Πανεπιστήμιο Πατρών

Τμήμα Μηχανικών Υπολογιστών και Πληροφορικής

Διπλωματική Εργασία

Υλοποίηση και Πειραματική Αξιολόγηση Αλγορίθμων Δρομολόγησης σε Ad-hoc Κινητά Δίκτυα

Νέαρχος Πασπαλλής

-Πάτρα 2001-

Διπλωματική Εργασία στο Τμήμα Μηχανικών

Υπολογιστών και Πληροφορικής

Υλοποίηση και Πειραματική Αξιολόγηση Αλγορίθμων Δρομολόγησης σε Ad-hoc Κινητά Δίκτυα

Πάτρα, 2001

Νέαρχος Πασπαλλής

Πανεπιστήμιο Πατρών

Επιβλέποντες: Χρήστος Ζαρολιάγκης
Επίκουρος Καθηγητής
Πανεπιστήμιο Πατρών

Παύλος Σπυράκης
Καθηγητής
Πανεπιστήμιο Πατρών

Ευχαριστίες

Αυτή η εργασία θα ήταν αδύνατο να υλοποιηθεί χωρίς την συνεργασία, τη βοήθεια, και το ενδιαφέρον αρκετών ανθρώπων. Με αυτή την ευκαιρία, θα ήθελα να ευχαριστήσω τους σημαντικότερους από αυτούς.

Αρχικά, θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Χρήστο Ζαρολιάγκη, που ήταν ο υπεύθυνος Καθηγητής για την διπλωματική μου εργασία. Χάρις στην εποικοδομητική του επίβλεψη και τις πολύτιμες συμβουλές που μου έδωσε στους τελευταίους δώδεκα μήνες, η προσπάθεια μου για την εκπόνηση μιας επιτυχημένης διπλωματικής εργασίας έγινε πιο εφικτή και πιο ενδιαφέρουσα.

Ακολούθως, θα πρέπει να ευχαριστήσω τον συν-επιβλέποντα, υποψήφιο Διδάκτωρ κ. Ιωάννη Χατζηγιαννάκη. Η διαρκής βοήθεια που μου προσέφερε και οι χρήσιμες ιδέες που ανταλλάξαμε ήταν καθοριστικές για την έρευνα που προηγήθηκε, καθώς και για την τελική μορφή της διπλωματικής μου εργασίας.

Τέλος, θα πρέπει να ευχαριστήσω τον Καθηγητή κ. Παύλο Σπυράκη, τον Διδάκτορα κ. Σωτήρη Νικολετσέα, καθώς και την υποψήφια Διδάκτορα, κ. Βίκυ Παπαδοπούλου, για τις συμβουλές, και τις γνώσεις που μου έδωσαν κατά τις συναντήσεις και τις συζητήσεις που είχαμε στα πλαίσια αυτής της διπλωματικής εργασίας.

Περίληψη

Τα τελευταία χρόνια η χρήση ad-hoc κινητών δικτύων προσελκύει ολοένα και περισσότερο τη προσοχή της διεθνούς ακαδημαϊκής και επιστημονικής κοινότητας. Η αυξανόμενη χρήση προσωπικών κινητών υπολογιστών (όπως είναι οι φορητοί υπολογιστές και τα PDAs) σε συνδυασμό με την ανάγκη για γρήγορη και διαρκή επικοινωνία καθιστούν πολύ σημαντική την ύπαρξη αποδοτικών πρωτοκόλλων επικοινωνίας.

Σε ένα ad-hoc κινητό δίκτυο, η επικοινωνία μεταξύ δύο οποιονδήποτε χρηστών προϋποθέτει τον σχηματισμό ενός προσωρινού δικτύου χωρίς τη βοήθεια κάποιου προϋπάρχοντος κεντρικοποιημένου συστήματος. Κάθε κόμβος που συμμετέχει στο δίκτυο θα πρέπει να μπορεί να ενεργεί και σαν host και σαν router. Άρα, όταν οι χρήστες του δικτύου λαμβάνουν πακέτα, θα πρέπει να είναι πρόθυμοι να τα προωθήσουν. Ο σκοπός ενός πρωτοκόλλου δρομολόγησης είναι ο συντονισμός των χρηστών και ο καθορισμός των απαιτούμενων ενεργειών για την μετάδοση πακέτων προς κάθε επιθυμητό στόχο.

Οι ιδιαιτερότητες των ad-hoc κινητών δικτύων, που οφείλονται κυρίως στην υψηλή κινητικότητα των χρηστών, εισάγουν συγκεκριμένες προϋποθέσεις και παραμέτρους που θα πρέπει να λαμβάνουν υπόψη οι σχεδιαστές των ζητούμενων πρωτοκόλλων. Στους περιορισμούς αυτών των δικτύων περιλαμβάνονται η δυναμική φύση των συνδέσεων, οι περιορισμένοι πόροι (επεξεργαστική ισχύς, μνήμη, ενέργεια - ικανότητα αυτόνομης λειτουργίας των σταθμών), καθώς και το περιορισμένο εύρος ζώνης επικοινωνίας (bandwidth).

Αυτή η εργασία, εκτός από την παρουσίαση και περιγραφή των υπαρχόντων πρωτοκόλλων, συμβάλλει και σε δύο άλλους τομείς. Αρχικά, εισάγουμε ένα νέο Περιβάλλον Προσομοίωσης που έχει σαν στόχο την εύκολη και γρήγορη προσομοίωση λειτουργίας πρωτοκόλλων για Ad-hoc Κινητά Δίκτυα. Επιπλέον, παρουσιάζουμε δύο καινούρια πρωτόκολλα επικοινωνίας, τα οποία και μελετούμε εκτενώς.

Πίνακας Περιεχομένων

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1. Εισαγωγή

1.1. Στόχοι της Εργασίας

Η διπλωματική αυτή εργασία στοχεύει στην περιγραφή των ασύρματων κινητών δικτύων, καθώς και στην παρουσίαση των υπαρχόντων πρωτοκόλλων δρομολόγησης. Επιπλέον, η εργασία αυτή παρουσιάζει δύο νέα πρωτόκολλα που δημοσιεύονται στην [1], μελετώντας τα σε θεωρητικό και πειραματικό επίπεδο. Ακόμη, η εργασία προσπαθεί να επαληθεύσει την ορθή λειτουργία ενός γενικού περιβάλλοντος προσομοίωσης, που δημιουργήθηκε για την πειραματική αξιολόγηση των πρωτοκόλλων αυτών. Το περιβάλλον αυτό έχει υλοποιηθεί με χρήση της γλώσσας προγραμματισμού C++, και κάνει εκτενή χρήση της βιβλιοθήκης αλγορίθμων και δομών δεδομένων LEDA[1].

1.1.1 Σημασία του Υπολογισμού σε Συστήματα με Κινητά Μέρη (Mobile Computing)

Παρόλο που τα ασύρματα δίκτυα έχουν κάνει την εμφάνιση τους εδώ και δεκαετίες, τα δίκτυα αυτά έχουν λάβει της προσοχής της διεθνούς επιστημονικής κοινότητας μόλις τα τελευταία χρόνια. Αυτό οφείλεται κυρίως στην πρόοδο που έχει παρατηρηθεί στη τεχνολογία των ασύρματων συσκευών, καθώς και στην πρόοδο των εφαρμογών τους. Τέτοιες συσκευές είναι κυρίως οι φορητοί υπολογιστές, οι υπολογιστές παλάμης, και τα κινητά τηλέφωνα. Τα τελευταία χρόνια οι συσκευές αυτές έχουν γίνει αρκετά προσιτές αφού έχει μειωθεί το βάρος τους, αυξάνοντας έτσι την μεταφερσιμότητα τους. Επιπλέον, έχει παρατηρηθεί σημαντική πρόοδος στο χώρο των μπαταριών, γεγονός που επιτρέπει στις συσκευές αυτές να μπορούν να λειτουργούν για μεγαλύτερο χρονικό διάστημα χωρίς να χρειάζεται να συνδεθούν στο ηλεκτρικό δίκτυο. Τέλος, λόγω της προόδου που παρατηρείται στο χώρο των ηλεκτρονικών, το κόστος αυτών των συσκευών έχει μειωθεί αρκετά καθιστώντας την απόκτηση τους αρκετά προσιτή.

Οι ασύρματες αυτές συσκευές μπορούν να δικτυωθούν μεταξύ τους είτε μέσω κατάλληλων θυρών που χρησιμοποιούν υπέρυθρες ακτίνες, είτε με χρήση κατάλληλων ασύρματων modems[2] που επικοινωνούν με ραδιοκύματα. Οι εφαρμογές των πρώτων είναι περιορισμένες λόγω της ανάγκης για οπτική επαφή μεταξύ των επικοινωνούντων συσκευών, καθώς και λόγω του μικρού ρυθμού μετάδοσης δεδομένων που επιτρέπουν οι υπέρυθρες ακτίνες. Αντίθετα, οι συσκευές που χρησιμοποιούν ραδιοκύματα έχουν την ικανότητα να

[1] Η LEDA [11] (Library of Efficient Data structures and Algorithms) είναι μια βιβλιοθήκη αλγορίθμων και δομών δεδομένων που έχει υλοποιηθεί σε C++. Η LEDA κατασκευάστηκε από την Algorithmic Solutions, της οποίας ο δικτυακός τόπος είναι: http://www.algorithmic-solutions.com/as_html/products/leda/products_leda.html .

[2] Τα modems είναι συσκευές που χρησιμοποιούνται αφενός για την μετατροπή της πληροφορίας σε μορφή τέτοια που να μπορεί να μεταδοθεί μέσω κάποιου κατάλληλου μέσου (π.χ. καλώδιο η αέρας), και αφετέρου για την αντίστροφη μετατροπή στην αρχική πληροφορία.

επικοινωνούν σε μεγαλύτερες αποστάσεις και με ταχύτερους ρυθμούς. Τα ραδιοκύματα πάντως μειονεκτούν στο ότι πάσχουν από προβλήματα παρεμβολών και εξασθένισης του σήματος. Σήμερα, υπάρχουν ασύρματες συσκευές που υποστηρίζουν πολλούς συνδυασμούς μεταξύ ισχύος μετάδοσης και ακτίνας επικοινωνίας. Επιπλέον, υπάρχουν συσκευές που μπορούν να επικοινωνήσουν σε ρυθμούς μέχρι και 11 Mbps σε αποστάσεις μέχρι και 600m, ανάλογα και με τις συνθήκες του περιβάλλοντος επικοινωνίας.

Τα ασύρματα κινητά δίκτυα έχουν πολλά ιδιαίτερα χαρακτηριστικά που καθιστούν τα παραδοσιακά πρωτόκολλα δρομολόγησης ακατάλληλα για αυτά. Η τοπολογία του δικτύου αλλάζει πολύ συχνά, λόγω της κινητής φύσης των σταθμών. Ενώ στα ενσύρματα δίκτυα, μια διακεκομμένη σύνδεση αποτελεί την εξαίρεση και δεν συμβαίνει συχνά, στο ασύρματα δίκτυα οι συνδέσεις διακόπτονται πολύ συχνά, κυρίως λόγω της μετακίνησης των κόμβων του δικτύου. Επιπλέον, οι ατμοσφαιρικές συνθήκες (όπως για παράδειγμα η βροχή), καθώς και τα φυσικά εμπόδια, επιδρούν στα κινητά δίκτυα περιορίζοντας την εμβέλεια των επικοινωνούντων κόμβων. Τέλος, στα χαρακτηριστικά των ασύρματων κινητών δικτύων περιλαμβάνονται η περιορισμένη ενέργεια, το περιορισμένο εύρος ζώνης επικοινωνίας που μπορούν να χρησιμοποιούν οι συσκευές αυτές, και τέλος ο ψηλός ρυθμός λαθών στην μετάδοση λόγω της ασύρματης φύσης της επικοινωνίας.

Με βάση τα πιο πάνω μπορούμε να δώσουμε ένα πρώτο ορισμό για τα ad-hoc δίκτυα ως εξής: *“Ένα ad-hoc δίκτυο είναι μια συλλογή από κινητούς κόμβους οι οποίοι διαθέτουν συσκευές ασύρματης επικοινωνίας και μπορούν να σχηματίσουν ένα προσωρινό δίκτυο χωρίς τη χρήση οποιασδήποτε σταθερής υποδομής ή κεντρικής διοίκησης”.*

Γενικά, τα πρωτόκολλα δρομολόγησης που έχουν σχεδιαστεί για ενσύρματα δίκτυα δεν αποδίδουν καλά στα ασύρματα δίκτυα. Παραδοσιακά, τα πρωτόκολλα δρομολόγησης των ενσύρματων δικτύων βασίζονται στα πρωτόκολλα distance-vector [12], και link-state [13]. Αυτά τα πρωτόκολλα διατηρούν διαδρομές από κάθε κόμβο προς κάθε άλλο κόμβο του δικτύου με τη περιοδική μετάδοση μηνυμάτων που περιέχουν τους πίνακες δρομολόγησης. Τα πρώτα πρωτόκολλα δρομολόγησης που σχεδιάστηκαν για ad-hoc δίκτυα προσπάθησαν να προσαρμόσουν τα βασικά αυτά πρωτόκολλα στις ανάγκες των κινητών δικτύων. Παρόλα αυτά, ήταν ξεκάθαρο ότι σε πολλά κινητά δίκτυα οι συνδέσεις διακόπτονται πολύ συχνά για να μπορούμε να διατηρούμε πληροφορίες για την τοπολογία ολόκληρου του δικτύου. Η προσθήκη τακτικών ενημερώσεων έτσι ώστε να κατανέμουμε ακριβείς πληροφορίες δρομολόγησης στο δίκτυο οδηγεί συχνά σε ανεπιθύμητα αποτελέσματα. Πιο συγκεκριμένα, έχουμε αυξημένη χρήση του εύρους επικοινωνίας του δικτύου, κάτι που κάνει την σύγκλιση των διαδρομών πολύ δύσκολη, αν όχι αδύνατη. Κάποιες άλλες προσεγγίσεις στο πρόβλημα της δρομολόγησης των ad-hoc δικτύων κάνουν επιπρόσθετες τροποποιήσεις στους αλγορίθμους distance-vector και

link-state, έτσι ώστε να τους κάνουν καταλληλότερους για κινητά δίκτυα. Αυτές οι προσεγγίσεις αφορούν κυρίως τη χρήση προτεραιοτήτων στη ανταλλαγή πληροφοριών έτσι ώστε πληροφορίες που αφορούν απομακρυσμένους κόμβους να μεταδίδονται λιγότερο συχνά από ότι πληροφορίες που αφορούν γειτονικούς κόμβους. Αυτά τα πρωτόκολλα έχουν το πλεονέκτημα του μειωμένου πρόσθετου κόστους (overhead[3]) λόγω της περιορισμένης ανταλλαγής πληροφοριών δρομολόγησης. Όμως, εξακολουθούν να έχουν το μειονέκτημα της ανάγκης ανταλλαγής δεδομένων ανάλογα με την κίνηση των σταθμών, κάτι που σε ένα κινητό δίκτυο μπορεί να οδηγήσει σε σημαντικό πρόσθετο κόστος και κατανάλωση του εύρους επικοινωνίας του δικτύου.

Για παρόμοιους λόγους, τα πρωτόκολλα multicast που σχεδιάστηκαν για ενσύρματα δίκτυα επικοινωνίας δεν είναι κατάλληλα για λειτουργία ανάμεσα σε κινητά δίκτυα. Για παράδειγμα, η χρήση δέντρων έχει το μειονέκτημα ότι όλες οι διαδρομές περνάνε μέσα από ένα μοναδικό κόμβο, τη ρίζα. Σε ένα κινητό δίκτυο όμως, παρατηρείται συχνά το φαινόμενο ένας κινητός κόμβος να “εξαφανίζεται” είτε λόγω φυσικών εμποδίων, είτε λόγω κάποιας μετακίνησης. Έτσι, αλγόριθμοι που μπορούν να αποτύχουν με την αποσύνδεση ενός και μόνο κόμβου είναι πολύ πιθανό να υποφέρουν από τις συχνές προσωρινές αποσυνδέσεις που παρατηρούνται στα κινητά δίκτυα. Επιπλέον, οι κλασσικοί αλγόριθμοι που χρησιμοποιούνται στα ενσύρματα δίκτυα για multicasting, συχνά απαγορεύουν τη μετάδοση ενός πακέτου από κάποια διαδρομή διαφορετική από την προκαθορισμένη. Αυτό όμως δεν είναι καθόλου αποδοτικό στα κινητά δίκτυα αφού πολύ συχνά τα πακέτα θα πρέπει να αλλάζουν δυναμικά τη διαδρομή τους λόγω των μετακινήσεων των σταθμών. Στην πραγματικότητα, λόγω της αυξημένης δυσκολίας δρομολόγησης στα ad-hoc δίκτυα, οι κόμβοι που συμμετέχουν στο multicast θα πρέπει να δέχονται τα πακέτα δεδομένων που προορίζονται για αυτούς από οποιαδήποτε κατεύθυνση και αν τα λαμβάνουν.

Τα κινητά ασύρματα δίκτυα έχουν όμως και μερικά πλεονεκτήματα σε σύγκριση με τα παραδοσιακά ενσύρματα δίκτυα. Τα ασύρματα δίκτυα μπορούν να αναπτυχθούν σε περιοχές όπου δεν υπάρχει εγκατεστημένη υποδομή δικτύωσης. Επιπλέον, τα ασύρματα συστήματα επιτρέπουν την άμεση ενεργοποίηση του δικτύου. Για παράδειγμα, μια ομάδα ανθρώπων που παρακολουθούν ένα συνέδριο μπορούν ανά πάσα στιγμή να ενεργοποιήσουν τους φορητούς τους υπολογιστές, και αμέσως να σχηματίσουν ένα δίκτυο που να τους επιτρέπει να μοιράζονται αρχεία μεταξύ τους ή να ανταλλάσσουν ηλεκτρονικά μηνύματα. Τα ασύρματα δίκτυα επιτρέπουν σύνδεση οπουδήποτε και οποτεδήποτε. Οι ίδιοι άνθρωποι που παρακολουθούν το συνέδριο μπορούν να καθίσουν στην αίθουσα αναμονής και να συνδεθούν στο Internet, αν η αίθουσα

[3] Σε αυτή την εργασία, με τον όρο *overhead* θα εννοούμε το κόστος (σε μηνύματα ή εύρος ζώνης επικοινωνίας) που εισάγεται από το δίκτυο για την εξυπηρέτηση της πραγματικής μεταδιδόμενης πληροφορίας. Ο όρος αυτός θα χρησιμοποιείται ισοδύναμα με τον Ελληνικό όρο *πρόσθετο κόστος*.

αυτή διαθέτει τη κατάλληλη υποδομή (wireless access points[4]). Παρόμοια, ένας επιχειρηματίας από μια εταιρεία μπορεί να ελέγξει τα e-mail του ενώ βρίσκεται σε μια άλλη εταιρεία, χωρίς όμως να μπορεί να προσπελάσει τα αρχεία αυτής της εταιρεία (προφανώς για λόγους ασφάλειας). Επιπλέον, η ασύρματη φύση αυτών των συσκευών επιτρέπει στους χρήστες τους να μπορούν να μετακινούνται ενώ τις χρησιμοποιούν. Τέλος, τα ασύρματα δίκτυα μπορούν να μειώσουν σημαντικά το "μπέρδεμα" που παρατηρείται με τα πολλά καλώδια στους χώρους των γραφείων, εξαλείφοντας την ανάγκη για καλώδια που να συνδέουν τους εκτυπωτές, τα πληκτρολόγια το δίκτυο, κλπ.

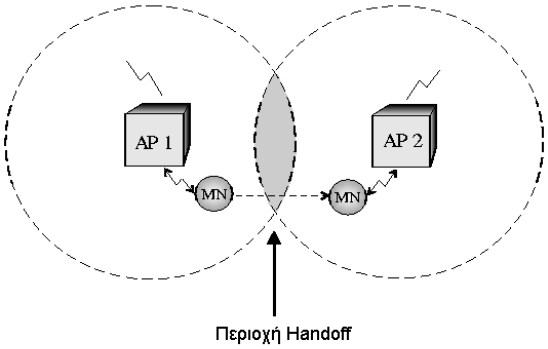
1.1.2 Τα Ασύρματα Δίκτυα Σταθερής Υποδομής

Τα ασύρματα δίκτυα σταθερής υποδομής διαθέτουν ένα ενσύρματο κορμό από σταθερούς κόμβους που είναι συνδεδεμένοι με το υπόλοιπο δίκτυο ή το Internet. Τους σταθερούς αυτούς σταθμούς τους καλούμε σταθμούς βάσης (base stations – BS) ή σημεία πρόσβασης (access points – AP). Οι κινητοί χρήστες επικοινωνούν απευθείας με τους σταθερούς σταθμούς και γενικά δεν εγκαθιστούν point-to-point συνδέσεις με άλλους κινητούς χρήστες. Κάθε AP έχει μια περιοχή κάλυψης, ή κυψέλη, μέσα στο οποίο μπορεί να στείλει μηνύματα σε άλλους κόμβους, και να λάβει μηνύματα από άλλους κόμβους. Αυτή η περιοχή κάλυψης εξαρτάται από την ακτίνα μετάδοσης του σταθμού AP. Όσοι κόμβοι βρίσκονται μέσα στη κυψέλη ενός AP μπορούν να επικοινωνήσουν απευθείας με αυτό τον AP. Επειδή οι κινητοί κόμβοι συνήθως κινούνται, είναι πιθανό ότι δεν θα βρίσκονται διαρκώς εντός της ακτίνας κάλυψης ενός και μόνο AP. Καθώς ο κόμβος αυτός μετακινείται από την περιοχή κάλυψης ενός AP στην περιοχή κάποιου άλλου AP, λαμβάνει χώρα μια διαδικασία που ονομάζουμε *handoff*. Αυτή η διαδικασία συμβαίνει όταν ένας χρήστης παύει να επικοινωνεί με τον AP σταθμό που επικοινωνούσε μέχρι στιγμής, και αρχίζει να επικοινωνεί με ένα καινούριο AP. Το σχήμα 1.1 παρουσιάζει την handoff διαδικασία ενός κινητού κόμβου από ένα AP σε έναν άλλο. Η διαδικασία του handoff θα πρέπει να είναι εντελώς διάφανη, έτσι ώστε ο χρήστης να μην αντιλαμβάνεται την μετάβαση από την μια περιοχή κάλυψης στην άλλη. Τα ασύρματα δίκτυα με σταθερή υποδομή χρησιμοποιούνται συχνά σε κτίρια εταιρειών και σε πανεπιστημιακούς χώρους, ή σε περιοχές όπου τα σημεία πρόσβασης μπορούν να συνδεθούν εύκολα σε ένα προϋπάρχον δίκτυο. Επιπλέον, θα πρέπει να αναφέρουμε ότι τα κυψελοειδή δίκτυα που χρησιμοποιούμε σήμερα στην κινητή τηλεφωνία είναι μια ειδική μορφή ασύρματων δικτύων σταθερής υποδομής.

Για όσο χρόνο ένας κινητός κόμβος βρίσκεται μέσα στην περιοχή του οικείου του δικτύου, τότε μπορεί να προσπελαύνει το Internet, να λαμβάνει e-mail, κλπ, σαν να ήταν

[4] Τα Wireless Access Points είναι σταθεροί κόμβοι που επικοινωνούν με τους κινητούς κόμβους ενός ad-hoc δικτύου, και ταυτόχρονα είναι συνδεδεμένοι με ένα σταθερό δίκτυο (όπως είναι για παράδειγμα το Internet). Στην ουσία δηλαδή, λειτουργούν σαν gateways μεταξύ ενός ad-hoc δικτύου και ενός σταθερού δικτύου.

κανονικά ένας ενσύρματα συνδεδεμένος κόμβος του δικτύου. Όμως, όταν ο κόμβος αυτός εγκαταλείψει το οικείο του δίκτυο, τότε εμφανίζονται διάφορες δυσκολίες στην δρομολόγηση επειδή το subnet κομμάτι της IP διεύθυνσης του κόμβου, και το δίκτυο στο οποίο μετακινείται είναι πιθανόν να διαφέρουν. Έτσι, ο κόμβος αυτός μπορεί να μην λαμβάνει πλέον τα πακέτα που προορίζονται σε αυτόν. Για να επιλυθεί αυτό το πρόβλημα έχει αναπτυχθεί το MobileIP πρωτόκολλο [14, 15]. Το MobileIP είναι μια επέκταση του IP που επιτρέπει σε ένα κινητό κόμβο να χρησιμοποιεί δύο IP διευθύνσεις. Η πρώτη χρησιμοποιείται σαν ταυτότητα του χρήστη, ενώ η δεύτερη χρησιμοποιείται για την δρομολόγηση. Αυτές οι διευθύνσεις επιτρέπουν σε κόμβους να στέλνουν και να λαμβάνουν δεδομένα μέσα σε δίκτυα διαφορετικά από τα δικά τους οικεία δίκτυα. Τέτοια μη τοπικά δίκτυα αναφέρονται συχνά σαν ξένα (foreign) δίκτυα. Η λειτουργία του Mobile IP στηρίζεται στην διάκριση συγκεκριμένων κόμβων σε home agents και foreign agents. Η ομάδα εργασίας Mobile IP working group[5] της IETF επιβλέπει την διαδικασία προτυποποίησης του Mobile IP πρωτοκόλλου.



Σχήμα 1.1: Handoff μεταξύ Access Points

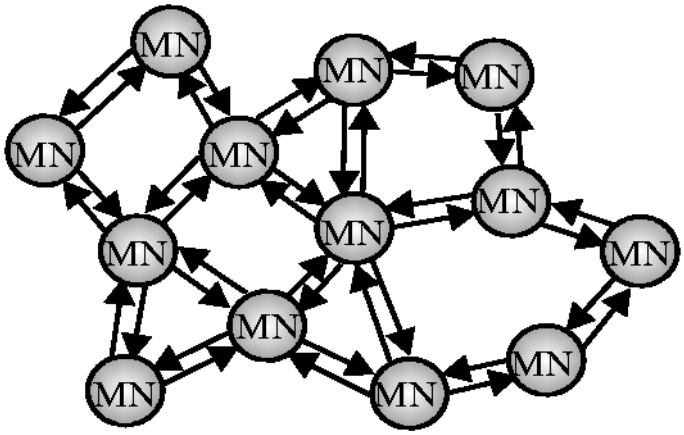
1.1.3 Τα Ασύρματα Δίκτυα Χωρίς Σταθερή Υποδομή (Ad-hoc Δίκτυα)

Σε αντίθεση με τα δίκτυα με σταθερή υποδομή, τα ad-hoc δίκτυα δεν έχουν οποιαδήποτε προ-εγκατεστημένη υποδομή (π.χ. καλώδια), και όλοι οι κόμβοι είναι ικανοί να μετακινούνται. Η επικοινωνία στα ad-hoc δίκτυα γίνεται σε βάση ομοτιμίας, αφού κάθε κινητός κόμβος επικοινωνεί απευθείας με κάποιον άλλο. Επειδή η ακτίνα επικοινωνίας των κόμβων είναι συνήθως περιορισμένη, συχνά απαιτούνται πολλά hops για να επικοινωνήσουν δύο κόμβοι. Έτσι, οι κόμβοι θα πρέπει να μπορούν να λειτουργούν και σαν δρομολογητές (routers) για τους άλλους κόμβους του δικτύου, έτσι ώστε τα πακέτα δεδομένων να μπορούν να προωθηθούν προς τους παραλήπτες τους. Στο σχήμα 1.2 παρουσιάζεται ένα παράδειγμα ενός ad-hoc δικτύου. Επειδή δεν υπάρχει κάποιος κορμός από συνδέσεις με καλώδια μέσω του οποίου να μπορεί να γίνει η δρομολόγηση, ένα ad-hoc δίκτυο χρειάζεται ένα πρωτόκολλο δρομολόγησης

[5] Internet Engineering Task Force (IETF) IP Routing for Wireless/Mobile Hosts (MobileIP) Working Group Charter. http://www.ietf.org/html.charters/mobileip-charter.html .

που να μπορεί να βρίσκει και να διατηρεί διαδρομές, ακόμη και σε δίκτυα που η τοπολογία αλλάζει δυναμικά.

Όπως έχουμε ήδη αναφέρει, οι κινητοί κόμβοι έχουν αρκετά χαρακτηριστικά που καθιστούν τα παραδοσιακά πρωτόκολλα δρομολόγησης μη εφαρμόσιμα. Εξαιτίας αυτών των περιορισμών που χαρακτηρίζουν τους ασύρματους, κινητούς κόμβους, ένα ad-hoc πρωτόκολλο δρομολόγησης θα πρέπει να μπορεί να παρέχει διαδρομές με το ελάχιστο δυνατό κόστος ελέγχου (control overhead) και καθόλου κόστος δεδομένων (data overhead). Επίσης, θα πρέπει να απαιτεί όσο το δυνατό μικρότερο χρόνο επεξεργασίας. Επιπλέον, εξαιτίας των χαρακτηριστικών της ασύρματης μετάδοσης, η ακτίνα επικοινωνίας των κόμβων είναι συνήθως περιορισμένη. Αφού η επικοινωνία μεταξύ δύο κόμβων απαιτεί συνήθως πολλά hops, θα πρέπει το πρωτόκολλο δρομολόγησης να μπορεί να βρίσκει διαδρομές πολλών hops που να συνδέουν τους συγκεκριμένους κόμβους. Το πρωτόκολλο θα πρέπει επίσης να είναι loop-free, δηλαδή να μην επιτρέπει δημιουργία κύκλων στις διαδρομές αφού κάτι τέτοιο θα μείωνε το ήδη περιορισμένο εύρος ζώνης στο κανάλι επικοινωνίας. Το Mobile Ad-hoc Networking Working Group (manet[6]) της IETF αναπτύσσει και θέτει πρότυπα για πρωτόκολλα δρομολόγησης, ειδικά για ad-hoc δίκτυα. Αρκετά πρωτόκολλα έχουν ήδη προταθεί για να γίνουν πρότυπα. Μερικά από αυτά θα παρουσιάσουμε στο επόμενο κεφάλαιο.



Σχήμα 1.2: Παράδειγμα Ad-hoc Δικτύου

Τα ad-hoc δίκτυα έχουν γενικά περιορισμένο κόστος διαχείρισης, σε σύγκριση με τα ενσύρματα δίκτυα. Αυτά τα δίκτυα είναι αυτο-ρυθμιζόμενα (self-configuring), και έτσι μπορούν να διατηρούν τις συνδέσεις του δικτύου και τις πληροφορίες δρομολόγησης χωρίς την ανάγκη

[6] Το manet ή Mobile Ad-hoc NETworking Working Group είναι μια ομάδα εργασίας, που στα πλαίσια του IETF, αναπτύσσει και θέτει τα πρότυπα για πρωτόκολλα δρομολόγησης σε ad-hoc δίκτυα. Η κεντρική σελίδα αυτής της ομάδας στο Internet βρίσκεται στη διεύθυνση: http://www.ietf.org/html.charters/manet-charter.html .

ρητής αρχικοποίησης από το διαχειριστή του δικτύου. Τυπικά παραδείγματα ad-hoc δικτύων είναι τα δίκτυα που αναπτύσσουν ομάδες διάσωσης, και τα δίκτυα που σχηματίζουν οι υπολογιστές ατόμων που συμμετέχουν σε συνέδρια. Τα ad-hoc δίκτυα είναι επίσης κατάλληλα για επικοινωνία σε περιοχές όπου δεν προϋπάρχει η κατάλληλη (ενσύρματη) δικτυακή υποδομή, όπως συμβαίνει σε απομακρυσμένες περιοχές και σε περιοχές που διεξάγονται στρατιωτικές επιχειρήσεις.

1.2. Το Πρόβλημα της Επικοινωνίας στα Ad-Hoc Κινητά Δίκτυα

Όπως έχουμε ήδη αναφέρει, ένα ad-hoc δίκτυο είναι μια συλλογή από κινητούς σταθμούς με δυνατότητα ασύρματης επικοινωνίας, οι οποίοι σχηματίζουν ένα προσωρινό δίκτυο χωρίς την ύπαρξη σταθερής υποδομής ή την υποστήριξη κάποιου κεντρικοποιημένου ελέγχου. Σε ένα τέτοιο δίκτυο, όταν δύο σταθμοί θέλουν να επικοινωνήσουν μεταξύ τους, μπορεί να μη βρίσκονται σε ακτίνα επικοινωνίας. Έτσι θα πρέπει να χρησιμοποιήσουν κάποιους ενδιάμεσους κόμβους, οι οποίοι να είναι πρόθυμοι να προωθήσουν τα πακέτα τους.

Ένα βασικό πρόβλημα επικοινωνίας σε ένα τέτοιο δίκτυο είναι η αποστολή πληροφορίας από κάποιο αποστολέα S σε κάποιο καθορισμένο παραλήπτη R. Θα πρέπει φυσικά να λαμβάνουμε υπόψη τη δυναμική φύση των ad-hoc δικτύων, δηλαδή το ότι οι τοπικές συνδέσεις είναι προσωρινές και μπορεί να αλλάζουν ανάλογα με την κίνηση των χρηστών. Ο ρυθμός μετακίνησης του κάθε χρήστη μπορεί να μεταβάλλεται, ενώ κάποιοι συγκεκριμένοι σταθμοί μπορεί να σταματήσουν (προσωρινά) προκειμένου να εκτελέσουν κάποια άλλα καθήκοντα (π.χ. να πάρουν μετρήσεις). Σε ένα τέτοιο περιβάλλον, η εκτέλεση ενός κατανεμημένου πρωτοκόλλου έχει πολύ μεγάλη πολυπλοκότητα, και η επικοινωνία μεταξύ δύο κόμβων μπορεί να αποδειχθεί πολύ δύσκολη υπόθεση.

Ο πιο απλός και κοινός τρόπος επικοινωνίας είναι ο σχηματισμός διαδρομών από ενδιάμεσους κόμβους (δηλ. χρήστες). Υποθέτουμε ότι υπάρχει μια σύνδεση μεταξύ δύο κόμβων αν ο ένας βρίσκεται εντός της ακτίνας μετάδοσης του άλλου, και έτσι μπορούν να επικοινωνήσουν απευθείας. Στην πραγματικότητα, αυτή η προσέγγιση χρησιμοποιείται σε ad-hoc δίκτυα που είτε καλύπτουν ένα μικρό χώρο (δηλαδή το προσωρινό δίκτυο έχει μικρή διάμετρο σε σχέση με την ακτίνα μετάδοσης), είτε είναι πυκνά (δηλαδή αποτελούνται από μεγάλο αριθμό κόμβων). Αφού όλες σχεδόν οι περιοχές καλύπτονται από κάποιους κόμβους, η μετάδοση χωρίς περιορισμούς σε όλους τους εντός ακτίνας επικοινωνίας κόμβους (broadcasting[7]) μπορεί να λειτουργήσει αποδοτικά.

[7] Ο όρος *broadcasting* χρησιμοποιείται για να περιγράψει τη μετάδοση ενός μηνύματος χωρίς περιορισμούς σε όλους τους γειτονικούς κόμβους που βρίσκονται εντός ακτίνας επικοινωνίας.

Όμως, σε ad-hoc δίκτυα μεγαλύτερου εύρους επιφάνειας το broadcasting δεν είναι πρακτικό, αφού δύο διαφορετικοί κόμβοι δεν μπορούν να επικοινωνήσουν με χρήση απλού broadcast λόγω του ότι δεν υπάρχουν οι απαραίτητοι κόμβοι που να καλύπτουν όλες τις ενδιάμεσες τους περιοχές. Δηλαδή, η εύρεση και διατήρηση μιας αρκετά μεγάλης διαδρομής είναι πολύ δύσκολη. Αν ένας μικρός αριθμός κόμβων που συμμετέχουν αρχικά στη διαδρομή κινηθούν κατά τρόπο που να μην είναι πλέον εντός ακτίνας επικοινωνίας, τότε η διαδρομή παύει να είναι έγκυρη. Επιπλέον, η διαδρομή που συνδέει δύο κόμβους μπορεί να είναι πολύ μεγάλη, ακόμη και αν οι κόμβοι αυτοί είναι σε κοντινές περιοχές.

Στην συνέχεια αυτής της εργασίας θα παρουσιάσουμε μια διαφορετική προσέγγιση για την επίλυση αυτού του προβλήματος, που εκμεταλλεύεται την φυσική κίνηση των κόμβων ενός αραιού και εξαπλωμένου ad-hoc δικτύου.

1.3. Συνεισφορά της Διπλωματικής

Η διπλωματική αυτή εργασία έχει σαν σκοπό να συνεισφέρει στην επίλυση του προβλήματος επικοινωνίας στα ad-hoc κινητά δίκτυα με τη δημιουργία ενός νέου γενικού περιβάλλοντος προσομοίωσης, καθώς και με την παρουσίαση και πειραματική αξιολόγηση δύο σύγχρονων πρωτοκόλλων.

Το περιβάλλον προσομοίωσης έχει δημιουργηθεί έτσι ώστε να είναι όσο το δυνατό πιο γενικό και ικανό να προσομοιώνει οποιοδήποτε πρωτόκολλο δρομολόγησης. Επιπλέον, το περιβάλλον αυτό μας επιτρέπει να προσομοιώνουμε ειδικές καταστάσεις που να αντανακλούν πραγματικές συνθήκες, όπως είναι για παράδειγμα διάφορα εμπόδια, περιοχές με μεγάλη πυκνότητα χρηστών, και αγαπημένες διαδρομές.

Τα δύο πρωτόκολλα που παρουσιάζουμε μπορούν να χαρακτηριστούν από την χρήση πιθανοτικών τεχνικών στην θεμελίωση της λειτουργίας τους. Πιο συγκεκριμένα, επανα-υλοποιήσαμε ένα υπάρχον πρωτόκολλο, το πρωτόκολλο του *Φιδιού*, στο οποίο οι κόμβοι της υποστήριξης κινούνται με μια δομή που μοιάζει με *φιδάκι*. Επιπρόσθετα αναλύσαμε και υλοποιήσαμε ένα νέο πρωτόκολλο, το πρωτόκολλο των *Δρομέων*, στο οποίο οι κόμβοι της υποστήριξης κινούνται ο ένας ανεξάρτητα απο τον άλλο σαν μεμονωμένοι *δρομείς*.

Επίσης όπως θα δούμε και στη συνέχεια, η ανάλυση που κάναμε δείχνει ότι το νεοσυσταθέν πρωτόκολλο των *Δρομέων* υπερέχει σε όλους τους τομείς έναντι του προϋπάρχοντος πρωτοκόλλου του *Φιδιού*.

1.4. Οργάνωση της Διπλωματικής

Μέχρι τώρα έχουμε μιλήσει για τη φύση και τη σημασία των ad-hoc δικτύων. Στην συνέχεια θα αναφέρουμε στο *Κεφάλαιο 2* κάποιες γενικές έννοιες που χρησιμοποιούνται στο χώρο αυτών των δικτύων. Συγκεκριμένα, θα αναφερθούμε στην επικοινωνία στο επίπεδο του φυσικού μέσου ή μέσου προσπέλασης, καθώς και στα προβλήματα και τις λύσεις που έχουν προταθεί για αυτού του είδους την επικοινωνία. Στο ίδιο κεφάλαιο αναφέρουμε επίσης τις επιθυμητές ιδιότητες που πρέπει να έχουν τα πρωτόκολλα που αναπτύσσονται για ad-hoc δίκτυα, καθώς και τα ανοιχτά προβλήματα που υπάρχουν σε αυτό το χώρο. Στο *Κεφάλαιο 3* κάνουμε μια σύντομη παρουσίαση των πρωτοκόλλων που έχουν προταθεί μέχρι στιγμής. Στο τέλος του κεφαλαίου παρουσιάζουμε μια συνοπτική σύγκριση των πρωτοκόλλων σε επίπεδο χαρακτηριστικών. Το *4ο Κεφάλαιο* μας εισάγει στο χώρο του περιβάλλοντος προσομοίωσης. Ειδικότερα, περιγράφουμε το γενικό περιβάλλον προσομοίωσης που αναπτύξαμε για αξιολόγηση πρωτοκόλλων δρομολόγησης σε ad-hoc δίκτυα. Στο *Κεφάλαιο 5* περιγράφουμε δύο σύγχρονα πρωτόκολλα, αυτά των *Δρομέων (runners)* και του *Φιδιού (snake)*, και επιπλέον παρουσιάζουμε τη πειραματική αξιολόγηση των πρωτοκόλλων αυτών στο προαναφερθέν περιβάλλον προσομοίωσης. Τέλος, στο *6ο Κεφάλαιο* αναφέρουμε τα συμπεράσματα μας τόσο για την απόδοση και ορθότητα των δύο πρωτοκόλλων, όσο και για τη λειτουργικότητα του περιβάλλοντος προσομοίωσης.

Θα πρέπει επίσης να αναφέρουμε ότι επεξηγήσεις των όρων και των συντμήσεων που χρησιμοποιούμε σε αυτή τη διπλωματική εργασία βρίσκονται στο Παράρτημα Α. Ακόμη, στο Παράρτημα Β παρουσιάζουμε ένα μικρό οδηγό που περιγράφει τη λειτουργία του συστήματος προσομοίωσης, και τέλος, στο Παράρτημα Γ δίνουμε επιλεγμένα κομμάτια από τον κώδικα υλοποίησης των πρωτοκόλλων των *Δρομέων* και του *Φιδιού*.

2. Γενικές Έννοιες στα Κινητά Δίκτυα

Σε αυτό το κεφάλαιο περιγράφουμε αρχικά την έννοια των ad-hoc δικτύων. Ακολούθως, εξηγούμε το ρόλο του επιπέδου Ελέγχου Προσπέλασης Μέσου (MAC) και επίσης περιγράφουμε ένα πρωτόκολλο αυτού του τύπου που βρίσκει εφαρμογή στα ad-hoc δίκτυα. Στην συνέχεια αναφέρουμε τις διάφορες κατηγορίες στις οποίες μπορούμε να κατατάξουμε τα πρωτόκολλα επικοινωνίας για ad-hoc δίκτυα, και τέλος αναφέρουμε μερικές επιθυμητές ιδιότητες που πρέπει να έχουν τα ad-hoc δίκτυα.

2.1. Ασύρματα Ad-Hoc Κινητά Δίκτυα

Ένα ασύρματο ad-hoc δίκτυο είναι μια συλλογή από κινητούς κόμβους[8] (nodes) που χωρίς την εκ των προτέρων ύπαρξη οποιασδήποτε σταθερής υποδομής, σχηματίζουν ένα προσωρινό δίκτυο. Καθένας από τους κόμβους του δικτύου διαθέτει ασύρματο σύστημα επικοινωνίας, και μπορεί να επικοινωνεί με τους γείτονες του είτε με τη χρήση ραδιοκυμάτων, είτε με την χρήση υπερύθρων. Ένα παράδειγμα τέτοιου δικτύου είναι μια ομάδα από φορητούς υπολογιστές που επικοινωνούν απευθείας μεταξύ τους μέσω των υπέρυθρων τους θυρών.

Ένα από τα σημαντικότερα χαρακτηριστικά των ad-hoc δικτύων είναι ότι δεν χρησιμοποιούν οποιοδήποτε κεντρικοποιημένο έλεγχο. Με αυτό τον τρόπο εξασφαλίζουμε ότι το δίκτυο δεν πρόκειται να καταρρεύσει απλά και μόνο επειδή κάποιος κόμβος έχει φύγει από την ακτίνα επικοινωνίας των υπολοίπων. Επιπλέον, οι κόμβοι θα πρέπει να μπορούν να εισέρχονται αλλά και να εξέρχονται από το δίκτυο κατά βούληση. Εξαιτίας της περιορισμένης εμβέλειας επικοινωνίας του κάθε χρήστη, μπορεί να χρειαστεί η παρέμβαση ενδιάμεσων κόμβων για να επιτύχουμε επικοινωνία με απομακρυσμένους κόμβους. Κάθε κόμβος που επιθυμεί να συμμετέχει σε ένα ad-hoc δίκτυο θα πρέπει να είναι πρόθυμος να προωθήσει πακέτα προς άλλους κόμβους. Άρα κάθε κόμβος θα πρέπει να μπορεί να λειτουργεί και σαν απλός χρήστης[9], αλλά και σαν δρομολογητής[10].

Επίσης, τα ad-hoc δίκτυα έχουν την δυνατότητα χειρισμού μεταβολών στην τοπολογία, καθώς και δυσλειτουργιών στους κόμβους του δικτύου. Για παράδειγμα, αν ένας κόμβος

[8] Ένας κόμβος (node) σε ένα ad-hoc δίκτυο είναι μια συσκευή που μπορεί να ανταλλάζει ψηφιακά δεδομένα με τους γείτονες της χρησιμοποιώντας ασύρματη επικοινωνία. Ισοδύναμα, χρησιμοποιούμε επίσης τους όρους χρήστης (user), και κινητός σταθμός (mobile host – mh). Σε αυτή την εργασία, όταν αναφερόμαστε σε κόμβους θα εννοούμε τους κινητούς σταθμούς, ενώ όταν αναφερόμαστε σε κορυφές θα εννοούμε τις κορυφές ενός γραφήματος.

[9] Ένας χρήστης (host) είναι ένας απλός κόμβος στο δίκτυο που χαρακτηρίζεται από την IP-διεύθυνση (ταυτότητα) του.

[10] Οι δρομολογητές ή διαδρομητές (routers) είναι ειδικοί κόμβοι ενός δικτύου (όπως είναι το Internet), που μεταξύ άλλων έχουν σαν σκοπό την επιλογή διαδρομών για την προώθηση πακέτων πληροφορίας. Για αυτό το λόγο τρέχουν ένα ειδικό αλγόριθμο που ονομάζουμε αλγόριθμο δρομολόγησης (routing protocol).

εγκαταλείψει το δίκτυο (π.χ. λόγω σφάλματος ή άδειας μπαταρίας) με αποτέλεσμα να υπάρχει διακοπή σε κάποια από τις συνδέσεις, τότε οι κόμβοι που επηρεάζονται πρέπει απλά να αναζητήσουν καινούριες διαδρομές και έτσι να επιλύσουν το πρόβλημα της επικοινωνίας. Αυτό φυσικά αυξάνει την καθυστέρηση στην επικοινωνία, αλλά το δίκτυο μας εξακολουθεί να είναι λειτουργήσιμο.

Τέλος, ένα από τα πλεονεκτήματα των ad-hoc δικτύων είναι ότι λόγω της ασύρματης φύσης του μέσου μετάδοσης, δεν έχουμε περιορισμούς λόγω της τοπολογίας της διασύνδεσης των κόμβων. Συγκεκριμένα, στα ενσύρματα δίκτυα η φυσική διασύνδεση γίνεται εκ των προτέρων με αποτέλεσμα να καθορίζεται και η τοπολογία της διασύνδεσης. Αντίθετα, στα ad-hoc δίκτυα δεν έχουμε αυτό τον περιορισμό αφού αν δύο κόμβοι βρίσκονται εντός ακτίνας επικοινωνίας τότε μπορούν ανά πάσα στιγμή να επικοινωνήσουν μεταξύ τους.

2.2. Το Επίπεδο Ελέγχου Προσπέλασης Μέσου (MAC Layer)

Στην εργασία τους [5], οι Micah Adler και Christian Scheideler, προτείνουν ένα μοντέλο τριών επιπέδων για την προσέγγιση και επίλυση προβλημάτων σε ένα αυθαίρετο στατικό ad-hoc δίκτυο. Αρχικά, έχουμε το επίπεδο ελέγχου προσπέλασης μέσου (Medium Access Control layer). Το επίπεδο αυτό είναι υπεύθυνο για την επικοινωνία από σημείο–σε-σημείο (node-to-node) στο φυσικό μέσο (που στα ασύρματα δίκτυα είναι προφανώς η ατμόσφαιρα ή καλύτερα το κενό). Ακολούθως έχουμε το επίπεδο επιλογής διαδρομής, το οποίο είναι υπεύθυνο για την εύρεση κατάλληλων διαδρομών για τα πακέτα. Τέλος, έχουμε το επίπεδο χρονοδρομολόγησης, που είναι υπεύθυνο για τον καθορισμό της σειράς αποστολής των πακέτων που στέλνει ένας χρήστης.

Στη συνέχεια μελετούμε το πρώτο επίπεδο, δηλαδή το επίπεδο ελέγχου προσπέλασης μέσου. Οι πληροφορίες αυτές θα χρησιμοποιηθούν και αργότερα στην περιγραφή της λειτουργίας του προσομοιωτή που αναπτύξαμε. Τα άλλα δύο επίπεδα αποτελούν στην ουσία την υλοποίηση του κάθε προτεινόμενου πρωτοκόλλου, και θα μας απασχολήσουν στα επόμενα κεφάλαια.

Τα τρέχοντα πρωτόκολλα που χρησιμοποιούνται στο επίπεδο MAC

Αρχικά περιγράφουμε τα MAC πρωτόκολλα που χρησιμοποιούνται σήμερα στον χώρο των τηλεπικοινωνιών, και ακολούθως το πιθανοτικό πρωτόκολλο που προτείνουν οι Micah Adler και Christian Scheideler. Το τελευταίο πρωτόκολλο βρίσκει εφαρμογή σε ad-hoc κινητά δίκτυα.

Σήμερα, υπάρχουν 3 βασικοί μέθοδοι που χρησιμοποιούνται στο χώρο των επικοινωνιών για την μετάδοση σημάτων: FDMA, TDMA και CDMA. Στο FDMA (Frequency

Division Multiple Access) τα σήματα μεταδίδονται ταυτόχρονα μέσω διαφορετικών συχνοτήτων. Στο TDMA (Time Division Multiple Access), τα σήματα χρησιμοποιούν την ίδια συχνότητα, αλλά διαχωρίζονται με την χρήση διαφορετικών, μη επικαλυπτόμενων χρονικών σχισμών (time slots) κατά την μετάδοση τους. Τέλος, η CDMA (Code Division Multiple Access) είναι μια τεχνική που συνδυάζει τις FDMA και TDMA. Η βασική ιδέα του CDMA είναι ότι οι χρήστες χρησιμοποιούν κώδικες για την μετάδοση που είναι "κάθετοι" ο ένας στον άλλο. Έτσι, με την χρήση ενός κατάλληλου συσχετιστή (correlator), ο δέκτης έχει την δυνατότητα να αποκωδικοποιήσει ένα σήμα από ένα συγκεκριμένο αποστολέα χωρίς να παρεμβάλλεται με τα σήματα άλλων.

Σήμερα, το TDMA χρησιμοποιείται στα περισσότερα πρότυπα (standards) όπως για παράδειγμα το DECT, το GSM (στην Ευρώπη), το PDC (στην Ιαπωνία), και το IS-54 (στις ΗΠΑ). Το CDMA θεωρείται ως η τεχνική που θα χρησιμοποιείται στα μελλοντικά ασύρματα δίκτυα τηλεπικοινωνιών, όπως το UMTS (Universal Mobile Communication System), που αναπτύσσεται στην Ευρώπη, ή το IMT-2000, που αναπτύσσεται από την ITU (το Διεθνές σώμα για τηλεπικοινωνιακά πρότυπα).

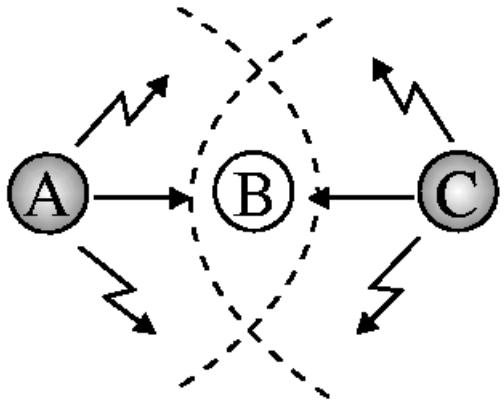
Όμως, οι δύο αυτές μέθοδοι, TDMA και CDMA, δεν είναι χρήσιμες στα ασύρματα ad-hoc δίκτυα αφού είναι πολύ ευαίσθητες ακόμη και σε μικρές διαφορές στην ταχύτητα λειτουργίας του κάθε κινητού σταθμού. Όταν υπάρχει κεντρικός έλεγχος (για παράδειγμα μέσω ενός σταθερού σταθμού βάσης), τότε το πρόβλημα λύνεται με την χρήση ενός σήματος (reference signal) που στέλνεται από το σταθμό βάσης σε τακτά χρονικά διαστήματα.

Στην ειδική περίπτωση των ασύρματων τοπικών δικτύων (mobile LANs), τα περισσότερα δίκτυα βασίζονται στη μέθοδο Πολλαπλής Μετάδοσης με Ανίχνευση Φέροντος (CSMA), στη μέθοδο polling, και στο TDMA. Αυτά περιλαμβάνονται στο IEEE 802.11 προτεινόμενο πρότυπο για ασύρματα τοπικά δίκτυα.

Στα CSMA πρωτόκολλα, κάθε κόμβος ακούει κατά πόσο υπάρχουν άλλοι κόμβοι που μεταδίδουν. Εάν όχι, προσπαθεί να μεταδώσει το μήνυμα του. Εάν αναγνωρίσει ότι υπάρχει σύγκρουση με κάποιο άλλο μήνυμα, υποχωρεί, και δοκιμάζει και πάλι να στείλει το μήνυμα του μετά από ένα τυχαίο χρονικό διάστημα. Η CSMA μέθοδος έχει χρησιμοποιηθεί με μεγάλη επιτυχία στο Ethernet. Δυστυχώς όμως αυτή η στρατηγική αποτυγχάνει στα ad-hoc δίκτυα, αφού ένας χρήστης δεν μπορεί να διακρίνει αν ο δέκτης του μηνύματος του δέχεται ταυτόχρονα ένα άλλο μήνυμα (από κάποιο τρίτο χρήστη) ή όχι[11].

[11] Στο σχήμα 2.1, οι A και C μεταδίδουν ταυτόχρονα στον B. Ο B δεν μπορεί να δεχθεί κανένα από τα δύο μηνύματα, αφού αυτά αλληλο-παρεμβάλλονται. Το πρόβλημα σε αυτή την περίπτωση είναι ότι ούτε ο A, ούτε ο C γνωρίζουν για την σύγκρουση (αφού η μεταξύ τους απόσταση είναι μεγαλύτερη από την ακτίνα επικοινωνίας) και έτσι δεν μπορούν να ξέρουν με βεβαιότητα αν η μετάδοση έγινε επιτυχώς ή όχι.

Όπως έχουμε ήδη αναφέρει, οι Micah Adler και Christian Scheideler προτείνουν μια διαφορετική κλάση από πρωτόκολλα επιπέδου MAC για ad-hoc δίκτυα. Το σημαντικότερο πλεονέκτημα αυτών των πρωτοκόλλων είναι ότι έχουν την δυνατότητα να προσαρμόζονται γρήγορα σε ευμετάβλητες καταστάσεις, και ως εκ τούτου να μπορούν να χρησιμοποιηθούν για την αποδοτική επικοινωνία σε δυναμικά ad-hoc δίκτυα.



Σχήμα 2.1: Το Hidden Terminal Πρόβλημα

Μια νέα κλάση πρωτοκόλλων για το επίπεδο MAC

Στην συνέχεια παρουσιάζουμε μια νέα κλάση από απλά, τοπικά, πιθανοτικά πρωτόκολλα ελέγχου για το επίπεδο MAC. Τα πρωτόκολλα αυτά επιτρέπουν σε κάθε κόμβο να αποφασίζει ανεξάρτητα από τους άλλους για το πότε να στείλει ένα πακέτο.

Ας αρχίσουμε περιγράφοντας την μετάδοση ενός πακέτου από ένα κόμβο v σε ένα κόμβο w. Καθορίζουμε ότι ένα επιτυχημένο hop[12] αποτελείται από δύο βήματα: Πρώτο, ο v στέλνει ένα πακέτο στο w, και δεύτερο, ο w στέλνει μια επιβεβαίωση (acknowledge) πίσω στον v. Άρα, ένα hop μπορεί να αποτύχει, είτε επειδή το w ήταν μπλοκαρισμένο[13] τη στιγμή που ο v έστελνε το πακέτο, είτε επειδή ο v ήταν μπλοκαρισμένος όταν ο w έστελνε την επιβεβαίωση. Ο v συνεχίζει να στέλνει το πακέτο στον w μέχρι να λάβει μια επιβεβαίωση από αυτόν.

Επειδή η στρατηγική μας για επιλογή διαδρομής βασίζεται σε σταθερές διαδρομές, ο w δεν χρειάζεται να ξέρει κατά πόσο ο v έχει λάβει την επιβεβαίωση ή όχι. Εφόσον λάβει το πακέτο μπορεί αμέσως να συνεχίσει την παραπέρα μετάδοση του κατά μήκος της διαδρομής.

[12] Στον χώρο των ad-hoc δικτύων, όταν αναφερόμαστε σε ένα hop εννοούμε τη μετάδοση της πληροφορίας από ένα κόμβο α σε ένα κόμβο β. Οι δύο αυτοί κόμβοι θα πρέπει να είναι σε απόσταση μικρότερη της ακτίνας μετάδοσης τους, και άρα να μπορούν να επικοινωνούν απευθείας.

[13] Όταν λέμε ότι ένας κόμβος είναι μπλοκαρισμένος, εννοούμε ότι τη συγκεκριμένη στιγμή είναι απασχολημένος λαμβάνοντας ένα μήνυμα από κάποιον άλλο κόμβο, και άρα σε αυτή την περίπτωση είναι αδύνατο να λάβει το μήνυμα που του στέλνει κάποιος τρίτος.

Εδώ θα πρέπει να επισημάνουμε ότι αγνοούμε την πιθανότητα λαθών κατά την μετάδοση, καθώς και την πιθανότητα ο κόμβος να είναι εκτός λειτουργίας. Σε αυτή την περίπτωση, η επιβεβαίωση μπορεί να χρησιμοποιηθεί για τον έλεγχο της ορθής μετάδοσης (επαναλαμβάνοντας την πιο πάνω διαδικασία σε περίπτωση λάθους) και για την εξακρίβωση της επαγρύπνησης του παραλήπτη (διαφορετικά θα πρέπει να επιλέγουμε ένα άλλο κόμβο στην περίπτωση που δεν λάβουμε επιβεβαίωση σε κάποιο προκαθορισμένο χρόνο).

Στην συνέχεια, ορίζουμε το γράφημα μετάδοσης (*transmission graph*) G = (V, τ), σε κάποια χρονική στιγμή t_0, ως το πλήρες μη κατευθυνόμενο γράφημα με σύνολο κορυφών V τους κόμβους του δικτύου, και ακμές που καθορίζονται απ την συνάρτηση τ: V x V → $\Re^+$. Για κάθε ακμή {u, v}, η τ{u, v} αναπαριστάνει την ελάχιστη ένταση μετάδοσης (transmission power) που επιτρέπει στον u να στείλει ένα μήνυμα στον v, και αντιστρόφως.

Ο πιθανοτικός αλγόριθμος που προτείνεται στην [5] έχει ως εξής. Αρχικά, θεωρούμε το γράφημα μετάδοσης (transmission graph) G = (V, τ). Ακολούθως, ορίζουμε ένα πίνακα προσπέλασης P = ($p_{v,w}$), v, w $\in$ V με $p_{v,w} \in [0, 1)$ για κάθε v, w $\in$ V. Αυτός ο πίνακας προσπέλασης χρησιμοποιείται ως ακολούθως: Έστω ότι ένας κόμβος v θέλει να στείλει ένα πακέτο Q στον κόμβο w (κάτι που επιτρέπεται αν $p_{v,w} > 0$). Για όσο χρόνο ο v δεν λαμβάνει επιβεβαίωση από τον w, αποφασίζει σε κάθε χρονικό βήμα με πιθανότητα $p_{v,w}$ να ξεκινήσει ένα hop του Q στο w.

Στην εργασία [5] αποδεικνύεται ότι αυτή η μέθοδος εξασφαλίζει ότι για κάθε δίκτυο που χρησιμοποιεί αυτό το MAC πρωτόκολλο, δεν απαιτείται καμιά συνεννόηση μεταξύ των επικοινωνούντων κόμβων. Κάθε κόμβος μπορεί να πάρει μια αυθαίρετη απόφαση κατά πόσο θα πρέπει να στείλει ένα πακέτο. Αυτή η τακτική εγγυάται ότι η πιθανότητα επιτυχίας για κάθε απόπειρα μετάδοσης είναι τουλάχιστον 50%. Αυτό το αποτέλεσμα είναι ιδιαίτερα χρήσιμο και θα το χρησιμοποιήσουμε και αργότερα, κατά την περιγραφή λειτουργίας του προσομοιωτή.

Τέλος, πολύ ενδιαφέρων είναι και η προσέγγιση της εργασίας [22]. Όπως χαρακτηριστικά αναφέρεται στην εργασία αυτή, μια επιτυχημένη δρομολόγηση πακέτων θα πρέπει να είναι δίκαιη, και να μεγιστοποιεί την ροή. Οι συγγραφείς της εργασίας προτείνουν ένα δι-επίπεδο μοντέλο εξυπηρέτησης, που παρέχει “δίκαιη” κατανομή του εύρους του καναλιού στην ροή πακέτων, και επιπρόσθετα μεγιστοποιεί την επαναχρησιμοποίηση του εύρους αυτού. Όπως θα δούμε και στο 4^{ο} κεφάλαιο, η υλοποίηση του προσομοιωτή μας έγινε έτσι ώστε η κατανομή του μέσου μετάδοσης να γίνεται με όσο το δυνατό πιο δίκαιο τρόπο.

2.3. Κατηγορίες Πρωτοκόλλων για Ad-hoc Κινητά Δίκτυα

Τα πρωτόκολλα δρομολόγησης για τα ad-hoc δίκτυα μπορούν να καταταχθούν σε διάφορες κατηγορίες ανάλογα με τις ιδιότητες τους. Στην συνέχεια παρουσιάζουμε τις σημαντικότερες από αυτές. Επίσης, επισημαίνουμε ότι υπάρχουν και πολλές κατηγορίες υβρίδια των παρακάτω, που συνδυάζουν τα χαρακτηριστικά αυτών των κατηγοριών.

- Κεντρικοποιημένα / Κατανεμημένα (Centralized / Distributed)

  Ένας τρόπος κατηγοριοποίησης των πρωτοκόλλων δρομολόγησης είναι η διάκριση τους ανάμεσα σε κεντρικοποιημένους και κατανεμημένους αλγόριθμους. Στην περίπτωση των κεντρικοποιημένων αλγορίθμων, όλες οι επιλογές διαδρομών γίνονται σε ένα κεντρικό κόμβο, ενώ στους κατανεμημένους αλγόριθμους ο υπολογισμός αυτών των διαδρομών γίνεται από κοινού στους κόμβους του δικτύου. Τα πρωτόκολλα για ad-hoc δίκτυα είναι ως επί το πλείστο κατανεμημένα, αφού διαφορετικά θα ήταν πολύ ευάλωτα στα σφάλματα. Συγκεκριμένα, τα κεντρικοποιημένα πρωτόκολλα υστερούν στο ότι η λειτουργία τους εξαρτάται απόλυτα από ένα και μόνο κόμβο, κάτι που τα κάνει ακατάλληλα για ad-hoc δίκτυα.

- Στατικά / Προσαρμοζόμενα (Static / Adaptive)

  Η επόμενη κατηγοριοποίηση πρωτοκόλλων δρομολόγησης αφορά το αν οι διαδρομές αλλάζουν ανάλογα με το φορτίο του δικτύου ή όχι. Στους στατικούς αλγόριθμους, η διαδρομή ανάμεσα σε ένα ζεύγος αποστολέα / παραλήπτη είναι σταθερή, ανεξάρτητα από τις συνθήκες που επικρατούν στο δίκτυο. Η διαδρομή αυτή μπορεί να μεταβληθεί μόνο στην περίπτωση που ένας κόμβος ή μια σύνδεση τεθεί εκτός λειτουργίας. Προφανώς, αυτοί οι αλγόριθμοι δεν μπορούν να επιτύχουν μεγάλο throughput[14] όταν οι συνθήκες κίνησης πακέτων στο δίκτυο μεταβάλλονται διαρκώς. Τα περισσότερα και σημαντικότερα δίκτυα πακέτων χρησιμοποιούν κάποιο προσαρμοζόμενο πρωτόκολλο για την δρομολόγηση. Αυτά τα πρωτόκολλα έχουν την ικανότητα να επιλέγουν κάποια διαδρομή, ανάλογα με τη συμφόρηση που παρατηρείται στους διάφορους κόμβους του δικτύου. Αξίζει πάντως να σημειωθεί πως μέχρι σήμερα δεν έχει δημιουργηθεί πρωτόκολλο δρομολόγησης για ad-hoc δίκτυα που να μπορεί να προσαρμόζεται στις συνθήκες κίνησης του δικτύου. Δηλαδή δεν υπάρχει πρωτόκολλο που να επιλέγει κάποια διαφορετική από την προκαθορισμένη διαδρομή, απλά και μόνο επειδή η πρώτη είναι αρκετά απασχολημένη.

[14] Το throughput είναι ένας όρος που χρησιμοποιείται για την αξιολόγηση της απόδοσης ενός δικτύου. Ποσοτικά, ισούται με τον αριθμό των πακέτων που μεταδίδονται με επιτυχία στο δίκτυο ανά μονάδα χρόνου

- Μετα-δραστικά / Προ-δραστικά (Reactive / Proactive)

  Η Τρίτη κατηγοριοποίηση έχει περισσότερη σχέση με τα ad-hoc δίκτυα και αφορά την κατάταξη των αλγορίθμων δρομολόγησης σε μετα-δραστικούς και προ-δραστικούς. Τα προ-δραστικά πρωτόκολλα προσπαθούν να διατηρούν διαρκώς έγκυρες τις διαδρομές μέσα στο δίκτυο. Έτσι, όταν ένα πακέτο θα πρέπει να προωθηθεί η διαδρομή είναι ήδη γνωστή και μπορεί να χρησιμοποιηθεί αμέσως. Αντίθετα, τα μετα-δραστικά πρωτόκολλα εισάγουν ένα μηχανισμό που βρίσκει μια διαδρομή μόνο όταν αυτό απαιτείται. Δηλαδή όταν χρειάζεται να προωθήσουμε κάποιο πακέτο, ξεκινάμε κάποια διαδικασία εύρεσης της κατάλληλης διαδρομής. Τα προ-δραστικά πρωτόκολλα πλεονεκτούν στο ότι όταν απαιτείται κάποια διαδρομή, η καθυστέρηση μέχρι να αρχίσει η μετάδοση των πακέτων είναι πολύ μικρή. Όμως, τα προ-δραστικά πρωτόκολλα μειονεκτούν στο ότι χρειάζονται κάποιο χρόνο πριν συγκλίνουν σε μια σταθερή κατάσταση. Αυτό μπορεί να προκαλέσει σημαντικά προβλήματα αν η τοπολογία του δικτύου αλλάζει γρήγορα.

- Ελεγχόμενης Μετάδοσης / Μη Ελεγχόμενης Μετάδοσης (Power Controlled / Simple)

  Οι Micah Adler και Christian Scheideler στην εργασία τους [5], εισάγουν την έννοια της ελεγχόμενης μετάδοσης. Σε αυτή την περίπτωση ο κάθε χρήστης μπορεί να μεταβάλει την ένταση εκπομπής και ως εκ τούτου την ακτίνα επικοινωνίας. Όπως αναφέρουν οι ίδιοι οι συγγραφείς, αυτό μπορεί να παρομοιαστεί με την ικανότητα των ανθρώπων να επικοινωνούν ρυθμίζοντας ανά πάσα στιγμή την ένταση της φωνής τους. Φανταστείτε για παράδειγμα τη περίπτωση όπου οι άνθρωποι θα μπορούσαν να επικοινωνήσουν χρησιμοποιώντας μόνο ένα επίπεδο στην ένταση της φωνής τους! Τότε θα είχαμε σημαντικό πρόβλημα επικοινωνίας. (Σκεφτείτε για παράδειγμα τι θα γινόταν σε μια συνωστισμένη αίθουσα εκδηλώσεων!)

  Χρησιμοποιώντας δίκτυα με δυνατότητα ελεγχόμενης μετάδοσης μπορούμε να φτιάξουμε πρωτόκολλα επικοινωνίας που να πλεονεκτούν έναντι των κλασσικών στους ακόλουθους τομείς:

  - Η συμφόρηση ανάμεσα στους χρήστες μπορεί να μειωθεί σημαντικά
  - Η κατανάλωση ενέργειας μπορεί να μειωθεί σημαντικά
  - Η ασφάλεια των μεταδόσεων μπορεί να αυξηθεί

- Επιτακτικά / Μη Επιτακτικά Δίκτυα (Compulsory / Non-compulsory)

  Τα δίκτυα των οποίων οι κόμβοι μπορούν να ελεγχθούν, δηλαδή να καθοριστεί η κίνηση τους, ονομάζονται επιτακτικά (compulsory). Αντίθετα, τα δίκτυα των οποίων οι κόμβοι μπορούν να κινούνται αυθαίρετα χωρίς κανένα έλεγχο ονομάζονται μη επιτακτικά (non-compulsory). Μια ενδιάμεση κατάσταση είναι τα ημι-επιτακτικά (semi-compulsory) δίκτυα, στα οποία ένας αριθμός από nodes μπορεί να ελέγχεται από το πρωτόκολλο, το οποίο και καθορίζει την κίνηση τους. Όπως θα δούμε στη συνέχεια, τα δύο πρωτόκολλα που μελετούμε σε αυτή την εργασία (snake και runners) είναι ημι-επιτακτικά, αφού το πρωτόκολλο ελέγχει ένα μικρό μόνο μέρος των κόμβων του δικτύου.

2.4. Επιθυμητές Ιδιότητες Πρωτοκόλλων για Ad-hoc Κινητά Δίκτυα

Εφόσον η εργασία αυτή ασχολείται κυρίως με τα υπάρχοντα, αλλά και κάποια προτεινόμενα, πρωτόκολλα επικοινωνίας για ad-hoc δίκτυα, θα πρέπει να αναφέρουμε τι χαρακτηριστικά επιθυμούμε να έχουν τα πρωτόκολλα αυτά. Πιο συγκεκριμένα, θα αναφέρουμε κάποιες από τις ιδιότητες που θα πρέπει να έχουν τα πρωτόκολλα αυτά έτσι ώστε να ικανοποιούν τις προσδοκίες μας για αποδοτική επικοινωνία.

- Κατανεμημένη λειτουργία

  Τα πρωτόκολλα θα πρέπει να είναι κατανεμημένα. Δηλαδή, σε καμιά περίπτωση δεν θα πρέπει η λειτουργία του δικτύου να εξαρτάται από ένα κεντρικό κόμβο ελέγχου. Αυτή η ιδιότητα είναι επιθυμητή ακόμη και στα σταθερά δίκτυα. Η διαφορά είναι ότι σε ένα ad-hoc δίκτυο οι κόμβοι μπορούν να εισέλθουν και να εξέλθουν από το δίκτυο πολύ εύκολα και ακριβώς λόγω αυτής της κινητικότητας των κόμβων, το δίκτυο μπορεί να διαμοιραστεί (δηλαδή να χωριστεί σε δύο τμήματα τα οποία δεν έχουν καμιά επικοινωνία μεταξύ τους).

- Διαδρομές χωρίς κύκλους

  Για να έχει υψηλή απόδοση ένα πρωτόκολλο, θα πρέπει να εγγυάται ότι οι διαδρομές που καθορίζει για την μεταφορά πακέτων δεν περιέχουν κύκλους. Δηλαδή, θα πρέπει ένα πακέτο που μεταδίδεται προς ένα συγκεκριμένο στόχο να μην περνάει δύο φορές από τον ίδιο ενδιάμεσο κόμβο. Με αυτό το τρόπο αποφεύγουμε να καταναλώνουμε άδικα πολύτιμο εύρος ζώνης επικοινωνίας, επεξεργαστική ισχύ, μνήμη, και φυσικά ενέργεια.

- Λειτουργία που να καθορίζεται από τις ανάγκες επικοινωνίας

  Για να περιορίσουμε το πρόσθετο κόστος του δικτύου, και άρα να μειώσουμε την κατανάλωση πόρων του δικτύου πέραν από αυτό που είναι απαραίτητο, τα πρωτόκολλα μας θα πρέπει να είναι μετα-δραστικά. Αυτό σημαίνει ότι το πρωτόκολλο θα πρέπει να κάνει τις απαιτούμενες ενέργειες για εύρεση διαδρομής και μετάδοση πακέτων μόνο όταν αυτό είναι απαραίτητο, και όχι να αποστέλλει περιοδικά πληροφορίες ελέγχου.

- Υποστήριξη αμφίδρομης επικοινωνίας στις συνδέσεις

  Λόγω της χρήσης ραδιοκυμάτων για την επικοινωνία, συνήθως όλες οι συνδέσεις μας μπορούν να είναι αμφίδρομες. Όμως μόνο αυτό δεν είναι αρκετό αν το πρωτόκολλο μας δεν εκμεταλλεύεται κατάλληλα αυτή την ιδιότητα.

- Ασφάλεια επικοινωνίας

  Ο χώρος της επικοινωνίας με χρήση ραδιοκυμάτων είναι ιδιαίτερα ευπαθής σε επιθέσεις υποκλοπής. Για να εξασφαλίσουμε την αναμενόμενη λειτουργία από το πρωτόκολλο δρομολόγησης θα πρέπει να χρησιμοποιήσουμε κάποια προληπτικά μέτρα ασφάλειας. Η χρήση συστημάτων αυθεντικοποίησης και κρυπτογράφησης είναι οι προφανείς μέθοδοι για την επίλυση του προβλήματος. Σε αυτή την περίπτωση όμως, εισάγεται το πρόβλημα της διανομής κλειδιών ανάμεσα στους κόμβους του δικτύου. Μια εναλλακτική μέθοδος (η οποία και βρίσκεται ακόμη υπό συζήτηση) είναι η χρήση του IP-sec [7] που χρησιμοποιεί τη τεχνική “tunneling” για την μεταφορά όλων των πακέτων.

- Εξοικονόμηση ενέργειας

  Οι κόμβοι σε ένα ad-hoc δίκτυο μπορεί να είναι Laptops ή PDAs τα οποία είναι πολύ περιορισμένα σε ενέργεια, και ως εκ τούτου μπορεί να χρησιμοποιούν συχνά την κατάσταση stand-by[15] για εξοικονόμηση ενέργειας. Άρα, είναι απαραίτητο τα πρωτόκολλα για τα ad-hoc δίκτυα να έχουν δυνατότητα υποστήριξης αυτών των καταστάσεων.

[15] Ο όρος stand-by είναι ένας όρος που χρησιμοποιείται για να περιγράψει μια συσκευή που βρίσκεται σε ημι-ενεργό κατάσταση. Μια συσκευή σε αυτή την κατάσταση διατηρεί λειτουργήσιμα μόνο τα εντελώς απαραίτητα υπο-συστήματα της. Στην περίπτωση των κινητών δικτύων για παράδειγμα, μια συσκευή που βρίσκεται σε κατάσταση stand-by θα μπορούσε μόνο να λάβει και όχι να μεταδώσει μηνύματα.

- Πολλαπλές διαδρομές

  Για να περιορίσουμε το κόστος επανάληψης υπολογισμού των διαδρομών σε ένα δίκτυο λόγω των τοπολογικών μεταβολών ή λόγω της συμφόρησης των πακέτων, μπορούμε να διατηρούμε περισσότερες από μια διαδρομές για κάθε προορισμό. Το δίκτυο θα μπορούσε για παράδειγμα, στην περίπτωση που κάποια διαδρομή προς ένα συγκεκριμένο προορισμό παύει να είναι έγκυρη, να χρησιμοποιεί κάποια δεύτερη διαδρομή η οποία και εξακολουθεί να είναι έγκυρη. Αυτό μας απαλλάσσει από το κόστος της επανάληψης του υπολογισμού της διαδρομής.

- Υποστήριξη για QoS[16]

  Ανάλογα με την χρήση που προορίζουμε για το κάθε δίκτυο, είναι πιθανό να απαιτούμε κάποιας μορφής εξασφάλιση στην ποιότητα της παρεχόμενης υπηρεσίας. Για παράδειγμα, στην περίπτωση που θέλουμε να υποστηρίζουμε real-time[17] audio, θα πρέπει να εξασφαλίσουμε ότι το δίκτυο μας μπορεί να εγγυηθεί κάποιο εύρος ζώνης επικοινωνίας (communication bandwidth[18]) σε κάθε επιτυχημένη σύνδεση. Πάντως, αυτό το χαρακτηριστικό είναι πολύ δύσκολο να επιτευχθεί αφού λόγω της δυναμικής φύσης των ad-hoc δικτύων είναι σχεδόν αδύνατο να εγγυηθούμε ότι δύο κόμβοι μπορούν να έχουν στη διάθεση τους ένα κανάλι δεδομένης χωρητικότητας που να τους συνδέει για κάποιο συγκεκριμένο χρονικό διάστημα.

[16] QoS – Quality of Service, αυτός ο όρος χρησιμοποιείται συνήθως για να περιγράψει ότι η παρεχόμενη υπηρεσία εγγυάται κάποια επίπεδα ποιότητας σε συγκεκριμένους τομείς

[17] Το real-time audio είναι η μεταφορά των ψηφιακών δεδομένων κωδικοποίησης της φωνής με επαρκή ρυθμό έτσι ώστε να μπορεί ο δέκτης να ακούει το ηχητικό σήμα του αποστολέα με ικανοποιητική ποιότητα και χωρίς διακοπές.

[18] Ο όρος bandwidth χρησιμοποιείται στο χώρο των δικτύων για να περιγράψει το εύρος ζώνης συχνοτήτων που μπορεί να χρησιμοποιηθεί κατά την μετάδοση μηνυμάτων μεταξύ δύο κόμβων. Ο όρος αυτός χρησιμοποιείται για να εκφράσει αναλογικά την ποσότητα της πληροφορίας που μπορούν να ανταλλάξουν δύο κόμβοι ανά μονάδα χρόνου.

3. Τα Υπάρχοντα Πρωτόκολλα Δρομολόγησης

Σε αυτό το κεφάλαιο περιγράφουμε μερικά από τα υπάρχοντα πρωτόκολλα που έχουν κατά καιρούς προταθεί για την δρομολόγηση μηνυμάτων στα ad-hoc δίκτυα. Στο τέλος του κεφαλαίου παρουσιάζουμε ένα συγκριτικό πίνακα των πρωτοκόλλων αυτών. Αξίζει να σημειώσουμε πως κανένα από αυτά τα πρωτόκολλα δεν έχει αναλυθεί σε θεωρητικό επίπεδο.

Destination Sequenced Distance Vector – DSDV

Περιγραφή

Το DSDV ([8]) πρωτόκολλο είναι ένα hop-by-hop[19] distance vector πρωτόκολλο που σε κάθε κόμβο του δικτύου διατηρεί ένα πίνακα δρομολόγησης στον οποίο για όλους τους εφικτούς προορισμούς αποθηκεύει τον επόμενο κόμβο (next-hop) καθώς και τον απαιτούμενο αριθμό hops μέχρι τον προορισμό αυτό (hop-count). Ο αλγόριθμος DSDV απαιτεί την περιοδική εκπομπή μηνυμάτων με σκοπό την ενημέρωση των πινάκων δρομολόγησης. Το πλεονέκτημα του έναντι των παραδοσιακών distance vector πρωτοκόλλων είναι ότι ο DSDV εγγυάται την μη ύπαρξη κύκλων.

Για να μπορεί να εγγυηθεί την μη ύπαρξη κύκλων, ο DSDV αριθμεί κάθε διαδρομή με ένα αριθμό (sequence number). Ο αριθμός αυτός δείχνει πόσο πρόσφατη είναι η κάθε διαδρομή και οι διαδρομές με το μεγαλύτερο αριθμό είναι οι προτιμότερες. Πιο συγκεκριμένα, μια διαδρομή R θεωρείται προτιμότερη από μια διαδρομή R’ αν η R έχει μεγαλύτερο αριθμό ή, αν οι διαδρομές έχουν τον ίδιο αριθμό αλλά η R έχει μικρότερο αριθμό hop-count. Ο αριθμός αυτός αυξάνεται όταν κάποιος κόμβος A αντιληφθεί ότι η διαδρομή μέχρι τον προορισμό D έχει πάψει να είναι έγκυρη. Έτσι, την επόμενη φορά που ο A θα κοινοποιήσει στους γείτονες του τον πίνακα δρομολόγησης του, θα δώσει στην διαδρομή προς τον D άπειρο hop-count και ένα αριθμό (sequence number) που είναι μεγαλύτερος από πριν.

Στην ουσία ο DSDV είναι ένας distance vector αλγόριθμος, τροποποιημένος κατάλληλα έτσι ώστε να προσαρμόζεται καλύτερα στα ad-hoc δίκτυα. Αυτές οι τροποποιήσεις αφορούν τις ενημερώσεις που λαμβάνουν χώρα όταν αλλάζει η τοπολογία του δικτύου στο χρόνο μεταξύ των περιοδικών μεταδόσεων. Για να μειώσουμε την ποσότητα της πληροφορίας σε αυτά τα πακέτα μπορούμε να χρησιμοποιήσουμε δύο τύπους από πακέτα ενημέρωσης: πλήρους και μερικής ενημέρωσης. Τα πακέτα πλήρους ενημέρωσης περιέχουν όλη την πληροφορία του

[19] Τα hop-by-hop πρωτόκολλα είναι αυτά που διαβιβάζουν τα μηνύματα τους από κόμβο σε κόμβο μέχρι να φτάσουν στον τελικό παραλήπτη.

πίνακα δρομολόγησης, ενώ τα πακέτα ενημέρωσης περιέχουν πληροφορία μόνο για εκείνες τις διαδρομές που έχουν αλλάξει από την τελευταία μετάδοση.

Ιδιότητες

Επειδή ο DSDV εξαρτάται από τις περιοδικές μεταδόσεις, χρειάζεται κάποιο χρόνο έτσι ώστε να συγκλίνει προτού να μπορεί να χρησιμοποιηθεί κάποια διαδρομή. Αυτός ο χρόνος σύγκλισης μπορεί να θεωρηθεί αμελητέος σε ένα στατικό δίκτυο, όπου η τοπολογία δεν αλλάζει και τόσο συχνά. Όμως, στα ad-hoc δίκτυα η τοπολογία περιμένουμε να μεταβάλλεται πολύ συχνά, και έτσι ο χρόνος σύγκλισης μπορεί να σημαίνει ότι ένας μεγάλος αριθμός πακέτων έχουν απορριφθεί προτού βρεθεί μια κατάλληλη διαδρομή. Επίσης, αυτή η περιοδική μετάδοση εισάγει μεγάλο πρόσθετο κόστος στο δίκτυο.

Ad-hoc On Demand Distance Vector – AODV

Περιγραφή

Το AODV πρωτόκολλο δρομολόγησης επιτρέπει την multi-hop[20] δρομολόγηση μεταξύ των συμμετεχόντων κόμβων που επιθυμούν να εγκαταστήσουν και να διατηρήσουν ένα ad-hoc δίκτυο. Το AODV βασίζεται στον distance vector αλγόριθμο. Η βασική διαφορά είναι ότι ο AODV είναι μετα-δραστικός, σε αντίθεση με τα προ-δραστικά πρωτόκολλα όπως είναι ο DV. Δηλαδή ο AODV ζητάει μια διαδρομή μόνο όταν αυτό απαιτείται, και δεν χρειάζεται να διατηρεί διαδρομές σε κόμβους με τους οποίους δεν επικοινωνεί. Για όσο χρόνο τα άκρα των συνδέσεων έχουν έγκυρες διαδρομές μεταξύ τους, ο AODV δεν παίζει κανένα ρόλο.

Τα χαρακτηριστικά αυτού του πρωτοκόλλου περιλαμβάνουν ανεξαρτησία από κύκλους στις διαδρομές, και επίσης ότι η ακύρωση κάποιας σύνδεσης προκαλεί άμεση ενημέρωση των επηρεαζόμενων και μόνο κόμβων. Επιπλέον, ο AODV έχει υποστήριξη για πολλαπλή δρομολόγηση (multicast routing), και αποφεύγει το πρόβλημα “μετρήματος στο άπειρο” (“counting to infinity”) του Bellman Ford [9]. Η χρησιμοποίηση αρίθμησης στις διαδρομές εγγυάται ότι μια διαδρομή είναι “φρέσκια”.

Ο αλγόριθμος χρησιμοποιεί διαφορετικά μηνύματα για την εξεύρεση και την διατήρηση των συνδέσεων. Όποτε ένας κόμβος χρειαστεί να προσπαθήσει και να βρει μια διαδρομή σε κάποιο άλλο κόμβο, στέλνει ένα μήνυμα Αίτησης Διαδρομής (Route Request ή RREQ), προς όλους τους γείτονες του. Το μήνυμα RREQ διαδίδεται μέσω του δικτύου μέχρι να φτάσει στον προορισμό του, ή μέχρι να μέχρι να φτάσει σε ένα κόμβο με αρκετά “φρέσκια” διαδρομή προς

[20] Ο όρος multi-hop χρησιμοποιείται για να περιγράψει πρωτόκολλα στα οποία η επικοινωνία γίνεται με αλυσιδωτή μετάδοση πακέτων πληροφορίας από κόμβο σε κόμβο μέχρι να φτάσει στον απαιτούμενο προορισμό.

τον απαιτούμενο προορισμό. Τότε, η διαδρομή γνωστοποιείται στέλνοντας προς τα “πίσω” ένα μήνυμα RREP μέχρι τον αρχικό αποστολέα.

Ο αλγόριθμος χρησιμοποιεί hello μηνύματα (ένα ειδικό τύπο RREP μηνύματος) τα οποία στέλνονται περιοδικά προς όλους τους άμεσους γείτονες. Αυτά τα μηνύματα έχουν σαν στόχο την διαρκή ενημέρωση όλων των κόμβων ως προς την διατήρηση των γειτόνων τους. Οι γείτονες που χρησιμοποιούν διαδρομές μέσω του συγκεκριμένου κόμβου θα εξακολουθήσουν να θεωρούν τις διαδρομές σαν έγκυρες. Αν τα hello μηνύματα σταματήσουν να φθάνουν από ένα συγκεκριμένο κόμβο, τότε οι γείτονες μπορούν να υποθέσουν ότι ο συγκεκριμένος κόμβος έχει απομακρυνθεί εκτός ακτίνας επικοινωνίας, και έτσι να σημαδέψει αυτή την σύνδεση σαν σπασμένη. Ταυτόχρονα, θα πρέπει να γνωστοποιήσει την αποτυχία της σύνδεσης σε όλους τους επηρεαζόμενους κόμβους (με την αποστολή ενός ειδικού μηνύματος RREP). Επιπλέον, ο AODV έχει ένα ειδικό μήνυμα για την ακύρωση πολλαπλών διαδρομών (multicast route).

Διαχείριση Πίνακα Δρομολόγησης

Ο AODV χρειάζεται να φυλάει τις ακόλουθες πληροφορίες για κάθε εγγραφή στον πίνακα διαδρομών:

- Destination IP Address: IP διεύθυνση του προορισμού.
- Destination Sequence Number: Ο σειριακός αριθμός για αυτόν το προορισμό.
- Hop Count: Ο αριθμός των hops μέχρι τον προορισμό.
- Next Hop: Ο γείτονας, ο οποίος έχει επιλεγεί για την αποστολή πακέτων στον προορισμό μέσω αυτής της διαδρομής.
- Lifetime: Ο χρόνος για τον οποίο η διαδρομή θεωρείται έγκυρη
- Active Neighbor List: Οι γειτονικοί κόμβοι οι οποίοι χρησιμοποιούν ενεργά αυτή την εγγραφή.
- Request Buffer: Εξασφαλίζει ότι μια αίτηση επεξεργάζεται μόνο μια φορά.

Εύρεση Διαδρομής

Ένας κόμβος στέλνει σε όλους τους γείτονες του ένα RREQ μήνυμα μόνο όταν χρειάζεται μια διαδρομή προς ένα προορισμό και δεν υπάρχει καμιά τέτοια διαδρομή διαθέσιμη. Αυτό μπορεί να συμβεί όταν η διαδρομή προς τον προορισμό είναι άγνωστη, ή όταν η προηγούμενη διαδρομή έχει ξεπεράσει το χρονικό όριο ζωής της. Μετά την μετάδοση του RREQ μηνύματος

στους γείτονες του, ο κόμβος περιμένει για ένα RREP μήνυμα. Αν η απάντηση δεν φθάσει μέσα σε ένα συγκεκριμένο χρονικό διάστημα, ο κόμβος μπορεί να επαναλάβει την αποστολή του RREQ μηνύματος, ή να υποθέσει ότι δεν υπάρχει κάποια διαδρομή προς τον απαιτούμενο προορισμό.

Η προώθηση των RREQ μηνυμάτων γίνεται όταν ο κόμβος που λαμβάνει το RREQ μήνυμα δεν έχει μια διαδρομή προς τον προορισμό. Τότε, επαναμεταδίδει το RREQ μήνυμα. Ο κόμβος δημιουργεί επίσης μια προσωρινή αντίστροφη διαδρομή προς την IP διεύθυνση του αρχικού κόμβου μέσα στον πίνακα δρομολόγησης του, θέτοντας στην τιμή του Next Hop την IP διεύθυνση του γειτονικού κόμβου από τον οποίο έλαβε το RREQ μήνυμα. Αυτό γίνεται έτσι ώστε να διατηρούμε μια διαδρομή προς τα πίσω (προς τον αρχικό κόμβο που έκανε την αίτηση), η οποία πιθανόν να χρειαστεί για την μετάδοση του RREP μηνύματος, αν τελικά βρεθεί η διαδρομή που ζήτησε ο αρχικός κόμβος. Η διαδρομή αυτή είναι προσωρινή με την έννοια ότι είναι έγκυρη για πολύ λιγότερο χρόνο από ότι μια πραγματική διαδρομή.

Όταν το RREQ μήνυμα φτάσει σε ένα κόμβο που είτε είναι ο τελικός προορισμός, είτε έχει μια έγκυρη διαδρομή προς αυτό τον προορισμό, τότε ένα RREP μήνυμα δημιουργείται και μεταδίδεται πίσω προς τον αιτούντα κόμβο. Ενόσω το RREP μήνυμα προωθείται προς τα πίσω, δημιουργείται μια διαδρομή προς τον προορισμό, και όταν τελικά το RREP μήνυμα φτάσει στον αρχικό κόμβος, τότε έχουμε μια διαδρομή από τον αρχικό στον τελικό κόμβο.

Συντήρηση Διαδρομών

Όταν ένας κόμβος αντιληφθεί ότι η διαδρομή προς ένα γείτονα δεν είναι πλέον έγκυρη, θα αφαιρέσει αυτή την διαδρομή από τον πίνακα δρομολόγησης του, και θα στείλει ένα μήνυμα που θα ανακοινώνει την ακύρωση της διαδρομής. Αυτό το μήνυμα έχει την μορφή ενός RREP μηνύματος προς τους γείτονες που χρησιμοποιούν αυτή την διαδρομή, και τους ενημερώνει ότι η διαδρομή αυτή δεν είναι πλέον έγκυρη. Για αυτό το σκοπό ο AODV αλγόριθμος χρησιμοποιεί μια λίστα με τους ενεργούς γείτονες για να ξέρει ανά πάσα στιγμή ποιοι γείτονες χρησιμοποιούν την συγκεκριμένη διαδρομή. Οι κόμβοι που λαμβάνουν αυτό το μήνυμα θα πρέπει να επαναλάβουν αυτή την διαδικασία ενημέρωσης. Το μήνυμα θα φτάσει τελικά στους επηρεαζόμενους κόμβους (που στέλνουν πακέτα μέσω των συγκεκριμένων διαδρομών) και τότε θα αποφασίσουν είτε να διακόψουν την αποστολή δεδομένων, είτε θα ζητήσουν μια καινούρια διαδρομή στέλνοντας ένα καινούριο RREQ μήνυμα.

Ιδιότητες

Ο AODV πλεονεκτεί σε σχέση με τους κλασσικούς αλγόριθμους δρομολόγησης, όπως τον Distance Vector και τον Link State, στο ότι έχει περιορίσει σημαντικά τον αριθμό των μηνυμάτων

δρομολόγησης μέσα στο δίκτυο. Ο AODV το πετυχαίνει αυτό με την χρήση μιας μετα-δραστικής προσέγγισης. Πιθανότατα αυτή η προσέγγιση είναι αναγκαία στα ad-hoc δίκτυα για να έχουν ικανοποιητική απόδοση όταν η τοπολογία αλλάζει συχνά.

Οι αριθμοί Sequence Number που χρησιμοποιεί ο AODV αντιπροσωπεύουν την φρεσκάδα μιας διαδρομής, και αυξάνουν όταν συμβαίνει κάτι στην περιβάλλουσα περιοχή. Ο αριθμός αυτός αποτρέπει την δημιουργία κύκλων, αλλά μπορεί πάντως να προκαλέσει άλλα προβλήματα. Τι συμβαίνει για παράδειγμα όταν οι αριθμοί αυτοί του δικτύου πάψουν να είναι συγχρονισμένοι; Αυτό θα μπορούσε να συμβεί όταν το δίκτυο χωρίζεται στα δύο, ή όταν οι αριθμοί αυτοί περιφέρονται κυκλικά στο δίκτυο.

Ο AODV υποστηρίζει μόνο μια διαδρομή για κάθε προορισμό. Ωστόσο, είναι σχετικά εύκολο να τροποποιήσουμε τον AODV έτσι ώστε να υποστηρίζει πολλαπλές διαδρομές προς κάθε προορισμό. Αντί να ζητάμε μια καινούρια διαδρομή όταν λήξει μια προηγούμενη, θα μπορούσαμε να δοκιμάζαμε την προηγούμενη διαδρομή προς τον ζητούμενο προορισμό. Η πιθανότητα να είναι ακόμη έγκυρη αυτή η διαδρομή θα πρέπει να είναι μεγάλη.

Το AODV δεν υποστηρίζει μονόδρομες συνδέσεις. Όταν ένας κόμβος λαμβάνει ένα RREQ μήνυμα, θα εγκαταστήσει μια αντίστροφη διαδρομή προς τον αρχικό κόμβο χρησιμοποιώντας τον κόμβο από τον οποίο έλαβε το RREQ μήνυμα σαν Next Hop. Αυτό σημαίνει ότι το RREP μήνυμα μεταδίδεται προς τα πίσω πάνω στην ίδια διαδρομή που χρησιμοποίησε και το RREQ μήνυμα. Η υποστήριξη μονόδρομων συνδέσεων θα μπορούσε να υποστηρίξει όλες τις συνδέσεις και όχι μόνο τις αμφίδρομες. Ωστόσο, δεν είναι σίγουρο εάν σε ένα πραγματικό περιβάλλον είναι επιθυμητές οι μονόδρομες συνδέσεις. Για παράδειγμα, τα μηνύματα επιβεβαίωσης (acknowledgements) του MAC πρωτοκόλλου της IEEE 802.11 δεν θα δούλευαν σε μονόδρομες συνδέσεις.

Dynamic Source Routing – DSR

Περιγραφή

Το Dynamic Source Routing (DSR) πρωτόκολλο ανήκει επίσης στην κλάση των μετα-δραστικών πρωτοκόλλων. Ο DSR επιτρέπει στους κόμβους του δικτύου να ανακαλύπτουν δυναμικά μια διαδρομή μέσω πολλών hops μέσα στο δίκτυο προς οποιοδήποτε προορισμό. Σε αυτό το πρωτόκολλο, κάθε πακέτο πληροφορίας περιέχει στην επικεφαλίδα (header) του μια πλήρη λίστα με διαταγμένους όλους τους κόμβους μέσω των οποίων πρέπει να περάσει το πακέτο. Ο DSR δεν χρησιμοποιεί περιοδικά μηνύματα δρομολόγησης, και έτσι μειώνεται το πρόσθετο κόστος στο δίκτυο, εξοικονομώντας ενέργεια στους κόμβους μας καθώς και πολύτιμο εύρος ζώνης επικοινωνίας. Αντί αυτού, ο DSR στηρίζεται στο MAC επίπεδο, του οποίου το

πρωτόκολλο τον ενημερώνει για πιθανόν σπασμένες συνδέσεις. Οι δύο βασικές λειτουργίες του DSR είναι η εύρεση και η συντήρηση διαδρομών.

Εύρεση Διαδρομών

Η εύρεση διαδρομής είναι ο μηχανισμός στον οποίο ένας κόμβος X που θέλει να στείλει ένα πακέτο πληροφορίας στον Y, ανακαλύπτει μια διαδρομή από τον X στον Y. Ο κόμβος X αιτείται μια διαδρομή στέλνοντας ένα πακέτο Αίτησης Διαδρομής (Route Request – RREQ) προς τους γείτονες του. Κάθε κόμβος που λαμβάνει αυτό το RREQ μήνυμα μέσω των διαδρομών που έχει φυλαγμένες στην μνήμη του για μια διαδρομή προς τον ζητούμενο προορισμό. Ο DSR αποθηκεύει όλες τις γνωστές διαδρομές στην μνήμη του. Εάν δεν βρεθεί κάποια διαδρομή, τότε το RREQ πακέτο προωθείται περαιτέρω και προσθέτει την διεύθυνση του στην ακολουθία hops του πακέτου. Αυτή η αίτηση μεταδίδεται μέσω του δικτύου είτε μέχρι να βρεθεί ο προορισμός, ή ένας κόμβος που να γνωρίζει μια διαδρομή προς τον προορισμό. Όταν συμβεί αυτό, τότε στέλνεται ένα RREP μήνυμα μέσω της αρχικής διαδρομής προς το κόμβο που έκανε αρχικά την αίτηση. Αυτό το πακέτο περιέχει την ακολουθία των hops που πρέπει να ακολουθήσει ο αρχικός κόμβος X για να στείλει τα μηνύματα του στον προορισμό, Y.

Στην διαδικασία Εύρεσης Διαδρομής, ένας κόμβος πρώτα στέλνει ένα RREQ μήνυμα με το μέγιστο όριο διάδοσης (όριο hops) ίσο με μηδέν, απαγορεύοντας έτσι στους γείτονες του να το επανα-μεταδώσουν. Με το κόστος μιας απλής μετάδοσης ενός πακέτου προς τους γείτονες του, αυτός ο μηχανισμός επιτρέπει σε ένα κόμβο να ζητάει διαδρομές που να περιέχονται στην μνήμη των γειτόνων του.

Οι κόμβοι μπορούν επίσης να τεθούν σε μια “αδιάκριτη” κατάσταση στην οποία απενεργοποιούν το φιλτράρισμα των διευθύνσεων των πακέτων, και έτσι μπορούν να λαμβάνουν όλα τα πακέτα που μεταδίδονται στην περιοχή τους. Ακολούθως, αυτά τα πακέτα μπορούν να ερευνηθούν για χρήσιμες διαδρομές ή για μηνύματα ακύρωσης διαδρομών, και ακολούθως να πεταχτούν.

Η πίσω διαδρομή προς τον αρχικό κόμβο που την αιτήθηκε μπορεί να υπολογιστεί με διάφορους τρόπους. Ο πιο απλός τρόπος είναι να αντιστρέψουμε την σειρά των hops μέσα στο πακέτο. Αυτό πάντως απαιτεί συμμετρικές συνδέσεις. Για να το αντιμετωπίσει αυτό, ο DSR ελέγχει τις διαδρομές που έχει στην μνήμη ο κόμβος από τον οποίο έλαβε το RREP μήνυμα. Αν βρεθεί κάποια διαδρομή, τότε χρησιμοποιεί αυτήν. Ένας άλλος τρόπος είναι να χρησιμοποιήσουμε ένα RREQ μήνυμα με προορισμό τον αρχικό κόμβο. Αυτό σημαίνει ότι ο DSR μπορεί να υπολογίσει ορθές διαδρομές στην παρουσία ασύμμετρων (αμφίδρομων)

συνδέσεων. Από τη στιγμή που ανακαλύπτεται μια διαδρομή, αυτή φυλάγεται στην μνήμη με μια χρονική σφραγίδα και ακολούθως ξεκινά η φάση της διατήρησης της διαδρομής.

Διατήρηση Διαδρομής

Η Διατήρηση της Διαδρομής είναι ο μηχανισμός μέσω του οποίου ο αποστολέας S ενός πακέτου ανιχνεύει εάν η τοπολογία του δικτύου έχει αλλάξει έτσι ώστε να μην μπορεί πλέον να χρησιμοποιήσει τη διαδρομή που έχει προς τον προορισμό D. Αυτό μπορεί να συμβαίνει επειδή ένας κόμβος που περιλαμβάνεται μέσα στην διαδρομή έχει μετακινηθεί εκτός ακτίνας επικοινωνίας, ή επειδή έχει βγει εκτός λειτουργίας, καθιστώντας την επικοινωνία μέσω της παλιάς διαδρομής αδύνατη. Μια αποτυχημένη σύνδεση μπορεί να ανιχνευθεί είτε παρακολουθώντας ενεργά τις επιβεβαιώσεις (acknowledgements) που αποστέλλονται, είτε μπαίνοντας στην “αδιάκριτη” κατάσταση, έτσι ώστε να παρακολουθεί παθητικά τα πακέτα που στέλνονται από τους γειτονικούς κόμβους.

Όταν ο μηχανισμός Διατήρησης Διαδρομών αντιληφθεί ένα πρόβλημα σε κάποια διαδρομή που χρησιμοποιείται, τότε αποστέλλει ένα πακέτο Λάθους Διαδρομής πίσω στον αρχικό κόμβο. Όταν παραληφθεί το πακέτο Λάθους Διαδρομής, τότε το hop που προκαλούσε το πρόβλημα αφαιρείται από όλες τις διαδρομές που περιείχαν αυτό το hop.

Ιδιότητες

Ο DSR χρησιμοποιεί το σημαντικό πλεονέκτημα της δρομολόγησης από τη πηγή. Οι ενδιάμεσοι κόμβοι δεν χρειάζεται να διατηρούν ενημερωμένες πληροφορίες για τις διαδρομές έτσι ώστε να δρομολογούν τα πακέτα που προωθούν. Επίσης, δεν υπάρχει ανάγκη για περιοδική μετάδοση μηνυμάτων που να περιέχουν πληροφορία για τις διαδρομές. Αυτό το γεγονός μειώνει το εύρος ζώνης επικοινωνίας που καταναλώνουμε λόγω πρόσθετου κόστους, εδικά σε περιόδους όπου δεν παρατηρείται σημαντική μετακίνηση των σταθμών του δικτύου. Επίσης, με αυτό τον τρόπο εξοικονομούμε ενέργεια αφού αφενός δεν στέλνουμε τέτοια μηνύματα ενημέρωσης, και αφετέρου δεν χρειάζεται να λαμβάνουμε. Έτσι, ένας κόμβος μπορεί να τεθεί σε κατάσταση sleep mode[21].

Αυτό το πρωτόκολλο έχει το πλεονέκτημα της μάθησης διαδρομών με τον έλεγχο της πληροφορίας που περιέχεται στα πακέτα που λαμβάνει. Μια διαδρομή από τον A στον C μέσω του B σημαίνει ότι ο A μαθαίνει την διαδρομή στον C, αλλά και ο B μαθαίνει την διαδρομή μέχρι τον C. Επιπλέον, η διαδρομή προς τον αρχικό κόμβο A σημαίνει ότι και ο B γνωρίζει την

[21] Η κατάσταση sleep mode είναι μια κατάσταση στην οποία μπορεί να τεθεί ένας κόμβος του δικτύου, απέχοντας από διάφορες βασικές λειτουργίες όπως είναι η λήψη και η μετάδοση μηνυμάτων, με σκοπό την εξοικονόμηση ενέργειας. Φυσικά, σε αυτή την κατάσταση ο κόμβος αυτός δεν μπορεί να χρησιμοποιηθεί από το πρωτόκολλο για την εξασφάλιση επικοινωνίας στο δίκτυο.

διαδρομή μέχρι τον A, και ακόμη ο C γνωρίζει την διαδρομή προς τον A και B. Αυτή η μορφή ενεργούς μάθησης είναι πολύ χρήσιμη αφού μειώνει το πρόσθετο κόστος του δικτύου.

Παρόλα αυτά, κάθε πακέτο περιέχει ένα μικρό πρόσθετο κόστος που περιέχει την διαδρομή μέχρι τον αρχικό κόμβο που έστειλε το πακέτο. Αυτό το πρόσθετο κόστος αυξάνεται όταν το πακέτο πρέπει να περάσει μέσα από πολλά hops μέχρι να φτάσει στον προορισμό του. Έτσι, τα πακέτα θα είναι ελαφρά μεγαλύτερα, λόγω αυτού του πρόσθετου κόστους.

Η χρήση του συστήματος επικοινωνίας στην “αδιάκριτη” κατάσταση εισάγει επίσης θέματα ασφάλειας. Αφού στο σύστημα της επικοινωνίας δεν χρησιμοποιούμε τον μηχανισμό φιλτραρίσματος διεύθυνσης, μπορούμε να διαβάσουμε όλα τα πακέτα που μεταδίδονται στην περιοχή. Ένας πιθανός εισβολέας θα μπορούσε να ακούσει όλα τα πακέτα και να τα ψάξει για πολύτιμες πληροφορίες όπως είναι οι αριθμοί πιστωτικών καρτών. Οι εφαρμογές θα πρέπει να παρέχουν ασφάλεια κρυπτογραφώντας τα δεδομένα πριν να τα αποστέλλουν. Τα πρωτόκολλα δρομολόγησης αποτελούν πρωταρχικούς στόχους σε επιθέσεις υποκλοπής και θα πρέπει ως εκ τούτου να κρυπτογραφούνται. Μια μέθοδος για να το πετύχουμε αυτό είναι το IP-sec ([10]).

Το DSR περιέχει επίσης υποστήριξη για αμφίδρομες συνδέσεις με τη χρήση νέων αιτήσεων από τον τελικό προς τον αρχικό κόμβο (piggybacking). Αυτό μπορεί να αυξήσει την απόδοση σε σενάρια όπου έχουμε πολλές αμφίδρομες συνδέσεις. Φυσικά θα πρέπει πρώτα να έχουμε το κατάλληλο MAC πρωτόκολλο που να υποστηρίζει αυτές τις λειτουργίες.

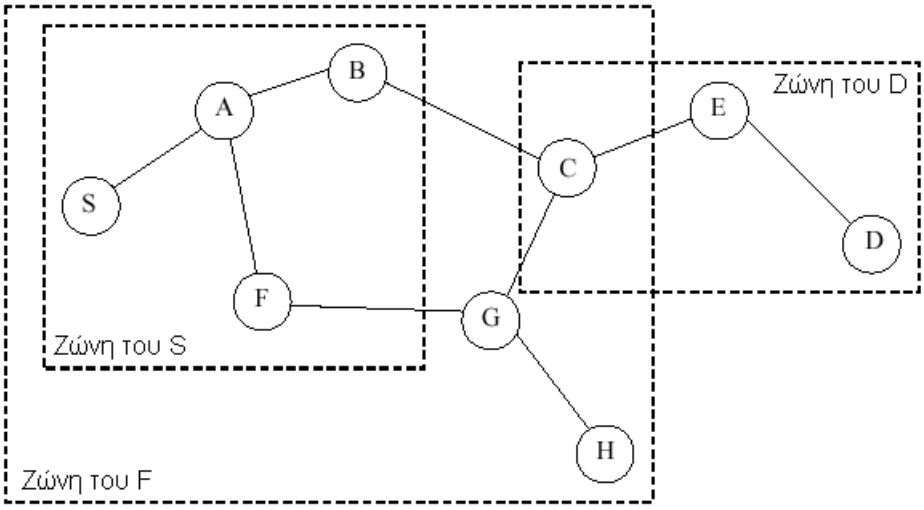
3.4. Zone Routing Protocol – ZRP

Περιγραφή

Το Zone Routing Protocol είναι ένα υβριδικό πρωτόκολλο, με χαρακτηριστικά μετα-δραστικού και προ-δραστικού πρωτοκόλλου. Το πρωτόκολλο διαιρεί το δίκτυο σε πολλές ζώνες δρομολόγησης, και χρησιμοποιεί δύο εντελώς διαφορετικά πρωτόκολλα που λειτουργούν εντός και μεταξύ των ζωνών αυτών.

Το IntrAzone Routing Protocol (IARP) λειτουργεί εντός της ζώνης δρομολόγησης και μαθαίνει την ελάχιστη απόσταση καθώς και τη διαδρομή προς κάθε κόμβο εντός της ζώνης. Αυτό το πρωτόκολλο δεν καθορίζεται ρητά, και μπορεί να είναι οποιοδήποτε προ-δραστικό πρωτόκολλο όπως το Distance-Vector ή το Link-state routing. Διαφορετικές ζώνες δρομολόγησης μπορούν να λειτουργούν και με διαφορετικά πρωτόκολλα εντός της κάθε ζώνης, αρκεί τα πρωτόκολλα αυτά να περιορίζονται στους εντός της ζώνης κόμβους. Μια αλλαγή στην τοπολογία σημαίνει ότι οι πληροφορίες ενημέρωσης διαδίδονται μόνο μέσα στις επηρεαζόμενες ζώνες, αντί να επηρεάζεται ολόκληρο το δίκτυο.

Το δεύτερο πρωτόκολλο, το IntErzone Routing Protocol (IERP) είναι μετα-δραστικό και χρησιμοποιείται για την εύρεση διαδρομών μεταξύ των διαφορετικών ζωνών. Αυτό είναι απαραίτητο αν ο αποστολέας και ο παραλήπτης δεν βρίσκονται εντός της ίδιας ζώνης δρομολόγησης. Το πρωτόκολλο σε αυτή την περίπτωση μεταδίδει ένα Route REQuest (RREQ) πακέτο σε όλους τους συνοριακούς κόμβους εντός της ζώνης δρομολόγησης, οι οποίοι με την σειρά τους προωθούν την αίτηση εάν ο παραλήπτης δεν βρίσκεται εντός της δικής τους ζώνης δρομολόγησης. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να βρεθεί ο ζητούμενος κόμβος (παραλήπτης), ο οποίος τότε στέλνει ένα Route Reply πίσω στον αποστολέα, υποδεικνύοντας του την διαδρομή. Ο IERP χρησιμοποιεί το πρωτόκολλο Bordercast Resolution Protocol (BRP) [16], το οποίο περιλαμβάνεται στον ZRP. Το BRP παρέχει υπηρεσίες bordercast (μετάδοση στους συνοριακούς κόμβους), οι οποίες δεν υπάρχουν στο IP. Το bordercasting είναι η διαδικασία αποστολής IP datagrams από ένα κόμβο σε όλους τους περιφερειακούς του κόμβους. Ο BRP κρατάει ενημερωμένες πληροφορίες για τους περιφερειακούς κόμβους και υπολογίζει μια border-cast διεύθυνση για κάθε μια από τις IP-διευθύνσεις των περιφερειακών κόμβων. Τότε, το μήνυμα που αποστέλλεται στους συνοριακούς κόμβους ενσωματώνεται σε ένα BRP πακέτο και αποστέλλεται σε κάθε συνοριακό κόμβο.



Σχήμα 3.1: Δίκτυο που υλοποιεί τον ZRP. Οι διακεκομμένες γραμμές περικλείουν τις ζώνες των S και D

Ζώνη Δρομολόγησης (Routing Zone)

Μια ζώνη δρομολόγησης καθορίζεται σαν ένα σύνολο από κόμβους, μέσα σε ένα συγκεκριμένο όριο απόστασης (που μετρείται σε hops) από τον κόμβο αναφοράς. Θα αναφερόμαστε σε αυτή την απόσταση σαν ακτίνα ζώνης. Στο παράδειγμα του σχήματος 3.1, όλοι οι κόμβοι (S, A, F, B, C, G και H) βρίσκονται το πολύ σε απόσταση 2 hops από τον κόμβο F. Παρόλο που ο κόμβος B έχει μια διαδρομή με απόσταση 3 hops από τον κόμβο F, εντούτοις βρίσκεται και αυτός στη

ζώνη του αφού η ελάχιστη απόσταση του από τον F είναι 2. Οι συνοριακοί ή περιφερειακοί κόμβοι είναι αυτοί που η ελάχιστη απόσταση τους από τον κόμβο αναφοράς είναι ακριβώς η ακτίνα της ζώνης. Στο σχήμα 3.1 οι κόμβοι B και F είναι περιφερειακοί κόμβοι του S.

Επανερχόμαστε στο δίκτυο του σχήματος 3.1. Έστω ότι ο κόμβος S θέλει να στείλει ένα πακέτο στον κόμβο D. Αφού ο D δεν είναι μέσα στη ζώνη δρομολόγησης του S, ο S στέλνει ένα Route Request μήνυμα στους συνοριακούς κόμβους B και F. Κάθε ένας από αυτούς ψάχνει να βρει αν ο D ανήκει στη δική του ζώνη δρομολόγησης. Κανένας από τους κόμβους B και F δεν βρίσκει τον ζητούμενο κόμβο μέσα στη ζώνη του και έτσι η αίτηση προωθείται στους αντίστοιχους συνοριακούς κόμβους. Ο F στέλνει την αίτηση στους S, B, C, και H, ενώ ο B στέλνει την αίτηση στους S, F, E, και G. Τώρα, ο ζητούμενος κόμβος D βρίσκεται στη ζώνη δρομολόγησης και του C και του E, έτσι δημιουργείται ένα μήνυμα Reply και στέλνεται πίσω στον αρχικό κόμβο S.

Για να αποφύγουμε την επιστροφή των αιτήσεων πίσω στους προηγούμενους κόμβους που ελέγχθηκαν, χρησιμοποιούμε μια λίστα (Processed Request List) στην οποία φυλάμε τις αιτήσεις που επεξεργαστήκαμε προηγουμένως. Αν ένας κόμβος λάβει μια αίτηση που έχει ήδη επεξεργαστεί, τότε απλά την αγνοεί.

Ιδιότητες

Ο ZRP είναι ένα πολύ ενδιαφέρον πρωτόκολλο δρομολόγησης που μπορεί να προσαρμόσει την λειτουργία του στις τρέχουσες συνθήκες λειτουργίας του δικτύου (μπορεί για παράδειγμα να αλλάξει την διάμετρο της ζώνης δρομολόγησης). Εντούτοις, αυτό δεν γίνεται δυναμικά αλλά η διάμετρος καθορίζεται από τον διαχειριστή του δικτύου ή τίθεται εκ των προτέρων σε μια προκαθορισμένη τιμή που δίνει ο κατασκευαστής του δικτύου. Η απόδοση του πρωτοκόλλου εξαρτάται σε μεγάλο βαθμό από αυτή την παράμετρο.

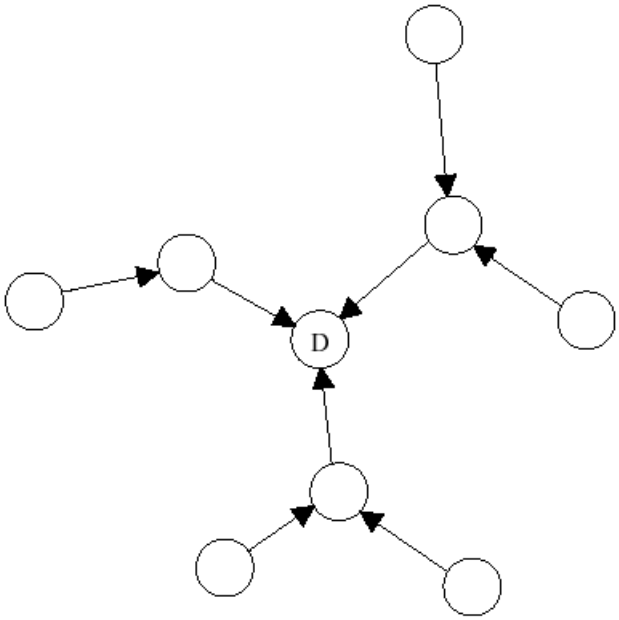
Αφού ο αλγόριθμος είναι συνδυασμός προ-δραστικών και μετα-δραστικών αλγορίθμων, το πρωτόκολλο εκμεταλλεύεται πλεονεκτήματα και από τα δύο σχήματα. Οι διαδρομές μπορεί να βρεθούν πολύ γρήγορα εντός της ζώνης δρομολόγησης, ενώ οι διαδρομές έξω από την ζώνη μπορούν να βρεθούν αποδοτικά χρησιμοποιώντας πακέτα Requests προς κατάλληλους κόμβους του δικτύου. Ένα πρόβλημα είναι πάντως το ότι δεν καθορίζεται το προ-δραστικό, εντός της ζώνης, πρωτόκολλο δρομολόγησης. Η χρησιμοποίηση διαφορετικών πρωτοκόλλων στην εσωτερική επικοινωνία μπορεί να σημαίνει ότι οι κόμβοι θα πρέπει να υποστηρίζουν διάφορα πρωτόκολλα. Αυτό δεν είναι επιθυμητό όταν οι χρήστες έχουν περιορισμένους πόρους και ιδιαίτερα μνήμη. Ίσως να είναι καλύτερα να χρησιμοποιείται το ίδιο πρωτόκολλο για την εντός της ζώνης επικοινωνία σε ολόκληρο το δίκτυο.

Ο ZRP επίσης περιορίζει την διάδοση πληροφορίας τοπολογικών αλλαγών στην γειτονιά όπου έγινε η αλλαγή και μόνο. Αυτό βρίσκεται σε αντίθεση με ένα πλήρως προ-δραστικό σχήμα στο οποίο οι πληροφορίες για αλλαγή τοπολογίας πλημμυρίζουν ολόκληρο το δίκτυο. Παρόλα αυτά μια αλλαγή στην τοπολογία μπορεί να επηρεάσει πολλές ζώνες δρομολόγησης.

3.5. Temporally-Ordered Routing Algorithm – TORA

Περιγραφή

Ο αλγόριθμος αυτός [17][18] είναι ένας κατανεμημένος αλγόριθμος δρομολόγησης. Ο βασικός αλγόριθμος πίσω από τον TORA είναι ένας της οικογένειας των αλγορίθμων αντίστροφων συνδέσεων (Link Reversal Algorithms). Ο TORA σχεδιάστηκε για την ελαχιστοποίηση των αναδράσεων σε τοπολογικές αλλαγές. Μια σημαντική έννοια στο σχεδιασμό του πρωτοκόλλου είναι ότι τα μηνύματα ελέγχου τυπικά περιορίζονται σε ένα μικρό αριθμό από γειτονικούς κόμβους. Εγγυάται επίσης ότι όλες οι διαδρομές είναι χωρίς κύκλους (μόνο προσωρινοί κύκλοι μπορεί να σχηματιστούν), και τυπικά παρέχει πολλαπλές διαδρομές για κάθε ζεύγος αποστολέα / παραλήπτη. Παρέχει μόνο το μηχανισμό δρομολόγησης και βασίζεται πάνω στο Internet MANET Encapsulation Protocol (IMEP [19]) για τις χαμηλότερου επιπέδου λειτουργίες.



Σχήμα 3.2: Κατευθυνόμενο Άκυκλο Γράφημα με ρίζα τον προορισμό D

Ο TORA μπορεί να διακριθεί σε τρεις βασικές λειτουργίες: δημιουργία διαδρομών, συντήρηση διαδρομών, και διαγραφή διαδρομών. Η δημιουργία διαδρομών βασικά αναθέτει κατευθύνσεις στις συνδέσεις ενός μη-κατευθυνόμενου δικτύου, ή σε τμήμα του δικτύου,

κτίζοντας ένα κατευθυνόμενο άκυκλο γράφημα με ρίζα στον προορισμό. Αυτό αναπαριστάνεται και στο σχήμα 3.2.

Ο TORA συσχετίζει ένα “ύψος” σε κάθε κόμβο του δικτύου. Όλα τα μηνύματα του δικτύου ρέουν προς τα κάτω, από ένα κόμβο με μεγαλύτερο ύψος σε ένα άλλο με μικρότερο ύψος. Οι διαδρομές ανακαλύπτονται χρησιμοποιώντας πακέτα αιτήσεως (QRY) και ενημέρωσης (UPD). Όταν ένας κόμβος χωρίς “προς τα κάτω” συνδέσεις (δηλαδή χωρίς γείτονα με μικρότερο ύψος) χρειάζεται μια διαδρομή σε ένα προορισμό κάνει broadcast ένα QRY πακέτο. Αυτό το πακέτο διαδίδεται μέσω του δικτύου μέχρι να φτάσει σε ένα κόμβο πού να έχει διαδρομή προς τον προορισμό ή στον ίδιο τον προορισμό. Ένας τέτοιος κόμβος κάνει τότε broadcast ένα UPD πακέτο που περιέχει το ύψος του κόμβου. Κάθε κόμβος που λαμβάνει αυτό το UPD πακέτο θα θέσει το ύψος του σε μια τιμή μεγαλύτερη από αυτή που περιέχεται στο πακέτο. Ακολούθως, ο κόμβος αυτός κάνει broadcast το δικό του UPD πακέτο. Το αποτέλεσμα αυτής της διαδικασίας θα είναι ένας αριθμός από κατευθυνόμενες συνδέσεις από τον αρχικοποιητή του QRY πακέτου στον προορισμό. Αυτή η διαδικασία μπορεί να οδηγήσεις σε πολλαπλές διαδρομές.

Η συντήρηση των διαδρομών γίνεται σαν αποτέλεσμα των τοπολογικών αλλαγών μέσα στο δίκτυο με ένα τρόπο τέτοιο ώστε οι διαδρομές προς τον προορισμό να επανεγκαθίστανται μέσα σε ένα πεπερασμένο χρονικό διάστημα. Μόλις ανιχνευθεί ένας διαμερισμός του δικτύου, όλες οι συνδέσεις μέσα στο τμήμα του δικτύου που έχει διαχωριστεί από τον προορισμό μαρκάρονται για διαγραφή σαν άκυρες. Η διαγραφή των διαδρομών γίνεται με τη χρήση μηνυμάτων τύπου CLR.

Ιδιότητες

Τα πρωτόκολλα που βρίσκονται πίσω από τον αλγόριθμο αντιστροφής συνδέσεων (link reversal algorithm) αντιδρούν στις αλλαγές των συνδέσεων μέσω ενός απλού τοπικού περάσματος του κατανεμημένου αλγόριθμου. Αυτό αποτρέπει τα CLR πακέτα από το να διαδίδονται πολύ μακριά μέσα στο δίκτυο. Μια σύγκριση που έχει γίνει από το CMU Monarch Project έχει δείξει ότι το πρόσθετο κόστος στον TORA είναι πολύ μεγάλο εξαιτίας της χρήσης του IMEP.

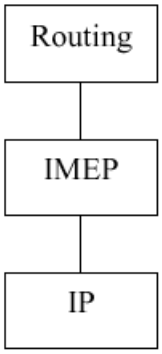
Το γράφημα έχει ρίζα τον προορισμό, ο οποίος έχει το μικρότερο ύψος. Εντούτοις, ο αρχικοποιητής του QRY μηνύματος δεν έχει απαραίτητα το μεγαλύτερο ύψος. Αυτό μπορεί να οδηγήσει στην κατάσταση όπου μπορεί να είναι δυνατές πολλές διαδρομές από τον αρχικοποιητή προς τον προορισμό, αλλά μόνο μια να ανακαλύπτεται. Ο λόγος που οδηγεί σε αυτό είναι ότι το ύψος βασίζεται αρχικά στην απόσταση σε hops από τον προορισμό.

3.6. Internet MANET Encapsulation Protocol – IMEP

Περιγραφή

Το IMEP [19] [20] είναι ένα πρωτόκολλο που σχεδιάστηκε για την υποστήριξη της λειτουργίας πολλών πρωτοκόλλων δρομολόγησης σε ad-hoc δίκτυα. Η ιδέα είναι να έχουμε ένα κοινό γενικό πρωτόκολλο το οποίο όλα τα πρωτόκολλα δρομολόγησης να μπορούν να χρησιμοποιήσουν, όπως φαίνεται και στο σχήμα 3.3. Εμπεριέχει πολλούς κοινούς μηχανισμούς τους οποίους μπορεί να χρειαστούν να ανώτερα στρώματα του πρωτοκόλλου. Αυτά περιλαμβάνουν:

- Αίσθηση Κατάστασης Σύνδεσης – Link Status Sensing
- Συγκέντρωση και Ομαδοποίηση Μηνυμάτων Ελέγχου – Control Message Aggregation and Encapsulation
- Αξιοπιστία Εκπομπής – Broadcast Reliability
- Ανάλυση Διεύθυνσης του Επιπέδου Δικτύου – Network-layer address resolution
- Διαδικασίες Αυθεντικοποίησης και Ασφάλειας – Hooks for Interrouter Security Authentication Procedures



Σχήμα 3.3: Το IMEP στη στοίβα των πρωτοκόλλων

Το IMEP επίσης παρέχει μια αρχιτεκτονική για την αναγνώριση των δρομολογητών του δικτύου, αναγνώριση του συστήματος διασύνδεσης, και της διευθυνσιοδότησης. Ο σκοπός του IMEP είναι να βελτιώσει την γενική απόδοση με την μείωση του αριθμού των μηνυμάτων ελέγχου, και επίσης να θέσει μια κοινή βάση στην λειτουργικότητα ενός ενιαίου, γενικού πρωτοκόλλου που να είναι χρήσιμο σε όλα τα πρωτόκολλα υψηλότερου επιπέδου.

Ιδιότητες

Η ιδέα ενός γενικού πρωτοκόλλου για την κοινή βάση χαρακτηριστικών και δυνατοτήτων είναι καλή, αλλά από άποψη απόδοσης έχει κάποια μειονεκτήματα. Προσθέτει ακόμη ένα επίπεδο στην στοίβα των πρωτοκόλλων. Όπως έχει δείξει το CMU Monarch Project [21], το IMEP παράγει μια μεγάλη ποσότητα από πρόσθετο κόστος, κυρίως επειδή ο μηχανισμός του IMEP

για την εύρεση γειτόνων παράγει τουλάχιστον ένα μήνυμα το δευτερόλεπτο, και επίσης λόγω της αξιόπιστης, σε-σειρά παράδοσης των πακέτων που παρέχει το ίδιο πρωτόκολλο.

3.7. Cluster Based Routing Protocol – CBRP

Περιγραφή

Η ιδέα πίσω από το CBRP [22] είναι η διαίρεση των κόμβων ενός ad-hoc δικτύου σε ένα αριθμό από επικαλυπτόμενα ή μη συμπλέγματα (clusters). Ένας κόμβος επιλέγεται σαν επικεφαλής του cluster σε κάθε ένα cluster. Αυτός ο επικεφαλής κόμβος διατηρεί πληροφορίες για τους συμμετέχοντες στο cluster. Οι εντός του cluster διαδρομές υπολογίζονται δυναμικά με χρήση της πληροφορίας για τους συμμετέχοντες στο cluster.

Το CBRP, όπως και το DSR, βασίζεται σε δρομολόγηση πηγής (source routing). Αυτό σημαίνει ότι οι διαδρομές μεταξύ των clusters υπολογίζονται πλημμυρίζοντας το δίκτυο με Route Request (RREQ) μηνύματα. Η διαφορά είναι ότι η δομή των clusters γενικά περιορίζει τον αριθμό των κόμβων που επηρεάζονται. Τα επίπεδα πρωτόκολλα δρομολόγησης, δηλαδή αυτά με ένα μόνο επίπεδο ιεραρχίας, μπορεί να μειονεκτούν λόγω υπερβολικού πρόσθετου κόστους όταν εξαπλώνονται πολύ.

Ο CBRP είναι όπως και τα άλλα πρωτόκολλα πλήρως κατανεμημένος. Αυτό είναι απαραίτητο λόγω της πολύ δυναμικής τοπολογίας των ad-hoc δικτύων. Επιπλέον, το πρωτόκολλο λαμβάνει υπόψη την ύπαρξη μονόδρομων συνδέσεων.

Αίσθηση Συνδέσεων – Link Sensing

Κάθε κόμβος στο CBRP γνωρίζει τις αμφίδρομες συνδέσεις προς τους γείτονες του, ως επίσης και τις μονο-κατευθυνόμενες συνδέσεις από τους γείτονες του προς τον ίδιο. Για την διαχείριση αυτού, κάθε κόμβος πρέπει να διατηρεί ένα Πίνακα Γειτόνων (Πίνακας 3.1).

Πίνακας 3.1: Πίνακας Γειτόνων

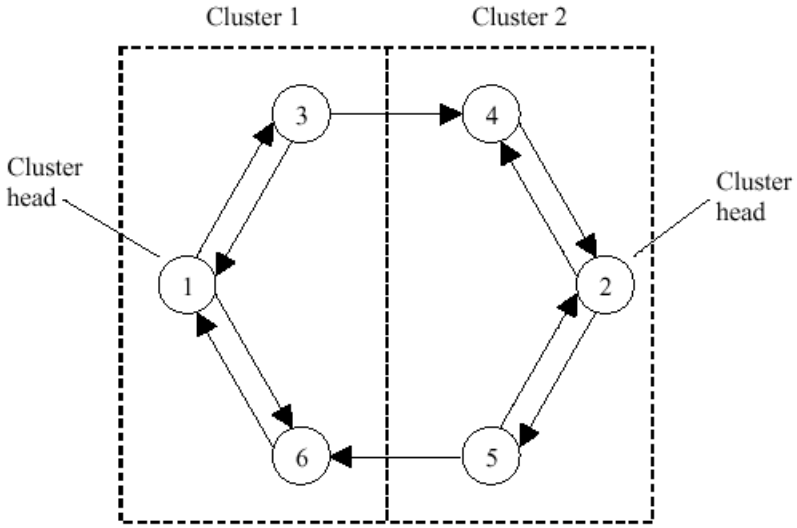
Ταυτότητα Γείτονα	Link Status	Role
Γείτονας 1	Αμφίδρομη / Μονόδρομη Σύνδεση σε μένα	Είναι ο 1 Κεφαλή του Cluster ή Μέλος
Γείτονας 2	Αμφίδρομη / Μονόδρομη Σύνδεση σε μένα	Είναι ο 2 Κεφαλή του Cluster ή Μέλος
…	...	
Γείτονας n	Αμφίδρομη / Μονόδρομη Σύνδεση σε μένα	Είναι ο n Κεφαλή του Cluster ή Μέλος

Κάθε κόμβος μεταδίδει περιοδικά τον Πίνακα Γειτόνων σε ένα hello μήνυμα. Το μήνυμα αυτό περιέχει την ταυτότητα του κόμβου, το ρόλο των κόμβων (κεφαλή του cluster, μέλος του

cluster, ή μη καθορισμένο), και τον Πίνακα Γειτόνων. Τα hello μηνύματα χρησιμοποιούνται για την ενημέρωση των Πινάκων Γειτόνων σε κάθε κόμβο. Εάν δεν ληφθεί κάποιο hello μήνυμα από ένα συγκεκριμένο κόμβο, τότε ο κόμβος αυτός αφαιρείται από τον πίνακα.

Συμπλέγματα – Clusters

Ο αλγόριθμος για τον σχηματισμό των clusters είναι πολύ απλός. Ο κόμβος με το χαμηλότερο ID εκλέγεται σαν κεφαλή του cluster. Οι κόμβοι χρησιμοποιούν την πληροφορία των hello μηνυμάτων για να αποφασίσουν κατά πόσο είναι ή όχι η κεφαλή του cluster. Ο κόμβος κεφαλή του cluster θεωρεί όλους τους κόμβους με τους οποίους έχει αμφίδρομες συνδέσεις σαν κόμβους μέλη του cluster. Ένας κόμβος θεωρεί τον εαυτό του σαν μέλος ενός cluster εάν έχει αμφίδρομη σύνδεση με την κεφαλή του cluster. Είναι δηλαδή δυνατό ένας κόμβος να ανήκει σε πολλά clusters.



Σχήμα 3.4: Αμφίδρομα συνδεδεμένα clusters

Τα clusters καθορίζονται από τον αντίστοιχο κόμβο κεφαλή, το οποίο σημαίνει ότι η κεφαλή του cluster πρέπει να αλλάζει όσο το δυνατό λιγότερο συχνά. Ως εκ τούτου, ο αλγόριθμος δεν είναι αυστηρά ένας αλγόριθμος clustering μικρότερου ID. Ένας κόμβος με μικρό ID ποτέ δεν αμφισβητεί ένα κόμβο που είναι ήδη κεφαλή του cluster. Μόνο όταν δύο κόμβοι που είναι κεφαλές σε clusters βρεθούν δίπλα-δίπλα, θα πρέπει ο ένας να χάσει τον ρόλο του σαν κεφαλή του cluster. Στο σχήμα 3.4 ο κόμβος 1 είναι η κεφαλή του cluster για το cluster 1 και ο κόμβος 2 είναι η κεφαλή του cluster για το cluster 2.

Δρομολόγηση

Η δρομολόγηση στο CBRP πρωτόκολλο βασίζεται στη δρομολόγηση πηγής και η εύρεση διαδρομών γίνεται με πλημμύρισμα του δικτύου με Route Requests (RREQ) μηνύματα. Η χρήση clusters όμως σημαίνει ότι επηρεάζονται λιγότεροι κόμβοι. Αυτό γίνεται επειδή μόνο οι κεφαλές των clusters πλημμυρίζονται. Εάν ο κόμβος X χρειάζεται μια διαδρομή προς τον κόμβο Y, ο κόμβος X στέλνει ένα RREQ μήνυμα που περιέχει την διαδρομή που αρχικά έχει μόνο αυτόν. Κάθε κόμβος που προωθεί αυτό το πακέτο θα προσθέτει το δικό του ID μέσα σε αυτό το μήνυμα. Κάθε κόμβος προωθεί ένα RREQ μήνυμα μόνο μια φορά και ποτέ δεν το προωθεί σε ένα κόμβο που βρίσκεται ήδη μέσα στη λίστα του μηνύματος.

Στο CBRP, ένα RREQ θα ακολουθεί πάντα μια διαδρομή με την ακόλουθη μορφή:

```
Source -> Cluster head -> Gateway -> Cluster head -> Gateway -> . . . -> Destination
```

Ένας κόμβος Gateway ενός cluster είναι ένας κόμβος που γνωρίζει ότι έχει μια αμφίδρομη ή μονόδρομη σύνδεση σε ένα κόμβο κάποιου άλλου cluster. Στο σχήμα 3.4, ο κόμβος 6 είναι ένας Gateway κόμβος για το cluster 1 και ο κόμβος 4 είναι ένας Gateway κόμβος για το cluster 2.

Ο αποστολέας στέλνει αρχικά το RREQ μήνυμα του στην κεφαλή του cluster. Κάθε κεφαλή του cluster στέλνει με την σειρά του το RREQ μήνυμα σε κάθε γείτονα με τον οποίο έχει αμφίδρομη σύνδεση, και ο οποίος δεν έχει ήδη εγγραφεί στη λίστα του μηνύματος που περιγράφει την αντίστοιχη διαδρομή. Κατ' ακρίβεια δεν είναι απαραίτητο να υπάρχει μια πραγματικά αμφίδρομη σύνδεση με ένα κόμβο του διπλανού cluster.Για παράδειγμα στο σχήμα 3.4, το cluster 1 έχει μια μονόδρομη σύνδεση προς το cluster 2 μέσω του κόμβου 3, και το cluster 2 έχει μια μονόδρομη σύνδεση προς το cluster 1 μέσω του κόμβου 5. Έτσι, τα δύο αυτά clusters είναι αμφίδρομα συνδεδεμένοι γείτονες. Αυτή η διαδικασία συνεχίζεται μέχρι να βρεθεί ο παραλήπτης, ή κάποιος άλλος κόμβος που να μπορεί να μας δώσει την διαδρομή προς αυτόν. Όταν το RREQ μήνυμα φτάσει στον προορισμό, ο κόμβος-παραλήπτης μπορεί να επιλέξει να απομνημονεύσει την αντίστροφη διαδρομή προς τον αποστολέα. Τότε αντιγράφει την διαδρομή που περιέχει το RREQ μήνυμα και το στέλνει πίσω στον αποστολέα.

Ιδιότητες

Αυτό το πρωτόκολλο έχει πολλές κοινές ιδιότητες με τα πρωτόκολλα που συζητήσαμε προηγουμένως. Η διαδικασίες του για εύρεση ή κατάργηση διαδρομής έχουν πολλά κοινά με τις αντίστοιχες διαδικασίες των DSR και AODV.

Η προσέγγιση διαίρεσης του δικτύου σε clusters είναι πιθανότατα πολύ καλή όταν έχουμε να αντιμετωπίσουμε μεγάλα ad-hoc δίκτυα. Η κλιμακωτή αυτή διαχείριση του δικτύου έχει το πλεονέκτημα του περιορισμού του αριθμού των μηνυμάτων που χρειάζεται να σταλούν. Ο CBRP έχει επίσης το πλεονέκτημα της εκμετάλλευσης των αμφίδρομων συνδέσεων. Μια ερώτηση που παραμένει αναπάντητη είναι το πόσο μεγάλα θα πρέπει να είναι τα clusters. Αυτή η απόφαση είναι κρίσιμη στην συμπεριφορά και απόδοση του δικτύου.

3.8. Location Aided Routing - LAR

Στην εργασία τους [23], οι Young-Bae Ko και Nitin Vaidya προσπαθούν να ξεπεράσουν τα προβλήματα που εμφανίζονται στην δρομολόγηση των ad-hoc κινητών δικτύων χρησιμοποιώντας πληροφορίες που χαρακτηρίζουν την θέση των κόμβων. Οι πληροφορίες αυτές παρέχονται συνήθως από ένα σύστημα τύπου GPS[22]. Χρησιμοποιώντας τις πληροφορίες για την θέση των κόμβων, το πρωτόκολλο Location-Aided Routing περιορίζει το ψάξιμο για μια νέα διαδρομή σε μια μικρότερη ζώνη αναζήτησης (request zone) του ad-hoc δικτύου. Σαν αποτέλεσμα αυτής της τροποποίησης, παρατηρούμε να υπάρχει σημαντική μείωση στον αριθμό των μηνυμάτων δρομολόγησης.

Πληροφόρηση Θέσης

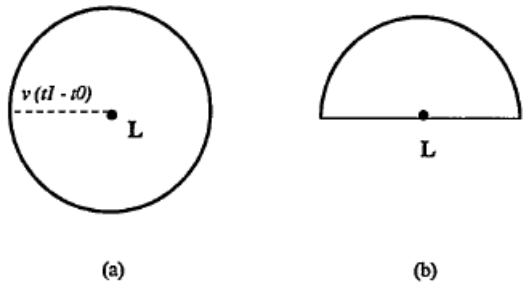
Η συγκεκριμένη προσέγγιση ονομάζεται Location-Aided Routing (LAR) αφού χρησιμοποιεί τη πληροφορία της θέσης των κόμβων για να μειώσει το πρόσθετο κόστος της δρομολόγησης. Η πληροφόρηση για την θέση του κόμβου μπορεί να παρέχεται από ένα GPS σύστημα. Με την χρήση του GPS, ένας χρήστης μπορεί να γνωρίζει τη φυσική του θέση. Παρόλο που τα συστήματα αυτά μας δίνουν τις πληροφορίες με κάποιο σχετικό σφάλμα, εντούτοις μπορούμε να θεωρήσουμε ότι μας δίνουν την ακριβή θέση των κόμβων.

Expected Zone και Request Zone

Το πρωτόκολλο LAR χρησιμοποιεί την έννοια των ζωνών Expected Zone και Request Zone. Η Expected Zone είναι η ζώνη στην οποία αναμένουμε να βρίσκεται ο κόμβος τον οποίο αναζητούμε. Αν για παράδειγμα γνωρίζουμε ότι τη χρονική στιγμή t_0 ο κόμβος R, βρισκόταν στη θέση (x, y) (θεωρώντας ότι ο χώρος μας είναι μια επίπεδη επιφάνεια), και έχει μέγιστη ταχύτητα V, τότε προφανώς η Expected Zone του R θα είναι ένας κύκλος με κέντρο την προηγούμενη

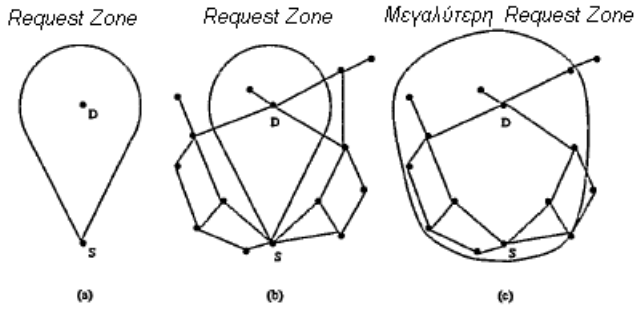
[22] Το GPS ή Global Positioning System είναι ένα σύστημα το οποίο μπορεί να μας δώσει τις ακριβείς συντεταγμένες μας στο χώρο σε οποιοδήποτε σημείο της γης και αν βρισκόμαστε. Για να το επιτύχει αυτό, μια συσκευή GPS λαμβάνει και χρησιμοποιεί τα σήματα που εκπέμπουν ειδικοί δορυφόροι που έχουν τεθεί σε τροχιά για αυτό το λόγο. Περισσότερες λεπτομέρειες μπορούν να αναζητηθούν στα [25, 26, 27, 28].
Τα σημερινά GPS συστήματα παρέχουν ακριβείς πληροφορίες για τη γεωγραφική θέση του χρήστη, τη ταχύτητα του, καθώς και την ακριβή παγκόσμια ώρα της θέσης του (Universal Time Coordinated) [24].

θέση του (x, y), και ακτίνα τη μέγιστη απόσταση που μπορεί να διανύσει στο χρόνο που μεσολάβησε (δηλαδή $r = (t_1 – t_0) V$). Επιπλέον, η ζώνη αυτή μπορεί να περιοριστεί περισσότερο αν γνωρίζουμε τη κατεύθυνση κίνησης του R. Για παράδειγμα, στο σχήμα 3.5 βλέπουμε πως μπορεί να είναι η Expected Zone ενός κόμβου L. Αν δεν γνωρίζουμε καμιά προηγούμενη θέση του R, τότε μπορούμε να θεωρήσουμε σαν Expected Zone ολόκληρο τον χώρο του ad-hoc δικτύου.



Σχήμα 3.5: Παραδείγματα Αναμενόμενης Ζώνης (Expected Zone) του L

Η Request Zone, είναι η ζώνη στην οποία θέλουμε να αναζητήσουμε τον κόμβο-προορισμό. Ο προτεινόμενος LAR αλγόριθμος χρησιμοποιεί τη μέθοδο του flooding για την εύρεση του ζητούμενου κόμβου. Όμως, υπάρχει μια σημαντική τροποποίηση σε αυτό τον αλγόριθμο που επιβάλλει να γίνεται το flooding εντός μιας καθορισμένης περιοχής, της Request Zone. Η ζώνη αυτή αποτελείται στην ουσία από τη ζώνη Expected Zone, και επιπλέον από τους κόμβους που μεσολαβούν μέχρι τους κόμβους της περιοχής αυτής. Αν, για παράδειγμα, ο αποστολέας S βρίσκεται εντός της Expected Zone του ζητούμενου κόμβου R, τότε οι δύο ζώνες ταυτίζονται (Expected Zone == Request Zone). Αντίθετα, ανάλογα με το πόσο γρήγορα επιθυμούμε να εντοπίσουμε το ζητούμενο κόμβο, μπορούμε να επιλέξουμε ένα διαφορετικό σχήμα Request Zone ζώνης, όπως φαίνεται στο σχήμα 3.6.



Σχήμα 3.6: Αιτούμενη Ζώνη (Request Zone): Μια ακμή συνδέει δύο κόμβους όταν είναι γειτονικοί

Συνήθως επιλέγουμε σαν Request Zone την Expected Zone, μαζί με τη διαδρομή που συνδέει τον αποστολέα S με τη παλιά θέση του παραλήπτη R.

Ουσιαστικά, το πρωτόκολλο αυτό μπορεί να χρησιμοποιηθεί με κάθε υπάρχον πρωτόκολλο για να βελτιώσει την απόδοση του. Αυτό το πετυχαίνει μειώνοντας κάθε φορά το "ωφέλιμο" δίκτυο σε ένα υποσύνολο του αρχικού δικτύου χρησιμοποιώντας πληροφορίες για τη φυσική θέση των κόμβων. Αυτή η τεχνική μειώνει το πρόσθετο κόστος του χρησιμοποιούμενου πρωτοκόλλου. Οι συγγραφείς του LAR χρησιμοποίησαν το πιο απλό πρωτόκολλο, δηλαδή το flooding. Τα πειραματικά τους αποτελέσματα έδειξαν ότι χρησιμοποιώντας αυτή την τεχνική μπορεί να υπάρξει σημαντική βελτίωση στην εκμετάλλευση των πόρων του δικτύου.

3.9. Σύγκριση των Υπαρχόντων Μοντέλων

Όλα τα πρωτόκολλα που έχουμε περιγράψει περιλαμβάνουν πολλά διαφορετικά χαρακτηριστικά και ιδιότητες. Πειραματικά, δεν έχει γίνει καμιά γενική σύγκριση, αφού κάτι τέτοιο θα ήταν πολύ δύσκολο. Επιπλέον, αφού πολλά πρωτόκολλα έχουν σχεδιαστεί για να ανταποκρίνονται καλά σε συγκεκριμένες συνθήκες, μια γενική σύγκριση θα "αδικούσε" κάποια από αυτά. Τέλος, υπάρχουν πρωτόκολλα που χρειάζονται ειδικές συνθήκες για να λειτουργήσουν, όπως για παράδειγμα το LAR που απαιτεί την ύπαρξη συσκευής GPS σε κάθε κόμβο.

Εδώ, επιχειρούμε μια σύγκριση των πρωτοκόλλων αυτών, σε επίπεδο χαρακτηριστικών και ιδιοτήτων. Στον πίνακα 3.2 συγκρίνουμε τα πρωτόκολλα που περιγράψαμε ως προς το αν πληρούν διάφορα χαρακτηριστικά όπως είναι η υποστήριξη πολλαπλών διαδρομών, η μεταδραστικότητα, η ασφάλεια, κλπ.

Όπως φαίνεται από αυτό τον πίνακα, κανένα από αυτά τα πρωτόκολλα δεν έχει υποστήριξη για εξοικονόμηση ενέργειας, ή εξασφάλιση στην ποιότητα των παρεχόμενων υπηρεσιών (QoS). Φυσικά, αυτή τη στιγμή διενεργείται έρευνα σε αυτή την περιοχή και αναμένουμε στο μέλλον τα περισσότερα πρωτόκολλα να περιλαμβάνουν και αυτά τα χαρακτηριστικά.

Όλα τα πρωτόκολλα είναι κατανεμημένα, και άρα κανένα από αυτά δεν εξαρτάται από ένα κεντρικοποιημένο κόμβο. Έτσι, το δίκτυο μας μπορεί εύκολα να προσαρμοστεί σε τοπολογικές μεταβολές, και να "αντισταθεί" σε περίπτωση βλάβης κάποιων κόμβων.

Ο DSDV είναι το μόνο προ-δραστικό πρωτόκολλο σε αυτή τη σύγκριση. Επίσης, είναι το πρωτόκολλο που έχει τα περισσότερα κοινά με τα παραδοσιακά πρωτόκολλα που χρησιμοποιούνται στα ενσύρματα δίκτυα. Η αρίθμηση των μηνυμάτων (sequence numbers) προστέθηκε για να εξασφαλίσουμε ότι οι διαδρομές μας δεν έχουν κύκλους. Ο DSDV είναι

πιθανότατα καλός σε δίκτυα που επιτρέπουν στο πρωτόκολλο να συγκλίνει γρήγορα. Αυτό πάντως, δεν σημαίνει ότι η κινητικότητα δεν μπορεί να είναι μεγάλη. Την ίδια γνώμη είχαν και οι κατασκευαστές του DSDV, και για αυτό το λόγο σχεδίασαν το AODV, το οποίο είναι η μετα-δραστική έκδοση του DSDV. Πρόσθεσαν επίσης την δυνατότητα multicast, η οποία αυξάνει την απόδοση σημαντικά όταν ένας κόμβος επικοινωνεί με πολλούς. Η μετα-δραστική προσέγγιση στον AODV έχει πολλές ομοιότητες με τη μετα-δραστική προσέγγιση στον DSR. Και οι δύο έχουν μια κατάσταση εύρεσης διαδρομής, στην οποία χρησιμοποιούν request μηνύματα για την εύρεση νέων διαδρομών. Η διαφορά τους είναι ότι ο DSR βασίζεται στην τεχνική του source-routing, και έτσι βρίσκει περισσότερες διαδρομές από τον AODV. Ο DSR έχει επίσης το πλεονέκτημα της υποστήριξης αμφίδρομων συνδέσεων. Παρόλα αυτά, ο DSR έχει ένα σημαντικό μειονέκτημα. Αυτό είναι ότι κάθε μήνυμα πρέπει να κουβαλάει τη διαδρομή μέχρι τον αρχικοποιητή του request μηνύματος. Αυτό μπορεί να είναι σημαντικό μειονέκτημα, ειδικά όταν θέλουμε να εξασφαλίσουμε την ποιότητα των υπηρεσιών του δικτύου, δηλαδή όταν το δίκτυο μας έχει την QoS ιδιότητα.

Ο ZRP και ο CBRP είναι δύο πολύ ενδιαφέρουσες προτάσεις που διαιρούν το δίκτυο σε πολλές ζώνες (clusters). Αυτή η προσέγγιση είναι πιθανότατα μια πολύ καλή λύση για πολύ μεγάλα δίκτυα. Μέσα στις ζώνες (ή clusters) τα πρωτόκολλα αυτά ακολουθούν ένα περισσότερο προ-δραστικό σχήμα. Αντίθετα, μεταξύ των ζωνών χρησιμοποιούν ένα περισσότερο μετα-δραστικό σχήμα, το οποίο έχει πολλές ομοιότητες με τη λειτουργία των AODV και DSR. Η διαφορά μεταξύ του ZRP και του CBRP είναι ο τρόπος με τον οποίο διαιρείται το δίκτυο. Στο ZRP όλες οι ζώνες είναι αλληλο-επικαλυπτόμενες, ενώ στο CBRP τα clusters μπορεί να είναι και αλληλο-επικαλυπτόμενα, και ανεξάρτητα. Τέλος, το LAR διαφέρει από τα υπόλοιπα πρωτόκολλα στο ότι χρησιμοποιεί επιπλέον τη πληροφορία της θέσης του κάθε κόμβου.

Κανένα από τα πρωτόκολλα που παρουσιάσαμε δεν είναι “προσαρμοζόμενο”, δηλαδή κανένα δεν μπορεί να πάρει κάποιες έξυπνες αποφάσεις δρομολόγησης στηριζόμενο στο φορτίο του δικτύου. Σαν κριτήριο επιλογής διαδρομής τα προτεινόμενα πρωτόκολλα χρησιμοποιούν μετρικές όπως είναι ο ελάχιστος αριθμός hops, και ο συντομότερος χρόνος απάντησης σε μια αίτηση. Αυτό μπορεί να οδηγήσει στη κατάσταση όπου όλο το φορτίο δρομολογείται μέσα από τον ίδιο κόμβο, παρόλο που υπάρχουν διαδρομές με λιγότερο φορτίο που θα μπορούσαν να εξυπηρετήσουν την επικοινωνία αποδοτικότερα.

	DSDV	AODV	DSR	ZRP	TORA/IMEP	CBRP	LAR
Loop-free	Ναι	Ναι	Ναι	Ναι	Όχι	Ναι	Ναι
Multiple Routes	Όχι	Όχι	Ναι	Όχι	Ναι	Ναι	Όχι
Distributed	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι	Ναι
Reactive	Όχι	Ναι	Ναι	Μερικώς	Ναι	Ναι	Ναι
Unidirectional	Όχι	Όχι	Ναι	Όχι	Όχι	Ναι	Όχι
QoS Support	Όχι	Όχι	Όχι	Όχι	Όχι	Όχι	Όχι
Multicast	Όχι	Ναι	Όχι	Όχι	Όχι	Όχι	Όχι
Security	Όχι	Όχι	Όχι	Όχι	Όχι	Όχι	Όχι
Power	Όχι	Όχι	Όχι	Όχι	Όχι	Όχι	Όχι
Periodic	Ναι	Ναι	Όχι	Ναι	Ναι (IMEP)	Ναι	Όχι
Utilizes GPS	Όχι	Όχι	Όχι	Όχι	Όχι	Όχι	Ναι

Πίνακας 3.2: Σύγκριση των Υπαρχόντων Πρωτοκόλλων Δρομολόγησης σε ad-hoc Δίκτυα

Επεξηγήσεις

Ακολούθως δίνουμε σύντομες επεξηγήσεις για το κάθε ένα από τα χαρακτηριστικά που συγκρίνουμε στον πίνακα 3.2. Περισσότερες πληροφορίες για αυτά τα χαρακτηριστικά αναφέρονται και στις προηγούμενες παραγράφους αυτού του κεφαλαίου.

***Loop-free**:* *οι σχηματιζόμενες διαδρομές δεν περιέχουν κύκλους*

***Multiple Routes**:* *υπάρχει υποστήριξη για περισσότερες από μια διαδρομές για κάθε σύνδεση*

***Distributed**:* *ο αλγόριθμος εκτελείται κατανεμημένα σε όλους τους κινητούς κόμβους*

***Reactive**:* *ο αλγόριθμος είναι μετα-δραστικός (και όχι προ-δραστικός)*

***Unidirectional**:* *ο αλγόριθμος μπορεί να λειτουργήσει σε δίκτυα με μονόδρομες συνδέσεις*

***QoS Support**:* *υπάρχει υποστήριξη για εξασφάλιση της ποιότητας της παρεχόμενης υπηρεσίας*

***Multicast**:* *υπάρχει υποστήριξη για μετάδοση σε πολλούς κόμβους*

***Security**:* *χρησιμοποιούνται μηχανισμοί που εισάγουν ασφάλεια στην επικοινωνία*

***Power**:* *χρησιμοποιείται μηχανισμός εξοικονόμησης ενέργειας*

***Periodic**:* *χρησιμοποιούν περιοδική μετάδοση μηνυμάτων ελέγχου*

***Utilizes GPS**:* *εκμεταλλεύεται το Global Positioning System για την εύρεση της ακριβούς τοποθεσίας των κινητών κόμβων*

4. Το Περιβάλλον Προσομοίωσης

Σε αυτό το κεφάλαιο μελετάμε τη λειτουργία του συστήματος που χρησιμοποιήσαμε για την λήψη μετρήσεων από τα προτεινόμενα πρωτόκολλα. Στην διεθνή ακαδημαϊκή κοινότητα έχει επικρατήσει να χρησιμοποιείται ο ns2[23] σαν περιβάλλον προσομοίωσης κινητών δικτύων. Το σύστημα αυτό προσομοιώνει πλήρως τη λειτουργία ενός δικτύου, μέχρι και σε φυσικό επίπεδο, δηλαδή επίπεδο MAC. Εμείς εισάγουμε ένα καινούριο περιβάλλον προσομοίωσης, το οποίο είναι απλούστερο από τον ns2, και πλεονεκτεί σε συγκεκριμένα σημεία που θεωρούμε απαραίτητα στην αποδοτική προσομοίωση συγκεκριμένων πρωτοκόλλων. Συγκεκριμένα, εκμεταλλευόμενοι τα αποτελέσματα των Micah Adler και Christian Scheideler στην εργασασία τους [5] (που έχουμε παρουσιάσει στο κεφάλαιο 2) μπορούμε να παρακάμψουμε το επίπεδο φυσικής διασύνδεσης (επίπεδο MAC) του δικτύου, και να υποθέσουμε ότι κάθε ζευγάρι κινητών κόμβων που βρίσκονται εντός ακτίνας επικοινωνίας μπορούν να επικοινωνήσουν εκμεταλλευόμενοι ένα κλάσμα του πραγματικού εύρους ζώνης επικοινωνίας (που δεν είναι μικρότερο από το ½ αυτού). Με βάση αυτή την υπόθεση, φτιάχνουμε ένα προσομοιωτή που παρακάμπτει το φυσικό επίπεδο, και αφοσιώνεται αποκλειστικά στο επίπεδο δικτύου, που είναι και το σημαντικότερο στην περίπτωση ανάπτυξης πρωτοκόλλων. Με αυτό το τρόπο, τα πειραματικά μας αποτελέσματα στηρίζονται σε προσομοιώσεις με μεγαλύτερο αριθμό κινητών κόμβων που επικοινωνούν για μεγαλύτερο χρονικό διάστημα.

4.1 Γενικά για Περιβάλλοντα Προσομοίωσης

Ο σχεδιασμός και η υλοποίηση ενός περιβάλλοντος προσομοίωσης είναι μια δύσκολη εργασία που απαιτεί λεπτομερειακή μελέτη των συνθηκών του περιβάλλοντος που προσομοιώνουμε. Ο σημαντικότερος (εξ' όσων γνωρίζουμε) προσομοιωτής μέχρι σήμερα είναι ο Network Simulator που αναπτύχθηκε (και συντηρείται) από κοινού από ερευνητές του University of California, Berkeley, του University of Southern California (USC), και των εργαστηρίων XeroxPARC και LBL. Όμως, ο προσομοιωτής αυτός είναι πολύ αναλυτικός για τις συνθήκες που μελετάμε και για αυτό το λόγο επιλέξαμε να αναπτύξουμε ένα καινούριο σύστημα προσομοίωσης.

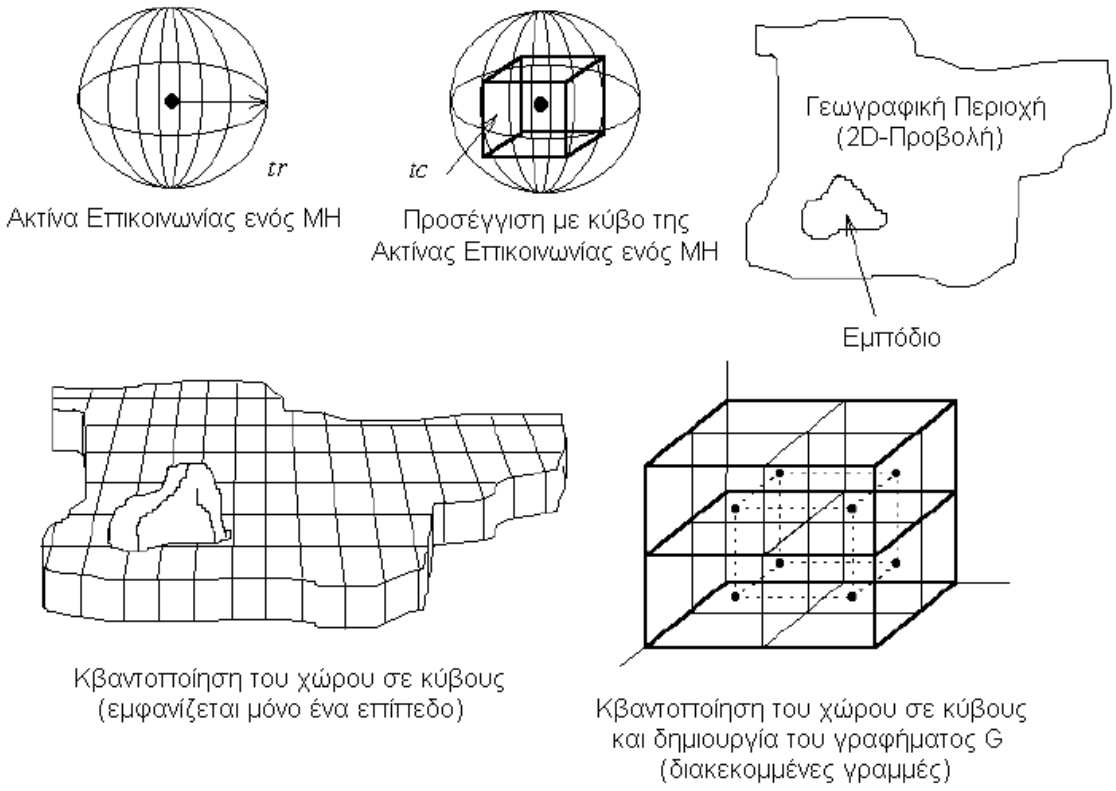
Στο προτεινόμενο σύστημα προσομοίωσης χρησιμοποιούμε ένα γράφημα (για παράδειγμα ένα δυσδιάστατο πλέγμα) το οποίο αποτελεί στην ουσία μια κβαντική μοντελοποίηση του χώρου του δικτύου. Η αρχική ιδέα ανήκει στους Χατζή, Πεντάρη, Σπυράκη, Ταμπακά, και Tan, οι οποίοι στην εργασία τους [3] έδειξαν πως μπορεί να μοντελοποιηθεί ο

[23] Ο network simulator είναι ένα γενικό περιβάλλον προσομοίωσης δικτύων που δημιουργήθηκε στα University of California, Berkeley και University of Southern California. Στην 2η έκδοση του περιλαμβάνονται τα απαραίτητα εργαλεία για την προσομοίωση ad-hoc κινητών δικτύων. Σχετικές πληροφορίες υπάρχουν στη διεύθυνση: http://www.isi.edu/nsnam/ns/ .

χώρος ενός ad-hoc δικτύου με ένα γράφημα. Στην επόμενη παράγραφο παρουσιάζουμε το μοντέλο αυτό που ουσιαστικά αποτελεί τον κορμό πάνω στον οποίο στηρίζεται η λειτουργία του προσομοιωτή μας.

4.2 Το Μοντέλο του Χώρου Κίνησης

Στην εργασία [3] παρουσιάζεται ένα πολύ ενδιαφέρον μοντέλο του χώρου κίνησης ενός ad-hoc κινητού δικτύου. Σε αυτό το μοντέλο χρησιμοποιείται ένα γράφημα του οποίου ο κάθε κόμβος συμβολίζει ένα κομμάτι της συνολικής περιοχής του δικτύου, ενώ οι ακμές του συνδέουν τους κόμβους εκείνους που συμβολίζουν γειτονικές περιοχές (δηλαδή περιοχές στις οποίες οι χρήστες μπορούν άμεσα να μετακινηθούν). Αυτό το μοντέλο είναι πολύ χρήσιμο και όπως θα δούμε αποτελεί τη βασική ιδέα πάνω στην οποία στηρίζεται η λειτουργία του προσομοιωτή που αναπτύξαμε.



Σχήμα 4.1: Η κατασκευή του G όταν το *tc* είναι κύβος

Στη πραγματικότητα, οι πιο συνηθισμένες εφαρμογές είναι αυτές στις οποίες οι χρήστες του δικτύου κινούνται σε ένα χώρο δύο διαστάσεων, δηλαδή πάνω στην επιφάνεια μιας γεωγραφικής περιοχής. Ωστόσο, το μοντέλο που παρουσιάζουμε είναι αρκετά γενικό και προσαρμόζεται και σε περιπτώσεις όπου οι χρήστες κινούνται σε χώρο τριών διαστάσεων.

Υποθέτουμε ότι στον τρισδιάστατο χώρο ένας κινητός χρήστης έχει ακτίνα επικοινωνίας που αντιστοιχεί σε μια σφαίρα *tr* με κέντρο τον εαυτό του. Αυτό σημαίνει ότι κάθε μήνυμα που μεταδίδεται από αυτόν το χρήστη μπορεί να ληφθεί από κάθε άλλο χρήστη που βρίσκεται εντός της σφαίρας *tr*. Προσεγγίζουμε αυτή την σφαίρα με ένα κανονικό πολύεδρο *tc* με όγκο V(*tc*), όπου V(*tc*) < V(*tr*). Το μέγεθος του πολυέδρου *tc* μπορεί να επιλεγεί με τέτοιο τρόπο έτσι ώστε να είναι ο μέγιστος δυνατός ακέραιος που ικανοποιεί την σχέση V(*tc*) ≤ V(*tr*), και επιπλέον όταν ένας χρήστης μεταδίδει ένα μήνυμα εντός του *tc*, το μήνυμα αυτό να λαμβάνεται από κάθε άλλο χρήστη εντός του *tc*. Για παράδειγμα, στο σχήμα 4.1 το *tc* είναι ένας κύβος. Στο συγκεκριμένο παράδειγμα, δοθέντος ότι οι κόμβοι κινούνται μέσα στο χώρο S, το S διαιρείται σε συνεχόμενους κόμβους όγκου V(*tc*) (βλέπε σχήμα 4.1). Το γράφημα G(V, E), |V| = *n*, |E| = *e*, που αντιστοιχεί στην κβαντοποίηση του χώρου S δημιουργείται με τον ακόλουθο τρόπο: μια κορυφή u ∈ V αντιπροσωπεύει ένα κύβο όγκου V(*tc*). Μια ακμή (u, v) ∈ E εάν οι αντίστοιχοι κύβοι μοιράζονται μια επιφάνεια.

Ένας κινητός κόμβος μπορεί να κινηθεί οπουδήποτε μέσα στο S αλλά σε κάθε χρονική στιγμή βρίσκεται εντός ενός συγκεκριμένου κύβου *tc*. Έτσι, σε κάθε χρονική στιγμή κάθε κόμβος βρίσκεται σε μια ακριβώς κορυφή του γραφήματος κίνησης G. Εάν ο κόμβος γνωρίζει την απόσταση που έχει διανύσει, μπορεί να υπολογίσει την αλλαγή της θέσης του από ένα κύβο σε ένα άλλο (δηλαδή από μια κορυφή του G σε μια άλλη). Εάν ένας κόμβος βρίσκεται σε μια κορυφή u, και στείλει ένα μήνυμα, αυτό το μήνυμα θα ληφθεί από όλους τους άλλους κόμβους που βρίσκονται την ίδια χρονική στιγμή πάνω στην ίδια κορυφή[24]. Κατά ακρίβεια, ο αριθμός των κορυφών *n* προσεγγίζει την αναλογία μεταξύ του όγκου του χώρου S, V(S), και του χώρου που καλύπτεται από την ακτίνα επικοινωνίας ενός κινητού κόμβου V(*tr*). Στην ακραία περίπτωση όπου V(S) ≈ V(*tr*) (η ακτίνα μετάδοσης των κόμβων είναι κατά προσέγγιση ο χώρος μέσα στον οποίο κινούνται), τότε *n* = 1. Δοθέντος της ακτίνας μετάδοσης *tr*, το *n* εξαρτάται γραμμικά από τον όγκο του χώρου S, ανεξάρτητα από την παράμετρο *tc*, και $n = O(\frac{V(S)}{V(tr)})$. Αφού οι ακμές του G αντιπροσωπεύουν γειτονικά πολύεδρα, κάθε κορυφή συνδέεται με ένα σταθερό αριθμό από γείτονες, που σημαίνει ότι $e = O(n)$. Στο παράδειγμα μας, όπου το *tc* είναι ένας κύβος, το G έχει μέγιστο βαθμό έξι και $e \leq 6n$.

Το μοντέλο αυτό υλοποιείται στην πράξη από το σύστημα προσομοίωσης που παρουσιάζουμε στη συνέχεια. Το γράφημα που μοντελοποιεί τον χώρο του δικτύου μας, το ονομάζουμε *γράφημα κίνησης* (*motion graph*).

[24] Όπως θα δούμε και στη συνέχεια, στο προσομοιωτή μας κάνουμε μια τροποποίηση σε αυτό το μοντέλο. Συγκεκριμένα, επιτρέπουμε στους κινητούς κόμβους να μεταδίδουν μηνύματα σε μεγαλύτερη απόσταση, και έτσι το μεταδιδόμενο μήνυμα να φτάνει σε κάθε κινητό κόμβο που βρίσκεται σε κορυφή που απέχει το πολύ K hops απο την κορυφή του κόμβου μετάδοσης, όπου K η ακτίνα μετάδοσης.

4.3 Το Προτεινόμενο Περιβάλλον Προσομοίωσης

Ο προσομοιωτής που αναπτύξαμε υλοποιήθηκε σε γλώσσα προγραμματισμού C++, ενώ χρησιμοποιήθηκε επιπλέον η βιβλιοθήκη αποδοτικών δομών δεδομένων και αλγορίθμων LEDA. Στην ανάπτυξη του συστήματος χρησιμοποιήσαμε αντικειμενοστραφή προγραμματισμό έτσι ώστε ο κώδικας να είναι όσο το δυνατό πιο ευανάγνωστος και εύκολα συντηρήσιμος. Επιπλέον, κατά την ανάπτυξη του προσομοιωτή προσπαθήσαμε να βρούμε την ιδανική ισορροπία μεταξύ αφενός της λειτουργικότητας και της ευκολίας προγραμματισμού του συστήματος, και αφετέρου της εκμετάλλευσης πόρων του συστήματος (κυρίως μνήμης και χρόνου εκτέλεσης). Η ανάπτυξη του συστήματος έγινε σταδιακά, ελέγχοντας κάθε φορά την ορθότητα της λειτουργίας του. Φυσικά, δεν αποκλείουμε τη πιθανότητα να υπάρχουν σφάλματα στον κώδικα, αλλά πιστεύουμε ότι δεν υπάρχει κάποιο σημαντικό σφάλμα που να καθιστά τα αποτελέσματα μας μη αξιόπιστα. Τέλος, θα πρέπει να επισημάνουμε ότι οι απαιτήσεις του συστήματος σε πόρους (μνήμη και χρόνο εκτέλεσης) εξαρτάται σε μεγάλο βαθμό από το συγκεκριμένο πρωτόκολλο που υλοποιούμε, καθώς επίσης και από τον τρόπο με τον οποίο το προγραμματίζουμε.

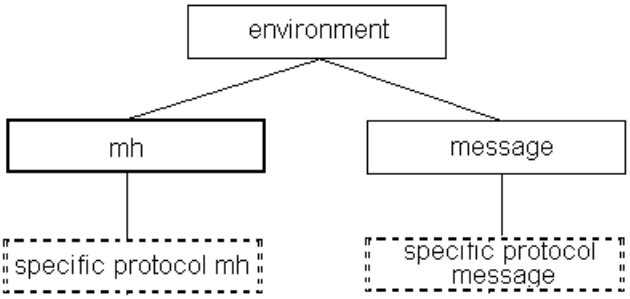
Η βασική κλάση που χρησιμοποιείται στον προσομοιωτή είναι η *environment,* η οποία και μοντελοποιεί τον χώρο (ή την περιοχή) του δικτύου, καθώς και τις διάφορες λειτουργίες που αναμένουμε να υποστηρίζει το δίκτυο. Η κλάση αυτή είναι, για παράδειγμα, εφοδιασμένη με μεθόδους που υποστηρίζουν τη μετάδοση μηνυμάτων, τη μετακίνηση κόμβων, κλπ.

Ακολούθως, έχουμε την κλάση *mh* η οποία μοντελοποιεί έναν κινητό κόμβο (*mobile host*). Η κλάση αυτή περιλαμβάνει μεθόδους για τις βασικότερες λειτουργίες που μπορεί να εκτελέσει ένας κινητός κόμβος, όπως είναι η μετακίνηση του, η δημιουργία ενός μηνύματος, ή η αποστολή και λήψη μηνυμάτων.

Η τρίτη σημαντική κλάση που περιέχει ο προσομοιωτής είναι αυτή των μηνυμάτων, την οποία ονομάζουμε *message*. Η κλάση αυτή αποτελεί ένα στοιχειώδες μήνυμα που περιέχει τις ελάχιστες απαιτούμενες πληροφορίες για την μετάδοση του καθώς και επιπλέον πληροφορίες που χρησιμοποιούνται από τον προσομοιωτή για την λήψη μετρήσεων. Η κλάση αυτή μπορεί να χρησιμοποιηθεί και αυτούσια (χωρίς δηλαδή να επεκταθεί με επιπλέον χαρακτηριστικά μέσω κληρονομικότητας), εάν αυτό δεν είναι αντίθετο με τις απαιτήσεις του πρωτοκόλλου.

Οι δύο προηγούμενες κλάσεις, δηλαδή η *mh* και η *message,* έχουν κατασκευαστεί έτσι ώστε να μπορούν εύκολα να επεκτείνονται (με τη χρήση της κληρονομικότητας που μας προσφέρει ο αντικειμενοστραφής προγραμματισμός) για να υλοποιήσουν τις απαραίτητες λειτουργίες κατά τη μοντελοποίηση συγκεκριμένων πρωτοκόλλων. Ειδικότερα, η κλάση *mh* περιέχει την εικονική μέθοδο *executeProtocol()* η οποία και είναι απαραίτητο να καθοριστεί σε

κάθε πρωτόκολλο που υλοποιούμε. Στο Παράρτημα Γ παρουσιάζουμε το τρόπο με τον οποίο η κλάση *mh* επεκτείνεται έτσι ώστε να μοντελοποιεί κόμβους που εκτελούν διάφορες λειτουργίες σε διάφορα πρωτόκολλα. Πιο συγκεκριμένα, παρουσιάζουμε τη γενική κλάση ενός mh, και ακολούθως τον τρόπο με τον οποίο επεκτείνουμε αυτή τη κλάση για να υλοποιήσουμε τους τρεις τύπους κινητών κόμβων που χρησιμοποιούμε στις μετρήσεις μας. Στο σχήμα 4.2 βλέπουμε την ιεραρχική δομή των βασικών κλάσεων του προσομοιωτή. Η κλάση *mh* διακρίνεται με έντονο πλαίσιο αφού, όπως θα δούμε στη συνέχεια, δεν μπορεί να χρησιμοποιηθεί εάν δεν υλοποιήσουμε πρώτα την *executeProtocol()* μέθοδο της. Με διακεκομμένες γραμμές σημειώνουμε τις παράγωγες κλάσεις που προέρχονται από τις βασικές κλάσεις *mh* και *message*, και χρησιμοποιούνται για την υλοποίηση κάποιου συγκεκριμένου πρωτοκόλλου.

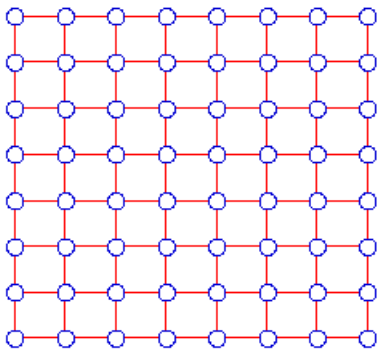


Σχήμα 4.2: Η Ιεραρχία των κλάσεων του Προσομοιωτή

Στη συνέχεια περιγράφουμε πιο αναλυτικά τη δομή και τη λειτουργία κάθε μιας από τις τρεις κλάσεις που έχουμε ήδη αναφέρει.

Κλάση *Environment*

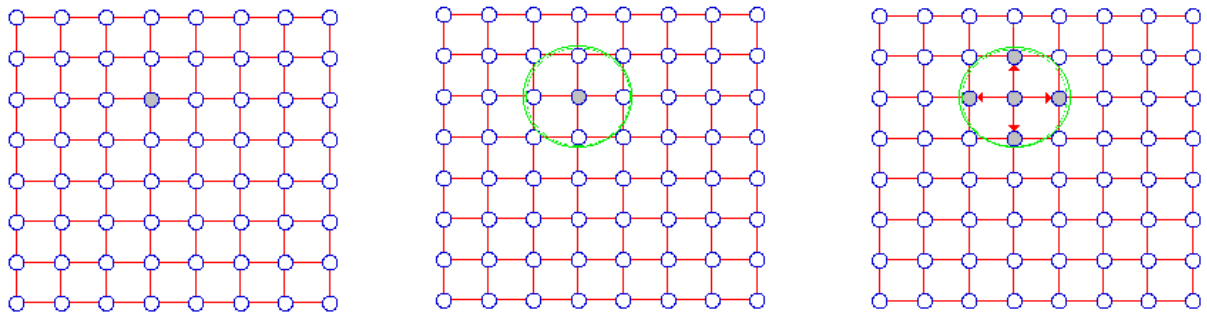
Η κλάση αυτή, όπως έχουμε ήδη αναφέρει, αποτελεί την βασική κλάση του προσομοιωτή. Η ουσιαστικότερη της λειτουργία είναι η μοντελοποίηση του χώρου του δικτύου σε ένα γράφημα, με τον τρόπο που έχουμε ήδη περιγράψει στη προηγούμενη παράγραφο. Η βασικότερη δομή αυτής της κλάσης είναι ένα γράφημα, οι κορυφές του οποίου αναπαριστούν από ένα τμήμα της ολικής περιοχής. Γειτονικές περιοχές συνδέονται με μια ακμή. Στο σχήμα 4.3 φαίνεται ένα τέτοιο απλό γράφημα με δομή πλέγματος και μέγεθος 8Χ8. Αυτό το γράφημα το ονομάζουμε γράφημα κίνησης (*motion graph*).

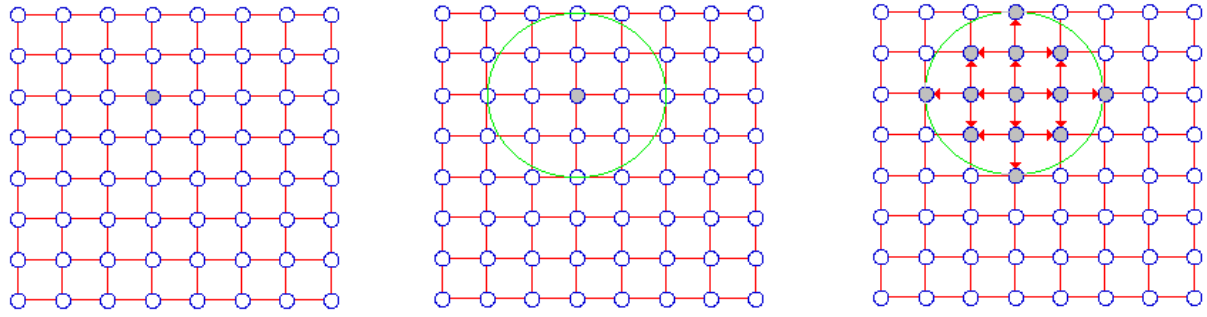


Σχήμα 4.3: Γράφημα κίνησης με δομή πλέγματος και μέγεθος 8 Χ 8

Όταν δημιουργείται κάποιος κινητός κόμβος, τότε τοποθετείται σε κάποια από τις κορυφές του γραφήματος κίνησης. Η επιλογή της θέσης του γίνεται με τυχαίο τρόπο, εκτός και αν το συγκεκριμένο πρωτόκολλο επιβάλλει κάτι διαφορετικό.

Η κάθε κορυφή του γραφήματος κίνησης περιέχει έναν σωρό μέσα στον οποίο αποθηκεύονται τα μηνύματα που μεταδίδονται. Το μέγεθος αυτού του σωρού καθορίζει ουσιαστικά το εύρος ζώνης επικοινωνίας του δικτύου. Όταν κάποιος κινητός κόμβος μεταδίδει ένα μήνυμα, τότε το μήνυμα αυτό αντιγράφεται στους σωρούς όλων των κορυφών του γραφήματος κίνησης που βρίσκονται σε απόσταση μικρότερη ή ίση από την ακτίνα μετάδοσης των κινητών κόμβων.





Σχήμα 4.4: Παράδειγμα Λειτουργίας του Προσομοιωτή κατά την Μετάδοση

Ένας κόμβος που βρίσκεται στην γκρίζα κορυφή θέλει να μεταδώσει ένα μήνυμα με παραλήπτες όλους τους κόμβους που βρίσκονται στις γειτονικές κορυφές. Στην πρώτη περίπτωση η ακτίνα μετάδοσης στο δίκτυο είναι 1 ενώ στην δεύτερη περίπτωση η ακτίνα μετάδοσης είναι 2

Στο σχήμα 4.2 βλέπουμε τη διαδικασία μετάδοσης ενός μηνύματος στην περίπτωση που το δίκτυο μας έχει ακτίνα μετάδοσης αρχικά 1 και ακολούθως 2. Όπως φαίνεται και στο σχήμα, το μήνυμα αντιγράφεται στους σωρούς των κόμβων που βρίσκονται εντός ακτίνας μετάδοσης και διακρίνονται με γκρίζο χρώμα. Ο υπολογισμός των γειτονικών κόμβων, για κάθε κόμβο του γραφήματος κίνησης, γίνεται κατά την αρχικοποίηση του συστήματος έτσι ώστε να αποφεύγεται

ο επαναλαμβανόμενος υπολογισμός τους, και άρα η αντίστοιχη καθυστέρηση, κατά τη διάρκεια της προσομοίωσης.

Με παρόμοιο τρόπο λειτουργεί και το μοντέλο κίνησης των κόμβων. Συγκεκριμένα, στους κόμβους παρέχεται η δυνατότητα τυχαίας μετακίνησης κατά την οποία το σύστημα επιλέγει με τυχαίο τρόπο μια γειτονική ακμή διαμέσου της οποίας ο χρήστης μεταφέρεται στην αντίστοιχη ακμή του γραφήματος κίνησης. Με αυτό το τρόπο υπάρχει η δυνατότητα οι χρήστες να εκτελούν τυχαίους περιπάτους. Φυσικά, η κίνηση του κάθε χρήστη μπορεί να καθορίζεται και ντετερμινιστικά, εάν αυτό απαιτείται από το πρωτόκολλο. Για παράδειγμα, θα δούμε στη συνέχεια ότι στο πρωτόκολλο του φιδιού μόνο ένας κόμβος της υποστήριξης (που ονομάζουμε αρχηγό) εκτελεί τυχαίο περίπατο, ενώ η κίνηση των υπολοίπων μελών της υποστήριξης καθορίζεται από την κίνηση του αρχηγού.

Επειδή το σύστημα μας είναι διακριτού χρόνου, και οι κινητοί κόμβοι που προσομοιώνουμε δεν μπορούν να εκτελούν το πρωτόκολλο τους ταυτόχρονα, καταφεύγουμε σε μια τεχνική η οποία μοντελοποιεί με “δίκαιο” τρόπο τη λειτουργία τους. Συγκεκριμένα, σε κάθε διακριτό βήμα του προσομοιωτή, κάθε κινητός κόμβος έχει την δυνατότητα να εκτελέσει ένα βήμα από το πρωτόκολλο του. Συγκεκριμένα, έχει τη δυνατότητα να διαβάσει κάθε μήνυμα που έχει λάβει κατά το προηγούμενο βήμα εκτέλεσης, να εκτελέσει όποιες εσωτερικές λειτουργίες απαιτούνται από το πρωτόκολλο, και τέλος να μεταδώσει κάποια μηνύματα. Αυτή η τελευταία ενέργεια όμως, κρύβει μια παγίδα.

Επειδή κατά τη μετάδοση μηνυμάτων, τα μηνύματα αποθηκεύονται προσωρινά στον σωρό της κορυφής (που μοντελοποιεί στην ουσία το μέσο μετάδοσης), υπάρχει ο κίνδυνος να έχουμε υπερχείλιση του σωρού. Όπως έχουμε ήδη αναφέρει, ο σωρός μπορεί να επιλεγεί να έχει ένα συγκεκριμένο μέγεθος, μοντελοποιώντας με αυτό το τρόπο το εύρος ζώνης του μέσου μετάδοσης. Αν η σειρά εκτέλεσης του πρωτοκόλλου μεταξύ των κινητών κόμβων είναι πάντα η ίδια, τότε υπάρχει ο κίνδυνος το σύστημα μας να μην είναι δίκαιο και συγκεκριμένοι κόμβοι να μην μεταδίδουν ποτέ. Αυτό θα συμβαίνει όταν οι κόμβοι που εκτελούν πρώτοι το πρωτόκολλο τους μεταδίδουν πολλά μηνύματα προκαλώντας το γέμισμα των σωρών. Σε αυτή την περίπτωση οι κόμβοι που εκτελούν το πρωτόκολλο τους τελευταίοι δεν μπορούν να μεταδώσουν αφού βρίσκουν τους σωρούς διαρκώς γεμάτους.

Για την επίλυση αυτού του προβλήματος εφαρμόσαμε τον ακόλουθο μηχανισμό. Σε μια λίστα αποθηκεύουμε τις ταυτότητες όλων των κινητών κόμβων. Πριν την εφαρμογή του κάθε βήματος προσομοίωσης επανα-διατάζουμε τα περιεχόμενα της λίστας με τυχαίο τρόπο. Η εκτέλεση του πρωτοκόλλου από τους κόμβους γίνεται σε κάθε βήμα με την σειρά που καθορίζεται από αυτή τη λίστα. Με αυτή τη μέθοδο εξασφαλίζουμε ότι το μέσο μετάδοσης

μοιράζεται ανάμεσα στους κινητούς κόμβους με δίκαιο τρόπο, αφού όλοι έχουν τις ίδιες πιθανότητες να μεταδώσουν ένα μήνυμα με επιτυχία.

Μια εξίσου σημαντική λειτουργία της κλάσης *environment* είναι η διατήρηση και αποθήκευση μετρήσεων κατά τη βηματική εκτέλεση της προσομοίωσης. Έχουμε επίσης τη δυνατότητα να επιλέξουμε τα αποτελέσματα αυτά να παρουσιάζονται στο χρήστη ανά τακτά χρονικά διαστήματα (για παράδειγμα κάθε 10 βήματα).

Κλάση *mh*

Η κλάση *mh* μοντελοποιεί μια οντότητα κινητού κόμβου. Η σημαντικότερη μέθοδος που περιέχει είναι η *executeProtocol()*, η οποία είναι μια *εικονική (virtual)* μέθοδος. Αυτό σημαίνει ότι δεν μπορούμε να υλοποιήσουμε ένα αντικείμενο τύπου *mh* εάν δεν έχουμε προηγουμένως ορίσει τη μέθοδο αυτή. Σε αυτή τη μέθοδο υλοποιούμε ουσιαστικά το πρωτόκολλο που θέλουμε να εκτελούν οι συγκεκριμένοι κόμβοι. Στο Παράρτημα παρουσιάζουμε το κώδικα που γράψαμε για την υλοποίηση τριών διαφορετικών κόμβων. Συγκεκριμένα παρουσιάζουμε το κώδικα που υλοποιεί κόμβους τύπου sender-receiver (απλοί κόμβοι που δημιουργούν και δέχονται μηνύματα), τύπου snake, και τύπου runner (οι οποίοι υλοποιούν τα αντίστοιχα πρωτόκολλα).

Φυσικά, η κλάση αυτή παρέχει και πολλές άλλες μεθόδους που μπορούν να χρησιμοποιηθούν κατά την υλοποίηση κάποιου πρωτοκόλλου. Για παράδειγμα υπάρχει η μέθοδος *TRANSMIT* η οποία χρησιμοποιείται για τη μετάδοση κάποιου μηνύματος, η μέθοδος *MOVE_RANDOMLY* η οποία χρησιμοποιείται για να μετακινήσει τον κόμβο σε κάποια άλλη, τυχαία επιλεγμένη, θέση, και η *GET_NEIGHBOURS* η οποία χρησιμοποιείται για να μας δώσει μια λίστα με τους γειτονικούς μας κόμβους. Γενικά, η κλάση αυτή προσφέρει όλες εκείνες τις λειτουργίες που θεωρούμε ότι θα πρέπει να παρέχονται στους σχεδιαστές των πρωτοκόλλων. Για παράδειγμα, αν θεωρήσουμε ότι οι κόμβοι μας διαθέτουν σύστημα GPS, που τους δίνει τη δυνατότητα να γνωρίζουν τη θέση τους στο χώρο, τότε θα πρέπει απλά να προσθέσουμε μια μέθοδο *GET_GPS_POS* η οποία και θα μας επιστρέφει την αντίστοιχη πληροφορία. Κατά ακρίβεια, αυτή είναι μια χρήσιμη μέθοδος που προς το παρόν όμως δεν παρέχεται από το σύστημα μας.

Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι για το συγκεκριμένο σύστημα προσομοίωσης αναφερόμαστε και στο Παράρτημα, όπου δίνουμε διάφορες χρήσιμες πληροφορίες για τη χρήση του. Επιπλέον στο Παράρτημα υπάρχουν και παραδείγματα κώδικα υλοποίησης των βασικών πρωτοκόλλων του Κεφαλαίου 5.

Κλάση *message*

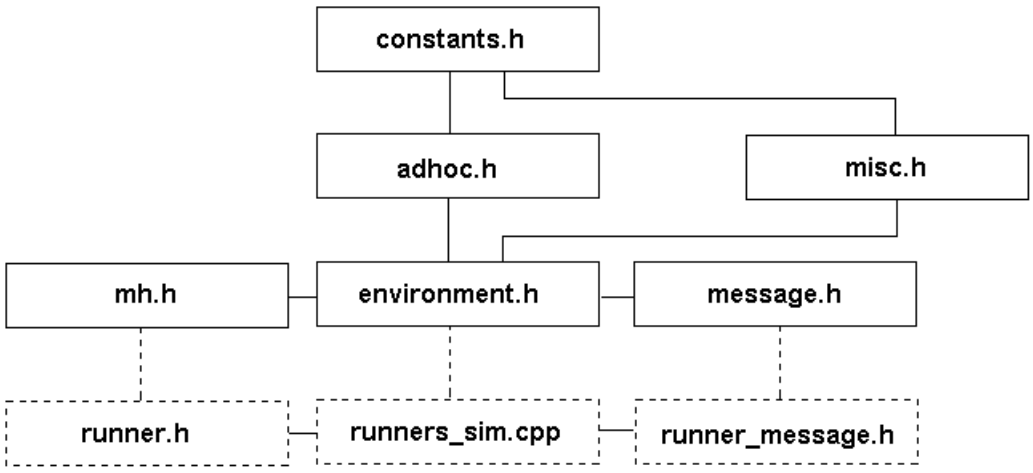
Η Τρίτη κλάση που μελετάμε είναι η κλάση *message*. Η κλάση αυτή μοντελοποιεί ένα απλό μήνυμα, το οποίο και δεν περιέχει οποιαδήποτε πληροφορία πλην των απαιτούμενων για την επικοινωνία. Φυσικά, για το κάθε πρωτόκολλο που υλοποιούμε μπορούμε να επεκτείνουμε τη συγκεκριμένη κλάση και να φτιάξουμε μηνύματα με τη δομή που απαιτεί το κάθε πρωτόκολλο.

Η δομή αυτής της κλάσης είναι αρκετά απλή, και οι περισσότερες πληροφορίες που περικλείει είναι αυτές που χρησιμοποιούνται για λήψη μετρήσεων κατά την προσομοίωση. Για παράδειγμα, το κάθε μήνυμα περιέχει πληροφορίες για το ποιος κόμβος το κατασκεύασε και πότε, πόσα hops έχει διαγράψει μέχρι στιγμής, κλπ.

Η επέκταση αυτής της κλάσης για την υλοποίηση μηνυμάτων με συγκεκριμένη μορφή γίνεται και πάλι με χρήση κληρονομικότητας. Οι πληροφορίες που αναφέραμε εξακολουθούν να υπάρχουν και φυσικά χρησιμοποιούνται και πάλι από το σύστημα για τη λήψη μετρήσεων.

Η Λειτουργία του Προσομοιωτή

Μετά τη περιγραφή των τριών βασικών κλάσεων του συστήματος είμαστε έτοιμοι να περιγράψουμε τα βήματα που ακολουθούνται κατά τη διαδικασία της προσομοίωσης.



Σχήμα 4.5: Τα Αρχεία που Συνθέτουν τον Προσομοιωτή και η μεταξύ τους επικοινωνία

Στα πλαίσια με συνεχόμενη γραμμή διακρίνουμε τα βασικά αρχεία του προσωμοιωτή, ενώ στα πλαίσια με διακεκομμένη γραμμή παρουσιάζουμε τα παραγόμενα αρχεία μιας υλοποίησης (στο παράδειγμα μας του πρωτοκόλλου των Δρομέων)

Αρχικά, θα πρέπει να αναφέρουμε ότι σε ένα διαφορετικό αρχείο κρατάμε τις τιμές διαφόρων σταθερών που χρησιμοποιούμε στον προσομοιωτή. Το αρχείο αυτό (*constants.h*[25]) περιέχει τις παραμέτρους του συστήματος, όπως είναι για παράδειγμα η ακτίνα μετάδοσης των κινητών κόμβων στο δίκτυο, το μέγεθος των σωρών (που αντιστοιχεί στο εύρος ζώνης) του δικτύου, καθώς και άλλες σταθερές που χρησιμοποιούνται από διάφορες ρουτίνες του συστήματος.

Το πρώτο βήμα του συστήματος προσομοίωσης είναι η αρχικοποίηση του. Σε αυτό το βήμα το σύστημα αρχικοποιεί τις διάφορες δομές που χρησιμοποιούνται, όπως είναι για παράδειγμα το γράφημα κίνησης και οι σωροί που αντιστοιχούν στις κορυφές τους γραφήματος κίνησης και αποθηκεύουν τα μηνύματα. Επίσης, σε αυτό το βήμα ο προσομοιωτής υπολογίζει μερικές άλλες δομές που χρησιμοποιούνται για τη ταχύτερη λειτουργία του συστήματος. Για παράδειγμα, υπολογίζονται εκ των προτέρων για κάθε κορυφή όλες οι γειτονικές της κορυφές (με βάση πάντα την εμβέλεια μετάδοσης), έτσι ώστε να μην χρειάζεται να επαναλαμβάνουμε κάθε φορά τους ίδιους υπολογισμούς κατά τη διάρκεια της προσομοίωσης, αλλά αντιθέτως να χρησιμοποιούμε τις δομές που έχουμε ήδη υπολογίσει.

Το επόμενο βήμα της αρχικοποίησης είναι η τοποθέτηση των κινητών κόμβων εντός του γραφήματος κίνησης με βάση τις απαιτήσεις του πρωτοκόλλου που υλοποιούμε. Μετά και από αυτό το βήμα είμαστε έτοιμοι για την εφαρμογή του κεντρικού βρόγχου προσομοίωσης που παρουσιάζουμε πιο κάτω:

1. **Αύξησε τον μετρητή βημάτων**
2. **Αναδιάταξε τη σειρά εκτέλεσης των κινητών κόμβων με τυχαίο τρόπο**
3. **Με βάση τη νέα διάταξη εκτέλεσε το πρωτόκολλο για το κάθε κινητό κόμβο**
4. **Παράδωσε τα μηνύματα από τους σωρούς του γραφήματος κίνησης στους σωρούς εισόδου των κινητών κόμβων.**

Η πιο πάνω διαδικασία επαναλαμβάνεται είτε για ένα σταθερό αριθμό βημάτων (που καθορίζουμε κατά την αρχικοποίηση), είτε μέχρι να συμβεί ένα συγκεκριμένο γεγονός (για παράδειγμα μέχρι να παραδοθεί ένας συγκεκριμένος αριθμός μηνυμάτων.

Το πρώτο βήμα έχει σαν σκοπό την καταμέτρηση των βημάτων της προσομοίωσης. Όπως έχουμε ήδη αναφέρει, το 2^{ο} βήμα αποσκοπεί στο δικαιότερο καταμερισμό των πόρων (και πιο ειδικά του μέσου προσπέλασης) στους κινητούς κόμβους. Στο 3^{ο} βήμα γίνεται

[25] Το σύνολο των αρχείων που χρησιμοποιούμε, και μεταξύ τους σχέσεις παρουσιάζονται στο σχήμα 4.5

ουσιαστικά η υλοποίηση του πρωτοκόλλου, αφού οι κινητοί κόμβοι εκτελούν την *executeProtocol()* μέθοδο τους. Τέλος, στο 4^{ο} βήμα γίνεται η παράδοση των μηνυμάτων από τους σωρούς του γραφήματος κίνησης, στους κινητούς κόμβους. Πιο συγκεκριμένα, για κάθε κορυφή του γραφήματος κίνησης, αντιγράφουμε τα μηνύματα που βρίσκονται στο σωρό της κορυφής αυτής σε κάθε ένα κινητό κόμβο που βρίσκεται εκείνη τη στιγμή εντός της ίδιας κορυφής.

4.4 Σύνοψη

Σε αυτό το κεφάλαιο αναφερθήκαμε στο τρόπο λειτουργίας και χρήσης του βασικού συστήματος προσομοίωσης που υλοποιήσαμε για την διεξαγωγή των μετρήσεων μας. Το σύστημα αυτό έχει φτιαχτεί έτσι ώστε να είναι γενικό και να μπορεί εύκολα να υλοποιήσει πρωτόκολλα που γράφονται για ad-hoc κινητά δίκτυα.

Τέλος, στο Παράρτημα αναφέρονται περισσότερες πληροφορίες για τη χρήση του συστήματος. Εκεί υπάρχουν ακόμη και χρήσιμα παραδείγματα κώδικα με τον οποίο υλοποιήσαμε τις μετρήσεις που φαίνονται στο Κεφάλαιο 5.

5. Τα Προτεινόμενα Πρωτόκολλα

Σε αυτό το κεφάλαιο ερευνούμε δύο βασικά πρωτόκολλα επικοινωνίας για ad-hoc κινητά δίκτυα. Τα πρωτόκολλα αυτά και η πειραματική ανάλυση που ακολουθεί στηρίζονται στην εργασία [1].

Στα πρωτόκολλα αυτά ακολουθούμε την ημι-επιτακτική (semi-compulsory) προσέγγιση στην οποία ένα μικρό τμήμα των κινητών χρηστών, που ονομάζουμε υποστήριξη (support) και συμβολίζουμε με *Σ*, κινείται με προκαθορισμένο τρόπο και χρησιμοποιείται σαν ένας ενδιάμεσος χώρος αποθήκευσης μηνυμάτων. Κάτω από αυτή την προσέγγιση, παρουσιάζουμε ένα ημι-επιτακτικό πρωτόκολλο που ονομάζουμε πρωτόκολλο των δρομέων (*runners protocol)*, και στο οποίο τα μέλη της υποστήριξης *Σ* κάνουν διαρκώς τυχαίους περιπάτους (random walks) και ανταλλάζουν μηνύματα που παίρνουν από τους αποστολείς και κατόπιν, όταν συναντηθούν με τους παραλήπτες, τους τα παραδίδουν. Επιπλέον, παρουσιάζουμε ένα δεύτερο (παλαιότερο) πρωτόκολλο, που ονομάζεται πρωτόκολλο του φιδιού (*snake protocol)*[26], στο οποίο τα μέλη της υποστήριξης *Σ* κινούνται με συντονισμένο τρόπο έτσι ώστε να μένουν πάντα ανά δύο συνδεδεμένα (δηλαδή εντός ακτίνας επικοινωνίας). Για την πειραματική αξιολόγηση των δύο αυτών πρωτοκόλλων χρησιμοποιήσαμε το γενικό περιβάλλον προσομοίωσης πρωτοκόλλων για ad-hoc κινητά δίκτυα που αναπτύξαμε και παρουσιάσαμε στο προηγούμενο κεφάλαιο. Τα πειραματικά αποτελέσματα έδειξαν ότι χρησιμοποιώντας ένα μικρό μέρος των χρηστών σαν υποστήριξη είναι αρκετό για να έχουμε αποδοτική επικοινωνία. Επιπλέον συμπεράναμε ότι το πρωτόκολλο των *runners* έχει καλύτερη απόδοση σε σχέση με το πρωτόκολλο *snake*, σχεδόν για κάθε τύπο εισόδου που χρησιμοποιήσαμε.

Τέλος, θα πρέπει να επισημάνουμε ότι τα πρωτόκολλα αυτά έχουν νόημα να χρησιμοποιούνται σε κινητά δίκτυα όπου ο ρυθμός μετακίνησης των κόμβων είναι πολύ μεγάλος (δηλαδή οι κόμβοι κινούνται με μεγάλη πιθανότητα και γρήγορα). Σε αυτή τη περίπτωση είναι πολύ δύσκολο να λειτουργήσει οποιοδήποτε από τα υπάρχοντα πρωτόκολλα που περιγράψαμε στο Κεφάλαιο 3, αφού η σύγκλιση του δικτύου θα είναι πρακτικά αδύνατη όταν οι συνδέσεις σπάνε μαζικά και με μεγάλη συχνότητα. Τα συγκεκριμένα όμως πρωτόκολλα που παρουσιάζουμε σε αυτό το κεφάλαιο, λόγω κυρίως της πιθανοθεωρητικής τους φύσης, μπορούν να επιλύσουν το πρόβλημα της επικοινωνίας με ικανοποιητικό (όπως θα δούμε) τρόπο.

[26] Τα ονόματα runners και snakes έχουν επιλεγεί για τα δύο αυτά πρωτόκολλα λόγω της ομοιότητας που παρουσιάζουν οι κόμβοι που αποτελούν την υποστήριξη *Σ* με δρομείς και φιδάκι αντίστοιχα.

5.1 Θεωρητική Μελέτη Προτεινόμενων Πρωτοκόλλων

Στο 1^{ο} Κεφάλαιο είχαμε πει ότι ο συνηθέστερος τρόπος επικοινωνίας σε ad-hoc δίκτυα είναι ο σχηματισμός μονοπατιών με χρήση ενδιάμεσων κόμβων. Η προσέγγιση αυτή όμως μπορεί να είναι μη αποδοτική στην περίπτωση που το δίκτυο είναι σχετικά αραιό, και η κινητικότητα των κόμβων αρκετά μεγάλη. Σε αυτή την περίπτωση, εισάγουμε μια διαφορετική προσέγγιση, στην οποία εκμεταλλευόμαστε την φυσική κίνηση των κόμβων.

Στην προσέγγιση αυτή, το δίκτυο μας έχει ημι-επιτακτική μορφή, δηλαδή ένας (μικρός) αριθμός από κόμβους δεν κινούνται αυθαίρετα αλλά ακολουθούν κάποιους κανόνες που καθορίζουν την κίνηση τους. Οι κόμβοι αυτοί ονομάζονται *υποστήριξη* (support) και συμβολίζονται με *Σ*.

Είναι προφανές ότι αν κάποιοι κόμβοι βρίσκονται απομονωμένοι σε απομακρυσμένες περιοχές, χωρίς ποτέ να ξεπερνούν τα όρια των περιοχών αυτών, τότε δεν υπάρχει τρόπος να φτάσει η πληροφορία σε αυτούς, εκτός και αν το πρωτόκολλο λαμβάνει υπόψη του αυτή την ιδιαιτερότητα του δικτύου. Ένας τρόπος να ξεπεραστεί αυτό το πρόβλημα είναι να αναγκάσουμε μερικούς χρήστες να κινούνται με κάποιους προκαθορισμένους κανόνες έτσι ώστε να ικανοποιούν τις απαιτήσεις του πρωτοκόλλου.

Το πρώτο ημι-επιτακτικό πρωτόκολλο για την επίλυση του βασικού προβλήματος της επικοινωνίας είναι το *πρωτόκολλο του φιδιού* που παρουσιάστηκε στην εργασία [2]. Χρησιμοποιεί μια ακολουθία από σταθμούς υποστήριξης που είναι διαρκώς ανά δύο συνδεδεμένοι, σχηματίζοντας έτσι ένα “φιδάκι”. Η πορεία αυτών των σταθμών καθορίζεται από τον πρώτο, που ονομάζουμε κεφαλή του φιδιού. Αυτός ο κόμβος κινείται εκτελώντας ένα τυχαίο περίπατο πάνω στην περιοχή που καλύπτει το δίκτυο. Θα αναφερόμαστε σε αυτό το πρωτόκολλο σαν το πρωτόκολλο του φιδιού. Αυτό το πρωτόκολλο έχει μάλιστα μελετηθεί θεωρητικά στην εργασία [2] χρησιμοποιώντας ενδιαφέρουσες ιδιότητες των τυχαίων περιπάτων καθώς και των χρόνων συνάντησης σε αυτούς. Μια πρώτη υλοποίηση του πρωτοκόλλου καθώς και πειραματική αξιολόγηση του έγιναν στην εργασία [2] με έμφαση στην επιβεβαίωση της θεωρητικής ανάλυσης. Τόσο η θεωρητική όσο και η πειραματική ανάλυση έδειξαν ότι ένας μικρός αριθμός από κόμβους υποστήριξης είναι αρκετός για αποδοτική επικοινωνία.

Στην συνέχεια, παρουσιάζουμε ένα νέο πρωτόκολλο για την επίλυση του προβλήματος της επικοινωνίας, που ονομάζουμε *πρωτόκολλο των δρομέων*. Η βασική ιδέα σε αυτό το πρωτόκολλο είναι ότι οι κόμβοι της υποστήριξης κινούνται σαν δρομείς, δηλαδή ο καθένας κινείται ανεξάρτητα από τους άλλους καλύπτοντας ολόκληρη την περιοχή του δικτύου. Όταν δύο δρομείς συναντιούνται τότε ανταλλάζουν πληροφορίες που τους έδωσαν οι κόμβοι που

συνάντησαν. Επιπλέον, το νέο αυτό πρωτόκολλο φαίνεται να είναι και πιο εύρωστο[27] από το αντίστοιχο πρωτόκολλο με το φιδάκι. Συγκεκριμένα, το πρωτόκολλο αυτό συνεχίζει να λειτουργεί (με μειωμένη απόδοση) ακόμη και αν t κόμβοι δεν λειτουργούν, όπου $t < |\Sigma|$. Αντίθετα, το πρωτόκολλο με το φιδάκι παύει να λειτουργεί σωστά ακόμη και στην περίπτωση που ένας μόνο κόμβος της υποστήριξης παύει να λειτουργεί.

5.1.1. Περιγραφή των Πρωτοκόλλων

Αρχίζουμε με μερικούς ορισμούς που θα χρησιμοποιηθούν στην περιγραφή των πρωτοκόλλων. Όπως έχουμε ήδη αναφέρει, τα *ημι-επιτακτικά* πρωτόκολλα είναι αυτά στα οποία ένας μικρός αριθμός από κόμβους εξυπηρετεί τις ανάγκες του δικτύου.

Το υποσύνολο των κόμβων ενός πρωτοκόλλου για ad-hoc κινητά δίκτυα των οποίων η κίνηση καθορίζεται από το πρωτόκολλο *P* ονομάζεται *υποστήριξη* Σ του *P*. Το τμήμα του *P* που καθορίζει τον τρόπο με τον οποίο τα μέλη του Σ κινούνται και επικοινωνούν ονομάζεται *υπο-πρωτόκολλο διαχείρισης υποστήριξης* (support management sub-protocol) M_Σ του *P*.

Επιπλέον, μπορεί να επιθυμούμε οι κόμβοι του Σ να κινούνται και να επικοινωνούν με τρόπο που να μπορούν να ανεχτούν βλάβες των κόμβων. Σε αυτή την περίπτωση λέμε ότι το πρωτόκολλο είναι *εύρωστο*. Αντίστοιχα, ένα πρωτόκολλο καλείται *αξιόπιστο* εάν επιτρέπει στον αποστολέα να ενημερώνεται για την παράδοση της πληροφορίας στον κατάλληλο παραλήπτη.

Υποθέτουμε ότι οι κινήσεις των κόμβων που δεν είναι μέλη της υποστήριξης Σ είναι αυθαίρετες αλλά και ανεξάρτητες από τη κίνηση της υποστήριξης. Δηλαδή αποκλείουμε την περίπτωση όπου κάποιοι από τους κόμβους εκτός της υποστήριξης προσπαθούν εσκεμμένα να αποφύγουν τους κόμβους της υποστήριξης.

Τέλος, υποθέτουμε ότι τα μηνύματα που ανταλλάσσονται μεταξύ των κόμβων που βρίσκονται εντός ακτίνας επικοινωνίας χρειάζονται αμελητέο χρόνο. Δηλαδή τα μηνύματα είναι σύντομα πακέτα και η ασύρματη επικοινωνία είναι πολύ γρήγορη. Επιπλέον, υποθέτουμε ότι όλοι οι χρήστες, ακόμη και αυτοί που δεν ανήκουν στην υποστήριξη, εκτελούν ταυτόχρονα τυχαίους περιπάτους.

Είμαστε τώρα έτοιμοι να περιγράψουμε τα δύο πρωτόκολλα, καθένα από τα οποία έχει διαφορετικό υπο-πρωτόκολλο διαχείρισης υποστήριξης M_Σ.

[27] Εύρωστο ή robust, είναι ένα πρωτόκολλο που έχει την ικανότητα να αντεπεξέρχεται σε περιπτώσεις που ένα τμήμα των κόμβων του δικτύου δεν λειτουργεί καθόλου ή δεν λειτουργεί σωστά.
Αξιόπιστο ή reliable είναι ένα πρωτόκολλο στο οποίο η παράδοση ενός πακέτου στον κατάλληλο παραλήπτη επιβεβαιώνεται με την παράδοση μιας βεβαίωσης στον αποστολέα.

5.1.2. Το Πρωτόκολλο του Φιδιού - The Snake Protocol

Η αρχική ιδέα του πρωτοκόλλου όπως είχε προταθεί στην εργασία [2] έχει ως ακολούθως. Υπάρχει πρώτα μια φάση αρχικοποίησης του ad-hoc δικτύου κατά την οποία ένας προκαθορισμένος αριθμός K από κόμβους επιλέγονται για την υποστήριξη. Τα μέλη αυτής της υποστήριξης εκτελούν μια διαδικασία εκλογής αρχηγού, η οποία γίνεται μόνο μια φορά και εισάγει ένα αρχικό κόστος επικοινωνίας. Ο εκλεχθείς αρχηγός, τον οποίο συμβολίζουμε με MS_0, χρησιμοποιείται για το συντονισμό της κίνησης του τμήματος υποστήριξης. Επιπρόσθετα, ο αρχηγός αναθέτει τοπικά ονόματα στα υπόλοιπα μέλη της υποστήριξης $MS_1, MS_2, \ldots, MS_{K-1}$.

Οι κόμβοι της υποστήριξης κινούνται με ένα συντονισμένο τρόπο, παραμένοντας διαρκώς σε επαφή ανά δύο, σχηματίζοντας έτσι μια ακολουθία από K κόμβους. Οι κόμβοι αυτοί σαρώνουν, δοθέντος κάποιου χρόνου, ολόκληρο το χώρο του γραφήματος κίνησης. Η κίνηση τους επιτυγχάνεται με ένα κατανεμημένο τρόπο μέσω ενός υπο-πρωτοκόλλου υποστήριξης κίνησης P_1. Ουσιαστικά το υπο-πρωτόκολλο κίνησης P_1 αναγκάζει την υποστήριξη να κινείται σαν ένα φίδι, με την κεφαλή (ο εκλεγμένος αρχηγός MS_0) να εκτελεί τυχαίο περίπατο, και καθένα από τους υπόλοιπους κόμβους MS_i να εκτελούν τον απλό αλγόριθμο: “πήγαινε εκεί που ήταν ο MS_{i-1} προηγουμένως”. Όταν ένας κόμβος της υποστήριξης είναι εντός ακτίνας επικοινωνίας ενός αποστολέα, ένα δεύτερο υπο-πρωτόκολλο (sensor sub-protocol) P_2 ειδοποιεί τον αποστολέα ότι μπορεί να στείλει τα μηνύματα του. Τα μηνύματα αποθηκεύονται τότε σε κάθε ένα από τους κόμβους υποστήριξης χρησιμοποιώντας ένα υπο-πρωτόκολλο συγχρονισμού (synchronization sub-protocol) P_3. Όταν ένας παραλήπτης έρχεται εντός της ακτίνας επικοινωνίας ενός από τους κόμβους της υποστήριξης, τότε ενημερώνεται ότι υπάρχει ένα μήνυμα για αυτόν και ακολούθως το μήνυμα προωθείται σε αυτόν. Τα πολλαπλά αντίγραφα των μηνυμάτων αφαιρούνται τότε από τους υπόλοιπους κόμβους της υποστήριξης.

Σε αυτό το πρωτόκολλο, οι κόμβοι της υποστήριξης Σ έχουν το ρόλο ενός (κινητού) κορμού υποδικτύου με σταθερή δομή (που εξασφαλίζεται από το υπο-πρωτόκολλο κίνησης P_1), μέσω του οποίου δρομολογείται ολόκληρο το επικοινωνιακό φορτίο. Η θεωρητική ανάλυση του πρωτοκόλλου στην εργασία [2] έχει δείξει ότι ο αναμενόμενος χρόνος επικοινωνίας T_{total} του πρωτοκόλλου του φιδιού φράσσεται εκ των άνω από τον τύπο: $E(T_{total}) \le \frac{2}{\lambda_2(G)}\Theta(\frac{n}{k}) + \Theta(k)$

όπου G είναι το γράφημα κίνησης, $\lambda_2(G)$ η δεύτερη ιδιοτιμή του G, n ο αριθμός των κορυφών του G, και $k = |\Sigma|$ (το μέγεθος της υποστήριξης). Περισσότερες λεπτομέρειες για το πρωτόκολλο του *Φιδιού* υπάρχουν στην εργασία [2]. Μπορεί επίσης να αποδειχθεί ([20]) ότι αυτό το πρωτόκολλο είναι αξιόπιστο, και μερικώς εύρωστο (ανθεκτικό σε ένα λάθος).

5.1.3. Το Πρωτόκολλο των Δρομέων – The Runners Protocol

Μια διαφορετική προσέγγιση στην υλοποίηση του υπο-πρωτοκόλλου διαχείρισης υποστήριξης M_{Σ} είναι να επιτρέπουμε σε κάθε μέλος της υποστήριξης Σ να πραγματοποιεί ένα *ανεξάρτητο τυχαίο περίπατο* πάνω στο γράφημα κίνησης G. Δηλαδή, τα μέλη της υποστήριξης Σ μπορούν να θεωρηθούν σαν "δρομείς" που τρέχουν πάνω στο G. Με άλλα λόγια, αντί να διατηρούμε τους κόμβους της υποστήριξης διαρκώς σε "επαφή", τους επιτρέπουμε να κινούνται σε ολόκληρη την επιφάνεια, ανεξάρτητα ο ένας από τον άλλο. Όταν δύο δρομείς συναντιούνται, ανταλλάζουν πληροφορίες που τους έδωσαν οι αποστολείς που συνάντησαν χρησιμοποιώντας ένα νέο *υπο-πρωτόκολλο συγχρονισμού, το P_3'*. Όπως και στην περίπτωση του φιδιού, όταν ένας κόμβος μέλος της υποστήριξης βρεθεί εντός ακτίνας επικοινωνίας με ένα αποστολέα, τότε το υπο-πρωτόκολλο (sensor sub-protocol) P_2 ειδοποιεί τον αποστολέα ότι μπορεί να στείλει τα μηνύματα του. Όταν ένας κόμβος έρχεται εντός της ακτίνας επικοινωνίας ενός κόμβου της υποστήριξης που έχει ένα μήνυμα για το συγκεκριμένο παραλήπτη R, τα μηνύματα αυτά προωθούνται προς τον παραλήπτη τους.

Το πρωτόκολλο των δρομέων δεν χρησιμοποιεί την ιδέα του (κινούμενου) κορμού υποδικτύου αφού δεν χρησιμοποιείται κάποιο υπο-πρωτόκολλο κίνησης P_1. Παρόλα αυτά, η επικοινωνία εξακολουθεί να συνεχίζει να δρομολογείται μέσω της υποστήριξης Σ και αναμένουμε ότι το μέγεθος k της υποστήριξης Σ (δηλαδή ο αριθμός των δρομέων) θα επηρεάζει την απόδοση με ένα πιο αποδοτικό τρόπο από ότι η προσέγγιση του φιδιού. Αυτή η πρόβλεψη στηρίζεται στη σκέψη ότι αφού οι κόμβοι της υποστήριξης θα συναντιούνται παράλληλα, η διάδοση της πληροφορίας θα επιταχύνεται, δηλαδή τα μηνύματα θα παραδίδονται πιο γρήγορα.

Ένα μέλος της υποστήριξης χρειάζεται να αποθηκεύει όλα τα μηνύματα που δεν έχουν ακόμη παραδοθεί, και επίσης να διατηρεί μια λίστα με τις βεβαιώσεις παράδοσης που θα πρέπει να φθάσουν στους αρχικούς αποστολείς. Για λόγους απλότητας θεωρούμε ένα γενικό σύστημα αποθήκευσης, όπου όλα τα μηνύματα που δεν έχουν ακόμη παραδοθεί ανήκουν στο σύνολο S_1, και όλα τα μηνύματα που έχουν παραδοθεί σε ένα δεύτερο σύνολο S_2. Κατά ακρίβεια στο σύνολο S_2 είναι αρκετό να κρατάμε μόνο την κεφαλίδα του μηνύματος που το χαρακτηρίζει μοναδικά.

Όταν δύο δρομείς συναντιούνται στην ίδια περιοχή του γραφήματος κίνησης G, τότε ενεργοποιείται το υπο-πρωτόκολλο συγχρονισμού P_3'. Το υπο-πρωτόκολλο αυτό αναγκάζει δύο δρομείς που συναντιούνται να συγχρονίζουν τα σύνολα τους S_1 και S_2. Με αυτό τον τρόπο, ένα μήνυμα που είναι προς παράδοση από κάποιον δρομέα αντιγράφεται στα σύνολα S_1 των υπόλοιπων δρομέων που συναντάει. Παρόμοια, οι κεφαλίδες των μηνυμάτων που έχουν ήδη

παραδοθεί θα αφαιρεθούν από τα σύνολα S_2 των υπολοίπων δρομέων. Το υπο-πρωτόκολλο συγχρονισμού *P_3'* βασίζεται μερικώς στον "*two-phase commit*" αλγόριθμο που παρουσιάζεται στο [21] και η λειτουργία του περιγράφεται στην συνέχεια.

Θεωρούμε τα μέλη του Σ που βρίσκονται στην ίδια περιοχή (π.χ. κορυφή u) του γραφήματος κίνησης G και έστω ότι αυτά είναι τα MS_1^u, MS_2^u, …, MS_j^u. Έστω επίσης ότι $S_1(i)$ (αντίστοιχα $S_2(i)$) συμβολίζει το σύνολο S_1 (αντίστοιχα S_2) του δρομέα MS_i^u, $1 \le i \le j$. Ο αλγόριθμος υποθέτει επιπλέον ότι το υπο-πρωτόκολλο (sensor sub-protocol) P_2 ενημερώνει όλους τους κόμβους για τον δρομέα με το μικρότερο id, δηλαδή τον δρομέα MS_1^u. Το πρωτόκολλο *P_3'* αποτελείται από δύο γύρους.

Γύρος 1: Όλοι οι κόμβοι MS_1^u, MS_2^u, …, MS_j^u που βρίσκονται στην κορυφή u του G, στέλνουν τα σύνολα τους S_1 και S_2 στο δρομέα MS_1^u. Ο τελευταίος συγκεντρώνει όλα τα σύνολα μαζί με το δικό του για να υπολογίσει τα νέα σύνολα S_1 και S_2 ως εξής: $S_2(1) = \bigcup_{1 \le l \le j} S_2(l)$

και $S_1(1) = \bigcup_{1 \le l \le j} S_1(l) - S_2(1)$

Γύρος 2: Ο δρομέας MS_1^u κάνει broadcast την απόφαση του σε όλους τους άλλους κόμβους υποστήριξης. Όλοι οι κόμβοι που λαμβάνουν το broadcast εφαρμόζουν τους ίδιους κανόνες που εφάρμοσε ο MS_1^u για να ενημερώσουν τα σύνολα τους S_1 και S_2 με τις τιμές που έλαβαν. Κάθε κόμβος που λαμβάνει το μήνυμα στο Γύρο 2 χωρίς να έχει συμμετάσχει στο Γύρο 1, δέχεται τις τιμές που λαμβάνει στο μήνυμα συγχρονισμού σαν να είχε συμμετάσχει κανονικά στο Γύρο 1.

Αυτός ο απλός αλγόριθμος εγγυάται ότι οι κινητοί κόμβοι που παραμένουν συνδεδεμένοι (δηλαδή μπορούν να ανταλλάσσουν απευθείας μηνύματα) για δύο συνεχόμενους γύρους, καταφέρνουν να συγχρονίσουν τα σύνολα τους S_1 και S_2. Επιπλέον, στην περίπτωση που ένας νέος κόμβος φθάνει, ή ένας άλλος αποσυνδέεται, κατά το Γύρο 2, η εκτέλεση του πρωτοκόλλου δεν επηρεάζεται. Στην περίπτωση όπου ο δρομέας MS_1^u αποτυγχάνει να μεταδώσει στο Γύρο 2 (είτε λόγω κάποιας προσωρινής απενεργοποίησης, ή διότι έχει φύγει από την περιοχή), τότε το πρωτόκολλο επανα-εκτελείται με τους υπόλοιπους κόμβους υποστήριξης.

Το πρωτόκολλο διαχείρισης υποστήριξης που περιγράψαμε είναι προφανώς αξιόπιστο. Είναι επίσης εύρωστο αφού μπορεί να ανεχτεί μέχρι t σφάλματα, για $0 \le t \le k$. Όταν δύο δρομείς συναντιούνται δημιουργούν αντίγραφα των μηνυμάτων που έχουν προς παράδοση. Στη χειρότερη περίπτωση, υπάρχουν $k - t$ αντίγραφα από κάθε μήνυμα. Στην περίπτωση όμως που υπάρχει ένα μόνο αντίγραφο στο κόμβο υποστήριξης που έχει το σφάλμα, τότε το μήνυμα θα πρέπει να ξανασταλθεί. Για να ξεπεράσουμε αυτό τον περιορισμό, ο κάθε αποστολέας

συνεχίζει να στέλνει κάθε μήνυμα του οποίου η παράδοση δεν έχει ακόμη επιβεβαιωθεί σε κάθε κόμβο υποστήριξης που συναντάει. Αυτό εγγυάται ότι περισσότερα από ένα αντίγραφα του μηνύματος θα βρίσκονται εντός της δομής της υποστήριξης *Σ*. Αυτό φυσικά έχει σαν παρενέργεια και την αύξηση του αριθμού των πακέτων που μεταδίδονται στο δίκτυο, το οποίο με τη σειρά του όμως μειώνει περαιτέρω τον χρόνο παράδοσης του κάθε πακέτου.

5.2 Πειραματική Αξιολόγηση Προτεινόμενων Πρωτοκόλλων

Για την πειραματική αξιολόγηση των δύο πρωτοκόλλων χρησιμοποιήσαμε το γενικό περιβάλλον προσομοίωσης ad-hoc κινητών δικτύων που περιγράψαμε στο προηγούμενο κεφάλαιο. Στα πειράματα μας χρησιμοποιήσαμε διάφορες εισόδους από γραφήματα κίνησης, δηλαδή γραφήματα που περιγράφουν τη μορφή του δικτύου. Χρησιμοποιήσαμε τόσο τυχαία γραφήματα όσο και πιο δομημένα γραφήματα, όπως είναι για παράδειγμα τα πλέγματα (grids). Στην εξαγωγή αποτελεσμάτων επικεντρωθήκαμε βασικά στο χρόνο επικοινωνίας, και πιο συγκεκριμένα στο μέσο χρόνο παράδοσης μηνυμάτων. Επιπλέον, μετρήσαμε τον ρυθμό παράδοσης μηνυμάτων, αλλά και τον μέσο αριθμό μηνυμάτων που περιείχαν οι κόμβοι υποστήριξης σε συνάρτηση με τον χρόνο.

Τα πειράματα αυτά έδειξαν ότι:

1. Ένας μικρός αριθμός από κόμβους υποστήριξης είναι αρκετός για αποδοτική επικοινωνία και στα δύο πρωτόκολλα.

2. Το πρωτόκολλο των δρομέων (runners protocol) υπερέχει του αντίστοιχου πρωτοκόλλου του φιδιού (snake protocol) σχεδόν για κάθε είσοδο που χρησιμοποιήσαμε στη διεξαγωγή των πειραματικών μετρήσεων. Πιο συγκεκριμένα το πρωτόκολλο των δρομέων πετυχαίνει καλύτερο μέσο χρόνο παράδοσης μηνυμάτων για κάθε είσοδο που χρησιμοποιήσαμε, εκτός από τα τυχαία γραφήματα με μικρό αριθμό από κόμβους υποστήριξης.

3. Το πρωτόκολλο των δρομέων πετυχαίνει υψηλότερο ρυθμό παράδοσης μηνυμάτων σχεδόν από την αρχή. Αντίθετα, το αντίστοιχο πρωτόκολλο με το φιδάκι απαιτεί αρχικά κάποιο χρόνο προτού ο ρυθμός του παράδοσης μηνυμάτων αυξηθεί και σταθεροποιηθεί σε μια τιμή, που είναι συνεχώς μικρότερη από αυτή των δρομέων.

4. Το πρωτόκολλο των δρομέων έχει μικρότερες απαιτήσεις σε μέγεθος τοπικής μνήμης ανά κόμβο που συμμετέχει στην υποστήριξη.

5.2.1 Πειραματικά Δεδομένα

Για την μοντελοποίηση των διάφορων πιθανών συνθηκών σχετικά με την γεωγραφική περιοχή που καλύπτουν τα ad-hoc δίκτυα, υλοποιήσαμε ένα μεγάλο αριθμό πειραμάτων. Πιο συγκεκριμένα χρησιμοποιήσαμε 5 διαφορετικές εισόδους, αποτελούμενες από μη δομημένα (τυχαία) γραφήματα, καθώς και από πιο δομημένα γραφήματα. Κάθε διαφορετικός τύπος εισόδου αντιστοιχεί σε ένα διαφορετικό γράφημα κίνησης. Αρχίζουμε με γραφήματα όμοια με αυτά που χρησιμοποιήθηκαν στην [2], δηλαδή τυχαία γραφήματα, δυσδιάστατα πλέγματα, και διμερή γραφήματα πολλαπλών επιπέδων. Επιπλέον, επεκτείνουμε αυτές τις εισόδους με τρισδιάστατα πλέγματα και με γραφήματα κίνησης δύο επιπέδων, που είναι πιο κοντά στις πραγματικές συνθήκες.

Θεωρούμε διάφορες τιμές για το n (αριθμός κορυφών του γραφήματος κίνησης) μέσα στο διάστημα [400, 6400] και διάφορες τιμές για το μέγεθος της υποστήριξης k = |Σ|, στο διάστημα [3, 45]. Για κάθε γράφημα κίνησης που δημιουργούμε, εισάγουμε 1000 χρήστες (κινητούς σταθμούς) σε τυχαίες θέσεις, οι οποίοι παράγουν 100 πακέτα ο καθένας διαλέγοντας τυχαίους προορισμούς. Κάθε πείραμα που διεξάγουμε εκτελείται σε διακριτά βήματα που ονομάζουμε βήματα προσομοίωσης (δηλαδή μετράμε τον χρόνο σε βήματα). Κάθε ένας από τους κινητούς χρήστες, έστω ο *h*, θεωρείται όμοιος με κάθε άλλο χρήστη από άποψη υπολογιστικών και επικοινωνιακών δυνατοτήτων. Κατά την διάρκεια ενός βήματος προσομοίωσης, ο *h* μετακινείται σε μια γειτονική κορυφή πάνω στο γράφημα κίνησης *G* και εκτελεί κάποιους υπολογισμούς. Εάν ο $h \in \Sigma$, τότε η κίνηση του καθώς και οι τοπικοί του υπολογισμοί καθορίζονται από το πρωτόκολλο του φιδιού ή των δρομέων. Εάν ο $h \notin \Sigma$, τότε η κίνηση του είναι τυχαία και με πιθανότητα p = 0.01 ο χρήστης h δημιουργεί ένα καινούριο μήνυμα (δηλαδή ένα καινούριο μήνυμα δημιουργείται από τον κάθε χρήστη κάθε περίπου 100 βήματα). Σε κάθε μήνυμα που δημιουργείται, ο κόμβος προορισμού επιλέγεται τυχαία. Η επιλογή της συγκεκριμένης τιμής για το p βασίζεται στην υπόθεση ότι οι χρήστες δεν εκτελούν μια εφαρμογή πραγματικού-χρόνου που να απαιτεί διαρκή ανταλλαγή μηνυμάτων. Βασισμένοι σε αυτή την επιλογή του p, κάθε πείραμα απαιτεί αρκετές χιλιάδες βήματα για να δώσει μια αποδεκτή μέση καθυστέρηση παράδοσης μηνύματος. Για κάθε πείραμα, αποστέλλονται συνολικά 100000 μηνύματα. Η εκτέλεση των πειραμάτων ολοκληρώνεται όταν παραδοθούν και τα 100000 μηνύματα στους κατάλληλους παραλήπτες. Οι είσοδοι των πειραμάτων μας έχουν ως εξής:

Τυχαία Γραφήματα: Αυτή η κλάση γραφημάτων αποτελεί ένα φυσικό σημείο εκκίνησης έτσι ώστε να πειραματιστούμε σε περιοχές με εμπόδια. Χρησιμοποιούμε το $G_{n,p}$ μοντέλο στο οποίο παράγουμε τα γραφήματα επιλέγοντας ανεξάρτητα τις ακμές ενός πλήρους γραφήματος με

πιθανότητα p. Χρησιμοποιούμε την τιμή $p = \frac{1.05 \log n}{n}$, η οποία είναι ελαφρώς μεγαλύτερη από το όριο συνεκτικότητας (connectivity threshold) για το $G_{n,p}$ μοντέλο. Στα πειράματα χρησιμοποιήσαμε τυχαία γραφήματα με αριθμό κορυφών n ∈ {1600, 3200, 6400}, πάνω σε διάφορες τιμές του k στο διάστημα [3, 45].

Δυσδιάστατα Γραφήματα: Αυτή η κλάση γραφημάτων είναι το απλούστερο μοντέλο κίνησης που μπορούμε να θεωρήσουμε (για παράδειγμα κινητοί χρήστες που κινούνται σε μια επίπεδη επιφάνεια χωρίς εμπόδια). Χρησιμοποιούμε δύο διαφορετικά $\sqrt{n} \times \sqrt{n}$ πλέγματα με αριθμό κορυφών n ∈ {400, 1600} και διάφορα μεγέθη υποστήριξης k. Για παράδειγμα, στην περίπτωση που το $V(tc) = 50m^2$ τότε η περιοχή που καλύπτεται από το ad-hoc δίκτυο θα είναι 20000m^2 στην περίπτωση που το n = 400.

Τρισδιάστατα Γραφήματα: Για να υπολογίσουμε την απόδοση των πρωτοκόλλων σε τρισδιάστατο χώρο, χρησιμοποιήσαμε 3D πλέγματα ($n^{1/3} \times n^{1/3} \times n^{1/3}$) για την μοντελοποίηση της κίνησης των κόμβων στις τρεις διαστάσεις. Για αυτή τη περίπτωση χρησιμοποιήσαμε τρία διαφορετικά γραφήματα με αριθμό κορυφών n ∈ {512, 1000, 1280} και μέγεθος υποστήριξης k στο διάστημα [3, 45].

Διμερή Γραφήματα Πολλαπλών Επιπέδων (Bipartite multi-stage Γραφήματα): Ένα διμερές γράφημα πολλαπλών επιπέδων είναι ένα γράφημα που αποτελείται από ένα αριθμό από επίπεδα (stages) *ξ*. Κάθε επίπεδο περιέχει n / *ξ* κορυφές και υπάρχουν ακμές μεταξύ κορυφών διαδοχικών επιπέδων. Αυτές οι ακμές επιλέγονται τυχαία με κάποια πιθανότητα p ανάμεσα σε όλες τις πιθανές ακμές μεταξύ των δύο επιπέδων. Αυτός ο τύπος γραφημάτων έχει ενδιαφέρον γιατί μοντελοποιεί την κίνηση κόμβων που πρέπει να περάσουν από συγκεκριμένες περιοχές, και έχουν διαφορετική δεύτερη ιδιοτιμή από ότι τα πλέγματα και τα $G_{n,p}$ γραφήματα (η δεύτερη τους ιδιοτιμή βρίσκεται μεταξύ των ιδιοτιμών των πλεγμάτων και των $G_{n,p}$ γραφημάτων). Στα πειράματα μας θεωρούμε ότι έχουμε ξ = log n επίπεδα, και επιλέγουμε $p = \frac{\xi}{n} \log \frac{n}{\xi}$ που είναι και το όριο για διμερή συνεκτικότητα (δηλαδή συνεκτικότητα μεταξύ κάθε ζεύγους επιπέδων). Οι πειραματικές συνθήκες περιλαμβάνουν διμερή γραφήματα πολλαπλών επιπέδων με 7, 8, και 9 επίπεδα, αριθμό κορυφών n ∈ {1600, 3200, 6400}, και διαφορετικά μεγέθη υποστήριξης k στο διάστημα [3, 45].

Γραφήματα Κίνησης δύο επιπέδων (Two-level motion graph): Το κίνητρο που μας ώθησε στην χρησιμοποίηση αυτών των γραφημάτων είναι το γεγονός ότι οι πιο πολλοί κινητοί χρήστες ταξιδεύουν συνήθως κατά μήκος συγκεκριμένων διαδρομών, που ονομάζουμε αγαπημένες

διαδρομές. Για παράδειγμα, ένας χρήστης ταξιδεύει πολύ συχνά από το σπίτι του προς την εργασία του. Συνήθως αυτές οι διαδρομές καλύπτουν ένα μικρό μέρος του δικτύου (π.χ. αυτοκινητόδρομους, γραμμές τρένων, κλπ). Επιπλέον, αυτά τα γραφήματα μοντελοποιούν και περιοχές με μεγάλη κίνηση, όπως είναι για παράδειγμα κεντρικές πλατείες, αεροδρόμια, και τουριστικά αξιοθέατα.

Έστω ότι το d(u, v) αναπαριστά την απόσταση μεταξύ των κορυφών u και v στο γράφημα G, δηλαδή την απόσταση της συντομότερης διαδρομής που τις συνδέει. Ένα δι-επίπεδο (two-level) γράφημα αποτελείται από υπο-γραφήματα του γραφήματος κίνησης (motion graph) που αναπαριστούν *συνωστισμένες περιοχές* που συνδέονται με ένα μικρό αριθμό από μονοπάτια που αντιπροσωπεύουν τις αγαπημένες διαδρομές μεταξύ αυτών. Ένα υπο-γράφημα G_c ορίζεται με την τυχαία επιλογή μιας κορυφής u του G και ακολούθως την προσθήκη των κορυφών v για τις οποίες ισχύει $v \in G$ με $d(u, v) < c$ στο γράφημα G_c, όπου c είναι μια σταθερά που χαρακτηρίζει την ακτίνα της συνωστισμένης περιοχής. Έστω ακόμη f ο αριθμός των υπο-γραφημάτων που ορίζουμε. Τα μονοπάτια που αντιπροσωπεύουν τις αγαπημένες διαδρομές ορίζονται ως ακολούθως: αντικαθιστούμε προσωρινά κάθε G_c με μια υπερ-κορυφή, ακολούθως βρίσκουμε τις συντομότερες διαδρομές ανάμεσα στις υπερ-κορυφές, και τέλος επιλέγουμε εκείνες τις ακμές που ανήκουν στις συντομότερες διαδρομές. Εάν καταλήξουμε με περισσότερες από $\alpha \cdot f$ διαδρομές, όπου $\alpha > 1$ είναι μια σταθερά, τότε είτε επιλέγουμε αυθαίρετα $\alpha \cdot f$ από αυτές, είτε βρίσκουμε το ελάχιστο γεννητικό δέντρο που εκτείνεται πάνω στις υπερ-κορυφές, και χρησιμοποιούμε τις ακμές του σαν επιλεγμένες. Η πλειοψηφία των κόμβων αναγκάζεται να κινηθεί είτε πάνω στα υπο-γραφήματα (συνωστισμένες περιοχές), είτε κατά μήκος των μονοπατιών που συνδέουν τα υπο-γραφήματα (αγαπημένες διαδρομές).

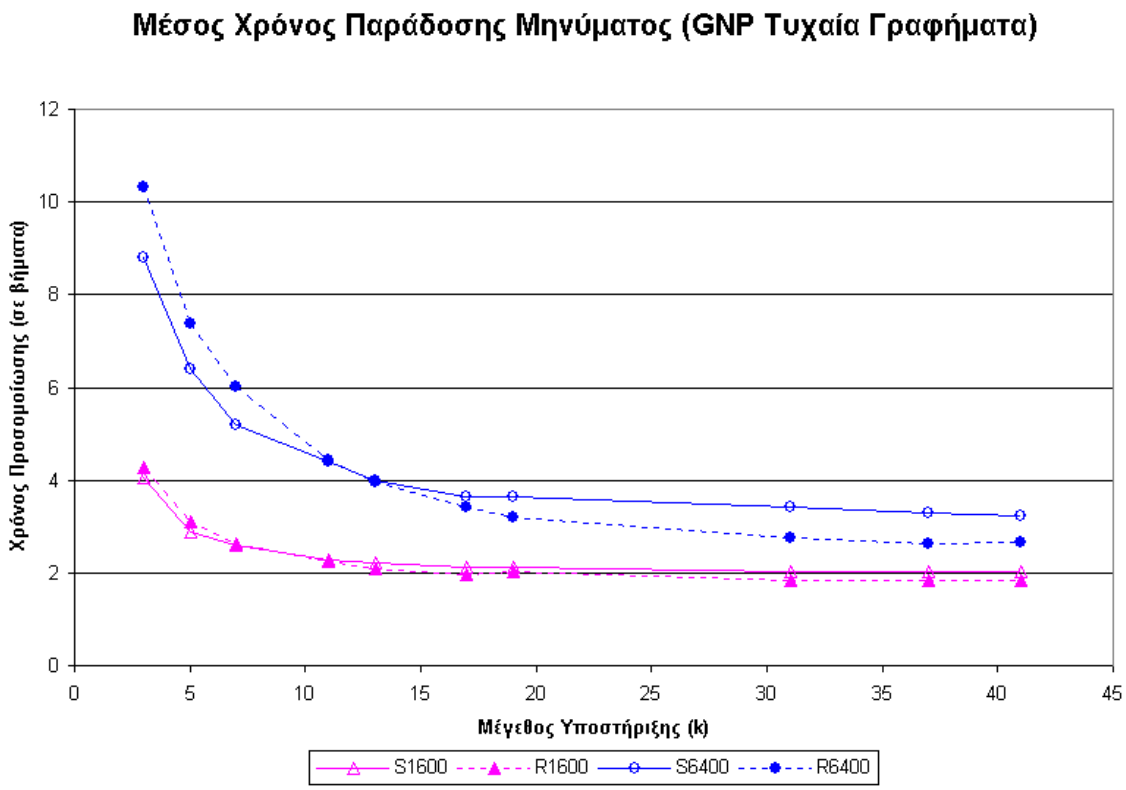
Δημιουργούμε ένα αριθμό από διαφορετικά δεδομένα εισόδου αλλάζοντας τις διάφορες παραμέτρους (όπως είναι για παράδειγμα η ακτίνα c της συνωστισμένης περιοχής, και ο αριθμός f των επιλεγμένων κορυφών). Σημειώστε ότι για κάθε μήνυμα που δημιουργείται, ο παραλήπτης επιλέγεται τυχαία μεταξύ όλων των χρηστών (δηλαδή δεν υπάρχουν σταθερά ζευγάρια αποστολέα-παραλήπτη).

5.2.2 Πειραματικά Αποτελέσματα

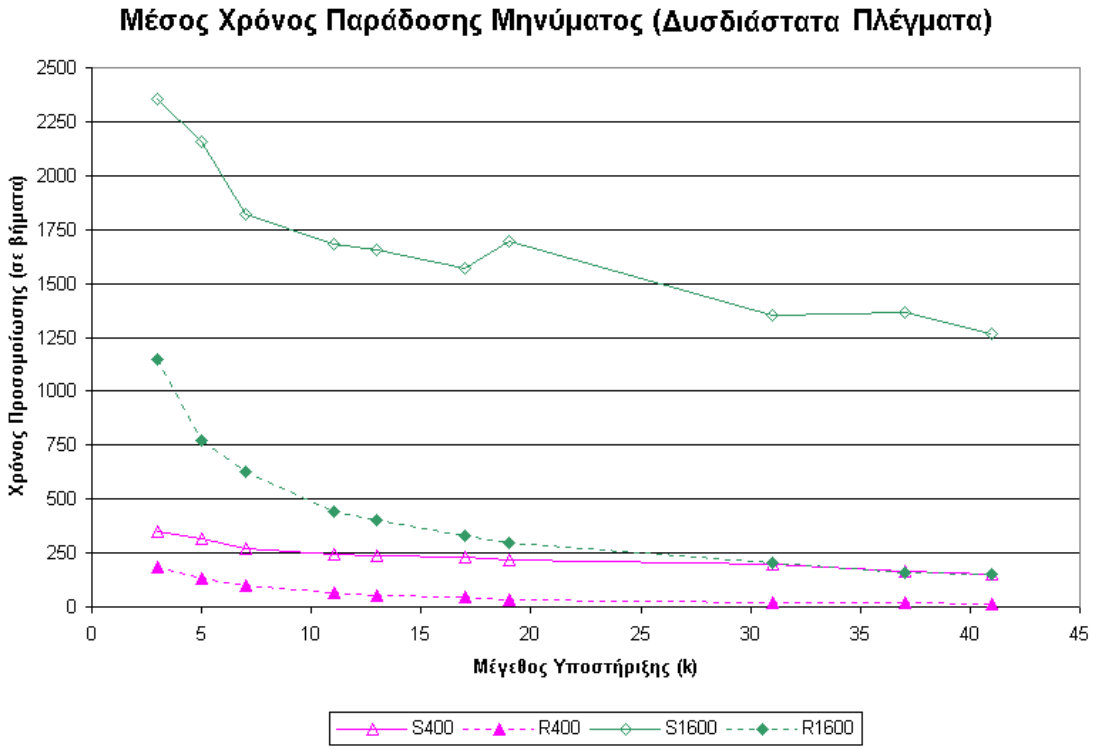
Μετρήσαμε την συνολική καθυστέρηση παράδοσης των μηνυμάτων που ανταλλάζονται μεταξύ ζευγών από αποστολείς και παραλήπτες. Για κάθε μήνυμα που δημιουργούμε, μετράμε τη συνολική καθυστέρηση (σε βήματα προσομοίωσης) μέχρι να φτάσει το μήνυμα στο παραλήπτη. Χρησιμοποιήσαμε αυτή την μέτρηση για να αξιολογήσουμε πειραματικά την απόδοση των δύο διαφορετικών υπο-πρωτοκόλλων διαχείρισης υποστήριξης.

Τα πειράματα, που έχουμε ήδη αναφέρει, με είσοδο τα πέντε διαφορετικά γραφήματα κίνησης παρουσιάζονται στα σχήματα 5.1 έως 5.5. Σε αυτά τα σχήματα, έχουμε προσθέσει τα αποτελέσματα για δύο διαφορετικά στιγμιότυπα του γραφήματος εισόδου σε συνάρτηση με τον αριθμό των κορυφών του (n). Συγκεκριμένα, δώσαμε σαν είσοδο ένα μεγάλο γράφημα (μεγάλο n) και ένα μικρό γράφημα (μικρότερο n). Κάθε καμπύλη στα σχήματα χαρακτηρίζεται από το όνομα 'Px', όπου το P αναφέρεται στο χρησιμοποιούμενο πρωτόκολλο (S για το φιδάκι, και R για τους δρομείς), και το x είναι ένας 3-ψήφιος ή 4-ψήφιος αριθμός που δείχνει την τιμή του n.

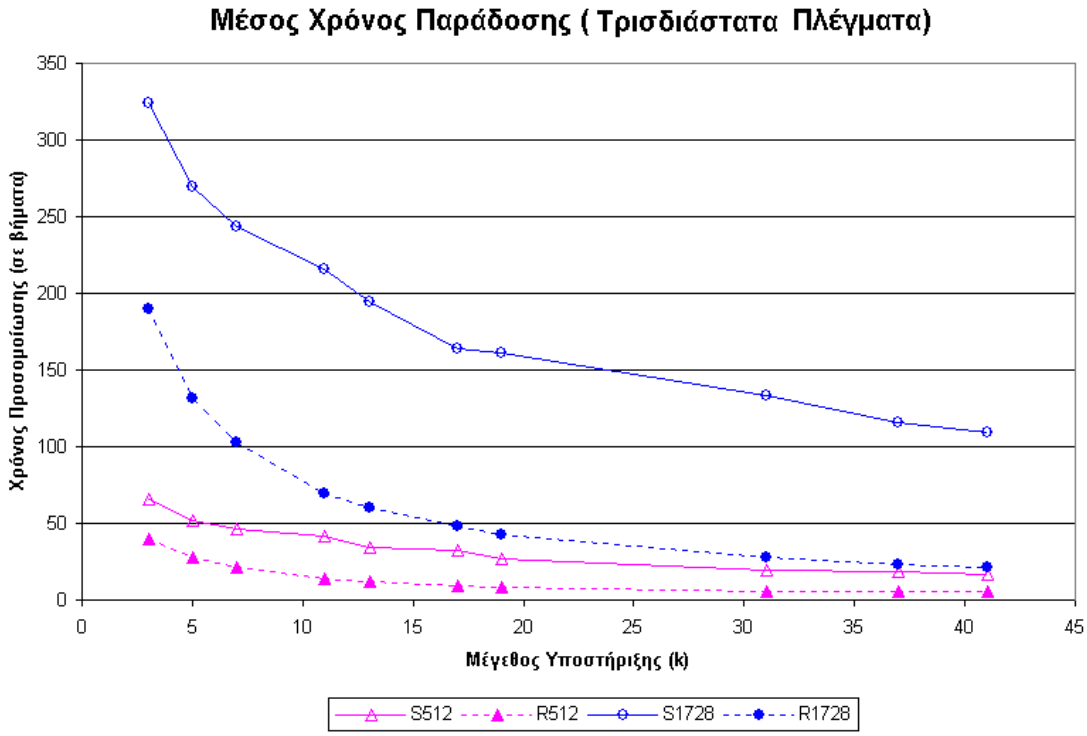
Οι καμπύλες που παίρνουμε για το πρωτόκολλο του φιδιού, επιβεβαιώνουν την θεωρητική ανάλυση της εργασίας [2]. Δηλαδή, ο μέσος χρόνος παράδοσης φθίνει σχετικά γρήγορα όταν το k είναι μικρό, αλλά μετά από κάποια τιμή κατωφλίου σταθεροποιείται και γίνεται σχεδόν ανεξάρτητο από το k. Αυτή η παρατήρηση εφαρμόζεται σε όλες τις εισόδους που χρησιμοποιήσαμε στα πειράματα μας.



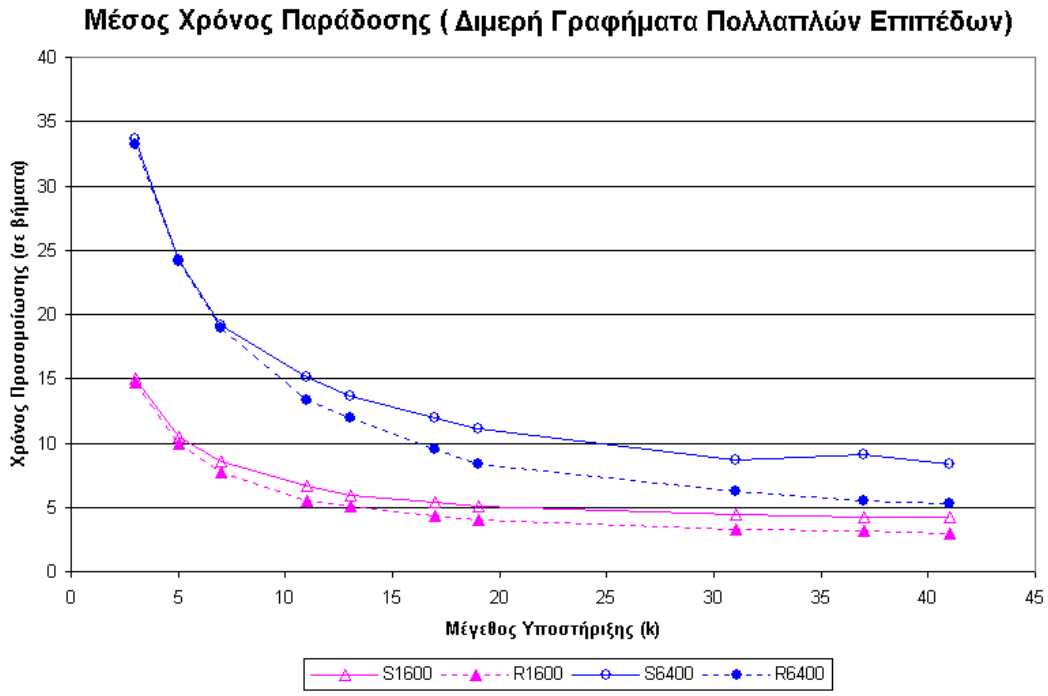
Σχήμα 5.1: Μέσος Χρόνος Παράδοσης σε τυχαία γραφήματα με διαφορετικό αριθμό κορυφών (n) και μέγεθος υποστήριξης (k)



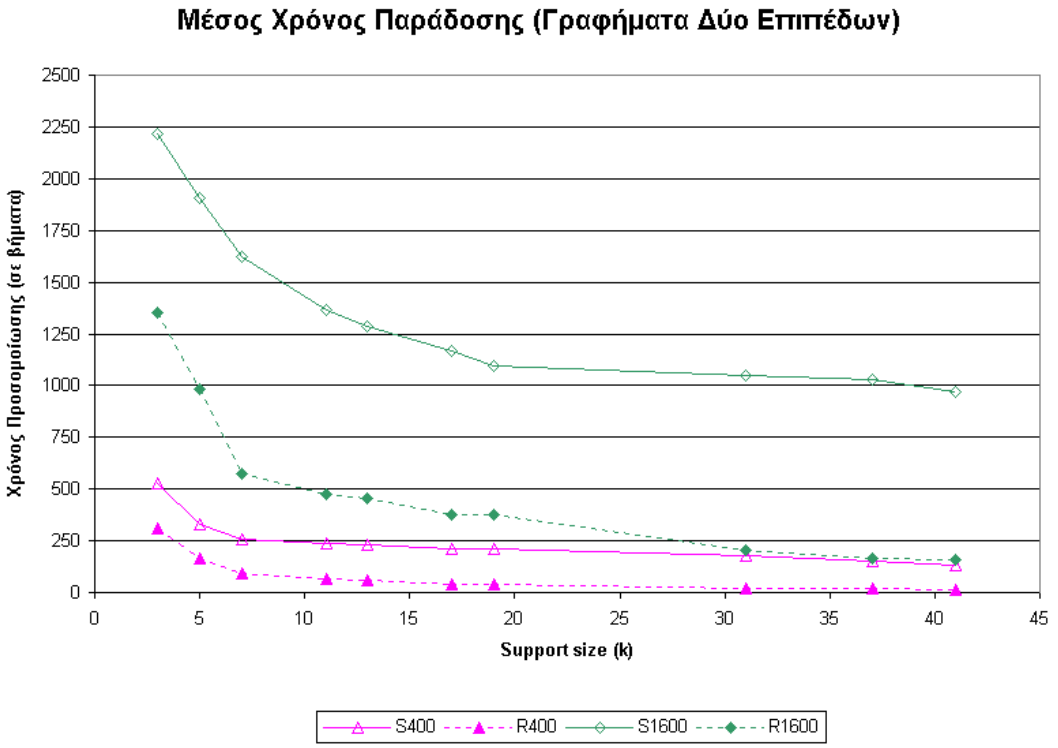
Σχήμα 5.2: Μέσος Χρόνος Παράδοσης σε δυσδιάστατα πλέγματαμε διαφορετικό αριθμό κορυφών (n) και μέγεθος υποστήριξης (k)



Σχήμα 5.3: Μέσος Χρόνος Παράδοσης σε τρισδιάστατα Πλέγματα με διαφορετικό αριθμό κορυφών (n) και μέγεθος υποστήριξης (k)



Σχήμα 5.4: Μέσος Χρόνος Παράδοσης σε διμερή γραφήματα κίνησης πολλαπλών επιπέδων με διαφορετικό αριθμό κορυφών (n) και μέγεθος υποστήριξης (k)

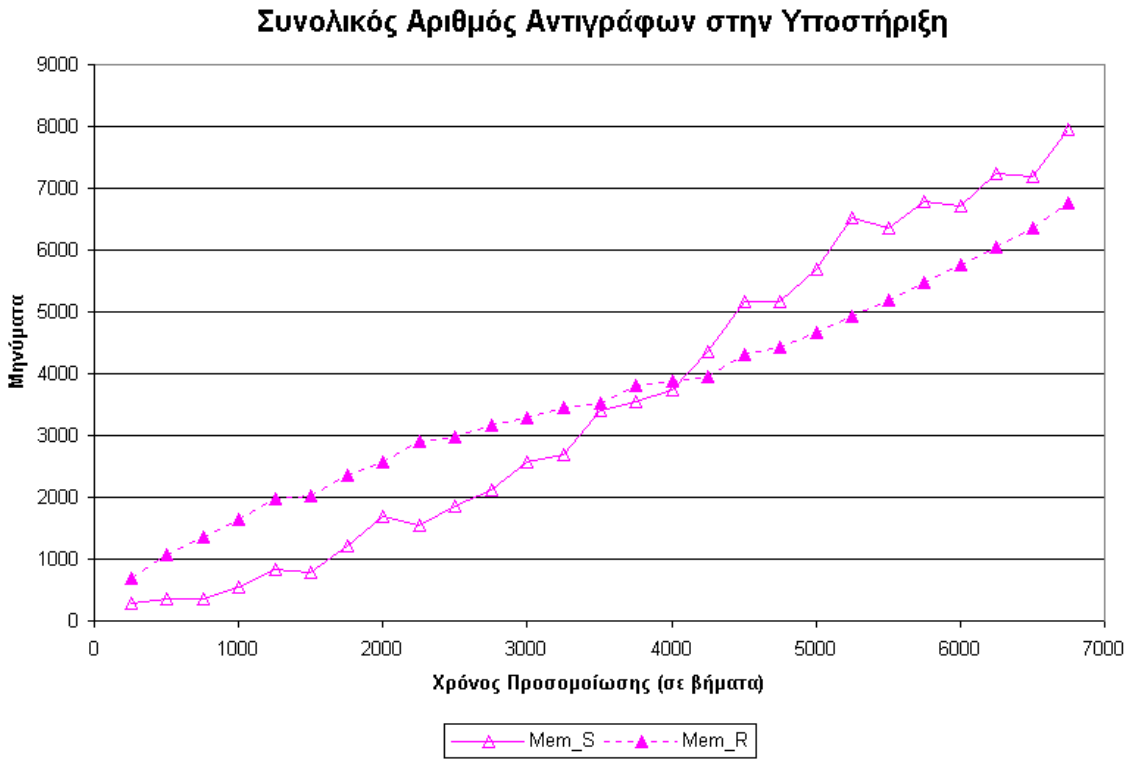


Σχήμα 5.5: Μέσος Χρόνος Παράδοσης σε ημι-τυχαία γραφήματα κίνησης δύο-επιπέδων με διαφορετικό αριθμό κορυφών (n) και διαφορετικά μεγέθη υποστήριξης (k)

Σχετικά με το πρωτόκολλο των δρομέων, παρατηρούμε πρώτα ότι οι καμπύλες του έχουν παρόμοια μορφή με αυτή του πρωτοκόλλου του φιδιού. Δυστυχώς, δεν έχουμε ακόμη καταφέρει να αναλύσουμε θεωρητικά το πρωτόκολλο αυτό έτσι ώστε να επιβεβαιώσουμε την ορθότητα της συμπεριφοράς του. Πάντως, από τα πειραματικά αποτελέσματα υποπτευόμαστε ότι η συμπεριφορά του πρωτοκόλλου των δρομέων είναι παρόμοια με αυτή του φιδιού, με μικρότερες όμως τιμές στους αναμενόμενους χρόνους παράδοσης. Η δεύτερη μας παρατήρηση είναι ότι η απόδοση του πρωτοκόλλου αυτού είναι ελαφρώς καλύτερη από αυτή του φιδιού για τυχαία γραφήματα (εκτός από την περίπτωση που έχουμε πολύ μικρό μέγεθος υποστήριξης, όπως φαίνεται και στο σχήμα 5.1) και για τα διμερή γραφήματα πολλαπλών επιπέδων (σχήμα 5.4). Όμως, το πρωτόκολλο των δρομέων είναι σημαντικά καλύτερο στις περιπτώσεις που το γράφημα κίνησης που χρησιμοποιούμε είναι πιο δομημένο (δηλαδή στα 2D και 3D πλέγματα, και στα γραφήματα κίνησης δύο επιπέδων, όπως φαίνεται στα σχήματα 5.2, 5.3, και 5.5). Μια εξήγηση αυτού του φαινομένου είναι ότι τα δομημένα γραφήματα έχουν μικρότερη δεύτερη ιδιοτιμή από ότι τα λιγότερο δομημένα. Μια μικρή τιμή για αυτή την ιδιοτιμή κάνει το μέσο χρόνο παράδοσης μηνύματος να είναι περισσότερο εξαρτημένος από τη κρυφή σταθερά στην ασυμπτωτική ανάλυση του πρωτοκόλλου του φιδιού ([2]), η οποία είναι προφανώς μεγαλύτερη από την αντίστοιχη σταθερά του πρωτοκόλλου των δρομέων.

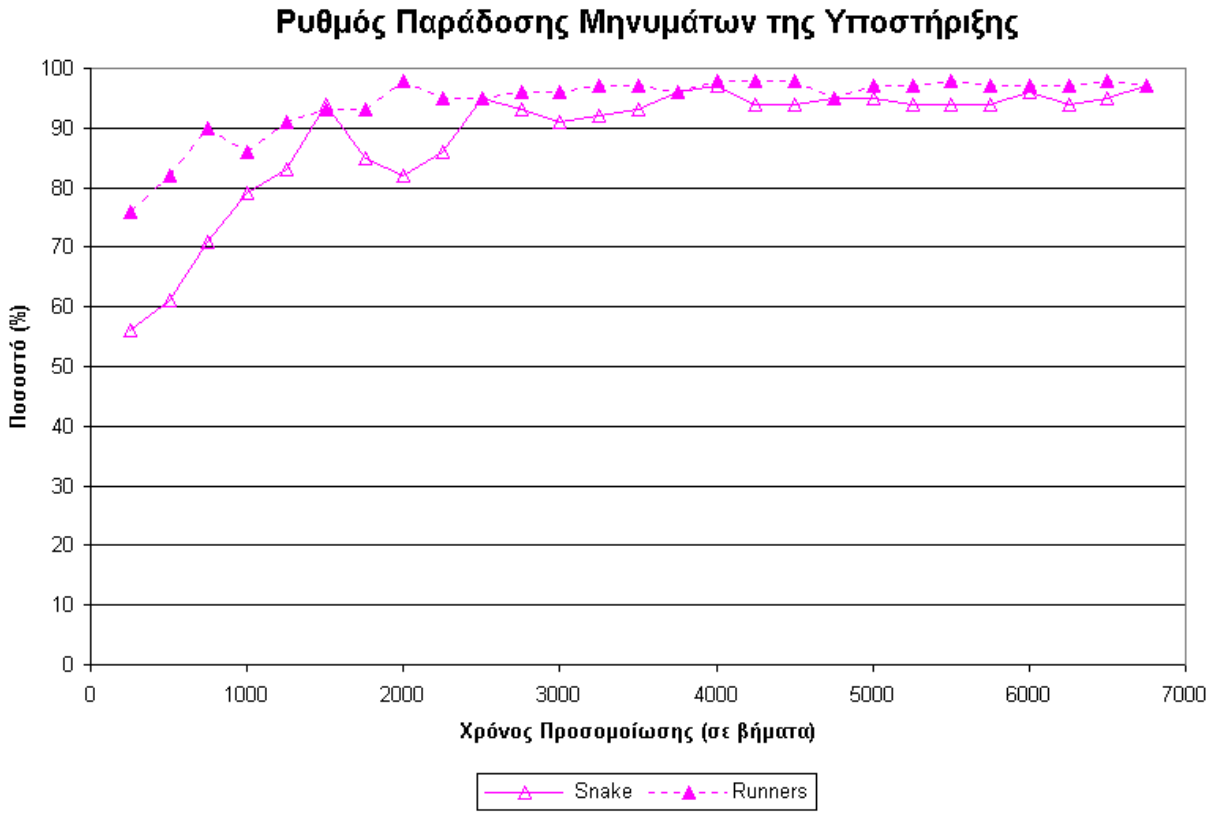
Μια ενδιαφέρουσα παρατήρηση αφορά την περίπτωση των γραφημάτων κίνησης δύο επιπέδων (σχήμα 5.5), όπου τα αποτελέσματα φαίνονται παρόμοια με αυτά των πλεγμάτων. Είναι ενδιαφέρον να παρατηρήσουμε ότι, σε αυτή την περίπτωση όπου οι κόμβοι δεν εκτελούν τυχαίους περιπάτους (όπως στην περίπτωση των πλεγμάτων), αλλά η κίνηση τους είναι περισσότερο περιορισμένη (χωρίς να παύει να είναι αυθόρμητη), τα χαρακτηριστικά της απόδοσης και των δύο πρωτοκόλλων παραμένουν αναλλοίωτα. Άρα, μπορούμε να συμπεράνουμε ότι στην περίπτωση των πιο δομημένων εισόδων, η υπόθεση ότι οι κόμβοι εκτελούν διαρκώς τυχαίους περιπάτους πάνω στο γράφημα κίνησης G δεν επηρεάζει την απόδοση. Πιστεύουμε ότι θα είναι ενδιαφέρον να εισάξουμε έξυπνους μηχανισμούς στο υπο-πρωτόκολλο διαχείρισης υποστήριξης έτσι ώστε να εκμεταλλευόμαστε αυτή την συμπεριφορά των κόμβων.

Σε αντίθεση με την εργασία [2], στην εργασία [1] έχουμε λάβει υπόψιν το βαθμό χρησιμοποίησης του πρωτοκόλλου υποστήριξης, δηλαδή τον συνολικό αριθμό αντιγράφων από μηνύματα που αποθηκεύονται στη δομή της υποστήριξης σε κάθε χρονική στιγμή. Επίσης, έχουμε μετρήσει τον ρυθμό παράδοσης μηνυμάτων, τον οποίο παρουσιάζουμε στο επόμενο σχήμα.



Σχήμα 5.6: Ο συνολικός αριθμός αντιγράφων μηνυμάτων μέσα στην δομή της υποστήριξης σε συνάρτηση με το χρόνο προσομοίωσης (σε βήματα).

Στο σχήμα 5.6 παρουσιάζεται ο συνολικός αριθμός αντιγράφων μηνυμάτων που δημιουργούνται από το κάθε πρωτόκολλο, για τα πρώτα 7000 βήματα προσομοίωσης σε ένα διμερές γράφημα πολλαπλών διαδρομών με n = 6400 και k = 15. Παρόμοια αποτελέσματα παίρνουμε και για διαφορετικές εισόδους και διαφορετικά μεγέθη εισόδου και υποστήριξης. Παρατηρούμε ότι το πρωτόκολλο του φιδιού δημιουργεί αρχικά λιγότερα αντίγραφα από ότι το αντίστοιχο των δρομέων. Ενώ προχωρά η προσομοίωση, ο ρυθμός δημιουργίας αντιγράφων μηνυμάτων στο πρωτόκολλο του φιδιού αυξάνεται γρηγορότερα από ότι στο πρωτόκολλο των δρομέων. Αυτό μπορεί να εξηγηθεί μερικώς, λαμβάνοντας υπόψιν το μέσο χρόνο παράδοσης μηνύματος. Στην περίπτωση του φιδιού, γενικά ο χρόνος παράδοσης είναι χειρότερος από αυτόν των δρομέων, που σημαίνει ότι σε αυτή την περίπτωση θα απαιτείται περισσότερος χρόνος για να συναντήσει ένα αποστολέα. Επιπρόσθετα, κάθε μήνυμα (και τα αντίγραφα του) θα παραμένει για περισσότερο χρόνο μέσα στη δομή της υποστήριξης (φιδάκι), μέχρι να συναντήσει τον κατάλληλο παραλήπτη R. Επομένως, στην αρχή το πρωτόκολλο του φιδιού δημιουργεί λιγότερα αντίγραφα, αφού έχει λάβει λιγότερα μηνύματα, ενώ όσο προχωρά ο χρόνος, ο συνολικός αριθμός των αντιγράφων αυξάνεται αφού υπάρχουν περισσότερα μηνύματα που περιμένουν να παραδοθούν. Αυτή η παρατήρηση μας οδηγεί στο συμπέρασμα ότι το πρωτόκολλο των δρομέων διαχειρίζεται πιο αποδοτικά τους διαθέσιμους πόρους, τουλάχιστον όσο αφορά την μνήμη.



Σχήμα 5.7: Ο ρυθμός παράδοσης μηνυμάτων της υποστήριξης σε συνάρτηση με το χρόνο προσομοίωσης (σε βήματα)

Ένας άλλος τρόπος για να αξιολογήσουμε την απόδοση των δύο πρωτοκόλλων είναι να μετρήσουμε τον ρυθμό παράδοσης μηνυμάτων. Στο σχήμα 5.7 φαίνεται ο συνολικός ρυθμός παράδοσης μηνυμάτων (επί τις εκατό του συνόλου) σε συνάρτηση με τον χρόνο προσομοίωσης (σε βήματα) για είσοδο ένα 3D πλέγμα με n = 6400, και k = 15 (παρόμοια είναι τα αποτελέσματα και για άλλες εισόδους). Το πρωτόκολλο του φιδιού είναι πιο αργό από το αντίστοιχο πρωτόκολλο των δρομέων για τους ίδιους λόγους που αναφέραμε και προηγουμένως. Ενώ ο χρόνος προσομοίωσης αυξάνεται, τα δύο πρωτόκολλα φτάνουν σε υψηλούς ρυθμούς παράδοσης μηνυμάτων. Παρόλα αυτά, είναι προφανές ότι το πρωτόκολλο των δρομέων φτάνει στους υψηλούς αυτούς ρυθμούς παράδοσης πιο γρήγορα από το αντίστοιχο πρωτόκολλο του φιδιού. Αυτό μας παρέχει και μια εξήγηση για το συνολικό αριθμό αντιγράφων μηνυμάτων που αποθηκεύονται στα μέλη της υποστήριξης στο πρωτόκολλο του φιδιού. Όπως έχουμε ήδη αναφέρει, το υπο-πρωτόκολλο συγχρονισμού υποστήριξης P_3 χρειάζεται O(k) χρόνο για να παράξει k αντίγραφα για κάθε μήνυμα. Άρα, αφού ο ρυθμός παράδοσης του φιδιού είναι αρχικά μικρός, ο συνολικός αριθμός αντιγράφων μηνυμάτων θα είναι επίσης μικρός. Όσο ο ρυθμός αυτός αυξάνεται με τον χρόνο, το P_3 θα δημιουργεί περισσότερα αντίγραφα μηνυμάτων και έτσι θα αυξάνει το συνολικό αριθμό αντιγράφων μηνυμάτων που αποθηκεύονται στη δομή της υποστήριξης.

6. Συμπεράσματα

Σε αυτό το Κεφάλαιο αναφέρουμε τα συμπεράσματα που εξάγονται από τη θεωρητική και πειραματική έρευνα που έγινε στα πλαίσια της Διπλωματικής αυτής Εργασίας. Επίσης αναφέρουμε μερικές μελλοντικές κατευθύνσεις, δηλαδή περαιτέρω σημεία τα οποία θα θέλαμε να μελετήσουμε, αλλά λόγω πίεσης χρόνου δεν τα καταφέραμε.

6.1 Αποτελέσματα

Τα βασικότερα αποτελέσματα που εξάγονται από αυτή τη διπλωματική αφορούν τη πειραματική μελέτη των δύο βασικών πρωτοκόλλων, του *Φιδιού* και των *Δρομέων*. Όπως έχουμε δει στο Κεφάλαιο 5, στη περίπτωση που το δίκτυο μας χαρακτηρίζεται από την πολύ μεγάλη "κινητικότητα" των ασύρματων σταθμών (mh), τότε η πιθανοθεωρητική προσέγγιση των πρωτοκόλλων του *Φιδιού* και των *Δρομέων* μπορούν να δώσουν πολύ ικανοποιητικά αποτελέσματα.

Έχουμε ακόμη δείξει ότι αυτά τα ημι-επιτακτικά πρωτόκολλα εγγυούνται ικανοποιητική λειτουργία, καθώς και βέβαιη παράδοση μηνύματος, χρησιμοποιώντας ένα μικρό μόνο μέρος του συνολικού αριθμού των κόμβων.

Έχουμε επαναλάβει τα πειράματα μας σε αρκετά είδη γραφημάτων, τα οποία αντιπροσωπεύουν διάφορα περιβάλλοντα καθώς και διαφορετικές συνθήκες σε αυτά. Όλα τα πειράματα έχουν δείξει ότι τα δύο αυτά πρωτόκολλα έχουν ικανοποιητική συμπεριφορά. Η συμπεριφορά τους είναι παρόμοια για όλα σχεδόν τα είδη γραφημάτων κίνησης, καθώς και για όλα τα μεγέθη τέτοιων γραφημάτων.

Μεταξύ τους τα δύο πρωτόκολλα συμπεριφέρονται με παρόμοιο τρόπο, όπως φαίνεται και από τις καμπύλες των πειραματικών τους αποτελεσμάτων. Σχεδόν σε κάθε είσοδο γραφήματος κίνησης, το πρωτόκολλο των *Δρομέων* συμπεριφέρεται καλύτερα, αφού δίνει μικρότερο μέσο χρόνο παράδοσης μηνύματος.

Τέλος, όπως φαίνεται από το γράφημα του σχήματος 5.7, το πρωτόκολλο των *Δρομέων* έχει από την αρχή μεγαλύτερο ρυθμό παράδοσης μηνυμάτων, ενώ η διαφορά αυτή μικραίνει όσο αυξάνει ο χρόνος. Σε κάθε περίπτωση πάντως, ο ρυθμός αυτός είναι πάντοτε μεγαλύτερος στη περίπτωση του πρωτοκόλλου των *Φιδιών*.

Ένα έμμεσο αποτέλεσμα αυτής της Διπλωματικής εργασίας είναι και ο προσομοιωτής που κατασκευάστηκε για την πειραματική μελέτη των προτεινόμενων πρωτοκόλλων. Ο προσομοιωτής αυτός κατασκευάστηκε έτσι ώστε να είναι αρκετά γενικός και να μπορεί με

σχετική ευκολία να υλοποιεί τα διάφορα πρωτόκολλα που προτείνονται για τη δρομολόγηση των ad-hoc δικτύων. Είναι πεποίθηση μου, ότι το σύστημα αυτό μπορεί με τις κατάλληλες βελτιώσεις και διορθώσεις να αποτελέσει ένα χρήσιμο εργαλείο προσομοίωσης και εξαγωγής πειραματικών αποτελεσμάτων.

6.2 Μελλοντικές Κατευθύνσεις

Σε αυτή τη παράγραφο αναφέρουμε τα πράγματα που θα θέλαμε να κάνουμε αλλά λόγω έλλειψης χρόνου αυτό δεν ήταν δυνατό.

Από πλευράς πρωτοκόλλων, πολύ σημαντικό θα ήταν να έχουμε θεωρητική ανάλυση του πρωτοκόλλου των *Δρομέων*. Ως γνωστό έχει ήδη αναλυθεί θεωρητικά το πρωτόκολλο του *Φιδιού (στην [2]),* και τα αποτελέσματα της θεωρητικής ανάλυσης συμφωνούν με τα πειραματικά μας αποτελέσματα.

Πολύ ενδιαφέρον θα ήταν επίσης να υλοποιούσαμε ένα πρωτόκολλο παρόμοιο με αυτό των *Δρομέων*, αλλά το οποίο να υποστηρίζει δυναμική μετάβαση των κόμβων του δικτύου από και προς τη κατάσταση υποστήριξης (*SUPPORT*). Δηλαδή, οι κόμβοι του δικτύου να αποφασίζουν (με κάποιο πιθανοτικό τρόπο) και να εκλέγουν τοπικά κάποιο κόμβο ο οποίος θα πηγαίνει προσωρινά στη κατάσταση υποστήριξης (*SUPPORT*). Με αυτό το τρόπο το δίκτυο μας θα είναι non-compulsory (και όχι semi-compulsory), αφού οι κόμβοι της υποστήριξης δεν θα είναι μόνιμοι αλλά θα επιλέγονται δυναμικά αναλόγως των συνθηκών. Το πρωτόκολλο αυτό έχει ήδη συζητηθεί αρκετά και ελπίζουμε να μελετηθεί σύντομα και σε πειραματικό επίπεδο σε κάποια μελλοντική εργασία.

Τέλος, θα μιλήσουμε για τις διάφορες βελτιώσεις που μπορούν να γίνουν στον προσομοιωτή. Οι βελτιώσεις αυτές αφορούν κυρίως πρόσθετα χαρακτηριστικά τα οποία θα του επιτρέπουν να υλοποιεί καινούρια πρωτόκολλα. Για παράδειγμα πολύ ενδιαφέρον θα ήταν η προσθήκη κάποιου συστήματος που να πληροφορεί το κόμβο για τη θέση του (π.χ. με τη προσθήκη μιας μεθόδου GET_GPS_POS()), και η οποία θα μπορεί να χρησιμοποιηθεί από τα πιο σύγχρονα πρωτόκολλα όπως είναι το LAR.

Βιβλιογραφία

[1]. I. Chatzigiannakis, S. Nikoletseas, N. Paspallis, P. Spirakis, and C. Zaroliagis. "An Experimental Study of Basic Communication Protocols in Ad-hoc Mobile Networks*" in Algorithm Engineering – WAE 2001, Lecture notes in Computer Science 2141 (Springer-Verlag 2001).*

[2]. I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis. Analysis and Experimental Evaluation of an Innovative and Efficient Routing Approach for Ad-hoc Mobile Networks. *In Proc. 4th Annual Workshop on Algorithmic Engineering – WAE'00 – 2000.*

[3]. K. Hatzis, G. Pentaris, P. Spirakis, B. Tampakas, and R. Tan. Fundamental Distributed Protocols in Mobile Networks. *In Proc. 11th Annual Symposium on Parallel Algorithms and Architectures – SPAA'99, 1999.*

[4]. T. Larsson, and N. Hedman. "Routing Protocols in Wireless Ad-hoc Networks – A Simulation Study". *Lulea University of Technology, Stockholm, 1998 - A Master Thesis.*

[5]. M. Adler, C. Scheideler. Efficient Communication Strategies for Ad-Hoc Wireless Networks. In Proc. *10th Annual Symposium on Parallel Algorithms and Architectures – SPAA'98, 1998.*

[6]. E. Royer, C-K Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. *IEEE Personal Communications, 6(2):46-55, April 1999.*

[7]. Stephen Kent and Randall Atkinson, "Security Architecture for the Internet Protocol", *Internet draft, draft-ietf-ipsec-arch-sec-07.txt, July 1998.*

[8]. Charles E. Perkins and Pravin Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers". In *Proceedings of the SIGCOM '94 Conference on Communications Architecture, protocols and Applications, pages 234-244, August 1994.* A revised version of the paper is available from http://www.cs.umd.edu/projects/mcml/papers/Sigcomm94.ps. (1998-11-29)

[9]. Martha Steenstrup, "Routing in communication networks". New Jersey*, Prentice Hall. ISBN 0-13-010752-2.*

[10]. Stephen Kent and Randall Atkinson, "Security Architecture for the Internet Protocol", *Internet draft, draft-ietf-ipsec-arch-sec-07.txt, July1998.*

[11]. K. Mehlhorn and S. Naher. LEDA: A Platform for Combinatorial and Geometric Computing. *Cambridge University Press, 1999.*

[12]. G. S. Malkin and M. E. Steenstrup. Distance-Vector Routing. *In M. Steenstrup, editor, Routing in Communications Networks, pages 83-98. Prentice Hall, 1995.*

[13]. J. Moy. Link-State Routing. In M. Steenstrup, editor, *Routing in Communication Networks, pages 135-157. Prentice Hall, 1995.*

[14]. C. E. Perkins. IP Mobility Support. *RFC 2002, October 1996.*

[15]. C. E. Perkins. Mobile IP. *IEEE Communications Magazine, 3(5):84-99, May 1997.*

[16]. Zygmount J. Haas and Marc R. Pearlman, "The Zone Routing Protocol (ZRP) for Ad-hoc Networks", *Internet draft, draft-ietf-manet-zone-zrp-01.txt, August 1998*.

[17]. Vincent D. Park and M. Scott Corson, "Temporally-Ordered Routing Algorithm (TORA) Version 1: Functional Specification". *Internet draft, draft-ietf-manet-tora-spec-01.txt, August 1998.*

[18]. Vincent D. Park and M. Scott Corson, "A performance comparison of the Temporally-Ordered Routing Algorithm and Ideal Link-state routing". *In Proceedings of IEEE Symposium on Computers and Communications '98, June 1998.*

[19]. Scott Corson and Joseph Macker, "Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations". *Internet-Draft, draft-ietf-manet-issues-01.txt, March 1998.*

[20]. I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis. Self-Organizing Ad-Hoc Mobile Networks: The problem of end-to-end communication. A short paper in *Proc. 20th Annual Symposium on Principles of Distributed Computing – PODC'01, 2001.*

[21]. N. A. Lynch. Distributed Algorithms. *Morgan Kaufmann Publishers Inc., 1996*.

[22]. Haiyun Luo, Songwu Lu, Vaduvur Bharghavan, "A New Model for Packet Scheduling in Multihop Wireless Networks". In *Proc. of ACM MOBICOM 2000, Boston, MA, August 2000.*

[23]. Young-Bae Ko and Nitin H. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks". *Tech. Rep. 98-012, CS Dept., Texas A&M University, June 1998.*

[24]. "GPS and precision timing applications". Web site at *http://wwwtmo.external.hp.com/tmo/pia/infinium/PIATop/Notes/English/5965-2791E.html.*

[25]. "Iowa State University GPS page". Web site at *http://www.cnde.iastate.edu/gps.html.*

[26]. "NAVSTAR GPS operations". Web site at *http://tycho.usno.navy.mil/gpsinfo.html.*

[27]. G. Dommety and R. Jain, "potential networking applications of global positioning systems (GPS)", *Tech. Rep. TR-24, CS Dept., The Ohio State University, April 1996.*

[28]. B. Parkinson and S. Gilbert, "NAVSTAR: global positioning system – ten years later", in *Proceeding of IEEE, pp. 1177-1186, 1983.*

[29]. Charles E. Perkins, *"Ad-hoc Networking" book*, Addison-Wesley.

[30]. D. Johnson. "Routing in Ad-Hoc Networks of Mobile Hosts". In *Proc. Of the IEEE Workshop on Mobile Computing Systems and Applications, December 1994.*

Παράρτημα

Α. Συντμήσεις

AP	Access Point
BS	Base Station
CDMA	Code Division Multiple Access
CSMA	Carrier Sense Multiple Access
DECT	Digital Enhanced Cordless Telecommunications
FDMA	Frequency Division Multiple Access
GSM	Global System for Mobile
LAN	Local Area Network
MAC	Medium Access Control
PDA	Personal Digital Assistant
QoS	Quality of Service
TDMA	Time Division Multiple Access
UMTS	Universal Mobile Communication System
WAN	Wide Area Network

Β. Εγχειρίδιο του Συστήματος Προσομοίωσης

Σε αυτή τη παράγραφο δίνουμε τις κατάλληλες πληροφορίες που απαιτούνται για τον προγραμματισμό και τη χρήση του συστήματος προσομοίωσης. Για την κατανόηση της χρήσης και της λειτουργίας του συστήματος μπορούν επίσης να μελετηθούν τα παραδείγματα του Παραρτήματος Γ.

Η βασική λειτουργία του συστήματος είναι να προσομοιώνει τη συμπεριφορά ενός κινητού κόμβου mh που υλοποιεί ένα συγκεκριμένο πρωτόκολλο. Για το σκοπό αυτό επεκτείνουμε τη κλάση mh με χρήση κληρονομικότητας. Δηλαδή, από την αρχική κλάση mh μπορούμε, υλοποιώντας τη μέθοδο executeProtocol(), να υλοποιήσουμε κινητούς κόμβους που να εκτελούν συγκεκριμένα πρωτόκολλα. Για παράδειγμα, στο Παράρτημα Γ φαίνεται ο κώδικας με τον οποίο υλοποιήσαμε τους κόμβους τύπου sender-receiver, runner, και snake.

Στη μέθοδο executeProtocol() χρησιμοποιούμε διάφορες λειτουργίες που παρέχονται από το προσομοιωτή και περιγράφονται πιο κάτω. Οι λειτουργίες αυτές χρησιμοποιούν με τη σειρά τους λειτουργίες που παρέχουν οι κλάσεις environment και message, αλλά αυτό γίνεται με τρόπο διάφανο στο χρήση και έτσι δεν θα μας απασχολήσει σε αυτό το σημείο.

Στη συνέχεια παρουσιάζουμε τις βασικές αυτές λειτουργίες μαζί με μια σύντομη περιγραφή τους. Ενθαρρύνουμε τον ενδιαφερόμενο χρήστη του προσομοιωτή να μελετήσεις τα παραδείγματα του Παραρτήματος Γ για καλύτερη κατανόηση της χρήσης του συστήματος.

bool IS_ACTIVE()

Αυτή η συνάρτηση επιστρέφει true αν ο mh μας είναι σε ενεργοποιημένη κατάσταση (ACTIVE flag == true), διαφορετικά επιστρέφει false (ACTIVE flag == false).

bool IS_SUPPORT()

Αυτή η συνάρτηση επιστρέφει true αν ο mh μας είναι σε κατάσταση υποστήριξης (SUPPORT flag == true), διαφορετικά επιστρέφει false (SUPPORT flag == false).

void SET_ACTIVE()

Θέτει τον mh σε ενεργό κατάσταση (ACTIVE flag = true).

void SET_INACTIVE()

Θέτει τον mh σε μη ενεργό κατάσταση (ACTIVE flag = false).

void SET_SUPPORT()

Θέτει τον mh σε κατάσταση υποστήριξης (SUPPORT flag = true).

void MOVE_RANDOMLY()

Μετακινεί τον mh σε μια κορυφή του γραφήματος κίνησης που επιλέγεται τυχαία και ισοπίθανα ανάμεσα στις γειτονικές κορυφές της κορυφής που βρίσκεται αρχικά ο mh.

void MOVE_AT(node next_node)

Μετακινεί τον mh στη κορυφή next_node του γραφήματος κίνησης. Εάν η κορυφή αυτή δεν είναι έγκυρη τότε εμφανίζεται ανάλογο μήνυμα λάθους στην έξοδο του προσομοιωτή.

int TRANSMIT(message * msg, int receiver = EVERYONE)

Αυτή η μέθοδος χρησιμοποιείται για τη μετάδοση ενός μηνύματος msg προς τον κόμβο receiver. Η μέθοδος παίρνει σαν παράμετρο ένα δείχτη στο μεταδιδόμενο μήνυμα, και μια ταυτότητα του κόμβου παραλήπτη. Εάν δεν καθοριστεί κάποιος παραλήπτης, τότε το μήνυμα λαμβάνεται από όλους τους κόμβους που βρίσκονται εντός ακτίνας μετάδοσης.

message * POP_MSG()

Η μέθοδος αυτή χρησιμοποιείται για την παραλαβή ενός μηνύματος από το σωρό των εισερχόμενων μηνυμάτων. Όταν ένα μήνυμα μεταδίδεται από ένα mh, τότε το μήνυμα αυτό αντιγράφεται αυτόματα στο σωρό του κάθε mh που βρίσκεται εντός ακτίνας επικοινωνίας. Ακολούθως, για να επεξεργαστούμε αυτό το μήνυμα πρέπει να το ανακτήσουμε από το σωρό χρησιμοποιώντας αυτήν ακριβώς την εντολή.

message * NEW_MSG(int type, int target)

Η εντολή αυτή δημιουργεί ένα νέο μήνυμα. Στις παραμέτρους της καθορίζουμε το τύπο του μηνύματος, καθώς και το τελικό προορισμό. Ο προορισμός αυτός θα πρέπει να είναι το ID ενός έγκυρου προορισμού. Η μέθοδος επιστρέφει ένα δείχτη στο καινούριο μήνυμα που δημιουργήθηκε. Οι δυνατοί τύποι ενός μηνύματος καθορίζονται στο αρχείο *constants.h*.

int DEL_MSG(message * msg)

Η μέθοδος αυτή απλά διαγράφει ένα μήνυμα. Το μήνυμα αυτό καθορίζεται μέσω ενός δείχτη σε αυτό που δίνεται σαν παράμετρος στη μέθοδο μας. Αξίζει να σημειωθεί ότι σε καμιά περίπτωση δεν κινδυνεύει το σύστημα μας με “κατάρρευση” λόγω λανθασμένης διαγραφής, αφού όλα τα μηνύματα είναι τοπικά αντίγραφα και άρα δεν υπάρχει περίπτωση να επέμβουμε σε τμήμα της μνήμης όπου δεν είμαστε “εξουσιοδοτημένοι”.

int GET_NEIGHBOURS_IDS(list< int > &)

Αυτή η μέθοδος επιστρέφει μια λίστα που περιέχει τα IDs όλων των γειτονικών κόμβων του συγκεκριμένου mh. Η λίστα αυτή είναι μια έτοιμη δομή που παρέχει το σύστημα της LEDA. Επιλέξαμε αυτή τη συγκεκριμένη δομή αφού μας παρέχει έτοιμες πολλές λειτουργίες που μπορούμε να εφαρμόσουμε σε μια λίστα και μάλιστα με αποδοτικό τρόπο.

int GET_SUPPORT_NEIGHBOURS_IDS(list< int > &)

Αυτή η μέθοδος είναι μια ειδική περίπτωση της προηγούμενης μεθόδου. Συγκεκριμένα, η μέθοδος αυτή επιστρέφει μια λίστα που περιέχει τα IDs όλων των γειτονικών κόμβων του τρέχον mh που ανήκουν στην υποστήριξη (SUPPORT flag == true).

int GET_RANDOM_ID(int max_id = 0)

Αυτή η μέθοδος, όπως και η επόμενη της, δεν έχει άμεση σχέση με την υλοποίηση του πρωτοκόλλου αλλά είναι περισσότερο βοηθητικού χαρακτήρα. Συγκεκριμένα επιστρέφει ένα τυχαίο ID μεταξύ των τιμών 1 και max_id. Εάν δεν καθορίσουμε κάποια παράμετρο, ή αν δώσουμε παράμετρο 0, τότε η μέθοδος επιστρέφει ένα ID μεταξύ όλων των έγκυρων.

int GET_RANDOM_DECISION(float prob)

Αυτή η μέθοδος, επιστρέφει true με πιθανότητα prob, και false με πιθανότητα 1 – p. Το p είναι μια παράμετρος που δίνει ο χρήστης και παίρνει τιμές στο διάστημα [0, 1].

int MSG_RCVD_SUCC(message *)

Αυτή η μέθοδος χρησιμοποιείται από το σύστημα για ενημέρωση των στατιστικών δεδομένων του προσομοιωτή. Απλά, το πρωτόκολλο καλεί αυτή την μέθοδο όταν λάβει με επιτυχία ένα μήνυμα που ο προορισμός του ήταν ο τρέχον κόμβος.

long GET_GLOBAL_TIME()

Αυτή η μέθοδος επιστρέφει τη τρέχον χρονική στιγμή, η οποία είναι κοινή σε ολόκληρο το σύστημα (δηλαδή ίδια για όλους τους mh). Το περιβάλλον προσομοίωσης διατηρεί μια μεταβλητή που ουσιαστικά μετράει τα βήματα προσομοίωσης που έχουν ήδη συμπληρωθεί. Η μεταβλητή αυτή είναι τύπου long και χρησιμοποιείται κυρίως για στατιστικούς λόγους.

Γ. Κώδικας Υλοποίησης των Προτεινόμενων Πρωτοκόλλων

Σε αυτή την παράγραφο παρουσιάζουμε τον κώδικα που γράψαμε για την υλοποίηση κινητών κόμβων τριών διαφορετικών τύπων. Σε αυτό το κώδικα κάνουμε εκτενή χρήση των δομών που μας προσφέρει το σύστημα της LEDA, μαζί με τις λειτουργίες τους. Για περισσότερες πληροφορίες σχετικά με τη LEDA και τη χρήση της μπορείτε να συμβουλευθείτε την σελίδα της στο Internet στο ακόλουθο URL: http://www.algorithmic-solutions.com/as_html/products/leda/products_leda.html

Στις ακόλουθες παραγράφους παραθέτουμε τον κώδικα για τρεις διαφορετικούς τύπους κινητών κόμβων. Ο πρώτος τύπος είναι ο *sender-receiver*, ο οποίος υλοποιεί ένα κόμβο που απλά στέλνει και λαμβάνει μηνύματα. Ο δεύτερος και τρίτος τύπος κινητών κόμβων είναι οι *runner* και *snake*, οι οποίοι υλοποιούν τους αντίστοιχους κόμβους υποστήριξης για το κάθε πρωτόκολλο.

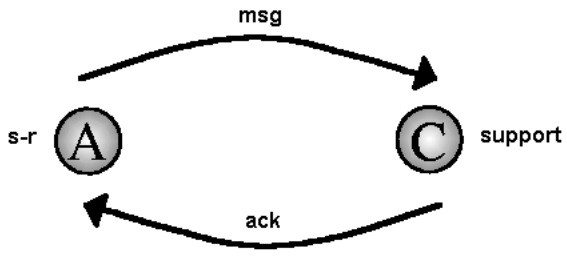
Για την κατανόηση της λειτουργίας των αλγορίθμων που ακολουθούν θα πρέπει πρώτα να δώσουμε μερικές εξηγήσεις για το τρόπο με τον οποίο ανταλλάζουν μηνύματα οι κινητοί κόμβοι. Επειδή η μετάδοση ενός μηνύματος δεν γίνεται πάντοτε σωστά, θα πρέπει να χρησιμοποιούμε ένα σύστημα επιβεβαίωσης. Για παράδειγμα, αν στείλουμε ένα μήνυμα σε ένα γείτονα μας Χ, τότε δεν μπορούμε να είμαστε βέβαιοι ότι ο γείτονας αυτός δεν έχει φύγει και άρα πήρε το μήνυμα, εκτός και αν λάβουμε μια επιβεβαίωση από αυτόν.

Προτού όμως παρουσιάσουμε τον κώδικα μας θα πρέπει να μελετήσουμε τα διάφορα προβλήματα κατά τη μετάδοση ενός μηνύματος, καθώς και την αντιμετώπιση τους .

ΦΑΣΗ Ι

Υποθέτουμε ότι οι κόμβοι της υποστήριξης δεν καταστρέφονται (και άρα δεν χάνουν τα μηνύματα τους). Τότε αν μεταδώσουμε ένα μήνυμα και πάρουμε επιβεβαίωση (ack) από ένα κόμβο της υποστήριξης, τότε το μήνυμα αυτό αργά ή γρήγορα θα παραδοθεί στον προορισμό του. Κάθε κόμβος sender-receiver έχει μια λίστα (OUTGOING) στην οποία τοποθετεί τα μηνύματα που πρέπει να αποσταλούν μόλις συναντήσει ένα κατάλληλο κόμβο υποστήριξης.

Ένα παράδειγμα παρουσιάζεται στο σχήμα Ε.1.



Σχήμα Ε.1: Το πρόβλημα της μετάδοσης απο κόμβο sender-receiver σε κόμβο υποστήριξης (support)

- 1η περίπτωση: το μήνυμα έφθασε κανονικά (msg ok)

  η επιβεβαίωση έφθασε κανονικά (ack ok)

  Σε αυτή τη περίπτωση το μήνυμα msg αφαιρείται από τη λίστα OUTGOING.

- 2η περίπτωση: το μήνυμα έφθασε κανονικά (msg ok)

  η επιβεβαίωση όχι (ack X)

  Σε αυτή τη περίπτωση το μήνυμα διατηρείται και το ξαναστέλνουμε μόλις συναντήσουμε κάποιο άλλο κόμβο υποστήριξης.

- 3η περίπτωση: το μήνυμα δεν έφθασε κανονικά (msg X)

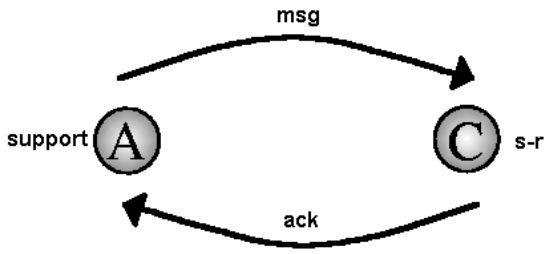
  και φυσικά ούτε η επιβεβαίωση έφθασε (ack X)

  Σε αυτή τη περίπτωση το μήνυμα διατηρείται και το ξαναστέλνουμε μόλις συναντήσουμε κάποιο άλλο κόμβο υποστήριξης.

ΦΑΣΗ ΙΙ

Με παρόμοιο τρόπο αντιμετωπίζουμε και το αντίστοιχο πρόβλημα που εμφανίζεται όταν μεταδίδει ένας κόμβος της υποστήριξης ένα μήνυμα προς τον παραλήπτη του. Κάθε κόμβος υποστήριξης διαθέτει δύο λίστες. Η πρώτη (UNDELIVERED) περιέχει τα μηνύματα που δεν έχουν ακόμη παραδοθεί. Η δεύτερη (ACKNOWLEDGED) περιέχει τα μηνύματα που έχουν παραδοθεί με επιτυχία (επιβεβαιωμένα) στους κατάλληλους παραλήπτες.

Ένα παράδειγμα παρουσιάζεται στο σχήμα Ε.2.



Σχήμα Ε.2: Το πρόβλημα της μετάδοσης απο κόμβο υποστήριξης σε κόμβο sender-receiver

1η περίπτωση: το μήνυμα έφθασε κανονικά (msg ok)

η επιβεβαίωση έφθασε κανονικά (ack ok)

Σε αυτή τη περίπτωση, ο κόμβος υποστήριξης αφαιρεί το μήνυμα msg από την UNDELIVERED, και το προσθέτει στη λίστα ACKNOWLEDGED.

Ο κόμβος s-r στέλνει την επιβεβαίωση ack στο κόμβο υποστήριξης, και ακολούθως διαγράφει το μήνυμα msg.

2η περίπτωση: το μήνυμα έφθασε κανονικά (msg ok)

το μήνυμα όχι (ack X)

Σε αυτή τη περίπτωση, ο κόμβος υποστήριξης δεν κάνει τίποτα, αλλά επαναλαμβάνει τη μετάδοση του μηνύματος msg όταν αυτό είναι δυνατό.

Ο κόμβος s-r στέλνει την επιβεβαίωση ack στο κόμβο υποστήριξης (το οποίο όμως δεν λαμβάνεται), και ακολούθως διαγράφει το μήνυμα msg.

3η περίπτωση: το μήνυμα δεν έφθασε κανονικά (msg X)

και φυσικά ούτε η επιβεβαίωση έφθασε (ack X)

Σε αυτή τη περίπτωση, ο κόμβος υποστήριξης δεν κάνει τίποτα, αλλά επαναλαμβάνει τη μετάδοση του μηνύματος msg όταν αυτό είναι δυνατό.

Ο κόμβος s-r δεν αντιλαμβάνεται την αποτυχημένη μετάδοση και άρα δεν κάνει τίποτα.

Περισσότερες λεπτομέρειες για τη λειτουργία των δύο πρωτοκόλλων (snakes και runners) μπορούν να αναζητηθούν στο 5ο Κεφάλαιο, όπου και παρουσιάζονται αναλυτικά.

Γ.1. Ο Κώδικας Υλοποίησης ενός Χρήστη Τύπου SR – Sender Receiver

```
#ifndef SR_H
#define SR_H

/*
 * define mobile hosts of type sender-receiver
 */
class sr : public mh
{
public:
    // derived constructor overrides base constructor
    sr::sr(environment * my_env, GRAPH<node, edge> * my_motion_graph)
    : mh(my_env, my_motion_graph)
    {
      // initializing...
      SET_ACTIVE();

      _outgoing.clear();
      _received.clear();
      _support_neighbours.clear();
      _messages_sent = 0;
      _messages_received = 0;
    }

    // override protocol implementation function
    int executeProtocol();

    // function used for statistics
    int memoryUsage() { return _outgoing.size() + _received.size(); }

private:
    // Support sensor subprotocol P2
    int subProtocolP2();

    // Generate new message
    int genNewMessages();

    // Process incoming messages
    int procIncMessages();

    // list for outgoing msgs
    set< message >      _outgoing;
    set< message >      _received;

    // list used to hold the support ids currently in neighborhood
    list< int >         _support_neighbours;

    // counter of messages sent
    int                 _messages_sent;
    int                 _messages_received;

}; // end of mh_sr class definition
```

```
/*
 * define 'executeProtocol' function for the derived class mh_simple
 */
int
sr::executeProtocol()
{
    clock_t prev_update = clock();

//-------------communicate with other hosts only if active-------------------

    if(IS_ACTIVE())
    {
      // Based on SEND_PROB and Total number of messages to be sent
      genNewMessages();

      // Use sensor sub-protocol P2 to send messages whenever a node of the
      // support is within transmission range
      subProtocolP2();

      // Process incoming messages
      procIncMessages();
    }

//-----------this part takes care of the mh's motion in the area-------------

    // make a random move on my motion graph
    if(!IS_ASLEEP())
      MOVE_RANDOMLY();

    return clock() - prev_update;
}


//-----this part randomly generates up to a max num of NORM type messages----

int
sr::genNewMessages() {
      if ((rand_int(0,1000000) <= 1000000*SEND_PROB) && (_messages_sent <
MAX_NUM_OF_MSGS_TO_SEND)) {
          // compute a valid id to deliver message at
          int mytarget = getId();
          while(mytarget == getId())
          {
            mytarget = GET_RANDOM_ID(NUM_OF_SRS);
          }

          // generate a message...
          message * msg_p = NEW_MSG(TYPE_NRM, mytarget);

          // ...and add it to outgoing list
          _outgoing.insert(*msg_p);

          // update statistics...
          MSG_SENT();

          // increase counter
          _messages_sent++;
      }
}
```

```
//-----this part is used to send messages whenever a support is around-------

int
sr::subProtocolP2() {
      // get a set with support neighbours
      GET_SUPPORT_NEIGHBOURS_IDS(_support_neighbours);

      if(_support_neighbours.size() > 0)
      {
          // select a random support neighbour to deliver your messages at
          int support_id = _support_neighbours.front();

          message msg;
          forall(msg, _outgoing)
          {
            TRANSMIT(&msg, support_id);
            _outgoing.del(msg);
          }
      }
}

//----------this part takes care of the messages read from the queue---------

int
sr::procIncMessages() {
      message * msg_p;
      while(msg_p = POP_MSG())
      {
          if(msg_p->getReceiver() == getId())
          {
            switch(msg_p->getType()) {
            case TYPE_NRM: {
                if(msg_p->getTarget() == getId())
                {
                  // check if this is a delivery receipt
                  message msg;
                  bool new_delivery = true;
                  forall(msg, _received)
                      if(msg == *msg_p)
                        new_delivery = false;

                  if(new_delivery) // a new delivery receipt
                  {
                      // add it to received msgs list
                      _received.insert(*msg_p);

                      // update statistics...
                      MSG_RCVD_SUCC(msg_p);

                      _messages_received++;
                  }

                  DEL_MSG(msg_p);
                }
                else
                {
                  // this should never happen in this protocol
                  cout << "Unexpected error: receiver != target in sr" <<
                              endl;
                  exit(1);
                }
```

```
                break;
              }
              default : {
                  // this should never happen in this protocol!!
                  cout << "Error in message processing in sr-host with id:" <<
                          getId() << endl;
                  exit(1);
              }
              }
            }
            else
              // this message is unnecessary (in this protocol)
              DEL_MSG(msg_p);
        }
}

#endif
```

Γ.2. Ο Κώδικας Υλοποίησης ενός Χρήστη Τύπου Snake

```
#ifndef SNAKE_H
#define SNAKE_H

#include <LEDA/map.h>

/*
 * define mobile hosts of type snake
 */
class snake : public mh
{
public:
    // derived constructor overrides base constructor
    snake::snake(environment * my_env, GRAPH<node, edge> * my_motion_graph,
int head, int tail, snake * next_mh, int _max_msg_life)
    : mh(my_env, my_motion_graph, (*my_motion_graph).first_node())
    {
      // initializing
      SET_SUPPORT();

      _head = head;
      _tail = tail;

      _next_mh = next_mh;

      _syn_mode = 0;

      _undelivered.clear();
      _acknowledged.clear();
      _acknowledge_min.clear();

      _pos_queue.clear();

      MAX_MSG_LIFE = _max_msg_life;
    }

    // structure used for the snake motion-holds the pos of the previous node
    int           POS_APPEND(node next_pos);

    //override protocol implementation function
    int executeProtocol();

    //override protocol implementation function
    int memoryUsage() {
            return _undelivered.size() + _acknowledged.size();
      }

protected:

    // Process incoming messages
    int procIncMessages();

    // Support Motion sub-protocol P1
    int subProtocolP1();

    // Sensor Subprotocol P2
    int subProtocolP2();
```

```
    // Synchronization sub-protocol P3
    int subProtocolP3(bool _send_front, bool _send_back);

    //
    int maintMinAck();
private:
    // functions used to determine whether this host is a HEAD or a TAIL
    bool IS_HEAD() { return ((_head == getId())? true : false ); }

    bool IS_TAIL() { return ((_tail == getId())? true : false ); }

    // info about the snake structure
    int                 _head;
    int                 _tail;

    // pointer to the next snake node
    snake               * _next_mh;

    // this queue holds the positioning history of the host
    queue< node >       _pos_queue;

    // synchronization mode
    int                         _syn_mode;

    // this is set to true whenever the _undelivered and _acknowledged sets
            are altered
    bool                _send_front;
    bool                _send_back;

    // this set is used to hold undelivered messages
    set< message >              _undelivered;

    // this set is used to hold acknowledged messages
    set< message >              _acknowledged;

    // this set is used to hold the ids of the neighboring hosts
    list< int >                 _neighbours;

    // this set is used to hold the ids of the neighboring support hosts
    list< int >                 _support_neighbours;

    int                         MAX_MSG_LIFE;

    map< int, int >             _acknowledge_min;

};


/*
 *function used to implement the snake motion
 */
int
snake::POS_APPEND(node next_pos)
{
    _pos_queue.append(next_pos);

    return 0;
}
```

```
/*
 * define 'executeProtocol' function for the derived class mh_snake
 */
int
snake::executeProtocol()
{
    clock_t prev_update = clock();

    if(IS_SUPPORT())
    {
      // Process incoming messages
      procIncMessages();

      // Synchronize Snake with synchronization subprotocol P3
      subProtocolP3(_send_front, _send_back);

      // Use sensor subprotocol P2 to send messages to designated Receiver
      subProtocolP2();

      // Move around using support subprotocol P1
      subProtocolP1();
    }

    return clock() - prev_update;
}


/*
 * define 'subProtocolP1' function
 * this part forces mh to make random walk on his motion graph if head, or
 * move as snake otherwise
 */
int
snake::subProtocolP1() {

    if(IS_HEAD())
    {
      MOVE_RANDOMLY();
      if(!IS_TAIL())
      {
          // inform your next support about your new location
          _next_mh->POS_APPEND(getMyPosition());
      }
      // else, i am all the snake, so just move around!
    }
    // in case of queue the size of transmition, start moving
    else
    {
      if(_pos_queue.size() >= (DEFAULT_TRANSMITION_RANGE))
      {
          node nxt_pos = _pos_queue.pop();
          MOVE_AT(nxt_pos);

          if(!IS_TAIL())
          {
            // inform your next support about your new location
            _next_mh->POS_APPEND(getMyPosition());
          }
      }     // else wait...
    }
}
```

```
//-----this part is used to send messages whenever a receiver is around------

int
snake::subProtocolP2() {
      // get a set with neighbours
      GET_NEIGHBOURS_IDS(_neighbours);

      int target_id;
      forall(target_id, _neighbours)
      {
          message msg;
          forall(msg, _undelivered)
          {
            int target = msg.getTarget();
            if(target_id == target)
            {
                TRANSMIT(&msg, target);

                _undelivered.del(msg);

                _acknowledged.insert(msg);

                maintMinAck();
            }
          }
      }
}

/*
 * define 'subProtocolP3' function
 * synchronize with other hosts of the snake the data
 */
int
snake::subProtocolP3(bool _send_front, bool _send_back) {

      if(_send_front || _send_back)
            maintMinAck();

      // in case we need to pass info to the front...
      if(_send_front && (!IS_HEAD()))
      {
          int front_id = getId()+1;
          message * msg_p;

          // send "undelivered" to the front
          msg_p = NEW_MSG(TYPE_SYN1, front_id);
          msg_p->assign(new set< message >(_undelivered));

          TRANSMIT(msg_p, front_id);

          DEL_MSG(msg_p);

          // send "acknowledged" to the front
          msg_p = NEW_MSG(TYPE_SYN2, front_id);
          msg_p->assign(new set< message >(_acknowledged));

          TRANSMIT(msg_p, front_id);

          DEL_MSG(msg_p);
      }
```

```
      if(_send_back && (!IS_TAIL()))
      {
          int back_id = getId()-1;
          message * msg_p;

          // send "undelivered" to the front
          msg_p = NEW_MSG(TYPE_SYN3, back_id);
          msg_p->assign(new set< message >(_undelivered));

          TRANSMIT(msg_p, back_id);

          DEL_MSG(msg_p);

          // send "acknowledged" to the front
          msg_p = NEW_MSG(TYPE_SYN4, back_id);
          msg_p->assign(new set< message >(_acknowledged));

          TRANSMIT(msg_p, back_id);

          DEL_MSG(msg_p);
      }
}


//----------this part takes care of the messages read from the queue---------

int
snake::procIncMessages() {
      _send_front = false;
      _send_back  = false;

      message * msg_p;
      while(msg_p = POP_MSG())
      {
          if(msg_p->getReceiver() == getId())
          {
            switch(msg_p->getType()) {
            case TYPE_NRM: {
                // TYPE_NRM is used to receive a message from the originating
                // sender.
                int new_message_arrived;

                new_message_arrived = _undelivered.size();

                _undelivered.insert(*msg_p);

                DEL_MSG(msg_p);

                maintMinAck();

                if (_undelivered.size() != new_message_arrived) {
                  // Indeed, this is a new message
                      _send_front = true;
                      _send_back  = true;
                }

                break;
            }
            // messages received by the back side of the snake
            case TYPE_SYN1: {
                int new_message_arrived;
```

```
            new_message_arrived = _undelivered.size();

            set< message > * s = (set< message > *)msg_p->getInfo();

            // merge sets...
            _undelivered += *s;

            delete s;           // release memory
            DEL_MSG(msg_p);

            maintMinAck();

            if (_undelivered.size() != new_message_arrived) {
              // Indeed, this is a new message
              // inform system about reqired list synchronization
              _send_front = true;
            }

            break;
        }
        // acknowledges received by the back side of the snake
        case TYPE_SYN2: {
            int new_ack_arrived;

            new_ack_arrived = _acknowledged.size();

            set< message > * s = (set< message > *)msg_p->getInfo();

            // merge sets...
            _acknowledged += *s;

            delete s;           // release memory
            DEL_MSG(msg_p);

            maintMinAck();

            if (_acknowledged.size() != new_ack_arrived) {
              // Indeed, this is a new message
              // inform system about reqired list synchronization
              _send_front = true;
            }

            break;
        }
        // messages received by the front side of the snake
        case TYPE_SYN3: {
            int new_message_arrived;

            new_message_arrived = _undelivered.size();

            set< message > * s = (set< message > *)msg_p->getInfo();

            // merge sets...
            _undelivered += *s;

            delete s;           // release memory
            DEL_MSG(msg_p);

            maintMinAck();
```

```
                if (_undelivered.size() != new_message_arrived) {
                  // Indeed, this is a new message
                  // inform system about reqired list synchronization
                  _send_back = true;
                }

                break;
            }
            // acknowledged received by the front side of the snake
            case TYPE_SYN4: {
                int new_ack_arrived;

                new_ack_arrived = _acknowledged.size();

                set< message > * s = (set< message > *)msg_p->getInfo();

                // merge sets...
                _acknowledged += *s;

                delete s;           // release memory
                DEL_MSG(msg_p);

                maintMinAck();

                if (_acknowledged.size() != new_ack_arrived) {
                  // Indeed, this is a new message
                  // inform system about reqired list synchronization
                  _send_back = true;
                }

                break;
            }
            default: {
                // this should never happen in this protocol...
                cout << "Error in runner[" << getId() << "] host!" << endl;
                exit(1);
            }
            }
          }
          else
            // this message is unnecessary (in this protocol)
            DEL_MSG(msg_p);
      }

}

int
snake::maintMinAck() {

      message msg;

      // we synchronize them
      _undelivered -= _acknowledged;

      // Find out maximum msgID of delivered messages
      // before which it is safe to delete acks
      forall(msg, _acknowledged) {
            if (_acknowledge_min.defined(msg.getSource()) == false)
                  _acknowledge_min[msg.getSource()] = 0;

            if (_acknowledge_min[msg.getSource()] == msg.getId()-1)
```

```
                    _acknowledge_min[msg.getSource()]++;
        }

        forall(msg, _undelivered)
              if (_acknowledge_min[msg.getSource()] > msg.getId()) {
                  _undelivered.del(msg);
              }

        forall(msg, _acknowledged)
              if (_acknowledge_min[msg.getSource()] >= msg.getId()) {
                  _acknowledged.del(msg);
              }

        // at this point if sets were altered, we synchronize them again
        _undelivered -= _acknowledged;
}

#endif
```

Γ.3. Ο Κώδικας Υλοποίησης ενός Χρήστη Τύπου Runner

```
#ifndef RUNNER_H
#define RUNNER_H

#include <LEDA/map.h>

/*
 * define mobile hosts of type runner
 */
class runner : public mh
{
public:
    // derived constructor overrides base constructor
    runner::runner(environment * my_env, GRAPH<node, edge> * my_motion_graph,
int _max_msg_life)
    : mh(my_env, my_motion_graph)
    {
      // initialization
      SET_SUPPORT();

      _syn_mode = 0;

      _undelivered.clear();
      _acknowledged.clear();
      _acknowledge_min.clear();

      MAX_MSG_LIFE = _max_msg_life;
    }

    //override protocol implementation function
    int executeProtocol();

    //function used for statistics
    int memoryUsage() { return _undelivered.size() + _acknowledged.size(); }

private:

    // Process incoming messages
    int procIncMessages();

    // Sensor Subprotocol P2
    int subProtocolP2();

    // Synchronization Subprotocol P2
    int subProtocolP3();

    // synchronization mode
    int                       _syn_mode;

    //
    int maintMinAck();

    // variable used to flag any alter to the sets
    bool sets_touched;

    // this set is used to hold undelivered messages
    set< message >            _undelivered;
```

```
    // this set is used to hold acknowledged messages
    set< message >            _acknowledged;

    // this set is used to hold the ids of the neighboring hosts
    list< int >               _neighbours;

    // this set is used to hold the ids of the neighboring support hosts
    list< int >               _support_neighbours;

    int                       MAX_MSG_LIFE;

    map< int, int >           _acknowledge_min;

};


/*
 * define 'executeProtocol' function for the derived class mh_runner
 */
int
runner::executeProtocol()
{
    clock_t prev_update = clock();

    if(IS_SUPPORT())
    {
      // Process incoming messages
      procIncMessages();

      // Support sensor sub-protocol P2 is used to send messages
      // whenever a designated Receiver is within transmission range
      subProtocolP2();

      // Support synchronization sub-protocol P3 is used to synchronize
      // sets _outgoing, _acknowledged with other members of the Support
      subProtocolP3();
    }


//-------- this part forces mh to make random walk on his motion graph ------

    // make a random move on my motion graph
    MOVE_RANDOMLY();

    return (clock() - prev_update);
}

//-----this part is used to send messages whenever a receiver is around------

int
runner::subProtocolP2() {
      // get a set with neighbours
      GET_NEIGHBOURS_IDS(_neighbours);

      int target_id;
      forall(target_id, _neighbours)
      {
          message msg;
          forall(msg, _undelivered)
          {
            int target = msg.getTarget();
```

```
            if(target_id == target)
            {
                TRANSMIT(&msg, target);

                _undelivered.del(msg);

                _acknowledged.insert(msg);

                maintMinAck();
            }
          }
      }
}


//-----------------synchronize with other hosts data-------------------------

int
runner::subProtocolP3() {
      // get a set with support neighbours
      GET_SUPPORT_NEIGHBOURS_IDS(_support_neighbours);

      int sn_id;
      int smaller_id = INT_MAX;
      forall(sn_id, _support_neighbours)
          if(sn_id < smaller_id)
            smaller_id = sn_id;

      switch (_syn_mode) {
      case 0: {
          // in the case that I have the smaller id...
          if(smaller_id == getId())
          {
            // just wait to receive synchronization packets from the others
            _syn_mode = 2;
          }
          // ...else in the case that there is one with a greater id...
          else
          {
            message * msg_p;
            // send "undelivered" to the one with the smaller id
            msg_p = NEW_MSG(TYPE_SYN1, smaller_id);
            msg_p->assign(new set< message >(_undelivered));

            TRANSMIT(msg_p, smaller_id);

            DEL_MSG(msg_p);

          }

          break;
      }
      case 1: {
          // simply change to mode 0 again
          _syn_mode = 0;
          break;
      }
      case 2: {
          // i was the one with the smaller id during the previous round so i
          // need to transmit my data to every host in the neighbourhood
```

```
            // remove myself from the neighborhood list
            _support_neighbours.remove(getId());

            while(!_support_neighbours.empty())
            {
              int support_id = _support_neighbours.front();

              message * msg_p;

              // send "undelivered" to the every support in the neighborhood
              msg_p = NEW_MSG(TYPE_SYN1, support_id);
              msg_p->assign(new set< message >(_undelivered));

              TRANSMIT(msg_p, support_id);

              DEL_MSG(msg_p);

              // send "acknowledged" to the every support in the neighborhood
              msg_p = NEW_MSG(TYPE_SYN2, support_id);
              msg_p->assign(new set< message >(_acknowledged));

              TRANSMIT(msg_p, support_id);

              DEL_MSG(msg_p);

              _support_neighbours.remove(support_id);
            }
            // and then change back to mode 0
            _syn_mode = 0;

            break;
        }
        default: {
            // this should never happen in this protocol...
            cout << "Error while synchronizing: Bad syn-mode!" << endl;
            exit(1);
        }
        }
}

//----------this part takes care of the messages read from the queue---------

int
runner::procIncMessages() {
        sets_touched = false;

        message * msg_p;
        while(msg_p = POP_MSG())
        {
            if(msg_p->getReceiver() == getId())
            {
              switch(msg_p->getType()) {
              case TYPE_NRM: {
                  // TYPE_NRM is used to receive a message from the originating
                  // sender.
                  int new_message_arrived;

                  new_message_arrived = _undelivered.size();

                  _undelivered.insert(*msg_p);
```

```
        DEL_MSG(msg_p);

        maintMinAck();

        if (_undelivered.size() != new_message_arrived) {
          // Indeed, this is a new message
          sets_touched = true;
        }

        break;
      }
      case TYPE_SYN1: {
        int new_message_arrived;

        new_message_arrived = _undelivered.size();

        set< message > * s = (set< message > *)msg_p->getInfo();

        // merge sets...
        _undelivered += *s;

        delete s;           // release memory
        DEL_MSG(msg_p);

        maintMinAck();

        if (_undelivered.size() != new_message_arrived) {
          // Indeed, this is a new message
          // inform system about reqired list synchronization
          sets_touched = true;
        }

        break;
      }
      // acknowledges received by the back side of the snake
      case TYPE_SYN2: {
        int new_ack_arrived;

        new_ack_arrived = _acknowledged.size();

        set< message > * s = (set< message > *)msg_p->getInfo();

        // merge sets...
        _acknowledged += *s;

        delete s;           // release memory
        DEL_MSG(msg_p);

        maintMinAck();

        if (_acknowledged.size() != new_ack_arrived) {
          // Indeed, this is a new message
          // inform system about reqired list synchronization
          sets_touched = true;
        }

        break;
      }
      default: {
        // this should never happen in this protocol...
        cout << "Error in runner[" << getId() << "] host!" << endl;
```

```
                exit(1);
              }
              }
            }
            else
              // this message is unnecessary (in this protocol)
              DEL_MSG(msg_p);
      }

}


int
runner::maintMinAck() {

      message msg;

      // we synchronize them
      _undelivered -= _acknowledged;

      // Find out maximum msgID of delivered messages
      // before which it is safe to delete acks
      forall(msg, _acknowledged) {
            if (_acknowledge_min.defined(msg.getSource()) == false)
                  _acknowledge_min[msg.getSource()] = 0;

            if (_acknowledge_min[msg.getSource()] == msg.getId()-1)
                  _acknowledge_min[msg.getSource()]++;
      }

      forall(msg, _undelivered)
            if (_acknowledge_min[msg.getSource()] > msg.getId()) {
                _undelivered.del(msg);
            }

      forall(msg, _acknowledged)
            if (_acknowledge_min[msg.getSource()] >= msg.getId()) {
                _acknowledged.del(msg);
            }

      // at this point if sets were altered, we synchronize them again
      _undelivered -= _acknowledged;
}

#endif
```