά στον κόμβο προορισμού.

3Α) Έλεγχος ύπαρξης βέλτιστης θέσης για να τοποθετηθεί ο κινητός κόμβος εξυπηρετώντας μόνο ένα κόμβο

Αρχικά εξετάζεται αν οποιοσδήποτε από τους κόμβους που στέλνουν τα δεδομένα του στον κόμβο που είναι σε συμφόρηση (congested node), τα στέλνει με sending rate μεγαλύτερο από το additional_resources. Με αυτόν τον τρόπο εξασφαλίζεται ότι πρώτα θα εξεταστεί αν υπάρχει η δυνατότητα να αλλάξει πατέρα μόνο ένας κόμβος και να αμβλυνθεί το θέμα της συμφόρησης. Αν υπάρχουν πολλοί τέτοιοι κόμβοι τότε για τον καθένα ξεχωριστά υπολογίζεται

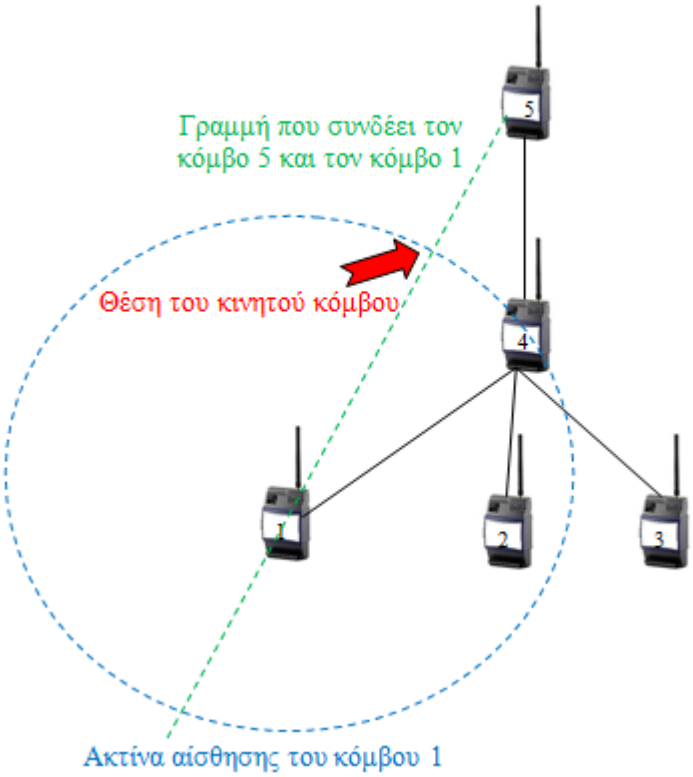
το σημείο στο οποίο θα πρέπει να τοποθετηθεί ο κινητός κόμβος για να εξυπηρετεί αυτό τον κόμβο.

Ο τρόπος υπολογισμού του σημείου στο οποίο θα πρέπει να τοποθετηθεί ο κινητός κόμβος ώστε να εξυπηρετεί τον αντίστοιχο κόμβο, άρα να είναι στο εύρος αίσθησης του, και ταυτόχρονα να είναι όσο το δυνατόν πιο κοντά στον κόμβο προορισμού είναι ο εξής:
Αρχικά υπολογίζονται τα δυο σημεία τομής του κύκλου που δημιουργείται με ακτίνα το εύρος αίσθησης του κόμβου που θα εξυπηρετείται από τον κινητό κόμβο και κέντρο τον κόμβο αυτόν, και της ευθείας που ενώνει τον κόμβο προορισμού και τον κόμβο που θα εξυπηρετείται από τον κινητό κόμβο. Από τα δυο σημεία επιλέγεται το σημείο το οποίο είναι πιο κοντά στον κόμβο προορισμού σε σχέση με τον κόμβο που θα εξυπηρετείται από τον κινητό κόμβο. Έστω (*xκ*, *yκ*) οι συντεταγμένες του κόμβου που θα εξυπηρετείται από τον κινητό κόμβο και (*xsink*, *ysink*) οι συντεταγμένες του κόμβου προορισμού. Στη περίπτωση που ο κόμβος έχει την ίδια συντεταγμένη-x (ή αντίστοιχα την ίδια συντεταγμένη-y) με τον κόμβο προορισμού, δηλαδή *xκ=xsink* (ή αντίστοιχα *yκ= ysink*), το σημείο τομής του θα είναι το (*xκ*, *yκ* + εύρος αίσθησης του κόμβου) αν *xκ<xsink* , και (*xκ*, *yκ* - εύρος αίσθησης του κόμβου) αν xκ>xsink (αντίστοιχα (*xκ* + εύρος αίσθησης του κόμβου, *yκ*) αν *yκ<ysink* , και (*xκ* - εύρος αίσθησης του κόμβου, *yκ*) αν *yκ>ysink*).

Όταν για κάθε κόμβο που έχει sending rate μεγαλύτερο από το additional_resources υπολογιστεί η θέση τοποθέτησης του κινητού κόμβου ώστε να τον εξυπηρετεί, εξετάζεται αν υπάρχει κάποιος κόμβος που να μην είναι σε συμφόρηση για να στέλνει τα δεδομένα του ο κινητός κόμβος.
Επιλέγεται να τοποθετηθεί κινητός κόμβος στη θέση που είναι πιο κοντά στον κόμβο προορισμού και για την οποία υπάρχει τουλάχιστον ένας κόμβος που να μην βρίσκεται σε συμφόρηση για να στέλνει τα δεδομένα του ο κινητός κόμβος που θα τοποθετηθεί.

Αν δεν υπάρχει κάποια θέση για να τοποθετηθεί ο κινητός κόμβος στην οποία να μπορεί να στέλνει τα δεδομένα του σε κάποιο κόμβο που να μην βρίσκεται

σε συμφόρηση, τότε οι πληροφορίες για τη θέση που είναι πιο κοντά στον κόμβο προορισμού αποθηκεύονται ως secondary_priority πληροφορίες ώστε να μπορούμε να επιστρέψουμε σε αυτή τη λύση όταν χρειαστεί. Οι secondary_priority πληροφορίες χρειάζονται όταν δε βρεθεί κάποια θέση για να τοποθετηθεί ο κινητός κόμβος στην οποία να μπορεί να στέλνει τα δεδομένα του σε κάποιο κόμβο που να μην είναι σε συμφόρηση.



Σχήμα 3.1 Επεξήγηση παραδείγματος τοποθέτησης κινητού κόμβου που να εξυπηρετεί μόνο ένα κόμβο

Επεξήγηση παραδείγματος στο Σχήμα 3.1:

Στο πιο πάνω παράδειγμα παρατηρούμε ότι ο κόμβος 4 είναι σε συμφόρηση γιατί τα πακέτα που λαμβάνει από τους κόμβους 1,2,3 είναι περισσότερα από τα πακέτα που μπορεί να στείλει. Ας υποθέσουμε ότι ο κόμβος 1 στέλνει τα δεδομένα του στον κόμβο 4 με sending rate μεγαλύτερο από το additional resources, και έστω ότι ο κόμβος 1 βρίσκεται στη θέση (0,0) , και ο κόμβος 5 ο οποίος είναι ο κόμβος προορισμού βρίσκεται στη θέση (2,4).

Η ευθεία που ενώνει τον κόμβο προορισμού και τον κόμβο 1, ο οποίος βρίσκεται στη θέση (0, 0), είναι η εξής:

$y = 2x$

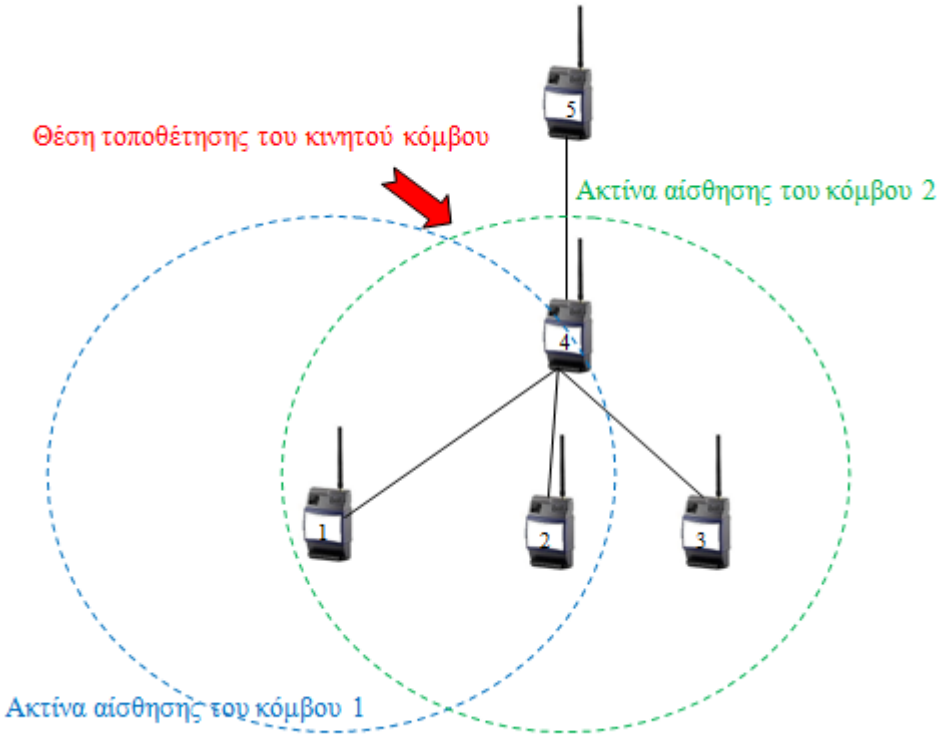
Τα σημεία τομής της ευθείας με τον κύκλο που δημιουργείται με κέντρο τον κόμβο 1, ο οποίος βρίσκεται στη θέση (0, 0) , και ακτίνα όσο το εύρος αίσθησης των κόμβων είναι τα εξής που για αυτό το παράδειγμα είναι 2.5 είναι τα εξής:
(1.118 , 2.236) (-1.118 , -2.236)
Η απόσταση μεταξύ του κινητού κόμβου και του κόμβου 1 είναι 2.5, δηλαδή όσο το εύρος αίσθησης του κόμβου 1 και παράλληλα η θέση του κινητού κόμβου είναι η πιο κοντινή θέση στον κόμβο προορισμού (στο εύρος αίσθησης του κόμβου 1).

3Β) Έλεγχος ύπαρξης βέλτιστης θέσης για να τοποθετηθεί ο κινητός κόμβος εξυπηρετώντας περισσότερους από ένα κόμβο – connectivity algorithm
Αν δεν βρεθεί κάποια θέση για να τοποθετηθεί ο κινητός κόμβος εξυπηρετώντας μόνο ένα κόμβο, τότε για κάθε αριθμό κόμβων (n) από 2 μέχρι 6 ή μέχρι το συνολικό αριθμό γειτόνων του κόμβου που βρίσκεται σε συμφόρηση, αν αυτός είναι μικρότερος από 6, εκτελείται η πιο κάτω διαδικασία. Σημειώνεται ότι οι σχετικές έρευνες [28-29] κατέληξαν στο συμπέρασμα ότι ο αριθμός των γειτόνων κάθε κόμβου πρέπει να είναι περίπου 6-8.

Αρχικά δημιουργούνται όλα τα υποσύνολα των κόμβων που στέλνουν τα δεδομένα τους στον κόμβο που βρίσκεται σε συμφόρηση, μεγέθους n, χρησιμοποιώντας τον αλγόριθμο στη δημοσίευση [27]. Κρατούμε μόνο τα υποσύνολα των οποίων το άθροισμα του sending rate των κόμβων του είναι μεγαλύτερο από το additional_resources που χρειάζεται ο κόμβος ο οποίος είναι σε συμφόρηση. Για κάθε ένα από αυτά τα υποσύνολα ελέγχουμε αν υπάρχει κάποιο σημείο το οποίο να είναι στο εύρος αίσθησης όλων των κόμβων του και παράλληλα να είναι όσο πιο κοντά γίνεται στον κόμβο προορισμού . Για να γίνει αυτό βρίσκουμε για κάθε ζευγάρι κόμβων του υποσυνόλου τα σημείο τομής των κύκλων που δημιουργούνται με κέντρο τον αντίστοιχο κόμβο και ακτίνα το εύρος αίσθησης του κόμβου. Στη συνέχεια ελέγχουμε αν αυτό το σημείο είναι στο εύρος αίσθησης όλων των υπόλοιπων κόμβων αισθητήρων του υποσυνόλου, εκτός από το ζευγάρι που εξετάζεται. Αν για ένα υποσύνολο

υπάρχουν πολλές τέτοιες θέσεις, τότε επιλέγεται ως κοινό σημείο του υποσυνόλου η θέση που είναι πιο κοντά στον κόμβο προορισμού.

Η διαδικασία σταματά όταν για κάποιο n [2-6] βρεθεί τουλάχιστον ένα κοινό σημείο για κάποιο υποσύνολο μεγέθους n. Αν υπάρχουν περισσότερα από ένα υποσύνολα μεγέθους n τα οποία να έχουν τουλάχιστον κοινό σημείο τότε επιλέγεται να τοποθετηθεί ένας κινητός κόμβος στο κοινό σημείο που είναι πιο κοντά στον κόμβο προορισμού. Με τον τρόπο αυτό, δηλαδή αρχίζοντας εξετάζοντας τα πιο μικρά υποσύνολα (υποσύνολα μεγέθους 2) μέχρι τα πιο μεγάλα (υποσύνολα μεγέθους 6), εξασφαλίζεται ότι το υποσύνολο που θα επιλεγεί να εξυπηρετείται από τον κινητό κόμβο είναι το μικρότερο δυνατόν. Έτσι εξασφαλίζουμε ότι δεν παραβιάζεται ο πρώτος περιορισμός του αλγορίθμου, που είναι να αλλάξουν πατέρα όσο το δυνατόν λιγότεροι κόμβοι.



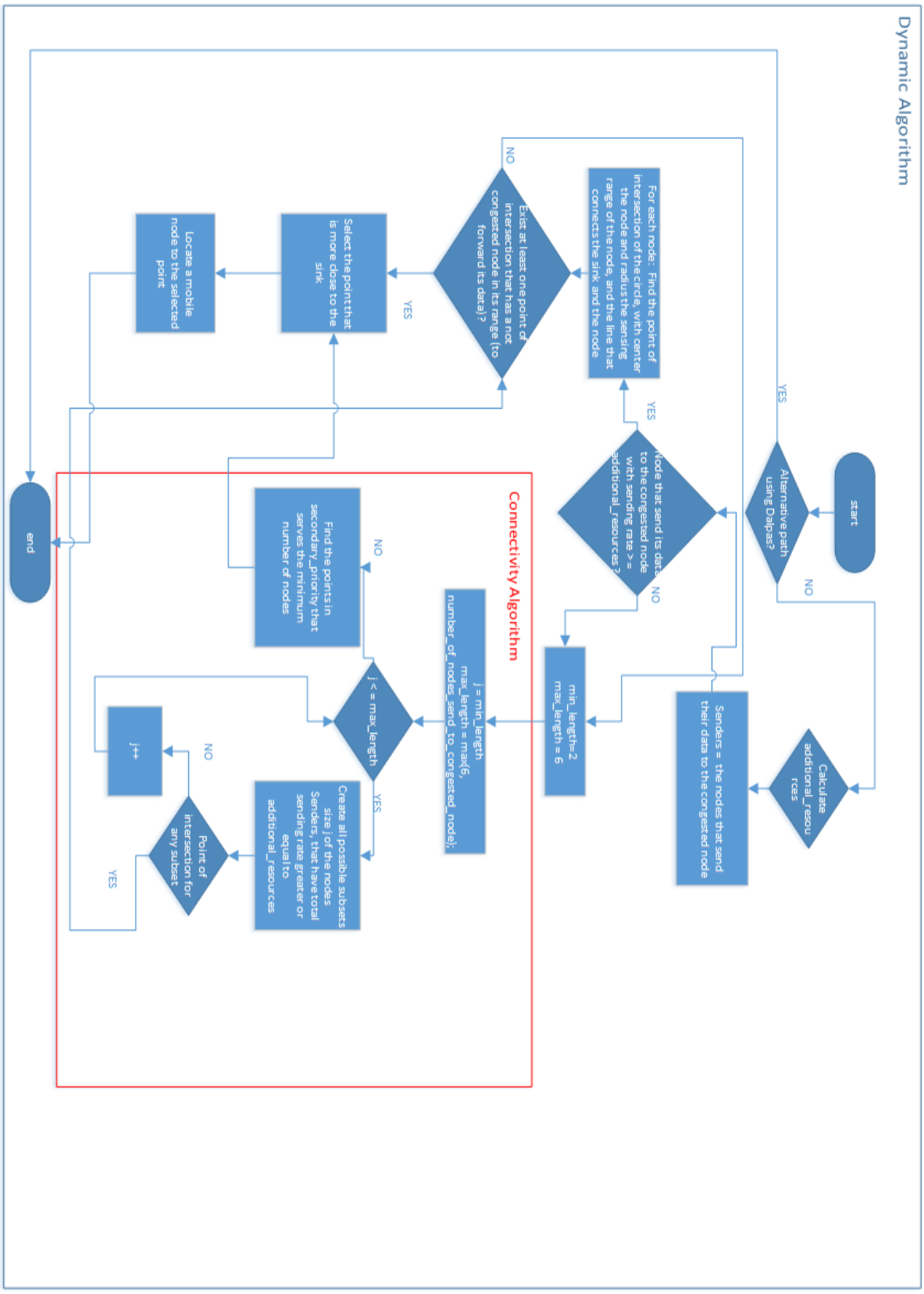
Σχήμα 3.2 Επεξήγηση παραδείγματος τοποθέτησης κινητού κόμβου που να εξυπηρετεί περισσότερους από ένα κόμβο

Επεξήγηση παραδείγματος στο Σχήμα 3.2:

Στο πιο πάνω παράδειγμα παρατηρούμε ότι ο κόμβος 4 είναι σε συμφόρηση γιατί τα πακέτα που λαμβάνει από τους κόμβους 1,2,3 είναι περισσότερα από τα πακέτα που μπορεί να στείλει. Αρχίζοντας από τα υποσύνολα μεγέθους 2 παρατηρούμε ότι τα πιθανά υποσύνολα κόμβων που θα εξυπηρετεί ο κινητός κόμβος είναι τα {1,2}, {1,3}, {2,3}.
Ας υποθέσουμε ότι μόνο το συνολικό sending rate των κόμβων 1 και 2 στον κόμβο 4 είναι μεγαλύτερο από το additional resources, και έστω ότι ο κόμβος 1 βρίσκεται στη θέση (0,0) , ο κόμβος 2 στη θέση (2,0) και ο κόμβος 5 ο οποίος είναι ο κόμβος προορισμού βρίσκεται στη θέση (2,4).
Ας θεωρήσουμε ότι το εύρος αίσθησης των κόμβων είναι 2.5 .
Τα σημεία τομής του κύκλου που δημιουργείται με κέντρο τον κόμβο 1 και ακτίνα όσο το εύρος αίσθησης των κόμβων και του κύκλου που δημιουργείται με κέντρο τον κόμβο 2 και ακτίνα όσο το εύρος αίσθησης των κόμβων είναι τα εξής: (2.291 , 1) (-2.291 , 1)
Η απόσταση μεταξύ του κινητού κόμβου και του κόμβου 1 και 2 είναι 2.5, δηλαδή όσο το εύρος αίσθησης των κόμβων και παράλληλα η θέση του κινητού κόμβου είναι η πιο κοντινή θέση στον κόμβο προορισμού (στο εύρος αίσθησης των κόμβων 1 και 2). Αν δεν υπήρχαν υποσύνολα μεγέθους 2 με συνολικό sending rate μεγαλύτερο από το additional resources τότε θα δημιουργούνταν υποσύνολα μεγέθους 3, δηλαδή θα δημιουργούταν το υποσύνολο {1, 2, 3} και θα υπολογιζόταν το σημείο τομής των κόμβων του υποσυνόλου που είναι πιο κοντά στον κόμβο προορισμού.

Τελειώνοντας όλα τα βήματα του αλγορίθμου, αν δεν βρέθηκε κάποια θέση για να τοποθετηθεί ο κινητός κόμβος στην οποία να μπορεί να στέλνει τα δεδομένα του σε κάποιο κόμβο που να μην βρίσκεται σε συμφόρηση, τότε επιλέγεται να τοποθετηθεί ένας κινητός κόμβος στη θέση που εξυπηρετεί το μικρότερο υποσύνολο κόμβων, παρόλο που δε θα μπορεί να στέλνει τα δεδομένα του σε κάποιο κόμβο που να μην βρίσκεται σε συμφόρηση. Σε περίπτωση που υπάρχουν πολλές θέσεις που εξυπηρετούν τον ίδιο αριθμό κόμβων, ο κινητός κόμβος τοποθετείται στη θέση που βρίσκεται πιο κοντά στον κόμβο προορισμού. Οι πληροφορίες αυτές ανακτώνται από τις secondary_priority πληροφορίες που δημιουργούνται κατά την εκτέλεση του αλγορίθμου. Όπως

εξηγείται και πιο πριν σε τέτοια περίπτωση , όταν τοποθετηθεί ο κινητός κόμβος στη θέση που του καθορίζεται θα αντιληφτεί την απουσία κάποιου κόμβου που να μην βρίσκεται σε συμφόρηση για να μεταδίδει τα δεδομένα που λαμβάνει και τότε θα ξαναεκτελεστεί ο αλγόριθμος για το νέο πρόβλημα συμφόρησης και θα τοποθετηθεί ένας άλλος κινητός κόμβος. Η διαδικασία αυτή θα συνεχιστεί μέχρι ο κινητός κόμβος που θα τοποθετηθεί να μπορεί να μεταδώσει τα δεδομένα του σε κάποιο κόμβο του δικτύου που να μην παρουσιάζει συμφόρηση ή να μπορεί να επικοινωνήσει απευθείας με τον κόμβο προορισμού.



Σχήμα 3.3 Διάγραμμα Ροής (Flow Chart) του αλγορίθμου Dynamic MobileCC

3.4 Απλά Παραδείγματα Εκτέλεσης του Αλγορίθμου

Ακολουθούν τρία πολύ απλά παραδείγματα εκτέλεσης του αλγορίθμου Dynamic MobileCC. Για όλα τα παραδείγματα που ακολουθούν το εύρος αίσθησης όλων των κόμβων είναι 2.5 .

Παράδειγμα 1:

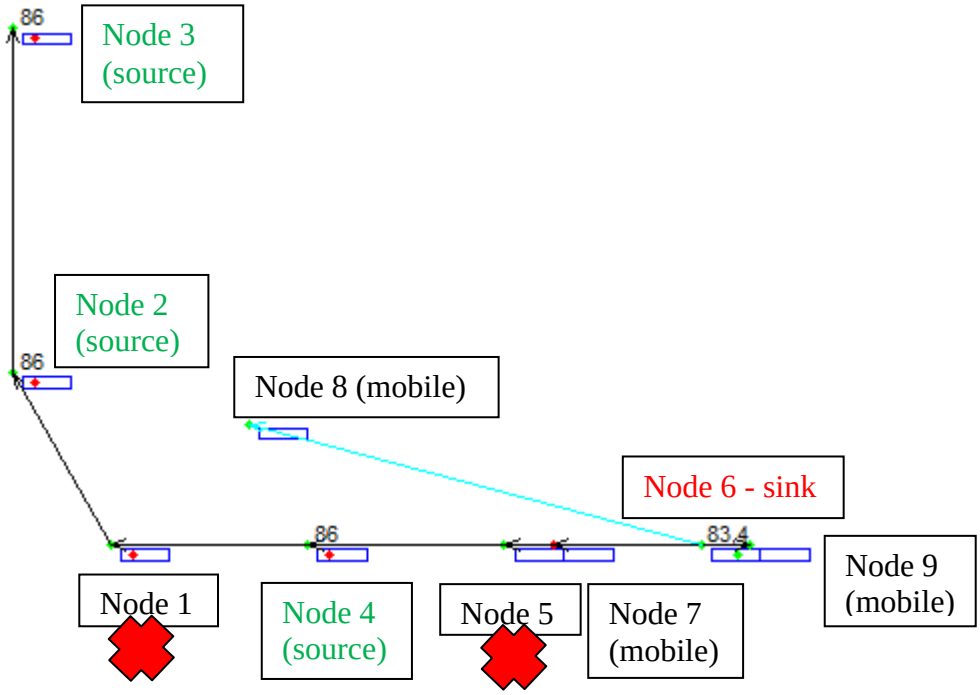
Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[1 2; 0 3; 0 5; 3 2; 5 2; 7 2; 7.5 2; 7.5 2; 7.5 2];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 6, ο οποίος βρίσκεται στη θέση (7.5, 2).

Ο κινητός κόμβος 7, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (7.5, 2), μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC τοποθετείται στη θέση (5.5 , 2).

Ο κινητός κόμβος 8, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (7.5, 2), μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC τοποθετείται στη θέση (2.4254, 2.6535).



Σχήμα 3.4 Αναπαράσταση του δικτύου μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC.

Οι κόμβοι 7, 8, 9 είναι mobile nodes και είναι τοποθετημένοι αρχικά κοντά στον κόμβο προορισμού.

Ο κόμβος 5 είναι σε συμφόρηση (congested) και έτσι τοποθετείται ο mobile node 7 στη θέση (5.5 , 2) για να εξυπηρετεί τον κόμβο 4 που είναι ο μόνος γείτονας του κόμβου 5 και δεν μπορεί να βρει εναλλακτική διαδρομή.

Επιλέγεται αυτό το σημείο γιατί ο κόμβος 4, ο οποίος είναι στη θέση (3, 2), έχει την ίδια συντεταγμένη y με τον κόμβο προορισμού, , ο οποίος είναι στη θέση (7,2), και άρα αυτό είναι το πιο μακρινό σημείο σε απόσταση 2.5 που είναι το communication range.

Στη συνέχεια ο κόμβος 1 είναι σε συμφόρηση (congested) και έτσι τοποθετείται ο mobile node 8 στη θέση (2.4 , 2.7) για να εξυπηρετεί τον κόμβο 2, ο οποίος είναι στη θέση (0, 3), και είναι ο μόνος γείτονας του κόμβου 1 και δεν μπορεί να βρει εναλλακτική διαδρομή.

Η ευθεία που ενώνει τον κόμβο προορισμού και τον κόμβο 2, ο οποίος βρίσκεται στη θέση (0, 3), είναι η εξής:$y = -1/7x + 3$

Τα σημεία τομής της ευθείας με τον κύκλο, με κέντρο τον κόμβο 2, ο οποίος βρίσκεται στη θέση (0, 3), και ακτίνα 2.5, όσο δηλαδή το εύρος αίσθησης των κόμβων, είναι τα εξής: (2.4254, 2.6535) (-2.4254, 3.3465)

Επιλέγεται να τοποθετηθεί ο mobile node 8 στο σημείο (2.4254, 2.6535) , γιατί αυτό το σημείο είναι πιο κοντά στον κόμβο προορισμού.

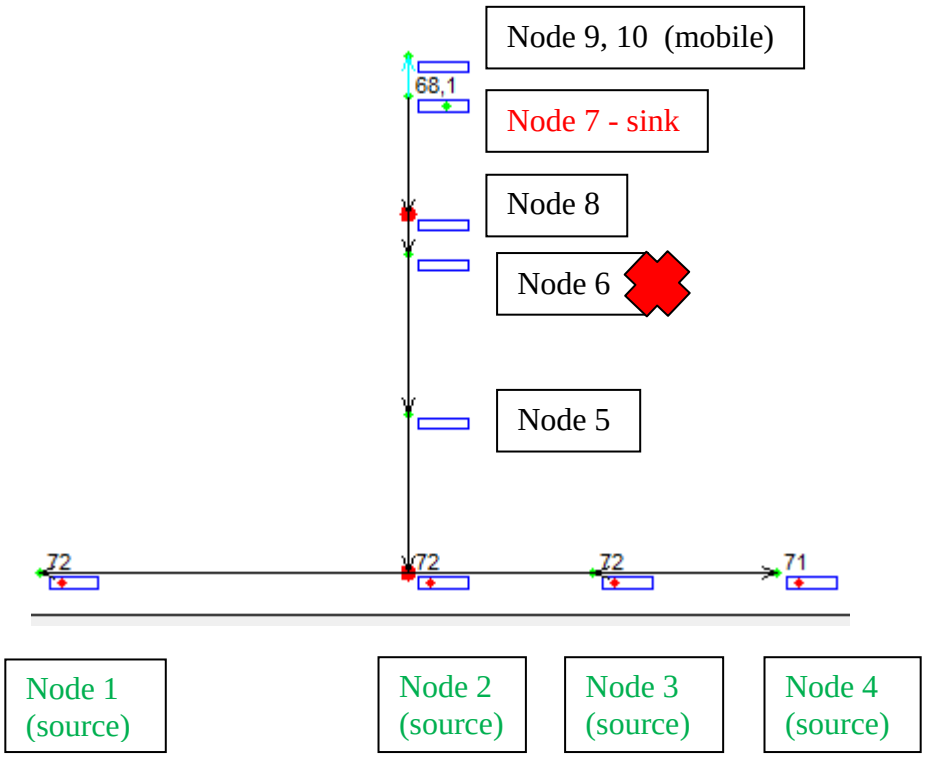
Παράδειγμα 2:

Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[0 0; 2 0; 3 0; 4 0; 2 2; 2 4; 2 6; 2 6.5; 2 6.5; 2 6.5];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 7, ο οποίος βρίσκεται στη θέση (2, 6).

Ο κινητός κόμβος 8, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (2, 6.5), μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC τοποθετείται στη θέση (2, 4.5).



Σχήμα 3.5 Αναπαράσταση του δικτύου μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC.

Οι κόμβοι 8, 9, 10 είναι mobile nodes και είναι τοποθετημένοι αρχικά κοντά στον κόμβο προορισμού.

Ο κόμβος 6 είναι σε συμφόρηση (congested) και έτσι τοποθετείται ο mobile node 8 στη θέση (2, 4.5) για να εξυπηρετεί τον κόμβο 5, ο οποίος βρίσκεται στη θέση (2, 2), που είναι γείτονας του κόμβου 6 και δεν μπορεί να βρει εναλλακτική διαδρομή.

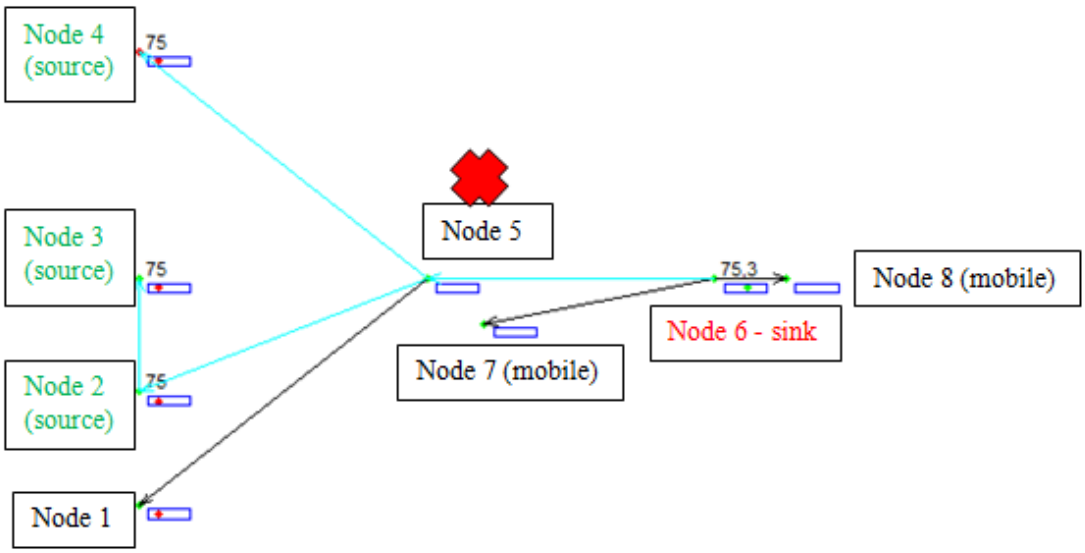
Επιλέγεται αυτό το σημείο γιατί ο κόμβος 5, ο οποίος βρίσκεται στη θέση(2, 2), έχει την ίδια συντεταγμένη x με τον κόμβο προορισμού (2, 6) και άρα αυτό είναι το σημείο που είναι πιο κοντά στον κόμβο προορισμού και ταυτόχρονα είναι σε απόσταση ίση με 2.5 από τον κόμβο 5 που είναι το εύρος αίσθησης των κόμβων.

Παράδειγμα 3:

Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[0 0; 0 1; 0 2; 0 4; 2 2; 4 2; 4.5 2; 4.5 2];
Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 6, ο οποίος βρίσκεται στη θέση (4, 2).

Ο κινητός κόμβος 7, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (4.5, 2), μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC τοποθετείται στη θέση (2.3768, 1.5942).



Σχήμα 3.6 Αναπαράσταση του δικτύου μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC.

Οι κόμβοι 7, 8 είναι mobile nodes και είναι τοποθετημένοι αρχικά κοντά στον κόμβο προορισμού.
Ο κόμβος 5 είναι σε συμφόρηση (congested) και έτσι τοποθετείται ο mobile node 7 στη θέση (2.3768, 1.5942) για να εξυπηρετεί τον κόμβο 2 που είναι γείτονας του κόμβου 5 και δεν μπορεί να βρει εναλλακτική διαδρομή.
Εξετάζονται όλα τα σημεία που μπορεί να τοποθετηθεί κάποιο mobile node το οποίο να εξυπηρετεί τους κόμβους (γείτονες) του congested node, που στέλνουν φορτίο στο congested node όσο το additional_resources που εξυπηρετεί ο congested node. Οι κόμβοι (γείτονες) του congested node που του στέλνουν φορτίο μεγαλύτερο από το additional_resources είναι ο 2 και ο 4.

Η ευθεία που ενώνει τον κόμβο προορισμού και τον κόμβο 2, ο οποίος βρίσκεται στη θέση (0, 1), είναι η εξής : y = 1/4x + 1

Τα σημεία τομής της ευθείας και του κύκλο που δημιουργείται με κέντρο τον κόμβο 2, ο οποίος βρίσκεται στη θέση (1,0) , και ακτίνα όσο το εύρος αίσθησης των κόμβων είναι τα εξής: (2.3768, 1.5942) (-2.3768, 0.4058)

Η ευθεία που ενώνει τον κόμβο προορισμού και τον κόμβο 4, ο οποίος βρίσκεται στη θέση (0, 4), είναι η εξής: y = -1/2x + 2

Τα σημεία τομής της ευθείας με τον κύκλο που δημιουργείται με κέντρο τον κόμβο 4, ο οποίος βρίσκεται στη θέση (0,4) , και ακτίνα όσο το εύρος αίσθησης των κόμβων είναι τα εξής: (2.1913, 2.9043) (-2.1913, 5.0957)

Επιλέγεται να τοποθετηθεί ένα mobile node στο σημείο (2.3768, 1.5942), γιατί αυτό το σημείο είναι πιο κοντά στον κόμβο προορισμού σε σχέση με τα υπόλοιπα.

Παράδειγμα 4:

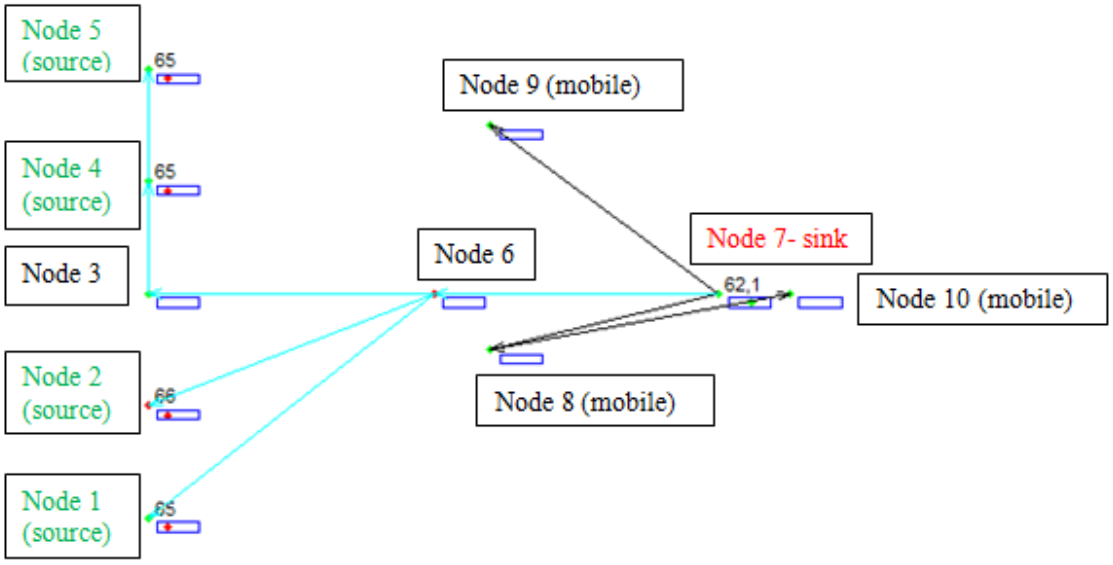
Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[0 1; 0 2; 0 3; 0 4; 0 5; 2 3; 4 3; 4.5 3; 4.5 3; 4.5 3];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 7, ο οποίος βρίσκεται στη θέση (4, 3).

Ο κινητός κόμβος 8, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (4.5, 3), μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC τοποθετείται στη θέση (2.3984, 2.5).

Ο κινητός κόμβος 9, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (4.5, 3), μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC τοποθετείται στη θέση (2.3984, 4.5).



Σχήμα 3.6 Αναπαράσταση του δικτύου μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC.

Οι κόμβοι 8,9,10 είναι mobile nodes και είναι τοποθετημένοι αρχικά κοντά στον κόμβο προορισμού.

Ο κόμβος 6 είναι congested και έτσι τοποθετείται ο κινητός κόμβος 8 στη θέση (2.3984, 2.5) για να εξυπηρετεί τους κόμβους 2 και 3 που είναι γείτονες του κόμβου 6 και δεν μπορούν να βρουν εναλλακτική διαδρομή.

Επιλέγονται δυο κόμβοι να αλλάξουν πατέρα γιατί δεν υπάρχει κάποιος κόμβος που να στέλνει τόσα πακέτα ανά δευτερόλεπτο στον congested node, όσο το additional resources που χρειάζεται το congested node.

Οι γείτονες του congested node που του στέλνουν δεδομένα είναι οι κόμβοι 1,2,3.

Τα σημεία τομής των κόμβων 1,2 είναι: (-2.3984, 1.5) (2.3984, 1.5)

Τα σημεία τομής των κόμβων 1,3 είναι: (-2.2366, 2) (2.23661, 2)

Τα σημεία τομής των κόμβων 2,3 είναι: (-2.3984, 2.5) (2.3984, 2.5)

Επιλέγεται να τοποθετηθεί ο κινητός κόμβος 8 στο σημείο (2.3984, 2.5) , γιατί αυτό το σημείο είναι πιο κοντά στον κόμβο προορισμού, ο οποίος βρίσκεται στη θέση (4,3).

Ο κόμβος 3 είναι congested και έτσι τοποθετείται ο κινητός κόμβος 9 στη θέση (2.4, 4.5) για να εξυπηρετεί τους κόμβους 4 και 5 που είναι γείτονες του κόμβου 3 και δεν μπορούν να βρουν εναλλακτική διαδρομή.

Επιλέγονται δυο κόμβοι να αλλάξουν πατέρα γιατί δεν υπάρχει κάποιος κόμβος που να στέλνει πακέτα όσο το additional resources στον congested node.
Οι γείτονες του congested node (που του στέλνουν πακέτα) είναι οι κόμβοι 4, 5.
Τα σημεία τομής των κόμβων 4,5 είναι: (2.3984, 4.5) (-2.3984, 4.5)
Επιλέγεται να τοποθετηθεί το mobile node 9 στο σημείο (2.3984, 4.5) , γιατί αυτό το σημείο είναι πιο κοντά στον κόμβο προορισμού, ο οποίος βρίσκεται στη θέση(4,3).

Κεφάλαιο 4

Ο Αλγόριθμος Direct Path MobileCC

Ο δεύτερος αλγόριθμος που αναπτύχθηκε είναι ο αλγόριθμος Direct Path MobileCC, ο οποίος λύνει κάθε θέμα συμφόρησης δημιουργώντας ένα μονοπάτι με κινητούς κόμβους, το οποίο αρχίζει από μερικούς από τους κόμβους που στέλνουν τα δεδομένα τους στον κόμβο που είναι σε συμφόρηση (congested node), και τελειώνει στον κόμβο προορισμού. Στο κεφάλαιο αυτό περιγράφεται η δομή του αλγορίθμου, παρουσιάζεται το διάγραμμα ροής δεδομένων του αλγορίθμου, και ακολούθως περιγράφεται αναλυτικά ο αλγόριθμος. Τέλος περιγράφονται μερικά παραδείγματα εκτέλεσης του.

4.1 Εισαγωγή

Ο Direct Path MobileCC είναι ένας αλγόριθμος ελέγχου συμφόρησης ο οποίος μπορεί να χρησιμοποιηθεί ως επέκταση οποιουδήποτε αλγορίθμου ελέγχου συμφόρησης. Σημειώνεται ότι ο Direct Path MobileCC δεν αντικαταστεί τους υπάρχον αλγόριθμους ελέγχου τοπολογίας ή τους αλγόριθμους ελέγχου συμφόρησης ή τους αλγόριθμους δρομολόγησης, αλλά τρέχει παράλληλα με αυτούς. Ο Direct Path MobileCC δημιουργεί εναλλακτικά μονοπάτια τοποθετώντας κινητούς κόμβους αισθητήρες, σε περίπτωση υπερφόρτωσης του δικτύου και αποτυχίας του αλγόριθμου ελέγχου συμφόρησης που χρησιμοποιείται ήδη να δημιουργήσει εναλλακτικά μονοπάτια. Ο αλγόριθμος

θεωρεί ότι ένας ικανοποιητικός αριθμός κινητών αισθητήρων, ανάλογα με το μέγεθος του δικτύου, είναι τοποθετημένοι αρχικά κοντά στο κόμβο προορισμού σε κατάσταση ύπνωσης (sleep state) ώστε να μην σπαταλείται η ενέργεια τους.

Ο Direct Path MobileCC λύνει μεμονωμένα κάθε πρόβλημα συμφόρησης, αλλά επιλύει ολοκληρωτικά το πρόβλημα δημιουργώντας ένα μονοπάτι αποτελούμενο από κινητούς κόμβους, που καταλήγει στον κόμβο προορισμού, σε αντίθεση με τον αλγόριθμο Dynamic MobileCC, ο οποίος λύνει τοπικά κάθε θέμα συμφόρησης. Για τον καθορισμό της θέσης του πρώτου κινητού κόμβου, ο οποίος θα λαμβάνει δεδομένα από μερικούς από τους κόμβους που έχουν πατέρα τον κόμβο που βρίσκεται σε συμφόρηση, εκτελείται ο αλγόριθμος Dynamic MobileCC. Στη συνέχεια δημιουργείται ένα μονοπάτι με κινητούς κόμβους από τον πρώτο κινητό κόμβο μέχρι τον κόμβο προορισμού (κάθε κόμβος εξυπηρετεί τον προηγούμενο που τοποθετήθηκε). Με τον τρόπο αυτό γίνεται η αποκατάσταση του θέματος συμφόρησης που εξετάζεται πολύ γρήγορα. Ακόμη με αυτόν τον τρόπο αποκλείουμε την περίπτωση να δημιουργηθεί συμφόρηση στον πρώτο κινητό κόμβο που θα τοποθετηθεί, λόγω απουσίας κάποιου κόμβου που να μην είναι σε συμφόρηση για να στέλνει τα δεδομένα του. Από την άλλη όμως με αυτόν τον τρόπο υπάρχουν περιπτώσεις όπου γίνεται σπατάλη κινητών κόμβων, οι οποίοι συνήθως κοστίζουν. Η σπατάλη γίνεται όταν υπάρχει τουλάχιστον ένας κόμβος που δεν βρίσκεται σε συμφόρηση και μπορεί ο κινητός κόμβος που θα τοποθετηθεί να στέλνει τα δεδομένα του σε αυτόν.

Επίσης λόγω του γεγονός ότι ο αλγόριθμος εξετάζεται μεμονωμένα κάθε πρόβλημα συμφόρησης (όπως και ο Dynamic MobileCC), δεν ελέγχει αν μόνο ένας κινητός κόμβος μπορεί να τοποθετηθεί σε τέτοια θέση ώστε να επιλύει δυο ή περισσότερα θέματα συμφόρησης ταυτόχρονα. Άρα σε περιπτώσεις όπου πολλά θέματα συμφόρησης μπορούν να επιλυθούν τοποθετώντας ένα μόνο κινητό κόμβο σε κατάλληλη θέση, ο Direct Path MobileCC σπαταλάει επιπλέον κινητούς κόμβους αφού όπως προαναφέρθηκε επιλύει μεμονωμένα κάθε πρόβλημα συμφόρησης δημιουργώντας ένα μονοπάτι με κινητούς κόμβους για το καθένα.

Ο αλγόριθμος Direct Path MobileCC προσπαθεί να διαχειριστεί θέματα συμφόρησης με όσο το δυνατόν πιο λίγη πολυπλοκότητα γίνετε, έτσι ώστε να είναι παράλληλα αποδοτικός ως προς την ενέργεια αλγορίθμου και να τοποθετούνται όσο το δυνατό γρηγορότερα οι κινητοί κόμβοι, έτσι ώστε οι συνέπειες της συμφόρησης, όπως απώλεια πακέτων, μεγάλη καθυστέρηση αποστολής πακέτων κλπ, να μην οξυνθούν.

4.2 Περιγραφή της Δομής του Αλγορίθμου

Ο αλγόριθμος Direct PAth MobileCC αποτελείται από τα εξής βήματα:

1) Υπολογισμός της θέσης του πρώτου κινητού κόμβου που θα τοποθετηθεί χρησιμοποιώντας τον αλγόριθμο Dynamic MobileCC

2) Δημιουργία μονοπατιού αποτελούμενου από κινητούς κόμβους, ξεκινώντας από τον πρώτο κινητό κόμβο, του οποίου υπολογίστηκε η θέση του στο προηγούμενο βήμα, και καταλήγοντας στον κόμβο προορισμού

4.3 Αναλυτική Περιγραφή του Αλγορίθμου

Πιο κάτω περιγράφεται βηματικά η λειτουργία του αλγορίθμου για ένα θέμα συμφόρησης (ένα congested node). Η ίδια λειτουργία ακολουθείτε για κάθε θέμα συμφόρησης.

1) Υπολογισμός της θέσης του πρώτου κινητού κόμβου που θα τοποθετηθεί χρησιμοποιώντας τον αλγόριθμο Dynamic MobileCC
Αρχικά καλείται ο αλγόριθμος Dynamic Mobile CC, ο οποίος περιγράφεται στο προηγούμενο κεφάλαιο, ο οποίος υπολογίζει τη βέλτιστη θέση για να τοποθετηθεί ο πρώτος κινητός κόμβος και το υποσύνολο των κόμβων που στέλνουν τα δεδομένα τους στον κόμβο που βρίσκεται σε συμφόρηση και πλέον θα τα στέλνουν στον πρώτο κινητό κόμβο που θα τοποθετηθεί.

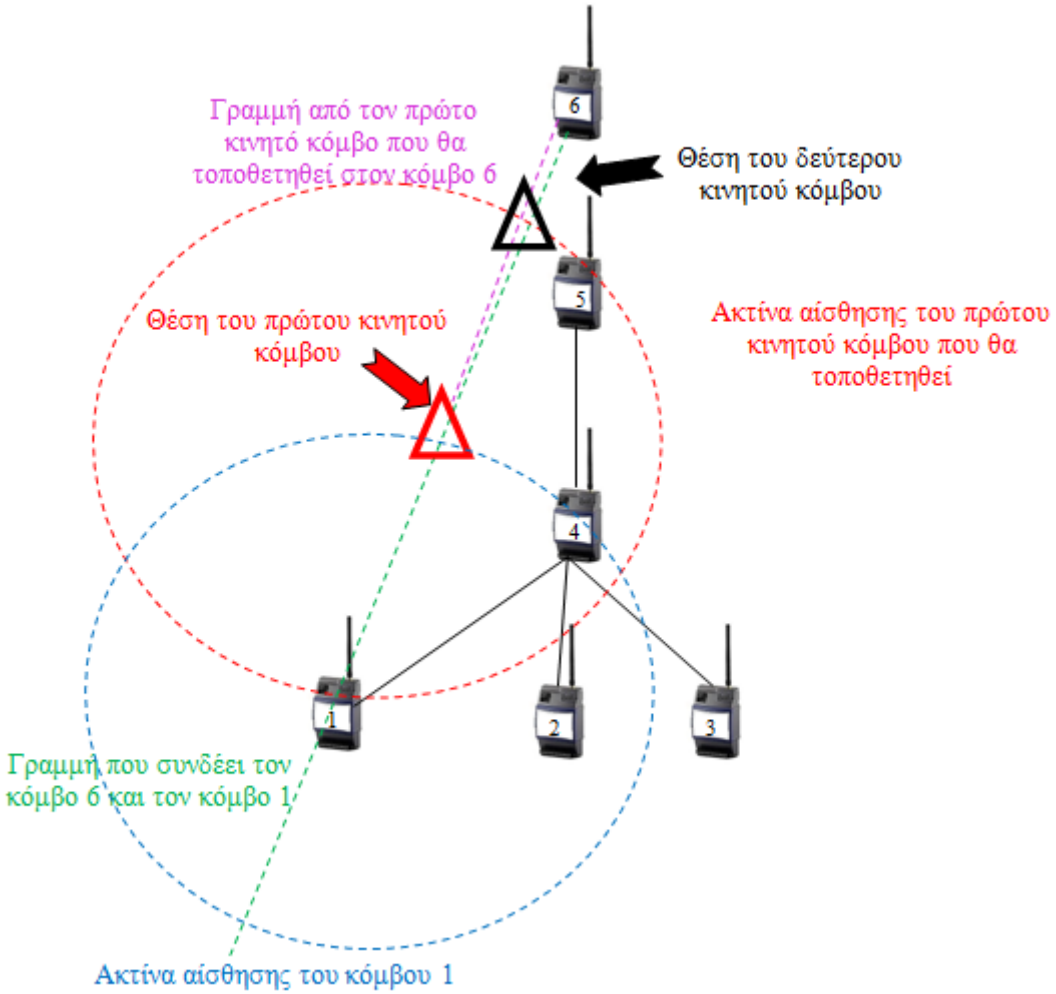
2) Δημιουργία μονοπατιού αποτελούμενου από κινητούς κόμβους, ξεκινώντας από τον πρώτο κινητό κόμβο, του οποίου υπολογίστηκε η θέση του στο προηγούμενο βήμα, και καταλήγοντας στον κόμβο προορισμού

Αρχικά εξετάζεται αν ο προηγούμενος κινητός κόμβος που τοποθετήθηκε μπορεί να επικοινωνήσει με τον κόμβο προορισμού, αν ναι τότε τερματίζεται ο αλγόριθμος και τοποθετούνται οι κινητοί κόμβοι στις θέσεις που υπολογιστήκαν κατά την εκτέλεση του. Για τον υπολογισμό της θέσης του επόμενου κινητού κόμβου, υπολογίζεται το σημείο στο οποίο θα πρέπει να τοποθετηθεί ο κινητός κόμβος ώστε να εξυπηρετεί τον κινητό κόμβο, του οποίου καθορίστηκε η θέση του στο προηγούμενο βήμα, άρα να είναι στο εύρος αίσθησης του και ταυτόχρονα να είναι όσο το δυνατόν πιο κοντά στον κόμβο προορισμού είναι ο εξής:

Αρχικά υπολογίζονται τα δυο σημεία τομής του κύκλου που δημιουργείται με ακτίνα το εύρος αίσθησης του προηγούμενου κινητού κόμβου και κέντρο τον κόμβο αυτόν, και της ευθείας που ενώνει τον κόμβο προορισμού και τον προηγούμενο κινητό κόμβο. Από τα δυο σημεία επιλέγεται το σημείο το οποίο είναι πιο κοντά στον κόμβο προορισμού σε σχέση με τον προηγούμενο κινητό κόμβο.

Έστω (*xκ*, *yκ*) οι συντεταγμένες του προηγούμενου κινητού κόμβου που θα εξυπηρετείται από τον νέο κινητό κόμβο που θα τοποθετηθεί και (*xsink*, *ysink*) οι συντεταγμένες του κόμβου sink. Στη περίπτωση που ο κόμβος έχει την ίδια συντεταγμένη-x (ή αντίστοιχα την ίδια συντεταγμένη-y) με τον κόμβο προορισμού, δηλαδή *xκ=xsink* (ή αντίστοιχα *yκ= ysink*), το σημείο τομής του θα είναι το (*xκ*, *yκ* + εύρος αίσθησης του κόμβου) αν *xκ<xsink* , και (*xκ*, *yκ* - εύρος αίσθησης του κόμβου) αν *xκ>xsink* (αντίστοιχα (*xκ* + εύρος αίσθησης του κόμβου, *yκ*) αν *yκ<ysink* , και (*xκ* - εύρος αίσθησης του κόμβου, *yκ*) αν *yκ>ysink*).

Επιλέγεται να τοποθετηθεί ο κινητός κόμβος στη θέση που είναι πιο κοντά στον κόμβο προορισμού.



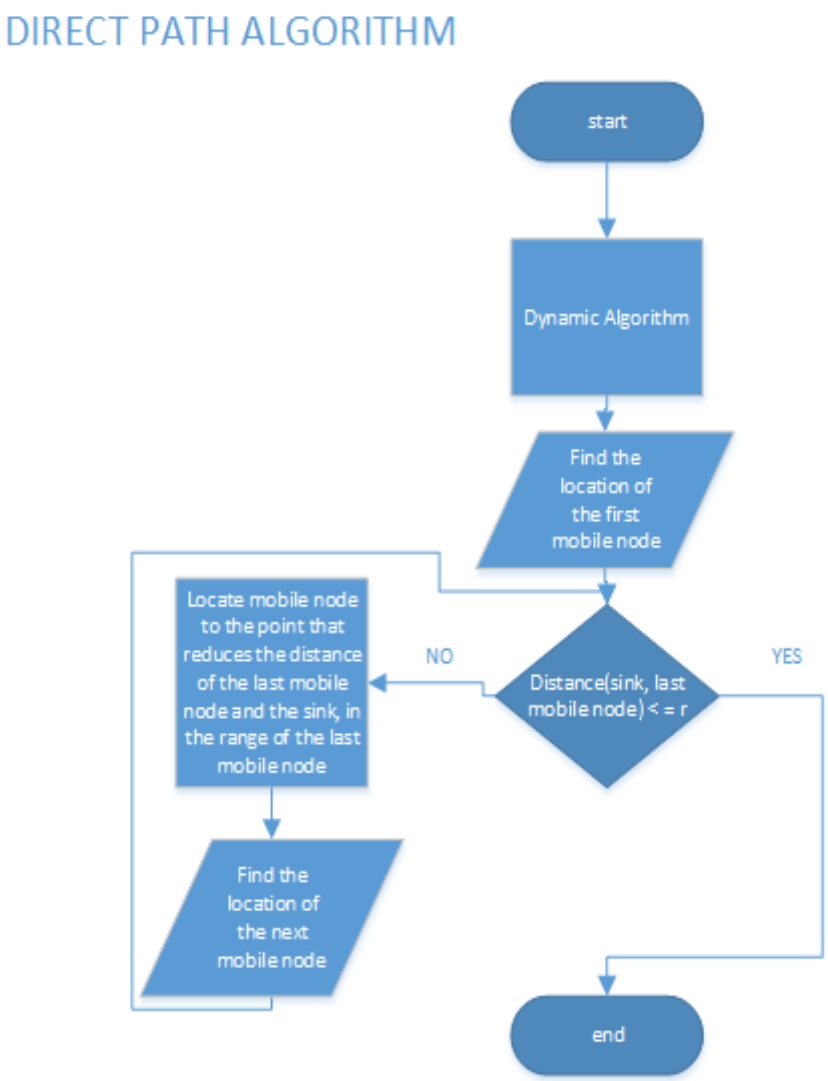
Σχήμα 4.1 Επεξήγηση παραδείγματος εκτέλεσης του Direct Path MobileCC

Επεξήγηση παραδείγματος στο Σχήμα 4.1:

Στο πιο πάνω παράδειγμα παρατηρούμε ότι ο κόμβος 4 είναι σε συμφόρηση και με βάση τον αλγόριθμο Dynamic MobileCC αποφασίζεται να τοποθετηθεί ο πρώτος κινητός κόμβος στη θέση (1 , 2.5) για να εξυπηρετεί τον κόμβο 1 ο οποίος στέλνει τα δεδομένα του στον κόμβο 4 και δεν μπορεί να βρει εναλλακτική διαδρομή. Η θέση του πρώτου κινητού κόμβου φαίνεται στο σχήμα με ένα κόκκινο τρίγωνο. Στη συνέχεια εκτελείται ο αλγόριθμος Direct Path MobileCC ο οποίος δημιουργεί ένα μονοπάτι από τον πρώτο κινητό κόμβο στον κόμβο προορισμού.

Ας θεωρήσουμε ότι το εύρος αίσθησης των κόμβων είναι 2.5 .

Η ευθεία που ενώνει τον κόμβο προορισμού ο οποίος βρίσκεται στη θέση (2, 6) και τον πρώτο κινητό κόμβο, ο οποίος βρίσκεται στη θέση (1, 2.5), είναι η εξής:
y = 7/2x - 1
Τα σημεία τομής της πιο πάνω ευθείας και του κύκλου που δημιουργείται με κέντρο τον πρώτο κινητό κόμβο που τοποθετήθηκε και ακτίνα όσο το εύρος αίσθησης των κόμβων είναι τα εξής: (1.7, 4.9) (0.3, 0.1)
Η θέση πιο κοντά στον κόμβο προορισμού, για να τοποθετηθεί ο δεύτερος κινητός κόμβος (στο εύρος αίσθησης του πρώτου κινητού κόμβου) είναι η (1.7, 4.9).



Σχήμα 4.2 Διάγραμμα Ροής (Flow Chart) του αλγορίθμου Direct Path MobileCC

4.4 Απλά Παραδείγματα Εκτέλεσης του Αλγορίθμου

Ακολουθούν τρία πολύ απλά παραδείγματα εκτέλεσης του αλγορίθμου Direct Path MobileCC. Για όλα τα παραδείγματα που ακολουθούν το εύρος αίσθησης όλων των κόμβων είναι 2.5 .

Παράδειγμα 1:

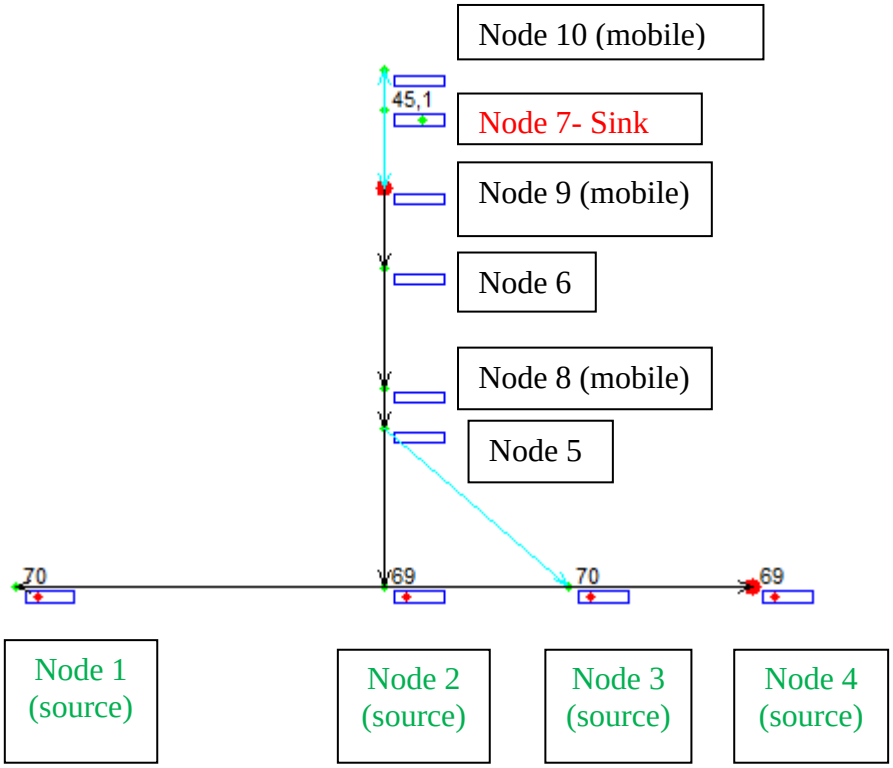
Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[0 0; 2 0; 3 0; 4 0; 2 2; 2 4; 2 6; 2 6.5; 2 6.5; 2 6.5];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 7, ο οποίος βρίσκεται στη θέση (2, 6).

Ο κινητός κόμβος 8, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (2, 6.5), μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC τοποθετείται στη θέση (2, 2.5).

Ο κινητός κόμβος 9, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (2, 6.5), μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC τοποθετείται στη θέση (2, 5).



Σχήμα 4.3 Αναπαράσταση του δικτύου μετά την εκτέλεση του αλγορίθμου Direct Path MobileCC.

Οι κόμβοι 8, 9, 10 είναι mobile nodes και είναι τοποθετημένοι αρχικά κοντά στον κόμβο προορισμού.

Ο κόμβος 5 είναι σε συμφόρηση και έτσι καλείται ο αλγόριθμος Direct Path MobileCC για να δημιουργηθεί μια διαδρομή αποτελούμενη από κινητούς κόμβους, η οποία να ξεκινά από κάποιο γείτονα του κόμβου 5, και να καταλήγει στον κόμβο προορισμού. Ο κινητός κόμβος 8 τοποθετείται στη θέση (2, 2.5) για να εξυπηρετεί τον κόμβο 2 που είναι γείτονας του κόμβου 5 και δεν μπορεί να βρει εναλλακτική διαδρομή. Η θέση του κινητού κόμβου 8 υπολογίζεται με τον αλγόριθμο Dynamic MobileCC. Ο κινητός κόμβος 9 τοποθετείται στη θέση (2, 5) για να εξυπηρετεί τον κινητό κόμβο 8 και έτσι να δημιουργείται διαδρομή από τον κινητό κόμβο 8 στον κόμβο προορισμού. Έτσι με την τοποθέτηση των δυο αυτών κινητών κόμβων δημιουργείται μια διαδρομή από τον κόμβο 2 στον κόμβο προορισμού.

Παράδειγμα 2:

Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

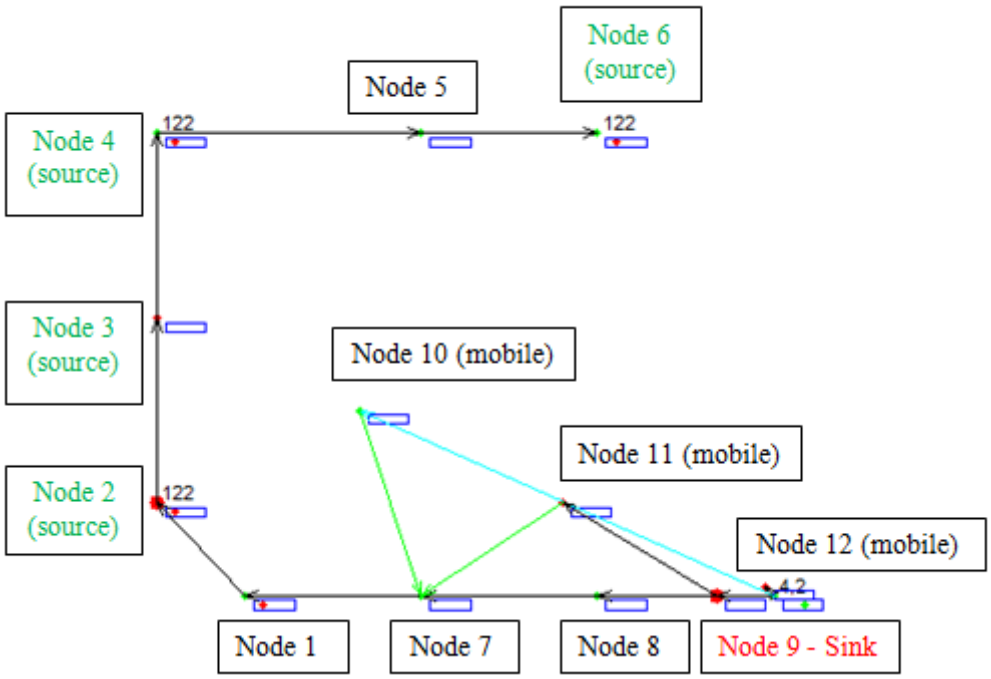
topology=[1 2; 0 3; 0 5; 0 7; 3 7; 5 7; 3 2; 5 2; 7 2; 7.5 2; 7.5 2; 7.5 2; 7.5 2; 7.5 2];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 9, ο οποίος βρίσκεται στη θέση (7, 2).

Ο κινητός κόμβος 10, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (7.5, 2), μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC τοποθετείται στη θέση (2.2519, 4.0349).

Ο κινητός κόμβος 11, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (7.5, 2), μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC τοποθετείται στη θέση (4.5544, 3.0407).

Ο κινητός κόμβος 12, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (7.5, 2), μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC τοποθετείται στη θέση (6.8615, 2.0577).



Σχήμα 4.4 Αναπαράσταση του δικτύου μετά την εκτέλεση του αλγορίθμου Direct Path MobileCC.

Οι κόμβοι 10, 11, 12, 13, 14 είναι mobile nodes και είναι τοποθετημένοι αρχικά κοντά στον κόμβο προορισμού.

Ο κόμβος 2 είναι σε συμφόρηση και έτσι καλείται ο αλγόριθμος Direct Path MobileCC για να δημιουργηθεί μια διαδρομή αποτελούμενη από κινητούς κόμβους, η οποία να ξεκινά από κάποιο γείτονα του κόμβου 5, και να καταλήγει στον κόμβο προορισμού. Ο κινητός κόμβος 10 τοποθετείται στη θέση (2.2519, 4.0349) για να εξυπηρετεί τον κόμβο 3 που είναι γείτονας του κόμβου 2 και δεν μπορεί να βρει εναλλακτική διαδρομή. Η θέση του κινητού κόμβου 10 υπολογίζεται εκτελώντας τον αλγόριθμο Dynamic MobileCC. Ο κόμβος (γείτονας) του congested node που του στέλνει φορτίο μεγαλύτερο από το additional_resources είναι ο 3.

Η ευθεία που ενώνει τον κόμβο προορισμού, ο οποίος βρίσκεται στη θέση (7, 2) , με τον κύκλο που δημιουργείται λόγω του range του κόμβου 3, ο οποίος βρίσκεται στη θέση (0, 5) , είναι η : y = -0.4286x + 5

Τα σημεία τομής της ευθείας με τον κύκλο είναι τα εξής:

(2.2519, 4.0349) (-2.2519, 5.9651)

Επιλέγεται να τοποθετηθεί ο κινητός κόμβος 10 στο σημείο (2.2519, 4.0349), γιατί αυτό το σημείο είναι πιο κοντά στον κόμβο προορισμού.

Η ευθεία που ενώνει τον κόμβο προορισμού, ο οποίος βρίσκεται στη θέση (7, 2) , με τον κύκλο που δημιουργείται λόγω του range του κόμβου 10, ο οποίος βρίσκεται στη θέση (2.2519, 4.0349) , είναι η εξής: y = -0.4255x + 4.97865

Τα σημεία τομής της ευθείας με τον κύκλο είναι τα εξής:

(4.5544, 3.0407) (0.0456, 4.9593)

Επιλέγεται να τοποθετηθεί το mobile node 11 στο σημείο (4.5544, 3.0407) γιατί αυτό το σημείο είναι πιο κοντά στο κόμβο προορισμού και παράλληλα είναι στο εύρος αίσθησης του κόμβου 10.

Η ευθεία που ενώνει τον κόμβο προορισμού, ο οποίος βρίσκεται στη θέση (7, 2), και τον κινητό κόμβο 11, ο οποίος βρίσκεται στη θέση (4.6, 3), είναι η εξής:

y = -0.4167x + 4.91682

Τα σημεία τομής της ευθείας με τον κύκλο είναι τα εξής:

(6.8615, 2.0577) (2.3385, 3.9423)

Επιλέγεται να τοποθετηθεί το mobile node 12 στο σημείο (6.8615, 2.0577) , γιατί αυτό το σημείο είναι πιο κοντά στον κόμβο προορισμού.

Έτσι δημιουργείται μια διαδρομή από τον κόμβο 3 στον κόμβο προορισμού.

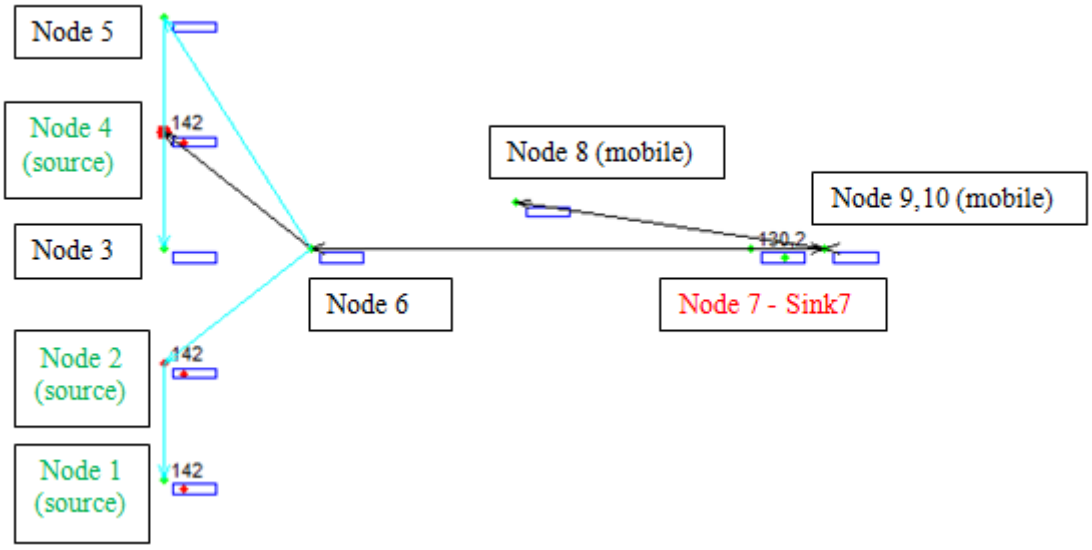
Παράδειγμα 3:

Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[0 1; 0 2; 0 3; 0 4; 0 5; 1 3; 4 3; 4.5 3; 4.5 3; 4.5 3];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 9, ο οποίος βρίσκεται στη θέση (7, 2).

Ο κινητός κόμβος 8, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (4.5, 3), μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC τοποθετείται στη θέση (2.4, 3.4).



Σχήμα 4.5 Αναπαράσταση του δικτύου μετά την εκτέλεση του αλγορίθμου Direct Path MobileCC.

Οι κόμβοι 8, 9, 10 είναι κινητοί κόμβοι και είναι τοποθετημένοι αρχικά κοντά στον κόμβο προορισμού.

Ο κόμβος 6 είναι congested και έτσι τοποθετείται ο κινητός κόμβος 8 στη θέση (2.4, 3.4) για να εξυπηρετεί τον κόμβο 4 που είναι γείτονας του κόμβου 6 και δεν μπορεί να βρει εναλλακτική διαδρομή.

Δεν τοποθετηθεί άλλος κινητός κόμβος γιατί ο κινητός κόμβος που τοποθετήθηκε είναι στο εύρος αίσθησης του κόμβου προορισμού.

Έτσι δημιουργείται μια διαδρομή από τον κόμβο 4 στον κόμβο προορισμού.

Κεφάλαιο 5

Ο αλγόριθμος Global MobileCC

Ο τρίτος αλγόριθμος που αναπτύχθηκε είναι ο αλγόριθμος Global MobileCC, ο οποίος γνωρίζει εκ των προτέρων όλα τα θέματα συμφόρησης που θα εμφανιστούν στο δίκτυο κι έτσι τα λύνει καθολικά, δηλαδή δε λύνει τοπικά κάθε πρόβλημα. Αυτός ο αλγόριθμος δημιουργεί τα βέλτιστα μονοπάτια για όλα τα θέματα συμφόρησης που θα εμφανιστούν στο δίκτυο αποτελούμενα από κινητούς και στατικούς κόμβους. Τα βέλτιστα μονοπάτια είναι τα μονοπάτια που δημιουργούνται χρησιμοποιώντας όσο το δυνατόν λιγότερους κινητούς κόμβους. Στο κεφάλαιο αυτό περιγράφεται η δομή του αλγορίθμου, παρουσιάζεται τα διαγράμματα ροής δεδομένων του αλγορίθμου, και ακολούθως περιγράφεται αναλυτικά ο αλγόριθμος. Τέλος περιγράφονται μερικά παραδείγματα εκτέλεσης του.

5.1 Εισαγωγή

Ο Global MobileCC είναι ένας αλγόριθμος ελέγχου συμφόρησης ο οποίος δεν μπορεί να υλοποιηθεί σε ένα πραγματικό δίκτυο ασύρματων δικτύων αισθητήρων λόγω του ότι χρειάζεται εκ των προτέρων να ξέρει όλα τα θέματα συμφόρησης που θα παρουσιαστούν στο δίκτυο. Ο Global MobileCC είναι ο καλύτερος αλγόριθμος που μπορεί να σχεδιαστεί για να δημιουργεί εναλλακτικά μονοπάτια, όταν αποτυγχάνει ο αλγόριθμος ελέγχου συμφόρησης που χρησιμοποιείται να δημιουργήσει εναλλακτικά μονοπάτια για την μετάδοση των

δεδομένων που δημιουργούνται από τους πηγαίους κόμβους στον κόμβο προορισμού.

Ο Global MobileCC μπορεί να χρησιμοποιηθεί ως επέκταση οποιουδήποτε αλγορίθμου ελέγχου συμφόρησης. Σημειώνεται ότι ο Global MobileCC δεν αντικαταστεί τους υπάρχον αλγόριθμους ελέγχου τοπολογίας ή τους αλγόριθμους ελέγχου συμφόρησης ή τους αλγόριθμους δρομολόγησης, αλλά τρέχει παράλληλα με αυτούς. Ο Global MobileCC δημιουργεί εναλλακτικά μονοπάτια τοποθετώντας κινητούς κόμβους αισθητήρες, σε περίπτωση υπερφόρτωσης του δικτύου και αποτυχίας του αλγόριθμου ελέγχου συμφόρησης που χρησιμοποιείται ήδη να δημιουργήσει εναλλακτικά μονοπάτια. Ο αλγόριθμος θεωρεί ότι ένας ικανοποιητικός αριθμός κινητών κόμβων αισθητήρων, ανάλογα με το μέγεθος του δικτύου, είναι τοποθετημένοι αρχικά κοντά στον κόμβο προορισμού σε κατάσταση ύπνωσης (sleep state) ώστε να μην σπαταλείται η ενέργεια τους.

Ο Global MobileCC δε λύνει μεμονωμένα κάθε πρόβλημα συμφόρησης, αλλά επιλύει ολοκληρωτικά όλα τα θέματα συμφόρησης για αυτό το λόγο κατά τη διάρκεια του αλγορίθμου δεν τοποθετείται κανένας κινητός κόμβος, αλλά όλοι οι κινητοί κόμβοι τοποθετούνται στο τέλος της εκτέλεσης του γιατί ο αλγόριθμος ελέγχει αν είναι δυνατόν με την αλλαγή του σημείου που θα τοποθετηθεί ο κινητός κόμβος να συνεχίσει να εξυπηρετεί τους κόμβους που εξυπηρετούσε ήδη αλλά ταυτόχρονα να εξυπηρετεί και κάποιους επιπλέον κόμβους.

Επίσης λόγω του γεγονός ότι ο αλγόριθμος εξετάζεται καθολικά όλα τα θέματα συμφόρησης που θα εμφανιστούν στο δίκτυο, σε αντίθεση με τους δυο προηγούμενους αλγορίθμους, ελέγχει αν μόνο ένας κινητός κόμβος μπορεί να τοποθετηθεί σε τέτοια θέση ώστε να επιλύει δυο ή περισσότερα θέματα συμφόρησης ταυτόχρονα. Άρα σε περιπτώσεις όπου πολλά θέματα συμφόρησης μπορούν να επιλυθούν τοποθετώντας ένα μόνο κινητό κόμβο σε κατάλληλη θέση, ο Global MobileCC δε σπαταλάει επιπλέον κινητούς κόμβους, όπως τους δυο προηγούμενους αλγορίθμους, αφού όπως προαναφέρθηκε

επιλύει καθολικά όλα τα θέματα συμφόρησης δημιουργώντας τα βέλτιστα μονοπάτια με κινητούς και στατικούς κόμβους.

Ο αλγόριθμος Global MobileCC αποτελείται από δυο φάσεις. Κατά την πρώτη φάση τοποθετούνται όσοι κινητοί κόμβοι χρειάζονται, έτσι ώστε μερικοί από τους γείτονες των κόμβων που είναι σε συμφόρηση να στέλνουν τα δεδομένα τους σε κάποιο κινητό κόμβο. Στη δεύτερη φάση του αλγορίθμου τοποθετούνται όσοι κινητοί κόμβοι χρειάζονται έτσι ώστε να δημιουργηθούν τα βέλτιστα μονοπάτια, χρησιμοποιώντας δηλαδή όσο το δυνατόν λιγότερους κινητούς κόμβους, από τους κινητούς κόμβους που τοποθετήθηκαν στην πρώτη φάση στον κόμβο προορισμού. Τα μονοπάτια αυτά αποτελούνται από κινητούς και στατικούς κόμβους.

5.2 Περιγραφή της Δομής του Αλγορίθμου

Ο αλγόριθμος Global MobileCC αποτελείται από τα εξής βήματα:
1) Εύρεση για κάθε κόμβο που βρίσκεται σε συμφόρηση τους κόμβους που στέλνουν τα δεδομένα τους σε αυτόν .

2) Εύρεση θέσεων για να τοποθετηθούν οι κινητοί κόμβοι για να δημιουργηθούν τα βέλτιστα μονοπάτια από τους γείτονες των κόμβων που είναι σε συμφόρηση στον κόμβο προορισμού και εύρεση των κόμβων που θα στέλνουν τα δεδομένα τους σε κάθε κινητό κόμβο.

> 2Α) Φάση Α – Εύρεση των θέσεων των κινητών κόμβων που θα εξυπηρετούν τους γείτονες των κόμβων που είναι σε συμφόρηση.
>
> 2Β) Φάση Β – Δημιουργία μονοπατιών χρησιμοποιώντας στατικούς και κινητούς κόμβους, από τους κινητούς κόμβους που τοποθετήθηκαν στην προηγούμενη φάση στον κόμβο προορισμού.

5.3 Αναλυτική Περιγραφή του Αλγορίθμου

Πιο κάτω περιγράφεται βηματικά η λειτουργία του αλγορίθμου για την επίλυση όλων των θεμάτων συμφόρησης ενός δικτύου.

1) Για κάθε κόμβο που είναι σε συμφόρηση βρίσκει τους κόμβους που στέλνουν τα δεδομένα τους σε αυτόν
Στη συνέχεια ο αλγόριθμος βρίσκει για κάθε κόμβο που βρίσκεται σε συμφόρηση (congested node) τους γείτονες του που στέλνουν τα δεδομένα τους σε αυτόν, δηλαδή τους κόμβους που ο πατέρας τους είναι ο συγκεκριμένος congested node.

2) Εύρεση των κατάλληλων θέσεων για να τοποθετηθούν οι κινητοί κόμβοι για να δημιουργηθούν τα βέλτιστα μονοπάτια από τους γείτονες των κόμβων που είναι σε συμφόρηση στον κόμβο προορισμού και εύρεση των κόμβων που θα στέλνουν τα δεδομένα τους σε κάθε κινητό κόμβο

Η επόμενη φάση είναι η επιλογή των κατάλληλων θέσεων για να τοποθετηθούν οι κινητοί κόμβοι που είναι απαραίτητοι ώστε να δημιουργηθούν τα βέλτιστα μονοπάτια και η επιλογή των κόμβων που θα στέλνουν τα δεδομένα τους σε κάθε ένα από τους κινητούς κόμβους που θα τοποθετηθούν. Ο κύριος στόχος του αλγορίθμου είναι να τοποθετηθούν οι λιγότερο δυνατοί κινητοί κόμβοι, σε αντίθεση με τους δυο προηγούμενους αλγορίθμους οι οποίοι χρησιμοποιούσαν μια αμετάβλητη μέθοδο για την αντιμετώπιση κάθε θέματος συμφόρηση, αλλά ταυτόχρονα για κάθε θέμα συμφόρησης οι κόμβοι που έστελναν τα δεδομένα τους στον κόμβο που είναι σε συμφόρηση και θα τα στέλνουν σε κάποιον κινητό κόμβο να έχουν συνολικό sending rate όσο το additional_resources. Ο δευτερεύων στόχος είναι να αλλάξουν πατέρα όσο το δυνατό λιγότεροι κόμβοι.

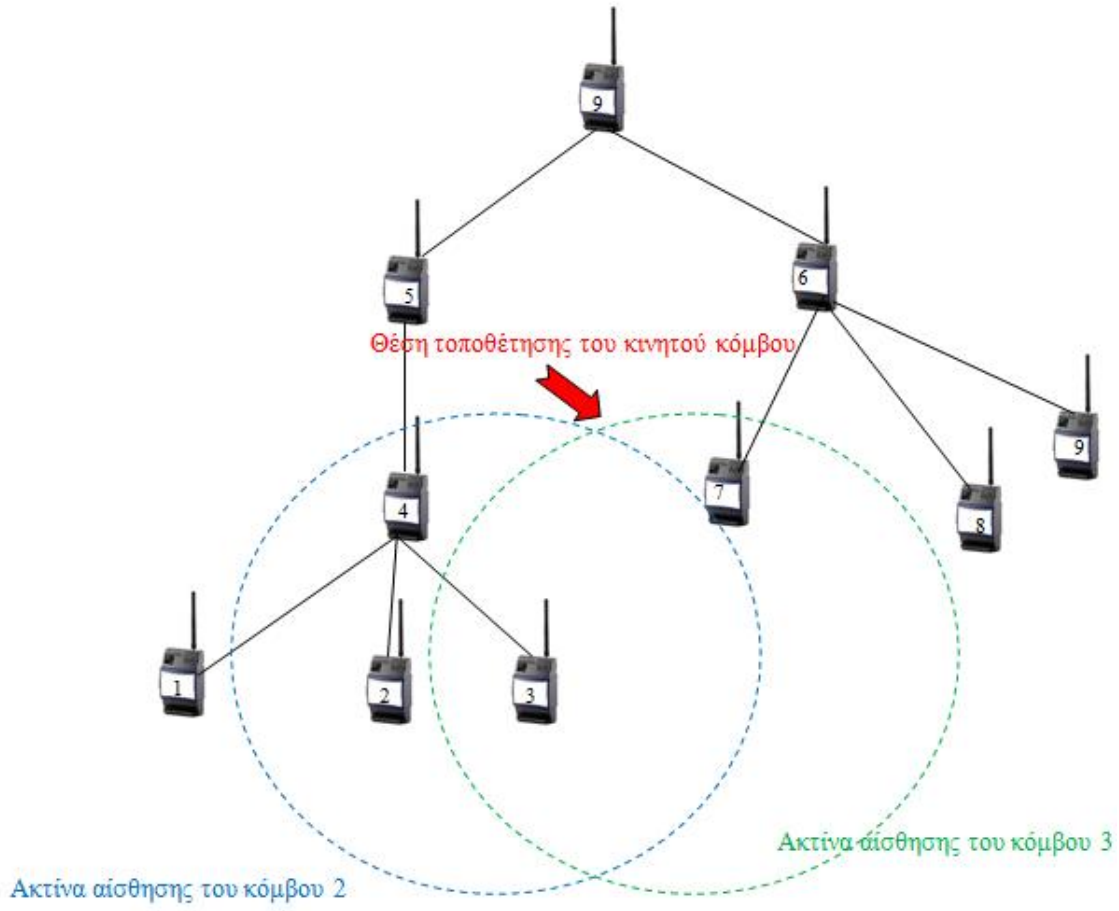
2Α) Φάση Α – Εύρεση των θέσεων των κινητών κόμβων που θα εξυπηρετούν τους γείτονες των κόμβων που είναι σε συμφόρηση

Οι κόμβοι που είναι σε συμφόρηση ταξινομούνται σε φθίνουσα σειρά με βάση την απόσταση τους από τον κόμβο προορισμού, δηλαδή ο πρώτος κόμβος είναι ο κόμβος που είναι πιο μακριά από τον κόμβο προορισμού και ο τελευταίος είναι ο κόμβος που είναι πιο κοντά στον κόμβο προορισμού.

Αρχικά για τον πρώτο κόμβο που είναι σε συμφόρηση, δηλαδή τον κόμβο που είναι σε συμφόρηση και είναι πιο μακριά από τον κόμβο προορισμού, σε σχέση με τους υπόλοιπους κόμβους που είναι σε συμφόρηση, καλείται ο αλγόριθμος Dynamic MobileCC (ο οποίος αναλύεται στο κεφάλαιο 3) για την εύρεση της θέση που θα τοποθετηθεί ο πρώτος κινητός κόμβος (η οποία μπορεί να αλλάξει) και για την εύρεση του υποσύνολου των κόμβων που στέλνουν τα δεδομένα τους στον κόμβο που είναι σε συμφόρηση με rate μεγαλύτερο από το additional_resources, και πλέον θα εξυπηρετούνται από τον κινητό κόμβο. Ο αλγόριθμος Dynamic MobileCC εκτελείται με μια μικρή διαφοροποίηση ώστε να κρατά όλα τα υποσύνολα των κόμβων με συνολικό rate μεγαλύτερο από το additional_resources που έχουν τουλάχιστον ένα κοινό σημείο και άρα μπορούν να εξυπηρετούνται από τον κινητό κόμβο αν αλλάξει βέβαια η θέση του.

Στη συνέχεια για τον επόμενο κόμβο που είναι σε συμφόρηση με βάση τη ταξινόμηση που αναφέρεται πιο πάνω, ελέγχεται για κάθε κινητό κόμβο που χρησιμοποιείται ήδη και έχει διαθέσιμο rate, αν υπάρχει κοινό σημείο τομής των κόμβων που εξυπηρετεί ο κινητός κόμβος και ενός ή περισσότερων κόμβων που στέλνουν τα δεδομένα τους στον κόμβο που είναι σε συμφόρηση. Η διαδικασία αυτή συνεχίζεται μέχρι να ελεγχτούν όλοι οι κινητοί κόμβοι ή το συνολικό rate των κόμβων που στέλνουν τα δεδομένου τους στον κόμβο που είναι σε συμφόρηση και μπορούν να εξυπηρετούνται από τους κινητούς κόμβους να γίνει μεγαλύτερο από το additional_resources. Αν ελεγχτούν όλοι οι κινητοί κόμβοι και το συνολικό rate των κόμβων που στέλνουν τα δεδομένα τους στον κόμβο που είναι σε συμφόρηση και μπορούν να τα στέλνουν στους κινητούς κόμβους που χρησιμοποιούνται ήδη είναι μικρότερο από το additional_resources, τότε αναπόφευκτα πρέπει να τοποθετηθεί ένας άλλος κινητός κόμβος σε κατάλληλη θέση ώστε να εξυπηρετεί κάποιους από τους κόμβους που στέλνουν τα δεδομένα τους στον κόμβο που είναι σε συμφόρηση.

Για τον σκοπό αυτό χρησιμοποιείται ο αλγόριθμος Dynamic MobileCC με μια μικρή διαφοροποίηση ώστε να κρατά όλα τα υποσύνολα των κόμβων με συνολικό rate μεγαλύτερο από το additional_resources που έχουν τουλάχιστον ένα κοινό σημείο και άρα μπορούν να εξυπηρετούνται από τον κινητό κόμβο αν αλλάξει βέβαια η θέση του.
Αυτή η διαδικασία γίνεται για κάθε θέμα συμφόρησης (εκτός από το πρώτο).



Σχήμα 5.1 Επεξήγηση παραδείγματος εκτέλεσης της Φάσης Α του αλγορίθμου Global MobileCC

Επεξήγηση παραδείγματος στο Σχήμα 5.1:
Στο πιο πάνω παράδειγμα παρατηρούμε ότι ο κόμβος 4 και ο κόμβος 6 είναι σε συμφόρηση γιατί τα πακέτα που λαμβάνουν είναι περισσότερα από τα πακέτα που μπορούν να στείλουν. Ας θεωρήσουμε ότι το εύρος αίσθησης όλων των κόμβων είναι 2.5 . Ο κόμβος που είναι σε συμφόρηση και είναι πιο μακριά από τον κόμβο προορισμού είναι ο κόμβος 4. Εφόσον δεν έχει τοποθετηθεί κανένας

κινητός κόμβος, καλείται ο αλγόριθμος Dynamic MobileCC για τον κόμβο 4 που είναι σε συμφόρηση. Ας θεωρήσουμε ότι κανένας από τους κόμβους 1,2, 3 δεν έχουν sending rate μεγαλύτερο από το additional_resources, τότε εξετάζονται τα υποσύνολα μεγέθους 2 παρατηρούμε ότι τα πιθανά υποσύνολα κόμβων που μπορεί να εξυπηρετεί ο κινητός κόμβος είναι τα {1,2}, {1,3}, {2,3}.
Ας υποθέσουμε ότι το συνολικό sending rate όλων των υποσυνόλων είναι μεγαλύτερο από το additional resources, και έστω ότι ο κόμβος 1 βρίσκεται στη θέση (0, 0) , ο κόμβος 2 στη θέση (1.5, 0) , ο κόμβος 3 στη θέση (4, 0), ο κόμβος 4 στη θέση (2, 2), ο κόμβος 5 στη θέση (2, 4), ο κόμβος 6 στη θέση (5, 4), ο κόμβος 7 βρίσκεται στη θέση (4.5, 2) , ο κόμβος 8 στη θέση (6.5, 2) , ο κόμβος 9 που είναι ο κόμβος προορισμού είναι στη θέση (4,0) και ο κόμβος 9 στη θέση (3.5, 6).
Ο Dynamic MobileCC θα έχει ως αποτέλεσμα την τοποθέτηση ενός κινητού κόμβου στο σημείο (0.75 , 2.38) για να εξυπηρετούνται οι κόμβοι 1 και 2 αλλά παράλληλα θα δώσει και την πληροφορία ότι ο κινητός κόμβος μπορεί να εξυπηρετεί οποιοδήποτε από τα υποσύνολα {1,2}, {1,3}, {2,3}.

Στη συνέχεια ελέγχεται το δεύτερο θέμα συμφόρησης που αφορά τον κόμβο 6. Ας θεωρήσουμε ότι ο κόμβος 7 στέλνει στον κόμβο 6 με rate μεγαλύτερο από το additional_resources, και άρα θα πρέπει να μεταδίδει τα δεδομένα του στον κόμβο προορισμού μέσω κάποιου κινητού κόμβου. Εξετάζουμε αν μπορεί ο κινητός κόμβος που τοποθετήθηκε να συνεχίσει να εξυπηρετεί κάποιο από τα υποσύνολα κόμβων που εξυπηρετούσε αλλά παράλληλα να εξυπηρετεί επίσης τον κόμβο 7. Παρατηρούμε ότι στη θέση (2.291 , 1) όπου αποφασίστηκε να τοποθετηθεί ο κινητός κόμβος δεν μπορεί να επικοινωνήσει με τον κόμβο 7, όμως υπάρχει σημείο τομής μεταξύ του υποσυνόλου {2, 3} και του κόμβου 7, το οποίο είναι το (2.29, 3). Έτσι αποφασίζεται ο κινητός κόμβος να τοποθετηθεί στη θέση (2.29, 3) για να εξυπηρετεί τους κόμβους {2, 3, 7}. Έτσι λύνονται τα δυο θέματα συμφόρησης με τη χρήση ενός μόνο κινητού κόμβου. Ο αλγόριθμος Dyanmic MobileCC και Direct Path MobileCC επειδή λύνουν κάθε πρόβλημα συμφόρησης μεμονωμένα σε τέτοια περίπτωση θα χρησιμοποιούσαν τουλάχιστον δυο κινητούς κόμβους.

2Β) Φάση Β – Δημιουργία μονοπατιών χρησιμοποιώντας στατικούς και κινητούς κόμβους, από τους κινητούς κόμβους που τοποθετήθηκαν στην προηγούμενη φάση στον κόμβο προορισμού.

Για τη δημιουργία των βέλτιστων μονοπατιών στον κόμβο προορισμού, από κάθε κινητό κόμβο που τοποθετήθηκε στην Φάση Α και δεν είναι στο εύρος αίσθησης του κόμβου προορισμού, χρησιμοποιείται ένας αλγόριθμος δυναμικού προγραμματισμού. Ο αλγόριθμος ονομάζεται dynamic_agorithm find_cost_for_paths και εκτελείται για κάθε κινητό κόμβο. Ο στόχος αυτού του αλγορίθμου είναι να δημιουργηθεί ένα μονοπάτι από τον κινητό κόμβο που εξετάζεται στον κόμβο προορισμού, τοποθετώντας όσο λιγότερους κινητούς κόμβους είναι δυνατόν. Ο δευτερεύων στόχος του αλγορίθμου είναι τα μονοπάτια που θα δημιουργηθούν να έχουν όσο το δυνατόν λιγότερα βήματα (hops). Ο δυναμικός προγραμματισμός είναι μία μέθοδος που είναι εφαρμόσιμη όταν τα υποπροβλήματα που υπάρχουν δεν είναι ανεξάρτητα μεταξύ τους. Οι αλγόριθμοι δυναμικού προγραμματισμού, επιλύουν μία φορά κάθε υποπρόβλημα και αποθηκεύουν αυτή τη λύση σε έναν πίνακα, στον οποίον θα καταφεύγει κάθε φορά που συναντά το συγκεκριμένο πρόβλημα, ώστε να μην χρειάζεται να το λύσει ξανά. Αποτελεί μία πολύ ισχυρή τεχνική για αλγοριθμική επίλυση προβλημάτων [36].

Το πρόβλημα που πρέπει να λυθεί είναι η εύρεση του μονοπατιού με το μικρότερο κόστος, δηλαδή με τους λιγότερους κινητούς κόμβους, από τον κινητό κόμβο που εξετάζεται στον κόμβο προορισμού. Τα υποπρόβλημα που πρέπει να επιλυθούν είναι η δημιουργία ενός μονοπατιού από κάθε κόμβο που είναι σε μικρότερο επίπεδο από τον κινητό κόμβο, αρχίζοντας από τους κόμβους που επικοινωνούν άμεσα με τον κόμβο προορισμού και συνεχίζοντας με τους γείτονες των κόμβων που επικοινωνούν άμεσα με τον κόμβο προορισμού, μέχρι την επίλυση του υποπροβλήματος εύρεσης του μονοπατιού με το μικρότερο κόστος από τον κινητό κόμβο στον κόμβο προορισμού. Για την επίλυση ενός υποπροβλήματος χρησιμοποιείται ο αριθμός των βημάτων (hops) και ο αριθμός των κινητών κόμβων που υπολογίστηκαν κατά την επίλυση προηγούμενων υποπροβλημάτων, δηλαδή ο αριθμός των βημάτων (hops) και ο αριθμός των κινητών κόμβων που χρειάζονται για τη δημιουργία ενός

μονοπατιού από έναν κόμβο σε μικρότερο επίπεδο στον κόμβο προορισμού. Για παράδειγμα για την επίλυση του υποπροβλήματος εύρεσης του βέλτιστου μονοπατιού από κάποιο άμεσο γείτονα του κόμβου προορισμού στον κόμβο προορισμού, χρησιμοποιείται ο αριθμός των βημάτων (hops) και ο αριθμός των κινητών κόμβων που χρειάζονται για να μεταβείς από τον κόμβο προορισμού στον κόμβο προορισμού.

Στον αλγόριθμο δυναμικού προγραμματισμού (dynamic_agorithm find_cost_for_paths) χρησιμοποιείται ο πίνακας *M* ο οποίος κρατά σε κάθε θέση *vi* τον αριθμό των κινητών κόμβων που χρειάζονται για να δημιουργηθεί το βέλτιστο μονοπάτι από τον κόμβο *vi* στον κόμβο προορισμού. Επίσης χρησιμοποιείται ο πίνακας *C* ο οποίος κρατά σε κάθε θέση *vi* τον αριθμό των βημάτων (hops) του βέλτιστου μονοπατιού από τον κόμβο *vi* στον κόμβο προορισμού. Επίσης χρησιμοποιείται ο πίνακας *B* ο οποίος κρατά σε κάθε θέση *vi* τον προηγούμενο κόμβο (πριν από τον κόμβο *vi*) στο βέλτιστο μονοπάτι από τον κόμβο *vi* στον κόμβο προορισμού. Ο πίνακας *C* και ο πίνακας *M* αρχικοποιούνται με άπειρο σε κάθε θέση τους εκτός από τις θέσεις που αναπαριστούν τον κόμβο προορισμού, οι οποίες αρχικοποιούνται με μηδέν γιατί ο αριθμός των βημάτων (hops) για να μεταβείς από τον κόμβο προορισμού στον εαυτό του είναι μηδέν, παρόμοιος και ο αριθμός των κινητών κόμβων που χρειάζονται είναι μηδέν.

Κατά τον αλγόριθμο dynamic_agorithm_find_cost_for_paths εκτελείται για κάθε κόμβο ο οποίος είναι σε μικρότερο επίπεδο από τον κινητό κόμβο που εξετάζεται και δεν είναι σε συμφόρηση ο αλγόριθμος find_best_ neighbor, αρχίζοντας από τους κόμβους που είναι πιο κοντά στον κόμβο προορισμού (πιο λίγα βήματα-hops από τον κόμβο προορισμού) και συνεχίζοντας με τους κόμβους που είναι πιο μακριά (πιο πολλά hops από τον κόμβο προορισμού). Ο σκοπός του αλγορίθμου find_best_neighbor είναι να βρει το γείτονα του κόμβου που εξετάζεται, αν υπάρχουν γείτονες, για τον οποίο έχει υπολογιστεί το μικρότερο κόστος για τη δημιουργία μονοπατιού από αυτόν στον κόμβο προορισμού. Γείτονες ενός κόμβου θεωρείται επίσης οποιοσδήποτε κινητός κόμβος για τον οποίο υπάρχει τουλάχιστον ένα σημείο τομής ανάμεσα στους

κόμβους που εξυπηρετεί ο κινητός κόμβος και των κόμβων που εξυπηρετεί ο κόμβος που εξετάζεται, γιατί έτσι τοποθετώντας τον κινητό κόμβο σε αυτό το σημείο τομής θα συνεχίσει να εξυπηρετεί τους κόμβους που εξυπηρετούσε και παράλληλα θα εξυπηρετεί επίσης τον κόμβο που εξετάζεται. Δηλαδή *M[vi] = min(M[vj]),* όπου *vj* κάθε γείτονας του κόμβου *vi* (στατικός ή κινητός κόμβος). Τονίζεται ότι ένας κινητός κόμβος είναι γείτονας του κόμβου vi αν υπάρχει τουλάχιστον ένα σημείο τομής μεταξύ των κόμβων που εξυπηρετεί ο κόμβος *vi* και των κόμβων που εξυπηρετεί ο κινητός κόμβος, γιατί εφόσον ακόμη δεν τοποθετήθηκε ο κινητός κόμβος μπορεί ο κινητός κόμβος να τοποθετηθεί σε αυτό το σημείο τομής. Αν δεν έχει κανένα γείτονα ο κόμβος *vi*, τότε πρέπει να τοποθετηθεί ένας κινητός κόμβος στο εύρος αίσθησης του και να δημιουργηθεί ένα μονοπάτι από τον κόμβο i στον κόμβο προορισμού.

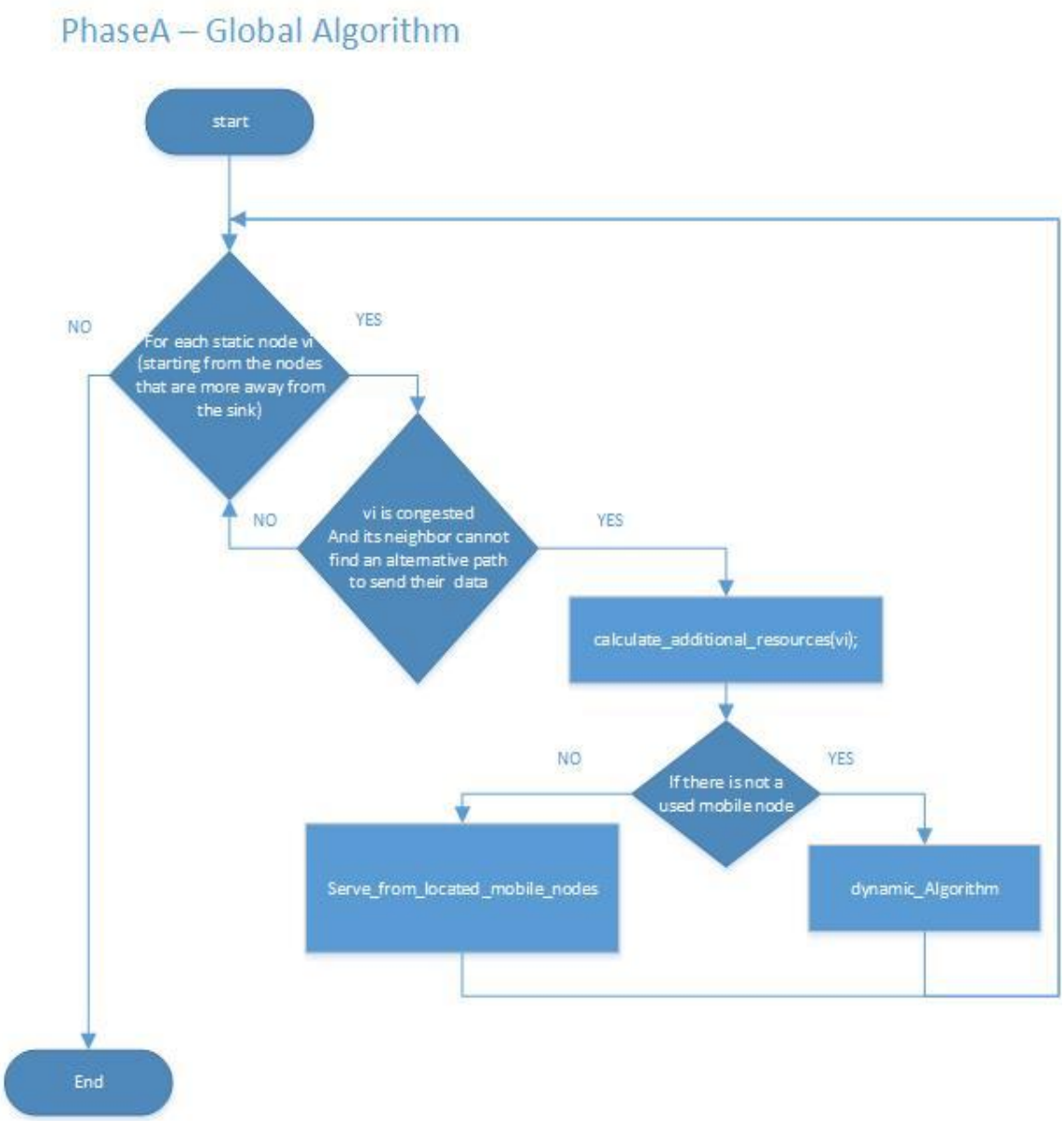
Ανάμεσα σε δυο μονοπάτια, το μονοπάτι με το μικρότερο κόστος ορίζεται ως το μονοπάτι που χρειάζεται τους λιγότερους κινητούς κόμβους και σε περίπτωση ίσου αριθμού κινητών κόμβων, ορίζεται ως το μονοπάτι με το μικρότερο αριθμό βημάτων (hops). Δηλαδή αν *M[vj] = M[vz]*, τότε *M[vi] = M[vz]* αν *C[vj]>C[vz]*, όπου vz και vj γείτονες του κόμβου *vi*.

Αν έχει τουλάχιστον ένα γείτονα ο κόμβος που εξετάζεται τότε ο αριθμός των βημάτων (hops) για τη δημιουργία του βέλτιστου μονοπατιού από τον κόμβο που εξετάζεται στον κόμβο προορισμού, είναι κατά ένα μεγαλύτερο από τον αριθμό των βημάτων (hops) του βέλτιστου μονοπατιού από το γείτονα του που χρειάζεται το μικρότερο αριθμό κινητών κόμβων για τη δημιουργία του μονοπατιού από αυτό στον κόμβο προορισμού. Ο αριθμός των κινητών κόμβων που χρειάζονται για τη δημιουργία του βέλτιστου μονοπατιού από τον κόμβο που εξετάζεται στον κόμβο προορισμού είναι ίσος με τον αριθμό των κινητών κόμβων που χρησιμοποιούνται για το βέλτιστο μονοπάτι από τον κόμβο γείτονα του κόμβο που εξετάζεται στον κόμβο προορισμού. Δηλαδή αν *vj* ο κόμβος που εξετάζεται και *vi* ο κόμβος γείτονας του κόμβου vj με το μικρότερο κόστος δημιουργίας μονοπατιού από αυτόν στον κόμβο προορισμού ανάμεσα στους γείτονες του κόμβου που εξετάζεται, τότε *C[vj] = C[vi] + 1* και *M[vj] = M[vi]*. Αν δεν έχει γείτονα ο κόμβος που εξετάζεται τότε πρέπει να τοποθετηθεί ένας

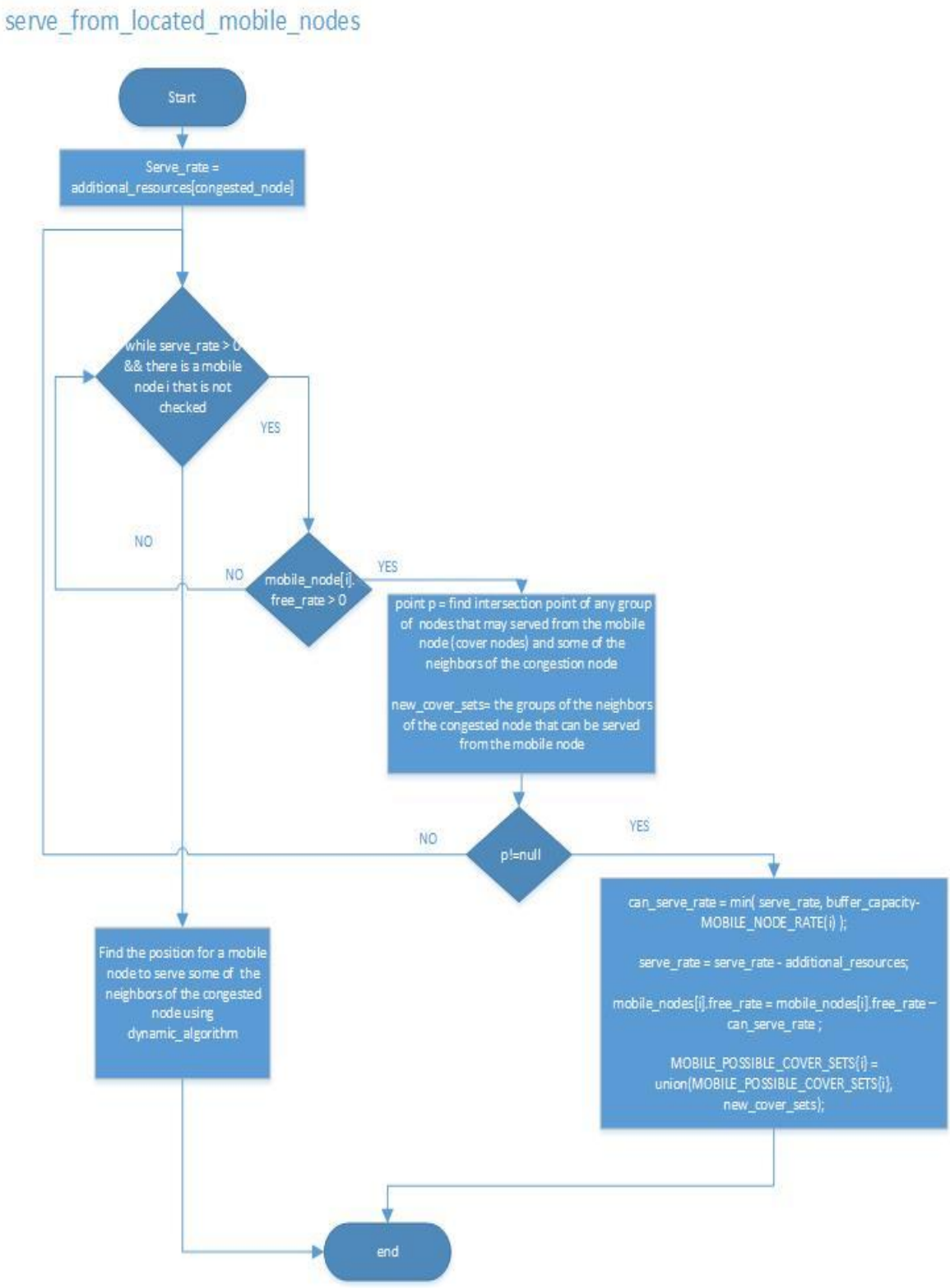
κινητός κόμβος στο εύρος αίσθησης του κόμβου που εξετάζεται. Όσο δεν υπάρχει κάποιος διαθέσιμος κόμβος για να επικοινωνήσει ο κινητός κόμβος που τοποθετείται θα πρέπει να τοποθετηθεί ένας άλλος κινητός κόμβος για να υπολογιστεί το κόστος για το συγκεκριμένο μονοπάτι.

Όταν υπολογιστούν τα μονοπάτια από τους κινητούς κόμβους που τοποθετηθήκαν στη Φάση Α στον κόμβο προορισμού, τοποθετούνται όλοι οι κινητοί κόμβοι στις κατάλληλες θέσεις για να δημιουργηθούν τα εναλλακτικά μονοπάτια.

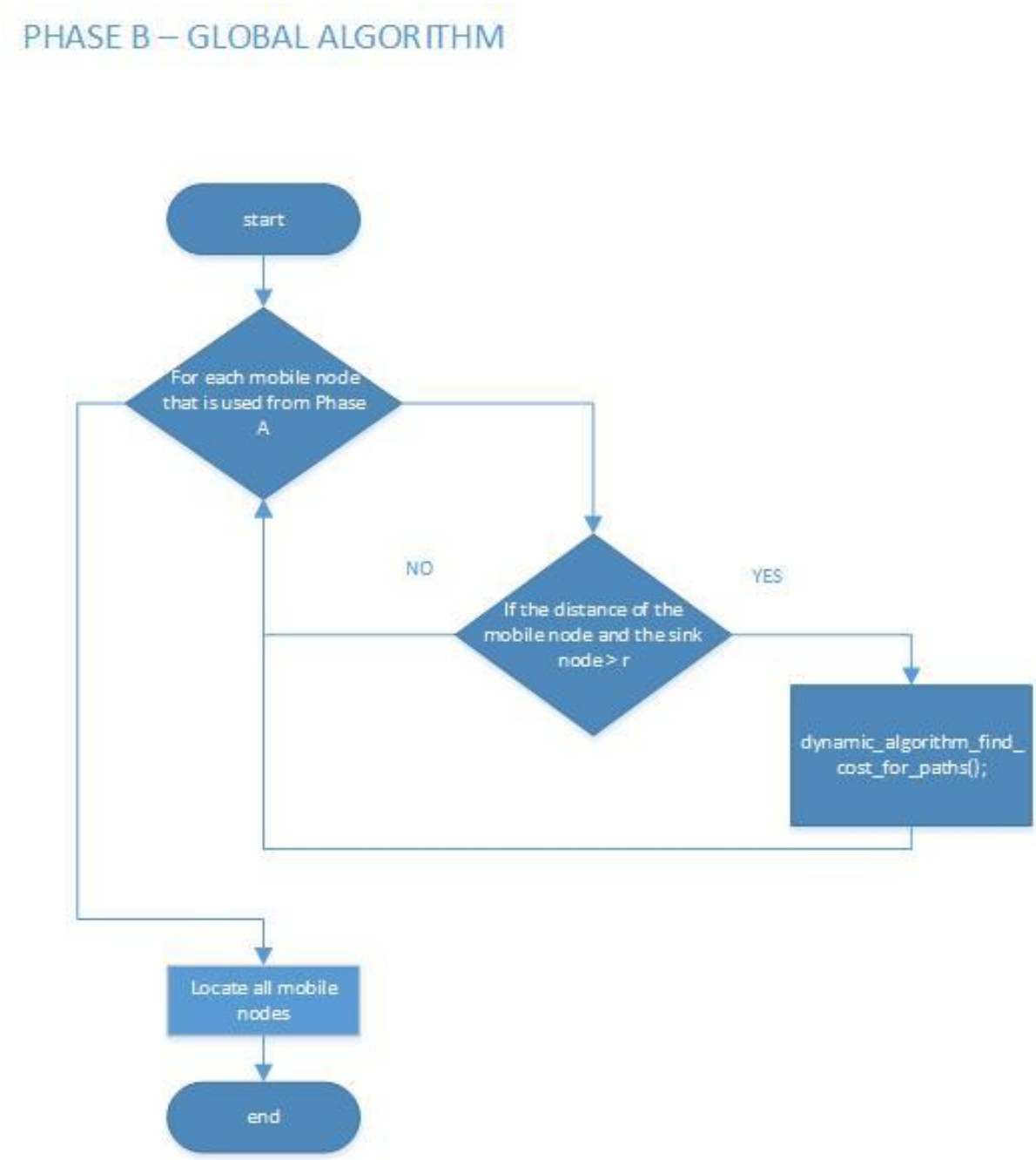
Συνέχεια παραδείγματος από το Σχήμα 5.1: Παρατηρούμε ότι κατά την εκτέλεση της Φάσης Β για το πιο πάνω παράδειγμα οι κόμβοι που είναι σε μικρότερο επίπεδο από τον κινητό κόμβο που θα τοποθετηθεί είναι οι κόμβοι 9, 5, 6. Αρχικά θα οριστεί ότι $M[9] = 0$ και $C[9]=0$ και για όλους τους υπόλοιπους κόμβους j θα οριστεί $M[j] = \infty$ και $C[j] = \infty$. Στη συνέχεια θα εξεταστούν οι κόμβοι 5, 6 . Παρατηρούμε ότι και για τους δυο κόμβους μπορεί να δημιουργηθεί μονοπάτι μέσω του κόμβου προορισμού, άρα $M[5]=M[6]=0$ και $C[5]=C[6]=1$.



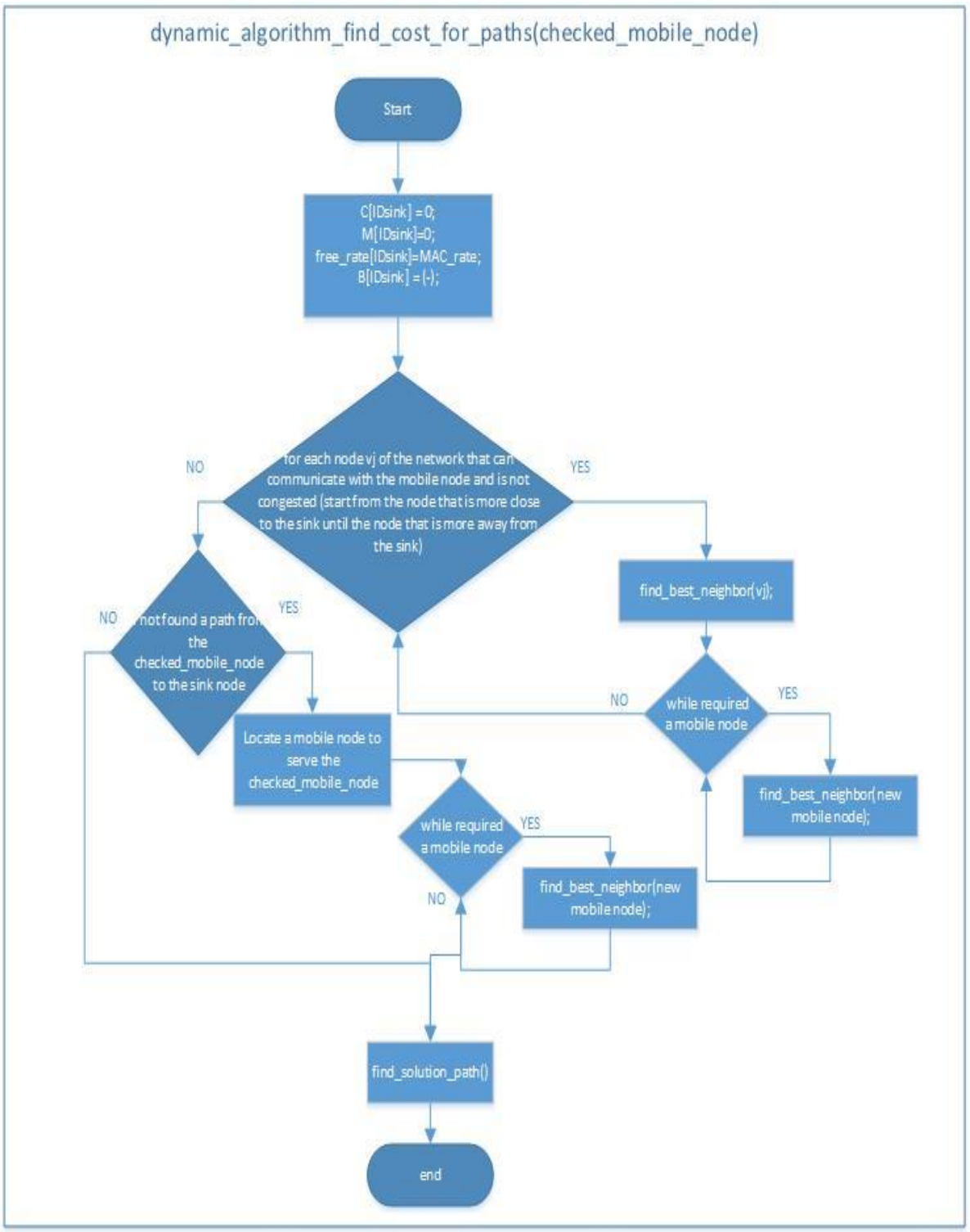
Σχήμα 5.2 Διάγραμμα Ροής (Flow Chart) της Φάσης Α του αλγορίθμου Global MobileCC



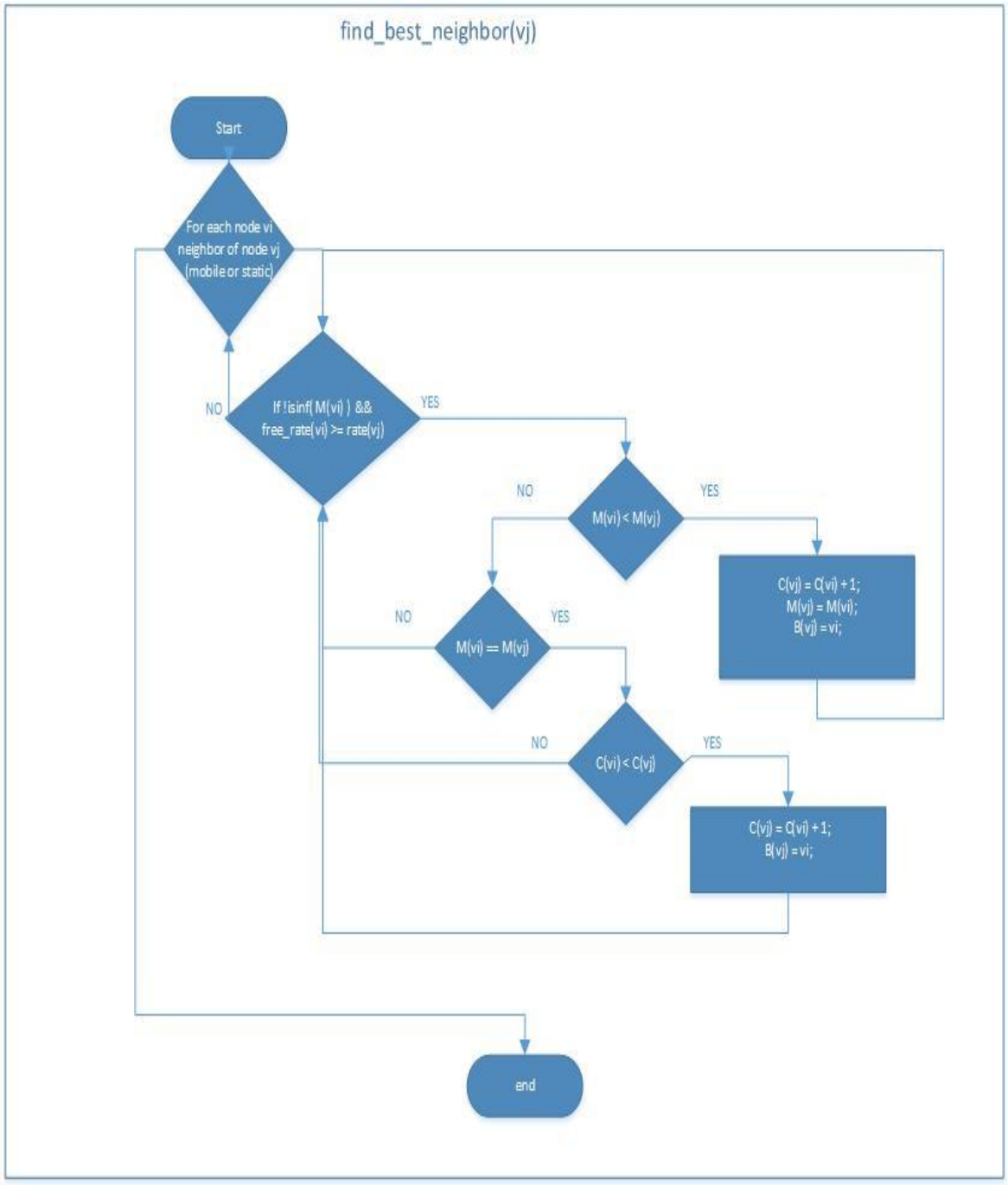
Σχήμα 5.3 Διάγραμμα Ροής (Flow Chart) της συνάρτησης Serve_from_located_mobile_nodes του αλγορίθμου Global MobileCC (Μέρος της Φάσης Α του αλγορίθμου)



Σχήμα 5.4 Διάγραμμα Ροής (Flow Chart) της Φάσης Β του αλγορίθμου Global MobileCC



Σχήμα 5.5 Διάγραμμα Ροής (Flow Chart) της συνάρτησης dynamic_agorithm_find_cost_for_paths της Φάσης Β του αλγορίθμου Global MobileCC



Σχήμα 5.6 Διάγραμμα Ροής (Flow Chart) της συνάρτησης find_best_neighbor(vj) της Φάσης Β του αλγορίθμου Global MobileCC

5.4 Απλά Παραδείγματα Εκτέλεσης του Αλγορίθμου

Ακολουθούν τέσσερα πολύ απλά παραδείγματα εκτέλεσης του αλγορίθμου Global MobileCC. Για όλα τα παραδείγματα που ακολουθούν το εύρος αίσθησης όλων των κόμβων είναι 2.5 .

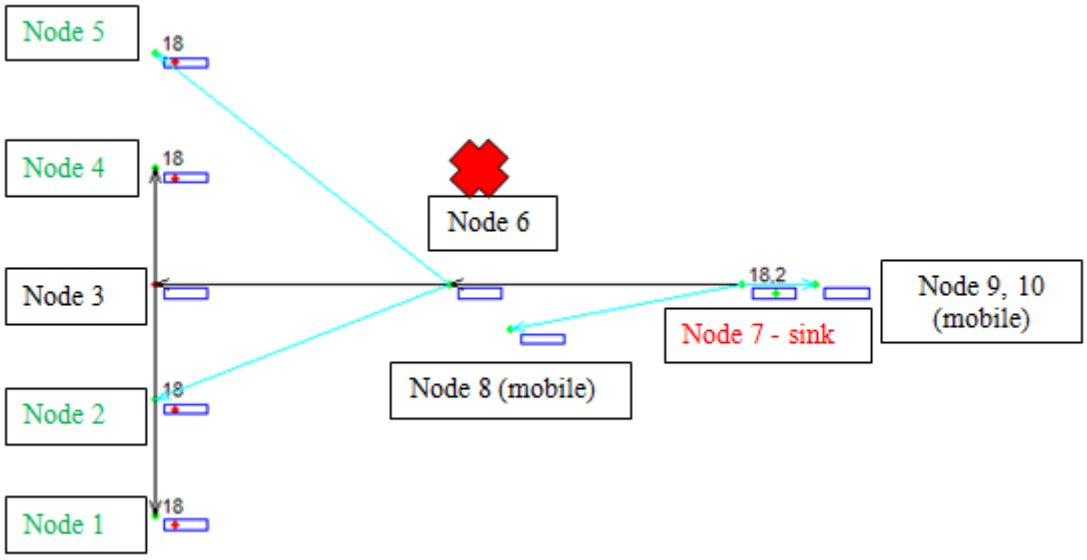
Παράδειγμα 1:

Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[0 1; 0 2; 0 3; 0 4; 0 5; 2 3; 4 3;4.5 3; 4.5 3; 4.5 3];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 7, ο οποίος βρίσκεται στη θέση (4, 3).

Ο κινητός κόμβος 8, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (4.5, 3), μετά την εκτέλεση του αλγορίθμου Global MobileCC τοποθετείται στη θέση (2.4254, 2.6063).



Σχήμα 5.7 Αναπαράσταση του δικτύου μετά την εκτέλεση του αλγορίθμου Global MobileCC.

Οι κόμβοι 8, 9, 10 είναι mobile nodes και είναι τοποθετημένοι αρχικά κοντά στο κόμβο προορισμού.

Φάση Α:

Ο κόμβος 6 είναι σε συμφόρηση (congested) και έτσι επειδή δεν έχει ήδη τοποθετηθεί κανένας κινητός κόμβος, καλείται ο dynamic algorithm κατά την εκτέλεση του οποίου τοποθετείται ο mobile node 8 στη θέση (2.4254, 2.6063) για να εξυπηρετεί τον κόμβο 2 που είναι γείτονας του κόμβου 6 και δεν μπορεί να βρει εναλλακτική διαδρομή.

Φάση Β:

Ο mobile node 8 είναι σε απόσταση μικρότερη από το range του από το κόμβο προορισμού έτσι δεν χρειάζεται να δημιουργηθεί μονοπάτι από το mobile node 8 στον κόμβο προορισμού.

Παράδειγμα 2:

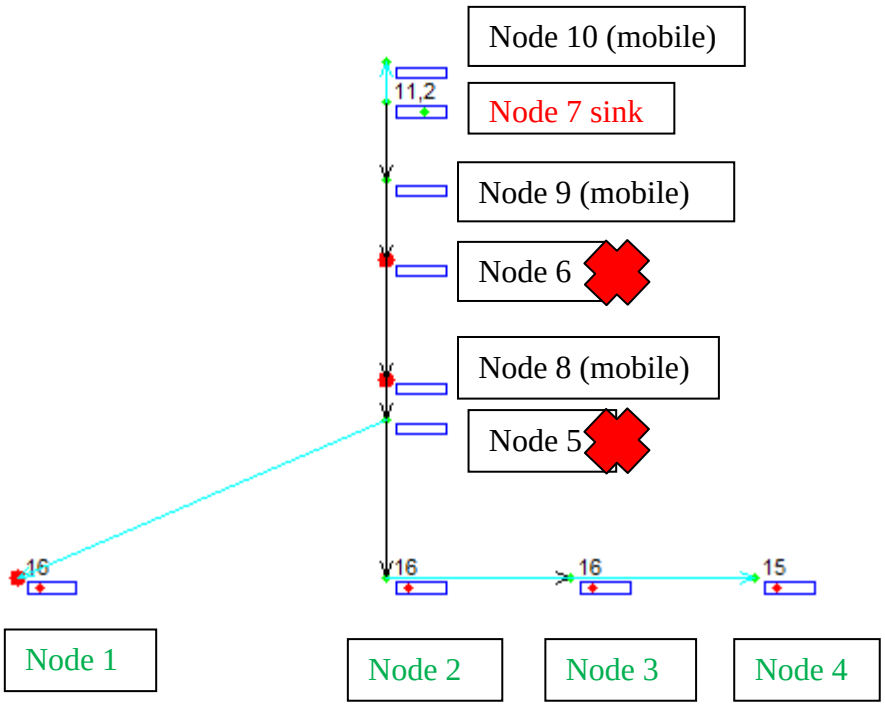
Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[0 0; 2 0; 3 0; 4 0; 2 2; 2 4; 2 6; 2 6.5; 2 6.5; 2 6.5];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 7, ο οποίος βρίσκεται στη θέση (2, 6).

Ο κινητός κόμβος 8, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (2, 6.5), μετά την εκτέλεση του αλγορίθμου Global MobileCC τοποθετείται στη θέση (2, 2.5).

Ο κινητός κόμβος 9, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (2, 6.5), μετά την εκτέλεση του αλγορίθμου Global MobileCC τοποθετείται στη θέση (2, 5).



Σχήμα 5.8 Αναπαράσταση του δικτύου μετά την εκτέλεση του αλγορίθμου Global MobileCC.

Οι κόμβοι 8, 9, 10 είναι κινητοί κόμβοι και είναι τοποθετημένοι αρχικά κοντά στον κόμβο προορισμού.

Φάση Α:

Οι κόμβοι 5, 6 είναι οι κόμβοι που είναι σε συμφόρηση (congested nodes) ταξινομημένοι σε φθίνουσα σειρά με βάση την απόσταση τους από τον κόμβο προορισμού.

Εφόσον δεν έχει τοποθετηθεί κανένας κινητός κόμβος αρχικά και ο κόμβος 5 είναι ο κόμβος που είναι πιο μακριά από το κόμβο προορισμού καλείται ο Dynamic MobileCC για τον κόμβο αυτό και το αποτέλεσμα του αλγορίθμου είναι να τοποθετηθεί ένας κινητός κόμβος στο σημείο p[2, 2.5] για να εξυπηρετεί τον κόμβο 2 που είναι γείτονας του κόμβου 5 και δεν μπορεί να βρει εναλλακτική διαδρομή.

Όσο αφορά τον κόμβο 6 εφόσον υπάρχει τουλάχιστον ένας από τους γείτονες του που του στέλνει δεδομένα (ο κόμβος 5) που είναι σε απόσταση μικρότερη από το range από το σημείο που θα τοποθετηθεί ο κινητός κόμβος που θα εξυπηρετεί τον κόμβο 5 και ο κινητός κόμβος έχει διαθέσιμο rate, δεν χρειάζεται

να γίνει κάποια αλλαγή του σημείου ή να τοποθετηθεί κάποιος επιπλέον κινητός κόμβος.

Φάση Β:

Δεν υπάρχει κανένας κόμβος που να μην είναι σερ συμφόρηση για να στέλνει τα δεδομένα του ο κινητός κόμβος που θα τοποθετηθεί, για αυτό πρέπει να δημιουργηθεί μονοπάτι από τον κινητό κόμβο στον κόμβο προορισμού. Για να δημιουργηθεί αυτό το μονοπάτι τοποθετείται ένας άλλος κινητός κόμβος στο σημείο q[2, 5] (είναι το σημείο που είναι όσο το δυνατόν πιο κοντά στον κόμβο προορισμού αλλά στο παράλληλα στο εύρος αίσθησης του κινητού κόμβου που θα τοποθετηθεί στο σημείο p). Ο mobile node που θα τοποθετηθεί στο σημείο q μπορεί να επικοινωνήσει απευθείας με τον κόμβο προορισμού.

Το μονοπάτι αυτό είναι βέλτιστο εφόσον δεν υπάρχει άλλο μονοπάτι, έτσι τοποθετούνται οι κινητοί κόμβοι 8 και 9 στα σημεία p και q αντίστοιχα.

Παράδειγμα 3:

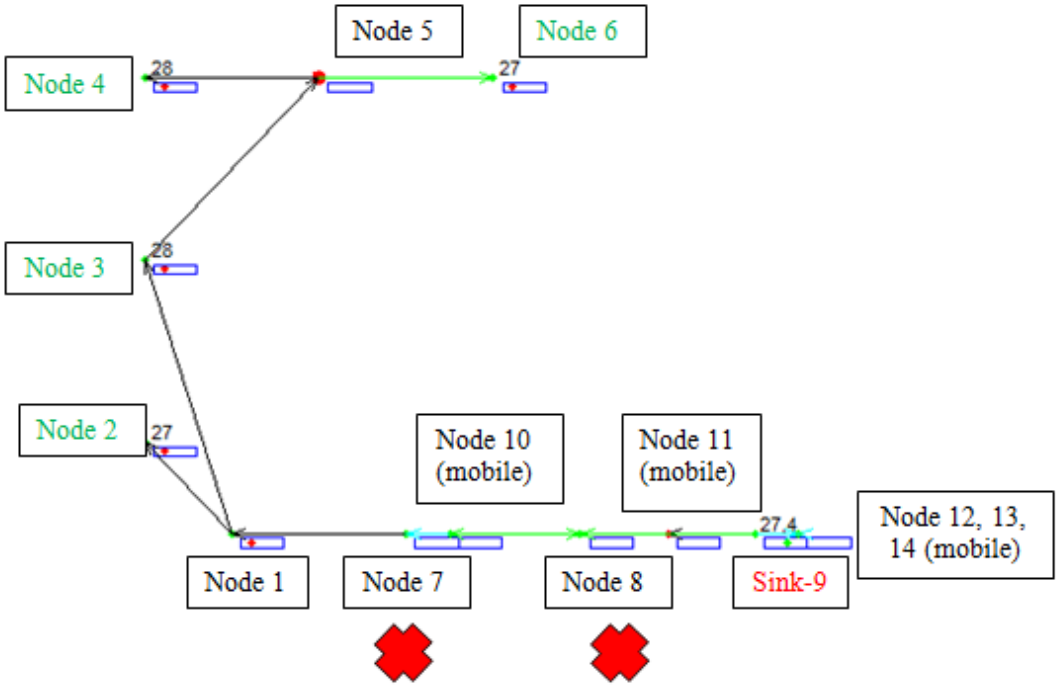
Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[1 2; 0 3; 0 5; 0 7; 2 7; 4 7; 3 2; 5 2; 7 2; 7.5 2; 7.5 2; 7.5 2; 7.5 2; 7.5 2];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 9, ο οποίος βρίσκεται στη θέση (7, 2).

Ο κινητός κόμβος 10, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (7.5, 2), μετά την εκτέλεση του αλγορίθμου Global MobileCC τοποθετείται στη θέση (3.5, 2).

Ο κινητός κόμβος 11, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (7.5, 2), μετά την εκτέλεση του αλγορίθμου Global MobileCC τοποθετείται στη θέση (6, 2).



Σχήμα 5.9 Αναπαράσταση του δικτύου μετά την εκτέλεση του αλγορίθμου Global MobileCC.

Οι κόμβοι 10, 11, 12, 13, 14 είναι mobile nodes και είναι τοποθετημένοι αρχικά κοντά στον κόμβο προορισμού.

Φάση Α:

Οι κόμβοι 7, 8 είναι σε συμφόρηση (congested node) ταξινομημένοι σε φθίνουσα σειρά με βάση την απόσταση τους από τον κόμβο προορισμού.

Εφόσον ο κόμβος 7 είναι ο κόμβος που είναι πιο μακριά από τον κόμβο προορισμού καλείται ο αλγόριθμος Dynamic MobileCC για τον κόμβο αυτό και το αποτέλεσμα του αλγορίθμου είναι να τοποθετηθεί ένας κινητός κόμβος στο σημείο p[3.5, 2] για να εξυπηρετεί τον κόμβο 1 που είναι γείτονας του κόμβου 7 και δεν μπορεί να βρει εναλλακτική διαδρομή.

Όσο αφορά τον κόμβο 8 εφόσον υπάρχει τουλάχιστον ένας από τους γείτονες του που του στέλνει δεδομένα (ο κόμβος 7) που είναι σε απόσταση μικρότερη από το εύρος αίσθησης από το σημείο που θα τοποθετηθεί ο κινητός κόμβος που θα εξυπηρετεί τον κόμβο 7, δεν χρειάζεται να γίνει κάποια αλλαγή του σημείου ή να τοποθετηθεί άλλος επιπλέον κινητός κόμβος.

Φάση Β:

Δεν υπάρχει κανένας κόμβος που να μην είναι σε συμφόρηση (congested) για να στέλνει τα δεδομένα του ο κινητός κόμβος που θα τοποθετηθεί για αυτό για να δημιουργηθεί μονοπάτι από τον mobile node στον κόμβο προορισμού τοποθετείται ένας άλλος κινητός κόμβος στο σημείο q[6, 2] (είναι το σημείο που είναι όσο πιο κοντά στον κόμβο προορισμού αλλά ταυτόχρονα είναι στο εύρος αίσθησης του κινητού κόμβου που θα τοποθετηθεί στο σημείο p). Ο κινητός κόμβος που θα τοποθετηθεί στο σημείο q μπορεί να επικοινωνήσει απευθείας με τον κόμβο προορισμού, έτσι δε χρειάζεται να δημιουργηθεί μονοπάτι από τον κινητό κόμβο στον κόμβο προορισμού.

Το μονοπάτι αυτό είναι βέλτιστο εφόσον δεν υπάρχει άλλο μονοπάτι, έτσι τοποθετούνται οι κινητοί κόμβοι 10 και 11 στα σημεία p και q αντίστοιχα.

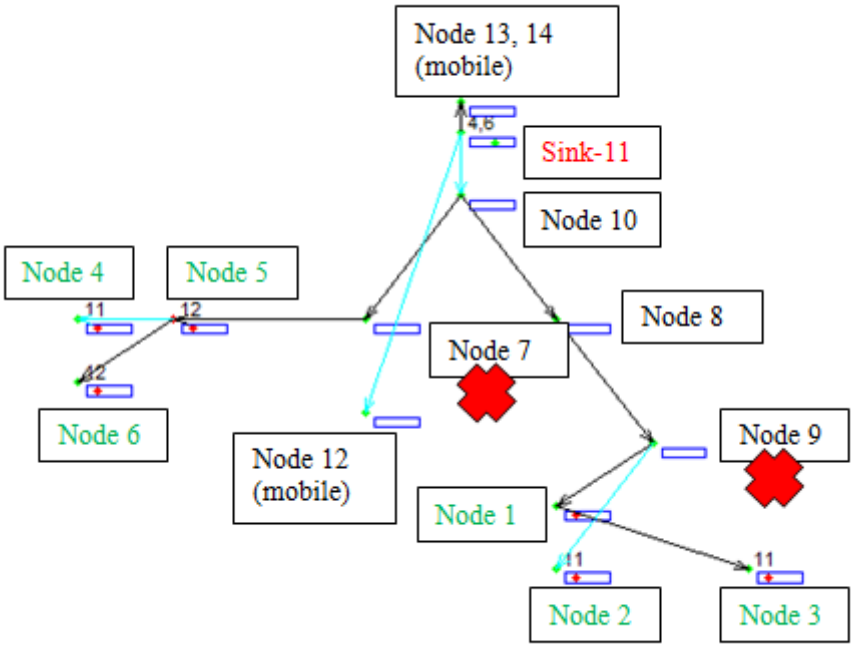
Παράδειγμα 4:

Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[5 1; 5 0; 7 0;0 4; 1 4; 0 3; 3 4; 5 4;6 2; 4 6; 4 7; 4 7.5; 4 7.5; 4 7.5];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 11, ο οποίος βρίσκεται στη θέση (4, 7).

Ο κινητός κόμβος 12, ο οποίος ήταν αρχικά τοποθετημένος στη θέση (4, 7.5), μετά την εκτέλεση του αλγορίθμου Global MobileCC τοποθετείται στη θέση (3.5, 2).



Σχήμα 5.10 Αναπαράσταση του δικτύου μετά την εκτέλεση του αλγορίθμου Global MobileCC.

Οι κόμβοι 12, 13, 14 είναι mobile nodes και είναι τοποθετημένοι αρχικά κοντά στον κόμβο προορισμού.

Φάση Α:

Οι κόμβοι 9, 7 είναι οι σε συμφόρηση (congested nodes) ταξινομημένοι σε φθίνουσα σειρά με βάση την απόσταση τους από τον κόμβο προορισμού.

Εφόσον ο κόμβος 9 είναι ο κόμβος που είναι πιο μακριά από τον κόμβο προορισμού καλείται ο Dynamic MobileCC για τον κόμβο αυτό και το αποτέλεσμα του αλγορίθμου είναι να τοποθετηθεί ένας κινητός κόμβος στο σημείο p[4.589, 3.466] για να εξυπηρετεί τον κόμβο 1 που είναι γείτονας του κόμβου 9 και δεν μπορεί να βρει εναλλακτική διαδρομή.

Όσο αφορά τον κόμβο 7 εφόσον κανένας από τους γείτονες του που του στέλνει δεδομένα (ο κόμβος 5 είναι ο μόνος κόμβος που στέλνει τα δεδομένα στον κόμβο 7) δεν βρίσκεται στο εύρος αίσθησης του σημείου που θα τοποθετηθεί ο κινητός κόμβος που θα εξυπηρετεί τον κόμβο 9, πρέπει να

εξεταστεί αν υπάρχει κάποιο άλλο σημείο που μπορεί να τοποθετηθεί ο κινητός κόμβος ώστε να συνεχίζει να εξυπηρετεί κάποιο γείτονα του κόμβου 9, και να εξυπηρετεί επίσης κάποιο γείτονα του κόμβου 7.

Τα πιθανά σύνολα για τα οποία πρέπει να βρεθεί αν υπάρχει σημείο τομής είναι τα εξής: [1, 5] [2,5] γιατί οι κόμβοι 1, 2 είναι οι κόμβοι που στέλνουν τα δεδομένα τους στον κόμβο 9 (με rate μεγαλύτερο από το additional_resources που χρειάζεται ο κόμβος 9) και ο κόμβος 5 είναι ο μόνος κόμβος που στέλνει τα δεδομένα του στον κόμβο 7 (με rate μεγαλύτερο από το additional_resources που χρειάζεται ο κόμβος 7).
Έτσι γίνεται αλλαγή του σημείου που θα τοποθετηθεί ο κινητός κόμβος και το σημείο γίνεται το [3.5, 2] που είναι το σημείο τομής των κόμβων του συνόλου [1, 5]. Σε αυτή τη θέση ο κινητός κόμβος θα επιλύει και τα δυο θέματα συμφόρησης.

Φάση Β:
Εφόσον στη θέση που θα τοποθετηθεί ο κινητός κόμβος δεν μπορεί να επικοινωνήσει με τον κόμβο προορισμού θα πρέπει να δημιουργηθεί το βέλτιστο μονοπάτι από αυτόν στον κόμβο προορισμού (με το μικρότερο κόστος δηλαδή να τοποθετηθούν όσο λιγότεροι mobile nodes γίνεται και να είναι έχει τα πιο λίγα βήματα-hops επίσης).
Οι κόμβοι που δεν είναι σε συμφόρηση και είναι στο εύρος αίσθησης του κινητού κόμβου που θα τοποθετηθεί είναι οι [10, 8, 5, 4, 1, 2]. Οι κόμβοι παρουσιάζονται ταξινομημένοι σε αύξουσα σειρά με βάση την απόσταση τους από τον κόμβο προορισμού.
Ακολουθεί η εκτέλεση του αλγορίθμου δυναμικού προγραμματισμού. Εφόσον ο κόμβος 11 είναι ο κόμβος προορισμού, το κόστος να επικοινωνήσει με τον κόμβο προορισμού είναι C(11)=0 και δεν χρειάζεται να τοποθετηθεί κάποιος κινητός κόμβος, άρα M(11)=0.
Όσο αφορά τον κόμβο 10 εφόσον ο κόμβος 11 είναι γείτονας του κόμβου 10, το κόστος για να επικοινωνήσει με τον κόμβο προορισμού είναι C(10)=1 και δεν χρειάζεται να τοποθετηθεί κάποιος κινητός κόμβος, άρα M(10)=0.

Όσο αφορά τον κόμβο 5 εφόσον ο κόμβος 10 είναι γείτονας του κόμβου 5 το κόστος για να επικοινωνήσει με τον κόμβο προορισμού είναι C(5)=2 μέσω του κόμβου 10 και δεν χρειάζεται να τοποθετηθεί κάποιος κινητός κόμβος, άρα M(5)=0.
Όσο αφορά τον κόμβο 4 εφόσον ο κόμβος 5 είναι γείτονας του κόμβου 4 το κόστος για να επικοινωνήσει με τον κόμβο προορισμού είναι C(4)=3 μέσω του κόμβου 5, και δεν χρειάζεται να τοποθετηθεί κάποιος κινητός κόμβος, άρα M(4)=0.
Όσο αφορά τον κόμβο 6 εφόσον ο κόμβος 5 είναι γείτονας του κόμβου 6 το κόστος για να επικοινωνήσει με τον κόμβο προορισμού είναι C(6)=3 μέσω του κόμβου 5 και δεν χρειάζεται να τοποθετηθεί κάποιος κινητός κόμβος, άρα M(6)=0.

Όσο αφορά τον κόμβο 1 εφόσον ο κόμβος 8 είναι γείτονας του κόμβου 1 το κόστος για να επικοινωνήσει με τον κόμβο προορισμού είναι C(1)=3 μέσω του κόμβου 8, και δεν χρειάζεται να τοποθετηθεί κάποιος κινητός κόμβος, άρα M(1)=0.
Όσο αφορά τον κόμβο 2 εφόσον ο κόμβος 1 είναι γείτονας του κόμβου 2 το κόστος για να επικοινωνήσει με τον κόμβο προορισμού είναι C(2)=4 μέσω του κόμβου 1, και δεν χρειάζεται να τοποθετηθεί κάποιος κινητός κόμβος, άρα M(2)=0.

Για την εύρεση του μονοπατιού με το μικρότερο κόστος παρατηρούμε ότι θα ελεγχτεί μόνο το ο αριθμός των βημάτων (hops) γιατί κανένα μονοπάτι δεν χρησιμοποιεί επιπλέον κινητούς κόμβους.
Για να μεταδίδει τα δεδομένα του στον κόμβο προορισμού ο κινητός κόμβος που θα τοποθετηθεί μπορεί να χρησιμοποιήσει το βέλτιστο μονοπάτι μέσω του κόμβου 5 ή μέσω του κόμβου 8 επειδή έχουν το μικρότερο κόστος (C(5)=2 και C(8)=2). Επιλέγεται το μονοπάτι μέσω του κόμβου 8 γιατί έχει περισσότερο διαθέσιμο rate.
Έτσι τοποθετείται ο κινητός κόμβος 12 στο σημείοο p και έχει πατέρα τον κόμβο 8.

Κεφάλαιο 6

Υλοποίηση και Πειραματική Αξιολόγηση

Σε αυτό το κεφάλαιο αναλύεται η υλοποίηση των τριών αλγορίθμων που ανατηχθήκαν στην παρούσα διπλωματική εργασία και περιγράφεται η πειραματική αξιολόγηση που έγινε. Στο πρώτο υποκεφάλαιο περιγράφεται ο προσομοιωτής που χρησιμοποιήθηκε. Στο δεύτερο υποκεφάλαιο περιγράφονται τα κριτήρια αξιολόγησης που χρησιμοποιηθήκαν. Στο τρίτο υποκεφάλαιο

περιγράφονται τα σενάρια που χρησιμοποιηθήκαν για την πειραματική αξιολόγηση και στο τελευταίο υποκεφάλαιο περιγράφονται τα αποτελέσματα της πειραματικής αξιολόγησης.

6.1 Περιγραφή του Προσομοιωτή

Για την αξιολόγηση της απόδοσης και για να γίνουν συγκρίσεις μεταξύ των τριών αλγόριθμων που αναπτύχθηκαν χρησιμοποιήθηκε ο προσομοιωτής Prowler και η εφαρμογή RMASE. Σε αυτή την ενότητα περιγράφονται αυτά τα εργαλεία και το πώς χρησιμοποιήθηκαν για την κατασκευή των αλγορίθμων υπό μελέτη.

6.1.1 Prowler

Ο προσομοιωτής Prowler είναι ένας πιθανολογικός προσομοιωτής ασύρματων δικτύων, που αναπτύχθηκε σε MATLAB [4]. Αν και ο Prowler παρέχει ένα γενικό περιβάλλον προσομοίωσης, η σημερινή πλατφόρμα του είναι η Berkeley MICA Mote που τρέχει TinyOS, το οποίο χρησιμοποιείται ευρέως στον τομέα της έρευνας για την ανάπτυξη πρωτόκολλων δρομολόγησης για ασύρματα δίκτυα αισθητήρων. Υποστηρίζει μια δομή βασισμένη σε γεγονότα (event-based), παρόμοια με το TinyOS/NesC η οποία καθιστά εύκολο να δημιουργηθεί ένας αλγόριθμος στην πλατφόρμα υλικού. Ο προσομοιωτής Prowler είναι ένας event-driven προσομοιωτής που μπορεί να ρυθμιστεί να λειτουργεί είτε ντετερμινιστικά, για την παραγωγή του αποτελέσματος κατά τον έλεγχο της εφαρμογής, ή πιθανολογικά, για την προσομοίωση της μη ντετερμινιστικής φύσης του διαύλου επικοινωνίας (communication channel) και του χαμηλού επιπέδου πρωτοκόλλου επικοινωνίας των κόμβων). Μπορεί να ενσωματώσει ένα αυθαίρετο αριθμό κόμβων, σε αυθαίρετες (πιθανώς δυναμικές) τοπολογίες και έχει σχεδιαστεί ώστε να μπορεί εύκολα να ενσωματωθεί σε αλγόριθμους βελτιστοποίησης.

Ο προσομοιωτής τρέχει κάτω από MATLAB, και έτσι προσφέρει ένα γρήγορο και εύκολο τρόπο για τη δημιουργία εφαρμογών, και έχει ωραίες δυνατότητες απεικόνισης. Ο Prowler μοντελοποιεί τις σημαντικές πτυχές όλων των

επιπέδων του διαύλου επικοινωνίας (communication channel) και την εφαρμογή. Η μη ντετερμινιστική φύση της ασύρματης μετάδοσης καθορίζεται από ένα πιθανολογικό μοντέλο. Ένα απλοποιημένο, αλλά ακριβές μοντέλο χρησιμοποιείται για να περιγράψει τη λειτουργία του Medium Access Control (MAC) layer. Οι εφαρμογές αλληλεπιδρούν με το MAC layer μέσω ενός συνόλου events και actions [30].

Το σύστημα του Prowler

Ο προσομοιωτής Prowler σχεδιάστηκε για να προσομοιώσει ένα πολύ επιτυχής, με χαμηλό κόστος πρωτότυπο κόμβου (Mote) που αναπτύχθηκε στο Μπέρκλεϊ. Η παραλλαγή που χρησιμοποιεί περιλαμβάνει ένα 8-bit, 4 MHz Atmel AT-MEGA 103 μικροελεγκτή, 128k μνήμη προγράμματος, 4kB RAM και ένα RFM TR1000 radio chip. Οι κόμβοι μπορούν επίσης να συμπεριλάβουν ένα σύνολο των εναλλάξιμων αισθητήρων (για μέτρηση της θερμοκρασίας, του φωτός, του ήχου, κλπ).

Μοντέλα ασύρματης μετάδοσης (Radio Propagation Models)

Το μοντέλο ασύρματης μετάδοσης καθορίζει την ισχύ του εκπεμπόμενου σήματος σε ένα συγκεκριμένο σημείο του χώρου για όλους τους αποστολείς του συστήματος. Βάσει αυτών των πληροφοριών οι συνθήκες υποδοχής σήματος για τους δέκτες μπορούν να αξιολογηθούν και οι συγκρούσεις μπορούν να ανιχνευθούν. Η ισχύς του σήματος από ένα αποστολέα σε ένα δέκτη καθορίζεται από ένα ντετερμινιστικό λειτουργία διάδοσης (μοντελοποίηση της φθοράς της ισχύος του σήματος με βάση την απόσταση), καθώς και από τυχαίες διαταραχές (μοντελοποίηση του φαινομένου της εξασθένησης - fading effect, της χρονικά μεταβαλλόμενης φύσης της ισχύος του σήματος, και άλλα σφάλματα μετάδοσης).

$$P_{rec.ideal}(d) = P_{transmit} \frac{1}{1 + d^{\gamma}} \qquad (1)$$

όπου Prec.ideal είναι η ιδανική ισχύς του σήματος που λαμβάνει κάποιος κόμβος, Ptransmit είναι η ισχύς του σήματος μετάδοσης, d είναι η απόσταση μεταξύ του αποστολέα και του δέκτη και γ είναι μια παράμετρος με τυπικές τιμές

2 ≤ γ ≤ 4. Τα πραγματικά σήματα, ωστόσο, συμπεριφέρονται με έναν πολύ διαφορετικό τρόπο. Η ισχύς του σήματος μπορεί να ποικίλλει σε πολύ μεγάλο βαθμό, καθώς αλλάζει η απόσταση. Επίσης, με το πέρας του χρόνου η ισχύς του σήματος μπορεί να αλλάξει
ακόμα και αν η απόσταση μεταξύ του αποστολέα και του δέκτη είναι σταθερή. Αυτό το φαινόμενο της εξασθένησης - fading effect μοντελοποιείται από τυχαίες διαταραχές στον προσομοιωτή. Η ισχύς του λαμβανόμενου σήματος από τον κόμβο j στον κόμβο i υπολογίζεται από τη συνάρτηση μετάδοσης (1), διαμορφώνοντας με τυχαία λειτουργίες:

$$P_{rec}(i,j) = \text{Prec.ideal}(di,j).[\,(1 + a(di,j)).(1 + \beta(t))\,] \qquad (2)$$

Οι παράμετροι α και είναι β είναι τυχαίες μεταβλητές με κανονική κατανομή N(0, σ_α) και N(0, σ_β) και ρυθμιζόμενες παραμέτρους σα και σβ. Μια επιπλέον παράμετρος είναι η perror η οποία μοντελοποιεί την πιθανότητα εμφάνισης σφαλμάτων μετάδοσης που προκαλούνται από οποιεσδήποτε επιδράσεις που δεν έχουν μοντελοποιηθεί (π.χ. εξωτερικές διαταραχές, αναξιόπιστο υλικό, κλπ).

Για τη λήψη του σήματος και για τη μοντελοποίηση των συγκρούσεων (collisions) υπάρχουν τρία μοντέλα. Στο πρώτο μοντέλο το σήμα λαμβάνεται εάν η ισχύς του σήματος είναι μεγαλύτερη από μια παράμετρο που δείχνει τη μικρότερη ισχύς που πρέπει να έχει το σήμα για να το λάβει σωστά ο δέκτης. Το κανάλι εντοπίζεται να είναι αδρανές, αν δεν υπάρχει σήμα που θα μπορούσε να λάβει. Υπάρχει μια σύγκρουση (collision) εάν δύο μεταδόσεις επικαλύπτονται χρονικά και μπορούν και οι δυο να ληφθούν από το κανάλι. Στο δεύτερο μοντέλο κάθε δέκτης έχει μια παράμετρο διακύμανσης του θορύβου. Σε αυτό το μοντέλο μετριέται η ισχύς του σήματος σε σχέση με τις παρεμβολές (Signal to Interference Ratio - SINR) και το σήμα λαμβάνεται εάν το SINR στο δέκτη είναι μεγαλύτερο από το κάποιο όριο κατά όλη τη διάρκεια της μετάδοσης του σήματος. Το κανάλι εντοπίζεται να είναι αδρανές, αν η συνολική ισχύς του σήματος είναι μικρότερη από ένα όριο, το οποίο εξαρτάται από την διακύμανση του θορύβου του δέκτη. Υπάρχει μια σύγκρουση (collision) εάν το SINR στο δέκτη γίνει μικρότερο από το όριο λήψης του σήματος ανά πάσα στιγμή κατά τη διάρκεια της λήψης. Το τρίτο μοντέλο είναι παρόμοιο με το δεύτερο με τη

διαφορά ότι χρησιμοποιεί τη Rayleigh λειτουργία για το fading effect. Το πρώτο μοντέλο είναι απλό και γρήγορο στους υπολογισμούς, ενώ τα δυο άλλα είναι πιο ακριβές. Στις προσομοιώσεις που έγιναν στα πλαίσια της παρούσας διπλωματικής εργασίας χρησιμοποιήθηκε το πρώτο μοντέλο κυρίως για σκοπούς υπολογισμού.

MAC-layer μοντέλο

Η επικοινωνία στο στρώμα MAC μοντελοποιείται από ένα απλοποιημένο event channel. Είναι ένα απλό CSMA MAC στρώμα. Όταν η εφαρμογή εκπέμπει το event SendPacket, μετά από ένα τυχαίο χρονικό διάστημα (WaitingTime) το στρώμα MAC ελέγχει αν το κανάλι είναι αδρανές. Αν όχι, συνεχίζει τον έλεγχο αδράνειας έως ότου το κανάλι να βρίσκεται σε αδράνεια, πριν από κάθε έλεγχο αδράνειας περιμένει τυχαία χρονικά διαστήματα (BackoffTime). Όταν το κανάλι είναι αδρανές η μετάδοση αρχίζει και μετά από το χρονικό διάστημα μετάδοσης του πακέτου (TransmissionTime) η εφαρμογή λαμβάνει το event PacketSent. Μετά τη λήψη ενός πακέτου από την πλευρά του δέκτη, η εφαρμογή λαμβάνει ένα PacketReceived ή CollidedPacketReceived συμβάν, ανάλογα με την επιτυχία της μετάδοσης. Οι παράμετροι WaitingTime και BackoffTime είναι τυχαίες ομοιόμορφα κατανεμημένες μεταβλητές σε προκαθορισμένα χρονικά διαστήματα, ενώ το TransmissionTime είναι σταθερό, δηλαδή όλα τα μηνύματα έχουν το ίδιο μήκος.

Το επίπεδο εφαρμογής

Οι εφαρμογές είναι βασισμένες σε γεγονότα (event-based), παρόμοια με το TinyOS. Στον προσομοιωτή τα ακόλουθα γεγονότα (events) μπορούν να εμφανιστούν: InitApplication, PacketSent, PacketReceived, CollidedPacketReceived, ClockTick. Η εφαρμογή μπορεί να ενεργοποιήσει τις δράσεις SetClock και SendPacket οι οποίες προκαλούν περαιτέρω γεγονότα.

6.1.2 Rmase

Το Rmase έχει αναπτυχθεί για την αντιμετώπιση της πρόκλησης της σύγκρισης διαφόρων αλγορίθμων δρομολόγησης στα ασύρματα δίκτυα αισθητήρων [31]. Το Rmase αποτελείται από ένα μοντέλο τοπολογίας δικτύου, ένα μοντέλο

εφαρμογής και ένα μοντέλο επιδόσεων. Το Rmase υποστηρίζει επίσης μια αρχιτεκτονική χωρισμένη σε επίπεδα δρομολόγησης για plug-and-play κοινά στοιχεία δρομολόγησης. Πολλές εφαρμογές έχουν μοντελοποιηθεί σε Rmase, και έχουν αναπτυχθεί πολλοί αλγόριθμοι δρομολόγησης και συστατικά. Το Rmase έχει χρησιμοποιηθεί για την ανάπτυξη νέων αλγορίθμων δρομολόγησης, για την ανάλυση των trade-offs απόδοσης, και για την επιλογή του καλύτερου αλγορίθμου δρομολόγησης για μια δεδομένη εφαρμογή και τις απαιτήσεις απόδοσης της εφαρμογής. Το Rmase υλοποιείται ως μια εφαρμογή στον προσομοιωτή Prowler, ο οποίος περιγράφεται στο προηγούμενο υποκεφάλαιο.

Ένα συστατικό χειρίζεται γεγονότα (π.χ., PacketReceived, PacketSent, ClockTick) και εκτελεί τις εντολές (π.χ., SendPacket). Από προεπιλογή τα γεγονότα περνούν πάνω και οι εντολές περνούν κάτω στα στρώματα. Ωστόσο, μπορεί να σταματήσει τη ροή θέτοντας τη μεταβλητή pass ίση με 0 (false) για μια εντολή ή ένα event.

Η ελάχιστη διαμόρφωση των στρωμάτων του Rmase περιλαμβάνει το στρώμα MAC (MAC layer), ένα στρώμα δρομολόγησης (routing layer), και ένα στρώμα εφαρμογής (application layer), με το στρώμα MAC στο κάτω μέρος και το στρώμα εφαρμογής στην κορυφή. Το στρώμα εφαρμογής είναι για την παραγωγή των σεναρίων εφαρμογής και τις εντολές SendPacket. Το στρώμα δρομολόγησης προωθεί τα πακέτα που λαμβάνονται για άλλους κόμβους ή προωθεί το πακέτο που λαμβάνει στο πάνω στρώμα εάν ο κόμβος είναι δέκτης του εν λόγω πακέτου. Το στρώμα MAC θα στείλει πραγματικά το πακέτο. Ένα πακέτο πρέπει να φθάσει στο στρώμα εφαρμογής, αν και μόνο αν ο κόμβος είναι ο δέκτης του πακέτου. Το στρώμα MAC προσομοιώνει τη λειτουργικότητα MAC στρώματος του TinyOS. Ένα πακέτο μπορεί να σταλεί στο επόμενο hop (data.address = NodeId) ή να γίνει broadcast (data.address = 0). Επιπρόσθετα ο τρόπος επικοινωνίας μπορεί να ρυθμιστεί σε promiscuous, έτσι ώστε όλοι οι γείτονες να μπορούν να λάβουν ένα πακέτο, ακόμα και αν το πακέτο στέλνεται στον επόμενο κόμβο. Το στρώμα MAC προσομοιώνει επίσης την κατανάλωση ενέργειας με ένα παραμετροποιημένο μοντέλο κατανάλωσης ενέργειας για τη μετάδοση, τη λήψη και την αδρανή κατάσταση. Ένας κόμβος μπορεί επίσης να

αλλάξει κατάσταση σε κατάσταση ύπνου κατά την οποία δεν καταναλώνεται ενέργεια.

Το στρώμα στατιστικές (statistics layer) προστίθεται μετά το στρώμα εφαρμογής για τη συλλογή στατιστικών στοιχείων δρομολόγησης (π.χ. το συνολικό αριθμό των απεσταλμένων πακέτων, το συνολικό αριθμό των ληφθέντων πακέτων, τις χρονικές καθυστερήσεις των πακέτων που ληφθήκαν, κλπ), για τον υπολογισμό των μετρικών απόδοσης.

Επιπλέον, μπορεί να υπάρξει ένα σύνολο από στρώματα ελέγχου μεταξύ του στρώματος δρομολόγησης και του στρώματος εφαρμογής και ένα σύνολο στρωμάτων στήριξης μεταξύ του στρώματος MAC και του στρώματος δρομολόγησης. Τα στρώματα ελέγχου χρησιμοποιούνται για την αρχικοποίηση ή τη συντήρηση του δικτύου, π.χ. την οικοδόμηση και τη διατήρηση ενός πίνακα δρομολόγησης ή ενός γεννητικού δένδρου (spanning tree). Τα στρώματα υποστήριξης είναι για την προσθήκη λειτουργιών, όπως τη μετάδοση και τη λήψη ουρών, τη συγκέντρωση/ κατακερματισμό δεδομένων, τον έλεγχο duplications, κλπ. Είναι επιλογή του σχεδιαστή του αλγορίθμου να θέσει επιμέρους λειτουργίες σε διαφορετικά στρώματα, έτσι ώστε κοινές λειτουργίες να μπορούν να χρησιμοποιηθούν από διαφορετικούς αλγορίθμους.

Συστατικά κοινής δρομολόγησης

Αυτή η αρχιτεκτονική που χωρίζεται σε επίπεδα επιτρέπει τη χρήση κοινών στοιχείων από διαφορετικούς αλγορίθμους. Για παράδειγμα, πολλοί αλγόριθμοι χρειάζονται διαχείριση των γειτόνων ενός κόμβου και των ουρών μετάδοσης, ένα ενιαίο μονοπάτι δρομολόγησης μπορεί να χρειάζεται επιβεβαίωση για την αξιόπιστη μετάδοση, και μια flooding-based δρομολόγηση μπορεί να προσθέσει καθυστερήσεις μετάδοσης για τη μείωση των συγκρούσεων (collisions). Έτσι, για ένα δεδομένο σενάριο εφαρμογής, πρέπει να επιλεχτεί εκτός από τον αλγόριθμο δρομολόγησης, και τα συστατικά υποστήριξης και έλεγχου. Ο Πίνακας 6.1 δείχνει μια λίστα με τα συστατικά υποστήριξης και έλεγχου που είναι υλοποιημένα στο Rmase.

Name	Type	Definition	Parameters	Affected data fields
Probabilistic Faulty Model	support	model the probabilities from alive to dead and from dead to alive	FailProb, WakeupProb	
Transmission Queue	support	maintain a transmission queue		
Aggregation Queue	support	pack a maximum of M packets from the queue into one packet to transmit. On the receiving side, a packed packet will be unpacked into the original packets	AggregatePackets, AggregateTimeout	
Maximum Hops	support	bound the maximum number of hops in transmission	maxhops	
Neighborhood Management	support	maintain a neighbor list	from	
Transmission Confirmation	support	trigger a timeout event when no confirmation is heard within a certain time	TransTimeout	
Duplication Check	support	mark duplicated packets		SeqID, source, duplicated
Transmission Delay	support	delay a packet transmission according to the timeDelay field of data, set by upper layers, and transmit a forwarding packet probabilistically according to the number of times the packet has been heard	randOff, minDelay	SeqID, source, duplicated
Hello Initialization	control	broadcast a couple of "hello" packets at a certain interval from each node		msgID<0
Backward Initialization	control	flood from the destination to set up hop counts or to build an initial spanning tree, which are used by many routing algorithms		msgID<0

Πίνακας 6.1 Κοινά συστατικά δρομολόγησης στην εφαρμογή Rmase [2]

6.1.3 Αιτιολόγηση σχετικά με την επιλογή του προσομοιωτή

Το Prowler / Rmase είναι ένας event-driven προσομοιωτής που χτίστηκε σε MATLAB και είναι απλός στη χρήση. Ικανοποιεί όλες τις μη-λειτουργικές και λειτουργικές απαιτήσεις για προσομοιωτές, όπως περιγράφονται στο [32]. Αν και ο προσομοιωτής NS-2 χρησιμοποιείται πιο συχνά για ερευνά, επιλέξαμε να μην το χρησιμοποιήσουμε για μια διάφορους λόγους.

Ενώ ο NS-2 είναι ένας ισχυρός προσομοιωτής δικτύων, μερικοί ερευνητές ισχυρίστηκαν ότι έχει κάποια μειονεκτήματα για την προσομοίωση ασύρματων δικτύων αισθητήρων [33, 34]. Ένας αντικειμενοστραφής σχεδιασμός σε ns-2 μπορεί να εισάγει κάποια περιττή αλληλεξάρτηση μεταξύ των μονάδων που μπορεί να οδηγήσει σε δυσκολίες στη νέα προσθήκη πρωτοκόλλων [34]. Επιπλέον, ο προσομοιωτής ns-2 έχει αναπτυχθεί πριν περίπου 10 χρόνια, έχει αρκετά μεγάλη και αναπτυσσόμενη υποδομή που οδηγεί σε μια απότομη καμπύλη εκμάθησης για έναν αρχάριο χρήστη.

Επιπλέον, ο προσομοιωτής Prowler χρησιμοποίει την πλατφόρμα MICA mote, η οποία χρησιμοποιείται ευρέως σε πραγματικές εφαρμογές στον τομέα της μελέτης των ασύρματων δικτύων αισθητήρων. Είναι μια event-based αρχιτεκτονική που καθιστά εύκολο τη μεταφορά ενός συστήματος άμεσα σε πραγματικό υλικό.

Τέλος, αν και υπάρχουν και άλλοι προσομοιωτές, όπως ο GloMoSim [34] ή ο TOSSIM [33], αποφάσισα να χρησιμοποιήσω το Prowler / Rmase λόγω της ήδη υπάρχουσας υλοποίησης του αλγορίθμου DalPaS στον εν λόγω προσομοιωτή. Στο [3] περιγράφεται η εμφανής αποτελεσματικότητα ενός προσομοιωτή υψηλού-επίπεδου όπως το Prowler / Rmase.

6.1.4 Περιγραφή της Αξιολόγησης του Αλγορίθμου DAlPaS

Ο αλγόριθμος DalPAs, ο οποίος έχει χρησιμοποιηθεί ως ο αλγόριθμος ελέγχου συμφόρησης για την επίτευξη της πειραματικής αξιολόγησης των αλγορίθμων MobileCC, έχει αξιολογηθεί στον προσομοιωτή Prowler με πολύ καλά αποτελέσματα. Για την τυχαία κατανεμημένη τοπολογία έχει γίνουν οι παρατηρήσεις που ακολουθούν. Όσο αφορά τον αριθμό των πακέτων που λαμβάνει ο κόμβος προορισμού, έναντι τον αριθμό των πακέτων που δημιουργούνται από τους πηγαίους κόμβους, η πειραματική αξιολόγηση έχει δείξει ότι ο DAlPaS είναι αποτελεσματικός και μπορεί να ισορροπήσει το φόρτου μεταξύ των κόμβων αποτελεσματικά, προκειμένου να αποφευχθούν οι απώλειες πακέτων. Παρατηρείται ότι ο αλγόριθμος DAlPaS παρουσιάζει το υψηλότερο ποσοστό των επιτυχώς ληφθέντων πακέτων (άρα τις λιγότερες απώλειες πακέτων) ενώ αυξάνεται το ποσό των πακέτων που δημιουργούν ανά δευτερόλεπτο οι πηγαίοι κόμβοι. Επιπλέον η απόδοση του δικτύου είναι επίσης υψηλότερη σε σύγκριση με τους υπόλοιπους αλγορίθμους που έχει συγκριθεί.

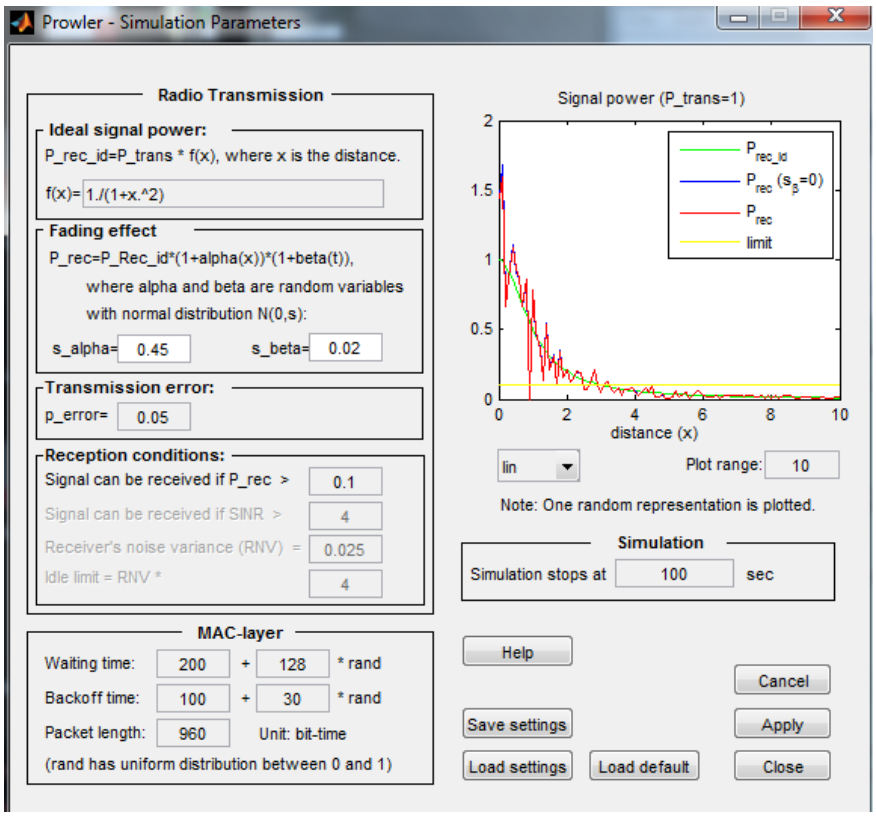
6.1.5 Περιβάλλον Προσομοίωσης

Οι παράμετροι προσομοίωσης που περιγράφονται πιο κάτω είναι αυτές που χρησιμοποιηθήκαν για την πειραματική αξιολόγηση του αλγορίθμου DalPaS στο [2].

Τα παρακάτω στοιχεία ισχύουν στο περιβάλλον προσομοίωσης.

Ασύρματη Διάδοση (Radio Propagation)

Η συνάρτηση της διάδοσης του σήματος είναι 1 / 1+x^2. Το όριο λήψης ενός πακέτου είναι 0,1. Η πιθανότητα σφάλματος κατά τη μετάδοσης είναι 0,05. Η διακύμανση τοπολογίας του σήματος ασύρματης μετάδοσης είναι 0,45 και η τυχαία διακύμανση του σήματος μετάδοσης είναι 0,02. Με τον τρόπο αυτό το φαινομένου της εξασθένησης - fading effect καταργείται, γεγονός που μειώνει την τυχαιότητα της ασύρματης μετάδοσης. Αυτό ελαχιστοποιεί τις απαιτούμενες επαναλήψεις αποστολής πακέτων μιας εκτέλεσης της προσομοίωσης και να παρέχει σαφέστερα αποτελέσματα για λόγους σύγκρισης. Στο MAC Layer ο ελάχιστος χρόνος αναμονής είναι 200, ενώ ο επιπρόσθετος μέγιστος τυχαίος χρόνος αναμονής είναι 128. Ο ελάχιστος BackoffTime χρόνος είναι 100 ενώ ο επιπρόσθετος μέγιστος τυχαίος BackoffTime χρόνος είναι 30. Όλες οι τιμές που αναφέρονται είναι σε χρόνο προσομοίωσης. Το μήκος ενός πακέτου είναι 200 bits, το οποίο αντιστοιχεί σε 25 bytes και το bit-time σε δευτερόλεπτα είναι 1/40000.



Σχήμα 6.1 Πίνακας με παραμέτρους

Χαρακτηριστικά τοπολογίας

Η ισχύς του σήματος μετάδοσης ορίζεται ως 0,7 στις παραμέτρους του Prowler. Οι πηγαίοι κόμβοι είναι τοποθετημένοι στην κάτω αριστερή γωνία, ενώ ο κόμβος προορισμού τοποθετείται στην πάνω δεξιά γωνία, για να δημιουργούνται ροές που συγκλίνουν ή διασταυρώνουν. Ο αριθμός των πακέτων που δημιουργούν ανα δευτερόλεπτο οι πηγαίοι κόμβοι (source rate) είναι 4, που οδηγεί στην παραγωγή 20 πακέτων ανά δευτερόλεπτο. Κάθε κόμβος έχει ρυθμιστεί να είναι σε θέση να διατηρεί στο buffer του το πολύ 15 πακέτα. Σε όλες τις προσομοιώσεις ισχύει το ακόλουθο πρότυπο μετάδοσης πακέτων: Στην αρχή οι πηγαίοι κόμβοι δεν μεταδίδουν πακέτα, καθώς εκτελείται η φάση αρχικοποίησης του δικτύου. Μετά από περίπου 1 δευτερόλεπτο όλοι οι πηγαίοι κόμβοι αρχίζουν να μεταδίδουν τα δεδομένα τους με τον προκαθορισμένο ρυθμό μετάδοσης δεδομένων (data rate). Η προσομοίωση τερματίζει μετά από 10 δευτερόλεπτα. Κάθε εκτέλεση της προσομοίωσης έχει γίνει 20 φορές και έχει εξαχθεί ο μέσος όρος των αποτελεσμάτων.

Σημειώστε ότι οι παράμετροι στο Prowler (όπως η ισχύς του σήματος) και οι μετρικές του Rmase (όπως η κατανάλωση της ενέργειας) δίδονται σε απλούς αριθμούς και όχι στις πραγματικές μονάδες μέτρησης τους (δηλαδή dB ή Watt ή Joule). Αυτό οφείλεται στο γεγονός ότι αυτά τα εργαλεία κατασκευάστηκαν για τη βελτιστοποίηση των αλγόριθμων δρομολόγησης και για σκοπούς σύγκρισης. Οι πηγαίοι κόμβοι επιλέγονται ντετερμινιστικά ή χρησιμοποιώντας μια πιθανολογική λειτουργία σε μια προκαθορισμένο περιοχή του δικτύου.

6.1.6 Υλοποίηση των Αλγορίθμων στο Prowler-Rmase

Σε αυτό το υποκεφάλαιο περιγράφονται συνοπτικά τα βήματα που ακολουθήθηκαν για να υλοποιηθούν οι αλγόριθμοι στην εφαρμογή Rmase. Όπως έχει ήδη αναφερθεί στο Rmase υπήρχε ήδη η υλοποίηση του αλγορίθμου DalPaS. Η υλοποίηση χρησιμοποιήθηκε με κάποιες αλλαγές . Αρχικά έγινε η επικοινωνία μεταξύ του νέου επιπέδου που δημιουργήθηκε του MobileCC και του επιπέδου DalPaS. Στη συνέχεια έγινε η υλοποίηση των τριών αλγορίθμων έτσι ώστε να έχουν ως αποτέλεσμα τις θέσεις που θα

τοποθετηθούν οι κινητοί κόμβοι. Στη συνέχεια εισάχθηκε η κίνηση των κόμβων με στόχο να φθάσουν σε μια συγκεκριμένη θέση. Τέλος δημιουργηθήκαν οι τοπολογίες που περιγράφονται πιο κάτω και έγινε η αυτοματοποίηση των προσημειώσεων για να έχουμε ως αποτέλεσμα τη συλλογή ων μετρήσεων που περιγράφονται πιο κάτω.

Όλος ο πηγαίος κώδικας της υλοποίησης των αλγορίθμων Dynamic MobileCC, Direct Path MobileCC και Global MobileCC και τα σχόλια είναι διαθέσιμος στο Παράρτημα Α.

6.2 Κριτήρια Αξιολόγησης

Σε αυτό το κεφάλαιο περιγράφονται οι μετρικές σύγκρισης των αλγορίθμων Dynamic MobileCC, Direct Path MobileCC και Global MobileCC.

Χρησιμοποιούνται διάφορες μετρικές αξιολόγησης της απόδοσης για τη σύγκριση διαφορετικών στρατηγικών ελέγχου συμφόρησης σε δίκτυα αισθητήρων. Ο προσομοιωτής Prowler παρέχει τις μετρικές που περιγράφονται πιο κάτω και αυτές χρησιμοποιήθηκαν για σκοπούς σύγκρισης του αλγορίθμου Dynamic MobileCC και Direct Path MobileCC. Η τοπολογία που χρησιμοποιήθηκε για αυτές τις προσομοιώσεις είναι μια 10x10 ομοιόμορφα κατανεμημένη τοπολογία (uniform topology) με τυχαία απόκλιση τιμών ίση με 0,1 και στους δύο άξονες. Το περιβάλλον προσομοίωσης περιγράφεται στο υποκεφάλαιο 6.1.4 . Οι προσομοιώσεις έγιναν για 5 διαφορετικά πλήθη κόμβων: 50, 60, 70, 80, 90 και 100 κόμβους.

Σημειώνεται ότι δεν έχει γίνει η τυχαιοποιημένη αξιολόγηση για τον αλγόριθμο Global MobileCC γιατί για δίκτυα με μεγάλο αριθμό κόμβων εφόσον δημιουργείται τυχαία η τοπολογία του δικτύου δεν υπάρχει εκ των προτέρων η γνώση των κόμβων που θα σημειώσουν συμφόρηση.

Μετρικές για τυχαιοποιημένη αξιολόγηση:

- Ποσοστό επιτυχίας (Success Rate): Ο συνολικός αριθμός των πακέτων που λαμβάνει ο κόμβος προορισμού έναντι του συνολικού αριθμού των πακέτων που αποστέλλονται από όλους τους πηγαίους κόμβους. Μετρά τη συνολική επιτυχία του δικτύου.

- Καθυστερήσεις (Delays): Ο μέσος χρόνος που χρειάζεται για να σταλθεί ένα μήνυμα από τον πηγαίο κόμβο στον κόμβο προορισμού. Αν n πακέτα έχουν φθάσει στον κόμβο προορισμού, το delay θα είναι ίσο με Σdi / n , όπου το di είναι η καθυστέρηση του i-οστού πακέτου.

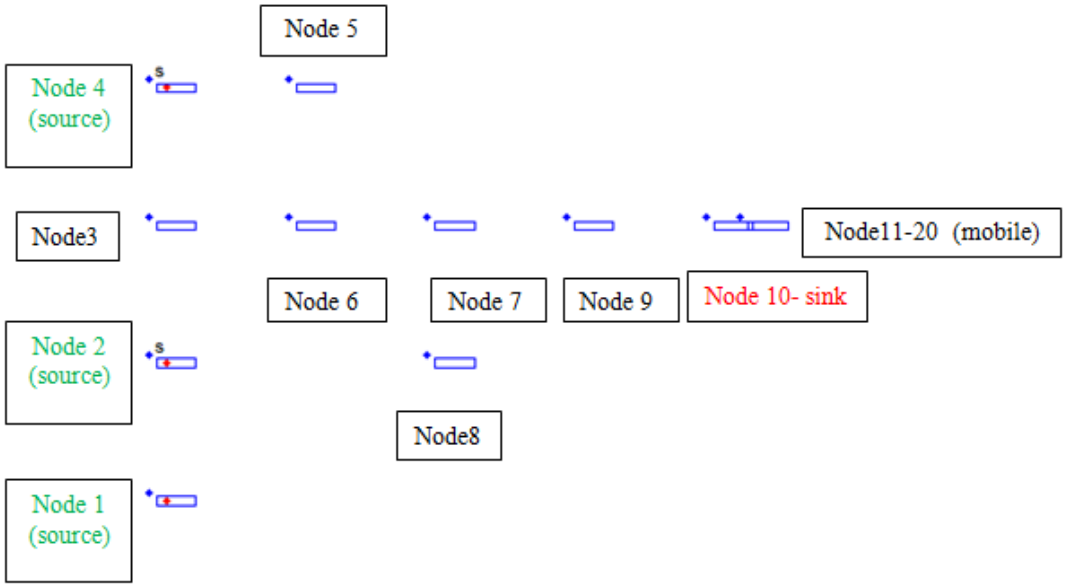
Μετρική για τυχαιοποιημένη και στοχευμένη αξιολόγηση:
Η πιο σημαντική μετρική η οποία χρησιμοποιήθηκε για την αξιολόγηση των αλγορίθμων είναι ο μέσος όρος του πλήθους των κινητών κόμβων που χρησιμοποιούνται κατά την εκτέλεση της προσομοίωσης.

6.3 Σενάρια

6.3.1 Στοχευμένη Αξιολόγηση

Για τη στοχευμένη αξιολόγηση χρησιμοποιήθηκε το περιβάλλον προσομοίωσης που περιγράφεται στο Κεφάλαιο 6.1.4 .Οι τοπολογίες που χρησιμοποιηθήκαν για κάθε σενάριο περιγράφονται αναλυτικά πιο κάτω.

ΣΕΝΑΡΙΑ ΜΕ 10 ΣΤΑΤΙΚΟΥΣ ΚΟΜΒΟΥΣ
Σενάριο 1 – 10 στατικοί κόμβοι:
Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):
topology=[0 1; 0 2; 0 3; 0 4; 2 4; 2 3; 4 3; 4 2; 6 3; 8 3; 8.5 3; 8.5 3; 8.5 3; 8.5 3; 8.5 3; 8.5 3; 8.5 3; 8.5 3; 8.5 3; 8.5 3;];
Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 10, ο οποίος βρίσκεται στη θέση (8, 3).



Σχήμα 6.2 Τοπολογία του Σεναρίου 1 με 10 κόμβους

Σε αυτή την τοπολογία παρατηρούμε ότι ο κόμβος 7 είναι bottleneck γιατί οι ροές των κόμβων 5 και 6 στέλνονται σε αυτόν τον κόμβο και έτσι σύντομα θα είναι σε συμφόρηση.

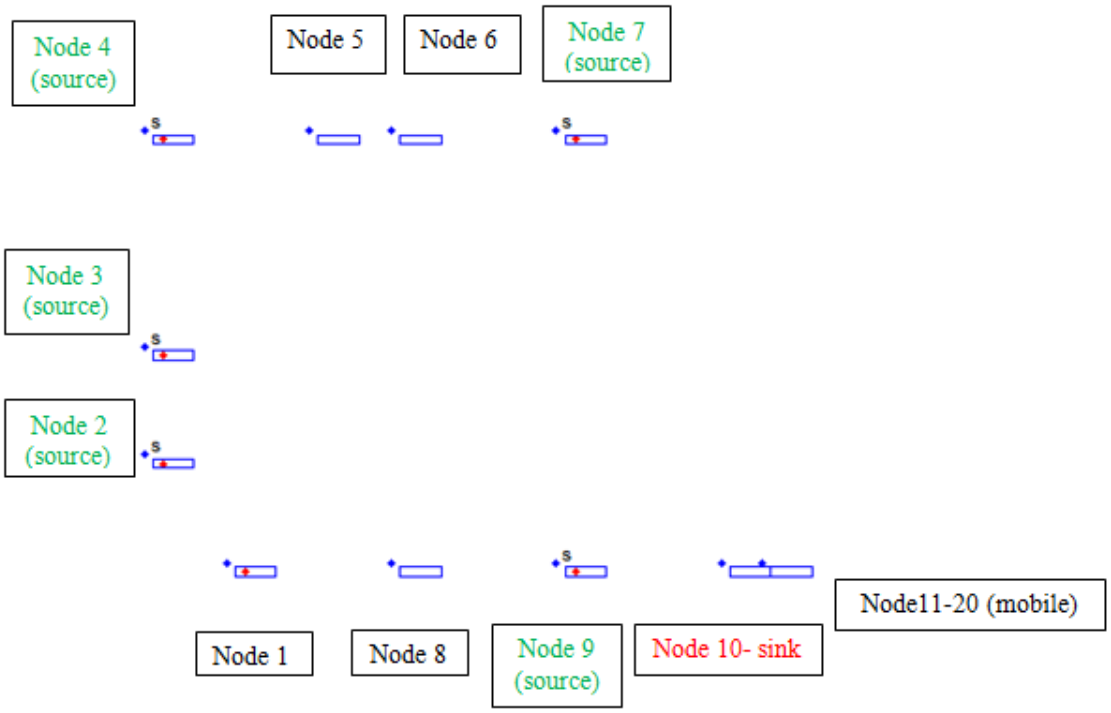
Επίσης μετά τον κόμβο bottleneck παρατηρούμε ότι δημιουργείται γραμμή μέχρι τον κόμβο προορισμού μέσω του κόμβου 9, και αναμένουμε ότι ο κόμβος 9 θα είναι επίσης σε συμφόρηση.

Σενάριο 2 - 10 κόμβοι:

Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[1 2; 0 3; 0 4; 0 6; 2 6; 3 6; 5 6; 3 2; 5 2; 7 2; 7.5 2; 7.5 2; 7.5 2; 7.5 2; 7.5 2; 7.5 2; 7.5 2; 7.5 2; 7.5 2; 7.5 2];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 10, ο οποίος βρίσκεται στη θέση (7, 2).



Σχήμα 6.3 Τοπολογία του Σεναρίου 2 με 10 κόμβους

Σε αυτή την τοπολογία παρατηρούμε ότι ο κόμβος 1 είναι bottleneck γιατί οι ροές των κόμβων 2 και 3 στέλνονται σε αυτόν τον κόμβο και έτσι σύντομα θα είναι σε συμφόρηση.

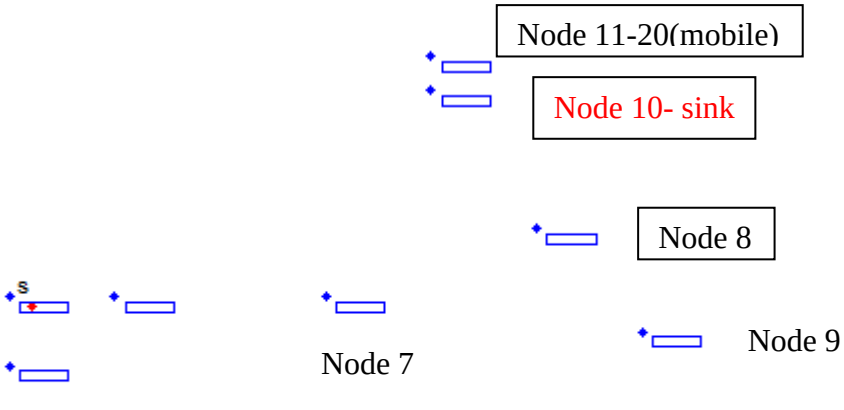
Επίσης μετά τον κόμβο bottleneck παρατηρούμε ότι δημιουργείται γραμμή μέχρι τον κόμβο προορισμού μέσω των κόμβων 8 και 9, και αναμένουμε ότι οι κόμβοι 8 και 9 θα είναι επίσης σε συμφόρηση.

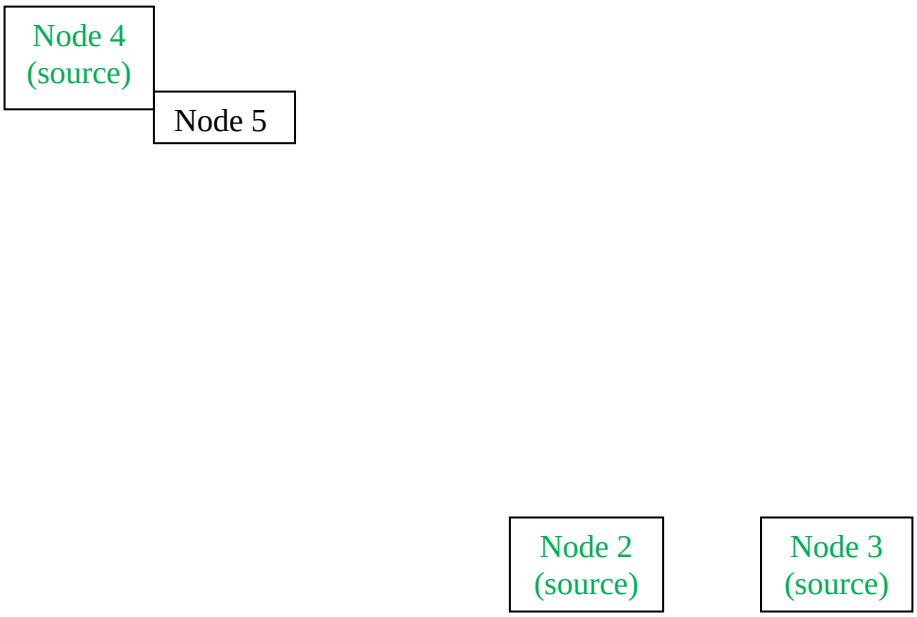
Σενάριο 3 - 10 κόμβοι:

Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[5 1.5; 5 0; 7 0; 0 4; 1 4; 0 3; 3 4; 5 5; 6 3.5; 4 7; 4 7.5; 4 7.5; 4 7.5; 4 7.5; 4 7.5; 4 7.5; 4 7.5; 4 7.5];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 10, ο οποίος βρίσκεται στη θέση (4, 7).

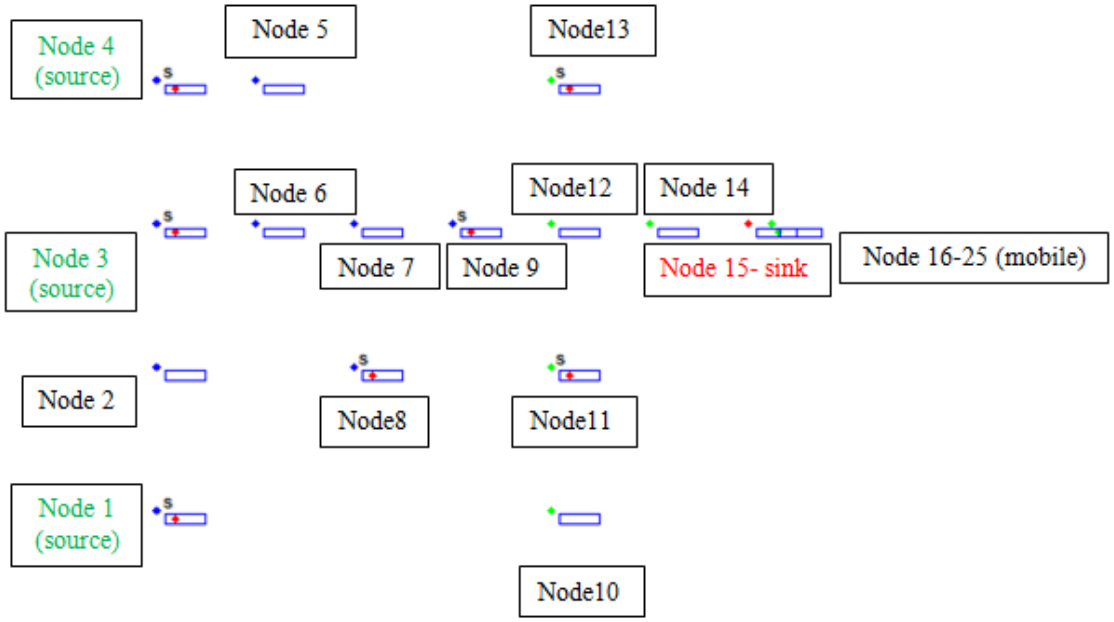




Σχήμα 6.4 Τοπολογία του Σεναρίου 3 με 10 κόμβους

Σε αυτή την τοπολογία παρατηρούμε ότι ο κόμβος 1 είναι bottleneck γιατί οι ροές των κόμβων 2 και 3 στέλνονται σε αυτόν τον κόμβο και έτσι σύντομα θα είναι σε συμφόρηση. Επίσης παρατηρούμε ότι ο κόμβος 8 είναι bottleneck γιατί οι ροές των κόμβων 7 και 9 στέλνονται σε αυτόν τον κόμβο και έτσι σύντομα θα είναι σε συμφόρηση.
Άρα εξετάζεται το σενάριο όπου δυο θέματα συμφόρησης εμφανίζονται σε δυο διαφορετικά μέρη του δικτύου.

ΣΕΝΑΡΙΑ ΜΕ 15 ΣΤΑΤΙΚΟΥΣ ΚΟΜΒΟΥΣ
Σενάριο 1 - 15 κόμβοι:
Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):
topology=[0 1; 0 2; 0 3; 0 4; 2 4; 2 3; 4 3; 4 2; 6 3; 8 1; 8 2;8 3;8 4; 10 3; 12 3; 12.5 3; 12.5 3; 12.5 3; 12.5 3; 12.5 3; 12.5 3; 12.5 3; 12.5 3; 12.5 3; 12.5 3;];
Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 15, ο οποίος βρίσκεται στη θέση (12, 3).



Σχήμα 6.5 Τοπολογία του Σεναρίου 1 με 15 κόμβους

Σε αυτή την τοπολογία παρατηρούμε ότι ο κόμβος 7 είναι bottleneck γιατί οι ροές των κόμβων 5 και 6 στέλνονται σε αυτόν τον κόμβο και έτσι σύντομα θα είναι σε συμφόρηση.

Επίσης μετά τον κόμβο bottleneck παρατηρούμε ότι δημιουργείται γραμμή μέσω του κόμβου 9, και αναμένουμε ότι ο κόμβος 9 θα είναι επίσης σε συμφόρηση.

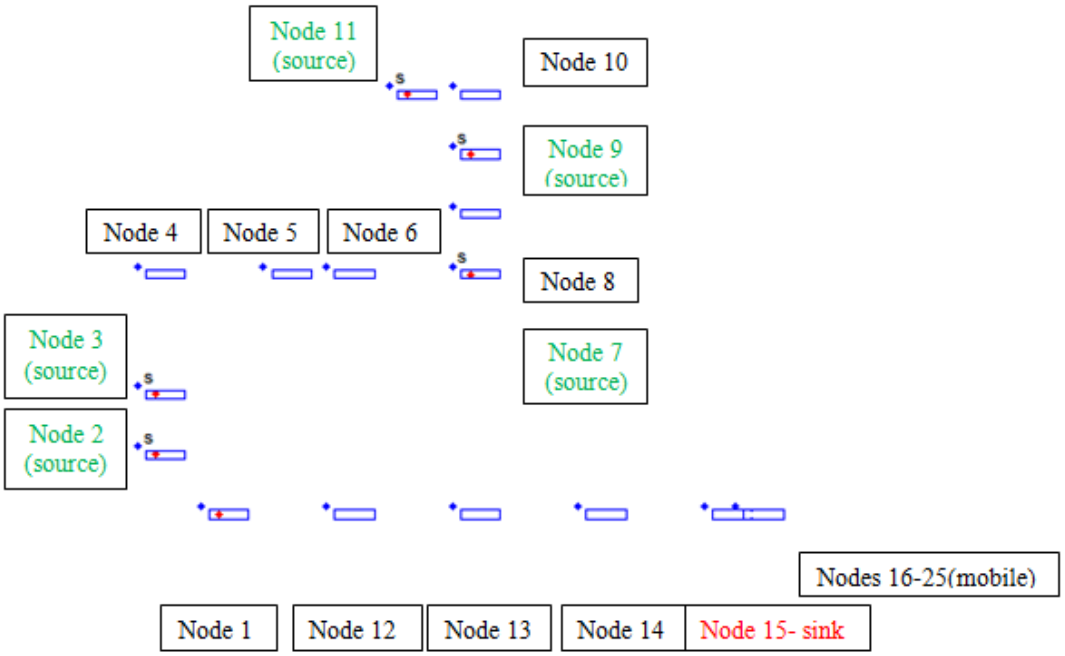
Ακόμη ο κόμβος 14 είναι bottleneck γιατί οι ροές των κόμβων 11, 12, 13 και 14 στέλνονται σε αυτόν τον κόμβο και έτσι σύντομα θα είναι σε συμφόρηση.

Σενάριο 2 - 15 κόμβοι:

Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[1 2; 0 3; 0 4; 0 6; 2 6; 3 6; 5 6; 5 7; 5 8; 5 9; 4 9; 3 2; 5 2; 7 2; 9 2; 9.5 2; 9.5 2; 9.5 2; 9.5 2; 9.5 2; 9.5 2; 9.5 2; 9.5 2; 9.5 2; 9.5 2];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 15, ο οποίος βρίσκεται στη θέση (9, 2).



Σχήμα 6.6 Τοπολογία του Σεναρίου 2 με 15 κόμβους

Σε αυτή την τοπολογία παρατηρούμε ότι ο κόμβος 1 είναι bottleneck γιατί οι ροές των κόμβων 2 και 3 στέλνονται σε αυτόν τον κόμβο και έτσι σύντομα θα είναι σε συμφόρηση.

Επίσης μετά τον κόμβο bottleneck παρατηρούμε ότι δημιουργείται γραμμή μέχρι τον κόμβο προορισμού μέσω των κόμβων 12 και 14, και αναμένουμε ότι οι κόμβοι αυτοί θα είναι επίσης σε συμφόρηση.

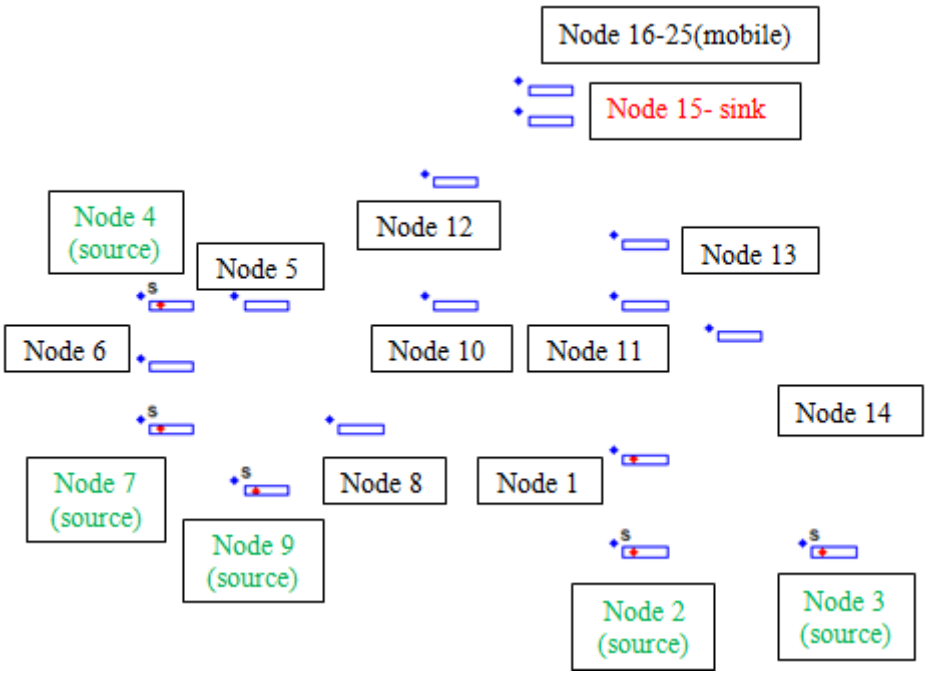
Το σενάριο αυτό είναι αντίστοιχο με το Σενάριο 2 με 10 κόμβους.

Σενάριο 3 – 15 κόμβοι:

Η τοπολογία του δικτύου πριν την εκτέλεση του αλγορίθμου είναι η εξής (ο i-οστός κόμβος είναι στην i-οστή θέση):

topology=[5 1.5; 5 0; 7 0;0 4;1 4; 0 3; 0 2; 2 2; 1 1;3 4;5 4; 3 6; 5 5;6 3.5; 4 7; 4 7.5; 4 7.5; 4 7.5; 4 7.5; 4 7.5; 4 7.5; 4 7.5; 4 7.5; 4 7.5; 4 7.5];

Ο κόμβος προορισμού του δικτύου είναι ο κόμβος 15, ο οποίος βρίσκεται στη θέση (4, 7).



Σχήμα 6.7 Τοπολογία του Σεναρίου 3 με 15 κόμβους

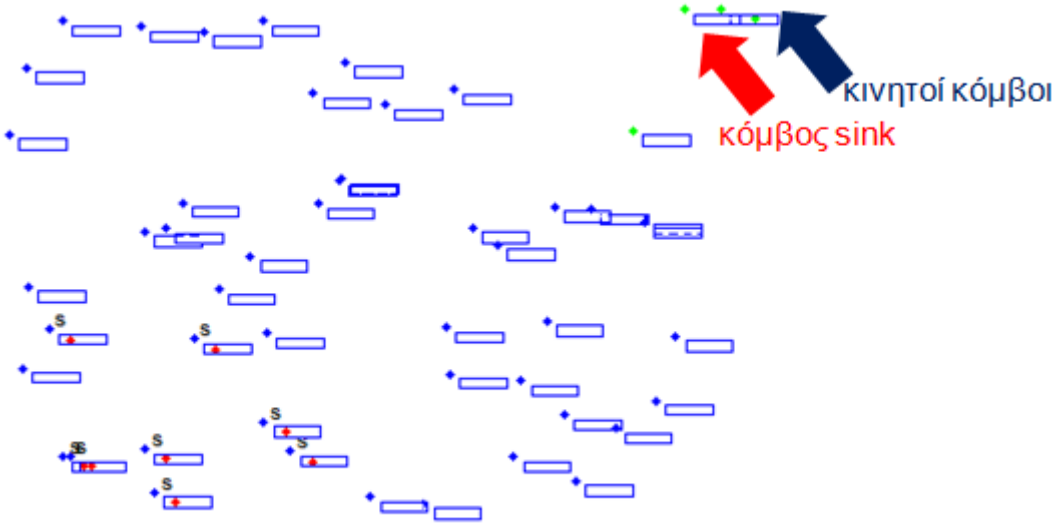
Σε αυτή την τοπολογία παρατηρούμε ότι ο κόμβος 1 είναι bottleneck γιατί οι ροές των κόμβων 2 και 3 στέλνονται σε αυτόν τον κόμβο και έτσι σύντομα θα είναι σε συμφόρηση. Επίσης παρατηρούμε ότι ο κόμβος 10 είναι bottleneck γιατί οι ροές των κόμβων 8 και 5 στέλνονται σε αυτόν τον κόμβο και έτσι σύντομα θα είναι σε συμφόρηση. Επίσης παρατηρούμε ότι ο κόμβος 13 είναι bottleneck γιατί οι ροές των κόμβων 11 και 14 στέλνονται σε αυτόν τον κόμβο και έτσι σύντομα θα είναι σε συμφόρηση.
Άρα εξετάζεται το σενάριο όπου τρία θέματα συμφόρησης εμφανίζονται σε τρία διαφορετικά μέρη του δικτύου.
Το σενάριο αυτό είναι αντίστοιχο με το Σενάριο 3 με 10 κόμβους.

6.3.2 Τυχαιοποιημένη Αξιολόγηση

Η τοπολογία που χρησιμοποιήθηκε για αυτές τις προσομοιώσεις είναι μια 10x10 ομοιόμορφα κατανεμημένη τοπολογία (uniform topology) με τυχαία απόκλιση τιμών ίση με 0,1 και στους δύο άξονες. Το περιβάλλον προσομοίωσης περιγράφεται στο υποκεφάλαιο 6.1.4 . Οι προσομοιώσεις έγιναν για 5

διαφορετικά πλήθη κόμβων: 50, 60, 70, 80, 90 και 100 κόμβους. Οι κόμβοι sources επιλέγονται χρησιμοποιώντας μια πιθανολογική λειτουργία στη κάτω αριστερή περιοχή του δικτύου. Ο κόμβος προορισμού τοποθετείται στην πάνω δεξιά γωνία, για να δημιουργούνται ροές που συγκλίνουν ή διασταυρώνουν. Στο Σχήμα 6.8 φαίνεται η ομοιόμορφα κατανεμημένη τοπολογία για 50 κόμβους. Οι κινητοί κόμβοι είναι τοποθετημένοι δίπλα από τον κόμβο προορισμού σε κατάσταση ύπνωσης. Η τοπολογία που επιλέχτηκε σε συνδυασμό με το μεγάλο ποσοστό των πηγαίων κόμβων (0.6), το μεγάλο αριθμό δημιουργίας πακέτων ανά δευτερόλεπτο (source rate), το οποίο όπως περιγράφεται πιο πριν είναι 4 και δημιουργεί δηλαδή 20 πακέτα ανά δευτερόλεπτο, και τη χωρητικότητα του buffer, η οποία είναι 15 πακέτα, οδηγεί στην άμεση δημιουργία θεμάτων συμφόρησης τα οποία δεν μπορεί να αντιμετωπίσει ο αλγόριθμος ελέγχου συμφόρησης που χρησιμοποιεί το δίκτυο κι έτσι καλείται ο αντίστοιχος αλγόριθμος MobileCC για επίλυση του συγκεκριμένου θέματος συμφόρησης.

Σημειώνεται ότι δεν έχει γίνει η τυχαιοποιημένη αξιολόγηση για τον αλγόριθμο Global MobileCC γιατί για δίκτυα με μεγάλο αριθμό κόμβων εφόσον δημιουργείται τυχαία η τοπολογία του δικτύου δεν υπάρχει εκ των προτέρων η γνώση των κόμβων που θα σημειώσουν συμφόρηση.



Σχήμα 6.8 Ομοιόμορφα κατανεμημένη τοπολογία για 50 κόμβους

6.4 Αποτελέσματα

6.4.1 Στοχευμένη Αξιολόγηση

Κατά τη στοχευμένη αξιολόγηση κατασκευαστήκαν κάποιες συγκεκριμένες τοπολογίες στις οποίες έχει ενδιαφέρον να ελέγξουμε τη συμπεριφορά των αλγορίθμων Dynamic MobileCC και Direct Path MobileCC, σε σχέση με τον αλγόριθμο Global MobileCC, ο οποίος όπως έχουμε προαναφέρει έχει σχεδιαστεί έτσι ώστε να είναι σε όλες τις περιπτώσεις βέλτιστος.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΜΕ 10 ΣΤΑΤΙΚΟΥΣ ΚΟΜΒΟΥΣ:

Σενάριο 1 :

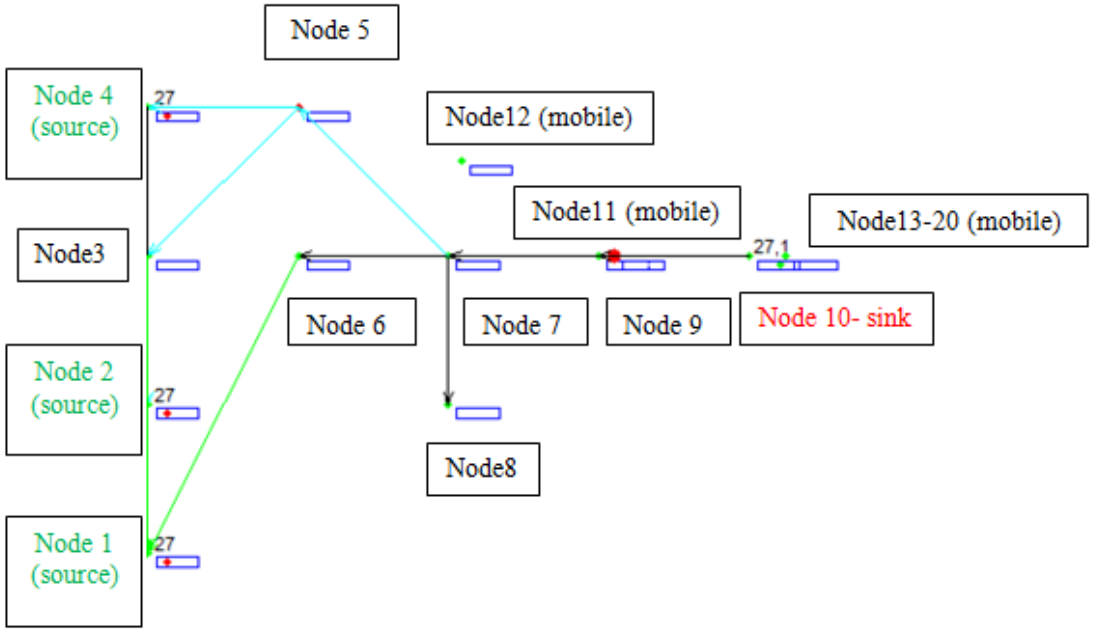
ΠΑΡΑΔΕΙΓΜΑ 1 – 10 ΚΟΜΒΟΙ

Ο κόμβος προορισμού βρίσκεται στη θέση (8, 3).

Οι κόμβοι 11-20, οι οποίοι βρίσκονται στη θέση (8.5, 3), δηλαδή δίπλα από τον κόμβο προορισμού, είναι κινητοί κόμβοι.

Σε αυτήν την τοπολογία ο αλγόριθμος DYNAMIC MOBILECC χρειάζεται τον ίδιο αριθμό κινητών κόμβων με τον αλγόριθμο GLOBAL MOBILECC, ενώ ο αλγόριθμος DIRECT PATH MOBILECC χρειάζεται το διπλάσιο αριθμό κινητών κόμβων.

DYNAMIC MOBILECC



Σχήμα 6.9 Τοπολογία μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC στο Σενάριο 1 με 10 κόμβους

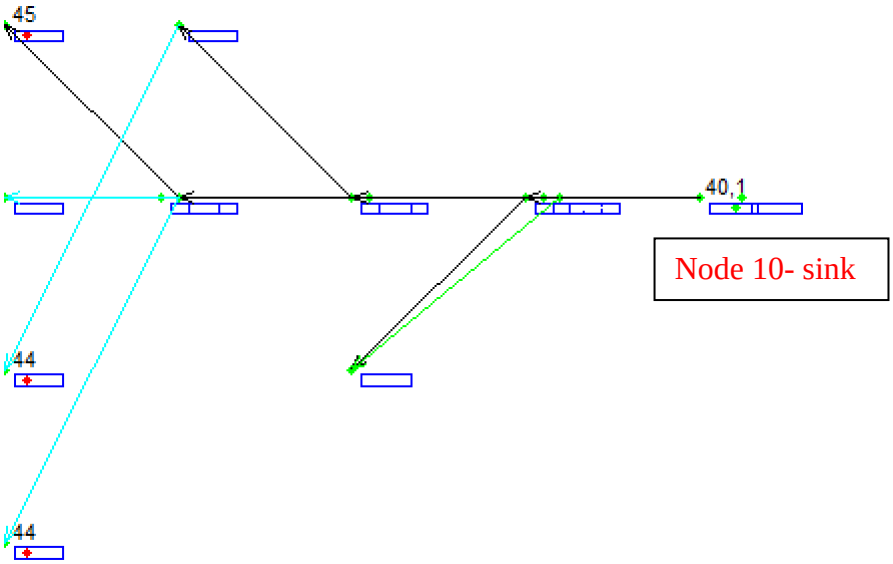
Χρησιμοποιούνται 2 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 7, 9.

Ο κινητός κόμβος 11, που είναι στη θέση (6.2, 3.0) , εξυπηρετεί τον κόμβο 7 που είναι γείτονας του κόμβου 9 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 12, που είναι στη θέση (4.1701, 3.6383) , εξυπηρετεί τον κόμβο 5 που είναι γείτονας του κόμβου 7 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 12 μπορεί να στέλνει τα δεδομένα του στον κινητό κόμβο 11.

DIRECT PATH MOBILECC



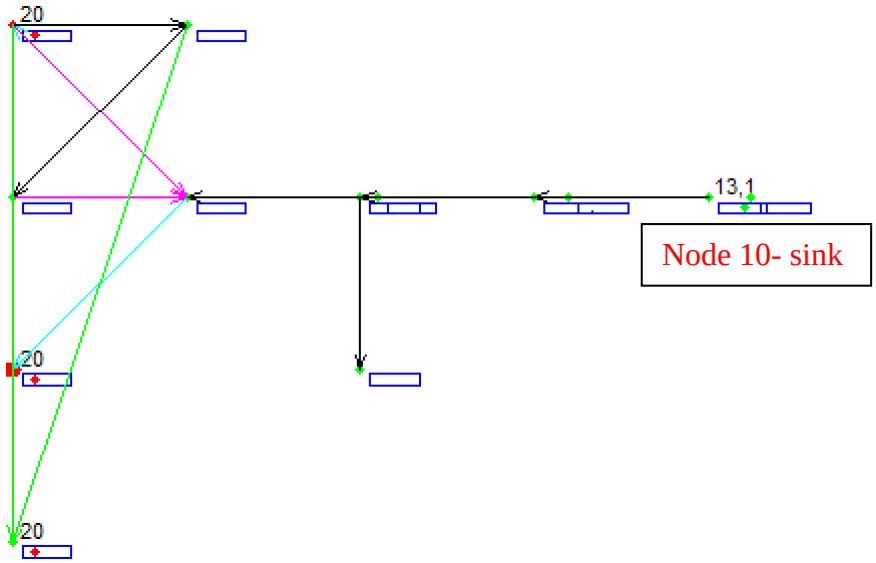
Σχήμα 6.10 Τοπολογία μετά την εκτέλεση του αλγορίθμου Direct Path MobileCC στο Σενάριο 1 με 10 κόμβους

Χρησιμοποιούνται 4 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 7, 9.

Ο κινητός κόμβος 11, που είναι στη θέση (6.2, 3.0) , εξυπηρετεί τον κόμβο 7 που είναι γείτονας του κόμβου 9 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 12, που είναι στη θέση (4.2, 3.0) , εξυπηρετεί τον κόμβο 6 που είναι γείτονας του κόμβου 7 ο οποίος είναι σε συμφόρηση.
Ο κινητός κόμβος 13, που είναι στη θέση (6.4, 3.0) , εξυπηρετεί τον κινητό κόμβο 12, για να δημιουργηθεί μονοπάτι από αυτόν στον κόμβο προορισμού.
Ο κινητός κόμβος 14, που είναι στη θέση (1.8, 3.0) , εξυπηρετεί τον κινητό κόμβο 11, για να δημιουργηθεί μονοπάτι από αυτόν στον κόμβο προορισμού.

GLOBAL MOBILECC



Σχήμα 6.11 Τοπολογία μετά την εκτέλεση του αλγορίθμου Global MobileCC στο Σενάριο 1 με 10 κόμβους

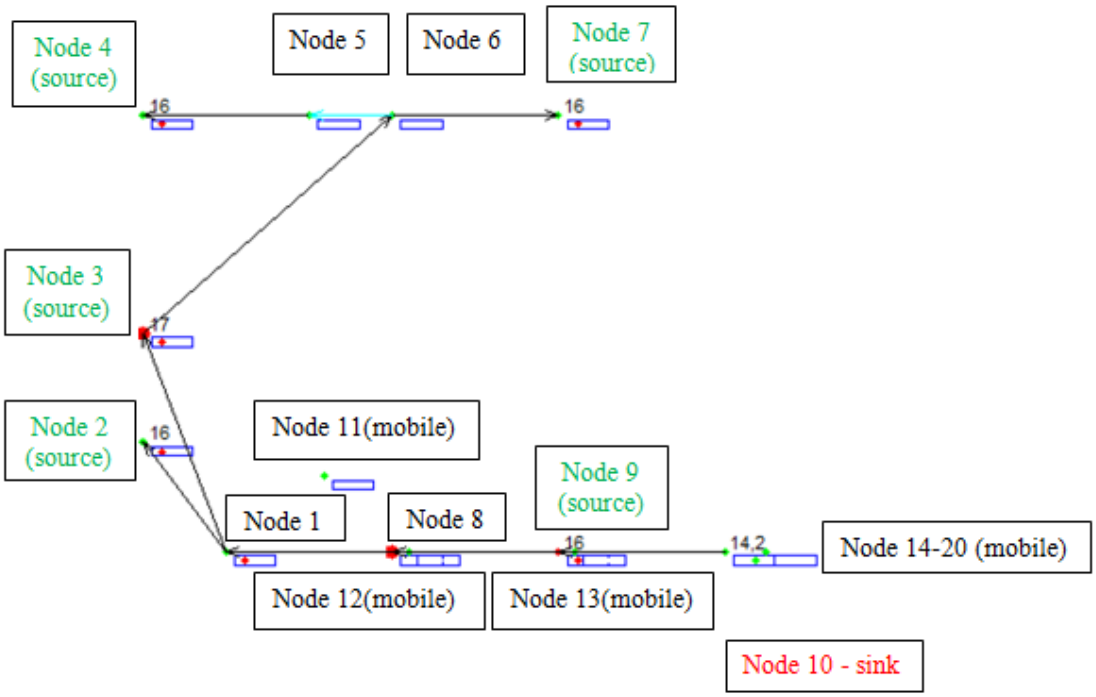
Χρησιμοποιούνται 2 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 7, 9.
Ο κινητός κόμβος 11, που είναι στη θέση (4.2, 3.0) , εξυπηρετεί τους κόμβους 6, 7 που είναι γείτονες των κόμβων 7, 9 οι οποίοι είναι σε συμφόρηση.
Ο κινητός κόμβος 12, που είναι στη θέση (6.4, 3.0) , εξυπηρετεί τον κινητό κόμβο 11, για να δημιουργηθεί μονοπάτι από αυτόν στον κόμβο προορισμού.

Σενάριο 2 :
ΠΑΡΑΔΕΙΓΜΑ 2 – 10 ΚΟΜΒΟΙ
Ο κόμβος προορισμού βρίσκεται στη θέση (7, 2)

Οι κόμβοι 11-20, οι οποίοι βρίσκονται στη θέση (7.5, 2), δηλαδή δίπλα από τον κόμβο προορισμού, είναι κινητοί κόμβοι.

Σε αυτήν την τοπολογία ο αλγόριθμος DYNAMIC MOBILECC χρειάζεται τον ίδιο αριθμό κινητών κόμβων με τον αλγόριθμο GLOBAL MOBILECC, ενώ ο αλγόριθμος DIRECT PATH MOBILECC χρειάζεται το διπλάσιο αριθμό κινητών κόμβων.

DYNAMIC MOBILECC

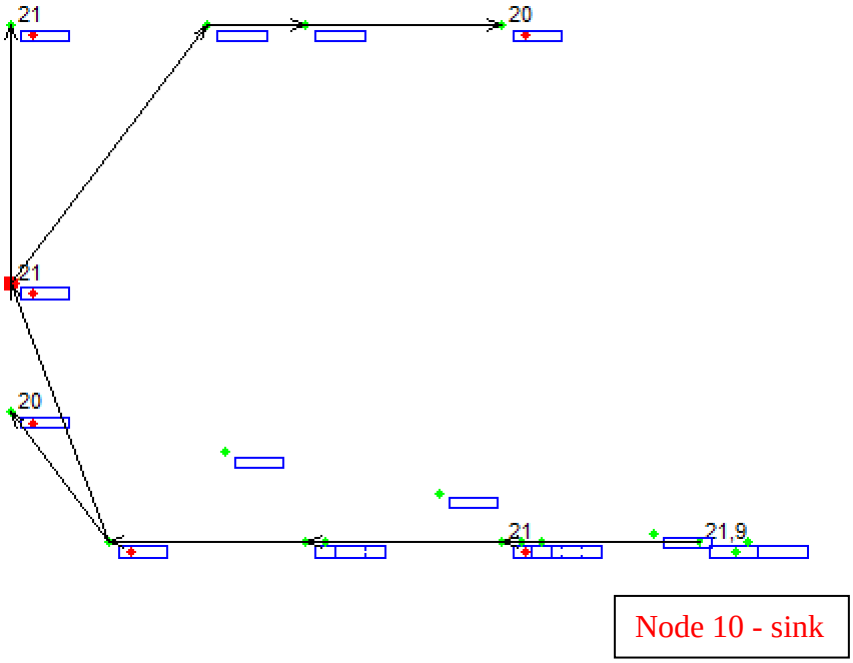


Σχήμα 6.12 Τοπολογία μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC στο Σενάριο 2 με 10 κόμβους

Χρησιμοποιούνται 3 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 1, 8, 9.
Ο κινητός κόμβος 11, που είναι στη θέση (2.1779, 2.6889) , εξυπηρετεί τον κόμβο 2 που είναι γείτονας του κόμβου 1 ο οποίος είναι σε συμφόρηση.
Ο κινητός κόμβος 12, που είναι στη θέση (3.2, 2.0) , εξυπηρετεί τον κόμβο 1 που είναι γείτονας του κόμβου 8 ο οποίος είναι σε συμφόρηση.
Ο κινητός κόμβος 13, που είναι στη θέση (5.2, 2.0) , εξυπηρετεί τον κόμβο 8 που είναι γείτονας του κόμβου 9 ο οποίος είναι σε συμφόρηση.

Παρόλο που ο αλγόριθμος λύνει κάθε πρόβλημα συμφόρησης τοπικά, παρατηρούμε ότι ο κινητός κόμβος 11 μπορεί να στέλνει τα δεδομένα του στον κινητό κόμβο 12, και ο κινητός κόμβος 12 μπορεί να στέλνει τα δεδομένα του στον κινητό κόμβο 13 και ο κινητός κόμβος 13 είναι στο εύρος αίσθησης του κόμβου προορισμού, για αυτό δεν είναι σε συμφόρηση κανένας από τους κινητούς κόμβους εφόσον καθένας μπορεί να στέλνει τα δεδομένα του σε κάποιον κόμβο.

DIRECT PATH MOBILECC



Σχήμα 6.13 Τοπολογία μετά την εκτέλεση του αλγορίθμου Direct Path MobileCC στο Σενάριο 2 με 10 κόμβους

Χρησιμοποιούνται 5 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 1, 8, 9.

Ο κινητός κόμβος 11, που είναι στη θέση (2.1779, 2.6889) , εξυπηρετεί τον κόμβο 2 που είναι γείτονας του κόμβου 1 ο οποίος είναι σε συμφόρηση.

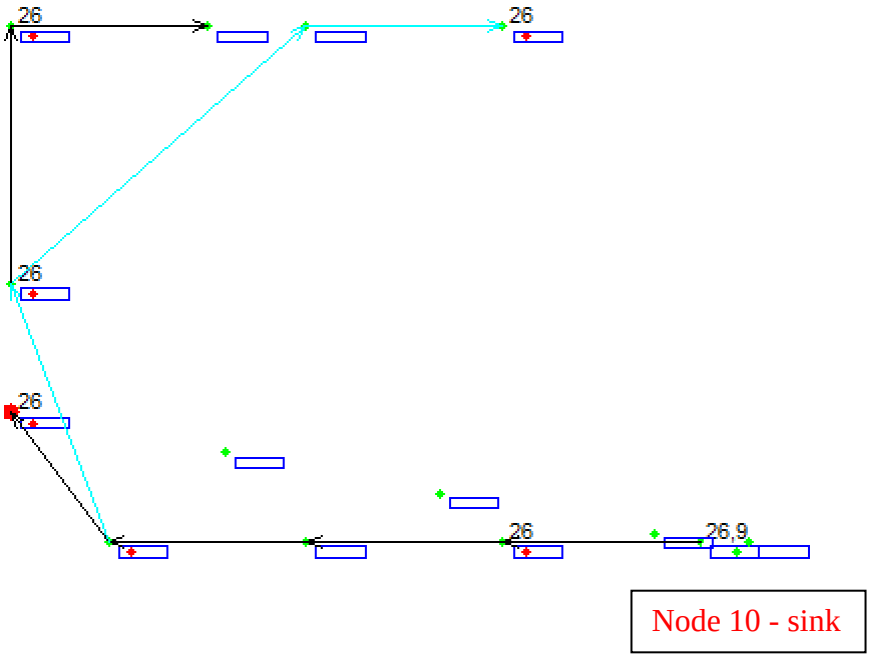
Ο κινητός κόμβος 12, που είναι στη θέση (4.3558, 2.3777) , εξυπηρετεί τον κόμβο 11, για να δημιουργηθεί μονοπάτι από τον κόμβο 11 στον κόμβο προορισμού.

Ο κινητός κόμβος 13, που είναι στη θέση (6.5337, 2.0666) , εξυπηρετεί τον κόμβο 12, για να δημιουργηθεί μονοπάτι από τον κόμβο 11 στον κόμβο προορισμού.

Ο κινητός κόμβος 14, που είναι στη θέση (5.2000, 2.0000) , εξυπηρετεί τον κόμβο 8 που είναι γείτονας του κόμβου 9 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 15, που είναι στη θέση (3.2, 2.0) , εξυπηρετεί τον κόμβο 1 που είναι γείτονας του κόμβου 8 ο οποίος είναι σε συμφόρηση.

GLOBAL MOBILECC



Σχήμα 6.14 Τοπολογία μετά την εκτέλεση του αλγορίθμου Global MobileCC στο Σενάριο 2 με 10 κόμβους

Χρησιμοποιούνται 3 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 1, 8, 9.

Ο κινητός κόμβος 11, που είναι στη θέση (2.1779, 2.6889) , εξυπηρετεί τους κόμβους 2, 1, 8 που είναι γείτονας των κόμβων 1, 8, 9 οι οποίοι είναι σε συμφόρηση.

Ο κινητός κόμβος 12, που είναι στη θέση (4.3558, 2.3777) , εξυπηρετεί τον κόμβο 11, για να δημιουργηθεί μονοπάτι από τον κόμβο 11 στον κόμβο προορισμού.

Ο κινητός κόμβος 13, που είναι στη θέση (6.5337, 2.0666) , εξυπηρετεί τον κόμβο 12, για να δημιουργηθεί μονοπάτι από τον κόμβο 11 στον κόμβο προορισμού.

Σενάριο 3:

ΠΑΡΑΔΕΙΓΜΑ 3 – 10 ΚΟΜΒΟΙ

Ο κόμβος προορισμού βρίσκεται στη θέση (4, 7).

Οι κόμβοι 11-20, οι οποίοι βρίσκονται στη θέση (4,7.5), δηλαδή δίπλα από τον κόμβο προορισμού, είναι κινητοί κόμβοι.

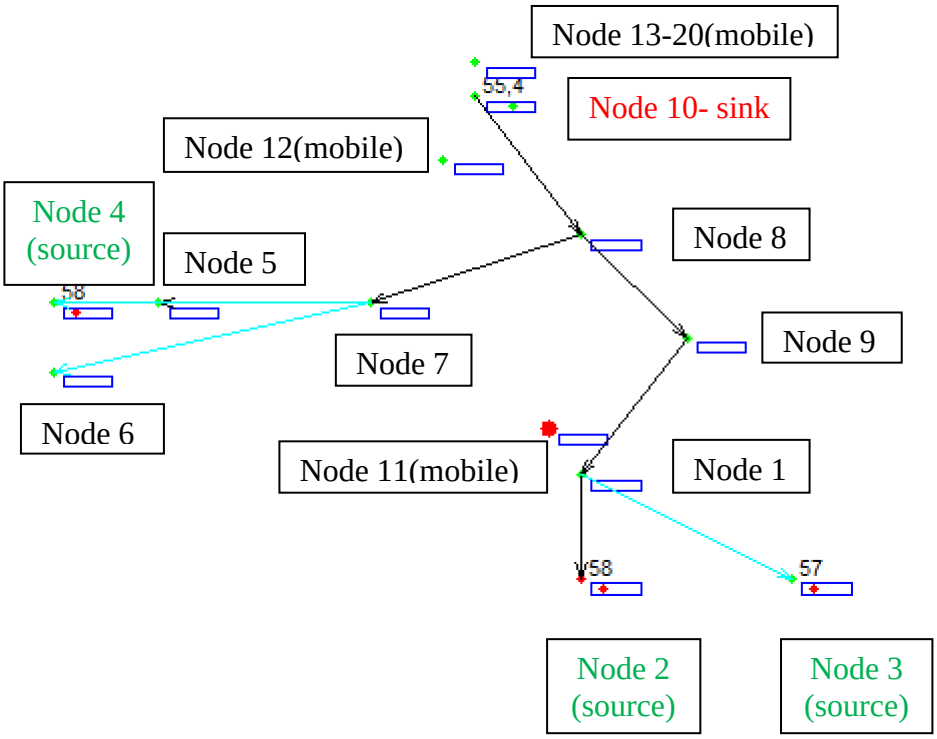
Ο DYNAMIC MOBILECC χρειάζεται τον ίδιο αριθμό κινητών κόμβων με τον αλγόριθμο DIRECT PATH MOBILECC.

Σε αυτήν την τοπολογία ο αλγόριθμος GLOBAL MOBILECC χρειάζεται λιγότερους κινητούς κόμβους από τους άλλους δυο αλγορίθμους.

Εφόσον με αυτή την τοπολογία εξετάζεται το σενάριο όπου δυο θέματα συμφόρησης εμφανίζονται σε δυο διαφορετικά μέρη του δικτύου, είναι αναμενόμενο ότι οι αλγόριθμοι DYNAMIC MOBILECC και DIRECT PATH MOBILECC θα επιλύσουν κάθε πρόβλημα συμφόρησης τοπικά και θα χρειαστούν τουλάχιστον ένα κινητό κόμβο για το καθένα, ενώ ο αλγόριθμος GLOBAL MOBILECC καταφέρνει να επιλύσει και τα δυο θέματα συμφόρησης τοποθετώντας μόνο ένα κινητό κόμβο, εφόσον βλέπει καθολικά όλα τα θέματα συμφόρησης του δικτύου.

Σε αυτό το σενάριο τυχαίνει ο DYNAMIC MOBILECC και oDIRECT PATH MOBILECC να χρειάζονται τον ίδιο αριθμό κινητών κόμβων γιατί και τα δυο θέματα συμφόρησης είναι κοντά στον κόμβο προορισμού κι έτσι δε χρειάζεται ο DIRECT PATH MOBILECC να δημιουργήσει μονοπάτι μέχρι τον κόμβο προορισμού.

DYNAMIC MOBILECC



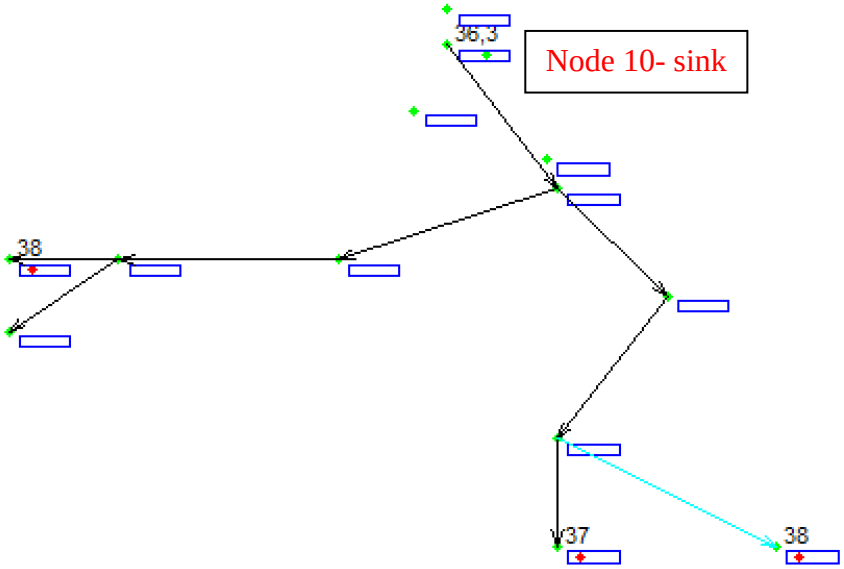
Σχήμα 6.15 Τοπολογία μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC στο Σενάριο 3 με 10 κόμβους

Χρησιμοποιούνται 2 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 1, 8.

Ο κινητός κόμβος 11, που είναι στη θέση (4.6889, 2.1779) , εξυπηρετεί τον κόμβο 2 που είναι γείτονας του κόμβου 1 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 12, που είναι στη θέση (3.6957, 6.0871) , εξυπηρετεί τον κόμβο 7 που είναι γείτονας του κόμβου 8 ο οποίος είναι σε συμφόρηση.

DIRECT PATH MOBILECC



Σχήμα 6.16 Τοπολογία μετά την εκτέλεση του αλγορίθμου Direct Path MobileCC στο Σενάριο 3 με 10 κόμβους

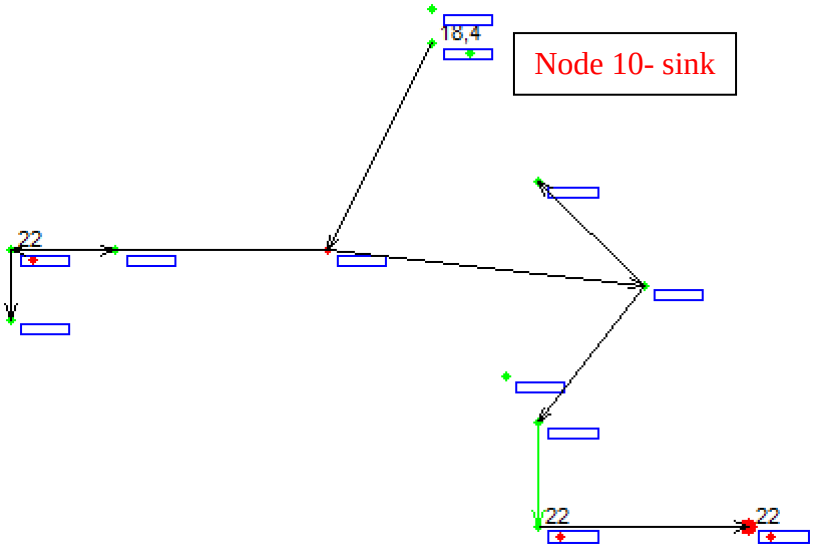
Χρησιμοποιούνται 2 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 8, 1.

Ο κινητός κόμβος 11, που είναι στη θέση (3.6957, 6.0871) , εξυπηρετεί τον κόμβο 7 που είναι γείτονας του κόμβου 8 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 12, που είναι στη θέση (4.9085, 5.4101) , εξυπηρετεί τον κόμβο 9 που είναι γείτονας του κόμβου 1 ο οποίος είναι σε συμφόρηση.

Δεν τοποθετούνται άλλοι κινητοί κόμβοι γιατί και οι δυο αυτοί κόμβοι είναι στο εύρος αίσθησης του κόμβου προορισμού.

GLOBAL MOBILECC



Σχήμα 6.17 Τοπολογία μετά την εκτέλεση του αλγορίθμου Global MobileCC στο Σενάριο 3 με 10 κόμβους

Χρησιμοποιείται μόνο 1 κινητός κόμβος. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 1, 8 .

Ο κινητός κόμβος 11, που είναι στη θέση (6.1334, 2.0221) , εξυπηρετεί τους κόμβους 6, 9 που είναι γείτονες των κόμβων 1, 8 οι οποίοι είναι σε συμφόρηση.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΜΕ 15 ΣΤΑΤΙΚΟΥΣ ΚΟΜΒΟΥΣ:

Σενάριο 1 - 15 κόμβοι:

ΠΑΡΑΔΕΙΓΜΑ 1 – 15 ΚΟΜΒΟΙ

Ο κόμβος προορισμού βρίσκεται στη θέση (12, 3).

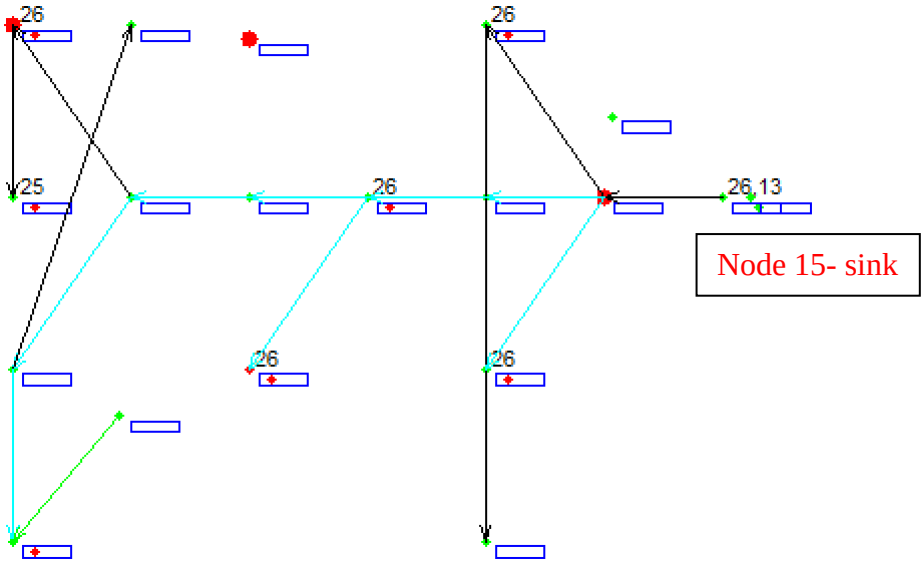
Οι κόμβοι 16-25, οι οποίοι βρίσκονται στη θέση (12.5, 3), δηλαδή δίπλα από τον κόμβο προορισμού, είναι κινητοί κόμβοι.

Σε αυτήν την τοπολογία ο αλγόριθμος GLOBAL MOBILECC χρειάζεται τον ίδιο αριθμό κινητών κόμβων με τον αλγόριθμο DYNAMIC MOBILECC.

Ο αλγόριθμος DIRECT MOBILECC χρειάζεται πολύ περισσότερους κινητούς κόμβους σε σχέση με τους άλλους δυο αλγόριθμους.

Οι παρατηρήσεις σε αυτό το σενάριο είναι οι ίδιες με το Σενάριο 1 με 10 κόμβους, εφόσον τα χαρακτηριστικά της τοπολογίας του δικτύου είναι τα ίδια.

DYNAMIC MOBILECC



Σχήμα 6.18 Τοπολογία μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC στο Σενάριο 1 με 15 κόμβους

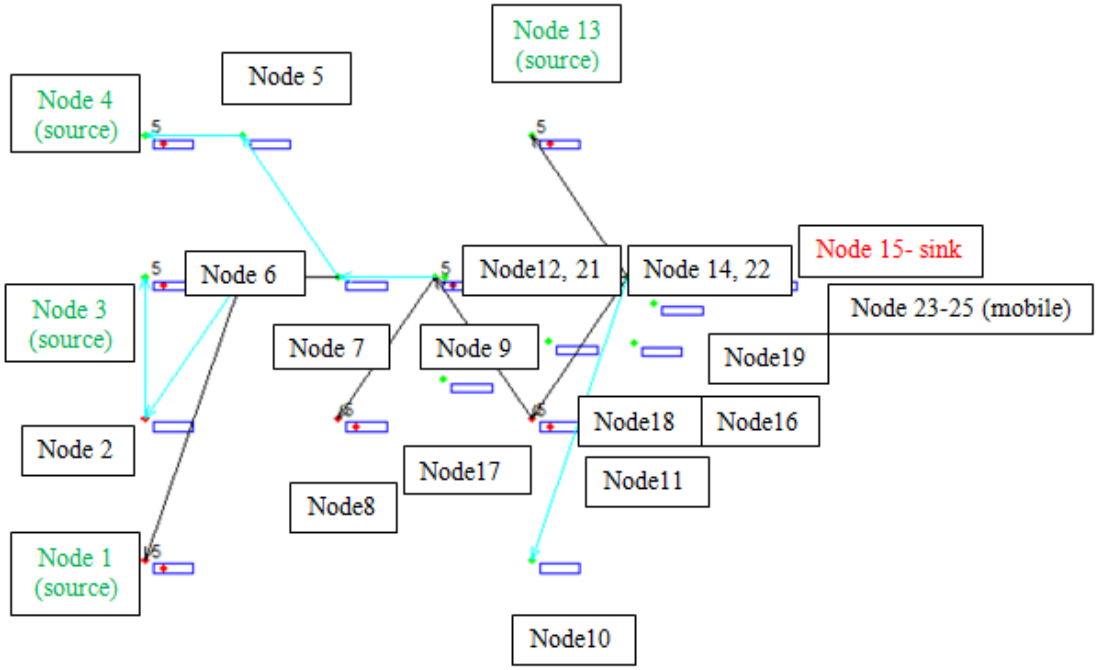
Χρησιμοποιούνται 3 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 7, 9, 14.

Ο κινητός κόμβος 15, που είναι στη θέση (10.1343, 3.4664) , εξυπηρετεί τον κόμβο 13 που είναι γείτονας του κόμβου 14 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 16, που είναι στη θέση (4.0, 3.9165) , εξυπηρετεί τον κόμβο 6 που είναι γείτονας του κόμβου 7 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 17, που είναι στη θέση (1.8170, 1.7271) , εξυπηρετεί τον κόμβο 8 που είναι γείτονας του κόμβου 9 ο οποίος είναι σε συμφόρηση.

DIRECT PATH MOBILECC



Σχήμα 6.19 Τοπολογία μετά την εκτέλεση του αλγορίθμου Direct Path MobileCC στο Σενάριο 1 με 15 κόμβους

Χρησιμοποιούνται 7 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 7, 9, 14.

Ο κινητός κόμβος 16, που είναι στη θέση (10.1343, 2.5336) , εξυπηρετεί τον κόμβο 11 που είναι γείτονας του κόμβου 14 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 17, που είναι στη θέση (6.183, 2.2729) , εξυπηρετεί τον κόμβο 8, που είναι γείτονας του κόμβου 7 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 18, που είναι στη θέση (8.366, 2.5458) , εξυπηρετεί τον κόμβο 17, για να δημιουργηθεί μονοπάτι από τον κόμβο 17 στον κόμβο προορισμού.

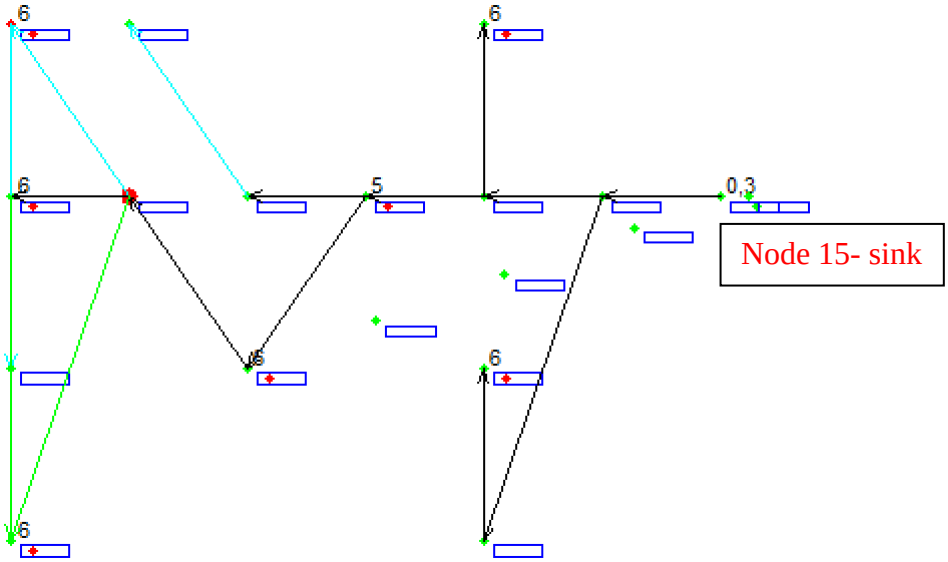
Ο κινητός κόμβος 19, που είναι στη θέση (10.549, 2.8186) , εξυπηρετεί τον κόμβο 17 για να δημιουργηθεί μονοπάτι από τον κόμβο 17 στον κόμβο προορισμού.

Ο κινητός κόμβος 20, που είναι στη θέση (6.2, 3.0) , εξυπηρετεί τον κόμβο 7 που είναι γείτονας του κόμβου 9 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 21, που είναι στη θέση (8.4, 3.0) , εξυπηρετεί τον κόμβο 20, για να δημιουργηθεί μονοπάτι από τον κόμβο 20 στον κόμβο προορισμού.

Ο κινητός κόμβος 22, που είναι στη θέση (10.6, 3.0) , εξυπηρετεί τον κόμβο 21, για να δημιουργηθεί μονοπάτι από τον κόμβο 20 στον κόμβο προορισμού.

GLOBAL MOBILECC



Σχήμα 6.20 Τοπολογία μετά την εκτέλεση του αλγορίθμου Global MobileCC στο Σενάριο 1 με 15 κόμβους

Χρησιμοποιούνται 3 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 7, 9, 14.

Ο κινητός κόμβος 16, που είναι στη θέση (6.1830, 2.2729) , εξυπηρετεί τους κόμβους 7, 8, 12 που είναι γείτονες των κόμβων 9, 7, 14 οι οποίοι είναι σε συμφόρηση.

Ο κινητός κόμβος 17, που είναι στη θέση (8.366, 2.5458) , εξυπηρετεί τον κινητό κόμβο 16, για να δημιουργηθεί μονοπάτι από τον κινητό κόμβο 16 στον κόμβο προορισμού.

Ο κινητός κόμβος 18, που είναι στη θέση (10.5490, 2.8186) , εξυπηρετεί τον κινητό κόμβο 17, για να δημιουργηθεί μονοπάτι από τον κινητό κόμβο 16 στον κόμβο προορισμού.

Σενάριο 2 - 15 κόμβοι:

ΠΑΡΑΔΕΙΓΜΑ 2 – 15 ΚΟΜΒΟΙ

Ο κόμβος προορισμού βρίσκεται στη θέση (9, 2).

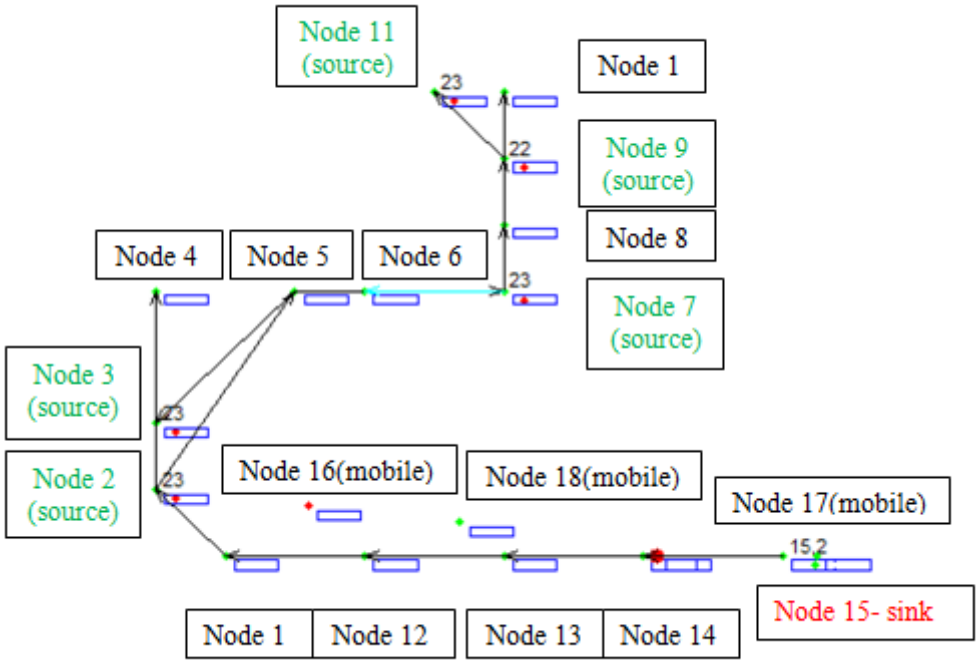
Οι κόμβοι 16-25, οι οποίοι βρίσκονται στη θέση (9.5, 2), δηλαδή δίπλα από τον κόμβο προορισμού, είναι κινητοί κόμβοι.

Σε αυτήν την τοπολογία ο αλγόριθμος GLOBAL MOBILECC χρειάζεται τον ίδιο αριθμό κινητών κόμβων με τον αλγόριθμο DYNAMIC MOBILECC.

Ο αλγόριθμος DIRECT MOBILECC χρειάζεται πολύ περισσότερους κινητούς κόμβους σε σχέση με τους άλλους δυο αλγόριθμους.

Οι παρατηρήσεις σε αυτό το σενάριο είναι οι ίδιες με το Σενάριο 2 με 10 κόμβους, εφόσον τα χαρακτηριστικά της τοπολογίας του δικτύου είναι τα ίδια.

DYNAMIC MOBILECC

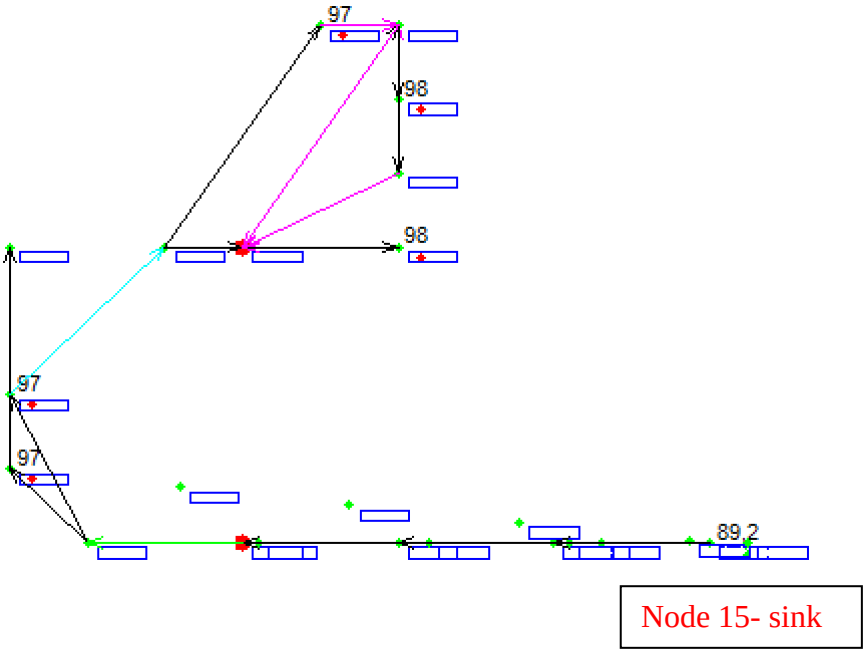


Σχήμα 6.21 Τοπολογία μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC στο Σενάριο 2 με 15 κόμβους

Χρησιμοποιούνται 3 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 1, 14, 12.

Ο κινητός κόμβος 16, που είναι στη θέση (2.1865, 2.7571) , εξυπηρετεί τον κόμβο 2 που είναι γείτονας του κόμβου 1 ο οποίος είναι σε συμφόρηση.
Ο κινητός κόμβος 17, που είναι στη θέση (7.2, 2.0) , εξυπηρετεί τον κόμβο 13 που είναι γείτονας του κόμβου 14 ο οποίος είναι σε συμφόρηση.
Ο κινητός κόμβος 18, που είναι στη θέση (4.3731, 2.5141) , εξυπηρετεί τον κόμβο 16 που είναι γείτονας του κόμβου 12 ο οποίος είναι σε συμφόρηση.

DIRECT PATH MOBILECC



Σχήμα 6.22 Τοπολογία μετά την εκτέλεση του αλγορίθμου Direct Path MobileCC στο Σενάριο 2 με 15 κόμβους

Χρησιμοποιούνται 8 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 1, 14, 12.
Ο κινητός κόμβος 16, που είναι στη θέση (3.2, 2.0) , εξυπηρετεί τον κόμβο 1 που είναι γείτονας του κόμβου 12 ο οποίος είναι σε συμφόρηση.
Ο κινητός κόμβος 17, που είναι στη θέση (5.4, 2.0) , εξυπηρετεί τον κόμβο 16, για να δημιουργηθεί μονοπάτι από τον κόμβο 16 στον κόμβο προορισμού.
Ο κινητός κόμβος 18, που είναι στη θέση (7.6, 2.0) , εξυπηρετεί τον κόμβο 17, για να δημιουργηθεί μονοπάτι από τον κόμβο 16 στον κόμβο προορισμού.

Ο κινητός κόμβος 19, που είναι στη θέση (7.2, 2.0) , εξυπηρετεί τον κόμβο 13 που είναι γείτονας του κόμβου 14 ο οποίος είναι σε συμφόρηση.

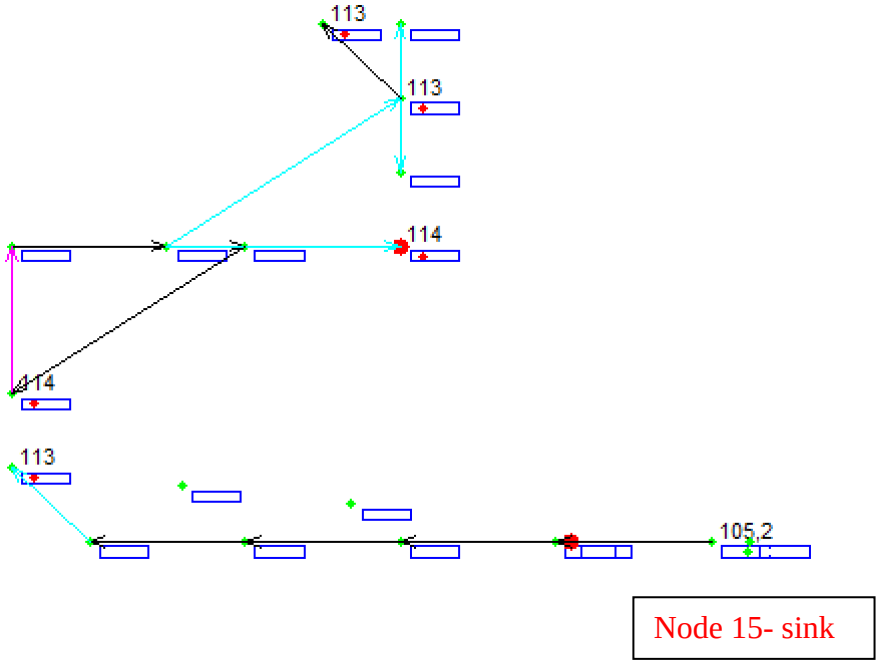
Ο κινητός κόμβος 20, που είναι στη θέση (2.1865, 2.7571) , εξυπηρετεί τον κόμβο 2 που είναι γείτονας του κόμβου 1 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 21, που είναι στη θέση (4.3731, 2.5141) , εξυπηρετεί τον κόμβο 20, για να δημιουργηθεί μονοπάτι από τον κόμβο 20 στον κόμβο προορισμού.

Ο κινητός κόμβος 22, που είναι στη θέση (6.5596, 2.2712) , εξυπηρετεί τον κόμβο 21, για να δημιουργηθεί μονοπάτι από τον κόμβο 20 στον κόμβο προορισμού.

Ο κινητός κόμβος 23, που είναι στη θέση (8.7462, 2.0282) , εξυπηρετεί τον κόμβο 22, για να δημιουργηθεί μονοπάτι από τον κόμβο 20 στον κόμβο προορισμού.

GLOBAL MOBILECC



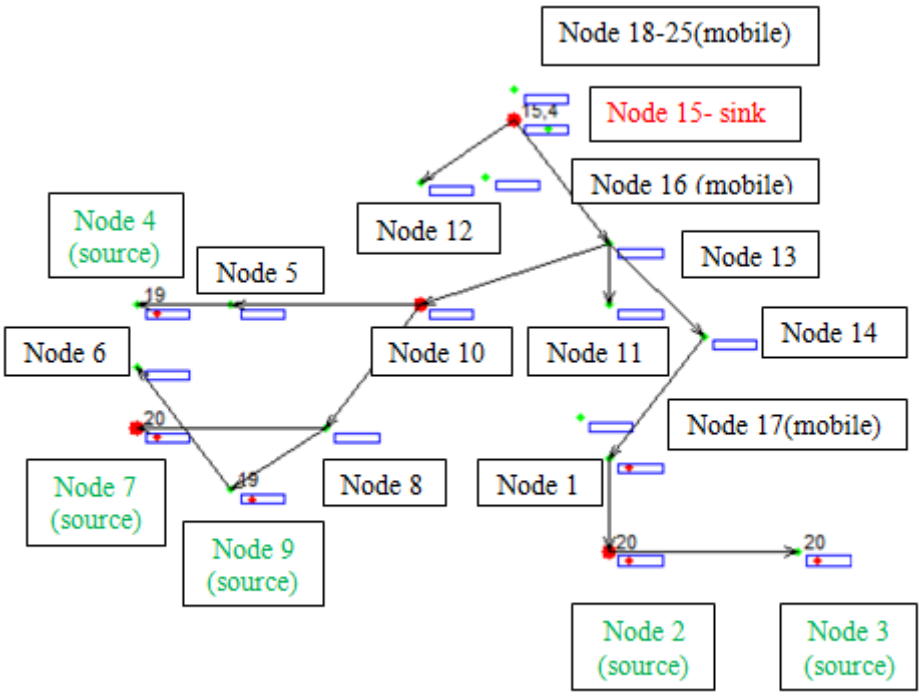
Σχήμα 6.23 Τοπολογία μετά την εκτέλεση του αλγορίθμου Global MobileCC στο Σενάριο 2 με 15 κόμβους

Χρησιμοποιούνται 3 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 1, 14, 12.
Ο κινητός κόμβος 16, που είναι στη θέση (2.1865, 2.7571) , εξυπηρετεί τους κόμβους 1, 2 που είναι γείτονες των κόμβων 1,12 οι οποίοι είναι σε συμφόρηση.
Ο κινητός κόμβος 17, που είναι στη θέση (7.2, 2.0) , εξυπηρετεί τον κόμβο 13, που είναι γείτονας του κόμβου 14 ο οποίος είναι σε συμφόρηση.
Ο κινητός κόμβος 18, που είναι στη θέση (4.3731, 2.5141) , εξυπηρετεί τον κινητό κόμβο 16, γιατί ο κόμβος 16 δεν μπορεί να βρει άλλο γείτονα που να μην είναι σε συμφόρηση για να στέλνει τα δεδομένα του έτσι ώστε να φθάσουν στον κόμβο προορισμού. Ο κινητός κόμβος 18 μπορεί να στέλνει τα δεδομένα του στον κόμβο 13, ο οποίος βρίσκεται στη θέση (5, 2) και στέλνει τα δεδομένα του στον κινητό κόμβο 17.

Σενάριο 3 - 15 κόμβοι:
ΠΑΡΑΔΕΙΓΜΑ 3 – 15 ΚΟΜΒΟΙ

Ο κόμβος προορισμού βρίσκεται στη θέση (4, 7).
Οι κόμβοι 16-25, οι οποίοι βρίσκονται στη θέση (4, 7.5), δηλαδή δίπλα από τον κόμβο προορισμού, είναι κινητοί κόμβοι.
Σε αυτήν την τοπολογία ο αλγόριθμος DYNAMIC MOBILECC χρειάζεται λιγότερους κινητούς κόμβους από τον αλγόριθμο DIRECT PATH MOBILECC.
Ο αλγόριθμος GLOBAL MOBILECC χρειάζεται λιγότερους κινητούς κόμβους από τους δυο άλλους αλγορίθμους.
Οι παρατηρήσεις σε αυτό το σενάριο είναι οι ίδιες με το Σενάριο 3 με 10 κόμβους, εφόσον τα χαρακτηριστικά της τοπολογίας του δικτύου είναι τα ίδια.
Σε αυτό το σενάριο όμως δε χρειάζονται τον ίδιο αριθμό κόμβων οι αλγορίθμου DYNAMIC MOBILECC και DIRECT PATH MOBILECC επειδή τα θέματα συμφόρησης που εμφανίζονται στα δίκτυα δεν είναι όλα κοντά στον κόμβο προορισμού, όπως συνέβαινε στο Σενάριο 3 με 10 κόμβους και συνεπώς ο αλγόριθμος DIRECT PATH MOBILECC χρειάζεται περισσότερους κινητούς κόμβους για να δημιουργήσει ολόκληρο το μονοπάτι.

DYNAMIC MOBILECC



Σχήμα 6.24 Τοπολογία μετά την εκτέλεση του αλγορίθμου Dynamic MobileCC στο Σενάριο 3 με 15 κόμβους

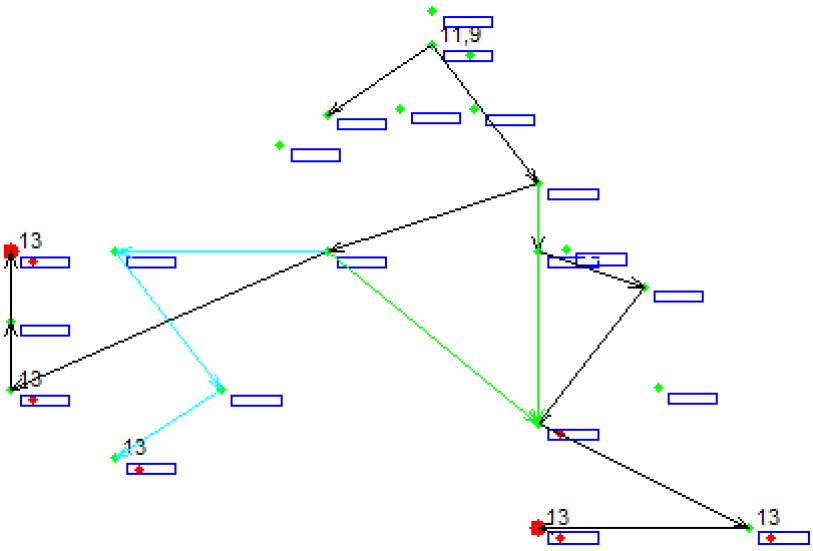
Χρησιμοποιούνται 3 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 13, 1, 10.

Ο κινητός κόμβος 16, που είναι στη θέση (3.6957, 6.0871) , εξυπηρετεί τον κόμβο 10 που είναι γείτονας του κόμβου 13 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 17, που είναι στη θέση (4.6889, 2.1779) , εξυπηρετεί τον κόμβο 2 που είναι γείτονας του κόμβου 1 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 18, που είναι στη θέση (4.3777, 4.3558) , εξυπηρετεί τον κόμβο 17 που είναι γείτονας του κόμβου 12 ο οποίος είναι σε συμφόρηση.

DIRECT PATH MOBILECC



Σχήμα 6.25 Τοπολογία μετά την εκτέλεση του αλγορίθμου Direct Path MobileCC στο Σενάριο 3 με 15 κόμβους

Χρησιμοποιούνται 5 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 10, 13, 1.

Ο κινητός κόμβος 16, που είναι στη θέση (2.5556, 5.5556) , εξυπηρετεί τον κόμβο 5 που είναι γείτονας του κόμβου 10 ο οποίος είναι σε συμφόρηση.

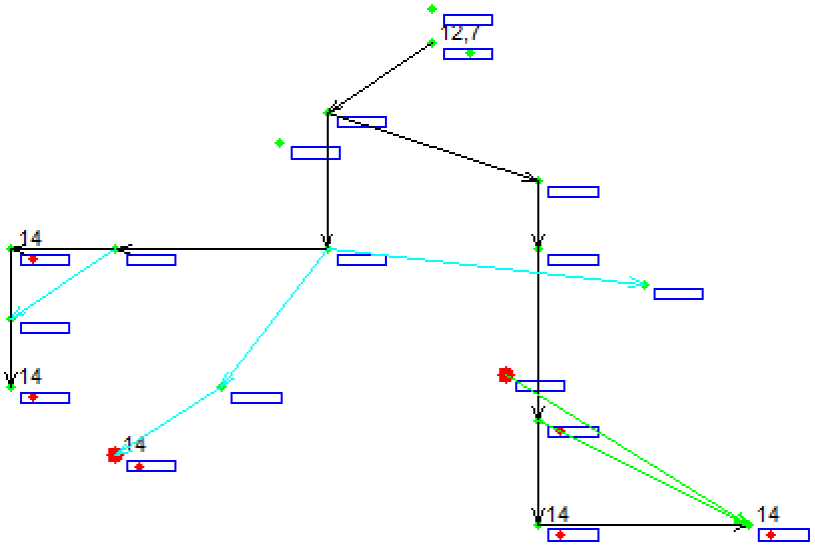
Ο κινητός κόμβος 17, που είναι στη θέση (3.6957, 6.0871) , εξυπηρετεί τον κόμβο 10 που είναι γείτονας του κόμβου 13 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 18, που είναι στη θέση (6.1334, 2.0221) , εξυπηρετεί τον κόμβο 3 που είναι γείτονας του κόμβου 1 ο οποίος είναι σε συμφόρηση.

Ο κινητός κόμβος 19, που είναι στη θέση (5.2668, 4.0442) , εξυπηρετεί τον κινητό κόμβο 18, για να δημιουργηθεί μονοπάτι από αυτόν στον κόμβο προορισμού.

Ο κινητός κόμβος 20, που είναι στη θέση (4.4001, 6.0664) , εξυπηρετεί τον κινητό κόμβο 19, για να δημιουργηθεί μονοπάτι από τον κινητό κόμβο 18 στον κόμβο προορισμού.

GLOBAL MOBILECC



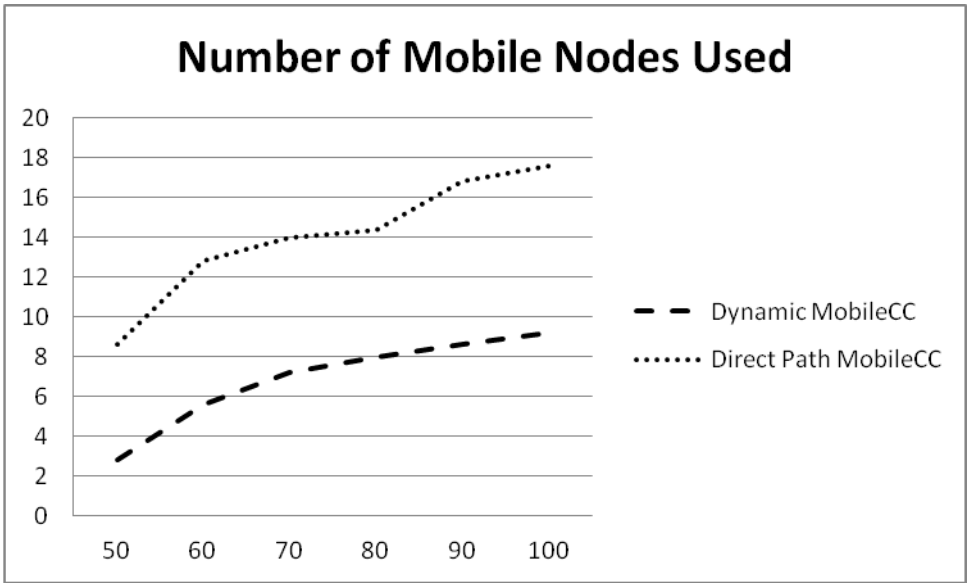
Σχήμα 6.26 Τοπολογία μετά την εκτέλεση του αλγορίθμου Global MobileCC στο Σενάριο 3 με 15 κόμβους

Χρησιμοποιούνται 2 κινητοί κόμβοι. Οι κόμβοι που είναι σε συμφόρηση είναι οι εξής: 1, 13, 10.

Ο κινητός κόμβος 11, που είναι στη θέση (4.6889, 2.1779) , εξυπηρετεί τους κόμβους 1,2 που είναι γείτονες των κόμβων 1,13 οι οποίοι είναι σε συμφόρηση.

Ο κινητός κόμβος 12, που είναι στη θέση (2.5556, 5.5556) , εξυπηρετεί τον κινητό κόμβο 5, που είναι γείτονας του κόμβου 10 ο οποίος είναι σε συμφόρηση.

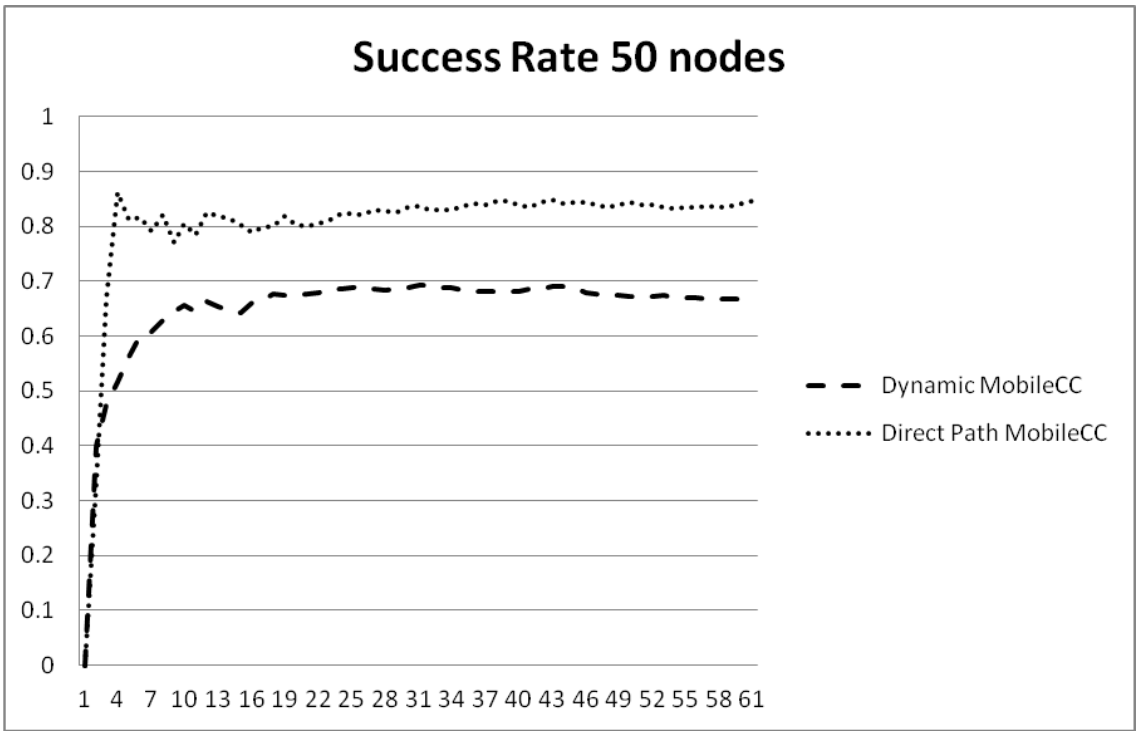
6.4.2 Τυχαιοποιημένη Αξιολόγηση



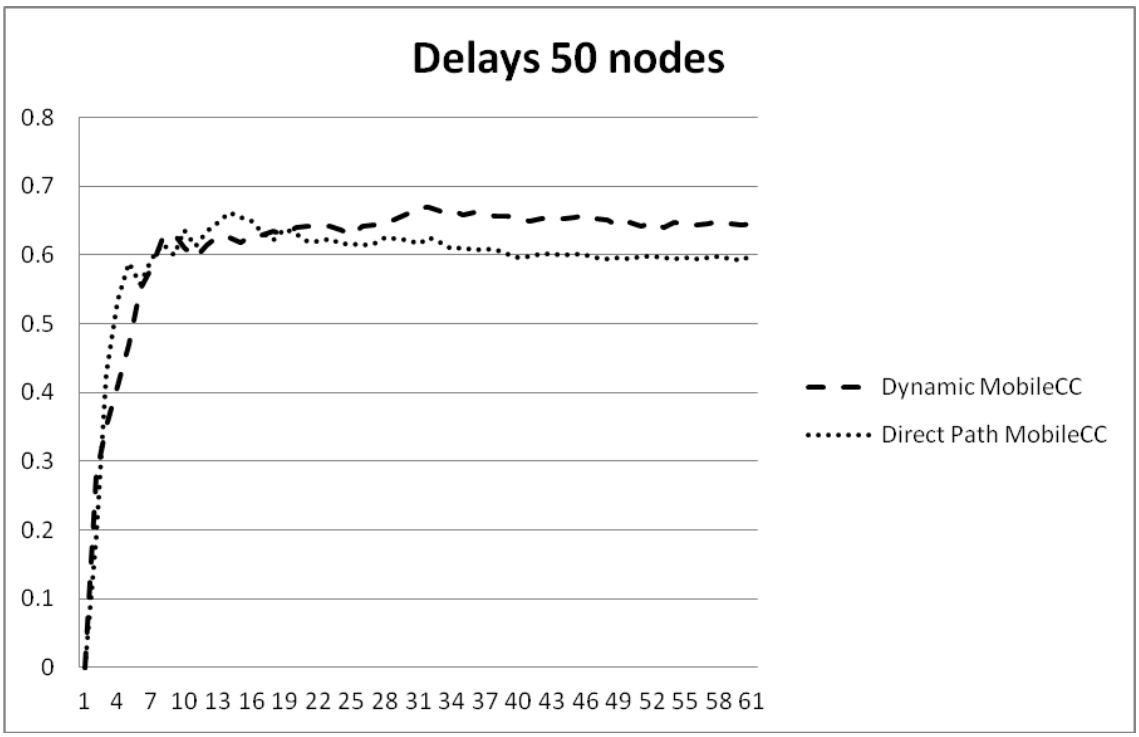
Σχήμα 6.27 Γραφική που περιγράφει τον αριθμό των κινητών κόμβων που χρησιμοποιούν οι αλγόριθμοι Dynamic MobileCC και Direct Path MobileCC

Στην πιο πάνω γραφική φαίνεται ο μέσος όρος του αριθμού των κινητών κόμβων που χρησιμοποιούν οι αλγόριθμοι Dynamic MobileCC και Direct Path MobileCC. Παρατηρούμε ότι ο αλγόριθμος Direct Path MobileCC χρησιμοποιεί τουλάχιστον τους διπλάσιους κινητούς κόμβους που χρειάζεται ο αλγόριθμος Dynamic MobileCC. Εφόσον η επίδοση των δυο αλγορίθμων είναι εξίσου καλή μπορούμε να συμπεραίνουμε ότι ο αλγόριθμος Direct Path MobileCC χρησιμοποιεί κινητούς κόμβους οι οποίοι πραγματικά δεν χρειάζονται για να αντιμετωπιστεί η συμφόρηση και το δίκτυο να συνεχίσει να λειτουργεί κανονικά.

Επίσης και για τους δυο αλγορίθμους μπορούμε να καταλήξουμε στο συμπέρασμα ότι όσο αυξάνεται το μέγεθος του δικτύου ο αριθμός των κινητών κόμβων που χρησιμοποιούνται αυξάνεται υπογραμμικά. Αυτό μας δείχνει ότι όσο αυξάνεται το μέγεθος του δικτύου η αύξηση των κινητών κόμβων που χρειάζονται είναι μικρή.



Σχήμα 6.28 Γραφική που περιγράφει το ποσοστό επιτυχίας (success rate) για δίκτυα με 50 κόμβους



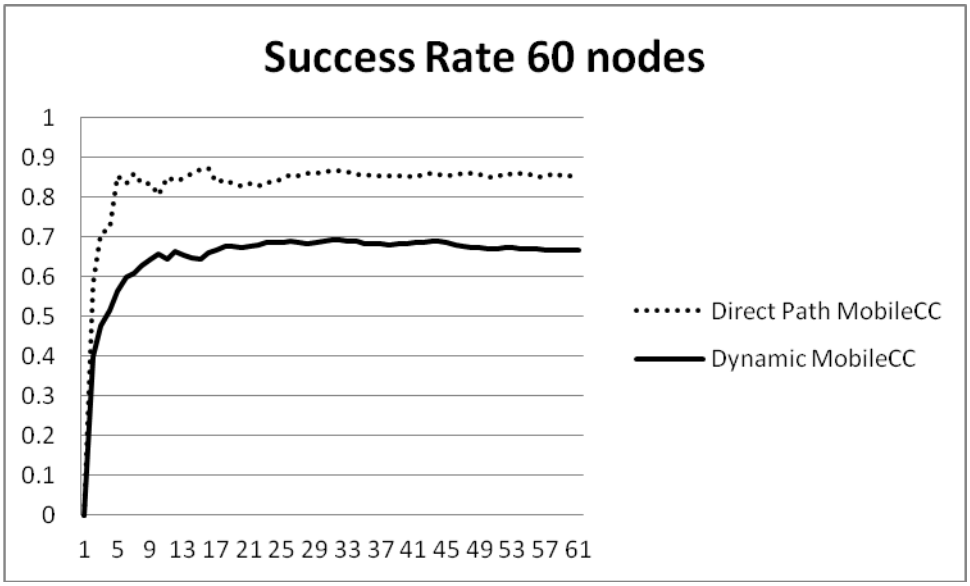
Σχήμα 6.29 Γραφική που περιγράφει τις καθυστερήσεις (delays) για δίκτυα με 50 κόμβους

Παρατηρούμε ότι για τα δίκτυα με 50 κόμβους ο συνολικός αριθμός των πακέτων που λαμβάνει ο κόμβος προορισμού έναντι του συνολικού αριθμού των πακέτων που αποστέλλονται από όλους τους πηγαίους κόμβους, δηλαδή το success rate, είναι πολύ ψηλό και για τους δυο αλγορίθμους, με το success rate του αλγορίθμου Direct Path MobileCC να είναι λίγο καλύτερο από του αλγορίθμου Dynamic MobileCC, το οποίο είναι αναμενόμενο επειδή ο αλγόριθμος Direct Path MobileCC χρησιμοποιεί τουλάχιστον τους διπλάσιους κινητούς κόμβους σε σχέση με τον αλγόριθμο Dynamic MobileCC έτσι παρέχει στο δίκτυο επιπλέον πόρους τους οποίους εκμεταλλεύεται για τη δημιουργία εναλλακτικών μονοπατιών, και με αυτό τον τρόπο μπορεί να αντιμετωπίσει μελλοντικά θέματα συμφόρησης, ενώ ο αλγόριθμος Dynamic MobileCC χρησιμοποιεί μόνο ένα κινητό κόμβο για κάθε θέμα συμφόρησης που παρουσιάζεται. Επίσης το μονοπάτι που δημιουργεί ο αλγόριθμος Direct Path MobileCC χρησιμοποιείται και από άλλους πηγαίους κόμβους, εφόσον στα event-based δίκτυα όλοι οι πηγαίοι κόμβοι είναι στην ίδια περιοχή του δικτύου.

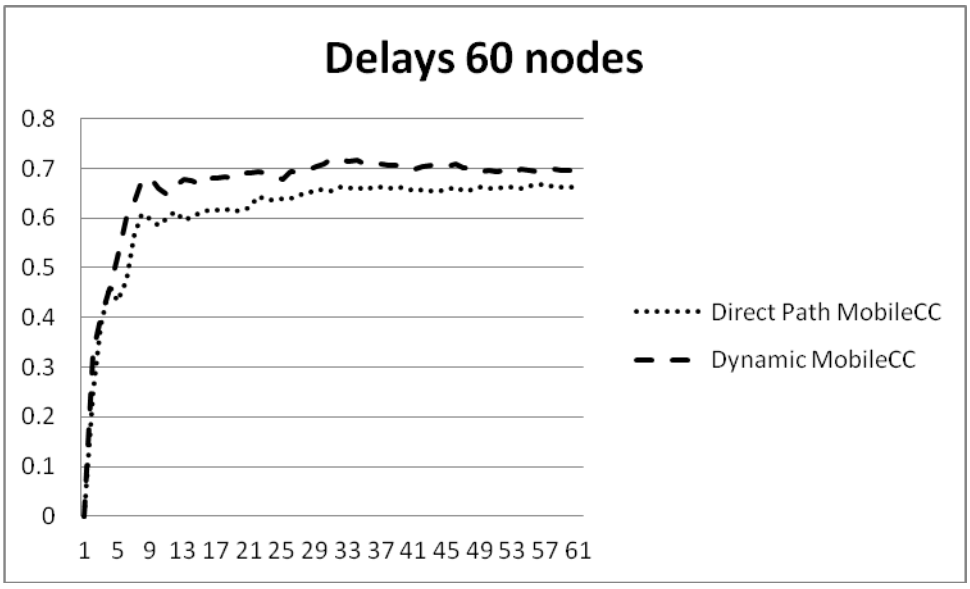
Όσο αφορά τον αλγόριθμο Direct Path MobileCC παρατηρούμε ότι το success rate σταθεροποιείται πιο σύντομα σε σχέση με τον αλγόριθμο Dynamic MobileCC. Αυτό συμβαίνει λόγω της λειτουργίας του αλγόριθμου Direct Path MobileCC, δηλαδή λόγω του ότι δημιουργεί ολόκληρο το μονοπάτι που αρχίζει από τους γείτονες του κόμβου που είναι σε συμφόρηση και τελειώνει στον κόμβο προορισμού, ενώ ο αλγόριθμος Dynamic MobileCC τοποθετεί μόνο ένα κινητό κόμβο για κάθε θέμα συμφόρησης που παρουσιάζεται στο δίκτυο και έτσι χρειάζεται περισσότερο χρόνο για να δημιουργήσει τα εναλλακτικά μονοπάτια από τους γείτονες των κόμβων που είναι σε συμφόρηση στον κόμβο προορισμού (γιατί τοποθετώντας ένα κινητό κόμβο για να επιλυθεί ένα πρόβλημα συμφόρησης τοπικά μπορεί να δημιουργήσει ένα άλλο πρόβλημα στο μονοπάτι που θα χρησιμοποιεί ο νέος κινητός κόμβος που θα τοποθετηθεί για να στέλνει τα δεδομένα που λαμβάνει στον κόμβο προορισμού ή μπορεί επίσης να μην υπάρχει κανένας κόμβος που να μην είναι σε συμφόρηση για στέλνει τα δεδομένα του ο κινητός κόμβος και σε αυτήν την περίπτωση θα πρέπει να τοποθετηθεί ένας επιπλέον κινητός κόμβος). Επίσης το μονοπάτι

που δημιουργεί ο αλγόριθμος Direct Path MobileCC αποτελείται από το μικρότερο αριθμό βημάτων (hops), γιατί κάθε κινητός κόμβος τοποθετείται σε απόσταση ακριβώς όσο το εύρος αίσθησης του προηγούμενου κινητού κόμβου, ενώ σε μια ομοιόμορφα κατανεμημένη τοπολογία είναι σπάνιο να συμβεί αυτό, άρα το μονοπάτι που χρησιμοποιεί ο αλγόριθμος Dynamic MobileCC αποτελείται από περισσότερα βήματα (hops).

Παρατηρούμε ότι ο μέσος χρόνος που χρειάζεται για να σταλθεί ένα μήνυμα από κάποιο πηγαίο κόμβο στον κόμβο προορισμού και για τους δυο αλγορίθμους είναι μικρός, με το χρόνο του αλγορίθμου Dynamic MobileCC να είναι λίγο μεγαλύτερος από το χρόνο του αλγορίθμου Direct Path MobileCC, λόγω της λειτουργίας του όπως εξηγείται πιο πάνω, και επειδή το μονοπάτι που δημιουργεί αποτελείται από λιγότερα hops..



Σχήμα 6.30 Γραφική που περιγράφει το ποσοστό επιτυχίας (success rate) για δίκτυα με 60 κόμβους

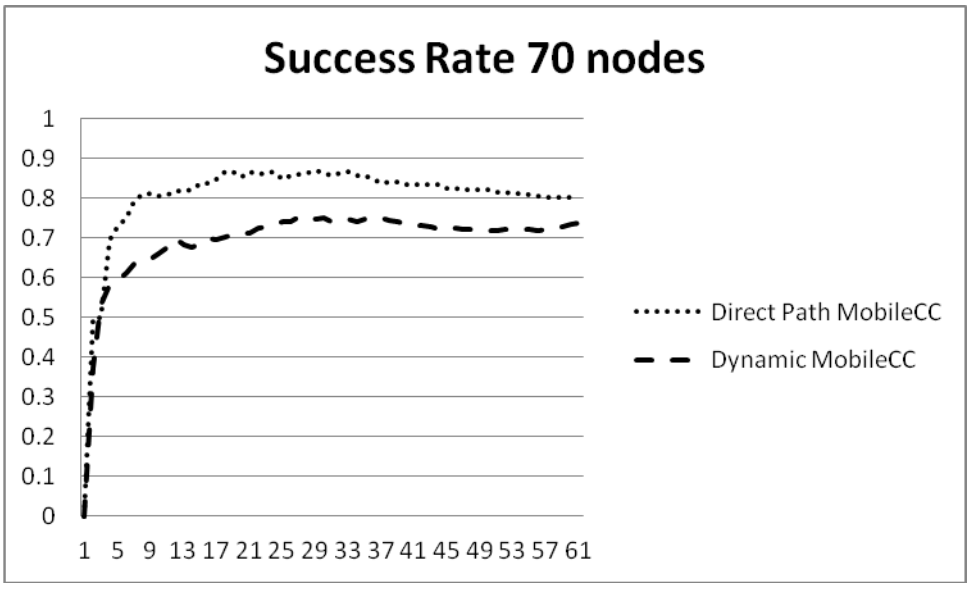


Σχήμα 6.31 Γραφική που περιγράφει τις καθυστερήσεις (delays) για δίκτυα με 60 κόμβους

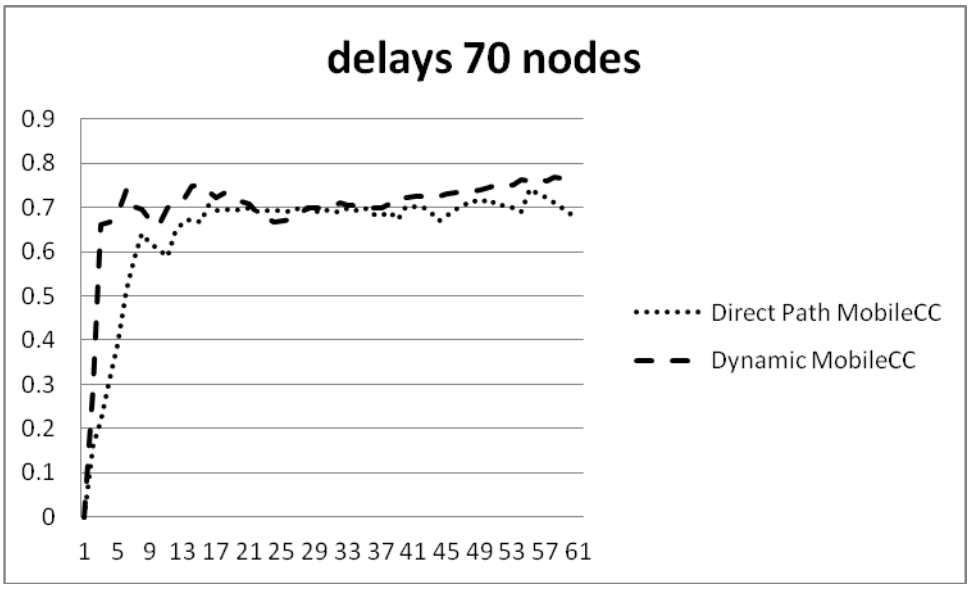
Παρατηρούμε ότι για τα δίκτυα με 60 κόμβους το success rate είναι πολύ ψηλό και για τους δυο αλγορίθμους, με το success rate του αλγορίθμου Direct Path MobileCC να είναι λίγο καλύτερο από του αλγορίθμου Dynamic MobileCC και είναι περίπου ίσο με το αντίστοιχο success rate για τα δίκτυα με 50 κόμβους. Αυτό συμβαίνει γιατί εφόσον ή τοπολογία έχει περισσότερους κόμβους, σημαίνει ότι το δίκτυο είναι πιο πυκνό και ότι περισσότεροι κόμβοι θα αισθανθούν το event και άρα θα γίνουν πηγαίοι κόμβοι. Αυτό έχει ως αποτέλεσμα περισσότερα θέματα συμφόρησης να δημιουργηθούν στο δίκτυο, αλλά επειδή υπάρχουν περισσότερες συνδέσεις στο δίκτυο και συνεπώς περισσότερα μονοπάτια από τους πηγαίους κόμβους στον κόμβο προορισμού, το success rate του δικτύου παραμένει το ίδιο.

Παρατηρούμε ότι ο μέσος χρόνος που χρειάζεται για να σταλθεί ένα μήνυμα από κάποιο πηγαίο κόμβο στον κόμβο προορισμού και για τους δυο αλγορίθμους είναι λίγο μεγαλύτερος από τον αντίστοιχο χρόνο για τα δίκτυα με 50 κόμβους, με το χρόνο του αλγορίθμου Dynamic MobileCC να είναι λίγο μεγαλύτερος από το χρόνο του αλγορίθμου Direct Path MobileCC, όπως συμβαίνει και στα δίκτυα με 50 κόμβους. Επειδή το δίκτυο έχει περισσότερους κόμβους συνήθως αυξάνεται ο αριθμός των βημάτων (hops) για να σταλθεί ένα

μήνυμα από ένα πηγαίο κόμβο στον κόμβο προορισμού, και εφόσον σε κάθε κόμβο χρειάζεται κάποιος χρόνος για να επεξεργαστεί το εισερχόμενο πακέτο, αυξάνεται επίσης και ο συνολικός χρόνος που χρειάζεται για να σταλθεί το πακέτο.

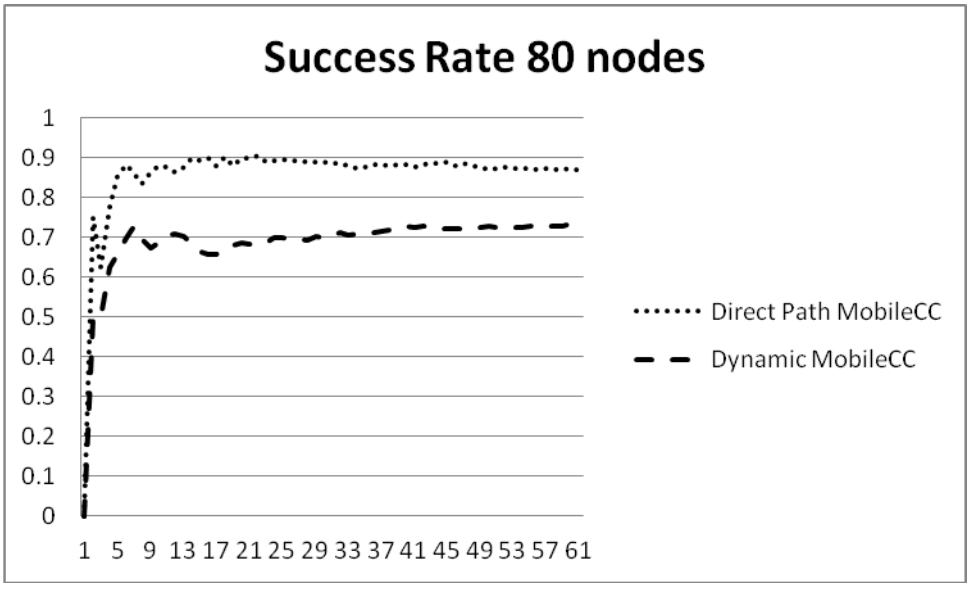


Σχήμα 6.32 Γραφική που περιγράφει το ποσοστό επιτυχίας (success rate) για δίκτυα με 70 κόμβους

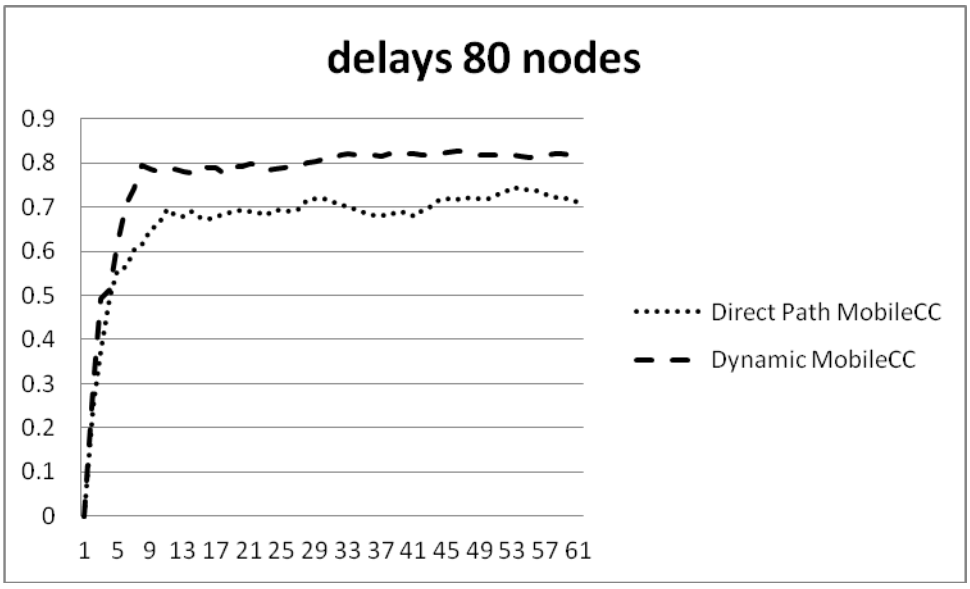


Σχήμα 6.33 Γραφική που περιγράφει τις καθυστερήσεις (delays) για δίκτυα με 70 κόμβους

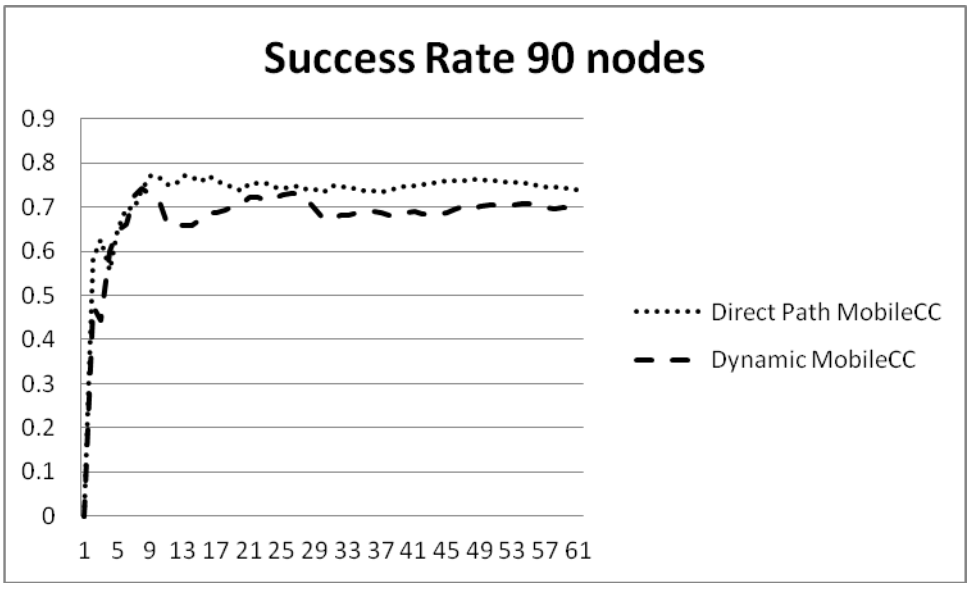
Σημειώνεται ότι ένα πακέτο έγινε drop στον αλγόριθμο Dynamic MobileCC.
Οι παρατηρήσεις είναι όμοιες με πριν.



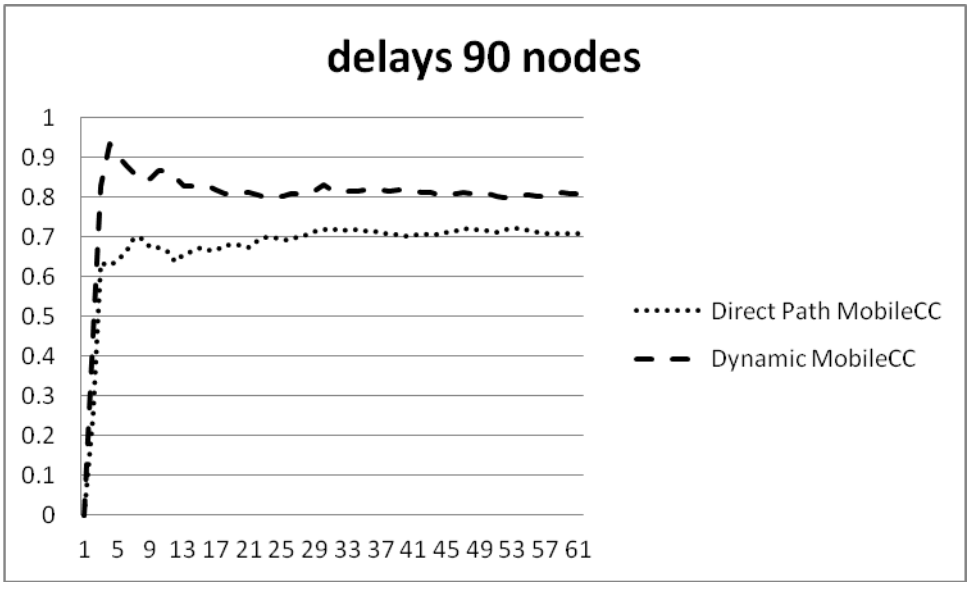
Σχήμα 6.34 Γραφική που περιγράφει το ποσοστό επιτυχίας (success rate) για δίκτυα με 80 κόμβους



Σχήμα 6.35 Γραφική που περιγράφει τις καθυστερήσεις (delays) για δίκτυα με 80 κόμβους
Οι παρατηρήσεις είναι όμοιες με πριν.

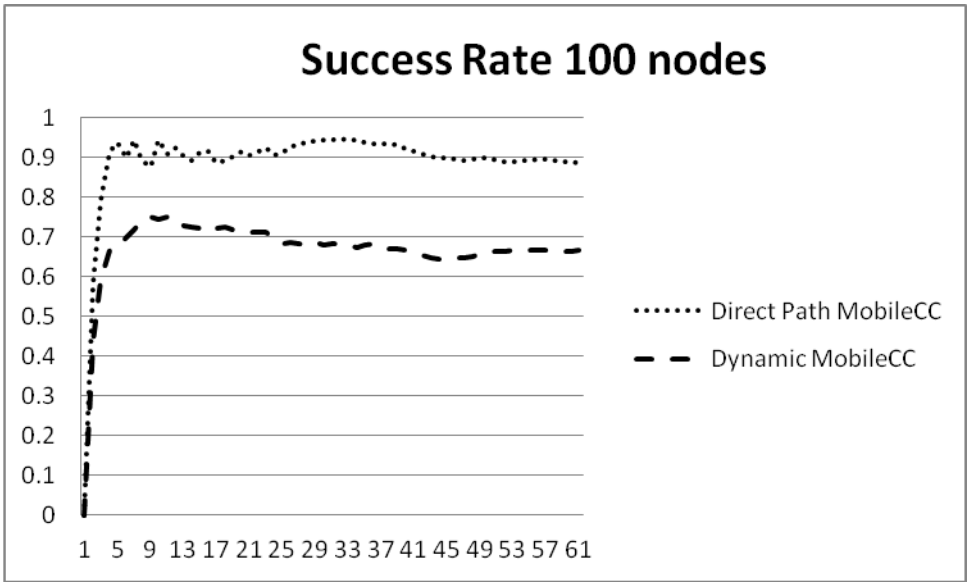


Σχήμα 6.36 Γραφική που περιγράφει το ποσοστό επιτυχίας (success rate) για δίκτυα με 90 κόμβους

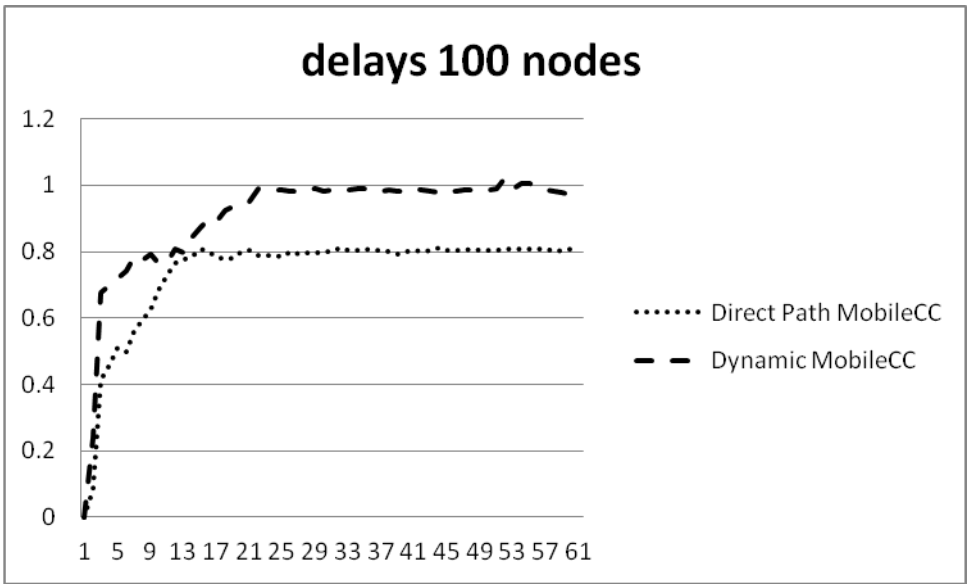


Σχήμα 6.37 Γραφική που περιγράφει τις καθυστερήσεις (delays) για δίκτυα με 90 κόμβους

Σημειώνεται ότι δυο πακέτα έγιναν drop στον αλγόριθμο Direct Path MobileCC, ενώ τρία πακέτα έγιναν drop στον αλγόριθμο Dynamic MobileCC.
Οι παρατηρήσεις είναι όμοιες με πριν.



Σχήμα 6.38 Γραφική που περιγράφει το ποσοστό επιτυχίας (success rate) για δίκτυα με 100 κόμβους



Σχήμα 6.39 Γραφική που περιγράφει τις καθυστερήσεις (delays) για δίκτυα με 100 κόμβους

Σημειώνεται ότι δυο πακέτα έγιναν drop στον αλγόριθμο Direct Path MobileCC ενώ τρία πακέτα έγιναν drop στον αλγόριθμο Dynamic MobileCC.
Οι παρατηρήσεις είναι όμοιες με πριν.

6.5 Συμπεράσματα

6.5.1 Στοχευμένη Αξιολόγηση

Παρατηρούμε ότι όταν υπάρχουν στο δίκτυο περισσότερα από ένα bottleneck, τα οποία οδηγούν σε συμφόρηση, σε διαφορετικά μέρη του δικτύου, και όχι διαδοχικά, τότε ο Global MobileCC, λόγω της καθολικής εξέτασης των θεμάτων συμφόρησης και της προϋπόθεσης ότι γνωρίζει όλα τα θέματα συμφόρησης που θα εμφανιστούν στο δίκτυο εκ των προτέρων, χρησιμοποιεί ένα λιγότερο κινητό κόμβο σε σχέση με τον αλγόριθμο Dynamic MobileCC, ο οποίος εξετάζει κάθε θέμα συμφόρησης τοπικά.

Όταν όμως υπάρχουν ένα ή περισσότερα bottlenecks, τα οποία οδηγούν σε συμφόρηση και το καθένα είτε είναι δίπλα από τον κόμβο προορισμού, είτε μόνο σε ένα κόμβο μπορεί να στείλει τα δεδομένα του, δηλαδή εμφανίζονται διαδοχικά θέματα συμφόρησης, τότε ο αριθμός των κινητών κόμβων που χρησιμοποιεί ο αλγόριθμος Global MobileCC είναι ίσος με τον αριθμό των κινητών κόμβων που χρησιμοποιεί ο αλγόριθμος Dynamic MobileCC.

Σε όλα τα παραδείγματα παρατηρούμε ότι ο αλγόριθμος Direct Path MobileCC χρειάζεται αρκετά μεγαλύτερο αριθμό κινητών κόμβων σε σχέση με τον αλγόριθμο Dynamic MobileCC, εκτός σε μια περίπτωση στην οποία τα θέματα συμφόρησης που εμφανίζονται είναι δίπλα από τον κόμβο προορισμού και έτσι δε χρειάζεται να δημιουργηθεί μονοπάτι μέχρι τον κόμβο προορισμού, και είναι αναμενόμενο από τη σχεδίαση του αλγορίθμου ότι ο αριθμός των κινητών κόμβων που θα χρησιμοποιήσει ο αλγόριθμος Direct Path MobileCC θα είναι ίσος με τον αριθμό των κινητών κόμβων που θα χρησιμοποιήσει ο αλγόριθμος Dynamic MobileCC.

6.5.2 Τυχαιοποιημένη Αξιολόγηση

Παρατηρούμε ότι ο συνολικός αριθμός των πακέτων που λαμβάνει ο κόμβος προορισμού έναντι του συνολικού αριθμού των πακέτων που αποστέλλονται από όλους τους πηγαίους κόμβους, δηλαδή το success rate, είναι πολύ ψηλό

και για τους δυο αλγορίθμους, με το success rate του αλγορίθμου Direct Path MobileCC να είναι λίγο καλύτερο από του αλγορίθμου Dynamic MobileCC, το οποίο είναι αναμενόμενο επειδή ο αλγόριθμος Direct Path MobileCC χρησιμοποιεί τουλάχιστον τους διπλάσιους κινητούς κόμβους σε σχέση με τον αλγόριθμο Dynamic MobileCC κι έτσι παρέχει στο δίκτυο επιπλέον πόρους τους οποίους εκμεταλλεύεται για τη δημιουργία εναλλακτικών μονοπατιών, και με αυτό τον τρόπο μπορεί να αντιμετωπίσει αποδοτικότερα τα θέματα συμφόρησης που εμφανίζονται μετέπειτα, ενώ ο αλγόριθμος Dynamic MobileCC χρησιμοποιεί μόνο ένα κινητό κόμβο για κάθε θέμα συμφόρησης που παρουσιάζεται.

Όσο αφορά τον αλγόριθμο Direct Path MobileCC παρατηρούμε στην αξιολόγηση του ότι το success rate σταθεροποιείται πιο σύντομα σε σχέση με τον αλγόριθμο Dynamic MobileCC. Αυτό συμβαίνει λόγω της λειτουργίας του αλγόριθμου Direct Path MobileCC, δηλαδή λόγω του ότι δημιουργεί ολόκληρο το μονοπάτι που αρχίζει από τους γείτονες του κόμβου που είναι σε συμφόρηση και τελειώνει στον κόμβο προορισμού, ενώ ο αλγόριθμος Dynamic MobileCC τοποθετεί μόνο ένα κινητό κόμβο για κάθε θέμα συμφόρησης που παρουσιάζεται στο δίκτυο και έτσι χρειάζεται περισσότερο χρόνο για να δημιουργήσει τα εναλλακτικά μονοπάτια από τους γείτονες των κόμβων που είναι σε συμφόρηση στον κόμβο προορισμού (γιατί τοποθετώντας ένα κινητό κόμβο για να επιλυθεί ένα πρόβλημα συμφόρησης τοπικά μπορεί να δημιουργήσει ένα άλλο πρόβλημα στο μονοπάτι που θα χρησιμοποιεί ο νέος κινητός κόμβος που θα τοποθετηθεί για να στέλνει τα δεδομένα που λαμβάνει στον κόμβο προορισμού ή μπορεί επίσης να μην υπάρχει κανένας κόμβος που να μην είναι σε συμφόρηση για στέλνει τα δεδομένα του ο κινητός κόμβος και σε αυτήν την περίπτωση θα πρέπει να τοποθετηθεί ένας επιπλέον κινητός κόμβος). Για αυτόν ακριβώς τον λόγο παρατηρούμε επίσης ότι ο μέσος χρόνος που χρειάζεται για να σταλθεί ένα μήνυμα από κάποιο πηγαίο κόμβο στον κόμβο προορισμού για τον αλγόριθμο Dynamic MobileCC είναι λίγο μεγαλύτερος από το χρόνο του αλγορίθμου Direct Path MobileCC (χρειάζεται περισσότερος χρόνος για να δημιουργηθεί ολόκληρο το μονοπάτι από τους γείτονες του κόμβου που είναι σε συμφόρηση μέχρι τον κόμβο προορισμού για

τον αλγόριθμο Dynamic MobileCC). Ακόμη ένας λόγος είναι το γεγονός ότι το μονοπάτι που δημιουργεί ο αλγόριθμος Direct Path MobileCC είναι το μονοπάτι με το μικρότερο αριθμό βημάτων (hops) από μερικούς από τους γείτονες του κόμβου που είναι σε συμφόρηση μέχρι τον κόμβο προορισμού (γιατί εκτός από τον πρώτο κινητό κόμβο κάθε επόμενος κόμβος τοποθετείται ακριβώς σε απόσταση όσο το εύρος αίσθησης του προηγούμενου κινητού κόμβου), ενώ ο κινητός κόμβος που τοποθετείται στον αλγόριθμο Dynamic MobileCC αφού χρησιμοποιεί τα ήδη υπάρχων μονοπάτια του δικτύου, είναι σπάνιο σε μια τυχαία κατανεμημένη τοπολογία να βρεθεί μονοπάτι στο οποίο κάθε κόμβος να είναι τοποθετημένος σε απόσταση όσο το εύρος αίσθησης του προηγούμενου κόμβου, άρα ο αριθμός των βημάτων (hops) θα είναι μεγαλύτερος.

Επιπρόσθετα όσο αυξάνεται ο αριθμός των κόμβων, δηλαδή αυξάνεται η πυκνότητα του δικτύου, συνήθως αυξάνεται επίσης και ο αριθμός των βημάτων (hops) για να σταλθεί ένα μήνυμα από ένα πηγαίο κόμβο στον κόμβο προορισμού, και εφόσον σε κάθε κόμβο χρειάζεται κάποιος χρόνος για να επεξεργαστεί το εισερχόμενο πακέτο, αυξάνεται επίσης και ο συνολικός χρόνος που χρειάζεται για να σταλθεί το πακέτο.

Παρατηρούμε ακόμη ότι ο αριθμός των πακέτων που γίνονται drop είναι πολύ μικρός, λόγω της παραδοχής ότι οι κινητοί κόμβοι κινούνται με πολύ μεγάλη ταχύτητα κι έτσι κάθε θέμα συμφόρησης αντιμετωπίζεται άμεσα και δεν λαμβάνουν χώρο οι αρνητικές συνέπειες της συμφόρησης.

Κεφάλαιο 7

Συμπεράσματα

7.1 Γενικά Συμπεράσματα

Έχει αποδειχτεί ότι η κινητικότητα στα ασύρματα δίκτυα αισθητήρων προσφέρει πολλά πλεονεκτήματα, όπως τη μεγιστοποίηση της κάλυψης της περιοχής του δικτύου, τον καλύτερο έλεγχο της τοπολογίας του δικτύου και την επιδιόρθωση τρυπών κάλυψης των δικτύων που μπορούν να δημιουργηθούν με το θάνατο συγκεκριμένων κόμβων κλπ. Σε αυτή τη διπλωματική εργασία χρησιμοποιείται η κινητικότητα για την επίλυση ενός από τα σημαντικότερα προβλήματα των ασύρματων δικτύων αισθητήρων που είναι η συμφόρηση. Έχουν αναπτυχθεί τρεις αλγόριθμοι ελέγχου συμφόρησης με χρήση κινητών κόμβων αισθητήρων, οι οποίοι εκτελούνται όταν αποτυγχάνει ο αλγόριθμος ελέγχου συμφόρησης που χρησιμοποιεί το δίκτυο, να αντιμετωπίσει τα θέματα συμφόρησης που παρουσιάζονται.

Το πιο σημαντικό συμπέρασμα που μπορούμε εύκολα να παρατηρήσουμε από την τυχαιοποιημένη αξιολόγηση που έγινε είναι ότι οι αλγόριθμοι Dynamic MobileCC και Direct Path MobileCC, καταφέρνουν να διατηρήσουν το success rate καθ' όλη τη διάρκεια της προσομοίωσης σταθερό σε ψηλό βαθμό. Η συνηθισμένη συμπεριφορά οποιουδήποτε αλγορίθμου ελέγχου συμφόρησης που δε χρησιμοποιεί κινητούς κόμβους αισθητήρες, όπως είναι και ο DalPaS, είναι η εξής: αρχικά το success rate είναι ψηλό αλλά μετά μειώνεται σταδιακά

λόγω της αποτυχίας του αλγορίθμου να δημιουργήσει εναλλακτικά μονοπάτια για την αντιμετώπιση της συμφόρησης.

Συγκρίνοντας τους δυο αλγορίθμους Dynamic MobileCC και Direct Path MobileCC παρατηρούμε ότι και οι δυο αλγόριθμοι καταφέρνουν να πετύχουν ψηλό success rate και χαμηλό delay, με τον αλγόριθμο Direct Path MobileCC να είναι ελάχιστα καλύτερος από τον αλγόριθμο Dynamic MobileCC (έχει περίπου 0.1 μεγαλύτερο success rate και περίπου 0.1 μικρότερο delay). Όμως ο αλγόριθμος Direct Path MobileCC για όλα τα σενάρια που έγιναν χρησιμοποιούσε κατά μέσο όρο τουλάχιστον το διπλάσιο αριθμό κινητών κόμβων σε σχέση με τον αλγόριθμο Dynamic MobileCC, οι οποίοι συνήθως κοστίζουν. Επιπλέον ο αριθμός των κινητών κόμβων που χρησιμοποιούν και οι δυο αλγόριθμοι αυξάνεται υπογραμμικά καθώς αυξάνεται το πλήθος των κόμβων του δικτύου.

Καταλήγουμε στο συμπέρασμα ότι ο αλγόριθμος Dynamic MobileCC επιτυχαίνει το σκοπό του με τη χρήση σχεδόν όσον κινητών κόμβων χρησιμοποιεί ο βέλτιστος αλγόριθμος Global MobileCC, ενώ ο αλγόριθμος Direct Path MobileCC κάνει σπατάλη κινητών κόμβων, οι οποίοι συνήθως κοστίζουν.

Με βάση τη στοχευμένη αξιολόγηση παρατηρούμε ότι όταν υπάρχουν στο δίκτυο περισσότερα από ένα bottleneck, τα οποία οδηγούν σε συμφόρηση, σε διαφορετικά μέρη του δικτύου, και όχι διαδοχικά, τότε ο αλγόριθμος Global MobileCC, λόγω της καθολικής εξέτασης των θεμάτων συμφόρησης και της προϋπόθεσης ότι γνωρίζει όλα τα θέματα συμφόρησης που θα εμφανιστούν στο δίκτυο εκ των προτέρων, χρησιμοποιεί ελαφρώς λιγότερους κινητούς κόμβους σε σχέση με τον αλγόριθμο Dynamic MobileCC, ο οποίος εξετάζει κάθε θέμα συμφόρησης τοπικά. Όταν όμως υπάρχουν ένα ή περισσότερα bottleneck, τα οποία οδηγούν σε συμφόρηση, και το καθένα είτε είναι δίπλα από τον κόμβο προορισμού, είτε μόνο σε ένα κόμβο μπορεί να στείλει τα δεδομένα του, δηλαδή εμφανίζονται διαδοχικά θέματα συμφόρησης, τότε ο αριθμός των κινητών κόμβων που χρησιμοποιεί ο αλγόριθμος Global MobileCC

είναι ίσος με τον αριθμό των κινητών κόμβων που χρησιμοποιεί ο αλγόριθμος Dynamic MobileCC.

7.2 Μελλοντική Εργασία

Η υλοποίηση που έγινε ίσως να είναι η βέλτιστη δυνατή αλλά υπάρχουν πολλά περιθώρια επέκτασης της. Πιο συγκεκριμένα θα μπορούσε ο αλγόριθμος να προβλέπει πότε θα υπάρξει ανάγκη για χρήση των κινητών κόμβων και να αρχίζει η αποστολή τους πριν την εμφάνιση της συμφόρησης. Θα μπορούσε επίσης να προστεθεί η επαναχρησιμοποίηση των κινητών κόμβων στον αλγόριθμο ώστε όταν αντιμετωπίζεται το θέμα συμφόρησης που εξυπηρετεί ο κινητός κόμβος να χρησιμοποιείται για άλλα θέματα συμφόρησης ή αν υπάρχει η δυνατότητα να αλλάζει η θέση του ώστε να εξυπηρετεί παράλληλα επιπλέον θέματα συμφόρησης. Μια άλλη επέκταση θα ήταν η συμβολή της καθυστέρησης της τοποθέτησης των κινητών κόμβων στη θέση που τους ανατέθηκε και σε αυτή την περίπτωση θα ήταν επίσης ενδιαφέρον να προστεθεί δυναμισμός στον αλγόριθμο ώστε μέχρι να τοποθετηθούν οι κινητοί κόμβοι στη θέση που αποφάσισε ο αλγόριθμος, να μπορεί να παρακολουθεί τις εξελίξεις στο δίκτυο και να αλλάζει δυναμικά η θέση του.

Επιπλέον η τυχαιοποιημένη αξιολόγηση του αλγορίθμου έγινε μόνο εξετάζοντας την απόδοση του στην ομοιόμορφα κατανεμημένη τοπολογία (uniform topology). Για αυτήν την τοπολογία εξετάστηκε ο αλγόριθμος μεταβάλλοντας τον αριθμό των κόμβων του δικτύου. Θα ήταν ενδιαφέρον να γίνει αξιολόγηση της συμπεριφοράς του αλγορίθμου σε διαφορετικές τοπολογίες. Παρομοίως μπορεί να επεκταθεί η στοχευμένη αξιολόγηση που έγινε ώστε να εξεταστεί η συμπεριφορά του αλγορίθμου Dynamic MobileCC και Direct Path MobileCC, σε σχέση με τον βέλτιστο αλγόριθμο Global MobileCC σε τοπολογίες με διαφορετικά χαρακτηριστικά.

Τέλος θα ήταν ενδιαφέρον να εξεταστεί η συμπεριφορά των αλγορίθμων σε πραγματικό δίκτυο ασύρματων αισθητήρων και να συγκριθούν τα αποτελέσματα, με τα αποτελέσματα της προσομοίωσης.

Βιβλιογραφία

[1] C. Sergiou, V. Vassiliou, “DAlPaS: A performance aware congestion control algorithm in Wireless Sensor Networks,” Telecommunications (ICT), 2011 18th International Conference on 8-11 May 2011, pp. 167-173, 2011.

[2] A. Pafitis, “Evaluation of resource control algorithms for congestion control in wireless sensor networks, Master’s thesis,” Department of Computer Science, University of Cyprus, 2007.

[3] Y. Zhang and G. Simon, “High-level sensor network simulations for routing performance evaluations,” Third International Conference on Networked Sensing Systems (INSS06), Chicago, IL, pp. 1-4, 2006

[4] Prowler: Probabilistic wireless network simulator. http://www.isis.vanderbilt.edu/Projects/nest/prowler/, 2010.

[5] C. Buratti, A. Conti, D. Dardari and R. Verdone, “An overview on wireless sensor networks technology and evolution,” Sensors, vol. 9, pp. 6869–6896, 2009.

[6] M.A. Labrador and P.M. Wightman, “Topology control in Wireless Sensor Networks,” Springer, 2009.

[7] R. Hashemzehi, R. Nourmandipour, F. koroupi, “Congestion in Wireless Sensor Networks and Mechanisms for Controling Congestion,” Indian Journal of Computer Science and Engineering (IJCSE), vol. 4, 2013.

[8] P. Santi, “Topology control in Wireless Ad hoc and Sensor Networks,” John Wiley & Sons, Ltd. , 2006.

[9] M. Koutroullos, C. Sergiou, V. Vassiliou, “Mobile-CC: Introducing mobility to WSNs for congestion mitigation in heavily congested areas,” Telecommunications (ICT), 2011 18th International Conference on 8-11 May 2011, pp. 400 – 405, 2011.

[10] M. Yongguo, X. Changjiu, D. Saumitra, Y. H. Charlie and L. Yung-Hsiang, “Repairing Sensor Network Using Mobile Robots,” Workshop on Wireless Ad hoc and Sensor Networks, 2006.

[11] R. Katsuma, Y. Murata, N. Shibata, K. Yasumoto, M. Ito, “Extending k-Coverage Lifetime of Wireless Sensor Networks Using Mobile Sensor Nodes,” Proceedings of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WIMOB 2009), pp. 48–54, 2009.

[12] R. Katsuma, Y. Murata, N. Shibata, K. Yasumoto, M. Ito, “Extending k-Coverage Lifetime of Wireless Sensor Networks with Surplus Nodes,” Mobile Computing and Ubiquitous Networking, pp. 9-16, 2010.

[13] J. Mena, M. Gerla, V. Kalogeraki, “Mitigate Funnel Effect in Sensor Networks with Multi-Interface Relay Nodes,” Proceedings of the 8th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), DCOSS '12, pp. 216 – 223, 2012.

[14] Chang WuYu, Chen, E., Chun Cheng Fang, “Deploying mobile nodes to connect wireless sensor networks using novel algorithms,” IEEE Proceedings of International Conference on Wireless Algorithms, Systems and Applications (WASA’07), pp. 199-204, 2007.

[15] M. Khan, W. Gansterer, G. Haring, “Congestion Avoidance and Energy Efficient Routing Protocol for Wireless Sensor Networks with a Mobile Sink”, Journal Of Networks, vol. 2, pp. 42-49, 2007.

[16] J. Sheu, K. Hsieh, P. Cheng, “Design and Implementation of Mobile Robot for Nodes Replacement in Wireless Sensor Networks,” Journal Of Information Science and Engineering 24, pp.393-410, 2005.

[17] H. Karl, A. Willig, “Protocols and Architectures For Wireless Sensor Networks,” Ed. John Wiley & Sons Inc., pages 60-63, 2005.

[18] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, “Habitat Monitoring: Application Driver for Wireless Communications Technology,” Proceedings of the ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean, San Jose, Costa Rica, pp. 20-41, 2001.

[19] C. Sergiou and V. Vassiliou, “Alternative path creation vs data rate reduction for congestion mitigation in wireless sensor networks,” in IPSN ’10: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (Poster Session). New York, NY, USA: ACM, pp. 394–395, 2010.

[20] C. Sergiou, “Performance-aware congestion control in wireless sensor networks using resource control,” Doctoral Dissertation, Department of Computer Science, University of Cyprus, pp. 2, 2012.

[21] Z. Shelby, C. Bormann, “6LoWPAN: The Wireless Embedded Internet,” Internet Engineering Task Force (6LoWPAN-WG), 2009.

[22] L. M. L. Oliveira, A. F. de Sousa and J. J. P. C. Rodrigues, “Routing and mobility approaches in IPv6 over LoWPAN mesh networks” International Journal of Communication Systems, vol. 24, pp. 1445-1466, 2011.

[23] B. White, A. Tsourdos, I. Ashokaraj, S. Subchan, R. Zbikowski, “Contaminant Cloud Boundary Monitoring Using Network of UAV Sensors”, IEEE Sensors Journal, vol. 8, pp. 1681-1692, 2008.

[24] A. Ghosha, S. K. Dasb, “Coverage and connectivity issues in wireless sensor networks: A survey”, Journal of Pervasive and Mobile Computing, vol. 4, pp 303-334, 2008.

[25] B. Liu, P. Brass, O. Dousse, P. Nain, D. Towsley, “Mobility Improves Coverage of Sensor Networks,” The ACM International Symposium on Mobile Ad Hoc Networking and Computing ”MobiHoc’05”, Urbana-Champaign, Illinois, USA, May 25-27, 2005.

[26] R. Nuno Mendo da Silva, “Mobility support in Low Power Wireless Personal Area Networks (MLoWPAN),” University of Coimbra, PORTUGAL, 2010.

[27] Knuth D.E., The Art of Computer Programming: Generating All Combinations and Permutations, vol. 4, fasc. 3, Addison-Wesley, 2005.

[28] H. Takagi and L. Kleinrock, “Optimal transmission ranges for randomly distributed packet radio terminals,” Communications, IEEE Transactions, vol. 32, pp. 246-257, 1984.

[29] T. C. Hou and V. 0. K. Li, “Transmission range control in multihop packet radio networks,’’ Communications, IEEE Transactions, vol. 34, pp. 38-44, 1986.

[30] Ajay K. Sharma and Deepti Gupta. Article: Performance evaluation of routing protocols for wsns based on energy-aware routing with different radio models. International Journal of Computer Applications, 3(12): 6–14, July 2010. Published By Foundation of Computer Science.

[31] Gyula Simon, Peter Volgyesi, Miklos Maroti, and Akos Ledeczi. Simulation-based optimization of communication protocols for large-scale

wireless sensor networks. In Proceedings of IEEE Aerospace Conference, volume 3, pages 1339–1346, Big Sky, MT, USA, 2003.

[32] Ittipong Khemapech, Ishbel Duncan, and Alan Miller. A Survey of Wireless Sensor Networks Technology. In M. Merabti and R. Pereira, editors, 6th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, Liverpool, UK, June 2005.

[33] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys ’03, pages 126–137, New York, NY, USA, 2003.

[34] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS’98), pages 154–161, May 1998.

[35] Boonma, P., Suzuki, J.: Monsoon: A coevolutionary multiobjective adaptation framework for dynamic wireless sensor networks. In: Proc. of IEEE Hawaii Int’l Conf on System Sciences, 2008.

[36] J.Kleinberg & E. Tardos – Algorithm Design, Pearson Education, New Delhi, 2006.

Παράρτημα Α

MATLAB source code – Παρουσιάζεται ο πηγαίος κώδικας που αναπτύχθηκε στην παρούσα διπλωματική εργασία. Ο πηγαίος κώδικας σχετικά με τον αλγόριθμο DalPaS ο οποίος χρησιμοποιείται μπορεί να βρεθεί στο [2].

```
function status = MobileCC_layer(N, S)

% Written by Antonia Nicolaou
% Last modified: April 6 2015 by Antonia Nicolaou
% Implementation of the Algorithms: Dynamic MobileCC,
Direct Path MobileCC
% and Global MobileCC

% DO NOT edit simulator code (lines that begin with S;)

S; %%%%%%%%%%%%%%%%%%   housekeeping
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
S;      persistent app_data
S;      global ID t
S;      [t, event, ID, data]=get_event(S);
S;      [topology, mote_IDs]=prowler('GetTopologyInfo');
S;      ix=find(mote_IDs==ID);
S;      if ~strcmp(event, 'Init_Application')
    S;          try memory=app_data{ix}; catch memory=[];
end,
    S;      end
S;      global ATTRIBUTES
S;      status = 1;
S;      pass = 1;
S;
global DESTINATIONS SOURCES NEIGHBORS RNUMBERS LTOTALS
RTOTAL STOTAL ROUTINGTABLE type_of_MobileCC  RNEIGH
global MOBILE_NODES NEIGH_NODES_USES_MOBILE_NODES
buffer_capacity start_send_data  MOBILE_ADD_RESOURCE
MOBILE_POSSIBLE_COVER_SETS
global USED_MOBILE_NODES DEST_MOBILE_NODES STOTAL_generated
MOBILE_NODE_RATE NODE_USES_MOBILE_NODES MOBILE_SERVE_TIME
global Threshold AckStrength Perf_Thr_Delay buff_threshold
initial_pwr pwr_threshold xsink ysink IDsink
MOBILE_REACH_POSITION
global firts_time IDS_OF_MOB_NODES counter_extra_nodes
called_mob_located set_parents_of_mob_nodes
%type_of_MobileCC=1 means the local algorithm will execute,
%type_of_MobileCC=2 means the direct path algorithm will
execute,
%type_of_MobileCC=3 means the global algorithm will
execute.
```

```
%NEIGHBORS: NEIGHBORS{ID} holds the list of nodes that node
ID can hear
%RNUMBERS:  RNUMBERS{ID} holds the total received packets
from each of its neighbors
%LTOTALS:   LTOTALS{ID} holds the total number of lost
packets to each of its neighbors
%persistent DATA_M   = 10;
ACK      = -11;
INIT_H   = -33;
CONNECT  = -22;
BACK     = -66;
MOBILE   = -44;
LOCATE_MOBILE = -88;
LOCATE_ALL_MOBILE=-99;


switch event

    case 'Init_Application'
        firts_time=0;
        called_mob_located=0;

        columns=length(ATTRIBUTES);

        buffer_capacity = sim_params('get_app',
'BufferSize');
        if (isempty(buffer_capacity)) buffer_capacity = 5;
end
        set_parents_of_mob_nodes=zeros(1, columns);
        MOBILE_NODES=zeros(1, columns);
        USED_MOBILE_NODES=zeros(1, columns);
        MOBILE_NODE_RATE=zeros(1, columns);
        MOBILE_SERVE_TIME=zeros(1, columns);
        MOBILE_REACH_POSITION=zeros(1, columns);
        for i=1:length(ATTRIBUTES)
            NODE_USES_MOBILE_NODES{i}=[];
            MOBILE_ADD_RESOURCE{i}=[];
            MOBILE_POSSIBLE_COVER_SETS{i}=[];
            DEST_MOBILE_NODES{i}=[];
            NEIGH_NODES_USES_MOBILE_NODES{i}=[];

        end
        if (ix==1)
            %initialize only once
            buff_threshold = sim_params('get_app',
'BufferThreshold');
            if (isempty(Threshold)) Threshold = 0.1; end
            pwr_threshold = sim_params('get_app',
'PowerThreshold');
            if (isempty(Threshold)) Threshold = 0.5; end
```

```
            %get init power (initial power reserve on each
node from transmission_set_params)
            initial_pwr = sim_params('get_app',
'InitPower');
            if (isempty(initial_pwr)) initial_pwr = 10; end


            %    Perf_Thr_Delay = sim_params('get_app',
'Perf_Thr_Delay');
            %    if (isempty(Perf_Thr_Delay))
Perf_Thr_Delay = 20; end
        end
        sim_params('set_app', 'Promiscuous', 0);

        %counter that gives a unique id to each of the
extra nodes that are
        %needed in order to be
        %created the paths from the mobile nodes to sink
for the dynamic
        %algorithm that creates optimal paths ONLY GLOBAL
ALGORITHM USED IT
        counter_extra_nodes=length(ATTRIBUTES)+1;

        ATTRIBUTES{ID}.desireability = 1;
        ATTRIBUTES{ID}.needs_mobile = 0;
        ATTRIBUTES{ID}.serve = 0;
        ATTRIBUTES{ID}.serve_neigh=0;


    case 'Send_Packet'
        try msgID = data.msgID; catch msgID=0; end

    case 'Packet_Received'
        rdata = data.data;
        try msgID = rdata.msgID; catch msgID=0; end



        if msgID < 0
            %do staff and then pass = 1
            if msgID == MOBILE
                called_mob_located=t;
if(DESTINATIONS(ID)==1)
                    disp(sprintf('mobile3: id: %d
DESTINATIONS STOTALS %d RTOTALS %d
t',ID,STOTAL{ID},RTOTAL{ID},t/40000));
                    xsink=ATTRIBUTES{ID}.x;
                    ysink=ATTRIBUTES{ID}.y;
                    IDsink=ID;
```

```
                    has_available_mobile_node=0; % gets the
value 1, if there is an available mobile node
                    for i=1:length(MOBILE_NODES)
                        if(MOBILE_NODES(i)==1 &&
USED_MOBILE_NODES(i)==0)
                            has_available_mobile_node=1;
                        end
                    end
                    if has_available_mobile_node==1
                        if type_of_MobileCC==1 %local
algorithm

                            [p,
mobID]=dynamic_algorithm(0,[],[]);
                            %locate node at p


                        else if type_of_MobileCC==2 %direct
path algorithm
                                direct_path_algorithm();

                            else if type_of_MobileCC==3 &&
firts_time==0 %global algorithm
                                    firts_time=1;
                                    global_algorithm();
                                end
                            end
                        end
                    end
                end
            end
        end


S; %%%%%%%%%%%%%%%%%%%%%% housekeeping
%%%%%%%%%%%%%%%%%%%%%%%%%%
S;        try app_data{ix}=memory; catch app_data{ix} = [];
end
S;        if (pass) status = common_layer(N, make_event(t,
event, ID, data)); end
S;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function b=Set_Flood_Clock(alarm_time);
global ID
data.type = 'spantree_flood';
prowler('InsertEvents2Q', make_event(alarm_time,
'Clock_Tick', ID, data));
```

```
function [additional_resources,
nodes_needs_mobile]=find_nodes_needs_mob()
global ID NEIGHBORS MOBILE_SERVE_TIME
NODE_USES_MOBILE_NODES USED_MOBILE_NODES ATTRIBUTES
MOBILE_NODES DESTINATIONS ROUTINGTABLE RTOTAL STOTAL
RNUMBERS t xsink ysink IDsink RNEIGH start_send_data
[x, y]=size(ATTRIBUTES);
additional_resources=[];
nodes_needs_mobile=[];
pos=1;

for i=1:y
    if i~=IDsink && (MOBILE_NODES(i)==0 || (
MOBILE_NODES(i)==1 && USED_MOBILE_NODES(i)==1))
        i
        ATTRIBUTES{i}.needs_mobile
        ATTRIBUTES{i}.serve

        if (ATTRIBUTES{i}.needs_mobile == 1 &&
ATTRIBUTES{i}.serve==0)
            if MOBILE_NODES(i)==1
                %the congested node is a mobile node
                additional_resources(pos)=(sum(RNEIGH{i})-
STOTAL{i})/((t-
MOBILE_SERVE_TIME(NODE_USES_MOBILE_NODES{i}(1)))/40000);
            else
                %the congested node is not a mobile node
                additional_resources(pos)=(sum(RNEIGH{i})-
STOTAL{i})/((t-start_send_data)/40000);
            end
            if(additional_resources(pos))<=0
                additional_resources(pos)=0.001;
            end
            nodes_needs_mobile(pos)=i;
            pos=pos+1;
        end
    end
end

function [p, mobID]=dynamic_algorithm(type,
nodes_needs_mobile, additional_resources)

global ID NEIGHBORS ATTRIBUTES DESTINATIONS ROUTINGTABLE
RTOTAL STOTAL RNUMBERS t xsink ysink IDsink RNEIGH
sink=[xsink ysink];

findp=0;
pos=1;
p=[];
mobID=[];
```

```
if type<2
    %find which nodes need mobile nodes
    nodes_needs_mobile=[];
    additional_resources=0;
    [additional_resources,
nodes_needs_mobile]=find_nodes_needs_mob();
end
nodes_needs_mobile
additional_resources

%check if nodes that are neighbors of the node that is
%congested have sending rate>=additional_resources
for j=1:length(nodes_needs_mobile) %for every congested
node
    just_served_flag =
check_if_just_served(nodes_needs_mobile(j));
    just_served_flag

    if just_served_flag==0
        findp=0;

        N=NEIGHBORS{nodes_needs_mobile(j)};

        if findp==0
            [findp, p,
mobID]=algorithm_check_point_single_node(nodes_needs_mobile
(j), N, additional_resources(j));

        end

        %not found a neighbor that has sending rate >
additional_resources(j)


        if(findp==0)

[p,neigh_serve,new_cover_nodes]=connectivity_algorithm(2,
6, nodes_needs_mobile(j), additional_resources(j),[]);
            if(~ isempty(p))
                p

ATTRIBUTES{nodes_needs_mobile(j)}.needs_mobile = 0;
                ATTRIBUTES{nodes_needs_mobile(j)}.serve=1;
                locate_out{1}(1)=nodes_needs_mobile(j);
                locate_out{2}=p;
                locate_out{3}=neigh_serve;

mobID=locate_mobile_node(locate_out,additional_resources(j)
,nodes_needs_mobile(j),new_cover_nodes);
                %locate mobile node at p
```

```
            end

        end
        mobID
        if type==1 && ~isempty(mobID) %if is
direct_algorithm
            direct_sequence(p,mobID);
        end
    end
end



function [findp, p,
mobID,cover_set]=algorithm_check_point_single_node(congeste
d_node, N, additional_resources)
global ID NEIGHBORS ATTRIBUTES DESTINATIONS ROUTINGTABLE
RNUMBERS xsink ysink IDsink RNEIGH t SOURCES
findp=0;
cover_set=[];
sink=[xsink ysink];
pos=1;
posc=1;
min_dist_p_sink=inf;
p=[];
mobID=[];
this_point=[];
for i=1:length(N)

    if(ATTRIBUTES{N(i)}.parent==congested_node &&
ATTRIBUTES{N(i)}.hops>=ATTRIBUTES{congested_node}.hops &&
RNEIGH{congested_node}(N(i))>0 &&
((RNEIGH{congested_node}(N(i))/(t/40000))>additional_resour
ces))

        has_children=0; %check if the neighbor node has at
least one child
        if SOURCES(N(i))==0
            for k=1:length(NEIGHBORS{N(i)})
                if
ATTRIBUTES{NEIGHBORS{N(i)}(k)}.parent==N(i)
                    has_children=1;
                    break;
                end
            end
        else
            has_children=1;
        end
        if has_children==1
            cover_set{posc}=N(i);
```

```
            posc=posc+1;


this_point=create_alt_path_single_node(ATTRIBUTES{N(i)}.x,
ATTRIBUTES{N(i)}.y, ATTRIBUTES{IDsink}.x,
ATTRIBUTES{IDsink}.y);
            if(~isempty(this_point))

                dist_p_sink=pdist2(sink, this_point,
'euclidean');
                if dist_p_sink < min_dist_p_sink
                    min_dist_p_sink=dist_p_sink;
                    neigh=N(i);
                    p=this_point;
                end

                %choose point p, if this point is closer to
the sink then
                %it is the point p
            end
        end
    end
end




if(~ isempty(p))
    findp=1;

    xout=p(1);
    yout=p(2);
    %locate mobile node at p
    findp=1;
    ATTRIBUTES{congested_node}.needs_mobile = 0;
    ATTRIBUTES{congested_node}.serve=1;
    locate_out{1}(1)=congested_node;
    locate_out{2}=p;
    locate_out{3}=neigh;

mobID=locate_mobile_node(locate_out,additional_resources,co
ngested_node,cover_set);

end
%return the nodes that send their data to the congested
node sorted
function [set]=find_neigh_senders(congested_node)
```

```
global ID NEIGHBORS ATTRIBUTES DESTINATIONS ROUTINGTABLE
RNUMBERS xsink ysink RNEIGH
set=[];
pos=1;

for k=1:length(ATTRIBUTES)
    if RNEIGH{congested_node}(k)>0
        set(pos)= k;
        pos=pos+1;

    end
end

%return the nodes that send their data to the congested
node sorted
%begining from the node that sends more packets
function
[sorted_neighbors_senders]=find_neigh_senders_sorted(conges
ted_node)
global ID NEIGHBORS ATTRIBUTES DESTINATIONS ROUTINGTABLE
RNUMBERS xsink ysink RNEIGH
set=[];
pos=1;
for k=1:length(ATTRIBUTES)
    if RNEIGH{congested_node}(k)>0
        set(pos)= RNEIGH{congested_node}(k);
        pos=pos+1;
    end
end
set=sort(set,'descend');
[a,b] = histc(set,unique(set));
%counter has the number of times that a number exist in the
array set
counter = a(b);
current_counter=ones(1, length(set));
for i=1:length(set)
    d=set(i);
    pos=find(RNEIGH{congested_node}== d);
    if counter(i)>1 %more than one number has this number
of rneigh
        current_counter(i)
        pos=pos(current_counter(i));
        pos
        for k=1:length(set)
            if d==set(k)
                current_counter(k)=current_counter(k)+1;
            end
        end

    else
```

```
        pos=pos(1);
    end

    %disp(sprintf('find pos %d find for distance
%f',pos,distances_congested_nodes_from_sink(i)));
    sorted_neighbors_senders(i)=pos;
end
sorted_neighbors_senders

function
[p,neigh_serve,new_cover_sets]=connectivity_algorithm(min_l
ength, max_length, congested_node, additional_resources_j,
cover_node_set)
global ID NEIGHBORS ATTRIBUTES DESTINATIONS ROUTINGTABLE
RNUMBERS xsink ysink RNEIGH type_of_MobileCC t

new_cover_sets=[];
j=min_length;
cover_node_set_1=[];
%the coordinates of the sink
sink=[xsink ysink];
pos=1;
possub=1;
set=[];
p=[];
subsets=[];
%cover_set=[1 5 3];
%cover_node_set{2}=[2 3];
cover_node_set_1=cover_node_set;
%disp('cover nodes - connectivity_algorithm');
if ~isempty(cover_node_set)
    for i=1:length(cover_node_set)
        cover_node_set{i}
    end

end
second_priority=[];
neigh_serve=[];

if additional_resources_j==0 ||
congested_node>length(ATTRIBUTES)
    set=congested_node;
else
    %find the nodes that send their data through the
congested_node
    set=find_neigh_senders(congested_node);
end
```

```
disp('set');


n=length(set);
%disp(sprintf('length of neighbors %d',n));
max=min(n,max_length);
min_dist_p_sink=inf;

pos_Sec=1;
if(max>=1)
    for j=min_length:max
        %disp(sprintf('%min_length %d',min_length));
        if j==1
            %additional_resources_j
            for z=1:length(set)
                %disp(sprintf('see if > add res for set(z)
%d RNEIGH{congested_node}(set(z))/(t/40000)
%f',set(z),RNEIGH{congested_node}(set(z))/(t/40000)));
                if
RNEIGH{congested_node}(set(z))/(t/40000)>=additional_resour
ces_j
                    subsets{possub}(1)=set(z);

                    possub=possub+1;
                end
            end
        else
            %create all possible subs of the neighbors of
the congested_node that
            %send data to the congested_node, subsets has
POSITIONS, no elements
            subsets=create_subs(j, max,
additional_resources_j, congested_node, set);
        end
        for f=1:length(subsets)
            subsets{f}
        end


        %find point of intersection of any subset
        s=size(subsets);


        %used for global algorithm
        type_of_MobileCC
        if type_of_MobileCC>=3
            posk=1;

            for d=1:length(new_cover_sets)
                new_cover_sets{d}=[];
```

```
            end
            if ~isempty(cover_node_set)
                for m=1:length(cover_node_set)
                    for k=1:length(subsets)
                        if ~isempty(subsets{k})


new_cover_sets{posk}=union(subsets{k},cover_node_set{m});
                            posk=posk+1;
                        end
                    end
                end
            else
                new_cover_sets=subsets;

            end

            for u=1:length(new_cover_sets)
                new_cover_sets{u}
            end
        end
        if ~isempty(cover_node_set_1)

            size_cover_node=length(cover_node_set_1);
        else
            size_cover_node=1;
        end


        for i=1:s(2)
            for g=1:size_cover_node

                %second_priority are the points that
service all nodes if rate_check_flag==0, or sum_rate >=
limit rate if rate_check_flag==1 but
                %there is not a neighbor to connect with
                if ~isempty(subsets{i})

                    if ~isempty(cover_node_set)
                        C = union(
subsets{i},cover_node_set_1{g});

                    else
                        C=subsets{i};

                    end
                    [point, second_priority{pos_Sec},
second_priority_rate{pos_Sec} ]  = find_intersection_point(
C,0,[],[] );
                    if ~(isempty(point))
```

```
                        dist_p_sink=pdist2(sink, point,
'euclidean');
                        if dist_p_sink < min_dist_p_sink
                            min_dist_p_sink=dist_p_sink;
                            neigh_serve=C;
                            p=point;
                        end

                    end

                end
            end

        end


        if ~(isempty(p))

            return;
        end
    end %for every k [min_length:max]


    if ~(isempty(p))
        return;
    end
    min_distance=inf;
    for j=1:length(second_priority)
        for m=1:length(second_priority{j})
            %second_priority{j}{m}(1)
            %second_priority{j}{m}(2)

            dis=pdist2(sink, second_priority{j}{m},
'euclidean');

            if(min_distance > dis )
                min_distance = dis;
                p=second_priority{j}{m};
            end


        end
    end
    %Locate mobile node at p

end %else there is only one neighbror that send its data
through the congested_node


%{
```

```
find the point of intersection of all the nodes in a subset
- set input
Input: the set of the nodes
Input: rate_check_flag, if flag = 1, this means that when
locate a mobile node at the point of intersection, it has
to cover at least a specific rate from the setcover
Input:set_cover, the set of nodes that the point of
intersection has to cover a specific rate
Output: the point of intersection of the nodes that is
closer to the sink, if exist one

if the type is 2 the algorithm should find the point of
intersection that
includes the more rate it can
%}
function [I,second_priority, second_priority_rate]  =
find_intersection_point(set,rate_check_flag,set_cover,
limit_rate)
global ID NEIGHBORS ATTRIBUTES DESTINATIONS ROUTINGTABLE
RTOTAL STOTAL RNUMBERS t xsink ysink DEST_MOBILE_NODES
type_of_MobileCC
I=[];
r=2.2;
min_distance = inf; %minimum distance of any point of
intersection to the sink
%for every pair of nodes in the set

%second_priority are the points that includes service all
nodes but
%there is not a neighbor to connect with
second_priority=[];
second_priority_rate=[];
pos_second_pr=1;
pos_all_points=1;
all_points=[];
point_output=[];

%the coordinates of the sink
sink=[xsink,ysink];

for i=1:length(set)
    for j=i+1:length(set)


        %find the points of intersection p, q of the
circles with centers vi and vj and radius r (using the
function: circcirc)
        %disp(sprintf('circicirc 1st node id %d STOTAL %f x
%f y %f, 2nd node id %d STOTAL %f x %f y
%f',set(i),STOTAL{set(i)},ATTRIBUTES{set(i)}.x,ATTRIBUTES{s
```

```
et(i)}.y,set(j),STOTAL{set(j)},ATTRIBUTES{set(j)}.x,ATTRIBU
TES{set(j)}.y));

        [xout,yout] =
circcirc(ATTRIBUTES{set(i)}.x,ATTRIBUTES{set(i)}.y,r,ATTRIB
UTES{set(j)}.x,ATTRIBUTES{set(j)}.y,r);


        if ~(isnan(xout(1))) && ~(isnan(yout(1)))
            %initialize the flags
            qflag=1;  %true if not found yet any node that
not covers the point q
            pflag=1;  %true if not found yet any node that
not covers the point p

            q_set_cover=1; %true if not found yet any node
in the SET_COVER that not covers the point q
            p_set_cover=1; %true if not found yet any node
in the SET_COVER that not covers the point P

            qneigh=0;  %true if exist a neighbor node to
trasmits it's data if the mobile node locate at point q
            pneigh=0;  %true if exist a neighbor node to
trasmits it's data if the mobile node locate at point p

            q=[ xout(2), yout(2)];

            p=[ xout(1), yout(1)];

            if ~isempty(xout(1)) && ~isempty(xout(2))
                point_served=[ATTRIBUTES{set(i)}.x,
ATTRIBUTES{set(i)}.y];
                [can_locate_p, can_locate_q,NEAR_POINTS_P,
NEAR_POINTS_Q
,newp,newq]=check_empty_place(p,q,point_served);

                sum_sending_rate_p = STOTAL{set(i)} +
STOTAL{set(j)};
                sum_sending_rate_q = STOTAL{set(i)} +
STOTAL{set(j)};

                sum_rate_q =0; %sum rates of nodes that are
included in set_cover
                sum_rate_p =0; %sum rates of nodes that are
included in set_cover
                if( any( set_cover(:)== set(i) ) )
                    sum_rate_q=sum_rate_q+STOTAL{set(i)};
                    sum_rate_p=sum_rate_p+STOTAL{set(i)};
                end
```

```
                if( any( set_cover(:)== set(j) ) )
                    sum_rate_q=sum_rate_q+STOTAL{set(j)};
                    sum_rate_p=sum_rate_p+STOTAL{set(j)};
                end


                for z=1:length(set)
                    %check if p or/and q are in the range
of all other circles if the rate_check_flag is 0, or if the
total rate of the nodes that can send their data through
the mobile node at the point p is greater than the limit if
rate_check_flag=1

                    if(set(z)~=set(i) && set(z)~=set(j))
%set(z) is not the same node as the nodes as set(i) or
set(j)
                        if(can_locate_q==1 && (
(rate_check_flag==0 && qflag==1) ||   (rate_check_flag==1
&& sum_rate_q>=limit_rate) ) )
                            %if not found yet any node that
not covers the point q if the rate_check_flag is 0 (must
cover all the nodes in the subset), or if the total rate of
the nodes that can send their data through the mobile node
at the point p is greater than the limit

                            X = [ATTRIBUTES{set(z)}.x,
ATTRIBUTES{set(z)}.y; xout(2), yout(2)];
                            if ( pdist(X,'euclidean')< r )
                                sum_sending_rate_q =
sum_sending_rate_q + STOTAL{set(z)};


%if set(z) is included to the set_cover
                                if( any( set_cover(:)==
set(z) ) )

                                    sum_rate_q = sum_rate_q
+ STOTAL{set(z)};

end
                            else
                                qflag=0;
                                if( any( set_cover(:)==
set(z) ) )
                                    % disp(sprintf('
included set cover and not covered node %d point q
',set(z)));
                                    q_set_cover = 0;
                                end
                            end
```

```
                        end

                        if(can_locate_p==1 && (
(rate_check_flag==0 && pflag==1) ||   (rate_check_flag==1
&& sum_rate_p>=limit_rate) ) )
                                                          X =
[ATTRIBUTES{set(z)}.x, ATTRIBUTES{set(z)}.y; xout(1),
yout(1)];
                            if ( pdist(X, 'euclidean') <r)
                                sum_sending_rate_p =
sum_sending_rate_p + STOTAL{set(z)};

                                %if set(z) is included to
the set_cover
                                if( any( set_cover(:)==
set(z) ) )

                                    sum_rate_p = sum_rate_p
+ STOTAL{set(z)};
                                    % disp(sprintf('
included set cover sum_rate_p: %f',sum_rate_p));
                                end
                            else
                                pflag=0;
                                if( any( set_cover(:)==
set(z) ) )

p_set_cover=0;
                                end
                            end
                        end
                    end
                end

                if type_of_MobileCC<2 %if is the dynamic-
local algorithm check if their is a neighbor for the mobile
node to send its packet

                    %disp(sprintf(' see qflag %d pflag %d
sum_rate_q %f sum_rate_p %f r
%f',qflag,pflag,sum_rate_q,sum_rate_p,r));
                    if( ( (rate_check_flag==0) & (qflag==1
|  pflag==1)) | ( (sum_rate_q>=limit_rate |
sum_rate_p>=limit_rate) & (rate_check_flag==1) ) )
                        l=1;
                        %Y=the coordinates of the node l

                        while( (l <length(ATTRIBUTES)) &
(pneigh==0 || qneigh==0) )
```

```
                                Y=[ATTRIBUTES{l}.x,
ATTRIBUTES{l}.y];
                                if( isempty(find(set == l)) &&
ATTRIBUTES{l}.hard_stage==0 )  %the node is not congested

                                    %disp(sprintf(' apostasi p
: x %f y %f x %f y %f  =  %f', xout(1), yout(1),
ATTRIBUTES{l}.x, ATTRIBUTES{l}.y,pdist2(p,Y,'euclidean')));

                                    if(
le(pdist2(p,Y,'euclidean'),r) )  %check if there is
neighbor to transmits the mobile node at point q its data
                                        pneigh=1;

                                    end


                                    if(
le(pdist2(q,Y,'euclidean'),r) )  %check if there is
neighbor to transmits the mobile node at point q its data
                                        qneigh=1;
                                    end

                                end
                                l=l+1;
                            end
                        end
                    else
                        qneigh=1;
                        pneigh=1;
                    end

                    %there is not a neighbor to connect with


                    if( (can_locate_q & qflag==1 & qneigh==1 &
rate_check_flag==0) | (sum_rate_q>=limit_rate &
rate_check_flag==1 & qneigh==1))  %all elements of the
subset intersect at p
                        %all elements of the subset intersect
at q and there is a neighbor to transmit its data
                        if(pdist2(sink,q,'euclidean')  <
min_distance )
                            all_points{pos_all_points} = q;
%point I
                            pos_all_points=pos_all_points+1;
                            min_distance =
pdist2(sink,q,'euclidean');
                        end
```

```
                else if( (can_locate_q & qflag==1 &
qneigh==0 & rate_check_flag==0) | (sum_rate_q>=limit_rate &
rate_check_flag==1 & qneigh==0)) %if there is not neighbor
to transmits its data the mobile node keep the point p in
table second_priority
                        %if there is not neighbor to
transmits its data the mobile node keep the point q in
table second_priority
                        %disp('add to second priority q');
                        second_priority{pos_second_pr} = q;

                        if rate_check_flag==0

second_priority_rate{pos_second_pr} = sum_sending_rate_q;
                        else

second_priority_rate{pos_second_pr} =  sum_rate_q;
                        end
                        pos_second_pr=pos_second_pr+1;
                    end
                end



                if( (can_locate_p & pflag==1 & pneigh==1 &
rate_check_flag==0) | ( (sum_rate_p>=limit_rate) &
(rate_check_flag==1) ))  %all elements of the subset
intersect at p
                    if( pdist2(sink,p,'euclidean') <
min_distance )
                        all_points{pos_all_points} = p;
%point I
                        pos_all_points=pos_all_points+1;
                        min_distance =
pdist2(sink,p,'euclidean');
                    end
                else if( (can_locate_p & pflag==1 &
pneigh==0 & rate_check_flag==0) | (sum_rate_p>=limit_rate &
rate_check_flag==1 & pneigh==0)) %if there is not neighbor
to transmits its data the mobile node keep the point p in
table second_priority
                        %disp('add to second priority p');
                        second_priority{pos_second_pr} =
p;

                        if rate_check_flag==0

second_priority_rate{pos_second_pr} = sum_sending_rate_p;
                        else
```

```
second_priority_rate{pos_second_pr} =  sum_rate_p;
                        end

                        pos_second_pr=pos_second_pr+1;
                    end

                end

            end
        end
        %disp('mindistance');
        %min_distance
    end %~(isempty(xout)) && ~(isempty(yout))
end %second for
%first for

%find the point that is closest to the sink
minimumm_dist=inf;
all_points
disp(sprintf('length(all_points) %d',length(all_points)));
for i=1:length(all_points)
    all_points{i}
    if minimumm_dist >
pdist2(sink,all_points{i},'euclidean');

minimumm_dist=pdist2(sink,all_points{i},'euclidean');
        point_output=all_points{i};
    end
end
if (~isempty(point_output))
    I=point_output;
end

%crate all the possible subsets of k elements from n
positions and save in
%variable subsets, that has total sending rate >=
additional_resources
%node is the id of the congested node
%set are the neighbors og the congested node that have
sending)rate to it >0
function subsets=create_subs(k, n, additional_resources_j,
node, set)
global ID NEIGHBORS ATTRIBUTES DESTINATIONS ROUTINGTABLE
RTOTAL STOTAL RNUMBERS t RNEIGH
disp('subsets');
for i = 1:k
    p ( i ) = i-1 ;
end
```

```
subsets=[];
pos=1;
while ( 1 )

    sum=0;
    for i = 1 : k
        cur_sub(i)=set( p ( i )+1) ;
        sum=sum+RNEIGH{node}(cur_sub(i));
        disp(sprintf ( 'here %d  RNEIGH{node}(cur_sub(i)
%d', p ( i )+1 ,RNEIGH{node}(cur_sub(i))));
    end

    if sum/(t/40000) >= additional_resources_j
        [ x y]=size(cur_sub);

        subsets{pos}=cur_sub(1:y);
        subsets{pos}
        pos=pos+1;
    end

    if(p(1) == (n-k))
        return;
    end
    i = k;
    %find the right element
    while (  i >= 1 && p(i)+k-(i-1) == n)
        i=i-1 ;
    end
    r = p(i) ;

    p(i)=p(i)+1;
    j=2 ;
    %disp(sprintf ( 'r: %d  i: %d  p(i): %d' ,r,i,p(i) ));

    i=i+1;

    while(i <= k)
                p ( i ) = r+j ;
        i=i+1;
        j=j+1;
    end

end

%{
CRETATE ALTERNATIVE PATH FOR SINGLE NODE
Input: sstart (sstart.x, sstart.y)  the coordinates of the
sensor from which will start the alternative path
Input: send (send.x, send.y)  the coordinates of the sensor
that will end the alternative path
```

```
Output: the coordinates of the mobile node
%}
function
point=create_alt_path_single_node(sstart_x,sstart_y,
send_x, send_y)
global ID NEIGHBORS type_of_MobileCC ATTRIBUTES
DESTINATIONS ROUTINGTABLE RTOTAL STOTAL RNUMBERS t xsink
ysink
sstart_x
sstart_y
send_x
send_y
r=2.2;
sink=[xsink ysink];
point=[];
closest_point=[]; %if the sensor that will comnunicate with
mobile node is in the same straight line as the sink, this
is the most close point
%the coordinates of the sink
disp('create_alt_path_single_node');

X=[sstart_x,sstart_y;  send_x, send_y];
if (pdist(X,'euclidean') < r )
    disp('no need of mobile node');
    point=[];
    return;
else
    %find the equation of the circle C with radius r and
centre the point sstart: (x-sstart.x)^2 + (y-sstart.y)^2 =
r^2;
    %find point p
    m = ( send_y - sstart_y) / ( send_x - sstart_x );

    m
    if isinf(m)

        intercept=send_x;
    else
        intercept = send_y - (m*send_x);

    end
    m
    intercept
    %find the equation E that intersect the points sstart
and send : y- sstart.y = m(x - sstart.x);
    %find the point of intersection p(x,y) of the circle C
and the equation E;
    [x, y] = linecirc(m,intercept,sstart_x,sstart_y,r);

    x
```

```
    y
    x1=x(1);
    y1=y(1);
    x2=x(2);
    y2=y(2);
    p=[x1 y1];
    q=[x2 y2];
    pneigh=0;
    qneigh=0;

    if abs(send_x-sstart_x)<0.0001
        if send_y>sstart_y
            sx=sstart_x;
            sy=sstart_y+r;
            closest_point=[sx, sy];

        else if send_y<sstart_y
                sx=sstart_x;
                sy=sstart_y-r;
                closest_point=[sx, sy];
            end
        end
    else
        if abs(sstart_y-send_y)<0.0001
            if sstart_x> send_x
                sx=sstart_x-r;
                sy=sstart_y;
                closest_point=[sx, sy];
            else if sstart_x< send_x
                    sx=sstart_x+r;
                    sy=sstart_y;
                    closest_point=[sx, sy];
                end
            end
        end
    end

    if ~isempty(closest_point)
        if
le(pdist2(q,sink,'euclidean'),pdist2(p,sink,'euclidean'))
            q=closest_point;
        else
            p=closest_point;
        end
    end

    closest_point
    l=1;
    while( (l <length(ATTRIBUTES)) && (pneigh==0 ||
qneigh==0) )
```

```
        Y=[ATTRIBUTES{l}.x, ATTRIBUTES{l}.y];
        if( ATTRIBUTES{l}.hard_stage==0 )  %the node is not
congested

            if( le(pdist2(p,Y,'euclidean'),r) )  %check if
there is neighbor to transmits the mobile node at point q
its data
                pneigh=1;
            end
            disp(sprintf(' apostasi q : x %f y %f  =  %f',
ATTRIBUTES{l}.x, ATTRIBUTES{l}.y,pdist2(q,Y,'euclidean')));

            if( le(pdist2(q,Y,'euclidean'),r) )  %check if
there is neighbor to transmits the mobile node at point q
its data
                qneigh=1;
            end
        end
        l=l+1;
    end

    p
    q
    pneigh
    qneigh

    min_dist=0.5;
    [can_locate_p, can_locate_q, NEAR_POINTS_P,
NEAR_POINTS_Q, newp,newq ]=check_empty_place(p,q,[sstart_x,
sstart_y]);


    if(pneigh==1 && can_locate_p==1)
        point=p;
        %disp('point=p;');
    else
        if(qneigh==1 && can_locate_q==1)
            point=q;

        end
    end
    if type_of_MobileCC>1
        qneigh=1;
        pneigh=1;
    end
    if(pneigh==1 && qneigh==1 && can_locate_p==1 &&
can_locate_q==1)
        if (pdist2(p, sink,'euclidean') < pdist2(q,
sink,'euclidean') )
```

```
            %disp('point=p;');
            point=p;
        else
            point=q;
            %disp('point=q;');
        end
    end
    if(isempty(point))
        if (can_locate_p==1 && can_locate_q==1)
            if (pdist2(p, sink,'euclidean') < pdist2(q,
sink,'euclidean') )

                point=p;
            else
                point=q;

            end
        else
            if can_locate_p==1
                point=p;
            else
                if can_locate_q==1

                    point=q;
                end
            end
        end
    end

end



%DIRECT PATH ALGORITHM

function direct_path_algorithm()
global IDS_OF_MOB_NODES xsink ysink IDsink ATTRIBUTES
MOBILE_NODES USED_MOBILE_NODES
sink=[xsink ysink];
r=2.2;
p=-1;
 [p, mobID]=dynamic_algorithm(1, [], []); %return also the
number of the mobile node that located at p

%creates the sequence from the first mobile node that will
locate
%(using the dynamic algorithm) until the sink (every node
is located at
%distance equal to r from the previous mobile node)
```

```
function direct_sequence(p,mobID)
%disp('direct_sequence');
r=2.2;
global xsink ysink IDsink ATTRIBUTES MOBILE_NODES
USED_MOBILE_NODES IDS_OF_MOB_NODES
sink=[xsink ysink];
% disp(sprintf('+++dynamic_algorithm apostasi:
%f',pdist2(p, sink,'euclidean')));
no_of_mobile_nodes=0;
for i=1:length(IDS_OF_MOB_NODES)
    if MOBILE_NODES(IDS_OF_MOB_NODES(i))==1 &
USED_MOBILE_NODES(IDS_OF_MOB_NODES(i))==0
        no_of_mobile_nodes=no_of_mobile_nodes+1;
    end
end
%no_of_mobile_nodes
%locate node at p
sink
p
if(~isempty(p) & p~=-1)
    while(no_of_mobile_nodes>0 && pdist2(p,
sink,'euclidean') > r)
        p=create_alt_path_single_node(p(1), p(2),
ATTRIBUTES{IDsink}.x, ATTRIBUTES{IDsink}.y);

        if(~ isempty(p))%locate node at p

            ATTRIBUTES{mobID}.serve=1;
            locate_out{1}(1)=mobID;
            locate_out{2}=p;
            locate_out{3}=mobID;

mobID=locate_mobile_node(locate_out,1,mobID,[]);
            mobID


        else
            return;

        end


        no_of_mobile_nodes=no_of_mobile_nodes-1;
    end

end
%create and send a msg to app_layer to move all the mobile
nodes to
%their location
function locate_all_mobile_node()
```

```
global ID N t MOBILE_NODES USED_MOBILE_NODES
MOBILE_ADD_RESOURCES
locatemobile.msgID = -99;
locatemobile.from = ID;
locatemobile.address = ID;
status = app_layer(N, make_event(t, 'Send_Packet', ID,
locatemobile));
%create and send a msg to app_layer to move the appropriate
mobile node to the
%location p{1}
function
[mobID]=locate_mobile_node(p,additional_resources,congested
_node,cover_set)
global ID N t MOBILE_NODES USED_MOBILE_NODES
MOBILE_ADD_RESOURCE MOBILE_POSSIBLE_COVER_SETS
locatemobile.msgID = -88;
locatemobile.from = ID;
locatemobile.address = ID;
mobID=[];
i=1;

while i< length(MOBILE_NODES)+1 && isempty(mobID)
    if(MOBILE_NODES(i)==1 && USED_MOBILE_NODES(i)==0)
        USED_MOBILE_NODES(i)=1;

        mobID=i;
        break;
    end
    i=i+1;
end
if ~isempty(mobID)
    if(~isempty(cover_set))
        MOBILE_POSSIBLE_COVER_SETS{mobID}=cover_set;
    end
    ATTRIBUTES{congested_node}.serve=1;
    ATTRIBUTES{congested_node}.hard_stage=0;
    p{4}=mobID;

    if(~isempty(additional_resources))
        additional_resources
        if length(additional_resources)>1

MOBILE_ADD_RESOURCE{mobID}(congested_node)=additional_resou
rces(1);
        else

MOBILE_ADD_RESOURCE{mobID}(congested_node)=additional_resou
rces;
        end
    end
```

```
end
if ~isempty(mobID)
    locatemobile.data=p;
    %send msg to locate the mobile node to the position p
    status = app_layer(N, make_event(t, 'Send_Packet', ID,
locatemobile));
    disp(sprintf('locatemobile %d',ID));
else
    % disp('there is not a mobile node available');
end

function [just_served_flag] = check_if_just_served(node)
global t MOBILE_SERVE_TIME
just_served_flag=0;

MOBILE_SERVE_TIME(node)
if MOBILE_SERVE_TIME(node)~=0 && t-
MOBILE_SERVE_TIME(node)<85000
    just_served_flag=1;
end
%round a number to format y (e.g. 0.1) decimal places
function z = round2(x,y)
%% defensive programming
error(nargchk(2,2,nargin))
error(nargoutchk(0,1,nargout))
if numel(y)>1
    error('Y must be scalar')
end


z = round(x/y)*y;

%check if the place that the mobile node will locate is
free and is not
%negative
function [can_locate_p, can_locate_q, NEAR_POINTS_P,
NEAR_POINTS_Q ,p,q]=check_empty_place(p, q, point_serve)
global ID NEIGHBORS ATTRIBUTES DEST_MOBILE_NODES
ROUTINGTABLE USED_MOBILE_NODES IDS_OF_MOB_NODES

if isempty(p)
    can_locate_p = 0;
end
if isempty(q)
    can_locate_q = 0;
end
if isempty(p) && isempty(q)
    return;
end
```

```
min_dist=0.5;
pos_q=1;
pos_p=1;
r=2.2;
NEAR_POINTS_P=[];
NEAR_POINTS_Q=[];
can_locate_p = 1; %true if not exist another node in the
poosition p
can_locate_q = 1; %true if not exist another node in the
poosition q
e=0.000001;

if(p(1)<0 || p(2)<0)
    %disp('cannot locate mob at p because at least on of
its coordinates are negative');
    can_locate_p=0;
end
if(q(1)<0 || q(2)<0)
    %disp('cannot locate mob at q because at least on of
its coordinates are negative');
    can_locate_q=0;
end

% check if there is another node located to the position
that the
% mobile node will locate
for k=1:length(ATTRIBUTES)
    if( abs(ATTRIBUTES{k}.x-p(1))<e && abs(ATTRIBUTES{k}.y-
p(2))<e) %there is another node in the position p
        can_locate_p=0;
        %disp('cannot locate mob 1');
    end
    if( abs(ATTRIBUTES{k}.x-q(1))<e && abs(ATTRIBUTES{k}.y-
q(2))<e) %there is another node in the position q
        can_locate_q=0;
        % disp('cannot locate mob 2');
    end
    X=[ATTRIBUTES{k}.x, ATTRIBUTES{k}.y];
    if (pdist2(X, p,'euclidean')<=min_dist)
        NEAR_POINTS_P{pos_p}=X;
        pos_p=pos_p+1;
    end
    if (pdist2(X, q,'euclidean')<=min_dist)
        NEAR_POINTS_Q{pos_q}=X;
        pos_q=pos_q+1;
    end
    if( can_locate_p==0 && can_locate_q==0 )
        return;
    end
end
```

```
%check the positions of mobile nodes that are traveled if
will be
%the same

if ~isempty(DEST_MOBILE_NODES)
    for k=1:length(IDS_OF_MOB_NODES)
        %disp('DEST_MOBILE_NODES');

        if USED_MOBILE_NODES(IDS_OF_MOB_NODES(k))==1
            DEST_MOBILE_NODES{IDS_OF_MOB_NODES(k)}
            if(
abs(DEST_MOBILE_NODES{IDS_OF_MOB_NODES(k)}(1)-p(1))<e &&
abs(DEST_MOBILE_NODES{IDS_OF_MOB_NODES(k)}(2)-p(2))<e)
%there is another node in the position p
                can_locate_p=0;
                %disp('cannot locate mob 3');
            end

if(abs(DEST_MOBILE_NODES{IDS_OF_MOB_NODES(k)}(1)-q(1))<e &&
abs(DEST_MOBILE_NODES{IDS_OF_MOB_NODES(k)}(2)-q(2))<e)
%there is another node in the position q
                can_locate_q=0;
                % disp('cannot locate mob 4');
            end
            if( can_locate_p==0 && can_locate_q==0 )
                return;
            end

            if
(pdist2(DEST_MOBILE_NODES{IDS_OF_MOB_NODES(k)},
p,'euclidean')<=min_dist)
                NEAR_POINTS_P{pos_p}=X;
                pos_p=pos_p+1;
            end
            if
(pdist2(DEST_MOBILE_NODES{IDS_OF_MOB_NODES(k)},
q,'euclidean')<=min_dist)
                NEAR_POINTS_Q{pos_q}=X;
                pos_q=pos_q+1;
            end
        end
    end
end
can_locate_p
can_locate_q

%global_algorithm
```

```
%creates the optimal paths using static and mobile nodes to
deal with congestion using information for the whole
network

function global_algorithm()

global set_parents_of_mob_nodes IDS_OF_MOB_NODES
USED_MOBILE_NODES ID DEST_MOBILE_NODES NEIGHBORS ATTRIBUTES
DESTINATIONS ROUTINGTABLE RTOTAL STOTAL RNUMBERS t xsink
ysink IDsink RNEIGH
sink=[xsink ysink];
additional_resources=0;
disp('globall');

findp=0;
p=[];
mobID=[];
r=2.2;
%start phase A that finds the minimum number of mobile
%nodes that will be used to solve the congestion

%positions to additiona_resources are according the
sorted_nodes_needs_mobile
sorted_nodes_needs_mobile =
sort_distance_from_sink(nodes_needs_mobile,0);
for i=1:length(sorted_nodes_needs_mobile)
maxrneigh=(max(RNEIGH{sorted_nodes_needs_mobile(i)}))/(t/40
000);

additional_resources(i)=min(maxrneigh,additional_resources(
i));
    end
for i=1:length(sorted_nodes_needs_mobile)
    %the first will called the dynamic algorithm
    if sum(USED_MOBILE_NODES)>=1
        serve_rate =
serve_from_located_mobile_nodes(sorted_nodes_needs_mobile(i
), additional_resources(i));
    else
        [p, mobID]=dynamic_algorithm(2,
sorted_nodes_needs_mobile(i),additional_resources(i));
    end
end

%end phase A
%start phase B
for i=1:length(IDS_OF_MOB_NODES)

    if( USED_MOBILE_NODES(IDS_OF_MOB_NODES(i)) &&
~isempty(DEST_MOBILE_NODES{IDS_OF_MOB_NODES(i)}))
```

```
        if
pdist2(sink,DEST_MOBILE_NODES{IDS_OF_MOB_NODES(i)},
'euclidean')>r
            %create paths for all the mobile nodes that
will locate

dynamic_algorithm_find_cost_for_paths(IDS_OF_MOB_NODES(i),
sorted_nodes_needs_mobile);  %dynamic algorithm

        else

set_parents_of_mob_nodes(IDS_OF_MOB_NODES(i))=IDsink;
        end
    end
end

%end phase B
%locate all mobile nodes
locate_all_mobile_node();

%sort the input nodes by the distance from the sink
%the node that has the largest distance will be first
%if type==0 then the sorting is descending - starting from
the node that
%is more away from the sink
%if type==1 then the sorting is ascending - starting from
the node that
%is more near from the sink
function  sorted_nodes =
sort_distance_from_sink(nodes_needs_mobile,type)
global ID notserved ATTRIBUTES MOBILE_NODES NEIGHBORS
buffer_capacity NODE_USES_MOBILE_NODES
NEIGH_NODES_USES_MOBILE_NODES USED_MOBILE_NODES
MOBILE_NODE_RATE RTOTAL STOTAL RNUMBERS t xsink ysink
IDsink RNEIGH DEST_MOBILE_NODES
sink=[xsink ysink];
sorted_nodes_needs_mobile=[];
for i=1:length(nodes_needs_mobile)
    if MOBILE_NODES(nodes_needs_mobile(i))==1

nodecooord=DEST_MOBILE_NODES{nodes_needs_mobile(i)};
    else

nodecooord=[ATTRIBUTES{nodes_needs_mobile(i)}.x,ATTRIBUTES{
nodes_needs_mobile(i)}.y];
    end

distances_congested_nodes_from_sink(i)=pdist2(nodecooord,si
nk, 'euclidean');
end
```

```
%distances_congested_nodes_from_sink
if type==0
    %start from the node that is more away from the sink-
sort descend

set=sort(distances_congested_nodes_from_sink,'descend');
else
    %start from the node that is more near to the sink-sort
ascend
    set=sort(distances_congested_nodes_from_sink,'ascend');
end

%set

[a,b] = histc(set,unique(set));
%counter has the number of times that a number exist in the
array set
counter = a(b);
current_counter=ones(1, length(set));
for i=1:length(set)
    d=set(i);
    pos=find(distances_congested_nodes_from_sink== d);
    if counter(i)>1 %more than one number has this number
of rneigh
        current_counter(i)
        pos=pos(current_counter(i));
        pos
        for k=1:length(set)
            if d==set(k)
                current_counter(k)=current_counter(k)+1;
            end
        end

    else
        pos=pos(1);
    end

        sorted_nodes(i)=nodes_needs_mobile(pos);
end
%check if the ccongested node can send some or all of its
%rate to the already located mobile nodes
function serve_rate =
serve_from_located_mobile_nodes(congested_node,
additional_resources_i)
global ID  ATTRIBUTES NEIGHBORS MOBILE_POSSIBLE_COVER_SETS
buffer_capacity NODE_USES_MOBILE_NODES
NEIGH_NODES_USES_MOBILE_NODES USED_MOBILE_NODES
MOBILE_NODE_RATE RTOTAL STOTAL RNUMBERS t xsink ysink
IDsink RNEIGH DEST_MOBILE_NODES
sink=[xsink ysink];
```

```
serve_rate=additional_resources_i;
i=1;
r=2.2;

while i<length(USED_MOBILE_NODES) && serve_rate>0
    i
    MOBILE_NODE_RATE(i)
    if(USED_MOBILE_NODES(i)==1 &
MOBILE_NODE_RATE(i)<(buffer_capacity-0.5))
        additional_resources = min(serve_rate,
buffer_capacity-MOBILE_NODE_RATE(i) );

                NODE_USES_MOBILE_NODES{i}
        NEIGH_NODES_USES_MOBILE_NODES{i}
        %find the set of nodes that send their data to the
        %congested node
        set=find_neigh_senders_sorted(congested_node);
        z=1;
        %check if any of the nodes that send their data to
        %the congested node can send them to the mobile
        %node i
        %start from the neighbor of the congested_node that
        %send the greater rate to it
        while z<length(set)+1 && serve_rate>0
            disp(sprintf('check neighbor %d',set(z)));

nodecooord=[ATTRIBUTES{set(z)}.x,ATTRIBUTES{set(z)}.y];
            dis=pdist2(nodecooord,DEST_MOBILE_NODES{i},
'euclidean');
            dist=round2(dis,0.1);
X=find(NEIGH_NODES_USES_MOBILE_NODES{i}==set(z));
            if (dist<=r ||
~isempty(find(NEIGH_NODES_USES_MOBILE_NODES{i}==set(z))))
                serve_rate=serve_rate-additional_resources;

NEIGH_NODES_USES_MOBILE_NODES{i}=union(NEIGH_NODES_USES_MOB
ILE_NODES{i},set(z));

s=union(congested_node,NODE_USES_MOBILE_NODES{i});
                NODE_USES_MOBILE_NODES{i}=s;
                for
m=1:length(MOBILE_POSSIBLE_COVER_SETS{i})

MOBILE_POSSIBLE_COVER_SETS{i}{m}=union(MOBILE_POSSIBLE_COVE
R_SETS{i}{m},set(z));
                                    end

            end
            z=z+1;
        end
```

```
    end
    i=i+1;
end
i=1;

while i<length(USED_MOBILE_NODES) && serve_rate>0
    if(USED_MOBILE_NODES(i)==1 &
MOBILE_NODE_RATE(i)<(buffer_capacity-0.5))
        additional_resources = min(serve_rate,
buffer_capacity-MOBILE_NODE_RATE(i) );
        %find if their is a point p to locate the mobile
        %node that covers both the
        %NEIGH_NODES_USES_MOBILE_NODES and a neighbor of
        %the congested node


[p,neigh_serve,new_cover_sets]=connectivity_algorithm(1, 6,
congested_node,
additional_resources,MOBILE_POSSIBLE_COVER_SETS{i});

        if ~isempty(p)
            MOBILE_POSSIBLE_COVER_SETS{i}=new_cover_sets;

            DEST_MOBILE_NODES{i}=p;
            serve_rate=serve_rate-additional_resources;

NEIGH_NODES_USES_MOBILE_NODES{i}=union(NEIGH_NODES_USES_MOB
ILE_NODES{i},neigh_serve);

s=union(congested_node,NODE_USES_MOBILE_NODES{i});
                        NODE_USES_MOBILE_NODES{i}=s;

        end

    end
    i=i+1;
end
%if still have unserved rate the congested node, must
%locate a new mobile node for congested_node
if serve_rate>0
    disp(sprintf('call  dynamic_algorithm in serve rate
because serve_rate is %f',serve_rate));
    [p, mobID]=dynamic_algorithm(2,
congested_node,serve_rate );
    %notserved=union(congested_node, notserved);
else
    ATTRIBUTES{congested_node}.needs_mobile = 0;
    ATTRIBUTES{congested_node}.serve=1;
end
```

```
%call when the number of the sets is greater than one


%GLOBAL ALGORITHM - CREATE THE SINGLE PATH FOR EVERY NODE
IN LEVEL SMALLER THAN THE LEVEL OF CONGESTED NODE THAT IS
NOT CONGESTED (FROM THE SINK TO NODE) ---
%Dynamic programming


%{
Input: the mobile node m, near the congested node
Output: a table C that has in each position i the cost of
the optimal path from the sink to node i for each node i
Output: a table free_rate that has in each position i the
free_rate of the optimal path from the sink to node i for
each node i
Output: a table M that has in each position i the number of
mobile nodes needed to create the optimal path from the
sink to node i for each node i
%}
function dynamic_algorithm_find_cost_for_paths(mobID,
congested_nodes)
global ID ATTRIBUTES start_send_data MOBILE_NODES NEIGHBORS
MOBILE_POSSIBLE_COVER_SETS buffer_capacity
NODE_USES_MOBILE_NODES NEIGH_NODES_USES_MOBILE_NODES
USED_MOBILE_NODES MOBILE_NODE_RATE RTOTAL STOTAL RNUMBERS t
xsink ysink IDsink RNEIGH DEST_MOBILE_NODES
IDS_OF_MOB_NODES
sink=[xsink ysink];
r=2.2;
for i=1:length(ATTRIBUTES)+50
    points_mob_nodes_may_used{i}=[];
end


%initalize the tables for the first node (the sink)
mobID_coord=DEST_MOBILE_NODES{mobID};
dist_mob_node_sink=pdist2(sink, mobID_coord, 'euclidean');
C(IDsink) = 0;  %table c[i] represents the cost to create
an optimal path from node i to the sink that includes
static and mobile nodes
M(IDsink)=0;
free_rate(IDsink)=5;
B(IDsink) = 0;
all_nodes=[];

for i=1:length(ATTRIBUTES)+50 %for each node static or
mobile (taht is used) of the network
    if(i~=IDsink)
```

```
        C(i) = inf;
        M(i)=inf;
        B(i)=0;
        C_mob_nodes_needed(i)=0;
        free_rate(i)=buffer_capacity;
    end
end

for i=1:length(ATTRIBUTES) %for each node static or mobile
(taht is used) of the network
    node_coord=[ATTRIBUTES{i}.x, ATTRIBUTES{i}.y];
    dis=pdist2(sink, node_coord, 'euclidean');
    if IDsink~=i && i~=mobID && dis<=dist_mob_node_sink+r
&& isempty(find(congested_nodes==i))%the sink have already
initialised
        if MOBILE_NODES(i)==1 %if is mobile node and it is
used
            disp(sprintf('free_rate %d',i));
            free_rate(i)=buffer_capacity-
MOBILE_NODE_RATE(i);
            if USED_MOBILE_NODES(i)==1
                all_nodes=union(all_nodes,i);
            end
        end

        if(MOBILE_NODES(i)==0)  %if is not mobile node
            free_rate(i)=buffer_capacity-(RTOTAL{i}/((t-
start_send_data)/40000));
            all_nodes=union(all_nodes,i);
        end
    end
end

%sort the nodes from the node that is closest to sink to
the node that
%is more away from it al nodes expe
if( ~isempty(all_nodes))
    sorted_nodes_used =
sort_distance_from_sink(all_nodes,1);
    sorted_nodes_used=[sorted_nodes_used, mobID];

    % sorted_nodes_used

    for j=1:length(sorted_nodes_used) %first for each
static node of the network, and then for each mobile node
of the network starts from the node closer to the sink,
(from nodes with level 1 to max_level)
        if j~=mobID %not the mob node located
            node_coord=[ATTRIBUTES{sorted_nodes_used(j)}.x,
ATTRIBUTES{sorted_nodes_used(j)}.y];
```

```
            dis=pdist2(sink, node_coord, 'euclidean');

            if (USED_MOBILE_NODES(sorted_nodes_used(j))==0
|| MOBILE_NODE_RATE(sorted_nodes_used(j))<buffer_capacity)
%&& ATTRIBUTES{sorted_nodes_used(j)}.hard_stage==0  %if the
level of the static node is smaller than the mobile node
and the node is not congested

[B,C,M,free_rate,mobi,points_mob_nodes_may_used]=find_best_
neighbor(sorted_nodes_used(j), B, C,M,
free_rate,congested_nodes,points_mob_nodes_may_used);

                while(~isempty(mobi))


[B,C,M,free_rate,mobi,points_mob_nodes_may_used]=find_best_
neighbor(mobi, B, C, M,
free_rate,congested_nodes,points_mob_nodes_may_used);
                end

            end

            if (~isinf(C(mobID)))
                %find a path for the mobile node to forward
it's data to
                %the sink
                disp(sprintf('find a path for the mobile
node %d C(mobID) %d',mobID,C(mobID)));
                return;
                %else %if there is not appropriate static
node, locate a mobile node


            end

        end %if j~=mobID

    end    %for
    %check if the optimal path from mobID to sink includes
mobile nodes and
    %if includes located them

    if M(mobID)>0
        %edit

find_solution_path(mobID,B,points_mob_nodes_may_used);

    end
else
    %must locate a new mob node
```

```
    if(~isempty(mobID))

this_point=create_alt_path_single_node(DEST_MOBILE_NODES{mo
bID}(1), DEST_MOBILE_NODES{mobID}(2), ATTRIBUTES{IDsink}.x,
ATTRIBUTES{IDsink}.y);

        %this_point
        if(~isempty(this_point))
            this_point
            locate_out{1}(1)=mobID;
            locate_out{2}=this_point;
            locate_out{3}=[mobID];
            add(1)=0.0;

mobID=locate_mobile_node(locate_out,[],mobID,[]);
            %call th find best neighbor for the new
            %mob that will located

            while(~isempty(mobID))

[B,C,M,free_rate,mobID,points_mob_nodes_may_used]=find_best
_neighbor(mobID, B, C,
M,free_rate,congested_nodes,points_mob_nodes_may_used);
            end

            %locate mobile node at p

        end
        if M(mobID)>0


find_solution_path(mobID,B,points_mob_nodes_may_used);

        end
    end
end



%{
--------------------------------- FIND BEST NEIGHBOR ------
-------------------
Input: the last node of the path
Output: find the best neighbor of node j
%}
function
[B,C,M,free_rate,mobi,points_mob_nodes_may_used]=find_best_
neighbor(j, B, C,
M,free_rate,congested_nodes,points_mob_nodes_may_used)
```

```
global ID set_parents_of_mob_nodes counter_extra_nodes
IDS_OF_MOB_NODES start_send_data ATTRIBUTES MOBILE_NODES
NEIGHBORS MOBILE_POSSIBLE_COVER_SETS buffer_capacity
NODE_USES_MOBILE_NODES NEIGH_NODES_USES_MOBILE_NODES
USED_MOBILE_NODES MOBILE_NODE_RATE RTOTAL STOTAL RNUMBERS t
xsink ysink IDsink RNEIGH DEST_MOBILE_NODES
sink=[xsink ysink];
r=2.2;
points_mob_nodes_may_used=points_mob_nodes_may_used;
mobi=[];
node_send_through=[];
sorted_NEIGHBORS_j=[];
NEIGH_MOB=[];
if(j<=length(ATTRIBUTES))
    if MOBILE_NODES(j)==1
        coordj = DEST_MOBILE_NODES{j};
    else
        coordj = [ATTRIBUTES{j}.x, ATTRIBUTES{j}.y];
    end
else
    coordj=points_mob_nodes_may_used{j};
        coordj
end
dis_j_sink=pdist2(coordj,sink, 'euclidean');

%calculates the NEIGH_MOB whick includes the neighbors
static and mobile nodes that
%have C(z) not inf and are in the range of node j
if(j<=length(ATTRIBUTES))
    if( MOBILE_NODES(j)==0 && ~isempty(NEIGHBORS{j})) %if
input node is not a mobile node
        for z=1:length(NEIGHBORS{j})
            if  ~isinf( C(NEIGHBORS{j}(z))) &&
(MOBILE_NODES(NEIGHBORS{j}(z))==0 ||
USED_MOBILE_NODES(NEIGHBORS{j}(z))==1) &&
isempty(find(congested_nodes==NEIGHBORS{j}(z)))
                NEIGH_MOB=union(NEIGH_MOB,NEIGHBORS{j}(z));
            end
        end
        %NEIGH_MOB
        if(~isempty(NEIGH_MOB))
            sorted_NEIGHBORS_j =
sort_distance_from_sink(NEIGH_MOB,1);
        end
    end

    %check if could move a mobile node to create the path
    if( MOBILE_NODES(j)==1) %if input node is a mobile node
                for z=1:length(ATTRIBUTES)
            coordz=[ATTRIBUTES{z}.x, ATTRIBUTES{z}.y];
```

```
            if  ( ~isinf( C(z)) && z~=j &&
(MOBILE_NODES(z)==0 || USED_MOBILE_NODES(z)==1) &&
pdist2(coordj,coordz, 'euclidean') <= r) &&
isempty(find(congested_nodes==z))
                %can communicate through the mobile node
                NEIGH_MOB=union(NEIGH_MOB,z);
            end
        end
        NEIGH_MOB
        if ~isempty(NEIGH_MOB)
            sorted_NEIGHBORS_j =
sort_distance_from_sink(NEIGH_MOB,1);
        end
    end
else
    for z=1:length(ATTRIBUTES)
        coordz=[ATTRIBUTES{z}.x, ATTRIBUTES{z}.y];
        if  ( ~isinf( C(z)) && z~=j && (MOBILE_NODES(z)==0
|| USED_MOBILE_NODES(z)==1) && pdist2(coordj,coordz,
'euclidean') <= r) && isempty(find(congested_nodes==z))
            %can communicate through the mobile node
            NEIGH_MOB=union(NEIGH_MOB,z);
        end
    end
    if ~isempty(NEIGH_MOB)
        sorted_NEIGHBORS_j =
sort_distance_from_sink(NEIGH_MOB,1);
    end
end

%check first if there is a path to sink through any static
neighbor of j
for i=1:length(sorted_NEIGHBORS_j)%for each static neighbor
of j

    coordi=[ATTRIBUTES{sorted_NEIGHBORS_j(i)}.x,
ATTRIBUTES{sorted_NEIGHBORS_j(i)}.y];


    if j<=length(ATTRIBUTES)
        a=(RTOTAL{j}/((t-start_send_data)/40000));

    else
        a=0;
    end
    if  ~isinf(M(sorted_NEIGHBORS_j(i)))  &
free_rate(sorted_NEIGHBORS_j(i))>=a  %for each static
neighbor of j or mobile node that has point of intersection
with j
```

```
        if( M(sorted_NEIGHBORS_j(i))< M(j)  || isinf(M(j)))
%the path cost through node vi is less than the path cost
already mesaured
            C(j) = C(sorted_NEIGHBORS_j(i)) + 1;
            M(j)=M(sorted_NEIGHBORS_j(i));
            node_send_through=sorted_NEIGHBORS_j(i);
        else if  ( M(sorted_NEIGHBORS_j(i)) == M(j) )  %the
path cost through node vj is equal to the path cost already
mesaured
                if( C(sorted_NEIGHBORS_j(i))+1<C(j))
%and the free rate through node vj is greater
                    C(j) = C(sorted_NEIGHBORS_j(i)) + 1;

node_send_through=sorted_NEIGHBORS_j(i);
                end
            end
        end
    end

end %for

if ~isempty(node_send_through)
    if j<=length(ATTRIBUTES)
        a=free_rate(node_send_through)-(RTOTAL{j}/((t-
start_send_data)/40000));
        a
        if MOBILE_NODES(j)==1
            set_parents_of_mob_nodes(j)=node_send_through;
        end
    end
    free_rate(node_send_through) = a;
    B(j) = node_send_through;

    M(j)=M(node_send_through);

else %isempty(node_send_through)
    %if there is NOT a path to sink through any static
neighbor of j
    if j<=length(ATTRIBUTES)
        %for each mobile node that will located in the
network check if there is a path to sink through any mobile
neighbor of j
        for i=1:length(IDS_OF_MOB_NODES)
            if(isempty(node_send_through))
                if ~isinf( C(IDS_OF_MOB_NODES(i))) &&
j~=IDS_OF_MOB_NODES(i) &&
USED_MOBILE_NODES(IDS_OF_MOB_NODES(i))==1

                    %can communicate through the mobile
node if the node
```

```
                    %located at point p
                    if
isempty(MOBILE_POSSIBLE_COVER_SETS{IDS_OF_MOB_NODES(i)})

coverset{1}=NEIGH_NODES_USES_MOBILE_NODES{IDS_OF_MOB_NODES(
i)};
                    else

coverset=MOBILE_POSSIBLE_COVER_SETS{IDS_OF_MOB_NODES(i)};
                    end


[p,neigh_serve,new_cover_sets]=connectivity_algorithm(1, 6,
j, 0,coverset);

                    if( ~isempty(p))

node_send_through=IDS_OF_MOB_NODES(i);
                        C(j) = C(IDS_OF_MOB_NODES(i)) + 1;
                        a=free_rate(IDS_OF_MOB_NODES(i))-
(RTOTAL{j}/((t-start_send_data)/40000));
                        a
                        free_rate(IDS_OF_MOB_NODES(i)) = a;
                        B(j) = IDS_OF_MOB_NODES(i);



MOBILE_POSSIBLE_COVER_SETS{i}=new_cover_sets;
                        for
m=1:length(MOBILE_POSSIBLE_COVER_SETS{i})

MOBILE_POSSIBLE_COVER_SETS{i}{m}
                        end
                        DEST_MOBILE_NODES{i}=p;

NEIGH_NODES_USES_MOBILE_NODES{i}=union(NEIGH_NODES_USES_MOB
ILE_NODES{i},neigh_serve);

s=union(j,NODE_USES_MOBILE_NODES{i});

                        NODE_USES_MOBILE_NODES{i}=s;



                    end
                end
            end
        end

    end
```

```
    % node_send_through=[];
end %~isempty(node_send_through)
if isempty(node_send_through) &&
pdist2(sink,coordj,'euclidean')>r %must locate a mobile
node


    this_point=create_alt_path_single_node(coordj(1),
coordj(2), ATTRIBUTES{IDsink}.x, ATTRIBUTES{IDsink}.y);

    if(~isempty(this_point))
        this_point
        C(j)=C(j)+1;
        M(j)=M(j)+1;
        B(j) = counter_extra_nodes;

points_mob_nodes_may_used{counter_extra_nodes}=[this_point]
;

        mobi=counter_extra_nodes;

        counter_extra_nodes=counter_extra_nodes+1;

        %locate mobile node at this_point if
        %the path through the node j is the
        %optimal
    end
end

%{
---------------- RECURSIVE METHOD - FIND SOLUTION PATH ----
----------------

Input: the last node of the path
Output: the optimal path of static and mobile nodes from
the vj to sink


     The nodes that has ID greater than the length of
ATTRIBUTES e.g. the
     nodes of the network, are the mobile node that may
located if the are
     path include a mobile node it will be locatedvif they
are
     included in the optimal path from mobid to sink
%}
function find_solution_path(vj,B,points_mob_nodes_may_used)
global ID IDS_OF_MOB_NODES start_send_data
set_parents_of_mob_nodes ATTRIBUTES MOBILE_NODES NEIGHBORS
MOBILE_POSSIBLE_COVER_SETS buffer_capacity
```

```
NODE_USES_MOBILE_NODES NEIGH_NODES_USES_MOBILE_NODES
USED_MOBILE_NODES MOBILE_NODE_RATE RTOTAL STOTAL RNUMBERS t
xsink ysink IDsink RNEIGH DEST_MOBILE_NODES
disp(sprintf('call find solution path for node %d B(vj)
%d', vj,B(vj)));
if(vj==IDsink)
    return;
else
    points_mob_nodes_may_used=points_mob_nodes_may_used;
        if B(vj)==0
                return;
    end
    points_mob_nodes_may_used{B(vj)}

    if(B(vj)>length(ATTRIBUTES))
        %Locate mobile node at
points_mob_nodes_may_used{vj};
        locate_out{1}(1)=vj;
        points_mob_nodes_may_used{vj}
        locate_out{2}=points_mob_nodes_may_used{B(vj)};
        locate_out{3}=[vj];
        mobi=locate_mobile_node(locate_out,[],vj,[]);
        mobi
        o=B(vj);
        set_parents_of_mob_nodes(mobi)=B(o);
        set_parents_of_mob_nodes(vj)=mobi;


find_solution_path(B(vj),B,points_mob_nodes_may_used);
    else
        if vj<length(ATTRIBUTES) && MOBILE_NODES(vj)==1 &&
B(vj)<length(ATTRIBUTES)
            set_parents_of_mob_nodes(vj)=B(vj);
        end

find_solution_path(B(vj),B,points_mob_nodes_may_used);
    end
end
```

```
function status = Dalpas_layer(N, S)

% Written by Aris Pafits, csp7pa2@cs.ucy.ac.cy
% Last modified: Feb. 24 2015 by Antonia Nicolaou
% Implementation of the Algorithm DalPaS and the connection
with the
% MobileCC layer
% DO NOT edit simulator code (lines that begin with S;)

S; %%%%%%%%%%%%%%%%%%   housekeeping
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
S;      persistent app_data
S;      global ID t
S;      [t, event, ID, data]=get_event(S);
S;      [topology, mote_IDs]=prowler('GetTopologyInfo');
S;      ix=find(mote_IDs==ID);
S;      if ~strcmp(event, 'Init_Application')
    S;          try memory=app_data{ix}; catch memory=[];
end,
    S;      end
S;      global ATTRIBUTES
S;      status = 1;
S;      pass = 1;
S;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%          APPLICATION STARTS
%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%               HERE
%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
vvvvvvvvvvvvv%
global DESTINATIONS SOURCES
global NEIGHBORS start_send_data send_mob_packet
global RNUMBERS LTOTALS ROUTINGTABLE RNEIGH STOTAL RTOTAL
MOBILE_NODES USED_MOBILE_NODES STOTAL_generated
IDS_OF_MOB_NODES
```

```
global buffer_capacity Threshold AckStrength Perf_Thr_Delay
buff_threshold initial_pwr pwr_threshold called_mob_located

%NEIGHBORS: NEIGHBORS{ID} holds the list of nodes that node
ID can hear
%RNUMBERS:  RNUMBERS{ID} holds the total received packets
from each of its neighbors
%SNUMBERS:  SNUMBERS{ID} holds the total send packets from
each of its neighbors
%LTOTALS:   LTOTALS{ID} holds the total number of lost
packets to each of its neighbors
%persistent DATA_M   = 10;
INIT_H_MOB=-55;
ACK      = -11;
INIT_H   = -33;
CONNECT  = -22;
BACK     = -66;
MOBILE   = -44;
LOCATE_MOBILE = -88;


switch event
    case 'Init_Application'
        if (ix==1)
            %initialize only once
            sim_params('set_app', 'Promiscuous', 0);
            buff_threshold = sim_params('get_app',
'BufferThreshold');
            buff_threshold
            ATTRIBUTES{ID}.RB
            buffer_capacity
            if (isempty(Threshold)) Threshold = 0.1; end
            pwr_threshold = sim_params('get_app',
'PowerThreshold');
            if (isempty(Threshold)) Threshold = 0.5; end
            %get init power (initial power reserve on each
node from transmission_set_params)
            initial_pwr = sim_params('get_app',
'InitPower');
            if (isempty(initial_pwr)) initial_pwr = 10; end
            send_mob_packet=zeros(1, length(ATTRIBUTES)+1);

        end

        ATTRIBUTES{ID}.desireability = 1;
        ATTRIBUTES{ID}.hard_stage = 0;
        ATTRIBUTES{ID}.needs_mobile = 0;
        ATTRIBUTES{ID}.serve = 0;
        ATTRIBUTES{ID}.serve_neigh=0;
        STOTAL{ID} = [0];
```

```
        STOTAL_generated(ID) = [0];
    case 'Send_Packet'

        try msgID = data.msgID; catch msgID=0; end

        %calculate the initial time for sending packets
        if isempty(start_send_data)
            start_send_data=t;
            start_send_data
        end
        try from = data.from; catch from = 0; end
        if(msgID >= 0 & from~=0)
            STOTAL{ID} = STOTAL{ID}+1; %increased only when
the packet is not created by the node
        else
            if(msgID >= 0 )
                STOTAL_generated(ID) =
STOTAL_generated(ID)+1; %increased only when the packet is
not created by the node
            end
        end
        if msgID==LOCATE_MOBILE
            pass=1;
        end
        if (msgID >= 0)
            if DESTINATIONS(ID)
                pass = 0;
            end
            if(ATTRIBUTES{ID}.parent>-inf)
                if (ATTRIBUTES{ID}.parent == 0 |
ATTRIBUTES{ATTRIBUTES{ID}.parent}.hard_stage == 1) &
(~DESTINATIONS(ID))
                    %the parent node does not accept
packets
                    %run the flag decision algorithm to
choose next hop or raise hard_stage flag


                    new_parent = flag_decision
                    ATTRIBUTES{ID}.parent
                    new_parent
                    if ~isempty(new_parent) & new_parent ~=
0 & new_parent~=ATTRIBUTES{ID}.parent & new_parent~=ID
                        %update entries in Attributes:
parent, desireability and hops
                        ATTRIBUTES{ID}.parent = new_parent;
                        disp(sprintf('id %d new parent %d
',ID,new_parent));
                        ATTRIBUTES{ID}.hops =
ATTRIBUTES{ATTRIBUTES{ID}.parent}.hops + 1;
```

```
                        pass = 1;
                    else
                        ATTRIBUTES{ID}.hard_stage = 1;
                        p=ATTRIBUTES{ID}.parent;
                        ATTRIBUTES{p}.RB
                        disp(sprintf('hahahaha %d
ATTRIBUTES{parent}.rb %f ATTRIBUTES{ID}.parent %d
sum(rneigh)OF PARENT %f stotal %f RTOTAL{ID} %f
ATTRIBUTES{ID}.PACKETDROPS
%f',ID,ATTRIBUTES{p}.RB,p,sum(RNEIGH{p}),STOTAL{p},RTOTAL{p
},ATTRIBUTES{p}.PACKETDROPS));

if(ATTRIBUTES{p}.RB<=buff_threshold)
                            disp(sprintf('inahahahahaha
parent %d ', ATTRIBUTES{ID}.parent));
                            no_of_mobile_nodes=0;
                            for
i=1:length(IDS_OF_MOB_NODES)
                                if
USED_MOBILE_NODES(IDS_OF_MOB_NODES(i))==0

no_of_mobile_nodes=no_of_mobile_nodes+1;
                                end
                            end
                            if no_of_mobile_nodes>0
                                p=ATTRIBUTES{ID}.parent;
                                ATTRIBUTES{p}.needs_mobile
= 1;
                                mobile.msgID = MOBILE;
                                mobile.from = ID;
                                %mobile.init = ID;

mobile.address=ATTRIBUTES{ID}.parent;
                                disp(sprintf('hahahaha
%d',ID));
                                disp(sprintf('
NEIGHBORS{ID} of %d',ID));
                                NEIGHBORS{ID}
                                status = Dalpas_layer(N,
make_event(t, 'Send_Packet', ID, mobile));

                            end
                        end


                    end
                end
            end
```

```
            ATTRIBUTES{ID}.desireability =
6*ATTRIBUTES{ID}.RB + 2*(initial_pwr-
ATTRIBUTES{ID}.power)/initial_pwr - 2*ATTRIBUTES{ID}.hops;
            if SOURCES(ID)
                ATTRIBUTES{ID}.desireability = -inf;
            end
            %disp(sprintf('Node %d desireability: %f', ID,
ATTRIBUTES{ID}.desireability));
            data.desireability =
ATTRIBUTES{ID}.desireability;

            data.address = ATTRIBUTES{ID}.parent;
            data.hops = ATTRIBUTES{ID}.hops;
        end

    case 'Packet_Received'

        rdata = data.data;
        try msgID = rdata.msgID; catch msgID=0; end

        if msgID >= 0 & ~DESTINATIONS(ID)
            if ATTRIBUTES{ID}.RB < 0
                pass=0;
                return;
            end
        end
        if msgID >= 0 & DESTINATIONS(ID)
            pass=0;
        end

        if msgID==LOCATE_MOBILE
            pass=1;
        end

        if (msgID == MOBILE)
            disp(sprintf('its mobile msg ID %d', ID));

            if DESTINATIONS(ID)
                disp(sprintf('mobile msg at destination: %d
',ID));
                pass = 1;
            else
                send_mob_packet(ID)=t;
                ATTRIBUTES{ID}.parent
                if ~(isinf(ATTRIBUTES{ID}.parent))
                    p=ATTRIBUTES{ID}.parent;
                    mobile_fwd.msgID = MOBILE;
                    disp(sprintf('sent mb msg to parent
%d',ATTRIBUTES{ID}.parent));
```

```
                    mobile_fwd.from = ID;


mobile_fwd.address=ATTRIBUTES{ID}.parent;
                    status = Dalpas_layer(N, make_event(t,
'Send_Packet', ID, mobile_fwd));

                end

                pass = 0;

            end
        end

        %Check REMAINING BUFFER (set by buffer_q_layer.m),
remaining power or lower level node Availability
        %so I can choose if I have to enter or exit
hard_stage
              if (ATTRIBUTES{ID}.RB <= buff_threshold ||
ATTRIBUTES{ID}.parent == 0 || ATTRIBUTES{ID}.power <
pwr_threshold) && ( ~DESTINATIONS(ID)  )

            %enter hard stage
            ATTRIBUTES{ID}.hard_stage = 1;

            disp(sprintf('gotcha at node : %d
ATTRIBUTES{ID}.RB %f buff_threshold %f stotal %f rtotal %f
sum(RNEIGH{ID}) %f',ID,ATTRIBUTES{ID}.RB,buff_threshold,
STOTAL{ID},sum(RNEIGH{ID})));
            disp(ATTRIBUTES{ID}.RB <= buff_threshold);
            disp(ATTRIBUTES{ID}.parent == 0);
            disp(ATTRIBUTES{ID}.power < pwr_threshold);
        else
            ATTRIBUTES{ID}.hard_stage = 0;
        end

        if msgID < 0
            %do staff and then pass = 1
            if msgID == BACK
                pass = 1; %pass 1 to deal with it on the
next layer (init_span), where the memory parent is set.
            end

        else
            if DESTINATIONS(ID)
                pass = 1;
            end

            if rdata.address == ID
```

```
                %    %RNUMBERS requires layer Neighborhood
to exist.
                try r= RNUMBERS{ID}; catch r=0;end
                try n= NEIGHBORS{ID}; catch n=0;end

                % SOFT STAGE IMPLEMENTATION
                if length(RNUMBERS{ID} > 1)
                    input_streams = find(r>1 & r<max(r));
                    if length(input_streams) > 0 &&
~DESTINATIONS(ID)
                        for i=1:length(input_streams)
                            back.msgID = BACK;
                            back.from = ID;
                            back.address =
n(input_streams(i));
                            status = Dalpas_layer(N,
make_event(t, 'Send_Packet', ID, back));
                            DrawLine('Arrow', back.address,
ID, 'color', [1 0 1]);
                        end

                    end
                end
                %real data to me => forward the packet to
the next hop;
                %disp(sprintf('data from %d',rdata.from));
                rdata.from=rdata.from;
                status = Dalpas_layer(N, make_event(t,
'Send_Packet', ID, rdata));
            end
        end

end %end switch

S; %%%%%%%%%%%%%%%%%%%%%% housekeeping
%%%%%%%%%%%%%%%%%%%%%%%%%%
S;        try app_data{ix}=memory; catch app_data{ix} = [];
end
S;        if (pass) status = common_layer(N, make_event(t,
event, ID, data)); end
S;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

function b=Set_Flood_Clock(alarm_time);
global ID
data.type = 'spantree_flood';
prowler('InsertEvents2Q', make_event(alarm_time,
'Clock_Tick', ID, data));
```

```
function [new_parent] = flag_decision()
global ID NEIGHBORS ATTRIBUTES DESTINATIONS ROUTINGTABLE
buff_threshold
disp(sprintf('flag decision %d',ID));
parent=ATTRIBUTES{ID}.parent;
parent
ATTRIBUTES{parent}.serve_neigh
ATTRIBUTES{parent}.RB

A=[];
maxim=-inf;
new_parent=[];
pos=1;
for i=1:length(NEIGHBORS{ID})
    if NEIGHBORS{ID}(i)~=ID &&
ATTRIBUTES{NEIGHBORS{ID}(i)}.RB>buff_threshold &&
NEIGHBORS{ID}(i)~=0 &&
NEIGHBORS{ID}(i)~=ATTRIBUTES{ID}.parent
        A(pos) =
ATTRIBUTES{NEIGHBORS{ID}(i)}.desireability;
        POSSIBLE_NEW_PARENT(pos)=NEIGHBORS{ID}(i);
        pos=pos+1;
    end
end
if ~isempty(A)
    A
    POSSIBLE_NEW_PARENT
    [megisto, index_m]=max(A);

    megisto
    index_m
    if
isempty(find(NEIGHBORS{POSSIBLE_NEW_PARENT(index_m)}== ID))
        A(index_m)=-inf;
        [megisto, index_m]=max(A);

        megisto
        index_m
        if ~isinf(megisto)
            new_parent = POSSIBLE_NEW_PARENT(index_m);
        end
    end

end
```

```
function status = app_layer(N, S)

% Generate an application instance and used for the
movement of the mobile
% nodes

% Written by Ying Zhang, yzhang@parc.com
% Last modified: Feb. 22, 2003  by Antonia Nicolaou

% DO NOT edit simulator code (lines that begin with S;)

S; %%%%%%%%%%%%%%%%%%%   housekeeping
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
S;       persistent app_data
S;       global ID t
S;       [t, event, ID, data]=get_event(S);
S;       [topology, mote_IDs]=prowler('GetTopologyInfo');
S;       ix=find(mote_IDs==ID);
S;       if ~strcmp(event, 'Init_Application')
S;          try memory=app_data{ix}; catch memory=[]; end,
S;       end
S;       global ATTRIBUTES
S;       status = 1;
S;       pass = 1;
S;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

global SOURCES DESTINATIONS TOTAL_SEND

persistent rate
persistent rrate

persistent sourceNofPackets
persistent sourceType
persistent destinationType
persistent sourceSpeed
persistent destinationSpeed
persistent randSourceSpeed
persistent randDestinationSpeed
persistent simInterval
persistent bittime
persistent initTime
persistent traces

global sourceCenterType  type_of_MobileCC
set_parents_of_mob_nodes IDsink NEIGHBORS RNEIGH
global destinationCenterType STOTAL_generated
MOBILE_NODE_RATE STOTAL MOBILE_SERVE_TIME
MOBILE_REACH_POSITION
```

```
global MOBILE_NODES USED_MOBILE_NODES DEST_MOBILE_NODES
NODE_USES_MOBILE_NODES NEIGH_NODES_USES_MOBILE_NODES
INIT_H   = -33;
LOCATE_MOBILE = -88;
LOCATE_ALL_MOBILE=-99;
INIT_H_MOB=-55;


N=N;
if ~strcmp(event, 'Init_Application')

         if (DESTINATIONS(ID)) LED('green on');
         else LED('green off'); end

         if (SOURCES(ID)) LED('red on');
         else LED('red off'); end
end

switch event
case 'Init_Application'

    DEST_MOBILE_NODES=[];

    if (ix==1) %only need to compute it once
        TOTAL_SEND = 0;

        rate = sim_params('get_app', 'SourceRate');
        if (isempty(rate)) rate = 4; end
        rrate = sim_params('get_app', 'RandSourceRate');
        if (isempty(rrate)) rrate = 0; end

        bittime = sim_params('get', 'BIT_TIME');

        sourceNofPackets = sim_params('get_app',
'SourceNofPackets');
        if (isempty(sourceNofPackets)) sourceNofPackets =
Inf; end

        sourceType = sim_params('get_app', 'SourceType');
        if (isempty(sourceType)) sourceType = 'static'; end
        sourceSpeedX = sim_params('get_app',
'SourceSpeedX');

        if (isempty(sourceSpeedX)) sourceSpeedX = 0; end
        sourceSpeedY = sim_params('get_app',
'SourceSpeedY');
        if (isempty(sourceSpeedY)) sourceSpeedY = 0; end
         %disp(sprintf('sourceSpeedX: %f',sourceSpeedX));
         %disp(sprintf('sourceSpeedY: %f',sourceSpeedY));
        sourceSpeed = [sourceSpeedX, sourceSpeedY];
```

```
        destinationType = sim_params('get_app',
'DestinationType');
        if (isempty(destinationType)) destinationType =
'static'; end
        destinationSpeedX = sim_params('get_app',
'DestinationSpeedX');
        if (isempty(destinationSpeedX)) destinationSpeedX =
0; end
        destinationSpeedY = sim_params('get_app',
'DestinationSpeedY');
        if (isempty(destinationSpeedY)) destinationSpeedY =
0; end
        destinationSpeed = [destinationSpeedX,
destinationSpeedY];

        randSourceSpeed = sim_params('get_app',
'RandSourceSpeed');
        if (isempty(randSourceSpeed)) randSourceSpeed = 0;
end
        randDestinationSpeed = sim_params('get_app',
'RandDestinationSpeed');
        if (isempty(randDestinationSpeed))
randDestinationSpeed = 0; end

        sourceCenterType = sim_params('get_app',
'SourceCenterType');
        if (isempty(sourceCenterType)) sourceCenterType =
'random'; end
        destinationCenterType = sim_params('get_app',
'DestinationCenterType');
        if (isempty(destinationCenterType))
destinationCenterType = 'random'; end

%-----------calculate simulation speed instead

        maxSpeedS = 0;
        maxSpeedD = 0;
        if (~strcmp(sourceType, 'static'))
            maxSpeedS = max(abs(sourceSpeed));
        end
        if (~strcmp(destinationType, 'static'))
            maxSpeedD = max(abs(destinationSpeed));
        end
        maxSpeed = max([maxSpeedS, maxSpeedD]);

%-------end of calculation

        maxSpeed
        sourceSpeed = sourceSpeed*5000;
        destinationSpeed = destinationSpeed*5000;
```

```
        randSourceSpeed = randSourceSpeed*5000;
        randDestinationSpeed = randDestinationSpeed*5000;


        initTime = sim_params('get_app', 'InitTime');
        if (isempty(initTime)) initTime = 0; end

        % one may set a trace file from GUI
        useTraceFile = sim_params('get_app',
'UseTraceFile');
        if (isempty(useTraceFile)) useTraceFile = 0; end

        traces = [];
        if (useTraceFile)
            fileName = sim_params('get_app',
'TraceFileName');
            if (~isempty(fileName))
                try
%                     traces = load(fileName);
%                     traces(:, 4) = (traces(:, 4) ./
1000000) .* 31.25; % 1 jiffie = 31.25 * realpow(10, -6)
seconds
%                     traces(:, 4) = traces(:, 4) -
min(traces(:, 4)) + initTime;   % allow initTime seconds
for the application initialization
                    traces = feval(fileName); %traces(:,1)-
> ID, traces(:,2)-> send time (in second)
                    traces(:, 2) = traces(:, 2) -
min(traces(:, 2)) + initTime; % allow initTime seconds for
the application initialization
                catch
                    disp('wrong file name for traces')
                end
            else
                disp('no file name for traces')
            end
        end
        %end of trace file

        Set_SourceDestination(traces);

        normdist = sim_params('get_app',
'PairNorminalDist');
        randdist = sim_params('get_app', 'PairRandDist');
        if (isempty(normdist)) normdist = 0.5; end
        if (isempty(randdist)) randdist = 0.1; end
        check = sim_params('get_app', 'CheckSourceDest');
        if (isempty(check)) check = 1; end
```

```
        if (check && isempty(traces) &&
(strcmp(sourceCenterType, 'random') ||
strcmp(destinationCenterType, 'random')))
            count=0;
            while (~pair_satisfied(normdist, randdist) &&
(count<20))
                Set_SourceDestination(traces);
                count=count+1;
            end
            if (count==20) disp('cannot find source
destination pair'); end
        end

        initTime = initTime/bittime; %bit time
        Set_Sim_Clock(5000);

    end % end of compute once for all

    memory.index=0;

    if (~isempty(traces)) %use trace file
        memory.myTrace = traces(find(traces(:, 1) == ID),
:);
        memory.isSource = ~isempty(memory.myTrace);
        SOURCES(ID) = 0;
        if memory.isSource
            PrintMessage('s');

            SOURCES(ID) = 1;
            memory.totalPackets = length(memory.myTrace);
            memory.packetPtr = 1;
            %added by guoliang, we need rate information
even when trace
            %file is used. The rate informatio will be
encoded in every
            %packet sent by the application.
            memory.rate = ...

size(memory.myTrace,1)/(max(memory.myTrace(:,2))-
min(memory.myTrace(:,2)));
            Set_App_Clock(memory.myTrace(memory.packetPtr,
2)/bittime); % schedule the first packet to be sent
        end
    else %use generated traces
        memory.totalPackets = sourceNofPackets;
        if (SOURCES(ID))
            PrintMessage('s');
            memory.rate = 1/bittime/(rate*(1+rrate*rand));
%variations between source nodes
            Set_App_Clock(memory.rate*rand+initTime);
```

```
        end
    end

case 'Send_Packet'

    try msgID = data.msgID; catch msgID=0; end


    if msgID== LOCATE_MOBILE
         a=data.data;

                %round to one decimal the postition of the
mobile node
                disp('infoo');
                a{1} %node needs mobile
                a{2} %p
                a{3} %neighbor of node that need mobile
node that will serve
                a{4} %the id of mobile nnode
                if ~isempty(a{4}) && ~isempty(a{4})
                i=a{4};
                disp('mob node');
                i

               DEST_MOBILE_NODES{i}=a{2};
               NODE_USES_MOBILE_NODES{i}=a{1};
               disp('node use ok');
               NODE_USES_MOBILE_NODES{i}
               NEIGH_NODES_USES_MOBILE_NODES{i} = a{3};
               NEIGH_NODES_USES_MOBILE_NODES{i}
               MOBILE_NODE_RATE(i)=0;


               summob=0;
               for
j=1:length(NEIGH_NODES_USES_MOBILE_NODES{i})

disp('NEIGH_NODES_USES_MOBILE_NODES{i}(j)');
                  NEIGH_NODES_USES_MOBILE_NODES{i}(j)
                  if
NEIGH_NODES_USES_MOBILE_NODES{i}(j)>length(ATTRIBUTES)

NEIGH_NODES_USES_MOBILE_NODES{i}(j)=i-1;
                      disp(sprintf('change mob node'));
                      NEIGH_NODES_USES_MOBILE_NODES{i}(j)
                  else

ATTRIBUTES{NEIGH_NODES_USES_MOBILE_NODES{i}(j)}.serve_neigh
=1;
```

```
                    disp(sprintf('serveneigh %d
',NEIGH_NODES_USES_MOBILE_NODES{i}(j)));
                    disp(sprintf('serveneigh %d
STOTAL_generated %f STOTAL %f
',NEIGH_NODES_USES_MOBILE_NODES{i}(j),STOTAL_generated(NEIG
H_NODES_USES_MOBILE_NODES{i}(j)),STOTAL{NEIGH_NODES_USES_MO
BILE_NODES{i}(j)}));

                     if MOBILE_NODES(i)==1

summob=summob+(STOTAL_generated(NEIGH_NODES_USES_MOBILE_NOD
ES{i}(j))+STOTAL{NEIGH_NODES_USES_MOBILE_NODES{i}(j)})/ (
(t-MOBILE_SERVE_TIME(NODE_USES_MOBILE_NODES{i}(1)))/40000
);
                        summob
                     else

summob=summob+(STOTAL_generated(NEIGH_NODES_USES_MOBILE_NOD
ES{i}(j))+STOTAL{NEIGH_NODES_USES_MOBILE_NODES{i}(j)})/ (
(t-start_send_data)/40000 );
                        summob
                     end
                    end
                end
                MOBILE_NODE_RATE(i)=summob;

                if type_of_MobileCC<3 %if is not the global
algorithm locate the nodes
                     movment_of_mobile_node(i,N);

                end %if type_of_MobileCC~=3

                 end

    end
         if msgID== LOCATE_ALL_MOBILE
             disp('LOCATE_ALL_MOBILE');
                      for i=1:length(USED_MOBILE_NODES)
                          if USED_MOBILE_NODES(i)==1
disp('USED_MOBILE_NODES');
i
                                movment_of_mobile_node(i,N);

                           end
                 end
        end

case 'Packet_Received'
    try msgID = data.data.msgID; catch msgID = 0; end
```

```
    try duplicated = data.duplicated; catch duplicated = 0;
end
    if (~DESTINATIONS(ID) | msgID < 0 | duplicated)
        pass = 0;
    else %received data
        if ~isempty(data.data)

PrintMessage([num2str(data.data.value),',',num2str(data.dat
a.source)]);
        end
    end

case 'Clock_Tick'

        try type = data.type; catch type = 'none'; end
        mob_node_id=[];
        if (strcmpi(type, 'loc_mob_node'))
            disp(sprintf('Buffer  %f',t));


            for i=1:length(MOBILE_REACH_POSITION)
                if MOBILE_REACH_POSITION(i)==t
                    mob_node_id=i;
                    break;
                end
            end
            mob_node_id
            if ~isempty(mob_node_id)
            ATTRIBUTES{mob_node_id}.x
            ATTRIBUTES{mob_node_id}.y
            DEST_MOBILE_NODES{mob_node_id}
            difx=ATTRIBUTES{mob_node_id}.x-
DEST_MOBILE_NODES{mob_node_id}(1);
            e=0.0001;
            if abs(difx) > e

                    sourceSpeed=[-difx, 0];

                set_mobile_motes(mob_node_id, sourceSpeed);

            end
            dify=ATTRIBUTES{mob_node_id}.y-
DEST_MOBILE_NODES{mob_node_id}(2);
                if abs(dify)> e

                        sourceSpeed=[0, -dify];
                    %disp(sprintf('ATTRIBUTES{sID}.y:
%f y: %f sourceSpeed y:',ATTRIBUTES{sID}.y,y_COR));
                    sourceSpeed
```

```
                        set_mobile_motes(mob_node_id,
sourceSpeed);
                end
                disp('--mob_node--');
                move_node(mob_node_id);
                          disp('---send hello msg');
                        hello.msgID = INIT_H_MOB;
                        hello.from = i;
                        %mobile.init = ID;
                        hello.address=0;
                        status = Dalpas_layer(N,
make_event(t, 'Send_Packet', i, hello));
                        %status =init_span_layer(N,
make_event(t, 'Send_Packet', i, hello));
                  disp('---send hello msg');
            end
        end
    if (strcmp(data.type, 'app_send'))
        if (~isempty(traces)) %use trace file
            if (memory.isSource)
                SendData(memory.index, memory.rate);
%myTrace(:, 3): sequence number
                memory.index = memory.index+1;
                memory.packetPtr = memory.packetPtr + 1;
                if memory.packetPtr <= memory.totalPackets

Set_App_Clock(memory.myTrace(memory.packetPtr, 2)/bittime);
% schedule the next packet to be sent, if any
                end
            end
        else %use generated traces
            if (SOURCES(ID))
                memory.totalPackets = memory.totalPackets -
1;
                try memory.rate; catch memory.rate =
1/bittime/(rate*(1+rrate*rand)); end
                if (memory.totalPackets>0)
                    Set_App_Clock(t+memory.rate);
                end
                SendData(memory.index,
1/(bittime*memory.rate));
                memory.index = memory.index+1;
            end
        end
    end
    if (strcmp(data.type, 'app_sim'))
        Set_Sim_Clock(t+5000);

        if (t>initTime) %start simulate motion after
initTime
```

```
            if (strcmp(destinationType, 'mobile'))
                dID = find(DESTINATIONS==1);
                set_mobile_motes(dID, 2*(rand(1,2)-
0.5).*randDestinationSpeed+destinationSpeed);
            elseif (strcmp(destinationType, 'dynamic'))
                destinations = DESTINATIONS;
                DESTINATIONS =
Set_Destination(2*(rand(1,2)-
0.5).*randDestinationSpeed+destinationSpeed);
            end
        end
    end
end


S; %%%%%%%%%%%%%%%%%%%%%% housekeeping
%%%%%%%%%%%%%%%%%%%%%%%%%%%
S;        try app_data{ix}=memory; catch app_data{ix} = [];
end
S;        if (pass) status = common_layer(N, make_event(t,
event, ID, data)); end
S;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

function b=Set_App_Clock(alarm_time);
global ID
clock.type = 'app_send';
prowler('InsertEvents2Q', make_event(alarm_time,
'Clock_Tick', ID, clock));

%set simulation clock

function b=Set_Sim_Clock(alarm_time);
global ID
clock.type = 'app_sim';
prowler('InsertEvents2Q', make_event(alarm_time,
'Clock_Tick', ID, clock));

%send data out

function status = SendData(varargin);
global ID t TOTAL_SEND
sdata.forward = 0;
sdata.value = varargin{1};
sdata.source = ID;
sdata.msgID = 0;
if (length(varargin)>2)
    sdata.seqID = varargin{3};
```

```
else
    sdata.seqID = varargin{1};
    if (length(varargin)>1)
        sdata.rate = varargin{2};
    end
end
%if (length(varargin)<2) sdata.maxhops = Inf; else
sdata.maxhops = varargin{2}; end
sdata.startTime = t;
PrintMessage(num2str(sdata.value));
N = find_layer('app');
status = app_layer(N, make_event(t, 'Send_Packet', ID,
sdata));
TOTAL_SEND = TOTAL_SEND + 1;

%initial source destination pair

function Set_SourceDestination(traces)

global SOURCES DESTINATIONS

if (isempty(traces))
    disp('innn empty');
    SOURCES = Set_Source;
    N=0;

    SOURCES
    while (~sum(SOURCES) && (N<10))
        prowler('PrintEvent', 'finding a source...');
        SOURCES = Set_Source;
        N=N+1;
    end
    if (N==10)
        disp('cannot find sources');
    end
end

DESTINATIONS = Set_Destination;
N=0;
while (~sum(DESTINATIONS) && (N<10))
    prowler('PrintEvent', 'finding a destination...');
    DESTINATIONS = Set_Destination;
    N=N+1;
end
if (N==10)
    disp('cannot find destinations');
end

%set source
```

```
function out = Set_Source(varargin);

persistent x y r p
global sourceCenterType

if (nargin==0)
disp('innn source');
    if (strcmp(sourceCenterType, 'random'))
        disp('innn rand');
        [minx, maxx, miny, maxy]=network_size;
        x = rand*(maxx-minx)+minx;
        y = rand*(maxy-miny)+miny;
    else
        disp('innn else');
        x = sim_params('get_app', 'SourceCenterX');
        y = sim_params('get_app', 'SourceCenterY');
        if isempty(x)
            disp('empty x');
            x=0;
        end
        if isempty(y)
             disp('empty y');
            y=0;
        end
    end

    r = sim_params('get_app', 'SourceRadius');
    if isempty(r); disp('empty r'); r=0.5; end

    p = sim_params('get_app', 'SourcePercentage');
    if isempty(p);  disp('empty p'); p=1; end
    disp(sprintf(' x: %f y: %f  r %f p %f ', x,y,r,p));

else
    new_s = [x, y] + varargin{1};
    x = new_s(1);
    y = new_s(2);

end

out = set_static_motes([x,y], r, p);

unique = sim_params('get_app', 'SourceUnique');
if (isempty(unique)) unique = 1;disp('empty p'); end
 disp(sprintf(' unique: %d',unique));
if (unique) out = Set_Unique(out); end

%set destination

function out = Set_Destination(varargin);
```

```
persistent x y r p
global destinationCenterType

if (nargin==0)

    if (strcmp(destinationCenterType, 'random'))
        [minx, maxx, miny, maxy]=network_size;
        x = rand*(maxx-minx)+minx;
        y = rand*(maxy-miny)+miny;
    else
        x = sim_params('get_app', 'DestinationCenterX');
        y = sim_params('get_app', 'DestinationCenterY');
        if isempty(x)
            x=0;
        end
        if isempty(y)
            y=0;
        end
    end

    r = sim_params('get_app', 'DestinationRadius');
    if isempty(r); r=0.5; end

    p = sim_params('get_app', 'DestinationPercentage');
    if isempty(p); p=1; end

else
    new_d = [x, y] + varargin{1};
    x = new_d(1);
    y = new_d(2);

end

out = set_static_motes([x,y], r, p);

unique = sim_params('get_app', 'DestinationUnique');
if (isempty(unique)) unique = 1; end
if (unique) out = Set_Unique(out); end

%pick one mote with the least ID

function out = Set_Unique(in);
disp('uni');
idx = find(in==1);

out = zeros(size(in));
if (~isempty(idx)) out(idx(1)) = 1; end

%find bounding box of topology
```

```
function [minx, maxx, miny, maxy]=network_size

topology = prowler('GetTopologyInfo');

minx=min(topology(:,1));
maxx=max(topology(:,1));
miny=min(topology(:,2));
maxy=max(topology(:,2));
%called when the mobile node is locates
%determine the MOBILE_SERVE_TIME of sid
%set the parent of nodes in
NEIGH_NODES_USES_MOBILE_NODES{sid} to sid
function move_node(sID)
global ATTRIBUTES RNEIGH NEIGHBORS IDsink STOTAL
STOTAL_generated xsink ysink USED_MOBILE_NODES
DEST_MOBILE_NODES MOBILE_NODES NODE_USES_MOBILE_NODES
set_parents_of_mob_nodes type_of_MobileCC
NEIGH_NODES_USES_MOBILE_NODES t MOBILE_SERVE_TIME;
r=2.5;
sink=[xsink ysink];
if(type_of_MobileCC==3)
    %set the parent of the mobile node - through the
optimal path that was
    %created using the dynamic algorithm
    disp('set parent');
    minid=set_parents_of_mob_nodes(sID);
    disp('first time');
minid
end

    if (type_of_MobileCC<3) || (minid==0) ||
(isempty(minid))
        minid=IDsink;
        mindist=inf;
        X=DEST_MOBILE_NODES{sID};
        for i=1:length(ATTRIBUTES)
            if (i~=sID && (MOBILE_NODES(i)==0 ||
USED_MOBILE_NODES(i)==1) &&
isempty(find(NEIGH_NODES_USES_MOBILE_NODES{sID}==i)) &&
isempty(find(NODE_USES_MOBILE_NODES{sID}==i)))
                Z=[ATTRIBUTES{i}.x, ATTRIBUTES{i}.y];
                d=pdist2(X,Z,'euclidean');
                dsink=pdist2(sink,Z,'euclidean');
            if d<=r && dsink<mindist &&
~isinf(ATTRIBUTES{i}.desireability)
                minid=i;
                mindist=dsink;
            end
            end
        end
```

```
    end

    disp('set parent');
    minid
    ATTRIBUTES{sID}.parent=minid;

        ATTRIBUTES{sID}.desireability = 1;
        ATTRIBUTES{sID}.hard_stage = 0;
        NEIGHBORS{sID}=NEIGH_NODES_USES_MOBILE_NODES{sID};
        if ~isempty(minid) && minid~=0
         NEIGHBORS{sID}=union( NEIGHBORS{sID},minid);
         ATTRIBUTES{sID}.hops=ATTRIBUTES{minid}.hops+1;
        end
     for i=1:length(ATTRIBUTES)
        RNEIGH{sID}(i)=[0];
        %in each position has the number of msgs received
from that node,
        %for all nodes in the network
    end
    if isempty(minid) && minid~=0 && type_of_MobileCC~=2
        ATTRIBUTES{sID}.needs_mobile = 0;
    else
        ATTRIBUTES{sID}.needs_mobile = 1;
    end
    ATTRIBUTES{sID}.serve = 0;
     ATTRIBUTES{sID}.serve_neigh=0;
     memory.queue = {};
        ATTRIBUTES{sID}.RB = 1;
        ATTRIBUTES{sID}.PACKETDROPS = 0;
     STOTAL{sID} = [0];
    STOTAL_generated(sID) = [0];


  disp('NODE_USES_MOBILE_NODES{sID}');
  NODE_USES_MOBILE_NODES{sID}
     for i=1:length(NODE_USES_MOBILE_NODES{sID})
         if
NODE_USES_MOBILE_NODES{sID}(i)<length(ATTRIBUTES)

ATTRIBUTES{NODE_USES_MOBILE_NODES{sID}(i)}.serve=0;
             if
MOBILE_SERVE_TIME(NODE_USES_MOBILE_NODES{sID}(i))==0

MOBILE_SERVE_TIME(NODE_USES_MOBILE_NODES{sID}(i))=t;
                 disp('MOBILE_SERVE_TIME')

MOBILE_SERVE_TIME(NODE_USES_MOBILE_NODES{sID}(i))
             end
             disp(sprintf('serve no more
%d',NODE_USES_MOBILE_NODES{sID}(i)));
```

```
        end
     end
     disp(' NEIGH_NODES_USES_MOBILE_NODES{sID(i)}');
    NEIGH_NODES_USES_MOBILE_NODES{sID}
    for i=1:length(NEIGH_NODES_USES_MOBILE_NODES{sID})
        ATTRIBUTES{NEIGH_NODES_USES_MOBILE_NODES{sID}(i)}

ATTRIBUTES{NEIGH_NODES_USES_MOBILE_NODES{sID}(i)}.parent
=sID;

NEIGHBORS{NEIGH_NODES_USES_MOBILE_NODES{sID}(i)}=union(sID,
NEIGHBORS{NEIGH_NODES_USES_MOBILE_NODES{sID}(i)});
    end



function movment_of_mobile_node(i, N)
global ATTRIBUTES INIT_H_MOB ID NODE_USES_MOBILE_NODES
MOBILE_REACH_POSITION IDsink NEIGH_NODES_USES_MOBILE_NODES
t MOBILE_SERVE_TIME DEST_MOBILE_NODES

                disp(sprintf('locate i %d for',i));
                disp(sprintf('timeout %f',t'));
                ATTRIBUTES{i}.sleep = 0;
                %ATTRIBUTES{i}.parent = IDsink;
                IDsink

                v=35;
                moblocation=[ATTRIBUTES{i}.x,
ATTRIBUTES{i}.y];
                moblocation
                DEST_MOBILE_NODES{i}
                dx=pdist2(moblocation,
DEST_MOBILE_NODES{i},'euclidean');
                dx
                mobile_timeout=dx/v;
                tout=(mobile_timeout*40000)+t; %conversion
to seconds
                MOBILE_REACH_POSITION(i)=[tout];
                Set_Timeout_Clock_Mobile_locate(tout,
'loc_mob_node');

function b=Set_Timeout_Clock_Mobile_locate(timeout, type)
global t ID
    clock.type = type;
    disp(sprintf('time out %f',t));
    prowler('InsertEvents2Q', make_event(timeout,
'Clock_Tick', ID, clock));
```