ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΜΕΣΟΛΟΓΓΙΟΥ ΤΜΗΜΑ

ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ & ΔΙΚΤΥΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Πειραματική και συγκριτική αξιολόγηση proactive και reactive πρωτοκόλλων δρομολόγησης, για ασύρματα ad hoc δίκτυα, με τον ns – 2»

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΣΠΟΥΔΑΣΤΗ

Μάριος Ρουσουνέλος

ΕΠΙΒΛΕΠΩΝ: Βασίλειος Δ. Τριανταφύλλου - Καθηγητής, Προϊστάμενος Τμήματος Τηλεπικοινωνιακών Συστημάτων και Δικτύων, Παράρτημα Ναυπάκτου, ΤΕΙ Μεσολογγίου

ΝΑΥΠΑΚΤΟΣ 2013

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Ναύπακτος, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1.

2.

3.

Κεφάλαιο 1: Εισαγωγή

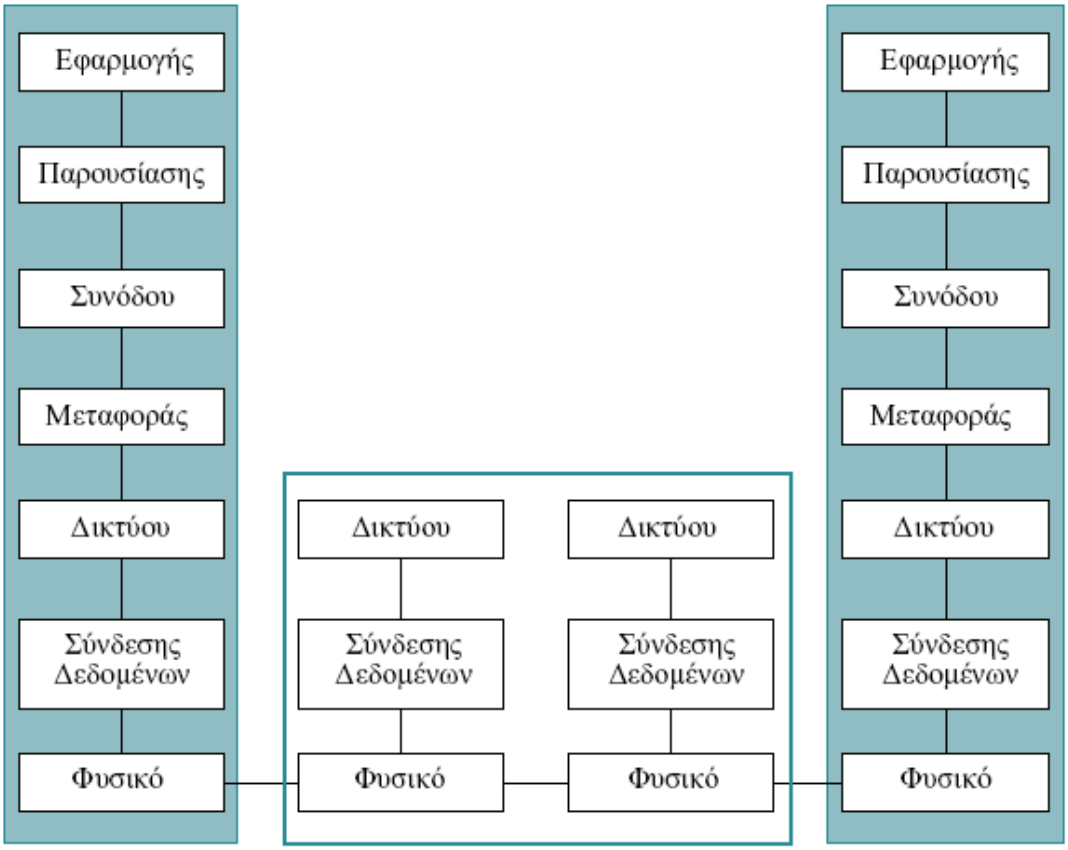
1.1 Τι είναι ασύρματο δίκτυο ad hoc

1.1.1 Δίκτυα μεταξύ υπολογιστών

Ένα δίκτυο υπολογιστών είναι ένα σύστημα που αναπτύσσεται με στόχο τη διασύνδεση δύο ή και περισσότερων υπολογιστών. Στο παρακάτω σχήμα φαίνεται μία περίπτωση δικτύου υπολογιστών, το οποίο αποτελείται από δικτυακές συσκευές που τις ονομάζουμε κόμβους και καθώς και τις συνδέσεις αυτών που τις ονομάζουμε ακμές.

Εικόνα 1: Παραδοσιακό δίκτυο υπολογιστών

Προκειμένου να επικοινωνήσουν οι κόμβοι ενός δικτύου, κάθε κόμβος εκτελεί κάποιες λειτουργίες σε επτά διαφορετικά και αυτόνομα επίπεδα. Η ανάλυση του δικτύου στα επίπεδα αυτά συμβαίνει με βάση το μοντέλο του OSI, όπως φαίνεται παρακάτω:



Εικόνα 2: Μοντέλο OSI

1. *Φυσικό επίπεδο (Physical layer)*. Καλύπτει τη φυσική διεπαφή ανάμεσα σε συσκευές στέλνοντας ακατέργαστα δεδομένα(bits).

2. *Επίπεδο ζεύξης δεδομένων (Data link layer)*. Στόχος αυτού του στρώματος είναι να κάνει τη φυσική ζεύξη αξιόπιστη και να παρέχει τα μέσα για ενεργοποίηση, επισκευή και απενεργοποίηση της ζεύξης. Έτσι το υψηλότερο στρώμα μπορεί να υποθέτει ότι η μετάδοση της ζεύξης είναι απαλλαγμένη από σφάλματα. Ένα υποεπίπεδο είναι το MAC(Medium Access Layer). Το MAC είναι υπεύθυνο για την ταυτόχρονη πρόσβαση των κόμβων στο ίδιο φυσικό μέσο.

3. *Επίπεδο δικτύου(Network layer)*. Σε αυτό το επίπεδο ο υπολογιστής έρχεται σε επικοινωνία με το δίκτυο για να καθορίσει τη διεύθυνση προορισμού και να ζητήσει συγκεκριμένες υπηρεσίες δικτύου, όπως προτεραιότητα.

4. *Επίπεδο μεταφοράς(Transport layer)*. Εξασφαλίζει ότι τα δεδομένα παραδίδονται χωρίς σφάλματα, στη σωστή σειρά και χωρίς απώλειες.

5. *Επίπεδο συνόδου(Session layer)*. Επιτρέπει σε χρήστες με διαφορετικού τύπου μηχανή να αποκαταστήσουν συνόδους μεταξύ τους. Επίσης παρέχει υπηρεσίες ανάκτησης, ομαδοποίησης και τρόπους διαλόγου.

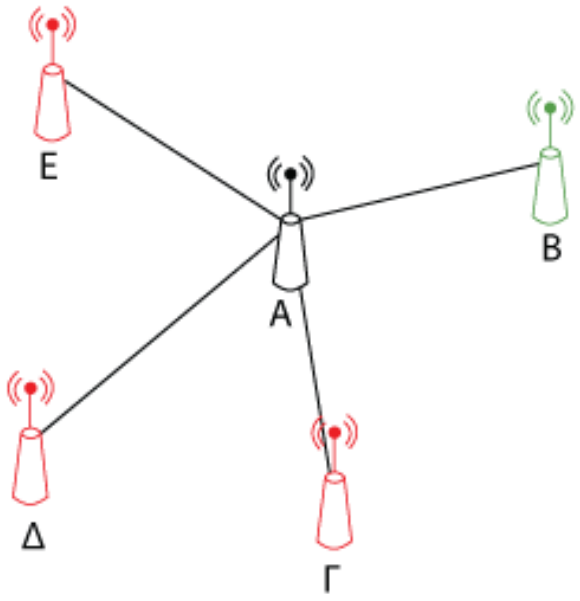
6. *Επίπεδο παρουσίασης(Presentation layer)*. Καθορίζει τη σύνταξη και γενικότερα τη μορφή της μεταδιδόμενης πληροφορίας. Παράδειγμα κάποιες από τις υπηρεσίες αυτού του στρώματος είναι η συμπίεση και η αποκρυπτογράφηση.

7. *Επίπεδο εφαρμογής(Application layer)*. Παρέχει το μέσο ώστε οι εφαρμογές να έχουνε πρόσβαση στο περιβάλλον του OSI. Επιπλέον εφαρμογές είναι η μεταφορά αρχείων ,το ηλεκτρονικό ταχυδρομείο κ.α.

Πέρα από την ανάλυση των δικτύων κατά OSI, υπάρχουνε και άλλα μοντέλα με λιγότερα επίπεδα όπως το μοντέλο TCP/IP. Στα πλαίσια της παρούσας εργασίας, μας ενδιαφέρει το επίπεδο δικτύου που είναι υπεύθυνο για τη δρομολόγηση.

1.1.2 Κινητά ασύρματα δίκτυα

Ασύρματο δίκτυο ονομάζεται ένα δίκτυο που αποτελείται από δύο η περισσότερους κόμβους, οι οποίοι συνδέονται μεταξύ τους με ασύρματους συνδέσμους (ραδιοσυνδέσμους). Στην περίπτωση που οι κόμβοι αυτοί κινούνται στο χώρο, τότε πρόκειται για ένα κινητό ασύρματο δίκτυο.

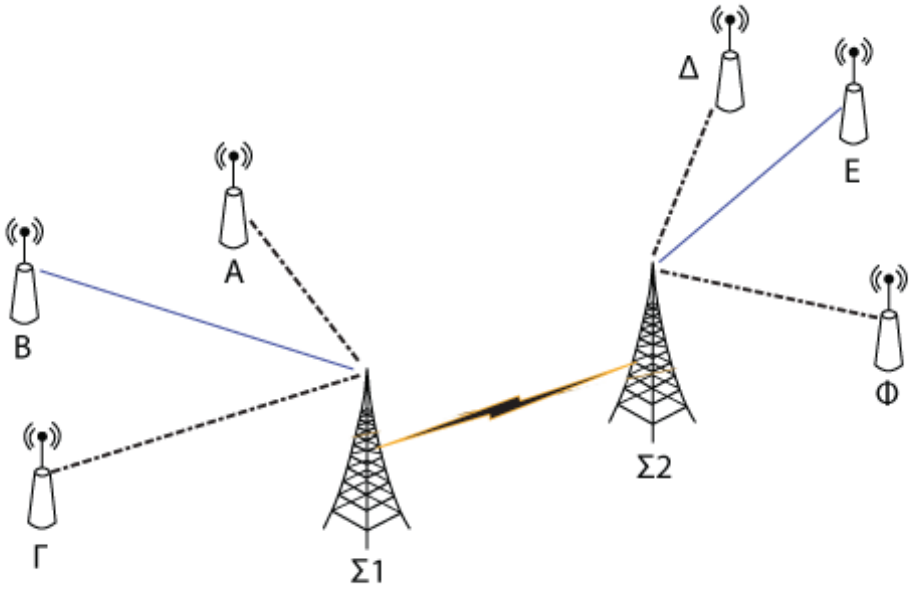


Εικόνα 3: Broadcast ενός ασύρματου δικτύου

Υπάρχουν διαφορετικού τύπου ασύρματα δίκτυα με κοινό τους χαρακτηριστικό την broadcast φύση τους ως μέσο επικοινωνίας. Η απόδοση ενός ασύρματου δικτύου μπορεί να επηρεαστεί από διάφορους παράγοντες. Όταν ένας κόμβος θέλει να επικοινωνήσει με έναν άλλο κόμβο, υπάρχει περίπτωση να βρίσκονται και άλλοι κόμβοι στην εμβέλεια του οπότε να λαμβάνουν και αυτοί το μήνυμα. Αυτό προκαλεί δύο σημαντικά προβλήματα του hidden terminal και του exposed terminal.

Σε ένα δίκτυο οι κόμβοι μπορούν να κινούνται στο χώρο, κάτι που σημαίνει ότι μπορούν να επικοινωνήσουν μόνο με τους κόμβους που βρίσκονται στην εμβέλεια τους. Βάσει αυτού τα κινητά δίκτυα χωρίζονται σε δύο κατηγορίες: τα ασύρματα δίκτυα σταθερής υποδομής και τα ad hoc δίκτυα.

Στα δίκτυα σταθερής υποδομής οι κόμβοι διακρίνονται στους σταθμούς βάσης(base stations) και τους κινητούς σταθμούς(mobile stations). Υπάρχει ένα δίκτυο κορμού, συνήθως είναι ένα ενσύρματο δίκτυο, στο οποίο διασυνδέονται οι σταθμοί βάσης μεταξύ τους και αποτελεί και τη συνολική έκταση που καλύπτει το δίκτυο. Ο κάθε κινητός σταθμός έχει επικοινωνία με το σταθμό βάσης στον όποιο βρίσκεται. Έτσι, η επικοινωνία μεταξύ δύο κινητών κόμβων γίνεται μέσω των αντίστοιχων σταθμών βάσης.



Εικόνα 4: Ασύρματο δίκτυο σταθερή υποδομή

Στο παραπάνω σχήμα ο κόμβος Β επιδιώκει να έρθει σε επικοινωνία με το κόμβο Ε. Για να συμβεί αυτό θα πρέπει αρχικά ο Β να επικοινωνήσει με τον σταθμό 1 (Σ1), στη συνέχεια μέσω του δικτύου των σταθμών βάσης να επικοινωνήσει με τον σταθμό 2 (Σ2) και τέλος μέσω του Σ2 να επικοινωνήσει με τον Ε.

1.1.3 Ασύρματα ad hoc δίκτυα

Στην γενικευμένη του μορφή, ένα ασύρματο ad hoc δίκτυο αποτελείται από ένα σύνολο κινητών κόμβων, οι οποίοι μπορούν να επικοινωνήσουν μεταξύ τους χωρίς την ύπαρξη κάποιας σταθερής υποδομής. Πρόκειται για ένα δίκτυο του οποίου η τοπολογία μπορεί να αλλάζει συνεχώς καθώς οι κόμβοι κινούνται στο χώρο, ανά πάσα στιγμή είναι δυνατό να προστεθούν νέοι κόμβοι ή αντίστοιχα κάποιοι άλλοι να εγκαταλείψουν το δίκτυο. Ακόμα, τη μετάδοση κάποιου μηνύματος μπορεί να επηρεάσουν και εξωγενείς παράγοντες. Τα ad hoc δίκτυα είναι γνωστά και ως MANET(Mobile Ad hoc NETworks). Τα τερματικά σε ένα ad hoc δίκτυο λειτουργούν όχι μόνο σαν συστήματα που στέλνουν ως αποστολείς ή λαμβάνουν ως παραλήπτες πληροφορία (end systems)

αλλά και ως ενδιάμεσα συστήματα αλλά και σαν ενδιάμεσα συστήματα (συστήματα που προωθούν πακέτα άλλων κόμβων). Κάθε κόμβος είναι σε θέση να μεταδίδει πληροφορία σε οποιονδήποτε κόμβο επιλέγει χωρίς τη μεσολάβηση ενός κεντρικού διαχειριστή (administrator) άλλα χρησιμοποιώντας τους ενδιάμεσους κόμβους σαν δρομολογητές (routers). Οι συσκευές λειτουργούν αυτόνομα σχηματίζοντας groups και έχουν την δυνατότητα να συνδεθούν με κάποιο άλλο σταθερό δίκτυο(υβριδικό σύστημα) ή ακόμα και στο Internet. Μπορούμε να αντιληφθούμε ότι ένα ad hoc δίκτυο δεν είναι κάτι συγκεκριμένο όσον αφορά π.χ τη τοπολογία, τους κόμβους αλλά κοινό χαρακτηριστικό είναι η απουσία μίας σταθερής υποδομής. Έτσι συμπεραίνουμε ότι οι κόμβοι, υπό κάποιες προϋποθέσεις, είναι υπεύθυνοι για την ομαλή λειτουργία του δικτύου. Όταν όλοι οι κόμβοι σε ένα δίκτυο έχουν τις ίδιες δυνατότητες και καθήκοντα, τότε το δίκτυο ονομάζεται συμμετρικό δίκτυο, στην αντίθετη περίπτωση ονομάζεται ασύμμετρο. Ακόμα, ορισμένοι από τους κόμβους σε ένα ασύμμετρο δίκτυο μπορεί να είναι οι αρχηγοί του group στο οποίο ανήκουν.

Σε ένα ad hoc δίκτυο όπως αναφέραμε παραπάνω, συμβαίνουν συχνά αλλαγές στη τοπολογία του δικτύου λόγω των κινούμενων κόμβων. Αυτό συνεπάγεται και την ανάγκη για διαρκή αναπροσαρμογή της ισχύς μετάδοσης των κόμβων αυτών. Επιπλέον, οι αλλαγές στην τοπολογία είναι δυνατό να οφείλονται στη διακοπή της λειτουργίας κάποιων κόμβων λόγω μπαταρίας, καθώς σε αυτή την περίπτωση το δίκτυο χάνει τις συγκεκριμένες διαδρομές και έτσι αλλάζουν οι ακμές του. Επίσης, λόγω της κινητικότητας των κόμβων δεν είναι εύκολο να υπάρχει ξεκάθαρη εικόνα του δικτύου και των πόρων που διαθέτει. Το χαρακτηριστικό αυτό οδηγεί τον κατασκευαστή στο να μην μπορεί να εγγυηθεί για την ποιότητα των υπηρεσιών (QoS). Έτσι για τα ad hoc δίκτυα τα πρωτόκολλα προσεγγίζουν το θέμα με μεγαλύτερη ανεκτικότητα (Soft QoS) αλλά και με τον περιορισμό ότι τα δίκτυα θα ικανοποιούν ορισμένες απαιτήσεις. Για παράδειγμα, υπάρχουν πρωτόκολλα που υπολογίζουν τη τοπολογία του δικτύου προβλέποντας τη κίνηση των κόμβων με σκοπό να υπολογιστούν τα καινούρια μονοπάτια χωρίς καθυστερήσεις. Από τις αλλαγές στη τοπολογία προκύπτει και το πρόβλημα για την ασφάλεια του δικτύου. Τα συγκεκριμένα δίκτυα δεν έχουν μία κεντρική λειτουργία διαχείρισης, οι κόμβοι δεν μπορούν να προστατευθούν, όπως επίσης την δυνατότητα που έχει ένας εξωτερικός κόμβος να εισέλθει στο δίκτυο με σκοπό να το βλάψει. Ο κάθε κόμβος σε κάποιες περιπτώσεις μπορεί να είναι υπεύθυνος για να μεταδώσει ξένα πακέτα, αυτό περιορίζει την ικανότητα εξυπηρέτησης που έχει ένα ad hoc δίκτυο όταν πρέπει να διαχειριστεί μια ενδεχόμενη αύξηση των χρηστών σε σχέση με τη χωρητικότητα που μπορεί να επιτευχθεί. Έρευνες έχουνε δείξει ότι όσο μεγαλώνει ένα Ad hoc δίκτυο τόσο μικρότερο γίνεται το throughput του δικτύου. Επίσης, άλλες έρευνες έχουνε δείξει ότι το throughput έχει καλύτερες τιμές όταν αυξάνεται η κινητικότητα των κόμβων. Όπως αναφέραμε παραπάνω κάποιο κόμβοι σταματούν λόγω μπαταρίας. Ένας κόμβος μπορεί να είναι σε 4 καταστάσεις λειτουργίας: στη κατάσταση μετάδοσης, κατάσταση παραλαβής ,κατάσταση αναμονής και κατάσταση αναστολής, με την τελευταία κατάσταση να χρησιμοποιεί την μικρότερη ποσότητα ενέργειας. Στη μετάδοση ο κόμβος χρειάζεται ενέργεια για να δημιουργήσει και να επεξεργαστεί το σήμα που θα μεταδοθεί. Στη παραλαβή ο κόμβος θα χρειαστεί ενέργεια για να επεξεργαστεί το σήμα που λαμβάνει, ενώ σε κατάσταση αναμονής δαπανάται ενέργεια για τη παρακολούθηση του φυσικού μέσου μετάδοσης. Στόχος είναι οι κόμβοι να μπορούν να προβλέψουν ποιές χρονικές στιγμές μπορούν να μπούνε σε κατάσταση

αναστολής. Τέλος άλλο ένα μειονέκτημα μπορεί να θεωρηθεί και ο περιορισμένος επεξεργαστής που έχουνε οι περισσότερες συσκευές, για λόγους οικονομίας, που αυτό αυτόματα σημαίνει παραπάνω χρόνο για υπολογισμούς.

1.1.3.1 Εφαρμογές των ad hoc δικτύων

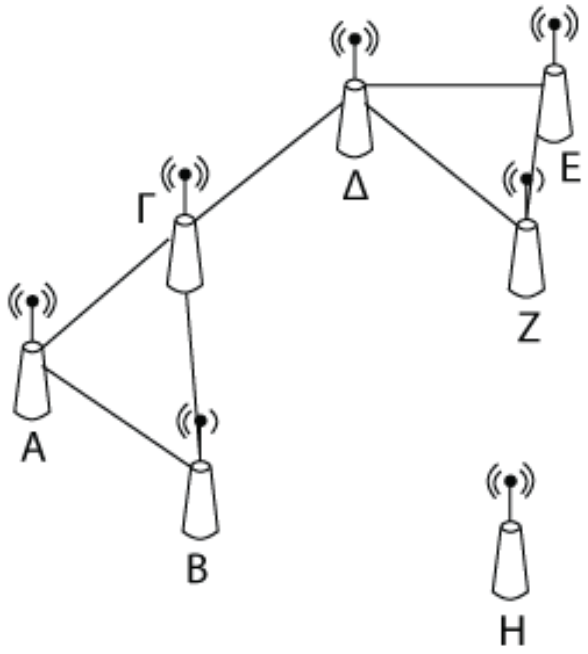
Τα πλεονεκτήματα που έχουν τα ad hoc δίκτυα τα κάνουν ιδιαίτερα χρήσιμα σε κάποιες εφαρμογές όπου με προηγούμενες τεχνολογίες δεν υπήρχε η δυνατότητα να πραγματοποιηθούν.

- Οι virtual αίθουσες, αίθουσες συνεδριάσεων ή ακαδημαϊκές πρέπει να δίνουνε την δυνατότητα στους χρήστες να μπορούν με τα laptops τους να δημιουργούν ένα ad hoc δίκτυο ώστε η επικοινωνία μεταξύ τους να είναι ευκολότερη.
- Ένα Ad hoc δίκτυο στο σπίτι θα έδινε τη δυνατότητα στο χρήστη μέσω του laptop να διαχειρίζεται όλους τους κόμβους(συσκευές) στο σπίτι ανεξάρτητα από που βρίσκετε. Χρησιμοποιώντας ένα πρωτόκολλο σαν το Mobile Ip, οι κόμβοι στο σπίτι θα λειτουργούν σαν να είναι συνδεδεμένοι με το κλασσικό υπολογιστικό περιβάλλον σε αντίθεση με το να συμμετέχει με περισσότερη ενέργεια στο να δημιουργήσει ένα ad hoc δίκτυο.
- Σε καταστάσεις έκτακτης ανάγκης μετά από κάποια καταστροφή όπου οι υποδομές θα έχουνε καταρρεύσει η ανάγκη για επικοινωνία είναι άμεση. Για παράδειγμα τα οχήματα της αστυνομίας και της πυροσβεστικής θα μπορούν να μείνουν περισσότερη διάρκεια σε επαφή και να προσφέρουν πληροφορίες πιο γρήγορα εάν μπορούν να δημιουργήσουν ένα ad hoc δίκτυο στα μέρη αυτά ή και να παρέχει πρόσβαση στο internet.
- Σε μέρη όπως το λιμάνια, αεροδρόμια και αυτοκινητόδρομους όπου η παροχή βοήθειας και πληροφοριών ανά πάσα στιγμή είναι πολλή σημαντική.
- Σε στρατιωτικές επιχειρήσεις οι ανάγκες για επικοινωνία είναι άμεσες. Χρησιμοποιώντας για κόμβους αεροπλάνα, στρατιώτες δημιουργείται ένα ad hoc δίκτυο χωρίς να επηρεάζει η μη ύπαρξη σταθερής υποδομής.
- Η τοποθέτηση μικροσκοπικών συσκευών(sensor dust) που είναι φθηνές στην κατασκευή τους μπορούν να βοηθήσουν ελένχοντας επικίνδυνες καταστάσεις για το περιβάλλον. Για παράδειγμα, σε κίνδυνο έκρηξης ενός εργοστασίου αντί να ρισκάρουμε στέλνοντας ένα άνθρωπο, θα γινόταν μέσω αεροπλάνου να πεταχτούν αισθητήρες που θα σχημάτιζαν ένα ad hoc δίκτυο και θα μάζευαν όλες τις απαραίτητες μετρήσεις.
- Η δημιουργία PAN(personal area network) είναι μία άλλη χρήση ad hoc δικτύου. Όπου θα υπάρχουν κόμβοι που σχετίζονται με ένα άτομο, οι οποίοι κόμβοι θα μπορούν να τοποθετηθούν στην τσάντα η ακόμα και στη ζώνη ενός ατόμου. Σχέδια στο μέλλον είναι να υπάρχουνε virtual reality συσκευές ενσωματωμένες στο κεφάλι και άλλες συσκευές που θα λειτουργού αποκλειστικά με αφή. Αυτές οι συσκευές θα επικοινωνούν μεταξύ τους αφού θα είναι συνδεδεμένες με τις δραστηριότητες αυτού του

ατόμου. Εκμεταλλευόμενοι τις τεχνολογίες που προσφέρει ένα ad hoc δίκτυο, κάθε PAN θα μπορεί να ενωθεί προσωρινά με κάποιο άλλο αφού ένας άνθρωπος θα μετακινηθεί και θα έρθει σε επαφή και με άλλα άτομα.

1.1.4. Δρομολόγηση στα ad hoc δίκτυα

Μία απλή απεικόνιση ενός ad hoc δικτύου φαίνεται στο σχήμα 5 . Με βάση τα χαρακτηριστικά που έχουμε αναφέρει έστω ότι ο κόμβος Α θέλει να στείλει ένα πακέτο στο κόμβο Z. Από τη στιγμή που ο κόμβος Z βρίσκεται εκτός της εμβέλειας του κόμβου Α, το πακέτο θα σταλεί με την βοήθεια ενδιάμεσων κόμβων. Το ότι υπάρχει η δυνατότητα μέσω ενδιάμεσων κόμβων να έρθουν σε επικοινωνία δύο κόμβοι δεν θα είναι πάντα εφικτό.



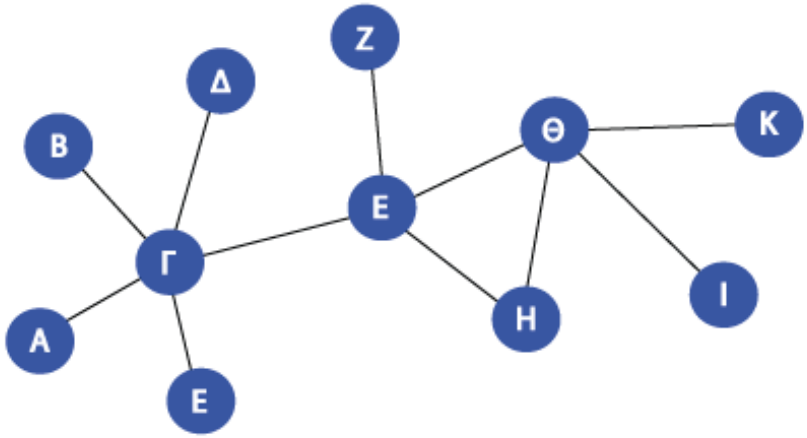
Εικόνα 5: Δρομολόγηση στα Ad hoc δίκτυα

Για παράδειγμα για το κόμβο Z δεν μπορεί να επικοινωνήσει κάποιος αφού βρίσκεται εκτός εμβέλειας από τους άλλους. Επίσης η διαδρομή που χρησιμοποιήθηκε δεν είναι η πιο σύντομη αφού αντί για την Α→Γ→Δ→Z, ακολουθήθηκε αυτή Α→Β→Γ→Δ→Z. Η διαδικασία που ακολούθησε ο κόμβος Α αρχικά ήτανε να ανακαλύψει τον Z. Όπως αναφέραμε στα ad hoc δίκτυα δεν υπάρχει σταθερή υποδομή οπότε ο Α δεν γνωρίζει ούτε αν υπάρχει ο κόμβος που ψάχνει στο δίκτυο, πόσο μάλλον την διαδρομή που πρέπει να ακολουθήσει για να φτάσει σε αυτόν. Στη συνέχεια πρέπει να βρεθεί ένα μονοπάτι για να γίνει η αποστολή των πακέτων. Σε κάποιους αλγόριθμους αυτά τα δύο βήματα θεωρούνται μία διαδικασία. Οι περισσότεροι αλγόριθμοι δρομολόγησης σε ad hoc δίκτυα χρησιμοποιούν την τεχνική της πλημμύρας.

Η δρομολόγηση στα ad hoc δίκτυα γενικότερα είναι πιο δύσκολη από ότι σε ένα παραδοσιακό ενσύρματο δίκτυο και αυτό λόγω ότι σε ένα ad hoc δίκτυο έχει συγκεκριμένη χωρητικότητα για όλους.

1.1.4.1. Πλημμύρα

Η συγκεκριμένη τεχνική λειτουργεί ως εξής: όταν ένας κόμβος χ θέλει να στείλει ένα πακέτο σε ένα κόμβο ψ, αρχίζει να στέλνει μηνύματα για να τον ανακαλύψει σε όλους τους γειτονικούς του κόμβους. Αυτοί με την σειρά τους κάνουν το ίδιο προς τους κόμβους που βρίσκονται γύρω τους, αυτή η διαδικασία συνεχίζεται μέχρι να βρεθεί ο κόμβος ψ. Όταν ο κόμβος βρεθεί τότε στην ουσία έχει βρεθεί και η διαδρομή για να φτάσει το πακέτο σε αυτόν. Ανάλογα τον αλγόριθμο η διαδικασία έχει διαφορετικούς χρόνους αλλά βασίζεται σε αυτήν την λογική. Για παράδειγμα στο παρακάτω δίκτυο εάν ο κόμβος Ε θέλει να στείλει ένα πακέτο στο κόμβο Κ, πρώτη κίνηση είναι να αρχίζει να ρωτάει τους γειτονικούς κόμβους για τον Κ. Με τη σειρά τους αυτοί θα ρωτήσουν τους γείτονες τους και τελικά ο Κ θα βρεθεί από τον Θ. Έτσι βρέθηκε το μονοπάτι για να αποσταλούν τα πακέτα. Το αρνητικό στη περίπτωση της πλημμύρας είναι ότι ο κόμβος Ε για το μήνυμα αναζήτησης που απέστειλε μόνο προς τη μεριά του Θ αλλά προς όλο το δίκτυο. Αυτό σημαίνει ότι ένα μεγάλο μέρος του δικτύου επιβαρύνθηκε χωρίς κανένα λόγο. Στο παρακάτω σχήμα (σχήμα 6) φαίνεται ότι μόνο τέσσερις κόμβοι χρησιμοποιήθηκαν στην τελική διαδρομή αντίθετα το μήνυμα για την ανακάλυψη του το έλαβαν όλοι οι κόμβοι. Στους αλγόριθμους δρομολόγησης σε ad hoc δίκτυα υπάρχουνε διάφορες τεχνικές όπου αυτό το μειονέκτημα της πλημμύρας μειώνεται, όπως και διαφορετική λογική στο τρόπο επικοινωνίας μεταξύ δύο κόμβων.



Εικόνα 6: Πλημμύρα

1.1.5. Κατηγοριοποίηση

Υπάρχουν διάφορες κατηγορίες αλγορίθμων στα ad hoc δίκτυα, η κάθε μία με διαφορετικά κριτήρια. Για παράδειγμα ένα κριτήριο ταξινόμησης των αλγορίθμων βασίζεται στην καλύτερη διαδρομή, δηλαδή αυτήν που έχει τα λιγότερα hops. Υπάρχουν οι Link state αλγόριθμοι όπου στηρίζονται στο ιστορικό κάθε κόμβου και στις ακμές με ισχυρό σήμα για να επιλέξουν ποιες ακμές θα χρησιμοποιήσουν. Επίσης υπάρχει κατηγορία αλγορίθμων που τους χωρίζει σε ιεραρχικούς και μη ιεραρχικούς. Η συγκεκριμένη κατηγορία χωρίζει τους κόμβους σε επίπεδα, με τους κόμβους που βρίσκονται στα ψηλά επίπεδα να συμπεριφέρονται σαν αρχηγοί(base stations) για την γειτονιά που ανήκουν, έχοντας παραπάνω αρμοδιότητες.

Στη συγκεκριμένη εργασία θα μας απασχολήσουν οι proactive(προνοητικοί) και οι reactive(αντιδραστικοί) αλγόριθμοι. Οι proactive αλγόριθμοι υπολογίζουν συνεχώς τις διαδρομές μέσα σε ένα δίκτυο. Έτσι οποιοδήποτε πακέτο πρέπει να αποσταλεί η διαδρομή είναι ήδη γνωστή και χρησιμοποιείται άμεσα. Άρα ο κάθε κόμβος μέσα στο δίκτυο έχει μία διαδρομή για όλους τους υπόλοιπους κόμβους ανά πάσα χρονική στιγμή. Το μειονέκτημα σε αυτήν τη λογική είναι ότι το overhead αυξάνεται αφού ο πίνακας κάθε κόμβου ενημερώνεται διαρκώς για τη τοπολογία του δικτύου. Ένα άλλο μειονέκτημα είναι ότι τα μονοπάτια κατά πάσα πιθανότητα λόγω της κίνησης των κόμβων θα χαθούν, άρα οποιαδήποτε προσπάθεια να διορθωθεί αυτό το link χωρίς να χρησιμοποιείται από κάποιο κόμβο θα πάει χαμένη. Αυτή η χαμένη προσπάθεια όμως θα στοιχίσει στο δίκτυο χαμένο bandwidth και θα προκαλέσει και μεγαλύτερες ζημιές, αφού πακέτα ελένχου που μπαίνουν στην ουρά θα χαθούν και αυτό με τη σειρά του σημαίνει καθυστερήσεις και συμφόρηση στο δίκτυο.

Αντίθετα τα reactive πρωτόκολλα η αλλιώς και on demand πρωτόκολλα στέλνουν την πληροφορία μόνο όταν ζητηθεί κάποιο μονοπάτι μεταξύ δύο κόμβων. Όπως είναι λογικό οι κόμβοι και γενικότερα το δίκτυο επιβαρύνεται λιγότερο(λιγότερο Bandwidth) από πακέτα με καινούρια μονοπάτια αλλά είναι πιο αργά σε σχέση με proactive στο να βρεθεί μια διαδρομή όταν χρειάζεται να σταλεί κάποιο πακέτο. Οι παραπάνω τύποι πρωτοκόλλων μπορούν να συνδυαστούν σε ένα(Zone Routing Protocol)

Κεφάλαιο 2: Δρομολόγηση σε ad hoc

2.1. Δρομολόγηση

Κάθε κόμβος σε ένα ad hoc δίκτυο, εάν αναλάβει να στείλει κάποιο πακέτο, αυτόματα συμμετέχει στην τοπολογία του δικτύου. Αυτός είναι και ένας παρόμοιος τρόπος που αντιδρούν οι κόμβοι στο internet, ή στο intranet κάποιας εταιρίας/ πανεπιστημίου για να δημιουργήσουν ένα σύστημα δρομολόγησης. Στο internet τα πρωτόκολλα δρομολόγησης παρέχουν πληροφορίες απαραίτητες για κάθε κόμβο ώστε να μπορεί να προωθήσει τα πακέτα στον επόμενο κόμβο από την πηγή μέχρι τον αποστολέα.

Στα ad hoc δίκτυα, τα πρωτόκολλα λειτουργούν αυτόνομα, προσαρμόζονται στις αλλαγές που γίνονται στη κατάσταση ενός δικτύου και προσφέρουν multihops μονοπάτια από την πηγή στο παραλήπτη.

Το OSPF είναι ένα παράδειγμα πρωτοκόλλου που ανήκει στη κατηγορία link state(ο κάθε κόμβος συλλέγει πληροφορίες για την κατάσταση των μονοπατιών που έχουν γίνει μεταξύ άλλων κόμβων στο δίκτυο). Ο SPF μπορεί να κατασκευάσει διαδρομές για κάποιους κόμβους εάν η κατάσταση των μονοπατιών είναι γνωστή. Το αρνητικό με αυτά τα πρωτόκολλα σε δυναμικά δίκτυα όπως τα ad hoc είναι ότι χρειάζονται ένα μεγάλο μέρος του bandwidth(εύρος ζώνης) για να γνωρίζουν την τρέχουσα κατάσταση του δικτύου. Όμως έχουν δημιουργηθεί τέτοιου τύπου πρωτόκολλα που δεν χρειάζεται όλοι οι κόμβοι να γνωρίζουν τη κατάσταση του δικτύου και ανακαλύπτουν διαδρομές εφόσον τους ζητηθεί ή είναι κόμβοι που θα επηρεαστούν από την αλλαγή αυτή (π.χ είναι γειτονικός κόμβος).

Μία άλλη κατηγορία αλγορίθμων είναι οι DV(distance vector) οι οποίοι ονομάζονται έτσι γιατί ο πίνακας δρομολόγησης κάθε κόμβου περιέχει έναν αριθμό που συνήθως είναι η απόσταση από τον κόμβο στο τελικό προορισμό. Οι συγκεκριμένοι αλγόριθμοι είναι εύκολοι για να προγραμματιστούν και χρησιμοποιούν λιγότερο μνήμη άλλα έχουνε ένα σημαντικό μειονέκτημα οι γειτονικοί κόμβοι μπορούν αν μπερδευτούν μεταξύ τους και να ανταλλάσσουν διαρκώς τα ίδια πακέτα και ταυτόχρονα να αυξάνουν την απόσταση για το τελικό κόμβο μέχρι η απόσταση να φτάσει στο επιτρεπόμενο όριο(counting to infinity).

Γενικότερα τα ad hoc δίκτυα είναι ένα από τα μεγαλύτερα πειράματα όσον αφορά το σχεδιασμό πρωτοκόλλων και θα συνεχίσουν να έχουν τεράστιο ενδιαφέρον. Ίσως στο μέλλον τα πρωτόκολλα αυτά μπορούν με κάποιες αλλαγές να βοηθήσουν και να αλλάξουν τα πρωτόκολλα δρομολόγησης που χρησιμοποιούνται για το Internet.

2.2. Proactive πρωτόκολλα

2.2.1. DSDV

Το DSDV είναι ένα proactive πρωτόκολλο που με τη χρήση πινάκων δρομολόγησης τα πακέτα μεταδίδονται μεταξύ των κόμβων σε ένα δίκτυο. Κάθε πίνακας δρομολόγησης, σε όλους τους κόμβους, περιέχει όλους τους διαθέσιμους προορισμούς και τα hops που χρειάζεται για να φτάσει σε κάποιον από τους υπόλοιπους κόμβους. Κάθε νέα είσοδο στο πίνακα έχει μία συγκεκριμένη ακολουθία αριθμών που προέρχεται από το κόμβο προορισμού. Ο ακολουθιακός αριθμός δείχνει το πόσο νέα ή παλιά είναι μία εγγραφή. Αν βρεθούν διαδρομές με τον ίδιο αριθμό θα επιλέξει εκείνη με το μικρότερο μήκος.

Για να μένουν ενημερωμένοι οι πίνακες, ανά διαστήματα μεταδίδουν ενημερώσεις, ώστε οι νέες πληροφορίες να είναι διαθέσιμες. Τα δεδομένα στον πίνακα ανανεώνονται μεταξύ της άφιξης της πρώτης και της καλύτερης διαδρομής για κάθε προορισμό. Ένα πλεονέκτημα αυτού του αλγόριθμου είναι η μετάδοση μίας μη σταθερής διαδρομής καθυστερεί από τους κόμβους και έτσι αυτό επιβαρύνει λιγότερο το δίκτυο αφού δεν γίνονται αναμεταδόσεις καλύτερων μονοπατιών που συνήθως φτάνουν την ίδια χρονική στιγμή και έχουνε το ίδιο ακολουθιακό αριθμό.

Το DSDV πρωτόκολλο απαιτεί από κάθε κόμβο να δείχνει, σε κάθε γειτονικό του κόμβο, το δικό του πίνακα δρομολόγησης. Αυτό πρέπει να γίνεται αρκετά συχνά για να είναι σίγουρο ότι κάθε κόμβος μπορεί να εντοπίσει ανά πάσα στιγμή κάποιο κόμβο. Σε αντίθεση κάθε κόμβος μοιράζει πληροφορίες σε άλλους κόμβους μετά από αίτημα. Αυτό βοηθάει στο να προσδιορίζει το συντομότερο μονοπάτι για ένα προορισμό. Με αυτό επίσης αποφεύγουμε να ενοχλούμε κόμβους που είναι σε sleep mode. Έτσι μπορούν να ανταλλαγούν πακέτα μεταξύ δύο κόμβων ακόμα και αν ο ένας κόμβος είναι εκτός εμβέλειας για άμεση επικοινωνία.

2.2.1.1. Πίνακες δρομολόγησης

Τα δεδομένα που στέλνει ο κάθε κόμβος περιέχει τον νέο ακολουθιακό αριθμό και τις παρακάτω πληροφορίες:

- Τη διεύθυνση προορισμού
- Τον αριθμό των βημάτων(hops) για να φτάσει στον προορισμό
- Τον ακολουθιακό αριθμό από τις πληροφορίες που έλαβε για τον προορισμό

Στα headers κάθε πακέτου, οι πίνακες δρομολόγησης περιέχουν την διεύθυνση του υλικού και όπου είναι απαραίτητο την διεύθυνση δικτύου για κάθε κόμβο που μετέδωσε. Επίσης, οι πίνακες δρομολόγησης περιέχουν έναν αριθμό δρομολόγησης που έχει δημιουργηθεί από τον αποστολέα.

Οι ασύρματες συσκευές διαφέρουν από τα παραδοσιακά ασύρματα δίκτυα. Το να λάβει ένας κόμβος ένα πακέτο από κάποιον άλλο δεν δείχνει την ύπαρξη κάποιας διαδρομής ενός βήματος(one hop) για να απαντήσει. Για να το αποφύγουν αυτό, κανένας κόμβος δεν μπορεί να βάλει στο πίνακα πληροφορίες που έχει λάβει από κάποιον

γειτονικό κόμβο, μόνο εκτός εάν αυτός ο γειτονικός κόμβος δείξει ότι μπορεί να λάβει πακέτα από τον άλλο κόμβο.

Ένας από τους πιο σημαντικούς παραμέτρους είναι η επιλογή μεταξύ της χρονικής στιγμής που γίνεται η μετάδοση των πακέτων με τις πληροφορίες δρομολόγησης. Παρόλα αυτά όταν ληφθεί μία πληροφορία δρομολόγησης από ένα κόμβο, αυτή θα αποσταλεί ξανά αμέσως, αλλάζοντας πολύ γρήγορα την πιθανή διάδοση πληροφοριών μεταξύ των συνεργάσιμων κόμβων. Αυτές οι γρήγορες αναμεταδόσεις είναι άλλη μία πρόκληση για τέτοια πρωτόκολλα όπου θα υπήρχαν δραματικές επιπτώσεις στο δίκτυο με την κίνηση κάποιου κόμβου αφού θα προκαλούσε χιλιάδες αναμεταδόσεις.

2.2.1.2. Αντίδραση όταν η τροπολογία αλλάζει

Όπως έχουμε αναφέρει η κίνηση των κόμβων από σημείο σε σημείο καταστρέφουν τα μονοπάτια. Ένα χαλασμένο μονοπάτι μπορεί να εντοπιστεί από κάποιο πρωτόκολλο του δεύτερου επιπέδου ή μπορεί να αναφερθεί αν δεν έχει λάβει κάποια μετάδοση μετά από ένα χρονικό διάστημα κάποιος γειτονικός κόμβος. Όταν εντοπιστεί κάτι τέτοιο ο κόμβος αυτός χαρακτηρίζεται με το σύμβολο ∞ και ένα νέο ακολουθιακό αριθμό. Η μόνη περίπτωση όπου ο ακολουθιακός αριθμός δημιουργείτε από οποιονδήποτε κόμβο πέρα από τον τελικό κόμβο, είναι όταν ένας κόμβος θέλει να μεταδώσει μία πληροφορία για ένα χαλασμένο link. Όταν ένας κόμβος λάβει πληροφορία με αυτό το σύμβολο ∞ το συγκρίνει με ίσο ή μικρότερο αριθμό δρομολόγησης, εάν βρεθεί αμέσως μεταδίδει ένα μήνυμα με τις νέες πληροφορίες για τους προορισμούς.

Σε ένα δίκτυο με πολλούς κόμβους, για να μπορεί να γίνει η δρομολόγηση των πακέτων θα πρέπει οι κόμβοι να έχουν τους πίνακες ενημερωμένους. Αυτό για να συμβεί χρειάζεται ανά τακτά διαστήματα να ενημερώνονται οι πίνακες, που απαιτεί μία σημαντική επιβάρυνση για το δίκτυο. Για να μειωθεί η ποσότητα της πληροφορίας χρησιμοποιεί δύο τύπους ενημερώσεων. Η μία ονομάζεται πλήρη ενημέρωση(full dump) και την αυξητική(incremental) ενημέρωση. Η δεύτερη είναι αυτή που χρησιμοποιείται περισσότερο γιατί στην ουσία απλά ενημερώνει αυτά που συνέβησαν στη προηγούμενη πλήρη ενημέρωση.

2.2.1.3. Κριτήρια επιλογής διαδρομής

Όταν ένας κόμβος λάβει ένα πακέτο ενημέρωσης, αυτό θα συγκρίθεί με τις πληροφορίες που είναι ήδη διαθέσιμες από προηγούμενα πακέτα. Η διαδρομή με πιο μικρό χρονικά αριθμό δρομολόγησης χρησιμοποιείται. Μία διαδρομή που έχει αριθμό δρομολόγησης ίδιο με κάποια άλλη, επιλέγεται αυτή με τα λιγότερα βήματα(hops). Οι διαδρομές που βρέθηκαν πιο πρόσφατα ετοιμάζονται για να σταλούν άμεσα στους γειτονικούς κόμβους.

Ένας κόμβος μπορεί πάντα να λάβει δύο διαδρομές για τον ίδιο προορισμό, με πιο πρόσφατο αριθμό δρομολόγησης, το ένα αμέσως μετά το άλλο, αλλά πάντα ελένχει

τη διαδρομή με τη χειρότερη μετρική. Αυτό θα οδηγήσει στην συνεχής αναμετάδοση των νέων διαδρομών για κάθε νέο αριθμό δρομολόγησης.

Μία λύση όπως αναφέραμε είναι να καθυστερεί η μετάδοση από τον κόμβο όταν μία καλύτερη διαδρομή είναι πιθανόν να εμφανιστεί σύντομα. Η διαδρομή αυτή θα πρέπει να είναι διαθέσιμη για χρήση, αλλά δεν χρειάζεται να μεταδοθεί αμέσως εκτός αν πρόκειται για διαδρομή που δεν έχει βρεθεί μονοπάτι από πριν.

2.2.2. OLSR

Το olsr είναι ένα πρωτόκολλο για ad hoc δίκτυα που ανήκει στη κατηγορία των proactive πρωτοκόλλων. Όπως έχουμε αναφέρει και παραπάνω για τη συγκεκριμένη κατηγορία έτσι και το olsr γνωρίζει όλες τις διαδρομές έτσι ανά πάσα στιγμή μπορεί να γνωρίζει όποια του ζητηθεί. Σε ένα link state πρωτόκολλο όλες οι συνδέσεις μεταξύ των γειτονικών κόμβων είναι γνωστές και μοιράζονται σε όλο το δίκτυο. Το olsr στην ουσία είναι ένα link state πρωτόκολλο προσαρμοσμένο για ad hoc δίκτυα. Αρχικά, μειώνει το μέγεθος των πακέτων ελένχου, δηλαδή αντί για όλες τις διαδρομές, επιλέγει κάποιες διαδρομές με έναν κόμβο να είναι υπεύθυνος και για τους γειτονικούς(multipoint relay selectors(2.2.2.1)) Στη συνέχεια, χρησιμοποιώντας αυτούς τους κόμβους(Multipoint relays) επιβαρύνει λιγότερο το δίκτυο, αφού μόνο οι συγκεκριμένοι που ανήκουν σε αυτόν τον κόμβο μπορούν να αναμεταδώσουν δικά του μηνύματα.

Το συγκεκριμένο πρωτόκολλο επίσης επιβαρύνει λιγότερο το δίκτυο όσον αφορά ένα σφάλμα σε μία διαδρομή. Το olsr διατηρεί όλες τις διαδρομές για όλους τους προορισμούς στο δίκτυο και αυτό βοηθάει όταν επικοινωνούν πολλοί κόμβοι μεταξύ τους ενώ την ίδια χρονική στιγμή ο αρχικός και ο τελικός κόμβος κινούνται. Το olsr, εάν η παραπάνω τεχνική λειτουργήσει σωστά(multipoint relays), είναι κατάλληλο για δίκτυα με μεγάλο αριθμό κόμβων.

Ένα άλλο χαρακτηριστικό είναι ότι λειτουργεί αυτόνομα και δεν χρειάζεται μία σταθερή σύνδεση για να μεταδώσει τα πακέτα ελένχου. Δηλαδή κάθε κόμβος στέλνει τα δικά του πακέτα ελένχου ανά διαστήματα, αυτό σημαίνει ότι κάποια πακέτα μπορεί και να χαθούν, που συμβαίνει πολύ συχνά λόγω προβλήματος στη διαδρομή αυτή ή λόγω άλλων προβλημάτων στη μετάδοση. Επίσης, τα πακέτα που στέλνονται δεν χρειάζεται να φτάνουν στον δέκτη με τη σειρά. Κάθε πακέτο περιέχει έναν αριθμό(sequence number), όπου βοηθάει όταν φτάσουν στο τελικό δέκτη να αναγνωριστούν από αυτό τον αριθμό και έτσι δεν γίνεται ένα παλιό πακέτο να αναγνωριστεί σαν καινούριο.

Κάθε κόμβος χρησιμοποιεί τις τελευταίες ενημερώσεις ώστε να δρομολογήσει ένα πακέτο. Όταν ένας κόμβος κινείται, τα πακέτα μπορούν να παραδοθούν σε αυτό, εάν η ταχύτητα του είναι τέτοια ώστε οι γειτονικοί κόμβοι να μπορούν να ακολουθήσουν την κίνηση του τουλάχιστον.

2.2.2.1. Multipoint relays

Όπως αναφέραμε και νωρίτερα στόχος της συγκεκριμένης λειτουργίας είναι να μειώσει να επιβαρύνει το δίκτυο στέλνοντας διαρκώς πακέτα ή κάνοντας αναμεταδόσεις στην ίδια περιοχή. Κάθε κόμβος στο δίκτυο επιλέγει ένα αριθμό γειτονικών κόμβων, οι

οποίοι αναμεταδίδουν τα πακέτα που θέλει να στείλει. Αυτό το σύνολο κόμβων ονομάζονται multipoint relays(MPRs) του αρχικού κόμβου. Οι γειτονικοί κόμβοι ενός κόμβου Ν οι οποίοι δεν ανήκουν σε αυτό το σύνολο(MPR), λαμβάνουν και διαβάζουν αυτό το πακέτο αλλά δεν το προωθούν. Για αυτό το σκοπό ο κάθε κόμβος Ν έχει ένα σύνολο κόμβων(MPR selectors), κάθε πακέτο που προέρχεται από αυτό το σύνολο κάποιου κόμβου Ν θεωρείται ότι θα μεταδοθεί από αυτό το κόμβο. Αυτό μπορεί να τροποποιηθεί σε βάθος χρόνου, το οποίο υποδεικνύεται στα πακέτα αναγνώρισης (Hello messages) των παραπάνω κόμβων.

Κάθε κόμβος επιλέγει αυτό το σύνολο(multipoint relay) από τους κόμβους που απέχουν από αυτόν ένα βήμα(hop) έτσι ώστε να καλύπτουν όλους τους κόμβους που βρίσκονται δύο βήματα(hops) μακριά. Όσο μικρότερο είναι αυτό το σύνολο κόμβων, τόσο καλύτερα λειτουργεί και το πρωτόκολλο.

2.2.2.2. Εντοπισμός γειτονικών κόμβων

Κάθε κόμβος πρέπει να εντοπίζει τους γειτονικούς κόμβους με τους οποίους θα έχει αμφίδρομη επικοινωνία. Για να ελεγχθεί αυτό το κριτήριο κάθε κόμβος πρέπει να μεταδίδει ανά κάποιες χρονικές στιγμές πακέτα(HELLO μηνύματα), που περιέχουν πληροφορίες σχετικά με γειτονικούς κόμβους αλλά και τη κατάσταση που βρίσκονται τα μονοπάτια αυτά.

Ένα μήνυμα αναγνώρισης(hello message) περιέχει αρχικά μια λίστα διευθύνσεων των γειτονικών κόμβων με διπλής κατεύθυνσης μονοπάτια που είναι ενεργά. Στη συνέχεια περιέχει μία λίστα διευθύνσεων των γειτονικών κόμβων από τους οποίους έχει λάβει ένα μήνυμα αναγνώρισης αλλά δεν γνωρίζει αν η διαδρομή είναι διπλής κατεύθυνσης. Εάν ο κόμβος βρει την διεύθυνση του σε ένα μήνυμα αναγνώρισης τότε θεωρεί το μονοπάτι ως προς τον αποστολέα σαν διπλής κατεύθυνσης.

2.2.2.3. Πίνακες Δρομολόγησης

Κάθε κόμβος περιέχει ένα πίνακα δρομολόγησης που του επιτρέπει να γνωρίζει πως μπορεί να στείλει κάποια πακέτα σε άλλους προορισμούς στο δίκτυο. Τα στοιχεία στο πίνακα αποτελούνται από τη διεύθυνση προορισμού, το επόμενο βήμα(hop) για ένα προορισμό και μία εκτίμηση για την απόσταση σε αυτό το προορισμό. Οι εγγραφές καταγράφονται στο πίνακα για κάθε προορισμό για το οποίο η διαδρομή είναι γνωστή. Για προορισμούς στους οποίους η διαδρομή έχει καταστραφεί ή δεν είναι εντελώς γνωστή δεν εισάγετε στο πίνακα. Ένας πίνακας δρομολόγησης βασίζεται στις πληροφορίες που περιέχει ένας γειτονικός πίνακας και στο πίνακα που περιέχει τοπολογίες. Εάν κάποιος από αυτούς τους πίνακες αλλάξει ο πίνακας δρομολόγησης υπολογίζεται από την αρχή ώστε να ενημερώσει τις πληροφορίες δρομολόγησης. Ο πίνακας υπολογίζεται από την αρχή αν υπάρξει αλλαγή όσον αφορά μια διπλή διαδρομή(bidirectional) ή εάν μία διαδρομή για κάποιο κόμβο δεν υπάρχει πια. Για αυτήν την διαδικασία δεν μεταδίδονται πακέτα σε όλο το δίκτυο αλλά ούτε και στους γειτονικούς κόμβους που απέχουν ένα βήμα(hop).

Η διαδικασία αυτή στην ουσία είναι η εξής. Αρχικά όλες οι εγγραφές του πίνακα δρομολόγησης διαγράφονται. Στη συνέχεια εισάγονται καινούριες εγγραφές ξεκινώντας από εκείνες που οι γειτονικοί κόμβοι είναι ένα βήμα μακριά. Για κάθε γειτονικό κόμβο που η διαδρομή δεν είναι αμφίδρομη, δημιουργείται μία καινούρια εγγραφή στο πίνακα δρομολόγησης όπου ο προορισμός και η διεύθυνση του επόμενου βήματος παίρνουν την τιμή 1. Στη συνέχεια γίνεται η ίδια διαδικασία για τους κόμβους που βρίσκονται ένα βήμα παραπάνω και γενικότερα συνεχίζεται η ίδια διαδικασία μέχρι να μην υπάρχει καινούρια εγγραφή. Τέλος αφού έχει υπολογιστεί ο πίνακας δρομολόγησης, οι εγγραφές που δεν χρησιμοποιούνται για τον υπολογισμό των διαδρομών διαγράφονται, εάν υπάρχει ανάγκη για εξοικονόμηση αποθηκευτικού χώρου. Αλλιώς, αυτές οι εγγραφές μπορεί να προσφέρουν πολλαπλές διαδρομές.

2.3. Reactive πρωτόκολλα

2.3.1. AODV

2.3.1.1. Γενικά

Το πρωτόκολλο δρομολόγησης AODV(Ad Hoc On-Demand Distance Vector) παρέχει γρήγορη και σταθερή επικοινωνία μεταξύ δύο κόμβων και έχει σχεδιαστεί ειδικά για ad hoc δίκτυα με την ελάχιστη χρονικά εύρεση βέλτιστης διαδρομής.

Στηρίζεται στη λογική του DSDV αλγόριθμου αλλά στόχος του είναι να μειώσει την επιβάρυνση του δικτύου όταν δύο κόμβοι θέλουν να έρθουν σε επικοινωνία. Επίσης στο dsdv εάν γινόταν κάποια τοπική αλλαγή επηρεαζόταν ολόκληρη η τοπολογία του δικτύου, αντίθετα στο aodv η μόνη περίπτωση να επηρεαστεί ολόκληρο το δίκτυο είναι εάν κάποιος απομακρυσμένος κόμβος χρησιμοποιήσει ένα μονοπάτι που δεν λειτουργεί. Στον αλγόριθμο aodv είναι πιο προσεκτικό και αναγνωρίζει τον έναν ή περισσότερους κόμβους που έχουνε χρησιμοποιήσει αυτό το μονοπάτι. Σε αυτή τη περίπτωση και μόνο ενημερώνονται μόνο οι συγκεκριμένοι κόμβοι που την χρησιμοποιούν. Τέλος σε μία άλλη συχνή περίπτωση όπου μία διαδρομή έχει αδρανοποιηθεί, δεν επιβαρύνεται το δίκτυο με κάποιο πακέτο εύρεσης νέας διαδρομής. Αυτό που συμβαίνει είναι ότι όποτε κάποιες διαδρομές δεν χρησιμοποιούνται στην ουσία καταργούνται, αυτό μειώνει τον αριθμό των μονοπατιών που δεν είναι σταθερά και παλιά χρονικά αλλά και την επιβάρυνση του δικτύου αφού δεν χρειάζεται να ελένχει τη κατάσταση όλων των διαδρομών.

Άλλο ένα θετικό χαρακτηριστικό σε σχέση με το dsdv είναι και η ενσωματωμένη διαχείριση που έχει όσον αφορά την πολλαπλή δρομολόγηση(multicast). Κάθε διαδρομή έχει ένα χαρακτηριστικό αριθμό και έτσι αποφεύγει να δημιουργήσει λούπες στη προσπάθεια του να δημιουργηθεί ένα multicast δέντρο.

2.3.1.2. Χαρακτηριστικά

Ένα βασικό χαρακτηριστικό του aodv είναι ότι δεν προσπαθεί να είναι δημιουργήσει διαδρομές για κάθε κόμβο προς όλους τους υπόλοιπους κόμβους στο δίκτυο. Οι διαδρομές δημιουργούνται εφόσον χρειάζονται και διαρκούν μόνο όσο είναι απαραίτητες. Το aodv αποφεύγει να επιβαρύνει το δίκτυο στέλνοντας διαρκώς πακέτα ακόμα και όταν προσπαθεί να επιδιορθώσει κάποιο ελαττωματικό μονοπάτι. Αυτό γίνεται δίνοντας σε κάθε μονοπάτι ένα χαρακτηριστικό αριθμό που αυξάνεται μόνο όταν μαθαίνει για κάποια αλλαγή στο γειτονικό του δίκτυο. Επίσης αυτό εξασφαλίζει ότι χρησιμοποιείται πάντα η πιο πρόσφατη χρονικά διαδρομή όταν γίνεται εύρεση μονοπατιού.

Το aodv παρέχει όλων των ειδών επικοινωνία(unicast,multicast, broadcast). Αυτό με τη σειρά του βοηθάει στο να κάνει πιο εύκολο το προγραμματισμό του σε ένα πρωτόκολλο. Το aodv μπορεί να έχει σχεδιαστεί ειδικά για ασύρματα δίκτυα αλλά λειτουργεί εξίσου και σε ενσύρματα δίκτυα.

Οι πίνακες δρομολόγησης στο aodv χρησιμοποιούνται για να αποθηκεύσουν το προορισμό και τις διευθύνσεις του επόμενου βήματος όπως και το χαρακτηριστικό αριθμό για κάθε προορισμό. Επίσης, για κάθε προορισμό ο κάθε κόμβος αποθηκεύει μια λίστα από κόμβους, μέσα από την οποία φτάνει στο κάθε προορισμό. Αυτή η λίστα διατηρείται με σκοπό να βοηθήσει πιθανό έλενχο της διαδρομής εάν παρουσιάσει κάποιο πρόβλημα. Εάν κάποια διαδρομή δεν έχει χρησιμοποιηθεί όσο καιρό έχει δημιουργηθεί, καταργείται.

Το aodv επιτρέπει στους κόμβους να διατηρούν πληροφορίες δρομολόγησης μόνο για το επόμενο βήμα(hop), βοηθώντας να μειώσει τον χώρο αποθήκευσης σε κάθε κόμβο. Τέλος δημιουργεί διαδρομές που χρειάζονται τους λιγότερους πόρους στο δίκτυο (χρόνο, εύρος ζώνης, μνήμη).

2.3.1.3. Εύρεση διαδρομής

Η εύρεση διαδρομής στο aodv γίνεται μόνο εάν ζητηθεί κάτι τέτοιο και χρησιμοποιεί την εξής λογική. Αρχικά όταν ένας κόμβος θέλει να βρει μία διαδρομή για ένα προορισμό, στέλνει ένα πακέτο(RREQ). Κάθε κόμβος που γνωρίζει μία διαδρομή, ακόμα και αν είναι και ο τελικός αποδέκτης, απαντάει σε αυτό το πακέτο με ένα καινούριο πακέτο(RREP). Οι πληροφορίες δρομολόγησης που έχουνε προέρθει από αυτά τα πακέτα διατηρούνται από κάθε κόμβο στο πίνακα δρομολόγησης του. Στη συνέχεια εξετάζοντας το χαρακτηριστικό αριθμό κάθε διαδρομής αποκλείονται οι παλιές χρονικά διαδρομές και ταυτόχρονα διαγράφονται.

Πιο συγκεκριμένα όταν ένας κόμβος θέλει να στείλει ένα πακέτο σε κάποιο άλλο κόμβο, ελένχει το πίνακα δρομολόγησης του εάν έχει κάποια διαδρομή σε αυτόν. Εάν συμβαίνει κάτι τέτοιο προωθεί το πακέτο στον επόμενο κόμβο. Εάν όχι, τότε δημιουργεί ένα πακέτο(RREQ). Αυτό το πακέτο περιέχει τη IP διεύθυνση του αποστολέα και το χαρακτηριστικό αριθμό που έχει εκείνη τη στιγμή, όπως επίσης τα ίδια στοιχεία όσον αφορά και τον αποστολέα. Επίσης περιέχει και έναν αριθμό που αφορά την μετάδοση

πακέτων και αυξάνεται κάθε φορά που δημιουργεί ένα RREQ πακέτο. Έτσι κάθε πακέτο είναι μοναδικό και αποστέλλεται αφού ο κόμβος ξεκινήσει κάποιο χρονομετρητή που περιμένει να δεχτεί απάντηση για στο πακέτο που έστειλε.

Όταν ένας κόμβος λάβει αυτό το πακέτο ελένχει εάν έχει λάβει ξανά αυτό το πακέτο μέσω της ip και του χαρακτηριστικού αριθμού μετάδοσης. Κάθε κόμβος διατηρεί στοιχεία για RREQ πακέτα που έχει λάβει για συγκεκριμένο χρονικό διάστημα. Εάν το έχει λάβει ξανά απορρίπτει το πακέτο χωρίς να επηρεάσει το υπόλοιπο δίκτυο.

Για να προωθήσει τώρα αυτό το πακέτο, ο κόμβος δημιουργεί μία εγγραφή με την διαδρομή που έχει κάνει το πακέτο με αντίθετη σειρά. Το πακέτο αυτό περιέχει τα ίδια στοιχεία προσθέτοντας και τα στοιχεία του γειτονικού κόμβου που έλαβε αυτό το πακέτο αλλά και τον αριθμό βημάτων για το προς την πηγή. Έτσι ο τελικός κόμβος στο τέλος θα γνωρίζει πως να στείλει ένα RREP πίσω στον αρχικό κόμβο.

Για να απαντήσει στο RREQ πακέτο πρέπει ο χαρακτηριστικός αριθμός που έχει σχετιστεί με το προορισμό στο πίνακα δρομολόγησης να είναι τουλάχιστον ίσος με αυτόν στο πακέτο. Αυτό βοηθάει στο να μην μεταδίδεται το ίδιο πακέτο ξανά και ξανά στον ίδιο κόμβο εξασφαλίζοντας ότι η διαδρομή που θα λάβει δεν θα είναι ποτέ παλιά αρκετά ώστε να μείνει σε κάποιο ενδιάμεσο κόμβο. Αλλιώς θα είχε απαντήσει ο προηγούμενος κόμβος. Έτσι αποστέλλεται ένα πακέτο RREP αλλιώς προωθεί το πακέτο σε γειτονικούς κόμβους και αυξάνει τον αριθμό βημάτων στο μετρητή του RREQ πακέτου. Συνήθως, ο τελικός αποδέκτης είναι πάντα διαθέσιμος να απαντήσει σε ένα RREQ πακέτο. Εάν χαθεί ένα τέτοιο πακέτο, ο αρχικός κόμβος έχει το δικαίωμα να ξαναδοκιμάσει να ξεκινήσει όλη τη διαδικασία από την αρχή.

Τέλος έχει συγκεκριμένες φορές που μπορεί να το κάνει αυτό που εξαρτάται από το πως έχει ρυθμιστεί το πρωτόκολλο.

2.3.2. DSR

Το DSR(Dynamic Source Routing) είναι ένα απλό και αποτελεσματικό πρωτόκολλο δρομολόγησης σχεδιασμένο ειδικά για ασύρματα Ad hoc δίκτυα κινητών κόμβων. Το πρωτόκολλο αυτό δίνει την δυνατότητα στο δίκτυο να είναι εντελώς αυτόνομο, χωρίς να χρειάζεται κάποια είδους υποδομή. Το πρωτόκολλο περιέχει δύο μηχανισμούς: εύρεση μονοπατιού(Route discovery) και συντήρηση των δρομολογίων(route maintenance), που συνδυάζονται για να μπορούν οι κόμβοι να ανακαλύψουν και να συντηρήσουν μία διαδρομή για τυχαίους προορισμούς στο δίκτυο. Οι πληροφορίες μετά από κάθε επικοινωνία δεν διαγράφεται αλλά αποθηκεύονται στους ενδιάμεσους κόμβους σε μία προσωρινή μνήμη(route cache) για κάποια χρονική στιγμή. Έτσι κάθε κόμβος στην μνήμη του έχει ένα πίνακα δρομολόγησης με διάφορα μονοπάτια προς διάφορους προορισμούς.

Ένα άλλο πλεονέκτημα με την προσωρινή μνήμη είναι ότι μειώνεται ο χρόνος αναζήτησης αφού κάποιος ενδιάμεσος κόμβος θα γνωρίζει τη διαδρομή που θέλει να φτάσει το πακέτο και θα απαντήσει άμεσα. Σε περίπτωση που έχουμε σύνδεσμο μία κατεύθυνσης ο παραλήπτης ανακαλύπτει με τον ίδιο τρόπο την αφετηρία ή χρησιμοποιεί το ίδιο μονοπάτι αντίστροφα.

Το DSR περιέχει και άλλη μία χρήσιμη τεχνική, τα passive acknowledgments. Αυτό που συμβαίνει είναι το εξής, ο κόμβος Α θέλει να στείλει ένα πακέτο. Έστω ότι το

στέλνει στον Β που δεν είναι ο τελικός προορισμός. Αυτό που κάνει είναι ότι ο Α ακούει τη στιγμή που ο Β το μεταδίδει στον Γ και έτσι έχει μία επιβεβαίωση για την προώθηση του πακέτου.

2.3.2.1. Βασικές ιδιότητες του DSR

Όπως αναφέραμε και πριν το DSR έχει δύο βασικούς μηχανισμούς που όταν λειτουργούν μαζί επιτρέπουν την εύρεση και συντήρηση διαδρομών σε ένα ad hoc δίκτυα:

- Εύρεση διαδρομής(Route Discovery) είναι ένας μηχανισμός όπου όταν ένας κόμβος Α θέλει να στείλει ένα πακέτο στο κόμβο Δ αποκτά μία διαδρομή για το Δ. Η εύρεση διαδρομής χρησιμοποιείται μόνο όταν ο Α προσπαθήσει να στείλει ένα πακέτο στον Δ και δεν γνωρίζει ήδη ένα μονοπάτι για το Δ.
- Συντήρηση δρομολογίων(Route Maintenance) είναι ένας μηχανισμός όπου ένας κόμβος Β μπορεί να εντοπίσει κατά τη διάρκεια που χρησιμοποιεί ένα μονοπάτι για τον Ε , εάν η τοπολογία του δικτύου έχει αλλάξει(η διαδρομή για τον κόμβο Ε δεν λειτουργεί γιατί η ακμή(link) δεν λειτουργεί). Όταν βρεθεί κάτι τέτοιο, ο Β μπορεί να προσπαθήσει να χρησιμοποιήσει όποια άλλη διαδρομή μπορεί να ξέρει για τον κόμβο Ε ή μπορεί να βρει μία καινούρια διαδρομή με το μηχανισμό εύρεσης διαδρομής. Ο μηχανισμός συντήρηση δρομολογίων χρησιμοποιείτε μόνο όταν ο κόμβος Β στέλνει πακέτα στο κόμβο Ε.

Ο κάθε ένας από αυτούς τους δύο μηχανισμούς χρησιμοποιείται κατ' απαίτηση. Αντίθετα με τα υπόλοιπα πρωτόκολλα το DSR δεν απαιτεί πακέτα του τύπου ενημέρωσης διαδρομών για τους κόμβους, τη κατάσταση ενός μονοπατιού, ή πακέτα για να εντοπίσει ένας κόμβος τους γειτονικούς του. Έτσι το δίκτυο επιβαρύνεται λιγότερο αφού ο αριθμός των πακέτων μειώνεται σημαντικά. Όταν οι κόμβοι αρχίσουν και κινούνται περισσότερο, το DSR προσαρμόζεται και εντοπίζει μόνο τις διαδρομές που χρησιμοποιούνται εκείνη την στιγμή.

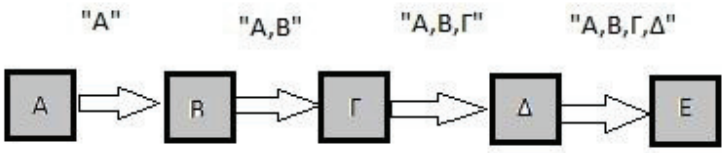
Στα ασύρματα δίκτυα είναι πιθανό μία ακμή δύο κόμβων να μην λειτουργεί σωστά και στις δύο κατευθύνσεις για διάφορους λόγους(κεραία, παρεμβολές). Το DSR έχει σχεδιαστεί έτσι ώστε τέτοιου τύπου μονοπάτια να χρησιμοποιούνται μόνο όταν είναι απαραίτητα. Αυτό βοηθάει την απόδοση και την συνδεσιμότητα στο σύστημα.

2.3.2.2. Εύρεση διαδρομής

Όταν ένας κόμβος Α θέλει να στείλει ένα πακέτο στον κόμβο Δ, τοποθετεί στη επικεφαλίδα(Header) του πακέτου μία διαδρομή που αποτελείται από μία ακολουθία βημάτων δηλαδή κόμβων που πρέπει να περάσει το πακέτο για να φτάσει στο προορισμό του. Συνήθως ο κόμβος Α εάν κοιτάξει την προσωρινή του μνήμη θα βρει κάποιο μονοπάτι, αλλιώς θα καλέσει το μηχανισμό εύρεσης διαδρομής.

Για παράδειγμα στο παρακάτω σχήμα, ο κόμβος Α θέλει να βρει μία διαδρομή για να φτάσει στον Ε. Για να ξεκινήσει ο μηχανισμός εύρεσης διαδρομής, ο Α μεταδίδει ένα RREQ (route request) μήνυμα σε ένα πακέτο το οποίο λαμβάνουν οι περισσότεροι

κόμβοι στο χώρο κάλυψης του κόμβου Α. Κάθε RREQ έχει ένα μοναδικό id όπως επίσης και μία λίστα που περιέχει κάθε ενδιάμεσο κόμβο που προωθήθηκε αυτό το πακέτο.



Εικόνα 7: Εύρεση μονοπατιού Α στον Ε

Όταν ένας κόμβος λάβει ένα RREQ μήνυμα, και είναι ο στόχος από τον route discovery επιστρέφει ένα RREP (Route Reply) μήνυμα με την διαδρομή που καταγράφηκε. Όταν λάβει αυτό το μήνυμα αποθηκεύει τη διαδρομή στη προσωρινή του μνήμη μέχρι να σταλούν τα πακέτα που θέλει. Εάν ένας κόμβος λάβει ένα rreq το οποίο έχει το ίδιο id ή ότι η διεύθυνσή του είναι ήδη στο μονοπάτι, τότε απορρίπτει το rreq.

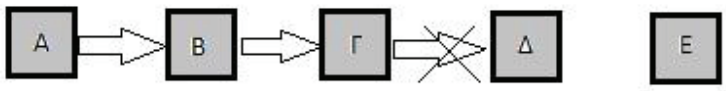
Για να απαντήσει τώρα ο κόμβος Ε στο rrep, αρχικά θα κοιτάξει την δική του μνήμη για να δει εάν υπάρχει κάποια διαδρομή προς τον Α και θα την χρησιμοποιήσει. Αλλιώς θα πρέπει να κάνει τη δική του εύρεση διαδρομής με την προϋπόθεση ότι απαντάει στο δικό του rreq. Άλλη μία δυνατότητα που έχει ο Ε είναι απλά να αντιστρέψει το μήνυμα την ακολουθία των κόμβων από το ιστορικό καταγραφής που θα είχε στο rrep και να χρησιμοποιήσει αυτήν σαν διαδρομή.

Όταν ενεργοποιηθεί ο μηχανισμός της εύρεσης διαδρομής, ο κόμβος που στέλνει δεδομένα δημιουργεί ένα αντίγραφο και το αποθηκεύει τοπικά σε μία μνήμη(send buffer). Αυτή η μνήμη περιέχει ένα αντίγραφο από όποιο πακέτο δεν μπορεί να μεταδοθεί εκείνη τη στιγμή γιατί δεν υπάρχει διαδρομή για το συγκεκριμένο προορισμό. Κάθε πακέτο στην μνήμη έχει έναν χρόνο που εισήλθε στην μνήμη και φεύγει στο τέλος του χρόνου. Ο buffer χρησιμοποιεί τον αλγόριθμο FIFO για να αποφύγει να χάσει πακέτα. Όσο μένει στον buffer ο κόμβος πρέπει να ελένχει για καινούριες βέλτιστες διαδρομές για τον προορισμό των πακέτων. Παρόλα αυτά ο κόμβος πρέπει να το κάνει με συγκεκριμένο ρυθμό αφού ο προορισμός μπορεί να μην είναι εφικτός. Έτσι επιβαρύνουμε και λιγότερο το δίκτυο αφού θα αποσταλούν πολύ λιγότερα πακέτα rreq βάζοντας ένα όριο στο κάθε πόσο μπορεί να γίνετε εύρεση διαδρομής για τον ίδιο προορισμό.

2.3.2.3. Συντήρηση μονοπατιών στο DSR

Όταν ένα πακέτο προωθείται μέσω μίας διαδρομής, κάθε κόμβος που μεταδίδει αυτό το πακέτο είναι υποχρεωμένο να επιβεβαιώσει ότι το πακέτο έχει ληφθεί από τον επόμενο κόμβο της διαδρομής. Το πακέτο θα συνεχίσει να μεταδίδεται(μέχρι ένα αριθμό μεταδόσεων) μέχρι να λάβει αυτήν την επιβεβαίωση. Αυτό γίνεται με το mac πρωτόκολλο η με την τεχνική passive acknowledgments.

Εάν ένα πακέτο ξεπεράσει το επιτρεπτό όριο που έχει μεταδοθεί και δεν έχει λάβει κάποιο μήνυμα επιβεβαίωσης τότε ο κόμβος αυτός στέλνει ένα route error μήνυμα στο αρχικό αποστολέα του πακέτου, αναγνωρίζοντας το μονοπάτι που δεν μπορεί το πακέτο να μεταδοθεί. Όπως στο παρακάτω σχήμα όπου ο κόμβος Γ δεν μπορεί να φτάσει τον Δ οπότε στέλνει στον Α ένα route error μήνυμα. Ο Α θα πρέπει να αντικαταστήσει το μονοπάτι αυτό από την μνήμη του και να βρει κάποιο άλλο για το κόμβο Ε και να το στείλει αμέσως. Διαφορετικά θα πρέπει να κάνει χρήση του μηχανισμού εύρεσης διαδρομής.



Εικόνα 8: Συντήρηση Διαδρομής (Δ εκτός εμβέλειας)

2.3.3 DYMO

Το DYMO είναι ένα πρωτόκολλο που δημιουργήθηκε για περιπτώσεις όπου οι χρήστες κινούνται και για να επικοινωνήσουν θα πρέπει να περάσουν μέσω άλλων χρηστών μέσω ενός ασύρματου ad hoc δικτύου. Όταν ένας κόμβος θέλει να επικοινωνήσει με κάποιον κόμβο, βρίσκεται μία διαδρομή μετά από αίτημα, και αυτό έχει σαν αποτέλεσμα ένα αμφίδρομο μονοπάτι. Το dymo δημιουργήθηκε για να μπορεί να αντιμετωπίζει δυναμικά πιθανές αλλαγές στο δίκτυο.

2.3.3.1. Γενικά

Οι βασικές λειτουργίες του dymo είναι η εύρεση διαδρομής και η διαχείριση μίας διαδρομής. Όταν ένας κόμβος, πηγή, θέλει να στείλει κάποια δεδομένα σε κάποιον κόμβο, παραλήπτη, η πηγή θα στείλει ένα αίτημα διαδρομής (RREQ) για να βρεί μία διαδρομή για το συγκεκριμένο προορισμό. Εάν το Dymo χρησιμοποιηθεί σαν πρωτόκολλο δρομολόγησης στο επίπεδο δικτύου, τα μηνύματα δρομολόγησης ενσωματώνονται στα udp πακέτα. Με το πως δημιουργούνται τα πακέτα στο dymo και την χρήση των ανάλογων διευθύνσεων ο αλγόριθμός μπορεί να χρησιμοποιηθεί σε όλα τα επίπεδα. Ένα πακέτο RREQ θα το λάβουν και οι υπόλοιποι κόμβοι που βρίσκονται εντός εμβέλειας του αρχικού κόμβου και αν δεν είναι οι τελικοί κόμβοι θα προωθήσουν το συγκεκριμένο πακέτο μέχρι να βρεθεί ο προορισμός ή μέχρι να φτάσει το μέγιστο αριθμό βημάτων(hops) και να απορριφτεί. Κάθε κόμβος που αναμεταδίδει αυτό το πακέτο (RREQ) καταγράφει την διαδρομή από τους κόμβους που πέρασε. Έτσι κάθε κόμβος θα έχει αποκτήσει μία διαδρομή για τον αρχικό κόμβο(αποστολέα). Όταν βρεθεί ο τελικός κόμβος, εκείνος απαντάει με ένα πακέτο (RREP) προς την πηγή αντίστοιχα

ακολουθώντας την ίδια διαδρομή. Όταν αυτό το πακέτο φτάσει στην πηγή τότε σημαίνει ότι και όλοι οι ενδιάμεσοι κόμβοι γνωρίζουν πως συνδέονται η πηγή με το παραλήπτη. Εάν όμως κατά τη διάρκεια αποστολής κάποιου πακέτου ένας ενδιάμεσος κόμβος αναγνωρίσει ότι δεν έχει την δυνατότητα να έρθει σε επαφή με τον δέκτη τότε πρέπει να στείλει ένα καινούριο πακέτο (RERR) που όταν το λάβει η αποστολέας θα γνωρίζει ότι θα πρέπει να στείλει ένα καινούριο RREQ πακέτο για να βρεθεί μία καινούρια διαδρομή προς το τελικό αποδέκτη, με την προϋπόθεση ότι υπάρχουν ακόμα πακέτα για να σταλούν μεταξύ τους.

Το Dymo χρησιμοποιεί συγκεκριμένους αριθμούς ώστε να μπορεί να ελένχει αν αυτό που περιέχει ένα πακέτο είναι καλύτερο, βάση το μονοπάτι, σε σχέση με την ήδη υπάρχουσα πληροφορία. Εάν βρεθεί μία τέτοια διαδρομή τότε ο κόμβος θα ενημερώσει το πίνακα δρομολόγησης του με την καινούρια πληροφορία.

2.3.3.2. Πίνακες δρομολόγησης

Ένας κόμβος πρέπει να ελένχει ανά τακτά διαστήματα το επόμενο βήμα (hop) για τους άλλους κόμβους που έχει στο πίνακα δρομολόγησης του. Εάν βρεθεί μία διαδρομή που παρουσιάζει κάποιο πρόβλημα πρέπει να χαρακτηριστεί ως ελαττωματική και να διαγραφεί από τη λίστα με τους πιθανούς κόμβους που μπορεί να προωθήσει πακέτα. Εάν κάποιο πακέτο φτάσει σε κάποιο κόμβο και δεν μπορεί να σταλεί στο επόμενο βήμα(hop) ή η διαδρομή δεν λειτουργεί και ένα πακέτο RERR δημιουργείται και αποστέλλεται στη διεύθυνση αποστολέα. Κάθε κόμβος που επηρεάζεται, π.χ κάποιες εγγραφές στο πίνακα δρομολόγησης που χρησιμοποιούν έναν κόμβο που δεν μπορεί να προσπελαστεί και είναι το επόμενο βήμα , πρέπει να προστεθεί σε αυτό το πακέτο. Όταν ένας κόμβος τώρα λάβει ένα τέτοιο πακέτο, επεξεργάζεται όλες τις διευθύνσεις του μηνύματος και αν η διεύθυνση που εξετάζεται έχει και τα παρακάτω κριτήρια χαρακτηρίζεται σαν χαλασμένη. Πρώτο κριτήριο είναι η διεύθυνση του επόμενου βήματος(hop) του πίνακα δρομολόγησης να είναι ίδια με την ip που έστειλε το RERR πακέτο. Επόμενο κριτήριο είναι η διεπαφή που έλαβε αυτό το πακέτο να είναι η ίδια με αυτήν που είναι το επόμενο βήμα(hop) στο πίνακα δρομολόγησης. Τέλος, ο χαρακτηριστικός αριθμός(sequence) του στο πίνακα δρομολόγησης να είναι μηδέν.

Κεφάλαιο 3: Προσομοιωτές δικτύων

3.1. Γενικά

Οι προσομοιώσεις παίζουν σημαντικό ρόλο στην εξέλιξη και στην αξιολόγηση καινούριων συστημάτων. Για ad hoc δίκτυα, συνηθίζεται να γίνονται προσομοιώσεις γιατί επιλέγουμε συνθήκες που θέλουμε στη τοπολογία και μεγάλο αριθμό κόμβων εάν θέλουμε. Σε πειράματα που γίνονται στην πραγματικότητα, οι συνθήκες που επικρατούν όταν θέλουμε να έχουμε μια ασύρματη επικοινωνία, είναι αδύνατο να μπορεί να γίνουν υπό τις ίδιες συνθήκες. Επίσης, σε έναν προσομοιωτή το πείραμα μπορεί να επαναληφθεί όσες φορές χρειάζεται ενώ το να στήσουμε ένα πείραμα απαιτεί αρκετό χρόνο. Ένα άλλο πρόβλημα που προκύπτει είναι στη περίπτωση που θέλουμε να δοκιμάσουμε κάποια πρωτόκολλα πως αντιδρούν με μεγάλο αριθμό κόμβων, δηλαδή το να ετοιμάσουμε ένα σενάριο με πολλούς κόμβους συνήθως είναι πολύ χρονοβόρο για να μπορούν να πραγματοποιηθούν ρεαλιστικά πειράματα.

Υπάρχουν κάποια βασικά σημεία που χαρακτηρίζουν τα πειράματα σε ad hoc δίκτυα. Αρχικά το μηχάνημα που θα γίνει η προσομοίωση. Το μηχάνημα είναι υπεύθυνο να εκτελέσει σωστά χρονικά το κώδικα. Τα πιο πολλά προγράμματα, ns2, OPNET Modeler, Omnet++ όπως και το Jist/SWANS στηρίζονται σε διακριτά γενονότα με ένα όριο χρόνου. Αυτό το χρόνο, το μηχάνημα αυτό πρέπει να τον πραγματοποιεί σε σύντομο χρονικό διάστημα σε σχέση με το πραγματικό χρόνο του πειράματος. Στη συνέχεια, το περιβάλλον προσομοίωσης σε ένα τέτοιο πρόγραμμα θα πρέπει να δημιουργηθεί ώστε να έχει πολλές λεπτομέρειες, όπως δωμάτια, μεγάλα κτίρια και δρόμους. Τρίτο χαρακτηριστικό είναι να καθορίσουμε το είδος των κόμβων που θα περιλαμβάνει μία προσομοίωση και στο επίσης στο πώς θα αντιδρούν μεταξύ τους. Δηλαδή η αντίδραση των κόμβων αρχικά συνολικά, αυτό περιλαμβάνει να στείλει κάποια πακέτα και να κινηθεί στο χώρο που έχει δημιουργηθεί. Στη συνέχεια, τη σχέση που θα έχει με τους υπόλοιπους κόμβους, με χαρακτηριστικά που υπάρχουν και στην πραγματικότητα, για παράδειγμα το πρωτόκολλο δρομολόγηση που θα χρησιμοποιηθεί, το μηχανισμό που θα χρησιμοποιήσει tcp ή udp. Τελευταίο χαρακτηριστικό που θα πρέπει να έχει απαραίτητα ένα πείραμα είναι αρχεία που καταγράφει την κίνηση των κόμβων και γενικότερα ότι συμβαίνει στο δίκτυο όπως επίσης και αρχεία με διάφορα αποτελέσματα από το πείραμα. Αυτό είναι λογικό από τη στιγμή που κάθε πείραμα μπορεί να περιέχει πάρα πολλούς κόμβους με διαφορετικά χαρακτηριστικά.

Όπως αναφέραμε παραπάνω, τα πειράματα πρέπει να τηρούν δύο προϋποθέσεις: να είναι ακριβή και να τηρούν τις χρονικές απαιτήσεις. Αυτό αποτελεί ένα βασικό πρόβλημα όταν ακολουθούμε αυτό το τρόπο γιατί πρέπει να έχουμε μία ισορροπία μεταξύ αυτών των απαιτήσεων και στο πόσο ρεαλιστικό θα κάνουμε το περιβάλλον για το πείραμα.

Για ad hoc δίκτυα έχουνε δημιουργηθεί ένας μεγάλος αριθμός από εργαλεία όπως και κατάλληλες βιβλιοθήκες. Το πιο διαδεδομένο είναι το ns2, το οποίο έχει χρησιμοποιηθεί και σε αυτήν την εργασία, που είχε δημιουργηθεί για ενσύρματα δίκτυα αρχικά. .Ωστόσο το πανεπιστήμιο CMU(Carnegie Mellon University) δημιούργησε δικές τους επεκτάσεις για ασύρματα δίκτυα. Λόγω ότι έχει δημιουργηθεί πολλά χρόνια, ο ns2 έχει πάρα πολλές επεκτάσεις διαθέσιμες. Παρόλα αυτά ο ns2 έχει ένα βασικό πρόβλημα, χρειάζεται πάρα πολύ χρόνο να εκτελέσει κάποιο πείραμα αλλά και μνήμη, όταν υπάρχουν εκατοντάδες κόμβοι. Γι αυτό το πρόβλημα έχει δημιουργηθεί ένα πακέτο που ονομάζεται PDNS. Σε συνδυασμό με αυτό, ο ns2 έχει δοκιμαστεί και έχει καταφέρει να εκτελέσει πείραμα με εξακόσιες χιλιάδες κόμβους με εκατόν τριανταέξι επεξεργαστές.

Σε σχέση με τον δημοφιλή ns2, υπάρχει το GloMoSim το οποίο είναι πιο αποδοτικό όταν έχουμε βασικό τερματικό και τα πειράματα δεν περιέχουν παραπάνω από δέκα χιλιάδες κόμβους. Το συγκεκριμένο πρόγραμμα βασίζεται στο Parsec, μία γλώσσα προγραμματισμού που μοιάζει με C για διακριτά γεγονότα. Περιέχει βιβλιοθήκη ειδικά για ad hoc δίκτυα αλλά όχι τόσο ενημερωμένη όσο ο ns2. Μία παρόμοια έκδοση είναι το QualNet το οποίο δεν είναι δωρεάν και οι δημιουργεί του αναφέρουν ότι έχει την δυνατότητα να αντεπεξέλθει για παραπάνω από δέκα χιλιάδες κόμβους.

Άλλο ένα πρόγραμμα είναι το Opnet το οποίο δεν είναι διαθέσιμο δωρεάν. Δίνει την δυνατότητα στο χρήστη να δημιουργεί και να διαχειρίζεται πειράματα χρησιμοποιώντας ένα σύνολο από γραφικά, διαγράμματα και C++. Σε σχέση με τον ns2 και το GloMoSim, τα πειράματα πριν από την εκτέλεση, «συντάσσεται» σε ειδικό εκτελέσιμο αρχείο. Σε σχέση με τις βιβλιοθήκες του αλλά και την απόδοση του είναι παρόμοιο με τον ns2.

Ακολουθεί ένας πίνακας με τα πιο δημοφιλή προγράμματα:

No.	Προσομοιωτής	Γλώσσα προγραμματισμού	Πλεονεκτήματα	Μειονεκτήματα
1	Ns-2	C++	• Εύκολο στην προσθήκη νέων πρωτοκόλλων • Πρόγραμμα παρουσίασης • Μεγάλος αριθμός πρωτοκόλλων	• Διαθέτει μόνο δύο ασύρματα MAC πρωτόκολλα 802.11 και TDMA
2	TOSSIM	nesC	• Υψηλό βαθμό ακρίβειας • Πρόγραμμα παρουσίασης	• Χάνει χρονικά (compilation) • Διακόπτει τις ιδιότητες του κώδικα
3	GloMoSim	Parsec	• Δυνατότητα παράλληλης προσομοίωσης • Ειδικά για ασύρματα	• Μη διαθέσιμα νέα πρωτόκολλα

			δίκτυα • Πρόγραμμα παρουσίασης	
4	UWSim	C++	• Διαθέσιμο δωρεάν • Αποκλειστικά για υποβρύχια συστήματα	• Συγκεκριμένες λειτουργίες • Προκαλεί extension
5	Avrora	Java	• Μπορεί να διαχειριστεί πάνω από 10000 κόμβους	Δεν υποστηρίζει κίνηση στους κόμβους
6	SENS	C++	• Ανεξάρτητο πλατφόρμας • Συγκεκριμένο περιβάλλον απο χρήστη	• Υποστηρίζει αισθητήρες και φυσικά φαινόμενα μόνο τον ήχο.
7	COOJA	Java (Πειράματα σε C)	• Υπολογίζει και υλικό και λογισμικό • Αλγόριθμοι και πρωτόκολλα μεγαλύτερης κλίμακας	• Δεν είναι αποτελεσματικό • Υποστηρίζει έναν συγκεκριμένο τύπο κόμβων
8	Castalia	C++	• Εκτελούνται πολλά πρωτόκολλα δρομολόγησης • Δυνατότητα επεξεργασίας.	• Δεν είναι αποκλειστικά για αισθητήρες
9	Shawn	Java	• Δεν περιορίζεται σε κατανεμημένα πρωτόκολλα • Δυνατότητα προσομοίωσης μεγάλων δικτύων.	• Δεν περιέχει αναλυτικά διάφορα στοιχεία σε ένα πείραμα(radio propagation)
10	EmStar	Linux	• Συνδυασμός προσομοιωτή και εξομοιωτή • Μπορεί να εκτελεστεί σε ένα σύνολο από πλατφόρμες.	• Για συγκεκριμένους τύπου κόμβους • Δεν υποστηρίζει παράλληλη προσομοίωση • Όχι τόσο ακριβή και γρήγορο
11	J-Sim	Java	• Υποστηρίζει κινητά ασύρματα δίκτυα και δίκτυα αισθητήρων • Αντικειμενοστραφή αρχιτεκτονική	• Μικρή ακρίβεια • Υποστηρίζει μόνο 802.11
12	SENCE	C++	• Μικρή ποσότητα μνήμης	• Δεν είναι ακριβής αξιολόγηση

			• Γρήγορο • Επεκτάσεις	όσον αφορά ασύρματα δίκτυα • Απουσία προγράμματος παρουσίασης
13	VisualSense	Ptolemy II	• Ακριβές μοντέλο	• Έλλειψη πρωτοκόλλων
14	(J)Prowler	Matlab/ Java	• Ειδικά για αισθητήρες	• Παρέχει μόνο ένα MAC πρωτόκολλο (TinyOs)

Πίνακας 1: Προγράμματα προσομοίωσης και χαρακτηριστικά

Το τελευταίο χρονικό διάστημα έχει γίνει ιδιαίτερα δημοφιλή το JiST, το οποίο στηρίζεται σε Java, μετά τη βελτίωση που έγινε από το πανεπιστήμιο Cornell. Παρόλα αυτά, το συγκεκριμένο πρόγραμμα βρίσκεται ακόμα σε αρχικό στάδιο και τα διαθέσιμα χαρακτηριστικά του είναι λίγα, αλλά το ενδιαφέρον για το συγκεκριμένο πρόγραμμα όλο και αυξάνεται λόγω της καλύτερης απόδοσης του.

Μετά από μελέτη, καταλήξαμε σε κάποιες βασικές διαφορές στα χαρακτηριστικά τους. Η πρώτη προφανή διαφορά μεταξύ του ns2 και του JiST είναι το πως προσεγγίζουν τη λογική της προσομοίωσης. Ο ns2 παρέχει ενδιαφέρον δυνατότητες χάρη στο χαρακτηριστικό που έχει με την tcl, δηλαδή την ικανότητα που μας δίνει να προσεγγίσουμε ένα αντικείμενο μόνο με C++. Το JiST περιέχει μια εικονική μηχανή Java που βασίζεται στον προσομοιωτή. Η ιδέα μίας τέτοιας μηχανής και του ανάλογου κώδικα δεν χρειάζεται καμία υποστήριξη από ειδικά πρωτόκολλα ή να μπαίνει στην διαδικασία αλλαγής των γεγονότων για να τρέξει μία προσομοίωση. Άλλα θετικά χαρακτηριστικά είναι η διαχείριση των σκουπιδιών από τη μνήμη, υποστήριξη όσον αφορά την χρήση της πλατφόρμας και πιθανότητες στο μέλλον να ανακαλυφθούν, μετά από βελτιώσεις, αρχιτεκτονικές που δεν υπάρχουνε αυτή τη στιγμή.

Η διαχείριση μνήμης είναι ένα σημείο όπου το JiST όχι μόνο πλεονεκτεί σε σχέση με άλλα παρόμοια προγράμματα, π.χ έχει παράλληλη συλλογή σκουπιδιών από την μνήμη αλλά επίσης παρέχει στο χρήστη την δυνατότητα να μειώσει τους πόρους που χρειάζονται σε μία προσομοίωση. Έτσι υπάρχουνε κάποια αντικείμενα(timeless) που ανταλλάσσονται χωρίς να χρειάζεται να τα κλωνοποιήθουν όπως τον ns2. Βέβαια μπορεί να κερδίσουμε αρκετή μνήμη με αυτό αλλά είναι αρκετά ριψοκίνδυνη απόφαση γιατί είναι υπό την ευθύνη αυτού που κάνει το πείραμα να ξεκαθαρίσει ότι πρόκειται για τέτοιου τύπου μεταβλητή ώστε να εξασφαλιστεί η ακεραιότητα στον χρόνο στο πείραμα. Το πακέτο JiST/SWANS έχει ένα άλλο επίσης χαρακτηριστικό οι θέσεις των κόμβων στο χώρο υπολογίζεται μόνο όταν χρειαστεί. Έτσι θα μπορούσαν να δημιουργηθούν τεχνικές που θα μειώνουν τον αριθμό που υπολογίζει τις ενημερώσεις όσον αφορά τις θέσεις σε ένα χώρο. Επίσης, στον ns2 οι κόμβοι από μόνοι τους μπαίνουν σε ένα κανάλι ώστε να ακούνε όταν κάποιος κόμβος θέλει να στείλει ένα πακέτο. Στο SWANS υπάρχει μία τεχνική που περιορίζει τον αριθμό των κόμβων που πρέπει να υπολογίσουν την ισχύ του σήματος τους ως τον αποστολέα σε σχέση με αυτούς που βρίσκονται πραγματικά μέσα σε αυτή τη περιοχή.

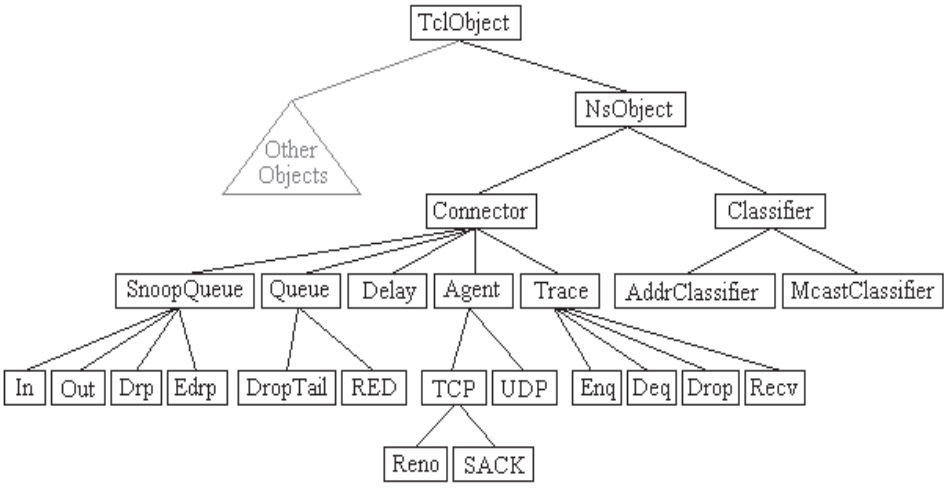
Γενικότερα, το JiST μπορεί να είναι η μόνη επιλογή για μεγάλα σενάρια. Ωστόσο όπως είπαμε πρόκειται για ένα πρόγραμμα που αναπτύσσεται τελευταία, οπότε η

αξιολόγηση του όσον αφορά την εφαρμοσιμοτητά του σε τέτοια δίκτυα δεν μπορεί ακόμα να εκτιμηθεί σε σχέση με τον ns2, αλλά θα μπορούσε να είναι μία μελλοντική μελέτη.

3.2. Περιγραφή του NS2

Ο ns2(Network Simulator version 2) είναι ένα απλό εργαλείο προσομοίωσης που στηρίζει τη λειτουργία του σε διακριτά γεγονότα και έχει αποδειχθεί πάρα πολύ χρήσιμο για την μελέτη αρκετών δικτύων επικοινωνίας. Το πρόγραμμα δίνει την δυνατότητα στο χρήστη να κάνει προσομοίωση σταθερού δικτύου αλλά και ασύρματου όπως επίσης και πρωτοκόλλων(αλγόριθμοι δρομολόγησης, tcp, udp). Γενικά, ο ns παρέχει στους χρήστες ένα τρόπο να ρυθμίζει τέτοιου τύπου πρωτόκολλα και να δείχνει πως αντιδρούν.

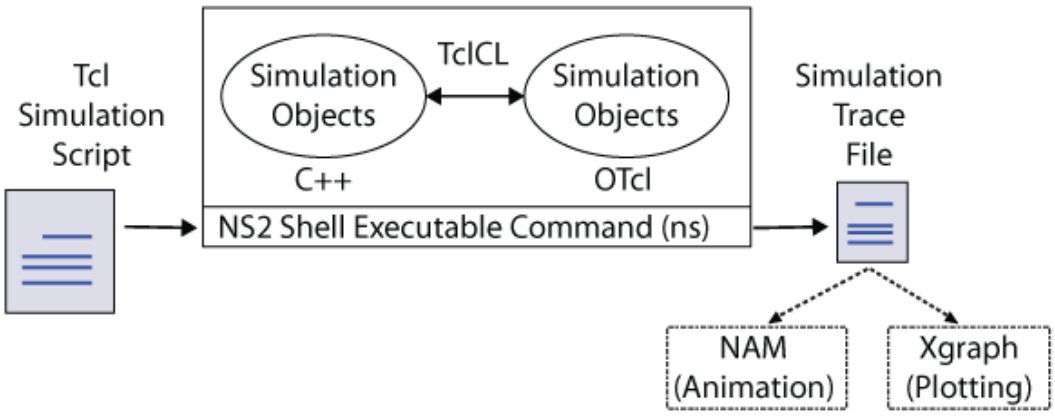
Ο NS βρίσκεται σε συνεχή εξέλιξη και βελτίωση καθώς όλο και περισσότερες δυνατότητες και χαρακτηριστικά προστίθενται. Επίσης πρέπει να σημειωθεί ότι στην εξέλιξη αυτή συμβάλλουν διάφορα εκπαιδευτικά ιδρύματα (π.χ. το ISI – Information Sciences Institute, www.isi.edu), όπως και απλοί χρήστες με τις παρατηρήσεις τους ή με ενεργό συμμετοχή στη δημιουργία ή στη βελτίωση των διάφορων συναρτήσεων. Ακόμα και ο χρήστης για τη δική του εφαρμογή μπορεί να δημιουργήσει καινούριους τύπους αντικειμένων (π.χ. agents με συγκεκριμένα χαρακτηριστικά και δυνατότητες).



Εικόνα 9: Αντικείμενα στον NS

3.3. Βασική αρχιτεκτονική

Όπως ήδη προαναφέρθηκε ο ns στηρίζεται σε διακριτά γεγονότα. Αυτό σημαίνει ότι ο χρήστης προκειμένου να φτιάξει ένα πρόγραμμα που θα το τρέξει με τη βοήθεια του NS και θα βγάλει τα αποτελέσματα που χρειάζεται πρέπει να δώσει ιδιαίτερη έμφαση στα γεγονότα και κυρίως πως αυτά εισάγονται στον NS. Για τη δημιουργία της τοπολογίας και των διάφορων γεγονότων ο NS υποστηρίζει τις γλώσσες προγραμματισμού OTCL (Object TCL) και C++ σε συνεργασία μεταξύ τους (εικόνα 10).



Εικόνα 2: Βασική αρχιτεκτονική

Πιο συγκεκριμένα η OTCL χρησιμοποιείται για τη δημιουργία της τοπολογίας, για την δημιουργία κάποιων αντικειμένων επί της τοπολογίας, καθώς και για την εν δυνάμει αλλαγή των συνθηκών σε κάποιες χρονικές στιγμές της προσομοίωσης. Αντίθετα η C++ χρησιμοποιείται για τον χειρισμό των πακέτων και για το χειρισμό αντικειμένων τα οποία δεν έχουν άμεση σχέση με τη συγκεκριμένη τοπολογία. Σε γενικές γραμμές η C++ προσφέρει πολλές περισσότερες δυνατότητες – λόγω βιβλιοθηκών, ύπαρξη πιο πλούσιων δομών – από ότι η OTCL και είναι πιο γρήγορη στην εκτέλεση. Όμως η OTCL έχει τη δυνατότητα να επικοινωνεί καλύτερα με τον NS και να προσομοιώνει τη δημιουργία των διάφορων γεγονότων όποτε εμείς θέλουμε. Επίσης υπάρχουν κάποιες έτοιμες συναρτήσεις αποκλειστικά για τον NS (π.χ. δημιουργία κόμβων, αντιπροσώπων – agents), ενώ η δυνατότητα να «τρέχει» των κώδικα δυναμικά – ανάλογα με τη δεδομένη κατάσταση του δικτύου είναι ιδιαίτερη χρήσιμη.

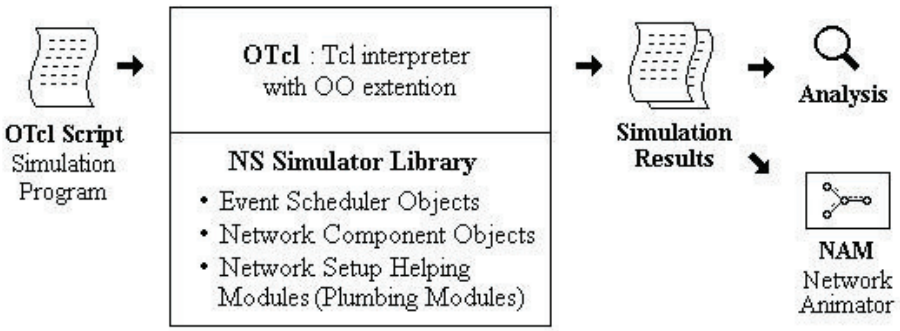
Πρέπει να αναφέρουμε ότι η σύνδεση μεταξύ των δύο αυτών γλωσσών προγραμματισμού γίνεται ιδιαίτερα αποδοτικά και η απόφαση για το βαθμό στον οποίο ο χρήστης θα χρησιμοποιήσει τη κάθε γλώσσα εξαρτάται κυρίως από την εξοικείωσή του με αυτήν και από τη συγκεκριμένη προσομοίωση. Σε γενικές όμως γραμμές ακολουθείται η τακτική που προαναφέρθηκε. Επίσης η δημιουργία αντικειμένων – κλάσεων που συνδέονται άμεσα με τα βασικά αντικείμενα που χρησιμοποιεί ο NS (κόμβοι, agents, ζεύξεις) είναι ιδιαίτερα βολική, καθώς η όλη υλοποίηση γίνεται πολύ πιο κατανοητή και είναι πιο κοντά στην πραγματικότητα. Στη δημιουργία αυτών των αντικειμένων η χρήση της OTCL και της C++ ενδείκνυται καθώς και οι δύο γλώσσες υποστηρίζουν τον αντικειμενοστραφή προγραμματισμό.

Το αρχείο το οποίο τρέχουμε με τον NS έχει τη μορφή name.tcl, όπου name είναι το όνομα που εμείς έχουμε δώσει στο αρχείο μας και η κατάληξη .tcl δηλώνει ότι είναι γραμμένο σε OTCL. Κατά την εκτέλεση της προσομοίωσης ο NS έχει τη δυνατότητα να δημιουργεί δύο βασικά αρχεία όπου αποθηκεύονται τα δεδομένα

(δυνητικά θα μπορούσε και σε περισσότερα, παρακολουθώντας π.χ. και την κίνηση πάνω στις ζεύξεις), το myname.tr και το myname.nam.

Το πρώτο είναι ένα αρχείο απλού κειμένου (plain text) στο οποίο περιγράφονται τα διάφορα γεγονότα (λήψη, είσοδος/ έξοδος από κάποια ουρά) που αφορούν όλα τα πακέτα που κινούνται στο δίκτυο. Αυτό το αρχείο είναι ιδιαίτερα χρήσιμο για τη μετέπειτα μελέτη των αποτελεσμάτων καθώς περιέχει λεπτομερώς όλες τις πληροφορίες που δυνητικά θα μπορούσαμε να χρειαστούμε για κάθε είδους υπολογισμό παραμέτρων του δικτύου σχετικά με την κίνηση σε αυτό. Αυτό το αρχείο το επεξεργαζόμαστε με οποιοδήποτε άλλο πρόγραμμα που μπορεί να δεχθεί είσοδο από αρχείο (π.χ. με AWK, OTCL, C++). Όπου είναι δυνατόν προτιμάται η επεξεργασία με AWKS αφού με αυτόν τον τρόπο η γραμμική ανάλυση του κειμένου (parsing) γίνεται πολύ πιο γρήγορα.

Το δεύτερο είναι ένα αρχείο που αποτελεί είσοδο για τον NAM (Network AniMator) ένα πρόγραμμα που συνοδεύει τον NS και δίνει την δυνατότητα για εποπτική παρακολούθηση της προσομοίωσης. Πιο συγκεκριμένα μπορούμε να παρατηρήσουμε στον NAM τους κόμβους και τις συνδέσεις μεταξύ τους, την κίνηση των πακέτων, το αν κάποια ζεύξη είναι ενεργή ή όχι. Επίσης δίνεται η δυνατότητα – στις πιο πρόσφατες εκδόσεις – να γίνει χρωματισμός των πακέτων ανάλογα με τις επιθυμίες του χρήστη (π.χ. τα πακέτα ανάλογα με το είδος τους να έχουν διαφορετικό χρώμα), να παριστάνονται οι κόμβοι με διάφορα σχήματα ανάλογα με το είδος τους – το οποίο καθορίζεται από τον χρήστη – ή να αλλάζουν εν δυνάμει χρώματα ανάλογα το γεγονός (event) που συμβαίνει. Γενικά ο NAM για μικρές τοπολογίες μας δίνει τη δυνατότητα εποπτικής παρακολούθησης της προσομοίωσης έτσι ώστε να διαπιστώνουμε γρήγορα και εύκολα τη σωστή λειτουργία του δικτύου. Βέβαια για μεγάλες τοπολογίες δεν είναι εύκολη η εποπτική παρακολούθηση της κίνησης, καθώς η παρακολούθηση πάνω από 50 κόμβων είναι πρακτικά αδύνατη, ενώ σε κάθε περίπτωση για ακριβή αποτελέσματα είναι απαραίτητη η επεξεργασία του myname.tr με κάποιες από τις μεθόδους που παρουσιάσθηκαν.



Εικόνα 3: Παράδειγμα χρήσης του NS

3.4. Περιγραφή ενεργειών στην εφαρμογή

Όπως αναφέραμε και παραπάνω για την συγκεκριμένη εργασία χρησιμοποιήθηκαν δύο κατηγορίες πρωτοκόλλων, τα proactive dsdv, olsr) και τα

reactive (aodv, dsr, dymo). Από τα παραπάνω, τα DSDV,AODV και DSR είναι διαθέσιμα στο NS-2, ενώ τα για τα DYMO και OLSR υπάρχουν υλοποιήσεις οι οποίες δεν βρίσκονται στην βασική έκδοση και πρέπει να εγκατασταθούν με το χέρι.

Για το DYMO, επιλέχθηκε η υλοποίηση (http://sourceforge.net/projects/dymoum/) η οποία εγκαταστάθηκε σε NS-2.34. Για το OLSR, επιλέχθηκε η υλοποίηση (http://hipercom.inria.fr/olsr/#simu), και χρησιμοποιήθηκε προεγκατεστημένη έκδοση του NS-2 σε chroot περιβάλλον (http://hipercom.inria.fr/olsr/NS2/Debian-NS2-OLSRv3-0.0.1.tar.gz).

3.5. Περιγραφή του περιβάλλοντος προσομοίωσης

3.5.1. Τοπολογία και δικτυακή κίνηση προσομοίωσης

Για τη μελέτη της απόδοσης των πρωτοκόλλων, χρησιμοποιήθηκαν οι εξής τοπολογίες:

- **Motion-No Gateway**: Πρόκειται για μια τοπολογία σε ένα επίπεδο 500x500 στο οποίο 5 κόμβοι κινούνται τυχαία και μιλάνε όλοι με όλους. Για τη παραγωγή της κίνησης δημιουργούνται CBR ροές (σταθερού ρυθμού ροή πακέτων UDP) σε τυχαίες χρονικές στιγμές.
- **No Motion - Gateway**: Πρόκειται για μια τοπολογία σε ένα επίπεδο 500x500 στο οποίο 4 σταθεροί κόμβοι μιλάνε με ένα άλλο συγκεκριμένο σταθερό κόμβο. Ο κόμβος βρίσκεται στο κέντρο του επιπέδου και δέχεται όλη τη ροή δεδομένων γιατί θεωρείται ότι είναι gateway. Οι ασύρματοι κόμβοι δεν έχουν λόγο να επικοινωνήσουν μεταξύ τους, παρά μονάχα μέσω του gateway με κάποιο εξωτερικό δίκτυο (π.χ. Internet). Για τη παραγωγή της κίνησης δημιουργούνται CBR ροές (σταθερού ρυθμού ροή πακέτων UDP) σε τυχαίες χρονικές στιγμές.
- **Motion - Gateway**: Πρόκειται για μια τοπολογία σε ένα επίπεδο 500x500 στο οποίο 4 κινούμενοι κόμβοι μιλάνε με ένα άλλο συγκεκριμένο σταθερό κόμβο. Ο κόμβος αυτός βρίσκεται στο κέντρο του επιπέδου και δέχεται όλη τη ροή δεδομένων γιατί θεωρείται ότι είναι gateway. Για τη παραγωγή της κίνησης δημιουργούνται CBR ροές (σταθερού ρυθμού ροή πακέτων UDP) σε τυχαίες χρονικές στιγμές

Για την εξαγωγή των αριθμητικών αποτελεσμάτων, δημιουργήθηκε μια σειρά από scripts που βασίζονται στην AWK, μια γλώσσα (data-driven), η οποία έχει σχεδιαστεί για την επεξεργασία δεδομένων text. Τα προγράμματα αυτά, που παρατίθενται στο παράρτημα, σχεδιάστηκαν προκειμένου να μπορούν να εξαχθούν από το trace file τα παρακάτω μετρικά:

- **packet loss**: το ποσοστό των πακέτων που χάθηκαν
- **delay**: καθυστέρηση στην αποστολή ενός πακέτου
- **jitter**: Διακύμανση καθυστέρησης

3.5.2. Εκτέλεση πειραμάτων και επεξεργασία αποτελεσμάτων

Για τα πειράματα χρησιμοποιήθηκε το αρχείο wifi.tcl που βρίσκεται στο παράρτημα, το οποίο φορτώνει την τοπολογία και την κίνηση από 4 διαφορετικά αρχεία (χρησιμοποιούμε 2 κάθε φορά ανάλογα με την τοπολογία που θέλουμε να προσομοιώσουμε (Motion-No Gateway/ No Motion-Gateway/Motion-Gateway).

- cbr-no-gateway.tcl
- cbr-gateway.tcl
- motion.tcl
- nomotion.tcl

Στο αρχείο wifi.tcl ορίζουμε κάθε φορά το πρωτόκολλο δρομολόγησης που θέλουμε να χρησιμοποιήσουμε:

```
# ======================================================================
# Define options
# ======================================================================
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy            ;# network interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq
set val(nn)             6                          ;# number of mobilenodes
set val(rp)             AODV                       ;# routing protocol
```

Στη μεταβλητή rp μπορούμε να θέσουμε μια από τις παρακάτω τιμές: DSDV, AODV, OLSR, DSR. Για το πρωτόκολλο DYMO χρησιμοποιείται το αρχείο dymoum.tcl, γιατί η συγκεκριμένη υλοποίηση χρειάζεται κάποιες παραπάνω ρυθμίσεις από τα υπόλοιπα πρωτόκολλα.

```
user@ubuntu:~/p-r$ ns wifi.tcl
num_nodes is set 6
warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl
INITIALIZE THE LIST xListHead
Starting Simulation...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
NS EXITING...
user@ubuntu:~/p-r$
```

Εικόνα 4

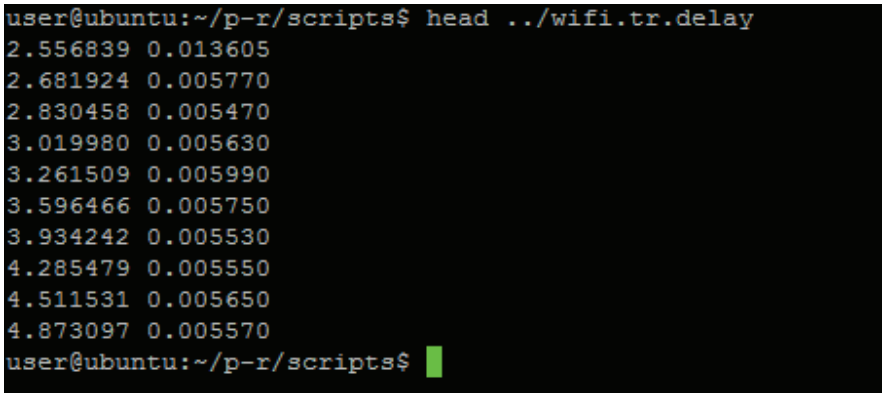
Η εκτέλεση του αρχείου tcl έχει ως αποτέλεσμα την παραγωγή ενός αρχείου wifi.tr. Το αρχείο αυτό περιέχει γεγονότα που αφορούν τη δημιουργία, μετάδοση και παραλαβή πακέτων από τους διάφορους ασύρματους και ενσύρματους κόμβους του δικτύου. Για την αριθμητική επεξεργασία των δεδομένων αυτών, χρησιμοποιήθηκαν AWK scripts. Η γλώσσα AWK χρησιμοποιείται για την επεξεργασία γραμμών κειμένου και την εξαγωγή στοιχείων.

Η επεξεργασία γίνεται με τη χρήση ενός κεντρικού sh script (statistics.sh) το οποίο μετρά packet loss, jitter και delay. Τα επιμέρους AWK script (measure-loss.awk, measure-delay.awk, measure-jitter.awk) τα οποία χρησιμοποιεί υπάρχουν αναλυτικά στο παράρτημα της εργασίας. Το κεντρικό script παρατίθεται στην συνέχεια:

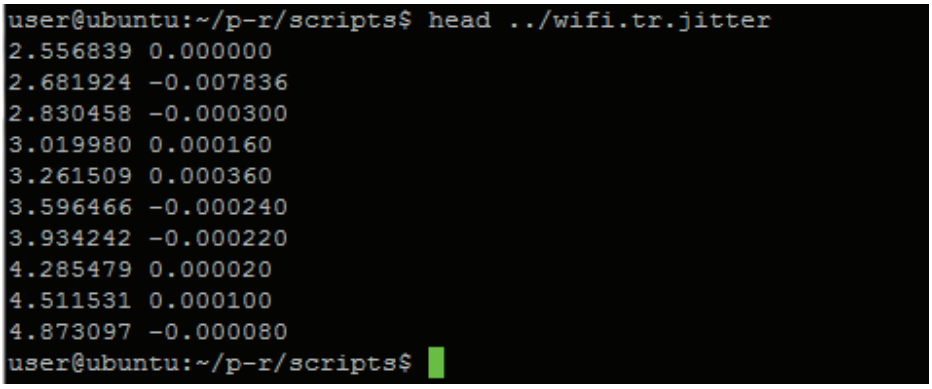
Εκτελώντας το statistics.sh παίρνουμε τα ακόλουθα αποτελέσματα:

```
user@ubuntu:~/p-r/scripts$ ./statistics.sh ../wifi.tr
Packets sent:1895 lost:401

delay:


jitter:
user@ubuntu:~/p-r/scripts$
```

Εικόνα 5: Οι τιμές για το packet loss τυπώνονται στην οθόνη, ενώ οι τιμές για το jitter και delay αποθηκεύονται σε αρχεία της μορφής wifi.tr.delay και wifi.tr.jitter.



Εικόνα 6: Η πρώτη στήλη αντιστοιχεί στο χρόνο, ενώ η δεύτερη στήλη είναι η μέτρηση της καθυστέρησης ή του jitter.



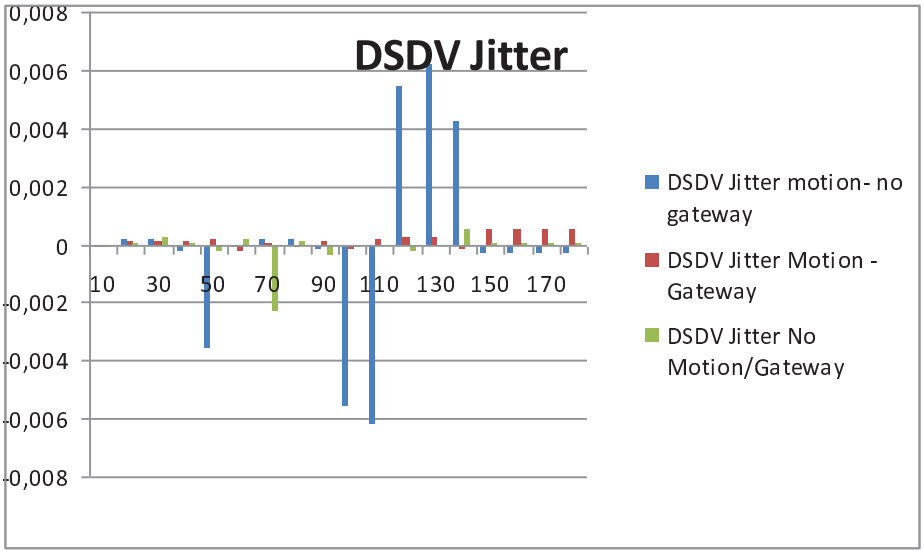
Εικόνα 7

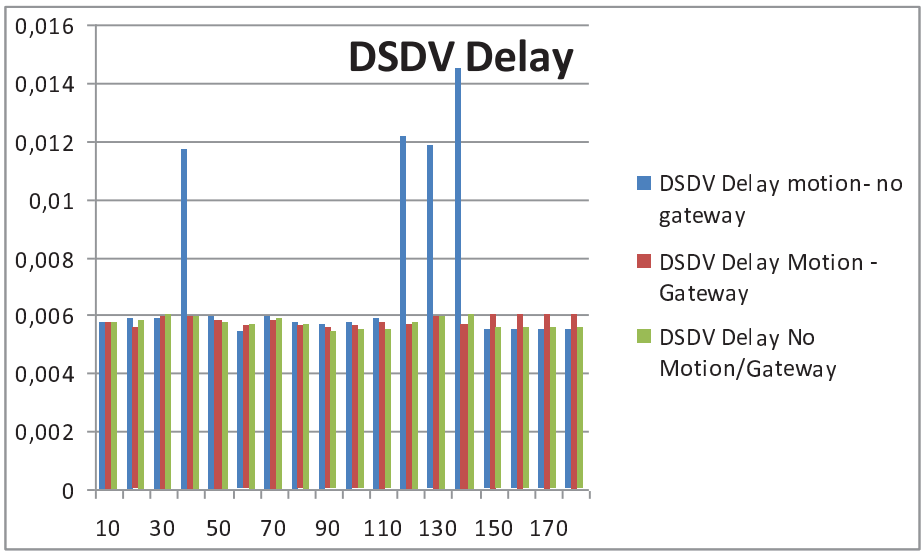
Τα δεδομένα τα αρχεία αυτά, φορτώθηκαν στο Microsoft Excel, στο οποίο έγινε ανάλυση των αποτελεσμάτων και παραγωγή ραβδογραμμάτων. Τα αριθμητικά αποτελέσματα παρουσιάζονται αναλυτικά στην επόμενη ενότητα.

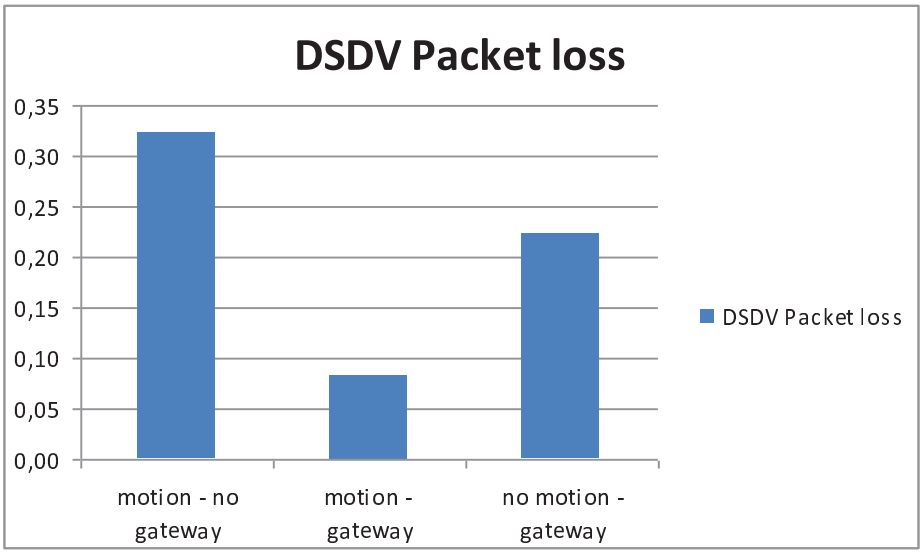
3.6. Μετρήσεις

3.6.1. Αριθμητικά αποτελέσματα για το DSDV

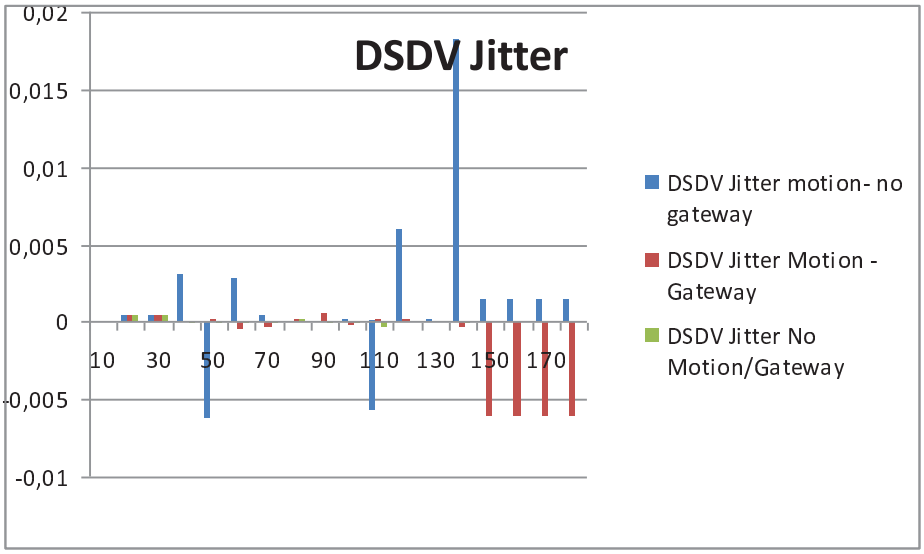
Ακολουθούν γραφικές παραστάσεις για το πρωτόκολλο DSDV. Εμφανίζονται κατά σειρά, ραβδογράμματα για το packet loss, την καθυστέρηση και το jitter, για κάθε ένα από τα τρία σενάρια.

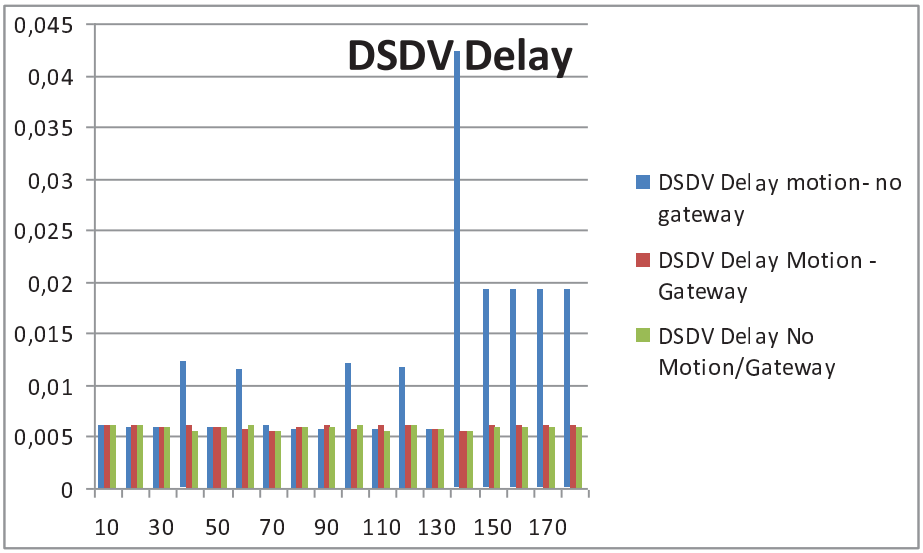

DSDV Jitter
0,008
0,006
0,004
0,002
0
-0,002
-0,004
-0,006
-0,008
10
30
50
70
90
110
130
150
170
DSDV Jitter motion- no gateway
DSDV Jitter Motion - Gateway
DSDV Jitter No Motion/Gateway

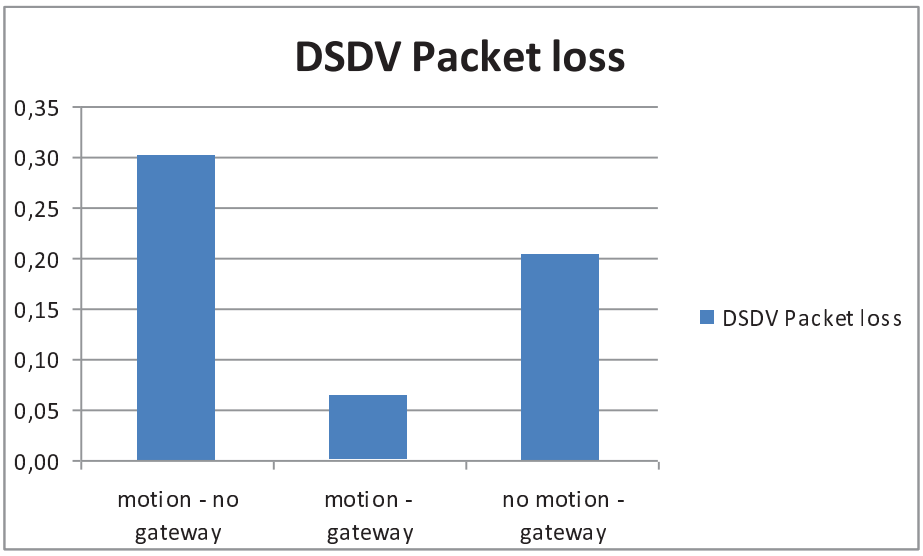

DSDV Delay
0,016
0,014
0,012
0,01
0,008
0,006
0,004
0,002
0
10
30
50
70
90
110
130
150
170
DSDV Delay motion- no gateway
DSDV Delay Motion - Gateway
DSDV Delay No Motion/Gateway



Στη συνέχεια εάν αλλάξουμε το ρυθμό αποστολής ενός πακέτου(interval) και το κάνουμε ανά 0.25 δευτερόλεπτα έχουμε τα εξής διαγράμματα.





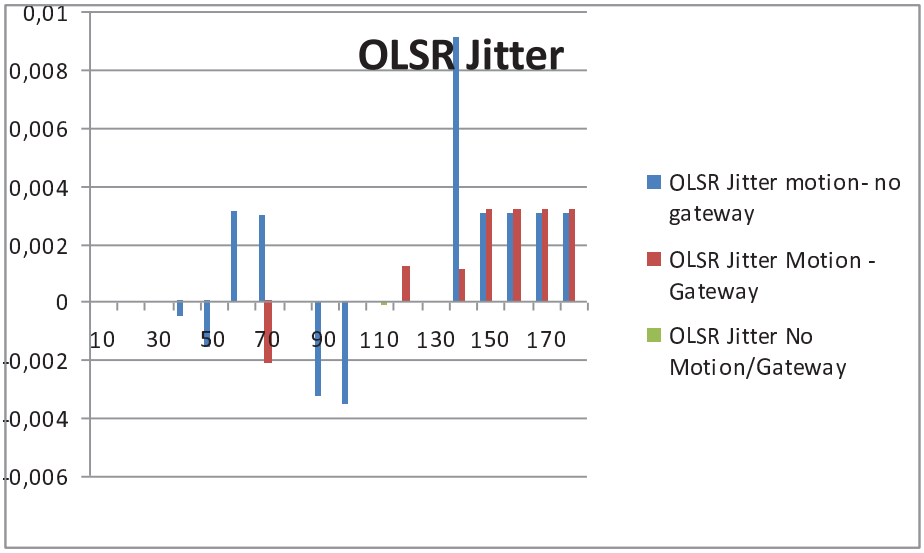


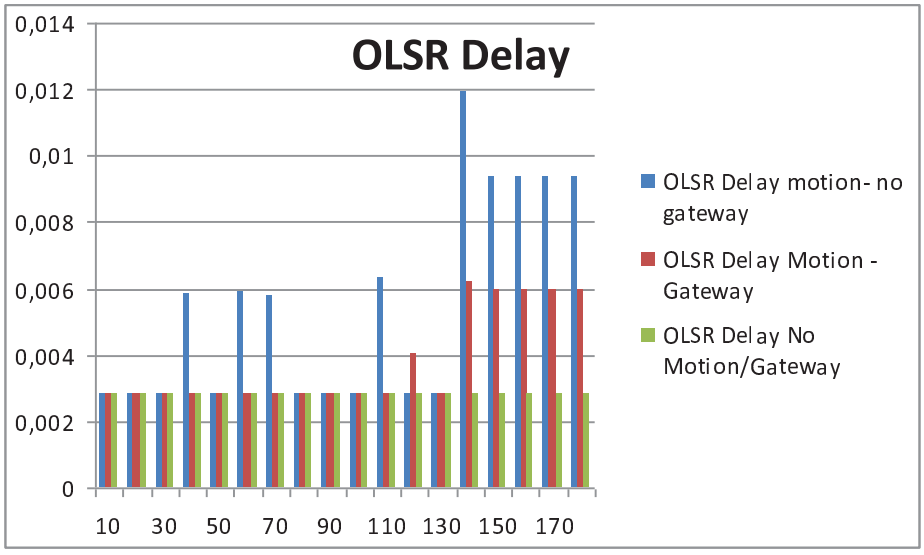
Το DSDV είναι ένα proactive πρωτόκολλο, το οποίο δημιουργεί τις ζεύξεις ανάμεσα στους κόμβους, πριν αυτές απαιτηθούν. Το πρωτόκολλο έχει πολύ καλή απόδοση, η κίνηση στους κόμβους δεν επηρεάζει αρνητικά τη λειτουργία του. Τα αποτελέσματα και για τους 2 ρυθμούς αποστολής είναι παρόμοια. Δεν παρατηρείται βελτίωση στην απόδοση του πρωτοκόλλου με την μείωση του ρυθμού αποστολής πακέτων.

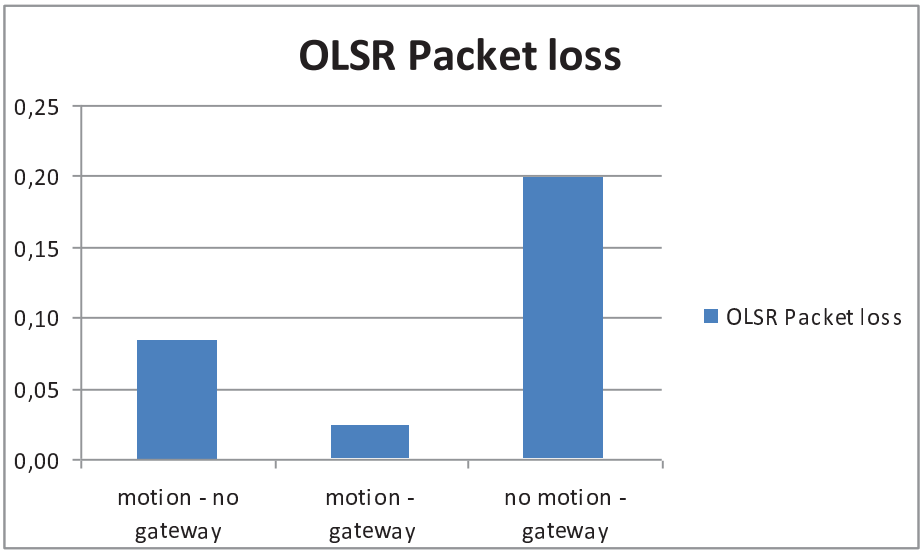
Παρατηρούμε ότι το DSDV, παρουσιάζει τη χειρότερη απόδοση στο σενάριο με κίνηση και χωρίς κεντρικό gateway. Το αποτέλεσμα αυτό είναι αναμενόμενο, γιατί η ύπαρξη του gateway στο κέντρο του επιπέδου, κάνει πιο εύκολη την αποστολή των πακέτων.

3.6.2. Αριθμητικά αποτελέσματα για το OLSR

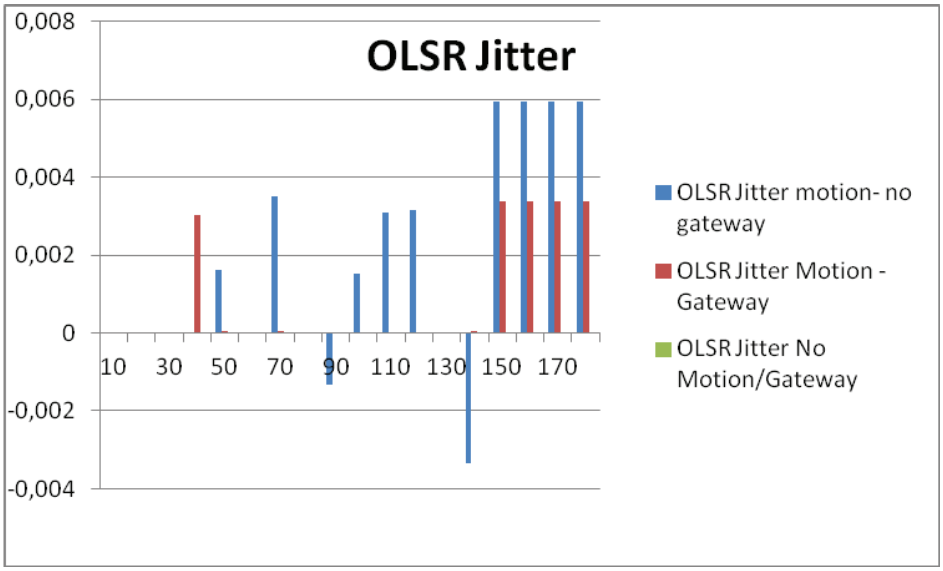
Το OLSR είναι ένα proactive πρωτόκολλο, το οποίο δημιουργεί τις ζεύξεις ανάμεσα στους κόμβους, πριν αυτές απαιτηθούν. Το πρωτόκολλο έχει πολύ καλή απόδοση, η κίνηση στους κόμβους δεν επηρεάζει αρνητικά τη λειτουργία του.

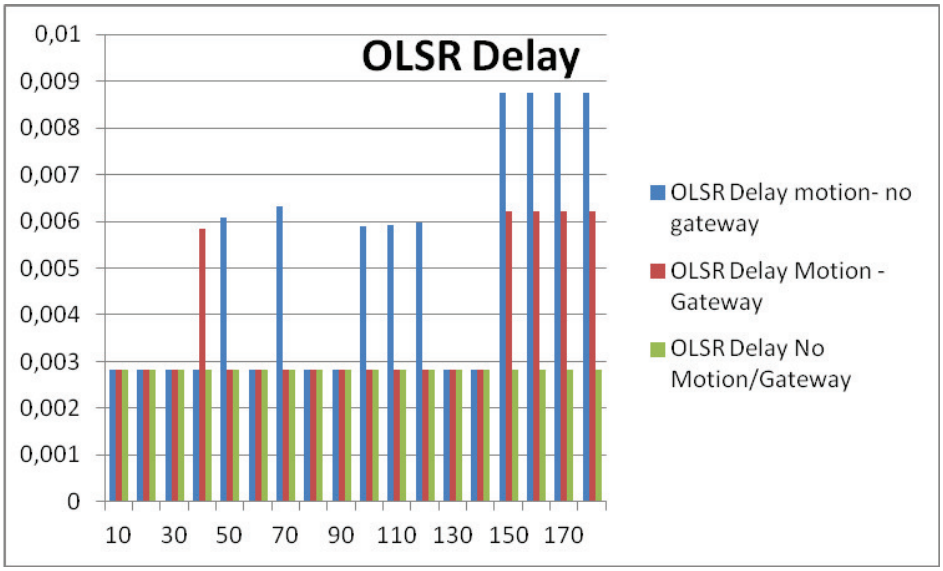


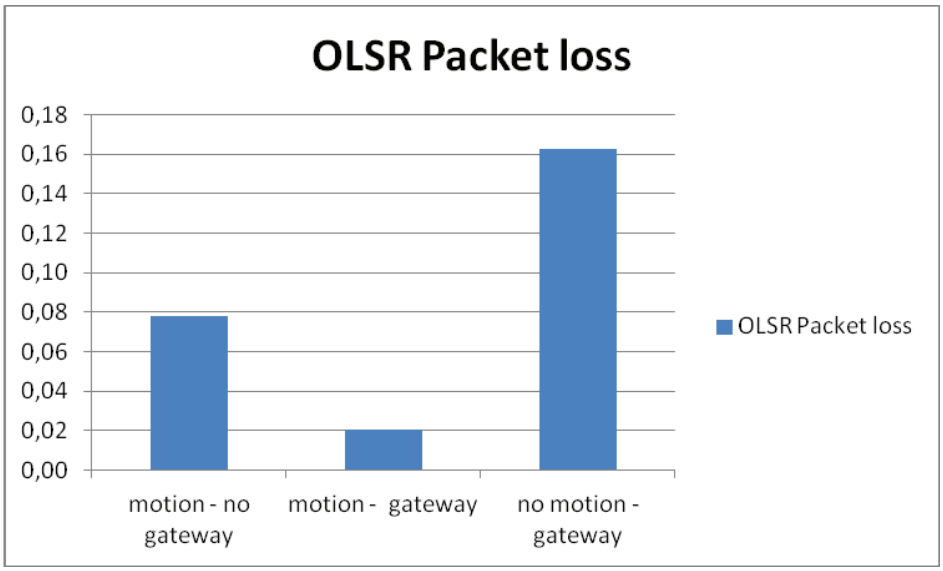




Ακολουθούν τα διαγράμματα τροποποιώντας πάλι το χρόνο ανάμεσα σε διαδοχικά πακέτα udp.







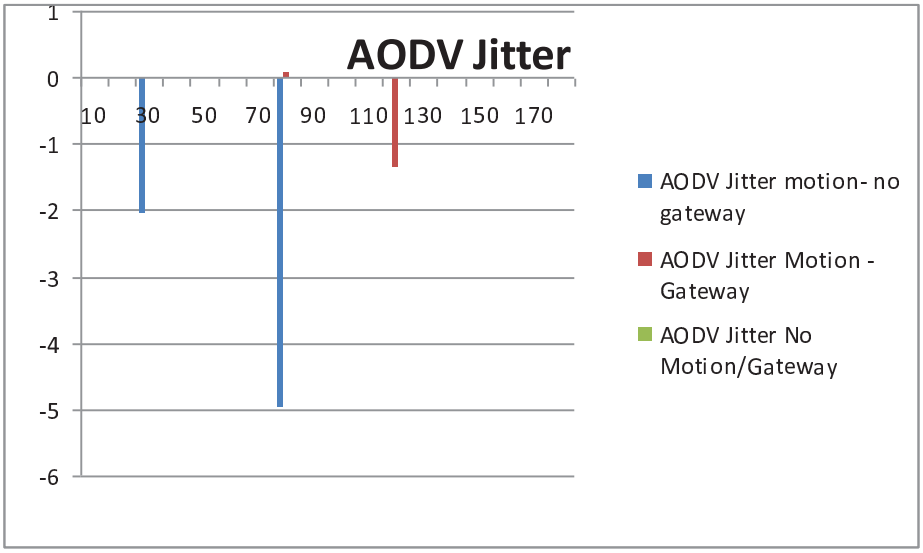
Παρατηρούμε ότι το OLSR, παρουσιάζει τη χειρότερη απόδοση στο σενάριο χωρίς κίνηση και με κεντρικό gateway. Το αποτέλεσμα αυτό προκύπτει από κάποιον κόμβο ο οποίος είναι απομακρυσμένος και έχει χαμηλή απόδοση η ζεύξη του με τον gateway.

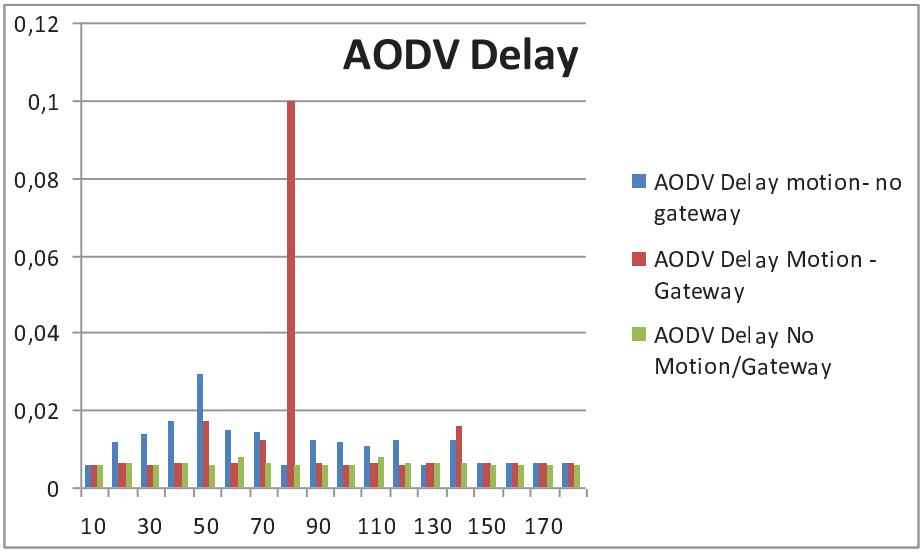
Τα αποτελέσματα και για τους 2 ρυθμούς αποστολής είναι παρόμοια. Παρατηρείται σχετική βελτίωση στην απόδοση του πρωτοκόλλου με την μείωση του ρυθμού αποστολής πακέτων στοιχείο που ήταν αναμενόμενο.

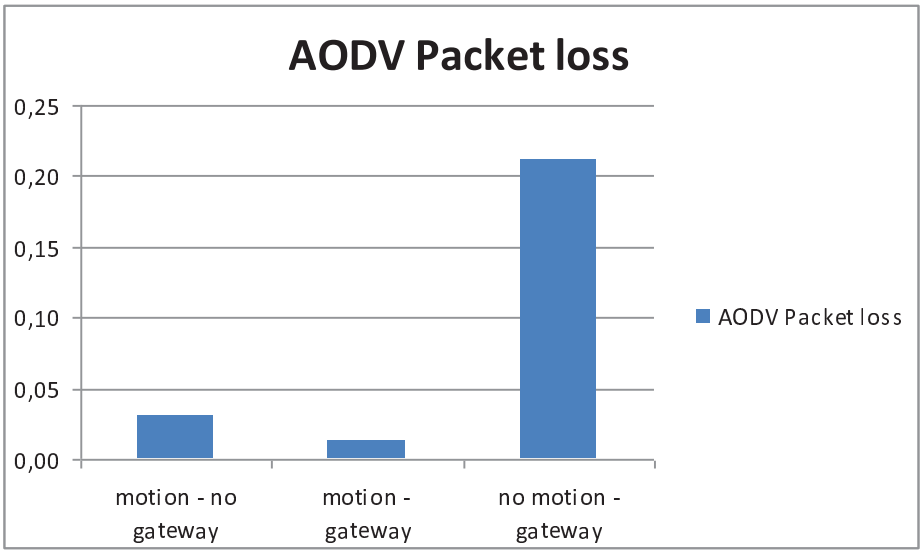
Καλύτερη απόδοση, έχει στην περίπτωση κίνησης με κεντρικό gateway. Το αποτέλεσμα αυτό είναι αναμενόμενο, γιατί η ύπαρξη του gateway στο κέντρο του επιπέδου, κάνει πιο εύκολη την αποστολή των πακέτων, ενώ λόγω της τυχαίας κίνησης κανένας κόμβος δεν βγαίνει εκτός εμβέλειας των άλλων για μεγάλο διάστημα

3.6.3. Αριθμητικά αποτελέσματα για το AODV

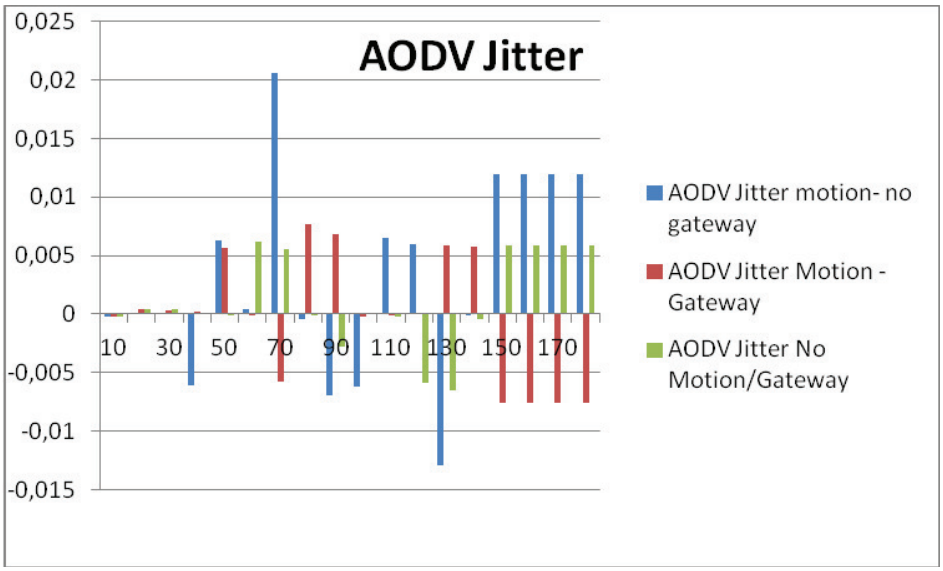
Το AODV είναι ένα reactive πρωτόκολλο, το οποίο δημιουργεί τις ζεύξεις ανάμεσα στους κόμβους όταν αυτές απαιτηθούν. Το πρωτόκολλο παρόμοια απόδοση με το DSDV αλλά χειρότερη από το OLSR, η κίνηση στους κόμβους δεν επηρεάζει αρνητικά τη λειτουργία του.

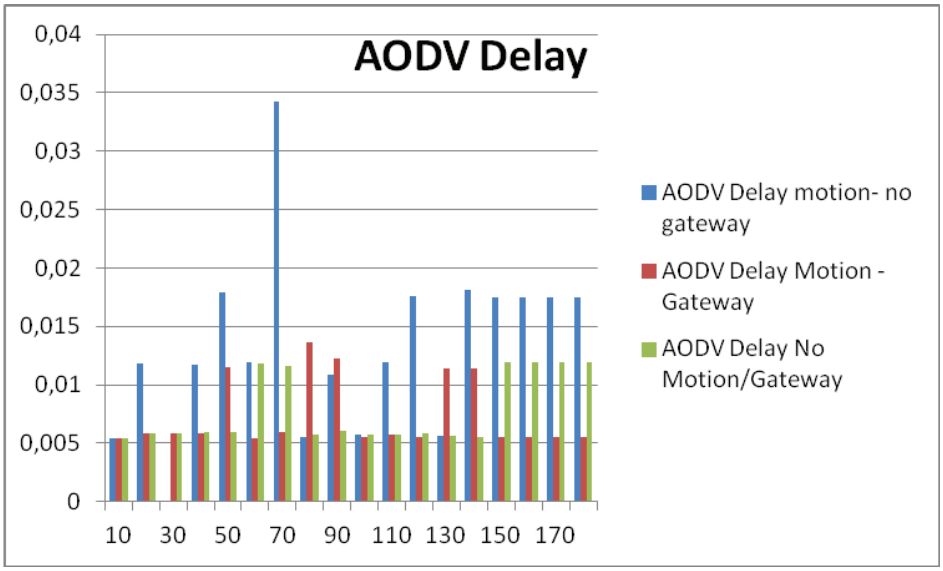


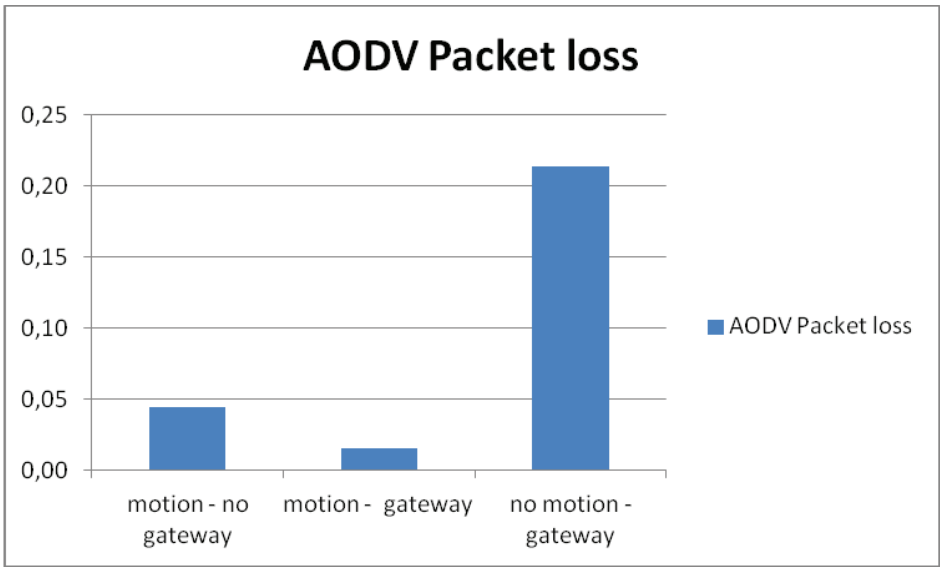




Ακολουθούν και τα διαγράμματα με λιγότερα πακέτα.







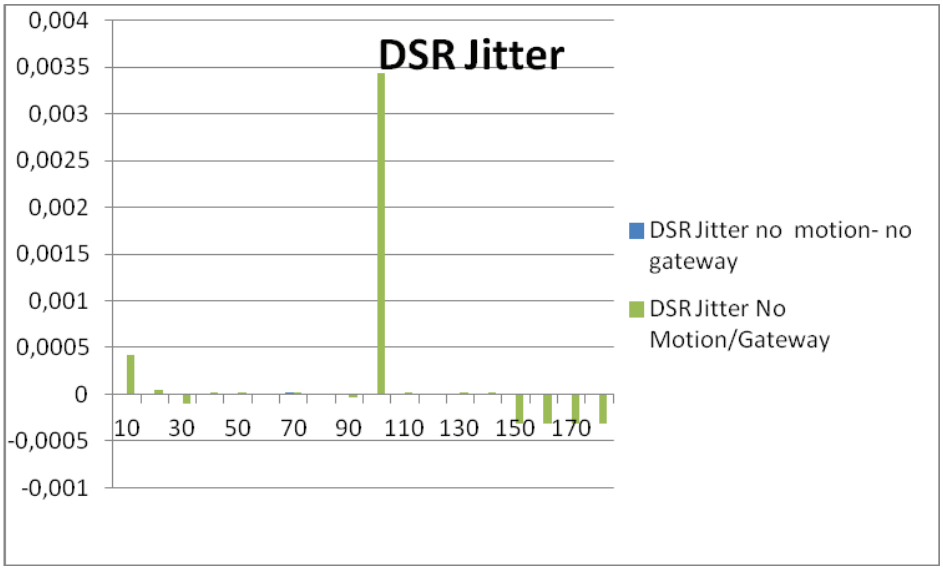
Παρατηρούμε ότι το AODV παρουσιάζει τη χειρότερη απόδοση στο σενάριο με κίνηση και χωρίς κεντρικό gateway. Το αποτέλεσμα αυτό προκύπτει από το γεγονός ότι δημιουργούνται καθυστερήσεις κάθε φορά που χρειάζεται να παραχθεί ξανά ο πίνακας δρομολόγησης.

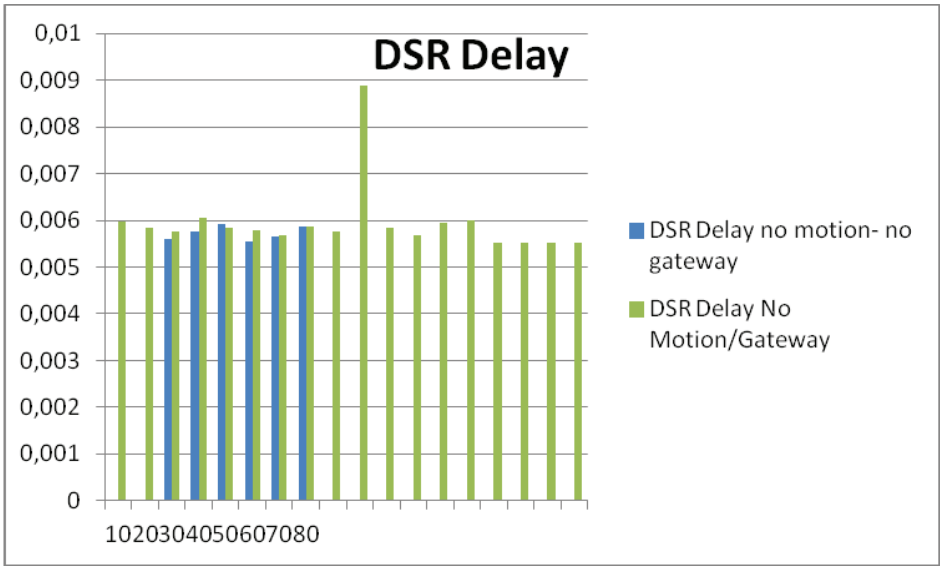
Τα αποτελέσματα και για τους 2 ρυθμούς αποστολής είναι παρόμοια. Δεν παρατηρείται ιδιαίτερη βελτίωση στην απόδοση του πρωτοκόλλου με την μείωση του ρυθμού αποστολής πακέτων.

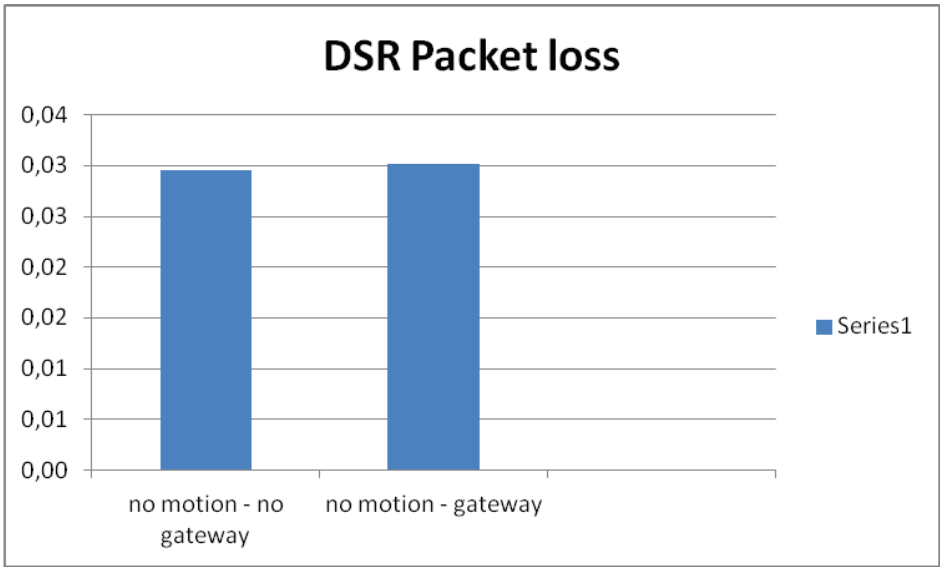
Μικρότερη καθυστέρηση και jitter έχει στην περίπτωση χωρίς κίνηση με κεντρικό gateway. Το αποτέλεσμα αυτό είναι αναμενόμενο, γιατί η ύπαρξη του gateway στο κέντρο του επιπέδου, κάνει πιο εύκολη την αποστολή των πακέτων ενώ το πρωτόκολλο δε χρειάζεται να δημιουργεί ξανά τους πίνακες δρομολόγησης και είναι ταχύτερο καθώς υπάρχει περισσότερο διαθέσιμο εύρος ζώνης για τα δεδομένα.

3.6.4. Αριθμητικά αποτελέσματα για το DSR

Ακολουθούν γραφικές παραστάσεις για το πρωτόκολλο DSR. Εμφανίζονται ραβδογράμματα για το packet loss, την καθυστέρηση και το jitter, για κάθε ένα από τα τρία σενάρια.



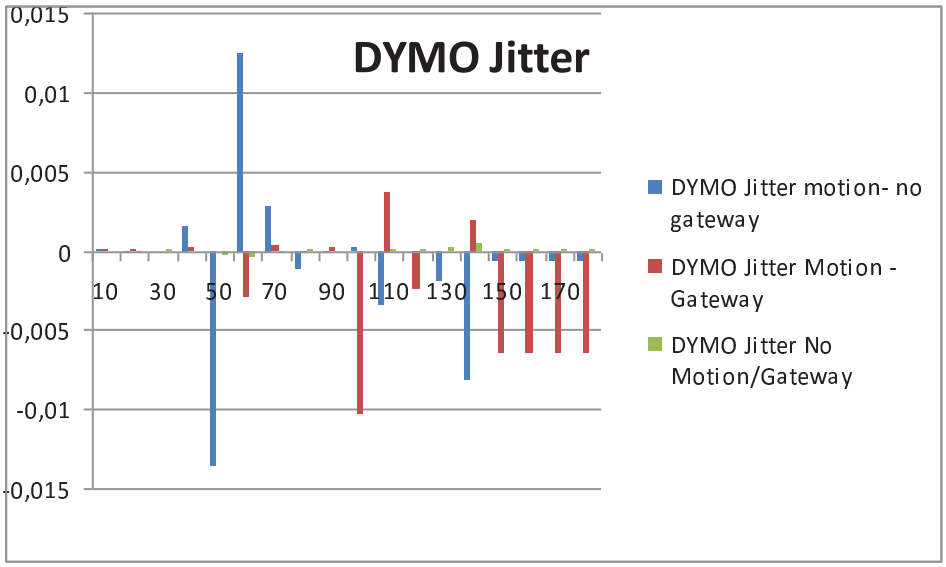


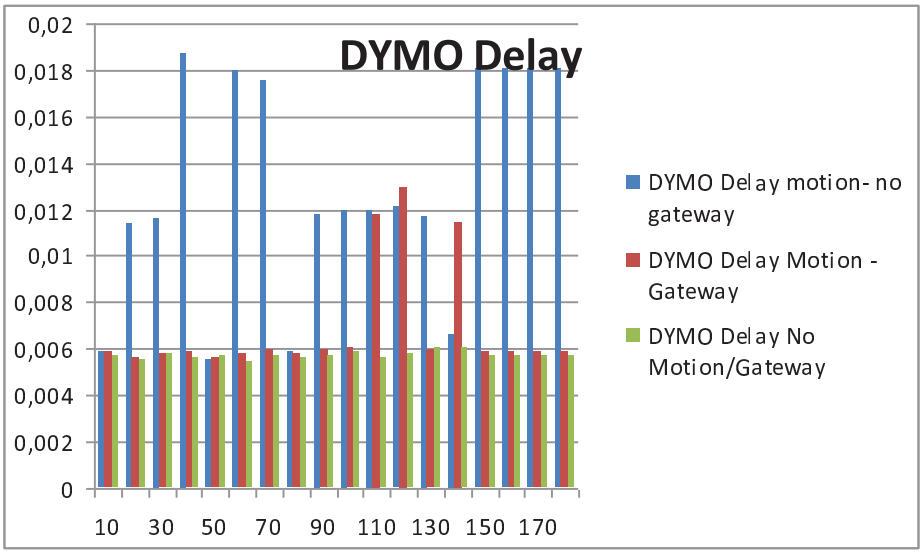


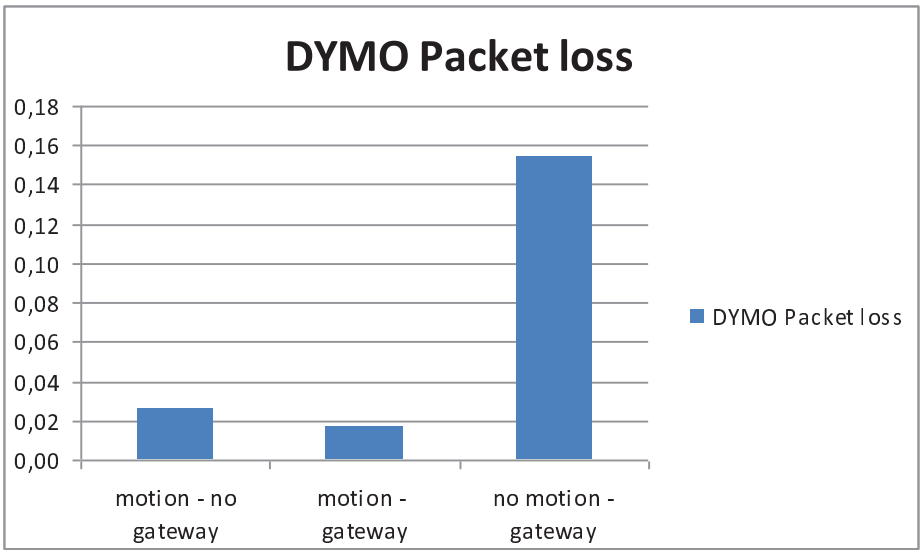
Το πρωτόκολλο DSR εξετάστηκε μονάχα σε σενάριο με σταθερούς κόμβους, λόγω αδυναμίας στο ns2 (ο κώδικας του DSR φαίνεται να έχει bug που προκαλεί exception και τον τερματισμό του, όταν οι κόμβοι κινούνται.)Το DSR είναι παρόμοιο πρωτόκολλο με το AODV. Παρατηρούμε και από τα αριθμητικά αποτελέσματα ότι η απόδοσή του είναι παρόμοια με το AODV.

3.6.5. Αριθμητικά αποτελέσματα για το DYMO

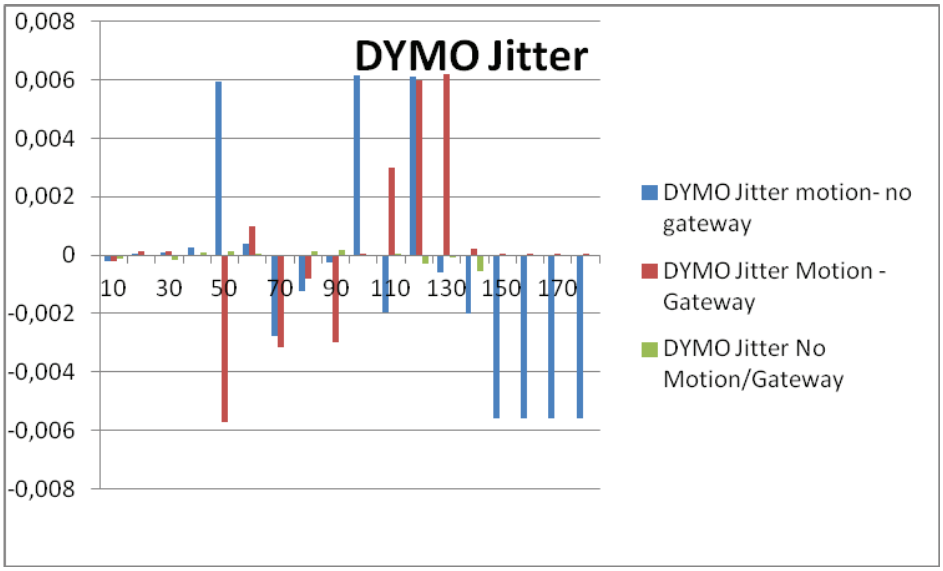
Το DYMO είναι ένα reactive πρωτόκολλο, το οποίο δημιουργεί τις ζεύξεις ανάμεσα στους κόμβους όταν αυτές απαιτηθούν. Το πρωτόκολλο έχει λίγο καλύτερη απόδοση με το AODV που είναι επίσης ένα reactive πρωτόκολλο.

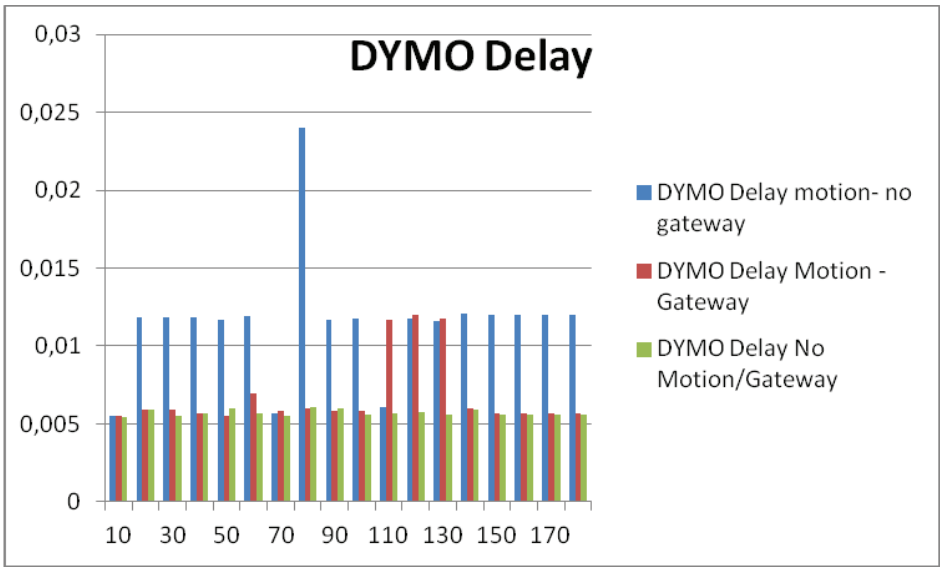


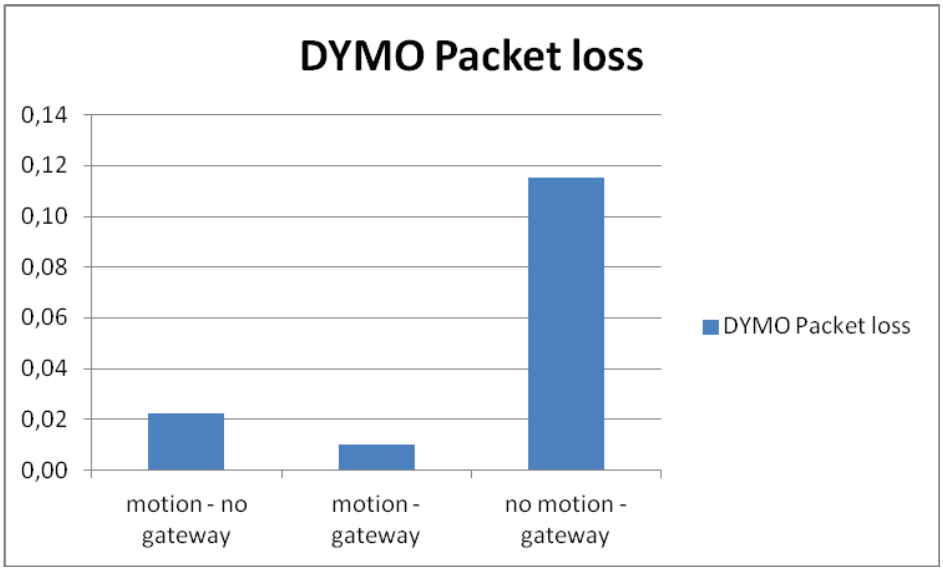




Με ρυθμό ανάμεσα σε διαδοχικά πακέτα να αυξάνεται έχουμε τα εξής αποτελέσματα.


DYMO Jitter
0,008
0,006
0,004
0,002
0
-0,002
-0,004
-0,006
-0,008
10 30 50 70 90 110 130 150 170
DYMO Jitter motion- no gateway
DYMO Jitter Motion - Gateway
DYMO Jitter No Motion/Gateway


DYMO Delay
0,03
0,025
0,02
0,015
0,01
0,005
0
10 30 50 70 90 110 130 150 170
DYMO Delay motion- no gateway
DYMO Delay Motion - Gateway
DYMO Delay No Motion/Gateway



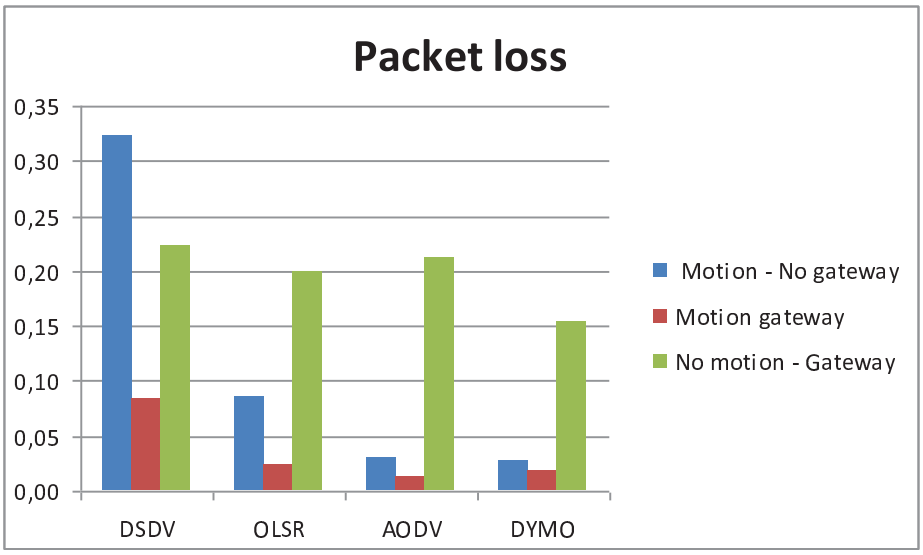
Τα αποτελέσματα και για τους 2 ρυθμούς αποστολής είναι παρόμοια. Παρατηρείται σχετική βελτίωση στην απόδοση του πρωτοκόλλου με την μείωση του ρυθμού αποστολής πακέτων. Μικρότερη καθυστέρηση και jitter έχει στην περίπτωση χωρίς κίνηση με κεντρικό gateway.

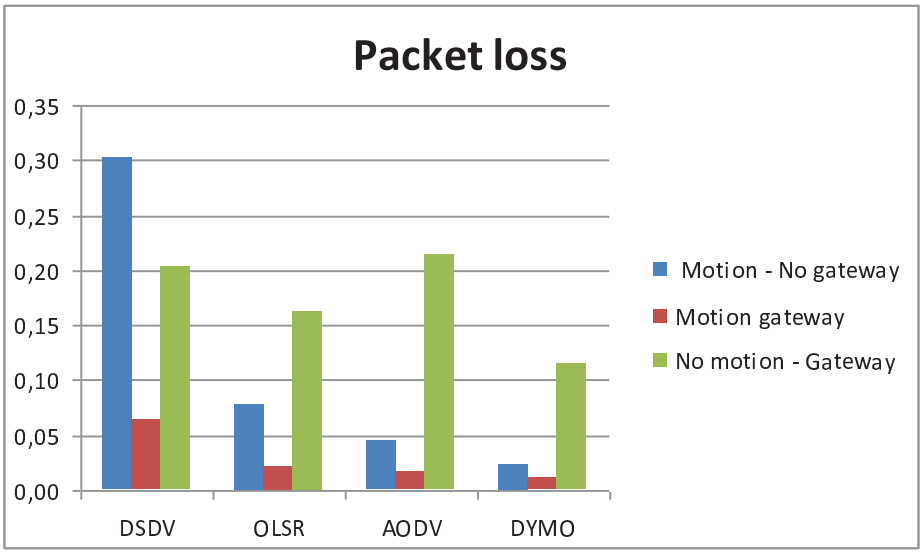
Παρατηρούμε ότι το DYMO παρουσιάζει τη χειρότερη απόδοση στο σενάριο με κίνηση και χωρίς κεντρικό gateway. Το αποτέλεσμα αυτό προκύπτει από το γεγονός ότι δημιουργούνται καθυστερήσεις κάθε φορά που χρειάζεται να παραχθεί ξανά ο πίνακας δρομολόγησης.

Κεφάλαιο 4: Συμπεράσματα

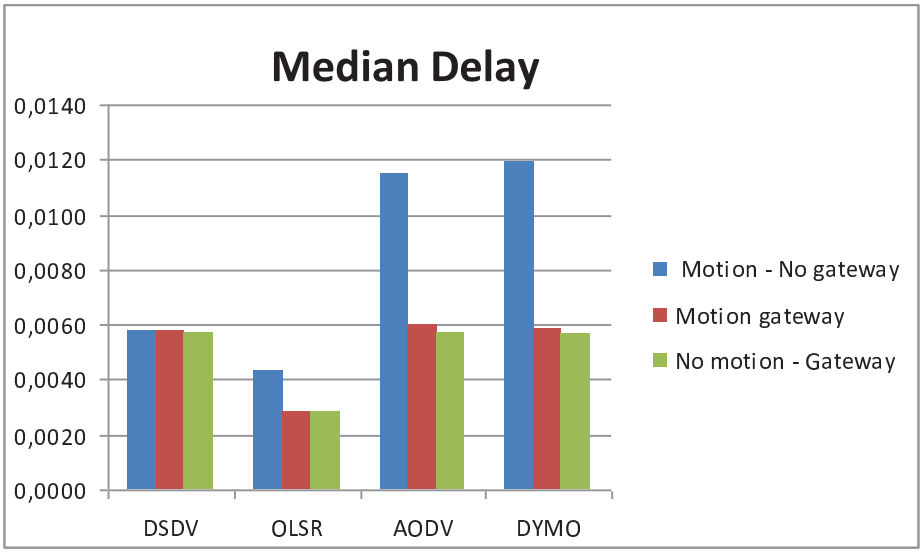
Στην ενότητα αυτή γίνονται συγκρίσεις ανάμεσα στα πρωτόκολλα DSDV, OLSR, AODV και DYMO. Παρουσιάζονται διαγράμματα τα οποία συναθροίζουν την απόδοση κάθε πρωτοκόλλου για τα τρία διαφορετικά σενάρια που δοκιμάστηκαν.

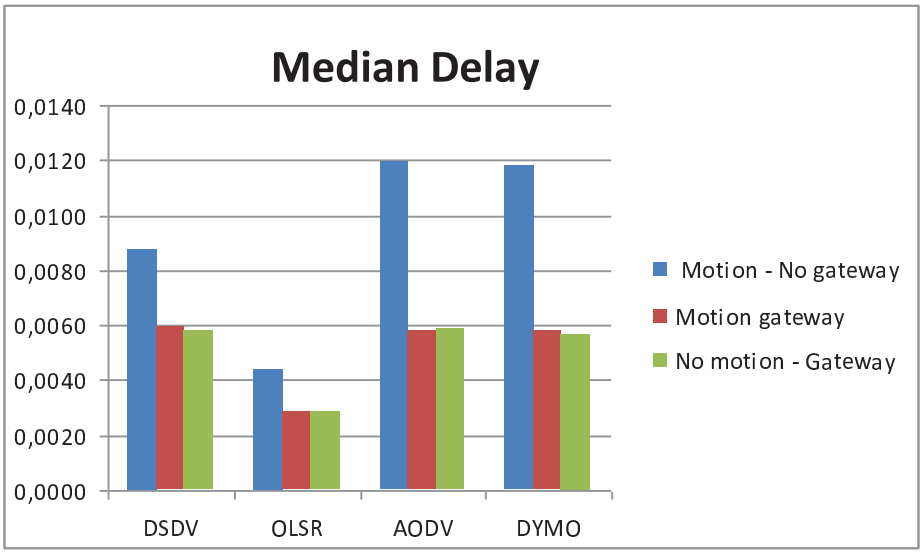
Αρχικά το πρώτο διάγραμμα αφορά τα σενάρια με ρυθμό μετάδοσης 0,25 και στη συνέχεια στο δεύτερο που ακολουθεί με 0,50.





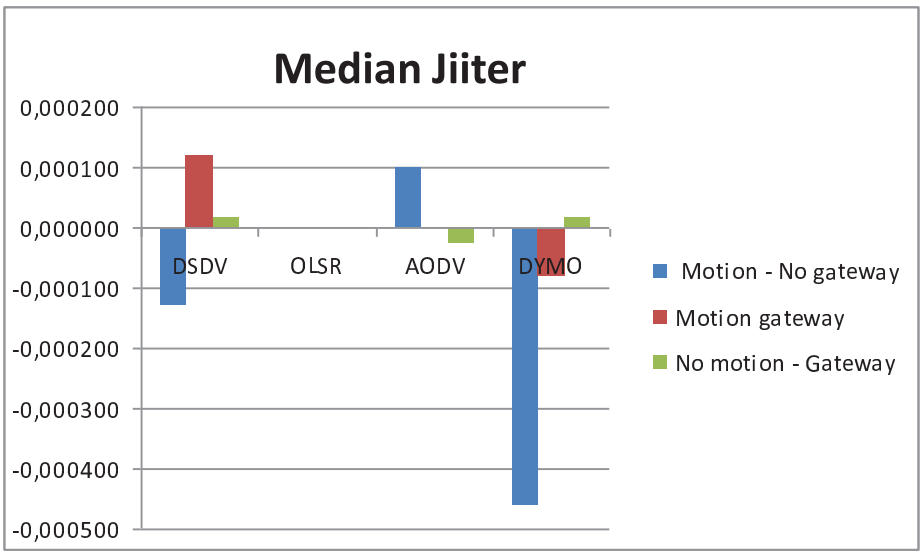
Παρατηρούμε ότι ενώ αλλάζει ο ρυθμός μετάδοσης πακέτων, η απόδοση για κάθε πρωτόκολλο στην ουσία δεν αλλάζει δραματικά. Όσον αφορά στην πιθανότητα να χαθεί πακέτο, παρατηρούμε ότι τα OLSR, AODV και DYMO έχουν παρόμοια απόδοση, ενώ το DSDV είναι χειρότερο. Το DYMO επειδή όπως έχουμε πει είναι reactive πρωτόκολλο δημιουργεί μόλις του ζητηθεί μία διαδρομή και με το συγκεκριμένο αριθμό κόμβων είναι λογικό που έχει παρόμοια αποτελέσματα με ένα proactive πρωτόκολλο όπως το OLSR.

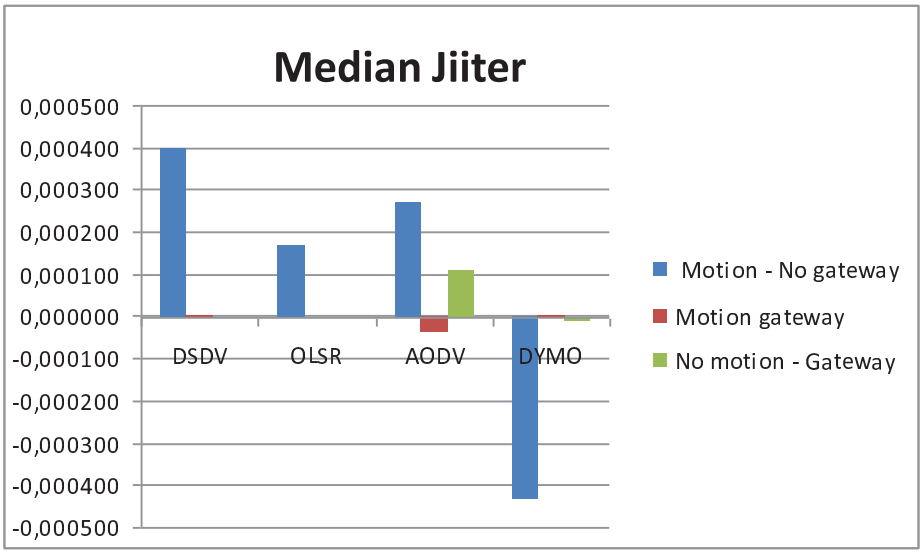




Το παραπάνω διάγραμμα μας δίνει τη διάμεσο της καθυστέρησης για κάθε σειρά δεδομένων και για τους δυο διαφορετικούς ρυθμούς αποστολής πακέτων. Η αλλαγή στο ρυθμό που στέλνονται τα πακέτα, φαίνεται να μην επηρεάζει την απόδοση των πρωτοκόλλων. Εκτός από το DSDV που δείχνει να αντιδρά χειρότερα όταν δεν υπάρχει στο κέντρο ένα gateway .

Παρατηρούμε ότι τα AODV και DYMO έχουν αυξημένη καθυστέρηση στο σενάριο χωρίς κεντρικό gateway και κίνηση. Η αιτία είναι ότι η ανάγκη για δημιουργία νέας δρομολόγησης συμβαίνει συχνά, και επειδή αυτή δεν γίνεται πριν ζητηθεί, παρατηρούνται καθυστερήσεις. Πιθανότητα σε κάποιο σενάριο στο οποίο η δικτυακή κίνηση ανάμεσα στους κόμβους να ήταν μεγαλύτερη τα AODV και DYMO να είχαν καλύτερη απόδοση.





Το παραπάνω διάγραμμα μας δίνει τη διάμεσο του jitter για κάθε σειρά δεδομένων. Τα αποτελέσματα δεν είναι αντίστοιχα με αυτά που αφορούν την μέτρηση της καθυστέρησης. Με την αύξηση του ρυθμού μετάδοσης των πακέτων παρατηρούμε ότι το jitter αυξάνεται πάρα πολύ.

Σε αυτή την εργασία στόχος ήτανε να γίνουν κάποια πειράματα που θα μας έδειχναν ποιο πρωτόκολλο αντιδρά καλύτερα υπό διαφορετικές συνθήκες σε κάθε πείραμα. Αρχικά αναλύσαμε τα χαρακτηριστικά τα οποία έχουν τα ad hoc δίκτυα και τη διαφορά τους με τα ασύρματα δίκτυα. Επίσης, αναφέραμε και τα βασικά χαρακτηριστικά αλλά και το πως λειτουργούν αυτά τα πρωτόκολλα. Στην συνέχεια αφού αναφέραμε κάποια εργαλεία προσομοίωσης αλλά και τα χαρακτηριστικά του προγράμματος που επιλέξαμε. Στη συνέχεια μετά από κάποιες μετρήσεις καταλήξαμε ότι το πρωτόκολλο OLSR έχει την καλύτερη απόδοση σε όλες τις περιπτώσεις. Οι διαφορές με τα υπόλοιπα πρωτόκολλα δεν είναι μεγάλες και είναι διαφορές που οφείλονται στην κατηγορία κάθε πρωτοκόλλου. Τα αποτελέσματα του olsr με βάση τα χαρακτηριστικά του μπορεί να μην αλλάξει όσον αφορά την απόδοση του ακόμα και σε μεγαλύτερο αριθμό κόμβων. Το αρνητικό για το oslr είναι ότι χρειάζεται να έχει διαθέσιμο εύρος ζώνης ώστε να μπορεί να δέχεται πακέτα με καινούριες αλλαγές στη τοπολογία. Βέβαια θα ήταν ενδιαφέρον για μελλοντική έρευνα να μελετηθεί τα αποτελέσματα που θα είχαμε σε αρκετά μεγαλύτερο δίκτυο όσον αφορά τα re active πρωτόκολλα που θεωρητικά αντιδρούν καλύτερα από τα proactive πρωτόκολλα που μπορεί να είναι πιο σταθερά, αλλά υστερούν αρκετά σε απόδοση.

Τέλος, θα ήταν ορθότερο να πούμε ότι η σωστή επιλογή και ρύθμιση των πρωτοκόλλων ανάλογα με την εφαρμογή και τα σενάρια για την κίνηση που έχουμε θα ήταν πιο ουσιαστικό για να πετύχουμε την καλύτερη απόδοση.

Βιβλιογραφία

1. Ad Hoc Networking, Charles E. Perkins, 2001
2. Inter-Domain Routing for Mobile Ad Hoc Networking, Chi-Kin Chau, Jon Crowcroft, Kang-Won Lee, Starsky H.Y. Wong
3. Routing And Multicasting Strategies in Wireless Mobile Ad hoc Network, Sung-Ju Lee,1999
4. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks
5. Introduction to Network Simulator NS2,Teerawat Issariyakul,Ekram Hossain,2009
6. Data and Computer Communications, William Stallings, 2007
7. Simulation and Analysis of a Wireless Ad-hoc,Network using energy aware DYMO, Phat Tran, Christer Wibom
8. Optimized Link State Routing Protocol for Ad Hoc Networks, P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Quayym, L Viennot
9. Comparing AODV and OLSR Routing Protocols, Aleksandr Huhtonen
10. Ns Simulator for beginners, Eitan Altman, Tania Jimenez
11. Simulation of Ad Hoc Networks,ns-2 compared to JiST/SWANS, Elamr Schol, Michael Feiri, Frank Kargl, Michael Weber
12. Wireless Sensor Network Simulators A Survey and Comparisons, Harsh Sundani, Haoyue Li, Vijay K. Devabhaktuni, Mansoor Alam, Prabir Bhattacharya

Παράρτημα

Wifi.tcl

```
# ======================================================================
# Define options
# ======================================================================
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy            ;# network interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq
set val(nn)             6                          ;# number of mobilenodes
set val(rp)             AODV                       ;# routing protocol

# ======================================================================
# Main Program
# ======================================================================



#
# Initialize Global Variables
#
set ns_        [new Simulator]
```

```
set tracefd     [open wifi.tr w]
$ns_ trace-all $tracefd

# set up topography object
set topo       [new Topography]

$topo load_flatgrid 500 500

#
# Create God
#
set god_ [create-god $val(nn)]

#
#  Create the specified number of mobilenodes [$val(nn)] and "attach" them
#  to the channel.
#  Here two nodes are created : node(0) and node(1)

# configure node

        $ns_ node-config -adhocRouting $val(rp) \
                         -llType $val(ll) \
                         -macType $val(mac) \
                         -ifqType $val(ifq) \
                         -ifqLen $val(ifqlen) \
                         -antType $val(ant) \
                         -propType $val(prop) \
                         -phyType $val(netif) \
                         -channelType $val(chan) \
                         -topoInstance $topo \
                         -agentTrace ON \
                         -routerTrace ON \
                         -macTrace OFF \
                         -movementTrace OFF

        for {set i 0} {$i < $val(nn) } {incr i} {
                set node_($i) [$ns_ node]
                $node_($i) random-motion 0           ;# enable random motion
        }
```

```
#
# Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes and motion
#
source "nomotion.tcl";
#source "motion.tcl";


#load traffic generation
source "cbr-gateway.tcl";
#source "cbr-no-gateway.tcl";

#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 150.0 "$node_($i) reset";
}
$ns_ at 150.0 "stop"
$ns_ at 150.01 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
}

puts "Starting Simulation..."
$ns_ run
```

dymoum.tcl

```
# ============================================================================
# Define options
# ============================================================================
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
```

```
set val(netif)          Phy/WirelessPhy            ;# network interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq
set val(nn)             6                          ;# number of mobilenodes
set val(rp)             DYMOUM                       ;# routing protocol

# ======================================================================
# Main Program
# ======================================================================



Agent/DYMOUM set debug_ false
Agent/DYMOUM set no_path_acc_ true
Agent/DYMOUM set reissue_rreq_ false
Agent/DYMOUM set s_bit_ true
Agent/DYMOUM set hello_ival_ 1

#
# Initialize Global Variables
#
set ns_        [new Simulator]
set tracefd     [open dymoum.tr w]
$ns_ trace-all $tracefd

# set up topography object
set topo       [new Topography]

$topo load_flatgrid 500 500

#
# Create God
#
set god_ [create-god $val(nn)]

#
```

```
#  Create the specified number of mobilenodes [$val(nn)] and "attach" them
#  to the channel.
#  Here two nodes are created : node(0) and node(1)

# configure node

        $ns_ node-config -adhocRouting $val(rp) \
                         -llType $val(ll) \
                         -macType $val(mac) \
                         -ifqType $val(ifq) \
                         -ifqLen $val(ifqlen) \
                         -antType $val(ant) \
                         -propType $val(prop) \
                         -phyType $val(netif) \
                         -channelType $val(chan) \
                         -topoInstance $topo \
                         -agentTrace ON \
                         -routerTrace ON \
                         -macTrace OFF \
                         -movementTrace OFF

        for {set i 0} {$i < $val(nn) } {incr i} {
                set node_($i) [$ns_ node]
                $node_($i) random-motion 0           ;# enable random motion
        }

#
# Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes and motion
#
source "nomotion.tcl";
#source "motion.tcl";


#load traffic generation
source "cbr-gateway.tcl";
#source "cbr-no-gateway.tcl";

#
# Tell nodes when the simulation ends
```

```
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 150.0 "$node_($i) reset";
}
$ns_ at 150.0 "stop"
$ns_ at 150.01 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
}

puts "Starting Simulation..."
$ns_ run
```

motion.tcl

```
#
# nodes: 6, pause: 2.00, max speed: 30.00, max x: 500.00, max y: 500.00
#
$node_(0) set X_ 255.024551737380
$node_(0) set Y_ 255.663679122470
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 304.793428722078
$node_(1) set Y_ 107.112554781933
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 225.594319268972
$node_(2) set Y_ 225.021185897385
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 244.501793374394
$node_(3) set Y_ 399.681021490519
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 43.370537930418
$node_(4) set Y_ 341.066449867054
$node_(4) set Z_ 0.000000000000
$god_ set-dist 0 1 1
```

```
$god_ set-dist 0 2 1
$god_ set-dist 0 3 1
$god_ set-dist 0 4 1
$god_ set-dist 0 5 1
$god_ set-dist 1 2 1
$god_ set-dist 1 3 2
$god_ set-dist 1 4 2
$god_ set-dist 1 5 2
$god_ set-dist 2 3 2
$god_ set-dist 2 4 2
$god_ set-dist 2 5 2
$god_ set-dist 3 4 1
$god_ set-dist 3 5 1
$god_ set-dist 4 5 2
$ns_   at   2.000000000000   "$node_(1)   setdest   390.917063646659   59.914875650381
4.424398162001"
$ns_   at   2.000000000000   "$node_(2)   setdest   396.306533816214   490.314414488477
11.396705273193"
$ns_   at   2.000000000000   "$node_(3)   setdest   285.728400440772   192.883536829561
11.910135500903"
$ns_   at   2.000000000000   "$node_(4)   setdest   265.191086721547   217.974666352088
19.437095984564"
$ns_ at 2.437801774268 "$god_ set-dist 4 5 1"
$ns_ at 3.771185536702 "$god_ set-dist 2 3 1"
$ns_ at 3.876944137774 "$god_ set-dist 2 4 1"
$ns_ at 5.426311295481 "$god_ set-dist 2 5 1"
$ns_ at 7.465070570999 "$god_ set-dist 1 3 1"
$ns_ at 8.956741356402 "$god_ set-dist 1 4 1"
$ns_   at   15.051575800092   "$node_(4)   setdest   265.191086721547   217.974666352088
0.000000000000"
$ns_   at   17.051575800092   "$node_(4)   setdest   291.473393357933   415.264043927873
2.952117231684"
$ns_ at 17.119425094238 "$god_ set-dist 0 2 2"
$ns_   at   19.704824628595   "$node_(3)   setdest   285.728400440772   192.883536829561
0.000000000000"
$ns_ at 20.991007587067 "$god_ set-dist 1 2 2"
$ns_   at   21.704824628595   "$node_(3)   setdest   214.235325125267   318.899678027055
21.611604554396"
$ns_   at   24.197021006568   "$node_(1)   setdest   390.917063646659   59.914875650381
0.000000000000"
$ns_ at 25.457396254537 "$god_ set-dist 1 3 2"
$ns_   at   26.197021006568   "$node_(1)   setdest   409.461995160229   353.733185696533
3.066470054403"
```

```
$ns_   at   28.408808203385   "$node_(3)   setdest   214.235325125267   318.899678027055
0.000000000000"

$ns_   at   30.408808203385   "$node_(3)   setdest   353.896839919161   323.430707159822
17.375671859863"

$ns_ at 34.898069587105 "$god_ set-dist 2 5 2"

$ns_ at 35.518499540761 "$god_ set-dist 1 3 1"

$ns_ at 36.351456627503 "$god_ set-dist 1 5 1"

$ns_ at 37.522520439050 "$god_ set-dist 0 3 2"

$ns_   at   38.450799127667   "$node_(3)   setdest   353.896839919161   323.430707159822
0.000000000000"

$ns_   at   38.496505411270   "$node_(2)   setdest   396.306533816214   490.314414488477
0.000000000000"

$ns_   at   40.450799127667   "$node_(3)   setdest   35.409626519587   360.685325232928
21.141867597473"

$ns_   at   40.496505411270   "$node_(2)   setdest   179.926164081470   405.270426021015
13.586233390015"

$ns_ at 40.792505909808 "$god_ set-dist 0 3 1"

$ns_ at 44.389270309221 "$god_ set-dist 1 3 2"

$ns_ at 49.511737445489 "$god_ set-dist 0 2 1"

$ns_ at 52.683639943494 "$god_ set-dist 2 5 1"

$ns_   at   55.617799572316   "$node_(3)   setdest   35.409626519588   360.685325232928
0.000000000000"

$ns_   at   57.608892955276   "$node_(2)   setdest   179.926164081470   405.270426021015
0.000000000000"

$ns_   at   57.617799572316   "$node_(3)   setdest   65.494221663754   305.687167468232
21.480616998851"

$ns_   at   59.608892955276   "$node_(2)   setdest   319.622314297930   7.236308535829
27.677461865105"

$ns_ at 60.415446965747 "$god_ set-dist 0 1 1"

$ns_   at   60.536186728070   "$node_(3)   setdest   65.494221663754   305.687167468232
0.000000000000"

$ns_ at 61.536372224548 "$god_ set-dist 1 5 2"

$ns_ at 62.451027721273 "$god_ set-dist 1 2 1"

$ns_   at   62.536186728070   "$node_(3)   setdest   15.603511069326   87.922840679053
17.729776433621"

$ns_ at 63.226246139132 "$god_ set-dist 4 5 2"

$ns_ at 65.234644722412 "$god_ set-dist 3 4 2"

$ns_ at 70.374434884327 "$god_ set-dist 2 4 2"

$ns_ at 70.480598260144 "$god_ set-dist 2 3 2"

$ns_ at 73.627025688650 "$god_ set-dist 0 2 2"

$ns_ at 74.769814748013 "$god_ set-dist 0 3 2"

$ns_ at 74.769814748013 "$god_ set-dist 1 3 3"

$ns_ at 74.769814748013 "$god_ set-dist 3 4 3"
```

```
$ns_   at   74.850054460417   "$node_(2)   setdest   319.622314297930   7.236308535829
0.000000000000"

$ns_   at   75.136815029484   "$node_(3)   setdest   15.603511069326   87.922840679053
0.000000000000"

$ns_   at   76.850054460417   "$node_(2)   setdest   462.593829426802   332.957237785474
25.180395719950"

$ns_   at   77.136815029484   "$node_(3)   setdest   219.260083769005   157.891175439899
18.039531657058"

$ns_ at 77.938060496467 "$god_ set-dist 0 3 1"

$ns_ at 77.938060496467 "$god_ set-dist 1 3 2"

$ns_ at 77.938060496467 "$god_ set-dist 3 4 2"

$ns_ at 78.848443218726 "$god_ set-dist 0 2 1"

$ns_ at 80.148029488825 "$god_ set-dist 2 5 2"

$ns_ at 83.735435160062 "$god_ set-dist 0 5 2"

$ns_ at 83.735435160062 "$god_ set-dist 1 5 3"

$ns_ at 83.735435160062 "$god_ set-dist 2 5 3"

$ns_ at 83.735435160062 "$god_ set-dist 4 5 3"

$ns_   at   84.471765670015   "$node_(4)   setdest   291.473393357933   415.264043927873
0.000000000000"

$ns_ at 84.775210499640 "$god_ set-dist 2 4 1"

$ns_ at 86.434983179217 "$god_ set-dist 1 3 1"

$ns_ at 86.434983179217 "$god_ set-dist 1 5 2"

$ns_   at   86.471765670015   "$node_(4)   setdest   19.606263627788   372.398580298504
9.468881965039"

$ns_   at   89.073964447730   "$node_(3)   setdest   219.260083769005   157.891175439899
0.000000000000"

$ns_   at   90.976820278015   "$node_(2)   setdest   462.593829426802   332.957237785474
0.000000000000"

$ns_   at   91.073964447730   "$node_(3)   setdest   87.599923070703   114.190222171966
22.311614118706"

$ns_ at 92.775159839112 "$god_ set-dist 1 3 2"

$ns_ at 92.775159839112 "$god_ set-dist 1 5 3"

$ns_   at   92.976820278015   "$node_(2)   setdest   428.135336854530   321.673945269589
23.645125999624"

$ns_   at   94.510277852256   "$node_(2)   setdest   428.135336854530   321.673945269589
0.000000000000"

$ns_   at   96.510277852256   "$node_(2)   setdest   314.674177428761   55.288234027092
5.645028754176"

$ns_   at   97.291504850840   "$node_(3)   setdest   87.599923070703   114.190222171966
0.000000000000"

$ns_ at 97.308357802621 "$god_ set-dist 2 4 2"

$ns_ at 97.849823217093 "$god_ set-dist 1 4 2"

$ns_   at   99.291504850840   "$node_(3)   setdest   200.511414838576   109.242018713229
19.594613186940"
```

```
$ns_ at 102.278315213062 "$god_ set-dist 0 5 1"

$ns_ at 102.278315213062 "$god_ set-dist 1 5 2"

$ns_ at 102.278315213062 "$god_ set-dist 2 5 2"

$ns_ at 102.278315213062 "$god_ set-dist 4 5 2"

$ns_ at 104.361499478971 "$god_ set-dist 4 5 1"

$ns_ at 105.059409685905 "$node_(3) setdest 200.511414838576 109.242018713229
0.000000000000"

$ns_ at 106.309058545665 "$god_ set-dist 2 5 1"

$ns_ at 106.954762542827 "$god_ set-dist 1 5 1"

$ns_ at 107.059409685905 "$node_(3) setdest 150.876924950257 73.770569601161
28.450314523109"

$ns_ at 109.203730744826 "$node_(3) setdest 150.876924950257 73.770569601161
0.000000000000"

$ns_ at 111.203730744826 "$node_(3) setdest 404.623087416776 24.998264317317
15.129912101436"

$ns_ at 111.524255060302 "$god_ set-dist 3 5 2"

$ns_ at 112.456634393048 "$god_ set-dist 0 4 2"

$ns_ at 112.456634393048 "$god_ set-dist 3 4 3"

$ns_ at 114.269682824541 "$god_ set-dist 0 4 16777215"

$ns_ at 114.269682824541 "$god_ set-dist 1 4 16777215"

$ns_ at 114.269682824541 "$god_ set-dist 2 4 16777215"

$ns_ at 114.269682824541 "$god_ set-dist 3 4 16777215"

$ns_ at 114.269682824541 "$god_ set-dist 4 5 16777215"

$ns_ at 114.484235694205 "$god_ set-dist 2 3 1"

$ns_ at 115.538100835456 "$node_(4) setdest 19.606263627788 372.398580298504
0.000000000000"

$ns_ at 117.538100835456 "$node_(4) setdest 134.975300537157 474.219194994679
29.943024674000"

$ns_ at 118.506721705148 "$god_ set-dist 0 4 2"

$ns_ at 118.506721705148 "$god_ set-dist 1 4 2"

$ns_ at 118.506721705148 "$god_ set-dist 2 4 2"

$ns_ at 118.506721705148 "$god_ set-dist 3 4 3"

$ns_ at 118.506721705148 "$god_ set-dist 4 5 1"

$ns_ at 121.041669398550 "$god_ set-dist 2 4 3"

$ns_ at 121.041669398550 "$god_ set-dist 2 5 2"

$ns_ at 122.204147561804 "$node_(1) setdest 409.461995160229 353.733185696533
0.000000000000"

$ns_ at 122.677020358773 "$node_(4) setdest 134.975300537158 474.219194994679
0.000000000000"

$ns_ at 124.204147561804 "$node_(1) setdest 353.070530094828 92.719757646720
0.205615603544"

$ns_ at 124.677020358773 "$node_(4) setdest 259.683663960128 334.695960771469
13.169675762600"
```

```
$ns_ at 125.768817973987 "$god_ set-dist 0 4 1"

$ns_ at 125.768817973987 "$god_ set-dist 2 4 2"

$ns_ at 125.768817973987 "$god_ set-dist 3 4 2"

$ns_ at 127.608590343207 "$god_ set-dist 0 3 2"

$ns_ at 127.608590343207 "$god_ set-dist 3 4 3"

$ns_ at 127.608590343207 "$god_ set-dist 3 5 3"

$ns_   at   128.281879620442   "$node_(3)   setdest   404.623087416776   24.998264317317
0.000000000000"

$ns_ at 128.929307206580 "$god_ set-dist 1 4 1"

$ns_   at   130.281879620442   "$node_(3)   setdest   443.580950089925   36.471883902827
5.227473966618"

$ns_ at 137.022741646811 "$god_ set-dist 1 2 2"

$ns_ at 137.022741646811 "$god_ set-dist 1 3 3"

$ns_   at   138.050890328136   "$node_(3)   setdest   443.580950089925   36.471883902827
0.000000000000"

$ns_ at 138.223609914407 "$god_ set-dist 2 4 1"

$ns_ at 138.223609914407 "$god_ set-dist 3 4 2"

$ns_   at   138.886436123493   "$node_(4)   setdest   259.683663960128   334.695960771469
0.000000000000"

#

# Destination Unreachables: 5

#

# Route Changes: 81

#

# Link Changes: 53

#

# Node | Route Changes | Link Changes

#    0 |             19 |           17

#    1 |             26 |           17

#    2 |             27 |           21

#    3 |             28 |           16

#    4 |             34 |           16

#
```

nomotion.tcl

```
$node_(0) set X_ 255.0
```

```
$node_(0) set Y_ 255.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 390.0
$node_(1) set Y_ 385.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 100.0
$node_(2) set Y_ 2.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 190.0
$node_(3) set Y_ 385.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 205.0
$node_(4) set Y_ 402.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 390.0
$node_(5) set Y_ 185.0
$node_(5) set Z_ 0.0
```

cbr-no-gateway.tcl

```
#
# nodes: 10, max conn: 11, send rate: 0.25, seed: 1.0
#
#
# 1 connecting to 2 at time 2.5568388786897245
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
```

```
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.25
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 50000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.5568388786897245 "$cbr_(0) start"
#
# 4 connecting to 5 at time 56.333118917575632
#
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 512
$cbr_(1) set interval_ 0.25
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 50000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 56.333118917575632 "$cbr_(1) start"

#
# 3 connecting to 5 at time 26.333118917575632
#
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(2)
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 512
$cbr_(2) set interval_ 0.25
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 50000
$cbr_(2) attach-agent $udp_(2)
$ns_ connect $udp_(2) $null_(2)
$ns_ at 26.333118917575632 "$cbr_(2) start"
```

```
#
# 0 connecting to 2 at time 36.333118917575632
#
set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(3)
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 512
$cbr_(3) set interval_ 0.25
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 50000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $null_(3)
$ns_ at 36.333118917575632 "$cbr_(3) start"



#
# 5 connecting to 0 at time 46.333118917575632
#
set udp_(4) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(4)
set null_(4) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(4)
set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 512
$cbr_(4) set interval_ 0.25
$cbr_(4) set random_ 1
$cbr_(4) set maxpkts_ 50000
$cbr_(4) attach-agent $udp_(4)
$ns_ connect $udp_(4) $null_(4)
$ns_ at 46.333118917575632 "$cbr_(4) start"
```

cbr-gateway.tcl

```
#
# nodes: 10, max conn: 11, send rate: 0.25, seed: 1.0
#
#
# 1 connecting to 0 at time 2.5568388786897245
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.25
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 50000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.5568388786897245 "$cbr_(0) start"
#
# 4 connecting to 0 at time 56.333118917575632
#
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 512
$cbr_(1) set interval_ 0.25
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 50000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 56.333118917575632 "$cbr_(1) start"

#
# 3 connecting to 0 at time 26.333118917575632
#
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(2)
```

```
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 512
$cbr_(2) set interval_ 0.25
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 50000
$cbr_(2) attach-agent $udp_(2)
$ns_ connect $udp_(2) $null_(2)
$ns_ at 26.333118917575632 "$cbr_(2) start"

#
# 2 connecting to 0 at time 36.333118917575632
#
set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(3)
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 512
$cbr_(3) set interval_ 0.25
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 50000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $null_(3)
$ns_ at 36.333118917575632 "$cbr_(3) start"
```

statistics.sh

```
#!/bin/sh



#usage example
# ./statistics.sh ../results/tracefile.tr



#measure loss
```

```
grep   ' cbr ' $1 |  awk -f measure-loss.awk

#measure delay
echo
echo "delay:"
grep   ' cbr ' $1 | awk -f measure-delay.awk   > $1.delay
echo

#measure jitter
echo
echo "jitter:"
grep   ' cbr ' $1 | awk -f measure-jitter.awk > $1.jitter
```

measure-jitter.awk

```
#This program is used to calculate the jitters for CBR
# jitter =((recvtime(j)-sendtime(j))-(recvtime(i)-sendtime(i)))/(j-i),  j > i

BEGIN {
# Initialization
     highest_packet_id = 0;
}
{
   action = $1;
   time = $2;
   seq_no = $6;
   packet_id = $6;




   if ( packet_id > highest_packet_id ) {
            highest_packet_id = packet_id;
        }
```

```
#Record the transmission time
   if ( start_time[packet_id] == 0 )  {
           start_time[packet_id] = time;
   }

#Record the receiving time for CBR
   if ( action != "D" ) {
      if ( action == "r" ) {
                       # Record the sequence number
                       pkt_seqno[packet_id] = seq_no;
             end_time[packet_id] = time;
      }
    } else {
      end_time[packet_id] = -1;
   }
}
END {
        last_seqno = 0;
        last_delay = 0;
        seqno_diff = 0;

    for ( packet_id = 0; packet_id <= highest_packet_id; packet_id++ ) {
       start = start_time[packet_id];
       end = end_time[packet_id];
       packet_duration = end - start;

       if ( start < end ) {
               seqno_diff = pkt_seqno[packet_id] - last_seqno;
               delay_diff = packet_duration - last_delay;
               if (seqno_diff == 0) {
                       jitter =0;
               } else {
                       jitter = delay_diff/seqno_diff;
               }

                             printf("%f %f\n", start, jitter);

               last_seqno = pkt_seqno[packet_id];
               last_delay = packet_duration;
```

```
       }
    }
```

measure-delay.awk

```
#This program is used to calculate the end-to-end delay

BEGIN {
     highest_packet_id = 0;
}
{
        action = $1;
        time = $2;
        packet_id = $6;

   if ( packet_id > highest_packet_id )
          highest_packet_id = packet_id;

   if ( start_time[packet_id] == 0 )
         start_time[packet_id] = time;

   if ( action != "D" ) {
      if ( action == "r" ) {
         end_time[packet_id] = time;
      }
   } else {
      end_time[packet_id] = -1;
   }
}
END {
    for ( packet_id = 0; packet_id <= highest_packet_id; packet_id++ ) {
       start = start_time[packet_id];
       end = end_time[packet_id];
       packet_duration = end - start;

       if ( start < end )
              printf("%f %f\n",  start, packet_duration);
```

```
   }
}
```

measure-loss.awk

```
BEGIN {
# Initialization. Set two variables. fsDrops: packets drop. numFs: packets sent
        numDrops = 0;
               highest_packet_id = 0;
}
{
   action = $1;
   packet_id = $6;

   if ( packet_id > highest_packet_id )
         highest_packet_id = packet_id;

       if (action == "D")
                      numDrops++;
}
END {
        printf("Packets sent:%d lost:%d\n", highest_packet_id, numDrops);
}
```