Δημιουργία Κινήτρων για Συνεργασία μεταξύ των Κόμβων σε Ασύρματα Δίκτυα Αυθαίρετης Δομής

Χαρίκλεια Αθανασοπούλου

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πανεπιστήμιο Κρήτης
Σχολή Θετικών και Τεχνολογικών Επιστημών
Τμήμα Επιστήμης Υπολογιστών

Ηράκλειο, Μάρτιος 2005

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Δημιουργία Κινήτρων για Συνεργασία μεταξύ των Κόμβων σε Ασύρματα Δίκτυα Αυθαίρετης Δομής

Εργασία που υποβλήθηκε από την
Χαρίκλεια Αθανασοπούλου
ως μερική εκπλήρωση των απαιτήσεων για την απόκτηση
Μεταπτυχιακού Διπλώματος Ειδίκευσης
στην Επιστήμη των Υπολογιστών

ΣΥΓΓΡΑΦΕΑΣ:

Χαρίκλεια Αθανασοπούλου

ΕΙΣΗΓΗΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Επόπτης:
Βασίλειος Σύρης

Μέλος:
Απόστολος Τραγανίτης

Μέλος:
Παναγιώτης Τσακαλίδης

ΔΕΚΤΗ:

Δημήτρης Πλεξουσάκης,
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Μάρτιος 2005

Στους γονείς μου & το Βαγγέλη

Δημιουργία κινήτρων για συνεργασία μεταξύ των κόμβων σε ασύρματα δίκτυα αυθαίρετης δομής

Χαρίκλεια Αθανασοπούλου

Μεταπτυχιακή Εργασία

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Περίληψη

Στα ασύρματα δίκτυα αυθαίρετης δομής οι κόμβοι που δεν ανήκουν στο ίδιο πεδίο εμβέλειας επικοινωνούν μεταξύ τους χρησιμοποιώντας άλλους κόμβους του δικτύου ως ενδιάμεσους σταθμούς αναμετάδοσης. Η κίνηση δρομολογείται και μεταφέρεται από τον ένα κόμβο στον επόμενο ενδιάμεσο, έτσι ώστε να καταλήξει στον τελικό αποδέκτη, ο οποίος ενδέχεται να είναι αρκετά απομακρυσμένος από την πηγή. Στην περίπτωση αυτή κάθε κόμβος πρέπει να συμπεριφέρεται όχι μόνο ως πηγή ή προορισμός, αλλά παράλληλα και ως αναμεταδότης.

Εξαιτίας της περιορισμένης ζωής της μπαταρίας των κόμβων στα ασύρματα δίκτυα, είναι πολύ πιθανό αυτοί να μην επιθυμούν την αποδοχή όλων των εισερχόμενων αιτήσεων προώθησης κίνησης για λογαριασμό άλλων κόμβων. Αν, όμως, όλοι οι κόμβοι συμπεριφέρονται βάσει αυτού του εγωιστικού τρόπου, θα επέλθει δραματική πτώση του συνολικού όγκου δεδομένων που διέρχεται από το δίκτυο (data throughput).

Ο σκοπός της παρούσας εργασίας είναι να προτείνει και να αξιολογήσει δύο αλγορίθμους παροχής κινήτρων σε κόμβους ασύρματων δικτύων αυθαίρετης δομής για προώθηση πακέτων από άλλους κόμβους, ώστε να αποφευχθεί το σενάριο μη συνεργασίας που περιγράφηκε παραπάνω.

Ο πρώτος αλγόριθμος βασίζεται σε πρόταση άλλης εργασίας, κατά την οποία κάθε κόμβος είναι έτοιμος να βοηθήσει, δηλαδή να

αποδεχτεί μία αίτηση για προώθηση κίνησης, μόνο στην περίπτωση που ο ίδιος έχει βοηθηθεί κατά ανάλογο ποσοστό στο παρελθόν από το υπόλοιπο δίκτυο συνολικά. Παρατηρείται, όμως, πως η εμφάνιση εγωιστικών συμπεριφορών επιδρούν ιδιαίτερα αρνητικά στη διαπερατότητα (throughput) του συνολικού δικτύου κατά τη χρήση αυτού του αλγορίθμου.

Στην παρούσα εργασία προτείνουμε μία παραλλαγή της παραπάνω ιδέας, η οποία αποσκοπεί στην αντιμετώπιση των προβλημάτων που δημιουργούνται από εγωιστικές συμπεριφορές. Συγκεκριμένα, η συμπεριφορά του κάθε γείτονα ενός ενδιάμεσου κόμβου καταγράφεται ξεχωριστά. Ο νέος αλγόριθμος υπαγορεύει πως κάθε κόμβος είναι έτοιμος να βοηθήσει, μόνο στην περίπτωση που ο ίδιος έχει βοηθηθεί κατά ανάλογο ποσοστό στο παρελθόν από τον κόμβο που κάνει τώρα την αίτηση.

Ο δεύτερος αλγόριθμος που προτείνεται διατηρεί τη διαφορά μεταξύ των ποσών της δοθείσας και της ληφθείσας βοήθειας κάθε κόμβου εντός συγκεκριμένων ορίων, τα οποία καθορίζονται βάσει ενός δοχείου κουπονιών (bucket).

Και οι δύο αλγόριθμοι μπορούν να επεκταθούν με τη χρήση διαφορετικών βαρών για την εξισορρόπηση των ποσών ληφθείσας και δοθείσας βοήθειας, η οποία συνιστάται κατά την ύπαρξη ενός ή και περισσοτέρων κεντρικών σημείων πρόσβασης (access points), στα οποία κατευθύνεται το μεγαλύτερο ποσοστό της κίνησης των κόμβων.

Τα βασικότερα πειραματικά αποτελέσματα έδειξαν βελτίωση του συνολικού throughput και περιορισμό της κακής επίδρασης των εγωιστών κόμβων μόνο στους γείτονες αυτών στην περίπτωση του πρώτου αλγορίθμου, ενώ η χρήση του δεύτερου αλγορίθμου υπερέχει σε σενάρια εκρηκτικής κίνησης (bursts).

Επόπτης: Βασίλειος Σύρης, Καθηγητής

Creating motives for packet forwarding using relay nodes in wireless ad hoc networks

Chariklia Athanasopoulou
Master of Science Thesis

UNIVERCITY OF CRETE
COMPUTER SCIENCE DEPRTMENT

Abstract

In wireless ad hoc networks nodes that do not belong in the same transmission range communicate with each other using intermediate nodes as relays. Traffic is routed from each node to the next relay until it reaches the final destination, which may be located far away from the source. In this case, every node must behave not only as a source or a destination, but as a router as well.

Since wireless networks have a limited battery life, it is possible for a node not to wish to accept all the incoming relay requests. But, if all nodes behave in this selfish way, there will be a dramatic drop of the network data throughput.

The purpose of this master thesis is to suggest and evaluate two algorithms that create motives between the nodes of wireless ad hoc networks, in order to avoid the bad scenario of misbehaviour just described above.

The first algorithm presented is based in some previous related work, which dictates that each node is ready to help, i.e. to accept a relay request, only if it has been already helped analogously by the whole network. However, the presence of selfish nodes affects badly the entire network data throughput.

We suggest an extension of the idea described above that aims to deal better with selfishness. The past behaviour of each neighbour of each relay node is recorded separately. The new algorithm dictates that each node is ready to help another one, only if it has been helped analogously by this specific node during the past.

The second algorithm we present is based on a "leaky bucket" approach. It maintains the difference of the given and received help amounts of each node within specific limits, which are determined by the size of a bucket.

Both algorithms can be expanded to a version that uses different weights for the equalization of given and received help amounts. This version is recommended when most of the network traffic is destined to one or more access points.

The basic experiment results showed that the first algorithm manages to reduce the geographic boundaries of the bad effects of selfish behaviors and improve the total throughput, while the second algorithm deals better with burst traffic scenarios.

Supervisor: Vasilios Siris, Professor

ΕΥΧΑΡΙΣΤΙΕΣ

Ολοκληρώνοντας την παρούσα μεταπτυχιακή εργασία νοιώθω έντονη την ανάγκη να ευχαριστήσω το Τμήμα Επιστήμης Υπολογιστών καθώς επίσης και το Ινστιτούτο Πληροφορικής του Ι.Τ.Ε., τόσο για την υλικοτεχνική υποδομή που μου προσέφεραν οποιαδήποτε στιγμή τη χρειάστηκα, όσο και για τη σημαντική οικονομική στήριξη που μου παρείχαν καθ' όλη τη διάρκεια των σπουδών μου.

Ένα ιδιαίτερα σημαντικό ευχαριστώ οφείλω στον καθηγητή και επόπτη της εργασίας μου κ. Βασίλειο Σύρη, ο οποίος μου έδωσε την ευκαιρία να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα και παράλληλα με καθοδήγησε με υπομονή και πολλές χρήσιμες συμβουλές και παρατηρήσεις καθ' όλη τη διάρκεια της διετούς συνεργασίας μας.

Ιδιαίτερα σημαντική υπήρξε για μένα και η βοήθεια του κ. Μανόλη Κατεβαίνη κατά τη διάρκεια τόσο των προπτυχιακών όσο και των μεταπτυχιακών μου σπουδών. Οι γνώσεις που απέκτησα κοντά του όσο καιρό ασχολήθηκα σθεναρά με το hardware σίγουρα μου έχουν φανεί ήδη πολύ χρήσιμες, κυρίως, όμως, τον ευχαριστώ για τις «πατρικές» συμβουλές του και τη μεθοδικότητα που μου δίδαξε ως βασική αρχή για κάθε εργασία.

Ευχαριστώ το σύνολο των καθηγητών μου για τις γνώσεις που μου εμφύσησαν, το ενδιαφέρον που μου δημιούργησαν για τις νέες τεχνολογίες από το πρώτο κιόλας έτος των σπουδών μου. Ιδιαίτερα ευχαριστώ τους κ. Τραγανίτη και κ. Σταμούλη που με τον τρόπο διδασκαλίας τους ώθησαν τα ερευνητικά μου ενδιαφέροντα στους τομείς του hardware και των δικτύων αντίστοιχα.

Σίγουρα, οφείλω να ευχαριστήσω και όλους μου τους συνεργάτες από το εργαστήριο των Δικτύων του Ι.Τ.Ε., οι οποίοι υπήρξαν πάντα πρόθυμοι να με βοηθήσουν σε οποιεσδήποτε δυσκολίες που παρουσιάζονταν.

Τέλος, ευχαριστώ τα πιο κοντινά και αγαπημένα μου πρόσωπα: το Βαγγέλη για τη βοήθεια που μου προσφέρει όποια στιγμή και αν τη ζητήσω τόσο σε προσωπικό όσο και σε επίπεδο εργασίας, και τους γονείς μου που με στηρίζουν και προσπαθούν να μου προσφέρουν το καλύτερο δυνατό όλα αυτά τα χρόνια.

Α' ΜΕΡΟΣ : ΕΙΣΑΓΩΓΗ

1. Χαρακτηριστικά ασύρματων δικτύων αυθαίρετης δομής

Τον τελευταίο καιρό παρατηρείται μία ερευνητική έξαρση στον τομέα των ασύρματων δικτύων αυθαίρετης δομής. Αυτού του είδους τα δίκτυα μοιάζουν να είναι αρκετά ελκυστικά, ιδιαίτερα από τη στιγμή που δύνανται να παρέχουν μία σημαντικού επιπέδου συνδεσιμότητα χωρίς την ανάγκη για σταθερή και αμετάβλητη τεχνική υποδομή. Μέχρι στιγμής πολλές εφαρμογές έχουν προταθεί, ειδικά σε καταστάσεις έκτακτης ανάγκης, όπως για παράδειγμα σε επιχειρήσεις διάσωσης. Φυσικά, οι ενεργειακοί περιορισμοί που τίθενται από τους κινητούς κόμβους πρέπει να λαμβάνονται υπό σοβαρή θεώρηση.

Ένα κινητό δίκτυο αυθαίρετης δομής είναι ένα ασύρματο δίκτυο το οποίο αποτελείται από ένα σύνολο ενεργειακά περιορισμένων κόμβων, όπου η κίνηση μεταφέρεται αναγκαστικά σε μοτίβο πολλαπλών μικρών βηματισμών από κόμβο σε κόμβο (multi-hop). Αυτοί οι κινητοί κόμβοι καλούνται να διαχειριστούν μόνοι τους την αυτό-διοργάνωσή τους με εσωτερική συνεργασία, αφού το δίκτυο δε συνοδεύεται από καμία προ-καθιερωμένη και εγκατεστημένη τεχνική υποδομή. Συνεπώς, κάθε κόμβος καλείται να εκτελέσει από μόνος του όλες τις απαραίτητες εργασίες που του υπαγορεύει το δίκτυο χωρίς την ύπαρξη κάποιου τύπου εξωτερικής διαιτησίας. Ως αποτέλεσμα, οι κόμβοι αναγκάζονται να επικοινωνούν με απομακρυσμένους προορισμούς αιτώντας από άλλους κόμβους να διαδραματίσουν το ρόλο μεταφορέων «ξένων» ροών κίνησης.

Είναι προφανές πως ο περιορισμένος χρόνος ζωής της μπαταρίας των ασύρματων κόμβων αποτελεί μεγάλο μειονέκτημα. Η ενέργεια είναι ένας πολύτιμος πόρος για κάθε κόμβο και πρέπει να διαχειρίζεται με ιδιαίτερη προσοχή, έτσι ώστε να αποφεύγονται ανεπιθύμητες καταστάσεις, όπως για παράδειγμα ένας πρόωρος τερματισμός της δραστηριότητάς τους στο δίκτυο. Με μια πρώτη παρατήρηση, εύκολα παρατηρεί κανείς για ποιο λόγο ένας κόμβος καταλήγει στην απόφαση για υιοθέτηση εγωιστικής συμπεριφοράς:

απώτερος σκοπός είναι πάντα η εξοικονόμηση ενέργειας για προσωπικό όφελος, η οποία φαίνεται να επιτυγχάνεται με την απόρριψη αιτήσεων προώθησης κίνησης για λογαριασμό άλλων κόμβων. Εντούτοις, αν ξαφνικά όλοι οι κόμβοι ενός συγκεκριμένου δικτύου υιοθετήσουν μια τέτοια συμπεριφορά, τα δυσάρεστα αποτελέσματα θα είναι άμεσα εμφανή και θα σχετίζονται με δραματική μείωση της συνολικής κίνησης που διέρχεται από το δίκτυο (data throughput). Ο μοναδικός τρόπος επίλυσης αυτού του άσχημου φαινομένου είναι η ανάπτυξη μεθόδων, οι οποίες θα φροντίζουν για τη δημιουργία κινήτρων συνύπαρξης σε περιβάλλον συνεργασίας μεταξύ των κόμβων. Αν κάθε κόμβος διαθέτει κίνητρο για αποδοχή αιτήσεων προώθησης κίνησης, τότε η συνδεσιμότητα του δικτύου αυξάνεται σημαντικά, ενώ παράλληλα αποφεύγονται κακόβουλες συμπεριφορές.

2. Σχετικές Εργασίες

Πολλοί ερευνητές έχουν ήδη ασχοληθεί με το πρόβλημα που παρουσιάστηκε και έχουν προτείνει κάποιες στρατηγικές που ωθούν την άνθηση της συνεργασίας μεταξύ των κόμβων που αποτελούν ένα δίκτυο αυθαίρετης δομής [1], [2], [3]. Στις παραγράφους που ακολουθούν παρουσιάζονται συνοπτικά οι κυριότερες από αυτές τις τεχνικές.

2.1. Ο αλγόριθμος Generous TIT-FOR-TAT (GTFT)

Πιο συγκεκριμένα, στη μελέτη [1] οι ερευνητές ξεκίνησαν την εργασία τους καθορίζοντας το ιδανικό – βέλτιστο throughput που κάθε κόμβος θα έπρεπε να λαμβάνει. Ο καθορισμός αυτός έγινε με βάση κάποιους συγκεκριμένους περιορισμούς χρόνου ζωής και την υπόθεση ότι οι κόμβοι συμπεριφέρονται ακολουθώντας ένα ορθολογικό μοτίβο. Αμέσως μετά τον προσδιορισμό αυτής της τιμής ως το βέλτιστο σημείο λειτουργίας που ακολουθεί την Pareto κατανομή, προχώρησαν στην παρουσίαση ενός κατανεμημένου και επεκτάσιμου αλγορίθμου αποδοχής, στον οποίο έδωσαν την ονομασία Generous TIT-FOR-TAT (GTFT). Τελικά, έδειξαν πως ο αλγόριθμος αυτός καταλήγει σε Nash equilibrium και απέδειξαν

ότι το σύστημα συγκλίνει στο ορθολογικό και βέλτιστο σημείο λειτουργίας.

Προκειμένου να λειτουργήσει σωστά αυτός ο αλγόριθμος πρέπει να τηρούνται τρία βασικά χαρακτηριστικά:

α) Η πολιτική που θα ακολουθηθεί δεν μπορεί να είναι τυχαίας και στατικής φύσης. Αντίθετα, απαιτείται η ανάπτυξη στρατηγικών, οι οποίες βασίζονται σε γνώση συμπεριφορών. Συγκεκριμένα, κάθε κόμβος βασίζει την απόφασή του στην παρελθούσα συμπεριφορά των κόμβων του συστήματος.
β) Πρέπει να υπάρχουν στρατηγικές προστασίας από τυχούσα εξωτερική εκμετάλλευση (protection from exploitation).
γ) Η στρατηγική που θα ακολουθηθεί πρέπει να είναι επεκτάσιμη (scalable).

Οι συγγραφείς της εργασίας [1] ξεκίνησαν κατατάσσοντας το πρόβλημα της συνεργασιμότητας των κόμβων ενός δικτύου αυθαίρετης δομής στην περιοχή των προβλημάτων που είναι γνωστή ως "Non – Cooperative Game Theory" και σύγκριναν το δίλημμα του κάθε κόμβου για αποδοχή ή απόρριψη μιας αίτησης για συνεργασία με το δίλημμα του φυλακισμένου.

Στο GTFT κάθε παίκτης μιμείται τη δράση του άλλου παίκτη στο προηγούμενο παιχνίδι. Παρ' όλα αυτά, υπάρχουν φορές που ο παίκτης οφείλει να δείξει λίγη παραπάνω γενναιοδωρία. Κάθε κόμβος διατηρεί ένα record με την παρελθούσα εμπειρία του για τη συνολική συμπεριφορά του δικτύου απέναντί του χρησιμοποιώντας δύο μεταβλητές και οι αποφάσεις παίρνονται μόνο βάσει αυτών των τιμών.

2.2. Οι αλγόριθμοι RANDOM και PAY-IT-FORWARD

Στην εργασία [2] οι συγγραφείς μελέτησαν την αντιστάθμιση συντελεστών (trade-off) που υφίσταται μεταξύ της κατανάλωσης ενέργειας και της πιθανότητας «μπλοκαρίσματος» μιας συνεδρίας κίνησης (session), όπως επίσης και την ικανότητα εγγύησης από πλευράς δικτύου για χαμηλή κατανάλωση ενέργειας σε χρήστες που είτε επιθυμούν είτε είναι απαραίτητο να συμπεριφέρονται

εγωιστικά. Συγκεκριμένα, προσδιόρισαν μία παράμετρο που ονόμασαν «συμπάθεια» (sympathy), η οποία αντικατοπτρίζει το επίπεδο εγωισμού / αλτρουισμού κάθε κόμβου και στη συνέχεια, προχώρησαν στην πρόταση δύο διαφορετικών στρατηγικών. Πρόκειται για τις στρατηγικές RANDOM και PAY-IT-FORWARD.

Με βάση τη RANDOM πολιτική, όταν ένας κόμβος k μίας ήδη γνωστής διαδρομή r λαμβάνει μία αίτηση για προώθηση από έναν άλλο κόμβο s, αυτός αποδέχεται την αίτηση με τυχαία πιθανότητα sympathy(k,r). Κατά συνέπεια, η απόφαση για αποδοχή ή απόρριψη μίας αίτησης λαμβάνεται αυθαίρετα με τη διαδικασία «στριψίματος ενός νομίσματος».

Αντίθετα, η πολιτική PAY-IT-FORWARD είναι εμπνευσμένη από τον αλγόριθμο TIT-FOR-TAT, έχει, όμως, τους εξής περιορισμούς:

α) Πρόκειται για ένα παιχνίδι με πολλούς παίκτες (multiplayer game).
β) Οι κόμβοι δεν έχουν μνήμη σχετική με το αν έχουν βοηθηθεί προηγουμένως από κάποιο άλλο κόμβο – παίκτη του δικτύου.
γ) Κάθε κόμβος σχετίζεται με δύο παραμέτρους: credit και debit. Το credit είναι το ποσό βοήθειας που ένας κόμβος έχει λάβει από τους υπόλοιπους κόμβους συνολικά, ενώ το debit είναι το ποσό βοήθειας που ο ίδιος κόμβος έχει δώσει στους υπόλοιπους, εξυπηρετώντας αιτήσεις τους.

Ο αλγόριθμος PAY-IT-FORWARD προσπαθεί να εξισορροπήσει τις τιμές των credit και debit σε κάθε κόμβο. Στην περίπτωση αυτή, η απόφαση κάθε κόμβου βασίζεται μόνο στις τιμές των credit και debit, στο ρυθμό παραγωγής και αποστολής πακέτων από την πηγή, στο μέγεθος των δεδομένων που αποστέλλονται και στο ποσό ενέργειας που απαιτείται προκειμένου να γίνει η μεταφορά ενός πακέτου από ένα κόμβο στον ακριβώς επόμενο.

Εκτελώντας πειράματα και με τους δύο αλγορίθμους, τα αποτελέσματα έδειξαν ότι οι εγωιστές κόμβοι καταφέρνουν και επιβιώνουν καλά όταν ο συνολικός αριθμός τους είναι μικρός. Αντιθέτως, καταλήγουν σε φτωχά νούμερα απόδοσης καθώς η πυκνότητά τους αυξάνεται.

2.3. Τεχνικές διάδοσης φήμης

Κάποιες άλλες εργασίες παρουσιάζουν λύσεις, οι οποίες στηρίζονται σε παραλλαγές της συνάρτησης προώθησης κίνησης ήδη υλοποιημένων αλγορίθμων δρομολόγησης κάνοντας χρήση τεχνικών διάδοσης φήμης στο δίκτυο για την καλή ή άσχημη συμπεριφορά κάθε κόμβου (π.χ. εργασία [3]). Στις περισσότερες περιπτώσεις ο υπολογισμός της «φήμης» για κάθε κόμβο υπολογίζεται κάνοντας χρήση όχι μόνο τοπικής πληροφορίας, αλλά και πληροφορίας που παρέχουν οι υπόλοιποι κόμβοι. Οι μηχανισμοί διάδοσης φήμης είναι συνήθως αρκετά πολύπλοκοι τόσο στην κατασκευή όσο και στη συντήρησή τους.

Υπάρχει ένας κεντρικός μηχανισμός παρακολούθησης (watchdog mechanism), ενώ κάθε κόμβος διαθέτει πίνακες, όπου διατηρούνται οι τιμές της φήμης για τους υπόλοιπους κόμβους (reputation tables). Κάθε γραμμή ενός τέτοιου πίνακα περιέχει πληροφορία για ένα συγκεκριμένο κόμβο. Το πρωτόκολλο υπαγορεύει την ύπαρξη δύο τύπων κόμβων: ενός αιτούντος (requestor) και ενός ή περισσοτέρων παροχών (provider). Δηλαδή, ο κόμβος που κάνει την αίτηση για συνεργασία είναι δυνατό να απευθύνεται σε περισσότερους από έναν πιθανούς αποδέκτες. Αν ένας κόμβος αρνηθεί τη συνεργασία απορρίπτοντας την αίτηση, τότε ο κεντρικός μηχανισμός του αλγορίθμου αντιδρά μειώνοντας την τιμή της φήμης του στο αντίστοιχο reputation table. Με τον τρόπο αυτό, αν ο συγκεκριμένος κόμβος εμμένει στο μέλλον σε μη συνεργατική συμπεριφορά καταλήγει σε αποκλεισμό.

Όπως μπορεί εύκολα να παρατηρήσει κανείς, στην περίπτωση αυτή δεν είναι εξαρχής γνωστό το δρομολόγιο που πρέπει να ακολουθηθεί από την πηγή προς τον προορισμό. Οι τεχνικές διάδοσης φήμης υπαγορεύουν αλλαγές στην υλοποίηση των πρωτοκόλλων δρομολόγησης, άρα τελικά καθορίζουν τη διαδικασία επιλογής δρομολογίου. Αντίθετα, στις προηγούμενες εργασίες που αναφέρθηκαν, το δρομολόγιο θεωρείται γνωστό εξαρχής και οι διαδικασίες για τη δημιουργία κινήτρων εντάσσονται στους

εκάστοτε αλγόριθμους δρομολόγησης χωρίς να επεμβαίνουν στη λειτουργία των πρωτοκόλλων δρομολόγησης.

2.4. Στρατηγικές κοστολόγησης

Οι συγγραφείς της εργασίας [4] παρουσιάζουν ένα μικρό-ηλεκτρονικό πλαίσιο βασισμένο στη θεωρία του ανταγωνισμού (game theory) και προτείνουν έναν αλγόριθμο οικονομικής φύσης, ο οποίος ενθαρρύνει την προώθηση «ξένων» ροών κίνησης μεταξύ κόμβων αποζημιώνοντας χρηματικά τους κόμβους με ρόλο ενδιάμεσου μεταφορέα.

2.5. Τεχνικές CASHnet και Nuglet

Στην εργασία [5] προτείνεται η τεχνική CASHnet. Ο μηχανισμός χρέωσης και αποζημίωσης αυτής της τεχνικής λειτουργεί ως εξής: Κάθε φορά που ένας κόμβος επιθυμεί να αποστείλει ένα πακέτο που ο ίδιος έχει δημιουργήσει, πρέπει να πληρώσει με "Traffic Credits". Το ποσό πληρωμής σχετίζεται με την απόσταση, δηλαδή τα βήματα (hops) που μεσολαβούν μέχρι την πύλη (gateway). Αντίθετα, κάθε φορά που ένας κόμβος προωθεί ένα πακέτο που ανήκει σε μία «ξένη» ροή, αποζημιώνεται λαμβάνοντας κάποια "Helper Credits". Τα "Traffic Credits" είναι δυνατόν να αγοραστούν με αληθινά χρήματα ή ανταλλάσσονται με "Helper Credits" σε σταθμούς εξυπηρέτησης (Service Stations). Ένας σταθμός εξυπηρέτησης μπορεί να παρομοιαστεί με ένα χαμηλού κόστους τερματικό που διαθέτει προπληρωμένες κάρτες και έχει μία ασφαλή και χαμηλού bandwidth σύνδεση με τον provider, η οποία χρησιμοποιείται για ενέργειες πιστοποίησης και πληρωμής.

Ας υποθέσουμε ότι ένας κόμβος-πηγή επιθυμεί να επικοινωνήσει με έναν άλλο κόμβο-προορισμό, ο οποίος είναι τοποθετημένος σε ένα διαφορετικό υποδίκτυο. Αρχικά, ο κόμβος-πηγή ενημερώνει όλους τους κόμβους του δρομολογίου ότι πρόκειται να τους χρησιμοποιήσει, αποστέλλοντας σε αυτούς κατάλληλα μηνύματα πιστοποίησης. Ο κόμβος-πηγή θα πληρώσει μόνο για το μήκος της διαδρομής (σε hop counts) μέχρι την πύλη (gateway) του δικού

του υποδικτύου και ο κόμβος-προορισμός θα πληρώσει για την αντίστοιχη απόσταση που αυτός έχει από την πύλη του δικού του υποδικτύου. Ένας ενδιάμεσος κόμβος αποζημιώνεται για τη βοήθεια που προσφέρει, αμέσως μετά την άφιξη του πακέτου στον επόμενο ενδιάμεσο κόμβο στο δρομολόγιο προς τον προορισμό.

Σε αντίθεση με την τεχνική CASHnet, ο αλγόριθμος Nuglet στηρίζεται στην ακόλουθη αρχή: Κάθε φορά που ένας κόμβος επιθυμεί να αποστείλει ένα πακέτο που ο ίδιος παράγει, οφείλει να πληρώσει με *nuglets*. Το ποσό πληρωμής αντιστοιχεί στον αριθμό των ενδιάμεσων κόμβων που θα διαδραματίσουν ρόλο μεσολαβητή κατά τη διαδικασία προώθησης. Κάθε φορά που ένας κόμβος προωθεί ένα πακέτο που ανήκει σε «ξένη» ροή κίνησης λαμβάνει ένα nuglet.

Όπως παρατηρούμε, στην τεχνική Nuglet κάθε κόμβος έχει μόνο μία πηγή εσόδων: τα ίδια τα nuglets που λαμβάνει καθώς αποζημιώνεται από το δίκτυο για τις υπηρεσίες που προσφέρει. Κατά συνέπεια, ένας κόμβος πρέπει να προωθήσει πολλά «ξένα» πακέτα προκειμένου να κερδίσει αρκετά χρήματα, έτσι ώστε να είναι σε θέση να αποστείλει τα δικά του. Αντίθετα, με την τεχνική CASHnet, κάθε κόμβος έχει δύο πηγές εσόδων: μπορεί είτε να εξαργυρώσει τα εικονικά χρήματα που κερδίζει καθώς προωθεί «ξένα» πακέτα για χάρη άλλων κόμβων (Helper Credits), είτε να πληρώσει με πραγματικά χρήματα.

Τα πειράματα έδειξαν πως η τεχνική Nuglet οδηγεί το δίκτυο πολύ γρήγορα σε φθίνον throughput που τελικά φτάνει στο μηδέν (network starvation). Προκύπτει πως ένας κόμβος δεν είναι σε θέση να καλύψει το κόστος αποστολής των δικών του πακέτων με μόνη πηγή εσόδων την αποζημίωση που αυτός λαμβάνει από το ρόλο του προωθητή. Η CASHnet τεχνική λύνει αυτό το πρόβλημα, δίνοντας το δικαίωμα σε κάθε κόμβο να αγοράσει με χρήματα το δικαίωμά του για αποστολή πακέτων. Τελικά, τα αποτελέσματα των πειραμάτων έδειξαν πως η τεχνική CASHnet ακόμα και με ένα μόνο σταθμό εξυπηρέτησης (service station) οδηγεί το δίκτυο σε καλύτερη απόδοση έναντι της Nuglet τεχνικής. Καθώς δε ο αριθμός των σταθμών εξυπηρέτησης αυξάνει η απόδοση του δικτύου

αυξάνει όλο και περισσότερο συγκρινόμενη με την αντίστοιχη απόδοση που προκύπτει από χρήση του Nuglet αλγορίθμου.

3. Οι βασικές αλγοριθμικές ιδέες της εργασίας

Στην παρούσα εργασία προτείνονται κάποιες αλγοριθμικές προσεγγίσεις που προσφέρουν ορισμένες λύσεις, δημιουργώντας κίνητρα θετικής και συνεργατικής διάθεσης στους κόμβους. Οι αλγόριθμοι που παρουσιάζονται είναι απλοί και εύκολα προσαρμόσιμοι σε κώδικα που υλοποιεί ήδη γνωστούς αλγόριθμους δρομολόγησης σε δίκτυα αυθαίρετης δομής, όπως είναι οι AODV και DSR.

Αρχικά παρουσιάζεται μία δική μας υλοποίηση του αλγορίθμου που προτείνεται στην εργασία [1]. Πρόκειται για τον αλγόριθμο GTFT, που όπως ήδη έχει αναφερθεί, υπαγορεύει πως κάθε κόμβος που λαμβάνει μία αίτηση για προώθηση κίνησης «ξένης» ροής, καλείται να αποφασίσει για το αν θα αποδεχτεί ή όχι την αίτηση αυτή. Η απόφαση βασίζεται στη σύγκριση που πρέπει να γίνει μεταξύ του συνολικού ποσοστού βοήθειας που ο κόμβος αυτός έχει λάβει στο παρελθόν από το σύνολο του δικτύου και του συνολικού ποσοστού βοήθειας που ο ίδιος έχει ήδη προσφέρει σε άλλους κόμβους για όμοιο λόγο. Σε αυτή την περίπτωση, το κίνητρο είναι άμεσο και τα δύο ποσοστά βοήθειας – δοθέν και ληφθέν – εξισορροπούνται πολύ κοντά στο 100% σε σύντομο χρονικό διάστημα από την ενεργοποίηση του δικτύου.

Παρόλα αυτά, πρόβλημα δημιουργείται όταν κάποιοι κόμβοι αρχίζουν να υιοθετούν εγωιστική συμπεριφορά, αφού η συνολική κίνηση που περνά από το δίκτυο μειώνεται δραματικά. Καθώς ένας κόμβος συμπεριφέρεται εγωιστικά, απορρίπτει συνεχώς όλες τις εισερχόμενες αιτήσεις για προώθηση πακέτων «ξένων» ροών κίνησης, με αποτέλεσμα όλοι οι κόμβοι να καταλήγουν σε χαμηλό σημείο σύγκλισης των ποσοστών δοθείσας και ληφθείσας βοήθειας. Το σημείο αυτής της σύγκλισης γίνεται ολοένα και πιο χαμηλό και τείνει στο μηδέν, καθώς αυξάνει η πυκνότητα των κακόβουλων κόμβων.

Στην εργασία αυτή προτείνεται μια παραλλαγή του αλγορίθμου, η οποία χρησιμοποιεί πληροφορία σχετική με την παρελθούσα συμπεριφορά των γειτόνων κάθε κόμβου. Οι κακόβουλες συμπεριφορές εντοπίζονται εύκολα και απομονώνονται αν ο αλγόριθμος επεκταθεί, έτσι ώστε κάθε κόμβος να διατηρεί ξεχωριστή πληροφορία για κάθε έναν από τους γείτονές του. Στην περίπτωση αυτή το αντίκτυπο στο συνολικό όγκο κίνησης που διέρχεται από το δίκτυο από ενδεχόμενη κακή συμπεριφορά είναι πολύ μικρότερο σε σχέση με τα αποτελέσματα που προκύπτουν από την απλή έκδοση του αλγορίθμου.

Ο δεύτερος αλγόριθμος που παρουσιάζεται εδώ προσπαθεί να εξισορροπήσει τους ρυθμούς λαμβανόμενης και προσφερόμενης βοήθειας για κάθε κόμβο. Η χρήση του προτιμάται ιδιαίτερα όταν οι κόμβοι επικοινωνούν μεταξύ τους με κίνηση που ακολουθεί εκρηκτική μορφή (burst). Συγκεκριμένα, σε αυτό τον αλγόριθμο κάθε κόμβος διατηρεί ένα κουβά, του οποίου η τιμή ανανεώνεται κάθε φορά που λαμβάνει χώρα ένα γεγονός, είτε πρόκειται για βοήθεια που λαμβάνεται είτε για βοήθεια που δίνεται. Με αυτό τον τρόπο ο κόμβος αυτός είναι δυνατό να ανταμείβεται από το δίκτυο με μία ποσότητα (θα μπορούσε να θεωρηθεί ως ένας αριθμός κουπονιών) την οποία μπορεί άμεσα να χρησιμοποιήσει για το ποσό πληρωμής που ο ίδιος οφείλει να καταβάλει προκειμένου να διαδραματίσει το ρόλο του ενδιάμεσου.

Στην περίπτωση σεναρίων όπου υπάρχουν κεντρικά σημεία πρόσβασης (access points) προτείνεται η χρήση μιας εκδοχής η οποία είναι δυνατό να προκύψει και από τους δύο αλγορίθμους μετά από κατάλληλη επεξεργασία. Η εκδοχή αυτή υπαγορεύει πως κάθε κόμβος θα προωθεί ποσό κίνησης που αντιστοιχεί σε διαφορετικό βάρος ανάλογα με τη γεωγραφική του θέση και την απόστασή του από το κομβικό σημείο. Για παράδειγμα, οι κόμβοι που βρίσκονται πιο κοντά στο access point θα παρακινούνται από τον αλγόριθμο να αποδέχονται μεγαλύτερο ποσοστό αιτήσεων από το αντίστοιχο ποσοστό δικών τους αιτήσεων που γίνεται αποδεκτό, έτσι ώστε να αποφεύγεται η περίπτωση σύγκλισης του δικτύου σε χαμηλό σημείο.

4. Οργάνωση της εργασίας

Το υπόλοιπο της εργασίας που ακολουθεί είναι οργανωμένο ως εξής: Στο 2ο Μέρος γίνεται μία κατά το δυνατό σύντομη περιγραφή του πρωτοκόλλου DSR, πάνω στο οποίο στηρίζεται η υλοποίηση όλων των αλγορίθμων που παρουσιάζονται στην παρούσα εργασία. Περιγράφονται οι βασικοί μηχανισμοί του πρωτοκόλλου, ενώ ιδιαίτερη έμφαση δίνεται στα σημεία εκείνα που θα υποστούν τροποποιήσεις, προκειμένου να γίνει η ενσωμάτωση των προτεινόμενων αλγορίθμων.

Στο 3ο Μέρος περιγράφεται μία δική μας υλοποίηση του αλγορίθμου GTFT. Πρώτα γίνεται μία αναλυτική θεωρητική περιγραφή του αλγορίθμου και κάποια κομμάτια περιγράφονται με ψευδοκώδικα. Στη συνέχεια, περιγράφεται η ενσωμάτωση του αλγορίθμου στον ήδη υπάρχοντα κώδικα του DSR με όσο το δυνατό πιο λεπτομερή ανάλυση των αλλαγών και προσθηκών που κρίθηκαν απαραίτητες (π.χ. δημιουργία νέων ρουτινών και τύπων πακέτων).

Το 4ο Μέρος πραγματεύεται τις νέες μεθόδους και στρατηγικές που προτείνονται. Ξεκινά με τη θεωρητική προσέγγιση κάθε αλγορίθμου και καταλήγει σε λεπτομέρειες σχετικές με την προγραμματιστική του υλοποίηση.

Οι παράμετροι των προσομοιώσεων, τα διάφορα πειραματικά σενάρια και τα αντίστοιχα αποτελέσματα αυτών παρουσιάζονται αναλυτικά στο 5ο Μέρος της εργασίας. Τα αποτελέσματα παρουσιάζονται κυρίως με τη μορφή γραφικών παραστάσεων, ενώ συγκεντρωτικοί πίνακες χρησιμοποιούνται όταν επιχειρούνται συγκριτικές μελέτες.

Το 6ο Μέρος αποτελεί την κατακλείδα της εργασίας συνοψίζοντας τα κυριότερα σημεία που εξετάστηκαν και αναφέροντας σημεία που θα μπορούσαν να αποτελέσουν ζητήματα μελλοντικής εργασίας.

Τέλος, υπάρχουν και τρία παραρτήματα. Στο παράρτημα Α υπάρχει συγκεντρωτικός κατάλογος με τα σχήματα και τους πίνακες που συναντά κανείς στην εργασία. Στο παράρτημα Β παρουσιάζονται δύο πίνακες που συνοψίζουν τα ονόματα και τα χαρακτηριστικά

των νέων συναρτήσεων και τύπων πακέτων που δημιουργήθηκαν και προσαρτήθηκαν στον κώδικα του DSR, ενώ το παράρτημα Γ φιλοξενεί το πιο σημαντικό μέρος του πηγαίου κώδικα που γράφτηκε γι' αυτή την εργασία.

Β' ΜΕΡΟΣ : ΤΟ ΠΡΩΤΟΚΟΛΛΟ DSR

Η υλοποίηση των αλγοριθμικών ιδεών που προτείνονται στην παρούσα εργασία έγινε με την προσθήκη κάποιων επιπλέον ρουτινών στον κώδικα που υλοποιεί τον αλγόριθμο δρομολόγησης DSR. Για το λόγο αυτό κρίνεται σκόπιμη μία τουλάχιστον γενική περιγραφή του αλγορίθμου αυτού.

1. Τα βασικά χαρακτηριστικά του πρωτοκόλλου DSR

Το πρωτόκολλο δυναμικής δρομολόγησης πηγής (DSR) είναι ένα απλό και αποδοτικό πρωτόκολλο δρομολόγησης που σχεδιάστηκε ειδικά για χρήση σε ασύρματα δίκτυα αυθαίρετης δομής πολλαπλών βημάτων. Κάθε δίκτυο που χρησιμοποιεί τον αλγόριθμο DSR

επιτυγχάνει την αυτό-οργάνωση και αυτό-ρύθμισή του, χωρίς να απαιτεί τη ύπαρξη δομημένης δικτυακής υποδομής η κεντρικής διαχείρισης. Οι κόμβοι συνεργάζονται για να προωθήσουν, μέσω πολλαπλών βημάτων, πακέτα ο ένας στον άλλο, επιτρέποντας έτσι την επικοινωνία απομακρυσμένων κόμβων, οι οποίοι βρίσκονται σε αποστάσεις μεγαλύτερες από την ακτίνα ασύρματης μετάδοσης. Καθώς διάφοροι κόμβοι στο δίκτυο μετακινούνται, αποσυνδέονται ή συνδέονται σε αυτό και καθώς οι συνθήκες της ασύρματης μετάδοσης αλλάζουν η δρομολόγηση καθορίζεται και συντηρείται αυτόματα από το DSR.

Το πρωτόκολλο DSR επιτρέπει στους κόμβους να ανακαλύπτουν δυναμικά μία διαδρομή πηγής, η οποία οδηγεί – ενδεχομένως κατά μήκος πολλαπλών βημάτων – σε οποιονδήποτε δυνατό προορισμό στο αδόμητο δίκτυο. Κάθε πακέτο που δρομολογείται μεταφέρει στην επικεφαλίδα του (header) μία πλήρη και διατεταγμένη λίστα των κόμβων τους οποίους θα χρησιμοποιήσει ως ενδιάμεσους σταθμούς μέχρι να φτάσει στον προορισμό του. Η τεχνική αυτή δίνει τη δυνατότητα στους ενδιάμεσους κόμβους να μη διατηρούν πλήρως ενημερωμένη πληροφορία δρομολόγησης προκειμένου να συμμετάσχουν στη διαδικασία προώθησης πακέτων που ανήκουν σε «ξένες» ροές.

2. Οι μηχανισμοί λειτουργίας του DSR

Η λειτουργία του DSR επιτυγχάνεται από δύο βασικούς μηχανισμούς, οι οποίοι διαχωρίζονται με σαφή όρια. Οι μηχανισμοί αυτοί συνεργάζονται άψογα μεταξύ τους και πραγματοποιούν την ανακάλυψη και συντήρηση των διαδρομών πηγής σε ένα δίκτυο αυθαίρετης δομής. Συγκεκριμένα:

- **Ο μηχανισμός «Ανακάλυψη Διαδρομής» ("Route Discovery Mechanism")** είναι αυτός χάρη στον οποίο ένας κόμβος – πηγή S που επιθυμεί να αποστείλει ένα πακέτο σε ένα κόμβο – προορισμό D αποκτά μία διαδρομή πηγής (source route) προς τον προορισμό. Ο μηχανισμός αυτός χρησιμοποιείται μονάχα όταν μία πηγή έχει

να στείλει ένα πακέτο σε κάποιο προορισμό και δε γνωρίζει ήδη μια διαδρομή προς αυτόν.

- **Ο μηχανισμός «Συντήρηση Διαδρομής» ("Route Maintenance Mechanism")** είναι αυτός χάρη στον οποίο ένας κόμβος – πηγή S διαπιστώνει εάν η τοπολογία του δικτύου άλλαξε, κατά τέτοιο τρόπο ώστε δεν είναι δυνατή η χρησιμοποίηση μίας ήδη γνωστής διαδρομής προς ένα προορισμό D (π.χ. κάποια ζεύξη έχει κοπεί). Αμέσως μόλις ο μηχανισμός αυτός διαπιστώσει ότι μία υπάρχουσα διαδρομή δεν ισχύει πια, ο S μπορεί στο μέλλον είτε να χρησιμοποιήσει μία εναλλακτική ήδη γνωστή διαδρομή προς τον D είτε να ενεργοποιήσει το μηχανισμό «Ανακάλυψη Διαδρομής». Ο μηχανισμός αυτός ενεργοποιείται σε κάθε περίπτωση που ο S ξεκινά τη διαδικασία αποστολής πακέτου προς τον D.

Όπως ήδη έγινε φανερό, στο πρωτόκολλο DSR τόσο ο μηχανισμός της ανακάλυψης μιας διαδρομής όσο και ο μηχανισμός συντήρησης αυτής λειτουργούν εξ' ολοκλήρου «κατ' απαίτηση». Σε αντίθεση με άλλα πρωτόκολλα δρομολόγησης, το DSR δεν απαιτεί περιοδικές αποστολές πακέτων κάποιου τύπου μέσα στο δίκτυο. Δεν υπάρχει, για παράδειγμα, ανάγκη για περιοδικές διαφημίσεις διαδρομών, πακέτα ανίχνευσης κατάστασης, ζεύξεων ή ανακάλυψης γειτόνων και δεν αφήνεται καμία από αυτές τις λειτουργίες σε πρωτόκολλα χαμηλότερων επιπέδων. Η εξ' ολοκλήρου «κατ' απαίτηση» λειτουργία του DSR και η παντελής έλλειψη περιοδικών λειτουργιών επιτρέπουν σε ακίνητα δίκτυα αυθαίρετης δομής να μηδενίζουν τα πλεονάζοντα (overhead) πακέτα ελέγχου μετά από μικρό χρόνο λειτουργίας που θα επιτρέψει την ανακάλυψη των διαδρομών που απαιτούνται για την επικοινωνία των κόμβων. Σε κινούμενα δίκτυα, η πλεονάζουσα πληροφορία που εισάγουν οι μηχανισμοί του DSR ανέρχεται στο ελάχιστο δυνατό ποσό που θα επιτρέψει τη σωστή διαχείριση των χρησιμοποιούμενων διαδρομών. Αυτό συμβαίνει διότι αλλαγές στην τοπολογία του δικτύου, οι οποίες δεν επηρεάζουν τις διαδρομές που χρησιμοποιούνται δεν προκαλούν την αντίδραση του πρωτοκόλλου.

3. Η μοντελοποίηση του DSR στο Opnet

Όπως έχει ήδη αναφερθεί, όλοι οι αλγόριθμοι υλοποιήθηκαν στον ήδη υπάρχοντα κώδικα του DSR. Συγκεκριμένα, χρησιμοποιήθηκε το μοντέλο του DSR για το εργαλείο προσομοίωσης OPNET, όπως αυτό έχει κατασκευαστεί από το WCTG (Wireless Communications Technologies Group) του NIST (National Institute of Standards and Technology) των ΗΠΑ. Πρόκειται για ένα ευρέως διαδεδομένο μοντέλο στην ερευνητική κοινότητα, το οποίο έχει χρησιμοποιηθεί και σε άλλες έρευνες. Παρ' όλα αυτά, όμως, αρκετά λάθη εντοπίσθηκαν τόσο στον κώδικα του DSR όσο και στον τρόπο με τον οποίο υποστηρίζεται ο κώδικας αυτός από το εργαλείο Opnet.

Προκειμένου να γίνει κατανοητή η λειτουργία του συγκεκριμένου μοντέλου κρίνεται σκόπιμο να προηγηθεί μία σύντομη και γενική περιγραφή του τρόπου μοντελοποίησης που χρησιμοποιείται από το Opnet. Η μοντελοποίηση αυτή διαιρείται σε τρία επίπεδα. Το ανώτερο επίπεδο είναι το επίπεδο δικτύου (network level), στο οποίο περιγράφονται τα συστατικά και η τοπολογία του δικτύου. Το μεσαίο είναι το επίπεδο κόμβου (node level), στο οποίο περιγράφεται κάθε στοιχείο που χρησιμοποιείται στο παραπάνω επίπεδο με τη μορφή μιας στοίβας από διαδικασίες. Τέλος, το τρίτο επίπεδο είναι το επίπεδο διαδικασίας (process level), στο οποίο περιγράφεται κάθε διαδικασία ενός κόμβου με τη βοήθεια μίας μηχανής πεπερασμένων καταστάσεων, η οποία υλοποιείται με χρήση της γλώσσας Proto – C (ένα υπερσύνολο της C).

Στις παραγράφους που ακολουθούν περιγράφονται αναλυτικά τα επίπεδα κόμβου και διαδικασίας, καθώς αυτά είναι που ορίζουν το καθ' αυτό πρωτόκολλο. Στο επίπεδο δικτύου απλώς τοποθετούνται οι κόμβοι σε κάποιες θέσεις και δημιουργείται η επιθυμητή τοπολογία προκειμένου να εκτελεστούν τα πειράματα. Επίσης, στο επίπεδο αυτό τίθενται οι βασικές παράμετροι, όπως για παράδειγμα η εμβέλεια μετάδοσης (transmission range) και η κινητικότητα των κόμβων (mobility).

3.1. Το επίπεδο κόμβου του DSR

Το επίπεδο κόμβου αποτελείται από μία στοίβα από διαδικασίες. Κάθε διαδικασία ή ομάδα από διαδικασίες αντιστοιχούν με τη σειρά τους σε ένα από τα επίπεδα του μοντέλου διαστρωμάτωσης OSI (βλέπε σχήμα 1).

Το φυσικό επίπεδο αποτελείται από ένα πομπό και ένα δέκτη: wireless_lan_transmitter_0 και wireless_lan_receiver_0. Τα μπλοκς αυτά δεν αποτελούν διαδικασίες του Opnet, αλλά ορίζουν τον κώδικα C που απαιτείται για τη χρήση του μηχανισμού ασύρματης επικοινωνίας. Ο κώδικας αυτός έχει αναπτυχθεί από τους ίδιους τους δημιουργούς του Opnet. Συγκεκριμένα, το φυσικό επίπεδο και το επίπεδο ζεύξης του μοντέλου χρησιμοποιούνται αυτούσια όπως δημιουργήθηκαν για το μοντέλο Wireless_Lan.

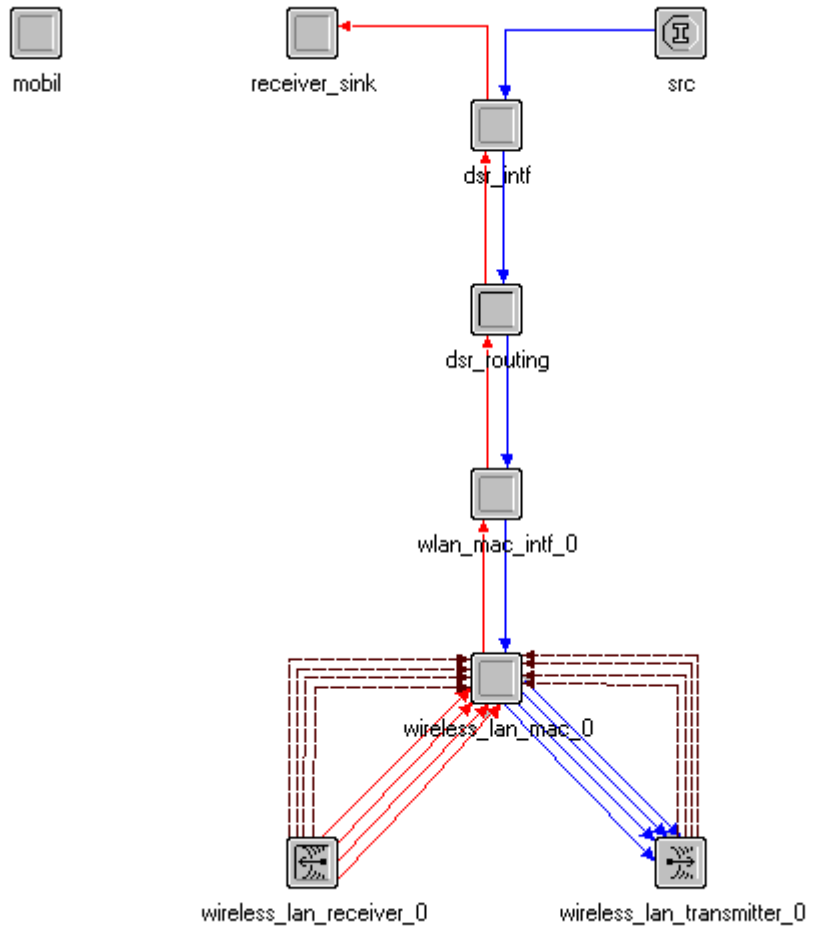
Το επίπεδο ζεύξης δεδομένων βασίζεται στο μοντέλο IEEE 802.11 του Opnet. Οι δημιουργοί του μοντέλου του DSR έχουν κάνει ορισμένες τροποποιήσεις, όπως για παράδειγμα την προσθήκη του μηχανισμού αδιάκριτης λήψης και την αποστολή συγκεκριμένων μηνυμάτων επιβεβαίωσης και σφαλμάτων επιπέδου MAC. Το επίπεδο ζεύξης αποτελείται από δύο διαδικασίες – μπλόκς: το wireless_lan_mac_0, το οποίο είναι το καθ’ αυτό πρωτόκολλο 802.11 και το wireless_lan_intf_0, το οποίο αποτελεί μία απαιτούμενη διεπαφή με το ανώτερο επίπεδο.

Το επίπεδο δικτύου αποτελεί τον πυρήνα του μοντέλου κόμβου του DSR, αφού περιέχει τη διαδικασία δρομολόγησης. Όπως και το προηγούμενο μοντέλο, έτσι και αυτό είναι χωρισμένο σε δύο διαδικασίες: η διαδικασία δρομολόγησης καλείται dsr_routing ενώ η διαδικασία διεπαφής με τα ανώτερα επίπεδα καλείται dsr_intf.

Τα ανώτερα επίπεδα προσομοιώνονται κάνοντας χρήση δύο διαδικασιών. Η src είναι μία διαδικασία του Opnet, η οποία παράγει τα πακέτα δεδομένων προκειμένου να δημιουργηθεί η απαιτούμενη κάθε φορά κίνηση πληροφορίας στο δίκτυο. Αντίθετα, η διαδικασία receiver είναι υπεύθυνη για την παραλαβή των πακέτων δεδομένων που καταφθάνουν σε ένα κόμβο. Αφού τα πακέτα

υποστούν την κατάλληλη επεξεργασία τελικά καταστρέφονται από τη διαδικασία αυτή.

Τέλος, στο μοντέλο κόμβου υπάρχει και η διαδικασία mobil. Η διαδικασία αυτή αναλαμβάνει τον καθορισμό της κινητικότητας κάθε κόμβου στο δίκτυο. Συγκεκριμένα, η default κατάσταση είναι η κίνηση μπιλιάρδου, όπου κάθε κόμβος επιλέγει μία κατεύθυνση και την ακολουθεί με σταθερή ταχύτητα μέχρι τα γεωγραφικά όρια του χώρου προσομοίωσης, οπότε και «ανακλάται» επιλέγοντας μία νέα τυχαία κατεύθυνση.



Σχήμα 1: Το επίπεδο κόμβου του DSR

3.2. Το επίπεδο διαδικασίας του DSR

Στην ενότητα αυτή γίνεται μία περιγραφή του τρόπου υλοποίησης του μοντέλου της διαδικασίας dsr_routing. Το ενδιαφέρον εστιάζεται κυρίως στους μηχανισμούς ανακάλυψης και συντήρησης διαδρομής, ενώ γίνεται και μία αναφορά στον τρόπο με τον οποίο οργανώνονται οι μνήμες cache που διατηρούν τις διαδρομές. Τέλος, περιγράφεται η λειτουργία της μηχανής καταστάσεων που είναι υπεύθυνη για την υλοποίηση των παραπάνω.

3.2.1. Ο μηχανισμός «Ανακάλυψης Διαδρομής»

Ο μηχανισμός ανακάλυψης διαδρομής υλοποιείται κάνοντας χρήση της διαδικασίας μη προωθητικών αιτήσεων, δηλαδή, όπως ακριβώς περιγράφεται αυτός στο πρωτόκολλο του DSR. Το όριο των βημάτων εισάγεται σαν πεδίο της επικεφαλίδας των πακέτων του DSR και δεν υλοποιείται με το πεδίο TTL του IP. Επίσης, η υλοποίηση υποθέτει ότι η διάμετρος του δικτύου – άρα και ο μέγιστος αριθμός βημάτων – είναι 7. Ακόμα, χρησιμοποιείται η τεχνική της απάντησης με χρήση αποθηκευμένων διαδρομών. Υπάρχει η τεχνική αποφυγής καταιγισμού απαντήσεων, η οποία υλοποιείται με χρήση της καθυστέρησης των απαντήσεων, αλλά και με χρήση του μηχανισμού αδιάκριτης λήψης. Δεν υλοποιούνται η αναζήτηση δακτυλίου και η αποθήκευση έμμεσης πληροφορίας δρομολόγησης – με εξαίρεση στα πακέτα σφάλματος.

Σε περίπτωση αποστολής πακέτου αίτησης διαδρομής μετά από τη λήψη ενός σφάλματος διαδρομής, δε χρησιμοποιείται ο μηχανισμός μη προωθητικής αίτησης, αφού οι γειτονικοί κόμβοι δεν είναι δυνατό να έχουν στην cache τους έγκυρη διαδρομή για τον προορισμό. Τέλος, δεν υλοποιείται ο μηχανισμός αυξημένης διάδοσης μηνυμάτων σφάλματος με την επισύναψη του πακέτου σφάλματος διαδρομής στο πακέτο της νέας αίτησης.

3.2.2. Ο μηχανισμός «Συντήρησης Διαδρομής»

Στο μοντέλο του DSR ως επίπεδο ζεύξης χρησιμοποιείται το επίπεδο MAC του IEEE 802.11. Έτσι, αυτό είναι το επίπεδο που παρέχει τα μηνύματα επιβεβαίωσης και σφαλμάτων που απαιτούνται

από το επίπεδο δρομολόγησης. Πιο συγκεκριμένα, το μοντέλο δε διαθέτει ενταμιευτή επαναμετάδοσης, αλλά μέσω ενός χρονομετρητή παρατηρεί ότι δεν έχει ληφθεί η απαραίτητη επιβεβαίωση για ένα πακέτο δεδομένων που μεταδόθηκε στο παρελθόν. Ο μηχανισμός αυτός παράλληλα ελέγχει και την ορθή λειτουργία του επιπέδου ζεύξης, καθώς σε προκαθορισμένο χρόνο πρέπει να ληφθεί είτε ένα πακέτο επιβεβαίωσης είτε ένα πακέτο σφάλματος για κάθε πακέτο που έχει ήδη αποσταλεί.

Όταν ένας κόμβος, μέσω του μηχανισμού αδιάκριτης λήψης, ακούσει ένα πακέτο σφάλματος διαδρομής, αμέσως ελέγχει την cache διαδρομών του για να διαγράψει τη ζεύξη που το πακέτο φέρει ως κομμένη, ανεξάρτητα από το εάν περιλαμβάνεται στη διαδρομή πηγής του πακέτου ή όχι.

3.2.3. Η οργάνωση των caches διαδρομών

Η cache διαδρομών κάθε κόμβου είναι ένας δι-διάστατος πίνακας. Οι γραμμές του δεικτοδοτούνται με τις διευθύνσεις επιπέδου δικτύου των κόμβων. Κάθε γραμμή του περιέχει μία λίστα από τις διευθύνσεις των κόμβων οι οποίοι αποτελούν τη διαδρομή πηγής από τον παρόντα κόμβο προς τον προορισμό. Δεδομένου αυτού του τρόπου οργάνωσης, προκύπτει ότι για κάθε δυνατό προορισμό στο δίκτυο μπορεί να αποθηκευτεί μόνο μία διαδρομή. Πειραματικά, αποδεικνύεται ότι η διαδρομή αυτή είναι και η συντομότερη.

3.3. Η μηχανή καταστάσεων του μοντέλου διαδικασίας του DSR

Στο σχήμα 2 παρουσιάζεται η μηχανή πεπερασμένων καταστάσεων που υλοποιεί το μοντέλο διαδικασίας του DSR. Ακολουθεί μία αναλυτική περιγραφή του ρόλου που διαδραματίζει κάθε μία από αυτές τις καταστάσεις.

Pre-init: Η πρώτη αρχικοποίηση του κόμβου λαμβάνει χώρα. Συγκεκριμένα, αποδίδεται στον κόμβο η διεύθυνση δικτύου και ελέγχεται η εγκυρότητα αυτής στο υπόλοιπο δίκτυο.

Init: Γίνεται αρχικοποίηση όλων των μεταβλητών, των στατιστικών και των παραμέτρων προσομοίωσης που χρησιμοποιούνται στο μοντέλο διαδικασίας.

Idle: Είναι η κατάσταση στην οποία η διαδικασία αναμένει να συμβεί ένα γεγονός (event).

Upper Layer Arrival: Η κατάσταση αυτή χειρίζεται κάθε πακέτο δεδομένων, το οποίο δημιουργείται από τη διαδικασία src του κόμβου και φτάνει στη διαδικασία δρομολόγησης μέσω της διεπαφής dsr_intf έχοντας ένα συγκεκριμένο κόμβο – προορισμό.

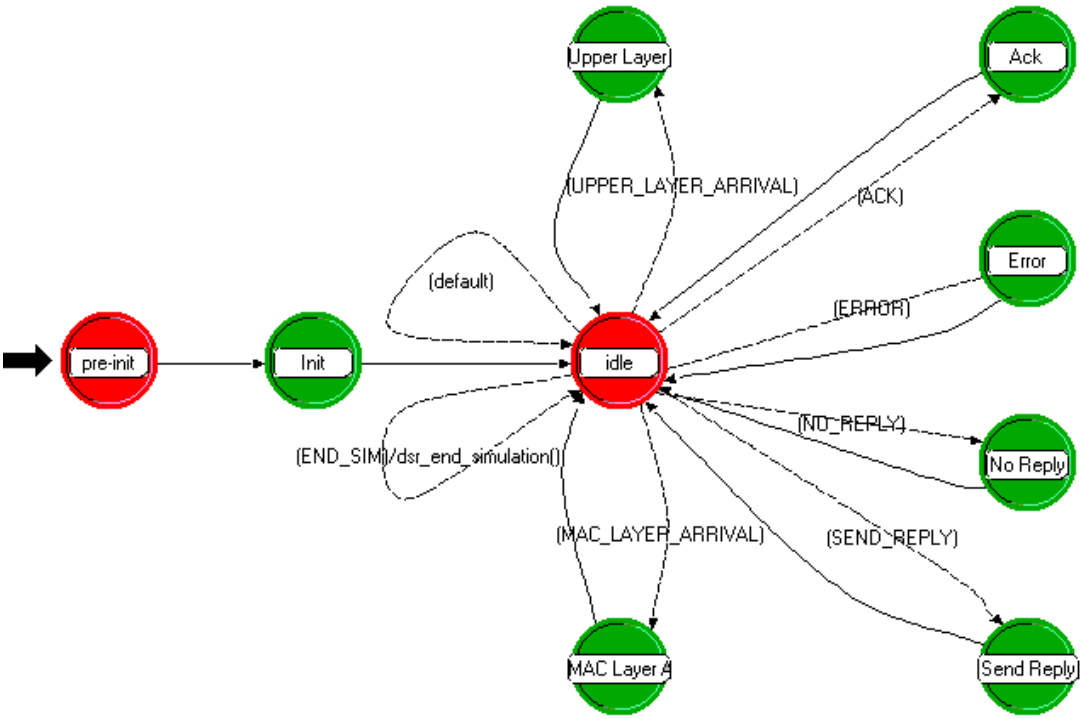
MAC Layer Arrival: Η κατάσταση αυτή χειρίζεται κάθε πακέτο που παραλαμβάνεται από το επίπεδο ζεύξης. Ανάλογα με τον τύπο του πακέτου (δεδομένων, αίτησης, απάντησης ή σφάλματος) καλείται ο αντίστοιχος μηχανισμός του DSR που χρειάζεται για να το χειριστεί.

Send Reply: Καλείται όταν παρέλθει το χρονικό διάστημα αναμονής (timeout) που αφορά σε ένα πακέτο αίτησης που έστειλε ένας κόμβος. Το γεγονός αυτό σηματοδοτεί ότι το προηγούμενο βήμα της αίτησης διαδρομής απέτυχε, συνεπώς ένα νέο πακέτο αίτησης παράγεται και στέλνεται από τον κόμβο.

Ack: Χειρίζεται κάθε επιβεβαίωση που φτάνει από το επίπεδο ζεύξης 802.11. Έτσι βεβαιώνεται ότι η ζεύξη που χρησιμοποιήθηκε για την αποστολή του τελευταίου πακέτου δεδομένων είναι όντως ενεργή και συνεπώς ο κόμβος μπορεί να την χρησιμοποιήσει ξανά σύντομα για να στείλει πακέτα δεδομένων μέσω αυτής. Επίσης, ακυρώνει την αντίστροφη μέτρηση που είχε τεθεί από το μηχανισμό ελέγχου σφάλματος, ο οποίος περίμενε αυτή την επιβεβαίωση.

Error: Καλείται όταν ένα σφάλμα ληφθεί από το επίπεδο ζεύξης 802.11. Η ζεύξη που χρησιμοποιήθηκε για να αποσταλεί το μη επιβεβαιωμένο (unacknowledged) πακέτο δεδομένων χαρακτηρίζεται πλέον κομμένη (broken), διαγράφεται από την cache διαδρομών ενώ ένα πακέτο σφάλματος αποστέλλεται με προορισμό την πηγή του πακέτου.

No Reply: Καλείται όταν δε λαμβάνεται απάντηση σε προηγηθείσα αίτηση διαδρομής. Αν βρισκόμαστε μέσα στα αποδεκτά χρονικά όρια αναμονής, κατασκευάζεται και μεταδίδεται ένα νέο πακέτο αίτησης στον επιθυμητό προορισμό. Διαφορετικά, θεωρείται ότι έχει γίνει κάποιο λάθος και η προσομοίωση τερματίζεται.

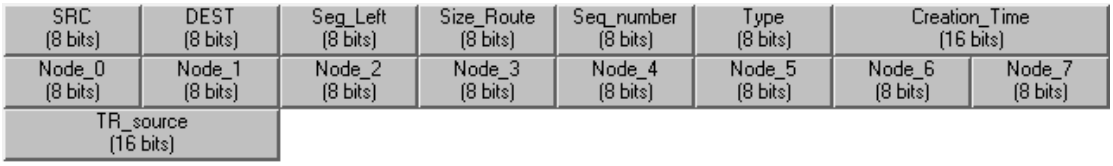


Σχήμα 2: Η μηχανή καταστάσεων του DSR

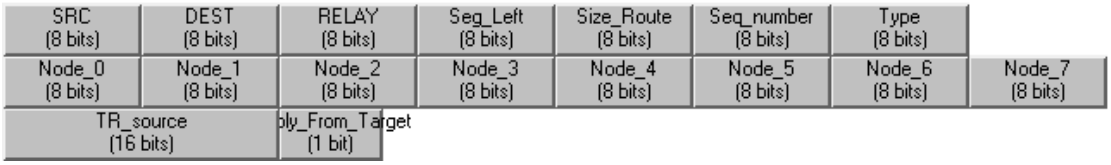
3.4. Οι τύποι των πακέτων του DSR

Το πρωτόκολλο DSR υπαγορεύει τη χρήση πέντε τύπων πακέτων. Πρόκειται για τα πακέτα αίτησης για προώθηση (request_packet), απάντησης σε αίτηση για προώθηση (reply_packet), δεδομένων (data_packet), δεδομένων ανώτερου επιπέδου (upper_data_packet) και λάθους (error_packet). Στα σχήματα που ακολουθούν φαίνεται η μορφή των πακέτων αυτών.

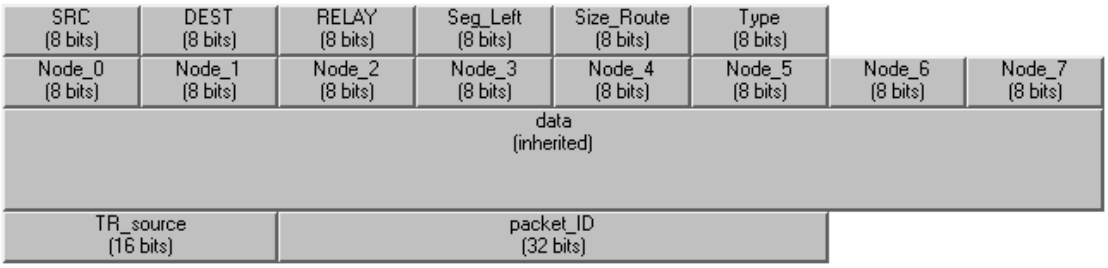
Πληροφορίες σχετικές με τη σημασία και τη χρησιμότητα των πεδίων υπάρχουν στο παράρτημα Α.



Σχήμα 3: dsr_request_packet

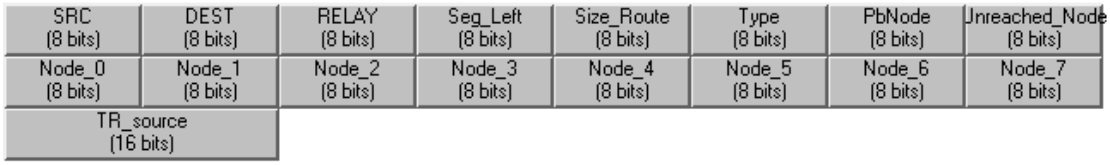


Σχήμα 4: dsr_reply_packet



Σχήμα 5: dsr_data_packet

Σχήμα 6: dsr_upper_data_packet



Σχήμα 7: dsr_error_packet

Γ' ΜΕΡΟΣ : ΥΛΟΠΟΙΗΣΗ ΤΟΥ GTFT – ΕΝΣΩΜΑΤΩΣΗ ΣΤΟ DSR

Σε αυτό το μέρος της εργασίας περιγράφεται ο τρόπος με τον οποίο ενσωματώθηκε ο αλγόριθμος GTFT στον κώδικα του πρωτοκόλλου DSR. Αρχικά, γίνεται μία θεωρητική παρουσίαση του αλγορίθμου, περιγράφονται τα χαρακτηριστικά του και προσδιορίζονται πλήρως οι σχέσεις που χρησιμοποιούνται. Στη συνέχεια, παρουσιάζεται ο τρόπος με τον οποίο έγινε η προγραμματιστική υλοποίηση στο Opnet και περιγράφονται οι βασικότερες αλλαγές και προσθήκες που έγιναν στον κώδικα του DSR.

1. Θεωρητική προσέγγιση του GTFT

Ο αλγόριθμος αυτός τείνει να εξισώνει το ποσοστό αιτήσεων για προώθηση κίνησης που εξυπηρετούνται από ένα κόμβο Α για λογαριασμό άλλων κόμβων με το αντίστοιχο ποσοστό αιτήσεων που εξυπηρετούνται από άλλους κόμβους για λογαριασμό του κόμβου Α. Ένας ενδιάμεσος κόμβος είναι σε θέση να αποδεχτεί μία συγκεκριμένη αίτηση μόνο στην περίπτωση που το ποσοστό αιτήσεων που έχει εξυπηρετηθεί συνολικά από το δίκτυο για λογαριασμό αυτού του κόμβου είναι μεγαλύτερο από το αντίστοιχο ποσοστό αιτήσεων που ο ίδιος αυτός κόμβος έχει εξυπηρετήσει για λογαριασμό άλλων.

Με αυτό τον τρόπο, κάθε κόμβος φαίνεται να έχει σημαντικό κίνητρο προκειμένου να αποδεχτεί μία εξωτερική αίτηση προώθησης. Πιο συγκεκριμένα, θεωρούμε το λόγο

$$\frac{\#\ of\ \ requests\ served\ by\ A}{\#\ of\ \ requests\ received\ by\ A} \quad (1)$$

ως το ποσοστό αιτήσεων που έχουν εξυπηρετηθεί από ένα κόμβο Α για λογαριασμό άλλων κόμβων. Ακολουθώντας το ίδιο πρότυπο, θεωρούμε το λόγο

$$\frac{\#\ of\ \ requests\ served\ \ \ for\ A}{\#\ of\ \ requests\ made\ by\ A} \quad (2)$$

ως το ποσοστό αιτήσεων που έχουν εξυπηρετηθεί για λογαριασμό ενός κόμβου Α από τους υπόλοιπους κόμβους του δικτύου συνολικά. Προκειμένου να αποφευχθεί κάθε πιθανότητα σύγχυσης, στο υπόλοιπο της εργασίας ακολουθείται η εξής σύμβαση:

$$amount\ of\ help\ i\ gave \;=\; \frac{requests\ handled\ for\ others}{requests\ made\ to\ me\ by\ others}$$

(3)

και

$$amount\ of\ help\ i\ received \;=\; \frac{requests\ handled\ for\ me}{requests\ made\ by\ me}$$

(4)

Σε αυτή την περίπτωση, η συνθήκη που ελέγχεται από κάθε κόμβο προκειμένου αυτός να αποφανθεί αν θα αποδεχτεί ή όχι μία συγκεκριμένη αίτηση προώθησης κίνησης κατά την άφιξή της παρουσιάζεται στη συνέχεια:

$$amount\ of\ help\ i\ received \;+\; gen \;>\; amount\ of\ help\ i\ gave$$

(5)

όπου *gen* είναι ένας μικρός θετικός ακέραιος αριθμός, ο οποίος χρησιμοποιείται για την αποφυγή φαινομένων αδιεξόδου (deadlock) κυρίως κατά τη διάρκεια της εκκίνησης του δικτύου.

Στις παραγράφους που ακολουθούν δίνεται μία κατά το δυνατό αναλυτική περιγραφή του αλγορίθμου. Για λόγους που αποσκοπούν στην καλύτερη κατανόηση του κειμένου, η περιγραφή του αλγορίθμου διακρίνεται σε δύο βασικές περιπτώσεις. Η πρώτη περίπτωση αφορά τις διαδικασίες που λαμβάνουν χώρα κατά την άφιξη ενός πακέτου που αναγγέλλει την αίτηση για προώθηση κίνησης, ενώ η δεύτερη περιγράφει τις διαδικασίες που ενεργοποιούνται κατά την άφιξη ενός πακέτου που γνωστοποιεί την απάντηση σε μία τέτοιου τύπου προηγούμενη αίτηση.

1.1. Άφιξη πακέτου αίτησης για προώθηση

Ας ξεκινήσουμε παρουσιάζοντας τις διαδικασίες που ενεργοποιούνται σε ένα κόμβο κατά την άφιξη σε αυτόν ενός πακέτου που ζητά την άδεια για προώθηση κίνησης. Αμέσως μόλις αναγνωριστεί από τον κόμβο ο τύπος του συγκεκριμένου πακέτου, γίνεται σύγκριση μεταξύ των λόγων (3) και (4) κάνοντας χρήση της συνθήκης (5).

Αν το αποτέλεσμα είναι θετικό, δηλαδή η συνθήκη (5) είναι αληθής, η αίτηση για προώθηση γίνεται αποδεκτή και στη συνέχεια:

- Αν μετά τον κόμβο Α υπάρχει επόμενος ενδιάμεσος σταθμός, τότε αρκεί η απλή προώθηση του πακέτου αίτησης στον επόμενο υποψήφιο ενδιάμεσο κόμβο, προφανώς, αφού το πακέτο υποστεί πρώτα κατάλληλη επεξεργασία ώστε να συνεχίσει σωστά το «ταξίδι» του.

- Σε διαφορετική περίπτωση, δηλαδή αν μετά τον κόμβο Α δεν υπάρχει επόμενος ενδιάμεσος σταθμός, ενεργοποιείται άμεσα η διαδικασία θετικής απάντησης. Συγκεκριμένα, κατασκευάζεται ένα νέο πακέτο - τύπου απάντησης – το οποίο έχει ως προορισμό τον κόμβο που ήταν η πηγή της αίτησης. Αυτό το πακέτο συμπεριλαμβάνει στις πληροφορίες του ολόκληρο το αντίθετο δρομολόγιο που οδηγεί πίσω στην πηγή της αίτησης. Επίσης, έχει τον ίδιο αύξοντα αριθμό με το πακέτο της αντίστοιχης αίτησης, ενώ στο πεδίο απάντησης που διαθέτει τίθεται η θετική τιμή 1 (ένα).

Διαφορετικά, αν το αποτέλεσμα είναι αρνητικό, δηλαδή η συνθήκη (5) είναι ψευδής, η αίτηση για προώθηση απορρίπτεται και στη συνέχεια:

- Ο κόμβος Α ενεργοποιεί άμεσα τη διαδικασία αρνητικής απάντησης. Συγκεκριμένα, κατασκευάζεται ένα νέο πακέτο - τύπου απάντησης – το οποίο έχει ως προορισμό τον κόμβο που ήταν η πηγή της αίτησης. Αυτό το πακέτο συμπεριλαμβάνει στις πληροφορίες του ολόκληρο το αντίθετο δρομολόγιο που οδηγεί πίσω στην πηγή της αίτησης. Επίσης, έχει τον ίδιο αύξοντα

αριθμό με το πακέτο της αντίστοιχης αίτησης, ενώ στο πεδίο απάντησης που διαθέτει τίθεται η αρνητική τιμή 0 (μηδέν).

1.2. Άφιξη πακέτου απάντησης σε αίτηση για προώθηση

Συνεχίζουμε παρουσιάζοντας τις διαδικασίες που ενεργοποιούνται σε ένα κόμβο κατά την άφιξη σε αυτόν ενός πακέτου απαντά σε προηγούμενη αίτηση άδειας για προώθηση κίνησης. Αμέσως μόλις αναγνωριστεί από τον κόμβο ο τύπος του συγκεκριμένου πακέτου, γίνεται ανάγνωση του πεδίου απάντησης. Συγκεκριμένα:

Αν ο κόμβος Α είναι ο προορισμός του πακέτου απάντησης, δηλαδή είναι η πηγή του πακέτου αίτησης, τότε:

- Αν η τιμή του πεδίου απάντησης είναι θετική, γεγονός που σημαίνει ότι η αίτηση για προώθηση έχει γίνει αποδεκτή από όλους τους ενδιάμεσους σταθμούς, ο κόμβος Α είναι έτοιμος για την αποστολή των πακέτων δεδομένων του, αφού πρώτα αυξήσει κατά ένα το μετρητή *requests handled form me*.

- Σε διαφορετική περίπτωση, δηλαδή αν η τιμή του πεδίου απάντησης είναι αρνητική, γεγονός που σημαίνει ότι η αίτηση για προώθηση απορρίφθηκε από κάποιο ενδιάμεσο σταθμό, ο κόμβος Α απλώς καταγράφει την απόρριψη με τη μορφή αρνητικής απάντησης, χωρίς φυσικά να ενεργοποιήσει τη συνάρτηση που είναι υπεύθυνη για την αποστολή των πακέτων δεδομένων.

Διαφορετικά, αν ο κόμβος Α δεν είναι ο προορισμός του πακέτου απάντησης, δηλαδή είναι κάποιος ενδιάμεσος σταθμός, τότε:

- Αν η τιμή του πεδίου απάντησης είναι θετική, γεγονός που σημαίνει ότι όλοι οι ενδιάμεσοι σταθμοί – συμπεριλαμβανομένου και του παρόντος – έχουν αποδεχτεί την αίτηση για προώθηση, ο κόμβος Α αναλαμβάνει την αποστολή του πακέτου απάντησης στον επόμενο ενδιάμεσο σταθμό ακολουθώντας το αντίθετο δρομολόγιο από εκείνο της αντίστοιχης αίτησης, αφού πρώτα αυξήσει κατά ένα το μετρητή *requests handled for others*.

- Σε διαφορετική περίπτωση, δηλαδή αν η τιμή του πεδίου απάντησης είναι αρνητική, ο κόμβος Α απλώς φροντίζει για την αποστολή του πακέτου απάντησης στον επόμενο ενδιάμεσο σταθμό.

2. Ενσωμάτωση του GTFT στο DSR

Ο αλγόριθμος GTFT που περιγράφηκε στην προηγούμενη ενότητα, υλοποιήθηκε εύκολα με κάποιες προσθήκες και αλλαγές στον κώδικα του DSR. Το κύριο ποσοστό του νέου κώδικα προστέθηκε ακριβώς πριν την κλήση της συνάρτησης, η οποία είναι υπεύθυνη για την αποστολή των πακέτων δεδομένων (dsr_transmit_data). Στο σημείο αυτό είναι γνωστός ο επιθυμητός προορισμός των πακέτων δεδομένων που περιμένουν να αποσταλούν καθώς και το πλήρες δρομολόγιο προς αυτόν, το οποίο είτε έχει προκύψει από τη διαδικασία αναζήτησης διαδρομής είτε έχει προσδιοριστεί από το μηχανισμό συντήρησης (υπάρχει ήδη γνωστό δρομολόγιο αποθηκευμένο στην αντίστοιχη cache).

2.1. Οι νέες συναρτήσεις

Συγκεκριμένα, δημιουργήθηκαν και ενσωματώθηκαν στον κώδικα του DSR έξι νέες συναρτήσεις. Η λειτουργία κάθε μιας από αυτές περιγράφεται στη συνέχεια.

start_forward_request: Η συνάρτηση αυτή είναι υπεύθυνη για την έναρξη της διαδικασίας ελέγχου για την αποδοχή ή απόρριψη μίας αίτησης για προώθηση. Καλείται από τον κόμβο – πηγή που επιθυμεί να αποστείλει δεδομένα, κατασκευάζει το πακέτο αίτησης για προώθηση και το αποστέλλει στον πρώτο ενδιάμεσο σταθμό. Κατά συνέπεια, το μόνο όρισμα που δέχεται είναι η διεύθυνση προορισμού, δηλαδή η διεύθυνση του πρώτου ενδιάμεσου σταθμού.

void start_forward_request (int destination_dsr_address)

handle_forward_request: Η συνάρτηση αυτή καλείται κάθε φορά που ένας κόμβος λαμβάνει ένα πακέτο τύπου «αίτησης για

προώθηση». Ο κόμβος αποφασίζει με βάση τη συνθήκη (5) αν θα αποδεχτεί ή όχι την αίτηση. Αν ο κόμβος τυχαίνει να είναι ο τελευταίος ενδιάμεσος σταθμός αναλαμβάνει να ξεκινήσει τη διαδικασία απάντησης (η διαδικασία είναι ίδια είτε πρόκειται για θετική είτε για αρνητική απάντηση). Αν ο κόμβος δεν είναι ο τελευταίος ενδιάμεσος σταθμός προωθεί το πακέτο αίτησης στον επόμενο σταθμό σε περίπτωση που ο ίδιος αποδέχτηκε την αίτηση, διαφορετικά ξεκινά τη διαδικασία αρνητικής απάντησης. Το μόνο όρισμα αυτής της συνάρτησης είναι ένας δείκτης στο πακέτο αίτησης για προώθηση.

void handle_forward_request (Packet fw_req_pk_ptr)*

forward_forward_request: Η συνάρτηση αυτή καλείται από κάποιο κόμβο, ο οποίος έχει ήδη λάβει και επεξεργαστεί ένα πακέτο αίτησης και τώρα πρέπει να το προωθήσει στον επόμενο ενδιάμεσο σταθμό. Κατά συνέπεια, το μόνο όρισμα που απαιτείται είναι ο δείκτης στο αντίστοιχο πακέτο αίτησης.

void forward_forward_request (Packet fw_req_pk_ptr)*

start_forward_reply: Η συνάρτηση αυτή καλείται από τον αρμόδιο κόμβο κάθε φορά που μία απάντηση σε αίτηση για προώθηση είναι έτοιμη. Αναλαμβάνει την κατασκευή του αντίστοιχου πακέτου απάντησης και ενεργοποιεί τη διαδικασία απάντησης αποστέλλοντας αυτό το πακέτο στον πρώτο ενδιάμεσο σταθμό ακολουθώντας ακριβώς το ανάποδο δρομολόγιο (από τον προορισμό της αρχικής αίτησης προς την πηγή αυτής). Τα ορίσματα που απαιτούνται σε αυτή την περίπτωση είναι ο δείκτης στο αντίστοιχο πακέτο αίτησης και η τιμή της απάντησης: 1 για θετική απάντηση και 0 για αρνητική.

void start_forward_reply (Packet fw_req_pk_ptr, int answer)*

handle_forward_reply: Η συνάρτηση αυτή καλείται κάθε φορά που ένας κόμβος λαμβάνει ένα πακέτο τύπου «απάντησης σε αίτηση για προώθηση». Αν ο κόμβος αυτός είναι η πηγή της αίτησης, τότε αν η απάντηση είναι θετική καλείται η συνάρτηση που είναι υπεύθυνη για την αποστολή των δεδομένων, διαφορετικά τα δεδομένα δεν αποστέλλονται και το περιστατικό καταγράφεται ως αρνητική απάντηση. Αν ο κόμβος είναι κάποιος ενδιάμεσος

σταθμός, τότε το πακέτο απλώς προωθείται στον επόμενο κόμβο. Το μόνο όρισμα που δέχεται αυτή η συνάρτηση είναι και πάλι ο δείκτης στο αντίστοιχο πακέτο απάντησης.
void handle_forward_reply (Packet fw_rep_pk_ptr)*

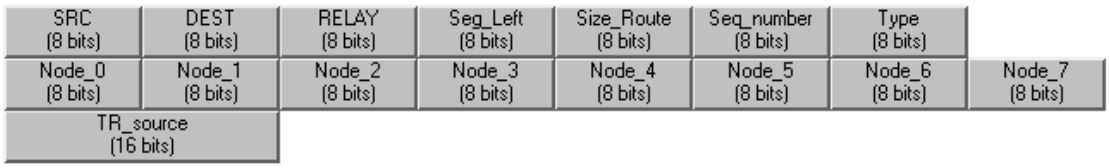
forward_forward_reply: Η συνάρτηση αυτή καλείται από κάποιο κόμβο, ο οποίος έχει ήδη λάβει και επεξεργαστεί ένα πακέτο απάντησης σε αίτηση και τώρα πρέπει να το προωθήσει στον επόμενο ενδιάμεσο σταθμό. Κατά συνέπεια, το μόνο όρισμα που απαιτείται είναι ο δείκτης στο αντίστοιχο πακέτο αίτησης.
void forward_forward_request (Packet fw_req_pk_ptr)*

2.2. Οι νέοι τύποι πακέτων

Στην ενότητα αυτή περιγράφονται οι δύο νέοι τύποι πακέτων που κατασκευάστηκαν προκειμένου να ενσωματωθεί ο GTFT στον κώδικα του DSR. Για λόγους συμβατότητας και ομοιομορφίας διατηρήθηκε ανάλογο μοτίβο κατασκευής με το ήδη υπάρχον, όπως προκύπτει από το πρωτόκολλο του DSR. Η μορφή των νέων πακέτων μοιάζει με εκείνη των πακέτων που περιγράφηκαν στην ενότητα 3.4. του δεύτερου μέρους.

2.2.1. Το πακέτο «αίτησης για προώθηση»

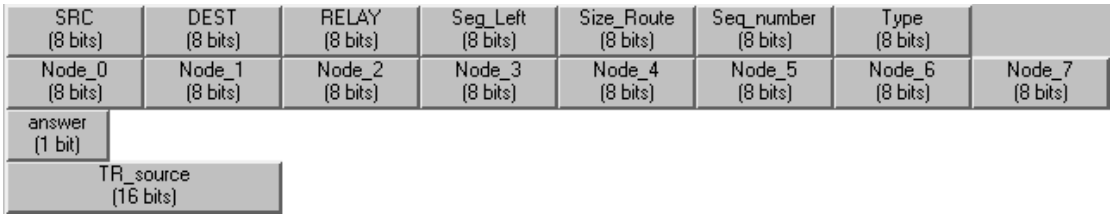
Το πακέτο αίτησης για προώθηση προκύπτει άμεσα και με ελάχιστες αλλαγές από το πακέτο που επιστρέφεται σαν αποτέλεσμα της συνάρτησης που υλοποιεί την αναζήτηση του δρομολογίου στον αλγόριθμο DSR. Κατά συνέπεια, το νέο πακέτο διατηρεί αναλλοίωτη την πληροφορία που σχετίζεται με ολόκληρο το μονοπάτι προς τον τελικό προορισμό, ενώ αλλάζει ο τύπος του πακέτου και προστίθεται ένα επιπλέον πεδίο, το οποίο θα διατηρεί έναν αύξοντα αριθμό αναγνωριστικού χαρακτήρα. Ο νέος τύπος αυτού του πακέτου καλείται “Forward_Request_Packet_Type” και η δομή του παρουσιάζεται στο σχήμα 8.



Σχήμα 8: Forward_Request_Packet_Type

2.2.2. Το πακέτο «απάντησης σε αίτηση για προώθηση»

Το πακέτο αυτό προκύπτει άμεσα από το αντίστοιχο «πακέτο αίτησης για προώθηση», αρκεί να αντιστραφεί το δρομολόγιο, αφού πλέον η κατεύθυνση είναι από τον προορισμό της αίτησης προς την πηγή αυτής. Μία ακόμα απαραίτητη αλλαγή είναι η προσθήκη ενός επιπλέον πεδίου, το οποίο θα περιέχει την απάντηση στην αίτηση για προώθηση. Προφανώς, το πεδίο με τον αύξοντα αριθμό αναγνωριστικού χαρακτήρα θα εξακολουθεί να υπάρχει και θα έχει την ίδια τιμή με το πακέτο της αντίστοιχης αίτησης, έτσι ώστε να αποφεύγονται συγχύσεις. Ο νέος τύπος αυτού του πακέτου καλείται "Forward_Reply_Packet_Type" και η δομή του παρουσιάζεται στο σχήμα 9.



Σχήμα 9: Forward_Reply_Packet_Type

Στο Παράρτημα Α παρατίθενται πίνακες οι οποίοι αναφέρουν το μέγεθος και τη λειτουργικότητα κάθε πεδίου των πακέτων. Επίσης, παρατίθεται συγκεντρωτικός πίνακας με τη λειτουργία των έξι νέων συναρτήσεων που περιγράφηκαν στην ενότητα 2.1.

2.3. Οι μετρητές requests_handled_for_me και requests_handled_for_others

Όπως έχει ήδη αναφερθεί (ενότητα Γ1), κάθε κόμβος διαθέτει δύο μετρητές: forward_requests_handled_for_me και forward_requests_handled_for_others. Όπως μαρτυρά και το όνομά τους ο πρώτος μετρητής απαριθμεί τις αιτήσεις που

εξυπηρετήθηκαν συνολικά για χάρη ενός κόμβου, ενώ ο δεύτερος μετρητής απαριθμεί το σύνολο των αιτήσεων που ο κόμβος αυτός εξυπηρέτησε για λογαριασμό άλλων ξένων ροών κίνησης.

Είναι ιδιαίτερα σημαντικό να παρατηρηθεί πως στο προγραμματιστικό μέρος, η αύξηση αυτών των μετρητών γίνεται κατά τη διαδικασία της απάντησης και όχι κατά τη διαδικασία της αίτησης για προώθηση. Με τον τρόπο αυτό αποφεύγεται να θεωρήσει λαθεμένα ένας ενδιάμεσος κόμβος ότι θα προωθήσει κίνηση δεδομένων «ξένης» ροής αν κάποιος άλλος επόμενος σταθμός στη συνέχεια αρνηθεί. Αντίθετα, κατά τη διαδικασία της απάντησης υπάρχει βεβαιότητα για τη συνολική τύχη της αίτησης.

Δ' ΜΕΡΟΣ : ΝΕΕΣ ΑΛΓΟΡΙΘΜΙΚΕΣ ΣΤΡΑΤΗΓΙΚΕΣ

Στην ενότητα αυτή παρουσιάζονται κάποιες νέες αλγοριθμικές τεχνικές με απώτερο σκοπό τη δημιουργία και ενδυνάμωση κινήτρων για προώθηση «ξένων» ροών κίνησης από όλους τους κόμβους ενός ασύρματου δικτύου αυθαίρετης δομής. Συγκεκριμένα, παρουσιάζονται μία παραλλαγή του GTFT και ένας νέος αλγόριθμος που στηρίζεται στην προσπάθεια εξισορρόπησης των ρυθμών των ποσών βοήθειας που λαμβάνονται και δίδονται από κάθε κόμβο.

1. Το πρόβλημα του απλού GTFT

Όπως θα δούμε αναλυτικά και στο πέμπτο μέρος της εργασίας, ο αλγόριθμος GTFT δουλεύει καλά και δίνει κίνητρο για συνεργασία στους κόμβους ενός δικτύου αυθαίρετης δομής. Κάθε κόμβος αποδέχεται μία αίτηση για προώθηση κίνησης «ξένης» ροής αρκεί και ο ίδιος να έχει βοηθηθεί αναλόγως στο παρελθόν από το σύνολο του δικτύου. Το κίνητρο για όλους τους κόμβους είναι εμφανές, οπότε ο κάθε κόμβος βοηθάει προκειμένου και ο ίδιος να βοηθηθεί αργότερα. Με τον τρόπο αυτό το κλίμα συνεργασίας επιτυγχάνεται πολύ γρήγορα και το throughput του δικτύου

αποκτά υψηλή τιμή. Επίσης, τα ποσοστά βοήθειας που δίδονται και λαμβάνονται από κάθε κόμβο συγκλίνουν μεταξύ τους και μάλιστα σε υψηλή τιμή που πλησιάζει το 100%. Οι αρνητικές απαντήσεις (απορρίψεις αιτήσεων) είναι μηδενικές ή ελάχιστες και παρατηρούνται μόνο κατά τα πρώτα δευτερόλεπτα αμέσως μετά την εκκίνηση του δικτύου.

Ο αλγόριθμος, όμως, παύει να δουλεύει καλά σε περιπτώσεις όπου στο δίκτυο εμφανίζονται εγωιστικές συμπεριφορές. Όπως και οι ίδιοι οι εμπνευστές του GTFT έχουν παρατηρήσει στην εργασία τους, η ύπαρξη ενός και μόνο κόμβου εγωιστή είναι δυνατό να έχει δραματική επίπτωση τόσο στο συνολικό throughput του δικτύου, όσο και στα επιμέρους σημεία σύγκλισης των ποσοστών ληφθείσας και δοθείσας βοήθειας κάθε κόμβου. Και το φαινόμενο αυτό είναι λογικό, αφού η συνεχής απόρριψη αιτήσεων από τον κόμβο εγωιστή αφαιρεί το κίνητρο για συνεργασία και από τους υπόλοιπους κόμβους: αφού ο ίδιος δεν εξυπηρετείται χάνει πλέον το κίνητρο να εξυπηρετήσει και αυτός. Το αποτέλεσμα είναι άμεσο και γίνεται όλο και πιο οδυνηρό για το δίκτυο καθώς το πλήθος των εγωιστών κόμβων αυξάνει σε μία συγκεκριμένη τοπολογία.

2.1. GTFT με χρήση της ιδιότητας της γειτονίας (N-GTFT)

Ο αλγόριθμος που περιγράφεται στην παρούσα ενότητα αποτελεί μία απλή επέκταση του αλγορίθμου GTFT. Σε αυτή την περίπτωση λαμβάνεται κατά νου η ιδιότητα που έχουν οι κόμβοι σε ένα δίκτυο να γειτονεύουν με κάποιους άλλους κόμβους. Συγκεκριμένα, σε ένα δίκτυο αυθαίρετης δομής κάθε κόμβος γειτονεύει με τόσους κόμβους όσοι ακριβώς ανήκουν στην περιοχή εμβέλειάς του, άρα είναι σε θέση να επικοινωνεί άμεσα με αυτούς, χωρίς την ανάγκη ύπαρξης διαμεσολαβητών.

Στο συγκεκριμένο αλγόριθμο η βασική ιδέα στηρίζεται στη διατήρηση ξεχωριστής πληροφορίας από κάθε κόμβο σε σχέση με τους γείτονές του. Η ιδέα, μάλιστα, προέκυψε μετά από παρατήρηση βάσει της οποίας σε ένα σενάριο με σταθερούς κόμβους, υπάρχει πολύ μεγάλη πιθανότητα ένας συγκεκριμένος

κόμβος να αποστέλλει τις πρώτες αιτήσεις του για προώθηση στους ίδιους γείτονες με μεγάλη συχνότητα. Σε αυτή την περίπτωση, η συμπεριφορά των γειτόνων αποκτά ιδιαίτερη σημασία.

Η υλοποίηση αυτής της έκδοσης του αλγορίθμου υπαγορεύει τη διατήρηση από κάθε κόμβο τόσων τετράδων από μετρητές όσοι είναι και οι γείτονές του. Κατά συνέπεια, ένας κόμβος A που έχει n γείτονες N_1 ... N_n, διατηρεί n τετράδες μετρητών, ενώ αν ισχύει $1 \le k \le n$, η τετράδα μετρητών για τον κόμβο k έχει ως εξής:

Requests handled for N_k, requests made to me by N_k και

(6)

Requests handled for me by N_k, requests made by me to N_k

Η διαδικασία απόφασης αποδοχής ή απόρριψης μίας αίτησης για προώθηση δε διαφέρει σημαντικά από την αντίστοιχη διαδικασία του βασικού αλγορίθμου. Αμέσως μόλις ο κόμβος A λάβει ένα πακέτο αίτησης για προώθηση, εξετάζει από ποιο γείτονα έχει προέλθει αυτό. Ας υποθέσουμε ότι το πακέτο έχει προέλθει από τον κόμβο k. Στην περίπτωση αυτή τα ποσοστά που προκύπτουν από τους μετρητές (6) κάνοντας χρήση των τύπων (3) και (4) έχουν ως εξής:

$$amount\ of\ help\ i\ gave\ to\ N_k \;=\; \frac{requests\ handled\ for\ N_k}{requests\ made\ to\ me\ by\ N_k}$$

(7)

και

$$amount\ of\ help\ i\ received\ by\ N_k \;=\; \frac{requests\ handled\ for\ me\ by\ N}{requests\ made\ by\ me\ to\ N_k}$$

(8)

Από τους τύπους (7) και (8) η συνθήκη (5) μετασχηματίζεται ως εξής:

$$amount\ of\ help\ i\ received\ by\ N_k \ + \ gen \ > \ amount\ of\ help\ i\ gave\ to\ N_k \quad (9)$$

Τελικά, η απόφαση για την αποδοχή ή απόρριψη της αίτησης λαμβάνεται με βάση τα στοιχεία που υπάρχουν σχετικά με τη συμπεριφορά που οι κόμβοι A και k είχαν αναπτύξει μεταξύ τους στο παρελθόν.

Αυτή η παραλλαγή του βασικού αλγορίθμου που χρησιμοποιεί την ιδιότητα της γειτονίας φαίνεται πως καταφέρνει να ανιχνεύει και να απομονώνει αρκετά γρήγορα κακόβουλες συμπεριφορές, όπως αυτές που προκύπτουν από κάποιο κόμβο ο οποίος συνεχώς αρνείται την προώθηση κίνησης «ξένων» ροών δεδομένων. Με το συγκεκριμένο αλγόριθμο εντοπίζεται όχι απλά η ύπαρξη μίας μη συνεργατικής συμπεριφοράς, αλλά και η προέλευσή αυτής. Με τον τρόπο αυτό το δίκτυο είναι σε θέση να τιμωρήσει το συγκεκριμένο κόμβο υιοθετώντας ανάλογη αρνητική συμπεριφορά απέναντι σε αυτόν και μόνο, προλαμβάνοντας έτσι δραματικές επιπτώσεις στο συνολικό όγκο δεδομένων που διέρχεται από το δίκτυο.

Η μείωση στο συνολικό throughput που παρατηρείται λόγω της ύπαρξης κακόβουλων συμπεριφορών κάνοντας χρήση αυτής της παραλλαγής του αλγορίθμου αναμένεται να είναι μικρότερη από αυτή που παρατηρείται κάνοντας χρήση του βασικού αλγορίθμου (απλός GTFT), ενώ αρχίζει και εδώ να αυξάνει καθώς αυξάνει το πλήθος και η πυκνότητα των εγωιστών.

Στο σημείο αυτό αξίζει να σημειωθεί πως όταν ένας κόμβος λαμβάνει ένα «πακέτο αίτησης για προώθηση», ως πηγή του πακέτου αυτού θεωρείται ο τελευταίος γείτονας που έστειλε το πακέτο και όχι η πραγματική αρχική πηγή της αίτησης (που είναι είτε κάποιος άλλος γειτονικός κόμβος είτε όχι). Βέβαια, σε παράλληλο χρόνο και κατά την εκτέλεση των πειραμάτων διατηρούνται και ανανεώνονται κανονικά όλοι οι μετρητές που παρουσιάστηκαν κατά την περιγραφή του βασικού αλγορίθμου, έτσι ώστε να συγκεντρώνονται και να υπολογίζονται στατιστικά στοιχεία που αφορούν τη συνολική σφαιρική εικόνα του δικτύου.

3. Αλγόριθμος εξίσωσης ρυθμών με τη χρήση κουβά (Bucket algorithm)

Ο δεύτερος αλγόριθμος που προτείνεται στην παρούσα εργασία ακολουθεί διαφορετική φιλοσοφία σε σχέση με τις στρατηγικές που παρουσιάστηκαν στις προηγούμενες ενότητες. Σε αυτή την περίπτωση η βασική ιδέα είναι η κατοχή ενός κουβά Β από κάθε κόμβο του δικτύου. Ο σκοπός αυτής της ιδέας είναι η συγκέντρωση ενός αριθμού από κουπόνια μέσα στον κουβά ενός κόμβου αμέσως μετά τη συνεχόμενη εξυπηρέτηση που αυτός έχει δεχτεί από το δίκτυο. Με τον τρόπο αυτό, ο ίδιος μπορεί άμεσα να εξυπηρετήσει μία σειρά εκρηκτικών αιτήσεων (burst), χωρίς να χρειάζεται να περιμένει κάποιο συγκεκριμένο χρονικό διάστημα, όπως θα συνέβαινε στην περίπτωση του αλγορίθμου που βασίζεται στην εξίσωση των ποσοστών δοθείσας και ληφθείσας βοήθειας.

Στην παρούσα εργασία προτείνεται μία υλοποίηση βάσει της οποίας η ανανέωση της τιμής του κουβά κάθε κόμβου γίνεται ακολουθώντας τη σχέση (10):

$$B \mathrel{+}= \left(HR_{i-1} - HG_{i-1}\right)\left(t_i - t_{i-1}\right)$$

(10)

όπου HR_{i-1} και HG_{i-1} είναι τα ποσοστά ληφθείσας και δοθείσας βοήθειας αντίστοιχα, μέχρι τη χρονική στιγμή t_{i-1}, όπως αυτά έχουν καταμετρηθεί από ένα συγκεκριμένο κόμβο.

Η διαδικασία ανανέωσης της τιμής του κουβά λαμβάνει χώρα αμέσως μετά την ολοκλήρωση ενός συμβάντος, είτε πρόκειται για βοήθεια που δόθηκε είτε για βοήθεια που λήφθηκε. Κατά συνέπεια, οι χρονικές στιγμές t_{i-1} και t_i αντιπροσωπεύουν δύο διαδοχικά συμβάντα. Όπως προκύπτει από τη σχέση (10) ο κουβάς κάθε κόμβου μοιάζει να γεμίζει με ρυθμό που ισούται με το ρυθμό της εισερχόμενης βοήθειας, ενώ αδειάζει με ρυθμό ίσο με εκείνον της εξερχόμενης βοήθειας.

Είναι προφανές ότι η μόνη συνθήκη που πρέπει να ισχύει προκειμένου ένας κόμβος να αποδεχτεί μία αίτηση για προώθηση

«ξένης» ροής κίνησης είναι η κατοχή ενός μη άδειου κουβά. Η συνθήκη που είναι υπεύθυνη για την απόφαση σχετικά με την αίτηση προώθησης παραμένει απλή:

Αν ο κουβάς δεν είναι άδειος
Τότε, η αίτηση γίνεται αποδεκτή
Διαφορετικά,
Η αίτηση απορρίπτεται

5. N-GTFT με χρήση διαφορετικών βαρών (W-GTFT)

Μία ακόμα παραλλαγή, η οποία είναι δυνατό να προκύψει και από τους δύο ήδη προταθέντες αλγορίθμους (N-GTFT και Bucket) είναι η χρήση διαφορετικών βαρών, τα οποία θα λαμβάνονται υπ' όψιν κατά τον εκάστοτε έλεγχο συνθήκης που είναι υπεύθυνος για την αποδοχή ή την απόρριψη μιας αίτησης. Η απόφαση για την τιμή που έχει το βάρος κάθε κόμβου βασίζεται στη θέση που έχει κάθε κόμβος σε μια συγκεκριμένη τοπολογία.

Για παράδειγμα, υπάρχει το ενδεχόμενο ένας κόμβος Α να είναι απαραίτητο να διαδραματίσει το ρόλο του ενδιάμεσου σταθμού σε «ξένες» ροές κίνησης πολύ περισσότερες φορές σε σχέση με έναν άλλο κόμβο Β. Η περίπτωση αυτή είναι δυνατό να συμβεί σε μία τοπολογία όπου ένα σημείο πρόσβασης (access point) είναι τοποθετημένο στο κέντρο, ενώ το ίδιο περιβάλλεται από ένα σύνολο κόμβων, οι οποίοι είναι παρατεταγμένοι σε μία σειρά από ομόκεντρους κύκλους. Υποθέτοντας ότι οι όλοι οι κόμβοι επιθυμούν την αποστολή κίνησης προς το κεντρικό σημείο πρόσβασης, παρατηρείται ότι οι εκείνοι οι κόμβοι που είναι τοποθετημένοι πιο κοντά στο κέντρο (εσωτερική ζώνη) αναγκάζονται να δρομολογούν πολύ περισσότερη «ξένη» κίνηση σε σχέση με τους πιο απομακρυσμένους. Οι αιτήσεις που καλούνται να εξυπηρετήσουν αυτοί οι κόμβοι είναι σίγουρα πολύ περισσότερες από τον αριθμό αιτήσεων για προώθηση που οι ίδιοι διοχετεύουν προς το δίκτυο. Ειδικά για τους κόμβους της εσωτερικότερης ζώνης προκύπτει ότι αυτοί πρέπει να εξυπηρετούν αιτήσεις από όλες τις εξωτερικές ζώνες, κατά συνέπεια πρέπει

να αποδέχονται αιτήσεις χρησιμοποιώντας διαφορετικό και συγκεκριμένα μεγαλύτερο βάρος.

Εύκολα παρατηρεί κανείς πως αν δεν ακολουθηθεί η προσέγγιση με τα βάρη οι κόμβοι της εσωτερικής ζώνης θα παράγουν ένα μεγάλο αριθμό από αρνητικές απαντήσεις, εφόσον θα θεωρούν, και μάλιστα δικαίως, ότι οι ίδιοι δεν έχουν βοηθηθεί αναλόγως στο παρελθόν.

Τελικά, προκύπτει ότι τα βάρη που χρησιμοποιούνται από τους κόμβους των εσωτερικών ζωνών – καθώς προχωρούμε προς το κέντρο της τοπολογίας – θα πρέπει να αυξάνονται αναλόγως καθώς μεγαλώνει και ο αριθμός των κόμβων που βρίσκονται συνολικά τοποθετημένοι στις πιο εξωτερικές ζώνες. Παρατηρείται, λοιπόν, ανάγκη για αύξηση της τιμής του βάρους που χρησιμοποιούν οι κόμβοι μίας συγκεκριμένης ζώνης, καθώς μεγαλώνει η πυκνότητα των κόμβων στις πιο εξωτερικές ζώνες.

Οι προσομοιώσεις που έγιναν βάσει της συγκεκριμένης παραλλαγής των αλγορίθμων κατέληξαν στον καθορισμό μίας σχέσης μεταξύ των βαρών και της πυκνότητας των κόμβων που περιβάλλουν μία ζώνη. Πιο συγκεκριμένα, η σχέση που προτείνεται δίνεται από τον ακόλουθο τύπο:

$$\frac{W_A}{W_B} = \frac{N_{A+B}}{N_B} \qquad (11)$$

όπου W_A και W_B είναι τα βάρη που χρησιμοποιούνται από τους κόμβους μιας εσωτερικής ζώνης A και μιας εξωτερικής ζώνης B αντίστοιχα, ενώ N_{A+B} και N_B είναι ο αριθμός των κόμβων που είναι τοποθετημένοι σε κάθε ζώνη. Προφανώς, πάντα υποθέτουμε ότι οι κόμβοι της πιο εξωτερικής ζώνης χρησιμοποιούν βάρος με τιμή 1 (ένα).

Ε' ΜΕΡΟΣ : ΑΠΟΤΕΛΕΣΜΑΤΑ ΠΡΟΣΟΜΟΙΩΣΕΩΝ

1. Εισαγωγή

Στις ενότητες που ακολουθούν παρουσιάζονται τα πιο βασικά πειράματα που εκτελέστηκαν κατά τη διάρκεια αυτής της εργασίας και παρατίθενται τα αντίστοιχα αποτελέσματα υπό μορφή γραφικών παραστάσεων και πινάκων πλαισιωμένα από διάφορα σχόλια και συμπεράσματα.

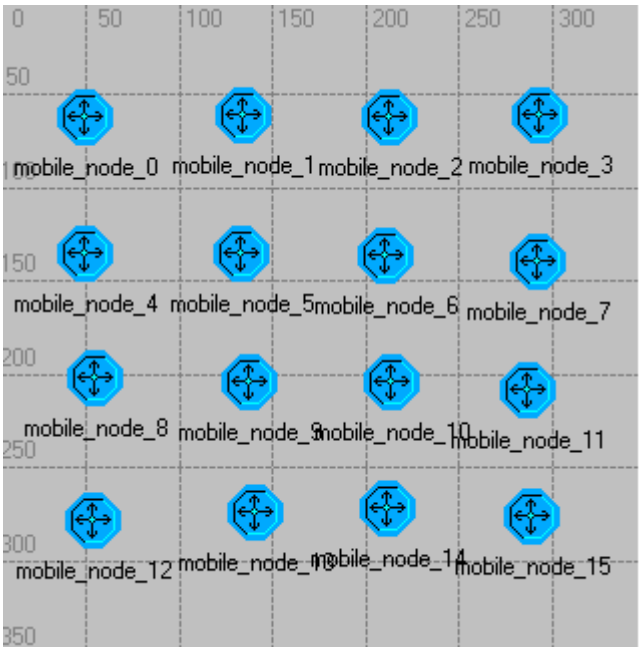
Κάθε υπό-ενότητα ασχολείται με την παρουσίαση ενός πειραματικού σεναρίου και των αντίστοιχων αποτελεσμάτων που προέκυψαν κατά την προσομοίωση. Στο τέλος της ενότητας παρατίθεται ένας συγκεντρωτικός πίνακας των πειραματικών αποτελεσμάτων. Βασική μετρική όλων των πειραμάτων τέθηκε ο συνολικός όγκος πακέτων δεδομένων που καταφέρνει να διοχετευτεί μέσα από το δίκτυο στη μονάδα του χρόνου.

2. Σενάρια Προσομοιώσεων

Όλα τα πειραματικά σενάρια προσομοιώθηκαν κάνοντας χρήση του εργαλείου Opnet. Σε κάθε περίπτωση ακίνητοι κόμβοι τοποθετούνταν σε συγκεκριμένα σημεία διάσπαρτα σε ένα γεωγραφικό χώρο συγκεκριμένων και προκαθορισμένων διαστάσεων.

2.1. Σενάριο 1: GTFT χωρίς εγωιστές κόμβους

Η προσομοίωση αυτή βασίζεται σε ένα σενάριο, στο οποίο 16 κόμβοι βρίσκονται τυχαία τοποθετημένοι σε μία γεωγραφική περιοχή διαστάσεων 350 x 350 m^2. Όλοι οι κόμβοι είναι σταθεροί και συμμορφώνονται με τον αλγόριθμο DSR. Όσον αφορά τα χαρακτηριστικά της κίνησης χρησιμοποιείται η εκθετική κατανομή, ενώ το χρονικό διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών αποστολών πακέτων από μία πηγή τίθεται στην τιμή 0.25 sec. Το μέγεθος των πακέτων είναι 512 bits, ενώ στην παράμετρο gen αποδίδεται η μικρή θετική τιμή 0.01. Επίσης, στην απόσταση εμβέλειας (transmission range) δόθηκε η τιμή 100 m.



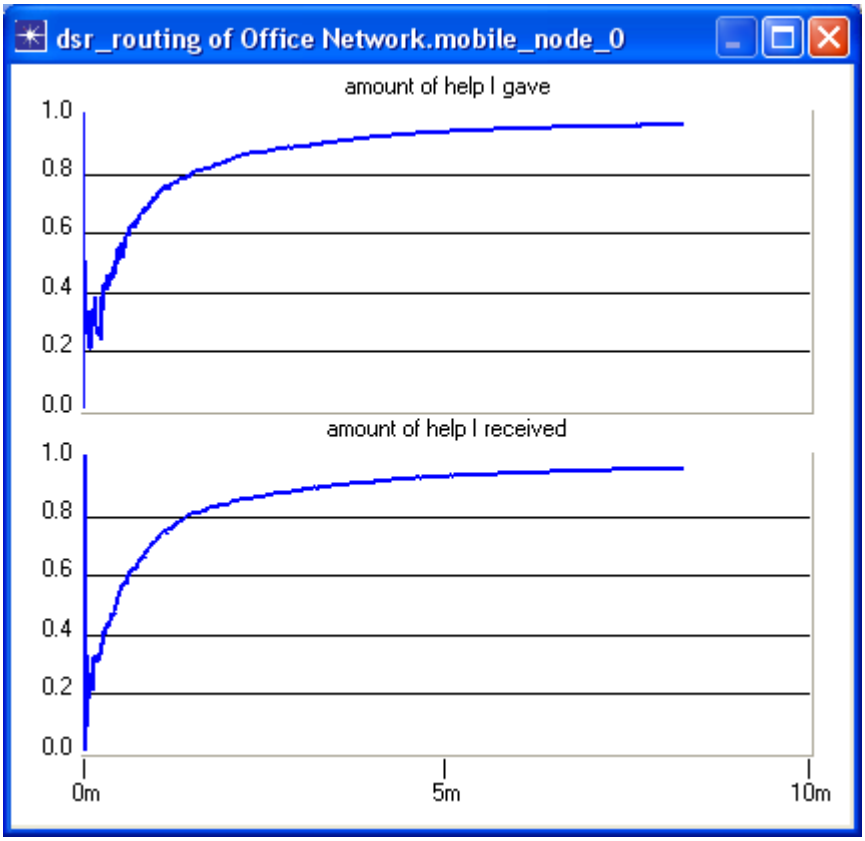
Σχήμα 10: Τοπολογία 16 κόμβων

Τα αποτελέσματα που προέκυψαν από ένα πείραμα των 10 λεπτών δείχνουν πως το ποσοστό βοήθειας που κάθε κόμβος λαμβάνει εξισορροπείται με το αντίστοιχο ποσοστό βοήθειας που δίνει πολύ κοντά στην τιμή 1. Η επίτευξη μάλιστα αυτής της εξισορρόπησης γίνεται ικανοποιητικά γρήγορα σε σχέση με τη χρονική στιγμή έναρξης της λειτουργίας του δικτύου. Στο συγκεκριμένο πείραμα ο συνολικός όγκος κίνησης που διέρχεται από το δίκτυο κυμαίνεται στα 64 πακέτα \ δευτερόλεπτο, ενώ ο κάθε επιμέρους κόμβος απολαμβάνει ισόποσο throughput ίσο με περίπου το 1/16 του συνολικού, δηλαδή γύρω στα 4 πακέτα \ δευτερόλεπτο.

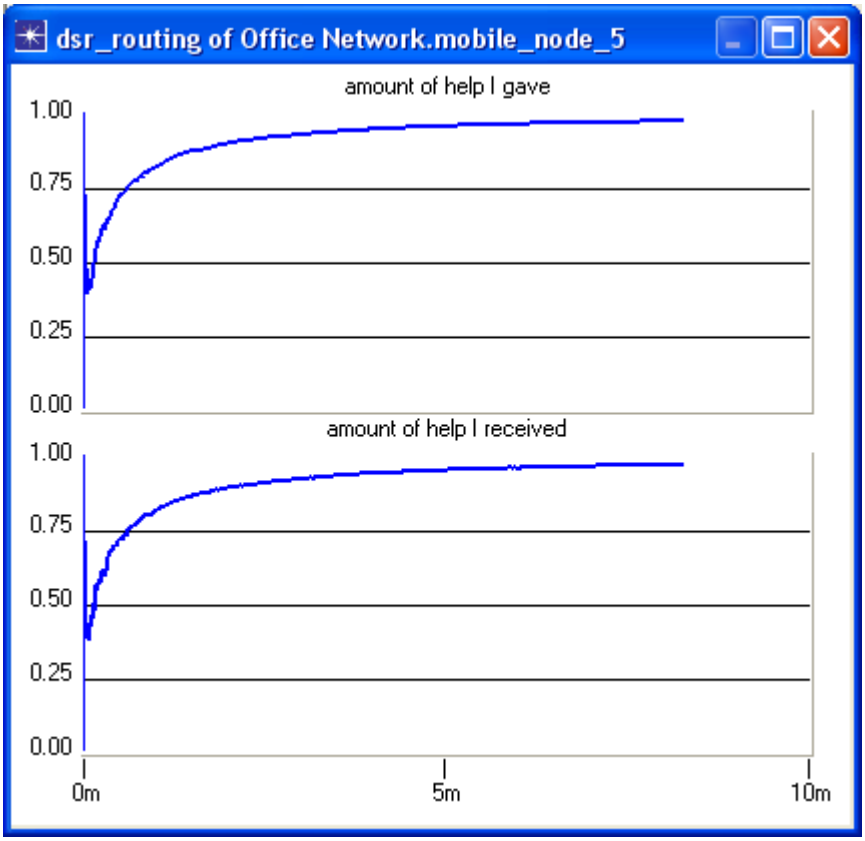
Στο σημείο αυτό αξίζει να παρατηρήσουμε πως οι κόμβοι που βρίσκονται τοποθετημένοι κοντά στο κέντρο της τοπολογίας του δικτύου, φθάνουν το σημείο εξισορρόπησης λίγο πιο γρήγορα από τους κόμβους που βρίσκονται σε περιφερικά σημεία. Το φαινόμενο αυτό οφείλεται στο γεγονός βάσει του οποίου οι κεντρικοί κόμβοι καλούνται να αντιμετωπίσουν περισσότερη κίνηση, καθώς δέχονται περισσότερες αιτήσεις λόγω της κομβικής τους θέσης.

Στα σχήματα 11 και 12 παρουσιάζεται η γραφική αναπαράσταση των ποσοστών βοήθειας που δίνονται και λαμβάνονται από δύο κόμβους: έναν τοποθετημένο σε σημείο της περιφέρειας

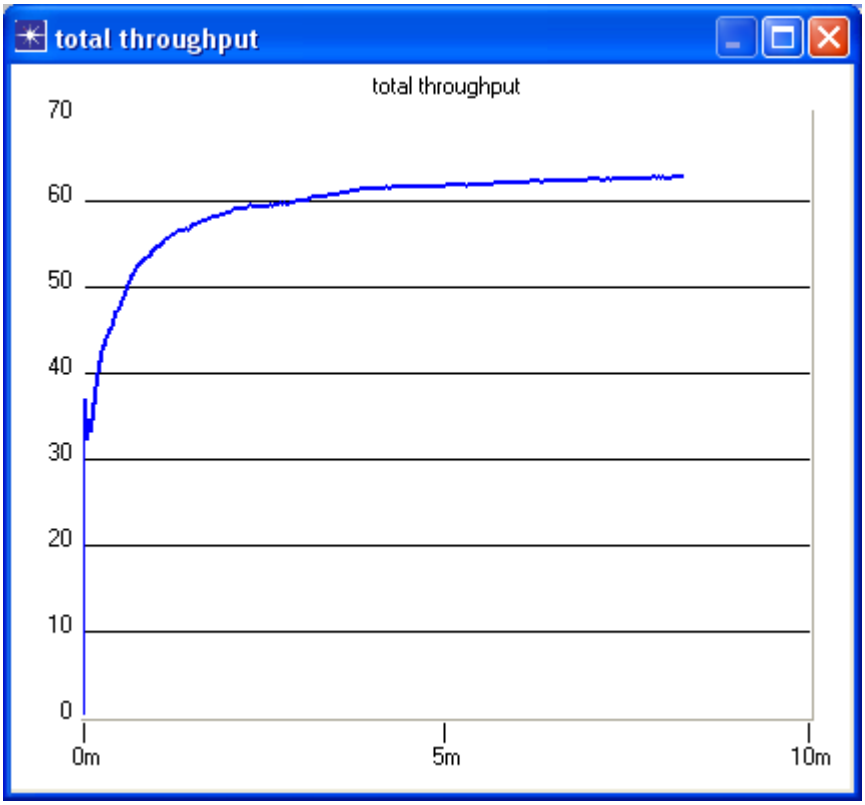
(mobile_node_0) και έναν τοποθετημένο πιο κεντρικά (mobile_node_5).



Σχήμα 11: Σύγκλιση ποσοστών βοήθειας κόμβου 0 σε ομοιόμορφη κίνηση χωρίς εγωιστές



Σχήμα12: Σύγκλιση ποσοστών βοήθειας κόμβου 5 σε ομοιόμορφη κίνηση χωρίς εγωιστές



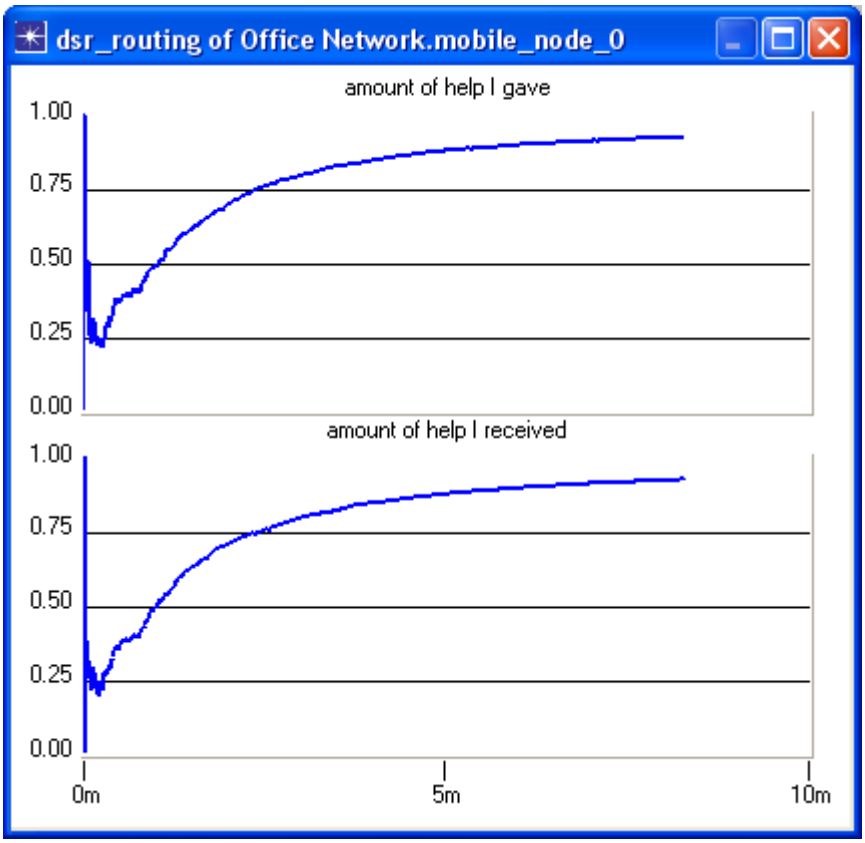
Σχήμα 13: Συνολικό data throughput σε ομοιόμορφη κίνηση χωρίς εγωιστές

2.2. Σενάριο 2: GTFT χωρίς εγωιστές κόμβους υπό συνθήκες υψηλού δικτυακού φόρτου

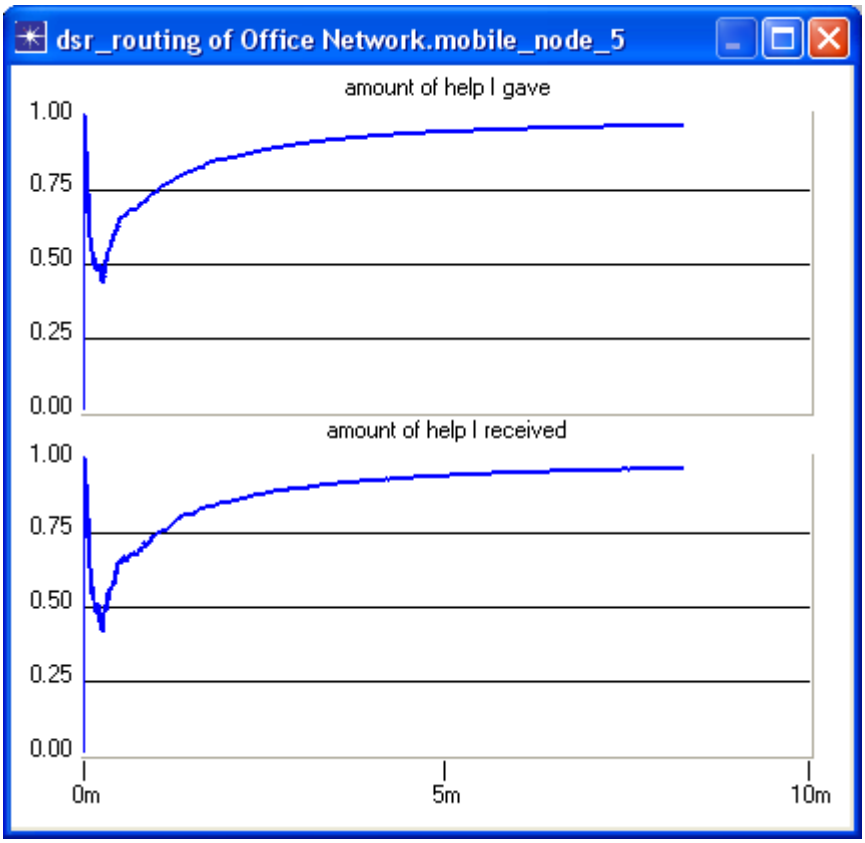
Το σενάριο αυτό προσεγγίστηκε με δύο διαφορετικούς τρόπους. Στην πρώτη περίπτωση διπλασιάστηκε το μέγεθος του πακέτου, ενώ στη δεύτερη περίπτωση υποδιπλασιάστηκε η τιμή της παραμέτρου που καθορίζει το μεσοδιάστημα ανάμεσα σε δύο διαδοχικές αποστολές πακέτων. Και στις δύο περιπτώσεις θεωρείται ότι αυξάνεται ο δικτυακός φόρτος και εξετάζεται η αντοχή του αλγορίθμου στις νέες συνθήκες. Οι υπόλοιπες παράμετροι διατηρούν τις τιμές του σεναρίου 1.

2.2.1. Σενάριο 2.1.: Διπλασιασμός μεγέθους πακέτου

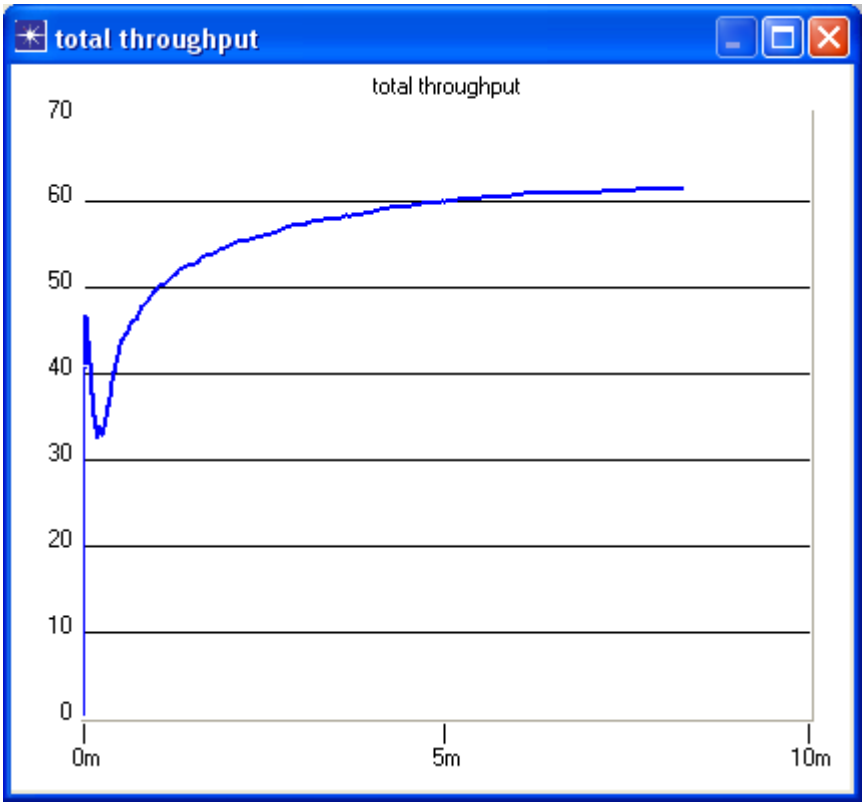
Στο πείραμα αυτό τα πακέτα δεδομένων που διακινούνται στο δίκτυο έχουν πλέον μέγεθος 1024 bits. Τα αποτελέσματα που προκύπτουν δείχνουν πως η μόνη επίπτωση που υπάρχει στον αλγόριθμο είναι μία μικρή χρονική καθυστέρηση στην προσέγγιση του σημείου σύγκλισης των ποσοστών ληφθείσας και δοθείσας βοήθειας (βλέπε σχήματα 14 και 15). Στο συνολικό throughput παρατηρείται μία πολύ μικρή πτώση, της τάξης των δύο πακέτων / δευτερόλεπτο, δηλαδή η νέα του τιμή είναι 62 πακέτα / δευτερόλεπτο.



Σχήμα 14: Σύγκλιση ποσοστών βοήθειας κόμβου 0 σε ομοιόμορφη κίνηση χωρίς εγωιστές και μέγεθος πακέτου δεδομένων 1024 bits



Σχήμα 15: Σύγκλιση ποσοστών βοήθειας κόμβου 5 σε ομοιόμορφη κίνηση χωρίς εγωιστές και μέγεθος πακέτου δεδομένων 1024 bits

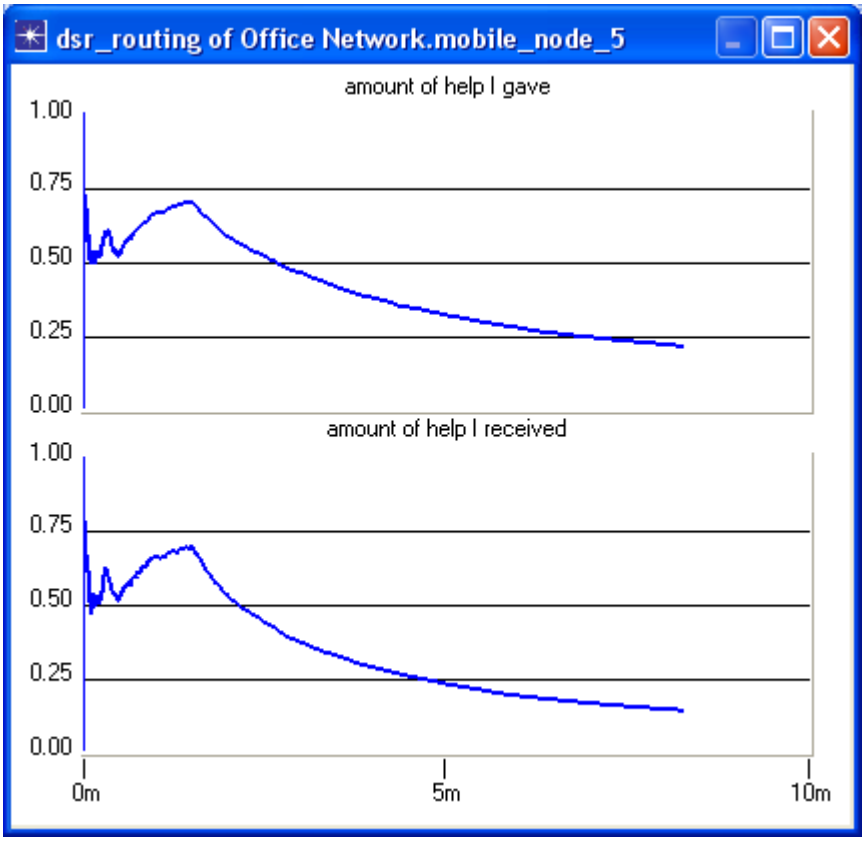


Σχήμα 16: Συνολικό data throughput σε ομοιόμορφη κίνηση χωρίς εγωιστές και μέγεθος πακέτου δεδομένων 1024 bits

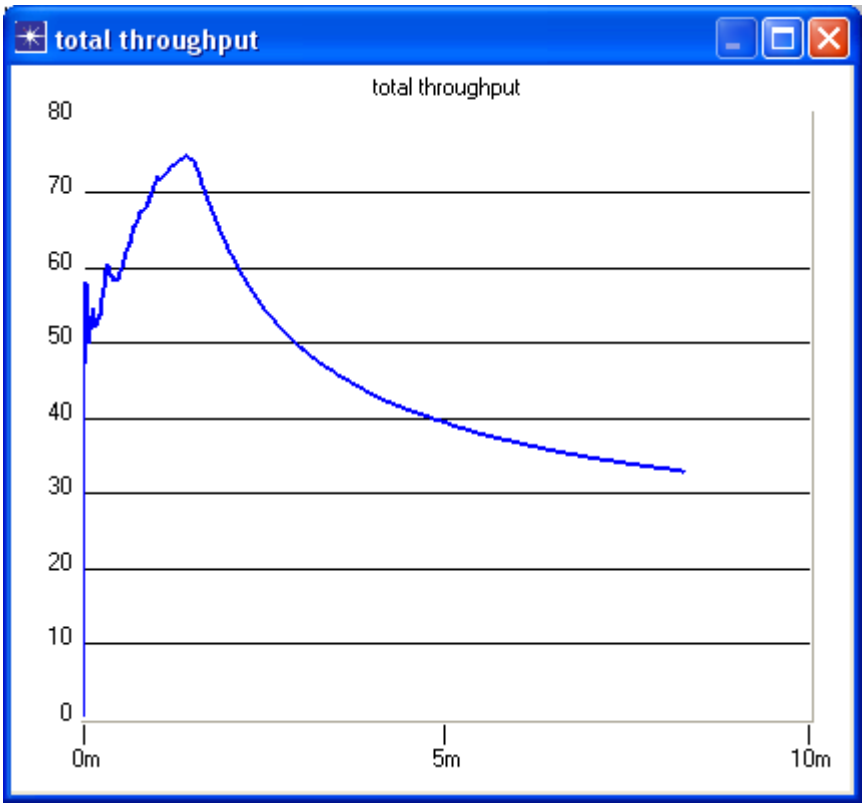
2.2.2. Σενάριο 2.2.: Υποδιπλασιασμός χρονικού διαστήματος μεταξύ δύο διαδοχικών αποστολών πακέτων δεδομένων

Στη δεύτερη απόπειρα για αύξηση του δικτυακού φόρτου, υποδιπλασιάστηκε το χρονικό διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών αποστολών πακέτων από μία πηγή, το οποίο τέθηκε στην τιμή 0.125 sec. Με τον τρόπο αυτό, η ροή των πακέτων στο σύνολό της γίνεται τώρα πιο γρήγορη.

Διατηρώντας τις υπόλοιπες παραμέτρους στις τιμές του σεναρίου 1 παρατηρείται μία πτωτική τάση των αποτελεσμάτων, όσον αφορά το συνολικό throughput του δικτύου, αλλά και το σημείο σύγκλισης των ποσοστών ληφθείσας και δοθείσας βοήθειας (βλέπε σχήματα 17 και 18).

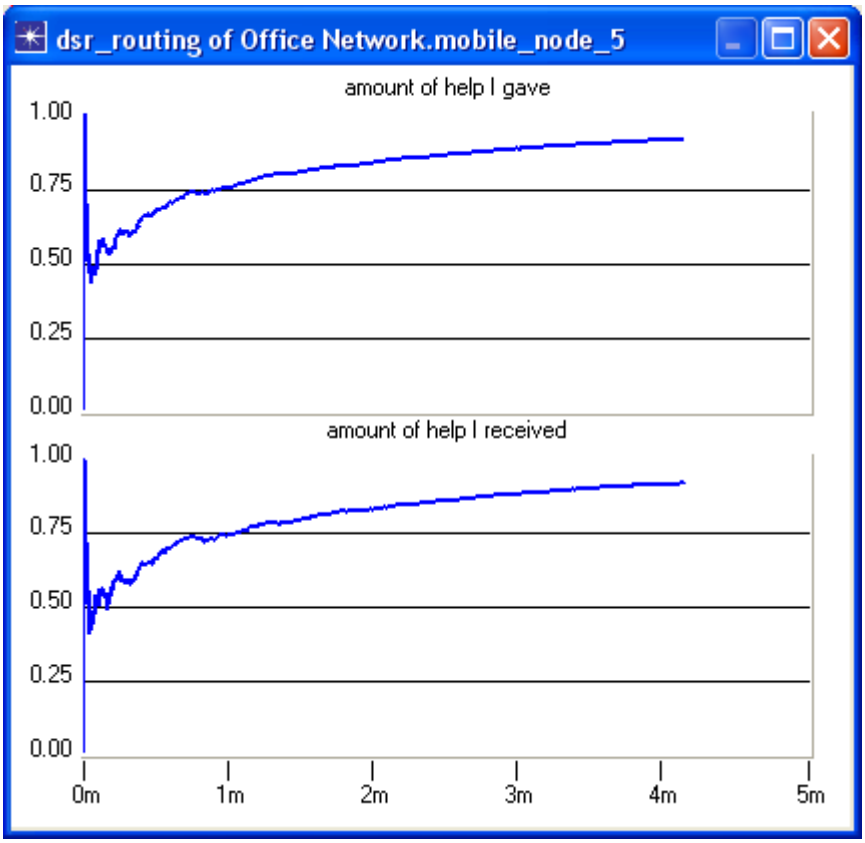


Σχήμα 17: Σύγκλιση ποσοστών βοήθειας κόμβου 5 σε ομοιόμορφη κίνηση χωρίς εγωιστές, exp_arg = 0.125 sec και gen = 0.01

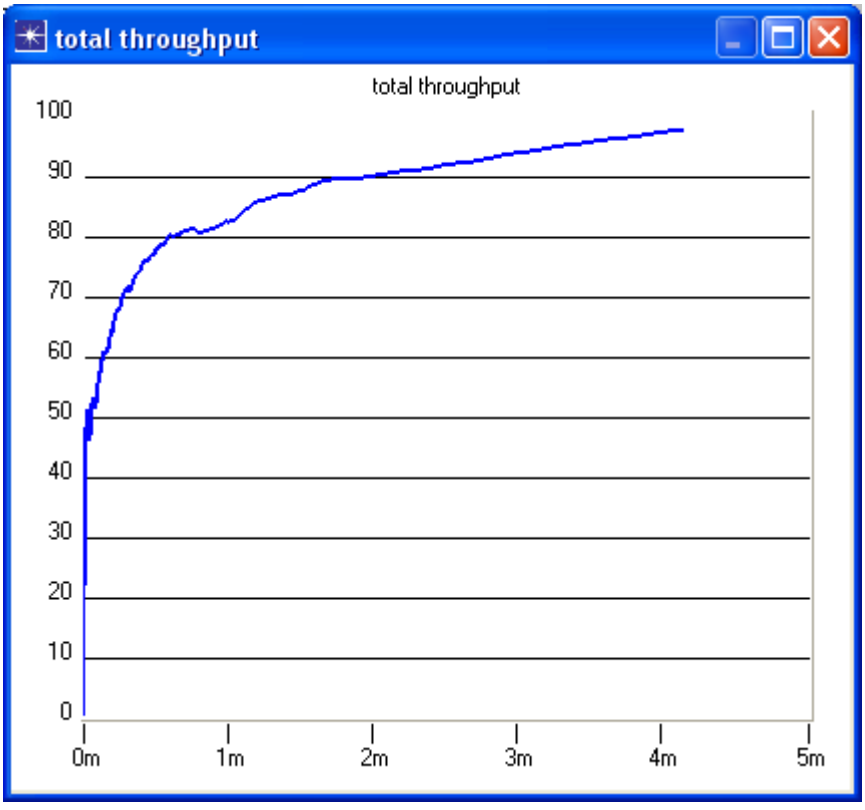


Σχήμα 18: Συνολικό data throughput σε ομοιόμορφη κίνηση χωρίς εγωιστές, exp_arg = 0.125 και gen = 0.01

Το γεγονός αυτό εξηγείται ως εξής: Η παράμετρος της γενναιοδωρίας gen που έχει τιμή 0.01 δεν καταφέρνει στην περίπτωση αυτή να αποτρέψει ένα μεγάλο ποσοστό αρνητικών απαντήσεων που προκύπτει λόγω του υπερβολικού φόρτου στο δίκτυο. Τα πακέτα αιτήσεων έρχονται με ιδιαίτερα γρήγορο ρυθμό με αποτέλεσμα οι κόμβοι να μην προλαβαίνουν έγκαιρα να φτάσουν στο επιθυμητό σημείο ισορροπίας. Η λύση σε αυτή την περίπτωση δίνεται με μία μικρή αύξηση της τιμής της παραμέτρου της γενναιοδωρίας. Τα αποτελέσματα που παρουσιάζονται στα σχήματα 19 και 20 προκύπτουν από την επανάληψη του παραπάνω πειράματος με τιμή γενναιοδωρίας gen = 0.02. Οι κόμβοι είναι τώρα περισσότερο γενναιόδωροι, οπότε αποφεύγονται οι πολλές αρνητικές απαντήσεις κατά την εκκίνηση του δικτύου και το σημείο σύγκλισης των ποσοστών βοήθειας προσεγγίζεται και πάλι πολύ γρήγορα.



Σχήμα19: Σύγκλιση ποσοστών βοήθειας κόμβου 5 σε ομοιόμορφη κίνηση χωρίς εγωιστές, exp_arg = 0.125 και gen = 0.02



Σχήμα 20: Συνολικό data throughput σε ομοιόμορφη κίνηση χωρίς εγωιστές, exp_arg = 0.125 και gen = 0.02

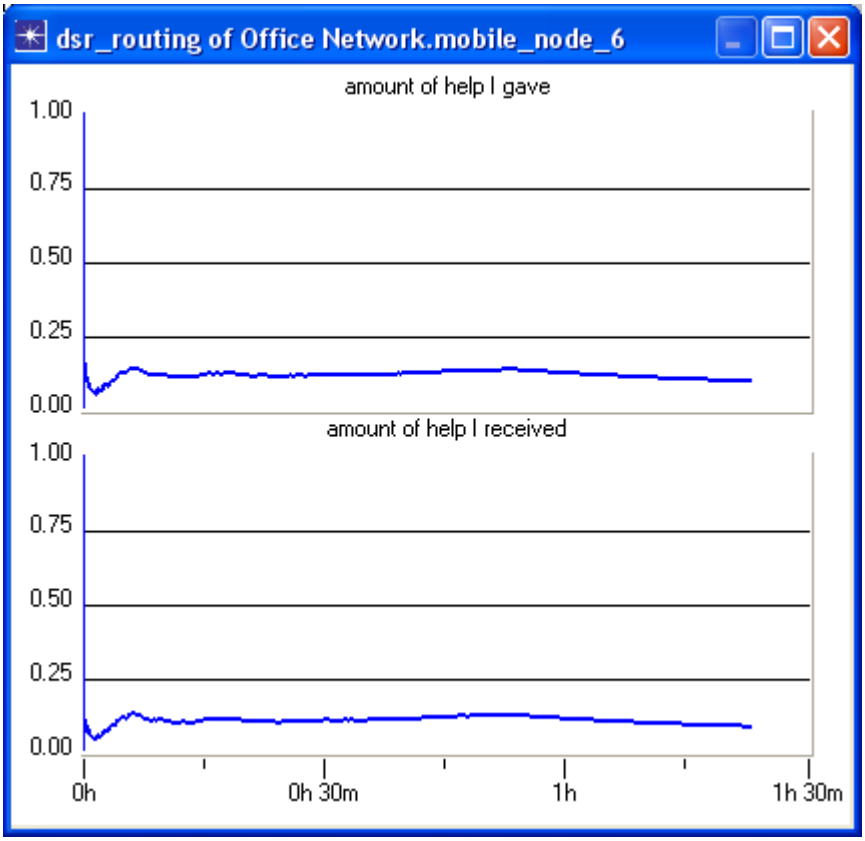
2.3. Σενάριο 3: GTFT υπό την παρουσία εγωιστών κόμβων

Η παρουσία εγωιστών κόμβων επηρεάζει ιδιαίτερα αρνητικά την επίδοση του απλού αλγορίθμου εξίσωσης ποσοστών. Στα πειράματα που ακολουθούν υποθέτουμε ότι κάποιοι κόμβοι υιοθετούν εγωιστική συμπεριφορά, άρα αρνούνται συνεχώς κάθε αίτηση για προώθηση που λαμβάνουν από άλλους κόμβους του δικτύου. Όπως προκύπτει και από τον αλγόριθμο δεν υπάρχει πλέον κάποιο κίνητρο συνεργασίας που να περιλαμβάνει τους κόμβους αυτούς από τώρα και στο εξής. Οι κόμβοι αυτοί απαντούν συνεχώς αρνητικά με αποτέλεσμα το σημείο σύγκλισης των ποσοστών βοήθειας να πέφτει χαμηλά και το συνολικό throughput να μειώνεται δραματικά.

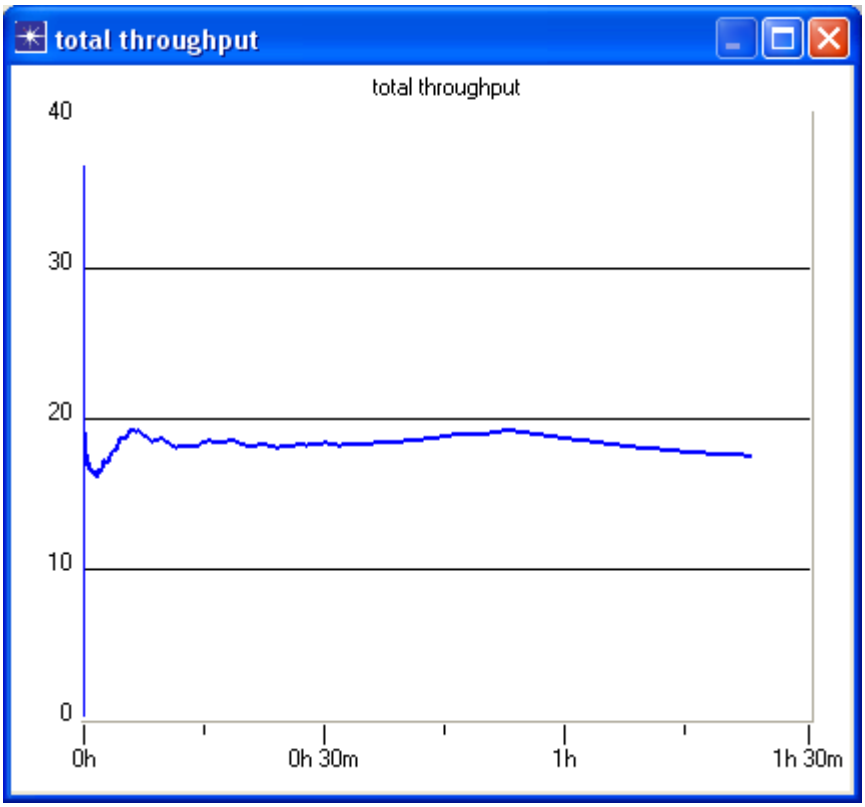
Συγκεκριμένα, παρουσιάζονται τα αποτελέσματα από δύο πειράματα: το πρώτο πείραμα θεωρεί εγωιστή τον κόμβο 5, ενώ το δεύτερο θεωρεί εγωιστές τους κόμβους 5, 9 και 10. Τα αποτελέσματα μαρτυρούν πως καθώς αυξάνεται ο αριθμός και η πυκνότητα των εγωιστών κόμβων, τόσο περισσότερο χειροτερεύει η γενική απόδοση του δικτύου.

2.3.1. Σενάριο 3.1.: GTFT με ένα εγωιστή κόμβο

Στο πείραμα αυτό ο κόμβος 5 (mobile_node_5) θεωρείται εγωιστής. Διατηρώντας όλες τις παραμέτρους στις τιμές που τους είχαν αποδοθεί στο σενάριο 1, η προσομοίωση καταλήγει σε μία δραματική μείωση του throughput, η οποία προσεγγίζει το 70%. Συγκεκριμένα, το throughput πέφτει στην τιμή 19 πακέτα / δευτερόλεπτο. Κατά συνέπεια, το σημείο σύγκλισης των ποσοστών βοήθειας για κάθε κόμβο εντοπίζεται σε μία πολύ χαμηλή τιμή, κοντά στο μηδέν. Στα σχήματα 21 και 22 παρουσιάζονται το σημείο σύγκλισης για τον κεντρικά τοποθετημένο κόμβο 6 και το συνολικό throughput του δικτύου.



Σχήμα 21: Σύγκλιση ποσοστών βοήθειας κόμβου 6 σε ομοιόμορφη κίνηση με τον κόμβο 5 εγωιστή



Σχήμα 22: Συνολικό data throughput σε ομοιόμορφη κίνηση με τον κόμβο 5 εγωιστή

2.3.2. Σενάριο 3.2: GTFT τρεις εγωιστές κόμβους

Στο πείραμα αυτό οι κόμβοι 5, 9 και 10 (mobile_node_5, mobile_node_9 και mobile_node_10) θεωρούνται εγωιστές. Τα αποτελέσματα δείχνουν πως το throughput πέφτει στην τιμή 4 πακέτα / δευτερόλεπτο, δηλαδή παρατηρείται μία πτώση της τάξης του 94 % σε σχέση με το αρχικό throughput που είχε τιμή 64 πακέτα / δευτερόλεπτο.

2.4. Σενάριο 4 : N-GTFT υπό την παρουσία εγωιστών κόμβων

Στα πειράματα που ακολουθούν επαναλαμβάνονται τα σενάρια 3.1 και 3.2 κάνοντας χρήση του αλγορίθμου N-GTFT, ο οποίος κάνει χρήση της ιδιότητας της γειτονίας. Οι πειραματικές παράμετροι παραμένουν ίδιες με εκείνες των αντίστοιχων σεναρίων.

2.4.1. Σενάριο 4.1.: N-GTFT με ένα εγωιστή κόμβο

Στο πείραμα αυτό ο κόμβος 5 (mobile_node_5) θεωρείται εγωιστής και χρησιμοποιείται ο αλγόριθμος N-GTFT. Τα αποτελέσματα της προσομοίωσης καταλήγουν σε θεαματική βελτίωση του συνολικού throughput του δικτύου. Το throughput είναι τώρα σχεδόν τρεις φορές καλύτερο από το αντίστοιχο του σεναρίου 3.1, αφού προσεγγίζει την τιμή των 60 πακέτων / δευτερόλεπτο. Επιπρόσθετα, το σημείο σύγκλισης των ποσοστών βοήθειας για κάθε κόμβο εντοπίζεται και πάλι σε υψηλή τιμή. Βέβαια, οι κόμβοι οι οποίοι είναι τοποθετημένοι πιο κοντά στον εγωιστή κόμβο έχουν χαμηλότερο σημείο σύγκλισης από τους υπόλοιπους – μιας και αυτοί λόγω θέσης ζητούν τη βοήθεια των εγωιστών πιο συχνά – είναι, όμως, και αυτό ικανοποιητικά υψηλό.

2.4.2. Σενάριο 4.2.: N-GTFT με τρεις εγωιστές κόμβους

Στο πείραμα αυτό οι κόμβοι 5, 9 και 10 (mobile_node_5, mobile_node_9 και mobile_node_10) θεωρούνται εγωιστές και χρησιμοποιείται ο αλγόριθμος N-GTFT. Τα αποτελέσματα της προσομοίωσης καταλήγουν και πάλι σε βελτίωση του συνολικού throughput, αντίστοιχη με εκείνη που παρατηρήθηκε στο σενάριο 4.1. Και πάλι το throughput σχεδόν τριπλασιάζεται, καθώς φτάνει στην τιμή των 13 πακέτων / δευτερόλεπτο.

2.5. Σενάριο 5: Περίπτωση εκρηκτικής κίνησης

Στα πειράματα που ακολουθούν εξετάζονται σενάρια, όπου η κίνηση των πακέτων δεδομένων, άρα και των αντίστοιχων πακέτων αίτησης και απάντησης για προώθηση ακολουθεί εκρηκτικό μοτίβο (burst traffic). Συγκεκριμένα, θεωρώντας και πάλι την τοπολογία με τους 16 κόμβους, όπως αυτή παρουσιάζεται στο σχήμα 10, ο κάθε κόμβος αποστέλλει κίνηση που διαρκεί για ένα τυχαίο χρονικό διάστημα και στη συνέχεια μένει αδρανής για επίσης τυχαίο χρόνο. Τα τυχαία αυτά χρονικά διαστήματα επιλέγονται κάθε φορά από ένα διάστημα μεταξύ του ενός και των δύο δευτερολέπτων, χρησιμοποιώντας την ομοιόμορφη κατανομή (uniformaly distributed).

2.5.1. Σενάριο 5.1.: GTFT με εκρηκτική κίνηση

Στο πείραμα αυτό όλες οι παράμετροι έχουν ίδιες τιμές με αυτές του σεναρίου 1, ενώ η κίνηση παράγεται και δρομολογείται με τον τρόπο που περιγράφηκε στην ενότητα 2.5. Προκύπτει πως ο απλός GTFT σε σενάριο εκρηκτικής κίνησης οδηγεί σε συνολικό data throughput με τιμή 45 πακέτα / δευτερόλεπτο.

2.5.2. Σενάριο 5.2.: Bucket με εκρηκτική κίνηση

Και στο πείραμα αυτό όλες οι παράμετροι παραμένουν όπως στο σενάριο 1, ενώ πλέον δε χρειάζεται καθόλου η παράμετρος της γενναιοδωρίας (gen). Η αρχική τιμή του κουβά κάθε κόμβου τίθεται στην τιμή 0.5, έτσι ώστε κάθε κουβάς να θεωρείται μισογεμάτος κατά την εκκίνηση της προσομοίωσης.

Προκύπτει πως ο bucket αλγόριθμος δίνει καλύτερα αποτελέσματα σε περιπτώσεις εκρηκτικής κίνησης. Συγκεκριμένα, παρατηρείται μία βελτίωση του συνολικού throughput του δικτύου κατά 37% σε σχέση με τον απλό GTFT, το οποίο λαμβάνει την τιμή των 61 πακέτων / δευτερόλεπτο.

Κατά την εκτέλεση πειραματικών σεναρίων με εκρηκτική κίνηση ο GTFT αλγόριθμος σταματά να αποδέχεται αιτήσεις για προώθηση

κίνησης αμέσως μόλις η συνθήκη (5) γίνει αληθής (ενότητα Γ.1). Αντίθετα, ο αλγόριθμος Bucket συνεχίζει να αποδέχεται αιτήσεις για μερικά ακόμα δευτερόλεπτα, μέχρι τη στιγμή που θα αδειάσει τελείως ο κουβάς. Αυτός είναι και ο λόγος που παρατηρείται αύξηση στο συνολικό data throughput του δικτύου.

Στο σημείο αυτό θα μπορούσαμε να παρατηρήσουμε ότι η ενημέρωση της τιμής του κουβά για κάθε κόμβο θα μπορούσε να ακολουθεί και ένα άλλο pattern διαφορετικό από εκείνο που περιγράφηκε στην ενότητα Δ.3. Το pattern αυτό δίνεται από την ακόλουθη σχέση:

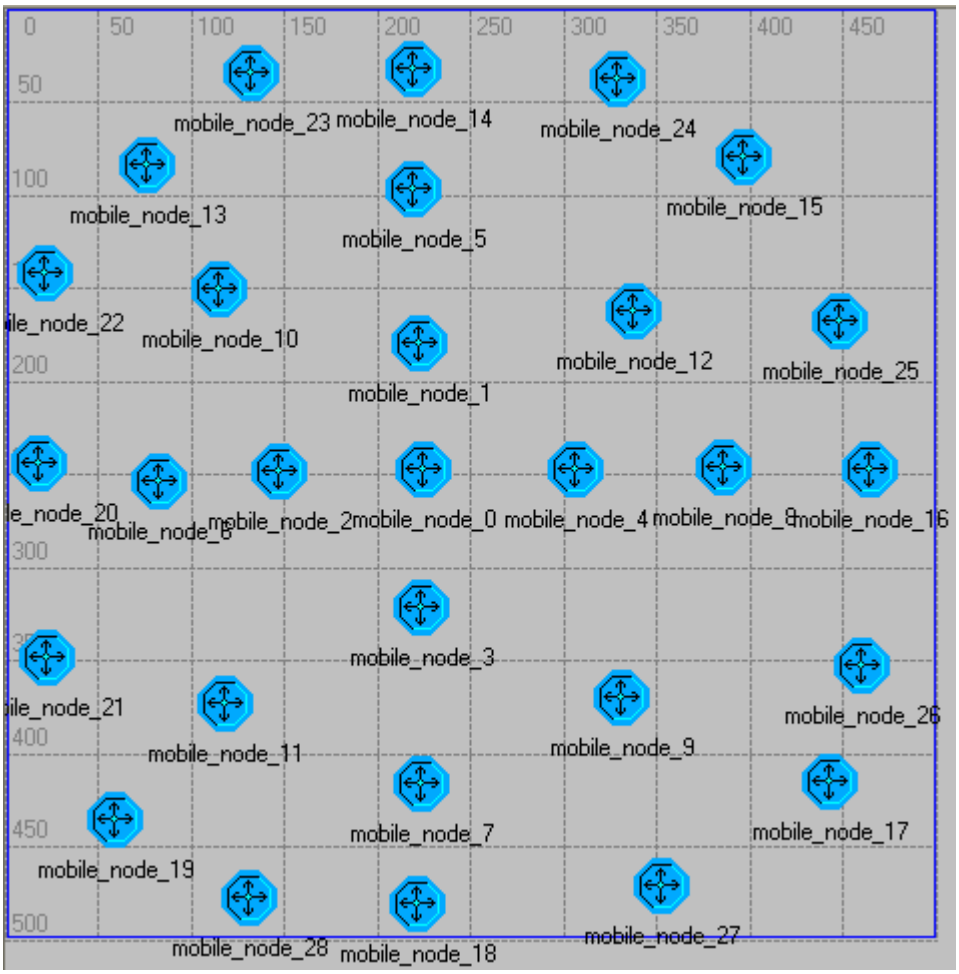
$$B += \frac{HR_{i-1}}{t_{i-1}} - \frac{HG_{i-1}}{t_{i-1}} \quad (12)$$

Με βάση αυτό το pattern ένας κόμβος έχει τη δυνατότητα να διαφοροποιήσει τη συμπεριφορά του. Για παράδειγμα, μπορεί να γίνει λίγο περισσότερο γενναιόδωρος ως προς τους άλλους σε περίπτωση που έχει βοηθηθεί κατά ένα συγκεκριμένο ποσοστό σε συντομότερο χρονικό διάστημα απ' ότι στο παρελθόν. Για παράδειγμα, αν ένας κόμβος αντιλαμβάνεται την ίδια διαφορά μεταξύ των μεταβλητών HR και HG κατά τη διάρκεια δύο συγκεκριμένων χρονικών περιόδων t_x και t_y με $t_x << t_y$, η σχέση (12) καταφέρνει να ανανεώσει τον κουβά με μεγαλύτερη τιμή για τη χρονική περίοδο t_x απ' ότι η σχέση (10). Προφανώς, οι χρονικές περίοδοι t_x και t_y αντιστοιχούν στη μεταβλητή t_{i-1} της σχέσης (12).

2.6. Σενάριο 6: Περίπτωση access point

Στα πειράματα αυτής της ενότητας θεωρείται η ύπαρξη ενός κόμβου που διαδραματίζει το ρόλο ενός access point. Ο κόμβος αυτός είναι τοποθετημένος στο κέντρο μιας «κυκλικής τοπολογίας». Όλοι οι κόμβοι του δικτύου αποστέλλουν πακέτα δεδομένων με 80% πιθανό προορισμό το access point και 20% πιθανό προορισμό οποιοδήποτε άλλο τυχαίο κόμβο στο δίκτυο. Φροντίζουμε, δηλαδή, να δρομολογείται το μεγαλύτερο ποσοστό

της συνολικής κίνησης προς το access point. Η ακριβής τοπολογία που χρησιμοποιούμε παριστάνεται στο σχήμα 23.



Σχήμα23: Τοπολογία 29 κόμβων με access point

Συγκεκριμένα, θεωρούμε πως το access point περιβάλλεται από τρεις ζώνες. Η πιο εσωτερική ζώνη (A) έχει τέσσερις κόμβους, η μεσαία ζώνη (B) έχει οκτώ κόμβους, ενώ η πιο εξωτερική ζώνη (C) διαθέτει δεκαέξι κόμβους.

2.6.1. Σενάριο 6.1.: GTFT με access point

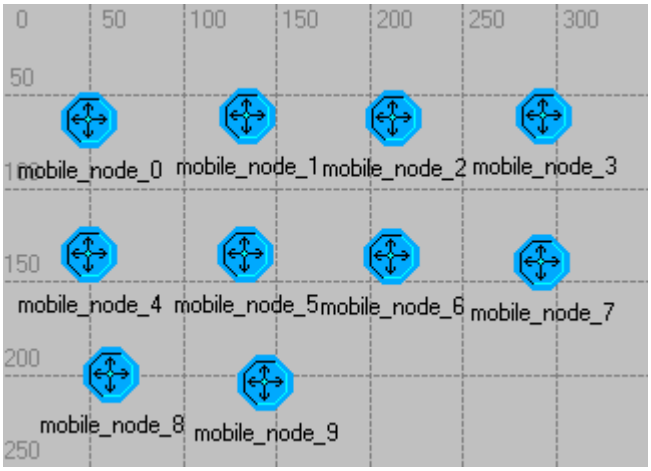
Στο πείραμα αυτό όλες οι παράμετροι έχουν ίδιες τιμές με αυτές του σεναρίου 1, ενώ η κίνηση παράγεται και δρομολογείται με τον τρόπο που περιγράφηκε στην ενότητα 2.6. Προκύπτει πως ο απλός GTFT σε σενάριο με access point οδηγεί σε συνολικό data throughput με τιμή 46 πακέτα / δευτερόλεπτο.

2.6.2. Σενάριο 6.2: W-GTFT με access point

Κάνοντας χρήση της εκδοχής που χρησιμοποιεί διαφορετικά βάρη όσον αφορά την απόφαση αποδοχής ή απόρριψης μιας αίτησης, τα οποία σχετίζονται με τη γεωγραφική θέση του κάθε κόμβου με τον τρόπο που περιγράφηκε στην ενότητα Δ.5, το συνολικό throughput αποκτά τιμή ίση με 83 πακέτα / δευτερόλεπτο. Κατά συνέπεια, η χρήση αυτής της εκδοχής του αλγορίθμου οδηγεί σε βελτίωση του συνολικού throughput της τάξης του 82%, σε σχέση με τα αποτελέσματα του σεναρίου 6.3.

3. Γενικές παρατηρήσεις – Συγκεντρωτικά αποτελέσματα

Σε παρόμοια αποτελέσματα και συμπεράσματα οδήγησε και η χρήση διαφορετικών γεωγραφικών τοπολογιών του δικτύου. Για παράδειγμα στην περίπτωση των σεναρίων απλής ομοιόμορφης και εκρηκτικής κίνησης χρησιμοποιήθηκε και μία τοπολογία με 10 κόμβους (Σχήμα 24), ενώ στην περίπτωση του access point χρησιμοποιήθηκε και μία εκδοχή με 22 κόμβους.



Σχήμα 24: Τοπολογία 10 κόμβων

Σε κάθε πειραματικό σενάριο τα αποτελέσματα που παρουσιάστηκαν προέκυψαν από το μέσο όρο ενός συνόλου από 100 επαναλήψεις του ίδιου πειράματος, όπου η κάθε επανάληψη είχε διαφορετικό seed. Με τον τρόπο αυτό θεωρούμε ότι εξασφαλίστηκε η απαραίτητη τυχαιότητα. Στον πίνακα που ακολουθεί παρουσιάζονται τα

αποτελέσματα των πιο σημαντικών πειραμάτων που εκτελέστηκαν κατά τη διάρκεια της παρούσας εργασίας.

Εκδοχή αλγορίθμου	**Συνολικό throughput**
GTFT χωρίς εγωιστές κόμβους	65 πακέτα / δευτερόλεπτο
GTFT με έναν εγωιστή κόμβο	19 πακέτα / δευτερόλεπτο
N-GTFT με έναν εγωιστή κόμβο	60 πακέτα / δευτερόλεπτο
GTFT με τρεις εγωιστές κόμβους	04 πακέτα / δευτερόλεπτο
N-GTFT με τρεις εγωιστές κόμβους	13 πακέτα / δευτερόλεπτο
GTFT με εκρηκτική κίνηση	45 πακέτα / δευτερόλεπτο
Bucket με εκρηκτική κίνηση	61 πακέτα / δευτερόλεπτο
Bucket με εκρηκτική κίνηση και έναν εγωιστή κόμβο	21 πακέτα / δευτερόλεπτο
N-Bucket με εκρηκτική κίνηση και έναν εγωιστή κόμβο	52 πακέτα / δευτερόλεπτο
N-GTFT με εκρηκτική κίνηση και έναν εγωιστή κόμβο	48 πακέτα / δευτερόλεπτο
GTFT με access point	46 πακέτα / δευτερόλεπτο
W-GTFT με access point	83 πακέτα / δευτερόλεπτο
W-Bucket με access point	85 πακέτα / δευτερόλεπτο

ΣΤ΄ ΜΕΡΟΣ : ΣΥΜΠΕΡΑΣΜΑΤΑ

1. Συμπεράσματα

Σε αυτή την εργασία παρουσιάστηκαν ορισμένες ιδέες για τη δημιουργία κινήτρων συνεργασίας μεταξύ των κόμβων ενός ασύρματου δικτύου αυθαίρετης δομής. Η υλοποίηση των ιδεών αυτών επιτεύχθηκε κάνοντας ορισμένες προσθήκες – αλλαγές στον αλγόριθμο δρομολόγησης DSR. Τα αποτελέσματα των προσομοιώσεων έδειξαν πως οι κόμβοι καταφέρνουν και συνεργάζονται με ωριμότητα και συνέπεια όταν υπάρχουν τα κατάλληλα κίνητρα, ενώ με τη χρήση κατάλληλων τεχνικών είναι σε θέση να εντοπίσουν και να απομονώσουν κακόβουλες συμπεριφορές. Όπως αποδείχθηκε, ειδικά η γνώση για την παρελθούσα συμπεριφορά κάθε γείτονα ξεχωριστά αποβαίνει ιδιαίτερα χρήσιμη στον εντοπισμό και την τιμωρία των εγωιστών κόμβων. Στην περίπτωση των access points η χρήση διαφορετικών βαρών ως προς το ποσοστό αιτήσεων που πρέπει να γίνονται αποδεκτές προτείνεται ως μία πολύ καλή λύση για την αποφυγή ενός χαμηλού σημείου σύγκλισης. Από την άλλη πλευρά, τα σενάρια που προσομοιώνουν εκρηκτικού τύπου κίνηση καταλήγουν σε καλύτερα αποτελέσματα όταν χρησιμοποιείται ο "bucket" αλγόριθμος, ο οποίος λειτουργεί με σκοπό την εξισορρόπηση των ρυθμών των δύο τύπων βοήθειας: δοθείσας και ληφθείσας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Παράρτημα Α

FORWARD_REQUEST_PACKET_TYPE

Field	bits	Description
SRC	8	The dsr address of the forward request packet source node
DEST	8	The dsr address of the forward request packet destination node
RELAY	8	The dsr address of the forward request packet next relay node
Seg_Left	8	The number of hops on which the forward request packet must be forwarded before reaching its destination
Size_Route	8	The size of the route that the forward request packet must follow
Seq_Number	8	The sequence number of the forward request packet
Type	8	The type of the packet (that is FORWARD_REQUEST_PACKET_TYPE for a forward request packet)
Node_0	8	The first node dsr address of the forward request packet route (that is the dsr address of the packet source node)
Node_1	8	The second node dsr address of the forward request packet route
Node_2	8	The third node dsr address of the forward request packet route
Node_3	8	The fourth node dsr address of the forward request packet route
Node_4	8	The fifth node dsr address of the forward request packet route
Node_5	8	The sixth node dsr address of the forward request packet route
Node_6	8	The seventh node dsr address of the forward request packet route
Node_7	8	The eighth node dsr address of the forward request packet route
TR_source	16	The objid of the last packet transmitor: used to compute the transmission range

FORWARD_REPLY_PACKET_TYPE

Field	Bits	Description
SRC	8	The dsr address of the forward request packet (to which this packet reply) final destination node
DEST	8	The dsr address of the forward reply packet destination node (that is the forward request source node)
RELAY	8	The dsr address of the forward reply packet next relay node

Seg_Left	8	The number of hops on which the forward reply packet must be forwarded before reaching its destination
Size_Route	8	The size of the route that the forward reply packet must follow
Seq_Number	8	The sequence number of the forward request packet to which this packet reply
Type	8	The type of the packet (that is FORWARD_REPLY_PACKET_TYPE for a forward reply packet)
Node_0	8	The first node dsr address of the forward request packet route (that is the dsr address of the forward request packet generator) to which this packet reply Thus it is the dsr address of the forward reply destination node.
Node_1	8	The second node dsr address of the route that the forward request packet to which this packet reply has followed
Node_2	8	The third node dsr address of the route that the forward request packet to which this packet reply has followed
Node_3	8	The fourth node dsr address of the route that the forward request packet to which this packet reply has followed
Node_4	8	The fifth node dsr address of the route that the forward request packet to which this packet reply has followed
Node_5	8	The sixth node dsr address of the route that the forward request packet to which this packet reply has followed
Node_6	8	The seventh node dsr address of the route that the forward request packet to which this packet reply has followed
Node_7	8	The eighth node dsr address of the route that the forward request packet to which this packet reply has followed
Answer	1	The final answer: if all relays accept the forward request the answer field is set to 1,otherwise, the answer field is set to 0
TR_source	16	The objid of the last packet transmitter: used to compute the transmission range

Function	**Description**
void start_forward_request(int destination_dsr_address);	initiates the check for forward permission
void handle_forward_request(Packet* pk_ptr);	handles the incoming forward request packets
void forward_forward_request(Packet* pk_ptr);	forwards the forward request packet received by a relay node
void start_forward_reply(Packet*	initiates the reply from relay of forward

pk_ptr, int answer);	permission request
void handle_forward_reply(Packet* pk_ptr);	handles the incoming forward reply packets
void forward_forward_reply(Packet* pk_ptr);	forwards the forward reply packet received by a relay node

Παράρτημα Β

```
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
/////////////// FORWARD REQUEST - REPLY FUNCTIONS
//////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

/*************************************************************
**********************/
/*                START_FORWARD_REQUEST
*/
/*************************************************************
**********************/
/* this function initiates the check for forward permission.
/* int destination_dsr_address: the dsr address of the forward
/* request destination
/*************************************************************
**********************/

void start_forward_request (int destination_dsr_address)
{
int size_route;
int relay_mac_address;
char field[10];
int i;

Packet* fw_req_pk_ptr;

printf("Node %d:The destination of the request packet is
%d\n",my_dsr_address,destination_dsr_address);
printf("Node %d:The sequence number of the request packet is
%d\n",my_dsr_address,my_sequence_number);
```

```
// create the forward request packet and set its type field
fw_req_pk_ptr=op_pk_create_fmt("Fw_Req");
op_pk_nfd_set(fw_req_pk_ptr,"Type",FORWARD_REQUEST_PACKET_TYPE
);

// set the DEST field of the forward request packet with this
mac address
op_pk_nfd_set(fw_req_pk_ptr,"DEST",destination_dsr_address);

// set the source field of this forward request packet (I am
the source)
op_pk_nfd_set(fw_req_pk_ptr,"SRC",my_dsr_address);

// extract the size of the route contained in the route cache
size_route=route_cache[destination_dsr_address].size_route;
printf("Node %d:The SIZE_ROUTE is:
%d\n",my_dsr_address,size_route);

// set the seg_left and size_route fields
op_pk_nfd_set(fw_req_pk_ptr,"Size_Route",size_route);
op_pk_nfd_set(fw_req_pk_ptr,"Seg_Left",size_route-2);

// write the route of the forward request packet in the Node_k
fields
printf("Node %d:The route is: ",my_dsr_address);
for (i=0;i<size_route;i++)
  {
  sprintf(field,"Node_%d",i);

op_pk_nfd_set(fw_req_pk_ptr,field,route_cache[destination_dsr_
address].route[i]);
  printf("%s: %d
",field,route_cache[destination_dsr_address].route[i]);
  }
printf("\n");

// set the RELAY field of the forward request packet (the
first relay is the first node of the route)
op_pk_nfd_set(fw_req_pk_ptr,"RELAY",route_cache[destination_ds
r_address].route[1]);

// convert this relay dsr address in a mac address via the dsr
support package
relay_mac_address=dsr_support_get_mac_from_dsr_address(route_c
ache[destination_dsr_address].route[1]);

// set the packet id of the forward request packet
op_pk_nfd_set
(fw_req_pk_ptr,"Seq_number",my_sequence_number++);

// set the TR_source field for the TR purpose
op_pk_nfd_set(fw_req_pk_ptr,"TR_source",my_node_objid);

// send the forward request packet to the first relay node
dsr_send_to_mac(fw_req_pk_ptr,relay_mac_address);
```

```
sprintf(message,"Node %d:I am sending the forward request
packet for %d to first relay dsr address
%d\n",my_dsr_address,destination_dsr_address,route_cache[desti
nation_dsr_address].route[1]);
printf("%s", message);

// increment (by one) the counter of total forward requests
generated by this node
my_total_req_made_by_me++;

op_stat_write(stat_my_total_req_made_by_me,my_total_req_made_b
y_me);

// breakpoint for debugging purpose
if (1)
  {
  op_prg_odb_bkpt("start_forward_request");
  }
}
```

```
/*************************************************************
************************************/
/*                HANDLE_FORWARD_REQUEST
*/
/*************************************************************
************************************/
/* this function handles the incoming forward request packets.
/* Packet* pk_ptr: a pointer to the forward request packet to
process
/*************************************************************
************************************/

void handle_forward_request (Packet* fw_req_pk_ptr)
{
int source_dsr_address;
int destination_dsr_address;
int relay_dsr_address;
int r_sequence_number;
int size_route;
int seg_left;
Objid trsource;

// extract some useful information from the forward request
packet
op_pk_nfd_get (fw_req_pk_ptr,"SRC",&source_dsr_address);
op_pk_nfd_get (fw_req_pk_ptr,"DEST",&destination_dsr_address);
op_pk_nfd_get (fw_req_pk_ptr,"RELAY",&relay_dsr_address);
op_pk_nfd_get (fw_req_pk_ptr,"TR_source",&trsource);
op_pk_nfd_get (fw_req_pk_ptr,"Seq_number",&r_sequence_number);
op_pk_nfd_get (fw_req_pk_ptr,"Size_Route",&size_route);
op_pk_nfd_get (fw_req_pk_ptr,"Seg_Left",&seg_left);
```

```
if(my_req_negative_answers_i_gave==0)
  {
  op_stat_write(stat_amount_of_help_i_gave, 1);
  }
else
  {
  op_stat_write(stat_amount_of_help_i_gave,

(double)my_req_handled_for_others/(double)my_total_req_made_to
_me_by_others);
  }

if(my_req_negative_answers_i_received==0)
  {
  op_stat_write(stat_amount_of_help_i_received, 1);
  }
else
  {
  op_stat_write(stat_amount_of_help_i_received,

(double)my_req_handled_for_me/(double)my_total_req_made_by_me)
;
  }



if(source_dsr_address != my_dsr_address)
  {
  // if the node can accept the forward request

if((((double)(my_req_handled_for_me)/(double)(my_total_req_mad
e_by_me)) +
  my_generous) >

((double)(my_req_handled_for_others)/(double)(my_total_req_mad
e_to_me_by_others))
  && (my_bad_node==0))
        {
        // if the current node is the final relay of the
forward request packet (just before
        the destination)
        if((seg_left==1)&&(my_dsr_address==relay_dsr_address))
              {
              printf("Node %d:Entering start_forward reply
function with positive
              answer\n", my_dsr_address);
              start_forward_reply(fw_req_pk_ptr, 1);
              my_req_handled_for_others++;
              op_stat_write(stat_my_req_handled_for_others,
              my_req_handled_for_others);
              printf("Node %d:Exiting start_forward reply
function with positive
              answer\n", my_dsr_address);
```

```
            // increment (by one) the counter of total forward
            requests made to this node by others
             my_total_req_made_to_me_by_others++;

op_stat_write(stat_my_total_req_made_to_me_by_others,
             my_total_req_made_to_me_by_others);
             }

       // if the current node is not the final relay of the
forward request packet
       else
if((seg_left>1)&&(my_dsr_address==relay_dsr_address))
             {
             printf("Node %d:Entering forward_forward_request
             function\n",my_dsr_address);
             forward_forward_request(fw_req_pk_ptr);
             printf("Node %d:Exiting forward_forward_request
             function\n",my_dsr_address);

            // increment (by one) the counter of total forward
           requests made to this
            node by others
             my_total_req_made_to_me_by_others++;

op_stat_write(stat_my_total_req_made_to_me_by_others,
             my_total_req_made_to_me_by_others);
             }




       // if the current node is not a relay of the forward
request packet
       else if(my_dsr_address!=relay_dsr_address)
             {
             // destroy the forward request packet
             printf("Node %d:I am not concerned by this request
             packet\n",my_dsr_address);
             op_pk_destroy(fw_req_pk_ptr);
             }
       }

// if the node can not accept the forward request
else
     {
     if((my_dsr_address==relay_dsr_address)||(my_bad_node==1))
             {
             printf("Node %d:Entering start_forward reply
function with negative
             answer\n", my_dsr_address);
             start_forward_reply(fw_req_pk_ptr, 0);
             my_req_negative_answers_i_gave++;
             op_stat_write(stat_my_req_negative_answers_i_gave,
             my_req_negative_answers_i_gave);
```

```
            printf("Node %d:Exiting start_forward reply
function with negative
            answer\n", my_dsr_address);
            printf("Node %d: Negative answers:
%d\n",my_dsr_address,
            my_req_negative_answers_i_gave);

            // increment (by one) the counter of total forward
requests made to this
            node by others
            my_total_req_made_to_me_by_others++;

op_stat_write(stat_my_total_req_made_to_me_by_others,
            my_total_req_made_to_me_by_others);
            }
            else
            {
            printf("Node %d:I am not concerned by this request
\
            packet\n",my_dsr_address);
            op_pk_destroy(fw_req_pk_ptr);
            }
        }
}

// breakpoint for debugging purpose
if (1)
  {
  op_prg_odb_bkpt("handle_forward_request");
  }
}

/*************************************************************
*************************************/
/*               START_FORWARD_REPLY
*/
/*************************************************************
*************************************/
/* this function initiates the reply from relay of forward
permission request.
/* Packet* pk_ptr: the forward request packet that will be
/* used to construct the forward reply packet
/* int answer: the final answer of the forward permission
/* request. If all relay nodes agree to forward the data, the
answer
/* is yes (1). Otherwise, the answer is no (0)
/*************************************************************
*************************************/

void start_forward_reply (Packet* fw_req_pk_ptr, int answer)
{
Packet* fw_rep_pk_ptr;
int forward_request_source_dsr_address;
int node_dsr_address;
int relay_dsr_address;
int relay_mac_address;
```

```
char field[10];
int seg_left,size_route;
int i;
int r_sequence_number;
int node_num = 0;

// create the forward reply packet and set its type field
fw_rep_pk_ptr = op_pk_create_fmt("Fw_Reply");
op_pk_nfd_set(fw_rep_pk_ptr,"Type",FORWARD_REPLY_PACKET_TYPE);

// since I am the forward request target I am also the forward
reply source
op_pk_nfd_set(fw_rep_pk_ptr,"SRC",my_dsr_address);

// set the forward reply DEST field => the forward request
source is the forward reply destination
op_pk_nfd_get(fw_req_pk_ptr,"SRC",&forward_request_source_dsr_
address);
op_pk_nfd_set(fw_rep_pk_ptr,"DEST",forward_request_source_dsr_
address);

printf("Node %d:The destination of the reply packet is
%d\n",my_dsr_address, forward_request_source_dsr_address);

// set the forward reply Seq_number field => same sequence
number than the forward request
op_pk_nfd_get(fw_req_pk_ptr,"Seq_number",&r_sequence_number);
op_pk_nfd_set(fw_rep_pk_ptr,"Seq_number",r_sequence_number);

printf("Node %d:The sequence number of the reply packet is
%d\n",my_dsr_address,r_sequence_number);


// set the answer field
op_pk_nfd_set(fw_rep_pk_ptr,"answer",answer);

op_pk_nfd_get(fw_req_pk_ptr,"RELAY",&relay_dsr_address);

while(1)
  {
  sprintf(field,"Node_%d",node_num);
  op_pk_nfd_get(fw_req_pk_ptr,field,&node_dsr_address);
  if(node_dsr_address==relay_dsr_address)
        {
        break;
        }
        else
        {
        node_num++;
        }
  }

printf("Node %d:Starting reply procedure\n",my_dsr_address);

sprintf(field,"Node_%d",(node_num-1));
op_pk_nfd_get(fw_req_pk_ptr,field,&relay_dsr_address);
```

```
op_pk_nfd_set(fw_rep_pk_ptr,"RELAY",relay_dsr_address);

op_pk_nfd_set(fw_rep_pk_ptr,"Size_Route",(node_num+1));
op_pk_nfd_set(fw_rep_pk_ptr,"Seg_Left",(node_num-1));

printf("Node %d:Forward Reply Route: ",my_dsr_address);

for (i=0;i<=node_num;i++)
  {
  sprintf(field,"Node_%d",(node_num-i));
  op_pk_nfd_get(fw_req_pk_ptr,field,&node_dsr_address);
  sprintf(field,"Node_%d",i);
  op_pk_nfd_set(fw_rep_pk_ptr,field,node_dsr_address);
  printf("Node %d...",node_dsr_address);
  }
printf("\n");

// set the TR_source fields for the Transmission range purpose
op_pk_nfd_set(fw_rep_pk_ptr,"TR_source",my_node_objid);

// convert the dsr address of the first relay in its mac
address
relay_mac_address =
dsr_support_get_mac_from_dsr_address(relay_dsr_address);


// and send the forward reply packet immediately since I am
the target of the forward request
dsr_send_to_mac(fw_rep_pk_ptr,relay_mac_address);

// and destroy the forward request packet
op_pk_destroy(fw_req_pk_ptr);

// breakpoint for debugging purpose
if (1)
  {
  op_prg_odb_bkpt("start_forward_reply");
  }
}
```

```
/*************************************************************
****************************************/
/*                  FORWARD_FORWARD_REQUEST
*/
/*************************************************************
****************************************/
/* this function forward the forward request packet received
by
/* a relay node
/* Packet* fw_req_pk_ptr: a pointer to the forward request
packet to forward
/*************************************************************
****************************************/

void forward_forward_request (Packet* fw_req_pk_ptr)
{
int size_route;
int seg_left;
int next_relay_dsr_address;
int next_relay_mac_address;
char field[10];

// read the seg_left and the size_route field in the packet
op_pk_nfd_get (fw_req_pk_ptr,"Seg_Left",&seg_left);
op_pk_nfd_get (fw_req_pk_ptr,"Size_Route",&size_route);

// extract the next node dsr address from the packet (node
size_route-seg_left)
sprintf(field,"Node_%d",size_route-seg_left);
op_pk_nfd_get(fw_req_pk_ptr,field,&next_relay_dsr_address);

// and set it as the next packet relay
op_pk_nfd_set(fw_req_pk_ptr,"RELAY",next_relay_dsr_address);
printf("Node %d:FORWARD_FORWARD_REQUEST FUNCTION: The next
relay address is Node
%d\n",my_dsr_address,next_relay_dsr_address);

// update the seg_left field of the packet
```

```
op_pk_nfd_set(fw_req_pk_ptr,"Seg_Left",--seg_left);

// set the TR_source field for the Transmission Range purpose
op_pk_nfd_set(fw_req_pk_ptr,"TR_source",my_node_objid);

// convert the dsr address of the next relay in its mac
address
next_relay_mac_address=dsr_support_get_mac_from_dsr_address(ne
xt_relay_dsr_address);

// and send the packet
dsr_send_to_mac(fw_req_pk_ptr,next_relay_mac_address);

// breakpoint for debugging purpose
if (1)
  {
  op_prg_odb_bkpt("forward_forward_request");
  }
}
/*************************************************************
***********************************/
/*                  HANDLE_FORWARD_REPLY
*/
/*************************************************************
***********************************/
/* this function handles the incoming forward reply packets.
/* Packet* pk_ptr: a pointer to the forward reply packet to
process
/*************************************************************
***********************************/

void handle_forward_reply (Packet* fw_rep_pk_ptr)
{
int source_address;
int destination_address;
int r_sequence_number;
int relay_address;
int relay_mac_address;
int size_route;
int seg_left;
int answer;

// extract some useful information from the forward reply
packet
op_pk_nfd_get (fw_rep_pk_ptr,"SRC",&source_address);
op_pk_nfd_get (fw_rep_pk_ptr,"DEST",&destination_address);
op_pk_nfd_get (fw_rep_pk_ptr,"RELAY",&relay_address);
op_pk_nfd_get (fw_rep_pk_ptr,"Seq_number",&r_sequence_number);
op_pk_nfd_get (fw_rep_pk_ptr,"Size_Route",&size_route);
op_pk_nfd_get (fw_rep_pk_ptr,"Seg_Left",&seg_left);
op_pk_nfd_get (fw_rep_pk_ptr,"answer",&answer);

if(source_address != my_dsr_address)
  {
  // if the forward reply packet is destinated to the current
node
```

```
  if (my_dsr_address==destination_address &&
destination_address == relay_address)
        {
        // if the answer is positive
        if(answer==1)
              {
                    my_req_handled_for_me++;


      op_stat_write(stat_my_req_handled_for_me,my_req_handled_f
      or_me);
              my_fw_answer=answer;
              printf("Node %d:The answer is 1, I can start
              transmission\n",my_dsr_address);

              // search the connected data list to find out to
      which data packet this
              forward reply is referred
              current_ptr = head_ptr;
              while(current_ptr!=OPC_NIL)
                    {
                    my_temp_seq_number = current_ptr-
>sequence_number;

                    // if the data packet is found
                    if(my_temp_seq_number == r_sequence_number)
                      {
                            my_temp_dsr_address = current_ptr-
>destination_address;
                     my_temp_data_pk_ptr = current_ptr->pk;


dsr_transmit_data(my_temp_data_pk_ptr,my_temp_dsr_address);
                     break;
                     }

                     current_ptr = current_ptr->next;
                   }
             }
          // if the answer is negative
            else
                  {
              my_fw_answer=answer;
              printf("Node %d:The answer is 0, I can NOT start
              transmission\n",my_dsr_address);

              // search the connected data list to find out to
which data packet this
              forward reply is refered
              current_ptr = head_ptr;
              while(current_ptr!=OPC_NIL)
                    {
                    my_temp_seq_number = current_ptr-
>sequence_number;

                    // if the data packet is found
```

```
                    if(my_temp_seq_number == r_sequence_number)
                      {
                       my_temp_dsr_address = current_ptr-
>destination_address;
                    my_temp_data_pk_ptr = current_ptr->pk;

                    dsr_transmit_data(my_temp_data_pk_ptr,my_temp_
                  dsr_address);
                      break;
                      }

                     current_ptr = current_ptr->next;
                     }

              my_req_negative_answers_i_received++;

op_stat_write(stat_my_req_negative_answers_i_received,
              my_req_negative_answers_i_received);
              }
      }
  // if the node is the relay of the forward reply packet
  else if ((my_dsr_address==relay_address) &&
(my_dsr_address!=destination_address))
              {


      // if the answer is positive
      if(answer==1)
              {
              // increment the counter
"forward_requests_handled_for_others"
              my_req_handled_for_others++;
              op_stat_write(stat_my_req_handled_for_others,
              my_req_handled_for_others);

              // and then forward the forward reply
              printf("Node %d:Entering forward_forward_reply
function with positive
              answer\n",my_dsr_address);
              forward_forward_reply(fw_rep_pk_ptr);
              printf("Node %d:Exiting forward_forward_reply
function with positive
              answer\n",my_dsr_address);
              }

      // if the answer is negative
      else
              {
              // just forward the forward reply
              printf("Node %d:Entering forward_forward_reply
function with negative
              answer\n",my_dsr_address);
              forward_forward_reply(fw_rep_pk_ptr);
              printf("Node %d:Exiting forward_forward_reply
function with negative
              answer\n",my_dsr_address);
```

```
            }
  }
  else
        {
        printf("Node %d:I am not concerned by this reply
packet\n",my_dsr_address);
        op_pk_destroy(fw_rep_pk_ptr);
        }
  }
else
  {
  printf("Node %d:I am not concerned by this reply
packet\n",my_dsr_address);
  op_pk_destroy(fw_rep_pk_ptr);
  }

// breakpoints for debugging purpose
if (1)
  {
  op_prg_odb_bkpt("handle_forward_reply");
  }
}



/*****************************************************************
*******************************/
/*                  FORWARD_FORWARD_REPLY
*/
/*****************************************************************
*******************************/
/* this function forwards the incoming forward reply packets.
/* Packet* pk_ptr: a pointer to the forward reply packet to
process
/*****************************************************************
*******************************/

void forward_forward_reply(Packet* fw_rep_pk_ptr)
{
int size_route;
int seg_left;
int next_relay_dsr_address;
int next_relay_mac_address;
char field[10];

// read the seg_left and the size_route field in the packet
op_pk_nfd_get (fw_rep_pk_ptr,"Seg_Left",&seg_left);
op_pk_nfd_get (fw_rep_pk_ptr,"Size_Route",&size_route);

// extract the next node dsr address from the packet (node
size_route-seg_left)
sprintf(field,"Node_%d",size_route-seg_left);
op_pk_nfd_get(fw_rep_pk_ptr,field,&next_relay_dsr_address);

// and set it as the next packet relay
op_pk_nfd_set(fw_rep_pk_ptr,"RELAY",next_relay_dsr_address);
```

```
// update the seg_left field of the packet
op_pk_nfd_set(fw_rep_pk_ptr,"Seg_Left",--seg_left);

// set the TR_source field for the Transmission Range purpose
op_pk_nfd_set(fw_rep_pk_ptr,"TR_source",my_node_objid);

// convert the dsr address of the next relay in its mac
address
next_relay_mac_address=dsr_support_get_mac_from_dsr_address(next
_relay_dsr_address);

// and send the packet
dsr_send_to_mac(fw_rep_pk_ptr,next_relay_mac_address);

printf("Node %d:FORWARD_FORWARD_REPLY:I am forwarding the
forward reply packet to
%d\n",my_dsr_address,next_relay_dsr_address);

// breakpoint for debugging purpose
if (1)
  {
  op_prg_odb_bkpt("forward_forward_reply");
  }
}
```