ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και Ανάλυση Επίδοσης Πρωτοκόλλου Δρομολόγησης βάσει Ενεργειακών Κριτηρίων σε Δίκτυα MANET

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΡΗΣΤΟΣ ΧΡΙΣΤΟΔΟΥΛΟΣ ΤΣΑΚΙΡΟΓΛΟΥ

Επιβλέπων : Ιάκωβος Στ. Βενιέρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2011

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και Ανάλυση Επίδοσης Πρωτοκόλλου Δρομολόγησης βάσει Ενεργειακών Κριτηρίων σε Δίκτυα MANET

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΡΗΣΤΟΣ ΧΡΙΣΤΟΔΟΥΛΟΣ ΤΣΑΚΙΡΟΓΛΟΥ

Επιβλέπων : Ιάκωβος Στ. Βενιέρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Μαρτίου 2011.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Ιάκωβος Βενιέρης Καθηγητής Ε.Μ.Π.	Δήμητρα-Θεοδώρα Κακλαμάνη Καθηγήτρια Ε.Μ.Π.	Νικόλαος Ουζούνογλου Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2011

.....

ΧΡΗΣΤΟΣ ΧΡΙΣΤΟΔΟΥΛΟΣ Α. ΤΣΑΚΙΡΟΓΛΟΥ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Οι ραγδαίες εξελίξεις στα συστήματα των ασύρματων τηλεπικοινωνιών και η συνεχώς αυξανόμενη κινητικότητα των χρηστών έχουν οδηγήσει στην ανάπτυξη των κινητών αδόμητων δικτύων (Mobile Ad hoc Networks – MANET), που αποτελούν ένα από τα πιο δραστήρια ερευνητικά πεδία στο χώρο των δικτύων επικοινωνιών. Τα συγκεκριμένα δίκτυα δεν απαιτούν την ύπαρξη δικτυακής υποδομής καθώς είναι αυτοοργανούμενα, αυτορυθμιζόμενα και αυτοελεγχόμενα, ενώ συντίθενται από κινητές συσκευές όπως φορητοί υπολογιστές, κινητά τηλέφωνα, PDAs κτλ. Μια σημαντική πρόκληση που πρέπει να αντιμετωπιστεί είναι οι περιορισμένοι ενεργειακοί πόροι των κόμβων, οι οποίοι λειτουργούν με μπαταρία.

Ο σκοπός της διπλωματικής εργασίας είναι η μελέτη των δικτύων MANET υπό το πρίσμα της ενεργειακής κατανάλωσης και η σχεδίαση ενός ενεργειακά αποδοτικού πρωτοκόλλου δρομολόγησης. Στα πλαίσια της εργασίας, αρχικά παρουσιάζονται τα ενεργειακά θέματα καθώς και η δρομολόγηση στα δίκτυα MANET. Έπειτα, υλοποιείται ένα μοντέλο μπαταρίας, με βάση το οποίο αξιολογούνται οι ενεργειακές επιδόσεις των πρωτοκόλλων δρομολόγησης AODV (Ad hoc On-demand Distance Vector) και DSR (Dynamic Source Routing). Τελικά, σχεδιάζεται, υλοποιείται και αξιολογείται μια επέκταση του πρωτόκολλου DSR που δρομολογεί βάσει ενεργειακών κριτηρίων. Το νέο πρωτόκολλο ονομάζεται Energy-aware DSR (EDSR), λαμβάνει υπόψη του την υπολειπόμενη ενέργεια των κόμβων και στοχεύει στην αύξηση της διάρκειας ζωής του δικτύου.

Λέξεις – Κλειδιά

MANET, κινητά αδόμητα δίκτυα, μπαταρία, δρομολόγηση, ενεργειακά κριτήρια, υπολειπόμενη ενέργεια κόμβου, διάρκεια ζωής δικτύου, πρωτόκολλο DSR, επέκταση πρωτοκόλλου, OPNET Modeler.

Abstract

Rapid advances in wireless telecommunication systems and the continuously increasing user mobility have led to the development of Mobile Ad hoc Networks (MANETs), one of the most active research areas in the field of communication networks. MANETs don't require network infrastructure, as they are self-organized, self-configured and self-controlled, and they are composed of mobile devices, such as laptops, mobile phones, PDAs, etc. A major challenge that needs to be addressed is the limited energy resources of the battery powered nodes.

The scope of this thesis is to study MANETs from an energy perspective and to design an energy-efficient routing protocol. Firstly, the energy issues as well as routing in MANETs are presented. Next, a battery model is implemented and used to evaluate the energy consumption behavior of AODV (Ad hoc On-demand Distance Vector) and DSR (Dynamic Source Routing) routing protocols. Finally, an extension of the DSR protocol that enables energy-aware routing is designed, implemented and evaluated. The novel protocol is referred to as Energy-aware DSR (EDSR), takes into consideration the nodes' remaining energy and aims at prolonging network lifetime.

Keywords

MANET, mobile ad hoc networks, battery, routing, energy-aware metrics, node's remaining energy, network lifetime, DSR protocol, protocol extension, OPNET Modeler.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Ιάκωβο Βενιέρη, για τη δυνατότητα που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον αντικείμενο. Επίσης, θα ήθελα να ευχαριστήσω θερμά τον διδάκτορα κ. Ευάγγελο Κόσμάτο για την καθοδήγησή του και την πολύτιμη βοήθεια που μου προσέφερε καθ' όλη τη διάρκεια εκπόνησης της εργασίας. Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου για τη στήριξη που μου προσέφεραν σε κάθε περίοδο της ζωής μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

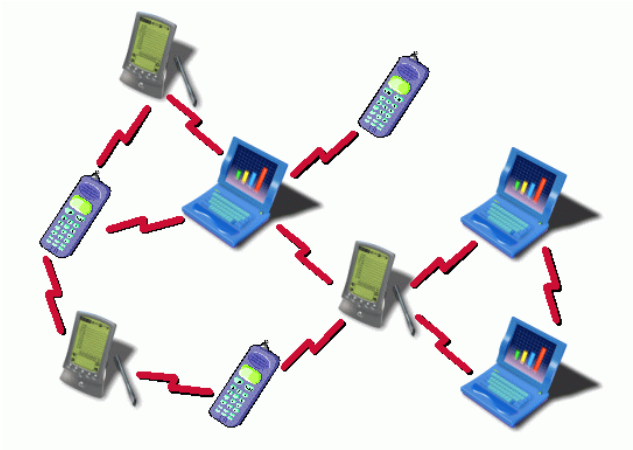
Κεφάλαιο 1

Εισαγωγή

Η πρόσφατη τεχνολογική πρόοδος στις ασύρματες επικοινωνίες και οι ολοένα και δημοφιλέστερες φορητές υπολογιστικές συσκευές τοποθετούν τα ασύρματα και τα κινητά αδόμητα (ad hoc) δίκτυα σε στρατηγική θέση για τους μελλοντικούς ιδιωτικούς και στρατιωτικούς χώρους όπου η ασύρματη πρόσβαση στο ενσύρματο δίκτυο κορμού είναι είτε αναποτελεσματική είτε αδύνατη. Τα κινητά αδόμητα δίκτυα (Mobile Ad hoc NETworks – MANET) αποτελούνται από ένα σύνολο σταθμών (κόμβων) που επικοινωνούν ασύρματα, χωρίς καμία σταθερή υποδομή. Είναι αυτοοργανούμενα, αυτορυθμιζόμενα και αυτοελεγχόμενα δίκτυα, που μπορούν να δημιουργηθούν και να λειτουργήσουν οπουδήποτε οποτεδήποτε, καθώς απαιτούν απλή ρύθμιση υποδομής, ενώ δεν απαιτούν κεντρική διαχείριση. Χρησιμοποιούνται κυρίως από χρήστες κοινοτήτων όπως οι ένοπλες δυνάμεις, ερευνητές, επιχειρηματίες, φοιτητές και από υπηρεσίες εκτάκτου ανάγκης.

1.1 Χαρακτηριστικά των δικτύων MANET

Στα δίκτυα MANET η επικοινωνία μεταξύ των κόμβων γίνεται ασύρματα. Επειδή οι κόμβοι είναι κινητοί και πιθανόν να συνδεθούν ή να εγκαταλείψουν το δίκτυο, τα δίκτυα αυτά έχουν δυναμική τοπολογία. Οι κόμβοι που βρίσκονται εντός ακτίνας διάδοσης μεταξύ τους λέγονται γειτονικοί και μπορούν να επικοινωνήσουν άμεσα. Ωστόσο, όταν ένας κόμβος πρέπει να στείλει δεδομένα σε ένα μη γειτονικό του κόμβο, τα δεδομένα δρομολογούνται μέσω μιας ακολουθίας πολλαπλών βημάτων, στην οποία οι ενδιάμεσοι κόμβοι λειτουργούν ως δρομολογητές. Ένα παράδειγμα δικτύου MANET φαίνεται στο Σχήμα 1.1.



Σχήμα 1.1 : Παράδειγμα δικτύου MANET

Τα βασικότερα προβλήματα που αντιμετωπίζουν τα δίκτυα MANET είναι τα εξής [21]:

- **Απρόβλεπτο περιβάλλον** : Πιθανόν τα MANET να λειτουργήσουν σε άγνωστο έδαφος, επικίνδυνες συνθήκες και εχθρικό περιβάλλον, με επικείμενη τη δυσλειτουργία ή την καταστροφή των κόμβων.
- **Αναξιοπιστία του ασύρματου διαύλου :** Η επικοινωνία μέσω του ασύρματου διαύλου είναι αναξιόπιστη και επιρρεπής σε σφάλματα. Ακόμα, οι ηλεκτρομαγνητικές παρεμβολές, η κακοκαιρία, η αδυναμία υποστήριξης πρωτοκόλλου μεταφοράς λόγω περιορισμένων πόρων οδηγούν σε σημαντική διακύμανση της ποιότητας της ζεύξης.
- **Περιορισμένοι πόροι :** Οι κόμβοι ενός MANET λειτουργούν συνήθως με μπαταρία και έχουν περιορισμένες δυνατότητες αποθήκευσης και επεξεργασίας. Επιπλέον, μπορεί να είναι αδύνατη η επαναφόρτιση των μπαταριών στο περιβάλλον τους, οπότε έχουν και περιορισμένη διάρκεια ζωής. Έτσι, οι αλγόριθμοι πρέπει να είναι ενεργειακά αποδοτικοί και να έχουν μικρές απαιτήσεις σε επεξεργαστική ισχύ και μνήμη. Λόγω του ενεργειακού περιορισμού πιθανόν να προκύπτει και περιορισμός στο διαθέσιμο εύρος ζώνης.
- **Δυναμική τοπολογία :** Η τοπολογία ενός MANET πιθανόν να μεταβάλλεται συνεχώς λόγω της κινητικότητας των κόμβων. Καθώς οι κόμβοι κινούνται εντός και εκτός της ακτίνας μετάδοσης άλλων κόμβων, κάποιες ζεύξεις διακόπτονται και κάποιες άλλες δημιουργούνται.

Λόγω των προβλημάτων αυτών, τα δίκτυα MANET είναι επιρρεπή σε σφάλματα όπως :

- **Σφάλματα μετάδοσης** : Η αναξιοπιστία του ασύρματου διαύλου και το απρόβλεπτο περιβάλλον αλλοιώνουν τα μεταδιδόμενα πακέτα, τα οποία λαμβάνονται εσφαλμένα.
- **Διακοπή λειτουργίας κόμβων** : Οι κόμβοι πιθανόν να σταματήσουν να λειτουργούν λόγω διαφόρων επικίνδυνων συνθηκών του περιβάλλοντος. Ακόμα, μπορεί να αποσυνδεθούν από το δίκτυο εκουσίως ή λόγω ενεργειακής εξάντλησης.
- **Διακοπή ζεύξεων** : Οι αποσυνδέσεις κόμβων και οι μεταβαλλόμενες περιβαλλοντικές συνθήκες πιθανόν να διακόψουν τις ζεύξεις μεταξύ κόμβων.
- **Διακοπή διαδρομών** : Λόγω της δυναμικής τοπολογίας του δικτύου κάποιες διαδρομές καταλήγουν απαρχαιωμένες, δηλαδή εσφαλμένες. Έτσι, εξαρτάται πλέον από το πρωτόκολλο μεταφοράς αν τα πακέτα που δρομολογήθηκαν μέσω των διαδρομών αυτών θα απορριφθούν ή θα παραδοθούν με καθυστέρηση.
- **Κόμβοι ή ζεύξεις με συμφόρηση :** Λόγω της τοπολογίας του δικτύου και της φύσης του πρωτοκόλλου δρομολόγησης, κάποιοι κόμβοι ή ζεύξεις μπορεί να υπερφορτώνονται. Αυτό θα οδηγήσει σε υψηλές καθυστερήσεις ή απώλειες πακέτων και σε άδικη ενεργειακή εξάντληση κάποιων κόμβων.

1.2 Εφαρμογές των δικτύων MANET

Το μεγάλο εύρος των εφαρμογών που υποστηρίζουν τα δίκτυα MANET εκτείνεται από περιβάλλοντα γραφείων μικρής επιχείρησης μέχρι μεγάλης κλίμακας δίκτυα αισθητήρων. Μια κατηγοριοποίηση των εφαρμογών των δικτύων MANET είναι η ακόλουθη [22] :

- **Δίκτυα αισθητήρων** : Αποτελούνται από κατανεμημένους στο χώρο αυτόνομους αισθητήρες που συνεργάζονται για την παρουσίαση φυσικών ή περιβαλλοντικών συνθηκών, όπως για παράδειγμα της θερμοκρασίας, του ήχου, των δονήσεων, της πίεσης, της κίνησης, των ρύπων κτλ. Η ανάπτυξη των ασύρματων δικτύων αισθητήρων ξεκίνησε για στρατιωτικές εφαρμογές, όπως η επιτήρηση σε πεδία μαχών, αλλά επεκτάθηκε σε αρκετούς τομείς της καθημερινότητας, όπως η παρακολούθηση και ο έλεγχος της παραγωγικής διαδικασίας στα εργοστάσια, η παρακολούθηση του περιβάλλοντος, αυτοματοποίηση των οικιακών διαδικασιών καθώς και εφαρμογές υγείας. Για ιατρικές και αθλητικές εφαρμογές έχουν αναπτυχθεί τα Body Area Networks (BAN), ενώ για εφαρμογές παρακολούθησης ασθενών στο σπίτι τα Home Area Networks (HAN).
- **Προσωπική χρήση** : Αφορά τη μετάδοση μη εμπορικών δεδομένων ανάμεσα σε συσκευές και χρήστες. Ένα μικρό ad hoc δίκτυο απλοποιεί τη διασύνδεση μεταξύ φορητών συσκευών όπως οι φορητοί υπολογιστές, τα κινητά τηλέφωνα, τα PDAs κτλ. Είναι δυνατό να γίνονται κλήσεις μέσω του τηλεφώνου και, αν υπάρχει πρόσβαση στο διαδίκτυο, οι κλήσεις αυτές να γίνονται μέσω VoIP χωρίς την παρεμβολή του χρήστη.
- **Εμπορική χρήση** : Αφορά τη δημιουργία δικτύου επικοινωνίας σε περιπτώσεις εκθέσεων, συνεδρίων ή παρουσιάσεων. Πιο συγκεκριμένα, φορητοί υπολογιστές ή PDAs μπορούν να συνδεθούν άμεσα για τα δημιουργία ενός τοπικού ad hoc δικτύου πολυμέσων. Γενικότερα, σε οποιαδήποτε περίπτωση συναθροίζονται πολλοί άνθρωποι για μικρό χρονικό διάστημα είναι δυνατή η εύκολη ανταλλαγή πληροφοριών μεταξύ τους χωρίς την ανάγκη ανάπτυξης υποδομής που μπορεί να είναι οικονομικά ασύμφορη.
- **Δίκτυα αυτοκινήτων (Vehicular Ad hoc NETworks – VANET) :** Είναι δίκτυα ανάμεσα σε αυτοκίνητα που κινούνται και σε δίκτυα σταθερής υποδομής που καθορίζουν την κυκλοφορία, όπως τα φανάρια ή τα ηλεκτρονικά σήματα. Έτσι, ο οδηγός μπορεί να γνωρίζει για την κυκλοφοριακή συμφόρηση αλλά και να ενημερώνεται για σημεία ενδιαφέροντος όπως βενζινάδικα ή συνεργεία.
- **Καταστάσεις εκτάκτου ανάγκης** : Αφορούν κυρίως επιχειρήσεις αναζήτησης και διάσωσης. Πιο συγκεκριμένα, χρησιμοποιούνται σε περιοχές χωρίς ικανοποιητική ασύρματη κάλυψη ή σε περίπτωση αποτυχίας της υπάρχουσας υποδομής επικοινωνιών. Οι καταστάσεις εκτάκτου ανάγκης προκαλούνται συνήθως από φυσικές καταστροφές, όπως σεισμοί και πυρκαγιές, και απαιτούν την άμεση ανάπτυξη ενός δικτύου επικοινωνιών. Η

επικοινωνία μπορεί να γίνεται με μικρές συσκευές ακόμα και σε μέρη όπου η επικοινωνία μέσω ενός σταθμού βάσης είναι αδύνατη. Άλλες εφαρμογές περιλαμβάνουν επικοινωνίες μεταξύ πλοίων και μεταξύ δυνάμεων τήρησης της τάξης ή στρατιωτικών δυνάμεων.

1.3 Σκοπός της εργασίας

Ο σκοπός της εν λόγω διπλωματικής εργασίας είναι η μελέτη των δικτύων MANET υπό το πρίσμα της ενεργειακής κατανάλωσης και η επέκταση ενός πρωτοκόλλου δρομολόγησης, ώστε να βελτιώνονται οι ενεργειακές επιδόσεις του δικτύου. Στα πλαίσια της εργασίας, αρχικά παρουσιάζονται τα ενεργειακά θέματα και η δρομολόγηση στα δίκτυα MANET. Έπειτα, υλοποιείται ένα μοντέλο μπαταρίας, με βάση το οποίο αξιολογούνται οι ενεργειακές επιδόσεις των πρωτοκόλλων δρομολόγησης AODV και DSR. Τελικά, επεκτείνεται το πρωτόκολλο DSR χρησιμοποιώντας το μοντέλο της μπαταρίας, ώστε να δρομολογεί βάσει ενεργειακών κριτηρίων. Το νέο πρωτόκολλο ονομάζεται Energy-aware DSR (EDSR), λαμβάνει υπόψη του την υπολειπόμενη ενέργεια των κόμβων και στοχεύει στην αύξηση της διάρκειας ζωής του δικτύου.

1.4 Οργάνωση της εργασίας

Η διπλωματική εργασία είναι οργανωμένη σε πέντε κεφάλαια :

- **Κεφάλαιο 2**: Παρουσιάζεται το στρώμα ζεύξης και το φυσικό στρώμα των MANET, δηλαδή το πρότυπο IEEE 802.11. Έπειτα, παρουσιάζονται τα ενεργειακά θέματα μοντελοποίησης της μπαταρίας και των απωλειών ενέργειας.
- **Κεφάλαιο 3**: Παρουσιάζεται το στρώμα δικτύου των MANET και γίνεται μια επισκόπηση των πρωτοκόλλων δρομολόγησης, με διεξοδική ανάλυση του πρωτοκόλλου DSR. Έπειτα, παρουσιάζεται η δρομολόγηση με ενεργειακά κριτήρια και μια επισκόπηση των αντίστοιχων πρωτοκόλλων.
- **Κεφάλαιο 4**: Στο πρώτο μέρος του κεφαλαίου παρουσιάζεται η υλοποίηση του μοντέλου μπαταρίας και οι προσομοιώσεις με τα αποτελέσματά τους. Στο δεύτερο μέρος παρουσιάζεται η υλοποίηση του πρωτοκόλλου δρομολόγησης EDSR (Energy-aware DSR), καθώς και οι προσομοιώσεις με τα αποτελέσματά τους.
- **Κεφάλαιο 5**: Παρουσιάζονται τα συμπεράσματα και προτάσεις για μελλοντική εργασία.

Κεφάλαιο 2

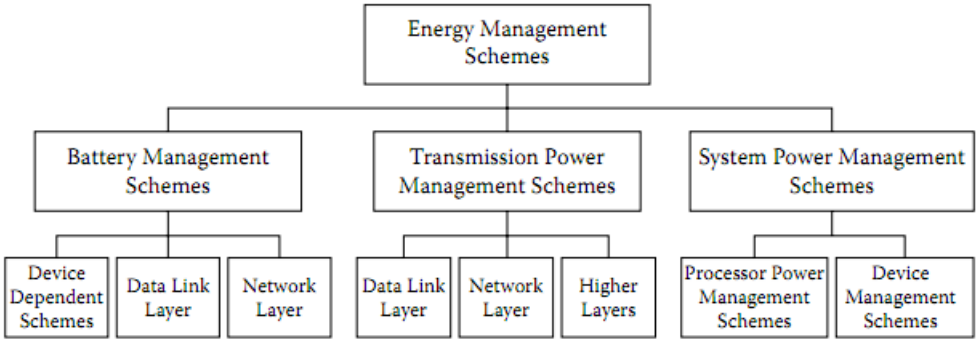
Στρώμα Ζεύξης και Ενεργειακά Θέματα

2.1 Εισαγωγή

Τα δίκτυα MANET περιορίζονται από την πεπερασμένη ενέργεια της μπαταρίας των συσκευών. Έτσι, η διαχείριση ενέργειας αποτελεί σημαντική λειτουργία που ασχολείται με την αποδοτική διαχείριση της μπαταρίας αλλά και της ισχύος εκπομπής. Οι κύριοι λόγοι που οδηγούν στη διαχείριση ενέργειας στα MANET είναι [17] :

- **Περιορισμένο ενεργειακό απόθεμα** : Η βελτίωση στις τεχνολογίες μπαταρίας είναι πολύ αργή σε σχέση με την πρόοδο στα πεδία των κινητών υπολογιστών και επικοινωνιών.
- **Δυσκολία στην αντικατάσταση μπαταριών** : Σε περιπτώσεις έκτακτης ανάγκης, όπως πολέμου ή φυσικών καταστροφών, είναι δύσκολη η αντικατάσταση και επαναφόρτιση των μπαταριών.
- **Απουσία κεντρικού συντονισμού** : Επειδή το ad hoc δίκτυο είναι κατανεμημένο, κάποιοι κόμβοι πρέπει να λειτουργούν ως αναμεταδότες, καταναλώνοντας έτσι μεγαλύτερη ενέργεια.
- **Περιορισμοί στο πρότυπο μπαταρίας** : Το βάρος των κόμβων πιθανόν να αυξάνεται με την αύξηση του βάρους της μπαταρίας. Έτσι, μειώνοντας το βάρος της, μειώνεται και η χωρητικότητά της άρα και η διάρκεια ζωής της, οπότε πρέπει να αξιοποιούνται βέλτιστα οι μειωμένοι ενεργειακοί πόροι.
- **Επιλογή βέλτιστης ισχύος εκπομπής** : Η αύξηση της ισχύος εκπομπής οδηγεί σε αύξηση της ενεργειακής κατανάλωσης. Επειδή η ισχύς εκπομπής επηρεάζει την προσιτότητα των κόμβων, η βέλτιστη επιλογή της μειώνει τις παρεμβολές, αυξάνοντας τον αριθμό των ταυτόχρονων μεταδόσεων.

Μια κατηγοριοποίηση των μηχανισμών διαχείρισης ενέργειας φαίνεται στο Σχήμα 2.1. Οι μηχανισμοί χωρίζονται σε τρεις βασικές κατηγορίες : Διαχείρισης μπαταρίας, διαχείριση ισχύος εκπομπής, διαχείριση ισχύος συστήματος [17]. Στην παρούσα εργασία θα ασχοληθούμε με τη διαχείριση της μπαταρίας και μάλιστα στο στρώμα δικτύου, δηλαδή με τον αλγόριθμο δρομολόγησης.



Σχήμα 2.1 : Κατηγοριοποίηση μηχανισμών διαχείρισης ενέργειας στα MANET

Η εκφόρτιση της μπαταρίας είναι άμεσα συνδεδεμένη με την αποστολή και λήψη πακέτων. Έτσι, για την κατανόηση της συμπεριφοράς της μπαταρίας ενός κόμβου είναι σκόπιμο να μελετήσουμε τη λειτουργία των δικτύων MANET στα κατώτερα στρώματα. Η απουσία κεντρικής οντότητας συνεπάγεται τη χρήση κατανεμημένου πρωτοκόλλου πρόσβασης στο ασύρματο μέσο. Τα ασύγχρονα πρωτόκολλα MAC είναι λοιπόν περισσότερο ελκυστικά για την περίπτωση αυτή, λόγω της κατανεμημένης λειτουργίας τους και της ευρωστίας τους. Τα περισσότερα ασύρματα τοπικά δίκτυα (WLANs) σήμερα χρησιμοποιούν το πρότυπο IEEE 802.11 [16].

Στο κεφάλαιο αυτό παρουσιάζεται συνοπτικά η λειτουργία των κατώτερων στρωμάτων των δικτύων MANET, δηλαδή του φυσικού στρώματος και του στρώματος ζεύξης δεδομένων, που λειτουργούν με βάση το πρότυπο IEEE 802.11. Έπειτα, συσχετίζεται η λειτουργία του IEEE 802.11 με την ενεργειακή κατανάλωση του κόμβου, αναλύοντας τη μοντελοποίηση της μπαταρίας αλλά και των απωλειών ενέργειας.

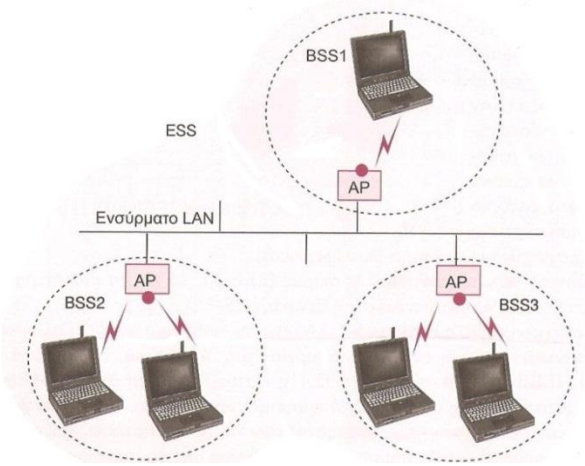
2.2 Πρότυπο IEEE 802.11/WiFi

Το WiFi είναι μια διαδεδομένη ασύρματη τεχνολογία που απευθύνεται σε οικιακούς χρήστες, μικρές επιχειρήσεις και μικρούς παρόχους διαδικτύου (Internet Service Providers – ISPs) βασισμένη στο πρότυπο IEEE 802.11. Η IEEE αναπτύσσει και δημοσιεύει τα πρότυπα αυτά χωρίς να ελέγχει τον εξοπλισμό για συμμόρφωση με αυτά. Η μη κερδοσκοπική Wi-Fi Alliance εγκαθιστά και επιβάλλει πρότυπα για τη διαλειτουργικότητα και συμβατότητα προς τα πίσω, καθώς προωθεί και την τεχνολογία ασύρματων τοπικών δικτύων.

Υπάρχουν αρκετές εκδόσεις των προτύπων IEEE 802.11 και τα κυριότερα χαρακτηριστικά της παρατίθενται στον Πίνακα 2.1. Όλες οι εκδόσεις υποστηρίζουν το ίδιο στρώμα MAC, το οποίο χρησιμοποιεί πολλαπλή πρόσβαση με ανίχνευση φέροντος και αποφυγή συγκρούσεων (CSMA/CA). Διακρίνονται δύο τοπολογίες δικτύου, η τοπολογία με δίκτυο υποδομής και η ad-hoc τοπολογία, που μας ενδιαφέρει για τα δίκτυα MANET.

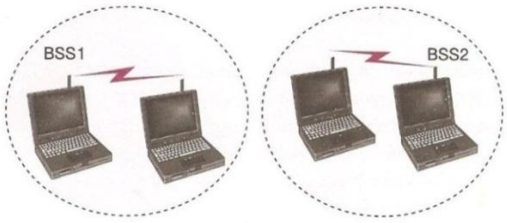
Στην τοπολογία με *δίκτυο υποδομής* τα κινητά τερματικά επικοινωνούν με το δίκτυο κορμού μέσω ενός σημείου πρόσβασης (Access Point – AP). Το AP είναι μια γέφυρα

που συνδέει το δίκτυο 802.11 με την υποδομή του ενσύρματου δικτύου κορμού. Το σύνολο των κινητών σταθμών που ελέγχονται από ένα AP αποτελεί τη βασική ομάδα εξυπηρέτησης (Basic Service Set – BSS). Πολλές BSS συνδεόμενες με κοινό δίκτυο κορμού σχηματίζουν μια ενιαία υποδομή που ονομάζεται εκτεταμένη ομάδα εξυπηρέτησης (Extended Service Set – ESS). Ένα κινητικό τερματικό μπορεί να περιφέρεται σε διαφορετικές BSS μιας ESS χωρίς να χάνει τη σύνδεσή του με το δίκτυο κορμού. Στην τυπική εφαρμογή αυτής της τοπολογίας, μια ομάδα από φορητούς υπολογιστές συνδέεται μέσω ενός WLAN σε ένα ενσύρματο LAN κορμού, όπως φαίνεται στο Σχήμα 2.2 [15].



Σχήμα 2.2 : Τοπολογία IEEE 802.11 με δίκτυο υποδομής

Στην τοπολογία *ad-hoc*, τα κινητά τερματικά επικοινωνούν μεταξύ τους σε ανεξάρτητη BSS χωρίς σύνδεση προς το ενσύρματο δίκτυο κορμού, όπως φαίνεται στο Σχήμα 2.3. Στην περίπτωση αυτή, μερικές από τις λειτουργίες του AP που χρειάζονται για τον σχηματισμό και τη διατήρηση μιας BSS παρέχονται από ένα κινητό τερματικό. Επιπρόσθετα με τη μέθοδο πολλαπλής πρόσβασης CSMA/CA, που είναι κατάλληλη για ασύγχρονες μεταδόσεις, το IEEE 802.11 παρέχει και έναν μηχανισμό με προτεραιότητες χωρίς ανταγωνισμό, ελεγχόμενο από ένα σημείο, για την υποστήριξη ισόχρονων εφαρμογών με χρονικούς περιορισμούς. Για την αντιμετώπιση του προβλήματος των κρυμμένων τερματικών το πρωτόκολλο MAC διαθέτει μηχανισμό με μηνύματα Request-To-Send/Clear-To-Send (RTS/CTS), που όμως παρέχει μικρή εξασφάλιση για την ποιότητα υπηρεσίας. Ακόμα, υποστηρίζονται μηχανισμοί για έλεγχο αυθεντικότητας, απόκρυψη, διαχείριση συχνοτήτων και εξοικονόμηση ισχύος.



Σχήμα 2.3 : Τοπολογία IEEE 802.11 ad hoc

Πρότυπο	Έκδοση	Συχνότητα (GHz)	Εύρος Ζώνης (MHz)	Μέγιστες ροές MIMO	Ρυθμός Μετάδοσης στο Φυσικό Στρώμα ανά ροή (Mbps)	Διαμόρφωση
802.11	Ιουν-97	2.4	20	1	1, 2	DSSS,FHSS
802.11a	Σεπ-99	5	20	1	6, 9, 12, 18, 24, 36, 48, 54	OFDM
				1		
802.11b	Σεπ-99	2.4	20	1	5.5, 11	DSSS
802.11g	Ιουν-03	2.4	20	1	6, 9, 12, 18, 24, 36, 48, 54	OFDM,DSSS
802.11n	Οκτ-09	2.4/5	20	1	7.2, 14.4, 21.7, 28.9, 43.3, 57.8, 65, 72.2	OFDM
			40	4	15, 30, 45, 60, 90, 120, 135, 150	

Πίνακας 2.1 : Πρότυπα WLAN της σειράς IEEE 802.11

Τα βασικά χαρακτηριστικά του *φυσικού στρώματος* του IEEE 802.11 παρουσιάζονται στον Πίνακα 2.1. Παρατηρούμε ότι ο ολοένα αυξανόμενος ρυθμός μετάδοσης ανοίγει το δρόμο για τις πιο απαιτητικές υπηρεσίες, όπως το κινητό εμπόριο, οι υπηρεσίες θέσης, οι υπηρεσίες πολυμέσων κτλ. Η πρώτη επιλογή λειτουργίας των WLAN ως προς το φάσμα είναι οι ζώνες ISM (Industrial, Scientific and Medical), για τις οποίες δεν απαιτείται άδεια λειτουργίας. Η επιλογή της ζώνης λειτουργίας στο φάσμα συχνοτήτων επηρεάζει τις απαιτήσεις ισχύος και τη ραδιοκάλυψη. Για παράδειγμα, τα δίκτυα IEEE 802.11a θα απαιτούν αρκετά μεγαλύτερη ισχύ, λόγω της υψηλής συχνότητας λειτουργίας. Όσον αφορά τις διαμορφώσεις, χρησιμοποιούνται το απλωμένο φάσμα ευθείας ακολουθίας (Direct-Sequence Spread Spectrum – DSSS), το απλωμένο φάσμα με μεταπήδηση συχνότητας (Frequency-Hopping Spread Spectrum – FHSS), η ορθογωνική πολυπλεξία διαίρεσης συχνότητας (Orthogonal Frequency-Division Multiplexing – OFDM), αλλά και η υπέρυθρη ακτινοβολία. Για εις βάθος ανάλυση των τεχνικών διαμόρφωσης προτείνεται το πρότυπο IEEE 802.11 [16].

Το *στρώμα MAC* του IEEE 802.11 υποδιαιρείται στο καθαυτό υπόστρωμα MAC και στο υπόστρωμα διαχείρισης MAC. Το υπόστρωμα MAC είναι υπεύθυνο για τις διαδικασίες εκχώρησης διαύλων, τη διευθυνσιοδότηση των μονάδων δεδομένων πρωτοκόλλου (Protocol Data Unit – PDU), τον σχηματισμό των πλαισίων, τον έλεγχο λαθών, τον τεμαχισμό και την επανασυναρμολόγηση των πακέτων. Το υπόστρωμα διαχείρισης MAC είναι υπεύθυνο για την περιαγωγή στην ESS, τον έλεγχο ισχύος και τις διαδικασίες συσχέτισης, αποσυσχέτισης και επανασυσχέτισης κατά τη διαχείριση των συνδέσεων. Υποστηρίζονται δύο διαφορετικά σχήματα πρόσβασης, για την εξυπηρέτηση ασύγχρονων υπηρεσιών και υπηρεσιών με χρονικούς περιορισμούς :

- **Λειτουργία Κατανεμημένου Συντονισμού (Distributed Coordination Function – DCF)** : Υποστηρίζει παράδοση των δεδομένων με την καλύτερη δυνατή προσπάθεια. Έχει σχεδιαστεί για την ασύγχρονη μετάδοση δεδομένων,

όπου όλοι οι σταθμοί που έχουν να μεταδώσουν δεδομένα έχουν ίσες ευκαιρίες για πρόσβαση στο δίκτυο.

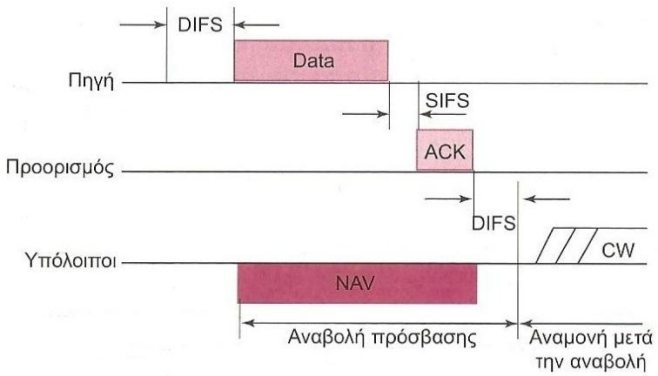
- **Λειτουργία Συντονισμού από ένα Σημείο (Point Coordination Function – PCF)** : Βασίζεται σε σύστημα polling που ελέγχεται από ένα AP και σχεδιάστηκε κυρίως για μεταφορά κίνησης που είναι ευαίσθητη σε καθυστερήσεις. Μπορεί επίσης να μεταπίπτει από τον τρόπο λειτουργίας με ανταγωνισμό, γνωστό και ως περίοδο ανταγωνισμού (Contention Period – CP), στην περίοδο λειτουργίας χωρίς ανταγωνισμό (Contention-Free Period – CFP). Στη λειτουργία CFP η χρησιμοποίηση του μέσου μετάδοσης ελέγχεται από το AP, καταργώντας έτσι την ανάγκη να ανταγωνίζονται οι σταθμοί για την πρόσβαση στο δίαυλο.

2.2.1 Λειτουργία κατανεμημένου συντονισμού (DCF)

Στα ad-hoc δίκτυα υπάρχει μόνο η *DCF λειτουργία*. Βασίζεται στην τεχνική πολλαπλής πρόσβασης με ανίχνευση φέροντος και αποφυγή συγκρούσεων CSMA/CA, αφού η τεχνική ανίχνευσης συγκρούσεων CSMA/CD δεν είναι δυνατή, καθώς οι σταθμοί δεν μπορούν να ακούν το δίαυλο για την ανίχνευση συγκρούσεων την ώρα που μεταδίδουν. Η ανίχνευση φέροντος στο IEEE 802.11 γίνεται και στην ασύρματη διεπαφή, αναφερόμενη ως φυσική ανίχνευση φέροντος, και στο υπόστρωμα MAC, αναφερόμενη ως νοητή ανίχνευση φέροντος. Στην ασύρματη διεπαφή, η παρουσία άλλων χρηστών διαπιστώνεται με την ανάλυση όλων των λαμβανόμενων πακέτων και μέσω της σχετικής έντασης του σήματος που προέρχεται από άλλες πηγές. Στο υπόστρωμα MAC, η νοητή ανίχνευση φέροντος πραγματοποιείται στέλνοντας πληροφορία για τη διάρκεια του πακέτου στην επικεφαλίδα των πλαισίων RTS/CTS και των πλαισίων δεδομένων. Ο δίαυλος χαρακτηρίζεται ως κατειλημμένος, όταν τουλάχιστον μία από τις τεχνικές δείξει ότι ο δίαυλος είναι κατειλημμένος.

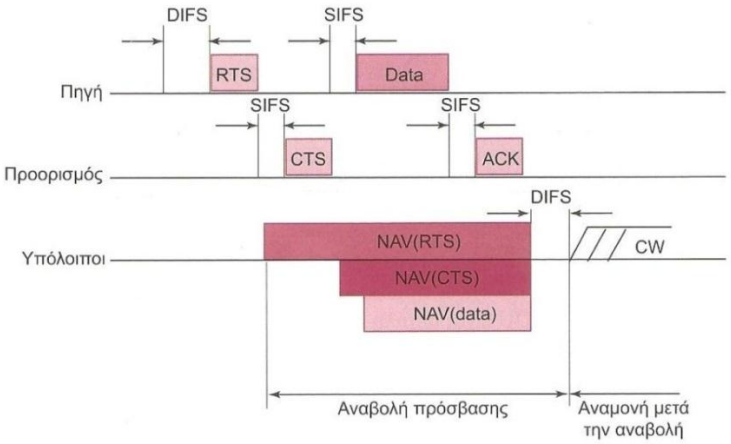
Στη μέθοδο βασικής πρόσβασης, όταν ένας σταθμός αντιληφθεί ότι ο δίαυλος είναι ελεύθερος, περιμένει περίοδο DIFS (DCF Inter-Frame Spacing) και ελέγχει εκ νέου το δίαυλο. Αν ο δίαυλος είναι ακόμη ελεύθερος, ο σταθμός μεταδίδει ένα πλαίσιο. Ο σταθμός που λαμβάνει υπολογίζει το άθροισμα ελέγχου και αποφασίζει κατά πόσο το πακέτο λήφθηκε σωστά ή όχι. Μετά τη λήψη ενός σωστού πακέτου, ο σταθμός λήψης περιμένει διάστημα SIFS (Short Inter-Frame Spacing) και κατόπιν μεταδίδει πλαίσιο θετικής επιβεβαίωσης (ACK) προς τον σταθμό αποστολής για να δείξει ότι η αποστολή ήταν επιτυχής. Το χρονοδιάγραμμα του Σχήματος 2.4 δείχνει την επιτυχή μετάδοση ενός πλαισίου. Όταν μεταδίδεται ένα πλαίσιο δεδομένων, το πεδίο διάρκειας του πλαισίου χρησιμοποιείται για να ενημερώσει τους σταθμούς της BSS για το πόσο χρόνο θα είναι κατειλημμένο το μέσο μετάδοσης. Όλοι οι σταθμοί που ακούν το πλαίσιο δεδομένων ρυθμίζουν το NAV (Network Allocation Vector) τους με βάση το περιεχόμενο πεδίου διαρκείας του πλαισίου, συμπεριλαμβάνοντας και ένα διάστημα SIFS καθώς και τον χρόνο μετάδοσης της επιβεβαίωσης που θα ακολουθήσει το πλαίσιο δεδομένων. Το NAV δείχνει τον χρόνο που πρέπει να

περάσει μέχρι να συμπληρωθεί η τρέχουσα σύνοδος μετάδοσης και να διαπιστωθεί ότι ο δίαυλος είναι πάλι ελεύθερος.



Σχήμα 2.4 : Μετάδοση ενός πλαισίου χωρίς τη χρησιμοποίηση RTS/CTS

Τα πλαίσια ελέγχου RTS και CTS μπορούν να χρησιμοποιηθούν από κάποιον σταθμό για να δεσμεύσει εύρος ζώνης πριν την αποστολή πλαισίου, ελαχιστοποιώντας το εύρος ζώνης που χάνεται λόγω συγκρούσεων, αφού έχουν μικρό μέγεθος. Όταν ένας σταθμός έχει να στείλει κάποιο πλαίσιο προς άλλο σταθμό, στέλνει πρώτα το πλαίσιο ελέγχου RTS. Όλοι οι σταθμοί που ακούνε το RTS διαβάζουν το πεδίο διαρκείας του και ρυθμίζουν κατάλληλα το NAV τους. Ο παραλήπτης σταθμός απαντά στο πλαίσιο RTS με ένα πλαίσιο CTS, αφού περιμένει να παρέλθει χρονικό διάστημα SIFS. Οι σταθμοί που ακούνε το πλαίσιο CTS βλέπουν το πεδίο διαρκείας του και ενημερώνουν και πάλι το NAV τους. Ύστερα από την επιτυχή λήψη του πλαισίου CTS ο σταθμός αποστολής επιβεβαιώνει νοητά ότι το μέσο είναι σταθερό και δεσμευμένο για την επιτυχή αποστολή του πλαισίου. Το χρονοδιάγραμμα του Σχήματος 2.5 δείχνει την επιτυχή μετάδοση ενός πλαισίου με το μηχανισμό RTS/CTS. Η χρησιμοποίηση του μηχανισμού RTS/CTS από τους σταθμούς είναι προαιρετική. Αν προκύψει σύγκρουση κατά τη μετάδοση ενός πλαισίου RTS ή CTS σπαταλιέται πολύ λιγότερο εύρος ζώνης απ' ότι όταν μεταδίδεται ένα μεγάλο πλαίσιο δεδομένων. Ωστόσο, για χαμηλό φορτίο η χρησιμοποίηση πλαισίων RTS/CTS εισάγει επιπρόσθετη καθυστέρηση.



Σχήμα 2.5 : Μετάδοση ενός πλαισίου με χρησιμοποίηση RTS/CTS

Η διαδικασία αποφυγής συγκρούσεων στην πρόσβαση CSMA/CA πραγματοποιείται με την τυχαία αναμονή. Αν κάποιος σταθμός που έχει πλαίσιο να στείλει ανιχνεύσει ότι ο δίαυλος είναι κατειλημμένος, περιμένει μέχρις ότου ο δίαυλος μείνει ελεύθερος. Μετά την απελευθέρωση του διαύλου περιμένει για περίοδο DIFS και στη συνέχεια υπολογίζει ένα τυχαίο χρόνο αναμονής. Σε περίπτωση σύγκρουσης γίνεται δυαδική εκθετική υποχώρηση.

2.3 Μοντελοποίηση μπαταρίας

Τα μοντέλα μπαταριών χρησιμοποιούνται για την πρόβλεψη της συμπεριφοράς τους κάτω από ποικίλες συνθήκες φόρτισης και εκφόρτισης. Είναι χρήσιμα εργαλεία για τη σχεδίαση συστημάτων με βάση τη μπαταρία, αφού επιτρέπουν την ανάλυση της συμπεριφοράς εκφόρτισης της μπαταρίας κάτω από διάφορες σχεδιαστικές επιλογές. Στη συνέχεια, κατηγοριοποιούμε τα μοντέλα σε τέσσερις κατηγορίες : αναλυτικά, ηλεκτρικά κυκλώματα, στοχαστικά, ηλεκτροχημικά [17].

2.3.1 Αναλυτικά μοντέλα

Είναι μοντέλα που διατυπώνουν αναλυτικές εκφράσεις για να υπολογίσουν τη χωρητικότητα της μπαταρίας και το χρόνο ζωής της, χρησιμοποιώντας ως παραμέτρους τις τιμές ρεύματος εκφόρτισης, τα χαρακτηριστικά του περιβάλλοντος λειτουργίας και τις φυσικές ιδιότητες της μπαταρίας. Ένα από τα πρώτα και απλούστερα μοντέλα είναι ο τύπος του Peukert, που εκφράζει τη μη γραμμική σχέση μεταξύ της χωρητικότητας της μπαταρίας και του ρυθμού εκφόρτισης. Ο τύπος του Peukert εκφράζει την πραγματική χωρητικότητα Q της μπαταρίας

$$Q = \frac{K}{I^k} \tag{2.1}$$

όπου I είναι το ρεύμα φορτίου (θεωρείται σταθερό), k είναι μια σταθερά που περιλαμβάνει τις ηλεκτροχημικές ιδιότητες, φυσική κατασκευή και περιβάλλον λειτουργίας της μπαταρίας, και K μια σταθερά που αφορά το φαινόμενο ρυθμού χωρητικότητας (rate capacity effect).

Πρόσφατα έχουν προταθεί εξεζητημένα αναλυτικά μοντέλα, όπου η χωρητικότητα εκφράζεται με πολυπλοκότερες συναρτήσεις του ρεύματος εκφόρτισης, της θερμοκρασίας και άλλων σταθερών. Τα μοντέλα αυτά αναφέρονται είτε σε σταθερό είτε σε μεταβλητό φορτίο. Είναι ευέλικτα και παραμετροποιούνται εύκολα για συγκεκριμένες μπαταρίες. Ακόμα, είναι υπολογιστικά αποδοτικά καθώς απαιτούν την αποτίμηση απλών αναλυτικών εκφράσεων.

2.3.2 Μοντέλα ηλεκτρικών κυκλωμάτων

Τα μοντέλα αυτά μοντελοποιούν την εκφόρτιση χρησιμοποιώντας ισοδύναμα ηλεκτρικά κυκλώματα. Οι περισσότερες τεχνικές βασίζονται στην κατασκευή ενός μοντέλου SPICE (small-profile intelligent coverage element) ενός συζευγμένου δικτύου που αναπαριστά τη μπαταρία. Το αποθηκευμένο φορτίο μοντελοποιείται με έναν πυκνωτή και η τάση στα άκρα του αναπαριστά την τάση εξόδου. Η διαδικασία

εκφόρτισης μοντελοποιείται με συνεχείς διορθώσεις τόσο στο αποθηκευμένο φορτίο όσο και στην τάση εξόδου. Τα μοντέλα αυτά είναι μεταβλητού φορτίου και έχουν τη δυνατότητα να συμπεριλάβουν φαινόμενα ρυθμού χωρητικότητας και θερμικά φαινόμενα, ενώ προσομοιώνονται εύκολα με τους υπάρχοντες προσομοιωτές κυκλωμάτων.

2.3.3 Στοχαστικά μοντέλα

Ένα στοχαστικό μοντέλο μπαταρίας αποτελείται από πεπερασμένο αριθμό μονάδων φορτίου και η συμπεριφορά εκφόρτισης μοντελοποιείται χρησιμοποιώντας μια μεταβατική στοχαστική διαδικασία διακριτού χρόνου. Καθώς η διαδικασία προχωρά στο χρόνο (που διαιρείται σε μια ακολουθία ισομεγεθών χρονοσχισμών), η κατάσταση της μπαταρίας παρακολουθείται με τον αριθμό των υπολειπόμενων μονάδων φορτίου. Σε κάθε χρονοσχισμή, το μέσο ρεύμα εκφόρτισης μετριέται και χρησιμοποιείται για να προσδιορισθεί ο αριθμός των μονάδων φορτίου που έχουν καταναλωθεί. Το μοντέλο είναι κατάλληλο για μεταβλητά φορτία, ενώ έχει χρησιμοποιηθεί για τη μοντελοποίηση μπαταριών ιόντων λιθίου. Συμπεριλαμβάνει τα φαινόμενα ρυθμού εκφόρτισης και ανάκαμψης (recovery) όχι όμως και τα θερμικά, είναι ακριβές και έχει μέτριες υπολογιστικές απαιτήσεις.

2.3.4 Ηλεκτροχημικά μοντέλα

Τα μοντέλα αυτά μελετούν άμεσα τις ηλεκτροχημικές διαδικασίες, τις θερμοδυναμικές διαδικασίες και τη φυσική κατασκευή κατά τη μοντελοποίηση της εκφόρτισης της μπαταρίας. Είναι πολύ πιο λεπτομερή και ακριβή σε σχέση με τα προηγούμενα, με πολλές παραμέτρους, ενώ η επίλυση των εξισώσεων γίνεται με αριθμητικές τεχνικές. Τα μοντέλα αυτά είναι άμεσα συνδεδεμένα με συγκεκριμένες μπαταρίες.

2.4 Μοντελοποίηση απωλειών ενέργειας

Η δικτυακή διεπαφή του MANET έχει τέσσερις πιθανές καταστάσεις κατανάλωσης ενέργειας: η *μετάδοση* (transmit) και η *λήψη* (receive) αφορούν στην μετ και λήψη πακέτων. Στην κατάσταση *αδράνειας* (idle), η διεπαφή ακούει το κανάλι και μπορεί να μεταδώσει ή να λάβει πακέτα. Αυτή είναι και η προεπιλεγμένη κατάσταση για ad hoc δίκτυα. Η κατάσταση *ύπνου* (sleep) έχει ιδιαίτερα χαμηλή ενεργειακή κατανάλωση, ενώ δεν επιτρέπει μετάδοσηε ή λήψη μέχρις ότου ο κόμβος ξυπνήσει.

Επειδή στα MANET δεν υπάρχει κεντρικός συντονισμός, οι κόμβοι αδυνατούν να προβλέψουν πότε θα λάβουν κίνηση, γι’ αυτό και η προεπιλεγμένη κατάστασή τους είναι η κατάσταση αδράνειας.

Στη συνέχεια θα μελετήσουμε δύο μοντέλα απωλειών: το βασικό μοντέλο, όπου οι απώλειες υπολογίζονται σύμφωνα με τις χαρακτηριστικές τιμές τάσης και ρεύματος της ασύρματης κάρτας διεπαφής δικτύου, και το εμπειρικό μοντέλο, που είναι βασισμένο σε πειραματικές μετρήσεις.

2.4.1 Βασικό μοντέλο απωλειών

Σύμφωνα με το βασικό μοντέλο απωλειών, που χρησιμοποιείται στην εργασία [13], κάθε φορά που μεταδίδεται ή λαμβάνεται ένα πακέτο, αφαιρείται η αντίστοιχη ενέργεια από τον κόμβο. Οι τύποι που υπολογίζουν την ενέργεια (σε Joule) είναι για μετάδοση και λήψη αντίστοιχα :

$$Energy_{T_x} = \frac{I_{T_x} \cdot V \cdot PacketSize}{R} \tag{2.2}$$

$$Energy_{R_x} = \frac{I_{R_x} \cdot V \cdot PacketSize}{R} \tag{2.3}$$

όπου I είναι η ένταση του ρεύματος σε Ampere, V η τάση τροφοδοσίας σε Volt, $PacketSize$ το μέγεθος του πακέτου σε bits, και R ο ρυθμός μετάδοσης σε bps.

Επομένως, η καταναλισκόμενη ενέργεια εξαρτάται από τα χαρακτηριστικά της κάρτας διεπαφής, από το μέγεθος του πακέτου και το το ρυθμό μετάδοσης. Ενέργεια όμως καταναλώνεται και στις καταστάσεις αδράνειας και ύπνου. Για να συμπεριληφθούν και αυτές οι απώλειες, απλώς χρησιμοποιούμε το αντίστοιχο ρεύμα για να βρούμε την ισχύ και έπειτα πολλαπλασιάζουμε με τον χρόνο t αδρανείας/ύπνου για να υπολογίσουμε την ενέργεια, δηλαδή :

$$Energy_{idle} = I_{idle} \cdot V \cdot t_{idle} \tag{2.4}$$

$$Energy_{sleep} = I_{sleep} \cdot V \cdot t_{sleep} \tag{2.5}$$

Αν το μοντέλο χρησιμοποιείται για σύγκριση της ενεργειακής απόδοσης μεταξύ διαφορετικών πρωτοκόλλων, τότε υποθέτουμε ότι όλα τα πρωτόκολλα έχουν την ίδια κατανάλωση κατά τον χρόνο αδρανείας, οπότε ασχολούμαστε μόνο με την κατανάλωση λόγω της μετάδοση και λήψης πακέτων. Έτσι απλοποιείται το μοντέλο, ενώ τα αποτελέσματα είναι ευκολότερο να συγκριθούν. Αν, όμως, το μοντέλο προορίζεται για την προσομοίωση ενός συστήματος υπό ρεαλιστικές συνθήκες, τότε κρίνεται σκόπιμο να συμπεριληφθούν οι απώλειες λόγω αδράνειας/ύπνου ή ακόμα και οι απώλειες λόγω των μεταβάσεων από τη μία κατάσταση στην άλλη.

2.4.2 Εμπειρικό μοντέλο απωλειών

Το εμπειρικό μοντέλο απωλειών, που περιγράφεται στην εργασία [19], είναι δομημένο πάνω στο πρότυπο IEEE 802.11. Είναι άμεσα συνδεδεμένο με την ασύρματη κάρτα διεπαφής δικτύου που χρησιμοποιείται, καθώς οι συντελεστές του μοντέλου προσδιορίζονται πειραματικά.

Το κόστος για την αποστολή ή λήψη ενός πακέτου στρώματος δικτύου από έναν κόμβο μοντελοποιείται γραμμικά. Υπάρχει ένα σταθερό κόστος σχετικό με την απόκτηση του καναλιού και ένα κόστος αυξανόμενο αναλογικά με το μέγεθος του πακέτου :

$$Cost = m \times size + b \tag{2.6}$$

Το συνολικό κόστος ενός πακέτου είναι το άθροισμα από όλα τα κόστη που επιφέρουν ο αποστολέας (s) και όλοι οι παραλήπτες. Οι πιθανοί παραλήπτες είναι ο προορισμός (d), οι κόμβοι εντός της εμβέλειας του s ($n \in S$), και οι κόμβοι εντός της εμβέλειας του d ($n \in D$). Σημειώνεται ότι τα σύνολα S και D είναι δυναμικά. Αναλύονται τώρα διεξοδικότερα οι απώλειες για κίνηση ευρυεκπομπής, κίνηση από σημείο προς σημείο και απόρριψη κίνησης.

- *Κίνηση ευρυεκπομπής*

Στο IEEE 802.11 ο αποστολέας ακούει το κανάλι και αν είναι ελεύθερο, στέλνει το μήνυμα, το οποίο λαμβάνουν όλοι οι κόμβοι εντός της εμβέλειάς του. Αν δεν είναι ελεύθερο, ο αποστολέας εκτελεί εκθετική υποχώρηση και ξαναπροσπαθεί αργότερα. Δεν ορίζονται επιβεβαιώσεις ή επαναμεταδόσεις για κίνηση ευρυεκπομπής.

Ορίζοντας τα σταθερά κόστη πρόσβασης στο κανάλι b_{send} και b_{recv} καθώς και τα μεταβλητά κόστη φορτίου m_{send} και m_{recv}, η (2.6) γίνεται

$$Cost = m_{send} \times size + b_{send} + \sum_{n \in S} (m_{recv} \times size + b_{recv)} \tag{2.7}$$

- *Κίνηση από σημείο προς σημείο*

Τα μεταβλητά κόστη m_{send} και m_{recv} είναι ίσα για ευρυεκπομπή και κίνηση από σημείο προς σημείο. Για την κίνηση από σημείο προς σημείο, τα σταθερά κόστη συμπεριλαμβάνουν το κόστος πρόσβασης στο κανάλι, το οποίο θεωρείται ίδιο με αυτό της ευρυεκπομπής, και τη διαπραγμάτευση MAC. Η πηγή στέλνει ένα μήνυμα RTS προς τον προορισμό, ο οποίος απαντά με μήνυμα CTS. Αφού η πηγή λάβει το CTS, στέλνει τα δεδομένα και περιμένει για την επιβεβαίωση (ACK). Χάριν απλότητας θεωρούμε ότι τα μικρά αυτά μηνύματα ελέγχου έχουν το ίδιο σταθερό κόστος αποστολής ($b_{sendctl}$) και λήψης ($b_{recvctl}$). Έτσι, προκύπτει για τον αποστολέα και τον προορισμό αντίστοιχα :

$$Cost = b_{sendctl} + b_{recvctl} + m_{send} \times size + b_{send} + b_{recvctl} \tag{2.8}$$

$$Cost = b_{recvctl} + b_{sendctl} + m_{recv} \times size + b_{send} + b_{recvctl} \tag{2.9}$$

Στην πράξη τα μηνύματα μπορεί να χαθούν λόγω συγκρούσεων ή άλλων σφαλμάτων, οπότε το πρωτόκολλο παρέχει επαναμεταδόσεις σε στρώμα MAC. Γι' αυτό κάθε στοιχείο του πρωτοκόλλου στις εξισώσεις (2.8) – (2.11) εμπεριέχει τον παράγοντα ($1+N_{επαναμεταδόσεις/διπλοεγγραφές}$) που υπονοείται. Χωρίζοντας έτσι τα μηνύματα ελέγχου γίνεται εφικτός ο υπολογισμός των επαναμεταδόσεων.

- *Απόρριψη Κίνησης*

Υπάρχουν κόμβοι που δεν είναι προορισμοί αλλά «κρυφακούν» (overhear) κίνηση εντός της εμβέλειας του αποστολέα ή του παραλήπτη. Οι κόμβοι εντός της εμβέλειας του αποστολέα αλλά όχι του προορισμού (εκτεθειμένα τερματικά), κρυφακούν μόνο την πλευρά του αποστολέα. Οι κόμβοι εντός της εμβέλειας του προορισμού αλλά όχι του αποστολέα (κρυμμένα τερματικά), κρυφακούν μόνο την πλευρά του προορισμού.

Το ενεργειακό κόστος απόρριψης κίνησης εξαρτάται σε μεγάλο βαθμό από την υλοποίηση του MAC. Για το IEEE 802.11, οι κόμβοι εντός της εμβέλειας του αποστολέα κρυφακούν το μήνυμα RTS και δεδομένα, ενώ κόμβοι εντός της εμβέλειας του προορισμού κρυφακούν το μήνυμα CTS και δεδομένα. Έτσι, για τον κόμβο που δεν είναι προορισμός έχουμε :

$$Cost = \sum_{n \in S} b_{discardctl} + \sum_{n \in D} b_{discardctl} + \sum_{n \in S} \left(m_{discard} \times size + b_{discard} \right) + \sum_{n \in D} b_{discardctl} \qquad (2.10)$$

Στη χειρότερη περίπτωση έχουμε $m_{discard} = m_{recv}$ και $b_{discard} = b_{recv}$, δηλαδή οι κόμβοι λαμβάνουν όλα τα πακέτα και στη συνέχεια αγνοούν όσα δεν προορίζονται για αυτούς. Μια αποδοτικότερη στρατηγική επιτρέπει τους κόμβους που δεν είναι προορισμοί να βρίσκονται σε μια κατάσταση μειωμένης ενεργειακής κατανάλωσης όσο το μέσο μεταφέρει αδιάφορη κίνηση.

Όσο μεγαλύτερη η χρονική διάρκεια μετάδοσης δεδομένων, τόσο περισσότερο χρόνο παραμένει κόμβος στην κατάσταση αυτή. Ειδικότερα, η κάρτα δικτύου Lucent WaveLAN IEEE 802.11 με βάση τις πληροφορίες μεγέθους του μηνύματος ελέγχου εισάγει του κόμβους εντός της εμβέλειας του αποστολέα που δεν είναι προορισμοί στην κατάσταση μειωμένης ενεργειακής κατανάλωσης κατά τη μετάδοση δεδομένων. Δεν υπάρχει ανάλογη στρατηγική, όμως, για τους κόμβους εντός της εμβέλειας του προορισμού, οι οποίοι απλά απορρίπτουν την κίνηση.

Ένας κόμβος που δεν αποτελεί προορισμό και λειτουργεί με «αδιάκριτο τρόπο» (promiscuous mode) ακούει όλη την κίνηση, είτε είναι ο προορισμός είτε όχι. Τα δεδομένα λαμβάνονται όπως και τα δεδομένα ευρυεκπομπής, όπως στην ανωτέρω χειρότερη περίπτωση. Η διαφορά ανάμεσα στα b_{recv}, m_{recv} και $b_{discard}$, $m_{discard}$ υποδεικνύει την επιπρόσθετη κατανάλωση ενέργειας λόγω της λειτουργίας σε promiscuous mode. Έτσι, για τον κόμβο που δεν είναι προορισμός και λειτουργεί σε promiscuous mode έχουμε :

$$Cost = \sum_{n_{promisc} \in S} b_{discardctl} + \sum_{n_{promisc} \in D} b_{discardctl} + \sum_{n_{promisc} \in S} \left(m_{recv} \times size + b_{recv} \right) + \sum_{n_{promisc} \in D} b_{discardctl} \quad (2.11)$$

Οι προδιαγραφές των ασύρματων καρτών διεπαφής δικτύου δεν παρέχουν τους συντελεστές που ορίστηκαν παραπάνω. Γι' αυτό εκτελέστηκαν πειράματα, τα οποία παρουσιάζονται στην εργασία [18], για τη μέτρηση της ενεργειακής κατανάλωσης

μιας διεπαφής βασισμένης στο IEEE 802.11, συγκεκριμένα της κάρτας Lucent IEEE 802.11 WaveLAN PC card. Στα πειράματα αυτά μετριέται η τάση εισόδου και η ένταση ρεύματος στη συσκευή, οπότε η στιγμιαία ισχύς προκύπτει ως το γινόμενό τους. Λόγω της έμμεσης αυτής μέτρησης των μεγεθών οι τιμές έχουν μια αξιοσημείωτη αβεβαιότητα της τάξεως του 10-15%. Παρ' όλα αυτά, παρέχουν μια ενδεικτική εικόνα για τα σχετικά κόστη, που είναι σημαντική για την ανάλυση επίδοσης υψηλότερου επιπέδου. Τα αποτελέσματα, δηλαδή οι συντελεστές του γραμμικού μοντέλου για την αποστολή, λήψη και απόρριψη δεδομένων, φαίνονται στον Πίνακα 2.2 ενώ οι σταθερές που χρησιμοποιήθηκαν παρουσιάζονται στον Πίνακα 2.3. Ακόμα, οι μετρήσεις της τάσης και του ρεύματος που χρησιμοποιήθηκαν για τον προσδιορισμό των συντελεστών αυτών φαίνονται στον Πίνακα 2.4, όπου παρατηρούμε συμφωνία με τις προδιαγραφές στα 11Mbps, αλλά μειωμένες τιμές ρεύματος στα 2Mbps.

		μW·s/byte (excl. MAC header)		μW·s
point-to-point send	$Cost =$	1.9	$\times$ *size*	+ 420
broadcast send	$Cost =$	1.9	$\times$ *size*	+ 250
point-to-point recv	$Cost =$	0.42	$\times$ *size*	+ 330
broadcast recv	$Cost =$	0.50	$\times$ *size*	+ 56
	non-destination $n \in \mathcal{S}, \mathcal{D}$			
promiscuous recv	$Cost =$	0.39	$\times$ *size*	+ 140
discard	$Cost =$	−0.49	$\times$ *size*	+ 97
	non-destination $n \in \mathcal{S}, n \notin \mathcal{D}$			
promiscuous recv	$Cost =$	0.54	$\times$ *size*	+ 66
discard	$Cost =$	−0.58	$\times$ *size*	+ 24
	non-destination $n \notin \mathcal{S}, n \in \mathcal{D}$			
promiscuous "recv"	$Cost =$	0.0029	$\times$ *size*	+ 63
discard	$Cost =$	−0.0058	$\times$ *size*	+ 56
idle	$Cost = 808$ mW			

Πίνακας 2.2 : Πειραματικοί συντελεστές του γραμμικού μοντέλου ενεργειακής κατανάλωσης για την κάρτα Lucent IEEE 802.11 WaveLAN PC card 2.4 GHZ direct sequence spread spectrum

m_{send}	1.89	mW·s/byte
b_{send}	246	mW·s
m_{recv}	0.494	mW·s/byte
b_{recv}	56.1	mW·s
m_{discard}	−0.490	mW·s/byte
b_{discard}	97.2	mW·s
$m_{\text{recv\_promisc}}$	0.388	mW·s/byte
$b_{\text{recv\_promisc}}$	136	mW·s
b_{sendctl}	120	mW·s
b_{recvctl}	29.0	mW·s

Πίνακας 2.3 : Σταθερές που χρησιμοποιήθηκαν στις προσομοιώσεις

Κατάσταση	Μετρήσεις	Προδιαγραφές
Ρεύμα (mA) στα 2 Mbps		
Ύπνος (sleep)	14	9
Αδράνεια (idle)	178	n/a
Λήψη (receive)	204	280
Μετάδοση (transmit)	280	330
Ρεύμα (mA) στα 11 Mbps		
Ύπνος	10	10
Αδράνεια	156	n/a
Λήψη	190	180
Μετάδοση	284	280
Τάση τροφοδοσίας (V)		
Όλες	4,74	5

Πίνακας 2.4 : Χαρακτηριστικά της κάρτας Lucent IEEE 802.11 WaveLAN PC card 2.4 GHZ

Από την παραπάνω ανάλυση προκύπτει ότι το εμπειρικό μοντέλο είναι ακριβέστερο, καθώς έχει προκύψει από πειραματικές μετρήσεις με χρήση της συγκεκριμένης ασύρματης κάρτα διεπαφής δικτύου. Παράλληλα, όμως, αυτό είναι και το μειονέκτημά του, καθώς η διεξαγωγή πειράματος για τον προσδιορισμό των συντελεστών αυτών είναι μια χρονοβόρα διαδικασία. Η ταξινόμηση των καρτών δικτύου σύμφωνα με τα χαρακτηριστικά τους και έπειτα η διεξαγωγή πειραμάτων, ώστε να σχηματισθούν πίνακες με τους συντελεστές για κάθε ομάδα καρτών, θα βοηθούσε στη χρηστικότητα του μοντέλου.

Τελικά, προκύπτουν κάποιες βασικές ερωτήσεις που αφορούν τη σχεδίαση πρωτοκόλλων με έμφαση στα ενεργειακά θέματα.

- *Είναι το κόστος λήψης μηνύματος σημαντικό;*

Το κόστος λήψης είναι πραγματικά σημαντικό. Αν ένα μήνυμα ευρυεκπομπής ληφθεί από περισσότερους από τέσσερις γείτονες, το συνολικό κόστος λήψης του πακέτου είναι μεγαλύτερο από το κόστος αποστολής του. Το σχετικό κόστος λήψης είναι πιθανόν να αυξηθεί, εκφράζοντας μια τάση προς μεγαλύτερη ευαισθησία και δυνατότητες επεξεργασίας σήματος στο δέκτη.

- *Ποια είναι τα σχετικά κόστη των μικρών και μεγάλων πακέτων;*

Το σταθερό κόστος αποστολής ή λήψης ενός πακέτου είναι σχετικά μεγάλο συγκρινόμενο με το μεταβλητό κόστος. Για μικρά πακέτα (130 Bytes για ευρυεκπομπή και 230 Bytes από σημείο προς σημείο), το σταθερό κόστος είναι μεγαλύτερο από το μεταβλητό. Έτσι, τα μικρά μηνύματα αίτησης διαδρομής (route request) ή “hello” είναι ένας σχετικά δαπανηρός μηχανισμός. Ακόμα, οι επικεφαλίδες πηγαίας διαδρομής (source route) είναι σχετικά φθηνές όσον αφορά την ενεργειακή κατανάλωση.

- *Ποιο είναι το κόστος απόρριψης ενός πακέτου;*

Η απόρριψη πακέτου είναι γενικά λιγότερο δαπανηρή από τη λήψη του, ειδικά για μεγάλα μηνύματα, καθώς οι κόμβοι που δεν είναι προορισμοί μπορούν να μειώσουν την ενεργειακή τους κατανάλωση κατά τη μετάδοση αδιάφορων δεδομένων. Σημειώνεται ότι η εξοικονόμηση αυτή προκύπτει από το υψηλό κόστος της άεργης κατάστασης που χρησιμοποιείται ως προεπιλογή στους κόμβους αδόμητων δικτύων. Παρ' όλα αυτά, ακόμα και αν το $m_{discard}$ ήταν μηδέν, η λήψη κίνησης σε promiscuous mode θα ήταν ακριβότερη από την απόρριψή της.

- *Ποια είναι τα σχετικά κόστη της κίνησης ευρυεκπομπής και από σημείο προς σημείο;*

Όταν το κόστος λήψης είναι σημαντικό, το κόστος ευρυεκπομπής κυριαρχείται από το κόστος στους δέκτες, το οποίο είναι ιδιαίτερα αυξημένο σε πυκνά δίκτυα. Η κίνηση από σημείο προς σημείο έχει μεγαλύτερο κόστος αποστολής (σχεδόν δύο φορές) και λήψης (σχεδόν έξι φορές) από την ευρυεκπομπή, αλλά επιτρέπει στους κόμβους που δεν είναι προορισμοί να απορρίψουν την κίνηση. Αν το κόστος απόρριψης είναι υψηλό, τότε το βασικό πλεονέκτημα της κίνησης από σημείο προς σημείο είναι η αποφυγή συγκρούσεων και η επιβεβαίωση λήψης των δεδομένων. Αν το κόστος απόρριψης είναι χαμηλό, τότε υπάρχει σημαντική εξοικονόμηση ενέργειας.

Κεφάλαιο 3

Δρομολόγηση

3.1 Εισαγωγή

Δρομολόγηση (routing) είναι η διαδικασία με την οποία επιλέγεται η διαδρομή μέσα σε ένα δίκτυο πάνω από την οποία θα σταλούν δεδομένα. Στα δίκτυα μεταγωγής πακέτων, όπως το διαδίκτυο, η δρομολόγηση εκτελεί την προώθηση πακέτων, τη μετάβαση των λογικά διευθυνσιοδοτημένων πακέτων από την πηγή τους προς τον απόλυτο προορισμό τους μέσω ενδιάμεσων κόμβων, οι οποίοι συνήθως είναι συσκευές όπως δρομολογητές, μεταγωγείς, γέφυρες, πύλες ή ακόμα και υπολογιστές γενικής χρήσης [1].

Το πρωτόκολλο (protocol) είναι ένα σύνολο κανόνων που καθορίζουν τη μορφή και τη σημασία των πακέτων που ανταλλάσσονται ανάμεσα σε ομότιμες οντότητες. Το πρωτόκολλο δρομολόγησης (routing protocol) είναι ένα πρωτόκολλο που προσδιορίζει τον τρόπο επικοινωνίας των δρομολογητών, την ανταλλαγή πληροφοριών ώστε να επιλέξουν βάσει ενός αλγορίθμου δρομολόγησης τη διαδρομή ανάμεσα σε δύο κόμβους σε ένα δίκτυο υπολογιστών. Σε κάθε αλγόριθμο δρομολόγησης είναι επιθυμητές μερικές ιδιότητες: ορθότητα, απλότητα, ανθεκτικότητα, σταθερότητα, δικαιοσύνη και βέλτιστη απόδοση [2].

Οι αλγόριθμοι δρομολόγησης στα δίκτυα μεταγωγής δεδομένων χωρίζονται σε δύο βασικές κατηγορίες, τους αλγορίθμους δρομολόγησης με διανύσματα απόστασης (distance vector routing) και τους αλγορίθμους δρομολόγησης με κατάσταση ζεύξεων (link state routing).

Ο αλγόριθμος δρομολόγησης με διανύσματα απόστασης λειτουργεί υποχρεώνοντας κάθε δρομολογητή να διατηρεί έναν πίνακα (ένα διάνυσμα) που δίνει την καλύτερη γνωστή απόσταση προς κάθε προορισμό, καθώς και τη γραμμή που πρέπει να χρησιμοποιηθεί για να φτάσουμε εκεί. Οι πίνακες αυτοί ενημερώνονται μέσω ανταλλαγής πληροφοριών με τους γείτονες. Συχνά αναφέρεται και ως κατανεμημένος αλγόριθμος δρομολόγησης Bellman-Ford.

Ο αλγόριθμος δρομολόγησης με κατάσταση ζεύξεων λειτουργεί υποχρεώνοντας κάθε δρομολογητή να κατασκευάζει ένα χάρτη σε μορφή γράφου με τις συνδέσεις μεταξύ όλων των κόμβων. Έπειτα, κάθε δρομολογητής υπολογίζει την καλύτερη διαδρομή από αυτόν προς κάθε πιθανό προορισμό. Η συλλογή των βέλτιστων διαδρομών σχηματίζει τον πίνακα δρομολόγησης του δρομολογητή.

3.2 Δρομολόγηση στα δίκτυα MANET

Τα δίκτυα MANET έχουν κάποια ειδικά χαρακτηριστικά, όπως δυναμική τοπολογία, περιορισμούς εύρους ζώνης, μεταβλητή χωρητικότητα στις ζεύξεις, ενεργειακούς περιορισμούς, επικοινωνία πολλαπλών βημάτων, περιορισμένη ασφάλεια. Έτσι, τα πρωτόκολλα δρομολόγησης πρέπει να εκτελούν τις παρακάτω λειτουργίες, ώστε να εξασφαλιστεί η επικοινωνία κάτω από τις συνθήκες αυτές : Παρακολούθηση και αναγνώριση αλλαγών στην τοπολογία του δικτύου, διατήρηση συνδεσιμότητας υπό μεταβαλλόμενες συνθήκες ραδιοδιαύλου και κινητικότητας, χρονοδρομολόγηση εκπομπών και εκχώρηση καναλιών, δρομολόγηση πακέτων. Για να επιτευχθούν οι λειτουργίες αυτές, τα πρωτόκολλα δρομολόγησης αυτοοργανούμενων δικτύων πρέπει να έχουν τις ακόλουθες ιδιότητες [4]:

- **Κατανεμημένη λειτουργία** : Καθώς τα δίκτυα είναι αυτόνομα και αυτοοργανούμενα, πρέπει κάθε κόμβος να έχει τη δυνατότητα να αποφασίζει για τη δρομολόγηση ανεξάρτητα από κεντρικούς κόμβους ελέγχου.
- **Αποδοτική χρησιμοποίηση του εύρους ζώνης** : Καθώς το εύρος ζώνης είναι περιορισμένο, πρέπει να περιοριστεί και η επιβάρυνσή του από πακέτα ελέγχου και σηματοδοσίας.
- **Αποδοτική χρησιμοποίηση της χωρητικότητας της μπαταρίας** : Καθώς η δρομολόγηση γίνεται διαμέσου άλλων κόμβων, ο περιορισμένος χρόνος ζωής των κόμβων προκαλεί μείωση των διαθέσιμων κόμβων, που σημαίνει πιθανή τμηματοποίηση του δικτύου και μείωση της επίδοσής του.
- **Βελτιστοποίηση των κριτηρίων δρομολόγησης (routing metrics)** : Μερικοί σημαντικοί παράγοντες, κάποιοι από τους οποίους είναι αντικρουόμενοι, είναι η μέγιστη διέλευση (throughput) από άκρο σε άκρο, η ελάχιστη καθυστέρηση από άκρο σε άκρο, η ελάχιστη διαδρομή σε βήματα, η ελάχιστη συνολική ισχύς, η εξισορρόπηση φορτίου, η ελάχιστη επιβάρυνση, η προσαρμοστικότητα στη δυναμική τοπολογία, η σταθερότητα των διαδρομών.
- **Ταχεία σύγκλιση αλγορίθμου** : Πρέπει να επιτυγχάνεται η εύρεση μιας νέας σταθερής διαδρομής όσο γρηγορότερα γίνεται, μετά από μια αλλαγή στην τοπολογία του δικτύου.
- **Ελευθερία από βρόχους (loops)** : Η δρομολόγηση πακέτων μέσω βρόχων προκαλεί σημαντική επιβάρυνση στο εύρος ζώνης και την κατανάλωση ενέργειας και πρέπει να αποφεύγεται.
- **Υποστήριξη μονόδρομων (unidirectional) ζεύξεων** : Ενώ συνήθως οι ζεύξεις θεωρούνται αμφίδρομες, υπάρχει περίπτωση λόγω διαφορετικών δυνατοτήτων εκπομπής ή παρεμβολών να χρειαστεί η λειτουργία δικτύου μέσω μονόδρομων ζεύξεων.

3.3 Επισκόπηση πρωτοκόλλων δρομολόγησης MANET

Τα πρωτόκολλα δρομολόγησης των αυτοοργανούμενων δικτύων ταξινομούνται με βάση [7]:

- Το σημείο λήψης αποφάσεων. Έτσι, υπάρχουν πρωτόκολλα που χρησιμοποιούν **συγκεντρωμένους** αλγορίθμους, όπου όλες οι αποφάσεις για τις διαδρομές παίρνονται σε έναν κεντρικό κόμβο, και **κατανεμημένους** αλγορίθμους, όπου κάθε κόμβος αποφασίζει ανεξάρτητα για τον εαυτό του.
- Την προσαρμοστικότητά τους στην κίνηση. Έτσι, υπάρχουν τα πρωτόκολλα που χρησιμοποιούν **στατικούς** αλγορίθμους, όπου οι αποφάσεις τους παραμένουν σταθερές και ανεξάρτητες της δικτυακής κίνησης, και **προσαρμοστικούς** αλγορίθμους, που αλλάζουν τις αποφάσεις τους ανάλογα με το μοτίβο της κίνησης.
- Τη στρατηγική δρομολόγησης. Έτσι, υπάρχουν τα **οδηγούμενα από πίνακα** (table-driven ή proactive), τα **οδηγούμενα κατ΄ απαίτηση** (on-demand ή reactive) και τα **υβριδικά** (hybrid) πρωτόκολλα.

Στη συνέχεια, θα γίνει μια σύντομη παρουσίαση των πρωτοκόλλων με βάση την τελευταία και δημοφιλέστερη κατηγοριοποίηση [5].

3.3.1 Οδηγούμενα από πίνακα πρωτόκολλα (Table-Driven, Proactive)

Στα οδηγούμενα από πίνακα πρωτόκολλα, οι κόμβοι συνεχώς αναζητούν πληροφορίες δρομολόγησης εντός του δικτύου έτσι, ώστε όταν χρειαστεί διαδρομή προς ένα κόμβο, η διαδρομή αυτή να είναι ήδη διαθέσιμη. Κάθε κόμβος διατηρεί έναν ή περισσότερους πίνακες για την αποθήκευση των πληροφοριών αυτών, οι οποίες ανανεώνονται περιοδικά είτε υπάρχει αίτηση για αποστολή πακέτων είτε όχι. Τα κυριότερα μειονεκτήματα των πρωτοκόλλων αυτών είναι η σημαντική ποσότητα δεδομένων για συντήρηση και η αργή αντίδραση σε βλάβες και αλλαγές στην τοπολογία. Μερικά από τα πιο γνωστά πρωτόκολλα της κατηγορίας είναι τα παρακάτω:

1) Destination-Sequenced Distance Vector Routing (DSDV)

Το DSDV είναι βασισμένο στον κλασικό αλγόριθμο δρομολόγησης Bellman-Ford. Κάθε κόμβος διατηρεί μια λίστα με όλους τους προορισμούς και τον αριθμό των βημάτων για κάθε προορισμό. Κάθε καταχώρηση είναι συσχετισμένη με έναν αριθμό ακολουθίας (sequence number). Η μόνη βελτίωση είναι η αποφυγή των βρόχων δρομολόγησης, ώστε η πληροφορία δρομολόγησης να είναι πάντα διαθέσιμη, ανεξάρτητα αν η πηγή χρειάζεται τη διαδρομή ή όχι.

2) Wireless Routing Protocol (WRP)

Το WRP ανήκει στους αλγορίθμους εύρεσης μονοπατιού (path-finding) με την εξαίρεση ότι αποφεύγει το πρόβλημα της μέτρησης ως το άπειρο υποχρεώνοντας κάθε κόμβο να εκτελεί ελέγχους συνοχής των πληροφοριών από όλους τους γειτονικούς κόμβους, ενώ αποφεύγει και τους βρόχους δρομολόγησης. Οι αλλαγές στις ζεύξεις διαδίδονται με μηνύματα ανανέωσης, ενώ μηνύματα χαιρετισμού (Hello) ανταλλάσσονται περιοδικά ανάμεσα σε γειτονικούς κόμβους.

3) Optimized Link State Routing (OLSR)

Το OLSR χρησιμοποιεί κόμβους αναμετάδοσης προς πολλαπλά σημεία (Multipoint Relay nodes), ώστε να ελαχιστοποιήσει τον αριθμό πακέτων που εκπέμπονται στο δίκτυο. Κάθε κόμβος επιλέγει ένα υποσύνολο από γειτονικούς κόμβους για να επαναμεταδώσει τα πακέτα του, επιλέγοντας το ελάχιστο μονοπάτι [6].

4) Clusterhead Gateway Switch Routing (CGSR)

Στο CGSR οι κινητοί κόμβοι ομαδοποιούνται, ώστε κάθε ομάδα να έχει έναν επικεφαλής ομάδας. Ο επικεφαλής ελέγχει μια ομάδα κόμβων, καθώς η ομαδοποίηση παρέχει ένα πλαίσιο για διαχωρισμό του δικτύου, πρόσβαση σε κανάλια, δρομολόγηση και εκχώρηση εύρους ζώνης. Υπάρχει, όμως, το πρόβλημα σε περίπτωση βλάβης των κρίσιμων αυτών κόμβων. Ως υποκείμενος αλγόριθμος δρομολόγησης χρησιμοποιείται ο DSDV.

3.3.2 Οδηγούμενα κατ' απαίτηση πρωτόκολλα (On-Demand-Driven, Reactive)

Τα οδηγούμενα κατ' απαίτηση πρωτόκολλα δημιουργούν διαδρομές μόνο όταν το επιθυμούν οι πηγές κίνησης. Όταν μια πηγή απαιτεί μια διαδρομή προς έναν προορισμό, εκτελεί της διαδικασία της ανακάλυψης διαδρομής εντός του δικτύου. Η διαδικασία αυτή ολοκληρώνεται μόλις βρεθεί μια διαδρομή ή εξεταστούν όλοι οι πιθανοί συνδυασμοί μονοπατιών. Αφού βρεθεί η διαδρομή, συντηρείται με τη διαδικασία της συντήρησης διαδρομής μέχρις ότου είτε ο προορισμός να γίνει απρόσιτος για την πηγή είτε να μην είναι πλέον επιθυμητή η διαδρομή. Τα κυριότερα μειονεκτήματα των πρωτοκόλλων αυτών είναι η υψηλή καθυστέρηση στην εύρεση διαδρομής και η ευρυεκπομπή μεγάλης κλίμακας που πιθανόν να οδηγήσει σε υπερφόρτωση του δικτύου. Μερικά από τα πιο γνωστά πρωτόκολλα της κατηγορίας είναι τα παρακάτω :

1) Associativity Based Routing (ABR)

Το ABR είναι βασισμένο στην ιδέα της συσχέτισης, ότι δηλαδή δεν υπάρχει λόγος να επιλεγεί μια διαδρομή με βάση την ελάχιστη διαδρομή, όταν αυτή πρόκειται να «σπάσει» ή να ακυρωθεί λόγω της κινητικότητας των κόμβων. Έτσι, κάθε κόμβος μαθαίνει τη συσχέτισή του με τους γειτονικούς τους κόμβους, ώστε να επιλέξει τη βέλτιστη διαδρομή με βάση κάποιο κριτήριο, όπως η καθυστέρηση, η ισχύς λήψης ή η υπολειπόμενη ενέργεια.

2) Signal Stability Routing (SSR)

Το SSR είναι απόγονος του ABR, επιλέγοντας διαδρομές με βάση την ισχύ του σήματος ανάμεσα στους κόμβους και τη σταθερότητα της θέσης ενός κόμβου. Έτσι, επιλέγονται οι διαδρομές με την πιο ισχυρή συνδεσιμότητα.

3) *Dynamic Source Routing (DSR)*

Το DSR είναι βασισμένο στην ιδέα της πηγαίας δρομολόγησης (source routing). Οι κόμβοι πρέπει να συντηρούν μνήμες διαδρομών που περιέχουν τις πηγαίες διαδρομές που γνωρίζει ο κόμβος. Οι καταχωρήσεις αυτές ανανεώνονται συνεχώς με τους μηχανισμούς της ανακάλυψης διαδρομής, με χρήση πακέτων αίτησης διαδρομής και απάντησης διαδρομής, και της συντήρησης διαδρομής, με χρήση πακέτων βλάβης διαδρομής και επιβεβαίωσης.

4) *Temporally-Ordered Routing Algorithm (TORA)*

Ο TORA είναι ένας σημαντικά προσαρμοστικός, ελεύθερος από βρόχους, κατανεμημένος αλγόριθμος δρομολόγησης βασισμένος στην αντιστροφή ζεύξης που χρησιμοποιείται σε ιδιαίτερα δυναμικά περιβάλλοντα. Ξεκινά από την πηγή και παρέχει πολλαπλά μονοπάτια για κάθε ζεύγος πηγής – προορισμού. Απαιτεί συγχρονισμό των ρολογιών και αποτελείται από τρεις βασικές λειτουργίες, τη δημιουργία διαδρομής, τη συντήρηση διαδρομής και τη διαγραφή διαδρομής.

5) *Ad-Hoc On-Demand Distance Vector (AODV)*

Το AODV είναι βασισμένο στο DSDV, βελτιωμένο με την ελαχιστοποίηση του αριθμού των αναγκαίων ευρυεκπομπών (broadcasts), καθώς δημιουργεί διαδρομές κατ' απαίτηση. Η ανακάλυψη μονοπατιού εκτελείται όταν δεν υπάρχει διαδρομή προς τον προορισμό με ευρυεκπομπή πακέτων αίτησης διαδρομής, ενώ για την ανακοίνωση βλάβης ζεύξης απαιτούνται συμμετρικές ζεύξεις. Επιτυγχάνεται αποδοτική χρησιμοποίηση του εύρους ζώνης, απόκριση σε αλλαγές τοπολογίας και αποφυγή βρόχων δρομολόγησης.

6) *Relative Distance Micro diversity Routing (RDMAR)*

Το RDMAR εκτιμά την απόσταση ανάμεσα σε δύο κόμβους, χρησιμοποιώντας τον αλγόριθμο εκτίμησης σχετικής απόστασης. Περιορίζει την ακτίνα της αναζήτησης διαδρομής, ώστε να μειώσει το κόστος ευρυεκπομπής της αίτησης διαδρομής. Προϋποθέτει ότι όλοι οι κινητοί κόμβοι κινούνται με την ίδια ταχύτητα.

3.3.3 Υβριδικά πρωτόκολλα (Hybrid)

Τα υβριδικά πρωτόκολλα συνδυάζουν τα πλεονεκτήματα των δύο κατηγοριών. Η δρομολόγηση ξεκινά με διαδρομές που βρέθηκαν από proactive αναζήτηση και στη συνέχεια εξυπηρετεί τη ζήτηση που προκύπτει με reactive ευρυεκπομπή. Τα κυριότερα μειονεκτήματα είναι η εξάρτηση του κέρδους από το πλήθος των κόμβων που είναι ενεργοί και η εξάρτηση της αντίδρασης στην κίνηση από τη μεταβολή του όγκου κίνησης.

Ένα γνωστό πρωτόκολλο της κατηγορίας είναι το Zone Routing Protocol (ZRP). Είναι παρόμοιο με το CGSR με τη διαφορά ότι κάθε κόμβος λειτουργεί σαν επικεφαλής ομάδας και μέλος των άλλων ομάδων. Η ζώνη δρομολόγησης αποτελείται από λίγους κινητούς κόμβους που απέχουν ένα, δύο ή περισσότερα

βήματα από τον κεντρικό κόμβο. Η επίδοση βελτιώνεται, αλλά το μονοπάτι μπορεί να μην είναι το ευνοϊκότερο λόγω της ιεραρχικής δρομολόγησης, ενώ αυξάνονται οι απαιτήσεις σε μνήμη.

3.4 Το πρωτόκολλο DSR

Στην ενότητα αυτή θα γίνει μια αναλυτική περιγραφή του πρωτοκόλλου DSR βάσει του RFC 4728 [3]. Το DSR είναι το πρωτόκολλο που θα επεκταθεί ώστε να λειτουργεί με ενεργειακά κριτήρια, όπως αναλύεται στο επόμενο κεφάλαιο. Έτσι, είναι απαραίτητη η διεξοδική ανάλυσή του, ώστε να γίνουν κατανοητοί οι μηχανισμοί και ο τρόπος λειτουργίας του, για να εξηγηθεί στη συνέχεια η επέκτασή του.

3.4.1 Εισαγωγή

Το πρωτόκολλο DSR (Dynamic Source Routing) είναι ένα απλό και αποδοτικό πρωτόκολλο δρομολόγησης σχεδιασμένο ειδικά για χρήση σε ασύρματα αδόμητα δίκτυα πολλαπλών βημάτων με κινητούς κόμβους. Δίνει τη δυνατότητα στο δίκτυο να είναι πλήρως αυτοοργανούμενο και αυτορυθμιζόμενο, χωρίς την ανάγκη για προϋπάρχουσα δικτυακή υποδομή ή διαχείριση. Οι κόμβοι συνεργάζονται στην προώθηση πακέτων, ώστε να είναι δυνατή η επικοινωνία μέσω πολλαπλών βημάτων μεταξύ κόμβων που δεν βρίσκονται ο ένας εντός της ακτίνας εκπομπής του άλλου. Καθώς οι κόμβοι κινούνται ή συνδέονται ή αποσυνδέονται από το δίκτυο, ενώ παράλληλα οι συνθήκες ασύρματης μετάδοσης, όπως η παρεμβολή, μεταβάλλονται, η δρομολόγηση καθορίζεται και διατηρείται αυτόματα από το πρωτόκολλο δρομολόγησης DSR. Η τοπολογία του δικτύου μπορεί να είναι γρήγορα μεταβαλλόμενη, καθώς ο αριθμός ή η ακολουθία των ενδιάμεσων βημάτων που χρειάζονται για την εύρεση ενός προορισμού πιθανόν να μεταβάλλεται συνεχώς.

Το DSR αποτελείται από δύο βασικούς μηχανισμούς, την Ανακάλυψης Διαδρομής (Route Discovery) και τη Συντήρηση Διαδρομής (Route Maintenance), με τους οποίους οι κόμβοι ανακαλύπτουν και συντηρούν διαδρομές για κάθε τυχαίο προορισμό στο αδόμητο δίκτυο:

Ανακάλυψη Διαδρομής : Είναι ο μηχανισμός με τον οποίο ο κόμβος S που θέλει να αποστείλει ένα πακέτο στον κόμβο προορισμού D αποκτά μια πηγαία διαδρομή (source route) προς τον D. Χρησιμοποιείται μόνο όταν ο κόμβος S προσπαθήσει να στείλει ένα πακέτο στον D και δεν γνωρίζει ήδη μια διαδρομή προς αυτόν.

Συντήρηση Διαδρομής : Είναι ο μηχανισμός με τον οποίο ο κόμβος S μπορεί να αντιληφθεί, ενώ χρησιμοποιεί μια διαδρομή πηγής προς τον D, εάν η τοπολογία του δικτύου έχει αλλάξει έτσι, ώστε η γνωστή αυτή διαδρομή να μην είναι χρησιμοποιήσιμη, αφού κάποια ζεύξη κατά μήκος της δε λειτουργεί πλέον. Αν μια διαδρομή πηγής σπάσει, τότε ο κόμβος S μπορεί είτε να χρησιμοποιήσει μια άλλη διαδρομή που γνωρίζει για τον D, ή να εκκινήσει μια Ανακάλυψη Διαδρομής για να βρει μια νέα διαδρομή προς τον D για τα υπόλοιπα πακέτα.

Όλες οι λειτουργίες του πρωτοκόλλου εκτελούνται κατ' απαίτηση (on demand), ώστε η επιβάρυνση (overhead) δρομολόγησης του DSR να αυξομειώνεται ανάλογα με την ανάγκη για αντίδραση στις αλλαγές των ήδη χρησιμοποιούμενων διαδρομών. Έτσι, δεν υπάρχει ανάγκη για περιοδική αποστολή πακέτων σε κανένα στρώμα του δικτύου. Επίσης, το πρωτόκολλο υποστηρίζει πολλαπλές διαδρομές προς κάθε προορισμό, ενώ κάθε αποστολέας επιλέγει και ελέγχει τις διαδρομές που χρησιμοποιεί για τη δρομολόγηση των πακέτων του, κάτι που παρέχει, για παράδειγμα, εξισορρόπηση φορτίου ή αυξημένη ευρωστία. Άλλα πλεονεκτήματα του DSR είναι η εγγυημένη ακυκλική (loop-free) δρομολόγηση, η λειτουργία σε δίκτυα με μονόδρομες ζεύξεις και η πολύ γρήγορη ανάκαμψη σε αλλαγές των διαδρομών του δικτύου. Το πρωτόκολλο είναι σχεδιασμένο κυρίως για δίκτυα MANET μεγέθους μέχρι διακοσίων κόμβων, ενώ λειτουργεί ικανοποιητικά και για πολύ υψηλούς ρυθμούς κινητικότητας.

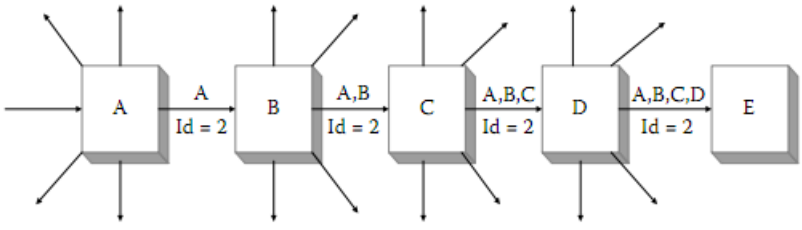
3.4.2 Βασικοί Μηχανισμοί

Η βασική έκδοση του DSR χρησιμοποιεί πηγαία δρομολόγηση (source routing), στην οποία κάθε πακέτο δεδομένων μεταφέρει στην επικεφαλίδα του πλήρη ταξινομημένη λίστα των κόμβων από τους οποίους θα περάσει. Έτσι, ο αποστολέας επιλέγει και ελέγχει τις διαδρομές που ακολουθούν τα πακέτα του, ενώ οι άλλοι κόμβοι που προωθούν ή κρυφακούν τα πακέτα μπορούν να αποθηκεύσουν τις πληροφορίες για μελλοντική χρήση.

3.4.2.1 Ανακάλυψη Διαδρομής

Όταν μια πηγή δημιουργεί ένα νέο πακέτο προς αποστολή, προσθέτει στην επικεφαλίδα του πακέτου την πηγαία διαδρομή με την ακολουθία των βημάτων προς τον προορισμό. Αρχικά, ο αποστολέας αναζητά μια κατάλληλη διαδρομή στην μνήμη διαδρομών (route cache), όπου υπάρχουν όλες οι διαδρομές που έχει ήδη μάθει. Αν δε βρει καμία διαδρομή, θα ξεκινήσει Ανακάλυψη Διαδρομής, ώστε να βρει μια νέα διαδρομή προς τον προορισμό.

Για παράδειγμα, έστω ότι στο Σχήμα 3.1 ο κόμβος Α επιχειρεί να ανακαλύψει μια διαδρομή προς τον κόμβο Ε.



Σχήμα 3.1 : Ανακάλυψη Διαδρομής: πηγή ο Α και προορισμός ο Ε

Κατά την εκκίνηση της Ανακάλυψης Διαδρομής, ο κόμβος Α εκπέμπει (broadcast) μια αίτηση διαδρομής (Route Request - RREQ), η οποία λαμβάνεται από σχεδόν όλους τους γειτονικούς κόμβους, συμπεριλαμβανομένου του κόμβου Β. Κάθε RREQ

περιέχει τον αρχικό κόμβο, τον προορισμό και ένα μοναδικό αναγνωριστικό αίτησης (Request Identification - ID). Ακόμα, περιέχει μια λίστα, αρχικά κενή, με τις διευθύνσεις των κόμβων τους οποίους έχει ήδη διασχίσει.

Κατά τη λήψη της RREQ, αν ο κόμβος είναι και ο προορισμός (κόμβος E), επιστρέφει μια Απάντηση Διαδρομής (Route Reply - RREP) προς τον αρχικό κόμβο που ξεκίνησε την Ανακάλυψη Διαδρομής, μαζί με τη διαδρομή που καταγράφηκε. Όταν ο αρχικός κόμβος λάβει τη RREP, αποθηκεύει τη καταγραφείσα διαδρομή στη μνήμη διαδρομών του, ώστε να τη χρησιμοποιήσει για την αποστολή πακέτων δεδομένων προς το συγκεκριμένο προορισμό.

Αν ο κόμβος είναι ενδιάμεσος και έχει πρόσφατα λάβει RREQ με το ίδιο ζεύγος πομπού – προορισμού και με ίδιο ID ή η διεύθυνσή του υπάρχει ήδη στη λίστα, τότε το πακέτο απορρίπτεται. Αλλιώς, ο κόμβος προσθέτει στη λίστα τη διεύθυνσή του και επανεκπέμπει τη RREQ με το ίδιο ID. Στο παράδειγμα, ο κόμβος B λαμβάνει το RREQ και το επανεκπέμπει, οπότε το λαμβάνει ο C και το εκπέμπει για να το λάβει ο D, που το εκπέμπει και το λαμβάνει ο προορισμός E.

Κατά την αποστολή της RREP προς τον αρχικό κόμβο, στο παράδειγμα από τον E στον A, ο κόμβος E ψάχνει στη μνήμη διαδρομών για μια διαδρομή προς τον A. Στην περίπτωση αμφίδρομων ζεύξεων μπορεί να χρησιμοποιήσει αυτή που μόλις έλαβε από τη RREQ. Σε περίπτωση μη αμφίδρομων ζεύξεων όπου δεν υπάρχει άλλη διαδρομή στη μνήμη του E, πρέπει να ξεκινήσει μια νέα Ανακάλυψη Διαδρομής προς τον A.

Κατά την εκκίνηση της Ανακάλυψης Διαδρομής, ο αποστολέας αποθηκεύει ένα αντίγραφο του αρχικού πακέτου δεδομένων που την πυροδότησε σε ένα τοπικό ενταμιευτή, τον ενταμιευτή αποστολής (send buffer). Ο ενταμιευτής αποστολής περιέχει ένα αντίγραφο για κάθε πακέτο που δεν εστάλη, επειδή δεν υπήρχε πηγαία διαδρομή προς τον προορισμό. Κάθε αντίγραφο είναι συσχετισμένο με το χρόνο εγγραφής του στον ενταμιευτή και απορρίπτεται αν ο χρόνος παραμονής του ξεπεράσει το χρονικό όριο (timeout) του ενταμιευτή.

Όσο ο ενταμιευτής εισόδου περιέχει πακέτα προς αποστολή, ο κόμβος περιοδικά εκτελεί Ανακαλύψεις Διαδρομής για τον προορισμό, μέχρι όμως ενός ορίου, καθώς είναι πιθανόν να μην υπάρχουν διαδρομές προς αυτόν λόγω διαίρεσης του δικτύου σε επιμέρους μικρότερα δίκτυα. Ακόμα, ο ρυθμός με τον οποίο εκτελούνται Ανακαλύψεις Διαδρομής πρέπει να μειώνεται σταδιακά, χρησιμοποιώντας έναν αλγόριθμο εκθετικής υποχώρησης.

Μερικές επιπλέον λειτουργίες της Ανακάλυψης Διαδρομής είναι οι παρακάτω :

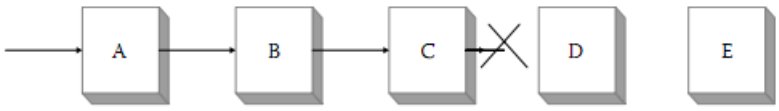
A) **Αποθήκευση πληροφοριών δρομολόγησης που κρυφακούγονται** : Ένας κόμβος που προωθεί ή κρυφακούει πακέτα, δηλαδή λαμβάνει πακέτα που προορίζονται για άλλο κόμβο, προσθέτει όλες τις χρήσιμες πληροφορίες από αυτά στη μνήμη διαδρομών του.

Β) **Απάντηση σε Αιτήσεις Διαδρομής με αποθηκευμένες διαδρομές** : Ένας κόμβος που λαμβάνει μια RREQ προς άλλο κόμβο αναζητά στη μνήμη διαδρομών για ήδη αποθηκευμένες διαδρομές προς αυτόν. Αν βρεθεί κάποια διαδρομή, ο κόμβος απαντά με RREP, αντί να προωθήσει τη RREQ.

Γ) **Περιορισμός βημάτων στις Αιτήσεις Διαδρομής** : Κάθε RREQ περιέχει ένα περιορισμό στον αριθμό των ενδιάμεσων κόμβων που μπορούν να επανεκπέμψουν το πακέτο. Ο περιορισμός υλοποιείται με το πεδίο Time-To-Live (TTL) της επικεφαλίδας IP.

3.4.2.2 Συντήρηση Διαδρομής

Κατά την αποστολή ή προώθηση πακέτων δεδομένων με χρήση πηγαίας διαδρομής, κάθε κόμβος είναι υπεύθυνος να επιβεβαιώνει ότι τα δεδομένα μπορούν να μεταδοθούν μέσω της ζεύξης από τον ίδιο προς τον επόμενο κόμβο. Για παράδειγμα, έστω ότι στο Σχήμα 3.2 ο κόμβος Α έστειλε ένα πακέτο προς τον κόμβο Ε χρησιμοποιώντας μια πηγαία διαδρομή μέσω των κόμβων Β, C, D. Στην περίπτωση αυτή ο κόμβος Α είναι υπεύθυνος για τη ζεύξη από τον Α στον Β, ο κόμβος Β για τη ζεύξη από τον Β στον C, ο κόμβος C για τη ζεύξη από τον C στον D και ο κόμβος D για τη ζεύξη από τον D στον Ε.



Σχήμα 3.2 : Συντήρηση Διαδρομής: αδυναμία προώθησης πακέτων από τον C στον D

Οι επιβεβαιώσεις συχνά παρέχονται χωρίς κόστος, είτε σαν υπάρχον τμήμα του πρωτοκόλλου MAC που χρησιμοποιείται είτε με «παθητική επιβεβαίωση», όπου για παράδειγμα ο Β επιβεβαιώνει τη λήψη από τον C, κρυφακούγοντας τον C να προωθεί το πακέτο στον D. Αν δεν υπάρχει ενσωματωμένος διαθέσιμος μηχανισμός επιβεβαιώσεων, τότε ο κόμβος που μεταδίδει τα δεδομένα, αποστέλλει και μια αίτηση επιβεβαίωσης προς τον επόμενο κόμβο. Μετά τη λήψη επιβεβαίωσης από κάποιο γειτονικό κόμβο, ο αποστολέας δεν απαιτεί επιβεβαιώσεις για κάποιο χρονικό διάστημα της επιλογής του.

Υπάρχει ένα όριο στον αριθμό επαναμεταδόσεων της αίτησης επιβεβαίωσης. Η επαναμετάδοση αποστέλλεται ως ένα αυτόνομο πακέτο ή μέσα σε πακέτο δεδομένων προς τον ίδιο προορισμό ή προς άλλο προορισμό που περνά όμως από τον εν λόγω κόμβο. Αν το όριο επαναμεταδόσεων ξεπεραστεί χωρίς λήψη επιβεβαίωσης, τότε ο αποστολέας θεωρεί τη ζεύξη προς το γειτονικό κόμβο «σπασμένη» (broken). Αφαιρεί από τη μνήμη διαδρομών του κάθε διαδρομή που περιέχει το γειτονικό κόμβο, ενώ στέλνει βλάβη διαδρομής (Route Error – RERR) σε κάθε κόμβο που έχει στείλει πακέτο με διαδρομή που περιέχει τον εν λόγω κόμβο.

Για παράδειγμα, αν στο Σχήμα 3.2 ο κόμβος C δε λάβει επιβεβαίωση από τον D μετά από έναν ορισμένο αριθμό επαναμεταδόσεων, επιστρέφει βλάβη διαδρομής προς τον

Α και κάθε άλλο κόμβο που χρησιμοποιούσε τη ζεύξη από τον C προς τον D. Ο κόμβος Α διαγράφει τη ζεύξη αυτή από τη μνήμη διαδρομών του και για να στείλει πακέτα προς τον Ε χρησιμοποιεί άλλες διαδρομές που πιθανόν έχει αποθηκεύσει, αλλιώς ξεκινά μια νέα Ανακάλυψη Διαδρομής.

Μερικές επιπλέον λειτουργίες της Συντήρησης Διαδρομής είναι οι παρακάτω :

Α) **Διάσωση πακέτων** : Αν κάποιος ενδιάμεσος κόμβος αντιληφθεί μέσω της Συντήρησης Διαδρομής ότι η ζεύξη προς τον επόμενο κόμβο έχει σπάσει, αναζητά στη μνήμη διαδρομών για εναλλακτική διαδρομή προς τον προορισμό του πακέτου, ώστε να «διασώσει» το πακέτο προωθώντας το, αντί να το απορρίψει.

Β) **Αποθηκευμένα σε ουρά πακέτα με διαδρομή μέσω σπασμένης ζεύξης** : Ο κόμβος πρέπει να χειριστεί ομοίως και αποθηκευμένα σε ενταμιευτές πακέτα που κανονικά θα αποστέλλονταν μέσω της σπασμένης ζεύξης, δηλαδή να αφαιρέσει το πακέτο από τον ενταμιευτή, να στείλει RERR στον αρχικό αποστολέα και να επιχειρήσει διάσωση του κάθε πακέτου.

Γ) **Αυτόματη συντόμευση διαδρομής** : Οι πηγαίες διαδρομές που χρησιμοποιούνται είναι πιθανόν να συντομευθούν, αν κάποιος ενδιάμεσος κόμβος δεν είναι πλέον αναγκαίος. Ειδικότερα, ένας κόμβος που κρυφακούει ένα πακέτο με πηγαία διαδρομή, εξετάζει αν ο εαυτός του υπάρχει στην πηγαία διαδρομή μετά από άλλους παρεμβαλλόμενους κόμβους. Στην περίπτωση αυτή θεωρεί ότι οι ενδιάμεσοι αυτοί κόμβοι δεν είναι πλέον αναγκαίοι και στέλνει μια απρόκλητη (gratuitous) RREP στον αρχικό αποστολέα, εμπεριέχοντας τη νέα συντομότερη διαδρομή.

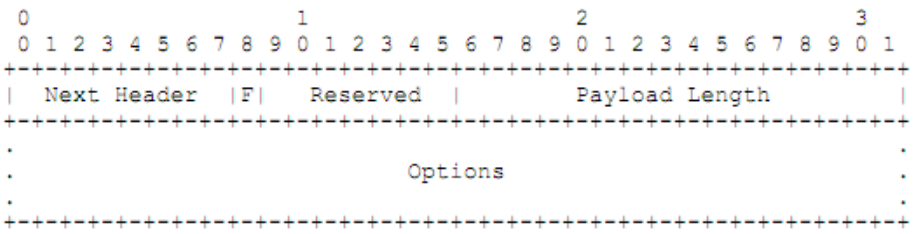
Δ) **Εξάπλωση των βλαβών διαδρομής** : Ένας κόμβος που λαμβάνει RERR για πακέτο που ο ίδιος αρχικά έστειλε, διαδίδει τη RERR στους γειτονικούς κόμβους, εμπεριέχοντάς τη στην επόμενη RREQ. Έτσι, κάθε γειτονικός κόμβος θα ανανεώσει τη μνήμη διαδρομών του, ώστε να μην περιέχει τη σπασμένη ζεύξη.

3.4.3 Μορφή επικεφαλίδας Επιλογών DSR

Το πρωτόκολλο DSR χρησιμοποιεί μια ειδική επικεφαλίδα με πληροφορίες ελέγχου που μπορεί να συμπεριληφθεί σε οποιοδήποτε πακέτο IP. Αυτή η επικεφαλίδα Επιλογών DSR (DSR Options) περιέχει ένα μικρό τμήμα σταθερού μεγέθους 4 οκτάδων και μια ακολουθία από μηδέν ή περισσότερες επιλογές που μεταφέρουν προαιρετικές πληροφορίες. Στο IPv4, η επικεφαλίδα Επιλογών DSR ακολουθεί αμέσως μετά την επικεφαλίδα IP.

3.4.3.1 Σταθερό τμήμα επικεφαλίδας Επιλογών DSR

Το σταθερό τμήμα της επικεφαλίδας Επιλογών DSR μεταφέρει πληροφορίες που είναι απαραίτητες σε κάθε επικεφαλίδα Επιλογών DSR και έχει την ακόλουθη μορφή (Σχήμα 3.3).



Σχήμα 3.3 : Μορφή Επικεφαλίδας Επιλογών DSR

Τα πεδία της επικεφαλίδας αναλύονται στον παρακάτω Πίνακα 3.1.

Πεδίο	Σημασία
Next Header	Επιλογέας 8bit που προσδιορίζει την αμέσως επόμενη επικεφαλίδα
Flow State Header (F)	Σημαία που πρέπει να είναι 0
Reserved	Πρέπει να σταλεί ίσο με 0 και να αγνοηθεί στη λήψη
Payload Length	Το μήκος της επικεφαλίδας εκτός του σταθερού τμήματος 4 οκτάδων. Ορίζει το συνολικό μήκος των επιλογών που μεταφέρονται από την επικεφαλίδα
Options	Πεδίο μεταβλητού μήκους, που ορίζεται από το Payload Length. Περιέχει μία ή περισσότερες επιλογές DSR, κωδικοποιημένες με μορφή TLV (type-length-value)

Πίνακας 3.1 : Ανάλυση των πεδίων της επικεφαλίδας Επιλογών DSR

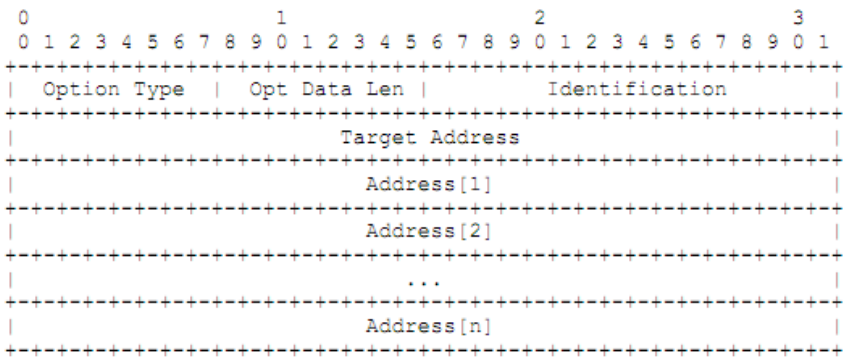
Λόγω του περιορισμένου διαθέσιμου εύρους ζώνης στα MANET οι επιλογές εντός της επικεφαλίδας δεν ευθυγραμμίζονται με παραγέμισμα. Ορίζονται τα επόμενα είδη επιλογών :

- Επιλογή αίτησης διαδρομής (Route Request option - RREQ)
- Επιλογή απάντησης διαδρομής (Route Reply option - RREP)
- Επιλογή βλάβης διαδρομής (Route Error option - RERR)
- Επιλογή αίτησης επιβεβαίωσης (Acknowledgement Request option – ACK REQ)
- Επιλογή επιβεβαίωσης (Acknowledgement option - ACK)
- Επιλογή πηγαίας διαδρομής DSR (DSR Source Route option)
- Επιλογή παραγεμίσματος 1 (Pad1 option)
- Επιλογή παραγεμίσματος N (PadN option)

Στη συνέχεια ακολουθεί αναλυτική περιγραφή των επιλογών αυτών.

3.4.3.2 Επιλογή αίτησης διαδρομής (RREQ)

Η επιλογή αίτησης διαδρομής παρουσιάζεται στο Σχήμα 3.4 και αναλύεται στον Πίνακα 3.2. Δεν πρέπει να υπάρχει πάνω από μία φορά στην επικεφαλίδα.



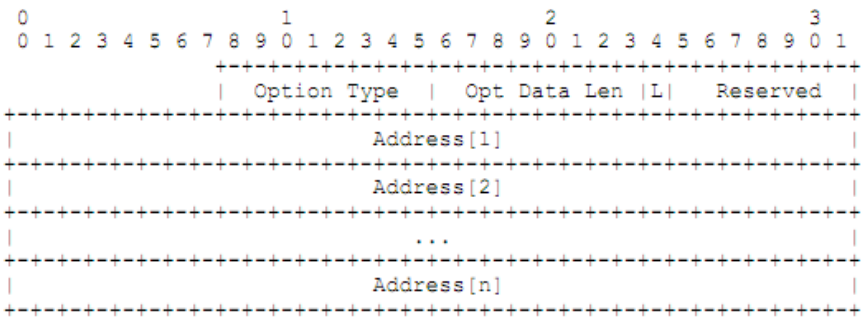
Σχήμα 3.4 : Μορφή επιλογής RREQ

Πεδίο	Σημασία
Source Address (πεδίο IP)	Η διεύθυνση του κόμβου που δημιούργησε το πακέτο. Ενδιάμεσοι κόμβοι που επαναμεταδίδουν το πακέτο δεν πρέπει να τροποποιήσουν αυτό το πεδίο
Destination Address (πεδίο IP)	Η διεύθυνση ευρυεκπομπής IP (255.255.255.255)
Hop Limit (TTL) (πεδίο IP)	Παίρνει τιμές από 1 έως 255
Option Type	Ισούται με 1. Κωδικός που αν δε καταλαβαίνει ο παραλήπτης, αγνοεί την επιλογή
Opt Data Len	Το μήκος της επιλογής σε οκτάδες, χωρίς τα πεδία Option Type και Opt Data Len. Ισούται με (4*n)+6, όπου n οι διευθύνσεις που περιέχονται στην RREQ
Identification	Μοναδική τιμή για κάθε RREQ, που δημιουργείται από τον αρχικό αποστολέα της RREQ. Ο παραλήπτης ξέρει αν έχει πρόσφατα λάβει την ίδια RREQ, αναζητώντας την τιμή αυτή στον πίνακα RREQ, και αν τη βρει απορρίπτει τη RREQ
Target Address	Η διεύθυνση του τελικού προορισμού της RREQ
Address [1..n]	Η Address[i] είναι η διεύθυνση του i-στού κόμβου από τον οποίο πέρασε η RREQ. Η διεύθυνση της πηγής δεν περιέχεται σε αυτή τη λίστα, αλλά στο πεδίο Source Address του πεδίου IP. Κάθε κόμβος που επανεκπέμπει τη RREQ προσθέτει τη διεύθυνσή του στη λίστα

Πίνακας 3.2 : Ανάλυση των πεδίων της επιλογής RREQ

3.4.3.3 Επιλογή απάντησης διαδρομής (RREP)

Η επιλογή απάντησης διαδρομής παρουσιάζεται στο Σχήμα 3.5 και αναλύεται στον Πίνακα 3.3. Πιθανόν να υπάρχει πάνω από μία φορά στην επικεφαλίδα.



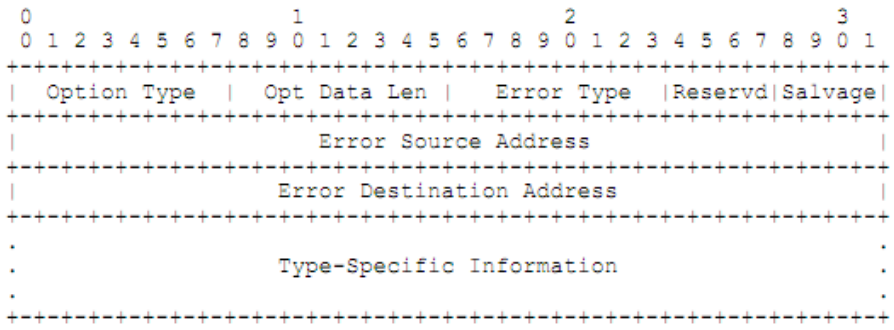
Σχήμα 3.5 : Μορφή επιλογής RREP

Πεδίο	Σημασία
Source Address (πεδίο IP)	Η διεύθυνση του κόμβου που στέλνει τη RREP. Πιθανόν να διαφέρει από τον προορισμό της Ανακάλυψης διαδρομής σε περίπτωση απάντησης από τη μνήμη διαδρομών ή απρόκλητης απάντησης
Destination Address (πεδίο IP)	Η διεύθυνση της πηγής της διαδρομής που επιστρέφεται
Option Type	Ισούται με 2. Κωδικός που αν δε καταλαβαίνει ο παραλήπτης, αγνοεί την επιλογή
Opt Data Len	Το μήκος της επιλογής σε οκτάδες, χωρίς τα πεδία Option Type και Opt Data Len. Ισούται με (4*n)+1, όπου n οι διευθύνσεις που περιέχονται στην RREP
Last Hop External (L)	Δείχνει ότι το τελευταίο βήμα της RREP (από τον Address[n-1] στον Address[n]) είναι ένα μονοπάτι σε ένα εξωτερικό προς το DSR δίκτυο. Η ακριβής διαδρομή εκτός του DSR δικτύου δε φαίνεται στη RREP
Reserved	Ισούται με 0 και αγνοείται κατά τη λήψη
Address [1..n]	Η πηγαία διαδρομή που επιστρέφεται από τη RREP. Η διαδρομή είναι μια ακολουθία βημάτων με πηγή τον κόμβο Destination Address και επόμενα βήματα τις διευθύνσεις Address[1] έως Address[n-1] και προορισμό τον Address[n]

Πίνακας 3.3 : Ανάλυση των πεδίων της επιλογής RREP

3.4.3.4 Επιλογή βλάβης διαδρομής (RERR)

Η επιλογή απάντησης διαδρομής παρουσιάζεται στο Σχήμα 3.6 και αναλύεται στον Πίνακα 3.4. Πιθανόν να υπάρχει πάνω από μία φορά στην επικεφαλίδα.



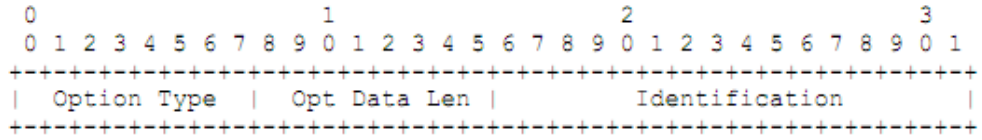
Σχήμα 3.6 : Μορφή επιλογής RERR

Πεδίο	Σημασία
Option Type	Ισούται με 3. Κωδικός που αν δε καταλαβαίνει ο παραλήπτης, αγνοεί την επιλογή
Opt Data Len	Το μήκος της επιλογής σε οκτάδες, χωρίς τα πεδία Option Type και Opt Data Len. Ισούται με 10 συν το μήκος του πεδίου Type-Specific Information
Error Type	Ο τύπος της βλάβης παίρνει τις ακόλουθες τιμές : 1 = Απρόσιτος κόμβος 2 = Μη υποστηριζόμενη Κατάσταση Ροής (Flow State) 3 = Μη υποστηριζόμενη επιλογή
Reservd	Ισούται με 0 και αγνοείται κατά τη λήψη
Salvage	Αριθμός συνολικών διασώσεων
Error Source Address	Ο κόμβος - πηγή της βλάβης διαδρομής
Error Destination Address	Ο κόμβος – προορισμός της βλάβης διαδρομής
Type-Specific Information	Πληροφορία σχετική με το Error Type

Πίνακας 3.4 : Ανάλυση των πεδίων της επιλογής RERR

3.4.3.5 Επιλογή αίτησης επιβεβαίωσης (ACK REQ)

Η επιλογή αίτησης επιβεβαίωσης παρουσιάζεται στο Σχήμα 3.7 και αναλύεται στον Πίνακα 3.5. Δεν πρέπει να υπάρχει πάνω από μία φορά στην επικεφαλίδα.



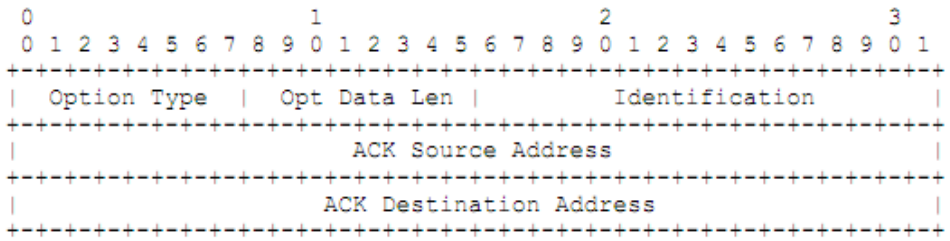
Σχήμα 3.7 : Μορφή επιλογής ACK REQ

Πεδίο	Σημασία
Option Type	Ισούται με 160. Κωδικός που αν δε καταλαβαίνει ο παραλήπτης, αφαιρεί την επιλογή και επιστρέφει RERR
Opt Data Len	Το μήκος της επιλογής σε οκτάδες, χωρίς τα πεδία Option Type και Opt Data Len
Identification	Μοναδική τιμή που αντιγράφεται στο αντίστοιχο πεδίο της επιλογής ACK που επιστρέφεται από τον παραλήπτη του ACK REQ μέσω της ίδιας ζεύξης

Πίνακας 3.5 : Ανάλυση των πεδίων της επιλογής ACK REQ

3.4.3.6 Επιλογή επιβεβαίωσης (ACK)

Η επιλογή επιβεβαίωσης παρουσιάζεται στο Σχήμα 3.8 και αναλύεται στον Πίνακα 3.6. Δεν πρέπει να υπάρχει πάνω από μία φορά στην επικεφαλίδα.



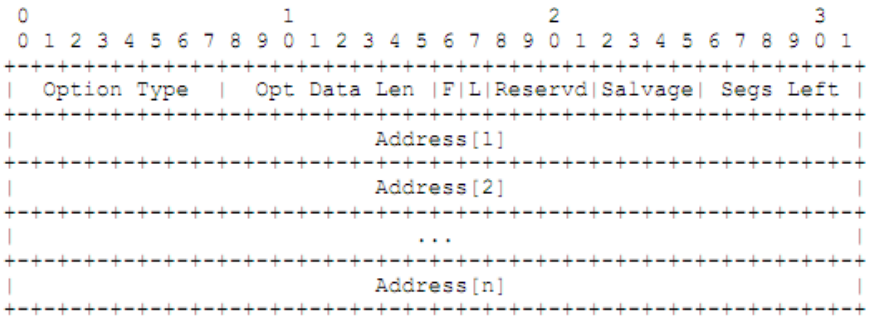
Σχήμα 3.8 : Μορφή επιλογής ACK

Πεδίο	Σημασία
Option Type	Ισούται με 32. Κωδικός που αν δε καταλαβαίνει ο παραλήπτης, αφαιρεί την επιλογή
Opt Data Len	Το μήκος της επιλογής σε οκτάδες, χωρίς τα πεδία Option Type και Opt Data Len
Identification	Μοναδική τιμή που αντιγράφεται από το αντίστοιχο πεδίο της επιλογής ACK REQ
ACK Source Address	Ο κόμβος – πηγή της επιβεβαίωσης
ACK Destination Address	Ο κόμβος – προορισμός της επιβεβαίωσης

Πίνακας 3.6 : Ανάλυση των πεδίων της επιλογής ACK

3.4.3.7 Επιλογή πηγαίας διαδρομής (Source Route)

Η επιλογή πηγαίας διαδρομής παρουσιάζεται στο Σχήμα 3.9 και αναλύεται στον Πίνακα 3.7. Δεν πρέπει να υπάρχει πάνω από μία φορά στην επικεφαλίδα. Κατά την προώθηση πακέτων βάσει της επιλογής πηγαίας διαδρομής, το πεδίο προορισμού στην επικεφαλίδα IP ταυτίζεται με τον τελικό κόμβο της πηγαίας διαδρομής. Ο παραλήπτης της επιλογής αυτής εξετάζει την πηγαία διαδρομή και αν είναι το επόμενο βήμα, συνεχίζει με την προώθηση του πακέτου στον επόμενο κόμβο.



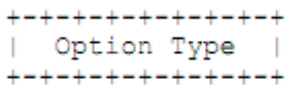
Σχήμα 3.9 : Μορφή επιλογής Source Route

Πεδίο	Σημασία
Option Type	Ισούται με 96. Κωδικός που αν δε καταλαβαίνει ο παραλήπτης, απορρίπτει το πακέτο
Opt Data Len	Το μήκος της επιλογής σε οκτάδες, χωρίς τα πεδία Option Type και Opt Data Len. Ισούται με (4*n)+2, όπου n το πλήθος των διευθύνσεων στη λίστα Address[n]
First Hop External (F)	Δείχνει ότι το πρώτο βήμα της Source Route είναι ένα μονοπάτι σε ένα εξωτερικό προς το DSR δίκτυο. Η ακριβής διαδρομή εκτός του DSR δικτύου δε φαίνεται στην επιλογή
Last Hop External (L)	Δείχνει ότι το τελευταίο βήμα της Source Route είναι ένα μονοπάτι σε ένα εξωτερικό προς το DSR δίκτυο. Η ακριβής διαδρομή εκτός του DSR δικτύου δε φαίνεται στην επιλογή
Reservd	Ισούται με 0 και αγνοείται κατά τη λήψη
Salvage	Αριθμός συνολικών διασώσεων
Segs Left	Αριθμός υπόλοιπων τμημάτων διαδρομής, δηλαδή των ενδιάμεσων κόμβων θα διασχισθούν μέχρι την άφιξη στον τελικό προορισμό
Address[1..n]	Η ακολουθία διευθύνσεων της πηγαίας διαδρομής. Επεξεργάζεται κατάλληλα κατά τη δρομολόγηση και προώθηση των πακέτων

Πίνακας 3.7 : Ανάλυση των πεδίων της επιλογής Source Route

3.4.3.8 Επιλογές παραγεμίσματος (Pad1 και PadN)

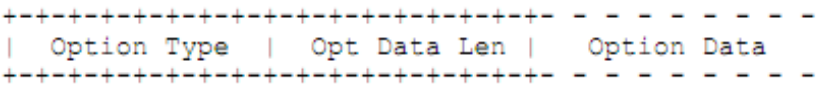
Οι επιλογές παραγεμίσματος 1 και N παρουσιάζονται στα Σχήματα 3.10 και 3.11 και αναλύονται στους Πίνακες 3.8 και 3.9 αντίστοιχα. Οι επιλογές παραγεμίσματος χρησιμοποιούνται, χωρίς να είναι απαραίτητες, για την ευθυγράμμιση διαδοχικών επιλογών DSR, ώστε το τελικό μήκος της επικεφαλίδας να είναι πολλαπλάσιο των 4 οκτάδων. Αν για την ευθυγράμμιση πρέπει να εισαχθούν πάνω από μία διαδοχικές οκτάδες, χρησιμοποιείται η επιλογή PadN.



Σχήμα 3.10 : Μορφή επιλογής Pad1

Πεδίο	Σημασία
Option Type	Ισούται με 224. Κωδικός που αν δε καταλαβαίνει ο παραλήπτης, απορρίπτει το πακέτο και επιστρέφει RERR

Πίνακας 3.8 : Ανάλυση των πεδίων της επιλογής Pad1



Σχήμα 3.11 : Μορφή επιλογής PadN

Πεδίο	Σημασία
Option Type	Ισούται με 0. Κωδικός που αν δε καταλαβαίνει ο παραλήπτης, αγνοεί την επιλογή
Opt Data Len	Το μήκος της επιλογής σε οκτάδες, χωρίς τα πεδία Option Type και Opt Data Len. Ισούται με το μέγεθος του πεδίου Opt Data
Opt Data	Αριθμός οκτάδων με τιμή 0, που ισούται με το Opt Data Len

Πίνακας 3.9 : Ανάλυση των πεδίων της επιλογής PadN

3.4.4 Θεμελιώδεις Δομές Δεδομένων

Στο κεφάλαιο αυτό περιγράφονται οι θεμελιώδεις δομές δεδομένων που υπάρχουν στο πρωτόκολλο DSR και η χρήση τους.

Μνήμη Διαδρομών (Route Cache)

Κάθε κόμβος έχει μνήμη διαδρομών, όπου αποθηκεύει χρήσιμες πληροφορίες για τη δρομολόγηση των πακέτων. Πληροφορίες προστίθενται όταν ο κόμβος μαθαίνει για νέες ζεύξεις μέσω των επιλογών RREQ, RREP, Source Route και διαγράφονται όταν μαθαίνει για ζεύξεις που πλέον δεν υφίστανται μέσω της επιλογής RERR ή του μηχανισμού επιβεβαιώσεων, δηλαδή αναπάντητη αίτηση επιβεβαίωσης. Κατά την προσθήκη νέων δεδομένων στη μνήμη διαδρομών, ο κόμβος πρέπει να ελέγχει κάθε πακέτο στον ενταμιευτή αποστολής (Send Buffer) και σε περίπτωση που υπάρχει

διαδρομή για τον προορισμό κάποιου πακέτου, να το στέλνει, σβήνοντάς το ταυτόχρονα από τον ενταμιευτή.

Η μνήμη διαδρομών υποστηρίζει την αποθήκευση πολλαπλών διαδρομών για κάθε προορισμό. Η αναζήτηση και επιλογή της βέλτιστης διαδρομής γίνεται με βάση το κριτήριο της ελάχιστης διαδρομής, δηλαδή επιλέγεται η διαδρομή με τα λιγότερα ενδιάμεσα βήματα. Το μέγεθος της μνήμης συνήθως είναι σταθερό, οπότε αν χρειαστεί να ελευθερωθεί χώρος, διαγράφεται η τελευταία διαδρομή με βάση τον αριθμό βημάτων, και σε περίπτωση ισότητας, η διαδρομή που χρησιμοποιήθηκε λιγότερο πρόσφατα (Least Recently Used – LRU). Ακόμα, υπάρχει ένα χρονικό όριο (timeout) που αν η διαδρομή ξεπεράσει χωρίς να χρησιμοποιηθεί, διαγράφεται.

Ενταμιευτής Αποστολής (Send Buffer)

Ο ενταμιευτής αποστολής ενός κόμβου είναι μια ουρά πακέτων που αποθηκεύονται επειδή δεν είναι δυνατόν να σταλούν, καθώς δεν υπάρχει ακόμα πηγαία διαδρομή για τον προορισμό του κάθε πακέτου. Κάθε πακέτο είναι συσχετισμένο με το χρόνο εγγραφής του στον ενταμιευτή και απορρίπτεται αν ο χρόνος παραμονής του ξεπεράσει το χρονικό όριο (timeout) του ενταμιευτή αποστολής. Ανά τακτά χρονικά διαστήματα εκτελούνται Ανακαλύψεις Διαδρομής για τους προορισμούς των πακέτων που υπάρχουν στον ενταμιευτή.

Πίνακας Ανακάλυψης Διαδρομής (Route Discovery Table)

Ο πίνακας Ανακάλυψης Διαδρομής ενός κόμβου αποθηκεύει πληροφορίες για αιτήσεις διαδρομών (RREQ) που έχουν πρόσφατα ξεκινήσει ή προωθηθεί από τον κόμβο αυτό. Το κλειδί του πίνακα είναι οι διευθύνσεις IP. Έτσι, υπάρχει έλεγχος στη διάδοση των RREQ, μέσω των πεδίων Time-To-Live (TTL), το μέγιστο αριθμό Ανακαλύψεων Διαδρομής για συγκεκριμένο προορισμό, και το ρυθμό επανάληψης των Ανακαλύψεων Διαδρομής σε περίπτωση που δε λαμβάνεται απάντηση.

Πίνακας Απρόκλητων Απαντήσεων Διαδρομής (Gratuitous Route Reply Table)

Ο πίνακας αυτός διατηρεί πληροφορίες για απρόκλητες RREP που έστειλε ο κόμβος σαν αποτέλεσμα της Αυτόματης Συντόμευσης Διαδρομής. Χρησιμεύει στον περιορισμό του ρυθμού με τον οποίο στέλνονται απρόκλητες RREP στον ίδιο αρχικό αποστολέα για τον ίδιο κόμβο. Για κάθε κόμβο στον οποίο έστειλε απρόκλητο RREP δημιουργείται μια εγγραφή, που διαγράφεται μετά από ένα χρονικό όριο, αποτρέποντας έτσι την αποστολή πολλαπλών απρόκλητων RREP προς τον ίδιο κόμβο.

Ενταμιευτής Συντήρησης (Maintenance Buffer)

Ο ενταμιευτής συντήρησης ενός κόμβου είναι μια ουρά πακέτων που έχουν σταλεί και περιμένουν την επιβεβαίωση για την προσβασιμότητα του επόμενου κόμβου στα πλαίσια της Συντήρησης Διαδρομής. Για κάθε πακέτο διατηρείται ο αριθμός επαναμεταδόσεων καθώς και ο χρόνος της τελευταίας επαναμετάδοσης. Σε περίπτωση που δε ληφθεί επιβεβαίωση μετά από ορισμένο αριθμό επαναμεταδόσεων,

τα πακέτα με το συγκεκριμένο επόμενο κόμβο διαγράφονται από τον ενταμιευτή, επιχειρείται διάσωσή τους, και στέλνονται βλάβες διαδρομής (RERR) στις πηγές των πακέτων αυτών.

Μαύρη Λίστα (Blacklist)

Η μαύρη λίστα είναι ένας πίνακας που περιέχει τους γειτονικούς κόμβους ενός κόμβου με τους οποίους δεν υπάρχει αμφίδρομη ζεύξη. Είναι απαραίτητη όταν η διεπαφή δικτύου του κόμβου απαιτεί αμφίδρομες ζεύξεις για εκπομπή προς έναν προορισμό (unicast).

3.5 Δρομολόγηση με ενεργειακά κριτήρια

Κάθε πρωτόκολλο δρομολόγησης σχεδιάζεται για κάποια εφαρμογή και χρησιμοποιεί τους δικούς του μηχανισμούς και κριτήρια δρομολόγησης. Στο πεδίο των MANET δημοσιεύονται πολλές εργασίες με νέα πρωτόκολλα ή βελτιώσεις των ήδη υπαρχόντων πρωτοκόλλων, που στοχεύουν στη βελτιστοποίηση κάποιου κριτηρίου.

Καθώς οι κινητοί κόμβοι ενός δικτύου MANET συνήθως λειτουργούν με μπαταρία, είναι σημαντική η ελαχιστοποίηση της ενεργειακής κατανάλωσης, ώστε να αυξηθεί ο χρόνος ζωής του δικτύου. Προς την κατεύθυνση αυτή μπορούν να υλοποιηθούν ιδέες σε όλα τα στρώματα πρωτοκόλλων, και πιο συγκεκριμένα έλεγχος ισχύος και διαχείριση κατάστασης ισχύος στο φυσικό στρώμα, ενεργειακά αποδοτικό MAC στο στρώμα ζεύξης δεδομένων, και ενεργειακά αποδοτική δρομολόγηση στο στρώμα δικτύου [8]. Στην παρούσα εργασία θα ασχοληθούμε μόνο με το στρώμα δικτύου και τη δρομολόγηση με βάση ενεργειακά κριτήρια. Στη συνέχεια παρουσιάζονται τέσσερις αλγόριθμοι δρομολόγησης που λαμβάνουν αποφάσεις έχοντας ως στόχο την ελαχιστοποίηση της κατανάλωσης ενέργειας ή την μεγιστοποίηση του χρόνου ζωής των κόμβων [4].

3.5.1 Δρομολόγηση ελάχιστης συνολικής ισχύος μετάδοσης (Minimum Total Transmission Power Routing – MTPR)

Στις ασύρματες επικοινωνίες, η διάδοση κυμάτων μοντελοποιείται αποτελεσματικά με τον παράγοντα μετάδοσης ισχύος $1/d^n$. Για να είναι επιτυχής η μετάδοση πρέπει ο σηματοθορυβικός λόγος (Signal-to-Noise Ratio – SNR) λήψης να είναι μεγαλύτερος από ένα ορισμένο κατώφλι, που επιλέγεται με βάση το ρυθμό λαθών ψηφίων (bit error rate – BER) του σήματος λήψης.

Επομένως, η ελάχιστη ισχύς εκπομπής εξαρτάται από το θόρυβο, τις παρεμβολές, την απόσταση μεταξύ των κόμβων και τον επιθυμητό BER. Για να εξασφαλιστεί η διαδρομή ελάχιστης συνολικής ισχύος, χρησιμοποιείται ως κριτήριο η ισχύς μετάδοσης $P(n_i, n_j)$ ανάμεσα στους κόμβους n_i και n_j, ώστε η συνολική ισχύς μετάδοσης P_l για τη διαδρομή l να είναι :

$$P_l = \sum_{i=0}^{D-1} P(n_i, n_{i+1}) \tag{3.1}$$

όπου n_0 και n_D είναι η πηγή και ο προορισμός αντίστοιχα. Έτσι, η επιθυμητή διαδρομή είναι :

$$P_k = \min_{l \in A} P_l \tag{3.2}$$

όπου A είναι το σύνολο όλων των πιθανών διαδρομών.

Η ανωτέρω συνάρτηση λύνεται εύκολα από ένα κλασικό αλγόριθμο ελάχιστης διαδρομής, επιλέγει όμως διαδρομές με περισσότερα βήματα από ότι άλλοι αλγόριθμοι, αυξάνοντας έτσι και την καθυστέρηση από άκρο σε άκρο. Για να ξεπεραστεί το πρόβλημα αυτό, μπορεί ως κριτήριο να χρησιμοποιηθεί το άθροισμα του κόστους λήψης και του κόστους μετάδοσης, ώστε να επιλέγονται διαδρομές με λιγότερους ενδιάμεσους κόμβους.

3.5.2 Δρομολόγηση ελάχιστου κόστους μπαταρίας (Minimum Battery Cost Routing – MBCR)

Η συνολική ισχύς μετάδοσης είναι ένα σημαντικό κριτήριο, καθώς μειώνει τη συνολική κατανάλωση ισχύος του δικτύου, χωρίς όμως να αυξάνει άμεσα το χρόνο ζωής κάθε κόμβου. Πιθανόν όλες οι διαδρομές που επιλέγονται να είναι μέσω ενός συγκεκριμένου κόμβου, οπότε αυτός θα σβήσει λόγω εξάντλησης της μπαταρίας του. Η υπολειπόμενη χωρητικότητα μπαταρίας κάθε κόμβου είναι ένα ακριβέστερο κριτήριο για την περιγραφή του χρόνου ζωής κάθε κόμβου.

Έστω c_i' η χωρητικότητα μπαταρίας του κόμβου n_i τη χρονική στιγμή t , με εύρος τιμών από 0 έως 100. Ορίζουμε ως $f_i(c_i')$ τη συνάρτηση κόσμους μπαταρίας του κόμβου n_i, που εκφράζει την απροθυμία του κόμβου να προωθήσει πακέτα συναρτήσει της υπολειπόμενης χωρητικότητας μπαταρίας. Όσο μικρότερη είναι η χωρητικότητα, τόσο μεγαλύτερη η απροθυμία, οπότε μια επιλογή είναι :

$$f_i(c_i') = \frac{1}{c_i'} \tag{3.3}$$

Το κόστος μπαταρίας R_j για τη διαδρομή j , που αποτελείται από D_j κόμβους είναι:

$$R_j = \sum_{i=0}^{D_j - 1} f_i(c_i') \tag{3.4}$$

Έτσι, για την εύρεση της διαδρομής με τη μέγιστη υπολειπόμενη χωρητικότητα μπαταρίας, επιλέγουμε τη διαδρομή i με το ελάχιστο κόστος μπαταρίας :

$$R_i = \min\{R_j \mid j \in A\} \tag{3.5}$$

Το κριτήριο αυτό αποτρέπει την υπερβολική χρήση κάποιων κόμβων, επιμηκύνοντας έτσι το χρόνο ζωής του δικτύου και τη στιγμή διαίρεσης του δικτύου. Το πρόβλημα

εντοπίζεται στο ότι λαμβάνεται υπ' όψιν μόνο το άθροισμα των τιμών των συναρτήσεων κόστους μπαταρίας, οπότε είναι πιθανόν και πάλι να επιλέγονται διαδρομές με κόμβους που έχουν λίγη υπολειπόμενη χωρητικότητα.

3.5.3 Δρομολόγηση ελαχίστου-μεγίστου κόστους μπαταρίας (Min-Max Battery Cost Routing – MMBCR)

Για να εξασφαλιστεί ότι κανένας κόμβος δε θα χρησιμοποιείται υπερβολικά, τροποποιείται το κόστος μπαταρίας R_j για τη διαδρομή j ως εξής :

$$R_j = \max_{i \in j} \left\{ f_i \left(c_i' \right) \right\} \tag{3.6}$$

Έτσι, με βάση την εξίσωση (3.5) επιλέγεται η διαδρομή με το ελάχιστο μέγιστο κόστος μπαταρίας, αποφεύγοντας τους κόμβους με την ελάχιστη χωρητικότητα μπαταρίας ανάμεσα σε όλους τους κόμβους όλων των πιθανών διαδρομών. Επομένως, προκύπτει πιο δίκαια χρησιμοποίηση της μπαταρίας κάθε κόμβου, δεν υπάρχει όμως η εγγύηση ότι θα επιλεγούν μονοπάτια ελάχιστης συνολικής ισχύος μετάδοσης, οπότε πιθανόν να καταναλώνεται τελικά μεγαλύτερη ισχύς για τη μετάδοση από την πηγή στον προορισμό, μειώνοντας το χρόνο ζωής όλων των κόμβων.

3.5.4 Δρομολόγηση υπό συνθήκη μέγιστης-ελάχιστης χωρητικότητας μπαταρίας (Conditional Max-Min Battery Cost Routing – CMMBCR)

Ο διπλός στόχος μεγιστοποίησης του χρόνου ζωής κάθε κόμβου και δίκαιης χρησιμοποίησης της μπαταρίας δεν επιτυγχάνεται από τους προηγούμενους αλγορίθμους, εκτός μόνο από τον MBCR κάτω από ειδικές συνθήκες (όλοι οι κόμβοι έχουν ίδια ισχύ εκπομπής και ίδια χωρητικότητα μπαταρίας). Για να επιτευχθεί ο στόχος, χρησιμοποιείται ως κριτήριο δρομολόγησης η χωρητικότητα της μπαταρίας αντί της συνάρτησης κόστους.

Η βασική ιδέα είναι ότι όταν όλοι οι κόμβοι στις πιθανές διαδρομές ανάμεσα στην πηγή και τον προορισμό έχουν επαρκή υπολειπόμενη χωρητικότητα μπαταρίας, επιλέγεται η διαδρομή με την ελάχιστη συνολική ισχύ μετάδοσης ανάμεσά τους. Αν όλες οι διαδρομές έχουν κόμβους με χαμηλή υπολειπόμενη χωρητικότητα μπαταρίας, οι διαδρομές που περιέχουν κόμβους με τη χαμηλότερη χωρητικότητα αποφεύγονται, ώστε να επιμηκυνθεί ο χρόνος ζωής των κόμβων. Ορίζουμε τη χωρητικότητα μπαταρίας R_j' για τη διαδρομή j τη χρονική στιγμή t ως :

$$R_j' = \min_{i \in j} \left\{ c_i' \right\} \tag{3.7}$$

Έστω A το σύνολο όλων των πιθανών διαδρομών j ανάμεσα σε δύο κόμβους τη χρονική στιγμή t που ικανοποιούν την ανισότητα :

$$R'_j \geq \gamma, \forall j \in A \tag{3.8}$$

όπου γ είναι ένα κατώφλι με εύρος τιμών από 0 έως 100.

Η διαδρομή επιλέγεται με βάση τη συνθήκη (3.8) :

- Αν υπάρχουν διαδρομές που ικανοποιούν την (3.8), δηλαδή διαδρομές όπου όλοι οι κόμβοι έχουν υπολειπόμενη χωρητικότητα μπαταρίας μεγαλύτερη του κατωφλίου γ, επιλέγεται η διαδρομή με βάση το κριτήριο MTPR.
- Αλλιώς, επιλέγεται η διαδρομή με τη μέγιστη υπολειπόμενη χωρητικότητα μπαταρίας.

Αν $\gamma = 0$, τότε η ανίσωση (3.8) είναι πάντα αληθής, και το κριτήριο λειτουργεί όπως το MTPR. Αν $\gamma = 100$, τότε η ανίσωση είναι πάντα ψευδής, και το κριτήριο λειτουργεί όπως το MMBCR, αφού οι διαδρομές με μικρότερη υπολειπόμενη χωρητικότητα μπαταρίας πάντα αποφεύγονται, με το γ να λειτουργεί σαν ένα περιθώριο προστασίας. Επομένως, η επίδοση του CCMBCR εξαρτάται από την τιμή του κατωφλίου γ.

Οι προσομοιώσεις που εκτελέστηκαν στην εργασία [4] υποδεικνύουν τελικά ότι οι δύο στόχοι της μεγιστοποίησης του χρόνου ζωής κάθε κόμβου και της δίκαιης χρησιμοποίησης της μπαταρίας τους δεν μπορούν να επιτευχθούν ταυτόχρονα, αλλά υπάρχει ένα tradeoff μεταξύ τους. Ανάλογα, λοιπόν, με την εφαρμογή και τους στόχους του σχεδιαστή ενός δικτύου, προσαρμόζεται η τιμή του γ ώστε ο αλγόριθμος να επικεντρώνεται στον ένα εκ των δύο στόχων.

3.6 Επισκόπηση πρωτοκόλλων δρομολόγησης MANET με ενεργειακά κριτήρια

Η βιβλιογραφία στο πεδίο των πρωτοκόλλων δρομολόγησης με κριτήρια που αφορούν την ενεργειακή κατανάλωση είναι εκτενής. Έχει προταθεί πλήθος μηχανισμών, αλγορίθμων και πρωτοκόλλων που διαχειρίζονται την δρομολόγηση με στόχο την ελαχιστοποίηση της κατανάλωσης ενέργειας ή την μεγιστοποίηση του χρόνου ζωής του δικτύου.

Τα ενεργειακά ενήμερα (energy-aware) πρωτόκολλα είναι κυρίως επεκτάσεις των πιο διαδεδομένων πρωτοκόλλων δρομολόγησης, δηλαδή των AODV, DSR, OLSR και TORA. Υπάρχουν, ωστόσο, και πρωτόκολλα που δημιουργήθηκαν εξ ολοκλήρου για να χρησιμοποιηθούν σε αυτόν τον τομέα. Κάποιες ιδέες που έχουν προταθεί χρησιμοποιούν τεχνικές όπως ο έλεγχος τοπολογίας, η γνώση της τοποθεσίας (location-aware), η ομαδοποίηση (clustering), τα πλέγματα (grid), τα πολλαπλά μονοπάτια (multipath) ή απλά αποφασίζουν με ενεργειακά κριτήρια δρομολόγησης (routing metrics). Στη συνέχεια, θα γίνει μια σύντομη αναφορά σε κάποια ενδεικτικά πρωτόκολλα που έχουν υλοποιηθεί με βάση το DSR.

1) Power Aware DSR (PADSR)

Το πρωτόκολλο PADSR [9] χρησιμοποιεί τον αλγόριθμο Location Aided Power Aware Routing (LAPAR), που προσδιορίζει τη βέλτιστη διαδρομή ανάμεσα σε ένα ζεύγος πηγής – προορισμού ελαχιστοποιώντας την απαιτούμενη συνολική ισχύ μετάδοσης. Με την προϋπόθεση ότι κάθε κόμβος γνωρίζει τη θέση του, θεωρείται ότι η ελάχιστη ισχύς μετάδοσης προκύπτει από την ελάχιστη απόσταση μεταξύ των κόμβων. Έτσι, κάθε κόμβος σχεδιάζει έναν επίπεδο γράφο (planar graph), ο οποίος ενώνει τον εαυτό του με τους κόμβους για τους οποίους ελαχιστοποιείται η απαιτούμενη ισχύς για απευθείας μετάδοση. Οι πληροφορίες για τον σχηματισμό του επίπεδου γράφου λαμβάνονται με ευρυεκπομπή. Οι αιτήσεις διαδρομής (RREQ) αποστέλλονται από την πηγή προς τους κόμβους του επίπεδου γράφου της, και προωθούνται με τον ίδιο τρόπο από τους ενδιάμεσους κόμβους μέχρι να φτάσουν στον προορισμό τους. Μετά τη λήψη των απαντήσεων διαδρομής (RREP), επιλέγεται η διαδρομή ελάχιστης ισχύος.

2) Extended Max-Min DSR (EMM-DSR)

Το πρωτόκολλο EMM-DSR [10] έχει ως στόχο να εξισορροπήσει την αύξηση του χρόνου ζωής του δικτύου με τις καλές επιδόσεις διέλευσης και καθυστέρησης από άκρο σε άκρο. Ο μηχανισμός του είναι μια επέκταση του αλγορίθμου Max-Min, ώστε να υποστηρίζεται το tradeoff ανάμεσα στην ενεργειακή αποδοτικότητα και τη συντομία του μονοπατιού. Ο αλγόριθμος Max-Min χρησιμοποιεί ως κριτήριο την υπολειπόμενη ενέργεια των κόμβων, οπότε έχοντας στη διάθεσή του την ελάχιστη ενέργεια κάθε διαδρομής, δηλαδή την ενέργεια του κόμβου με την ελάχιστη ενέργεια σε αυτή τη διαδρομή, επιλέγει τη διαδρομή με τη μέγιστη ελάχιστη ενέργεια. Έτσι επιλέγονται, όμως, και διαδρομές με μεγάλο αριθμό βημάτων, γι' αυτό η επέκταση προτείνει σε περίπτωση σχεδόν ίσων ελαχίστων ενεργειών, το κριτήριο να είναι τα λιγότερα ενδιάμεσα βήματα. Η συνθήκη που ελέγχεται για τις διαδρομές *i* και *j* είναι:

$$\text{Αν } \left(\text{MinEnergy}_j - \text{MinEnergy}_i < \varepsilon\right) \text{ και } \left(\text{Hops}_j > \text{Hops}_i\right) \Rightarrow \text{Επίλεξε τη διαδρομή } i \quad (3.9)$$

όπου *MinEnergy* είναι η ελάχιστη ενέργεια της διαδρομής, *Hops* είναι ο αριθμός βημάτων της διαδρομής και ε είναι ένας μικρός θετικός αριθμός που ορίζει τη μικρή διαφορά μεταξύ των ενεργειών δύο διαδρομών. Η υπολειπόμενη ενέργεια των κόμβων μεταδίδεται μέσω της εκτεταμένης επικεφαλίδας των πακέτων RREQ, RREP.

3) Energy Dependent DSR (EDDSR)

Το πρωτόκολλο EDDSR [11] προτείνει ως κριτήριο την προθυμία του κόμβου να συμμετάσχει στην προώθηση πακέτων βάσει του τρέχοντος επιπέδου ενέργειας και στοχεύει στην αύξηση της διάρκειας ζωής του δικτύου. Κάθε κόμβος n_i του δικτύου υπολογίζει περιοδικά την υπολειπόμενη ισχύ μπαταρίας του, RBP_i (Residual Battery Power). Αν πέσει κάτω από ένα όριο, τότε ο κόμβος καθυστερεί την επανεκπομπή του RREQ για ένα διάστημα αντιστρόφως ανάλογο της προβλεπόμενης διάρκειας

ζωής του, η οποία εκφράζεται από το λόγο $\frac{RBP_i}{DR_i}$, όπου DR_i (Drain Rate) είναι μια εκτίμηση της μέσης καταναλισκόμενης ενέργειας ανά δευτερόλεπτο. Ακόμα, κατά τη συντήρηση διαδρομής, αν η ενέργεια ενός κόμβου ενεργής διαδρομής πέσει κάτω από ένα όριο, τότε στέλνει ένα πακέτο RERR στην πηγή, ώστε να επιλέξει άλλη διαδρομή προς τον προορισμό. Τέλος, η μνήμη διαδρομών (route cache) χρησιμοποιείται όπως στο πρωτόκολλο LEAR (Local Energy-Aware Routing), όπου χρησιμοποιούνται πακέτα Route Cache (RCAC), που στέλνονται προς τον προορισμό από κάποιον ενδιάμεσο κόμβο που βρίσκει στη μνήμη του αποθηκευμένη διαδρομή προς αυτόν. Τα RCAC προωθούνται μόνο αν η RBP_i είναι πάνω από ένα κατώφλι, αλλιώς στέλνονται τα πακέτα Drop Route Cache (DRCAC) προς τον προορισμό και Cancel Route Cache (CRCAC) προς την πηγή. Το πρώτο ενημερώνει τους κόμβους για το απορριφθέν πακέτο, ενώ το δεύτερο ανανεώνει τη μνήμη διαδρομών της πηγής.

Κεφάλαιο 4

Υλοποίηση και Αποτελέσματα

4.1 Εισαγωγή

Στα πλαίσια της διπλωματικής αυτής αναπτύχθηκε μια επέκταση του πρωτοκόλλου DSR, ώστε η δρομολόγηση να γίνεται με ενεργειακά κριτήρια. Το νέο πρωτόκολλο ονομάστηκε Energy-aware DSR (EDSR) και ως στόχος τέθηκε η αύξηση του χρόνου ζωής του δικτύου.

Η υλοποίηση της ιδέας έγινε στο λογισμικό προσομοίωσης δικτύων OPNET Modeler v14.5. Το πρόγραμμα αυτό υλοποιεί προσομοίωση διακριτών γεγονότων (discrete event simulation), δημιουργώντας ένα περιβάλλον τεχνητού λειτουργικού συστήματος στο οποίο κάθε τμήμα του δικτύου θεωρείται ως μια ανεξάρτητη διεργασία. Οι διεργασίες αυτές αλληλεπιδρούν μεταξύ τους με κατάλληλα σήματα διακοπών (interrupt signals) και το τελικό αποτέλεσμα είναι η προσομοίωση της συνολικής συμπεριφοράς του δικτύου. Η γλώσσα προγραμματισμού με την οποία υλοποιούνται οι διεργασίες είναι η Proto C, μια παραλλαγή της γλώσσας C, η οποία μπορεί να χρησιμοποιεί και έτοιμες συναρτήσεις από τον πυρήνα (Kernel) του OPNET.

Το OPNET παρέχει ακόμα εκτενή βιβλιοθήκη προτύπων και πρωτοκόλλων, μέσα στα οποία είναι και τα δίκτυα MANET, με διαθέσιμα πρωτόκολλα δρομολόγησης τα AODV, DSR, OLSR, TORA, και GRP (Geographic Routing Protocol). Οι κόμβοι των δικτύων MANET, όμως, δεν έχουν ενσωματωμένη διεργασία για την προσομοίωση της μπαταρίας των κόμβων.

Για το λόγο αυτό η διπλωματική υλοποιήθηκε σε δύο στάδια : Αρχικά, δημιουργήθηκε το μοντέλο της μπαταρίας και ενσωματώθηκε στον κόμβο του δικτύου MANET. Στη συνέχεια, υλοποιήθηκε το νέο πρωτόκολλο EDSR με βάση το αρχικό μοντέλο DSR, και ενσωματώθηκε στον κόμβο, ώστε να μπορούν να εκτελεστούν προσομοιώσεις και με τις δύο εκδοχές του DSR και να αξιολογηθούν τα αποτελέσματα της ενεργειακής επέκτασης.

4.2 Μοντέλο της Μπαταρίας

4.2.1 Περιγραφή

Το μοντέλο της μπαταρίας είναι ένα απλό, κατανοητό και αξιόπιστο μοντέλο για την προσομοίωσή της. Ουσιαστικά, πρόκειται για το βασικό μοντέλο της παραγράφου 2.4.1, θα γίνει όμως μια σύντομη περιγραφή για λόγους πληρότητας.

Το μοντέλο αυτό υποστηρίζει τρεις καταστάσεις κατανάλωσης ενέργειας, την μετάδοση (transmit), τη λήψη (receive) και την αδράνεια (idle). Η κατάσταση ύπνου

(sleep) δύσκολα χρησιμοποιείται σε δίκτυα MANET λόγω έλλειψης κεντρικού συντονισμού, γι' αυτό και δε συμπεριλήφθηκε.

Κάθε φορά που μεταδίδεται ή λαμβάνεται ένα πακέτο, αφαιρείται η αντίστοιχη ενέργεια από την υπολειπόμενη ενέργεια του κόμβου. Οι τύποι που υπολογίζουν την ενέργεια (σε Joule) είναι για μετάδοση και λήψη αντίστοιχα :

$$Energy_{T_x} = \frac{I_{T_x} \cdot V \cdot PacketSize}{R} \tag{4.1}$$

$$Energy_{R_x} = \frac{I_{R_x} \cdot V \cdot PacketSize}{R} \tag{4.2}$$

όπου I είναι η ένταση του ρεύματος σε Ampere, V η τάση τροφοδοσίας σε Volt, *PacketSize* το μέγεθος του πακέτου σε bits, και R ο ρυθμός μετάδοσης σε bps. Ουσιαστικά πρόκειται για το γινόμενο της ισχύος ($I \cdot V$) με τη χρονική διάρκεια του πακέτου (*bits/bps*).

Η κάρτα διεπαφής δικτύου λειτουργεί με «αδιάκριτο» τρόπο (promiscuous mode). Αυτό σημαίνει ότι οι κόμβοι ακούνε και λαμβάνουν όλα τα πακέτα, είτε προορίζονται για αυτούς είτε όχι. Η ιδιότητα αυτή οδηγεί σε πολύ μεγάλη κατανάλωση ενέργειας λόγω λήψης πακέτων που τελικά θα απορριφθούν, όμως αυτή είναι η υλοποίηση του OPNET για το στρώμα ζεύξης του 802.11.

Η κατανάλωση ενέργειας λόγω αδράνειας είναι προαιρετική στο μοντέλο,οπότε ο χρήστης αν θέλει μπορεί να την ενεργοποιήσει. Υπολογίζεται από τον τύπο :

$$Energy_{idle} = I_{idle} \cdot V \cdot t_{idle} \tag{4.3}$$

όπου I είναι η ένταση του ρεύματος σε Ampere, V η τάση τροφοδοσίας σε Volt και t_{idle} η διάρκεια αδράνειας. Όταν είναι ενεργοποιημένη η κατάσταση αδράνειας, τότε ανά τακτά χρονικά διαστήματα που απέχουν μεταξύ τους χρόνο *idle_interval* αφαιρείται η ενέργεια αδράνειας που αντιστοιχεί στο χρονικό αυτό διάστημα. Η υλοποίηση αυτή συμβάλλει στο να είναι σχετικά ομαλή η γραφική παράσταση της ενέργειας του κόμβου, χωρίς παράλληλα να έχει μεγάλες υπολογιστικές απαιτήσεις λόγω συνεχών διακοπών (interrupts). Όταν η ενέργεια του κόμβου έχει σχεδόν μηδενιστεί, το *idle_interval* μειώνεται στο 1sec, ώστε να είναι ακριβέστερη η αναπαράσταση της ενέργειας κοντά στο μηδέν.

Αν ενεργοποιηθεί η κατάσταση αδράνειας, τότε μεταβάλλεται και ο τρόπος υπολογισμού της καταναλισκόμενης ενέργειας. Με βάση τη συγκεκριμένη υλοποίηση, ο κόμβος θεωρεί ότι βρίσκεται πάντα σε αδράνεια, ακόμα και αν μεταδίδει ή λαμβάνει πακέτα. Εκμεταλλευόμενοι τη γραμμική σχέση ρεύματος και ενέργειας, μπορούμε με βάση την αρχή της επαλληλίας να προσθέσουμε την κατανάλωση ενέργειας λόγω μετάδοσης/λήψης και λόγω αδράνειας. Έτσι, διατηρείται η πραγματική τιμή του ρεύματος αδράνειας I_{idle}, ενώ από τα ρεύματα

μετάδοσης και λήψης αφαιρείται το I_{idle} έτσι ώστε η αντίστοιχη ενέργεια να υπολογίζεται τώρα με βάση τα ρεύματα:

$$I'_{R_x} = I_{R_x} - I_{idle} \tag{4.4}$$

$$I'_{T_x} = I_{T_x} - I_{idle} \tag{4.5}$$

Σημειώνεται ότι αν ο χρήστης ενεργοποιήσει την κατάσταση αδράνειας, το μοντέλο προσαρμόζει τις τιμές των ρευμάτων χωρίς την παρέμβαση του χρήστη.

Τέλος, αν η ενέργεια κάποιου κόμβου μηδενιστεί, τότε ο κόμβος αυτός «πεθαίνει», οπότε δε συμμετέχει πλέον σε καμία λειτουργία του δικτύου, δηλαδή δε λαμβάνει ούτε μεταδίδει πακέτα.

4.2.2 Υλοποίηση

Το μοντέλο της μπαταρίας έχει ενσωματωθεί στον κόμβο *akis_manet_station_node*. Για να γίνει αυτό, χρειάστηκε να γίνουν κάποιες αλλαγές στις ήδη υπάρχουσες διεργασίες (οι οποίες διατηρούν το όνομά τους, με την προσθήκη του "*akis*" στην αρχή) αλλά και να δημιουργηθούν κάποιες νέες. Το μοντέλο βασίστηκε στην υλοποίηση ανάλογου μοντέλου για το πρωτόκολλο IEEE 802.15.4 που περιγράφεται στην εργασία [24].

Αρχικά, τροποποιήθηκε η διεργασία *akis_wlan_dispatch* στο στρώμα ζεύξης του κόμβου που γεννά την κατάλληλη διεργασία MAC, ώστε να γεννά την *akis_wlan_mac* που συμπεριλαμβάνει τη μπαταρία.

Η διεργασία *akis_wlan_mac* περιέχει δύο συναρτήσεις που καλούνται κάθε φορά που λαμβάνεται ή μεταδίδεται ένα πακέτο και στέλνουν διακοπές (remote interrupts) στη διεργασία της μπαταρίας *akis_wlan_battery_process* μέσω του μηχανισμού ICI (Interface Control Information). Το ICI *akis_wlan_battery_ici_format* παρέχει στη μπαταρία τις πληροφορίες για το ρυθμό μετάδοσης και το μέγεθος του πλαισίου στρώματος MAC, ώστε η διεργασία της μπαταρίας να υπολογίσει την ενέργεια που καταναλώθηκε λόγω του πακέτου.

Η διεργασία *akis_wlan_battery_process* δεν ανήκει στα δικτυακά στρώματα αλλά είναι ανεξάρτητη και επικοινωνεί μόνο με το στρώμα ζεύξης μέσω του ICI (επικοινωνεί και με τη διεργασία του EDSR, όπως θα δούμε στην επόμενη ενότητα). Έτσι, μόλις λάβει κάποιο remote interrupt από την *akis_wlan_mac*, υπολογίζει την ενέργεια που καταναλώθηκε και ενημερώνει κατάλληλα τις μεταβλητές και τα στατιστικά. Αν είναι ενεργοποιημένη η κατάσταση αδράνειας, τότε όταν λαμβάνει self interrupt κάθε *idle_interval*, ενημερώνει κατάλληλα και πάλι μεταβλητές και στατιστικά.

Όσον αφορά τη διεπαφή με το χρήστη, ο κόμβος έχει τις ακόλουθες *ιδιότητες* (attributes) κάτω από την ετικέτα *Battery* :

- Power Supply: Η τάση τροφοδοσίας της κάρτας διεπαφής δικτύου (Volt).

- Initial Energy: Η αρχική ενέργεια του κόμβου (Joule).
- Current Rx: Η ένταση του ρεύματος λήψης (mA).
- Current Tx: Η ένταση του ρεύματος μετάδοσης (mA).
- Current Idle: Η ένταση του ρεύματος αδράνειας (mA).
- Idle Mode: (Απ)ενεργοποίηση αδρανούς κατάστασης (On/Off).
- Idle Interval: Χρονικό διάστημα μεταξύ ανανεώσεων της ενέργειας που καταναλώνεται σε κατάσταση αδράνειας (sec).
- Battery Log: (Απ)ενεργοποίηση αρχείου μπαταρίας (On/Off). Το αρχείο αυτό περιέχει για κάθε μετάδοση ή λήψη πακέτου από οποιονδήποτε κόμβο το μέγεθος του πακέτου σε bits και την ενέργεια που καταναλώθηκε σε Joule, στη μορφή :

  ```
  Rx - mobile_node_11 - pksize = 2592.000000 Consumed energy due to Reception = 0.000212
  ```

Τα *στατιστικά* που καταγράφονται και πάλι κάτω από την ετικέτα *Battery* είναι :

- Consumed Energy (Local): Η ενέργεια που έχει καταναλωθεί από κάποιον κόμβο (Joule).
- Remaining Energy (Local): Η ενέργεια που υπολείπεται σε κάποιον κόμβο. Ισούται με τη διαφορά της καταναλωθείσας από την αρχική ενέργεια (Joule).
- Consumed Energy (Global): Η συνολική ενέργεια που έχει καταναλωθεί από όλους τους κόμβους του δικτύου (Joule).
- Remaining Energy (Global): Η συνολική ενέργεια που υπολείπεται στο δίκτυο. Προϋπόθεση αποτελεί όλοι οι κόμβοι του δικτύου να έχουν την ίδια αρχική ενέργεια (Joule).
- Failed Nodes (Global): Ο αριθμός των κόμβων που έχουν πλήρως εξαντληθεί ενεργειακά.
- Average Remaining Energy (Global): Η μέση υπολειπόμενη ενέργεια των κόμβων. Ισούται με την συνολική υπολειπόμενη ενέργεια του δικτύου προς το πλήθος των κόμβων. Προϋπόθεση αποτελεί όλοι οι κόμβοι του δικτύου να έχουν την ίδια αρχική ενέργεια (Joule).

Περισσότερες λεπτομέρειες για την υλοποίηση καθώς και ο πηγαίος κώδικας είναι διαθέσιμα στο Παράρτημα Α.

4.2.3 Προσομοιώσεις Σεναρίων Μπαταρίας

Για τη δοκιμή και αξιολόγηση του μοντέλου εκτελέστηκε πλήθος προσομοιώσεων. Τα σενάρια προσομοιώθηκαν με δύο διαφορετικά πρωτόκολλα δρομολόγησης, τα AODV και DSR, ώστε να συγκριθούν έπειτα ως προς την ενεργειακή τους απόδοση.

Αρχικά, κατασκευάσαμε ένα βασικό σενάριο με τα χαρακτηριστικά του Πίνακα 4.1.

Σταθερά	Τιμή Σταθεράς
Χρόνος προσομοίωσης	15 min (900 sec)
Διαστάσεις χώρου	500m x 500m
Ραδιοσυχνότητα	2.4 GHz
Ρυθμός Μετάδοσης	11 Mbps
Ισχύς μετάδοσης	10 mW (10 dBm)
Ευαισθησία δέκτη	-76 dBm
Αρχική ενέργεια μπαταρίας	50 J
Τάση παροχής μπαταρίας	5 V
Ένταση ρεύματος μετάδοσης	280 mA
Ένταση ρεύματος λήψης	180 mA
Μοντέλο κινητικότητας	Random Waypoint Mobility Model
Κατανομή Ταχύτητας	Uniform(0,15) m/s
Χρόνος Πάυσης	0 sec
Πλήθος κόμβων	20
Πλήθος πηγών κίνησης	20
Μέγεθος Πακέτου	1024 bits
Χρόνος μεταξύ διαδοχικών αφίξεων	1 sec
Κατάσταση αδράνειας	Απενεργοποιημένη

Πίνακας 4.1 : Χαρακτηριστικά προσομοιώσεων μπαταρίας

Η τοπολογία του δικτύου είναι τυχαία. Επίσης, όλοι οι κόμβοι παράγουν κίνηση προς τυχαίο προορισμό με ρυθμό 1Kbps, που προκύπτει από το σταθερό μέγεθος πακέτων στα 1024 bits (128 Bytes) και το σταθερό χρόνο μεταξύ διαδοχικών αφίξεων πακέτων που είναι 1 sec. Οι τιμές που αφορούν τη μπαταρία ορίστηκαν με βάση τον Πίνακα 2.4.

Η κίνηση των κόμβων στο χώρο είναι τυχαία καθώς ακολουθεί το μοντέλο Random Waypoint, όπως ορίζεται στο μοντέλο Mobility Config. Αυτό σημαίνει ότι κάθε κόμβος επιλέγει έναν τυχαίο προορισμό εντός της περιοχής κίνησης, που ορίζεται να είναι όλος ο χώρος του δικτύου, και κινείται προς αυτόν με τυχαία ταχύτητα που ακολουθεί την ομοιόμορφη κατανομή ακεραίων στο διάστημα [0,15] m/s. Μόλις φθάσει στον προορισμό σταματά για χρόνο πάυσης, που ορίζεται στα σενάρια αυτά να είναι μηδέν, και στη συνέχεια επαναλαμβάνει τη διαδικασία επιλέγοντας διαφορετικό τυχαίο προορισμό. Έτσι, οι τροχιές κίνησης των κόμβων είναι διαφορετικές για κάθε εκτέλεση της προσομοίωσης.

Η κατάσταση αδράνειας είναι απενεργοποιημένη, καθώς ο χρόνος προσομοίωσης είναι κοινός για όλα τα σενάρια, οπότε θεωρούμε προσεγγιστικά ότι και η κατανάλωση ενέργειας στην κατάσταση αυτή είναι κοινή για όλα τα σενάρια. Ακόμα, ο ρυθμός μετάδοσης για τα πλαίσια επιβεβαίωσης είναι από την προεπιλογή σταθερός στο 1Mbps.

Η ισχύς μετάδοσης και η ευαισθησία του δέκτη ρυθμίστηκαν με βάση την εξίσωση του Friis [14],

$$P_R(d) = P_T G_T G_R \left(\frac{\lambda}{4\pi d^2} \right) \tag{4.6}$$

όπου P_R και P_T οι ισχείς λήψης και μετάδοσης αντίστοιχα, G_R και G_T τα κέρδη των κεραιών λήψης και μετάδοσης αντίστοιχα, d η απόσταση μεταξύ πομπού και δέκτη και λ το μήκος κύματος.

Για ισοτροπικές κεραίες ($G_T = G_R = 1$) έχουμε το μοντέλο απωλειών διάδοσης ελευθέρου χώρου. Έτσι, από την (4.6) για $P_R = P_{R,\min}$, δηλαδή την ευαισθησία του δέκτη, βρίσκουμε την ακτίνα διάδοσης $R_{\max}$

$$R_{\max} = \frac{\lambda}{4\pi}\sqrt{\frac{P_T}{P_{R,\min}}} \tag{4.7}$$

Έτσι, επιλέχθηκαν η ισχύς μετάδοσης και η ευαισθησία δέκτη ώστε να μην υπάρχει μεγάλη αλληλοεπικάλυψη των ακτινών διάδοσης, οπότε προκύπτει ακτίνα διάδοσης :

$$R_{\max} = 198m \tag{4.8}$$

Το βασικό σενάριο εκτελέστηκε για τα πρωτόκολλα AODV και DSR. Έπειτα, μελετήσαμε το ίδιο σενάριο μεταβάλλοντας επιλεγμένες παραμέτρους, δηλαδή το πλήθος των κόμβων (10, 20, 30) και τη μέγιστη ταχύτητα των κόμβων (1, 5, 15 m/s). Επισημαίνεται ότι σε κάθε σενάριο, το σύνολο των κόμβων είναι και πηγές κίνησης. Τελικά, το βασικό σενάριο μεταβλήθηκε κατάλληλα ώστε να γίνει προσομοίωση του δικτύου με ενεργοποιημένη την κατάσταση αδράνειας για τη μπαταρία.

Τα *κριτήριο (metric)* με βάση το οποίο θα αξιολογηθούν οι ενεργειακές επιδόσεις των πρωτοκόλλων είναι η μέση υπολειπόμενη ενέργεια των κόμβων. Σε περίπτωση εξάντλησης της ενέργειας των κόμβων, θα χρησιμοποιηθεί η διάρκεια ζωής του δικτύου, δηλαδή το χρονικό διάστημα μέχρι τον πρώτο ενεργειακό «θάνατο» κόμβου, καθώς και το πλήθος των νεκρών κόμβων.

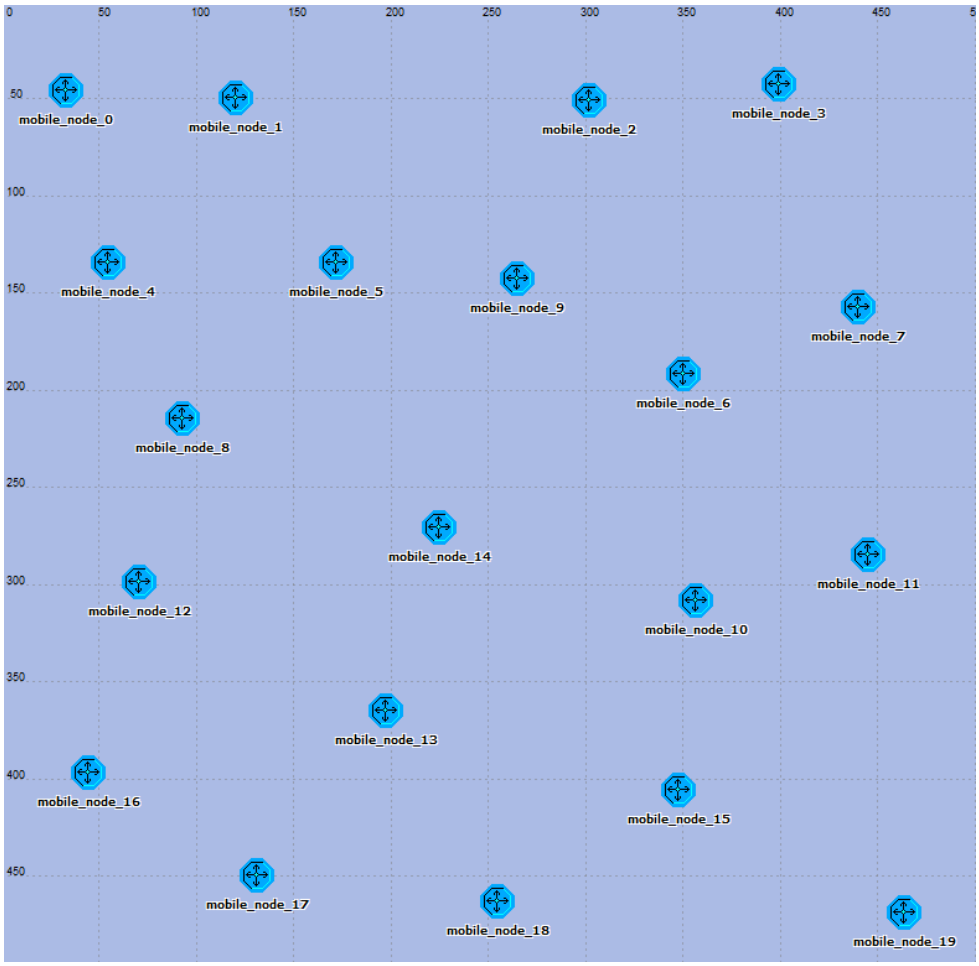
4.2.4 Αποτελέσματα Σεναρίων Μπαταρίας

Αρχικά, παρουσιάζονται αναλυτικά οι γραφικές παραστάσεις για τα αποτελέσματα του βασικού σεναρίου, ώστε να γίνουν αντιληπτές και οι δυνατότητες του μοντέλου μπαταρίας που υλοποιήθηκε. Έπειτα, τα επόμενα σενάρια μελετώνται πιο συνοπτικά, με έμφαση στη σύγκριση των πρωτοκόλλων.

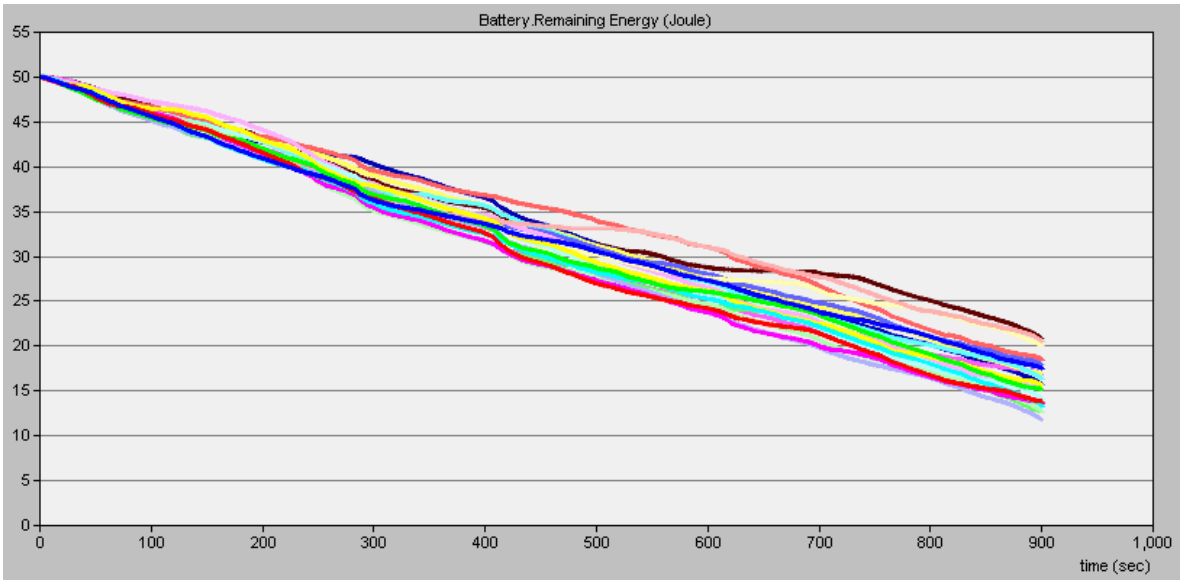
4.2.4.1 Βασικό Σενάριο Μπαταρίας

Το βασικό σενάριο, όπως περιγράφεται στον Πίνακα 4.1 και με την τυχαία τοπολογία του Σχήματος 4.1, προσομοιώθηκε με χρήση των πρωτοκόλλων AODV και DSR και τα αποτελέσματα φαίνονται στα Σχήματα 4.2 – 4.4 που παράγει το OPNET. Στο Σχήμα 4.2 παρουσιάζεται η υπολειπόμενη ενέργεια κάθε κόμβου του δικτύου συναρτήσει του χρόνου για το πρωτόκολλο DSR. Όμοια γραφική παράσταση υπάρχει και για την καταναλωθείσα ενέργεια ανά κόμβο, η οποία προκύπτει από τα ίδια δεδομένα, απλά ξεκινά από το μηδέν και είναι αύξουσα. Βλέπουμε ότι η αξιολόγηση της επίδοσης από αυτή τη γραφική παράσταση είναι ιδιαίτερα δύσκολη λόγω του

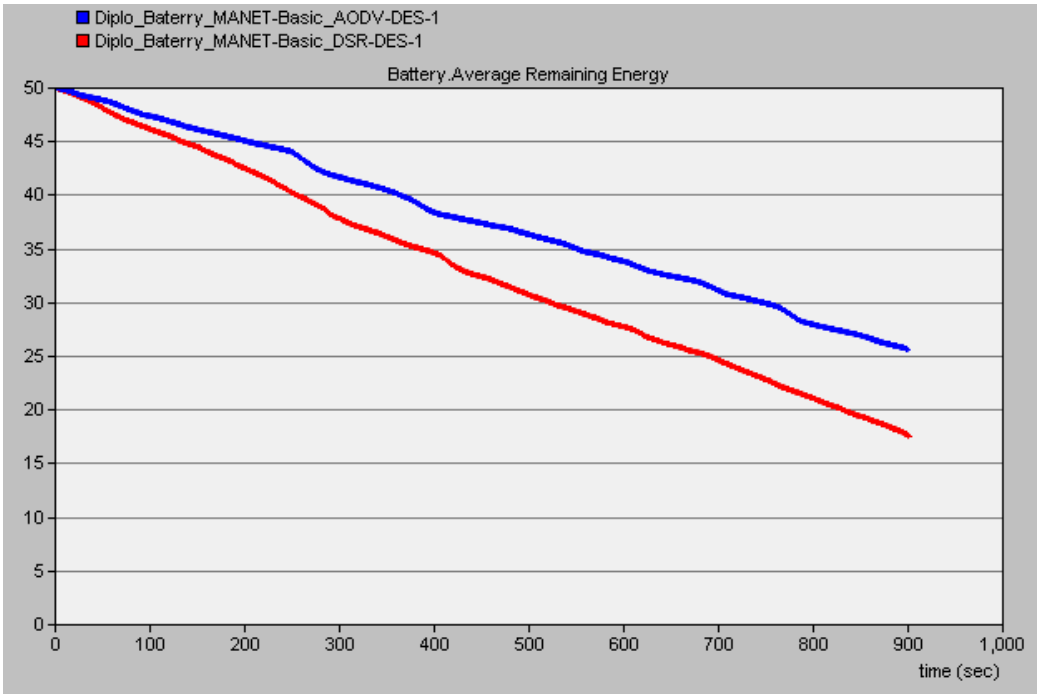
πλήθους των κόμβων. Γι' αυτό και χρησιμοποιούμε την μέση υπολειπόμενη ενέργεια ανά κόμβο (Σχήμα 4.3), που προκύπτει από τη συνολική υπολειπόμενη ενέργεια διαιρεμένη με το πλήθος των κόμβων, και μας επιτρέπει να συγκρίνουμε άμεσα την ενεργειακή κατανάλωση διαφορετικών πρωτοκόλλων. Έπειτα, στο Σχήμα 4.4 φαίνεται η συνολική καταναλωθείσα ενέργεια από το δίκτυο, ενώ ανάλογη γραφική παράσταση υπάρχει και για την συνολική υπολειπόμενη ενέργεια, που μάλιστα είναι όμοια με τη μέση υπολειπόμενη ενέργεια. Τέλος, υπάρχει και το στατιστικό των νεκρών κόμβων, στη συγκεκριμένη περίπτωση όμως δεν έχουμε ενεργειακούς θανάτους, οπότε η εν λόγω γραφική παράσταση θα παρουσιαστεί σε επόμενο σενάριο.



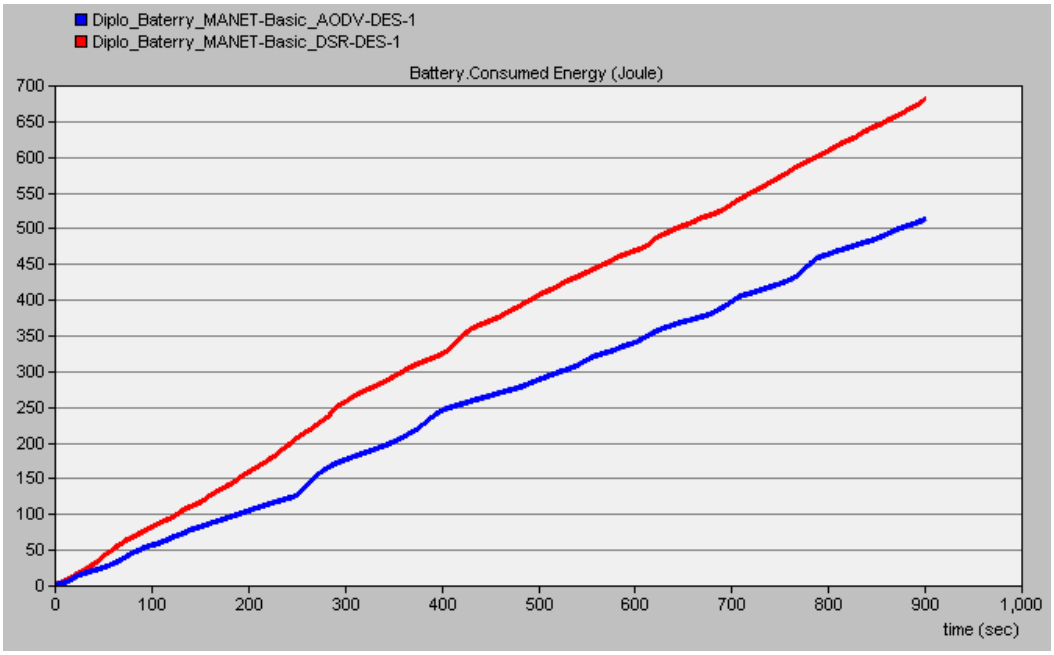
Σχήμα 4.1 : Τοπολογία του Βασικού Σεναρίου Μπαταρίας



Σχήμα 4.2 : Υπολειπόμενη ενέργεια κόμβων συναρτήσει του χρόνου στο Βασικό Σενάριο Μπαταρίας για το πρωτόκολλο DSR



Σχήμα 4.3 : Μέση υπολειπόμενη ενέργεια κόμβων συναρτήσει του χρόνου στο Βασικό Σενάριο Μπαταρίας για τα πρωτόκολλα AODV και DSR



Σχήμα 4.4 : Συνολική καταναλωθείσα ενέργεια συναρτήσει του χρόνου στο Βασικό Σενάριο Μπαταρίας για τα πρωτόκολλα AODV και DSR

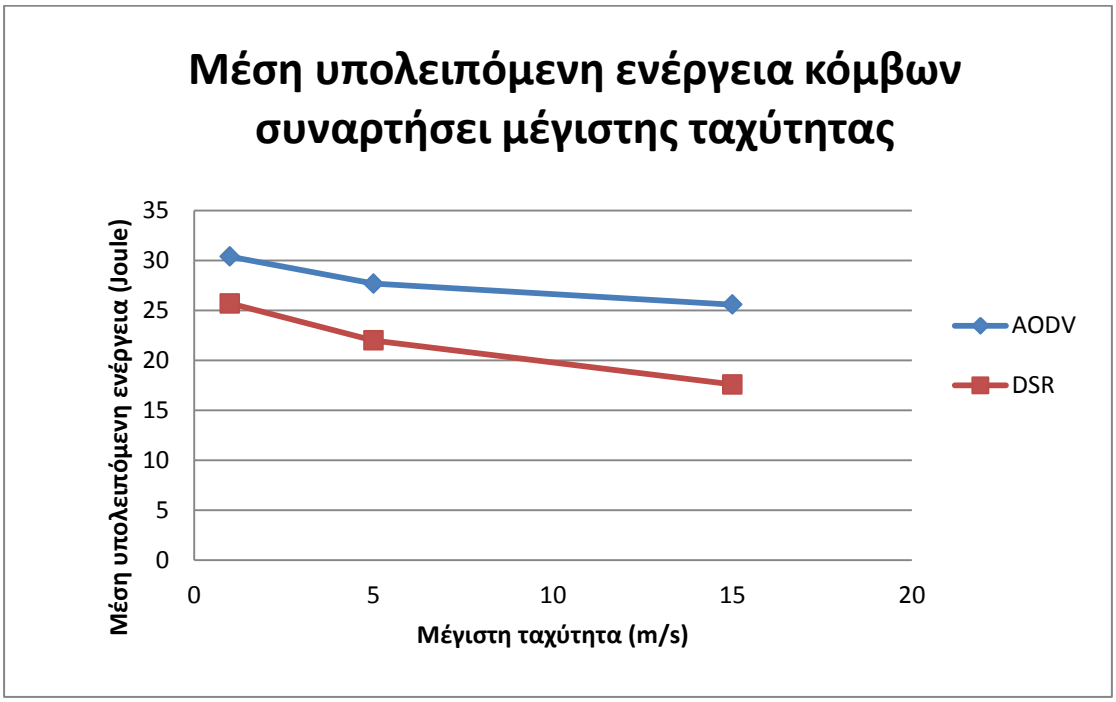
Από τις γραφικές παραστάσεις προκύπτει ότι το πρωτόκολλο DSR καταναλώνει περισσότερη ενέργεια από το AODV, στη συγκεκριμένη περίπτωση 32% περισσότερη. Μια πιθανή εξήγηση είναι η δρομολόγηση πηγαίας διαδρομής του DSR, σύμφωνα με την οποία όλα τα πακέτα δεδομένων εμπεριέχουν στην επικεφαλίδα τους και τη διαδρομή που θα ακολουθήσουν προς τον προορισμό, αυξάνοντας έτσι το μέγεθος του πακέτου, άρα και την κατανάλωση ενέργειας σε μετάδοση και λήψη δεδομένων. Για μια γενικότερη εικόνα, θα συγκρίνουμε τα πρωτόκολλα μεταβάλλοντας κάποιες παραμέτρους του σεναρίου στην επόμενη ενότητα.

4.2.4.2 Γενικά Σενάρια Μπαταρίας

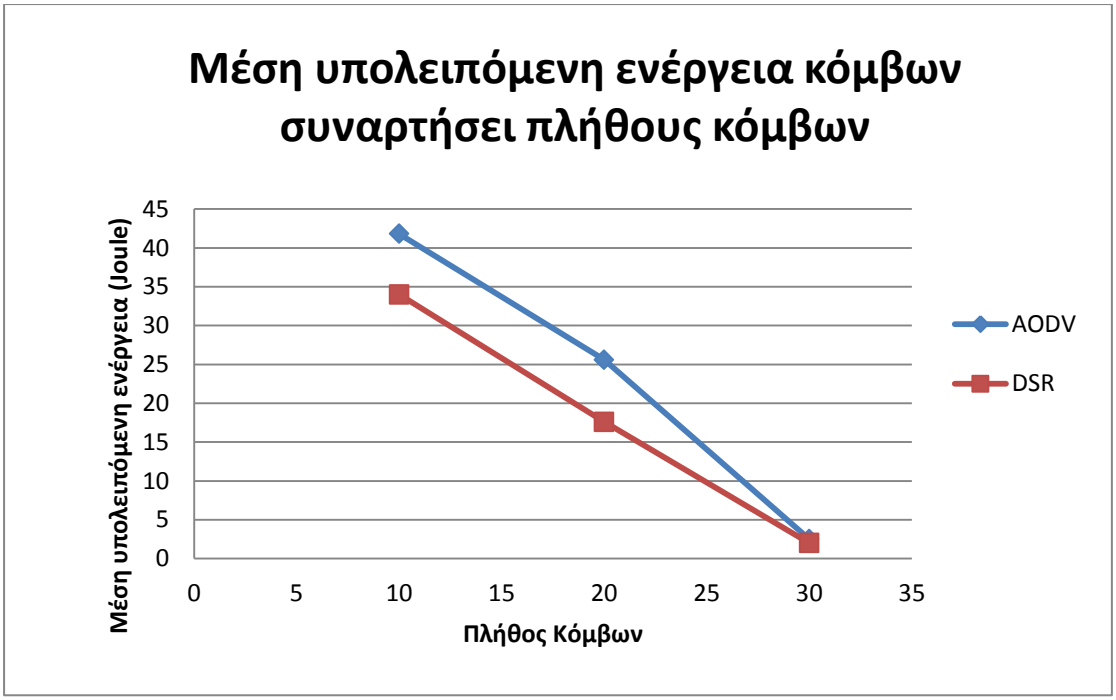
Στα γενικά σενάρια χρησιμοποιούμε το βασικό σενάριο και μεταβάλλουμε αρχικά το πλήθος των κόμβων και έπειτα τη μέγιστη ταχύτητα των κόμβων. Στην περίπτωση αυτή δε θα συγκρίνουμε τις γραφικές παραστάσεις του OPNET συναρτήσει του χρόνου, αλλά τις τελικές τιμές – τη χρονική στιγμή 900sec – των στατιστικών που μας ενδιαφέρουν. Έτσι, θα σχηματίσουμε τις γραφικές παραστάσεις συναρτήσει των μεταβαλλόμενων παραμέτρων που φαίνονται στα Σχήματα 4.5 – 4.6 με βάση τα αποτελέσματα του Πίνακα 4.2. Ακόμα, στην περίπτωση των 30 κόμβων έχουμε πλήρη εξάντληση της ενέργειας κάποιων κόμβων, κάτι που φαίνεται στο Σχήμα 4.7.

Πλήθος κόμβων	Πρωτόκολλο	Μέση υπολειπόμενη ενέργεια (J)	Διάρκεια Ζωής Δικτύου (sec)
10	AODV	41,8	-
	DSR	34	-
20	AODV	25,6	-
	DSR	17,6	-
30	AODV	2,5	511
	DSR	2	293
Μέγιστη ταχύτητα (m/s)			
1	AODV	30,4	-
	DSR	25,7	-
5	AODV	27,7	-
	DSR	22	-
15	AODV	25,6	-
	DSR	17,6	-

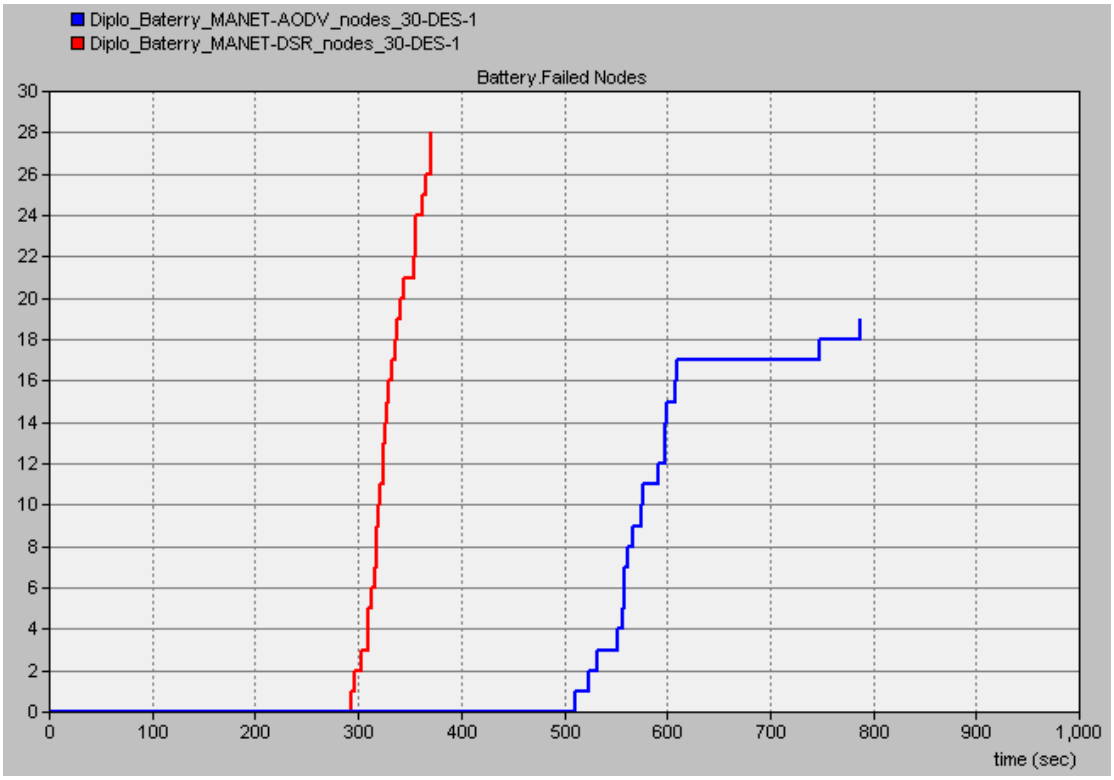
Πίνακας 4.2 : Αποτελέσματα Γενικών Σεναρίων Μπαταρίας για τα πρωτόκολλα AODV και DSR



Σχήμα 4.5 : Μέση υπολειπόμενη ενέργεια κόμβων συναρτήσει μέγιστης ταχύτητας στα Γενικά Σενάρια Μπαταρίας για τα πρωτόκολλα AODV και DSR



Σχήμα 4.6 : Μέση υπολειπόμενη ενέργεια κόμβων συναρτήσει πλήθους κόμβων στα Γενικά Σενάρια Μπαταρίας για τα πρωτόκολλα AODV και DSR



Σχήμα 4.7 : Πλήθος νεκρών κόμβων συναρτήσει του χρόνου στο Γενικό Σενάριο Μπαταρίας των 30 κόμβων για τα πρωτόκολλα AODV και DSR

Τα αποτελέσματα του βασικού σεναρίου επιβεβαιώνονται από τα γενικά σενάρια. Το DSR έχει γενικά μεγαλύτερη κατανάλωση ενέργειας από αυτήν του AODV, με

διαφορά που κυμαίνεται από 23% στην περίπτωση ταχύτητας 1m/s έως 95% στην περίπτωση 10 κόμβων. Επισημαίνεται ότι αναφερόμαστε στη μέση κατανάλωση ενέργειας, ενώ οι γραφικές παραστάσεις παρουσιάζουν τη μέση υπολειπόμενη ενέργεια.

Παρατηρούμε στο Σχήμα 4.5 ότι η κατανάλωση ενέργειας αυξάνεται με την αύξηση της μέγιστης ταχύτητας των κόμβων. Σημειώνεται ότι η κατανομή της ταχύτητας των κόμβων είναι ομοιόμορφη στο διάστημα (0, u_{max}). Η ταχύτητα 1m/s προσομοιώνει ένα δίκτυο ανθρώπων που περπατάνε, η ταχύτητα 5m/s ένα δίκτυο ποδηλατών, και η ταχύτητα 15m/s ένα δίκτυο αυτοκινήτων στην πόλη [13]. Ακόμα, το DSR είναι πιο επιρρεπές στις υψηλές ταχύτητες από το AODV, όπως φαίνεται από τη μεγαλύτερη κλίση της γραφικής παράστασης του DSR καθώς αυξάνονται οι ταχύτητες.

Ακόμα, όσον αφορά το Σχήμα 4.6, παρατηρούμε ότι με την αύξηση του πλήθους των κόμβων αυξάνεται και η ενεργειακή κατανάλωση. Αφού το μέγεθος του δικτύου παραμένει σταθερό, η πυκνότητα των κόμβων στο δίκτυο αυξάνεται δραματικά με αποτέλεσμα πολλοί κόμβοι να «κρυφακούνε» πακέτα που δεν προορίζονται για αυτούς. Το φαινόμενο αυτό οδηγεί σε μεγάλη κατανάλωση λόγω λήψης αδιάφορων πακέτων. Μάλιστα, στους 30 κόμβους η μέση υπολειπόμενη ενέργεια έχει σχεδόν μηδενιστεί, κάτι που σημαίνει ότι πολλοί κόμβοι έχουν «πεθάνει», οπότε το κριτήριο αυτό παύει να είναι αξιόπιστο. Γι' αυτό καταφεύγουμε στο κριτήριο του πλήθους νεκρών κόμβων, όπως παρουσιάζεται στο Σχήμα 4.7, συναρτήσει του χρόνου. Παρατηρούμε ότι και στην περίπτωση αυτή το AODV πετυχαίνει μεγαλύτερη διάρκεια ζωής από το DSR κατά 74%, ενώ τελικά οι νεκροί κόμβοι είναι λιγότεροι, κάτι που όμως έχει μικρή σημασία αφού η αποστολή και λήψη πακέτων έχει ήδη διακοπεί.

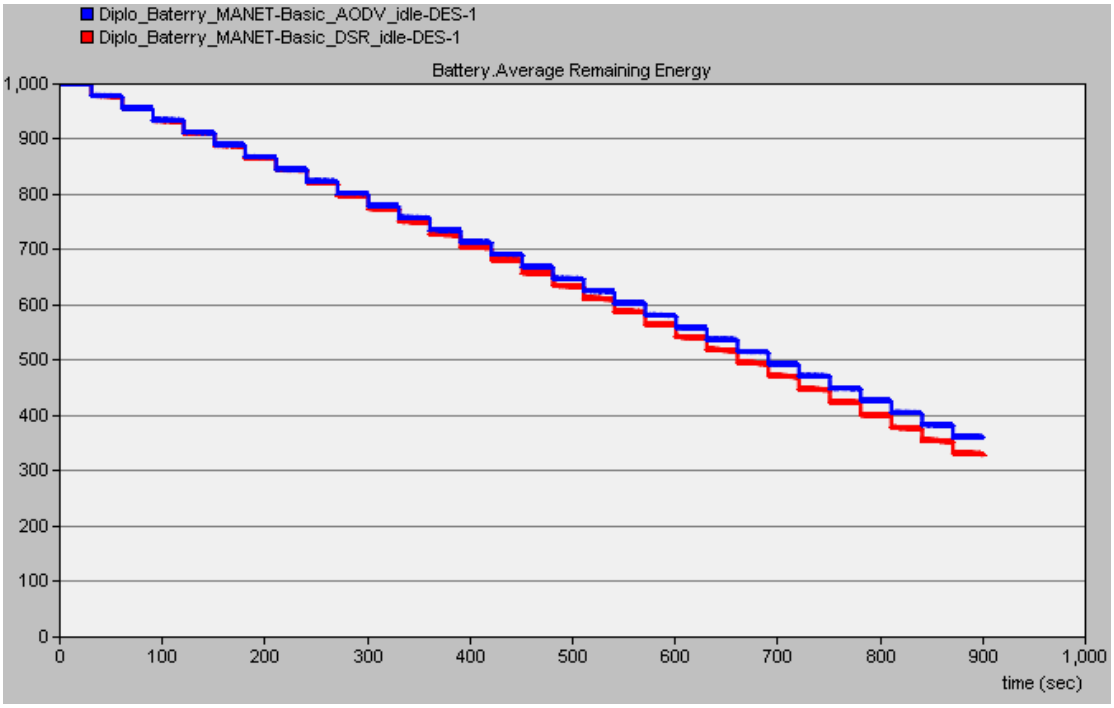
Γενικά, η ενεργειακή απόδοση του DSR είναι χειρότερη από αυτή του AODV λόγω του μεγαλύτερου μεγέθους πακέτων που προκύπτει από τη δρομολόγηση πηγαίας διαδρομής σε συνδυασμό με το μεγάλο πλήθος πακέτων που «κρυφακούγονται» λόγω του promiscuous mode της ασύρματης κάρτας διεπαφής δικτύου. Τα αποτελέσματα αυτά επιβεβαιώνονται στην εργασία [19], όπου ωστόσο προκύπτει ότι το DSR είναι πιο αποδοτικό από το AODV ως προς τη χρησιμοποίηση του εύρους ζώνης, δηλαδή του αριθμού πακέτων που μεταδίδονται και λαμβάνονται. Σε κάθε περίπτωση, το κόστος λειτουργίας σε promiscuous mode είναι υψηλό, ειδικά σε δίκτυα υψηλής πυκνότητας, όπου κάθε μετάδοση πακέτου και η ευρυεκπομπή σε μεγαλύτερη κλίμακα κοστίζουν επιπλέον ενέργεια. Έτσι, κρίνεται σκόπιμη μια μελέτη των πρωτοκόλλων αυτών πάνω από στρώμα ζεύξης με απενεργοποιημένο το promiscuous mode, όπου οι κόμβοι αντί να λαμβάνουν όλα τα πακέτα, απορρίπτουν αυτά που δεν προορίζονται για αυτούς στο στρώμα MAC με βάση την επικεφαλίδα του πλαισίου.

4.2.4.3 Βασικό Σενάριο Μπαταρίας με αδράνεια

Επιστρέφουμε στο βασικό σενάριο, στο οποίο όμως είναι πλέον ενεργοποιημένη η κατάσταση της αδράνειας. Αυτό σημαίνει ότι ο κόμβος ακούει συνεχώς το κανάλι καταναλώνοντας ενέργεια. Το ρεύμα αδράνειας είναι 150mA και η καταναλισκόμενη

ενέργεια υπολογίζεται κάθε 30 sec. Προσεγγιστικά θεωρούμε ότι η κάρτα διεπαφής δικτύου βρίσκεται καθ' όλη τη διάρκεια σε κατάσταση αδράνειας και στην κατανάλωση αυτή προσθέτουμε την επιπλέον ενέργεια που προκύπτει από τη μετάδοση και λήψη πακέτων. Μεταβάλλουμε την κίνηση που παράγεται σε πακέτα των 8192 bits κάθε 0.5 sec, ώστε να γίνει εμφανής η ενέργεια λόγω μετάδοσης και λήψης πακέτων. Παρ' όλη την αύξηση, προκύπτει ότι ο χρόνος μετάδοσης και λήψης πακέτων αντιστοιχεί σε λιγότερο από 1% του συνολικού χρόνου, αν κάθε κόμβος έχει το πολύ μέχρι 6 γείτονες στην ακτίνα λήψης του. Ακόμα, η αρχική ενέργεια των κόμβων γίνεται 1000 J, ώστε να μην εξαντληθεί πολύ γρήγορα.

Η μέση υπολειπόμενη ενέργεια παρουσιάζεται στο Σχήμα 4.8. Το «πριονωτό» σχήμα της οφείλεται στο γεγονός ότι η ενέργεια που καταναλώνεται λόγω αδράνειας δεν αφαιρείται ομοιόμορφα αλλά κάθε 30sec, όπως φαίνεται από τις κατακόρυφες πτώσεις της ενέργειας. Το διάστημα των 30 sec (idle interval) είναι μεταβλητό, επιλέχθηκε όμως έτσι ώστε η γραφική παράσταση να είναι κατανοητή και κοντά στην πραγματικότητα χωρίς να αυξάνονται υπερβολικά οι υπολογιστικές απαιτήσεις από συνεχείς διακοπές. Τα σχεδόν οριζόντια τμήματα της γραφικής αφορούν την κατανάλωση ενέργειας λόγω μετάδοσης και λήψης πακέτων.



Σχήμα 4.8 : Μέση υπολειπόμενη ενέργεια κόμβων συναρτήσει του χρόνου στο Βασικό Σενάριο Μπαταρίας με αδράνεια για τα πρωτόκολλα AODV και DSR

Παρατηρούμε ότι η κατανάλωση ενέργειας στην κατάσταση αδράνειας κυριαρχεί επί του συνόλου. Το AODV διατηρεί καλύτερη επίδοση από το DSR, όμως η διαφορά μεταξύ τους στην κατανάλωση ενέργειας έχει πλέον περιοριστεί μόλις στο 5% από το 32% του ίδιου σεναρίου χωρίς αδράνεια. Εξάλλου, τα ρεύματα αδράνειας και λήψης έχουν τιμές 150 mA και 180 mA αντίστοιχα, οπότε είναι λογικό το κόστος

αδράνειας να είναι τόσο μεγάλο, αφού επικρατεί στο συνολικό χρόνο λειτουργίας. Η προοπτική της ενσωμάτωσης της κατάστασης ύπνου στα δίκτυα MANET μπορεί να οδηγήσει σε σημαντική μείωση της κατανάλωσης ενέργειας, αφού το αντίστοιχο ρεύμα είναι της τάξεως των 10mA. Ωστόσο, η δυσκολία του εγχειρήματος έγκειται στην έλλειψη κεντρικού συντονισμού του δικτύου στη DCF λειτουργία, οπότε πρέπει να εφευρεθούν πολύπλοκοι μηχανισμοί με απαιτήσεις συγχρονισμού και σηματοδοσίας.

4.3 Μοντέλο του Πρωτοκόλλου Δρομολόγησης EDSR

4.3.1 Περιγραφή

Το πρωτόκολλο EDSR έχει ως στόχο την αύξηση του χρόνου ζωής ενός δικτύου MANET. Η αρχιτεκτονική του είναι όμοια με αυτήν του Min-Max Battery Cost Routing (MMBCR), αλλά στην ουσία πρόκειται για Max-Min αλγόριθμο. Ως κριτήριο δρομολόγησης χρησιμοποιείται η υπολειπόμενη ενέργεια των διαδρομών από την πηγή προς τον προορισμό. Έτσι, επεκτείνοντας τον μηχανισμό Ανακάλυψης Διαδρομής επιτυγχάνεται δρομολόγηση με ενεργειακά κριτήρια, ενώ ο μηχανισμός Συντήρησης Διαδρομής συμβάλλει στη σταθερότητα του πρωτοκόλλου. Ακόμα, ο μηχανισμός ανανέωσης της μνήμης διαδρομών (Route Cache Update) συμβάλλει ώστε οι πληροφορίες για το επίπεδο ενέργειας κάθε διαδρομής να ανταποκρίνονται στην πραγματικότητα.

Οι επικεφαλίδες των επιλογών στα πακέτα DSR επεκτείνονται ως εξής :

- Στην αίτηση διαδρομής (RREQ – Σχήμα 3.4) προστίθενται τα πεδία *RouteUpdate* και *MinEnergy*. Το πεδίο *RouteUpdate* είναι μια σημαία ενός bit που λαμβάνει τις τιμές μηδέν ή ένα αν το πακέτο αφορά ανανέωση της διαδρομής ή όχι. Το πεδίο *MinEnergy* είναι μια πραγματική μεταβλητή που βρίσκει την ελάχιστη υπολειπόμενη ενέργεια μιας διαδρομής, δηλαδή την ελάχιστη από τις ενέργειες των κόμβων που ανήκουν στη διαδρομή.
- Στην απάντηση διαδρομής (RREP – Σχήμα 3.5) προστίθενται επίσης τα πεδία *RouteUpdate* και *MinEnergy*, με την ίδια λειτουργικότητα.
- Στην επιβεβαίωση (ACK – Σχήμα 3.8) προστίθεται το πεδίο *MinEnergy*, που λαμβάνει μια σταθερή τιμή *MaxEnergy*, ορισμένη ως μέγιστη δυνατή ενέργεια, που πρέπει να είναι μεγαλύτερη από την τιμή της υπολειπόμενης ενέργειας κάθε κόμβου.

Ακόμα, η μνήμη διαδρομών (Route Cache) τροποποιείται κατάλληλα, ώστε να αποθηκεύονται οι πληροφορίες των νέων μηχανισμών. Προστίθενται οι σταθερές *MaxEnergy*, η μέγιστη δυνατή ενέργεια ενός κόμβου, και η *RouteActivityTime*, ένα χρονικό όριο για χαρακτηρισμό των διαδρομών ως ενεργές ή ανενεργές. Ακόμα, σε κάθε εγγραφή διαδρομής στη μνήμη διαδρομών προστίθεται το πεδίο *MinEnergy*, η ελάχιστη ενέργεια της διαδρομής. Από το DSR ήδη υποστηρίζεται η αποθήκευση πολλαπλών διαδρομών για κάθε προορισμό, αλλά στο EDSR βέλτιστη διαδρομή θεωρείται αυτή με τη μέγιστη *MinEnergy*, εξ ου και ο Max-Min αλγόριθμος. Αν

χρειαστεί να ελευθερωθεί χώρος, διαγράφεται η διαδρομή με την ελάχιστη *MinEnergy*, και σε περίπτωση ισότητας, η διαδρομή που χρησιμοποιήθηκε λιγότερο πρόσφατα (Least Recently Used – LRU). Ακόμα, υπάρχει ένα χρονικό όριο λήξης διαδρομής (*PathExpiryTime*) που αν η διαδρομή ξεπεράσει χωρίς να χρησιμοποιηθεί, διαγράφεται.

4.3.2 Βασικοί Μηχανισμοί

Στη συνέχεια, περιγράφεται αναλυτικά η λειτουργία των μηχανισμών του EDSR.

Ανακάλυψη Διαδρομής

Η Ανακάλυψη Διαδρομής λειτουργεί όπως και στο DSR, με τη διαφορά ότι εκτός του μονοπατιού προς τον προορισμό, προσδιορίζεται και η ελάχιστη ενέργεια του μονοπατιού αυτού. Για την ακρίβεια, ενδιαφέρει η ελάχιστη ενέργεια των ενδιάμεσων κόμβων, οπότε οι συγκρίσεις δεν αφορούν την πηγή και τον προορισμό. Το πακέτο RREQ εκπέμπεται αρχικά με *MinEnergy* τη σταθερά μέγιστης ενέργειας (*MaxEnergy*). Κάθε ενδιάμεσος κόμβος που λαμβάνει τη RREQ συγκρίνει τη δική του υπολειπόμενη ενέργεια με αυτήν του πακέτου και αν είναι μικρότερη, ανανεώνει το πεδίο του RREQ.

Όταν το πακέτο φτάσει στον προορισμό περιέχει ήδη την ελάχιστη ενέργεια της διαδρομής, οπότε ο προορισμός απαντά στην πηγή με RREP, όπου αναγράφεται και η *MinEnergy* της διαδρομής. Προϋπόθεση είναι ότι οι ζεύξεις είναι αμφίδρομες, οπότε η RREP ακολουθεί την ίδια διαδρομή με αντίθετη κατεύθυνση. Όταν η πηγή λάβει τη RREP, αποθηκεύει τη καταγραφείσα διαδρομή στη μνήμη διαδρομών του, που είναι ταξινομημένη σε φθίνουσα σειρά με βάση την ελάχιστη ενέργεια. Το πεδίο *RouteUpdate* των RREQ και RREP είναι προφανώς ψευδές.

Η αποθήκευση πληροφοριών δρομολόγησης που κρυφακούγονται χρησιμοποιείται περιορισμένα σε σχέση με το DSR, καθώς πάντα ελέγχεται αν η ελάχιστη ενέργεια του πακέτου είναι έγκυρη για τη διαδρομή που θέλει ο κόμβος να αποθηκεύσει στην μνήμη του. Επίσης, η απάντηση σε Αιτήσεις Διαδρομής με αποθηκευμένες διαδρομές περιορίζεται, καθώς μια αποθηκευμένη διαδρομή πρέπει να είναι ενεργή για να θεωρηθεί έγκυρη και να σταλεί ως απάντηση.

Συντήρηση Διαδρομής

Η κύρια αλλαγή που έγινε στη Συντήρηση Διαδρομής είναι η προσθήκη του πεδίου *MinEnergy* στα πακέτα ACK, η τιμή του οποίου είναι σταθερή και ίση πάντα με *MaxEnergy*. Έτσι, όταν πηγή και προορισμός είναι γειτονικοί κόμβοι, δηλαδή είναι δυνατή η απευθείας ζεύξη μεταξύ τους, επιλέγεται πάντα ως βέλτιστη διαδρομή η απευθείας ζεύξη, καθώς έχει *MinEnergy* μεγαλύτερη από κάθε άλλη διαδρομή περισσότερων βημάτων. Η συντήρηση αυτής της διαδρομής αποτρέπει την επιλογή άλλης διαδρομής, που θα είχε ως αποτέλεσμα την αναίτια σπατάλη ενέργειας ενδιάμεσων κόμβων.

Ακόμα, απενεργοποιήθηκε η αυτόματη συντόμευση διαδρομής, καθώς πλέον το κριτήριο δεν είναι η συντομότερη διαδρομή αλλά η μέγιστη ελάχιστη ενέργεια.

Ανανέωση Μνήμης Διαδρομών

Ο μηχανισμός της ανανέωσης είναι αναγκαία προσθήκη στο EDSR, καθώς οι αποθηκευμένες πληροφορίες που αφορούν την ενέργεια στην μνήμη κάθε κόμβου γίνονται ανακριβείς μετά από κάποιο χρονικό διάστημα, διότι η ενέργεια των κόμβων ελαττώνεται λόγω της αποστολής και λήψης πακέτων [12]. Αν δεν υπήρχε ο μηχανισμός αυτός, τότε μια διαδρομή που κάποτε ήταν η βέλτιστη, λόγω της συνεχούς χρησιμοποίησής της για την αποστολή πακέτων, θα κατέληγε σε πολύ χειρότερη κατάσταση από τις υπόλοιπες διαδρομές.

Ο μηχανισμός αυτός συνίσταται από την περιοδική ανανέωση της μνήμης διαδρομών των κόμβων μέσω πακέτων ανανέωσης διαδρομής που αποστέλλονται σε όλες τις ενεργές διαδρομές. Ορίζονται δύο νέοι τύποι πακέτων, που είναι παραλλαγές των RREP και RREQ.

- Η αίτηση ανανέωσης διαδρομής (Route Update Request – RU-REQ) είναι ουσιαστικά μια RREQ με αληθή τη σημαία *RouteUpdate* που γνωρίζει εξαρχής τη διαδρομή που θα ακολουθήσει, ώστε να ανανεώσει το πεδίο *MinEnergy* της διαδρομής αυτής. Τα πακέτα RU-REQ αποστέλλονται περιοδικά από κάθε κόμβο για όλες τις διαδρομές προς τους προορισμούς με τους οποίους υπάρχει τουλάχιστον μία ενεργή διαδρομή.
- Η απάντηση ανανέωσης διαδρομής (Route Update Reply – RU-REP) είναι ουσιαστικά ταυτόσημη με τη RREP, απλά η σημαία *RouteUpdate* είναι αληθής ώστε να αποφεύγονται κάποιες ενέργειες.

Πιο συγκεκριμένα, όταν ένας κόμπος εκπέμπει μια RU-REQ, δεν πρόκειται περί ευρυεκπομπής, αλλά περί unicast μετάδοσης στον ήδη γνωστό επόμενο κόμβο της διαδρομής. Ο επόμενος κόμβος που τη λαμβάνει, απλά εξετάζει το πεδίο *MinEnergy*, το ανανεώνει αν χρειάζεται, και επαναμεταδίδει το πακέτο στον επόμενο κόμβο της διαδρομής. Όταν το πακέτο φτάσει στον προορισμό, αποστέλλεται προς την πηγή RU-REP με την νέα *MinEnergy* της διαδρομής. Η πηγή της RU-REQ τελικά ανανεώνει τη μνήμη διαδρομών της μετά τη λήψη της RU-REP. Σημειώνεται ότι κατά τις ανανεώσεις διαδρομών, είναι απενεργοποιημένη η απάντηση ενδιάμεσων κόμβων με ήδη αποθηκευμένες διαδρομές, ώστε να γίνεται πραγματική ανανέωση των δεδομένων ανά τακτά χρονικά διαστήματα και όχι απλά ανακύκλωση παλαιότερων τιμών. Ακόμα, είναι απενεργοποιημένη η ανανέωση της μνήμης των ενδιάμεσων κόμβων, ώστε να μη δημιουργούνται νέες ενεργές διαδρομές, που θα επιβαρύνουν στη συνέχεια το δίκτυο με επιπρόσθετα πακέτα ανανέωσης διαδρομών.

Η μνήμη διαδρομών κάθε κόμβου εκτελεί τον παρακάτω αλγόριθμο ανά τακτά χρονικά διαστήματα (*RouteUpdateInterval),* ώστε να είναι πάντα ενημερωμένη, μειώνοντας παράλληλα την επιβάρυνση του δικτύου από άχρηστα πακέτα :

```
Για κάθε προορισμό που υπάρχει στη μνήμη {

Αν είναι γειτονικός κόμβος,
      Διάγραψε όλες τις διαδρομές εκτός της απευθείας ζεύξης.
Αλλιώς,
      Διάγραψε κάθε διαδρομή που περιέχει μέσα της μια άλλη
      συντομότερη διαδρομή (επικαλυπτόμενες διαδρομές).

Αν υπάρχει έστω και μία ενεργή διαδρομή προς τον προορισμό,
      Αν δεν είναι γειτονικός και υπάρχει μόνο μία διαδρομή,
            Στείλε RREQ (broadcast).
      Αλλιώς,
            Στείλε  RU-REQ για κάθε διαδρομή.
Αλλιώς (είναι όλες ανενεργές),
      Αν υπάρχει έστω και μία ληγμένη διαδρομή προς τον προορισμό,
            Διάγραψε όλες τις διαδρομές προς τον προορισμό.
}
```

Τα RU-REQ αποστέλλονται με κάποιο *Route Update Jitter*, που είναι σκόπιμα μεγάλο (της τάξης των δευτερολέπτων), ώστε να αποφεύγεται μεγάλη συμφόρηση στο δίκτυο. Για να εξηγήσουμε τη δεύτερη συνθήκη, θα δούμε πρώτα πώς χαρακτηρίζεται μια εγγραφή διαδρομής στην μνήμη διαδρομών :

- Ενεργή (active) διαδρομή, αν το χρονικό διάστημα από την τελευταία προσπέλασή της είναι μικρότερο του χρονικού ορίου *RouteActivityTime*.
- Ανενεργή (inactive) διαδρομή, αν το χρονικό διάστημα από την τελευταία προσπέλασή της είναι μεγαλύτερο του χρονικού ορίου *RouteActivityTime* και μικρότερο του χρονικού ορίου *RouteExpiryTime*.
- Ληγμένη (expired) διαδρομή, αν το χρονικό διάστημα από την τελευταία προσπέλασή της είναι μεγαλύτερο του χρονικού ορίου *RouteExpiryTime*.

Για να λειτουργεί το σχήμα αυτό, έχει οριστεί :

$$RouteActivityTime = \frac{RouteUpdateInterval}{2} \tag{4.9}$$

οπότε ενεργή θεωρείται μια διαδρομή που έχει χρησιμοποιηθεί κατά το δεύτερο ήμισυ του *RouteUpdateInterval*.

Ακόμα, πρέπει :

$$RouteExpiryTime \geq RouteUpdateInterval \tag{4.10}$$

Το *RouteExpiryTime* είναι συνήθως πολλαπλάσιο του *RouteUpdateInterval*.

Έτσι, η συνθήκη «αν υπάρχει έστω και μία *ενεργή* διαδρομή» σημαίνει ότι υπάρχει πρόσφατη αποστολή πακέτων προς τον προορισμό, οπότε πρέπει να ανανεωθούν όλες οι διαδρομές προς τον προορισμό αυτό, για να επιλέγεται πάντα η βέλτιστη.

Αν όλες οι διαδρομές προς έναν προορισμό είναι *ανενεργές*, τότε δε διαγράφονται από τη μνήμη γιατί υπάρχει περίπτωση να χρειαστούν κάποια στιγμή στο μέλλον, φυσικά

πριν λήξουν. Έτσι, αν προκύψει κάποιο πακέτο προς τον προορισμό αυτό, δε χρειάζεται να ξεκινήσει ανακάλυψη διαδρομής, αλλά χρησιμοποιείται μια ήδη υπάρχουσα διαδρομή, ενώ αποστέλλεται παράλληλα RU-REQ για κάθε διαδρομή, ώστε να ανανεωθούν οι τιμές της ενέργειας. Το κέρδος, λοιπόν, είναι αφενός η μείωση της καθυστέρησης στην αποστολή δεδομένων προς τον προορισμό, αφετέρου η μείωση της επιβάρυνσης δρομολόγησης, καθώς γίνεται ανανέωση πεπερασμένου αριθμού διαδρομών και όχι ευρυεκπομπή RREQ. Το μοναδικό κόστος αφορά τη διαθέσιμη μνήμη, όπου αν προκύψει πρόβλημα χώρου ξεπερνιέται εύκολα με την πολιτική διαγραφής Least Recently Used (LRU) της μνήμης.

Επομένως, ο μηχανισμός ανανέωσης εξασφαλίζει ότι η μνήμη είναι πάντα ενημερωμένη όσον αφορά τους ενεργούς προορισμούς, χωρίς όμως ένα επιβαρύνει το δίκτυο με επιπλέον φορτίο για ανενεργούς προορισμούς, ενώ διαγράφει τις απαρχαιωμένες διαδρομές ώστε να μην αποθηκεύονται ανακριβείς πληροφορίες.

4.3.3 Υλοποίηση

Το μοντέλο του EDSR έχει ενσωματωθεί σε έναν νέο κόμβο, τον *akis_manet_station_node_both_DSR*. Ο κόμβος αυτός περιέχει το μοντέλο της μπαταρίας που περιγράφηκε στην ενότητα 4.2, αλλά έχει τροποποιηθεί κατάλληλα και το στρώμα δικτύου του, δηλαδή η διεργασία ip, ώστε να μπορεί να δρομολογεί και με βάση το πρωτόκολλο EDSR. Στη συνέχεια, θα δούμε ένα προς ένα ποια αρχεία τροποποιήθηκαν και ποιες βασικές λειτουργίες επιτελεί το κάθε ένα.

Η διεργασία *akis_ip_dispatch* διαβάζει από τον χρήστη την ιδιότητα energy-aware, με βάση την οποία αποφασίζει αν θα δημιουργήσει τη διεργασία *akis_manet_mgr*, που θα καλέσει το EDSR, ή τη διεργασία *manet_mgr*, που θα καλέσει το DSR.

Η διεργασία *akis_manet_mgr* έχει τροποποιηθεί ώστε όταν ως πρωτόκολλο δρομολόγησης επιλέγεται το DSR, να γεννά τη διεργασία *akis_dsr_rte*, δηλαδή τη διεργασία του EDSR.

Η διεργασία *akis_dsr_rte* είναι ο πυρήνας του πρωτοκόλλου EDSR, όπου έχουν οι γίνει όλες οι βασικές αλλαγές και προσθήκες στη λειτουργία του. Έχει τροποποιηθεί κατάλληλα το διάβασμα παραμέτρων, ώστε να λαμβάνει υπόψη του τις νέες παραμέτρους του EDSR. Ακόμα, έχουν τροποποιηθεί οι συναρτήσεις λήψης πακέτων καθώς και οι συναρτήσεις επεξεργασίας των πακέτων ανάλογα με το είδος του πακέτου, οι συναρτήσεις δημιουργίας πακέτων, οι συναρτήσεις χειρισμού λήξης των χρονομέτρων καθώς και η συνάρτηση ανανέωσης της μνήμης διαδρομών. Επίσης, έχει δημιουργηθεί η συνάρτηση που διαβάζει την τρέχουσα ενέργεια του κόμβου από το αντικείμενο της μπαταρίας για να χρησιμοποιηθεί στη δρομολόγηση, η συνάρτηση που παράγει τα πακέτα RU-REQ και εκτελεί τις κατάλληλες ενέργειες στη μνήμη διαδρομών κάθε *RouteUpdateInterval*, η οποία καλεί τον εαυτό της με self interrupts, και η συνάρτηση που εκτυπώνει σε αρχείο τα περιεχόμενα της μνήμης διαδρομών, η οποία καλείται από την προηγούμενη συνάρτηση.

Οι συναρτήσεις της διεργασίας *akis_dsr_rte* κάνουν εκτεταμένη χρήση των συναρτήσεων που ορίζονται στα εξωτερικά αρχεία κώδικα, ενώ οι σταθερές και οι δομές δεδομένων που χρησιμοποιούνται έχουν οριστεί στα αντίστοιχα αρχεία επικεφαλίδων. Στη συνέχεια, γίνεται μια σύντομη αναφορά στα εξωτερικά αυτά αρχεία.

Στο αρχείο επικεφαλίδας *akis_dsr* ορίζονται κάποιες σταθερές καθώς και οι δομές δεδομένων για τα στατιστικά, τη μνήμη διαδρομών, τον ενταμιευτή αποστολής, τον πίνακα αιτήσεων διαδρομής, τον πίνακα απρόκλητων απαντήσεων διαδρομής, τον ενταμιευτή συντήρησης και τη μαύρη λίστα. Από τις δομές αυτές τροποποιήθηκε μόνο η μνήμη διαδρομών, ώστε να συμπεριλάβει την πληροφορία της ενέργειας του μονοπατιού.

Στο αρχείο επικεφαλίδας *akis_dsr_pkt_support* ορίζονται κάποιες σταθερές και δομές δεδομένων για την επικεφαλίδα Επιλογών DSR, δηλαδή οι επικεφαλίδες των επιλογών RREQ, RREP, RERR κ.ο.κ. Τροποποιήθηκαν οι επιλογές αίτησης διαδρομής, απάντησης διαδρομής και επιβεβαίωσης.

Στο αρχείο επικεφαλίδας *akis_dsr_ptypes* ορίζονται τα πρωτότυπα των συναρτήσεων που υλοποιούνται στα επόμενα εξωτερικά αρχεία πηγαίου κώδικα. Εδώ τροποποιήθηκαν πολλές συναρτήσεις και δημιουργήθηκαν νέες.

Τα αρχεία κώδικα στα οποία τροποποιήθηκαν κάποιες συναρτήσεις είναι τα *akis_dsr_route_cache* και *akis_dsr_pkt_support*, ενώ δημιουργήθηκε το *akis_dsr_extras*, για να συμπεριλάβει μια συνάρτηση από τη βιβλιοθήκη των MANET.

Τα υπόλοιπα αρχεία κώδικα που χρησιμοποιούνται αλλά στα οποία δεν υπήρξε ουσιαστική τροποποίηση του κώδικα είναι τα *akis_dsr_send_buffer*, *akis_dsr_maintenance_buffer*, *akis_dsr_route_discovery*, *akis_dsr_support*, *akis_dsr_notif_log_support.*

Σημειώνεται ότι όλα τα ονόματα αρχείων αλλά και οι συναρτήσεις που υπάρχουν εντός των αρχείων άλλαξαν όνομα, με την προσθήκη του προθέματος *akis* στην περίπτωση την αρχείων και τη μετατροπή του *dsr* σε *edsr* στην περίπτωση των συναρτήσεων. Αυτό έγινε για να μπορεί ο ίδιος κόμβος να χρησιμοποιήσει κατ' επιλογήν και τα δύο πρωτόκολλα DSR και EDSR, χωρίς να υπάρχουν διενέξεις μεταξύ των αρχείων που χρησιμοποιεί το καθένα.

Όσον αφορά τη διεπαφή με το χρήστη, οι *ιδιότητες* (attributes) που προστίθενται στον κόμβο είναι :

- Κάτω από την ετικέτα IP:
    - Energy-aware: (Απ)ενεργοποίηση του Energy-aware DSR (EDSR) (On/Off).
- Κάτω από την ετικέτα AD-HOC Routing Parameters:ip.akis_manet_mgr.DSR Parameters:

- Route Update Interval: Χρονικό διάστημα μεταξύ αποστολών πακέτων αίτησης ανανέωσης διαδρομής RU-REQ (sec).
- Route Update Jitter: Χρονικό διάστημα μέσα στο οποίο γίνεται η μετάδοση των πακέτων RU-REQ από τους κόμβους με τυχαίο τρόπο, ώστε να αποφευχθούν οι συγκρούσεις (sec).
- Route Cache Log: (Απ)ενεργοποίηση αρχείου μνήμης διαδρομών (On/Off). Αν ενεργοποιηθεί, τότε εκτυπώνονται σε αρχείο τα περιεχόμενα της μνήμης διαδρομών όλων των κόμβων κάθε *RouteUpdateInterval*.

Αυτή ήταν μια γενική περιγραφή της υλοποίησης του πρωτοκόλλου EDSR στο OPNET. Περισσότερες λεπτομέρειες καθώς και μέρος του πηγαίου κώδικα είναι διαθέσιμα στο Παράρτημα Β.

4.3.4 Προσομοιώσεις Σεναρίων EDSR

Το πρωτόκολλο EDSR υλοποιήθηκε στο OPNET, επειδή όμως ο κώδικας είναι εκτενής, οι λεπτομέρειες υλοποίησής του βρίσκονται στο παράρτημα. Για την αξιολόγηση του EDSR εκτελέστηκε πλήθος προσομοιώσεων, στις οποίες μελετήθηκαν σενάρια χρησιμοποιώντας ως πρωτόκολλο δρομολόγησης το DSR και έπειτα το EDSR, μεταβάλλοντας μάλιστα κάποιες παραμέτρους του, ώστε να συγκριθούν οι επιδόσεις τους.

Τα *χαρακτηριστικά* των προσομοιώσεων ρυθμίζονται όπως φαίνεται στον Πίνακα 4.3.

Σταθερά	**Τιμή Σταθεράς**
Χρόνος προσομοίωσης	15 min (900 sec)
Διαστάσεις χώρου	500m x 500m
Ραδιοσυχνότητα	2.4 GHz
Ρυθμός Μετάδοσης	11 Mbps
Ισχύς μετάδοσης	5 mW (6,9 dBm)
Ευαισθησία δέκτη	-80dBm
Αρχική ενέργεια μπαταρίας	40 J
Τάση παροχής μπαταρίας	5 V
Ένταση ρεύματος μετάδοσης	280 mA
Ένταση ρεύματος λήψης	180 mA
Μοντέλο κινητικότητας	Random Waypoint Mobility Model
Κατανομή Ταχύτητας	Uniform(0,10) m/s
Χρόνος Πάυσης	100 sec
Μέγεθος Πακέτου	1024 bits
Χρόνος μεταξύ διαδοχικών αφίξεων	1 sec
Κατάσταση αδράνειας	Απενεργοποιημένη
Κατανομή Jitter Ανανέωσης Διαδρομών	Uniform(0,3) sec
Route Expiry Time	300 sec (5 min)
Διάσωση Πακέτων	Ενεργοποιημένη
Μη Διαδιδόμενη RREQ (TTL = 1)	Ενεργοποιημένη

Πίνακας 4.3 : Χαρακτηριστικά προσομοιώσεων EDSR

Τα περισσότερα χαρακτηριστικά αναλύονται στην παράγραφο 4.2.3. Όσον αφορά την ακτίνα διάδοσης, με παρατήρηση των αποτελεσμάτων των προσομοιώσεων προέκυψε

ότι το EDSR λειτουργεί ικανοποιητικά σε δίκτυα όπου οι ακτίνες διάδοσης δεν αλληλοεπικαλύπτονται σε μεγάλο βαθμό, κάτι που σχετίζεται με την τοπολογία του δικτύου (πυκνότητα κόμβων) αλλά και τις ρυθμίσεις ισχύος μετάδοσης και λήψης. Δεδομένων και των διαστάσεων του χώρου, λοιπόν, επιλέγουμε κατάλληλες τιμές ώστε με βάση την εξίσωση (4.7) να έχουμε

$$R_{\max} = 222,4\text{m} \tag{4.11}$$

Καθώς οι προσομοιώσεις αφορούν τη σύγκριση των επιδόσεων μεταξύ δύο πρωτοκόλλων, η κατάσταση αδράνειας, η οποία αναφέρεται στα διαστήματα που ο κόμβος δε λαμβάνει ή μεταδίδει πακέτα αλλά «ακούει» το κανάλι, δε λαμβάνεται υπ' όψιν καθώς παραμένει σταθερή σε κάθε περίπτωση.

Το jitter ανανέωσης διαδρομών ακολουθεί ομοιόμορφη κατανομή στο διάστημα [0,3] sec, ώστε να αποφεύγονται οι συγκρούσεις κατά την αποστολή των πακέτων RU-REQ. Η τιμή του χρόνου λήξης διαδρομών είναι η προεπιλεγμένη, ενώ οι επιλογές διάσωσης πακέτων και μη διαδιδόμενες RREQ ενεργοποιούνται ώστε να μειωθεί η επιβάρυνση του δικτύου και να αυξηθεί η αποτελεσματικότητά του.

Οι *παράμετροι* των προσομοιώσεων EDSR, που μεταβάλλονται ανάλογα με την περίπτωση που μελετούμε, παρουσιάζονται στον Πίνακα 4.4.

Παράμετρος	**Πεδίο Τιμών**
Αριθμός κόμβων	10, 25, 50
Αριθμός πηγών	50%, 100% επί του αριθμού κόμβων
Κινητικότητα	On, Off
EDSR – Απάντηση με αποθηκευμένες διαδρομές	On, Off
EDSR – Route Update Interval	30, 60, 120 sec

Πίνακας 4.4 : Παράμετροι προσομοιώσεων EDSR

Σημειώνεται ότι σε όλες τις προσομοιώσεις του πρωτοκόλλου DSR, που χρησιμοποιούνται ως σημείο αναφοράς, ισχύουν τα χαρακτηριστικά του Πίνακα 4.3, ενώ επιπλέον η επιλογή για «απάντηση με αποθηκευμένες διαδρομές» είναι πάντα ενεργοποιημένη.

Τα *κριτήρια (metrics)* με βάση τα οποία θα αξιολογηθεί το πρωτόκολλο αφορούν τόσο την ενέργεια όσο και την αποδοτικότητα του πρωτοκόλλου. Πιο συγκεκριμένα είναι :

- Η διάρκεια ζωής του δικτύου : Ορίζεται ως το χρονικό διάστημα από την αρχή αποστολής δεδομένων έως τη στιγμή που για πρώτη φορά εξαντλείται η ενέργεια ενός κόμβου. Αν κανένας κόμβος δεν «πεθάνει» κατά την προσομοίωση, το χρησιμοποιούμενο κριτήριο σύγκρισης είναι η υπολειπόμενη ενέργεια του κόμβου με την ελάχιστη ενέργεια, αφού αναλογικά θα ήταν ο πρώτος που θα πέθαινε αν συνεχιζόταν η προσομοίωση. Αποτελεί και το βασικό κριτήριο αξιολόγησης του πρωτοκόλλου, καθώς το EDSR σχεδιάστηκε με στόχο την αύξηση της διάρκειας ζωής του δικτύου.

- Ο αριθμός των νεκρών κόμβων, αν υπάρχουν.
- Η μέση υπολειπόμενη ενέργεια των κόμβων.
- Η μέση από-άκρο-σε-άκρο καθυστέρηση (end-to-end delay) του δικτύου. Η από-άκρο-σε-άκρο καθυστέρηση ενός πακέτου είναι το χρονικό διάστημα που διανύεται από τη δημιουργία του πακέτου από την πηγή μέχρι την καταστροφή του πακέτου από τον προορισμό. Το κριτήριο αυτό χρησιμοποιείται στα γενικά σενάρια.

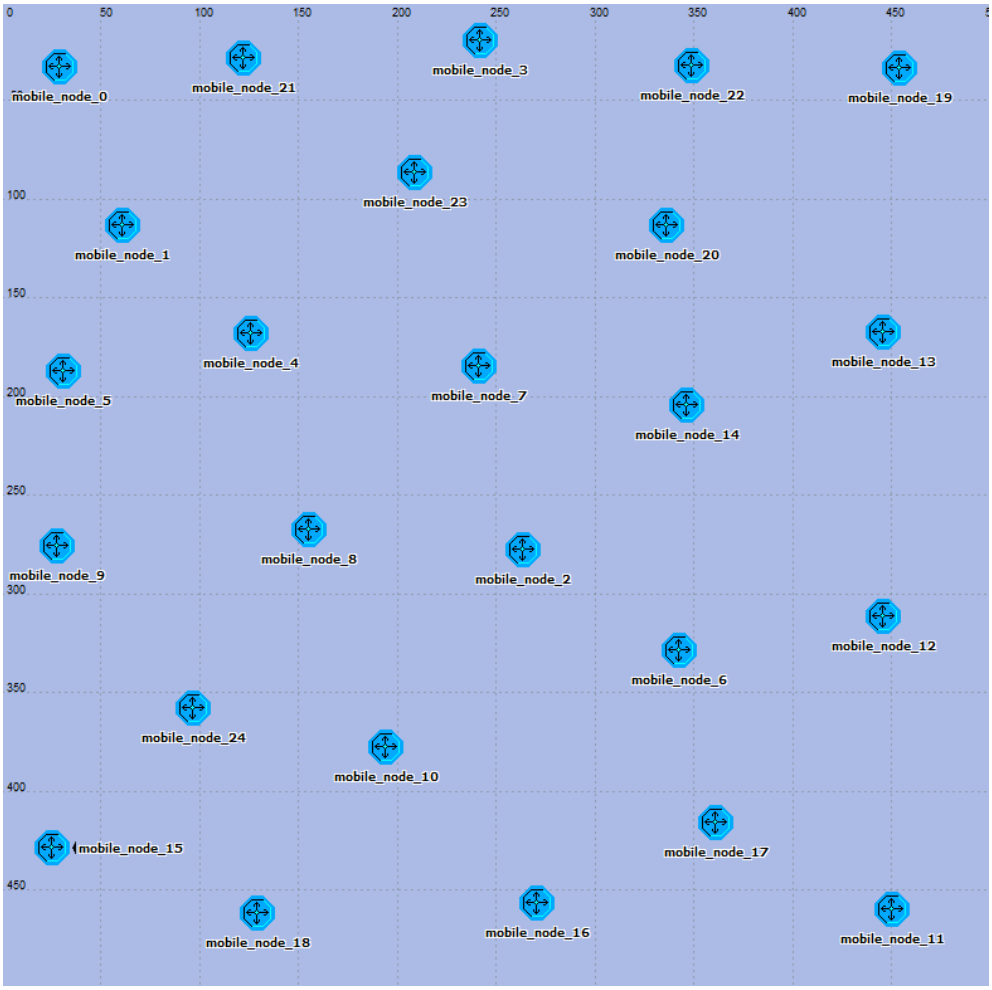
Οι προσομοιώσεις θα γίνουν σε δύο στάδια : Αρχικά, θα μελετηθεί ένα βασικό σενάριο μεταβάλλοντας τις παραμέτρους του EDSR, ώστε να βρεθεί ποιος είναι ο βέλτιστος συνδυασμός τους ανάλογα με το φορτίο και την κινητικότητα. Έπειτα, χρησιμοποιώντας τις προσαρμοσμένες αυτές παραμέτρους, θα μελετήσουμε τη συμπεριφορά του πρωτοκόλλου ως προς γενικότερες παραμέτρους, όπως ο αριθμός των κόμβων.

4.3.5 Προσομοίωση Βασικού Σεναρίου EDSR

Το Βασικό Σενάριο EDSR έχει τα χαρακτηριστικά του Πίνακα 4.3 με το δίκτυο να αποτελείται από 25 κόμβους που στέλνουν πακέτα των 1024 Bytes με χρόνο μεταξύ διαδοχικών πακέτων 1 sec προς τυχαίους προορισμούς.

Η τοπολογία του δικτύου είναι επίσης τυχαία, όπως φαίνεται στο Σχήμα 4.9. Οι μεταβλητές παράμετροι είναι :

- Ο αριθμός των πηγών κίνησης, που είναι είτε 13 είτε 25, και αντιστοιχούν σε χαμηλό και υψηλό φορτίο αντίστοιχα.
- Η κινητικότητα των κόμβων, η οποία αν είναι ενεργοποιημένη, προσδιορίζεται από ταχύτητες ομοιόμορφα κατανεμημένες στο διάστημα 0 – 10 m/s με χρόνο παύσης 100 sec.
- Η παράμετρος EDSR «Απάντηση με αποθηκευμένες διαδρομές», η οποία αν είναι ενεργοποιημένη, τότε ένας κόμβος που λαμβάνει μια RREQ προς άλλο κόμβο αναζητά στη μνήμη διαδρομών για ήδη αποθηκευμένες διαδρομές προς αυτόν. Αν βρεθεί κάποια *ενεργή* διαδρομή, ο κόμβος απαντά με RREP, αντί να προωθήσει τη RREQ. Δεν αφορά τα πακέτα RU-REQ.
- Η παράμετρος EDSR *Route Update Interval*, δηλαδή το διάστημα που μεσολαβεί ανάμεσα στις εκπομπές αιτήσεων ανανέωσης διαδρομής (RU-REQ) λαμβάνει τις τιμές 30, 60 και 120 sec.



Σχήμα 4.9 : Τοπολογία του Βασικού Σεναρίου EDSR

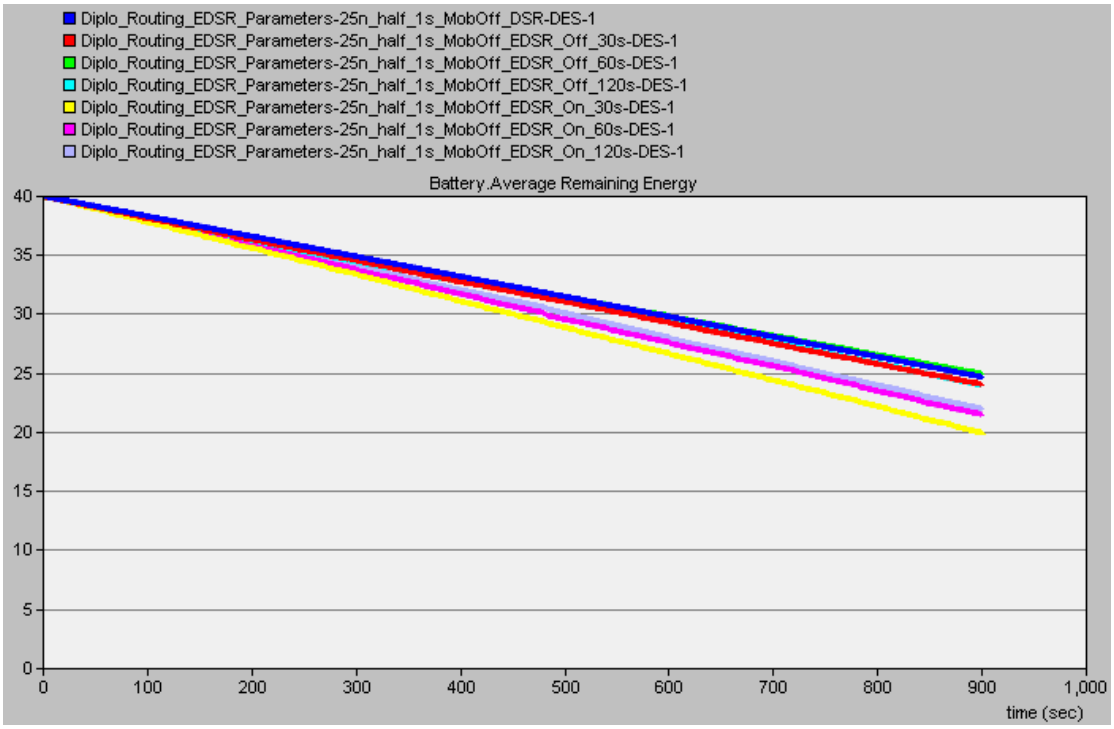
Εξετάζονται τώρα οι τέσσερις παραλλαγές του βασικού σεναρίου που προσομοιώθηκαν, δηλαδή όλοι οι συνδυασμοί φορτίου και κινητικότητας. Για κάθε παραλλαγή, μελετώνται όλοι οι συνδυασμοί (εκδοχές) των παραμέτρων «Απάντηση με αποθηκευμένες διαδρομές» και Route Update Interval.

4.3.5.1 Βασικό Σενάριο EDSR – Χαμηλό Φορτίο – Χωρίς Κινητικότητα

Στον Πίνακα 4.5 φαίνονται οι τιμές των παραμέτρων για κάθε εκδοχή του σεναρίου που προσομοιώθηκε. Εφόσον κανένας κόμβος δεν πεθαίνει, κριτήριο χρόνου ζωής του δικτύου είναι η ελάχιστη ενέργεια των κόμβων. Οι τιμές της ελάχιστης ενέργειας κόμβου καθώς και της μέσης υπολειπόμενης ενέργειας είναι οι τελικές τιμές, κατά τη χρονική στιγμή 900 sec. Οι γραφικές παραστάσεις των αποτελεσμάτων θα σχεδιαστούν συνολικά, μετά την εκτέλεση όλων των παραλλαγών του σεναρίου. Στο Σχήμα 4.10 απεικονίζεται η μέση υπολειπόμενη ενέργεια κάθε εκδοχής συναρτήσει του χρόνου, σε μια γραφική παράσταση που παράγει το OPNET. Παρατηρούμε ότι η μέση υπολειπόμενη ενέργεια κόμβου είναι γενικά μειωμένη στο EDSR, γεγονός που εξηγείται από τη μεγάλη επιβάρυνση που επιβάλλει ο μηχανισμός ανανέωσης διαδρομής υπό συνθήκες χαμηλού φορτίου.

Βασικό Σενάριο EDSR - Χαμηλό Φορτίο (13 πηγές) - Χωρίς Κινητικότητα					
Εκδοχή	**Πρωτόκολλο**	**Απαντήσεις με αποθηκευμένες διαδρομές**	**Route Update Interval (sec)**	**Ελάχιστη Ενέργεια Κόμβου (J)**	**Μέση Υπολειπόμενη Ενέργεια (J)**
1	DSR	On		18,5	24,7
2	EDSR	On	30	14,5	20
3	EDSR	On	60	17,5	21,5
4	EDSR	On	120	17,2	22
5	EDSR	Off	30	20,1	24
6	EDSR	Off	60	20,8	25
7	EDSR	Off	120	20	24

Πίνακας 4.5 : Αποτελέσματα Βασικού Σεναρίου EDSR με χαμηλό φορτίο, χωρίς κινητικότητα



Σχήμα 4.10 : Μέση υπολειπόμενη ενέργεια κόμβων συναρτήσει του χρόνου στο Βασικό Σενάριο EDSR με χαμηλό φορτίο, χωρίς κινητικότητα

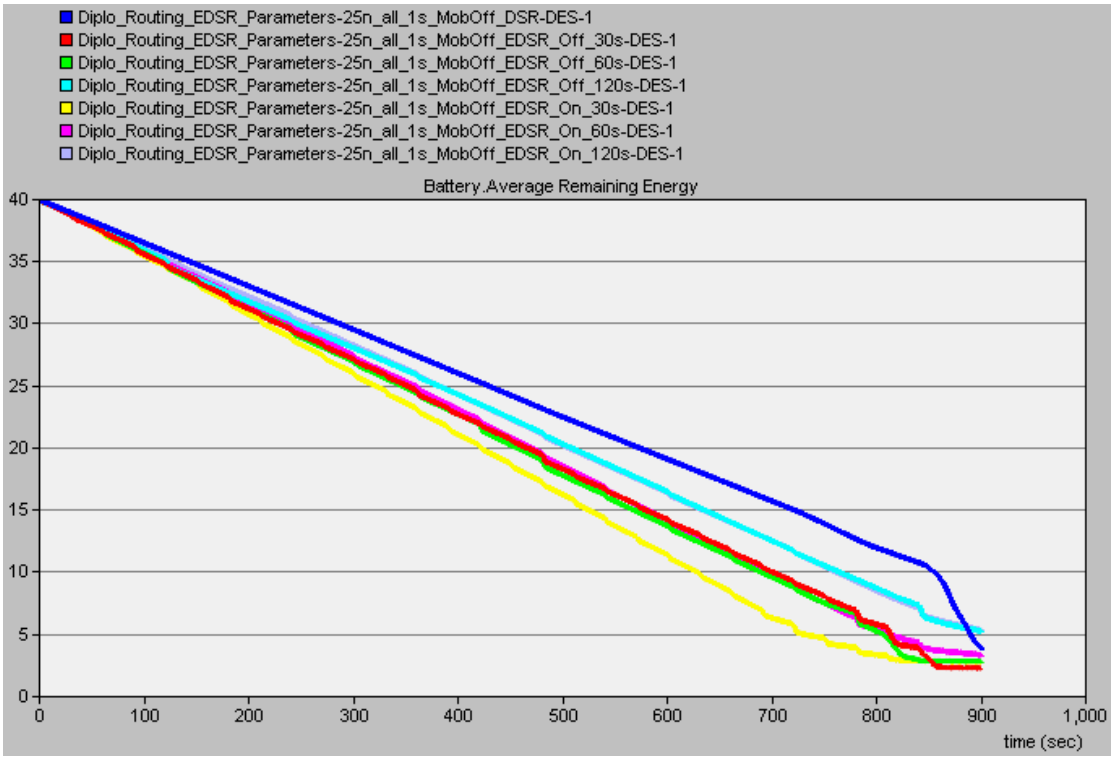
4.3.5.2 Βασικό Σενάριο EDSR – Υψηλό Φορτίο – Χωρίς Κινητικότητα

Στην περίπτωση αυτή τουλάχιστον ένας κόμβος του δικτύου πεθαίνει λόγω ενεργειακής εξάντλησης. Έτσι, αντί για ελάχιστη ενέργεια κόμβων, χρησιμοποιούμε ως χρονικό κριτήριο τη διάρκεια ζωής του δικτύου, όπως ορίστηκε προηγουμένως. Στο Σχήμα 4.11 απεικονίζεται η μέση υπολειπόμενη ενέργεια συναρτήσει του χρόνου. Παρατηρούμε ότι η μέση υπολειπόμενη ενέργεια κόμβων είναι και πάλι χαμηλότερη στο EDSR, παρά την απότομη τελική πτώση του DSR που οφείλεται στο εκρηκτικό φορτίο δρομολόγησης που προκάλεσε ο σχεδόν ταυτόχρονος θάνατος πολλών κόμβων. Στο Σχήμα 4.12 παρουσιάζεται ο αριθμός των νεκρών κόμβων για

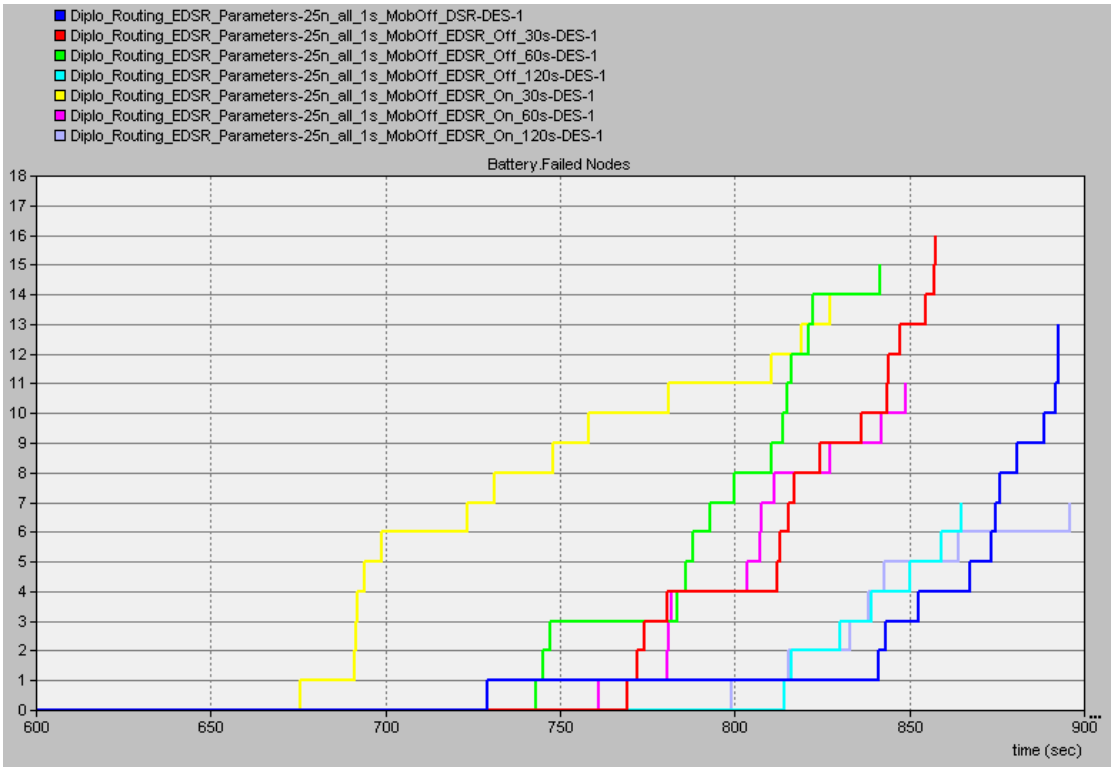
κάθε εκδοχή συναρτήσει του χρόνου, όπου επιβεβαιώνεται η καλή επίδοση του DSR μέχρις ότου συνέβη ο μαζικός θάνατος των κόμβων περί τα 850 sec.

Βασικό Σενάριο EDSR - Υψηλό Φορτίο (25 πηγές) - Χωρίς Κινητικότητα						
Εκδοχή	**Πρωτόκολλο**	**Απαντήσεις με αποθηκευμένες διαδρομές**	**Route Update Interval (sec)**	**Διάρκεια Ζωής Δικτύου (sec)**	**Μέση Υπολειπόμενη Ενέργεια (J)**	**Πλήθος Νεκρών Κόμβων**
1	DSR	On		729	4	13
2	EDSR	On	30	676	2,8	14
3	EDSR	On	60	761	3,3	11
4	EDSR	On	120	799	5,4	7
5	EDSR	Off	30	769	2,2	16
6	EDSR	Off	60	743	2,8	15
7	EDSR	Off	120	814	5,5	7

Πίνακας 4.6 : Αποτελέσματα Βασικού Σεναρίου EDSR με υψηλό φορτίο, χωρίς κινητικότητα



Σχήμα 4.11 : Μέση υπολειπόμενη ενέργεια κόμβων συναρτήσει του χρόνου στο Βασικό Σενάριο EDSR με υψηλό φορτίο, χωρίς κινητικότητα



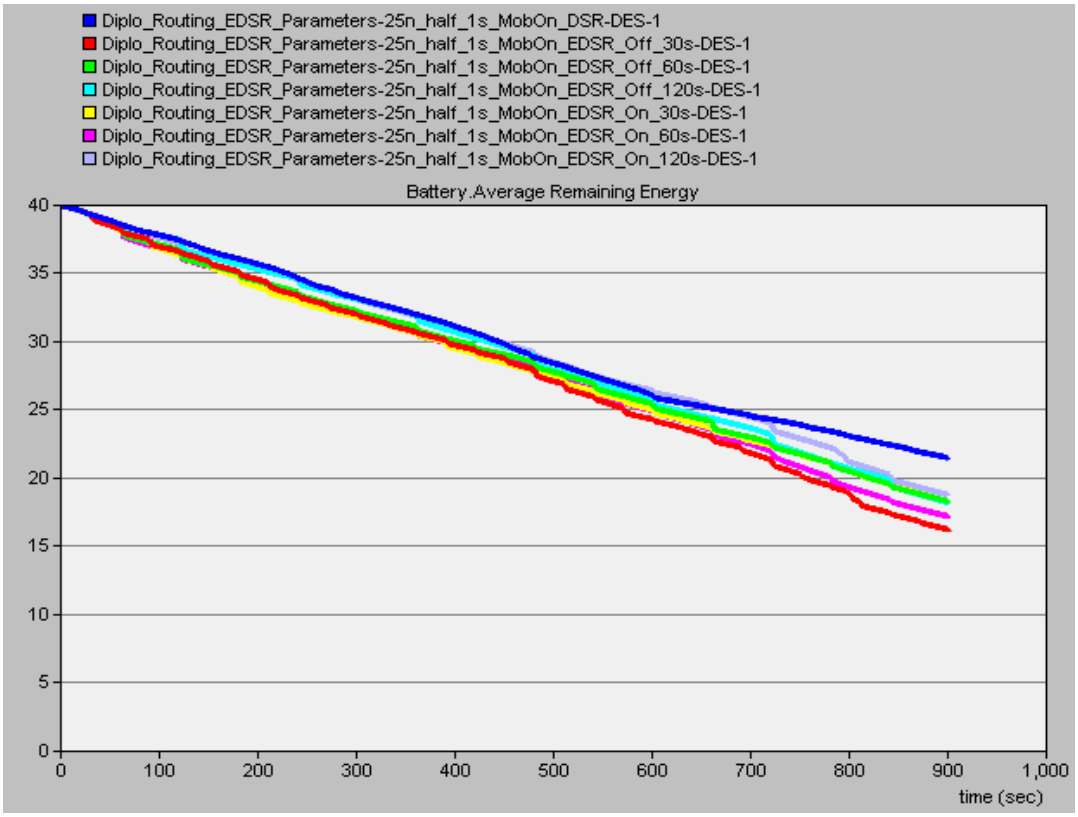
Σχήμα 4.12 : Πλήθος νεκρών κόμβων συναρτήσει του χρόνου στο Βασικό Σενάριο EDSR με υψηλό φορτίο, χωρίς κινητικότητα

4.3.5.3 Βασικό Σενάριο EDSR – Χαμηλό Φορτίο – Με Κινητικότητα

Στην περίπτωση αυτή οι κόμβοι κινούνται τυχαία με βάση το μοντέλο Random Waypoint. Η ταχύτητά τους είναι ομοιόμορφα κατανεμημένη στο διάστημα (0,10) m/s, ενώ ο χρόνος παύσης είναι 100 sec. Σημειώνεται ότι σε κάθε εκτέλεση προσομοίωσης η τροχιά κάθε κόμβου είναι τυχαία και διαφορετική από κάθε άλλη εκτέλεση. Στο Σχήμα 4.13 απεικονίζεται η μέση υπολειπόμενη ενέργεια συναρτήσει του χρόνου, όπου το DSR φαίνεται να υπερτερεί του EDSR.

Βασικό Σενάριο EDSR - Χαμηλό Φορτίο (13 πηγές) - Με Κινητικότητα					
Εκδοχή	**Πρωτόκολλο**	**Απαντήσεις με αποθηκευμένες διαδρομές**	**Route Update Interval (sec)**	**Ελάχιστη Ενέργεια Κόμβου (J)**	**Μέση Υπολειπόμενη Ενέργεια (J)**
1	DSR	On		17	21,4
2	EDSR	On	30	14,1	18,3
3	EDSR	On	60	12,2	17,2
4	EDSR	On	120	14,3	18,8
5	EDSR	Off	30	11,2	16,2
6	EDSR	Off	60	13,9	18,3
7	EDSR	Off	120	13,2	18,2

Πίνακας 4.7 : Αποτελέσματα Βασικού Σεναρίου EDSR με χαμηλό φορτίο, με κινητικότητα



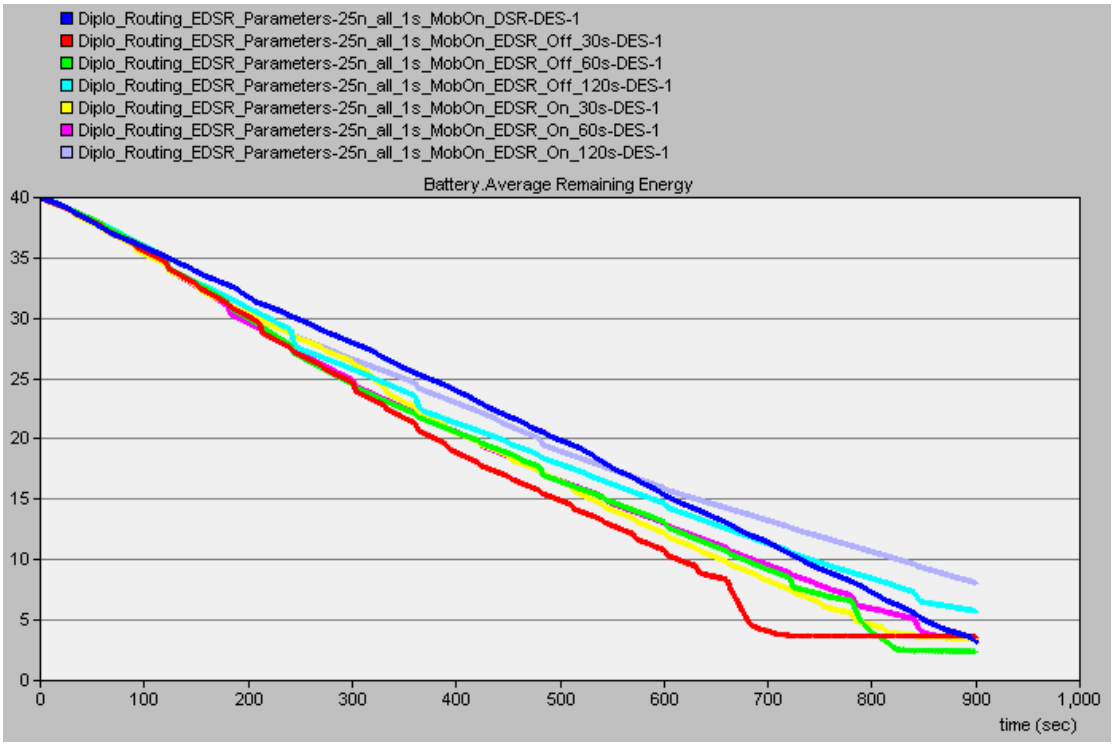
Σχήμα 4.13 : Μέση υπολειπόμενη ενέργεια κόμβων συναρτήσει του χρόνου στο Βασικό Σενάριο EDSR με χαμηλό φορτίο, με κινητικότητα

4.3.5.4 Βασικό Σενάριο EDSR – Υψηλό Φορτίο – Με Κινητικότητα

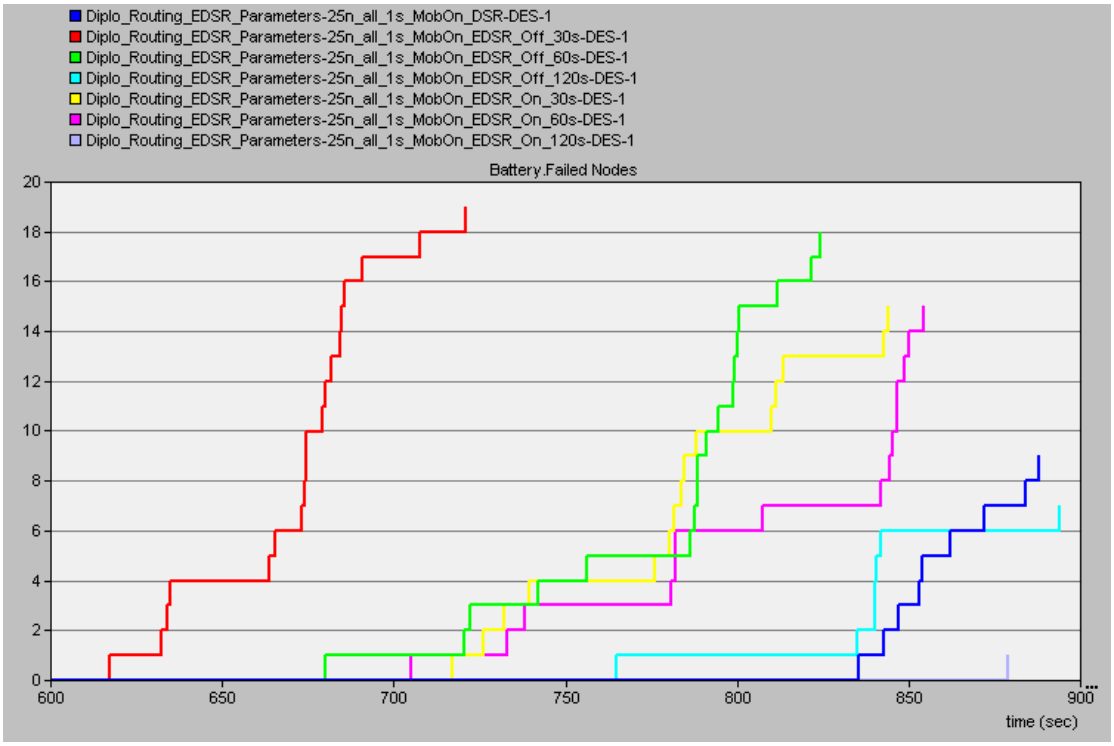
Στο υψηλό φορτίο προκύπτουν και πάλι νεκροί κόμβοι, οπότε ως κριτήριο χρησιμοποιείται η διάρκεια ζωής του δικτύου. Οι κόμβοι κινούνται τυχαία με βάση το μοντέλο Random Waypoint όπως και προηγουμένως. Στο Σχήμα 4.14 παρατηρούμε ότι οι εκδοχές του EDSR με υψηλό Route Update Interval αυξάνουν τη μέση υπολειπόμενη ενέργεια των κόμβων, ενώ στο Σχήμα 4.15 παρατηρείται και πάλι μαζική πτώση των κόμβων στο DSR περί τα 850 sec.

Βασικό Σενάριο EDSR - Υψηλό Φορτίο (25 πηγές) - Με Κινητικότητα						
Εκδοχή	**Πρωτόκολλο**	**Απαντήσεις με αποθηκευμένες διαδρομές**	**Route Update Interval (sec)**	**Διάρκεια Ζωής Δικτύου (sec)**	**Μέση Υπολειπόμενη Ενέργεια (J)**	**Πλήθος Νεκρών Κόμβων**
1	DSR	On		836	3,2	9
2	EDSR	On	30	617	3,6	19
3	EDSR	On	60	680	2,4	18
4	EDSR	On	120	765	5,7	7
5	EDSR	Off	30	717	3,4	15
6	EDSR	Off	60	705	3,6	15
7	EDSR	Off	120	879	8,1	1

Πίνακας 4.8 : Αποτελέσματα Βασικού Σεναρίου EDSR με υψηλό φορτίο, με κινητικότητα



Σχήμα 4.14 : Μέση υπολειπόμενη ενέργεια κόμβων συναρτήσει του χρόνου στο Βασικό Σενάριο EDSR με υψηλό φορτίο, με κινητικότητα

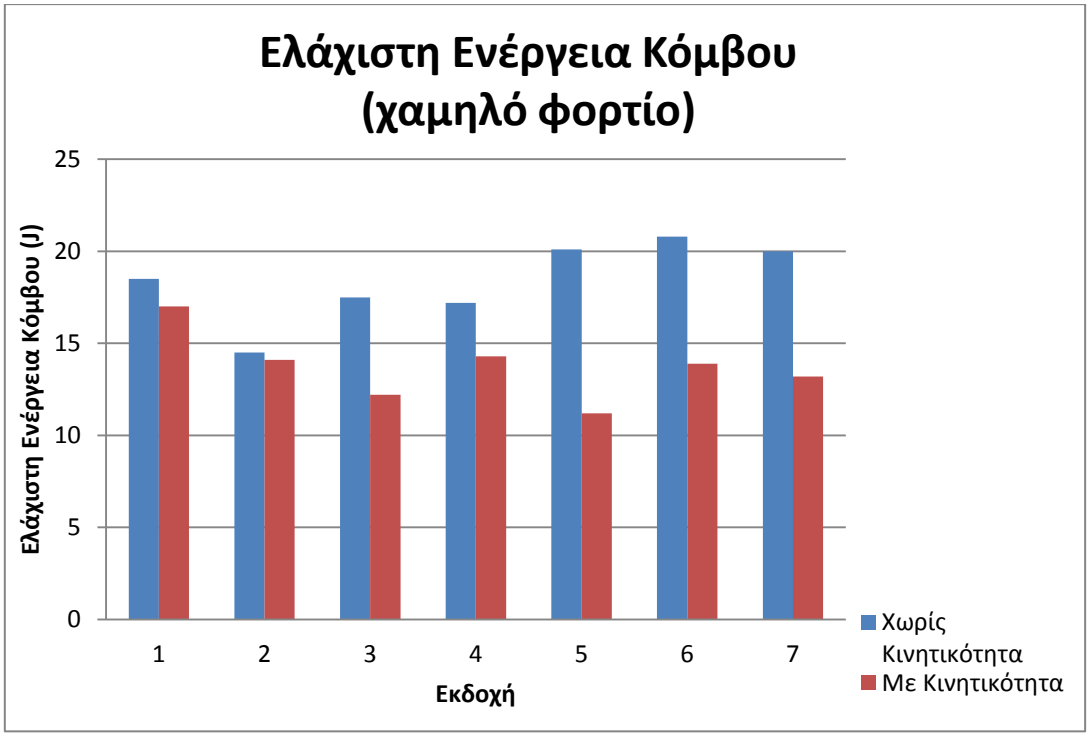


Σχήμα 4.15 : Πλήθος νεκρών κόμβων συναρτήσει του χρόνου στο Βασικό Σενάριο EDSR με υψηλό φορτίο, με κινητικότητα

4.3.6 Αποτελέσματα Βασικού Σεναρίου

Τα αποτελέσματα των Πινάκων 4.5 – 4.8 παρουσιάζονται στα επόμενα Σχήματα 4.16 – 4.19. Παρατηρείται η μεγάλη εξάρτηση της επίδοσης του πρωτοκόλλου EDSR από τις παραμέτρους του. Γι' αυτό και θα αναζητηθούν οι βέλτιστες επιδόσεις, ώστε να συσχετισθούν με τις αντίστοιχες παραμέτρους του EDSR. Υπενθυμίζεται ότι το EDSR σχεδιάστηκε με στόχο την αύξηση της διάρκειας ζωής του δικτύου, που αποτελεί και το βασικό κριτήριο αξιολόγησης της επίδοσής του. Η μέση υπολειπόμενη ενέργεια είναι δευτερεύον κριτήριο, που απλά συμβάλλει στην καλύτερη εποπτεία της κατάστασης του δικτύου.

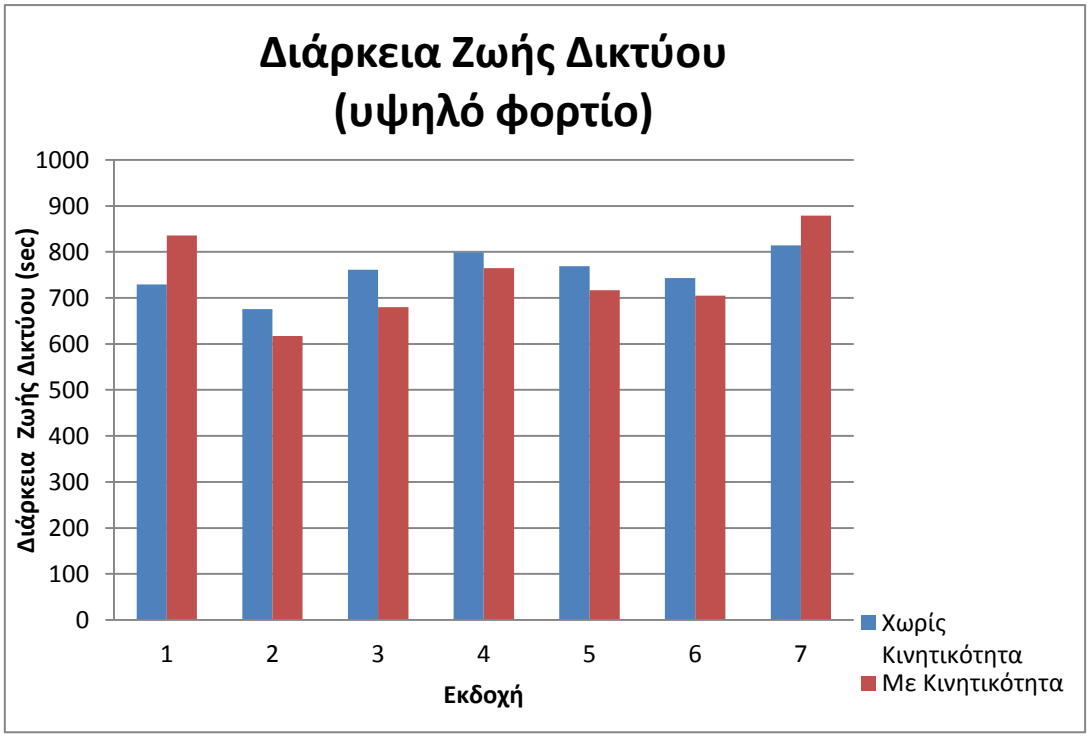
Στις προσομοιώσεις χαμηλού φορτίου η σύγκριση γίνεται με την ελάχιστη ενέργεια κόμβου, ενώ στις προσομοιώσεις υψηλού φορτίου με τη διάρκεια ζωής του δικτύου και το πλήθος των νεκρών κόμβων. Έτσι, παρουσιάζονται συγκεντρωτικά τα αποτελέσματα με βάση το φορτίο τους. Η μέση υπολειπόμενη ενέργεια αποτελεί κριτήριο σε κάθε περίπτωση, γι' αυτό και περιέχει και τις τέσσερις παραλλαγές. Οι εκδοχές είναι συγκεκριμένες και αμετάβλητες για κάθε παραλλαγή του σεναρίου. Έτσι, συγκρίνεται η εκδοχή 1, που αφορά το DSR, με καθεμία από τις άλλες εκδοχές, που αφορούν το EDSR με διαφορετικές τιμές παραμέτρων.



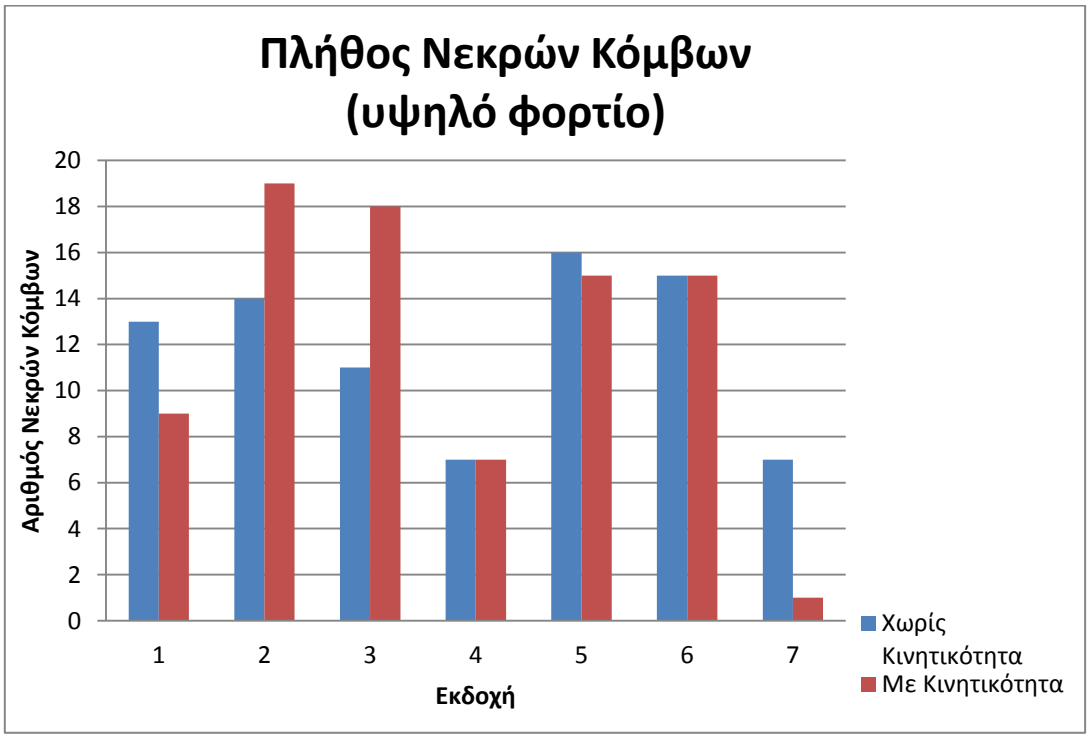
Σχήμα 4.16 : Ελάχιστη ενέργεια κόμβου στο Βασικό Σενάριο EDSR με χαμηλό φορτίο

Από το Σχήμα 4.16 προκύπτει ότι σε συνθήκες χαμηλού φορτίου η κινητικότητα επηρεάζει άμεσα την ενεργειακή απόδοση του πρωτοκόλλου, καθώς είναι απρόβλεπτη και πολλές φορές έχει τυχαίες επιπτώσεις στη λειτουργία του. Χωρίς κινητικότητα παρατηρούμε αύξηση της ελάχιστης ενέργειας κόμβου όταν η

«Απάντηση με αποθηκευμένες διαδρομές» είναι απενεργοποιημένη, ενώ η μεγαλύτερη αύξηση 12% παρατηρείται για Route Update Interval ίσο με 60 sec (Εκδοχή 6). Με κινητικότητα, το πρωτόκολλο EDSR έχει χειρότερες ενεργειακές επιδόσεις σε σχέση με το DSR.



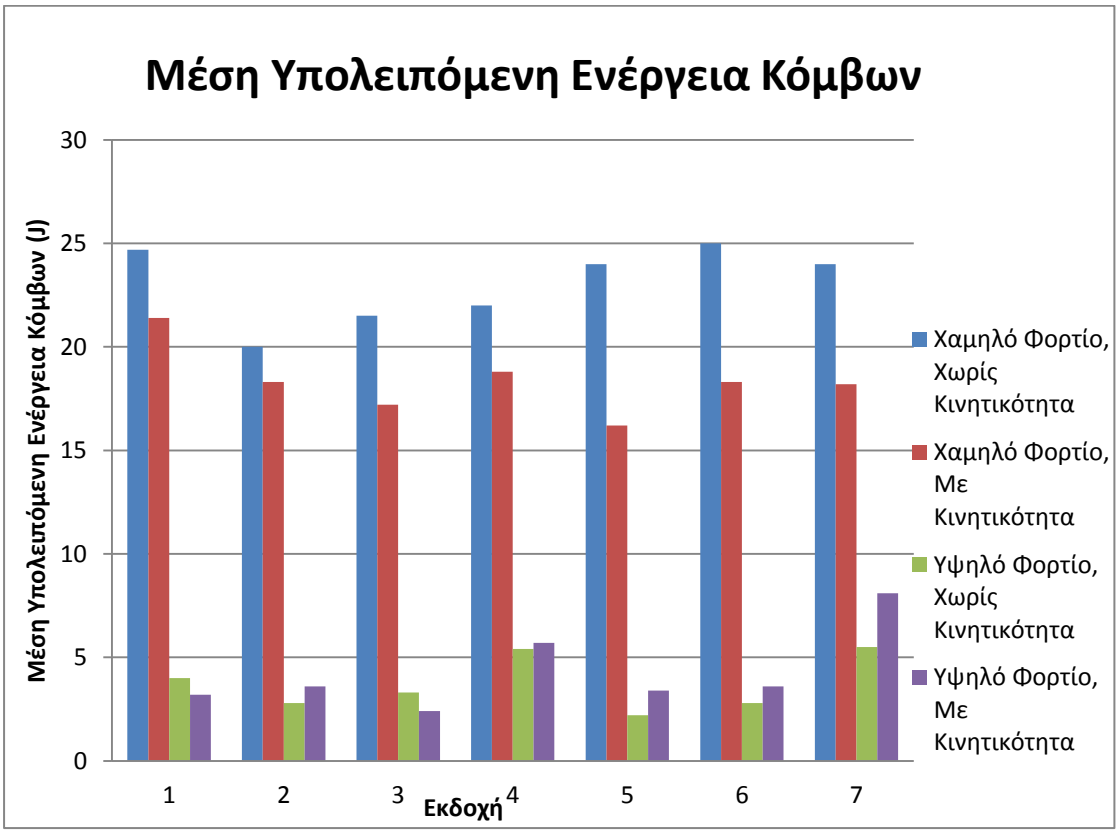
Σχήμα 4.17 : Διάρκεια ζωής δικτύου στο Βασικό Σενάριο EDSR με υψηλό φορτίο



Σχήμα 4.18 : Πλήθος νεκρών κόμβων στο Βασικό Σενάριο EDSR με υψηλό φορτίο

Από τα Σχήματα 4.17 και 4.18 προκύπτει ότι και σε συνθήκες υψηλού φορτίου η κινητικότητα έχει απρόβλεπτες επιδράσεις. Χωρίς κινητικότητα παρατηρείται βελτίωση των επιδόσεων με την αύξηση του Route Update Interval, ενώ η μεγαλύτερη αύξηση της διάρκειας ζωής του δικτύου είναι 11% για απενεργοποιημένη «Απάντηση με αποθηκευμένες διαδρομές» και Route Update Interval ίσο με 120 sec (Εκδοχή 7). Παρόμοια συμπεριφορά προκύπτει και με βάση το κριτήριο του πλήθους νεκρών κόμβων.

Με κινητικότητα, οι επιδόσεις γενικά χειροτερεύουν εκτός από την εκδοχή 7, όπου έχουμε αύξηση 5% της διάρκειας ζωής του δικτύου και μείωση των νεκρών κόμβων σε μόλις έναν από τους εννιά του DSR, κάτι που πιθανόν οφείλεται στην τυχαία κινητικότητα των κόμβων. Έτσι, είναι δυνατόν το EDSR να έχει θετικά αποτελέσματα και σε δίκτυα με κινητούς κόμβους, δεν έχει όμως σταθερή απόδοση αλλά εξαρτάται από την κίνηση των κόμβων στο χώρο.



Σχήμα 4.19 : Μέση υπολειπόμενη ενέργεια κόμβων στο Βασικό Σενάριο EDSR

Στο Σχήμα 4.19 παρουσιάζεται το δευτερεύον κριτήριο αξιολόγησης του EDSR, η μέση υπολειπόμενη ενέργεια των κόμβων. Υπό συνθήκες χαμηλού φορτίου η μέση ενέργεια των κόμβων είναι γενικά χαμηλότερη σε σχέση με το DSR, καθώς η επιβάρυνση από το επιπλέον φορτίο του δικτύου λόγω ανανεώσεων διαδρομής αποτελεί μεγάλο τμήμα της κίνησης του δικτύου. Εξάλλου είναι λογικό οι κόμβοι να είναι πιο ενεργοβόροι στο EDSR λόγω του μηχανισμού ανανέωσης διαδρομής, ο οποίος στοχεύει στην αύξηση της διάρκειας ζωής του δικτύου, κάτι που επιτυγχάνεται όταν η ενέργεια καταναλώνεται ομοιόμορφα ανάμεσα στους κόμβους.

Υπό συνθήκες υψηλού φορτίου, η επιβάρυνση αυτή κατέχει μικρότερο ποσοστό της συνολικής κίνησης, γι' αυτό και είναι πιθανόν το EDSR να βελτιώσει την επίδοση του ως προς το κριτήριο αυτό, όπως φαίνεται και στο Σχήμα 4.15. Όταν, όμως, η μέση υπολειπόμενη ενέργεια πλησιάζει το μηδέν, παύει να αποτελεί αξιόπιστο κριτήριο, γιατί πλέον πολλοί κόμβοι του δικτύου είναι νεκροί και με μειωμένες τις πηγές και τους προορισμούς, δεν είναι δυνατή η αποστολή δεδομένων από τους κόμβους που έχουν μείνει ζωντανοί.

Γενικότερα, φαίνεται ότι η «*απάντηση με αποθηκευμένες διαδρομές*» επιδρά αρνητικά στην επίδοση του EDSR, καθώς οι πληροφορίες που διακινεί μπορεί να είναι απαρχαιωμένες και να δυσχεραίνουν έτσι την επιλογή της βέλτιστης ενεργειακά διαδρομής. Αυτό σημαίνει επιπλέον ότι η επιβάρυνση που προέρχεται από τη μη χρησιμοποίησή του είναι αμελητέα. Επίσης, σε περίπτωση υψηλής κινητικότητας όπου οι διαδρομές από έναν κόμβο προς έναν προορισμό μεταβάλλονται γρήγορα, οι πληροφορίες του μπορεί να είναι ακόμα και εσφαλμένες, δηλαδή οι διαδρομές να μην υπάρχουν πια.

Το χρονικό διάστημα *Route Update Interval* έχει εμφανή επίδραση στις επιδόσεις του EDSR. Για μικρές τιμές όπως 30 sec, η επιβάρυνση του δικτύου λόγω ανανεώσεων διαδρομής είναι τόσο μεγάλη, ώστε πάντα υπάρχουν καλύτερες επιδόσεις για μεγαλύτερες τιμές του. Η τιμή 120 sec επικρατεί, κάτι που δείχνει ότι όσο λιγότερο είναι το επιπλέον φορτίο στο δίκτυο, τόσο καλύτερες οι ενεργειακές επιδόσεις του EDSR. Έτσι, στο tradeoff μεταξύ επιβάρυνσης δρομολόγησης (routing overhead) και μη έγκυρων πληροφοριών δρομολόγησης προκύπτει ότι η επιβάρυνση έχει μεγαλύτερη επίπτωση στα ενεργειακά κριτήρια και αξίζει μεγαλύτερης προσοχής κατά τη σχεδίαση πρωτοκόλλων δρομολόγησης με ενεργειακά κριτήρια.

Επομένως, το πρωτόκολλο δρομολόγησης EDSR είναι δυνατόν να αυξήσει τη διάρκεια ζωής των κόμβων ενός δικτύου MANET με την κατάλληλη επιλογή των παραμέτρων του. Με βάση τους πίνακες, τις γραφικές παραστάσεις αλλά και τα όσα συζητήθηκαν παραπάνω, γίνεται η επιλογή των βέλτιστων παραμέτρων του πρωτοκόλλου EDSR για κάθε συνδυασμό φορτίου και κινητικότητας, όπως φαίνεται στον Πίνακα 4.9. Οι τιμές αυτές θα χρησιμοποιηθούν στην επόμενη ενότητα για τις προσομοιώσεις των γενικών σεναρίων.

Φορτίο	**Κινητικότητα**	**Βέλτιστη Εκδοχή**	**Απαντήσεις με αποθηκευμένες διαδρομές**	**Route Update Interval (sec)**
Χαμηλό	Όχι	6	Όχι	60
Χαμηλό	Ναι	4	Ναι	120
Υψηλό	Όχι	7	Όχι	120
Υψηλό	Ναι	7	Όχι	120

Πίνακας 4.9 : Επιλογή παραμέτρων EDSR συναρτήσει φορτίου και κινητικότητας

4.3.7 Προσομοίωση Γενικών Σεναρίων EDSR

Χρησιμοποιώντας τον Πίνακα 4.9, που περιέχει τις βέλτιστες τιμές των παραμέτρων του EDSR κατά περίπτωση, θα μελετήσουμε τώρα τη συμπεριφορά του EDSR σε γενικότερα σενάρια. Στόχος πλέον δεν είναι η επιλογή παραμέτρων, αλλά η αξιολόγηση της επίδοσης του πρωτοκόλλου σε κάθε περίπτωση. Τα χαρακτηριστικά του δικτύου παραμένουν αυτά του Πίνακα 4.3, ενώ η τοπολογία είναι τυχαία. Οι παράμετροι των γενικών σεναρίων φαίνονται στον Πίνακα 4.10.

Παράμετρος	**Πεδίο Τιμών**
Αριθμός κόμβων	10, 25, 40
Αριθμός πηγών	50%, 100% επί του αριθμού κόμβων
Κινητικότητα	Ναι, Όχι

Πίνακας 4.10 : Παράμετροι Γενικών Σεναρίων EDSR

Έτσι, για τρεις διαφορετικούς αριθμούς κόμβων θα συγκριθεί η επίδοση του EDSR με αυτή του DSR υπό συνθήκες χαμηλού και υψηλού φορτίου, με ή χωρίς κινητικότητα.

Για να είναι δυνατή η άμεση σύγκριση των αποτελεσμάτων κάθε σεναρίου, ο χρόνος προσομοίωσης αυξάνεται τόσο, ώστε πάντα να υπάρχει τουλάχιστον ένας κόμβος που «πεθαίνει» ενεργειακά. Επομένως, τα *κριτήρια (metrics)* αξιολόγησης μεταβάλλονται ως εξής :

- Η διάρκεια ζωής του δικτύου : Ορίζεται ως το χρονικό διάστημα από την αρχή αποστολής δεδομένων έως τη στιγμή που για πρώτη φορά εξαντλείται η ενέργεια ενός κόμβου.
- Η μέση υπολειπόμενη ενέργεια των κόμβων τη χρονική στιγμή του πρώτου «θανάτου» κόμβου στο δίκτυο.
- Η μέση από-άκρο-σε-άκρο καθυστέρηση (end-to-end delay) του δικτύου. Η από-άκρο-σε-άκρο καθυστέρηση ενός πακέτου είναι το χρονικό διάστημα που διανύεται από τη δημιουργία του πακέτου από την πηγή μέχρι την καταστροφή του πακέτου από τον προορισμό. Εξετάζεται τη χρονική στιγμή του πρώτου «θανάτου» κόμβου στο δίκτυο.

Επειδή πλέον ο χρόνος προσομοίωσης είναι μεταβλητός, επικεντρωνόμαστε στη χρονική στιγμή του πρώτου ενεργειακού θανάτου. Έτσι, το πλήθος νεκρών κόμβων δεν έχει πλέον σημασία, καθώς τη συγκεκριμένη στιγμή είναι προφανώς μονάδα, ενώ στο τέλος της προσομοίωσης δεν είναι χρήσιμο δεδομένο, αφού κάθε σενάριο έχει διαφορετική διάρκεια. Η μέση υπολειπόμενη ενέργεια των κόμβων τη συγκεκριμένη χρονική στιγμή δείχνει τη συμπεριφορά του πρωτοκόλλου ως προς τη μέση κατανάλωση ενέργειας στο δίκτυο. Η καθυστέρηση είναι ένα μη ενεργειακό μέτρο σύγκρισης, που χρησιμοποιείται κυρίως σε δίκτυα δεδομένων ευαίσθητων ως προς την καθυστέρηση, όπως η φωνή ή το βίντεο.

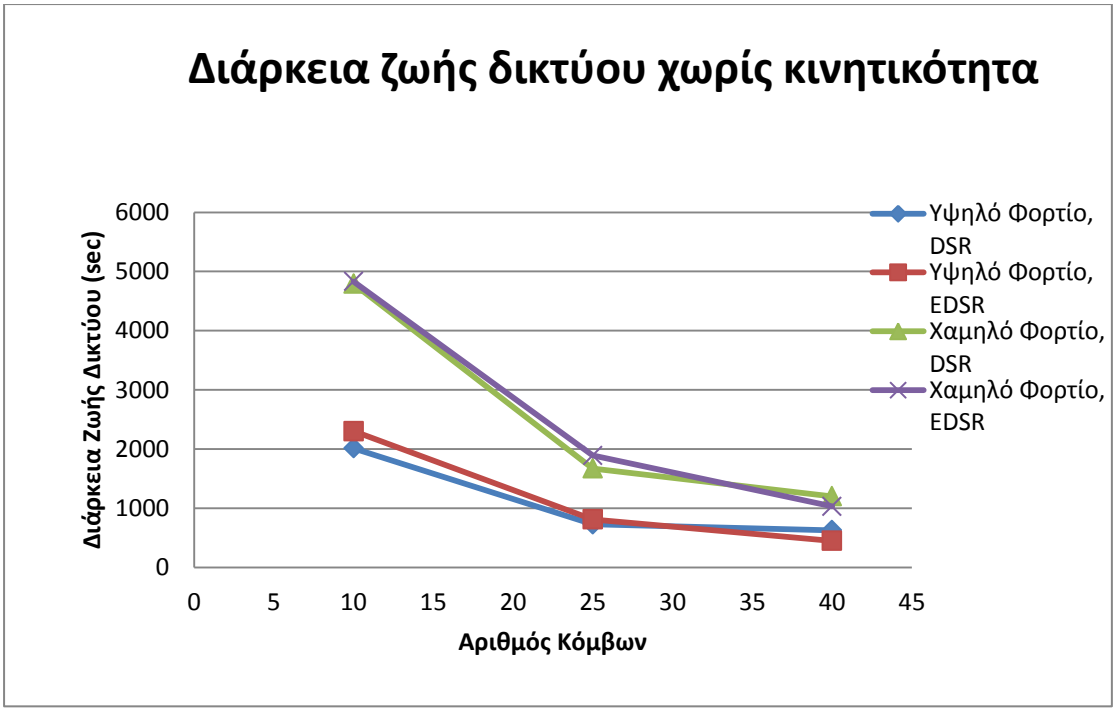
4.3.8 Αποτελέσματα Γενικών Σεναρίων EDSR

Τα αποτελέσματα των προσομοιώσεων παρουσιάζονται στον Πίνακα 4.11. Σημειώνεται ότι χαμηλό/υψηλό φορτίο σημαίνει 5/10, 13/25, και 20/40 πηγές για αριθμό κόμβων 10, 25 και 40 αντίστοιχα. Ακόμα, υπενθυμίζεται ότι ανάλογα με το συνδυασμό φορτίου – κινητικότητας χρησιμοποιήθηκαν οι παράμετροι όπως φαίνονται στον Πίνακα 4.9. Η μέση υπολειπόμενη ενέργεια και η μέση καθυστέρηση έχουν καταγραφεί τη στιγμή του πρώτου θανάτου ενός κόμβου του δικτύου (διάρκεια ζωής δικτύου).

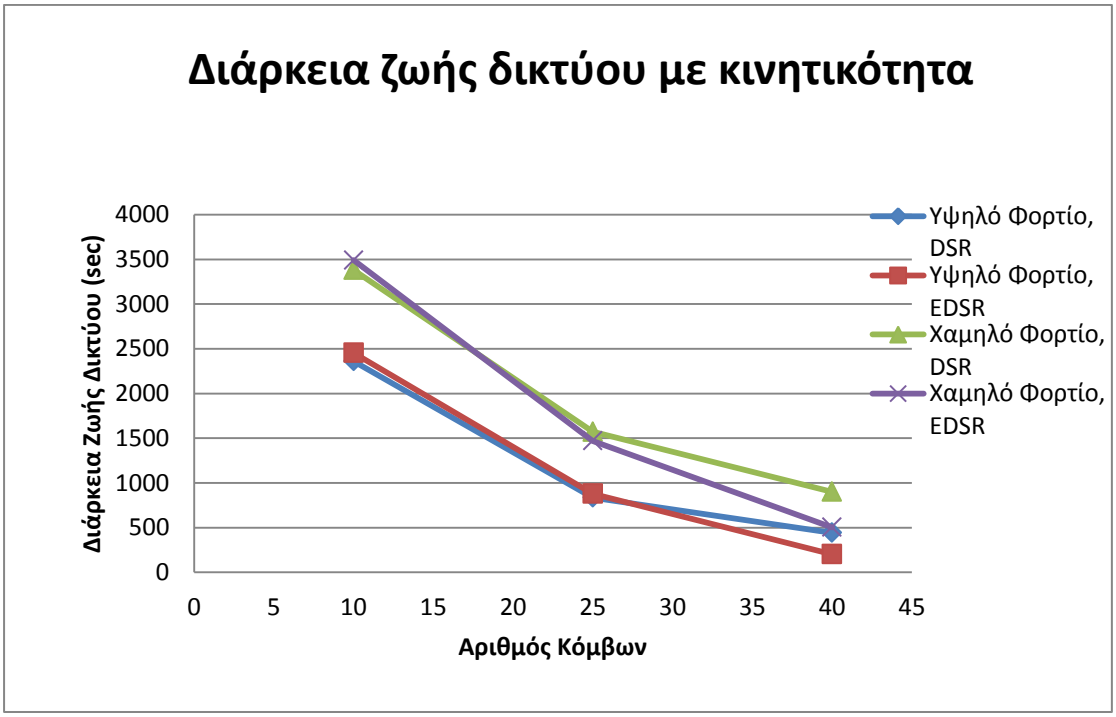
Αριθμός Κόμβων	Κινητικότητα	Φορτίο	Πρωτόκολλο	Διάρκεια Ζωής Δικτύου (sec)	Μέση Υπολειπόμενη Ενέργεια (J)	Μέση Καθυστέρηση (ms)
10	Όχι	Υψηλό	DSR	2014	10,2	21,5
			EDSR	**2304**	7,2	23,7
		Χαμηλό	DSR	4794	7,5	10,4
			EDSR	**4831**	6,9	10,5
	Ναι	Υψηλό	DSR	2361	8,2	162
			EDSR	**2454**	6,8	97
		Χαμηλό	DSR	3382	16,2	37
			EDSR	**3490**	14,3	22
25	Όχι	Υψηλό	DSR	729	14,7	130
			EDSR	**814**	8,1	141
		Χαμηλό	DSR	1673	11,5	80
			EDSR	**1891**	8,6	58
	Ναι	Υψηλό	DSR	836	5,7	852
			EDSR	**879**	8,2	18.000
		Χαμηλό	DSR	1571	9,8	91
			EDSR	1470	7,4	76
40	Όχι	Υψηλό	DSR	629	10,4	156
			EDSR	450	7,8	209
		Χαμηλό	DSR	1203	10,6	95
			EDSR	1033	10,3	86
	Ναι	Υψηλό	DSR	442	6,2	281
			EDSR	203	6,3	14.453
		Χαμηλό	DSR	900	9,8	152
			EDSR	503	9,1	691

Πίνακας 4.11 : Αποτελέσματα Γενικών Σεναρίων EDSR

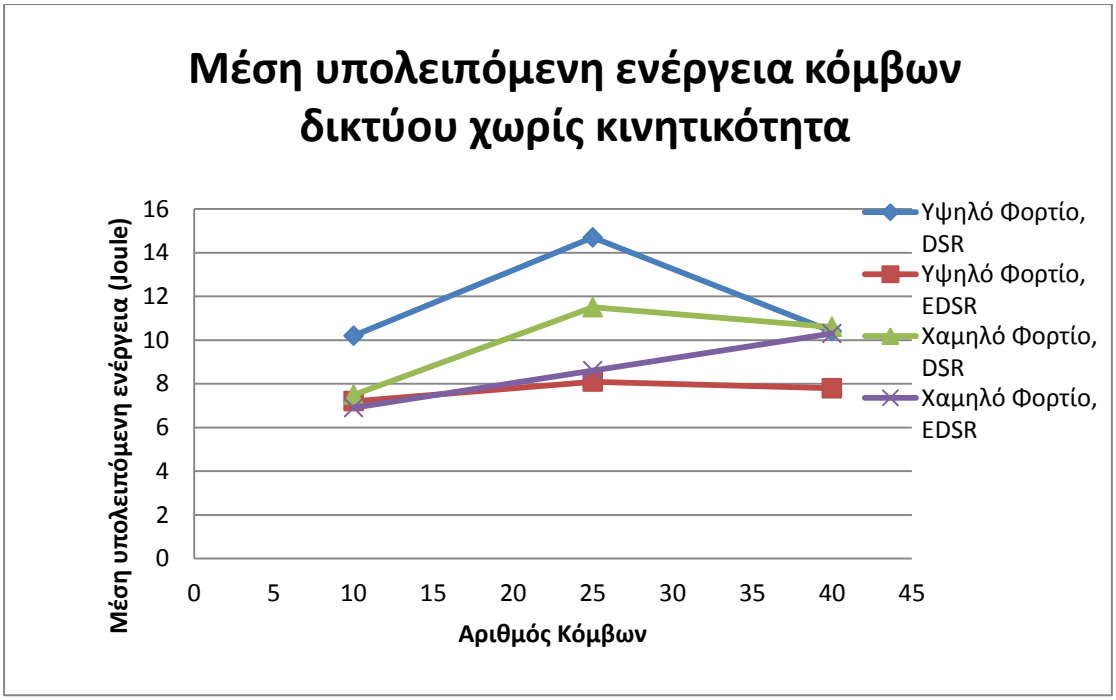
Το ζητούμενο του EDSR είναι η αύξηση της διάρκειας ζωής του δικτύου, οπότε στον πίνακα σημειώνονται με έντονη γραφή οι βελτιώσεις, προς διευκόλυνση του αναγνώστη. Μια καλύτερη εικόνα δίνεται από τα Σχήματα 4.20 – 4.25, όπου σχεδιάζονται τα τρία κριτήρια αξιολόγησης του δικτύου συναρτήσει του πλήθους των κόμβων για δίκτυα χωρίς κινητικότητα και με κινητικότητα.



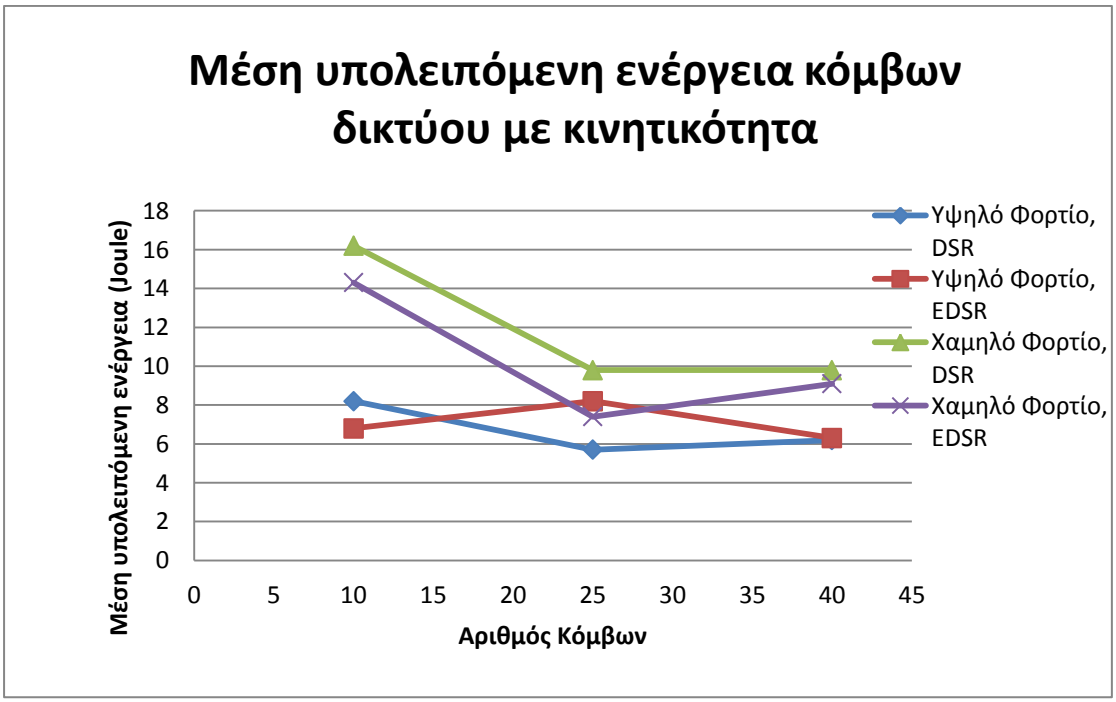
Σχήμα 4.20 : Διάρκεια ζωής δικτύου συναρτήσει πλήθους κόμβων στα Γενικά Σενάρια EDSR χωρίς κινητικότητα



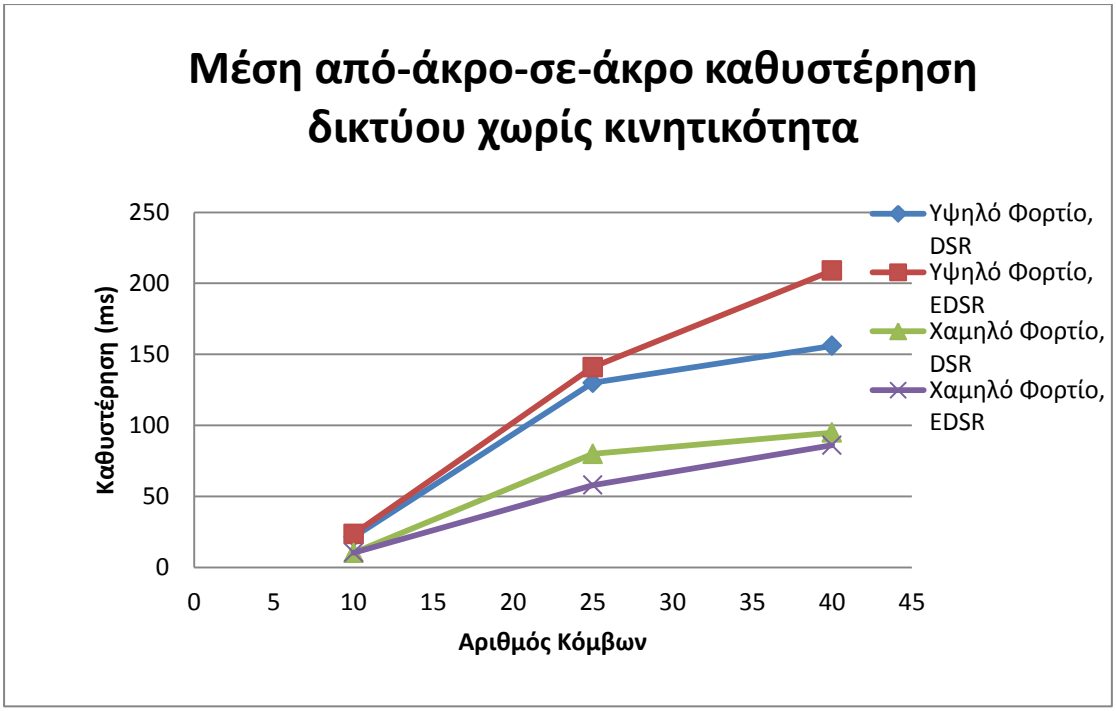
Σχήμα 4.21 : Διάρκεια ζωής δικτύου συναρτήσει πλήθους κόμβων στα Γενικά Σενάρια EDSR με κινητικότητα



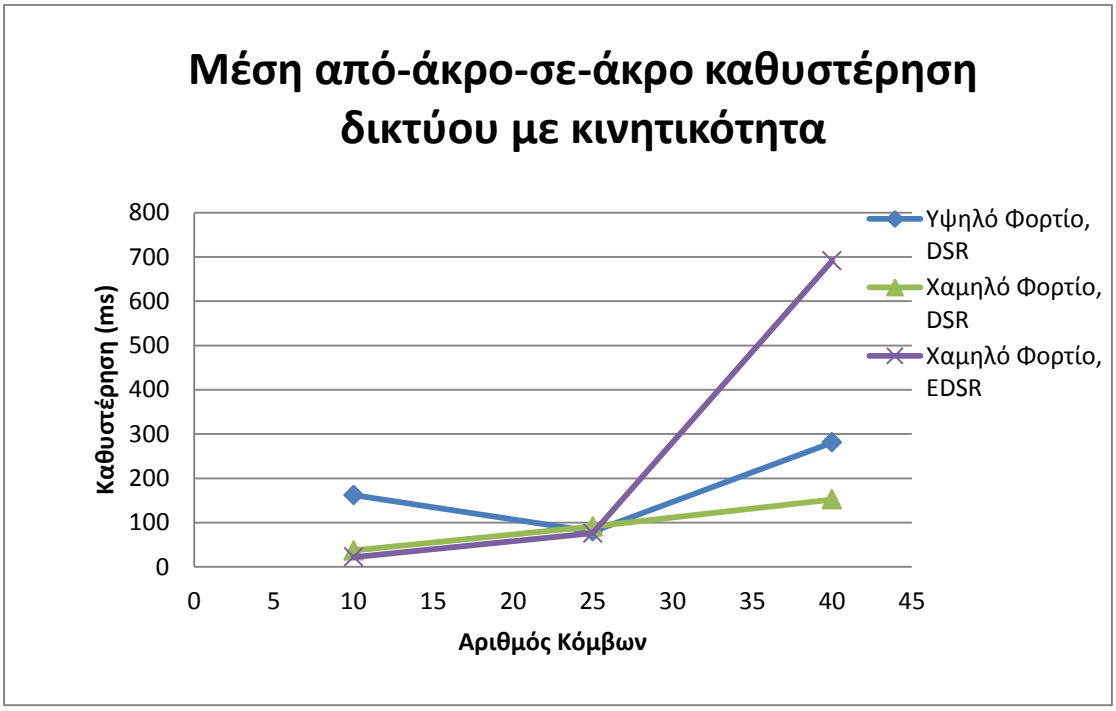
Σχήμα 4.22 : Μέση υπολειπόμενη ενέργεια κόμβων συναρτήσει πλήθους κόμβων στα Γενικά Σενάρια EDSR χωρίς κινητικότητα



Σχήμα 4.23 : Μέση υπολειπόμενη ενέργεια κόμβων συναρτήσει πλήθους κόμβων στα Γενικά Σενάρια EDSR με κινητικότητα



Σχήμα 4.24 : Μέση από-άκρο-σε-άκρο καθυστέρηση συναρτήσει πλήθους κόμβων στα Γενικά Σενάρια EDSR χωρίς κινητικότητα



Σχήμα 4.25 : Μέση από-άκρο-σε-άκρο καθυστέρηση συναρτήσει πλήθους κόμβων στα Γενικά Σενάρια EDSR με κινητικότητα

Αρχικά, παρατηρούμε ότι το EDSR αυξάνει τη *διάρκεια ζωής* στα δίκτυα με μικρό πλήθος κόμβων από 10 έως 20 με 25 κόμβους. Η βελτίωση αυτή φθάνει μέχρι και το

14%. Για μεγαλύτερα δίκτυα οι επιδόσεις χειροτερεύουν, κάτι που όμως έχει άμεση σχέση με την πυκνότητα των κόμβων στο δίκτυο. Εφόσον το μέγεθος του χώρου διατηρείται σταθερό, η επικάλυψη των ακτινών διάδοσης αυξάνεται μη γραμμικά, άρα περισσότεροι κόμβοι κρυφακούν (overhear) πακέτα καταναλώνοντας άσκοπα ενέργεια. Ειδικότερα, στα δίκτυα με κινητικότητα παρατηρείται μικρότερη βελτίωση για λίγους κόμβους και σημαντική επιδείνωση για πολλούς κόμβους.

Όσον αφορά τη *μέση υπολειπόμενη ενέργεια*, το EDSR προκαλεί μεγαλύτερη μέση κατανάλωση, κάτι που ήταν αναμενόμενο, λόγω της αυξημένης επιβάρυνσης δρομολόγησης που επιβάλλει. Ο μηχανισμός του μειώνει ομοιόμορφα την ενέργεια των κόμβων, ώστε να αυξάνεται η διάρκεια ζωής του δικτύου, επομένως η μείωση της ενέργειας δεν αποτελεί πρόβλημα, αλλά επιβεβαιώνει τη σωστή λειτουργία του μηχανισμού. Παρατηρείται μια εξαίρεση στην περίπτωση υψηλού φορτίου με 25 κινητούς κόμβους όπου η υπολειπόμενη ενέργεια του EDSR είναι μεγαλύτερη από αυτή του DSR, εξηγείται όμως από την πολύ μεγάλη καθυστέρηση του δικτύου (18 sec), που υποδεικνύει ότι δεν υπήρχε αποστολή δεδομένων για μεγάλο χρονικό διάστημα.

Τέλος, η *μέση καθυστέρηση* του δικτύου από άκρο σε άκρο είναι χαμηλότερη στο EDSR όταν το φορτίο είναι χαμηλό. Παρ' όλα αυτά, η κινητικότητα των κόμβων προκαλεί σημαντικά προβλήματα στο EDSR ειδικά σε μεγάλο πλήθος κόμβων, όπως φαίνεται από τις υπερβολικά μεγάλες καθυστερήσεις που παρατηρούνται (14 – 18 sec, δε συμπεριλήφθηκαν στο σχήμα) και ουσιαστικά παραλύουν το δίκτυο, παρεμποδίζοντας κάθε μετάδοση δεδομένων. Το φαινόμενο αυτό δείχνει ότι το EDSR είναι επιρρεπές σε υψηλές κινητικότητες, με πιθανότητα τελικής κατάρρευσης του δικτύου. Η συμπεριφορά αυτή ερμηνεύεται από το γεγονός ότι το μέσο μήκος διαδρομών στο EDSR είναι μεγαλύτερο του DSR, δηλαδή παρεμβάλλονται περισσότεροι ενδιάμεσοι κόμβοι, οπότε οι διαδρομές είναι ευκολότερο να «σπάσουν» όταν υπάρχει μεγάλη κινητικότητα. Παρ' όλα αυτά, το EDSR παραμένει ισχυρό σε χαμηλά φορτία καθώς μειώνει την καθυστέρηση αυξάνοντας παράλληλα τη διάρκεια ζωής του δικτύου.

Συμπερασματικά, το EDSR πετυχαίνει καλές επιδόσεις σε δίκτυα με χαμηλό έως μεσαίο αριθμό κόμβων (έως 25) αυξάνοντας σημαντικά τη διάρκεια ζωής του δικτύου. Ακόμα, υπό χαμηλό φορτίο ενδείκνυται για εφαρμογές ευαίσθητες ως προς το χρόνο, καθώς μειώνει την μέση καθυστέρηση στο δίκτυο. Όσον αφορά μεγαλύτερα δίκτυα, μπορεί να ανταπεξέλθει ενεργειακά σε δίκτυα χωρίς κινητικότητα, εμφανίζει όμως σημαντικά προβλήματα όταν υπάρχουν κινητοί κόμβοι.

Κεφάλαιο 5

Επίλογος

5.1 Συμπεράσματα

Στην ενότητα αυτή παρουσιάζονται συνοπτικά τα βασικά συμπεράσματα στα οποία οδήγησε η εκπόνησης της διπλωματικής εργασίας.

- Η βιωσιμότητα των δικτύων MANET εξαρτάται σημαντικά από το στρώμα δικτύου, δηλαδή το πρωτόκολλο δρομολόγησης. Το AODV παρουσιάζει εν γένει καλύτερες επιδόσεις από το DSR ως προς την κατανάλωση ενέργειας.
- Η ενεργειακή κατανάλωση εξαρτάται επίσης από την τεχνολογία των κατώτερων στρωμάτων. Η κατάσταση αδράνειας έχει το μεγαλύτερο μερίδιο στην κατανάλωση, καθώς οι κόμβοι βρίσκονται στην κατάσταση αυτή για μεγάλο χρονικό διάστημα (όταν δε λαμβάνουν ή μεταδίδουν πακέτα). Ακόμα, ο αδιάκριτος τρόπος λειτουργίας (promiscuous mode) οδηγεί σε αύξηση της ενεργειακής κατανάλωσης.
- Το πρωτόκολλο δρομολόγησης EDSR αυξάνει τη διάρκεια ζωής του δικτύου έως 14% σε δίκτυα με χαμηλό έως μεσαίο πλήθος κόμβων (έως 25 κόμβους). Ακόμα, υπό χαμηλό φορτίο ενδείκνυται για εφαρμογές ευαίσθητες στην καθυστέρηση, καθώς μειώνει την μέση καθυστέρηση στο δίκτυο. Σε μεγαλύτερα δίκτυα δε λειτουργεί ικανοποιητικά, ενώ είναι ασταθές σε υψηλές κινητικότητες.
- Η κινητικότητα στα MANET εισάγει μια τυχαιότητα που μπορεί να οδηγήσει σε αποτυχία του πρωτοκόλλου δρομολόγησης, ειδικά αν αυτό δε λειτουργεί με κριτήριο την ελάχιστη διαδρομή.
- Το EDSR προκαλεί μεγαλύτερη μέση κατανάλωση ενέργειας από το DSR, λόγω της αυξημένης επιβάρυνσης δρομολόγησης που επιβάλλει. Ο μηχανισμός του είναι σχεδιασμένος να μειώνει ομοιόμορφα την ενέργεια των κόμβων, ώστε να αυξάνεται η διάρκεια ζωής του δικτύου.
- Στο tradeoff μεταξύ της επιβάρυνσης δρομολόγησης (routing overhead) και των μη έγκυρων πληροφοριών δρομολόγησης, προκύπτει ότι η επιβάρυνση δρομολόγησης έχει μεγαλύτερη επίπτωση στην ενεργειακή κατανάλωση. Δεν μπορεί, όμως, ένα πρωτόκολλο να κρίνεται μόνο ενεργειακά, καθώς η διέλευση (throughput), το ποσοστό χαμένων πακέτων και η καθυστέρηση αποτελούν πάντα σημαντικά κριτήρια της αποδοτικότητάς του.

5.2 Μελλοντική Εργασία

Οι μελλοντικές επεκτάσεις της εργασίας χωρίζονται σε δύο κατηγορίες, αυτές που αφορούν τη μπαταρία και το στρώμα ζεύξης και αυτές που αφορούν το πρωτόκολλο δρομολόγησης EDSR.

- Το μοντέλο της μπαταρίας είναι γενικό και εύκολο να προσαρμοσθεί σε άλλες ασύρματες τεχνολογίες του OPNET για την μελέτη της ενεργειακής κατανάλωσης (ZigBee, WiMAX κτλ.).
- Είναι δυνατή η βελτίωση του μοντέλου απωλειών, αν κρίνεται αναγκαίο, ώστε να προσαρμοσθεί σε κάποια συγκεκριμένη ασύρματη κάρτα διεπαφής δικτύου με βάση το εμπειρικό μοντέλο απωλειών.
- Κρίνεται σκόπιμη η τροποποίηση του στρώματος ζεύξης ώστε να έχει την δυνατότητα επιλογής ή όχι του αδιάκριτου τρόπου λειτουργίας (promiscuous mode), οπότε θα είναι δυνατή η αξιολόγηση του ενεργειακού κόστους του.
- Η υλοποίηση μηχανισμού που ενσωματώνει την κατάσταση ύπνου (sleep) στο στρώμα ζεύξης των κόμβων μπορεί να οδηγήσει σε σημαντική μείωση της καταναλισκόμενης ενέργειας, καθώς θα μειώσει τον χρόνο αδράνειας των κόμβων.

Οι μελλοντικές επεκτάσεις που αφορούν το πρωτόκολλο δρομολόγησης είναι :

- Για τις παραμέτρους του πρωτοκόλλου EDSR που στις προσομοιώσεις προσαρμόζονται από τον χρήστη, μπορεί να υλοποιηθεί ένας προσαρμοστικός αλγόριθμος που θα μετρά στατιστικά του δικτύου που αφορούν το φορτίο ή την κινητικότητα και αυτόματα θα προσαρμόζει τις παραμέτρους αυτές (για παράδειγμα το Route Update Interval). Έτσι, το πρωτόκολλο θα είναι ευέλικτο και καλύτερα προσαρμοσμένο στο δυναμικό περιβάλλον των MANET.
- Η λήψη αποφάσεων για τη δρομολόγηση είναι δυνατόν να γίνει πιο σύνθετη, βασισμένη σε περισσότερα κριτήρια, όπως με ένα σταθμισμένο μέσο της μέγιστης ελάχιστης ενέργειας και της ελάχιστης διαδρομής ή με επιλογή των κριτηρίων υπό συνθήκες, ώστε να επιτευχθεί μεγαλύτερη αξιοπιστία, σταθερότητα και ενεργειακή απόδοση. Ακόμα, η έρευνα αυτή μπορεί να οδηγήσει σε υποστήριξη ποιότητας υπηρεσίας (Quality of Service – QoS) από το πρωτόκολλο, ώστε η χρήση του να επεκταθεί σε εφαρμογές πολυμέσων.
- Κρίνεται σκόπιμη η περαιτέρω μελέτη του πρωτοκόλλου σε δίκτυα μεγάλου πλήθους κόμβων, αλλά και σε δίκτυα υψηλής κινητικότητας, ώστε να εντοπιστούν οι αδυναμίες κλιμάκωσης και να βελτιωθούν οι μηχανισμοί και οι επιδόσεις του.
- Το πρωτόκολλο EDSR είναι δυνατόν να χρησιμοποιηθεί ως μέρος ευρύτερων μελετών, όπως μια ολοκληρωμένη διαστρωματική προσέγγιση ενεργειακά ενήμερων δικτύων MANET, με έλεγχο ισχύος μετάδοσης, διαχείριση καταστάσεων μπαταρίας και άλλες λειτουργίες στα κατώτερα στρώματα.

Βιβλιογραφία

[1] Wikipedia, the free encyclopedia. <http://en.wikipedia.org>.

[2] Tanenbaum, Andrew S. *Computer Networks*. Upper Saddle River, NJ: Prentice Hall PTR, 2003.

[3] Johnson, D., Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4, RFC 4728, Rice, UIUC, and Microsoft, February 2007.

[4] Toh, C.-K. Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks. *Communications Magazine, IEEE*. June 2001, Vol.39, Issue 6: 138 – 147.

[5] Buruhanudeen S., Othman Mo., Othman Ma., and Ali B. M. Existing MANET Routing Protocols and Metrics used Towards the Efficiency and Reliability – An Overview, in Telecommunications and Malaysia International Conference on Communications, 2007. ICT-MICC 2007. IEEE: 231 – 236.

[6] Gowrishankar S., Basavaraju T.G., and SubirKumarSarkar. Mobility Based Energy Analysis of Five Mobility Models in MANET Using Five Routing Protocols. *IIJCSNS International Journal of Computer Science and Network Security*. May 2010, Vol.10, No.5: 64 – 72.

[7] Αρσλάνογλου Γ. *Ανάπτυξη Λογισμικού Προσομοίωσης Ασύρματων Δικτύων Τύπου MANET – Ανάλυση Πρωτοκόλλων Δρομολόγησης*. Διπλωματική Εργασία. ΑΠΘ, 2007.

[8] Tseng Y.C., and Hsieh T.Y. Fully Power-Aware and Location-Aware Protocols for Wireless Multi-hop Ad Hoc Networks, in Computer Communications and Networks, Eleventh International Conference, 2002. Proceedings: 608 – 613.

[9] Ramakrishnan S., Thyagarajan T., Ram P.K., and Vinodh R. Design and Analysis of PADSR Protocol for Routing in MANETs, in IEEE Indicon 2005 Conference, Chennai, India, Dec. 2005 : 193 – 197.

[10] Didi, F.,Bouyedou, B., Feham, M., and Labiod, H. Mobility Impact on DSR-Based Power-Aware Ad hoc Routing Protocol, in New Technologies, Mobility and Security (NTMS ‘08), 2008 : 1 – 6.

[11] Garcia J.-E., Kallel A., Kyamakya K., Jobmann K., Cano J.-C., and Manzoni P. A Novel DSR-based Energy-efficient Routing Algorithm for Mobile Ad-hoc Networks, in Vehicular Technology Conference (VTC), Fall 2003, Vol.5: 2849 – 2854.

[12] De Rango F., Lonetti P., and Marano S. Energy-aware Metrics impact on Multipath DSR in MANETs Environment, in Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2008: 130 – 137.

[13] Cano J.-C., and Manzoni P. A Performance Comparison of Energy Consumption for Mobile Ad Hoc Network Routing Protocols, in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2000 : 57 – 63.

[14] Κανάτας Α., Κωνσταντίνου Φ., και Πάντος Γ. *Συστήματα Κινητών Επικοινωνιών*. Αθήνα : Παπασωτηρίου, 2008.

[15] Θεολόγου Μ. Ε. *Δίκτυα Κινητών και Προσωπικών Επικοινωνιών*. Θεσσαλονίκη : Εκδόσεις Τζιόλα, 2008.

[16] *IEEE Std 802.11-2007: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (2007 revision)*. IEEE-SA, 12 June 2007.

[17] Sarkar Kumar, T. G. Basavaraju, and C. Puttamadappa. *Ad Hoc Mobile Wireless Networks: Principles, Protocols, and Applications*. New York: Auerbach Publications, 2008.

[18] Feeney L.M., and Nilsson M. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment, in IEEE INFOCOM 2001, Vol 3 : 1548 – 1557.

[19] Feeney L.M. An Energy Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks. *Mobile Networks and Applications*. 2001, Vol.6 : 239-249.

[20] Boukerche, A. *Algorithms and Protocols for Wireless and Mobile Ad Hoc Networks*. Hoboken, NJ: Wiley, 2009

[21] Mueller S., Tsang R.P., Ghosal D. Multipath Routing in Mobile Ad Hoc Networks: Issues and Challenges, in *Performance Tools and Applications to Networked Systems (LNCS 2965)*. Berlin: Springer-Verlag, 2004: 209-234.

[22] Κατσιγιάννης Χ. *Αρχιτεκτονική αύξησης βιωσιμότητας MANET με βάση ενεργειακούς περιορισμούς*. Διδακτορική Διατριβή. ΕΜΠ, 2010.

[23] Γιαννακάκης Σ. *Μελέτη ανακάλυψης διαδρομής σε δίκτυα MANET, με ενεργειακά κριτήρια*. Διπλωματική εργασία. ΕΜΠ, 2010.

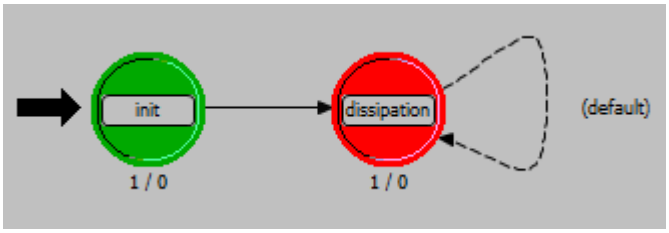
[24] Petr Jurcik, and Anis Koubaa. The IEEE 802.15.4 OPNET Simulation Model: Reference Guide v2.0. IPP-HURRAY Technical Report, HURRAY-TR-070509, May 2007.

Παράρτημα Α

Μοντέλο Μπαταρίας

A.1 Κώδικας διεργασίας akis_wlan_battery_process

Κώδικας εισόδου της κατάστασης init: `akis_wlan_battery_init();`
Κώδικας εισόδου της κατάστασης dissipation: `akis_wlan_battery_update();`



Διάγραμμα καταστάσεων της διεργασίας

Header Block

```
#include <string.h>

/* Codes for remote self interrupts     */
#define PACKET_TX_CODE 101
#define PACKET_RX_CODE 102

/* Special attribute values             */
#define milli 0.001

/* Structures       */
typedef struct {

       Objid         own_id;
       Objid         parent_id;
       double        power_supply;              // in Volt
       double        initial_energy;            // in Joule
       double        current_rx_mA;
       double        current_tx_mA;
       double        current_idle_mA;
       double        current_sleep_mA;

       double        power_tx;                  // in Watts
       double        power_rx;
       double        power_idle;

       Boolean       idle_mode;                 // idle mode consumption can be turned on/off
       int           idle_interval;             // depends on initial energy & accuracy level
       Boolean       idle_interval_change; // flag to indicate interval change, happens only once

       double        current_energy;            // in Joule
       double        failure_time;

} akis_wlan_battery_attributes;


typedef struct {

       Stathandle    remaining_energy;
       Stathandle    consumed_energy;

} akis_wlan_battery_statistics;


typedef struct {

       Stathandle    consumed_energy;
```

```
        Stathandle    remaining_energy;
        Stathandle    fail_nodes;
        Stathandle    average_remaining_energy;

} akis_wlan_global_battery_statistics;


/* Function prototypes */
static void akis_wlan_battery_init (void);
static void akis_wlan_battery_update (void);
static void battery_update_statistics(double,char[16]);
static int akis_wlan_num_nodes(void);

/* Global Variables */
double global_consumed_energy;
double global_remaining_energy;
Boolean global_first_node_to_init = OPC_TRUE;
int global_fail_nodes;
```

Function Block

```
static void akis_wlan_battery_init() {

   // The global variables are declared in the Header Block
   int num_nodes;

   FIN (akis_wlan_battery_init);   //stack tracing entry/exit points

   // get the ID of this module
   battery.own_id = op_id_self ();

   // get the ID of the node
   battery.parent_id = op_topo_parent (battery.own_id);

   // get the attributes
   op_ima_obj_attr_get (battery.own_id, "Power Supply", &battery.power_supply);
   op_ima_obj_attr_get (battery.own_id, "Initial Energy", &battery.initial_energy);

   op_ima_obj_attr_get (battery.own_id, "Current Rx", &battery.current_rx_mA);
   op_ima_obj_attr_get (battery.own_id, "Current Tx", &battery.current_tx_mA);
   op_ima_obj_attr_get (battery.own_id, "Current Idle", &battery.current_idle_mA);
   op_ima_obj_attr_get (battery.own_id, "Current Sleep", &battery.current_sleep_mA);

   op_ima_obj_attr_get (battery.own_id, "Idle Mode", &battery.idle_mode);
   op_ima_obj_attr_get (battery.own_id, "Idle Interval", &battery.idle_interval);

   op_ima_obj_attr_get (battery.own_id, "Battery Log", &battery_log);


   battery.current_energy = battery.initial_energy;
   manet_api_current_energy = battery.current_energy;   // manet API

   // register the statistics that will be maintained by this model
   statistics.remaining_energy = op_stat_reg ("Battery.Remaining Energy (Joule)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
   statistics.consumed_energy    = op_stat_reg ("Battery.Consumed Energy (Joule)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
   statisticsG.remaining_energy = op_stat_reg ("Battery.Remaining Energy (Joule)",
OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
   statisticsG.consumed_energy = op_stat_reg ("Battery.Consumed Energy (Joule)",
OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
   statisticsG.fail_nodes = op_stat_reg ("Battery.Failed Nodes", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
   statisticsG.average_remaining_energy = op_stat_reg ("Battery.Average Remaining Energy",
OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);

   // write the values to the local registered handles
   op_stat_write(statistics.remaining_energy,battery.current_energy);
   op_stat_write(statistics.consumed_energy,0.0);



   // Global variables //
```

```
   // global variables are initialized only once, by the first node to initialize
   if (global_first_node_to_init) {

      num_nodes = akis_wlan_num_nodes();
      //all nodes have the same initial energy
      global_remaining_energy = num_nodes*battery.initial_energy;
      global_consumed_energy = 0;
      global_fail_nodes = 0;

      // write the values to the global registered handles
      op_stat_write(statisticsG.consumed_energy, global_consumed_energy);
      op_stat_write(statisticsG.remaining_energy, global_remaining_energy);
      op_stat_write(statisticsG.fail_nodes, global_fail_nodes);

      // initialize failed_nodes file (overwrite mode)
      failed_nodes = fopen("C:\\opnet_logs\\failed_nodes.txt","w");
      fprintf(failed_nodes,"New Simulation - PC time = %lf\n\n",time(NULL));
      fclose(failed_nodes);

      global_first_node_to_init = OPC_FALSE;
   }

   // Idle Mode //
   // if idle mode is on, interrupt every idle_interval seconds in order to update
   // the idle consumed energy
   // else, no self interrupts are invoked
   // idle mode consumes all the time,
   // so Rx/Tx consumption must be the difference between idle & Rx/Tx energy
   // so we subtract idle current from Rx/Tx current
   if (battery.idle_mode == OPC_TRUE) {
      battery.idle_interval_change = OPC_TRUE;
      battery.current_tx_mA -= battery.current_idle_mA;
      battery.current_rx_mA -= battery.current_idle_mA;
      op_intrpt_schedule_self(op_sim_time() + battery.idle_interval, OPC_NIL);
   }

   // initializations and calculations
   battery.failure_time = -1;
   battery.power_tx = battery.current_tx_mA * milli * battery.power_supply;
   battery.power_rx = battery.current_rx_mA * milli * battery.power_supply;
   battery.power_idle = battery.current_idle_mA * milli * battery.power_supply;


   FOUT;
}

static void akis_wlan_battery_update() {

   Ici * iciptr;
   double pksize;
   double wlan_data_rate;
   //double packet_time;

   double consumed_energy;

   char node_name[16];

   FIN (akis_wlan_battery_update);

   //get the ID of this module
   battery.own_id = op_id_self ();

   //get the ID of the node
   battery.parent_id = op_topo_parent (battery.own_id);

   //get the name of the node
   op_ima_obj_attr_get (battery.parent_id, "name", &node_name);

   if (battery_log == OPC_TRUE)
      log = fopen("C:\\opnet_logs\\log_battery.txt","a"); //print
```

```
   // Remote interrupts indicate Rx/Tx
   if (op_intrpt_type() == OPC_INTRPT_REMOTE) {

      // get the ICI information associated to the remote interrupt
      iciptr = op_intrpt_ici();
      op_ici_attr_get(iciptr, "Packet Size", &pksize);
      op_ici_attr_get(iciptr, "Data Rate", &wlan_data_rate);
      op_ici_destroy(iciptr);

      // This is either Rx or Tx time of the packet
      switch (op_intrpt_code()) {

         case PACKET_TX_CODE : {
            // compute the consumed energy when transmitting a packet
            consumed_energy = pksize * battery.power_tx/wlan_data_rate;

            if (battery_log == OPC_TRUE)
               {
               fprintf(log,"Tx - %s - pksize = %lf\t",node_name,pksize);
               fprintf(log,"Consumed energy due to Transmission = %lf\t\n",consumed_energy);
               }

            break;

         }

         case PACKET_RX_CODE : {
            // compute the consumed energy when receiving a packet
            consumed_energy = pksize * battery.power_rx/wlan_data_rate;

            if (battery_log == OPC_TRUE)
               {
               fprintf(log,"Rx - %s - pksize = %lf\t",node_name,pksize);
               fprintf(log,"Consumed energy due to Reception = %lf\t\n",consumed_energy);
               }

            break;
         }

         default :   {
               if (battery_log == OPC_TRUE)
               fprintf(log,"\n\nERROR error in Remote Interrupt Code!!!! (this is the default
case)\n\n");
            break;
         }
      }

      battery_update_statistics(consumed_energy,node_name);

   }

   // Self interrupts indicate Idle consumption update
   else if (op_intrpt_type() == OPC_INTRPT_SELF) {

      // update the consumed energy with the one of in idle state
      consumed_energy = battery.idle_interval * battery.power_idle;

      if (battery_log == OPC_TRUE)
         fprintf(log,"Idle Consumption = %lf\n",consumed_energy);

      battery_update_statistics(consumed_energy,node_name);

      // change the interval if battery is close to its end (once it's below 10%)
      if ((battery.current_energy < 0.1*battery.initial_energy) && (battery.idle_interval_change
== OPC_TRUE)) {
            while (battery.idle_interval/2 >= 1)
                  battery.idle_interval /= 2; // interval must be at least 1 sec
            battery.idle_interval_change = OPC_FALSE;

            failed_nodes = fopen("C:\\opnet_logs\\failed_nodes.txt","a");
            fprintf(failed_nodes,"Node \"%s\" changed its idle_interval to:\t
%d\n",node_name,battery.idle_interval);
```

```
            fclose(failed_nodes);

      }

      // interrupt every IDLE_INTERVAL seconds in order to update the idle consumed energy
      op_intrpt_schedule_self(op_sim_time() + battery.idle_interval, OPC_NIL);
   }

   if (battery_log == OPC_TRUE)
      fclose(log);

   FOUT;
}

static void battery_update_statistics(double last_consumed_energy,char nodename[16]) {

   double residue;
   int num_nodes;

   FIN (battery_update_statistics(last_consumed_energy,nodename));

   // check if node has just failed
   if ((battery.current_energy - last_consumed_energy) <= 0) {

      // extra energy "consumed" after node's failure, used below
      residue = last_consumed_energy - battery.current_energy;
      // cannot consume more energy than available
      last_consumed_energy = battery.current_energy;
      battery.current_energy = 0;
      manet_api_current_energy = battery.current_energy;   // manet API

      op_ima_obj_attr_set (battery.parent_id, "condition", OPC_BOOLINT_DISABLED);   //fail node

      // The real node failure time is not used for energy updates,
      //so as not to mess the global battery statistics
      battery.failure_time = op_sim_time() - (residue/(battery.current_idle_mA * milli *
battery.power_supply));   // real node failure time
      global_fail_nodes++;
      op_stat_write_t(statisticsG.fail_nodes, global_fail_nodes, battery.failure_time);

      // update log of nodes that fail
      failed_nodes = fopen("C:\\opnet_logs\\failed_nodes.txt","a");
      fprintf(failed_nodes,"Node \"%s\" failed at time :\t
%lf\n",nodename,battery.failure_time);
      fprintf(failed_nodes,"Current Time = %lf,\tResidue = %lf,\tFailed_Nodes =
%d\n\n",op_sim_time(),residue,global_fail_nodes);
      fclose(failed_nodes);
   }
   // else, node is still up
   else {
      battery.current_energy -= last_consumed_energy;
      manet_api_current_energy = battery.current_energy;   // manet
   }

   // Update the local statistics
   op_stat_write(statistics.remaining_energy, battery.current_energy);
   op_stat_write(statistics.consumed_energy, battery.initial_energy-battery.current_energy);

   // update the global statistics
   global_consumed_energy += last_consumed_energy;
   global_remaining_energy -= last_consumed_energy;
   op_stat_write(statisticsG.consumed_energy, global_consumed_energy);
   op_stat_write(statisticsG.remaining_energy, global_remaining_energy);

   num_nodes = akis_wlan_num_nodes();
   op_stat_write(statisticsG.average_remaining_energy, global_remaining_energy/num_nodes);


   FOUT;
}
```

```
static int akis_wlan_num_nodes(void) {

	int		num_objects = 0;
	int		num_nodes = 0;
	int		i;

	char		node_model[40], temp_node_model[40];
	Objid		obj_id;

	FIN (akis_wlan_num_nodes (void));

//if Nodes are Mobile, then the number of Mobile Objects is equal to the number of Mobile Nodes
//if Nodes are Fixed, then the number of Fixed Objects may be unequal to the number of Fixed
Nodes
//because models like Rx Config and Mobility Config also are considered as Fixed Objects.
// We have to find how many Fixed Nodes exist.

	num_objects = op_topo_object_count (OPC_OBJTYPE_NODE_MOB);

	if (num_objects > 0) {

		num_nodes = num_objects;
	}

	else {

		op_ima_obj_attr_get (battery.parent_id, "model", &node_model);

		num_objects = op_topo_object_count (OPC_OBJTYPE_NODE_FIX);

		for (i = 0; i < num_objects; i++) {
			obj_id = op_topo_object (OPC_OBJTYPE_NODE_FIX, i);
			op_ima_obj_attr_get (obj_id, "model", &temp_node_model);
			if (strcmp (node_model, temp_node_model) == 0)
				num_nodes++;
		}
	}

	FRET (num_nodes);
}
```

A.2 Κώδικας διεργασίας akis_wlan_dispatch

Κώδικας εισόδου της κατάστασης spawn:

```
/* Find out whether the surrounding WLAN MAC module         */
/* supports Hybrid Coordination Function (HCF),                    */
/* specified in the IEEE 802.11e standard. Access the */
/* WLAN configuration attribute.                                   */
op_ima_obj_attr_get (op_id_self (), "Wireless LAN Parameters", &comp_attr_objid);
comp_attr_row_objid = op_topo_child (comp_attr_objid, OPC_OBJTYPE_GENERIC, 0);

/* Read the value of the corresponding attribute under      */
/* HCF Parameters.                                                                 */
op_ima_obj_attr_get (comp_attr_row_objid, "HCF Parameters", &comp_attr_objid);
comp_attr_row_objid = op_topo_child (comp_attr_objid, OPC_OBJTYPE_GENERIC, 0);
op_ima_obj_attr_get (comp_attr_row_objid, "Status", &hcf_support_int);

/* Create the appropriate MAC process model.                */
mac_prohandle = (hcf_support_int == OPC_BOOLINT_ENABLED) ?
                                op_pro_create ("wlan_mac_hcf", OPC_NIL) :
                                op_pro_create ("akis_wlan_mac", OPC_NIL); //dcf
/* akis: MANET are DCF by default - no need to change HCF operation */

/* Make the child process the recipient of the              */
/* interrupts of the module.                                                  */
op_intrpt_type_register (OPC_INTRPT_STRM,   mac_prohandle);
op_intrpt_type_register (OPC_INTRPT_STAT,   mac_prohandle);
op_intrpt_type_register (OPC_INTRPT_REMOTE, mac_prohandle);

/* Spawn the MAC child process.                                               */
op_pro_invoke (mac_prohandle, OPC_NIL);
```

A.3 Κώδικας διεργασίας akis_wlan_mac

Η μορφή του ICI που στέλνει η `akis_wlan_mac` στην `akis_wlan_battery_process` είναι η ακόλουθη:

Attribute Name	Type	Default Value	Description
Packet Size	double	0	in bits
Data Rate	double	11000000	in bps

akis_wlan_battery_ici_format

Header Block

Στο Header Block της διεργασίας έγιναν οι ακόλουθες προσθήκες :

```
/* akis: Codes for remote self interrupts      */
/* The same values should be in the akis_wlan_battery_process Module for remote process */

#define PACKET_TX_CODE 101
#define PACKET_RX_CODE 102

// akis: battery update functions
static void akis_wlan_battery_update_tx (double pksize, double data_rate);
static void akis_wlan_battery_update_rx (double pksize, double data_rate);
```

Function Block

Στο Function Block της διεργασίας έγιναν οι ακόλουθες προσθήκες :

```
// Sends information about Transmission to the battery module
// pksize [bits], data_rate [bps]
static void akis_wlan_battery_update_tx (double pksize, double data_rate) {

	Ici * iciptr;

	FIN (akis_wlan_battery_update_tx(pksize,data_rate));

	iciptr = op_ici_create ("akis_wlan_battery_ici_format");    //create ICI
	op_ici_attr_set (iciptr, "Packet Size", pksize);
	op_ici_attr_set (iciptr, "Data Rate", data_rate);
	op_ici_install (iciptr);   // ICI is automatically associated with outgoing interrupts
	                           // scheduled by the invoking process.
	op_intrpt_schedule_remote (op_sim_time(), PACKET_TX_CODE, akis_battery);
	op_ici_install (OPC_NIL);  // An ICI remains installed until another takes its place.

	FOUT;
}



// Sends information about Reception to the battery module
// pksize [bits], data_rate [bps]
static void akis_wlan_battery_update_rx (double pksize, double data_rate) {

	Ici * iciptr;

	FIN (akis_wlan_battery_update_rx(pksize,data_rate));


	iciptr = op_ici_create ("akis_wlan_battery_ici_format");
	op_ici_attr_set (iciptr, "Packet Size", pksize);
	op_ici_attr_set (iciptr, "Data Rate", data_rate);
	op_ici_install (iciptr);
	op_intrpt_schedule_remote (op_sim_time(), PACKET_RX_CODE, akis_battery);
	op_ici_install (OPC_NIL);

	FOUT;
}
```

Στη συνάρτηση `wlan_mac_sv_init` προστέθηκε η γραμμή:

```
/* akis:get the battery process object ID */
akis_battery = op_id_from_name (my_node_objid, OPC_OBJTYPE_PROC, "Battery");
```

Στη συνάρτηση `wlan_prepare_frame_to_send` προστέθηκε μετά από κάθε αποστολή πλαισίου δεδομένων η γραμμή:

```
akis_wlan_battery_update_tx (total_pk_size, tx_data_rate);
```

Στη συνάρτηση `wlan_physical_layer_data_arrival` προστέθηκε μετά τη λήψη πλαισίου δεδομένων η γραμμή:

```
akis_wlan_battery_update_rx (rcvd_pk_size, rcvd_frame_drate);
```

Παράρτημα Β

Μοντέλο Πρωτοκόλλου EDSR

Το μοντέλο του πρωτοκόλλου EDSR υλοποιήθηκε με την τροποποίηση όλων των αρχείων που σχετίζονται με το DSR. Λόγω του μεγάλου μεγέθους των αρχείων και της διασποράς των τροποποιήσεων που έγιναν σε αυτά, δε θα παρατεθεί εδώ όλος ο κώδικας του μοντέλου. Αντ' αυτού, θα γίνει μια αναφορά με τη μορφή πινάκων στις τροποποιήσεις αυτές και θα παρατεθεί ο κώδικας από κάποιες συναρτήσεις ενδεικτικές για τη λειτουργία του πρωτοκόλλου.

Όλες οι τροποποιήσεις που έγιναν στα αρχεία συνοδεύονται από σχόλια με τη λέξη 'akis', ώστε είναι πλήρως κατανοητή η επέκταση του πρωτοκόλλου DSR. Ακόμα, καθίσταται δυνατή η περαιτέρω επέκταση του μοντέλου μελλοντικά.

Στη συνέχεια, παρουσιάζεται η υλοποίηση του πρωτοκόλλου σε τρεις ενότητες : Αρχεία επικεφαλίδων, εξωτερικά αρχεία, διεργασίες.

Β.1 Αρχεία επικεφαλίδων

Τα αρχεία επικεφαλίδων που χρησιμοποιούνται από το μοντέλο είναι τα εξής :

1. akis_dsr.h
2. akis_dsr_pkt_support.h
3. akis_dsr_ptypes.h

Για κάθε αρχείο παρουσιάζεται ο κώδικας των δομών που τροποποιήθηκαν.

akis_dsr.h

```
/*******************************************/
/*************  ROUTE CACHE   ***************/
/*******************************************/

/* The route cache table stores all routes     */
/* learned by a particular node in a hash      */
/* table indexed by the destination IP         */
/* address string as its key. Each             */
/* destination index has a list of routes      */
typedef struct
   {
   PrgT_String_Hash_Table* route_cache_table;
   int                     current_cache_size;
   int                     max_cache_size;
   double                  path_expiry_time;
   double                  route_activity_time;       // akis
   double                  max_energy;                // akis
   DsrT_Stathandles*       dsr_stat_ptr;
   } DsrT_Route_Cache;


/* Array of hops along a single path. Each     */
/* path is associated with a timeout value     */
/* to delete the entry if not used within      */
/* a certain period of time                    */
typedef struct
   {
   List*            path_hops_lptr;
   Boolean          first_hop_external;
```

```
   Boolean          last_hop_external;
   int              num_hops;                  // this was built-in
   double           min_energy;                // akis : minimum node energy along the path
   double           installed_time;
   double           last_access_time;
   } DsrT_Path_Info;
```

akis_dsr_pkt_support.h

```
/* Route Request Option   */
typedef struct
   {
   Boolean          route_update;      // akis : flag
   double           min_energy;        // akis : min energy of the route so far
   long int         identification;
   InetT_Address    target_address;
   List*            route_lptr;
   } DsrT_Route_Request_Option;

/* Route Reply Option   */
typedef struct
   {
   Boolean          route_update;       // akis : flag
   double           min_energy          // akis : the minimum node energy of the path
   Boolean          last_hop_external;
   List*            route_lptr;
   } DsrT_Route_Reply_Option;

/* Acknowledgement Option   */
typedef struct
   {
   long int         identification;
   InetT_Address    ack_source_address;
   InetT_Address    ack_dest_address;
   double           min_energy;         // akis
   } DsrT_Acknowledgement;
```

akis_dsr_ptypes.h

Στο αρχείο αυτό απλώς αναφέρονται τα πρωτότυπα των συναρτήσεων που χρησιμοποιούνται από τη διεργασία *akis_dsr_rte*, οπότε δε γίνεται περαιτέρω αναφορά, καθώς οι νέες και οι τροποποιημένες συναρτήσεις θα παρουσιαστούν αναλυτικά σε πίνακες στη συνέχεια.

B.2 Εξωτερικά αρχεία

Τα εξωτερικά αρχεία που χρησιμοποιούνται από το μοντέλο είναι τα εξής :

1. akis_dsr_ route_cache
2. akis_dsr_extras
3. akis_dsr_ pkt_support
4. akis_dsr_route_discovery
5. akis_dsr_send_buffer
6. akis_dsr_maintenance_buffer
7. akis_dsr_support
8. akis_dsr_notif_log_support

Σε όλα τα αρχεία οι συναρτήσεις μετονομάστηκαν από *dsr_** σε *edsr_**. Αυτό έγινε για να μπορεί ο ίδιος κόμβος να χρησιμοποιήσει κατ' επιλογήν και τα δύο πρωτόκολλα DSR και EDSR, χωρίς να υπάρχουν διενέξεις μεταξύ των αρχείων που χρησιμοποιεί το καθένα.

Επιπλέον, στα αρχεία 1 – 3 προστέθηκαν νέες συναρτήσεις και έγιναν τροποποιήσεις στις ήδη υπάρχουσες. Οι συναρτήσεις χωρίζονται στις external, που καλούνται και από διεργασίες εκτός του ίδιου αρχείου (εν προκειμένω από τη διεργασία *akis_dsr_rte*), και στις internal, που καλούνται μόνο από συναρτήσεις εντός του ίδιου αρχείου. Αν η συνάρτηση δημιουργήθηκε εκ του μηδενός για το μοντέλο EDSR, τότε γράφεται σε παρένθεση η λέξη «Νέα». Οι αλλαγές αυτές φαίνονται στους παρακάτω πίνακες. Στη συνέχεια, παρατίθεται ο πηγαίος κώδικας ορισμένων ενδεικτικών συναρτήσεων, είτε νέων είτε σημαντικά τροποποιημένων.

Συνάρτηση	**Περιγραφή αλλαγής/λειτουργίας**
External	
edsr_route_cache_create	Προσθήκη πεδίου route_activity_time
edsr_route_cache_entry_add	Διαγραφή της edsr_route_cache_all_expired_routes_delete και αλλαγές που αφορούν την αλληλεξάρτηση με άλλες συναρτήσεις
edsr_route_cache_entry_access	Διαγραφή της edsr_route_cache_all_expired_routes_delete και επιλογή της καλύτερης διαδρομής που είναι ενεργή. Αν όλες είναι ανενεργές, τότε επιλογή της καλύτερης από αυτές
edsr_route_cache_all_routes_to_dest_access	Διαγραφή της edsr_route_cache_all_expired_routes_delete
edsr_route_cache_all_routes_to_dest_need_update (Νέα)	Επιστρέφει OPC_TRUE αν υπάρχει ενεργή διαδρομή προς τον εν λόγω προορισμό, οπότε όλες οι διαδρομές πρέπει να ανανεωθούν
edsr_route_cache_all_routes_to_dest_need_delete (Νέα)	Επιστρέφει OPC_TRUE αν υπάρχει ληγμένη διαδρομή προς τον εν λόγω προορισμό, οπότε όλες οι διαδρομές πρέπει να διαγραφούν
edsr_route_cache_all_routes_to_dest_delete (Νέα)	Διαγράφει όλες τις διαδρομές προς έναν προορισμό
edsr_route_cache_neighbouring_nodes (Νέα)	Επιστρέφει OPC_TRUE αν δύο κόμβοι είναι γειτονικοί
edsr_route_cache_neighbours_longer_routes_delete (Νέα)	Διαγράφει όλες τις διαδρομές ανάμεσα σε δύο γειτονικούς κόμβους εκτός από την απευθείας ζεύξη
edsr_route_cache_overlapping_routes_delete (Νέα)	Διαγράφει κάθε διαδρομή που περιέχει μέσα της μια άλλη συντομότερη διαδρομή, δηλαδή τις επικαλυπτόμενες διαδρομές για μη γειτονικούς κόμβους
edsr_route_cache_print_to_string (Νέα)	Εκτυπώνει τη Route Cache σε αρχείο, με την προσθήκη των Last Access Time και Min Energy
Internal	
edsr_route_cache_entry_create	Συμπεριλαμβάνεται το νέο πεδίο min_energy
edsr_route_cache_route_exists_update_cache	Αν η διαδρομή ήδη υπάρχει, τότε ανανεώνεται η τιμή του πεδίου min_energy
edsr_route_cache_insert_route_sorted	Αλλαγή του κριτηρίου ταξινόμησης διαδρομών στη μέγιστη ελάχιστη ενέργεια
edsr_route_cache_insert_route_priority	Αλλαγή που αφορά το κριτήριο min_energy
edsr_route_cache_entry_mem_alloc	Συμπεριλαμβάνεται το νέο πεδίο min_energy

Τροποποιήσεις στο αρχείο **akis_dsr_route_cache**

Συνάρτηση	Περιγραφή αλλαγής/λειτουργίας
External	
edsr_extras_manet_rte_to_cpu_pkt_send_schedule_with_jitter	Εισαγωγή jitter ως παραμέτρου
Internal	
edsr_extras_manet_rte_invoke_data_mem_alloc	Καμία αλλαγή, αλλά αναγκαία για την external συνάρτηση

Τροποποιήσεις στο αρχείο **akis_dsr_extras**

Συνάρτηση	Περιγραφή αλλαγής/λειτουργίας
External	
edsr_pkt_support_route_request_hop_insert_energy_update	Σε RREQ εισάγει το νέο βήμα και ανανεώνει τη μπαταρία, σε RU-REQ απλά ανανεώνει τη μπαταρία (αν χρειάζεται)
edsr_pkt_support_is_route_update (Νέα)	Επιστρέφει OPC_TRUE αν το πακέτο RREQ/RREP είναι RU-REQ/RU-REP αντίστοιχα
edsr_pkt_support_route_request_tlv_create	Συμπεριλαμβάνονται τα νέα πεδία route_update και min_energy, καθώς και η διαδρομή σε περίπτωση RU-REQ
edsr_pkt_support_route_reply_tlv_create	Συμπεριλαμβάνονται τα νέα πεδία route_update και min_energy
edsr_pkt_support_acknowledgement_tlv_create	Συμπεριλαμβάνονται το νέο πεδίο min_energy
Internal	
edsr_pkt_support_route_request_mem_copy	Συμπεριλαμβάνονται τα νέα πεδία route_update και min_energy
edsr_pkt_support_route_reply_mem_copy	Συμπεριλαμβάνονται τα νέα πεδία route_update και min_energy
edsr_pkt_support_acknowledgement_mem_copy	Συμπεριλαμβάνονται το νέο πεδίο min_energy

Τροποποιήσεις στο αρχείο **akis_dsr_pkt_support**

```
Boolean edsr_route_cache_all_routes_to_dest_need_update (DsrT_Route_Cache* route_cache_ptr,
List* dest_routes_lptr)
   {
   int                     num_paths, count;
   DsrT_Path_Info*         path_ptr;
   double                  current_time;
   /** checks all routes' to a specific destination activity (last access time)
       supposing 1 or more paths are active, they all need update, so it returns true
       used @ generate_route_update_requests **/

   FIN (edsr_route_cache_all_routes_to_dest_need_update (<args>));

   current_time = op_sim_time();
   num_paths = op_prg_list_size (dest_routes_lptr);

   // for all paths to this destination
   for (count = 0; count < num_paths; count++)
      {
      /* Get each path */
      path_ptr = (DsrT_Path_Info*) op_prg_list_access (dest_routes_lptr, count);

      if ((current_time - path_ptr->last_access_time) < route_cache_ptr->route_activity_time)
       // (t - LAT) < route_activity_time
         {
         FRET (OPC_TRUE);   // if even one path is used, then we update all of them
         }
      }
   FRET (OPC_FALSE);
   }

void edsr_route_cache_all_routes_to_dest_delete (DsrT_Route_Cache* route_cache_ptr, List*
dest_routes_lptr, char* dest_key_str)
   {
```

```
   double           current_time;
   double           update_time;
   int              num_paths;
   int              size = 0;
   DsrT_Path_Info*  path_ptr = OPC_NIL;

   /** Deletes all routes to a specific destination in the route cache that have expired   **/

   FIN (edsr_route_cache_all_routes_to_dest_delete (<args>));

   current_time = op_sim_time();
   /* Get the number of paths to this destination   */
   num_paths = op_prg_list_size (dest_routes_lptr);

   // num_paths decrements every loop
   while (size < num_paths)
      {
      /* Get each path to that destination   */
      path_ptr = (DsrT_Path_Info*) op_prg_list_access (dest_routes_lptr, size);

      /* Delete this entry   */
      path_ptr = (DsrT_Path_Info*) op_prg_list_remove (dest_routes_lptr, size);
      route_cache_ptr->current_cache_size -= 1;

      /* Update the route cache size statistic at the   */
      /* time this route was to be deleted         */
      update_time = path_ptr->installed_time + route_cache_ptr->path_expiry_time;
      op_stat_write_t (route_cache_ptr->dsr_stat_ptr->route_cache_size_shandle, (double)
route_cache_ptr->current_cache_size, update_time);

      edsr_route_cache_entry_mem_free (path_ptr);
      num_paths--;
      }

   /* If there are no routes to this destination   */
   /* free its memory                         */
   if (op_prg_list_size (dest_routes_lptr) == 0)
      {
      dest_routes_lptr = (List*) prg_string_hash_table_item_remove (route_cache_ptr-
>route_cache_table, dest_key_str);
      op_prg_list_free (dest_routes_lptr);
      op_prg_mem_free (dest_routes_lptr);
      }

   FOUT;
   }

void edsr_pkt_support_route_request_hop_insert_energy_update (Packet* ip_pkptr, InetT_Address
node_address, double node_energy, Boolean route_update, Boolean is_destination) // akis2
   {
   List*                        tlv_options_lptr = OPC_NIL;
   Packet*                      dsr_pkptr = OPC_NIL;
   DsrT_Packet_Option*          dsr_tlv_ptr = OPC_NIL;
   DsrT_Route_Request_Option*   route_request_ptr = OPC_NIL;
   int                          num_hops, count, num_options;
   int                          total_length = 0;
   int                          address_length;
   InetT_Address*               copy_address_ptr = OPC_NIL;
   OpT_Packet_Size              dsr_pkt_size;

   /** Inserts the hop address into the list of hops   **/
   /** traversed by this route request         **/
   FIN (edsr_pkt_support_route_request_hop_insert_energy_update (<args>));

   /* Get the DSR packet from the IP datagram   */
   op_pk_nfd_get (ip_pkptr, "data", &dsr_pkptr);

   /* Get the list of options   */
   op_pk_nfd_get (dsr_pkptr, "Options", &tlv_options_lptr);

   /* Get the number of options   */
   num_options = op_prg_list_size (tlv_options_lptr);
```

```
   for (count = 0; count < num_options; count++)
      {
      dsr_tlv_ptr = (DsrT_Packet_Option*) op_prg_list_access (tlv_options_lptr, count);

      if (dsr_tlv_ptr->option_type == DSRC_ROUTE_REQUEST)
         break;
      }

   /* Get a handle to route request structure   */
   route_request_ptr = (DsrT_Route_Request_Option*) dsr_tlv_ptr->dsr_option_ptr;
   num_hops = op_prg_list_size (route_request_ptr->route_lptr);   // get num_hops now

   /* akis : update the energy value if this node has less energy */
   // not for the destination, as the min_energy field concerns
   // only intermediate nodes of the path
   if (is_destination == OPC_FALSE)
      {
      if (node_energy < route_request_ptr->min_energy)
         route_request_ptr->min_energy = node_energy;
      }

   // akis2: the hop is inserted only if it's NOT route update request
   // the size of the options field is also re-determined
   // if it IS a route update, the size is fixed, as the whole route is known from start!
   if (route_update == OPC_FALSE)
      {
      copy_address_ptr = inet_address_create_dynamic (node_address);
      /* Set the node address in the hop list   */
      op_prg_list_insert (route_request_ptr->route_lptr, copy_address_ptr, OPC_LISTPOS_TAIL);

      /* Determine the size of the options field   */
      address_length = (inet_address_family_get (copy_address_ptr) == InetC_Addr_Family_v4 ?
IP_V4_ADDRESS_LENGTH : IP_V6_ADDRESS_LENGTH);
       // update num_hops, in case we added a node to the list
      num_hops = op_prg_list_size (route_request_ptr->route_lptr);
      dsr_tlv_ptr->option_length = DSR_HEADER_OPTIONS + (num_hops + 1)*address_length + 64 + 1;
      }

   /* Calculate the length of the options field   */
   for (count = 0; count < num_options; count++)
      {
      dsr_tlv_ptr = (DsrT_Packet_Option*) op_prg_list_access (tlv_options_lptr, count);
      total_length += dsr_tlv_ptr->option_length;
      }


   /* Set the options in the packet   */
   op_pk_fd_set (dsr_pkptr, fd_options_index, OPC_FIELD_TYPE_STRUCT, tlv_options_lptr,
total_length,   edsr_pkt_support_tlv_options_mem_copy, edsr_pkt_support_tlv_options_mem_free,
sizeof (List));

   /* Get new dsr packet size             */
   dsr_pkt_size = op_pk_total_size_get (dsr_pkptr);

   /* Set the DSR packet in the IP datagram      */
   op_pk_nfd_set (ip_pkptr, "data", dsr_pkptr);

   /* Set the bulk size of the IP packet to model      */
   /* the space occupied by the encapsulated data.    */
   op_pk_bulk_size_set (ip_pkptr, dsr_pkt_size);

   FOUT;
   }

Boolean edsr_pkt_support_is_route_update (DsrT_Packet_Option* dsr_tlv_ptr)
   {
   DsrT_Route_Request_Option*    route_request_option_ptr = OPC_NIL;
   DsrT_Route_Reply_Option*      route_reply_option_ptr = OPC_NIL;

   FIN (edsr_pkt_support_is_route_update (<args>));
```

```
   /* This function is called only when the packet is route_request or route_reply.
   returns true if the route_request or route_reply is a route update request/reply */

   /* Determine the type of option   */
   switch (dsr_tlv_ptr->option_type)
      {
      case (DSRC_ROUTE_REQUEST):
         {
         route_request_option_ptr = (DsrT_Route_Request_Option*) dsr_tlv_ptr->dsr_option_ptr;
         FRET (route_request_option_ptr->route_update);
         break;
         }
      case (DSRC_ROUTE_REPLY):
         {
         route_reply_option_ptr = (DsrT_Route_Reply_Option*) dsr_tlv_ptr->dsr_option_ptr;
         FRET (route_reply_option_ptr->route_update);
         break;
         }
      default:
         {
         FRET (OPC_FALSE);
         break;
         }
      }
   }

DsrT_Packet_Option* edsr_pkt_support_route_request_tlv_create (long int request_id,
InetT_Address dest_address, double min_energy, Boolean route_update, List* route_lptr)   // akis
   {
   DsrT_Route_Request_Option*    route_request_ptr = OPC_NIL;
   DsrT_Packet_Option*           dsr_tlv_ptr = OPC_NIL;
   int                           address_length;
   int                           count;
   int                           num_hops = 0;   // in case it is not a route update
   InetT_Address*                hop_address_ptr = OPC_NIL;
   InetT_Address*                copy_address_ptr = OPC_NIL;

   /** Creates the route request option in the DSR packet   **/
   FIN (edsr_pkt_support_route_request_tlv_create (<args>));

   address_length = (inet_address_family_get (&dest_address) == InetC_Addr_Family_v4 ?
IP_V4_ADDRESS_LENGTH : IP_V6_ADDRESS_LENGTH);
   route_request_ptr = edsr_pkt_support_route_request_mem_alloc ();
   route_request_ptr->identification = request_id;
   route_request_ptr->target_address = inet_address_copy (dest_address);
   route_request_ptr->min_energy = min_energy;            // akis
   route_request_ptr->route_update = route_update;         // akis

   num_hops = op_prg_list_size (route_lptr);
   if (route_lptr != OPC_NIL/* && route_update == OPC_TRUE*/)   // i.e. it's route update
      {     // akis2 : if it's route update, we have to copy the route!
      for (count = 1; count < num_hops; count++)
         {  // hope: count starts at 1, so that we omit the source node address from the path
         hop_address_ptr = (InetT_Address*) op_prg_list_access (route_lptr, count);
         copy_address_ptr = inet_address_copy_dynamic (hop_address_ptr);
         op_prg_list_insert (route_request_ptr->route_lptr, copy_address_ptr, OPC_LISTPOS_TAIL);
         }
      }

   dsr_tlv_ptr = edsr_pkt_support_option_mem_alloc ();
   dsr_tlv_ptr->option_type = DSRC_ROUTE_REQUEST;
   dsr_tlv_ptr->option_length = DSR_HEADER_OPTIONS + address_length*(1 + num_hops) + 64 + 1;
   dsr_tlv_ptr->dsr_option_ptr = (void*) route_request_ptr;

   FRET (dsr_tlv_ptr);
   }
```

B.3 Διεργασίες

Οι διεργασίες που χρησιμοποιούνται από το μοντέλο είναι οι εξής :

1. akis_ip_dispatch
2. akis_manet_mgr
3. akis_dsr_rte

Στη συνέχεια, περιγράφονται σύντομα οι λειτουργίες των διεργασίων και παρουσιάζονται οι τροποποιήσεις τους.

akis_ip_dispatch

Η διεργασία *akis_ip_dispatch* διαβάζει από τον χρήστη την ιδιότητα energy-aware, με βάση την οποία αποφασίζει αν θα δημιουργήσει τη διεργασία *akis_manet_mgr*, που θα καλέσει το EDSR, ή τη διεργασία *manet_mgr*, που θα καλέσει το DSR.

Οι τροποποίησεις φαίνονται στον επόμενο πίνακα.

Σημείο τροποποίησης	**Περιγραφή αλλαγής**
Process State Variables	Boolean edsr - παράμετρος Energy-aware Objid own_id
State init_too.Exit executives	Διάβασμα παραμέτρου Energy-aware (edsr)
Συνάρτηση ip_dispatch_cleanup_and_create_child_processes	Δημιουργία διεργασίας akis_manet_mgr αν η ιδιότητα Energy-aware είναι ενεργοποιημένη για το EDSR, αλλιώς δημιουργία manet_mgr για το DSR
Συνάρτηση ip_dispatch_do_init	Διάβασμα πρωτοκόλλου δρομολόγησης MANET από akis_manet_mgr ή manet_mgr
Συνάρτηση ip_dispatch_init_phase_2	Διάβασμα παραμέτρου MANET Gateway από akis_manet_mgr ή manet_mgr

Τροποποιήσεις στη διεργασία akis_ip_dispatch

akis_manet_mgr

Η διεργασία *akis_manet_mgr* έχει τροποποιηθεί ώστε όταν ως πρωτόκολλο δρομολόγησης επιλέγεται το DSR, να γεννά τη διεργασία *akis_dsr_rte*, δηλαδή τη διεργασία του EDSR.

Όλες οι συναρτήσεις της διεργασίας μετονομάστηκαν από *manet_mgr_** σε *akis_manet_mgr_** για να μην υπάρχουν διενέξεις μεταξύ των δύο συναρτήσεων. Οι τροποποιήσεις φαίνονται στον επόμενο πίνακα.

Σημείο τροποποίησης	**Περιγραφή αλλαγής**
Συνάρτηση akis_manet_mgr_routing_process_create	Δημιουργία της διεργασίας akis_dsr_rte

Τροποποιήσεις στη διεργασία akis_manet_mgr

akis_dsr_rte

Η διεργασία *akis_dsr_rte* είναι ο πυρήνας του πρωτοκόλλου EDSR. Οι συναρτήσεις της αφορούν κάθε λειτουργία του πρωτοκόλλου, ενώ γίνεται εκτενής χρήση και των συναρτήσεων που έχουν οριστεί στα εξωτερικά αρχεία. Οι τροποποιήσεις που έγιναν παρουσιάζονται στον παρακάτω πίνακα. Στη συνέχεια, παρατίθεται ο πηγαίος κώδικας ορισμένων ενδεικτικών συναρτήσεων, είτε νέων είτε σημαντικά τροποποιημένων.

Σημείο τροποποίησης	Περιγραφή αλλαγής/λειτουργίας
Process State Variables	int route_update_interval (sec) double route_update_jitter (sec) Boolean route_cache_log
State init.Enter executives	Χρονοδρομολόγηση διακοπών για τις συναρτήσεις edsr_rte_generate_route_update_requests και edsr_rte_print_route_cache_to_file
Header Block	Δήλωση νέων συναρτήσεων και επικεφαλίδων
Συναρτήσεις (Function Block)	
edsr_rte_attributes_parse_buffers_create	Διάβασμα των νέων παραμέτρων (attributes)
edsr_rte_generate_route_update_requests (Νέα)	Καλείται κάθε route_update_interval, ανανεώνει τη route cache και στέλνει πακέτα RU-REQ.
edsr_rte_print_route_cache_to_file (Νέα)	Καλείται από την edsr_rte_generate_route_update_requests και εκτυπώνει τις πληροφορίες της Route Cache σε αρχείο.
edsr_rte_received_pkt_handle	Αλλαγές στην επεξεργασία λαμβανόμενων πακέτων Route Request, Route Reply, Ack Request, Source Route που αφορούν το πεδίο ενέργειας των πακέτων και τη Route Cache.
edsr_rte_app_pkt_arrival_handle	Σε περίπτωση που υπάρχει διαδρομή για τον προορισμό αλλά είναι ανενεργή, χρησιμοποιείται προσωρίνα ενώ στέλνονται πακέτα RU-REQ.
edsr_rte_received_route_request_process	Αν το ληφθέν RREQ είναι RU-REQ, τότε προωθείται στον επόμενο κόμβο. Αλλιώς, αν υπάρχει ενεργή αποθηκευμένη διαδρομή, αποστέλλεται RREP με αυτήν. Αλλιώς, γίνεται επανεκπομπή του RREQ.
edsr_rte_received_route_reply_process	Αν ο κόμβος είναι ο προορισμός του RREP, ανανεώνεται η Route Cache του.
edsr_rte_received_dsr_source_route_option_process	Απενεργοποίηση της λειτουργίας Automatic Route Shortening
edsr_rte_route_request_send	Διαχωρισμός αποστολής RREQ και RU-REQ
edsr_rte_route_reply_send	Αλλαγές που αφορούν το πεδίο ενέργειας στα πακέτα RREP, RU-REP
edsr_rte_maintenance_send	Αλλαγές που αφορούν το πεδίο ενέργειας στα πακέτα ACK
edsr_rte_cached_route_reply_send	Επιλογή ελάχιστης ενέργειας ανάμεσα στην αποθηκευμένη στη μνήμη και στην τιμή του πακέτου
edsr_rte_automatic_route_shortening_check	Απενεργοποίηση της συνάρτησης
edsr_rte_route_cache_update	Αλλαγές που αφορούν το πεδίο ενέργειας σε όλα τα πακέτα
edsr_rte_route_request_expiry_handle	Αλλαγές που αφορούν την αλληλεξάρτηση με άλλες συναρτήσεις
edsr_rte_node_energy (Νέα)	Διαβάζει την υπολειπόμενη ενέργεια του κόμβου από το αντικείμενο της μπαταρίας του κόμβου

Τροποποιήσεις στη διεργασία akis_dsr_rte

```
static void edsr_rte_generate_route_update_requests (void* unused_state_ptr, int unused_code)
      // unused code used for intrtpt schedule call
   {
   List*            keys_lptr = OPC_NIL;
   List*            dest_routes_lptr = OPC_NIL;
   int              num_nodes,num_paths,size,count;
   DsrT_Path_Info*  path_ptr = OPC_NIL;
   char*            key_str = OPC_NIL;
   InetT_Address    dest_addr;
   Boolean          neighbours;
```

```
   FIN (edsr_rte_generate_route_update_requests (<args>));

   /* Get all the keys (destinations) in the hash table   */
   keys_lptr = prg_string_hash_table_keys_get (route_cache_ptr->route_cache_table);

   /* Get the size of the table   */
   num_nodes = op_prg_list_size (keys_lptr);

   // for all nodes (destinations)
   for (count = 0; count < num_nodes; count++)
      {
      /* Reinitialize the variables   */
      num_paths = 0;

      /* Get each destination key   */
      key_str = (char*) op_prg_list_access (keys_lptr, count);

      dest_routes_lptr = (List *) op_prg_mem_alloc (sizeof (List));

      /* Access the routes to each destination */
      dest_routes_lptr = (List*) prg_string_hash_table_item_get (route_cache_ptr-
>route_cache_table, key_str);

      if (dest_routes_lptr == OPC_NIL)
         {
         // No Routes exist to this Destination
         continue;
         }

      // if the nodes (source & destination) are neighbours,
      // then we don't want to keep any other route than the immediate
      if ((neighbours = edsr_route_cache_neighbouring_nodes (dest_routes_lptr)) == OPC_TRUE)
         {
         edsr_route_cache_neighbours_longer_routes_delete (route_cache_ptr, dest_routes_lptr,
key_str);
         }
      else      // they are not neighbours, so delete any possible overlapping routes
         {
         edsr_route_cache_overlapping_routes_delete (route_cache_ptr, dest_routes_lptr);
         }

       // if even one route to this destination is active,
       // then they all need update, so send route update requests (case 1)
      if (edsr_route_cache_all_routes_to_dest_need_update (route_cache_ptr, dest_routes_lptr) ==
OPC_TRUE)      // checked with last_access_time and route_activity_time
         {
         /* Create the IP address from the string   */
         dest_addr = inet_address_create (key_str, InetC_Addr_Family_Unknown);

         num_paths = op_prg_list_size (dest_routes_lptr);

       // if there is only one active path and the nodes aren't neighbours,
       // then find more routes (route request)
         if ((num_paths == 1) && (!neighbours))
            {
            edsr_rte_route_request_send (dest_addr, OPC_FALSE, OPC_FALSE, OPC_NIL);
            }
         else
            {
            // send route update request for all paths to this destination
            for (size = 0; size < num_paths ; size++)
               {
               /* Get each path and send the route update request   */
               path_ptr = (DsrT_Path_Info*) op_prg_list_access (dest_routes_lptr, size);

               edsr_rte_route_request_send (dest_addr, OPC_FALSE, OPC_TRUE, path_ptr-
>path_hops_lptr);   // route update!
               }
            }

         /* Free the address   */
```

```
         inet_address_destroy (dest_addr);
         }

      // else means that all of routes to this destination are inactive
      // if even one of them has expired, then they are too old, so delete all of them (case 2)
      else if (edsr_route_cache_all_routes_to_dest_need_delete (route_cache_ptr,
dest_routes_lptr) == OPC_TRUE)   // checked with last_access_time and path_expiry_time
         {
         edsr_route_cache_all_routes_to_dest_delete (route_cache_ptr, dest_routes_lptr,
key_str);
         }
       // if the conditions above are false,
       // then we just keep inactive routes in the cache for some time, until they expire
       // if these routes are needed in the meantime
       // we will use the route temporarily and send route update requests immediately
       // (case 3) see: app_pkt_arrival function

       // this policy ensures our route cache is always up-to-date for active paths and
       // that we get rid of obsolete paths and
       // keep inactive but not yet obsolete paths, just in case
       // (decrease delay and route traffic)
      }

   op_prg_list_free (keys_lptr);
   op_prg_mem_free (keys_lptr);


   /* Schedule the next check to send route update requests   */
   op_intrpt_schedule_call (op_sim_time () + route_update_interval, 666,
edsr_rte_generate_route_update_requests, OPC_NIL);   // akisopt


   // write route cache file, every route_update_interval, but 10 seconds after
   // we send the route update requests (so that the replies return in the meantime)
   if (route_cache_log)
      op_intrpt_schedule_call (op_sim_time() + 10, 667, edsr_rte_print_route_cache_to_file,
OPC_NIL);

   FOUT;
   }

static void edsr_rte_print_route_cache_to_file (void* unused_state_ptr, int unused_code)
   {
   char*       cache_str = OPC_NIL;
   FILE*       fp = OPC_NIL;

   FIN (edsr_rte_print_route_cache_to_file(<args>));

   cache_str = edsr_route_cache_print_to_string (route_cache_ptr);

   fp = fopen ("C:\\opnet_logs\\route_cache.txt","a");
   fprintf (fp,"-----------------------------------------------------\n\n\n\n");
   fprintf (fp, "%s", cache_str);
   fclose (fp);

   FOUT;
   }

static void edsr_rte_received_pkt_handle (void)
   {
   Packet*                        ip_pkptr = OPC_NIL;
   Packet*                        copy_pkptr = OPC_NIL;
   Packet*                        return_pkptr = OPC_NIL;
   IpT_Rte_Ind_Ici_Fields*        intf_ici_fdstruct_ptr = OPC_NIL;
   IpT_Dgram_Fields*              ip_dgram_fd_ptr = OPC_NIL;
   Packet*                        dsr_pkptr = OPC_NIL;
   List*                          tlv_options_lptr = OPC_NIL;
   int                            num_options, count;
   DsrT_Packet_Option*            dsr_tlv_ptr = OPC_NIL;
   DsrT_Packet_Type               packet_type = DsrC_Undef_Packet;
   char                           addr_str [INETC_ADDR_STR_LEN];
   char                           node_name [OMSC_HNAME_MAX_LEN];
```

```
   char                          temp_str [256];
   Compcode                      status;
   Boolean                       app_pkt_set = OPC_FALSE;
   double                        current_energy;                 // akis
   Boolean                       route_update = OPC_FALSE;       // akis2
   Boolean                       is_destination;                 // akisopt
   double                        path_energy;                    // akisopt
   DsrT_Route_Request_Option*    route_request_ptr = OPC_NIL;    // akisopt

   // akis: this function is called by wait state, every PACKET_ARRIVAL

   /** A packet has arrived. Handle the packet   **/
   /** appropriately based on its various TLV   **/
   /** options set in the DSR header         **/
   FIN (edsr_rte_received_pkt_handle (void));

   /* The process was invoked by the parent   */
   /* MANET process indicating the arrival      */
   /* of a packet. The packet can either be   */
   /* 1. A higher layer application packet      */
   /*    waiting to be transmitted when a       */
   /*    route is found.            */
   /* 2. A MANET signaling/routing packet       */
   /*    arrival which may or may not be a      */
   /*    broadcast packet.            */

   /* Access the argument memory to get the   */
   /* packet pointer.               */
   ip_pkptr = (Packet*) op_pro_argmem_access ();

   if (ip_pkptr == OPC_NIL)
      edsr_rte_error ("Could not obtain the packet from the argument memory", OPC_NIL, OPC_NIL);

   /* Access the information from the incoming IP packet   */
   manet_rte_ip_pkt_info_access (ip_pkptr, &ip_dgram_fd_ptr, &intf_ici_fdstruct_ptr);

   /* Determine the packet type   */
   packet_type = edsr_rte_packet_type_determine (ip_dgram_fd_ptr, intf_ici_fdstruct_ptr);

   /* Check if this IP packet is carrying a DSR header   */
   if (packet_type == DsrC_Higher_Layer_Packet)
      {
      if (LTRACE_ACTIVE)
         {
         inet_address_print (addr_str, ip_dgram_fd_ptr->dest_addr);
         inet_address_to_hname (ip_dgram_fd_ptr->dest_addr, node_name);
         sprintf (temp_str, "to destination %s (%s)", addr_str, node_name);
         op_prg_odb_print_major (pid_string, "An application packet has arrived at this node",
temp_str, OPC_NIL);
         }

      /* This IP datagram does not have a DSR header   */
      /* It should be a higher layer packet         */
      // akis : that means it is an APP packet we just created, going down to be transmitted
      edsr_rte_app_pkt_arrival_handle (ip_pkptr, intf_ici_fdstruct_ptr, ip_dgram_fd_ptr,
OPC_FALSE);   // akiserr

      FOUT;
      }

   /* This packet is received from the MAC layer   */

   /* Update the statistic for the total traffic   */
   edsr_support_total_traffic_received_stats_update (stat_handle_ptr, global_stathandle_ptr,
ip_pkptr);

   /* Get the DSR packet from the IP datagram   */
   op_pk_nfd_get (ip_pkptr, "data", &dsr_pkptr);

   /* Check if this is an application packet   */
   /* or just a DSR routing packet            */
   app_pkt_set = op_pk_nfd_is_set (dsr_pkptr, "data");
```

```
   /* Get the list of options   */
   op_pk_nfd_access (dsr_pkptr, "Options", &tlv_options_lptr);

   /* Set the DSR packet into the IP datagram   */
   op_pk_nfd_set (ip_pkptr, "data", dsr_pkptr);

   if (app_pkt_set == OPC_FALSE)
      {
      /* This is a DSR routing packet. Update   */
      /* the statistic for routing traffic   */
      edsr_support_routing_traffic_received_stats_update (stat_handle_ptr,
global_stathandle_ptr, ip_pkptr);
      }
   else
      {
      /* This is an application packet. Decrease the TTL    */
      /* if i am not the destination node             */
      if (manet_rte_address_belongs_to_node (module_data_ptr, ip_dgram_fd_ptr->dest_addr) ==
OPC_FALSE)
         ip_dgram_fd_ptr->ttl--;
      }

   /* Get the number of options   */
   num_options = op_prg_list_size (tlv_options_lptr);

   for (count = 0; count < num_options; count++)
      {
      /* Make a copy of the incoming packet for each option   */
      copy_pkptr = manet_rte_ip_pkt_copy (ip_pkptr);

      /* Get the DSR packet from the IP datagram   */
      op_pk_nfd_get (copy_pkptr, "data", &dsr_pkptr);

      /* Get the list of options   */
      op_pk_nfd_access (dsr_pkptr, "Options", &tlv_options_lptr);

      /* Set the DSR packet into the IP datagram   */
      op_pk_nfd_set (copy_pkptr, "data", dsr_pkptr);

      /* Get each option   */
      dsr_tlv_ptr = (DsrT_Packet_Option*) op_prg_list_access (tlv_options_lptr, count);

      /* Process the option based on the type   */
      switch (dsr_tlv_ptr->option_type)
         {
         case (DSRC_ROUTE_REQUEST):
            {
            /* Get the node's energy */
            current_energy = edsr_rte_node_energy();   // akis

            /* Check if it's a route update request */
            route_update = edsr_pkt_support_is_route_update (dsr_tlv_ptr);

            /* Read the path's min_energy so far */
            /* We'll need it for route cache update, so we need to save it,
                           as it may be changed by hop_insert_energy_update function */
            route_request_ptr = (DsrT_Route_Request_Option*) dsr_tlv_ptr->dsr_option_ptr;
            path_energy = route_request_ptr->min_energy;

            /* Check if it is the route request destination,
                          so that we don't update the energy */
            is_destination = manet_rte_address_belongs_to_node (module_data_ptr,
ip_dgram_fd_ptr->dest_addr);

            /* Next function inserts this node into the route request list
                          only if it's a regular route request  */
            /* If it is a route update request, it just updates energy */
            edsr_pkt_support_route_request_hop_insert_energy_update (copy_pkptr,
intf_ici_fdstruct_ptr->interface_received, current_energy, route_update, is_destination); //akis
            /* Route cache updated only for regular route requests  */
            if (route_update == OPC_FALSE)   // akis
```

```
      {
      /* Insert the route in the route cache based on   */
      /* the requirement for caching overheard information.   */
      edsr_rte_route_cache_update (dsr_tlv_ptr, ip_dgram_fd_ptr, path_energy);
      }

   /* After possibly inserting the route in the route cache,   */
   /* process the received route request option         */
   edsr_rte_received_route_request_process (copy_pkptr, dsr_tlv_ptr);

   break;
   }

case (DSRC_ROUTE_REPLY):
   {
   /* The packet contains a route reply option   */

   // Now we also update the route cache energy field, for a route reply,
   // but ONLY for the originator of the Route Request
   edsr_rte_received_route_reply_process (copy_pkptr, dsr_tlv_ptr);

   break;
   }

case (DSRC_ROUTE_ERROR):
   {
   /* The packet contains a route error option   */
   /* Process the received route error         */
   edsr_rte_received_route_error_process (copy_pkptr, dsr_tlv_ptr);

   break;
   }

case (DSRC_ACK_REQUEST):
   {
   /* The packet contains an acknowledgement   */
   /* request option. Process the option      */
   status = edsr_rte_received_ack_request_process (copy_pkptr);
   // akis 2: now it sends back the node energy with the acknowledgement

   if (status == OPC_COMPCODE_SUCCESS)
      {
      /* An acknowledgement was sent out for the   */
      /* received acknowledgement request. Remove   */
      /* the acknowledgement request option from   */
      /* the packet received                  */
      op_pk_nfd_get (ip_pkptr, "data", &dsr_pkptr);
      edsr_pkt_support_option_remove (dsr_pkptr, DSRC_ACK_REQUEST);
      op_pk_nfd_set (ip_pkptr, "data", dsr_pkptr);
      num_options--;
      count--;
      }

   break;
   }

case (DSRC_ACKNOWLEDGEMENT):
   {
   /* The packet contains an acknowledgement option   */
   /* The node should add to its route cache the      */
   /* single link from the node identified by the ACK   */
   /* source address to the node identified by the ACK   */
   /* destination address.              */

   /* Insert the route in the route cache based on   */
   /* the requirement for caching overheard information   */
   edsr_rte_route_cache_update (dsr_tlv_ptr, ip_dgram_fd_ptr, 0);

   /* After possibly inserting the route in the route cache   */
   /* process the received acknowledgement option         */
   edsr_rte_received_acknowledgement_option_process (copy_pkptr, dsr_tlv_ptr);
```

```
            break;
            }

         case (DSRC_SOURCE_ROUTE):
            {
            /* The packet contains a DSR source route option   */

            /* Insert the route in the route cache based on         */
            /* the requirement for caching overheard information   */


            /* After possibly inserting the route in the route cache   */
            /* process the received DSR source route option      */
            edsr_rte_received_dsr_source_route_option_process (copy_pkptr);

            break;
            }

         default:
            {
            /* Invalid option in packet   */
            edsr_rte_error ("Invalid Option Type in DSR packet", OPC_NIL,OPC_NIL);
            }
         }
      }
   /* If the destination address in the IP packet      */
   /* matches one of the receiving node's own IP      */
   /* addresses and this is an application packet,    */
   /* remove the DSR header and all DSR options and    */
   /* pass the packet to the higher layer            */
   if (manet_rte_address_belongs_to_node (module_data_ptr, ip_dgram_fd_ptr->dest_addr) ==
OPC_TRUE)
      {
      /* Decapsulate the DSR packet   */
      return_pkptr = edsr_rte_ip_datagram_decapsulate (ip_pkptr);

      if (return_pkptr != OPC_NIL)
         {
         /* Send the IP packet to the higher layer   */
         manet_rte_to_higher_layer_pkt_send_schedule (module_data_ptr, parent_prohandle,
return_pkptr);
         }
      else
         {
         /* Destroy the packet   */
         manet_rte_ip_pkt_destroy (ip_pkptr);
         }
      }
   else
      {
      /* Destroy the packet   */
      manet_rte_ip_pkt_destroy (ip_pkptr);
      }

   FOUT;
   }

static void edsr_rte_route_request_send (InetT_Address dest_address, Boolean
non_prop_route_request, Boolean route_update, List* route_lptr)
   {
   DsrT_Packet_Option*           dsr_tlv_ptr = OPC_NIL;
   IpT_Dgram_Fields*             ip_dgram_fd_ptr = OPC_NIL;
   Packet*                       dsr_pkptr = OPC_NIL;
   Packet*                       ip_pkptr = OPC_NIL;
   char                          dest_node_name [OMSC_HNAME_MAX_LEN];
   char                          dest_hop_addr_str [INETC_ADDR_STR_LEN];
   char                          temp_str [2048];
   Ici*                          ip_iciptr;
   int                           mcast_major_port = IPC_MCAST_ALL_MAJOR_PORTS;
   ManetT_Nexthop_Info*          manet_nexthop_info_ptr = OPC_NIL;      // akis2
   InetT_Address*                next_hop_addr_ptr;            // akis2
```

```
   /** Initiates a route request to a destination   **/
   // if it's a route update, we also need the route path (route_lptr). else it's OPC_NIL

   FIN (edsr_rte_route_request_send (<args>));

   /* Create a route request TLV option   */
   // akis : if it is a route update, it already knows the route path!
   // the min_energy field concerns the intermediate nodes of the path
   dsr_tlv_ptr = edsr_pkt_support_route_request_tlv_create (route_request_identifier,
dest_address, route_cache_ptr->max_energy, route_update, route_lptr);   // akis2

   /* Create the DSR packet   */
   dsr_pkptr = edsr_pkt_support_pkt_create (IpC_Protocol_Unspec);

   /* Set the route request option in the DSR packet header   */
   edsr_pkt_support_option_add (dsr_pkptr, dsr_tlv_ptr);

   /* Set the DSR packet in a newly created IP datagram   */
   /* The source address of the IP datagram is the node's   */
   /* own IP address and the destination address of the   */
   /* IP datagram is the limited broadcast address       */
   /* (255.255.255.255) for IPv4 or the all node link       */
   /* layer multicast address for IPv6            */

   if (inet_address_family_get (&dest_address) == InetC_Addr_Family_v4)
      {

      if (route_update == OPC_FALSE)
         {
         ip_pkptr = edsr_rte_ip_datagram_create (dsr_pkptr,
InetI_Broadcast_v4_Addr,InetI_Broadcast_v4_Addr, OPC_NIL);   // if a regular route request
         }
      else   // akis2 : if it's a route update request, it has different parameters
         {
         // Allocate memory for manet_nexthop_info_ptr
         manet_nexthop_info_ptr = (ManetT_Nexthop_Info *) op_prg_mem_alloc (sizeof
(ManetT_Nexthop_Info));

         // The first element in the path is this source node
         // The second element in the path is the next hop node
         next_hop_addr_ptr = (InetT_Address*) op_prg_list_access (route_lptr, 1);
         ip_pkptr = edsr_rte_ip_datagram_create (dsr_pkptr, dest_address, *next_hop_addr_ptr,
manet_nexthop_info_ptr);
         }
      }
   else
      {
      // akis2 : modification doesn't support IPv6
      ip_pkptr = edsr_rte_ip_datagram_create (dsr_pkptr, InetI_Ipv6_All_Nodes_LL_Mcast_Addr,
InetI_Ipv6_All_Nodes_LL_Mcast_Addr, OPC_NIL);

      /* Install the ICI for IPv6 case */
      ip_iciptr = op_ici_create ("ip_rte_req_v4");
      op_ici_attr_set (ip_iciptr, "multicast_major_port", mcast_major_port);
      op_ici_install (ip_iciptr);
      }

   if (LTRACE_ACTIVE)
      {
      inet_address_print (dest_hop_addr_str, dest_address);
      inet_address_to_hname (dest_address, dest_node_name);
      sprintf (temp_str, "destined to node %s (%s) with ID (%d)", dest_hop_addr_str,
dest_node_name, route_request_identifier);
      op_prg_odb_print_major ("Broadcasting a route request option in packet", temp_str,
OPC_NIL);
      }

   /* Increment the route request identifier   */
   route_request_identifier++;

   /* Access the IP datagram fields   */
   op_pk_nfd_access (ip_pkptr, "fields", &ip_dgram_fd_ptr);
```

```
   /* If the non-propagating route request feature   */
   /* has been enabled, set the TTL field in the   */
   /* route request packet to one               */
   if (non_prop_route_request)
      {
      /* Set the TTL to one   */
      ip_dgram_fd_ptr->ttl = 1;
      }
   else
      {
      /* Set the TTL to the default   */
      ip_dgram_fd_ptr->ttl = IPC_DEFAULT_TTL;
      }

   /* Update the statistic for the total traffic sent   */
   edsr_support_total_traffic_sent_stats_update (stat_handle_ptr, global_stathandle_ptr,
ip_pkptr);

   /* Update the statistics for the routing traffic sent   */
   edsr_support_routing_traffic_sent_stats_update (stat_handle_ptr, global_stathandle_ptr,
ip_pkptr);

   /* Update the statistic for the total number of route requests sent   */
   edsr_support_route_request_sent_stats_update (stat_handle_ptr, global_stathandle_ptr,
non_prop_route_request);

   /* insert the originating route request information in   */
   /* the originating route request table               */

   // we use only originating table for both requests
   edsr_route_request_originating_table_entry_insert (route_request_table_ptr, dest_address,
ip_dgram_fd_ptr->ttl);      // akisru

   if (route_update == OPC_FALSE)
      {
      /* Send the packet to the CPU which will broadcast it   */
      /* after processing the packet                       */
      manet_rte_to_cpu_pkt_send_schedule_with_jitter (module_data_ptr, parent_prohandle,
parent_pro_id, ip_pkptr);
      }
   else
      // akis2 : it's a route update
      {
      /* Install the event state    */
      /* This event will be processed in ip_rte_support.ex.c while receiving     */
      /* DSR control packets. manet_nexthop_info_ptr will point to structure      */
      /* containing nexthop info, so IP table lookup is not again done for them. */
      op_ev_state_install (manet_nexthop_info_ptr, OPC_NIL);

      /* Send the packet after a jitter   */
      /* to the CPU                  */
      edsr_extras_manet_rte_to_cpu_pkt_send_schedule_with_jitter (module_data_ptr,
parent_prohandle, parent_pro_id, ip_pkptr, route_update_jitter);

      op_ev_state_install (OPC_NIL, OPC_NIL);

      }

   /* Clear the ICI if installed */
   op_ici_install (OPC_NIL);

   FOUT;
   }

static double edsr_rte_node_energy (void)
   {
   Objid battery_module_objid;
   double* battery_current_energy_ptr = OPC_NIL;

   /** This function gets the node's remaining energy    **/
   /** from the battery module            **/
```

```
   FIN (edsr_rte_node_energy (void));

   // get the battery module's objid
   battery_module_objid = op_id_from_name (own_node_objid, OPC_OBJTYPE_PROC, "Battery");

   // get the State Variable's battery.current_energy pointer from the battery module
   battery_current_energy_ptr = (double *) op_ima_obj_svar_get (battery_module_objid,
"manet_api_current_energy");

   // return value of pointer
   FRET (*battery_current_energy_ptr)
   }
```